Lattice Radiant Software 2024.2 Help



December 20, 2024

Copyright

Copyright © 2024 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. QuestaSim is a trademark or registered trademark 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 **Getting Started with Lattice Radiant Software** **9**
Tutorials **9**
User Guides **10**
Getting Help **10**
Debugging the Software License **14**
Copyright, Trademarks, and Disclaimers **15**

Chapter 2 **Managing Projects** **17**
Running the Radiant Software **18**
Creating a New Project **19**
Modifying a Project **21**
Importing Lattice Diamond Projects **22**
Targeting a Device **22**
Viewing Project Properties **23**
Saving Project Files **23**
Managing Project Sources **25**
Working with Implementations **41**
Using Strategies **43**
Analyzing a Design **49**
Running Processes **63**
Clearing Tool Memory **67**
Setting Options for Synthesis and Simulation **68**
Working with Run Manager **78**
Finding Results **84**
Viewing Logs and Reports **85**
Setting Tool and Environment Options **92**
Block-Based Design - Using Macro Blocks **94**

Chapter 3 Entering the Design 111
HDL Design Entry **112**
Packaging IP Using IP Packager **139**
Designing with Soft IP, Modules, and PMI **141**

Chapter 4 Securing the Design 152
HDL File Encryption Steps **154**
Synthesizing Encrypted IP **158**
Cross-probing in Encrypted Design **159**
Secure Objects in the Design **159**
Securing the Bitstream **170**

Chapter 5 Simulating the Design 180
Simulation in the Radiant Software **180**
Timing Simulation **186**
Third-Party Simulators **187**

Chapter 6 Applying Design Constraints 196
Unified Constraints Flow **196**
Understanding Implications of Radiant Software Constraint Flows **199**
Applying Differential Signal Design Constraints **201**
Timing and Physical Constraints **201**
Checking Constraint Coverage **202**
Migrating from Former Lattice Diamond Preferences **204**
Migrating Pin Assignments **207**
Using Radiant Software Pre-Synthesis Constraints **211**
Constraint Propagation **216**
Applying Radiant Software Physical Constraints **283**
Constraint Conflict Resolution **293**
Checking Design Rules **297**
Analyzing SSO **303**
Exporting Pin Files **307**

Chapter 7 Implementing the Design 311
Synthesizing the Design **311**
Mapping **320**
Place and Route **327**
Bit Generation **341**

Chapter 8 Analyzing Static Timing 350
Options for Timing Analysis Reports **351**
Running Post-Synthesis Constraint Scenario Timing Reports **351**
Reading Timing Analysis Reports **352**
Understanding I/O Timing Analysis **355**
Using Timing Analyzer **361**

Using Standalone Timing Analyzer **364**

Chapter 9 Analyzing Power Consumption 369
Starting Power Calculator from the Radiant Software **370**
Starting Power Calculator as a Stand-Alone Tool **371**
Running Power Calculator from the Tcl Console **372**
Power Analysis Design Flow **372**
Inputs **373**
Outputs **374**
Static and Dynamic Power Consumption **374**
Activity Factor Calculation **374**
Enable Factor Calculation **375**
Power Calculator Window Features **375**
Working with Power Calculator Files **386**
Entering Data **393**
Reverting to Calculation Mode **397**
Changing the Global Default Activity Factor **397**
Importing a Value Change Dump (.vcd) File **398**
Changing the Global Default Frequency Setting **398**
Estimating Resource Usage **400**
Estimating Routing Resource Usage **400**
Running Immediate Calculation **401**
Controlling Operating Temperature **402**
Viewing and Printing Results **407**

Chapter 10 Using IBIS Model Files 411
IBIS Overview **411**
Lattice Semiconductor IBIS Model Files **412**
Generating Design-Specific IBIS Models in the Radiant Software **412**
Copying General-Purpose IBIS Models from Radiant Installation Folder **415**
Downloading General-Purpose IBIS Models from the Lattice Website **417**

Chapter 11 Programming the FPGA 418
About the Programmer Window **419**
File Formats **420**
SPI Flash Support **420**
Using the Radiant Programmer **432**
Programmer Options **482**
Programming and Configuring iCE40 Devices with Programmer **501**
Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer **507**
Deploying the Design with the Deployment Tool **527**
Debugging SVF, STAPL, and VME Files **583**
Download Debugger Options **593**

Using Programming File Utility **598**
Programming File Utility Options **603**

Chapter 12 Testing and Debugging On-Chip 607
About Reveal Logic Analysis **607**
Creating Reveal Modules **616**
Using JTAGhub **650**
Performing Logic Analysis **660**

Chapter 13 Applying Engineering Change Orders 692
Editing sysIO Settings in ECO Editor **693**
Setting Memory Initialization Values in ECO Editor **693**
Running Design Rule Check **696**

Chapter 14 Strategy Reference Guide 697
Constraint Propagation Options **703**
Synplify Pro Options **703**
LSE Options **709**
Post-Synthesis Options **717**
Post-Synthesis Timing Analysis Options **718**
Map Design Options **719**
Map Timing Analysis Options **721**
Place & Route Design Options **722**
Place & Route Timing Analysis Options **725**
IO Timing Analysis Options **727**
Timing Simulation Options **727**
Bitstream Options **728**

Chapter 15 Constraints Reference Guide 731
HDL Attributes **732**
Lattice Synthesis Engine Constraints **750**

Chapter 16 Lattice Module Reference Guide 821
Finding Modules in This Guide **821**

Chapter 17 FPGA Libraries Reference Guide 824
Primitive Library - LAV-AT (Lattice Avant) **826**
Primitive Library - LFCPNX (CertusPro-NX) / UT24CP (CertusPro-NX-RT) **835**
Primitive Library - LFD2NX (Certus-NX) / UT24C (Certus-NX-RT) **841**
Primitive Library - LFMXO5 (MachXO5-NX) **847**
Primitive Library - LIFCL (CrossLink-NX) **853**
Primitive Library - iCE40UP (iCE40 UltraPlus) **859**
Alphanumeric Primitives List **863**
DSP Inference Guide **1468**
Synthesis Attributes **1470**

Chapter 18 Command Line Reference Guide 1471
Command Line Program Overview **1471**
Command Line Basics **1473**
Command Line Tool Usage **1478**

Chapter 19 Tcl Command Reference Guide 1556
Launching the Tcl Console **1558**
Running Radiant Tcl **1559**
Accessing Command Help and Command Options in the Tcl Console **1572**
Creating and Running Custom Tcl Scripts **1573**
Running Tcl Scripts When Launching the Radiant Software **1576**
Radiant Software Tool Tcl Command Syntax **1577**
Radiantc TCL Commands **1697**

Revision History 1725

Getting Started with Lattice Radiant Software

Lattice Radiant™ software is the complete design environment for Lattice Semiconductor Field Programmable Gate Arrays (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 help you experiment further, example projects are available. These examples are designed to illustrate different aspects of designing with Radiant software. To see the examples, click the (Open Example) button, or choose **File > Open > Design Example**. In the Open Example dialog box, choose an example you wish to open, and click **OK.**

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

See Also

Lattice on the Web

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

Tutorials

To quickly get some hands-on experience, try the following:

- Lattice Radiant Block-Based Design Tutorial
- Lattice Radiant Post-Synthesis Reveal Debug Flow Tutorial

- Lattice Radiant Tutorial with CrossLink-NX (LIFCL)
- Lattice Power Calculator Tutorial

User Guides

To learn more about Radiant software, see the following User Guides and Reference Manuals:

- Lattice Radiant 2024.2 Release Notes
- Lattice Radiant Software 2024.2 User Guide
- Lattice Radiant Software Guide for Lattice Diamond Users
- Migrating iCEcube2 iCE40 UltraPlus Designs to Lattice Radiant Software
- Lattice Radiant Software IP User Guide
- Programming Tools User Guide for Radiant Software
- Reveal User Guide for Radiant Software
- Reveal Troubleshooting Guide for Lattice Radiant Software
- Lattice Radiant Software 2024.2 Help (HTML)
- Lattice Radiant Software 2024.2 Help (PDF)
- Lattice Radiant Software 2024.2 Installation Guide for Windows
- Lattice Radiant Software 2024.2 Installation Guide for Linux/Ubuntu
- QuestaSim User Manual
- QuestaSim GUI Reference Manual
- QuestaSim Command Reference Manual
- QuestaSim Tutorial
- Synopsys Synplify Pro for Lattice User Guide
- Synopsys Synplify Pro for Lattice Reference Manual
- Synopsys Synplify Pro for Lattice Language Support Reference Manual

Getting Help

For almost all questions, the place to start is this Help. It describes the FPGA design flow using the Radiant software, the libraries of logic design elements, and the details of the Radiant 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 Radiant Software > Accessories > Radiant Software Help**.
- In the main window, choose **Help > Lattice Radiant Software Help**.
- To go directly to the Help for the tool that you're using, choose **Help >** <*Tool Name*> **Help**.

JavaScript must be enabled in your default browser. If you are having trouble opening the Help, see "Troubleshooting the Help" on page 12.

Using the Help

The Help has several features to help you find information.

Contents The Contents pane 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 The Search function locates all topics that contain specific text. This can be the fastest way to find information once you are familiar with the contents of the Help.

Just start typing in the Search field. The Search page opens after three 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 are on the Search page and want to go back to the topic you were looking at, click the ✖ next to the Search field.

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 having trouble with the Help, check for the following situations:

Active Content or Scripts Are Blocked Opening the 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 Radiant 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 not always 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.

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.

Debugging the Software License

The License Debug tool can help you debug license problems you may encounter when setting up a license for the Radiant software.

If the Radiant software closes after starting due to a license issue, the License Debug tool will automatically launch. The missing license feature which caused the Radiant software to exit will be displayed in a text box to help with debugging.

You can also launch the tool manually using the **Help > License Debug** command.

Using the License Information Tab

1. Click **Help > License Debug** to launch the License Debug tool. The License Information tab is displayed by default. The License File text box displays one or more license file(s) or license server(s), separated by semicolon(s).
2. Click the ... button to browse to the location of the license file (license.dat).
3. Click **Start** to display the feature list based on the specified license file(s). The **Feature List** box displays Feature(s) and file **Location**.

   The **Local System Information** box at the bottom displays the Hostname and Network Feature Card (NIC) ID. If multiple cards are available, all the NIC IDs are displayed.
4. If you need to administer the license server, click **LmTools…** to launch LMTOOLS Utility by Flexera.

   The LMTOOLS Utility is a graphical user interface that enables you to administer the license server.

   Some of the functions LMTOOLS performs include:

   - Starting, stopping, and configuring license servers.
   - Getting system information, including HostID's.
   - Getting server status.

   LMTOOLS has two modes in which to configure a license server:

   - Configuration using a license file.
   - Configuration using services.

   You must run LMTOOLS as an administrator. If you do not run this executable as an administrator, the User Account Control (UAC) dialog will be displayed when it is launched, if the UAC prompt is not disabled on the system.

   For more information about LMTOOLS, please refer to the *FlexNet Publisher License Administration Guide* at https://www.revenera.com.

Using the License Debug Tab

1. Choose **Help > License Debug** to launch the License Debug tool.

2. Click the License Debug tab.
3. In the **Search License Feature** box, type in the license feature text string you wish to find.

   Click the **Match Whole String** box if you wish to match the entire string. If you uncheck the Match Whole String box, you can search for partial string.
4. Click the ... button to browse to the location of the license file (license.dat)

   or
5. Click the **Use Default License Search Path** button.
6. Click **Start** to display the feature list based on the specified license file(s). The Feature List box displays Feature(s) and file Location.

To know more about licensing, visit the Software Licensing page.

Copyright, Trademarks, and Disclaimers

Copyright Copyright © 2024 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. QuestaSim 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.

Managing Projects

You can create a project using the Lattice Radiant software with the New Project wizard. Launch the New Project Wizard by clicking the **New Project** button, or choose **File > New > Project**. The Radiant software creates a folder with a project file (.rdf) containing basic information about the project, such as the Radiant version number; the beginning of a physical design constraints (.pdc) file, 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. The Radiant software 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 contains a source folder, which contains project source files.

The project folder also contains an implementation folder. Implementation folders contain 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. 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 Radiant software graphical user interface or through scripts. The Radiant software comes with command-line and Tcl commands for many of its functions.

See Also

- “Running the Radiant Software” on page 18
- “Creating a New Project” on page 19

- "Modifying a Project" on page 21
- "Importing Lattice Diamond Projects" on page 22
- "Targeting a Device" on page 22
- "Viewing Project Properties" on page 23
- "Saving Project Files" on page 23
- "Managing Project Sources" on page 25
- "Working with Implementations" on page 41
- "Using Strategies" on page 43
- "Analyzing a Design" on page 49
- "Running Processes" on page 63
- "Clearing Tool Memory" on page 67
- "Setting Options for Synthesis and Simulation" on page 68
- "Finding Results" on page 84
- "Viewing Logs and Reports" on page 85

Running the Radiant Software

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

To run the Radiant software main window:

- From your Windows desktop, choose **Start > All Programs > Lattice Radiant Software <version number> > Radiant Software**.

  Notes

  - **Lattice Radiant Software** is the default Programs folder name when you install the Radiant software. Change this name accordingly if you have chosen another folder name during installation.
  - You can run current and previous versions of Radiant by selecting the **Lattice Radiant Software <version number>** from the Start menu.

- From your Linux platform shell window or C-shell window, execute the following (case-sensitive):

  ```
  <install_path>/bin/lin64/radiant
  ```

See Also "Managing Projects" on page 17

Pin the Radiant Software to Start Menu or Taskbar

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

To pin the Radiant software to the Start menu or Taskbar:

1. Choose the Windows **Start** menu and find the **Lattice Software** icon.
2. Right-click the **Lattice Software** icon.
3. Click **Pin to Start Menu** or **Pin to Taskbar**.

Note

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

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

See Also ▶"Managing Projects" on page 17

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 Radiant software main window, click the **New Project** button, or 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 Radiant software 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 (_). Spaces are allowed.

- 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 41.
- 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**.

   Click **Add Source**. You can import HDL files at this time.

   a. 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.

   b. Repeat the above to add more files.

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

   d. To create empty Lattice Design Constraint (.ldc) file and Post-Synthesis Constraint File (.pdc) files that can be edited at a later time, select **Create empty constraint files.** Refer to "Timing and Physical Constraints" on page 201 for more information about constraint files.

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

5. In the Select Device dialog box, select a device family and a specific device within that family.

   a. Choose the options you want for that device.

   b. Click **Next**.

6. In the Select Synthesis Tool dialog box, the default synthesis tool is Synplify Pro. This choice can be changed at any time. See "Selecting a Synthesis Tool" on page 70 for more information. When you finish, click **Next**.

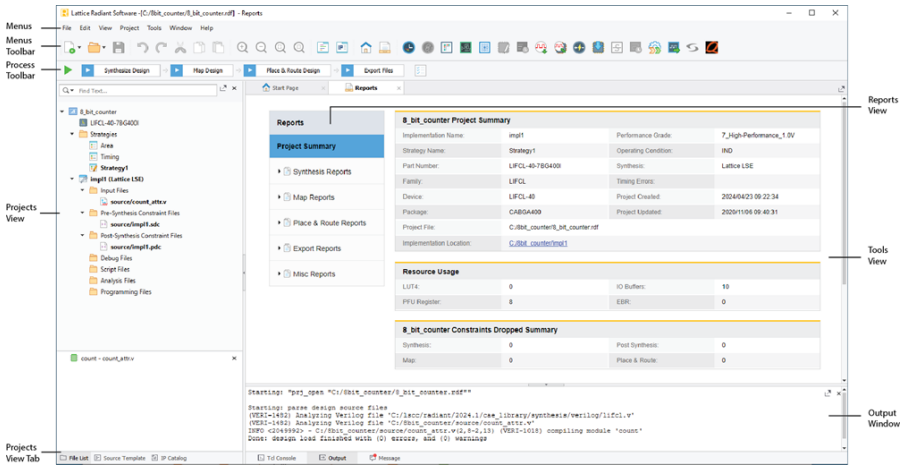
7. In the Project Information dialog box, make sure the project settings are correct and then click **Finish**.

   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.

Once a project has been created, the Radiant software workspace opens in the main window and displays the following:

Figure 1: Radiant Software Main Window Workspace



See Also ▶"Managing Projects" on page 17

Modifying a Project

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

To modify a project:

1. Click on the File List tab at the bottom left side of the screen to display the File List view.
2. 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.

   - 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.
   - You can also edit properties of the project in the Project Properties dialog box. You can set the top-level unit, specify the VHSIC Hardware Description Language (VHDL) Library name, and Verilog Include Search path in the Project Properties dialog box.

See Also

- "Managing Projects" on page 17

Importing Lattice Diamond Projects

Design projects created in Lattice Diamond software can be imported into the Radiant software using the Import Diamond Project wizard. Imported Lattice Diamond projects are targeted to devices supported in the Radiant software. Lattice Diamond design preferences will be converted into Radiant software design constraints.

To import a Diamond software project into the Radiant software:

1. In the Radiant software, choose **File > Open > Import Diamond Project**.
2. In the **Select Diamond Project** dialog box, browse to the Diamond Project file (.ldf) and select it.

   The Import Diamond Project wizard appears. Click **Next.**
3. In the Select Device dialog box, choose Device Family, Device, Operating Condition, Package, Performance Grade, and Part Number. Click **Next.**
4. In the Project Name dialog box, specify the Name and Location of your project. Lists of files to be imported, and files not to be imported, appear.
5. Click **Copy to Radiant project folder** if you wish to copy the imported files into your new Radiant software project.
6. Click **Finish**.

See Also

- “Managing Projects” on page 17

Targeting a Device

The Radiant software lets you re-target 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.
3. When you finish, click **OK**.

   The specified target device appears in the File List view.

See Also

- “Managing Projects” on page 17

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 name, category, and location of the selected file. You can enter values for some of the selected sections of the project in the Value field.

See Also

- "Managing Projects" on page 17

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. The Radiant software offers three save commands for this:

- **File > Save**: Saves the currently active item.
- **File > Save All**: Saves all the content changes.
- **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.
- Click the **Save** icon ().

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 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 24 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. 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.

Archiving a Project

The Radiant software 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. Click **Save**.

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

To reload the archived project:

1. In the Radiant software 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 the Radiant software.

See Also

- "Managing Projects" on page 17
- "Copying a Project" on page 23

Managing Project Sources

The Radiant software combines design sources into different categories and lists them in the File List view, as well as any folders associated with them. The source files are classified and listed under these folders, which include Strategies, Input Files, Synthesis Constraint Files, Debug Files, Script Files, Analysis Files, and Programming Files.

The following files can be found in their respective folder, as shown in Figure 1

Table 1: Source File Locations

Files	Folder
HDL/IP Module	Input Files
Logic Constraint	Post-Synthesis Constraint Files
Reveal Project	Debug Files
Power Calculator/Timing Constraints	Analysis Files

You can also see implementations listed in the File List view. For details about implementations, see "Working with Implementations" on page 41.

File List View The File List view of the Radiant software 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.

Table 2: Files Listed in the File List View

File Type	File Extension
Project Title	None
Target Device	None
Predefined Strategy (Area, I/O Assistant, Quick, and Timing)	.sty
Strategy1 (that can be customized)	.sty
Implementation	None
Verilog Files	.v, .veri, .ver, .vo, .h
Structural Verilog	.vm
SystemVerilog	.sv

Table 2: Files Listed in the File List View

File Type	File Extension
IP Module Config Files	.ipx
Undefined or incorrect	Any source reference
Synthesis Constraint Files	.sdc, .fdc, .ldc
Reveal Project File	.rvl
Simulation Project File	.spf
Reveal Analyzer Files	.rva
Power Calculator Files	.pcf
Programmer Project File	.xcf

Note for Linux

In the Linux version of the Radiant software, 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.

Figure 2: File List



See Also

- “Managing Projects” on page 17
- “Running Processes” on page 63

Creating a New Source File

You can create a new source from within the Radiant software or external to the Radiant software.

To create a new source file:

1. In the Radiant software 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 under Categories and Source Files.
   a. Fill in the file name.
   b. Browse to choose a location for the file.
   c. Check the **Add to Implementation** option.
   d. Choose the Implementation you want the file to be part of.
3. Click **New**.

   The Radiant software starts an editor that you can use to enter the information for your new source file.
4. In the editor, create a source file.

Note

- File names for Radiant software 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 the Radiant software, 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

- “Managing Projects” on page 17
- “Analyzing a Design” on page 49

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 the Radiant software. Or, with a project opened in the Radiant software, 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. In the **Files of type** drop down, choose the type of file you wish to import. **If All Files (*.*)** is selected, the **Force File Type** drop down is activated in the lower left.

   If you wish to specify the language type of the file to either Verilog or VHDL, select either **Verilog** or **VHDL** in the drop down. **Default** will not force file type.

4. 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.

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

Tip

- Radiant software project creation supports mixed language source files, allowing you to freely mix Verilog HDL and VHDL design. The Radiant 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

- "Managing Projects" on page 17
- "Analyzing a Design" on page 49
- "Setting the Top-Level Unit for Your Project" on page 38

Adding Reveal Debug Information

Reveal Inserter manages the addition of Reveal debug information into the Radiant software 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 a .rvl file.

To import a Reveal project file (.rvl):

1. In the Radiant software 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 607.

See Also

▶ "Managing Projects" on page 17

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. The Radiant software will run all these test files in a hierarchical order during the simulation process.

To import a test stimulus file:

1. In the Radiant software, choose **Tools > Simulation Wizard**.

   The Preparing the Simulator Interface page opens.
2. Click **Next**.
3. In the Simulator Project Name and Stage page:
   a. Enter a name for your simulation project in the Project name field.
   b. Browse to find a desired location for your project.
   c. Under Simulator, choose either QuestaSim or Active-HDL .
   d. Under Process Stage, all the available process stages of the FPGA implementation strategy are automatically displayed. Choose the stage in which to run simulation.
4. Click **Next**.
5. In the Add and Reorder Source page, browse for your desired Verilog test fixture (.v) or VHDL test bench (.vhd) and select it. Check the **Automatically set simulation compilation file order** option if needed.
6. Click **Next**.
7. In the Parse HDL Files for Simulation page, the HDL files are listed. The top level module is also indicated.
8. Click **Next**.
9. In the Summary page, all your simulation project information is listed, including simulator name, project name, project location, simulation stage, simulation files, and simulation libraries.

   Check the **Run Simulation** option if you want to run the simulation right away.

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

Note

The simulation flow in the Radiant software supports 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 178 for more details on how to use the Simulation Wizard.

See Also

- "Managing Projects" on page 17

Using Revision Control for Radiant Projects

Radiant supports revision control for your projects. When changes are logged into the revision control system, you can get all the change logs and switch to previous milestones if necessary. Compile times can be reduced when using revision control with Radiant design. The compilation will only start when inputs are changed.

Recommended Revision Control System

Radiant recommends the following revision control systems:

- GIT
- SVN
- Perforce

Radiant Revision Control Strategies

You can select different revision control systems to work with Radiant, the software does not directly integrate a specific revision control into the system. Radiant recommends two strategies:

- **Minimum Files Strategy (Radiant Recommendation)** -- The revision control system only manages the necessary files of the project. Other files -- including some intermediate and configuration files, are only used in your project and will not be submitted to the version management system.

The following files are recommended to commit to revision control systems:

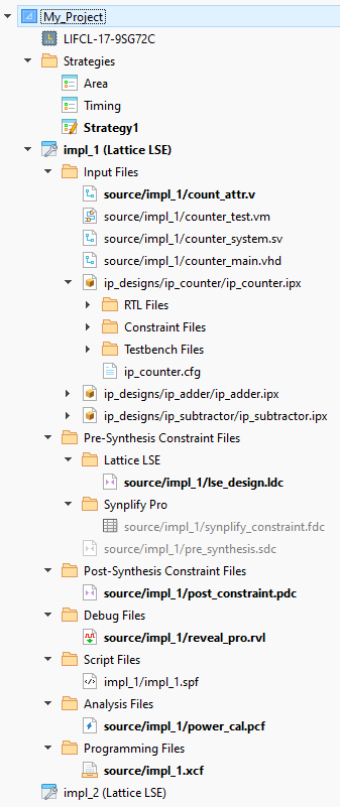
- .rdf project file and .sty strategy file
- All source files under source directory
- All files under ip_directory

- **Complete Strategy** -- You can also add some intermediate files on the project to the revision control system.

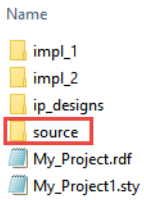
  For this strategy, the files under implementation can be committed to the revision control system. You can select files to submit to the system under each implementation, which mainly includes intermediate and analysis files.

It is recommended to use the following strategy directory structure to manage source code and designs:

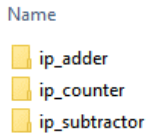
- Radiant design can use one source directory to put all source or design files. The software can automatically identify various types of files and display them in categories in the **File List View**.



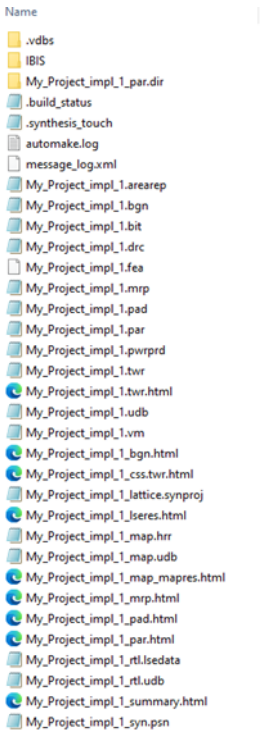
- Use a source directory to place all source and design files for every implementation. Under the source folder, each implementation has its own file directory (e.g. impl_1, impl_2).



- Create an IP directory (e.g. ip_designs) under the source directory of the project to store all IP designs of Radiant. Each IP design has a subdirectory under the source directory to store IP source code and design files.



- Use one implementation directory (e.g. impl_1, impl_2) to compile the relative implementation of the project. All intermediate files will be generated under this folder (.udb, status, report, and other intermediate type files are all here).



Radiant File Types

The Radiant project directory contains the following file types:

- .rdf file -- Radiant project file
- .sty file -- Radiant strategy file

Common source or design file types under the source directory:

- .v file -- Verilog file
- .vhdl file -- VHDL file
- .sv file -- SystemVerilog file
- .ldc file -- LSE Design Constraints file
- .pdc file -- Post-Synthesis Constraint file
- .pcf file -- Power Calculator file
- .sdc file -- Pre-Synthesis Constraint file
- .rvl file -- Reveal Project file
- .rva file -- Reveal Analyzer file
- .vm file -- Structural Verilog file
- .fdc file -- Synplify Pro Constraint file
- IP_ folder -- One IP instance file under the folder

Other intermedia file types:

- .ini file -- Initialization file
- .log file -- Log file
- .build_status file -- Status file
- .xml file -- XML configuration file
- .html file -- HTML file
- .ibs file -- Input/Output Buffer Information Specification file
- .udb file -- User design Database file
- .bit file -- Bitstream
- synthesis.log -- LSE synthesis report
- .mrp -- Radiant map report
- .twr -- Radiant timing report

Note:

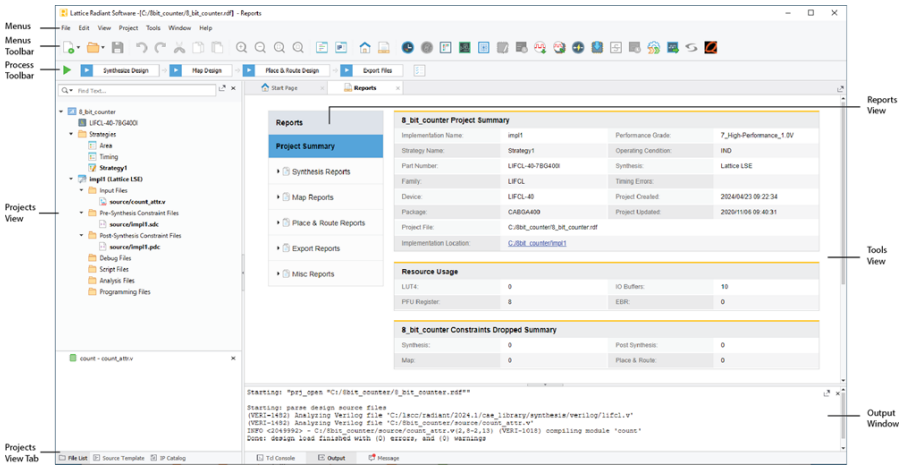
Intermedia files might be re-generated if you rerun the flow.

Radiant Revision Control Workflow

The following diagrams show the revision control workflow of Radiant.

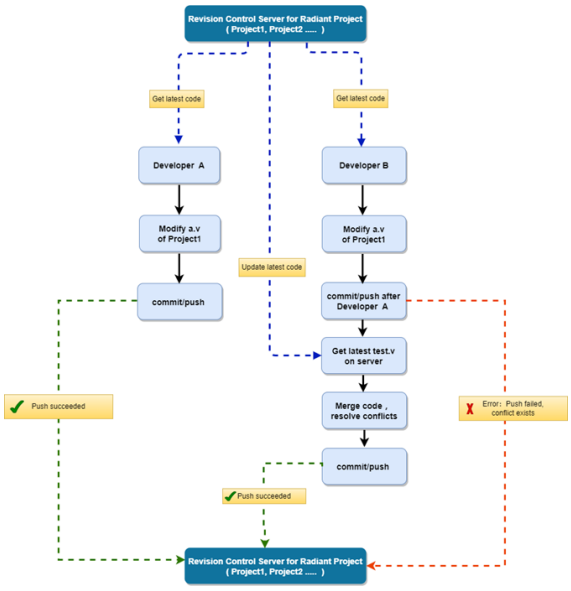
Revision Control Normal Workflow

You can get the latest code from the Radiant version server and develop in your own environment and working directory. Later on, you can submit the modified code or new design to the radiant version server as needed.



Multiple developers working on the same file

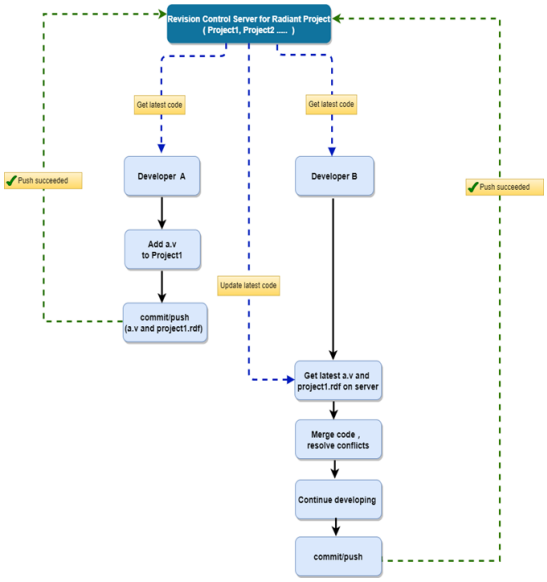
You may have your code changes and modify the same source file or design. For example, if you submit a file and another user updates it, you will get an error. At this time, you need to update the latest code locally from the server, integrate it with your modifications, and submit it again. This is the collaborative operation of multiple users for the revision control system.



Multiple developers adding files

If you add a new a.v file in the design, and submit this file and the Radiant project file to the revision control server, another user can do the following:

- Get the latest code from the server
- Merge the local code
- Resolve the conflict
- Continue the subsequent development



Managing Constraint Files

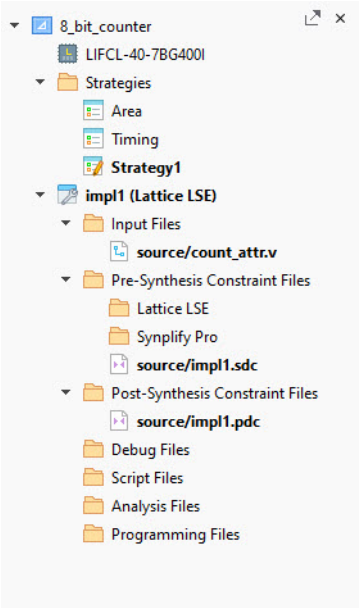
The following are the different types of constraint files available in the Radiant software.

- .ldc (via Lattice Synthesis Engine (LSE))
- .sdc (via Synplify Pro and LSE)
- .fdc (via Synplify Pro)
- .pdc constraint files

About Synthesis Constraint Files

You can add .fdc files for Synplify Pro or .ldc files for LSE, depending on the synthesis tool you choose. If the selected synthesis tool is LSE or Synplify Pro, the constraint files can be added as an .sdc file. The files are listed in the Synthesis Constraint Files folder in the File List view.

Figure 3:



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

To add a pre-synthesis constraint file:

1. Right-click the **Pre-Synthesis Constraint Files** folder in the File List view, and choose **Add**.
2. Choose **New File** if you want to add a new pre-synthesis constraint file.
   a. Select the source files to add.
   b. Enter a file name and select a location.
   c. Click **New**.
3. Choose **Existing File** if you want to add an existing pre-synthesis constraint file.
   a. Browse to the location of the pre-synthesis constraint file to add and select it.
   b. Click **Add**. The selected file is added to your project.

To activate a pre-synthesis constraint file:

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

The selected pre-synthesis constraint file is now activated.

Note

- For LSE, you can make zero or one (.sdc or .ldc) constraint file active in your project.
- For Synplify Pro, you can make multiple (.sdc or .fdc) constraint files active in your project.

To edit a pre-synthesis constraint file:

- Double-click the pre-synthesis constraint file you want to edit from the File List view of the Radiant software. Or, right-click the pre-synthesis constraint file you want to edit from the File List view of the Radiant software and click **Open**.

  The selected pre-synthesis constraint file is opened in the Source Editor. You can edit the selected file in the editor. See "Using Templates" on page 116 for more information.

To remove a pre-synthesis constraint file:

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

  The selected file is removed from your project.

About Post-Synthesis Constraint Files

You can add one or more Physical Constraint files (.pdc) to your project. They are listed in the Post-Synthesis Constraint File folder of the File List view.

Note

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

Adding, Activating, Editing, and Removing Post-Synthesis Constraint Files You can add, activate, edit, or remove post-synthesis constraint files (.pdc) of your project.

To add a Post-Synthesis Constraint file:

1. Right-click the **Post-Synthesis Constraint Files** folder in the File List view of the Radiant software, and choose **Add**.
2. Choose **New File** if you want to add a new post-synthesis constraint file.
   a. Select the source files to add.
   b. Enter a file name and select a location.
   c. Click **New**.
3. Choose **Existing File** if you want to add an existing post-synthesis constraint file.

a. Browse to the location of the post-synthesis constraint file to add and select it.

b. Click **Add**. The selected file is added to your project.

To activate a Post-Synthesis Constraint file:

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

  The selected file is activated.

Note

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

To edit a Post-Synthesis Constraint file:

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

  The selected post-synthesis constraint file is opened in Source Editor. You can edit the selected file in the editor. See "Using Templates" on page 116 for more information.

To remove a Post-Synthesis Constraint file:

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

  The selected file is removed from your project.

See "Applying Design Constraints" on page 196 for more information.

See Also

- "Managing Projects" on page 17

Setting the Top-Level Unit for Your Project

The top-level unit (or module) of the design must be specified. If it's not, the Radiant software will attempt to determine the top-level unit on its own.

Since the possibility exists that the Radiant software could choose the incorrect unit, it is recommended that you manually set the top-level unit.

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 **Set Top-Level Unit**.

The Project Properties dialog box opens with the cursor in the **Top-Level Unit** Value cell.

3. Type the name of the top-level unit.

   The Value cell also shows an arrow for a drop-down menu. Usually the menu shows one name or just “<Edit>.” But if the Radiant software 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.

4. Click **OK**.

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

Another way to set the top-level unit for your project is to right-click any source file in File List view and choose **Set Top-Level Unit** from the drop-down menu.

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

See Also

- “Managing Projects” on page 17

Modifying a Source File

Double-click on a source file to edit it using the Source Editor.

Note

For Windows file associations to work properly with the Radiant software, choose **Tools > Options > File Associations** and assign default programs for any of the programs listed.

You can use the Source Editor to enter or edit VHDL (.vhd), Verilog HDL (.v), or constraints (.sdc/.ldc/.pdc) files.

Modifying a Source File using a Different Text Editor Alternatively, you can use the text editor of your choice to edit your source files, and then import them into your project using **File > Add > Existing File** from the Radiant software main window.

Note

- A .sdc or .ldc file can also be created or edited in Pre-Synthesis Constraint Editor.
- A .pdc file can also be created or edited in Post-Synthesis Timing Constraint Editor, Device Constraint Editor, or Physical Designer.

Do the following to open a source file using a different text editor:

1. In the File List view, right-click the text file and choose **Open with** from the pop-up menu.

2. Click **Add** if you want to add your favorite text editor.

   Note

   If you have already added a new text editor, click Edit to open the source file in that editor.

3. 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

   a. In the Options dialog box, choose **File Associations**. From the File Associations table, choose the file type you want to open/edit in the new text editor.

   b. Click **Add**, and use the **External Program** field to add the text editor, then click **OK**.

   c. Use the **Edit** or **Remove** button to manage your file and associated editor list.

   d. After adding or editing the external text editor, you can save it as your default text editor by clicking **OK**.

See Also

- "Managing Projects" on page 17

Excluding a Source File

You can exclude specific source files from logic synthesis and simulation. The source files do remain in your design project, but 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 and the target file is grayed out.

Note

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

See Also

- "Managing Projects" on page 17

Removing a Source File

You can remove a source file from a project by using the Radiant software **Remove** command or the Windows' **Delete** command.

Removed source files are no longer part of your project.To simply exclude a file from logic synthesis and simulation, see "Excluding a Source File" on page 40.

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 Radiant software removes the source from the project.

See Also

- "Managing Projects" on page 17

Working with Implementations

Implementations define the design structural elements of 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 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, which includes all related files and settings.

Implementations consist of the following folders/files.

- Gate-Level Simulation File
- Input Files
- Pre-Synthesis Constraint Files
- Post-Synthesis Constraint Files
- Debug Files
- Script Files
- Analysis Files
- Programming FIles

Creating New Implementations

To create a new implementation:

1. With the project opened in the Radiant software, 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. Choose the synthesis tool from the drop-down menu.
5. Choose the default strategy from the drop-down menu.
6. 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**.
7. 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 source directory** option.
8. To view or set properties of the source files, select a file and click **Properties**. The Properties dialog box opens.
   a. To change a property, click in its **Value** cell and enter or choose the new setting.
   b. Click **OK** to close the Properties dialog box.
9. Click **OK** to close the New Implementation dialog box.

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. Choose the synthesis tool from the drop-down menu.
4. Decide how you want to handle files that are outside of the original implementation directory. Select one of the following 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. The Location and the Default Strategy text boxes are not editable.

5. Click **OK**.

Customizing Files in an Implementation After creating an implementation, you can choose to include or exclude source files to or from the implementation folders in File List view.

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 or Deactivate Files You can have multiple files under each of the implementation folders in File List view. Except for Synplify Pro constraints files, only one file can be active in each of the folders. To change a file's status to active or inactive, right-click the file and choose **Set as Active** or **Set as Inactive**.

Note

- For LSE, you can make zero or one (.sdc or .ldc) constraint file active in your project.
- For Synplify Pro, you can make multiple (.sdc or .fdc) constraint files active in your project.

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

See Also

- "Managing Projects" on page 17
- "Using Strategies" on page 43

Using Strategies

A **strategy** provides a unified view of all settings related to the optimization controls of an implementation tool, such as logic synthesis, mapping, and place-and-route. There are two different kinds of strategies: predefined strategy and customized strategy.

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 fast placement and routing attempt to take a quick look at the design, and another that's higher level and more thorough that takes longer 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. Open the dialog box by double-clicking a strategy name in the File List view. Each strategy is stored as an .sty file.

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**

  Use this strategy to achieve area requirements. 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. However, when it works, you will see an area reduction.

- **Timing**

  The Timing strategy attempts 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.

  This strategy may increase your runtime on place-and-route compared to the Quick and Area strategies. However, you will get improved time performance results when it works.

Customized Strategy Excluding the predefined strategies, all other strategies can be customized in the Strategies dialog box. Edit the settings of a strategy by:

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

See Also

- “Managing Projects” on page 17
- “Creating a Strategy” on page 45
- “Cloning a Strategy” on page 45
- “Adding an Existing Strategy” on page 46
- “Specifying Strategy Options” on page 46
- “Saving a Strategy” on page 47
- “Removing a Strategy” on page 48
- “Strategy Reference Guide” on page 697

Creating a Strategy

New strategies can be created for your design.

To create a new strategy in the Radiant software:

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

   The New Strategy dialog box opens.

2. Enter a name for the new strategy.
3. Specify a file name for the new strategy
4. Choose a directory to save the strategy file (.sty) to.

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

To save the strategy changes to your current project, choose **File > Save Project** from the Radiant software main window.

See Also

- "Managing Projects" on page 17

Cloning a Strategy

Clone any previously created strategy or any of the Lattice predefined strategies from the File List view. The new cloned file contains 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 Strategy**.

   The Clone Strategy dialog box opens.

2. Enter a name for the new strategy in the Strategy ID field. Note that the name used automatically appears as the file name.
3. Change the file name, if desired.
4. Choose a directory to save the strategy file (.sty) to.

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

To save the strategy changes to your current project, choose **File > Save Project** from the Radiant software main window.

See Also

- "Managing Projects" on page 17

Adding an Existing Strategy

A previously created strategy can be added to your design.

Note

A strategy copied from another design project may include incorrect 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 the Radiant software, 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 new strategy. The name chosen appears in the File List view.

Note

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 to avoid using a file being used by another project.

4. Click **OK**.

   The new strategy appears in the File List view.
5. Double-click the new strategy to open the Strategies dialog box.
6. Review the settings to ensure they are appropriate for your current project. Be sure to check the following:
   - Under Synthesize Design > LSE:
     - Macro Search Path
     - Memory Initial Value File Search Path
   - Under Post-Synthesis:
     - External Module Files (.udb)
7. After completing your review, click **OK**.

To save the strategy to your current project, choose **File > Save Project** from the Radiant software main window.

See Also

- “Managing Projects” on page 17

Specifying Strategy Options

Use the Strategy dialog box to specify settings for a specific design strategy process.

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. The left-hand pane shows a list of processes. Each process has its own settings.
3. In the right pane, double-click the Value column for the setting to be changed. A pulldown menu is displayed with a choice of settings. You can also enter filenames or directory paths into the field shown.
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 of your changes are saved to your strategy (.sty) file.

Tip

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

See Also

- “Managing Projects” on page 17
- “Strategy Reference Guide” on page 697

Saving a Strategy

To save a new or cloned strategy, or any change of a strategy setting to your project, click **File > Save Project** from the Radiant software main window.

See Also

- “Managing Projects” on page 17

Setting Active Strategy

Set an active strategy if you have more than one strategy in a project.

To set an active strategy:

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

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

See Also

- “Managing Projects” on page 17

Removing a Strategy

You can remove an inactive user strategy from your design.

To remove an inactive user strategy:

Right-click on the target strategy and choose **Remove**.

See Also

- “Managing Projects” on page 17

Getting Help on Strategies

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

- Open the “Strategy Reference Guide” in the Contents pane of the Radiant software Help to display all process options in the order they appear in the Strategies dialog box. See “Strategy Reference Guide” on page 697.
  - Choose the desired process option. Strategies associated with this process are displayed.
  - Use the scroll bar to select a setting and view its description.
- In the Radiant software File List view, double-click a strategy name to open the Strategies dialog box.
  - 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

- “Managing Projects” on page 17
- “Strategy Reference Guide” on page 697
- “Creating a Strategy” on page 45
- “Cloning a Strategy” on page 45
- “Adding an Existing Strategy” on page 46
- “Specifying Strategy Options” on page 46
- “Saving a Strategy” on page 47
- “Setting Active Strategy” on page 47
- “Removing a Strategy” on page 48

Analyzing a Design

You can analyze your design in the following ways while it is being implemented.

Before synthesis:

- The Hierarchy view provides a nested list of all modules and commands to access the source code for individual modules, set up test benches, and more. See "Hierarchy View" on page 49.
- The Netlist Analyzer provides the schematic view to analyze user RTL in schematic form. The Netlist Analyzer RTL view is only available when LSE is used.

After synthesis, schematic views are also available. These views depend on which synthesis tool you are using:

- LSE: you can use the Radiant software's Netlist Analyzer to see and analyze the synthesized and mapped design.
- Synplify Pro: you can use Netlist Analyzer to see and analyze the synthesized design or you can use its HDL Analyst. See *Synplify Pro for Lattice User Guide*.

If you are using LSE, Netlist Analyzer shows how the design is mapped to the device's architecture.

You can view the order of cells representing the sequential flow of signals through a circuit. Netlist Analyzer follows the path of the net and displays the cells along that path, maintaining the proper sequence from the source (start point) to the destination (end point).

You can cross-probe from the Critical Paths Summary tab or the Query tab of Timing Analyzer to Netlist Analyzer. Refer to "Path Details in Timing Analyzer" on page 363.

See Also

- "Managing Projects" on page 17

Hierarchy View

The Hierarchy view appears in the left pane below the project file list view and shows the design hierarchy as a nested list of modules, and launches automatically when you open a project. If the Hierarchy view is not open, choose **View > Show Views > Hierarchy** from the Radiant software main window.

The Hierarchy view also shows the full path name of the files that define each module.

See Also

- "Managing Projects" on page 17

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 Netlist Analyzer

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 (“<*module_name*>_tf.v”) 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 “<*module_name*>_tf.v” file.

To create a Verilog test fixture from a Verilog Test Fixture Template (“<*module_name*>_tf.v”) file:

1. Open the “<*module_name*>_tf.v” file from the project directory.
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 (“<*entity_name*>_tb.vhd”) 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.

Use the following procedure to create a VHDL Test Bench from the “<*entity_name*>_tb.vhd” file.

To create a VHDL test bench from a VHDL Test Bench Template ("<*entity_name*>_tb.v") file:

1. Open the "<*entity_name*>_tb.vhd" file from the project directory.
2. Import the new .vhd file into the project as a VHDL Test Bench. Then add codes to the user defined section to complete the stimulus for your design.

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.

Set as Top-Level Unit

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

See Also

- "Managing Projects" on page 17

About Netlist Analyzer

Netlist Analyzer works with LSE and Synplify Pro to produce schematic views of your design while it is being implemented. 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 or Synplify Pro.
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.

Note:

RTL view is not supported in Synplify Pro.

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

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 consists of 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 consist 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.

Note that you can adjust the view of a schematic and navigate through the hierarchy in multiple ways.

See Also

- “Managing Projects” on page 17

Setting Netlist Analyzer Options

With Netlist Analyzer you can specify the following:

- Labels in the schematics
- How the hierarchy gets expanded
- Highlight color
- Size of the sheets

To change tool options:

1. Choose **Tools > Options**.

   The Options dialog box opens.
2. In the left-hand list, click **Tools**.
3. Click the **Netlist Analyzer** tab.
4. Set the options.
5. Click **OK**.

Refer to the following instructions for specific options.

To specify labels:

In the **Netlist Analyzer** tab, under **Text**, select the labels that you want to see in the schematics.

To control how the hierarchy gets expanded:

In the **Netlist Analyzer** tab, under **Schematic,** adjust the values of the following items:

- Hierarchy Depth: The number of levels to show when using the Dissolve Instances command. The default, 0, displays 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, displays all levels.

To change the sheet size:

In the **Netlist Analyzer** tab, under **Sheet**, type the preferred width and height values.

See Also

- “Managing Projects” on page 17

Adjusting the Schematic View

Netlist Analyzer schematics are usually displayed on several sheets, depending on the size of the design. More than one schematic view can be open at the same time.

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 change the layout in a sheet:

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

See Also

- “Managing Projects” on page 17

Zooming and Panning

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

Toolbar Commands

The following commands are available on the Radiant software 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

Zoom by holding the left 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 left. 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-left corner of the area that you want to see better and drag to the lower-right corner of the area. The area that you drag across is adjusted to fill the window.

Zooming and Panning with World View

The World View is a small display representation of a schematic that appears in the lower-left corner of the Netlist Analyzer window. The World View shows the entire sheet with a purple rectangle marking the area that is actually showing in the schematic view.

If the World View is not shown, click the World View button at the lower-left corner of the schematic view.

To enlarge the World View, 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.

Click on the spot that you want to see. The schematic view jumps to that spot and centers the view.

See Also

- "Managing Projects" on page 17

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 PDF or SVG.
6. Click **Save**.

See Also

- "Managing Projects" on page 17

Printing Schematics

Use the print functions of the Radiant software to print the schematics.

To print a schematic:

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

   The Print Preview window opens.
2. Expand the Print Preview window to the desired size.
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**.

See Also

- "Managing Projects" on page 17

Navigating the Design with Netlist Analyzer

Explore a design by selecting one or more objects and then right-clicking in the schematic view and choosing a command from the drop-down menu. These commands help you to:

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

Most commands can be reversed by clicking the Back button. They can be repeated by clicking the Forward button. To jump back to the original view after using one or more 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, drag the port, net, or register to the relevant tab of LDC Editor. See "Handling Device Constraints" on page 212.

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 on all sheets of the schematic.
- Right-click in the schematic view and choose **Select All Schematic > Ports**. This command selects all ports displayed on all sheets of the schematic.
- Right-click in the schematic view and choose **Select All Sheet > Instances**. This command selects all module instances displayed on the current sheet of the schematic.
- Right-click in the schematic view and choose **Select All Sheet > Ports**. This command selects all ports displayed on 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 highlights the selected objects using the selected color while you continue to explore the design. Highlighted objects are not “selected” for commands. Highlighting can be erased by using the Back button.

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

See Also

- “Managing Projects” on page 17

Focusing on Part of the Design

You can focus on a specific part of a design in one of several ways:

- Filter the schematic.
- Reduce it to selected modules.
- Trim away objects that are in the way.
- Request more details of a single module.

To filter the schematic:

1. Select one or more modules from either the netlist browser or schematic view.
2. Click the Filter Schematic button.

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

To filter with signal paths:

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

The schematic is replaced with 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 to hide.
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, if the module is not already a primitive), the cursor changes to a blue down arrow.
3. Click in the module with the blue down 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, continue to click in a module with a down arrow.

To go back up the hierarchy, click a green up arrow when it is not over a module.

To perform other commands in the schematic, click the Select button.

To quickly push down or pop up in a module:

Do the following 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 one module in RTL view:

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 Netlist Analyzer**.

   This opens Netlist Analyzer with a schematic view of the design element. The rest of the design is not available in this schematic.

See Also

- “Managing Projects” on page 17

Flattening the Design

An alternative to viewing a hierarchy is to flatten the design, which displays all the layers together in a larger, more detailed schematic. By doing so, 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, labels of the module, and all submodules are included to help identify and select the modules and their ports.

See Also

- "Managing Projects" on page 17

Tracing Paths between Objects

Use Netlist Analyzer to trace a signal within a complex schematic.

To trace a signal path:

1. Select two or more objects on the same or 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.

See Also

- "Managing Projects" on page 17

Expanding from a Module

After focusing a schematic on one or more modules, do the following 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, and click the green up-pointing arrow when it appears.

See Also

- "Managing Projects" on page 17

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 a port pointing out or 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 other object that is on the net.
2. If the net appears as a dashed line, only part of the net is being shown. Double-click the port again to see another branch of the net.

   If the net shows as solid line, the full net is being displayed.

   If the net is fully visible, nothing else can be done.

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.

See Also

- "Managing Projects" on page 17

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.

See Also

- "Managing Projects" on page 17

Searching in Netlist Analyzer

With larger 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 search for by clicking one of the tabs at the top of the dialog box.
3. Select the part of the design to search for 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 a previously used search term.
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 results, click either **Next 200** or **Find All**.
7. To highlight found objects in the schematic view, do one of the following:
   - Select the desired objects in the UnHighlight box and then click the -> (right arrow) button.
   - For all found objects, click the **All ->** button.

   The selected objects move to the Highlight box and are highlighted in the netlist browser and schematic view. 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.

See Also

- “Managing Projects” on page 17

Getting More Information about an Object

Additional information about an object in a schematic view is available, including names, number of fanouts, source code, and pins that the nets connect to.

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 properties for 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. Click **Close** when finished.

To get 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**.

   The Source Editor opens with the code highlighted.

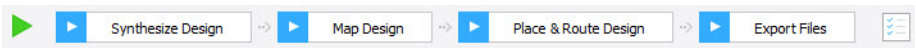
See Also

- "Managing Projects" on page 17

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 Toolbar.

Process Toolbar



Click the Task Detail View to see detailed information of the processes.

Processes are grouped into categories according to their functions.

- **Synthesize Design**

Click on this process to have LSE synthesize the design. By default, this process runs the LSE tool. For details, see "Synthesizing the Design" on page 311.

If you are using Synplify Pro, choose Synplify Pro as the synthesis tool (**Project > Active Implementation >Select Synthesis Tool).** See "Selecting a Synthesis Tool" on page 70 for details).

- Post-Synthesis Timing Analysis

  Runs timing analysis after the Synthesize Design process.

- Post-Synthesis Simulation File

  Generates a netlist file <file_name>_syn.vo used for functional verification.

- **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 (configurable logic blocks, for example). For details, see the **Implementing the Design >** "Mapping" on page 320.

  - **Map Timing Analysis**

    Runs Map timing analysis.

- **Place & Route Design**

  After a design has been translated into the Native Circuit Description (.ncd) format, you can run the Place & Route Design process. This process takes a mapped physical design .ncd file, places and routes the design, and outputs a file that can be processed by the design implementation tools. For details, see "Place and Route" on page 327.

  - **Place & Route Timing Analysis**

    Runs Place & Route timing analysis.

  - **I/O Timing Analysis**

    Runs I/O timing analysis.

- **Export Files**

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

  - **Bitstream File**

    This process takes a fully routed physical design or .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.

    To confirm that your production bitstream does not include debuggers, you can check the bitstream report under Export Report. Refer to "Viewing Logs and Reports" on page 85.

    For other details on generating bitstreams, refer to "Bit Generation" on page 341.

See Also

- "Managing Projects" on page 17
- "Managing Project Sources" on page 25
- "Process State" on page 65
- "Starting a Process" on page 66
- "Forcing a Process to Run" on page 66
- "Stopping a Process" on page 67
- "Cleaning Up Processes" on page 67

Process State

The Process Toolbar 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	Process Toolbar 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.
Completed		Process has been completed with unprocessed subtask(s)
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 .pdc file will cause the map process to be rerun.

  Note

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

- Process settings have changed.
- Target device has changed. All processes will be re-initialized.
- A **Force Run** function is run.

See Also

- “Managing Projects” on page 17
- “Running Processes” on page 63
- “Starting a Process” on page 66
- “Forcing a Process to Run” on page 66
- “Stopping a Process” on page 67
- “Cleaning Up Processes” on page 67

Starting a Process

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

To start a process, do one of the following:

- In the Process Toolbar, click the desired process.
- Right-click the process in the Process Toolbar and choose **Run**.

See Also

- “Managing Projects” on page 17
- “Running Processes” on page 63
- “Process State” on page 65
- “Forcing a Process to Run” on page 66
- “Stopping a Process” on page 67
- “Cleaning Up Processes” on page 67

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:

- Right-click the process in the Process Toolbar and choose **Force Run**.

See Also

- “Managing Projects” on page 17
- “Running Processes” on page 63
- “Process State” on page 65
- “Starting a Process” on page 66
- “Stopping a Process” on page 67
- “Cleaning Up Processes” on page 67

Stopping a Process

Do the following to stop running a certain process:

- Right-click the process in the Process Toolbar and choose **Stop**.

See Also

- “Managing Projects” on page 17
- “Running Processes” on page 63
- “Process State” on page 65
- “Starting a Process” on page 66
- “Forcing a Process to Run” on page 66
- “Cleaning Up Processes” on page 67

Cleaning Up Processes

Reset all processes by right-clicking a process in the Process Toolbar and choosing the **Clean Up Process** command. This command returns the status of all of the processes back to their initial state.

See Also

- “Managing Projects” on page 17

Clearing Tool Memory

The Clear Tool Memory command, available from the Tools menu, clears the device, design, and preference information from system memory. Clearing the tool memory can speed up memory-intensive processes such as place-and-route. When your design is very large, it is good practice to clear memory prior to running place-and-route.

If you have open tool views that are affected by clearing the tool memory, a confirmation dialog box opens to give you the opportunity to cancel the memory clear.

To clear tool memory:

1. In the Radiant software main window, choose **Tools > Clear Tool Memory**.
2. In the dialog box, choose the tool you wish to clear memory information from.
3. Click **OK**.

For more information about shared memory, refer to Chapter 10 of the *Lattice Radiant Software User Guide*.

Setting Options for Synthesis and Simulation

The Radiant software is integrated with LSE, the Synplify Pro synthesis tool, and the QuestaSim simulation tool. Besides the OEM tools, you can also let the Radiant software use LSE or Synplify Pro as the synthesis tool, and full-featured versions of QuestaSim or Active-HDL as the simulation tool.

This section covers steps on how to do the following:

- Selecting synthesis and simulation tools
- Defining file order for synthesis and simulation
- Customizing synthesis tool processes
- Specifying VHDL library name
- Specifying the search path for Verilog include files

Performing interactive synthesis is also discussed in this section.

See Also

- "Managing Projects" on page 17

About Lattice Synthesis Engine

For most devices, LSE can be used as your synthesis tool instead of Synplify Pro for Lattice or any other third-party synthesis tool.

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

In addition, 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.
- Post-synthesis Verilog netlist suitable for simulation.

When considering LSE, note the following limitations:

- 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.
- Does not include interclock domain paths in timing analysis.

See Also

- “Managing Projects” on page 17
- “Selecting a Synthesis Tool” on page 70

Switching to Lattice Synthesis Engine

When changing to LSE from another synthesis tool, the following must be done:

- Consider creating a new design implementation. You can experiment with different settings without losing your previous work. See “Working with Implementations” on page 41.
- Modify the Synopsys Design Constraint (.sdc) file into an .ldc or create a new .ldc file. See “Lattice Synthesis Engine Constraints” on page 750 and “Handling Device Constraints” on page 212.
- Adjust the LSE strategy settings. See “LSE Options” on page 709 and “Optimizing LSE for Area and Timing” on page 314.
- Review the LSE coding tips and consider applying them to your design. See “Coding Tips for LSE” on page 121.
- Review the instructions for integrated synthesis. There are some differences with LSE. See “Integrated Synthesis” on page 317.

See Also

- “Managing Projects” on page 17

Bus Naming in LSE Output

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

sum_c_0
sum_c_1
sum_c_2
sum_c_3

See Also

- “Managing Projects” on page 17

Selecting a Synthesis Tool

The Radiant software supports Verilog HDL and VHDL designs using LSE or Synplify Pro as the synthesis tool.

- The Radiant software is fully integrated with LSE and Synplify Pro. “Fully integrated” means that you can set options and run synthesis entirely from within the Radiant software. When you create a new project for any device, the default synthesis tool is Synplify Pro.
- If you prefer, you can use other synthesis tools by running them independently of the Radiant software.

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 or Synplify Pro.
2. From the Radiant software 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**.

You can also select a default synthesis tool by doing the following:

1. From the Radiant software main window, choose **Tools > Options > Tools > Synthesis**.

   The Options dialog box opens.
2. In the **Synthesis** tab, you can choose LSE or Synplify Pro as the default synthesis tool.
3. Click **OK**.

   When you create a new project, you can see the chosen synthesis tool in the Select Synthesis Tool dialog box.

If you have chosen **Synplify Pro**, you can either use Synplify Pro for Lattice or your own full-featured Synplify Pro. By default, the Radiant software uses 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 it is in by doing the following:

1. Choose **Tools** > **Options** > **General**.

   The Options dialog box opens.
2. In the **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 a different synthesis tool, you need to specify the directory in the Options dialog:

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

See Also

- “Managing Projects” on page 17
- “Customizing Synthesis” on page 77

Specifying VHDL Library Name

The Radiant software supports VHDL design synthesis using LSE Synplify Pro. 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 tool. See Synplify Pro 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 tool uses the specified library name to synthesize the selected VHDL source file or module.

See Also

- “Managing Projects” on page 17

Specifying Search Path for Verilog Include Files

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

Note

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

You can also specify a Verilog include file search path directly in LSE, or Synplify Pro. See “Running Synthesis from the Command Line” on page 1489, and the Synplify Pro 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.
3. The current project path is added automatically by the Radiant software if you do not specify either the 'include' directive or the directories specified in the “Search Path” option for the entire project. Therefore, you do not need to specify the project path in the Radiant software. The path will be added automatically.

Specifying Search Path in the Radiant software

To specify the search path in the Radiant software:

1. In the File List view, right-click the implementation name if you want to specify the search path for the entire project.
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:
   - To add a single search path, click the Value field to enter a path.
   - To add multiple paths, click the **...** button from the Value field to get the Verilog Include Search Path dialog box. Next, 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 finished.

4. Click **OK** to exit the Project Properties dialog box.

   The search paths you added are applied. The Synplify Pro tool uses 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.

See Also

- "Managing Projects" on page 17

Setting Top-Level HDL Parameter

- Top-level parameters can be passed on the command line or in the .synproj file with the **-hdl_param <PARAM> <value>** option.
- Numeric values should be entered directly, and string values should be surrounded by double quotes. For example:
  - -hdl_param EXP 3.5e5
  - -hdl_param LOG $clog2(512)
  - -hdl_param REG "BYPASSED"
- This setting can also be configured in the GUI via Project Properties in the HDL Parameters field: **<PARAM>=<value>**.

To configure HDL parameters using Project Properties:

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

   The Project Properties dialog box opens.

2. In the **HDL Parameters** row, click the cell under **Value**.

   When entering values using Project Properties, you should ensure that the values are kept as you entered them and correctly interpreted as either a number or a string.

3. When done, click **OK**.

See Also

- "Managing Projects" on page 17

Modifying Verilog and VHDL Parameters

You can modify Verilog parameters or VHDL generics by doing one of the following:

- Parameters can be configured in RTL code for both top and sub modules.

- Project Properties can set HDL Parameters exclusively in the top module.

You may find the following method an easier way to experiment with these settings.

To modify Verilog 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 modified statements as a single text line with different statements separated by semi-colons (;).
4. When done, click **OK**.

See Also

- "Managing Projects" on page 17

Specifying Verilog Directives

Verilog directives are instructions to the Verilog compiler. They control how the compiler processes the code. A directive in one file can control compilation behavior in multiple files.

Verilog compiler directives begin with the grave accent ` and do not end with a semi-colon. The macro_name must also be preceded by the grave accent. A directive is implemented from the point it appears in the code and remains in effect through all the file until it is superseded by another directive or the last file to be processed has ended.

Here are some Verilog compiler directives. For complete details, refer to the IEEE Standard for Verilog Hardware Description Language.

Directive	Description
`include	Copies all the contents of one Verilog source file and includes them in the current file during compilation. Syntax: `include "file_name"
`define	Declares a macro or defines a custom data type. Syntax: `define macro_name [(argument, ...)] text
`undef	Removes a defined macro. Syntax: `undef macro name

Directive	Description
`ifdef	Compiles the underlying code only when the macro is defined. It is similar to an if-else conditional statement but is processed during compilation. The `ifdef directive is always closed by an `endif directive to mark the end of the condition. Syntax: `ifdef macro_name <code> `endif
`ifndef	Compiles the underlying code only when the macro is not defined. It is the opposite of the `ifdef directive. Syntax: `ifndef macro_name <code> `endif
`elseif	Provides other options when used with `ifdef or `ifndef. This is similar to if-else-if statements.
`else	Defines a default case. It is not required to include this directive with `ifdef or `ifndef.
`timescale	Defines the time scale of the simulation. The time scale is divided into unit and precision. Syntax: `timescale unit base/precision base

To add Verilog directives:

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 Verilog Directives row, click under **Value**.
3. Type the statements as a single text line with different statements separated by semi-colons (;).

   For example:

   directive1=value1;directive2=value2;directive3=value3

   The '=value' part is optional and the ';' is a separator character.

Note

Do not leave spaces around the '='. The ';' is not allowed to exist in your directive name or value.

The Tcl command to set this option is:

```
'prj_set_impl_opt VERILOG_DIRECTIVES <value>'
```

The <value> should be recognized by the Tcl interpreter as a Tcl list with each element in 'key=value' format. If you want to define a directive value containing space, the value Tcl format should use '{}' to surround the 'key=value' element in your Tcl command.

For example:

```
prj_set_impl_opt -impl impl_1 VERILOG_DIRECTIVES
{USRACCESS_DATA=32'h1234_5678 ANOTHER_DATA=32'h5678_1234}
```

For prj_set_impl_opt -append, you can append one key=value element.

Keep in mind that PN does not check if the definition is correct in the Verilog syntax. PN only passes the directive name and its value to the synthesis engine or to the simulator. The user has the responsibility to make sure the directive name and value are accepted in the Verilog parser.

See Also

- "Managing Projects" on page 17

Selecting a Simulation Tool

The Radiant software supports functional and timing simulation for Lattice FPGA devices using QuestaSim or Active-HDL. You can specify either one 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 178.

See Also

- "Managing Projects" on page 17

Specifying Input Files for Simulation

If your design includes one or more test bench files to use in your simulation but you don't want to synthesize them, 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**.

See Also

- "Managing Projects" on page 17

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 and Verilog files. If you have not manually set a top-level module (in the Project Properties dialog box of the main window), the last file on the file list is normally 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 several files, specify them in the following file order:

1. The file containing the entity,
2. The architecture file.
3. The file with the configuration.

Verilog file order is 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 desired location.

  The new file order takes effect immediately. However, the order will only be saved once you save the design project.

See Also

- “Managing Projects” on page 17

Customizing Synthesis

The Radiant software supports Verilog and VHDL synthesizing using LSE, and Synplify Pro.

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

To adjust synthesis tool options and perform the synthesis process:

1. Choose **Project > Active Strategy >** <*synthesis tool*> **Settings**.
2. The option settings of the selected process, LSE, or Synplify Pro are listed by default in the Strategies dialog box.
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 311 for more details.

See Also

- "Managing Projects" on page 17

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 Radiant software, 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 43
- "Working with Implementations" on page 41

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 724.)
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 724.)
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. Highlight the implementation, right-click, and choose **Show/Hide Implementation.**

   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.

Running Run Manager

After setting up Run Manager (see "Setting Up Run Manager" on page 78), 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 81.

   If you set up an implementation to do multiple place-and-route runs, click the arrow 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 85.

   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 83.

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 85.

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.

  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.

  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 the Radiant software, you can find the information you need by doing the following.

1. Choose **Edit > Find in Files** from the Radiant software main window.
2. Type the text to find in the pop-up Find In Files dialog box.
3. Specify the search path and search filters.
4. Select the search parameters desired: **Search subdir**, **Include hidden files**, **Match case**, **Match whole word**, and **Regular expressions**.
5. Press **Find**. The results are 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

If the Find Results are not automatically displayed in your Radiant software main window, select **View > Show Views > Find Results** to display them.

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

See Also ▶"Managing Projects" on page 17

- "Viewing Logs and Reports" on page 85
- "Navigating Errors and Warnings" on page 88
- "Finding Text in Logs and Reports" on page 88

Viewing Logs and Reports

The Radiant software automatically 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 to view all log information.

Errors are displayed in red, while warnings are displayed in orange. 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 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, resource usage, IP summary, and the number of constraints dropped in Synthesis, Post-Synthesis, Map, and Place & Route.
▶ IP Summary	Generates a summary report in the implementation folder. The IP Summary report provides the following list of information: ▶ IP Name – displays the name of the latest generated IP or, by default, displays the IP internal name. ▶ Vendor – displays the IP vendor name, or by default displays the IP internal vendor. ▶ IP Module Name – shows the top module name ▶ Version – specifies the current IP version. ▶ License Feature – indicates the license feature and starts with “LSC_IP_” in hdl files of IP. This “-” denotes that there is no license feature in IP. ▶ License Status – indicates if the license check is successful or failed. This “-” denotes that there is no license feature in IP.
Synthesis Reports	Lists Constraint Propagation, synthesis tool (Lattice LSE or Synplify Pro), Synthesis Resource Usage, Post-Synthesis Report, and Post-Synthesis Timing reports in HTML format.
▶ Constraint Propagation	Shows the result of the Constraint Propagation Engine.
▶ Synthesis Tool (Lattice LSE or Synplify Pro)	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.
▶ Synthesis Resource Usage	Categorizes resource usage of an instance.
▶ Constraint Checker Report	The Constraint Checker Report is part of the Synthesis Report section when Synplify Pro is used. Information about invalid timing constraint syntax, constraint applicability, and any errors or warnings are displayed in this report.

Type of Report	Description
▶ Post-Synthesis Report	This report has a Constraints Summary section that lists the total number of constraints and the number of constraints dropped.
▶ Post-Synthesis Timing Report	Describes the Design Checking, Clock Summary, Timing Analysis Summary, and Detailed Report sections. For more information about Timing Reports, see "Reading Timing Analysis Reports" on page 352.
Map Reports	Lists Map, Map Resource Usage, and Map Timing Analysis reports in HTML format.
▶ Map	Lists various sections that show information about the device. MAP Report also lists the total number of constraints and the number of constraints dropped. For a detailed information about Map Report, see "MAP Report File" on page 322.
▶ Map Resource Usage	Categorizes the LUT usage to logic, distributed RAM, and carry chain ripple logic.
▶ Map Timing Analysis	Describes the Design Checking, Clock Summary, Timing Analysis Summary, and Detailed Report sections. For more information about Timing Reports, see "Reading Timing Analysis Reports" on page 352.
Place & Route Reports	Lists Place & Route, Signal/Pad, and Place & Route Timing Analysis, and I/O Timing Analysis reports in HTML format.
▶ Place & Route	Contains execution information and shows the steps taken as the process converges on a placement and routing solution. For a detailed information about Place & Route Report, see "PAR Report File" on page 330
▶ Signal/Pad	Lists all Programmable I/O Cells (PICs) used in the design and their associated primary pins. For a detailed information about PAD Report, see "PAD Specification File" on page 334
▶ Place & Route / I/O Timing Analysis	Describes the Design Checking, Clock Summary, Timing Analysis Summary, and Detailed Report sections. For more information about Timing Reports, see "PAR Report File" on page 330.
Export Reports	Lists Bitstream and IBIS Model reports in HTML format.
▶ Bitstream	Shows output information of the bitstream process.
▶ IBIS Model	Shows information about the generated design-specific model file.
Misc Reports	Lists Last Build Log, Tcl Command Log, I/O SSO Analysis, and Constraint DRC in HTML format.
▶ Last Build Log	Lists information about the project and shows error/warning messages.
▶ TCL Command Log	Shows output information performed by the Tcl Console Pane.
▶ I/O SSO Analysis	Shows information produced by the Device Constraint Editor. For more information about I/O SSO Analysis, see "Viewing SSO Analysis Results" on page 306
▶ Constraint DRC	Shows output information performed by Constraint DRC.

Cross probing Reports Between the different tool views in Radiant, a user can click on a hyper-link icon to cross probe into that tool.

- Post-Synthesis & Map timing report links to Netlist Analyzer.
- In the PAR timing report, user can cross probe to Netlist Analyzer, Physical Designer Placement Mode and Routing Mode.

Note

When the Reports tab is detached from Radiant main GUI, Reports window and Radiant main GUI are same level windows, which means they could be put in different displays, and also be put side by side.

But, If they are both in maximum size mode and displayed in same monitor, you may only see one. In this case If you view Reports window in maximum mode and use cross probing, it will cross probe to Radiant main GUI which is not activated and invisible. To check the cross probing result, please activate Radiant main GUI to make it visible.

Tracing Signals If your project uses LSE as the synthesis tool, you can highlight text (such as a pin, or a net, etc.) in any report view or output window; trace the signal; and then display the signal in either the RTL Netlist view of Netlist Analyzer, or the Technology Netlist view of Netlist Analyzer.

Note

The Trace feature is a search engine. The quality of your result will depend upon whether or not the text you highlight (pin, or net, etc.) is searchable.

To trace a signal in Report View or Output Window:

1. Highlight the feature you wish to trace (pin, net, etc.) in the Report View or Output Window.
2. Right-click, and in the pop-up menu choose either:
   - **Trace to > RTL**
   - **Trace to > Technology View**

   The signal will be highlighted in either the RTL Netlist View of Netlist Analyzer or the Technology Netlist View of Netlist Analyzer.

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 by right-clicking the implementation name in the File List view and choosing **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, as it 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, <*top_module*>.arearep.

In **Synthesis and MAP Resource Usage Reports**, each row shows the resource usage of an instance. There are two values in one resource type:

1. The first value is the total resource usage of the instance and its sub-instances.
2. The second value in parentheses is the resource usage value of the instance itself.

   The values in Resource Usage match according to the following formula:

   Value of the instance itself + Value of the sub-instances = Total.

See Also

- “Managing Projects” on page 17
- “Finding Results” on page 84

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 Output View, located in the lower right of the Start Page, double-click the line describing the error or warning.
- Right-click the message and choose **Location in > Text Editor**. If the command is dimmed, there is no link to a source file. Depending on the message, more than one tool may be available to view the source.

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

- “Managing Projects” on page 17
- “Finding Results” on page 84

Finding Text in Logs and Reports

You can search for text in the Output View.

To search for text in the Output View:

1. Right-click in the Output View and choose **Find in Text**.

   The Find toolbar appears.

2. In the **Find** text box, enter the text to search for. Check the **Match Case** or **Match whole word only** options as needed. As you type each character, the window displays any matches 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. Choose **Edit > Find**.
2. In the Find text box, enter the text to search for. Check the **Match Case** option if needed.

See Also

- "Managing Projects" on page 17
- "Finding Results" on page 84

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
- Severity (information, warning, and error)
- ID number
- Text

To set filter options:

As filters are added, the message list changes to show the latest results.

1. After running part or all of the implementation process, choose **View > Show Views > Message**.
2. Select the Messages Tab at the bottom of the screen.
3. Click on one of the following icons: **Error** ( ), **Critical** ( ), **Warning**( ), or **Information** ( ).
   - **Error** -- The project cannot be compiled due to an error.
   - **Critical** -- Severe issue that is not an error but could result in a functional problem.
   - **Warning** -- An issue that is not an error but could result in a functional problem.
   - **Information** -- General messages that cannot be changed.
4. Specify any message ID to hide by right-clicking the desired message ID and choosing **Filter > Filter Messages with This ID**.
5. To hide a specific group of messages, right-click on a message and choose **Filter > Filter Messages Exactly Like This**, **Filter Messages**

from this Process, or **Filter Messages with this Severity**. This hides all messages with the same text, process, and severity.

To list only messages that have been hidden, right-click in the Message area and choose **Filter > Filter Details List**. The Detailed Filter List dialog box opens showing a list of the hidden messages.

6. To unhide messages, right-click in the Message area and choose **Filter > Filter Details List**. From the Detailed Filter List dialog box, uncheck the box in the Filter Enable column of the message(s) to unhide and click **Apply.**
7. To filter messages by process, in the **Filter Message by Process**  , button, click the down arrow to display the pop up menu. By default, all processes are selected with a checkmark. Processes include **Synthesis**, **Map**, **Place & Route**, **Export**, and **Project**. To filter messages to show only messages associated with a specific process or processes, uncheck the processes that you do not wish to view. Leave checked the process or processes for which you wish to view messages.

Changing Warnings to Errors

There may be warning messages in your design project that you want to place emphasis on. You can do so by changing these warnings to show the error severity level. You can also save these “promotions” to use in other design projects.

Promotions only show for messages with future actions. Promotions do not affect the display of messages from past actions.

Note:

After changing a warning to an error, the original flow will fail during the rerun process.

To change a warning to an error in the Messages Tab:

1. Right-click on the message to highlight and choose **Change Severity with This ID to**.

   Not all warning messages can be highlighted in this way, so this command may be dimmed.

To change a warning to an error without the Reports view:

1. Choose **Project > Message Promotion/Demotion**.

   The Message Promotion dialog box opens.
2. In the IDs box, type the ID number of the message(s) 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. Click **Close** when finished.

To restore a message ID to the warning level:

1. Choose **Project > Message Promotion/Demotion**.

   The Message Promotion dialog box opens.

2. Select the message to restore.
3. Click **Remove**.
4. Click **Close** when finished.

To save the list of promoted messages for another project:

1. Choose **Project > Message Promotion/Demotion**.

   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. Click **Close** when finished.

To use an existing list of promoted messages:

1. Choose **Project > Message Promotion/Demotion**.

   The Message Promotion dialog box opens.

2. Click **Import**.
3. In the Import Promotion dialog box, browse to the list you previously saved and want to import.
4. Click **Open**.

   The list of messages appears in the Message Promotion dialog box.

5. Click **Close** when finished.

See Also

- "Managing Projects" on page 17

Getting Help for Messages

Some messages offer a link to the Help that provides additional information. This information may help you better understand an error message or how to fix a specific problem.

To check for help on a message:

1. Right-click on the message in the Output View.

   A menu appears. The last item in the menu is Help. If it is dimmed, there is no help link.

2. If the Help command is available, click on it.

   The Help opens with the appropriate topic.

See Also

- "Managing Projects" on page 17

Setting Tool and Environment Options

The Radiant software provides many environment control and tool view display options that enable you to customize your settings.

To access these options:

Choose **Tools > Options**.

The Options dialog box is organized into functional folders.

Commonly configured items include:

- General settings -- allows you to set some common options, including.
  - Startup – enables you to configure the default action at startup and also to control the frequency of checking for software updates.
  - File Associations – allows you to set the programs to be associated with different file types based on the file extensions.
  - Directories – Set directory location for Synthesis and Simulation tools.
  - Network Settings – Apply a proxy server and specify a host and port.
- Color settings -- allows you to set colors for such GUI features as fonts and backgrounds for various Radiant software tools, including:
  - General
  - Device
  - Design
  - Group
  - Reveal Analyzer
  - Source Editor

  You can also change the Theme color of Radiant software (Dark or Light) using the Themes drop-down menu.
- Tools settings -- allows you to set options for various Radiant software tools, including the Device Constraint Editor, Netlist Analyzer, Pre-Synthesis Constraint Editor, Post-Synthesis Timing Constraint Editor, and

Source Editor. You can also to check Constraint design rule checking (DRC), prior to saving or when launching a tool.

Block-Based Design - Using Macro Blocks

A macro block is a portion of an FPGA design that can be preserved for reuse afterwards. Some of the benefits of implementing macros in a design include the ability to reuse blocks in other designs, and the ability to design more effectively as a team.

Macros contain logic or physical netlist (placed or routed) and is packaged as an IPM format for reuse. After a macro has been created, it is no longer parameterizable.

Macro Types:

There are three types of macro preservation levels:

1. **Logical (Logic Macro)**
   - Logic Macro is defined using Pre-Synthesis Constraint Editor.
   - Placement regions for Logic Macros are not required. However, if you want the macro to be placed at a specific area in both macro creation and reuse, a placement region is required.
   - Placement region is constrained inside Macro block but not fixed.
   - Routing region is constrained inside Macro block or Buffer Zone (if present) but not fixed.
   - Macro logic preserved.
   - Logic Macro can be exported after synthesis.
2. **Logical & Physical with Place Info (Firm Macro)**
   - Firm Macro is defined using Physical Designer and after running Map Design (MAP).
   - Placement regions are constraints for placement. They may not be good for achieving better performance as the macro has connections to I/O ports and other instances placed far away. However, you need to specify the region for the following cases:
     - Create Firm Macro as IP for other designs.
     - Create Firm Macro to be reused in other designs.
   - Routing regions for Firm Macros are not required.
   - Firm Macro can only be exported after Place and Route.
3. **Logic & Physical with Placement & Routing Info (Hard Macro)**
   - Hard Macro is defined using Physical Designer and after running Place & Route Design (PAR).
   - Placement region is required for Hard Macros.
   - You need to create routing regions for the following cases:
     - Create Hard Macro as IP for other designs.
     - Create Hard Macro to be reused in other designs.

- Create Hard Macro for functional safety applications.
- Hard Macro can only be exported after Place and Route.

Macro Creation:

During macro creation, the macro is implemented as a design block and gets instantiated in the RTL design. The design can be used to contain additional logic to test macro functionality. Unified Design Database (.udb) supports the design block to be a macro.

- **Macro Module** – Module in the design database which implements macro netlist.
- **Macro Instance –** An instance in the design database where a macro module is instantiated.

The **ldc_create_macro** constraint defines the macro instance during the macro creation phase. It is expected that the instance name provided by the constraint can be found by the potentially hierarchical path specified in the constraint. If found, the module of the instance will be synthesized as macro during the synthesis run.

You can also define regions to guide placement and routing for a macro.

- **Placement Region** – Physical boundary of placed macro.
- **Routing Region** – Zone surrounding a macro region for additional resource for routing.

Routing region is always equal to or larger than placement region. It can be defined as extra buffer on left/right/top/bottom sides compared to placement region.

Export Macro:

Modules representing macros will be extracted from the design and exported as a .ipm file. The macro will be the top module in the .ipm file. The name specified in **ldc_create_macro** constraint will be used as the name of the top module.

Macro Reuse:

During macro reuse, you need to add the IPM files for each of the macros to be reused. These provide black box descriptions for the blocks which can be instantiated in the reuse design. A compatibility check is performed between the macro block and the current design target during import.

This section lists the options available in Block-Based Design.

Topics include:

- Creating a Macro Block
- Creating a Macro Region
- Exporting Macro

- Reusing a Macro Block
- Macro Usage Guidelines

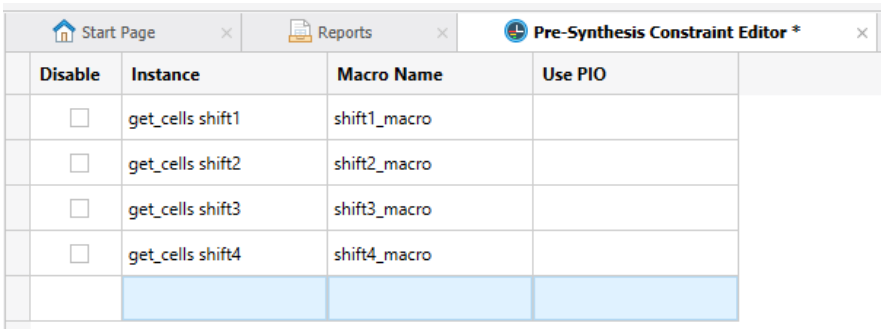
To get some hands-on experience, try the Lattice Radiant Block-Based Design Tutorial.

Creating a Macro Block

Macros must be created prior to synthesis because they are defined at logic instances. For macro creation, you can use a new or an existing project.

To create a macro block:

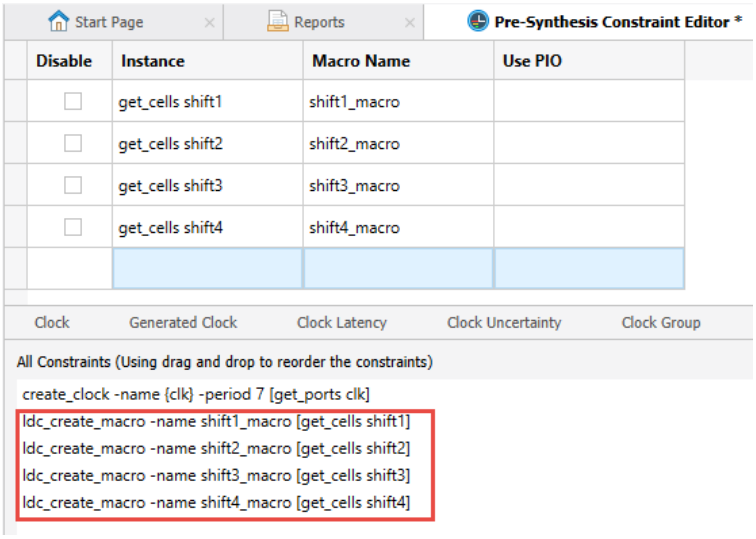
- Click **Tools** > **Pre-Synthesis Constraint Editor** > **Macro** tab.



The Macro tab contains the following columns:

- **Instance** – The Instance column specifies the hierarchical path of the instance which is defined as a macro. It specifies only one instance in a constraint. These constraints define the macro blocks.
- **Macro Name** – The name that is automatically set for the macro instance. The text boxes under the Macro Name column are editable.

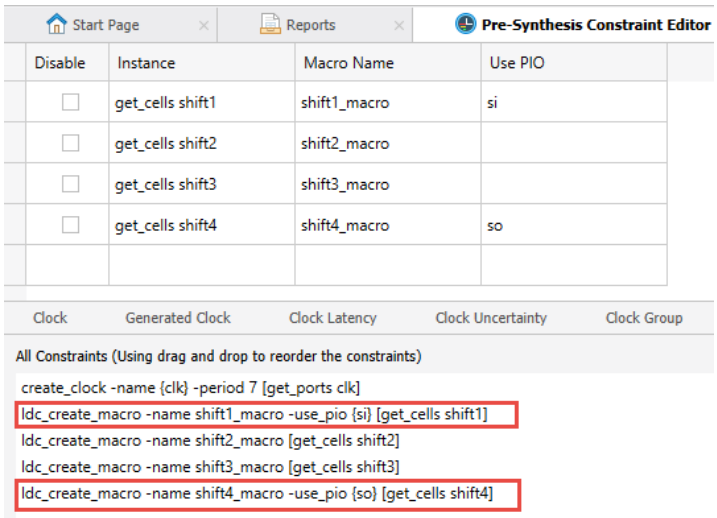
  When **Instance** and **Macro Name** have been added, the command line will be updated as follows:



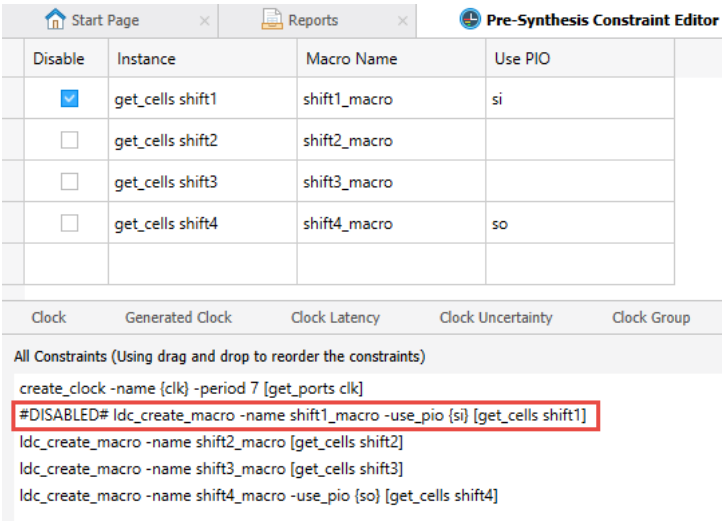
The **ldc_create_macro** constraint command tells the logic synthesis tool which instances are macros.

- **Use PIO** – The Use PIO column specifies the ports of the selected instance to define which ports need to use PIO so that it will be included inside the macro. If you do not manually specify ports, the PIO in the macro module will be automatically inferred during reuse.

  When you add ports to the Use PIO column, the command line will be updated as follows:



- **Disable** – If you do not want to synthesize macro blocks in a design containing macro constraints, the constraints can be disabled by selecting the checkbox in this column. You will see the #DISABLED# appended in front of ldc_create_macro constraint.



After saving the changes in **Pre-Synthesis Constraint Editor**, the defined macro is stored in pre-synthesis constraints (.sdc) file for logic synthesis.

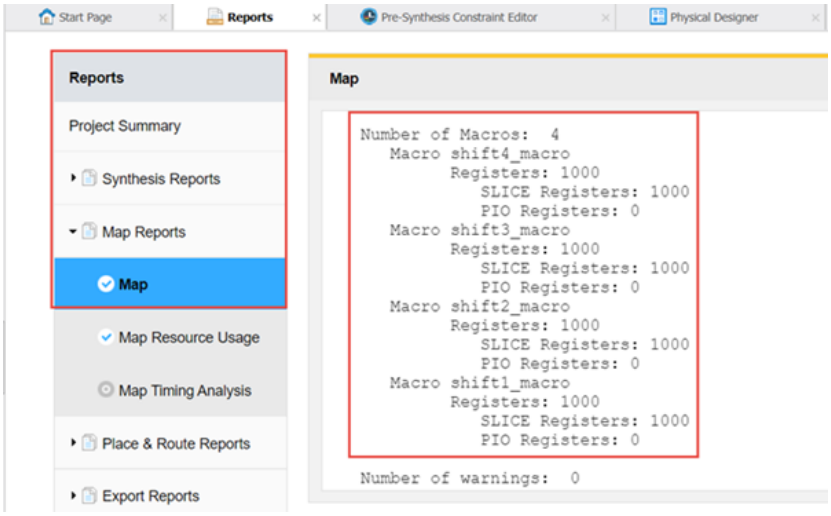
You can also implement macros to a design by adding the **ldc_create_macro** constraint to the active .sdc file.

Note:

By default, Radiant tools produce hierarchical paths with a "/" (forward slash) as separator. However, Synplify Pro uses a "." (period). In order for the flow to work, any hierarchical path needs to be adjusted for Synplify by either setting **set_hierarchy_separator {/}** to the constraint file or by manually changing any hierarchical path created in Radiant to use "." instead of "/."

To check Macro information in Map Report:

- After running MAP, you can view information about the created macros in the **Design Summary** section of the Map Report. The details in this report during the creation stage are based on the **ldc_create_macro** command in the .sdc file.
- This section displays the total number of resources used by the created macro (e.g., LUT, SLICE Register, PIO Register, and Block RAM).



Creating a Macro Region

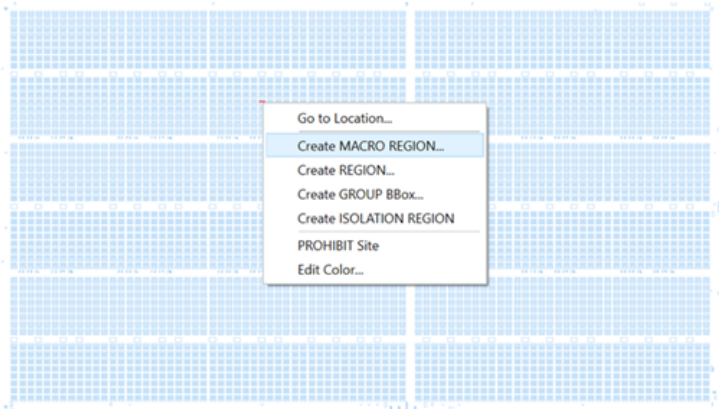
Once macro is created, you can create a macro region in **Physical Designer** (optional).

- If you already know the macro region, you can add the **ldc_create_region** constraint to the post-synthesis constraint (.pdc) file, or start **Physical Designer** to create a macro region before running **MAP**.
- If you want to use the placement result to guide the region creation, you can run **PAR** first. After running **PAR**, open **Physical Designer**, and highlight the macro instance from the netlist.

  Based on the macro placement result, you can create a macro region for the selected macro. The regions will be saved to the .pdc file. As a result, you need to rerun **MAP** and **PAR** to use the specified macro regions.

To create a macro region:

- In **Physical Designer**, right-click on the floorplan layout and select the **Create MACRO REGION…** option.



The **Create New MACRO REGION** dialog box opens.

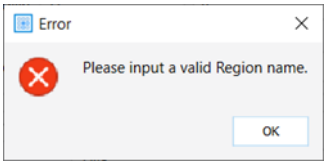


The dialog box contains the following fields:

- **Macro Region Name** – The name that you set for the macro region.

  When specifying the macro region name, you can use any name you like.

  This option is mandatory. You may encounter the following error if you do not set the macro region name.



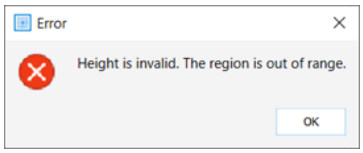
- **Anchor** – Anchor of the placement region used to place the macro.
- **BBox** – Width and height of the placement region.

  When you enter a value in the Anchor and BBox fields, the range of the following items might also change:

  - Macro Resource (LUTs, REGs, EBRs, DSPs)
  - Buffer Size range
  - BBox range

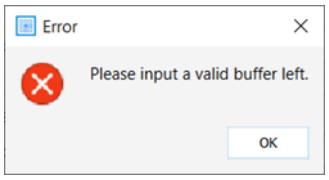
  The range changes depending on the value that you enter in the Anchor and BBox fields.

  If you do not enter a value within range in Anchor and BBox fields, you may encounter the following error.

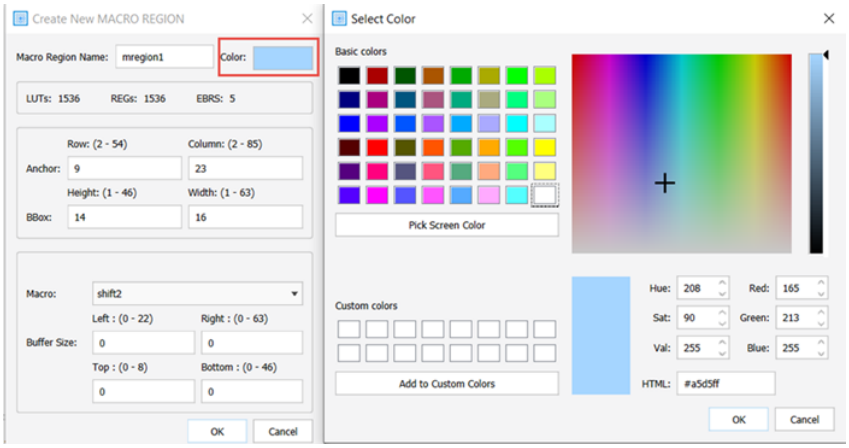


- **Macro dropdown menu** – You can select the predefined macro instance from the pre-synthesis macro creation.
- **Buffer size** – Buffer zone of the routing region. It gives PAR an extra room to do the routing. The routing is confined within the macro region and its surrounding ring.
  - **Buffer left** – Left buffer zone of the routing region.
  - **Buffer right** – Right buffer zone of the routing region.
  - **Buffer top** – Top buffer zone of the routing region.
  - **Buffer bottom** – Bottom buffer zone of the routing region.

  If you do not enter a value within range, you may encounter an error. The following example shows a buffer size error.



- **Color option** – Allows you to change the color of the macro region.



After creating a macro region, click the **Save** icon in **Physical Designer**. The created macro regions are saved into a post-synthesis constraint (.pdc) file.

```
ldc_create_region -name mregion1 -site R4C2D -width 13 -height 13 -buffer_left 1 -buffer_right 1 -buffer_top 1 -buffer_bottom 1
ldc_set_location -region mregion1 [get_cells shift1]

ldc_create_region -name mregion2 -site R22C20D -width 13 -height 13 -buffer_left 1 -buffer_right 1 -buffer_top 1 -buffer_bottom 1
ldc_set_location -region mregion2 [get_cells shift2]

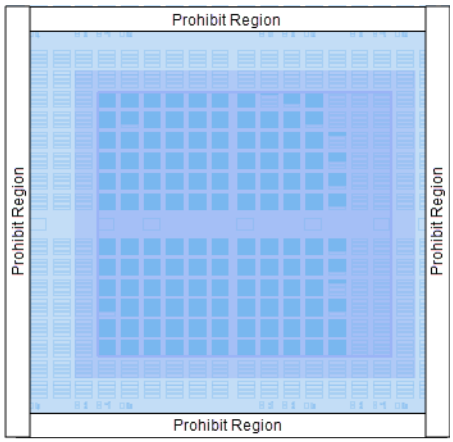
ldc_create_region -name mregion3 -site R39C44D -width 13 -height 13 -buffer_left 1 -buffer_right 1 -buffer_top 1 -buffer_bottom 1
ldc_set_location -region mregion3 [get_cells shift3]
```

The **ldc_create_region** constraint is designed to define placement region. It supports additional information of routing region. In the example above, -buffer_left/-buffer_right/-buffer_top/-buffer_bottom (optional) are used to define routing region. Without these options, routing region is identical to placement region.

Once macro regions are defined, you can assign a macro instance to the region by using **ldc_set_location** constraint.

To create an isolated region:

- In **Physical Designer**, right-click on the floorplan layout and select the **Create ISOLATION REGION** option.
  - Draw a rectangle around the sites where you want to place the isolated region.



  - Physical Designer automatically creates prohibited regions on each side. Once prohibited regions are created, the content of the macro can only be accessed by approved signals from the main design.
  - The inner macro locks placement and routing. It also excludes any other logic from using the region for placement or routing.
  - The outer macro is set as empty macro, no logic or routing can be used inside of it.
- If a gap exists between isolation and secure region, some design instances may be placed within it by PAR. The gap will not be considered as an empty/secure region. You need to manually cover the prohibited region by expanding it.
- You can expand the isolated regions by doing the following:
  - Double-clicking the region that you want to expand and changing the BBox field values in the Edit Region Property dialog box.

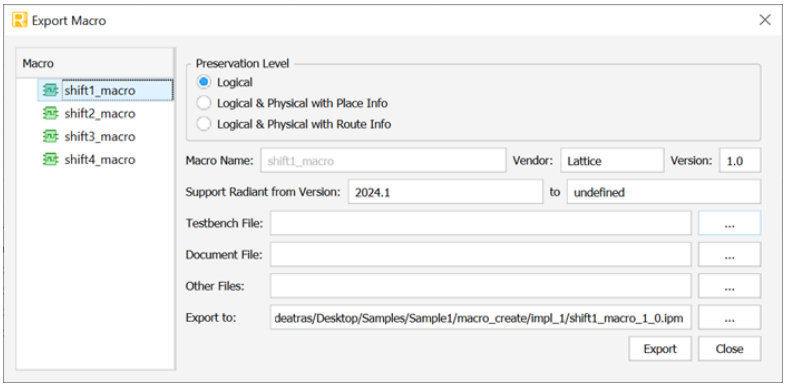
- Selecting the region that you want to expand and dragging it to the desired size.
- Editing the height or width of the isolated region in the ldc_create_region constraint within the .pdc file.

Exporting Macro

Once synthesis is passed, you can launch the Export Macro dialog box from the Tools menu. The dialog box lists all defined macros on the left-hand pane. You can select them and set their preservation levels. When macro is exported, a .ipm file will be generated.

To export macro:

- Click **Tools > Export Macro**.
- In the **Export Macro** dialog box, you can choose the macro preservation levels and specify the default path where the macro package will be exported.



The macro data can be exported according to a specified preservation level. During macro reuse, you can set the preservation levels to a lower value.

Logical (Logic Macro)

- After running **Synthesize Design** and **MAP**, you can only export to **Logical** preservation level.

Logical & Physical with Place Info (Firm Macro)

- For Firm Macro, you can only export to **Logical** and **Logical & Physical with Place Info** preservation levels after running **Place & Route Design**.

Logic & Physical with Placement & Routing Info (Hard Macro)

- After running **PAR**, you can export to **Logical**, **Logical & Physical with Place Info**, and **Logic & Physical with Placement & Routing Info** preservation levels.

The Export Macro dialog box also contains the following fields:

- **Macro Name** – Displays the name of the selected macro from the Macro column in the left-hand pane.
- **Vendor** – Displays the vendor name.
- **Version** – Displays the Macro IP version, the default is always 1.0.
- **Support Radiant from Version to Version** – Displays the supported maximum and minimum Radiant version. The maximum version could be empty, but the minimum is the same as the current Radiant version.
- **Testbench File** – Testbenches that allow you to do simulation or evaluate the macro (optional).
- **Document File** – Contains any document such as help, user guide, or introduction file included in the macro (optional).
- **Other Files** – Any files such as readme files and other documentation (optional).
- **Export to** – Default path where the macro package will be saved.

After exporting, a macro .ipm package will be generated. It contains the following files:

- **<macro_name>.mac** – Contains general information of the macro project.

  The following example shows the information included in the .mac file.


```
"Device": "LIFCL-40",
"Device Package": "CABGA400",
"Family": "LIFCL",
"Files": {
    "Constraints": "",
    "Document": "",
    "LogicDB": "shiftreg1_logic.mdb",
    "OtherFiles": "",
    "PlaceDB": "D:\/ut\/shiftregDemo\/jedi_create\/demo\/demo_create\/impl_1\/demo_create_impl_1.udb",
    "RouteDB": "D:\/ut\/shiftregDemo\/jedi_create\/demo\/demo_create\/impl_1\/demo_create_impl_1.udb",
    "Testbench": "",
    "Interface": "shiftreg1_bb.v",
    "SimulationModel": "shiftreg1_sim.vo"
},
"Name": "shiftreg1",
"Preservation": "logic",
"SupportRadiantMax": "undefined",
"SupportRadiantMin": "1.0",
"Vendor": "Lattice",
"Version": "1.0",
"ResourceUsage": {
    "lut": "0",
    "ff": "1000"
}
```


- **<macro_name>_bb.v** – Synthesis header file of the macro project. It is exported to define module interface for synthesis.
- **<macro_name>_bb_extra.v** – Synthesis header file exported with additional information.

  The bb.v and bb_extra.v files may contain the following information:

  - **TYPE** – Specifies the type of the module. Currently, only the "MACRO" type is supported.
  - **syn_allowed_resources** – Allowed resource in the macro region. It is needed to help create a reasonable-sized macro region.
  - **Locked** – The macro with locked attribute is fixed, not movable.

  These information are to be propagated to top design by synthesis tools when the macro is reused as a black box. Tools can get the information before the macro is integrated to the top design.

The following example shows the information included in the bb.v file.

```
module example_macro (inp, outp1, outp2)
/* synthesis TYPE = "MACRO"

            syn_black_box
            black_box_pad_pin = "outp1,outp2"

            syn_resources = "luts=500,regs=400"

*/;
```

The following example shows the information included in the bb_extra.v file.

```
module example_macro (inp, outp1, outp2)
/* synthesis TYPE = "MACRO"

            syn_black_box
            black_box_pad_pin = "outp1,outp2"

            syn_resources = "luts=500,regs=400"
            syn_allowed_resources = "luts=550,regs=450"

        ARCHITECTURE  = "je5d00"
            ANCHOR        = "R9C25"
            BBOX          = "3,7"
            BUFFER_LEFT   = "1"
            BUFFER_RIGHT  = "1"
            BUFFER_TOP    = "1"
            BUFFER_BOTTOM = "2"

            LOCKED
*/;
```

- **<macro_name>_sim.vo** – Exported for macro simulation. The file is generated based on logic macro.
- **<macro_name>.mdb** – Exported Macro in unified constraints database (.udb) format. Modules representing macros will be extracted from design and exported to .ipm files. The macro will be the top module in .ipm file. The name specified in ldc_create_macro constraint will be used as the name of the top module.

Reusing a Macro Block

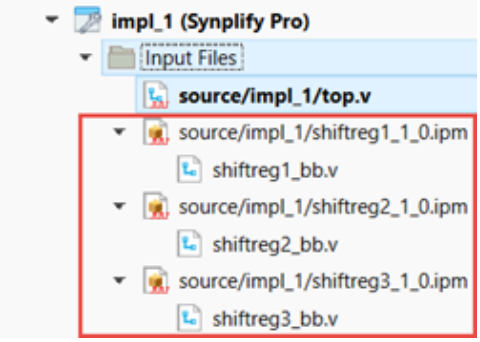
During macro reuse, you need to add the IPM files for each of the macros to be reused. These provide black box descriptions for the blocks which can be instantiated in the reuse design. A compatibility check is performed between the macro block and the current design target during import.

Macros can be reused with different devices of the same family:

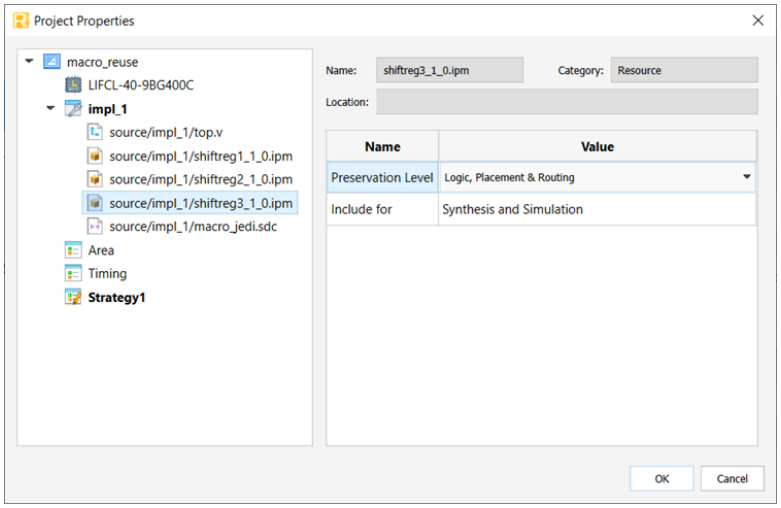
- In general, Logic Macros can be reused in different devices from the same family depending on the preservation level and resources used.
- Firm and Hard Macros can only be reused with the same device.

To reuse a macro within another project:

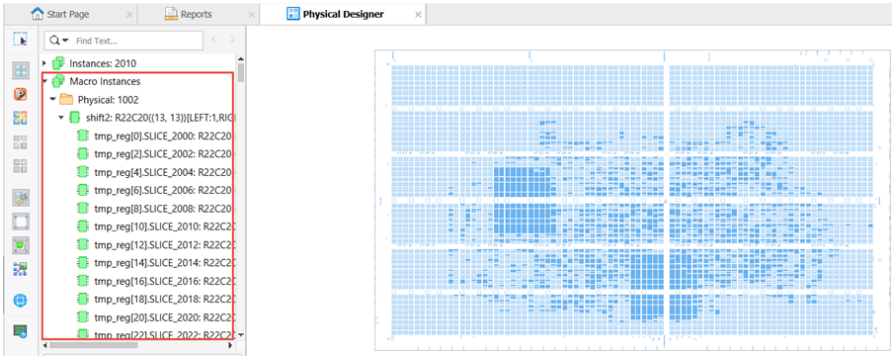
- For macro reuse, the .ipm file for each respective macro that is exported can be used by importing the existing .ipm files into your project. The imported files will be stored under the **Input Files** folder in the Radiant software.



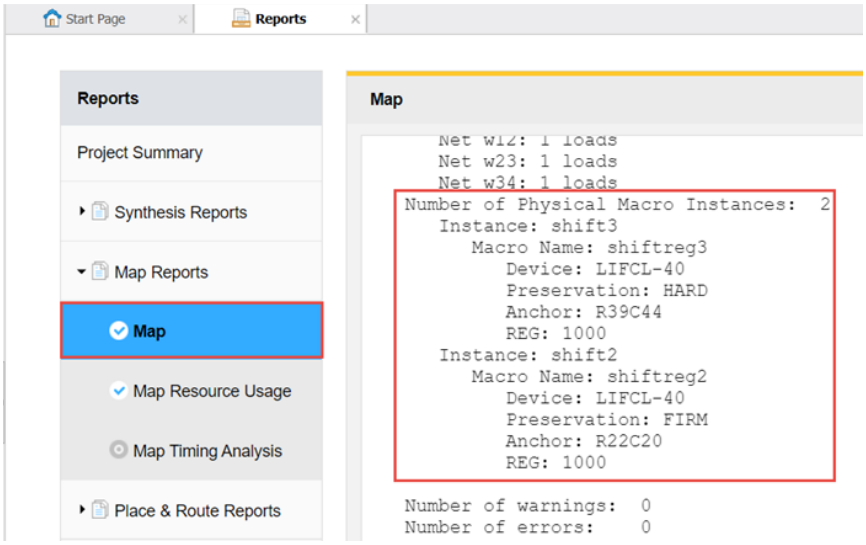
- After running **Synthesize Design**, the post-synthesis Logic Macro is merged into your design.
- You can lower the preservation level of the selected .ipm file in **Project Properties** (optional). Preservation levels must be equal or less than the preservation level when macro is exported.
  - If the imported macro's preservation level is Firm Macro, you can lower it to Logic Macro.
  - If the imported macro's preservation level is Hard Macro, you can lower it to Logic and Firm Macro.



- When you open **Physical Designer**, the netlist shows the reused macro instances. They contain information of the original region from the create stage.



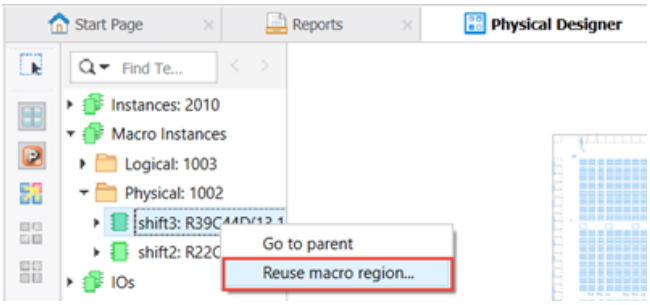
- Run **MAP** with reused macro.
  - Firm Macro – The post-MAP macro design with placement constraint is merged into your design.
  - Hard Macro – The post-MAP macro design with placement and route constraint is merged into your design.
  - After running MAP, you can view information about the reused macros in the **Design Summary** section of the Map Report.
    - This section lists Firm and Hard macro information and attributes such as device family, preservation level, anchor, and main resources.
    - The syn_resources information generated when exporting macro (e.g., LUT, REG, Block RAM, LRAM) determines the main resources listed in the Map Report.

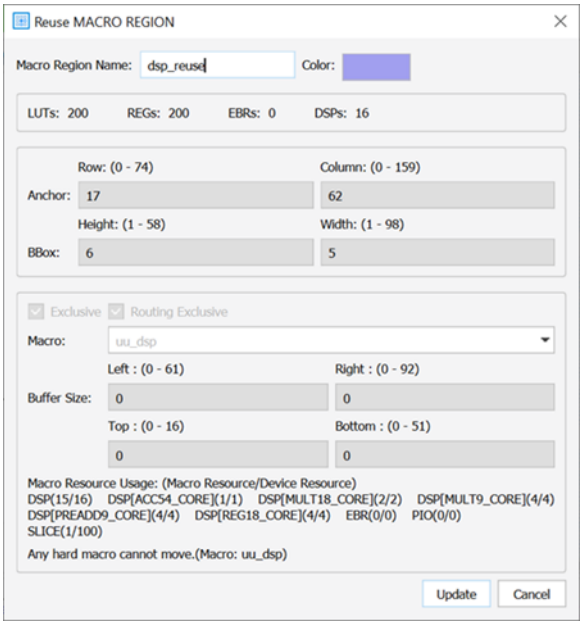


- You can also create a macro region with the Exclusive options in Physical Designer:
  - **Exclusive** – This option excludes other logic from the region for placement and routing when reusing the exported Firm and Hard macro.

- **Routing Exclusive** – This option excludes other logic routing when reusing the exported Firm and Hard macro.

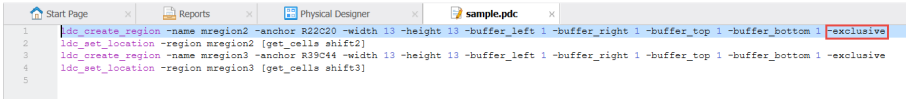
The **Reuse Macro Region** dialog box opens when you right-click on a macro instance from the netlist and select the **Reuse macro region...** option.





- Once done, click the **Save** icon in Physical Designer.

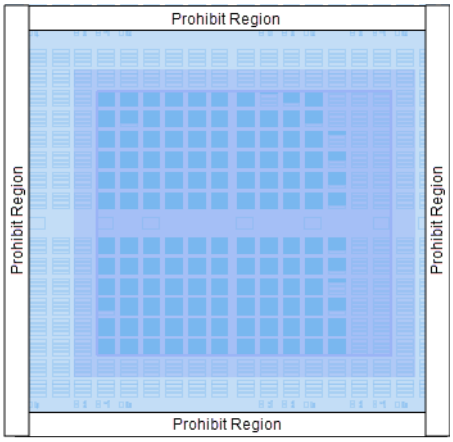
  The created macro regions are saved into a post-synthesis constraint (.pdc) file. The **-exclusive** option will be added to the **ldc_create_region** command line if Exclusive was enabled from the Reuse Macro Region dialog box.



- In **Physical Designer**, you can also right-click on the floorplan layout and select the **Create ISOLATION REGION** option.
  - Draw a rectangle around the sites where you want to place the isolated region.

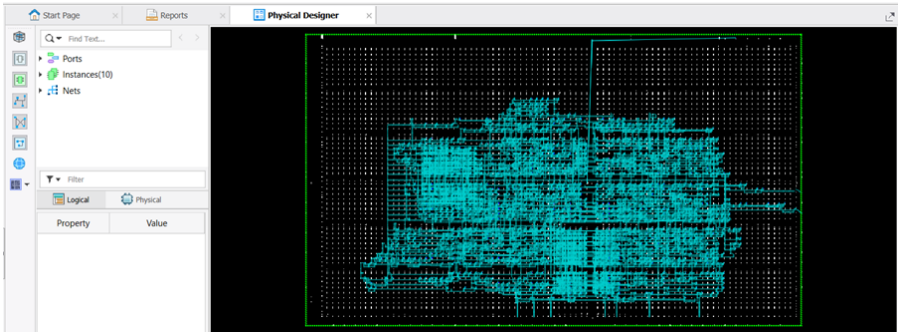
- Physical Designer automatically creates prohibited regions on each side. Once prohibited regions are created, the content of the macro can only be accessed by approved signals from the main design.
- The inner macro locks placement and routing. It also excludes any other logic from using the region for placement or routing.
- The outer macro is set as empty macro, no logic or routing can be used inside of it.

- If a gap exists between isolation and secure region, some design instances may be placed within it by PAR. The gap will not be considered as an empty/secure region. You need to manually cover the prohibited region by expanding it.
- You can expand the isolated regions by doing the following:
  - Double-clicking the region that you want to expand and changing the BBox field values in the Edit Region Property dialog box.
  - Selecting the region that you want to expand and dragging it to the desired size.
  - Editing the height or width of the isolated region in the ldc_create_region constraint within the .pdc file.



- Run **PAR** with reused macro**.**

  After running **PAR**, you can open **Physical Designer** and select **Routing Mode** to see the macro instances in the physical netlist.



Macro Usage Guidelines

Macro creation and reuse have the following guidelines:

- Macro is created from logic instance.
- For multiple macro instances, you need to create a wrapper on top of them.
- Macro creation is instance based. If the whole design is a macro, you need to instantiate it in a top design with a wrapper.
- For macro reuse, macro black box timing model is not supported. Synthesis cannot optimize macro's interface data paths. PAR will need to see full netlist to run timing analysis.
- Macro cannot contain JTAG and PCS hard-IP, which may require merging with others during reuse.
- Place/Route region only supports a single rectangular region.
- A module to be created as a macro block cannot contain black boxes.

Entering the Design

You can enter your design, or describe the logic of our design, in several different ways, and you can use those ways in almost any combination. Radiant software accepts and helps create Verilog, VHDL, and SystemVerilog files. Design files can be created using Synplify Pro.

Setting specifications for controlling hardware implementation of the design, synthesis, or simulation is treated as a separate topic. You may want to start entering your design before setting the constraints.

Verilog, SystemVerilog, and VHDL

The hardware design languages (HDL) Verilog, SystemVerilog, and VHSIC Hardware Description Language (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 Radiant software design project, or you can create or edit them within the Radiant software.

Modules

Modules are functional bits of design that can be re-used wherever that function is needed. To help your design, the Radiant software provides a variety of modules for common functions. They are optimized for Radiant software device architectures and can be customized. Use these modules to speed your design work and to get the most effective results. See “Designing with Soft IP, Modules, and PMI” on page 141.

Structural Verilog

Lattice Synthesis flow by default generates a netlist which is a Structural Verilog (.vm) file. This .vm file can be used by itself as a whole design, or it can be referenced as an IP module to be used in another design. Any Verilog file -- Structural or Behavioral (.v or .vhd) -- can be used for design entry.

HDL Design Entry

The Radiant software 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 built into the Radiant software. 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). 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.
- Use underscores (_) instead of leaving blank spaces in file names.
- Begin file names with an alpha character (letter) rather than a numeric character (number).
- 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 LSE” on page 121.

See Also

- “Running Source Editor” on page 112
- “Viewing Logs and Reports” on page 85

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 file—for example, filename.vhd

- From the Radiant software 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 Radiant software main window, choose **File > New > File**. In the New File dialog, choose Verilog Files, SystemVerilog, or VHDL Files from the Source Files list.
2. In the pop-up New File dialog, fill in the Name and Location, and choose the file extension in the Ext field.
3. Check the **Add to Implementation** 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 Radiant software main window.

Tip

You can detach Source Editor from the Radiant software 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 press and hold the left mouse button. Move the mouse over the text you want to copy. The selected text becomes highlighted in blue.

   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 115, 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 add comment to text:

1. Highlight the text on which you want to add a 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).

A comment explains the purpose of a line or block of code.

Example in Verilog:

```
// Declare a 2D array with 4 rows and 2 columns, each element
is 8 bits wide
reg [7:0] array2D [0:3][0:1];
```

Example in VHDL:

```
signal temp_data : std_logic_vector(7 downto 0);  -- Temporary
signal for data processing
```

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 to enter or edit your variables.

To insert bookmarks in a file:

1. In Source Editor, open the file you want to insert bookmarks into.
2. Click the location in the file you want to insert a bookmark.
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 desired.

To find marks in a file:

1. In Source Editor, open the file where the bookmarks you want to find reside.
2. Choose **Edit > Bookmarks > Previous Bookmark**, or **Edit > Bookmarks > Next Bookmark**.

   The cursor jumps to the bookmark in the direction selected. 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 bookmarks, if desired.

Using Complete Word

When you type a keyword into Source Editor, choose **Edit > Advanced > Complete Word**. A list of the matching values are 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 pressing **Enter**.
- Continue typing the rest of the value. If no matching value is found, nothing will be entered.

Searching and Replacing Tips

Use the following tips when searching or replacing text such as signal names:

- Use **Control + F** to launch the Find and Replace bar with the Find tab enabled. Enter the search term in the **Find** box to find the search term in the Source Editor document.
- Use **Control + H** to launch the Find and Replace bar with the Replace tab enabled. Enter the search term in the **Find** box and the term you wish to replace it with in the Replace With box. Select desired Find Options, and click either **Replace**, **Replace All**, or **Find Next**.

Using Templates

The Source Template tab provides templates for creating VHDL, Verilog, and constraint files, or you can create your own template. Templates increase the speed and accuracy of design entry.

Note:

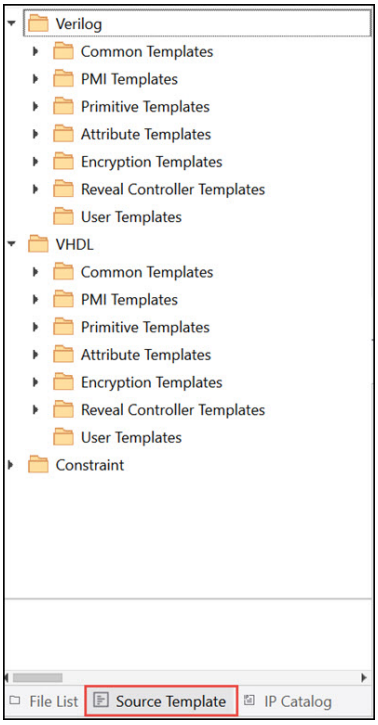
Remember to replace dummy variables in the template with your own logic.

To access templates, click the Source Template tab in the lower left of the Radiant software main window to view and access template files. If the Source Template tab is not visible, click  in the tool bar, or choose **View > Show Views > Source Template**.

Templates are available for:

- Verilog
- VHDL
- Constraint

Figure 4:



The following templates are available for each of the categories:

Table 3: Templates Available in Source Editor

Category	Template Folders
Verilog/VHDL	▶ Common Templates ▶ PMI Templates ▶ Primitive Templates ▶ Attribute Templates ▶ Encryption Templates ▶ Reveal Controller Templates ▶ User Templates
Constraint	▶ Timing Constraints ▶ Physical Constraints ▶ User Templates

For Constraint templates, the Timing Constraints folder and Physical Constraints folder contain 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 821

- “Lattice Synthesis Engine Constraints” on page 750

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 112.
2. Click the Source Template the tab in the lower left of the Radiant software main window to view and access template files. If the Source Template tab is not visible, choose **View > Show Views > Source Template**.

   Two panes open in Source Template tab. The upper pane shows the list of template folders. The lower pane is blank.
3. Place the cursor in your source document 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. Drag and drop to template from the template pane into the Source Editor.

   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 needed, make further changes such as adding code to execute within a case statement.

See Also

- “Instantiating a PMI Component” on page 149

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 how to organize them before you start. There is no easy way to move templates once they have been created.

To create a new template:

1. Click the Source Template the tab in the lower left of the Radiant software main window to view and access template files. If the Source Template tab is not visible, choose **View > Show Views > Source Template**.

Two panes open in Source Template tab. The upper pane shows the list of template folders. The lower pane is blank.

2. In the Source Template tab, 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.
3. 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**.
4. Right-click **User Templates** or one of its sub-folders and choose **New Template**.

   A new template icon appears under the folder.
5. Right-click the new template icon and choose **Rename**. Type in the desired template name and press **Enter**.
6. 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, so once a template is deleted it is permanently gone. So ensure you have chosen the correct template before deleting it.

To delete a template:

1. Click the Source Template tab in the lower left of the Radiant software main window to view and access template files. If the Source Template tab is not visible, choose **View > Show Views > Source Template**.

   Two panes open in Source Template tab. The upper pane shows the list of template folders. The lower pane is blank.
2. In Source Template tab, expand the folders so that the desired folder or template is showing.

   Caution

   Deleting a folder also deletes all templates under the folder.
3. Right-click that template or folder and choose **Delete**.

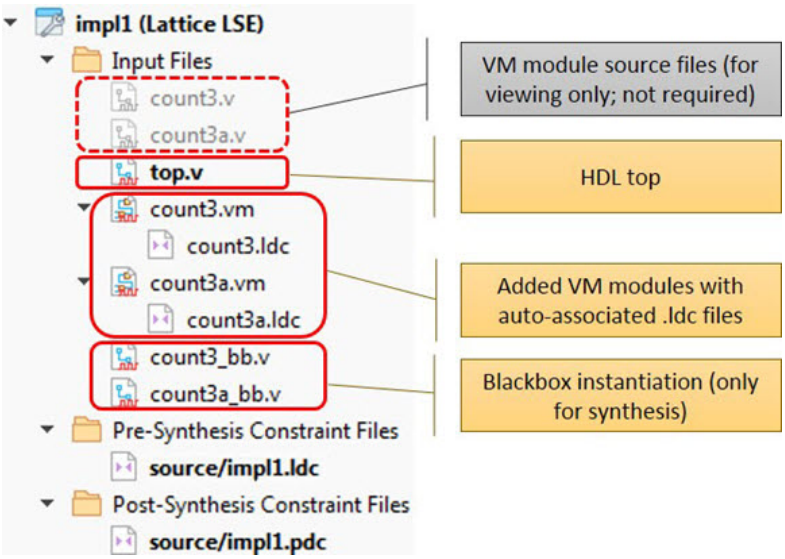
Using Structural Verilog (.vm) in the Design Flow

The Radiant software supports structural Verilog files (.vm) as an input module source for both LSE and Synplify Pro. Structural Verilog design entry is similar to Electronic Design Interchange Format (EDIF) design entry. Benefits of Structural Verilog over EDIF include:

- Allows individual module-level constraints.
- Allows simulation.

The following are Structural Verilog flow requirements:

- The top module must be HDL.
- The .vm and its associated .ldc files must have the same name.
- I/Os cannot be inserted during .vm module synthesis.
- A Verilog Black Box instantiation module (_bb.v) is required for each .vm source file.



How VM Flow Works

The following provides an example of creating and using Structural Verilog (.vm) in the design flow.

1. Create a Radiant software project.
2. Add HDL sources.
3. Add timing constraints and save them to an .ldc file.
4. Synthesize the project without I/O insertion to create the .vm file.
5. Rename .vm and .ldc file to the module name. Repeat steps 2 through 5 until all .vm modules and .ldc files are created.
6. Add .vm files to the main project. The .ldc files are auto-associated.

7. Prepare a Verilog Black Box instantiation module for each .vm file.

Figure 5: Example .vm File

```
module count(clk, rst, count)/* synthesis syn_black_box
syn_declare_black_box=1 */;
input clk, rst;
output [23:0] count;
endmodule
```

   For Synplify Pro, you need to add the **syn_declare_black_box** attribute to the .vm flow. This attribute indicates user-defined black box in Synplify Pro flow.

8. Run the processes as normal after instantiating them from a top level. If processed properly, the Post-Synthesis Timing Constraint Editor will show all constraints for the .vm files.

Coding Tips for LSE

The following are some recommendations if you choose to use LSE to synthesize your design. Tips include how to write code to infer different types of I/O ports, avoiding conflicts between VHDL packages, using VHDL coding, and preparing Verilog blocking assignments.

About VHDL Coding

LSE typically applies the VHDL specification strictly, sometimes more strictly than other synthesis tools. The 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 may fail to synthesize because operators would have multiple definitions. Some synthesis tools may allow this but LSE does not.

Strict Variable Typing

LSE is more strict about variable type requirements than most 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.
```

Mismatched assignments like this will generate errors that stop synthesis.

About Inferring Memory

To produce RAM and ROM in a design, you can write code for the design so that the synthesis tool infers the memory.

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) instead of registers. PFU is used for small memories and EBR for large. 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.

The following sections describe how to write code to infer different kinds of memory with LSE.

See Also

- “Inferring RAM” on page 122
- “Inferring ROM” on page 133

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 when needed. The following lists the rules for coding inferred RAM. Figure 6 on page 123 (Verilog) and Figure 7 on page 124 (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 795.
- Control the RAM with a clock edge and a write enable signal.
- For synchronous reads, see “Inferring RAM with Synchronous Read” on page 125.
- 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 128.
- If desired, assign initial values to the RAM as described in “Initializing Inferred RAM” on page 132.

Figure 6: 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;
  output [data_width-1:0] dout;
  reg [data_width-1:0] dout1; //Register for output.
  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;
    dout1 <= mem[addr]; // Output register controlled by clock.
    dout <= dout1;
  end
 endmodule
```

See Also

- “About Verilog Blocking Assignments” on page 134

Figure 7: 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. The following examples show the simple RAM for Verilog and VHDL modified for synchronous reads. Changes are in bold text.

Verilog Examples

Figure 8: 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] dout1; // Register for output.
  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;
    dout1 = mem[addr]; // Output register controlled by clock.
  dout <=dout1;
  end
endmodule
```

Figure 9: 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 10: 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 11: 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;
   signal raddr : std_logic_vector(addr_width - 1 downto 0);
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 122

Inferring Dual-Port RAM

The following applies for dual-port, pseudo or true RAM.

- When using 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 134.

The examples below are based on the simple RAM of Figure 6 on page 123 (for Verilog) and Figure 7 on page 124 (for VHDL).

Verilog Examples

Figure 12: 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 13: 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 14: 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 15: 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 122

Initializing Inferred RAM

Create initial values for inferred RAM to initialize memory.

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

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.

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 122

Inferring ROM

To code the ROM to be inferred, do the following:

- Define the ROM with a case statement or equivalent IF statements.
- Assign constant values, all with 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 799.

Figure 16: 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 17: 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 assignment. Multiple blocking assignments, such as you might use for true dual-port RAM (see Figure 18 on page 134), or a mix of blocking and non-blocking assignments are not supported. Use non-blocking assignments instead (<=). See Figure 19 on page 135.

Figure 18: 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 19: 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 each block have a different name. Assign names with begin statements. See Figure 20 on page 135.

Figure 20: 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 a type of I/O port, 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

The following can be performed in Source Editor:

- Select a default file language
- Choose window modes (attached or detached).
- Set encoding.
- Configure line number showing.
- Configure text wrapping.
- Set printing options.

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, a specified type of file, or any new files created in Source Editor.

To select a default file language:

1. In Source Editor, choose **View > Language**. The following options are available: C++, Hypertext, Perl, Python, TCL, Verilog, VHDL, EDIF, SystemVerilog, Timing Constraint, and Physical Constraint.
2. Choose the desired language from the list. The language selected becomes the default language in Source Editor.

To attach or detach Source Editor from the Radiant software:

1. To detach Source Editor from the Radiant software main window, click the Detach Tool icon in the upper-right corner of Source Editor.
2. To attach Source Editor to the Radiant software main window, choose **Window > Attach Window**, or click on the Attach Tool icon in the upper-right corner of Source Editor.

Customizing Source Editor from the Radiant software Main Window

To customize Source Editor settings in the Radiant software main window:

1. In the Radiant software main window, choose **Tools > Options**.
2. In the Tools window, 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 Bookmark Margin: Displays the side bar allowing you 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.

   Restore Returns to the default setting.

Font

Lists the current font used in Source Editor. To change the current font, click the drop-down menu next to Font Family and select the desired font. To change the font size, click the drop-down menu next to Font Size and select the desired size. Click **OK** to exit. The selected font name, font type, and font size will be displayed in Source Editor.

Restore settings.	Returns all selections to their default

3. In the Color window, you can set the default colors for the text, comments, numbers, keywords, and a variety of other options under the Source Editor tab.

   Colors

   Click in the Themes drop-down menu to select either a light or dark overall color theme for Source Editor.

   To select a different color, double-click the color box next to the desired element to open the Select Color window. Click on the new color to use.

Restore settings.	Returns all selections to their default

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 associated files to your own text editor program in the Options dialog. Select **Tools > Options** from the Radiant software main window, click on **General**, and then select the **File Associations** tab.

References

This section includes the following references.

Synthesis Tool User Reference

For more details on the Synopsys Synplify Pro for Lattice tool user interface, commands, and features, refer to the *Synopsys Synplify Pro for Lattice Reference Manual*.

HDL Language Reference

For more details on Verilog and VHDL language support, refer to *Synopsys Synplify Pro for Lattice Language Support Reference Manual*.

If you have installed Synplify Pro for Lattice, these documents are available in the installation directory *\\lscc\radiant\2024.2\synpbase\doc*.

Packaging IP Using IP Packager

IP Packager is a tool for you to create an IP package easily. You can edit ports, files, parameters, and memory in the IP Packager, and pack a customized IP directly.

See Also

- “Running IP Packager and Viewing Documentation” on page 139
- “Installing IP Created with IP Packager into IP Catalog” on page 139

Running IP Packager and Viewing Documentation

IP Packager is a stand-alone tool. IP Packager can be run from both Windows and Linux. IP Packager documentation is contained within the IP Packager tool.

To run IP Packager:

- In Windows, go to the Windows Start menu and choose **Lattice Radiant Software <Version> > IP Packager**.
- In Linux: from the ./<Radiant Software Install Path>/bin/lin64 directory, enter the following on a command line:

  ```
  ./ippack.sh
  ```

The IP Packager tool opens.

To view IP Packager documentation:

- In the IP Packager main window, choose **Help > Help**.

The IP Packager can also be run from the command line. Refer to “IP Packager” on page 1548.

See Also

- “Packaging IP Using IP Packager” on page 139
- “Installing IP Created with IP Packager into IP Catalog” on page 139

Installing IP Created with IP Packager into IP Catalog

You can download and add IP created with IP Packager into IP Catalog.

To download and add a User IP:

1. In the Radiant IP Catalog, click on the **Install a User IP** button.

2. In the **Select user IP package file to install** dialog box, browse to the IP Package (.ipk) file, and click **Open**.

- The Soft IP will be installed into a folder in the user's personal directory. For example: C:/Users/<*username*>/RadiantIPLocal/<*IP_name*>.
- The Soft IP will be added into the IP Catalog.

See Also

- "Packaging IP Using IP Packager" on page 139
- "Running IP Packager and Viewing Documentation" on page 139

Designing with Soft IP, Modules, and PMI

Modules are functional bits of design that can be re-used wherever that function is needed. Creating such components with hardware design languages is common practice. To help your design along, the Radiant software provides a variety of components for common functions. They are optimized for Radiant software device architectures and can be customized. Use these components to speed up your design work and to get the most effective results.

Radiant software components come in a variety of forms:

- Modules: These basic, configurable blocks come with IP Catalog. They provide a variety of functions including I/O, arithmetic, memory, and more. Open IP Catalog to see the full list of what's available. Also see the "Lattice Module Reference Guide" on page 821.
- IP: Intellectual property (IP) are more complex, configurable blocks. They are accessible through IP Catalog, 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 IP Catalog. To see all that's available and to learn about licensing and other vendors of IP, go to the Lattice website: www.latticesemi.com/ip.

IP Catalog provides a variety of functions ranging from the most basic, such as arithmetic and memory, to much more complex functions. With IP Catalog these components can be extensively customized and created as part of a specific project or as a library for multiple projects. See "Creating IP Catalog Components" on page 142.

However, many of these components can also be used with PMI (see next item). To decide which method to use, see "PMI or IP Catalog?" on page 142.

- PMI (Parameterized Module Instantiation) is an alternate way to use some of the components that come with IP Catalog. Instead of using IP Catalog, PMI can directly instantiate a component into your HDL and customize it by setting parameters in the HDL. You may find this easier than using IP Catalog if your design requires many variations of the same component. To decide which method to use, see "PMI or IP Catalog?" on page 142.
- Reference designs provide you with a starting point on creating your own components. Lattice Reference Designs are available in Verilog and VHDL, and can be downloaded from the Lattice website: 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 Radiant software Library Primitives" on page 151.

PMI or IP Catalog?

Many IP Catalog components are also available as PMI modules, but PMI doesn't offer error checking. This and the next two sections discuss the pros and cons of both options to help you decide which is best for your project.

PMI is a convenient way to use components of the same type but that vary from instance to instance. This eliminates the need to create a separate component for each instance using IP Catalog.

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 IP Catalog to generate different components for each variation and then insert the instantiation template for each of them. In this situation, you might find the PMI method easier and faster.

Before deciding whether to use a PMI module, compare it to the equivalent IP Catalog module. Often IP Catalog provides more options than PMI does. IP Catalog also assures that all of your option selections and parameter settings are legal. PMI offers no error checking.

See Also

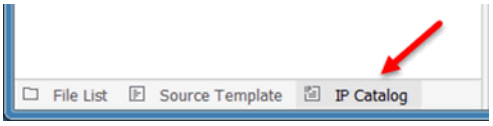
- "Parameterized Module Instantiation (PMI)" on page 694
- "Creating IP Catalog Components" on page 142
- "Using PMI" on page 148

Creating IP Catalog Components

IP Catalog is an easy way to use a collection of functional blocks from the Radiant software. There are two types of components available through IP Catalog: modules and IP. IP Catalog enables you to extensively customize these components. They can be created as part of a specific project or as a library for multiple projects.

Overview of the IP Catalog Process Below are the basic steps of using IP Catalog modules and IP. For details of performing these steps, see the following topics.

1. Open IP Catalog. IP Catalog is accessed via a tab at the lower left of the Radiant software. Click the tab, or click in the tool bar, to view the list of available components.



2. Customize the component. These modules and IP can be extensively customized for your design. The options range from setting the width of a data bus to selecting features in a communications protocol.

3. Generate the component and bring its .ipx file into your project. After generating the component, select the option “Insert to project.” This will automatically bring the .ipx file into your project after the generation step completes. If you do not select this option, add the .ipx file to your project as you would with any other source file (such as a Verilog or VHDL file) after the generation is complete.
4. Instantiate the component into the project’s design. An HDL instance template is generated during the generation step to simplify this step.
5. IP Catalog modules and IP can be further modified or updated later. After the .ipx file has been added to the Radiant software project, it is visible in the project’s file list. Double-clicking the .ipx file brings up the component’s configuration dialog box where changes can be made and the generation process repeated.

See Also

- “Generating a Component with IP Catalog” on page 145
- “IP Catalog Output Files for Components” on page 146
- “Importing an IP Catalog Component into a Project” on page 146
- “Instantiating an IP Catalog Component” on page 147
- “PMI Modules” on page 822

Downloading IP from IP on Server

You can use the IP on Server tab of IP Catalog to download and install the latest IP available from Lattice Semiconductor. Before you can download any IP, ensure that you have an Internet connection. If you haven’t already, choose **Tools > Options > General > Network Settings** and fill in the dialog box.

The website has links to other vendors of IP. To see all available vendor IP and learn about licensing, go to the Lattice Web site: www.latticesemi.com/ip.

To download Lattice IP on Server:

1. In IP Catalog, click the **IP on Server** tab, located at the upper-left of the tool. The software connects to the Lattice Semiconductor Web site.
2. Click the refresh icon to update the list.
3. Expand the folder tree and select the IP you want to download.

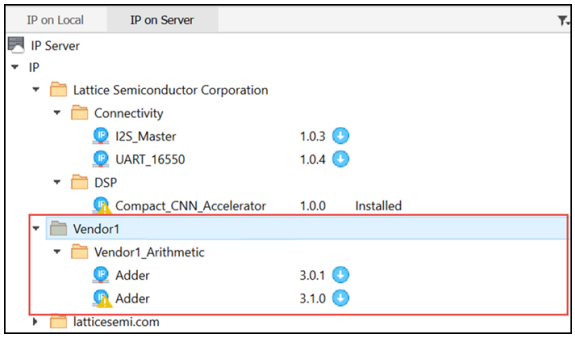
   Information about the IP appears in the right pane including links for additional information.

   IP that are compatible with your selected device have icons highlighted in dark blue . IP that are not compatible with your version of Radiant software have blue icons with a yellow exclamation point .

   IP that are not supported by the device used in the current design have gray icons with a red box bearing a cross sign .

Look for device support information in the right pane.

4. To download the IP, click the IP and choose to the right of the IP name in the list.
5. The downloaded IP is now added to the IP list in the **IP on Local** tab.
6. If you have a third-party IP installed, it will appear under a sub-folder in the **IP on Server** tab.



Note

A new version of an IP that you installed from the server to your local machine may become available.

See Also

- “Lattice Module Reference Guide” on page 821

Using IP Catalog Search Features

IP Catalog has search features that allow you to search the list of IP and modules by keywords, and also search by by type.

To use the IP Catalog search feature:

1. IP Catalog, click the icon in the upper right.
2. Search by bus type by clicking one or more of the ABH-Lite, APB, or AS14-Stream boxes. The list of modules and IP will display modules that support these bus types.
3. Search by keyword to display module names or IP that contain the keyword in the name. For example, typing “adder” will display all modules that have adder in the module name.

See Also

- “Lattice Module Reference Guide” on page 821

Generating a Component with IP Catalog

Use IP Catalog to generate a customized functional block from any module or installed IP.

To generate a component:

1. In the Module/IP tree, select the component that you want to generate.
2. To get more information about the component, in either the IP on Local tab, or IP on Server tab, click on the module or IP, and then click on the blue question mark . The IP Information page will appear on the right.
3. Double-click the component, or click the Generate Module/IP Instance button, located at the upper-left of the tool the In the Module/IP Block Wizard, specify general project information and the base file name for the component.
   - Component Name is the base name for the component's files (that is, with no extension).
   - Create In is the location for the customized component's files.
4. Click **Next.**

   The component's dialog box opens.
5. In the dialog box, select your desired options.When generating a PLL, click **Calculate** to calculate divider settings and actual output frequencies based on CLKI and desired frequencies.

   The IP User Guide is available on the bottom left of the dialog box. You may download the IPUG from the website by clicking this link.
6. Click **Generate**.
7. To automatically import the .ipx file into your project when the component is generated, select **Insert to project** in the Check Generating Result dialog box.
8. Click **Finish**.

See Also

- "IP Catalog Output Files for Components" on page 146
- "Lattice Module Reference Guide" on page 821

Regenerating IPs

The regenerate function is an easy way to update your IP without needing to invoke the IP Catalog tool.

If your design was created in an older version of Radiant software, or if your design was created for a different Lattice device, your IP may need to be regenerated. Regenerating IP will update the IP to the current version.

You can regenerate one or regenerate all IPs to update device information or apply a new version to an existing .ipx file.

To regenerate as single IP:

1. In the Radiant software file list, right-click the IP file you wish to regenerate.
2. In the dropdown menu, choose **Regenerate**.

To regenerate all IP in your design:

1. In the Radiant software file list, right-click anywhere in the File List.
2. In the dropdown menu, choose **Regenerate All IPs...**. A dialog box will pop up to allow you to select one or more IP to regenerate.

See Also

- “Lattice Module Reference Guide” on page 821

IP Catalog Output Files for Components

IP Catalog creates the following output files for components under the specified Project Path. The <*file_name*> comes from the File Name specified in the Module/IP Block Wizard.

IP Catalog creates some different files for IP. These are documented in the IP’s associated user guide.

Table 4:

File Name	Description
<*file_name*>.ipx	Manifest file. This file is loaded into IP Catalog so that modifications can be made to the component.
<*file_name*>.tmpl.v	Instantiation template for Verilog netlist.
<*file_name*>.v	Verilog HDL file for both synthesis and simulation. Verilog output files declare implicit wire types.
tb_top.v	Testbench for associated component.

See Also

- “Lattice Module Reference Guide” on page 821

Importing an IP Catalog Component into a Project

After generating component source files using IP Catalog, you can import the component by importing the IP Catalog manifest file (.ipx). Components 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 component.

Importing the component may not be necessary if the “Import IPX to project” option was selected when the component was generated.

To import a component:

1. In the File List view, right-click the implementation folder ( ) and choose **Add > Existing File**.
2. Browse for the customized component's .ipx file, <*file_name*>**.ipx**, and select it.
3. Click **Add**.

   The .ipx file is added to the File List view.
4. Check the Output Panel for error messages. If the component is not targeted for the current device, try double-clicking the file to regenerate the component.

After importing the component, you can further customize it for your design project by changing options specific to the component. You can also update older components or IP to the latest version.

See Also

- "Lattice Module Reference Guide" on page 821

Instantiating an IP Catalog Component

IP Catalog components are instantiated the same way other components are in your HDL. When you generate components in IP Catalog, the tool also generates Verilog and VHDL reference templates. You can instantiate the reference templates for implementation or simulation flow of any design, as needed.

Before instantiating any IP Catalog component, check that it is compatible with your design project's device. When they were created, the components were optimized for a specific device and may depend on that device's architecture. A component may not work with a different type of device and may need to be regenerated.

Before running any processes on the design, make sure that the IP Catalog component has been imported into the project. See "Importing an IP Catalog Component into a Project" on page 146.

The instantiation file is located in the folder in which the component was created. The file name is based on the component's name: <*component name*>_tmpl.v or <*component name*>_tmpl.vhd.

You can instantiate the IP Catalog Module or IP in one of the following ways:

1. If you selected the option of not inserting the generated IP into your project, browse to the <*component name*>_tmpl.v or <*component name*>_tmpl.vhd, open the file in a text editor, copy the text, and paste into your source file. Then, add an instance and signal names to the component ports.
2. If you selected the option of inserting the generated IP into your project, the IP component configuration file (.ipx) appears in your design

implementation. You can instantiate the generated component or IP as follows:

- In the File List, right-click the component or IP .ipx file, and choose the instantiation command.
  - For Verilog, choose **Copy Verilog Instantiation**, and paste it into your source file. Then, add an instance and signal names to the component ports.
  - For VHDL, choose **Copy VHDL Component**, and paste it into your source file; then choose **Copy VHDL Instantiation**, and paste it into your source file. Then, add an instance and signal names to the component ports.

See Also

- “Lattice Module Reference Guide” on page 821

Using PMI

As mentioned earlier, PMI is an alternate way to use some of the components that come with IP Catalog. Instead of using IP Catalog, you directly instantiate a component into your HDL and customize it by setting parameters in the HDL using PMI. You may find this easier than using IP Catalog if your design requires many variations of the same component.

See Also

- “PMI or IP Catalog?” on page 142
- “PMI Process Steps” on page 148
- “Instantiating a PMI Component” on page 149
- PMI Modules

PMI Process Steps

This section provides a brief listing of the process steps taken when using PMI.

1. Using Source Template, insert PMI components in the HDL source files. This is the recommended method because it includes the PMI instance templates, which require minimal editing.
2. If using stand-alone synthesis and simulation tools, add the PMI soft IP from the <install_path>/ip/pmi/ directory.
3. Perform the remaining implementation steps as you would normally.

See Also

- “Lattice Module Reference Guide” on page 821

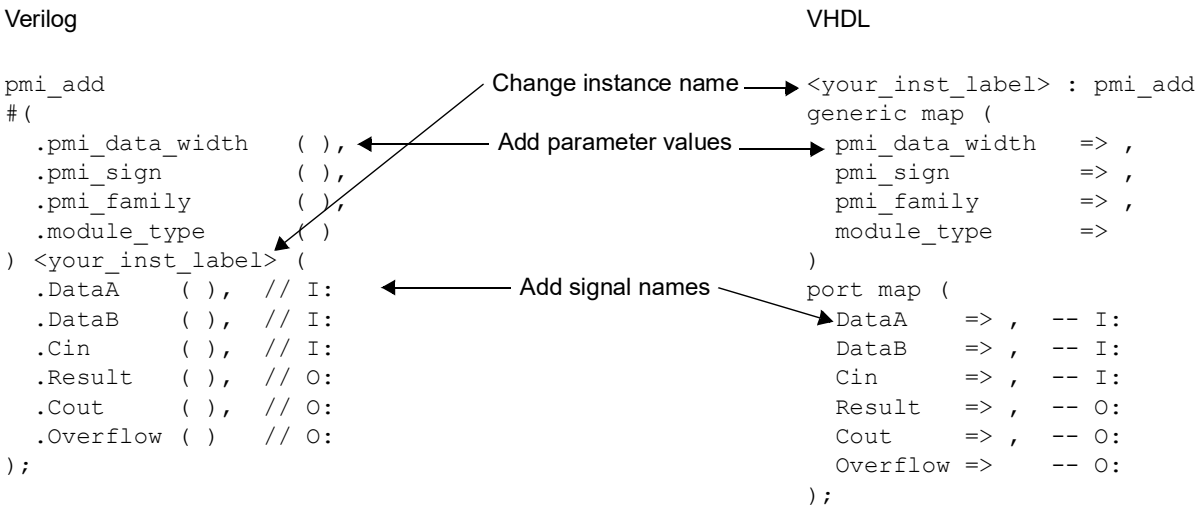
Instantiating a PMI Component

PMI components are instantiated the same way other components are in your HDL. The Radiant software provides a template for the Verilog or VHDL instantiation command that specifies the customized component's ports and parameters. Customize the component by changing its parameters.

To instantiate a PMI component:

1. Click **Tools > Source Template > Verilog/VHDL > PMI Templates**.
2. With Source Editor, open the source file that will receive the component.
3. Drag and drop the text into your source file.
4. Add an instance name, set parameter values, and add signal names to the corresponding component ports in the instantiation command. Refer to the PMI Instantiation Command Example. For information about allowed parameter values, see the section on the component in the "Lattice Module Reference Guide" on page 821.

Figure 21: PMI Instantiation Command Example



5. Save and close the source file.

See Also

- "Lattice Module Reference Guide" on page 821

Requirements for Simulation and Synthesis with PMI

The PMI module code is a black box definition of an IP Catalog module. For this reason, simulation models for these blocks must be available to supplement the code for simulation.

VHDL designs have a few additional requirements for both synthesis and simulation.

PMI Simulation Models When using ModelSim/QuestaSim, or any other simulator, you must first compile the simulation library (pmi_work) from the sources, which are in <*install_dir*>/ip/pmi, where <*install_dir*> is the installation folder of the Radiant software.

To compile PMI library, run the following command in ModelSim/ QuestaSim TCL console:

```
vlib pmi_work

vlog -work pmi_work  <install_dir>/ip/pmi/pmi.v
```

Note

The PMI library name should be “pmi_work” as it will be used in Simulation Wizard .f file to launch ModelSim/QuestaSim.

VHDL Requirements When performing simulation for VHDL PMI, you also need to compile the PMI component declaration file into your work library. The file locations for each device are listed in the following table:

Table 5: PMI Files

Device	Location
iCE40UP	<*install_dir*>/ip/pmi/pmi_iCE40UP.vhd
LAV-AT	<*install_dir*>/ip/pmi/pmi_lav-at.vhd
LFCPNX	<*install_dir*>/ip/pmi/pmi_lfcpnx.vhd
LFD2NX	<*install_dir*>/ip/pmi/pmi_lfd2nx.vhd
LFMXO5	<*install_dir*>/ip/pmi/pmi_lfmxo5.vhd
LIFCL	<*install_dir*>/ip/pmi/pmi_lifcl.vhd
UT24C	<*install_dir*>/ip/pmi/pmi_ut24c.vhd
UT24CP	<*install_dir*>/ip/pmi/pmi_ut24cp.vhd

Using the PMI header as a library in VHDL is acceptable, but it is not required by the synthesis tools. While the lines below may be added at the top of the source file, these are not required. You can still run synthesis and compile for simulation without these lines.

```
library pmi;
use pmi.components.all;
```

The first line includes the PMI components defined in the header file. The second line is used by some simulators for successful library binding.

See Also

- “Lattice Module Reference Guide” on page 821

Designing with Radiant software Library Primitives

Any Radiant software library primitive described in the “FPGA Libraries Reference Guide” on page 824 can be instantiated as a Verilog component or VHDL component in your RTL design. This “gate-level” design approach 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 IP Catalog to generate components that are built with Radiant software library primitives.

To minimize the amount of code overhead required to design using a library primitive, the Radiant software provides a Verilog and VHDL synthesis header library file for each major FPGA device family. Refer to the “Lattice Synthesis Header Libraries” on page 313 topic for details. The component is generally treated as a “black box”, which causes the synthesis tool to pass instances of the library primitive into the target netlist untouched.

The primitive libraries contain descriptions, pin outs, and schematic diagrams of all library primitives for Radiant software FPGA libraries.

Securing the Design

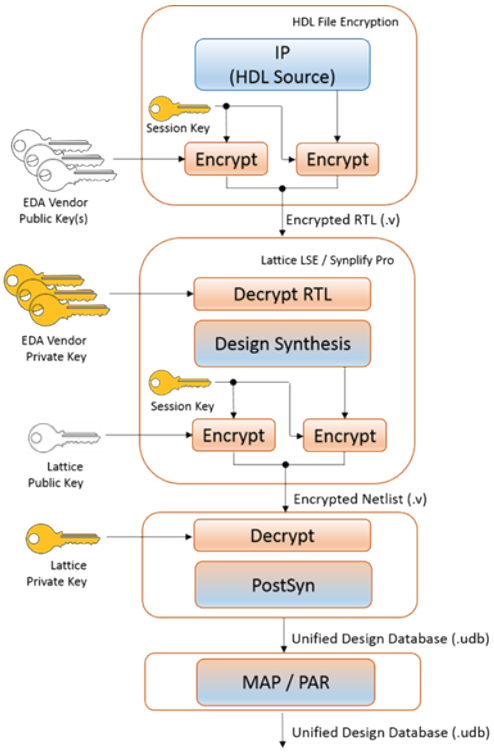
The Radiant software provides the IP security based on the *Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP)* of the IEEE 1735-2014 version 1 standard.

The implementation enables securing IP while ensuring the design flow and the interoperability across different EDA tools. Upon entering the design into Radiant software, IP author selects the desired level of security by encrypting HDL source files. The set security level propagates through the design and impacts the visibility of secured objects in various Radiant software tools. The current version supports encryption of VHDL and Verilog HDL files. Bitstream Security Settings tool generates various encryption key sets. IP Consumer has the flexibility to use the generated key sets during the bitstream generation.

The Radiant software encrypted IP can be processed by other third-party EDA tools including Aldec Active-HDL, Cadence NCSim, Mentor Graphics ModelSim/QuestaSim, Synopsys Synplify Pro, and Synopsys VCS simulator.

IP Encryption Flow

IP encryption flow enables you to protect your IP design. Following the industry standard, the Radiant software, through the IP encryption flow, allows the partnership between the IP Vendor, supported EDA vendor, and Lattice.



The encryption flow uses symmetric and asymmetric encryption methods to maximize the design security. The symmetric method only involves a single symmetric key for both, encryption and decryption. The asymmetric method involves the public-private key pair. The public key is published by a vendor and is used by the Radiant software. The private key is never revealed to the public.

The Radiant software supports these cryptographic algorithms:

- AES-128/AES-256: symmetric algorithm used to encrypt the content of the HDL source file.
- RSA-2048: asymmetric algorithm used to obfuscate a key used in HDL file encryption. The RSA is defined by the public-private key pair. You must be familiar with both keys in order to perform RSA decryption.

HDL File Encryption Flow

The current software version supports encryption of a single HDL source file per a single command.

The overall HDL file encryption flow is summarized in these steps:

- The source file of the IP design is AES encrypted using a symmetric Session key. The AES encryption uses CBC-128 or CBC-256 algorithm. In the source files, this section is referred to as a data block.
- The Session key is RSA encrypted using the vendor's Public Key. In the source files, this section is referred to as a key block. Multiple key blocks may be present in the source file.

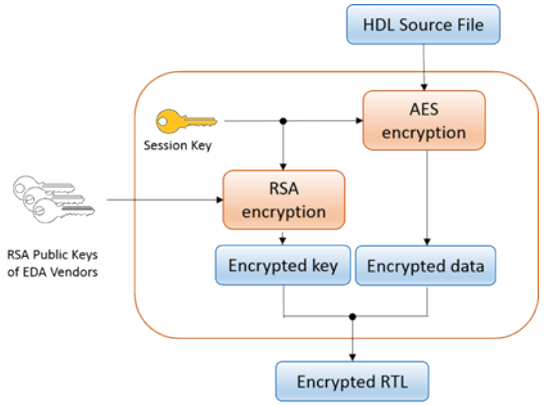
- The encrypted Session key and the encrypted design are merged to file generally named the Encrypted RTL

Each encrypted source file contains one or more data blocks. Each data block is associated with one or more key blocks. The number of key blocks depends on the number of provided vendor's public key.

NOTE

To decrypt an encrypted source file, you must perform the IP encryption flow steps in the reverse order.

During the next step in the design flow, typically a synthesis, the Encrypted RTL is decrypted to access the original IP design, as shown in the following figure.



By separating the encryption of data and key, you can use public keys from different vendors to encrypt same HDL file.

For more information on how to perform HDL encryption, refer to "Running HDL Encryption from the Command Line" on page 1481.

See Also "HDL File Encryption Steps" on page 154

- "Secure Objects in the Design" on page 159

HDL File Encryption Steps

This section provides step-by-step instructions on how to encrypt an HDL file.

The Radiant software provides the key templates you can simply drag-and-drop into an HDL file. Each key template is specific to an EDA vendor providing the value of a public key.

To view the templates in the Radiant software, choose **Source Template** > **Verilog** > **Encryption Templates** and select the EDA specific key template.

Currently, the Radiant software supports these encryption templates:

- Lattice Semiconductor
- Synplicity-1
- Synplicity-2
- Mentor
- Synopsys
- Aldec
- Cadence
- Combined Sample: provides an example of file holding multiple keys.

Step 1: Prepare the HDL file

Annotate the HDL source file with protected pragmas. Protected pragmas provide information regarding type of the key used to encrypt the HDL file, the name of the key, and the encryption algorithm.

In this example, HDL source file will be encrypted by the Lattice Public Key.

```
`pragma protect version=1
`pragma protect encoding=(enctype="base64")

// optional information
`pragma protect author="<Your_Name>"
`pragma protect author_info="<Your_Information>"
```

Step 2: Specify the portion of HDL source file that shall be encrypted

Annotate the HDL file to specify the encryption. Only the portion defined within these protected pragmas is encrypted.

```
`pragma protect begin
// HDL portion that shall be encrypted
`pragma protect end
```

Step 3: Prepare Key

Define the key with which the HDL file should be encrypted. Each key definition must contain the following information:

- key_keyowner: specify the owner of the key
- key_keyname: specify the name of the key. Same owner may provide multiple keys.
- key_keymethod: specify the used cryptographic algorithm. Current version supports RSA algorithm.
- key_public_key: speficy the exact value of the key.

The key definition can be done in two ways:

Defining the key in the key.txt file:

The public encryption key or keys can be defined in any .txt file. The key file may contain a single public key or a list of all available public keys. In the Radiant software, all common EDA vendor public keys are located in *<Radiant_installed_directory>/ispfpga/data/key.txt* file. You can also use the source template to get the encryption templates.

The following is an example of Lattice Public Key defined in key.txt file:

```
`pragma protect key_keyowner= "Lattice Semiconductor"
`pragma protect key_keyname= "LSCC_RADIANT_2"
`pragma protect key_method="rsa"
`pragma protect key_public_key
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0EZKUUhbuB6vSsc7OhQJ
iNAWJR5unW/OWp/LFI71eAl3s9bOYE2OlKdxbai+ndIeo8xFt2btxetUzuR6Srvh
xR2Sj9BbW1QToo2u8JfzD3X7AmRvlwKRX8708DPo4LDHZMA3qh0kfDDWkp2Eausf
LzE2cVxgq7fy/bDhUeN8xKQCSKJ7aguG6kOI6ROoZz21ljzDLUQzhm2qYF8SpU1o
tD8/uw53wLfSuhR3MBOB++xcn2imvSLqdgHWuhX6CtZIx5CD4y8inCbcLy/0Qrf6
sdTN5SAg2OZhjeNdzmqSWqhL2JTDw+Ou2fWzhEd0i/HN0y4NMr6h9fNn8nqxRyE7
IwIDAQAB
```

Defining the key directly in the HDL file:

You may define the Public Key directly in the HDL file. You may define one or more keys.

```
`pragma protect key_keyowner= "Lattice Semiconductor"
`pragma protect key_keyname= "LSCC_RADIANT_2"
`pragma protect key_method="rsa"
`pragma protect key_public_key
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0EZKUUhbuB6vSsc7OhQJ
iNAWJR5unW/OWp/LFI71eAl3s9bOYE2OlKdxbai+ndIeo8xFt2btxetUzuR6Srvh
xR2Sj9BbW1QToo2u8JfzD3X7AmRvlwKRX8708DPo4LDHZMA3qh0kfDDWkp2Eausf
LzE2cVxgq7fy/bDhUeN8xKQCSKJ7aguG6kOI6ROoZz21ljzDLUQzhm2qYF8SpU1o
tD8/uw53wLfSuhR3MBOB++xcn2imvSLqdgHWuhX6CtZIx5CD4y8inCbcLy/0Qrf6
sdTN5SAg2OZhjeNdzmqSWqhL2JTDw+Ou2fWzhEd0i/HN0y4NMr6h9fNn8nqxRyE7
IwIDAQAB
```

If the key is defined directly in the HDL file, you don't need to provide `-k` option in `encrypt_hdl` command.

Note

The key defined directly in HDL source file has preference over the key defined in the key.txt file.

Step 4: Select the encryption algorithm for data encryption

The Radiant software supports both, a 128-bit and a 256-bit advanced encryption standard (AES) with CBC mode. Select the type of algorithm by defining one of the two options. The default is set to 256-bit AES with CBC mode.

```
`pragma protect data_method="aes128-cbc"
```

or

```
`pragma protect data_method="aes256-cbc"
```

Step 5: Run 'encrypt_hdl' Tcl Command

In the Tcl console window, type in the command to encrypt an HDL file. The option '`-k`' may or may not be used depending the location of the key file. The language selection '`-l`' and creation of new output file '`-o`' are optional; selects Verilog by default. If you don't specify the output file, the tool generates a new output file named <input_file_name>_enc.v.

Note

If you encounter an error such as 'ERROR - Can't open '<file_name>.v', you should point to the correct file path and perform the encrypt_hdl command.

If key was defined in the key.txt file:

The command will encrypt the HDL file with all keys defined in the key.txt file.

```
encrypt_hdl -k <keyfile> -l <verilog | vhdl> -o <output_file>
<input_file>
```

If key was defined directly in the HDL file:

```
encrypt_hdl -l <verilog | vhdl> -o <output_file> <input_file>
```

The encrypted file is located at the path specified in the `encrypt_hdl` command.

Step 6: Activate the encrypted HDL source file in Project File

In the Radiant software File List, add the generated file into the project. Right-click on the encrypted file and select **Include in Implementation**.

To view example of Verilog or VHDL pragma annotated HDL source file, visit "Defining Pragmas" on page 1484.

See Also

- "Running HDL Encryption from the Command Line" on page 1481

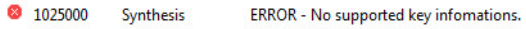
Synthesizing Encrypted IP

The Radiant software supports Lattice Synthesis Engine (LSE) and Synplify Pro synthesis tool. To synthesize an encrypted design, the synthesis tool must decrypt the encrypted design prior the synthesis. Depending on the selected synthesis tool, LSE or Synplify Pro, you must provide the valid vendor's public key to process this step.

You may observe these common issues due to the flow obfuscation:

The design is encrypted with an incorrect key

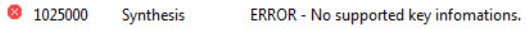
The user observes an error message and synthesis process fails.



The error appears when there is a mismatch between a key used to encrypt the IP design and the vendor's public key.

The design is missing an encryption key

The user observes an error message and synthesis process fails.



The error appears when the user encrypts the IP with only Synplify Pro Public Key. During the synthesis step, given you have selected the Synplify Pro synthesis tool, the design is decrypted and synthesized as expected.

If source HDL files are encrypted, the Synplify Pro result is also encrypted. To decrypt it in Postsyn, the Synplify Pro result should be encrypted by the Lattice Public Key, such as LSCC_RADIANT_3. Otherwise, Postsyn is not able to decrypt.

When running the post-synthesis flow, the Lattice Public Key must be present to properly decrypt the synthesized IP. You must add the missing public key to the key file or directly define it in the HDL source file and rerun the encryption tool.

Note

It is recommended to always define and encrypt your source files with both, Lattice and Synplify Pro public keys. This ensures that your design is processed through the full Radiant software flow independent of the selection of the synthesis tool.

See Also

- "Key File" on page 1488

Cross-probing in Encrypted Design

The Radiant software supports a path cross-probing between Netlist Analyzer, Placement Mode, and Routing Mode.

In the **Reports** tab, view any analysis report and identify a path to view. If cross-probing is available, the specific icon tools become visible, as shown in the following figure.

Figure 22: Available tools for Path Cross-probing



Click on the icon and the specific tool opens with the selected path view.

Encrypted objects are not visible and you cannot cross-probe them. In the case of an encrypted design, the path cross-probing is unable to view. The message “Cannot open encrypted design.” appears.

Note:

Cross-probing to Netlist Analyzer is available only if selected synthesis tool is Lattice LSE.

Secure Objects in the Design

Radiant software protects all aspects of the IP at all times while allowing the tool to work for its intended purpose. The tool also has an option to correct overprotected part of the design if proven to be unusable without certain visibility, such as external ports of the design.

You can choose the level of the design obfuscation.

- Unsecured design: The design consists of the original HDL source files.
- Partially-secured design: The design consists of mixture of encrypted HDL source file(s) and original HDL source files.
- Fully-secured design: The entire design is encrypted.

When a design is encrypted, the Radiant tools reflect the changes in displaying the secured objects. Secured objects include instances, nets, ports, and pins. Clock related objects are typically not encrypted.

In the case of the post-place & route debug flow in Reveal, the previously routed design is not altered when a new post-PAR project is started. The post-PAR debug cores are synthesized, mapped, and place & routed using unused fabric logic.

Visibility

The input and output names inside of the protected design parts contain no reference to real name and are not recognized as names inside of IP.

Messaging

Design names, RTL identifiers names of IP, and RTL source reference are not referenced in messages.

Report files

Design names, RTL identifiers names of IP, and RTL source reference are not referenced in report files.

GUI/Tcl Interface

Design names, RTL identifiers names of IP, and RTL source reference are not referenced in GUI.

The following tools have been enhanced to support this feature.

- “Hierarchical View” on page 160
- “Netlist Analyzer” on page 161
- “Reveal Inserter” on page 162
- “Timing Constraint Editor” on page 163
- “Device Constraint Editor” on page 164
- “Placement Mode” on page 166
- “Routing Mode” on page 167
- “Power Calculator” on page 168
- “Timing Analyzer” on page 169

See Also

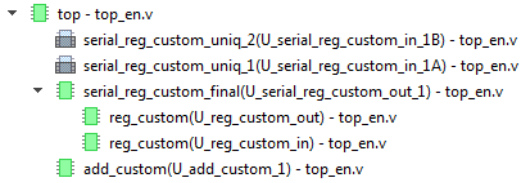
- “Running HDL Encryption from the Command Line” on page 1481

Hierarchical View

Hierarchical view allows you to view the overall design structure.

Unencrypted design allows you to view all design instances and expand all instance’s sub-folders. In partially secured designs, only pragma defined section of an HDL file is encrypted. The Hierarchical view shows the design icon next to the partially encrypted source file. The sub-folders of this file are no longer viewable.

In fully secured design, the entire module is encrypted, and the Hierarchical view shows the secured design icon next to the encrypted source files.The instance is hidden from viewing its sub folder.



See Also

- "Secure Objects in the Design" on page 159

Netlist Analyzer

Netlist Analyzer produces a schematic view of your design. Once design is partially or fully encrypted, the encrypted objects and in some cases, the entire design, become hidden from the schematic view.

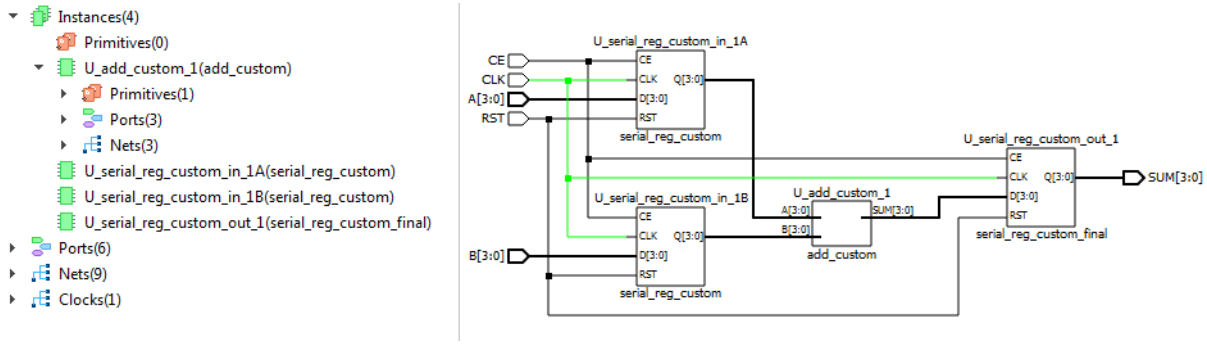
The visibility of secured objects is described below. In unsecured designs, the entire hierarchical structure is visible. In partially secured designs, the ports of encrypted objects are visible; however, you won't be able to view the content and/or sub-structure of a secured object. In fully secured design, the entire design is encrypted; thus, the Netlist Analyzer is unable to produce schematic view and displays an error message instead.

Table 6: Secured Objects in Netlist Analyzer

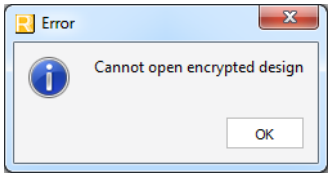
Object Name	Unsecured Design	Partially Secured Design	Fully Secured Design
Instances			
▶ Name	Keeps the original name.	Keeps the original name.	Keeps the original name.
▶ Visibility	Drop-down icon allows to expand and view the sub-directories.	Unable to view or expand its sub-hierarchy.	Unable to view or expand its sub-hierarchy.
Ports	Visible	Visible	Visible
Nets	Visible	Hidden	Hidden

In the following figure, two serial_reg_custom modules are fully encrypted and serial_reg_custom_final module is partially encrypted. You are only able to view the peripheral I/O ports.

Be aware of visual representation of secured and unsecured module.



In specific cases when the entire design is encrypted, the Netlist Analyzer is unable to show the design. Instead, an error message appears.



See Also

- "Secure Objects in the Design" on page 159

Reveal Inserter

Reveal Inserter performs the logic analysis on your design. When design is partially of fully encrypted, the secured signals are not available for viewing.

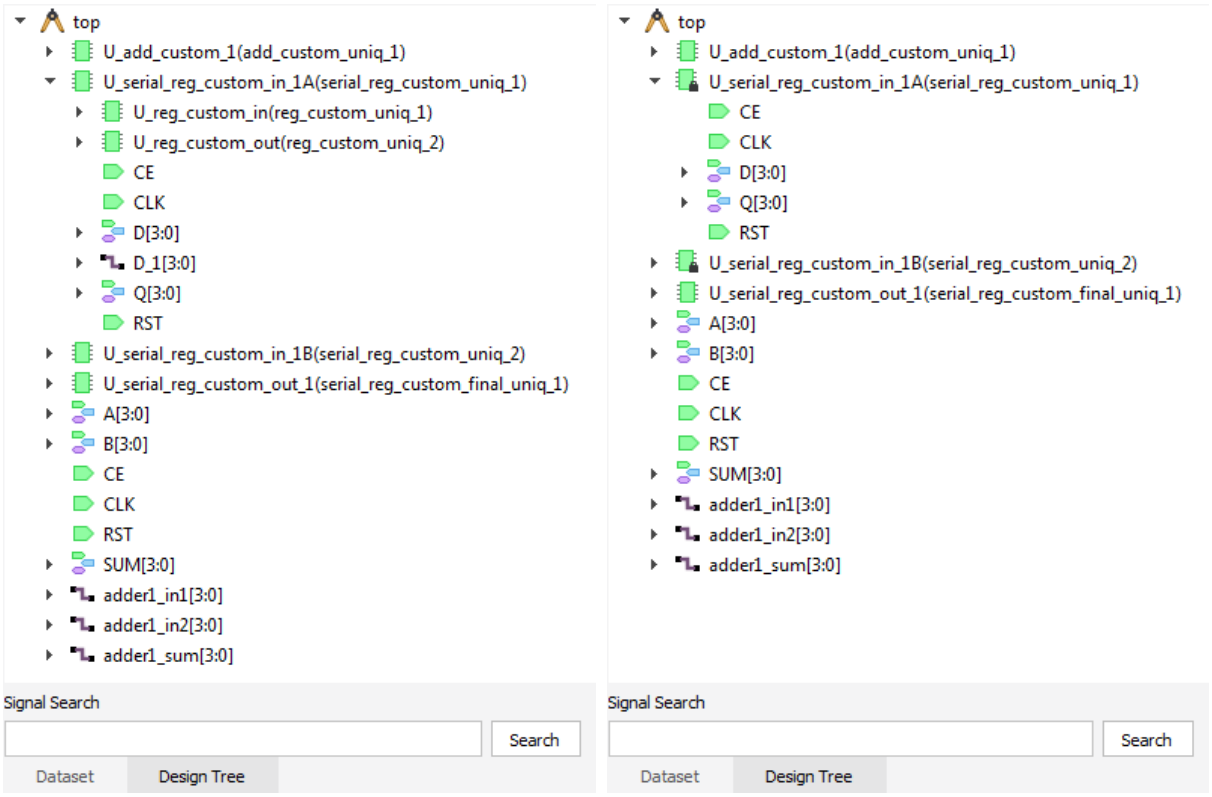
The visibility of secured objects is described below.

Table 7: Secured Objects in Reveal Inserter

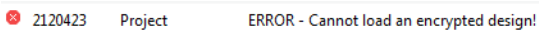
Object Name	Unsecured Design	Partially Secured Design	Fully Secured Design
Instances			
▶ Names	Keeps the original name.	Keeps the original name.	Keeps the original name.
▶ Visibility	Drop-down icon allows to expand and view the sub-directories.	Hidden	Hidden
Ports	Visible	Visible	Visible
Nets	Visible	Hidden	Hidden

The visual representation of unsecured and secured objects visible in Design Tree pane of the Reveal Inserter. As shown in the following figure, the

secured modules are hidden from the hierarchical structure of partially encrypted design.



In specific cases when the entire design is encrypted, the Reveal Inserter is unable to show the design. Instead, an error message appears.



When designing for Nexus and Avant, you can use the Post-Place & Route debug flow to insert debug logic after running place and route. Also referred to as non-intrusive debugging, you can add your debug core without having to alter your original design that has previously completed synthesis, mapping and place & route.

See Also

- “Secure Objects in the Design” on page 159
- “Post-Place & Route Debug Flow” on page 614

Timing Constraint Editor

Timing Constraint Editor is updated to support encrypted design.

In both constraint editors, Pre-Synthesis Constraint Editor and Post-Synthesis Timing Constraint Editor, the tool will not write protected design names into constraints.

Table 8: Secured Objects in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

Object Name	Unsecured Design	Partially Secured Design	Fully Secured Design
Instance	Visible	Hidden	Hidden
Net	Visible	Hidden	Hidden
Pin	Visible	Hidden	Hidden
Clock related Pins, Ports, and Nets	Visible	Visible	Visible
Memory Instance	Visible	Visible	Visible
Modules and Registers used in synthesis attribute constraints	Visible	Visible	Visible

IMPORTANT

The user must not encrypt designs that require to set the constraints at the user's top level.

See Also

- "Secure Objects in the Design" on page 159

Device Constraint Editor

Device Constraint Editor is updated to support encrypted design.

In both constraint editors, Pre-Synthesis Timing Constraint Editor and Post-Synthesis Timing Constraint Editor, the tool will not write protected design names into constraint.

The user must not encrypt designs that require setting the constraints at the top level.

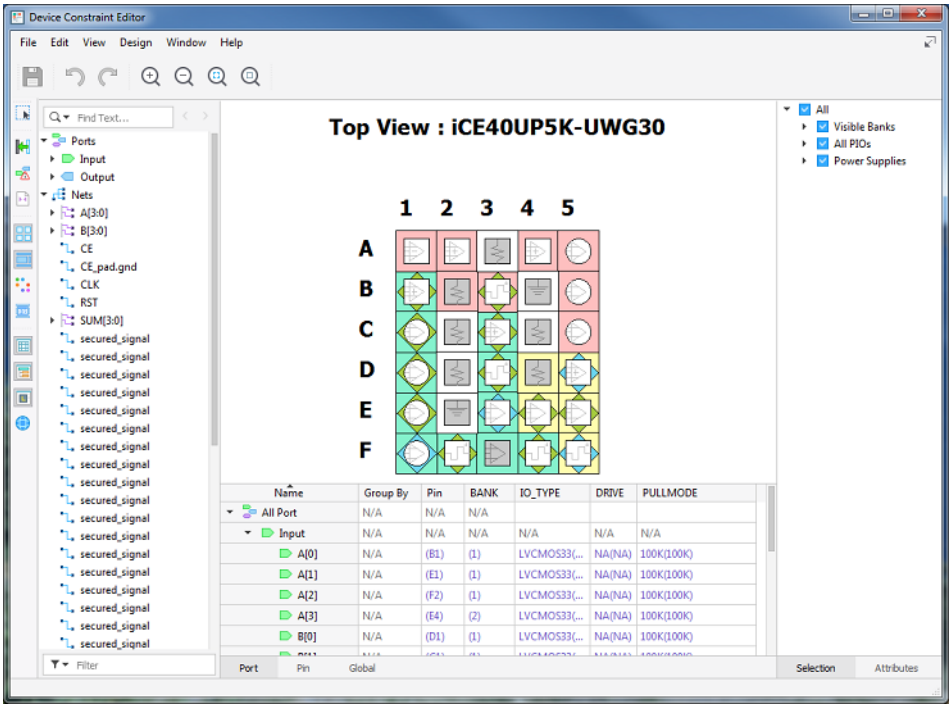
Table 9: Secured Objects in Device Constraint Editor

Object Name	Unsecured Design	Partially Secured Design	Fully Secured Design
Instance	Visible	Hidden	Hidden
Net	Visible	Replaced with 'secured_signal'	Replaced with 'secured_signal'

Table 9: Secured Objects in Device Constraint Editor

Object Name	Unsecured Design	Partially Secured Design	Fully Secured Design
Pin	Visible	Hidden	Hidden
Clock related Pins, Ports, and Nets	Visible	Visible	Visible
Memory Instance	Visible	Visible	Visible
Modules and Registers used in synthesis attribute constraints	Visible	Visible	Visible

In the following figure, the signal names were replaced by ‘secured_signal’ names.



See Also

- “Secure Objects in the Design” on page 159

Physical Designer

The Physical Designer is a combined GUI interface that features:

- “Placement Mode” on page 166 – Provides a large-component layout for your design. When design is partially of fully encrypted, the secured signals are not available for viewing.
- “Routing Mode” on page 167 – Provides a detailed layout of your design. When design is partially or fully encrypted, secure objects are renamed or hidden. The logic block of secured object is not viewable.

The Physical Designer can be accessed through **Tools > Physical Designer** or by clicking on the toolbar icon . Upon opening, the GUI displays the Placement mode.

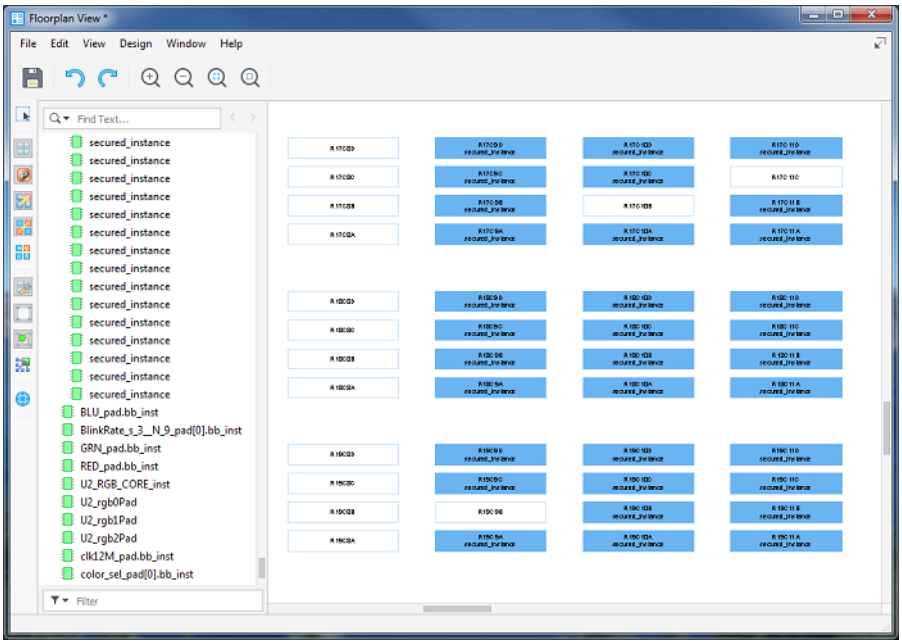
Placement Mode

Placement Mode provides a large-component layout for your design. When design is partially of fully encrypted, the secured signals are not available for viewing. Additionally, the secured components and nets cannot be found using the “Find” dialog box.

Table 10: Secured Objects in Placement Mode

Visibility	Unsecured Design	Partially Secured Design	Fully Secured Design
Component	Visible	Hidden	Hidden
Instance	Visible	Replaced with ‘secured_instance’	Replaced with ‘secured_instance’
Grouping Secured Instances	Allowed	Not allowed. Secured instances are hidden in Available Instances list of the Create GROUP window.	Not allowed. Secured instances are hidden in Available Instances list of the Create GROUP window.
Logic Block View	Enabled	Disabled	Disabled

In following the figure, the signal and instance names were replaced by ‘secured_instance’.



See Also

▶ "Secure Objects in the Design" on page 159

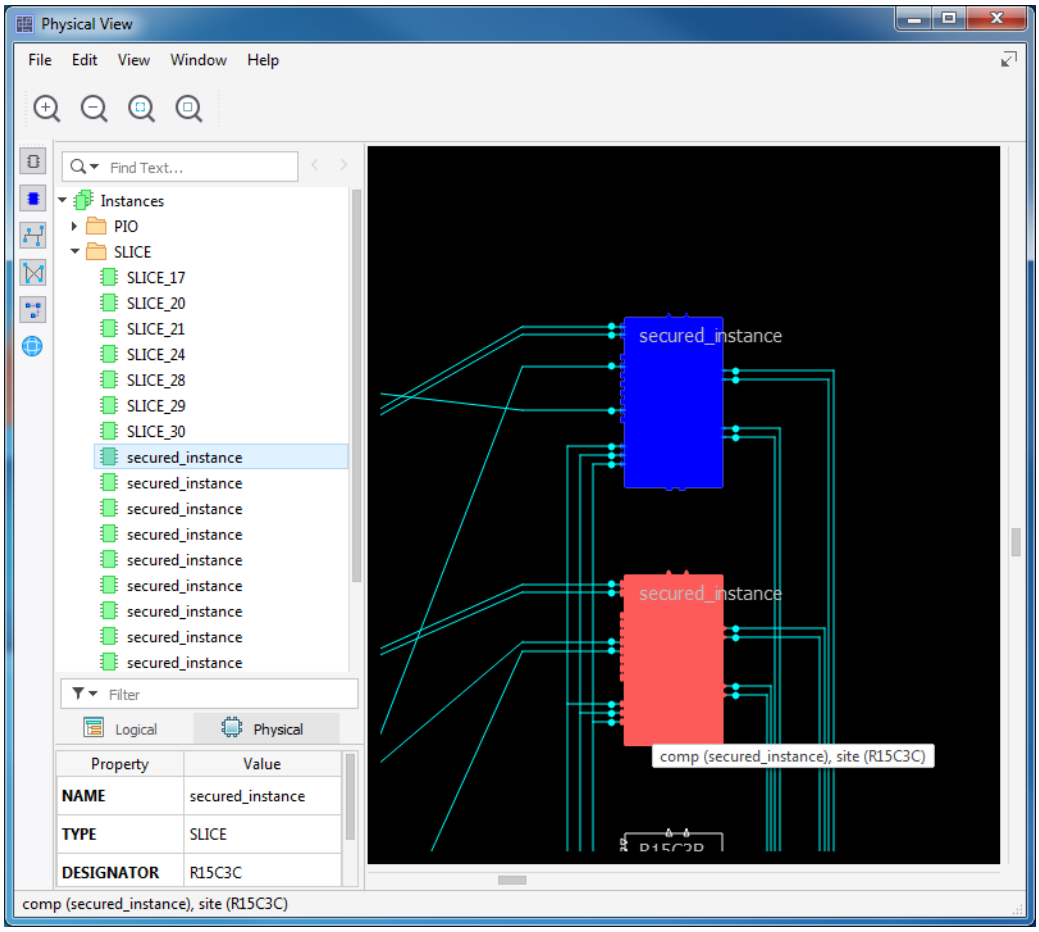
Routing Mode

Routing Mode shows a detailed layout of your design. When design is partially or fully encrypted, secure objects are renamed or hidden. The logic block of secured object is not viewable.

Table 11: Secured Objects in Routing Mode

Visibility	Unsecured Design	Partially Secured Design	Fully Secured Design
Primitive	Visible	Available with a hidden name	Available with a hidden name
Component	Visible	Hidden	Hidden
Instance	Visible	Replaced with 'secured_instance'	Replaced with 'secured_instance'
Nets	Visible	Replaced with 'secured_signal'	Replaced with 'secured_signal'
Grouping Secured Instances	Allowed	Not allowed	Not allowed
Logic Block View	Enabled	Disabled	Disabled

The following figure shows the Routing Mode figure with SLICEs were replaced by ‘secured_instance’ due to encrypted design



See Also

- “Secure Objects in the Design” on page 159

Power Calculator

Power Calculator estimates the power dissipation in a design. Typically, Power Calculator displays the design objects relative to clock. When design is partially or fully encrypted, the encrypted signal names are hidden with the exception of clock related nets.

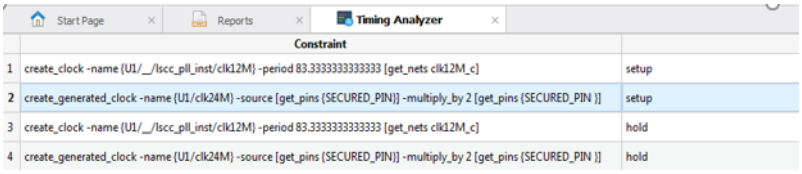
See Also

- “Secure Objects in the Design” on page 159

Timing Analyzer

The timing report file .twr and .twl used by Timing Analyzer is updated to support secured designs.

Constraint Table The secured pin name is replaced with “SECURED_PIN.”



Start Page | Reports | Timing Analyzer

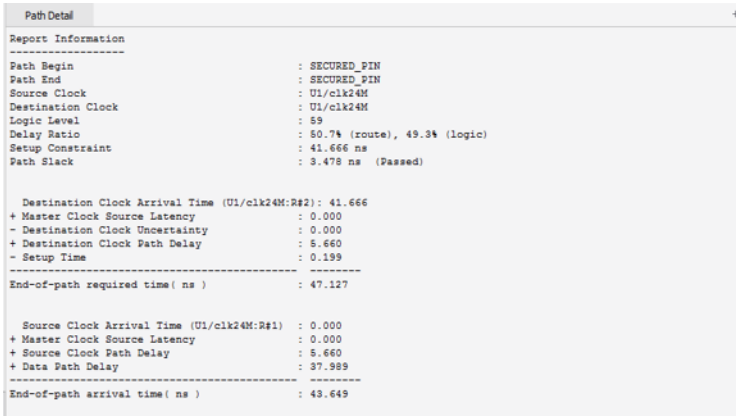
	Constraint	
1	create_clock -name {U1/__/lscc_pll_inst/clk12M} -period 83.3333333333333 [get_nets clk12M_c]	setup
2	create_generated_clock -name {U1/clk24M} -source [get_pins {SECURED_PIN}] -multiply_by 2 [get_pins {SECURED_PIN }]	setup
3	create_clock -name {U1/__/lscc_pll_inst/clk12M} -period 83.3333333333333 [get_nets clk12M_c]	hold
4	create_generated_clock -name {U1/clk24M} -source [get_pins {SECURED_PIN}] -multiply_by 2 [get_pins {SECURED_PIN }]	hold

Paths for All Timing Constraints Table The secured pin name is replaced with “SECURED_PIN.”

Paths summary : 10 timing path(s)

	Start Point	End Point	Setup/Hold Constraint	Slack	Delay	Source Clock	Destination Clock	Analysis Type
1	SECURED_PIN	SECURED_PIN	41.666	3.478	38.188	U1/clk24M	U1/clk24M	setup
2	SECURED_PIN	SECURED_PIN	41.666	3.624	38.042	U1/clk24M	U1/clk24M	setup
3	SECURED_PIN	SECURED_PIN	41.666	4.020	37.646	U1/clk24M	U1/clk24M	setup
4	SECURED_PIN	SECURED_PIN	41.666	4.166	37.500	U1/clk24M	U1/clk24M	setup
5	SECURED_PIN	SECURED_PIN	41.666	4.351	37.315	U1/clk24M	U1/clk24M	setup
6	SECURED_PIN	SECURED_PIN	41.666	4.378	37.288	U1/clk24M	U1/clk24M	setup
7	SECURED_PIN	SECURED_PIN	41.666	4.497	37.169	U1/clk24M	U1/clk24M	setup
8	SECURED_PIN	SECURED_PIN	41.666	4.524	37.142	U1/clk24M	U1/clk24M	setup
9	SECURED_PIN	SECURED_PIN	41.666	4.576	37.090	U1/clk24M	U1/clk24M	setup
10	SECURED_PIN	SECURED_PIN	41.666	4.722	36.944	U1/clk24M	U1/clk24M	setup

Path Detail The secured pin name is replaced by “SECURED_PIN.”



Path Detail

```
Report Information
------------------
Path Begin                                : SECURED_PIN
Path End                                  : SECURED_PIN
Source Clock                              : U1/clk24M
Destination Clock                         : U1/clk24M
Logic Level                               : 59
Delay Ratio                               : 50.7% (route), 49.3% (logic)
Setup Constraint                          : 41.666 ns
Path Slack                                : 3.478 ns  (Passed)


  Destination Clock Arrival Time (U1/clk24M:R#2): 41.666
+ Master Clock Source Latency               : 0.000
- Destination Clock Uncertainty             : 0.000
+ Destination Clock Path Delay              : 5.660
- Setup Time                                : 0.199
-------------------------------------------  --------
End-of-path required time( ns )             : 47.127


  Source Clock Arrival Time (U1/clk24M:R#1)  : 0.000
+ Master Clock Source Latency                : 0.000
+ Source Clock Path Delay                    : 5.660
+ Data Path Delay                            : 37.989
-------------------------------------------  --------
End-of-path arrival time( ns )               : 43.649
```

Data Path Table The secured pin name is replaced by “SECURED_PIN”, secured signal name is replaced by “SECURED_SIGNAL.”

Data Path | Clock Paths

	Name	Cell/Site Name	Delay Name	Delay	Arrival Time	Fanout
1	SECURED_PIN->SECURED_PIN	SLICE_R25C11C	CLK_TO_Q1_DELAY	1.391	7.051	3
2	SECURED_SIGNAL		NET DELAY	2.622	9.673	1
3	SECURED_PIN->SECURED_PIN	SLICE_R24C9C	C0_TO_F0_DELAY	0.450	10.123	1
4	SECURED_SIGNAL		NET DELAY	3.033	13.156	1
5	SECURED_PIN->SECURED_PIN	SLICE_R23C7A	C1_TO_COUT1_DELAY	0.344	13.500	2
6	SECURED_SIGNAL		NET DELAY	0.000	13.500	1
7	SECURED_PIN->SECURED_PIN	SLICE_R23C7B	CIN0_TO_COUT0_DELAY	0.278	13.778	2
8	SECURED_SIGNAL		NET DELAY	0.000	13.778	1
9	SECURED_PIN->SECURED_PIN	SLICE_R23C7B	CIN1_TO_COUT1_DELAY	0.278	14.056	2
10	SECURED_SIGNAL		NET DELAY	0.000	14.056	1
11	SECURED_PIN->SECURED_PIN	SLICE_R23C7C	CIN0_TO_COUT0_DELAY	0.278	14.334	2
12	SECURED_SIGNAL		NET DELAY	0.000	14.334	1
13	SECURED_PIN->SECURED_PIN	SLICE_R23C7C	CIN1_TO_COUT1_DELAY	0.278	14.612	2

See Also

- “Secure Objects in the Design” on page 159

Securing the Bitstream

Bitstream security enables various options of soft and hard IP obfuscations during the bitstream generation. The type of bitstream encryption depends on the key generated in the Bitstream Security Settings tool and the status of your IP license.

If IP license is detected during the Radiant software Bitstream File generation process, the bitstream may be encrypted by user-provided AES key, Bitstream Security Settings tool generated key, or it is not encrypted. Various types of keys for bitstream encryption are generated in “Bitstream Security Settings Tool” on page 171.

If IP license is not detected and you wish to perform the IP evaluation only, before generating the bitstream file, you must first set the IP Evaluation strategy. Refer to “Bitstream Options” on page 728. If the IP Evaluation strategy is enabled, Radiant software enables Soft IP timer to allow usage of temporary IP license. During the evaluation period, the bitstream is always encrypted with Lattice Soft IP Evaluation Key, and if necessary, overwrites user-provided AES key. Alternative types of keys for bitstream encryption can be generated in Bitstream Security Settings Tool, assuming the Lattice Soft IP Evaluation Key is enforced.

IMPORTANT

During the IP Evaluation, the bitstream is always encrypted with Lattice Soft IP Evaluation Key. The Lattice Soft IP Evaluation Key has a precedence over user-provided AES key.

See Also

- "Bitstream Security Settings Tool" on page 171

Bitstream Security Settings Tool

The Bitstream Security Settings tool (Security GUI) allows you to generate/verify keys that are used for bitstream obfuscation.

Security GUI provides the user entry for Password Protection (128-bit device password), AES-256 Encryption, HMAC Authentication, and ECDSA Authentication.

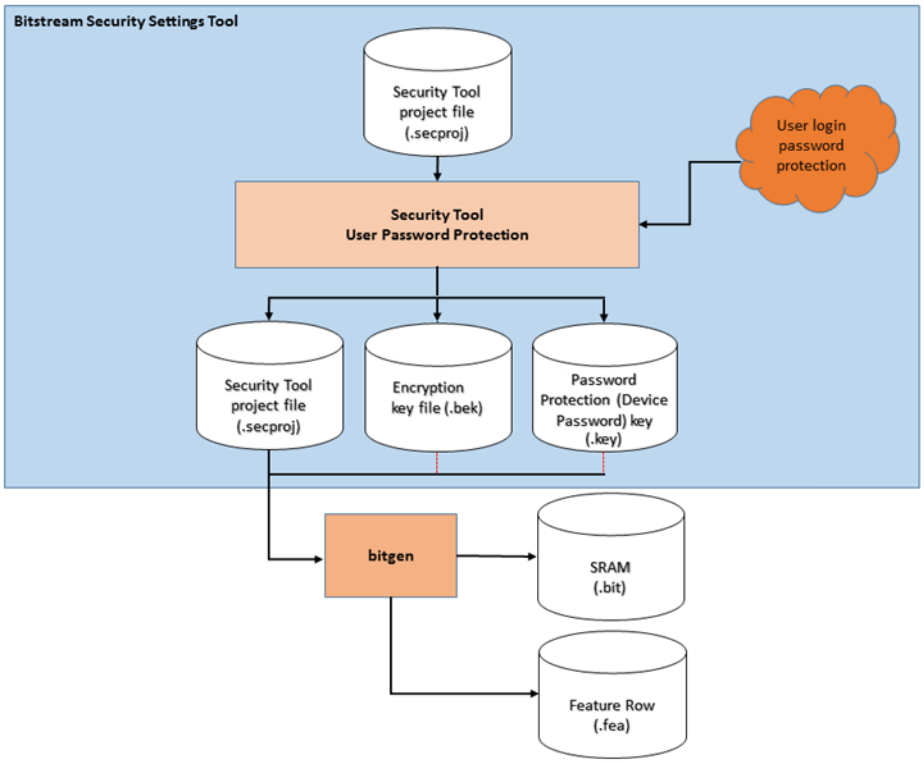
User login password protection is required in Security GUI. You can either define a new password or use the preset default LATTICESEMI password.

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.
- Password Protection 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 Private Key file (.prv, .pem, or .der) contains either:
  - 256-bit Private Key for ECDSA Authentication.
  - 256-bit Signature for ECDSA Authentication.
  - 256-bit HMAC Key for HMAC Authentication.
- Authentication Public Key file (.pub, .pem, or .der) contains 512-bit ECDSA Public Key.

Note:

- The .secproj, .key, .bek, .prv, .pem, and .der files should not be edited by user.
- HMAC Authentication and ECDSA Authentication are mutually exclusive.



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

Password Protection (Device Password)

Password Protection generates a 128-bit device password key.

To run the Bitstream Security Settings tool and set security settings:

1. In Lattice Radiant, choose **Tools > Bitstream Security Settings**.

   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 **Password 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.

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 Bitstream Security Settings tool and set security settings:

1. In Lattice Radiant, choose **Tools > Bitstream Security Settings**.

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

HMAC Authentication

A keyed-hash message authentication code (HMAC) is a specific type of message authentication code involving a cryptographic hash function and a secret cryptographic key.

To keep the HMAC Key protected, HMAC Authentication is always used together with AES Encryption. The bitstream, with the HMAC key, is first run through the SHA-2 hash function. The hashed function is then AES encrypted with the 256-bit AES key.

To run the Bitstream Security Settings tool and set security settings:

1. In Lattice Radiant, choose **Tools > Bitstream Security Settings**.

   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. To set up HMAC key, check the **HMAC Authentication** option in the Security Settings dialog box.
8. 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 HMAC key in hex format.

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

   A Security Setting message informs you that the encrypted files have been produced.
10. Click **OK** to close the Security Setting dialog box.

Note:

You will see ‘AES must be enabled to make HMAC authentication work.’ message if you did not enabled the AES Encryption.

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-2 Signature and Public key pair. The Signature size is 64 Bytes.

An optional feature allows to verify the SHA-2 Signature against the SHA-2 Digest.

IMPORTANT

The SHA-2 digest is used to verify Signature and Public key and is **not** saved to Security project file.

To run the Bitstream Security Settings tool and set security settings:

1. In Lattice Radiant, choose **Tools > Bitstream Security Settings**.

   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. Select authentication modes from the **Authentication Mode** drop-down list.

   b. 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.

   c. Check the **Public key and SHA-2 signature** option. Public key and SHA-2 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-2 digest. The digest size is 256 bits. Check **Verify** to verify the SHA-2 signature against the SHA-2 digest; otherwise, Public key and Signature will be saved, but **not** verified as a valid pair.

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

   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 173 and “ECDSA Authentication” on page 175.

See Also

- “Secure Objects in the Design” on page 159
- “Securing the Bitstream” on page 170

ECDSA Authentication (Avant Devices)

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 1056 bits, the Private key size is 528 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-2 Signature and Public key pair. The Signature size is 132 Bytes.

An optional feature allows to verify the SHA-2 Signature against the SHA-2 Digest.

IMPORTANT

The SHA-2 digest is used to verify Signature and Public key and is **not** saved to Security project file.

To run the Bitstream Security Settings tool and set security settings:

1. In Lattice Radiant, choose **Tools > Bitstream Security Settings**.

   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. Select authentication modes from the **Authentication Mode** drop-down list.

      - The supported authentication mode for the Avant device is only **ECDSA-521**.

   b. 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 1056-bit Public and 528-bit Private key pair in hex format.

c. Check the **Public key and SHA-2 signature** option. Public key and SHA-2 signature dialog boxes are enabled. Both Public key and Signature, has size of 1056 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-2 digest. The digest size is 256 bits. Check **Verify** to verify the SHA-2 signature against the SHA-2 digest; otherwise, Public key and Signature will be saved, but **not** verified as a valid pair.

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

   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 173 and “ECDSA Authentication” on page 175.

See Also

- “Secure Objects in the Design” on page 159
- “Securing the Bitstream” on page 170

Simulating the Design

This topic explains how to perform functional and timing simulation by using the Radiant software tools and third-party simulators.

See Also

- “Functional Simulation” on page 181
- “Simulation Environment” on page 181
- “Creating a New Simulation Wizard Project in the Radiant Software” on page 182
- “Exporting Gate-Level Simulation Files” on page 186
- “Verifying Designs with Timing Simulation” on page 187
- “Third-Party Simulators” on page 187

Simulation in the Radiant Software

The Radiant software provides you with a wizard to create a new simulation project file (.spf) that you can use to launch a simulation in a standalone simulator. Radiant supports launching the Active-HDL and QuestaSim™ simulators through generated scripts by the Simulation Wizard.

See Also

- “Functional Simulation” on page 181
- “Simulation Environment” on page 181
- “Creating a New Simulation Wizard Project in the Radiant Software” on page 182
- “Exporting Gate-Level Simulation Files” on page 186

Functional Simulation

Functional simulation is the process of verifying that the model of the design matches the functional specification, and helps identify logic errors in a design before it is programmed into a device.

To perform functional simulation on designs modeled with Verilog HDL in the Radiant software design flow:

- Perform pre-synthesis functional simulation on your source HDL or netlist using a third party simulator.
- In Radiant, use the subprocesses **Post-Synthesis Simulation File** or **Gate-Level Simulation File** in the **Task Detail View** to produce a simulation file for functional simulation of primitive gate-level logic during that stage in the flow.

  For more information about these subprocesses, refer to "Creating a New Simulation Wizard Project in the Radiant Software" on page 182.

HDL Simulators

The Radiant software and model libraries are qualified with a variety of popular HDL simulators, including Cadence® Xcelium™, Siemens QuestaSim, and Synopsys® VCS®.

See Also

- "Simulation in the Radiant Software" on page 180

Simulation Environment

In Radiant, you can simulate your projects using one of the following methods:

1. **Simulation Wizard** – Generates simulation scripts for you. You can also use the **sim_generate_script** Tcl command to generate template files from the current Radiant project settings.
2. **QuestaSim** – Radiant includes a stand-alone version of QuestaSim for project simulation. To simulate successfully with QuestaSim, you must be familiar with scripting or the tool's graphical user interface.

   For more information about scripting and commands using QuestaSim, see the QuestaSim User Manual located in the User Guides section of the Radiant Help.
3. **Third-party simulation tools** – Simulation using third-party tools require you to create your own simulation libraries using the cmpl_libs script. To simulate successfully with third-party tools, you must be familiar with scripting or the tool's graphical user interface.

See Also

- "Creating a New Simulation Wizard Project in the Radiant Software" on page 182

- "Simulation Libraries Compilation Tcl Commands" on page 1643
- "Performing Simulation with QuestaSim" on page 191
- "Third-Party Simulators" on page 187

Creating a New Simulation Wizard Project in the Radiant Software

By generating simulation scripts, Simulation Wizard simplifies the process of setting up a project for simulation. This section describes how to use the Simulation Wizard to create a simulation project (.spf) file.

Note:

If your project is created with ModelSim, it will be configured as a legacy project. A dialog box will appear indicating that legacy simulation projects will be replaced by QuestaSim projects.

1. In the Radiant software, click **Tools > Simulation Wizard** or click the  button in the toolbar. The Simulation Wizard opens.
2. In the **Preparing the Simulator Interface** page, click **Next**.
3. In the **Simulator Project Name and Stage** page, enter the name of your project in the **Project name** text box.

   In the **Project location** box, browse to the file path location where you want to save your simulation project.

   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 dialog box that appears.
4. In the Simulation Wizard dialog box, select either the **QuestaSim** or **Active-HDL** simulator.

   QuestaSim Qrun Option

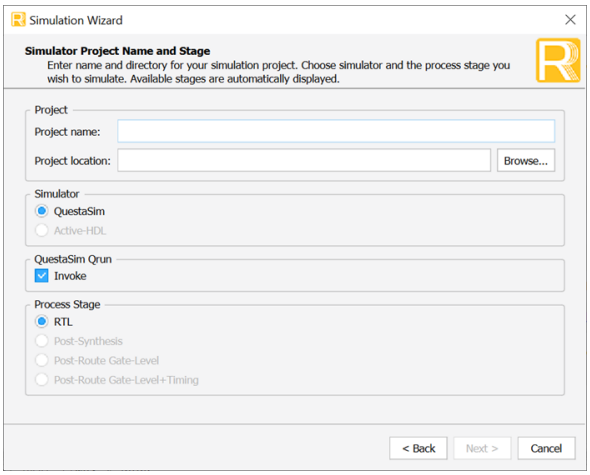
   - If QuestaSim is selected, the I**nvoke QuestaSim Qrun** can be toggled to determine if the simulation wizard project generates an argument file (<Project Name>.f) for Qrun or a do script (<Project Name>.mdo) to compile and launch the simulation.
   - If you invoke Qrun for simulation, Questa will not use a project based flow to compile your design and launch the simulation. You will not be able to use the compile all button in the QuestaSim GUI to recompile your design as this GUI button is only available when you have an open project.

     To recompile your design and launch simulation in a single step with Qrun, invoke Qrun with the generated argument file in the transcript window with `qrun -f <Project Name>.f` and Qrun will recompile files

that have been edited since the last Qrun invocation and launch the simulation.

To only perform a single step of a simulation flow, invoke Qrun with one of the -compile, -optimize, or -simulation arguments described in the Qrun user manual.

- If you do not invoke Qrun for the simulation, the Simulation Wizard will generate a do script that sets up a QuestaSim project, similar to how it did for ModelSim in previous versions of Radiant. The simulation script is different from Qrun; with project mode, the compile all button can be used in the simulator GUI.
- For more information, see the QuestaSim GUI Reference Manual.



Note:

- If Active-HDL is not installed, it will not be selectable.
- If you change simulation tool, project name will not change. Any previous Simulation Wizard results will be deleted, and a new simulation will be started using new simulation files.
- Qrun is a utility to simplify the use of Questa to compile, optimize, and simulate designs by taking all input files and options and automatically executing the correct tools and command-line arguments to start a simulation. Refer to the Qrun User Manual for more information about Qrun's supported options.

5. In the **Process Stage** box, choose the stage where you want to start creating the simulation project.

   The following options are valid types:

   - **RTL** – The RTL option allows you to run simulation before synthesis. This option is always selectable.
   - **Post-Synthesis**
     - To enable this option, you must first generate a post-synthesis simulation file (<file_name>_syn.vo) as follows:

In the Process Toolbar, click Task Detail View and select **Post-Synthesis Simulation File**.

Run **Synthesize Design**.

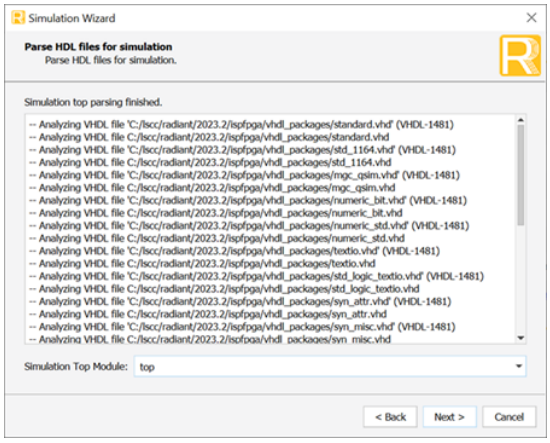
- **Post-Route Gate-Level** and **Post-Route Gate-Level+Timing**
  - To enable these options, you must first generate a post-routed, back-annotated simulation file (<file_name>_vo.vo) and a timing constraint file (<file_name>_vo.sdf) as follows:

    In the Process Toolbar, ensure that **Gate-Level Simulation File** is selected in Task Detail View and click **Export Files**.

6. Click **Next**.
7. In the **Add and Reorder Source** page, the files included in the simulation are listed in the main **Source Files** window.
   - Files will be compiled in the order they appear here, with the top file being the first to be compiled.
   - You can also manually add and remove files using the buttons at the top of the window.

   By default, the **Automatically set simulation compilation file order** option is enabled.
   - If this option is enabled, Radiant automatically detects the order of the simulation files. Otherwise, you have to manually specify the order of the files for your simulation using the buttons.
8. Click **Next**.

   The **Parse HDL files for simulation** page appears.



   Radiant automatically infers the **Simulation Top Module**.

   If you want to select a different testbench for your project, use the **Simulation Top Module** dropdown menu.

   Click **Next** after choosing the correct testbench module.
9. A **Summary** page appears that provides information on the project selections, including the simulation libraries.

Review all of the information on this page to ensure everything is correct.

10. By default, the **Launch Simulator GUI**, **Add top-level signals to waveform display**, **Design Optimization - Full Debug**, **Run simulation**, and **Simulator Resolution** options are enabled in the Summary page. Their functionality is as follows:
    - The **Launch Simulator GUI** option runs the generated script and launches the simulation tool you chose earlier in the Simulation Wizard setup page.
    - The **Add top-level signals to waveform display** option adds top-level signals to the waveform in the simulation tool.
    - The **Design Optimization - Full Debug** option sets the option for disabling QuestaSim's optimizer to preserve full visibility of the design in the simulator. If this option is not selected, QuestaSim's optimizations will take place, which will improve the runtime of the simulation at the expense of design visibility, so signals in the design may not be able to be added to the waveform when the Full Debug option is not selected.
    - The **Run simulation** option runs the simulation for the specified amount of time. If the option is set to 0, the **run -all** command will be executed, meaning the simulation will be run forever or until it hits a breakpoint or break event.
    - The **Simulator Resolution** option sets the simulator's time resolution. If you select a different value from the dropdown menu instead of default, the Resolution setting adds -t <value> to the vsim invocation.
11. Click **Finish**.
    - After completing the Simulation Wizard setup:
      - A **Simulation Wizard Project (.spf) file** and a **qrun argument (.f) file** are generated.

        The qrun argument file contains generated commands and arguments. For example:

        To compile all of the Verilog modules in your design, the Simulation Wizard adds the path to your Verilog files in the argument file, and generates a run command to run the simulation in the duration specified in step 11.

        The generation of this file is triggered by Simulation Wizard or the **sim_generate_script** command.
    - The simulation is launched and displayed in the QuestaSim GUI.

See Also

- “Simulation in the Radiant Software” on page 180
- “Simulation Libraries Compilation Tcl Commands” on page 1643

Exporting Gate-Level Simulation Files

In the Radiant software, you can export gate-level project simulation source files into your third-party vendor simulation tool for timing simulation.

To export a Gate-Level simulation files in the Radiant software:

1. From the Quick Run Toolbar, click Task Detail View and select the **Gate-Level Simulation File** subprocess under **Export Files**.
2. Double click **Export Files** to run the export feature. This will generate a back-annotated version of your design file in a Verilog netlist (.vo) simulation file and an SDF timing file.

Note:

You can also use Tcl Radiant commands in the Tcl Console to accomplish this task, as shown below:

```
prj_run Export -task TimingSimFileVlg
```

See Also

- "Simulation in the Radiant Software" on page 180

Timing Simulation

The Radiant software supports two types of timing verification:

- Dynamic timing simulation
- 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, Radiant 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 **Gate-Level Simulation File** to export a simulation project file to a standalone 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 187

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.

Radiant simulation files can be imported into HDL simulators like QuestaSim 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 Radiant software TRACE program.

Note

You cannot generate text fixtures in the Radiant software. All simulation must be done outside of the Radiant software using simulation files in a supported simulator. For guidelines using third-party EDA simulators with Radiant software, see your vendor documentation or the following references.

See Also

- "Timing Simulation" on page 186

Third-Party Simulators

This section explains how to use Cadence Xcelium, QuestaSim, and Synopsys VCS software to simulate designs that target Lattice Semiconductor FPGAs. It shows you how to use these simulators to perform functional register-transfer-level (RTL) simulation and place-and-route gate-level simulation with and without timing simulation.

Before simulating with third party simulators, you must compile the device library for use with your chosen simulator using the cmpl_libs.tcl script.

For details on how to run cmpl_libs.tcl, refer to "Running cmpl_libs.tcl from the Command Line" on page 1479.

Note

Lattice Semiconductor does not supply the Cadence Xcelium, Synopsys VCS, or QuestaSim simulators. You must obtain them independently.

See Also

- "Simulation Library Files" on page 188
- "Performing Simulation with Cadence Xcelium" on page 189
- "Performing Simulation with Synopsys VCS" on page 190
- "Performing Simulation with QuestaSim" on page 191

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 QuestaSim located at: <*install_folder*>/cae_library/simulation/libs.

Verilog library source files are also installed in the cae_library directory. A tcl script to compile the libraries from the source files is available at *<install_dir>/cae_library/simulation/scripts/cmpl_libs.tcl*. You can refer to *<install_dir>/cae_library/simulation/scripts/pkg/cmpl_vlg-pkg.tclproc* to view compile options and order for the device libraries.

Verilog Source Library Files

The Verilog functional and behavioral simulation library files are installed with the Radiant software in the locations shown in the following table.

Table 12: Verilog Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
iCE40UP	<*install_dir*>/cae_library/simulation/verilog/iCE40UP
LFCPNX	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/lfcpnx
LFD2NX	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/lfd2nx
LFD2NX1	<install directory>/cae_library/simulation/verilog/uaplatform <install directory>/cae_library/simulation/verilog/lfd2nx1
LFD2NX2	<install directory>/cae_library/simulation/verilog/uaplatform <install directory>/cae_library/simulation/verilog/lfd2nx2
LIFCL	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/lifcl
LFMXO5	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/lfmxo5
LFMXO5T	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/lfmxo5t

Table 12: Verilog Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
LAV-AT	<*install_dir*>/cae_library/simulation/verilog/applatform <*install_dir*>/cae_library/simulation/verilog/ap6a00 <*install_dir*>/cae_library/simulation/verilog/ap6a00b <*install_dir*>/cae_library/simulation/verilog/ap6a00c
LAV-AT-ES	<*install_dir*>/cae_library/simulation/verilog/applatform <*install_dir*>/cae_library/simulation/verilog/ap6a00 <*install_dir*>/cae_library/simulation/verilog/ap6a00b
UT24C	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/ut24c
UT24CP	<*install_dir*>/cae_library/simulation/verilog/uaplatform <*install_dir*>/cae_library/simulation/verilog/ut24cp

See Also

- "Third-Party Simulators" on page 187

Performing Simulation with Cadence Xcelium

This section explains how to perform simulation with the Cadence Xcelium simulator.

To provide the compiled library from the cmpl_libs.tcl script:

- An example of compiling the libraries for Xcelium for Avant devices to a directory set to the variable LIBPATH from the Radiant tcl shell:

  ```
  cmpl_libs -sim_path /tools/dist/cadence/XCELIUM/
  XCELIUM23.03.003/Linux/tools/bin -device lavat -sim_vendor
  cadence -target_path $LIBPATH
  ```

- When specifying your top-level design, pass the path to the compiled library to **xrun** with the **-reflib** argument.

  For example, if you compiled the lav-at library to a path specified in the LIBPATH variable, run the following commands when creating a top-level design with the name **top**:

  ```
  xrun -reflib $LIBPATH/lav_at-elaborate -top top
  ```

To run Xcelium:

- If the process is successful, a simulation snapshot that can be implemented will be the result. Run the following commands to launch the most recent snapshot that has been created.

  ```
  xrun -R
  ```

- For more simulation options, refer to the Xcelium User Guide.

Performing Simulation with Synopsys VCS

This section explains how to perform simulation with the Synopsys VCS simulator. This is supported in designs using the Avant device.

1. Run the **cmpl_libs.tcl** script to compile the device library to be used with the VCS simulator.

   For example, you can compile an Avant device library to a directory set to the variable LIBPATH from the Radiant tcl shell by running the following commands.

   ```
   cmpl_libs -sim_path /tools/dist/synopsys/VCS/U-2023.03-SP2/
   bin/ -device lavat -sim_vendor synopsys -target_path
   $LIBPATH
   ```

   - For details on how to run cmpl_libs.tcl, refer to "Running cmpl_libs.tcl from the Command Line" on page 1479.

   Note:

   When compiling library using VCS, the library that you compiled must be consistent with the device you are using. You can also add PMI library for compilation if you are using PMI primitives.

2. The **synopsys_sim.setup** file is created. Set up the synopsys_sim.setup file to map the design library to the host directories, indicate search paths, and assign values to simulation control variables.

   - For details on how to set up the synopsys_sim.setup file, refer to the *Creating a synopsys_sim.setup File* section of the VCS User Guide.

3. Copy the configured synopsys_sim.setup file to the design directory from where you invoke and run VCS. The mapping tells VCS what libraries to look for. The libraries are compiled to the database format of the VCS.

   For example, after copying the synopsys_sim.setup file to the current working directory, you can run the following command to process a design with the top module "top":

   ```
   vcs top
   ```

4. Analyze and elaborate your design. See the *VCS Workflow* section of the VCS User Guide.

   Successful elaboration means that VCS was able to resolve all of the device primitives. During elaboration, using the intermediate files generated, VCS creates a binary executable **simv** file. You can use the simv to run the simulation.

5. To simulate your design, run the generated simv executable either in interactive mode with the **-ucli** parameter or without any arguments.

   ```
   ./simv -ucli
   ```

VCS also has a utility called llib that you can use to print the library information. By default, it will only print the modules in the WORK library. However, you can use the **-lib** argument to get the information for a specific library.

For more simulation options, refer to the VCS User Guide and for more advanced usage, contact Synopsys user support.

Performing Simulation with QuestaSim

This section explains how to perform simulation with QuestaSim.

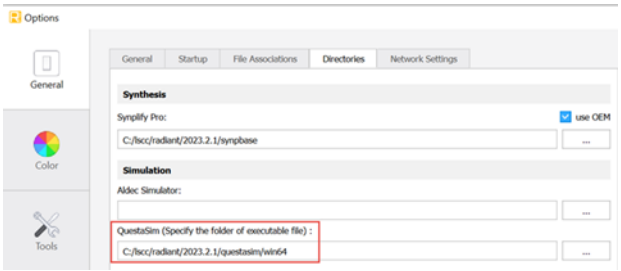
To set Lattice Semiconductor libraries:

Before simulating Lattice Semiconductor FPGA designs in Siemens' QuestaSim, you must have the compiled Lattice device and PMI libraries, which are included in Radiant or the libraries you compiled manually using the cmpl_libs script that Radiant provides.

- For details on how to run cmpl_libs.tcl, refer to "Running cmpl_libs.tcl from the Command Line" on page 1479.
- For details on PMI simulation library compilation, refer to "Requirements for Simulation and Synthesis with PMI" on page 149.

To enable the Radiant software's batch interface to QuestaSim, change the default simulation tool setup and directory reference as follows:

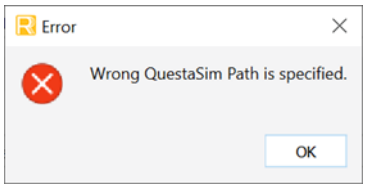
1. Choose **Tools > Options**.
2. Click on the **Directories** tab.
3. Change the path setting to the new version of the QuestaSim executable.



The following are default paths of QuestaSim:

- **Windows**: <*install_directory*>/questasim /win64
- **Linux**: <*install_directory*>/questasim /linux_x86_64

If there is no available executable file under a specified path, an error message will be issued.



4. Click **OK**.

To map QuestaSim libraries:

The simulator looks for referenced library names by doing the following:

1. Search the **modelsim.ini** file for library mappings.
2. Search the current working directory for a subdirectory that matches the library name.

The library compilation target folder should also be your simulation working directory. You must do one of the following procedures if the library does not exist in either the modelsim.ini or the simulation directory:

- Copy the library directory to the simulation directory.
- Use the **vmap** command to add a mapping to the library directory in the modelsim.ini.
- You can also manually add the mapping to the modelsim.ini file either in the simulation directory or the installation directory of QuestaSim.

Library mapping example using the LIFCL device:

```
vmap lifcl D:/radiant_lib/lifcl
vmap pmi_work D:/radiant_lib/pmi_work
```

or

1. Open the modelsim.ini in the simulation or installation directory
2. Go to the **Library** section.
3. Add **lifcl = D:/radiant_lib/lifcl**
4. Add **pmi_work = D:/radiant_lib/pmi_work**

Performing Verilog Simulation

Use the procedures described in this section to perform a Verilog standalone simulation with QuestaSim.

Functional RTL Simulation

Use the following commands to perform a functional RTL simulation with one of the libraries shown in the Verilog Functional and Behavioral Simulation Library File Locations table:

```
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
```

See Also

- "Third-Party Simulators" on page 187

Single-Step Simulation Using qrun Command

The qrun command simplifies the QuestaSim verification flow by combining the compile, optimize, and simulate functions into a single step. With the qrun command, you do not need to explicitly invoke vlog, vopt/vcom, and vsim and you can eliminate the manual command line construction and linking associated with the three-step flow.

Here is the basic syntax of the qrun command.

```
qrun <myfiles> <optional arguments>
```

qrun arguments are optional. When run, the command compiles, optimizes, and simulates "<myfiles>" with or without explicit direction or arguments from the user.

Here is an example of a Hello World System Verilog module with the file name hello.sv.

```
module IP;
initial begin
$display("Hello world");
end
endmodule
```

To compile, optimize, and simulate the **hello.sv** file, use the qrun command.

```
qrun hello.sv
```

Comparing ModelSim and QuestaSim

The following table highlights the differences between ModelSim and QuestaSim.

Table 13: Major differences between ModelSim and QuestaSim

	License	Optimization Modes[1]	Instantiation limit[2]	Simulation library files
ModelSim	Requires a license with the **latticemsim** feature	No optimization	ModelSim issues a warning when going over the 5000 instantiation limit with no further action needed to launch the simulation.	ModelSim pre-compiled library files are located at: **<install_folder>/modeltech/lib**.
QuestaSim	Requires a license with the **latticeqsim** feature	Optimized and Debug	QuestaSim issues a suppressible error when simulating a design that goes over the 5000 instantiation limit.	QuestaSim pre-compiled library files are located at: **<install_folder>/cae_library/simulation/libs**.

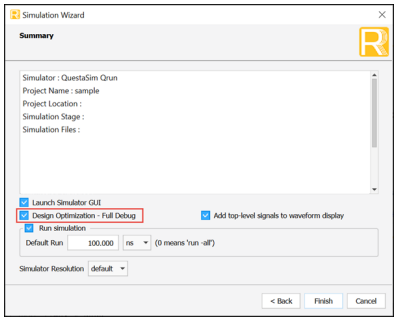
Notes:

1. QuestaSim performs built-in optimizations on your design to maximize simulator performance, while ModelSim does not perform these optimizations. The optimizations limit the visibility of design objects, but you can increase visibility of any objects for debugging purposes. For more details on options regarding optimization and preserving visibility, refer to the QuestaSim User Manual

   It is recommended to use the **+acc** option to vopt to preserve design visibility so that the resulting waveform database can be viewed in the Questa Classic GUI provided with Questa Lattice OEM.

   You can enable the +acc and +noacc options by selecting the **Design Optimization - Full Debug** option from the Simulation Wizard dialog box. The +acc option can slow down design performance, whereas the +noacc option improves performance by re-enabling optimizations on specified design objects, negating the disabling of optimizations that had been specified through previous +acc arguments.

   For details on the negation of arguments to QuestaSim's optimization engine, refer to the “Negation Arguments and Resolution with vopt” section in the QuestaSim User Manual.



2. In both Questa Lattice OEM and ModelSim Lattice OEM, running the simulation with a design that goes over the instantiation limit will result in severely degraded simulation performance.

Applying Design Constraints

Constraints are instructions applied to design elements that 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 assignments, but constraints are also available for placement, routing, and many other functions.

Constraints can include timing or physical constraints defined in Constraint Files (.ldc/.pdc/.sdc/.fdc) or HDL attributes (for physical pin locking) using the Radiant software and third party synthesis tools. Files .ldc/.fdc/.sdc are used for the synthesis tool and specify design goals. Synthesis, map, and place-and-route work to meet these goals. Post-synthesis constraints (.pdc) can also be specified. The flow combines these based on different entry points within the Radiant software design flow. The timing analysis tool reports whether or not the goals were met.

See Also

- "Constraints Reference Guide" on page 731
- "Unified Constraints Flow" on page 196
- "Integrated Synthesis" on page 317
- "Handling Device Constraints" on page 212

Unified Constraints Flow

The goal of the unified constraints flow is to have one constraint file that can be used by both the synthesis tools (LSE & Synplify Pro). From Radiant 3.0 onwards, pre-synthesis constraint editor can be launched with Synplify Pro, and constraints can be saved into an .sdc file, which can be used by both LSE and Synplify Pro, along with .ldc (for LSE) and .fdc (Synplify Pro).

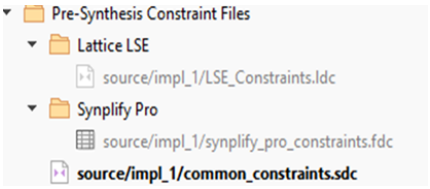
Table 14: Supported Constraints File Type

Constraint Type	Synthesis Tool	Constraint Editor Support
.ldc	LSE	Yes
.sdc	LSE/Synplify Pro	Yes
.fdc	Synplify Pro	No

Table 15: Difference between Radiant 3.0 (or later) and previous versions

	Radiant 1.x/2.x	Radiant 3.0 or later
.sdc file	For Synplify Pro only	For both LSE and Synplify Pro

The following figure shows the Radiant folder structure. The .ldc constraints are stored under the Lattice LSE folder, the .fdc constraints are stored under Synplify Pro, and the newly created .sdc constraints are stored outside these two folders, which can be commonly used by both synthesis tools.



Notes:

1. If the previous project includes an .sdc file, Radiant 3.0 or later will use it in the LSE flow, but it can cause unexpected syntax errors.
2. If the previous project includes both .ldc and .sdc files, Radiant 3.0 or later will force one of them to be inactive. Only one constraint file can be active. The Pre-Synthesis Constraint Editor can only be launched with the active constraints file.

Known Limitations

- Constraints on internal objects from Synplify Pro may not be fully compatible. The pre-synthesis constraint editor uses Verific parser for

compilation whereas Synplify Pro uses their own parser. This may cause some name changes on the internal objects when Synplify Pro is used.

- Pre-Synthesis Constraint Editor "Attribute" tab is only supported by LSE. The attributes are passed using a Lattice Proprietary constraint "ldc_define_attribute," which is not supported by Synplify Pro.

See Also

- "Constraints Reference Guide" on page 731
- "Using Radiant Software Pre-Synthesis Constraints" on page 211
- "Device Constraint Editor" on page 228

Constraint Implementation

The following steps illustrate how you might assign constraints and implement them at each stage of the design flow, using synthesis constraints 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.
   - Define timing constraints using Synplify Pro or LSE. Constraints are saved in .ldc file.
2. Synthesize the design.

   When synthesis is performed, the synthesis process synthesizes the design and stores it in a unified database.

   The timing constraints will be displayed in Radiant software Design Constraint Editor View and color-coded as synthesis values.
3. Open one or more of the following views in the Pre-Synthesis Constraint Editoror Device Constraint Editor to define new constraints:

   (Pre/post-synthesis) Pre-Synthesis Constraint Editor– Timing constraints, exceptions, delays and synthesis attributes including physical pin locations can be defined in these two tools.

   Device Constraint Editor– Define signaling standards and make pin assignments. Assign clocks to primary or secondary routing resources. Define groups of ports only with current release. Other global settings such as temperature and voltage can be set. Run Constraint design rule checking.

   Placement Mode – View the device layout. Draw bounding boxes for GROUPs. Draw REGIONs for the assignment of groups or to reserve areas. Reserve sites and REGIONs that should be excluded from placement and routing. Run Constraint design rule checking.
4. Save the constraints to the .pdc file.
5. Run the **Map Design** process (map).

   This process reads all data from the unified design database and processes from there.

6. Run the **Place & Route Design** process (par).
7. Open one of the following views, as desired, to examine timing and placement.

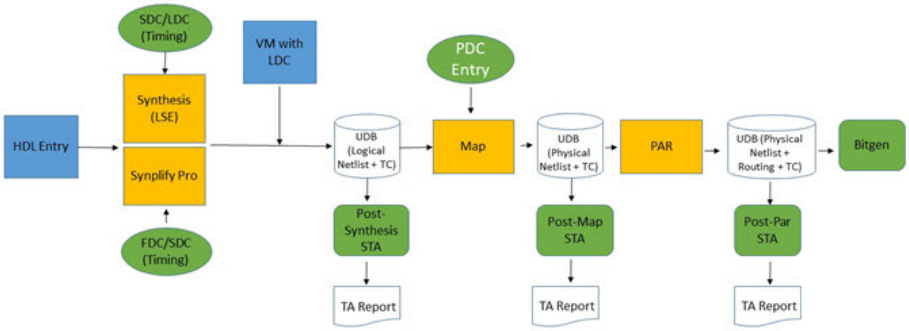
   Timing Analyzer – Examine details of timing paths. Cross-probe selected paths to Placement Mode and Routing Mode.
8. Modify constraints or create new ones using any of the design / timing constraint editing views. Save the changes and rerun the **Place & Route Design** process.
9. Repeat Steps 3 through 8 until design performance objectives are met.

See Also

- "Analyzing Static Timing" on page 350

Understanding Implications of Radiant Software Constraint Flows

In order to help understand the Radiant software constraints flow process, the figure below details the entry mechanism for input files and its data flow through the constraints process. The blue boxes indicate design entry mechanism in the form of HDL and synthesized via our internal LSE or third party Synplify Pro synthesis tools. Other yellow boxes also indicate processes run by Radiant software. Green boxes are used to indicate input or output constraints files or related output report files such as Static Timing Analysis reports out of Map and PAR. Note that Radiant software also generates a post-synthesis timing report just for LSE.



The important process to note is that the Unified Database (.udb) used to store the processed data dictates that certain stages of the constraints need not be re-run in the tool saving processing time. For example, physical constraints on RTL entered as attributes in the HDL, or pre-synthesis constraints entered using the Pre-Synthesis Constraint Editor/Synplify Pro SCOPE (or text editor) including .ldc/.fdc (.sdc converted to .fdc) timing constraints would initially run through synthesis and the data stored is in the .udb file. Subsequent timing and physical constraints entered using tools such as the Post-Synthesis Timing Constraint Editor, Placement Mode or Device Constraint Editor and stored in the .pdc file are processed via the mapping run without the need to re-synthesize the design since the pre-synthesis data

resides in the .udb database. As the constraints flow proceeds with each process (i.e. synthesis, Map, PAR) the .udb successively stores a physical netlist, routing and timing constraint.

Transparent to the user, the storage of accumulating data throughout the constraints flow in the .udb and reduction in re-running processes is one reason for the improved speed and efficiency for Radiant software users.

In terms of the necessity of pre versus post-synthesis timing constraints, usage depends on the complexity and desired performance of the design results. A simple design requiring few timing constraints with a relaxed f_{MAX} may require only pre-synthesis timing constraints and runs through the flow from synthesis to place and route. A complex design requiring a higher f_{MAX} performance may have initial pre-synthesis timing constraints entered for synthesis. Later in the flow, a user may then want to override or fine tune these timing constraints in addition to adding physical constraints to reach the desired performance by driving the place and route process. One example would be a user who initially specifies a higher f_{MAX} to reduce logic levels, then later in the post-synthesis flow lowers the f_{MAX} constraint so that the place and route process is not strained when routing the design.

Managing Constraints: .ldc/.sdc/.fdc vs .pdc

The pre-synthesis constraints .ldc/.sdc/.fdc are passed to the .udb and shown in the Post-Synthesis Timing Constraint Editor as read only. If a user attempts to further re-constrain these constraints, the constraints are saved to the .pdc file.

If two identical constraints exist in the .ldc and .pdc files respectively, the constraint in the .pdc file takes precedence.

Constraints Precedence

Regarding physical/timing constraints, constraints entered via any GUI tool such as Device Constraint Editor or Pre-Synthesis Constraint Editor will take precedence over the constraints that were entered via an HDL attribute in the RTL or .ldc and .sdc TCL commands.

Constraints entered later in the design flow such as Post-Synthesis Timing Constraint (.pdc) will override a constraint entered in the Pre-Synthesis Constraint (.ldc) editor tool.

If there are conflicts in the constraint file such as a ldc_set_location TCL command conflicting with a ldc_prohibit on the same location, then the Radiant software will issue an error message in the design flow.

Applying Differential Signal Design Constraints

Lattice software treats all signals as single ended in RTL all the way to the Top Module port list. Differential buffer primitives are not instantiated in RTL.

Instead, I/O constraints are used to set the I/O type to differential.

The RTL signal should be assigned to the P side of a differential pin pair, and the software will set the N side to the matching differential pin pair's compliment pin.

The differential pin assignment for both true and compliment can then be seen in Reports > Place & Route > Signal/Pad > Pinout by Pin Number.

The true pin is shown with a "+" after the RTL signal name and the compliment pin with a "–" after the RTL signal name.

Since there is no primitive for differential buffers, the post-route netlist also treats the net as single ended.

You can also find the pad locations of the differential pair from the generated .pad file under "Pinout by Pin Number" with "+" and "-" signs after the RTL signal name.

Timing and Physical Constraints

There are three types of constraints that are supported in Radiant software for constraining your design for maximum performance: Lattice Design Constraints, Post-Synthesis Design Constraints, and Synopsys Design Constraints.

Lattice Design Constraints (.ldc): A combination of both Synopsys Design Constraints (.sdc version 2.0) and the Radiant software's synthesis attributes. Files have .ldc suffix.

Refer to "Synopsys Design Constraints" on page 750 in the Help for more details.

Post-Synthesis Design Constraints (.pdc): The Radiant software design constraints contain both physical constraints and any post-synthesis timing constraints. These files have .pdc suffix.

Below is a supported list:

- "ldc_create_group" on page 815
- "ldc_create_region" on page 815
- "ldc_create_macro" on page 816
- "ldc_create_vref" on page 816
- "ldc_set_location" on page 817
- "ldc_set_vcc" on page 817

- "ldc_set_port" on page 818
- "ldc_set_sysconfig" on page 818
- "ldc_set_attribute" on page 819
- "ldc_prohibit" on page 820

See Also

- "Synopsys Design Constraints" on page 750

Synopsys Design Constraints (.sdc): Synopsys design constraints are industry standard timing constraints. Files with .sdc suffix. Its currently in its version 2.0 iteration.

Below is a supported list:

- "create_clock" on page 753
- "create_generated_clock" on page 755
- "set_clock_groups" on page 757
- "set_clock_latency" on page 758
- "set_clock_uncertainty" on page 759
- "set_false_path" on page 760
- "set_input_delay" on page 761
- "set_load" on page 762
- "set_max_delay" on page 762
- "set_max_skew" on page 763
- "set_min_delay" on page 764
- "set_multicycle_path" on page 765
- "set_output_delay" on page 766

See Also

- "Lattice Synthesis Engine-Supported HDL Attributes" on page 767

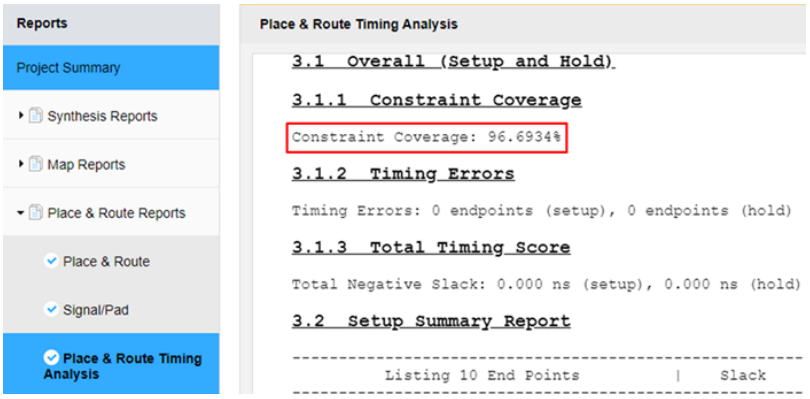
Checking Constraint Coverage

Timing Constraints are important in FPGA design because they tell the tools what to optimize and what to report on. A design that is unconstrained will go unoptimized and unreported because no timing constraints have been set to tell it how fast it needs to run, and there is no instruction about the required information.

To check if the applied timing constraints on the design is sufficient, you can check the constraint coverage on the **Timing Analysis Summary** of the Place and Route Timing Analysis report. Constraint coverage refers to the percentage of paths covered by the constraints for timing analysis.

You must aim for a constraint coverage ranging from 90% to 100%. At a minimum, you must do the following:

1. Ensure all clocks are defined with the required frequencies (create_clock, create_generated_clock).
2. Ensure all input and output ports are defined with the required delay requirements (set_input_delay and set_output_delay).



You can use a variety of timing constraints depending both on their preference and the requirements of the design. Depending on the design, some timing constraints may have a higher effect on the constraint coverage than others. There are timing constraints that do not affect constraint coverage. Refer to the following table for more details.

Table 16: Constraint Coverage

Constraint Category	Constraint	Affects Constraint Coverage?
Clock	create_clock	Yes
Generated Clock	create_generated_clock	Yes
Clock Latency	set_clock_latency	Yes
Clock Uncertainty	set_clock_uncertainty	Yes
Clock Group	set_clock_groups	No
Input/Output delay	set_input_delay	Yes
	set_output_delay	Yes
Timing Exception	set_min_delay	No
	set_max_delay	Yes
	set_false_path	Yes
	set_multicycle_path	No
	set_max_skew	No

Coverage

When extracting a timing graph from the design netlist, the engine removes netlist components that are not part of a complete path. The connections (driver to load section of a net) associated are classified as ignored connections.

The timing engine does the following:

- Indicates how many connections are ignored. The remaining connections are classified as counted connections.
- Checks each of the counted connections to identify which of the connections are part of a path that is properly constrained. These connections are considered as covered connections.

The engine prints the following message to the standard output:

```
Connections ignored 9 counted 34 covered 19.
```

The **Constraint Coverage** section expresses the ration of the covered connections to the counted connections as a percentage.

Most of the timing constraints that are supported affect the constraint coverage percentage. There are a few constraints that do not affect coverage. Those constraints are:

- set_multicycle_path
- set_min_delay
- set_max_skew

These constraints do affect the behavior of the various applications (synthesis, place and route) but do not affect the coverage of the connections.

Migrating from Former Lattice Diamond Preferences

New to the Radiant software flow are Lattice Design Constraints (.ldc/.pdc). These are a combination of both Synopsys Design Constraints (.sdc version 2.0) and the Radiant software proprietary physical constraints.

All Lattice Diamond constraints were known as Preferences and were defined as .ldc or .pdc files written as .sdc constraints. Lattice Diamond uses a combination of “preferences” and constraints, while Radiant software uses only .ldc constraints. Lattice Diamond preferences that can be supported as constraints in the Radiant software are listed below.

These are the .sdc timing commands that are supported in the Radiant software:

- create_clock
- create_generated_clock

- set_clock_groups
- set_input_delay
- set_output_delay
- set_false_path
- set_max_delay
- set_max_skew
- set_min_delay
- set_multicycle_path
- set_clock_uncertainty
- set_clock_latency

The use of .sdc constraints involve some object access commands as indicated below:

- get_cells
- get_pins
- get_nets
- get_ports
- all_inputs
- all_outputs
- get_clocks
- all_clocks

The following table shows the most commonly used Lattice Diamond preferences for applying physical constraints and how to apply them with the new Radiant software.sdc commands. Listed in alphabetical order, examples are also shown on how to convert Lattice Diamond preferences to a Radiant software .ldc constraint.

Table 17: Lattice Diamond Preferences Compared to Radiant Software Constraints

Lattice Diamond Preference Usage	Radiant Software .ldc Constraint Usage
BANK 1 VCCIO 3.3 V	ldc_set_vcc -bank 1 3.3
BLOCK PATH FROM PORT “LOCAL_CMD0” TO PORT “RAM_RD”;	set_false_path -from [get_ports LOCAL_CMD0] -to [get_ports RAM_RD]
CLOCK_TO_OUT ALLPORTS OUTPUT_DELAY 10ns CLKPORT CKIN1;	set_output_delay 10 -clock [get_clocks CKIN1] [get_ports all_outputs]
DEFINE BUS “busA” NET “pen0” NET “pen1” NET “pen2” NET “pen3” NET “pen4” NET “pen5” NET “pen6” NET “pen7”;	ldc_create_group -name busA [get_nets pen0 pen1 pen2 pen3 pen4 pen5 pen6 pen7]
DEFINE PORT GROUP “grp1” “a1” “a2” “a3”;	ldc_create_group -name grp1 [get_ports a1 a2 a3]
FREQUENCY NET “clk1” 100 Mhz	create_clock -period 10 -name clk1 [get_nets clk1]

Table 17: Lattice Diamond Preferences Compared to Radiant Software Constraints (Continued)

Lattice Diamond Preference Usage	Radiant Software .ldc Constraint Usage
GROUP “rot_grp” BBOX 8 3 BLKNAME rotate_1;	ldc_create_group -name rot_grp -bbox 8 3_cells {rotate_1}
GSR_NET NET “rst_pcie”;	ldc_set_attribute {GSR_NET = TRUE} [get_nets rst_pcie]
INPUT_SETUP PORT “in1” INPUT_DELAY 2 ns CLKPORT “clk”;	set_input_delay 2 -clock [get_clocks clk] [get_ports in1]
IOBUF PORT bchk IO_TYPE=LVCMOS33;	ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports bchk]
LOCATE COMP “A” SITE “11”;	ldc_set_location -site 11 [get_ports A]
LOCATE GROUP “GROUP_0” SITE “R2C2D”;	ldc_set_location -site R2C2D [ldc_get_groups GROUP_0]
LOCATE GROUP “xyz_GROUP” REGION “my_region”	ldc_set_location -region my_region [ldc_get_groups GROUP_0]
LOCATE VREF “VREF1_BANK_3” SITE “n21”;	ldc_create_vref -name VREF1_BANK_3 -site N21
MAXDELAY FROM PORT “LOCAL_CMD0” TO PORT “RAM_RD” 18 NS	set_max_delay 18 -from [get_ports LOCAL_CMD0] -to [get_ports RAM_RD]
MAXDELAY -MIN FROM CELL “state_reg4” TO CELL “RAM_OE_N_reg” 15 NS;	set_min_delay 15 -from [get_cells state_reg4] -to [get_cells RAM_OE_N_reg]
MULTICYCLE FROM CLKNET “clk1” TO CLKNET “clk2” 2 X;	set_multicycle_path 2 -from [get_clocks clk1] -to [get_clocks clk2]
OUTPUT PRT “A’ LOAD 10 pF;	set_load 10 [get_ports A]
PERIOD NET “NetA” 150 NS HIGH 75 NS;	create_clock -period 100 -name clk0 -waveform {0.75 1.50} [get_nets NetA]
PERIOD PRT “Clk1” 30 NS;	create_clock -period 30 -name clk1 [get_ports Clk1]
PROHIBIT PRIMARY NET “bf_clk”;	ldc_set_attribute USE_PRIMARY=FALSE [get_nets bf_clk]
PROHIBIT SITE “AB”	ldc_prohibit -site AB
REGION “Region_0” R16C2D” 11 30;	ldc_create_region -name Region_0 -site R16C2D -width 11 -height 30
SYSCONFIG DONE_OD=ON;	ldc_set_sysconfig {JTAG_PORT=ENABLE MCCLK_FREQ=56.2 DONE_OD=ON}
TEMPERATURE 45C;	ldc_set_attribute {TEMPERATURE=45C}
USERCODE HEX “53656D69”;	ldc_set_attribute {FORMAT=HEX CODE=53656D69}
VCC_DERATE PERCENT 5	ldc_set_vcc -core -derate 5
VCC_NOMINAL 1.2V;	ldc_set_vcc -core 1.2
VCCIO_DERATE BANK 0 PERCENT 5;	ldc_set_vcc -bank 0 -derate 5
VOLTAGE 1.5V;	ldc_set_vcc -core 1.5

Migrating Pin Assignments

The Radiant software 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 tab of the Device Constraint Editor 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 for LIFCL, LFD2NX, and UT24C.

See Also

- "Selecting a Device for Possible Pin Migration" on page 207
- "Viewing Incompatible Pins" on page 208
- "Prohibiting Incompatible Pins" on page 209

Selecting a Device for Possible Pin Migration

The Incompatible Pins tab of Device Constraint Editor 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 Device Constraint Editor by choosing **Tools > Device Constraint Editor**.
2. Open the Incompatible Pin tab. The Incompatible Pins tab 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.

   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, the incompatible pins will include those between the current device and the combination of all selected devices.

4. In the Check Areas tab, select the types of incompatibility that you want included for the selected devices: pin type, configuration, LVDS/ highspeed, and bank number differ between the current device and the selected devices.

   All of the pins that are incompatible will be displayed in a dim font and with a dark gray circle in the Package View.

5. To unlock any previously assigned pins that would now be incompatible, choose **Design > Disable Incompatible Pin Assignment.**

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 and marked as "[Incomaptible]" in the "Signal Name" column 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 the incompatible pin assignments will not be disabled.

See Also

- “Migrating Pin Assignments” on page 207
- “Viewing Incompatible Pins” on page 208
- “Prohibiting Incompatible Pins” on page 209
- “Exporting Pin Migration Information” on page 210

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 checked the "Disable Incompatible Pin Assignments". If you need to use any of these sites for your design, you must uncheck the menu or adjust the incompatible device/type 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 Ports 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

- “Migrating Pin Assignments” on page 207
- “Selecting a Device for Possible Pin Migration” on page 207
- “Prohibiting Incompatible Pins” on page 209
- “Exporting Pin Migration Information” on page 210

Prohibiting Incompatible Pins

Package View enables 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 the Incompatible Pins tab, select the devices for possible pin migration.
2. Click **Design > Prohibit Incompatible Pins.**

   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 dark blue.

To release all prohibited incompatible pins:

- Click **Design > Release Incompatible Pins**.

To prohibit one or several incompatible pins:

- 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

- “Migrating Pin Assignments” on page 207
- “Selecting a Device for Possible Pin Migration” on page 207
- “Viewing Incompatible Pins” on page 208
- “Exporting Pin Migration Information” on page 210

Exporting Pin Migration Information

The Radiant software 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 options in the Incompatible Pins tab, then selecting **Pin Migration** in the Export Pin Layout File dialog box.

See Also

- “Exporting a Pin Layout File” on page 308

Using Radiant Software Pre-Synthesis Constraints

Enter constraints in the synthesis design constraints .sdc file using one of the following three synthesis tools:

- Pre-Synthesis Constraint Editor/Post-Synthesis Timing Constraint Editor

  For more information about entering constraints with Pre-Synthesis Constraint Editor/Post-Synthesis Timing Constraint Editor, refer to “Handling Device Constraints” on page 212.

- Synplify Pro SCOPE Constraints Editor

  For more information about entering constraints with Synplify Pro SCOPE Constraints Editor, refer to “SCOPE Constraints Editor” help in the Synplify Pro Help.

- Lattice Synthesis Engine (LSE).

The following Synopsis Design Constraints can be set in the .sdc:

- “create_clock” on page 753
- “create_generated_clock” on page 755
- “set_clock_groups” on page 757
- “set_clock_latency” on page 758
- “set_false_path” on page 760
- “set_input_delay” on page 761
- “set_load” on page 762
- “set_max_delay” on page 762
- “set_min_delay” on page 764
- “set_multicycle_path” on page 765
- “set_output_delay” on page 766
- “set_clock_uncertainty” on page 759
- “Lattice Synthesis Engine-Supported HDL Attributes” on page 767

.

Note

Some IPs such as PLL requires the user to only supply the input reference clock and the Radiant software tool will automatically write out the associated generated clock constraints based on the supplied PLL parameter division ratios.

See Also

- “Handling Device Constraints” on page 212
- “Device Constraint Editor” on page 228

Handling Device Constraints

Here are some general tips for when applying device IP constraints or any primitives that generates or modifies clocking frequencies (i.e. PLL, I2C, SPI etc.). Note this does not include IP primitives from the .GBB timing files.

1. The precedence of device IP constraints supersedes user constraints.
2. User constraints will override device IP constraints.
3. TCL commands for all_xxx and get_xxx with wild-cards are only allowed on top level designs. When an IP is being processed as a top level design, it could contain these commands.
4. If the IPs are using all_xxx and get_xxx with wild cards, the commands have to be converted to get_xxx with a list when the final SDC of the IP is written out for consumption into the top level design.
5. Any device constraints in the IP will be converted to user constraints.
6. The device constraints will be placed before the user constraints in the .SDC file.

Applying Lattice Pre-Synthesis Timing Constraints

Radiant software LSE enables you to set pre-synthesis Synopsys® Design (formatted) Constraints, 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.

The Source Editor and the Pre-Synthesis Constraint Editor are available for creating and editing .ldc files. Pre-Synthesis Constraint Editor provides a spreadsheet style user interface that enables you to quickly create and edit Synopsys Design Constraints.

Pre-Synthesis Constraint Editor runs a design rule check (DRC) to verify that the constraints you enter are legitimate. These checks are done in real time as you enter them in the GUI.

See “Synopsys Design Constraints” on page 750 for descriptions of the .sdc constraints that are supported by the LSE and Pre-Synthesis Constraint Editor.

See Also

- “Creating a New .sdc Synthesis Constraint File” on page 213
- “Integrated Synthesis” on page 317
- “Optimizing LSE for Area and Timing” on page 314

Creating a New .sdc Synthesis Constraint File

Before you create a new .sdc file, verify that LSE 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 > Active Implementation > Select Synthesis Tool**, and select **Lattice LSE**.

To make sure that the .ldc file opens in Pre-Synthesis Constraint Editor instead of Source Editor, verify that Pre-Synthesis Constraint Editor has been selected as the default program in **Tool > Options > General > File Associations**.

To create a new .sdc synthesis constraint file:

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

   Alternatively, right-click the **Timing Constraint Files** folder and choose **Add > New File**.

2. In the New File dialog box, select **Source Files** from the Categories list. Select **Pre-Synthesis Constraint Files** from the Source Files list.

3. Type a name for the new .sdc file and specify the directory location.

4. Click in the **Add to Implementation** box if you want to use the file with the currently active implementation.

5. Click **New**.

   Pre-Synthesis Constraint 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 .sdc file will appear in the Pre-Synthesis Constraint Files folder.

   Pulldown menus for source elements are provided when you double-click the Clock Source cells or the Inputs/Outputs Ports cells.

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 pulldown list.

   b. Use the arrow keys or the Tab key to move to another cell and select it for editing.

   c. New rows are inserted automatically. You can delete existing rows with data entered by right clicking and selecting **Remove Row** from the pop-up menu.

7. Choose **File > Save**.

See Also

- “Defining Clocks Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 267
- “Setting Input and Output Delays Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 268
- “Defining Timing Exceptions Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 270

- “Synopsys Design Constraints” on page 750
- “Managing Constraint Files” on page 35
- “Integrated Synthesis” on page 317
- “Optimizing LSE for Area and Timing” on page 314

Using Source Editor for .ldc Files

Any .ldc file that you have created using the Pre-Synthesis Constraint 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 the Pre-Synthesis Constraint Editor.

One advantage of using Source Editor is that it allows you to open multiple .ldc files at once, whereas Pre-Synthesis Constraint 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 Pre-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 214

Changing the Default Editor for .ldc Files

The Radiant software gives you the option of setting Source Editor or Pre-Synthesis Constraint Editor as the default program for .ldc files.

To change the default editor for .ldc files:

1. Choose **Tools > Options**.
2. In the General section of the Options dialog box, select **File Associations**.
3. In the Extensions column of File Associations, scroll down to **ldc** and/or **ldc<active>**.
4. In the Default Programs column, choose **Source Editor** from the pulldown menu. The **ldc** option also allows for Add Program, an option to open with your choice of text editors.

   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 35

Optimizing for Timing

The pre-synthesis 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. Select **LSE** from the Synthesis Design folder in the Process pane in the Strategies dialog box.
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” on page 750
▶ “Optimizing LSE for Area and Timing” on page 314

Adding an .ldc File to an Implementation

You can add multiple .ldc files to an implementation and view each of them in Pre-Synthesis Constraint Editor.

To add an existing .ldc file to an implementation:

1. In the File List view, right-click the Pre-Synthesis Constraint Files 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 Pre-Synthesis Constraint Files folder list.

See Also

▶ “Managing Constraint Files” on page 35

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**.

By default, the active .ldc file is compiled when you synthesize the design.

See Also

- "Managing Constraint Files" on page 35

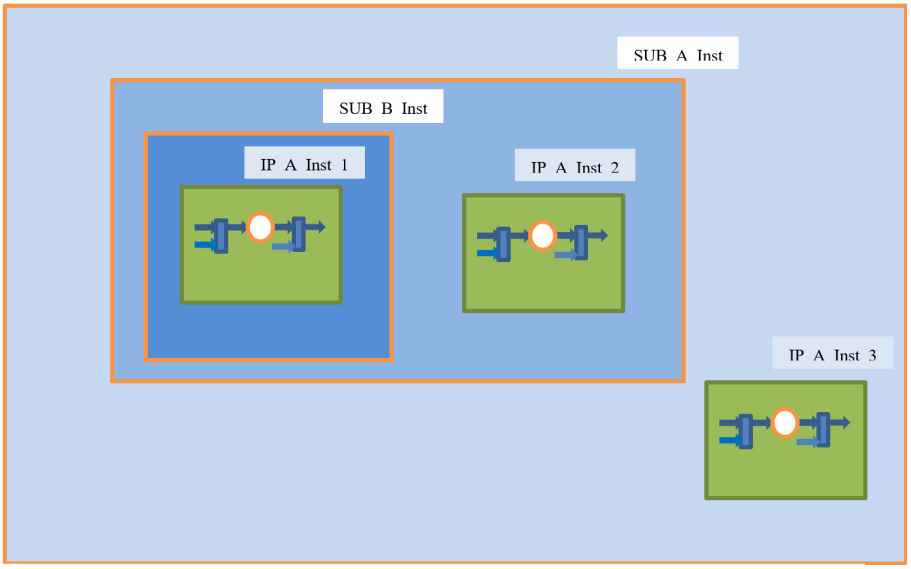
Constraint Propagation

Constraints are usually defined for their top level design as well as include many other constraints files for any custom included IP or IPs that are generated via Radiant tools such as IP Catalog. Constraints defined at the module IP level may not contain the correct hierarchical names and hence will not be applied correctly when synthesized. To help the user be able to honor as much of their supplied constraints as possible, a constraint propagation engine will run a DRC on all the input constraints and write out a new constraint file to support hierarchical constraints, such as soft IP constraints and be honored in the top-module as to enable optimization across IP boundaries during logic synthesis.

Constraint Handling for Multiple IPs

The diagram below gives clear examples of how instantiated IPs in the top level design will uniquify ports/pins/net names as it is referenced from the top level.

IP Modules can be instantiated within other hierarchies. This will typically cause implementation tools such as synthesis to rename the design objects within the IP module with hierarchy information. While renaming of design objects is a valid operation that preserves the design functionality it causes the timing/physical constraints to become invalid since the Name/ID of design objects referred in the constraint no longer matches the Name/ID of design objects in the design. This is how constraint propagation can help resolve the issue.



1. set_max_delay -from IP_A_Inst_3/REG_A -to IP_A_Inst_3/REG_B 5ns
2. set_max_delay -from SUB_A_Inst/IP_A_Inst_2/REG_A -to SUB_B_Inst/SUB_B_Inst/IP_A_Inst_2/REG_B 5ns
3. set_max_delay -from SUB_A_Inst/SUB_B_Inst/IP_A_Inst_1/REG_A -to SUB_A_Inst/SUB_B_Inst/IP_A_Inst_1/REG_B 5ns

Types of Constraint Resolution

In a typical IP flow design methodology, IP modules can be instantiated more than once at potentially multiple hierarchies and at different hierarchical depths. This can cause scenarios where there is a conflict in timing constraints especially if the timing constraints applies to design objects that cross IP boundaries. Constraints are either:

1. Resolved
   - Constraint preserved as written
   - Propagated and name adjusted
2. Ignored
   - Conflict with another constraint i.e. same electrical connection Constraint may be removed.

After constraint propagation finishes analyzing all the constraints, it outputs two files. A full report file detailing the reasons of each constraint if propagated and preserved or conflicted with another and ignored. The output reports file is generated in the implementations directory and called *CPEReport.txt.*

If constraint propagation decides that there are any constraints that it needs to replace or edit due to conflicts of hierarchical names, it will out put an intermediary constraint file with the new constraints written in *<project name>_<implementation name>_cpe.ldc*. This file will automatically be fed

into synthesis so that all the correct constraints can be processed correctly and proceed forward in the software design flow. If a user opens the post-synthesis constraint editor, all the processed constraints (i.e. removed or refined) are already reflected in this tool. A user cannot edit annotated pre-synthesis constraints but can add new post-synthesis constraints.

It should also be noted that:

1. Constraints at the IP-level may conflict with the design as a whole rather than a specific constraint, in which the constraint will be removed even if there is no similar constraint at the top level.

For example, *set_false_path - from [get_clocks clk1] -to [get_port o]* will be removed because it has a clock in it.

2. Constraints that are dropped from the IP-level will now still appear in the output .ldc file, but they will be commented out. This is to allow users to visually understand which constraints were dropped so they can examine them again if needed. The resources on these constraints will be updated top-level constraint names with hierarchy i.e. ‘top/ip1/a’ rather than just ‘a’ so theey can be copy/pasted and used in the top level if necessary.

From the example above, the output constraint will be: *#set_false_path - from [get_clocks clk1] -to [get_port o]*. The reasoning for why the constraint was removed will also be as specific as possible for the user in the CPEReport.txt file.

Processed Constraint Types

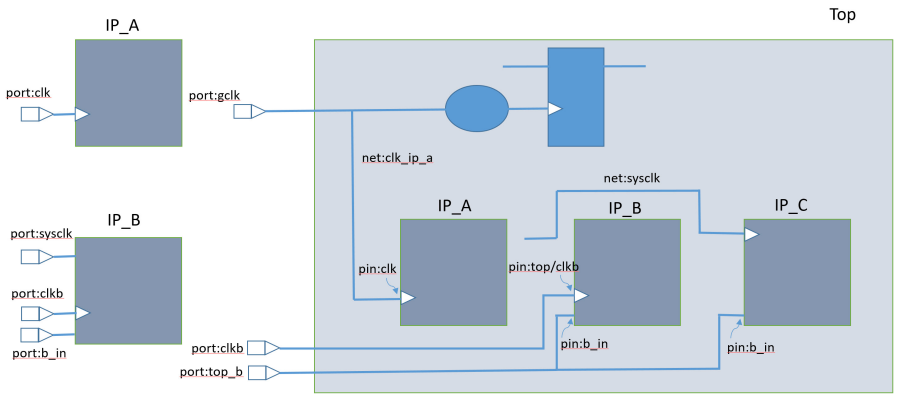
Constraint Propagation will only process the following constraint types. All other types will be passed through the flow to our other tools.

- create_clock
- create_generated_clock
- set_clock_groups
- set_clock_latency
- set_clock_uncertainty
- set_false_path
- set_input_delay
- set_output_delay
- set_load
- set_max_delay
- set_max_skew
- set_min_delay
- set_multicycle_path
- ldc_set_port

- ldc_set_location

Timing Constraint Resolution

The diagram depicts a sample design and the example table exhibits the analysis that constraint propagation will perform. constraint propagation will print out a resolution report (aforementioned) including the source constraint, source module of the constraint, resolution status, resolved constraint and the resolution method as shown in the table.



Note

In the following table, in the “User Action to Keep the Constraint” column, “Redefine” means to assess the IP-level constraint objects, map to top-level hierarchy which can be either simply adding hierarchy to the pin name or choosing a new top-level object, and then creating a new constraint at top-level with those objects.

Table 18: Timing Constraint Resolution Summary

	Constraint Input	Source Module	Resolution Status	Constraint Output	Resolution Method	User Action to Keep the Constraint
1	create_clock -name {clk_top} -period 5 [get_ports gclk]	Top	Resolved	create_clock -name {clk_top} -period 5 [get_ports gclk]	Constraint preserved	No action needed.
2	create_clock -name {clk_ip_a} -period 10 [get_ports clk]	IP_A	Ignored	Constraint #1	Conflict with Constraint #1	Confirm satisfaction with top-level clock.

Table 18: Timing Constraint Resolution Summary

	Constraint Input	Source Module	Resolution Status	Constraint Output	Resolution Method	User Action to Keep the Constraint
3	create_clock -name {clk_ip_b} -period 10 [get_ports clkb]	IP_B	Ignored	Defined on IP Input Port	Ignore. Warn user	Redefine IP-level clock on appropriate top-level port.
4	create_generated_clock -divide_by 2 -source [get_ports clkb] [get_pins b_out]	IP_B	Resolved	create_generated_clock -divide_by 2 -source [get_pins IP B/clkb] [get_pins IP B/b_out]	Propagated and name adjusted	No action needed.
5	set_input_delay -clock [get_clock sysclk] 9 [get_ports b_in]	IP_B	Ignored	Removed	IP-Level input delay not propagated	Redefine at top-level input port.
6	set_clock_groups [get_clocks_clk] -group [get_clocks clk2]	IP_B	Ignored	Removed	At least one clock is not internal	Translate to set_false_path and use appropriate user/ custom IP-level objects.
7	set_max_delay -from [get_ports b_in] -to [get_ports b_out] 5	IP_B	Resolved	set_max_delay -from [get_pins IP_B/b_in] -to [get_pins IP_B/b_out] 5	Max and Min delay always propagated	No action needed.

Constraint propagation processes all timing constraints that are entered in the original user constraint file.

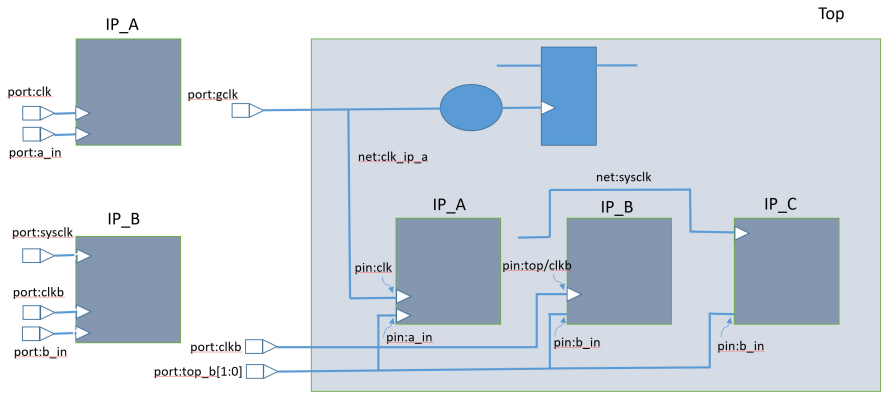
Physical Constraint Resolution

Constraint propagation will resolve two different physical constraints: “ldc_set_location” on page 817 and “ldc_set_port” on page 818. IP modules that can also have physical constraints such as “LOC” specified. All design objects with “LOC” constraint will be resolved with hierarchical names. LOC constraints now check for conflict with top layer, and will accept IP LOC constraints if it is electrically connected to top level port.

All physical attributes (“HDL Attributes” on page 732) that able to be set by ldc_set_port will also be analyzed and processed accordingly.

Table 19: Physical Constraint Resolution Summary

	Constraint Input	Source Module	Resolution Status	Constraint Output	Resolution Method
1	ldc_set_location -site A1 [get_ports top_b]	Top	Resolved	ldc_set_location -site A1 [get_ports top_b]	Constraint preserved
2	ldc_set_location -site B1 [get_ports b_in]	IP_B	Ignored	Constraint #1	Conflict with Constraint #1
3	Attribute example Module instance dsp_a /* synthesis LOC = “MULT_R1C9” */;	IP_B	Resolved	ldc_set_location -site MULT_R9C9 [get_cells IP_B/dsp_a]	Propagated, constraint preserved
4	ldc_set_port -iobuf {IO_TYPE=LVTTL33} [get_ports {top_b[0]}]	Top	Resolved	ldc_set_port -iobuf {IO_TYPE=LVTTL33} [get_ports {top_b[0]}]	Constraint preserved
5	ldc_set_port -iobuf {IO_TYPE=LVDS} [get_ports a_in]	IP_A	Ignored	Constraint #1	Conflict with Constraint #1
6	input [1:0] top_b /* synthesis IO_TYPE=LVDS */;	Top	Resolved	ldc_set_port -iobuf {IO_TYPE=LDVDS} [get_ports {top_b[1]}]	Propagated, conflicting port removed
7	ldc_set_port -iobuf {IO_TYPE=LVDS PULLMODE=UP} [get_ports b_in]	IP_B	Resolved	ldc_set_port -iobuf {PULLMODE=UP} {get_pins IP_B/b_in]	Propagated, conflicting type removed



Constraint propagation processes all timing constraints that are entered in the original user constraint file.

Constraint Resolution Specifics

Timing constraints have different situations and environments which they can be applied in. Sometimes a constraint that is valid on an IP module cannot be accepted even when updated to top-level resources due to restrictions on the constraint. The following restrictions have been placed on timing constraints which Constraint Propagation needs to follow.

- create_clock: Two create_clock constraints cannot exist on the same net. No two create_clock constraints can have the same -name setting. All IP-level clocks will have their -name changed to be <instanceName>_<originalName>
- create_generated_clock: This is the same as create_clock rule above.
- set_clock_latency / set_clock_uncertainty: When some or all of the clocks that these constraints have been applied to have been removed due to conflicts with top or other IP, to avoid constraining incorrect parts of the design, the constraint will be dropped.
- set_input_delay / set_output_delay: These constraints can only exist when applied to a top-level port. Constraint Propagation will remove all IP-Level set_input / output_delay constrains and warn the user.

Table 20: Constraint Resolution Specific Scenarios

Constraint Type (IP Level)	When it is Propagated	When it is Ignored
create_clock	When defined internally to the IP or on an output port.	When defined on IP-level input port or net driven by input port. Warn user to redefine at top-level.
create_generated_clock	All cases.	N/A
set_clock_groups	N/A	Always, warning given to user to redefine set_clock_groups at top level.
set_clock_latency	N/A	Always.
set_clock_uncertainty	Only when clock referenced is internal to IP.	When clock referenced is removed or if the clock is on input port.
set_input_delay	N/A	All cases.
set_output_delay	N/A	All cases.
set_max_delay	All cases.	N/A
set_max_skew	All cases.	N/A
set_min_delay	All cases.	N/A
set_false_path	Only when all clocks referenced are IP-level clocks and those clocks are not removed. A warning will always be issued.	When any of the clocks referenced are removed or when only one clock is referenced.

Table 20: Constraint Resolution Specific Scenarios

Constraint Type (IP Level)	When it is Propagated	When it is Ignored
set_multicycle_path	Only if there are no clocks referenced.	Whenever clocks are referenced.
set_load	N/A	All cases.

Attributes and Constraint Propagation

Constraint propagation will not convert attributes from within RTL to find physical constraints. Constraint propagation will act only on physical constraints present in the input .ldc files. These attributes will be processed by Lattice Synthesis engine and post-synthesis engines instead.

CDC (Clock Domain Crossing) Register

You can constrain CDC registers using the **CDC_register** attribute.

Attribute Usage Example

```
/* synthesis syn_preserve=1 CDC_Register=<int>*/
```

Note:

- For Avant device, the value of <int> could only be 1, 2,...12.
- For other devices, the value of <int> could only be 1, 2,...8.

Using the CDC_register attribute

When this attribute is applied on RTL registers, it will be forward annotated on flop primitives in the .vm netlist. The RTL registers with the **CDC_register** attribute will be preserved during synthesis and will not be used for timing optimization such as re-timing, pipelining, or packing into complex cells such as RAMs and DSP primitives. This attribute is not supported for the iCE40UP device.

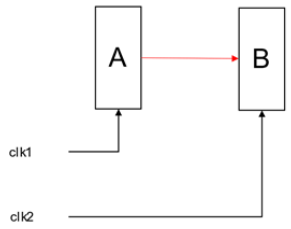
Note

The CDC_register attribute is converted to a set_max_delay constraint by the timing engine. The engine will go through all the CDC_register attributes and group registers that can be combined into a single set_max_delay constraint. The designer can use the set_max_delay constraint directly instead of adding the CDC_register attribute.

The following example shows the CDC_register=1 attribute on the B flip-flop

- It tells the timing engine to ignore the path between A and B.

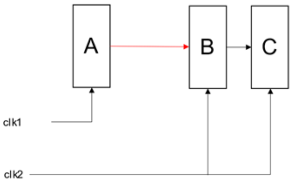
- Synthesis preserves and avoids optimizing the CDC register and implements it using a flip-flop.



Using CDC register with multi-flop synchronizers

This option lets you use more than one register for synchronization.

The following shows an example of CDC register with multi-flop synchronizers.



- If B and C are synchronization registers, and the CDC_register=2 is on the B flip flop, it can be used to inform Radiant about the two registers.
- The timing engine ignores the path from A to B.
- Synthesis preserves and avoids optimizing the CDC registers and implements them using individual flip-flops.
- The placement engine places B and C close to each other.
- The routing engine uses a short wire for the path from B to C.
- The clock domain crossing input pin of the B flop will not be considered for timing analysis. The timing engine will not report the path for A to B.

Note:

There must be no components between the CDC registers. The fanout of the nets between the registers B, C,…must be 1.

Limitations for Constraint Propagation

1. Parameterized module - constraint propagation flow does not support parameterized constraints at the top level.

2. Object Name Compatibility between LSE and Synplify - Internal design object names may be different between LSE and Synplify Pro flows. Hence, user constraints at the sub-module level for a synthesis tool cannot be guaranteed. This limitation does not apply to Lattice IP.

Post-Synthesis IP Constraint Propagation

This section describes the components and features of Post-synthesis IP constraint propagation in Radiant.

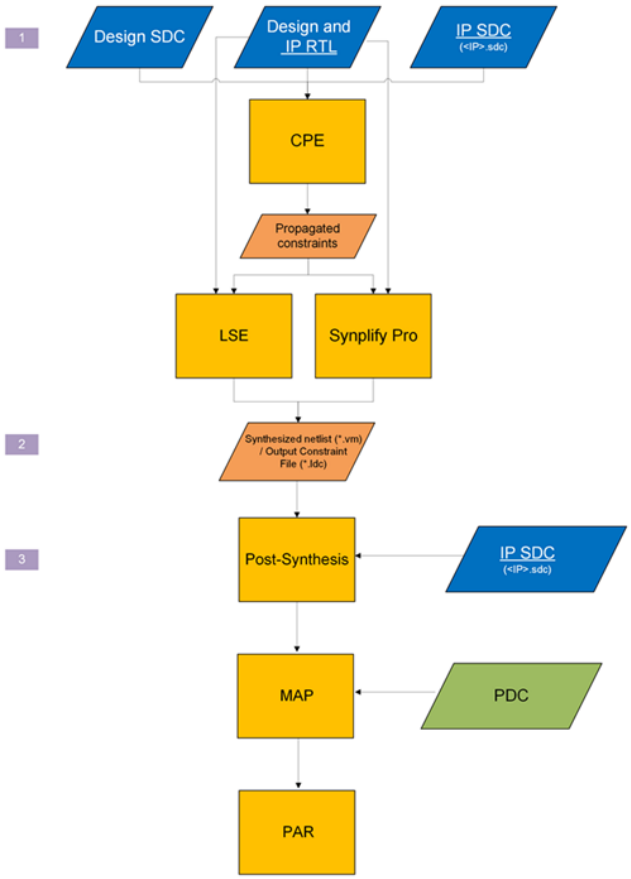
Overview

Post-synthesis IP constraint propagation in Radiant enables a single IP constraint file with Tcl statements. This supports multiple constraints files to specify different stages and synthesis tools through pre-defined Radiant variables.

For more information:

- “Understanding Implications of Radiant Software Constraint Flows” on page 199
- “Unified Constraints Flow” on page 196

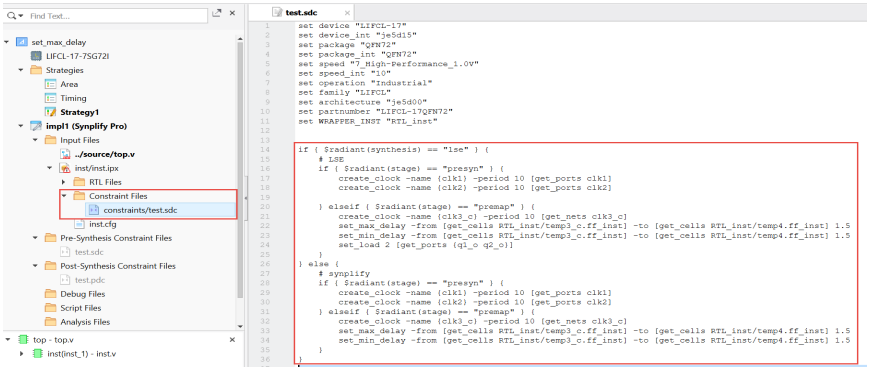
Post-Synthesis IP Constraint Propagation Process Overview



1. **Constraint File Parsing:** Design SDC, Design and IP RTL, and IP SDC files are parsed through Constraints Propagation Engine (CPE) which compiles all inputs from multiple .sdc files, from IP and user constraints and creates a propagated LDC file which will be consumed by Lattice Synthesis Engine (LSE) and Synplify Pro.
2. **Synthesis Output:** A synthesized structural Verilog file (*.vm) will be an outcome of the LSE and Synplify Pro.
3. **Post-Synthesis Stage:** The synthesized Verilog file (*.vm) and (*.ldc) constraints file will be used in the post-synthesis stage. You can also enter post-synthesis constraints using PDC file. Go to **Tools > Design Constraint Editor** and save as PDC (*.pdc) file, or directly edit PDC file which will be consumed by MAP and PAR.

Opening the Single Constraint File

To open the single constraint file in GUI, open your project and look for the .sdc file under ‘Constraint Files’ folder.



(*.sdc) Script Example

A single IP constraint file with Tcl scripting commands covers the synthesis tool for LSE and Synplify Pro, and presyn for pre-synthesis, and premap for post-synthesis design stage as shown in the reference code below.

```
if { $radiant(synthesis) == "lse" } {
  # LSE Configuration
  if { $radiant(stage) == "presyn" } {
  #Pre-synthesis stage constraints
    create_clock -name {clk1} -period 10 [get_ports clk1]
    create_clock -name {clk2} -period 10 [get_ports clk2]
  } elseif { $radiant(stage) == "premap" } {
#Post-synthesis stage constraints
    create_clock -name {clk3_c} -period 10 [get_nets clk3_c]
    set_max_delay -from [get_cells RTL_inst/temp3_c.ff_inst]
-to [get_cells RTL_inst/temp4.ff_inst] 1.5
    set_min_delay -from [get_cells RTL_inst/temp3_c.ff_inst]
-to [get_cells RTL_inst/temp4.ff_inst] 1.5
    set_load 2 [get_ports {q1_o q2_o}]
  }
} else {
  # Synplify Pro Configuration
  if { $radiant(stage) == "presyn" } {
  #Pre-synthesis stage constraints
    create_clock -name {clk1} -period 10 [get_ports clk1]
    create_clock -name {clk2} -period 10 [get_ports clk2]
  } elseif { $radiant(stage) == "premap" } {
  #Post-synthesis stage constraints
    create_clock -name {clk3_c} -period 10 [get_nets clk3_c]
    set_max_delay -from [get_cells RTL_inst/temp3_c.ff_inst]
-to [get_cells RTL_inst/temp4.ff_inst] 1.5
    set_min_delay -from [get_cells RTL_inst/temp3_c.ff_inst]
-to [get_cells RTL_inst/temp4.ff_inst] 1.5
  }
}
```

Constraint	Usage Instructions
$radiant(synthesis) == "lse"	When the current synthesis tool is ‘lse’.
$radiant(stage) == "presyn"	When the current stage is ‘presyn’.
create_clock -name {clk1} -period 10 [get_ports clk1]	Define clock constraint for clk1.
$radiant(stage) == "premap"	When the current stage is ‘premap’.
set_max_delay -from [get_cells RTL_inst/temp3_c.ff_inst]	Specify maximum delay for the timing paths.
set_false_path -from [get_ports {q}]	Specify paths that must be excluded from timing analysis.

See also:

- “Radiant Software Tool Tcl Command Syntax” on page 1577
- “Constraints Reference Guide” on page 731

Using Radiant Software Tools

The Radiant software provides tools for editing design constraints. These tools, which are available from the Radiant software toolbar and Tools menu, include the following:

- “Device Constraint Editor” on page 228
- “Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 265
- “Physical Designer” on page 274
- “Physical Designer: Routing Mode” on page 282

The Radiant software constraints tools enable you to develop constraints that shorten turn-around time and achieve a design that conforms to critical circuit performance requirements.

This section describes these Radiant software constraints tools and how to use them.

Device Constraint Editor

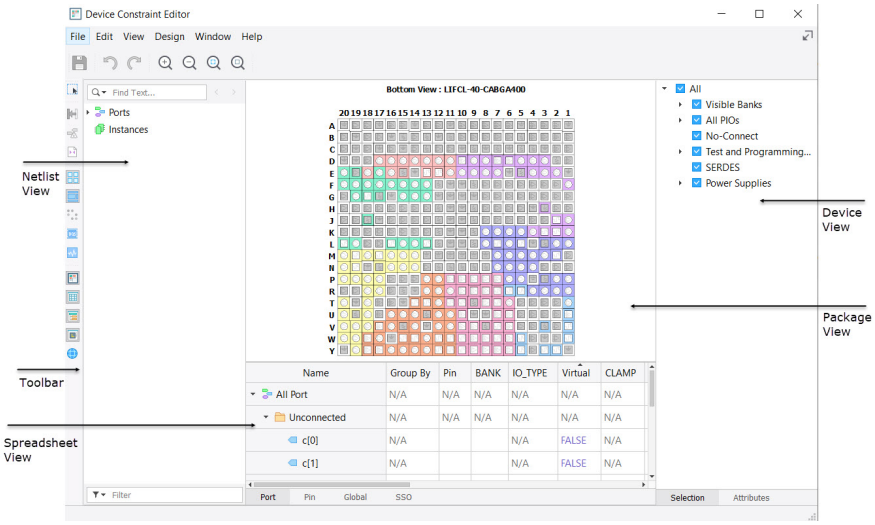
Device Constraint Editor is used to edit .pdc constraints. These constraint editing views are available from the Radiant software toolbar and Tools menu.

All modified constraints will be saved to a .pdc file and the flow will return to Map. The Device Constraint Editor shows the pin layout of the device and displays the assignments of signals to device pins. This view allows the user

to edit these assignments, reserve sites on the layout to exclude from placement and routing. The Device Constraint Editor is also the entry mechanism for physical constraints.

Device Constraint Editor views enable you to develop constraints that will shorten turn-around time and achieve a design that conforms to critical circuit performance requirements.

The following figure shows the Device Constraints Editor.



The Device Constraints Editor allows you to do the following tasks:

- Define physical 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.
- Perform DRC tests to ensure legal pin assignments or to detect incomplete or illegal conditions within the physical design database.
- Back annotate post-route pin assignments to a logical constraint file.

Viewing Pin, Port and Global Assignments in Device Constraint Editor

The Device Constraint Editor has a spreadsheet view that provides a tabular format for viewing and assigning logical constraints. It displays all constraints, including those that have been assigned.

As soon as you have specified the target device, you can use this view to set global constraints. After synthesis and translation, you can use all of the following constraint sheets to create and modify constraints:

Port The Port tab provides a signal list of the design and shows any pin assignments that have been made.

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 on Outputs, for example, by closing the display of Input ports or vice versa.

The Port tab 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 Ports dialog box.

When the Assign Ports dialogue appears, the pins from the right side of the window can be dragged into the desired port shown in the left hand side of the window.

Pin The Pin tab 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.

You can show differential pin pairs by selecting **View > Show Differential Pairs**.

The Pin tab enables you to edit signal assignments or assign new signals by right-clicking selected pins and opening the Assign Pins dialog box.

Global The Global tab provides general device information such as junction temperature, voltage, derating, VCCIO (and banks), Global Set/Reset nets and Use Primary Net to select the primary clock or critical nets. When the Global tab is selected, the package view is no longer shown unlike when the Pin or Port tab is selected.

See Also

- “Assigning Signals” on page 284
- “Setting Port, Net, and Cell Attributes” on page 287
- “Setting Global Settings” on page 241

Viewing Package View and Pin Layout in the Device Constraint Editor

The Device Constraint Editor has a Package View that shows the pin layout of the device and displays the assignments of signals to device pins. The Package View interacts with the Netlist View, which is a listed on the left side of the screen, to assign pins that enable you to drag selected signals to the desired locations on the pin layout.

Each pin that is assigned with a constraint is color-coded to indicate the port direction of the related signal port. The 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.

The Netlist view displays the RTL netlist objects including ports, instances and nets from RTL view. It will be docked in the package view for easy signal assignment. Selected signals from this view.

See Also ▶"Modifying Signal Assignments using the Device Constraint Editor" on page 286

- "Using the Zoom and Pan Tools in Device Constraint Editor" on page 238

Using the "Show" Commands

When Device Constraint Editor is open, the View menu 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 Only one command can be used at a time:

- Show Bank IO
- Show Port Group
- Show DQS Group
- Show SSO Group

Non-mutually Exclusive Commands The following commands can be activated while another "Show" command is active:

- Displaying I/O Placement Assignments in Device Constraint Editor Package View
- Show Differential Pairs

Displaying Bank I/O Assignments

By default, Package View displays any assigned I/Os within each color-coded bank. However, the bank I/O display is turned off when the display for Port groups or DQS groups is turned on.

To display bank I/Os:

- In Device Constraint Editor, choose **View > Show Bank IO**.

  The I/O assignments are displayed within each bank. Port groups and DQS groups, and SSO groups are hidden.

Displaying Port Groups in Device Constraint Editor

Device Constraint Editor 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 the Spreadsheet View.

To display port groups:

- In Device Constraint Editor, choose **View > Show Port Group**.

  The color-coded port groups are displayed. Bank I/O assignments, DQS groups, and SSO groups are hidden.

Displaying DQS Groups in Device Constraint Editor

You can view the placement of DQS groups in color-coded format on Device Constraint Editor pin layout by turning on the “Show DQS Groups” command.

To display DQS Groups:

- In Device Constraint editor, choose **View > Show DQS Group**.

  The color background of each DQS group is displayed. Bank I/O assignments, port groups, and SSO groups are hidden.

Displaying SSO Groups in Device Constraint Editor

The “Show SSO View” command, available from Device Constraint Editor, 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 Device Constraint Editor, 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.

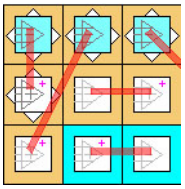
Displaying Differential Pin Pairs in Device Constraint Editor

The “Show Differential Pairs” command enables you to identify all the differential pin pairs on a 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 Device Constraint Editor, 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. Select **View > Show Differential Pairs** to turn off the display.

See Also

- “Prohibiting Incompatible Pins” on page 288

Displaying I/O Placement Assignments in Device Constraint Editor Package View

After placement and routing, Package View enables you to examine all pin assignments, 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 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. The border color around the box indicates the Bank info which is shown in the right hierarchy view.Pin assignments that originated from the .pdc file are displayed with a color fill. Blue is output signal; green is input signal.



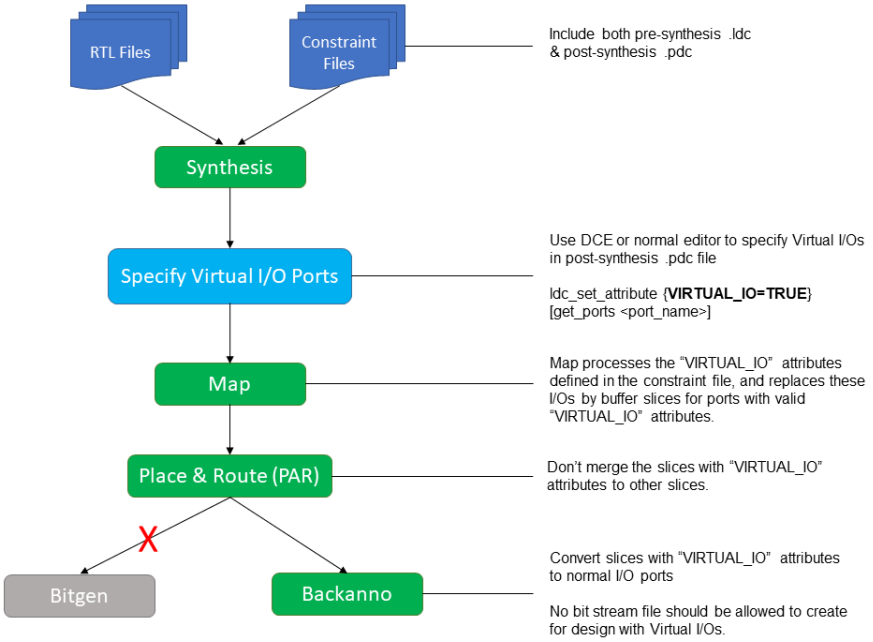
The “Display IO Placement” command acts as a toggle key and displays a check mark when the assignments are displayed. Select the command again to turn off the display.

When you edit pin assignments after placement and routing and save them, the placement indicators are cleared from Device Constraint Editor. The “Display IO Placement” command is then unavailable until you rerun Place & Route.

Specifying Virtual I/O Ports

The Radiant Virtual I/O flow is useful in checking the design timing, resource utilization, or performance of a selected device to determine if your modules exceeded the number of physical pins present in the FPGA (e.g., during an IP design). Without the Virtual I/O flow, the implementation will result in an error due to the number of I/Os exceeding the device capacity. You can use the Virtual I/O flow to specify certain ports as "virtual" ports at the top level.

Refer to the following flow chart.



A Virtual I/O is terminated in the LUT instead of a PIO. This is reported as the "Number of feedthrough LUT4s" in the MAP report, and is not included in the number of LUTs in the design implementation.

If the total number of real ports in the design is less than or equal to the total number of I/O ports in the device, you can use the regular Radiant flow for such IP without the Virtual I/O flow.

To specify the Virtual I/O ports in the selected I/Os:

- In the .pdc file using ldc_set_attribute.

Syntax: `ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports <port_name>]`

```
ldc_set_attribute {VIRTUAL_IO = TRUE} [get_ports ch0_sdrxp]
```

Notes:

Wildcard "*" is supported for the port names:

```
ldc_set_attribute {VIRTUAL_IO = TRUE} [get_ports datain*]
```

- Using the DCE Spreadsheet View Virtual column, set the required ports value to "TRUE." By default, the value is set to "FALSE."

See Also

- “Checking the Virtual I/O Ports” on page 235
- “Verifying Virtual I/O Ports” on page 236

Checking the Virtual I/O Ports

The Virtual I/O attribute for a port is ignored if one of the following conditions is met:

- The I/O port is with tristate signal.

  e.g. tristate output or bi-directional I/O. Tristate support is only in I/O port.
- The I/O port is with LOCATION constraint.

  When user specifies the location (either to pin or bank), it is a real external I/O.
- The I/O port is with IO_TYPE attribute in RTL or constraint.

  When user specifies IO_TYPE, it is a real external I/O. Virtual I/O should not be specified for the ports with IO_TYPE attribute.
- The I/O port is connected with DELAY module.
- The I/O port is not a simple input buffer (IB) or output buffer (OB) but a special I/O buffer, either from an IP or RTL instantiation.

  e.g. MIPI, SEIO18_CORE, DIFFIO18_CORE, etc.

  Its special usage means it is a real external I/O.
- The I/O port drives a clock signal or is driven by a clock signal.
- The I/O port is connected to an external I/O directly in a hardware device, such as:
  - APIO pin of a component (e.g. High speed I/O in PCIE)
  - JTAG ports
  - Some ports of ADC/PMU/I2C etc. components
  - IDDR's D input
  - ODDR's Q output
  - DQS's DQSI input

Notes:

I/O registers are treated differently in the LFCPNX (CertusPro-NX) and UT24CP (CertusPro-NX-RT) device versus the iCE40UP (iCE40 UltraPlus) device. For the LFCPNX and UT24CP device, IOLOGIC is a whitebox cell model. I/O registers are mapped into IOLOGIC. So, when IREG's D input or OREG's Q output is a Virtual I/O port, the I/O REG will be pushed into PLC (SLICE).

However, for the iCE40UP device, I/O register is in IOLOGIC black box. It is packed by the synthesis tool. As a result, the Virtual I/O constraints will be ignored for ports associated with I/O registers.

See Also

- "Specifying Virtual I/O Ports" on page 234
- "Verifying Virtual I/O Ports" on page 236

Verifying Virtual I/O Ports

There are several ways to verify Virtual I/Os.

Viewing Virtual I/Os in DCE If you have specified the Virtual I/O attributes for some I/O ports in the Post-Synthesis constraint file (.pdc), you can see them in the "Virtual" column. As shown in the following figure, the ports "datain[0]" to "datain[7]" are virtual I/Os.

Name	Group By	Pin	BANK	IO_TYPE	Virtual	CLAMP
clk	N/A	(E12)	(0)	LVCMOS33(LVCM...	FALSE	ON(ON)
datain[0]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[1]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[2]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[3]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[4]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[5]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[6]	N/A	N/A	N/A	N/A	TRUE	N/A
datain[7]	N/A	N/A	N/A	N/A	TRUE	N/A

Changing Virtual I/O attributes in DCE If you save the result, the Radiant software will ask you to rerun the Map Design and Place & Route Design processes, since the .pdc file has been changed.

Viewing Virtual I/Os in the Map Report file The number of Virtual I/Os are reported in the Map Report (.mrp) file. Here is an example:

```
Number of LUT4s:                    8 out of 32256 (<1%)
```

```
Number used as logic LUT4s:          0
Number used as distributed RAM:      0 (6 per 16X4 RAM)
Number used as ripple logic:         0 (2 per CCU2)
Number of inserted feedthrough LUT4s:8
Number for virtual IO:               8
Number of Virtual IOs:               8
```

The report shows that the "Number of Virtual I/Os" is 8, and it used 8 LUTs to replace the specified Virtual I/O ports. These eight LUTs are part of "Number of inserted feedthrough LUT4s."

You can find the individual Virtual I/Os in the "I/O (PIO) Attributes" section in the same .mrp file. If a port is a Virtual I/O, the value is marked as "VIRTUAL" in column "Levelmode IO_TYPE."

I/O Name	Direction	Levelmode IO_TYPE	I/O REG	I/O DDR	Special I/O Buffer
clk	INPUT	LVCMOS33			
datain[7]	INPUT	**VIRTUAL**			
datain[6]	INPUT	**VIRTUAL**			
datain[5]	INPUT	**VIRTUAL**			

See Also

- “Specifying Virtual I/O Ports” on page 234
- “Checking the Virtual I/O Ports” on page 235

Changing Device Constraint Editor 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 by going to **Tool > Options, Color** tab on the left and **Device** tab in the main pain. The Color Legend Tab 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. Go to **Tool > Options, Color** tab on the left and **Device** tab in the main pain.

2. Double-click the color box under the Color column of the desired bank or pin type listed in the Name column that you want to change.
3. In the Select Color dialog box, select a different color and click **OK**.
4. If you want to change more than one color, click **Apply** in the dialog box, and then repeat Steps 3 and 4. To revert to the default colors, click **Restore**.
5. When you have finished color editing, click **Close**.

Filtering the Display of Banks and Pin Types in Device Constraint Editor

When you open Device Constraint editor, it shows all banks and pin types by default. In the Radiant software, the window to the right of the main device view shows the banks and pin types so that only selected banks and pin types are displayed via the Selection and Attributes tabs.

To filter the display of banks and pin types in Package View:

1. On the right hand side select Selection tab, you can check or uncheck the different types of IOs.

See Also

- “Using the “Show” Commands” on page 231

Reversing the Pin Layout View in Device Constraint Editor

You can reverse the view of Package View 's device pin layout from a bottom to top perspective, or vice versa.

To reverse the pin layout view:

- From the Device Constraint Editor, 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.

Using the Zoom and Pan Tools in Device Constraint Editor

The Radiant software toolbar provides zoom tools for the Device Constraint Editor. In addition, the vertical toolbar included with Package View and Placement Mode provides commands that work together with the zoom tools to help you navigate to different objects and areas of the layout. Also, both Routing Mode and Logic Block View provide an area zoom that is controlled

through the left mouse button. All of the views enable you to zoom in and out using the mouse wheel and function key combinations, and to pan using the arrow keys.

Radiant Software Toolbar Zoom Commands The following commands are available on the Radiant software 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 view to only show a selected pin.

Area Zoom with the Mouse in Device Constraint Editor and Placement MODE In Routing Mode 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. Area Selection Mode – 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.

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++ (numeric keyboard)
- Zoom Out – Ctrl+- (numeric keyboard)

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. You can also press the ‘Ctrl’ key and move your mouse while pressing the left mouse button.

Searching for Design Elements in Device Constraint Editor

Device Constraint Editor and Placement Mode enable you to search for design elements—either through a Find dialog box or through Find and Filter text boxes within the view.

Each provide Find and Filter text boxes at the top of the window. Radiant software highlights the found objects and enables you to navigate through them and cross-probe to other views.

Device Constraint Editor and Placement Mode each provide a Find dialog box. The Find dialog box in Placement Mode enables you to search for components, nets, and sites.

To search for design elements in Device Constraint Editor:

1. Open the desired view.
2. The Find text box appears at the top of the window.
3. To narrow the scope of the search, use wildcards in the following ways.
   - 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.

Using Drag-and-Drop to Make Assignments

Radiant software constraint-editing views provide a convenient drag-and-drop feature for making assignments or editing them. This feature is available between Device Constraint Editor and Placement Mode. The drag-and-drop feature is also available within Package View and within Placement Mode. This feature allows you to perform the following actions:

- Drag signals from Netlist View to Package View to assign them to pin locations all within Device Constraint Editor.
- Drag assigned signals within Device Constraint Editor to reassign them to other locations.
- Drag EBR, DSP, PCS, PLL, and ASIC block instances from Netlist View to Placement Mode all within Device Constraint Editor to set constraints.
- Drag non-SLICE logical components within Placement Mode to reassign them to other locations.
- Drag GROUPs and REGIONS within Placement Mode to reassign them to other locations.

Undoing and Redoing Assignments

Radiant software **Undo** and **Redo** commands allow you to reverse constraint changes that were made in a selected constraint 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 constraint changes, the undo and redo commands are disabled.

Note

The **Undo** command will not remove the in-memory constraint change indicators: the asterisk from a constraint view tab or the “constraints 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 Radiant software toolbar.

The standard Windows keyword shortcuts are also available:

- Press **Ctrl+Z** to undo an assignment.
- Press **Ctry+Y** to redo an assignment.

Printing a Constraint View from Device Constraint Editor

You can print any from the Device Constraint Editor and Placement Mode. The Radiant software provides a preview for each view, enabling you to select options before printing.

Printing a Layout View You can print a copy of the layout from Placement Mode or Device Constraint Editor.

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 **Printer** icon to print.

Setting Global Settings

In the Device Constraint Editor tool at the bottom of the tool, there are a series of tabs to allow the user to set many of the device settings such as Junction Temperature, Voltage, SysConfig, User Code, Deratings, Bank VCCIO, Global Set/Reset, Use Primary Net and Vref Locate Constraints.

Junction Temperature Setting

Assigns an operating junction temperature for the part that changes the derating factor for timing. You can use this constraint 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.

To set Junction Temperature value:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Enter Junction Temperature value in the text box.

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

Syntax

ldc_set_attribute {TEMPERATURE=<value>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 65 degrees Celsius.

```
ldc_set_attribute {TEMPERATURE=65C}
```

Voltage Setting

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.

To set voltage value:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Enter a Voltage value in the text box.

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP

Syntax ldc_set_vcc -core <value>

A space between the number and the voltage symbol is optional. See example.

Example The following constraint command assigns a value for nominal voltage of 0.95 V.

```
ldc_set_vcc -core 0.95
```

sysCONFIG Settings

Defines system configuration option settings for the sysCONFIG feature. If you do not specify these settings in the .pdc file, using the Global tab in Device Constraint Editor or manually, some default sysCONFIG constraints will automatically be generated based on the device selection.

The SYSCONFIG constraint 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 configurations streams. The devices also support daisy chaining.

To set SysConfig parameter values:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. In the SysConfig drop-down setting, for each parameter in the value column, double-click to display the drop-down to select a value or enter the actual value.

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 constraint. They are ordered as they appear in the Global tab in the Device Constraint Editor.

Device Support

LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

Syntax

ldc_set_sysconfig <keyword>=<value>+

You should set the constraint configuration using the Device Constraint Editor or by editing the .pdc file directly. If you do not set sysCONFIG options, default SYSCONFIG constraints are automatically set.

Examples

The SYCSCONFIG constraint allows you to set a series of keyword values in succession after the ldc_set_sysconfig tcl command.

Figure 23: Example of a SYSCONFIG in the .pdc constraint file

```
ldc_set_sysconfig {JTAG_PORT=ENABLE PROGRAMN_PORT=ENABLE
MCCLK_FREQ=56.2}
```

Table 21: sysCONFIG Values

Keyword	Default and Selectable Values	Supported Devices
SLAVE_SPI_PORT	**DISABLE** [DISABLE, SERIAL, DUAL, QUAD] for Nexus Devices **DISABLE** [DISABLE, SERIAL, DUAL, QUAD, XSPI] for Avant Devices	LAV-AT, LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
MASTER_SPI_PORT	**DISABLE** [DISABLE, SERIAL, DUAL, QUAD] for Nexus Devices **DISABLE** [DISABLE, SERIAL, DUAL, QUAD, XSPI, XSPI_DIFF_CLK] for Avant Devices	LAV-AT, LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
SLAVE_I2C_PORT	**DISABLE** [DISABLE, ENABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
SLAVE_I3C_PORT	**DISABLE** [DISABLE, ENABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
JTAG_PORT	**ENABLE** [ENABLE, DISABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
DONE_PORT	**ENABLE** [ENABLE, DISABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
INITN_PORT	**ENABLE** [ENABLE, DISABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
PROGRAMN_PORT	**ENABLE** [ENABLE, DISABLE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
BACKGROUND_RECONFIG	**OFF** [OFF, ON, SRAM_EBR, SRAM_ONLY] for Nexus Devices **OFF** [OFF, SRAM_ONLY, ON] for Avant Devices	LAV-AT, LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
DONE_EX	**OFF** [OFF, ON]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
DONE_OD	**ON** [OFF, ON]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
MCCLK_FREQ	See MCCLK section below	LAV-AT, LFCPNX, LFD2NX, LIFCL, UT24C, UT24CP

Keyword	Default and Selectable Values	Supported Devices
FLASH_CLK_FREQ	**3.5** [3.5, 7.0, 14.1, 28.1, 56.2, 75, 90, 112.5]	LFMXO5
TRANSFR	**OFF** [OFF, ON] for LAV-AT, LFCPNX, LFMXO5, LIFCL-33U, and UT24CP **OFF** [OFF] for LFD2NX, LIFCL, and UT24C	LAV-AT, LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
CONFIG_IOSLEW	**SLOW** [SLOW, MEDIUM, FAST]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
CONFIGIO_VOLTAGE_BANK0/1	**NOT_SPECIFIED** [NOT_SPECIFIED, 2.5, 1.2, 1.5, 1.8, 3.3] for Nexus devices	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
CONFIG_SECURE	**OFF** [OFF, ON]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
WAKE_UP	**ENABLE**[DISABLE]**_DONE_SYNC**	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
COMPRESS_CONFIG	**OFF** [OFF, ON]	LAV, LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
EARLY_IO_RELEASE	**OFF** [OFF, ON]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
BOOTMODE	**DUAL** [DUAL, SINGLE, NONE]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP
MASTER_PREAMBLE_TIMER_CYCLES	**600000** [600000, 400, 2000, 4000, 20000, 40000, 200000, 400000]	LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP

Table 22: Avant sysCONFIG Values

Keyword	Default and Selectable Values	Supported Devices
BOOT_SEL	**DUAL** [DUAL, SINGLE]	LAV-AT
MSPI_RESET	**DISABLE** [DISABLE, ENABLE]	LAV-AT
MSPI_RESET_PORT	**DISABLE** [DISABLE, ENABLE]	LAV-AT
MULTI_BOOT_MODE	**DISABLE** [DISABLE, ENABLE]	LAV-AT
DAISY_CHAIN	**DISABLE** [DISABLE, BYPASS, FLOW_THROUGH]	LAV-AT
DAISY_CHAIN_WAIT_DONE	**DISABLE** [DISABLE, ENABLE]	LAV-AT
MSPI_SIGNATURE_TIMER	**200MS** [200MS, 100MS, 50MS, 40MS, 20MS, 1MS, 500US, 100US, INFINITE]	LAV-AT
MSPI_CPOL	**IDLE_CLOCK_LOW** [IDLE_CLOCK_LOW, IDLE_CLOCK_HIGH]	LAV-AT
MSPI_RX_EDGE	**RISING** [RISING, FALLING]	LAV-AT
MSPI_TX_EDGE	**FALLING** [FALLING, RISING]	LAV-AT

Table 22: Avant sysCONFIG Values

Keyword	Default and Selectable Values	Supported Devices
MSPI_CPHA	**FIRST_EDGE** [FIRST_EDGE, SECOND_EDGE]	LAV-AT
SSPI_RX_EDGE	**RISING** [RISING, FALLING]	LAV-AT
SSPI_TX_EDGE	**FALLING** [FALLING, RISING]	LAV-AT
SSPI_CPHA	**FIRST_EDGE** [FIRST_EDGE, SECOND_EDGE]	LAV-AT
SSPI_CPOL	**IDLE_CLOCK_LOW** [IDLE_CLOCK_LOW, IDLE_CLOCK_HIGH]	LAV-AT
SSPI_SHIFT_ORDER	**MSB_FIRST** [MSB_FIRST, LSB_FIRST]	LAV-AT
SSPI_DAISY_CHAIN_MODE	**DISABLE** [DISABLE, SCM, AUTO]	LAV-AT
ERASE_EBR_ON_REFRESH	**ENABLE** [ENABLE, DISABLE]	LAV-AT
SIGNATURE_CHECK	**ENABLE_LSCC_SIGNATURE** [ENABLE_LSCC_SIGNATURE, DISABLE, ENABLE_SFDP_SIGNATURE]	LAV-AT
MSPI_ADDRESS_32BIT	**DISABLE** [DISABLE, ENABLE]	LAV-AT
MSPI_COMMAND_32BIT	**DISABLE** [DISABLE, ENABLE]	LAV-AT
SSPI_IDLE_TIMER	**DISABLE** [DISABLE, 200S, 100S, 50S, 25S, 10S, 5S, 1S, 750MS, 500MS, 250MS, 100MS, 75MS, 50MS, 25MS, 10MS]	LAV-AT
MSPI_PREAMBLE_DETECTION_TIMER	**200MS** [200MS, 100MS, 50MS, 40MS, 20MS, 1MS, 500US, 100US]	LAV-AT
MULTI_BOOT_SEL	**STATIC** [STATIC, DYNAMIC]	LAV-AT
CONFIGIO_VOLTAGE_BANK1/2	**NOT_SPECIFIED** [NOT_SPECIFIED, 2.5,1.2,1.8, 3.3]	LAV-AT

SLAVE_SPI_PORT

DISABLE (default) | SERIAL | DUAL | QUAD – Nexus Devices

DISABLE (default) | SERIAL | DUAL | QUAD | XSPI – Avant Devices

Specifies if the TARGET_SPI port should be available after configuration. This option will preserve the dual-purpose pins for PAR.

MASTER_SPI_PORT

DISABLE (default) | SERIAL | DUAL | QUAD – Nexus Devices

DISABLE (default) | SERIAL | DUAL | QUAD | XSPI | XSPI_DIFF_CLK – Avant Devices

Specifies if the CSPI port should be available after configuration. Used for background programming of the connected SPI flash. This option will preserve the dual-purpose pins for PAR. When enabled, it allows the use of the external Controller SPI port for configuring the SRAM fuses using the bitstream.

Note

As part of the Lattice inclusive language guidelines, the Avant device's MASTER_SPI_PORT has been changed to CONTROLLER_SPI_PORT and SLAVE_SPI_PORT to TARGET_SPI_PORT. If you want to know more about the Lattice Inclusive Language, please visit Lattice Answer Database.

SLAVE_I2C_PORT

DISABLE (default) | ENABLE

Specifies if the SLAVE I2C port should be available after configuration. This option will preserve the dual-purpose pins for PAR.

SLAVE_I3C_PORT

DISABLE (default) | ENABLE

Specifies if the SLAVE I3C port should be available after configuration. This option will preserve the dual-purpose pins for PAR.

JTAG_PORT

ENABLE (default) | DISABLE

Specifies if the JTAG port should be available after configuration. This option will preserve the dual-purpose pins for PAR. When ENABLED, it is used for configuration settings only, PAR will prohibit them. When DISABLED, it can be used as GPIOs.

DONE_PORT

ENABLE (default) | DISABLE

When set to ENABLE, the pin is not available in user mode.

INITN_PORT

ENABLE (default) | DISABLE

When set to ENABLE, the pin is not available in user mode.

PROGRAMN_PORT

ENABLE (default) | DISABLE

When set to ENABLE, the pin is not available in user mode.

BACKGROUND_RECONFIG

OFF (default) | ON | SRAM_EBR | SRAM_ONLY – Nexus Devices

OFF (default) | SRAM_ONLY | ON – Avant Devices

DONE_EX

OFF (default) | ON

You can select if the device should wake up on its own after the DONE bit is set or wait for an external DONE signal to drive the DONE pin high. The DONE_EX attribute will determine if the wake up sequence is driven by an external DONE signal.

If set to ON, the user wishes to delay wake up until the DONE pin is driven high by an external signal and synchronous to the clock. You can select OFF if you want to synchronously wake up when the DONE bit is set and ignore any external driving of the DONE pin.

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.

The DONE pin used in sysCONFIG to indicate that configuration is done. The DONE pin can be linked to other DONE pins and used to initiate the wake up process. The DONE pin can be open collector or active drive. Default is ON and insures the user needs to make a conscious decision to drive the pin.

MCCLK_FREQ

3.5 (default) | 7.0 | 14.1 | 28.1 | 56.2 | 75 | 90 | 112.5 | 150 – Nexus devices.

3.1 (default) | 7.1 | 14.3 | 28.6 | 57.1 | 66.7 | 80.0 | 100.0 | 106.7 | 133.3 | 160.0 – Avant devices.

Controls the Master Clock frequency. When a master configuration mode is used, MCLK will become an output clock with the frequency set by the user. Until the device is configured, the default Master Clock frequency is the lowest frequency support by the device.

For LFCPNX, LFD2NX,LIFCL, UT24C, and UT24CP devices, a 450 Mhz oscillator is divided and made available for configuration.

For LAV-AT devices, a 400 Mhz oscillator is divided and made available for configuration.

FLASH_CLK_FREQ

3.5 (default) | 7.0 | 14.1 | 28.1 | 56.2 | 75| 90 | 112.5

TRANSFR

OFF (default) | ON

Enables TransFR (Transparent Field Reconfiguration).

CONFIG_IOSLEW

SLOW (default) | MEDIUM | FAST

Controls config output pin slew rate. Sets CR0[7:6].

CONFIGIO_VOLTAGE_BANK0/1

NOT_SPECIFIED (default) | 2.5 | 1.2 | 1.5 | 1.8 | 3.3 – Nexus Devices

The CONFIGIO_VOLTAGE_BANK0/1 attribute supports all devices except LAV-AT and iCE40UP.

Setting this attribute informs the software which voltage is required in Bank0/1 to satisfy the user's sysCONFIG requirements. The sysCONFIG pins used for configuration may or may not be used in the design and hence the user shall be able to declare the voltage interface for this bank.

DRC errors can then be generated based on CONFIG_IOVOLTAGE and usage of the dual-purpose sysCONFIG pins. BANK0 and 1 values can be different and a combination of the available values including NOT_SPECIFIED.

CONFIGIO_VOLTAGE_BANK1/2

NOT_SPECIFIED (default) | 2.5 | 1.2 | 1.8 | 3.3

The CONFIGIO_VOLTAGE_BANK1/2 attribute only supports the LAV-AT devices.

CONFIG_SECURE

OFF (default) | ON

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. Prevents readback of the configuration memory contents.

WAKE_UP

ENABLE/DISABLE_DONE_SYNC (DONE_EX = OFF) /(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.

COMPRESS_CONFIG

OFF (default)

Bitstream Compression. When set to ON, for those devices that support bitstream compression, the software generates a compressed version of the bitstream file.

EARLY_IO_RELEASE

OFF (default) | ON [once]

Program and wakeup the left/right I/O banks early, before the rest of the device is programmed.

BOOTMODE

DUAL (default) | SINGLE | NONE

This setting enables you to select the booting mode at power up.

- DUAL - Performs the dual boot for the booting event. In case the primary booting image (bitstream) fails, the secondary boot image is invoked to boot the device.
- SINGLE - Performs a single boot only. If it fails, the device will move to unprogrammed mode directly.
- NONE - No controller SPI booting at startup, waits for target configuration port to configure the device.

MASTER_PREAMBLE_TIMER_CYCLES

600000 (default) | 400 | 2000 | 4000 | 20000 | 40000 | 200000 | 400000

This attribute sets the master preamble timer count value.

The preamble_count value below is the number of clock cycles to get a valid preamble before an error is declared.

000 -> 600000

001 -> 400000

010 -> 200000

011 -> 40000

100 -> 20000

101 -> 4000

110 -> 2000

111 -> 400

FLASH_PREAMBLE_TIMER_CYCLES

600000 (default)

This attribute sets the flash preamble timer count value.

The preamble_count value is the number of clock cycles to get a valid preamble before an error is declared.

BOOT_SEL

DUAL (default) | SINGLE

The BOOT_SEL preference allows you to select the booting mode (DUAL or SINGLE) at power up (POR), PROGRAMN pin toggle or REFRESH command execution with the following available options:

- DUAL (default) - Performs the dual boot for the booting event. In case the primary booting image (bitstream) failed, the secondary booting image is automatically invoked to boot up the device.
- SINGLE - Performs the single boot only. If it fails, the device goes directly to unprogrammed mode.

MSPI_RESET

DISABLE (default) | ENABLE

Enables the hardware reset signal to the flash device.

The MSPI_RESET attribute has the following options:

- DISABLE - Enables reset port to flash device is disabled.
- ENABLE - Enables reset port to flash device.

MSPI_RESET_PORT

DISABLE (default) | ENABLE

Specifies if the MSPI_RESET port is available to use for configuration purposes during usermode.

The MSPI_RESET_PORT attribute has the following options:

- DISABLE - Not available for configuration during usermode (user IO).
- ENABLE - Available for configuration during usermode (persisted as config IO).

MULTI_BOOT_MODE

DISABLE (default) | ENABLE

The MULTI_BOOT_MODE preference allows you to enable the multi-boot feature. It has the following options:

- DISABLE - Disables multi-boot feature.
- ENABLE - Enables the multi-boot feature.

DAISY_CHAIN

DISABLE (default) | BYPASS | FLOW_THROUGH

You can select the daisy chain mode for upstream device using DAISY_CHAIN preference with the following available options:

- DISABLE - Daisy Chain is disabled (default).
- BYPASS - Enables the transparent SRAM access mode.
- FLOW_THROUGH - Enables the transparent SRAM or/and INIT Registers access mode.

DAISY_CHAIN_WAIT_DONE

DISABLE (default) | ENABLE

The DAISY_CHAIN_WAIT_DONE preference allows you to wait for the done pin before “wakeup” for the daisy chain upstream. It has the following options:

- DISABLE (default) - Does not wait for DONE.
- ENABLE - Waits for DONE.

MSPI_SIGNATURE_TIMER

200MS (default) | 100MS | 50MS | 40MS | 20MS | 1MS | 500US | 100US | INFINITE

MSPI_SIGNATURE_TIMER allows you to specify the time to get a valid LSCC/SFDP signature.

MSPI_CPOL

IDLE_CLOCK_LOW (default) | IDLE_CLOCK_HIGH

MSPI_CPOL specifies if the clock is low or high in idle state.

- IDLE_CLOCK_LOW (default) - Clock is low in idle state.

- IDLE_CLOCK_HIGH - Clock is high in idle state.

MSPI_RX_EDGE

RISING (default) | FALLING

Enables the selection of rising/falling edge for RX.

MSPI_TX_EDGE

FALLING (default) | RISING

Enables the selection of rising/falling edge for TX.

MSPI_CPHA

FIRST_EDGE (default) | SECOND_EDGE

Enables the selection of the clock edge combined with MSPI_CPOL.

SSPI_RX_EDGE

RISING (default) | FALLING

Enables the selection of rising/falling edge for RX.

SSPI_TX_EDGE

FALLING (default) | RISING

Enables the selection of rising/falling edge for TX.

SSPI_CPHA

FIRST_EDGE (default) | SECOND_EDGE

Enables the selection of the clock edge combined with SSPI_CPOL.

SSPI_CPOL

IDLE_CLOCK_LOW (default) | IDLE_CLOCK_HIGH

SSPI_CPOL specifies if the clock is low or high in idle state with the following available options:

- IDLE_CLOCK_LOW (default) - Clock is low in idle state.
- IDLE_CLOCK_HIGH - Clock is high in idle state.

SSPI_SHIFT_ORDER

MSB_FIRST (default) | LSB_FIRST

The SSPI_SHIFT_ORDER bit specifies the SSPI shift order with the following available options:

- MSB_FIRST (default) - Shifts the most significant bit first.
- LSB_FIRST - Shifts the least significant bit first.

SSPI_DAISY_CHAIN_MODE

DISABLE (default) | SCM | AUTO

The SSPI_ DAISY_CHAIN_MODE allows you to enable the SSPI auto function for downstream daisy chain with the following available options:

- DISABLE (default) - Disables SSPI Auto/SCM mode.
- SCM - Persists CLK and MOSI.
- AUTO - SPI Auto mode based on SSPI_PORT selection for SERAIL, DUAL, QUAD or XSPI.

ERASE_EBR_ON_REFRESH

ENABLE (default) | DISABLE

The ERASE_EBR_ON_REFRESH allows you to enable or disable “erase” on refresh.

SIGNATURE_CHECK

ENABLE_LSCC_SIGNATURE (default) | DISABLE | ENABLE_SFDP_SIGNATURE

The SIGNATURE_CHECK preference allows you to check the signature with the following available options:

- ENABLE_LSCC_SIGNATURE (default) - Checks the LSCC signature.
- DISABLE - Skips signature checking.
- ENABLE_SFDP_SIGNATURE - Checks SFDP signature.

MSPI_ADDRESS_32BIT

DISABLE (default) | ENABLE

The MSPI_ ADDRESS_32BIT enables you to send the 32-bit command set on MSPI with the following available options:

- DISABLE (default) - 24-bit command set.
- ENABLE - 32-bit command set.

MSPI_COMMAND_32BIT

DISABLE (default) | ENABLE

The MSPI_ ADDRESS_32BIT enables the 32-bit MSPI addressing mode with the following available options:

- DISABLE (default) - 24-bit MSPI addressing mode.

- ENABLE - 32-bit MSPI addressing mode.

SSPI_IDLE_TIMER

DISABLE (default) | 200S | 100S | 50S | 25S | 10S | 5S | 1S | 750MS | 500MS | 250MS | 100MS | 75MS | 50MS | 25MS | 10MS

The SSPI_IDLE_TIMER allows you to specify the time to get a valid preamble.

MSPI_PREAMBLE_DETECTION_TIMER

200MS (default) | 100MS | 50MS | 40MS | 20MS | 1MS | 500US | 100US

The MSPI_PREAMBLE_DETECTION_TIMER allows you to specify the time to get a valid preamble.

MULTI_BOOT_SEL

STATIC (default) | DYNAMIC

The MULTI_BOOT_SEL preference allows you to select the multi-boot address. It has the following options:

- STATIC - Selects the multi-boot address from sysConfig MULTI_BOOT_OFFSET preference.
- DYNAMIC - Selects the multi-boot address from the MULTI_BOOT_OFFSET parameter setting of CONFIG_LMMIC primitive.

User Code Settings

To set User Code parameter values:

1. After opening the design project, choose **Tools > Device Constraint Editor**.
2. Select a the Global tab at the bottom of the view.
3. For each parameter in the value column, double-click to display the drop-down to select a value or enter the actual value.

The following table shows the default settings in bold type and the selectable settings for all of the keyword values for the User Code setting. They are ordered as they appear in the Global tab in the Device Constraint Editor.

Keyword	Default and Selectable Values	Supported Devices
UserCode Format	Binary (default) [Binary, Hex, ASCII, Auto]	LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP
	Binary (default) [Binary, Hex, ASCII]	LAV-AT
UserCode	**32 Binary Zeros** [value based on Format]	LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT
Unique ID (for Nexus Devices)	**0000** (default) 16-bit user defined code of above USERCODE space.	LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP

UserCode Format

Binary (default) | Hex | ASCII | Auto

The default is Binary but can be any of the above format types.

UserCode

value

The UserCode formatted into a 32 bit binary code in the bitstream but can be entered by the user based on the UserCode Format selection.

Unique ID (for Nexus Devices)

0000 (default in HEX)

Works in conjunction with UserCode. An automatic Usercode will be created where the upper 16-bits of the Usercode is created by the user to uniquely identify their design. The upper 16-bits is the Unique Id entered in HEX.

Note:

For Unique ID, **000** is the only allowed value for this release.

MultiBoot Offset Setting

The MULTI_BOOT_OFFSET preference allows you to set the multiboot address. The default MULTI_BOOT_OFFSET is 48 binary zeros.

The value is between 48'b000000000000000000000000000000000000000000000000 and 48'b111111111111111111111111111111111111111111111111.

To set MultiBoot offset value:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Enter Multiboot Offset value in the text box.

Device Support LAV-AT

Syntax

ldc_set_attribute {MULTI_BOOT_OFFSET=<value>}

Example The following command sets the multiboot offset value to 111000111.

```
ldc_set_attribute {MULTI_BOOT_OFFSET=111000111}
```

Bitstream Revision Settings

The section under 'Bitstream Revision' in Device Constraint Editor allows the user to set the revision number of the bitstream.

If Multiboot Bitstream Security Configuration is disabled and BBRAM is enabled, the revision number is compared against the minimum revision number stored in BBRAM. If Multiboot Bitstream Security Configuration is enabled and the bitstream is a multiboot bitstream, the provided revision number is compared against the minimum revision number stored in the Security Engine volatile memory.

If the user provides a revision number that is greater or equal the minimum revision number, the bitstream is processed. Otherwise, the bitstream is rejected. This provides protection against bitstream revision rollback attacks.

To set Bitstream Revision number:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. In the Bitstream Revision drop-down section, for each parameter in the Value column, double-click to display the drop-down to select or enter a value.

The following table shows the default settings in bold type and the selectable settings for all of the keyword values for the Bitstream Revision settings. They are ordered as they appear in the Global tab in the Device Constraint Editor.

Keyword	Default and Selectable Values	Supported Devices
BitstreamRevision Format	**Binary** (default) [Binary, Hex, ASCII, Timestamp]	LAV-AT
BitstreamRevision	**32 Binary Zeros** [value based on Format]	LAV-AT

BitstreamRevision Format

Binary (default) | Hex | ASCII | Timestamp

The default is Binary but can be any of the above format types.

Syntax

ldc_set_attribute {BITSTREAM_REV_FORMAT=<Binary/Hex/ACII/Timestamp>}

Example The following sets BitstreamRevision Format to Binary.

```
ldc_set_attribute BITSTREAM_REV_FORMAT=Binary
```

BitstreamRevision

value

BitstreamRevision is formatted into a 32-bit binary code in the bitstream but can likewise be entered by the user based on the selected BitstreamRevision Format.

Syntax

ldc_set_attribute {BITSTREAM_REV=<value>}

Example The following sets BitstreamRevision number to 00011.

```
ldc_set_attribute BITSTREAM_REV=00011
```

Derating Settings

The section under ‘Derating’ in the Device Constraint editor allows a user to define voltage deratings for core voltage, power supplies and bank voltage deratings.

To set Derating parameter values:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. In the Derating drop-down section, for each parameter in the value column, double-click to display the drop-down to select a value.

The following shows the default settings in bold type and the selectable settings for all of the keyword values for the Derating settings. They are ordered as they appear in the Global tab in the Device Constraint Editor.

Core=NOMINAL (default) | +/- 5% in increments of 1%

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

Syntax

ldc_set_vcc -core -derate <% value>

This defines the core voltage for derating. Radiant, will automatically generate a UserCode to uniquely identify their design. The upper bits will be defined with the Unique Id setting. The lower 16 bits will sequentially auto increment for every bitstream generation.

Nominal VCC core voltage value as specified in Supported Temperature and Voltage Range for Derating topic.

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 shows how to set the .sdc constraint which is stored in the .pdc file. Core voltage is set at a derating of -3%.

```
ldc_set_vcc -core -derate -3
```

DPHY0/1=value

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

This is used to derate the MIPIDPHY power supply. Valid values are from -5% to 5%, starting from a base 1.2V.

NOMINAL: 1.2V

PERCENT: 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.

Syntax

ldc_set_vcc -dphy <value> -derate <% value>

Example

The following constraint include sets MIPIDPHY voltage derating to -4%.

```
ldc_set_vcc -dphy DPHY0 -derate -4;
ldc_set_vcc -dphy DPHY1 -derate -4;
```

VCCIO=value

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

Defines the VCCIO bank voltage for derating. The value can be set for Bank0, 1 ,3, 5.

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

Syntax

ldc_set_vcc -bank <value> -derate <% value>

Examples

The following constraint sets VCCIO derating of Bank0 to -3% and Bank3 to -5%.

```
ldc_set_vcc -bank 0 -derate -3
ldc_set_vcc -bank 3 -derate -5
```

Bank VCCIO Settings

Sets the VCCIO voltage level of the I/O bank. In cases where design signals are not assigned to a device site or bank (floating), the BANK constraint can be applied to direct the I/O placer algorithm of Place and Route (par) to assign only design I/O signals with compatible signal standards to the bank.

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

To set Bank VCCIO parameter values:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. In the Bank VCCIO drop-down section, for each parameter in the value column, double-click to display the drop-down to select a value.

Bank0, 1 3, 5 can be set and the values range from: 1.0, 1.2, 1.35, 1.50, 1.80, 2.50, 3.30.

Syntax

ldc_set_vcc -bank <value> <voltage>

Example

The following shows how to set the Bank VCCIO for Bank3 to 3.3V and Bank5 to 2.5V.

```
ldc_set_vcc -bank 3 3.3
ldc_set_vcc -bank 5 2.5
```

Global Set/Reset Net Setting

Assigns the specified net as the input to the global set/reset (GSR) buffer inferred by Radiant. The GSR_NET constraint 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 database, the tool 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 constraint must drive at least one registered element's set/reset input, and the element must be enabled to be sensitive to GSR.

Device Support

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

To set Global Set/Reset Net:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Double-click the cell beside the ‘Global Set/Reset Net’ and a dialogue will appear.

4. Click on the “Available Logic Nets” on the left and click on the right arrow to move to “Selected Logic Nets”.

Syntax

ldc_set_attribute {GSR_NET=TRUE/FALSE} {get_nets <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:

```
ldc_set_attribute {GSR_NET=TRUE} [get_nets VCC_net]
ldc_set_attribute {GSR_NET=TRUE} [get_nets rst_c]
```

Use Primary Net Setting

Uses a primary clock resource to route the specified net. The specified net should be a clock net.

The USE PRIMARY physical constraint is 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

iCE40UP, LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

To set User Primary Net:

1. After opening the design project, choose **Tools >**  **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Double-click the cell beside the ‘Use Primary Net’ and a dialogue will appear.
4. For each available clock, click on the “Selection” drop-down box and select either Primary / Prohibit.

Syntax

ldc_set_attribute {USE_PRIMARY= TRUE/FALSE (USE_PRIMARY_REGION=<region_value>)} [get_nets <clock_net>]

where:

<clock_net> = specific clock net name

<region_value> is a region value of 0/1

Examples

The following command specifies that a primary clock resource will be used to route a clock net named **clk_c**:

```
ldc_set_attribute {USE_PRIMARY=TRUE} [get_nets clk_c]
```

The following command specifies that a primary clock resource will be used to route the clock net **clk_c** but with a primary region specified.

```
ldc_set_attribute {USE_PRIMARY=TRUE USE_PRIMARY_REGION=1}
[get_nets clk_c]
```

Prohibit Primary

Prohibits the use of primary clock resources for routing the net. The specified net should be a clock net. This constraint can be used with all devices that have primary clocking structures.

Syntax

ldc_set_attribute {USE_PRIMARY=TRUE/FALSE} [get_nets <clk_net>];

where:

<clk_net> = clock name string

Example

This command prohibits the use of a primary clock in routing a clock net named clk_c.

```
ldc_set_attribute {USE_PRIMARY=FALSE} [get_nets clk_c]
```

Vref Locate Setting

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

LIFCL, LFCPNX, LFD2NX, LFMXO5, UT24C, UT24CP, LAV-AT

To set User Vref Locate:

1. After opening the design project, choose **Tools >** **Device Constraint Editor**.
2. Select the Global tab at the bottom of the view.
3. Double-click the cell beside the 'Vref Locate' and a dialogue will appear.
4. For each available sites, click on the desired row and enter a unique name in the text box 'VREF Name".

Syntax

ldc_create_vref -name <vref_name> -site <site_value>

ldc_set_port -iobuf -vref <vref_name> {VREF=vref_name} [get_ports <port name>]

where:

<*vref_name*> = string

<*site_value*> = already pre-filled by Radiant.

For more details regarding I/O type, see the sysIO Usage Guide for your target device family.

Example

There are a total of six VREF IOs on the LIFCL device, each bank (3, 4, and 5) has two VREF IOs.

"T6" is VREF1 and "R11" is VREF2, both of them are in bank 4.

The Mapper always assigns "VREF1_LOAD" by default, if you use VREF2 IO as the VREF input, you need to manually add "VREF = VREF2_LOAD."

You also need to set these two constraints below:

```
ldc_create_vref -name VREF_BANK -site R11
ldc_set_port -iobuf -vref VREF_BANK {IO_TYPE=HSTL15_I
VREF=VREF2_LOAD} [get_ports clk_third]
```

Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor tool are used to enter all the timing constraints in the Radiant software tool, as well as to edit .ldc/.pdc and .sdc constraints. There are a total eight tabs to enter different constraint types:

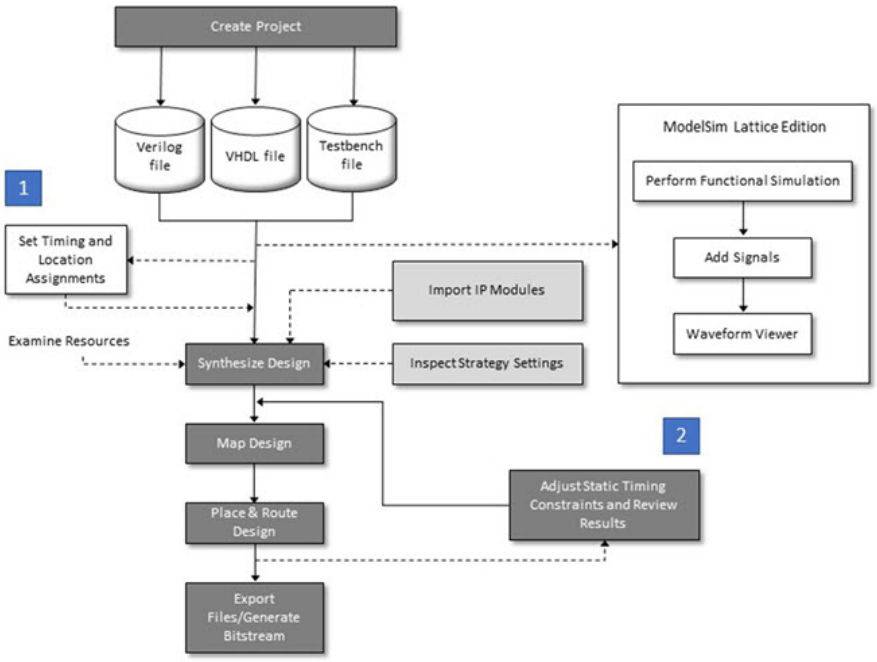
1. Clock - define the clocking scheme of the design.
2. Generated Clock - define generated clocks such as from PLLs.
3. Clock Latency - define the latency in terms of Rise, Fall times.
4. Clock Uncertainty - define From and To constraints.
5. Clock Group - used for complex clocking schemes and groups in terms of being Logically / Physically Exclusive and Asynchronous groups.
6. Load Capacitance - set the capacitive loading for I/Os.
7. Input/Output Delay - set Input and Output delays.
8. Timing Exception - set Min / Max, False and Multicycle path constraints.
9. Attribute - set synthesis attributes.

The above will be described in more detail below.

You will notice under **Tools**, there are there are two selections:

1. **Pre-Synthesis Constraint Editor**
2. **Post-Synthesis Timing Constraint Editor**

The two GUIs are identical and the entry mechanism of the constraints also the same. The key difference as noted in the tool name is that pre-synthesis are (.sdc) constraints are entered pre-synthesis and will be synthesized by the chosen synthesis tools. Post-Synthesis constraints (.pdc) will populate pre-synthesis timing constraints and allow for entering physical as well as post-synthesis timing constraints.The Post-Synthesis Timing Constraint Editor is the same GUI with the varying constraints already synthesized and populated in each of the different constraints tab. A user cannot modify the constraints in the post-synthesis constraints that were populated from pre-synthesis, but a user can supply new constraints to either one, override an existing, or supply totally new constraints to better constrain the design for better performance upon analysis.



Defining Synthesis Constraints Using Pre-Synthesis Constraint Editor

Pre-Synthesis Constraint Editor presents the contents of a Lattice Design Constraints File (.ldc) in a spreadsheet style format. You can use Pre-Synthesis Constraint 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.

Each sheet of Pre-Synthesis Constraint Editor allows you to define synthesis constraints by double-clicking a cell and selecting or typing a value.

Each row includes a Disable check box, which is unchecked by default. If you check this check box, the constraint is ignored by LSE. This can be an easy way to enable / disable the constraint.

See Also

- “Creating a New .sdc Synthesis Constraint File” on page 213
- “Defining Clocks Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 267
- “Setting Input and Output Delays Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 268
- “Defining Timing Exceptions Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 270
- “Defining Clock Groups in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 270

- “Defining Generated Clocks in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 271
- “Setting Attributes in Pre-Synthesis Constraint Editor” on page 273
- “Adding an .ldc File to an Implementation” on page 215
- “Synopsys Design Constraints” on page 750

Running Pre-Synthesis Constraint Editor

Launch the Pre-Synthesis Constraint 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 > Active Implementation > Select Synthesis Tools** and select **Lattice LSE**.

When a user disables a constraint in GUI of Pre-Synthesis Constraint Editor, you will see the #DISABLED# appended in front of actual constraint.

You can have multiple .ldc files for an implementation, but only one can be opened in Pre-Synthesis Constraint Editor at a time.

To run Pre-Synthesis Constraint Editor, do one of the following:

- Double-click an .ldc file name in the Pre-Synthesis Constraint 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 Pre-Synthesis Constraint Editor and click **OK**.
- If an .ldc file doesn’t yet exist for the active implementation, create an .ldc file as described in Creating a New .sdc Synthesis Constraint File.

See Also

- “Managing Constraint Files” on page 35
- “Using Source Editor for .ldc Files” on page 214

Defining Clocks Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Clock tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To define a clock using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. Double-click the **Object Clock** cell to select an existing clock pin, port, or net. A CLOCKPIN is applied on instances and (CLOCKPORT) ports is applied to top level modules. Clocks coming from the outside should ideally be defined on a port.
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 cell and enter the length of the high portion followed by low portion of the cycle in nanosecond. The default duty cycle is 50%. The waveform option is used to control the arrival times of the positive and negative edges and the duty cycle. The waveform edges have to be within the period of the clock. Enter the values with at least one decimal i.e. 5.0;2.0.

To define a clock in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor using Netlist Analyzer:

1. Start Netlist Analyzer and detach it by clicking the detach tool in the upper right corner. Refer to "About Netlist Analyzer" on page 51.
2. Select desired clock object (port or net) from Netlist Analyzer and drag-and-drop into Clock Object cell of the Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 cell and enter the length of the high portion followed by low portion of the cycle in nanosecond. The default duty cycle is 50%. The waveform option is used to control the arrival times of the positive and negative edges and the duty cycle. The waveform edges have to be within the period of the clock Enter the values with at least one decimal i.e. 5.0;2.0.

See Also

- "create_clock" on page 753
- "Creating a New .sdc Synthesis Constraint File" on page 213
- "About Netlist Analyzer" on page 51

Setting Input and Output Delays Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Inputs/Output Delay tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To set an input or output delay using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. Double-click the **Constraint 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 the Clock Fall check box if you want to clock on negative edge.
5. Select either Max or Min or neither. Do not select both. You can also select Add Delay.
   - 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.
   - Add Delay is used to define more than one input delay on a given port.
6. Enter a delay value in nanoseconds in the Value (ns) cell.

To set an input or output delay in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor using Netlist Analyzer:

1. Start Netlist Analyzer and detach it by clicking the detach tool in the upper right corner. Refer to “About Netlist Analyzer” on page 51.
2. Select from the input or output ports from Netlist Analyzer and drag-and-drop into the Port cell of the Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 the Clock Fall check box if you want to clock on negative edge.
5. Select either Max or Min or neither. Do not select both. You can also select Add Delay.
6. Enter a delay value in nanoseconds in the Value(ns) cell.

See Also

- “Defining Clocks Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor” on page 267
- “set_input_delay” on page 761
- “set_output_delay” on page 766
- “Creating a New .sdc Synthesis Constraint File” on page 213
- “About Netlist Analyzer” on page 51

Defining Timing Exceptions Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Timing Exception tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To define a timing exception using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. Double-click the **Constraint 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor using Netlist Analyzer:

1. Start Netlist Analyzer and detach it by clicking the detach tool in the upper right corner. Refer to “About Netlist Analyzer” on page 51.
2. Select object (port, net, or register) from Netlist Analyzer and drag-and-drop into the From or To cell of the Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor timing exception tab.
3. Specify the path information, delay, and cycles as appropriate for the selected delay type.

See Also

- “set_multicycle_path” on page 765
- “set_load” on page 762
- “set_min_delay” on page 764
- “set_false_path” on page 760
- “Creating a New .sdc Synthesis Constraint File” on page 213
- “About Netlist Analyzer” on page 51

Defining Clock Groups in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Clock Group tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor enables you to define clock groups that are logically / physically 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. Double-click the **Group** cell and select the appropriate clock group.
2. Now you will be able to select logically / physically or asynchronous type.

See Also

- "set_clock_groups" on page 757
- "Creating a New .sdc Synthesis Constraint File" on page 213

Defining Generated Clocks in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Generated Clock tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor enables you to define internally generated clocks. You can also perform the same functions by dragging-and-dropping from Netlist Analyzer into Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To define generated clocks using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor using Netlist Analyzer:

1. Start Netlist Analyzer and detach it by clicking the detach tool in the upper right corner. Refer to "About Netlist Analyzer" on page 51.
2. Select the source clock from Netlist Analyzer and drag-and-drop into the Source cell of the Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint 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 755
- "Creating a New .sdc Synthesis Constraint File" on page 213

Defining Clock Latency in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Clock Latency tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor enables you to define the behavior of the clock outside the FPGA. You can also perform the same functions by dragging-and-dropping from Netlist Analyzer into Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To define clock latency using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. In the **Clock** cell, double-click and select the three dots for the list of source clock for latency reference.
2. Specify the latency value.
3. Check off the check boxes for Rise, Fall, Early, Late and Source check boxes.

See Also

- “set_clock_latency” on page 758
- “Creating a New .sdc Synthesis Constraint File” on page 213

Defining Load Capacitance in Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

The Load Capacitance tab of Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor enables you to define load capacitance of ports in the design. You can also perform the same functions by dragging-and-dropping from Netlist Analyzer into Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor.

To define load capacitance using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor:

1. In the **Port** cell, double-click and select the three dots for the list of ports available for setting load capacitance.
2. Enter the value of the load.

See Also

- “set_load” on page 762
- “Creating a New .sdc Synthesis Constraint File” on page 213

Setting Attributes in Pre-Synthesis Constraint Editor

The Attribute tab of Pre-Synthesis Constraint Editor enables you to specify Synplify Lattice Attributes that are supported by the LSE. You can also define objects by dragging-and-dropping from Netlist Analyzer into Pre-Synthesis Constraint Editor.

To set attributes using Pre-Synthesis Constraint Editor:

1. Double-click the **Type** cell and select the desired attribute.
2. Specify object, value type, and value, as appropriate for the attribute.

To set attributes in Pre-Synthesis Constraint Editor using Netlist Analyzer:

1. Start Netlist Analyzer and detach it by clicking the detach tool in the upper right corner. Refer to "About Netlist Analyzer" on page 51.
2. In Pre-Synthesis Constraint Editor Attribute tab, double-click the **Type** 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 Pre-Synthesis Constraint Editor Attribute tab.

Note

You can also drag-and-drop an object from Netlist Analyzer to the Object cell of Pre-Synthesis Constraint 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 767
- "About Netlist Analyzer" on page 51

Compiling the Design Using Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor

By default, Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor is set to compile the design automatically. With automatic compilation, Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor automatically performs checks as to whether the design data is outdated. This happens whenever the Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor is launched or activated. If the design data is found to be outdated, the compilation will begin immediately.

Physical Designer

The Physical Designer is a combined GUI interface of both the Placement Mode and Routing Mode accessible within one Radiant software tool. This provides one central location where a user can do all the floor-planning and be able to view the physical layout of the design.

The tool can be invoked by either: **Tools > Physical Designer** or by clicking on the toolbar icon . Upon opening, the GUI you will be shown is the Placement Mode.

To access the **Routing Mode**, select the Placement/IO / view icon on the side tool and within this drop-down select the Routing icon to start routing view.

See Also

- "Physical Designer: Placement Mode" on page 276
- "Physical Designer: Routing Mode" on page 282

I/O Planning in IO Mode

Placement Mode is for GROUPS and REGIONs assignment. You can use the IO Mode tool for I/O planning and I/O assignment such as DDR interface, DQS and clock assignments. As you move your mouse pointer slowly over the I/O Abstract layout, details are displayed in tool tips: the number of resources for each items such as ECLCK, DDR, PLL, and IO Banks utilization are displayed.

Note

The Physical Designer IO mode is only supported in CrossLink-NX based device families.

Switching Views: Placement / IO Mode

To toggle between the Placement Model and IO Mode, click the from the drop-down menu.

IO Mode Operations

Design elements can be placed are shown in the left data panel and a chip-level view of the FPGA and available resources are shown on the right. The search text box at the top can be used to do searches for instances and IOs deep into the hierarchy.

Assignment is merely dragging elements from the left data panel onto the appropriate resource on the main layout I/O view window to be placed. Design elements (i.e. DDR) can be dragged as a group or individual element.

When clicking on the instance/IOs on the IO Abstract view main window, the associated instance/IO in the data pane on the left will be highlighted and vice versa.

 For regular I/O Banks - I/O already placed.

 For regular I/O Banks - I/O constrained and locked.

 For regular I/O Banks – Available for placement.

 For regular Serdes I/Os - I/O already placed.

 For regular Serdes I/Os– Available for placement.

Selection and assigning is performed by clicking and dragging of the I/O selected. Once you find the correct site, the site will be shown in green and the tool-tip will show the “+” to place the selected I/O object, you can release the left button. The tool does a final check and does the assignment accordingly.

Once an I/O object has been placed, you can move it again to a different place. If you select a leaf cell, on the cell or its associated “move-together” instances are selected.

You can also select the I/O object at the data panel. Right-click to activate the action menu and then choose Unplace.

Note that you cannot drag-n-drop for an I/O object which has been partially placed. You need to select low-level groups or leaf cells which have not been placed.

By default, a user cannot change user constraints in the IO view. If a component is constrained by a user “locate” constraint, the component is treated as a locked component. A locked component cannot be drag-n-dropped.

To drag-n-drop a locked component. From the Radiant main window, choose **Tools > Options > Tools** tab on the left and then choose **Physical Designer** heading tab**,** toggle the **Overwrite constraint:** true

Exporting I/O Planning Results

Once you complete your I/O planning whether pre/post P&R, you can export the I/O constraints to a PDF file. This is done by choosing **File > Export Placement Result**. You can overwrite an existing PDC or save to a new .PDC constraint file.

See Also

- “Displaying Assignments and Connections” on page 277
- “Assigning Signals” on page 284
- “Creating GROUPs” on page 289

Physical Designer: Placement Mode

Placement Mode provides a large-component layout of your design. All connections are displayed as fly-lines. Placement Mode allows you to create REGIONs and bounding boxes for GROUPS 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 GROUP and REGION; the number of utilized slices for each PLC component; and the name and location of each component, port, net, and site.

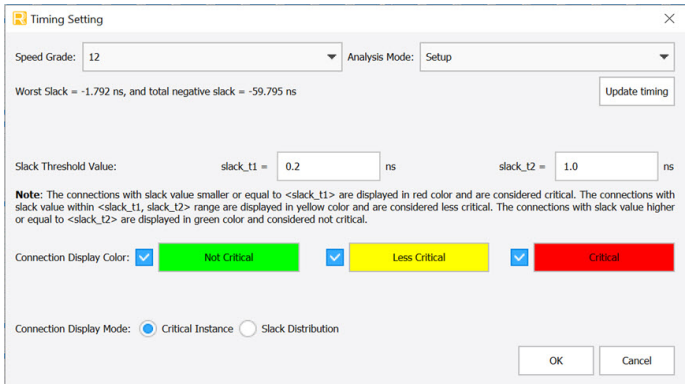
Placement Mode is available as soon as the target device has been specified.

See Also

- "Creating GROUPs" on page 289
- "Using the Zoom and Pan Tools in Device Constraint Editor" on page 238

Congestion Timing Mapping Display

The congestion timing display allows you to debug timing congested areas of your design. This congestion timing display gives the timing of the most critical paths within the slack threshold input. Clicking on the timing icon will bring up Time Setting components dialogue in which once set, will display the congestion map.

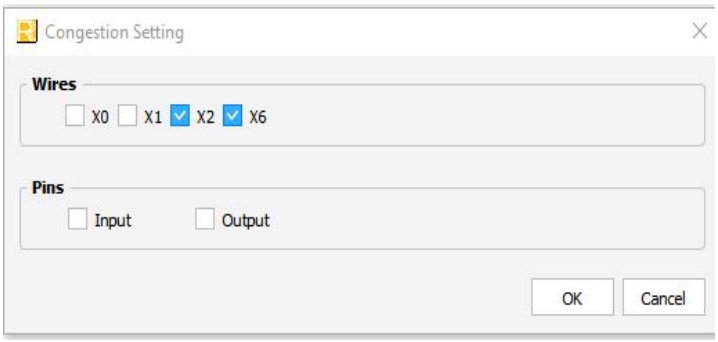


1. Speed Grade - Choose the speed grade to perform analysis on.
2. Analysis Mode - Choose Setup/Hold.
3. Enter Slack Threshold Values for t1 and t2. Any value less than t1 will be displayed as red, less than t2 as yellow, and the rest green.
4. Select the Connection Display Mode - Critical Instance or Slack Distribution. (The connection display colors can be changed.)
   - Critical Instance - Displays all red colored congestion lines connecting to the critical instances.
   - Slack Distribution - Shows the congestion lines in weighted equal distribution of each of the congestion levels.

5. Click Update Timing. A Worst/Total negative slack is calculated.
6. Click OK. A congestion map is drawn.

Congestion Display

The congestion display is a read-only view for viewing the most congested regions based on the wire length and pins selected. Clicking on the timing icon will bring up congestion display view after selecting options.



1. Wires - Select the wire connection from switch box. (x0, x1, x2, x6)
2. Pins - Select Input / Output.

Displaying Assignments and Connections

The Placement Mode toolbar allows you to select the types of elements that will be displayed on the layout, including IOs, DQS GROUPS, placed components, and component connections. These commands are also available from the View menu.

Area Selection Mode - used to drag and select sections of the floorplan.

Display Placement – displays all placed components, including assignments originating from the logical constraint file (.pdc) and those made by Place & Route.

Display Placement Groups – highlights the placed GROUP component sites with the designated color fill.

Display Connections To/From Selection(s) – 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 component, 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 Constraints - shows placement assignments that originated from the ..pdc file. When this button is active, placed components assigned with a LOC constraint are distinguished with a cross-hatch pattern.

 Display Regions – displays any defined REGIONS.

 Display Groups – displays any defined GROUPS.

 Display Logical Connections - shows the logical connections between GROUPs, including those assigned to REGIONs, and between GROUP and assigned Constraint sites. The connections are based on the logical instances within each GROUP.

 Show World View– toggles the World View of DCE to On or Off

 /  Placement/IO Abstract View - Toggles between these two views for GROUPS/REGIONS or I/O Placement.

 Congestion Timing Map View - Turns on timing congestion map tool for analysis.

 Congestion Display - Turns on congestion display analysis tool.

See Also

- “Using the Zoom and Pan Tools in Device Constraint Editor” on page 238
- “Viewing Assignments in Placement Mode” on page 279

Searching for Design Elements in Placement Mode

The Find Text... dialog box in Placement Mode enables you to search for components, nets, and sites.

To search for components, nets, or sites in Placement Mode:

1. Choose **Tools > Physical Designer**, 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 pulldown 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.

Viewing Assignments in Placement Mode

Placement Mode displays assignments at any stage of the design flow. These can include assigned sites, GROUPS, REGIONs, and reserved resources. After routing, all placements assigned by the Place & Route process are displayed.

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 Constraints,” or both are activated.

Assigned Sites After placement and routing, the site a component has been assigned appears on the layout with a solid fill.



If the assignment originated from the HDL source, the site will be filled and overlaid with a cross-hatch pattern when both “Display Placement” and “Display Placement Constraints” are activated.



When only “Display Placement Constraints” is activated, an assignment originating from the logical constraint file or HDL source is displayed with the cross-hatch pattern but without the fill.

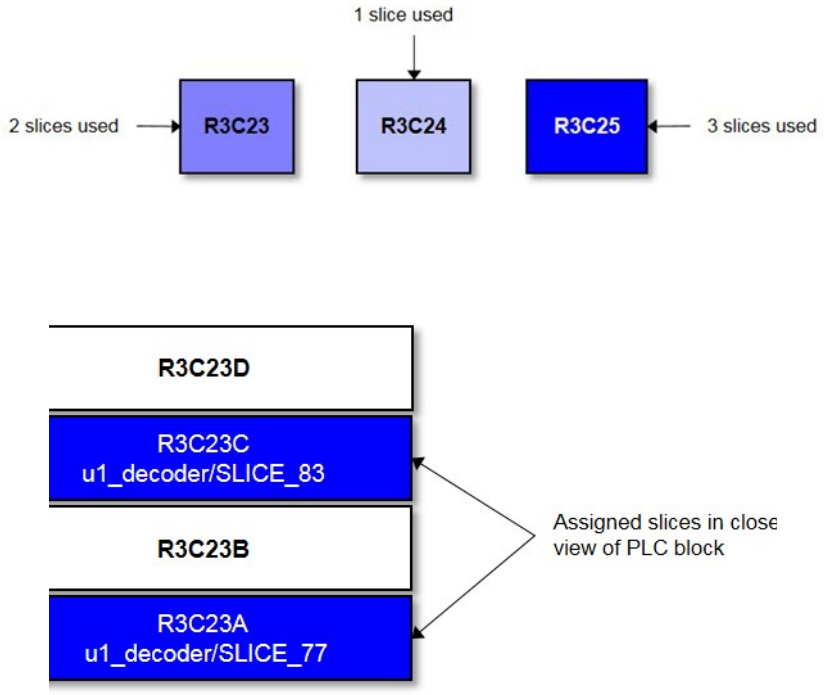


Reserved Sites Sites reserved are displayed with a gray pattern.

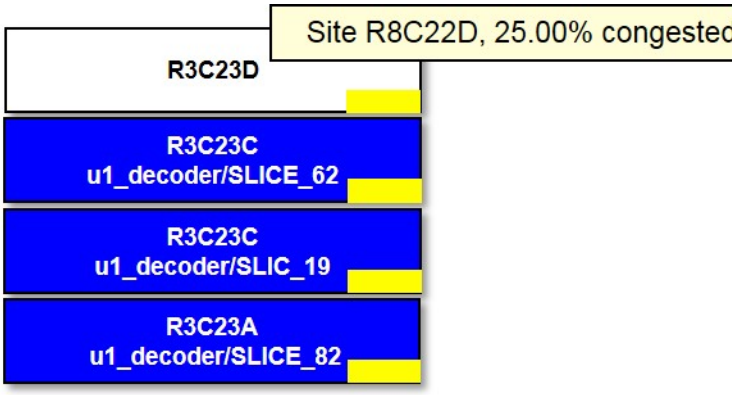


PLC Blocks (PFU and PFF) After placement and routing, a programmable logic cell (PLC) in which one or more logical components have been placed will appear on the layout with a solid fill. The color fill will be darker or lighter in shade, depending on how many sites are used.

As you zoom in so that individual slices are visible, only the slices that contain placed logical components are displayed with a color fill.



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.

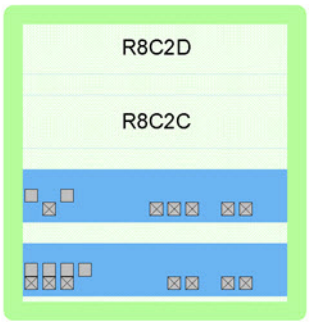


GROUPS

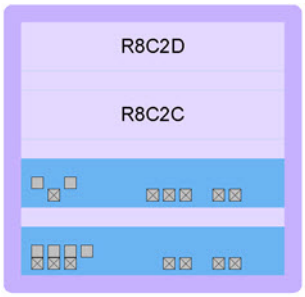
Anchored GROUPS are displayed when both the “Display Placement Constraints” and “Display GROUPS” commands are activated. An anchored

unbounded GROUP 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 GROUP is displayed with a solid border, corresponding to the assigned width and height, and a dotted pattern between blocks. GROUPS are distinguished from one another by their designated colors. After placement, the sites that contain the member elements of the GROUP are filled with the designated color when the “Display Placement Groups” command is activated.



REGIONs REGIONs are displayed when both the “Display Placement Constraints” 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 277
- “Assigning Signals” on page 284
- “Creating GROUPs” on page 289

Going to a Specific Site

Placement Mode 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 Placement Mode, 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**.

   Placement Mode zooms in to the location you specified.

See Also

- "Using the Zoom and Pan Tools in Device Constraint Editor" on page 238
- "Searching for Design Elements in Device Constraint Editor" on page 239

Physical Designer: Routing Mode

Routing Mode 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.

On the left hierarchy view in this tool, a user can do design exploration by the Logical tab whereby the netlist is hierarchy and signals exist at the module and instance level, or the Physical tab which displays a flattened netlist showing signals in the sub-module level.

Displaying Components and Connections

The routing mode toolbar allows you to select the types of elements that will be displayed on the layout.

 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 Analyzer.

Highlighting Elements

Use the View menu's Highlight / Unhighlight commands to highlight or unhighlight selected nets or sites.

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.

See Also

- "Using the Zoom and Pan Tools in Device Constraint Editor" on page 238

Applying Radiant Software Physical Constraints

Radiant software constraint-editing views allow you to specify FPGA pre-synthesis constraints without having to write out logical constraints syntax manually. These views include Spreadsheet View, Package View, Netlist View of Device Constraint Editor and Placement Mode.

Setting Global Constraints

The Global tab in Spreadsheet View enables you to set constraints that affect the entire design, such as voltage, temperature, global BLOCK constraints, and configuration constraints. The list of these constraints varies by device family.

To set global constraints:

1. In Spreadsheet View, select the **Global** tab.
2. To perform a global edit, 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 pulldown menu, choose the value from the list.
   - Right-click the constraint 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**.

Assigning Signals

Radiant software 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 Device Constraint Editor 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.

Assigning Pins Using the Port Assignments Sheet

For a single pin assignment, 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. From the Pin column, select the pin cells of the signals to assign.

   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 Device Constraint Editor View and detach it. Select the bus from the Ports list of Netlist View widget and drag it to the Port Assignments Sheet of Spreadsheet View in the lower half. 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 shows 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 pins at once and release the mouse button. Next, 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

5. From the Assign Pin pulldown menu at the bottom left, select one of the following Assign Pin options: 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**.

Modifying Assignments in Device Constraint Editor Spreadsheet View

You can modify pin assignments in Spreadsheet View before or after placement and routing. After routing, you can also modify assignments that automatically made 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 in Assigning Pins Using the Port Assignments Sheet.

To modify assignments made automatically by Place & Route:

1. Choose **View > Display IO Placement**.

   All 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 in Assigning Pins Using the Port Assignments Sheet.

   The assignments are now displayed in black font. Those that were assigned by Place & Route appear in parentheses, preceded by the new assignments.

To remove assignments:

- Select the assigned pins or signals that you want to delete, and 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.

Modifying Signal Assignments using the Device Constraint Editor

Device Constraint Editor 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 as well as swap the two assignments.

To modify one or more assignments:

1. Select the device pins on the layout that contains the assignments you want to modify.

   To select only one pin, click it to highlight it. Use the **Ctrl** or **Shift** key or to select more than one pin, or drag your mouse around multiple pins to select them.
2. 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 dragging it. If you try to drag a pin before it is highlighted, you will activate the multiple selection tool.

To remove assignments:

1. Select the pins that contain the assignments you want to remove.

2. Right-click the selected pins and choose **Unlock**.

See Also

- “Assigning Signals in Device Constraint Editor Spreadsheet View” on page 284
- “Modifying Assignments in Device Constraint Editor Spreadsheet View” on page 285
- “Prohibiting Incompatible Pins” on page 288

Modifying Signal Assignments in Placement Mode

In Placement Mode, you can modify a signal assignment by dragging it from one Constraint to another. You can also modify an assignment that originated from the logical constraint file (.pdc) or one that was made automatically by Place & Route (PAR).

To modify a signal assignment in Placement Mode:

1. Choose **Tools > Physical Designer**.
2. Make sure that the “Display Placement” button  is turned on if you will be modifying a PAR-assigned I/O.
3. In Placement mode, zoom in so that you can easily view the placed Constraint component that you want to reassign and the targeted location.

   Constraint components that were assigned automatically by PAR are displayed with a solid color fill. Constraint components whose assignments originated from the .pdc file are displayed with a color fill and a cross-hatch overlay.
4. 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.

Setting Port, Net, and Cell Attributes

Spreadsheet View's Port Assignments Sheet, Clock Resource sheet, Route Priority sheet, Cell Mapping sheet, and Global constraints 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

The Radiant software allows you to set values for all ports in your design at once. You can do this by using the ALLPORTS selection, which is available in the Spreadsheet view section of Device Constraint Editor.

Defining I/O Defaults Using ALL PORTS

The ALL PORTS row in Device Constraint Editor's 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 ALL PORTS:

1. In Spreadsheet View in the Device Constraint Editor, select the Port tab.
2. Double-click the desired attribute cell in the ALL PORTS row and select a value from the pulldown 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 ALL PORTS 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.

Prohibiting Incompatible Pins

Both Spreadsheet View and Package View of Device Constraint Editor enable you to create a PROHIBIT constraints 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 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**.

Creating GROUPs

The GRP constraint consists of logical components that are to be packed close together. GROUPs can include PFU, PFF, EBR, and DSP blocks. You can create GROUPs in Spreadsheet View, Netlist View and Placement Mode. After placement and routing, Placement Mode displays the group member elements with the GROUP color fill.

Note

A GROUP must be anchored if it includes a DSP block. If such a GROUP is not anchored, the Radiant software will issue a warning and ignore the DSP block.

Creating GROUPs in Netlist View in Device Constraint Editor

To create a GROUP in Spreadsheet View:

1. Right click on Netlist View on Ports (Input or Output).
2. Click on **Create Port Group**.
3. A new Group dialog appears. You can now move available ports to selected ports.
4. After entering a Group name, this constraint will be saved in the .pdc file as a ldc_create_group constraint.

Creating GROUPs in Placement Mode

In Placement Mode, you can draw a bounding box for a new GROUP, and then select the instances to be included. For routed designs, you can create a GROUP from selected components displayed on the layout.

To create a GROUP bounding box and add instances:

1. Choose **Tools > Physical Designer**.
2. Zoom in, as needed, to the desired area on the floorplan layout.
3. Right-click and choose **Create GROUP 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 GROUP. Make sure that the bounding box does not overlap the bounding box of another GROUP or REGION.

   When you release the mouse button, the Create GROUP 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. Under GROUP name, the resources covered by the bounding box are shown: LUTs, DSPs, registers, and EBRs.

   Note

   A GROUP must be anchored in order to be displayed in Placement Mode.
5. In the boxes at the top of the dialog box, specify a name for the GROUP and select a color.

6. From the list of Available Instances, select those that you want included in the GROUP.
   - Use a string and wildcards (asterisk or question mark) in the Filter text box to narrow the list.
   - Use the **Ctrl** or **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 **OK**.

   The new GROUP is displayed in Placement Mode with a bounding box in the color you selected. The new GROUP is added to the Group constraint sheet in Spreadsheet View.

To create a GROUP from selected placed components in Placement Mode:

1. In Placement Mode, zoom in to the desired area as needed.
2. Select the placed components that you want to include in the GROUP. 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 GROUP**.

   The Create New GROUP dialog box opens. In the Selected Instances list, the placed components from the blocks you selected in Placement Mode are shown.
4. In the boxes at the top of the dialog box, specify a name for the GROUP and select a color.
5. To select a different anchor location, select either **Region** or **Site**.
   - For Region, select a REGION from the pulldown 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.
   - For 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 GROUP'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 GROUP or REGION.

The group's revised resource coverage is now displayed at the top under GROUP Name, including the number of LUTs, registers, and EBRs.

7. Click **Add**.

   The new GROUP is displayed in Placement Mode with a bounding box in the color you selected, and it is added to the Group constraint sheet in Spreadsheet View.

Grouping Components Along an Individual Signal

In Placement Mode, you can group components along an individual signal of a timing path by cross-probing a signal from Timing Analyzer to Placement Mode. From Placement Mode, you can then select the components along the displayed connection to create a new GROUP.

To group components along an individual signal:

1. After placement and routing, choose **Tools >**  **Timing Analyzer**.
2. In the timing constraint list in the lower left pane, expand Analysis Results, and then select the desired timing constraint.

   The path table opens in the upper right pane.
3. Select a signal from the path table, right-click, and choose **Show in > Placement Mode**.

   The connection is highlighted in Placement Mode.
4. While holding the **Ctrl** key, click each component related to the net.
5. Right-click the selected components and choose **Create > GROUP**.

   The Create New GROUP dialog box displays the anchor location and bounding box size of the components you selected, and lists the instances in the Selected Instances pane.
6. In the top part of the dialog box, assign a name to the GROUP. Click the color box to assign a different color.
7. Click **Add**.

   The new GROUP is displayed in Placement Mode with a bounding box in the color you selected. The new GROUP is added to the Group constraint sheet in Spreadsheet View.

See Also

- "Using Timing Analyzer" on page 361

Modifying GROUPs

Use the Edit GROUP Property dialog box to quickly modify the name, color, and membership of a group constraint. You can access the dialog box from the Placement Mode. You can also use Placement Mode to delete a GROUP, or you can use the Group constraint sheet to delete a GROUP or GROUP elements.

To modify a group constraint:

1. To access the Edit GROUP 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 Placement Mode, zoom in to the GROUP and click the border or an empty space inside so that the group's border is highlighted. Right-click and choose **Edit GROUP**.
2. In the dialog box, make the desired changes and click **Update**.

To remove elements from a group on the Group constraint sheet:

1. Select the Group constraint sheet and expand the desired GROUP.
2. Select the group elements you want to remove, right-click, and choose **Remove GROUP Elements**.

To delete a group constraint on the Group sheet:

- Right-click the group you want to delete and choose **Delete**, or press **Delete**.

To delete a GROUP using Placement Mode:

1. Zoom in to the GROUP and click the border or an empty space inside so that the group's border is highlighted.
2. Right-click and choose **Remove GROUP**.

See Also

- "Creating GROUPs" on page 289

Resizing or Moving a GROUP in Placement Mode

You can resize a GROUP or change its anchor location in Placement Mode.

To resize a GROUP using Placement Mode:

1. In Placement Mode, zoom in to the GROUP and select it by clicking the border or an empty space inside. The border becomes highlighted, and you should see small circles on the GROUP's border—one in each corner and one on each side.
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.

   Placement Mode displays the new bounding box dimensions. The Group sheet in Spreadsheet View shows the updated dimensions.

To change the GROUP's anchor location:

1. Zoom out until you can view the current location on the layout and the targeted location.
2. Click the GROUP'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.

3. Hold down the left mouse button, drag the GROUP box to the targeted location, and then release the mouse button.

   Placement Mode displays the GROUP in its new location. The Group sheet in Spreadsheet View shows the updated location.

See Also

- "Creating GROUPs" on page 289

Constraint Conflict Resolution

When more than one constraint is ascribed to a single design element, some constraints 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 Lattice design constraints file (.ldc). The software resolves such conflicts by adhering to the following rules of precedence:

- Timing constraints that have a higher priority than others take precedence when the same design objects are constrained in two or more preferences.

  See "Timing Constraints Priority" on page 294

- When more than one constraint applies to a net or path, more specific preferences are honored before less specific ones. For example, individual net/path constraints supersede group (bus) constraints, and group constraints supersede global constraints.
- When more than one constraint at the same specificity level exists for a block/comp/net/path, the constraint that occurs last in the constraint file takes precedence.
- When an attribute in the HDL conflicts with a constraint in the Lattice design constraint file, the constraint in the .ldc file has precedence. For example, when a given register is included in both a GRP attribute and a logical constraint definition, the logical constraint is honored.

  See "Conflicting HDL Attributes and Constraints" on page 295

- Attributes in the HDL do not appear in the .ldc unless they have been modified and saved to the .ldc file. Afterwards, the groups in the .ldc file take precedence over those in the HDL. Conversely, logical groups created in the Radiant constraint views are kept in the .ldc file and are not back annotated to the HDL design file.
- When two or more GRPs are identified with the same name in the .ldc file, the group that appears last is honored.
- When different GRPs in the .ldc file contain one or more of the same members, the GRP that appears last is honored.

Timing Constraints Priority

The timer uses the following priorities for timing constraints. The highest priority is given to constraints that completely block a path. Under this category we have the set_false_path and the set_clock_groups constraints. Next in line are the timing exception commands. These are the set_max_delay, set_min_delay and set_multicycle_path constraints. These constraints are prioritized by their options from highest to lowest as follows:

1. From pin/port/cell, through pin/net/cell and to pin/port/cell
2. From pin/port/cell and to pin/port/cell
3. From pin/port/cell, through pin/net/cell and to clock
4. From pin/port/cell and to clock
5. From pin/port/cell and through pin/net/cell
6. From pin/port/cell
7. From clock, through pin/net/cell and to pin/port/cell
8. From clock and to pin/port/cell
9. Through pin/net/cell and to pin/port/cell
10. To pin/port/cell
11. From clock through pin/net/cell to clock
12. From clock to clock
13. From clock through pin/net/cell
14. From clock
15. Through pin/net/cell to clock
16. To clock
17. Through pin/net/cell

The set_input_delay and set_output_delay affect the arrival and required times at the design boundaries. The set_clock_latency and set_clock_uncertainty affect the arrival time and behavior of the edges of the clocks.

When multiple IO constraints with the same clock are used on an output port, the last constraint will take priority. Also, when the -add_delay is used, the

worst constraint will take precedence. the worst case timing for setup and hold are not the same.

Physical Constraints Priority

Conflict Between ldc_set_location of ldc_create_group and ldc_set_location

The following example contains a combination of group and location constraints that conflict:

Figure 24:

```
ldc_create_group -name Group_0 -bbox {9,9} [get_cells
{pdp_ram_0 pdp_ram_1}]
ldc_set_location -site {EBR_R36C6} [ldc_get_groups {Group_0}]
ldc_set_location -site {EBR_R36C22} [get_cells {pdp_ram_0}]
```

The Group_0 contains two RAMs of the design. One RAM (EBR) named pdp_ram_0, is placed at a specific EBR device site (EBR_R36C22) using the ldc_set_location constraint.

When the place and route program encounters this condition, it honors the ldc_set_location on the specific instance constraint. A warning will be issued to let the user know that the instance is discarded from the ldc_create_group constraint.

Conflicting HDL Attributes and Constraints

Using a combination of HDL attributes and constraints is a powerful technique for constraining the implementation tools. However, it is possible to introduce a combination of HDL attributes and logical constraints that are redundant and in conflict with each other. When such conflicts occur, the constraint in the .ldc file takes precedence.

Conflicting Pin Assignments:

When there is a conflicting pin assignment between the HDL and the constraint file, the assignment in the constraint file will take precedence. For example, SPI_PIN_D has a LOC attribute of W25 in the source file. Later, in the Device Constraint Editor Spreadsheet, the pin is assigned to site U21 and saved to the .ldc 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 ldc_set_location constraint from the .ldc 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 overwrites HDL:

ldc_set_location -site {U21} [get_cells {SPI_PIN_D}] ; (this is LDC, it overrides HDL))

The same is true for ldc_set_port -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.

Redundant GRP Declarations

In the following example, two group constraints with the same name contain different members:

Figure 25:

```
ldc_create_group -name Group_1 [get_cells {reg6 reg7}]
ldc_create_group -name Group_2 [get_cells {reg6 reg7}]
```

During place and route, an error will occur in the place and route report and terminate the process.

Conflict between ldc_create_group and ldc_set_location

The following example contains a combination of group and location preferences that conflict:

Figure 26:

```
ldc_create_group -name Group_0 -bbox {9,9} [get_cells
{pdp_ram_0 pdp_ram_1}]
ldc_set_location -site {EBR_R36C6} [ldc_get_groups {Group_0}]
ldc_set_location -site {EBR_R36C22} [get_cells {pdp_ram_0}]
```

The Group_0 contains two RAMs of the design. One RAM (EBR) named pdp_ram_0, is placed at a specific EBR device site (EBR_R36C22) using the ldc_set_location constraint.

When the place and route program encounters this condition, it honors the ldc_set_location of the ldc_create group constraint. No warnings or errors are generated.

Conflicting I/O Buffer Attributes for the Same Port

When two ldc_set_port constraints appear in the .pdc file for the same port, the one that occurs later will be honored and will override the one that appears earlier. The one exception is IO_TYPE, which can be maintained unless the constraint that occurs later specifies a different IO_TYPE.

For example, the following constraints for port A appear in two different places in the .pdc file:

Figure 27:

```
ldc_set_port -iobuf {SLEWRATE=SLOW PULLMODE=DOWN} [get_ports
{A}]
ldc_set_port -iobuf {SLEWRATE =FAST} [get_ports {A}]
```

When the design is mapped, the SLEWRATE=FAST constraint, which appears later, will be honored. The earlier constraint will be completely overridden, which means that PULLMODE will revert to the default setting.

See Also

- "Timing and Physical Constraints" on page 201

Checking Design Rules

The Radiant software provides DRC, which enables you to validate I/O placement, logic, and routing.

During load time and initialization, the Radiant software constraints views interpret the database and constraint files in a similar manner as the Map and Place & Route processes. Syntax or semantic violations within the .ldc and .pdc files are also treated in a manner similar to the Map and Place & Route processes. Illegal constraints are ignored but retained in the logical and physical constraint files. Radiant loads and applies the legal subset it detects and displays error and warning messages.

After the project is loaded, Radiant provides access to on-demand and real-time Constraint DRC Check. All DRC reports are presented in the Output window for the current session.

See Also

- "Constraint DRC" on page 297

Constraint DRC

Constraint DRC can catch many common errors and prevent wrong pin usage at the early stage of the design process, and 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.

Constraint 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.

Constraint DRC displays all of its messages in the Radiant software Output window.

See Also

- "Performing On-Demand Constraint Design Rule Check" on page 298

Performing On-Demand Constraint Design Rule Check

The Constraint DRC feature validates pin assignments to banks based on legal combinations of signal standards, including true and emulated LVDS combinations. The Radiant software performs immediate real-time Constraint DRC on any constraint changes. But you can run Constraint DRC on the entire design after modifying constraints, by running on-demand Constraint DRC.

To perform Constraint design rule checking:

1. From the Radiant software Tools menu, select **Device Constraint Editor.** The following views are available:
   - Spreadsheet View
   - Package View
2. Select **Constraint DRC**  from the Device Constraint Editor toolbar.

   Radiant 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.

See Also

- "Constraint DRC" on page 297
- "Modifying Assignments in Device Constraint Editor Spreadsheet View" on page 285

Post-Synthesis DRC Check

DRC runs automatically throughout the LSE synthesis flow. After the synthesis process, a report is generated providing information on any combinational loops or clocks that have been found connected to black box modules.

Table 23: Supported Design Rules for LSE

Design Rule	Support
Combinational loops ▶ Excludes inferred nets.	Yes
Clocks connected to a black box ▶ Excludes Lattice primitive black boxes. ▶ Excludes macro black boxes.	Yes

Figure 28: Example of LSE DRC Report

```
######################### DRC Report #########################
-------------------------------------------------------------
Total number of design rule violations: 13
-------------------------------------------------------------
-------------------------------------------------------------
Rule Message ID | Violation Count
-------------------------------------------------------------
   <35103000>   | 1
-------------------------------------------------------------
   <35103100>   | 1
-------------------------------------------------------------
   <35103101>   | 3
-------------------------------------------------------------
   <35103301>   | 2
-------------------------------------------------------------
   <35103302>   | 1
-------------------------------------------------------------
   <35103303>   | 1
-------------------------------------------------------------
   <35103400>   | 2
-------------------------------------------------------------
   <35103401>   | 2
-------------------------------------------------------------
#############################################################

ERROR <35108000> - Found 13 DRC violations.
Please fix the design before continuing to synthesis.
Alternatively, ignore the violated rules with -drc_off or disable DRC.
```

If any DRC violations are found, an error message will appear at the end of the report.

LSE Design Rule Check

LSE has an independent design rule verification process to lint user RTL. The synthesis flow checks user RTL designs for any design rule violations.

- The flow is enabled by directly supplying the argument -drc to synthesis. For example: **synthesis -f impl.synproj -drc**.
- Any design rule can be disabled by message ID via command line argument or .synproj line: **-drc_off <msg_id>**.

  - Multiple message IDs, separated by spaces, can be added to one -drc_off setting.
  - All design rules are enabled by default.
- Certain design rules can be configured by message ID via command line argument or .synproj line: **-drc_cfg <msg_id> <value>**
- When a design rule violation is detected, LSE creates a warning.
  - You can promote this warning to an error via existing methods in Tcl or GUI (using the message ID).
- A DRC report will be provided at the end of the flow, and all output will be written to the log file: **synthesis.drc**.

Note:

radiant and **radiantc** flows will automatically change the log filename to: <prjname>_<impl>_lattice.drc.

Table 24: LSE DRC Rules

Design Rule	Message ID	Configuration
Invalid attribute ▶ Checks all nets ▶ Validates attribute name (not value)	35103000	N/A
Inferred latch	35103100	N/A
Combinational loop ▶ Excludes inferred nets	35103101	N/A
Register with high fanout	35103300	Fanout limit - values less than or equal to 0 will imply the default value of 20.
Register uses both async set and async reset	35103301	N/A
Same signal source drives clock port and another port of a register	35103302	N/A
Same signal source drives sync and async ports of same register	35103303	N/A

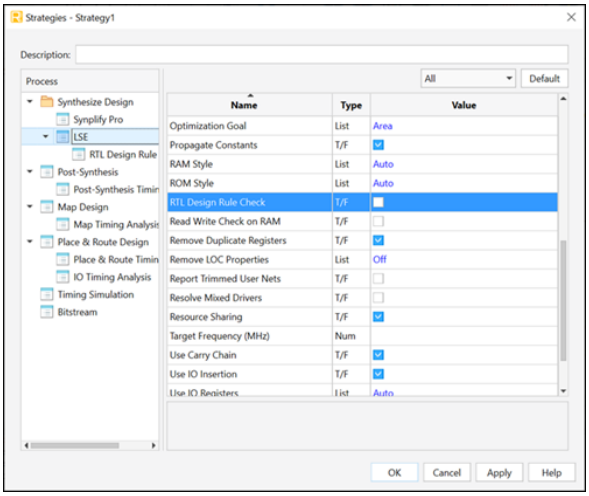
Table 24: (Continued)LSE DRC Rules

Design Rule	Message ID	Configuration
Clock connected to a blackbox ▶ Excludes Lattice primitive blackboxes ▶ Excludes macro blackboxes	35103400	N/A
Clock with logic on its net	35103401	N/A

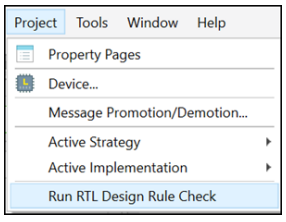
Using RTL Design Rule Check

1. In the File List view, double-click the active strategy, for example **Strategy1**. The Strategies dialog box opens.

   Under LSE and Synplify Pro strategies, you can enable the **RTL Design Rule Check Option**. In Synplify flow, Radiant inserts **project -run design_rule_check** in the Tcl script.

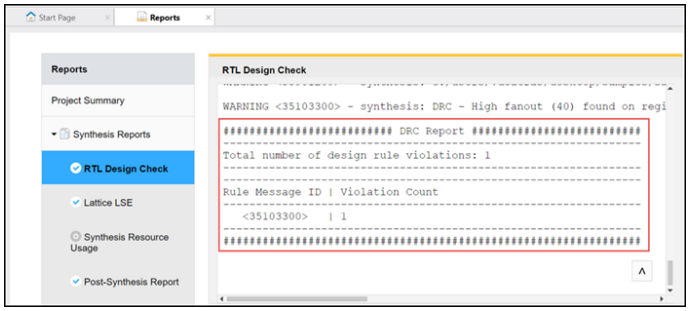


2. In the Menu, click **Project > Run RTL Design Rule Check**.



   - LSE lint runs if you are using LSE implementation.
   - Synplify lint runs if you are using Synplify implementation.

3. In the Reports tab, you can view the **RTL Design Check** section under Synthesis Reports.



Post-Map DRC Check

In the post-map stage, DRC performs the following logical and physical tests:

- Block
- Net
- Pad
- Clock Buffer
- Name
- Primitive Pin

Reported errors from DRC indicate conditions where routing or component logic does not operate correctly. Warnings indicate conditions where routing or logic is incomplete or the condition is incorrect but not serious. Some conditions listed as warnings might be reported as errors when DRC is performed by the Generate Bitstream Data (bitgen) process.

For Nexus devices, if the following conditions are met, JTAG pins are not available for GPIO, and DRC will be issued if any GPIO is locked into the JTAG pins:

1. **SLAVE_I2C_PORT**, **SLAVE_I3C_PORT**, **JTAG_PORT**, **SLAVE_SPI_PORT**, and **MASTER_SPI_PORT** sysCONFIG attributes are all set to **DISABLE**.
2. None of the **OSC, WDT**, **JTAG**, **MULTIBOOT**, **SEDC**, and **CONFIG_LMMI** primitives are used in the design.

Post-Map DRC Check is available in the post-map or post-PAR stage in the Spreadsheet View, Package View, Device View in Device Constraint Editor and Placement Mode.

See Also

- “Performing On-Demand Constraint Design Rule Check” on page 298

Analyzing SSO

For many FPGA devices, Radiant 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 Device Constraint Editor View to stagger package pin assignments.
- Use slow slew outputs.
- Use lower drive strengths

Note

Note when there are multiple SSO constraints defined with ldc_set_port, the latest constraint SSO value will always overwrite a previously defined value.

See Also

- “Running SSO Analysis” on page 304
- “Viewing SSO Analysis Results” on page 306

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 Device Constraint Editor 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 constraints, 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 physical design constraint file (.pdc).

To run SSO analysis:

1. In a generic sense, you can edit SSO values in:
   - The Device Constraint Editor View as indicated from Step 4 onward.
   - In the "Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor" on page 265 you can set values in the Load Capacitance tab. Note that change in SSO values will be saved in the physical design constraint file. (.pdc)
2. In Device Constraint Editor View, select the Port Assignments tab.
3. Assign pin locations or just the bank locations to the output or bidirectional signals that you wish to analyze.
4. 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
5. Specify settings in the following SSO columns for each signal to be analyzed:
   - Switching ID – 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 All Port row to quickly set the values for all outputs.
6. Once you save entered values from either the Device Constraint Editor (DCE) tool or the Pre-Synthesis Constraint Editor, the constraints are written and saved in the physical design constraint (.pdc) file.
7. In the Device Constraint Editor toolbar, click the Run and Show SSO View button  to produce SSO analysis and report.
8. In the Process bar, re-run Map Design and re-open DCE to see new SSO analysis results.

To view the full I/O SSO Analysis Report, select the Reports tab > Misc Reports > I/O SSO Analysis.

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 Map Design.

See Also

- “Viewing SSO Analysis Results” on page 306
- “Analyzing SSO” on page 303

Viewing SSO Analysis Results

To view results, you can open the I/O SSO Analysis Report in the Reports tab or view it directly in the Device Constraints Editor > SSO tab.

To view the SSO Analysis Report:

- In the Report tab, expand the Misc 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 Device Constraint Editor View:

1. In Device Constraint Editor View, choose **SSO tab** at the bottom of the window.

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 303
- "Running SSO Analysis" on page 304

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 304

Exporting Pin Files

The Design Constraint Editor 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 constraints. The following options are provided for exporting the contents of constraint 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.

Pinout File

A pinout file is a comprehensive report, in CSV format, of all available and unbonded pins in the device.

See Also

- "Exporting a Pin Layout File" on page 308

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.

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. The incompatible selection in the tab will affect the export result.

3. Select a value separator that will be used to separate the data into columns: comma, semicolon, space, or tab.

4. 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.

5. 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**.

6. 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.

Note

For exported pin layout files, the values of Package_Delay and Trace_Length are presented in picoseconds (ps) and micrometer/micron (µm), respectively.

See Also

- “Exporting a Pinout File in CSV Format” on page 309
- “Migrating Pin Assignments” on page 207

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 Device Constraint Editor 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 Pin Out File” dialog box, specify the file name and location that you want, and then click Save.
3. 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 308

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.
- Establishing physical connections that will join the components in an electrical network.

The Radiant software environment enables you to set options for each of these processes to help optimize results, including placement effort level. 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 697.

See Also

- “Using Strategies” on page 43
- “Synthesizing the Design” on page 311
- “Mapping” on page 320
- “Place and Route” on page 327
- “Applying Design Constraints” on page 196
- “Command Line Reference Guide” on page 1471

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, Radiant software can be used with almost any synthesis tool. The Radiant software 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 the Radiant software. If you prefer, you can use other synthesis tools by running them independently of the Radiant software.

Synthesis tools can be run three ways with the Radiant software:

- Integrated synthesis is the simplest way. You work entirely within the Radiant software. See “Integrated Synthesis” on page 317.
- Interactive synthesis provides much greater control of how synthesis is done. You set up a design project in the Radiant software, but set up and run synthesis directly in the synthesis tool. See “Interactive Synthesis” on page 318.
- Stand-alone synthesis allows use of other, non-supported synthesis tools. You set up and run synthesis directly in the synthesis tool and then the Radiant software imports the result. See “Stand-Alone Synthesis” on page 319.

LSE can only be run in integrated synthesis.

Whichever method you use, see “Pre-Synthesis Check List” on page 312 to make sure your project is ready to synthesize.

See Also

- “Selecting a Synthesis Tool” on page 70
- TN1008, “HDL Synthesis Guidelines for Lattice Semiconductor FPGAs” for how coding style influences performance and area utilization
- Synopsys Synplify Pro for Lattice User Guide
- Synopsys 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, add or adjust the constraints, attributes, and directives used by the synthesis tool in the source code. See the synthesis tool’s user documentation. For LSE, see “Lattice Synthesis Engine Constraints” on page 750.

Integrated synthesis:

- Specify a synthesis tool. See “Selecting a Synthesis Tool” on page 70.

- Specify the top-level unit. See "Setting the Top-Level Unit for Your Project" on page 38. 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 72.
- Specify a VHDL library name. The default is "work." See "Specifying VHDL Library Name" on page 71.
- 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 43. If using LSE, also see "Optimizing LSE for Area and Timing" on page 314.

Interactive and stand-alone synthesis:

- Reference the Lattice synthesis header library in the source code. See "Lattice Synthesis Header Libraries" on page 313.
- For interactive synthesis, if you want to use the Synplify Pro strategy settings instead of Synplify Pro's own defaults, change the "Export Radiant Settings to Synplify Pro GUI" option to **Yes** or **Only on First Launch**. See "Export Radiant Software Settings to Synplify Pro GUI" on page 705.

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 Synplify Pro. 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 the Radiant software with interactive or stand-alone synthesis, you need to manually add a reference to the appropriate library. Look in Table 25 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 is Verilog-based, add the .v file.

The files are located at:

<*install_dir*>/cae_library/synthesis/verilog
<*install_dir*>/cae_library/synthesis/vhdl.

Table 25: Synthesis Header Files

Device Family	Header Library File	Library Name
iCE40UP	iCE40UP.v iCE40UP.vhd	ice40UP
LAV-AT-E	lav-ate.v lav-ate.vhd	lav_ate
LAV-AT-G	lav-atg.v lav-atg.vhd	lav_atg
LAV-AT-X	lav-atx.v lav-atx.vhd	lav_atx
LFCPNX	lfcpnx.v lfcpnx.vhd	lfcpnx
LFD2NX	lfd2nx.v lfd2nx.vhd	lfd2nx
LFMXO5	lfmxo5.v lfmxo5.vhd	lfmxo5
LIFCL	lifcl.v lifcl.vhd	lifcl
UT24C	ut24c.v ut24c.vhd	ut24c
UT24CP	ut24cp.v ut24cp.vhd	ut24cp

Optimizing LSE for Area and Timing

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 by choosing **Project > Active Strategy > LSE Settings**. In the Strategy dialog box, set the following options, which are found in **Synthesize Design > LSE**.

The [Optimization Goal] strategy can be used as a global high level guidance to guide LSE.

The individual strategies in the table allow lower level control for specific optimization.

For example, you may set [Optimization Goal] to either Area or Timing, and leave the rest to default by LSE.

Or, for example, you can set the global [Optimization Goal] to Area, then continue to control lower level setting (more granular) such as FSM encoding to one-hot encoding, although one-hot encoding is generally recommended for Timing.

See the following text for explanations and more details.

Table 26: LSE Strategy Settings for Area and Speed

Option	Area	Speed
FSM Encoding Style	Binary or Gray	One-Hot
Max Fanout Limit	<*maximum*>	<*minimum*>
Remove Duplicate Registers	True	False
Resource Sharing	True	False
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 797. 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 789. 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.

Remove Duplicate Registers

Removing duplicate registers reduces area, but keeping duplicate registers may reduce delays.

Resource Sharing

If set to True (box checked), 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 (MHz)

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 813. Set Use IO Registers to meet your most demanding requirement. Then add syn_useioff to help other requirements.

See Also

- “Using Strategies” on page 43
- “LSE Options” on page 709

Integrated Synthesis

In integrated synthesis, you create, synthesize, and implement a design completely within the Radiant software environment. This is the simplest method, but it limits your control of synthesis to tools and features directly supported by the Radiant software.

To synthesize your design in the Radiant software:

If you haven't already, check the Pre-Synthesis Check List before running synthesis.

1. If desired, change the strategy settings by choosing **Project > Active Strategy >** <*Synthesis Tool*> **Settings**. See also "Using Strategies" on page 43.

   Note

   If you are switching from interactive synthesis to integrated, you need to reset all the options in the Radiant software. Options set directly in the synthesis tool have no effect in the Radiant software.

2. In the Process Toolbar, click **Synthesize Design**. Alternatively, right-click **Synthesize Design** and choose **Run Synthesis** from the pop-up menu.

   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 **Force Run Synthesis** from the pop-up menu. Otherwise from each of the process bar items, you can force the whole flow to start from synthesis by selecting **Force Run from Start.**

3. When finished, check the icon next to Synthesize Design in the Process frame. A green check mark indicates success; 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 Project Summary, choose **Synthesis 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 85.

   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 *.tws. It is similar to the report described in "Reading Timing Analysis Reports" on page 352.

See Also

- "Optimizing LSE for Area and Timing" on page 314
- "Running Processes" on page 63

Interactive Synthesis

In interactive synthesis, you set up a design project in the Radiant software, 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 that LSE cannot be used in interactive synthesis.

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, the Radiant software will synthesize the design with integrated synthesis, which may give unexpected and inferior results.

To synthesize your design using interactive synthesis:

If you haven't already, check the "Pre-Synthesis Check List" on page 312 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. To re-use the settings, save the synthesis tool's project file as <*project_name*>**_syn.prj** (Synplify) in the project folder.

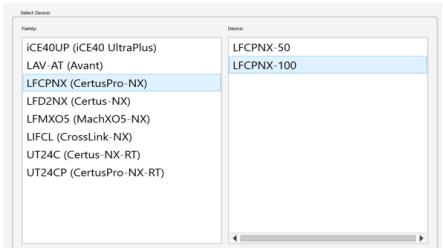
   Make sure the Radiant software strategy option, "Export Radiant Settings to Synplify Pro GUI" (under Synplify Pro in the Strategies dialog box) is set to **No** or **Only on First Launch**. See "Export Radiant Software Settings to Synplify Pro GUI" on page 705.

4. Synthesize the design.

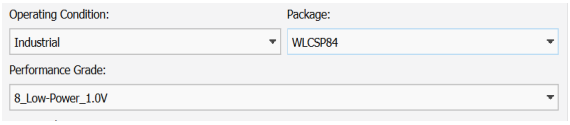
5. Analyze the results in the synthesis tool.

Note:

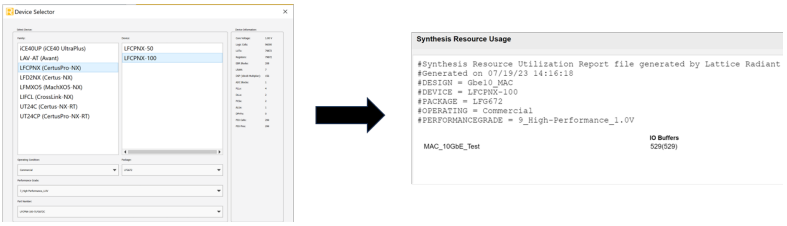
In the device selector, changing the Family or Device requires re-running the synthesis.



While changing the Package, Operating Condition, or Performance grade, does not require to re-run the synthesis. This rule applies to all devices.



After synthesis, resource usage reports can be triggered to compare selected device usage as the figure shown below.



For Avant devices, the synthesis process requires additional time to rerun to ensure a precise report.

Stand-Alone Synthesis

In stand-alone synthesis, you can use a synthesis tool that is not directly supported by the Radiant software. In this flow you set up and run synthesis directly in the synthesis tool. Then create a design project in the Radiant software using the output of the synthesis tool as the source file. Note that LSE cannot be used in stand-alone synthesis.

If you haven't already, check the "Pre-Synthesis Check List" on page 312 before running synthesis.

To set up a project using a stand-alone tool:

1. Open the synthesis tool directly without using the Radiant software.
2. Using your synthesis tool, create a project, including specifying a device, source files, and options.
3. Synthesize the design.
4. Analyze the results in the synthesis tool.
5. When you are satisfied with the synthesis, create a design project in the Radiant software.

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 the Radiant software.
2. In the synthesis tool, open the project.
3. If desired, change the options.
4. Analyze the results in the synthesis tool.

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 Radiant software environment or the MAP program from the command line.

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.

See Also

- “Running MAP from the Command Line” on page 1497
- “Setting Map Design Options” on page 321
- “Mapping Output Files” on page 321

Setting Map Design Options

When you run the Map Design process in the Radiant software, 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, or allow you to over map 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 pulldown menu, enter text, or click the browse button, as appropriate.
3. Click **OK**.

See Also

- “Map Design Options” on page 719
- “Mapping” on page 320
- “Mapping Output Files” on page 321

Mapping Output Files

The following files are created from the Map Design process:

- 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 320
- “Setting Map Design Options” on page 321

Running Map Design

In the Radiant software environment, the Map Design process automatically maps the design based on the active strategy settings.

To run Map Design in the Radiant software environment:

- In the Process Toolbar, click **Map Design**.

See Also

- “Running MAP from the Command Line” on page 1497
- “Running Processes” on page 63

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 Radiant software environment or the command line.

To view the MAP Report file in the Radiant software:

- Click the **Reports** tab to activate the Report view and select **Map Reports.** 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 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. The MAP Report lists the number of slice registers, I/O registers, LUT4s, and other logic. The number of SLICEs generated from MAP could exceed the device limitation. The Place and Route (PAR) process will determine whether or

not it is necessary to merge two SLICEs into one.

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.

```
Design Summary
--------------
   Number of slice registers: 2140 out of  5280 (41%)
   Number of I/O registers:     0 out of    56 (0%)
   Number of LUT4s:           3833 out of  5280 (73%)
      Number of logic LUT4s:             3026
      Number of inserted feedthru LUT4s:  39
      Number of ripple logic:            384 (768 LUT4s)
   Number of IO sites used:   22 out of 56 (39%)
      Number of IO sites used for general PIOs: 22
      Number of IO sites used for I3Cs: 0 out of 2 (0%)
      Number of IO sites used for PIOs+I3Cs: 22 out of 53 (42%)
      (note: If I3C is not used, its site can be used as
general PIO)
     Number of IO sites used for OD+RGB IO buffers: 0 out of 3
(0%)
   Number of DSPs:             4 out of 8 (50%)
   Number of I2Cs:             0 out of 2 (0%)
   Number of High Speed OSCs:  0 out of 1 (0%)
   Number of Low Speed OSCs:   0 out of 1 (0%)
   Number of RGB PWM:          0 out of 1 (0%)
   Number of RGB Drivers:      0 out of 1 (0%)
   Number of SCL FILTERs:      0 out of 2 (0%)
   Number of SRAMs:            0 out of 4 (0%)
   Number of WARMBOOTs:        0 out of 1 (0%)
   Number of SPIs:             0 out of 2 (0%)
   Number of EBRs:             0 out of 30 (0%)
   Number of PLLs:             0 out of 1 (0%)
   Number of Clocks:  1
      Net clk_c: 2022 loads, 2022 rising, 0 falling (Driver:
Port clk)
   Number of Clock Enables:  8
      Net n18836: 38 loads, 38 SLICEs
      Net n9306: 128 loads, 128 SLICEs
      Net n18680: 36 loads, 36 SLICEs
      Net n18773: 36 loads, 36 SLICEs
      Net n18139: 36 loads, 36 SLICEs
      Net n9301: 128 loads, 128 SLICEs
      Net n9296: 128 loads, 128 SLICEs
      Net n9311: 128 loads, 128 SLICEs
   Number of LSRs:  6
      Net maxfan_replicated_net_203: 169 loads, 169 SLICEs
      Net data_in_reg_63__N_501: 925 loads, 925 SLICEs
      Net mult32_2/n1818: 4 loads, 4 SLICEs
      Net n9303: 4 loads, 4 SLICEs
      Net n9298: 4 loads, 4 SLICEs
      Net mult32_3/n1821: 4 loads, 4 SLICEs
   Top 10 highest fanout non-clock nets:
      Net reset_n_c: 926 loads
      Net data_in_reg_63__N_501: 925 loads
```

The Number of SLICEs information in the Design Summary will vary.

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.
  - 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.
  - 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.
- 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 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 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 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.

- ASIC Components – Lists any ASIC instances in the design by name and specifies the type used for each. These components mostly include IPs, such as Block RAM, DSP, and others.
- 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 Radiant software 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 Radiant software environment.
- 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.
- Constraint Summary – Lists the total number of constraints and the number of constraints dropped.
  - Reports the number of constraints before the wildcard expansion.
  - Constraints with syntax or semantic errors could be dropped by the constraint database before the wildcard expansion.
    - These erroneous constraints are reported by the constraint database but not counted as they are dropped.
- Run Time and Memory Usage – Lists total CPU time, real time, and peak memory usage related to the Map run.

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.

Place and Route

After a design has undergone the necessary translation to bring it into the physical design 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 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.

In the Radiant software Task Detail View, there is a Place & Route Timing Analysis process available that runs static timing analysis. This process reports any timing errors and generates a report. The Place & Route Timing Analysis report can be accessed from the Place & Route Reports folder of the Reports window.

For the Avant device, running Place & Route Timing Analysis is required. You cannot skip this process. The timing score reported by PAR only provides an overall indication of the timing quality and not the precise timing at each corner, which is why Avant requires multi-corner timing analysis for timing sign-off.

For other devices, running Place & Route Timing Analysis is optional.

See Also

- "Setting Place & Route Design Options" on page 327
- "Options and Processes to Improve PAR Results" on page 328
- "Place & Route Output Files" on page 328
- "Running Place & Route Design" on page 329
- "PAR Report File" on page 330

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 affect the way the results are reported. You can set Place & Route Design options in the Strategies dialog box in the Radiant software or use the par command-line program.

To set Place & Route Design options in the Radiant software 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 pull down menu, enter text, or click the browse button, as appropriate.
4. Click **OK**.

See Also

- “Place & Route Design Options” on page 722
- “Running PAR from the Command Line” on page 1499
- “PAR Report File” on page 330

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.
- Timing period/frequency objectives are not met.
- Try increasing the number of times the placer runs.

See Also

- “Running Place & Route Design” on page 329
- “Running PAR from the Command Line” on page 1499
- “Place & Route Design Options” on page 722
- “Place & Route Considerations” on page 336
- “Place & Route Output Files” on page 328

Place & Route Output Files

The following files are the output files generated by the Place & Route Design process:

- 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.

See Also

- “Place and Route” on page 327

Running Place & Route Design

In the Radiant software environment, the Place & Route Design process automatically assigns device-specific components to locations and connects them.

To run Place & Route Design in the Radiant software environment:

- In the Process Toolbar, click **Place & Route Design**.
- Alternatively, right-click **Place & Route Design** and choose **Run PAR** from the pop-up menu. You can force run PAR from the pop-up menu as well.

See Also

- “Setting Place & Route Design Options” on page 327
- “Running PAR from the Command Line” on page 1499
- “Running Processes” on page 63

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 par file and .pad file.

If you run multiple placement and routing iterations, you produce a .par and .pad file for each iteration. After each iteration, the program saves the physical design for every instance where the timing core has improved. This means that during long PAR runs, the current 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 a .par 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.par 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 "Place & Route Report Files" on page 330. 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 327

Place & Route Report Files

The Reports view in the Radiant software 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.

When you run multiple iterations of Place & Route, a .par file and .pad file, are included in your project directory for each run. You can open these files with a text editor.

See Also

- "Multiple PAR Iterations" on page 329
- "PAR Report File" on page 330

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 view the PAR Report file in the Radiant software:

- In the Design Summary pane of the Reports window, select **Place & Route Reports** and select one of the three reports:
  - Place & Route
  - Signal/Pad
  - Place & Route Timing Analysis
  - I/O Timing Analysis

  The chosen report will appear in the right pane.
- Alternatively, use a text editor to open the .par ASCII file from your project directory.

By default, the Cost Table Summary is sorted by worst slack for multiple placement seeds, as follows:

1. Worst slack (setup)
2. Timing score (setup)
3. Worst slack (hold)
4. Timing score (hold)
5. Seed #

Device Utilization Summary

This section summarizes the number and percentage utilization of device resources, including I/O, logic, and global signals. 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 of them, or about 38 percent. The following line shows that 194 bonded out pads are being used out of a possible 372 bonded I/Os, or 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:
Number of Vref pins used: <#VREF pins>
```

Best Par Run

This section shows the Radiant software version and the P&R commands with any non-default options of the run. In addition, the section lists the design name and device information, including family name, device name, package name and performance grade.

Example:

```
PAR: Place And Route Radiant Software (64-bit) 2.0.t.311.0.
Command Line: par -w -t 1 -cores 1 -exp parPathBased=OFF
top_impl1_map.udb top_impl1_par.dir/5_1.udb

Design:  dphy
Family:  LIFCL
Device:  LIFCL-40
Package: CSBGA289
Performance Grade:   7_High-Performance_1.0V
```

NBR Summary

This section summarizes the post-routing results, including the number of unrouted connections, number of timing violated connections, worst setup slack and setup timing score for the design.

NBR Summary Example:

Post-Route Summary Example (Positive Setup Slack):

```
---------------------------------------------------------------------------------------

   PAR_SUMMARY::Run status = Completed
   PAR_SUMMARY::Number of unrouted conns = 0
   PAR_SUMMARY::Estimated worst slack<setup/<ns>> = 28.700
   PAR_SUMMARY::Timing score<setup/<ns>> = 0.000
   PAR_SUMMARY::Estimated worst slack<hold/<ns>> = 0.126
   PAR_SUMMARY::Timing score<hold/<ns>> = 0.000
   PAR_SUMMARY::Number of errors = 0
```

```
   Note: You must run "timing" to get the timed closure signoff
   The design with positive setup slack is also computed for
   the hold slack.
```

Post-Route Summary Example (Negative Setup Slack):

When setup slack is negative, the router will not execute hold analysis since the design did not meet the constrained frequency and timing.

```
--------------------------------------------------------------------------------

   PAR_SUMMARY::Run status = Completed
   PAR_SUMMARY::Number of unrouted conns = 0
   PAR_SUMMARY::Estimated worst slack<setup/<ns>> = -1.514
   PAR_SUMMARY::Timing score<setup/<ns>> = 12.218
   PAR_SUMMARY::Estimated worst slack<hold/<ns>> = <n/a>
   PAR_SUMMARY::Timing score<hold/<ns>> = <n/a>
   PAR_SUMMARY::Number of errors = 0

Note: You must run "timing" to get the timed closure signoff.
Hold slack is not computed for the negative setup slack design.
```

I/O Timing Analysis

The I/O timing report identifies the FPGA. Before running I/O timing, the FPGA must be fully constrained.

The tables in the I/O timing report contain the total number of setup and hold times associated with input ports and the clock-to-out delays linked to the output ports. The I/O timing report uses these values when computing the input and output delay constraints of devices connected to the FPGA.

I/O timing runs across all speed grades of interest. When the I/O timing tables are generated, the table displays the worst number for all speed grades.

See Also

- "Place & Route Output Files" on page 328

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 Radiant software environment:

- In the Design Summary pane of the Reports window, select **Signal/Pad**. Your report will appear in the pane to the right.

Figure 29: Example of I/O Timing Analysis Report

```
Worst FPGA Input Port Results for Performance Grade: 2, 3
---------------------------------------------------------------
Port Name| Setup |Grade|Edge| Hold |Grade|edge|
Clock Port
---------------------------------------------------------------
i[3]     |2.566 ns |  3  |  R |-2.088 ns |  2  |  R |i_my_pll/
lscc_pll_inst/
gen_ext_outclkdiv.gen_no_clk7.u_pll.PLLC_MODE_inst/
CLKOP(internal clock)
---------------------------------------------------------------
Worst FPGA Output Port Results for Performance Grade: 2, 3
---------------------------------------------------------------
Port Name|Clock To Out (MAX)|Grade|Edge|Clock To Out
(MIN)|Grade|edge|
Clock Port
---------------------------------------------------------------
c[6]     |         8.937 ns |  2  |  R |         7.511 ns |  3
|  R |i_my_pll/lscc_pll_inst/
gen_ext_outclkdiv.gen_no_clk7.u_pll.PLLC_MODE_inst/
CLKOP(internal clock)
---------------------------------------------------------------

Detailed Analysis of Each Performance Grade:
---------------------------------------------------------------

FPGA Input Port Results for Performance Grade: 2
---------------------------------------------------------------
Port Name| Setup |Edge|  Hold  |edge|
Clock Port
---------------------------------------------------------------
i[3]     |2.493 ns |  R |-2.088 ns |  R |i_my_pll/
lscc_pll_inst/
gen_ext_outclkdiv.gen_no_clk7.u_pll.PLLC_MODE_inst/
CLKOP(internal clock)
---------------------------------------------------------------
FPGA Output Port Results for Performance Grade: 2
---------------------------------------------------------------
Port Name|Clock To Out (MAX)|Edge|Clock To Out (MIN)|edge|
Clock Port
---------------------------------------------------------------
c[6]     |         8.937 ns |  R |         7.943 ns |  R
|i_my_pll/lscc_pll_inst/
gen_ext_outclkdiv.gen_no_clk7.u_pll.PLLC_MODE_inst/
CLKOP(internal clock)
---------------------------------------------------------------
```

- Alternatively, use a text editor to open the .pad ASCII file from your project directory.

The .pad file may include 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 design will not include both true and complement sides of differential port signals. The designer assigns I/Os to pads (via LOCATE) using a single signal, assuming that it will represent the true (+) signal. Based on an IOBUF type of LVDS or LVPECL, the Radiant software 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.

The Properties column of the Pinout by Port Name section shows the list of IO properties for each device. See the following table for details:

Table 27: Pinout by Port Name IO Properties

Device	Properties
LAV-AT	DRIVE, SLEWRATE, GLITCHFILTER, HYSTERSIS, TERMINATION, OPENDRAIN, IO_TYPE, DIFFRESISTOR, DIFFDRIVE, PULLMODE, VREF
LFCPNX, LFD2NX, LFMXO5, LIFCL	DRIVE, SLEWRATE, GLITCHFILTER, HYSTERSIS, TERMINATION, OPENDRAIN, IO_TYPE, DIFFRESISTOR, DIFFDRIVE, PULLMODE, CLAMP, VREF
iCE40UP	DRIVE, IO_TYPE, PULLMODE

- 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 the primary pin number, the component name or reference voltage type, and the buffer type. 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 have an internal pull-up.

Place & Route Considerations

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.
- By default, PAR supports high fanout CE and LSR nets to use for the primary clock tree to reduce potential congestion on general routing, which may help fMAX in certain cases.

  However, because the primary clock has dedicated entries that are only located at the edges and the center of the device, the driver of the high fanout CE/LSR net has to be placed at one of these dedicated sites, separating it from its related logics (module).

  These high fanout CE nets are on a critical route in some designs, which may result in inconsistent critical path placement.

  With the `ldc_set_attribute USE_PRIMARY=FALSE [ get_nets {u1/PCLK_c} ]` constraint, the high fanout CE net will not be able to use the primary clock tree. It should be acceptable to restrict the use of high fanout CE nets, provided that the design is properly routed and fMAX is unaffected.

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 Radiant software 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.

The 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 one 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 Windows or Linux, you must first create an ASCII-based node list file input that specifies the candidate machines that 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* | *windows*>

Corenum = <*number_of_cores*>

Env = <*file_name*>

Workdir = <*directory_name*>

[<*node_name2*>]

System = < *linux* | *windows*>

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 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 Windows, there will only be one node description block in the node list since you can only employ one windows 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 Windows as a value.

Corenum specifies how many CPU cores there are on the machine. This line can be omitted when there is only one CPU 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.

For more details, see the subsection “PAR Multi-Tasking (-m) Option” on page 1502 of the topic “Running PAR from the Command Line” on page 1499.

General Node List Rules:

- You must use the equal sign for each node description line and the same line and character spacing illustrated in the node list format example.
- 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*>] and System lines are required for any node list file.
- The Corenum, Env, and Workdir node description lines are optional.
- For Windows, 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 Windows network will be discarded by the PAR multi-tasking feature. If you do not have a nodelist file created, you can specify the number of cores to be used in your local machine with a -cores option in your command line, or use the “Number of Host Machine Cores” strategy in the Strategies dialog box.
- For UNIX /Linux, the nodelist file can have multiple machine nodes; but it should not include Windows type nodes, which will be discarded. If you do not have a nodelist file created, you can specify the number of cores to be used in your local machine with a -cores option in your command line, or use the “Number of Host Machine Cores” strategy in the Strategies dialog box. For cases when the user specifies both -cores and -m with a valid node list file, PAR should apply both settings (merge). If the user repeats the host machine in the node list file, the settings in the node list file take precedence over the setting in -cores.
- By beginning a line with double forward slashes, “//”, you can place a comment line anywhere in the node list file. The PAR multi-tasking feature will ignore its syntax.

PAR Multi-Tasking Environment Setup

For Windows, 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 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.
  - 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 1476. 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.
  - After it reads the node list, the PAR Multi-tasking option will know which script to run and in which working directory to run the par command. The spawned task on the remote machine will run the “Env” file, then it will “cd” to the working directory before it runs PAR.

There is also Linux environment setup information for this feature in the “PAR Multi-Tasking (-m) Option” on page 1502 section of the topic “Running PAR from the Command Line” on page 1499.

Running Multiple PAR Jobs in Parallel in the Radiant software

When running the multi-tasking option on Windows, 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 Windows with multiple processors:

1. Prepare a node list ASCII text file that identifies your machine (lpass4), system type (Windows), 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 Linux, or Windows values, depending upon your platform. However, the Windows value cannot be used with Linux because it is not possible to create a multiple computer farm with Windows. 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 Windows.

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 the Radiant software, click **Project > Active Strategy > Place and Route Design 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 elsewhere, 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, set the placement iteration starting point to 3, and limit the number of best saved runs to 2.

7. In the Process view in the Radiant software 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 the one process over again and nothing else. Rerun All will rerun the design flow up to the Place and Route process.

   In the Output view or Automake Log, the multipar run shows 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.
```

   Note that we chose the starting point of 3, so the jobs start at 5_3 instead of 5_1.

   When the job run is successfully completed, PAR creates a <project_name>.dir directory in your top-level project directory.

To learn more about how to use this feature from the command line, see the subsection “PAR Multi-Tasking (-m) Option” on page 1502 of the topic “Running PAR from the Command Line” on page 1499. 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 1499
- “PAR Multi-Tasking (-m) Option” on page 1502

Bit Generation

The **Bitstream File** process in the Process view or bit generation (**bitgen**) program takes a fully routed physical design 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 Radiant software window by double-clicking the **Export Files** process or from the command line.

Note

To confirm that your production bitstream does not include debuggers, you can check the bitstream report under Export Report. Refer to "Viewing Logs and Reports" on page 85.

See Also

- "Generating Bitstream Files" on page 343
- "Bit Generation Considerations" on page 344
- "Bit Generation Output Files" on page 342
- "Bit Generation Options" on page 342
- "Running Bit Generation from the Command Line" on page 1511

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.

Bit Generation Output Files

The following files are possible output to the **Bitstream File** process or the **bitgen** program:

- **Bit File** (Binary) — binary (.bin) bitstream. Binary bitstream files are the default output of the bitstream process and contain the configuration information in bitstream (zeros and ones) that is represented in the physical design.
- **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 .bin file in that it contains design information in the first six lines.

See Also

▶ “Bit Generation” on page 341

Generating Bitstream Files

In the Radiant software, bitstream generation is automatically performed when you run the **Export Files** process for either a full or partial design flow. You can set bit generation options before running a design flow by setting a strategy.

To generate bitstream files from the Radiant software:

1. In the Radiant software 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 pulldown list, and click **Apply**.
4. When you finish, click **OK** to close the Strategies dialog box.
5. In the Radiant software Process view, double-click **Bitstream File** to generate the bitstream files.

See Also

▶ “Bit Generation” on page 341
▶ “Bit Generation Output Files” on page 342
▶ “Bit Generation Considerations” on page 344
▶ “Running Bit Generation from the Command Line” on page 1511

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 cable connector.

See Also

▶ “Running Bit Generation from the Command Line” on page 1511

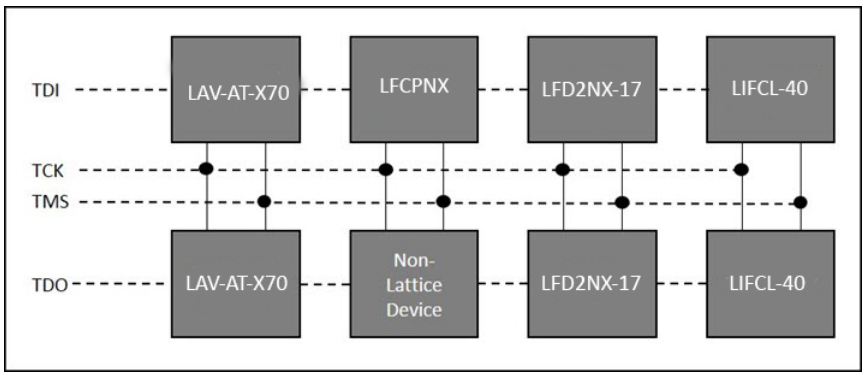
JTAG Scan Chains

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 in Figure 30.

Figure 30: Diagram of a JTAG Scan Chain



Bit Generation Considerations

Note the following 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 keyword values.

See Also

- “Bit Generation Options” on page 342

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 error or multi-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:

- LFD2NX
- LFCPNX
- LIFCL
- LIFCL-33
- LIFCL-33U
- LFMXO5
- UT24C
- UT24CP

SEI Editor is available after you have placed and routed your design and generated a bitstream.

See Also

- "Running SEI Editor" on page 345
- "Adding and Configuring a New Error with SEI Editor" on page 348
- "Removing an Error with SEI Editor" on page 348
- "Inserting an Error into a Device Using SEI Editor" on page 349

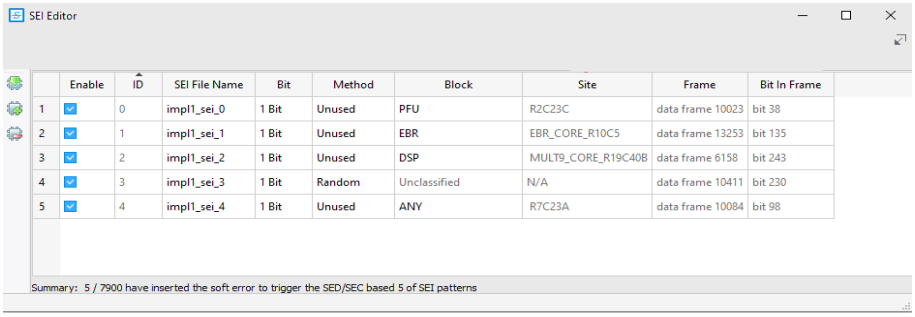
Running SEI Editor

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.

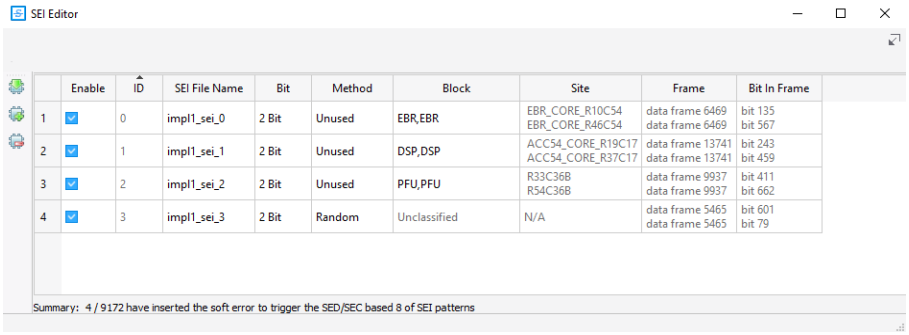
SEI Editor has the following columns:

- **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.
- **Bit** — This feature allows you to choose between a single-bit error or multi-bit error injection to add new errors to your design:
  - 1-bit
  - 2-bit

Note

In the same frame, 2-bit only supports the following:

1. PFU to PFU
2. EBR to EBR
3. DSP to DSP
4. DSP to PFU
5. EBR to PFU



- **Method** — This option determines whether the error is inserted into an unused or random block.
  - Unused — inserts the error into an unused block, which do not affect the design logic. When this method is chosen, you need block type input from PFU, DSP, and ANY only. "ANY" means bitgen will randomly choose the block type among PFU, DSP, and Routing Bit. Then bitgen reads UDB, randomly picks one unused site of such type, and flips one random bit in it. The BFD equation of such random bit has nothing to do with any used sites or any routing mux, i.e., this bit will not impact any used sites in the user logic. The bit flipped in the case of a 2-bit error will be from the same data frame.

Bitgen displays unused site name and random bit location within the frame, including frame row and bit number, and reports flipped frames or bits.

- 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.

  Bitgen displays the random bit location and category, including block type, device location, frame row number, and frame bit number, excluding padding bits in the front.

Note:

There will be no site location when a routing bit or unclassified is encountered.

- **Block**

  If Method option is set to "Unused", this option specifies the block where the error was inserted, in one of the following categories:

  - PFU
  - DSP
  - EBR
  - ANY (any of PFU or DSP)

  You can change the destination Block by double-clicking it and selecting a new value from the drop-down menu.

  If Method option is set to "Random", after you run in SEI Editor, the Block will automatically display either Routing or Unclassified, which indicate errors in Routing or any blocks such as PFU, EBR, DSP or Routing.

- **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** — Displays the address of the SEI frame.

  This will generate a data frame with flipped random bits, producing partial bitstream output in binary.bit or ASCII.rbt file formats.

  The file name is the same as the bitstream output name, but with .bgn extension and no site will be created when a routing bit or unclassified is encountered.

  Currently, we specify the following categories:

  - PFU
  - EBR
  - DSP
  - Routing or Unclassified (if Method is Random)

  For more information about Soft Error Injection, refer to FPGA-TN-02076-2.1 Soft Error Detection (SED)/Correction (SEC) User Guide for Nexus Platform.

See Also

- “Analyzing your Design using Soft Error Injection” on page 344
- “Adding and Configuring a New Error with SEI Editor” on page 348
- “Removing an Error with SEI Editor” on page 348
- “Inserting an Error into a Device Using SEI Editor” on page 349

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 344
- “Running SEI Editor” on page 345
- “Removing an Error with SEI Editor” on page 348
- “Inserting an Error into a Device Using SEI Editor” on page 349

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 344
- “Running SEI Editor” on page 345
- “Adding and Configuring a New Error with SEI Editor” on page 348

- "Inserting an Error into a Device Using SEI Editor" on page 349

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 Radiant 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 Radiant Programmer. Refer to "Programming the FPGA" on page 418.

3. In Radiant Programmer, program the partial bitstream into the device using the **Static Random Access Memory (SRAM) > Background Programming > SEI Fast Program** Programmer settings**.**

See Also

- "Analyzing your Design using Soft Error Injection" on page 344
- "Running SEI Editor" on page 345
- "Adding and Configuring a New Error with SEI Editor" on page 348
- "Removing an Error with SEI Editor" on page 348

Analyzing Static Timing

Static timing analysis can determine if your circuit design meets timing constraints. Rather than simulation, it employs conservative modeling of gate and interconnect delays that reflect specific operating conditions with a specific FPGA.

You can produce timing analysis reports as part of the synthesize, map, and place-and-route processes. Before running a process, click the Task Detail View in the Process Toolbar and select Timing Analysis for that process.

The reports have similar information shown in the same format. But they are based on information from each process:

- Post-synthesis timing analysis is based on pre-synthesis constraints and estimates of delays.
- Map timing analysis is based on post-synthesis constraints, the actual types of components, and estimates of the routing delays.
- Place-and-route timing analysis is based on post-synthesis constraints, and the actual components and routing.
- Place-and-route I/O timing analysis reports the setup and hold slack for each port that has a timing constraint.

All the reports can be read in the Reports tab.

An alternative to generating the Place & Route Timing Analysis report is the Timing Analyzer tool. Instead of a text report, Timing Analyzer gives you a spreadsheet view that you might find easier to read. Timing Analyzer also has a search function to help you find different data paths.

Options for Timing Analysis Reports

There are several options that can be set for each timing analysis. For the post-synthesis, map, and place-and-route reports, you can specify whether you want hold or setup analysis, the device performance grade, and the number of paths and endpoints in the report.

You can also change the style of the report with the Report Format option. The default is the current Lattice Standard, which follows industry standards for timing analysis tools. But if you have been using the Lattice Diamond tool set, you might prefer the older Diamond Style. One of the key differences is that Lattice Standard provides more information in its detailed description of data paths.

I/O timing analysis has an option to analyze the ports for all the speed grades of the FPGA.

Options are set in the implementation strategies. For more information, see:

- “Specifying Strategy Options” on page 46
- “Post-Synthesis Timing Analysis Options” on page 718
- “Map Timing Analysis Options” on page 721
- “Place & Route Timing Analysis Options” on page 725
- “IO Timing Analysis Options” on page 727

Running Post-Synthesis Constraint Scenario Timing Reports

You can run different Map and Place & Route Timing Analysis timing reports without needing to re-run the synthesis through the Place & Route Design process. The Radiant software will use the data within its own database to generate real time timing reports. In the Post-Synthesis Constraint File folder, only one .pdc file can be active among any other available .pdc file.

To run the real time timing reports on post-synthesis .pdc files:

1. Right-click on one of the inactive .pdc files.
2. Choose **Run > Map Timing Analysis** or **Run > Place & Route Timing Analysis.**

   A real-time report is generated and displayed in the Reports view of the main Radiant window.

Note

In order to run the timing report scenario for Map or Place & Route, the design flow from synthesis through to Place & Route must be run at least once on the active .pdc file.

Reading Timing Analysis Reports

These sections will vary depending on the FPGA architecture, timing constraints, design elements, and reporting style.

There are major sections that are detailed in the table of contents:

1. Timing Overview
2. Setup at Speed Grade 1 Corner at 85 Degrees (up to 125 degrees).
3. Setup at Speed Grade 1 Corner at Zero Degrees (-40 degrees).
4. Hold at Speed Grade M Corner at Zero Degrees (-40 degrees).

Each line item in the table of contents is also a hyperlink that will take you directly to that section of the report.

Timing Overview

Shows the constraints and operating conditions that guided the analysis. It also shows a list of combinational loops that could not be analyzed.

SDC Constraints

Lists all the .ldc constraints that you created and also any constraints that are automatically generated from PLL device constraints.

Constraint Coverage

Displays the percentage of timing constraints used in the design.

Overall Summary

This section gives an overall roundup of all the Setup and Hold timing results. Timing Errors indicates the number of endpoints which violate setup or hold violations. There is also a sum of the Total Negative Slack in terms of setup and hold.

Unconstrained Report

Reports the paths that are not constrained and shows the start point and end point of each path. Suggested timing constraints are provided to constrain the given paths.

Combinational Loop

Lists combinational / asynchronous loops in the circuit for which timing paths cannot be analyzed.

Setup at Speed Grade 1 Corner at 85 Degrees

This section shows the clock constraint added in the design, Single Clock Domain table, and Clock Domain Crossing table.

If a clock net goes to a pin with a minimum pulse width (MPW) arc, the corresponding clock's report will contain an MPW report immediately after the constraint. Each clock constraint has a maximum of one MPW report in the Single Clock Domain table.

Clock Summary

Shows an analysis for each clock domain defined in the constraints. The analysis shows the target frequency versus the actual (or maximum possible) and MPW (minimum pulse width) frequencies.

The "Clock Domain Crossing" section shows the slack with any other clocks that connect with the given domain. That is, a data path that has different clocks for its start and end points.

Endpoint Slacks

Lists the endpoints, slack, and the total setup number of endpoints with negative slack.

Detailed Report

Shows details of the worst paths for setup slack and for hold slack. Each path has a section that starts with a summary of the path and the results of the analysis. This is followed by a table calculating the delay step by step through the path, beginning with the clock at the start of the path and ending with the clock at the end of the path. Each step includes the name of the pin within the FPGA and the hierarchical name of the module's port, the type of delay, and the fanout from the pin.

If you want to visualize the path, the reports have links to other tools. Physical Designer Placement Mode shows the sites within the FPGA. Physical Designer Routing Mode shows the route within the FPGA. (Physical Designer is only available after place-and-route.) Netlist Analyzer shows a schematic view of the design. However, Netlist Analyzer often cannot show the detailed path.

The setup time of the IOLOGIC register input is now independent of the DELAY timing. It is constant and unchanging, regardless of the value set for DEL_VALUE in the DELAY component.

The IOLOGIC had a programmable DELAYB component in it. This delay was modeled as part of the setup and hold timing arcs of the IOLOGIC block.

The DELAYB component has been removed from the IOLOGIC block. As a result, the setup and hold timing arcs of the IOLOGIC block no longer have the DELAYB-related delays added to them. The DELAYB will appear separately.

Figure 31: Example of Timing Analysis Report Enhancement for Delay Buffer

```
++++Path 1  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

  Destination Clock Arrival Time (clk_i:R#2)    10.000
+ Destination Clock Source Latency               0.000
- Destination Clock Uncertainty                  0.000
+ Destination Clock Path Delay                   4.172
- Setup Time                                     0.215
-----------------------------------------   -------
End-of-path required time( ns )                13.957
```

Example:

If the current DEL_VALUE is 50, and you changed it to a higher value, the **Setup Time** remains constant at 0.215.

Prior to this enhancement, the value of **Setup Time** keeps changing.

Setup at Speed Grade 1 Corner at 0 Degrees

Shows the clock constraint added in the design, Single Clock Domain table, and Clock Domain Crossing table. If a clock net goes to a pin with a minimum pulse width (MPW) arc, the corresponding clock's report will contain an MPW report immediately after the constraint. Each clock constraint has a maximum of one MPW report in the Single Clock Domain table.

This section also describes the Clock Summary, Endpoint Slacks, and Detailed Report.

Hold at Speed Grade M Corner at 0 Degrees

Displays the number of endpoints, slacks, and the total hold number of endpoints with negative slack. This section also describes the Endpoint Slacks and Detailed Report.

Figure 32: Example of Timing Analysis Report Table of Contents

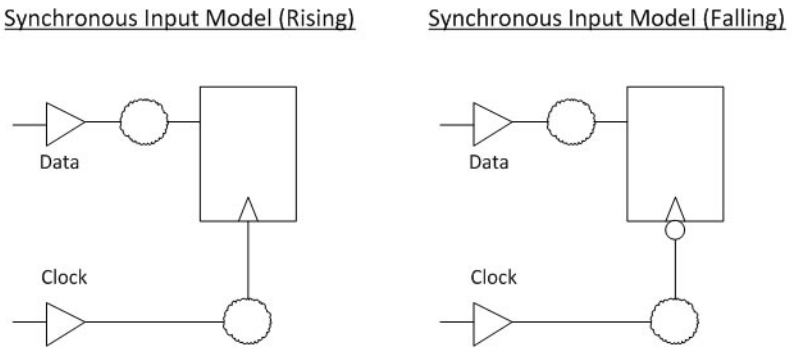
```
1 Timing Overview
    1.1  SDC Constraints
    1.2  Constraint Coverage
    1.3  Overall Summary
    1.4  Unconstrained Report
    1.5  Combinational Loop
2 Setup at Speed Grade 1 Corner at 85 Degrees
    2.1  Clock Summary
    2.2  Endpoint Slacks
    2.3  Detailed Report
3 Setup at Speed Grade 1 Corner at 0 Degrees
    3.1  Clock Summary
    3.2  Endpoint Slacks
    3.3  Detailed Report
4 Hold at Speed Grade M Corner at 0 Degrees
    4.1  Endpoint Slacks
    4.2  Detailed Report
```

Understanding 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.

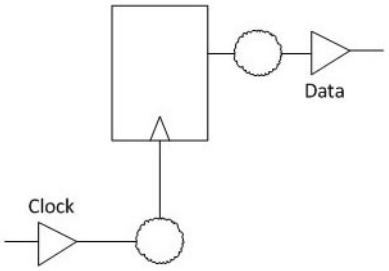
```
Port     Clock     Edge  Setup Performance_Grade  Hold Performance_Grade
------------------------------------------------------------------------
din_f    clkin     F    -0.855      M            2.344     7
din_r    clkin     R    -0.855      M            2.344     7
```

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 reference 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 analyzed 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:

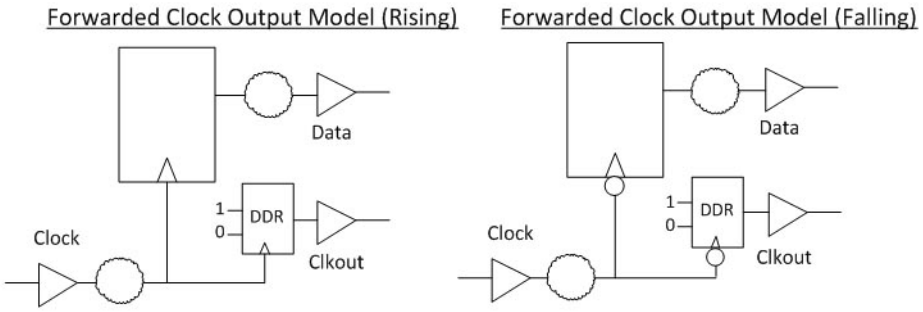
```
Port       Clock     Edge  Max_Delay Performance_Grade  Min_Delay Performance_Grade
------------------------------------------------------------------------------
dout       clkin      R     7.031         7              3.365          M
```

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:

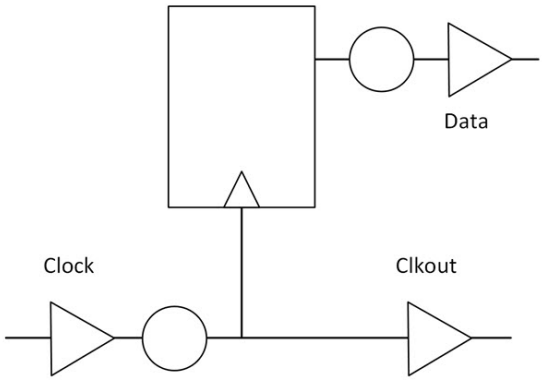
To find out about specific Lattice FPGA architectures and high-speed interface recommendations, visit the Lattice website and search for device-specific application notes.

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 reference 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 356. 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.

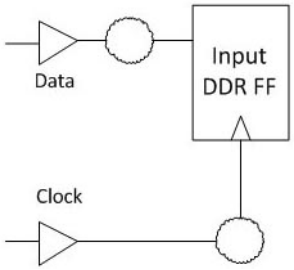


Not Reported

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** – A centered interface has the clock edge centered in the middle of the data valid of the data input before entering the device.
- **Aligned** – 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.

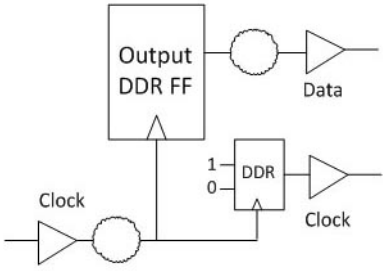
The following is an example report line for an input DDR interface.

```
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** – A centered interface has the clock edge centered in the middle of the data valid of the data output when leaving the device.
- **Aligned** – 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.
- 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 I/O Timing Analysis “All Performance Grade” option in the active strategy is set to False. When the False setting 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 Toolbar, click the Task Detail View and select **I/O Timing Analysis**. Click Place & Route Design to run the whole process.

  Radiant 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 I/O 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 Value column, select the checkbox and click **OK**.
3. In the Process Toolbar, click the Task Detail View and select **I/O Timing Analysis**. Click Place & Route Design to run the whole process.

   Radiant 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 tab as soon as the process has finished.

To view the I/O Timing Analysis report:

- In the Reports tab, expand Place & Route Reports and select **I/O Timing Analysis**.

Using Timing Analyzer

Timing Analyzer is a different way to look at the place-and-route timing analysis that you might find easier to read. Timing Analyzer runs the timing analysis and presents the results on three spreadsheet tabs. Plus, there is a Query tab so you can search through the paths. The information in Timing Analyzer is very similar to that in the Place & Route Timing Analysis report but is presented differently.

Timing Analyzer can be run anytime after completing the place-and-route process. You do not need to select timing analysis in the Task Detail View of the Process Toolbar.

To use Timing Analyzer:

- Choose **Tools > Timing Analyzer**.

  A progress indicator opens, showing that the Radiant software is calculating the delays. It takes a moment for the Timing Analyzer to appear in the Tool Area.

Setting Options in Timing Analyzer

Timing Analyzer has the same options as the Place & Route Timing Analysis Report. But the options are set independently in Timing Analyzer.

The current option settings can be seen on Timing Analyzer's General Information tab.

To change the timing option settings:

1. Choose **Edit > Timing Option Setting.**

   The Timing Option Setting dialogue box appears.
2. Choose option settings. See below for more information.
3. Click **OK**.

   A progress indicator opens, showing that the Radiant software is calculating the delays. It takes a moment for the Timing Analyzer tabs to be updated.

Here are definitions of each option:

- Run Mode: Choose whether you want setup or hold analysis, or both.
- Speed for setup: Choose the performance grade of the FPGA. "m" is for the fastest performance grade.

- Speed for hold: Choose the performance grade of the FPGA. “m” is for the fastest performance grade and is usually the worst case for hold analysis.
- Report Format: Choose the style of the report. The default is the current Lattice Standard, which follows industry standards for timing analysis tools. But if you have been using the Lattice Diamond tool set, you might prefer the older Diamond Style.
- Critical endpoints path number limit: Enter the number of data paths you want in the Critical Endpoint Summary tab.
- Unconstrained endpoints path number limit: Enter the number of data paths you want in the Unconstrained Endpoint Summary tab.
- Number of paths per constraint: Enter the number of data paths you want for each constraint in the Critical Paths Summary tab.
- End point number limit: Enter the number of endpoints you want in the Critical Endpoint Summary tab.
- Maximum slack limit: Enter the maximum delay value for the slack in picoseconds.

Reading Timing Analyzer

Timing Analyzer has five tabs arranged along the bottom of the view. These tabs have information similar to sections 3 and 4 of the Place & Route Timing Analysis report. The information from sections 1 and 2 are not included. See “Reading Timing Analysis Reports” on page 352.

General Information This tab shows basic information about the FPGA and the option settings used in the analysis.

Critical Paths Summary This tab shows the same information as section 4, “Detailed Report,” of the text report. At first you just see introductory information for the paths. You can see the full details on any path by clicking anywhere in the row of that path. See “Path Details in Timing Analyzer” on page 363.

Note

The data in this tab is presented in picoseconds, not nanoseconds.

Critical Endpoint Summary

This tab shows the same information as section 3.2, “Setup Summary Report,” and section 3.3, “Hold Summary Report,” of the text report. You can see the full details on any path by clicking anywhere in the row of that path. See “Path Details in Timing Analyzer” on page 363.

Unconstrained Endpoint Summary

This tab shows the same information as section 3.4, “Unconstrained Report,” of the text report.

Query

This tab shows a query form to search for data paths. Select a source clock, destination clock, start point, or end point. You can select one or several items. Click **Search** to find associated data paths.

Any paths found are shown in a spreadsheet view at the bottom of the form. Again, you might want to enlarge the view by detaching the tool as a separate window. You can see the full details on any path by clicking anywhere in the row of that path. See “Path Details in Timing Analyzer” on page 363.

You can also save queries for future use.

Path Details in Timing Analyzer

In Timing Analyzer, you can see details on any data path listed in the Critical Paths Summary, Critical Endpoint Summary, and Query tabs. These details are similar to what you see in the “Detailed Report” section of the text report.

If you want to visualize the data path, you can cross-probe to Physical Designer.

You can also export the data to a .csv (comma-separated value) file that can be read by other software, such as a spreadsheet tool. See “Exporting a Path Analysis” on page 364.

To see path details:

1. Click anywhere in the row of the desired path.

   The window is divided into three parts. On the right are two new sections with tabs that provide extra information about the selected path. You might want to enlarge the view by detaching the tool as a separate window.

   The Path Detail tab shows the same the introduction to the path seen in the text report. There are also some delay calculations for the destination and source clocks.

   The third section has three tabs that calculate the delay step by step through the path. The Data Path tab shows the steps. The Clock Paths tab shows the clocks at the start and end of the path. The Waveform tab allows you to visually reference the Data Path data. You can use the tab to easily see how and where the constraints apply.

   You can cross-probe from the Critical Paths Summary tab or the Query tab of Timing Analyzer to Netlist Analyzer.

To cross-probe to Physical Designer:

1. Right-click any row in the Data Path tab or the Clock Paths tab and choose:
   - **Show in Physical Designer Placement Mode**
   - **Show in Physical Designer Routing Mode**

Physical Designer opens showing the path in the desired mode.

If you would like to view multiple paths, click and drag over the required rows and right-click with the rows highlighted. You can also use the Shift and Ctrl keys to select contiguous and non-contiguous rows.

You can cross-probe from the Critical Paths Summary tab or the Query tab of Timing Analyzer to Netlist Analyzer.

To cross-probe to Netlist Analyzer:

1. Right-click any row in the Data Path tab or the Clock Paths tab and choose **Show in Netlist Analyzer.**

   Netlist Analyzer opens and highlights the selected path.

   If you would like to view multiple paths, click and drag over the required rows and right-click with the rows highlighted. You can also use the Shift and Ctrl keys to select contiguous and non-contiguous rows.

Exporting a Path Analysis

You can export the analysis from an individual path to a .csv (comma separated file) that can be read by other software, such as a spreadsheet tool. You can do this from any path in the Critical Path Summary, Critical Endpoint Summary, and Query tabs. The Export command saves all the data in the tab plus all the detail data for the selected path.

To export path data:

1. Right-click anywhere in the row of the desired path and choose **Export**.

   The Export dialog box opens.

2. Browse to where you want to save the .csv file.
3. Enter a file name.
4. Click **Save**.

Using Standalone Timing Analyzer

Standalone Timing Analyzer can be run on designs using post-synthesis, post-map, post-PAR Unified Design Database (.udb), and associated timing constraints specified in the .pdc file of the design. Using these input files, it provides static timing analysis, path tables, and a report of each timing constraint like the Timing Analyzer in Radiant.

Running Standalone Timing Analyzer from Radiant Timing Analyzer

When you open the Standalone Analyzer tool, it will automatically create the Unified Design Database, which includes logic and physical data (standalone_tav_*.udb). This file is saved to the current project. You can use this database to run tavmain.exe.

To open and run a timing analysis in Standalone Timing Analyzer:

- In the Radiant software main window, click the button in the upper left. You can select a different .udb file.
- If you wish to open the Standalone Timing Analyzer directly, click the button in the upper left or choose **Tools** > **Standalone Timing Analyzer**.

See Also

- Standalone Timing Analyzer

Loading Timing Constraint File and Running Timing Analysis in Standalone Timing Analyzer

To load a timing constraint file and run timing analysis:

1. In the Timing Analyzer main window, choose **File** > **Load Timing Constraint File**. This menu can be used after opening the design.
2. After selecting the constraint file, you can load the timing constraint and do the timing analysis based on the selected constraint file.
3. If you select Load Timing Constraint File (Incremental Flow), do the timing analysis based on the constraints in the .udb file and the selected constraint file.

Exporting Timing Report and Exporting Timing Analysis Reports in Standalone Timing Analyzer

To export timing report and timing analysis report:

- In the Timing Analyzer main window, choose **File** > **Export** > **Timing Report For Setup/Hold**. This menu can export timing analysis reports for setup or hold (.twr).

Exporting Timing Constraints to PDC File in Standalone Timing Analyzer

To export timing constraints to PDC file:

- In the Timing Analyzer main window, choose **File** > **Export** > **Export Timing Constraints To PDC file**.

  This menu can be used after opening the design.

- You can export timing constraints in the current .udb file to a .pdc file.

Saving Design Files in Standalone Timing Analyzer

To save design files in Standalone Timing Analyzer:

- You can save changes to design data. In the Timing Analyzer main window, choose **File** > **Save Design As**.

Setting Options in Standalone Timing Analyzer

Setting options in Standalone Timing Analyzer is the same as setting options in Radiant Timing Analyzer." Refer to “Setting Options in Timing Analyzer” on page 361.

To set options in Timing Analyzer:

- In the Timing Analyzer main window, choose **Edit > Timing Option Setting**, or click the  button in the upper left.

  This menu can be used after opening the design.

- When loading the post-synthesis .udb file, you cannot change the speed in the Timing Option Setting dialog box. This feature is the same as the timing report.

- The default for the Report Format is Lattice Standard. You can change the Report Format in the Timing Option Setting dialog box.

Setting Operating Conditions in Standalone Timing Analyzer

To set operating conditions in Standalone Timing Analyzer:

1. In the Timing Analyzer main window, choose **Edit** > **Operating Conditions Setting**.

   You can change the voltage and temperature based on the current design data.

2. When you click the **OK** button, the timing analysis process starts based on the edited design data.

3. When loading the post-synthesis .udb file, you cannot change the settings in the Operating Conditions Setting dialog box. This feature is the same as timing report.

Running Timing Constraint Editor in Standalone Timing Analyzer

To run timing constraint editor in Standalone Timing Analyzer:

1. In the Timing Analyzer main window, choose **Edit** > T**iming Constraint Editor**. After selecting the Timing Constraint Editor, you can edit the timing constraint design based on the current design data.

2. When you close the Post-Synthesis Timing Constraint Editor window, a dialog box appears. If you click yes, the timing analysis process starts based on the edited design data.

Refreshing Timing in Standalone Timing Analyzer

To refresh timing in Standalone Timing Analyzer:

- In the Timing Analyzer main window, choose **Edit** > **Refresh**. Based on the current timing constraint in the .pdc file, you can do the timing analysis to check if the new constraint works as expected.

To edit the location of the current .pdc file:

- You can add the new constraint in the current .pdc file.

  Click the **Refresh** button to start the timing analysis process.

- To add the new constraint in the .pdc file, you can copy the text from the timing table.

- If the location of the current .pdc file is not specified, you can also choose **File** > **Load Timing Constraint File** to specify the .pdc file and start the timing analysis process.

Analyzing Power Consumption

Included with the Radiant 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 estimated power consumption.

Power Calculator allows you to import frequency and 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: Used before completing the design;
- Calculation mode: Based on the physical netlist file (.udb) after placement and routing.

Power Calculator also features:

- User defined Junction temperature
  - Users can directly enter Tj now, as opposed to having to use workarounds to modify it indirectly
  - Modify Tj for your design
- General Flow checks using power calculator

You can open Power Calculator from within the Radiant software 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 the Radiant software installed.

Calculation Mode

Within the Radiant software environment, Power Calculator calculates power based on the project's files, including device information. This information is automatically extracted when you open Power Calculator from within the Radiant software. In the stand-alone Power Calculator, power consumption is calculated based on a selected native circuit description (.udb) 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 Radiant software environment, Power Calculator estimates power for an 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 376
- "Starting Power Calculator from the Radiant Software" on page 370
- "Starting Power Calculator as a Stand-Alone Tool" on page 371
- "Power Analysis Design Flow" on page 372

Starting Power Calculator from the Radiant Software

Power Calculator is available from the Radiant software as soon as a project is opened.

To start Power Calculator from the Radiant software Tools menu or toolbar:

1. 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.

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.

See Also

- "Running Power Calculator from the Tcl Console" on page 372
- "Saving a Power Calculator File" on page 388
- "Adding Power Calculator Files to the Analysis Files Folder" on page 390
- "Setting a Power Calculator File as the Active Analysis File" on page 392

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 the Radiant software. 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. In the Start menu, choose **Lattice Radiant Power Calculator**.
2. After the Power Calculator Startup Wizard appears, do one of the following and click **OK**:
   - To calculate power consumption based on a mapped or placed and routed design, select **Calculate power with design (udb)**.

     Click the UDB File Browse button to navigate to the .udb file.

     After you select the .udb 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 .udb file name, type the name in the File Name text box.
   - 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.

See Also

- "Running Power Calculator from the Tcl Console" on page 372
- "Saving a Power Calculator File" on page 388

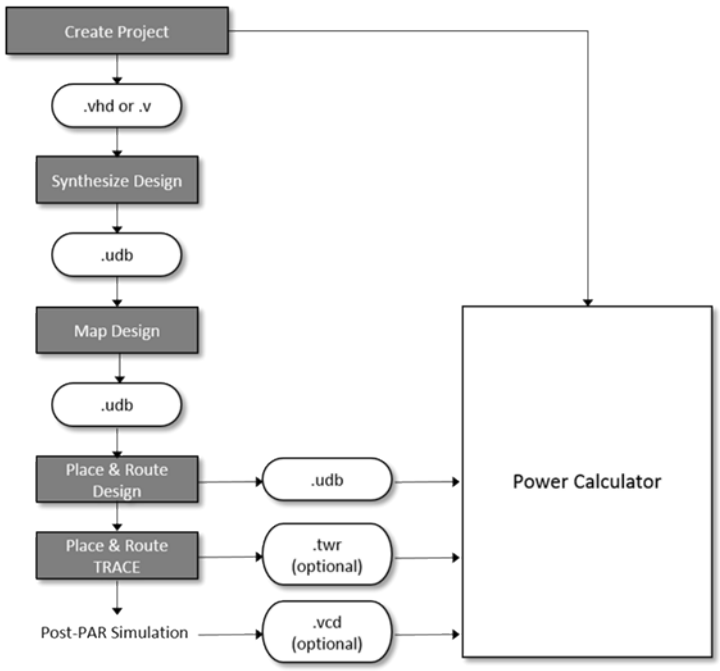
Running Power Calculator from the Tcl Console

The Radiant software'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 1676 in the Tcl Command Reference Guide.

Power Analysis Design Flow

Power Calculator supports all Lattice FPGA devices. The design flow involved in using Power Calculator in the Radiant software is as follows.

Figure 33: Power Calculator Design Flow



See Also

- "Inputs" on page 373
- "Outputs" on page 374

Inputs

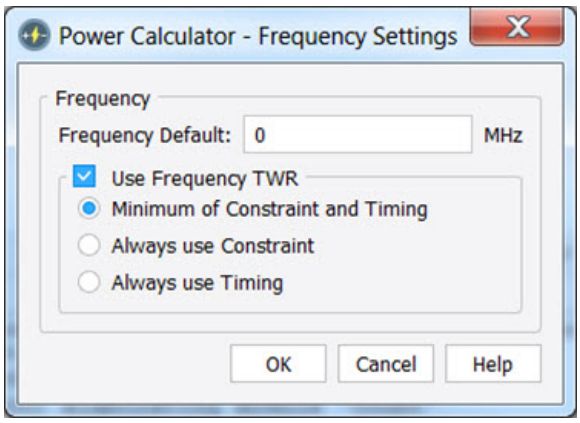
When you first launch Power Calculator from the Radiant software, it displays information from your design project. For an unrouted design, it shows default resource information based on the targeted device. For a routed design, it extracts information from the placed and routed .udb file.

Additionally, Power Calculator accepts the following 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 QuestaSim.

  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 all, in the pages that contain these columns.
- Timing report file, <*project_name*>.twr, is an ASCII file containing activity and frequency information. The type of frequency and setting can be imported from the timing report file as shown in Figure 34.

Figure 34: Frequency Settings



See Also

- "Importing a Value Change Dump (.vcd) File" on page 398
- "Changing the Global Default Activity Factor" on page 397

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 388
- "Generating Power Graphs" on page 407
- "Viewing the Power Calculator Report" on page 409

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 (.udb) 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 a 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.

Understanding the Default Activity Factor

Power Calculator assigns a default activity factor of 12.5% to most data signals. This means that these signals, on the average, change state once every 8 clock cycles. For example, the AF% of a signal in the Logic Block page is set at 12.5% by default.

Clock-related signals, on the average, change once every cycle. As such, these signals are assigned a default activity factor of 100%. A PLL is an example of such periodic signals that change every cycle.

Some default AF% values are also set at 10%. An example is the default activity factor of some categories on a RAM. This means that these signals, on the average, change every 10 clock cycles.

Most of these default activity factor values are widely accepted in the industry.

Note

MPP data is transferred at very fast rates that a default activity factor of 100% is provided/recommended for MPPHYX4 and MPPCIEX8.

See Also

- “Importing a Value Change Dump (.vcd) File” on page 398
- “Changing the Global Default Activity Factor” on page 397
- “Changing the Global Default Frequency Setting” on page 398

Enable Factor Calculation

To calculate the power consumption in some IP, such as Logic Block, DSP, etc., Power Calculator requires an enable factor (EF) percentage from 0 to 100%. The EF is the percentage of time that the IP is enabled. Enter the enable factor percentage in the "EF (%)" column.

The enable factor percentage appears as the “EF (%)” column.

See Also

- “Importing a Value Change Dump (.vcd) File” on page 398

Power Calculator Window Features

Power Calculator 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 bottom 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 .udb file, or from an external file such as a .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 before Place & Route 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 397 for more information.

Power Summary

Power Summary provides an overview of power consumption conditions. It is the first page that opens 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. As this is device dependent, please refer to the device data sheet for more accuracy.

- 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.”

Environment Section

The Environment section displays information about the operating temperature of the device.

If Computed T_J is selected, the Thermal Profile button is enabled for 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*

User defined Junction temperature

- Users can directly enter Tj now, as opposed to having to use workarounds to modify it indirectly
- Modify Tj for your design

You can enter junction temperature directly through User Entered Tj and Computed Tj. The Computed Tj is the default when the Environment section is displayed.

When User Entered Tj is selected:

- Thermal Profile is disabled.
- Ambient Temperature is hidden.
- Junction Temperature is enabled and allows user to enter values.
- Effective Theta-JA is not applicable and displays N/A.

Junction Temperature specifies the temperature of the device 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 higher than 125 degrees Celsius or higher than the maximum junction temperature allowed for a particular device and operating conditions, a red text appears in the Junction Temperature box.

Note

When you select User Entered T_J, you can directly type a value 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, the value of which is rounded to three digits after the decimal point.

Power by Block

This section displays the total power consumption by the different types of blocks in watts, the value of which is rounded to three digits after the decimal point.

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.

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 bottom 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 depends on the selected device.

Pages Available for Most Device Families

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.

The Block RAM page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing “SP RAM > # EBR Blocks,” the Utilization table is updated after power calculation.

Misc

The Misc page includes any non-generic IP, such as oscillators, i2c, and spi. These IPs do not have their own page due to negligible power contribution.

The Misc page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing “OSC > # OSC,” the Utilization table is updated after power calculation.

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 internal feedback architecture of PLL in the device.

The PLL page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing “PLL > # PLL,” the Utilization table is updated after power calculation.

The PLL page is available for all devices.

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.

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 and enable factor of LUTs, distributed RAMs, ripple-carry logic circuits, and registers driven by the clock.

The Logic Block page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing “Logic > # Logic LUTs,” the Utilization table is updated after power calculation.

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.

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 .udb file, Power Calculator extracts the information for the I/O page directly from the .udb file. If you do not use an .udb file, Power Calculator assigns the default resource usage values according to the design's family.

You can enable I/O standby mode from Bank I/O table or from user's design. If "Allow Standby INR" is "Yes" in bank voltage table, all differential input/bidi I/Os are standby-enabled; if "Allow Standby INR" is "No" in bank voltage table, all differential input/bidi I/Os are standby-disabled except MIPI DPHY input/bidi I/Os, MIPI DPHY input/bidi I/Os can be either standby-enabled or standby-disabled. If "Allow Standby DIFFIO" is "Yes" in bank voltage table, all differential output I/Os are standby-enabled; if "Allow Standby DIFFIO" is "No" in bank voltage table, all differential output I/Os are standby-disabled except MIPI DPHY output I/Os, MIPI DPHY output I/Os can be either standby-enabled or standby-disabled.

For certain devices that support boundary scanning, there is a radio option at the top of the I/O tab which allows users to enable or disable boundary scanning. Disabling this option will set Vccjb to 0 volts in the power summary page. Enabling this option will update Vccjb in the power summary tab to reflect that boundary scanning is on. By default, this option is disabled and will not account for boundary scanning.

Devices that do not support boundary scanning do not have this option in the I/O tab, and as such will not account for boundary scanning in the power estimation results, or the power summary tab.

The I/O page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing "Input Output WRIO > # I/P," the Utilization table is updated after power calculation.

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.

The I/O Termination page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing "Termination > # I/P," the Utilization table is updated after power calculation.

Graph

The Graph page displays three types of graphs:

- Power vs. VCC Supply Voltage
- Power vs. Ambient Temperature

- 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 407 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 both text (ASCII) and HTML. It is updated each time you make a change to any of the data in the editable cells.

Pages Dependent on Selected Device/ Device Architecture

LED

The LED page allows you to generate the constant current sources. The signal name for Input is "EN" and "LEDPU" for Output. Default Signal Value for unconnected port: Input is "0"

Input EN : Logic "0"

The LED page is available for all devices.

DDRPHY

The DDRPHY page displays the estimated power consumed by DDRPHY in the design and the factors affecting it. The frequency, activity factor, enable factor, and type and number of DDRPHY's powered by each clock are all required for the power-consumption calculation on this page.

The DDRPHY page also displays Utilization table, showing the percentage of resources used for the IP. When editing "DDRPHY > # DDRPHY," the Utilization table is updated after power calculation.

MPP

The MPP page displays the estimated power consumed by MPPHYX4 and MPPCIEX8 in the design and the factors affecting it. The protocol, data rate, activity factor, enable factor, link width, and number of MPPHYX4s and MPPCIEX8s are all required for the power-consumption calculation on this page. The MPP uses 100% activity factor by default.

The MPPHYX4s and MPPCIEX8s options can be selected by using the pull down arrow above the table entry. When selected, options are displayed with corresponding protocol note.

The MPPs can only be added for devices supporting it, and there is a DRC which checks to ensure users do not add more MPPs than their device supports. Also, only protocol and type can be selected with the dropdown, the other settings are user-entered directly.

The MPP page also displays Utilization table for each tab, showing the percentage of resources used for each IP. When editing "MPPHYX4_MPQ0 > # MPPHYX4_MPQ0," the Utilization table is updated after power calculation.

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, enable factor, and the type and number of DSPs driven by each clock.

The DSP page also displays Utilization table, showing the percentage of resources used for the IP. When editing "DSP > # DSP," the Utilization table is updated after power calculation.

SRAM

The SRAM page displays the power consumed by the single port RAM used by the design and the factors that affect it. Power-consumption calculation for the SRAM page requires the frequency and activity factor per clock domain.

LRAM

The LRAM page displays the power consumed by the Large RAM blocks and the factors that affect it. It allows the entry of LRAM type, frequency, activity factor, enable factor, and number of blocks.

SGMIICDR

The SGMIICDR page displays the power consumed by the SGMII (Serial Gigabit Media Independent Interface)/CDR (Clock Data Recovery) blocks and the factors that affect it. It allows the entry of frequency, activity factor, enable factor, and number of blocks.

DDRDLL

The DDRDLL page displays the power consumed by the DDRDLL (Double Data Rate Delay Locked Loop) master delay control blocks and the factors that affect it. It allows the entry of frequency, activity factor, enable factor, and number of blocks.

The DDRDLL page also displays Utilization table, showing the percentage of resources used for the IP. When editing “DDRDLL > # DDRDLL,” the Utilization table is updated after power calculation.

DLLDEL

The DLLDEL page displays the power consumed by the DLLDEL (Delay Locked Loop Delay) slave delay blocks and the factors that affect it. It allows the entry of frequency, activity factor, enable factor, and number of blocks.

The DLLDEL page also displays Utilization table, showing the percentage of resources used for the IP. When editing “DLLDEL > # DLLDEL,” the Utilization table is updated after power calculation.

DQS

The DQS page displays the power consumed by the DQS (DDR memory strobe) blocks and the factors that affect it. It allows the entry of DQS gearing mode, frequency, activity factor, enable factor, and number of blocks.

MIPIDPHY

The MIPIDPHY page displays the power consumed by the MIPIDPHY (Mobile Industry Processor Interface Physical layer) blocks and the factors that affect it. It allows the entry of data rate, activity factor, and duty cycles for transmit/receive and low power/high speed. The MIPIDPHY has VCCDPHY and others power domains, and it uses 100% activity factor and 80% HS duty cycle for all dynamic power calculation. Customer can specify HS duty cycle in either continuous (normal power consumption) or non-continuous (power saving condition) clocking mode. For continuous clocking mode, the MIPIDPHY supports X4/X3/X2/X1 mode operation.

The MIPIDPHY page is available for CrossLink-NX device only.

ADC

The ADC page displays the power consumed by the ADC (Analog to Digital Converter) block and the factors that affect it. It allows the entry of frequency, activity factor, enable factor, number of blocks, and comparator enables.

ALU

The ALU page displays the power consumed by the ALU (Arithmetic Logic Unit) block and the factors that affect it. It allows the entry of frequency, activity factor, enable factor, and number of blocks.

PCIE

The PCIE page displays the power consumed by the PCIe (Peripheral Component Interconnect Express) hardened block and the factors that affect it. It allows the entry of activity factor, enable factor, number of blocks, and PCIe speed. The data rate (2.5G, 5G, 8G) cannot be input directly, it is based on the type. The PCIE uses 100% activity factor by default for all dynamic power calculation including VCC power domain.

PCS

The PCS page displays the power consumed by the PCS hardened block and the factors that affect it. It allows the entry of activity factor, enable factor, number of blocks, number of lanes and data rate. It includes PCIE_LL (PCIe Link layer) and PCS table. The data rate (2.5G, 5G, 8G) in PCIE_LL cannot be input directly, it is based on the type. When PCIE_LL is used (there is a line in PCIE_LL table and # PCIE = 1), the PCS data rate will be set to the same value as PCIE_LL data rate automatically. If no PCIE_LL is used, the PCS data rate can be set by user.

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.
- Green – The cell is read-only or contains output from the software.
- Lite Yellow – The cell contains data extracted from a design file, such as an .udb 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 higher than 125 degrees Celsius or higher than the maximum junction temperature allowed for a particular device and operating conditions. The Junction Temperature box is the only cell that can display red text.

The font colors in the cells on Power Calculator pages have the following significance:

- Blue – Indicates default values.
- Grey out – 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. The cells in the # of Outputs and the # of Bidi columns, however, would display values in gray 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 .udb 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 .udb 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) for power calculation.

Creating a New Power Calculator File in the Radiant Software

You can create a new .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 the Radiant software, 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) is to 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 Implementation” option adds the new .pcf file to the Analysis files folder and 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 Implementation” 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 the Radiant software, 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 Implementation” 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 373
- “Saving a Power Calculator File” on page 388
- “Outputs” on page 374
- “Opening an Existing Power Calculator File in the Radiant Software” on page 389

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 .udb file for the project, browse to the location of the .udb file. Otherwise, leave the .udb box empty.
4. Click **OK**.

   If you selected an .udb file in Step 3, Power Calculator loads the new project in calculation mode. If you did not select an .udb file, Power Calculator loads the new project in estimation mode.

See Also

- "Inputs" on page 373
- "Saving a Power Calculator File" on page 388
- "Opening an Existing Power Calculator File in the Stand-Alone Power Calculator" on page 390

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 390
- "Setting a Power Calculator File as the Active Analysis File" on page 392
- "Outputs" on page 374

Opening an Existing Power Calculator File in the Radiant Software

You can open an existing Power Calculator (.pcf) file in the Radiant software before or after opening Power Calculator. When you open a .pcf file in the Radiant software, 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 as the default file type.
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.

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 the Radiant software, 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 390
- "Setting a Power Calculator File as the Active Analysis File" on page 392

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 .udb file is associated with the selected .pcf file, Power Calculator loads the project in calculation mode. If no .udb 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 388
- "Creating a New Power Calculator File in the Stand-Alone Power Calculator" on page 387

Adding Power Calculator Files to the Analysis Files Folder

The Radiant software 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 the Radiant software, 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 391
- “Creating a New Power Calculator File in the Radiant Software” on page 386
- “Setting a Power Calculator File as the Active Analysis File” on page 392

Removing Power Calculator Files from the Analysis Files Folder

There may be times when you 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 order for 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 .udb file when you open Power Calculator. If no .udb 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 PCF**.

   The Radiant software displays the .pcf file name in bold type, indicating that it is the active power analysis file for the current design implementation. The Radiant software does not automatically load the newly activated file into Power Calculator if a different .pcf file is already open.
3. To open the active .pcf file, if a previous file is 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

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 .udb file. “Untitled” appears in the title bar.

Entering Data

When you have a design open in the Radiant software, 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, enable 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 yet 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, activity factor by changing the default setting in the Frequency Settings, Activity Factor, and Enable Factor Settings dialog boxes. You can use a simulation file to populate Frequency, AF (activity factor) cells.

See Also

- “Software Mode” on page 376
- “Reverting to Calculation Mode” on page 397
- “Power Calculator Pages” on page 379

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 379
- "Color Coding of Cells" on page 385
- "Cutting and Pasting Cell Contents" on page 395
- "Copying and Pasting Cell Contents" on page 396
- "Changing Values Automatically" on page 396

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, the number can't be entered, user may move cursor to the editable cell, and refer to the value range 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.
   - Select from User Entered Tj and Computed Tj to enter junction temperature directly.

     The Computed Tj is the default when the Environment section is displayed. Change the ambient temperature, as desired.

     When User Entered Tj is selected:

     - Thermal Profile is disabled.

- Ambient Temperature is hidden.
- Junction Temperature is enabled and allow us to enter values.
- Effective Theta-JA is not applicable and displays N/A.

See “Controlling Operating Temperature” on page 402

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 379
- Saving a Power Calculator File
- “Adding Power Calculator Files to the Analysis Files Folder” on page 390
- “Setting a Power Calculator File as the Active Analysis File” on page 392
- “Editing Cells” on page 393

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 light yellow background. Some columns with a light yellow 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 393

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 393
- "Changing Values Automatically" on page 396

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 393
- "Changing Values Automatically" on page 396

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, activity factor cells.

See Also

- "Changing the Global Default Activity Factor" on page 397
- "Changing the Global Default Frequency Setting" on page 398

- "Importing a Value Change Dump (.vcd) File" on page 398

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:

- 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 376

Changing the Global Default Activity Factor

Power Calculator automatically assigns a default activity factor 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 374
- "Importing a Value Change Dump (.vcd) File" on page 398

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, 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 Frequency and AF cells are now populated with the data from the .vcd file.

See Also

- "Changing the Global Default Frequency Setting" on page 398

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 in one of two ways:

- Specify a value in the Frequency Settings dialog box.
- In the Frequency Settings dialog box, select **Use Frequency TWR** options to import frequencies from the timing report.

Note

The frequency of some clocks, such as those that have high dependency on user constraints and usage, cannot be imported by using **Use Frequency TWR** options. In cases where clock frequency cannot be imported, you must specify a value.

To globally change the default frequency setting by specifying a value:

1. Choose **Edit** > **Frequency Settings**.
2. In the Frequency Settings dialog box, enter the new frequency 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 of these options, make sure timing report contains the names of the clocks in the pages for which you want default frequency values.
   - Minimum of Constraint And Timing – 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 timing report. The frequency is in megahertz.
   - Always Use Constraint– Specifies that the default frequency in the Frequency (MHz) column of the Power Calculator pages be taken from the constrained frequency in timing report. The frequency is in megahertz.
   - Always Use Timing– Specifies that the default frequency in the Frequency (MHz) column of the Power Calculator pages be taken from the actual frequency in timing report. The frequency is in megahertz.
3. Click **OK**.

Power Calculator now populates the Frequency cells of its pages with the frequency data from the timing report.

See Also

- “Inputs” on page 373

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 400

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.

- **Low** – Uses a small amount of routing resources.

  Select this option if you have a datapath oriented FPGA architecture with low fanout nets.

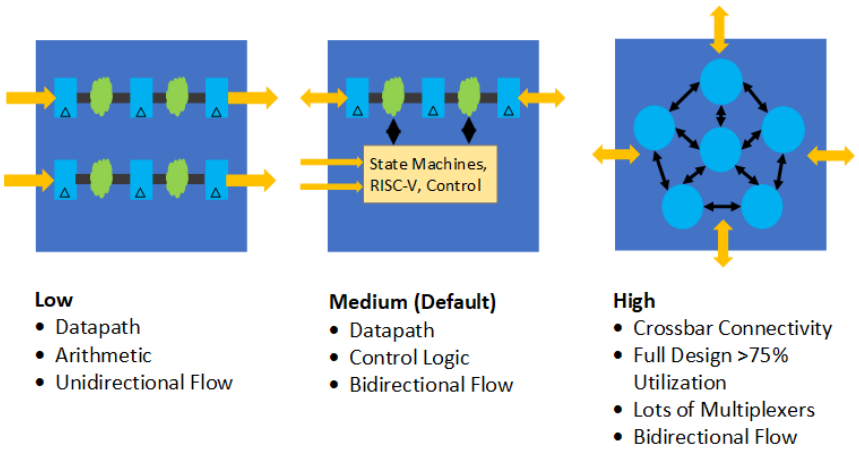
- **Medium** – Uses an average amount of routing resources. This is the default setting

  Select this option if your design is a mix of datapath and control style logic. If unsure of your design style, retain the Medium default setting.

- **High** – Uses a large amount of routing resources.

  Select this option if your design has high numbers of high fanout nets that span large areas of the FPGA.

These settings are further illustrated in the figure below.



3. Click **OK**.

See Also

- "Estimating Resource Usage" on page 400

Running Immediate Calculation

The Calculate button allows you to run power calculation with a mouse-click after you have made changes to your power calculation.

To enable the Calculate button:

1. Choose **Edit**, then uncheck **Calculate Immediately**. The Calculate button appears on the top-right of the Power Summary page.

   Note:

   Any changes cause a note "**Power result is dirty. Manually click 'Calculate' button to run power calculation.**" shown in red color.

2. Click the **Calculate** button. The power calculation result will be updated, and the red note will disappear.

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 selecting **Computed T_J** and 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 379

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, select **Computed T_J** and 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 Effective Theta-JA**.

3. If you chose Use Thermal Models:

   a. Select the size of the board that to use 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, including:

   - The number of board layers.
   - Airflow over the board.
   - Number of devices powered up around the device package and their distance from the package.
   - Thickness of the copper and the width of the traces on each layer.
   - Number of thermal vias.
   - Shape and thickness of each power and ground plane below the transfer of heat from the die to the ambient environment.

   As a result, 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 402
- “Changing the Ambient Temperature” on page 405

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:

1. In the Environment section of the Power Summary page, select **Computed T_J**.

2. 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 (A) and Total (A) columns of the Current by Power Supply and Power by Power Supply 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 402
- “Selecting a Thermal Impedance Model” on page 403

Changing the Junction Temperature

The junction temperature is the temperature of the die during operation. A higher junction temperature increases the chances of having thermal issues. Check the valid junction temperature range as specified in Operating Conditions in your device data sheet.

To change the junction temperature:

1. In the Environment section of the Power Summary page, select **User Entered T_J**.
2. Enter a value, in degrees Celsius in the **Junction Temperature (C)** box and press **Enter** or click anywhere outside the cell.

   The value entered must be with the valid range in degrees Celsius. The default value is 25 degrees Celsius for all devices.

Once you select **User Entered T_J**, Thermal Profile is disabled, Ambient Temperature (C)” is hidden, and Effective Theta-JA(C/W)” is N/A, which means the value is zero.

If the junction temperature of the die is not within the valid temperature ranges, the performance, reliability, and functionality can get affected. 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 402

- "Selecting a Thermal Impedance Model" on page 403

Viewing and Printing Results

In addition to the automatically generated reports from power settings, Power Calculator also 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
- 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 specific type of block power to display, 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. Once the graphic data has been prepared, 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 specific 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. Once the graphic data has been prepared, 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 the Power Calculator. It is available in text (ASCII), HTML, and .csv formats, 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 **View HTML Report**.
- To view the results in .csv format, click **Generate CSV Report**.

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 arrows 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 393
- “Power Calculator Pages” on page 379

Generating a Power Calculator Report in CSV Format

A consolidated report of data in all the Power Calculator pages can be generated in .csv format. This allows you to analyze the results in an external spreadsheet application that supports CSV.

To generate a Power Calculator report in CSV format:

1. Click the **Generate CSV Report** button at the upper right-hand corner of the Power Calculator window.
2. In the Generate CSV Report dialog box, browse to the target directory for the report.
3. Type a name in the File name box and click Save.

Chapter 10

Using IBIS Model Files

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 Input/Output Buffer Information Specification (IBIS) format models.

IBIS Overview

IBIS is 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.

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.

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.

For detailed and updated information about IBIS, visit the IBIS Open Forum website.

Lattice Semiconductor IBIS Model Files

Lattice provides two types of IBIS model files for use in your project:

- Design-specific IBIS models that you can generate from within your Radiant project.
- General-purpose IBIS models based on your device, which are available in the Radiant installation folder or from the specific device web page in the Lattice website.

Both types can be used on various EDA tools supporting the IBIS standard.

Design specific IBIS models that are generated within Radiant software represent the behavior of the input/output buffers based on the I/O standard and settings of the design. The generated IBIS files include the package parasitics information such as resistance, inductance, and capacitance that are specific to the design, resulting in simulation that may be more precise.

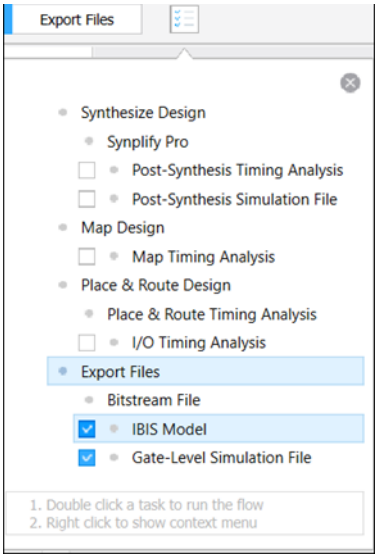
General purpose IBIS models are used for common device families and not specific to any particular design. These are ideal for initial simulations or when specific design details are not yet available.

Generating Design-Specific IBIS Models in the Radiant Software

IBIS generates .ibs and .pkg files for the design.

You can generate your own IBIS models by performing the following steps:

1. In Task Detail view, select the **IBIS Model** option under Export Files.



Click **Export Files** to run the whole process.

2. A **.pkg** and a **.ibs** file will be created in your project directory after the process has been completed.

   - .ibs file – this process generates a design-specific model file *project_name.ibs* in your project directory.
   - .pkg file – IBIS generation copies the correct .pkg file from *<install_dir>/cae_library/ibis* folder to your project directory.

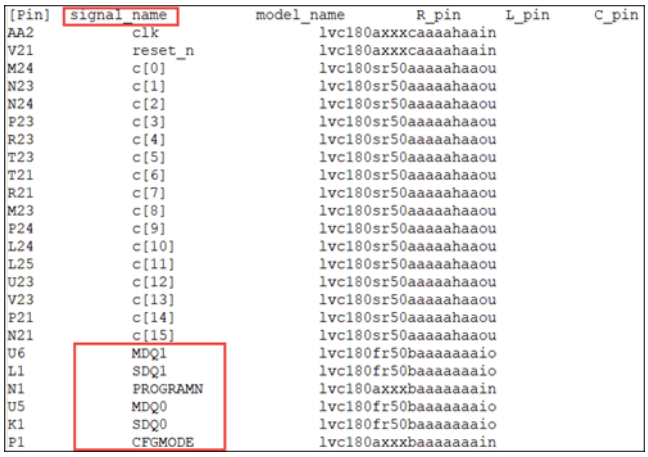
     For example, if you are using the following device:

     - Family: LIFCL (CrossLink-NX)
     - Device: LIFCL-40
     - Package: CABGA400

     The **lifcl-40-cabga400.pkg** file from the Radiant install directory gets copied to the location where your project is stored.

3. If you are using the Avant device, IBIS incorporates sysCONFIG pin data to the **Pin** column of the generated. ibs file. The CFG (Configuration) of the pin is listed in the **signal_name** column.

   A pin cannot be used in designs when it is being used by sysCONFIG.



[Pin]	signal_name	model_name	R_pin	L_pin	C_pin
AA2	clk	lvc180axxxcaaaahaain			
V21	reset_n	lvc180axxxcaaaahaain			
M24	c[0]	lvc180sr50aaaaahaaou			
N23	c[1]	lvc180sr50aaaaahaaou			
N24	c[2]	lvc180sr50aaaaahaaou			
P23	c[3]	lvc180sr50aaaaahaaou			
R23	c[4]	lvc180sr50aaaaahaaou			
T23	c[5]	lvc180sr50aaaaahaaou			
T21	c[6]	lvc180sr50aaaaahaaou			
R21	c[7]	lvc180sr50aaaaahaaou			
M23	c[8]	lvc180sr50aaaaahaaou			
P24	c[9]	lvc180sr50aaaaahaaou			
L24	c[10]	lvc180sr50aaaaahaaou			
L25	c[11]	lvc180sr50aaaaahaaou			
U23	c[12]	lvc180sr50aaaaahaaou			
V23	c[13]	lvc180sr50aaaaahaaou			
P21	c[14]	lvc180sr50aaaaahaaou			
N21	c[15]	lvc180sr50aaaaahaaou			
U6	MDQ1	lvc180fr50baaaaaaaio			
L1	SDQ1	lvc180fr50baaaaaaaio			
N1	PROGRAMN	lvc180axxxbaaaaaaain			
U5	MDQ0	lvc180fr50baaaaaaaio			
K1	SDQ0	lvc180fr50baaaaaaaio			
P1	CFGMODE	lvc180axxxbaaaaaaain			

Refer to sysCONFIG Settings for more information about sysCONFIG pins.

Copying General-Purpose IBIS Models from Radiant Installation Folder

In the Lattice Radiant software, Lattice Semiconductor provides **IBIS (.ibs) model files** for behavioral simulation. IBIS models are located in the following installation directory path in Radiant: <*install_dir*>/cae_library/ibis.

The following table lists all available IBIS model files in the software.

Table 28: Available Lattice IBIS Models

Family	Lattice Semiconductor IBIS Model Files
iCE40UP	iCE40UP.ibs
LAV-AT	lav-at.ibs
LFCPNX	lfcpnx.ibs
LFD2NX	lfd2nx.ibs
LFMXO5	lfmxo5.ibs
LIFCL	lifcl.ibs
UT24C	ut24c.ibs
UT24CP	ut24cp.ibs

Lattice Semiconductor IBIS models characterize all available devices in a given device family. Files contain multiple data matrices 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 the table above.

Package (.pkg) files are also included in the <*install_dir*>/cae_library/ibis folder. These files specify each pin's LRC (Lower Right Corner) in order to see the skew between pins in a DDR bus. The following table lists all available .pkg files in the software.

Table 29: Available Package Files

Family	Package Files
LAV-AT	lav-at-e70-lfg676.pkg
	lav-at-e70-lfg1156.pkg
	lav-at-g70-lfg676.pkg
	lav-at-g70-lfg1156.pkg
	lav-at-x70-lfg676.pkg
	lav-at-x70-lfg1156.pkg
LFCPNX	lfcpnx-50-asg256.pkg
	lfcpnx-50-bbg484.pkg
	lfcpnx-50-bfg484.pkg
	lfcpnx-50-cbg256.pkg
	lfcpnx-100-asg256.pkg
	lfcpnx-100-bbg484.pkg
	lfcpnx-100-bfg484.pkg
	lfcpnx-100-cbg256.pkg
	lfcpnx-100-lfg672.pkg
LFD2NX	lfd2nx-17-csfbga121.pkg
	lfd2nx-40-cabga196.pkg
	lfd2nx-40-cabga256.pkg
	lfd2nx-40-csfbga121.pkg
LFMXO5	lfmxo5-25-bbg256.pkg
	lfmxo5-25-bbg400.pkg
	lfmxo5-55T-bbg400.pkg
	lfmxo5-100T-bbg400.pkg
LIFCL	lifcl-17-cabga256.pkg
	lifcl-17-csfbga121.pkg
	lifcl-17-qfn72.pkg
	lifcl-17-wlcsp72.pkg
	lifcl-33u-fccsp104.pkg
	lifcl-33u-wlcsp84.pkg
	lifcl-33-wlcsp84.pkg
	lifcl-40-cabga256.pkg
	lifcl-40-cabga400.pkg
	lifcl-40-csbga289.pkg
	lifcl-40-csfbga121.pkg
	lifcl-40-qfn72.pkg

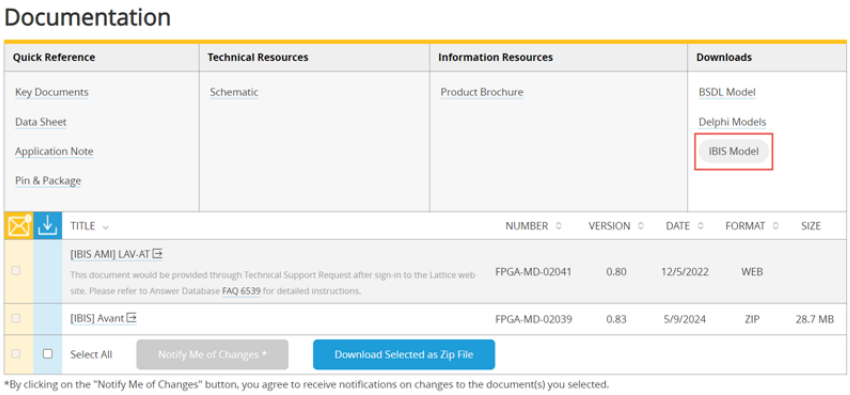
Table 29: Available Package Files

Family	Package Files
UT24C	ut24c40-cabga256.pkg
UT24CP	ut24cp100-bbg484.pkg

Downloading General-Purpose IBIS Models from the Lattice Website

The same IBIS file available in the Radiant installation folder can be downloaded from the Lattice website.

To download IBIS models from the lattice website, navigate to the **Home > Products > FPGA & CPLD > *<Device>*** web page. Scroll down to the Documentation section and click **IBIS Model**. The IBIS models available for the device are listed on a table for you to select and download.



For some devices, Lattice provides IBIS-AMI (Algorithmic Modeling Interface) model files through the Technical Support Answer Database.

Programming the FPGA

After you have created and verified your design, you can use the final output data file to download or upload a bitstream to or from an FPGA device using the Radiant Programmer. Programmer supports serial and microprocessor programming of Lattice devices in PC and Linux environments. A device can be scanned automatically.

The Radiant Programmer is available with the Radiant software, and is also available in a stand-alone version.

Features include:

- Scan and display contents (.xcf file)
- Download data files to devices
- Create/modify/display .xcf file
- Generate embedded file on the .xcf file
- Generate Tester serial vector format (.svf) on the .xcf file

The Radiant Programmer uses a Single Document Interface (SDI) where a single .xcf project is displayed per Programmer instance. Opening an additional .xcf file closes the current .xcf file. To open additional .xcf files in stand-alone mode, a separate instance of the Radiant Programmer needs to be launched.

See Also

- Programming the FPGA
- File Formats
- SPI Flash Support
- Using the Radiant Programmer
- Programmer Options
- Programming and Configuring iCE40 Devices with Programmer

- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

About the Programmer Window

The Radiant Programmer offers several views to help you set up your connection to a target board and to program the FPGAs. If an item is not showing, choose it in the View menu.

Table View The Programmer window includes a table with the devices in the current Programmer project. Use this table to see information about the devices and to set options for programming them.

- To adjust what information is showing in the table, go to **View > Columns** and choose what you want to see.
- To add or remove devices from the table, right-click in the table to see commands for editing the rows.
- To adjust options for a device, double-click in its row. The Device Properties dialog box opens. This dialog box gives access to many operations and options. See Device Properties Dialog Box.

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.

System Viewer

The Programmer window includes a schematic showing the devices and how they are connected to the host processor. Double-clicking a device opens the Device Properties dialog box, which gives access to many operations and options. See Device Properties Dialog Box.

Output

Displays results from Programmer operations similar to the Output view in the Radiant software.

Tcl Console

Displays the Tcl commands used similar to the Tcl Console view in the Radiant software.

Cable Setup

A dialog box for identifying the cable connection to the target board and setting options for it. See Cable Setup Dialog Box.

Embedded Options

A dialog box for specifying options for an embedded design file before generating it. See Generating Embedded Design Files.

SVF Options

A dialog box for specifying options for an SVF file before generating it. See Generating Tester Serial Vector Format (SVF) Files.

File Formats

Lattice supports the following file formats.

Data File

A data file can be a hex, or bitstream file. Each of these files is based upon an IEEE programming standard:

Bitstream

Data files used for configuring volatile memory (SRAM) of our FPGAs.

Hex

Hexadecimal PROM data files used for Programming into external non-volatile memory, such as parallel or Serial Peripheral Interface (SPI) Flash devices.

See Also

- Programming the FPGA
- Using the Radiant Programmer
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

SPI Flash Support

The Radiant Programmer, combined with a Lattice cable fly-wire, supports the programming of SPI flash devices. All Lattice FPGA devices supported by the Radiant software support SPI Flash.

Each Lattice FPGA device contains Lattice's Reveal IP. Radiant 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 Radiant 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.

Cypress

GigaDevice

ISSI

Macronix

Micron

Spansion

Winbond

Return to Top

Table 30: Cypress

S25FL116K
S25FL132K
S25FL164K
S25FL064L
S25FL128L
S25FL256L
S25FL128S
S25FL256S
S25FL512S
S25FS064S
S25FS128S
S25FS256S
S25FS512S

Return to Top

Table 31: GigaDevice

GD25Q20C
GD25Q40C
GD25Q80C
GD25Q16C
GD25Q32C
GD25Q64C
GD25Q127C
GD25Q256D
GD25LQ20C
GD25LQ40C
GD25LQ80C
GD25LQ16C
GD25LQ32D
GD25LQ64C
GD25LQ128D
GD25LQ256D

Return to Top

Table 32: 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 33: 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 34: 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 35: Spansion

S25FL064A
S25FL040A
S25FL004A
S25FL008A
S25FL016A
S25FL032A
S25FL064A
S25FL001D
S25FL002D
S25FL004D
S25FL032P
S25FL064P
S25FL128P
S25FL132K
S25FL164K
S25FL204K
S25FL208K
S25FL216K

Return to Top

Table 36: 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 Radiant 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.

Adesto

AMIC

Atmel

Eon Silicon

Intel

Macronix

NewFlash

Numonyx

SSTI

STMicro

WindBond.

Table 37: Adesto

AT25SF041

Return to Top of Legacy Device List

Table 38: AMIC

A25L10P
A25L20P
A25L40P
A25L80P
A25L16P

Return to Top of Legacy Device List

Table 39: Atmel

MX25L1005
MX25L2005
MX25L4005(A)
MX25L8005
MX25L1605
MX25L3205
MX25L6405
MX25L12805
MX25U8035

Return to Top of Legacy Device List

Table 40: Eon Silicon

EN25Q32B

Return to Top of Legacy Device List

Table 41: Intel

25F016S33
25F160S33
25F320S33
25F640S33

Return to Top of Legacy Device List

Table 42: Macronix

MX25L1005
MX25L2005
MX25L4005(A)
MX25L8005
MX25L1605
MX25L3205
MX25L6405
MX25L12805
MX25U8035

Return to Top of Legacy Device List

Table 43: NewFlash

NX25P20
NX25P40
NX25P80
NX25P16
NX25P32

Return to Top of Legacy Device List

Table 44: 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 45: SSTI

SST25VF020
SST25VF040
SST25VF080
SST25LF020A
SST25LF040A
SST25LF080A
SST25VF040B
SST25VF080B
SST25VF016B
SST25VF032B
SST25VF064C
SST26VF016B
SST26VF032B
SST26VF064B

Return to Top of Legacy Device List

Table 46: STMicro

M25P20
M25P40
M25P80
M25P16
M25P32
M25P64
M25PE20
M25PE40
M25PE80
M25PE16
M45PE20
M45PE40
M45PE80
M45PE16

Return to Top of Legacy Device List

Table 47: WindBond

W25P20
W25P40
W25P80
W25P16
W25P32
W25X10
W25X20
W25X40
W25X80
W25X16
W25X32
W25X64
W25Q128BV
W25Q64DW

See Also

- Programming the FPGA
- Using the Radiant Programmer
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Using the Radiant Programmer

This section provides procedures for using the Radiant Programmer. Topics include:

- Plugging the Cable into the PC
- Detecting a Cable
- Selecting from Multiple Cables
- Creating a New Radiant Programmer Project
- Opening an Existing Radiant Programmer Project
- Checking the XCF Project
- Scanning the Device
- Selecting from a Matching Device ID List
- Adding a Lattice Device to a Chain

- Adding a Generic JTAG Device to a Chain
- Removing a Device from a Chain
- Editing Device Properties
- Moving a Device Up or Down
- Setting Target Memory, Port Interface, Access Mode, and Operation
- Setting Programming Characteristics
- Downloading Design Files
- Adding a Custom SPI Flash Device
- Editing a Custom SPI Flash Device
- Removing a Custom SPI Flash Device
- Programming Using Custom SPI Flash Device
- Unlocking Secure SPI Flash Support
- Configuring SPI Flash Options
- Changing the Port Assignment
- Testing the Cable Signal
- Viewing the Log File

Plugging the Cable into the PC

You can plug the USB cable into the PC whether the PC is turned on or off. However, make sure that Radiant Programmer is closed and that the target board's power is off before plugging the cable into your PC or unplugging it.

To plug the cable into the PC:

1. Make sure that the Radiant Programmer is closed, and that the target board is powered off.
2. Plug the cable into your PC and the target board.
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 varies, depending on your PC's speed.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Detecting a Cable

You can use the Radiant Programmer Detect Cable feature to determine which cable and port you are using.

To detect a cable:

1. If Cable Setup dialog box is not showing in the Radiant Programmer window, choose **View > Cable Setup**.
2. In the Cable Setup dialog box, click **Detect Cable**.

   The Radiant Programmer detects all available cables connected to your PC, and lists the cable type, port settings, and descriptions in the Output view.

See Also

- Using the Radiant Programmer
- Cable Setup Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Selecting from Multiple Cables

The Radiant 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 Setup dialog box.

To select a USB cable from multiple cables:

- In the Cable Setup dialog box, select the USB cable you want to use from the Port pulldown list.

See Also

- Using the Radiant Programmer
- Cable Setup Dialog Box
- Programming the FPGA

- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Creating a New Radiant Programmer Project

By creating a new Radiant 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.

Pathnames in the .xcf files are written in “cross platform compatible” format by default. This format uses forward-slashes (/), which work with both Linux and Windows. For Windows-only systems you might prefer to use Native Delimiter format, which uses back-slashes (\). To use the Windows Native Delimiter format, open the Programmer initialization file in a text editor:

Users\<*name*>\AppData\Roaming\LatticeSemi\DiamondNG**programmer.ini**

Go to the end of the file and add a line with: **PathDelimiter=1**

Programmer can be started from the Radiant software, the Windows Start menu (integrated or stand-alone version), or a Linux command line.

To create a new Radiant 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. Do one of the following to start Programmer:
   - In the Radiant software, choose **Tools > Programmer**.
   - In the Radiant software, click  in the toolbar.
   - In the Windows Start menu, choose **Lattice Radiant Software > Accessories > Radiant Programmer**.
   - In the Windows Start menu, choose **Lattice Radiant Programmer > Radiant Programmer**.
   - In Linux, from the <*Programmer install path*>/bin/lin64 directory, enter the following on a command line:

     ./programmer
3. In the Getting Started dialog box, enter a project name in the Project Name box.
4. Browse to the location where you wish your project to reside in the Project Location box.
5. Choose one of the following:

- **Create a new project from a 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 pulldown lists.
- **Create a New Blank Project**. This option creates a new blank Radiant Programmer project.

6. Click **OK**.

See Also

- Using the Radiant Programmer
- Getting Started Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Opening an Existing Radiant Programmer Project

To open an existing Radiant 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.

Programmer can be started from the Radiant software, the Windows Start menu (integrated or stand-alone version), or a Linux command line.

To start the Radiant Programmer with an existing project:

1. Do one of the following to start Programmer:
   - In the Radiant software, choose **Tools > Programmer**.
   - In the Radiant software, click  in the toolbar.
   - In the Windows Start menu, choose **Lattice Radiant Software > Accessories > Radiant Programmer**.
   - In the Windows Start menu, choose **Lattice Radiant Programmer > Radiant Programmer**.
   - In Linux, from the <*Programmer install path*>/bin/lin64 directory, enter the following on a command line:

     ./programmer

2. In the Getting Started dialog box, choose **Open an Existing Programmer Project.**
3. Click **OK**.

See Also

- Using the Radiant Programmer
- Getting Started Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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 the Radiant Programmer, choose **Run > Check XCF Project** or click  in the toolbar. The Radiant Programmer checks the XCF project and indicates in the output pane whether or not the project is valid.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Scanning the Device

There are several ways to scan a device. When you create a new project, you can choose to scan your 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 *Lattice Radiant Software 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 the Radiant Programmer.
2. In the Radiant Programmer, choose **Run > Scan Device** or click  in the toolbar. The Radiant Programmer opens a new configuration file, scans the printed circuit board connected to your computer, and lists the devices in the new file.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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 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 drop-down menu.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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 the Radiant Programmer, choose **Edit > Add Device,** click the  button on the toolbar, or right-click and choose **Add Device**.

2. Choose a Lattice device family from the **Device Family** column drop-down menu.
3. Choose a device from the **Device** column drop-down menu.

   The software inserts the new device into the active chain.

To add an iCE40 UltraPlus device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose **Add Device**.
2. Select iCE40 UltraPlus from the **Device Family** column drop-down menu.
3. Select iCE40UP3K or iCE40UP5K from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose **Device Properties**.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add a LAV-AT device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose **Add Device**.
2. Select LAV-AT from the **Device Family** column drop-down menu.
3. Select LAV-AT-E70, LAV-AT-G70, LAV-AT-X70 from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose **Device Properties**.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add an LIFCL device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose **Add Device**.
2. Select LIFCL from the **Device Family** column drop-down menu.
3. Select LIFCL-40 from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose **Device Properties**.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add an LIFCL Engineering Sample device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LIFCL_ENG from the **Device Family** column drop-down menu.
3. Select LIFCL-40-ES from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose **Device Properties**.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add an LFCPNX or UT24CP device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LFCPNX or UT24CP from the **Device Family** column drop-down menu.
3. Select LFCPNX-100 or UT24CP-100 from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open.**

   The software inserts the new device into the active chain.

To add an LFD2NX or UT24C device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LFD2NX or UT24C from the **Device Family** column drop-down menu.
3. Select LFD2NX-40 or UT24C-40 from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add an LFMNX device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LFMNX from the **Device Family** column drop-down menu.
3. Select LFMNX-50 from the **Device** column drop-down menu.

4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add and LFMXO5 device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LFMXO5 from the **Device Family** column drop-down menu.
3. Select LFMXO5-25, LFMXO5-55T, LFMXO5-100T from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add an LFMXO5D device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select LFMXO5D from the **Device Family** column drop-down menu.
3. Select LFMXO5-15D from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add a MachXO3D device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the button on the toolbar, or right-click and choose Add Device.
2. Select MachXO3D from the **Device Family** column drop-down menu.
3. Select a device from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add a MachXO3L device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the  button on the toolbar, or right-click and choose Add Device.
2. Select MachXO3L from the **Device Family** column drop-down menu.
3. Select a device from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the  button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

To add a MachXO3LF device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the  button on the toolbar, or right-click and choose Add Device.
2. Select MachXO3LF from the **Device Family** column drop-down menu.
3. Select a device from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the  button on the toolbar, or right-click and choose Device Properties.
5. In the Device Properties dialog box, click the Browse button and navigate to the bitstream for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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

To add a Generic JTAG device:

1. In the Radiant Programmer, choose **Edit > Add Device,** click the  button on the toolbar or right-click and choose **Add Device**.

2. Select Generic JTAG Device from the **Device Family** column drop-down menu.
3. Select a JTAG-NOP from the **Device** column drop-down menu.
4. Highlight the row, and choose **Edit > Device Properties**, click the  button on the toolbar, or right-click and choose **Device Properties**.
5. In the Device Properties dialog box, at the bottom, enter a value for the instruction register length or click **Load from File** and browse to the .svf file.
6. Click **OK**.

   The software inserts the new device into the active chain.

See Also

- Using the Radiant Programmer
- Device Properties Dialog Box
- Programming the FPGA

Removing a Device from a Chain

To remove a device from the active chain:

1. In the Radiant Programmer, select the device that you want to delete.
2. Choose **Edit > Remove Device**, click the  button on the toolbar, or right-click and chose **Remove Device**.

   Note

   There is no Undo for this command.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Editing Device Properties

To edit device properties in a chain:

1. In the Radiant Programmer, select the device that you want to edit.

2. Choose **Edit > Device Properties**, click the  button 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 Radiant Programmer
- Device Properties Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Moving a Device Up or Down

To move a device up or down in the Radiant 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 the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Setting Target Memory, Port Interface, Access Mode, and Operation

To set a device operation:

1. Select a device.
2. Double-click the appropriate column for the device.
3. In the Device Properties dialog box, select the specific Target Memory, Port Interface, Access Mode, and Operation for the device in the drop-down menu boxes.

See Also

- Using the Radiant Programmer
- Device Properties Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Setting Programming Characteristics

Use the Settings dialog box to specify how the Radiant Programmer processes the configuration. You can also specify other options.

To specify Programming characteristics:

1. In the Radiant Programmer, choose **Edit > Settings**.

   The Settings dialog box opens.
2. Select the options you want, and then click **OK**.

See Also

- Using the Radiant Programmer
- Settings Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Downloading Design Files

After you have specified all of the processing options, use the Program command to download the design files to the devices.

- In the Radiant Programmer, choose **Run > Program Device,** or click  on the toolbar.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer

- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Generating Embedded Design Files

Lattice Embedded VME enables in-field upgrades of Lattice programmable devices by suitable embedded processors. Lattice provides the option to generate several different file formats for different embedded target options.

For iCE40UP, the following embedded solution is supported:

- Slave SPI Embedded: Enables field upgrades via the target SPI port.

For LAV-AT, the following embedded solutions are supported:

- Slave SPI Embedded: Enables field upgrades via the target SPI port.
- JTAG Embedded: Enables field upgrades via the JTAG port.

For LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, and UT24CP, the following embedded solutions are supported:

- Slave SPI Embedded: Enables field upgrades via the target SPI port.
- I2C Embedded: Enables field upgrades via the I2C port.
- JTAG Embedded: Enables field upgrades via the JTAG port.

To generate embedded design files:

1. In the Embedded Options tab, select the desired embedded options. For more information on embedded options, refer to the following tables.
2. Choose **Run > Generate Embedded Code**, or click on the toolbar.
   The files are generated in the specified directory. See the following tables for file types.

Table 48: Slave SPI Embedded—Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
iCE40UP	XCF File (.xcf) Bitstream (.bit/ .hex)	Algorithm VME File (*_algo.iea). This is the algorithm file that specifies which operations will be executed. It must be used with a VME data file. This file is used in device programming or in field upgrading.	Data File (*_data.ied)
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	XCF File (.xcf) Bitstream (.bit/ .rbt)	Algorithm VME File (*_algo.iea). This is the algorithm file that specifies which operations will be executed. It must be used with a VME data file. This file is used in device programming or in field upgrading.	Data File (*_data.ied)

Table 49: Slave SPI Embedded Options—Compress Embedded Files, Convert VME files to HEX (c.) Prom-Based Embedded VME, and Insert Source Code

Device Family	Compress Embedded Files		Convert VME files to HEX (c.) Prom-Based Embedded VME		Include Source Code	
iCE40UP	On	Off	On	Off	On	Off

Table 49: Slave SPI Embedded Options—Compress Embedded Files, Convert VME files to HEX (c.) Prom-Based Embedded VME, and Insert Source Code

Device Family	Compress Embedded Files		Convert VME files to HEX (c.) Prom-Based Embedded VME		Include Source Code	
LAV-AT	On	Off	On	Off	On	Off
LFCPNX						
LFD2NX						
LFMNX						
LFMXO5						
LFMXO5D						
LIFCL						
MachXO3D						
MachXO3L						
MachXO3LF						
UT24C						
UT24CP						
LIFCL ES						

Table 49: Slave SPI Embedded Options—Compress Embedded Files, Convert VME files to HEX (c.) Prom-Based Embedded VME, and Insert Source Code

Device Family	Compress Embedded Files		Convert VME files to HEX (c.) Prom-Based Embedded VME		Include Source Code	
	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.	This operation creates a VME file that is used in file-based embedded programming where the programming of the device is based on the input file. When this option is turned on, it creates a .c file, which is used in EPROM base programming to create a single complied image that is loaded into the CPU. Then the CPU uses this image to program the device.	Do not generate .c file.	Copy contents of sspiembedded/ sourcecode directory to working project directory. There are two directories of the Target SPI Embedded source code: ▶ sspiem ▶ sspiem_eprom **Note**: If the "Convert VME files to HEX (c.) Prom-Based Embedded VME" option is selected, the contents of the sspiem_ecprom directory is copied. If the "Convert VME files to HEX (c.) Prom-Based Embedded VME" option is not selected, the contents of the sspiem directory is copied.	Do not copy.

Table 50: I2C Embedded—Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	XCF File (.xcf) Bitstream (.bit/ .rbt)	Algorithm VME File (*_algo.iea). This is the algorithm file that specifies which operations will be executed. It must be used with a VME data file. This file is used in device programming or in field upgrading.	Data File (*_data.ied)

Table 51: I2C Embedded Options—Compress Embedded Files, Convert VME Files to HEX (c.) Prom-Based Embedded VME, and Insert Source Code

Device Family	Compress Embedded Files		Convert VME files to HEX (c.) Prom-Based Embedded VME		Include Source Code	
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	On	Off	On	Off	On	Off
	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.	This operation creates a VME file that is used in file-based embedded programming where the programming of the device is based on the input file. When this option is turned on, it creates a .c file, which is used in EPROM base programming to create a single complied image that is loaded into the CPU. Then the CPU uses this image to program the device.	Do not generate .c file.	Copy contents of i2cembedded/ sourcecode directory to working project directory. There are two directories of the Target SPI Embedded source code: ▶ i2cem ▶ i2cem_eprom **Note**: If the “Convert VME files to HEX (c.) Prom-Based Embedded VME” option is selected, the contents of the i2cem_ecprom directory is copied. If the “Convert VME files to HEX (c.) Prom-Based Embedded VME” option is not selected, the contents of the i2cem directory is copied.	Do not copy.

Table 52: Full JTAG Embedded—Device Family, Input File 1, Output File 1

Device Family	Input File	Output File 1
LAV-AT	XCF File (.xcf)	Algorithm VME File (*_algo.iea). This is the algorithm file that specifies which operations will be executed. It must be used with a VME data file. This file is used in device programming or in field upgrading.
LFCPNX	Bitstream (.bit/.rbt)	
LFD2NX		
LFMNX		
LFMXO5		
LFMXO5D		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		

Table 53: JTAG Full VME Embedded Options—Compress VME File, Convert VME Files to HEX (c.) File-Based Embedded VME

Device Family	Compress Embedded Files		Convert VME files to HEX (c.) Prom-Based Embedded VME	
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	On	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.	Uncompressed data.		By default this operation is turned off. This creates a VME file that is used in file-based embedded programming where the programming of the device is based on the input file. When this option is turned on, it creates a .c file that is used in EPROM-base programming to create a single complied image that will be loaded into the CPU. Then the CPU will use this image to program the device.

Table 54: JTAG Full VME Embedded Options—Compact VME File, Fixed Pulse Width (Rev. D)

Device Family	Compact VME File		Fixed Pulse Width (Rev. D)	
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**Off (Default)**	**On**	**Off (Default)**	**On**
	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.

Table 55: JTAG Full VME Embedded Options—Verify USERCODE, Program Device If Fails

Device Family	Verify USERCODE, Program Device If Fails	
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**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.

Table 56: JTAG Full VME Embedded Options—Include Header, Include Comment, Maximum Memory Allocation Size Per Row of Data (Kbytes)

Device Family	Include Header		Include Comment		Maxumum Memory Alocation Size Per Row of Data (kB)
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**Off (Default)**	**On**	**Off (Default)**	**On**	
	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.	Omit header from VME file.	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 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.

The following table lists device family, input file types, and output file. extensions for JTAG Slim VME embedded deployment.

Table 57: JTAG Slim VME Embedded Options -- Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	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.

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 58: JTAG Slim VME Embedded Options -- Compressed VME Data File, Convert VME files to HEX (c.) File-Based Embedded VME

Device Family	Compress VME Data File		Convert VME files to HEX (c.) File-Based Embedded VME	
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LFMXO5D LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	On (Default)	Off	Off (Default	On
	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Unchecking 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. Then the CPU will use this image to then program the device.	Generate Hex (.c) file

See Also:

- Lattice Radiant Software User Guide

Generating Tester Serial Vector Format (SVF) Files

Lattice provides the option to generate the Serial Vector Format (SVF) files.

To generate an SVF file:

1. In the SVF Options tab, select the desired options. For more information on embedded options, refer to the following tables.
2. Choose **Run > Generate SVF File**, or click SVF on the toolbar. The files are generated in the specified directory.

Table 59: SVF JTAG Chain—Device Family, Input File 1, and Output File

Device Family	Input File 1	Output File
Lattice Chain	XCF File (.xcf)	SVF File (.svf)

Table 60: SVF JTAG Chain—Options—Operation

Device Family	Operation
Lattice Single Device	Contained in XCF file.

Table 61: SVF JTAG Chain—Options—Write Header and Comments, Write Rev D Standard SVF File

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 FPGAs, Lattice Extended SVF.	Rev. D Standard SVF.

Table 62: SVF JTAG Chain—Options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify

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.

Table 63: SVF JTAG Chain—Options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)

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

Adding a Custom SPI Flash Device

This feature allows you users to add your own SPI Flash device to the Radiant Programmer database.

To create a Custom SPI Flash device:

1. Choose **Tools > Custom Flash Device**.
2. In the Edit Custom Device dialog box, select **SPI Serial Flash Custom**, and click **Add**.
3. In the Custom SPI Flash dialog box, enter:
   - **Device Description**—This can be any alphanumeric string that describes the device (for example: M25P32-VMF6C).
   - **Device Name**—This can be any alphanumeric string that describes the device name (for example: SPI-M25P32).
   - **Package—**This can be any alphanumeric string that describes the package (for example: 16-pin SOIC).
4. In the Device Vendor drop-down menu, choose device vendor.
5. In the Device Density drop-down, choose device density.
6. Device vendor ID—This can be any alphanumeric string that describes the device vendor ID (for example: 0x2C).
7. Device memory ID—This can be any alphanumeric string that describes the device memory capacity ID (for example: 0x15).
8. In the Byte Per Sector drop-down, choose the desired value.
9. If the SPI Flash device supports sector protection, check the **Protection Options On** box.
10. Click **OK**.
11. Once the custom SPI flash has been added, it can be selected in the Device Properties dialog box.

See Also

- Using the Radiant Programmer
- Device Properties Dialog Box
- Custom SPI Flash Dialog Box

- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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.

To edit the properties of a Custom SPI Flash device:

1. Choose **Tools > Custom Flash Device**.
2. In the Edit Custom Device dialog box, 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, enter:
   - **Device Description**—This can be any alphanumeric string that describes the device (for example: M25P32-VMF6C).
   - **Device Name**—This can be any alphanumeric string that describes the device name (for example: SPI-M25P32).
   - **Package**—This can be any alphanumeric string that describes the package (for example: 16-pin SOIC).
4. In the Device Vendor drop-down menu, choose device vendor.
5. In the Device Density drop-down, choose device density.
6. Device vendor ID—This can be any alphanumeric string that describes the device vendor ID (for example: 0x2C).
7. Device memory capacity ID—This can be any alphanumeric string that describes the device vendor memory capacity ID (for example: 0x15).
8. In the Byte Per Sector drop-down, choose the desired value.
9. If the SPI Flash device supports sector protection, check the **Protection Options ON** box.
10. Click **OK**.

See Also

- Using the Radiant Programmer
- Custom SPI Flash Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Removing a Custom SPI Flash Device

Once you have created a custom SPI Flash device, you can remove it from the Custom SPI Flash dialog box.

To edit the properties of a Custom SPI Flash device:

1. Choose **Tools > Custom Flash Device**.
2. In the Edit Custom Device dialog box, select **SPI Serial Flash Custom**.
3. Select the SPI Flash device you want to remove.
4. Click **Remove**.
5. Click **OK**.

See Also

- Using the Radiant Programmer
- Custom SPI Flash Dialog Box
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Programming Using Custom SPI Flash Device

After a custom SPI Flash device has been created, you can program it using the Radiant Programmer.

To program using a Custom SPI Flash device:

1. Left-click on the row corresponding to the device you want to edit, and choose **Edit > Device Properties,** to display the Device Properties dialog box.
2. In the Target Memory Mode drop-down list, choose **External SPI Flash memory (SPI FLASH)**.
3. In the Family drop-down list, choose **SPI Serial Flash Custom**.
4. In the Device drop-down list, choose the name of the custom device you created.
5. Click **OK.**

See Also

- Using the Radiant Programmer
- Device Properties Dialog Box

- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Unlocking Secure SPI Flash Support

If a custom SPI flash device or a SPI flash device was secured and you wish to reprogram the SPI flash, you must first unlock the device before reprogramming it.

To unlock a secure SPI flash:

1. Create a new the Radiant Programmer project as described in Creating a New Radiant Programmer Project or open an existing Radiant Programmer project as described in Opening an Existing Radiant Programmer Project.
2. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
3. In the Operation drop-down list, chose **SPI Flash Unlock Device**.
4. Ensure that the **Secure SPI Flash Golden Pattern Sectors** box is unchecked.
5. Click **OK.**
6. In the Radiant Programmer, choose **Run > Program,** or click  on the toolbar.

The SPI flash is now unlocked and can be reprogrammed.

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Adesto	AT25SF321	No
Adesto	AT25QL641	No
Adesto	AT25SF128	No
Cypress	S25FL116K	Yes
Cypress	S25FL132K	Yes
Cypress	S25FL164K	Yes
Cypress	S25FL064L	No
Cypress	S25FL128L	No
Cypress	S25FL256L	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Cypress	S25FL128S	No
Cypress	S25FL256S	Yes
Cypress	S25FL512S	No
Cypress	S25FS064S	No
Cypress	S25FS128S	No
Cypress	S25FS256S	Yes
Cypress	S25FS512S	No
Cypress	S28HS512T	No
ON Semiconductor	LE25U40CMD	No
GigaDevice	GD25Q20C	No
GigaDevice	GD25Q20E	No
GigaDevice	GD25Q40C	No
GigaDevice	GD25Q40E	No
GigaDevice	GD25Q80C	No
GigaDevice	GD25Q80E	No
GigaDevice	GD25Q16C	Yes
GigaDevice	GD25Q16E	Yes
GigaDevice	GD25Q32C	Yes
GigaDevice	GD25Q32E	Yes
GigaDevice	GD25Q64C	Yes
GigaDevice	GD25Q64E	Yes
GigaDevice	GD25Q127C	Yes
GigaDevice	GD25Q128E	Yes
GigaDevice	GD25Q256D	Yes
GigaDevice	GD25Q256E	Yes
GigaDevice	GD25LQ20C	No
GigaDevice	GD25LQ20E	No
GigaDevice	GD25LQ40C	No
GigaDevice	GD25LQ40E	No
GigaDevice	GD25LQ80C	No
GigaDevice	GD25LQ80E	No
GigaDevice	GD25LQ16C	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
GigaDevice	GD25LQ16E	Yes
GigaDevice	GD25LQ32D	Yes
GigaDevice	GD25LQ32E	Yes
GigaDevice	GD25LQ64C	Yes
GigaDevice	GD25LQ64E	Yes
GigaDevice	GD25LQ128D	Yes
GigaDevice	GD25LB128E	Yes
GigaDevice	GD25LQ128E	Yes
GigaDevice	GD25LQ256D	Yes
ISSI	IS25LP064	Yes
ISSI	IS25LP128	Yes
ISSI	IS25LP032A	Yes
ISSI	IS25LP064A	Yes
ISSI	IS25LP080D	Yes
ISSI	IS25LP016D	Yes
ISSI	IS25LP032D	Yes
ISSI	IS25LP256D	Yes
ISSI	IS25LP128F	Yes
ISSI	IS25LP512M	No
ISSI	IS25LQ020B	No
ISSI	IS25LQ040B	No
ISSI	IS25LQ080B	Yes
ISSI	IS25LQ016B	Yes
ISSI	IS25LQ032B	Yes
ISSI	IS25WP128	Yes
ISSI	IS25WP064A	Yes
ISSI	IS25WP040D	No
ISSI	IS25WP080D	Yes
ISSI	IS25WP016D	Yes
ISSI	IS25WP032D	Yes
ISSI	IS25WP128F	Yes
ISSI	IS25WP256D	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
ISSI	IS25WP512M	No
ISSI	IS25WQ020	No
ISSI	IS25WQ040	No
SPANSION	S25FL064A	No
SPANSION	S25FL040A	No
SPANSION	S25FL004A	No
SPANSION	S25FL008A	No
SPANSION	S25FL016A	No
SPANSION	S25FL032A	No
SPANSION	S25FL064A	No
SPANSION	S25FL001D	No
SPANSION	S25FL002D	No
SPANSION	S25FL004D	No
SPANSION	S25FL032P	No
SPANSION	S25FL064P	No
SPANSION	S25FL128P	No
SPANSION	S25FL132K	Yes
SPANSION	S25FL164K	Yes
SPANSION	S25FL204K	Yes
SPANSION	S25FL208K	Yes
SPANSION	S25FL216K	Yes
Macronix	MX25V1006F	No
Macronix	MX25L2006E	Yes
Macronix	MX25V20066	Yes
Macronix	MX25L4006E	Yes
Macronix	MX25L8006E	Yes
Macronix	MX25L1606E	Yes
Macronix	MX25L3206E	Yes
Macronix	MX25L6406E	Yes
Macronix	MX25U2033E	Yes
Macronix	MX25U4033E	Yes
Macronix	MX25U8033E	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Macronix	MX25L1635E	Yes
Macronix	MX25U3235E	Yes
Macronix	MX25L6435E	Yes
Macronix	MX25U6435E	Yes
Macronix	MX25L12835E	Yes
Macronix	MX25L25635E	Yes
Macronix	MX25L6436E	Yes
Macronix	MX25L12836E	Yes
Macronix	MX25L6445E	Yes
Macronix	MX25L12845E	Yes
Macronix	MX25U1633F	Yes
Macronix	MX25L3233F	Yes
Macronix	MX25L6433F	Yes
Macronix	MX25L12833F	Yes
Macronix	MX25V1035F	No
Macronix	MX25R1035F	No
Macronix	MX25R2035F	Yes
Macronix	MX25V2035F	Yes
Macronix	MX25R4035F	Yes
Macronix	MX25U4035F	Yes
Macronix	MX25V4035F	Yes
Macronix	MX25R8035F	Yes
Macronix	MX25V8035F	Yes
Macronix	MX25R1635F	Yes
Macronix	MX25U1635F	Yes
Macronix	MX25V1635F	Yes
Macronix	MX25R3235F	Yes
Macronix	MX25U3235F	Yes
Macronix	MX25R6435F	Yes
Macronix	MX25U6435F	Yes
Macronix	MX25L12835F	Yes
Macronix	MX25U12835F	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Macronix	MX25L25635F	Yes
Macronix	MX25L12845G	Yes
Macronix	MX25L25645G	Yes
Macronix	MX25U25645G	Yes
Macronix	MX25L51245G	Yes
Macronix	MX25U51245G	No
Macronix	MX66L51235F	No
Macronix	MX66L1G45G	Yes
Macronix	MX66U1G45G	Yes
Macronix	MX35LF1G24AD	Yes
Macronix	MX35LF2G24AD	Yes
Micron	M25P10	No
Micron	M25P20	No
Micron	M25P40	No
Micron	M25P80	No
Micron	M25P16	Yes
Micron	M25P32	Yes
Micron	M25P64	Yes
Micron	M25P128	Yes
Micron	M25PE10	No
Micron	M25PE20	No
Micron	M25PE40	No
Micron	M25PE80	No
Micron	M25PE16	No
Micron	M45PE10	No
Micron	M45PE20	No
Micron	M45PE40	No
Micron	M45PE80	No
Micron	M45PE16	No
Micron	M25PX80	No
Micron	M25PX16	No
Micron	M25PX32	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Micron	M25PX64	Yes
Micron	N25Q016A	No
Micron	N25Q032	Yes
Micron	N25Q032A	No
Micron	N25Q064	Yes
Micron	N25Q128	Yes
Micron	N25Q128A	Yes
Micron	N25Q256	Yes
Micron	N25Q512	No
Micron	MT25QL128	Yes
Micron	MT25QL256	Yes
Micron	MT25QL512	No
Micron	MT25QL01	Yes
Micron	MT25QL02	Yes
Micron	MT25QU128	Yes
Micron	MT25QU256	Yes
Micron	MT35XL256	Yes
Micron	MT35XU256	Yes
Micron	MT25QU512	Yes
Micron	MT35XL512	Yes
Micron	MT35XU512	Yes
Micron	MT25QU01	Yes
Micron	MT35XL01	Yes
Micron	MT35XU01	Yes
Micron	MT25QU02	Yes
Micron	MT35XL02	Yes
Micron	MT35XU02	Yes
WinBond	W25Q10	Yes
WinBond	W25Q20	Yes
WinBond	W25Q80	Yes
WinBond	W25Q16	Yes
WinBond	W25Q32	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
WinBond	W25Q64	Yes
WinBond	W25X10CL	No
WinBond	W25X20CL	Yes
WinBond	W25X40CL	No
WinBond	W25Q20CL	Yes
WinBond	W25Q40CL	No
WinBond	W25Q80DV	No
WinBond	W25Q32FV	Yes
WinBond	W25Q64FV	Yes
WinBond	W25Q128FV	Yes
WinBond	W25Q256FV	Yes
WinBond	W25Q16FW	Yes
WinBond	W25Q32FW	Yes
WinBond	W25Q64FW	Yes
WinBond	W25Q128FW	Yes
WinBond	W25Q16JV	Yes
WinBond	W25Q32JV	Yes
WinBond	W25Q64JV	Yes
WinBond	W25Q128JV	Yes
WinBond	W25Q256JV	Yes
WinBond	W25Q512JV	No
WinBond	W25M512JV	No
WinBond	W25Q16JW	Yes
WinBond	W25Q32JW	Yes
WinBond	W25Q64JW	Yes
WinBond	W25Q128JW	Yes
WinBond	W25Q256JW	Yes
WinBond	W35T51NW	No
WinBond	W35N01JW	No
Adesto	AT25SF041	No
AMIC	A25L10P	No
AMIC	A25L20P	No

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
AMIC	A25L40P	No
AMIC	A25L80P	No
AMIC	A25L16P	No
Eon Silicon	EN25Q32B	No
ATMEL	AT25DF021	No
ATMEL	AT25DF041A	No
ATMEL	AT25DF081	No
ATMEL	AT25DF161	No
ATMEL	AT25DF321	No
ATMEL	AT25DF641	No
ATMEL	AT25F2048	No
ATMEL	AT25F4096	No
ATMEL	AT25FS040N	No
ATMEL	AT26F004	No
ATMEL	AT26DF081A	No
ATMEL	AT26DF161	No
ATMEL	AT26DF161A	No
ATMEL	AT26DF321	No
ATMEL	AT45DB041D	No
ATMEL	AT45DB081D	No
ATMEL	AT45DB161D	No
ATMEL	AT45DB321D	No
ATMEL	AT45DB642D	No
INTEL	25F016S33	Yes
INTEL	25F160S33	Yes
INTEL	25F320S33	Yes
INTEL	25F640S33	Yes
NexFlash	NX25P20	No
NexFlash	NX25P40	No
NexFlash	NX25P80	No
NexFlash	NX25P16	No
NexFlash	NX25P32	No

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
Numonyx	M25P20	No
Numonyx	M25P40	No
Numonyx	M25P80	No
Numonyx	M25P16	Yes
Numonyx	M25P32	Yes
Numonyx	M25P64	Yes
Numonyx	M25P128	Yes
Numonyx	M25PE20	No
Numonyx	M25PE40	No
Numonyx	M25PE80	No
Numonyx	M25PE16	No
Numonyx	M25PX80	No
Numonyx	M25PX16	No
Numonyx	M25PX32	Yes
Numonyx	M25PX64	Yes
Numonyx	M45PE20	No
Numonyx	M45PE40	No
Numonyx	M45PE80	No
Numonyx	M45PE16	No
Numonyx	N25Q032	Yes
Numonyx	N25Q064	Yes
Numonyx	N25Q128	Yes
Numonyx	N25Q128A	Yes
Macronix	MX25L1005	No
Macronix	MX25L2005	No
Macronix	MX25L4005(A)	No
Macronix	MX25L8005	No
Macronix	MX25L1605	Yes
Macronix	MX25L3205	Yes
Macronix	MX25L6405	Yes
Macronix	MX25L12805	Yes
Macronix	MX25U8035	Yes

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
SSTI	SST25VF010A	No
SSTI	SST25VF020	No
SSTI	SST25VF040	No
SSTI	SST25VF080	No
SSTI	SST25LF020A	No
SSTI	SST25LF040A	No
SSTI	SST25LF080A	No
SSTI	SST25VF040B	No
SSTI	SST25VF080B	No
SSTI	SST25VF016B	No
SSTI	SST25VF032B	No
SSTI	SST25VF064C	No
SSTI	SST26VF016B	Yes
SSTI	SST26VF032B	Yes
SSTI	SST26VF064B	Yes
STMicro	M25P20	No
STMicro	M25P40	No
STMicro	M25P80	No
STMicro	M25P16	No
STMicro	M25P32	No
STMicro	M25P64	Yes
STMicro	M25PE20	No
STMicro	M25PE40	No
STMicro	M25PE80	No
STMicro	M25PE16	No
STMicro	M45PE20	No
STMicro	M45PE40	No
STMicro	M45PE80	No
STMicro	M45PE16	No
WinBond	W25P10	No
WinBond	W25P20	No
WinBond	W25P40	No

Table 64: SPI Flash Devices

SPI Device	SPI Package	Unlock Supported?
WinBond	W25P80	No
WinBond	W25P16	No
WinBond	W25P32	No
WinBond	W25X10	No
WinBond	W25X20	No
WinBond	W25X40	No
WinBond	W25X80	No
WinBond	W25X16	No
WinBond	W25X32	No
WinBond	W25X64	No
WinBond	W25Q128BV	No
WinBond	W25Q64DW	Yes

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Configuring SPI Flash Options

You can use the Device Properties dialog box to configure SPI Flash options for Ultra Plus 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.
3. In the Target Memory pulldown menu, choose **External SPI Flash Memory (SPI FLASH)**.
4. In the Port Interface pulldown menu, choose **SPI** for iCE40UP, MachXO3L, MachXO3LF or **JTAG2SPI** or **SSPI2SPI** for LAV-AT, LFCPNX, LFD2NX, LFMNX, LIFCL, MachXO3D, UT24C or UT24CP.

5. In the Access Mode pulldown menu, choose **Direct Programming**.
6. In the Options pulldown menu, chose the desired SPI Flash option.
7. In the Programming File box, browse to and select the programming file (.bit, .rbt, .bin, .hex, .mcs, .exo, or .xtek).
8. Select the desired **Family**, **Vendor**, **Device**, and **Package** options.
9. Click **Load from File** to select the data size entered. 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.
10. Select start address and end address from the **Start Address (Hex)** and **End Address (Hex)** drop-down menus. For the AMD parallel flash, the starting address is automatically selected and cannot be changed.
11. To erase the flash device if a verify error occurs, click, the **Erase SPI Part on Programming Error** box.
12. Click **OK.**

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Device Properties Dialog Box
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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:

1. In Programmer, choose **View > Cable Setup**.
2. In the Cable Setup dialog box, 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 the Radiant Programmer
- Programming the FPGA
- Cable Setup Dialog Box

- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

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 Programmer, choose **View > Cable Setup**.
2. In the Cable Setup dialog box, click the **Debug Mode** button.

   The Cable Signal Tests dialog box opens.
3. In the dialog box, click **Power Check** to test the connection.

   The VCC Status LED blinks green if the cable and power are detected. It produces an error message and blinks red if the cable or power is not detected.
4. In the dialog box, select an option for the pins you want to test. The options are defined as follows:
   - **Toggle** – Alternates between logic 1 and logic 0.
   - **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 options available for cable signal testing varies, depending on the type of cable you are using and the options you have selected in the Cable Setup dialog box. Options not available are grayed out.
5. 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.
6. To perform continuous testing, click **Loop Test**.

   This causes the pins that have a Toggle setting to toggle continuously until you stop the process.
7. To stop the loop process, press **ESC**.

See Also

- Using the Radiant Programmer
- Cable Signal Tests Dialog Box

- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Viewing the Log File

An ASCII text log file summarizes the results of the Programmer operations.

To view the log file:

- Click the LOG button on the toolbar. An ASCII text file opens showing the contents of the log.

See Also

- Using the Radiant Programmer
- Programming the FPGA
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Using Programmer Tcl Commands

This section provides the Programmer extended Tcl command syntax, command options, and usage examples. These commands are only supported in standalone Programmer.

Programmer Tcl Command Descriptions

The following table provides a listing of all valid Programmer Tcl command options and describes option functionality.

Table 65: Programmer Tcl Commands

Command	Function (Argument)	Description
pgr_project	**pgr_project open** <XCF file>	The open command will open the specified project file in-memory.
	pgr_project save [<XCF file>]	Writes the current project to the specified path. If there is no file path specified then it will overwrite the original file.
	pgr_project 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.
	pgr_project help	Displays help for the pgr_project command.
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 Radiant software interface. These commands can override connection options that are set in user defaults.
	pgr_program set -cable <LATTICE | USB | USB2>	Sets the cable for downloading.
	pgr_program set -portaddress <0x0378|0x0278|0x03bc|0x0378|0x0278|0x03bc|0x<custom address>> <EzUSB-0|EzUSB-1|EzUSB-2|...|EzUSB-15> <FTUSB-0|FTUSB-1|FTUSB-2|...|FTUSB-15>	Sets the port address for the downloading.
	pgr_program run	Executes the current xcf with the current settings. Note that there may be warnings that are displayed in the Tcl Console window. These warnings will be ignored and processing will continue.
	pgr_program help	Displays help for pgr_program command.

Table 65: Programmer Tcl Commands

Command	Function (Argument)	Description
pgr_genfile	<no_argument>	Programmer generate files command (not supported for customer use)
	pgr_genfile set -process <svf | vme12>	Sets file type for file generation.
	pgr_genfile set -outfile <file path>	Sets the output file.
	pgr_genfile run	Generates file based on the current xcf and current settings.
	pgr_genfile help	Displays help for pgr_genfile command.

This section displays the usage information of the commands:

pgr_project

Commands to open, save or close XCF file.

Syntax

```
pgr_project open <XCF file>|save <XCF file>|close
```

Arguments

- **open <XCF file>:** This option specify XCF file name and open it.
- **save <XCF file>:** This option specify XCF file name and save current XCF project.
- **close:** Closes current XCF project.

Example

```
pgr_project open test.xcf
pgr_project save test.xcf
pgr_project close
```

See Also

- "Using Programmer Tcl Commands" on page 478

pgr_program

Commands to Save current design to disk.

Syntax

```
pgr_program set [-cable LATTICE|USB|USB2] [-portaddress
<portaddress>]|run
```

Arguments

- **set:** No arguments return the status of the current settings.

- **-cable LATTICE|USB|USB2:** (Optional) Sets the cable for downloading.
- **-portaddress <portaddress>:** (Optional) Sets the port address for downloading. Where port address for cable type can be:

  1. LATTICE --- 0x0378|0x0278|0x03bc|0x0378|0x0278|
  0x03bc|0x<custom address>
  2. USB --- EzUSB-0|EzUSB-1|EzUSB-2|...|EzUSB-15
  3. USB2 --- FTUSB-0|FTUSB-1|FTUSB-2|...|FTUSB-15
- **run:** Executes the current XCF with the current settings.

Example

```
pgr_program set -cable LATTICE
pgr_program set -portaddress 0x0378
pgr_program run
```

See Also

- "Using Programmer Tcl Commands" on page 478

pgr_genfile

Commands to generate files, not supported for customer use.

Syntax

```
pgr_genfile set [-process <file type>] [-outfile <file
    path>]|run
```

Arguments

- **set:** No arguments return the status of the current settings.
- **-process <file type>:** (Optional) Sets the file type for file generation.

  where file type for process can be:

  file type --- svf|vme12
- **-outfile <file path>:** (Optional) Sets the output file.
- **run:** Generates file based on the current XCF and current settings.

Example

```
pgr_genfile set -process svf
pgr_genfile set -outfile test
pgr_genfile run
```

See Also

- "Using Programmer Tcl Commands" on page 478

Programmer Options

This section lists the options available in Programmer.

Topics include:

- Getting Started Dialog Box
- Device Properties Dialog Box
- Custom SPI Flash Dialog Box
- Settings Dialog Box
- Cable Setup Dialog Box
- Cable Signal Tests Dialog Box

Getting Started Dialog Box

The following options are available in the Getting Started dialog box:

Project Name

Specifies the name of the project.

Project Location

Specifies the location of the project.

Create a New Project from a 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 where the download cable is connected.

Blank Programmer 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
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Device Properties Dialog Box

The Device Properties dialog box consists of two tabs: General Tab and Device Information Tab.

General Tab

The following options are available in the Device Properties dialog box General tab, depending on selected Access Mode and Operation.

Target Memory Target memory is the storage memory type on the FGPA.

Table 66: Target Memory for iCE40UP

Target Memory	Description
Compressed Random Access Memory (CRAM)	The memory type used for CRAM configuration.
Non Volatile Configuration Memory (NVCM)	This memory type is used to program the non-volatile configuration memory (NVCM) while the device is not operational.
External SPI Flash Memory (SPI FLASH)	Serial peripheral interface (SPI) memory flash memory.

Table 67: Target Memory for LAV-AT, LFCPNX, LFD2NX, LIFCL, UT24C or UT24CP

Target Memory	Description
Static Random Access Memory (SRAM)	The memory type used for SRAM configuration.
Non Volatile Configuration Memory (EFUSE)	This memory type is used to program the non-volatile configuration memory (EFUSE).
External SPI Flash Memory (SPI FLASH)	Serial peripheral interface (SPI) memory flash memory.

Table 68: Target Memory for LFMNX, MachXO3D, MachXO3L, or MachXO3LF

Target Memory	Description
Static Random Access Memory (SRAM)	The memory type used for SRAM configuration.
FLASH Configuration Memory	
Non Volatile Configuration Memory (EFUSE)	This memory type is used to program the non-volatile configuration memory (EFUSE).
External SPI Flash Memory (SPI FLASH)	Serial peripheral interface (SPI) memory flash memory.

Table 69: Target Memory for LFMXO5LFMXO5D

Target Memory	Description
Static Random Access Memory (SRAM)	The memory type used for SRAM configuration.
Non Volatile Configuration Memory (EFUSE)	This memory type is used to program the non-volatile configuration memory (EFUSE).
Internal Flash Memory (FLASH)	Flash memory.

Port Interface The following tables list the port interfaces...

Table 70: Port Interface for iCE40UP

Port Interface	Description
Slave SPI	The four-wire serial peripheral interface (SPI) communications protocol.
SPI	Serial peripheral interface (SPI) communications protocol.

Table 71: Port Interface for LAV-AT

Port Interface	Description
JTAG	The four-wire JTAG interface communications protocol.
Slave SPI	The four-wire serial peripheral interface (SPI) communications protocol.
SPI	Serial peripheral interface (SPI) communications protocol.

Table 72: Port Interface for LFCPNX, LFD2NX, LFMXO5, LFMXO5D, LIFCL, UT24C or UT24CP

Port Interface	Description
JTAG	The four-wire JTAG interface communications protocol.
Slave SPI	The four-wire serial peripheral interface (SPI) communications protocol.
I2C	The two-wire Integrated Circuit (I2C communications protocol.
SPI	Serial peripheral interface (SPI) communications protocol.
I3C Bridge	For Lattice Nexus Platform devices, the I3C Configuration Cable Bridge provides an interface with bridging logic to covert HW-USBN-2B data traffic into I3C bus format, Radiant Programmer 3.1X (and later) allows you to access Lattice FPGA configuration logic via the I3C configuration interface.

Table 73: Port Interface for LFMNX, MachXO3D, MachXO3L, MachXO3LF

Port Interface	Description
JTAG	The four-wire JTAG interface communications protocol.
JTAG2SPI	The JTAG to SSPI bridge.

Access Mode Selects the mode for programming the device.

Table 74: Access Mode for iCE40UP

Access Mode	Description
Direct Programming	Used for direct programming while the device is not in operation.

Table 75: Access Mode for LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP

Access Mode	Description
Feature Rows Programming	Used to program, read back, and edit the Feature Row sector of the LAV-AT, LFCPNX, LFD2NX, LFMXO5, LFMXO5D, LIFCL, UT24C or UT24CP device. This mode also is used to read back and edit the non-volatile Control Register 1 (which also belongs to Feature Row sector).
Direct Programming	Used for direct programming while the device is not in User mode.

Table 75: Access Mode for LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP

Access Mode	Description
Background Programming	Used for background programming while the device is in User mode.
Advanced Security Programming	Used for programming the security keys.

Operation Lists the available operation modes for the device. Choose one from the pull-down list.

Table 76: Operations for iCE40UP

Operation	Description
Blank Check	Checks if the device is erased (blank).
CRAM Read and Save	Shifts in a bitstream by directly loading the iCE40 CRAM over the SPI bus. The contents in memory are then read and saved to an output file.
Erase, Program, Verify	Erases, programs, and verifies the new configuration memory.
Fast Configuration	Shifts in a bitstream directly into device and the device takes care of the rest for configuration.
Fast Program, Read and Save	Shifts in a bitstream directly into device and the device takes care of the rest for configuration. The contents in memory are then read and saved to an output file.
Fast Program, Read and Save without DONE bit	Shifts in a bitstream directly into device and the device takes care of the rest for configuration, but without the DONE bit. The contents in memory are then read and saved to an output file.
Program Only	Programs a new configuration into memory. Issues programming instructions and then the data is taken from the data file.
Program, Verify	Programs the memory of the device and then verifies the contents of that memory.
Program, Verify, Secure	Programs the memory of a device, verifies the contents of memory, secures the configuration block. Secure inhibits readback of the device.
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.
Secure Device	Secures the configuration block. Secure inhibits readback of the device.
Verify ID	Verifies the ID of the device.
Verify Only	Verifies the new configuration memory.

Table 77: Operations for LAV-AT

Target Memory	Ports Interface	Access Mode	Operations
Static Random Access Memory (SRAM)	JTAG Slave SPI	Direct Programming	Fast Configuration Erase Only Verify ID Display ID Program Control Register0 Display Control Register0 Program Control Register1 Display Control Register1 Display USERCODE Read Status Register Refresh Bypass
External SPI flash Memory (SPI FLASH) Programming	SPI	Direct Programming	Erase, Program, Verify Erase, Program Verify Only Verify ID Erase All Calculate File Size Checksum Calculate Device Size Checksum Read and Save Erase, Program, Verify Quad 1 Scan SPI Flash Device

Table 78: Operations for LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP

Target Memory	Ports Interface	Access Mode	Operations
Static Random Access Memory (SRAM)	JTAG Slave SPI, I2C, I3C	Direct Programming	Fast Configuration
			Erase, Program, Verify
			Verify Only
			Erase Only
			Verify ID
			Display ID
			Program Control Register0
			Display Control Register0
			Program Control Register1
			Display Control Register1
			Display USERCODE
			Read and Save
			Read Status Register
			Refresh
			Secure Plus
			Bypass
			External Primary Dry Run
			External Golden Dry Run
		Background Programming	SEI Fast Program
			Verify Only
			Verify ID
			Display ID
			Display Control Register0
			Display Control Register1
			Display USERCODE
			Read and Save
			Read Status Register
			Refresh
			External Primary Dry Run
			External Golden Dry Run
			Bypass
Non Volatile Configuration Memory	JTAG Slave SPI, I2C	Feature Rows Programming	Program Feature Row
			Update Feature Row
			Read Feature Row
			Program Control NV Register1
			Display Control NV Register1

Table 78: Operations for LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP

Target Memory	Ports Interface	Access Mode	Operations
		Advanced Security Keys Programming	Program Password Key
			Read Password Key
			Program Encryption Key
			Read Encryption Key
			Program Public Key
			Read Public Key
			Program TraceID
			Display TraceID
			Program Locks Policies
			Read Locks Policies
			Update Locks Policies
			Lock Ports Interface
			Enable HMAC Authentication Mode
			Enable ECDSA Authentication Mode
External SPI flash Memory (SPI FLASH) Programming Note: LFMXO5, LFMXO5D does not support this mode	JTAG2SPI, SSPI2SPI	Direct Programming	Erase, Program, Verify
			Erase, Program
			Verify Only
			Verify ID
			Erase All
			Calculate File Size Checksum
			Calculate Device Size Checksum
			Read and Save
			Erase, Program, Verify Quad 1
			Scan SPI Flash Device
			Fail Safe Upgrade

Table 79: Flash Operations (LFMXO5, LFMXO5D only)

Target Memory	Ports Interface	Access Mode	Operations
Flash Configuration Memory	JTAG Slave SPI	Direct Programming	Erase, Program, Verify
			Erase, Program, Verify, Refresh
			Program Only
			Verify Only
			Verify ID
			Erase All
			Blank Check
			Read and Save
			Bypass
		Background Programming	Erase, Program, Verify
			Erase, Program, Verify, Refresh
			Program Only
			Verify Only
			Verify ID
			Erase All
			Blank Check
			Read and Save
			Bypass

The following is an alphabetic list of all available operations with a definition of each operation.

Table 80: Operations

Operation	Description
Blank Check	Check if entire configuration memory is blank,
Bypass	No operation is performed on the device.
Display Control NV Register1	Reads and displays the content of non-volatile control register 1, which is in the same location as the Feature Row.
Display Control Register0	Reads and displays the contents of Control Register 0.
Display Control Register1	Reads and displays the contents of the SRAM Control Register 1.
Display ID	Reads and displays the device ID.
Display Status Register	Reads and displays the device's Status Register.
Display USERCODE	Reads and displays the USERCODE.
Erase All	Erases all configuration memory.

Table 80: Operations (Continued)

Operation	Description
Erase Feature Row	Erases the feature row.
Erase Only	Erases the configuration memory.
Erase, Program, Verify	Erases, programs, and verifies the new configuration memory. In SRAM mode, this operation only supported to program and verify the CFG data array, All other instructions or data related to the LMNI INIT Bus will be ignored.
Erase, Program, Verify, Refresh	Erases, programs, and verifies the new configuration memory; and at the end perform a refresh.
External Primary Dry Run	Execute Dry Run on Primary image off External SPI Flash.
External Golden Dry Run	Execute Dry Run on Golden image off External SPI Flash.
Enable HMAC Authentication Mode	Program HMAC Authentication Mode Enable Bit.
Enable ECDSA Authentication Mode	Program ECDSA Authentication Mode Enable Bit.
Fast Configuration	Shifts in a bitstream directly into device and the device take care of the rest for configuration. **Note**: For LIFCL ES devices, power-cycling is required if an error occurs when using SRAM “Fast Configuration” operation.
Program/Program Only	Programs a new configuration into memory. Issues programming instructions and then the data are taken from the data file.
Program Control NV Register1	Programs the content of non-volatile control register 1 which is in the same location of the Feature Row.
Program Control Register0	Read Back and update the Control Register 0.
Program Control Register1	Read Back and update the SRAM Control Register 1.
Program Encryption Key	Programs the encryption key into a device.
Program Feature Row	Programs the Feature Row. Programs the 128-bit feature row data into a device. The setting is from the .fea file.
Program Password Key	Programs the password keys into the device.
Program Public Key	Programs the ECDSA public keys into the device.
Program Locks Policies	Program the device’s Locks Policies.
Lock Ports Interface	Program the Ports interface locks.

Table 80: Operations (Continued)

Operation	Description
Read and Save	Reads the contents of memory and saves it to an output file. In SRAM mode, this operation only supported to read and save the CFG data array, All other instructions or data related to the LMNI INIT Bus will be ignored.
Read Encryption Key	Reads the contents of encryption key.
Read Feature Row	Reads the contents of the Feature Row.
Read Password Key	Reads the contents of Password Key.
Read Public Key	Reads the contents of ECDSA Public Key.
Read Locks Policies	Reads the contents of device's Locks Policies.
Read Status Register	Reads the contents of the Status Register.
Refresh	Configures the pattern in non-volatile memory into SRAM.
Secure Plus	Secures the configuration block and the user's fuses memory. Secure inhibits readback of the device.
SEI Fast Program	Fast Program the SEI (Soft Error Injection) bitstream to support SED features.
Scan SPI Flash Device	Scans the SPI Flash in the board.
SPI Flash Calculate Device Size Checksum	Calculates the Checksum of the entire device.
SPI Flash Calculate File Size Checksum	Calculates the Checksum of the data contained in the device for the address range of the input file.
SPI Flash Erase All	Erases all of the configuration memory.
SPI Flash Erase, Program, Verify	Erases, programs, and verifies the new configuration memory.
SPI Flash Erase, Program, Verify Quad 1	Erases, programs, and verifies the new configuration memory; and enable the SPI-Flash Quad read mode.
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.
Upload to Static RAM	Uploads data to SRAM.
Update Feature Row	Read back and update the feature rows in the device
Update Locks Policies	Update the Locks Policies.
Verify ID	Verifies the ID of the device.

Table 80: Operations (Continued)

Operation	Description
Verify Only	Verifies the new configuration memory. In SRAM mode, this operation only supported to verify the CFG data array, All other instructions or data related to the LMNI INIT Bus will be ignored.
Verify USERCODE	Verifies the USERCODE of the device.

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
- Programming Pins
- JTAG IDCODE
- JTAG IDCODE MASK
- IDCODE Length
- SOFT IDCODE
- Maximum Erase Pulse Width (ms)
- Maximum Program Pulse Width (ms)

See Also

- Programmer Options
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Custom SPI Flash Dialog Box

The following options are available in the Custom SPI Flash dialog box:

Device Family

Lists the device family being used.

Device Description

The name that was assigned to the device.

Device Name

This can be any alphanumeric string that describes the device name (for example: SPI-M25P32).

Package

This can be any alphanumeric string that describes the package (for example: 16-pin SOIC).

Device Vendor

The vendor of the custom SPI Flash device.

Device Density

The size of the Serial SPI Flash device (512 kilobits - 512 MB).

Manufacturing ID

The manufacturing ID of the vendor of the custom SPI Flash device. (for example: 0xC2 Macronix).

Device Memory Capacity ID

This can be any alphanumeric string that describes the device ID (for example: 0x15).

Byte Per Sector

Allows you to choose desired bytes per sector value.

Protection Options ON If the SPI Flash device supports sector protection, select this option to enable protection options. This allows you to select **Secure SPI Flash Golden Pattern Sectors** in the Device Properties Dialog Box.

See Also

- Programmer Options
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Settings Dialog Box

The following options are available in the Settings dialog box:

General Tab

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 pull-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 the log file each time the application starts. If the option is not selected, the log file continues increase in size until the file is manually erased.

Programming Tab

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
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Cable Setup Dialog Box

The following options are available in the Cable Setup dialog box:

Detect Cable Automatically detects where 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

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 TCK Divider Setting option.

TCK Divider Setting (0 - 30x)

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 81: 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 82: 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 83: 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

Table 83: USB-2A and Parallel port (Continued)

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 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.

Debug Mode

Opens the Cable Signal Tests dialog box. See Cable Signal Tests Dialog Box.

See Also

- Programmer Options
- Programming and Configuring iCE40 Devices with Programmer
- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Cable Signal Tests Dialog Box

The following options are available in the Cable Signal Tests dialog box:

Toggle

Alternates between logic 1 and logic 0.

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
- Programming and Configuring iCE40 Devices with Programmer

- Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

Programming and Configuring iCE40 Devices with Programmer

The Radiant Programmer supports programming and configuration of iCE40 devices.

There are three basic flows for programming or configuring iCE40 devices:

- iCE40 CRAM Configuration Flow using .bin and .hex files output from the Radiant software. The default is .bin.

  CRAM configuration is accomplished by directly loading the iCE40 CRAM over the SPI bus.
- iCE40 NVCM Programming Flow using Nonvolatile Configuration Memory (.nvcm) files output from the Radiant software.

  NCVM programming involves transmitting programming data over the SPI bus to the NVCM array internal to the iCE40 device. The NVCM is one-time programmable (OTP).
- iCE40 SPI Flash Programming Flow using a separate SPI Flash device to configure an iCE40 device.

  In this flow, the iCE40 device acts as the SPI bus master and therefore controls the data flow from the configuration device.

See Also

- Configuring an iCE40 Device Using the CRAM Flow
- Programming an iCE40 Device Using the NVCM Flow
- Programming an iCE40 Device Using SPI Flash Device Flow
- Using the Radiant Programmer
- Lattice Radiant Software User Guide
- TN1248 - iCE40 Programming and Configuration

Configuring an iCE40 Device Using the CRAM Flow

iCE40 CRAM Configuration Flow uses .hex files, .bin files, or Intel-Hex files output from the Radiant software. CRAM configuration is accomplished by directly loading the iCE40 CRAM over the SPI bus.

When using the Radiant Programmer to configure an iCE40 device, you must first create a new Programmer project, or open an existing Programmer project. By creating a new Programmer project, you create a chain file (.xcf).

To create a new Programmer project:

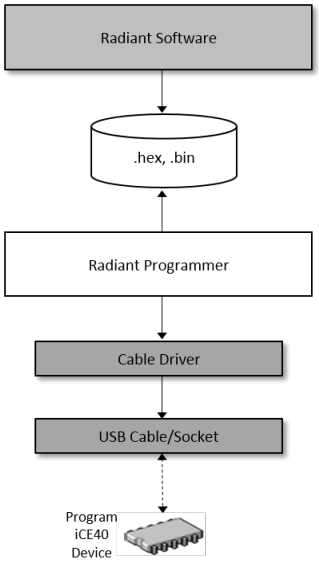
1. Make sure that the board is turned on and that the Lattice USB programming cable is properly connected to the computer.
2. Issue the start command.
   - In Windows, go to the Windows Start menu and choose **All Programs > Lattice Radiant Software> Accessories > Radiant Programmer.**
   - In Linux, from the *<install_path>/programmer/bin/lin64* directory, enter the following on a command line:

     ```
     ./programmer
     ```
3. In the Getting Started dialog box, enter a project name in the **Project Name** box.
4. Browse to the location where you wish your project to reside in the **Project Location** box.
5. Choose one of the following:
   - **Create a new project from a 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** pulldown lists.
   - **Create a New Blank Project** - This option creates a new blank Programmer project.
   - Click **OK**.
6. In Programmer:
   - Select the iCE40 family in the Device Family drop-down menu.
   - Select the desired iCE40 device in the Device drop-down menu.
7. Choose **Edit > Device Properties**, or click the  button on the toolbar, or right-click and choose **Device Properties**.
8. In the Device Properties dialog box:
   - In the Target Memory pulldown menu, choose **Compressed Random Access Memory (CRAM)**.
   - In the Port Interface pulldown menu, choose **Slave SPI**.
   - In the Access Mode pulldown menu, choose **Direct Programming**.
   - In the Operation pulldown menu, choose the desired operation, as described in the Device Properties dialog box.
9. In the Programming File box, browse to the Radiant software-generated programming file (.hex or .bin).
10. Click **OK** to close the Device Properties dialog box.

After you have specified all of the processing options, use the Program command to download the design file to the iCE40 device.

- In Programmer, choose **Run > Program,** or click  on the toolbar.

The diagram below shows the iCE40 CRAM configuration flow



See Also

- Programming and Configuring iCE40 Devices with Programmer
- Getting Started Dialog Box
- Device Properties Dialog Box
- Using the Radiant Programmer
- TN1248 - iCE40 Programming and Configuration
- Lattice Radiant Software User Guide

Programming an iCE40 Device Using the NVCM Flow

iCE40 NVCM Programming Flow uses Nonvolatile Configuration Memory (.nvcm) files output from the Radiant software.

When using the Radiant Programmer to configure an iCE40 device, you must first create a new Programmer project, or open an existing Programmer project. By creating a new Programmer project, you create a chain file (.xcf).

To create a new Programmer project:

1. Make sure that the board is turned on and that the Lattice USB programming cable is properly connected to the computer.
2. Issue the start command.

- In Windows, go to the Windows Start menu and choose **All Programs > Lattice Radiant Software > Accessories > Radiant Programmer.**
- In Linux, from the *<install_path>/programmer/bin/lin64* directory, enter the following on a command line:

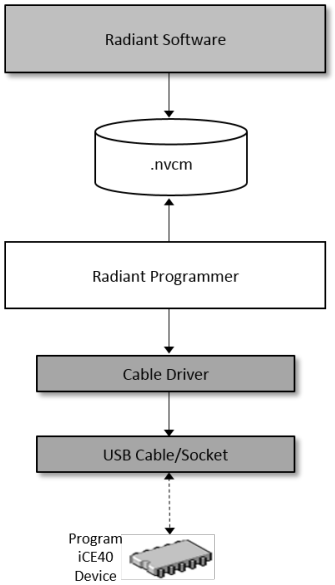
  ```
  ./programmer
  ```

3. In the Getting Started dialog box, enter a project name in the **Project Name** box.
4. Browse to the location where you wish your project to reside in the **Project Location** box.
5. Choose one of the following:
   - **Create a new project from a 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** pulldown lists.
   - **Create a New Blank Project** - This option creates a new blank Radiant Programmer project.
   - Click **OK**.
6. In Programmer:
   - Select the iCE40 family in the Device Family drop-down menu.
   - Select the desired iCE40 device in the Device drop-down menu.
7. Choose **Edit > Device Properties**, or click the  button on the toolbar, or right-click and choose **Device Properties**.
8. In the Device Properties dialog box:
   - In the Target Memory pulldown menu, choose **Non Volatile Configuration Memory (NVCM)**.
   - In the Port Interface pulldown menu, choose **Slave SPI**.
   - In the Access Mode pulldown menu, choose **Direct Programming**.
   - In the Operation pulldown menu, choose the desired operation, as described in the Device Properties dialog box.
9. In the Programming File box, browse to the Radiant software-generated NVCM programming file (.nvcm).
10. Click **OK** to close the Device Properties dialog box.

After you have specified all of the processing options, use the Program command to download the design file to the iCE40 device.

In Programmer, choose **Run > Program,** or click  on the toolbar.

The diagram below shows the iCE40 NVCM programming flow



See Also

- Programming and Configuring iCE40 Devices with Programmer
- Getting Started Dialog Box
- Device Properties Dialog Box
- Using the Radiant Programmer
- Lattice Radiant Software User Guide
- TN1248 - iCE40 Programming and Configuration

Programming an iCE40 Device Using SPI Flash Device Flow

iCE40 SPI Flash Programming Flow involves using a separate SPI Flash device to configure an iCE40 device.

When using the Radiant Programmer to configure an iCE40 device, you must first create a new Programmer project, or open an existing Programmer project. By creating a new Programmer project, you create a chain file (.xcf).

To create a new Programmer project:

1. Make sure that the board is turned on and that the Lattice USB programming cable is properly connected to the computer.
2. Issue the start command.
   - In Windows, go to the Windows Start menu and choose All **Programs > Lattice Radiant Software > Accessories > Radiant Programmer.**

- In Linux, from the *<install_path>/programmer/bin/lin64* directory, enter the following on a command line:

  ```
  ./programmer
  ```

3. In the Getting Started dialog box, enter a project name in the **Project Name** box.
4. Browse to the location where you wish your project to reside in the **Project Location** box.
5. Choose one of the following:
   - **Create a new project from a 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** pulldown lists.
   - **Create a New Blank Project** - This option creates a new blank Radiant Programmer project.

   Note

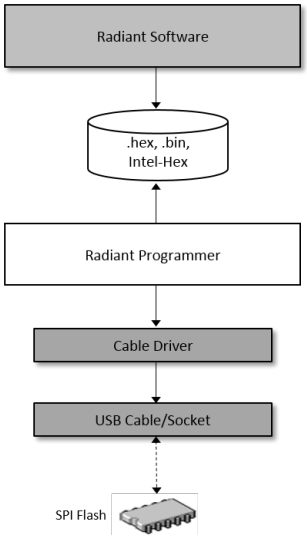
   Because iCE40 devices are Target SPI, and not JTAG, Programmer does not support the "**Create a New Project from a Scan**" option for iCE40 devices.

6. Click **OK**.
7. In Programmer:
   - Select the iCE40 family in the Device Family drop-down menu.
   - Select the desired iCE40 device in the Device drop-down menu.
8. Choose **Edit > Device Properties**, or click the  button on the toolbar, or right-click and choose **Device Properties**.
9. In the Device Properties dialog box:
   - In the Target Memory pulldown menu, choose **External SPI Flash Memory (SPI FLASH)**.
   - In the Port Interface pulldown menu, choose **SPI**.
   - In the Access Mode pulldown menu, choose **Direct Programming**.
   - In the Operation pulldown menu, choose the desired operation, as described in Device Properties dialog box.
10. In the Programming File box, browse to the Radiant software-generated programming file (.hex or .bin).
11. In the Advanced Flash Options box, select **Family**, **Vendor**, **Device**, and **Package**.
12. Click **Load from 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.

13. Select start address and base address from **Start Address (Hex)** and **Base Address (Hex)** drop-down menus.
14. If you want to erase the flash device if a verify error occurs, click, the **Erase SPI Part on Programming Error** box.
15. Click **OK**.

After you have specified all of the processing options, use the Program command to download the design file to the SPI Flash device.

In Programmer, choose **Run > Program,** or click on the toolbar. The diagram below shows the iCE40 SPI Flash programming flow.



See Also

- Programming and Configuring iCE40 Devices with Programmer
- Adding a Custom SPI Flash Device
- Editing a Custom SPI Flash Device
- Removing a Custom SPI Flash Device
- Programming Using Custom SPI Flash Device
- Getting Started Dialog Box
- Device Properties Dialog Box
- Using the Radiant Programmer
- Lattice Radiant Software User Guide
- TN1248 - iCE40 Programming and Configuration

Programming and Configuring LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL,

MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Devices with Programmer

The Radiant Programmer supports programming and configuration of LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP devices.

This section describes:

- Reading the Feature Row (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Only)
- Programming the Feature Row (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)
- Programming the Password Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)
- Programming the Encryption Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)
- Programming the ECDSA Public Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)
- Executing LFMNX, LFMXO5, LFMXO5D, MachXO3D, MachXO3L, MachXO3LF Dry Run Features
- Setting Lock Policies (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP)
- Setting External Configuration Ports Interfaces Locking Features (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP)

The following tables describe the feature row that contains user-specified information necessary for device operations. For example, you can specify the device's I2C slave address without using the default address from the factory.

Table 84: Feature Row Definition - Feature Bit

Bit	Name	Note
1:0	Authentication	00 - No Auth
2	Decompression Enable	
4:3	Decryption Enable	00 - OFF 01 - Undefined 10 - ON 11 - OFF

Table 84: Feature Row Definition - Feature Bit

Bit	Name	Note
5	CID_EN	
8:6	Boot Select	
9	PWD_EN	
10	PWD_ALL	
11	I3C_ONLY	I3C Only / I2C Disable
12	SLV_ON	Slave port only when PROGN_EN set
13	PROGN_EN	Enable user IO on PROGN
14	INITN_EN	Enable user IO on INITN
15	DONE_en	Enable user IO on DONE

Table 85: Feature Row Definition - Feature

Bit	Name	Note
3:0	I3C Device Character [3:0]	
7:4	I3C Instance ID [3:0]	7:4
15:8	I3C Slave Address [7:0]	15:8
16	I3C_bit32	16
31:17	RESERVED	31:17
39:32	Unique ID MSB [7:0]	39:32
47:40	I2C Slave Address [7:0]	47:40
63:48	RESERVED	63:48
95:64	Custom ID Code [31:0]	

Table 86: Feature Row Definition - CR1

Bit	Name	Note
31:25	RESERVED	
24	DPA_EN	DPA enable for bitsteam decryption.
23:18	RESERVED	

Table 86: Feature Row Definition - CR1

Bit	Name	Note
17	32-bit SPIM address	Enable 32-bit MSPI addressing mode, default 24-bit addressing.
16	Bulk Erase Enable	Control bit to enable auto SRAM Bulk Erase for refresh or PROGRAMN pin toggle.
15	SFDP Enable	Enables checking the SFDP signature during controller SPI booting. Should be enabled for flash devices that support SFDP. 1-enable reading SFDP signature from flash SFDP header. 0-enable reading LSCC signature from flash starting at address 0x000
14	32-bit SPIM commands	Send 32-bit command set on MSPI, default 24-bit command set.
13	Disable IO glitch filter	Disable IO Glitch Filter during config
12	cfg_noise	CFG can enable the HSE noise generation by setting this to 1. If set to 0, gen_con[1] controls the noise generation.
[11:8]	Slave Idle Timer count value	Slave Idle Timer count value.
[7:5]	Master Preamble Timer count value	preamble_count: number of clock cycles to get a valid preamble. 0 600000 1 400000 2 200000 3 40000 4 20000 5 4000 6 2000 7 400

Table 86: Feature Row Definition - CR1

Bit	Name	Note
4	signature_dis	If sfdp_en=0 do not read LSCC signature from flash, go directly to reading preamble. 1=skip checking signature when sfdp_en=0, 0=signature enabled (default 0).
[3:2]	Master Preamble Timer retry value	Number of times the Master Preamble detection should be retried. 0-No retry 1-Retry 1 time 2-Retry 3 times 3-Retry forever
1	RESERVED	
0	IO Ready Disable	When set to 1 disables waiting for IO ready.

Reading the Feature Row (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP Only)

You can read the contents of the feature row of LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP, edit the settings, and program the new settings into the feature row of the device.

To read the feature row of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Feature Row Programming**.

To program data into Shadow Register instead of OTP eFUSE, choose **Feature Row Pseudo Programming**.

5. In the Operation box, select **Read Feature Row**.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

   The Feature Row dialog box displays. The fields in the dialog box are not editable.
8. Click **Close** to close the Feature Row dialog box.

To read the non-volatile control register 1 of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Feature Row Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Feature Row Pseudo Programming**.
5. In the Operation box, select **Display Control NV Register1**.
6. Click **OK** to close the Device Properties dialog box.
7. 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.
8. Click **Close** to close the dialog box.

Programming the Feature Row (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)

You can program the contents of a .fea file to the feature row of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP device.

To program the feature row of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Feature Row Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Feature Row Pseudo Programming**.
5. In the Operation box, select **Program Feature Row**.
6. In the Feature Row Programming Options box, browse to the Feature Row (.fea) file.
7. Select the sector lock options and **Lock Read** access, **Lock Write** access, and/or **Lock Fabric Port** access based on choice of security.
8. Click **OK** to close the Device Properties dialog box.
9. Choose **Design > Program,** or click on the toolbar.

To edit the feature row of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Feature Row Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Feature Row Pseudo Programming**.
5. In the Operation box, select **Update Feature Row**.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

   The Feature Row 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 feature row is programed as edited.

To edit the Non-Volatile Control Register 1 of an LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Feature Row Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Feature Row Pseudo Programming**.
5. In the Operation box, select **Program Control NV Register1**.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.
8. In the non-volatile Control Register 1 dialog box displays.
9. 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 device non-volatile Control Register 1 is programed as edited.

Programming the Password Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)

You can program the password key from a .key file to the LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP device.

To read the password key of an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.

4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.

5. In the Operation box, select **Read Password Key.** if the device is empty, you can read the all 0 password directly,. If the device is programmed with password, you need to enable **Password Protection Options** and select the key file.
6. Click OK to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

To program the password key to an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.

5. In the Operation box, select **Program Password Key.**
6. In the Password Key Programming Options box, browse to the key (.key) file.
7. Select the sector lock options and **Lock Read** access, **Lock Write** access, and/or **Lock Fabric Port** access based on choice of security.
8. Click **OK** to close the Device Properties dialog box.
9. Choose **Design > Program,** or click on the toolbar.

Programming the Encryption Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)

You can program the encryption key from a .bek file to the LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP device.

To read the encryption key of an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
5. In the Operation box, select **Read Encryption Key**.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

To program the encryption key to an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
5. In the Operation box, select **Program Encryption Key.**
6. In the Encryption Key Programming Options box, browse to the key (.bek) file.
7. If designed, select “Encrypted Bitstream Only” option, type “yes” to confirm.
8. Select the sector lock options and Lock Read access, Lock Write access, and/or **Lock Fabric Port** access based on choice of security.
9. Click OK to close the Device Properties dialog box.
10. Choose **Design > Program,** or click on the toolbar.

Programming the ECDSA Public Key (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP Only)

You can program the public key from a .pub, .pem, or .der file to the LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP device.

To read the public key of an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
5. In the Operation box, select **Read Public Key.**
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

To program the public key to an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
5. In the Operation box, select **Program Public Key**.

6. In the Public Key Programming Options box, browse to the key (.pub, .pem, or .der) file.
7. Select the sector lock options and Lock Read access, Lock Write access, and/or **Lock Fabric Port** access based on choice of security.
8. Click **OK** to close the Device Properties dialog box.
9. Choose **Design > Program,** or click  on the toolbar.

Executing LFMNX, LFMXO5, LFMXO5D, MachXO3D, MachXO3L, MachXO3LF Dry Run Features

LFMNX, LFMXO5, LFMXO5D, MachXO3D, MachXO3L, MachXO3LF can trigger a dry run bitstream downloading from external flash through the controller 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 does not override “SRAM USERCODE” or “Flash USERCODE”.

After the “dry run” finishes, it reports the execution status, such as execution pass/fail and authentication pass/fail. You 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.
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 is 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.
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 is displayed if the “Dry Run” executed successfully.

Setting Lock Policies (LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP)

LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP provide read, program permission to access the FPGA configuration sectors. The security setup can be applied inside each individual sector to store security setting bits.

The sectors include:

- SRAM
- Feature Row sector
- Password Key sector
- Public Key sector
- Encryption Key sector

The centralized security settings can be applied to store central security setting bits for all of the FPGA configuration sectors. The detailed explanation of each central security bit is given as follows:

Read Security Lock Inside dedicated security policy sector. This bit provides Read Protection for a particular sector.

Erase Security Lock Stops erase activity to the SRAM sector. This bit prevents accidental erasure of user set-up.

Write Security Lock Inside dedicated security policy sector. This bit provides Write Protection for a particular sector.

To read the locks policies of an LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Port Interface drop-down menu, choose **JTAG Interface**.
5. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
6. In the Operation box, select **Read Locks Policies**.
7. Click **OK** to close the Device Properties dialog box
8. Choose **Design > Program,** or click on the toolbar.

To program the locks policies 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Port Interface drop-down menu, choose **JTAG Interface**.
5. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
6. In the Operation box, select **Program Locks Policies**.
7. Select the sector **Lock Fabric Port Access** based on choice of security.
8. Click **OK**.

Setting External Configuration Ports Interfaces Locking Features (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP)

LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LFMXO5D, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C, or UT24CP have Write access permission control for external CFG ports and an internal system bus to access the FPGA configuration sectors. External CFG ports include JTAG, target SPI and slave I2C.

To setup the Ports interfaces security lock policy 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.
3. In the Target Memory drop-down menu, choose **Non Volatile Configuration Memory**.
4. In the Port Interface drop-down menu, choose **JTAG Interface**.
5. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.

   To program data into Shadow Register instead of OTP eFUSE, choose **Advanced Security Keys Pseudo Programming**.
6. In the Operation box, select **Lock Ports Interface**.
7. Select **Lock Ports Access** and/or **Lock System Bus Access** based on desired choices of security.
8. Click **OK**.

Setting LFMNX, MachXO3D, MachXO3L, MachXO3LF Version Rollback Protection Features

The LFMNX, MachXO3D, MachXO3L, MachXO3LF device families can support version rollback downloading from either on-die flash or external flash through the controller 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 execution 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.
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.

Programming the Flash Memory (LFMXO5, LFMXO5D Only)

You can program the contents of a .jed file to the configuration block of the flash memory of LFMXO5, LFMXO5D. The LFMXO5, LFMXO5D flash memory contains Flash Header, CFG0,CFG1,CFG2, UFM0, UFM1, UFM2, and 0-8 User Data memory blocks.

To program the flash memory for the LFMXO5, LFMXO5D device:

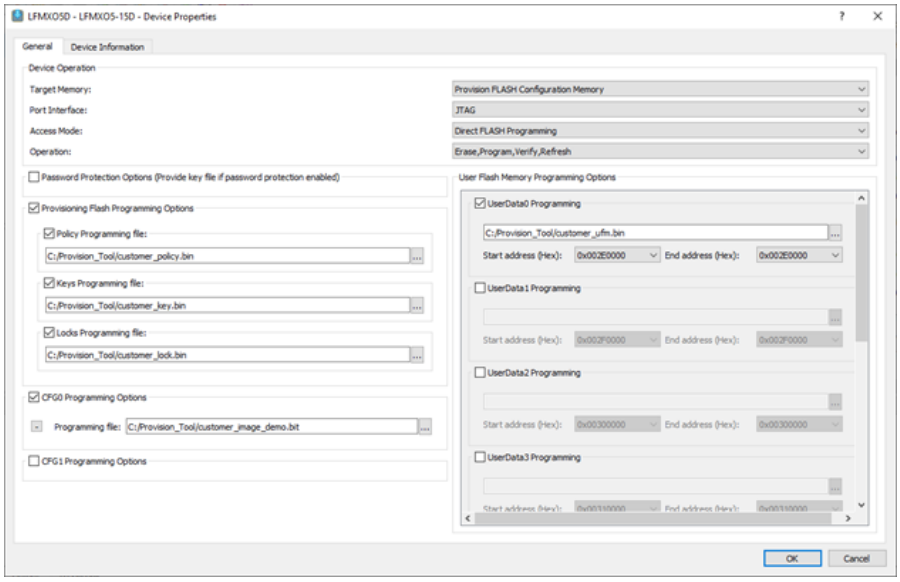
1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties,** or click the  button on the toolbar, or right-click and choose **Device Properties** to display the Device Properties dialog box.
3. In the Target Memory drop-down menu, choose **Flash Configuration Memory**.
4. In the Port Interface drop-down menu, choose **JTAG Interface** or **Slave SPI Interface**.
5. In the Access Mode drop-down menu, choose **Direct Programming** or **Background Programming**.
6. In the Operation box, select **Erase,Program,Verify**.
7. Select the memory block, or blocks (Flash Header, CFG0,CFG1,CFG2, UFM0, UFM1, UFM2, or 0-8 User Data) that you wish to program.
8. In the Programming File box, browse to and select the programming file (.jed for CFG0,CFG1,CFG2, UFM0, UFM1, UFM2. .mcs for Flash Header and 0-8 User Data).
9. Click **OK** to close the Device Properties dialog box.
10. Choose **Design > Program,** or click  on the toolbar.

Executing LFMXO5D Provision

This section describes the step-by-step procedure on how to perform provision using the LFMXO5D device.

To execute provision for the LFMXO5D device:

1. Connect the USB cable between your PC and MachXO5-NX Development Board's J11.
2. Launch Radiant Programmer.
3. Scan the device and select the following:
   - **Device Family**: LFMXO5D
   - **Device**: LFMXO5-15D
4. In the Device Properties dialog box, select the following:
   - **Target Memory**: Provision FLASH Configuration Memory
   - **Port Interface**: JTAG
   - **Access Mode**: Direct FLASH Programming
   - **Operation**: Erase, Program, Verify, Refresh



The **Operation** dropdown menu has the following options:

Table 87:

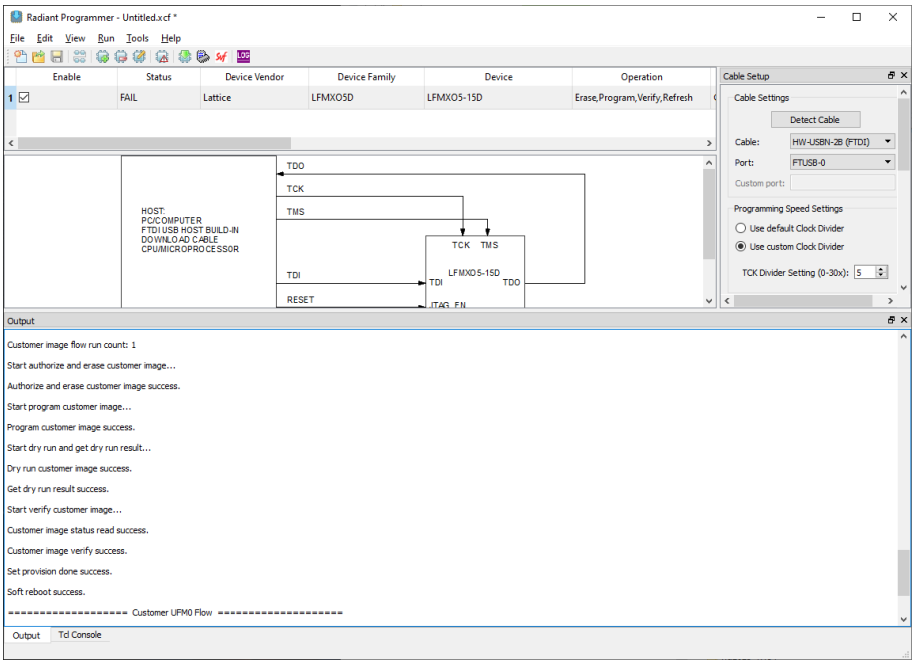
Operation	Description
Erase, Program, Verify, Refresh	Initial step to provision a new device with Customer Policy, Key, and Locks files. You cannot provision a device again once the process has been completed. Re-provisioning is allowed after the zeroization is executed successfully.
Customer Image Filed Upgrade	Executes multiple ESFB APIs to update the customer image in CFG0/CFG1, depending on the active image that is running. For example, when the current image is running at CFG0, this item will target CFG1 for an update or vice versa. This requires two input files, the keyblob_normal for Erase Customer Image, and the Customer Image .bit file. The sequence of API execution includes: ▶ Read Current Image Status ▶ Erase Customer Image ▶ Program Customer Image ▶ Dry-run Customer Image ▶ Set Primary Image
Read Current Image Status	Reads the current running customer image status.
Read Customer Image Status	Reads the specified customer image status.
Erase Customer Image	Erases the specified customer image.
Program Customer Image	Programs customer image into On-Chip flash.

Table 87:

Operation	Description
Dry-run Customer Image	Executes a dry run of the customer image. Dry-run can only be executed on non-actively running image.
Set Primary Image	Sets either image 1 or image 2 as the primary image.
Software Reboot	Reconfigures the FPGA with the primary image, similar to pulse PROGRAMN low, or REFRESH command via JTAG.
Set Customer Image Valid	Sets valid on an image. This function can only be executed on the current active image.
Read Minimum Version	Reads the minimum image version that the device can roll back.
Set Minimum Version	Sets the minimum image version that the device can roll back.
Read UFM	Reads the specified page in the UFM.
Lock UFM	Locks a certain UFM sector specified by you for the erase & write operation.
Unlock UFM	Unlocks a certain UFM sector specified by you for the erase & write operation.
Erase UFM	Erases the specified UFM
Program UFM	A program-specific page of the UFM.
Read Unique ID	Reads the Unique ID of the device.
Read Policy Sector	Reads customer policy related services that you can request from the ESFB IP.
Read Key Sector	Reads customer keys related services that you can request from the ESFB IP.
Read Lock Sector	Reads locks related services that you can request from the ESFB IP.
KAK Revoke	Authorizes the KAK revoke key blob and revokes the specified KAK.
Zeroization	Sets the device to zero state so that the device can be recommissioned.

5. In the **Provisioning Flash Programming Options** section, navigate to the location where these files are stored and click **Open**:
   - <file_name>.bin file
   - <file_name_key>.bin file
   - <file_name_lock>.bin file
6. In the **CFG0 Programming Options** section, browse to the location of your <file_name>.bit file and click **Open**.

7. In the **User Flash Memory Programming Options** pane, browse to the location of your UserData0 (ufm.bin) file and click **Open**.
8. Click **OK** to program the files into the device's FLASH.



Deploying the Design with the Deployment Tool

The Radiant 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.

The Radiant Deployment Tool is a stand-alone tool available from the Radiant Programmer software.

To run the Deployment Tool:

- Do one of the following:
  - In the Programmer window, choose **Tools > Deployment Tool**.
  - In the Windows Start menu, choose **All Programs > Lattice Radiant Programmer > Deployment Tool**. Requires the stand-alone Programmer.
  - In Linux, go to the <*Programmer install_path*>/bin/lin64 directory, and enter the following on a command line:

    ```
    ./deployment
    ```

    The Deployment Tool window opens.

A four-step wizard allows you to select deployment type, input file type, and output file type.

See Also

- Deployment Function Types
- iCE40 Warm Boot/Cold Boot Bitstream Output Options
- Input File Descriptions
- Input File Formats
- Bitstream Output File Descriptions
- Bitstream Output Formats
- SVF, and STAPL Output Formats
- Output Options Descriptions
- Creating a New Deployment
- Opening an Existing Deployment
- Encrypting a JEDEC with Deployment Tool
- Encrypting a Bitstream with Deployment Tool
- Authenticating a Bitstream with Deployment Tool (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP)
- Opening an Existing Deployment While Running the Deployment Tool
- Creating a New Deployment While Running the Deployment Tool
- Using Quick Launch Button to Create a New Deployment

- Viewing the Deployment Tool Log File
- Changing the Deployment Tool Log File Settings

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.

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
- Tester Deployment Function Type
- Embedded System Deployment Function Type
- External Memory Deployment Type

File Conversion Deployment Function Type

Not supported for iCE40UP devices.

This topic provides tables that list device family, input file types, output file extensions, and options for File Conversion deployment function type:

Table 88: IEEE 1532 ISC Data File—Device Family, Input File 1, and Output File

Device Family	Input File 1	Output File
MachXO3D	JEDEC (.jed)	ISC Data (.isc)
MachXO3L		
MachXO3LF		

Table 89: IEEE 1532 ISC Data File—Options

Device Family	Overwrite USERCODE		
	Off (Default)	On	Checksum
MachXO3D	Input File USERCODE.	User specified value.	JEDEC checksum.
MachXO3L			
MachXO3LF			

Table 90: Application Specific BSDL File—Device Family, Input File 1, Input File 2, and Output File

Device Family	Input File 1	Input File 2	Output File
LAV-AT	JEDEC (.jed)	Generic BSDL (.bsm)	BSDL File (.bsm)
LFCPNX	Argument File (.alt)		
LFD2NX			
LIFCL			
LFMNX			
LFMXO5			
UT24C			
UT24CP			
MachXO3D			
MachXO3L			
MachXO3LF			

Table 91: Application Specific BSDL File—Options

Device Family	Convert Bi-directional I/O's to Input and Output	
	Off (Default)	**On**
LAV-AT	Do not convert.	Convert.
LFCPNX		
LFD2NX		
LIFCL		
LFMNX		
LFMXO5		
UT24C		
UT24CP		
MachXO3D		
MachXO3L		
MachXO3LF		

Table 92: JEDEC File—Device Family, Input File 1, and Output File

Device Family	Input File 1	Output File
MachXO3D MachXO3L MachXO3LF	JEDEC (.jed)	JEDEC (.jed)
	JEDEC (.jed) Bitstream (.bit, .rbt)	
	JEDEC (.jed) ISC Data (.isc)	

Table 93: JEDEC File—Options—Overwrite USERCODE

Device Family	Overwrite USERCODE			
	Off (Default)	**On**	**Checksum**	**Disable (Internal Command Line Only)**
MachXO3D MachXO3L MachXO3LF	Input File USERCODE.	User specified value.	JEDEC checksum.	Omit USERCODE from JEDEC.

Table 94: JEDEC File—Options—Program Security Bit

Device Family	Program Security Bit		
	Default (Default)	**Off**	**On**
MachXO3D MachXO3L MachXO3LF	Use setting from input file.	Do not program security fuses.	Program security fuses.

Table 95: JEDEC File—Options—Encryption

Device Family	Encryption
	Encryption Key
MachXO3D	Load from BEK file.
MachXO3L	
MachXO3LF	

Table 96: Bitstream—Device Family, Input File 1, Format, and Output File

Device Family	Input File 1	Format	Output File
LAV-AT	Bitstream (.bit, .rbt)	Binary Bitstream	Bitstream (.bit)
LFCPNX			
LFD2NX			
LIFCL			
LFMNX			
LFMXO5			
UT24C			
UT24CP			
MachXO3D			
MachXO3L			
MachXO3LF			
LIFCL ES		ASCII Bitstream	Bitstream (.rbt)
		Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)

Table 97: Bitstream—Options—Encryption

Device Family	Encryption	
	Configuration Mode	**Encryption Key**
LAV-AT	Selects the appropriate configuration mode for the bitstream encryption.	Load from BEK File Load from BEK file.
LFCPNX		
LFD2NX		
LIFCL		
LFMNX		
LFMXO5		
UT24C		
UT24CP		
MachXO3D		
MachXO3L		
MachXO3LF		
LIFCL ES		

Table 98: Bitstream—Options—Encryption

Device Family	Encryption
LFCPNX	Encryption Key
LFD2NX	Load from BEK File
LFMNX	
LFMXO5	
LIFCL	
MachXO3D	
MachXO3L	
MachXO3LF	
UT24C	
UT24CP	
LIFCL ES	

Table 99: Bitstream—Options—Authentication

Device Family	Encryption
LFCPNX	Authentication
LFD2NX	Public Key and Private Key Signature
LIFCL	Load from GUI and load Public Key (.pub, .pem, or .der) file and Private File (.prv, .pem, or .der).
LFMNX	
LFMXO5	
UT24C	
UT24CP	
MachXO3D	
MachXO3L	
MachXO3LF	

Table 100: Bitstream Options—Verify ID Code, Frequency, Compression

Device Family	Verify ID Code			Frequency		Compression		
	Default (Default)	On	Off	Default (Default)	Frequency Selection	Default (Default)	On	Off
LAV-AT	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 bitstream data.	Do not compress.
LFCPNX								
LFD2NX								
LIFCL								
LFMNX								
LFMXO5								
UT24C								
UT24CP								
MachXO3D								
MachXO3L								
MachXO3LF								
LIFCL ES								

Table 101: Bitstream Options—CRC Calculation, Overwrite USERCODE, Byte Wide Bit Mirror, and Retain Bitstream Header

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**
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.

Table 102: JEDEC to Hex File Conversion

Device Family	Input File 1	Format	Output File
MachXO3D MachXO3L MachXO3LF	JEDEC (.jed)	ASCII Raw Hex	ASCII Hex (.hex)
		Binary Raw Hex	Binary Hex (.bin)

Tester Deployment Function Type

Not supported for iCE40UP devices.

This topic provides tables that list device family, input file types, output file extensions, and options for Tester deployment function type:

- SVF Single Device
- SVF JTAG Chain
- STAPL Single Device
- STAPL JTAG Chain

Table 103: SVF Single Device

Device Family	Input File 1	Output File
Lattice Single Device	Bitstream (.bit, .rbt)	SVF file (.svf)

Table 104: SVF Single Device—Options—Operation

Device Family	Operation
Lattice Single Device	Device-dependent list

Table 105: SVF Single Device—Options—Write Header and Comments, Write Rev D Standard SVF File

Device Family	Write Header and Comments		Write Rev D Standard SVF File	
	On (Default)	Off	Off (Default)	On
Lattice Single Device	Write headers and comments.	Omit headers and comments.	Rev. E Standard SVF. For Flash-based FPGAs, Lattice Extended SVF.	Rev. D Standard SVF.

Table 106: SVF Single Device—Options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify

Device Family	Use RUNTEST from Rev C		For Erase, Program, and Verify Operations, Skip Verify	
	Off (Default)	On	Off (Default)	On
Lattice Single Device	Standard RUNTEST.	Rev. C RUNTEST.	Include verify in EPV operations.	Omit verify in EPV operations.

Table 107: SVF Single Device—Options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)

Device Family	Include RESET at the End of the SVF File		Set Maximum Data Size per Row (Kbits)	
	Off (Default)	On	Off (Default)	On
Lattice Single Device	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

Table 108: SVF JTAG Chain

Device Family	Input File 1	Output File
Lattice Chain	XCF file (.xcf)	SVF file (.svf)

Table 109: SVF JTAG Chain—Options—Operation

Device Family	Operation
Lattice Single Device	Contained in XCF file.

Table 110: SVF JTAG Chain—Options—Write Header and Comments, Write Rev D Standard SVF File

Device Family	Write Header and Comments		Write Rev D Standard SVF File	
	On (Default)	Off	Off (Default)	On
Lattice Single Device	Write headers and comments.	Omit headers and comments.	Rev. E Standard SVF. For Flash-based FPGAs, Lattice Extended SVF.	Rev. D Standard SVF.

Table 111: SVF JTAG Chain—Options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify

Device Family	Use RUNTEST from Rev C		For Erase, Program, and Verify Operations, Skip Verify	
	Off (Default)	On	Off (Default)	On
Lattice Single Device	Standard RUNTEST.	Rev. C RUNTEST.	Include verify in EPV operations.	Omit verify in EPV operations.

Table 112: SVF JTAG Chain—Options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)

Device Family	Include RESET at the End of the SVF File		Set Maximum Data Size per Row (Kbits)	
	Off (Default)	On	Off (Default)	On
Lattice Single Device	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

Table 113: STAPL Single Device

Device Family	Input File 1	Output File
Lattice Single Device	Bitstream (.bit, .rbt)	STAPL file (.stp)

Table 114: STAPL Single Device—Options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify

Device Family	ACA Compression		Include Print Statements		For Erase, Program, and Verify Operations, Skip Verify	
	On (Default)	Off	Off (Default)	On	Off (Default)	On
Lattice Single Device	Compress data.	Uncompress data.	Print statements as comments.	Include print statements.	Include verify in EPV operations.	Omit verify in EPV operations.

Table 115: STAPL JTAG Chain

Device Family	Input File 1	Output File
Lattice Chain	XCF file (.xcf)	STAPL file (.stp)

Table 116: STAPL JTAG Chain—Options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify

Device Family	ACA Compression		Include Print Statements		For Erase, Program, and Verify Operations, Skip Verify	
	On (Default)	Off	Off (Default)	On	Off (Default)	On
Lattice Chain	Compress data.	Uncompress data.	Print statements as comments.	Include print statements.	Include verify in EPV operations.	Omit verify in EPV operations.

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.

Table 117: JTAG Full VME Embedded -- Device Family, Input XCF FIe, and Output File

Device Family	Input XCF File	Output File
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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)

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 118: JTAG Full VME Embedded Options -- Compress VME File, Convert VME Files to HEX (c.) File-Based Embedded VME

Device Family	Compress VME		Convert VME Files to HEX (c.) File-Based Embedded VME	
LAV-AT	On (Default)	Off	Off (Default)	On
LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Unchecking 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.

The following table lists options -- Compact VME File, Fixed Pulse Width (Rev. D) -- for JTAG Full VME embedded deployment.

Table 119: JTAG Full VME Embedded Options -- Compact VME File, Fixed Pulse Width (Rev. D)

Device Family	Compact VME File		Fixed Pulse Width (Rev. D)	
LAV-AT	Off (Default)	On	Off (Default)	On
LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.

The following table lists options -- Verify USERCODE, Program Device if Fails -- for JTAG Full VME embedded deployment.

Table 120: JTAG Full VME Embedded Options -- Verify USERCODE, Program Device if Fails

Device Family	Verify USERCODE, Program Device if Fails	
LAV-AT	Off (Default)	On
LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.

The following table lists options -- Maximum Memory Allocation Size Per Row of Data (Kbytes) -- for JTAG Full VME embedded deployment.

Table 121: JTAG Full VME Embedded Options -- Include Header, Include Comment, Maximum Memory Allocation Size Per Row of Data (Kbytes)

Device Family	Include Header		Include Comment		Maximum Memory Allocation Size Per Row of Data (Kbytes)
LAV-AT	Off (Default)	On	Off (Default)	On	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.
LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.	Omit header from VME file	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 in VME file.	

The following table lists device family, input file types, and output file. extensions for JTAG Slim VME embedded deployment.

Table 122: JTAG Slim VME Embedded -- Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.

The following table lists options -- Compressed Embedded Files, convert VME files to HEX (c.) File-Based Embedded VME -- for JTAG Slim VME embedded deployment..

Table 123: JTAG Slim VME Embedded Options -- Compressed VME Data File, Convert VME files to HEX (c.) File-Based Embedded VME

Device Family	Compress VME Data File		Convert VME files to HEX (c.) File-Based Embedded VME	
LAV-AT	On (Default)	Off	Off (Default)	On
LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Unchecking 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.

The following table lists device family, input file types, and output file extensions for Target SPI embedded deployment.

Table 124: Slave SPI Embedded -- Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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)
iCE40	XCF File (.xcf) NVCM (.nvcm)		

The following table lists options -- Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME -- for Target SPI embedded deployment..

Table 125: Slave SPI Embedded Options -- Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME

Device Family	Device FamilyConvert VME files to HEX (c.) File-Based Embedded VME		Generate Hex (.c) File		Operation (.xcf and .jed files only)
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	On (Default)	Off	On (Default)	On	The list of options and operation is dependent on which device is selected.
	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Unchecking 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.	

The following table lists device family, input file types, and output file extensions for I2C embedded deployment.

Table 126: I2C Embedded -- Device Family, Input File 1, Output File 1, and Output File 2

Device Family	Input File	Output File 1	Output File 2
LAV-AT	XCF File (.xcf)	Algorithm File (*_algo.sea)	Data File (*_data.sed)
LFCPNX	Bitstream (.bit)		
LFD2NX			
LIFCL			
LFMNX			
LFMXO5			
UT24C			
UT24CP			
MachXO3D			
MachXO3L			
MachXO3LF			
LIFCL ES			

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 127: 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

Device Family	Compressed Embedded Files		Convert VME files to HEX (c.) File-Based Embedded VME		Operation (.jed files only)	I2C Slave Address
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	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.		

The following table lists options -- Include Comments for I2C embedded dployment.

Table 128: I2C Embedded Options -- Include Comments

Device Family	Include Comments	
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Off (Default)	On
	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.

The following table lists options -- Fixed Pulse Width for I2C embedded deployment.

Table 129: I2C Embedded Options -- Fixed Pulse Width

Device Family	Include Comments	
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Off (Default)	On
	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.

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
- Dual Boot
- Ping-Pong Boot
- Advanced SPI Flash
- sysCONFIG Daisy Chain

Table 130: Hex Conversion

Device Family	Input File	Output File 1	Output File 2
iCE40	Bitstream (.bin)	Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tectronix Hex	Hex (.xtek)
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tectronix Hex	Hex (.xtek)

Table 131: Hex Conversion Options—Program Security Bit

Device Family	Program Security Bit		
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	**Default (Default)**	**Off**	**On**
	Use setting from input file.	Do not program security fuses.	Program security fuses.

Table 132: Hex Conversion Options—Encryption

Device Family	Encryption	
	Configuration Mode	**Encryption Key**
LAV-AT LIFCL LFMNX LFMXO5 LFD2NX UT24C MachXO3D MachXO3L MachXO3LF LIFCL ES	Selects the appropriate configuration mode for the bitstream encryption.	Load from BEK File.

Table 133: Hex Conversion Options—Authentication

Device Family	Encryption
LAV-AT LIFCL LFD2NX LFMNX LFMXO5 UT24C MachXO3D MachXO3L MachXO3LF LIFCL ES	**Authentication**
	Public Key and Private Key Signature
	Load from GUI and load Public Key (.pub, .pem, or .der) file and Private File (.prv, .pem, or .der).

Table 134: Hex Conversion Options—Verify ID Code, Frequency, Compression

Device Family	Verify ID Code			Frequency		Compression		
LAV-AT LIFCL LFD2NX LFMNX LFMXO5 LIFCL ES MachXO3D MachXO3L MachXO3LF	**Default (Default)**	**On**	**Off**	**Default (Default)**	**Frequency Selection**	**Default (Default)**	**On**	**Off**
UT24C								
	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 bitstream data.	Do not compress.

Table 135: Hex Conversion Options—CRC Calculation

Device Family	CRC Calculation		
	Default (Default)	**On**	**Off**
LAV-AT LFCPNX LFD2NX LIFCL LFMNX LFMXO5 UT24C UT24CP MachXO3D MachXO3L MachXO3LF LIFCL ES	Use setting from input file.	Include CRC for each data frame.	Include Global CRC only.

Table 136: Hex Conversion Options—Byte Wide Mirror

Device Family	Off (Default)	On
iCE40	No change to bit order.	No change to bit order.
LAV-AT	No change to bit order.	No change to bit order.
LFCPNX		
LFD2NX		
LIFCL		
LFMNX		
LFMXO5		
UT24C		
UT24CP		
MachXO3D		
MachXO3L		
MachXO3LF		
LIFCL ES		

Table 137: Hex Conversion Options—Retain Bitstream Header

Device Family	Off (Default)	On
LAV-AT	Replace Bitstream Header.	Retain Bitstream Header.
LFCPNX		
LFD2NX		
LIFCL		
LFMNX		
LFMXO5		
UT24C		
UT24CP		
MachXO3D		
MachXO3L		
MachXO3LF		
LIFCL ES		

Table 138: Hex Conversion Options—Starting Address

Device Family	Starting Address	
LAV-AT	**Off (Default)**	**Starting Address**
LFCPNX		
LFD2NX		
LIFCL		
LFMNX		
LFMXO5		
UT24C		
UT24CP		
MachXO3D		
MachXO3L		
MachXO3LF		
LIFCL ES		
	No address offset.	Starting address in 0x010000 increments.

Table 139: Dual Boot

Device Family	Golden Pattern Input File 1	Primary Pattern Input File 2	Format	Output File
LAV-AT	Bitstream (.bit, .rbt)	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
LIFCL			Motorola Hex	Hex (.exo)
LFD2NX			Extended Tektronix Hex	Hex (.xtek)
LFMNX				
LFMXO5				
LIFCL ES				
MachXO3D				
MachXO3L				
MachXO3LF				

Table 140: Dual Boot Options—SPI Flash Size

Device Family	SPI Flash Size
LAV-AT	1/2/4/8/16/32/64/128/256/513 (512 Mb)
LIFCL	
LFD2NX	
LFMNX	
LFMXO5	
LIFCL ES	
MachXO3D	
MachXO3L	
MachXO3LF	

Table 141: Dual Boot Options—Protect Golden Sector

Device Family	Protect Golden Sector	
LAV-AT	**Off (Default)**	**On**
LIFCL		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL ES		
MachXO3D		
MachXO3L		
MachXO3LF		
	Locate golden in next available sector.	Locate Golden at the beginning of the upper half.

Table 142: Dual Boot Options—Bye Wide Mirror and Retain Bitstream Header

Device Family	Byte Wide Mirror		Retain Bitstream Header	
LAV-AT	**Off (Default)**	**On**	**Off (Default)**	**On**
LFCPNX				
LFD2NX				
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				
	No change to bit order.	Flip each byte of data.	Replace Bitstream Header.	Retain Bitstream Header.

Table 143: Dual Boot Options—SPI Flash Read Mode

Device Family	SPI Flash Mode			
LAV-AT	**Off (Default)**	**Fast Read**	**Dual I/O**	**Quad I/O**
LFCPNX				
LFD2NX				
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				
	Standard Read Opcode	Fast Read Opcode	Dual I/O Read Opcode	Quad I/O Read Opcode

Table 144: Dual Boot Options—Optimize Memory Space

Device Family	SPI Flash Mode	
	Off (Default)	**On**
LAV-AT	Use worst case bitstream size.	Use file size.
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		

Table 145: Dual Boot Options—Encryption

Device Family	Off (Default)	On
LAV-AT	Normal bitstream	Primary encrypted bitstream
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		

Table 146: Dual Boot Options—Authentication

Device Family	Encryption
LAV-AT	**Authentication**
LFCPNX	
LFD2NX	
LFMNX	
LFMXO5	
LIFCL	
MachXO3D	
MachXO3L	
MachXO3LF	
UT24C	
UT24CP	
	Public Key and Private Key Signature
	Load from GUI and load Public Key (.pub, .pem, or .der) file and Private File (.prv, .pem, or .der).

Table 147: Ping-Pong Boot

Device Family	Golden Pattern Input File 1	Primary Pattern Input File 2	Format	Output File
LAV-AT	Bitstream (.bit, .rbt)	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
LFCPNX			Motorola Hex	Hex (.exo)
LFD2NX			Extended Tektronix Hex	Hex (.xtek)
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				

Table 148: Ping-Pong Boot Options—SPI Flash Size

Device Family	SPI Flash Size
LAV-AT	1/2/4/8/16/32/64/128/256/513 (512 Mb)
LFCPNX	
LFD2NX	
LFMNX	
LFMXO5	
LIFCL	
MachXO3D	
MachXO3L	
MachXO3LF	
UT24C	
UT24CP	
LIFCL ES	

Table 149: Ping-Pong Boot Options—Generate Jump Table Only

Device Family	Protect Golden Sector	
LAV-AT	**Off (Default)**	**On**
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		
	Full images	Jump table only

Table 150: Ping-Pong Boot Options—Byte Wide Mirror and Retain Bitstream Header

Device Family	Byte Wide Mirror		Retain Bitstream Header	
LAV-AT	**Off (Default)**	**On**	**Off (Default)**	**On**
LFCPNX				
LFD2NX				
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				
	No change to bit order.	Flip each byte of data.	Replace Bitstream Header.	Retain Bitstream Header.

Table 151: Ping-Pong Boot Options—SPI Flash Read Mode

Device Family	SPI Flash Mode			
LAV-AT	**Off (Default)**	**Fast Read**	**Dual I/O**	**Quad I/O**
LFCPNX				
LFD2NX				
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				
	Standard Read Opcode	Fast Read Opcode	Dual I/O Read Opcode	Quad I/O Read Opcode

Table 152: Ping-Pong Boot Options—Primary Starting Address—Secondary Starting Address

Device Family	SPI Flash Mode	
LAV-AT	Use default.	Use custom.
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		

Table 153: Ping-Pong Boot Options—Encryption

Device Family	Off (Default)	On
LAV-AT	Normal bitstream	Primary encrypted bitstream
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		

Table 154: Ping-Pong Boot Options—Authentication

Device Family	Encryption
LAV-AT	**Authentication**
LFCPNX	
LFD2NX	
LFMNX	
LFMXO5	
LIFCL	
MachXO3D	
MachXO3L	
MachXO3LF	
UT24C	
UT24CP	
	Public Key and Private Key Signature
	Load from GUI and load Public Key (.pub, .pem, or .der) file and Private File (.prv, .pem, or .der).

Table 155: Advanced SPI Flash

Device Family	Input File 1	Format	Output File
LAV-AT	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
LFCPNX		Motorola Hex	Hex (.exo)
LFD2NX		Extended Tektronix Hex	Hex (.xtek)
LFMNX			
LFMXO5			
LIFCL			
MachXO3D			
MachXO3L			
MachXO3LF			
UT24C			
UT24CP			
LIFCL ES			

Table 156: Advanced SPI Flash - iCE40

Device Family	Enable Cold Boot		Number of Patterns					
iCE40	**Off (Default)**	**On**	**Number of Patterns**	**Pattern 1**	**Starting Address 1**	**Pattern 2 - 3**	**Pattern 4**	**Starting Address 4**
	POR Boot Pattern		2/3/4	Selected Pattern 1	0x010000 Sectors	User Data 2 - 3	Selected Pattern 4	0x010000 Sectors
	1/2/3/4	Cold Boot						

Table 157: Advanced SPI Flash Options—SPI Flash Size

Device Family	SPI Flash Size
iCE40	512 (512Kb)/1/2/4/8/16/32/64/128/256/513 (512Mb)
LAV-AT	
LFCPNX	
LFD2NX	
LFMNX	
LFMXO5	
LIFCL	
MachXO3D	
MachXO3L	
MachXO3LF	
UT24C	
UT24CP	
LIFCL ES	

Table 158: Advanced SPI Flash Options—Byte Wide Mirror

Device Family	Byte Wide Mirror	
LAV-AT	**Off (Default)**	**On**
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		
	No change to bit order.	Flip each byte of data.

Table 159: Advanced SPI Flash Options—Retain Bitstream Header

Device Family	Retain Bitstream Header	
LAV-AT	**Off (Default)**	**On**
LFCPNX		
LFD2NX		
LFMNX		
LFMXO5		
LIFCL		
MachXO3D		
MachXO3L		
MachXO3LF		
UT24C		
UT24CP		
LIFCL ES		
	Replace Bitstream Header.	Retain Bitstream Header.

Table 160: Advanced SPI Flash Options—User Hex Files

Device Family								
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**Off (Default)**	**On**	**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**
	Off	User data	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

Table 161: Advanced SPI Flash Options—SPI Flash Read Mode

Device Family				
LFCPNX LFD2NX LFMNX LFMXO5 LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**Standard Read (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

Table 162: Advanced SPI Flash Options—SPI Flash Read Mode (Avant)

Device Family						
LAV-AT	**Standard Read (Default)**	**Fast Read**	**Dual I/O**	**Quad I/O**	**xSPI Profile 1 Read**	**xSPI Profile 2 Read**
	Standard Read Opcode	Fast Read Opcode	Dual I/O Read Opcode	Quad I/O Read Opcode	Bitstream only command. Selects the mode and frequency to boot from MSPI.	Bitstream only command. Selects the mode and frequency to boot from MSPI.

Table 163: Advanced SPI Flash Options—Clock Mode

Device Family	Standard/Fast Read Dual/Quad IO	xSPI Profile 1 Read xSPI Profile 2 Read
LAV-AT	Single-ended clock	Single-ended clock
		Differential Clock

Table 164: Advanced SPI Flash Options—Multiple Boot

Device Family	Multiple Boot												
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP LIFCL ES	**Off (Default)**	**On**										**Optimize Memory Space**	
			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**
	No Multiboot	Multiboot	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.

Table 165: sysCONFIG Daisy Chain

Device Family	Input File 1	Input File 2	Format	Output File
LAV-AT LIFCL LFD2NX LFMNX LFMXO5 LIFCL ES MachXO3D MachXO3L MachXO3LF UT24C	Bitstream (.bit, .rbt)	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
			Motorola Hex	Hex (.exo)
			Extended Tektronix Hex	Hex (.xtek)

Table 166: sysCONFIG Daisy Chain Options—Merge Format

Device Family	Merge Format			
	Intelligent (Default)	Intelligent Merge		Combine
		Frequency	Slayer Output	
LAV-AT LIFCL LFD2NX LFMNX LFMXO5 LIFCL ES UT24C	Intelligent Merge	Default/(device-specific selections)		Combine Merge

Table 167: sysCONFIG Daisy Chain Options—Byte Wide Mirror and Retain Bitstream Header

Device Family	Byte Wide Mirror		Retain Bitstream Header	
LAV-AT	**Off (Default)**	**On**	**Off (Default)**	**On**
LFCPNX				
LFD2NX				
LFMNX				
LFMXO5				
LIFCL				
MachXO3D				
MachXO3L				
MachXO3LF				
UT24C				
UT24CP				
LIFCL ES				
	No change to bit order.	Flip each byte of data.	Replace Bitstream Header.	Retain Bitstream Header.

iCE40 Warm Boot/Cold Boot Bitstream Output Options

The following output format options are available for generating iCE40 warm boot/cold boot bitstreams. Except where otherwise noted, the default setting is listed first for each option.

Intel Hex/Motorola Hex/Extended Tektronix Hex

SPI Flash Size: 64 Mb (512 Kb - 256 Mb): Possible PROM sizes are 512 Kb, 1 Mb, 2 Mb, 4 Mb, 8 Mb, 16Mb, 32 Mb, 64 Mb, 128 Mb, and 256 Mb.

Byte Wide Bit Mirror: OFF/ON: Flips each byte. Default: bytes are not flipped. Optional.

Number of Patterns: 2 (2 - 4): Specifies the number of iCE40 configuration files.

Enable Cold Boot: OFF/ON: Optional. Default is warm boot. If selected, cold boot is chosen.

Power On Reset Boot Image 1 (1 - 4): Indicates the input configuration file path and name. Must be specified for each configuration file.

Pattern (1 - 4) Starting Address: SPI flash address location for the configuration file. Must be specified for each configuration file.

See Also

- External Memory Deployment Type
- Creating a New Deployment

Input File Descriptions

The following table shows how files produced by the Radiant software 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 Radiant for a Lattice 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 168: Input Files

Input File	Description	Output
.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, ISC, BIT, or RBT.	Bitstream, ISC, JED, Intel, Motorola, and Extended Tektronix Hex
.rbt	Bitstream ASCII file (RBT) generated from a JEDEC, 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 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

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 169: Input File Formats

Device Family	Input for Bitstream Output	Multiple Input/ Output	Input for SVF Output	Input for ISC Output	Input for JEDEC Output
iCE40	.bin	Not available	Not available	Not available	Not available
LAV-AT	.bit, .rbt	.bit, .rbt	.bit, .rbt	Not available	Not available
LFCPNX					
LFD2NX					
LFMNX					
LFMXO5					
LIFCL					
MachXO3D					
MachXO3L					
MachXO3LF					
UT24C					
UT24CP					

See Also

- Input File Descriptions

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.

Table 170: Bitstream Output File

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.

Table 170: Bitstream Output File (Continued)

File Extension	Description
.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	Motorola hexadecimal output data file.
.mcs	Intel hexadecimal output data file.
.xtek	Extended Tektronix hexadecimal output data file.

Bitstream Output Formats

The Deployment Tool generates bitstream data files. The table below shows the types of bitstreams and output formats that can be generated for each device family.

Table 171: Bitstream Output Formats

Device Family	Single Bitstream	Merged Bitstream
iCE40	Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)	Not available
LAV-AT LFCPNX LFD2NX LFMNX LFMXO5 LIFCL MachXO3D MachXO3L MachXO3LF UT24C UT24CP	Bitstream Binary (.bit) Bitstream ASCII (.rbt) Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)	Intel Hex Merge (.mcs) Motorola Hex Merge (.exo) Extended Tektronix Hex Merge (.xtek)

See Also

- Bitstream Output File Descriptions

SVF, and STAPL Output Formats

The Deployment Tool generates Serial Vector Format (SVF), In-System Configuration (ISC), and Standard Test and Programming Language (STAPL) data files for Lattice devices.

SVF Output Files

SVF files are generated from ISC, binary bitstream (BIT), or ASCII bitstream (RBT) input files.

ISC Output Files

ISC files are generated from binary bitstream (BIT), ASCII bitstream (RBT), binary readback (RBK), ASCII readback (RBKA), binary mask (MSK), and ASCII mask (MSKA) input files.

STAPL Output Files

STAPL files are generated from JEDEC input files.

See Also

- Input File Descriptions

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

The Deployment Tool has a wizard interface that guides you 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 one of the following ways:
   - To start Deployment Tool that is integrated in the Radiant software:
     - In the Radiant software, open Programmer by choosing **Tools > Programmer**.
     - In the Programmer window, choose **Tools > Deployment Tool**.
   - To start Deployment Tool from stand-alone Radiant Programmer:
     - In the Programmer window, choose **Tools > Deployment Tool**.

       or

     - From the Windows Start menu, choose **All Programs > Lattice Radiant Programmer > Deployment Tool.**
   - In Linux, from the *<Programmer install_path>/bin/lin64* 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
   - External Memory
4. In the Output File Type box, choose an output file type.
5. Click **OK**. The Deployment Tool loads the device database. A four step process must now be completed.
6. In “Step 1”, do one of the following, depending on the type of file you are using:
   - If you are creating a deployment from an .xcf file, use the  **(Browse)** button to browse to the project’s .xcf file.
   - If you are creating a deployment from a data file, select Input File(s) window, click **... (Browse)** and browse to the projects .jed file. The Device Family and Device boxes are automatically populated by the Deployment Tool.
7. Click **Next**.
8. In “Step 2” (Options dialog box), select the desired options for your output file type.
9. Click **Next**.

10. In the “Step 3” (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” (Generate Deployment dialog box), review the Deployment Tool Summary and Command Line Information.
13. Choose **File > Generate**, or click the button, to perform the Deployment. Results of the generation are available for review.
14. Choose **File > Save**, or click the button, to save the deployment (.ddt) file to your desired location.

See Also

- iCE40 Warm Boot/Cold Boot Bitstream Output Options

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 one of the following ways:
   - To start Deployment Tool that is integrated in the Radiant software:
     - In the Radiant software, open Programmer by choosing **Tools > Programmer**.
     - In the Programmer window, choose **Tools > Deployment Tool**.
   - To start Deployment Tool from stand-alone Radiant Programmer:
     - In the Programmer window, choose **Tools > Deployment Tool**.

     or
     - From the Windows Start menu, choose **All Programs > Lattice Radiant Programmer > Deployment Tool.**
   - In Linux, from the <*Programmer install_path*>/bin/lin64 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. A four step process must now be completed.
4. In “Step 1”, do one of the following, depending on the type of file you are using:
   - If you are creating a deployment from an .xcf file, use the ... **(Browse)** button to browse to the project’s .xcf file.

- If you are creating a deployment from a data file, select Input File(s) window, click **... (Browse)** and browse to the projects .jed file. The Device Family and Device boxes are automatically populated by the Deployment Tool.

5. Click **Next**.
6. In “Step 2” (Options dialog box), select the desired options for your output file type.
7. Click **Next**.
8. In the “Step 3” (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” (Generate Deployment dialog box), review the Deployment Tool Summary and Command Line Information.
11. Choose **File > Generate**, or click the button, to perform the Deployment. Results of the generation are available for review.
12. Choose **File > Save**, or click the button, to save the deployment (.ddt) file to your desired location.

See Also

- Creating a New Deployment
- iCE40 Warm Boot/Cold Boot Bitstream Output Options

Encrypting a JEDEC with Deployment Tool

Encryption settings are initially set in the Diamond software using the Security Setting Tool. Refer to Bitstream Security Settings Tool.

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, LFMXO5, MachXO3D, MachXO3L, and MachXO3LF families 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
- FPGA-TN-02202-1.7, Advanced Security Encryption Key Programming Guide for ECP Device Family

See Also

- Bitstream Security Settings Tool
- Encrypting a Bitstream with Deployment Tool

Encrypting a Bitstream with Deployment Tool

Encryption settings are initially set in the Diamond software using the Security Setting Tool. Refer to Bitstream Security Settings Tool.

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

LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP 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
- FPGA-TN-02202-1.7, Advanced Security Encryption Key Programming Guide for ECP Device Family

See Also

- Bitstream Security Settings Tool
- Encrypting a JEDEC with Deployment Tool

Authenticating a Bitstream with Deployment Tool (LAV-AT, LFCPNX, LFD2NX, LFMNX, LFMXO5, LIFCL, MachXO3D, MachXO3L, MachXO3LF, UT24C or UT24CP)

Authentication settings are initially set in the Diamond software using the Security Setting Tool. Refer to Bitstream Security Settings Tool.

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.

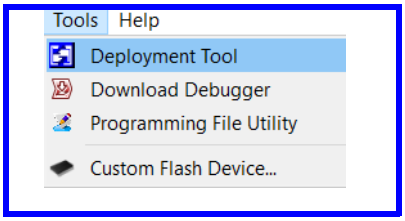
See Also

- Bitstream Security Settings Tool

Generating STAPL File for LFMXO5D Provision

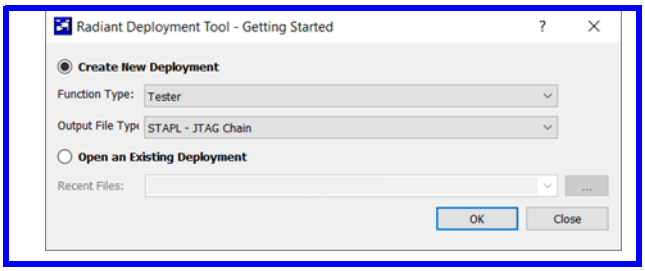
The following steps describe how to generate a STAPL file.

1. In **Programmer,** save your project to create a .xcf file.
2. Launch the Deployment Tool by clicking **Tools > Deployment Tool** from the toolbar.

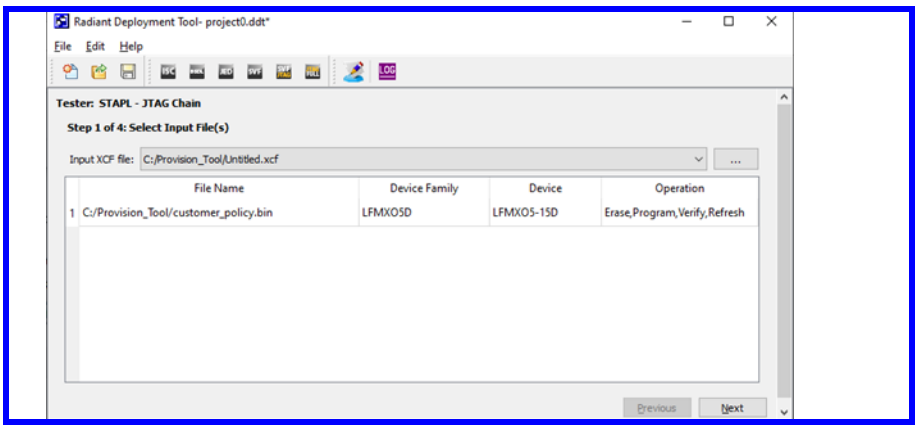


3. In the Deployment Tool dialog box, select the following options:
   - **Function Type**: Tester
   - **Output File Type**: STAPL: JTAG Chain

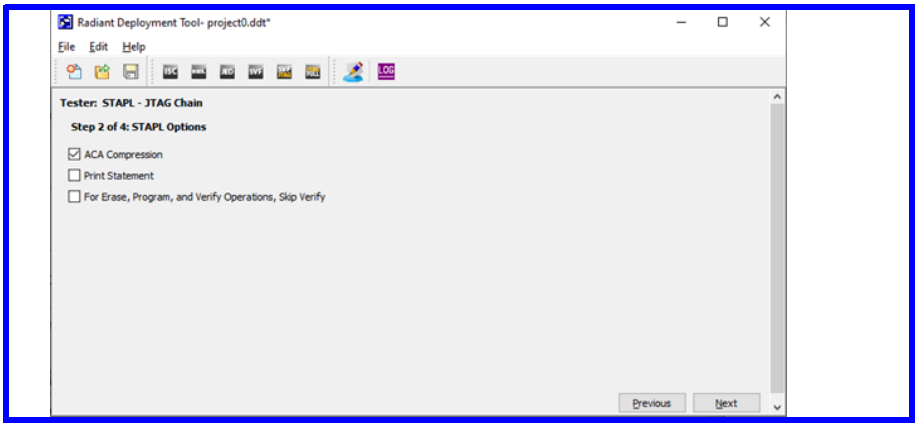
   Click **OK**.



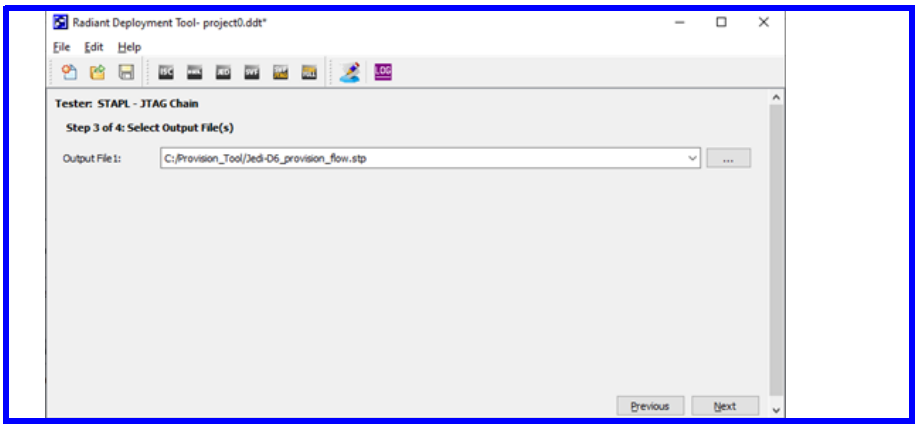
4. Input your .xcf file by browsing to the directory where you saved the file. Click **Next**.



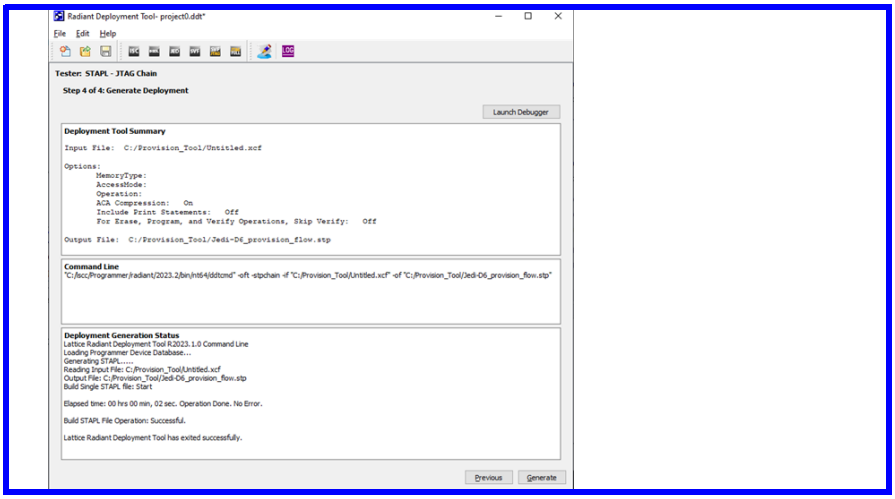
5. In **STAPL Options**, select the options you want to add and click Next.



6. In **Output STAPL**, browse to the location of your .stp file and click Open. Once added, click Next.



7. Click the **Generate** button to generate the STAPL file.



If the file has been generated successfully, click the **Launch Debugger** button in the upper-right corner of the dialog box to launch the tool and process your STAPL file.

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**.
2. In the Open dialog box, browse to the deployment (.ddt) file that you wish to open and select it.
3. Click **Open**.

See Also

- Creating a New Deployment
- iCE40 Warm Boot/Cold Boot Bitstream Output Options

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:

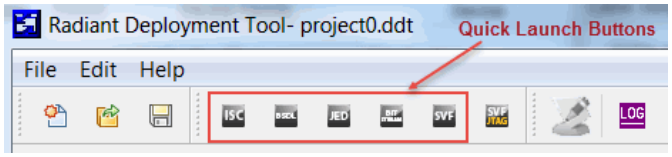
1. Choose **File > New**, or click the button.
2. Deployment Tool prompts you to save the .ddt file for the project you are currently working on. If you wish to save the deployment you are currently working on, click **Yes**, and in the Save As dialog box, create a name for the .ddt file and save in the desired location.
3. In the Function Type drop-down menu, choose from one of the options.
4. In the Output File Type box, choose an output file type.
5. Click **OK**. The Deployment Tool loads the device database.

See Also

- Creating a New Deployment
- Using Quick Launch Button to Create a New Deployment
- iCE40 Warm Boot/Cold Boot Bitstream Output Options

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
- JEDEC to Hex
- SVF - Single Device
- SVF - JTAG Chain
- STAPL - Single Device
- STAPL - JTAG Chain
- JTAG Full VME Embedded
- JTAG Slim VME Embedded
- Slave SPI Embedded
- I2C Embedded
- Hex Conversion
- Dual Boot
- Ping Pong Boot
- Advanced SPI Flash
- sysCONFIG Daisy Chain

To customize which quick launch buttons appear in the Deployment Tool toolbar:

1. Place the cursor over a quick launch button, 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
- Creating a New Deployment While Running the Deployment Tool
- iCE40 Warm Boot/Cold Boot Bitstream Output Options

Viewing the Deployment Tool Log File

The log file shows recent actions in a text editor.

To view the log file:

Click  button on the toolbar.

See Also

- Deploying the Design with the Deployment Tool

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 Log File Path dialog box, browse to the folder where you wish to save the deployment_tool.log file.

To clear the log file each time the application starts:

1. Choose **Edit > Settings**.
2. 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

Debugging SVF, STAPL, and VME Files

Download Debugger is a stand-alone software tool for debugging the following file types:

- Serial Vector Format (SVF)
- Standard Test And Programming Language (STAPL)
- Lattice Embedded (VME)

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
- Download Debugger Software Support of SVF Operations
- Running Download Debugger
- Opening an Existing SVF File
- Creating a New SVF File
- Setting Device Programming Options in Download Debugger
- Setting Port Assignments and Options in Download Debugger
- Setting and Removing Breakpoints in an SVF File
- Processing an SVF File
- Viewing the Download Debugger Log File
- Saving a Log File in Download Debugger
- Clearing the Contents of a Log File in Download Debugger
- Editing an SVF File
- Opening an Existing STAPL File
- Creating a New STAPL File
- Viewing STAPL Processing
- Setting and Removing Breakpoints in a STAPL File
- Processing a STAPL File in Download Debugger
- Viewing the Download Debugger Log File
- Saving a Log File in Download Debugger
- Clearing the Contents of a Log File in Download Debugger
- Editing a STAPL File
- Setting Windows Options in Download Debugger

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

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.
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

Running Download Debugger

Download Debugger runs separately from the Radiant software environment.

To start the Download Debugger:

1. Issue the start command in one of the following ways:
   - To start Download Debugger that is integrated in the Radiant software:
     - In the Radiant software, open Programmer by choosing **Tools > Programmer**.
     - In the Programmer window, choose **Tools > Download Debugger**.
   - To start Download Debugger from stand-alone Radiant Programmer:
     - In the Programmer window, choose **Tools > Download Debugger**.

       or

     - From the Windows Start menu, choose **All Programs > Lattice Radiant Programmer > Download Debugger.**
   - In Linux, from the *<Programmer install_path>/bin/lin64* directory, enter the following on a command line:

     ```
     ./debugger
     ```

See Also

- Debugging SVF, STAPL, and VME Files

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

Creating a New SVF File

To create a new empty SVF file:

- In Download Debugger, choose **File > > New > SVF** to open the SVF Debugger Edit window and create a new empty SVF file.

See Also

- Debugging SVF, STAPL, and VME Files

Setting Device Programming Options in Download Debugger

To set SVF options:

1. In Download Debugger, choose **Configuration > Options**.

   The Options Dialog Box opens.

2. Select the options that you want and click **OK**.

See Also

- Debugging SVF, STAPL, and VME Files

Setting Port Assignments and Options in Download Debugger

Use the Cable and I/O Port Settings Dialog Box to specify the preferred location for download, select the port address or cable, and choose 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 opens.

2. Select the options that you want and click **OK**.

See Also

- Debugging SVF, STAPL, and VME Files

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. 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 > Toggle 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

Processing an SVF File

After setting device programming options and port assignments, you can process your SVF file with the Download Debugger. You can program a device with a download cable, step through each line of the process, reset it and begin again.

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 reset a process and start a process from the beginning:

- Choose **Command > Reset**.

See Also

- Debugging SVF, STAPL, and VME Files

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, 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
- Debugging SVF, STAPL, and VME Files

Saving a Log File in Download Debugger

To save a log file:

1. In Download Debugger, choose **Configuration > Options**.
2. The Options Dialog Box opens.
3. Use the Browse button to specify the path of the directory you wish to save your log file to.
4. Click **OK**.

See Also

- Debugging SVF, STAPL, and VME Files

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

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

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) file, and then click **Open**.

   The selected STAPL file opens in the Download Debugger.

See Also

- Debugging SVF, STAPL, and VME Files

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

Viewing STAPL Processing

To view the output information during STAPL processing, you should open the Output Information dialog box before processing the STAPL file.

See Also

- Debugging SVF, STAPL, and VME Files

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. 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 > Toggle 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

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.

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 reset a process and start a process from the beginning:

- Choose **Command > Reset**.

See Also

- Debugging SVF, STAPL, and VME Files

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 Error** in the Options Dialog Box, 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
- Clearing the Contents of a Log File in Download Debugger

- Debugging SVF, STAPL, and VME Files

Saving a Log File in Download Debugger

To save a log file:

1. In Download Debugger, choose **Configuration > Options**.

   The Options Dialog Box opens.
2. Use the Browse button to specify the path of the directory you want to save your log file to.
3. Click **OK**.

See Also

- Debugging SVF, STAPL, and VME Files

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

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

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**.

See Also

- Debugging SVF, STAPL, and VME Files

Download Debugger Options

This section lists the options available in Download Debugger.

Topics include:

- Cable and I/O Port Settings Dialog Box
- Options Dialog Box

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 target SPI port of the selected device. This allows you to debug target SPI device programming

Using JTAGI2C Interface Connection (HW-USBN-2B Cable Only)

Only use an HW-USBM-2B cable for a JTAGI2C interface connection.

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 - 30x)

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 172: 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 173: 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

Table 173: USB-2B (2232D FTDI USB host chip) (Continued)

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 174: 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 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 PROGRAM 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

Options Dialog Box

The following options are available in the within the three tabs (General, SVF, STAPL) found in the Options dialog box.

Log File Name

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 Line 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.

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.

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).

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.

See Also

- Download Debugger Options

Using Programming File Utility

The Radiant 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
- Viewing Data Files
- Comparing Two Data Files
- Editing Feature Row Values
- Editing Control Register0 or Control Register1 Values
- Control Register1 Editor Dialog Box
- Editing the USERCODE in the Data File

Running Programming File Utility

Open Programming File Utility from the Tools menu of Programmer. Or you can also open it directly if you have the stand-alone Programmer installed or are working on a Linux system.

To run the Programming File Utility:

- Do one of the following:
  - In the Programmer window, choose **Tools > Programming File Utility**.
  - In the Windows Start menu, choose **All Programs > Lattice Radiant Programmer > Programming File Utility**. Requires the stand-alone Programmer.
  - In Linux, go to the <*Programmer install_path*>/bin/lin64 directory, and enter the following on a command line:

    ```
    ./fileutility
    ```

  The Programming File Utility window opens.

See Also

- Using Programming File Utility

Viewing Data Files

The data file can be in any of the following formats: .jed, .isc, .rbt, .rbka, .mska, .bit, .rbk, .msk, .bsm, .bsd, .bin, .hex, .mcs, .exo, xtek, nvcm, .out.

To view a data file:

- In the Programming File Utility, choose **File > Open** and select the data file you want to view.

Use the **Edit > Find** and **Edit > Find Next** 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

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 Files 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 which shows dots where the files are the same. When 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, and then a comparison line. The comparison line shows dots where the two readback files are the same. When different, the

line is highlighted in red and the data from the second readback file is displayed.

See Also

- Using Programming File Utility

Editing Feature Row Values

You can use the Feature Row Editor to enable or disable silicon features in Lattice FPGA 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 Dialog Box opens.
2. Using the **(Browse)** button, select the data (.fea) file you want to edit.
3. Click **Read** to display current data file settings.

   The software displays the default values in the top row. The second row shows the values that can be modified in black, and those that cannot be modified in red.
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

Editing Control Register0 or Control Register1 Values

You can use either the Control Register0 Editor or the Control Register1 Editor 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** or choose **Tools > Control Register1 Editor**.

   The Control Register 0 editor or Control Register 1 editor opens (see Control Register0 Editor Dialog Box or Control Register1 Editor Dialog Box for more information).

2. Using the **(Browse)** button, browse to the .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.
4. The software displays the default values in the top row.

   The software displays the default values in the top row. The second row shows the values that can be modified in black, and those that cannot be modified in red.

5. 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.
6. To overwrite the existing data file, choose **Save.**
7. To create a new data file with a different name, choose **Save As** and save as a different file name.
8. A message box pops up, displaying the new control register value. Click **OK**.
9. A confirmation message box informs you that Write to file was successful. Click **OK**.
10. Click **Close** to exit the Control Register 0 or Control Register 1 dialog box.

See Also

- Using Programming File Utility

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.

Note:

The LFMXO5 USERCODE in a JEDEC file for internal flash cannot be updated.

To use the USERCODE Editor:

1. Choose **Tools > USERCODE Editor**.

   The USERCODE Editor opens (see USERCODE Editor Dialog Box for more information).

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 in decimal (0-9), ASCII, or hexadecimal (0-F) characters.
5. Click **Save** or **Save As** to save the USERCODE data when finished.
6. Click **Yes** when prompted to overwrite the data file.
7. A confirmation message box informs 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
- Running Programming File Utility

Securing the Device and Setting the Persistent Bit

The Security and Persistent Fields Editor allows you to read both the security settings and the persistent bit of the bitstream file and to 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.**

   The Security and Persistent Fields Editor opens (see Security and Persistent Fields Editor Dialog Box for more information).

2. Click the **Browse** button and select the desired bitstream data file.
3. Click **Read** to view the settings for Security and Persistent Bit and display them in the Status cells.
4. To change a setting from Disabled to Enabled 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 **Save**. 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 **Save As**. In the Save As Data File dialog box, specify file name and location, and click **Save**.
6. A confirmation message box informs 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
- Running Programming File Utility

Programming File Utility Options

This section lists the options available in Programming File Utility. Topics include:

- Feature Row Editor Dialog Box
- Control Register0 Editor Dialog Box
- Control Register1 Editor Dialog Box
- USERCODE Editor Dialog Box
- Security and Persistent Fields Editor Dialog Box

Feature Row Editor Dialog 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 them 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

- Editing Feature Row Values
- Programming File Utility Options

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 them 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

- Editing Control Register0 or Control Register1 Values
- Programming File Utility Options

Control Register1 Editor Dialog Box

The following options are available in the Control Register1 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 them 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

- Editing Control Register0 or Control Register1 Values
- Programming File Utility Options

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

- Editing the USERCODE in the Data File
- Programming File Utility Options

Security and Persistent Fields Editor Dialog Box

The following options are available in the Security and Persistent Fields 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 chip value information.

Read

Reads the security and persistent field in the data file and displays it in the display field.

Save

Writes the value in the security and persistent field and saves the data file.

Save As

Writes the value in the security and persistent field and allows you to save the data file with another file name.

Close

Closes this dialog box.

See Also

- Securing the Device and Setting the Persistent Bit
- Programming File Utility Options

Testing and Debugging On-Chip

The final stage of developing a design is testing it on the actual Field Programmable Gate Arrays (FPGA) on either a test board or in your system. Lattice Semiconductor offers the following tools for debugging both the hardware aspect of the design and— if you are using Lattice Propel — the software aspect:

- Reveal Inserter to add debug modules to your design
- Reveal Analyzer to check for and to analyze specific events on signals

Before you start debugging, Reveal Analyzer requires that a special interface module be added to the design. After it is added, rerun the design implementation process (Synthesize Design, Map Design, Place & Route Design) and generate bitstream data file to program the FPGA. The tools can share the same ispDOWNLOAD cable and JTAG port that Programmer uses to download the design.

See Also

- Debugging with Reveal Usage Guidelines and Tips
- Reveal User Guide for Radiant Software
- Reveal Troubleshooting Guide for Lattice Radiant Software

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 both with all supported FPGA devices.

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 QuestaSim, or to an ASCII tabular format that can be used with tools such as Excel.

Before running Reveal Analyzer, use Reveal Inserter to add Reveal modules to your design. In these modules, specify the signals to monitor, define the trigger conditions, and other options. Reveal supports multiple logic analyzer cores using hard or soft JTAG interface. Hard JTAG is basically a JTAG block IP that is part of the device. Soft JTAG is built in RTL, which uses the device fabric logic. When working with Nexus and Avant devices, you can minimize logic resources by merging hard and soft JTAGhubs in Reveal. See "Using JTAGhub" on page 650.

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 file to program the FPGA.

A feature is enabling an added module called Reveal Controller. This controller module enables:

- Access to the control and status registers of the hard IPs such as I2CFIFO, PLL, PCIe, CDR, and DPHY via the LMMI (Lattice Memory Mapped Interface) slave interface.
- Virtual switches and LEDs emulating on-board switches and LEDs to control and monitor a user design. Up to 32 switches and 32 LEDs are supported.
- Read/write access to a bank of user registers and/or initialize memory post-configuration.

The main purpose is narrowing down to problem areas during debug cycles using a divide and conquer method into many small functional blocks to control and monitor the status of each block.

Before starting a test run, set up Reveal Analyzer. This includes setting 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.

Note:

Reveal now supports IEEE-P1735 encryption. If this encryption is applied to a design, the design tree will allow only the visible ports and signals that are not encrypted to be inserted by the Reveal Inserter for triggering purposes.

See Also

- Lattice Radiant Software User Guide

Using Reveal Example Projects

For hands-on experience using the Reveal tools, the following example projects targeting various devices are available.

Name	Device	Description
LFD2NX_Reveal_controller_example	LFD2NX	A basic design for the Certus-NX device, which displays an example pattern and state using LEDs and a 7-segment display. The design can be interfaced using Reveal Controller, or the I/Os on the device to control functionality and the pattern output. It also includes a PLL to generate the clock as well as a DPRAM to store the sample patterns. You can find a detailed description of this example project in the header of the top.v file.
LFMXO5_Reveal_controller_example	LFMXO5	A basic design for the MachXO5-NX device, which displays an example pattern and state using LEDs and a 7-segment display. The design can be interfaced using Reveal Controller, or the I/Os on the device to control functionality and the pattern output. It also includes an oscillator to generate the clock as well as a DPRAM to store the sample patterns. You can find a detailed description of this example project in the header of the top.v file.
counter_reveal	iCE40UP	A basic design for the iCE40 UltraPlus device, which displays an RGB LED. It also includes PWM (pulse-width modulation) control for the RGB channels. The design can be interfaced using Reveal Controller, or the I/Os on the device to control functionality and the RGB LED output.
example_reveal_avant	LAV-AT	A simple example design demonstrating Reveal capabilities targeting Avant device. It is a 32-bit counter with an asynchronous reset and an output to toggle LEDs. It counts up and toggles LEDs based on the internal oscillator clock.
example_reveal_nexus	LIFCL	A simple example design demonstrating Reveal capabilities targeting the CrossLink-NX device. It is a 32-bit counter with an asynchronous reset and an output to toggle LEDs. It counts up and toggles LEDs based on the internal oscillator clock.

These example projects are coded in Verilog with a Reveal module already inserted. Trace data is also configured and can be viewed in Reveal Analyzer. A test board is not required to open the Reveal tools or to view the existing trace data. To actually run Reveal Analyzer to collect new data, however, you will need a test board.

To start a Reveal example project:

1. Choose **File > Open > Design Example**.

The Open Example dialog box opens.

2. Select the example project.
3. Browse to select the location.
4. Click **OK**.

   The 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 Process bar on top, double-click **Export Files** to implement the design.

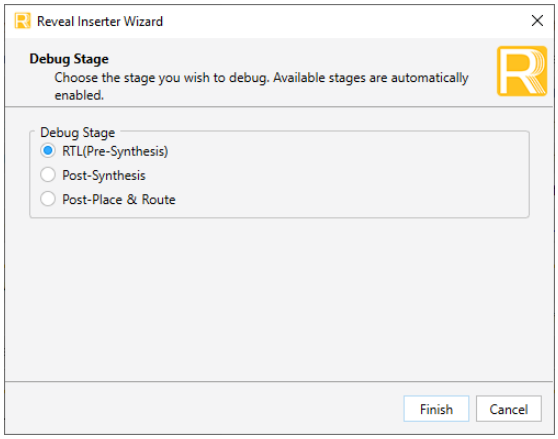
If you want to use the example project with a test board, change the device type to match the board by double-clicking the device in the File List view and running the design implementation process.

Debug Flows

Reveal Inserter features three debug flows. When you open Reveal Inserter, the Reveal Inserter Wizard prompts you to choose the debug flow to use.

- RTL (Pre-Synthesis) – This is the standard flow for debugging a design all at once. The user inserts the logic core before the design is synthesized the first time. Refer to "RTL (Pre-Synthesis) Debug Flow" on page 611 for details.
- Post-Synthesis – This flow allows you to insert debug logic after running synthesis. As such, this option is available if you have already run synthesis. It is useful for isolating or monitoring certain signals and focus your changes or improvements in specific areas of the design. Refer to "Post-Synthesis Debug Flow" on page 612 for details.
- Post-Place & Route – This flow allows you to insert debug logic after running place and route. As such, this option is available if you have already completed synthesis, mapping, and placement and routing. Also referred to as non-intrusive debugging, you can add your debug core after place and route without having to alter your original design. The post-place & route debug flow is available when designing for Avant and Nexus devices. Refer to "Post-Place & Route Debug Flow" on page 614 for details.

Figure 35: Reveal Inserter Wizard



RTL (Pre-Synthesis) Debug Flow

The following are the general steps in the Reveal Inserter RTL debug flow.

1. Start Reveal Inserter.
2. In the Reveal Inserter Wizard, the **RTL (Pre-Synthesis)** debug stage is selected. Click **Finish**.
3. Create a new Reveal Inserter project or open an existing Reveal Inserter project.
4. Add and set up analyzer and controller modules.
5. Insert the debug logic.

   This process generates and synthesizes the debug logic.

   The generated .rvl is automatically imported into the Radiant software if you enabled the “Activate Reveal file in design project” option in the Insert Debug to Design dialog box.
6. Map, place, and route the design.
7. Generate the bitstream file.

   If you want to perform logic analysis with Reveal Analyzer, continue with these steps:
8. Set up the cable connection with Programmer.
9. Download the design onto the device.
10. Start Reveal Analyzer and perform logic analysis with it.

Post-Synthesis Debug Flow

In post-synthesis debugging, it is required that you mark the specific signal objects that you want to monitor by adding the syn_rvl_debug synthesis attribute in the RTL source files. This allows you to keep track of these signals more easily as they are highlighted on the user interface. The syn_rvl_debug attribute is the same for both Synplify and LSE tools.

The syn_rvl_debug attribute performs two functions:

- This attribute highlights the signal so it can be easily identified in the user interface.
- After synthesis, this attribute tells the synthesis tool to preserve the signal without optimizing it. If it is a bus, the data width is also preserved. The synthesis tool passes the same attribute to the signal in the post-synthesis netlist (*.vm).

The syntax in Verilog/System-Verilog is:

```
/* synthesis syn_rvl_debug = 1 */;
```

The syntax in VHDL is:

```
attribute syn_rvl_debug : boolean;
attribute syn_rvl_debug of sig1 : signal is true
```

For example:

```
input clki /*synthesis syn_rvl_debug = 1 */,
wire clk1 /* synthesis syn_rvl_debug = 1 */;
reg [39:0] cnt /* synthesis syn_rvl_debug = 1 */;
reg [31:0] cnti /* synthesis syn_rvl_debug = 1 */;
```

The following are the general steps in the Reveal Inserter Post-Synthesis debug flow.

1. After running synthesis, start Reveal Inserter.
2. In the Reveal Inserter Wizard, select **Post-Synthesis** as debug stage and click **Finish**.

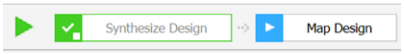
   In the list of signals displayed in Reveal Inserter, the signals with the syn_rvl_debug attributes are highlighted for easy tracking.
3. Add and set up analyzer and controller modules, if needed.

   Note

   When Reveal Controller is used for switches in post-synthesis, make sure the signals are not driven by other signals in the design. Multiple driver issues produce an error in Place & Route.
4. Click the **Design Rule Check** button and fix errors, if needed.
5. Click the **Insert Debug** button.
6. In the Insert Debug to Design dialog box, select the modules to insert. Select **Activate Reveal file in design project** to include it in synthesis. This also sets other .rvl files as inactive.

7. Click **OK**. The .rvl file is listed in the File List view under Debug Files.

   You will note that the Synthesize Design process bar now has a small white box to indicate that only Reveal is to be synthesized in post-synthesis stage.



8. Run **Synthesize Design**. Only the changes in Reveal are processed.

   In the Task Detail View, you will observe the added Post-Synthesis Reveal task in progress.

9. Map, place, and route the design.
10. Generate the bitstream file.

    If you want to perform logic analysis with Reveal Analyzer, continue with these steps:

11. Set up the cable connection with Programmer.
12. Download the design onto the device.
13. Start Reveal Analyzer and perform logic analysis with it.

Identifying Signals in Post-Synthesis Debug Stage

The synthesis tool may add a suffix to the name of a signal with the syn_rvl_debug attribute. This change is minimal and the signal remains recognizable to the user.

The syn_rvl_debug attribute may be attached to top level input/output ports, which will be connected to the input/output buffers. In case of an input port with the syn_rvl_debug attribute, in the .vm file, the output of the input buffer should get the passed attribute. In case of an output port with attribute, in the .vm file, the input of the output buffer should get the passed attribute.

Example 1

RTL Code:

```
input clki /* synthesis syn_rvl_debug = 1 */,
```

The expected output is:

```
wire clki_c /* synthesis syn_rvl_debug = 1 */;
```

where clki is driving the input port of IB and clki_c is connected to the output port. This is similar in the output port:

```
output dout /* synthesis syn_rvl_debug = 1 */,
```

The expected output is:

```
wire dout_c /* synthesis syn_rvl_debug = 1 */
```

Example 2

RTL Code:

```
reg [31:0] counter /*synthesis syn_rvl_debug = 1*/;
```

This is the expected .vm code when the exact signal name is preserved:

```
wire [31:0] counter /*synthesis syn_rvl_debug = 1*/;
```

When the exact signal name cannot be preserved, some suffix(*) may be added.

```
wire [31:0] counter_* /*synthesis syn_rvl_debug = 1*/;
```

Example 3

If the attribute is provided in the following way, it is applied to the last signal:

RTL Code:

```
reg sig1, sig2, sig3 /*synthesis syn_rvl_debug = 1*/;
```

The expected .vm code is:

```
wire sig3 /*synthesis syn_rvl_debug = 1*/;
```

The synthesis tool may also expand/elaborate the RTL code if the attribute is attached to the states in the FSM. This depends on the type of encoding such as binary, one-hot and others.

Post-Place & Route Debug Flow

Similar to the post-synthesis debug flow, in post-place & route debugging, it is required that you mark the specific signal objects that you want to monitor by adding the syn_rvl_debug synthesis attribute in the RTL source files. The syntax for Verilog and VHDL are the same. See “Post-Synthesis Debug Flow” on page 612.

The following are the general steps in the Reveal Inserter Post-Place & Route debug flow.

Note

When you initially run synthesis, map, and place & route, make sure there is no active Reveal (.rvl) project. If there is an active .rvl project, make it inactive by right-clicking the .rvl file In the File List view and choosing Set as Inactive.

1. After running synthesis, map, and place & route, start Reveal Inserter.

2. In the Reveal Inserter Wizard, select **Post-Place & Route** as debug stage and click **Finish**.

   Note

   In the Tcl Console, you will see that a new Reveal project is created in the post-PAR stage. The placed and routed design is preserved in the PAR output udb.

   ```
   > rvl_new_project -stage postpar
   ```

   In the design tree, the entire database after place and route is shown. The signals with the syn_rvl_debug attributes are highlighted.

3. Add and set up logic analyzer cores

   Note

   Adding Reveal Controller cores is not supported in post-Place & Route debug flow.

4. Click the **Design Rule Check** button and fix errors, if needed.
5. Click the **Insert Debug** button.
6. In the Insert Debug to Design dialog box, select the modules to insert. Select **Activate Reveal file in design project** to include it in synthesis. This also sets other .rvl files, if any, as inactive.

   Click **OK**.

7. In the Save Reveal Project dialog box, type the file name of the .rvl file and click **Save**. The .rvl file is listed in the File List view under Debug Files.

   You will note that the Place & Route process bar now has a small white box to indicate that only Reveal is to be processed in post-place and route stage.

8. In the Task Detail View, you will observe that the following sub-tasks are unchecked. These will be run when you click the Place & Route button.
   - Place & Route Reveal
   - Place & Route Timing Analysis
9. Run **Place & Route**. Only the changes in Reveal are processed.
10. Generate the bitstream file.

    If you want to perform logic analysis with Reveal Analyzer, continue with these steps:

11. Set up the cable connection with Programmer.
12. Download the design onto the device.
13. Start Reveal Analyzer and perform logic analysis with it.

Creating Reveal Modules

Create the modules for Reveal logic analysis with Reveal Inserter. The process 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.

Currently, you can only have one module due to the soft JTAG core. 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.

Take some time to plan the logic analysis, as 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, and 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.

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 pulldown menu. See "Managing the Modules in a Project" on page 617.

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 619.

Design Tree also keeps track of how signals are being used. See “Viewing Signals in Design Tree” on page 618.

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 641.

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 625.

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 629
- “Setting Up Trigger Units” on page 631
- “Setting Up Trigger Expressions” on page 634
- “Setting Trigger Options” on page 641

Controller Setup

Reveal Controller Setup is where you setup all your Virtual (Configuring) Switch/LED Settings, User Registers, and Hard IP selections. See:

- “Setting Up Virtual Switch/LED Settings” on page 642
- “Configuring User Memory Setup” on page 643
- “Configuring Hard IP Setup” on page 647

Managing the Modules in a Project

Each Reveal Inserter project can include up to 15 modules and are listed in the Dataset view. You can add, rename, and remove modules in the Dataset view.

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 name 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 619.

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 used in the Reveal module, it appears in the Design Tree pane in bold, 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.

Note that if a design, modules or instances are encrypted with the IEEE-P1735 encryption standard, a user will be able to see either only the ports and signals or only the ports of a module or no module altogether depending on where the encryption was placed.

Searching for Signals

You can find signals in the Design Tree pane with a text search that includes wild card characters.

Note:

If you added the syn_rvl_debug attribute to your signals, these signals are highlighted in yellow.

To search for signals:

1. In the Signal Search box in the Design Tree pane, type the full name of the signal or part of the name with wild cards (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 for “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 found.

3. Select the desired signals and click **OK**.

   Click on a signal to select it. To select more than one signal, control-click on each one. To select all signals and buses in a range, click on the first signal to select and Shift-click on the last one.

   The selected signals are highlighted in the Design Tree pane.

See Also

- "Viewing Signals in Design Tree" on page 618

Checking the Design Rules

Use the Design Rule Check tool to verify 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 rule check are displayed in the Output tab.

Saving a Project

Reveal Inserter automatically performs a design rule check whenever a project is saved. 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 new .rvl file name in the File Name box, and click **Save**.

See Also

- "Setting Up Virtual Switch/LED Settings" on page 642

Limitations

Reveal Inserter has the following limitations:

Unsupported VHDL, Verilog and System Verilog Features in Reveal Inserter

The following features are valid in the VHDL, Verilog and System Verilog languages but 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 declare variables at the beginning of a module body to avoid obtaining different results from various synthesis tools.
- In VHDL, you must declare synthesis attributes within an entity, and 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.
- Some signals in a System Verilog design appear in the signal hierarchy but are not available for triggering or tracing.

  Array types of two dimensions or more are not shown in the port or node section.

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, System Verilog or VHDL code are available for use with Reveal Inserter.

Synthesis Parameters

VHDL generics for synthesis must be added via HDL Parameters field in Project Properties. The current version does not support adding parameters via Command Line Options field in Synthesis Strategy setting.

Relative Path Syntax

When using LSE, there are instances when Reveal is unable to read files, such as .coe INIT, that use the relative path syntax.

For example:

```
parameter pSPI_TF_Init    = "RF_Init.coe", // Rx FIFO Memory
Init
```

You must use the full absolute path of these files instead.

For example:

```
parameter pSPI_TF_Init    = "C:/Users/<user name>/
nexus_solutions/Reveal_roadmap/readmemh/22_ml6c5x-path/source/
RF_Init.coe", // Rx FIFO Memory Init
```

Creating Logic Analysis Modules

The major steps in creating a Logic Analysis module are shown below.

The design must be successfully synthesized before running Reveal Inserter.

Note

Reveal Inserter has limitations and requirements in how it works with the design. See "Limitations" on page 621 and "Creating Logic Analysis Modules" on page 622.

To create a Logic Analyzer module:

1. After opening the design project, choose **Tools > Reveal Inserter**.

   Reveal Inserter launches with the active Reveal project (.rvl) file open. 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.

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 617.
3. Set up the trace signals as desired. See "Setting Up Trace Signals" on page 625.
4. Create trigger units, which are collections of signals and the values that are part of causing a trigger signal. See "Setting Up Trigger Units" on page 631.
5. Create trigger expressions, which are logical combinations of trigger units. See "Setting Up Trigger Expressions" on page 634.
6. Select trigger options. See "Event Counter" on page 641 and "Trigger Out" on page 641.
7. Once you have set up of 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 648.

To create a Reveal Controller module:

1. After opening the design project, choose **Tools > Reveal Inserter**.

   Reveal Inserter launches with the active Reveal project (.rvl) file open. 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.

   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. Click **Add Core > Add Controller** a module in the Dataset view or you can add a controller by choosing **Debug > Add New Core > Add Controller**. Note that only one Reveal Controller can be added to a project. See also "Managing the Modules in a Project" on page 617.
3. On the bottom tab, select Virtual Switch Switch & LED Setup tab to set up the Virtual Switch/LED Setting as desired. In the Datasets pane, a Signal Search box is located at the bottom for searching signals. Individual bits or buses can be dragged into the LED List pane. The Address Width field on the top of the pane will adjust accordingly. See "Setting Up Virtual Switch/LED Settings" on page 642.

4. On the bottom pane, select User Register Setup tab to setup the user registers. See "Configuring User Memory Setup" on page 643.
5. On the bottom pane, select the Hard IP Setup tab to setup IP debug. See "Configuring Hard IP Setup" on page 647
6. Once you have set up of 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 648.

Using the Reveal Controller Simulation Model

The Reveal Controller Simulation Model replicates the functionality of Reveal Controller in simulation.

To set up the Reveal Controller Simulation Model:

1. Choose **Tools > Reveal Inserter** to insert a reveal module into the design.
2. Run **Synthesize Design** to generate the reveal_workspace file.

- If you insert Reveal Controller only, the tool will create the sim_src folder with the pre-synthesis Reveal generated source file under reveal_workspace.
- If you insert Reveal Controller and Analyzer, the Reveal engine will not create the folder and no source files will be provided.

3. Click **Tools > Simulation Wizard** or click the icon in the toolbar.
4. In the Simulator Project Name and Stage dialog box, enter the project name.

   Click **Next**.
5. In the Add and Reorder Source dialog box, select and add the source files from the sim_src folder.

   Click **Next**.

The following compiler directives are all enabled in the source file. You can use the OEM simulator or any stand-alone simulator to compile and simulate the Controller:

- `define en_sw
- `define en_led
- `define en_user_mem
- `define en_user_creg
- `define en_user_sreg

Setting Up Trace Signals

Setting up the trace signals for a module consists of selecting the signals and buses to watch and dragging them into the Trace Signal Setup tab. You can have up to 512 trace signals in a module.

There is no limit to the bus organization of the design, so you can organize the signals according to how you want to view the data. Order them from top to bottom and organize them into buses, or rearrange the trace signals into your own organization of trace buses. You can manage them 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 619.

   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. 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 626.
6. Set the trace options. There are several options that enhance the trace signals or control how data is sampled:
   - “Sample Clock” on page 626
   - “Sample Enable” on page 626
   - “Buffer Depth” on page 627
   - “Timestamp” on page 627
   - “Implementation” on page 628
   - “Data Capture Mode” on page 628

See Also

- "Viewing Signals in Design Tree" on page 618

Include Trigger Signals in Trace Data

In order to monitor what happens on the signals that make up a trigger, you can add all signals used in the trigger units to the trace signals.

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 contains 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 619

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 during 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 that are associated with a single signal.

An example is a design that contains different sections, with some sections only working during certain clock phases. The design uses a master clock and generates different signals for the phases. You can 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 occurs in this region, but it cannot determine during which clock cycle the trigger occurs.

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 619

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 for your test run. Be generous to avoid re-implementing the design to get a larger 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 contains extra data in each sample and 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 want to do.
3. In the pulldown menu next to “Timestamp,” select 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 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 EBR would be selected, but distributed RAM can be used if you are short of EBR.

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 on the screen. 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 “Power-on Reset (POR) Debug” on page 629.

Power-on Reset (POR) Debug

The POR Debug is a helpful tool in recording unusual events in the device that occurs after configuration. An automatic "trigger enable" signal must be built into the Reveal module to monitor POR functions. Before Reveal Analyzer starts, these functions happen immediately after the power-on of the test board. When the trigger enables signal transitions to "active," the module will watch for the trigger and collect samples. It 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 not supported in iCE40UP devices.

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**.
4. Continue to setup **Trigger Unit** and **Trigger Expression**.

Note:

Only one Trigger Expression is supported.

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 175.

Changes in Reveal Inserter means 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 and access to signals and memory. Try to give yourself capacity to define all the triggering events that you might want to analyze later on.

Table 175: 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		✓
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 631
- “Setting Up Trigger Expressions” on page 634
- “Setting Trigger Options” on page 641
- “Setting Up the Trigger Signals” on page 666

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. 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 using 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 618 and “Searching for Signals” on page 619.
   - Double-click in the Signals column to open the TU Signals dialog box. See “Selecting and Ordering Trigger Signals” on page 632.
3. To change the order of the signals or to remove them, double-click in the **Signals** column.

   The TU Signals dialog box opens. See “Selecting and Ordering Trigger Signals” on page 632.
4. Click in the **Operator** column and choose an operator from the pulldown menu. See “About Trigger Unit Operators” on page 632.

   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 pulldown 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 674.
6. Enter the comparison value in the **Value** column. The form of the value must match the specified radix. If you’ve selected a token set in the Radix

column, a pulldown 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 side of the TU Signals dialog box, select signals that you want to use in the trigger unit. Click on a signal to select it. To select more than one signal, control-click on each one. To select all signals and buses in a range, click on the first signal to select and Shift-click on the last one.
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.

Operators can be changed in Reveal Analyzer, with the exception of “serial compare”.

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 Operators 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 is 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] is high, but did not rise.

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 second clock, a **1** on the third clock, and a **1** on the last clock. Only after those four conditions are met in the 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 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 631

Setting Up Trigger Expressions

You can set up the initial trigger expressions in Reveal Inserter and 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. Click **Add** in the Trigger Expression section of the Trigger Signal Setup tab to add a new trigger expression.
2. Enter the trigger expression in the **Expression** column. See "Trigger Expression Syntax" on page 635.

   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 is needed. Choose EBR if available. If 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 it 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 it 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.

The Max Event Counter setting can only be changed it in Reveal Inserter.

See Also

- "Example Trigger Expressions" on page 638

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 core 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 636 for more information on THEN statements.
- NEXT – Creates a sequence of wait conditions similar to 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 636 for more information on NEXT statements.

- # (count) – Inserts a counter into a sequence. See "Sequences and Counters" on page 636 for information on counters.
- ## (consecutive count) – Inserts a counter into a sequence similar to 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 636 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.

The following 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."

Here's the same example 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
    ```
  - 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 638

Example Trigger Expressions

The following is a series of examples that demonstrate the flexibility of trigger expressions.

Example 1: 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-

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-

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-

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-

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:

1. True twice
2. Wait for TU3 to be true
3. Wait for TU4 to be true five times
4. 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 to know an event you want to capture occurs 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 635

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 in Reveal Inserter but not 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** pulldown menu, choose the maximum size of the count of all the trigger expression outputs combined. You can choose powers of two 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** from the bottom of the Trigger Signal Setup view.
2. In the Net pulldown 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 on another module and can connect to an I/O pin.
3. If you want to change the signal's name, double-click the desired 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.

Note:

For soft IP or single core devices, only IO is selectable in the NET pulldown menu.

Setting Up Virtual Switch/LED Settings

The data pane consists of Virtual Switch/LED panels in which you drag and drop the RTL-defined switch and LED signals. A maximum of 32 bits can be selected for both Virtual Switch/LED signals.

To set up Virtual Switch/LED signals:

1. Select the top_Controller core in the Dataset view. (There can only be one Controller core.)
2. Click the **Virtual Switch & LED Setup** tab.
3. Select signals in the Design Tree by expanding the hierarchy or doing a name search in the Signal Search text box and clicking on Search. For help finding signals, see "Searching for Signals" on page 619.
4. Drag the desired signals to the Virtual Switch or LED data panels to the right. The width of the signals will automatically update between 1-32 entries at the top.

5. Beside the header title, Virtual Switch/LED Setting, there is a check box in which the Controller function for Virtual Switch/LED can be enabled/ disabled.

Note:

In order for this tab to display the Switch/LED names correctly and for the compilation tool flow to run through successfully, please use the **Reveal Controller template** from the Source Template Editor tool. See "Using Templates" on page 116.

Selecting and Ordering Virtual Switch/LED Signals

To select and order virtual switch/LED signals:

- In the Switch List and LED List tables, click on a signal to select it. To select more than one signal, control-click on each one. To select all signals and buses in a range, click on the first signal to select and Shift-click on the last one.
- To move selected signals up and down the list, use the up and down arrows   .

Note

Reveal controller allows you to re-order signals on the virtual LEDs and Switches.

- To remove signals from the trigger unit, select them and click the Delete icon  . You can also press the Delete key.
- To order the signals from the most significant bit (MSB) to the least significant bit (LSB) and vice versa, click the **MSB** and **LSB** buttons.

Configuring User Memory Setup

Setting up the User Memory setting consists of assigning six mandatory ports (which must be defined in the user RTL design). The maximum data width is 32 bits.

The user design must include a memory block (EBR, Distributed Memory or PMI Memory) to be able setup User Memory.

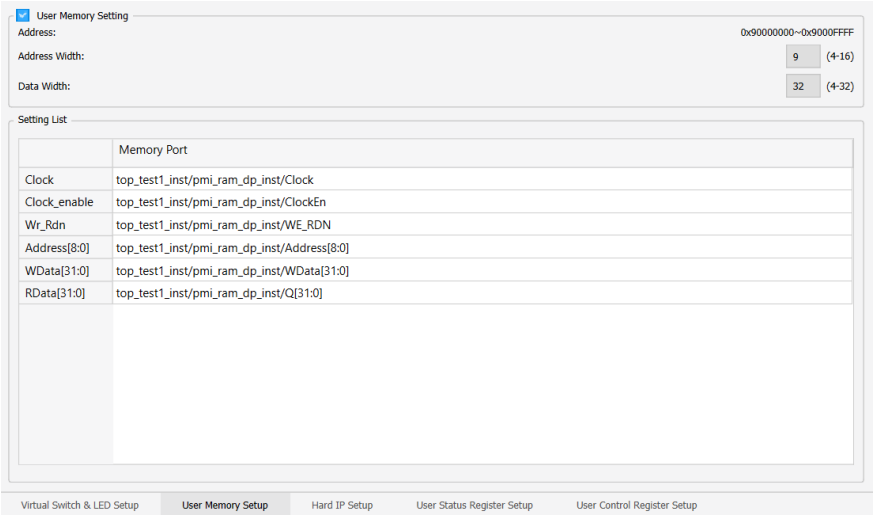
The mandatory signals in the user design to set up User Memory are:

- Clock
- Clock_enable
- Wr_Rdn
- Address
- WData

- RData

To set up User Memory signals:

1. To select the top Controller core in the Dataset view, click on it. There can only be one Controller core. When you click on Add Controller, the control panel will appear. You can enable User Memory and connect to any lower-level hierarchy under the top-level, provided that all the ports are connected without any unconnected ports before and by selecting the corresponding memory port names. The Address and Data Width will be automatically updated.
2. Reveal Controller will add a mux between user logic and the Reveal Controller, which can be left blank or connected to the user logic. Once checked, click the **User Memory Setup** tab.



3. Select signals in the Design Tree and drag the signals to the desired position under Setting List in the User Memory Setup tab. As the pre-synthesis controller debug, no need to add any floating wires. For help finding signals see, “Searching for Signals” on page 619.
4. Drag the desired signals into the User Memory data panel to the right. You do not need to provide a clock if there is no user logic connected while the width of the signals will automatically update between 4-32 entries at the top.
5. Beside the title User Memory Setting, there is a check box in which the Controller function for User Memory signals can be enabled/disabled. If there is a noticeable reset port, it should be connected to an inactive reset signal so that it does not impact the controller functionality and it should not be in a floating state.

Configuring User Control Register Setup

The User Control Register Setup tab displays a table-like panel where you can enable the Control Register. The Control Register address ranges from 0x81000000 up to 0x810000f8.

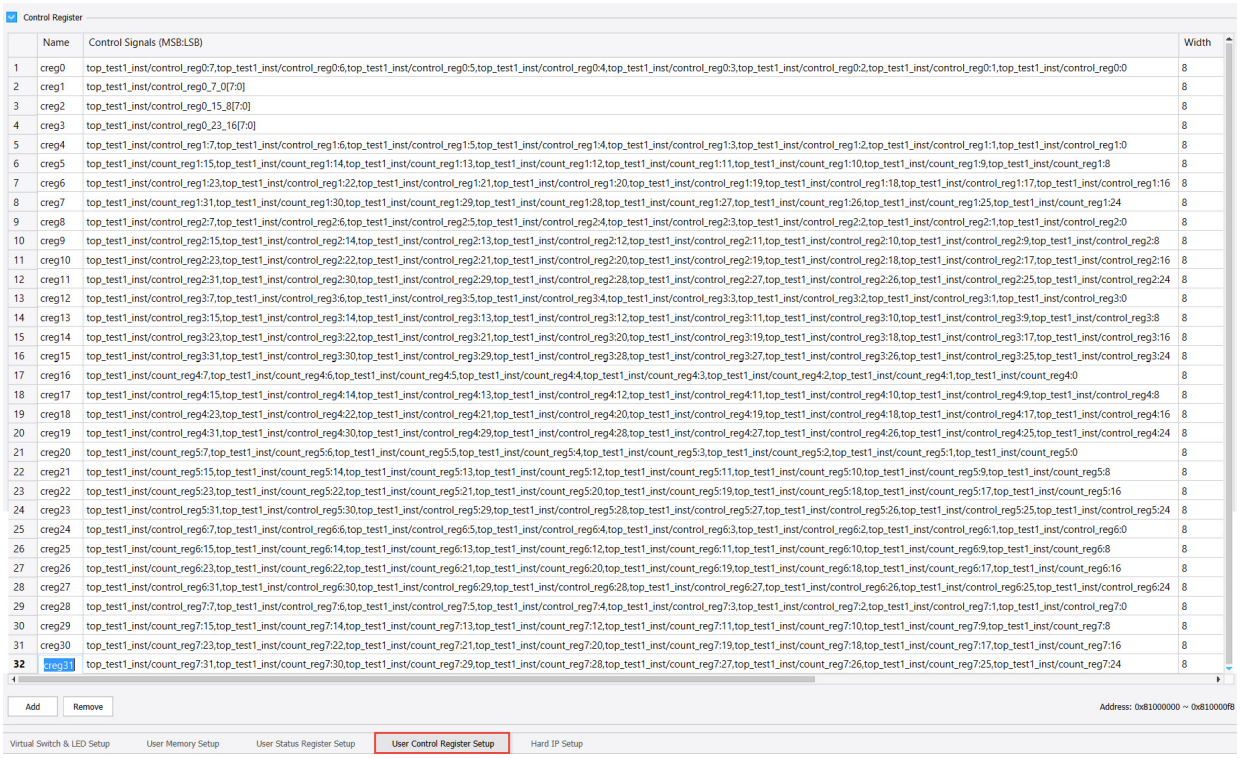
You can add up to 32-address locations with a maximum data-width of 8-bit. If the signal has a data width greater than 8, it splits into 8-bit per address location with a unique label.

The user design must include the control registers as wires, which should not have any other drivers. They should be treated as asynchronous switches as they are changed with the JTAG clock.

These signals are read-writable from the controller.

To set up User Control Register:

1. Click the **User Control Register Setup** tab.



2. Click the **Add** button. By default, a unique label is assigned to the Name column. You can edit this label by double-clicking it.
3. Drag-and-drop the corresponding signals from Design View to the **Control Signals** column.

   Or

4. Select the existing row and double-click the Control Signals column.

   The **Edit Signals** dialog box appears.

5. In the Edit Signals dialog box, do the following:
   a. Choose signals from the **Select Signals** pane.
   b. Click the forward > button to move the signals to the right-hand pane.
   c. Change the order of the signals as desired.
   d. Click **OK**.
6. If you wish to delete a row from the Control Signals column, click the **Remove** button to delete the selected row.

Configuring User Status Register Setup

The User Status Register Setup tab displays a table-like panel where you can enable the Status Register. The Status Register address ranges from 0x81001000 up to 0x810010f8.

You can add up to 32-address locations with a maximum data-width of 8-bit. If the signal has a data width greater than 8, it splits into 8-bit per address location with a unique label.

The user design must include status registers as wires, which should be driven by registers to show the status of the different functional blocks of the design. These signals are read-only.

To set up User Status Register:

1. Click the **User Status Register Setup** tab.



2. Click the **Add** button. By default, a unique label is assigned to the Name column. You can edit this label by double-clicking it.
3. Drag-and-drop the corresponding signals from Design View to the **Status Signals** column.

   Or
4. Select the existing row and double-click the Status Signals column.

   The **Edit Signals** dialog box appears.
5. In the Edit Signals dialog box, do the following:
   a. Choose signals from the **Select Signals** pane.
   b. Click the forward > button to move the signals to the right-hand pane.
   c. Change the order of the signals as desired.
   d. Click **OK**.
6. If you wish to delete a row from the Status Signals column, click the **Remove** button to delete the selected row.

Configuring Hard IP Setup

The Hard IPs are automatically extracted from the RTL design and displayed in the data pane.

For the Inserter, the starting addresses for the IPs as shown is only for informational purpose. The addresses can be experimented in the Analyzer.

To set up Hard IP:

1. Click the **Hard IP Setup** tab.
2. Select Controller function for Hard IPs for analysis by enabling check boxes.

When implementing Reveal Controller for Hard IPs of PCS and PCIE, the implementation will not remove the user logic but will mux it with Reveal Controller logic. The following information explains this process:

- The user logic will access the LMMI interface and be active in the beginning of design for functions like link state machine. After that, the function needs to be inactive.
- When Reveal Controller gets access to the LMMI registers, it will select the mux during that time only. When there is no Reveal read/write command, it will de-select the mux.

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. Make sure your design project is set up for integrated synthesis if not done already.

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**.

   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 view under Debug Files.
5. Implement the design in the usual way. See “Implementing the Design” on page 311.

   Adding Reveal modules can sometimes cause design implementation to fail. If you have any problems, see “Troubleshooting Design Implementation Errors” on page 649 and “Limitations” on page 621.

Troubleshooting Design Implementation Errors

If the design implementation process fails after inserting a Reveal module, check this section for solutions.

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 some examples.

Example 1:

An example error message 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

Example 2:

In this example error message, the serin signal is a hard route, but 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:

1. 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.
2. Manually look through the design to determine the problem. If you have many signals, this approach would be the best.

Example 3:

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 621

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:

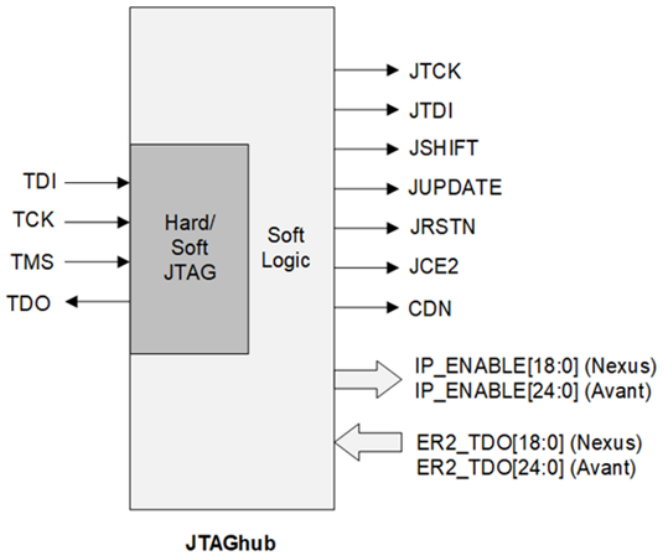
1. In the File List view, right-click the .rvl file.
2. Do one of the following:
   - To remove the Reveal modules but keep the project, choose **Set as Inactive**.
   - To delete the Reveal project, choose **Remove**.
3. Implement the design in the usual way. See "Implementing the Design" on page 311.

Using JTAGhub

A JTAGhub is a module that contains a JTAG core, which is either hard or soft, and around it is soft logic. A hard JTAG core wrapped by soft logic is a hard JTAGhub. A soft JTAG RTL core wrapped by soft logic is a soft JTAGhub.

In Nexus, a hard JTAGhub is referred to as JTAGH19 and a soft JTAGhub is called JTAGH19SOFT. In Avant, a hard JTAGhub is referred to as JTAGH25 and a soft JTAGhub is called JTAGH25SOFT. In pre-synthesis, you can have a maximum of 19 connected cores in Nexus and 25 connected cores in Avant

Figure 36: JTAGhub Architecture



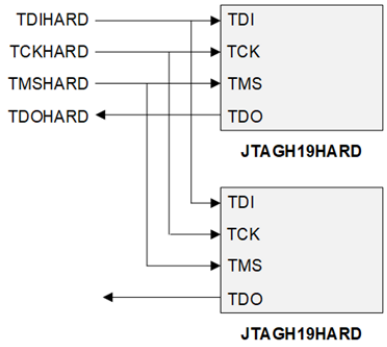
There can only be one hard JTAG in a design. So if there are multiple hard JTAGhub instances, all these are merged by the post-synthesis utility. Multiple soft JTAGhubs, with or without merging, are acceptable.

A JTAGhub, either soft or hard is inserted as a black-box by Reveal Inserter for normal debugging and treated as a primitive for synthesis. It is added to the synthesis header file as a black-box. The logic is not optimized by the synthesis tool. The user can also manually instantiate the JTAGhub in the design.

Hard JTAGhub – Merged

If there is already one hard JTAGhub in the design and the user inserts another hard JTAGhub in Reveal, then the two hard JTAGhubs are merged.

Figure 37: Hard JTAGhub - Merged



On the other hand, there can be multiple soft JTAGhub instances in a design. These soft JTAGhubs may or may not be merged. Each soft JTAGhub needs four I/O ports and a JTAG cable.

Soft JTAGhub – Merged

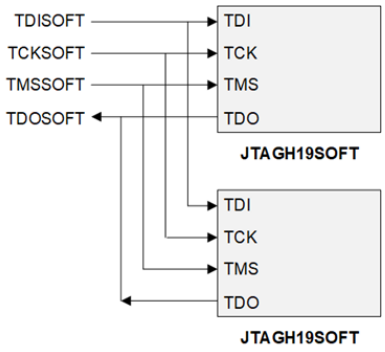
This is an example of two soft JTAG instances that are merged. In this example, all the JTAG ports are connected. TDISOFT, TCKSOFT, TMSSOFT and TDOSOFT are the top-level ports of the design. If the user needs to insert Reveal debugger, then the user must use TDISOFT, TCKSOFT, TMSSOFT, TDOSOFT as the top level port names.

By merging JTAGhubs, only one set of I/O pins is required and only one set of logic is involved. Since merging minimizes logic resources, the design utilizes a minimal number LUTs.

Soft JTAGhub – Not Merged

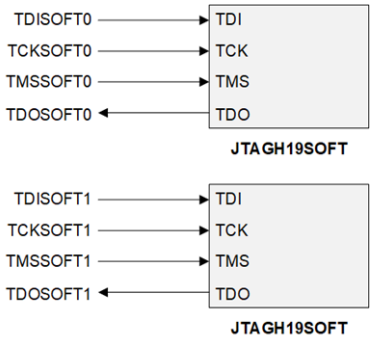
This is an example of two soft JTAGhub instances that are not merged. In this diagram, the TDI, TCK, TMS, and TDO ports are connected to their corresponding top-level ports. The design has two independent sets of soft JTAG ports that are connected to two different cables. Hard JTAG is used for

Figure 38: Soft JTAGhub - Merged



both programming and debugging. Both of these operations can be performed using the same cable and connection. Note that this is not true for soft JTAG.

Figure 39: Soft JTAGhub - Not Merged



Notes

- Designs generated in Lattice Propel can use hard JTAG, soft JTAG, or a mix of the two types, depending on the processor. If there are two processors, two independent debugging GDBs are used. In some devices, however, Propel can use only one soft JTAGhub.
- The JTAG primitive and JTAGH19/JTAGH25 cannot co-exist in a design. Likewise, JTAGA and JTAGH19SOFT/JTAGH25SOFT cannot be added together in one design. These result in an error.

JTAGhub Input and Output Ports

In JTAGhub, the standard TCK, TMS, TDI, and TDO ports are connected to the I/Os and go off-chip. All the other ports are connected to the FPGA fabric. When multiple sets of JTAGhubs are in the same design, all the JTAG ports are unique. Each set connects to one JTAG cable.

Signal	Description
TDI	Data shifted into the test or programming logic of the device.
TCK	Signal that synchronizes the internal state machine operations. In JTAG, TCK is not like a regular clock. It is pulsed when data is transmitted/received.
TMS	Signal sampled at the rising edge of TCK to determine the next state.
TDO	Data shifted out of the test or programming logic of the device.
JTCK	Signal similar to TCK and controlled by IP_ENABLE and ER2_TDO.
JTDI	Signal similar to TDI and controlled by IP_ENABLE and ER2_TDO.
JSHIFT	Signal asserted and de-asserted at the pos-edge of JTCK. It remains high as long as valid data is available in JTDI.
JUPDATE	Signal that gives a positive pulse of one-cycle after JSHIFT is de-asserted. It should be used to capture the shifted-data (JTDI). Ii is asserted and de-asserted at the pos-edge of JTCK.
JRSTN	An active low reset signal. Initially set low and then remains high.
JCE2	Signal that goes high when valid data becomes available in JTDI for the selected core. It remains high as long as the data is valid. It is asserted one clock cycle before JSHIFT is asserted and de-asserted at the same time as JSHIFT.
CDN	Not used
IP_ENABLE[18:0] (Nexus) IP_ENABLE[24:0] (Avant)	Communicates with 19 cores in Nexus and 25 cores in Avant. Each bit enables one core. When one core is enabled, others are disabled. When IP_ENABLE bit and JCE2 and JSHIFT are high, then valid data is shifted in with JTDI and shifted out with corresponding bit of ER2_TDO.
ER2_TDO[18:0] (Nexus) ER2_TDO[24:0] (Avant)	Input data to be shifted out of TDO. Each bit corresponds to equivalent bit of IP_ENABLE.

A soft JTAGhub uses four GPIO ports with locate constraints. Hard JTAGhub pins are fixed in the device.

The TDI, TCK, TMS and TDO ports can be connected to the top-level of the user design.

The user or the tools must connect top-level ports to the TDI, TCK, and TMS of the JTAGhub. The TDO port may be connected to the top-level port for synthesis and may not be connected to the top-level port for simulation to avoid having multiple drivers.

If simulation is not required, it is recommended that the TDI, TCK, TMS input ports be tied to low and the TDO be unconnected.

Notes

- For debug cores inserted by Reveal Inserter, there is no support for simulation.
- For netlists generated in Lattice Propel, it is possible to simulate the user design.
- Encrypted source files for JTAGH19/JTAGH25 and JTAGH19SOFT/ JTAGH25SOFT are provided to compile and run the simulation. In an Avant project, the models will also be available in the OEM simulator.

JTAGhub Ecosystem

In a typical design, several Reveal debug cores and, normally, one Reveal Controller core may be present. There may likewise be user design logic and one or more RISC-V cores. These can all be connected to one JTAGhub and can communicate with the JTAG one at a time.

The JTAG cable connects to the machine with the Debug Analyzer and Reveal Controller tools. To set parameters and connectivity, the user can use the interface or run Tcl commands since these are all standard JTAG protocol commands. The RISC-V GDB debugging protocol can also be included. All these communicate with the cableserver, then the device driver, and onto the device through the JTAG cable.

JTAGhub Usage and Design Examples

This section provides some examples of how JTAGhub is used in a design and recommendations for debugging. A general rule is to avoid using both hard and soft JTAGhubs if using only one cable. With multiple cables, hard and soft JTAGhubs can be mixed in the design.

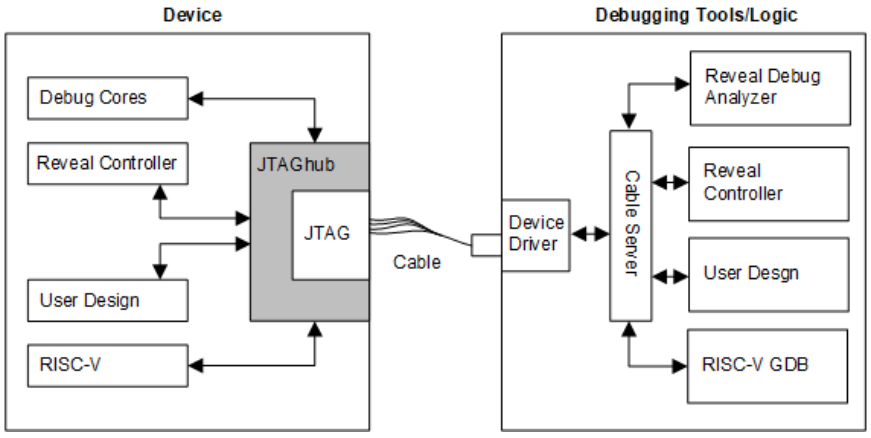
Note: A design with RISC-V does not necessarily come from Propel.

Example 1. The design has no existing JTAGhub

In this case, the user can insert either a hard JTAGhub or a soft JTAGhub.

Example 2. The design contains one RISC-V with hard JTAGhub

Figure 40: JTAGhub Ecosystem



In this case, the user should use hard JTAGhub for Reveal debugging with just one cable. If soft JTAG is used for Reveal debugging, another cable is required.

Example 3. The design contains one RISC-V with soft JTAGhub

In this case, the user should use soft JTAGhub for Reveal debugging. The soft JTAGhubs are merged and only one cable is required.

Example 4. The design contains two RISC-V cores with hard JTAGhub

In this case, the user should use hard JTAGhub for Reveal debugging. The hard JTAGhubs are merged and only one cable is required.

Example 5. The design contains two RISC-V cores with soft JTAGhub

In this case, the user should use soft JTAGhub for Reveal debugging.

Example 6. The design contains one RISC-V with hard JTAGhub and the user instantiates another one hard JTAGhub

In this case, the three hard JTAGhubs are merged and the user should use hard JTAGhub for Reveal debugging.

Example 7. The design contains one RISC-V with soft JTAGhub and the user instantiates another one soft JTAGhub

In this case, the three soft JTAGhubs are merged and the user should use soft JTAGhub for Reveal debugging.

Example 8. The design contains multiple RISC-V cores with independent soft JTAGhubs

In this case, the user must use multiple cables, one for each RISC-V processor and its associated soft JTAGhub.

JTAGhub Address Scheme

The following addressing scheme is used to address the individual cores from the Cableserver clients (Reveal Debug, Controller, RISCV GDB, User Instance - Avant). If the user manually instantiates JTAGhub in the design, it is recommended to use only CORE-17 and CORE-18.

The address length is 24 bits for Nexus and 32 bits for Avant.

Address	Description
Nexus	
Bit[23]	Bit[23]
Bit[22:4]	19 bits for IP_ENABLE[18:0], one-bit for each core. To enable a core, the IP_ENABLE bit of that core is set high and other IP_ENABLE bits set low.
Bit[3]	Reserved bit for future use to clear data.
Bit[2:0]	3-bit control, value of 6 means read
Avant	
Bit[29]	Bit[29]
Bit[28:4]	25 bits for IP_ENABLE[24:0], one bit for each core. To enable a core, the IP_ENABLE bit of that core is set high and other IP_ENABLE bits set low.
Bit[3]	Reserved bit for future use to clear data
Bit[2:0]	3-bit control, value of 6 means read
Bit[31:30]	2 bits reserved

JTAGhub CORES

Example Commands

To enable read of the ID of a CORE

```
! enable ER1
SIR 8 TDI (32);      // 8-bit data with hex value 32
! set reveal core
SDR 24 TDI (800006);     // 24-bit data with hex value 800006
```

To enable a CORE (CORE0 n this case)

```
! enable ER1
SIR 8 TDI (32);
SDR 24 TDI (000016);
```

This table lists the CORE hex values for Nexus and Avant devices. In Nexus, CORE17 and CORE18 are open to any user instantiation. In Avant, CORE17 to CORE 24 are cores open to any user instantiation.

Nexus	Avant
CORE17 => SDR 24 TDI (200006);	CORE17 => SDR 32 TDI (00200006);
CORE18 => SDR 24 TDI (400006);	CORE18 => SDR 32 TDI (00400006);
	CORE19 => SDR 32 TDI (00800006);
	CORE20 => SDR 32 TDI (01000006);
	CORE21 => SDR 32 TDI (02000006);
	CORE22 => SDR 32 TDI (04000006);
	CORE23 => SDR 32 TDI (08000006);
	CORE24 => SDR 32 TDI (10000006);

Setting Parameters and Connectivity

When Reveal Inserter inserts a core, it sets the HUB_* value as “0b1” and connects the IP_ENABLE bit and the ER2_TDO bit.

Examples:

- For any user instantiation, the settings and connections are as follows for CORE-17

  ```
  parameter HUB_17 = “0b1”;
  ```

  IP_ENABLE[17] is connected to the IP_ENABLE input port of the core.

  ER2_TDO[17] is connected to the ER2_TDO output port of the core.

- For any user instantiation, the settings and connections are as follows for CORE-18

  ```
  parameter HUB_18 = “0b1”;
  ```

  IP_ENABLE[18] is connected to the IP_ENABLE input port of the core.

  ER2_TDO[18] is connected to the ER2_TDO output port of the core.

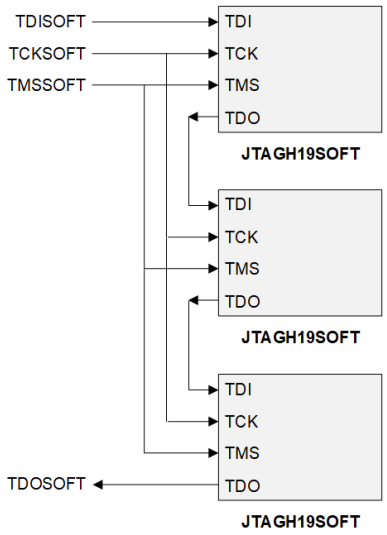
JTAG BYPASS Instruction in SOFTJTAG Chain

JTAG BYPASS instruction allows you to reduce overhead by passing over devices in a chain that do not need to be involved in the current action. These devices remain in functional mode but allow data to flow through to the next device in the chain. When debugging, you can also use BYPASS instruction to isolate what may be offending devices in the chain.

You can build a SOFTJTAG chain of Nexus devices, Avant devices, or a mix of both Nexus and Avant devices. Radiant supports up to 22 devices in a single chain.

The TDO port of the first device is connected to the TDI port of the next device in the chain. The TCK and TMS ports are connected together.

Figure 41: Soft JTAGhub Chain of Three Nexus Devices



Each device with softJTAG interface can be debugged in Reveal when connected in a chain. For example, if multiple devices are connected in a chain, a specific target device can be selected to be debugged. You can mix hardJTAG and softJTAG interfaces as long as they are connected in a chain. The softJTAG interface only has access to the FPGA fabric and that is how debugging is done.

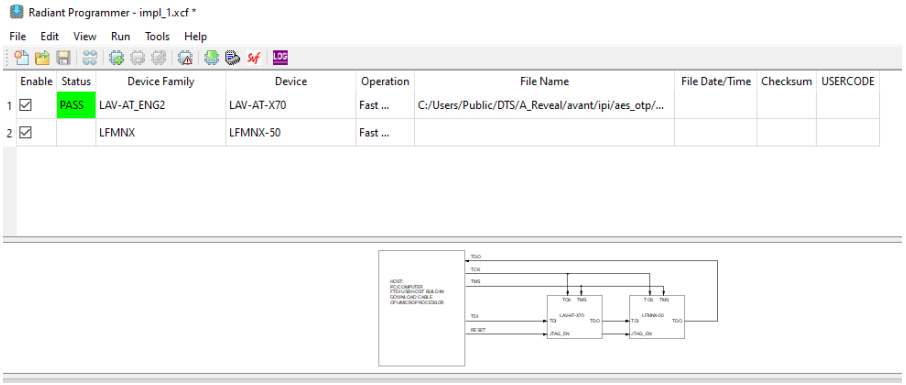
Note:

The softJTAG interface does not have access to configuration logic, so there are no programming and scanning functions.

To scan debug devices in a chain with hardJTAG and softJTAG interfaces:

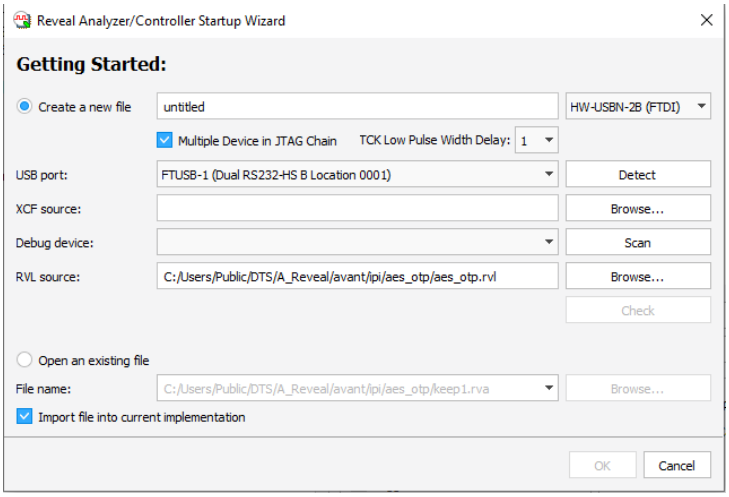
1. Scan the devices in Radiant Programmer tool. The hardJTAG interface will be used in scanning these devices. Open Radiant Programmer and choose **Run > Scan Device**. The tool scans and lists the devices in a new .xcf file.
2. If there are devices connected in the chain using softJTAG interface, manually add these devices using **Edit > Add Device.**

Figure 42: Manually Adding Device



3. Create the .xcf file. This .xcf file will be common for all the Reveal projects.
4. Open Reveal Analyzer and create a new project.
5. In the Reveal Analyzer/Controller Startup Wizard, select the **Multiple Device in JTAG Chain** option.

Figure 43: Multiple Devices in JTAG Chain Option



6. In the XCF source row, click **Browse** to locate and load the .xcf file that you created in Step 2.

7. In the Debug device row, click **Scan** to find the FPGA device.

Note:

Always load the .xcf source file before running scan. If you do not specify the .xcf source file, there will be an error message.

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.

To perform logic analysis:

1. If you are 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 661.
2. Connect your evaluation board and program the FPGA. See "Programming the FPGA" on page 662.
3. Start Reveal Analyzer by selecting **Create a new file** in the Reveal Analyzer Startup Wizard. See "Starting Reveal Analyzer" on page 662.
4. Set up the trigger signals for each module that you want to use. See "Setting Up the Trigger Signals" on page 666.
5. Capture data. See "Capturing Data" on page 682.
6. View the resulting waveforms for each module. See "Viewing Waveforms" on page 683.
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 689.

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 662.
2. View the waveforms. See "Viewing Waveforms" on page 683.
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 689.

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. However, if your evaluation system is in a secured lab that is off the network, 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 files you need to run Reveal Analyzer in the lab are:

- Bitstream (.bin or .hex) file produced by the Export Files stage of the design implementation process. This file is required to program the FPGA.
- 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.

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.

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.

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 690.

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 pulldown 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 418

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 the Radiant software 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.The following steps can also be seen in Figure 44

To start Reveal Analyzer with a new file:

1. Issue the start command. To start:
   - For integrated with the Radiant software, go to the Radiant software main window and choose **Tools > Reveal Analyzer**.
   - For stand-alone in Windows, go to the Windows Start menu and choose **Programs > Lattice Radiant Reveal > Reveal Logic Analyzer**.
   - For stand-alone in Linux, go to a command line and enter the following:

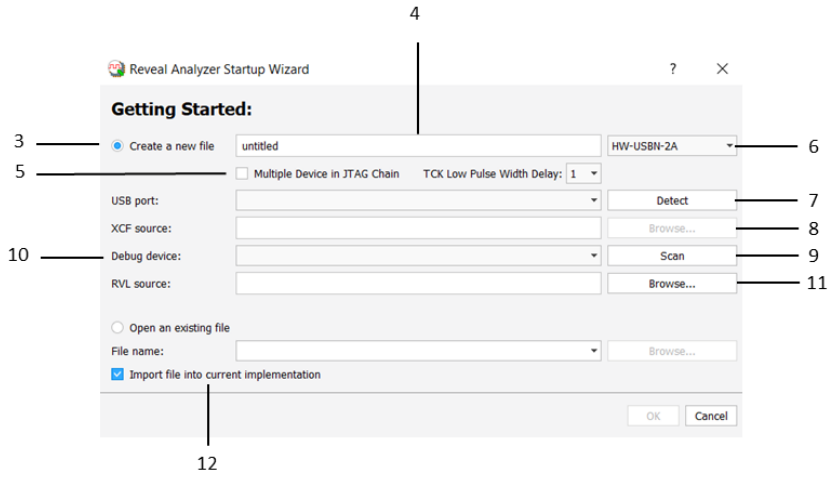
     <Reveal install path>/bin/lin64/**revealrva**

   The Reveal Analyzer Startup Wizard dialog box appears.
2. If Reveal Analyzer opens with an existing file, choose **File > Save 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).

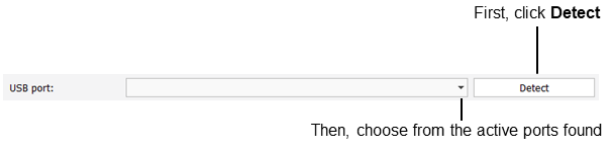
   The dialog box presents a few rows of boxes that need to be filled in. In the figure below, the numbers match up with the steps in this procedure and show where the boxes, buttons, and menus are for each step.

Figure 44: Startup Wizard with Steps for a New File

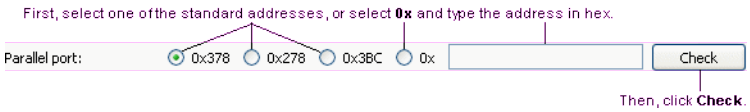


4. In the first second row, type in the base name of the file. The extension is added automatically.
5. If there are daisy-chained devices, select **Multiple Device in JTAG Chain**.
6. To the right of this row is a pulldown menu. Choose the type of cable that your board is connected to.

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 26

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 Radiant software main window, choose **Tools > Reveal Analyzer**.

- The stand-alone Reveal Analyzer in Windows, go to the Windows Start menu and choose **Programs > Lattice Radiant Reveal > Reveal Logic Analyzer**.
- The stand-alone Reveal Analyzer in Linux, enter the following on a command line:

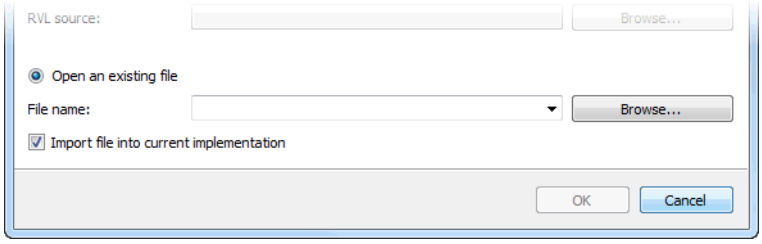
  <*Reveal install path*>/bin/lin64/**revealrva**

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, elect **Open an existing file** (in the lower part of the dialog box). See Figure 45.

Figure 45: 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 665.

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 664.
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 pulldown 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 pulldown menu. You cannot change or select serial compare in Reveal Analyzer. See "About Trigger Unit Operators" on page 667.
   - Radix of the value: Choose a radix from the pulldown 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 pulldown 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 669.

5. In the Trigger Expression Name column, select the trigger expressions that you want to use in the next test.
6. In the POR Debug section, if you wish to connect POR when Analyzer opens, click the **Connect POR when Analyzer opens** check box.
7. 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.
8. 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, and 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.
9. 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.
10. 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.

It should be noted that for each of the Pre-, Center- and Post-Trigger samples, a user is only guaranteed to see the waveforms from that point on. For example if you choose Post-Trigger (15/16), then a user will only see the sample waveforms for the last 1/16 of 256 sample data.

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 Operators

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 have 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 clock after that, 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 core 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 636 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 636 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 636 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 636 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 at 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, however. 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 672

Example Trigger Expressions

The 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

The 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 Expressions
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 know that an event that you want to capture occurs 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.

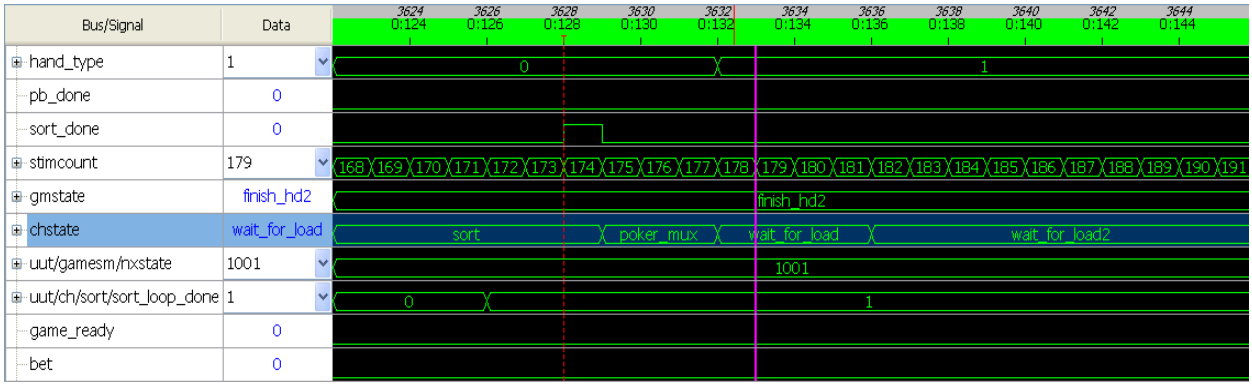
See Also

- "Trigger Expression Syntax" on page 669

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, and Load. Tokens can make reading the waveforms in Reveal Analyzer easier and can highlight the occurrence of key values. See Figure 46 for an example. The row for the chstate bus uses tokens.

Figure 46: 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 (up to 256) in bits. 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 as 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 and 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.

Debugging with Reveal Controller

Reveal Controller is another divide-and-conquer mechanism for you to emulate an otherwise unavailable environment for power debug. For example, your evaluation board would only have a limited number of LEDs or switches but the virtual environment enables up to 32 bits. Register memory mapping and dumping of values is also easily manifested while visibility into Hard IPs is also enabled.

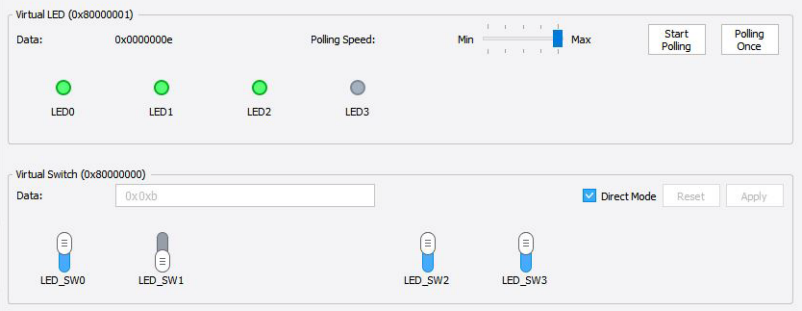
Note

It should be noted that during a debug session with Controller, the constant polling of the signals ties up the JTAG ports for other operations such as attempting to program an updated bitstream as a parallel operation. You must end the debugging session first.

Virtual LED/Switch Console

Once Reveal Analyzer is set up and started, you are presented with three tabs, the first of which is a virtual LED/Switch console to emulate the board.

Figure 47: Virtual LED/Switch



The top section, Virtual LED, shows all the LEDs that were defined in the insertion stage. Once the board is running, you can see the LEDs flash in red and green as they would in the real hardware environment. You can use the slider to adjust polling speed.

The lower section shows the virtual switch in which you can either enter the data as a hex value or can set the virtual dip switches.

Select Direct Mode to see the data and switches in real time instead of waiting until the Apply button is clicked.

Note

Switches are of type in/out. When defining switches in RTL, use the WIRE definition otherwise multiple input driver errors will be encountered later in the flow.

User Memory Analysis

The second tab is for User Memory Analysis. The title indicates the range of the User Memory map. Analysis follows no particular steps or order.

- Default Data: A value that will initialize all memory locations.
- Write Address/Data: Address and data in hex to be written.
- Read Address/Data: Enter address. The data is read back when 'Read' is selected.
- Memory File: You can dump from/to a range of memory addresses to a .mem file. A user can also **Load MemFile** to load a pre-configured memory file.

Running User Control Register

1. In the **User Control Register** tab, select a label from the **Name** column, then click the **Rd** button.

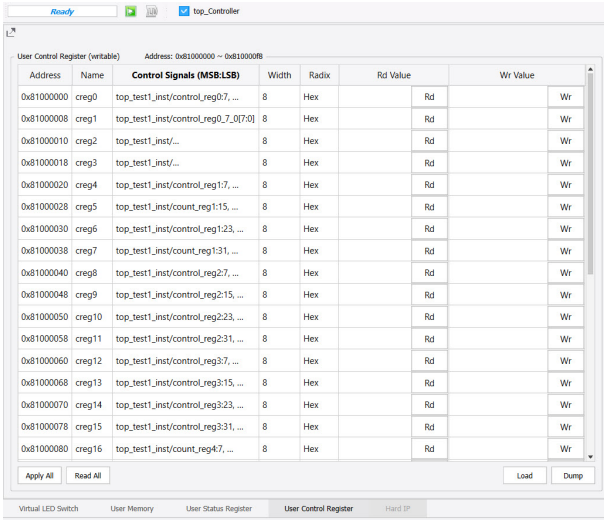
   User Control Register reads the value of the corresponding address location and executes the following TCL command:

   ```
   rva_run_controller -read_control "label"
   ```

2. Select a label from the Name column.
3. In the **WR Value** column, do the following:
   a. Enter desired value.
   b. Click the **Wr** button.

      User Control Register executes the following TCL command:

      ```
      rva_run_controller -write_control "label" -data value
      ```



4. Double-click a row in the **Control Signals** column to display the list of signals with their corresponding value.

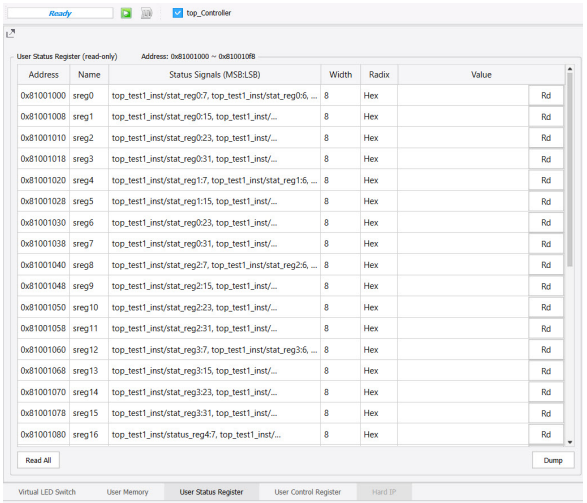
5. In the User Control Register tab, click the **Apply All** button to apply the value of all address locations assigned by the Name column and execute multiple TCL commands sequentially.
6. Click the **Read All** button to read the value of all address locations assigned by the Name column and execute multiple TCL commands sequentially.
7. Click the **Load** button to load the value from the register file to all address locations of Control Register.
8. Click the **Dump** button to save the value of all address locations to a register file.

Running User Status Register

1. In the **User Status Register** tab, select a label from the **Name** column, then click the **Rd** button.

   User Status Register reads the value of the corresponding address location and executes the following TCL command:

   ```
   rva_run_controller -read_status "label"
   ```



2. On the bottom left corner, click the **Read All** button.

   The Read All button reads the value of all address locations assigned by the Name column and executes multiple TCL commands sequentially.
3. On the bottom right corner, click the **Dump** button.

   The Dump button saves the value of all address locations to a register file.
4. Double-click a row in the **Status Signals** column to display the list of signals with their corresponding value.

Hard IP Debug

All the IP selected for analysis in Reveal Inserter are displayed here.

Main operations are reading/writing to memory locations.

Note

The detail settings of each IP is beyond the scope of this help. Links will be provided to Application Notes to explain its usage.

Running Reveal Eye-Opening Monitor

The Reveal Eye-Opening Monitor can be launched from the Reveal Analyzer/ Controller. You need to be connected to a board to be able to run the Eye-Opening Monitor.

To launch the Eye-Opening Monitor:

1. In the Radiant software main window choose **Tools** > **Reveal Analyzer/ Controller**

   In the drop-down menu, choose **top_Controller** to display the **Hard IP** tab.

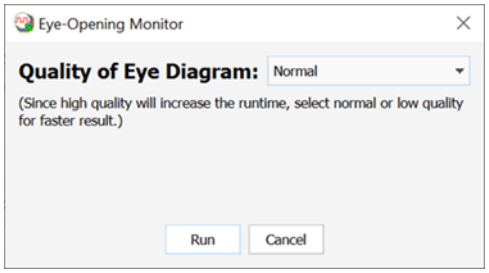
2. Click the **Hard IP** tab.

   If debugging a non-Avant device, scroll down to the PCS channels to locate the **Eye-Opening Monitor** button.

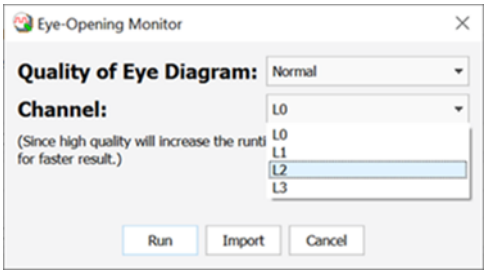
   If debugging an Avant device, the **Eye-Opening Monitor** button is located at the upper-right of the current IP instance.

3. Click the **Eye-Opening Monitor** button. The Eye-Opening Monitor dialog box is displayed.

   The figure below shows the Eye-Opening Monitor dialog box when debugging a non-Avant device.



   The figure below shows the Eye-Opening Monitor dialog box when debugging an Avant device. The Channel and Import options are added.



4. In the Eye-Opening Monitor dialog box, select the quality of the eye diagram from the drop-down menu.

   It is recommended to select normal or low quality for a faster result, and select high quality for the final result. The default is normal quality.

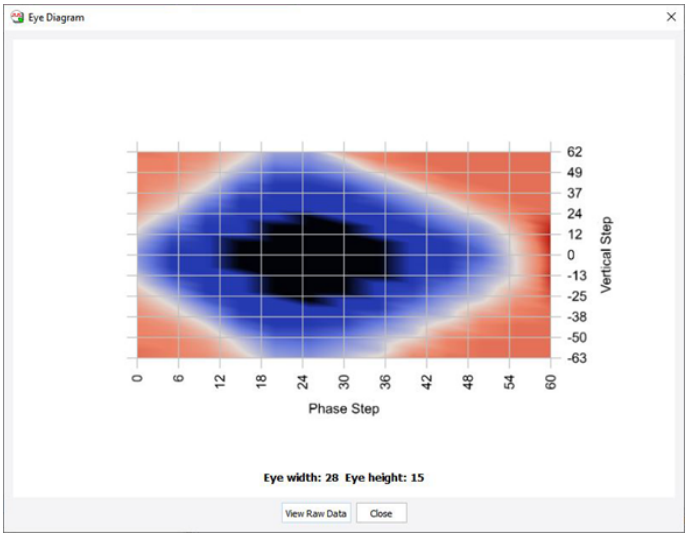
   If you are debugging Avant SerDes multiple channels, select the channel to debug in the Channel drop-down menu.

   By using this function, you can measure the relative performance of a SerDes through comparison, analyze data with regards to the height and width of the eye diagram, and check where the passing eye begins and ends.

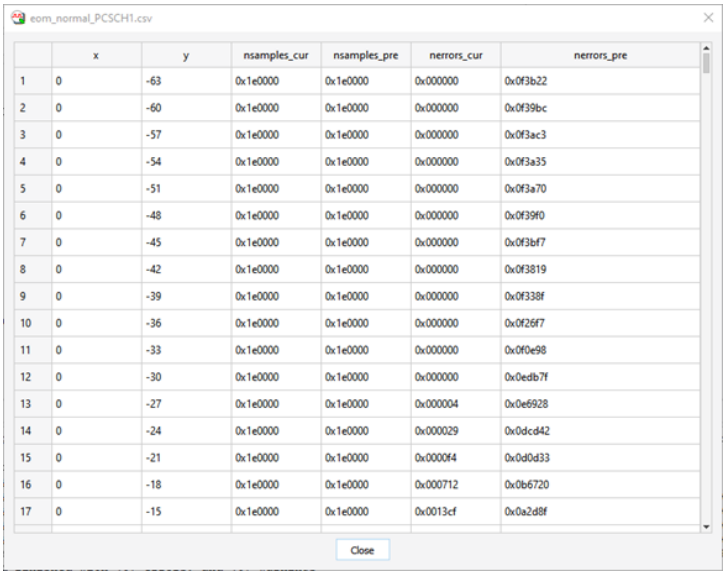
5. Click the **Run** button to start the EOM process. You can stop the process at any time by clicking the **Stop** button.

6. After the EOM has finished running, it will display the eye diagram for the current channel. The figure below shows the Eye Diagram when debugging non- Avant devices.

Figure 48: Example of Eye Diagram for Non-Avant Devices



At the bottom of the eye diagram for non-Avant devices, you can click the **View Raw Data** button to open the raw data file containing the number of samples and errors used in calculating the density of the eye diagram.



	x	y	nsamples_cur	nsamples_pre	nerrors_cur	nerrors_pre
1	0	-63	0x1e0000	0x1e0000	0x000000	0x0f3b22
2	0	-60	0x1e0000	0x1e0000	0x000000	0x0f39bc
3	0	-57	0x1e0000	0x1e0000	0x000000	0x0f3ac3
4	0	-54	0x1e0000	0x1e0000	0x000000	0x0f3a35
5	0	-51	0x1e0000	0x1e0000	0x000000	0x0f3a70
6	0	-48	0x1e0000	0x1e0000	0x000000	0x0f39f0
7	0	-45	0x1e0000	0x1e0000	0x000000	0x0f3bf7
8	0	-42	0x1e0000	0x1e0000	0x000000	0x0f3819
9	0	-39	0x1e0000	0x1e0000	0x000000	0x0f338f
10	0	-36	0x1e0000	0x1e0000	0x000000	0x0f26f7
11	0	-33	0x1e0000	0x1e0000	0x000000	0x0f0e98
12	0	-30	0x1e0000	0x1e0000	0x000000	0x0edb7f
13	0	-27	0x1e0000	0x1e0000	0x000004	0x0e6928
14	0	-24	0x1e0000	0x1e0000	0x000029	0x0dcd42
15	0	-21	0x1e0000	0x1e0000	0x0000f4	0x0d0d33
16	0	-18	0x1e0000	0x1e0000	0x000712	0x0b6720
17	0	-15	0x1e0000	0x1e0000	0x0013cf	0x0a2d8f

- The raw data comes from the eom register.
- The numbers from nsample_cur, nsample_pre, nerrors_cur, and nerrors_pre columns calculate the result.
- The numbers are collected by reading the eom register of the PCS IP.

If you change the Reveal Controller options, you can re-run the EOM to display the new eye diagram and view the new raw data until the desired result is achieved.

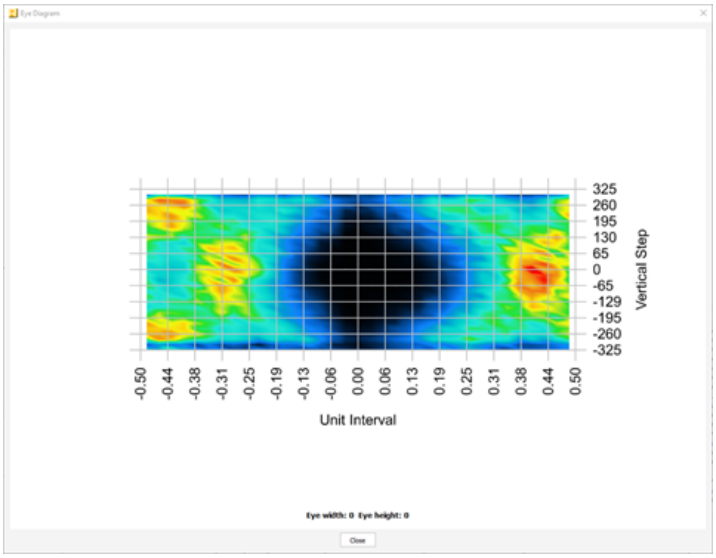
Importing an Existing EOM Result

If you are debugging Avant SerDes, each time you run the EOM, the result is automatically saved as a .txt file in the project folder. The file name provides information on the EOM settings. For example, the file name *eom_normal_MPPHYX4Q1_L2.txt* tells you that the EOM result was generated using normal quality for the MPPHYX4Q1 instance with L2 or lane 2 as the selected channel.

To import an existing EOM result:

1. Click the **Import** button in the Eye-Opening Monitor dialog box.
2. The Select Existing EOM File dialog box is displayed.
   Selecting AVANT EOM (*.txt) displays only the EOM results for debugging the Avant device.
3. Select the .txt file and click **Open.** The Eye Diagram is displayed. The figure below shows the Eye Diagram when debugging Avant devices.

Figure 49: Example of Eye Diagram for Avant Devices



Capturing Data

After you have configured trigger settings in the LA Trigger tab, you can capture data.

Before capturing data, however, your evaluation board must be connected and the design downloaded. See "Programming the FPGA" on page 662.

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 683.

See Also

- "Stopping Data Capture" on page 683

Stopping Data Capture

You can stop capturing data at any time.

To stop data capture:

1. Choose a module from the pulldown 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 pulldown 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.

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:

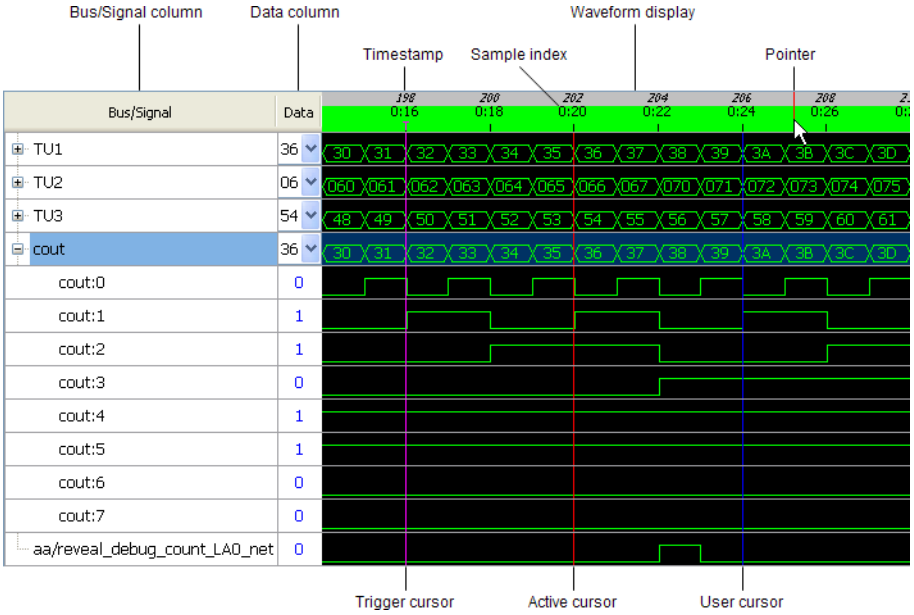
1. Click the **LA Waveform** tab.
2. Choose a module from the pulldown menu in the Reveal Analyzer tool bar.

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 50 along with several features to help you find and analyze the data.

Figure 50: 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 pulldown 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 688.

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 627.
- 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 685.

See Also

- “Zooming In and Out” on page 686
- “Moving around the Waveform” on page 687

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 50 on page 684. 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 685.

Working with User Cursors

You can create any number of user cursors that 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. Do so by right-clicking in the LA Waveform view and choosing **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 Cursors**.

See Also

- “Moving around the Waveform” on page 687

Zooming In and Out

You can zoom in to expand the waveform and see more detail, or 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 687

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 pan by sliding it left and right (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 one of the following 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 685
- "Working with User Cursors" on page 685
- "Zooming In and Out" on page 686

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 radixes 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.

   Click on a bus to select it. To select multiple buses, 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 pulldown menu, choose the radix.
4. In the Range pulldown 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 **Colors > Reveal Analyzer**.
3. Double-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.

Saving the Reveal Analyzer Settings and Data

You can save the waveform data in the following 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 QuestaSim 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. See "Saving a Reveal Analyzer File" on page 689. To save to a different format, see "Saving to Other Formats" on page 690.

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 QuestaSim or Active-HDL, or to an ASCII text (.txt) file.

To export data:

1. Choose the module from the pulldown 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 690.
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**.

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 pulldown 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 is filled in automatically.

   Only integers are allowed. If you try to specify a frequency that requires 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 six and a frequency of 166.

4. Click **OK**.

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 Unified Database (.udb) 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 Radiant software 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 constraints, and memory initialization values. These features enable you to edit the Unified Database (.udb) file created by the Place & Route Design process. You can also sort the sysIO constraints list and save the sorting order. The ECO Editor provides a Change Log window that enables you to track the changes between the modified .udb file and the post-PAR .udb file.

Note

After you have edited your post-PAR UDB file, your functional simulation and timing simulation will no longer match.

ECO memory initialization is not supported directly to use instantiate EBR primitives, such as SP16K/DP16K.

Editing sysIO Settings in ECO Editor

The ECO Editor provides separate sheets for exporting sysIO constraints. After a design is placed and routed, you can open the ECO Editor to export these constraints. The changes you make in the ECO Editor are written into the Unified Database (.udb) file without requiring you to go through the entire design implementation process.

To export sysIO constraints, click the sysIO Setting tab at the bottom of the ECO Editor window.

Editing sysIO Settings

You can use the ECO Editor to export sysIO constraints after the design has been placed and routed.

To export sysIO constraints:

1. In Radiant software, 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, values from design sources are shown in blue, and modified values are shown in black.

   You can only edit the values of cells that have white backgrounds.
3. In the sysIO Setting window, double-click the constraint 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>.udb** to save the changes to the current .udb file.

Setting Memory Initialization Values in ECO Editor

Memory Initialization is a feature that allows you to change initial values to EBRs (Embedded Block RAM) 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 Unified Database (.udb) file without having to repeat the entire flow.

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 Update" on page 694

- "Updating Initial Values for EBRs" on page 694

Memory Initialization Update

Setting initial memory values actually writes the initial values directly into your post-PAR Unified Database (.udb) file. For this to be accomplished, you need to provide the following information:

- Unified Database file (.udb) – the <*design_name*>.udb 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. See

  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 "Applying Engineering Change Orders" on page 692
- 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 pure RTL design, the software can update memory initialization from LSE flow.
- Instance name (not a file) – the name that was given to the memory module in your top-level vm.

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 modules created by LSE flow..

Note

Once you edit initial memory values in the post-PAR UDB file, your functional simulation and timing simulation results will no longer match. You might need to preserve an original version of your UDB file.

To update memory initialization values:

1. Customize a memory initialization file (.mem) in any ASCII text editor. 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 Radiant software 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 parameters for memory modules that were created by LSE flow.
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 an LIFCL 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.
7. 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.
8. Click **OK**.

   The Update Initial Memory dialog box closes after a successful initialization. If the initialization fails, the dialog box will stay open.

9. After the initialization succeeds, choose **File > Save File** to save the memory initialization changes to the current UDB file.

Running Design Rule Check

When you export constraints in the ECO Editor, the software automatically runs design rule check and reports errors, if any, in the Radiant software Output view.

You can also manually run the design rule check by choosing **View > Run DRC**  .

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 or Lattice Synthesis Engine (LSE) options).

For detailed information on how to apply strategies to your project, see “Using Strategies” on page 43.

Constraint Propagation

Constraint Propagation strategy options are listed below:

“Propagation Debug Mode” on page 703

Synplify Pro Options

Synplify Pro strategy options are listed below:

- “Allow Duplicate Modules (for Synplify Pro)” on page 703
- “Area (for Synplify Pro)” on page 703
- “Arrange VHDL Files” on page 704
- “Auto Compile Point” on page 704
- “Automatic Read/Write Check Insertion for RAM” on page 704

- “Clock Conversion” on page 704
- “Command Line Options (for Synplify Pro)” on page 704
- “Default Enum Encoding” on page 705
- “Disable IO Insertion (for Synplify Pro)” on page 705
- “Export Radiant Software Settings to Synplify Pro GUI” on page 705
- “FDC Constraint Files” on page 706
- “FSM Compiler (for Synplify Pro)” on page 706
- “Fanout Guide” on page 706
- “Force GSR (for Synplify Pro)” on page 706
- “Frequency (MHz) (for Synplify Pro)” on page 707
- “Library Directories” on page 707
- “Number of Critical Paths (for Synplify Pro)” on page 707
- “Number of Start/End Points” on page 707
- “Output Netlist Format (for Synplify Pro)” on page 707
- “Pipelining and Retiming” on page 707
- “Push Tristates” on page 708
- “RTL Design Rule Check Option” on page 708
- “Resolve Mixed Drivers (for Synplify Pro)” on page 708
- “Resource Sharing (for Synplify Pro)” on page 708
- “Resynthesize All” on page 709
- “Update Compile Point Timing Data” on page 709
- “Use Clock Period for Unconstrained I/O” on page 709
- “VHDL 2008 (for Synplify Pro)” on page 709

Lattice Synthesis Engine (LSE) Options

LSE strategy options are listed below:

- “Allow Duplicate Modules (for LSE)” on page 709
- “Allow Mixed Assignments (for LSE)” on page 710
- “Carry Chain Length” on page 710
- “Command Line Options (for LSE)” on page 710
- “DSP Style” on page 710
- “DSP Utilization” on page 710
- “Decode Unreachable States” on page 710
- “Disable Distributed RAM” on page 711
- “EBR Utilization” on page 711

- "FSM Encoding Style" on page 711
- "Fix Gated Clocks" on page 711
- "Force GSR (for LSE) (Not applicable for iCE40 UltraPlus)" on page 711
- "Ignore Constraint Errors" on page 712
- "Ignore Constraint Errors" on page 712
- "Intermediate File Dump" on page 712
- "Loop Limit" on page 712
- "Macro Search Path (for LSE)" on page 712
- "Max Fanout Limit" on page 713
- "Memory Initial Value File Search Path (for LSE)" on page 713
- "Optimization Goal" on page 713
- "Propagate Constants" on page 714
- "RAM Style" on page 714
- "ROM Style" on page 714
- "RTL Design Rule Check" on page 715
- "Register With High Fanout" on page 715
- "Rules Disabled" on page 715
- "Read Write Check on RAM" on page 715
- "Remove Duplicate Registers" on page 716
- "Remove LOC Properties (for LSE)" on page 716
- "Report Trimmed User Nets (for LSE)" on page 716
- "Resolve Mixed Drivers (for LSE)" on page 716
- "Resource Sharing (for LSE)" on page 716
- "Target Frequency" on page 716
- "Use Carry Chain" on page 716
- "Use IO Insertion" on page 717
- "Use IO Registers" on page 717
- "VHDL 2008 (for LSE)" on page 717
- "Use Partition (for LSE)" on page 680

Post-Synthesis

Post-Synthesis strategy options are listed below:

- "Command Line Options (for Post-Synthesis)" on page 717
- "External Module Files (.udb)" on page 717

Post-Synthesis Timing Analysis Options

Post-Synthesis Timing Analysis strategy options are listed below:

- "Command Line Options (for Post-Synthesis Timing Analysis)" on page 718
- "Number of End Points" on page 719
- "Number of End Points" on page 719
- "Number of Paths Per Constraint (for Post-Synthesis Timing Analysis)" on page 719
- "Number of Paths Per Endpoint (for Post-Synthesis Timing Analysis)" on page 719
- "Number of Unconstrained Paths (for Post-Synthesis Timing Analysis)" on page 719
- "Report Format (for Post-Synthesis Timing Analysis)" on page 719
- "Timing Analysis Options (for Post-Synthesis Timing Analysis)" on page 719

MAP Design Options

MAP Design strategy options are listed below:

- "Command Line Options (for Map Design)" on page 719
- "Ignore Constraint Errors" on page 720
- "Infer GSR" on page 720
- "Report Signal Cross Reference" on page 720
- "Report Symbol Cross Reference" on page 721

MAP Timing Analysis Options

MAP Timing Analysis strategy options are listed below:

- "Command Line Options (for Map Timing Analysis)" on page 721
- "Number of End Points" on page 721
- "Number of Paths Per Constraint (for Map Timing Analysis)" on page 721
- "Number of Paths Per Endpoint (for Map Timing Analysis)" on page 721
- "Number of Unconstrained Paths (for Map Timing Analysis)" on page 721
- "Report Format (for Map Timing Analysis)" on page 721
- "Speed for Hold Analysis" on page 721
- "Speed for Setup Analysis" on page 722

- "Timing Analysis Options (for Map Timing Analysis)" on page 722

Place & Route Design Options

Place & Route Design strategy options are listed below:

- "Command Line Options (for Place & Route Design)" on page 722
- "Disable Auto Hold Timing Correction (for Place & Route Design)" on page 722
- "Disable Timing Driven (for Place & Route Design)" on page 722
- "Impose Hold Time Correction" on page 723
- "Multi-Tasking Node List" on page 723
- "Number of Host Machine Cores" on page 723
- "Pack Logic Block Utility" on page 723
- "Path-based Placement" on page 724
- "Placement Iteration Start Point" on page 724
- "Placement Iterations" on page 724
- "Placement Save Best Run" on page 724
- "Prioritize Hold Correction Over Setup Performance (for Place & Route Design)" on page 724
- "Run Placement Only" on page 725
- "Set Speed Grade for Hold Optimization (for Place & Route Design)" on page 725
- "Set Speed Grade for Setup Optimization" on page 725
- "Stop Once Setup Timing is Met (for Place & Route Design)" on page 725

Place & Route Timing Analysis Options

Place & Route Timing Analysis strategy options are listed below:

- "Command Line Options (for Place & Route Timing Analysis)" on page 725
- "Number of End Points (for Place & Route Timing Analysis)" on page 725
- "Number of Paths Per Constraint (for Place & Route Timing Analysis)" on page 726
- "Number of Paths Per Endpoint (for Place & Route Timing Analysis)" on page 726
- "Number of Unconstrained Paths (for Place & Route Timing Analysis)" on page 726
- "Report Format (for Place & Route Timing Analysis)" on page 726

- "Speed for Hold Analysis" on page 726
- "Speed for Setup Analysis" on page 726
- "Timing Analysis Options (for Place & Route Timing Analysis)" on page 726

IO Timing Analysis Options

The following option is associated with the IO Timing Analysis process:

- "All Performance Grade" on page 727

Timing Simulation Options

Timing Simulation strategy options are listed below:

- "Generate PUR in the Netlist" on page 727
- "Generate X for Setup/Hold Violation" on page 727
- "Multichip Module Prefix" on page 727
- "Negative Setup-Hold Times" on page 727
- "Retarget Speed Grade" on page 727
- "Timing Check With Min Speed Grade" on page 727
- "Timing Simulation Max Delay between Buffers (ps)" on page 728
- "Verilog Hierarchy Separator" on page 728

Bitstream Options

Bitstream strategy options are listed below:

- "Bitstream Mode" on page 728
- "Command Line Options" on page 728
- "ES Parts" on page 728
- "Enable Early IO Wakeup" on page 728
- "Enable Timing Check" on page 728
- "Enable Warm Boot" on page 729
- "Initialize EBR Quadrant 0" on page 729
- "Initialize EBR Quadrant 1" on page 729
- "Initialize EBR Quadrant 2" on page 729
- "Initialize EBR Quadrant 3" on page 729
- "IP Evaluation" on page 729

- "No header" on page 729
- "Set All Unused IO No Pullup" on page 729
- "Oscillator Frequency Range" on page 729
- "Output Format" on page 730
- "Register Initialization" on page 730
- "Run DRC (iCE40UP only)" on page 730
- "SPI Flash Low Power Mode" on page 730
- "Set NVCM Security" on page 730

Constraint Propagation Options

This page lists all the strategy options associated with the Constraint Propagation process.

Propagation Debug Mode PROP_CPE_IPDebugMode

When set to True, the ignored constraints will be enabled.

Using Propagation Debug Mode can affect timing of other circuits and can lead to false timing reports.

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 *Synopsys Synplify Pro for Lattice Reference Manual*.

Allow Duplicate Modules (for Synplify Pro)
PROP_SYN_EdfAllowDUPMod

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) PROP_SYN_EdfArea

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 (MHz) (for Synplify Pro).

The default is False.

This option is equivalent to the "set_option -frequency 1" command in Synplify Pro.

Arrange VHDL Files PROP_SYN_EdfArrangeVHDLFiles

Allows Synplify Pro to reorder the VHDL source files for synthesis.

The default is True for VHDL or 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 Radiant software File List view.

Auto Compile Point PROP_SYN_Auto_CompilePoint

With Automatic Compile Points, synthesis decides based on existing hierarchy and utilization estimates. The default is True.

Compile Points minimize the amount of netlist changes resulting from RTL code changes. Compile points create logical boundaries where optimization will not occur. This may sacrifice some design performance for design result predictability. No user intervention is needed aside from enabling the flow.

Note:

Auto Compile Point does not support the iCE40UP device.

Automatic Read/Write Check Insertion for RAM
PROP_SYN_RamRWCheck

When this option is set in the strategy, the synthesis tool inserts glue logic around inferred RAM to avoid simulation mismatches caused by indeterminate output values when the read and write addresses are same. By default this option is OFF.

You should design to make sure the read and write addresses are never the same.

Clock Conversion PROP_SYN_ClockConversion

Controls gated and generated clock conversion.

Values are True and False but the default is False.

Command Line Options (for Synplify Pro) PROP_SYN_CmdLineArgs

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 PROP_SYN_EdfDefEnumEncode

(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) PROP_SYN_EdfInsertIO

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
PROP_SYN_DisableRegisterRep

(Not applicable for iCE40 UltraPlus): 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.

Export Radiant Software Settings to Synplify Pro GUI
PROP_SYN_ExportSetting

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 – Synplify Pro opens with its own defaults, ignoring the strategy settings.
- Yes (default) – 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 590.

FDC Constraint Files PROP_SYN_FDCFiles

Specifies the Synopsys FPGA Design Constraint (FDC) files. The FDC file contains Synopsys SDC Standard timing constraints, such as create_clock, set_input_delay, and set_false_path, along with the non-timing constraints (design constraints), such as define_attribute, define_scope_collection, and define_io_standard.

FSM Compiler (for Synplify Pro) PROP_SYN_EdfSymFSM

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.

Fanout Guide PROP_SYN_EdfFanout

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) PROP_SYN_EdfGSR

Forces Global Set/Reset Pin usage.

Available options are:

- False (default) – Does not infer Global Set/Reset in your design.

- Auto – Allows the software to decide whether to 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 (MHz) (for Synplify Pro) PROP_SYN_EdfFrequency

Specifies the global design frequency (in MHz). Nothing in the Value column means "auto" 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.

Library Directories PROP_SYN_LibPath

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)
PROP_SYN_EdfNumCritPath

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 PROP_SYN_EdfNumStartEnd

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) PROP_SYN_EdfOutNetForm

Outputs a mapped VHDL netlist for post-synthesis simulation.

Available options are: None (default) and VHDL.

This option is equivalent to the “set_option -write_vhdl 1 | 0” command in Synplify Pro.

Pipelining and Retiming PROP_SYN_EdfRunRetiming

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 PROP_SYN_EdfPushTirstates

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.

RTL Design Rule Check Option PROP_SYN_RTLDRC

If RTL Design Rule Check is On, Radiant flow runs the DRC step first. If DRC fails, Radiant does not run the synthesis step anymore. If DRC step succeeds, Radiant continues to run the synthesis step as before.

If RTL Design Rule Check is Off, Radiant runs the synthesis step directly without running DRC.

Resolve Mixed Drivers (for Synplify Pro)
PROP_SYN_ResolvedMixedDrivers

If a net is driven by a VCC or GND and active drivers, setting this option to True will connect the net to the VCC or GND driver. The default value is False.

This option is equivalent to the “set_option -resolve_multiple_driver 1 | 0” command in Synplify Pro.

Resource Sharing (for Synplify Pro) PROP_SYN_EdfResSharing

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.

Resynthesize All PROP_SYN_ResynthesizeAll

When this is set to True (default), Synplify Pro will resynthesize all portions, modules, or files of the RTL source. When this is set to False, Synplify Pro will synthesize only the updated portions, modules or files of the RTL source since the last run result.

If you run the “Force Run From Start' process, all portions, modules, or files of the RTL source will be resynthesized, regardless of whether this option is set to True or False.

Update Compile Point Timing Data
PROP_SYN_UpdateCompilePtTimData

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 PROP_SYN_EdfUnconsClk

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.

This option is also limited to a certain device.

VHDL 2008 (for Synplify Pro) PROP_SYN_VHDL2008

When this is set to True, VHDL 2008 is selected as the VHDL standard for the project.

LSE Options

This page lists all the strategy options associated with the LSE synthesis process.

Allow Duplicate Modules (for LSE) PROP_LST_AllowDUPMod

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.

Allow Mixed Assignments (for LSE)

PROP_LST_AllowMixedAssignments

When set to True, having both blocking and non-blocking assignments to the same variable will be allowed. A warning will be issued instead of an error.

Carry Chain Length PROP_LST_CarryChainLength

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.

This option is equivalent to the “-allow_duplicate_modules” option in the SYNTHESIS command.

Command Line Options (for LSE) PROP_LST_CmdLineArgs

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 description on LSE command line options, see “Running Synthesis from the Command Line” on page 1405.

DSP Style PROP_LST_DSPStyle

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 PROP_LST_DSPUtil

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.

Decode Unreachable States PROP_LST_DecodeUnreachableStates

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 PROP_LST_DisableDistRam

When set to True, inferred memory will not use the distributed RAM of the PFUs.

EBR Utilization PROP_LST_EBRUtil

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.

FSM Encoding Style PROP_LST_FSMEncodeStyle

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 implementation.

Note

The encoding type “gray” only works with less than or equal to four machine states. When the number of machine states is larger 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.

Fix Gated Clocks PROP_LST_FIXGATEDCLKS

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. 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) (Not applicable for iCE40 UltraPlus) PROP_LST_ForceGSRInfer

Forces Global Set/Reset Pin usage. (Not applicable for iCE40 UltraPlus).

Available options are:

- No (default) – Does not infer Global Set/Reset in your design.
- Auto – Allows the software to decide whether to infer Global Set/Reset in your design.
- Yes – Always infers Global Set/Reset in your design.

Ignore Constraint Errors PROP_LST_IgnoreSDCError

Enables or disables the LSE synthesis process to ignore constraint errors.

By default, the box is unchecked (False), and LSE terminates processing and issue an error message when constraint errors are encountered.

When the box is checked (True), LSE will ignore constraint errors and continue processing.

The LSE synthesis report includes the setting of this option and the constraint errors found.

Intermediate File Dump PROP_LST_InterFileDump

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 PROP_LST_LoopLimit

Specifies the maximum number of iterations of "for" and "while" loops in the source code before determining potential design issue. If the loop limit is reached and the design loop has not terminated yet, an error will be produced. 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) PROP_LST_EdfInLibPath

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 do not need to specify a search path if the necessary 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:

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 PROP_LST_MaxFanout

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)
PROP_LST_EdfMemPath

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.

Optimization Goal PROP_LST_OptimizeGoal

Enables LSE to optimize the design for area or speed.

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 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 assign 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.

The default setting depends on the device type.

For more information, see "Optimizing LSE for Area and Timing" on page 304.

This option is equivalent to the "-optimization_goal" option in the SYNTHESIS command.

Propagate Constants PROP_LST_PropagatConst

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 PROP_LST_RAMStyle

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.

ROM Style PROP_LST_ROMStyle

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.
- 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.

RTL Design Rule Check PROP_LST_RTLDRC

If RTL Design Rule Check is On, Radiant flow runs the DRC step first. If DRC fails, Radiant does not run the synthesis step anymore. If DRC step succeeds, Radiant continues to run the synthesis step as before.

If RTL Design Rule Check is Off, Radiant runs the synthesis step directly without running DRC.

Register With High Fanout PROP_LINT_RegWithHighFanout

Config fanout limit. Values less than or equal to 0 will imply the default value of 20.

Rules Disabled PROP_LINT_DRCOff

Disable DRC rules by message IDs.

Read Write Check on RAM PROP_LST_RWCheckOnRam

When set to True, LSE inserts additional glue logic to the RAM read/write port to resolve read/write conflicts. The default is False.

A warning message indicating a potential simulation mismatch will be issued if RAM has a read/write conflict.

Remove Duplicate Registers PROP_LST_RemoveDupRegs

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) PROP_LST_EdfInRemLoc

Setting this to On removes LOC properties in the synthesized design.

Report Trimmed User Nets (for LSE)
PROP_LST_ReportTrimmedUserNets

When set to True, LSE will report whenever a user-declared HDL net is trimmed from the netlist.

Resolve Mixed Drivers (for LSE) PROP_LST_ResolvedMixedDrivers

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) PROP_LST_ResourceShare

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.

Target Frequency PROP_LST_EdfFrequency

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 PROP_LST_CarryChain

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 PROP_LST_IOInsertion

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 PROP_LST_UseIOReg

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 772.

VHDL 2008 (for LSE) PROP_LST_VHDL2008

When this is set to True, VHDL 2008 is selected as the VHDL standard for the project.

Use Partition (for LSE) PROP_LST_Partition

When set to True, the -partition synthesis flow is enabled, and the partition synthesis passes through the LSE command option.

Post-Synthesis Options

This page lists all strategy options associated with the Post-Synthesis process. The options available for user setting are dependent on the target device of your project.

Command Line Options (for Post-Synthesis)
PROP_POSTSYN_CmdLineArgs

Enables additional command line options for the associated process.

External Module Files (.udb) PROP_POSTSYN_ExtModuleFiles

Allows the user to supply already synthesized modules for design assembly into the top level design. These modules must be already synthesized in Unified Database (.udb) format.

For example:

c:\case1\ip1.udb; d:\example\aaa\abc.udb

Multiple .udb files can be separated by ';'

This internal process, Post-Synthesis, performed after logic synthesis resolves and assembles lower level modules to top level to complete the design contents.

All modules contents must be resolved at this stage before the flow can continue to the next stage.

Ignore Constraint Errors PROP_POSTSYN_IgnoreSDCErr

Enables or disables the post-synthesis process to ignore constraint errors.

By default, the box is unchecked (False), and post-synthesis terminates processing and issue an error message when constraint errors are encountered.

When the box is checked (True), post-synthesis will ignore constraint errors and continue processing.

The post-synthesis report includes the setting of this option and the constraint errors found.

Hardware Evaluation PROP_POSTSYN_HardTimer

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.

Note:

The Hardware Evaluation option is only for a device that does not support bitstream encryption ability even with a built-in hardware evaluation timer (i.e., LAV-AT device).

Post-Synthesis Timing Analysis Options

This page lists all strategy options associated with the Post-Synthesis Timing Analysis process.

Command Line Options (for Post-Synthesis Timing Analysis) PROP_SYNSTA_CmdLineArgs

Enables additional command line options for the associated process.

Number of End Points PROP_SYNSTA_EndPtNumber

Controls the number of endpoints in the critical endpoint summary.

Number of Paths Per Constraint (for Post-Synthesis Timing Analysis)

PROP_SYNSTA_NumPathsPerClock

Lists detailed logic and route delays for all constrained paths and nets in the design.

Number of Paths Per Endpoint (for Post-Synthesis Timing Analysis)

PROP_SYNSTA_NPerEnd

Controls maximum number of paths that could be reported for each endpoint.

Number of Unconstrained Paths (for Post-Synthesis Timing Analysis)

PROP_SYNSTA_UnconstrainedPathsNumber

Reports paths not covered by a timing preference.

Report Format (for Post-Synthesis Timing Analysis)

PROP_SYNSTA_ReportFormat

Specifies the report format type.

- Lattice Standard – Displays timing report format that is similar to formats used in many industry timing tools.
- Diamond Style – Displays timing report that is similar to Diamond software TRACE report.

Timing Analysis Options (for Post-Synthesis Timing Analysis)

PROP_SYNSTA_AnalysisOption

Specifies the analysis type.

- Hold Analysis – Performs hold analysis.
- Standard Setup Analysis – Performs setup time checks.
- Standard Setup and Hold Analysis – Performs both the Standard Setup Analysis and the Hold Analysis.

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 (for Map Design) PROP_MAP_MapModArgs

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" on page 1413.

Ignore Constraint Errors PROP_MAP_IgnoreSDCErr

Enables or disables the Map Design process to ignore constraint errors.

By default, the box is unchecked (False), and Map Design terminates processing and issues an error message when constraint errors are encountered.

When the box is checked (True), Map Design will ignore constraint errors and continue processing.

The Map Design process report includes the setting of this option and the constraint errors found.

Infer GSR PROP_MAP_MAPInferGSR

(Not applicable for iCE40 UltraPlus): Enables or disables the GSR inferencing.

GSR inference is only applicable to PLC slice registers by default. Each black-box can have its own rules to guide mapper to perform GSR inference. Rules to be applied according to device architecture specification. For example, in LIFCL device, it is applicable to SLICE / IO registers, block RAM, large RAM, and DDR components.

When multiple GSR instances are found in the design, mapper is able to merge them if their outputs are actually the same wire in the netlist. Otherwise mapper will issue an error.

Mapper can also create a GSR component if user specified the GSR signal in the constraint file.

Report Signal Cross Reference PROP_MAP_SigCrossRef

When this is set to True, the map report (.mrp) will show where nets in the logical design were mapped in the physical design.

The default is False.

Report Symbol Cross Reference PROP_MAP_SymCrossRef

When this is set to True, the map report (.mrp) will show where symbols in the logical design were mapped in the physical design.

The default is False.

Map Timing Analysis Options

This page lists all strategy options associated with the Map Timing Analysis process.

Command Line Options (for Map Timing Analysis)
PROP_MAPSTA_CmdLineArgs

Enables additional command line options for the associated process.

Number of End Points PROP_MAPSTA_EndPtNumber

Controls the number of endpoints in the critical endpoint summary.

Number of Paths Per Constraint (for Map Timing Analysis)
PROP_MAPSTA_NumPathsPerClock

Lists detailed logic and route delays for all constrained paths and nets in the design.

Number of Paths Per Endpoint (for Map Timing Analysis)

PROP_MAPSTA_NPerEnd

Controls maximum number of paths that could be reported for each endpoint.

Number of Unconstrained Paths (for Map Timing Analysis)
PROP_MAPSTA_UnconstrainedPathsNumber

Reports paths not covered by a timing preference.

Report Format (for Map Timing Analysis)

PROP_MAPSTA_ReportFormat

Specifies the report format type.

- Lattice Standard – Displays timing report format that is similar to formats used in many industry timing tools.
- Diamond Style – Displays timing report that is similar to Diamond software TRACE report.

Speed for Hold Analysis PROP_MAPSTA_SpeedForHoldAnalysis

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 PROP_MAPSTA_SpeedForSetupAnalysis

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.

Timing Analysis Options (for Map Timing Analysis)
PROP_MAPSTA_AnalysisOption

Specifies the analysis type.

- Hold Analysis – Performs hold analysis.
- Standard Setup Analysis – Performs setup time checks.
- Standard Setup and Hold Analysis – Performs both the Standard Setup Analysis and the Hold Analysis.

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.

Command Line Options (for Place & Route Design)

PROP_PAR_PARModArgs

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” on page 1415.

Disable Auto Hold Timing Correction (for Place & Route Design)

PROP_PAR_DisableAutoHldTiming

When the switch is used, PAR will not check the hold timing of the design, so no correction of potential hold timing violations will be performed.

Disable Timing Driven (for Place & Route Design)
PROP_PAR_DisableTDParDes

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 constraints 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 constraints 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.

Impose Hold Time Correction PROP_PAR_ImposeHoldTimeCorrect

Forces PAR to fix hold time violations even if there are setup time violations.

Multi-Tasking Node List PROP_PAR_ParMultiNodeList

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” on page 327.

Number of Host Machine Cores PROP_PAR_NumOfHostMachineCores

Allows you to run multiple threads on your local machine by specifying how many cores to be used from your local machine. The range is from 0 to 64.

Pack Logic Block Utility PROP_PAR_PackLogicUtil

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 a percentage that depends on which device is selected. For example, for LIFCL, the default is 75 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 placer 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 placer 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 placer will make an

aggressive packing effort to meet your target. However, this may adversely impact the design f_{MAX} performance.

Path-based Placement PROP_PAR_parPathBased

Allows you to apply path-based placement. Path-based placement gives better performance and more predictable results.

Options are Off and On.

Note:

The Pack Logic Block Utility and Path-based Placement strategies are not supported in the LAV-AT device.

Placement Iteration Start Point PROP_PAR_PlcStCostTblParDes

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 PROP_PAR_PlcIterParDes

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 Uniform Database (.udb) file. If you specify a Starting Cost Table, the iterations begin at that table number.

Placement Save Best Run PROP_PAR_SaveBestRsltParDes

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.

Prioritize Hold Correction Over Setup Performance (for Place & Route Design) PROP_PAR_PriHldCorrectOverSetup

During hold timing correction, there may be situations when correcting a hold timing violation would cause a setup requirement to become violated. By default, we do not correct the hold violation on such connections so as to

preserve the setup performance, but when this switch is used, we will attempt to correct the hold violation and let the setup requirement become violated.

Run Placement Only PROP_PAR_ParRunPlaceOnly

Setting this to True prevents the design from being routed. PAR will output a placed, but not routed Unified Design Database(.udb) file.

This option defaults to False.

Set Speed Grade for Hold Optimization (for Place & Route Design)

PROP_PAR_SpdGradeHoldOpt

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.

Set Speed Grade for Setup Optimization

PROP_PAR_SpdGradeSetupOpt

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.

Stop Once Setup Timing is Met (for Place & Route Design)
PROP_PAR_StopZero

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 Timing Analysis report for each iteration" option is set to True or if using Run Manager to produce multiple place-and-route runs.

Place & Route Timing Analysis Options

This page lists all strategy options associated with the Place & Route Timing Analysis process.

Command Line Options (for Place & Route Timing Analysis)
PROP_PARSTA_CmdLineArgs

Enables additional command line options for the associated process.

Number of End Points (for Place & Route Timing Analysis)

PROP_PARSTA_EndPtNumber

Controls the number of endpoints in the critical endpoint summary.

Number of Paths Per Constraint (for Place & Route Timing Analysis)
PROP_PARSTA_NumPathsPerClock

Lists detailed logic and route delays for all constrained paths and nets in the design.

Number of Paths Per Endpoint (for Place & Route Timing Analysis)

PROP_PARSTA_NPerEnd

Controls maximum number of paths that could be reported for each endpoint.

Number of Unconstrained Paths (for Place & Route Timing Analysis)
PROP_PARSTA_UnconstrainedPathsNumber

Reports paths not covered by a timing preference.

Report Format (for Place & Route Timing Analysis)

PROP_PARSTA_ReportFormat

Specifies the report format type.

- Lattice Standard – Displays timing report format that is similar to formats used in many industry timing tools.
- Diamond Style – Displays timing report that is similar to Diamond software TRACE report.

Speed for Hold Analysis PROP_PARSTA_SpeedForHoldAnalysis

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 PROP_PARSTA_SpeedForSetupAnalysis

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.

Timing Analysis Options (for Place & Route Timing Analysis)
PROP_PARSTA_AnalysisOption

Specifies the analysis type.

- Hold Analysis – Performs hold analysis.
- Standard Setup Analysis – Performs setup time checks.
- Standard Setup and Hold Analysis – Performs both the Standard Setup Analysis and the Hold Analysis.

IO Timing Analysis Options

The following option is associated with the IO Timing Analysis process.

All Performance Grade PROP_IOTIMING_AllSpeed

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 PROP_TIM_TimSimGenPUR

When this is set to False, the timing simulation file generation process will not write PUR instance in the Verilog/VHDL back-annotation netlist. Then you have to instantiate PUR in the test bench.

Generate X for Setup/Hold Violation PROP_TIM_TimSimGenX

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.

Multichip Module Prefix PROP_TIM_ModPreSimDes

Adds a prefix to module names to make them unique for multi-chip simulation.

Negative Setup-Hold Times PROP_TIM_NegStupHldTim

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 PROP_TIM_TrgtSpeedGrade

Retargets back annotation to a different performance grade than the one used to create the Uniform Database (.udb) file.

You are limited to those performance grades available for the device used in the UDB file.

Timing Check With Min Speed Grade PROP_TIM_MinSpeedGrade

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)
PROP_TIM_MaxDelSimDes

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).

Verilog Hierarchy Separator PROP_TIM_TimSimHierSep

Specifies the hierarchy separator character which will be used in name generation when the design hierarchy is flattened.

You can specify the following two special characters as the hierarchy separator: “/” (back-slash) or “.” (period).

Enter the character as is in the edit box. Encapsulate the character with double quotes, for example, “/”, “.”.

The option is only available for Verilog designs.

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.

Bitstream Mode PROP_BIT_MODE

Generates bitstream for use in special bitstream update flow.

Command Line Options PROP_BIT_CmdLineArgs

Enables additional command line options for the associated process.

ES Parts PROP_BIT_ES

Generates SMM key per device and introduce a new bitgen flag to specify the ES parts.

Enable Early IO Wakeup PROP_BIT_EnableIOWakeup

Enable IO output as soon as possible. (Not applicable for iCE40 UltraPlus)

Enable Timing Check PROP_TMCHK_EnableCheck

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.

Enable Warm Boot PROP_BIT_WarmBoot

Enables the Warm Boot functionality, provided the design contains an instance of the WARMBOOT primitive. (iCE40UP)

Initialize EBR Quadrant 0 PROP_BIT_INITEBR0

Write the EBR initialization data into the bitstream for quadrant 0. (iCE40UP)

Initialize EBR Quadrant 1 PROP_BIT_INITEBR1

Write the EBR initialization data into the bitstream for quadrant 1. (iCE40UP)

Initialize EBR Quadrant 2 PROP_BIT_INITEBR2

Write the EBR initialization data into the bitstream for quadrant 2. (iCE40UP)

Initialize EBR Quadrant 3 PROP_BIT_INITEBR3

Write the EBR initialization data into the bitstream for quadrant 3. (iCE40UP)

IP Evaluation PROP_BIT_IPEval

When enabled, a bitstream will be generated for evaluation purposes when an IP license is not found, but will be limited to a maximum of four hours before the device resets itself. When disabled, a bitstream will not be generated if an IP license is not found (Not applicable for iCE40 UltraPlus).

All the encrypted and authenticated bitstreams can only be programmed using fast configuration.

The following operations are not supported when using raw data frame by frame in programming:

- Erase, Program, Verify
- Verify Only

No header PROP_BIT_NoHeader

Do not write the bitstream header section. (iCE40UP)

Set All Unused IO No Pullup PROP_BIT_NoPullup

Removes the pullup on the unused I/Os, except Bank 3 I/Os which do not have pullup. (iCE40UP)

Oscillator Frequency Range PROP_BIT_OSCFREQ

Options include Medium and Slow. Fast is not supported for iCE40UP. Only for NVCM configuration, not for programming

Depending on the speed of external PROM, this options adjusts the frequency of the internal oscillator used by the iCE40UP device during configuration.

This is only applicable when the iCE40UP device is used in SPI Controller Mode for configuration.

Output Format PROP_BIT_OutFormatBitGen

Specifies the type of bitstream to create for an FPGA device.

The following options are available:

- Bit File (Binary) – Generates a binary configuration file (.bin) 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.
- Hex File (Headecimal) – Generates a Hexadecimal (.hex) PROM data file used for Programming into external non-volatile memory, such as parallel or Serial Peripheral Interface (SPI) Flash devices (iCE40UP only).
- NVCM File (Non-volatile Configuration Memory) – Generates a Non-volatile Configuration Memory (.nvcm). Each line in this file is a separate NVCM instruction for bitmap programming. The exceptions are comment lines (usually the first few lines in the file) which start with the # sign. Comment lines should be ignored when programming the NVCM array. The comment lines contain comments as well as header information (iCE40UP only).

Register Initialization PROP_BIT_RegInit

Enable register initialization section of the bitstream (Not applicable for iCE40 UltraPlus).

Run DRC (iCE40UP only) PROP_BIT_RunDRCBitGen

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.

SPI Flash Low Power Mode PROP_BIT_SPILowPower

Places the PROM in low-power mode after configuration. (iCE40UP)

This option is applicable only when the iCE40UP device is used as SPI Controller Mode for configuration.

Set NVCM Security PROP_BIT_NVCMSecurity

Ensures that the contents of the Non-Volatile Configuration Memory (NVCM) are secure and the configuration data cannot be read out of the device. (iCE40UP)

Constraints Reference Guide

Constraints are instructions applied to design elements that 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 assignments, but constraints are also available for placement, routing, and many other functions. In the Radiant software design flow, constraints include logical constraints and HDL attributes.

You can assign constraints using one or both of the following methods:

- Assign Lattice design (timing) constraints (.PDC) via the Device Constraint editor tool.

  See the "Device Constraint Editor" on page 228 for complete descriptions of each constraint, including syntax rules and examples.

- Assign Lattice design timing constraints (.LDC) via the "Pre-Synthesis Constraint Editor / Post-Synthesis Timing Constraint Editor" on page 265.

- Assign HDL or schematic-based attributes using design source files. These attributes are used to direct Map and Place & Route.

  See the "HDL Attributes" on page 732 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 Constraints" on page 750 for descriptions of these LSE constraints.

See Also

- "Applying Design Constraints" on page 196
- "Integrated Synthesis" on page 317

HDL Attributes

HDL attributes are constraints that are attached as text to design objects and interpreted by the Radiant software. A design object can be a specific port, component pin, net, instance, instantiation, or even an 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. They generally apply to all designs and are not specific to any architecture.

HDL attributes are typically declared using comment notation in Verilog HDL or the VHDL Attribute keyword..

Note

A comprehensive guide to library element HDL attributes is available in the FPGA Libraries Reference Guide. For specific applications, refer to the technical notes that are available on the Lattice web site. Editing these library element attributes is not recommended, as they are usually generated from an array of choices that are made for module generation using IP Catalog.

BBOX

Convention BBOX= x,x

Description Indicates the bounding box or the area given in number of rows and columns for a given GRP. The attribute must appear on the same block as the GRP attribute.

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 51: Example of a BBOX Parameter

```
BBOX= 3,7
```

You will see BBOX with the GRP definition below it.

CLAMP

Convention CLAMP = OFF | ON

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 CLAMP attribute is not supported for LVCMOS I/O types used in an under-drive or over-drive mode.

Device Support LFCPNX, LFD2NX, LFMXO5, LIFCL, UT24C, UT24CP

Note

The LAV-AT device does not support the CLAMP attribute.

VHDL Syntax ATTRIBUTE CLAMP: string;
ATTRIBUTE CLAMP OF [pin_name]: SIGNAL IS "[mode]";

VHDL Example Code
```
ATTRIBUTE CLAMP OF Q: SIGNAL IS "ON";
```

Verilog Syntax – Synplify PinType [pin_name] /* synthesis CLAMP="[mode]"*/;
[mode] = OFF | ON

Verilog Example Code – Synplify
```
module rtl_top #(parameter width = 8) (DA, DB, CLK, Q);
output [width-1:0]Q/* synthesis CLAMP=”ON”
```

.ldc TCL command
```
ldc_set_port -iobuf {CLAMP=ON} [get_ports Q[1]]
```

DIFFDRIVE

Convention DIFFDRIVE = 2.0, 3.5, NA

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 is set to 2.0mA for MIPI and 3.5 for LVDS.

Device Support LAV-AT, LFD2NX, LIFCL, UT24C

VHDL Syntax ATTRIBUTE DIFFDRIVE: string;
ATTRIBUTE DIFFDRIVE OF [pin_name]: SIGNAL IS "[diffdrive_strength]";

VHDL Example Code `ATTRIBUTE DIFFDRIVE OF portD: SIGNAL IS "2.0";`

Verilog Syntax PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DIFFDRIVE=”[diffdrive_strength]”*/;

Verilog Example Code `output mypin /* synthesis IO_TYPE=”MIPI_DPHY” DIFFDRIVE=”2.0”*/;`

.ldc TCL command `ldc_set_port -iobuf {DIFFDRIVE=2.0} [get_ports portD[1]]`

DIFFRESISTOR

Convention DIFFRESISTOR = OFF (default), 100 (Available only for differential I/O types.)

Description Attribute attached to input and output buffers (e.g., IB, OB) and is used to provide differential termination, which is only available for differential I/O types. The differential output termination is only accessible on the PIO primary pads.

Device Support LAV-AT, LFD2NX, LIFCL, UT24C

VHDL Syntax ATTRIBUTE DIFFRESISTOR: string;
ATTRIBUTE DIFFRESISTOR OF [pin_name]: SIGNAL IS "[value]";

VHDL Example Code `ATTRIBUTE DIFFRESISTOR OF portD: SIGNAL IS "100";`

Verilog Syntax PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DIFFRESISTOR=”[value]”*/;

Verilog Example Code `output mypin /* synthesis IO_TYPE=”LVDS” DIFFRESISTOR=”100”*/;`

.ldc TCL command `ldc_set_port -iobuf {DIFFRESISTOR=100} [get_ports portD[1]]`

DRIVE

Description Attached to bidirectional and output buffers (e.g., BB, OB, OBZ_B, OB_RGB), the drive strength attribute is available for output standards that support programmable drive strength.

The programmable drive available on a pad depends 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.

For more information about the supported values of the DRIVE attribute, refer to the following sysI/O User Guides:

- FPGA-TN-02067, sysI/O User Guide for Nexus Platform
- FPGA-TN-02297, Lattice Avant sysI/O User Guide
- FPGA-TN-02068, MachXO3D sysI/O User Guide

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 – 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"*/;
```

.ldc TCL command
```
ldc_set_port -iobuf {DRIVE=16} [get_ports mypin]
```

See Also

- IO_TYPE

GRP

Convention GRP = <identifier>

Description Universal grouping construct. Use the GRP attribute to group blocks within different hierarchies or with no hierarchy.

You can add GRP anchor and bounding box information to the HDL with BBOX attribute and anchoring it with a SDC constraint. Alternatively, you can allow a GRP to float within a REGION.

Note that RLOC/GLOC (Region Lock/Group Lock) can only be used with REGION or GRP attributes. The value it takes is a site value on the device itself which can be deciphered as such:

RLOC/GLOC = “R5C19D”

i.e. Row 5, Column 19, Slice D

Device Support All

Syntax GRP=”<GRP_name>”

[BBOX=”<height>,<width>”]

where:

<*GRP_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>.

Note:

The SLICE D location is valid for most devices except Avant.

For the Avant device, you must use SLICE F. Avant has 6 SLICEs (A-F), while other devices have 4 SLICEs (A-D).

<height> is the number of device rows of a rectangular bounding box.

<width> is the number of device columns of a rectangular bounding box.

The VHDL and Verilog examples show anchored GRPs. The current mechanism is to define the BBOX and GRP name in the attribute and the anchor location must be set via the SDC constraint ldc_set_location.

VHDL Syntax attribute GRP: string;
attribute GRP of <object>: label is "<GRP_name>";
attribute BBOX: string;
attribute BBOX of <object>: label is "<height>,<width>";

VHDL Example Code

```
attribute GRP: string;
attribute GRP of x: label is "reg_group";
attribute BBOX: string;
attribute BBOX of <object>: label is "9,7";
```

Verilog Syntax – Synplify /* synthesis GRP= "<GRP_name>"
BBOX= "<h,w>"
*/;

Verilog Example Code – Synplify
```
"module serial_reg_custom(D,
CLK, CE, RST, Q)
 /* synthesis GRP = "reg_group" GLOC= "R5C19D"  BBOX= "6,4"*/;
serial_reg_custom inst1A(.D(A), .CLK(CLK), .CE(CE), .RST(RST),
.Q(adder1_in1));
serial_reg_custom inst1B(.D(B), .CLK(CLK), .CE(CE), .RST(RST),
.Q(adder1_in2));
serial_reg_custom inst1C(.D(adder1_sum), .CLK(CLK), .CE(CE),
.RST(RST), .Q(SUM));
```

Note

Synthesis directives to preserve hierarchy are typically needed to retain the instances constrained by the GRP attribute.

.ldc TCL command
```
ldc_create_group -name reg_GROUP -bbox {6 4} [get_cells {inst1A
inst1B inst1C}]
ldc_set_location -site {R5C19D} [ldc_get_groups reg_GROUP]
```

Setting Anchor location using PDC Constraint

The anchor of a user group has to be the top slice in a PFU. The examples below specify ‘F’ for Avant {R5C19F} and ‘D’ in {R5C19D} for other devices.

```
ldc_set_location -site {R5C19F} [ldc_get_groups reg_GROUP]
ldc_set_location -site {R5C19D} [ldc_get_groups reg_GROUP]
```

GRP Attribute Usage Rules and Restrictions Observe the following conditions for proper GRP attribute usage:

- All elements within the GRP block belong to that particular GRP.
- Nested GRP blocks are considered as unique individual GRPs.
- All blocks belong to the GRP attached to their nearest ancestor in the hierarchy.
- Unlike GRP, the mapper will not append the hierarchical path plus the block instance name.

GSR

Convention GSR= value
GSR=ENABLED | DISABLED

Description Enables or disables the global set/reset function. Can be applied to a module or cell.

Device Support LAV-AT, LFD2NX, LIFCL, UT24C

VHDL Syntax ATTRIBUTE GSR : string;
ATTRIBUTE GSR OF [module_name]: entity IS "[value]";

VHDL Example Code
```
ATTRIBUTE GSR : string;
ATTRIBUTE GSR OF behave: entity IS "ENABLED";
```

Verilog Syntax – Synplify module [module_name] (<ports>) /* synthesis GSR= "[value]"*/;

Verilog Example Code – Synplify
```
module pci_top (portA, portB) /*
synthesis GSR = "ENABLED" */;
```

.ldc TCL command
```
ldc_set_attribute GSR=ENABLED
ldc_set_attribute GSR=DISABLED [get_cells {pll}]
```

HYSTERESIS

Convention HYSTERESIS=ON | NA

Description The ratioed input buffers have two 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.

LVCMOS12/12H and LVCMOS10/10H do not support HYSTERESIS so this setting is always disabled for these input types. HYSTERESIS is likewise 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 LAV-AT, LFD2NX, LIFCL, UT24C

VHDL Syntax – Synplify ATTRIBUTE HYSTERESIS: string;
ATTRIBUTE HYSTERESIS OF [signal_name]: SIGNAL IS "[ON | NA]";

VHDL Example Code
```
ATTRIBUTE HYSTERESIS OF q_lvttl33_17:
SIGNAL IS "ON";
```

Verilog Syntax – Synplify /* synthesis HYSTERESIS= "ON" | "NA" */;

Verilog Example Code
```
/* synthesis attribute HYSTERESIS of
q_lvttl33_17 is "ON"*/;
```

.ldc TCL command `ldc_set_port -iobuf {HYSTERESIS=ON} [get_ports q_lvttl33_17]`

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.

Device Support All

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 – Synplify module [module_name] (ports) /* synthesis INBUF= "[value]"*/;

Verilog Example Code – Synplify
```
module pci_top (portA, portB) /* synthesis INBUF= "OFF" */;
```

.ldc TCL command
```
ldc_set_port -iobuf {INBUF=OFF} [get_ports portA]
```

INIT

Convention INIT= value
INIT=ENABLED | DISABLED

Description Initializes the look-up table values for all device families.

Device Support All

VHDL Syntax ATTRIBUTE INIT : string;
ATTRIBUTE INIT OF [module_name]: entity IS "[value]";

VHDL Example Code
```
ATTRIBUTE INIT : string;
ATTRIBUTE INIT OF behave: entity IS "ENABLED";
```

Verilog Syntax – Synplify module [module_name] (ports) /* synthesis INIT= "[value]"*/;

Verilog Example Code – Synplify `module pci_top (portA, portB) /* synthesis INIT= "ENABLED" */;`

.ldc TCL command `ldc_set_port -iobuf {INIT=ENABLED} [get_ports portA]`

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). Multiple values may be required 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 constraint. Refer to IOBUF to see how global constraints are honored for IO_TYPE. For valid buffer types, refer to the specific element in the FPGA Libraries Help. See the Lattice Avant sysI/O User Guide.and the sysI/O User Guide for Nexus Platform for legal IO_TYPE for inputs and outputs.

Table 176: IO_TYPE Values

Device Support	Default Value	IO_TYPE User Guide
LAV-AT	LVCMOS33: default value for regular-speed IOs. LVCMOS18: default value for high-speed IOs.	For a more detailed information about Avant IO_TYPE values, refer to the Lattice Avant sysI/O User Guide.
LFCPNX LFD2NX LIFCL LFMXO5	LVCMOS33: default value for regular-speed IOs. LVCMOS18H: default value for high-speed IOs.	For a more detailed information about IO_TYPE values, refer to the sysI/O User Guide for Nexus Platform.

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 – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]"

SLEWRATE="[value]"*/;
[type_name] = (valid I/O type, such as LVCMOS18)

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"*/;`

.ldc TCL command `ldc_set_port -iobuf {IO_TYPE=LVTTL33} [get_ports portA[1]]`

LOC

Convention LOC= site_name (PIN | POS)

Description Specifies a site location for the component that is created when this block is mapped. Attaching 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. It then attempts 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 constraint, MAP also writes a LOCATE constraint for the component into the constraint file. The LOCATE constraint 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 the generation of a locate constraint. 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 – Synplify PinType [pin_name] /* synthesis LOC="[site_name]"*/;

Verilog Example Code – Synplify
```
//input rst  /* synthesis LOC="H5" */ ;
```

.ldc TCL command
```
ldc_set_location -site {H5} [get_ports rst]

ldc_define_attribute -attr loc -value A12 -object_type port -object {clk1}
```

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";
```

NOCLIP

Convention NOCLIP=1

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.

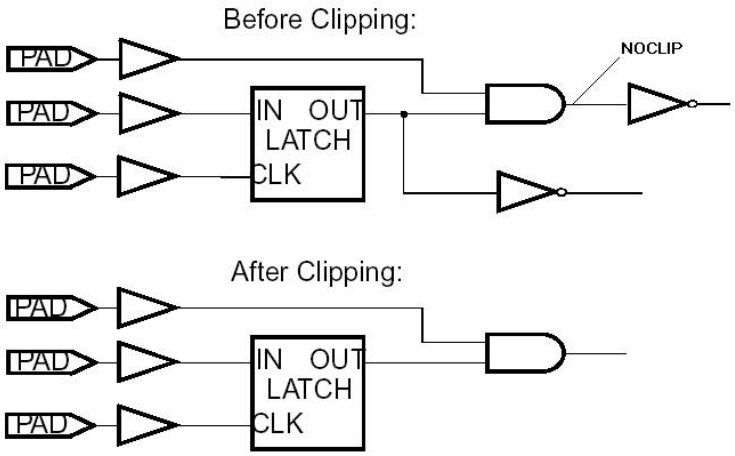
Valid path outputs include:

- 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 following figure shows an example of how a NOCLIP attribute affects clipping:

Figure 52: Example of 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

Verilog example `wire net_a /* synthesis syn_keep=1 */ /* synthesis NOCLIP=1 */;`

or

```
wire net_a /* synthesis syn_keep=1  NOCLIP=1 */;
```

"syn_keep" is a logic synthesis directive. "NOCLIP" applies to physical design. Both need to be specified if you want to keep the net throughout the flow and only use syn_keep for synthesis cross-probing without NOCLIP.

VHDL example `signal net_a: std_logic;`

```
attribute syn_keep : boolean;
attribute NOCLIP : boolean;

attribute syn_keep of net_a : signal is true;
attribute NOCLIP of net_a : signal is true;
```

Although NOCLIP is declared as boolean in VHDL, synthesis tool writes NOCLIP=1 in synthesis output.

.ldc TCL command `ldc_set_attribute {NOCLIP} [get_nets net_a]`

NOMERGE

Convention NOMERGE= “ON”

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.

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

VHDL Syntax ATTRIBUTE NOMERGE: string;
ATTRIBUTE NOMERGE OF [signal_name]: SIGNAL IS "ON";

VHDL Example Code `ATTRIBUTE NOMERGE OF q_lvttl33_17: SIGNAL IS "ON";`

Verilog Example Code – Synplify `output q_lvttl33_17 /* synthesis NOMERGE="ON" */;`

.ldc TCL command `ldc_set_location {NOMERGE} [get_nets q_lvttl33_17]`

See Also

- “IO_TYPE” on page 740

OPENDRAIN

Convention OPENDRAIN = OFF | ON

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” 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.

Device Support LAV-AT, LFD2NX, LIFCL, UT24C

VHDL Syntax ATTRIBUTE OPENDRAIN: string;
ATTRIBUTE OPENDRAIN OF [signal_name]: SIGNAL IS "[value]";

VHDL Example Code `ATTRIBUTE OPENDRAIN OF q_lvtt133_17: SIGNAL IS "ON";`

Verilog Syntax Output signal_name /* synthesis OPENDRAIN=”ON | OFF” */;

Verilog Example Code `output q_lvttl133_17 /* synthesis OPENDRAIN=”ON” */;`

.ldc TCL command `ldc_set_port -iobuf {OPENDRAIN=ON} [get_ports {q_vttl133_17}]`

PULLMODE

Convention PULLMODE= DOWN (default), UP, NONE, I3C, KEEPER, FAILSAFE,100K, 3P3K, 6P8K, 10K, NA

Description Attached to IO buffers, the PULLMODE attribute has the following mode parameters

- DOWN (default) – Pull down
- UP – Pull up
- NONE – No pull up/down
- KEEPER – Retain the last state if bus connected to this IO goes to tri-state.
- FAILSAFE – For differential IOs, uses a weak pullup on the "true" pad and a weak pulldown on the "complement" pad to give a determinate value when undriven.
- I3C
  - If the BB_I3C_B primitive is used, the I3C value will be set for Avant devices. This is for the Avant I3C buffer.
  - If the BB_I3C_A primitive is used, the I3C value will be set for Nexus devices. This is for the Nexus I3C buffer.

The PULLMODE options can be enabled for each I/O independently.

Device Support All

VHDL Syntax ATTRIBUTE PULLMODE: string;
ATTRIBUTE PULLMODE OF [pin_name]: SIGNAL IS "[mode]";

VHDL Example Code: `ATTRIBUTE PULLMODE OF md: SIGNAL IS "3P3K";`

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]"*/;

Note

Example given for Pin I/O Type configuration.

Verilog Example Code – Synplify `portA /* synthesis IO_TYPE="LVCMOS33" DRIVE="NA" PULLMODE="3P3K" */;`

.ldc TCL command `ldc_set_port -iobuf {PULLMODE=100K} [get_ports {A[4]}]`

See Also

- "IO_TYPE" on page 740

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

See Also

- "GRP" on page 735 (attribute)

REGION

Convention REGION = <identifier>

Description REGION indicates the region to which a given GRP belongs. This attribute must appear on a block that has a GRP attribute. This has

replaced the old PREGION attribute which is also still valid for backwards compatibility.

Note that RLOC / GLOC (Region Lock / Group Lock) can only be used with REGION or GRP attributes. The value it takes is a site value on the device itself which can be deciphered as such:

RLOC/GLOC = “R5C19D”

i.e. Row 5, Column 19, Slice D

Device Support All

VHDL Syntax attribute REGION: string;
attribute REGION of <object>: label is "<REGION_name>";
attribute RLOC: string;
attribute RLOC of <object>: label is "<site>";
attribute RBBOX: string;
attribute RBBOX of <object>: label is "<height>,<width>";
attribute GRP: string;
attribute GRP of <object>: label is "<GRP_name>";
attribute GLOC: string;
attribute GLOC of <object>: label is "<site>";
attribute BBOX: string;
attribute BBOX of <object>: label is "<height>,<width>";

VHDL Example Code:
```
attribute REGION: string;
attribute REGION of x: label is "reg_group";
attribute RLOC: string;
attribute RLOC of <x>: label is "R5C19D";
attribute RBBOX: string;
attribute RBBOX of <x>: label is "20,15";
attribute GRP: string;
attribute GRP of <x>: label is "reg_GRP";
attribute GLOC: string;
attribute GLOC of <x>: label is "R10C20D";
attribute BBOX: string;
attribute BBOX of <x>: label is "5,5";
```

Verilog Syntax – Synplify /* synthesis REGION="<REGION_name>" RBBOX= "<h,w>" */;

// The attribute must appear on the same block as the RBBOX attribute.

Verilog Example Code Synplify
```
"module serial_reg_custom(D, CLK, CE, RST, Q) /*  synthesis REGION="reg_REGION"
 RLOC= "R5C19D"
 RBBOX= "20,15"
 GRP= "reg_GRP"
 GLOC="R10C20D"
 BBOX= "5,5"*/;

serial_reg_custom inst1A(.D(A), .CLK(CLK), .CE(CE), .RST(RST), .Q(adder1_in1));
```

```
serial_reg_custom inst1B(.D(B), .CLK(CLK), .CE(CE), .RST(RST),
.Q(adder1_in2));
serial_reg_custom inst1C(.D(adder1_sum), .CLK(CLK), .CE(CE),
.RST(RST), .Q(SUM));

count count_inst(A,CLK,RST) /* synthesis REGION="reg_REGION" */
;
```

.ldc TCL command

```
ldc_create_region -name reg_REGION -site R5C19D -width 20 -
height 15
ldc_create_group -name reg_GROUP -bbox {5 5} [get_cells {inst1A
inst1B inst1C}]
ldc_set_location -site {R10C20D} [ldc_get_groups reg_GROUP]
ldc_set_location -region reg_REGION [get_cells {inst1A inst1B
inst1C count_inst}]
```

See Also

- "GRP" on page 735 (attribute)

RGB_TO_GPIO

Convention RGB_TO_GPIO = String

Description Specific to the iCE40UP device by default, there are three dedicated pins for RGB use. In order to free up these pins for use as general purpose IOs (GPIO), an attribute can be applied to make use of these three pins.

When an IO port is manually located to one of the RGB pins by the user and the RGB primitive is not used for the port signal, the software can automatically configure the port as a GPIO without using this attribute.

This attribute is applicable to the module level in Verilog and architecture level in VHDL.

Note

1. When an I/O port is manually located to one of the RGB pins by the user and the RGB primitive is not used for the port signal, the software can automatically configure the port as a GPIO without using this attribute.
2. This attribute is not applicable to RTL simulation and post synthesis, but applies to post route simulation. Also, a pull-up is needed in the test bench to avoid a High-Z.

Device Support iCE40UP

VHDL Syntax ATTRIBUTE RGB_TO_GPIO : string;
ATTRIBUTE RGB_TO_GPIO of [architecture name]: architecture is "[pin_name]";

VHDL Example Code
```
ATTRIBUTE RGB_TO_GPIO : string;
ATTRIBUTE RGB_TO_GPIO OF behav : architecture is
"cam_data[0:0], cam_pclk";
```

Verilog Syntax /* synthesis RGB_TO_GPIO = "[pin_name]"*/;

Verilog Example Code
```
module abc (cam_pclk, cam_data, q) /*
synthesis RGB_TO_GPIO = "cam_data[0:1], cam_pclk" */ ;
```

SLEWRATE

Convention SLEWRATE = FAST | SLOW | MED

Description Attached to all output buffers (e.g., OB, OBW) and bidirectional buffers, the SLEWRATE attribute mode parameters are FAST, SLOW OR MED for all families.

Each I/O pin has an individual slew rate control. This allows designer to specify slew rate control on pin by pin basis. The slew rate controls affects both the rising edge and the falling edges. The default SLEWRATE is MEDIUM.

Device Support LAV-AT, LFD2NX, LIFCL, UT24C

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 PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]" */;

Verilog Example Code
```
output [4:0] portA /* synthesis
IO_TYPE="LVCMOS33" DRIVE="16" PULLMODE="UP" SLEWRATE="FAST" */;
```

.ldc TCL command
```
ldc_set_port -iobuf {SLEWRATE=FAST}
[get_ports {portA}]
```

USERCODE

Description Defines the 32-bit code used to identify the design.

Note that the TCL command uses a combination of FORMAT and UNIQUE_ID instead to represent USERCODE with no such keyword. See ldc_set_attribute for details.

Device Support All devices except iCE40UP

.ldc TCL command `ldc_set_attribute {FORMAT=HEX CODE=0123ABCD}`

Lattice Synthesis Engine Constraints

The LSE, enables you to use constraints that are directly interpreted by the synthesis engine. This new category of constraints includes "Synopsys Design Constraints" on page 750, and "Lattice Synthesis Engine-Supported HDL Attributes" on page 767. These constraints influence the optimization or structure of the output netlist.

See Also
- "Integrated Synthesis" on page 317
- "Handling Device Constraints" on page 212

Synopsys Design Constraints

This section describes the SDC language elements for timing-driven synthesis that are supported by the 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 Source Editor. You can select the .ldc file to be the active synthesis constraint file for an implementation.

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 Timing analysis report (.twr). Also, some elements are modeled as black boxes in LSE.

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 751
- "Object Access Commands" on page 753
- "create_clock" on page 753
- "create_generated_clock" on page 755
- "set_clock_groups" on page 757
- "set_clock_latency" on page 758
- "set_clock_uncertainty" on page 759

- "set_false_path" on page 760
- "set_input_delay" on page 761
- "set_load" on page 762
- "set_max_delay" on page 762
- "set_max_skew" on page 763
- "set_min_delay" on page 764
- "set_multicycle_path" on page 765
- "set_output_delay" on page 766
- "Integrated Synthesis" on page 317
- "Handling Device Constraints" on page 212

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 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 can used to define the target for a create_generated_clock command.

Wildcard support

Only one wildcard (*) is supported. In addition, 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.

Notes:

1. Wildcard * can be used in any position of object name such as a*, *b, a*b etc.
2. Wildcard * can't cross hierarchical separator /. For example get_cells sub1/* only returns sub1/sub2 and doesn't return sub1/sub2/sub1 or sub1/sub2/sub1/sub2 etc.
3. When -hierarchical is set, it will search all levels of the design hierarchy starting at the current instance.

See Also ▶"Synopsys Design Constraints" on page 750

Object Access Commands

The table below summarizes standard SDC access commands and their options.

Command	Options
all_clocks	N/A.
all_inputs	-level_sensitive, -edge_triggered, -clock
all_outputs	-level_sensitive, -edge_triggered, -clock
current_design	N/A
current_instance	N/A
get_cells	-hierarchical, -regexp, -nocase, -of_objects
get_clocks	-regexp, -nocase
get_nets	-hierarchical, -hsc_separator, -regexp, -nocase, -of_objects
get_pins	-hierarchical, -hsc_separator, -regexp, -nocase, -of_objects
get_ports	-regexp, -nocase

Example The following example shows how -of_objects is used:

```
get_pins -of_objects [get_nets {c[2]}]
get_cells -of_objects [get_nets -of_objects [get_pins {c/clock}]
c
```

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 -period <period>[-name <clock name>] [-waveform <value1 value2>] [<source_object>] [-add]

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.

-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. Note that each value must also be less than the clock period.

source_object

This value is the object on which the clock constraint is defined. The source object can be a port, a pin, 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 in 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 displays one port or net object.

-add

This option allows you to define another clock to an object that has an existing clock.

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]
```

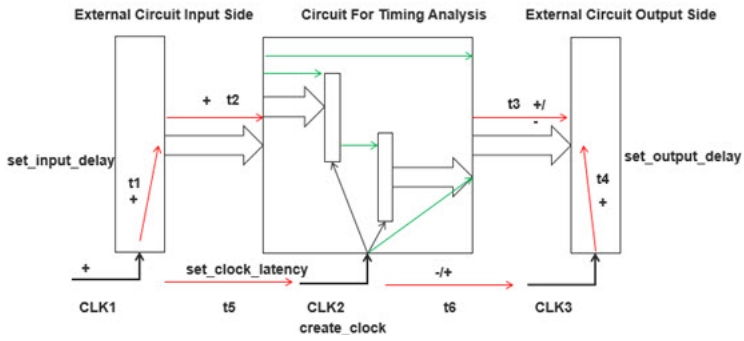
Example The following example creates a clock on port CK3 with a period of 7.1, and has two edges at 0 and 4.1:

```
create_clock -period 7.1 -waveform {0 4.1} [get_ports CK3]
```

Example Radiant Virtual Clocks are external to the FPGA and only trigger flip flops that are outside the FPGA. These clocks do not trigger flip flops from

inside the FPGA and are only used as the clocks are constrained with set_input_delay and set_output_delay.

Example In the following example, CLK1 and CLK3 are virtual clocks.




```
Input and output delay constraints must take external delays into
consideration.
Input side:
create_clock -period T -name CLK1
create_clock -period T -name CLK2 [get_ports CLK]
set_clock_latency t5 [get_clocks CLK2]
set_input_delay t1 + t2 -clock [get_clocks CLK1] [all_inputs]

Output side:
create_clock -period T -name CLK3
set_output_delay t3 + t4 - t6 -clock [get_clocks CLK3]
[all_outputs]

If ther is clock uncertainty:
set_clock_uncertainty tu [get_clocks CLK2]
```

See Also ▶"Synopsys Design Constraints" on page 750

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 is considered a clock when defining constraints such as input_delay.

Syntax create_generated_clock –source [get_ports | get_pins/get_nets <*reference_object*>] [[[-divide_by <factor>] [-multiply_by <factor>] [-duty_cycle <value>]] | [-edges <edge specs>]] [-invert] [-name <clock name>] [-add] [get_pins | get_nets <pin/net name>] [<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.

-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.

-edges {*value1 value2 value3}*

A maximum of three values can be entered. The values entered correspond to the edge of the source clock from which the generated clock has been obtained.

-edge_shift

Specifies a list of floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified.

For example, 1 indicates that the corresponding edge is to be shifted by 1 library time unit.

-invert

This inverts the clock edge.

-add

This option allows defining another clock on an object that already has a generated clock.

-master_clock

This value specifies the master clock for the generated clock object if there are multiple clocks found on the source object.

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 three quarters 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" on page 750

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 [-logically_exclusive | -physically_exclusive | -asynchronous] –group [get_clocks <*clock_objects*>] -group [get_clocks <*clock_objects*>]

Arguments -asynchronous

Specifies that the clock groups are asynchronous to each other (while the Radiant software assume all clocks defined by create_clock and create_generated_clock are synchronous). Two clocks are asynchronous with respect to each other if they have no phase relationship at all.

-pyhsically_exclusive

Specifies that clocks are mutually exclusive. Only one clock group is active at any given time.

-logically_exclusive

Specifies that clocks are mutually exclusive and will not reach the flip flops at the same time due to logic implementation.

-group *clock_object*

Specifies the clock objects in a group. Specifying one group indicates 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 -logically_exclusive -group [get_clocks
{clka_port clkb_port}] -group [get_clocks clkc_port]
```

See Also ▶"Synopsys Design Constraints" on page 750

set_clock_latency

Specifies the behavior of the clock outside of the FPGA.

Syntax set_clock_latency <value> -source [-rise] [-fall] [-early | -late] [get_clocks <clock name>]

The –source option of the constraint is required because only source latencies are supported by the timer. The value of the latency is the delay of the clock outside the FPGA. The value could be a rise, fall, early or late value. The max and min flags indicate delays that will be used for setup and hold requirement calculations. The early delay will be used as the capture clock delay for setup calculations and the launch delay for hold time calculations. The late delay will be used on the launch clock for setup and on the capture

clock for hold. The rise and fall options indicate the delay of the rising and falling edges of the clock.

Examples:

```
set_clock_latency 3 -source -early [get_clocks clk1]
set_clock_latency 4 -source -late [get_clocks clk1]
```

Setup analysis always uses longest path and hold analysis uses shortest path. In setup analysis, “early" value is used to calculate required time, "late" value is to calculate arrival time. While in hold analysis, the "late" value is used to calculate the required time and "early" is used to calculate arrival time.

In the example above, 3ns is in the destination clock of setup and source clock of hold; while 4ns is in source clock of setup and destination clock of hold.

set_clock_uncertainty

This constraint indicates that the clock of interest has uncertainties in its period.

Syntax set_clock_uncertainty <value> [-from <from clock>] [-to <to clock>] [-setup] [-hold] [get_clocks <clock list>]

This constraint requires at least one of the clock options. The number given by value, is the amount by which the clock period in uncertain. The timer will use this value to add additional requirement to all paths affected by the selected clock. The timer requires the ‘from’ option to be accompanied by a ‘to’ option. The setup option indicates the constraint for setup analysis and the hold constraint for hold analysis.

Examples The following two examples specify clock uncertainty, one with a from/to.

```
set_clock_uncertainty 0.5 [get_clocks clk1]
set_clock_uncertainty 0.4 -from [get_clocks clk2] -to
[get_clocks clk3]
```

The first constraint sets the uncertainty of clk1 to half a nanosecond and the second constraint sets the uncertainty to 400 picoseconds when a path is launched by clk2 and captured by clk3.

set_false_path

Identifies paths that are considered false and excluded from timing analysis.

The command requires a minimum of one from, through or to option. This constraint defines paths that should be removed from consideration during timing analysis. All paths that start at one of the 'from' objects, pass through one of the 'through' objects and arrive at one of the 'to' objects will be treated as an unconstrained path. If any of the options are missing, all objects in the design that satisfy the criteria of the missing option will be used as objects of the missing option. A missing "from" option, for instance, implies that all timing starting points are part of the 'from' list for the constraint.

Syntax set_false_path [-from <port_list|pin_list|instance_list|clock_list>] [-to <port_list|pin_list|instance_list|clock_list>] [-through <port_list|pin_list|instance_list|net_list>] [-rise_from <clock_list>] [-rise_to <clock_list>] [-fall_from <clock_list>] [-fall_to <clock_list>][-setup] [-hold]

Arguments -from | rise_from | fall _from *from object_list*

Specifies the timing path start point. A valid timing starting point is a clock, a primary input, or a sequential cell.

-to *to object_list*

Specifies the timing path end point. A valid timing end point is a primary output, a sequential cell, or a clock.

-through *object_list*

Specifies a net through which the paths should be blocked.

-setup

This option eliminates setup timing analysis for specified timing path.

-hold

This option eliminates hold timing analysis for specified timing path.

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:

```
set_false_path -from [get_ports clk1] -to [get_cells reg_2]
```

The following example specifies all paths through the net UO/sigA as false:

```
set_false_path -through [get_nets UO/sigA]
```

See Also ▶"Synopsys Design Constraints" on page 750

set_input_delay

Defines the arrival time of an input relative to a clock.

Syntax set_input_delay -clock <*clock_object*> [-clock_fall] [-min | -max] [-add_delay] <*delay_value*> <*input_port_object*>

<*input_port_object*> = [get_ports <names>] | [all_inputs]

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.

-clock_fall

A switch to specify the falling edge of clock to trigger.

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.

add_delay

Used to define more than one output delay on a given port.

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]

set_output_delay 4.0 -clock [get_clocks vclk] -add_delay
[get_ports out2]
```

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:

See Also ▶“Synopsys Design Constraints” on page 750

```
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]
```

set_load

To accurately perform timing and SSO analysis, the tool needs information regarding the external load capacitance of nets connected to output ports of the FPGA.

Syntax set_load value objects

Arguments *delay_value*

Specifies the load in capacitance in pF.

objects

Provides one or more output ports in the current design to which delay_value is assigned. Uses TCL search strings.

Example The following example sets a capacitive load of 25 pF for output port OUT:

```
set_load 25 [get_ports {out}]
```

set_max_delay

Specifies the maximum delay for the timing paths.

Syntax set_max_delay [-from <port_list|pin_list|instance_list|clock_list>] [-to <port_list|pin_list|instance_list|clock_list>] [-through <port_list|pin_list|instance_list|net_list>] [-rise_from <clock_list>] [-rise_to <clock_list>] [-fall_from <clock_list>] [-fall_to <clock_list>] [-datapath_only] <delay_value>

<… list> = [(get_ports | get_pins | get_cells | get_nets | get_clocks) <names>]

Arguments *-datapath_only*

The **-datapath_only** option calculates the Clock-to-Q delay of the first flop, the total delay between the flops, and the setup time of the second flop. This option will not consider clock skew and jitter.

If the “-from” option is missing from the -datapath_only syntax, it will trigger an error and may cause the timing engine to stop.

The timing engine will not compute a hold check for a path constrained with the -datapath_only constraint.

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 points. Valid timing start points are top level input ports or the triggered output pins of sequential cells. You can also use the keyword "all_registers" to include all registers.

-to *to port_object or cell_object*

Specifies the timing path end points. Valid timing end points are primary output ports or clocked input ports of sequential cells.

-through *port_object or cell_object*

Specifies the timing path pass through point. The timing path must go through this object.

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 -from [get_cells ff1a] -to [get_cells ff2e] 5.0

set_max_delay -from [get_pins ff1a:CLK] -through [get_pins
and1/B] -to [get_pins ff2e:D] 5.0
```

See Also ▶"Synopsys Design Constraints" on page 750

set_max_skew

The set_max_skew command is used to define the max skew of a path. You can only apply this command to data nets.

Syntax set_max_skew <object_list> <skew_value>

<object_list> = [(get_pins | get_nets) <names>]

Arguments object_list

Contains the pins or nets of the object list. The constraint requires both the object list and the skew value.

Note:

A net or a pin can only be an object in one set_max_skew constraint.

-skew_value

Controls the delay to the various loads of the net relative to each other.

The constraint indicates the expected maximum skew of the delays from the drivers to the loads of the nets. The difference between the maximum and minimum delay to the loads of the nets is expected to be less than the skew value.

Examples The following example shows that the first constraint requires the maximum difference between the net delays of the loads on the net connected to pin “A” to be less than or equal to 250 picoseconds.

The second constraint requires the difference between the net delays to the various loads of the net “ff1/Q_c” to be less than or equal to 370 picoseconds:

```
set_max_skew [get_pins A] 0.25
set_max_skew [get_nets ff1/Q_c] 0.37
```

See Also “Synopsys Design Constraints” on page 750

set_min_delay

Specifies the minimum delay for the timing paths.

Syntax set_min_delay [(-from)<*port_object or cell_object*>] [-through *port_object or cell_object*] [(-to) <*port_object or cell_object*>] <delay_value>

<object_list> = [(get_ports | get_pins | get_nets | get_cells) <names>]

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 sequential cell or a top-level input port. You can also use the keyword “all_registers” to include all registers.

-to *to port_object or cell_object*

Specifies the timing path end point. A valid timing end point is a primary output or a sequential cell.

-through *port_object or cell_object*

Specifies the timing path pass through point. The timing path must go through this object.

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:

See Also ▶“Synopsys Design Constraints” on page 750

```
set_min_delay -from [get_cells ff1a] -to [get_cells ff2e] 5.0
set_min_delay –from [get_pins ff1a:CLK] –through [get_pins
and1/B] –to [get_pins ff2e:D] 5.0
```

set_multicycle_path

Defines a path that takes multiple clock cycles.

Syntax

set_multicycle_path [-from <port_list|pin_list|instance_list|clock_list>]
[-to <port_list|pin_list|instance_list|clock_list>]
[-through <port_list|pin_list|instance_list|net_list>]
[-rise_from <clock_list>] [-rise_to <clock_list>]
[-fall_from <clock_list>] [-fall_to <clock_list>]
[-setup | -hold] [-start | -end] <path_multiplier>

Arguments

-from port_list|pin_list|instance_list|clock_list

(Optional) This value specifies the timing path start point.

-to port_list|pin_list|instance_list|clock_list

(Optional) This value specifies the timing path end point.

-through port_list|pin_list|instance_list|net_list

(Optional) This value specifies the timing path pass through point. The timing path must go through this object.

-rise_from|fall_from clock_list

Same as '-from' but only applies to clock objects.

Option '-from', 'rise_from' and '-fall_from' are mutually exclusive.

-rise_to|fall_to clock_list

Same as '-to' but only applies to clock objects.

Option '-to', 'rise_to' and '-fall_to' are mutually exclusive.

At least one option among '-from|-rise_from|-fall_from|-to|-rise_to|-fall_to|-through' is needed.

-setup

(Optional) This option set Setup multiplier.

-hold

(Optional) This option set Hold multiplier.

Option '-setup' and '-hold' are mutually exclusive.

-start

(Optional) This option indicates that Multiplier measured against startpoint.

-end

(Optional) This option indicates that Multiplier measured against endpoint.

Option '-start' and '-end' are mutually exclusive.

path_multiplier

This value specifies a value that represents the number of cycles.

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 -setup -from [get_cells reg1] -to
[get_cells reg2]

set_multicycle_path 3 -from [get_pins ff1/Q] -through [get_pins
and1/B] -to [get_pins ff2/D]
```

See Also ►"Synopsys Design Constraints" on page 750

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*

set_output_delay (-clock | -clock_fall) <clock_name> [-min | -max] [-add_delay] <delay_value> <port list>

<port_list> = [get_ports <names>] | [all_outputs]

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_fall *clock_object*

Specifies the clock reference to which the specified input delay is related. clock_fall is on the falling edge of clock and 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. Use the keyword “all_outputs” to include all output ports.

add_delay

Used to define more than one output delay on a given port.

Example The following example sets an output delay of 1.2 ns for all outputs relative to CLK1

```
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” on page 750

Lattice Synthesis Engine-Supported HDL Attributes

This section describes the Synplify Lattice Attributes that are supported by the LSE. These attributes are directly interpreted by the engine and influence the optimization or structure of the output netlist.

All HDL attributes have priority over Strategy settings.

See Also ▶“black_box_pad_pin” on page 769

▶ “full_case” on page 771

- “loc” on page 772
- “ldc_define_attribute” on page 773
- “parallel_case” on page 775
- “syn_black_box” on page 777
- “syn_encoding” on page 778
- “syn_force_pads” on page 782
- “syn_hier” on page 784
- “syn_insert_pad” on page 785
- “syn_keep” on page 787
- “syn_maxfan” on page 789
- “syn_multstyle” on page 790
- “syn_noprune” on page 792
- “syn_preserve” on page 792
- “syn_ramstyle” on page 795
- “syn_replicate” on page 797
- “syn_romstyle” on page 799
- “syn_srlstyle” on page 801
- “syn_sharing” on page 804
- “syn_state_machine” on page 806
- “syn_use_carry_chain” on page 810
- “syn_useenables” on page 811
- “syn_useioff” on page 813
- “translate_off/translate_on” on page 814

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, the ports are listed 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 space free, 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" ;

The object is the architecture name of a black box, while the data type is string. The portList is a space free, 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 BBDLHS;

architecture BBDLHS_behav of BBDLHS is
end BBDLHS_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;
   endcase
end
endmodule
```

loc

The loc attribute specifies pin locations for Lattice I/Os, instances, and registers, and forward-annotates them to the place-and-route tool. 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 space free, comma-separated list of pin locations.

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 space free, comma-separated list of pin locations.

VHDL Example

```
entity mycomp is port(DATA0 : in std_logic_vector (3 downto 0);
end entity mycomp;
attribute loc : string;
attribute loc of DATA0 : signal is "p10,p12,p11,p15";
```

ldc_define_attribute

The ldc_define_attribute Tcl command is used to define synthesis attributes for the LSE flow. Its main entry mechanism is through the Attribute tab of the Pre-Synthesis Constraint Editor tool. When you drag and drop or enter the appropriate values for the attribute options below, it will be echoed as a Tcl command in the .LDC constraint file.

Usage

ldc_define_attribute -attr <attr_type> -value <attr_value> -object_type <object type> -object <object> [-disable] [-comment <comment>]

Options

- -attr: LSE attributes name, such as "loc"
- -value: LSE attribute value such as "1" and "ENABLED"
- -object_type: the type of object to which the attribute belong, such as "port"
- -object: the specific object to which the attribute apply, such as "clk"
- -disable: the command is disabled
- -comment: comment of the command

Example

```
ldc_define_attribute -attr loc -value 2 -object_type port -
object {clk}
```

ldc_define_global_attribute

Commands to define global LSE attribute

Syntax:

```
ldc_define_global_attribute -attr <attr_name> -value
                <attr_value> [-disable] [-comment <comment>]
```

Arguments

- -attr attr_name: This value indicates LSE attributes name, such as 'syn_ramstyle'.
- -value attr_value: This value indicates LSE attribute value such as 'registers'.
- -disable: This option indicates that the command is disabled (Optional).
- -comment: This option specifies comment of the command (Optional).

Example

```
ldc_define_global_attribute -attr syn_ramstyle -value registers
```

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'b1???: 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.

Note:

For Synplify Pro flow, the **syn_black_box** attribute works with **syn_declare_black_box** to indicate user-defined black box.

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 769.

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
attribute syn_black_box : boolean;
attribute syn_black_box of bl_box_behav : architecture is true;
begin
end bl_box_behav;
```

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: 0000, 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

```
module  fsm (clk, reset, x1, outp);
input clk, reset, x1;
output outp;
reg outp;
reg [1:0] state /* synthesis syn_encoding = "onehot" */;
parameter s1 = 2'b00; parameter s2 = 2'b01;
parameter s3 = 2'b10; parameter s4 = 2'b11;

always @(posedge clk)
begin
   if (reset)
      state <= s1;
   else begin
      case (state)
      s1: if (x1 == 1'b1)
         state <= s2;
      else
         state <= s3; s2: state <= s4;
      s3: state <= s4;
      s4: state <= s1;
      endcase
   end
   end

always @(state) begin
   case (state)
      s1: outp = 1'b1;
      s2: outp = 1'b1;
      s3: outp = 1'b0;
      s4: outp = 1'b0;
   endcase
   end
endmodule
```

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 : boolean;
  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);
  attribute syn_force_pads of rtl : architecture is true

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 is completed.

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;
```

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.

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 /*synthesis syn_multstyle = "block_mult"*/;
  wire [15:0] product;

  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.numeric_std.all;

entity mult is
port (clk : in std_logic ;
 a : in std_logic_vector(7 downto 0) ;
 b : in std_logic_vector(7 downto 0) ;
 c : out std_logic_vector(15 downto 0));
end mult ;

architecture rtl of mult is
signal mult_i : std_logic_vector(15 downto 0) ;
attribute syn_multstyle : string ;
attribute syn_multstyle of mult_i : signal is "logic" ;

begin
mult_i <= std_logic_vector(unsigned(a)*unsigned(b)) ;
process(clk)
begin
 if (clk'event and clk = '1') then
 c <= mult_i ;
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 and 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 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 *objectType* is boolean, and *object* is a component.

VHDL Example

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.

Note: Only the objects that are existent in the RTL view netlist (i.e. appearing in Netlist Analyzer) after synthesis will honor the attribute through to the technology mapping.

Verilog Syntax object /* synthesis syn_preserve = 1 */ ;

where *object* is a register definition signal or a module.

Verilog Example

VHDL Syntax attribute syn_preserve of object : objectType is true ;

```
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 u1 : label 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;
```

```
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 = "block_mult"*/;
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
```

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 "block_mult";
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 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.

Note:

If the memory size defined (depth x width) is less than thirty registers, then EBRs will not be inferred.

Verilog Syntax

object /* synthesis syn_ramstyle = "string" */ ;

where *object* is a register definition (reg) signal. The data type is string.

You can also use the following syntax as an alternative:

object; // synthesis syn_ramstyle = "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" */;
```

Verilog Example with Alternative Syntax

```
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. While 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;
```

syn_romstyle

This attribute allows you to implement ROM architectures using dedicated or distributed ROM. Infer ROM architectures uses 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_rvl_debug

This attribute highlights debug signals so that they can be easily identified in the user interface.

In the Reveal post-synthesis stage, this attribute tells the synthesis tool to preserve the signal without optimizing it. If it is a bus, the data width is also preserved. The synthesis tool passes the same attribute to the signal in the post-synthesis netlist (*.vm).

Verilog Syntax /* synthesis syn_rvl_debug = 1 */;

Verilog Example

```
input clki /*synthesis syn_rvl_debug = 1 */,
wire clk1 /* synthesis syn_rvl_debug = 1 */;
reg [39:0] cnt /* synthesis syn_rvl_debug = 1 */;
reg [31:0] cnti /* synthesis syn_rvl_debug = 1 */;
```

VHDL Syntax

attribute syn_rvl_debug : boolean;

attribute syn_rvl_debug of sig1 : signal is true

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", but 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.

Note: If the memory size defined (depth x width) is less than thirty registers, then EBRs will not be inferred.

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 " ;

```
// 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
```

In the above syntax, *object* is a register.

VHDL 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, meaning 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/1 | off/0” */ ;

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 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;
```

The following is the source code used for the previous example.

```
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. Exploiting clock enables on registers is usually beneficial. However, in some cases, using clock enable signals may make it challenging to achieve timing closure. 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;
```

Synopsys Design Constraints Timing/ Physical Constraints

This section describes the new physical constraints in the Radiant software that are formatted similar to SDC formats to set physical constraints.

See Also ▶“ldc_create_group” on page 815

- “ldc_create_region” on page 815
- “ldc_create_macro” on page 816

- "ldc_create_vref" on page 816
- "ldc_set_location" on page 817
- "ldc_set_vcc" on page 817
- "ldc_set_port" on page 818
- "ldc_set_sysconfig" on page 818
- "ldc_set_attribute" on page 819
- "ldc_prohibit" on page 820

ldc_create_group

Description Defines a single identifier that refers to a group of objects. Only slice and IO can be created currently.

Usage ldc_create_group -name group_name [-bbox {height width}] <objects>

Options

- -name - group name
- [-bbox]: is used to optionally define the maximum number of rows (R) and columns (C), of the group's bounding box, -bbox is not applicable to port
- <objects>: objects named in the group, either port, instance, pin or net.

Example

```
ldc_create_group -name group1[get_ports{a*}]
```

ldc_create_region

Commands to define a rectangular area.

Syntax

```
ldc_create_region -name <region_name> (-anchor <anchor>|-site
                      <site>) -width <width> -height <height>
                      [-buffer_left <left>] [-buffer_right
                      <right>] [-buffer_top <top>]
                      [-buffer_bottom <bottom>] [-exclusive]
                      [-routing_exclusive]
```

Arguments

- -name region_name: This value specifies the region name.
- -anchor anchor: This value specifies anchor.
- -site site: This value specifies the site name. Option '-anchor' and '-site' are mutually exclusive.
- -width width: This value specifies the width of the placement region in columns.

- -height height: This value specifies the height of the placement region in rows.
- -buffer_left left: This value specifies the left buffer zone of the routing region (Optional).
- -buffer_right right: This value specifies the right buffer zone of the routing region (Optional).
- -buffer_top top: This value specifies the top buffer zone of the routing region (Optional).
- -buffer_bottom bottom: This value specifies the bottom buffer zone of the routing region (Optional). Options '-buffer_left','-buffer_right', '-buffer_top' and '-buffer_bottom' are only for creating macro region. Without these options routing region is identical to placement region.
- -exclusive: This option indicates that the current buffer cannot be used to place and route other logic during the Reuse stage (Optional).
- -routing_exclusive: This option indicates that the current buffer cannot be used to route other logic during the Reuse stage (Optional). Options '-exclusive' and '-routing_exclusive' are only for creating macro region.

Example

```
ldc_create_region -name top_region -site R17C2D -width 23 -
height 39
```

ldc_create_macro

Description Defines an instance in a design as a macro.

Usage ldc_create_macro -name <macro_name> -use_pio <ports_list> [get_cells <instance_path>]

Options

- -name: specifies the macro name
- -use_pio: specifies macro ports that are directly connected outside of the design.
- <ports_list>: comma-separated value.
- get_cells: specifies the hierarchical path of the instance which is defined as a macro.

Example

```
ldc_create_macro -name mac1 -use_pio {portA, portBus*}
[get_cells top/inst1]
```

ldc_create_vref

Description Define a voltage reference. The PIO site serves as the input pin for an on-chip voltage reference.

Usage ldc_create_vref -name vref_name -site site_name

Options

- -name: voltage reference name
- -site: PIO site of the target device

Example

```
ldc_create_vref -name VREF1_BANK_3 -site N21
```

ldc_set_location

Description When applied to a specified component, it places the component at a specified site or bank and locks the component to the site or bank. In this case, -site or -bank and <object> are valid combinations.

When applied to a specified group, it places the group at a specified site or within a region. In this case, -site or -region and -group are valid combinations.

Usage ldc_set_location [-site site_name]|[-bank bank_num]|[-region region_name] <object>

Options

- -site: a row/column Slice D location or PIO site of the target device
- -bank: bank number
- -region: user defined region name
- Object: object to be located

Example

```
ldc_set_location -site 11 [get_ports {A}]
ldc_set_location -region region0 [ldc_get_groups {group1}]
```

ldc_set_vcc

Commands to set the voltage and/or derate for the bank, core or dphy.

Syntax

```
ldc_set_vcc (-bank <bank_number>|-core|-dphy DPHY0/DPHY1)
            [-derate <derate>] [<voltage>]
```

Arguments

- -bank bank_number: This value specifies the bank to be set with voltage or voltage derating percent.

- -core: This option indicates that voltage or voltage derating percent is set to core.
- -dphy DPHY0/DPHY1: This option indicates that voltage or voltage derating percent is set to dphy. Value of option '-dphy' can only be 'DPHY0' or 'DPHY1'. Option '-bank', '-core' and '-dphy' are mutually exclusive.
- -derate: This value specifies voltage derating percent (Optional).
- voltage: This value can be any compatible voltage supply to that bank or the core or dphy; for example, 2.5, 3.3 (Optional). At least one option between '-derate' and 'voltage' is needed.

Examples

```
ldc_set_vcc -bank 1 3.3
ldc_set_vcc -dphy DPHY0 -derate -3
```

ldc_set_port

Description Set constraint attributes to ports; -iobuf, is used exclusively; -vref must be combined with –iobuf.

Usage ldc_set_port [-iobuf [-vref <vref_name>]] | [-sso] <key-value list> <ports>

Options

- -iobuf: set IOBUF attributes. valid key-values are available in LCT file.
- -vref: voltage reference name, must be bound to -iobuf.
- -sso: This option indicates that SSO attributes are set.
  Option ‘-iobuf’ and ‘-sso’ are mutually exclusive.
- <key-value list>: key-value list must be enclosed with curly braces.
- <ports>: ports to be set. If no port specified, set constraint to all ports.

Example

```
ldc_create_vref -name VERF2 -site R11
ldc_set_port -iobuf -vref VERF2
      {IO_TYPE=HSTL15_I VREF=VREF2_LOAD} [get_ports clk_third]
```

ldc_set_sysconfig

Description Set sysconfig attributes.

Usage ldc_set_sysconfig <key-value list>

Options <key-value list>: SYSCONFIG attributes, valid key-values are available in sysConfig file. key-value list must be enclosed with curly braces.

Example

```
ldc_set_sysconfig {JTAG_PORT=ENABLE PROGRAMN_PORT=ENABLE
MCCLK_FREQ=56.2 DONE_OD=ON}
```

ldc_set_attribute

Description Set constraint attributes to the objects or the design if no object is specified.

Usage ldc_set_attribute <key-value list> <objects>

Options key-value list: constraint attributes to be set to object(s) or design if no object specified. Valid key-values are:

Global attributes:

- FORMAT and UNIQUE_ID is a valid combination to implement USERCODE.
- FORMATSpecifies one of the available formats for codes: binary (BIN), hexadecimal (HEX), text (ASCII), AUTO.
- CODEFor 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. For text or ASCII format, specify a four character user code string using only alpha and numeric values.
- TEMPERATURE<number> (C|F|K). No <object> specified.

Net attributes:

- GSR_NET is set to TRUE / FALSE:
  - TRUE: specify the net to be GSR net (global reset source driver).
  - FALSE: don't infer the net to be GSR_NET net. <object> must be net.

  Note

  The default Synthesis strategy is "Force GSR = Auto".

  The default Map strategy is "infer GSR = FALSE".

  When using "GSR_NET" constraint, user must change the above two strategies to:

  - Synplify - "Force GSR = False",
  - LSE - "Force GSR = No"
  - MAP - "infer GSR = True"

- USE_PRIMARY is set to TRUE / FALSE.
  - TRUE: sets USE_PRIMARY net to primary clock resource.
  - FALSE: prohibits USE_PRIMARY net from using primary clock resource; <object> must be net.

- USE_PRIMARY_REGION: Comma-separated list of integers denoting primary clock region(s). Primary clock regions are numbered from top to bottom, left to right based on their position in the chip and start from 0. <object> must be net.
- USE_REGIONAL: This attribute is for the Avant device only to support the regional clock feature. When using this attribute, the Avant device can be divided into regions, and each region has its own clock resources.

  USE_REGIONAL also has TRUE / FALSE options:

  - TRUE: sets USE_REGIONAL net to primary clock resource.
  - FALSE: prohibits USE_REGIONAL net from using primary clock resource; <object> must be net.

  Generally, PAR can do automatic regional clock selection, but you can also manually assign certain signals to regional clocks. See the following example:

```
ldc_set_attribute USE_REGIONAL=TRUE [ get_nets {signal name}
]
```

Note: GSR NET is not applicable for iCE40 Ultra Plus.

Example

```
ldc_set_attribute {FORMAT=HEX CODE=536*56D69}
ldc_set_attribute {USE_PRIMARY=TRUE} {USE_PRIMARY_REGION=0,1}
[get_nets {clk1_c}]
ldc_set_attribute GSR_NET=TRUE [get_nets {my_gsr}]
```

ldc_prohibit

Description Prohibits the use of a site or all sites in a region.

Usage ldc_prohibit –site site OR ldc_prohibit –region region

Options site: site name, region: region name

Example

```
ldc_prohibit -site AB
ldc_prohibit -region regionA
```

Lattice Module Reference Guide

IP Catalog 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 up your design work and to get the most effective results.

Parameterized Module Instantiation (PMI) is an alternate way to use some of the modules that come with IP Catalog. Instead of using IP Catalog, you directly instantiate a module into your HDL and customize it by setting parameters in the HDL using PMI. As a result, you may find this easier to use than IP Catalog in certain situations.

This guide describes the modules that come with IP Catalog 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

- "PMI or IP Catalog?" on page 142
- "Creating IP Catalog Components" on page 142
- "Using PMI" on page 148

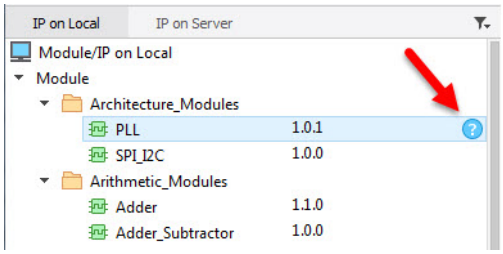
Finding Modules in This Guide

Each module and IP has an associated IP Information page.

The IP information page provides description, devices supported, link to User Guide, and revision history.

To open the IP Information page for a module or IP:

- In either the IP on Local tab, or IP on Server tab, click on the module or IP, and then click on the blue question mark . The IP Information page will appear on the right.



Parameterized Module Instantiation (PMI) The following table lists Radiant modules that support PMI. The table contains:

- PMI module name.
- IP Catalog module name.
- Links to the IP User Guides in which the PMI modules are described in detail.
- Columns that indicate PMI support for iCE40UP, LFCPNX/LFD2NX/ LFMXO5/LIFCL/UT24C/UT24CP, and LAV-AT devices.

Table 177: PMI Modules

PMI Module Name	IP Catalog Module Name	Lattice Radiant Software User Guide	iCE40UP	LFCPNX LFD2NX, LFMXO5 LIFCL UT24C UT24CP	LAV-AT
pmi_add	Adder	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_addsub	Adder Subtractor	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_complex_mult	Complex_Mult	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_counter	Counter	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_dsp	Mult_Accumulate	Arithmetic Modules User Guide	Yes	No	No
pmi_mac	Mult_Accumulate	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_multaddsub	Mult_Add_Sub	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_multaddsubsum	Mult_Add_Sub_Sum	Arithmetic Modules User Guide	Yes	Yes	Yes

Table 177: PMI Modules (Continued)

pmi_mult	Multiplier	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_sub	Subtractor	Arithmetic Modules User Guide	Yes	Yes	Yes
pmi_fifo	FIFO	Memory Modules User Guide	Yes	Yes	Yes
pmi_fifo_dc	FIFO_DC	Memory Modules User Guide	Yes	Yes	Yes
pmi_distributed_shift_reg	Shift_Register	Memory Modules User Guide	No	Yes	Yes
pmi_distributed_dpram	Distributed_DPRAM	Memory Modules User Guide	No	Yes	Yes
pmi_distributed_rom	Distributed_ROM	Memory Modules User Guide	No	Yes	Yes
pmi_distributed_spram	Distributed_SPRAM	Memory Modules User Guide	No	Yes	Yes
pmi_ram_dp	RAM_DP	Memory Modules User Guide	Yes	Yes	Yes
pmi_ram_dp_be	RAM_DP	Memory Modules User Guide	Yes	Yes	Yes
pmi_ram_dp_true	RAM_DP_True	Memory Modules User Guide	No	Yes	Yes
pmi_ram_dq	RAM_DQ	Memory Modules User Guide	Yes	Yes	Yes
pmi_ram_dq_be	RAM_DQ	Memory Modules User Guide	Yes	Yes	Yes
pmi_rom	ROM	Memory Modules User Guide	Yes	Yes	Yes

See Also

- “PMI or IP Catalog?” on page 142
- “Creating IP Catalog Components” on page 142
- “Using PMI” on page 148

FPGA Libraries Reference Guide

Lattice Semiconductor 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.

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 - iCE40UP (iCE40 UltraPlus)" on page 859
- "Primitive Library - LAV-AT (Lattice Avant)" on page 826
- "Primitive Library - LFCPNX (CertusPro-NX) / UT24CP (CertusPro-NX-RT)" on page 835
- "Primitive Library - LFD2NX (Certus-NX) / UT24C (Certus-NX-RT)" on page 841
- "Primitive Library - LFMXO5 (MachXO5-NX)" on page 847
- "Primitive Library - LIFCL (CrossLink-NX)" on page 853

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 1: Information Provided for Each Primitive

Fields	Description
Name	Primitive name
Definition	Brief description of primitive

Table 1: Information Provided for Each Primitive (Continued)

Fields	Description
Architecture Supported	List of FPGA families supported by the primitive
Port Interface Symbol	Graphic representing the primitive port interface. Data, address, clock, clock enable type ports appear on the left-hand side of the block. Synchronous ports appear on the top of the symbol. Asynchronous control ports appear on the bottom of the symbol. The output ports appear on the right-hand side of the block. Some of the graphic symbols are shown in bus notation for proper layout. Such primitives must be instantiated in expanded bus notation format with each individual bit.
Port Description	List of port names, polarity, and function.
Parameters	List of parameters 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.
Synthesis Inference Rules	Guideline on whether the component can be inferred by supported synthesis tools.
User Instantiation Rules	Guideline on whether the component can be directly instantiated in the design.

Primitive Library - LAV-AT (Lattice Avant)

This library includes compatible primitives supported by the LAV-AT devices.

- “Flip-Flops” on page 826
- “Input/Output Buffer” on page 827
- “Memories” on page 827
- “Multipliers” on page 827
- “Combinatorial Primitives” on page 828
- “Clock/PLL” on page 828
- “Interfaces” on page 828
- “Dual Data Rate Cells” on page 828
- “Miscellaneous” on page 829

Table 1: Flip-Flops

Primitive	Description
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Synchronous Clear (Clear Overrides Enable)
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Synchronous Preset (Preset Overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous clear and Positive Level Enable (Clear Overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset Overrides Enable)

Table 2: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_I3C_B	Bidirectional I3C Buffer Primitive
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State

Table 3: Memories

Primitive	Description
DP32K	32 Kb Dual Port Block RAM
DPR16X4A	Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read
DPR32X2	Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read
FIFO32K	32 Kb FIFO using block RAM
PDP32K	32 Kb pseudo-dual port block RAM
PDPSC32K	32Kb Pseudo Dual Port Single Clock Block RAM
SP16KA	16 Kb Single Port Block RAM
SP32K	32 Kb Single Port Block RAM
SPR16X4A	Distributed Single Port RAM. Synchronous write, asynchronous read
SPR32X2	Distributed Single Port RAM. Synchronous write, asynchronous read

Table 4: Multipliers

Primitive	Description
DOTPRODADDSUB9X9	Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor
DOTPRODADDSUB9X9A	Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor
MULT8X8	8x8 Multiplier with Optional Input/Output Registers
MULT9X9A	9x9 Multiplier with Optional Input/Output Registers
MULT18X18A	18x18 Multiplier with Optional Input/Output Registers
MULTADDSUB18X18A	18x18 Multiplier and Adder/Subtractor

Table 5: Combinatorial Primitives

Primitive	Description
CCU2C	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 6: Clock/PLL

Primitive	Description
DCCA	Dynamic clock control
DCSA	Dynamic clock select
ECLKDIVA	Wrapper for Clock Divider for Edge Clock
ECLKSYNCA	Wrapper for Clock Synchronizer for Edge Clock
OSCE	High-frequency oscillator
PLLC	PLL functional block
PLLREFCSA	2:1 mux with dynamic select

Table 7: Interfaces

Primitive	Description
MIPIA	Soft version of bidirectional MIPI

Table 8: Dual Data Rate Cells

Primitive	Description
DDRDLLA	DLL for creating the 90° clock delay
DDRPHY16D	Supports DDR4 and DDR3L protocols with X16 data width
DDRPHY32D	Supports DDR4 and DDR3L protocols with X32 data width
DDRPHY40D	Supports DDR4 and DDR3L protocols with X40 data width
DDRPHY64D	Supports DDR4 and DDR3L protocols with X64 data width
DDRPHY72D	Supports DDR4 and DDR3L protocols with X72 data width
DDRPHY16E	Supports LPDDR4 protocol with X16 data width
DDRPHY32E	Supports LPDDR4 protocol with X32 data width
DDRPHY64E	Supports LPDDR4 protocol with X64 data width

Table 8: Dual Data Rate Cells

Primitive	Description
DELAYD	Input or output delay for SCLK interface with dynamically control delay
DELAYE	Input delay or output delay for SCLK interface with static delay
DELAYF	Input or output delay for ECLK interface with dynamically control delay
DLLDELA	DDR RX modes to shift input clock phase by 90 degrees
IDDR71A	7:1 LVDS IDDR
IDDRX1A	Generic DDR with 1X gearing
IDDRX2A	Generic DDR with 2X gearing
IDDRX4A	Generic DDR with 4X gearing
IDDRX5A	Generic DDR with 5X gearing
IMONDELAY	ECLK interface with dynamically control delay and edge monitor (IMON)
ODDR71A	7:1 LVDS ODDR implementation
ODDRX1A	Generic x1 ODDR
ODDRX2A	Generic x2 ODDR implementation
ODDRX4A	Generic x4 ODDR implementation
ODDRX5A	Generic x5 ODDR implementation

Table 9: Miscellaneous

Primitive	Description
BUF	Non-Inverting Buffer
CONFIG_DONE	CONFIG final done status
CONFIG_LMMIC	LMMI Slave Primitive
GSRA	Global Set/Reset
INV	Inverter
JTAGA	Supports JTAG operations
JTAGH25	Supports hard JTAG operations
JTAGH25SOFT	Supports soft JTAG operations
SGSR	Synchronous GSR Primitive
TSALLB	Tri-state all PIO outputs and bidirectional PIO
VHI	Logic High Generator
VLO	Logic Low Generator
WDTA	Watchdog Timer Primitive

Primitive Library - Certus-KH (LN2-CT)

This library includes compatible primitives supported by the LAV-AT devices.

- “Flip-Flops” on page 826
- “Input/Output Buffer” on page 827
- “Memories” on page 827
- “Multipliers” on page 827
- “Combinatorial Primitives” on page 828
- “Clock/PLL” on page 828
- “Interfaces” on page 828
- “Dual Data Rate Cells” on page 828
- “Miscellaneous” on page 829

Table 10: Flip-Flops

Primitive	Description
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Synchronous Clear (Clear Overrides Enable)
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Synchronous Preset (Preset Overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous clear and Positive Level Enable (Clear Overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset Overrides Enable)

Table 11: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_I3C_B	Bidirectional I3C Buffer Primitive
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State

Table 12: Memories

Primitive	Description
DP32K	32 Kb Dual Port Block RAM
DPR16X4A	Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read
DPR32X2	Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read
FIFO32K	32 Kb FIFO using block RAM
PDP32K	32 Kb pseudo-dual port block RAM
PDPSC32K	32Kb Pseudo Dual Port Single Clock Block RAM
SP16KA	16 Kb Single Port Block RAM
SP32K	32 Kb Single Port Block RAM
SPR16X4A	Distributed Single Port RAM. Synchronous write, asynchronous read
SPR32X2	Distributed Single Port RAM. Synchronous write, asynchronous read

Table 13: Multipliers

Primitive	Description
DOTPRODADDSUB9X9	Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor
DOTPRODADDSUB9X9A	Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor
MULT8X8	8x8 Multiplier with Optional Input/Output Registers
MULT9X9A	9x9 Multiplier with Optional Input/Output Registers
MULT18X18A	18x18 Multiplier with Optional Input/Output Registers
MULTADDSUB18X18A	18x18 Multiplier and Adder/Subtractor

Table 14: Combinatorial Primitives

Primitive	Description
CCU2C	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 15: Clock/PLL

Primitive	Description
DCCA	Dynamic clock control
DCSA	Dynamic clock select
ECLKDIVA	Wrapper for Clock Divider for Edge Clock
ECLKSYNCA	Wrapper for Clock Synchronizer for Edge Clock
OSCE	High-frequency oscillator
PLLC	PLL functional block
PLLREFCSA	2:1 mux with dynamic select

Table 16: Interfaces

Primitive	Description
MIPIA	Soft version of bidirectional MIPI

Table 17: Dual Data Rate Cells

Primitive	Description
DDRDLLA	DLL for creating the 90° clock delay
DDRPHY16D	Supports DDR4 and DDR3L protocols with X16 data width
DDRPHY32D	Supports DDR4 and DDR3L protocols with X32 data width
DDRPHY40D	Supports DDR4 and DDR3L protocols with X40 data width
DDRPHY64D	Supports DDR4 and DDR3L protocols with X64 data width
DDRPHY72D	Supports DDR4 and DDR3L protocols with X72 data width
DDRPHY16E	Supports LPDDR4 protocol with X16 data width
DDRPHY32E	Supports LPDDR4 protocol with X32 data width
DDRPHY64E	Supports LPDDR4 protocol with X64 data width

Table 17: Dual Data Rate Cells

Primitive	Description
DELAYD	Input or output delay for SCLK interface with dynamically control delay
DELAYE	Input delay or output delay for SCLK interface with static delay
DELAYF	Input or output delay for ECLK interface with dynamically control delay
DLLDELA	DDR RX modes to shift input clock phase by 90 degrees
IDDR71A	7:1 LVDS IDDR
IDDRX1A	Generic DDR with 1X gearing
IDDRX2A	Generic DDR with 2X gearing
IDDRX4A	Generic DDR with 4X gearing
IDDRX5A	Generic DDR with 5X gearing
IMONDELAY	ECLK interface with dynamically control delay and edge monitor (IMON)
ODDR71A	7:1 LVDS ODDR implementation
ODDRX1A	Generic x1 ODDR
ODDRX2A	Generic x2 ODDR implementation
ODDRX4A	Generic x4 ODDR implementation
ODDRX5A	Generic x5 ODDR implementation

Table 18: Miscellaneous

Primitive	Description
BUF	Non-Inverting Buffer
CONFIG_DONE	CONFIG final done status
CONFIG_LMMIC	LMMI Slave Primitive
GSRA	Global Set/Reset
INV	Inverter
JTAGA	Supports JTAG operations
JTAGH25	Supports hard JTAG operations
JTAGH25SOFT	Supports soft JTAG operations
SGSR	Synchronous GSR Primitive
TSALLB	Tri-state all PIO outputs and bidirectional PIO
VHI	Logic High Generator
VLO	Logic Low Generator
WDTA	Watchdog Timer Primitive

Primitive Library - LFCPNX (CertusPro-NX) / UT24CP (CertusPro-NX-RT)

This library includes compatible primitives supported by the LFCPNX (CertusPro-NX) / UT24CP (CertusPro-NX-RT) device family.

- "Adders/Subtractors" on page 835
- "Flip-Flops" on page 835
- "Input/Output Buffer" on page 836
- "Memories" on page 836
- "Multipliers" on page 837
- "Clock/PLL" on page 837
- "Combinatorial Primitives" on page 838
- "Interfaces" on page 838
- "Dual Data Rate Cells" on page 838
- "Oscillators" on page 840
- "Miscellaneous" on page 840

Table 1: Adders/Subtractors

Primitive	Description
ACC54	54-bit Accumulator for DSP

Table 2: Flip-Flops

Primitive	Description
BFD1P3KX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Synchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
BFD1P3LX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Asynchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear

Table 2: Flip-Flops

Primitive	Description
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 3: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_ADC	Bidirectional Buffer for Analog-to-Digital Converter
BB_CDR	Bidirectional Buffer for CDR
BB_I3C_A	Input/Output Buffer with User Controllable Pull-Up Resistors
DIFFIO18A	Single Ended + Differential I/O block for 1.8V block
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State
SEIO18A	Single Ended I/O block for 1.8V
SEIO33	Single Ended I/O Block for 3.3V

Table 4: Memories

Primitive	Description
DP16K	16 kb Dual Port Block RAM
DPR16X4	Distributed Pseudo Dual Port RAM with Synchronous Write and Asynchronous Read
DPSC512K	512 kb Single Clock Dual Port Block RAM
FIFO16K	16 kb FIFO

Table 4: Memories

Primitive	Description
PDP16K	16 kb Pseudo Dual Port Block RAM
PDPSC16K	16 kb Pseudo Dual Port Single Clock Block RAM
PDPSC512K	512 kb Single Clock Pseudo Dual Port Block RAM
SP16K	16 kb Single Port Block RAM
SP512K	512 kb Single Port Block RAM
SPR16X4	Distributed Single Port RAM with Synchronous Write and Asynchronous Read

Table 5: Multipliers

Primitive	Description
ALUREG	Arithmetic Logic Unit and Register File for RISCV Soft Processor
MULT18X18	18x18 Multiplier with Optional Input/Output Registers
MULT18X36	18x36 Multiplier with Optional Input/Output Registers
MULT36X36	36x36 Multiplier with Optional Input/Output Registers
MULT9X9	9x9 Multiplier with Optional Input/Output Registers
MULT18X18	18x18 Multiplier and Accumulator
MULTADDSUB18X18WIDE	18x18 Wide Multiplier and Adder/Subractor
MULTADDSUB18X36	18x36 Multiplier and Adder/Subtractor
MULTADDSUB36X36	36x36 Multiplier and Adder/Subtractor
MULTADDSUB9X9WIDE	9x9 Wide Multiplier and Adder/Subtractor
MULTPREADD18X18	18x18 Multiplier with Pre-Adder
MULTPREADD9X9	9x9 Multiplier with Pre-Adder

Table 6: Clock/PLL

Primitive	Description
DCC	Dynamic Clock Control with Positive Enable
DCS	Dynamic Clock Selection
DIFFCLKIO	Differential to single-end clock buffer.
ECLKDIV	Clock Divider for Edge Clock Wrapper
ECLKSYNC	Clock Synchronizer for Edge Clock Wrapper
PCLKDIV	Clock Divider for Primary Clock

Table 6: Clock/PLL

Primitive	Description
PCLKDIVSP	Clock Divider for Primary Clock
PCSREFMUX	Multiplexer of reference clock for PCS
PLL	Phase-Locked Loop
PLLA	Phase-Locked Loop
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 7: Combinatorial Primitives

Primitive	Description
CCU2	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 8: Interfaces

Primitive	Description
I2CFIFO	Hardened I2C Interface Block
MIPI	Soft MIPI Interface Block
MULTIBOOT	Interface for Multiboot Functionality
PCIELL	PCIe logical link layer
PCSX1	Wrapper of PCSX1 and MPCS
PCSX2	Wrapper of PCSX2 and MPCS
PCSX4	Wrapper of PCSX4 and MPCS
SEDC	Interface for Soft Error Detection and Correction Functionality
SGMIICDR	Serial Gigabit Media Independent Interface

Table 9: Dual Data Rate Cells

Primitive	Description
DDRDLL	Delay-Locked Loop Master Block for DDR Functionality
DELAYA	Dynamic Input/Output Delay Element
DELAYB	Static Input/Output Delay Element

Table 9: Dual Data Rate Cells

Primitive	Description
DLLDEL	Delay-Locked Loop Slave Block for DDR Functionality
DQSBUF	Block generating DQS Signal for Memory DDR Functionality
DQSBUFA	Block generating DQS Signal for Memory LPDDR4 Functionality
DQSBUF_IVREF	Block generating DQS Signal for Memory DDR4 Functionality
IDDR71	7:1 LVDS IDDR
IDDRX1	Generic X1 IDDR
IDDRX2	Generic X1 IDDR
IDDRX2DQ	DQ Input for DDR2 & DDR3 memory
IDDRX4	Generic X4 IDDR
IDDRX4DQ	DQ Input for DDR3 memory
IDDRX5	Generic X5 IDDR
ODDR71	7:1 LVDS ODDR
ODDRX1	Generic X1 ODDR
ODDRX2	Generic X2 ODDR
ODDRX2DQ	DQ output for DDR2 & DDR3 memory
ODDRX2DQS	DQS Output for DDR2 & DDR3 memory
ODDRX4	Generic X4 ODDR
ODDRX4DQ	DQ Input for DDR3 memory
ODDRX4DQS	DQS Output for DDR3 memory
ODDRX5	Generic X5 ODDR
OSHX2	This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling
OSHX4	This primitive is used to generate the address and command for DDR3 memory with x4 gearing and write leveling
OUTDELAYA	Add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode
OUTDELAYB	Add static delay to DOUT and TOUT for write training in LPDDR4 mode.
TSHX2DQ	DQS tristate control for DDR2 & DDR3 memory
TSHX2DQS	DQS tristate control for DDR2 & DDR3 memory
TSHX4DQ	DQS tristate control for DDR3 memory
TSHX4DQS	DQS tristate control for DDR3 memory

Table 10: Oscillators

Primitive	Description
OSC	Oscillator Block
OSCA	Oscillator Block

Table 11: Miscellaneous

Primitive	Description
ADC	Analog-to-Digital Converter
BNKREF18	Bank Reference Block for 1.8V I/O
BUF	Non-Inverting Buffer
CONFIG_LMMIA	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CRE	Cryptographic Engine
GSR	Global Set/Reset
INV	Inverter
JTAG	JTAG Block for Reveal Debugging Tool
TSALLA	Global Tristate Interface
VHI	Logic High Generator
VLO	Logic Low Generator
WDT	Interface for Configuration Watchdog Timer

Primitive Library - LFD2NX (Certus-NX) / UT24C (Certus-NX-RT)

This library includes compatible primitives supported by the LFD2NX (Certus-NX) / UT24C (Certus-NX-RT) device family.

- "Adders/Subtractors" on page 841
- "Flip-Flops" on page 841
- "Input/Output Buffer" on page 842
- "Memories" on page 842
- "Multipliers" on page 843
- "Clock/PLL" on page 843
- "Combinatorial Primitives" on page 844
- "Interfaces" on page 844
- "Dual Data Rate Cells" on page 844
- "Oscillators" on page 845
- "Miscellaneous" on page 846

Table 1: Adders/Subtractors

Primitive	Description
ACC54	54-bit Accumulator for DSP

Table 2: Flip-Flops

Primitive	Description
BFD1P3KX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Synchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
BFD1P3LX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Asynchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear

Table 2: Flip-Flops

Primitive	Description
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 3: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_ADC	Bidirectional Buffer for Analog-to-Digital Converter
BB_CDR	Bidirectional Buffer for CDR
BB_I3C_A	Input/Output Buffer with User Controllable Pull-Up Resistors
DIFFIO18	Single Ended + Differential I/O block for 1.8V block
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State
SEIO18	Single Ended I/O block for 1.8V
SEIO33	Single Ended I/O Block for 3.3V

Table 4: Memories

Primitive	Description
DP16K	16 kb Dual Port Block RAM
DPR16X4	Distributed Pseudo Dual Port RAM with Synchronous Write and Asynchronous Read
DPSC512K	512 kb Single Clock Dual Port Block RAM
FIFO16K	16 kb FIFO

Table 4: Memories

Primitive	Description
PDP16K	16 kb Pseudo Dual Port Block RAM
PDPSC16K	16 kb Pseudo Dual Port Single Clock Block RAM
PDPSC512K	512 kb Single Clock Pseudo Dual Port Block RAM
SP16K	16 kb Single Port Block RAM
SP512K	512 kb Single Port Block RAM
SPR16X4	Distributed Single Port RAM with Synchronous Write and Asynchronous Read

Table 5: Multipliers

Primitive	Description
ALUREG	Arithmetic Logic Unit and Register File for RISCV Soft Processor
MULT18X18	18x18 Multiplier with Optional Input/Output Registers
MULT18X36	18x36 Multiplier with Optional Input/Output Registers
MULT36X36	36x36 Multiplier with Optional Input/Output Registers
MULT9X9	9x9 Multiplier with Optional Input/Output Registers
MULTADDSUB18X18	18x18 Multiplier and Accumulator
MULTADDSUB18X18WIDE	18x18 Wide Multiplier and Adder/Subractor
MULTADDSUB18X36	18x36 Multiplier and Adder/Subtractor
MULTADDSUB36X36	36x36 Multiplier and Adder/Subtractor
MULTADDSUB9X9WIDE	9x9 Wide Multiplier and Adder/Subtractor
MULTPREADD18X18	18x18 Multiplier with Pre-Adder
MULTPREADD9X9	9x9 Multiplier with Pre-Adder

Table 6: Clock/PLL

Primitive	Description
DCC	Dynamic Clock Control with Positive Enable
DCS	Dynamic Clock Selection
ECLKDIV	Clock Divider for Edge Clock Wrapper
ECLKSYNC	Clock Synchronizer for Edge Clock Wrapper
PCLKDIV	Clock Divider for Primary Clock
PCLKDIVSP	Clock Divider for Primary Clock

Table 6: Clock/PLL

Primitive	Description
PLL	Phase-Locked Loop
PLLA	Phase-Locked Loop
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 7: Combinatorial Primitives

Primitive	Description
CCU2	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 8: Interfaces

Primitive	Description
I2CFIFO	Hardened I2C Interface Block
MIPI	Soft MIPI Interface Block
MULTIBOOT	Interface for Multiboot Functionality
PCIE	Hardened PCIE Interface
SEDC	Interface for Soft Error Detection and Correction Functionality
SGMIICDR	Serial Gigabit Media Independent Interface

Table 9: Dual Data Rate Cells

Primitive	Description
DDRDLL	Delay-Locked Loop Master Block for DDR Functionality
DELAYA	Dynamic Input/Output Delay Element
DELAYB	Static Input/Output Delay Element
DELAYC	Supports edge monitor feature
DLLDEL	Delay-Locked Loop Slave Block for DDR Functionality
DQSBUF	Block generating DQS Signal for Memory DDR Functionality
IDDR71	7:1 LVDS IDDR
IDDRX1	Generic X1 IDDR

Table 9: Dual Data Rate Cells

Primitive	Description
IDDRX2	Generic X1 IDDR
IDDRX2DQ	DQ Input for DDR2 & DDR3 memory
IDDRX4	Generic X4 IDDR
IDDRX4DQ	DQ Input for DDR3 memory
IDDRX5	Generic X5 IDDR
ODDR71	7:1 LVDS ODDR
ODDRX1	Generic X1 ODDR
ODDRX2	Generic X2 ODDR
ODDRX2DQ	DQ output for DDR2 & DDR3 memory
ODDRX2DQS	DQS Output for DDR2 & DDR3 memory
ODDRX4	Generic X4 ODDR
ODDRX4DQ	DQ Input for DDR3 memory
ODDRX4DQS	DQS Output for DDR3 memory
ODDRX5	Generic X5 ODDR
OSHX2	This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling
OSHX4	This primitive is used to generate the address and command for DDR3 memory with x4 gearing and write leveling
TSHX2DQ	DQS tristate control for DDR2 & DDR3 memory
TSHX2DQS	DQS tristate control for DDR2 & DDR3 memory
TSHX4DQ	DQS tristate control for DDR3 memory
TSHX4DQS	DQS tristate control for DDR3 memory

Table 10: Oscillators

Primitive	Description
OSC	Oscillator Block
OSCA	Oscillator Block

Table 11: Miscellaneous

Primitive	Description
ADC	Analog-to-Digital Converter
BNKREF18	Bank Reference Block for 1.8V I/O
BUF	Non-Inverting Buffer
CRE	Cryptographic Engine
CONFIG_LMMI	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CONFIG_LMMIG	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CONFIG_LMMIH	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
GSR	Global Set/Reset
INV	Inverter
JTAG	JTAG Block for Reveal Debugging Tool
TSALLA	Global Tristate Interface
VHI	Logic High Generator
VLO	Logic Low Generator
WDT	Interface for Configuration Watchdog Timer

Primitive Library - LFMXO5 (MachXO5-NX)

This library includes compatible primitives supported by the LFMXO5 (MachXO5-NX) device family.

- "Adders/Subtractors" on page 847
- "Flip-Flops" on page 847
- "Input/Output Buffer" on page 848
- "Memories" on page 848
- "Multipliers" on page 849
- "Clock/PLL" on page 849
- "Combinatorial Primitives" on page 850
- "Interfaces" on page 850
- "Dual Data Rate Cells" on page 850
- "Oscillators" on page 852
- "Miscellaneous" on page 852

Table 1: Adders/Subtractors

Primitive	Description
ACC54	54-bit Accumulator for DSP

Table 2: Flip-Flops

Primitive	Description
BFD1P3KX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Synchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
BFD1P3LX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Asynchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)

Table 2: Flip-Flops

Primitive	Description
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 3: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_ADC	Bidirectional Buffer for Analog-to-Digital Converter
BB_CDR	Bidirectional Buffer for CDR
BB_I3C_A	Input/Output Buffer with User Controllable Pull-Up Resistors
DIFFIO18	Single Ended + Differential I/O block for 1.8V block
DIFFIO18A	Single Ended + Differential I/O block for 1.8V block
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State
SEIO18	Single Ended I/O block for 1.8V
SEIO18A	Single Ended I/O block for 1.8V
SEIO33	Single Ended I/O Block for 3.3V

Table 4: Memories

Primitive	Description
DP16K	16 kb Dual Port Block RAM
DPR16X4	Distributed Pseudo Dual Port RAM with Synchronous Write and Asynchronous Read
DPSC512K	512 kb Single Clock Dual Port Block RAM
FIFO16K	16 kb FIFO

Table 4: Memories

Primitive	Description
PDP16K	16 kb Pseudo Dual Port Block RAM
PDPSC16K	16 kb Pseudo Dual Port Single Clock Block RAM
PDPSC512K	512 kb Single Clock Pseudo Dual Port Block RAM
SP16K	16 kb Single Port Block RAM
SP512K	512 kb Single Port Block RAM
SPR16X4	Distributed Single Port RAM with Synchronous Write and Asynchronous Read

Table 5: Multipliers

Primitive	Description
ALUREG	Wrapper for Arithmetic Logic Unit and Register File for RISC-V Soft Processor
MULT18X18	18x18 Multiplier with Optional Input/Output Registers
MULT18X36	18x36 Multiplier with Optional Input/Output Registers
MULT36X36	36x36 Multiplier with Optional Input/Output Registers
MULT9X9	9x9 Multiplier with Optional Input/Output Registers
MULTADDSUB18X18	18x18 Multiplier and Accumulator
MULTADDSUB18X18WIDE	18x18 Wide Multiplier and Adder/Subractor
MULTADDSUB18X36	18x36 Multiplier and Adder/Subtractor
MULTADDSUB36X36	36x36 Multiplier and Adder/Subtractor
MULTADDSUB9X9WIDE	9x9 Wide Multiplier and Adder/Subtractor
MULTPREADD18X18	18x18 Multiplier with Pre-Adder
MULTPREADD9X9	9x9 Multiplier with Pre-Adder

Table 6: Clock/PLL

Primitive	Description
DCC	Dynamic Clock Control with Positive Enable
DCS	Dynamic Clock Selection
DIFFCLKIO	Differential to single-end clock buffer.
ECLKDIV	Clock Divider for Edge Clock Wrapper
ECLKSYNC	Clock Synchronizer for Edge Clock Wrapper
PCLKDIV	Clock Divider for Primary Clock

Table 6: Clock/PLL

Primitive	Description
PCLKDIVSP	Clock Divider for Primary Clock
PCSREFMUX	Multiplexer of reference clock for PCS
PLL	Phase-Locked Loop
PLLA	Phase-Locked Loop
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 7: Combinatorial Primitives

Primitive	Description
CCU2	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 8: Interfaces

Primitive	Description
I2CFIFO	Hardened I2C Interface Block
MIPI	Soft MIPI Interface Block
MULTIBOOT	Interface for Multiboot Functionality
PCIELL	PCIe logical link layer
PCSX1	Wrapper of PCSX1 and MPCS
PCSX2	Wrapper of PCSX2 and MPCS
PCSX4	Wrapper of PCSX4 and MPCS
SEDC	Interface for Soft Error Detection and Correction Functionality
SGMIICDR	Serial Gigabit Media Independent Interface

Table 9: Dual Data Rate Cells

Primitive	Description
DDRDLL	Delay-Locked Loop Master Block for DDR Functionality
DELAYA	Dynamic Input/Output Delay Element
DELAYB	Static Input/Output Delay Element

Table 9: Dual Data Rate Cells

Primitive	Description
DELAYC	Supports edge monitor feature
DLLDEL	Delay-Locked Loop Slave Block for DDR Functionality
DQSBUF	Block generating DQS Signal for Memory DDR Functionality
DQSBUFA	Block generating DQS Signal for Memory LPDDR4 Functionality
DQSBUF_IVREF	Block generating DQS Signal for Memory DDR4 Functionality
IDDR71	7:1 LVDS IDDR
IDDRX1	Generic X1 IDDR
IDDRX2	Generic X1 IDDR
IDDRX2DQ	DQ Input for DDR2 & DDR3 memory
IDDRX4	Generic X4 IDDR
IDDRX4DQ	DQ Input for DDR3 memory
IDDRX5	Generic X5 IDDR
ODDR71	7:1 LVDS ODDR
ODDRX1	Generic X1 ODDR
ODDRX2	Generic X2 ODDR
ODDRX2DQ	DQ output for DDR2 & DDR3 memory
ODDRX2DQS	DQS Output for DDR2 & DDR3 memory
ODDRX4	Generic X4 ODDR
ODDRX4DQ	DQ Input for DDR3 memory
ODDRX4DQS	DQS Output for DDR3 memory
ODDRX5	Generic X5 ODDR
OSHX2	This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling
OSHX4	This primitive is used to generate the address and command for DDR3 memory with x4 gearing and write leveling
OUTDELAYA	Add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode.
OUTDELAYB	Add static delay to DOUT and TOUT for write training in LPDDR4 mode.
OUTDELAYC	Add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode.
TSHX2DQ	DQS tristate control for DDR2 & DDR3 memory
TSHX2DQS	DQS tristate control for DDR2 & DDR3 memory
TSHX4DQ	DQS tristate control for DDR3 memory
TSHX4DQS	DQS tristate control for DDR3 memory

Table 10: Oscillators

Primitive	Description
OSC	Oscillator Block
OSCA	Oscillator Block

Table 11: Miscellaneous

Primitive	Description
ADC	Analog-to-Digital Converter
BNKREF18	Bank Reference Block for 1.8V I/O
BUF	Non-Inverting Buffer
CONFIG_LMMIB	Includes Flash access through LMMI
CONFIG_LMMIE	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CONFIG_LMMIF	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CRE	Cryptographic Engine
GSR	Global Set/Reset
INV	Inverter
JTAG	JTAG Block for Reveal Debugging Tool
TSALLA	Global Tristate Interface
VHI	Logic High Generator
VLO	Logic Low Generator
WDT	Interface for Configuration Watchdog Timer

Primitive Library - LIFCL (CrossLink-NX)

This library includes compatible primitives supported by the LIFCL (CrossLink-NX) device family.

- “Adders/Subtractors” on page 853
- “Flip-Flops” on page 853
- “Input/Output Buffer” on page 854
- “Memories” on page 854
- “Multipliers” on page 855
- “Clock/PLL” on page 855
- “Combinatorial Primitives” on page 856
- “Interfaces” on page 856
- “Dual Data Rate Cells” on page 856
- “Oscillators” on page 857
- “Miscellaneous” on page 858

Table 1: Adders/Subtractors

Primitive	Description
ACC54	54-bit Accumulator for DSP

Table 2: Flip-Flops

Primitive	Description
BFD1P3KX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Synchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
BFD1P3LX	Positive Edge Triggered Bidirectional D-Flip Flop with Positive Level Enable and Asynchronous Set/Reset for Input/Output/Tri-State Signals (to/from I/O)
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)
IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)

Table 2: Flip-Flops

Primitive	Description
IFD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 3: Input/Output Buffer

Primitive	Description
BB	Bidirectional Buffer with Tri-State
BB_ADC	Single-Ended I/O Block for 1.8V
BB_CDR	Single-Ended I/O Block for 1.8V
BB_I3C_A	Single-Ended I/O Block for 3.3V
BB_PROGRAMN	Single-Ended I/O Block for PROGRAMN
DIFFIO18	Single Ended + Differential I/O block for 1.8V block
IB	Input Buffer
OB	Output Buffer
OBZ	Output Buffer with Tri-State
SEIO18	Single Ended I/O block for 1.8V
SEIO33	Single Ended I/O Block for 3.3V

Table 4: Memories

Primitive	Description
DP16K	16 kb Dual Port Block RAM
DPR16X4	Distributed Pseudo Dual Port RAM with Synchronous Write and Asynchronous Read
DPSC512K	512 kb Single Clock Dual Port Block RAM
FIFO16K	16 kb FIFO

Table 4: Memories

Primitive	Description
PDP16K	16 kb Pseudo Dual Port Block RAM
PDPSC16K	16 kb Pseudo Dual Port Single Clock Block RAM
PDPSC512K	512 kb Single Clock Pseudo Dual Port Block RAM
SP16K	16 kb Single Port Block RAM
SP512K	512 kb Single Port Block RAM
SPR16X4	Distributed Single Port RAM with Synchronous Write and Asynchronous Read

Table 5: Multipliers

Primitive	Description
ALUREG	Arithmetic Logic Unit and Register File for RISCV Soft Processor
MULT18X18	18x18 Multiplier with Optional Input/Output Registers
MULT18X36	18x36 Multiplier with Optional Input/Output Registers
MULT36X36	36x36 Multiplier with Optional Input/Output Registers
MULT9X9	9x9 Multiplier with Optional Input/Output Registers
MULTADDSUB18X18	18x18 Multiplier and Accumulator
MULTADDSUB18X18WIDE	18x18 Wide Multiplier and Adder/Subractor
MULTADDSUB18X36	18x36 Multiplier and Adder/Subtractor
MULTADDSUB36X36	36x36 Multiplier and Adder/Subtractor
MULTADDSUB9X9WIDE	9x9 Wide Multiplier and Adder/Subtractor
MULTPREADD18X18	18x18 Multiplier with Pre-Adder
MULTPREADD9X9	9x9 Multiplier with Pre-Adder

Table 6: Clock/PLL

Primitive	Description
DCC	Dynamic Clock Control with Positive Enable
DCS	Dynamic Clock Selection
ECLKDIV	Clock Divider for Edge Clock Wrapper
ECLKSYNC	Clock Synchronizer for Edge Clock Wrapper
PCLKDIV	Clock Divider for Primary Clock
PCLKDIVSP	Clock Divider for Primary Clock

Table 6: Clock/PLL

Primitive	Description
PLL	Phase-Locked Loop
PLLA	Phase-Locked Loop
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 7: Combinatorial Primitives

Primitive	Description
CCU2	Carry Chain
LUT4	4-Input Look Up Table
WIDEFN9	9-Input Single Output Logic Block

Table 8: Interfaces

Primitive	Description
DPHY	Hardened MIPI DPHY Interface Block
I2CFIFO	Hardened I2C Interface Block
MIPI	Soft MIPI Interface Block
MULTIBOOT	Interface for Multiboot Functionality
PCIE	Hardened PCIE Interface
SEDC	Interface for Soft Error Detection and Correction Functionality
SGMIICDR	Serial Gigabit Media Independent Interface

Table 9: Dual Data Rate Cells

Primitive	Description
DDRDLL	Delay-Locked Loop Master Block for DDR Functionality
DELAYA	Dynamic Input/Output Delay Element
DELAYB	Static Input/Output Delay Element
DELAYC	Supports edge monitor feature
DLLDEL	Delay-Locked Loop Slave Block for DDR Functionality
DQSBUF	Block generating DQS Signal for Memory DDR Functionality
IDDR71	7:1 LVDS IDDR

Table 9: Dual Data Rate Cells

Primitive	Description
IDDRX1	Generic X1 IDDR
IDDRX2	Generic X1 IDDR
IDDRX2DQ	DQ Input for DDR2 & DDR3 memory
IDDRX4	Generic X4 IDDR
IDDRX4DQ	DQ Input for DDR3 memory
IDDRX5	Generic X5 IDDR
ODDR71	7:1 LVDS ODDR
ODDRX1	Generic X1 ODDR
ODDRX2	Generic X2 ODDR
ODDRX2DQ	DQ output for DDR2 & DDR3 memory
ODDRX2DQS	DQS Output for DDR2 & DDR3 memory
ODDRX4	Generic X4 ODDR
ODDRX4DQ	DQ Input for DDR3 memory
ODDRX4DQS	DQS Output for DDR3 memory
ODDRX5	Generic X5 ODDR
OSHX2	This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling
OSHX4	This primitive is used to generate the address and command for DDR3 memory with x4 gearing and write leveling
TSHX2DQ	DQS tristate control for DDR2 & DDR3 memory
TSHX2DQS	DQS tristate control for DDR2 & DDR3 memory
TSHX4DQ	DQS tristate control for DDR3 memory
TSHX4DQS	DQS tristate control for DDR3 memory

Table 10: Oscillators

Primitive	Description
OSC	Oscillator Block
OSCA	Oscillator Block
OSCD	High-frequency oscillator

Table 11: Miscellaneous

Primitive	Description
ADC	Analog-to-Digital Converter
AON	Always On
BNKREF18	Bank Reference Block for 1.8V I/O
BUF	Non-Inverting Buffer
CRE	Cryptographic Engine
CONFIG_LMMI	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
CONFIG_LMMIA	LMMI (Lattice Memory Mapped Interface) interface to the configuration block
GSR	Global Set/Reset
INV	Inverter
JTAG	JTAG Block for Reveal Debugging Tool
TSALLA	Global Tristate Interface
USB23	The software interface of USB2.0 & USB3.0 hard IP
VHI	Logic High Generator
VLO	Logic Low Generator
WDT	Interface for Configuration Watchdog Timer

Primitive Library - iCE40UP (iCE40 UltraPlus)

This library includes compatible primitives supported by the iCE40UP (iCE40 UltraPlus) device family. For further information, a variety of technical notes for the iCE40 UltraPlus family are available on the Lattice website.

- "Adders/Subtractors" on page 859
- "Flip-Flops" on page 859
- "Input/Output Buffer" on page 860
- "Memory" on page 860
- "Multiplier" on page 860
- "PIC Cells" on page 860
- "Special Cells" on page 861
  - "Clock/PLL" on page 861
  - "Combinatorial Primitives" on page 861
  - "Interfaces" on page 861
  - "Oscillators" on page 861
  - "Miscellaneous" on page 862

Table 1: Adders/Subtractors

Primitive	Description
CCU2_B	Carry Chain.
FA2	Carry chain two bit full adder. USER INSTANTIATION: Not Recommended; prefer RTL synthesis inference.

Table 2: Flip-Flops

Primitive	Description
FD1P3BZ	Positive edge triggered D flip-flop with positive level enable and positive level asynchronous preset.
FD1P3DZ	Positive edge triggered D flip-flop with positive level enable and positive level asynchronous clear.
FD1P3IZ	Positive edge triggered D flip-flop with positive level synchronous clear and positive level enable (clear overrides enable).
FD1P3JZ	Positive edge triggered D flip-flop with positive level synchronous preset and positive level enable (enable overrides preset).
FD1P3XZ	Positive edge triggered D flip-flop with synchronous or asynchronous set/reset.

Table 3: Input/Output Buffer

Primitive	Description
BB_B	Bidirectional I/O buffer with tri-state.
IB	Input buffer.
OB	Output buffer.
OBZ_B	Output buffer with tristate.
OB_RGB	Output Buffer. USER INSTANTIATION: Illegal. Please refer to RGB and RGB1P8V.

Table 4: Memory

Primitive	Description
EBR_B	4 Kb pseudo-dual port block ram with configurable write/read width & optional bitstream initialization. USER INSTANTIATION: Not recommended; prefer either RTL synthesis inference or IP Generation Tool.
PDP4K	4Kb pseudo-dual port block RAM.
SP256K	Single Port RAM that can be configured in 16K x 16 mode. This block can be cascaded using logic implemented in fabric to create larger memories.
VFB_B	256Kb single port RAM. USER INSTANTIATION: Not recommended. Please refer to SP256K.

Table 5: Multiplier

Primitive	Description
MAC16	DSP block capable of being configured as a multiplier, adder, subtractor, accumulator, multiply-adder or multiply-subtractor.

Table 6: PIC Cells

Primitive	Description
IFD1P3AZ	Positive edge triggered input D flip-flop with positive level enable.
IOL_B	Input/output registers, for both single data and double data rate.

Special Cells

Table 7: Clock/PLL

Primitive	Description
PLL_B	Phase locked loop. **For internal use**. USER INSTANTIATION: Not recommended; prefer IP Generation Tool.

Table 8: Combinatorial Primitives

Primitive	Description
LUT4	4-Input Look Up Table.

Table 9: Interfaces

Primitive	Description
I2C_B	The iCE40 UltraPlus device supports two I2C hard IP primitives.
BB_I3C	I/O with user controllable pull up resistors.
SPI_B	Hard SPI interface.

Table 10: Oscillators

Primitive	Description
HSOSC	High-frequency oscillator. Generates 48MHz nominal clock, ±10%, with user programmable divider. Can drive global clock network or fabric routing.
HSOSC1P8V	High-frequency oscillator. Generates 48MHz nominal clock, ±10%, with user programmable divider. Can drive global clock network or fabric routing. For use when VPP_2V5 is connected to a voltage below 2.3V.
HSOSC_CORE	High-frequency oscillator. Generates 48MHz nominal clock, ±10%, with user programmable divider. Can drive global clock network or fabric routing.
LSOSC	Low-frequency oscillator. Generates 10KHz nominal clock, ±10%. Can drive global clock network or fabric routing.
LSOSC1P8V	Low-frequency oscillator. Generates 10KHz nominal clock, ±10%. Can drive global clock network or fabric routing. For use when VPP_2V5 is connected to a voltage below 2.3V.
LSOSC_CORE	Low-frequency oscillator. Generates 10KHz nominal clock, ±10%. Can drive global clock network or fabric routing.

Table 11: Miscellaneous

Primitive	Description
BB_OD	Input/Output buffer.
BUF	Non-inverting buffer. USER INSTANTIATION: Yes; refer to Template Editor.
FILTER	Adds a 50ns delay to a signal.
INV	Inverter.
OFD1P3AZ	Positive edge triggered output D flip-flop with positive level enable.
RGB	RGB LED drive module, containing three open drain I/O pins for RGB LED outputs.
RGB1P8V	RGB LED drive module, containing three open drain I/O pins for RGB LED outputs. Trim bits are driven from Fabric.
RGB_CORE	RGB LED drive module, containing three open drain I/O pins for RGB LED outputs.
RGBPWM	Generates the PWM signals for the RGB LED drivers.
VHI	Logic High Generator.
VLO	Logic Low Generator.
WARMBOOT	Allows for loading a different configuration during run time.

iCECube2 Primitives vs. Radiant Software Primitives

For complete information on updating iCECube2 Primitives for iCE40 UltraPlus devices to Radiant Software Primitives for iCE40 UltraPlus devices, refer to the *Migrating iCEcube2 iCE40 UltraPlus Designs to Lattice Radiant 2.0 Software* document.

Alphanumeric Primitives List

This section lists all the Radiant library primitives in alphanumeric order.

- “ACC54” on page 869
- “ADC” on page 873
- “ALUREG” on page 877
- “AON” on page 879
- “BB” on page 881
- “BB_ADC” on page 882
- “BB_B” on page 883
- “BB_CDR” on page 884
- “BB_I3C” on page 884
- “BB_I3C_A” on page 886
- “BB_I3C_B” on page 887
- “BB_OD” on page 888
- “BB_PROGRAMN” on page 889
- “BFD1P3KX” on page 890
- “BFD1P3LX” on page 892
- “BNKREF18” on page 894
- “BUF” on page 895
- “CCU2” on page 896
- “CCU2_B” on page 898
- “CCU2C” on page 903
- “CONFIG_DONE” on page 893
- “CONFIG_LMMI” on page 905
- “CONFIG_LMMIA” on page 907
- “CONFIG_LMMIB” on page 908
- “CONFIG_LMMIC” on page 909
- “CONFIG_LMMIE” on page 910
- “CONFIG_LMMIF” on page 901
- “CONFIG_LMMIG” on page 902
- “CONFIG_LMMIH” on page 903
- “CRE” on page 912
- “CREA” on page 906
- “DCC” on page 922
- “DCCA” on page 923
- “DCS” on page 924

- “DCSA” on page 925
- “DDRDLL” on page 926
- “DDRDLLA” on page 928
- “DDRPHY16D” on page 929
- “DDRPHY16E” on page 936
- “DDRPHY32D” on page 942
- “DDRPHY32E” on page 950
- “DDRPHY40D” on page 957
- “DDRPHY64D” on page 967
- “DDRPHY64E” on page 980
- “DDRPHY72D” on page 990
- “DELAYA” on page 1003
- “DELAYB” on page 1006
- “DELAYC” on page 1008
- “DELAYD” on page 1010
- “DELAYE” on page 1012
- “DELAYF” on page 1013
- “DIFFCLKIO” on page 1015
- “DIFFIO18” on page 1016
- “DIFFIO18A” on page 1018
- “DLLDEL” on page 1020
- “DLLDELA” on page 1021
- “DOTPRODADDSUB9X9” on page 1023
- “DOTPRODADDSUB9X9A” on page 1029
- “DP16K” on page 1034
- “DP32K” on page 1037
- “DPHY” on page 1040
- “DPR16X4” on page 1053
- “DPR16X4A” on page 1055
- “DPR32X2” on page 1056
- “DPSC512K” on page 1057
- “DQSBUF” on page 1060
- “DQSBUFA” on page 1065
- “DQSBUF_IVREF” on page 1069
- “EBR_B” on page 1074
- “ECLKDIV” on page 1077

- “ECLKDIVA” on page 1078
- “ECLKSYNC” on page 1079
- “ECLKSYNCA” on page 1080
- “FA2” on page 1081
- “FD1P3BX” on page 1086
- “FD1P3BZ” on page 1088
- “FD1P3DX” on page 1089
- “FD1P3DZ” on page 1090
- “FD1P3IX” on page 1091
- “FD1P3IZ” on page 1092
- “FD1P3JX” on page 1093
- “FD1P3JZ” on page 1095
- “FD1P3XZ” on page 1096
- “FIFO16K” on page 1097
- “FIFO32K” on page 1100
- “FILTER” on page 1102
- “GSR” on page 1102
- “GSRA” on page 1103
- “HSOSC” on page 1104
- “HSOSC1P8V” on page 1105
- “HSOSC_CORE” on page 1106
- “I2C_B” on page 1107
- “I2CFIFO” on page 1109
- “IB” on page 1113
- “IDDR71” on page 1114
- “IDDR71A” on page 1116
- “IDDRX1” on page 1117
- “IDDRX1A” on page 1118
- “IDDRX2” on page 1119
- “IDDRX2A” on page 1121
- “IDDRX2DQ” on page 1122
- “IDDRX4” on page 1123
- “IDDRX4A” on page 1125
- “IDDRX4DQ” on page 1126
- “IDDRX5” on page 1128
- “IDDRX5A” on page 1129

- “IFD1P3AZ” on page 1131
- “IFD1P3BX” on page 1132
- “IFD1P3DX” on page 1133
- “IFD1P3IX” on page 1135
- “IFD1P3JX” on page 1136
- “INV” on page 1138
- “IMONDELAY” on page 1139
- “IOL_B” on page 1140
- “JTAG” on page 1142
- “JTAGA” on page 1144
- “JTAGH25” on page 1146
- “JTAGH25SOFT” on page 1149
- “LSOSC” on page 1151
- “LSOSC1P8V” on page 1152
- “LSOSC_CORE” on page 1153
- “LUT4” on page 1154
- “MAC16” on page 1156
- “MIPI” on page 1162
- “MIPIA” on page 1163
- “MULT18X18” on page 1164
- “MULT18X18A” on page 1166
- “MULT18X36” on page 1168
- “MULT36X36” on page 1170
- “MULT8X8” on page 1172
- “MULT9X9” on page 1174
- “MULT9X9A” on page 1175
- “MULTADDSUB18X18” on page 1177
- “MULTADDSUB18X18A” on page 1180
- “MULTADDSUB18X18WIDE” on page 1185
- “MULTADDSUB18X36” on page 1189
- “MULTADDSUB36X36” on page 1192
- “MULTADDSUB9X9WIDE” on page 1195
- “MULTIBOOT” on page 1200
- “MULTPREADD18X18” on page 1200
- “MULTPREADD9X9” on page 1203
- “OB” on page 1205

- “OB_RGB” on page 1206
- “OBZ” on page 1207
- “OBZ_B” on page 1208
- “ODDR71” on page 1209
- “ODDR71A” on page 1210
- “ODDRX1” on page 1212
- “ODDRX1A” on page 1213
- “ODDRX2” on page 1214
- “ODDRX2A” on page 1216
- “ODDRX2DQ” on page 1217
- “ODDRX2DQS” on page 1219
- “ODDRX4” on page 1220
- “ODDRX4A” on page 1221
- “ODDRX4DQ” on page 1223
- “ODDRX4DQS” on page 1225
- “ODDRX5” on page 1226
- “ODDRX5A” on page 1228
- “OFD1P3AZ” on page 1230
- “OFD1P3BX” on page 1231
- “OFD1P3DX” on page 1232
- “OFD1P3IX” on page 1233
- “OFD1P3JX” on page 1235
- “OSC” on page 1236
- “OSCA” on page 1239
- “OSCE” on page 1241
- “OSCD” on page 1242
- “OSHX2” on page 1243
- “OSHX4” on page 1244
- “OUTDELAYA” on page 1246
- “OUTDELAYB” on page 1247
- “OUTDELAYC” on page 1248
- “PCIE” on page 1250
- “PCIELL” on page 1289
- “PCLKDIV” on page 1294
- “PCLKDIVSP” on page 1295
- “PCSREFMUX” on page 1296

- “PCSX1” on page 1297
- “PCSX2” on page 1307
- “PCSX4” on page 1324
- “PDP4K” on page 1352
- “PDP16K” on page 1354
- “PDP32K” on page 1357
- “PDPSC16K” on page 1361
- “PDPSC32K” on page 1364
- “PDPSC512K” on page 1367
- “PLL” on page 1369
- “PLLA” on page 1382
- “PLL_B” on page 1395
- “PLLC” on page 1400
- “PLLREFCS” on page 1406
- “PLLREFCSA” on page 1407
- “RGB” on page 1408
- “RGB1P8V” on page 1410
- “RGB_CORE” on page 1412
- “RGBPWM” on page 1414
- “SEDC” on page 1415
- “SEDCA” on page 1416
- “SEIO18” on page 1417
- “SEIO18A” on page 1419
- “SEIO33” on page 1421
- “SGMIICDR” on page 1422
- “SGSR” on page 1427
- “SP16K” on page 1427
- “SP16KA” on page 1430
- “SP32K” on page 1432
- “SP256K” on page 1435
- “SP512K” on page 1436
- “SPI_B” on page 1438
- “SPR16X4” on page 1441
- “SPR16X4A” on page 1442
- “SPR32X2” on page 1442
- “TSALLA” on page 1443

- “TSALLB” on page 1444
- “TSHX2DQ” on page 1445
- “TSHX2DQS” on page 1446
- “TSHX4DQ” on page 1447
- “TSHX4DQS” on page 1449
- “UMXSPI” on page 1450
- “USXSPI” on page 1454
- “USB23” on page 1457
- “VFB_B” on page 1461
- “VHI” on page 1462
- “VLO” on page 1463
- “WARMBOOT” on page 1463
- “WDT” on page 1464
- “WDTA” on page 1465
- “WIDEFN9” on page 1466

ACC54

54-bit Accumulator component for DSP

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

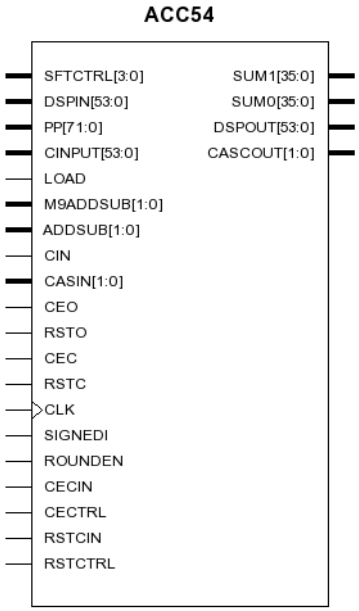


Table 1: Input Ports

Name	Range	Description
SFTCTRL	3:0	Arithmetical right shift 0 - 15
DSPIN	53:0	Previous ACC54 data in
PP	71:0	Input form 4 groups of REG18
CINPUT	53:0	Input form 4 groups of REG18
LOAD		Dynamic load control for C register
M9ADDSUB	1:0	Dynamic add/sub control for state 1 adder. 1=subtraction
ADDSUB	1:0	Dynamic add/sub control for 54-bit adder
CIN		Carry in from CIB
CASIN	1:0	Carry in from previous ACC54
CEO		Clock enable for output register
RSTO		Synchronous reset for output register
CEC		Clock enable for C register
RSTC		Synchronous reset for C register
CLK		Clock for output register
SIGNEDI		Signed input
ROUNDEN		1=ACCU54 round enable

Table 1: Input Ports (Continued)

Name	Range	Description
CECIN		Input carryin enable
CECTRL		Input control (addsub, m9addsub, signed, and loadc) enable
RSTCIN		Input carryin reset
RSTCTRL		Input control (addsub, m9addsub, signed, and loadc) reset

Table 2: Output Ports

Name	Range	Description
SUM1	35:0	Results Output to CIB
SUM0	35:0	Results Output to CIB
DSPOUT	53:0	Output to next ACC54
CASCOUT	1:0	Carry output to next ACC54

Table 3: Parameters

Name	Values	Description
SIGN	"DISABLED" *(default)* "ENABLED"	Static sign operation enable for ACCU54: Enable
M9ADDSUB_CTRL	"ADDITION" *(default)* "ADDSUB" "SUBADD" "SUBTRACTION"	Static add/sub control for stage 1 adder 1: Subtraction
ADDSUB_CTRL	"ADD_ADD_CTRL_54_BIT_ADDER" *(default)* "SUB_ADD_CTRL_54_BIT_ADDER" "ADD_SUB_CTRL_54_BIT_ADDER" "SUB_SUB_CTRL_54_BIT_ADDER"	Static add/sub control for 54-bit adder
STATICOPCODE_EN	"DISABLED" *(default)* "ENABLED"	1: Static opcode enable
OUTREGBYPS	"BYPASS" *(default)* "REGISTER"	1: Output register bypass
GSR	"DISABLED" *(default)* "ENABLED"	1: GSRN enable
PROGCONST	"0'b00000000000000000000000000000000000000000000000000000000" (default)	Constant value for C
CONSTSEL	"BYPASS" *(default)* "SELECT"	1: Select constant value for C
DSPCASCADE	"DISABLED" *(default)* "ENABLED"	1: DSP cascaded enable

Table 3: Parameters (Continued)

Name	Values	Description
ACC108CASCADE	"BYPASSCASCADE" *(default)* "CASCADE2ACCU54TOFORMACCU108"	1: cascade two ACCU54 to form ACCU108
ACCUMODE	"MODE0" *(default)* "MODE1" "MODE2" "MODE3" "MODE4" "MODE5" "MODE6" "MODE7"	Accumulator mode
ACCUBYPS	"USED" *(default)* "BYPASS"	1: Bypass ACC54
CREGBYPS1	"REGISTER" *(default)* "BYPASS"	1: Bypass C register 1
CREGBYPS2	"REGISTER" *(default)* "BYPASS"	1: Bypass C register 2
CREGBYPS3	"REGISTER" *(default)* "BYPASS"	1: Bypass C register 3
CINREGBYPS1	"REGISTER" *(default)* "BYPASS"	1: Bypass Cin register 1
CINREGBYPS2	"REGISTER" *(default)* "BYPASS"	1: Bypass Cin register 2
CINREGBYPS3	"REGISTER" *(default)* "BYPASS"	1: Bypass Cin register 3
LOADREGBYPS1	"REGISTER" *(default)* "BYPASS"	1: Bypass load register 1
LOADREGBYPS2	"REGISTER" *(default)* "BYPASS"	1: Bypass load register 2
LOADREGBYPS3	"REGISTER" *(default)* "BYPASS"	1: Bypass load register 3
M9ADDSUBREGBYPS1	"REGISTER" *(default)* "BYPASS"	1: Bypass M9AddSub register 1
M9ADDSUBREGBYPS2	"REGISTER" *(default)* "BYPASS"	1: Bypass M9AddSub register 2
M9ADDSUBREGBYPS3	"REGISTER" *(default)* "BYPASS"	1: Bypass M9AddSub register 3
ADDSUBSIGNREGBYPS1	"REGISTER" *(default)* "BYPASS"	1: Bypass AddSubSign register 1
ADDSUBSIGNREGBYPS2	"REGISTER" *(default)* "BYPASS"	1: Bypass AddSubSign register 2
ADDSUBSIGNREGBYPS3	"REGISTER" *(default)* "BYPASS"	1: Bypass AddSubSign register 3

Table 3: Parameters (Continued)

Name	Values	Description
ROUNDHALFUP	"DISABLED" *(default)* "ENABLED"	1: ACCU54 round half up mode enable
ROUNDRTZI	"ROUND_TO_ZERO" *(default)* "ROUND_TO_INFINITE"	0: ACCU54 round to zero 1: ACCU54 round to infinite
ROUNDBIT	"ROUND_TO_BIT0" *(default)* "ROUND_TO_BIT1" "ROUND_TO_BIT2" "ROUND_TO_BIT3" "ROUND_TO_BIT4" "ROUND_TO_BIT5" "ROUND_TO_BIT6" "ROUND_TO_BIT7" "ROUND_TO_BIT8" "ROUND_TO_BIT9" "ROUND_TO_BIT10" "ROUND_TO_BIT11" "ROUND_TO_BIT12" "ROUND_TO_BIT13" "ROUND_TO_BIT14" "ROUND_TO_BIT15"	0000: round to bit0 ... 1111: round too bit15
CASCOUTREGBYPS	"REGISTER" *(default)* "BYPASS"	1: DSP cascaded enable
SFTEN	"DISABLED" *(default)* "ENABLED"	Shift enable
RESET	"SYNC" (default)	

ADC

Wrapper for Analog-to-Digital Converter

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP

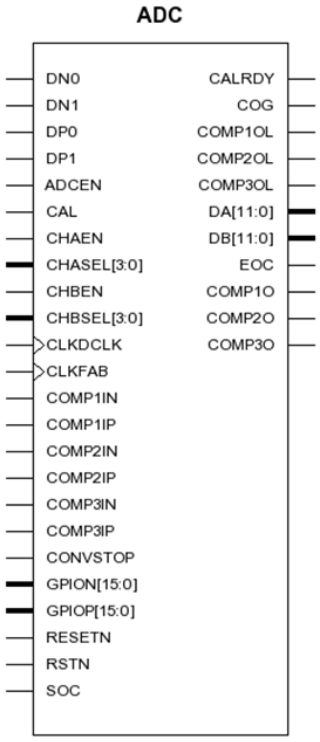


Table 4: Input Ports

Name	Description
DN0	
DN1	
DP0	
DP1	
ADCEN	
CAL	
CHAEN	
CHASEL[3:0]	
CHBEN	
CHBSEL[3:0]	
CLKDCLK	
CLKFAB	
COMP1IN	

Table 4: Input Ports (Continued)

Name	Description
COMP1IP	
COMP2IN	
COMP2IP	
COMP3IN	
COMP3IP	
CONVSTOP	
GPION[15:0]	
GPIOP[15:0]	
RESETN	
RSTN	
SOC	

Table 5: Output Ports

Name	Description
CALRDY	
COG	
COMP1OL	
COMP2OL	
COMP3OL	
DA[11:0]	
DB[11:0]	
EOC	
COMP1O	
COMP2O	
COMP3O	

Table 6: Parameters

Name	Values	Description
ADC_ENP	"ENABLED" *(default)* "DISABLED"	ADC enabling signal by MIB
CLK_DIV	*"2" (default)*	DCLK divider control
CTLCOMPSW1	"DISABLED" *(default)* "ENABLED"	Internal comparator switch control

Table 6: Parameters (Continued)

Name	Values	Description
CTLCOMPSW2	"DISABLED" *(default)* "ENABLED"	Internal comparator switch control
CTLCOMPSW3	"DISABLED" *(default)* "ENABLED"	Internal comparator switch control
DF	"STRAIGHT_BINARY" *(default)* "TWOS_COMPLEMENT"	Output data format control between straight binary and 2's complement coding
EN_COMP1	"ENABLED" *(default)* "DISABLED"	Auxiliary Comparator 1 enable
EN_COMP2	"ENABLED" *(default)* "DISABLED"	Auxiliary Comparator 2 enable
EN_COMP3	"ENABLED" *(default)* "DISABLED"	Auxiliary Comparator 3 enable
OMA	"BIPOLAR" *(default)* "UNIPOLAR"	Channel A Analog input bi-polar/uni-polar configuration
OMB	"BIPOLAR" *(default)* "UNIPOLAR"	Channel B Analog input bi-polar/uni-polar configuration
REFBUFAEN	"DISABLED" *(default)* "ENABLED"	Channel A internal reference buffer enable
REFBUFBEN	"DISABLED" *(default)* "ENABLED"	Channel B internal reference buffer enable
SLEEP	"DISABLED" *(default)* "ENABLED"	Sleep mode enable
VREFACFG	"1P0_TO_1P2" *(default)* "1P2_TO_1P4" "1P4_TO_1P6" "1P6_TO_1P8"	Channel A external reference voltage level selection
VREFASEL	"EXTERNAL" *(default)* "INTERNAL"	Channel A reference voltage input configuration between internal and external reference voltage
VREFBCFG	"1P0_TO_1P2" *(default)* "1P2_TO_1P4" "1P4_TO_1P6" "1P6_TO_1P8"	Channel B external reference voltage level selection
VREFBSEL	"EXTERNAL" *(default)* "INTERNAL"	Channel B reference voltage input configuration between internal and external reference voltage

LIFCL 17K devices have ADC usage restriction. GPIOP[4, 10, 11, 12, 15] and GPION[4, 10, 11, 12, 15] are not available.

The ADC macro is composed of two blocks: the external IP and its wrapper.

The external IP consists of two ADCs and three continuous-time comparators. Two ADCs convert unipolar or bipolar input signal into 12-bit resolution data with maximum 1MSPS (Mega Samples Per Second) conversion speed

sequentially or simultaneously. This XML covers the wrapper for ADC IP block.

ALUREG

Wrapper for Arithmetic Logic Unit and Register File for RISC-V Soft Processor

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5TD
- LFMXO5T
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP

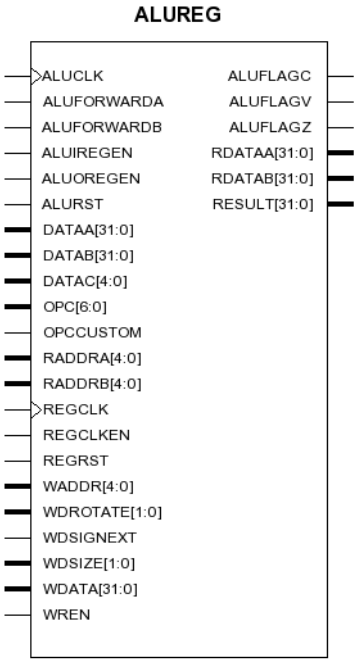


Table 7: Input Ports

Name	Description
ALUCLK	
ALUFORWARDA	

Table 7: Input Ports (Continued)

Name	Description
ALUFORWARDB	
ALUIREGEN	
ALUOREGEN	
ALURST	
DATAA[31:0]	
DATAB[31:0]	
DATAC[4:0]	
OPC[6:0]	
OPCCUSTOM	
RADDRA[4:0]	
RADDRB[4:0]	
REGCLK	
REGCLKEN	
REGRST	
WADDR[4:0]	
WDROTATE[1:0]	
WDSIGNEXT	
WDSIZE[1:0]	
WDATA[31:0]	
WREN	

Table 8: Output Ports

Name	Description
ALUFLAGC	
ALUFLAGV	
ALUFLAGZ	
RDATAA[31:0]	
RDATAB[31:0]	
RESULT[31:0]	

Table 9: Parameters

Name	Values	Description
ALURST_ACTIVELOW	"DISABLE" (default) "ENABLE	
GSR	"ENABLED" (default) "DISABLED"	
INREG	"DISABLE" (default) "ENABLE	
MULFXP_ROUND	"ENABLED" (default) "DISABLED"	
OUTREG	"DISABLE" (default) "ENABLE	
REGRST_ACTIVELOW	"DISABLE" (default) "ENABLE	
RETAIN	"ENABLED" (default) "DISABLED"	
RFASYNC_RD	"SYNC_RD" (default) "ASYNC_RD"	
RFR0_RO	"R0READONLY" *(default)*	
RFUNALIA_WR	"DISABLE" (default) "ENABLE	
RFWCLK_INV	"SIG" (default) "INV"	

AON

Always On

Architectures supported:

- LIFCL-33U

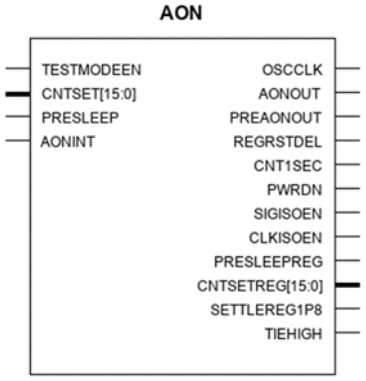


Table 10: Input Ports

Name	Range	Description
TESTMODEEN		Enable test mode
CNTSET	15:0	AON timer counting edges
PRESLEEP		FPGA ready to sleep
AONINT		External interrupt of AON

Table 11: Output Ports

Name	Range	Description
OSCCLK		AON built in oscillator output
AONOUT		AON output to control FPGA power
PREAONOUT		FPGA power down in test mode
REGRSTDEL		Power down internal state signal for test mode
CNT1SEC		Power down internal state signal for test mode
PWRDN		Power down internal state signal for test mode
SIGISOEN		Power down internal state signal for test mode
CLKISOEN		Power down internal state signal for test mode
PRESLEEPREG		Power down internal state signal for test mode

Table 11: Output Ports

Name	Range	Description
CNTSETREG	15:0	Power down internal state signal for test mode
SETTLEREG1P8		Power down internal state signal for test mode
TIEHIGH		Power down internal state signal for test mode

BB

Bidirectional Buffer with Tri-state

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

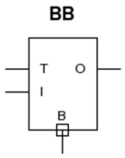


Table 12: Input Ports

Name	Description
I	Data to pad
T	Tri-state control

Table 13: Output Ports

Name	Description
O	Data from pad

Table 14: Bidirectional Ports

Name	Description
B	Connection to pad

Table 15: Truth Table

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

BB_ADC

Single-Ended I/O Block for 1.8 V

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP

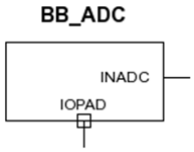


Table 16: Output Ports

Name	Description
INADC	Output to ADC

Table 17: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

BB_B

Bidirectional I/O Buffer with Tri-state

Architectures supported:

- iCE40 UltraPlus

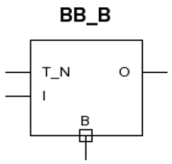


Table 18: Input Ports

Name	Description
T_N	Tri-state control, active low meaning T_N = 0 ' O = Z
I	Data to pad

Table 19: Output Ports

Name	Description
O	Data from pad

Table 20: Bidirectional Ports

Name	Description
B	Connection to pad

Table 21: Truth Table

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T_N	O	B
X	0	U	Z
X	0	1	1
X	0	0	0
0	1	0	0
1	1	1	1

X = Don’t care

U = Unknown

BB_CDR

Single-Ended I/O Block for 1.8V

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

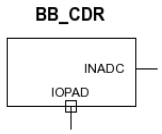


Table 22: Output Ports

Name	Description
INADC	Output to ADC

Table 23: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

BB_I3C

I/O with User Controllable Pull-Up Resistors

Architectures supported:

- iCE40 UltraPlus

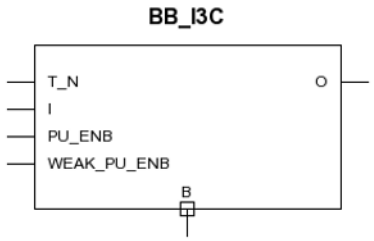


Table 24: Input Ports

Name	Description
PU_ENB	Pull-up enable
WEAK_PU_ENB	Weak pull-up enable
T_N	Tri-state control active low
I	Data to pad

Table 25: Output Ports

Name	Description
O	Data from pad

Table 26: Bidirectional Ports

Name	Description
B	Pad

Table 27: Parameters

Name	Values
PULLMODE	"100K" *(default)* "3P3K" "6P8K" "10K" “NA”
IO_TYPE	"LVCMOS33" *(default)* "LVCMOS25" "LVCMOS18" "LVCMOS12" "LVDSE"

Table 27: Parameters (Continued)

Name	Values
BANK_VCCIO	"3.3" *(default)*
	"2.5"
	"1.8"
	"1.2"
DRIVE	"6" *(default)*
	"NA"
	"2"
	"4"
	"6"
	"10"
	"12"
	"16"

LVDSE for BIDI may not be supported, but for stability-sake, better to leave it as an option to make it 1:1 with PIO physical cell model. LCT table will correctly stop LDVSE from going through for I3C.

BB_I3C_A

Single-Ended I/O Block for 3.3V

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

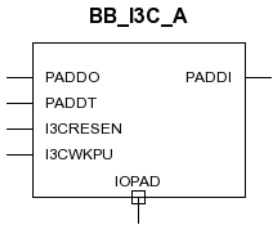


Table 28: Input Ports

Name	Description
PADDO	Data output from fabric to IO pad
PADDT	Data tri-state control; high = tri-state
I3CRESEN	I3C resistor enable
I3CWKPU	I3C weak pullup enable

Table 29: Output Ports

Name	Description
PADDI	Data input from IO pad to fabric

Table 30: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

BB_I3C_B

Bidirectional I3C Buffer Primitive

Architectures supported:

- LAV-AT-ES1
- LN2-CT

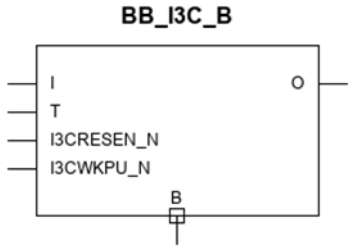


Table 31: Input Ports

Name	Range	Type	Active State	Description
I	1	Data		Input of I3C Bidirectional Buffer
T	1	Control		Tristate control of I3C Bidirectional Buffer
I3CRESEN_N	1	Control	Active Low	Resistor Enable control
I3CWKPU_N	1	Control	Active Low	Weak PULLUP control

Table 32: Output Ports

Name	Range	Type	Active State	Description
O	1			Output of I3C Bidirectional Buffer

Table 33: Bidirectional Ports

Name	Range	Type	Active State	Description
B				IOPAD

BB_OD

Input/Output Buffer

Architectures supported:

- iCE40 UltraPlus

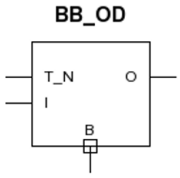


Table 34: Input Ports

Name	Description
T_N	Tri-state control, active low
I	Data to pad

Table 35: Output Ports

Name	Description
O	Data from pad

Table 36: Bidirectional Ports

Name	Description
B	Connection to pad

Table 37: Truth Table

Inputs		Outputs	Bidirectional
I	T_N	O	B
X	0	U	Z
X	0	1	1
X	0	0	0
0	1	0	0
1	1	Z	Z

X = Don’t care

U = Unknown

BB_PROGRAMN

Single-Ended I/O Block for PROGRAMN

Architectures supported:

- LIFCL-33

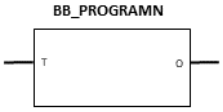


Table 38: Input Ports

Name	Description
T	Tri-state control

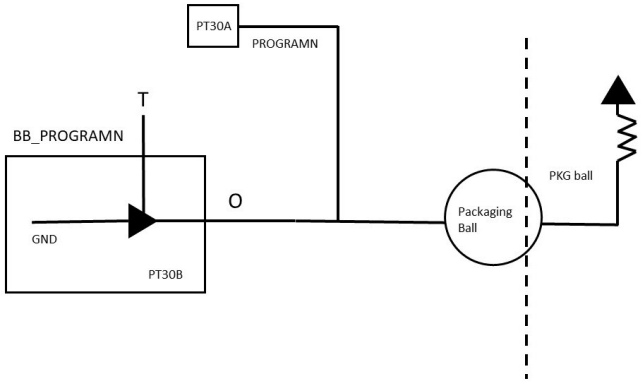
Table 39: Output Ports

Name	Description
O	Data from pad

Table 40: Truth Table

Output	Description
T = 0	Reset the PROGRAMN FPGA.
T = 1	Release reset that allows the FPGA to boot.

Figure 1: BB_PROGRAMN Diagram



BFD1P3KX

Positive Edge Triggered Bidirectional D Flip-Flop with Positive Level Enable and Synchronous Set/Reset for Input/Output/Tri-state Signals (to/from IO)

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

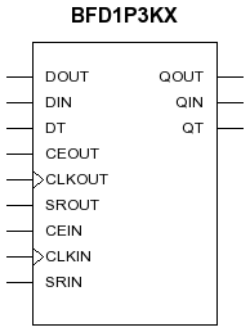


Table 41: Input Ports

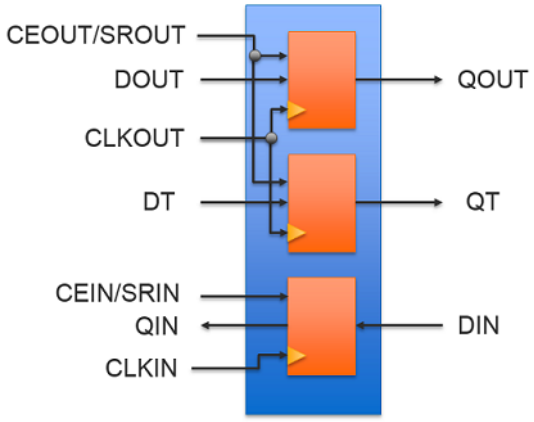
Name	Description
DOUT	Data out from fabric
DIN	Data in from I/O
DT	Tri-state in from fabric
CEOUT	Output/Tri-state clock enable, active high
CLKOUT	Output/Tri-state clock
SROUT	Output/Tri-state set/reset
CEIN	Input clock enable, active high
CLKIN	Input clock
SRIN	Input set/reset

Table 42: Output Ports

Name	Description
QOUT	Data out to IO
QIN	Data in to fabric
QT	Tri-state out to IO

Table 43: Parameters

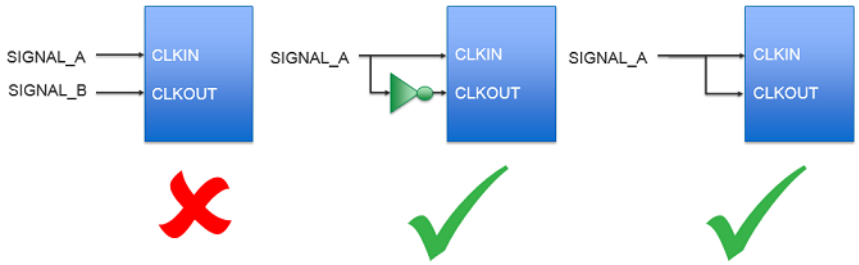
Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
OUTSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the output register
INSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the input register
TSSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the tri-state register



When INSET/OUTSET/TSSET = “SET”, truth table is the same as *FD1P3JX.

When INSET/OUTSET/TSSET = “RESET”, truth table is the same as *FD1P3IX.

CEOUT/CEIN, SROUT/SRIN, CLKOUT/CLKIN must be driven by the same signal, with the exception that they may be driven by the logical inversion of one another.



BFD1P3LX

Positive Edge Triggered Bidirectional D Flip-Flop with Positive Level Enable and Asynchronous Set/Reset, for Input/Output/Tri-state Signals (to/from IO)

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

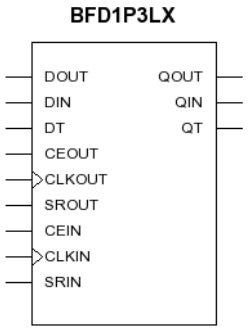


Table 44: Input Ports

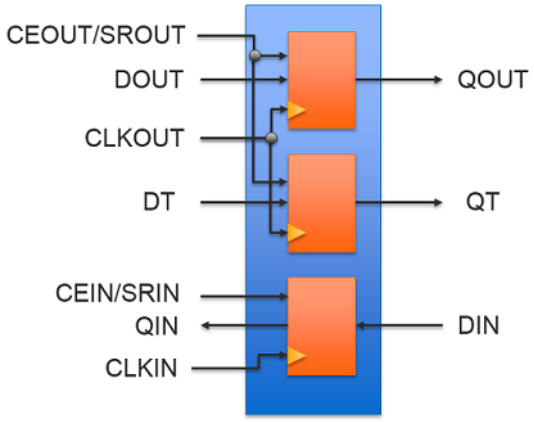
Name	Description
DOUT	Data out from fabric
DIN	Data in from I/O
DT	Tri-state in from fabric
CEOUT	Output/Tri-state clock enable, active high
CLKOUT	Output/Tri-state clock
SROUT	Output/Tri-state set/reset
CEIN	Input clock enable, active high
CLKIN	Input clock
SRIN	Input set/reset

Table 45: Output Ports

Name	Description
QOUT	Data out to IO
QIN	Data in to fabric
QT	Tri-state out to IO

Table 46: Parameters

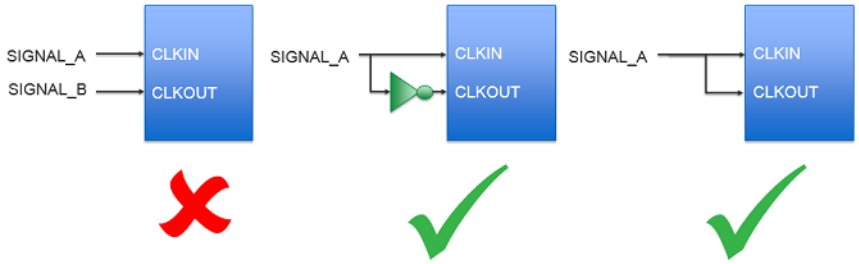
Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
OUTSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the output register
INSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the input register
TSSET	"RESET" *(default)* "SET"	Controls the set/reset behavior of the tri-state register



When INSET/OUTSET/TSSET = “SET”, truth table is the same as *FD1P3BX.

When INSET/OUTSET/TSSET = “RESET”, truth table is the same as *FD1P3DX.

CEOUT/CEIN, SROUT/SRIN, CLKOUT/CLKIN must be driven by the same signal, with the exception that they may be driven by the logical inversion of one another.



BNKREF18

Bankref Block for IOS18

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

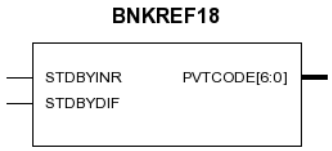


Table 47: Input Ports

Name	Description
STDBYINR	
STDBYDIF	

Table 48: Output Ports

Name	Description
PVTCODE	

Table 49: Parameters

Name	Values	Description
BANK	"0'b0000" (default)	Virtual attribute, used to tell placer where to place it.
STANDBY_DIFFIO	"DISABLED" (default) "ENABLED"	Enables SLVS/LVDS output standby mode
STANDBY_INR	"DISABLED" (default) "ENABLED"	Enables INR standby mode: 0 off; 1 on

BUF

Non-Inverting Buffer

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C

- UT24CP

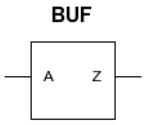


Table 50: Input Ports

Name	Description
A	BUF function input A

Table 51: Output Ports

Name	Description
Z	BUF output Z

CCU2

Carry Chain

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

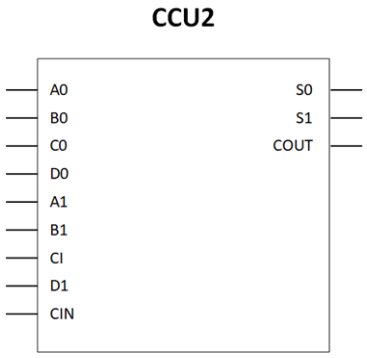


Table 52: Input Ports

Name	Description
A0	Input bit 0 for non-operand signal
B0	Input bit 0 of the first operand
C0	Input bit 0 of the second operand
D0	Input bit 0 of the third operand
A1	Input bit 1 for non-operand signal
B1	Input bit 1 of the first operand
C1	Input bit 1 of the second operand
D1	Input bit 1 of the third operand
CIN	Carry in

Table 53: Output Ports

Name	Description
S0	Output bit 0 of the sum
S1	Output bit 1 of the sum
COUT	Carry out

Table 54: Parameters

Name	Values	Description
INIT0	"0x0000" (default)	
INIT1	"0x1111" (default)	
INJECT	"YES" (default)	

CCU2_B

Carry Chain

Architectures supported:

- iCE40 UltraPlus

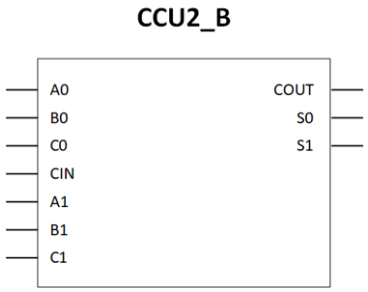


Table 55: Input Ports

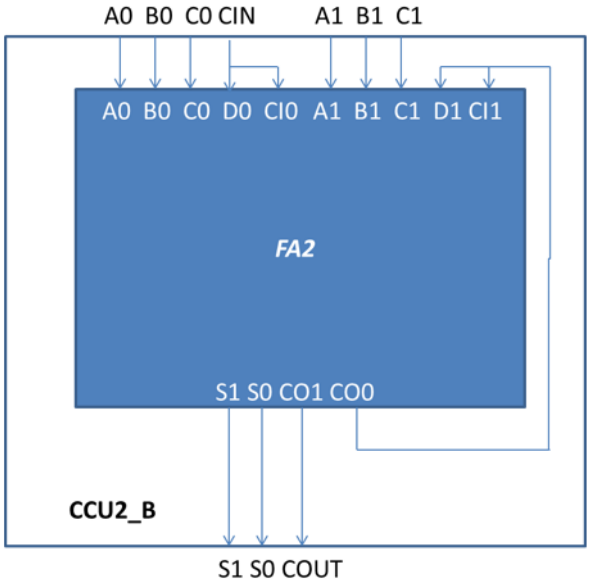
Name	Description
A0	Input bit 0 for non-operand signal
B0	Input bit 0 of the first operand
C0	Input bit 0 of the second operand
CIN	Carry in
A1	Input bit 1 for non-operand signal
B1	Input bit 1 of the first operand
C1	Input bit 1 of the second operand

Table 56: Output Ports

Name	Description
COUT	Carry out
S0	Output bit 0 of the sum
S1	Output bit 1 of the sum

Table 57: Parameters

Name	Values	Description
INIT0	Hexadecimal value "0xc33c" *(default)*	
INIT1	Hexadecimal value "0xc33c" *(default)*	



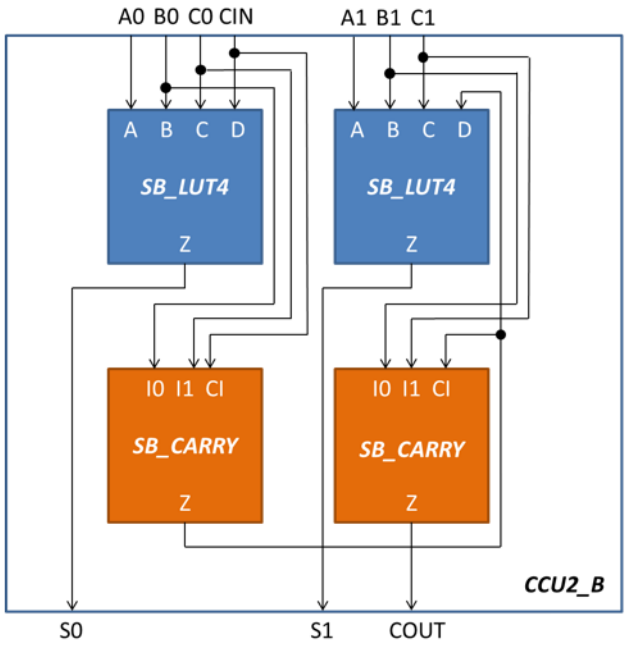


Table 58: Truth Table

Inputs					Outputs		
B1	**B0**	**C1**	**C0**	**CIN**	**S1**	**S0**	**COUT**
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	1	1	1	0
0	0	1	1	0	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	1	0	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	1	0	0	1
0	1	1	1	0	0	0	1
0	1	1	1	1	0	1	1

Table 58: Truth Table (Continued)

Inputs					Outputs		
B1	**B0**	**C1**	**C0**	**CIN**	**S1**	**S0**	**COUT**
1	0	0	0	0	1	0	0
1	0	0	0	1	1	1	0
1	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	0	1
1	0	1	0	1	0	1	1
1	0	1	1	0	0	1	1
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	1	0	0	0	1
1	1	0	1	1	0	1	1
1	1	1	0	0	0	1	1
1	1	1	0	1	1	0	1
1	1	1	1	0	1	0	1
1	1	1	1	1	1	1	1

Note:

This is the truth table for an example full adder using CCU2_B. B1, B0 and C1, C0 are used as operands, while A1, A0 are not used and are tied to ground.

Rules & Restrictions

COUT can only go to D input of LUT above it or next CIN.

To drive VHI to the carry-in “half” a CCU2 must be used. This is done by driving B0 or C0 or both (if S0 can be a don’t care value) to VHI and leaving CIN floating. Your real CCU2 will then start with B1, C1 and the carry-in will be VHI.

To drive VLO to the carry-in “half” a CCU2 must be used. This is done by driving VLO for B0 and C0 and leaving CIN floating. Your real CCU2 starts with B1, C1 and CIN1 will be driven by VLO.

If you want to bring out last COUT to fabric, simply connect COUT to whatever is your fabric logic. MAP and Placer will work to insert a LUT and

place it in the appropriate location automatically (thus ensuring rule 1 is followed).

See examples below that takes into account the above rules.

Note:

The examples below do not make use of the A0 or A1 ports as they are not needed for carry-chain functionality.

Burning half a CCU2 for carry-chain:

```
CCU2_B counter_12_add_4_0 (
    .B0(GND_net),
    .C0(GND_net),
    .CIN(),
    .B1(firstB),
    .C1(firstC),
    .COUT(firstCarry),
    .S0(),
    .S1(firstSum)
);
```

Using LUT to bring last Carry Out:

```
CCU2_B counter_12_add_4_7 (
    .B0(GND_net),
    .C0(out_c_5),
    .CIN(COUT_prev),
    .B1(GND_net),
    .C1(out_c_6),
    .COUT(LAST_CARRY),
    .S0(n40),
    .S1(n39)
);

LUT  lutInst(
    .D(LAST_CARRY),
    .F(CARRYOUT_TO_FABRIC)
); //eqn : F=D
```

Using CCU2 to bring last Carry Out:

Why it works: Last Sum = GND XOR GND XOR CIN_PREV = lastCarryOut.

```
CCU2_B counter_12_add_4_9 (
    .B0(GND_net),
    .C0(out_c_7),
    .CIN(COUT_prev),
    .B1(GND net),
    .C1(GND_net),
    .COUT(),
    .S0(n40),
    .S1(lastCarryOut)
);
```

CCU2C

Carry Chain

Architectures supported:

- LAV-AT-ES1
- LN2-CT

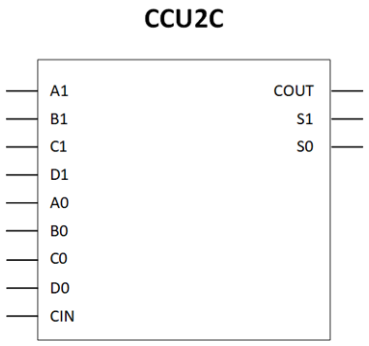


Table 59: Input Ports

Name	Type	Description
A1	Data	Function input A1
B1	Data	Function input B1
C1	Control	Function input C1
D1	Control	Function input D1
A0	Data	Function input A0
B0	Data	Function input B0
C0	Control	Function input C0
D0	Control	Function input D0
CIN	Control	Carry in

Table 60: Output Ports

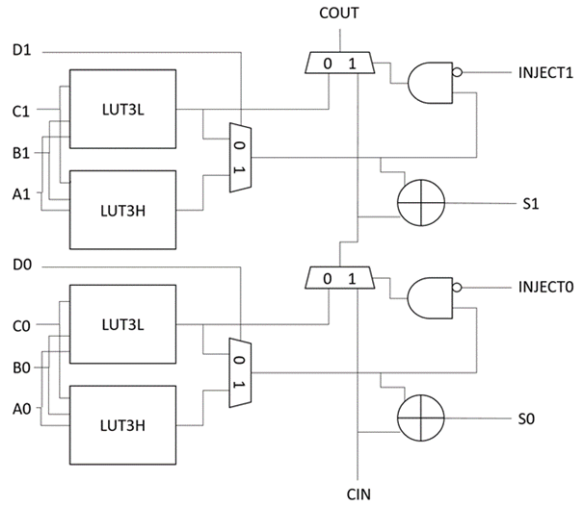
Name	Description
COUT	Carry out
S1	Sum 1 output
S0	Sum 0 output

Table 61: Parameters

Name	Values	Description
INJECT0	"YES" (default) "NO"	INJECT0
INJECT1	"YES" (default) "NO"	INJECT1
INIT0	"0x0000" (default)	CCU2 initialization data 0
INIT1	"0x1111" (default)	CCU2 initialization data 1

- At the beginning of the carry chain, CIN must always be tied to a constant 0.
- At the beginning of the carry chain, if a CIN is desired (either a constant 1 or a live signal), use one of the A/B/C/D pins and adjust the INIT equation accordingly.
- At the beginning of the carry chain, CIN is a "don’t care", but it MUST be connected to low for synthesis netlist purpose. All the inputs must have known values.
- In Avant, it is not required to consume a half-slice at the beginning of the carry chain. If the CIN is desired (either constant or a live signal), use the C0 pin and adjust the INIT0 equation as provided in the table for first bit.
- In general, COUT must only drive CIN except at the end of a shift chain. At the end of a chain, COUT may connect directly to other design logic. If the chain ends halfway through a CCU2, the carry out value may be brought out through S1 by setting INIT1 to 0x000.

Logical Diagram of CCU2C



CONFIG_DONE

CONFIG final done status

Architectures supported:

- LAV-AT-ES1
- LN2-CT

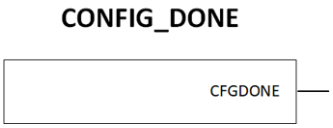


Table 62: Output Ports

Name	Description
CFGDONE	CONFIG final done status

CONFIG_LMMI

LMMI (Lattice Memory Mapped Interface) interface to the configuration block

Architectures supported:

- LFD2NX

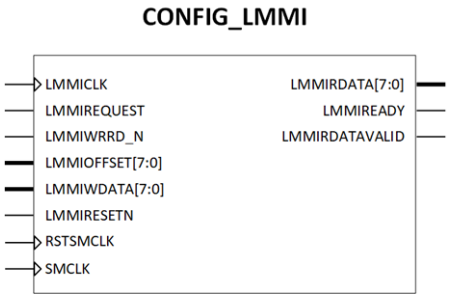


Table 63: Input Ports

Name	Description
LMMICLK	Clock for LMMI interface
LMMIREQUEST	Start transaction
LMMIWRRD_N	1=write, 0=read
LMMIOFFSET[7:0]	Register offset
LMMIWDATA[7:0]	Write data
LMMIRESETN	Reset the LMMI interface
RSTSMCLK	Reset SMCLK
SMCLK	Configuration block clock

Table 64: Output Ports

Name	Description
LMMIRDATA[7:0]	Read data
LMMIREADY	LMMI ready signal
LMMIRDATAVALID	Read transaction is complete and LMMIRDATA contains valid data

Table 65: Parameters

Name	Values	Description	Inferable
LMMI_EN	"DIS" (default) "EN"	Enable the LMMI interface	No

CONFIG_LMMIA

LMMI (Lattice Memory Mapped Interface) interface to the configuration block

Architectures supported:

- LFCPNX
- LIFCL-33, LIFCL-33U
- UT24CP

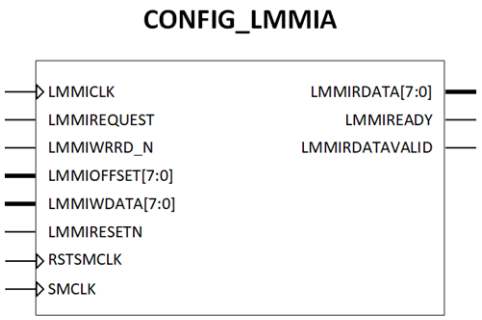


Table 66: Input Ports

Name	Description
LMMICLK	Clock for LMMI interface
LMMIREQUEST	Start transaction
LMMIWRRD_N	1=write, 0=read
LMMIOFFSET[7:0]	Register offset
LMMIWDATA[7:0]	Write data
LMMIRESETN	Reset the LMMI interface

Table 66: Input Ports

Name	Description
RSTSMCLK	Reset SMCLK
SMCLK	Configuration block clock

Table 67: Output Ports

Name	Description
LMMIRDATA[7:0]	Read data
LMMIREADY	LMMI ready signal
LMMIRDATAVALID	Read transaction is complete and LMMIRDATA contains valid data

Table 68: Parameters

Name	Values	Description	Inferable
LMMI_EN	"DIS" (default) "EN"	Enable the LMMI interface	No

CONFIG_LMMIB

Includes Flash access through LMMI

Architectures supported:

- LFMXO5-25

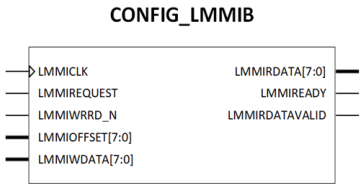


Table 69: Input Ports

Name	Description
LMMICLK	Clock for LMMI interface
LMMIREQUEST	Start transaction

Table 69: Input Ports (Continued)

Name	Description
LMMIWRRD_N	1=write, 0=read
LMMIOFFSET	Register offset
LMMIWDATA	Write data

Table 70: Output Ports

Name	Description
LMMIRDATA	Read data
LMMIREADY	LMMI ready signal
LMMIRDATAVALID	Read transaction is complete and LMMIRDATA contains valid data

Table 71: Parameters

Name	Candidates	Default	Description
LMMI_EN	DIS, EN	DIS	Enable the LMMI interface

CONFIG_LMMIC

LMMI Slave Primitive

Architectures supported:

- LAV-AT-ES1
- LN2-CT-10

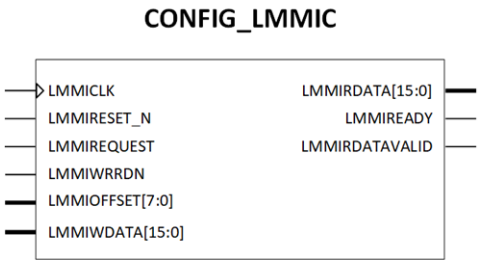


Table 72: Input Ports

Name	Range	Description
LMMICLK	1	LMMI Clock
LMMIRESET_N	1	LMMI Resetn
LMMIREQUEST	1	LMMI Request
LMMIWRRDN	1	LMMI Write Enable
LMMIOFFSET	7:0	LMMI Offset
LMMIWDATA	15:0	LMMI Write Data

Table 73: Output Ports

Name	Range	Description
LMMIRDATA	15:0	LMMI Read Data
LMMIRDATAVALID	1	LMMI Read Data Valid
LMMIREADY	1	LMMI Slave Ready

CONFIG_LMMIE

LMMI (Lattice Memory Mapped Interface) interface to the configuration block

Architectures supported:

- LFMXO5-100T

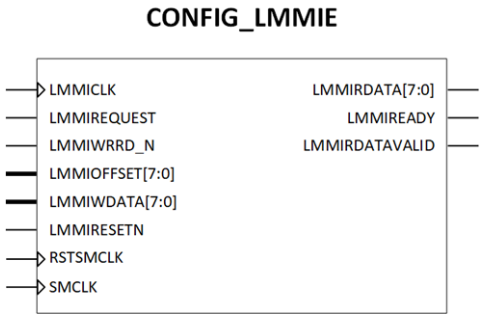


Table 74: Input Ports

Name	Description
LMMICLK	Clock for LMMI interface
LMMIREQUEST	Start transaction
LMMIWRRD_N	1=write, 0=read
LMMIOFFSET[7:0]	Register offset
LMMIWDATA[7:0]	Write data
LMMIRESETN	Reset the LMMI interface
RSTSMCLK	Reset SMCLK
SMCLK	Configuration block clock

Table 75: Output Ports

Name	Description
LMMIRDATA[7:0]	Read data
LMMIREADY	LMMI ready signal
LMMIRDATAVALID	Read transaction is complete and LMMIRDATA contains valid data

Table 76: Parameters

Name	Values	Description	Inferable
LMMI_EN	"DIS" (default) "EN"	Enable the LMMI interface	No

CRE

Cryptographic Engine

Note:

CRE is only available with the CRE Control Pack.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5-25
- LFMXO5T
- LIFCL
- UT24C
- UT24CP

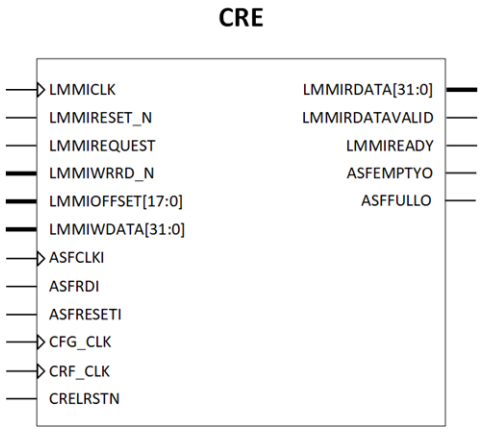


Table 77: Input Ports

Name	Range	Description
LMMICLK		Clock for LMMI interface
LMMIRESET_N		LMMI Resetn
LMMIREQUEST		Start transaction
LMMIWRRD_N		Write = HIGH, Read = LOW
LMMIOFFSET	17:0	Register offset
LMMIWDATA	31:0	Write data
ASFCLKI		Clock for ASF
ASFRDI		Read strobe
ASFRESETI		Resets the ASF logic
ASFWRI		Write strobe
CFG_CLK		Config clock
CRE_CLK		CRE clock
CRELRSTN		CRE Resetn

Table 78: Output Ports

Name	Range	Description
LMMIRDATA	31:0	Read data
LMMIRDATAVALID		Read transaction is complete and LMMIRDATA contains valid data

Table 78: Output Ports (Continued)

Name	Range	Description
LMMIREADY		LMMI ready signal
ASFEMPTYO		Indicates FIFO is empty
ASFFULLO		Indicates FIFO is full

Table 79: Parameters

Name	Value	Description
CRE_DISABLE	ENABLED *(default)* DISABLED	Disable CRE
GSR	ENABLED *(default)* DISABLED	Enable Global Set/Reset Control
OTP_EN	ENABLED DISABLED *(default)*	Enable OTP

CREA

Controls security registers to provide the security capability

Architectures supported:

- LAV-AT-ES1

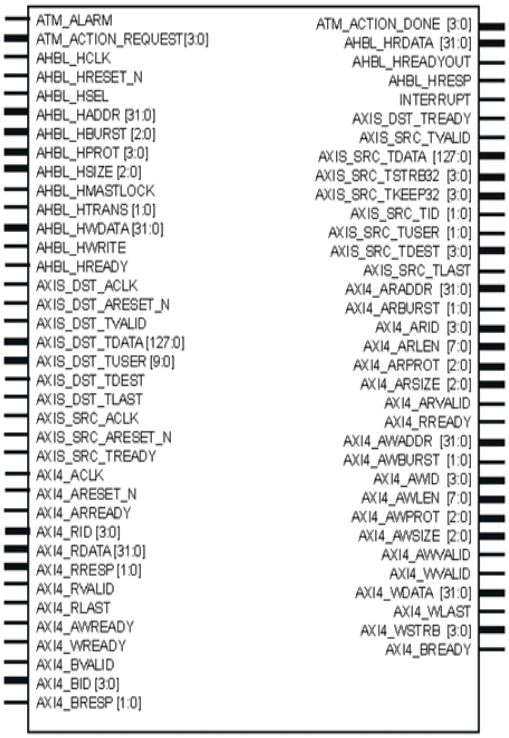


Table 80: Input Ports

Name	Range	Description
ATM_ALARM	1	ATM Alarm
ATM_ACTION_REQUEST	3:0	ATM ACTION Request
AHBL_HCLK	1	Bus Clock
AHBL_HRESET_N	1	Reset Active Low
AHBL_HSEL	1	Slave select indicates current transfer is intended for selected slave
AHBL_HADDR	31:0	Address bus
AHBL_HBURST	2:0	Indicates single or burst transfer - 4,8,and 16
AHBL_HPROT	3:0	Protection control signals about bus access
AHBL_HSIZE	2:0	Size of the transfer
AHBL_HMASTLOCK	1	When HIGH, this signal indicates that the current transfer is part of a locked sequence.
AHBL_HTRANS	1:0	Indicates transfer type - idle, busy, nonsequential, sequential
AHBL_HWDATA	31:0	Write data bus

Table 80: Input Ports (Continued)

Name	Range	Description
AHBL_HWRITE	1	Indicates transfer direction - high write / low read
AHBL_HREADY	1	When HIGH, the HREADY signal indicates to the master and all slaves, that theprevious transfer is complete.
AXIS_DST_ACLK	1	Clock signal
AXIS_DST_ARESET_N	1	Reset signal (active low)
AXIS_DST_TVALID	1	Indicates tdata[127:0] is valid when asserted (1b). When de-asserted (0b) the data on tdata is invalid.
AXIS_DST_TDATA	127:0	128-bit stream data input for AES High Speed, and 32-bit for DPA resistant, valid when tvalid is high (1b). Depending on the tuser bits, this bus contains payload or control data such as keys, IVs, etc. For all transfers, all bytes on this bus are valid.

Table 80: Input Ports (Continued)

Name	Range	Description
AXIS_DST_TUSER	9:0	10-bit for AES High Speed as follows: Bit[0] aad_in_done, indicates tdata[127:0] is the last AAD data when 1b. Bit[1] indicates ctrl_av when 1b (for stand-alone mode only). Bit[2] indicates tdata[127:0] is mode_in data (for stand-alone mode only). Bit[3] indicates tdata[127:0] is key_in[127:0] data (for stand-alone mode only). Bit[4] indicates tdata[127:0] is key_in[255:127] data (for stand-alone mode only). Bit[5] indicates tdata[127:0] is key2_in[127:0] data (for stand-alone mode only). Bit[6] indicates tdata[127:0] is key2_in[255:127] data (for stand-alone mode only). Bit[7] indicates tdata[127:0] is hash_key_in[127:0] data (for stand-alone mode only). Bit[8] indicates tdata[127:0] is iv_in[127:0] data. Bit[9] indicates tdata[127:0] is length_in[127:0] data {aad_len[63:0], ct_len[63:0]}. All tuser[9:1] bits zero means tdata[127:0] is payload data. Not used for DPA resistant.
AXIS_DST_TDEST	1	Selects data destination 0 - DPA resistant 1 - High Speed AES
AXIS_DST_TLAST	1	Indicates the last data word of a transfer. This signal must be asserted high (1b) when the last packet data is provided on the tdata bus
AXIS_SRC_ACLK	1	Clock signal. All signals are sampled on rising edge
AXIS_SRC_ARESET_N	1	Reset signal (active low)
AXIS_SRC_TREADY	1	Indicates that the slave is ready to accept the data on tdata. Data transfer will occur when both tvalid and tready are asserted.

Table 80: Input Ports (Continued)

Name	Range	Description
AXI4_ACLK	1	Clock signal
AXI4_ARESET_N	1	Reset signal (active low)
AXI4_ARREADY	1	This signal indicates that the slave is ready to accept an address and associated control signals.
AXI4_RID	3:0	This signal is the identification tag for the read data group of signals generated by the slave.
AXI4_RDATA	31:0	Read data
AXI4_RRESP	1:0	This signal indicates the status of the read transfer. 2'b00 : OKAY 2'b01 : EXOKAY 2'b10 : SLVERR 2'b11 : DECERR
AXI4_RVALID	1	This signal indicates that the channel is signaling the required read data.
AXI4_RLAST	1	This signal indicates the last transfer in a read burst.
AXI4_AWREADY	1	This signal indicates that the slave is ready to accept an address and associated control signals.
AXI4_WREADY	1	This signal indicates that the slave can accept the write data.
AXI4_BVALID	1	This signal indicates that the channel is signaling a valid write response.
AXI4_BID	3:0	This signal is the ID tag of the write response.
AXI4_BRESP	1:0	This signal indicates the status of the write transaction.

Table 81: Output Ports

Name	Range	Description
ATM_ACTION_DONE	3:0	ATM Action Done
AHBL_HRDATA	31:0	Read data bus

Table 81: Output Ports (Continued)

Name	Range	Description
AHBL_HREADYOUT	1	When HIGH, the HREADYOUT signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
AHBL_HRESP	1	When low, it indicates transfer is ok when high, it indicates transfer is error
INTERRUPT	1	Interrupt output signal
AXIS_DST_TREADY	1	Indicates that the security engine is ready to accept the data on tdata[127:0]. Data transfer will occur when both tvalid and tready are asserted.
AXIS_SRC_TVALID	1	Indicates tdata[127:0] is valid when asserted (1b). When de-asserted (0b) the data on tdata[127:0] is invalid. Data transfer will occur when both tvalid and tready are asserted.
AXIS_SRC_TDATA	127:0	128-bit stream data output from the AES High Speed, and 32-bit for DPA resistant, valid when tvalid is high (1b). Depending on the tuser[1:0] bits, this bus contains payload data, IV data or TAG data. For all transfers, all bytes on this bus are valid.
AXIS_SRC_TSTRB32	3:0	32-bit word qualifier indicates data word or position word. Valid values are 0x0 or 0x1 for data from DPA resistant core, 0x0 or 0xF for data from AES High Speed core.
AXIS_SRC_TKEEP32	3:0	32-bit word qualifier indicates data word or position word. Valid values are 0x0 or 0x1 for data from DPA resistant core, 0x0 or 0xF for data from AES High Speed core.
AXIS_SRC_TID	1:0	Indicates the master which initiated the transaction. DPA resistant: 0x0 = DMA, 0x2 = CPU AES High Speed - 0x3
AXIS_SRC_TUSER	1:0	AES High Speed: 0x0 - payload data 0x1 - IV data 0x2 - TAG data DPA resistant: 0x0 - payload data

Table 81: Output Ports (Continued)

Name	Range	Description
AXIS_SRC_TDEST	1:0	Determined by parameter passed to Security APIs
AXIS_SRC_TLAST	1	Indicates the last data word of a transfer. This signal must be asserted high (1b) when the last packet data is provided on the tdata bus.
AXI4_ARADDR	31:0	The read address gives the address of the first transfer in a read burst transaction.
AXI4_ARBURST	1:0	The burst type and the size information determine how the address for each transfer within the burst is calculated. 2'b00 : FIXED 2'b01 : INCR 2'b10 : WRAP 2'b11 : Reserved
AXI4_ARID	3:0	This signal is the identification tag for the read address group of signals.
AXI4_ARLEN	7:0	This signal indicates the exact number of transfers in a burst. 1 to beat INCR bursts supported. WRAP burst lengths of 2, 4, 8, and 16 are supported.
AXI4_ARPROT	2:0	This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
AXI4_ARSIZE	2:0	This signal indicates the size of each transfer in the burst. Byte sizes of 1 - 4 are supported.
AXI4_ARVALID	1	This signal indicates that the channel is signaling valid read address and control information.
AXI4_RREADY	1	This signal indicates that the master can accept the read data and response information.

Table 81: Output Ports (Continued)

Name	Range	Description
AXI4_AWADDR	31:0	The write address gives the address of the first transfer in a write burst transaction.
AXI4_AWBURST	1:0	The burst type and the size information, determine how the address for each transfer within the burst is calculated. This field is encoded as follows: 2'b00 : FIXED 2'b01 : INCR 2'b10 : WRAP 2'b11 : Reserved
AXI4_AWID	3:0	This signal is the identification tag for the write address group of signals.
AXI4_AWLEN	7:0	The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. 1 to beat INCR bursts supported. 2, 4, 8, or 16 length WRAP bursts supported.
AXI4_AWPROT	2:0	This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
AXI4_AWSIZE	2:0	This signal indicates the size of each transfer in the burst. 1 - 4-byte sizes are supported.
AXI4_AWVALID	1	This signal indicates that the channel is signaling valid write address and control information.
AXI4_WVALID	1	This signal indicates that valid write data and strobes are available.
AXI4_WDATA	31:0	Write data.
AXI4_WLAST	1	This signal indicates the last transfer in a write burst.

Table 81: Output Ports (Continued)

Name	Range	Description
AXI4_WSTRB	3:0	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
AXI4_BREADY	1	This signal indicates that the master can accept a write response.

Table 82: Parameters

Name	Value	Description
BITSTREAM_AUTH	"DISABLED" (default) "ENABLED"	Enable JTAG 1
BBRAM_PUBKEY_REVOKE	"DISABLED" (default) "ENABLED"	Enable BBRRAM Public Key Revoke
BBRAM	"DISABLED" "ENABLED" (default)	Enable BBRAM
ERASE_KEY_PIN	"DISABLED" (default) "ENABLED"	Enable Key Pin Erase

DCC

Dynamic Clock Control with Positive Enable

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

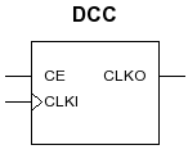


Table 83: Input Ports

Name	Description
CLKI	Clock in
CE	Clock enable

Table 84: Output Ports

Name	Description
CLKO	Clock out. Zero when CE is 0. CLKI when CE is 1.

Table 85: Parameters

Name	Value	Description
DCCEN	"0" *(default)*	Enables dynamic control.
	"1"	"0": CLKO = CLKI regardless of the CE input.
		"1": CLKO depends on the CE input.

DCCA

Dynamic clock control

Architectures supported:

- LAV-AT-ES1
- LN2-CT

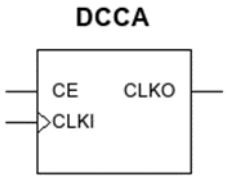


Table 86: Input Ports

Name	Description
CLKI	Input clock
CE	Clock enable

Table 87: Output Ports

Name	Description
CLKO	Output clock

DCS

Dynamic Clock Selection

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

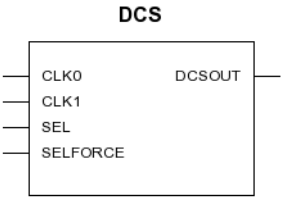


Table 88: Input Ports

Name	Description
CLK0	
CLK1	
SEL	
SELFORCE	

Table 89: Output Ports

Name	Description
DCSOUT	

Table 90: Parameters

Name	Value	Description	Output (SEL = 0)	Output (SEL = 1)
DCSMODE	"VCC"	VCC	1	1
	"GND"	VSS	0	0
	“DCS”	Falling edge triggered, latched state is low	CLK0	CLK1
	“DCS_1”	Rising edge triggered, latched state is high	CLK0	CLK1
	“BUFGCECLK1”	SEL is active high, disabled output is low	0	CLK1
	"BUFGCECLK1_1"	SEL is active high, disabled output is high	1	CLK1
	"BUFGCECLK0"	SEL is active low, disabled output is low	CLK0	0
	"BUFGCECLK0_1"	SEL is active low, disabled output is high	CLK0	1
	"BUF0"	Buffer for CLK0	CLK0	CLK0
	"BUF1"	Buffer for CLK1	CLK1	CLK1

DCSA

Dynamic clock select

Architectures supported:

- LAV-AT-ES1
- LN2-CT

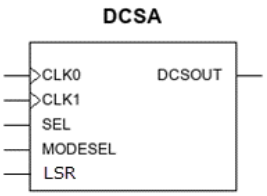


Table 91: Input Ports

Name	Description
CLK0	Input clock 0
CLK1	Input clock 1
SEL	Select between CLK0 and CLK1
MODESEL	Select between glitchless (0) and non-glitchless (1)
LSR	Local Set/Reset

Table 92: Output Ports

Name	Description
DCSOUT	Output clock

Table 93: Parameters

Name	Values	Description
DCSMODE	"CLK0" (default) "CLK1" "NEG" "POS" "CLK0_LOW" "CLK0_HIGH" "CLK1_LOW" "CLK1_HIGH" "LOW" "HIGH"	DCS Mode

DDRDLL

Delay-Locked Loop Master Block for DDR Functionality

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

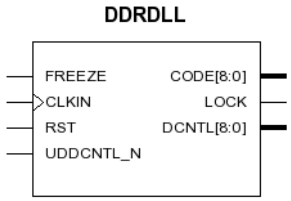


Table 94: Input Ports

Name	Description
FREEZE	
CLKIN	
RST	
UDDCNTL_N	

Table 95: Output Ports

Name	Description
CODE[8:0]	
LOCK	
DCNTL[8:0]	

Table 96: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	GSRN signal enable/disable select

Table 96: Parameters (Continued)

Name	Values	Description
ENA_ROUNDOFF	"ENABLED" *(default)* "DISABLED"	DLL counter round-off select. 0: Disable round-off 1: Enable round-off, more stable code
FORCE_MAX_DELAY	"CODE_OR_LOCK_FROM_DLL_LOOP" *(default)* "FORCE_LOCK_AND_CODE"	DLL control code forcing. 0: Code/lock generated from DLL loop 1: Force lock and code is set to maximum of 255 for lower input frequency mode.

DDRDLL can generate a phase shift code (90 degree) according to its running frequency and provide this code to every individual DQS block and DLLDEL slave delay element located in two adjacent sides if available.

DDRDLLA

DLL for creating the 90° clock delay

Architectures supported:

- LAV-AT-ES1
- LN2-CT

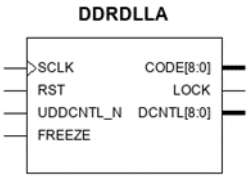


Table 97: Input Ports

Name	Range	Type	Active State	Description
CLK	1	Data	High	Clock input
RST	1	Control	High	Reset input
UDDCNTL_N	1	Control	Low	Update control to update the delay code. 0: the delay code out of the DDRDLL is updated. Should not be active during a read or a write cycle
FREEZE	1	Control	High	Releases the DDRDLL input clock

Table 98: Output Ports

Name	Range	Type	Active State	Description
CODE	8:0	Data	High	The delay codes from the DDRDLL to be used to delay input of DQS, DLLDEL and output of LPDDR4
LOCK	1	Data	High	Lock output to indicate the DDRDLL has valid delay output
DCNTL	8:0	Data	High	The delay codes from the DDRDLL available for the user IP

Table 99: Parameters

Name	Values	Type
CODE_LOCK_GENERATION	"DLL_LOOP" "FORCE_MAX"	DLL control code forcing. "DLL_LOOP": Code/lock generated from DLL loop "FORCE_MAX": Force lock and code is set to maximum of 255 for lower input frequency mode
ROUNDOFF	"ENABLED" "DISABLED"	DLL counter round-off select. "ENABLED": Enable round-off, more stable code "DISABLED": Disable round-off
DIV	"1" "2"	Clock divider
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

DDRPHY16D

Supports DDR4 and DDR3L protocols with X16 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

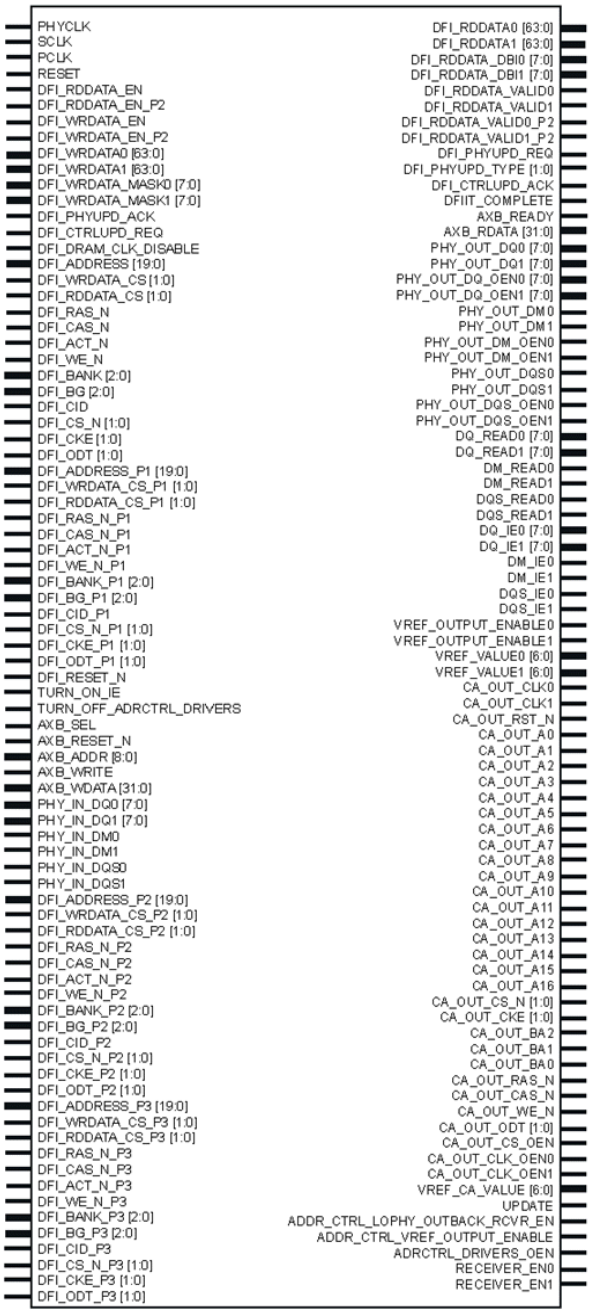


Table 100: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset

Table 100: Input Ports (Continued)

Name	Range	Description
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	19:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_RAS_N	1	DDRPHY RAS Enable
DFI_CAS_N	1	DDRPHY CAS Enable
DFI_ACT_N	1	DDRPHY ACT
DFI_WE_N	1	DDRPHY Write Enable
DFI_BANK	2:0	DDRPHY Bank Address
DFI_BG	2:0	DDRPHY Bank Group
DFI_CID	1	DDRPHY Chip ID
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	19:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_RAS_N_P1	1	DDRPHY RAS Enable Phase1
DFI_CAS_N_P1	1	DDRPHY CAS Enable Phase1
DFI_ACT_N_P1	1	DDRPHY ACT Phase1
DFI_WE_N_P1	1	DDRPHY Write Enable Phase1
DFI_BANK_P1	2:0	DDRPHY Bank Address Phase1

Table 100: Input Ports (Continued)

Name	Range	Description
DFI_BG_P1	2:0	DDRPHY Bank Group Phase1
DFI_CID_P1	1	DDRPHY Chip ID Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
DFI_ADDRESS_P2	19:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_RAS_N_P2	1	DDRPHY RAS Enable Phase2
DFI_CAS_N_P2	1	DDRPHY CAS Enable Phase2
DFI_ACT_N_P2	1	DDRPHY ACT Phase2
DFI_WE_N_P2	1	DDRPHY Write Enable Phase2
DFI_BANK_P2	2:0	DDRPHY Bank Address Phase2
DFI_BG_P2	2:0	DDRPHY Bank Group Phase2
DFI_CID_P2	1	DDRPHY Chip ID Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2

Table 100: Input Ports (Continued)

Name	Range	Description
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_RAS_N_P3	1	DDRPHY RAS Enable Phase3
DFI_CAS_N_P3	1	DDRPHY CAS Enable Phase3
DFI_ACT_N_P3	1	DDRPHY ACT Phase3
DFI_WE_N_P3	1	DDRPHY Write Enable Phase3
DFI_BANK_P3	2:0	DDRPHY Bank Address Phase3
DFI_BG_P3	2:0	DDRPHY Bank Group Phase3
DFI_CID_P3	1	DDRPHY Chip ID Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 101: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data

Table 101: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized

Table 101: Output Ports (Continued)

Name	Range	Description
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_A6	1	DRAM MA output from PHY
CA_OUT_A7	1	DRAM MA output from PHY
CA_OUT_A8	1	DRAM MA output from PHY
CA_OUT_A9	1	DRAM MA output from PHY
CA_OUT_A10	1	DRAM MA output from PHY
CA_OUT_A11	1	DRAM MA output from PHY
CA_OUT_A12	1	DRAM MA output from PHY
CA_OUT_A13	1	DRAM MA output from PHY
CA_OUT_A14	1	DRAM MA output from PHY
CA_OUT_A15	1	DRAM MA output from PHY
CA_OUT_A16	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_BA2	1	DRAM BA output from PHY
CA_OUT_BA1	1	DRAM BA output from PHY
CA_OUT_BA0	1	DRAM BA output from PHY
CA_OUT_RAS_N	1	DRAM RAS_N output from PHY
CA_OUT_CAS_N	1	DRAM CAS_N output from PHY
CA_OUT_WE_N	1	DRAM WE_N output from PHY
CA_OUT_ODT	1:0	DRAM ODT output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit

Table 101: Output Ports (Continued)

Name	Range	Description
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 102: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"DDR3L" (default) "DDR4"	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor (virtual attribute)

DDRPHY16E

Supports LPDDR4 protocol with X16 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

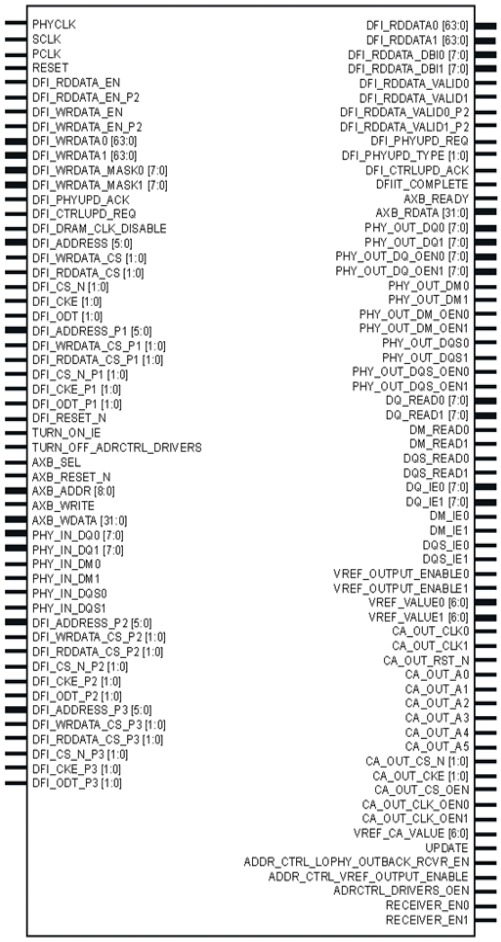


Table 103: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask

Table 103: Input Ports (Continued)

Name	Range	Description
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	5:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	5:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
DFI_ADDRESS_P2	5:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2

Table 103: Input Ports (Continued)

Name	Range	Description
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 104: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY

Table 104: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY

Table 104: Output Ports (Continued)

Name	Range	Description
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 105: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"LPDDR4" (default)	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY32D

Supports DDR4 and DDR3L protocols with X32 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

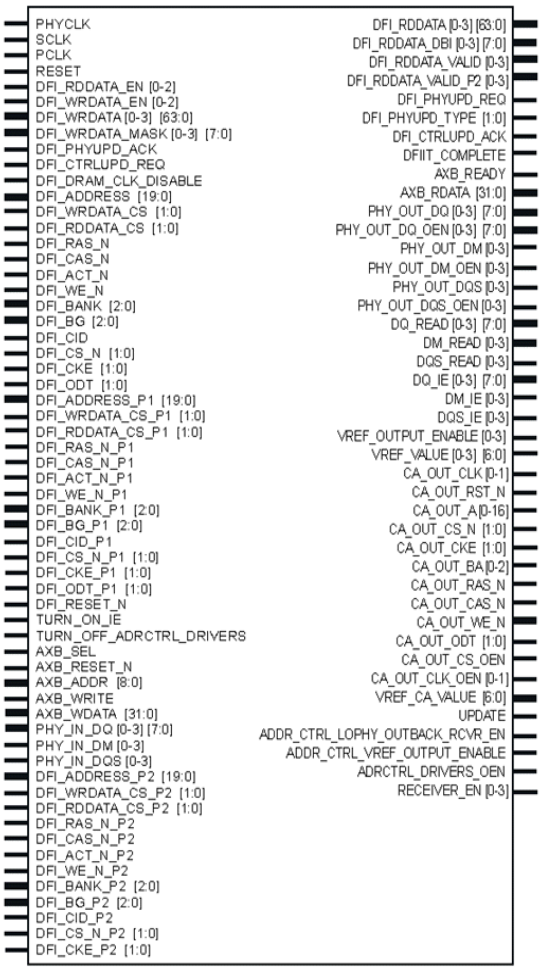


Table 106: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable

Table 106: Input Ports (Continued)

Name	Range	Description
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	19:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_RAS_N	1	DDRPHY RAS Enable
DFI_CAS_N	1	DDRPHY CAS Enable
DFI_ACT_N	1	DDRPHY ACT
DFI_WE_N	1	DDRPHY Write Enable
DFI_BANK	2:0	DDRPHY Bank Address
DFI_BG	2:0	DDRPHY Bank Group
DFI_CID	1	DDRPHY Chip ID
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	19:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_RAS_N_P1	1	DDRPHY RAS Enable Phase1
DFI_CAS_N_P1	1	DDRPHY CAS Enable Phase1

Table 106: Input Ports (Continued)

Name	Range	Description
DFI_ACT_N_P1	1	DDRPHY ACT Phase1
DFI_WE_N_P1	1	DDRPHY Write Enable Phase1
DFI_BANK_P1	2:0	DDRPHY Bank Address Phase1
DFI_BG_P1	2:0	DDRPHY Bank Group Phase1
DFI_CID_P1	1	DDRPHY Chip ID Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
DFI_ADDRESS_P2	19:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2

Table 106: Input Ports (Continued)

Name	Range	Description
DFI_RAS_N_P2	1	DDRPHY RAS Enable Phase2
DFI_CAS_N_P2	1	DDRPHY CAS Enable Phase2
DFI_ACT_N_P2	1	DDRPHY ACT Phase2
DFI_WE_N_P2	1	DDRPHY Write Enable Phase2
DFI_BANK_P2	2:0	DDRPHY Bank Address Phase2
DFI_BG_P2	2:0	DDRPHY Bank Group Phase2
DFI_CID_P2	1	DDRPHY Chip ID Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_RAS_N_P3	1	DDRPHY RAS Enable Phase3
DFI_CAS_N_P3	1	DDRPHY CAS Enable Phase3
DFI_ACT_N_P3	1	DDRPHY ACT Phase3
DFI_WE_N_P3	1	DDRPHY Write Enable Phase3
DFI_BANK_P3	2:0	DDRPHY Bank Address Phase3
DFI_BG_P3	2:0	DDRPHY Bank Group Phase3
DFI_CID_P3	1	DDRPHY Chip ID Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 107: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI

Table 107: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY

Table 107: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)

Table 107: Output Ports (Continued)

Name	Range	Description
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_A6	1	DRAM MA output from PHY
CA_OUT_A7	1	DRAM MA output from PHY

Table 107: Output Ports (Continued)

Name	Range	Description
CA_OUT_A8	1	DRAM MA output from PHY
CA_OUT_A9	1	DRAM MA output from PHY
CA_OUT_A10	1	DRAM MA output from PHY
CA_OUT_A11	1	DRAM MA output from PHY
CA_OUT_A12	1	DRAM MA output from PHY
CA_OUT_A13	1	DRAM MA output from PHY
CA_OUT_A14	1	DRAM MA output from PHY
CA_OUT_A15	1	DRAM MA output from PHY
CA_OUT_A16	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_BA2	1	DRAM BA output from PHY
CA_OUT_BA1	1	DRAM BA output from PHY
CA_OUT_BA0	1	DRAM BA output from PHY
CA_OUT_RAS_N	1	DRAM RAS_N output from PHY
CA_OUT_CAS_N	1	DRAM CAS_N output from PHY
CA_OUT_WE_N	1	DRAM WE_N output from PHY
CA_OUT_ODT	1:0	DRAM ODT output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable mclk output
CA_OUT_CLK_OEN1	1	Disable mclk output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 107: Output Ports (Continued)

Name	Range	Description
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 108: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"DDR3L" (default) "DDR4"	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY32E

Supports LPDDR4 protocol with X32 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

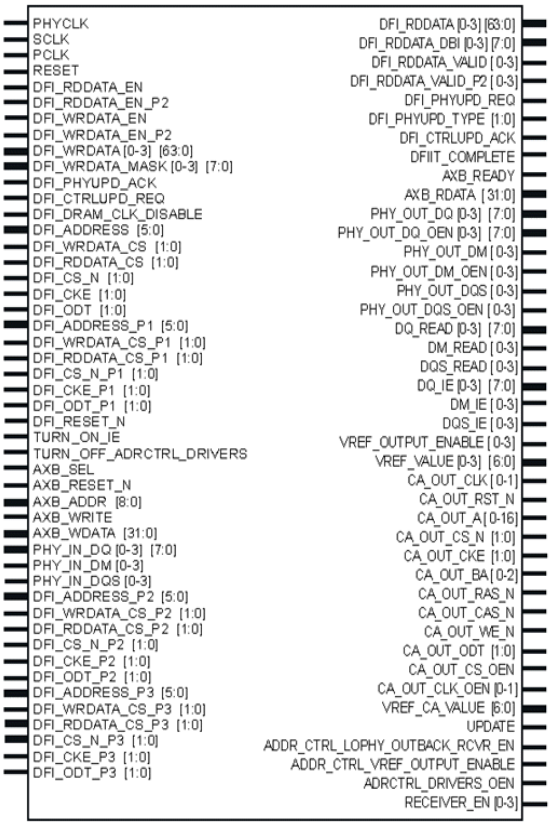


Table 109: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask

Table 109: Input Ports (Continued)

Name	Range	Description
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	5:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	5:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY

Table 109: Input Ports (Continued)

Name	Range	Description
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
DFI_ADDRESS_P2	5:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	5:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 110: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid

Table 110: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY

Table 110: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads

Table 110: Output Ports (Continued)

Name	Range	Description
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable mclk output
CA_OUT_CLK_OEN1	1	Disable mclk output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring

Table 110: Output Ports (Continued)

Name	Range	Description
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 111: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"LPDDR4" (default)	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY40D

Supports DDR4 and DDR3L protocols with X40 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

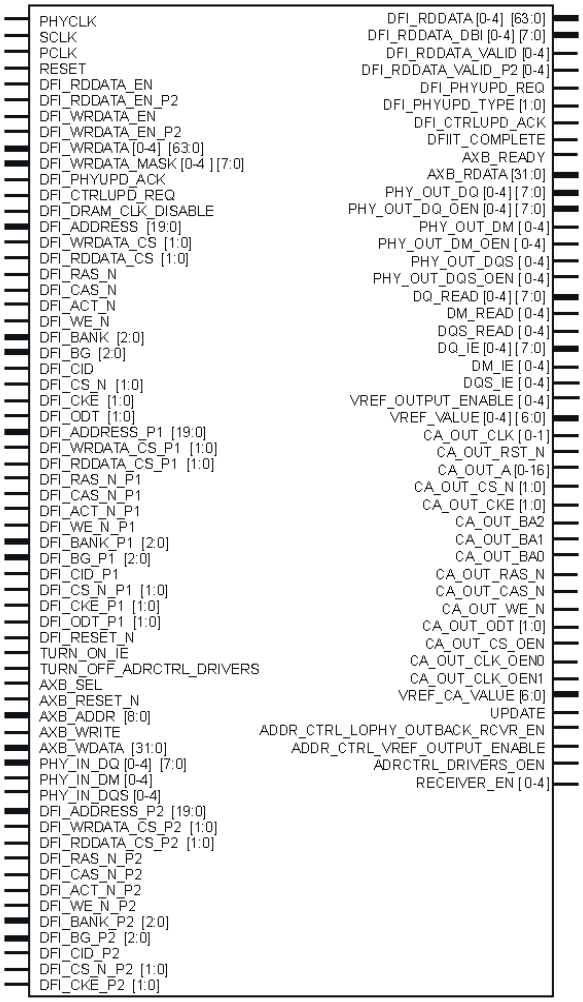


Table 112: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data

Table 112: Input Ports (Continued)

Name	Range	Description
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA4	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK4	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_ADDRESS	19:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_RAS_N	1	DDRPHY RAS Enable
DFI_CAS_N	1	DDRPHY CAS Enable
DFI_ACT_N	1	DDRPHY ACT
DFI_WE_N	1	DDRPHY Write Enable
DFI_BANK	2:0	DDRPHY Bank Address
DFI_BG	2:0	DDRPHY Bank Group
DFI_CID	1	DDRPHY Chip ID
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	19:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_RAS_N_P1	1	DDRPHY RAS Enable Phase1
DFI_CAS_N_P1	1	DDRPHY CAS Enable Phase1
DFI_ACT_N_P1	1	DDRPHY ACT Phase1
DFI_WE_N_P1	1	DDRPHY Write Enable Phase1
DFI_BANK_P1	2:0	DDRPHY Bank Address Phase1
DFI_BG_P1	2:0	DDRPHY Bank Group Phase1

Table 112: Input Ports (Continued)

Name	Range	Description
DFI_CID_P1	1	DDRPHY Chip ID Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DQ4	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM4	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
PHY_IN_DQS4	1	DQS data strobe to PHY
DFI_ADDRESS_P2	19:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_RAS_N_P2	1	DDRPHY RAS Enable Phase2

Table 112: Input Ports (Continued)

Name	Range	Description
DFI_CAS_N_P2	1	DDRPHY CAS Enable Phase2
DFI_ACT_N_P2	1	DDRPHY ACT Phase2
DFI_WE_N_P2	1	DDRPHY Write Enable Phase2
DFI_BANK_P2	2:0	DDRPHY Bank Address Phase2
DFI_BG_P2	2:0	DDRPHY Bank Group Phase2
DFI_CID_P2	1	DDRPHY Chip ID Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_RAS_N_P3	1	DDRPHY RAS Enable Phase3
DFI_CAS_N_P3	1	DDRPHY CAS Enable Phase3
DFI_ACT_N_P3	1	DDRPHY ACT Phase3
DFI_WE_N_P3	1	DDRPHY Write Enable Phase3
DFI_BANK_P3	2:0	DDRPHY Bank Address Phase3
DFI_BG_P3	2:0	DDRPHY Bank Group Phase3
DFI_CID_P3	1	DDRPHY Chip ID Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 113: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA4	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI

Table 113: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI4	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID4	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID4_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ4	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN4	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY

Table 113: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM4	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DM_OEN4	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY
PHY_OUT_DQS4	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN4	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DQ_READ4	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DM_READ4	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe

Table 113: Output Ports (Continued)

Name	Range	Description
DQS_READ4	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE4	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE4	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE4	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required

Table 113: Output Ports (Continued)

Name	Range	Description
VREF_OUTPUT_ENABLE4	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE4	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_A6	1	DRAM MA output from PHY
CA_OUT_A7	1	DRAM MA output from PHY
CA_OUT_A8	1	DRAM MA output from PHY
CA_OUT_A9	1	DRAM MA output from PHY
CA_OUT_A10	1	DRAM MA output from PHY
CA_OUT_A11	1	DRAM MA output from PHY
CA_OUT_A12	1	DRAM MA output from PHY
CA_OUT_A13	1	DRAM MA output from PHY
CA_OUT_A14	1	DRAM MA output from PHY
CA_OUT_A15	1	DRAM MA output from PHY
CA_OUT_A16	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_BA2	1	DRAM BA output from PHY

Table 113: Output Ports (Continued)

Name	Range	Description
CA_OUT_BA1	1	DRAM BA output from PHY
CA_OUT_BA0	1	DRAM BA output from PHY
CA_OUT_RAS_N	1	DRAM RAS_N output from PHY
CA_OUT_CAS_N	1	DRAM CAS_N output from PHY
CA_OUT_WE_N	1	DRAM WE_N output from PHY
CA_OUT_ODT	1:0	DRAM ODT output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN4	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 114: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate

Table 114: Parameters (Continued)

Name	Values	Type
PROTOCOL	"DDR3L" (default) "DDR4"	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY64D

Supports DDR4 and DDR3L protocols with X64 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

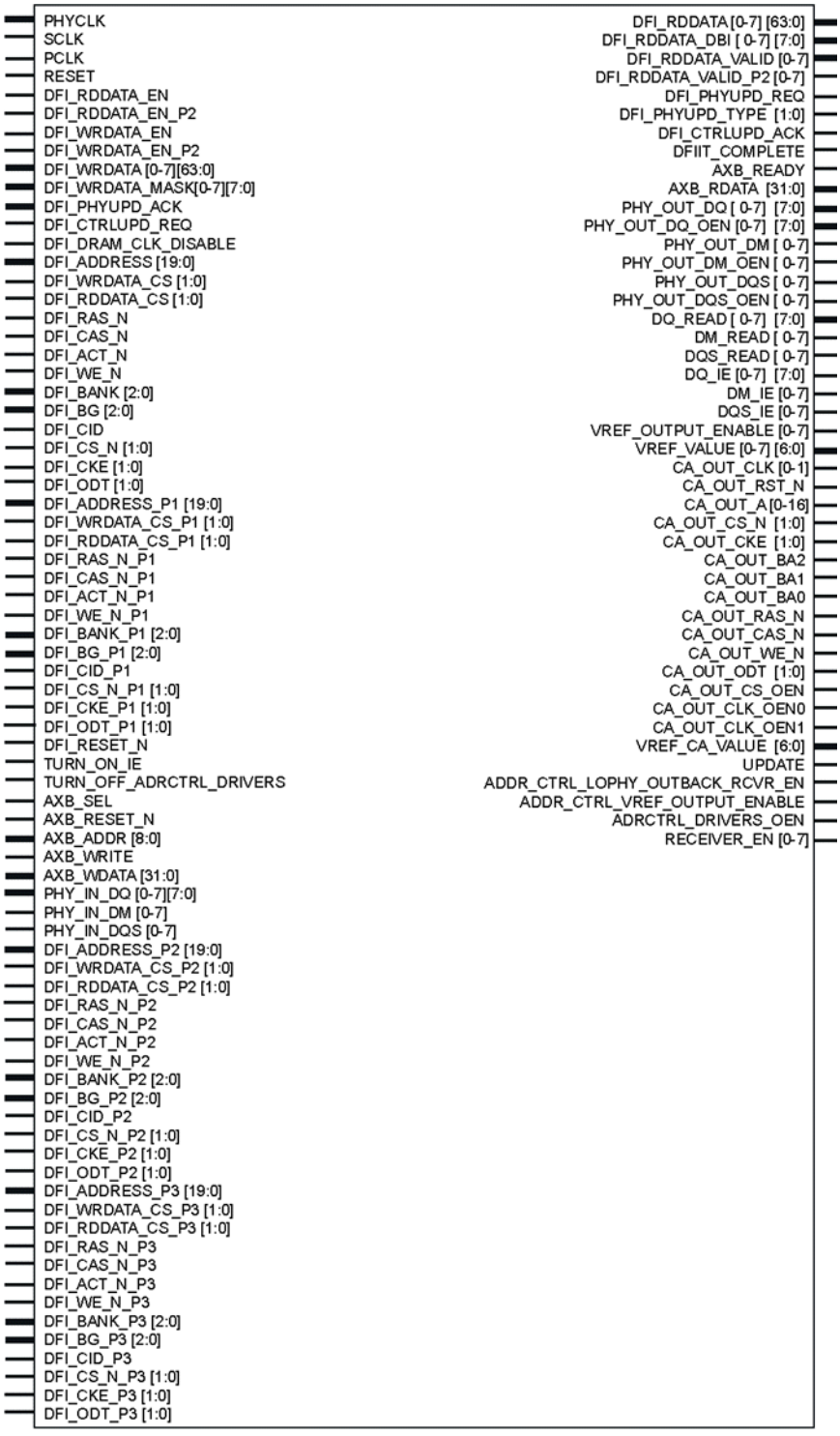

PHYCLK
SCLK
PCLK
RESET
DFI_RDDATA_EN
DFI_RDDATA_EN_P2
DFI_WRDATA_EN
DFI_WRDATA_EN_P2
DFI_WRDATA [0-7][63:0]
DFI_WRDATA_MASK[0-7][7:0]
DFI_PHYUPD_ACK
DFI_CTRLUPD_REQ
DFI_DRAM_CLK_DISABLE
DFI_ADDRESS [19:0]
DFI_WRDATA_CS [1:0]
DFI_RDDATA_CS [1:0]
DFI_RAS_N
DFI_CAS_N
DFI_ACT_N
DFI_WE_N
DFI_BANK [2:0]
DFI_BG [2:0]
DFI_CID
DFI_CS_N [1:0]
DFI_CKE [1:0]
DFI_ODT [1:0]
DFI_ADDRESS_P1 [19:0]
DFI_WRDATA_CS_P1 [1:0]
DFI_RDDATA_CS_P1 [1:0]
DFI_RAS_N_P1
DFI_CAS_N_P1
DFI_ACT_N_P1
DFI_WE_N_P1
DFI_BANK_P1 [2:0]
DFI_BG_P1 [2:0]
DFI_CID_P1
DFI_CS_N_P1 [1:0]
DFI_CKE_P1 [1:0]
DFI_ODT_P1 [1:0]
DFI_RESET_N
TURN_ON_IE
TURN_OFF_ADRCTRL_DRIVERS
AXB_SEL
AXB_RESET_N
AXB_ADDR [8:0]
AXB_WRITE
AXB_WDATA [31:0]
PHY_IN_DQ [0-7][7:0]
PHY_IN_DM [0-7]
PHY_IN_DQS [0-7]
DFI_ADDRESS_P2 [19:0]
DFI_WRDATA_CS_P2 [1:0]
DFI_RDDATA_CS_P2 [1:0]
DFI_RAS_N_P2
DFI_CAS_N_P2
DFI_ACT_N_P2
DFI_WE_N_P2
DFI_BANK_P2 [2:0]
DFI_BG_P2 [2:0]
DFI_CID_P2
DFI_CS_N_P2 [1:0]
DFI_CKE_P2 [1:0]
DFI_ODT_P2 [1:0]
DFI_ADDRESS_P3 [19:0]
DFI_WRDATA_CS_P3 [1:0]
DFI_RDDATA_CS_P3 [1:0]
DFI_RAS_N_P3
DFI_CAS_N_P3
DFI_ACT_N_P3
DFI_WE_N_P3
DFI_BANK_P3 [2:0]
DFI_BG_P3 [2:0]
DFI_CID_P3
DFI_CS_N_P3 [1:0]
DFI_CKE_P3 [1:0]
DFI_ODT_P3 [1:0]
DFI_RDDATA [0-7] [63:0]
DFI_RDDATA_DBI [0-7] [7:0]
DFI_RDDATA_VALID [0-7]
DFI_RDDATA_VALID_P2 [0-7]
DFI_PHYUPD_REQ
DFI_PHYUPD_TYPE [1:0]
DFI_CTRLUPD_ACK
DFIIT_COMPLETE
AXB_READY
AXB_RDATA [31:0]
PHY_OUT_DQ [0-7] [7:0]
PHY_OUT_DQ_OEN [0-7] [7:0]
PHY_OUT_DM [0-7]
PHY_OUT_DM_OEN [0-7]
PHY_OUT_DQS [0-7]
PHY_OUT_DQS_OEN [0-7]
DQ_READ [0-7] [7:0]
DM_READ [0-7]
DQS_READ [0-7]
DQ_IE [0-7] [7:0]
DM_IE [0-7]
DQS_IE [0-7]
VREF_OUTPUT_ENABLE [0-7]
VREF_VALUE [0-7] [6:0]
CA_OUT_CLK [0-1]
CA_OUT_RST_N
CA_OUT_A [0-16]
CA_OUT_CS_N [1:0]
CA_OUT_CKE [1:0]
CA_OUT_BA2
CA_OUT_BA1
CA_OUT_BA0
CA_OUT_RAS_N
CA_OUT_CAS_N
CA_OUT_WE_N
CA_OUT_ODT [1:0]
CA_OUT_CS_OEN
CA_OUT_CLK_OEN0
CA_OUT_CLK_OEN1
VREF_CA_VALUE [6:0]
UPDATE
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN
ADDR_CTRL_VREF_OUTPUT_ENABLE
ADRCTRL_DRIVERS_OEN
RECEIVER_EN [0-7]

Table 115: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA4	63:0	DDRPHY Write Data
DFI_WRDATA5	63:0	DDRPHY Write Data
DFI_WRDATA6	63:0	DDRPHY Write Data
DFI_WRDATA7	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK4	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK5	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK6	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK7	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	19:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_RAS_N	1	DDRPHY RAS Enable
DFI_CAS_N	1	DDRPHY CAS Enable

Table 115: Input Ports (Continued)

Name	Range	Description
DFI_ACT_N	1	DDRPHY ACT
DFI_WE_N	1	DDRPHY Write Enable
DFI_BANK	2:0	DDRPHY Bank Address
DFI_BG	2:0	DDRPHY Bank Group
DFI_CID	1	DDRPHY Chip ID
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	19:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_RAS_N_P1	1	DDRPHY RAS Enable Phase1
DFI_CAS_N_P1	1	DDRPHY CAS Enable Phase1
DFI_ACT_N_P1	1	DDRPHY ACT Phase1
DFI_WE_N_P1	1	DDRPHY Write Enable Phase1
DFI_BANK_P1	2:0	DDRPHY Bank Address Phase1
DFI_BG_P1	2:0	DDRPHY Bank Group Phase1
DFI_CID_P1	1	DDRPHY Chip ID Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY

Table 115: Input Ports (Continued)

Name	Range	Description
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DQ4	7:0	DQ Data Bus to PHY
PHY_IN_DQ5	7:0	DQ Data Bus to PHY
PHY_IN_DQ6	7:0	DQ Data Bus to PHY
PHY_IN_DQ7	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM4	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM5	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM6	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM7	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
PHY_IN_DQS4	1	DQS data strobe to PHY
PHY_IN_DQS5	1	DQS data strobe to PHY
PHY_IN_DQS6	1	DQS data strobe to PHY
PHY_IN_DQS7	1	DQS data strobe to PHY
DFI_ADDRESS_P2	19:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_RAS_N_P2	1	DDRPHY RAS Enable Phase2
DFI_CAS_N_P2	1	DDRPHY CAS Enable Phase2
DFI_ACT_N_P2	1	DDRPHY ACT Phase2
DFI_WE_N_P2	1	DDRPHY Write Enable Phase2
DFI_BANK_P2	2:0	DDRPHY Bank Address Phase2
DFI_BG_P2	2:0	DDRPHY Bank Group Phase2
DFI_CID_P2	1	DDRPHY Chip ID Phase2

Table 115: Input Ports (Continued)

Name	Range	Description
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_RAS_N_P3	1	DDRPHY RAS Enable Phase3
DFI_CAS_N_P3	1	DDRPHY CAS Enable Phase3
DFI_ACT_N_P3	1	DDRPHY ACT Phase3
DFI_WE_N_P3	1	DDRPHY Write Enable Phase3
DFI_BANK_P3	2:0	DDRPHY Bank Address Phase3
DFI_BG_P3	2:0	DDRPHY Bank Group Phase3
DFI_CID_P3	1	DDRPHY Chip ID Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 116: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA4	63:0	DDRPHY Read Data
DFI_RDDATA5	63:0	DDRPHY Read Data
DFI_RDDATA6	63:0	DDRPHY Read Data
DFI_RDDATA7	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI

Table 116: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA_DBI4	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI5	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI6	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI7	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID4	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID5	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID6	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID7	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID4_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID5_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID6_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID7_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ4	7:0	DQ data bus from PHY
PHY_OUT_DQ5	7:0	DQ data bus from PHY

Table 116: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQ6	7:0	DQ data bus from PHY
PHY_OUT_DQ7	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN4	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN5	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN6	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN7	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM4	1	DM (Data Mask) from PHY
PHY_OUT_DM5	1	DM (Data Mask) from PHY
PHY_OUT_DM6	1	DM (Data Mask) from PHY
PHY_OUT_DM7	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DM_OEN4	1	Enable for DM output from PHY
PHY_OUT_DM_OEN5	1	Enable for DM output from PHY
PHY_OUT_DM_OEN6	1	Enable for DM output from PHY
PHY_OUT_DM_OEN7	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY
PHY_OUT_DQS4	1	DQS data strobe from PHY
PHY_OUT_DQS5	1	DQS data strobe from PHY

Table 116: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQS6	1	DQS data strobe from PHY
PHY_OUT_DQS7	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN4	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN5	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN6	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN7	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DQ_READ4	7:0	FPGA IO ODT Enable For Data
DQ_READ5	7:0	FPGA IO ODT Enable For Data
DQ_READ6	7:0	FPGA IO ODT Enable For Data
DQ_READ7	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DM_READ4	1	FPGA IO ODT enable for data mask
DM_READ5	1	FPGA IO ODT enable for data mask
DM_READ6	1	FPGA IO ODT enable for data mask
DM_READ7	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe
DQS_READ4	1	FPGA IO ODT enable for data strobe
DQS_READ5	1	FPGA IO ODT enable for data strobe

Table 116: Output Ports (Continued)

Name	Range	Description
DQS_READ6	1	FPGA IO ODT enable for data strobe
DQS_READ7	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE4	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE5	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE6	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE7	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE4	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE5	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE6	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE7	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads

Table 116: Output Ports (Continued)

Name	Range	Description
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE4	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE5	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE6	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE7	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE4	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE5	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE6	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE7	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE4	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE5	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE6	6:0	Vref value set for each byte lane - width can be customized

Table 116: Output Ports (Continued)

Name	Range	Description
VREF_VALUE7	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_A6	1	DRAM MA output from PHY
CA_OUT_A7	1	DRAM MA output from PHY
CA_OUT_A8	1	DRAM MA output from PHY
CA_OUT_A9	1	DRAM MA output from PHY
CA_OUT_A10	1	DRAM MA output from PHY
CA_OUT_A11	1	DRAM MA output from PHY
CA_OUT_A12	1	DRAM MA output from PHY
CA_OUT_A13	1	DRAM MA output from PHY
CA_OUT_A14	1	DRAM MA output from PHY
CA_OUT_A15	1	DRAM MA output from PHY
CA_OUT_A16	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_BA2	1	DRAM BA output from PHY
CA_OUT_BA1	1	DRAM BA output from PHY
CA_OUT_BA0	1	DRAM BA output from PHY
CA_OUT_RAS_N	1	DRAM RAS_N output from PHY
CA_OUT_CAS_N	1	DRAM CAS_N output from PHY
CA_OUT_WE_N	1	DRAM WE_N output from PHY
CA_OUT_ODT	1:0	DRAM ODT output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit

Table 116: Output Ports (Continued)

Name	Range	Description
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN4	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN5	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN6	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN7	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 117: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"DDR3L" (default) "DDR4"	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY64E

Supports LPDDR4 protocol with X64 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

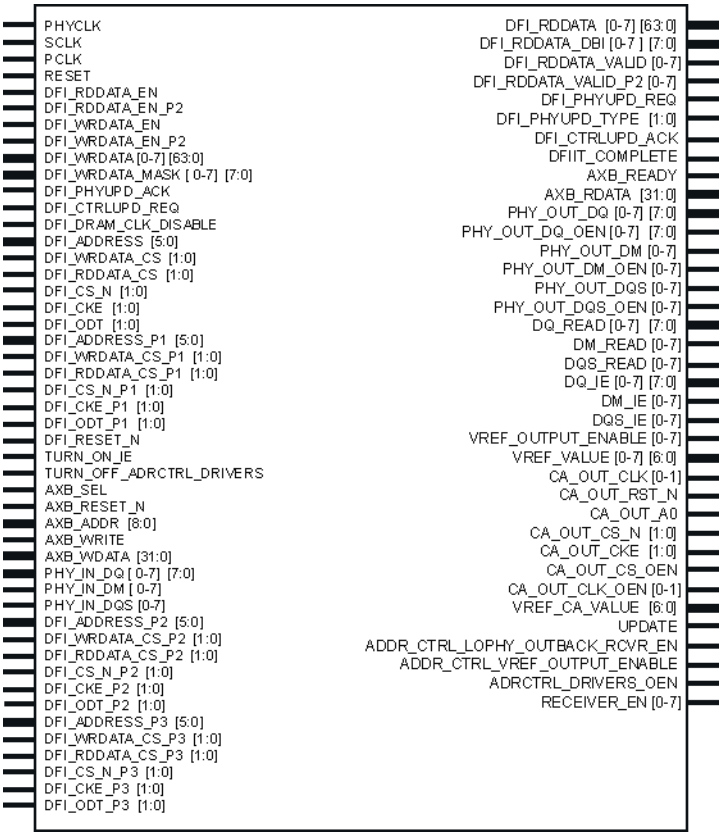


Table 118: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3

Table 118: Input Ports (Continued)

Name	Range	Description
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA4	63:0	DDRPHY Write Data
DFI_WRDATA5	63:0	DDRPHY Write Data
DFI_WRDATA6	63:0	DDRPHY Write Data
DFI_WRDATA7	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK4	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK5	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK6	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK7	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	5:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	5:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1

Table 118: Input Ports (Continued)

Name	Range	Description
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DQ4	7:0	DQ Data Bus to PHY
PHY_IN_DQ5	7:0	DQ Data Bus to PHY
PHY_IN_DQ6	7:0	DQ Data Bus to PHY
PHY_IN_DQ7	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM4	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM5	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM6	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM7	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
PHY_IN_DQS4	1	DQS data strobe to PHY
PHY_IN_DQS5	1	DQS data strobe to PHY

Table 118: Input Ports (Continued)

Name	Range	Description
PHY_IN_DQS6	1	DQS data strobe to PHY
PHY_IN_DQS7	1	DQS data strobe to PHY
DFI_ADDRESS_P2	5:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	5:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 119: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA4	63:0	DDRPHY Read Data
DFI_RDDATA5	63:0	DDRPHY Read Data
DFI_RDDATA6	63:0	DDRPHY Read Data
DFI_RDDATA7	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI4	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI5	7:0	DDRPHY Read Data DBI

Table 119: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA_DBI6	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI7	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID4	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID5	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID6	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID7	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID4_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID5_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID6_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID7_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ4	7:0	DQ data bus from PHY
PHY_OUT_DQ5	7:0	DQ data bus from PHY
PHY_OUT_DQ6	7:0	DQ data bus from PHY
PHY_OUT_DQ7	7:0	DQ data bus from PHY

Table 119: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN4	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN5	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN6	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN7	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM4	1	DM (Data Mask) from PHY
PHY_OUT_DM5	1	DM (Data Mask) from PHY
PHY_OUT_DM6	1	DM (Data Mask) from PHY
PHY_OUT_DM7	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DM_OEN4	1	Enable for DM output from PHY
PHY_OUT_DM_OEN5	1	Enable for DM output from PHY
PHY_OUT_DM_OEN6	1	Enable for DM output from PHY
PHY_OUT_DM_OEN7	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY
PHY_OUT_DQS4	1	DQS data strobe from PHY
PHY_OUT_DQS5	1	DQS data strobe from PHY
PHY_OUT_DQS6	1	DQS data strobe from PHY
PHY_OUT_DQS7	1	DQS data strobe from PHY

Table 119: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN4	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN5	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN6	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN7	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DQ_READ4	7:0	FPGA IO ODT Enable For Data
DQ_READ5	7:0	FPGA IO ODT Enable For Data
DQ_READ6	7:0	FPGA IO ODT Enable For Data
DQ_READ7	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DM_READ4	1	FPGA IO ODT enable for data mask
DM_READ5	1	FPGA IO ODT enable for data mask
DM_READ6	1	FPGA IO ODT enable for data mask
DM_READ7	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe
DQS_READ4	1	FPGA IO ODT enable for data strobe
DQS_READ5	1	FPGA IO ODT enable for data strobe
DQS_READ6	1	FPGA IO ODT enable for data strobe
DQS_READ7	1	FPGA IO ODT enable for data strobe

Table 119: Output Ports (Continued)

Name	Range	Description
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE4	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE5	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE6	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE7	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE4	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE5	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE6	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE7	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads

Table 119: Output Ports (Continued)

Name	Range	Description
DQS_IE4	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE5	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE6	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE7	1	Optional dynamic input enable for DQS IO receivers during reads
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE4	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE5	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE6	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE7	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE4	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE5	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE6	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE7	6:0	Vref value set for each byte lane - width can be customized

Table 119: Output Ports (Continued)

Name	Range	Description
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN4	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN5	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 119: Output Ports (Continued)

Name	Range	Description
RECEIVER_EN6	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN7	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 120: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"LPDDR4" (default)	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DDRPHY72D

Supports DDR4 and DDR3L protocols with X72 data width

Architectures supported:

- LAV-AT-ES1
- LN2-CT

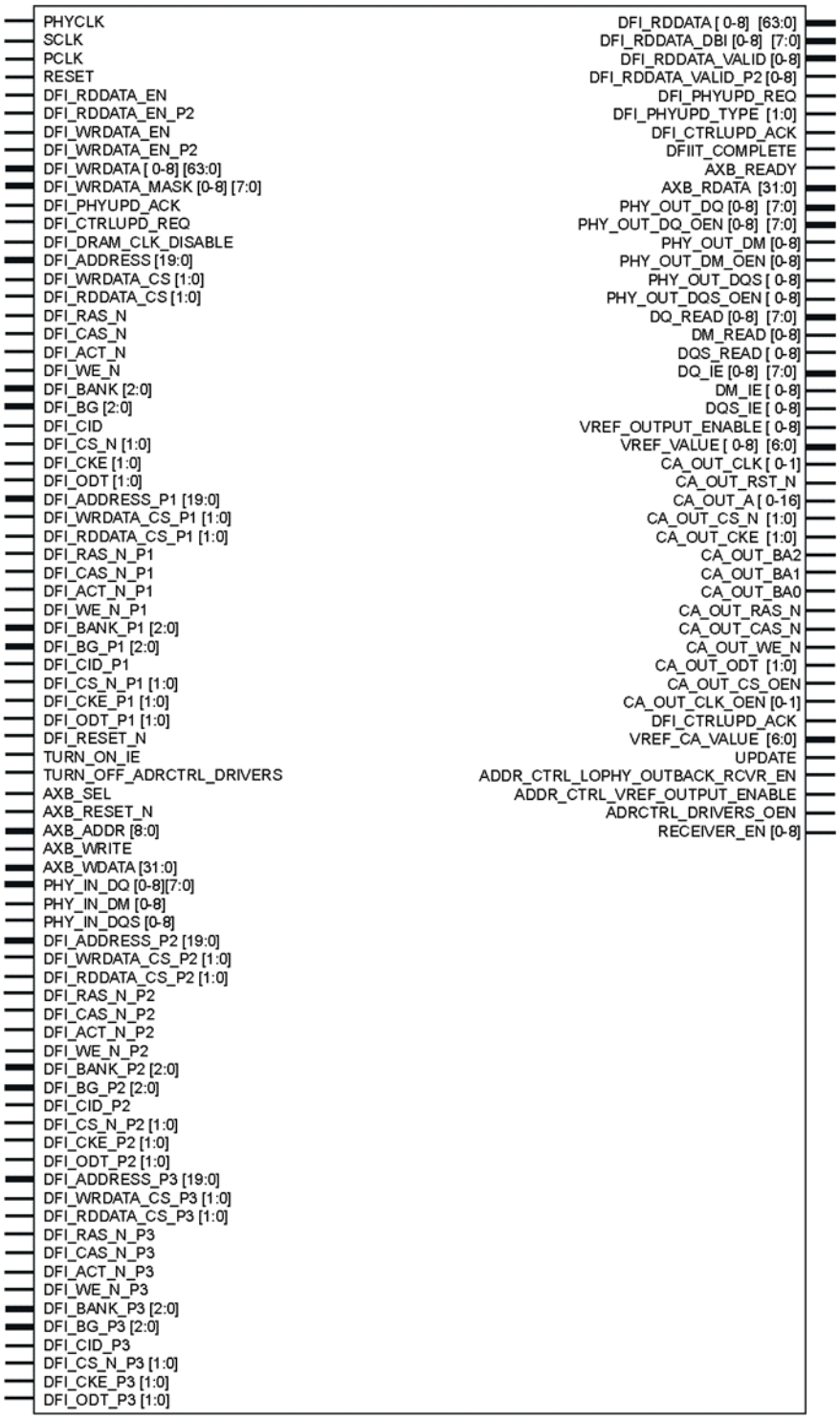

PHYCLK
SCLK
PCLK
RESET
DFI_RDDATA_EN
DFI_RDDATA_EN_P2
DFI_WRDATA_EN
DFI_WRDATA_EN_P2
DFI_WRDATA [0-8] [63:0]
DFI_WRDATA_MASK [0-8] [7:0]
DFI_PHYUPD_ACK
DFI_CTRLUPD_REQ
DFI_DRAM_CLK_DISABLE
DFI_ADDRESS [19:0]
DFI_WRDATA_CS [1:0]
DFI_RDDATA_CS [1:0]
DFI_RAS_N
DFI_CAS_N
DFI_ACT_N
DFI_WE_N
DFI_BANK [2:0]
DFI_BG [2:0]
DFI_CID
DFI_CS_N [1:0]
DFI_CKE [1:0]
DFI_ODT [1:0]
DFI_ADDRESS_P1 [19:0]
DFI_WRDATA_CS_P1 [1:0]
DFI_RDDATA_CS_P1 [1:0]
DFI_RAS_N_P1
DFI_CAS_N_P1
DFI_ACT_N_P1
DFI_WE_N_P1
DFI_BANK_P1 [2:0]
DFI_BG_P1 [2:0]
DFI_CID_P1
DFI_CS_N_P1 [1:0]
DFI_CKE_P1 [1:0]
DFI_ODT_P1 [1:0]
DFI_RESET_N
TURN_ON_IE
TURN_OFF_ADRCTRL_DRIVERS
AXB_SEL
AXB_RESET_N
AXB_ADDR [8:0]
AXB_WRITE
AXB_WDATA [31:0]
PHY_IN_DQ [0-8][7:0]
PHY_IN_DM [0-8]
PHY_IN_DQS [0-8]
DFI_ADDRESS_P2 [19:0]
DFI_WRDATA_CS_P2 [1:0]
DFI_RDDATA_CS_P2 [1:0]
DFI_RAS_N_P2
DFI_CAS_N_P2
DFI_ACT_N_P2
DFI_WE_N_P2
DFI_BANK_P2 [2:0]
DFI_BG_P2 [2:0]
DFI_CID_P2
DFI_CS_N_P2 [1:0]
DFI_CKE_P2 [1:0]
DFI_ODT_P2 [1:0]
DFI_ADDRESS_P3 [19:0]
DFI_WRDATA_CS_P3 [1:0]
DFI_RDDATA_CS_P3 [1:0]
DFI_RAS_N_P3
DFI_CAS_N_P3
DFI_ACT_N_P3
DFI_WE_N_P3
DFI_BANK_P3 [2:0]
DFI_BG_P3 [2:0]
DFI_CID_P3
DFI_CS_N_P3 [1:0]
DFI_CKE_P3 [1:0]
DFI_ODT_P3 [1:0]
DFI_RDDATA [0-8] [63:0]
DFI_RDDATA_DBI [0-8] [7:0]
DFI_RDDATA_VALID [0-8]
DFI_RDDATA_VALID_P2 [0-8]
DFI_PHYUPD_REQ
DFI_PHYUPD_TYPE [1:0]
DFI_CTRLUPD_ACK
DFIIT_COMPLETE
AXB_READY
AXB_RDATA [31:0]
PHY_OUT_DQ [0-8] [7:0]
PHY_OUT_DQ_OEN [0-8] [7:0]
PHY_OUT_DM [0-8]
PHY_OUT_DM_OEN [0-8]
PHY_OUT_DQS [0-8]
PHY_OUT_DQS_OEN [0-8]
DQ_READ [0-8] [7:0]
DM_READ [0-8]
DQS_READ [0-8]
DQ_IE [0-8] [7:0]
DM_IE [0-8]
DQS_IE [0-8]
VREF_OUTPUT_ENABLE [0-8]
VREF_VALUE [0-8] [6:0]
CA_OUT_CLK [0-1]
CA_OUT_RST_N
CA_OUT_A [0-16]
CA_OUT_CS_N [1:0]
CA_OUT_CKE [1:0]
CA_OUT_BA2
CA_OUT_BA1
CA_OUT_BA0
CA_OUT_RAS_N
CA_OUT_CAS_N
CA_OUT_WE_N
CA_OUT_ODT [1:0]
CA_OUT_CS_OEN
CA_OUT_CLK_OEN [0-1]
DFI_CTRLUPD_ACK
VREF_CA_VALUE [6:0]
UPDATE
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN
ADDR_CTRL_VREF_OUTPUT_ENABLE
ADRCTRL_DRIVERS_OEN
RECEIVER_EN [0-8]

Table 121: Input Ports

Name	Range	Description
PHYCLK	1	DDR Clock
SCLK	1	Primary Clock
PCLK	1	AXB Bus Interface Clock
RESET	1	Reset
DFI_RDDATA_EN	1	DDRPHY Read Data Enable
DFI_RDDATA_EN_P2	1	DDRPHY Read Data Enable P2P3
DFI_WRDATA_EN	1	DDRPHY Write Data Enable
DFI_WRDATA_EN_P2	1	DDRPHY Write Data Enable P2P3
DFI_WRDATA0	63:0	DDRPHY Write Data
DFI_WRDATA1	63:0	DDRPHY Write Data
DFI_WRDATA2	63:0	DDRPHY Write Data
DFI_WRDATA3	63:0	DDRPHY Write Data
DFI_WRDATA4	63:0	DDRPHY Write Data
DFI_WRDATA5	63:0	DDRPHY Write Data
DFI_WRDATA6	63:0	DDRPHY Write Data
DFI_WRDATA7	63:0	DDRPHY Write Data
DFI_WRDATA8	63:0	DDRPHY Write Data
DFI_WRDATA_MASK0	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK1	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK2	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK3	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK4	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK5	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK6	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK7	7:0	DDRPHY Write Data Mask
DFI_WRDATA_MASK8	7:0	DDRPHY Write Data Mask
DFI_PHYUPD_ACK	1	DDRPHY PHY update acknowledge
DFI_CTRLUPD_REQ	1	DDRPHY PHY update request
DFI_DRAM_CLK_DISABLE	1	DRAM clock disable
DFI_ADDRESS	19:0	DDRPHY address
DFI_WRDATA_CS	1:0	DDRPHY Write Data Chip Select
DFI_RDDATA_CS	1:0	DDRPHY Read Data Chip Select

Table 121: Input Ports (Continued)

Name	Range	Description
DFI_RAS_N	1	DDRPHY RAS Enable
DFI_CAS_N	1	DDRPHY CAS Enable
DFI_ACT_N	1	DDRPHY ACT
DFI_WE_N	1	DDRPHY Write Enable
DFI_BANK	2:0	DDRPHY Bank Address
DFI_BG	2:0	DDRPHY Bank Group
DFI_CID	1	DDRPHY Chip ID
DFI_CS_N	1:0	DDRPHY Chip Select
DFI_CKE	1:0	DDRPHY Clock Enable
DFI_ODT	1:0	DDRPHY ODT
DFI_ADDRESS_P1	19:0	DDRPHY Phase1 Address
DFI_WRDATA_CS_P1	1:0	DDRPHY Write Data Chip Select Phase1
DFI_RDDATA_CS_P1	1:0	DDRPHY Read Data Chip Select Phase1
DFI_RAS_N_P1	1	DDRPHY RAS Enable Phase1
DFI_CAS_N_P1	1	DDRPHY CAS Enable Phase1
DFI_ACT_N_P1	1	DDRPHY ACT Phase1
DFI_WE_N_P1	1	DDRPHY Write Enable Phase1
DFI_BANK_P1	2:0	DDRPHY Bank Address Phase1
DFI_BG_P1	2:0	DDRPHY Bank Group Phase1
DFI_CID_P1	1	DDRPHY Chip ID Phase1
DFI_CS_N_P1	1:0	DDRPHY Chip Select Phase1
DFI_CKE_P1	1:0	DDRPHY Clock Enable Phase1
DFI_ODT_P1	1:0	DDRPHY ODT Phase1
DFI_RESET_N	1	DDRPHY DFI Reset
TURN_ON_IE	1	DDRPHY Enable
TURN_OFF_ADRCTRL_DRIVERS	1	CIB Input control to tristate DDRPHY command address pad
AXB_SEL	1	AXB Bus Interface: Select
AXB_RESET_N	1	AXB Bus Interface: Reset
AXB_ADDR	8:0	AXB Bus Interface: Address
AXB_WRITE	1	AXB Bus Interface: Write Enable
AXB_WDATA	31:0	AXB Bus Interface: Write Data

Table 121: Input Ports (Continued)

Name	Range	Description
PHY_IN_DQ0	7:0	DQ Data Bus to PHY
PHY_IN_DQ1	7:0	DQ Data Bus to PHY
PHY_IN_DQ2	7:0	DQ Data Bus to PHY
PHY_IN_DQ3	7:0	DQ Data Bus to PHY
PHY_IN_DQ4	7:0	DQ Data Bus to PHY
PHY_IN_DQ5	7:0	DQ Data Bus to PHY
PHY_IN_DQ6	7:0	DQ Data Bus to PHY
PHY_IN_DQ7	7:0	DQ Data Bus to PHY
PHY_IN_DQ8	7:0	DQ Data Bus to PHY
PHY_IN_DM0	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM1	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM2	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM3	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM4	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM5	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM6	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM7	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DM8	1	DBI (Data Bus Inversion enable) to PHY
PHY_IN_DQS0	1	DQS data strobe to PHY
PHY_IN_DQS1	1	DQS data strobe to PHY
PHY_IN_DQS2	1	DQS data strobe to PHY
PHY_IN_DQS3	1	DQS data strobe to PHY
PHY_IN_DQS4	1	DQS data strobe to PHY
PHY_IN_DQS5	1	DQS data strobe to PHY
PHY_IN_DQS6	1	DQS data strobe to PHY
PHY_IN_DQS7	1	DQS data strobe to PHY
PHY_IN_DQS8	1	DQS data strobe to PHY
DFI_ADDRESS_P2	19:0	DDRPHY Phase2 Address
DFI_WRDATA_CS_P2	1:0	DDRPHY Write Data Chip Select Phase2
DFI_RDDATA_CS_P2	1:0	DDRPHY Read Data Chip Select Phase2
DFI_RAS_N_P2	1	DDRPHY RAS Enable Phase2
DFI_CAS_N_P2	1	DDRPHY CAS Enable Phase2

Table 121: Input Ports (Continued)

Name	Range	Description
DFI_ACT_N_P2	1	DDRPHY ACT Phase2
DFI_WE_N_P2	1	DDRPHY Write Enable Phase2
DFI_BANK_P2	2:0	DDRPHY Bank Address Phase2
DFI_BG_P2	2:0	DDRPHY Bank Group Phase2
DFI_CID_P2	1	DDRPHY Chip ID Phase2
DFI_CS_N_P2	1:0	DDRPHY Chip Select Phase2
DFI_CKE_P2	1:0	DDRPHY Clock Enable Phase2
DFI_ODT_P2	1:0	DDRPHY ODT Phase2
DFI_ADDRESS_P3	19:0	DDRPHY Phase3 Address
DFI_WRDATA_CS_P3	1:0	DDRPHY Write Data Chip Select Phase3
DFI_RDDATA_CS_P3	1:0	DDRPHY Read Data Chip Select Phase3
DFI_RAS_N_P3	1	DDRPHY RAS Enable Phase3
DFI_CAS_N_P3	1	DDRPHY CAS Enable Phase3
DFI_ACT_N_P3	1	DDRPHY ACT Phase3
DFI_WE_N_P3	1	DDRPHY Write Enable Phase3
DFI_BANK_P3	2:0	DDRPHY Bank Address Phase3
DFI_BG_P3	2:0	DDRPHY Bank Group Phase3
DFI_CID_P3	1	DDRPHY Chip ID Phase3
DFI_CS_N_P3	1:0	DDRPHY Chip Select Phase3
DFI_CKE_P3	1:0	DDRPHY Clock Enable Phase3
DFI_ODT_P3	1:0	DDRPHY ODT Phase3

Table 122: Output Ports

Name	Range	Description
DFI_RDDATA0	63:0	DDRPHY Read Data
DFI_RDDATA1	63:0	DDRPHY Read Data
DFI_RDDATA2	63:0	DDRPHY Read Data
DFI_RDDATA3	63:0	DDRPHY Read Data
DFI_RDDATA4	63:0	DDRPHY Read Data
DFI_RDDATA5	63:0	DDRPHY Read Data
DFI_RDDATA6	63:0	DDRPHY Read Data

Table 122: Output Ports (Continued)

Name	Range	Description
DFI_RDDATA7	63:0	DDRPHY Read Data
DFI_RDDATA8	63:0	DDRPHY Read Data
DFI_RDDATA_DBI0	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI1	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI2	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI3	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI4	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI5	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI6	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI7	7:0	DDRPHY Read Data DBI
DFI_RDDATA_DBI8	7:0	DDRPHY Read Data DBI
DFI_RDDATA_VALID0	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID1	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID2	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID3	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID4	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID5	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID6	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID7	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID8	1	DDRPHY Read Data Valid
DFI_RDDATA_VALID0_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID1_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID2_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID3_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID4_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID5_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID6_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID7_P2	1	DDRPHY Read Data Valid P2P3
DFI_RDDATA_VALID8_P2	1	DDRPHY Read Data Valid P2P3
DFI_PHYUPD_REQ	1	DDRPHY PHY Update Request
DFI_PHYUPD_TYPE	1:0	DDRPHY PHY Update Type
DFI_CTRLUPD_ACK	1	DDRPHY Control Update Acknowledge

Table 122: Output Ports (Continued)

Name	Range	Description
DFIIT_COMPLETE	1	DDRPHY Initialization Complete
AXB_READY	1	AXB bus interface: Read Enable
AXB_RDATA	31:0	AXB bus interface: Read Data
PHY_OUT_DQ0	7:0	DQ data bus from PHY
PHY_OUT_DQ1	7:0	DQ data bus from PHY
PHY_OUT_DQ2	7:0	DQ data bus from PHY
PHY_OUT_DQ3	7:0	DQ data bus from PHY
PHY_OUT_DQ4	7:0	DQ data bus from PHY
PHY_OUT_DQ5	7:0	DQ data bus from PHY
PHY_OUT_DQ6	7:0	DQ data bus from PHY
PHY_OUT_DQ7	7:0	DQ data bus from PHY
PHY_OUT_DQ8	7:0	DQ data bus from PHY
PHY_OUT_DQ_OEN0	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN1	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN2	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN3	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN4	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN5	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN6	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN7	7:0	Enable for DQ output from PHY
PHY_OUT_DQ_OEN8	7:0	Enable for DQ output from PHY
PHY_OUT_DM0	1	DM (Data Mask) from PHY
PHY_OUT_DM1	1	DM (Data Mask) from PHY
PHY_OUT_DM2	1	DM (Data Mask) from PHY
PHY_OUT_DM3	1	DM (Data Mask) from PHY
PHY_OUT_DM4	1	DM (Data Mask) from PHY
PHY_OUT_DM5	1	DM (Data Mask) from PHY
PHY_OUT_DM6	1	DM (Data Mask) from PHY
PHY_OUT_DM7	1	DM (Data Mask) from PHY
PHY_OUT_DM8	1	DM (Data Mask) from PHY
PHY_OUT_DM_OEN0	1	Enable for DM output from PHY
PHY_OUT_DM_OEN1	1	Enable for DM output from PHY

Table 122: Output Ports (Continued)

Name	Range	Description
PHY_OUT_DM_OEN2	1	Enable for DM output from PHY
PHY_OUT_DM_OEN3	1	Enable for DM output from PHY
PHY_OUT_DM_OEN4	1	Enable for DM output from PHY
PHY_OUT_DM_OEN5	1	Enable for DM output from PHY
PHY_OUT_DM_OEN6	1	Enable for DM output from PHY
PHY_OUT_DM_OEN7	1	Enable for DM output from PHY
PHY_OUT_DM_OEN8	1	Enable for DM output from PHY
PHY_OUT_DQS0	1	DQS data strobe from PHY
PHY_OUT_DQS1	1	DQS data strobe from PHY
PHY_OUT_DQS2	1	DQS data strobe from PHY
PHY_OUT_DQS3	1	DQS data strobe from PHY
PHY_OUT_DQS4	1	DQS data strobe from PHY
PHY_OUT_DQS5	1	DQS data strobe from PHY
PHY_OUT_DQS6	1	DQS data strobe from PHY
PHY_OUT_DQS7	1	DQS data strobe from PHY
PHY_OUT_DQS8	1	DQS data strobe from PHY
PHY_OUT_DQS_OEN0	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN1	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN2	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN3	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN4	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN5	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN6	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN7	1	Enable for DQS output from PHY
PHY_OUT_DQS_OEN8	1	Enable for DQS output from PHY
DQ_READ0	7:0	FPGA IO ODT Enable For Data
DQ_READ1	7:0	FPGA IO ODT Enable For Data
DQ_READ2	7:0	FPGA IO ODT Enable For Data
DQ_READ3	7:0	FPGA IO ODT Enable For Data
DQ_READ4	7:0	FPGA IO ODT Enable For Data
DQ_READ5	7:0	FPGA IO ODT Enable For Data
DQ_READ6	7:0	FPGA IO ODT Enable For Data

Table 122: Output Ports (Continued)

Name	Range	Description
DQ_READ7	7:0	FPGA IO ODT Enable For Data
DQ_READ8	7:0	FPGA IO ODT Enable For Data
DM_READ0	1	FPGA IO ODT enable for data mask
DM_READ1	1	FPGA IO ODT enable for data mask
DM_READ2	1	FPGA IO ODT enable for data mask
DM_READ3	1	FPGA IO ODT enable for data mask
DM_READ4	1	FPGA IO ODT enable for data mask
DM_READ5	1	FPGA IO ODT enable for data mask
DM_READ6	1	FPGA IO ODT enable for data mask
DM_READ7	1	FPGA IO ODT enable for data mask
DM_READ8	1	FPGA IO ODT enable for data mask
DQS_READ0	1	FPGA IO ODT enable for data strobe
DQS_READ1	1	FPGA IO ODT enable for data strobe
DQS_READ2	1	FPGA IO ODT enable for data strobe
DQS_READ3	1	FPGA IO ODT enable for data strobe
DQS_READ4	1	FPGA IO ODT enable for data strobe
DQS_READ5	1	FPGA IO ODT enable for data strobe
DQS_READ6	1	FPGA IO ODT enable for data strobe
DQS_READ7	1	FPGA IO ODT enable for data strobe
DQS_READ8	1	FPGA IO ODT enable for data strobe
DQ_IE0	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE1	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE2	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE3	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE4	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE5	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE6	7:0	Optional dynamic input enable for DQ IO receivers during reads

Table 122: Output Ports (Continued)

Name	Range	Description
DQ_IE7	7:0	Optional dynamic input enable for DQ IO receivers during reads
DQ_IE8	7:0	Optional dynamic input enable for DQ IO receivers during reads
DM_IE0	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE1	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE2	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE3	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE4	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE5	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE6	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE7	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DM_IE8	1	Optional dynamic input enable for DM/DBI IO receivers during reads (00,01,10 controlled by PHY/CTRLR)
DQS_IE0	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE1	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE2	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE3	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE4	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE5	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE6	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE7	1	Optional dynamic input enable for DQS IO receivers during reads
DQS_IE8	1	Optional dynamic input enable for DQS IO receivers during reads

Table 122: Output Ports (Continued)

Name	Range	Description
VREF_OUTPUT_ENABLE0	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE1	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE2	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE3	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE4	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE5	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE6	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE7	1	Global control input to enable/disable VREF generation if required
VREF_OUTPUT_ENABLE8	1	Global control input to enable/disable VREF generation if required
VREF_VALUE0	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE1	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE2	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE3	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE4	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE5	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE6	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE7	6:0	Vref value set for each byte lane - width can be customized
VREF_VALUE8	6:0	Vref value set for each byte lane - width can be customized
CA_OUT_CLK0	1	DRAM clock output from PHY
CA_OUT_CLK1	1	DRAM clock output from PHY
CA_OUT_RST_N	1	DRAM reset output from PHY
CA_OUT_A0	1	DRAM MA output from PHY

Table 122: Output Ports (Continued)

Name	Range	Description
CA_OUT_A1	1	DRAM MA output from PHY
CA_OUT_A2	1	DRAM MA output from PHY
CA_OUT_A3	1	DRAM MA output from PHY
CA_OUT_A4	1	DRAM MA output from PHY
CA_OUT_A5	1	DRAM MA output from PHY
CA_OUT_A6	1	DRAM MA output from PHY
CA_OUT_A7	1	DRAM MA output from PHY
CA_OUT_A8	1	DRAM MA output from PHY
CA_OUT_A9	1	DRAM MA output from PHY
CA_OUT_A10	1	DRAM MA output from PHY
CA_OUT_A11	1	DRAM MA output from PHY
CA_OUT_A12	1	DRAM MA output from PHY
CA_OUT_A13	1	DRAM MA output from PHY
CA_OUT_A14	1	DRAM MA output from PHY
CA_OUT_A15	1	DRAM MA output from PHY
CA_OUT_A16	1	DRAM MA output from PHY
CA_OUT_CS_N	1:0	DRAM CS_N output from PHY
CA_OUT_CKE	1:0	DRAM CKE output from PHY
CA_OUT_BA2	1	DRAM BA output from PHY
CA_OUT_BA1	1	DRAM BA output from PHY
CA_OUT_BA0	1	DRAM BA output from PHY
CA_OUT_RAS_N	1	DRAM RAS_N output from PHY
CA_OUT_CAS_N	1	DRAM CAS_N output from PHY
CA_OUT_WE_N	1	DRAM WE_N output from PHY
CA_OUT_ODT	1:0	DRAM ODT output from PHY
CA_OUT_CS_OEN	1	DDR5 mode register bit
CA_OUT_CLK_OEN0	1	Disable MCLK output
CA_OUT_CLK_OEN1	1	Disable MCLK output
VREF_CA_VALUE	6:0	VREF value driven for CA IOs
UPDATE	1	Pulsed to indicate that the last TX & RX calibrated trim value may be driven on the pad-ring
ADDR_CTRL_LOPHY_OUTBACK_RCVR_EN	1	Used to enable loopback test on Address/Control/Clock (CA) IOs

Table 122: Output Ports (Continued)

Name	Range	Description
ADDR_CTRL_VREF_OUTPUT_ENABLE	1	VREF output enable
ADRCTRL_DRIVERS_OEN	1	DDRPHY IP tristate output to PIC DDRPHY command address tristate control DDRPHY_TO
RECEIVER_EN0	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN1	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN2	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN3	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN4	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN5	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN6	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN7	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled
RECEIVER_EN8	1	IO will connect to stdby_inr_cib to support memory & non-memory; Bank based & User Controlled

Table 123: Parameters

Name	Values	Type
RATE	"HALF" (default) "FULL"	DDR rate
PROTOCOL	"DDR3L" (default) "DDR4"	DDR protocol
EXT_RES	"PER_BANK" (default) "PER_DDRPHY"	External Resistor for PVT Cal

DELAYA

Dynamic Input/Output Delay Element.

Architectures supported:

- LFCPNX

- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

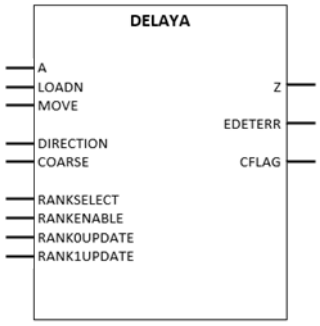


Table 124: Input Ports

Name	Description
A	Data input from pin or output register block.
LOADN	0 resets to default delay setting.
MOVE	A pulse changes delay setting. The DIRECTION pin is sampled at falling edge of MOVE.
DIRECTION	1 to decrease delay and 0 to increase delay.
COARSE[1:0]	Dynamic coarse delay control: 00: No coarse delay 01: 800 ps delay 10: 1600 ps delay 11: Invalid
RANKSELECT	0 - Select delay rank 0 1 - Select delay rank 1
RANKENABLE	0 - Select bypassed actual path delay code 1 - Select registered actual path delay code
RANK0UPDATE	0 - Rank-0 registers not enabled 1 - Enable Rank-0 registers
RANK1UPDATE	0 - Rank-1 registers not enabled 1 - Enable Rank-1 registers

Table 125: Output Ports

Name	Description
Z	Delayed data to input register block or to pin.
EDETERR	Error detected when using the edge monitor logic.
CFLAG	Flag indicating the delay counter has reached the max (when moving up) or min (when moving down) value.

Table 126: Parameters

Attribute	Values	Description
DEL_MODE	"USER_DEFINED" *(default)* "SCLK_ZEROHOLD" "ECLK_ALIGNED" "ECLK_CENTERED" "SCLK_ALIGNED" "SCLK_CENTERED" "DQS_CMD_CLK" "DQS_ALIGNED_X2" "DQS_ALIGNED_X4" "DQS_CENTERED_X2" "DQS_CENTERED_X4"	Sets the delay mode. DQS_CMD_CLK is only for the DDR Memory CMD and CLK outputs. DQS_ALIGNED_X2/X4 is shared by DQS generic and the DDR memory inputs. DQS_CENTERED_X2/X4 is used for DQS generic inputs. When DEL_MODE is not “USER_DEFINED,” DEL_VALUE and COARSE_DELAY are set to a value based on DEL_MODE and device performance. MAP will use fixed values for these components during the MAP.
DEL_VALUE	“0” *(default)*	Sets delay value when DEL_MODE is set to USER_DEFINED. Step is 12.5 ps.
COARSE_DELAY_MODE	"STATIC" *(default)* "DYNAMIC"	Selects coarse delay code control: 0 - Selects from MC1 (STATIC) 1 - Selects from CIB (DYNAMIC)
COARSE_DELAY	"0NS" *(default)* "0P8NS" "1P6NS"	Coarse delay setting for lol delay cell.
EDGE_MONITOR	"DISABLED" *(default)* "ENABLED"	Enables edge monitor when in IDDR X2, X7to1, X4, or X5 mode.
WAIT_FOR_EDGE	"DISABLED" *(default)* "ENABLED"	Used for SPI4.2 implementation.
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset.

This primitive can be used to delay the input data from the input pin to the IDDR, IREG, or FPGA OR to delay the output data from the ODDR, OREG, or FPGA fabric to the output pin. DELAYA is useful to adjust for any skews

amongst the input or output data buses. DELAYA can also be used to generate skew between the bits of an output bus to reduce SSO noise.

DELAYA can be used with IDDR, ODDR, or SDR modules, as well as on the direct input to the FPGA. DELAYA is shared by the input and output paths, and so it can only be used to delay the input data or the output data on a given pin.

The data input to this primitive can be delayed using:

- Pre-determined delay value (for Zero Hold time, delay based on Interface Type)
- Fixed Delay values entered by user

The delay can also be dynamically updated using counter up and down controls.

You can optionally bypass DELAYA completely as well.

By default, DELAYA is configured to factory delay settings based on the clocking structure. You can overwrite the DELAYA setting using the MOVE and DIRECTION control inputs. The LOADN pin will reset the delay back to the default value.

DELAYB

Static Input/Output Delay Element.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

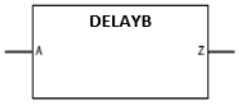


Table 127: Input Ports

Name	Description
A	Data input from pin or output register block.

Table 128: Output Ports

Name	Description
Z	Delayed data to input register block or to pin.

Table 129: Parameters

Attribute	Values	Description
COARSE_DELAY	"0NS" *(default)* "0P8NS" "1P6NS"	Coarse delay setting for lol delay cell.
DEL_VALUE	“0” *(default)*	Sets delay value when DEL_MODE is set to USER_DEFINED. Step is 12.5 ps.
DEL_MODE	"USER_DEFINED" *(default)* "SCLK_ZEROHOLD" "ECLK_ALIGNED" "ECLK_CENTERED" "SCLK_ALIGNED" "SCLK_CENTERED" "DQS_CMD_CLK" "DQS_ALIGNED_X2" "DQS_ALIGNED_X4" "DQS_CENTERED_X2" "DQS_CENTERED_X4"	Sets the delay mode. DQS_CMD_CLK is only for the DDR Memory CMD and CLK outputs. DQS_ALIGNED_X2/X4 is shared by DQS generic and the DDR memory inputs. DQS_CENTERED_X2/X4 is used for DQS generic inputs. When DEL_MODE is not “USER_DEFINED,” DEL_VALUE and COARSE_DELAY are set to a value based on DEL_MODE and device performance. MAP will use fixed values for these components during the MAP.
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset.

This primitive can be used to delay the input data from the input pin to the IDDR, IREG, or FPGA OR to delay the output data from the ODDR, OREG, or FPGA fabric to the output pin. DELAYB is useful to adjust for any skews amongst the input or output data buses. DELAYB can also be used to generate skew between the bits of an output bus to reduce SSO noise.

DELAYB can be used with IDDR, ODDR, or SDR modules, as well as on the direct input to the FPGA. DELAYB is shared by the input and output paths, and so it can only be used to delay the input data or the output data on a given pin.

The data input to this primitive can be delayed using:

- Predetermined delay value (for Zero Hold time, delay based on Interface Type)
- Fixed Delay values entered by user

The delay can also be dynamically updated using counter up and down controls.

You can optionally bypass DELAYB completely as well.

By default, DELAYB is configured to factory delay settings based on the clocking structure. You cannot change the delay when using this module.

DELAYC

Supports edge monitor feature

Architectures supported:

- LFD2NX
- LFMXO5
- LIFCL
- UT24C

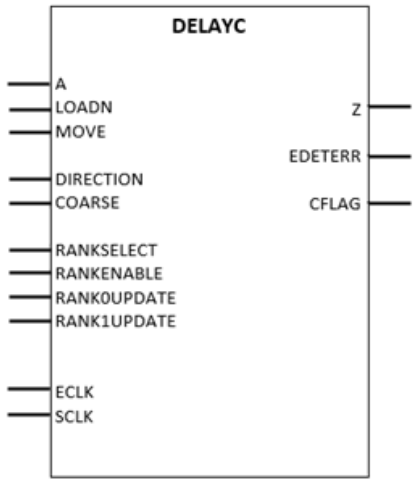


Table 130: Input Ports

Name	Description
A	Data input from pin or output register block '0' on LOADN will reset to default delay setting
LOAD_N	'0' on LOADN will reset to default delay setting
MOVE	"Pulse" on MOVE will change delay setting. DIRECTION will be sampled at falling edge of MOVE.
DIRECTION	'1' to decrease delay and '0' to increase delay

Table 130: Input Ports (Continued)

Name	Description
COARSE[1:0]	Dynamic coarse delay control (2 bits) 00: no coarse delay 01: 800 ps delay 10: 1600 ps delay 11: invalid
RANKSELECT	Select which rank 0 - Select delay rank 0 1 - Select delay rank 1
RANKENABLE	Enable rank register 0 - Select bypassed actual path delay code 1 - Select registered actual path delay code
RANK0UPADTE	Enable RANK0 register 0 - Rank-0 registers not enabled 1 - Enable Rank-0 registers
RANK1UPADTE	Enable RANK1 register 0 - Rank-1 registers not enabled 1 - Enable Rank-1 registers
ECLK	Fast edge clock
SCLK	Primary clock input (divide-by-2 of ECLK)

Table 131: Output Ports

Name	Description
Z	Delayed data to input register block or to pin
EDETERR	Error detected when using the edge monitor logic
CFLAG	Flag indicating the delay counter has reached the max (when moving up) or min (when moving down) value

Table 132: Parameters

Attribute	Values	Description
DEL_MODE	"USER_DEFINED" *(default)* "SCLK_ZEROHOLD" "ECLK_ALIGNED" "ECLK_CENTERED" "SCLK_ALIGNED" "SCLK_CENTERED" "DQS_CMD_CLK" "DQS_ALIGNED_X2" "DQS_ALIGNED_X4" "DQS_CENTERED_X2" "DQS_CENTERED_X4"	Sets the delay mode. DQS_CMD_CLK is only for the DDR Memory CMD and CLK outputs. DQS_ALIGNED_X2/X4 is shared by DQS generic and the DDR memory inputs. DQS_CENTERED_X2/X4 is used for DQS generic inputs. When DEL_MODE is not “USER_DEFINED,” DEL_VALUE and COARSE_DELAY are set to a value based on DEL_MODE and device performance. MAP will use fixed values for these components during the MAP.
DEL_VALUE	“0” *(default)*	Sets delay value when DEL_MODE is set to USER_DEFINED. Step is 12.5 ps.
COARSE_DELAY_MODE	"STATIC" *(default)* "DYNAMIC"	Selects coarse delay code control: 0 - Selects from MC1 (STATIC) 1 - Selects from CIB (DYNAMIC)
COARSE_DELAY	"0NS" *(default)* "0P8NS" "1P6NS"	Coarse delay setting for Iol delay cell.
EDGE_MONITOR	"DISABLED" *(default)* "ENABLED"	Enables edge monitor when in IDDR X2, X7to1, X4, or X5 mode.
WAIT_FOR_EDGE	"DISABLED" *(default)* "ENABLED"	Used for SPI4.2 implementation.
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset.

DELAYD

Input or output delay for SCLK interface with dynamically control delay.

Architectures supported:

- LAV-AT-ES1
- LN2-CT

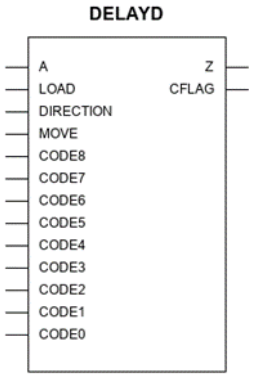


Table 133: Input Ports

Name	Range	Type	Active State	Description
A	1	Data	High	Data input from pin
LOAD	1	Control	High	LOAD is set to 1 to reset to delay setting (DEL_VALUE)
MOVE	1	Control	High	MOVE will change delay setting. DIR will be sampled at "falling edge" of MOVE
DIRECTION	1	Control	High	DIRECTION 1 to decrease delay 0 to increase delay
CODE8	1	Data	High	Delay Code from DDRDLL
CODE7	1	Data	High	Delay Code from DDRDLL
CODE6	1	Data	High	Delay Code from DDRDLL
CODE5	1	Data	High	Delay Code from DDRDLL
CODE4	1	Data	High	Delay Code from DDRDLL
CODE3	1	Data	High	Delay Code from DDRDLL
CODE2	1	Data	High	Delay Code from DDRDLL
CODE1	1	Data	High	Delay Code from DDRDLL
CODE0	1	Data	High	Delay Code from DDRDLL

Table 134: Output Ports

Name	Range	Type	Active State	Description
Z	1	Flag	High	Delay Output
CFLAG	1	Flag	High	Flag indicating the delay counter has reached max (when moving up) or min (when moving down) value

Table 135: Parameters

Name	Values	Description
DEL_MODE	USER_DEFINED SCLK_ZEROHOLD SCLK_ALIGNED SCLK_CENTERED	Set delay mode
DEL_VALUE	"0" to "511"	Static Delay Value
DEL_SOURCE	"DEL_VALUE" "DDRDLL"	Static Delay Value Source

DELAYE

Input delay or output delay for SCLK interface with static delay.

Architectures supported:

- LAV-AT-ES1
- LN2-CT

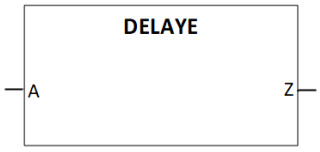


Table 136: Input Ports

Name	Range	Type	Active State	Description
A	1	Data	High	Data input from pin

Table 137: Output Ports

Name	Range	Type	Active State	Description
Z	1	Flag	High	Delay Output

Table 138: Parameters

Name	Values	Description
DEL_MODE	USER_DEFINED SCLK_ZEROHOLD SCLK_ALIGNED SCLK_CENTERED	Set delay mode
DEL_VALUE	"0" to "511"	"0"

DELAYF

Input or output delay for ECLK interface with dynamically control delay

Architectures supported:

- LAV-AT-ES1
- LN2-CT

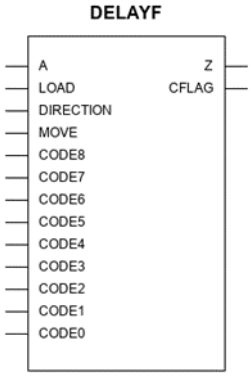


Table 139: Input Ports

Name	Range	Type	Active State	Description
A	1	Data	High	Data input from pin
LOAD	1	Control	High	LOAD is set to 0 to reset to delay setting (DEL_VALUE)
MOVE	1	Control	High	MOVE will change delay setting. DIR will be sampled at "falling edge" of MOVE
DIRECTION	1	Control	High	DIRECTION 1 to decrease delay 0 to increase delay
CODE8	1	Data	High	Delay Code from DDRDLL
CODE7	1	Data	High	Delay Code from DDRDLL
CODE6	1	Data	High	Delay Code from DDRDLL
CODE5	1	Data	High	Delay Code from DDRDLL
CODE4	1	Data	High	Delay Code from DDRDLL
CODE3	1	Data	High	Delay Code from DDRDLL
CODE2	1	Data	High	Delay Code from DDRDLL
CODE1	1	Data	High	Delay Code from DDRDLL
CODE0	1	Data	High	Delay Code from DDRDLL

Table 140: Output Ports

Name	Range	Type	Action State	Description
Z	1	Data	High	Delay Output
CFLAG	1	Flag	High	Flag indicating the delay counter has reached max (when moving up) or min (when moving down) value

Table 141: Parameters

Name	Value	Description
DEL_MODE	USER_DEFINED ECLK_ALIGNED ECLK_CENTERED	Set delay mode
DEL_VALUE	"0" to "511"	Static Delay Value
DEL_SOURCE	"DEL_VALUE" "DDRDLL"	Static Delay Value Source

DIFFCLKIO

Differential to single-end clock buffer.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP



Table 142: Input Ports

Name	Description
CLKIN0_P	1st diff reference clock input
CLKIN0_N	1st diff reference clock input
CLKIN1_P	2nd diff reference clock input
CLKIN1_N	2nd diff reference clock input

Table 143: Output Ports

Name	Description
CLKOUT0	1st single end clock output
CLKOUT1	2nd single end clock output

Table 144: Parameters

Name	Values	Description
TERM_RD	"ENABLED" (default) "DISABLED"	Differential termination resistor control
WEAK_BIAS	"DISABLED" (default) "ENABLED"	Internal weak bias enable control

DIFFIO18

Single Ended + Differential I/O block for 1.8V block

Architectures supported:

- LFD2NX
- LFMXO5D
- LFMXO5-25
- LIFCL
- UT24C

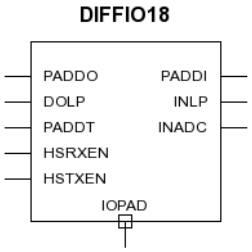


Table 145: Input Ports

Name	Description
PADDO	Data output from fabric to IO pad
DOLP	Data output for MIPI low-power mode
PADDT	Data tri-state control; high = tri-state
HSRXEN	RX enable for MIPI high-speed mode
HSTXEN	TX enable for MIPI high-speed mode

Table 146: Output Ports

Name	Description
PADDI	Data input from IO pad to fabric
INLP	Data input for MIPI low-power mode
INADC	Output to ADC

Table 147: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

Table 148: Parameters

Name	Values	Description
PULLMODE	"DOWN" (default) "NONE" "UP" "KEEPER" "FAILSAFE"	Enables pullup and pulldown mode
ENADC_IN	"DISABLED" (default) "ENABLED"	Enables pad A path to adc block
MIPI	"DISABLED" (default) "ENABLED"	Enable MIPI_DPHY IO TYPE

DIFFIO18A

Single Ended + Differential I/O block for 1.8V block

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

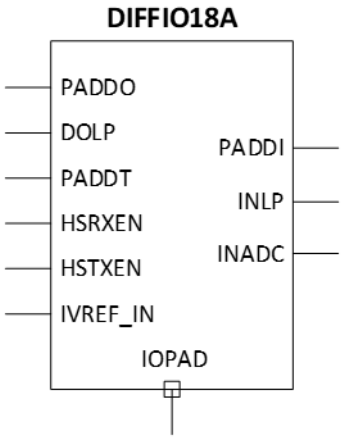


Table 149: Input Ports

Name	Description
PADDO	HS Data out to PAD
DOLP	LP Data out to PAD
PADDT	Tri-state input signals to IO A, 0: driver output enabled 1: diver enabled to tri-state(high-Z) output
HSRXEN	Enable signal for HSRX termination ibuf_rd:0 disable; 1 enable
HSTXEN	Enable signal for HSTX termination
IVREF_IN	Internal Receiver Reference Voltage Input. Each bank will support 3 VREF analog input voltages for flexibility to receive three different referenced input levels on the same bank. VREFs are used to set the threshold for the referenced input receiver. Each referenced receiver has a programmable option to select VREF1/2 and a internal VREF.

Table 150: Output Ports

Name	Description
PADDI	Data in
INLP	MIPI LP RX output
INADC	Output to ADC through switch from PAD

Table 151: Bidirectional Ports

Name	Description
IOPAD	A virtual port, which is connected to the external pad.

Table 152: Parameters

Name	Values	Description
PULLMODE	"DOWN" (default) "NONE" "UP"	Pull mode
ENADC_IN	"DISABLED" (default) "ENABLED"	Enables pad A path to adc block
MIPI	"DISABLED" (default) "ENABLED"	MIPI usage

DLLDEL

Slave delay elements are used to support generic DDR RX modes to shift input clock phase by 90 degrees.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

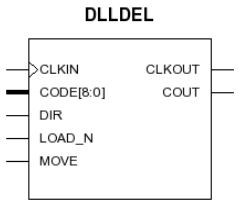


Table 153: Input Ports

Name	Description
CLKIN	Input clock
CODE[8:0]	Code control
DIR	Direction

Table 153: Input Ports (Continued)

Name	Description
LOAD_N	'0' on LOADN will reset to default delay setting
MOVE	"Pulse" on MOVE will change delay setting. DIRECTION will be sampled at falling edge of MOVE

Table 154: Output Ports

Name	Description
CLKOUT	Output clock to fabric
COUT	Carry out

Table 155: Parameters

Name	Values	Description
ADJUST	0 (default)	Offset for slave delay adjustment. mc1_adjust[9] is the sign bit. Examples: mc1_adjust[9:0] = 0_000000000: adds 0 units of delay mc1_adjust[9:0] = 0_000000010: adds 2 units of delay mc1_adjust[9:0] = 0_111111111: adds 511 units of delay
DEL_ADJUST	"PLUS" (default) "MINUS"	Sign bit for mc1_adjust, bit [9]. The DEL_ADJUST and ADJUST are set to a value based on device performance. MAP will use fixed values for these components during the MAP.
ENABLE	"ENABLED" (default) "DISABLED"	Enables the block and ties off the clk output when not using this block.

DLLDELA

DDR RX modes to shift input clock phase by 90 degrees

Architectures supported:

- LAV-AT-ES1
- LN2-CT

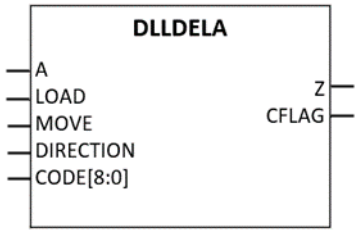


Table 156: Input Ports

Name	Range	Type	Active State	Description
A	1	Data	High	Data input from pin
LOAD	1	Control	High	LOAD is set to 1 to reset to default delay setting
MOVE	1	Control	High	MOVE will change delay setting. DIR will be sampled at "rising edge" of MOVE
DIRECTION	1	Control	High	DIRECTION 1 to decrease delay 0 to increase delay
CODE	9	Control	High	Delay code from DDRDLL

Table 157: Output Ports

Name	Range	Type	Active State	Description
Z	1	Data	High	Delayed data to input register block
CFLAG	1	Flag	High	Flag indicating the delay counter has reached max (when moving up) or min (when moving down) value

Table 158: Parameters

Name	Values	Type
DEL_ADJ	"PLUS" "MINUS"	DLL control code forcing. "DLL_LOOP": Code/lock generated from DLL loop "FORCE_MAX": Force lock and code is set to maximum of 511 for lower input frequency mode

Table 158: Parameters (Continued)

Name	Values	Type
DEL_VALUE	"0" to "511"	Offset for slave delay adjustment
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

DOTPRODADDSUB9X9

Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor

Architectures supported:

- LAV-AT-ES1
- LN2-CT

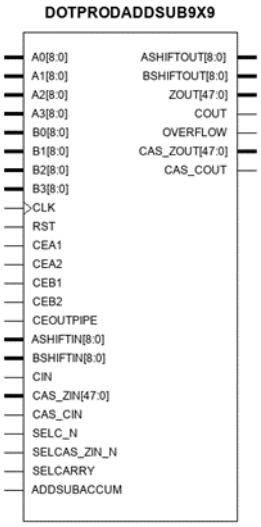


Table 159: Input Ports

Name	Range	Type	Active State	Description
A0	8:0	Data	Active high	Operand A0
B0	8:0	Data	Active high	Operand B0
A1	8:0	Data	Active high	Operand A1
B1	8:0	Data	Active high	Operand B1
A2	8:0	Data	Active high	Operand A2
B2	8:0	Data	Active high	Operand B2
A3	8:0	Data	Active high	Operand A3

Table 159: Input Ports (Continued)

Name	Range	Type	Active State	Description
B3	8:0	Data	Active high	Operand B3
CLK	1	Clock	Active high	Clock
RST	1	Reset	Active high	Synchronous reset for all registers
CEA1	1	Clock Enable	Active high	Input A register clock enable, stage one
CEA2	1	Clock Enable	Active high	Input A register clock enable, stage two
CEB1	1	Clock Enable	Active high	Input B register clock enable, stage one
CEB2	1	Clock Enable	Active high	Input B register clock enable, stage two
CEOUTPIPE	1	Clock Enable	Active high	Pipeline and output register clock enable
ASHIFTIN	8:0	Data	Active high	Operand A shift chain input
BSHIFTIN	8:0	Data	Active high	Operand B shift chain input
CIN	1	Control	Active high	Carry in
CAS_ZIN	47:0	Control	Active low	Cascade input
CAS_CIN	1	Control	Active low	Cascade carry in
SELC_N	1	Control	Active high	Select between C or accumulator output for input to the accumulator
SELCAS_ZIN_N	1	Control	Active low	Select between CAS_ZIN or accumulator output for input to the accumulator
SELCARRY	1	Control	Active high	Select between CIN or accumulator output Z[47] for input to the accumulator
ADDSUBACCUM	1	Control	Active high	Add/sub control for accumulator, 0=add, 1=subtract

Table 160: Output Ports

Name	Range	Type	Active State	Description
ASHIFTOUT	8:0	Control	Active high	Operand A shift chain output
BSHIFTOUT	8:0	Control	Active high	Operand B shift chain output
ZOUT	47:0	Control	Active high	Multiplication/accumulator output result
COUT	1	Control	Active high	Carry out
OVERFLOW	1	Flag	Active high	Overflow flag
CAS_ZOUT	47:0	Control	Active low	Cascade multiplication/accumulator output result
CAS_COUT	1	Control	Active low	Cascade carry out

Table 161: Parameters

Name	Values	Description
REGINPUTA0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A0
REGINPUTB0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B0
REGSHIFTOUTA0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A0
REGSHIFTOUTB0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B0
SHIFTINPUTA0	"DISABLED" (default) "ENABLED"	Shift input A0
SHIFTINPUTB0	"DISABLED" (default) "ENABLED"	Shift input B0
REGINPUTA1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A1
REGINPUTB1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B1
REGSHIFTOUTA1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A1
REGSHIFTOUTB1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B1
SHIFTINPUTA1	"DISABLED" (default) "ENABLED"	Shift input A1
SHIFTINPUTB1	"DISABLED" (default) "ENABLED"	Shift input B1

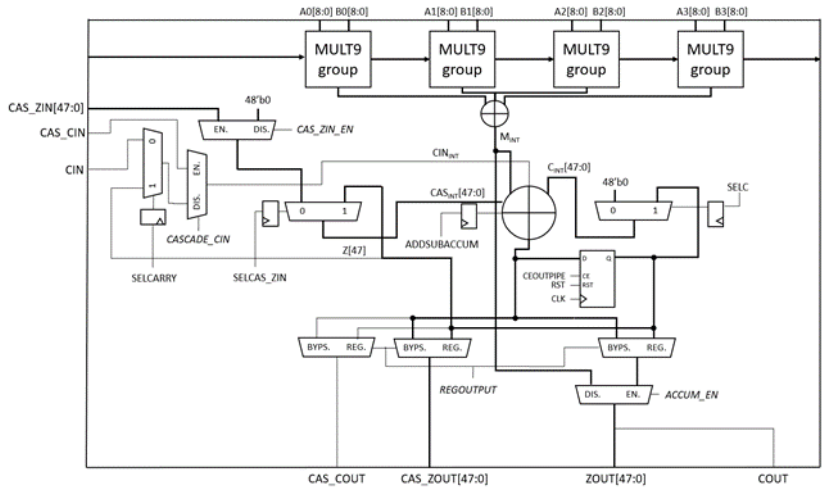
Table 161: Parameters (Continued)

Name	Values	Description
REGINPUTA2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A2
REGINPUTB2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B2
REGSHIFTOUTA2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A2
REGSHIFTOUTB2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B2
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
SHIFTINPUTA2	"DISABLED" (default) "ENABLED"	Shift input A2
SHIFTINPUTB2	"DISABLED" (default) "ENABLED"	Shift input B2
REGINPUTA3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A3
REGINPUTB3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B3
REGSHIFTOUTA3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A3
REGSHIFTOUTB3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B3
SHIFTINPUTA3	"DISABLED" (default) "ENABLED"	Shift input A3
SHIFTINPUTB3	"DISABLED" (default) "ENABLED"	Shift input B3

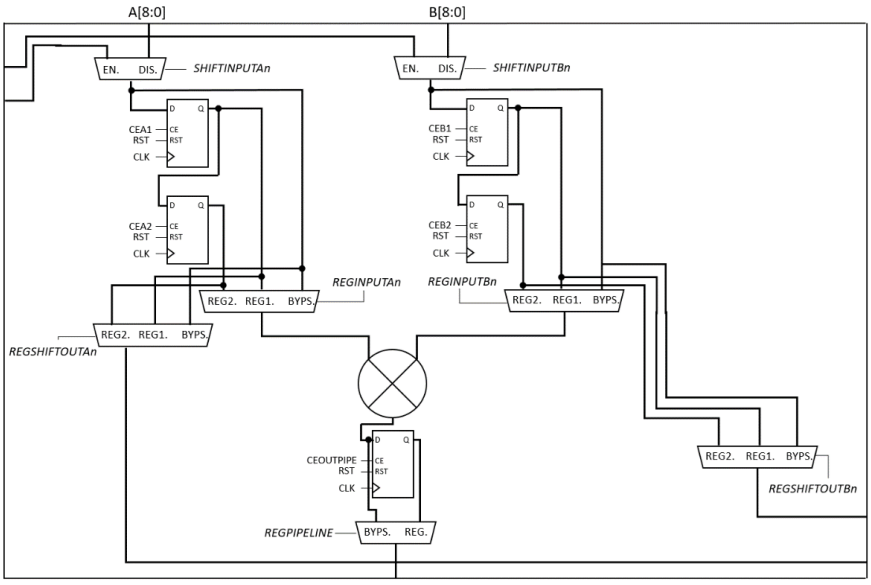
Table 161: Parameters (Continued)

Name	Values	Description
REGPIPE	"REGISTERED" (default) "BYPASSED"	Register pipeline
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
REGCAS_ZOUT	"REGISTERED" (default) "BYPASSED"	Register CAS_COUT and CAS_ZOUT[47:0]
REGACCUMCONTROLS	"REGISTERED" (default) "BYPASSED"	Register accumulator controls (CIN, SELCARRY, SELCAS_ZIN, ADDSUBACCUM, SELC)
ACCUM_EN	"DISABLED" (default) "ENABLED"	Enable accumulator
CAS_ZIN_EN	"DISABLED" (default) "ENABLED"	Enable dynamic CAS_ZIN for accumulator input (1) or tie to constant 0 (0)
CAS_CIN_EN	"DISABLED" (default) "ENABLED"	Use CAS_CIN for carry input
ROUNDMODE	"ROUND_TO_ZERO" (default) "ROUND_TO_INFINITY"	Rounding mode
SATURATION	"DISABLED" (default) "TO_NEG_MAX" "TO_ZERO"	Saturation enable
SATURATION_BITS	"48" (default) "3-48"	Saturation bits
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:



The function of "MULT9 group" in the above diagram is shown below.



When the accumulator is enabled, it implements the function:

M_INT = (A_0 * B_0 + (A_1 * B_1) + (A_2 * B_2) + (A_3 * B_3)

Z_out = CAS_INT + C_INT + CIN_INT +/- M_INT

When the accumulator is disabled, it implements the function:

Z_OUT = (A_0 * B_0) + (A_1 * B_1) + (A_2 * B_2) + (A_3 * B_3)

See the first diagram for the internal node corresponding to the above terms.

Although not shown in the above diagram, the register for accumulator inputs CIN, CAS_ZIN_EN, SELCAS_ZIN, and ADDSUBACCUM is bypassable with the parameter REGACCUMCONTROLS.

DOTPRODADDSUB9X9A

Four 9x9 Multiplier with Optional Input/output registers and adder/subtractor

Architectures supported:

- LAV-AT-ES1
- LN2-CT

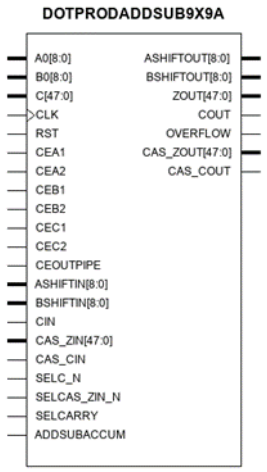


Table 162: Input Ports

Name	Range	Type	Active State	Description
A0	8:0	Data	Active high	Operand A0
B0	8:0	Data	Active high	Operand B0
CLK	1	Clock	Active high	Clock
RST	1	Reset	Active high	Synchronous reset for all registers
CEA1	1	Clock Enable	Active high	Input A register clock enable, stage one
CEA2	1	Clock Enable	Active high	Input A register clock enable, stage two
CEB1	1	Clock Enable	Active high	Input B register clock enable, stage one
CEB2	1	Clock Enable	Active high	Input B register clock enable, stage two
CEOUTPIPE	1	Clock Enable	Active high	Pipeline and output register clock enable
C	47:0	Control	Active high	Operand C

Table 162: Input Ports (Continued)

Name	Range	Type	Active State	Description
CEC1	1	Clock Enable	Active high	Input C register clock enable, stage one
CEC2	1	Clock Enable	Active high	Input C register clock enable, stage two
ASHIFTIN	8:0	Data	Active high	Operand A shift chain input
BSHIFTIN	8:0	Data	Active high	Operand B shift chain input
CIN	1	Control	Active high	Carry in
CAS_ZIN	47:0	Control	Active low	Cascade input
CAS_CIN	1	Control	Active low	Cascade carry in
SELC_N	1	Control	Active high	Select between C or accumulator output for input to the accumulator
SELCAS_ZIN_N	1	Control	Active low	Select between CAS_ZIN or accumulator output for input to the accumulator
SELCARRY	1	Control	Active high	Select between CIN or accumulator output Z[47] for input to the accumulator
ADDSUBACCUM	1	Control	Active high	Add/sub control for accumulator, 0=add, 1=subtract

Table 163: Output Ports

Name	Range	Type	Active State	Description
ASHIFTOUT	8:0	Control	Active high	Operand A shift chain output
BSHIFTOUT	8:0	Control	Active high	Operand B shift chain output
ZOUT	47:0	Control	Active high	Multiplication/accumulator output result
COUT	1	Control	Active high	Carry out
OVERFLOW	1	Flag	Active high	Overflow flag
CAS_ZOUT	47:0	Control	Active low	Cascade multiplication/accumulator output result
CAS_COUT	1	Control	Active low	Cascade carry out

Table 164: Parameters

Name	Values	Description
REGINPUTA0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A0
REGINPUTB0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B0
REGSHIFTOUTA0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A0
REGSHIFTOUTB0	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B0
SHIFTINPUTA0	"DISABLED" (default) "ENABLED"	Shift input A0
SHIFTINPUTB0	"DISABLED" (default) "ENABLED"	Shift input B0
REGINPUTA1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A1
REGINPUTB1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B1
REGSHIFTOUTA1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A1
REGSHIFTOUTB1	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B1
REGINPUTA2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A2
REGINPUTB2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B2

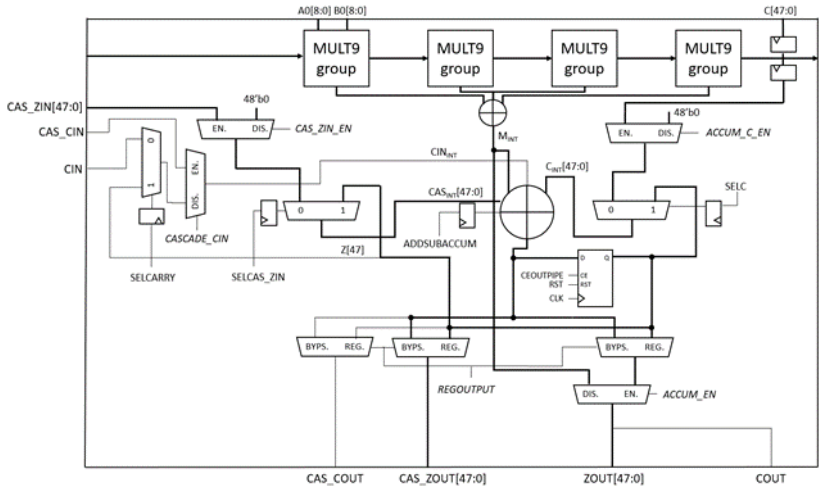
Table 164: Parameters (Continued)

Name	Values	Description
REGSHIFTOUTA2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A2
REGSHIFTOUTB2	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B2
REGINPUTA3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A3
REGINPUTB3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B3
REGSHIFTOUTA3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A3
REGSHIFTOUTB3	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B3
REGINPUTC	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand C
REGPIPE	"REGISTERED" (default) "BYPASSED"	Register pipeline
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
REGCAS_ZOUT	"REGISTERED" (default) "BYPASSED"	Register CAS_COUT and CAS_ZOUT[47:0]
REGACCUMCONTROLS	"REGISTERED" (default) "BYPASSED"	Register accumulator controls (CIN, SELCARRY, SELCAS_ZIN, ADDSUBACCUM, SELC)
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
ACCUM_EN	"DISABLED" (default) "ENABLED"	Enable accumulator
ACCUM_C_EN	"DISABLED" (default) "ENABLED"	Enable dynamic C for accumulator input (1) or tie to constant 0 (0)

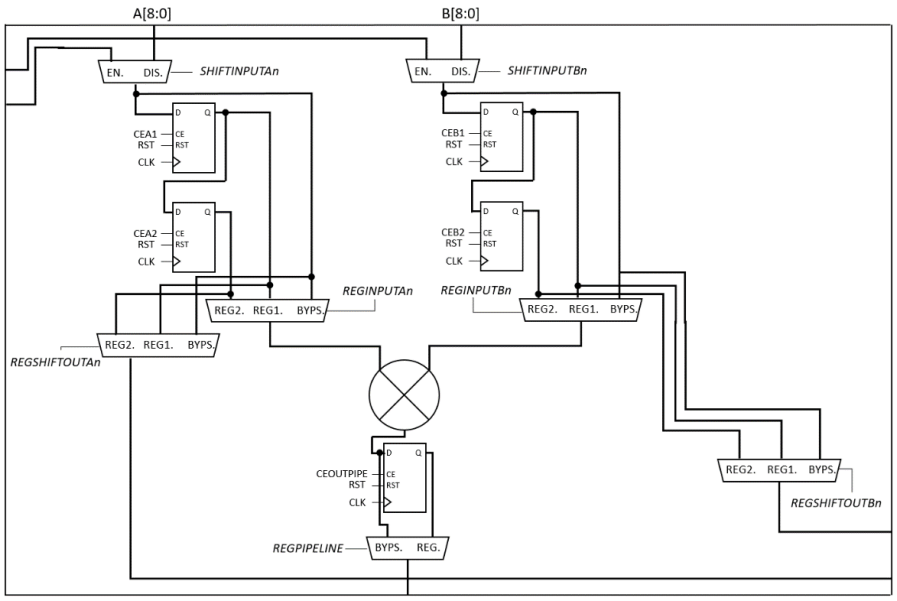
Table 164: Parameters (Continued)

Name	Values	Description
CAS_ZIN_EN	"DISABLED" (default) "ENABLED"	Enable dynamic CAS_ZIN for accumulator input (1) or tie to constant 0 (0)
CAS_CIN_EN	"DISABLED" (default) "ENABLED"	Use CAS_CIN for carry input
ROUNDMODE	"ROUND_TO_ZERO" (default) "ROUND_TO_INFINITY"	Rounding mode
SATURATION	"DISABLED" (default) "TO_NEG_MAX" "TO_ZERO"	Saturation enable
SATURATION_BITS	"48" (default) "3-48"	Saturation bits
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:



The function of "MULT9 group" in the above diagram is shown below.



When the accumulator is enabled, it implements the function:

M_INT = (A_0 * B_0) + (A_1 * B_1) + (A_2 * B_2) + (A_3 * B_3)

Z_out = CAS_INT + C_INT + CIN_INT +/- M_INT

When the accumulator is disabled, it implements the function:

Z_out = (A_0 * B_0) + (A_1 * B_1) + (A_2 * B_2) + (A_3 * B_3)

See the first diagram for the internal node corresponding to the above terms.

Although not shown in the above diagram, the register for accumulator inputs CIN, CAS_ZIN_EN, SELCAS_ZIN, and ADDSUBACCUM is bypassable with the parameter REGACCUMCONTROLS.

DP16K

16Kb Dual Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

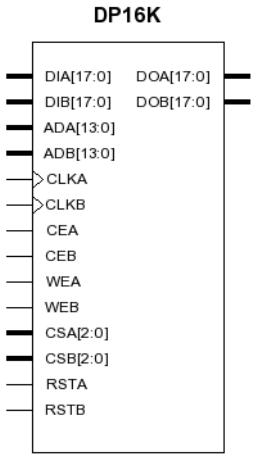


Table 165: Input Ports

Name	Description
DIA[17:0]	Port A data in
DIB[17:0]	Port B data in
ADA[13:0]	Port A address
ADB[13:0]	Port B address
CLKA	Port A clock
CLKB	Port B clock
CEA	Port A clock enable
CEB	Port B clock enable
WEA	Port A write enable
WEB	Port A write enable
CSA[2:0]	Port A chip select
CSB[2:0]	Port B chip select
RSTA	Port A output register reset
RSTB	Port B output register reset

Table 166: Output Ports

Name	Description
DOA[17:0]	*Port A data out*
DOB[17:0]	Port B data out

Table 167: Parameters

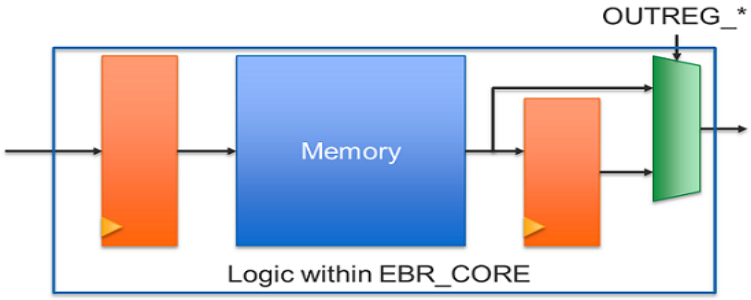
Name	Values	Description
DATA_WIDTH_A	"X18" *(default)* "X9" "X4" "X2" "X1"	Port A data width
DATA_WIDTH_B	"X18" *(default)* "X9" "X4" "X2" "X1"	Port B data width
OUTREG_A	"BYPASSED" *(default)* "USED"	Register port A output
OUTREG_B	"BYPASSED" *(default)* "USED"	Register port B output
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset for the output registers
RESETMODE_A	"SYNC" *(default)* "ASYNC"	Port A synchronous/ asynchronous reset control
RESETMODE_B	"SYNC" *(default)* "ASYNC"	Port B synchronous/ asynchronous reset control
INITVAL_00...INITVAL_3F	Hex string, 320 bits, must include the "0x" prefix	EBR initialization data
CSDECODE_A	000 *(default)* 001 010 011 100 101 110 111	Port A chip select active setting
CSDECODE_B	000 *(default)* 001 010 011 100 101 110 111	Port B chip select active setting
ASYNC_RST_RELEASE_A	"SYNC" *(default)* "ASYNC"	Port A synchronous/ asynchronous reset release

Table 167: Parameters (Continued)

Name	Values	Description
ASYNC_RST_RELEASE_B	"SYNC" *(default)* "ASYNC"	Port B synchronous/ asynchronous reset release
INIT_DATA	"STATIC" (default) "DYNAMIC" "NO_INIT"	

It is **not** legal to drive the two clocks such that their active edges switch simultaneously.

All inputs (except the register control signals) and all outputs are registered in the following manner:



Port assignments for different data widths follow the table below:

Data Width	Input Data	Output Data	Address A (MSB to LSB)	Address B (MSB to LSB)
16Kx1	DI[0]	DO[0]	ADA[13:0]	ADB[13:0]
8Kx2	DI[1:0]	DO[1:0]	ADA[13:1]	ADB[13:1]
4Kx4	DI[3:0]	DO[3:0]	ADA[13:2]	ADB[13:2]
2Kx9	DI[8:0]	DO[8:0]	ADA[13:3]	ADB[13:3]
1Kx18	DI[17:0]	DO[17:0]	ADA[13:4]	ADB[13:4]

DP32K

32 Kb Dual Port Block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

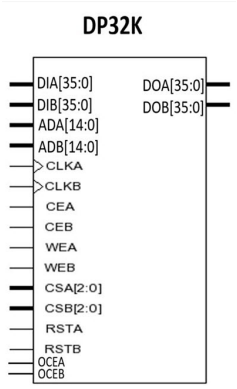


Table 168: Input Ports

Name	Range	Type	Active State	Description
DIA	35:0	Data	Active High	A-port Data Inputs
DIB	35:0	Data	Active High	B-port Data Inputs
ADA	14:0	Addr	Active High	A-port Address
ADB	14:0	Addr	Active High	B-port Address
CLKA	1	Clock	Rising Edge	A-port Clock Input
CLKB	1	Clock	Rising Edge	B-port Clock Input
CEA	1	Clock Enable	Active High	A-port Clock Enable
CEB	1	Clock Enable	Active High	B-port Clock Enable
OCEA	1	Control	Active High	A-port Output Clock Register Enable
OCEB	1	Control	Active High	B-port Output Clock Register Enable
WEA	1	Control	Active High	A-port Write Enable 1: Write 0: Read
WEB	1	Control	Active High	B-port Write Enable 1: Write 0: Read
CSA	2.0	Control	Active High	A-port Cascade Select
CSB	2.0	Control	Active High	B-port Cascade Select
RSTA	1	Reset	Active High	A-port output registers reset
RSTB	1	Reset	Active High	B-port output registers reset

Table 169: Output Ports

Name	Range	Type	Active State	Description
DOA	35:0	Data	Active High	A-port Data Output
DOB	35:0	Data	Active High	B-port Data Output

Table 170: Parameters

Name	Values	Description
DATA_WIDTH_A	“X36”(default) "X18" "X9" "X4" "X2" "X1"	Port A data width
DATA_WIDTH_B	“X36”(default) "X18" "X9" "X4" "X2" "X1"	Port B data width
OUTREG_A	"BYPASSED" (default) "USED"	Register port A output
OUTREG_B	"BYPASSED" (default) "USED"	Register port B output
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
RESETMODE_A	"SYNC" (default) "ASYNC"	Port A sync/async reset control
RESETMODE_B	"SYNC" (default) "ASYNC"	Port B sync/async reset control
CSDECODE_A	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Port A chip select active setting

Table 170: Parameters (Continued)

CSDECODE_B	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Port B chip select active setting
ASYNC_RST_RELEASE_A	"SYNC" (default) "ASYNC"	Port A sync/async reset release
ASYNC_RST_RELEASE_B	"SYNC" (default) "ASYNC"	Port B sync/async reset release
INITVAL_00...INITVAL_7F	Hex string, 320 bits, must include the "0x" prefix	Initialization value
INIT_DATA	"STATIC" (default) "DYNAMIC"	Specifies if the initialization data (as defined by the INITVAL parameters) may be changed after the synthesis stage. When INIT_DATA=STATIC, it is assumed that the initialization data will not change after synthesis, so an optimization to reduce the bitstream size can be applied. When INIT_DATA=DYNAMIC it is assumed that the initialization data may change after synthesis, so the aforementioned optimization is disabled.

Note:

If writing and reading at the same address simultaneously, the behavior is not guaranteed (i.e. either the read or write may happen first).

DPHY

The MIPI wrapper IP is used for MIPI DPHY v1.2: 4 channels DPHY with VCC= 0.9V/1.0V

Warning

You should not try to use this primitive directly. Instead, use the related IP in IP Catalog. The following information is for reference only.

Architectures supported:

- LIFCL-40, LIFCL-17

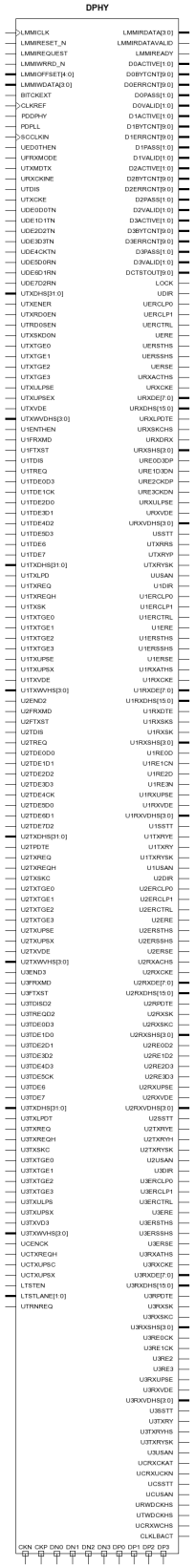

DPHY
LMMICLK
LMMIRESET_N
LMMIREQUEST
LMMIWRRD_N
LMMIOFFSET[4:0]
LMMIWRDATA[3:0]
BITCKEXT
CLKREF
PDDPHY
PDPLL
SCCLKIN
UED0THEN
UFRXMODE
UTXMDTX
URXCKINE
UTDIS
UTXCKE
UDE0D0TN
UDE1D1TN
UDE2D2TN
UDE3D3TN
UDE4CKTN
UDE5D0RN
UDE6D1RN
UDE7D2RN
UTXDHS[31:0]
UTXENER
UTXRD0EN
UTRDISEN
UTXSKD0N
UTXTGE0
UTXTGE1
UTXTGE2
UTXTGE3
UTXULPSE
UTXULPSEX
UTXVDE
UTXWVDHS[3:0]
U1ENTHEN
U1FRXMD
U1FTXST
U1TDIS
U1TREQ
U1TDE0D3
U1TDE1CK
U1TDE2D0
U1TDE3D1
U1TDE4D2
U1TDE5D3
U1TDE6
U1TDE7
U1TXDHS[31:0]
U1TXLPD
U1TXREQ
U1TXREQH
U1TXSK
U1TXTGE0
U1TXTGE1
U1TXTGE2
U1TXTGE3
U1TXULPSE
U1TXULPSX
U1TXVDE
U1TXWVHS[3:0]
U2ENC2
U2FRXMD
U2FTXST
U2TDIS
U2TREQ
U2TDE0D0
U2TDE1D1
U2TDE2D2
U2TDE3D3
U2TDE4CK
U2TDE5D0
U2TDE6D1
U2TDE7D2
U2TXDHS[31:0]
U2TRDTE
U2TXREQ
U2TXREQH
U2TXSKC
U2TXTGE0
U2TXTGE1
U2TXTGE2
U2TXTGE3
U2TXUPSE
U2TXULPSX
U2TXVDE
U2TXWVHS[3:0]
U3ENC3
U3FRXMD
U3FTXST
U3TDISD2
U3TREQD2
U3TDE0D3
U3TDE1D0
U3TDE2D1
U3TDE3D2
U3TDE4D3
U3TDE5CK
U3TDE6
U3TDE7
U3TXDHS[31:0]
U3TXLPDT
U3TXREQ
U3TXREQH
U3TXSKC
U3TXTGE0
U3TXTGE1
U3TXTGE2
U3TXTGE3
U3TXULPS
U3TXUPSX
U3TXVD3
U3TXWVHS[3:0]
UCENCK
UCTXREQH
UCTXUPSC
UCTXULPSX
LTSTEN
LTSTLANE[1:0]
UTRNREQ
LMMIRDATA[3:0]
LMMIRDATAVALID
LMMIREADY
D0ACTIVE[1:0]
D0BYTCNT[9:0]
D0ERRCNT[9:0]
D0PASS[1:0]
D0VALID[1:0]
D1ACTIVE[1:0]
D1BYTCNT[9:0]
D1ERRCNT[9:0]
D1PASS[1:0]
D1VALID[1:0]
D2ACTIVE[1:0]
D2BYTCNT[9:0]
D2ERRCNT[9:0]
D2PASS[1:0]
D2VALID[1:0]
D3ACTIVE[1:0]
D3BYTCNT[9:0]
D3ERRCNT[9:0]
D3PASS[1:0]
D3VALID[1:0]
DCTSTOUT[9:0]
LOCK
UDIR
UERCLP0
UERCLP1
UERCTRL
UERE
UERSTHS
UERSSHS
UERSE
URXACTHS
URXCKE
URXDE[7:0]
URXDHS[15:0]
URXLPDTE
URXSKCHS
URXDRX
URXSHS[3:0]
URE0D3DP
URE1D3DN
URE2CKDP
URE3CKDN
URXULPSE
URXVDE
URXVDHS[3:0]
USSTT
UTXRRS
UTXRY
UTXRYSK
UUSAN
U1DIR
U1ERCLP0
U1ERCLP1
U1ERCTRL
U1ERE
U1ERSTHS
U1ERSSHS
U1ERSE
U1RXATHS
U1RXCKE
U1RXDE[7:0]
U1RXDHS[15:0]
U1RXDTE
U1RXSKS
U1RXSK
U1RXSHS[3:0]
U1RE0D
U1RE1CN
U1RE2D
U1RE3N
U1RXUPSE
U1RXVDE
U1RXVDHS[3:0]
U1SSTT
U1TXRYE
U1TXRY
U1TXRYSK
U1USAN
U2DIR
U2ERCLP0
U2ERCLP1
U2ERCTRL
U2ERE
U2ERSTHS
U2ERSSHS
U2ERSE
U2RXACHS
U2RXCKE
U2RXDE[7:0]
U2RXDHS[15:0]
U2RPDTE
U2RXSK
U2RXSKC
U2RXSHS[3:0]
U2RE0D2
U2RE1D2
U2RE2D3
U2RE3D3
U2RXUPSE
U2RXVDE
U2RXVDHS[3:0]
U2SSTT
U2TXRYE
U2TXRYH
U2TXRYSK
U2USAN
U3DIR
U3ERCLP0
U3ERCLP1
U3ERCTRL
U3ERE
U3ERSTHS
U3ERSSHS
U3ERSE
U3RXATHS
U3RXCKE
U3RXDE[7:0]
U3RXDHS[15:0]
U3RPDTE
U3TXSK
U3RXSKC
U3RXSHS[3:0]
U3RE0CK
U3RE1CK
U3RE2
U3RE3
U3RXUPSE
U3RXVDE
U3RXVDHS[3:0]
U3SSTT
U3TXRY
U3TXRYHS
U3TXRYSK
U3USAN
UCRXCXAT
UCRXUCKN
UCSSTT
UCUSAN
URWDCKHS
UTWDCKHS
UCRXWCHS
CLKLBACT
CKN
CKP
DN0
DN1
DN2
DN3
DP0
DP1
DP2
DP3

Table 171: Input Ports

Name	Description
LMMICLK	
LMMIRESET_N	
LMMIREQUEST	
LMMIWRRD_N	
LMMIOFFSET[4:0]	
LMMIWDATA[3:0]	
BITCKEXT	
CLKREF	
PDDPHY	
PDPLL	
SCCLKIN	
UED0THEN	
UFRXMODE	
UTXMDTX	
URXCKINE	
UTDIS	
UTXCKE	
UDE0D0TN	
UDE1D1TN	
UDE2D2TN	
UDE3D3TN	
UDE4CKTN	
UDE5D0RN	
UDE6D1RN	
UDE7D2RN	
UTXDHS[31:0]	
UTXENER	
UTXRD0EN	
UTRD0SEN	
UTXSKD0N	
UTXTGE0	
UTXTGE1	

Table 171: Input Ports (Continued)

Name	Description
UTXTGE2	
UTXTGE3	
UTXULPSE	
UTXUPSEX	
UTXVDE	
UTXWVDHS[3:0]	
U1ENTHEN	
U1FRXMD	
U1FTXST	
U1TDIS	
U1TREQ	
U1TDE0D3	
U1TDE1CK	
U1TDE2D0	
U1TDE3D1	
U1TDE4D2	
U1TDE5D3	
U1TDE6	
U1TDE7	
U1TXDHS[31:0]	
U1TXLPD	
U1TXREQ	
U1TXREQH	
U1TXSK	
U1TXTGE0	
U1TXTGE1	
U1TXTGE2	
U1TXTGE3	
U1TXUPSE	
U1TXUPSX	
U1TXVDE	
U1TXWVHS[3:0]	

Table 171: Input Ports (Continued)

Name	Description
U2END2	
U2FRXMD	
U2FTXST	
U2TDIS	
U2TREQ	
U2TDE0D0	
U2TDE1D1	
U2TDE2D2	
U2TDE3D3	
U2TDE4CK	
U2TDE5D0	
U2TDE6D1	
U2TDE7D2	
U2TXDHS[31:0]	
U2TPDTE	
U2TXREQ	
U2TXREQH	
U2TXSKC	
U2TXTGE0	
U2TXTGE1	
U2TXTGE2	
U2TXTGE3	
U2TXUPSE	
U2TXUPSX	
U2TXVDE	
U2TXWVHS[3:0]	
U3END3	
U3FRXMD	
U3FTXST	
U3TDISD2	
U3TREQD2	
U3TDE0D3	

Table 171: Input Ports (Continued)

Name	Description
U3TDE1D0	
U3TDE2D1	
U3TDE3D2	
U3TDE4D3	
U3TDE5CK	
U3TDE6	
U3TDE7	
U3TXDHS[31:0]	
U3TXLPDT	
U3TXREQ	
U3TXREQH	
U3TXSKC	
U3TXTGE0	
U3TXTGE1	
U3TXTGE2	
U3TXTGE3	
U3TXULPS	
U3TXUPSX	
U3TXVD3	
U3TXWVHS[3:0]	
UCENCK	
UCTXREQH	
UCTXUPSC	
UCTXUPSX	
LTSTEN	
LTSTLANE[1:0]	
UTRNREQ	

Table 172: Output Ports

Name	Description
LMMIRDATA[3:0]	
LMMIRDATAVALID	

Table 172: Output Ports (Continued)

Name	Description
LMMIREADY	
D0ACTIVE[1:0]	
D0BYTCNT[9:0]	
D0ERRCNT[9:0]	
D0PASS[1:0]	
D0VALID[1:0]	
D1ACTIVE[1:0]	
D1BYTCNT[9:0]	
D1ERRCNT[9:0]	
D1PASS[1:0]	
D1VALID[1:0]	
D2ACTIVE[1:0]	
D2BYTCNT[9:0]	
D2ERRCNT[9:0]	
D2PASS[1:0]	
D2VALID[1:0]	
D3ACTIVE[1:0]	
D3BYTCNT[9:0]	
D3ERRCNT[9:0]	
D3PASS[1:0]	
D3VALID[1:0]	
DCTSTOUT[9:0]	
LOCK	
UDIR	
UERCLP0	
UERCLP1	
UERCTRL	
UERE	
UERSTHS	
UERSSHS	
UERSE	
URXACTHS	

Table 172: Output Ports (Continued)

Name	Description
URXCKE	
URXDE[7:0]	
URXDHS[15:0]	
URXLPDTE	
URXSKCHS	
URXDRX	
URXSHS[3:0]	
URE0D3DP	
URE1D3DN	
URE2CKDP	
URE3CKDN	
URXULPSE	
URXVDE	
URXVDHS[3:0]	
USSTT	
UTXRRS	
UTXRYP	
UTXRYSK	
UUSAN	
U1DIR	
U1ERCLP0	
U1ERCLP1	
U1ERCTRL	
U1ERE	
U1ERSTHS	
U1ERSSHS	
U1ERSE	
U1RXATHS	
U1RXCKE	
U1RXDE[7:0]	
U1RXDHS[15:0]	
U1RXDTE	

Table 172: Output Ports (Continued)

Name	Description
U1RXSKS	
U1RXSK	
U1RXSHS[3:0]	
U1RE0D	
U1RE1CN	
U1RE2D	
U1RE3N	
U1RXUPSE	
U1RXVDE	
U1RXVDHS[3:0]	
U1SSTT	
U1TXRYE	
U1TXRY	
U1TXRYSK	
U1USAN	
U2DIR	
U2ERCLP0	
U2ERCLP1	
U2ERCTRL	
U2ERE	
U2ERSTHS	
U2ERSSHS	
U2ERSE	
U2RXACHS	
U2RXCKE	
U2RXDE[7:0]	
U2RXDHS[15:0]	
U2RPDTE	
U2RXSK	
U2RXSKC	
U2RXSHS[3:0]	
U2RE0D2	

Table 172: Output Ports (Continued)

Name	Description
U2RE1D2	
U2RE2D3	
U2RE3D3	
U2RXUPSE	
U2RXVDE	
U2RXVDHS[3:0]	
U2SSTT	
U2TXRYE	
U2TXRYH	
U2TXRYSK	
U2USAN	
U3DIR	
U3ERCLP0	
U3ERCLP1	
U3ERCTRL	
U3ERE	
U3ERSTHS	
U3ERSSHS	
U3ERSE	
U3RXATHS	
U3RXCKE	
U3RXDE[7:0]	
U3RXDHS[15:0]	
U3RPDTE	
U3RXSK	
U3RXSKC	
U3RXSHS[3:0]	
U3RE0CK	
U3RE1CK	
U3RE2	
U3RE3	
U3RXUPSE	

Table 172: Output Ports (Continued)

Name	Description
U3RXVDE	
U3RXVDHS[3:0]	
U3SSTT	
U3TXRY	
U3TXRYHS	
U3TXRYSK	
U3USAN	
UCRXCKAT	
UCRXUCKN	
UCSSTT	
UCUSAN	
URWDCKHS	
UTWDCKHS	
UCRXWCHS	
CLKLBACT	

Table 173: Bidirectional Ports

Name	Description
CKN	
CKP	
DN0	
DN1	
DN2	
DN3	
DP0	
DP1	
DP2	
DP3	

Table 174: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	
AUTO_PD_EN	"POWERED_UP" *(default)* "POWERED_DOWN"	Powers down inactive lanes reported by CFG_NUM_LANES input bus
CFG_NUM_LANES	"ONE_LANE" *(default)* "TWO_LANES" "THREE_LANES" "FOUR_LANES"	Sets the number of active lanes
CM	0'b00000000 *(default)*	PLL dividers should be set to produce the required high speed BITCLK to be used for testing
CN	0'b00000 *(default)*	PLL dividers should be set to produce the required high speed BITCLK to be used for testing
CO	0'b000 *(default)*	PLL dividers should be set to produce the required high speed BITCLK to be used for testing
CONT_CLK_MODE	"DISABLED" *(default)* "ENABLED"	This static input pin enables the slave clock lane feature to maintain HS reception state during continuous clock mode operation, despite line glitches
DESKEW_EN	"DISABLED" *(default)* "ENABLED"	Enables Deskew feature that modifies ERRSYNC/NOSYNC behavior
DSI_CSI	"CSI2_APP" *(default)* "DSI_APP"	Selects the PHY IP Application
EN_CIL	"CIL_ENABLED" *(default)* "CIL_BYPASSED"	This signal should start with 1b1 then move to 1b0 to reset internal registers when in CIL Bypass mode
HSEL	"DISABLED" *(default)* "ENABLED"	High Speed Select
LANE0_SEL	"LANE_0" *(default)* "LANE_1" "LANE_2" "LANE_3"	This input determines which lane will act as data lane0 in HS Operation mode
LOCK_BYP	"GATE_TXBYTECLKHS" *(default)* "NOT_GATE_TXBYTECLKHS"	When clock lane exits from ULPS, this input determines if the PLL LOCK signal will be used to gate the TxByteClkHS
MASTER_SLAVE	"SLAVE" *(default)* "MASTER"	Should be set to 1 for HS-TX and LP-TX Tests and set to 0 for remaining tests
PLLCLKBYPASS	"REGISTERED" *(default)* "BYPASSED"	PLL Clock bypass
RSEL	0'b00 *(default)*	

Table 174: Parameters (Continued)

Name	Values	Description
RXCDRP	0'b00 *(default)*	On-chip termination control bits for manual calibration of HS-RX
RXDATAWIDTHHS	0'b00 *(default)*	One-time programming bits that adjust the threshold voltage of LP-CD default setting 2b01
RXLPRP	0'b000 *(default)*	High-Speed Receive Data Width Select
TEST_ENBL	0'b000000 *(default)*	Enable TEST_PATTERN
TEST_PATTERN	0'b00000000000000000000000000000000 *(default)*	32-bit BIST pattern to check loopback.
TST	0'b1001 *(default)* 0000	PLL charge pump current control
TXDATAWIDTHHS	0'b00 *(default)*	High-Speed Transmit Data Width Select
U_PRG_HS_PREPARE	0'b00 *(default)*	Bits used to program T_HS_PREPARE time in the beginning of high speed transmission mode
U_PRG_HS_TRAIL	0'b000000 *(default)*	Bits used to program T_HS_TRAIL time in the end of high speed transmission mode
U_PRG_HS_ZERO	0'b000000 *(default)*	Bits used to program T_HS_ZERO time in the beginning of high speed transmission mode
U_PRG_RXHS_SETTLE	0'b000000 *(default)*	Bits used to program T_HS_SETTLE. HS-RX waits for Time-out T_HS_SETTLE in order to neglect transition effects
UC_PRG_HS_PREPARE	"1P0_TXCLKESC" *(default)* "1P5_TXCLKESC"	
UC_PRG_HS_TRAIL	0'b00000 *(default)*	
UC_PRG_HS_ZERO	0'b0000000 *(default)*	
UC_PRG_RXHS_SETTLE	0'b000000 *(default)*	

DPR16X4

Distributed Pseudo Dual Port RAM with Synchronous Write and Asynchronous Read

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

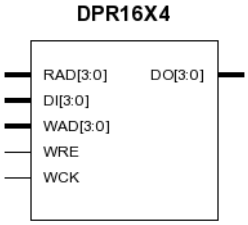


Table 175: Input Ports

Name	Description
DI[3:0]	Data in
WAD[3:0]	Write address
WCK	Write clock
WRE	Write enable
RAD[3:0]	Read address

Table 176: Output Ports

Name	Description
DO[3:0]	Data out

Table 177: Parameters

Name	Values	Description
INITVAL	"0x0000000000000000" *(default)*	16-digit hex string, 64 bits total

Table 178: INIT Setting

Address	Bit Index 3	Bit Index 2	Bit Index 1	Bit Index 0
Address 0	INIT[48]	INIT[32]	INIT[16]	INIT[0]
Address 1	INIT[49]	INIT[33]	INIT[17]	INIT[1]
Address 2	INIT[50]	INIT[34]	INIT[18]	INIT[2]
Address 3	INIT[51]	INIT[35]	INIT[19]	INIT[3]
Address 4	INIT[52]	INIT[36]	INIT[20]	INIT[4]
Address 5	INIT[53]	INIT[37]	INIT[21]	INIT[5]
Address 6	INIT[54]	INIT[38]	INIT[22]	INIT[6]
Address 7	INIT[55]	INIT[39]	INIT[23]	INIT[7]
Address 8	INIT[56]	INIT[40]	INIT[24]	INIT[8]
Address 9	INIT[57]	INIT[41]	INIT[25]	INIT[9]

Table 178: INIT Setting (Continued)

Address	Bit Index 3	Bit Index 2	Bit Index 1	Bit Index 0
Address 10	INIT[58]	INIT[42]	INIT[26]	INIT[10]
Address 11	INIT[59]	INIT[43]	INIT[27]	INIT[11]
Address 12	INIT[60]	INIT[44]	INIT[28]	INIT[12]
Address 13	INIT[60]	INIT[45]	INIT[29]	INIT[13]
Address 14	INIT[62]	INIT[46]	INIT[30]	INIT[14]
Address 15	INIT[63]	INIT[47]	INIT[31]	INIT[15]

For the output to be Data = Address (Address 0 = Data 0 and so on), the INIT must be:

INIT = 0xFF00F0F0CCCCAAAA

LUT3.INIT = "FF00"

LUT2.INIT = "F0F0"

LUT1.INIT = "CCCC"

LUT0.INIT = "AAAA"

DPR16X4A

Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read

Architectures supported:

- LAV-AT-ES1
- LN2-CT

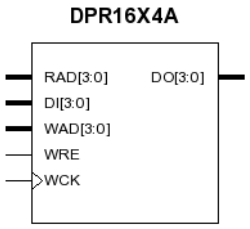


Table 179: Input Ports

Name	Range	Type	Description
RAD	3:0	Addr	Read address
DI	3:0	Data	Write data
WAD	3:0	Addr	Write address
WRE	1	Control	Write enable
WCK	1	Clock	Write clock

Table 180: Output Ports

Name	Range	Type	Description
DO	3:0	Data	Read data

Table 181: Parameters

Name	Values	Description
INITVAL	"0x0000000000000000" (default)	RAM initialization value

DPR32X2

Distributed Pseudo Dual Port RAM. Synchronous write, asynchronous read

Architectures supported:

- LAV-AT-ES1
- LN2-CT

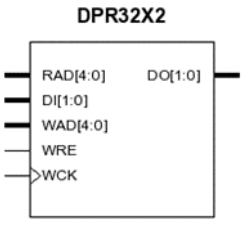


Table 182: Input Ports

Name	Range	Type	Description
RAD	4:0	Addr	Read address
DI	1:0	Data	Write data
WAD	4:0	Addr	Write address
WRE	1	Control	Write enable
WCK	1	Clock	Write clock

Table 183: Output Ports

Name	Range	Type	Description
DO	1:0	Data	Read data

Table 184: Parameters

Name	Values	Description
INITVAL	"0x0000000000000000" (default)	RAM initialization value

DPSC512K

512Kb Single Clock Dual Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

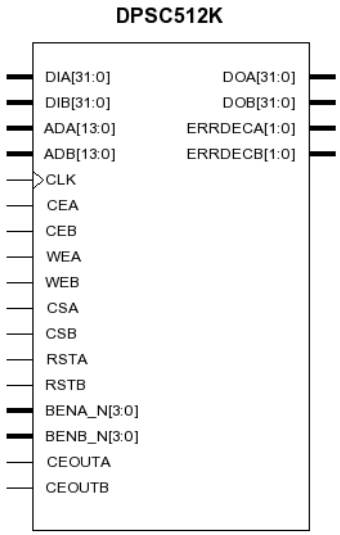


Table 185: Input Ports

Name	Description
DIA[31:0]	Port A data in
DIB[31:0]	Port B data in
ADA[13:0]	Port A address
ADB[13:0]	Port B address
CLK	Clock
CEA	Port A clock enable
CEB	Port B clock enable
WEA	Port A write enable
WEB	Port A write enable
CSA	Port A chip select
CSB	Port B chip select
RSTA	Port A output register reset
RSTB	Port B output register reset
BENA_N[3:0]	Write byte enable, 0=write 1=disable. BENA_N[3] corresponds to the MSB of DI, BENA_N[0] corresponds to the LSB
BENB_N[3:0]	Write byte enable, 0=write 1=disable. BENB_N[3] corresponds to the MSB of DI, BENB_N[0] corresponds to the LSB
CEOUTA	Port A output register clock enable
CEOUTB	Port B output register clock enable

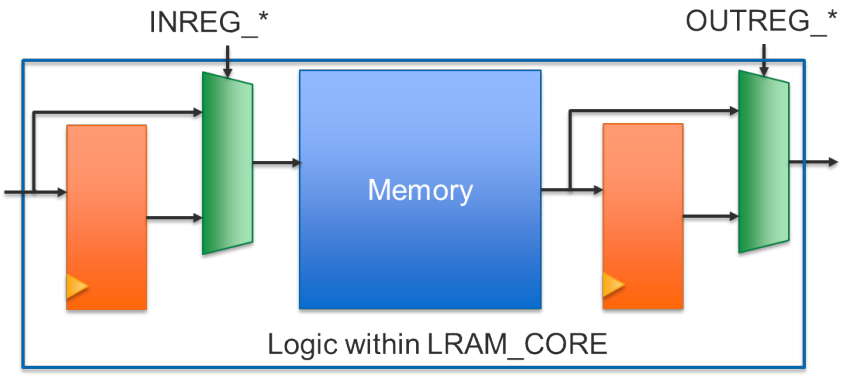
Table 186: Output Ports

Name	Description
DOA[31:0]	Port A data out
DOB[31:0]	Port B data out
ERRDECA[1:0]	Port A one and two bit error flag
ERRDECB[1:0]	Port B one and two bit error flag

Table 187: Parameters

Name	Values	Description
OUTREG_A	"NO_REG" *(default)* "OUT_REG"	Register output A
OUTREG_B	"NO_REG" *(default)* "OUT_REG"	Register output B
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(default)* "ASYNC"	Reset async/sync control
INITVAL_00...INITVAL_7F	Hex string, 5120 bits, must include the "0x" prefix	LRAM initialization data
ASYNC_RESET_RELEASE	"SYNC" *(default)* "ASYNC"	Reset release async/sync
ECC_BYTE_SEL	"ECC_EN" (default) "BYTE_EN"	Enable ECC or Byte-enable support

All inputs (except the register control signals) and all outputs are registered in the following manner:



DQSBUF

To support DDR memory interfaces (DDR2/3, LPDDR2/3), the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP

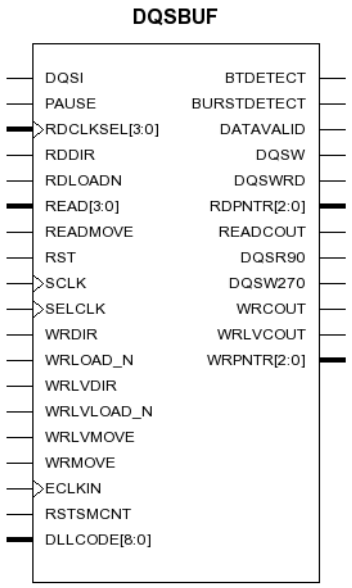


Table 188: Input Ports

Name	Description
DQSI	DQS signal from IO through the PIC
PAUSE	To stop ECLK for DDR3 Write leveling and DLL code update
RDCLKSEL[3:0]	Select read clock source and polarity control (from CIB)
RDDIR	0 to increase the code, 1to decrease the code for DDR read

Table 188: Input Ports (Continued)

Name	Description
RDLOADN	1b0 - When mc1_mt_en_read=1b1 and read_load_n=1b0 the read_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_read,mc1_s_read[8:0]} value; 1b1 - When counter has preload value, read_move pulse can be used to increment and decrement the counter based on the read_direction signal's value and mc1_mt_en_write should be set 1b1.
READ[3:0]	Read signal for DDR read mode (from CIB)
READMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the read_direction port.
RST	DQS reset control for both DDR/CDR modes (from CIB)
SCLK	SCLK from SCLK tree (CIB)
SELCLK	Select the clock to be used between the output of the read section's delay cell or sclk
WRDIR	0 - to increase the code, 1 - to decrease the code for DDR write
WRLOAD_N	1b0 - When mc1_mt_en_write=1b1 and write_load_n=1b0 the write_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_write,mc1_s_write[8:0]} value; 1b1 - When counter has preload value, write_move pulse can be used to increment and decrement the counter based on the write_direction signal's value and mc1_mt_en_write should be set 1b1.
WRLVDIR	0 - to increase the code, 1 - to decrease the code for DDR write leveling
WRLVLOAD_N	1b0 - 9-bit counter in reset operation; 1b1 - When mc1_mt_en_write_leveling=1b1 and write_leveling_load_n=1b1 the counter can be incremented/ decremented based on the direction signal using the write_leveling_move signal
WRLVMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the write_leveling_direction port.
WRMOVE	Move pulse needs to be at least 1 sclk cycle ans should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the write_direction port.
ECLKIN	ECLK from 4 different ECLK tree output

Table 188: Input Ports (Continued)

Name	Description
RSTSMCNT	Signal to reset the smoothing counters used for the Read, Write, and Write leveling delays.
DLLCODE[8:0]	DLL code selected from the DLL code routing mux

Table 189: Output Ports

Name	Description
BTDETECT	READ burst detect output (to CIB)
BURSTDETECT	The burst_det_sclk signal is generated using burst_det and is asserted on the rising edge of SCLK
DATAVALID	Data Valid Flag for READ mode (to CIB)
DQSW	ECLK phase shifted or delayed, goes to the dqsw tree through the PIC
DQSWRD	The read training clock adjusted in the write section. The read_clk_sel[3:0] determines the selected delay and read enable position
RDPNTR[2:0]	FIFO control READ pointer (3-bits) to FIFO in PIC (through each tree to IOL)
READCOUT	Margin test output flag for READ to indicate the underflow or overflow
DQSR90	DQSI phase shifted or delayed by 90 degree output (through DQSR tree to IOL)
DQSW270	ECLK phase shifted or delayed by 270 degree output (through DQSW270 tree to IOL)
WRCOUT	Margin test output flag for WRITE to indicate the underflow or overflow
WRLVCOUT	Margin test output flag for WRITE LEVELING to indicate the underflow or overflow
WRPNTR[2:0]	FIFO control WRITE pointer (3-bits) to FIFO in PIC (through each tree to IOL)

Table 190: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	GSRN signal enable/disable select
ENABLE_FIFO	"DISABLED" *(default)* "ENABLED"	Enable FIFO control
FORCE_READ	"DISABLED" *(default)* "ENABLED"	

Table 190: Parameters (Continued)

Name	Values	Description
FREE_WHEEL	"DDR" *(default)* "GDDR"	FIFO mode select
MODX	"NOT_USED" (default) "MDDRX2_WL_DDR2" "MDDRX2_WL_DDR3" "MDDRX4_WL_DDR3"	DQS mode selection for different DDRX2 modes: WL = 0T: MDDRX2 mode (DDR2/DDR3) without WL; WL = 1T: MDDRX2 mode (DDR3) with WL
MT_EN_READ	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for READ mode 1 - Enable
MT_EN_WRITE	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE mode 1 - Enable
MT_EN_WRITE_LEVELING	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE LEVELING mode 1 - Enable
RD_PNTR	0'b000 *(default)*	FIFO control READ pointer (3-bits) to FIFO in PIC (through each tree to IOL)
READ_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for read
RX_CENTERED	"ENABLED" *(default)* "DISABLED"	For GDDR-DQS RX Centered mode [delay code = 0]
S_READ	0 *(default)*	Offset for READ slave delay adjustment. For example: ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000000 = +0: add 0 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000010 = +2: add 2 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_11111111 = +255: add 255 unit delay
S_WRITE	0 *(default)*	Offset for WRITE slave delay adjustment. For example: ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_11111111 = -1: subtract 1 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000001 = -255: subtract 255 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000000 = -256: subtract 256 unit delay

Table 190: Parameters (Continued)

Name	Values	Description
SIGN_READ	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for READ slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]
SIGN_WRITE	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for WRITE slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]
UPDATE_QU	"UP1_AND_UP0_SAME" *(default)* "UP1_AHEAD_OF_UP0"	Update 0/1 timing relationship control adjustment. 0: update1 ahead of update0 one cycle 1: update1 and update0 same
WRITE_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for write
SEL_READ_BIT_ENABLE_CYCLES	"NORMAL" *(default)* "DELAYED_0P5_ECLK"	Used to select the cycle latency for the read enable signal generated by the read[3:0] bits and read_clk_sel[3:0].
BYPASS_WR_LEVEL_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write leveling smoothing path 1b1: Select the write leveling smoothing bypass path, counter control based on Sapphire
BYPASS_WR_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write smoothing path 1b1: Select the write smoothing bypass path, counter control based on Sapphire
BYPASS_READ_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the read smoothing path 1b1: Select the read smoothing bypass path, counter control based on Sapphire

Note:

The SIGN_READ, S_READ, SIGN_WRITE, and S_WRITE are all set to a value based on device performance. MAP will use fixed values for these components during the MAP.

To support DDR memory interfaces (DDR2/3, LPDDR2/3), the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads.

DQSBUFA

Block generating DQS Signal for Memory LPDDR4 Functionality

Architectures supported:

- LFCPNX
- LFMXO5T
- UT24CP

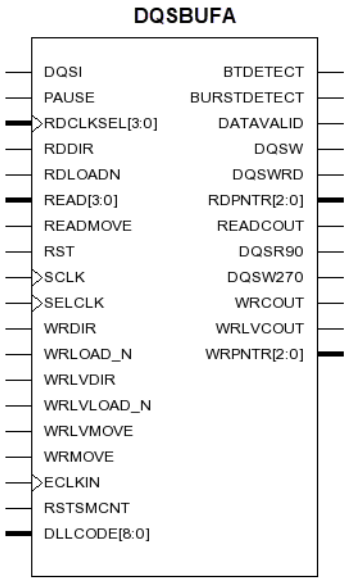


Table 191: Input Ports

Name	Description
DQSI	DQS signal from IO through the PIC
PAUSE	To stop ECLK for DDR3 Write leveling and DLL code update
RDCLKSEL[3:0]	Select read clock source and polarity control (from CIB)
RDDIR	0 to increase the code, 1to decrease the code for DDR read
RDLOADN	1b0 - When mc1_mt_en_read=1b1 and read_load_n=1b0 the read_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_read,mc1_s_read[8:0]} value; 1b1 - When counter has preload value, read_move pulse can be used to increment and decrement the counter based on the read_direction signal's value and mc1_mt_en_write should be set 1b1.
READ[3:0]	Read signal for DDR read mode (from CIB)

Table 191: Input Ports (Continued)

Name	Description
READMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the read_direction port.
RST	DQS reset control for both DDR/CDR modes (from CIB)
SCLK	SCLK from SCLK tree (CIB)
SELCLK	Select the clock to be used between the output of the read section's delay cell or sclk
WRDIR	0 - to increase the code, 1 - to decrease the code for DDR write
WRLOAD_N	1b0 - When mc1_mt_en_write=1b1 and write_load_n=1b0 the write_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_write,mc1_s_write[8:0]} value; 1b1 - When counter has preload value, write_move pulse can be used to increment and decrement the counter based on the write_direction signal's value and mc1_mt_en_write should be set 1b1.
WRLVDIR	0 - to increase the code, 1 - to decrease the code for DDR write leveling
WRLVLOAD_N	1b0 - 9-bit counter in reset operation; 1b1 - When mc1_mt_en_write_leveling=1b1 and write_leveling_load_n=1b1 the counter can be incremented/decremented based on the direction signal using the write_leveling_move signal
WRLVMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the write_leveling_direction port.
WRMOVE	Move pulse needs to be at least 1 sclk cycle ans should be greater than 5ns at TT corner. Pulse is used along with the eclk to generate the internal "mov" signal to update the counter by one value. The count up or down is determined by the write_direction port.
ECLKIN	ECLK from 4 different ECLK tree output
RSTSMCNT	Signal to reset the smoothing counters used for the Read, Write, and Write leveling delays.
DLLCODE[8:0]	DLL code selected from the DLL code routing mux

Table 192: Output Ports

Name	Description
BTDETECT	READ burst detect output (to CIB)
BURSTDETECT	The burst_det_sclk signal is generated using burst_det and is asserted on the rising edge of SCLK
DATAVALID	Data Valid Flag for READ mode (to CIB)
DQSW	ECLK phase shifted or delayed, goes to the dqsw tree through the PIC
DQSWRD	The read training clock adjusted in the write section. The read_clk_sel[3:0] determines the selected delay and read enable position
RDPNTR[2:0]	FIFO control READ pointer (3-bits) to FIFO in PIC (through each tree to IOL)
READCOUT	Margin test output flag for READ to indicate the underflow or overflow
DQSR90	DQSI phase shifted or delayed by 90 degree output (through DQSR tree to IOL)
DQSW270	ECLK phase shifted or delayed by 270 degree output (through DQSW270 tree to IOL)
WRCOUT	Margin test output flag for WRITE to indicate the underflow or overflow
WRLVCOUT	Margin test output flag for WRITE LEVELING to indicate the underflow or overflow
WRPNTR[2:0]	FIFO control WRITE pointer (3-bits) to FIFO in PIC (through each tree to IOL)

Table 193: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	GSRN signal enable/disable select
ENABLE_FIFO	"DISABLED" *(default)* "ENABLED"	Enable FIFO control
FORCE_READ	"DISABLED" *(default)* "ENABLED"	
FREE_WHEEL	"DDR" *(default)* "GDDR"	FIFO mode select
MODX	"NOT_USED" (default) "MDDRX2_WL_DDR2" "MDDRX2_WL_DDR3" "MDDRX4_WL_DDR3"	DQS mode selection for different DDRX2 modes: WL = 0T: MDDRX2 mode (DDR2/DDR3) without WL; WL = 1T: MDDRX2 mode (DDR3) with WL

Table 193: Parameters (Continued)

Name	Values	Description
MT_EN_READ	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for READ mode 1 - Enable
MT_EN_WRITE	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE mode 1 - Enable
MT_EN_WRITE_LEVELING	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE LEVELING mode 1 - Enable
RD_PNTR	0'b000 *(default)*	FIFO control READ pointer (3-bits) to FIFO in PIC (through each tree to IOL)
READ_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for read
RX_CENTERED	"ENABLED" *(default)* "DISABLED"	For GDDR-DQS RX Centered mode [delay code = 0]
S_READ	0 *(default)*	Offset for READ slave delay adjustment. For example: ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000000 = +0: add 0 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000010 = +2: add 2 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_11111111 = +255: add 255 unit delay
S_WRITE	0 *(default)*	Offset for WRITE slave delay adjustment. For example: ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_11111111 = -1: subtract 1 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000001 = -255: subtract 255 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000000 = -256: subtract 256 unit delay
SIGN_READ	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for READ slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]
SIGN_WRITE	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for WRITE slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]

Table 193: Parameters (Continued)

Name	Values	Description
UPDATE_QU	"UP1_AND_UP0_SAME" *(default)* "UP1_AHEAD_OF_UP0"	Update 0/1 timing relationship control adjustment. 0: update1 ahead of update0 one cycle 1: update1 and update0 same
WRITE_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for write
SEL_READ_BIT_ENABLE_CYCLES	"NORMAL" *(default)* "DELAYED_0P5_ECLK"	Used to select the cycle latency for the read enable signal generated by the read[3:0] bits and read_clk_sel[3:0].
BYPASS_WR_LEVEL_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write leveling smoothing path 1b1: Select the write leveling smoothing bypass path, counter control based on Sapphire
BYPASS_WR_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write smoothing path 1b1: Select the write smoothing bypass path, counter control based on Sapphire
BYPASS_READ_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the read smoothing path 1b1: Select the read smoothing bypass path, counter control based on Sapphire
LPDDR4_BL16_EN	"DISABLED" (default) "ENABLED"	Training burst length of DQS setting. Enabled: Enable burst length of 16 Disabled: Enable burst length of 8

Note:

The SIGN_READ, S_READ, SIGN_WRITE, and S_WRITE are all set to a value based on device performance. MAP will use fixed values for these components during the MAP.

DQSBUF_IVREF

To support DDR4 memory interfaces, the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads, and the LPDDR4 internal VREF generator should be used to generated trained reference voltage for LPDDR4 IO buffers. DQSBUF_IVREF combines the functions together.

Architectures supported:

- LFCPNX
- LFMXO5T

- LFMXO5TD
- UT24CP

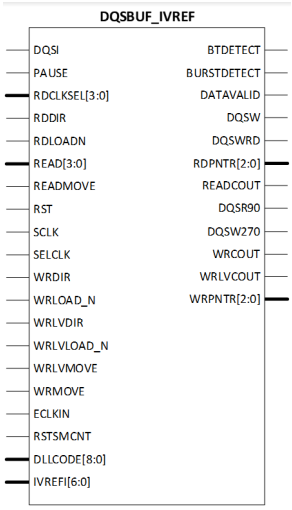


Table 194: Input Ports

Name	Description
DQSI	DQS signal from IO through the PIC
PAUSE	To stop ECLK for DDR3 Write leveling and DLL code update
RDCLKSEL[3:0]	Select read clock source and polarity control (from CIB)
RDDIR	0 to increase the code, 1to decrease the code for DDR read
RDLOADN	1b0 - When mc1_mt_en_read=1b1 and read_load_n=1b0 the read_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_read,mc1_s_read[8:0]} value. 1b1 - When counter has preload value, read_move pulse can be used to increment and decrement the counter based on the read_direction signal's value and mc1_mt_en_write should be set 1b1.
READ[3:0]	Read signal for DDR read mode (from CIB)
READMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5 ns at TT corner. Pulse is used along with the eclk to generate the internal “mov” signal to update the counter by one value. The count up or down is determined by the read_direction port.
RST	DQS reset control for both DDR/CDR modes (from CIB)
SCLK	SCLK from SCLK tree (CIB)
SELCLK	Select the clock to be used between the output of the read section's delay cell or sclk

Table 194: Input Ports (Continued)

Name	Description
WRDIR	0 - to increase the code 1 - to decrease the code for DDR write
WRLOAD_N	1b0 - When mc1_mt_en_write=1b1 and write_load_n=1b0 the write_move pulse needs to be generated to the load the preload value consisting of the {mc1_sign_write,mc1_s_write[8:0]} value. 1b1 - When counter has preload value, write_move pulse can be used to increment and decrement the counter based on the write_direction signal's value and mc1_mt_en_write should be set 1b1.
WRLVDIR	0 - to increase the code 1 - to decrease the code for DDR write leveling
WRLVLOAD_N	1b0 - 9-bit counter in reset operation 1b1 - When mc1_mt_en_write_leveling=1b1 and write_leveling_load_n=1b1 the counter can be incremented/ decremented based on the direction signal using the write_leveling_move signal
WRLVMOVE	Move pulse needs to be at least 1 sclk cycle and should be greater than 5 ns at TT corner. Pulse is used along with the eclk to generate the internal “mov” signal to update the counter by one value. The count up or down is determined by the write_leveling_direction port.
WRMOVE	Move pulse needs to be at least 1 sclk cycle ans should be greater than 5 ns at TT corner. Pulse is used along with the eclk to generate the internal “mov” signal to update the counter by one value. The count up or down is determined by the write_direction port.
ECLKIN	ECLK from 4 different ECLK tree output
RSTSMCNT	Signal to reset the smoothing counters used for the Read, Write, and Write leveling delays.
DLLCODE[8:0]	DLL code selected from the DLL code routing mux
IVREFI[6:0]	VREF output control from cib. The step size is 5 mV.

Table 195: Output Ports

Name	Description
BTDETECT	READ burst detect output (to CIB)
BURSTDETECT	The burst_det_sclk signal is generated using burst_det and is asserted on the rising edge of SCLK
DATAVALID	The burst_det_sclk signal is generated using burst_det and is asserted on the rising edge of SCLK
DQSW	ECLK phase shifted or delayed, goes to the dqsw tree through the PIC

Table 195: Output Ports (Continued)

Name	Description
DQSWRD	ECLK phase shifted or delayed, goes to the dqsw tree through the PIC
RDPNTR[2:0]	FIFO control READ pointer (3-bits) to FIFO in PIC (through each tree to IOL)
READCOUT	Margin test output flag for READ to indicate the underflow or overflow
DQSR90	DQSI phase shifted or delayed by 90 degree output (through DQSR tree to IOL)
DQSW270	ECLK phase shifted or delayed by 270 degree output (through DQSW270 tree to IOL)
WRCOUT	Margin test output flag for WRITE to indicate the underflow or overflow
WRLVCOUT	Margin test output flag for WRITE to indicate the underflow or overflow
WRPNTR[2:0]	FIFO control WRITE pointer (3-bits) to FIFO in PIC (through each tree to IOL)

Table 196: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	GSRN signal enable/disable select
ENABLE_FIFO	"DISABLED" *(default)* "ENABLED"	Enable FIFO control
FORCE_READ	"DISABLED" *(default)* "ENABLED"	
FREE_WHEEL	"DDR" *(default)* "GDDR"	FIFO mode select
MODX	"NOT_USED" (default) "MDDRX2_WL_DDR2" "MDDRX2_WL_DDR3" "MDDRX4_WL_DDR3"	DQS mode selection for different DDRX2 modes: WL = 0T: MDDRX2 mode (DDR2/DDR3) without WL; WL = 1T: MDDRX2 mode (DDR3) with WL
MT_EN_READ	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for READ mode 1 - Enable
MT_EN_WRITE	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE mode 1 - Enable
MT_EN_WRITE_LEVELING	"DISABLED" *(default)* "ENABLED"	0 - Disable margin test for WRITE LEVELING mode 1 - Enable

Table 196: Parameters (Continued)

Name	Values	Description
RD_PNTR	0'b000 *(default)*	FIFO control READ pointer (3 bits) to FIFO in PIC (through each tree to IOL)
READ_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for read
RX_CENTERED	"ENABLED" *(default)* "DISABLED"	For GDDR-DQS RX Centered mode [delay code = 0]
S_READ	0 *(default)*	Offset for READ slave delay adjustment. For example: ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000000 = +0: add 0 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_00000010 = +2: add 2 unit delay ▶ {mc1_sign_rd, mc1_s_rd[7:0]} = 0_11111111 = +255: add 255 unit delay
S_WRITE	0 *(default)*	Offset for WRITE slave delay adjustment. For example: ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_11111111 = -1: subtract 1 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000001 = -255: subtract 255 unit delay ▶ {mc1_sign_wr, mc1_s_wr[7:0]} = 1_00000000 = -256: subtract 256 unit delay
SIGN_READ	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for READ slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]
SIGN_WRITE	"POSITIVE" *(default)* "COMPLEMENT"	Sign bit for WRITE slave delay adjustment. 0: Positive [Addition] 1: Complement [Subtraction]
UPDATE_QU	"UP1_AND_UP0_SAME" *(default)* "UP1_AHEAD_OF_UP0"	Update 0/1 timing relationship control adjustment. 0: update1 ahead of update0 one cycle 1: update1 and update0 same
WRITE_ENABLE	"DISABLED" *(default)* "ENABLED"	DQS delay enable for write
SEL_READ_BIT_ENABLE_CYCLES	"NORMAL" *(default)* "DELAYED_0P5_ECLK"	Used to select the cycle latency for the read enable signal generated by the read[3:0] bits and read_clk_sel[3:0].

Table 196: Parameters (Continued)

Name	Values	Description
BYPASS_WR_LEVEL_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write leveling smoothing path 1b1: Select the write leveling smoothing bypass path, counter control based on Sapphire
BYPASS_WR_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the write smoothing path 1b1: Select the write smoothing bypass path, counter control based on Sapphire
BYPASS_READ_SMTH_LATCH	"SMOOTHING_PATH" *(default)* "BYPASS_PATH"	1b0: Select the read smoothing path 1b1: Select the read smoothing bypass path, counter control based on Sapphire
LPDDR4_BL16_EN	"DISABLED" *(default)* "ENABLED"	Training burst length of DQS setting. Enabled: Enable burst length of 16 Disabled: Enable burst length of 8

To support DDR4 memory interfaces, the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads, and the LPDDR3 internal VREF generator should be used to generated trained reference voltage for LPDDR4 IO buffers. DQSBUF_IVREF combines the functions together.

EBR_B

4 Kb pseudo-dual port block ram with configurable write/read width and optional bitstream initialization.

Architectures supported:

- iCE40 UltraPlus

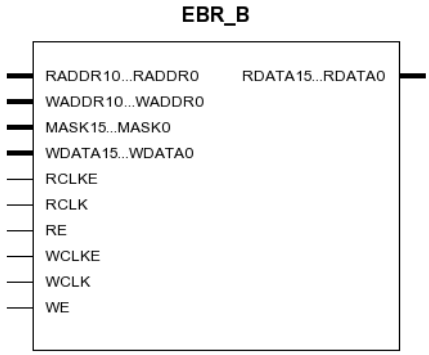


Table 197: Input Ports

Name	Description
RADDR10...RADDR0	Read address
WADDR10...WADDR0	Write address
MASK_N15...MASK_N0	Write mask
WDATA15...WDATA0	Write data
RCLKE	Read clock enable
RCLK	Read clock
RE	Read enable
WCLKE	Write clock enable
WCLK	Write clock
WE	Write enable

Table 198: Output Ports

Name	Description
RDATA15...RDATA0	Read data

Table 199: Parameters

Name	Values	Description
INIT_0	"0x00000000000000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_1	"0x00000000000000000000000000000000000000000000000000000000000000000000000000" *(default)*	

Table 199: Parameters (Continued)

Name	Values	Description
INIT_2	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_3	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_4	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_5	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_6	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_7	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_8	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_9	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_A	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_B	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_C	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_D	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_E	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
INIT_F	"0x0000000000000000000000000000000000000000000000000000000000000000" *(default)*	
DATA_WIDTH_W	"2" *(default)*, "4", "8", "16"	Write data width in bits.
DATA_WIDTH_R	"2" *(default)*, "4", "8", "16"	Read data width in bits.

Same as SB_RAM4K.

ECLKDIV

Wrapper for Clock Divider for Edge Clock

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



Table 200: Input Ports

Name	Description
ECLKIN	Clock signal after the DLLDEL
DIVRST	ECLKDIV reset
SLIP	Slip output

Table 201: Output Ports

Name	Description
DIVOUT	Output divided ECLK

Table 202: Parameters

Name	Values	Description
ECLK_DIV	"DISABLE" (default) "2" "3P5" "4" "5"	Clock Divider
GSR	"ENABLED" (default) "DISABLED"	Global set/reset

ECLKDIVA

Wrapper for Clock Divider for Edge Clock

Architectures supported:

- LAV-AT-ES1
- LN2-CT

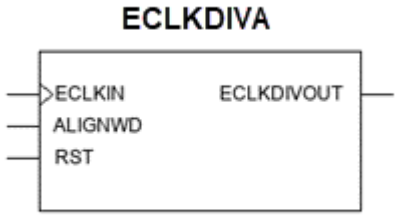


Table 203: Input Ports

Name	Description
ECLKIN	Clock signal after the DLLDEL
ALIGNWD	Align word function inside the ECLKDIV
RST	ECLKDIV reset

Table 204: Output Ports

Name	Description
ECLKDIVOUT	Output divided ECLK

Table 205: Parameters

Name	Values	Description
CLK_DIV	"2" (default) "3P5" "4" "5"	Clock Divider
ALIGN	"ENABLED" (default) "DISABLED"	Align
RATE	"HALF" (default) "FULL"	DDR Rate
PHYBY4_SEL	"ENABLED" (default) "DISABLED"	PHYCLKDIV by 4 enable
GSR	"ENABLED" (default) "DISABLED"	Global set/reset

ECLKSYNC

Wrapper for Clock Synchronizer for Edge Clock

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

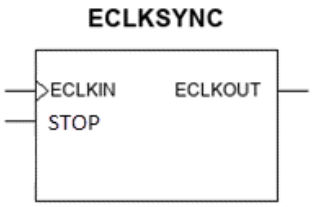


Table 206: Input Ports

Name	Description
ECLKIN	Clock signal after the DLLDEL
STOP	Control signal to stop the edge clock

Table 207: Output Ports

Name	Description
ECLKOUT	Output ECLK after the ECLKSYNC

Table 208: Parameters

Name	Values	Description
STOP_EN	"DISABLE" (default) "ENABLE"	Stop enable control

Note:

SYNC = 1 to SYNC = 0, it takes 4 clock cycles before ECLKIN appears in ECLKOUT.

SYNC = 0 to SYNC = 1, after 4 clock cycles, ECLKOUT goes to 0.

ECLKSYNCA

Wrapper for Clock Synchronizer for Edge Clock

Architectures supported:

- LAV-AT-ES1
- LN2-CT

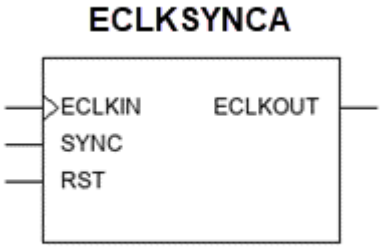


Table 209: Input Ports

Name	Description
ECLKIN	Clock signal after the DLLDEL
SYNC	ECLK sync function
RST	ECLKSYNC reset function

Table 210: Output Ports

Name	Description
ECLKOUT	Output ECLK after the ECLKSYNC

Note:

FA2

Carry chain 2-bit full adder

Architectures supported:

- iCE40 UltraPlus

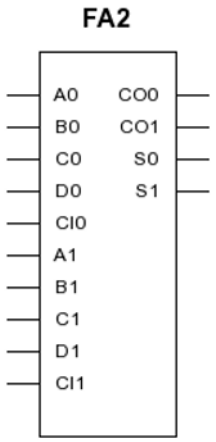


Table 211: Input Ports

Name	Description
A0	Non-operand A bit 0
B0	Operand B bit 0
C0	Operand C bit 0
D0	Operand D bit 0
CI0	Carry-in 0
A1	Non-operand A bit 1
B1	Operand B bit 1
C1	Operand C bit 1
D1	Operand D bit 1
CI1	Carry-in 1

Table 212: Output Ports

Name	Description
CO0	Carry-out 0
CO1	Carry-out 1
S0	Output bit 0 of the sum
S1	Output bit 1 of the sum

Table 213: Parameters

Name	Values	Description
INIT0	"0xc33c" *(default)*	Hexadecimal value.
INIT1	"0xc33c" *(default)*	Hexadecimal value.

Table 214: Truth Table

Inputs								Outputs			
B1	**C1**	**D1**	**BO**	**CO**	**DO**	**CI1**	**CI0**	**S1**	**SO**	**CO1**	**CO0**
0	0	=CI1	0	0	=CI0	=CO0	0	0	0	0	0
0	0	=CI1	0	0	=CI0	=CO0	1	0	1	0	0
0	0	=CI1	0	1	=CI0	=CO0	0	0	1	0	0
0	0	=CI1	0	1	=CI0	=CO0	1	1	0	0	1
0	1	=CI1	0	0	=CI0	=CO0	0	1	0	0	0
0	1	=CI1	0	0	=CI0	=CO0	1	1	1	0	0
0	1	=CI1	0	1	=CI0	=CO0	0	1	1	0	0
0	1	=CI1	0	1	=CI0	=CO0	1	0	0	1	1
0	0	=CI1	1	0	=CI0	=CO0	0	0	1	0	0
0	0	=CI1	1	0	=CI0	=CO0	1	1	0	0	1
0	0	=CI1	1	1	=CI0	=CO0	0	1	0	0	1
0	0	=CI1	1	1	=CI0	=CO0	1	1	1	0	1
0	1	=CI1	1	0	=CI0	=CO0	0	1	1	0	0
0	1	=CI1	1	0	=CI0	=CO0	1	0	0	1	1
0	1	=CI1	1	1	=CI0	=CO0	0	0	0	1	1
0	1	=CI1	1	1	=CI0	=CO0	1	0	1	1	1
1	0	=CI1	0	0	=CI0	=CO0	0	1	0	0	
1	0	=CI1	0	0	=CI0	=CO0	1	1	1	0	0
1	0	=CI1	0	1	=CI0	=CO0	0	1	1	0	0
1	0	=CI1	0	1	=CI0	=CO0	1	0	0	1	1
1	1	=CI1	0	0	=CI0	=CO0	0	0	0	1	0
1	1	=CI1	0	0	=CI0	=CO0	1	0	1	1	0
1	1	=CI1	0	1	=CI0	=CO0	0	0	1	1	0
1	1	=CI1	0	1	=CI0	=CO0	1	1	0	1	1
1	0	=CI1	1	0	=CI0	=CO0	0	1	1	0	0
1	0	=CI1	1	0	=CI0	=CO0	1	0	0	1	1
1	0	=CI1	1	1	=CI0	=CO0	0	0	0	1	1
1	0	=CI1	1	1	=CI0	=CO0	1	0	1	1	1
1	1	=CI1	1	0	=CI0	=CO0	0	0	1	1	0
1	1	=CI1	1	0	=CI0	=CO0	1	1	0	1	1

Table 214: Truth Table (Continued)

Inputs								Outputs			
B1	**C1**	**D1**	**BO**	**CO**	**DO**	**CI1**	**CI0**	**S1**	**SO**	**CO1**	**CO0**
1	1	=CI1	1	1	=CI0	=CO0	0	1	0	1	1
1	1	=CI1	1	1	=CI0	=CO0	1	1	1	1	1

Note:

In the Truth Table, ports A0 and A1 are tied low or are unconnected.

Rules and restrictions:

- Carry-out (CO0, CO1) can only go to D input of LUT above it or next (CI0, CI1).
- To drive VHI to the first carry-in, CI0, do not tie CI0 to 1 or VHI. Instead, leave it dangling. Bitgen will ensure that CI0 will correctly be VHI as expected.
- To drive VLO to the first carry-in, CI0 should be left dangling and half a FA2 must be used. This is done by grounding B0 and C0 and leaving D0 floating. This will ensure that CI1 will correctly be VLO as expected and the second half of FA2 can be used normally.
- If you want to bring out last carry-out (CO0, CO1) to fabric, simply connect the carry-out to whatever is your fabric logic. Map and Placer will work to insert a LUT and place it in the appropriate location automatically (thus ensuring the first rule is followed).

See Figure 2, Figure 3, and Figure 4, below, that take into account the above rules. The examples below do not make use of the A0 or A1 ports as they are not needed for 2-bit full-adder functionality.

Figure 2: Burning Half an FA2 for Carry-Chain

```
FA2 counter_12_add_4_1 (
   .B0(GND_net), .C0(GND_net),
   .B1(VCC_net), .C1(out_c_0),
   .D1(n192), .CI1(n192),
   .CO0(n192), .CO1(n78),
   .S1(n45)
);

FA2 counter_12_add_4_9 (
   .B0(GND_net), .C0(out_c_7),
   .D0(n84), .CI0(n84),
   .B1(GND_net), .C1(GND_net),
   .D1(n204), .CI1(n204),
   .CO0(n204), .S0(n38)
);

FA2 counter_12_add_4_7 (
   .B0(GND_net), .C0(out_c_5),
   .D0(n82), .CI0(n82),
   .B1(GND_net), .C1(out_c_6),
   .D1(n201), .CI1(n201),
   .CO0(n201), .CO1(n84),
   .S0(n40), .S1(n39)
);

FA2 counter_12_add_4_5 (
   .B0(GND_net), .C0(out_c_3),
   .D0(n80), .CI0(n80),
   .B1(GND_net), .C1(out_c_4),
   .D1(n198), .CI1(n198),
   .CO0(n198), .CO1(n82),
   .S0(n42), .S1(nn41)
);

FA2 counter_12_add_4_3 (
   .B0(GND_net), .C0(out_c_1),
   .D0(n78), .CI0(n78),
   .B1(GND_net), .C1(out_c_2),
   .D1(n195), .CI1(n195),
   .CO0(n195), .CO1(n80),
   .S0(n44), .S1(nn43)
);
```

Figure 3: Using LUT to Bring Last Carry Out

```
FA2 counter_12_add_4_7 (
    .B0(GND_net),
    .C0(out_c_5),
    .D0(n82),
    .CI0(n82),
    .B1(GND_net),
    .C1(out_c_6),
    .D1(n201),
    .CI1(n201),
    .CO0(n201),
    .CO1(LAST_CARRY),
    .S0(n40),
    .S1(n39)
);

LUT  lutInst(
    .D(LAST_CARRY),
    .F(CARRYOUT_TO_FABRIC)
); //eqn : F=D
```

Figure 4: Using FA2 to Bring Last Carry Out

```
FA2 counter_12_add_4_9 (
    .B0(GND_net),
    .C0(out_c_7),
    .D0(n84),
    .CI0(n84),
    .B1(GND_net),
    .C1(GND_net),
    .S1(CARRYOUT_TO_FABRIC),
    .D1(lastCarryOut),
    .CI1(lastCarryOut),
    .CO0(lastCarryOut),
    .S0(n38)
);
```

FD1P3BX

Positive-Edge-Triggered D Flip-Flop with Positive-Level Enable and Positive-Level Asynchronous Preset

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C

- UT24CP

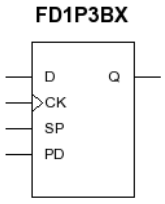


Table 215: Input Ports

Name	Description
D	Data in
SP	Clock enable
CK	Clock
PD	Preset

Table 216: Output Ports

Name	Description
Q	Data out

Table 217: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset

Table 218: Truth Table

Inputs				Output
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)

FD1P3BZ

Positive-Edge-Triggered D Flip-Flop with Positive-Level Enable and Positive-Level Asynchronous Preset

Architectures supported:

- iCE40 UltraPlus

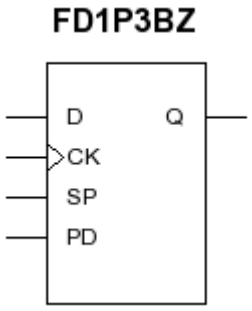


Table 219: Input Ports

Name	Description
D	Data in
CK	Clock
SP	Clock enable, active high
PD	Preset, active high

Table 220: Output Ports

Name	Description
Q	Data out

Table 221: Truth Table

Inputs				Output
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

FD1P3DX

Positive-Edge-Triggered D Flip-Flop with Positive-Level Enable and Positive-Level Asynchronous Clear

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

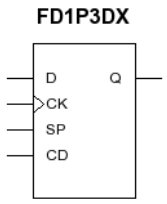


Table 222: Input Ports

Name	Description
D	Data in
SP	Clock enable
CK	Clock
CD	Clear

Table 223: Output Ports

Name	Description
Q	Data out

Table 224: Parameters

Name	Values	Description
GSR	ENABLED *(default)* DISABLED	Enable global set/reset.

Table 225: Truth Table

Inputs				Output
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)

FD1P3DZ

Positive-Edge-Triggered D Flip-flop with Positive-Level Enable and Positive-Level Asynchronous Clear

Architectures supported:

- iCE40 UltraPlus

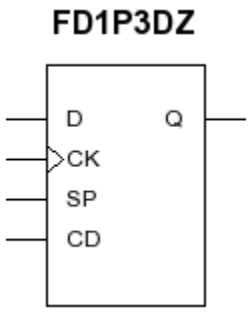


Table 226: Input Ports

Name	Description
D	Data in
SP	Clock enable
CK	Clock
CD	Clear

Table 227: Output Ports

Name	Description
Q	Data out

Table 228: Truth Table

Inputs				Output
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

FD1P3IX

Positive-Edge-Triggered D Flip-Flop with Positive-Level Synchronous Clear and Positive-Level Enable

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

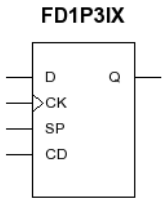


Table 229: Input Ports

Name	Description
D	Data in
SP	Clock enable

Table 229: Input Ports (Continued)

Name	Description
CK	Clock
CD	Clear, active high. Clock enable must be active to clear.

Table 230: Output Ports

Name	Description
Q	Data out

Table 231: Parameters

Name	Values	Description
GSR	ENABLED *(default)* DISABLED	Enable global set/reset.

Table 232: Truth Table

Inputs				Output
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)

FD1P3IZ

Positive-Edge-Triggered D Flip-Flop with Positive-Level Synchronous Clear and Positive-Level Enable

Architectures supported:

- iCE40 UltraPlus

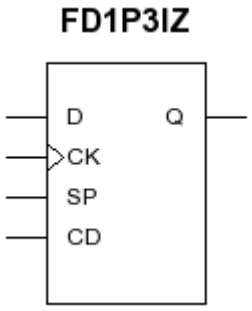


Table 233: Input Ports

Name	Description
D	Data in
SP	Clock enable, active high
CK	Clock
CD	Clear, active high. Clock enable must be active to clear.

Table 234: Output Ports

Name	Description
Q	Data out

Table 235: Truth Table

Inputs				Output
D	**SP**	**CK**	**CD**	**Q**
X	0	X	0	Q
X	1	↑	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don't care

Clear overrides enable.

FD1P3JX

Positive-Edge-Triggered D Flip-Flop with Positive-Level Synchronous Preset and Positive-Level Enable

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

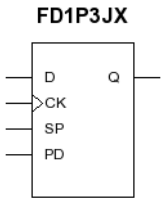


Table 236: Input Ports

Name	Description
D	Data in
SP	Clock enable
CK	Clock
PD	Preset

Table 237: Output Ports

Name	Description
Q	Data out

Table 238: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset

Table 239: Truth Table

Inputs				Output
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)

Preset overrides enable.

FD1P3JZ

Positive-Edge-triggered D Flip-flop with Positive-level Synchronous Preset and Positive-level Enable

Architectures supported:

- iCE40 UltraPlus

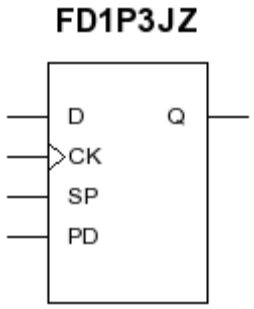


Table 240: Input Ports

Name	Description
D	Data in
SP	Clock enable, active high
CK	Clock
PD	Preset, active high. Clock enable must be active to preset.

Table 241: Output Ports

Name	Description
Q	Data out

Table 242: FD1P3JZ Truth Table

Inputs				Output
D	**SP**	**CK**	**PD**	**Q**
X	0	X	0	Q
X	1	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don't care

Preset overrides enable.

FD1P3XZ

Positive-Edge-triggered D Flip-flop with Synchronous or Asynchronous Set/ reset

Architectures supported:

- iCE40 UltraPlus

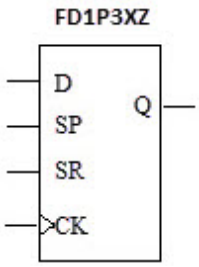


Table 243: Input Ports

Name	Description
D	Data in
SP	Clock enable, active high
CK	Clock
SR	Set/reset, active high

Table 244: Output Ports

Name	Description
Q	Data out

Table 245: Parameters

Name	Values
REGSET	"RESET" (*default*), "SET"
SRMODE	"CE_OVER_LSR" (*default*), "ASYNC"

FIFO16K

16 Kb FIFO using block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

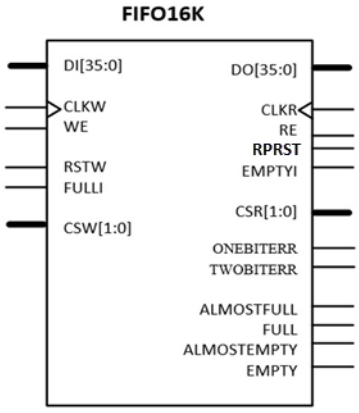


Table 246: Input Ports

Name	Description
DI[35:0]	Input data.
CLKW	Write clock. Active on the rising edge. Has polarity control.
WE	Write enable. Active high. Has polarity control.
FULLI	Write inhibit. Active high. Has polarity control.
CSW[1:0]	Write cascade select. Active high. Has polarity control.
RSTW	Write reset. Active high. Has polarity control.
CLKR	Read clock. Active on the rising edge. Has polarity control.
RE	Read enable. Active high. Has polarity control.
EMPTYI	Read inhibit. Active high. Has polarity control.
CSR[1:0]	Read cascade select. Active high. Has polarity control.
RPRST	Read point reset. Active high. Has polarity control.

Table 247: Output Ports

Name	Description
DO[35:0]	Output data.
ONEBITERR	ECC error flag for a 1-bit error.
TWOBITERR	ECC error flag for a 2-bit error.
ALMOSTFULL	Almost full flag.
FULL	Full flag.

Table 247: Output Ports (Continued)

Name	Description
ALMOSTEMPTY	Almost empty flag.
EMPTY	Empty flag.

Table 248: Parameters

Name	Values	Description
DATA_WIDTH_W	"X36" *(default)* "X32" "X18" "X9" "X4" "X2" "X1"	
DATA_WIDTH_R	"X36" *(default)* "X32" "X18" "X9" "X4" "X2" "X1"	
OUTREG	"BYPASSED" *(default)* "USED"	
GSR	"ENABLED" *(default)* "DISABLED"	
RESETMODE	"SYNC" *(default)* "ASYNC"	
ASYNC_RST_RELEASE	"SYNC" *(default)* "ASYNC"	
ECC	"DISABLED" *(default)* "ENABLED"	
CSDECODE_W	00 *(default)* 01 10 11	
CSDECODE_R	00 *(default)* 01 10 11	
FULLBITS	0b10000000000000 *(default)*	Specifies the FIFO size for FULL flag. By default, it is 18k.
ALMOST_FULL	0b00000000000000 *(default)*	The value must be less than FULLBITS.
ALMOST_EMPTY	0b00000000000000 *(default)*	The value must be greater than 0.

FIFO32K

32 Kb FIFO using block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

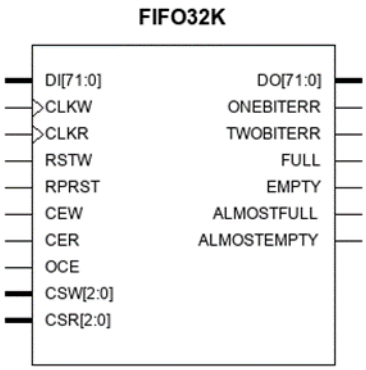


Table 249: Input Ports

Name	Range	Type	Active State	Description
DI	71:0	Data	Active High	Data Inputs
CLKW	1	Clock	Rising Edge	Write Clock Input
CLKR	1	Clock	Rising Edge	Read Clock Input
RSTW	1	Control	Active High	Write Reset
RPRST	1	Control	Active High	Read point reset
CEW	1	Control	Active High	Write Clock Enable
CER	1	Control	Active High	Read Clock Enable
OCE	1	Control	Active High	Output Register Clock Enable
CSW	2:0	Control	Active High	Write Cascade Select
CSR	2:0	Control	Active High	Read Cascade Select

Table 250: Output Ports

Name	Range	Type	Active State	Description
DO	71:0	Data	Active High	Data Output
ONEBITERR	1	Data	Active High	Single Bit Error Flag

Table 250: Output Ports (Continued)

Name	Range	Type	Active State	Description
TWOBITERR	1	Data	Active High	Double Bit Error Flag
FULL	1	Data	Active High	Full Flag
EMPTY	1	Data	Active High	Empty Flag
ALMOSTFULL	1	Data	Active High	Almost Full Flag
ALMOSTEMPTY	1	Data	Active High	Almost Empty Flag

Table 251: Parameters

Name	Values	Description
ALMOST_EMPTY	"0b000000000000000"(default)	The value must be greater than 0
ALMOST_FULL	"0b000000000000000"(default)	The value must be less than FULLBITS
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release sync/async control
CSDECODE_R	"000"	Read chip select active setting, 0=active high 1=active low
CSDECODE_W	"000"	Write chip select active level, 0=active high 1=active low
DATA_WIDTH_R	"X72" (default) "X64" "X36" "X18" "X9" "X4" "X2" "X1"	Read port width
DATA_WIDTH_W	"X72" (default) "X64" "X36" "X18" "X9" "X4" "X2" "X1"	Write port width
ECC	"DISABLED" (default) "ENABLED	Enable ECC

Table 251: Parameters (Continued)

Name	Values	Description
FULLBITS	"0b100000000000000"(default)	Specifies the FIFO size for FULL flag; by default FIFO size is 36K
FWFT	"DISABLED" (default)	First Word Fall Through Mode
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
OUTREG	"BYPASSED" (default) "USED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control

FILTER

Adds a 50-ns delay to a signal

Architectures supported:

- iCE40 UltraPlus

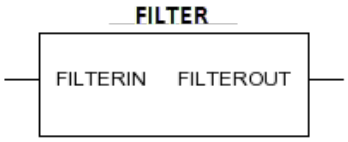


Table 252: Input Ports

Name	Description
FILTERIN	Filter input

Table 253: Output Ports

Name	Description
FILTEROUT	Filter output

GSR

Wrapper for Global Set/Reset

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5

- LIFCL
- UT24C
- UT24CP

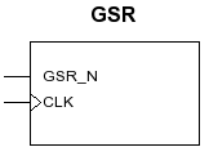


Table 254: Input Ports

Name	Description
GSR_N	User GSR input
CLK	User clock for sync mode

Table 255: Parameters

Name	Values	Description
SYNCMODE	"ASYNC" *(default)* "SYNC"	

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. When input GSR is HIGH, the global signal is released at the positive edge of the clock (CLK) if SYNCMODE is SYNC; otherwise, the signal is released asynchronously.

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 primitive description.

In general, The GSR instantiation in testbench is required when RTL design is composed of Lattice Components/Primitives that requires a GSR net connection in its simulation model. Some of examples of these components are IO/DDR, Registers, DSP, and Hard Ips (like PCIe, SGMII, DPHY). It is also required if a Netlist simulation (post-synthesis and post-route) is performed as it contains synthesized and translated primitives from the User RTL. The best place to add the GSR is the testbench, do not instantiate the GSR primitive in the RTL unless it is needed.

GSRA

Global Set/Reset

Architectures supported:

- LAV-AT-ES1
- LN2-CT

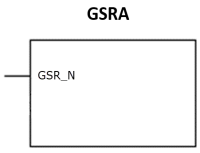


Table 256: Input Ports

Name	Description
GSR_N	User GSR input, asynchronous

Note:

The GSRA primitive needs to be instantiated in the user design.

HSOSC

High-frequency oscillator. Generates 48-MHz nominal clock, +/- 10 percent, with user-programmable divider. Can drive global clock network or fabric routing.

Architectures supported:

- iCE40 UltraPlus

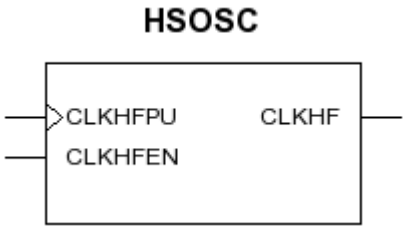


Table 257: Input Ports

Name	Description
CLKHFPU	Power up the oscillator. After power up, output will be stable after 100 µs. Active high.
CLKHFEN	Enable the clock output. Enable should be low for the 100-µs power-up period. Active high.

Table 258: Output Ports

Name	Description
CLKHF	Oscillator output

Table 259: Parameters

Name	Values	Description
CLKHF_DIV	"0'b00" (*default*) "0'b01" "0'b10" "0'b11"	Clock divider selection: 0'b00 = 48 MHz 0'b01 = 24 MHz 0'b10 = 12 MHz 0'b11 = 6 MHz

Same as: SB_HFOSC

HSOSC1P8V

High-frequency oscillator. Generates 48-MHz nominal clock, +/- 10 percent, with user-programmable divider. Can drive global clock network or fabric routing. For use when VPP_2V5 is connected to a voltage below 2.3 V.

Architectures supported:

- iCE40 UltraPlus

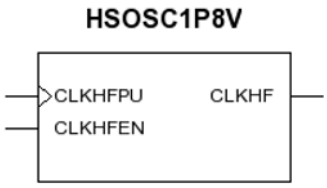


Table 260: Input Ports

Name	Description
CLKHFPU	Power up the oscillator. After power up, output will be stable after 100 µs. Active high
CLKHFEN	Enable the clock output. Enable should be low for the 100-µs power up period. Active high

Table 261: Output Ports

Name	Description
CLKHF	Oscillator output

Table 262: Parameters

Name	Values	Description
CLKHF_DIV	0’b00" (default), "0’b01", "0’b10", "0’b11	Clock divider selection. 0’b00 = 48 MHz, 0’b01 = 24 MHz, 0’b10 = 12 MHz, 0’b11 = 6 MHz

HSOSC_CORE

High-Frequency Oscillator

Generates 48MHz nominal clock, +/- 10 percent, with user-programmable divider. Can drive global clock network or fabric routing.

Architectures supported:

- iCE40 UltraPlus

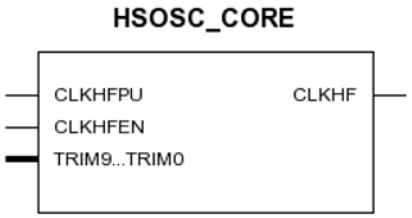


Table 263: Input Ports

Name	Description
CLKHFPU	Power up the oscillator. After power up, output will be stable after 100us. Active high
CLKHFEN	Enable the clock output. Enable should be low for the 100us power up period. Active high
TRIM9...TRIM0	

Table 264: Output Ports

Name	Description
CLKHF	Oscillator output

Table 265: Parameters

Name	Values	Description
CLKHF_DIV	0’b00" (default), "0’b01", "0’b10", "0’b11	Clock divider selection. 0’b00 = 48 MHz, 0’b01 = 24 MHz, 0’b10 = 12 MHz, 0’b11 = 6 MHz
FABRIC_TRIME	"DISABLE" (default), "ENABLE"	Trim bits source selection. ENABLE - Trim bits sourced from Fabric. DISABLE - Trim bits sourced from NVCM.

I2C_B

Hard I2C interface

Architectures supported:

- iCE40 UltraPlus

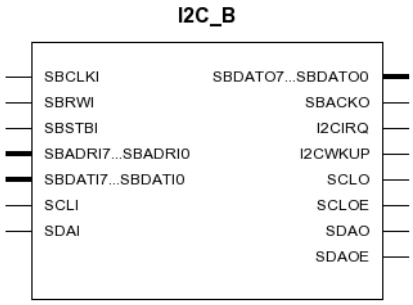


Table 266: Input Ports

Name	Description
SBCLKI	System clock input
SBRWI	System read/write input
SBSTBI	Strobe signal
SBADRI7	SBADRI0 (System bus control register address)
SBDATI7	SBDATI0 (System data input)
SCLI	Serial clock input
SDAI	Serial data input

Table 267: Output Ports

Name	Description
SBDATO7...SBDATO0	System data output
SBACKO	System acknowledgment
I2CIRQ	I2C interrupt output
I2CWKUP	I2C wakeup from standby signal
SCLO	Serial clock output
SCLOE	Serial clock output enable, active high
SDAO	Serial data output
SDAOE	Serial data output enable, active high

Table 268: Parameters

Name	Values	Description
I2C_SLAVE_INIT_ADDR	"0'b1111100001" (default), "0'b1111100010"	Upper bits [9:2] can be changed through control registers. Lower bits [1:0] are fixed to 01 in the upper left and 10 in the upper right
BUS_ADDR74	"0'b0001" (default), "0'b0011"	Fixed value. Upper left corner should be 0'b0001, upper right corner should be 0'b0011. SBADRI[7:4] should match this value to activate the IP
I2C_CLK_DIVIDER	"0" (default), "1...1023"	Ratio of SBCLKI/SCLO
SDA_INPUT_DELAYED	"0" (default), "1"	Add 50-ns delay to the SDAI signal
SDA_OUTPUT_DELAYED	"0" (default), "1"	Add 50-ns delay to the SDAO signal
FREQUENCY_PIN_SBCLKI	"NONE" (default)	SDC constraint entry on SBCLKI port of I2C.

SAME AS: SB_I2C

I2CFIFO

Wrapper for Hardened I2C Interface Block

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

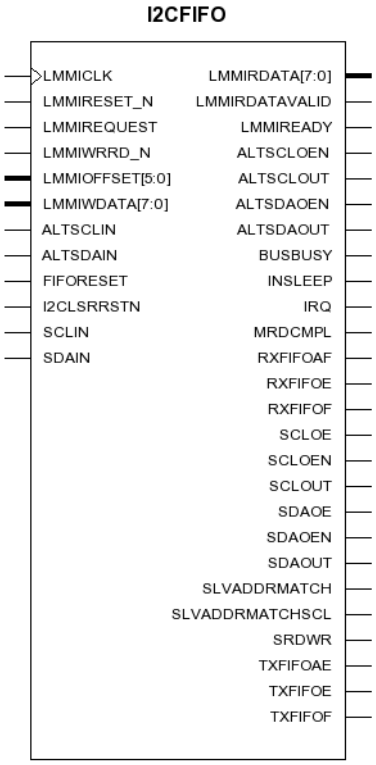


Table 269: Input Ports

Name	Description
LMMICLK	
LMMIRESET_N	
LMMIREQUEST	
LMMIWRRD_N	
LMMIOFFSET[5:0]	
LMMIWDATA[7:0]	
ALTSCLIN	
ALTSDAIN	
FIFORESET	
I2CLSRRSTN	
SCLIN	
SDAIN	

Table 270: Output Ports

Name	Description
LMMIRDATA[7:0]	
LMMIRDATAVALID	
LMMIREADY	
ALTSCLOEN	
ALTSCLOUT	
ALTSDAOEN	
ALTSDAOUT	
BUSBUSY	
INSLEEP	
IRQ	
MRDCMPL	
RXFIFOAF	
RXFIFOE	
RXFIFOF	
SCLOE	
SCLOEN	
SCLOUT	
SDAOE	
SDAOEN	
SDAOUT	
SLVADDRMATCH	
SLVADDRMATCHSCL	
SRDWR	
TXFIFOAE	
TXFIFOE	
TXFIFOF	

Table 271: I2CFIFO Parameters

Name	Values	Description
BRNBASEDELAY	0'b0000 *(default)*	
CR1CKDIS	"EN" *(default)* "DIS"	
CR1FIFOMODE	"REG" *(default)* "FIFO"	
CR1GCEN	"DIS" *(default)* "EN"	
CR1I2CEN	"DIS" *(default)* "EN"	
CR1SDADELSEL	"NDLY0" *(default)* "NDLY1" "NDLY2" "NDLY4"	
CR1SLPCLKEN	"DIS" *(default)* "EN"	
CR2CORERSTN	"DIS" *(default)* "RST"	
CR2HARDTIE	"TIE" *(default)* "NOTIE"	
CR2INTCLREN	"DIS" *(default)* "EN"	
CR2MRDCMPLWKUP	"DIS" *(default)* "EN"	
CR2RXFIFOAFWKUP	"DIS" *(default)* "EN"	
CR2SLVADDRWKUP	"DIS" *(default)* "EN"	
GSR	"ENABLED" *(default)* "DISABLED"	
I2CRXFIFOAFVAL	0'b00000 *(default)*	
I2CSLVADDRA	0'b0000000000 *(default)*	
I2CTXFIFOAEVAL	0'b0000 *(default)*	
INTARBLIE	"DIS" *(default)* "EN"	
INTBUSFREEIE	"DIS" *(default)* "EN"	
INTHGCIE	"DIS" *(default)* "EN"	

Table 271: I2CFIFO Parameters (Continued)

Name	Values	Description
INTMRDCMPLIE	"DIS" *(default)* "EN"	
INTRNACKIEORRSVD	"DIS" *(default)* "EN"	
INTRSVDORTROEIE	"DIS" *(default)* "EN"	
INTRSVDORTRRDYIE	"DIS" *(default)* "EN"	
INTRXOVERFIEORRSVD	"DIS" *(default)* "EN"	
INTRXUNDERFIE	"DIS" *(default)* "EN"	
INTTXOVERFIE	"DIS" *(default)* "EN"	
INTTXSERRIEORRSVD	"DIS" *(default)* "EN"	
LMMI_EXTRA_ONE	"DIS" *(default)* "EN"	
LMMI_EXTRA_TWO	"DIS" *(default)* "EN"	
NCRALTIOEN	"FABRIC" *(default)* "IO"	
NCRFILTERDIS	"EN" *(default)* "DIS"	
NCRSDAINDLYEN	"DIS" *(default)* "EN"	
NCRSDAOUTDLYEN	"DIS" *(default)* "EN"	
NONUSRTESTSOFTTRIMEN	"DIS" *(default)* "EN"	
NONUSRTSTSOFTTRIMVALUE	0’b000 *(default)*	
REGI2CBR	0’b0000000000 *(default)*	
TSPTIMERVALUE	0’b10010010111 *(default)* 00000000000	

IB

Input Buffer

Architectures supported:

- iCE40 UltraPlus

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

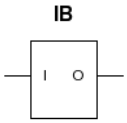


Table 272: Input Ports

Name	Description
I	Data from pad

Table 273: Output Ports

Name	Description
O	Data to fabric

In iCE40 UtraPlus, refer to Source Template.

Table 274: IB Truth Table

Input	Output
I	O
1	1
0	0
Z	U

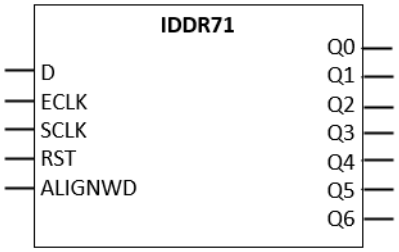
IDDR71

7:1 LVDS IDDR

Architectures supported:

- LFCPNX

- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for 7:1 LVDS input implementation.

Table 275: Input Ports

Name	Description
D	DDR data input
ECLK	Fast edge clock
SCLK	Primary clock input (divide-by-3.5 of ECLK)
RST	Reset to DDR registers
ALIGNWD	Shifts word alignment by one bit

Table 276: Output Ports

Name	Description
Q[6:0]	Output data

Table 277: IDDR71 Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDR71A

7:1 LVDS input side implementation

Architectures supported:

- LAV-AT-ES1
- LN2-CT

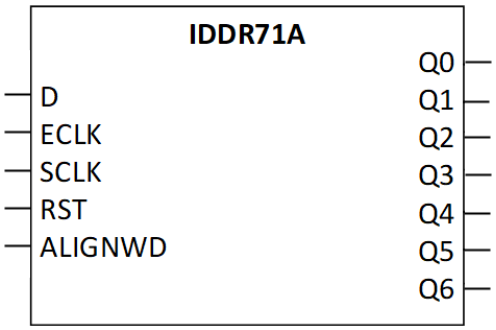


Table 278: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-3.5 of ECLK)
RST	1	Control	High	Reset to DDR registers
ALIGNWD	1	Control	High	This signal is used for word alignment. It will shift the word by one bit.

Table 279: Output Ports

Name	Range	Type	Active State	Description
Q0, Q2, Q4, Q6	4	Data	High	Data at the positive edge of ECLK
Q1, Q3, Q5	3	Data	High	Data at the negative edge of ECLK

Table 280: Parameters

Name	Value	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

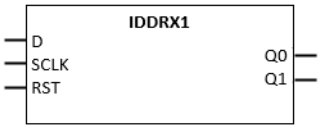
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX1

Generic X1 IDDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for the Generic x1 IDDR implementation.

Table 281: Input Ports

Name	Description
D	DDR data input
SCLK	Primary clock input
RST	Reset to DDR registers

Table 282: Output Ports

Name	Description
Q0	Data at the positive edge of the clock
Q1	Data at the negative edge of the clock

Table 283: IDDRX1 Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

IDDRX1A

Generic DDR with 1X gearing

Architectures supported:

- LAV-AT-ES1
- LN2-CT

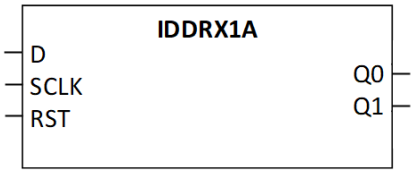


Table 284: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	DDR data input
SCLK	1	Clock	Rising Edge	Primary clock input
RST	1	Control	High	Reset to DDR registers

Table 285: Output Ports

Name	Range	Type	Active State	Description
Q0	1	Data	High	Data at the positive edge of the clock
Q1	1	Data	High	Data at the negative edge of the clock

Table 286: Parameters

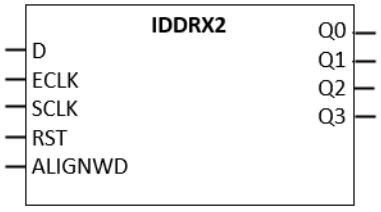
Name	Value	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

IDDRX2

Generic X2 IDDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for the generic X2 IDDR implementation.

Table 287: Input Ports

Name	Range	Description
D		DDR data input.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset to DDR registers.
ECLK		Fast edge clock.
ALIGNWD		Shifts word alignment by one bit.

Table 288: Output Ports

Name	Range	Description
Q	3:0	Parallel data output. Q0, Q2 are the data at the positive edges of the input ECLK. Q1 and Q3 are the data at negative edge of input ECLK.

Table 289: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX2A

Generic DDR with 2X gearing

Architectures supported:

- LAV-AT-ES1
- LN2-CT

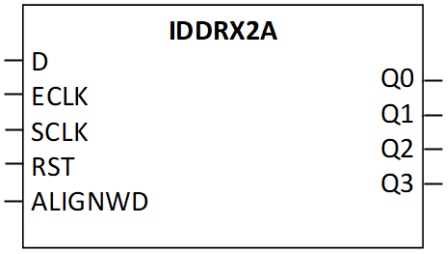


Table 290: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-2 of ECLK)
RST	1	Control	High	Reset to DDR registers
ALIGNWD	1	Control	High	This signal is used for word alignment. It will shift the word by one bit.

Table 291: Output Ports

Name	Range	Type	Active State	Description
Q0, Q2	2	Data	High	Data at the positive edge of ECLK
Q1, Q3	2	Data	High	Data at the negative edge of ECLK

Table 292: Parameters

Name	Value	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

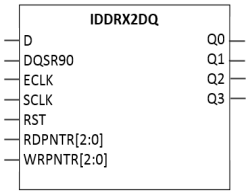
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX2DQ

DQ Input for DDR2 and DDR3 Memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to implement DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces.

Table 293: Input Ports

Name	Range	Description
D		DDR data input.
DQSR90		DQS clock input.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset to DDR registers.
RDPNTR	2:0	Read pointer from the DQSBUF module used to transfer data to ECLK.
WRPNTR	2:0	Write pointer from the DQSBUF module used to transfer data to ECLK.

Table 294: Output Ports

Name	Range	Description
Q	3:0	Parallel data output. Q0 and Q2 are the data at the positive edges of the input ECLK. Q1 and Q3 are the data at negative edge of input ECLK.

Table 295: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

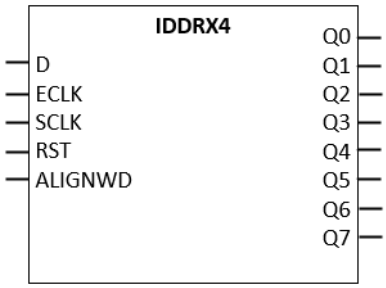
IDDRX4

Generic X4 IDDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL

- UT24C
- UT24CP



This primitive is used for the Generic x4 IDDR implementation.

Table 296: Input Ports

Name	Range	Description
D		DDR data input.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset to DDR registers.
ECLK		Fast edge clock.
ALIGNWD		Shifts word alignment by one bit.

Table 297: Output Ports

Name	Range	Description
Q	7:0	Parallel data output. Q0, Q2, Q4, Q6 are the data at the positive edges of the input ECLK. Q1, Q3, Q5, Q7 are data at negative edge of input ECLK

Table 298: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX4A

Generic DDR with 4X gearing

Architectures supported:

- LAV-AT-ES1
- LN2-CT

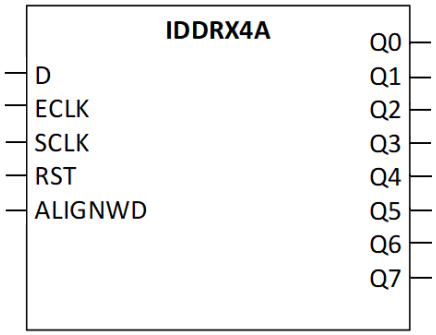


Table 299: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-4 of ECLK)
RST	1	Control	High	Reset to DDR registers
ALIGNWD	1	Control	High	This signal is used for word alignment. It will shift the word by one bit.

Table 300: Output Ports

Name	Range	Type	Active State	Description
Q0, Q2, Q4, Q6	4	Data	High	Data at the positive edge of ECLK
Q1, Q3, Q5, Q7	4	Data	High	Data at the negative edge of ECLK

Table 301: Parameters

Name	Value	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

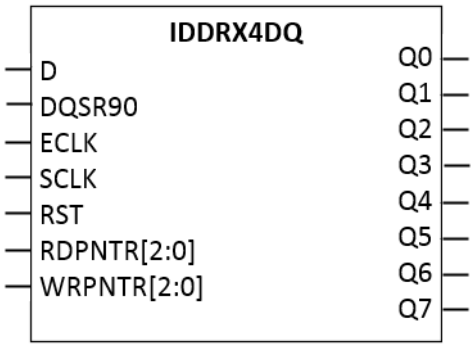
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX4DQ

DQ Input for DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to implement DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces at higher speeds.

Table 302: Input Ports

Name	Range	Description
D		DDR data input.
DQSR90		DQS clock input.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset to DDR registers.
RDPNTR	2:0	Read pointer from the DQSBUF module used to transfer data to ECLK.
WRPNTR	2:0	Write pointer from the DQSBUF module used to transfer data to ECLK.

Table 303: Output Ports

Name	Range	Description
Q	7:0	Parallel data output. Q0, Q2, Q4, Q6 are the data at the positive edges of the input ECLK. Q1, Q3, Q5, Q7 are data at negative edge of input ECLK.

Table 304: Parameters

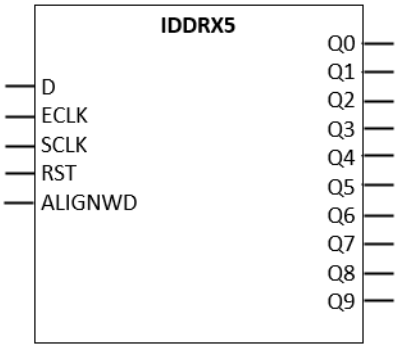
Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

IDDRX5

Generic X5 IDDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for the Generic x5 IDDR implementation.

Table 305: Input Ports

Name	Range	Description
D		DDR data input.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-5 of ECLK).
RST		Reset to DDR registers.
ALIGNWD		Shifts word alignment by one bit.

Table 306: Output Ports

Name	Range	Description
Q	9:0	Parallel data output. Q0, Q2, Q4, Q6, Q8 are the data at the positive edges of the input ECLK. Q1, Q3, Q5, Q7, Q9 are data at negative edge of input ECLK.

Table 307: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	"ENABLED"

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IDDRX5A

Generic DDR with 5X gearing

Architectures supported:

- LAV-AT-ES1
- LN2-CT

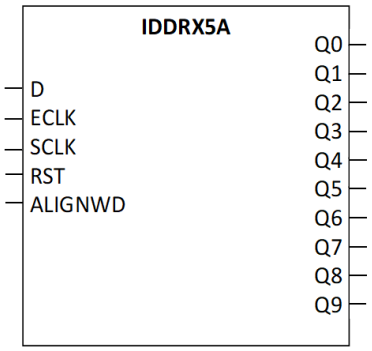


Table 308: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-5 of ECLK)
RST	1	Control	High	Reset to DDR registers
ALIGNWD	1	Control	High	This signal is used for word alignment. It will shift the word by one bit.

Table 309: Output Ports

Name	Range	Type	Active State	Description
Q0, Q2, Q4, Q6, Q8	5	Data	High	Data at the positive edge of ECLK
Q1, Q3, Q5, Q7, Q9	5	Data	High	Data at the negative edge of ECLK

Table 310: Parameters

Name	Value	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

IFD1P3AZ

Positive Edge Triggered Input D Flip-Flop with Positive Level Enable

Architectures supported:

- iCE40 UltraPlus

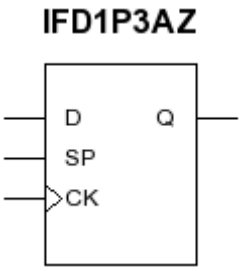


Table 311: Input Ports

Name	Range	Description
D		Data in
SP		Clock enable, active high
CK		Clock

Table 312: Output Ports

Name	Range	Description
Q		Data out

Table 313: Truth Table

Inputs			Output
D	SP	CK	Q
X	0	X	Q
0	1	↑	0
1	1	↑	1

X = Don't care

IFD1P3BX

Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

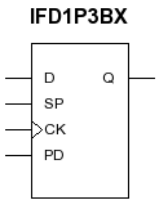


Table 314: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
CK	1	Clock	Rising Edge	Clock Input

Table 314: Input Ports

Name	Range	Type	Active State	Description
SP	1	Control	High	Positive Clock Enable
PD	1	Control	High	Asynchronous Preset

Table 315: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data Output

Table 316: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable or disable GSR feature

Note:

Any IO driving an IO register may **only** drive that IO register and nothing else.

Table 317: IFD1P3BX Truth Table

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)

IFD1P3DX

Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX

- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

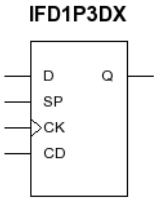


Table 318: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
CK	1	Clock	Rising Edge	Clock Input
SP	1	Control	High	Positive Clock Enable
CD	1	Control	High	Asynchronous Reset

Table 319: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data Output

Table 320: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable or disable GSR feature

Note:

Any IO driving an IO register may **only** drive that IO register and nothing else.

Table 321: IFD1P3DX Truth Table

Inputs				Output
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)

IFD1P3IX

Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear Overrides Enable)

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

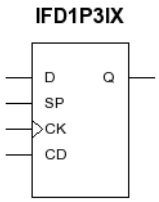


Table 322: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
CK	1	Clock	Rising Edge	Clock Input

Table 322: Input Ports

Name	Range	Type	Active State	Description
SP	1	Control	High	Positive Clock Enable
CD	1	Control	High	Synchronous Reset

Table 323: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data Output

Table 324: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable or disable GSR feature

Note:

Any IO driving an IO register may **only** drive that IO register and nothing else.

Table 325: IFD1P3IX Truth Table

Inputs				Output
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)

IFD1P3JX

Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset Overrides Enable)

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

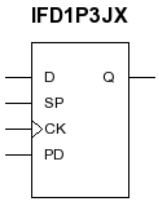


Table 326: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
CK	1	Clock	Rising Edge	Clock Input
SP	1	Control	High	Positive Clock Enable
PD	1	Control	High	Synchronous Preset

Table 327: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data Output

Table 328: Parameters

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable or disable GSR feature

Note:

Any IO driving an IO register may **only** drive that IO register and nothing else.

Table 329: IFD1P3JX Truth Table

Inputs				Output
D	SP	CK	PD	Q
X	0	X	0	Q
X	X	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

When GSR = 0, Q = 1 (D = SP = CK = PD = X)

INV

Inverter

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

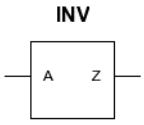


Table 330: Input Ports

Name	Description
A	

Table 331: Output Ports

Name	Description
Z	

IMONDELAY

ECLK interface with dynamically control delay and edge monitor (IMON)

Architectures supported:

- LAV-AT-ES1
- LN2-CT

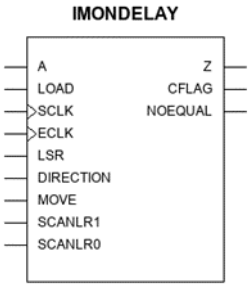


Table 332: Input Ports

Name	Range	Type	Active State	Description
A	1	Data	High	Data input from pin
SCLK	1	Clock	High	Primary clock for IREG/IDDR
ECLK	1	Clock	Rising Edge	Fast Edge Clock
LSR	1	Reset	High	Local Set/Reset
LOAD	1	Control	High	

Table 332: Input Ports (Continued)

Name	Range	Type	Active State	Description
DIRECTION	1	Control	High	DIRECTION 1 to decrease delay 0 to increase delay
MOVE	1	Control	High	
SCANLR1	1	Control	High	Edge Monitor Window Adjustment
SCANLR0	1	Control	High	Edge Monitor Window Adjustment

Table 333: Output Ports

Name	Range	Type	Active State	Description
Z	1	Data	High	Delayed data to input register block
CFLAG	1	Flag	High	Flag indicating the delay counter has reached max (when moving up) or min (when moving down) value
NOEQUAL	1	Flag	High	Equal Flag

Table 334: Parameters

Name	Value	Description
DEL_MODE	USER_DEFINED ECLK_ALIGNED ECLK_CENTERED	Set delay mode
DEL_VALUE	"0" to "511"	Static Delay Value
DEL_ADJUST	"0" to "15"	Static Delay Window Adjustment

IOL_B

Input/Output Registers, for Both Single Data and Double Data Rate

Architectures supported:

- iCE40 UltraPlus

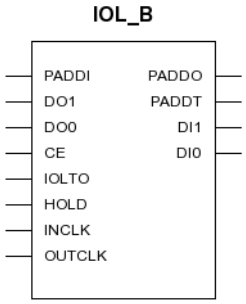


Table 335: Input Ports

Name	Description
PADDI	Data from pad
DO1	Data to pad
DO0	Data to pad
CE	Clock enable
IOLTO	Tri-state enable (active low)
HOLD	Hold DI0 state
INCLK	Input clock
OUTCLK	Output clock

Table 336: Output Ports

Name	Description
PADDO	Data to pad
PADDT	Tri-state control to pad
DI1	Data from pad
DI0	Data from pad

Table 337: Parameters

Name	Values	Description
LATCHIN	"NONE_REG" (default), "LATCH_REG", "LATCH_BYPASS", “NONE DDR”	NONE REG = no latch, just register, LATCH_REG = latch DI0 when HOLD is asserted, LATCH_BYPASS = latch non-registered DO0, and NONE DDR = No latch, DDR operation.
DDROUT	"NO" (default), "YES"	Select double data rate output

JTAG

JTAG Block for Reveal Debugging Tool

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

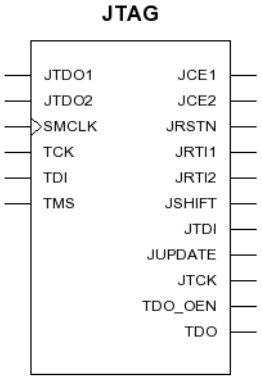


Table 338: Input Ports

Name	Description
JTDO1	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.
JTDO2	JTAG Test Data Output two
SMCLK	Configuration Block Clock
TCK	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.
TDI	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.
TMS	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.

Table 339: Output Ports

Name	Description
JCE1	JTAG Clock Enable one (BS is to boundary scan ring (L2).
JCE2	JTAG Clock Enable two
JRSTN	JTAG ResetN Active low reset signal. Set low initially and then remains high after initialization.
JRTI1	JTAG Run-Test/Idle one
JRTI2	JTAG Run-Test/Idle two
JSHIFT	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JTDI	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.

Table 339: Output Ports (Continued)

Name	Description
JUPDATE	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JTCK	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.
TDO_OEN	TAP Data Output Enable (OE to TDO)
TDO	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 340: JTAG Parameters

Name	Values	Description
MCER1EXIST	"NEXIST" (default) "EXIST"	“DISABLED”: TDO is driven by TDI (bypass register) when the IP_A or IPTEST_A instruction is sent. “ENABLED”: TDO is driven by JTDO1 (ER1 register) when the IP_A or IPTEST_A instruction is sent.
MCER2EXIST	"NEXIST" (default) "EXIST"	“DISABLED”: TDO is driven by TDI (bypass register) when the IP_B or IPTEST_B instruction is sent. “ENABLED”: TDO is driven by JTDO2 (ER2 register) when the IP_B or IPTEST_B instruction is sent.

JTAGA

Supports JTAG operations

Architectures supported:

- LAV-AT-ES1

- LN2-CT

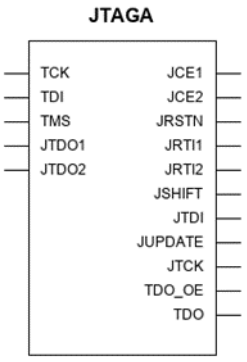


Table 341: Input Ports

Name	Range	Description
TCK	1	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.
TDI	1	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.
TMS	1	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.
JTDO1	1	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.
JTDO2	1	JTAG Test Data Out two

Table 342: Output Ports

Name	Range	Description
JCE1	1	JTAG Clock Enable one (BS is to boundary scan ring (L2).
JCE2	1	JTAG Clock Enable two

Table 342: (Continued)Output Ports

Name	Range	Description
JRSTN	1	JTAG ResetN Active low reset signal. Set low initially and then remains high after initialization.
JRTI1	1	JTAG Run-Test/Idle one
JRTI2	1	JTAG Run-Test/Idle two
JSHIFT	1	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JTDI	1	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.
JUPDATE	1	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JTCK	1	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.
TDO_OE	1	TAP Data Output Enable
TDO	1	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 343: Parameters

Name	Values	Description
ER1	"DISABLED" (default) "ENABLED"	Enable JTAG 1
ER2	"DISABLED" (default) "ENABLED"	Enable JTAG 2

JTAGH25

Supports hard JTAG operations

Architectures supported:

- LAV-AT-ES1
- LN2-CT

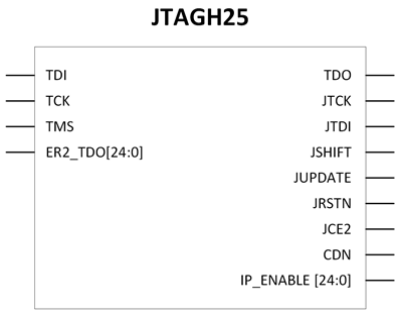


Table 344: Input Ports

Name	Range	Description
TDI	1	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.
TCK	1	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	1	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.
ER2_TDO	24:0	Input data to be shifted out of TDO.

Table 345: Output Ports

Name	Range	Description
TDO	1	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	1	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	1	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.

Table 345: (Continued)Output Ports

Name	Range	Description
JSHIFT	1	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JUPDATE	1	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JRSTN	1	JTAG ResetN Active low reset signal. Set low initially and then remains high after initialization.
JCE2	1	JTAG Clock Enable two
CDN	1	JTAG RTI1
IP_ENABLE	24:0	JTAG RTI2

Table 346: Parameters

Name	Values	Description
LSCC_FAMILY	"AVANT"	String in both Verilog and VHDL
JTAGHUB_ID	"0x43"	String in both Verilog and VHDL
IP_ENABLE_VAL	"0x00000"	String in both Verilog and VHDL
HUB_00	"0b0"	String in both Verilog and VHDL
HUB_01	"0b0"	String in both Verilog and VHDL
HUB_02	"0b0"	String in both Verilog and VHDL
HUB_03	"0b0"	String in both Verilog and VHDL
HUB_04	"0b0"	String in both Verilog and VHDL
HUB_05	"0b0"	String in both Verilog and VHDL
HUB_06	"0b0"	String in both Verilog and VHDL
HUB_07	"0b0"	String in both Verilog and VHDL
HUB_08	"0b0"	String in both Verilog and VHDL
HUB_09	"0b0"	String in both Verilog and VHDL
HUB_10	"0b0"	String in both Verilog and VHDL
HUB_11	"0b0"	String in both Verilog and VHDL
HUB_12	"0b0"	String in both Verilog and VHDL
HUB_13	"0b0"	String in both Verilog and VHDL
HUB_14	"0b0"	String in both Verilog and VHDL
HUB_15	"0b0"	String in both Verilog and VHDL

Table 346: Parameters (Continued)

Name	Values	Description
HUB_16	"0b0"	String in both Verilog and VHDL
HUB_17	"0b0"	String in both Verilog and VHDL
HUB_18	"0b0"	String in both Verilog and VHDL
HUB_19	"0b0"	String in both Verilog and VHDL
HUB_20	"0b0"	String in both Verilog and VHDL
HUB_21	"0b0"	String in both Verilog and VHDL
HUB_22	"0b0"	String in both Verilog and VHDL
HUB_23	"0b0"	String in both Verilog and VHDL
HUB_24	"0b0"	String in both Verilog and VHDL

JTAGH25SOFT

Supports soft JTAG operations

Architectures supported:

- LAV-AT-ES1
- LN2-CT

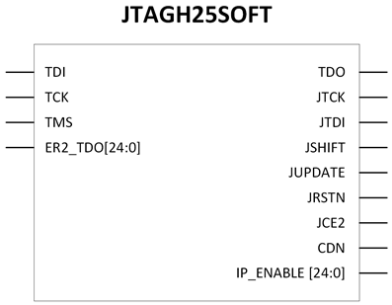


Table 347: Input Ports

Name	Range	Description
TDI	1	TAP Data Input
TCK	1	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.

Table 347: Input Ports (Continued)

Name	Range	Description
TMS	1	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.
ER2_TDO	24:0	Input data to be shifted out of TDO

Table 348: Output Ports

Name	Range	Description
TDO	1	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	1	JTAG Clock
JTDI	1	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	1	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JUPDATE	1	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JRSTN	1	JTAG ResetN Active low reset signal. Set low initially and then remains high after initialization.
JCE2	1	JTAG Clock Enable two
CDN	1	JTAG RTI1
IP_ENABLE	24:0	JTAG RTI2

Table 349: Parameters

Name	Values	Description
LSCC_FAMILY	"AVANT"	String in both Verilog and VHDL
JTAGHUB_ID	"0x43"	String in both Verilog and VHDL
IP_ENABLE_VAL	"0x00000"	String in both Verilog and VHDL
HUB_00	"0b0"	String in both Verilog and VHDL

Table 349: Parameters (Continued)

Name	Values	Description
HUB_01	"0b0"	String in both Verilog and VHDL
HUB_02	"0b0"	String in both Verilog and VHDL
HUB_03	"0b0"	String in both Verilog and VHDL
HUB_04	"0b0"	String in both Verilog and VHDL
HUB_05	"0b0"	String in both Verilog and VHDL
HUB_06	"0b0"	String in both Verilog and VHDL
HUB_07	"0b0"	String in both Verilog and VHDL
HUB_08	"0b0"	String in both Verilog and VHDL
HUB_09	"0b0"	String in both Verilog and VHDL
HUB_10	"0b0"	String in both Verilog and VHDL
HUB_11	"0b0"	String in both Verilog and VHDL
HUB_12	"0b0"	String in both Verilog and VHDL
HUB_13	"0b0"	String in both Verilog and VHDL
HUB_14	"0b0"	String in both Verilog and VHDL
HUB_15	"0b0"	String in both Verilog and VHDL
HUB_16	"0b0"	String in both Verilog and VHDL
HUB_17	"0b0"	String in both Verilog and VHDL
HUB_18	"0b0"	String in both Verilog and VHDL
HUB_19	"0b0"	String in both Verilog and VHDL
HUB_20	"0b0"	String in both Verilog and VHDL
HUB_21	"0b0"	String in both Verilog and VHDL
HUB_22	"0b0"	String in both Verilog and VHDL
HUB_23	"0b0"	String in both Verilog and VHDL
HUB_24	"0b0"	String in both Verilog and VHDL

LSOSC

Low-frequency oscillator. Generates 10-kHz nominal clock, +/- 10 percent. Can drive global clock network or fabric routing.

Architectures supported:

- iCE40 UltraPlus

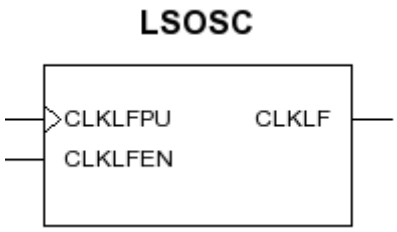


Table 350: Input Ports

Name	Description
CLKLFPU	Power up the oscillator. After power up, output will be stable after 100 µs. Active high
CLKLFEN	Enable the clock output. Enable should be low for the 100-µs power-up period. Active high

Table 351: Output Ports

Name	Description
CLKLF	Oscillator output)

SAME AS: SB_LFOSC

LSOSC1P8V

Low-frequency oscillator. Generates 10-kHz nominal clock, +/- 10 percent. Can drive global clock network or fabric routing. For use when VPP_2V5 is connected to a voltage below 2.3 V.

Architectures supported:

- iCE40 UltraPlus

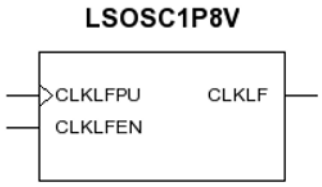


Table 352: Input Ports

Name	Description
CLKLFPU	Power up the oscillator. After power up, output will be stable after 100 µs. Active high
CLKLFEN	Enable the clock output. Enable should be low for the 100-µs power-up period. Active high

Table 353: Output Ports

Name	Description
CLKLF	Oscillator output

LSOSC_CORE

Low-frequency oscillator. Generates 10-kHz nominal clock, +/- 10 percent. Can drive global clock network or fabric routing.

Architectures supported:

- iCE40 UltraPlus

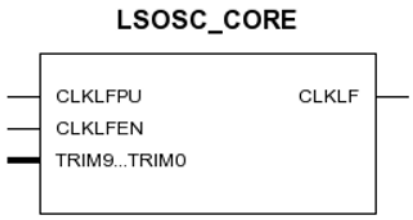


Table 354: Input Ports

Name	Description
CLKLFPU	Power up the oscillator. After power up, output will be stable after 100 µs. Active high
CLKLFEN	Enable the clock output. Enable should be low for the 100-µs power-up period. Active high
TRIM9...TRIM0	

Table 355: Output Ports

Name	Description
CLKLF	Oscillator output

Table 356: Parameters

Name	Values	Description
FABRIC_TRIME	"DISABLE" (default), "ENABLE"	Trim bits source selection. ENABLE - Trim bits sourced from Fabric, DISABLE - Trim bits sourced from NVCM.

LUT4

4-Input Look-Up Table

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

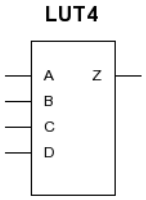


Table 357: Input Ports

Name	Description
A	LUT input A
B	LUT input B

Table 357: Input Ports

Name	Description
C	LUT input C
D	LUT input D

Table 358: Output Ports

Name	Description
Z	LUT output Z

Table 359: Parameters

Name	Values	Description
INT	0x0000 (default)	

Note:

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.

Table 360: LUT4 Truth Table

Inputs				Output
D	C	B	A	Z
0	0	0	0	INIT[0]
0	0	0	1	INIT[1]
0	0	1	0	INIT[2]
0	0	1	1	INIT[3]
0	1	0	0	INIT4]
0	1	0	1	INIT[5]
0	1	1	0	INIT[6]
0	1	1	1	INIT[7]
1	0	0	0	INIT[8]
1	0	0	1	INIT[9]
1	0	1	0	INIT[10]
1	0	1	1	INIT[11]
1	1	0	0	INIT[12]

Table 360: LUT4 Truth Table (Continued)

Inputs				Output
D	**C**	**B**	**A**	**Z**
1	1	0	1	INIT[13]
1	1	1	0	INIT[14]
1	1	1	1	INIT[15]

MAC16

DSP block capable of being configured as a multiplier, adder, subtractor, accumulator, multiply-adder, or multiply-subtractor.

Architectures supported:

- iCE40 UltraPlus

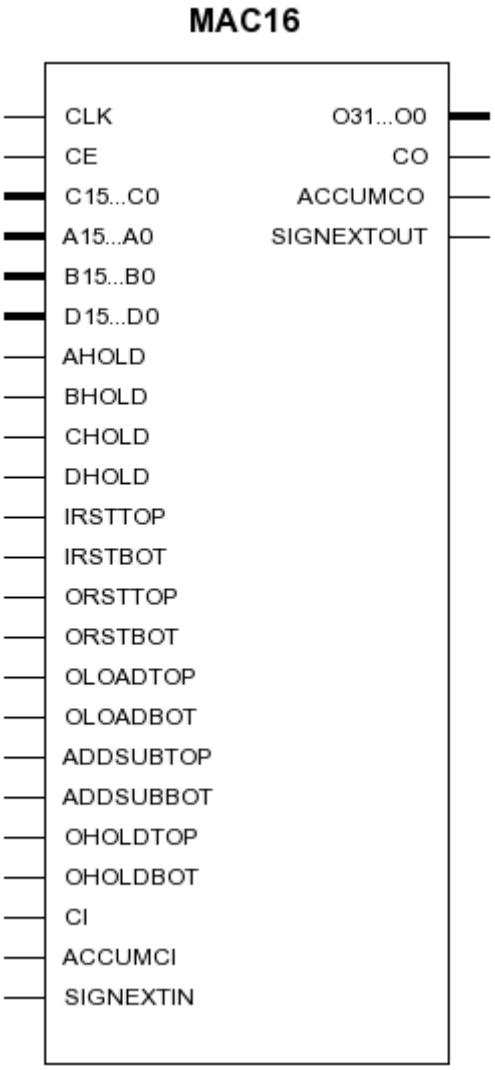


Table 361: Input Ports

Name	Description
CLK	Clock input. Applies to all clocked elements in the MAC16A block.
CE	Clock Enable input. Active High.
C15...C0	Input to the C Register/Direct input to the adder accumulator.
A15...A0	Input to the A Register/Direct input to the multiplier blocks/Direct input to the adder accumulator
B15...B0	Input to the B Register/Direct input to the multiplier blocks/Direct input to the adder accumulator
D15...D0	Input to the D Register/Direct input to the adder accumulator.

Table 361: Input Ports (Continued)

Name	Description
AHOLD	Hold A registers Data. Controls data flow into the input register A. Active High. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.
BHOLD	Hold B registers Data. Controls data flow into the B input register. Active High. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.
CHOLD	Hold C registers Data. Controls data flow into the C input register. Active High. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.
DHOLD	Hold D registers Data. Controls data flow into the D input register. Active High. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.
IRSTTOP	Reset input to the A and C input registers, and the pipeline registers in the upper half of the multiplier block. Active High.
IRSTBOT	Reset input to the B and C input registers, and the pipeline registers in the lower half of the multiplier block, and the 32-bit multiplier result pipeline register. Active High.
ORSTTOP	Reset the high-order bits of the accumulator register ([31:16]). Active High.
ORSTBOT	Reset the low-order accumulator register bits ([15:0]). Active High.
OLOADTOP	High-order Accumulator Register Accumulate/Load. Controls whether the accumulator register accepts the output of the adder/subtractor or whether the register is loaded with the value from Input C (or Register C, if configured). 0 - Accumulator Register [31:16] loaded with output from adder/subtractor. 1 - Accumulator Register [31:16] loaded with Input C or Register C, depending on primitive parameter value.
OLOADBOT	Low-order Accumulator Register Accumulate/Load. Controls whether the low-order accumulator register bits (15:0] accepts the output of the adder/subtractor or whether the register is loaded with the value from Input D (or Register D, if configured). 0 - Accumulator Register [15:0] loaded with output from adder/subtractor. 1 - Accumulator Register [15:0] loaded with Input D or Register D, depending on primitive parameter value.
ADDSUBTOP	High-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts. 0 - Add: W+X+HCI 1 - Subtract: W-X-HCI.
ADDSUBBOT	Low-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts. 0 - Add: Y+Z+LCI 1 - Subtract: Y-Z-LCI.
OHOLDTOP	High-order Accumulator Register Hold. Controls data flow into the high-order ([31:16]) bits of the accumulator. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.

Table 361: Input Ports (Continued)

Name	Description
OHOLDBOT	Low-order Accumulator Register Hold. Controls data flow into the high-order ([15:0]) bits of the accumulator. 0 - Update (load) register at next active clock edge. 1 - Hold (retain) current register value, regardless of clock.
CI	Carry/borrow input from lower logic tile.
ACCUMCI	Cascade carry/borrow input from lower MAC16 block.
SIGNEXTIN	Sign extension input from lower MAC16 block.

Table 362: Output Ports

Name	Description
O31...O0	32-bit MAC16 output.
O[31:0]	32-bit result of a 16x16 multiply operation or a 32-bit adder/ accumulate function.
O[31:16]	16-bit result of an 8x8 multiply operation or a 32-bit adder/ accumulate function.
O[15:0]	16-bit result of an 8x8 multiply operation or a 32-bit adder/ accumulate function).
CO	Carry/borrow output to higher logic tile.
ACCUMCO	Cascade carry/borrow output to higher MAC16 block.
SIGNEXTOUT	Sign extension output to higher MAC16 block.

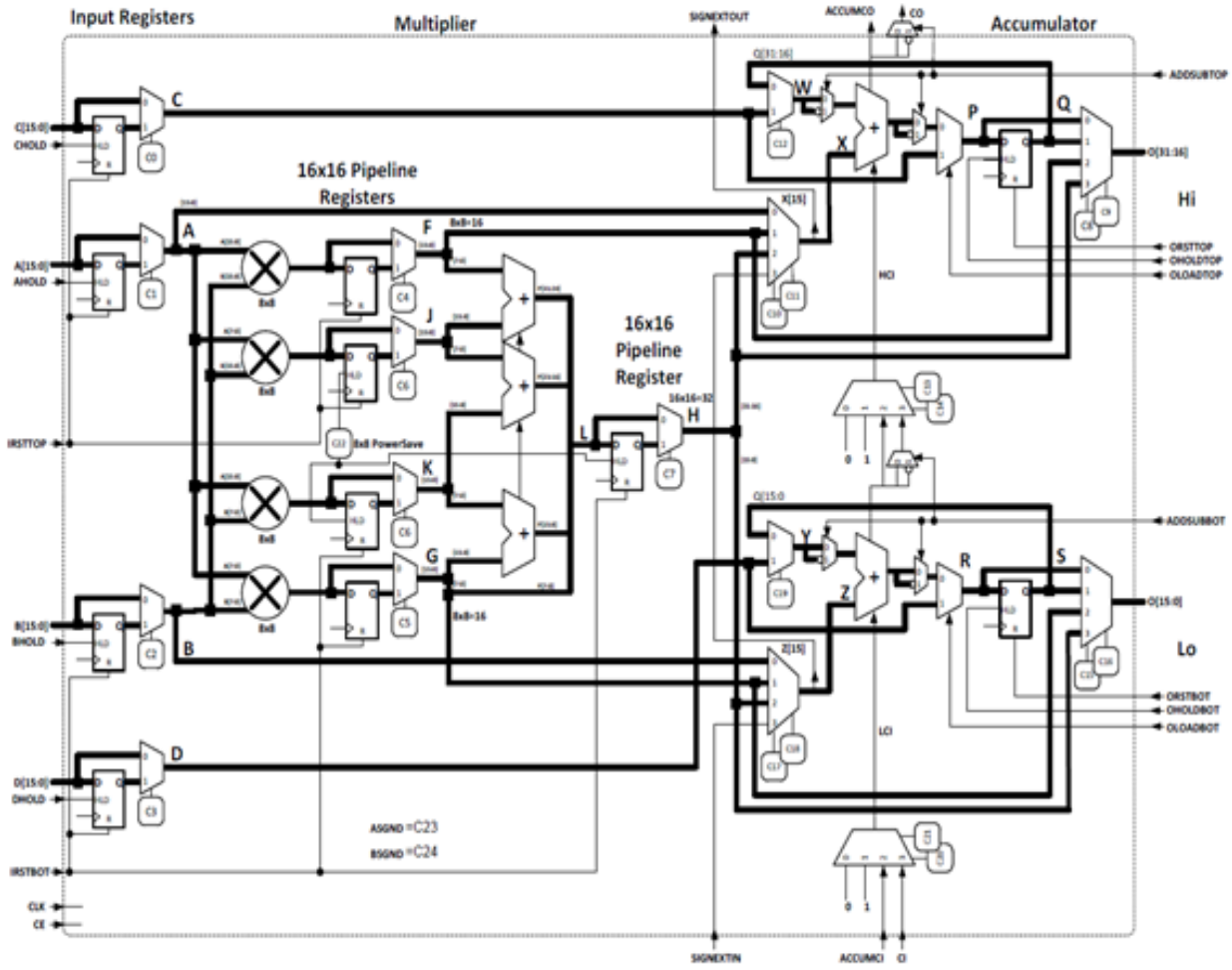
Table 363: Parameters

Name	Values	Description
NEG_TRIGGER	"0’b0" (default), "0’b1"	Controls input clock polarity.
A_REG	"0’b0" (default), "0’b1"	Input A register Control, 0’b1 = registered (C1)
B_REG	"0’b0" (default), "0’b1"	Input B register Control, 0’b1 = registered (C2)
C_REG	"0’b0" (default), "0’b1"	Input C register Control, 0’b1 = registered (C0)
D_REG	"0’b0" (default), "0’b1"	Input D register Control, 0’b1 = registered (C3)
TOP_8x8_MULT_REG	"0’b0" (default), "0’b1"	Top 8x8 multiplier output register control, 0’b1 = registered (C4)
BOT_8x8_MULT_REG	"0’b0" (default), "0’b1"	Bottom 8x8 multiplier output register control, 0’b1 = registered (C5)

Table 363: Parameters (Continued)

Name	Values	Description
PIPELINE_16x16_MULT_REG1	"0'b0" (default), "0'b1"	16x16 intermediate register control, 0'b1 = registered (C6)
PIPELINE_16x16_MULT_REG2	"0'b0" (default), "0'b1"	6x16 multiplier pipeline register control, 0'b1 = registered (C7)
TOPOUTPUT_SELECT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Selects top output O[31:16], 0'b00 = top accumulator, 0'b01 = top accumulator (registered), 0'b10 = top mult8x8, 0'b11 = mult16x16 (C8, C9)
TOPADDSUB_LOWERINPUT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Selects lower input for the upper adder/subtractor, 0'b00 = input A, 0'b01 = top mult8x8, 0'b10 = mult16x16, 0'b11 = sign extend from lower adder/ subtractor (C10, C11)
TOPADDSUB_UPPERINPUT	"0'b0" (default), "0'b1"	Selects upper input for the upper adder/subtractor, 0'b0 = accumulate, 0'b1 = input C (C12)
TOPADDSUB_CARRYSELECT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Carry/borrow input select to upper adder/subtractor, 0'b00 = constant 0, 0'b01 = constant 1, 0'b10 = carry from lower adder/subtractor, 0'b11 = carry from lower adder/subtractor (C13, C14)
BOTOUTPUT_SELECT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Selects lower output O[15:0], 0'b00 = bottom adder/ subtractor, 0'b01 = bottom adder/subtractor (registered), 0'b10 = bottom mult8x8, 0'b11 = mult16x16 (C15, C16)
BOTADDSUB_LOWERINPUT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Selects lower input for the lower adder/subtractor, 0'b00 = input B, 0'b01 = bottom mult8x8, 0'b10 = mult16x16, 0'b11 = sign extend from previous MAC16 (C17, C18)
BOTADDSUB_UPPERINPUT	"0'b0" (default), "0'b1"	Selects upper input for the lower adder/subtractor, 0'b0 = accumulate, 0'b1 = input C (C19)
BOTADDSUB_CARRYSELECT	"0'b00" (default), "0'b01", "0'b10", "0'b11"	Carry/borrow input select to lower adder/subtractor, 0'b00 = constant 0, 0'b01 = constant 1, 0'b10 = ACCUMCI, 0'b11 = CI (C20, C21)
MODE_8x8	"0'b0" (default), "0'b1"	Selects 8x8 Multiplier mode and 8x8 Low-Power Multiplier Blocking Option, 0'b1 = 8x8 mode (C22)
A_SIGNED	"0'b0" (default), "0'b1"	Indicates whether multiplier input A is signed or unsigned, 0'b1 = signed
B_SIGNED	"0'b0" (default), "0'b1"	Indicates whether multiplier input B is signed or unsigned. 0'b1 = signed

Note:



SAME AS: SB_MAC16

Rules & Restrictions:

As noted by pin description, ACCUMCI and SIGNEXTIN can only come from previous MAC16's ACCUMCO and SIGNEXTOUT.

Failure to do so will cause connectivity DRC error.

If MODE_8x8 is 1, then PIPELINE_16x16_MULT_REG1 and PIPELINE_16x16_MULT_REG2 must be 0'b0.

Note:

If the operation is 8x8, then these PIPELINE attributes are irrelevant. However, we have to be strict because we don't know the intended use of the DSP. If operation is truly 16x16, then this illegal combination setting would cause an incorrect operation.

MIPI

Soft MIPI Interface Block

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

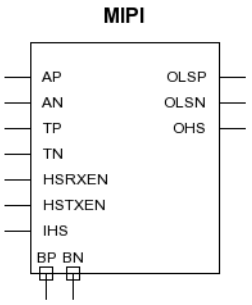


Table 364: Input Ports

Name	Range	Description
AP		
AN		
TP		
TN		
HSRXEN		
HSTXEN		
IHS		

Table 365: Output Ports

Name	Range	Description
OLSP		
OLSN		
OHS		

Table 366: Bidirectional Ports

Name	Range	Description
BP		
BN		

Table 367: Parameters

Name	Values	Description
MIPI_ID	0 (default)	

MIPIA

Soft version of bidirectional MIPI

Architectures supported:

- LAV-AT-ES1
- LN2-CT

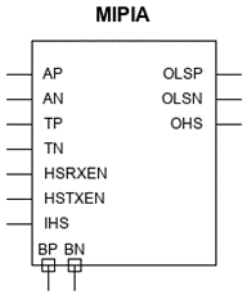


Table 368: Input Ports

Name	Range	Type	Description
BP	1	Data	IOPAD for P-port
BN	1	Data	IOPAD for N-port
AP	1	Data	LP Data for P-port
AN	1	Data	LP Data for N-port
TP	1	Control	LP Tri-sate control for P-port
TN	1	Control	LP Tri-sate control for N-port
HSRXEN	1	Control	HSRXEN for MIPI RX: 0 disable; 1 enable
HSTXEN	1	Control	HSTXEN for MIPI TX to support BIDI: 0 disable, 1 enable
IHS	1	Data	

Table 369: Output Ports

Name	Range	Type	Description
OLSP	1	Clock	Low Power MIPI output for P-port
OLSN	1	Clock	Low Power MIPI output for N-port
OHS	1	Clock	High Speed output

Table 370: Parameters

Name	Values	Description
MIPI_ID	"0" (default) Any integer	The MIPI_ID is used to place 2 PIO in a pair of PIO. 0 is not used for MIPI >0 is for MIPI pair

MULT18X18

18x18 Multiplier with Optional Input/Output Registers

Architectures supported:

- LFCPNX

- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

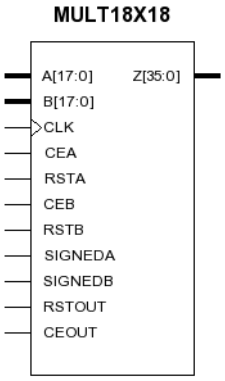


Table 371: Input Ports

Name	Range	Description
A	17:0	Operand A
B	17:0	Operand B
CLK		Clock
CEA		Input A enable
RSTA		Input A reset
CEB		Input clock enable
RSTB		Input register reset
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 372: Output Ports

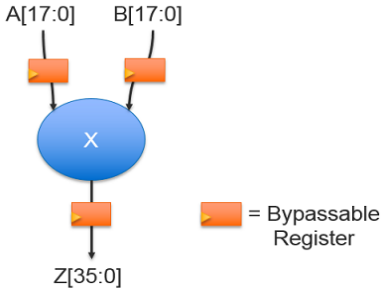
Name	Range	Description
Z	35:0	Multiplication result

Table 373: MULT18X18 Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	

Note:

Implements Z[35:0] = A[17:0] x B[17:0].



MULT18X18A

18x18 Multiplier with Optional Input/Output Registers

Architectures supported:

- LAV-AT-ES1
- LN2-CT

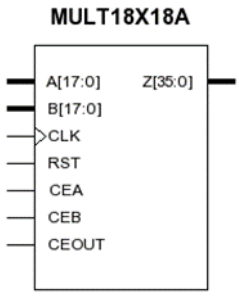


Table 374: Input Ports

Name	Range	Type	Description
A	17:0	Data	Operand A
B	17:0	Data	Operand B
CLK	1	Clock	Clock
RST	1	Reset	Synchronous reset for all registers
CEA	1	Clock Enable	Input A register clock enable
CEB	1	Clock Enable	Input B register clock enable
CEOUT	1	Clock Enable	Output register clock enable

Table 375: Output Ports

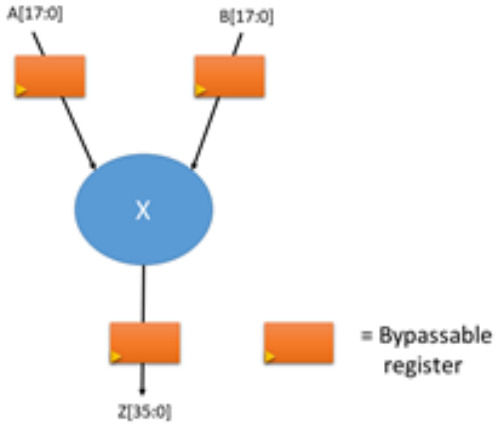
Name	Range	Type	Description
Z	35:0	Control	Multiplication result

Table 376: Parameters

Name	Values	Description
REGINPUTA	"REGISTERED" (default) "BYPASSED"	Register operand A
REGINPUTB	"REGISTERED" (default) "BYPASSED"	Register operand B
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:

Implements the function Z[35:0] = A[17:0] * B[17:0]



MULT18X36

18x36 Multiplier with Optional Input/Output Register

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

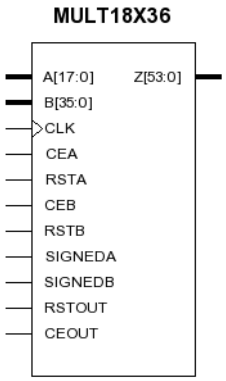


Table 377: Input Ports

Name	Range	Description
A	17:0	Operand A
B	35:0	Operand B
CLK		Clock
CEA		Input clock enable
RSTA		Input register reset
CEB		Input clock enable
RSTB		Input register reset
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 378: Output Ports

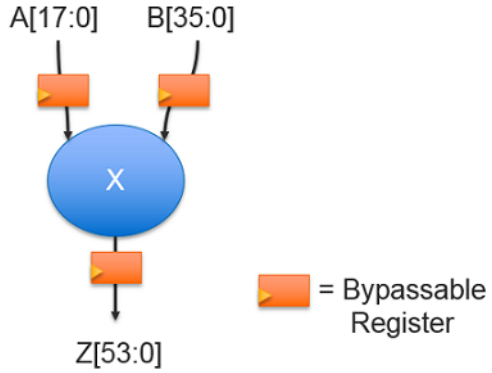
Name	Range	Description
Z	53:0	Multiplication result

Table 379: MULT18X36 Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	

Note:

Implements Z[53:0] = A[17:0] x B[35:0].



MULT36X36

36x36 Multiplier with Optional Input/Output Registers

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

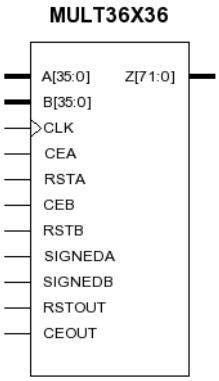


Table 380: Input Ports

Name	Range	Description
A	35:0	Operand A
B	35:0	Operand B
CLK		Clock

Table 380: Input Ports (Continued)

Name	Range	Description
CEA		Input A clock enable
RSTA		Input A register reset
CEB		Input B clock enable
RSTB		Input B register reset
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 381: Output Ports

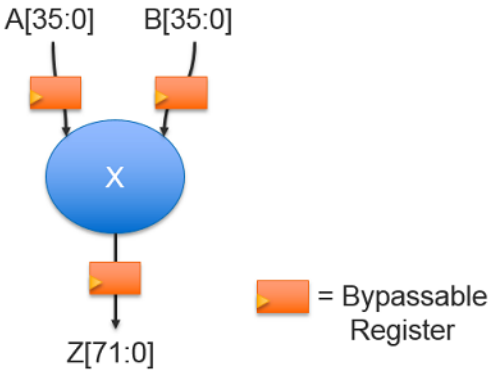
Name	Range	
Z	71:0	Multiplication result

Table 382: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	

Note:

Implements Z[71:0] = A[35:0] x B[35:0].



MULT8X8

8x8 Multiplier with Optional Input/Output Registers

Architectures supported:

- LAV-AT-ES1
- LN2-CT

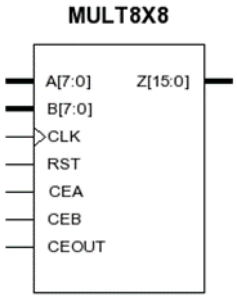


Table 383: Input Ports

Name	Range	Type	Description
A	7:0	Data	Operand A
B	7:0	Data	Operand B
CLK	1	Clock	Clock
RST	1	Reset	Synchronous reset for all registers
CEA	1	Clock Enable	Input A register clock enable

Table 383: Input Ports (Continued)

Name	Range	Type	Description
CEB	1	Clock Enable	Input B register clock enable
CEOUT	1	Clock Enable	Output register clock enable

Table 384: Output Ports

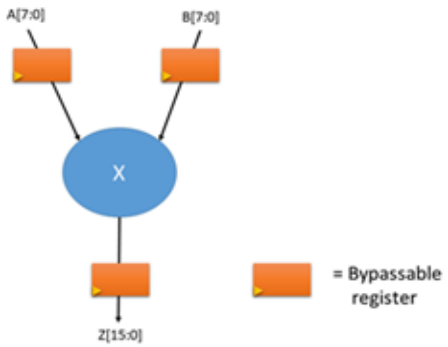
Name	Range	Type	Description
Z	15:0	Control	Multiplication result

Table 385: Parameters

Name	Values	Description
REGINPUTA	"REGISTERED" (default) "BYPASSED"	Register operand A
REGINPUTB	"REGISTERED" (default) "BYPASSED"	Register operand B
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:

Implements the function Z[15:0] = A[7:0] * B[7:0]



MULT9X9

9x9 Multiplier with Optional Input/Output Registers

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

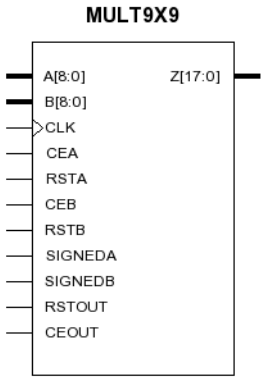


Table 386: Input Ports

Name	Range	Description
A	8:0	Operand A
B	8:0	Operand B
CLK		Clock
CEA		Input A clock enable
RSTA		Input A reset
CEB		Input B clock enable
RSTB		Input B reset
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 387: Output Ports

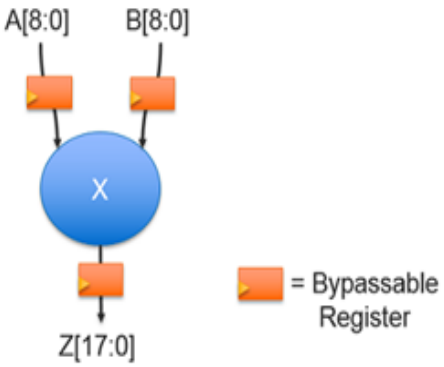
Name	Range	Description
Z	17:0	Multiplication result

Table 388: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	Synchronous/asynchronous reset control. Reset release is always asynchronous.

Note:

Implements Z[17:0] = A[8:0] x B[8:0]



MULT9X9A

9x9 Multiplier with Optional Input/Output Registers

Architectures supported:

- LAV-AT-ES1
- LN2-CT

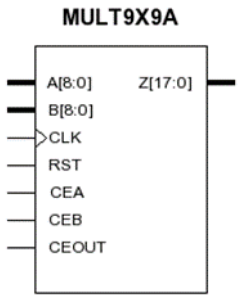


Table 389: Input Ports

Name	Range	Type	Description
A	8:0	Data	Operand A
B	8:0	Data	Operand B
CLK	1	Clock	Clock
RST	1	Reset	Synchronous reset for all registers
CEA	1	Clock Enable	Input A register clock enable
CEB	1	Clock Enable	Input B register clock enable
CEOUT	1	Clock Enable	Output register clock enable

Table 390: Output Ports

Name	Range	Type	Description
Z	17:0	Control	Multiplication result

Table 391: Parameters

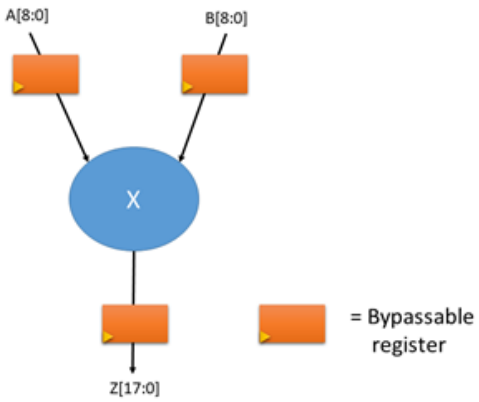
Name	Values	Description
REGINPUTA	"REGISTERED" (default) "BYPASSED"	Register operand A
REGINPUTB	"REGISTERED" (default) "BYPASSED"	Register operand B

Table 391: Parameters (Continued)

Name	Values	Description
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:

Implements the function Z[17:0] = A[8:0] * B[8:0]



MULTADDSUB18X18

18x18 Multiplier and Accumulator

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

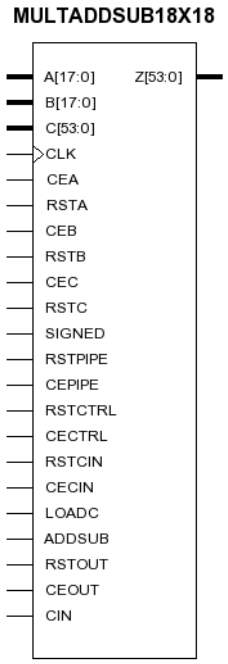


Table 392: Input Ports

Name	Range	Description
A	17:0	Operand A
B	17:0	Operand B
C	53:0	Operand C
CLK		Input clock
CEA		Input clock enable
RSTA		Input register reset
CEB		Input clock enable
RSTB		Input register reset
CEC		Input clock enable
RSTC		Input register reset
SIGNED		One SIGNED port for all inputs due to ACC54 hardware limitation
RSTPIPE		Output register reset
CEPIPE		Output clock enable

Table 392: Input Ports (Continued)

Name	Range	Description
RSTCTRL		Input control (addsub, m9addsub, signed, and loadc) reset
CECTRL		Input control (addsub, m9addsub, signed, and loadc) enable
RSTCIN		Input carryin reset
CECIN		Input carryin enable
LOADC		1=Load value from C input
ADDSUB		0=Add, 1=subtract
RSTOUT		Output register reset
CEOUT		Output clock enable
CIN		Carry in

Table 393: Output Ports

Name	Range	Description
Z	53:0	Multiplication result

Table 394: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGINPUTC	"REGISTER" (default) "BYPASS"	Register operand C
REGADDSUB	"REGISTER" (default) "BYPASS"	Register ADDSUB
REGLOADC	"REGISTER" (default) "BYPASS"	Register LOADC
REGCIN	"REGISTER" (default) "BYPASS"	Register CIN
REGPIPELINE	"REGISTER" (default) "BYPASS"	Register between multiplier and accumulator
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output

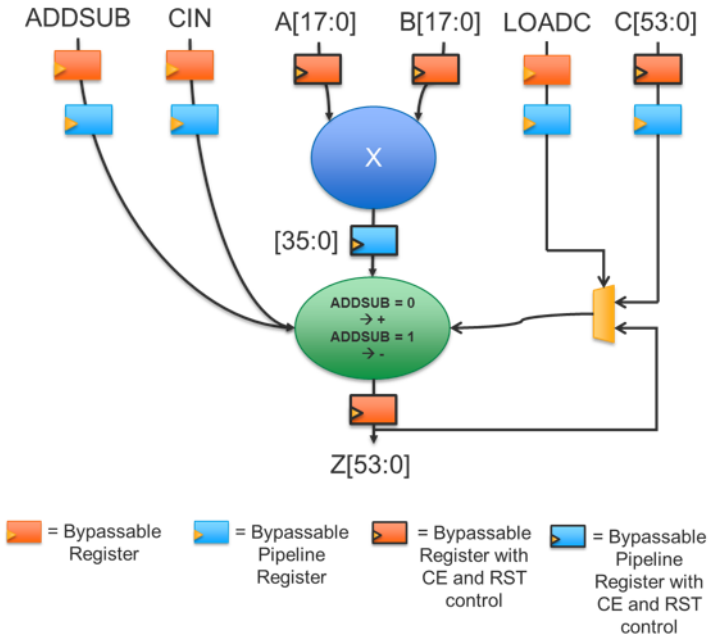
Table 394: Parameters

Name	Values	Description
GSR	"DISABLED" *(default)* "ENABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	

Note:

If LOADC = 1: Z[53:0] = C[53:0] +/- (A[17:0] x B[17:0]) +/- CIN

If LOADC = 0: Z[53:0] = Z_{prev}[53:0] +/- (A[17:0] * B[17:0]) +/- CIN.



MULTADDSUB18X18A

18x18 Multiplier and Adder/Subtractor

Architectures supported:

- LAV-AT-ES1
- LN2-CT

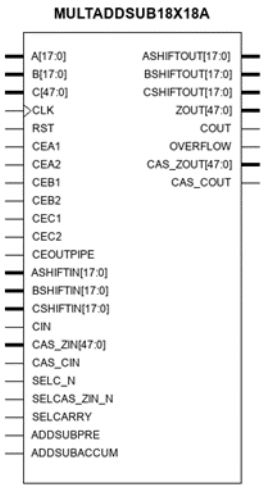


Table 395: Input Ports

Name	Range	Type	Active State	Description
A	17:0	Data	Active high	Operand A
B	17:0	Data	Active high	Operand B
CLK	1	Clock	Active high	Clock
RST	1	Reset	Active high	Synchronous reset for all registers
CEA1	1	Clock Enable	Active high	Input A register clock enable, stage one
CEA2	1	Clock Enable	Active high	Input A register clock enable, stage two
CEB1	1	Clock Enable	Active high	Input B register clock enable, stage one
CEB2	1	Clock Enable	Active high	Input B register clock enable, stage two
CEOUTPIPE	1	Clock Enable	Active high	Pipeline and output register clock enable
C	47:0	Control	Active high	Operand C
CEC1	1	Clock Enable	Active high	Input C register clock enable, stage one
CEC2	1	Clock Enable	Active high	Input C register clock enable, stage two
ASHIFTIN	17:0	Data	Active high	Operand A shift chain input
BSHIFTIN	17:0	Data	Active high	Operand B shift chain input
CSHIFTIN	17:0	Control	Active high	Operand C shift chain input
CIN	1	Control	Active high	Carry in
CAS_ZIN	47:0	Control	Active low	Cascade input
CAS_CIN	1	Control	Active low	Cascade carry in

Table 395: Input Ports (Continued)

Name	Range	Type	Active State	Description
SELC_N	1	Control	Active high	Select between C or accumulator output for input to the accumulator
SELCAS_ZIN_N	1	Control	Active low	Select between CAS_ZIN or accumulator output for input to the accumulator
SELCARRY	1	Control	Active high	Select between CIN or accumulator output Z[47] for input to the accumulator
ADDSUBPRE	1	Control	Active high	Add/sub control for pre-adder, 0=add, 1=subtract
ADDSUBACCUM	1	Control	Active high	Add/sub control for accumulator, 0=add, 1=subtract

Table 396: Output Ports

Name	Range	Type	Active State	Description
ASHIFTOUT	17:0	Control	Active high	Operand A shift chain output
BSHIFTOUT	17:0	Control	Active high	Operand B shift chain output
CSHIFTOUT	17:0	Control	Active high	Operand C shift chain output
ZOUT	47:0	Control	Active high	Multiplication/accumulator output result
COUT	1	Control	Active high	Carry out
OVERFLOW	1	Flag	Active high	Overflow flag
CAS_ZOUT	47:0	Control	Active low	Cascade multiplication/accumulator output result
CAS_COUT	1	Control	Active low	Cascade carry out

Table 397: Parameters

Name	Values	Description
REGINPUTA	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand A
REGINPUTB	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand B

Table 397: Parameters (Continued)

Name	Values	Description
REGSHIFTOUTA	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output A
REGSHIFTOUTB	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output B
SHIFTINPUTA	"DISABLED" (default) "ENABLED"	Shift input A
SHIFTINPUTB	"DISABLED" (default) "ENABLED"	Shift input B
REGINPUTC	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register operand C
SHIFTINPUTC	"DISABLED" (default) "ENABLED"	Shift input C
REGSHIFTOUTC	"REGISTERED_ONCE" (default) "BYPASSED" "REGISTERED_TWICE"	Register shift output C
REGPREPIPE	"REGISTERED" (default) "BYPASSED"	Register pre-adder output, if PREADD_EN=ENABLED
REGPIPE	"REGISTERED" (default) "BYPASSED"	Register pipeline
REGOUTPUT	"REGISTERED" (default) "BYPASSED"	Register output
REGCAS_ZOUT	"REGISTERED" (default) "BYPASSED"	Register CAS_COUT and CAS_ZOUT[47:0].
REGACCUMCONTROLS	"REGISTERED" (default) "BYPASSED"	Register accumulator controls (CIN, SELCARRY, SELCAS_ZIN, ADDSUBACCUM, SELC)
RESETMODE	"SYNC" (default) "ASYNC"	Synchronous/asynchronous reset control
ACCUM_EN	"DISABLED" (default) "ENABLED"	Enable accumulator
CAS_ZOUT_RSHIFT	"DISABLED" (default) "ENABLED"	Cascade output 18 bit right shift

Table 397: Parameters (Continued)

Name	Values	Description
PREADD_EN	"DISABLED" (default) "ENABLED"	Enable pre-adder
REGADDSUBPRE	"REGISTERED" (default) "BYPASSED"	Register ADDSUBPRE signal
ASIGNED	"SIGNED" (default) "UNSIGNED"	Input A signed control (only used when cascading to build wider multipliers)
BSIGNED	"SIGNED" (default) "UNSIGNED"	Input B signed control (only used when cascading to build wider multipliers)
ACCUM_C_EN	"DISABLED" (default) "ENABLED"	Enable dynamic C for accumulator input (1) or tie to constant 0 (0)
CAS_ZIN_EN	"DISABLED" (default) "ENABLED"	Enable dynamic CAS_ZIN for accumulator input (1) or tie to constant 0 (0)
CAS_CIN_EN	"DISABLED" (default) "ENABLED"	Use CAS_CIN for carry input
ROUNDMODE	"ROUND_TO_ZERO" (default) "ROUND_TO_INFINITY"	Rounding mode
SATURATION	"DISABLED" (default) "TO_NEG_MAX" "TO_ZERO"	Saturation enable
SATURATION_BITS	"48" (default) "3-48"	Saturation bits
GSR	"DISABLED" (default) "ENABLED"	Enable global set/reset

Note:

When the pre-adder and accumulator are both enabled, it implements the function:

M_INT = A * (B+/-C)

Z_out = CAS_INT + C_INT + CIN_INT +/- M_INT

Note that the pre-adder and accumulator share the C input. If the pre-adder is used, typically ACCUM_C_EN should be set to DISABLED so the accumulator ignores the value of the C port.

When the pre-adder is enabled and the accumulator is disabled, it implements the function:

Z_out = A * (B+/-C)

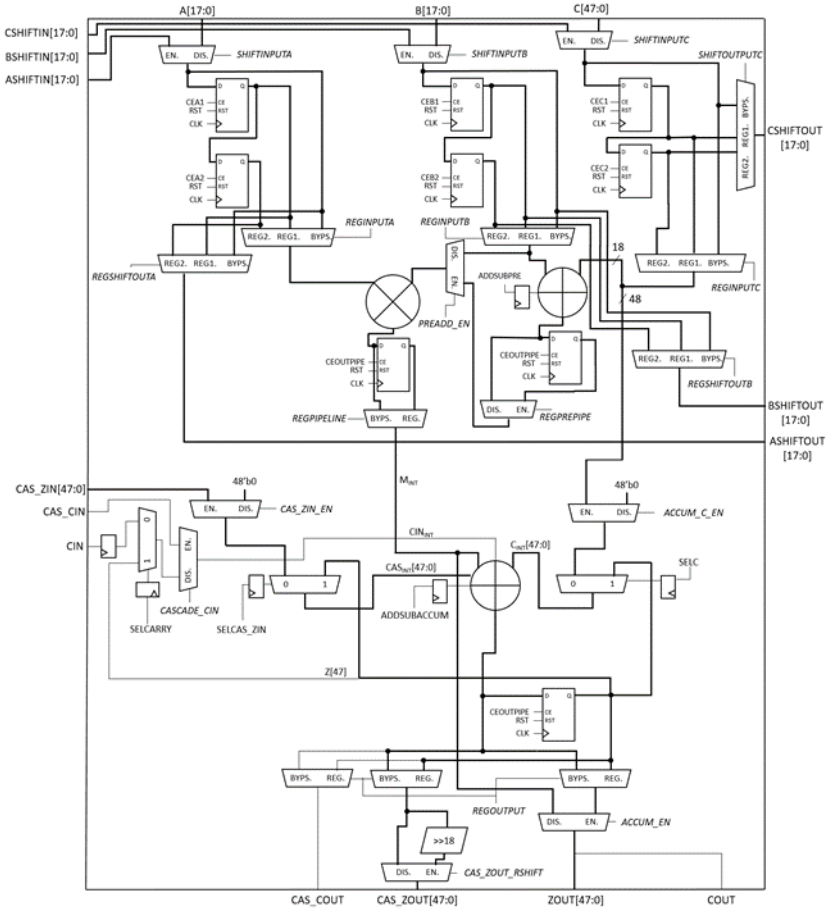
When the pre-adder is disabled and the accumulator is enabled, it implements the function:

M_int = A * B

Z_out = CAS_INT + C_INT + CIN_INT +/- M_INT

See the diagram below for the internal node corresponding to the above terms.

Although not shown in the diagram below, the register for accumulator inputs CIN, CAS_ZIN_EN, SELCAS_ZIN, and ADDSUBACCUM is bypassable with the parameter REGACCUMCONTROLS.



MULTADDSUB18X18WIDE

18x18 Wide Multiplier and Adder/Subtractor

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

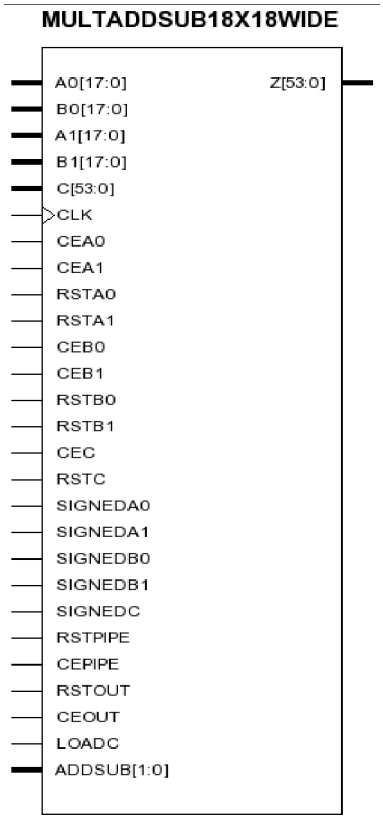


Table 398: Input Ports

Name	Range	Description
A0	17:0	Operand A0
B0	17:0	Operand B0
A1	17:0	Operand A1
B1	17:0	Operand B1
C	53:0	Operand C
CLK	1	Clock

Table 398: Input Ports (Continued)

Name	Range	Description
CEA0	1	Input A0 clock enable
CEA1	1	Input A1 clock enable
RSTA0	1	Input A0 register reset
RSTA1	1	Input A1 register reset
CEB0	1	Input B0 clock enable
CEB1	1	Input B1 clock enable
RSTB0	1	Input B0 register reset
RSTB1	1	Input B1 register reset
CEC	1	Input C clock enable
RSTC	1	Input C reset
SIGNEDA0	1	Operand A0 signed
SIGNEDA1	1	Operand A1 signed
SIGNEDB0	1	Operand B0 signed
SIGNEDB1	1	Operand B1 signed
SIGNEDC	1	Operand C signed
RSTPIPE	1	Pipeline register reset
CEPIPE	1	Pipeline clock enable
RSTOUT	1	Output register reset
CEOUT	1	Output clock enable
LOADC	1	Load value from input C (active high)
ADDSUB	1:0	Two bits ADD/SUB control (0=add, 1=subtract)

Table 399: Output Ports

Name	Range	Description
Z	53:0	Multiplication result

Table 400: Parameters

Name	Values	Description
REGINPUTAB0	"REGISTER" (default) "BYPASS"	Register operand A/B 0
REGINPUTAB1	"REGISTER" (default) "BYPASS"	Register operand A/B 1
REGINPUTC	"REGISTER" (default) "BYPASS"	Register operand C
REGADDSUB	"REGISTER" (default) "BYPASS"	Register ADDSUB
REGLOADC	"REGISTER" (default) "BYPASS"	Register LOADC
REGLOADC2	"REGISTER" (default) "BYPASS"	Register LOADC, second stage
REGPIPELINE	"REGISTER" (default) "BYPASS"	Register between multiplier and accumulator
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"DISABLED" *(default)* "ENABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	Synchronous/ asynchronous reset control

Note:

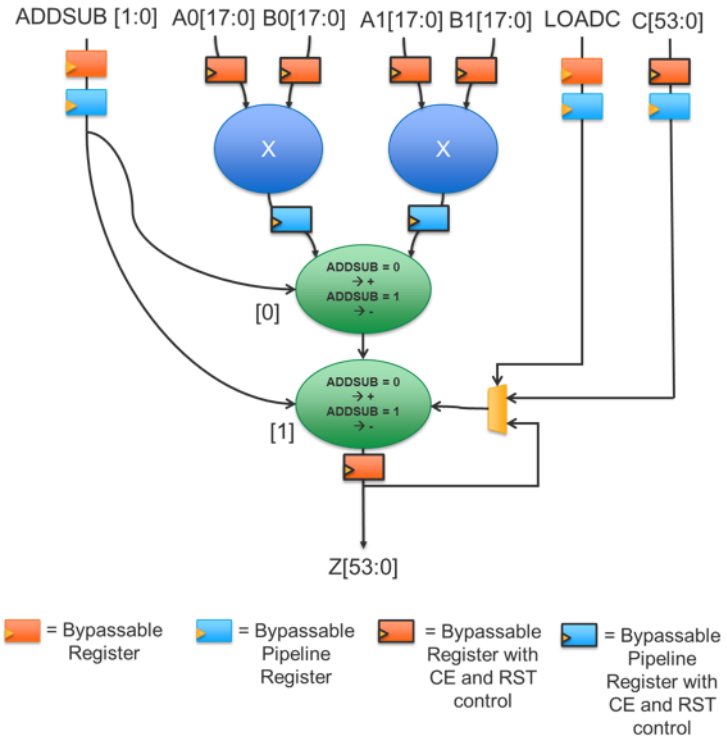
If LOADC = 1:
Z[53:0] = C[53:0] +/- (A0[17:0] x B0[17:0]) +/- (A1[17:0] x B1[17:0])

Else: Z[53:0] = Z_{prev}[53:0] +/- (A0[17:0] x B0[17:0]) +/- (A1[17:0] x B1[17:0])

If LOADC = 0, REGOUTPUT must be REGISTERED. Else, will result into a combinatorial lock.

ADDSUB[1] controls add/sub between C/Zprev and A0B0.

ADDSUB[0] controls add/sub between A0B0 and A1B1.



MULTADDSUB18X36

18x36 Multiplier and Adder/Subtractor

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

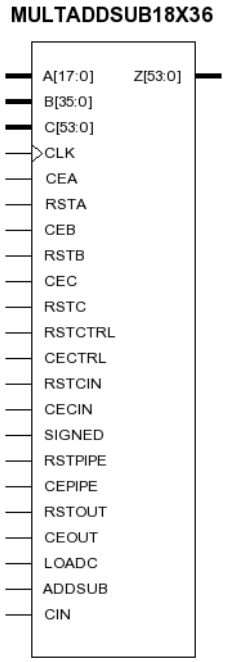


Table 401: Input Ports

Name	Range	Description
A	17:0	Operand A
B	35:0	Operand B
C	53:0	Operand C
CLK		Clock
CEA		Input A clock enable
RSTA		Input A register reset
CEB		Input B clock enable
RSTB		Input B register reset
CEC		Input C clock enable
RSTC		Input C register reset
RSTCTRL		Input control (addsub, m9addsub, signed, and loadc) reset
CECTRL		Input control (addsub, m9addsub, signed, and loadc) enable
RSTCIN		Input carryin reset
CECIN		Input carryin enable

Table 401: Input Ports (Continued)

Name	Range	Description
SIGNED		One SIGNED port for all inputs due to ACC54 hardware limitation
RSTPIPE		Pipeline register reset
CEPIPE		Pipeline clock enable
RSTOUT		Output register reset
CEOUT		Output clock enable
LOADC		1=Load value from input C
ADDSUB		0=add, 1=subtract
CIN		Carry in

Table 402: Output Ports

Name	Range	Description
Z	53:0	Multiplication result

Table 403: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register input A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register input B
REGINPUTC	"REGISTER" (default) "BYPASS"	Register input C
REGADDSUB	"REGISTER" (default) "BYPASS"	Register ADDSUB
REGLOADC	"REGISTER" (default) "BYPASS"	Register LOADC
REGLOADC2	"REGISTER" (default) "BYPASS"	Register LOADC, second stage
REGCIN	"REGISTER" (default) "BYPASS"	Register CIN
REGPIPELINE	"REGISTER" (default) "BYPASS"	Register between multiplier and accumulator
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output

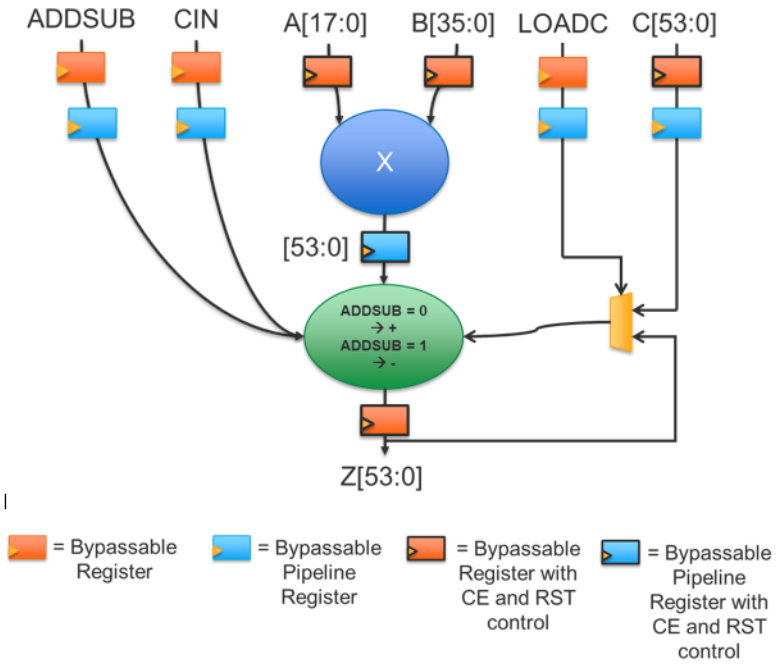
Table 403: Parameters

Name	Values	Description
GSR	"DISABLED" *(default)* "ENABLED"	Enable global set/reset
RESETMODE	"SYNC" (default)	

Note:

If LOADC = 1: Z[53:0] = C[53:0] +/- ((A[17:0] * B[35:0]) +/- CIN)

If LOADC = 0: Z[53:0] = Z_{prev}[53:0] +/- ((A[17:0] * B[35:0]) +/- CIN)



MULTADDSUB36X36

36x36 Multiplier and Adder/Subtractor

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

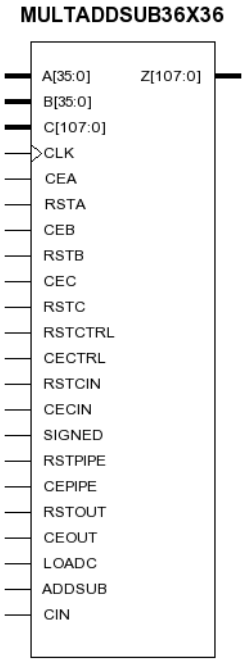


Table 404: Input Ports

Name	Range	Description
A	35:0	Operand A
B	35:0	Operand B
C	107:0	Operand C
CLK		Clock
CEA		Input A clock enable
RSTA		Input A register reset
CEB		Input B clock enable
RSTB		Input B register reset
CEC		Input C clock enable
RSTC		Input C register reset
RSTCTRL		Input control (addsub, m9addsub, signed, and loadc) reset
CECTRL		Input control (addsub, m9addsub, signed, and loadc) enable
RSTCIN		Input carry-in reset
CECIN		Input carry-in enable

Table 404: Input Ports (Continued)

Name	Range	Description
SIGNED		One SIGNED port for all inputs due to ACC54 HW limitation.
RSTPIPE		Pipeline register reset
CEPIPE		Pipeline clock enable
RSTOUT		Output register reset
CEOUT		Output clock enable
LOADC		1=Load value from input C
ADDSUB		0=add, 1=subtract
CIN		Carry in

Table 405: Output Ports

Name	Range	Description
Z	107:0	Multiplication result

Table 406: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register input A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register input B
REGINPUTC	"REGISTER" (default) "BYPASS"	Register input C
REGADDSUB	"REGISTER" (default) "BYPASS"	Register ADDSUB
REGLOADC	"REGISTER" (default) "BYPASS"	Register LOADC
REGLOADC2	"REGISTER" (default) "BYPASS"	Register LOADC, Second stage
REGCIN	"REGISTER" (default) "BYPASS"	Register CIN
REGPIPELINE	"REGISTER" (default) "BYPASS"	Register between multiplier and accumulator
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output

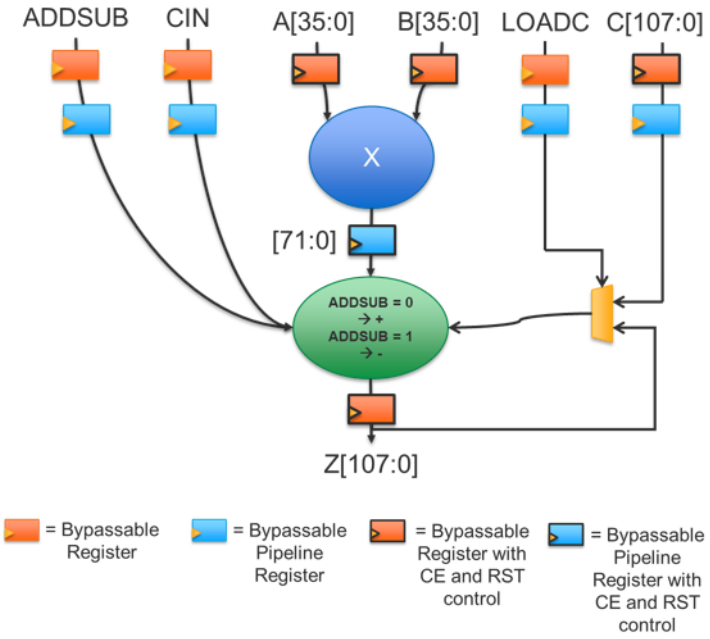
Table 406: Parameters (Continued)

Name	Values	Description
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(default)* "ASYNC"	Synchronous/asynchronous reset control. Reset release is always asynchronous.

Note:

If LOADC = 1: Z[107:0] = C[107:0] +/- ((A[35:0] * B[35:0]) +/- CIN)

If LOADC = 0: Z[107:0] = Z_{prev}[107:0] +/- ((A[35:0] * B[35:0]) +/- CIN)



MULTADDSUB9X9WIDE

9x9 Wide Multiplier and Adder/Subtractor

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

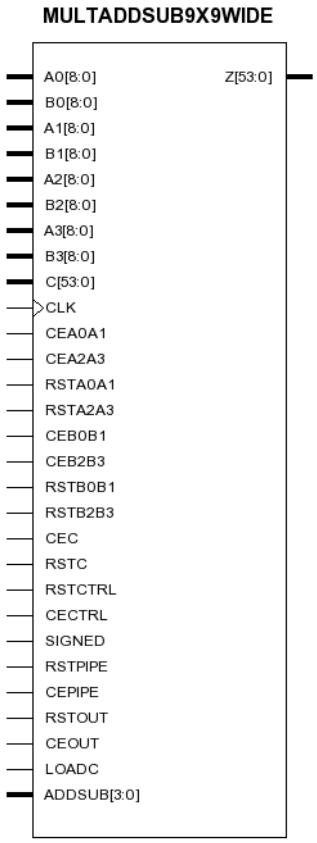


Table 407: Input Ports

Name	Range	Description
A0	8:0	Operand A0
B0	8:0	Operand B0
A1	8:0	Operand A1
B1	8:0	Operand B1
A2	8:0	Operand A2
B2	8:0	Operand B2
A3	8:0	Operand A3
B3	8:0	Operand B3
C	53:0	Operand C
CLK		Clock

Table 407: Input Ports (Continued)

Name	Range	Description
CEA0A1		Input A0 and A1 clock enable
CEA2A3		Input A2 and A3 clock enable
RSTA0A1		Input A0 and A1 register reset
RSTA2A3		Input A2 and A3 register reset
CEB0B1		Input B0 and B1 clock enable
CEB2B3		Input B2 and B3 clock enable
RSTB0B1		Input B0 and B1 register reset
RSTB2B3		Input B2 and B3 register reset
CEC		Input C clock enable
RSTC		Input C register reset
RSTCTRL		Input control (addsub, m9addsub, signed, and loadc) reset
CECTRL		Input control (addsub, m9addsub, signed, and loadc) enable
SIGNED		One SIGNED port for all inputs due to ACC54 HW limitation
RSTPIPE		Output register reset
CEPIPE		Output clock enable
RSTOUT		Output register reset
CEOUT		Output register clock enable
LOADC		1=Load value from input C
ADDSUB	3:0	0=add, 1=subtract

Table 408: Output Ports

Name	Range	Description
Z	53:0	Multiplication result

Table 409: Parameters

Name	Values	Description
REGINPUTAB0	"REGISTER" (default) "BYPASS"	Register operand A/B 0
REGINPUTAB1	"REGISTER" (default) "BYPASS"	Register operand A/B 1
REGINPUTAB2	"REGISTER" (default) "BYPASS"	Register operand A/B 2
REGINPUTAB3	"REGISTER" (default) "BYPASS"	Register operand A/B 3
REGINPUTC	"REGISTER" (default) "BYPASS"	Register operand C
REGADDSUB	"REGISTER" (default) "BYPASS"	Register ADDSUB
REGLOADC	"REGISTER" (default) "BYPASS"	Register LOADC
REGLOADC2	"REGISTER" (default) "BYPASS"	Register LOADC, Second stage
REGPIPELINE	"REGISTER" (default) "BYPASS"	Register between multiplier and accumulator
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" *(*default*)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(*default*)* "ASYNC"	Synchronous/asynchronous reset control. Reset release is always asynchronous.

Note:

If LOADC = 1:

Z[53:0] = C[53:0] +/- (((A0[8:0] * B0[8:0]) +/- (A1[8:0] * B1[8:0]))

+/- ((A2[8:0] * B2[8:0]) +/- (A3[8:0] * B3[8:0])))

If LOADC = 0:

Z[53:0] = Zprev[53:0] +/- (((A0[8:0] * B0[8:0]) +/- (A1[8:0] * B1[8:0]))

+/- ((A2[8:0] * B2[8:0]) +/- (A3[8:0] * B3[8:0])))

If LOADC = 0: REGOUTPUT must be **registered**; otherwise, it will result in a combinatorial lock.

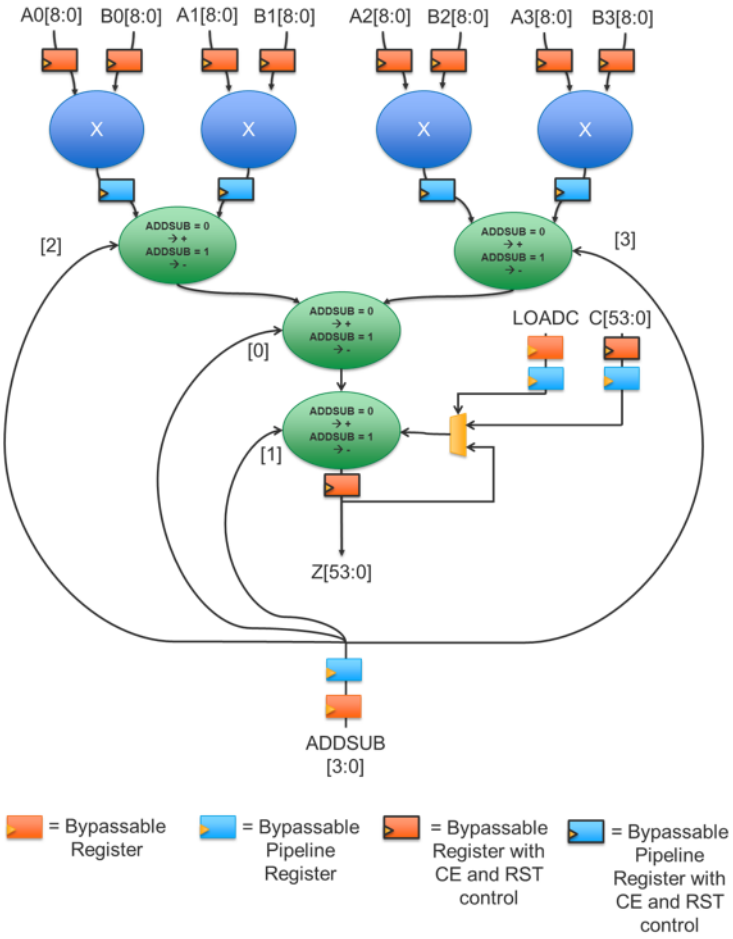
Bits ADDSUB[3:0] controls the ‘addition’ or ‘subtraction’ options:

ADDSUB[3]: controls selection of addition/subtraction between A2B2 and A3B3

ADDSUB[2]: controls selection of addition/subtraction between A0B1 and A1B1

ADDSUB[1]: controls selection of addition/subtraction between C/Z_{prev} and (A0B0 +/- A1B1)

ADDSUB[0]: controls selection of addition/subtraction between (A0B0 +/- A1B1) and (A2B2 +/- A3B3)



MULTIBOOT

Wrapper for Interface for Multiboot Functionality

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

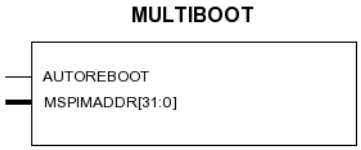


Table 410: Input Ports

Name	Range	Description
AUTOREBOOT		
MSPIMADDR	31:0	

Table 411: Parameters

Name	Values	Description
MSPIADDR	0’b00000000000000000000000000000000 *(default)*	
SOURCESEL	"DIS" *(default)* "EN"	

MULTPREADD18X18

18x18 Multiplier with Pre-Adder

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL

- UT24C
- UT24CP

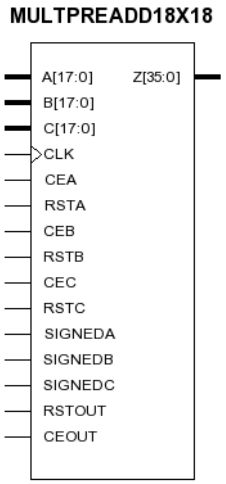


Table 412: Input Ports

Name	Range	Description
A	17:0	Operand A
B	17:0	Operand B
C	17:0	Operand C
CLK		Clock
CEA		Input clock enable
RSTA		Input register reset
CEB		Input clock enable
RSTB		Input register reset
CEC		Input clock enable
RSTC		Input register reset
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
SIGNEDC		Operand C signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 413: Output Ports

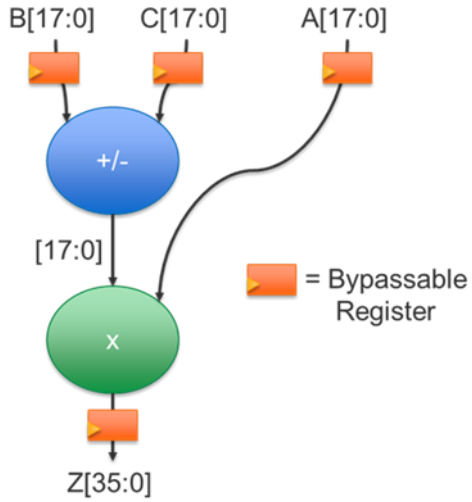
Name	Range	Description
Z	17:0	Multiplication result

Table 414: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGINPUTC	"REGISTER" (default) "BYPASS"	Register operand C
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(default)*	

Note:

Implements Z[36:0] = (B[17:0] + C[17:0]) * A[17:0]



MULTPREADD9X9

9x9 Multiplier with Pre-adder

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

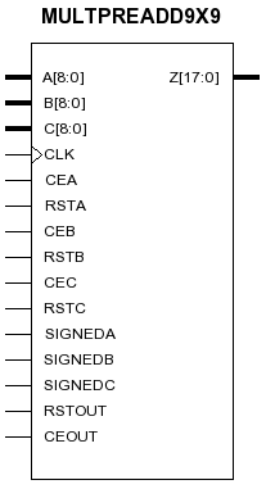


Table 415: Input Ports

Name	Range	Description
A	8:0	Operand A
B	8:0	Operand B
C	8:0	Operand C
CLK		Clock
CEA		Input clock enable
RSTA		Input register reset
CEB		Input clock enable
RSTB		Input register reset
CEC		Input clock enable
RSTC		Input register reset

Table 415: Input Ports (Continued)

Name	Range	Description
SIGNEDA		Operand A signed
SIGNEDB		Operand B signed
SIGNEDC		Operand C signed
RSTOUT		Output register reset
CEOUT		Output clock enable

Table 416: Output Ports

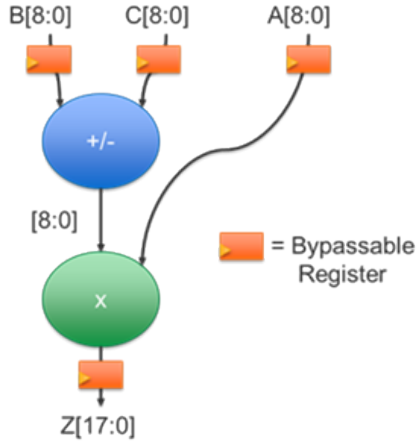
Name	Range	Description
Z	17:0	Multiplication result

Table 417: Parameters

Name	Values	Description
REGINPUTA	"REGISTER" (default) "BYPASS"	Register operand A
REGINPUTB	"REGISTER" (default) "BYPASS"	Register operand B
REGINPUTC	"REGISTER" (default) "BYPASS"	Register operand C
REGOUTPUT	"REGISTER" (default) "BYPASS"	Register output
GSR	"ENABLED" *(*default*)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(*default*)* "ASYNC"	Synchronous/asynchronous reset control. Reset release is always asynchronous.

Note:

Z[17:0] = A[8:0] * (B[8:0] + C[8:0])



OB

Output Buffer

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

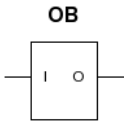


Table 418: Input Ports

Name	Description
I	Data from fabric

Table 419: Output Ports

Name	Description
O	Data to pad

In iCE40 UltraPlus, refer to Source Template.

Table 420: OB Truth Table

Input	Output
I	O
1	1
0	0
Z	U

OB_RGB

Architectures supported:

- iCE40 UltraPlus

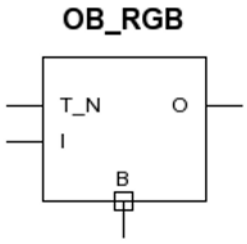


Table 421: Input Ports

Name	Range	Description
T_N		Tri-state control, active low
I		Data to pad

Table 422: Output Ports

Name	Range	Description
O		Data from pad

Table 423: Bidirectional Ports

Name	Range	Description
B		Pad

OBZ

Output Buffer with Tri-state

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

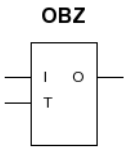


Table 424: Input Ports

Name	Description
I	Data to pad
T	Tri-state control

Table 425: Output Ports

Name	Description
O	Output from pad

Table 426: OBZ Truth Table

Inputs		Output
I	T	O
X	1	Z
0	0	0
1	0	1

X = Don’t care

OBZ_B

Output Buffer with Tri-state

Architectures supported:

- iCE40 UltraPlus

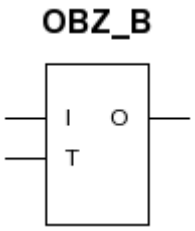


Table 427: Input Ports

Name	Range	Description
I		Data from fabric
T_N		Tri-state control, active low meaning T_N=0 > O = Z

Table 428: Output Ports

Name	Range	Description
O		Data to pad

Table 429: OBZ_B Truth Table

Inputs		Output
I	T_N	O
X	0	Z
0	1	0
1	1	1

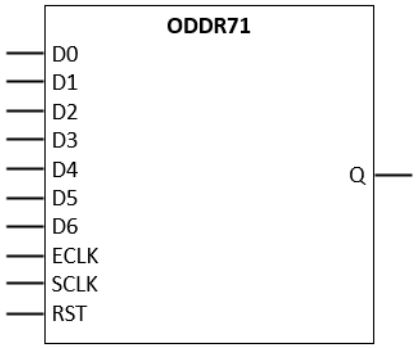
X = Don't care

ODDR71

7:1 LVDS ODDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for 7:1 LVDS ODDR implementation.

Table 430: Input Ports

Name	Range	Description
D	6:0	Parallel data input to the ODDR. D0 is sent out first and D6 last.
ECLK		Fast edge clock
SCLK		Primary clock input (divide-by-3.5 of ECLK).
RST		Reset to DDR registers.

Table 431: Output Ports

Name	Range	Description
Q		DDR data output on both edges of ECLK.

Table 432: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDR71A

7:1 LVDS ODDR implementation

Architectures supported:

- LAV-AT-ES1
- LN2-CT

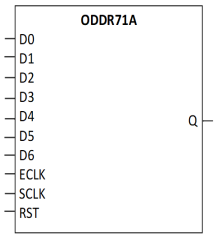


Table 433: Input Ports

Name	Range	Type	Active State	Description
D0	1	Data	High	DDR data input
D1	1	Data	High	DDR data input
D2	1	Data	High	DDR data input
D3	1	Data	High	DDR data input
D4	1	Data	High	DDR data input
D5	1	Data	High	DDR data input
D6	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-2 of ECLK)
RST	1	Control	High	Reset to DDR registers

Table 434: Output Ports

Name	Range	Type	Active State	Description
Q	1	Data	High	DDR data output on both edges of SCLK D0, D2, D4, D6: Positive edge of ECLK D1, D3, D5: Negative edge of ECLK

Table 435: Parameters

Name	Values	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

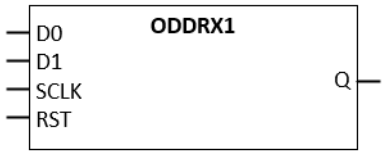
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX1

Generic X1 ODDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used to generate the address, command, and control except for CS_N for DDR3/DDR3L memory interfaces using X2 gearing.

Table 436: Input Ports

Name	Range	Description
D0, D1		Parallel data input to ODDR. D0 is sent out first, then D1.
SCLK		Primary clock input.
RST		Reset to DDR registers.

Table 437: Output Ports

Name	Range	Description
Q		DDR data output on both edges of SCLK.

Table 438: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

ODDRX1A

Generic x1 ODDR

Architectures supported:

- LAV-AT-ES1
- LN2-CT

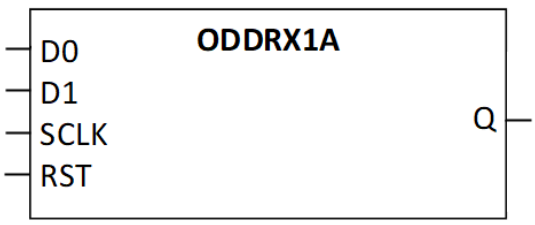


Table 439: Input Ports

Name	Range	Type	Active State	Description
D0	1	Data	High	DDR data input
D1	1	Data	High	DDR data input
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-5 of ECLK)
RST	1	Control	High	Reset to DDR registers

Table 440: Output Ports

Name	Range	Type	Active State	Description
Q	1	Data	High	DDR data output on both edges of SCLK D0: Positive edge of SCLK D1: Negative edge of SCLK

Table 441: Parameters

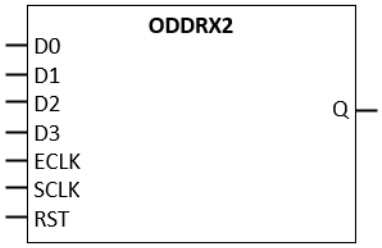
Name	Values	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

ODDRX2

Generic X2 ODDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used to generate the address, command, and control except for CS_N for DDR3/DDR3L memory interfaces using X4 gearing.

Table 442: Input Ports

Name	Range	Description
D	3:0	Parallel data input to the ODDR. D0 is sent out first and D3 last.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset to DDR registers.

Table 443: Output Ports

Name	Range	Description
Q		DDR data output on both edges of ECLK.

Table 444: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX2A

Generic x2 ODDR implementation

Architectures supported:

- LAV-AT-ES1
- LN2-CT

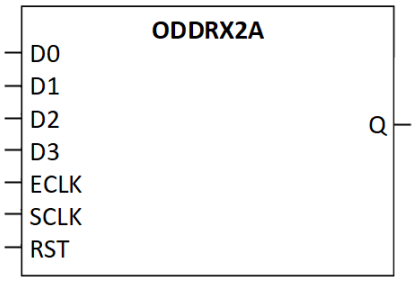


Table 445: Input Ports

Name	Range	Type	Active State	Description
D0	1	Data	High	DDR data input
D1	1	Data	High	DDR data input
D2	1	Data	High	DDR data input
D3	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-2 of ECLK)
RST	1	Control	High	Reset to DDR registers

Table 446: Output Ports

Name	Range	Type	Active State	Description
Q	1	Data	High	DDR data output on both edges of SCLK D0, D2: Positive edge of ECLK

Table 447: Parameters

Name	Values	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

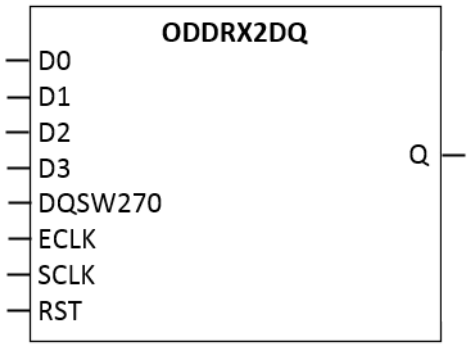
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX2DQ

DQ output for DDR2 & DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the DQ data output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces. It is also used to generate the CA bus for LPDDR2/LPDDR3 interfaces.

Table 448: Input Ports

Name	Range	Description
D	3:0	Data input to the ODDR. D0 is output first, D3 last.
DQSW270		Clock that is 90° ahead of clock used to generate the DQS output.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset to DDR registers.

Table 449: Output Ports

Name	Range	Description
Q		DDR data output on both edges of DQSW270

Table 450: ODDRX2DQ Parameters

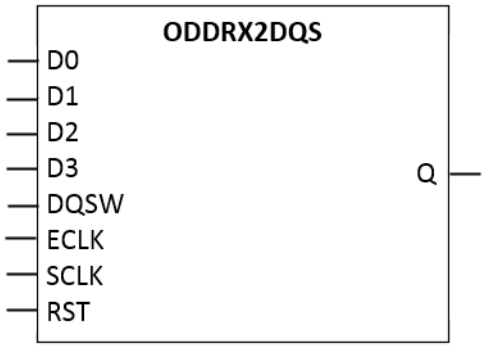
Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

ODDRX2DQS

DQS Output for DDR2 & DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the DQS data output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces. It is also used to generate CK and control signals (CKE, CS_N, ODT) for LPDDR2/LPDDR3 memory interfaces in X2 gearing mode.

Table 451: Input Ports

Name	Range	Description
D	3:0	Data input to the ODDR. D0 is output first, D3 last.
DQSW		DQSW includes write leveling phase shift from ECLK.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset to DDR registers.

Table 452: Output Ports

Name	Range	Description
Q		DDR data output on both edges of DQSW.

Table 453: ODDRX2DQS Parameters

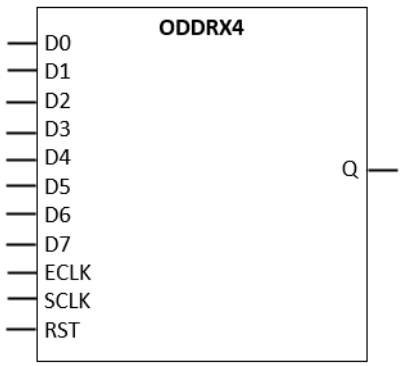
Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

ODDRX4

Generic X4 ODDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for Generic x4 ODDR implementation.

Table 454: Input Ports

Name	Range	Description
D	7:0	Parallel data input to the ODDR. D0 is sent out first and D7 last.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset to DDR registers.

Table 455: Output Ports

Name	Range	Description
Q		DDR data output on both edges of ECLK.

Table 456: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX4A

Generic x4 ODDR implementation

Architectures supported:

- LAV-AT-ES1
- LN2-CT

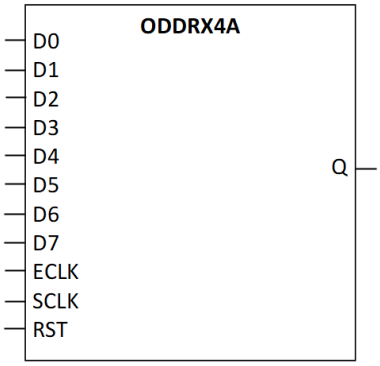


Table 457: Input Ports

Name	Range	Type	Active State	Description
D0	1	Data	High	DDR data input
D1	1	Data	High	DDR data input
D2	1	Data	High	DDR data input
D3	1	Data	High	DDR data input
D4	1	Data	High	DDR data input
D5	1	Data	High	DDR data input
D6	1	Data	High	DDR data input
D7	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-4 of ECLK)
RST	1	Control	High	Reset to DDR registers

Table 458: Output Ports

Name	Range	Type	Active State	Description
Q	1	Data	High	DDR data output on both edges of SCLK D0, D2, D4, D6: Positive edge of ECLK D1, D3, D5, D7: Negative edge of ECLK

Table 459: Parameters

Name	Values	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

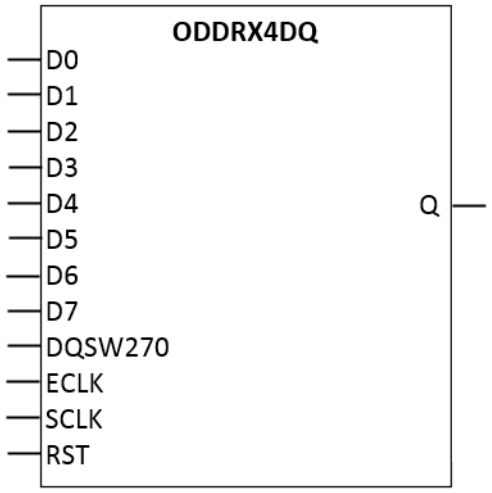
- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX4DQ

DQ output for DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate DQ data output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces using x4 gearing. It is also used to generate the CA bus for LPDDR2/LPDDR3 interfaces using X4 gearing.

Table 460: Input Ports

Name	Range	Description
D	7:0	Data input to the ODDR. D0 is output first, D7 last.
DQSW270		Clock that is 90° ahead of DQSW used to generate DQ. DQSW includes write leveling phase shift from ECLK.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset to DDR registers.

Table 461: Output Ports

Name	Range	Description
Q		DDR data output on both edges of DQSW270

Table 462: Parameters

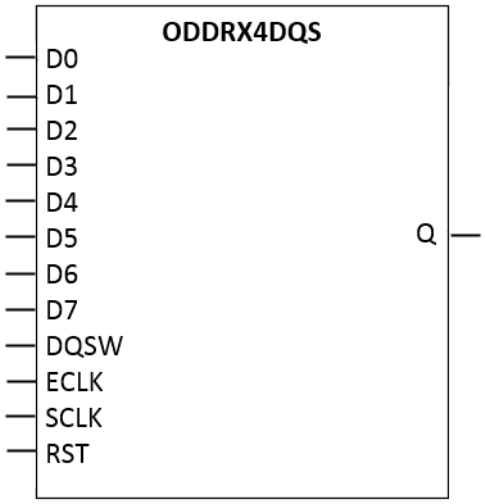
Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

ODDRX4DQS

DQS Output for DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate DQS clock output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces using x4 gearing. It is also used to generate CK and control (CKE, CS_N, ODT) for LPDDR2/LPDDR3 memory interfaces using x4 gearing.

Table 463: Input Ports

Name	Range	Description
D	7:0	Data input to the ODDR. D0 is output first, D7 last.
DQSW		DQSW includes write leveling phase shift from ECLK.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset to DDR registers.

Table 464: Output Ports

Name	Range	Description
Q		DDR data output on both edges of DQSW.

Table 465: ODDRX4DQS Parameters

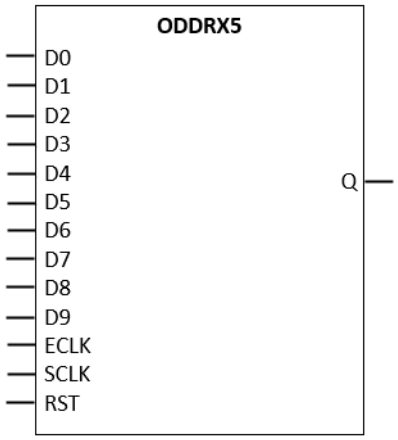
Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

ODDRX5

Generic X5 ODDR

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP



This primitive is used for Generic x5 ODDR implementation.

Table 466: Input Ports

Name	Range	Description
D	9:0	Parallel data input to the ODDR. D0 is sent out first and D9 last.
ECLK		Fast edge clock.
SCLK		Primary clock input (divide-by-5 of ECLK).
RST		Reset to DDR registers.

Table 467: Output Ports

Name	Range	Description
Q		DDR data output on both edges of ECLK.

Table 468: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

ODDRX5A

Generic x5 ODDR implementation

Architectures supported:

- LAV-AT-ES1
- LN2-CT

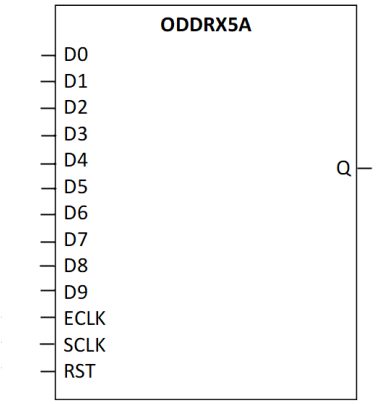


Table 469: Input Ports

Name	Range	Type	Active State	Description
D0	1	Data	High	DDR data input
D1	1	Data	High	DDR data input
D2	1	Data	High	DDR data input
D3	1	Data	High	DDR data input
D4	1	Data	High	DDR data input
D5	1	Data	High	DDR data input
D6	1	Data	High	DDR data input

Table 469: Input Ports (Continued)

Name	Range	Type	Active State	Description
D7	1	Data	High	DDR data input
D2	1	Data	High	DDR data input
D3	1	Data	High	DDR data input
ECLK	1	Clock	Rising Edge	Fast edge clock
SCLK	1	Clock	Rising Edge	Primary clock input (divide-by-5 of ECLK)
RST	1	Control	High	Reset to DDR registers

Table 470: Output Ports

Name	Range	Type	Active State	Description
Q	1	Data	High	DDR data output on both edges of SCLK D0, D2, D4, D6, D8: Positive edge of ECLK D1, D3, D5, D7, D9: Negative edge of ECLK

Table 471: Parameters

Name	Values	Description
GSR	"ENABLED" "DISABLED"	Enable or disable GSR feature

Note

Avoid using IDDRX2/ODDRX2 and above gearing on a BIDI I/O.

- Only IDDRX1 and ODDRX1 can be used with a single BIDI I/O.
- If using IDDRX2 and above gearing, this must be used only with an INPUT.
- If using ODDRX2 and above gearing, this must be used only with an OUTPUT.

OFD1P3AZ

Positive edge triggered output D flip-flop with positive level enable.

Architectures supported:

- iCE40 UltraPlus

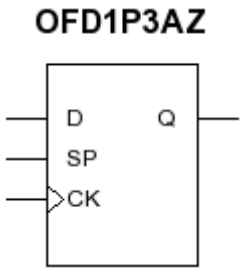


Table 472: Input Ports

Name	Range	Description
D		Data in
SP		Clock enable, active high
CK		Clock

Table 473: Output Ports

Name	Range	Description
Q		Data out

Table 474: OFD1P3AZ Truth Table

Inputs			Output
D	SP	CK	Q
X	0	X	Q
0	1	↑	0
1	1	↑	1

X = Don't care

OFD1P3BX

Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

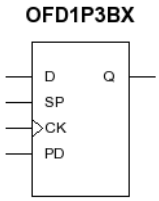


Table 475: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
CK	1	Clock	Rising Edge	Clock Input
SP	1	Control	High	Positive Clock Enable
PD	1	Control	High	Asynchronous Preset

Table 476: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data output

Table 477: OFD1P3BX Parameters

Name	Values	Description
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset

Table 478: OFD1P3BX Truth Table

Inputs				Output
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)

OFD1P3DX

Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

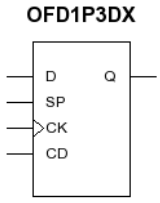


Table 479: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
SP	1	Clock	Rising Edge	Clock Input
CK	1	Control	High	Positive Clock Enable
CD	1	Control	High	Asynchronous Reset

Table 480: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data out to IO

Table 481: OFD1P3DX Parameters

Name	Values	Description
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset

Table 482: OFD1P3DX Truth Table

Inputs				Output
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)

OFD1P3IX

Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous clear and Positive Level Enable (Clear Overrides Enable)

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

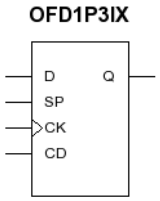


Table 483: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
SP	1	Clock	Rising Edge	Clock Input
CK	1	Control	High	Positive Clock Enable
CD	1	Control	High	Asynchronous Reset

Table 484: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data out to IO

Table 485: OFD1P3IX Parameters

Name	Values	Description
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset

Table 486: OFD1P3IX Truth Table

Inputs				Output
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)

OFD1P3JX

Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset Overrides Enable)

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

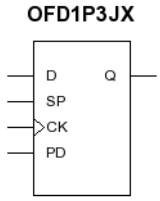


Table 487: Input Ports

Name	Range	Type	Active State	Description
D	1	Data	High	Data Input
SP	1	Clock	Rising Edge	Clock Input
CK	1	Control	High	Positive Clock Enable
CD	1	Control	High	Asynchronous Reset

Table 488: Output Ports

Name	Range	Type	Active State	Description
Q	1		High	Data out to IO

Table 489: OFD1P3JX Parameters

Name	Values	Description
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset

Table 490: OFD1P3JX Truth Table

Inputs				Output
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)

OSC

Wrapper for Oscillator Block

It includes a trimmed low frequency oscillator, and a trimmed high frequency oscillator.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

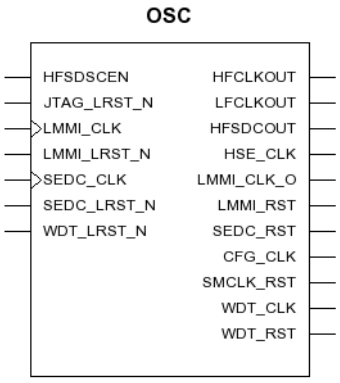


Table 491: Input Ports

Name	Description
HFSDSCEN	SED/SEC enable from fabric
JTAG_LRST_N	JTAG local reset input from fabric (Active Low)
LMMI_CLK	LMMI clock input from fabric
LMMI_LRST_N	LMMI local reset input from fabric (Active Low)
SEDC_CLK	SEDC Clock, hardware inferred from OSC.HFSDCOUT
SEDC_LRST_N	SEDC local reset input from fabric (Active Low)
WDT_LRST_N	Watch Dog Timer Local Reset from fabric (Active Low)

Table 492: Output Ports

Name	Description
HFCLKOUT	High frequency clock output to PCLK tree
LFCLKOUT	Low frequency clock output to PCLK tree
HFSDCOUT	SED/SEC clock output, hardware inferred to OSC.SED_CLK
HSE_CLK	HSE clock output, hardware inferred to CRE.CRE_CLK
LMMI_CLK_O	LMMI clock output to CONFIG_LMMI(A).LMMICLK
LMMI_RST	LMMI reset output to CONFIG_LMMI(A).LMMIRESETN
SEDC_RST	SEDC reset output to SEDC.RSTSEDC
CFG_CLK	Config clock output to CONFIG_LMMI.SMCLK, CRE.CFG_CLK, JTAG.SMCLK
SMCLK_RST	Config clock reset output to CONFIG_LMMI.RSTSMCLK
WDT_CLK	Watchdog timer clock output to WDT.WDT_CLK
WDT_RST	Watchdog timer reset output to WDT.WDT_RST

Table 493: Parameters

Name	Values	Description
DTR_EN	"ENABLED" (default) "DISABLED"	Digital Temperature Readout block enable from MIB
HF_CLK_DIV	"1" (default) "1" - "255"	User assignable high frequency oscillator output clock divider configuration (div2~div256)
HF_SED_SEC_DIV	"1" (default) "1" - "255"	User assignable high frequency oscillator output SED and SEC clock divider configuration (div2~div256)
HF_FABRIC_EN	"DISABLED" (default) "ENABLED"	High frequency oscillator trim input source mux select (for TEST only)
HF_OSC_EN	"ENABLED" (default) "DISABLED"	High frequency oscillator output enable, controlled by the user
HFDIV_FABRIC_EN	"ENABLED" (default) "DISABLED"	High frequency oscillator divider configuration mux select, fabric divider enable
LF_FABRIC_EN	"DISABLED" (default) "ENABLED"	Low frequency oscillator trim input source mux select, fabric driven enable (for TEST only)

Table 493: Parameters

Name	Values	Description
LF_OUTPUT_EN	"DISABLED" (default) "ENABLED"	Low frequency clock output enable
DEBUG_N	"DISABLED" (default) "ENABLED"	Oscillator debug mode enable
MCJTAGGSRNDIS	"EN" (default) "DIS"	JTAG GSR enable/disable control
MCLMMIGSRNDIS	"EN" (default) "DIS"	CONFIG_LMMI GSR enable/disable control
MCSEDCGSRNDIS	"EN" (default) "DIS"	SEDC GSR enable/disable control
MCWDTGSRNDIS	"EN" (default) "DIS"	WDT GSR enable/disable control
SMCLK_DIV	"3" (default) "3" - "10"	Config clock divider for simulation only

Note:

The timer uses these device constraint equations to know what the periods of the output clocks of the Oscillator will be based on user design settings.

The numerical values in the equations have units of picoseconds.

For LFCLKOUT (created clock): ADJUSTED_PERIOD=31250000

For I2CCKOUT (created clock): ADJUSTED_PERIOD=7812500

For HFCLKOUT (created clock): ADJUSTED_PERIOD=6666.66667 * HF_CLK_DIV

For HFSDCOUT (created clock): ADJUSTED_PERIOD=6666.66667 * HF_SED_SEC_DIV

OSCA

Internal Oscillator

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

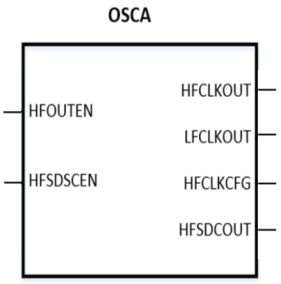


Table 494: Input Ports

Name	Description
HFOUTEN	High frequency clock output enable
HFSDSCEN	SED/SEC enable from fabric

Table 495: Output Ports

Name	Description
HFCLKOUT	High frequency clock output
LFCLKOUT	Low Frequency clock output
HFCLKCFG	High Frequency Reference Clock; 450 MHz
HFSDCOUT	450 MHz with Programmable Divider (2~256) to User for SED/SEC Application

Table 496: Parameters

Name	Values	Description	Inferable
HF_CLK_DIV	1 (default)	HF oscillator output divider configuration (div2~div256)	No
HF_SED_SEC_DIV	1 (default)	HF oscillator output divider configuration (div2~div256)	No

Table 496: Parameters

Name	Values	Description	Inferable
HF_OSC_EN	"ENABLED" (default)	HF oscillator enable	No
LF_OUTPUT_EN	"DISABLED" (default)	Low frequency clock output enable	No

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 OSCA or Sysbus (with sys_clk_sel=OSC option). OSCA may be used as a general-purpose clock to drive FPGA logic.

OSCE

High-frequency oscillator

Architectures supported:

- LAV-AT-ES1
- LN2-CT

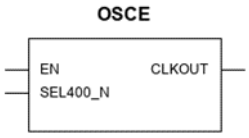


Table 497: Input Ports

Name	Description
EN	Enable CLKOUT
SEL400_N	Switch between 400 and 320 MHz

Table 498: Output Ports

Name	Description
CLKOUT	Output clock to fabric

Table 499: Parameters

Name	Values	Description
CLK_DIV	"1" (default) Integer string "1" to "256"	CLKOUT divider value

OSCD

High-frequency oscillator

Architectures supported:

- LIFCL-33U

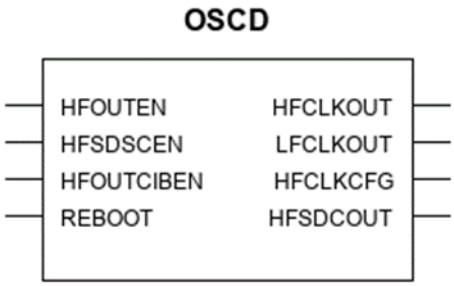


Table 500: Input Ports

Name	Description
HFOUTEN	HF clock (225MHz) output enable: just for test
HFSDSCEN	HF user clock output enable
HFOUTCIBEN	CIB control to turn on/off HF-OSC, function during user mode with mc1_en_hf_osc_cib high
REBOOT	CIB control to enable/disable hf_clk_config output, function during user mode with mc1_en_hf_config_on_cib high

Table 501: Output Ports

Name	Description
HFCLKOUT	450MHz with programmable divider (2~256) to user
LFCLKOUT	Low frequency clock output after div4: 32KHz
HFCLKCFG	450MHz going to Config block. Config Block would generate the clk_config, as well as the other blocks it needs, such as SPI or SED clocks
HFSDCOUT	450MHz with programmable divider (2~256) to CFG

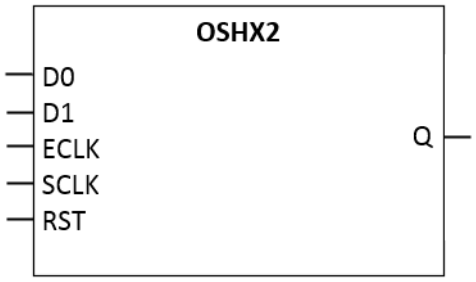
Table 502: Parameters

Name	Description
DTR_EN	DTR block enable from MIB
HF_CLK_DIV	User assignable HF oscillator output divider configuration (div2~div256)
HF_SED_SEC_DIV	User assignable HF oscillator output divider configuration (div2~div256)
HF_FABRIC_EN	High frequency oscillator trim source mux select
HF_OSC_EN	HF oscillator enable, controlled by the user
LF_FABRIC_EN	Low frequency oscillator trim source mux select, fabric driven trim enable
LF_OUTPUT_EN	Low frequency clock output enable
DEBUG_N	Ignore/enable the SLEEP/STOP function during the USER mode. 0: Ignore the SLEEP/STOP mode

OSHX2

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the CS_N control for DDR3/DDR3L memory with x2 gearing.

Table 503: Input Ports

Name	Range	Description
D0, D1		Data input. D0 is output first, followed by D1
ECLK		ECLK input.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset input.

Table 504: Output Ports

Name	Range	Description
Q		Address and command output.

Table 505: OSHX2 Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	"ENABLED" "DISABLED"	ENABLED

OSHX4

Architectures supported:

- LFCPNX
- LFD2NX

- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP

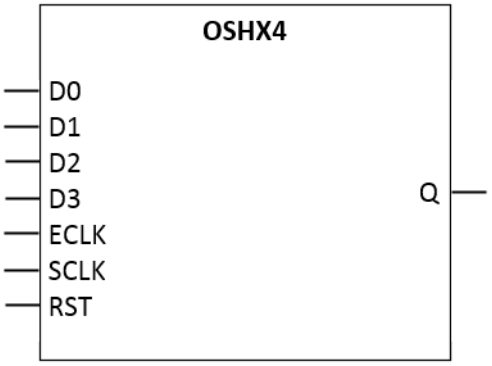


Table 506: Input Ports

Name	Range	Description
D	3:0	Data input. D0 is output first, D3 last.
ECLK		ECLK input.
SCLK		Primary clock input (divide-by-4 of ECLK)
RST		Reset input.

Table 507: Output Ports

Name	Range	Description
Q		Address and command output.

This primitive is used to generate the CS_N control for DDR3 memory with x4 gearing and write leveling.

Table 508: Parameters

Name	Description	Values	Default
GSR	Enable global set/reset.	ENABLED DISABLED	ENABLED

OUTDELAYA

Add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

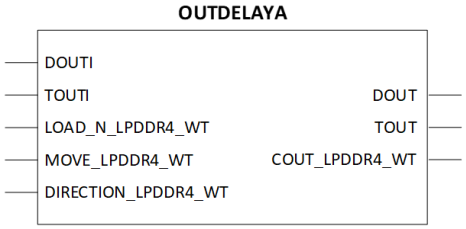


Table 509: Input Ports

Name	Description
DOUTI	The input for DOUT
TOUTI	The input for TOUT
LOAD_N_LPDDR4_WT	For LPDDR4 write training, asynchronous reset the output final delay code to factory default value.
MOVE_LPDDR4_WT	For LPDDR4 write training, move pulse's rising edge will change the code according to direction value.
DIRECTION_LPDDR4_WT	For LPDDR4 write training fine delay, "0" to increase the code, "1" to decrease the code.

Table 510: Output Ports

Name	Description
DOUT	PIC data output, to ASICIO
TOUT	PIC tri-state output, to ASICIO
COUT_LPDDR4_WT	For LPDDR4 write training, write training margin test output flag to indicate the underflow or overflow.

Table 511: Parameters

Name	Values	Description
MARGIN_TEST_ENABLE	DISABLED (default) ENABLED	For LPDDR4 write training, margin test enable 0: disable margin test 1: enable margin test
DEL_VALUE	0…127 (default = 0)	For LPDDR4 write training, output fine delay fuse setting: 7b0000_0000: minimum delay 7b1111_1111: maximum delay

OUTDELAYB

Add static delay to DOUT and TOUT for write training in LPDDR4 mode.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

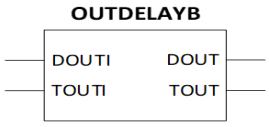


Table 512: Input Ports

Name	Description
DOUTI	The input for DOUT
TOUTI	The input for TOUT

Table 513: Output Ports

Name	Description
DOUT	PIC data output, to ASICIO
TOUT	PIC tri-state output, to ASICIO

Table 514: Parameters

Name	Values	Description
DEL_VALUE	0…127 (default = 0)	For LPDDR4 write training, output fine delay fuse setting: 7b0000_0000: minimum delay 7b1111_1111: maximum delay

OUTDELAYC

Add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

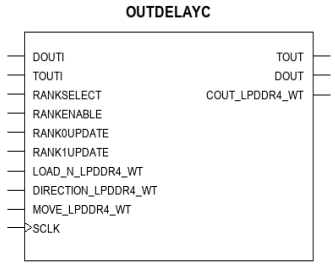


Table 515: Input Ports

Name	Description
DOUTI	The input for DOUT
TOUTI	The input for TOUT
LOAD_N_LPDDR4_WT	For LPDDR4 write training, asynchronous reset the output final code for factory default value.
MOVE_LPDDR4_WT	For LPDDR4 write training, move pulse's rising edge will change the code according to direction value.
DIRECTION_LPDDR4_WT	For LPDDR4 write training fine delay, "0" to increase the code, "1" to decrease the code.
RANKSELECT	0 - Select delay rank 0 1 - Select delay rank 1
RANKENABLE	0 - Select bypassed actual path delay code 1 - Select registered actual path delay code
RANK0UPDATE	0 - Rank - 0 registers not enabled 1 - Enable Rank - 0 registers
RANK1UPDATE	0 - Rank - 1 registers not enabled 1 - Enable Rank - 1 registers
SCLK	SCLK clock for inputs from CIB

Table 516: Output Ports

Name	Description
DOUT	PIC data output, to ASICIO
TOUT	PIC tri-state output, to ASICIO
COUT_LPDDR4_WT	For LPDDR4 write training, write training margin test output flag to indicate the underflow or overflow.

Table 517: Parameters

Name	Values	Description
MARGIN_TEST_ENABLE	ENABLED (default) DISABLED	For LPDDR4 write training, margin test enable 0: disable margin test 1: enable margin test
DEL_VALUE	0…127 (default = 0)	For LPDDR4 write training, output fine delay fuse setting: 7b0000_0000: minimum delay 7b1111_1111: maximum delay

PCIE

Wrapper for Hardened PCIE Interface

Note:

You should not try to use this primitive directly. Instead, use the related IP in IP Catalog. The following information is for reference only.

Architectures supported:

- LFD2NX
- LIFCL-40,LIFCL-17
- UT24C

PCIE

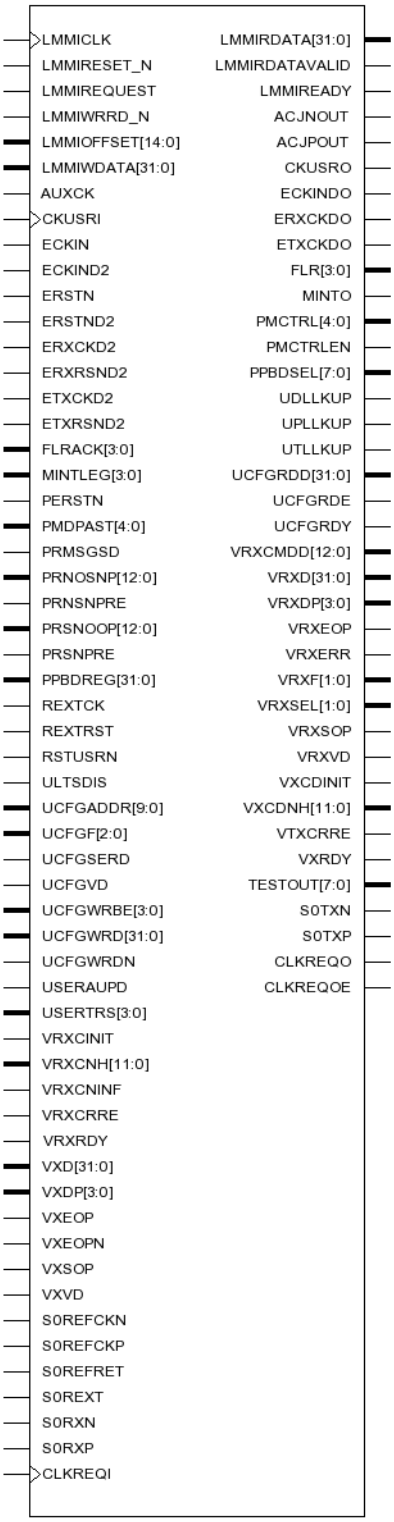

LMMICLK
LMMIRESET_N
LMMIREQUEST
LMMIWRRD_N
LMMIOFFSET[14:0]
LMMIWDATA[31:0]
AUXCK
CKUSRI
ECKIN
ECKIND2
ERSTN
ERSTND2
ERXCKD2
ERXRSND2
ETXCKD2
ETXRSND2
FLRACK[3:0]
MINTLEG[3:0]
PERSTN
PMDPAST[4:0]
PRMSGSD
PRNOSNP[12:0]
PRNSNPRE
PRSNOOP[12:0]
PRSNPRE
PPBDREG[31:0]
REXTCK
REXTRST
RSTUSRN
ULTSDIS
UCFGADDR[9:0]
UCFGF[2:0]
UCFGSERD
UCFGVD
UCFGWRBE[3:0]
UCFGWRD[31:0]
UCFGWRDN
USERAUPD
USERTRS[3:0]
VRXCINIT
VRXCNH[11:0]
VRXCNINF
VRXCRRE
VRXRDY
VXD[31:0]
VXDP[3:0]
VXEOP
VXEOPN
VXSOP
VXVD
S0REFCKN
S0REFCKP
S0REFRET
S0REXT
S0RXN
S0RXP
CLKREQI
LMMIRDATA[31:0]
LMMIRDATAVALID
LMMIREADY
ACJNOUT
ACJPOUT
CKUSRO
ECKINDO
ERXCKDO
ETXCKDO
FLR[3:0]
MINTO
PMCTRL[4:0]
PMCTRLEN
PPBDSEL[7:0]
UDLLKUP
UPLLKUP
UTLLKUP
UCFGRDD[31:0]
UCFGRDE
UCFGRDY
VRXCMDD[12:0]
VRXD[31:0]
VRXDP[3:0]
VRXEOP
VRXERR
VRXF[1:0]
VRXSEL[1:0]
VRXSOP
VRXVD
VXCDINIT
VXCDNH[11:0]
VTXCRRE
VXRDY
TESTOUT[7:0]
S0TXN
S0TXP
CLKREQO
CLKREQOE

Table 518: Input Ports

Name	Range	Description
LMMICLK		Clock
LMMIRESET_N		Reset
LMMIREQUEST		Start transaction
LMMIWRRD_N		Write = HIGH, Read = LOW
LMMIOFFSET	16:0	Word Address/Offset
LMMIWDATA	31:0	Write data
AUXCK		
CKUSRI		
ECKIN		
ECKIND2		
ERSTN		
ERSTND2		
ERXCKD2		
ERXRSND2		
ETXCKD2		
ETXRSND2		
FLRACK	3:0	
MINTLEG	3:0	
PERSTN		
PMDPAST	4:0	
PRMSGSD		
PRNOSNP	12:0	
PRNSNPRE		
PRSNOOP	12:0	
PRSNPRE		
PPBDREG	31:0	
REXTCK		
REXTRST		
RSTUSRN		
ULTSDIS		

Table 518: Input Ports (Continued)

Name	Range	Description
UCFGADDR	9:0	
UCFGF	2:0	
UCFGSERD		
UCFGVD		
UCFGWRBE	3:0	
UCFGWRD	31:0	
UCFGWRDN		
USERAUPD		
USERTRS	3:0	
VRXCINIT		
VRXCNH	11:0	
VRXCNINF		
VRXCRRE		
VRXRDY		
VXD	31:0	
VXDP	3:0	
VXEOP		
VXEOPN		
VXSOP		
VXVD		
S0REFCKN		
S0REFCKP		
S0REFRET		
S0REXT		
S0RXN		
S0RXP		
CLKREQI		

Table 519: Output Ports

Name	Range	Description
LMMIRDATA	31:0	Read data
LMMIRDATAVALID		Valid data indicator
LMMIREADY		Slave ready signal
ACJNOUT		
ACJPOUT		
CKUSRO		
ECKINDO		
ERXCKDO		
ETXCKDO		
FLR	3:0	
MINTO		
PMCTRL	4:0	
PMCTRLEN		
PPBDSEL	7:0	
UDLLKUP		
UPLLKUP		
UTLLKUP		
UCFGRDD	31:0	
UCFGRDE		
UCFGRDY		
VRXCMDD	12:0	
VRXD	31:0	
VRXDP	3:0	
VRXEOP		
VRXERR		
VRXF	1:0	
VRXSEL	1:0	
VRXSOP		
VRXVD		
VXCDINIT		
VXCDNH	11:0	

Table 519: Output Ports (Continued)

Name	Range	Description
VTXCRRE		
VXRDY		
TESTOUT	7:0	
S0TXN		
S0TXP		
CLKREQO		
CLKREQOE		

Table 520: Parameters

Name	Values	Description
A_CHNGD_MAX	"0’b100" (default)	
A0_FORCE	"DISABLED" (default)	
A0_FREEZE	"DISABLED" (default)	
A0_INIT	"0’b000000" (default)	
A0DIR_VAL	"DISABLED" (default)	
A1_FORCE	"DISABLED" (default)	
A1_FREEZE	"DISABLED" (default)	
A1_INIT	"0’b000000" (default)	
A1DIR_VAL	"DISABLED" (default)	
A2_FORCE	"DISABLED" (default)	
A2_FREEZE	"DISABLED" (default)	
A2_INIT	"0’b000000" (default)	
A2GAIN_CALIB	"0’b100110" (default)	
ACJTAG_REG	"0’b00" (default)	
ADDR_LIMIT_PRE_MTHD_CTRL	"0’b0100" (default)	
ADDR_LIMIT_TABLE_MTHD_CTRL	"0’b00101" (default)	
ADIR_OVR	"DISABLED" (default)	
ADV_CH_CD_SEL	"IMPLEMENT_CH" (default)	
ADV_TARGET_LINK_SPEED_USPORT_A	"DISABLED" (default)	
ADV_TARGET_LINK_SPEED_USPORT_B	"DISABLED" (default)	
ADV_TARGET_LINK_SPEED_USPORT_C	"DISABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
ADV_TARGET_LINK_SPEED_USPORT_D	"DISABLED" (default)	
ADVANCE	"DISABLED" (default)	
ALERT_ENABLE	"0’b000" (default)	
ALMOST_EMPTY_10B	"0’b001110" (default)	
ALMOST_EMPTY_20B	"0’b001111" (default)	
ALMOST_EMPTY_GEN3	"0’b001100" (default)	
ALMOST_FULL_10B	"0’b110000" (default)	
ALMOST_FULL_20B	"0’b011000" (default)	
ALMOST_FULL_GEN3	"0’b010110" (default)	
ALT_CLK_SEL_VCC	"PAD" (default)	
ARRAY_DA	"0’b0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000" (default)	
ARRAY_MT	"0’b0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000000 0000000000000000000000000000000000000010 0000000000111000000000000100000000000000001 0010000000000000110000000000000001010000000 000000100" (default)	
ARXCAL_OUT	"DISABLED" (default)	
ARXCDRICP_RATE0	"0’b011" (default)	
ARXCDRICP_RATE1	"0’b011" (default)	
ARXCDRICP_RATE2	"0’b011" (default)	
ARXDMPWRDWN	"DISABLED" (default)	
ARXDPPWRDN	"DISABLED" (default)	
ARXEOM_PWRDN	"POWERED_DOWN" (default)	
ARXICP_RATE0	"0’b011" (default)	
ARXICP_RATE1	"0’b011" (default)	
ARXICP_RATE2	"0’b011" (default)	
ARXOVR_OUT	"DISABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
ARXRSACTAT	"0’b0001" (default)	
ARXRSAPTAT	"0’b1000" (default)	
ARXRSVCTL	"0’b00000000" (default)	
ARXSEL_OUT	"DISABLED" (default)	
ASPM_L1_1_SUPPORTED	"SUPPORTED" (default)	
ASPM_L1_2_SUPPORTED	"SUPPORTED" (default)	
ASPM_SUPPORT	"L0S_AND_L1_SUPPORTED" (default)	
ATTENTION_BUTTON_PRESENT	"NOT_SUPPORTED" (default)	
ATTENTION_INDICATOR_PRESENT	"NOT_SUPPORTED" (default)	
ATXICP_RATE0	"0’b101" (default)	
ATXICP_RATE1	"0’b101" (default)	
ATXICP_RATE2	"0’b100" (default)	
AUTO_SHIFT	"ENABLED" (default)	
AUTONOMOUS_WIDTH_CHANGE	"OTHERWISE" (default)	
AUX_CLK_PERIOD	"0’b1111010000100100" (default)	
AUX_CURRENT	"SELF_POWERED" (default)	
AUXCLK1US_MAX	"0’b00001001" (default)	
AUXIDL_MAX	"0’b00000100" (default)	
BAR_INDEX_CFG0_A	"0’b000" (default)	
BAR_INDEX_CFG0_B	"0’b000" (default)	
BAR_INDEX_CFG0_C	"0’b000" (default)	
BAR_INDEX_CFG0_D	"0’b000" (default)	
BAR_INDEX_CFG1_A	"0’b001" (default)	
BAR_INDEX_CFG1_B	"0’b001" (default)	
BAR_INDEX_CFG1_C	"0’b001" (default)	
BAR_INDEX_CFG1_D	"0’b001" (default)	
BAR_INDEX_CFG2_A	"0’b010" (default)	
BAR_INDEX_CFG2_B	"0’b010" (default)	
BAR_INDEX_CFG2_C	"0’b010" (default)	
BAR_INDEX_CFG2_D	"0’b010" (default)	
BAR_INDEX_CFG3_A	"0’b011" (default)	
BAR_INDEX_CFG3_B	"0’b011" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
BAR_INDEX_CFG3_C	"0'b011" (default)	
BAR_INDEX_CFG3_D	"0'b011" (default)	
BAR_INDEX_CFG4_A	"0'b100" (default)	
BAR_INDEX_CFG4_B	"0'b100" (default)	
BAR_INDEX_CFG4_C	"0'b100" (default)	
BAR_INDEX_CFG4_D	"0'b100" (default)	
BAR_INDEX_CFG5_A	"0'b101" (default)	
BAR_INDEX_CFG5_B	"0'b101" (default)	
BAR_INDEX_CFG5_C	"0'b101" (default)	
BAR_INDEX_CFG5_D	"0'b101" (default)	
BIR_MSIX_PBA_A	"BAR0" (default)	
BIR_MSIX_PBA_B	"BAR0" (default)	
BIR_MSIX_PBA_C	"BAR0" (default)	
BIR_MSIX_PBA_D	"BAR0" (default)	
BIR_MSIX_TABLE_A	"BAR0" (default)	
BIR_MSIX_TABLE_B	"BAR0" (default)	
BIR_MSIX_TABLE_C	"BAR0" (default)	
BIR_MSIX_TABLE_D	"BAR0" (default)	
BYP_AVG	"USED" (default)	
BYPASS	"PERFORM_RECEIVER_DETECTION" (default)	
BYPASS_ADDR_DEC	"NORMAL" (default)	
C_RX_LIM_D	"DISABLED" (default)	
C_RX_LIM_H	"DISABLED" (default)	
C_TX_LIM_D	"DISABLED" (default)	
C_TX_LIM_H	"DISABLED" (default)	
CALIB_3DB	"ENABLED" (default)	
CALIB_SETTLE_MAX	"0'b001" (default)	
CALIB_STABLE_MAX	"0'b11000" (default)	
CAPABILITY_VERSION	"0'b0010" (default)	
CDR_ERR	"DISABLED" (default)	
CDR_P1	"CDR_PLL_RESET" (default)	
CDR_PLL_DELTA	"0P4_PERCENT" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
CDR_REFERENCE	"0'b00" (default)	
CDRPLL_CMP_MAX	"0'b00010100" (default)	
CDRPLL_CNT_MAX	"0'b00000100" (default)	
CDRPLL_PRE_RXEQ_COARSE_TIMER	"0'b01010000" (default)	
CDRPLL_PRE_RXEQ_FINE_TIMER	"0'b10110100" (default)	
CDRPLL_PST_RXEQ_COARSE_TIMER	"0'b01000000" (default)	
CDRPLL_PST_RXEQ_FINE_TIMER	"0'b01000010" (default)	
CFG_A_BAR0	"0'b11111111111111110000000000001100" (default)	
CFG_A_BAR1	"0'b11111111111111111111111111111111" (default)	
CFG_A_BAR2	"0'b11111111111111111110000000001100" (default)	
CFG_A_BAR3	"0'b11111111111111111111111111111111" (default)	
CFG_A_BAR4	"0'b11111111111111111110000000001100" (default)	
CFG_A_BAR5	"0'b11111111111111111111111111111111" (default)	
CFG_B_BAR0	"0'b11111111111111110000000000001100" (default)	
CFG_B_BAR1	"0'b11111111111111111111111111111111" (default)	
CFG_B_BAR2	"0'b11111111111111111110000000001100" (default)	
CFG_B_BAR3	"0'b11111111111111111111111111111111" (default)	
CFG_B_BAR4	"0'b11111111111111111110000000001100" (default)	
CFG_B_BAR5	"0'b11111111111111111111111111111111" (default)	
CFG_C_BAR0	"0'b11111111111111110000000000001100" (default)	
CFG_C_BAR1	"0'b11111111111111111111111111111111" (default)	
CFG_C_BAR2	"0'b11111111111111111110000000001100" (default)	
CFG_C_BAR3	"0'b11111111111111111111111111111111" (default)	
CFG_C_BAR4	"0'b11111111111111111110000000001100" (default)	
CFG_C_BAR5	"0'b11111111111111111111111111111111" (default)	
CFG_D_BAR0	"0'b11111111111111110000000000001100" (default)	
CFG_D_BAR1	"0'b11111111111111111111111111111111" (default)	
CFG_D_BAR2	"0'b11111111111111111110000000001100" (default)	
CFG_D_BAR3	"0'b11111111111111111111111111111111" (default)	
CFG_D_BAR4	"0'b11111111111111111110000000001100" (default)	
CFG_D_BAR5	"0'b11111111111111111111111111111111" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
CFG_EXP_ROM_A	"0'b00000000000000000000000000000000" (default)	
CFG_EXP_ROM_B	"0'b00000000000000000000000000000000" (default)	
CFG_EXP_ROM_C	"0'b00000000000000000000000000000000" (default)	
CFG_EXP_ROM_D	"0'b00000000000000000000000000000000" (default)	
CIS_POINTER_CARDBUS_A	"0'b00000000000000000000000000000000" (default)	
CIS_POINTER_CARDBUS_B	"0'b00000000000000000000000000000000" (default)	
CIS_POINTER_CARDBUS_C	"0'b00000000000000000000000000000000" (default)	
CIS_POINTER_CARDBUS_D	"0'b00000000000000000000000000000000" (default)	
CLASS_CODE_ID3A	"0'b000100011000000000000000" (default)	
CLASS_CODE_ID3B	"0'b000100011000000000000000" (default)	
CLASS_CODE_ID3C	"0'b000100011000000000000000" (default)	
CLASS_CODE_ID3D	"0'b000100011000000000000000" (default)	
CM_RESTORE_TIME	"0'b00000000" (default)	
CNT250NS_MAX	"0'b001111100" (default)	
COARSE_GAIN	"DISABLED" (default)	
COEF_EN_LPBK_MASTER	"OTHERWISE" (default)	
COEF_EN_LPBK_SLAVE	"OTHERWISE" (default)	
COEF_ENABLE	"DETERMINE_LOCAL_PHY" (default)	
COEF_ENABLE_8G	"DISABLED" (default)	
COEF_EQTX_FORCE	"0'b0000000000000000" (default)	
COEF_LPBK_MASTER	"0'b0000000000000000" (default)	
COEF_LPBK_SLAVE	"0'b0000000000000000" (default)	
COEF0_POST	"0'b000101" (default)	
COEF0_POST_CURSOR	"0'b000000" (default)	
COEF0_PRE	"0'b000000" (default)	
COEF0_PRE_CURSOR	"0'b000000" (default)	
COEF1_POST	"0'b000011" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
COEF1_POST_CURSOR	"0'b000000" (default)	
COEF1_PRE	"0'b000000" (default)	
COEF1_PRE_CURSOR	"0'b000000" (default)	
COEF10_POST	"0'b000111" (default)	
COEF10_PRE	"0'b000000" (default)	
COEF2_POST	"0'b000100" (default)	
COEF2_POST_CURSOR	"0'b000000" (default)	
COEF2_PRE	"0'b000000" (default)	
COEF2_PRE_CURSOR	"0'b000000" (default)	
COEF3_POST	"0'b000010" (default)	
COEF3_POST_CURSOR	"0'b000000" (default)	
COEF3_PRE	"0'b000000" (default)	
COEF3_PRE_CURSOR	"0'b000000" (default)	
COEF4_POST	"0'b000000" (default)	
COEF4_PRE	"0'b000000" (default)	
COEF5_POST	"0'b000000" (default)	
COEF6_POST	"0'b000000" (default)	
COEF6_PRE	"0'b000010" (default)	
COEF7_POST	"0'b000100" (default)	
COEF7_PRE	"0'b000011" (default)	
COEF8_POST	"0'b000011" (default)	
COEF8_PRE	"0'b000010" (default)	
COEF9_POST	"0'b000000" (default)	
COEF9_PRE	"0'b000011" (default)	
COMP_128_SUPPORTED	"ENABLED" (default)	
COMP_32_SUPPORTED	"ENABLED" (default)	
COMP_64_SUPPORTED	"ENABLED" (default)	
COMPLETE	"DISABLED" (default)	
COMPLIANCE	"ENABLED" (default)	
CONV_METHOD	"COMPUTE_PCIE_SPEC" (default)	
CORE_BYPASS	"NORMAL" (default)	
CORE_EN	"ENABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
COUNT_ACK_TO_NAK	"0'b00000000" (default)	
CPL_TIMEOUT_DISABLE_SUPPORTED	"SUPPORTED" (default)	
CPL_TIMEOUT_RANGES_SUPPORTED	"NOT_SUPPORTED" (default)	
CRS_ENABLE	"DISABLED" (default)	
CSTAT_DATA_SCALE	"UNKNOWN_SCALE" (default)	
CSTAT_DATA_SELECT	"D0_POWER_CONSUMED" (default)	
CTLE_SETTLE	"0'b100" (default)	
CTLEBIAS_1	"0'b1000" (default)	
CTLEBYPASS	"DISABLED" (default)	
CUR_FOM	"NUMBER_OF_CLOCK" (default)	
CUR_FOM_AVG	"0'b101" (default)	
CUST_AUTO	"DISABLED" (default)	
CUST_CHK	"SET" (default)	
CUST_SEL	"DISABLED" (default)	
CUST_SKIP	"DISABLED" (default)	
CUST_TYP	"0'b000" (default)	
CUSTOM_PATTERN	"0'b00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000" (default)	
D1_SUPPORT	"SUPPORTED" (default)	
D2_SUPPORT	"SUPPORTED" (default)	
DATA_DETECT0	"OTHERWISE" (default)	
DATA_DETECT1	"OTHERWISE" (default)	
DATA_DETECT2	"OTHERWISE" (default)	
DATA_DETECT3	"OTHERWISE" (default)	
DATA_INJECT	"0'b00000000000000000000000000000000" (default)	
DATA_PM	"0'b00000000" (default)	
DB_UPSTREAM	"6DB" (default)	
DEEMPH_5G_3_5DB_6DB_N	"6DB" (default)	
DEEMPH_5G_ENABLE	"DISABLED" (default)	
DEEMPH_LPBK_MASTER	"6P0DB" (default)	
DEEMPH_LPBK_SLAVE	"6P0DB" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
DEVICE_ID_ID1A	"0'b1110000000000100" (default)	
DEVICE_ID_ID1B	"0'b1110000000000100" (default)	
DEVICE_ID_ID1C	"0'b1110000000000100" (default)	
DEVICE_ID_ID1D	"0'b1110000000000100" (default)	
DEVICE_PORT_TYPE	"PCIE_ENDPOINT" (default)	
DFE_BIAS	"0'b0001" (default)	
DFE_PWDN	"DISABLED" (default)	
DIR	"RECEIVE" (default)	
DIR_PRE_GAIN	"0'b00" (default)	
DIR_PST_GAIN	"0'b01" (default)	
DIRECT_SPEED_CHANGE	"OTHERWISE" (default)	
DIRECT_TO_DETECT_FAST	"OTHERWISE" (default)	
DIRECT_TO_DISABLE	"OTHERWISE" (default)	
DIRECT_TO_HOT_RESET	"OTHERWISE" (default)	
DIRECT_TO_LPBK_ENTRY	"OTHERWISE" (default)	
DIRECT_TO_RCVRY_CH_BOND	"OTHERWISE" (default)	
DIRECT_TO_RCVRY_FRAME	"OTHERWISE" (default)	
DIRECT_TO_RCVRY_PHY	"OTHERWISE" (default)	
DIRECT_TO_RCVRY_REPLAY	"OTHERWISE" (default)	
DIS_ARI_CAP	"ENABLED" (default)	
DIS_CSR_RST	"ENABLED" (default)	
DIS_FUNC_B	"ENABLED" (default)	
DIS_FUNC_C	"ENABLED" (default)	
DIS_FUNC_D	"ENABLED" (default)	
DIS_INTERRUPT	"ENABLED" (default)	
DIS_INTERRUPT_B	"ENABLED" (default)	
DIS_INTERRUPT_C	"ENABLED" (default)	
DIS_INTERRUPT_D	"ENABLED" (default)	
DIS_MSI_CAP	"ENABLED" (default)	
DIS_MSI_CAP_B	"ENABLED" (default)	
DIS_MSI_CAP_C	"ENABLED" (default)	
DIS_MSIX_CAP	"ENABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
DIS_MSIX_CAP_B	"ENABLED" (default)	
DIS_MSIX_CAP_C	"ENABLED" (default)	
DIS_MSIX_CAP_D	"ENABLED" (default)	
DIS_PREVENT	"ENABLED" (default)	
DISABLE_FLR_CAPABILITY	"ENABLED" (default)	
DLLP_CRC_ERR_ENABLE	"DISABLED" (default)	
DLLP_CRC_ERR_RATE	"0'b000000000000" (default)	
DLLP_INJECT_ENABLE	"DISABLED" (default)	
DOUBLE_TX_DATA_VALID	"ONE_CLK_EVERY_64_CLKS" (default)	
DOWNSTREAM_EQ_SKIP_PHASE_2_3	"NORMAL_OPERATION" (default)	
DS_DRIVE_CLKREQ	"ENABLED" (default)	
DS_PORT_RX_PRESET_HINT	"0'b001" (default)	
DS_PORT_TX_PRESET	"0'b0011" (default)	
DS_US_N_PORTTYPE	"UPSTREAM" (default)	
DSI	"NO_DSI_NECESSARY" (default)	
DSP_DIR	"ANALYSIS_OF_DATA_BY_DSP" (default)	
DSPDIR_PRESGN	"0'b11110000" (default)	
DSPDIR_PREVAL	"0'b00011000" (default)	
DSPDIR_PSTSGN0	"0'b11111111" (default)	
DSPDIR_PSTSGN1	"0'b00000000" (default)	
DSPDIR_PSTVAL0	"0'b00000010" (default)	
DSPDIR_PSTVAL1	"0'b01000000" (default)	
EARLY_RX_EVAL	"RX_SIGNAL_AFTER_TS1" (default)	
ECRC_GEN_CHK_CAPABLE	"SUPPORTED" (default)	
EFF_LPBK	"PASSED" (default)	
EI4	"EI_IV" (default)	
EIE_DETECT0	"OTHERWISE" (default)	
EIE_DETECT1	"OTHERWISE" (default)	
EIE_DETECT2	"OTHERWISE" (default)	
EIE_DETECT3	"OTHERWISE" (default)	
EIOS_DETECT0	"OTHERWISE" (default)	
EIOS_DETECT1	"OTHERWISE" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
EIOS_DETECT2	"OTHERWISE" (default)	
EIOS_DETECT3	"OTHERWISE" (default)	
EM_INTERLOCK_PRESENT	"NOT_SUPPORTED" (default)	
EN	"DISABLED" (default)	
EN_ACK_TO_DIV	"ACK_SPEC" (default)	
EN_ACK_TO_NAK	"DO_NOTHING" (default)	
EN_ACS_VIOLATION	"DISABLED" (default)	
EN_ASPM_L0S	"ENABLED" (default)	
EN_ASPM_L1	"ENABLED" (default)	
EN_ATOMIC_OP_CAP	"ENABLED" (default)	
EN_ATOMICOP_EGRESS_BLOCKED	"DISABLED" (default)	
EN_ATS_CAP	"ENABLED" (default)	
EN_BDGT_CAP	"DISABLED" (default)	
EN_CAP	"ENABLED" (default)	
EN_CAP_B	"ENABLED" (default)	
EN_CAP_C	"ENABLED" (default)	
EN_CAP_D	"ENABLED" (default)	
EN_COMPLETER_ABORT	"DISABLED" (default)	
EN_COMPLETION_TIMEOUT	"ENABLED" (default)	
EN_CORR_INTERNAL_ERROR	"DISABLED" (default)	
EN_DPA_CAP	"DISABLED" (default)	
EN_DRCT_SCR_OFF	"OTHERWISE" (default)	
EN_DRCT_TO_LPBK	"OTHERWISE" (default)	
EN_EQTX_OVERRIDE	"PIPE_LOCAL_FS_AND_PIPE_LOCAL_LF" (default)	
EN_FORCE_SCR_OFF_FAST	"OTHERWISE" (default)	
EN_L1	"ENABLED" (default)	
EN_L1PMSS_CAP	"ENABLED" (default)	
EN_L2	"ENABLED" (default)	
EN_LPBK_ERR_RST	"MASTER_LPBK_INCREMENT" (default)	
EN_LTR_CAP	"ENABLED" (default)	
EN_MC_BLOCKED_TLP	"DISABLED" (default)	
EN_NWL_VSEC_CAP	"ENABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
EN_PIPE_IF_CTRL	"DISABLED" (default)	
EN_PORT_DIS	"DISABLED" (default)	
EN_PORT_INTLEG	"ENABLED" (default)	
EN_RBAR_CAP_A	"ENABLED" (default)	
EN_RBAR_CAP_C	"ENABLED" (default)	
EN_RBAR_CAP_D	"ENABLED" (default)	
EN_RECEIVER_OVERFLOW	"DISABLED" (default)	
EN_RX_ALLOC_SEL	"HW" (default)	
EN_SELF_XLINK	"OTHERWISE" (default)	
EN_SURPRISE_DOWN_ERROR	"DISABLED" (default)	
EN_TLP_PREFIX_BLOCKED	"DISABLED" (default)	
EN_TX_ALLOC_SEL	"HW" (default)	
EN_UCORR_INTERNAL_ERROR	"DISABLED" (default)	
EN_USER_WRITE	"READ_WRITE_ACCESS" (default)	
ENABLE_USER_CFG	"DISABLED" (default)	
END_END_PREFIXES_SUPPORTED	"NOT_SUPPORTED" (default)	
END_ON_HOLD	"YES_EXIT_ON_HOLD" (default)	
ENDCALIB_MAX	"0’b10000100" (default)	
ENDPOINT_L0S_ACCEPTABLE_LATENCY	"MAX_64_NS" (default)	
ENDPOINT_L1_ACCEPTABLE_LATENCY	"MAX_1_US" (default)	
ENTRY_TIME_ASPM_L0S	"0’b0000000000000000" (default)	
ENTRY_TIME_ASPM_L1	"0’b0000000000000000" (default)	
EOM_TIME	"0’b0000000000000000" (default)	
EOM0DIR	"SELECT_DIR_1" (default)	
EOM1DIR	"SELECT_DIR_0" (default)	
EOMCTRL0_LOW	"DISABLED" (default)	
EOMDIVDIS	"DISABLED" (default)	
EOMMODE	"0’b00" (default)	
EOMRDSEL	"DISABLED" (default)	
EOMSTART	"DISABLED" (default)	
EOMX	"0’b000000" (default)	
EOMX_UPDATE_CNT_VALUE	"0’b0011111" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
EOMY	"0'b00000000" (default)	
ERR_AER_BAD_DLLP	"OTHERWISE" (default)	
ERR_AER_BAD_TLP	"OTHERWISE" (default)	
ERR_AER_DL_PROTOCOL_ERROR	"OTHERWISE" (default)	
ERR_AER_RECEIVER_ERROR	"OTHERWISE" (default)	
ERR_AER_REPLAY_NUM_ROLLOVER	"OTHERWISE" (default)	
ERR_AER_REPLAY_TIMER_TIMEOUT	"OTHERWISE" (default)	
ERR_AER_SURPRISE_DOWN	"OTHERWISE" (default)	
ERR_AER_TX_PAR2	"OTHERWISE" (default)	
ERR_AER_TX_REPLAY_ECC1	"OTHERWISE" (default)	
ERR_AER_TX_REPLAY_ECC2	"OTHERWISE" (default)	
ERR_REC_ENTRY_SEL	"RCVRY_AFTER" (default)	
ERR_TX_PIPE_UNDERFLOW	"OTHERWISE" (default)	
ERRCNT_DEC	"0'b00100000" (default)	
ERRCNT_THR	"0'b1000" (default)	
ES_PWDN	"DISABLED" (default)	
EVAL_RST	"DISABLED" (default)	
EXCLUDE_CFG_COMPLETE	"INCLUDE" (default)	
EXCLUDE_CFG_IDLE	"INCLUDE" (default)	
EXCLUDE_L0	"INCLUDE" (default)	
EXCLUDE_LOOPBACK_MASTER	"INCLUDE" (default)	
EXCLUDE_REC_IDLE	"INCLUDE" (default)	
EXCLUDE_REC_RCVR_CFG	"INCLUDE" (default)	
EXIT_DIRECT_TO_DETECT	"DO_NOT_EXIT_TO_DETECT" (default)	
EXT_CONTROL	"DISABLED" (default)	
EXTENDED_TAG_FIELD_EN_DEFAULT	"EIGHT_BIT" (default)	
EXTENDED_TAG_FIELD_SUPPORTED	"EIGHT_BIT" (default)	
F_ARXCTLEDIR	"IGNORED" (default)	
F_ARXCTLENULL	"0'b0000" (default)	
F_ARXDMDIR	"DISABLED" (default)	
F_ARXDMNULL	"0'b00000" (default)	
F_ARXDPDIR	"IGNORED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
F_ARXDPNULL	"0’b00000" (default)	
F_ARXEOMDIR	"IGNORED" (default)	
F_ARXEOMNULL	"0’b00000" (default)	
F_ARXESDIR	"IGNORED" (default)	
F_ARXESNULL	"0’b00000" (default)	
F_ARXTDIR	"IGNORED" (default)	
F_ARXTNULL	"0’b00000" (default)	
F_ASCHCAL	"IGNORED" (default)	
F_ASCHDIR	"IGNORED" (default)	
F_ASCHNULL	"0’b0000" (default)	
FAIL_LIMIT_ERR	"RXEQ_NOT_FAIL" (default)	
FAST	"L0" (default)	
FC_UPDATE_TIMER_DISABLE	"ENABLED" (default)	
FC_UPDATE_TIMER_DIV	"PCIE_REC_VALUES" (default)	
FILTER	"0’b1001" (default)	
FINE_GAIN	"DISABLED" (default)	
FOM_COMPARE	"0’b00000000" (default)	
FOM_HIRES	"DISABLED" (default)	
FOM_ITERCNT	"0’b101" (default)	
FOM_THR	"0’b0100" (default)	
FORCE_ATXDRA	"0’b00000000000000000000" (default)	
FORCE_ATXDRP	"0’b00000000000000000000" (default)	
FORCE_ATXDRR	"0’b00000000000000000000" (default)	
FORCE_ATXDRT	"0’b00000000000000000000" (default)	
FORCE_DIR_RSLT	"0’b000000" (default)	
FORCE_FOM_RSLT	"0’b00000000" (default)	
FORCE_IDLE	"DISABLED" (default)	
FORCE_RX_DETECT	"DISABLED" (default)	
FORCE_SIGNAL	"DISABLED" (default)	
FREQ_LOCK	"DISABLED" (default)	
FS	"0’b110000" (default)	
FTS_DETECT0	"OTHERWISE" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
FTS_DETECT1	"OTHERWISE" (default)	
FTS_DETECT2	"OTHERWISE" (default)	
FTS_DETECT3	"OTHERWISE" (default)	
GEN12_ENA_POST_A0	"DISABLED" (default)	
GEN12_ENA_POST_A1A2	"DISABLED" (default)	
GEN12_ENA_PREA0	"DISABLED" (default)	
GEN3_ENA_POST_A0	"ENABLED" (default)	
GEN3_ENA_POST_A1A2	"ENABLED" (default)	
GEN3_ENA_PREA0	"ENABLED" (default)	
GLOBAL_INVAL_SUPPORT	"ENABLED" (default)	
GSR	"ENABLED" (default)	
HINT	"0’b000" (default)	
HINT0_3DB	"ENABLED" (default)	
HINT0_A0GAIN	"0’b111" (default)	
HINT0_A2GAIN	"0’b111" (default)	
HINT1_3DB	"ENABLED" (default)	
HINT1_A0GAIN	"0’b011" (default)	
HINT1_A2GAIN	"0’b101" (default)	
HINT2_3DB	"ENABLED" (default)	
HINT2_A0GAIN	"0’b011" (default)	
HINT2_A2GAIN	"0’b111" (default)	
HINT3_3DB	"ENABLED" (default)	
HINT3_A0GAIN	"0’b000" (default)	
HINT3_A2GAIN	"0’b111" (default)	
HINT4_3DB	"DISABLED" (default)	
HINT4_A0GAIN	"0’b111" (default)	
HINT4_A2GAIN	"0’b111" (default)	
HINT5_3DB	"DISABLED" (default)	
HINT5_A0GAIN	"0’b011" (default)	
HINT5_A2GAIN	"0’b101" (default)	
HINT6_3DB	"DISABLED" (default)	
HINT6_A0GAIN	"0’b011" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
HINT6_A2GAIN	"0’b111" (default)	
HINT7_3DB	"DISABLED" (default)	
HINT7_A0GAIN	"0’b000" (default)	
HINT7_A2GAIN	"0’b111" (default)	
HINT7_OVR	"DISABLED" (default)	
HLD_RST	"WRITE_1" (default)	
HOT_PLUG_CAPABLE	"NOT_SUPPORTED" (default)	
HOT_PLUG_SURPRISE	"NOT_POSSIBLE" (default)	
HOT_RESET	"ENABLED" (default)	
ID_DS_PORT	"0’b0000000000000000" (default)	
ID_NWL_VSEC_CAP	"0’b0000000000000001" (default)	
IDDQ_PCS	"DISABLED" (default)	
IDLE_INFER_L0_TO_REC_RCVR_LOCK	"OTHERWISE" (default)	
IDLE_INFER_LPBK_SLAVE	"OTHERWISE" (default)	
IDLE_INFER_REC_RCVR_CFG	"OTHERWISE" (default)	
IDLE_INFER_REC_SPEED2_SUCCESS	"OTHERWISE" (default)	
IDLE_INFER_REC_SPEED2_UNSUCCESS	"OTHERWISE" (default)	
IGNORE_ECRC	"DISABLED" (default)	
IGNORE_POISON	"ENABLED" (default)	
INDICATOR	"0’b00000000000000000000000000000000" (default)	
INFO_BAD_TLP_CRC_ERR	"OTHERWISE" (default)	
INFO_BAD_TLP_ECRC_ERR	"OTHERWISE" (default)	
INFO_BAD_TLP_MALF_ERR	"OTHERWISE" (default)	
INFO_BAD_TLP_NULL_ERR	"OTHERWISE" (default)	
INFO_BAD_TLP_PHY_ERR	"OTHERWISE" (default)	
INFO_BAD_TLP_SEQ_ERR	"OTHERWISE" (default)	
INFO_DESKEW_OVERFLOW_ERROR	"OTHERWISE" (default)	
INFO_NAK_RECEIVED	"OTHERWISE" (default)	
INFO_REPLAY_STARTED	"OTHERWISE" (default)	
INFO_SCHEDULE_DUPL_ACK	"OTHERWISE" (default)	
INFO_TX_DATA_UNDERFLOW	"OTHERWISE" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
INHIBIT	"PERFORM_RECEIVER_DETECTION" (default)	
INJECT_DATA_ERROR_0	"DISABLED" (default)	
INJECT_DATA_ERROR_1	"DISABLED" (default)	
INJECT_DATA_ERROR_2	"DISABLED" (default)	
INJECT_DATA_ERROR_3	"DISABLED" (default)	
INJECT_DATA_ERROR_EN	"DISABLED" (default)	
INJECT_ERR_LANE_SELECT_0	"DISABLED" (default)	
INJECT_ERR_LANE_SELECT_1	"DISABLED" (default)	
INJECT_ERR_LANE_SELECT_2	"DISABLED" (default)	
INJECT_ERR_LANE_SELECT_3	"DISABLED" (default)	
INJECT_RX_1BIT_DATA_ERR	"DISABLED" (default)	
INJECT_RX_2BIT_DATA_ERR	"DISABLED" (default)	
INJECT_RX_SKP_ERR	"DISABLED" (default)	
INJECT_RX_VALID_ERR	"DISABLED" (default)	
INT_CLR	"DISABLED" (default)	
INT_EN	"DISABLED" (default)	
INTERRUPT_MESSAGE_NUMBER	"0’b00000" (default)	
INVAL_Q_DEPTH	"0’b00000" (default)	
ITERATION_MAX	"0’b000000" (default)	
L0_TO_REC_RCVR_LOCK_RX_8G_EIE	"OTHERWISE" (default)	
L0_TO_REC_RCVR_LOCK_RX_INFER	"OTHERWISE" (default)	
L0_TO_REC_RCVR_LOCK_RX_TS12	"OTHERWISE" (default)	
L0S_ADJ	"0’b00000110000000" (default)	
L0S_EXIT_LATENCY	"MORE_4_US" (default)	
L1_ENTER_PLL_RESET_TIME	"0’b100" (default)	
L1_EXIT_LATENCY	"MORE_64_US" (default)	
L1_EXIT_PLL_LOCK_TIME	"0’b01110" (default)	
L1PM_SUPPORTED	"SUPPORTED" (default)	
L2_D3HOT_ENABLE	"DISABLED" (default)	
LANE	"0" (default)	
LANE_SELECT	"0’b0000" (default)	
LEGACY_MODE	"MATCH_TS" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
LF	"0'b001000" (default)	
LF_PHY	"0'b001010" (default)	
LINK_LANE	"ENABLED" (default)	
LOOPBACK	"ENABLED" (default)	
LPBK_EN	"DISABLED" (default)	
LW_START_UPDN_ACK_EN	"DISABLED" (default)	
LW_START_UPDN_COUNT	"0'b000011111010" (default)	
LW_START_UPDN_EIE_EN	"DISABLED" (default)	
LW_START_UPDN_EN_DIR_DS	"DO_NOT_ASSERT" (default)	
LW_START_UPDN_END_DELAY	"0'b1001" (default)	
LW_START_UPDN_RATE_EN_16G	"DISABLED" (default)	
LW_START_UPDN_RATE_EN_2P5G	"ENABLED" (default)	
LW_START_UPDN_RATE_EN_5G	"ENABLED" (default)	
LW_START_UPDN_RATE_EN_8G	"ENABLED" (default)	
LW_START_UPDN_START_DELAY	"0'b1000" (default)	
LW_START_UPDN_TIMER_EN	"DISABLED" (default)	
MARGIN_ENABLE	"PCIE_SPEC" (default)	
MARGIN_VALUE	"0'b000" (default)	
MASK_0	"SKIP_RCVR_DETECTION" (default)	
MASK_1	"SKIP_RCVR_DETECTION" (default)	
MASK_2	"SKIP_RCVR_DETECTION" (default)	
MASK_3	"SKIP_RCVR_DETECTION" (default)	
MAX_LINK_WIDTH	"1_LANE" (default)	
MAX_PAYLOAD_SIZE_SUPPORTED	"256_BYTES" (default)	
MAX_RSA_WAIT	"0'b00101000" (default)	
MAX_SPEED	"8G" (default)	
MAX_VAR	"0'b00100" (default)	
MERGE_LMMI_RDATA	"DISABLED" (default)	
MESO_LPBK	"DISABLED" (default)	
METHOD_FMERIT_CTRL	"STEP_PCIE_TX_PRESETS" (default)	
METHOD_TX_CRED_CLEANUP	"HEADER" (default)	
MGMT_INTLEG	"0'b0000" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
MGMT_LTSSM_DIS	"DISABLED" (default)	
MID_VALUE_10B	"0’b011110" (default)	
MID_VALUE_20B	"0’b010100" (default)	
MID_VALUE_GEN3	"0’b001110" (default)	
MIN_TIME	"0_MS" (default)	
MIN_TIME_CFG	"4US" (default)	
MIX_DIR	"DISABLED" (default)	
MODE_BFF	"RESTART" (default)	
MPS_VIOLATION_RX	"DISABLED" (default)	
MPS_VIOLATION_TX	"DISABLED" (default)	
MRL_SENSOR_PRESENT	"NOT_SUPPORTED" (default)	
MULT_ENABLE	"RECOMMENDED_VALUES" (default)	
MULT_MESSAGE_CAPABLE_MSICAP_A	"EIGHT" (default)	
MULT_MESSAGE_CAPABLE_MSICAP_B	"EIGHT" (default)	
MULT_MESSAGE_CAPABLE_MSICAP_C	"EIGHT" (default)	
MULT_MESSAGE_CAPABLE_MSICAP_D	"EIGHT" (default)	
N_RX_LIM_D	"DISABLED" (default)	
N_RX_LIM_H	"DISABLED" (default)	
N_TX_LIM_D	"DISABLED" (default)	
N_TX_LIM_H	"DISABLED" (default)	
NFTS	"0’b11111111" (default)	
NO_COMMAND_COMPLETED_SUPPORT	"SW_NOTIF_PROVIDED" (default)	
NO_FCMP	"DISABLED" (default)	
NO_REMOTE_CHANGE	"DISABLED" (default)	
NO_TX_IDLE_DELAY	"DATA_VALID_GAP" (default)	
NUM_LANES	"1_LANE" (default)	
NUMBER_DSLINK	"0’b00000" (default)	
NUMHI_A	"0’b00000000000000000000000000000000" (default)	
NUMHI_B	"0’b00000000000000000000000000000000" (default)	
NUMHI_C	"0’b00000000000000000000000000000000" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
NUMHI_D	"0'b00000000000000000000000000000000" (default)	
NUMHOLD	"SINGLE_HOLD_RESPONSE" (default)	
NUMLO_A	"0'b00000000000000000000000000000000" (default)	
NUMLO_B	"0'b00000000000000000000000000000000" (default)	
NUMLO_C	"0'b00000000000000000000000000000000" (default)	
NUMLO_D	"0'b00000000000000000000000000000000" (default)	
OBFF_SUPPORTED	"NOT_SUPPORTED" (default)	
OFFSET_MSIX_PBA_A	"0'b00000000000000000111000000000" (default)	
OFFSET_MSIX_PBA_B	"0'b00000000000000000111000000000" (default)	
OFFSET_MSIX_PBA_C	"0'b00000000000000000111000000000" (default)	
OFFSET_MSIX_PBA_D	"0'b00000000000000000111000000000" (default)	
OFFSET_MSIX_TABLE_A	"0'b00000000000000000110000000000" (default)	
OFFSET_MSIX_TABLE_B	"0'b00000000000000000110000000000" (default)	
OFFSET_MSIX_TABLE_C	"0'b00000000000000000110000000000" (default)	
OFFSET_MSIX_TABLE_D	"0'b00000000000000000110000000000" (default)	
OVER_CTLE	"DISABLED" (default)	
OVER_RX	"DISABLED" (default)	
OVER_RXDM	"DISABLED" (default)	
OVER_RXDP	"DISABLED" (default)	
OVER_RXES	"DISABLED" (default)	
OVER_RXT	"DISABLED" (default)	
OVER_SCH	"DISABLED" (default)	
OVER_TX	"DISABLED" (default)	
OVERFLOW	"DISABLED" (default)	
OVR_CDR	"DISABLED" (default)	
OVR_DIR	"DISABLED" (default)	
OVR_FOM	"DISABLED" (default)	
OVR_GAIN3DB	"ENABLED" (default)	
OVR_HINT3DB	"ENABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
P_CLK_PERIOD	"0'b0000111110100000" (default)	
P_RX_LIM_D	"DISABLED" (default)	
P_RX_LIM_H	"DISABLED" (default)	
P_TX_LIM_D	"DISABLED" (default)	
P_TX_LIM_H	"DISABLED" (default)	
PAR_LPBK	"DISABLED" (default)	
PAS	"10X" (default)	
PATTERN_0	"UNSCRAMBLED" (default)	
PATTERN_1	"UNSCRAMBLED" (default)	
PATTERN_2	"UNSCRAMBLED" (default)	
PCIPM_L1_1_SUPPORTED	"SUPPORTED" (default)	
PCIPM_L1_2_SUPPORTED	"SUPPORTED" (default)	
PCLKREQ_N	"DEASSERTED" (default)	
PERIOD_SRIS_128B130B	"0'b000000" (default)	
PERIOD_SRIS_8B10B	"0'b00000000" (default)	
PERIOD_SRNS_128B130B	"0'b00000000" (default)	
PERIOD_SRNS_8B10B	"0'b00000000" (default)	
PHANTOM_FUNCTIONS_SUPPORTED	"NO_FUNCTION_BITS" (default)	
PHY_MODE	"0'b0000" (default)	
PHYSICAL_SLOT_NUMBER	"0'b0000000000001" (default)	
PIN_INTERRUPT_A	"INTA" (default)	
PIN_INTERRUPT_B	"INTA" (default)	
PIN_INTERRUPT_C	"INTA" (default)	
PIN_INTERRUPT_D	"INTA" (default)	
PIPE_PWRDN	"P1" (default)	
PIPE_TX_SWING	"FULL_SWING" (default)	
PLESIO_LPBK	"DISABLED" (default)	
PM_REDUCE_TIMEOUTS	"DISABLED" (default)	
PMA_DRIVEN_MODE	"PCS_DRIVEN" (default)	
PMCSR_B2_B3_SUPPORT	"DISABLED" (default)	
PMCSR_BUS_P_C_EN	"DISABLED" (default)	
PME_CLOCK	"DISABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
PME_SUPPORT	"0'b11111" (default)	
PMFF_ALL	"DISABLED" (default)	
PORT_CM_RESTORE_TIME	"0'b00000000" (default)	
PORT_NUMBER	"0'b00000000" (default)	
PORT_TPOWER_ON_SCALE	"2_US" (default)	
PORT_TPOWER_ON_VALUE	"0'b00000" (default)	
POST	"0'b000000" (default)	
POST_A0COEF	"0'b001" (default)	
POST_A1COEF	"0'b001" (default)	
POST_A2COEF	"0'b001" (default)	
POST_CURSOR_LIMIT	"0'b100000" (default)	
POST_CURSOR_STEP_SIZE	"0'b001000" (default)	
POST_ITERCNT	"0'b100" (default)	
POST_STEP	"STEP_SIZE_4" (default)	
POWER_CONTROLLER_PRESENT	"NOT_SUPPORTED" (default)	
POWER_INDICATOR_PRESENT	"NOT_SUPPORTED" (default)	
POWER_REQUIRED	"AUX_POWER_NOT_REQUIRED" (default)	
PRBS_CHK	"DISABLED" (default)	
PRBS_GEN	"DISABLED" (default)	
PRBS_TYP	"PRBS7" (default)	
PRE	"0'b000000" (default)	
PRE_A0COEF	"0'b101" (default)	
PRE_A1COEF	"0'b101" (default)	
PRE_A2COEF	"0'b101" (default)	
PRE_CURSOR_LIMIT	"0'b010000" (default)	
PRE_CURSOR_STEP_SIZE	"0'b000100" (default)	
PRE_FOM	"ENABLED" (default)	
PRE_FOM_AVG	"0'b100" (default)	
PRE_ITERCNT	"0'b100" (default)	
PRE_RXEQ_TIMER	"0'b00010100" (default)	
PRE_STEP	"STEP_SIZE_2" (default)	
PRESET_COUNT_INI	"0'b0000" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
PRESET_EN_LPBK_MASTER	"OTHERWISE" (default)	
PRESET_EN_LPBK_SLAVE	"OTHERWISE" (default)	
PRESET_ENABLE	"NORMAL_OP" (default)	
PRESET_ENABLE_8G	"DISABLED" (default)	
PRESET_EQTX_FORCE	"0'b0000" (default)	
PRESET_LPBK_MASTER	"0'b0000" (default)	
PRESET_LPBK_SLAVE_0	"OTHERWISE" (default)	
PRESET_LPBK_SLAVE_1	"OTHERWISE" (default)	
PRESET_LPBK_SLAVE_3	"OTHERWISE" (default)	
PRESET_REJECT	"0'b00000000000" (default)	
PRESET0_POSTCURSOR	"0'b1000" (default)	
PRESET0_PRECURSOR	"0'b0000" (default)	
PRESET1_POSTCURSOR	"0'b0101" (default)	
PRESET1_PRECURSOR	"0'b0000" (default)	
PRESET10_POSTCURSOR	"0'b1010" (default)	
PRESET10_PRECURSOR	"0'b0000" (default)	
PRESET2_POSTCURSOR	"0'b0110" (default)	
PRESET2_PRECURSOR	"0'b0000" (default)	
PRESET3_POSTCURSOR	"0'b0100" (default)	
PRESET3_PRECURSOR	"0'b0000" (default)	
PRESET4_POSTCURSOR	"0'b0000" (default)	
PRESET4_PRECURSOR	"0'b0000" (default)	
PRESET5_POSTCURSOR	"0'b0000" (default)	
PRESET5_PRECURSOR	"0'b0011" (default)	
PRESET6_POSTCURSOR	"0'b0000" (default)	
PRESET6_PRECURSOR	"0'b0100" (default)	
PRESET7_POSTCURSOR	"0'b0110" (default)	
PRESET7_PRECURSOR	"0'b0011" (default)	
PRESET8_POSTCURSOR	"0'b0100" (default)	
PRESET8_PRECURSOR	"0'b0100" (default)	
PRESET9_POSTCURSOR	"0'b0000" (default)	
PRESET9_PRECURSOR	"0'b0101" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
PS_REENTRY_TIME	"0'b00000000" (default)	
PWDN_N	"DISABLED" (default)	
RATE	"2P5G" (default)	
RATE_ENABLE	"INITIAL_SPEED_CHANGES" (default)	
RCB	"DISABLED" (default)	
REC_SPD_INFER_EQ_PH0123	"EXCLUDE_TIME_SPENT" (default)	
REC_SPD_INFER_RCVR_CFG	"EXCLUDE_TIME_SPENT" (default)	
REC_SPD_INFER_RCVR_LOCK	"EXCLUDE_TIME_SPENT" (default)	
REDO	"DISABLED" (default)	
REDUCE_TIMEOUTS_LTSSMSIM	"DISABLED" (default)	
REDUCE_TIMEOUTS_SIM	"DISABLED" (default)	
REDUCE_TS1	"DISABLED" (default)	
REENTRY_DISABLE	"ENABLED" (default)	
REENTRY_TIME	"0'b00000000000000" (default)	
REQ_EQ_MAX_COUNT	"0'b10" (default)	
REQ_FEEDBACK	"0'b00000000" (default)	
RESET_EIEOS_INTERVAL_COUNT	"DISABLED" (default)	
REVISION_ID_ID3A	"0'b00000100" (default)	
REVISION_ID_ID3B	"0'b00000100" (default)	
REVISION_ID_ID3C	"0'b00000100" (default)	
REVISION_ID_ID3D	"0'b00000100" (default)	
RL1	"0'b0011" (default)	
RL2	"0'b0101" (default)	
RL3	"0'b0011" (default)	
ROUTING_SUPPORTED	"DISABLED" (default)	
RP_COMPLETER_EN	"DISABLED" (default)	
RSTCDR_ERR	"ENABLED" (default)	
RSTCDR_FRQ	"ENABLED" (default)	
RSTCDR_IDL	"DISABLED" (default)	
RX_BYPASS_DECODE_EN	"ENABLED" (default)	
RX_BYPASS_MSG_DEC	"NORMAL_OPERATION" (default)	
RX_CONVERT_UR_TO_CA	"NORMAL_OPERATION" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
RX_D_ALLOC_C	"0'b0000000001100000" (default)	
RX_D_ALLOC_N	"0'b0000000000000110" (default)	
RX_D_ALLOC_P	"0'b0000000001101100" (default)	
RX_DIV_MODE0	"DIV_2" (default)	
RX_DIV_MODE1	"0'b10" (default)	
RX_DIV_MODE2	"0'b00" (default)	
RX_DL_ACTIVE_DISABLE	"BLOCK_RECEPTION_TLP" (default)	
RX_EARLY_FORWARD_DISABLE	"FWD_RX_DATA_LL" (default)	
RX_ECC1_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
RX_ECC1_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
RX_ECC1_INJECT_M_1_N	"INJECT_1_ERR" (default)	
RX_ECC1_INJECT_TYPE	"POSTED_DATA_RAM" (default)	
RX_ECC1_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
RX_ECC2_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
RX_ECC2_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
RX_ECC2_INJECT_M_1_N	"INJECT_1_ERR" (default)	
RX_ECC2_INJECT_TYPE	"POSTED_DATA_RAM" (default)	
RX_ECC2_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
RX_EI_DIS	"DEASSERTED" (default)	
RX_ERR_COR	"DISABLED" (default)	
RX_ERR_ECC1	"OTHERWISE" (default)	
RX_ERR_ECC2	"OTHERWISE" (default)	
RX_ERR_PAR	"OTHERWISE" (default)	
RX_ERR_UCOR	"DISABLED" (default)	
RX_ESP_RESP_WAIT	"0'b01000000" (default)	
RX_FORCE_RO	"DISABLED" (default)	
RX_H_ALLOC_C	"0'b000000100000" (default)	
RX_H_ALLOC_N	"0'b000000001000" (default)	
RX_H_ALLOC_P	"0'b000000010000" (default)	
RX_HIZ	"IGNORED" (default)	
RX_IMPED_RATIO	"0'b10000000" (default)	
RX_INHIBIT_ACK_NAK	"PROCESS_RCVD_ACK" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
RX_INHIBIT_TLP	"PROCESS_RCVD_TLP" (default)	
RX_L0S_DIRECT_TO_RCVRY	"OTHERWISE" (default)	
RX_LCRC_INJECT_EN	"DO_NOT_INJECT_ERROR" (default)	
RX_MALF_INJECT_EN	"DO_NOT_INJECT_ERROR" (default)	
RX_PAR_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
RX_PAR_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
RX_POLINV	"NORMAL" (default)	
RX_PRIORITY	"DISABLED" (default)	
RX_PRIORITY_N_STARVE_THRESH	"0’b00010000" (default)	
RX_PRIORITY_P_STARVE_THRESH	"0’b00010000" (default)	
RX_PWRDN	"IGNORED" (default)	
RX_TLP_VALID	"DISABLED" (default)	
RXEQ_ALGO	"0’b111" (default)	
RXEQ_ENABLE	"0’b100" (default)	
RXEQ_EVAL_MAX	"0’b11111111" (default)	
RXEQ_MANUAL	"DISABLED" (default)	
RXEQ_STATE	"0’b0000" (default)	
RXF_A	"0’b0100" (default)	
RXF_B	"0’b0100" (default)	
RXF_C	"0’b0100" (default)	
RXHF_CLKDN	"ENABLED" (default)	
RXIDLE_MAX	"0’b1111" (default)	
RXIDLE_MAX2	"0’b0000010000000000" (default)	
RXIDLE_MSB	"0’b11" (default)	
RXM_A	"0’b10" (default)	
RXM_B	"0’b01" (default)	
RXM_C	"0’b00" (default)	
RXN_A	"0’b00100" (default)	
RXN_B	"0’b01001" (default)	
RXN_C	"0’b01111" (default)	
RXOFF_SETTLE_MAX	"0’b011" (default)	
RXOFF_STABLE_MAX	"0’b10000" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
RXPLLINIT	"DISABLED" (default)	
RXPLLRST	"SET" (default)	
SDS_DETECT0	"OTHERWISE" (default)	
SDS_DETECT1	"OTHERWISE" (default)	
SDS_DETECT2	"OTHERWISE" (default)	
SDS_DETECT3	"OTHERWISE" (default)	
SEL_PCLK_DIV2	"PCLK_DIV2" (default)	
SELECT_DIR_FOM_N	"FME_METHOD" (default)	
SELECTABLE_DEEMPHASIS	"6P0DB" (default)	
SEQ_NUM	"0'b000000000000" (default)	
SIGNAL_DETECT_THRESHOLD	"125_MV" (default)	
SIZE_CFG0_A	"0'b00000" (default)	
SIZE_CFG0_B	"0'b00000" (default)	
SIZE_CFG0_C	"0'b00000" (default)	
SIZE_CFG0_D	"0'b00000" (default)	
SIZE_CFG1_A	"0'b00000" (default)	
SIZE_CFG1_B	"0'b00000" (default)	
SIZE_CFG1_C	"0'b00000" (default)	
SIZE_CFG1_D	"0'b00000" (default)	
SIZE_CFG2_A	"0'b00000" (default)	
SIZE_CFG2_B	"0'b00000" (default)	
SIZE_CFG2_C	"0'b00000" (default)	
SIZE_CFG2_D	"0'b00000" (default)	
SIZE_CFG3_A	"0'b00000" (default)	
SIZE_CFG3_B	"0'b00000" (default)	
SIZE_CFG3_C	"0'b00000" (default)	
SIZE_CFG3_D	"0'b00000" (default)	
SIZE_CFG4_A	"0'b00000" (default)	
SIZE_CFG4_B	"0'b00000" (default)	
SIZE_CFG4_C	"0'b00000" (default)	
SIZE_CFG4_D	"0'b00000" (default)	
SIZE_CFG5_A	"0'b00000" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
SIZE_CFG5_B	"0'b00000" (default)	
SIZE_CFG5_C	"0'b00000" (default)	
SIZE_CFG5_D	"0'b00000" (default)	
SKIP_FINAL_COEF_CHECK	"DISABLED" (default)	
SKP_DETECT0	"OTHERWISE" (default)	
SKP_DETECT1	"OTHERWISE" (default)	
SKP_DETECT2	"OTHERWISE" (default)	
SKP_DETECT3	"OTHERWISE" (default)	
SLOT_CLOCK_CONFIGURATION	"REFCLK_BY_SLOT" (default)	
SLOT_IMPLEMENTED	"UNCONNECTED" (default)	
SLOT_POWER_LIMIT_SCALE	"0'b00" (default)	
SLOT_POWER_LIMIT_VALUE	"0'b00001010" (default)	
SPEED_CHANGE_FAIL	"OTHERWISE" (default)	
SPEED_LPBK_CTRL	"2P5G" (default)	
START_PRESET	"PRESET_VALUE" (default)	
START_REMOTE_ADV	"OTHERWISE" (default)	
STATE_DATA_N	"USE_RX_DATA_OBSERVATION" (default)	
STEP_SELECT	"0'b00000" (default)	
STP_OVERRIDE_EN	"DISABLED" (default)	
STP_OVERRIDE_LEN	"0'b00000000000" (default)	
STP_OVERRIDE_NEW_LEN	"0'b00000000000" (default)	
STS_PHY_STATUS	"DEASSERTED" (default)	
STS_PIPE_RSTN	"DEASSERTED" (default)	
SUBSTATE_MAX	"0'b00000" (default)	
SUBSYSTEM_ID_ID2A	"0'b1110000000000100" (default)	
SUBSYSTEM_ID_ID2B	"0'b1110000000000100" (default)	
SUBSYSTEM_ID_ID2C	"0'b1110000000000100" (default)	
SUBSYSTEM_ID_ID2D	"0'b1110000000000100" (default)	
SUBSYSTEM_VENDOR_ID_ID2A	"0'b0001100110101010" (default)	
SUBSYSTEM_VENDOR_ID_ID2B	"0'b0001100110101010" (default)	
SUBSYSTEM_VENDOR_ID_ID2C	"0'b0001100110101010" (default)	
SUBSYSTEM_VENDOR_ID_ID2D	"0'b0001100110101010" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
SUPP_SIZE_CFG0_A	"0'b0000000000000001111" (default)	
SUPP_SIZE_CFG0_B	"0'b0000000000000001111" (default)	
SUPP_SIZE_CFG0_C	"0'b0000000000000001111" (default)	
SUPP_SIZE_CFG0_D	"0'b0000000000000001111" (default)	
SUPP_SIZE_CFG1_A	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG1_B	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG1_C	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG1_D	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG2_A	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG2_B	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG2_C	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG2_D	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG3_A	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG3_B	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG3_C	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG3_D	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG4_A	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG4_B	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG4_C	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG4_D	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG5_A	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG5_B	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG5_C	"0'b00000000000000000000" (default)	
SUPP_SIZE_CFG5_D	"0'b00000000000000000000" (default)	
SYS_ALLOC	"PWR_BUDGET_CAP_VALUES" (default)	
T0_RX_BYPASS_MSG_DEC	"NORMAL_OPERATION" (default)	
TABLE_SIZE_MSIXCAP_A	"0'b00000000111" (default)	
TABLE_SIZE_MSIXCAP_B	"0'b00000000111" (default)	
TABLE_SIZE_MSIXCAP_C	"0'b00000000111" (default)	
TABLE_SIZE_MSIXCAP_D	"0'b00000000111" (default)	
TARGET_LINK_SPEED	"8G" (default)	
TARGET_ONLY	"DISABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
TD1_MEANS_ADD_HAS_N	"ECRC_CONTAINED" (default)	
TIMEOUT_THRESHOLD_PME	"0'b000000000000" (default)	
TIMEOUT_THRESHOLD_PME_TO_ACK_DS	"0'b00000000" (default)	
TIMER	"0'b00000000" (default)	
TLP_LCRC_ERR_ENABLE	"DISABLED" (default)	
TLP_LCRC_ERR_RATE	"0'b000" (default)	
TLP_SEQ_ERR_ENABLE	"DISABLED" (default)	
TLUNIT	"1_MS" (default)	
TO_EXTEND	"0'b01111111" (default)	
TRNG_A0COEF	"0'b101" (default)	
TRNG_A1COEF	"0'b101" (default)	
TRNG_A2COEF	"0'b101" (default)	
TRNG_FAST	"DISABLED" (default)	
TRNG_ITERCNT	"0'b100" (default)	
TRNG_RXEQ_TIMER	"0'b00100000" (default)	
TS1_ACK_BLOCK_USE_PRESET	"FORCED_TO_ZERO" (default)	
TS1_ACK_DELAY	"0'b00011111" (default)	
TS1_ACK_MASK_USE_PRESET	"IGNORES_USE_PRESET" (default)	
TS1_DETECT0	"OTHERWISE" (default)	
TS1_DETECT1	"OTHERWISE" (default)	
TS1_DETECT2	"OTHERWISE" (default)	
TS1_DETECT3	"OTHERWISE" (default)	
TS1I_DETECT0	"OTHERWISE" (default)	
TS1I_DETECT1	"OTHERWISE" (default)	
TS1I_DETECT2	"OTHERWISE" (default)	
TS1I_DETECT3	"OTHERWISE" (default)	
TS2_DETECT0	"OTHERWISE" (default)	
TS2_DETECT1	"OTHERWISE" (default)	
TS2_DETECT2	"OTHERWISE" (default)	
TS2_DETECT3	"OTHERWISE" (default)	
TS2I_DETECT0	"OTHERWISE" (default)	
TS2I_DETECT1	"OTHERWISE" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
TS2I_DETECT2	"OTHERWISE" (default)	
TS2I_DETECT3	"OTHERWISE" (default)	
TX_AMP_RATIO_MARGIN0_FULL	"0'b10000000" (default)	
TX_AMP_RATIO_MARGIN0_HALF	"0'b01010000" (default)	
TX_AMP_RATIO_MARGIN1_FULL	"0'b01111000" (default)	
TX_AMP_RATIO_MARGIN1_HALF	"0'b01011000" (default)	
TX_AMP_RATIO_MARGIN2_FULL	"0'b01101000" (default)	
TX_AMP_RATIO_MARGIN2_HALF	"0'b01001000" (default)	
TX_AMP_RATIO_MARGIN3_FULL	"0'b01100000" (default)	
TX_AMP_RATIO_MARGIN3_HALF	"0'b01000000" (default)	
TX_AMP_RATIO_MARGIN4_FULL	"0'b01011000" (default)	
TX_AMP_RATIO_MARGIN4_HALF	"0'b00111000" (default)	
TX_AMP_RATIO_MARGIN5_FULL	"0'b01010000" (default)	
TX_AMP_RATIO_MARGIN5_HALF	"0'b00110000" (default)	
TX_AMP_RATIO_MARGIN6_FULL	"0'b01001000" (default)	
TX_AMP_RATIO_MARGIN6_HALF	"0'b00101000" (default)	
TX_AMP_RATIO_MARGIN7_FULL	"0'b01000000" (default)	
TX_AMP_RATIO_MARGIN7_HALF	"0'b00100000" (default)	
TX_BYPASS_DECODE_EN	"ENABLED" (default)	
TX_BYPASS_MSG_DEC	"NORMAL_OPERATION" (default)	
TX_CM_DIS	"DEASSERTED" (default)	
TX_COMP_RECEIVE	"DOES_NOT_ASSERT" (default)	
TX_CONVERT_UR_TO_CA	"NORMAL_OPERATION" (default)	
TX_D_ALLOC_N	"0'b0000000000000110" (default)	
TX_D_ALLOC_P	"0'b0000000001101100" (default)	
TX_DIV_MODE0	"0'b10" (default)	
TX_DIV_MODE1	"0'b10" (default)	
TX_DIV_MODE2	"0'b10" (default)	
TX_ECC1_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
TX_ECC1_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_ECC1_INJECT_M_1_N	"INJECT_1_ERR" (default)	
TX_ECC1_INJECT_TYPE	"POSTED_DATA_RAM" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
TX_ECC1_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_ECC2_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
TX_ECC2_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_ECC2_INJECT_M_1_N	"INJECT_1_ERR" (default)	
TX_ECC2_INJECT_TYPE	"POSTED_DATA_RAM" (default)	
TX_ECC2_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_EQ_EVAL_CNT_SEL	"WAIT_8_CLKS" (default)	
TX_ERR_COR	"DISABLED" (default)	
TX_ERR_ECC1	"OTHERWISE" (default)	
TX_ERR_ECC2	"OTHERWISE" (default)	
TX_ERR_PAR	"OTHERWISE" (default)	
TX_ERR_UCOR	"DISABLED" (default)	
TX_FORCE_RO	"DISABLED" (default)	
TX_GAP_INJECT_EN	"DO_NOT_INJECT_GAP" (default)	
TX_H_ALLOC_C	"0’b000000100000" (default)	
TX_H_ALLOC_N	"0’b000000001000" (default)	
TX_H_ALLOC_P	"0’b000000010000" (default)	
TX_HIZ	"IGNORED" (default)	
TX_IMPED_RATIO	"0’b10000000" (default)	
TX_PAR_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_PAR_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_PAR1_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_PAR2_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
TX_PAR2_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_PAR2_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_POLINV	"NORMAL" (default)	
TX_PRE_RATIO	"0’b00000000" (default)	
TX_PRE_RATIO_DEEMP0_FULL	"0’b00000000" (default)	
TX_PRE_RATIO_DEEMP0_HALF	"0’b00000000" (default)	
TX_PRE_RATIO_DEEMP1_FULL	"0’b00000000" (default)	
TX_PRE_RATIO_DEEMP1_HALF	"0’b00000000" (default)	
TX_PRIORITY	"DISABLED" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
TX_PRIORITY_N_STARVE_THRESH	"0'b00010000" (default)	
TX_PRIORITY_P_STARVE_THRESH	"0'b00010000" (default)	
TX_PST_RATIO	"0'b00010101" (default)	
TX_PST_RATIO_DEEMP0_FULL	"0'b00100000" (default)	
TX_PST_RATIO_DEEMP0_HALF	"0'b00100000" (default)	
TX_PST_RATIO_DEEMP1_FULL	"0'b00010101" (default)	
TX_PST_RATIO_DEEMP1_HALF	"0'b00010101" (default)	
TX_QUIESCE	"DISABLED" (default)	
TX_REPLAY_ECC1_HANDLE_DISABLE	"ENABLE_CORRECTION" (default)	
TX_REPLAY_ECC1_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_REPLAY_ECC1_INJECT_M_1_N	"INJECT_1_ERR" (default)	
TX_REPLAY_ECC1_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_REPLAY_ECC2_HANDLE_DISABLE	"ENABLE_HANDLING" (default)	
TX_REPLAY_ECC2_INJECT_EN	"DO_NOT_INJECT_ERR" (default)	
TX_REPLAY_ECC2_INJECT_M_1_N	"INJECT_1_ERR" (default)	
TX_REPLAY_ECC2_REPORT_DISABLE	"ENABLE_REPORTING" (default)	
TX_REQ_EQ	"DISABLED" (default)	
TX_SELECT_RX_FEEDBACK	"REFCLK" (default)	
TX_TLP_VALID	"DISABLED" (default)	
TXF_A	"0'b0100" (default)	
TXF_B	"0'b0100" (default)	
TXF_C	"0'b0100" (default)	
TXHF_CLKDN	"ENABLED" (default)	
TXM_A	"0'b10" (default)	
TXM_B	"0'b01" (default)	
TXM_C	"0'b00" (default)	
TXN_A	"0'b00100" (default)	
TXN_B	"0'b01001" (default)	
TXN_C	"0'b01111" (default)	
TXPLL_INIT	"DISABLED" (default)	
TXPLLRST	"DISABLED" (default)	
TYPE1_TYPE0_N	"ENDPOINT" (default)	

Table 520: Parameters (Continued)

Name	Values	Description
U_CLK_PERIOD	"0'b0001111101000000" (default)	
US_PORT_PS_ENTRY_TIME	"0'b0000000000000000" (default)	
US_PORT_RX_PRESET_HINT	"0'b010" (default)	
US_PORT_TX_PRESET	"0'b0100" (default)	
USE_COEF_PRE_MTHD_CTRL	"PRESET_VALUE" (default)	
USE_COEF_UPDN_CTRL	"PRESET_VALUE" (default)	
USER_AUTO_N	"AUTOMATIC_ON_RECEPTION" (default)	
VALUE_8G_POST	"0'b000000" (default)	
VALUE_8G_PRE	"0'b000000" (default)	
VEC_MASK_CAPABLE_MSICAP_A	"ENABLED" (default)	
VEC_MASK_CAPABLE_MSICAP_B	"ENABLED" (default)	
VEC_MASK_CAPABLE_MSICAP_C	"ENABLED" (default)	
VEC_MASK_CAPABLE_MSICAP_D	"ENABLED" (default)	
VENDOR_ID_ID1A	"0'b0001100110101010" (default)	
VENDOR_ID_ID1B	"0'b0001100110101010" (default)	
VENDOR_ID_ID1C	"0'b0001100110101010" (default)	
VENDOR_ID_ID1D	"0'b0001100110101010" (default)	
VENDOR0_UR	"REPORT" (default)	
VERSION_AER_CAP	"VER_0X2" (default)	
VERSION_PM_CAP	"0'b011" (default)	
XLCY0	"0'b00000000" (default)	
XLCY1	"0'b00000000" (default)	
COEF5_PRE	"0'b000001" (default)	
DIRECTED_RETRAIN_LINK	"OTHERWISE" (default)	
DIS_MSI_CAP_D	"ENABLED" (default)	
EN_RBAR_CAP_B	"ENABLED" (default)	
GAIN_TIMER1	"0'b0101" (default)	
MIN_SPEED	"2P5G" (default)	
OVERRIDE	"DISABLED" (default)	
PRESET_LPBK_SLAVE_2	"OTHERWISE" (default)	
SPEED	"8G" (default)	
TX_D_ALLOC_C	"0'b0000000001100000" (default)	

PCIELL

PCIe logical link layer

Note:

You should not try to use this primitive directly. Instead, use the related IP in IP Catalog. The following information is for reference only.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

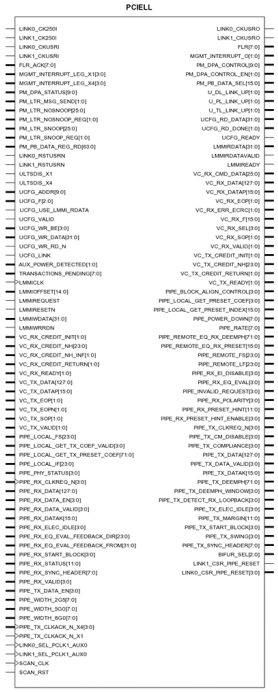


Table 521: Input Ports

Name	Description
LINK0_CK250I	Connect to pcs pipe clk internally for link0
LINK1_CK250I	Connect to pcs pipe clk internally for link1
LINK0_CKUSRI	User clock for link0 from Fabric clock
LINK1_CKUSRI	User clock for link1 from Fabric clock

Table 521: Input Ports (Continued)

Name	Description
FLR_ACK[7:0]	Function level reset acknowledge
MGMT_INTERRUPT_LEG_X1[3:0]	Legacy Interrupt input for link1
MGMT_INTERRUPT_LEG_X4[3:0]	Legacy Interrupt input for link0
PM_DPA_STATUS[9:0]	Dynamic Power Allocation for 2 links
PM_LTR_MSG_SEND[1:0]	Latency Tolerance Reporting
PM_LTR_NOSNOOP[25:0]	Latency Tolerance Reporting
PM_LTR_NOSNOOP_REQ[1:0]	Latency Tolerance Reporting
PM_LTR_SNOOP[25:0]	Latency Tolerance Reporting
PM_LTR_SNOOP_REQ[1:0]	Latency Tolerance Reporting
PM_PB_DATA_REG_RD[63:0]	Power Budgeting
LINK0_RSTUSRN	User link0 reset
LINK1_RSTUSRN	User link1 reset
ULTSDIS_X1	Disables LTSSM for link1
ULTSDIS_X4	Disables LTSSM for link0
UCFG_ADDR[9:0]	Register Address/Offset
UCFG_F[2:0]	Function number
UCFG_USE_LMMI_RDATA	Use LMMI read data port
UCFG_VALID	Transaction request valid signal
UCFG_WR_BE[3:0]	Byte Enable
UCFG_WR_DATA[31:0]	Write data
UCFG_WR_RD_N	Write=1, Read=0
UCFG_LINK	Which link is making the request
AUX_POWER_DETECTED[1:0]	Auxiliary Power detected
TRANSACTIONS_PENDING[7:0]	Non-posted transactions pending
LMMICLK	LMMI clock
LMMIOFFSET[14:0]	LMMI address offset
LMMIREQUEST	LMMI start transaction
LMMIRESETN	LMMI reset
LMMIWDATA[31:0]	LMMI write data
LMMIWRRDN	LMMI write and read control, 1: write, 0: read
VC_RX_CREDIT_INIT[1:0]	Transaction layer, initialize credit counters
VC_RX_CREDIT_NH[23:0]	Transaction layer, number of NH credit

Table 521: Input Ports (Continued)

Name	Description
VC_RX_CREDIT_NH_INF[1:0]	Transaction Layer, infinite NH credits
VC_RX_CREDIT_RETURN[1:0]	Transaction Layer, NH credit return signal
VC_RX_READY[1:0]	Transaction layer, destination ready signal
VC_TX_DATA[127:0]	TLP Tx Data which can support up to 4 lanes
VC_TX_DATAP[15:0]	TLP Tx Data Parity
VC_TX_EOP[1:0]	Transaction layer, end of packet
VC_TX_EOPN[1:0]	Transaction layer, end of nullified packet
VC_TX_SOP[1:0]	Transaction layer, start of packet
VC_TX_VALID[1:0]	Transaction layer, source valid signal
PIPE_LOCAL_FS[23:0]	Pipe signal for 4 lanes
PIPE_LOCAL_GET_TX_COEF_VALID[3:0]	Pipe signal for 4 lanes
PIPE_LOCAL_GET_TX_PRESET_COEF[71:0]	Pipe signal for 4 lanes
PIPE_LOCAL_IF[23:0]	Pipe signal for 4 lanes
PIPE_PHY_STATUS[3:0]	Pipe signal for 4 lanes
PIPE_RX_CLKREQ_N[3:0]	Pipe signal for 4 lanes
PIPE_RX_DATA[127:0]	Pipe signal for 4 lanes
PIPE_RX_DATA_EN[3:0]	Pipe signal for 4 lanes
PIPE_RX_DATA_VALID[3:0]	Pipe signal for 4 lanes
PIPE_RX_DATAK[15:0]	Pipe signal for 4 lanes
PIPE_RX_ELEC_IDLE[3:0]	Pipe signal for 4 lanes
PIPE_RX_EQ_EVAL_FEEDBACK_DIR[23:0]	Pipe signal for 4 lanes
PIPE_RX_EQ_EVAL_FEEDBACK_FROM[31:0]	Pipe signal for 4 lanes
PIPE_RX_START_BLOCK[3:0]	Pipe signal for 4 lanes
PIPE_RX_STATUS[11:0]	Pipe signal for 4 lanes
PIPE_RX_SYNC_HEADER[7:0]	Pipe signal for 4 lanes
PIPE_RX_VALID[3:0]	Pipe signal for 4 lanes
PIPE_TX_DATA_EN[3:0]	Pipe signal for 4 lanes
PIPE_WIDTH_2G5[7:0]	Pipe signal for 4 lanes
PIPE_WIDTH_5G0[7:0]	Pipe signal for 4 lanes
PIPE_WIDTH_8G0[7:0]	Pipe signal for 4 lanes
PIPE_TX_CLKACK_N_X4[3:0]	PCS status for link0
PIPE_TX_CLKACK_N_X1	PCS status for link1

Table 521: Input Ports (Continued)

Name	Description
LINK0_SEL_PCLK1_AUX0	PCS select pipe clock or aux clock as main clock in link0
LINK1_SEL_PCLK1_AUX0	PCS select pipe clock or aux clock as main clock in link1
SCAN_CLK	Scan clock for PCIe-LL
SCAN_RST	Scan reset for PCIe-LL

Table 522: Output Ports

Name	Description
LINK0_CKUSRO	User clock for link0 from Fabric clock
LINK1_CKUSRO	User clock for link1 from Fabric clock
FLR[7:0]	Function level reset
MGMT_INTERRUPT_O[1:0]	Core interrupt output for 2 links
PM_DPA_CONTROL[9:0]	Dynamic Power Allocation for 2 links
PM_DPA_CONTROL_EN[1:0]	Dynamic Power Allocation for 2 links
PM_PB_DATA_SEL[15:0]	Power Budgeting
U_DL_LINK_UP[1:0]	Data Link Layer Link Up Status (1) Up (0) Down
U_PL_LINK_UP[1:0]	Physical Link Layer Link Up Status (1) Up (0) Down
U_TL_LINK_UP[1:0]	Transaction Link Layer Link Up Status (1) Up (0) Down
UCFG_RD_DATA[31:0]	Read data
UCFG_RD_DONE[1:0]	Read data available for 2 links
UCFG_READY	Core ready signal
LMMIRDATA[31:0]	Read data
LMMIRDATAVALID	Valid data indicator
LMMIREADY	Slave ready signal.
VC_RX_CMD_DATA[25:0]	TLP decode information
VC_RX_DATA[127:0]	TLP Rx Data which can support up to 4 lanes
VC_RX_DATAP[15:0]	TLP Rx Data Parity
VC_RX_EOP[1:0]	End of packet for 2 links
VC_RX_ERR_ECRC[1:0]	End of packet with ECRC error
VC_RX_F[15:0]	Physical Function hit by transaction
VC_RX_SEL[3:0]	TLP type (0=Posted, 1=Non Posted, 2=Completion)
VC_RX_SOP[1:0]	Start of packet for 2 links
VC_RX_VALID[1:0]	Source Valid signal for 2 links

Table 522: Output Ports (Continued)

Name	Description
VC_TX_CREDIT_INIT[1:0]	Initialize credit counters
VC_TX_CREDIT_NH[23:0]	Number of NH credits
VC_TX_CREDIT_RETURN[1:0]	NH Credit return signal
VC_TX_READY[1:0]	Destination ready signal
PIPE_BLOCK_ALIGN_CONTROL[3:0]	Pipe signal for 4 lanes
PIPE_LOCAL_GET_PRESET_COEF[3:0]	Pipe signal for 4 lanes
PIPE_LOCAL_GET_PRESET_INDEX[15:0]	Pipe signal for 4 lanes
PIPE_POWER_DOWN[7:0]	Pipe signal for 4 lanes
PIPE_RATE[7:0]	Pipe signal for 4 lanes
PIPE_REMOTE_EQ_RX_DEEMPH[71:0]	Pipe signal for 4 lanes
PIPE_REMOTE_EQ_RX_PRESET[15:0]	Pipe signal for 4 lanes
PIPE_REMOTE_FS[23:0]	Pipe signal for 4 lanes
PIPE_REMOTE_LF[23:0]	Pipe signal for 4 lanes
PIPE_RX_EI_DISABLE[3:0]	Pipe signal for 4 lanes
PIPE_RX_EQ_EVAL[3:0]	Pipe signal for 4 lanes
PIPE_INVALID_REQUEST[3:0]	Pipe signal for 4 lanes
PIPE_RX_POLARITY[3:0]	Pipe signal for 4 lanes
PIPE_RX_PRESET_HINT[11:0]	Pipe signal for 4 lanes
PIPE_RX_PRESET_HINT_ENABLE[3:0]	Pipe signal for 4 lanes
PIPE_TX_CLKREQ_N[3:0]	Pipe signal for 4 lanes
PIPE_TX_CM_DISABLE[3:0]	Pipe signal for 4 lanes
PIPE_TX_COMPLIANCE[3:0]	Pipe signal for 4 lanes
PIPE_TX_DATA[127:0]	Pipe signal for 4 lanes
PIPE_TX_DATA_VALID[3:0]	Pipe signal for 4 lanes
PIPE_TX_DATAK[15:0]	Pipe signal for 4 lanes
PIPE_TX_DEEMPH[71:0]	Pipe signal for 4 lanes
PIPE_TX_DEEMPH_WINDOW[3:0]	Pipe signal for 4 lanes
PIPE_TX_DETECT_RX_LOOPBACK[3:0]	Pipe signal for 4 lanes
PIPE_TX_ELEC_IDLE[3:0]	Pipe signal for 4 lanes
PIPE_TX_MARGIN[11:0]	Pipe signal for 4 lanes
PIPE_TX_START_BLOCK[3:0]	Pipe signal for 4 lanes
PIPE_TX_SWING[3:0]	Pipe signal for 4 lanes

Table 522: Output Ports (Continued)

Name	Description
PIPE_TX_SYNC_HEADER[7:0]	Pipe signal for 4 lanes
BIFUR_SEL[2:0]	bifur_sel to PCS_top, PCS_top will use this signal to do ln2lk processing
LINK1_CSR_PIPE_RESET	Pipe reset for lane3 in link1
LINK0_CSR_PIPE_RESET[3:0]	Pipe reset for 4 lanes in link0

PCLKDIV

Clock Divider for Primary Clock

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

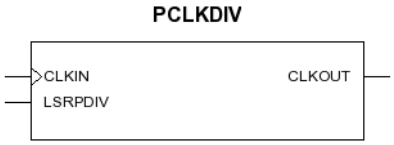


Table 523: Input Ports

Name	Range	Description
CLKIN	1	Input clock
LSRPDIV	1	Local set/reset

Table 524: Output Ports

Name	Range	Description
CLKOUT	1	Output divided clock

Table 525: Parameters

Name		Description	Outputs
DIV_PCLKDIV	X1 (default)	Clock divider	CLKIN/1
	X2		CLKIN/2
	X4		CLKIN/4
	X8		CLKIN/8
	X16		CLKIN/16
	X32		CLKIN/32
	X64		CLKIN/64
	X128		CLKIN/128
GSR	"ENABLED" (default) "DISABLED"	Global set/reset	

PCLKDIVSP

Clock Divider for Primary Clock

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

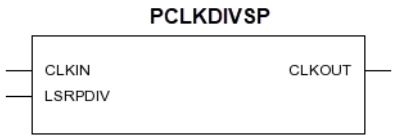


Table 526: Input Ports

Name	Description
CLKIN	Input clock
LSRPDIV	Local set/reset

Table 527: Output Ports

Name	Description
CLKOUT	Output divided clock

Table 528: Parameters

Name	Values	Description
DIV_PCLKDIV	"X1"	Divider value. Other values X2, X4, X8, X16, X32, X64, X128
GSR	"ENABLED"	Enable global set/reset

PCSREFMUX

Multiplexer of reference clock for PCS. SW provides one set of reference clock, 40K could instantiate one, 80k could instantiate twice and 120k could have three instances.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

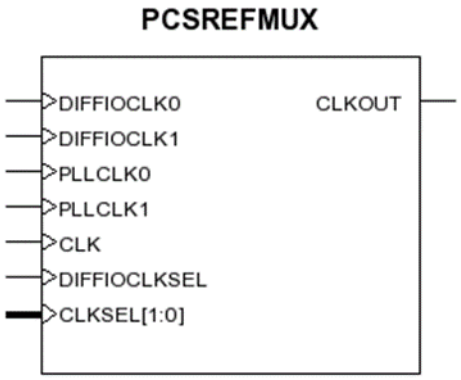


Table 529: Input Ports

Name	Description
DIFFIOCLK0	PMA PLL refclk from external IO pad0
DIFFIOCLK1	PMA PLL refclk from external IO pad1
PLLCLK0	MA PLL refclk from GPLL
PLLCLK1	MA PLL refclk from GPLL
CLK	PMA PLL refclk from CIB
DIFFIOCLKSEL	Dynamic selection between I/O pad0 and IO pad1 for PCS quad0
CLKSEL[1:0]	Dynamic selection among I/O pad, PLL and CIB for PCS quad0

Table 530: Output Ports

CLKOUT	PMA PLL refclk output from refclk_mux_tree

PCSX1

Wrapper of PCSX1 and MPCS. PCS hard IP block is quad-base but each channel has independent ports, controls, registers and setting to support channel-base, dual-channel-base and quad-base usage.

There are four software parameters:

1. **PROTOCOL:** The protocol implemented in all channels of current PCS instance. The protocol should be set when soft IP instantiates the primitive.
2. **DATARATE:** The data rate of the protocol. The data rate should be set if corresponding data rate exists for the protocol.

3. **LANE_ID:** User could specify it to locate the PCS instance in device. LANE_ID should be unique among all PCS instances in the design. Otherwise, software will report error. ‘AUTO’ is default that SW could determine automatically.
4. **GRP_NAME:** If multiple PCS instances are used to implement the protocol, same GRP_NAME should be set in these instances so that software binds them together.

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP



Table 531: Input Ports

Name	Description
PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
TXCLK	User interface tx clock input (From FPGA)
RXCLK	User interface rx clock input (From FPGA)
CCCLK	CTC clock from Fabric
TX_ALIGN_CLKIN	Clock for multi quad usage in TX mode
RX_ALIGN_CLKIN	Clock for multi quad usage in RX mode
PERSTN	Fundamental reset. Will trigger PCS auto calibration
MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
LMMICLK	LMMI Clock (from FPGA)
LMMIRESETN	LMMI Reset
LMMIREQUEST	LMMI start transaction
LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
LMMIOFFSET8 LMMIOFFSET0	LMMI Word Address/Offset
LMMIWDATA7 LMMIWDATA0	LMMI write data
SDRXP	Analog RX diffIO
SDRXN	Analog RX diffIO
SDREXT	External resistance
SDREFRET	
SDQ_REFCLKP	Refclk of IO EOM pll in one quad
SDQ_REFCLKN	Refclk of IO EOM pll in one quad
AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
ALTCLKIN	This is from refclk_mux_tree
ALTCLK_SEL	Selection for alt_clk_in and independent diffIO refclk
CLKREQI	Open drain bidirectional CLKREQ# input
TX_LALIGN_IN_UP7 TX_LALIGN_IN_UP0	The connection signals between Quads for lane alignment across Quad boundary
TX_LALIGN_IN_DOWN7 TX_LALIGN_IN_DOWN0	The connection signals between Quads for lane alignment across Quad boundary

Table 531: Input Ports (Continued)

PIPE_POWER_DOWN1 PIPE_POWER_DOWN0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RATE1 PIPE_RATE0	Pipe signal open for user, mpcs and ecps sharing
PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_PRESET_HINT2 PIPE_RX_PRESET_HINT0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_MARGIN2 PIPE_TX_MARGIN0	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DEEMPH17 PIPE_TX_DEEMPH2	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DEEMPH_PCS_TX_DATA1 PIPE_TX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DATA_PCS_TX_DATA31 PIPE_TX_DATA_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DATAK_PCS_TX_DATA3 PIPE_TX_DATAK_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_SYNC_HDR_PCS_TX_DATA1 PIPE_TX_SYNC_HDR_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing

Table 531: Input Ports (Continued)

PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
PCIELL_MODESEL2 PCIELL_MODESEL0	0=x4 mode, 1=x2 mode, 2=x1 mode, 3=x2/x1 mode, 4=x1/x1 mode
PIPE_POWER_DOWN_LL1 PIPE_POWER_DOWN_LL0	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
PIPE_RATE_LL1 PIPE_RATE_LL0	PHY Rate Control.11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise
PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal
PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommneds not to use
PIPE_RX_PRESET_HINT_LL2 PIPE_RX_PRESET_HINT_LL0	Local PHY Rx Preset Hint
PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing)
PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
PIPE_L1PMSSEN_LL	L1 substates enable
PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing

Table 531: Input Ports (Continued)

PIPE_TX_MARGIN_LL2 PIPE_TX_MARGIN_LL0	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
PIPE_TX_DEEMPH_LL17 PIPE_TX_DEEMPH_LL0	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
PIPE_LOCAL_GET_PRESET_INDEX_LL3 PIPE_LOCAL_GET_PRESET_INDEX_LL0	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC asks to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
PIPE_REMOTE_FS_LL5 PIPE_REMOTE_FS_LL0	Link partner: FS value the link partner advertised at the beginning of Equalization training
PIPE_REMOTE_LF_LL5 PIPE_REMOTE_LF_LL0	Link partner: LF value the link partner advertised at the beginning of Equalization training
PIPE_REMOTE_EQ_RX_DEEMPH_LL17 PIPE_REMOTE_EQ_RX_DEEMPH_LL0	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
PIPE_REMOTE_EQ_RX_PRESET_LL3 PIPE_REMOTE_EQ_RX_PRESET_LL0	Link partner: 8G preset value we requested the link partner use during Equalization training
PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
PIPE_TX_SYNC_HEADER_LL1 PIPE_TX_SYNC_HEADER_LL0	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
PIPE_TX_DATA_LL31 PIPE_TX_DATA_LL0	PIPE Tx Data
PIPE_TX_DATAK_LL3 PIPE_TX_DATAK_LL0	PIPE Tx Data K indicator for 8b10b
PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active

Table 531: Input Ports (Continued)

PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
PIPE_PRESET_INDX_PCS_TX_DATA3 PIPE_PRESET_INDX_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_REMOTE_FS_PCS_TX_DATA5 PIPE_REMOTE_FS_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_REMOTE_LF_PCS_TX_DATA5 PIPE_REMOTE_LF_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_DEEMPH_PCS_TX_DATA17 PIPE_RX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_PRESET_PCS_TX_DATA3 PIPE_RX_PRESET_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
ACTMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
SCAN_CLK	Scan clock for PCS
SCAN_RST	Scan reset for PCS

Table 532: Output Ports

Name	Description
PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
RXOUTCLK	PCS RX clock to Fabric
TX_ALIGN_CLKOUT	Clock for multi quad usage in TX mode
RX_ALIGN_CLKOUT	Clock for multi quad usage in RX mode
LMMIRDATA_VALID	LMMI read data valid
LMMIREADY	LMMI Bus ready
LMMIRDATA7 LMMIRDATA0	LMMI read data

Table 532: Output Ports (Continued)

SDTXP	Analog TX diffIO
SDTXN	Analog TX diffIO
PCLKAUX_SEL_LL	Pipe clock selection for user
CLKREQO	Open drain bidirectional CLKREQ# output
CLKREQOE	Open drain bidirectional CLKREQ# output enable
TX_LALIGN_OUT_UP7 TX_LALIGN_OUT_UP0	The connection signals between Quads for lane alignment across Quad boundary
TX_LALIGN_OUT_DOWN7 TX_LALIGN_OUT_DOWN0	The connection signals between Quads for lane alignment across Quad boundary
PIPE_WIDTH_5G01 PIPE_WIDTH_5G00	Pipe signal open for user, mpcs and ecps sharing
PIPE_WIDTH_8G01 PIPE_WIDTH_8G00	Pipe signal open for user, mpcs and ecps sharing
PIPE_RXEQ_FOM7 PIPE_RXEQ_FOM0	Pipe signal open for user, mpcs and ecps sharing
PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
PIPE_LOCAL_GET_TX_PRESET_COEF3 PIPE_LOCAL_GET_TX_PRESET_COEF0	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_PRESET_COEF_PCS_RX_DATA17 PIPE_TX_PRESET_COEF_PCS_RX_DATA4	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_DATA_PCS_RX_DATA31 PIPE_RX_DATA_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_STATUS_PCS_RX_DATA2 PIPE_RX_STATUS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_SYNC_HDR_PCS_RX_DATA1 PIPE_RX_SYNC_HDR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing

Table 532: Output Ports (Continued)

PIPE_WIDTH_2G5_LL1 PIPE_WIDTH_2G5_LL0	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved,10==[31:0], 01==[15:0], 00=[7:0]. Must be <=PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
PIPE_WIDTH_5G0_LL1 PIPE_WIDTH_5G0_LL0	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
PIPE_WIDTH_8G0_LL1 PIPE_WIDTH_8G0_LL0	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL7 PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL0	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1
PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL5 PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL0	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
PIPE_LOCAL_FS_LL5 PIPE_LOCAL_FS_LL0	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
PIPE_LOCAL_LF_LL5 PIPE_LOCAL_LF_LL0	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted
PIPE_LOCAL_GET_TX_PRESET_COEF_LL17 PIPE_LOCAL_GET_TX_PRESET_COEF_LL0	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0

Table 532: Output Ports (Continued)

PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
PIPE_RX_SYNC_HEADER_LL1 PIPE_RX_SYNC_HEADER_LL0	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
PIPE_RX_DATA_LL31 PIPE_RX_DATA_LL0	Pipe RX data
PIPE_RX_DATAK_LL3 PIPE_RX_DATAK_LL0	PIPE Rx Data K indicator for 8b10b
PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
PIPE_RX_STATUS_LL2 PIPE_RX_STATUS_LL0	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
PIPE_PCLKOUT_LL	Pipe clock connect to harden PCIe_LL. connect to pipe_pclkout_LL[0] goes to clk_250_i_link0 and pipe_pclkout_LL[3] goes to clk_250_i_link1
PIPE_WIDTH_2G51 PIPE_WIDTH_2G50	Pipe signal open for user, mpcs and ecps sharing
PIPE_RXEQ_DIR_PCS_RX_DATA3 PIPE_RXEQ_DIR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_RXEQ_DIR_PCS_RX_FIFO_ST5 PIPE_RXEQ_DIR_PCS_RX_FIFO_ST4	Pipe signal open for user, mpcs and ecps sharing
PIPE_LOCAL_FS_PCS_RX_DATA5 PIPE_LOCAL_FS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_LOCAL_LF_PCS_RX_DATA5 PIPE_LOCAL_LF_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
PIPE_RX_DATAK_PCS_RX_DATA3 PIPE_RX_DATAK_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
PCS_RX_DATA79 PCS_RX_DATA74	Dedicated ports used for pcs rx data

Table 532: Output Ports (Continued)

PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
PCS_TX_FIFO_ST3 PCS_TX_FIFO_ST2	Tx fifo status (empty, full)
ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

PCSX2

Wrapper of PCSX2 and MPCS. PCS hard IP block is quad-base but each channel has independent ports, controls, registers and setting to support channel-base, dual-channel-base and quad-base usage.

Architectures supported:

- LFCPNX
- UT24CP



Table 533: Input Ports

Name	Description
CH1_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH0_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH1_TXCLK	User interface tx clock input (From FPGA)
CH0_TXCLK	User interface tx clock input (From FPGA)
CH1_RXCLK	User interface rx clock input (From FPGA)

Table 533: Input Ports (Continued)

CH0_RXCLK	User interface rx clock input (From FPGA)
CH1_CCCLK	CTC clock from Fabric
CH0_CCCLK	CTC clock from Fabric
TX_ALIGN_CLKIN	Clock for multi quad usage in TX mode
RX_ALIGN_CLKIN	Clock for multi quad usage in RX mode
CH1_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH0_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH1_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH0_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH1_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH0_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH1_LMMICLK	LMMI Clock (from FPGA)
CH0_LMMICLK	LMMI Clock (from FPGA)
CH1_LMMIRESETN	LMMI Reset
CH0_LMMIRESETN	LMMI Reset
CH1_LMMIREQUEST	LMMI start transaction
CH0_LMMIREQUEST	LMMI start transaction
CH1_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH0_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH1_LMMIOFFSET8 CH1_LMMIOFFSET0	LMMI Word Address/Offset
CH0_LMMIOFFSET8 CH0_LMMIOFFSET0	LMMI Word Address/Offset
CH1_LMMIWDATA7 CH1_LMMIWDATA0	LMMI write data
CH0_LMMIWDATA7 CH0_LMMIWDATA0	LMMI write data
CH1_SDRXP	Analog RX diffIO
CH0_SDRXP	Analog RX diffIO
CH1_SDRXN	Analog RX diffIO
CH0_SDRXN	Analog RX diffIO
CH1_SDREXT	External resistance
CH0_SDREXT	External resistance
CH1_SDREFRET	

Table 533: Input Ports (Continued)

CH0_SDREFRET	
SDQ_REFCLKP	Refclk of IO EOM pll in one quad
SDQ_REFCLKN	Refclk of IO EOM pll in one quad
CH1_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
CH0_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
ALTCLKIN	This is from refclk_mux_tree
ALTCLK_SEL	Selection for alt_clk_in and independent diffIO refclk
CH1_CLKREQI	Open drain bidirectional CLKREQ# input
CH0_CLKREQI	Open drain bidirectional CLKREQ# input
TX_LALIGN_IN_UP7 TX_LALIGN_IN_UP0	The connection signals between Quads for lane alignment across Quad boundary
TX_LALIGN_IN_DOWN7 TX_LALIGN_IN_DOWN0	The connection signals between Quads for lane alignment across Quad boundary
CH1_PIPE_POWER_DOWN1 CH1_PIPE_POWER_DOWN0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_POWER_DOWN1 CH0_PIPE_POWER_DOWN0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RATE1 CH1_PIPE_RATE0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RATE1 CH0_PIPE_RATE0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_HINT2 CH1_PIPE_RX_PRESET_HINT0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_HINT2 CH0_PIPE_RX_PRESET_HINT0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing

Table 533: Input Ports (Continued)

CH0_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_MARGIN2 CH1_PIPE_TX_MARGIN0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_MARGIN2 CH0_PIPE_TX_MARGIN0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DEEMPH17 CH1_PIPE_TX_DEEMPH2	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DEEMPH17 CH0_PIPE_TX_DEEMPH2	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DEEMPH_PCS_TX_DATA1 CH1_PIPE_TX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DEEMPH_PCS_TX_DATA1 CH0_PIPE_TX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_PCS_TX_DATA31 CH1_PIPE_TX_DATA_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing

Table 533: Input Ports (Continued)

CH0_PIPE_TX_DATA_PCS_TX_DATA31 CH0_PIPE_TX_DATA_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATAK_PCS_TX_DATA3 CH1_PIPE_TX_DATAK_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATAK_PCS_TX_DATA3 CH0_PIPE_TX_DATAK_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_SYNC_HDR_PCS_TX_DATA1 CH1_PIPE_TX_SYNC_HDR_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_SYNC_HDR_PCS_TX_DATA1 CH0_PIPE_TX_SYNC_HDR_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
CH1_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
CH0_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
PCIELL_MODESEL2 PCIELL_MODESEL0	0=x4 mode, 1=x2 mode, 2=x1 mode, 3=x2/x1 mode, 4=x1/x1 mode
CH1_PIPE_POWER_DOWN_LL1 CH1_PIPE_POWER_DOWN_LL0	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
CH0_PIPE_POWER_DOWN_LL1 CH0_PIPE_POWER_DOWN_LL0	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
CH1_PIPE_RATE_LL1 CH1_PIPE_RATE_LL0	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH0_PIPE_RATE_LL1 CH0_PIPE_RATE_LL0	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH1_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH0_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH1_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings

Table 533: Input Ports (Continued)

CH0_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
CH1_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal.
CH0_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal.
CH1_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommneds not to use.
CH0_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommneds not to use.
CH1_PIPE_RX_PRESET_HINT_LL2 CH1_PIPE_RX_PRESET_HINT_LL0	Local PHY Rx Preset Hint
CH0_PIPE_RX_PRESET_HINT_LL2 CH0_PIPE_RX_PRESET_HINT_LL0	Local PHY Rx Preset Hint
CH1_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing).
CH0_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing).
CH1_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH0_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH1_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise

Table 533: Input Ports (Continued)

CH0_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
CH1_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH0_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH1_PIPE_L1PMSSEN_LL	L1 substates enable
CH0_PIPE_L1PMSSEN_LL	L1 substates enable
CH1_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH0_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH1_PIPE_TX_MARGIN_LL2 CH1_PIPE_TX_MARGIN_LL0	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH0_PIPE_TX_MARGIN_LL2 CH0_PIPE_TX_MARGIN_LL0	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH1_PIPE_TX_DEEMPH_LL17 CH1_PIPE_TX_DEEMPH_LL0	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH0_PIPE_TX_DEEMPH_LL17 CH0_PIPE_TX_DEEMPH_LL0	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH1_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
CH0_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
CH1_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH0_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH1_PIPE_LOCAL_GET_PRESET_INDEX_LL3 CH1_PIPE_LOCAL_GET_PRESET_INDEX_LL0	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.

Table 533: Input Ports (Continued)

CH0_PIPE_LOCAL_GET_PRESET_INDEX_LL3 CH0_PIPE_LOCAL_GET_PRESET_INDEX_LL0	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH1_PIPE_REMOTE_FS_LL5 CH1_PIPE_REMOTE_FS_LL0	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH0_PIPE_REMOTE_FS_LL5 CH0_PIPE_REMOTE_FS_LL0	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH1_PIPE_REMOTE_LF_LL5 CH1_PIPE_REMOTE_LF_LL0	Link partner: LF value the link partner advertised at the beginning of Equalization training
CH0_PIPE_REMOTE_LF_LL5 CH0_PIPE_REMOTE_LF_LL0	Link partner: LF value the link partner advertised at the beginning of Equalization training
CH1_PIPE_REMOTE_EQ_RX_DEEMPH_LL17 CH1_PIPE_REMOTE_EQ_RX_DEEMPH_LL0	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH0_PIPE_REMOTE_EQ_RX_DEEMPH_LL17 CH0_PIPE_REMOTE_EQ_RX_DEEMPH_LL0	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH1_PIPE_REMOTE_EQ_RX_PRESET_LL3 CH1_PIPE_REMOTE_EQ_RX_PRESET_LL0	Link partner: 8G preset value we requested the link partner use during Equalization training
CH0_PIPE_REMOTE_EQ_RX_PRESET_LL3 CH0_PIPE_REMOTE_EQ_RX_PRESET_LL0	Link partner: 8G preset value we requested the link partner use during Equalization training
CH1_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH0_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH1_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH0_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH1_PIPE_TX_SYNC_HEADER_LL1 CH1_PIPE_TX_SYNC_HEADER_LL0	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH0_PIPE_TX_SYNC_HEADER_LL1 CH0_PIPE_TX_SYNC_HEADER_LL0	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH1_PIPE_TX_DATA_LL31 CH1_PIPE_TX_DATA_LL0	PIPE Tx Data
CH0_PIPE_TX_DATA_LL31 CH0_PIPE_TX_DATA_LL0	PIPE Tx Data

Table 533: Input Ports (Continued)

CH1_PIPE_TX_DATAK_LL3 CH1_PIPE_TX_DATAK_LL0	PIPE Tx Data K indicator for 8b10b
CH0_PIPE_TX_DATAK_LL3 CH0_PIPE_TX_DATAK_LL0	PIPE Tx Data K indicator for 8b10b
CH1_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH0_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH1_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH0_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH1_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
CH0_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
CH1_PIPE_PRESET_INDX_PCS_TX_DATA3 CH1_PIPE_PRESET_INDX_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_PRESET_INDX_PCS_TX_DATA3 CH0_PIPE_PRESET_INDX_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_REMOTE_FS_PCS_TX_DATA5 CH1_PIPE_REMOTE_FS_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_REMOTE_FS_PCS_TX_DATA5 CH0_PIPE_REMOTE_FS_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_REMOTE_LF_PCS_TX_DATA5 CH1_PIPE_REMOTE_LF_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_REMOTE_LF_PCS_TX_DATA5 CH0_PIPE_REMOTE_LF_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DEEMPH_PCS_TX_DATA17 CH1_PIPE_RX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DEEMPH_PCS_TX_DATA17 CH0_PIPE_RX_DEEMPH_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_PCS_TX_DATA3 CH1_PIPE_RX_PRESET_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_PCS_TX_DATA3 CH0_PIPE_RX_PRESET_PCS_TX_DATA0	Pipe signal open for user, mpcs and ecps sharing
ACTMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

Table 533: Input Ports (Continued)

CH1_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
SCAN_CLK	Scan clock for PCS
SCAN_RST	Scan reset for PCS

Table 534: Output Ports

Name	Description
CH1_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH0_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH1_RXOUTCLK	PCS RX clock to Fabric
CH0_RXOUTCLK	PCS RX clock to Fabric
TX_ALIGN_CLKOUT	Clock for multi quad usage in TX mode
RX_ALIGN_CLKOUT	Clock for multi quad usage in RX mode
CH1_LMMIRDATA_VALID	LMMI read data valid
CH0_LMMIRDATA_VALID	LMMI read data valid
CH1_LMMIREADY	LMMI Bus ready
CH0_LMMIREADY	LMMI Bus ready
CH1_LMMIRDATA7 CH1_LMMIRDATA0	LMMI read data
CH0_LMMIRDATA7 CH0_LMMIRDATA0	LMMI read data
CH1_SDTXP	Analog TX diffIO

Table 534: Output Ports (Continued)

CH0_SDTXP	Analog TX diffIO
CH1_SDTXN	Analog TX diffIO
CH0_SDTXN	Analog TX diffIO
CH1_PCLKAUX_SEL_LL	Pipe clock selection for user
CH0_PCLKAUX_SEL_LL	Pipe clock selection for user
CH1_CLKREQO	Open drain bidirectional CLKREQ# output
CH0_CLKREQO	Open drain bidirectional CLKREQ# output
CH1_CLKREQOE	Open drain bidirectional CLKREQ# output enable
CH0_CLKREQOE	Open drain bidirectional CLKREQ# output enable
TX_LALIGN_OUT_UP7 TX_LALIGN_OUT_UP0	The connection signals between Quads for lane alignment across Quad boundary
TX_LALIGN_OUT_DOWN7 TX_LALIGN_OUT_DOWN0	The connection signals between Quads for lane alignment across Quad boundary
CH1_PIPE_WIDTH_5G01 CH1_PIPE_WIDTH_5G00	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_5G01 CH0_PIPE_WIDTH_5G00	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_WIDTH_8G01 CH1_PIPE_WIDTH_8G00	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_8G01 CH0_PIPE_WIDTH_8G00	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEQ_FOM7 CH1_PIPE_RXEQ_FOM0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_FOM7 CH0_PIPE_RXEQ_FOM0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing

Table 534: Output Ports (Continued)

CH1_PIPE_LOCAL_GET_TX_PRESET_COEF3 CH1_PIPE_LOCAL_GET_TX_PRESET_COEF0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_TX_PRESET_COEF3 CH0_PIPE_LOCAL_GET_TX_PRESET_COEF0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_PRESET_COEF_PCS_RX_DATA17 CH1_PIPE_TX_PRESET_COEF_PCS_RX_DATA4	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_PRESET_COEF_PCS_RX_DATA17 CH0_PIPE_TX_PRESET_COEF_PCS_RX_DATA4	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_PCS_RX_DATA31 CH1_PIPE_RX_DATA_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_PCS_RX_DATA31 CH0_PIPE_RX_DATA_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_STATUS_PCS_RX_DATA2 CH1_PIPE_RX_STATUS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_STATUS_PCS_RX_DATA2 CH0_PIPE_RX_STATUS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_SYNC_HDR_PCS_RX_DATA1 CH1_PIPE_RX_SYNC_HDR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_SYNC_HDR_PCS_RX_DATA1 CH0_PIPE_RX_SYNC_HDR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_WIDTH_2G5_LL1 CH1_PIPE_WIDTH_2G5_LL0	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH0_PIPE_WIDTH_2G5_LL1 CH0_PIPE_WIDTH_2G5_LL0	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH

Table 534: Output Ports (Continued)

CH1_PIPE_WIDTH_5G0_LL1 CH1_PIPE_WIDTH_5G0_LL0	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH0_PIPE_WIDTH_5G0_LL1 CH0_PIPE_WIDTH_5G0_LL0	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH1_PIPE_WIDTH_8G0_LL1 CH1_PIPE_WIDTH_8G0_LL0	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH0_PIPE_WIDTH_8G0_LL1 CH0_PIPE_WIDTH_8G0_LL0	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH1_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL7 CH1_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL0	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH0_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL7 CH0_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL0	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH1_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL5 CH1_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL0	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH0_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL5 CH0_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL0	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH1_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH0_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH1_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
CH0_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
CH1_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
CH0_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock

Table 534: Output Ports (Continued)

CH1_PIPE_LOCAL_FS_LL5 CH1_PIPE_LOCAL_FS_LL0	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH0_PIPE_LOCAL_FS_LL5 CH0_PIPE_LOCAL_FS_LL0	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH1_PIPE_LOCAL_LF_LL5 CH1_PIPE_LOCAL_LF_LL0	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
CH0_PIPE_LOCAL_LF_LL5 CH0_PIPE_LOCAL_LF_LL0	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
CH1_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH0_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH1_PIPE_LOCAL_GET_TX_PRESET_COEF_LL17 CH1_PIPE_LOCAL_GET_TX_PRESET_COEF_LL0	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH0_PIPE_LOCAL_GET_TX_PRESET_COEF_LL17 CH0_PIPE_LOCAL_GET_TX_PRESET_COEF_LL0	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH1_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0
CH0_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0
CH1_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH0_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH1_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH0_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding

Table 534: Output Ports (Continued)

CH1_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH0_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH1_PIPE_RX_SYNC_HEADER_LL1 CH1_PIPE_RX_SYNC_HEADER_LL0	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH0_PIPE_RX_SYNC_HEADER_LL1 CH0_PIPE_RX_SYNC_HEADER_LL0	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH1_PIPE_RX_DATA_LL31 CH1_PIPE_RX_DATA_LL0	Pipe RX data
CH0_PIPE_RX_DATA_LL31 CH0_PIPE_RX_DATA_LL0	Pipe RX data
CH1_PIPE_RX_DATAK_LL3 CH1_PIPE_RX_DATAK_LL0	PIPE Rx Data K indicator for 8b10b
CH0_PIPE_RX_DATAK_LL3 CH0_PIPE_RX_DATAK_LL0	PIPE Rx Data K indicator for 8b10b
CH1_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH0_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH1_PIPE_RX_STATUS_LL2 CH1_PIPE_RX_STATUS_LL0	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
CH0_PIPE_RX_STATUS_LL2 CH0_PIPE_RX_STATUS_LL0	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
PIPE_PCLKOUT_LL1 PIPE_PCLKOUT_LL0	Pipe clock connect to harden PCIe_LL. connect to pipe_pclkout_LL[0] goes to clk_250_i_link0, and pipe_pclkout_LL[3] goes to clk_250_i_link1
CH1_PIPE_WIDTH_2G51 CH1_PIPE_WIDTH_2G50	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_2G51 CH0_PIPE_WIDTH_2G50	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEQ_DIR_PCS_RX_DATA3 CH1_PIPE_RXEQ_DIR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_DIR_PCS_RX_DATA3 CH0_PIPE_RXEQ_DIR_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing

Table 534: Output Ports (Continued)

CH1_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST5 CH1_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST4	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST5 CH0_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST4	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_FS_PCS_RX_DATA5 CH1_PIPE_LOCAL_FS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_FS_PCS_RX_DATA5 CH0_PIPE_LOCAL_FS_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_LF_PCS_RX_DATA5 CH1_PIPE_LOCAL_LF_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_LF_PCS_RX_DATA5 CH0_PIPE_LOCAL_LF_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATAK_PCS_RX_DATA3 CH1_PIPE_RX_DATAK_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATAK_PCS_RX_DATA3 CH0_PIPE_RX_DATAK_PCS_RX_DATA0	Pipe signal open for user, mpcs and ecps sharing
CH1_PCS_RX_DATA79 CH1_PCS_RX_DATA74	Dedicated ports used for pcs rx data
CH0_PCS_RX_DATA79 CH0_PCS_RX_DATA74	Dedicated ports used for pcs rx data
CH1_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH0_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH1_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH0_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH1_PCS_TX_FIFO_ST3 CH1_PCS_TX_FIFO_ST2	Tx fifo status (empty, full)
CH0_PCS_TX_FIFO_ST3 CH0_PCS_TX_FIFO_ST2	Tx fifo status (empty, full)
CH1_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

Table 534: Output Ports (Continued)

CH0_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

PCSX4

Wrapper of PCSX4 and MPCS. PCS hard IP block is quad-base but each channel has independent ports, controls, registers and setting to support channel-base, dual-channel-base and quad-base usage.

PCSX1 uses one channel of PCSX4. PCSX2 uses two channels of PCSX4.

Note:

You should not try to use this primitive directly. Instead, use the related IP in IP Catalog. The following information is for reference only.

Architectures supported:

- LFCPNX
- LFMXO5T
- UT24CP



Table 535: Input Ports

Name	Description
CH3_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH2_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH1_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH0_PMACLKIN	This low speed clock will also drive PMA 'epcs_clkin' clock port
CH3_TXCLK	User interface tx clock input (From FPGA)
CH2_TXCLK	User interface tx clock input (From FPGA)

Table 535: Input Ports (Continued)

Name	Description
CH1_TXCLK	User interface tx clock input (From FPGA)
CH0_TXCLK	User interface tx clock input (From FPGA)
CH3_RXCLK	User interface rx clock input (From FPGA)
CH2_RXCLK	User interface rx clock input (From FPGA)
CH1_RXCLK	User interface rx clock input (From FPGA)
CH0_RXCLK	User interface rx clock input (From FPGA)
CH3_CCCLK	CTC clock from Fabric
CH2_CCCLK	CTC clock from Fabric
CH1_CCCLK	CTC clock from Fabric
CH0_CCCLK	CTC clock from Fabric
TX_ALIGN_CLKIN	Clock for multi quad usage in TX mode
RX_ALIGN_CLKIN	Clock for multi quad usage in RX mode
CH3_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH2_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH1_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH0_PERSTN	Fundamental reset. Will trigger PCS auto calibration
CH3_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH2_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH1_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH0_MPCS_TX_URSTN	MPCS core logic reset (TX path), but do not reset PMA phy
CH3_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH2_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH1_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH0_MPCS_RX_URSTN	MPCS core logic reset (RX path), but do not reset PMA phy
CH3_LMMICLK	LMMI Clock (from FPGA)
CH2_LMMICLK	LMMI Clock (from FPGA)
CH1_LMMICLK	LMMI Clock (from FPGA)
CH0_LMMICLK	LMMI Clock (from FPGA)
CH3_LMMIRESETN	LMMI Reset
CH2_LMMIRESETN	LMMI Reset
CH1_LMMIRESETN	LMMI Reset
CH0_LMMIRESETN	LMMI Reset

Table 535: Input Ports (Continued)

Name	Description
CH3_LMMIREQUEST	LMMI start transaction
CH2_LMMIREQUEST	LMMI start transaction
CH1_LMMIREQUEST	LMMI start transaction
CH0_LMMIREQUEST	LMMI start transaction
CH3_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH2_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH1_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH0_LMMIWR_RDN	LMMI Write=HIGH, LMMI Read=LOW
CH3_LMMIOFFSET[8:0]	LMMI Word Address/Offset
CH2_LMMIOFFSET[8:0]	LMMI Word Address/Offset
CH1_LMMIOFFSET[8:0]	LMMI Word Address/Offset
CH0_LMMIOFFSET[8:0]	LMMI Word Address/Offset
CH3_LMMIWDATA[7:0]	LMMI write data
CH2_LMMIWDATA[7:0]	LMMI write data
CH1_LMMIWDATA[7:0]	LMMI write data
CH0_LMMIWDATA[7:0]	LMMI write data
CH3_SDRXP	Analog RX diffIO
CH2_SDRXP	Analog RX diffIO
CH1_SDRXP	Analog RX diffIO
CH0_SDRXP	Analog RX diffIO
CH3_SDRXN	Analog RX diffIO
CH2_SDRXN	Analog RX diffIO
CH1_SDRXN	Analog RX diffIO
CH0_SDRXN	Analog RX diffIO
CH3_SDREXT	External resistance
CH2_SDREXT	External resistance
CH1_SDREXT	External resistance
CH0_SDREXT	External resistance
CH3_SDREFRET	
CH2_SDREFRET	
CH1_SDREFRET	
CH0_SDREFRET	

Table 535: Input Ports (Continued)

Name	Description
SDQ_REFCLKP	Refclk of IO EOM pll in one quad
SDQ_REFCLKN	Refclk of IO EOM pll in one quad
CH3_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
CH2_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
CH1_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
CH0_AUXCLK	Auxiliary clock from Fabric for PCIe-PCS core clock
ALTCLKIN	This is from refclk_mux_tree
ALTCLK_SEL	Selection for alt_clk_in and independent diffIO refclk
CH3_CLKREQI	Open drain bidirectional CLKREQ# input
CH2_CLKREQI	Open drain bidirectional CLKREQ# input
CH1_CLKREQI	Open drain bidirectional CLKREQ# input
CH0_CLKREQI	Open drain bidirectional CLKREQ# input
TX_LALIGN_IN_UP[7:0]	The connection signals between Quads for lane alignment across Quad boundary
TX_LALIGN_IN_DOWN[7:0]	The connection signals between Quads for lane alignment across Quad boundary
CH3_PIPE_POWER_DOWN[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_POWER_DOWN[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_POWER_DOWN[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_POWER_DOWN[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RATE[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RATE[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RATE[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RATE[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TXDETECTRX	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_EQEVAL	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_INVALID_REQUEST	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_HINTEN	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_PRESET_HINT[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_PRESET_HINT[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_HINT[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_HINT[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_POL	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TXCOMM_DIS	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEI_DIS	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1PMSSEN	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1X_ENTRY_REQ	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_SWING	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_MARGIN[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_MARGIN[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_MARGIN[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_MARGIN[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DEEMPH[15:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_DEEMPH[15:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DEEMPH[15:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DEEMPH[15:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DEEMPH_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_DEEMPH_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DEEMPH_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DEEMPH_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_PRESET_COEF	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATA_VLD	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_EI	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DATA_PCS_TX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_TX_DATA_PCS_TX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_PCS_TX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATA_PCS_TX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DATAK_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_DATAK_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATAK_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATAK_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_COMP_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_START_BLK_PCS_TX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_SYNC_HDR_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_SYNC_HDR_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_SYNC_HDR_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_SYNC_HDR_PCS_TX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_BLKALGNCTRL	Pipe signal open for user, mpcs and ecps sharing
LINK1_CSR_PIPE_RESET	Register control to reset lane3 in PCIe mode for link1
CH3_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
CH2_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
CH1_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
CH0_LINK0_CSR_PIPE_RESET	Register control to reset lane0-3 in PCIe mode for link0
PCIELL_MODESEL[2:0]	0=x4 mode, 1=x2 mode, 2=x1 mode, 3=x2/x1 mode, 4=x1/x1 mode
CH3_PIPE_POWER_DOWN_LL[1:0]	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_POWER_DOWN_LL[1:0]	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
CH1_PIPE_POWER_DOWN_LL[1:0]	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
CH0_PIPE_POWER_DOWN_LL[1:0]	PHY Power Down control. 00==P0 (normal operation), 01==P0s (Tx L0s), 10==P1 (L1 or Reset), 11==P2 (L2)
CH3_PIPE_RATE_LL[1:0]	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH2_PIPE_RATE_LL[1:0]	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH1_PIPE_RATE_LL[1:0]	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH0_PIPE_RATE_LL[1:0]	PHY Rate Control. 11==16G, 10==8G, 01==5G, 00==2.5G. Must be set to speed supported by the core
CH3_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH2_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH1_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH0_PIPE_TX_DETECT_RX_LOOPBACK_LL	If pipe_power_down==P1, 1==PHY Tx Detection Request. If pipe_tx_elec_idle==0, 1==PHY Rx to Tx Loopback Request. 0==Otherwise.
CH3_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
CH2_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
CH1_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
CH0_PIPE_RX_EQ_EVAL_LL	Set and hold pipe_rx_eq_eval==1 to request the PHY to evaluate the far end transmitter Tx Eq settings
CH3_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal
CH1_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal
CH0_PIPE_INVALID_REQUEST_LL	Rx Equalization Invalid Request: This signal is a per-lane signal which is used by the MAC layer to report to the PHY that the link partner has indicated an invalid request (transmit coefficient that the link partner is not able to apply). When this signal is asserted, the PHY will stop current evaluation and will provide a new direction through pipe linkevalchange[5:0] signal
CH3_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommends not to use.
CH2_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommends not to use.
CH1_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommends not to use.
CH0_PIPE_RX_PRESET_HINT_ENABLE_LL	Local PHY Rx Preset Hint Load Enable. Load pipe_rx_preset_hint into local PHY receiver when pipe_rx_preset_hint_enable == 1. Optional. PIPE Spec. recommends not to use.
CH3_PIPE_RX_PRESET_HINT_LL[2:0]	Local PHY Rx Preset Hint
CH2_PIPE_RX_PRESET_HINT_LL[2:0]	Local PHY Rx Preset Hint
CH1_PIPE_RX_PRESET_HINT_LL[2:0]	Local PHY Rx Preset Hint
CH0_PIPE_RX_PRESET_HINT_LL[2:0]	Local PHY Rx Preset Hint
CH3_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing)

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing)
CH1_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing)
CH0_PIPE_RX_POLARITY_LL	Receive Polarity: This signal is a per-lane signal which is generated by each link PCI-Express controller. As all per-lane signal the application is responsible for generating the association of link to lane and lane to link(multiplexing)
CH3_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH2_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH1_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH0_PIPE_TX_CLKREQ_N_LL	Open drain CLKREQ# pin output. 1==Drive CLKREQ# pin High-Z. 0==Drive CLKREQ# pin low to enable PCIe reference clock
CH3_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
CH2_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
CH1_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
CH0_PIPE_TX_CM_DISABLE_LL	TX Common Mode Disable. 1==PHY is permitted to disable TX Common Mode management. 0==Otherwise
CH3_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH2_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH1_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH0_PIPE_RX_EI_DISABLE_LL	RX Electrical Idle Disable. 1==PHY is permitted to disable Electrical Idle detection. 0==Otherwise
CH3_PIPE_L1PMSSEN_LL	L1 substates enable
CH2_PIPE_L1PMSSEN_LL	L1 substates enable
CH1_PIPE_L1PMSSEN_LL	L1 substates enable

Table 535: Input Ports (Continued)

Name	Description
CH0_PIPE_L1PMSSEN_LL	L1 substates enable
CH3_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH2_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH1_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH0_PIPE_TX_SWING_LL	2.5G/5G Transmitter Amplitude: 1==Reduced Swing. 0==Full Swing.
CH3_PIPE_TX_MARGIN_LL[2:0]	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH2_PIPE_TX_MARGIN_LL[2:0]	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH1_PIPE_TX_MARGIN_LL[2:0]	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH0_PIPE_TX_MARGIN_LL[2:0]	Transmitter Test Amplitude Selection. Used for test only. 000==Normal operation
CH3_PIPE_TX_DEEMPH_LL[17:0]	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH2_PIPE_TX_DEEMPH_LL[17:0]	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH1_PIPE_TX_DEEMPH_LL[17:0]	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH0_PIPE_TX_DEEMPH_LL[17:0]	Per lane Transmitter De-Emphasis. 2.5G:(1==-3.5dB). 5G:(1==-3.5dB,0==-6dB). 8G:([17:12]==Post-Cursor,[11:6]==Cursor,[5:0]==Pre-Cursor)
CH3_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
CH2_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
CH1_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis

Table 535: Input Ports (Continued)

Name	Description
CH0_PIPE_TX_DEEMPH_WINDOW_LL	1==In a window of time where phy_tx_deemph may change (such as in Recovery.Speed, Loopback, and Polling.Compliance). 0==Otherwise. Optional. May be useful for PHY which use a protocol interface for controlling the TX Deemphasis
CH3_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH2_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH1_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH0_PIPE_LOCAL_GET_PRESET_COEF_LL	Preset to Coefficient Mapping Request/Acknowledgement for >= 8G
CH3_PIPE_LOCAL_GET_PRESET_INDEX_LL[3:0]	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH2_PIPE_LOCAL_GET_PRESET_INDEX_LL[3:0]	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH1_PIPE_LOCAL_GET_PRESET_INDEX_LL[3:0]	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH0_PIPE_LOCAL_GET_PRESET_INDEX_LL[3:0]	Local Preset Index: This signal is a per-lane signal which is used by the MAC layer to indicate which coefficient the MAC ask to get the coefficient mapping. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted.
CH3_PIPE_REMOTE_FS_LL[5:0]	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH2_PIPE_REMOTE_FS_LL[5:0]	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH1_PIPE_REMOTE_FS_LL[5:0]	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH0_PIPE_REMOTE_FS_LL[5:0]	Link partner: FS value the link partner advertised at the beginning of Equalization training
CH3_PIPE_REMOTE_LF_LL[5:0]	Link partner: LF value the link partner advertised at the beginning of Equalization training
CH2_PIPE_REMOTE_LF_LL[5:0]	Link partner: LF value the link partner advertised at the beginning of Equalization training

Table 535: Input Ports (Continued)

Name	Description
CH1_PIPE_REMOTE_LF_LL[5:0]	Link partner: LF value the link partner advertised at the beginning of Equalization training
CH0_PIPE_REMOTE_LF_LL[5:0]	Link partner: LF value the link partner advertised at the beginning of Equalization training
CH3_PIPE_REMOTE_EQ_RX_DEEMPH_LL[17:0]	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH2_PIPE_REMOTE_EQ_RX_DEEMPH_LL[17:0]	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH1_PIPE_REMOTE_EQ_RX_DEEMPH_LL[17:0]	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH0_PIPE_REMOTE_EQ_RX_DEEMPH_LL[17:0]	Link partner: 8G deemphasis value we requested the link partner use during Equalization training
CH3_PIPE_REMOTE_EQ_RX_PRESET_LL[3:0]	Link partner: 8G preset value we requested the link partner use during Equalization training
CH2_PIPE_REMOTE_EQ_RX_PRESET_LL[3:0]	Link partner: 8G preset value we requested the link partner use during Equalization training
CH1_PIPE_REMOTE_EQ_RX_PRESET_LL[3:0]	Link partner: 8G preset value we requested the link partner use during Equalization training
CH0_PIPE_REMOTE_EQ_RX_PRESET_LL[3:0]	Link partner: 8G preset value we requested the link partner use during Equalization training
CH3_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH2_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH1_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH0_PIPE_TX_DATA_VALID_LL	PIPE Tx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH3_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH2_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH1_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH0_PIPE_TX_START_BLOCK_LL	PIPE Tx Start Block. 1==Start of block. 0==Otherwise. All blocks transmitted are 128-bit
CH3_PIPE_TX_SYNC_HEADER_LL[1:0]	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data

Table 535: Input Ports (Continued)

Name	Description
CH2_PIPE_TX_SYNC_HEADER_LL[1:0]	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH1_PIPE_TX_SYNC_HEADER_LL[1:0]	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH0_PIPE_TX_SYNC_HEADER_LL[1:0]	PIPE Tx Sync Header. When pipe_tx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH3_PIPE_TX_DATA_LL[31:0]	PIPE Tx Data
CH2_PIPE_TX_DATA_LL[31:0]	PIPE Tx Data
CH1_PIPE_TX_DATA_LL[31:0]	PIPE Tx Data
CH0_PIPE_TX_DATA_LL[31:0]	PIPE Tx Data
CH3_PIPE_TX_DATAK_LL[3:0]	PIPE Tx Data K indicator for 8b10b
CH2_PIPE_TX_DATAK_LL[3:0]	PIPE Tx Data K indicator for 8b10b
CH1_PIPE_TX_DATAK_LL[3:0]	PIPE Tx Data K indicator for 8b10b
CH0_PIPE_TX_DATAK_LL[3:0]	PIPE Tx Data K indicator for 8b10b
CH3_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH2_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH1_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH0_PIPE_TX_COMPLIANCE_LL	PIPE Tx Compliance. 1==Force disparity to negative before encoding pipe_tx_data[7:0] (for 8b10b compliance pattern). 0==Otherwise
CH3_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH2_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH1_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH0_PIPE_TX_ELEC_IDLE_LL	PIPE Tx Electrical Idle. 1==Idle, 0==Active
CH3_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
CH2_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
CH1_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1
CH0_PIPE_BLOCK_ALIGN_CONTROL_LL	PIPE Block Align Control. Set to 0 when within a datastream (except for Slave Loopback); otherwise set to 1

Table 535: Input Ports (Continued)

Name	Description
CH3_PIPE_PRESET_INDX_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_PRESET_INDX_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_PRESET_INDX_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_PRESET_INDX_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_REMOTE_FS_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_REMOTE_FS_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_REMOTE_FS_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_REMOTE_FS_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_REMOTE_LF_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_REMOTE_LF_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_REMOTE_LF_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_REMOTE_LF_PCS_TX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_DEEMPH_PCS_TX_DATA[17:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_DEEMPH_PCS_TX_DATA[17:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DEEMPH_PCS_TX_DATA[17:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DEEMPH_PCS_TX_DATA[17:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_PRESET_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_PRESET_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_PRESET_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_PRESET_PCS_TX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
ACTMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH3_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTEN	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH3_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

Table 535: Input Ports (Continued)

Name	Description
CH1_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTACMD	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH3_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTDR11	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH3_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACTHIGHZ	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
SCAN_CLK	Scan clock for PCS
SCAN_RST	Scan reset for PCS

Table 536: Output Ports

Name	Description
CH3_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH2_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH1_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH0_PIPE_PCS_TXCLKOUT	Pipe and PCS tx clock to Fabric
CH3_RXOUTCLK	PCS RX clock to Fabric
CH2_RXOUTCLK	PCS RX clock to Fabric
CH1_RXOUTCLK	PCS RX clock to Fabric
CH0_RXOUTCLK	PCS RX clock to Fabric
TX_ALIGN_CLKOUT	Clock for multi quad usage in TX mode
RX_ALIGN_CLKOUT	Clock for multi quad usage in RX mode
CH3_LMMIRDATA_VALID	LMMI read data valid
CH2_LMMIRDATA_VALID	LMMI read data valid

Table 536: Output Ports (Continued)

Name	Description
CH1_LMMIRDATA_VALID	LMMI read data valid
CH0_LMMIRDATA_VALID	LMMI read data valid
CH3_LMMIREADY	LMMI Bus ready
CH2_LMMIREADY	LMMI Bus ready
CH1_LMMIREADY	LMMI Bus ready
CH0_LMMIREADY	LMMI Bus ready
CH3_LMMIRDATA[7:0]	LMMI read data
CH2_LMMIRDATA[7:0]	LMMI read data
CH1_LMMIRDATA[7:0]	LMMI read data
CH0_LMMIRDATA[7:0]	LMMI read data
CH3_SDTXP	Analog TX diffIO
CH2_SDTXP	Analog TX diffIO
CH1_SDTXP	Analog TX diffIO
CH0_SDTXP	Analog TX diffIO
CH3_SDTXN	Analog TX diffIO
CH2_SDTXN	Analog TX diffIO
CH1_SDTXN	Analog TX diffIO
CH0_SDTXN	Analog TX diffIO
CH3_PCLKAUX_SEL_LL	Pipe clock selection for user
CH2_PCLKAUX_SEL_LL	Pipe clock selection for user
CH1_PCLKAUX_SEL_LL	Pipe clock selection for user
CH0_PCLKAUX_SEL_LL	Pipe clock selection for user
CH3_CLKREQO	Open drain bidirectional CLKREQ# output
CH2_CLKREQO	Open drain bidirectional CLKREQ# output
CH1_CLKREQO	Open drain bidirectional CLKREQ# output
CH0_CLKREQO	Open drain bidirectional CLKREQ# output
CH3_CLKREQOE	Open drain bidirectional CLKREQ# output enable
CH2_CLKREQOE	Open drain bidirectional CLKREQ# output enable
CH1_CLKREQOE	Open drain bidirectional CLKREQ# output enable
CH0_CLKREQOE	Open drain bidirectional CLKREQ# output enable
TX_LALIGN_OUT_UP[7:0]	The connection signals between Quads for lane alignment across Quad boundary

Table 536: Output Ports (Continued)

Name	Description
TX_LALIGN_OUT_DOWN[7:0]	The connection signals between Quads for lane alignment across Quad boundary
CH3_PIPE_WIDTH_5G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_WIDTH_5G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_WIDTH_5G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_5G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_WIDTH_8G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_WIDTH_8G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_WIDTH_8G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_8G0[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RXEQ_FOM[7:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RXEQ_FOM[7:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEQ_FOM[7:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_FOM[7:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_PHY_STATUS	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_EI	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_CLKREQ	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_L1X_ENTRY_ACK	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_LOCAL_GET_TX_PRESET_COEF[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_LOCAL_GET_TX_PRESET_COEF[3:0]	Pipe signal open for user, mpcs and ecps sharing

Table 536: Output Ports (Continued)

Name	Description
CH1_PIPE_LOCAL_GET_TX_PRESET_COEF[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_TX_PRESET_COEF[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_PRESET_COEF_PCS_RX_DATA[13:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_PRESET_COEF_PCS_RX_DATA[13:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_PRESET_COEF_PCS_RX_DATA[13:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_PRESET_COEF_PCS_RX_DATA[13:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_START_BLOCK	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_DATA_PCS_RX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_DATA_PCS_RX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_PCS_RX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_PCS_RX_DATA[31:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_STATUS_PCS_RX_DATA[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_STATUS_PCS_RX_DATA[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_STATUS_PCS_RX_DATA[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_STATUS_PCS_RX_DATA[2:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_EN_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATA_VLD_PCS_RX_DATA	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_SYNC_HDR_PCS_RX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_SYNC_HDR_PCS_RX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing

Table 536: Output Ports (Continued)

Name	Description
CH1_PIPE_RX_SYNC_HDR_PCS_RX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_SYNC_HDR_PCS_RX_DATA[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_WIDTH_2G5_LL[1:0]	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH2_PIPE_WIDTH_2G5_LL[1:0]	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH1_PIPE_WIDTH_2G5_LL[1:0]	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH0_PIPE_WIDTH_2G5_LL[1:0]	Width of pipe_tx/rx_data at 2.5G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH3_PIPE_WIDTH_5G0_LL[1:0]	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH2_PIPE_WIDTH_5G0_LL[1:0]	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH1_PIPE_WIDTH_5G0_LL[1:0]	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH0_PIPE_WIDTH_5G0_LL[1:0]	Width of pipe_tx/rx_data at 5.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH3_PIPE_WIDTH_8G0_LL[1:0]	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH2_PIPE_WIDTH_8G0_LL[1:0]	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH1_PIPE_WIDTH_8G0_LL[1:0]	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH

Table 536: Output Ports (Continued)

Name	Description
CH0_PIPE_WIDTH_8G0_LL[1:0]	Width of pipe_tx/rx_data at 8.0G speed. 11==Reserved, 10==[31:0], 01==[15:0], 00=[7:0]. Must be <= PHY_DATA_WIDTH. 2-bits only for >= 32-bit PHY_DATA_WIDTH
CH3_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL[7:0]	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH2_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL[7:0]	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH1_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL[7:0]	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH0_PIPE_RX_EQ_EVAL_FEEDBACK_FOM_LL[7:0]	PHY provides the relative received signal quality on pipe_rx_eq_eval_feedback_fom coindident with pipe_phy_status == 1.
CH3_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL[5:0]	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH2_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL[5:0]	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH1_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL[5:0]	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH0_PIPE_RX_EQ_EVAL_FEEDBACK_DIR_LL[5:0]	PHY provides pre/post coefficient up/dn guidance on pipe_rx_eq_eval_feedback_dir coindident with pipe_phy_status == 1
CH3_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH2_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH1_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH0_PIPE_PHY_STATUS_LL	Set to 1 by PHY for one clock to communicate Tx detection, rate change, power down change, and equalization evaluation completion
CH3_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide

Table 536: Output Ports (Continued)

Name	Description
CH2_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
CH1_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
CH0_PIPE_RX_ELEC_IDLE_LL	PIPE Rx Electrical Idle. 1==Idle, 0==Active. Required valid at 2.5G. Allowance for inaccuracy at >= 5G. See core User Guide
CH3_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
CH2_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
CH1_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
CH0_PIPE_RX_CLKREQ_N_LL	Open drain CLKREQ# pin input. Current value of open drain CLKREQ# pin. 1==Request to disable PCIe reference clock. 0==Request to enable PCIe reference clock
CH3_PIPE_LOCAL_FS_LL[5:0]	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH2_PIPE_LOCAL_FS_LL[5:0]	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH1_PIPE_LOCAL_FS_LL[5:0]	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH0_PIPE_LOCAL_FS_LL[5:0]	Local PHY FS value. Captured on first pipe_phy_status == 1 after reset deassertion
CH3_PIPE_LOCAL_LF_LL[5:0]	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
CH2_PIPE_LOCAL_LF_LL[5:0]	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
CH1_PIPE_LOCAL_LF_LL[5:0]	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion
CH0_PIPE_LOCAL_LF_LL[5:0]	Local PHY LF value. Captured on first pipe_phy_status == 1 after reset deassertion

Table 536: Output Ports (Continued)

Name	Description
CH3_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted
CH2_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted
CH1_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted
CH0_PIPE_LOCAL_GET_TX_COEF_VALID_LL	Local Tx coefficient valid: This signal is a per-lane signal which is used by the PHY layer to indicate the required preset to coefficient mapping is valid on pipe_local_tx_pset_coef[17:0] output signal. This signal is used by the PHY layer only when the pipe_getlocal_pset_coef signal is asserted
CH3_PIPE_LOCAL_GET_TX_PRESET_COEF_LL[17:0]	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH2_PIPE_LOCAL_GET_TX_PRESET_COEF_LL[17:0]	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH1_PIPE_LOCAL_GET_TX_PRESET_COEF_LL[17:0]	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH0_PIPE_LOCAL_GET_TX_PRESET_COEF_LL[17:0]	Local Tx coefficient mapping: This signal is a per-lane signal which is used by the PHY layer to indicate the coefficient value corresponding to the required preset
CH3_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0
CH2_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0
CH1_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0

Table 536: Output Ports (Continued)

Name	Description
CH0_PIPE_TX_DATA_ENABLE_LL	PIPE Tx Data Enable. Tx Data is taken by PHY when pipe_tx_data_enable==1. Tx Data outputs hold their value when pipe_tx_data_enable==0
CH3_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH2_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH1_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH0_PIPE_RX_DATAEN_LL	PIPE Rx Data Enable. Data is provided by PHY when pipe_rx_data_enable==1. Rx Data inputs are unused when pipe_rx_data_enable==0
CH3_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH2_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH1_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH0_PIPE_RX_DATA_VALID_LL	PIPE Rx Data Valid for >= 8G only. 1==Data is valid. 0==Data is invalid. De-asserted only to compensate for 128b130b encoding
CH3_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH2_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH1_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH0_PIPE_RX_START_BLOCK_LL	PIPE Rx Start Block. 1==Start of block. 0==Otherwise. All blocks received are 128-bit except SKP OS which is variable size
CH3_PIPE_RX_SYNC_HEADER_LL[1:0]	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH2_PIPE_RX_SYNC_HEADER_LL[1:0]	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data

Table 536: Output Ports (Continued)

Name	Description
CH1_PIPE_RX_SYNC_HEADER_LL[1:0]	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH0_PIPE_RX_SYNC_HEADER_LL[1:0]	PIPE Rx Sync Header. When pipe_rx_start_block==1, indicates the block type: 01==Ordered Set or 10==Data
CH3_PIPE_RX_DATA_LL[31:0]	Pipe RX data
CH2_PIPE_RX_DATA_LL[31:0]	Pipe RX data
CH1_PIPE_RX_DATA_LL[31:0]	Pipe RX data
CH0_PIPE_RX_DATA_LL[31:0]	Pipe RX data
CH3_PIPE_RX_DATAK_LL[3:0]	PIPE Rx Data K indicator for 8b10b.
CH2_PIPE_RX_DATAK_LL[3:0]	PIPE Rx Data K indicator for 8b10b.
CH1_PIPE_RX_DATAK_LL[3:0]	PIPE Rx Data K indicator for 8b10b.
CH0_PIPE_RX_DATAK_LL[3:0]	PIPE Rx Data K indicator for 8b10b.
CH3_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH2_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH1_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH0_PIPE_RX_VALID_LL	1==PHY is locked to Rx data stream and data is valid - use pipe_rx_data. 0==PHY is not locked - ignore pipe_rx_data
CH3_PIPE_RX_STATUS_LL[2:0]	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
CH2_PIPE_RX_STATUS_LL[2:0]	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
CH1_PIPE_RX_STATUS_LL[2:0]	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
CH0_PIPE_RX_STATUS_LL[2:0]	PIPE Rx Status. 000=OK, 001/010 SKP Add/Del (unused), 011=Receiver Detected, 100/111 encoding error, 101/110 over/underflow error
PIPE_PCLKOUT_LL[3:0]	Pipe clock connect to harden PCIe_LL. connect to pipe_pclkout_LL[0] goes to clk_250_i_link0, and pipe_pclkout_LL[3] goes to clk_250_i_link1
CH3_PIPE_WIDTH_2G5[1:0]	Pipe signal open for user, mpcs and ecps sharing

Table 536: Output Ports (Continued)

Name	Description
CH2_PIPE_WIDTH_2G5[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_WIDTH_2G5[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_WIDTH_2G5[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RXEQ_DIR_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RXEQ_DIR_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEQ_DIR_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_DIR_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RXEQ_DIR_PCS_RX_FIFO_ST[1:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_LOCAL_FS_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_LOCAL_FS_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_FS_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_FS_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_LOCAL_LF_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_LOCAL_LF_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_LF_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_LF_PCS_RX_DATA[5:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_LOCAL_GET_TX_COEF_VLD	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_TX_DATA_EN	Pipe signal open for user, mpcs and ecps sharing
CH3_PIPE_RX_DATAK_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH2_PIPE_RX_DATAK_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH1_PIPE_RX_DATAK_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH0_PIPE_RX_DATAK_PCS_RX_DATA[3:0]	Pipe signal open for user, mpcs and ecps sharing
CH3_PCS_RX_DATA[5:0]	Dedicated ports used for pcs rx data

Table 536: Output Ports (Continued)

Name	Description
CH2_PCS_RX_DATA[5:0]	Dedicated ports used for pcs rx data
CH1_PCS_RX_DATA[5:0]	Dedicated ports used for pcs rx data
CH0_PCS_RX_DATA[5:0]	Dedicated ports used for pcs rx data
CH3_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH2_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH1_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH0_PCS_RX_HI_BER	The high level of this signal indicates High BER is detected
CH3_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH2_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH1_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH0_PCS_RX_BLK_LOCK	The high level of this signal indicates the block lock is achieved
CH3_PCS_TX_FIFO_ST[1:0]	TX fifo status (empty, full)
CH2_PCS_TX_FIFO_ST[1:0]	TX fifo status (empty, full)
CH1_PCS_TX_FIFO_ST[1:0]	TX fifo status (empty, full)
CH0_PCS_TX_FIFO_ST[1:0]	TX fifo status (empty, full)
CH3_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACJPOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH3_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH2_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH1_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly
CH0_ACJNOUT	Jtag mode for PMA. When PMA in jtag mode, user can force data and ctrl to PMA though JTAG signal directly

PDP4K

4 kb Pseudo-dual Port Block RAM

Architectures supported:

- iCE40 UltraPlus

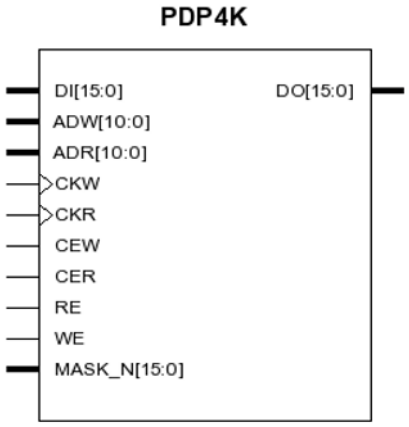


Table 537: Input Ports

Name	Range	Description
DI	15:0	Data in
ADW	10:0	Write address
ADR	10:0	Read address
CKW		Write clock
CKR		Read clock
CEW		Write clock enable, active high
CER		Read clock enable, active high
RE		Read enable, active high
WE		Write enable, active high
MASK_N	15:0	Per-bit write enable mask, active low

Note:

MASK_N input is functional only for the 16-bit mode. PDP4K is used to configure 2-, 4-, 8- or 16-bit mode EBR operations. MASK_N functions only when it is configured to 16-bit data.

Table 538: Output Ports

Name	Range	Description
DO	15:0	Data output

Table 539: Parameters

Name	Description
INITVAL_0	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_1	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_2	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_3	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_4	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_5	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_6	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_7	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_8	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_9	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_A	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_B	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_C	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_D	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_E	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)
INITVAL_F	"0x0000000000000000000000000000000000000000000000000000000000000000000000" (default)

Table 539: Parameters (Continued)

Name	Description
DATA_WIDTH_W	"2" (default), "4", "8", "16" (Write data width)
DATA_WIDTH_R	"2" (default), "4", "8", "16" (Read data width)

Rules & Restrictions:

See EBR_B section. The same rules regard port mapping and initial value parameter conversion for EBR_B apply for PDP4K (only difference is that for EBR_B ⟶ INIT_#, while for PDP4K ⟶ INITVAL_#).

PDP16K

16Kb Pseudo Dual Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

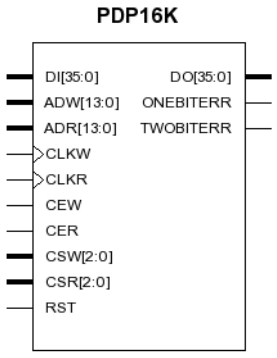


Table 540: Input Ports

Name	Range	Description
DI	35:0	Data in
ADW	13:0	Write address
ADR	13:0	Read address
CLKW		Write clock
CLKR		Read clock
CEW		Clock enable
CER		Clock enable
CSW	2:0	Write chip select
CSR	2:0	Read chip select
RST		Output register reset

Table 541: Output Ports

Name	Range	Description
DO	35:0	Data out
ONEBITERR		ECC error flag (single bit error
TWOBITERR		ECC error flag (two bit error)

Table 542: Parameters

Name	Values	Description
DATA_WIDTH_W	"X36" *(default)* "X32" "X18" "X9" "X4" "X2" "X1"	Write port width
DATA_WIDTH_R	"X36" *(default)* "X32" "X18" "X9" "X4" "X2" "X1"	Read port width
OUTREG	"BYPASSED" *(default)* "USED"	Register output
RESETMODE	"SYNC" *(default)* "ASYNC"	Reset sync/async control
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset for the output registers
ECC	"DISABLED" *(default)* "ENABLED"	Enable ECC
INITVAL_00...INITVAL_3F	Hex string, 320 bits, must include the "0x" prefix	EBR initialization data
CSDECODE_W	"000" *(default)* "001" "010" "011" "100" "101" "110" "111"	Write chip select active level, 0=active high 1=active low
CSDECODE_R	"000" *(default)* "001" "010" "011" "100" "101" "110" "111"	Read chip select active setting, 0=active high 1=active low

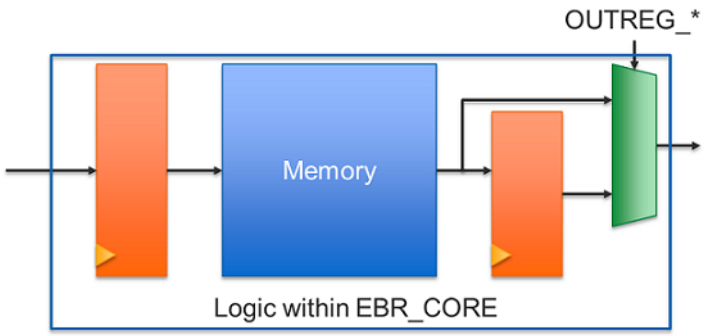
Table 542: Parameters (Continued)

Name	Values	Description
ASYNC_RST_RELEASE	"SYNC" *(default)* "ASYNC"	Reset release sync/async control
INIT_DATA	"STATIC" *(default)* "DYNAMIC" "NO_INIT"	

Note:

It is **not** legal to drive the read and write clock such that their active edges switch simultaneously.

All inputs (except the register control signals) and all outputs are registered in the following manner:



Port assignments for different data widths will follow the table below:

Data Width	Input Data	Output Data	Write Address (MSB to LSB)	Read Address (MSB to LSB)
16Kx1	DI[0]	DO[0]	ADW[13:0]	ADR[13:0]
8Kx2	DI[1:0]	DO[1:0]	ADW[13:1]	ADR[13:1]
4Kx4	DI[3:0]	DO[3:0]	ADW[13:2]	ADR[13:2]
2Kx9	DI[8:0]	DO[8:0]	ADW[13:3]	ADR[13:3]
1Kx18	DI[17:0]	DO[17:0]	ADW[13:4]	ADR[13:4]
512x36	DI[35:0]	DO[35:0]	ADW[13:4]	ADR[13:4]

For data widths that do not use all available address bits, the unused bits **must** be tied high for proper functionality.

PDP32K

32 Kb pseudo-dual port block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

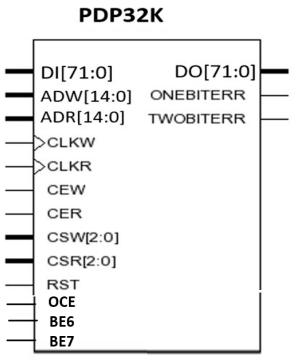


Table 543: Input Ports

Name	Range	Type	Active State	Description
DI	71:0	Data	Active High	Data Inputs
ADW	14:0	Addr	Active High	Write Address
ADR	14:0	Addr	Active High	Read Address
CLKW	1	Clock	Rising Edge	Write Clock Input
CLKR	1	Clock	Rising Edge	Read Clock Input
CEW	1	Clock Enable	Active High	Write Clock Enable
CER	1	Clock Enable	Active High	Read Clock Enable
OCE	1	Clock Enable	Active High	Output Register Clock Enable
BE6	1	Control	Active High	Byte enable for x72/x64 highest-1 byte
BE7	1	Control	Active High	Byte enable for x72/x64 highest byte
CSW	2:0	Control	Active High	Write Cascade Select
CSR	2:0	Control	Active High	Read Cascade Select
RST	1	Reset	Active High	Read output registers reset

Table 544: Output Ports

Name	Range	Type	Description
DO	71:0	Data	Data Output
ONEBITERR	1	Flag	ECC error flag (single bit error)
TWOBITERR	1	Flag	ECC error flag (two bit error)

Table 545: Parameters

Name	Values	Description
DATA_WIDTH_W	"X72" (default) "X64" (virtual for ECC) "X36" "X18" "X9" "X4" "X2" "X1"	
DATA_WIDTH_R	"X72" (default) "X64" "X36" "X18" "X9" "X4" "X2" "X1"	Read port width
OUTREG	"BYPASSED" (default) "USED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Reset sync/async control
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers

Table 545: Parameters (Continued)

Name	Values	Description
CSDECODE_W	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Write chip select active level, 0=active high 1=active low
CSDECODE_R	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Read chip select active setting, 0=active high 1=active low
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release sync/async control
ECC	"DISABLED" (default) "ENABLED"	Enable ECC
INITVAL_00...INITVAL_7F	Hex string, 320 bits, must include the "0x" prefix	Initialization value
INIT_DATA	"STATIC" (default) "DYNAMIC"	Specifies if the initialization data (as defined by the INITVAL parameters) may be changed after the synthesis stage. When INIT_DATA=STATIC, it is assumed that the initialization data will not change after synthesis, so an optimization to reduce the bitstream size can be applied. When INIT_DATA=DYNAMIC it is assumed that the initialization data may change after synthesis, so the aforementioned optimization is disabled.

Note:

If writing and reading at the same address simultaneously, the behavior is not guaranteed (i.e. either the read or write may happen first).

PDPSC16K

16Kb Pseudo Dual Port Single Clock Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

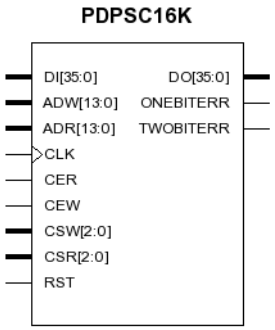


Table 546: Input Ports

Name	Range	Description
DI	35:0	Data in
ADW	13:0	Write address
ADR	13:0	Read address
CLK		Clock
CER		Read clock enable
CEW		Write clock enable

Table 546: Input Ports (Continued)

Name	Range	Description
CSW	2:0	Write chip select
CSR	2:0	Read chip select
RST		Output register reset

Table 547: Output Ports

Name	Range	Description
DO	35:0	Data out
ONEBITERR		ECC error flag (single bit error)
TWOBITERR		ECC error flag (two bit error)

Table 548: Parameters

Name	Values	Description
DATA_WIDTH_W	"X36" *(default)* "X18" "X9" "X4" "X2" "X1"	Write port width
DATA_WIDTH_R	"X36" *(default)* "X18" "X9" "X4" "X2" "X1"	Read port width
OUTREG	"BYPASSED" *(default)* "USED"	Register output
RESETMODE	"SYNC" *(default)* "ASYNC"	Reset sync/async control
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset for the output registers
ECC	"DISABLED" (default) "ENABLED"	Enable ECC
INITVAL_00...INITVAL_3F	Hex string, 320 bits, must include the "0x" prefix	EBR Initialization data

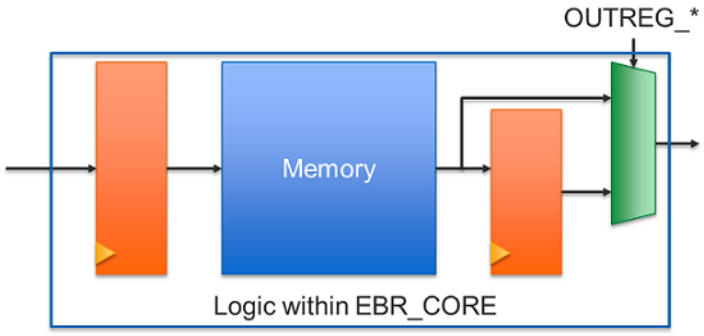
Table 548: Parameters

Name	Values	Description
CSDECODE_W	"000" *(default)* "001" "010" "011" "100" "101" "110" "111"	Write chip select active setting 0: active high 1: active low
CSDECODE_R	"000" *(default)* "001" "010" "011" "100" "101" "110" "111"	Read chip select active setting 0: active high 1: active low
ASYNC_RST_RELEASE	"SYNC" *(default)* "ASYNC"	Reset release sync/async control
INIT_DATA	"STATIC" (default) "DYNAMIC" "NO_INIT"	

Note:

When a pseudo-dual port RAM is desired, if only a single clock is required it is preferred to use PDPSC16K over PDP16K as performance in hardware will be faster with PDPSC16K.

All inputs (except the register control signals) and all outputs are registered in the following manner:



Port assignments for different data widths will follow the table below:

Data Width	Input Data	Output Data	Write Address (MSB to LSB)	Read Address (MSB to LSB)
16Kx1	DI[0]	DO[0]	ADW[13:0]	ADR[13:0]
8Kx2	DI[1:0]	DO[1:0]	ADW[13:1]	ADR[13:1]
4Kx4	DI[3:0]	DO[3:0]	ADW[13:2]	ADR[13:2]
2Kx9	DI[8:0]	DO[8:0]	ADW[13:3]	ADR[13:3]
1Kx18	DI[17:0]	DO[17:0]	ADW[13:4]	ADR[13:4]
512x36	DI[35:0]	DO[35:0]	ADW[13:4]	ADR[13:4]

For data widths that don't use all available address bits, the unused bits must be tied high for proper functionality.

PDPSC32K

32Kb Pseudo Dual Port Single Clock Block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

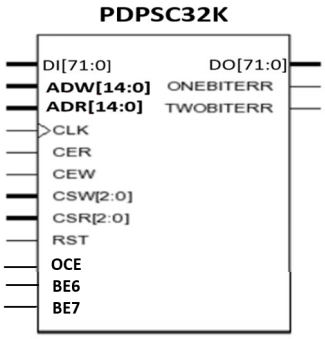


Table 549: Input Ports

Name	Range	Type	Active State	Description
DI	71:0	Data	Active High	Data Inputs
ADW	14:0	Addr	Active High	Write Address
ADR	14:0	Addr	Active High	Read Address
CLK	1	Clock	Rising Edge	Write & Read Clock Input

Table 549: (Continued)Input Ports

Name	Range	Type	Active State	Description
CEW	1	Clock Enable	Active High	Write Clock Enable
CER	1	Clock Enable	Active High	Read Clock Enable
OCE	1	Clock Enable	Active High	Output register Clock Enable
BE6	1	Control	Active High	Byte enable for x72/x64 highest-1 byte
BE7	1	Control	Active High	Byte enable for x72/x64 highest byte
CSW	2:0	Control	Active High	Write Cascade Select
CSR	2:0	Control	Active High	Read Cascade Select
RST	1	Reset	Active High	Read output registers reset

Table 550: Output Ports

Name	Range	Type	Description
DO	71:0	Data	Data Output
ONEBITERR	1	Flag	ECC error flag (single bit error)
TWOBITERR	1	Flag	ECC error flag (two bit error)

Table 551: Parameters

Name	Values	Description
DATA_WIDTH_W	"X72" (default) "X64" "X36" "X18" "X9" "X4" "X2" "X1"	Write port width
DATA_WIDTH_R	"X72" (default) "X64" "X36" "X18" "X9" "X4" "X2" "X1"	Read port width

Table 551: Parameters (Continued)

Name	Values	Description
OUTREG	"BYPASSED" (default) "USED"	Register output
RESETMODE	"SYNC" (default) "ASYNC"	Reset sync/async control
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
CSDECODE_W	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Write chip select active level, 0=active high 1=active low
CSDECODE_R	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Read chip select active setting, 0=active high 1=active low
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release sync/async control
ECC	"DISABLED" (default) "ENABLED"	Enable ECC

Table 551: Parameters (Continued)

Name	Values	Description
INITVAL_00...INITVAL_7F	Hex string, 320 bits, must include the "0x" prefix	Initialization value
INIT_DATA	"STATIC" (default) "DYNAMIC"	Specifies if the initialization data (as defined by the INITVAL parameters) may be changed after the synthesis stage. When INIT_DATA=STATIC, it is assumed that the initialization data will not change after synthesis, so an optimization to reduce the bitstream size can be applied. When INIT_DATA=DYNAMIC it is assumed that the initialization data may change after synthesis, so the aforementioned optimization is disabled.

Note:

If deterministic behavior is desired, you can use PDPSC32K. In this situation, the read occurs before the write.

PDPSC512K

512Kb Single Clock Pseudo Dual Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

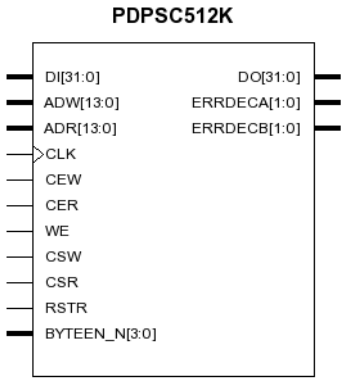


Table 552: Input Ports

Name	Range	Description
DI	31:0	Data in
ADW	13:0	Write address
ADR	13:0	Read address
CLK		Clock
CEW		Write clock enable
CER		Read clock enable
WE		Write enable
CSW		Write chip select
CSR		Read chip select
RSTR		Read register reset
BYTEEN_N	3:0	Write byte enable, 0=write, 1=disable. BYTEEN_N[3] corresponds to the MSB of DI, BYTEEN_N[0] corresponds to the LSB

Table 553: Output Ports

Name	Range	Description
DO	31:0	Data out

Table 553: Output Ports (Continued)

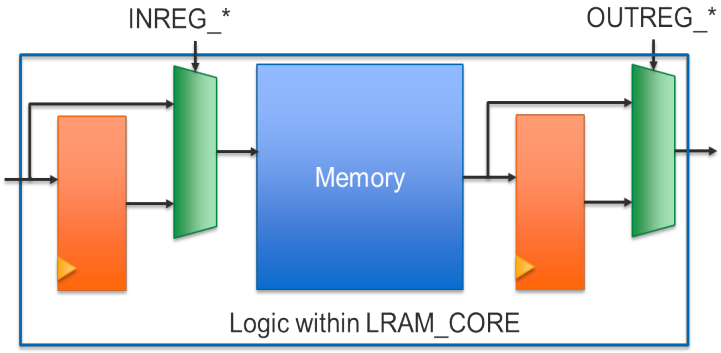
Name	Range	Description
ERRDECA	1:0	Port A one and two bit error flag
ERRDECB	1:0	Port B one and two bit error flag

Table 554: Parameters

Name	Values	Description
OUTREG	"NO_REG" *(default)* "OUT_REG"	Register output
GSR	"ENABLED" *(default)* "DISABLED"	Enable global set/reset
RESETMODE	"SYNC" *(default)* "ASYNC"	Reset sync/async control
INITVAL_00...INITVAL_7F	Hex string, 5120 bits, must include the "0x" prefix	LRAM initialization data
ASYNC_RESET_RELEASE	"SYNC" *(default)* "ASYNC"	Reset release sync/async
ECC_BYTE_SEL	"ECC_EN" (default) "BYTE_EN"	Enable ECC or Byte-enable support

Note:

All inputs (except the register control signals) and all outputs are registered in the following manner:



PLL

Wrapper for Phase Locked Loop

PLL is capable of frequency synthesis and clock phase management including clock injection delay cancellation.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

PLL

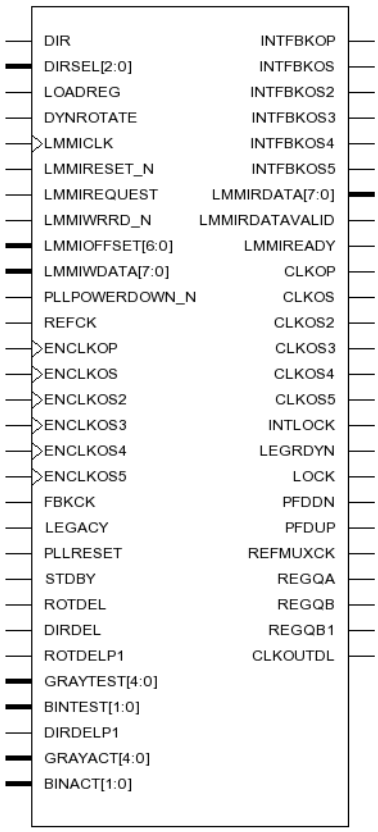


Table 555: Input Ports

Name	Range	Description
DIR		Valid if MC1_DYN_SOURCE = 1. Specifies direction that CIB_ROTATE changes VCO phase. 0 - phase rotates to later phase. 1 - Phase rotates to earlier phase.
DIRSEL	2:0	Valid if MC1_DYN_SOURCE = 1. CIB_DYN_SEL must be stable before initiating a rotation with CIB_LOAD_REG.
LOADREG		Valid if MC1_DYN_SOURCE = 1. Initiates a divider output phase shift on negative edge of CIB_LOAD_REG.
DYNROTATE		Valid if MC1_DYN_SOURCE = 1. Initiates a change from current VCO clock phase to an earlier or later phase on the negative edge of CIB_ROTATE.
LMMICLK		LMMI clock from fabric.
LMMIRESET_N		LMMI reset signal to reset the state machine if the IP gets locked.
LMMIREQUEST		LMMI request signal from fabric.
LMMIWRRD_N		LMMI write-high/read-low from fabric.
LMMIOFFSET	6:0	LMMI offset address from fabric. Not all bits are required for an IP.
LMMIWDATA	7:0	LMMI write data from fabric. Not all bits are required for an IP.
PLLPOWERDOWN_N		Active low to power down PLL from CIB port.
REFCK		
ENCLKOP		Enable A output (CLKOP). Active high. PLL CIB input.
ENCLKOS		Enable B output (CLKOS). Active high. PLL CIB input.
ENCLKOS2		Enable C output (CLKOS2). Active high. PLL CIB input.
ENCLKOS3		Enable D output (CLKOS3). Active high. PLL CIB input.
ENCLKOS4		Enable E output (CLKOS4). Active high. PLL CIB input.
ENCLKOS5		Enable F output (CLKOS5). Active high. PLL CIB input.
FBKCK		

Table 555: Input Ports (Continued)

Name	Range	Description
LEGACY		PLL legacy mode signal. Active high to enter the mode. Enabled by lmmi_legacy fuse. PLL CIB input.
PLLRESET		Active high to reset PLL. Enabled by MC1_PLLRESET. PLL CIB input.
STDBY		PLL standby signal. Active high to put PLL clocks in low. Not used.
ROTDEL		
DIRDEL		
ROTDELP1		
GRAYTEST	4:0	
BINTEST	1:0	
DIRDELP1		
GRAYACT	4:0	
BINACT	1:0	

Table 556: Output Ports

Name	Range	Description
INTFBKOP		
INTFBKOS		
INTFBKOS2		
INTFBKOS3		
INTFBKOS4		
INTFBKOS5		
LMMIRDATA	7:0	LMMI read data to fabric.
LMMIRDATAVALID		LMMI read data valid to fabric.
LMMIREADY		LMMI ready signal to fabric.
CLKOP		Primary (A) output clock.
CLKOS		Secondary (B) output clock.
CLKOS2		Secondary (C) output clock.

Table 556: Output Ports (Continued)

Name	Range	Description
CLKOS3		Secondary (D) output clock.
CLKOS4		Secondary (E) output clock.
CLKOS5		Secondary (F) output clock.
INTLOCK		PLL internal lock indicator. PLL CIB output.
LEGRDYN		PLL lock indicator. PLL CIB output.
LOCK		
PFDDN		PFD DN output signal to PLL CIB port.
PFDUP		PFD UP output signal to PLL CIB port.
REFMUXCK		Reference CLK mux output. PLL CIB output.
REGQA		
REGQB		
REGQB1		
CLKOUTDL		

Table 557: Parameters

Name	Values	Description
BW_CTL_BIAS	"0'b0101" *(default)* "0000"	Input control signal to tune the bias current of ppath cp, when the bit increase, the bias current increase. This current branch is combined with the Ivco_fb current.
CLKOP_TRIM	"0'b0000" *(default)*	CLKOP output trim control bits
CLKOS_TRIM	"0'b0000" *(default)*	CLKOS output trim control bits
CLKOS2_TRIM	"0'b0000" *(default)*	CLKOS2 output trim control bits
CLKOS3_TRIM	"0'b0000" *(default)*	CLKOS3 output trim control bits
CLKOS4_TRIM	"0'b0000" *(default)*	CLKOS4 output trim control bits
CLKOS5_TRIM	"0'b0000" *(default)*	CLKOS5 output trim control bits

Table 557: Parameters

Name	Values	Description
CRIPPLE	"5P" *(default)* "1P" "3P" "7P" "9P" "11P" "13P" "15P"	LPF cap control
CSET	"40P" *(default)* "8P" "12P" "16P" "20P" "24P" "28P" "32P" "36P" "44P" "48P" "52P" "56P" "60P" "64P" "68P"	LPF cap control
DELAY_CTRL	"200PS" *(default)* "300PS"	Control signal to adjust the delay of the PFD: default 1b0 = 200 ps; 1b1 = 300 ps.
DELA	"0" *(default)*	Output divider phase shift. Works with lmmi_diva.
DELB	"0" *(default)*	Output divider phase shift. Works with lmmi_divb.
DELC	"0" *(default)*	Output divider phase shift. Works with lmmi_divc.
DELD	"0" *(default)*	Output divider phase shift. Works with lmmi_divd.
DELE	"0" *(default)*	Output divider phase shift. Works swith lmmi_dive.
DELF	"0" *(default)*	Output divider phase shift. Works with lmmi_divf.
DIRECTION	"DISABLED" *(default)* "ENABLED"	LMMI equivalent of CIB direction signal
DIVA	"0" *(default)*	Output dividers for CLKOP
DIVB	"0" *(default)*	Output dividers for CLKOS
DIVC	"0" *(default)*	Output dividers for CLKOS2
DIVD	"0" *(default)*	Output dividers for CLKOS3
DIVE	"0" *(default)*	Output dividers for CLKOS4
DIVF	"0" *(default)*	Output dividers for CLKOS5
DYN_SEL	"0’b000" *(default)*	Dynamic phase selection
DYN_SOURCE	"STATIC" *(default)* "DYNAMIC"	1'b1 selects CIB signals for dynamic phase shift.

Table 557: Parameters

Name	Values	Description
ENCLK_CLKOP	"DISABLED" *(default)* "ENABLED"	Bit 0. Clock output enable. Active high.
ENCLK_CLKOS	"DISABLED" *(default)* "ENABLED"	Bit 1. Clock output enable. Active high.
ENCLK_CLKOS2	"DISABLED" *(default)* "ENABLED"	Bit 2. Clock output enable. Active high.
ENCLK_CLKOS3	"DISABLED" *(default)* "ENABLED"	Bit 3. Clock output enable. Active high.
ENCLK_CLKOS4	"DISABLED" *(default)* "ENABLED"	Bit 4. Clock output enable. Active high.
ENCLK_CLKOS5	"DISABLED" *(default)* "ENABLED"	Bit 5. Clock output enable. Active high.
ENABLE_SYNC	"DISABLED" *(default)* "ENABLED"	1'b1 enables output sync with CLKOP.
FAST_LOCK_EN	"ENABLED" *(default)* "DISABLED"	Enables signal for fast lock.
V2I_1V_EN	"DISABLED" *(default)* "ENABLED"	1V supply enable or disable
FBK_CUR_BLE	"0'b00000000" *(default)*	Bleeding current for PI to adjust the linearity
FBK_EDGE_SEL	"POSITIVE" *(default)* "NEGATIVE"	Select the positive or negative phase of PI output. 0: positive phase; 1: negative phase
FBK_IF_TIMING_CTL	"0'b00" *(default)*	Interface timing control for feedback divider
FBK_INTEGER_MODE	"DISABLED" *(default)* "ENABLED"	Enable the integer mode for feedback divider
FBK_MASK	"0'b00001000" *(default)* "00000000"	Minimum divider ratio control word for feedback divider. For example if n_pll[7:0] or mmd_dig[7:0] is less than lmmi_fbk_mask[7:0], then the MMD divider ratio is determined by lmmi_fbk_mask[7:0]. Otherwise the divider ratio will be determined by n_pll[7:0] or mmd_dig[7:0].
FBK_MMD_DIG	"8" *(default)* "0"	MMD divider ratio setting in integer mode
FBK_MMD_PULS_CTL	"0'b0000" *(default)*	Pulse width control for MMD output clock. If lmmi_fbk_mmd_puls_ctl[3:0]=4'b0110, it means that there's 6 VCO cycles in the MMD output clock.
FBK_MODE	"0'b00" *(default)*	Reserved floating control bits
FBK_PI_BYPASS	"NOT_BYPASSED" *(default)* "BYPASSED"	PI bypass control bit. 0; PI not bypass, 1; PI bypass; it should be connected to lmmi_ssc_pi_bypass
FBK_PI_RC	"0'b1100" *(default)* "0000"	RC time constant control in PI
FBK_PR_CC	"0'b0000" *(default)*	Current control for PI to adjust the linearity

Table 557: Parameters

Name	Values	Description
FBK_PR_IC	"0'b1000" *(default)* "0000"	Bias current control for PI
FBK_RSV	"0'b0000000000000000" *(default)*	Reserved control bit for feedback divider.
FLOAT_CP	"DISABLED" *(default)* "ENABLED"	Active high to tri-state the ICP output.
FLOCK_CTRL	"2X" *(default)* "1X" "4X" "8X"	2 bits control the fast lock period, 00 is 1x, 01 is 2x, 10 is 4x and 11 is 8x
FLOCK_EN	"ENABLED" *(default)* "DISABLED"	1b1 to enable fast lock; after char, default to enabled
FLOCK_SRC_SEL	"REFCLK" *(default)* "FBCLK"	Fast lock source selection; 0 is ref clock; 1 is feedback clock
FORCE_FILTER	"DISABLED" *(default)* "ENABLED"	force internal vctrl=analog pad
I_CTRL	"10UA" *(default)* "8P3UA" "14P9UA" "12P4UA" "19P8UA" "17P3UA" "24P8UA" "22P3UA"	Current tuning
IPI_CMP	"0'b1000" *(default)* "0000"	i-path CP compensate up/dn mismatch at process variation
IPI_CMPN	"0'b0011" *(default)* "0000"	Input control bits to compensate the i-path bias current
IPI_COMP_EN	"DISABLED" *(default)* "ENABLED"	Enable ipi_cmp
IPP_CTRL	"0'b1000" *(default)* "0000"	Input control signal to tune the bias current of ppath cp
IPP_SEL	"0'b1111" *(default)* "0000"	Input control signal to select which ppath cp is on, there are 4 branches at max
KP_VCO	"0'b00000" *(default)*	ICO gain controls
LDT_INT_LOCK_STICKY	"DISABLED" *(default)* "ENABLED"	Active high to have INT_LOCK_STICKY. Default to be 1b1. 1b0 is for PDE/DE purpose.
LDT_LOCK	"98304CYC" *(default)* "24576CYC" "6144CYC" "1536CYC"	Frequency lock-detector resolution sensitivity

Table 557: Parameters

Name	Values	Description
LDT_LOCK_SEL	"U_FREQ" *(default)* "UFREQ" "SPHASE" "SFREQ" "UFREQ_SPHASE" "U_PHASE" "S_FREQ" "U_FREQ_S_PHASE"	Lock-detector type select
LEGACY_ATT	"DISABLED" *(default)* "ENABLED"	Active HIGH; enable Legacy mode
LOAD_REG	"DISABLED" *(default)* "ENABLED"	For load control of divider phase control
OPENLOOP_EN	"DISABLED" *(default)* "ENABLED"	Open loop mode enable for mfg testing
PHIA	"0" *(default)*	Select VCO phase-shift (0..7) for A section
PHIB	"0" *(default)*	Select VCO phase-shift (0..7) for B section
PHIC	"0" *(default)*	Select VCO phase-shift (0..7) for C section
PHID	"0" *(default)*	Select VCO phase-shift (0..7) for D section
PHIE	"0" *(default)*	Select VCO phase-shift (0..7) for E section
PHIF	"0" *(default)*	Select VCO phase-shift (0..7) for F section
PLLPDN_EN	"DISABLED" *(default)* "ENABLED"	
PLLPD_N	"UNUSED" *(default)* "USED"	Active LOW PLL power-down; PLL is NOT USED
PLLRESET_ENA	"DISABLED" *(default)* "ENABLED"	Active HIGH; Enable PLLRESET CIB Signal
REF_INTEGER_MODE	"DISABLED" *(default)* "ENABLED"	Integer mode control bit for reference clock pre-divider. Default to Integer, 1b1.
REF_MASK	"0'b00000000" *(default)*	Minimum divider ratio control word for reference pre-divider
REF_MMD_DIG	"8" *(default)* "0"	MMD divider ratio setting for reference pre-divider when lmmi_ref_integer_mode=1
REF_MMD_IN	"0'b00001000" *(default)* "00000000"	MMD divider ratio setting for reference pre-divider when lmmi_ref_integer_mode=0
REF_MMD_PULS_CTL	"0'b0000" *(default)*	Pulse width control for MMD output clock in reference pre-divider
REF_TIMING_CTL	"0'b00" *(default)*	Interface timing control for reference divider. Default to be 2b00.
REFIN_RESET	"SET" *(default)* "RESET"	Lmmi_refin_reset = 1b1 and with switching of REFin_SEL (either L-H or H-L) will generate a PLL reset

Table 557: Parameters

Name	Values	Description
RESERVED	"0'b0000000" *(default)*	Reserved fuses for PLL internal setting
RESET_LF	"DISABLED" *(default)* "ENABLED"	LPF reset enable
ROTATE	"DISABLED" *(default)* "ENABLED"	For VCO phase rotation
SEL_OUTA	"DISABLED" *(default)* "ENABLED"	Select output to CLKOP
SEL_OUTB	"DISABLED" *(default)* "ENABLED"	Select output to CLKOS
SEL_OUTC	"DISABLED" *(default)* "ENABLED"	Select output to CLKOS2
SEL_OUTD	"DISABLED" *(default)* "ENABLED"	Select output to CLKOS3
SEL_OUTE	"DISABLED" *(default)* "ENABLED"	Select output to CLKOS4
SEL_OUTF	"DISABLED" *(default)* "ENABLED"	Select output to CLKOS5
SLEEP	"DISABLED" *(default)* "ENABLED"	Active HIGH: Enable STOP PMU signal
SSC_DITHER	"DISABLED" *(default)* "ENABLED"	Dither enable or disable for SDM
SSC_EN_CENTER_IN	"DOWN_TRIANGLE" *(default)* "CENTER_TRIANGLE"	Down triangle or center triangle control bit in SSC profile generator.
SSC_EN_SDM	"ENABLED" *(default)* "DISABLED"	Enable or disable SDM
SSC_EN_SSC	"DISABLED" *(default)* "ENABLED"	Enable or disabled SSC profile generator
SSC_F_CODE	"0'b000000000000000" *(default)*	Fractional part of the feedback divider ratio
SSC_N_CODE	"0'b000010100" *(default)* "000000000"	Integer part of the feedback divider ratio
SSC_ORDER	"SDM_ORDER2" *(default)* "SDM_ORDER1"	SDM order control bit; 0 - SDM order = 1; 1 - SDM order = 2
SSC_PI_BYPASS	"NOT_BYPASSED" *(default)* "BYPASSED"	PI bypass control bit. 0; PI not bypass, 1; PI bypass; it should be connected to lmmi_fbk_pi_bypass
SSC_REG_WEIGHTING_SEL	"0'b000" *(default)*	Weighting control bit for lmmi_ssc_step_in
SSC_SQUARE_MODE	"DISABLED" *(default)* "ENABLED"	Two-point FSK modulation control bit
SSC_STEP_IN	"0'b0000000" *(default)*	SSC modulation depth control bit

Table 557: Parameters

Name	Values	Description
SSC_TBASE	"0'b000000000000" *(default)*	SSC modulation frequency control. The frequency should be 30-33 kHz.
STDBY_ATT	"DISABLED" *(default)* "ENABLED"	Enable STDBY CIB signal
TRIMOP_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOP output trim
TRIMOS_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOS output trim
TRIMOS2_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOS2 output trim
TRIMOS3_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOS3 output trim
TRIMOS4_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOS4 output trim
TRIMOS5_BYPASS_N	"BYPASSED" *(default)* "USED"	1b0: bypass CLKOS5 output trim
V2I_KVCO_SEL	"85" *(default)* "40" "45" "50" "55" "60" "65" "70" "75" "80" "90" "95" "100" "105" "110" "115"	Mlk choose such as frequency range of ICO from 800 MHz - 1600 MHz over PVT. Mlk larger (more current withdraw from ICO), ICO runs slower.
V2I_PP_ICTRL	"0'b00110" *(default)* "0'b00000"	P-path v2i gm control
V2I_PP_RES	"10K" *(default)* "11P3K" "11K" "10P7K" "10P3K" "9P7K" "9P3K" "9K"	P-path high frequency pole resistor control

Table 557: Parameters

Name	Values	Description
CLKMUX_FB	"CMUX_CLKOP" *(default)* "CMUX_CLKOS" "CMUX_CLKOS2" "CMUX_CLKOS3" "CMUX_CLKOS4" "CMUX_CLKOS5"	Fuses used to determine the feedback selection in the wake up stage, center mux about the clock tree feedback
SEL_FBK	"DIVA" *(default)* "DIVB" "DIVC" "DIVD" "DIVE" "DIVF" "RESERVED" "FBKCLK0" "FBKCLK1" "FBKCLK2" "FBKCLK3" "FBKCLK4" "FBKCLK5" "FBKCLK6" "FBKCLK7" "FBKCLK8"	Select feedback clock
DIV_DEL	"0'b0000001" *(default)* "0000000"	The internal path delay selection path
PHASE_SEL_DEL	"0'b000" *(default)*	The internal phase delay selection path
PHASE_SEL_DEL_P1	"0'b000" *(default)*	The internal phase delay selection path1
EXTERNAL_DIVIDE_FACTOR	"0" *(default)*	External divider value for feedback clock
INTFBKDEL_SEL	"DISABLED" *(default)* "ENABLED"	Associated with the attribute INTFBKDEL_SEL. 1'b1 to enable the internal path switching during POR/Sleep/Standby modes.
PMU_WAITFORLOCK	"ENABLED" *(default)* "DISABLED"	Sync with pmu to wait for PLL lock.
REF_OSC_CTRL	"3P2" *(default)* "1P0"	Select monitoring clock frequency: 3.2 MHz or 1 MHz.
SIM_FLOAT_PRECISION	"0.0001" *(default)* "0.001" "0.01" "0.1"	Tolerance control for frequency locking.
CONFIG_WAIT_FOR_LOCK	"DISABLED" (default) "ENABLED"	Wait for PLL lock during configuration Note: This attribute is not available in Avant.

PLL Device Constraints:

Below are the device constraint equations used by the timer for the periods, phases, and delays of the output clocks based on user settings. START_PERIOD must be provided as a constraint. These equations only apply for the integer mode divider values.

For CLKOP (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVA + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHIA * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELA

For CLKOS (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVB + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHIB * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELB

For CLKOS2 (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVC + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHIC * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELC)

For CLKOS3 (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVD + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHID * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELD

For CLKOS4 (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVE + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHIE * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELE

For CLKOS5 (generated clock):

- ADJUSTED_PERIOD = START_PERIOD * (REF_MMD_DIG + 1) * (DIVF + 1) * (1/(FBK_MMD_DIG + 1))
- ADJUSTED_PHASE = ((PHIF * 45)/360) * ADJUSTED_PERIOD
- ADJUSTED_DELAY = START_PERIOD * (REF_MMD_DIG + 1) * (1/(FBK_MMD_DIG + 1)) * DELF

PLLA

Wrapper for Phase Locked Loop

PLLA is capable of frequency synthesis and clock phase management including clock injection delay cancellation.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

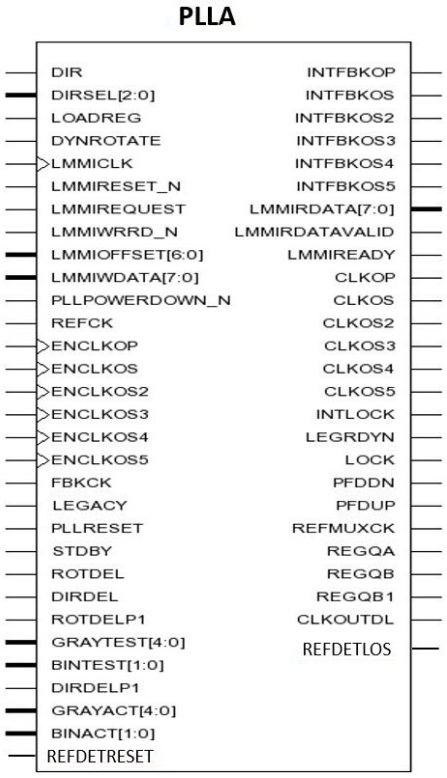


Table 558: Input Ports

Name	Range	Description
DIR		Valid if MC1_DYN_SOURCE = 1. Specify direction that CIB_ROTATE changes VCO phase. 0 - phase rotates to later phase. 1 - Phase rotates to earlier phase
DIRSEL	2:0	Valid if MC1_DYN_SOURCE = 1. CIB_DYN_SEL must be stable before initiating a rotation with CIB_LOAD_REG
LOADREG		Valid if MC1_DYN_SOURCE = 1. Initiate a divider output phase shift on negative edge of CIB_LOAD_REG
DYNROTATE		Valid if MC1_DYN_SOURCE = 1. Initiate a change from current VCO clock phase to an earlier or later phase on the negative edge of CIB_ROTATE
LMMICLK		LMMI Clock from Fabric
LMMIRESET_N		LMMI Reset Signal to reset the state machine if the IP gets locked
LMMIREQUEST		LMMI Request signal from Fabric
LMMIWRRD_N		LMMI Write High / Read Low from Fabric
LMMIOFFSET	6:0	LMMI Offset Address from Fabric, not all bits are required for an IP
LMMIWDATA	7:0	LMMI Write Data from Fabric, not all bits are required for an IP
PLLPOWERDOWN_N		Active low to power down PLL from CIB port
REFCK		
ENCLKOP		Enable A output (CLKOP). Active high. PLL CIB input
ENCLKOS		Enable B output (CLKOS). Active high. PLL CIB input
ENCLKOS2		Enable C output (CLKOS2). Active high. PLL CIB input
ENCLKOS3		Enable D output (CLKOS3). Active high. PLL CIB input
ENCLKOS4		Enable E output (CLKOS4). Active high. PLL CIB input
ENCLKOS5		Enable F output (CLKOS5). Active high. PLL CIB input
FBKCK		

Table 558: Input Ports (Continued)

Name	Range	Description
LEGACY		PLL legacy mode signal. Active high to enter the mode. Enabled by lmmi_legacy fuse. PLL CIB input
PLLRESET		Active high to reset PLL. Enabled by MC1_PLLRESET. PLL CIB input
STDBY		PLL standby signal. Active high to put PLL clocks in low. Not used
ROTDEL		
DIRDEL		
ROTDELP1		
GRAYTEST	4:0	
BINTEST	1:0	
DIRDELP1		
GRAYACT	4:0	
BINACT	1:0	
REFDETRESET		

Table 559: Output Ports

Name	Range	Description
INTFBKOP		
INTFBKOS		
INTFBKOS2		
INTFBKOS3		
INTFBKOS4		
INTFBKOS5		
LMMIRDATA	7:0	LMMI Read Data to Fabric
LMMIRDATAVALID		LMMI Read Date Valid to Fabric
LMMIREADY		LMMI Ready signal to Fabric
CLKOP		Primary (A) output clock
CLKOS		Secondary (B) output clock
CLKOS2		Secondary (C) output clock
CLKOS3		Secondary (D) output clock
CLKOS4		Secondary (E) output clock

Table 559: Output Ports (Continued)

Name	Range	Description
CLKOS5		Secondary (F) output clock
INTLOCK		PLL internal lock indicator. PLL CIB output
LEGRDYN		
LOCK		PLL lock indicator. PLL CIB output
PFDDN		PFD DN output signal to PLL CIB port
PFDUP		PFD UP output signal to PLL CIB port
REFMUXCK		Reference CLK mux output. PLL CIB output
REGQA		
REGQB		
REGQB1		
CLKOUTDL		
REFDETLOS		

Table 560: Parameters

Name	Values	Description
BW_CTL_BIAS	"0b0101" (default) "0000"	Input control signal to tune the bias current of ppath cp, when the bit increases, the bias current also increases. This current branch is combined with the Ivco_fb current.
CLKOP_TRIM	"0b0000" (default)	CLKOP output trim control bits
CLKOS_TRIM	"0b0000" (default)	CLKOS output trim control bits
CLKOS2_TRIM	"0b0000" (default)	CLKOS2 output trim control bits
CLKOS3_TRIM	"0b0000" (default)	CLKOS3 output trim control bits
CLKOS4_TRIM	"0b0000" (default)	CLKOS4 output trim control bits
CLKOS5_TRIM	"0b0000" (default)	CLKOS5 output trim control bits

Table 560: Parameters (Continued)

Name	Values	Description
CRIPPLE	"5P" (default) "1P" "3P" "7P" "9P" "11P" "13P" "15P"	LPF cap control
CSET	"40P" (default) "8P" "12P" "16P" "20P" "24P" "28P" "32P" "36P" "44P" "48P" "52P" "56P" "60P" "64P" "68P"	LPF cap control
DELAY_CTRL	"200PS" (default) "300PS"	Control signal to adjust the delay of the PFD; default 1b0 = 200ps; 1b1 = 300ps
DELA	"0" (default)	Output divider phase shift (work with lmmi_diva)
DELB	"0" (default)	Output divider phase shift (work with lmmi_divb)
DELC	"0" (default)	Output divider phase shift (work with lmmi_divc)
DELD	"0" (default)	Output divider phase shift (work with lmmi_divd)
DELE	"0" (default)	Output divider phase shift (work with lmmi_dive)
DELF	"0" (default)	Output divider phase shift (work with lmmi_divf)
DIRECTION	"DISABLED" (default) "ENABLED"	lmmi equivalent of CIB direction signal
DIVA	"0" (default)	Output dividers for clkop
DIVB	"0" (default)	Output dividers for clkos

Table 560: Parameters (Continued)

Name	Values	Description
DIVC	"0" (default)	Output dividers for clkos2
DIVD	"0" (default)	Output dividers for clkos3
DIVE	"0" (default)	Output dividers for clkos4
DIVF	"0" (default)	Output dividers for clkos5
DYN_SEL	"0b000" (default)	Select dynamic phase selection
DYN_SOURCE	"STATIC" (default) "DYNAMIC"	1'b1 select CIB signals for dynamic phase shift
ENCLK_CLKOP	"DISABLED" (default) "ENABLED"	Bit 0. Active HIGH: Clock output enable
ENCLK_CLKOS	"DISABLED" (default) "ENABLED"	Bit 1. Active HIGH: Clock output enable
ENCLK_CLKOS2	"DISABLED" (default) "ENABLED"	Bit 2. Active HIGH: Clock output enable
ENCLK_CLKOS3	"DISABLED" (default) "ENABLED"	Bit 3. Active HIGH: Clock output enable
ENCLK_CLKOS4	"DISABLED" (default) "ENABLED"	Bit 4. Active HIGH: Clock output enable
ENCLK_CLKOS5	"DISABLED" (default) "ENABLED"	Bit 5. Active HIGH: Clock output enable
ENABLE_SYNC	"DISABLED" (default) "ENABLED"	1'b1 enable output(s) sync with clkop
FAST_LOCK_EN	"ENABLED" (default) "DISABLED"	Enable signal for fast lock
V2I_1V_EN	"DISABLED" (default) "ENABLED"	1 V supply enable or disable
FBK_CUR_BLE	"0b00000000" (default)	Bleeding current for PI to adjust the linearity
FBK_EDGE_SEL	"POSITIVE" (default) "NEGATIVE"	Select the positive or negative phase of PI output. 0: positive phase; 1: negative phase
FBK_IF_TIMING_CTL	"0b00" (default)	Interface timing control for feedback divider
FBK_INTEGER_MODE	"DISABLED" (default) "ENABLED"	Enable the integer mode for feedback divider

Table 560: Parameters (Continued)

Name	Values	Description
FBK_MASK	"0b00001000" (default) "00000000"	Minimum divider ratio control word for feedback divider. For example, if n_pll[7:0] or mmd_dig[7:0] is less than lmmi_fbk_mask[7:0], then the MMD divider ratio is determined by lmmi_fbk_mask[7:0]. Otherwise, the divider ratio will be determined by n_pll[7:0] or mmd_dig[7:0]
FBK_MMD_DIG	"8" (default) "0"	MMD divider ratio setting in integer mode
FBK_MMD_PULS_CTL	"0b0000" (default)	Pulse width control for MMD output clock. If lmmi_fbk_mmd_puls_ctl[3:0]=4'b0110, it means that there are 6 VCO cycles in the MMD output clock
FBK_MODE	"0b00" (default)	Reserved floating control bits
FBK_PI_BYPASS	"NOT_BYPASSED" (default) "BYPASSED"	PI bypass control bit. 0; PI not bypass, 1; PI bypass; it should be connected to lmmi_ssc_pi_bypass
FBK_PI_RC	"0b1100" (default) "0000"	RC time constant control in PI
FBK_PR_CC	"0b0000" (default)	Current control for PI to adjust the linearity
FBK_PR_IC	"0b1000" (default) "0000"	Bias current control for PI
FBK_RSV	"0b0000000000000000" (default)	Reserved control bit for feedback divider
FLOAT_CP	"DISABLED" (default) "ENABLED"	Active HIGH to tri-state the ICP output
FLOCK_CTRL	"2X" (default) "1X" "4X" "8X"	2 bits control the fast lock period, 00 is 1x, 01 is 2x, 10 is 4x and 11 is 8x
FLOCK_EN	"ENABLED" (default) "DISABLED"	1b1 to enable fast lock; after char, default to enabled
FLOCK_SRC_SEL	"REFCLK" (default) "FBCLK"	Fast lock source selection; 0 is ref clock; 1 is feedback clock
FORCE_FILTER	"DISABLED" (default) "ENABLED"	Force internal vctrl=analog pad

Table 560: Parameters (Continued)

Name	Values	Description
I_CTRL	"10UA" (default) "8P3UA" "14P9UA" "12P4UA" "19P8UA" "17P3UA" "24P8UA" "22P3UA"	Current tuning
IPI_CMP	"0b1000" (default) "0000"	i-path CP compensate up/dn mismatch at process variation
IPI_CMPN	"0b0011" (default) "0000"	Input control bits to compensate the i-path bias current
IPI_COMP_EN	"DISABLED" (default) "ENABLED"	Enable ipi_cmp
IPP_CTRL	"0b1000" (default) "0000"	Input control signal to tune the bias current of ppath cp
IPP_SEL	"0b1111" (default) "0000"	Input control signal to select which ppath cp is on, there are 4 branches at max
KP_VCO	"0b00000" (default)	ICO gain controls
LDT_INT_LOCK_STICKY	"DISABLED" (default) "ENABLED"	Active HIGH to have INT_LOCK_STICKY. Default to be 1b1. 1b0 is for PDE/DE purpose
LDT_LOCK	"98304CYC" (default) "24576CYC" "6144CYC" "1536CYC"	Frequency lock-detector resolution sensitivity
LDT_LOCK_SEL	"U_FREQ" (default) "UFREQ" "SPHASE" "SFREQ" "UFREQ_SPHASE" "U_PHASE" "S_FREQ" "U_FREQ_S_PHASE"	Lock-detector type select
LEGACY_ATT	"DISABLED" (default) "ENABLED"	Active HIGH; enable Legacy mode

Table 560: Parameters (Continued)

Name	Values	Description
LOAD_REG	"DISABLED" (default) "ENABLED"	For load control of divider phase control
MFG_CTRL	"0b0000" (default)	MFG internal vctrl selection
MFG_EN	"DISABLED" (default) "ENABLED"	MFG feature enable pin
MFG_SEL	"ICP_UP" (default) "PCP_UP" "IV2I" "PV21" "ICP_DN" "PCP_DN" "RSVD"	MFG current mux selection
MFGOUT1_SEL	"0b000" (default)	PLL_MFGOUT1 selection bits
MFGOUT2_SEL	“0b000" (default)	PLL_MFGOUT2 selection bits
OPENLOOP_EN	"DISABLED" (default) "ENABLED"	Open loop mode enable for mfg testing
PHIA	"0" (default)	Select VCO phase-shift (0.7) for A section
PHIB	"0" (default)	Select VCO phase-shift (0.7) for B section
PHIC	"0" (default)	Select VCO phase-shift (0.7) for C section
PHID	"0" (default)	Select VCO phase-shift (0.7) for D section
PHIE	"0" (default)	Select VCO phase-shift (0.7) for E section
PHIF	"0" (default)	Select VCO phase-shift (0.7) for F section
PLLPDN_EN	"DISABLED" (default) "ENABLED"	
PLLPD_N	"UNUSED" (default) "USED"	Active LOW PLL power-down; PLL is NOT USED
PLLRESET_ENA	"DISABLED" (default) "ENABLED"	Active HIGH; Enable PLLRESET CIB Signal

Table 560: Parameters (Continued)

Name	Values	Description
REF_COUNTS	"0b0000" (default) "0b0001" "0b0010" "0b0011" "0b0100" "0b1000" "0b1001" "0b1010" "0b1011" "0b1100"	Internal count control for reference clock detection
REF_ENABLE	"DISABLED" (default) "ENABLED"	Enable reference clock detection output to REFDETLOS swPort Always "ENABLED" in User Primtive SW primitive uses the default in 2nd column
REF_INTEGER_MODE	"DISABLED" (default) "ENABLED"	Integer mode control bit for reference clock pre-divider. Default to Integer, 1b1
REF_MASK	"0b00000000" (default)	Minimum divider ratio control word for reference pre-divider
REF_MMD_DIG	"8" (default) "0"	MMD divider ratio setting for reference pre-divider when lmmi_ref_integer_mode=1
REF_MMD_IN	"0b00001000" (default) "00000000"	MMD divider ratio setting for reference pre-divider when lmmi_ref_integer_mode=0
REF_MMD_PULS_CTL	"0b0000" (default)	Pulse width control for MMD output clock in reference pre-divider
REF_TIMING_CTL	"0b00" (default)	Interface timing control for reference divider. Default to be 2b00
REF_USE	"DISABLED" (default) "ENABLED"	Enable internal reset based on reference clock detection output Always "ENABLED" in User Primtive SW primitive uses the default in 2nd column
REFIN_RESET	"SET" (default) "RESET"	Lmmi_refin_reset = 1b1 and with switching of REFin_SEL (either L-H or H-L) will generate a PLL reset
RESERVED	"0b0000000" (default)	Reserved fuses for PLL internal setting
RESET_LF	"DISABLED" (default) "ENABLED"	LPF reset enable
ROTATE	"DISABLED" (default) "ENABLED"	For VCO phase rotation

Table 560: Parameters (Continued)

Name	Values	Description
SEL_OUTA	"DISABLED" (default) "ENABLED"	Select output to CLKOP
SEL_OUTB	"DISABLED" (default) "ENABLED"	Select output to CLKOS
SEL_OUTC	"DISABLED" (default) "ENABLED"	Select output to CLKOS2
SEL_OUTD	"DISABLED" (default) "ENABLED"	Select output to CLKOS3
SEL_OUTE	"DISABLED" (default) "ENABLED"	Select output to CLKOS4
SEL_OUTF	"DISABLED" (default) "ENABLED"	Select output to CLKOS5
SLEEP	"DISABLED" (default) "ENABLED"	Active HIGH: Enable STOP PMU signal
SSC_DELTA	"0b000000000000000" (default)	Not used in LFCPNX, LFD2NX, or LIFCL
SSC_DELTA_CTL	"0b00" (default)	Not used in LFCPNX, LFD2NX, or LIFCL
SSC_DITHER	"DISABLED" (default) "ENABLED"	Dither enable or disable for SDM
SSC_EN_CENTER_IN	"DOWN_TRIANGLE" (default) "CENTER_TRIANGLE"	Down triangle or center triangle control bit in SSC profile generator
SSC_EN_SDM	"ENABLED" (default) "DISABLED"	Enable or disable SDM
SSC_EN_SSC	"DISABLED" (default) "ENABLED"	Enable or disabled SSC profile generator
SSC_F_CODE	"0b000000000000000" (default)	Fractional part of the feedback divider ratio
SSC_N_CODE	"0b000010100" (default) "000000000"	Integer part of the feedback divider ratio
SSC_ORDER	"SDM_ORDER2" (default) "SDM_ORDER1"	SDM order control bit; 0 - SDM order = 1; 1 - SDM order = 2
SSC_PI_BYPASS	"NOT_BYPASSED" (default) "BYPASSED"	PI bypass control bit. 0; PI not bypass, 1; PI bypass; it should be connected to lmmi_fbk_pi_bypass
SSC_REG_WEIGHTING_SEL	"0b000" (default)	Weighting control bit for lmmi_ssc_step_in

Table 560: Parameters (Continued)

Name	Values	Description
SSC_SQUARE_MODE	"DISABLED" (default) "ENABLED"	Two-point FSK modulation control bit
SSC_STEP_IN	"0b0000000" (default)	SSC modulation depth control bit
SSC_TBASE	"0b000000000000" (default)	SSC modulation frequency control. The frequency should be 30-33kHz
STDBY_ATT	"DISABLED" (default) "ENABLED"	Enable STDBY CIB signal
TRIMOP_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOP output trim
TRIMOS_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOS output trim
TRIMOS2_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOS2 output trim
TRIMOS3_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOS3 output trim
TRIMOS4_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOS4 output trim
TRIMOS5_BYPASS_N	"BYPASSED" (default) "USED"	1b0: bypass CLKOS5 output trim
V2I_KVCO_SEL	"85" (default) "10" "15" "20" "25" "30" "35" "40" "45" "50" "55" "60" "65" "70" "75" "80"	Mlk chooses such as frequency range of ICO from 800 MHz - 1600 MHz over PVT. Mlk larger (more current withdraw from ICO) ICO run slower
V2I_PP_ICTRL	"0b00000" (default)	P-path v2i gm control

Table 560: Parameters (Continued)

Name	Values	Description
V2I_PP_RES	"11P3K" (default) "11K" "10P7K" "10P3K" "10K" "9P7K" "9P3K" "9K"	P-path high frequency pole resistor control
CLKMUX_FB	"CMUX_CLKOP" (default) "CMUX_CLKOS" "CMUX_CLKOS2" "CMUX_CLKOS3" "CMUX_CLKOS4" "CMUX_CLKOS5"	Fuses used to determine the feedback selection in the wake up stage, center mux about the clock tree feedback
SEL_FBK	"DIVA" (default) "DIVB" "DIVC" "DIVD" "DIVE" "DIVF" "RESERVED" "FBKCLK0" "FBKCLK1" "FBKCLK2" "FBKCLK3" "FBKCLK4" "FBKCLK5" "FBKCLK6" "FBKCLK7" "FBKCLK8"	Select feedback clock
DIV_DEL	"0b0000001" (default) "0000000"	The internal path delay selection path
PHASE_SEL_DEL	"0b000" (default)	The internal phase delay selection path
PHASE_SEL_DEL_P1	"0b000" (default)	The internal phase delay selection path1
EXTERNAL_DIVIDE_FACTOR	"0" (default)	External divider value for feedback clock

PLL_B

Phase Locked Loop

Architectures supported:

- iCE40 UltraPlus

USER INSTANTIATION: Illegal. Please refer to RGB.

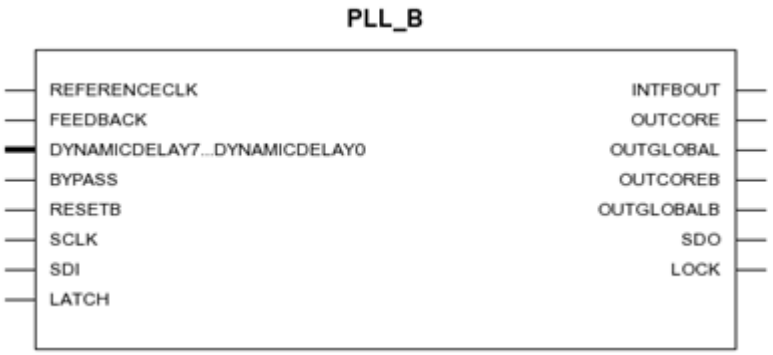


Table 561: Input Ports

Name	Range	Description
REFERENCECLK		Reference clock
FEEDBACK		External feedback input
DYNAMICDELAY7... DYNAMICDELAY0		Dynamic control of delay for adjusting the phase alignment. Only used when DELAY_ADJUSTMENT_MODE is set to DYNAMIC. RESETB must be active before this bus can change value in order to ensure proper PLL functionality. Once value is changed user must set RESETB high and wait for PLL to lock
BYPASS		Bypass PLL core (directly connect REFERENCECLK to OUTCORE/OUTGLOBAL pins)
RESET_N		Asynchronously reset the PLL, active low
SCLK		Clock for internal testing
SDI		Data for internal testing
LATCH		Enable low power mode, active high. OUTGLOBAL*/OUTCORE* ports are held static at their last value when ENABLE_ICEGATE_PORT* is set to 1

Table 562: Output Ports

Name	Range	Description
INTFBOUT		For internal feedback PLL operation, must route this output to FEEDBACK
OUTCORE		Output clock A, drives regular fabric routing
OUTGLOBAL		Output clock A, drives global clock network
OUTCOREB		Output clock B, drives regular fabric routing
OUTGLOBALB		Output clock B, drives global clock network
SDO		Data for internal testing
LOCK		Indicates that the output signal on OUTGLOBALB/OUTCOREB is locked to the REFERENCECLK port, active high

Table 563: Parameters

Name	Description
FEEDBACK_PATH	"SIMPLE" (default), "DELAY", "PHASE_AND_DELAY" (Selects the feedback path. SIMPLE = internal feedback, directly from VCO. DELAY = internal feedback, through the delay adjustment block. PHASE_AND_DELAY = internal feedback, through the phase shifter and delay adjustment block)
DELAY_ADJUSTMENT_MODE_FEEDBACK	"FIXED" (default), "DYNAMIC" (Selects the mode for the delay adjustment block in the feedback path. FIXED = fixed delay, selected with the FDA_FEEDBACK parameter. DYNAMIC = dynamic delay, selected by the DYNAMICDELAY port)
FDA_FEEDBACK	"0" (default), "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15" (Sets a constant value for the delay adjustment block. OUTGLOBALA and OUTCOREA are delayed by (n+1)*150ps)
DELAY_ADJUSTMENT_MODE_RELATIVE	"FIXED" (default), "DYNAMIC" (Selects the mode for the delay adjustment block)
FDA_RELATIVE	"0" (default), "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15" (Selects a constant value for the delay adjustment block. OUTGLOBALA and OUTCOREA are delayed with regards to the B output signals, by (n+1)*150ps)
SHIFTREG_DIV_MODE	"0" (default), "1" (Selects shift register configuration. 0 = divide by 4, 1 = divide by 7. Used when FEEDBACK_PATH is set to PHASE_AND_DELAY)

Table 563: Parameters (Continued)

Name	Description
PLLOUT_SELECT_PORTA	SHIFTREG_0deg" (default), "SHIFTREG_90deg", "GENCLK", "GENCLK_HALF" (Configures the OUTCOREA and OUTGLOBALA ports. SHIFTREG_0deg = 0 degree phase shift (FEEDBACK_PATH must also be set to PHASE_AND_DELAY), SHIFTREG_90deg = 90 degree phase shift (FEEDBACK_PATH must be set to PHASE_AND_DELAY, and SHIFTREG_DIV_MODE to 0), GENCLK = the internally generated frequency and no phase shift, GENCLK_HALF = half of the internally generated frequency and no phase shift)
PLLOUT_SELECT_PORTB	"SHIFTREG_0deg" (default), "SHIFTREG_90deg", "GENCLK", "GENCLK_HALF" (Configures the OUTCOREB and OUTGLOBALB ports. SHIFTREG_0deg = 0 degree phase shift (FEEDBACK_PATH must also be set to PHASE_AND_DELAY), SHIFTREG_90deg = 90 degree phase shift (FEEDBACK_PATH must be set to PHASE_AND_DELAY, and SHIFTREG_DIV_MODE to 0), GENCLK = the internally generated frequency and no phase shift, GENCLK_HALF = half of the internally generated frequency and no phase shift)
DIVR	"0" (default), "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15" (REFERENCECLK divider)
DIVF	"0" (default), "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58", "59", "60", "61", "62", "63", "64" , "65", "66", "67", "68", "69", "70", "71", "72", "73", "74" , "75", "76", "77", "78", "79", "80", "81", "82", "83", "84" , "85", "86", "87", "88", "89", "90", "91", "92", "93", "94" , "95", "96", "97", "98", "99", "100", "101", "102", "103", "104" , "105", "106", "107", "108", "109", "110", "111", "112", "113", "114" , "115", "116", "117", "118", "119", "120", "121", "122", "123", "124" , "125", "126", "127" (Feedback divider)
DIVQ	"1" (default), "2", "3", "4", "5", "6" (VCO divider)
FILTER_RANGE	"0" (default), "1", "2", "3", "4", "5", "6", "7" (PLL filter range)
EXTERNAL_DIVIDE_FACTOR	"NONE" (default), (Divide-by factor of a divider in the external feedback path. Can be set to any integer value)
ENABLE_ICEGATE_PORTA	"0" (default), "1" (Enables the PLL power-down control for port A, 0 = disabled, 1 = controlled by LATCH input)
ENABLE_ICEGATE_PORTB	"0" (default), "1" (Enables the PLL power-down control for port B, 0 = disabled, 1 = controlled by LATCH input)
TEST_MODE	"0" (default), "1" (Enables the use of test ports, SCLK, SDI, SDO. Disabled = 0, Enabled = 1)
FREQUENCY_PIN_REFERENCECLK	"NONE" (default), (SDC Constraint entry on REFERENCECLK port of PLL)

Rules & Restrictions:

FEEDBACK port (external feedback input pin) CANNOT be VHI, VLO, or floating. Otherwise, PLL in HW will not operate correctly.

When EXTERNAL_DIVIDE_FACTOR != NONE, DELAY_ADJUSTMENT_MODE should be set to either FIXED or DYNAMIC.

When EXTERNAL_DIVIDE_FACTOR != NONE, PLLOUT_SELECT_PORTA != SHIFTREG_0deg, PLLOUT_SELECT_PORTA != SHIFTREG_90deg, PLLOUT_SELECT_PORTB != SHIFTREG_0deg, PLLOUT_SELECT_PORTB != SHIFTREG_90deg.

When FEEDBACK_PATH = DELAY, DELAY_ADJUSTMENT_MODE_FEEDBACK should be set to either FIXED or DYNAMIC.

When FEEDBACK_PATH = DELAY, PLLOUT_SELECT_PORTA != SHIFTREG_0deg, PLLOUT_SELECT_PORTA != SHIFTREG_90deg, PLLOUT_SELECT_PORTB != SHIFTREG_0deg, PLLOUT_SELECT_PORTB != SHIFTREG_90deg.

When FEEDBACK_PATH =PHASE_AND_ DELAY, DELAY_ADJUSTMENT_MODE_FEEDBACK should be set to either FIXED or DYNAMIC.

When FEEDBACK_PATH = PHASE_AND_DELAY both PLLOUT_SELECT_PORTA and PLLOUT_SELECT_PORTB must be set to either SHIFTREG_0deg or SHIFTREG_90deg.

When EXTERNAL_DIVIDE_FACTOR = NONE, make sure INTFBOUT is connected to FEEDBACK.

When EXTERNAL_DIVIDE_FACTOR != NONE, make sure INTFBOUT is NOT connected to FEEDBACK.

REFERNCECLK input must be between 10 and 133 MHz.

How to calculate output frequency:

When FEEDBACK_PATH = “SIMPLE,” the calculation of the period of the output clock of PLL_B is:

Output_period = reference_clk_period * __EQ__[EXTERNAL_DIVIDE_FACTOR,NONE,(2^DIVQ),1/EXTERNAL_DIVIDE_FACTOR] * (DIVR + 1) * (1/(DIVF + 1)) * __EQ__[PLLOUT_SELECT_PORTB,GENCLK_HALF,2,1]

When FEEDBACK_PATH = “DELAY,” the calculation of the period of the output clock of PLL_B is:

Output_period = reference_clk_period * __EQ__[EXTERNAL_DIVIDE_FACTOR,NONE,1,1/

EXTERNAL_DIVIDE_FACTOR] * (DIVR + 1) * (1/(DIVF + 1)) *
__EQ__[PLLOUT_SELECT_PORTB,GENCLK_HALF,2,1]

When FEEDBACK_PATH = “PHASE_AND_DELAY,” the calculation of the period of the output clock of PLL_B is:

Output_period = reference_clk_period *
__EQ__[EXTERNAL_DIVIDE_FACTOR,NONE,1,1/
EXTERNAL_DIVIDE_FACTOR] * (DIVR + 1) * (1/(DIVF + 1)) *
__EQ__[PLLOUT_SELECT_PORTA,GENCLK_HALF,2,1] *
__EQ__[PLLOUT_SELECT_PORTA,SHIFTREG 0deg,
__EQ__[SHIFTREG_DIV_MODE,1, 7, 4], 1] *
__EQ__[PLLOUT_SELECT_PORTA,SHIFTREG 0deg,
__EQ__[SHIFTREG_DIV_MODE,1, 7, 4],1]

In the above equations, the sections beginning with “__EQ__” and enclosed in “[ ]” are treated as if/else statements during calculation. For example, “__EQ__[EXTERNAL_DIVIDE_FACTOR,NONE,1,1/
EXTERNAL_DIVIDE_FACTOR]” means that if EXTERNAL_DIVIDE_FACTOR is set to NONE then use the value “1,” otherwise use the value “1/EXTERNAL_DIVIDE_FACTOR” in the calculation.

Note:

EXTERNAL_DIVIDE_FACTOR is embedded in each of the equations, so even if you are using external feedback in your design, you will still need to choose one of the above equations based on the FEEDBACK_PATH setting.

Example PLL_B Instantiation (External Feedback):

```
PLL_B PLL_B_inst (
.REFERENCECLK      (REFERENCECLK   ),
.FEEDBACK          (FEEDBACK       ),
.DYNAMICDELAY7     (DYNAMICDELAY7  ),
.DYNAMICDELAY6     (DYNAMICDELAY6  ),
.DYNAMICDELAY5     (DYNAMICDELAY5  ),
.DYNAMICDELAY4     (DYNAMICDELAY4  ),
.DYNAMICDELAY3     (DYNAMICDELAY3  ),
.DYNAMICDELAY2     (DYNAMICDELAY2  ),
.DYNAMICDELAY1     (DYNAMICDELAY1  ),
.DYNAMICDELAY0     (DYNAMICDELAY0  ),
.BYPASS            (BYPASS         ),
.RESET_N           (RESET_N        ),
.SCLK              (SCLK           ),
.SDI               (SDI            ),
.LATCH             (LATCH          ),
.INTFBOUT          (/*NOT CONNECTED*/),
.OUTCORE           (OUTCORE        ),
.OUTGLOBAL         (               ),
.OUTCOREB          (OUTCOREB       ),
.OUTGLOBALB        (               ),
.SDO               (SDO            ),
.LOCK              (LOCK           )
);

defparam PLL_B_inst.EXTERNAL_DIVIDE_FACTOR = "15";
```

```
defparam PLL_B_inst.DELAY_ADJUSTMENT_MODE_FEEDBACK = "DYNAMIC";
```

Example PLL_B Instantiation (Internal Feedback):

```
PLL_B PLL_B_inst (
.REFERENCECLK       (REFERENCECLK   ),
.FEEDBACK           (INTFBOUT       ),
.DYNAMICDELAY7      (DYNAMICDELAY7  ),
.DYNAMICDELAY6      (DYNAMICDELAY6  ),
.DYNAMICDELAY5      (DYNAMICDELAY5  ),
.DYNAMICDELAY4      (DYNAMICDELAY4  ),
.DYNAMICDELAY3      (DYNAMICDELAY3  ),
.DYNAMICDELAY2      (DYNAMICDELAY2  ),
.DYNAMICDELAY1      (DYNAMICDELAY1  ),
.DYNAMICDELAY0      (DYNAMICDELAY0  ),
.BYPASS             (BYPASS         ),
.RESET_N            (RESET_N        ),
.SCLK               (SCLK           ),
.SDI                (SDI            ),
.LATCH              (LATCH          ),
.INTFBOUT           (INTFBOUT       ),
.OUTCORE            (OUTCORE        ),
.OUTGLOBAL          (               ),
.OUTCOREB           (OUTCOREB       ),
.OUTGLOBALB         (               ),
.SDO                (SDO            ),
.LOCK               (LOCK           )
);

defparam PLL_B_inst.EXTERNAL_DIVIDE_FACTOR = "NONE";
defparam PLL_B_inst.FEEDBACK_PATH = "SIMPLE";
```

PLLC

PLL functional block

Architectures supported:

- LAV-AT-ES1
- LN2-CT

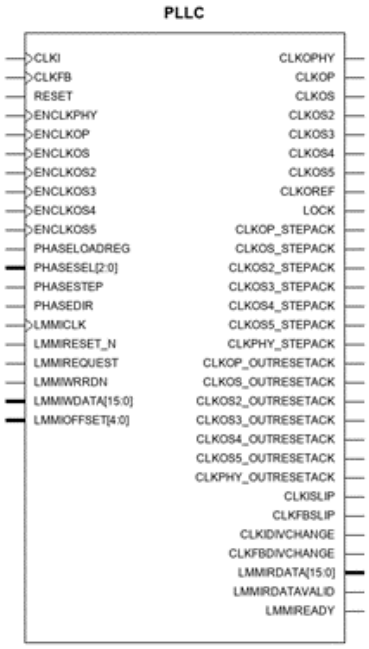


Table 564: Input Ports

Name	Range	Type	Active State	Description
CLKI	1	Clock	Active high	Reference Clock
CLKFB	1	Clock	Active high	Feedback Clock
RESET	1	Reset	Active high	PLL Reset
ENCLKPHY	1	Clock	Active high	Enable Physical clock output
ENCLKOP	1	Clock	Active high	Enable CLKOP (DIVA) clock output
ENCLKOS	1	Clock	Active high	Enable CLKOS (DIVB) clock output
ENCLKOS2	1	Clock	Active high	Enable CLKOS2 (DIVC) clock output
ENCLKOS3	1	Clock	Active high	Enable CLKOS3 (DIVD) clock output
ENCLKOS4	1	Clock	Active high	Enable CLKOS4 (DIVE) clock output
ENCLKOS5	1	Clock	Active high	Enable CLKOS5 (DIVF) clock output
PHASELOADREG	1	Control	Active high	Phase control, Initiate a divider output phase shift on negative edge of DYN_LOAD
PHASESEL	2:0	Control	Active high	Phase control to select which divider for phase-shift changes

Table 564: Input Ports (Continued)

Name	Range	Type	Active State	Description
PHASESTEP	1	Control	Active high	Phase control to rotate the VCO clock phase on the negative edge of DYN_ROTATE
PHASEDIR	1	Control	Active high	Phase control for the direction of DYN_ROTATE changes VCO phase. 0 -> phase rotates to later phase 1 -> Phase rotates to earlier phase
LMMICLK	1	Clock	Active high	LMMI clock
LMMIRESET_N	1	Reset	Active low	LMMI Reset
LMMIREQUEST	1	Data	Active high	LMMI Request
LMMIWRRDN	1	Data	Active high	LMMI write High /read Low
LMMIWDATA	15:0	Data	Active high	LMMI write data
LMMIOFFSET	4:0	Addr	Active high	LMMI address
ENCLKRES	1	Clock	Active high	Enable CLKRES clock output

Table 565: Output Ports

Name	Range	Type	Active State	Description
CLKOPHY	1	Clock	Active high	High speed PHYCLK
CLKOP	1	Clock	Active high	Primary CLKOP (DIVA) output
CLKOS	1	Clock	Active high	Secondary CLKOS (DIVB) output
CLKOS2	1	Clock	Active high	Secondary CLKOS2 (DIVC) output
CLKOS3	1	Clock	Active high	Secondary CLKOS3 (DIVD) output
CLKOS4	1	Clock	Active high	Secondary CLKOS4 (DIVE) output
CLKOS5	1	Clock	Active high	Secondary CLKOS5 (DIVF) output
CLKOREF	1	Clock	Active high	The reference CLK output
LOCK	1	Data	Active high	PLL lock indicator
CLKOP_STEPACK	1	Clock	Active high	CLKOP VCO PS status
CLKOS_STEPACK	1	Clock	Active high	CLKOS VCO PS status
CLKOS2_STEPACK	1	Clock	Active high	CLKOS2 VCO PS status
CLKOS3_STEPACK	1	Clock	Active high	CLKOS3 VCO PS status
CLKOS4_STEPACK	1	Clock	Active high	CLKOS4 VCO PS status
CLKOS5_STEPACK	1	Clock	Active high	CLKOS5 VCO PS status

Table 565: Output Ports (Continued)

Name	Range	Type	Active State	Description
CLKPHY_STEPACK	1	Clock	Active high	CLKPHY VCO PS status
CLKOP_OUTRESETACK	1	Clock	Active high	CLKOP reset status
CLKOS_OUTRESETACK	1	Clock	Active high	CLKOS reset status
CLKOS2_OUTRESETACK	1	Clock	Active high	CLKOS2 reset status
CLKOS3_OUTRESETACK	1	Clock	Active high	CLKOS3 reset status
CLKOS4_OUTRESETACK	1	Clock	Active high	CLKOS4 reset status
CLKOS5_OUTRESETACK	1	Clock	Active high	CLKOS5 reset status
CLKPHY_OUTRESETACK	1	Clock	Active high	CLKPHY reset status
CLKISLIP	1	Clock	Active high	Reference cycle slip output
CLKFBSLIP	1	Clock	Active high	Feedback cycle slip output
CLKIDIVCHANGE	1	Clock	Active high	Reference divider NR count change enable output
CLKFBDIVCHANGE	1	Clock	Active high	Feedback divider NF count change enable output
LMMIRDATA	15:0	Data	Active high	LMMI read data
LMMIRDATAVALID	1	Data	Active high	LMMI data valid
LMMIREADY	1	Data	Active high	LMMI Ready
CLKRES	1	Clock	Active high	CLKRES output
CLKRES_STEPACK	1	Clock	Active high	CLKRES VCO PS status
CLKRES_OUTRESETACK	1	Clock	Active high	CLKRES reset status

Table 566: Parameters

Name	Values	Description
FCLKI	"100" (default)	None
FVCO	"1600 MHz" - “4000 MHz”	The actual FVCO value is not user defined. In the Lattice Avant PLL Module configuration interface, when Enable Dynamic Phase Ports is checked, a Use Phase Advance option is added. When this is enabled, the maximum VCO frequency is reduced. Otherwise, the value remains at 4000 MHz.
CLKOP_OUT_SEL	"DIVA" (default)	Select output to CLKOP 0: DIVA: 1: CLKI
CLKOS_OUT_SEL	"DIVB" (default)	Select output to CLKOS0: DIVB: 1: CLKI
CLKOS2_OUT_SEL	"DIVC" (default)	Select output to CLKOS20: DIVC: 1: CLKI

Table 566: Parameters (Continued)

Name	Values	Description
CLKOS3_OUT_SEL	"DIVD" (default)	Select output to CLKOS30: DIVD: 1: CLKI
CLKOS4_OUT_SEL	"DIVE" (default)	Select output to CLKOS40: DIVE: 1: CLKI
CLKOS5_OUT_SEL	"DIVF" (default)	Select output to CLKOS50: DIVF: 1: CLKI
CLKPHY_OUT_SEL	"DIVPHY" (default)	Select output to CLKPHY0: DIVPHY: 1: CLKI
CLKRES_OUT_SEL	"DIVRES" (default)	Select output of divider (DIVRES) or reference clock (CLKI) to CLKRES
SYNC_CLKOP	"DISABLED" (default)	Enable output(s) sync with CLKOP0: disabled; 1: enabled
PHASE_SOURCE	"STATIC" (default)	1b1 select DYN (CIB signals) for dynamic phase shift
STATIC_PHASE_SEL	"CLKOP" (default)	To select which output clocks is phase shifted
STATIC_PHASE_LOADREG	"NO" (default)	lmmi equivalent of CIB load signal
STATIC_VCO_PHASE_STEP	"NO" (default)	Trigger for VCO phase rotation, 1: YES; 0: NO
STATIC_VCO_PHASE_DIR	"DELAYED" (default)	VCO phase rotation direction, 1b0 for phase lagging0 -> phase rotates to later phase (lagging, delayed)1 -> Phase rotates to earlier phase (leading, advanced)
CLKOP_FPHASE	"1" (default)	VCO phase A step control
CLKOS_FPHASE	"1" (default)	VCO phase B step control
CLKOS2_FPHASE	"1" (default)	VCO phase C step control
CLKOS3_FPHASE	"1" (default)	VCO phase D step control
CLKOS4_FPHASE	"1" (default)	VCO phase E step control
CLKOS5_FPHASE	"1" (default)	VCO phase F step control
CLKPHY_FPHASE	"1" (default)	VCO phase PHY step control
CLKRES_FPHASE	"1" (default)	VCO phase RES step control
CLKOP_CPHASE	"1" (default)	Delay A section output DELA VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-divider phase-shift)
CLKOS_CPHASE	"1" (default)	Delay B section output DELB VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-DELAYider phase-shift)
CLKOS2_CPHASE	"1" (default)	Delay C section output DELC VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-DELAYider phase-shift)
CLKOS3_CPHASE	"1" (default)	Delay D section output DELD VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-DELAYider phase-shift)

Table 566: Parameters (Continued)

Name	Values	Description
CLKOS4_CPHASE	"1" (default)	Delay E section output DELE VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-DELAYider phase-shift)
CLKOS5_CPHASE	"1" (default)	Delay F section output DELF VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-divider phase-shift)
CLKOPHY_CPHASE	"1" (default)	Delay F section output DELPHY VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-divider phase-shift)
CLKRES_CPHASE	"1" (default)	Delay RES section output DELRES VCO clock cycles with respect to VCO phase 0, or REFBUF if in VCO bypass mode (post-divider phase-shift)
FAST_LOCK	"ENABLED" (default)	Fast lock enable0: disabled; 1: enabled
LOSS_LOCK_DETECTION	"ENABLED" (default)	1b1 enable cycle slip detection
CLKI_DIV	"8" (default)	Reference clock divider; NR=CLKR[5:0]+1, CLKR[0] is LSB
CLKI_SEL	"REFMUX0" (default)	Static setting (Immi version) of refin_sel (CIB) from REFMUX0 or REFMUX1
CLKFB_DIV	"2" (default)	Feedback clock divider; CLKF[25:14]
FRACTIONAL_FBK	"0b00000000000000" (default)	Precise & SSC fractional multiplication
CLKFB_PATH	"EXTERNAL" (default)	Feedback path0: external feedback1: the internal feedback is selected
EXT_FB_DELAY	"0b00" (default)	Determine the output enable delayed time for external feedback mode
LOOP_BW	"0b000000000000" (default)	Loop BW control.: NB=BWADJ[11:0]+1,BWADJ[0] is LSB
CLKV_SSC_SLOPE	"0b00000000000000000000000000000" (default)	SSC Slope multiplier
CLKS_SSC_RATE	"0b000000000000" (default)	SSC rate multiplier
SCC_SS	"DISABLED" (default)	Enable or disable the Spread Spectrum0: disabled; 1: enabled
SCC_FRACTIONAL	"DISABLED" (default)	Enable fractional accumulation. 0: disabled; 1: enabled
CLKOP_DIV	"8" (default)	CLKOP Divider Setting, BFD bit setting is user primitive DIV bit stting = value-1
CLKOS_DIV	"8" (default)	CLKOS Divider Setting
CLKOS2_DIV	"8" (default)	CLKOS2 Divider Setting
CLKOS3_DIV	"8" (default)	CLKOS3 Divider Setting
CLKOS4_DIV	"8" (default)	CLKOS4 Divider Setting

Table 566: Parameters (Continued)

Name	Values	Description
CLKOS5_DIV	"8" (default)	CLKOS5 Divider Setting
CLKPHY_DIV	"1" (default)	CLKPHY Divider Setting
INT_CLK7_DIV	"8" (default)	CLK7 Divider Setting
CLKRES_DIV	"8" (default)	CLKRES Divider Setting
SATURATION	"ENABLED" (default)	Saturation behavior enable0: disabled; 1: enabled
EN_PLL	"DISABLED" (default)	1b1 to select GPLL IP
EN_PLLRESET	"DISABLED" (default)	1b1 to enable pllreset control (cib port)
EN_CLKOP	"YES" (default)	Enable CLKOP (DIVA) clock output 0: DISABLED (NO); 1: ENABLED (YES)
EN_CLKOS	"YES" (default)	Enable CLKOS (DIVB) clock output 0: DISABLED; 1: ENABLED
EN_CLKOS2	"YES" (default)	Enable CLKOS2 (DIVC) clock output 0: DISABLED; 1: ENABLED
EN_CLKOS3	"YES" (default)	Enable CLKOS3 (DIVD) clock output 0: DISABLED; 1: ENABLED
EN_CLKOS4	"YES" (default)	Enable CLKOS4 (DIVE) clock output 0: DISABLED; 1: ENABLED
EN_CLKOS5	"YES" (default)	Enable CLKOS5 (DIVF) clock output0: DISABLED; 1: ENABLED
EN_CLKPHY	"YES" (default)	Enable CLKPHY (DIVY) clock output 0: DISABLED; 1: ENABLED
EN_CLKRES	"NO" (default)	Enable CLKRES clock output
EN_CLKOP_OUT	"OFF" (default)	Output CLKOP 0: ON: 1: OFF
EN_CLKOS_OUT	"OFF" (default)	Output CLKOS0: ON;1: OFF
EN_CLKOS2_OUT	"OFF" (default)	Output CLKOS20: ON:1: OFF
EN_CLKOS3_OUT	"OFF" (default)	Output CLKOS30: ON; 1: OFF
EN_CLKOS4_OUT	"OFF" (default)	Output CLKOS40: ON; 1: OFF
EN_CLKOS5_OUT	"OFF" (default)	Output CLKOS50: ON; 1: OFF
EN_CLKPHY_OUT	"OFF" (default)	Output CLKPHY0: ON; 1: OFF
EN_CLKRES_OUT	"OFF" (default)	Output CLKRES: ON; 1: OFF
CONFIG_WAIT_FOR_LOCK	"DISABLED" (default)	Wait for PLL lock during configuration

PLLREFCS

PLL Dynamic Reference Clock Switching

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

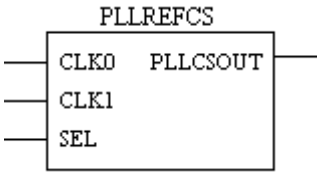


Table 567: Input Ports

Name	Range	Description
CLK0		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.
CLK1		
SEL		Selects which signal goes to the input clock MUX.

Table 568: Output Ports

Name	Range	Description
PLLCSOUT		

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.

PLLREFCSA

2:1 mux with dynamic select

Architectures supported:

- LAV-AT-ES1
- LN2-CT

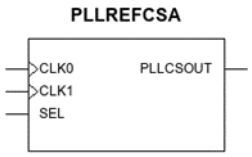


Table 569: Input Ports

Name	Range	Type	Active State	Description
CLK0	1	Clock	Rising Edge	CLK0 input
CLK1	1	Clock	Rising Edge	CLK1 input
SEL	1	Control	High	Select the CLK0 (SEL=0) and CLK1 (SEL=1)

Table 570: Output Ports

Name	Range	Type	Active State	Description
PLLCSOUT	1	Clock		Clock output

RGB

RGB LED drive module, containing three open drain I/O pins for RGB LED outputs.

Architectures supported:

- iCE40 UltraPlus

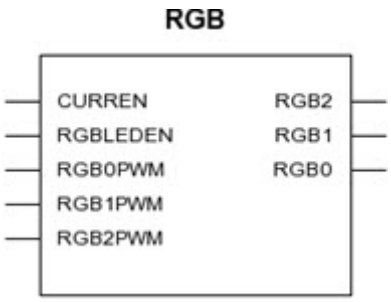


Table 571: Input Ports

Name	Range	Description
CURREN		Enable reference current to IR drivers. Enabling the drivers takes 100us to reach a stable reference current value. Active high.
RGBLEDEN		Enable the RGB driver. Active high.
RGB0PWM		Input data to drive RGB0 LED pin.
RGB1PWM		Input data to drive RGB1 LED pin.
RGB2PWM		Input data to drive RGB2 LED pin.

Table 572: Output Ports

Name	Range	Description
RGB2		RGB2 LED output.
RGB1		RGB1 LED output.
RGB0		RGB0 LED output.

Table 573: Parameters

Name	Value	Description
CURRENT_MODE	"0" (default), "1" (0 = full current, 1 = half current)	
RGB0_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	

Table 573: Parameters

Name	Value	Description
RGB1_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	
RGB2_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	

ATTRIBUTES: syn_black_box black_box_pad_pin="RGB2,RGB1,RBG0"

RGB1P8V

RGB LED drive module, containing three open drain I/O pins for RGB LED outputs. Trim bits are driven from Fabric.

Architectures supported:

- iCE40 UltraPlus

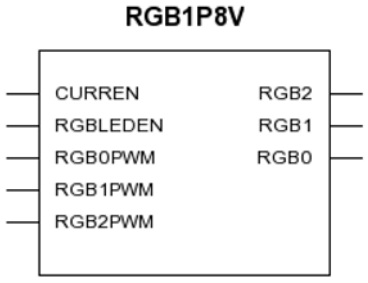


Table 574: Input Ports

Name	Range	Description
CURREN		Enable reference current to IR drivers. Enabling the drivers takes 100us to reach a stable reference current value. Active high.
RGBLEDEN		Enable the RGB driver. Active high.
RGB0PWM		Input data to drive RGB0 LED pin.
RGB1PWM		Input data to drive RGB1 LED pin.
RGB2PWM		Input data to drive RGB2 LED pin.

Table 575: Output Ports

Name	Range	Description
RGB2		RGB LED output.
RGB1		RGB LED output.
RGB0		RGB LED output.

Table 576: Parameters

Name	Value	Description
CURRENT_MODE	"0" (default), "1" (0 = full current, 1 = half current)	
RGB0_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	

Table 576: Parameters (Continued)

Name	Value	Description
RGB1_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	
RGB2_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111" (0'b000000 = 0mA, 0'b000001 = 4mA, 0'b000011 = 8mA, 0'b000111 = 12mA, 0'b001111 = 16mA, 0'b011111 = 20mA, 0'b111111 = 24mA. Divide by 2 for half current mode)	

ATTRIBUTES: syn_black_box black_box_pad_pin="RGB2,RGB1,RBG0"

RGB_CORE

RGB LED drive module containing three open drain I/O pins for RGB LED outputs.

Architectures supported:

- iCE40 UltraPlus

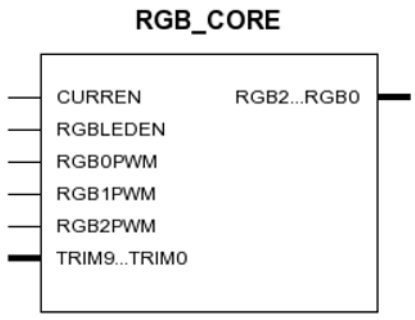


Table 577: Input Ports

Name	Range	Description
CURREN		Enable reference current to IR drivers. Enabling the drivers takes 100us to reach a stable reference current value. Active high.
RGBLEDEN		Enable the RGB driver. Active high.
RGB0PWM		Input data to drive RGB0 LED pin.
RGB1PWM		Input data to drive RGB1 LED pin.
RGB2PWM		Input data to drive RGB2 LED pin.
TRIM9…TRIM0		Trim bit ports to trim output PWM signals. Pre-set value.

Table 578: Output Ports

Name	Range	Description
RGB2…RGB0		RGB LED outputs.

Table 579: Parameters

Name	Value	Description
CURRENT_MODE	"0" (default), "1"	
RGB0_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111"	
RGB1_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111"	
RGB2_CURRENT	"0'b000000" (default), "0'b000001", "0'b000011", "0'b000111", "0'b001111", "0'b011111", "0'b111111"	
FABRIC_TRIME	"DISABLE" (default), "ENABLE"	

SAME AS: SB_RGBA_DRV

RGBPWM

Generates the PWM signals for the RGB LED drivers.

Architectures supported:

- iCE40 UltraPlus

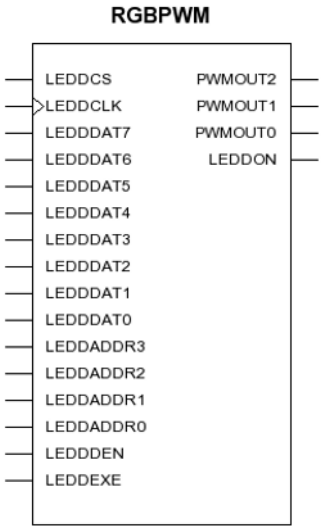


Table 580: Input Ports

Name	Range	Description
LEDDCS		Chip select.
LEDDCLK		Write clock.
LEDDDAT7...LEDDDAT0		Data input.
LEDDADDR3...LEDDADDR0		Data address.
LEDDDEN		Data enable input to indicate data and address are stable.
LEDDEXE		Enable the IP to run the blinking sequence. When it is low, the sequence stops at the nearest OFF state.

Table 581: Output Ports

Name	Value	Description
PWMOUT2...PWMOUT0		PWM outputs.
LEDDON		Indicates the LED is on.

SAME AS: SB_LEDDA_IP

SEDC

Interface for Soft Error Detection and Correction Functionality

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

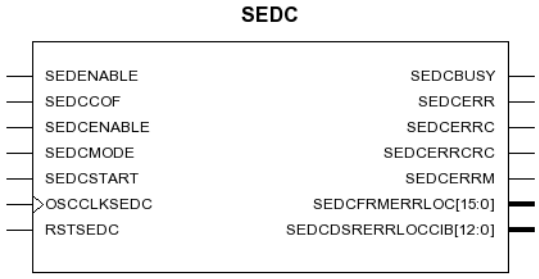


Table 582: Input Ports

Name	Range	Description
SSEDENABLE		
SEDCCOF		
SEDCENABLE		
SEDCMODE		

Table 582: Input Ports (Continued)

Name	Range	Description
SEDCSTART		
OSCCLKSEDC		
RSTSEDC		

Table 583: Output Ports

Name	Range	Description
SEDCBUSY		
SEDCERR		
SEDCERRC		
SEDCERRCRC		
SEDCERRM		
SEDCFRMERRLOC	15:0	
SEDCDSRERRLOCCIB	12:0	

Table 584: Parameters

Name	Values	Description
SEDCEN	"DIS" (default) "EN"	

SEDCA

Soft Error Detection and Correction

Architectures supported:

- LAV-AT-ES1

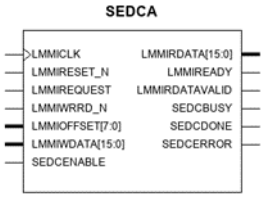


Table 585: Input Ports

Name	Range	Description
LMMICLK	1	Clock signal from fabric
LMMIRESET_N	1	Resets the LMM interface logic. Does not change the register contents or reset the internals of the Hard IP block
LMMIREQUEST	1	Start LMMI transaction
LMMIWRRD_N	1	Write = HIGH, Read = LOW
LMMIOFFSET	7:0	Register offset within the slave, starting at offset 0
LMMIWDATA	15:0	Write Data
SEDCENABLE	1	Start the SEDC engine

Table 586: Output Ports

Name	Range	Description
LMMIRDATA	15:0	Read Data
LMMIREADY	1	Slave is ready to start new transaction. Wait states inserted by holding signal low
LMMIRDATAVALID	1	Read transaction is complete
SEDCBUSY	1	SEDC busy status
SEDCDONE	1	SEDC done status
SEDCERROR	1	SEDC error detected

Table 587: Parameters

Name	Values	Description
CLK_DIV	2 (default) 256	SED Clock Divider based on OSC Frequency 450Mhz
GSR	ENABLED (default) DISABLED	ENABLED: 0 DISABLED: 1

SEIO18

Single Ended I/O block for 1.8V

Architectures supported:

- LFD2NX
- LFMXO5D
- LFMXO5-25
- LIFCL
- UT24C

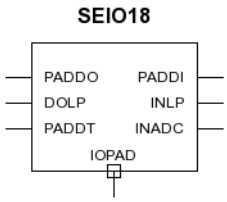


Table 588: Input Ports

Name	Description
PADDO	Data output from fabric to IO pad
DOLP	Data output for MIPI low-power mode
PADDT	Data tri-state control; high = tri-state

Table 589: Output Ports

Name	Description
PADDI	Data input from IO pad to fabric
INLP	Data input for MIPI low-power mode
INADC	Output to ADC

Table 590: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

Table 591: Parameters

Name	Values	Description
PULLMODE	"DOWN" (default) "NONE" "UP" "KEEPER" "FAILSAFE"	Enables pullup and pulldown mode
ENADC_IN	"DISABLED" (default) "ENABLED"	Enables pad A path to adc block
MIPI	"DISABLED" (default) "ENABLED"	Enable MIPI_DPHY IO TYPE

SEIO18A

Single Ended I/O block for 1.8V

Architectures supported:

- LFCPNX
- LFMXO5T
- LFMXO5TD
- UT24CP

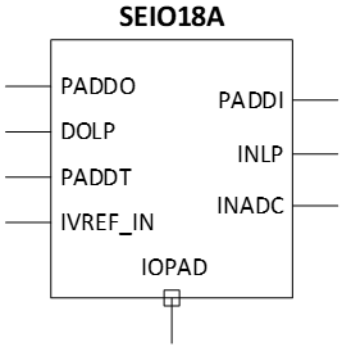


Table 592: Input Ports

Name	Description
PADDO	HS Data out to PAD
DOLP	LP Data out to PAD

Table 592: Input Ports

Name	Description
PADDT	Tri-state input signals to IO A, 0: driver output enabled 1: diver enabled to tri-state(high-Z) output
IVREF_IN	Internal Receiver Reference Voltage Input. Each bank will support 3 VREF analog input voltages for flexibility to receive three different referenced input levels on the same bank. VREFs are used to set the threshold for the referenced input receiver. Each referenced receiver has a programmable option to select VREF1/2 and a internal VREF.

Table 593: Output Ports

Name	Description
PADDI	Data in
INLP	MIPI LP RX output
INADC	Output to ADC through switch from PAD

Table 594: Bidirectional Ports

Name	Description
IOPAD	A virtual port, which is connected to the external pad.

Table 595: Parameters

Name	Values	Description
PULLMODE	"DOWN" (default) "NONE" "UP"	Pull mode
ENADC_IN	"DISABLED" (default) "ENABLED"	Enables pad A path to adc block
MIPI	"DISABLED" (default) "ENABLED"	MIPI usage

SEIO33

Single Ended I/O Block for 3.3V

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

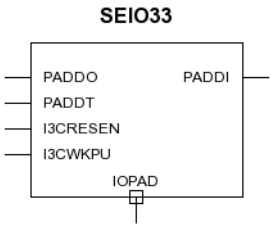


Table 596: Input Ports

Name	Description
PADDO	Data output from fabric to IO pad
PADDT	Data tri-state control; high = tri-state
I3CRESEN	I3C resistor enable
I3CWKPU	I3C weak pullup enable

Table 597: Output Ports

Name	Description
PADDI	Data input from IO pad to fabric

Table 598: Bidirectional Ports

Name	Description
IOPAD	IO Pad connection

Table 599: Parameters

Name	Values	Description
PULLMODE	"DOWN" (default)	Pull down
	"NONE"	No pull up/down
	"UP"	Pull up
	"KEEPER"	Retain the last state if bus connected to this IO goes to tri-state.
	"FAILSAFE"	For differential IOs, uses a weak pullup on the "true" pad and a weak pulldown on the "complement" pad to give a determinate value when undriven.

SGMIICDR

SGMII stands for Serial Gigabit Media Independent Interface, widely used for the interface to a discrete Ethernet PHY chip, it is defined as a 1.25 Gbps DDR interface with both clock and data pairs.

Note:

You should not try to use this primitive directly. Instead, use the related IP in IP Catalog. The following information is for reference only.

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40
- LIFCL-17
- UT24C
- UT24CP

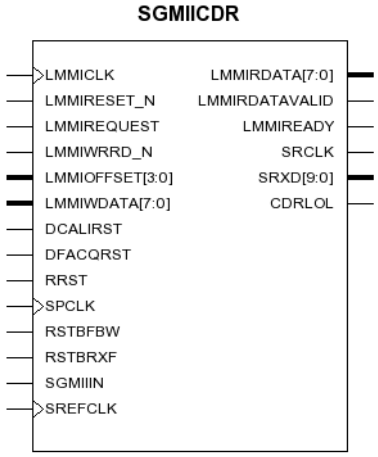


Table 600: Input Ports

Name	Range	Description
LMMICLK	1	Clock for LMMI interface
LMMIRESET_N	1	LMMI Reset
LMMIREQUEST	1	LMMI request
LMMIWRRD_N	1	LMMI write-high/read-low from fabric
LMMIOFFSET	3:0	LMMI Offset
LMMIWDATA	7:0	LMMI Write Data
DCALIRST	1	Reset signal to trigger the DCO calibration process when it is necessary
DFACQRST	1	Reset signal to trigger the DCO frequency acquisition process when it is necessary
RRST	1	Reset for the SGMII CDR IP
SPCLK	1	125MHz Fabric clock
RSTBFBW	1	FIFO write reset
RSTBRXF	1	FIFO read reset
SGMIIIN	1	125MHz reference clock
SREFCLK	1	125MHz reference clock

Table 601: Output Ports

Name	Range	Description
LMMIRDATA	7:0	Read data
LMMIRDATAVALID	1	Read transaction is complete and LMMIRDATA contains valid data
LMMIREADY	1	LMMI ready signal
SRCLK	1	125MHz recovered byte clock to Fabric
SRXD	9:0	10-bit parallel data to Fabric
CDRLOL	1	CDR loss of lock

Table 602: Parameters

Name	Values	Description
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
DCOITUNE4LSB	"0_PERCENT" (default) "15_PERCENT" "34_PERCENT" "60_PERCENT" "NEG_34_PERCENT" "NEG_28_PERCENT" "NEG_20_PERCENT" "NEG_12_PERCENT"	DACLSB current tuning.
DCOCTLGI	"0_PERCENT" (default) "15_PERCENT" "34_PERCENT" "60_PERCENT" "NEG_34_PERCENT" "NEG_28_PERCENT" "NEG_20_PERCENT" "NEG_12_PERCENT"	DCO global current control.
DCOSTEP	"100_PERCENT" (default) "17_PERCENT" "NEG_34_PERCENT" "NEG_17_PERCENT"	Calibration step current adjust.
DCOCALDIV	"100_PERCENT" (default) "NEG_10_PERCENT" "NEG_17_PERCENT" "NEG_25_PERCENT" "60_PERCENT" "42_PERCENT" "25_PERCENT" "11_PERCENT"	Calibration Input current adjust.

Table 602: Parameters (Continued)

Name	Values	Description
DCOIOSTUNE	"0_PERCENT" (default) "10_PERCENT" "NEG_40_PERCENT" "NEG_20_PERCENT" "39_PERCENT" "49_PERCENT" "20_PERCENT" "30_PERCENT"	DCO offset current tuning.
DCOFLTDAC	"80MHZ" (default) "51MHZ" "181MHZ" "103MHZ"	DAC output current filter option.
DCOSTARTVAL	"NOMINAL" (default) "12P5_PERCENT" "NEG_50_PERCENT" "NEG_25_PERCENT" "100_PERCENT" "125_PERCENT" "50_PERCENT" "75_PERCENT"	Calibration start band current adjust.
DCONUOFLSB	"NEG_60_PERCENT" (default) "NEG_80_PERCENT" "NEG_20_PERCENT" "NEG_40_PERCENT" "20_PERCENT" "0_PERCENT" "60_PERCENT" "40_PERCENT"	Up/down current option.
RPWDNB	"POWER_UP" (default) "POWER_DOWN"	SGMII CDR IP power-down control.
CDR_CNT4SEL	"BYPASSED" (default) "BY_2" "BY_3" "BY_4"	Selection for CDR random walk divider4.
DCOITUNE	"100_PERCENT" (default) "NEG_33_PERCENT" "NEG_50_PERCENT" "NEG_60_PERCENT"	ICO current adjustment.
BAND_THRESHOLD	0'b000000 (default)	Stop point for DCO band calibration.
AUTO_FACQ_EN	"ENABLED" (default) "DISABLED"	Enable automatic DCO frequency acquisition.
AUTO_CALIB_EN	"ENABLED" (default) "DISABLED"	Enable automatic DCO band calibration.
CDR_LOL_SET	"1000_PPM" (default) "2000_PPM" "4000_PPM" "300_PPM"	CDR loss of lock setting.

Table 602: Parameters (Continued)

Name	Values	Description
FC2DCO_FLOOP	"DISABLED" (default) "ENABLED"	Force DCO lock to the frequency loop.
FC2DCO_DLOOP	"DISABLED" (default) "ENABLED"	Force DCO lock to the data loop.
CALIB_TIME_SEL	"24_CYC" (default) "26_CYC" "28_CYC" "210_CYC"	Time to wait for DCO to be stable.
CALIB_CK_MODE	"BY_2" (default) "BYPASSED"	DCO calibration clock mode.
BAND_CALIB_MODE	"256_FDBK_CLK_CYC" (default) "512_FDBK_CLK_CYC"	Waiting time to move to the next band.
REG_BAND_SEL	0'b00000 (default)	DCO band selection.
REG_BAND_OFFSET	0'b0000 (default)	DCO band offset control.
REG_IDAC_SEL	0'b00000000 (default)	Manual idac selection when reg_idac_en = 1.
LB_CTL	"DISABLED" (default) "DATA_OUT" "CLK_OUT"	Loopback control for testing lb_ctl.
REG_IDAC_EN	"DISABLED" (default) "ENABLED"	Enable manual idac selection.
ATDCFG	"0_PS" (default) "92P5_PS" "160_PS" "248_PS" "NEG_565_PS" "NEG_475P5_PS" "NEG_407P5_PS" "NEG_343P5_PS"	Adjust atd pulse width (unit width 2 inv delay).
ATDDLY	"0_PS" (default) "62_PS" "78P5_PS" "NEG_12_PS"	Delay atd pulse (unit width 2 inv delay).
BDAVOID_ENB	"ENABLED" (default) "DISABLED"	DAC 10b boundary code avoidance mode: 0 = enable, 1 = disable.
BYPASSATD	"NOT_BYPASS" (default) "BYPASS"	Bypass atd.
DCOIUPDNX2	"1X" (default) "2X"	Up/down current option.
IDAC_EN	"DISABLED" (default) "ENABLED"	Enable manual idac selection.

Table 602: Parameters (Continued)

Name	Values	Description
FB_CLK_DIV	0'b010 (default) 000	Feedback divider
EN_RECALIB	"ENABLED" (default) "DISABLED"	Enable re-calibration.

SGSR

Synchronous GSR Primitive

Architectures supported:

- LAV-AT-ES1
- LN2-CT



Table 603: Input Ports

Name	Range	Description
GSR_N	1	User GSR input
CLK	1	User clock for sync mode

Note:

The SGSR primitive needs to be instantiated in the user design.

SP16K

16 kb Single Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX

- LFMXO5
- LIFCL
- UT24C
- UT24CP

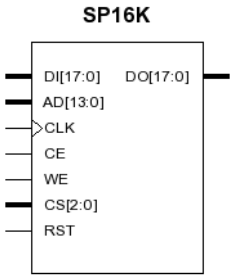


Table 604: Input Ports

Name	Range	Description
DI	17:0	Data in
AD	13:0	Address
CLK		Clock
CE		Clock enable
WE		Write enable
CS	2:0	Chip select
RST		Output register reset

Table 605: Output Ports

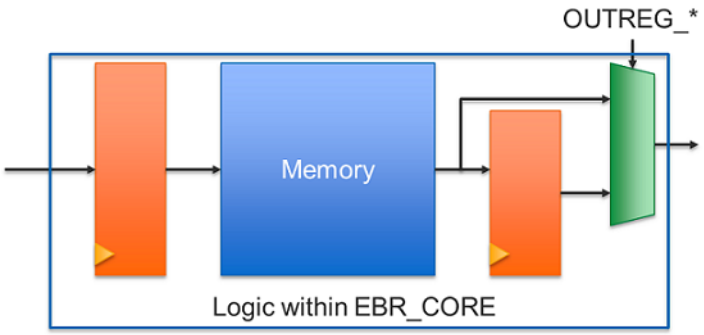
Name	Range	Description
DO	17:0	Data out

Table 606: Parameters

Name	Values	Description
DATA_WIDTH	"X18" (default) "X9" "X4" "X2" "X1"	Data width
OUTREG	"BYPASSED" (default) "USED"	Enable output register
RESETMODE	"SYNC" (default) "ASYNC"	Reset mode
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
INITVAL_00...INITVAL_3F	Hex string, 320 bits, must include the "0x" prefix	EBR initialization data
CSDECODE	000 (default) 001 010 011 100 101 110 111	Chip select active setting
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release mode
INIT_DATA	"STATIC" (default) "DYNAMIC" "NO_INIT"	

Note:

All inputs (except the register control signals) and all outputs are registered in the following manner:



Port assignments for different data widths will follow the table below:

Data Width	Input Data	Output Data	Address (MSB to LSB)
16Kx1	DI[0]	DO[0]	AD[13:0]
8Kx2	DI[1:0]	DO[1:0]	AD[13:1]
4Kx4	DI[3:0]	DO[3:0]	AD[13:2]
2Kx9	DI[8:0]	DO[8:0]	AD[13:3]
1Kx18	DI[17:0]	DO[17:0]	AD[13:4]

For data widths that do not use all available address bits, the unused bits **must** be tied high for proper functionality.

The initial value of the memory may be given in the INITVAL property. The format is little endian, with bytes (up to 9 bits) packed as follows, where byte[n] is the most significant byte and byte[0] is the least significant byte:

INITVAL_XX = { {0, byte[n]}, {0, byte[n-1]}, … {0, byte[1]}, {0, byte[0]} }

SP16KA

16 Kb Single Port Block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

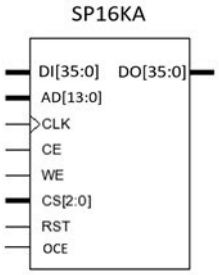


Table 607: Input Ports

Name	Range	Type	Active State	Description
DI	35:0	Data	Active High	Data Inputs
AD	13:0	Addr	Active High	Address

Table 607: Input Ports (Continued)

Name	Range	Type	Active State	Description
CLK	1	Clock	Rising Edge	Clock Input
CE	1	Clock Enable	Active High	Clock Enable
OCE	1	Control	Active High	Clock Enable
WE	1	Control	Active High	Write Enable 1: Write 0: Read
CS	2:0	Control	Active High	Cascade Select
RST	1	Reset	Active High	Reset

Table 608: Output Ports

Name	Range	Type	Description
DO	35:0	Data	Data Output

Table 609: Parameters

Name	Values	Description
DATA_WIDTH	“X36” (default) "X18" "X9" "X4" "X2" "X1"	Data Width
OUTREG	"BYPASSED" (default) "USED"	Enable output register
RESETMODE	"SYNC" (default) "ASYNC"	Reset mode
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers

Table 609: Parameters (Continued)

Name	Values	Description
CSDECODE_A	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Chip select active setting
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release mode
INITVAL_00...INITVAL_3F	Hex string, 320 bits, must include the "0x" prefix	Initialization value
INIT_DATA	"STATIC" (default) "DYNAMIC"	Specifies if the initialization data (as defined by the INITVAL parameters) may be changed after the synthesis stage. When INIT_DATA=STATIC, it is assumed that the initialization data will not change after synthesis, so an optimization to reduce the bitstream size can be applied. When INIT_DATA=DYNAMIC it is assumed that the initialization data may change after synthesis, so the aforementioned optimization is disabled.

Note:

When two SP16KAs are instantiated in the design, one will use Port A, and the other Port B.

SP32K

32 Kb Single Port Block RAM

Architectures supported:

- LAV-AT-ES1
- LN2-CT

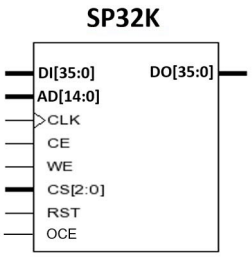


Table 610: Input Ports

Name	Range	Type	Active State	Description
DI	35:0	Data	Active High	Data Inputs
AD	14:0	Addr	Active High	Address
CLK	1	Clock	Rising Edge	Clock Input
CE	1	Clock Enable	Active High	Clock Enable
OCE	1	Control	Active High	Output Clock Register Enable
WE	1	Control	Active High	Write Enable 1: Write 0: Read
CS	2:0	Control	Active High	Cascade Select
RST	1	Reset	Active High	Reset

Table 611: Output Ports

Name	Range	Type	Description
DO	35:0	Data	Data Output

Table 612: Parameters

Name	Values	Description
DATA_WIDTH	"X36" (default) "X18" "X9" "X4" "X2" "X1"	Data width
OUTREG	"BYPASSED" (default) "USED"	Enable output register
WRITE_MODE_A	"NORMAL" (default) "WRITETHROUGH"	Write mode
RESETMODE	"SYNC" (default) "ASYNC"	Reset mode
GSR	"ENABLED" (default) "DISABLED"	Enable global set/reset for the output registers
CSDECODE	"000" (default) "001" "010" "011" "100" "101" "110" "111"	Chip select active setting
ASYNC_RST_RELEASE	"SYNC" (default) "ASYNC"	Reset release mode
INITVAL_00...INITVAL_7F	Hex string, 320 bits, must include the "0x" prefix	Initialization value
INIT_DATA	"STATIC" (default) "DYNAMIC"	Specifies if the initialization data (as defined by the INITVAL parameters) may be changed after the synthesis stage. When INIT_DATA=STATIC, it is assumed that the initialization data will not change after synthesis, so an optimization to reduce the bitstream size can be applied. When INIT_DATA=DYNAMIC it is assumed that the initialization data may change after synthesis, so the aforementioned optimization is disabled.

SP256K

Single Port RAM that can be configured in 16K x 16 mode. This block can be cascaded using logic implemented in fabric to create larger memories.

Architectures supported:

- iCE40 UltraPlus

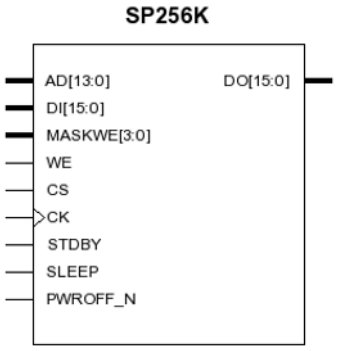


Table 613: Input Ports

Name	Range	Description
AD	13:0	This Address input port is used to address the location to be written during the write cycle and read during the read cycle.
DI	15:0	The data input bus used to write the data into the memory location specified by address input port during the write cycle.
MASKWE	3:0	The Bit Write feature where selective write to individual I/O's can be done using the Maskable Write Enable signals. Each bit masks a nibble of DI.
WE		When the Write Enable input is Logic High, the memory is in the write cycle. When the Write enable is Logic Low, the memory is in the Read Cycle.
CS		When the memory enable input is Logic High, the memory is enabled and read/write operations can be performed. When memory Enable input is logic Low, the memory is deactivated.
CK		This is the external clock for the memory.
STDBY		When this pin is active then memory goes into low leakage mode, there is no change in the output state.

Table 613: Input Ports (Continued)

Name	Range	Description
SLEEP		This pin shut down power to periphery and maintain memory contents. The outputs of the memory are pulled low.
PWROFF_N		This pin turns off the power to the memory core when it is driven low. There is no memory data retention when it is driven low. When PWROFF_N is driven high, the SPRAM is powered on.

Table 614: Output Ports

Name	Value	Description
DO	15:0	It outputs the contents of the memory location addressed by the Address Input signals.

SP512K

512Kb Single Port Block RAM

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

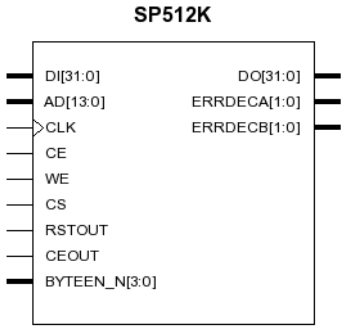


Table 615: Input Ports

Name	Range	Description
DI	31:0	Data in
AD	13:0	Address
CLK		Clock
CE		Clock enable
WE		Write enable
CS		Chip select
RSTOUT		Output register reset
CEOUT		Output register clock enable
BYTEEN_N	3:0	Write byte enable: 0=enable, 1=disable. BYTEEN_N[3] corresponds to the MSB of DI. BYTEEN_N[0] corresponds to the LSB.

Table 616: Output Ports

Name	Range	Description
DDO	31:0	Data out
ERRDECA	1:0	Port A 1- and 2-bit error flag.
ERRDECB	1:0	Port B 1- and 2-bit error flag.

Table 617: Parameters

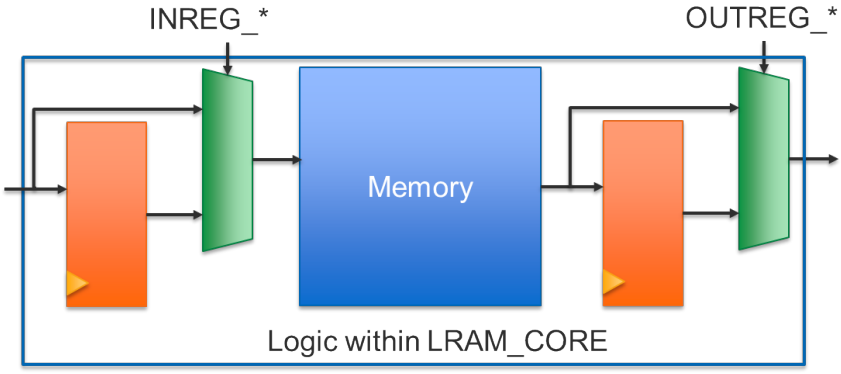
Name	Values	Description
OUTREG	"NO_REG" (default) "OUT_REG"	Register output
GSR	"GSR_EN" (default) "GSR_DIS"	Enable global set/reset
RESETMODE	"SYNC" (default) "ASYNC"	Reset sync/async control
INITVAL_00...INITVAL_7F	Hex string, 5120 bits, must include the "0x" prefix	

Table 617: Parameters

Name	Values	Description
ASYNC_RESET_RELEASE	"SYNC" (default) "ASYNC"	Reset release sync/async
ECC_BYTE_SEL	"BYTE_EN" "ECC_EN" (default)	

Note:

All inputs (except the register control signals) and all outputs are registered in the following manner:



SPI_B

Hard SPI interface

Architectures supported:

- iCE40 UltraPlus

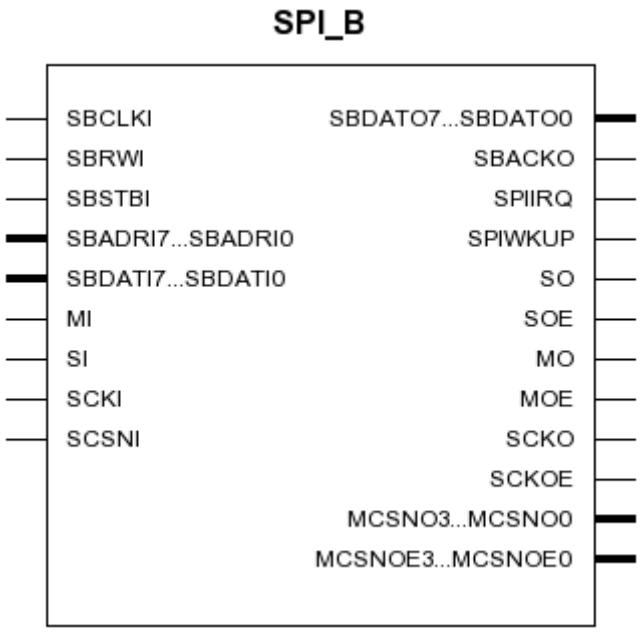


Table 618: Input Ports

Name	Range	Description
SBCLKI		System clock input.
SBRWI		System read/write input.
SBSTBI		Strobe signal.
SBADRI7...SBADRI0		System bus control registers address.
SBDATI7...SBDATI0		System data input.
MI		Master input from PAD.
SI		Slave input from PAD.
SCKI		Slave clock input from PAD.
SCSNI		Slave chip select input from PAD.

Table 619: Output Ports

Name	Range	Description
SBDATO7...SBDATO0		System data output.
SBACKO		System acknowledgment
SPIIRQ		SPI interrupt output

Table 619: Output Ports (Continued)

Name	Range	Description
SPIWKUP		SPI wake up from standby signal
SO		Slave output to PAD
SOE		Slave output enable to PAD. Active high
MO		Master output to PAD
MOE		Master output enable to PAD. Active high
SCKO		Slave clock output to PAD
SCKOE		Slave clock output enable to PAD
MCSNO3...MCSNO0		Master chip select output to PAD
MCSNOE3...MCSNOE0		Master chip select output enable to PAD

Table 620: Parameters

Name	Value	Description
BUS_ADDR74	"0’b0000" (default), "0’b0010". Fixed value. Upper left corner should be 0’b0000. Upper right corner should be 0’b0010. SBARDI[7:4] should match this value to activate the IP.	
SPI_CLK_DIVIDER	Ratio of SCBLKI/SCKO. "0" (default), "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58", "59", "60", "61", "62", "63".	
FREQUENCY_PIN_SBCLKI	SDC constraint entry on SBCLKI port of SPI. "NONE" (default).	

SPR16X4

Distributed Single Port RAM with Synchronous Write and Asynchronous Read

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

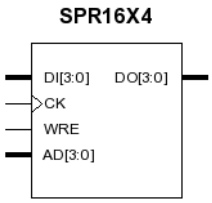


Table 621: Input Ports

Name	Range	Description
DI	3:0	Data in
AD	3:0	Read/write address
CK		Write clock
WRE		Write enable

Table 622: Output Ports

Name	Range	Description
DO	3:0	Data out

Table 623: Parameters

Name	Value	Description
INITVAL	"0x0000000000000000"	16-digit hex string, 64 bits total) (default: all zeros

SPR16X4A

Distributed Single Port RAM. Synchronous write, asynchronous read

Architectures supported:

- LAV-AT-ES1
- LN2-CT

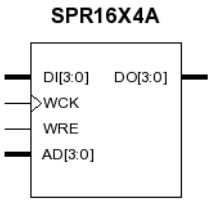


Table 624: Input Ports

Name	Range	Type	Description
DI	3:0	Data	Write data
WCK	1	Clock	Write clock
WRE	1	Control	Write enable
AD	3:0	Addr	Address

Table 625: Output Ports

Name	Range	Type	Description
DO	3:0	Data	Read data

Table 626: Parameters

Name	Values	Description
INITVAL	"0x0000000000000000" (default)	RAM initialization value

SPR32X2

Distributed Single Port RAM. Synchronous write, asynchronous read

Architectures supported:

- LAV-AT-ES1
- LN2-CT

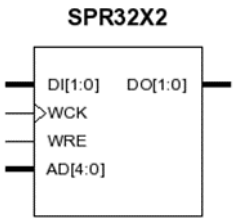


Table 627: Input Ports

Name	Range	Type	Description
DI	1:0	Data	Write data
WCK	1	Clock	Write clock
WRE	1	Control	Write enable
AD	4:0	Addr	Address

Table 628: Output Ports

Name	Range	Type	Description
DO	1:0	Data	Read data

Table 629: Parameters

Name	Values	Description
INITVAL	"0x0000000000000000" (default)	RAM initialization value

TSALLA

Global Tristate Interface

Architectures supported:

- LFCPNX

- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

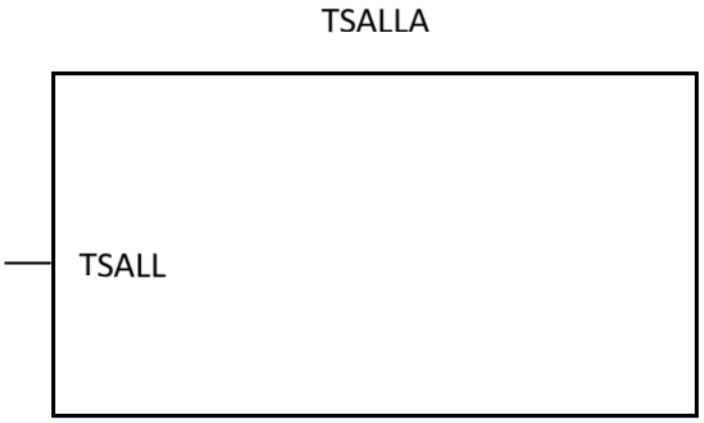


Table 630: Input Ports

Name	Range	Description
TSALL		

TSALLA is used to tristate buffers in your design. The TSALLA 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.

TSALLB

Tri-state all PIO outputs and bidirectional PIO

Architectures supported:

- LAV-AT-ES1
- LN2-CT

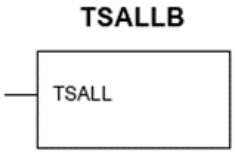


Table 631: Input Ports

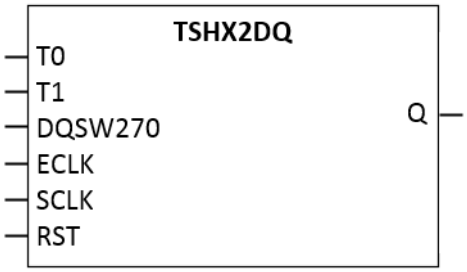
Name	Range	Description
TSALL	1	Tri-state control to all outputs and bidirectional PIO

TSHX2DQ

DQ tristate control for DDR2 & DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the tristate control for DQ data output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interface with x2 gearing.

Table 632: Input Ports

Name	Range	Description
T0, T1		Tristate input. T0 is output first, followed by T1.
DQSW270		Clock that is 90° ahead of the clock used to generate the DQS output.
ECLK		ECLK input.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset input.

Table 633: Output Ports

Name	Range	Description
Q		Tristate output

Table 634: Parameters

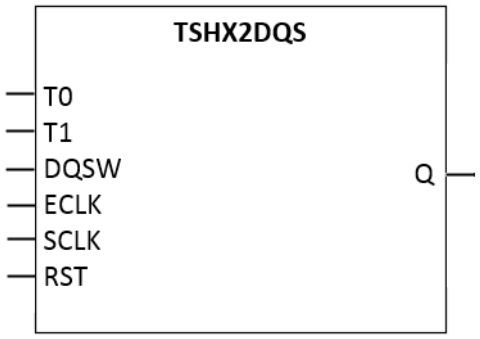
Name	Description	Values	Default
GSR	Enable global set/ reset.	"ENABLED" "DISABLED"	"ENABLED"

TSHX2DQS

DQS tristate control for DDR2 & DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the tristate control for DQS output.

Table 635: Input Ports

Name	Range	Description
T0, T1		Tristate input. T0 is output first, followed by T1.
DQSW		DQSW includes write leveling phase shift from ECLK.
ECLK		ECLK input.
SCLK		Primary clock input (divide-by-2 of ECLK).
RST		Reset input.

Table 636: Output Ports

Name	Range	Description
Q		Tristate output

Table 637: TSHX2DQS Parameters

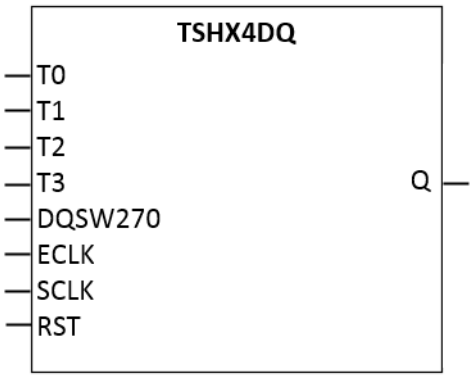
Name	Description	Values	Default
GSR	Enable global set/ reset.	"ENABLED" "DISABLED"	"ENABLED"

TSHX4DQ

DQ tristate control for DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the tristate control for DQ data output for DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces with x4 gearing.

Table 638: Input Ports

Name	Range	Description
T	3	Tristate input. T0 is output first, T3 last.
DQSW270		Clock that is 90° ahead of the clock used to generate the DQS output.
ECLK		ECLK input.
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset input.

Table 639: Output Ports

Name	Range	Description
Q		Tristate output

Table 640: Parameters

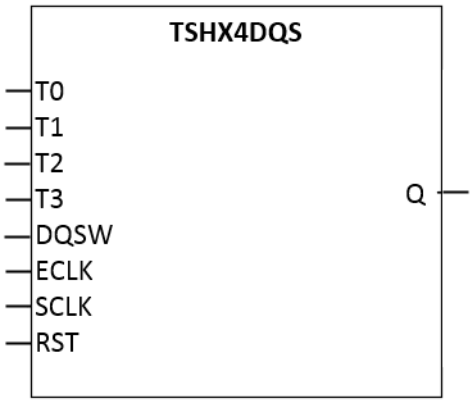
Name	Description	Values	Default
GSR	Enable global set/ reset.	"ENABLED" "DISABLED"	"ENABLED"

TSHX4DQS

DQS tristate control for DDR3 memory

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL-40,LIFCL-17
- UT24C
- UT24CP



This primitive is used to generate the tristate control for DQS output for DDR3/ DDR3L and LPDDR2/LPDDR3 memory interfaces with x4 gearing.

Table 641: Input Ports

Name	Range	Description
T	3:0	Tristate input. T0 is output first, T3 last.
DQSW		DQSW includes write leveling phase shift from ECLK.
ECLK		ECLK input.

Table 641: Input Ports

Name	Range	Description
SCLK		Primary clock input (divide-by-4 of ECLK).
RST		Reset input.

Table 642: Output Ports

Name	Range	Description
Q		Tristate output

Table 643: Parameters

Name	Description	Values	Default
GSR	Enable global set/ reset.	"ENABLED" "DISABLED"	"ENABLED"

UMXSPI

Controller SPI

Architectures supported:

- LAV-AT-ES1



Table 644: Input Ports

Name	Range	Description
SCLK		Primary clock input for OREG/ODDR
DS		Data Strobe Input
DQ_TRISTATE		Tristate Control for DQ PIOs
FABRIC_CS_N		Chip Select (Active Low) from fabric
LOAD_DLLDEL		Control for DLLDEL
MOVE_DLLDEL		Control for DLLDEL
DIRECTION_DLLDEL		Control for DLLDEL
CODE_DLLDEL	8:0	Code control for DLLDEL
FABRIC_DATA_OUT	15:0	Data output from fabric to DQ
LOAD_DELAYD		Control for DELAYD
DIRECTION_DELAYD		Control for DELAYD
MOVE_DELAYD		Control for DELAYD
CODE8_DELAYD		Code control for DELAYD

Table 644: Input Ports (Continued)

Name	Range	Description
CODE7_DELAYD		Code control for DELAYD
CODE6_DELAYD		Code control for DELAYD
CODE5_DELAYD		Code control for DELAYD
CODE4_DELAYD		Code control for DELAYD
CODE3_DELAYD		Code control for DELAYD
CODE2_DELAYD		Code control for DELAYD
CODE1_DELAYD		Code control for DELAYD
CODE0_DELAYD		Code control for DELAYD
LOAD_DELAYD_CK		Control for DELAYD of CK
DIRECTION_DELAYD_CK		Control for DELAYD of CK
MOVE_DELAYD_CK		Control for DELAYD of CK
CODE8_DELAYD_CK		Code control for CK delay buffer
CODE7_DELAYD_CK		Code control for CK delay buffer
CODE6_DELAYD_CK		Code control for CK delay buffer
CODE5_DELAYD_CK		Code control for CK delay buffer
CODE4_DELAYD_CK		Code control for CK delay buffer
CODE3_DELAYD_CK		Code control for CK delay buffer
CODE2_DELAYD_CK		Code control for CK delay buffer
CODE1_DELAYD_CK		Code control for CK delay buffer
CODE0_DELAYD_CK		Code control for CK delay buffer
LOAD_DELAYD_DS		Control for DS delay buffer
DIRECTION_DELAYD_DS		Control for DS delay buffer
MOVE_DELAYD_DS		Control for DS delay buffer
CODE8_DELAYD_DS		Code control for DS delay buffer
CODE7_DELAYD_DS		Code control for DS delay buffer
CODE6_DELAYD_DS		Code control for DS delay buffer
CODE5_DELAYD_DS		Code control for DS delay buffer
CODE4_DELAYD_DS		Code control for DS delay buffer
CODE3_DELAYD_DS		Code control for DS delay buffer
CODE2_DELAYD_DS		Code control for DS delay buffer
CODE1_DELAYD_DS		Code control for DS delay buffer

Table 644: Input Ports (Continued)

Name	Range	Description
CODE0_DELAYD_DS		Code control for DS delay buffer
RST		Reset of IDDRFIFO (DS & DQ)

Table 645: Output Ports

Name	Range	Description
CS_N		Chip Select (Active Low) output to external slave
CK		Clock generated from ODDR gearing of SCLK input
CFLAG_DLLDEL		Output flag of DLLDEL
FABRIC_DATA_IN	15:0	Data input from DQ to fabric
CFLAG_DELAYD0		Output flag for DELAYD instance 0
CFLAG_DELAYD1		Output flag for DELAYD instance 1
CFLAG_DELAYD2		Output flag for DELAYD instance 2
CFLAG_DELAYD3		Output flag for DELAYD instance 3
CFLAG_DELAYD4		Output flag for DELAYD instance 4
CFLAG_DELAYD5		Output flag for DELAYD instance 5
CFLAG_DELAYD6		Output flag for DELAYD instance 6
CFLAG_DELAYD7		Output flag for DELAYD instance 7
CFLAG_DELAYD_CK		Output flag for CK delay buffer
CFLAG_DELAYD_DS		Output flag for DS delay buffer
Q1_DS		Output of IDDRFIFO (DS)
Q0_DS		Output of IDDRFIFO (DS)

Table 646: Bidirectional Ports

Name	Range	Description
DQ	7:0	Bidirectional DQ bus

Table 647: Parameters

Name	Values	Description
XSPI_ID	"0" (default) "0-1024"	
GSR	"ENABLED" (default) "DISABLED"	Enable Global Set/Reset Control
DEL_ADJ	"PLUS" (default) "MINUS"	Static Delay Value Adjust
DEL_VALUE_DLLDELB	"0" (default) "0-511"	Static Delay Value
DEL_MODE	"USER_DEFINED" (default) "SCLK_ZEROHOLD" "SCLK_ALIGNED" "SCLK_CENTERED"	Delay Mode
DEL_SOURCE	"DEL_VALUE" (default) "DDRDLL"	Static Delay Value Source
DEL_VALUE_DELAYD	"0" (default) "0-511"	Static Delay Value
IO_TYPE	"LVCMOS33" (default) "LVCMOS25" "LVCMOS18" "LVCMOS12"	None

USXSPI

Target SPI

Architectures supported:

- LAV-AT-ES1

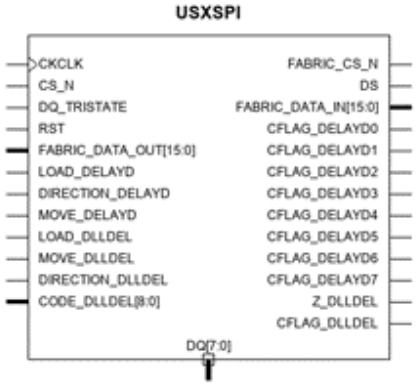


Table 648: Input Ports

Name	Range	Description
CKCLK		Octal SPI Slave Clock Input from WRIO sector
CS_N		Chip Select (Active Low) from external master
DQ_TRISTATE		Tristate Control for DQ PIOs
RST		IDDR & ODDR Reset
FABRIC_DATA_OUT	15:0	Data output from fabric to DQ
LOAD_DELAYD		Control for DELAYD
DIRECTION_DELAYD		Control for DELAYD
MOVE_DELAYD		Control for DELAYD
LOAD_DLLDEL		Control for DLLDEL
MOVE_DLLDEL		Control for DLLDEL
DIRECTION_DLLDEL		Control for DLLDEL
CODE_DLLDEL	8:0	Code control for DLLDEL

Table 649: Output Ports

Name	Range	Description
FABRIC_CS_N		Chip Select (Active Low) to fabric
DS		Data Strobe
FABRIC_DATA_IN	15:0	Data input from DQ to fabric

Table 649: Output Ports (Continued)

Name	Range	Description
CFLAG_DELAYD0		Output flag for DELAYD instance 0
CFLAG_DELAYD1		Output flag for DELAYD instance 1
CFLAG_DELAYD2		Output flag for DELAYD instance 2
CFLAG_DELAYD3		Output flag for DELAYD instance 3
CFLAG_DELAYD4		Output flag for DELAYD instance 4
CFLAG_DELAYD5		Output flag for DELAYD instance 5
CFLAG_DELAYD6		Output flag for DELAYD instance 6
CFLAG_DELAYD7		Output flag for DELAYD instance 7
Z_DLLDEL		Output of DLLDEL
CFLAG_DLLDEL		Output flag of DLLDEL

Table 650: Bidirectional Ports

Name	Range	Description
DQ	7:0	Bidirectional DQ bus

Table 651: Parameters

Name	Values	Description
XSPI_ID	"0" (default) "0-1024"	
GSR	"ENABLED" (default) "DISABLED"	Enable Global Set/Reset Control
DEL_ADJ	"PLUS" (default) "MINUS"	Static Delay Value Adjust
DEL_VALUE_DLLDELB	"0" (default) "0-511"	Static Delay Value
DEL_MODE	"USER_DEFINED" (default) "SCLK_ZEROHOLD" "SCLK_ALIGNED" "SCLK_CENTERED"	Delay Mode

Table 651: Parameters (Continued)

Name	Values	Description
DEL_SOURCE	"DEL_VALUE" (default) "DDRDLL"	Static Delay Value Source
DEL_VALUE_DELAYD	"0" (default) "0-511"	Static Delay Value
IO_TYPE	"LVCMOS33" (default) "LVCMOS25" "LVCMOS18" "LVCMOS12"	None

USB23

The software interface of USB2.0 & USB3.0 hard IP.

Architectures supported:

- LIFCL-33U

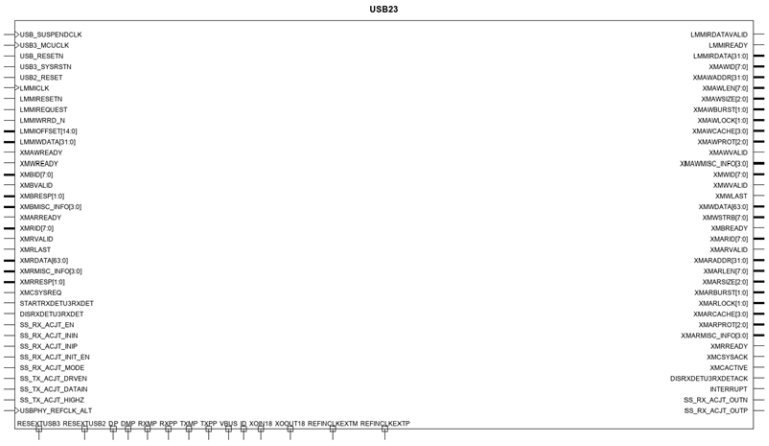


Table 652: Input Ports

Name	Range	Description
USB_SUSPENDCLK		Suspend clock for usb23 controller in LP mode
USB3_MCUCLK		MCU clock for USB3 phy register setting

Table 652: Input Ports (Continued)

Name	Range	Description
USB_RESETN		Global reset of usb23 controller from user
USB3_SYSRSTN		Global reset of usb3 phy from user
USB2_RESET		Global reset of usb2 phy from user
LMMICLK		Clock of lmmi bus
LMMIRESETN		Reset of lmmi bus
LMMIREQUEST		Request of lmmi bus
LMMIWRRD_N		Write or read of lmmi bus
LMMIOFFSET	14:0	Address of lmmi bus
LMMIWDATA	31:0	Data of lmmi bus
XMAWREADY		AXI bus for DMA
XMWREADY		AXI bus for DMA
XMBID	7:0	AXI bus for DMA
XMBVALID		AXI bus for DMA
XMBRESP	1:0	AXI bus for DMA
XMBMISC_INFO	3:0	AXI bus for DMA
XMARREADY		AXI bus for DMA
XMRID	7:0	AXI bus for DMA
XMRVALID		AXI bus for DMA
XMRLAST		AXI bus for DMA
XMRDATA	63:0	AXI bus for DMA
XMRMISC_INFO	3:0	AXI bus for DMA
XMRRESP	1:0	AXI bus for DMA
XMCSYSREQ		AXI bus for Lower Power
STARTRXDETU3RXDET		Start rx dectection from user
DISRXDETU3RXDET		Disable rx detection from user
SS_RX_ACJT_EN		ACJTAG for PMA
SS_RX_ACJT_ININ		ACJTAG for PMA
SS_RX_ACJT_INIP		ACJTAG for PMA
SS_RX_ACJT_INIT_EN		ACJTAG for PMA
SS_RX_ACJT_MODE		ACJTAG for PMA
SS_TX_ACJT_DRVEN		ACJTAG for PMA
SS_TX_ACJT_DATAIN		ACJTAG for PMA

Table 652: Input Ports (Continued)

Name	Range	Description
SS_TX_ACJT_HIGHZ		ACJTAG for PMA
USBPHY_REFCLK_ALT		Alternative refclk for USB PHY (60Mhz)

Table 653: Output Ports

Name	Range	Description
LMMIRDATAVALID		Read data valid of lmmi bus
LMMIREADY		Ready of lmmi bus
LMMIRDATA	31:0	Read data of lmmi bus
XMAWID	7:0	AXI bus for DMA
XMAWADDR	31:0	AXI bus for DMA
XMAWLEN	7:0	AXI bus for DMA
XMAWSIZE	2:0	AXI bus for DMA
XMAWBURST	1:0	AXI bus for DMA
XMAWLOCK	1:0	AXI bus for DMA
XMAWCACHE	3:0	AXI bus for DMA
XMAWPROT	2:0	AXI bus for DMA
XMAWVALID		AXI bus for DMA
XMAWMISC_INFO	3:0	AXI bus for DMA
XMWID	7:0	AXI bus for DMA
XMWVALID		AXI bus for DMA
XMWLAST		AXI bus for DMA
XMWDATA	63:0	AXI bus for DMA
XMWSTRB	7:0	AXI bus for DMA
XMBREADY		AXI bus for DMA
XMARID	7:0	AXI bus for DMA
XMARVALID		AXI bus for DMA
XMARADDR	31:0	AXI bus for DMA
XMARLEN	7:0	AXI bus for DMA
XMARSIZE	2:0	AXI bus for DMA
XMARBURST	1:0	AXI bus for DMA
XMARLOCK	1:0	AXI bus for DMA

Table 653: Output Ports (Continued)

Name	Range	Description
XMARCACHE	3:0	AXI bus for DMA
XMARPROT	2:0	AXI bus for DMA
XMARMISC_INFO	3:0	AXI bus for DMA
XMRREADY		AXI bus for DMA
XMCSYSACK		AXI bus for Lower Power
XMCACTIVE		AXI bus for Lower Power
DISRXDETU3RXDETACK		ACK of disabling rx detection to user
INTERRUPT		Interrupt of usb23 controller
SS_RX_ACJT_OUTN		ACJTAG for PMA
SS_RX_ACJT_OUTP		ACJTAG for PMA

Table 654: Bidirectional Ports

Name	Description
RESEXTUSB3	PMA signals
RESEXTUSB2	PMA signals
DP	PMA signals
DMP	PMA signals
RXMP	PMA signals
RXPP	PMA signals
TXMP	PMA signals
TXPP	PMA signals
VBUS	PMA signals
ID	PMA signals
XOIN18	PMA signals
XOOUT18	PMA signals
REFINCLKEXTM	PMA signals
REFINCLKEXTP	PMA signals

Table 655: Parameters

Name	Description
USB_MODE	The usb mode
GSR_DIS	

VFB_B

256 kb Single Port RAM

Architectures supported:

- iCE40 UltraPlus

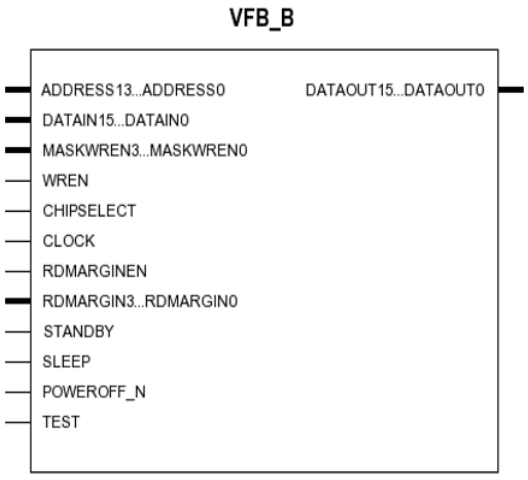


Table 656: Input Ports

Name	Range	Description
ADDRESS13...ADDRESS0		Read/write address
DATAIN15...DATAIN0		Data input
MASKWREN3...MASKWREN0		Write mask. Each bit masks a nibble of DATAIN.
WREN		Write enable
CHIPSELECT		Chip select
CLOCK		Clock
RDMARGINEN		Read margin enable

Table 656: Input Ports

Name	Range	Description
RDMARGIN3...RDMARGIN0		Read margin
STANDBY		Standby mode enable. Active high.
SLEEP		Sleep enable. Active high.
POWEROFF_N		Power off enable. Active low.
TEST		Test enable

Table 657: Output Ports

Name	Range	Description
DATAOUT15...DATAOUT0		Data output.

VHI

Logic High Generator

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP



Table 658: Output Ports

Name	Description
Z	VHI output Z

VLO

Logic Low Generator

Architectures supported:

- iCE40 UltraPlus
- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

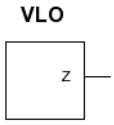


Table 659: Output Ports

Name	Description
Z	VLO output Z

WARMBOOT

Allows for loading a different configuration during run time

Architectures supported:

- iCE40 UltraPlus

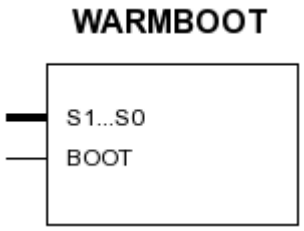


Table 660: Input Ports

Name	Range	Description
S1...S0		Select desired configuration image.
BOOT		Trigger the re-configuration process.

WDT

Interface for Configuration Watchdog Timer

Architectures supported:

- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- UT24C
- UT24CP

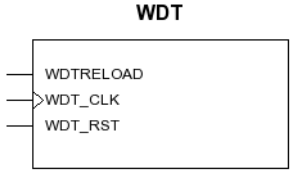


Table 661: Input Ports

Name	Range	Description
WDTRELOAD	1	Reload signal
WDT_CLK	1	Clock source
WDT_RST	1	Synchronous Reset signal

Table 662: Parameters

Name	Values	Description
WDTEN	"DIS" (default) "EN"	WDT Enable
WDTMODE	"SINGLE" (default) "CONTINUOUS"	Sets WDT Mode
WDTVALUE	0'b00000000000000000 (default)	WDT value

WDTA

Watchdog Timer Primitive

Architectures supported:

- LAV-AT-ES1
- LN2-CT

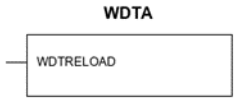


Table 663: Input Ports

Name	Range	Description
WDTRELOAD	1	Watchdog timer reload

Table 664: Parameters

Name	Values	Description
WDTMODE	"SINGLE" (default) "CONTINUOUS"	user_wdt_mode
WDTVALUE	"0b00000000000000000000000000000000" (default)	user_wdt_cnt
GSR	"ENABLED" (default) "DISABLED"	user_wdt_gsrn_dis

WIDEFN9

9-Input Single Output Logic Block

Architectures supported:

- LAV-AT-ES1
- LFCPNX
- LFD2NX
- LFMXO5
- LIFCL
- LN2-CT
- UT24C
- UT24CP

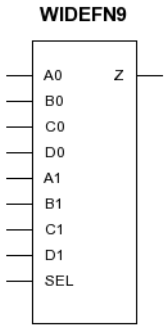


Table 665: Input Ports

Name	Description
A0	LUT0 Function input A0
B0	LUT0 Function input B0
C0	LUT0 Function input C0
D0	LUT0 Function input D0
A1	LUT0 Function input A1
B1	LUT0 Function input B1
C1	LUT0 Function input C1
D1	LUT0 Function input D1
SEL	LUT0/1 select

Table 666: Output Ports

Name	Description
Z	Function output

Table 667: Parameters

Name	Values	Description
INIT0	0x0000 (default)	LUT0 initialization data
INIT1	0x0000 (default)	LUT1 initialization data

DSP Inference Guide

The following are examples of DSP Inference.

```
// Multiply without register example, corresponding to the
following parameter settings of MAC16 (for iCE40UP primitive
only):
//      A_REG = "0b0"
//      B_REG = "0b0"
//      TOP_8x8_MULT_REG = "0b0"
//      BOT_8x8_MULT_REG = "0b0"
//      PIPELINE_16x16_MULT_REG1 = "0b0"
//      PIPELINE_16x16_MULT_REG0 = "0b0"
//      TOPOUTPUT_SELECT = "0b10"
//      BOTOUTPUT_SELECT = "0b10"

module mult_unsign_7_6(a,b,c);
    parameter A_WIDTH = 7;
    parameter B_WIDTH = 6;

    input unsigned [(A_WIDTH - 1):0] a;
    input unsigned [(B_WIDTH - 1):0] b;
    output unsigned [(A_WIDTH + B_WIDTH - 1):0] c;

    assign c = a * b;
endmodule


// Multiply/add without register example, corresponding to the
following parameter settings of
// MAC16:
//      A_REG = "0b0"
//      B_REG = "0b0"
//      TOP_8x8_MULT_REG = "0b0"
//      BOT_8x8_MULT_REG = "0b0"
//      PIPELINE_16x16_MULT_REG1 = "0b0"
//      PIPELINE_16x16_MULT_REG0 = "0b0"
//      TOPOUTPUT_SELECT = "0b00"
//      BOTOUTPUT_SELECT = "0b00"
//      TOPADDSUB_LOWERINPUT = "0b01"
//      BOTADDSUB_LOWERINPUT = "0b01"
module multaddsub_add_unsign_7_6(a,b,c,din);
    parameter A_WIDTH = 7;
    parameter B_WIDTH = 6;

    input unsigned [(A_WIDTH - 1):0] a;
    input unsigned [(B_WIDTH - 1):0] b;
    input unsigned [(A_WIDTH + B_WIDTH - 1):0] din;
    output unsigned [(A_WIDTH + B_WIDTH - 1):0] c;

    assign c = a * b + din;
endmodule

// Mult/sub with input registers, corresponding to the
following parameter settings of MAC16:
//      A_REG = "0b1"
//      B_REG = "0b1"
//      TOP_8x8_MULT_REG = "0b0"
//      BOT_8x8_MULT_REG = "0b0"
```

```
//      PIPELINE_16x16_MULT_REG1 = "0b0"
//      PIPELINE_16x16_MULT_REG0 = "0b0"
//      TOPOUTPUT_SELECT = "0b00"
//      BOTOUTPUT_SELECT = "0b00"
//      TOPADDSUB_LOWERINPUT = "0b01"
//      BOTADDSUB_LOWERINPUT = "0b01"

module multaddsub_sub_sign_ir_7_6(clk,a,b,din,c,rst,set);
    parameter A_WIDTH = 7;
    parameter B_WIDTH = 6;

    input rst;
    input set;
    input clk;
    input signed [(A_WIDTH - 1):0] a;
    input signed [(B_WIDTH - 1):0] b;
    input signed [(A_WIDTH + B_WIDTH - 1):0] din;
    output signed [(A_WIDTH + B_WIDTH - 1):0] c;

    reg signed [(A_WIDTH - 1):0] reg_a;
    reg signed [(B_WIDTH - 1):0] reg_b;
    reg signed [(A_WIDTH + B_WIDTH - 1):0] reg_din;
    assign c = reg_din - reg_a * reg_b;
    always @(posedge clk)
    begin
        if(rst)
        begin
            reg_a <= 0;
            reg_b <= 0;
            reg_din <= 0;
        end
        else if(set)
        begin
            reg_a <= -1;
            reg_b <= -1;
            reg_din <= -1;
        end
        else
        begin
            reg_a <= a;
            reg_b <= b;
           reg_din <= din;
        end
    end
endmodule
```

Note:

All Lattice architectures may not support implementation of this RTL code in MULTADDSUB DSP primitives.

Synthesis Attributes

This table lists synthesis attributes, and compares Radiant software and Lattice Diamond.

Function	Synplify Pro Diamond			Synplify Pro Radiant		
	Attribute	**Value**	**Description**	**Attribute**	**Value**	**Description**
ROM Synthesis	syn_romstyle	logic	Uses discrete logic primitives.	syn_romstyle	auto	Allows the synthesis tool to chose the best implementation to meet the design requirements for speed
		block_rom	Specifies that the inferred ROM be mapped to the appropriate vendor-specific memory.		logic	Uses discrete logic primitives.
		distributed	Implements the ROM structure as distributed ROM.		EBR	Specifies that the inferred ROM be mapped to the appropriate vendor-specific memory.
					distributed	Implements the ROM structure as distributed ROM.
DSP Synthesis	syn_dspstyle	DSP	Implements the multipliers as dedicated hardware blocks.	syn_multstyle	DSP	Implements the multipliers as dedicated hardware blocks.
		logic	Implements the multipliers as logic.		logic	Implements the multipliers as logic.

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 Radiant software graphical user interface, use Tcl commands to perform commands that are described in the Tcl Command Reference Guide.

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 that often precludes the need for some syntax, making commands less complex. See Command Line General Guidelines and Command Line Syntax Conventions.

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 Radiant software 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 lists all of the command line tools used for various design functions and their graphical user interface counterparts. The table also provides the functional description of each command line tool.

Table 1: The Radiant Software Core Design and Tool Chart

Design Function	Command Line Tool	Radiant Process	Description
Core Implementation and Auxiliary Tools			
Encryption	Encryption	Encrypt Verilog/VHDL files	Encrypts the input HDL source file with provided encryption key.
Synthesis	SYNTHESIS	Synthesize Design	Runs input source files through synthesis based on Lattice Synthesis Engine options set in Strategies.
Synthesis	Synpwrap	Synthesize Design	Used to manage Synopsys Synplify Pro synthesis programs.
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	Timing	Post-Synthesis Timing Report, MAP Timing Report, Place & Route Timing Report	Generates reports that can be used for static timing analysis.
Back Annotation	Backanno	Tool does not exist in the Radiant software interface as process but employed in file export.	Distributes the physical design information back to the logical design to generate a timing simulation file.

Table 1: The Radiant Software Core Design and Tool Chart

Design Function	Command Line Tool	Radiant Process	Description
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.
IP Packager	IP Packager	IP Packager	IP author select files from disks and pack them into one IPK file.

See Also

- Command Line Data Flow
- Command Line General Guidelines
- Command Line Syntax Conventions
- Invoking Core Tool Command Line Programs

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
- Command Line General Guidelines
- Command Line Syntax Conventions
- Setting Up the Environment to Run Command Line
- Invoking Core Tool Command Line Programs
- Invoking Core Tool Command Line Tool Help

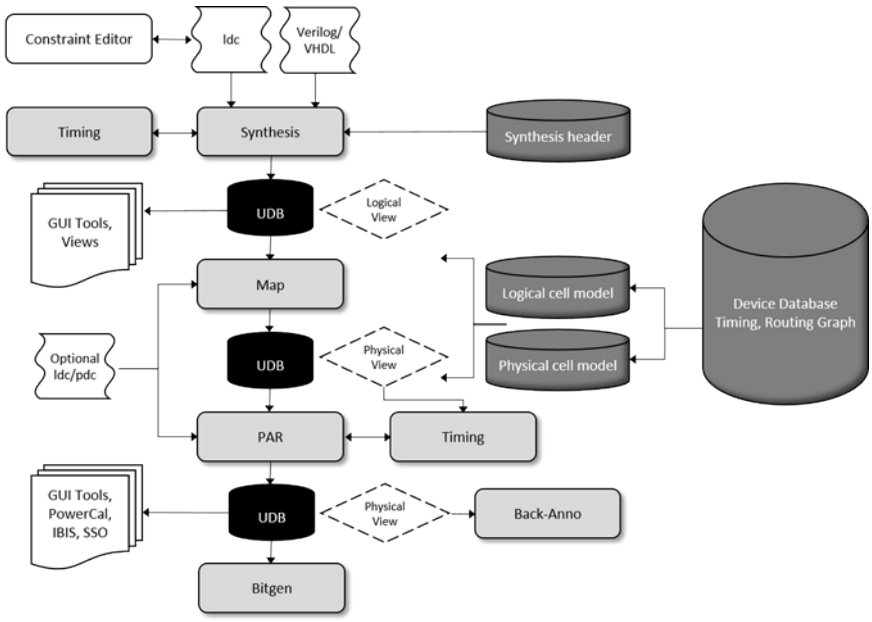
Command Line Data Flow

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
- Command Line General Guidelines



Command Line General Guidelines

You can run the FPGA family Radiant software 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> (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 Radiant software FPGA command line program, you can invoke help on available options with the **-h** or **-help** command. See Invoking Core Tool Command Line Programs 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 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 Radiant software 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 Radiant software 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.

See Also

- Command Line Reference Guide
- Invoking Core Tool Command Line Tool Help
- Command Line Syntax Conventions
- Using Command Files
- Command Line Data Flow

Command Line Syntax Conventions

The following conventions are used when commands are described:

Table 2: 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[**.udb**]> indicates that the .udb extension is optional but that the extension must be UDB.
{ }	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 Linux command, or a DOS command (for example, “The **playback** command is used to execute the macro you created.”).

Table 2: Command Line Syntax Conventions

Convention	Meaning
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/Radiant directory”).
Monospace	Indicates text that appears on the screen (for example, “File already exists.”) and text from Linux 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”

See Also

- Command Line Reference Guide
- Command Line General Guidelines
- Invoking Core Tool Command Line Tool Help
- Using Command Files

Setting Up the Environment to Run Command Line

For Windows The environments for both the Radiant Tcl Console window or Radiant Standalone Tcl Console window (radiant.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 the Radiant software from the Windows command line (via cmd.exe), you will need to add the following values to the following environment variables:

- PATH includes, for 64-bit

  ```
  <Install_directory>\bin\nt64;<Install_directory>\ispfpga\bin\nt64
  ```

  Example <*Install_directory*>:

  ```
  c:\lscc\radiant\1.0\bin\nt64;c:\lscc\radiant\1.0\ispfpga\bin\nt64
  ```

- FOUNDRY includes

  ```
  set FOUNDRY= <Install_directory>\ispfpga
  ```

For Linux On Linux, the Radiant software provides a similar standalone Tcl Console window (radiantc) 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, you need to run "bash" to switch to BASH" first, then run the following command.

- For BASH (64-bit):

```
export bindir=<Install_directory>/bin/lin64
source $bindir/radiant_env
```

After setting up for either Windows or PC, you can run the Radiant software executable files directly. For example, you can invoke the Place and Route program by:

```
par test_map.udb test_par.udb
```

See Also

- Invoking Core Tool Command Line Programs
- Invoking Core Tool Command Line Tool Help

Invoking Core Tool Command Line Programs

This topic provides general guidance for running the Radiant software FPGA flow core tools. Refer to Command Line Program Overview to see what these tools include and for further information.

For any the Radiant software 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 Linux™ or DOS™ command line. You can also run command line scripts or command files.

See Table 2 on page 1475 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
- Command Line Syntax Conventions
- Invoking Core Tool Command Line Tool Help
- Setting Up the Environment to Run Command Line
- Using Command Files

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 Radiant software 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 Radiant software applications that have architecture-specific command lines (e.g., iCE40 UltraPlus), 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
- Command Line Data Flow
- Command Line General Guidelines
- Command Line Syntax Conventions
- Setting Up the Environment to Run Command Line
- Using Command Files

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_libs.tcl from the Command Line
- Running HDL Encryption from the Command Line
- Running Synthesis from the Command Line
- Running Postsyn from the Command Line
- Running MAP from the Command Line
- Running PAR from the Command Line
- Running Timing from the Command Line

- Running Backannotation from the Command Line
- Running Bit Generation from the Command Line
- Running Various Utilities from the Command Line
- Using Command Files
- Using Command Line Shell Scripts

Running cmpl_libs.tcl from the Command Line

The cmpl_libs.tcl command allows you to perform simulation library compilation 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 do not have Tcl installed, or you have an older version, perform the following:

- Add <*Radiant_install_path*>/tcltk/windows/BIN to the front of your PATH, and
- For Linux users only, add <*Radiant_install_path*>/tcltk/linux/bin 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|]
```

Notes

- If Siemens 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.
- 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.
- To execute this script error free, **Siemens Questa 2020.4** or a later version should be used for compilation.

Check log files under <*target_path*> (defualt = .) for any errors, as follows:

- For Windows, type:

```
find /i "error" *.log
```

- For Linux, type:

```
grep -i error *.log
```

Subjects included in this topic:

- Running compl_lib.tcl
- Command Line Syntax
- cmpl_libs.tcl Options
- Example

Running compl_lib.tcl

cmpl_libs.tcl allows you to compile simulation libraries from the command line.

Command Line Syntax

tclsh <*Radiant_install_path*>/cae_library/simulation/scripts/cmpl_libs.tcl -sim_path <sim_path> [-sim_vendor \{mentor<default>|cadence|aldec}] [-device \{ice40up|lifcl|lfcpnx|lfd2nx|lfmxo5-25|lfmxo5-15d|lfmxo5-100t|lfmxo5-55t|lavat|ut24c|ut24cp|all<default>}] [-target_path <target_path>]

cmpl_libs.tcl Options

The table below contains all valid options for cmpl_libs.tcl.

Table 3: cmpl_libs.tcl Command Line Options

Option	Description
-sim_path <sim_path>	The -sim_path argument specifies the path to the simulation tool executable (binary) folder. This option is mandatory. Currently, only QuestaSim simulators from Mentor, Xcelium simulator from Cadence and Active-HDL simulator from Aldec are supported. NOTE: If <*sim_path*> has spaces, then it must be surrounded by " ". Do not use { }.
[-sim_vendor {mentor<default>|cadence|aldec}]	The -sim_vendor argument specifies simulation vendor. This argument is optional. If you do not use this argument, Mentor is chosen by default, and the libraries are compiled for QuestaSim. If you are using QuestaSim choose mentor, Xcelium choose cadence, Active-HDL choose aldec for -sim_vendor.

Table 3: cmpl_libs.tcl Command Line Options

Option	Description
[-device \{ice40up|lifcl|lfcpnx|lfd2nx|lfmxo5-25|lfmxo5-15d|lfmxo5-100t|lfmxo5-55t|lavat|ut24c|ut24cp|all<default>}]	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.
[-target_path <target_path>]	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. **Note**: 1. This argument is recommended if the current folder is the Radiant software'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 { }.

Example

This section illustrates and describes an example of a Simulation Libraries Compilation Tcl command.

The following command will compile all the Lattice FPGA libraries for Verilog simulation and place them under the folder specified by -target_path. The path to QuestaSim is specified by -sim_path.

```
tclsh <c:/lscc/radiant/1.0/>/cae_library/simulation/script/
cmpl_libs.tcl -sim_path C:/modeltech64_10.0c/win64 -target_path
c:/mti_libs
```

See Also

- Command Line Program Overview

Running HDL Encryption from the Command Line

Radiant software allows you to encrypt the individual HDL source files.

The tool supports encryption of Verilog HDL and VHDL files. Per command's execution, single source file is encrypted.

The HDL file can be partially or fully encrypted depending on pragmas' placements within the HDL file. To learn more about pragmas' placements, see Defining Pragmas.

Subjects included in this topic:

- Running HDL Encryption
- Command Line Syntax
- Encryption Options
- Examples

Running HDL Encryption

Before running the utility, you need to annotate the HDL file with the appropriate pragmas. Additionally, you may need to create a key file containing an encryption key. To view the key file's proper formating, see Key File.

Command Line Syntax

```
encrypt_hdl [-k <keyfile>] [-l language]
[-o <output_file>] [-comment <comment>] <input_HDL_file>
```

Encryption Options

The table below contains descriptions of all valid options for HDL encryption.

Table 4: Encryption Command Line Options

Option	Description
-h(elp)	Print command help message.
-k <keyfile>	A key repository file. Depending on the location of the key, this option is required or optional. ▶ If the HDL source file contains no pragma, the key file is required. The tool encrypts the entire HDL file using all key sets declared in the key file. ▶ If the HDL source file contains only `begin` and `end` pragmas and no key pragmas, the key file is required. The tool encrypts the section between `begin` and `end` using all key sets declared in the key file. ▶ If the HDL source file contains the proper key pragma, key_keyowner, key_keyname, but the key file is missing the provided key_public_key, the tool fetches the first public key string matching the key_keyowner and key_keyname requirement in the key file If the HDL source file contains the proper definition of key, this option is not required. NOTE: If the same key name is defined in both, HDL source file and key.txt file, the key defined in HDL source file has a precedence.
-l <language>	Directive language, vhdl or verilog (default).
-o <output_file>	An encrypted HDL file. This is an optional field. If not defined during the encryption, the tool generates a new output file <input_file_name>_enc.v.

Table 4: Encryption Command Line Options

Option	Description
<input>	Input Verilog or VHDL file.
-comment <comment>	Additional comment.

Examples

This section illustrates and describes a few examples of HDL encryption using Tcl commands.

Example 1

This example shows a successful encryption of HDL file with default options. It is assumed that key is properly defined in HDL file. Since output file name was not specified, the tool generates an output file <file_name>_enc.v in the same directory as the location of the input file.

```
> encrypt_hdl -k source/impl_1/keys.txt -o top.v top.v
Options:
  Key repository file:     source/impl_1/keys.txt
  Directive language:     <not specified>, use verilog as default
  Output file:  top.v
Processed 2 envelopes.
```

Example 2

This example shows a successful encryption of HDL file by generating a new output file.

```
> encrypt_hdl -k source/impl_1/keys.txt -o remote_files/top_v1_en.v remote_files/sources/top_v1_part.
Options:
  Key repository file:     source/impl_1/keys.txt
  Directive language:     <not specified>, use verilog as default
  Output file:  remote_files/top_v1_en.v
Processed 2 envelopes.
```

Example 3

This example shows unsuccessful HDL encryption due to a missing key file. To correct this issue, the user must either define the appropriate key file key.txt or annotate the HDL file with appropriate pragmas. To correct the issue, define the key either in key.txt file or directly in HDL source file.

```
> encrypt_hdl -o remote_files/sources/top_v1_part_en.v remote_files/sources/top_v1_part.v
Options:
  Key repository file:     <not specified>
  Directive language:     <not specified>, use verilog as default
  Output file:  remote_files/sources/top_v1_part_en.v
ERROR - remote_files/sources/top_v1_part.v at line 88: missing key.
```

Note

A key is always required in the encryption tool while key file is optional. If the complete key: `key_keyowner`, `key_keyname`, `key_method`, and `key_public_key`, is defined within HDL source file, key file is not required.

For specific steps and information on how to encrypt HDL files in the Radiant software, refer to the following section in the Radiant Software Help: **User Guides > Securing the Design**.

Defining Pragmas

Pragmas are used to specify the portion of the HDL source file that must be encrypted. Pragma' definition is compliant with IEEE 1735-2014 V1 standard.

Pragma syntax in Verilog HDL file:

```
`pragma protect <pragma's option>
```

Pragma syntax in VHDL file:

```
`protect <pragma's option>
```

Table 5: List of Available Pragma Options

Name	Available Values	Description
version	1 (default)	Specifies the current Radiant software encryption version.
author	string	Specifies the file creator.
author_info	string	Additional information you would like to include in file.
encoding	base64	The output format of processed data.
begin		The start point for data obfuscation.
end		The end point for data obfuscation.
key_keyowner	string	The key creator.
key_keyname	string	The RSA key name to specify the private key.
key_method	rsa	The cryptographic algorithm used for key obfuscation.
key_public_key		The RSA public key file name.
data_method	aes128-cbc aes256-cbc (default)	The AES encryption data method.

To encrypt HDL source file, encryption version, encoding type, and key specific pragmas must be defined in the HDL source file by HDL designer; only the content within the pragmas is encrypted.

Note

Multiple key sets can be declared in a single key file.

Example of Verilog source file marked with Pragmas:

```
// 3 bit counter with asynchronous reset
module count(c,clk,rst);

input clk,rst;
output [2:0]c;
reg [2:0]c;

`pragma protect version=1
`pragma protect author="<Your Name>"
`pragma protect author_info="<Your info>"
`pragma protect key_keyowner="Lattice Semiconductor"
`pragma protect key_keyname="LSCC_RADIANT_2"
`pragma protect key_method="rsa"
`pragma protect key_public_key
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0EZKUUhbuB6vSsc7OhQJ
iNAWJR5unW/OWp/LFI71eAl3s9bOYE2OlKdxbai+ndIeo8xFt2btxetUzuR6Srvh
xR2Sj9BbW1QToo2u8JfzD3X7AmRvlwKRX8708DPo4LDHZMA3qh0kfDDWkp2Eausf
LzE2cVxgq7fy/bDhUeN8xKQCSKJ7aguG6kOI6ROoZz21ljzDLUQzhm2qYF8SpU1o
tD8/uw53wLfSuhR3MBOB++xcn2imvSLqdgHWuhX6CtZIx5CD4y8inCbcLy/0Qrf6
sdTN5SAg2OZhjeNdzmqSWqhL2JTDw+Ou2fWzhEd0i/HN0y4NMr6h9fNn8nqxRyE7
IwIDAQAB

//put a blank line above
`pragma protect data_method="aes256-cbc"

`pragma protect begin
always @(posedge clk or posedge rst)
     begin
        if (rst)
           c = 3'b000;
         else
           c = c + 1;
      end

`pragma protect end

endmodule
```

The encrypted file may contain multiple encrypted key sets.

Example of encrypted Verilog file:

```
// 3 bit counter with asynchronous reset
module count(c,clk,rst);

input clk,rst;
output [2:0]c;
reg [2:0]c;

//put a blank line above

`pragma protect begin_protected
`pragma protect version=1
`pragma protect author="<Your Name>"
`pragma protect author_info="<Your info>"
`pragma protect encrypt_agent="Radiant encrypt_hdl"
`pragma protect encrypt_agent_info="Radiant encrypt_hdl Version 1.0"

`pragma protect encoding=(enctype="base64", line_length=64, bytes=256)
`pragma protect key_keyowner="Lattice Semiconductor"
`pragma protect key_keyname="LSCC_RADIANT_2"
`pragma protect key_method="rsa"
`pragma protect key_block
gOatWm+1rVPbcQIqaGf2gvNdUH/vTXzyRA4C+tdNxpqNWeeTXrTF+uIa21e9Io7S
K6ce3BVAXydtADq5Wy50EjcHzUi3Ym1eyEdfVn2pxCp+3csuiZSeNgbtYutonZjh
8ReQYzPqKt6fZvhZ1AqHiuuZhFUsZQFXqnT8IW4dQ7HWudREzx6jB7h+7vT+wJvH
N5kZMiHBFGRhiTePZ+yDOQwFVvwITezEoS099I8MoRGW1lU9kb4/Kenk96MIqE3W
lKAiQivIjXeWvLRqmOb0hNGRoOEYwjylY1pjq9GyelHoDC6czdyqOOWrunt//XNu
v/QtpeEe/co7o9arRtHbvw==

`pragma protect data_method="aes256-cbc"
`pragma protect encoding=(enctype="base64", line_length=64, bytes=176)
`pragma protect data_block
z+c6t504d6NkyrL5x6j6/raeaQmg0v9xmOQVaj3oq45SAsUIhGaihFVt+sS2kbNJ
/KBaqdciXAJHW8uRJtE/4nB+gZbVQsVnhRUlH/beksDnhdhWhNw/fcX6v6xptGwh
wQxsY+TjzT+pGC9rOEmAo2tndK63cWjHlg8hIZYnnw7yZAzv2OqylztSWFkdR9T4
mHclplFr96xb59oCRqbpQQgeESGgX9L1Yfd8j0ZXkSM=

`pragma protect end_protected

endmodule
```

Example of VHDL source file marked by Pragmas:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity top_test is

        port(
                cout : out std_logic_vector(7 downto 0);
                reset : in  std_logic;
                clk  : in  std_logic
            );
end top_test;

architecture top_test_arch of top_test is

signal count : std_logic_vector(7 downto 0);

begin

`protect version=1

`protect begin

  P100 : PROCESS(clk, reset)
  BEGIN
     IF (reset = '1') THEN
        count <= (others => '0');
     ELSE
        IF (clk'event and clk = '1') THEN
           count <= count + 1;
           cout <= count;
        END IF;
     END IF;
  END PROCESS;

`protect end


end top_test_arch;
```

Example of encrypted VHDL file:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity top_test is

        port(
                cout : out std_logic_vector(7 downto 0);
                reset : in  std_logic;
                clk  : in  std_logic
            );
end top_test;

architecture top_test_arch of top_test is

signal count : std_logic_vector(7 downto 0);

begin

`protect begin_protected
`protect version=1
`protect author="Lattice Semiconductor Corporation"
`protect author_info="Lattice Semiconductor Corporation"
`protect encrypt_agent="Radiant encrypt_hdl"
`protect encrypt_agent_info="Radiant encrypt_hdl Version 1.0"

`protect encoding=(enctype="base64", line_length=64, bytes=256)
`protect key_keyowner="Lattice Semiconductor"
`protect key_keyname="LSCC_RADIANT_2"
`protect key_method="rsa"
`protect key_block
Vi2YLO+x6rdARfQ9Dy5nkhsQDsmm1w6mdX+BGfDMCLHH9OVHbd1SmeHawCz0jXY8
ZfRK8VF/h1zOBmoosgY1pKd/Dpc5/4xkcEtnLzRbiXr1PhvP+tAFMMYrlFsqzJF/
0yoej8yT6mayYbd5mcEr5rzBmX29HuPBZbYr1ziac5IBpHZUa0mcwwhFnj5kz4OB
bVSqiay7v8ECP41vNzoRKrsTYKBOhiTa6UoG08ut2E4d8wJIXNBgZOuShYYzuOuv
1goVgwaRtFWrpTNXEmrZJPr/ikXRHTFzORpkDM7yNwGVTNJPMJ2aQde2wOi6EZWe
1NnRF4K2HK0Oz1NRbffIjQ==

`protect data_method="aes128-cbc"
`protect encoding=(enctype="base64", line_length=64, bytes=272)
`protect data_block
B7yNcwT9w4purXSxU1ln4f3rs1psxSPlV3pnWYipDj6rQSA7wniBxcC/laFebwKE
fvbKngTIn7N+W/1Den3kjpuznIvLy5cV/GTANFP0cWt9rnRrDCM5CYtNWqaMEZu7
o2QPLifpCvwEgygI0RO6NQS5frKo/jQLgOhf68+VpqFPozfrGAYI/YkEofh0foDH
9ScyO6grmJQCqtkqX2N3p6738N3iCFdKWJ6Udo5t++AT3YYeZZ7bprDN941k/BkU
YZij8fJx+qovJUWQmSUJYNnt2Vyoac4hsevXsFRxm439ssQtP4vHHEnraBSrHdVG
DJn0GqFID+tpC67tE61U2Y/yi18psFhZGs8jmuv9yGk=

`protect end_protected

end top_test_arch;
```

See Also

- Running HDL Encryption from the Command Line
- Key File

Key File

The key repository file defines the cryptographic public key used for RSA encryption. In Radiant software, the key file contains Lattice public key. Additionally, it may contain some of the common EDA vendors public keys.

The Lattice public key file key.txt is located at <*Radiant_installed_directory*>/ispfpga/data/ folder. Aside from the Lattice Public Key, the current version contains the public key for Synopsys, Aldec, and Cadence.

Note

If using Synplify Pro synthesis tool, both, the Lattice Public Key and the Synplify Pro Public Key must be defined in the key file. The Synplify Pro Public Key is used during the synthesis step to decrypt an encrypted design. The Lattice Public Key is used during the post-synthesis flow to decrypt an encrypted design.

A key file must contain properly declared pragmas such as `key_keyowner`, `key_keyname`, `key_method`, and `key_public_key` for each of the specified keys. The key value follows the `key_public_key` pragma.

The key file typically also contains the `data_method` pragma. It defines the algorithm used in data block encryption of HDL source file.

Example of a Key File:

```
// Use Verilog pragma syntax in this file
`pragma protect version=1
`pragma protect author="<Your Name>"
`pragma protect author_info="<Your info>"
`pragma protect key_keyowner="Lattice Semiconductor"
`pragma protect key_keyname="LSCC_RADIANT_2"
`pragma protect key_method="rsa"
`pragma protect key_public_key
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0EZKUUhbuB6vSsc7OhQJ
iNAWJR5unW/OWp/LFI71eAl3s9bOYE2OlKdxbai+ndIeo8xFt2btxetUzuR6Srvh
xR2Sj9BbW1QToo2u8JfzD3X7AmRvlwKRX8708DPo4LDHZMA3qh0kfDDWkp2Eausf
LzE2cVxgq7fy/bDhUeN8xKQCSKJ7aguG6kOI6ROoZz21ljzDLUQzhm2qYF8SpU1o
tD8/uw53wLfSuhR3MBOB++xcn2imvSLqdgHWuhX6CtZIx5CD4y8inCbcLy/0Qrf6
sdTN5SAg2OZhjeNdzmqSWqhL2JTDw+Ou2fWzhEd0i/HN0y4NMr6h9fNn8nqxRyE7
IwIDAQAB

// Put a blank line above
// Add additonal public keys below this line

`pragma protect data_method="aes256-cbc"

// End of File
```

See Also

- Running HDL Encryption from the Command Line
- Defining Pragmas

Running Synthesis from the Command Line

The Lattice Synthesis command tool allows you to synthesize Verilog and VHDL HDL source files into netlists for design entry into the Radiant software environment. Based on your strategy settings you specify in the Radiant software, a synthesis project (.synproj) file is created and then used by synthesis using the -f option. The Radiant software 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**).

Subjects included in this topic:

- Running Synthesis
- Command Line Syntax
- Synthesis Options
- Examples

Running Synthesis Synthesis will convert your input netlist (.v) file into a structural Verilog file that is used for the remaining mapping process.

- To run synthesis, type **synthesis** on the command line with valid options. A sample of a typical synthesis command would be as follows:

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>] [**-p** <device>] [**-sp** <performance_grade>] [**-t** <package_name>] [{**-path** <searchpath>}] [**-top** <top_module_name>] [**-ver** {<verilog_file.v>}] [**-sver** {systemverilog_file.sv}] [**-lib** <libname>] [**-vhd** {<vhdl_file.vhd/vhdl>}] [**-udb** <udb_file.udb>] [**-hdl_param** < param_name param_value >] [**-vh2008**] [**-optimization_goal** <area | timing | balanced (default)>] [**-force_gsr** <no (default) | yes>] [**-ramstyle** <auto (default) | distributed | block_ram(EBR) | registers>] [**-romstyle** <auto(default) | logic | EBR>] [**-output_edif** <filename.edf>] [**-output_hdl** <filename.v>] [**-sdc** <sdc_file.ldc>] [**-logfile** <synthesis_logfile>] [**-frequency** <target_frequency (default 200.0MHz (ICE40))>] [**-max_fanout** <max_fanout (default 1000)>] [**-bram_utilization** <bram_utilization (default 100%)>] [**-use_dsp** <0|1(default)>] [**-dsp_utilization** <dsp_utilization (default 100%)>] [**-mux_style** <auto(default) | pfu_mux | L6Mux_single | L6Mux_multiple>] [**-fsm_encoding_style** <auto(default) | one-hot | gray | binary>] [**-partition**] [**-partition_reset**] [**-resolve_mixed_drivers** <0(default)|1>] [**-fix_gated_clocks** <0|1(default)] [**-use_carry_chain** <0|1(default)>] [**-carry_chain_length** <chain_length>] [**-use_io_insertion** <0|1(default)>] [**-use_io_reg** <0|1|auto(default)>] [**-resource_sharing** <0|1(default)>] [**-propagate_constants** <0|1(default)>] [**-remove_duplicate_regs** <0|1(default)>] [**-loop_limit** <max_loop_iter_cnt (default 1950)>] [**-twr_paths** <timing_path_cnt>] [**-dt**] [**-comp**] [**-syn**] [**-ifd**] [**-f** <project_file_name>] [**-drc**] [**-drc_off** {<message_id>}] [**-drc_cfg** <rule> <value>]

Synthesis Options

The table below contains descriptions of all valid options for synthesis.

Table 6: 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.
-p <device>	Specifies the device type for the architecture (optional).
-sp <performance_grade>	Sets the device performance speed grade. If no performance grade specified, default performance value is used.
-f <proj_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.
-t <package_name>	Specifies the package type of the device.
-path <searchpath>	Add searchpath for Verilog “include” files (optional).
-top <top_module_name>	Name of top module (optional, but better to have to avoid ambiguity).
-lib <lib_name>	Name of VHDL library (optional).
-vhd <vhdl_file.vhd/vhdl>	Names of VHDL design files (must have, if language is VHDL or mixed language).
-ver <verilog_file.v>	Names of Verilog design files (must have, if language is Verilog, or mixed language).
-sver <systemverilog_file.sv>	Names of SystemVerilog design files (must have, if language is SystemVerilog or mixed language).
-hdl_param <param_name param_value>	Allows you to override HDL parameter pairs in the design file.
-optimization_goal <area | timing (default)>	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 ice40tp default to balanced.
-force_gsr <no(default) | yes>	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.

Table 6: Synthesis Command Line Options

Option	Description
-ramstyle <auto(default) | distributed | block_ram(EBR) | 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 (EBR)** 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 6: Synthesis Command Line Options

Option	Description
-romstyle <auto (default) | 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 performance, size, etc. ▶ **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 <filename.edf>	Specifies the name of the output EDIF netlist file.
-output_hdl <filename.v>	Specifies the name of the output Verilog netlist file.
-sdc <sdc_file.ldc>	Specifies a Lattice design constraint (.ldc) file input.
-loop_limit <max_loop_iter_cnt (default 1950)>	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 <synthesis_logfile>	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 <target_frequency (default 200.0MHz (ICE40))>	Specifies the target frequency setting. Default frequency value is 200.0 MHz.

Table 6: Synthesis Command Line Options

Option	Description
-max_fanout <max_fanout (default 1000)>	Specifies maximum global fanout limit to the entire design at the top level. Default value is 1000 fanouts.
-bram_utilization <bram_utilization (default 100%)>	Specifies block RAM utilization target setting in percent of total vacant sites. Default is 100 percent.
-use_dsp <0|1(default)>	Specifies how DSP modules are implemented and allows DSP inference.
-dsp_utilization <dsp_utilization (default 100%)>	Specifies the percentage of DSP that is allowed to be used.
-mux_style <auto(default) | pfu_mux | L6Mux_single | L6Mux_multiple>	Specifies the MUX style setting, which controls the way the macrogenerator implements the multiplexer macros. Valid options are *auto, L6Mux Multiple, L6Mux Single, and PFU Mux*. The default value is auto, meaning that the tool looks for the best implementation.
-fsm_encoding_style <auto(default) | 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(default)>	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(default)>	Specifies the use of I/O insertion. The 1 or true setting is the default.
-use_io_reg <0|1|auto(default)>	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

Table 6: Synthesis Command Line Options

Option	Description
-resource_sharing <0|1(default)>	Specifies the resource sharing option. The 1 or true setting is the default.
-propagate_constants <0|1(default)>	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(default)>	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. (This option is for internal use only.)
-comp	Indicates HDL compilation only without synthesis. (This option is for internal use only.)
-syn	Allows you to synthesize without HDL compilation. Reads the .db file. (This option is for internal use only.)
-udb <udb_file.udb>	Writes UDB output.
-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 <0|1(default)>	Allows you to enable/disable gated clock optimization. By default, the option is enabled.
-vh2008	Enables VHDL 2008 support.
-drc	Run LSE RTL design rule checking.
-drc_cfg <rule>	Configure a DRC rule
-drc_off <message_id>	Disable one or more DRC rules.
-partition	Enables the partition flow.
-partition_reset	Resets the partition flow (auto enables -partition).
-resolve_mixed_drivers <0 default | 1>	Connects the net to the VCC or GND driver if it is driven by them.

Examples

The following are examples of synthesis command lines and their respective descriptions:

synthesis -a "ice40tp" -p itpa08 -t SG48 -sp "6" -mux_style Auto

-use_io_insertion 1

-sdc "C:/my_radiant_tutorial/impl1/impl1.ldc"

-path "C:/lscc/radiant/1.0/ispfpga/ice40tp/data" "C:/my_radiant_tutorial/impl1" "C:/my_radiant_tutorial"

-ver "C:/my_radiant_tutorial/impl1/source/LED_control.v"

"C:/my_radiant_tutorial/impl1/source/spi_gpio.v"

"C:/my_radiant_tutorial/impl1/source/spi_gui_led_top.v"

-path "C:/my_radiant_tutorial"

-top spi_gui_led_top

-output_hdl "LEDtest_impl1.vm"

See Also

- Command Line Program Overview

Running Postsyn from the Command Line

The Postsyn process converts synthesized VM and integrates IPs into a completed design in UDB format for the remaining mapping process.

Subjects included in this topic:

- Command Line Syntax
- Postsyn Options

Command Line Syntax

```
postsyn [-w] [-a <architecture>] [-p <device>] [-t <package>]
[-sp <performance>] [-oc <operating_condition>] [-ldc
<ldc_file>] [-iplist <iplist_file>] [-macro <macro_list_file>]
[-gui] [-msgset <msgtypefile>] [-verbose] [-o <output.udb>] [-
keeprtl] [-top] [-cont_on_sdc_err] <input.vm>
```

Postsyn Options

Table 7: Postsyn Command Line Options

Option	Description
-h(elp)	Print command help message.
-w	Overwrite output file.
-a	Target architecture name.
-p	Target device name.
-t	Target package name.

Table 7: Postsyn Command Line Options

Option	Description
-sp	Target performance grade.
-oc	Target operating condition: commercial | industrial | automotive.
-ldc	Load LDC file.
-iplist	Load IP list file.
-macro	Load list file if any macro is used in the design.
-o	Output UDB file.
-keeprtl	Keep RTL view if it exists in UDB file.
-cont_on_sdc_err	Continue on SDC error instead of exit.
-verbose	Print more detailed messages.
-top	Indicate that the input is for the top design.
<input.vm>	Input structural Verilog file.

See Also

- Command Line Program Overview

Running MAP from the Command Line

The **Map Design** process in the Radiant software environment can also be run through the command line using the **map** program. The **map** program takes an input database (.udb) 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 Unified Database (.udb) file.

Subjects included in this topic:

- Running MAP
- Command Line Syntax
- MAP Options
- Example

Running MAP

MAP uses the database (.udb) file that was the output of the **Synthesis** process and outputs a mapped Unified Database (.udb) file with constraints embedded.

- 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 performance grade), the input .udb along with the input .ldc file. The output .udb file specified by the **-o** option. That

additional physical constraint file (*.pdc) can be applied optionally. A sample of a typical MAP command would be as follows:

map counter_impl1_syn.udb impl1.pdc -o counter_impl1.udb

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 constraint file here, but if you do, it must be specified after the input .udb file name. The constraint file automatically takes the name "**output**" in this case, which is the name given to the output .udb 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.udb into the current working directory, taking the name of the input file. If you specify the input.ldc 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.

Command Line Syntax

map [-h <arch>] <infile[.udb]> [<options>]

MAP Options

The table below contains descriptions of all valid options for MAP.

Table 8: MAP Command Line Options

Option	Description
-h <arch>	Displays all of the available MAP command options for mapping to the specified architecture.
<infile[.udb]>	Specifies the output design file name in .udb format. The .udb extension is optional.
-help	Print usage.
-inferGSR	GSR inferencing if applicable.
-o <name[.udb]>	Optional output design file (.udb).
-f	Reads map command line arguments and switches from file.
-mp <name[.mrp]>	Optional report file (.mrp).
-xref_sig	Report signal cross reference for renamed signals.
-xref_sym	Report symbol cross reference for renamed symbols.
-hierrpt	Print out hierarchical resource usage in .hrr file.
-cont_on_sdc_err	Ignore Constraint errors.
-u	Unclip unused instances.

Example

The following command maps an input database file named mapped.udb and outputs a mapped Unified Database file named mapped.udb.

```
map counter_impl1_syn.udb impl1.pdc -o counter_impl1.udb
```

See Also

- Command Line Data Flow
- Command Line Program Overview

Running PAR from the Command Line

The **Place & Route Design** process in the Radiant software environment can also be run through the command line using the **par** program. The **par** program takes an input mapped Unified Database (.udb) 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 .udb file.

The Implementation Engine multi-tasking option available in Linux is explained in detail here because the option is not available for PCs.

Subjects included in this topic:

- Running PAR
- Command Line Syntax
- General PAR Options
- Examples
- PAR Multi-Tasking (-m) Option

Running PAR

PAR uses your mapped Unified Database (.udb) file that were the outputs of the **Map Design** process or the **map** program. With these inputs, **par** outputs a new placed-and-routed .udb 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 .udb file and the desired name of the output .udb file. Design constraints from previous stages are automatically embedded in the input .udb file, however the par program can accept additional constraints with either a .pdc or .sdc file. A sample of a basic PAR command would be as follows:

par input.udb output.udb

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.

Command Line Syntax

par [-w] [-n <iterations:0,100>] [-t <iteration:0,100>] [-stopzero] [-s <savecount:0,100>] [-m <nodelistfile>] [-cores <number of cores>] [-r] [-k] [-p] [-x] [-pack <density:0,100>] [-sp <setupspeedgrade>] [-hsp <holdspeedgrade>] [-dh] [-hos] [-sort <method>] <infile> <outfile> [<pdcfile>]

Note

All filenames without special switches must be in the order <infile> <outfile> <pdcfile>. Options may exist in any order.

General PAR Options

Table 9: General PAR Command Line Options

Option	Description
-f	Read par command line arguments and switches from file.
-w	Overwrite. Allows overwrite of an existing file (including input file).
-n	Number of iterations (seeds). Use "-n 0" to run until fully routed and a timing score of zero is achieved. Default: 1.
-t	Start at this placer cost table entry. Default is 1.
-stopzero	Stop running iterations once a timing score of zero is achieved.
-s	Save "n" best results for this run. Default: Save All.
-m	Multi task par run. File "<node list file>", contains a list of node names to run the jobs on.
-cores	Run multiple threads on the local machine. You can specify "<number of cores>" to run the jobs. For cases when the user specifies both -cores and -m with a valid node list file, PAR should apply both settings (merge). If the user repeats the host machine in the node list file, the settings in the node list file take precedence over the setting in -cores (for backwards compatibility).
-p	Do not run placement.
-r	Do not run router.
-k	Keep existing routing in input UDB file. Note: only meaningful when used with -p.
-x	Ignore timing constraints.

Table 9: General PAR Command Line Options

Option	Description
-pack	Set the packing density parameter. Default: auto.
-sp	Change performance grade for setup optimization. Default: Keep current performance grade.
-hsp	Change performance grade for hold optimization. Default: M.
-dh	Disable hold timing correction.
-hos	Prioritize hold timing correction over setup performance.
-sort	Set the sorting method for ranking multiple iterations. <method> "c" sorts by cumulative slack, "w" sorts by worst slack. Default: c.
<infile>	Name of input UDB file.
<outfile>	Name of output UDB file.

Table 10: PAR Placement Command Line Option

Option	Description
<pdcfile>	Name of optional constraint file. Note: the contents of <pdcfile> will overwrite all constraints saved in the input UDB file <infile>.

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.udb and writes the placed and routed design to output.udb.

```
par input.udb output.udb
```

Example 2

The following command runs 20 place and route iterations. The iterations begin at cost table entry 5. Only the best 3 output design files are saved.

```
par -n 20 -t 5 -s 3 input.udb output.udb
```

Example 3

(Lattice FPGAs only) This is an example of **par** using the **-io** switch to generate .udb files that contain only I/O for viewing in the PAD Specification file for adjustment of ldc_set_location constraints for optimal I/O placement. You can display I/O placement assignments in the Radiant Spreadsheet View and choosing **View > Display IO Placement**.

par -io -w lev1bist.udb lev1bist_io.udb

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 -n 10 mydesign.udb output.udb
```

tells PAR to run 10 place and route passes (**-n 10**). It runs each of the 10 jobs consecutively, generating an output .udb file for each job, i.e., output_par.dir/5_1.udb, output_par.dir/5_2.udb, 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 below 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 <udb_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 -n 10 mydesign.udb output.udb
```

This runs the following jobs on the nodes specified.

```
Starting job 5_1 on node NODE1 at ...
Starting job 5_2 on node NODE2 at ...
Starting job 5_3 on node NODE3 at ...
Starting job 5_4 on node NODE4 at ...
Starting job 5_5 on node NODE5 at ...
```

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.

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=<install_directory>/ispfpga/bin/lin64
export PATH=$FOUNDRY/bin/lin64:$PATH
export LD_LIBRARY_PATH=$FOUNDRY/bin/lin:$LD_LIBRARY_PATH
64
```

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

- **User Guides > Implementing the Design** in the Radiant Software Help
- Command Line Data Flow
- Command Line Program Overview

Running Timing from the Command Line

The **MAP Timing** and **Place & Route Timing** processes in the Radiant software environment can also be run through the command line using the **timing** program. Timing can be run on designs using the placed and routed Unified Design Database (.udb) and associated timing constraints specified in the design's (.ldc,.fdc, .sdc or .pdc) file or device constraints extracted from the design. Using these input files, **timing** provides static timing analysis and outputs a timing report file (.tw1/.twr).

Timing checks the delays in the Unified Design Database (.udb) file against your timing constraints. If delays are exceeded, Timing issues the appropriate timing error. See “Implementing the Design” in the Radiant Software Help and associated topics for more information.

Subjects included in this topic:

- Running Timing
- Command Line Syntax
- Timing Options
- Examples

Running Timing

Timing uses your input mapped or placed-and-routed Unified Design Database (.udb) file and associated constraint file to create a Timing Report.

- To run Timing, type **timing** on the command line with, at minimum, the names of your input .udb and sdc files to output a timing report (.twr) file. A sample of a typical Timing command would be as follows:

  timing design.udb (constraint is embedded in udb)

Note

The above command automatically generates the report file named design.twr which is based on the name of the .udb file.

There are several command line options that give you control over the way Timing generates timing reports for analysis. Please refer to the rest of the subjects in this topic for more details. See Examples.

Command Line Syntax timing <udb file name> [-sdc <sdc file name>] [-hld | -sethld] [-o <output file name>] [-v <integer>] [-endpoints <integer>] [-help]

Timing Options

The following tables contain descriptions of all valid options for Timing.

Table 11: Compulsory Timing Command Line Options

Compulsory Option	Description
-db-file arg	Design database file name.

Table 12: Optional Timing Command Line Options

Optional Option	Description
-endpoints arg (=10)	Number of end points.
-logical-db-file arg	Logical design database file name.
-logical-sdc-file arg	Alternate sdc file name.
-u arg (=10)	Number of unconstrained end points printed in the table.
-ports arg (=10)	Number of top ports printed in the table.
-help	Print the usage and exit.
-hld	Hold report only.
-sp arg (=None)	Setup speed grade.
-hsp arg (=M)	Hold speed grade.
-rpt-file arg	Timing report file name.
-o arg	Timing report file name.
-alt_report	Diamond-like report.
-report_sdc	Parsed file appears in report file.
-sdc-file arg	sdc file name.
-sethld	Both setup and hold report.
-v arg (=10)	Number of paths per constraint.
-time_through_async	Timer will time through async resets.
-iotime	Compute the input setup/hold and clock to output delays of the FPGA.
-io_allspeed	Get worst IO results for all speed grades.
-pwrprd	Output clock information for PowerCalculator.
-nperend arg (=1)	Number of paths per end point.
-html	HTML format report.

Table 12: Optional Timing Command Line Options

Optional Option	Description
-gui	Call from GUI.
-msg arg	Message log file.
-msgset arg	Message setting.
-io_allspeed	Get worst IO results for all speed grades.
-dump_cdc	Dump CDC registers.
-false_covers	False path improves coverage
-dump_uncovered	Dump load pins of connections that are not covered.

Examples

Following are a few examples of Timing command lines and a description of what each does.

Example 1

The following command verifies the timing characteristics of the design named design1.udb, generating a summary timing report. Timing constraints contained in the file group1.prf are the timing constraints for the design. This generates the report file design1.twr.

timing design1.udb (constraint is embedded in udb)

Example 2

The following command produces a file listing all delay characteristics for the design named design1.udb. Timing constraints 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.

timing -v design1.udb -o output.twr

Example 3

The following command analyzes the file design1.udb and reports on the three worst errors for each constraint embedded in design1.udb and in additional timing constraint file (e.g. test.ldc). The report is called design1.twr.

timing -e 3 -sdc-file test.ldc design1.udb

Example 4

The following command analyzes the file design1.udb and produces a verbose report to check on hold times on any timing constraints embedded in design1.udb. With the output report file name unspecified here, a file using the root name of the .udb file (i.e., design1.twr) will be output by default.

timing -v -hld design1.udb

Example 5

The following command analyzes the file design1.udb and produces a summary timing report to check on both setup and hold times on any timing constraints embedded in design1.udb. With the output report file name unspecified here, a file using the root name of the .udb file (i.e., design1.twr) will be output by default.

timing -sethld design1.udb

See Also

- Command Line Program Overview
- Command Line Data Flow

Running Backannotation from the Command Line

The **Generate Timing Simulation Files** process in the Radiant software environment can also be run through the command line using the **backanno** program. The **backanno** 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 **backanno** is a Unified Database file (.udb) a mapped and partially or fully placed and/or routed design.

Subjects included in this topic:

- Running Backanno
- Command Line Syntax (Verilog)
- Backanno Options
- Examples

Running Backanno

Backanno uses your input mapped and at least partially placed-and-routed Unified Database (.udb) file to produce a back-annotated netlist (.v) and standard delay (.sdf) file. This tool supports all FPGA design architecture flows. Only Verilog netlist is generated.

- To run backanno, type **backanno** on the command line with, at minimum, the name of your input .udb file. A sample of a typical backanno command would be as follows:

backanno backanno.udb

Note

The above command back annotates backanno.udb and generates a Verilog 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 backanno generates back-annotated netlists for simulation. Please refer to the rest of the subjects in this topic for more details.

Command Line Syntax (Verilog)

backanno [**-w**] [**-pre** <prfx>] [**-sp** <grade>] [**-neg**] [**-pos**] [-sup] [**-min**] [**-x**] [**-fc**] [-slice] [-slice0] [-slice1] [-noslice] [**-t**] [**-dis** [<del>]] [**-m** [<limit>]] [**-u**] [**-i**] [**-nopur**] [**-l** <libtype>] [**-s** <separator>] [**-o** <verilog<<.v>>] [**-d** <delays[.sdf]>] [**-gui**] [**-msg** <msglogfile>] [**-msgset** <msgtypefile>] [<udbfile>]

Backanno Options

The table below contains descriptions of all valid options for backanno.

Table 13: Backanno Options

Option	Description
-w	Overwrite the output files.
-sp <grade>	Override performance grade for backannotation.
-pre <prfx>	Prefix to add to module name to make them unique for multi-chip simulation.
-min	Override performance grade to minimum timing for hold check.
-dis <del>	Distribute routing delays by splitting the signal and inserting buffers. <del> is the maximum delay (in ps) between each buffer (1000ps by default).
-m <limit>	Shortens the block names to a given character limit in terms of some numerical integer value.
-u	Add pads for top-level dangling nets.
-neg	Negative setup/hold delay support. Without this option, all negative numbers are set to 0 in SDF.
-pos	Write out 0 for negative setup/hold time in SDF for SC.
-x	Generate x for setup/hold timing violation.
-i	Create a buffer for each block input that has interconnection delay.

Table 13: Backanno Options

Option	Description
-nopur	Do not write PUR instance in the backannotation netlist. Instead, user has to instantiate it in a test bench.
<type>	Netlist type to write out.
<libtype>	Library element type to use.
<netfile>	The name of the output netlist file. The extension on this file will change depending on which type of netlist is being written. Use -h <type>, where <type> is the output netlist type, for more specific information.
<udb file>	Input file '.udb '.

Examples

Following are a few examples of backanno command lines and a description of what each does.

Example 1

The following command back annotates design.udb and generates a Verilog file design.vo and an SDF file design.sdf. If the target files exist, they will be overwritten.

```
backanno -w design.udb
```

Example 2

The following command back annotates design.udb and generates a Verilog file backanno.vo 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.

```
backanno -dis 2000 -o backanno design.udb
```

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).

```
backanno -sp 2 -i design.udb
```

Example 4

The following command generates Verilog netlist and SDF files without setting the negative setup/hold delays to 0:

```
backanno -neg -n verilog design.udb
```

See Also

- Command Line Program Overview
- Command Line Data Flow

Running Bit Generation from the Command Line

The **Bitstream** process in the Radiant software 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 Unified Database (.udb) file as input and produces a configuration bitstream (bit images) needed for programming the target device.

Subjects included in this topic:

- Running BITGEN
- iCE40UP Command Line Syntax
- All devices except iCE40UP Command Line Syntax
- BITGEN Options
- Examples

Running BITGEN

BITGEN uses your input, fully placed-and-routed Unified Database (.udb) 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 .udb file if you run **bitgen** from the directory where it resides and there is no other .udb present.

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.

iCE40UP Command Line Syntax

bitgen [**-d**] [**-b**] [**-a**] [**-w**] [**-noebrinitq1**] [**-noebrinitq2**] [**-noebrinitq3**] [**-noheader**] [**-simbitmap**] [**-nvcm**] [**-freq** <frequency_bit_setting>] [**-spilowpower**] [**-warmboot**] [**-nvcmsecurity**] {**-g** <setting_value>} <infile> [<outfile>]

All devices except iCE40UP Command Line Syntax

bitgen [**-d**] [**-w**] [**-m** <format>] [**-g** <opt:val>] [**-f** <*.t2b>] [**-s** <*.secproj>] {**-site** <seirule>} {**-site** <seitype>} <infile> [<outfile>]

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 14: iCE40UP BITGEN Command Line Options

Option	Description
-d	Disable DRC.
-b	Produce .rbt file (ASCII form of binary).
-a	Produce .hex file.
-w	Overwrite an existing output file.
-freq <frequency_bit_setting>	Can setup different frequency: 0 = slow, 1 = medium, 2 = fast. Depending on the speed of external PROM, this options adjusts the frequency of the internal oscillator used by the iCE40UP device during configuration. This is only applicable when the iCE40UP device is used in SPI Controller Mode for configuration.
-nvcm	Produce NVCM file.
-nvcmsecurity	Set security. Ensures that the contents of the Non-Volatile Configuration Memory (NVCM) are secure and the configuration data cannot be read out of the device.
-spilowpower	SPI flash low power mode. Places the PROM in low-power mode after configuration. This option is applicable only when the iCE40UP device is used as SPI Controller Mode for configuration.
-warmboot	Enable warm boot. Enables the Warm Boot functionality, provided the design contains an instance of the WARMBOOT primitive.
-noheader	Do not include the bitstream header.
-noebrinitq0	Do not include EBR initialization for quadrant 0.
-noebrinitq1	Do not include EBR initialization for quadrant 1.
-noebrinitq2	Do not include EBR initialization for quadrant 2.
-noebrinitq3	Do not include EBR initialization for quadrant 3.

Table 14: iCE40UP BITGEN Command Line Options

Option	Description
-g NOPULLUP:ENABLED	No IO pullup. Removes the pullup on the unused I/Os, except Bank 3 I/Os which do not have pullup.
-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>	The input post-PAR design database file (.udb).
<outfile>	The output file. If you do not specify an output file, BITGEN creates one in the input file's directory If you specify -b, the extension is .rbt. If you specify –a, the extension is .hex. If you specify –nvcm, the extension is .nvcm. Otherwise the extension is .bin. A report (.bgn) file containing all of BITGEN's output is automatically created under the same directory as the output file.

Table 15: BITGEN Command Line Options (all devices except iCE40UP)

Option	Description
-d	Disable DRC.
-w	Overwrite an existing output file.
-m <format>	Create "mask" and "readback" files. Valid formats are: 0: Output files in ASCII 1: Output files in binary.
-g <opt:val>	Set option to value, options are (First is default): CfgMode Disable, Flowthrough, Bypass RamCfg Reset, NoReset DONEPHASE T3, T2, T1, T0 GOEPHASE T1, T3, T2 GSRPHASE T2, T3, T1 GWEPHASE T2, T3, T1 ES Yes, No.
-f <*.t2b>	Load additional command line arguments from a .t2b file.
-s <*.secproj>	Load a bitstream security settings file.
-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.

Table 15: BITGEN Command Line Options (all devices except iCE40UP)

Option	Description
<infile>	The input post-PAR design database file (.udb).
<outfile>	The output file. If you do not specify an output file, BITGEN creates one in the input file's directory If you specify -b, the extension is .rbt. If you specify –a, the extension is .hex. If you specify –nvcm, the extension is .nvcm. Otherwise the extension is .bin. A report (.bgn) file containing all of BITGEN's output is automatically created under the same directory as the output file.

Examples

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**, and specifies a Raw Bit File (.rbt) output with **-b**. Notice how these three options can be combined with the **-wdb** syntax.

```
bitgen -wdb <design.udb>
```

Example 2

There are two command line options to set the SEI Editor tool with the following examples:

- 1-bit Error Injection

```
bitgen –w  <other options>  -ebit 1 -sei unused –site
<unused site type> <des>.udb  <output file>
    Avant: -where <unused site type> = [PFU  | DSP |ANY]
    Nexus: -where <unused site type> = [PFU  | DSP | EBR
    |ANY]
```

  - Unused

```
bitgen –w -ebit 1  –sei unused –site PFU  des.udb
des_partial.bit
```

  - Random

```
bitgen –w -ebit 1 –sei random - des.udb  des_partial.bit
```

- 2-bit Error Injection

```
bitgen –w  <other options> -ebit 2 -sei unused –site
<unused site type>  <des>.udb  <output file>
    Avant: -where <unused site type> = [PFU  | DSP |ANY]
    Nexus: -where <unused site type> = [PFU  | DSP | EBR
    |ANY]
```

- Unused

```
bitgen -w -ebit 2 -sei unused -site DSP des.udb
des_partial.bit

bitgen -b -ebit 2 -sei unused -site DSP,DSP < udb design
>
```

- Random

```
bitgen -w -ebit 2 -sei  random des.udb  des_partial.bit
```

Note

The <other options> option does not support compressed or encrypted partial bitstream, and if it includes “-b”, it generates an ASCII bitstream output file.

See Also

- Command Line Program Overview
- Command Line Data Flow

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
- Example

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>] [**-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.

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

Calculation formula for USB-2B (2232H FTDI USB host chip): Frequency = 60 MHz / (1 + ClockDivider) *2

Calculation formula for USB-2B (2232D FTDI USB host chip): Frequency = 12 MHz / (1 + ClockDivider) *2

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
-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

Example

The following is a PGRCMD example.

```
pgrcmd -infile c:\test.xcf
```

See Also

- Command Line Data Flow
- Command Line Program Overview

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 you 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>]

[**-freq**]

[**-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>]

|

-oft <**-int | mot | xtek**>

[**-dev** <Device Name>]

-if "<Data File Path>"

[-**of** "<Data File Path>"]

[**-compress** <on | off>]

[**-verifyid** <on | off>]

[**-freq**]

[**-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**]

[**-preamble**]

[**-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**]

[**-preamble**]

[**-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. **-freq**: 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. **-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, 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. **-freq**: 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. **-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, 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
-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. **-freq**: 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. **-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, 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. **-freq**: 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. **-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, 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. **-freq**: 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 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 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, 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 (optiona)l. 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 (optiona)l. **-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 Target 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 Target 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:** Inserts the Golden File at the beginning of the upper half of the SPI flash (optional). 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). **-preamble**: Replaces the header Preamble of the primary bitstream with 0xFFs (for Fail Safe Upgrade feature) (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.

Option	Description
-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). **-preamble**: Replaces the header Preamble of the bitstream with 0xFFs (for Fail Safe Upgrade feature) (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. **-freq**: 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/radiant/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/radiant/3.5/bin/nt/ddtcmd" -oft -svfchain -if
"C:/example/demo.xcf" -of  "C:/ example/demo.svf"
```

STAPL - Single Device:

```
"C:/lscc/radiant/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/radiant/3.5/bin/nt/ddtcmd" -oft -stpchain -if
"C:/example/demo.xcf" -of  "C:/example/demo.svf"
```

ATE - Generic Vector Format:

```
"C:/lscc/radiant/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type tst -of " C:/example/demo.tst"
```

ATE - GenRad:

```
"C:/lscc/radiant/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type gr -of " C:/example/demo.gr"
```

ATE - HP3070:

```
"C:/lscc/radiant/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type hp3070 -of " C:/example/
demo.pcf"
```

ATE - HP3065:

```
"C:/lscc/radiant/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type hp3065 -of " C:/example/
demo.pcf"
```

ATE - Teradyne 1800:

```
"C:/lscc/radiant/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/radiant/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/radiant/3.5/bin/nt/ddtcmd" -oft -fullvme -if
"C:/example/demo.xcf" -of "C:/example/demo.vme"
```

JTAG Slim VME Embedded:

```
"C:/lscc/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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/radiant/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
- Command Line Program Overview

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 Radiant software environment. Click the links below for details.

- Synpwrap
- IP Generation
- IP Packager
- ECO Editor
- Standalone Timing Analyzer

Note

For information on commonly-used FPGA command line tools, see Command Line Basics.

Synpwrap

The **synpwrap** command line utility (wrapper) is used to manage Synplicity Synplify and Synplify Pro synthesis programs from the Radiant software 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 Radiant Software Help. The

synpwrap program drives **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 Pro for Lattice synthesis scripts from the command line. For more information on synthesis automation of Synplify Pro, see the “User Batch Mode” section of the *Synplicity Synplify and Synplify Pro for Lattice User Guide*.

If you use Synplify Pro, the Lattice OEM license requires that the command line executables **synplify_pro** be run by the Lattice “wrapper” program, **synpwrap**.

Command Line Syntax

synpwrap [**-nolog**] [**-int** <*command_file*>] [**-gui**] [**-int** <*prj_file*> | **-prj** <*project_file*>] [**-notoem**] [**-notpro**] [**-options** <*arguments*>]

Table 16: SYNPWRAP Command Line Options

Option	Description
-options <*arguments*>	Passes all arguments to Synplify/Synplify Pro. Ignores all other options except -notOEM and -notPro. It must be at the end of other options.
-gui	Invokes the Synplify GUI instead of batch running.
-notOEM	Does not use the Lattice OEM version of Synplify/Synplify Pro.
-notPro	Does not use Synplify Pro. It will automatically use Synplify if SynplifyPro does not exist.
-int <*prj_file*>	Same as the -gui and -prj <file> options. Invokes Synplify GUI with project per command file.
-nolog	Does not print out the log file after the process is finished if there is no error (batch mode only).
-prj <*project_file*>	Uses Synplify .prj file instead of the command file (recommended).
-fg	Forces to run in foreground mode, and waits for Synplify to quit. Only available with the -options command.
-h|-help	Prints this usage. Same as running the process without any arguments.

Example

Below shows a synpwrap command line example.

```
synpwrap -gui -prj C:/lscc/radiant/2023.2/sample/sample.prj
```

See Also

- Command Line Program Overview
- Command Line Data Flow

IP Generation

The IP generation command allows you to generate or update existing IPs in the design.

Command Line Syntax

ipgen.EXE [**-h**] [**-o** OUTPUT_DIR] [**-proj_dir** PROJ_DIR] [**-ip** IP_MODULE_DIR] [**-cfg** IP_CONFIG_FILE] [**-skip_uniquify** SKIP_UNIQUIFY_HDL] [**-cfg_value** IP_CONFIG_VALUE] **-name** IP_INSTANCE_NAME [**-platform** IPGEN_PLATFORM] [**-vlnv** IPGEN_VLNV] [**-rtl_library_path** RTL_LIBRARY_PATH] [**-a** ARCHITECTURE] [**-p** DEVICE] [**-t** PACKAGE] [**-sp** PERFORMANCE] [**-f** FAMILY] [**-op** OPERATION]

Table 17: IP Generation Command Line Options

Option	Description
-h, --help	Shows this help message.
-o OUTPUT_DIR	Location where the IP instance package will be generated. Default is the current directory.
-proj_dir PROJ_DIR	Location of the Radiant project, which owns generated IP instance package. Default is the current directory.
-ip IP_MODULE_DIR	Location of the IP module package.
-cfg IP_CONFIG_FILE	Location of the IP configuration file.
-skip_uniquify SKIP_UNIQUIFY_HDL	Controls whether uniquified HDL files are True or False.
-cfg_value IP_CONFIG_VALUE	IP configuration value: 'id:value; id1:value1'
-name IP_INSTANCE_NAME	Name of the IP instance package.
-platform IPGEN_PLATFORM	Name of ipgen's runtime environment: Radiant or Propel. The default is Radiant.
-vlnv IPGEN_VLNV	Location of the IP configuration file by vendor library name and version.
-rtl_library_path RTL_LIBRARY_PATH	The RTL library path for IP.
-a ARCHITECTURE	Target architecture name.
-p DEVICE	Target device name.
-t PACKAGE	Target package name.
-sp PERFORMANCE	Target performance name.

Table 17: IP Generation Command Line Options

Option	Description
-f FAMILY	Target family name.
-op OPERATION	Specify the device operation.

See Also

- Command Line Program Overview
- Command Line Data Flow

IP Packager

The IP Packager (ippkg) tool can be run from the command line, allowing IP developers to select files from disks and pack them into one IPK file.

The following describes the process of IP Packager:

- IP author prepares metadata files, RTL files, HTML files, etc (all files of a Soft IP).
- IP Packager GUI provides UI for IP author to select files from the disk, and call IP Packaging engine to pack them into an IPK file.
- IP Packaging engine encrypts RTL files if IEEE P1735-2014 V1 pragmas are specified in RTL source

Command Line Syntax

ippkg [**-h**] (**-metadata** METADATA_FILE | **-metadata_files** METADATA_LIST_NAME) (**-rtl** RTL_FILE | **-rtl**_files RTL_LIST_NAME) [**-encrypt** FORCE_ENCRYPT_FILE | **-encrypt_files** FORCE_ENCRYPT_LIST_NAME] [**-plugin** PLUGIN_FILE] [**-ldc** LDC_FILE] [**-fdc** FDC_FILE] [**-testbench** TESTBENCH_FILE | **-testbench_files** TESTBENCH_LIST_NAME] [**-driver_file** DRIVER_FILE | **-driver_files** DRIVER_LIST_NAME] [**-eval_file** EVAL_FILE | **-eval_files** EVAL_LIST_NAME] (**-help_file** HELP_FILE | **-help_files** HELP_LIST_NAME) [**-license_file** LICENSE_FILE] [**-o** OUTPUT_ZIP_FILE] [**-key_file** KEY_FILE]

Table 18: IPPKG Command Line Options

Option	Description
-metadata	The file name will be fixed to ‘metadata.xml.
-metadata_files	Location of the file which stores the metadata files. One line is a file path in specified file. Must have a file named metadata.xml.
-rtl	Specify the IP RTL file.
-rtl_files	One line is a file path in specified file.

Table 18: IPPKG Command Line Options

Option	Description
-encrypt	Encrypt the whole RTL files.
-encrypt_files	One line is a file path in specified file.
-plugin	The file name will be fixed to 'plugin.py'.
-ldc	Specify the LDC file.
-fdc	Specify the FDC file.
-testbench	Specify the testbench file.
-testbench_files	One line is a file path in specified file.
-driver	Specify the driver file.
-driver_files	One line is a file path in specified file.
-eval	Specify the IP evaluation file.
-eval_files	One line is a file path in specified file.
-help_file	Specify the help file, must be <path>/introduction.html.
-help_files	One line is a file path in specified file.
-license_file	Specify the license file.
-o	Specify the output zip file.
-key_file	Specify the key file to encrypt the RTL files.

Example

The following is an ippkg command line example:

```
ippkg -metadata c:/test/test.xml -rtl_files c:/test/rtl_list -
help_file c:/test/introduction.html
```

See Also

- Command Line Program Overview
- Command Line Data Flow

ECO Editor

The ECO Editor tool can be run from the command line.

ECO Editor is also able to dump the ECO Tcl commands which user performed in GUI view without saving any UDB file.

In the meanwhile, we will have one non-GUI ECO engine tool, which accepts the dumped Tcl script file with a UDB file and outputs a new UDB file.

User can set ‘Place & Route design‘ milestone post-script by Tcl command prj_set_postscript par <eco.tcl>, then Radiant flow runs the ECO Tcl script automatically after running place & route.

Command Line Syntax

ecoc [**-s** <*script_file*>] [**-o** <*output.udb*>] <in*put.udb*>**]**

Table 19: ECO Editor Command Line Options

Option	Description
-s	ECO Tcl script file.
-o	Output UDB file.
<*input.udb*>	Input UDB file.

Example

The following is an ecoc command line example:

```
ecoc -s mem.tcl ebr_test_impl_1.udb
```

See Also

- Engineering Change Order Tcl Commands
- Command Line Program Overview
- Command Line Data Flow

Standalone Timing Analyzer

The Standalone Timing Analyzer can be run from the command line.

Command Line Syntax

tavmain <udb file name> [<pdc file name>]

Example

The following is a tavmain command line example:

```
tavmain design.udb design.pdc
```

See Also

- Command Line Program Overview
- Command Line Data Flow

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 Linux 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.udb
-a -n 10
-w
-l 5
-s 2 #will save the two best results
/home/users/jimbob/designs/mine.udb
#output design name
/home/users/jimbob/designs/output.dir
#use timing constraint 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 Linux 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.udb design_o.udb

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 Linux-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 Linux 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 Linux 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 Linux 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 Linux 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 example shows running design “counter” on below device package

Architecture: iCE40UP

Device: iCE40UP3K

Package: UWG30

Performance: Worst Case

Command 1: logic synthesis

```
synthesis -f counter_impl1_lattice.synproj
     which the *.synproj contains
-a "iCE40UP"
-p iCE40UP3K
-t UWG30
-sp "Worst Case"
-optimization_goal Area
-bram_utilization 100
-ramstyle Auto
-romstyle auto
-dsp_utilization 100
-use_dsp 1
-use_carry_chain 1
-carry_chain_length 0
-force_gsr Auto
-resource_sharing 1
-propagate_constants 1
-remove_duplicate_regs 1
-mux_style Auto
-max_fanout 1000
-fsm_encoding_style Auto
-twr_paths 3
-fix_gated_clocks 1
-loop_limit 1950
-use_io_reg auto
-use_io_insertion 1
-resolve_mixed_drivers 0
-sdc "impl1.ldc"
-path "C:/lscc/radiant/1.0/ispfpga/ice40tp/data" "impl1"
-ver "C:/lscc/radiant/1.0/ip/pmi/pmi.v"
-ver "count_attr.v"
-path "."
-top count
-udb "counter_impl1.udb"
-output_hdl "counter_impl1.vm"
```

Command 2: post synthesis process

```
postsyn -a iCE40UP -p iCE40UP3K -t UWG30 -sp Worst Case -top -
ldc counter_impl1.ldc -keeprtl -w -o counter_impl1.udb
counter_impl1.vm
```

Command 3: Mapper

```
map "counter_impl1_syn.udb" "impl1.pdc" -o "counter_impl1.udb"
```

Command 4: Placer and router

```
par -f "counter_impl1.p2t" "counter_impl1_map.udb"
"counter_impl1.udb"
```

Command 5: Timer

```
timing -sethld -v 10 -u 10 -endpoints 10  -nperend 1 -html -rpt
"counter_impl1_twr.html" "counter_impl1.udb"
```

Command 6: back annotation

```
backanno "counter_impl1.udb" -n Verilog  -o
"counter_impl1_vo.vo"        -w -neg
```

Command 7: bitstream generation

```
bitgen -w "counter_impl1.udb" -f "counter_impl1.t2b"
```

Tcl Command Reference Guide

The Radiant software supports Tcl (Tool Command Language) scripting and provides extended Radiant software Tcl commands that enable a batch capability for running tools in the Radiant software's graphical/console 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 Radiant software tools are already designed to perform.

In addition to describing how to run the Radiant software's Tcl Console, this guide provides you with a reference for Tcl command line usage and syntax for all Radiant software point tools within the graphical user interface so that you can create command scripts, modify commands, or troubleshoot existing scripts.

About the Radiant software Tcl Scripting Environment

The Radiant software 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 Radiant software-specific support commands. The combination of fundamental Tcl along with the commands specialized for use with the Radiant software allow the entire Radiant software 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 Radiant software command interpreter provides an environment for managing your designs that are more abstract and easier to work with than

using the core Radiant software engines. The Radiant software 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.

Information about TCL features and flow are listed below:

- Launching the Tcl Console
- Running Radiant Tcl
  - Log and Tcl Files
  - Lattice Implementation Directory
  - Attributes
  - Understanding Design Flows
  - Using Implementation Strategies and Options
  - Running Project Flow
  - Running Non-Project Flow
  - Switching From Project Flow to Non-Project Flow
  - Running Milestone Results in Non-Project Flow
  - Opening GUI in Non-Project Flow
  - Design Flow Examples
- Accessing Command Help and Command Options in the Tcl Console
  - Changes in the Interface
- Creating and Running Custom Tcl Scripts
- Running Tcl Scripts When Launching the Radiant Software
- Radiant Software Tool Tcl Command Syntax
  - Radiant Software Tcl Console Commands
  - Radiant Software Timing Constraints Tcl Commands
  - Radiant Software Physical Constraints Tcl Commands
  - Synthesis Tcl Command
  - Design Object Tcl Commands
  - System Tcl Commands
  - Radiant Software Project Tcl Commands
  - Device Tcl Commands
  - Simulation Libraries Compilation Tcl Commands
  - Reveal Inserter Tcl Commands
  - Reveal Analyzer Tcl Commands
  - Power Calculator Tcl Commands
  - Engineering Change Order Tcl Commands
  - IP Version Update Tcl Commands

- Message Control Tcl Commands
- Physical Synthesis Tcl Commands

- Radiantc TCL Commands
  - Bitstream Generation Tcl Commands
  - Design Tcl Commands
  - Device Tcl Commands
  - Technology Mapping Tcl Commands
  - Place & Route Tcl Commands
  - Placement Tcl Commands
  - Routing Tcl Commands
  - Timing Analysis Tcl Commands

Additional References

If you are unfamiliar with the Tcl language you can get help by visiting the Tcl/tk web site at https://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 Radiant software processes, such as synthesis, map, par, backanno, and timing, see the Command Line Reference Guide.

Launching the Tcl Console

The Radiant software 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 Radiant software you must gain access to the Radiant Tcl Console user interface.

On Windows On Windows, you can interact with the Tcl Console by anyone of the following methods:

- To launch the Radiant software GUI from the Windows Start menu, choose **Start > Lattice Radiant Software (version_number) > Radiant Software**.

  After the Radiant 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 Radiant software-specific support commands.

- To launch the **Tcl Console** independently from the Radiant software GUI from the Windows Start menu choose **Start > Lattice Radiant Software (version_number) > Tcl Console**.

  A Windows command interpreter will be launched that automatically runs the **Tcl Console**.

- To run the interpreter from the command line, type the following:

  `c:/lscc/radiant/<version_number>/bin/nt64/radiantc`

The Radiant **Tcl Console** is now available to run.

- To run the interpreter from a Windows PowerShell from the Windows Start menu choose **Start > Windows PowerShell > Windows PowerShell (x86)**.

  A PowerShell interpreter window will open. At the command line prompt type the following:

  ```
  c:/lscc/radiant/<version_number>/bin/nt64/radiantc
  ```

  The Radiant **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 On Linux operating systems, you can interact with the Tcl Console by one of the following methods:

- To launch the Radiant software GUI from the command line, type the following:

  ```
  /usr/<user_name>/radiant/<version_number>/bin/lin64/radiant
  ```

  The path provided assumes the default installation directory and that the Radiant software is installed. After the Radiant 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 Radiant software specific support commands.

- To launch the **Tcl Console** independently from the Radiant software GUI from the command line, type the following:

  ```
  /usr/<user_name>/Radiant/<version_number>/bin/lin64/radiantc
  ```

  The path provided assumes the default installation directory and that the Radiant software is installed, and that you have followed the Radiant software for Linux installation procedures. The Radiant **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.

Running Radiant Tcl

This section describes the following features:

- How to run Radiant Tcl

- Using log/tcl files
- Accessing the Lattice run directory

Tcl supports the following command modes:

- **GUI Mode (radiant)** – you can use the **radiant** command to launch the GUI mode. Radiant's Tcl Console window is still accessible, however it currently only accepts project commands.
- **Pure interactive shell mode (radiantc)** – starts a pure interactive shell mode. If a tcl file is specified in the command line, the run is done in batch mode. Both project and non-project commands are supported by radiantc.
- **Interactive shell/GUI mode (radiantc -gui)** – starts as a shell mode. The **gui_start** command can be used to launch GUI views. When GUI views are closed, it returns to shell mode.

Log and Tcl Files

When using radiantc, the log message is automatically stored in the **radiantc.log.<pid>** file, and the tcl commands are kept in the **radiant.tcl.<pid>** file, where the <pid> is the process id for radiantc run.In the case the file "**radiantc.log.<pid>**" does exist, the file will be renamed to "**radiantc.log.<pid>.bak**". We do the same check for "**radiant.tcl.<pid>** "

The log message from Radiant (GUI) run is still stored in the **automake.log** file as before.

With radiantc mode, you can do following:

- Use the **RAT_LOG_DIR** environment variable to specify the directory where you want to keep the log/tcl files.

  If there is no environment variable, the log/tcl directory is set as *C:\Users\<User dir>\AppData\Roaming\LatticeSemi\DiamondNG\tcl* if you are using NT, and the current directory **./** when using Linux.
- Specify the run name to be used as the prefix of log/tcl file names.

  For example, if you run *radiantc -run mytest*, the log file will be named *mytest.log.<pid>* and the tcl file will be named *mytest.tcl*.<pid>.

Valid Characters in File Name and Project Path

In file names and project paths, use only the valid characters below. Using unsupported characters result in errors.

- Alphabet (A - Z; a - z)
- Number (0 - 9)
- Underscore (_)
- Minus (-)

- Space ( )

Notes:

1. Brackets [ ] in file names are acceptable in Windows but their use is not recommended.
2. For Linux, using the following symbols in file names are acceptable but not recommended:

- Vertical bar (|)
- Backslash (\)
- Greater than sign(>)
- Question mark (?)
- Less than sign (<)
- Colon (:)
- Asterisk (*)
- Quotation mark (")

Lattice Implementation Directory

In non-project flow, the lattice working directory **lattice_workdir** keeps the intermediate files during the run. The directory is created under your implementation directory.

- Under *lattice_workdir*, you can find map, plc, and rte subdirectories which keep the intermediate run results for technology map (map), placement (plc), and routing (rte).
- For example, if you want to find the placement log file *<file_name>.par*, you can find it under the *plc* subdirectory.

Attributes

In Radiant Tcl system, attributes are used for objects, such as system, design, or instance.

For example:

- *des_set_attribute* is used to set design attributes.
- *des_list_attribute* is used to list the current design attributes.

Understanding Design Flows

Radiant Tcl supports both project and non-project flows.

In project flow, you can do the following:

- Create a new project or load a saved project from an existing project file.
- You can then add, delete, or list project files (i.e., RTL, constraint, and strategy setting files). During the whole project flow, you can save it into a project (.rdf) file any time.

For non-project flow:

- With two specific commands, Radiant Tcl may start a non-project flow from an existing project flow. The Radiant database (.udb) file can be read as input to begin the non-project flow, after which the design process can continue.

There are four major differences between project and non-project flows:

1. The project flow design is saved as disk files instead of being stored in memory. You can use different strategies to create different implementations for the same project. On the other hand, design in non-project flow is kept in memory and can be accessed or changed interactively using commands.

   For example, you can use timing analysis commands to report timing results at different paths.

2. In project flow, you need to add project files. The time stamps of these files are checked for any new commands. If there are any changes in the project files, the flow will ask you to rerun the design at the right stages.

   For example, if you change a post-synthesis constraint (.pdc) file, you need to rerun technology mapping. However, if you change any RTL files or the pre-synthesis constraint files, you need to rerun logic synthesis.

   In non-project flow, the input file is a Radiant database (.udb) file, and the time stamp of such file is not checked during the design flow.

3. You cannot skip any design stage from the run commands in non-project flow.

   For example, you need to run placement (plc_run) before you run routing (rte_run) in non-project flow. However, you can use one of the prj_run commands (prj_run_bitstream, prj_run_map, prj_run_par, prj_run_synthesis) in project flow to run the design flow automatically from the current stage to the target stage.

4. Implementation strategies are used in project flow and implementation options in non-project flow.

   Implementation strategy is a subset of implementation options.

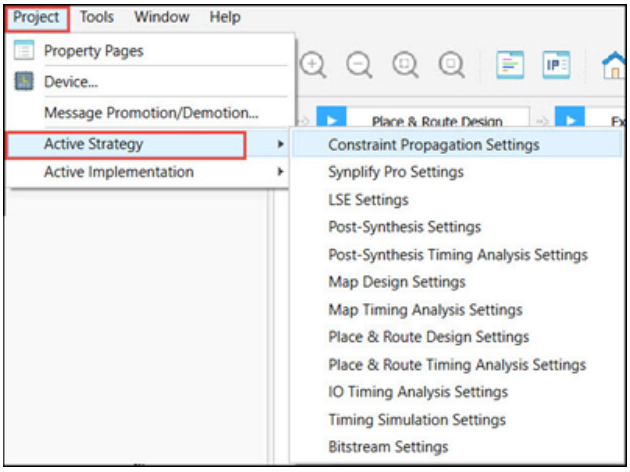
Using Implementation Strategies and Options

In Radiant, **strategies** are used to specify how to run specific tasks, such as logic synthesis, technology mapping, and placement and routing.

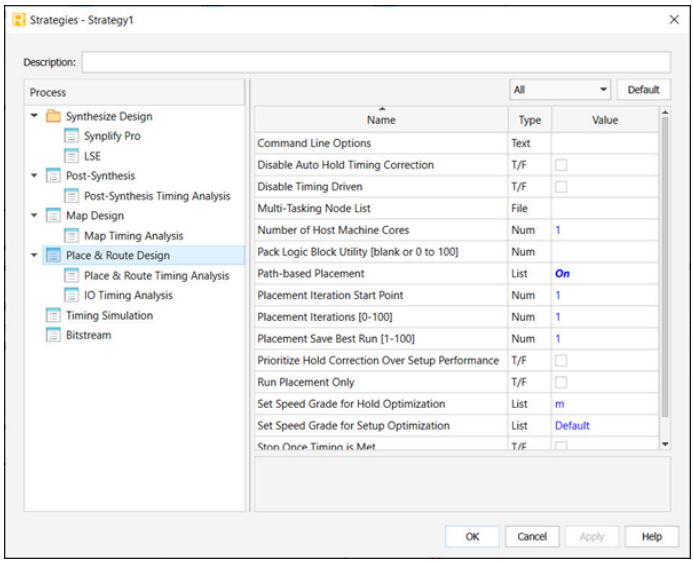
To see the list of strategies in Radiant:

- In the toolbar, click **Project > Active Strategy**.

  The list of strategies appear.



- For instance, if you click **Place & Route Design Settings** from the list, it redirects to the Place & Route Design section in the **Strategies** dialog box.



In TCL, **options** are used to specify how to run specific tasks.

- For example, you can use **par_list_option** after a design has been loaded to check the current placement and routing option setting, and use **par_set_option** to change them:

```
% par_list_option
PAR Options:
     command_line_options                              :
     disable_auto_hold_timing_correction               :
false
     disable_timing_driven                             :
false
```

```
     multi_tasking_node_list                                  :
     number_of_host_machine_cores                             : 1
     placement_iteration_start_point                          : 1
     placement_iterations                                     : 1
     placement_save_best_run                                  : 1
     prioritize_hold_correction_over_setup_performance        :
false
     run_placement_only                                       :
false
     set_speed_grade_for_hold_optimization                    : m
     set_speed_grade_for_setup_optimization                   :
default
     stop_once_timing_is_met                                  :
false
```

Strategy and Option Guidelines:

1. Both strategy and option are device-dependent. You can only change them after a device is selected.
2. You do not need to specify all options in the Radiant Strategy Settings. However, all strategies used in Radiant must be converted to options with the following rules:
   - If you are using options, change the space ( ) used in strategy to an underscore (_) to separate words.
   - In strategy, each word is capitalized. You need to change them to the lower case format if you are using options.
   - For example:
     - Strategy uses the following format: **Disable Auto Hold Timing Correction**
     - Option uses the following format: **disable_auto_hold_timing_correction**

Running Project Flow

Running the project flow in Radiant Tcl is the same as the current process. You can use the Radiant software tools to add project files, run the design flow, and output specific timing reports at different design stages.

Most *prj_<command_name>* commands are designed for project flow run except **prj_open_milestone**. This command is used to switch from project flow to non-project flow.

prj_open_milestone

Commands to open project milestone netlist files.

Syntax

```
prj_open_milestone -postsyn | -postmap | -postpar
```

Arguments

- **-postsyn:** Open post-syn logical netlist (optional).
- **-postmap:** Open post-map physical netlist (optional).
- **-postpar:** Open post-par physical netlist (optional). Option '-postsyn' '-postmap' and '-postpar' are mutually exclusive.

Example

```
prj_open_milestone -postsyn
prj_open_milestone -postmap
prj_open_milestone -postpar
```

See Also

- “Running Project Flow” on page 1564

Running Non-Project Flow

You must load the design from a disk file in order to start a non-project flow. For instance, the following commands load a *mapped.udb* file into memory, report the current design, and run placement.

```
des_read_udb mydb_map.udb
des_report
plc_run
```

Switching From Project Flow to Non-Project Flow

In project flow, you can use **prj_open_milestone** to open the run result at a given design stage and switch to non-project flow.

For instance, you can see the **postsyn** run result after opening the *mydb_syn.rdf* project by using the commands listed below.

The design is currently in memory, and you can report and save it as a Radiant database (.udb) file.

```
prj_open mydb_syn.rdf
prj_open_milestone -postsyn
des_report
des_write_udb -w mydb_syn.udb
```

Using **syn_run** instead of **prj_run_synthesis** for logic synthesis is another way to transition to a non-project flow from a project flow.

- When **syn_run** is used, the tool will first execute logical synthesis using the designated vendor tool before loading the post-synthesis design into memory.
- After that, the flow switches to non-project flow, and you can use non-project commands to complete the rest of the design process.

```
prj_create -name mytest -dev "LIFCL-40-8BG400C" -performance
"8_High-Performance_1.0V" -impl impl_prj_flow -impl_dir
impl_prj_flow -synthesis synplify
prj_add_source ./source/001_register_simple.v
prj_add_source ./others/constraints.sdc
prj_add_source ./others/register_simple1.sty
syn_run
des_report
des_write_udb -w mydb_syn.udb
```

With one of the two methods described above, you can change from project flow to non-project flow. Once the design is in memory (i.e., in non-project flow mode), it is not advised to use project commands.

Running Milestone Results in Non-Project Flow

In non-project flow, the current design is stored in memory, unlike in project flow which keeps previous designs at different milestones. To rerun a previous step due to constraint or option changes, you need to reload the previous milestone result and make a change, then rerun the command.

Note

Milestone refers to processing tasks, which include synthesizing, mapping, placing, and routing a design.

To rerun *par_run* with new par options, you need to do the following:

```
% par_run
% des_write_udb -w mydb_par.udb
% des_reload_milestone -postmap <your_mapped_udb>
% par_set_option (set your par run option)
% par_run
```

To add a new constraint as a post-synthesis constraint, reload the post-syn milestone as the constraints need to be processed through technology mapping.

```
% par_run
% des_write_udb -w mydb_par.udb
% des_reload_milestone -postsyn <your_synthesized_udb>
% ldc_set_location (set your constraint)
% map_run
```

```
% par_run
```

To modify timing constraints for logical synthesis, change your constraint file and restart the process from the beginning.

Opening GUI in Non-Project Flow

In order to open Radiant GUI views in non-project flow, you must run **radiantc** with **-gui** option to enable the linking of the GUI libraries during the run. You can open a **post-par udb** and use **gui_start** command to open GUI views.

```
>radiantc -gui
% des_read_udb <your_post_par.udb>
% gui_start
```

Certain GUI views require a post-synthesis netlist as the netlist view, which can be added by incrementally reloading the post-synthesis udb.

```
>radiantc -gui
% des_read_udb <your_post_par.udb>
% des_reload_milestone -inc -postsyn <your_synthesized_udb>
% gui_start
```

Design Flow Examples

1. **Running Pure-Project Flow**

   Tcl commands for pure-project run is shown in this use-case. To run a pure-project, add Register Transfer Level (RTL) files, constraint files then run the project using one of the prj_run commands (prj_run_bitstream, prj_run_map, prj_run_par, prj_run_synthesis) and save the project files at different stages.

   Note

   Pure-project design flow is saved as disk file and not kept in memory. As a result, you cannot run any non-project commands in a project flow. For further information, refer to Opening a Post-Synthesis Milestone.

   Open your console and type the following commands to create, add, run and exit the program:

   - To create a project, use prj_create command followed by choosing the right device and synthesis tool name:

     ```
     prj_create -name mytest -dev "LIFCL-40-8BG400C" -
     performance "8_High-Performance_1.0V" -impl impl_prj_flow
     -impl_dir impl_prj_flow -synthesis synplify
     ```

   - To add project files, use prj_add_source command followed by the path of the file you want to add:

```
prj_add_source ./source/001_register_simple.v
prj_add_source ./others/constraints.sdc
prj_add_source ./others/register_simple1.sty
```

- To save an updated project file, use prj_save command:

```
prj_save mydb_syn.rdf
prj_save new_test.rdf
prj_save mydb_map.rdf
prj_save mydb_par.rdf
```

- To run synthesis project files, use the prj_run_synthesis command.

```
prj_run_synthesis
```

- To end the program, quit the command to end the flow:

```
quit
```

2. Creating Project Files

This section illustrates how to create project files, add Register Transfer Level (RTL) file, and constraint files.

- To create a project, use prj_create command followed by choosing the right device and synthesis tool name:

```
prj_create -name mytest -dev "LIFCL-40-8BG400C" -
performance "8_High-Performance_1.0V" -impl impl_prj_flow
-impl_dir impl_prj_flow -synthesis synplify
```

- To add project files, use prj_add_source command followed by the file directory path you want to add:

```
prj_add_source ./source/001_register_simple.v
prj_add_source ./others/constraints.sdc
prj_add_source ./others/register_simple1.sty
```

- To switch from project flow to non-project flow, use the **syn_run** command:

```
syn_run
```

  Use **syn_run** instead of **prj_run_synthesis** to change from project flow to non-project flow. After execution, the design is in memory and you can continue the rest of the flow with non-project commands.

Note

It is not recommended to run the flow with any of the prj_run commands (prj_run_synthesis, prj_run_map, prj_run_par, prj_run_bitstream) once switched to non-project flow.

- To run the design information, use des_report command:

```
des_report
```

- To run the map tool, use map_run command, followed by des_write_udb -w command to save the current design:

```
map_run
des_write_udb -w mydb_map.udb
```

- To run the placement tool and show the report placement information, use plc_run and plc_report commands followed by des_write_udb -w command to save the current design:

```
plc_run
plc_report
des_write_udb -w mydb_plc.udb
```

- To run the routing tool and show the routing result, use rte_run and rte_report commands followed by des_write_udb -w command to save the current design:

```
rte_run
rte_report
des_write_udb -w mydb_rte.udb
```

- To view the detailed report or summary timing, use the following command:

```
sta_report_timing -summary
```

- To generate bitsream file, use bit_generate command followed by -w command to force to overwrite the existing file:

```
bit_generate -w mydb_bit
```

- To end the program, quit the command to end the flow:

```
quit
```

3. Reading Post-Synthesis Design

This section shows you how to read in a design database and continue the design flow.

- To read the post-synthesis database, use des_read_udb command followed by the database file name. To view the report design information, use des_report command:

```
des_read_udb mydb_syn.udb
des_report
```

- To set design attributes, use des_set_attribute command followed by -impl_dir to implement name:

```
des_set_attribute -impl_dir impl_rdb_syn
```

- To run the map tool and to write the design as .udb file, use map_run command followed by des_write_udb and overwrite to save using -w command to completely run the design:

```
map_run
des_write_udb -w myrdbsyn_map.udb
```

- To run the placement tool and show the report placement information, use plc_run command and plc_report command followed by des_write_udb -w command to save the current design:

```
plc_run
plc_report
des_write_udb -w myrdbsyn_plc.udb
```

- To run the routing tool and show the routing result, use rte_run and rte_report commands followed by des_write_udb -w command to save the current design:

```
rte_run
rte_report
des_write_udb -w myrdbsyn_rte.udb
```

- To view the detailed report or summary timing, use sta_report_timing -summary command:

```
sta_report_timing -summary
```

- To generate bitstream file, use bit_generate command followed by -w command to force to overwrite the existing file:

```
bit_generate -w mydb_bit
```

- To end the program, quit the command to end the flow:

```
quit
```

4. Reading the Post-Map UDB File

This section shows you how to read a post-map .udb file and continue the design process.

- To read the post-map UDB file, use des_read_udb command followed by the database file name. In the next line, use des_set_attribute command followed by - impl_dir to define the directory where the project file is stored:

```
des_read_udb mydb_map.udb
des_set_attribute -impl_dir impl_udb_map
```

- To run the placement tool and show the report placement information, use plc_run and plc_report commands followed by des_write_udb -w command to save the current design:

```
plc_run
plc_report
des_write_udb -w myudbmap_plc.udb
```

- To end the program, quit the command to end the flow:

```
quit
```

5. Reading and Routing the Post-Placement Design

This section describes the process of reading the post-placement database, routing, and saving the design.

- To read the post-synthesis database, use des_read_udb command followed by the database file name. Execute using des_set_attribute command followed by - impl_dir to define the directory where the project file is stored.

```
des_read_udb mydb_plc.udb
des_set_attribute -impl_dir impl_rdb_plc
```

- To run the routing tool and show the routing result, use rte_run and rte_report commands followed by des_write_udb -w command to save the current design.

  Use sta_report_timing - summary command to view the detailed report or timing summary and use des_write_udb command with -w command to overwrite or save the design.

```
rte_run
rte_report
sta_report_timing -summary
des_write_udb -w myrdbplc_rte.udb
```

- To end the program, quit the command to end the flow.

```
quit
```

6. Opening a Post-Synthesis Milestone

This section shows how to open a post-synthesis milestone in a project, and load the run result in memory. When the design is in memory, you can continue to use non-project commands to continue the flow and access the design data.

- To open a project, use prj_open command followed by the file name.

```
prj_open mydb_syn.rdf
```

- To open a project milestone, use prj_open_milestone command followed by - postsyn to open the logical netlist.

```
prj_open_milestone -postsyn
```

- To run the map tool and write the design as a .udb file, use map_run command followed by des_write_udb and save using -w command to run the design.

```
map_run
des_write_udb -w myprjsyn_map.udb
```

- To end the program, quit the command to end the flow

```
quit
```

See Also

- Launching the Tcl Console
- Running Radiant Tcl
- Accessing Command Help and Command Options in the Tcl Console
- Creating and Running Custom Tcl Scripts
- Running Tcl Scripts When Launching the Radiant Software
- Radiant Software Tool Tcl Command Syntax
- Tcl Manual

Accessing Command Help and Command Options in the Tcl Console

There are two ways to find all the Tcl command syntax help to generate all the tools in the Tcl console.

To access command syntax help in the Tcl Console:

1. n the prompt, type **help <tool_*keyword*>** and press **Enter:**

   ```
   % help {des_*instance*}
   ```

   A list of valid command options appears in the Tcl Console.

2. In the Tcl Console, type the name of the command or function for more details on syntax and usage.

   For example:

   - des_list_instance is used to list design instances.
   - des_report_instance is used to report instance information.

   A list of valid arguments for that function appears.

   ```
   % help des_report_instance
   Usage information:
       Var/FlagName Type   Value Help
       ------------ ----   ----- ----
       (-help                    gives this help)
       -file        string ()    File name

       ?objects?    string ()    List instances.
   ```

Note

In interactive mode, you can use command **-help** to get the list of command information. However, most commands in the -help option are not true option, and the Tcl interpreter interprets command -help as a usage error, which prevents a script from finishing execution if it encounters this option. It is recommended to use **help {command}** as this works in both interactive and script settings.

Changes in the Interface

The Radiant Tcl has the same Windows settings and appearance as the Radiant Tcl, with some GUI improvements.

- Running the **radiantc** replaces the standalone tools such as Map, PAR, and Bitgen for milestone runs for **Map Design, Place & Route,** and **Export Files** using specific Tcl files, which are stored in the implementation directory as **<prj>_<impl>_map/par/bit.tcl.**
- When you open the milestone run result, the **File > Open** menu displays the **Post-Synthesis Milestone, Post-Map Milestone**, and **Post-Par Milestone**, allowing you to perform non-project commands such as des_report in the new **radiantc** terminal.

See Also

- Launching the Tcl Console
- Running Radiant Tcl
- Accessing Command Help and Command Options in the Tcl Console
- Creating and Running Custom Tcl Scripts
- Running Tcl Scripts When Launching the Radiant Software
- Radiant Software Tool Tcl Command Syntax
- Tcl Manual

Creating and Running Custom Tcl Scripts

This topic describes how to easily create Tcl scripts using the Radiant software'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, .vm 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 flexibility 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 the Radiant software Tcl scripts. This section will discuss each one and provide step-by-step instructions for you to get started Tcl scripting repetitious Radiant software commands or entire workflows.

One method you have available is to use your favorite text editor to enter a sequence of the Radiant software Tcl commands. The syntax of each the Radiant software Tcl commands is available in later topics in this portion of the Help. This method should only be used by very experienced Radiant software Tcl command line users.

The preferred method is to let the Radiant software GUI assist you in getting the correct syntax for each Tcl command. When you interact with the Radiant software user interface each time you launch a *scriptable* process and the corresponding Radiant software Tcl command is echoed to the Tcl Console. This makes it much simpler to get the correct command line syntax for each Radiant software 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 the Radiant software:

1. Start the Radiant software 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 Radiant software 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 bar 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 Radiant software 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 Radiant Software Tcl Script

The following Radiant software Tcl script shows a 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 1: Sample Radiant Software Script

```
prj_archive -dir "C:/my_radiant/counter" -extract "C:/lscc/
radiant/1.1/examples/counter.zip"
prj_run_par
prj_close
```

Running Tcl Scripts

The Radiant software Tcl scripts are run exclusively from the Radiant Tcl Console. You can use either the Tcl Console integrated into the Radiant software UI, or by launching the stand-alone Tcl Console.

To run a Tcl script in the Radiant software:

1. Launch the Radiant software GUI, or the stand-alone Tcl Console.

   Open the Radiant software but do not open your project. If your project is open, choose **File > Close Project**.

2. If you are using the Radiant software main window, click the small arrow pane switch in the bottom of the Radiant software main window, and then 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/radiant/<version_number>/examples/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

- Launching the Tcl Console
- Running Radiant Tcl
- Accessing Command Help and Command Options in the Tcl Console
- Creating and Running Custom Tcl Scripts
- Running Tcl Scripts When Launching the Radiant Software
- Radiant Software Tool Tcl Command Syntax
- Tcl Manual

Running Tcl Scripts When Launching the Radiant Software

This topic describes how launch the Radiant software and automatically run Tcl scripts using a command line shell or the stand-alone Tcl console. Your Tcl script can be standard Tcl commands as well as the Radiant software-specific Tcl commands.

Refer to Creating and Running Custom Tcl Scripts for more information on creating custom Tcl scripts.

To launch the Radiant software and run a Tcl script from a command line shell or the stand-alone Tcl console:

- Enter the following command:

  On Windows:

  ```
  radiant.exe -t <tcl_path_file>
  ```

  On Linux:

  ```
  radiant -t <tcl_path_file>
  ```

Sample Radiant software Tcl Script The following Radiant software 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 2: Simple Radiant Software Script

```
prj_open C:/test/iobasic_radiant/io1.rdf
prj_run_map
```

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 Radiant software GUI starts, the project io1.rdf opens, and the MAP process automatically runs.

```
radiant.exe -t c:/test/mytcl.tcl
```

See Also

- Launching the Tcl Console
- Running Radiant Tcl
- Accessing Command Help and Command Options in the Tcl Console
- Creating and Running Custom Tcl Scripts
- Running Tcl Scripts When Launching the Radiant Software
- Radiant Software Tool Tcl Command Syntax
- Tcl Manual

Radiant Software Tool Tcl Command Syntax

This part of the Tcl Command Reference Guide introduces the syntax of each of the Radiant software tools and provides you with examples to help you construct your own commands and scripts.

The Radiant software 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 the Radiant software.

The Tcl Commands are broken into major categories. The major categories are:

- Radiant Software Tcl Console Commands
- Radiant Software Timing Constraints Tcl Commands
- Radiant Software Physical Constraints Tcl Commands
- Technology Mapping Tcl Commands
- Synthesis Tcl Command
- Design Object Tcl Commands
- System Tcl Commands
- Radiant Software Project Tcl Commands
- Device Tcl Commands
- Place & Route Tcl Commands

- Device Tcl Commands
- Place & Route Tcl Commands
- Simulation Libraries Compilation Tcl Commands
- Reveal Inserter Tcl Commands
- Reveal Analyzer Tcl Commands
- Power Calculator Tcl Commands
- Engineering Change Order Tcl Commands
- IP Version Update Tcl Commands
- Message Control Tcl Commands
- Physical Synthesis Tcl Commands
- Physical Synthesis Tcl Commands
- Radiantc TCL Commands

Radiant Software Tcl Console Commands

The Radiant software Tcl Console provides a small number of commands that allow you to perform some basic actions upon the Tcl Console Pane. The Radiant software Tcl Console commands differ from the other Tcl commands provided in the Radiant software. 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 the Radiant software.

Note

Tcl Command Log is always listed after the project is closed. You can find it in the Reports section under Misc Report > Tcl Command Log.

The following table provides a listing of all valid Radiant software Tcl Console-related commands.

Table 20: Radiant Software Tcl Console Commands

Command	Description
clear	Erase anything present in the Tcl Console pane and print the current *prompt* character in the upper left corner of the Tcl Console pane without erasing the command history.
history	List 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	Clear the Tcl Console pane and erase all entries in the command line history. **This command is only used in the GUI Tcl console and not supported in the standalone Tcl console.
save_script	Save 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. Usage: save_script <filename.ext> **This command is only used in the GUI Tcl console and not supported in the standalone Tcl console.
set_prompt	Set the default prompt 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. Usage: set_prompt <new_character> **This command is only used in the GUI Tcl console and not supported in the standalone Tcl console.

Radiant Software Tcl Console Command Examples

This section illustrates and describes a few samples of Radiant Tcl Console commands.

Example 1

To save a script, 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/radiant/myproject/scripts/myscript.tcl
save_script myscript.tcl
```

See Creating and Running Custom Tcl Scripts for details on how to save and run scripts in the Radiant software.

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
```

Radiant Software Timing Constraints Tcl Commands

The section describes the commands are used to set timing constraints.

Radiant Software Timing Constraints Tcl Command Descriptions

The following table provides a listing of all valid Radiant software timing constraints-related Tcl command options and describes option functionality.

Table 21: Radiant Software Timing Constraints Tcl Commands

Command	Description
create_clock	Create a named or virtual clock.
create_generated_clock	Create a generated clock object.
set_clock_groups	Set clock groups.

Table 21: Radiant Software Timing Constraints Tcl Commands

Command	Description
set_clock_latency	Define a clock's source or network latency.
set_clock_uncertainty	Set clock uncertainty.
set_false_path	Define false path.
set_input_delay	Set input delay on ports.
set_load	Set capacitance on ports.
set_max_delay	Specify maximum delay for timing paths.
set_max_skew	Define maximum skew of path.
set_min_delay	Specify minimum delay for timing paths.
set_multicycle_path	Define multicycle path.
set_output_delay	Set output delay on ports.
set_hierarchy_separator	Set or get hierarchy separator.

create_clock

Commands to create a new clock object and define its characteristics.

Syntax

```
create_clock -period <period_value> [-name <clock_name>]
    [-waveform {<value1 value2>}] [<source_object>] [-add]
```

Arguments

- **-period period_value:** This value 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.
- **-name clock_name:** This 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 (optional).
- **-waveform {value1 value2}:** These 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. Note that value1 must also be less than the clock period (optional).
- **-source_object:** This value is the object on which the clock constraint is defined. The source object can be a port, pin or a net object in the design. If you specify a clock constraint on a source object that already has a clock, the new clock replaces the existing one (optional).

- **-add:** This option allows defining another clock on an object that already has a clock (optional).

Example 1

```
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 2

```
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

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

create_generated_clock

Commands to create an internally generated clock object and define its characteristics.

Syntax

```
create_generated_clock [-name <clock_name>] -source
  <source_object> [-edges {<value1 value2 value3>}]
  [-edge_shift{<value1 value2 value3>]
  [-divide_by <factor>][-multiply_by <factor>]
  [-duty_cycle <percent>] [-invert] [-add]
  [-master_clock <clock>] <object_list>
```

Arguments

- **-name clock_name:** This name string specifies the name of the generated clock (optional).
- **-source source_object:** This value is an object on which the source clock of the generated clock is defined. The source object can be a port, pin or a net object. The period of the generated clock is derived from the clock on the source object using the multiply and divide factors.
- **-edges {value1 value2 value3}:** These values are a list of three numbers which correspond to the edge of the source clock from which the generated clock has been obtained. This option cannot be used with '-divide_by', '-multiply_by' and '-duty_cycle' (optional).
- **-edge_shift {value1 value2 value3}:** These values are a list of three numbers that represents the amount of shift (optional).

- **-divide_by factor:** This value specifies the frequency division factor. Default value is 1. 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 (optional).
- **-multiply_by factor:** This value specifies the frequency multiplication number to be used when finding the generated clock frequency. Default value is 1 (optional).
- **-duty_cycle factor:** 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. This option cannot be used with '-divide_by' (optional).
- **-invert:** This option inverts the clock edge (optional).
- **-add:** This option allows defining another clock on an object that already has a generated clock.
- **-master_clock clock:** This value specifies the master clock for the generated clock object if there are multiple clocks found on the source object (optional).
- **-object_list:** This value is a list of clock source pins,ports,or nets.

Example 1

```
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]
```

Example 2

```
The following example creates a generated clock at the
primary output of myPLL with a period three quarters 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]
```

Example 3

```
The following example shows a clock with a duty cycle of
60percent:
create_generated_clock -duty_cycle 60 -source
[get_ports clk]  [get_pins myPLL/CLK1]
```

See Also

- “Radiant Software Timing Constraints Tcl Commands” on page 1580

set_clock_groups

Commands to specify 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 (-logically_exclusive |
-physically_exclusive | -asynchronous)
[-group <clock_objects>] [-group <clock_objects>]
```

Arguments

- **-logically_exclusive:** This option specifies that clocks are mutually exclusive and will not reach the flip flops at the same time due to logic implementation (optional).
- **-physically_exclusive:** This option specifies that clocks are mutually exclusive. Only one clock group is active at any given time (optional).
- **-asynchronous:** This option specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. One option among '-logically_exclusive', '-physically_exclusive' and '-physically_exclusive' is needed and they are mutually exclusive (optional).
- **-group clock_objects:** This value specifies the clock objects in a group. At least one '-group' is needed. Each group of clocks is exclusive or asynchronous with the clocks which are specified in the other group. Specifying one group indicates that the clocks in this group are exclusive or asynchronous with all other clocks in the design including one clock which is created in the future (optional).

Example 1

```
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]
```

Example 2

```
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 -logically_exclusive -group
[get_clocks
{clka_port clkb_port}] -group [get_clocks clkc_port]
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_clock_latency

Commands to define a clock's source or network latency.

Syntax

```
set_clock_latency [-rise] [-fall] [-early | -late] -source
<latency> <object_list>
```

Arguments

- **-rise**: This option specifies the latency for the rising clock edge (optional).
- **-fall:** This option specifies the latency for the falling clock edge (optional).
- **-early|-late:** This option specifies how early|late the clock edge arrives. Option '-early' and '-late' are mutually exclusive (optional).
- **-source:** This option indicates the specified latency as a source latency.
- **<latency>:** This value specifies the value of latency.
- **<object_list>:** This value is a list of clocks. You can also use the keyword '[all_clocks]' to include all clocks.

Examples

```
In the following example, 3ns is in the destination clock of
setup and source clock of hold;
while 4ns is in source clock of setup and destination clock
of hold.
     set_clock_latency 3 -source -early [get_clocks clk1]
     set_clock_latency 4 -source -late [all_clocks]
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_clock_uncertainty

Commands to indicate that the clock of interest has uncertainties in its period.

Syntax

```
set_clock_uncertainty [-setup] [-hold] [-from <clock>]
[-to <clock>] <uncertainty> [<clock_list>]
```

Arguments

- **-setup:** This option indicates the constraint for setup analysis (optional).
- **-hold:** This option indicates the constraint for hold analysis (optional).
- **-from clock:** This value specifies uncertainty source clock and the timer requires the ‘from’ option to be accompanied by a ‘to’ option (optional).
- **-to clock:** This value specifies uncertainty destination clock (optional).
- **<uncertainty>:** This value specifies the amount by which the clock period in uncertain.
- **<clock_list>:** This value is a list of clocks. You can also use the keyword '[all_clocks]' to include all clocks. One option between '-from' and 'clock_list' is needed and they are mutually exclusive (optional).

Examples

```
The following two examples specify clock uncertainty,
one with a from/to. The first constraint sets
the uncertainty of clk1 to half a nanosecond and
the second constraint sets the uncertainty to 400
picoseconds when a path is launched by clk2 and
captured by clk3.
     set_clock_uncertainty 0.5 [get_clocks clk1]
     set_clock_uncertainty 0.4 -from [get_clocks clk2] -to
        [get_clocks clk3]
```

See Also

- “Radiant Software Timing Constraints Tcl Commands” on page 1580

set_false_path

Commands to identify paths that are considered false and excluded from timing analysis.

Syntax

```
set_false_path [-from
<port_list|pin_list|instance_list|clock_list>]
[-to <port_list|pin_list|instance_list|clock_list>]
[-through <port_list|pin_list|instance_list|net_list>]
[-rise_from <clock_list>] [-rise_to <clock_list>]
[-fall_from <clock_list>] [-fall_to <clock_list>]
[-setup] [-hold]
```

Arguments

- **-from port_list|pin_list|instance_list|clock_list:** This value specifies the timing path start points (optional).
- **-to port_list|pin_list|instance_list|clock_list:** This value specifies the timing path end points (optional).
- **-through object_list:** This value is a list of through points which the paths should be blocked (optional).

- **-rise_from|fall_from clock_list:** Same as '-from' but only applies to clock objects. Option '-from', 'rise_from' and '-fall_from' are mutually exclusive.
- **-rise_to|fall_to object_list clock_list:** Same as '-to' but only applies to clock objects. Option '-to', 'rise_to' and '-fall_to' are mutually exclusive. At least one option among '-from|-rise_from|-fall_from|-to|-rise_to|-fall_to|-through' is needed.
- **-setup:** This option indicates to eliminate setup timing analysis for specified timing paths (optional).
- **-hold:** This option indicates to eliminate hold timing analysis for specified timing paths (optional).

Example 1

```
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:
set_false_path -from [get_ports clk1] -to [get_cells reg_2]
```

Example 2

```
The following example specifies all paths through the net
UO/sigA as false: set_false_path -through
[get_nets UO/sigA]
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_input_delay

Commands to define the arrival time of an input relative to a clock.

Syntax

```
set_input_delay -clock <clock_object> [-clock_fall]
[-max |-min] [-add_delay] <delay_value> <port_list>
```

Arguments

- **-clock clock_object:** This value specifies the clock reference to which the specified input delay is related.
- **-clock_fall:** This option is a switch to specify the falling edge of clock to trigger (optional).
- **-max:** This option specifies that the delay value is the maximum delay (optional).
- **-min:** This option specifies that the delay value is the minimum delay. Option '-max' and '-min' are mutually exclusive (optional).

- **add_delay:** This option is used to define more than one input delay on a given port (optional).
- **<delay_value>:** This 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.
- **<port_list>:** This value 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 1

```
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] set_input_delay 4.0 -clock
[get_clocks vclk] -add_delay [get_ports out2]
```

Example 2

```
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

- “Radiant Software Timing Constraints Tcl Commands” on page 1580

set_load

Commands to set capacitance on ports.

Syntax

```
set_load <capacitance> <objects>
```

Arguments

- **<capacitance>:** This value specifies the load in capacitance in pF.
- **<objects>:** This value provides one or more output ports in the current design to which capacitance is assigned. You can also use the keyword '[all_outputs]' to include all output ports.

Example

```
The following example sets a capacitive load of 25 pF
for output port OUT:
set_load 25 [get_ports {out}]
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_max_delay

Commands to specifies the maximum delay for the timing paths.

Syntax

```
set_max_delay
[-from <port_list|pin_list|instance_list|clock_list>]
[-to <port_list|pin_list|instance_list|clock_list>]
[-through <port_list|pin_list|instance_list|net_list>]
[-datapath_only] [-rise_from <clock_list>]
[-rise_to <clock_list>] [-fall_from <clock_list>]
[-fall_to <clock_list>] <delay_value>
```

Arguments

- **-from port_list|pin_list|instance_list|clock_list:** This value specifies the timing path start points (optional).
- **-to port_list|pin_list|instance_list|clock_list:** This value specifies the timing path end points (optional).
- **-through object_list:** This value specifies the timing path pass through points. The timing paths must go through these points (optional).
- **-rise_from|fall_from clock_list:** Same as '-from' but only applies to clock objects. Option '-from', 'rise_from' and '-fall_from' are mutually exclusive.
- **-rise_to|fall_to clock_list:** Same as '-to' but only applies to clock objects. Option '-to', 'rise_to' and '-fall_to' are mutually exclusive. At least one option among '-from|-rise_from|-fall_from|-to|-rise_to|-fall_to|-through' is needed.
- **-datapath_only:** This option calculates the Clock-to-Q delay of the first flop, the total delay between the flops, and the setup time of the second flop (optional).
- **<delay_value>:** This value specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

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 -from [get_cells ff1a] -to
[get_cells ff2e] 5.0 set_max_delay -from
[get_pins ff1/Q] -through [get_pins and1/B] -to
[get_pins ff2/D] 7.0
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_max_skew

Commands to define the max skew of a path.

Syntax

```
set_max_skew <net_list|pin_list> <skew_value>
```

Arguments

- **<net_list|pin_list:** This value is a list of pins or nets.
- **<skew_value>:** This value controls the delay to the various loads of the net relative to each other.

Examples

```
The following example shows that the first constraint
requires the maximum difference between the net delays
of the loads on the net connected to pin "A" to be less
than or equal to 250 picoseconds. The second constraint
requires the difference between the net delays to the
various loads of the net "ff1/Q_c" to be less than or
equal to 370 picoseconds:
set_max_skew [get_pins A] 0.25
set_max_skew [get_nets ff1/Q_c] 0.37
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_min_delay

Commands to specifie the minimum delay for the timing paths.

Syntax

```
set_min_delay
[-from <port_list|pin_list|instance_list|clock_list>]
[-to <port_list|pin_list|instance_list|clock_list>]
[-through <port_list|pin_list|instance_list|net_list>]
[-rise_from <clock_list>] [-rise_to <clock_list>]
[-fall_from <clock_list>] [-fall_to <clock_list>]
<delay_value>
```

Arguments

- **-from port_list|pin_list|instance_list|clock_list:** This value specifies the timing path start points (optional).
- **-to port_list|pin_list|instance_list|clock_list:** This value specifies the timing path end points (optional).

- **-through object_list:** This value specifies the timing path pass through points. The timing paths must go through these points (optional).
- **-rise_from|fall_from clock_list:** Same as '-from' but only applies to clock objects. Option '-from', 'rise_from' and '-fall_from' are mutually exclusive.
- **-rise_to|fall_to object_list:** Same as '-to' but only applies to clock objects. Option '-to', 'rise_to' and '-fall_to' are mutually exclusive. At least one option among '-from|-rise_from|-fall_from| -to|-rise_to|-fall_to|-through' is needed.
- **<delay_value>:** This value specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

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 -from [get_cells ff1a] -to
[get_cells ff2e] 5.0 set_min_delay -from
[get_pins ff1/Q] -through [get_pins and1/B] -to
[get_pins ff2/D] 7.0
```

See Also

- “Radiant Software Timing Constraints Tcl Commands” on page 1580

set_multicycle_path

Commands to define a path that takes multiple clock cycles.

Syntax

```
set_multicycle_path
[-from <port_list|pin_list|instance_list|clock_list>]
[-to <port_list|pin_list|instance_list|clock_list>]
[-through <port_list|pin_list|instance_list|net_list>]
[-rise_from <clock_list>] [-rise_to <clock_list>]
[-fall_from <clock_list>] [-fall_to <clock_list>]
[-setup | -hold] [-start | -end] <path_multiplier>
```

Arguments

- **-from port_list|pin_list|instance_list|clock_list:** This value specifies the timing path start point (optional).
- **-to port_list|pin_list|instance_list|clock_list:** This value specifies the timing path end point (optional).
- **-through port_list|pin_list|instance_list|net_list:** This value specifies the timing path pass through point. The timing path must go through this object (optional).
- **-rise_from|fall_from clock_list:** Same as '-from' but only applies to clock objects. Option '-from', 'rise_from' and '-fall_from' are mutually exclusive.

- **-rise_to|fall_to clock_list:** Same as '-to' but only applies to clock objects. Option '-to', 'rise_to' and '-fall_to' are mutually exclusive. At least one option among '-from|-rise_from|-fall_from|- to|-rise_to|-fall_to|-through' is needed.
- **-setup:** This option set Setup multiplier (optional).
- **-hold:** This option set Hold multiplier. Option '-setup' and '-hold' are mutually exclusive (optional).
- **-start:** This option indicates that Multiplier measured against startpoint (optional).
- **-end:** This option indicates that Multiplier measured against endpoint. Option '-start' and '-end' are mutually exclusive (optional).
- **<path_multiplier>:**This value specifies a value that represents the number of cycles.

Examples

```
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]
set_multicycle_path 3 -from [get_pins ff1/Q] -through
[get_pins and1/B] -to [get_pins ff2/D]
```

See Also

- “Radiant Software Timing Constraints Tcl Commands” on page 1580

set_output_delay

Commands to define the output delay of an output relative to a clock.

Syntax

```
set_output_delay -clock <clock_object> [-clock_fall]
[-max | -min] [-add_delay] <delay_value> <port_list>
```

Arguments

- **-clock clock_object:** This value specifies the clock reference to which the specified output delay is related.
- **-clock_fall:** This option is a switch to specify the falling edge of clock to trigger (optional).
- **-max:** This option specifies that the delay value is the maximum delay (optional).
- **-min:** This option specifies that the delay value is the minimum delay. Option '-max' and '-min' are mutually exclusive (optional).
- **-add_delay:** This option is used to define more than one output delay on a given port (optional).

- **<delay_value>:** This value specifies the amount of time from a reference clock to a primary output port.
- **<port_list>:** This value provides one or more 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 1

```
The following example sets an output delay of 1.2 ns for
all outputs relative to CLK1
set_output_delay 1.2 -clock [get_clocks CLK1]
[get_ports OUT1]
set_output_delay 1.2 -clock [get_clocks CLK1]
[all_outputs]
```

Example 2

```
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

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

set_hierarchy_separator

Commands to set or get hierarchy separator.

Syntax

```
set_hierarchy_separator [<separator>]
```

Arguments

- **separator**: This option specifies the separator which is going to be set as hierarchy separator (Optional).

  If no separator is specified, current separator is print out.

Example 1

```
The following example sets '/'(forward slash)
as hierarchy separator:
     set_hierarchy_separator /
```

Example 2

```
The following example gets the current hierarchy separator:
     set_hierarchy_separator
```

See Also

- "Radiant Software Timing Constraints Tcl Commands" on page 1580

Radiant Software Physical Constraints Tcl Commands

The **ldc** commands are used to set physical constraints.

Radiant Software Physical Constraints Tcl Command Descriptions

The following table provides a listing of all valid Radiant software physical constraints-related Tcl command options and describes option functionality.

Table 22: Radiant Software Physical Constraints Tcl Commands

Command	Description
ldc_create_constraint_set	Create a new constraint set.
ldc_create_group	Define a single identifier that refers to a group of objects.
ldc_create_macro	Define a macro.
ldc_create_region	Define a rectangular area.
ldc_create_vref	Define a voltage reference.
ldc_current_constraint_set	Switch the constraint set by the name.
ldc_define_attribute	Define LSE attributes.
ldc_define_global_attribute	Define LSE global attributes.
ldc_delete_constraint	Clear constraints by given IDs.
ldc_delete_constraint_set	Delete the constraint set by the name.
ldc_get_groups	Return group objects.
ldc_list_constraint	List constraints from database.
ldc_list_constraint_sets	List all existing constraint sets.
ldc_prohibit	Prohibit site or region.
ldc_read	Read constraint file.
ldc_set_attribute	Set object attributes.
ldc_set_location	Set object location.
ldc_set_port	Set port constraint attributes.
ldc_set_sysconfig	Set sysconfig attributes.

Table 22: Radiant Software Physical Constraints Tcl Commands

Command	Description
ldc_set_vcc	Set the voltage and/or derate for the bank, core or dphy.
ldc_write	Write constraint file.

ldc_create_constraint_set

Commands to create a new constraint set.

Syntax

```
ldc_create_constraint_set [-name_style logic ]
[-copy_from <copy_from_name>] <name>
```

Arguments

- **-name_style logic**: This value indicates that name style of new constraint set is logic. When this option is set, only constraint file with logical names can be loaded in this constraint set (optional).
- **-copy_from copy_from_name:** This value specifies an existing constraint set from which the new set will copy constraints (optional).
- **name:** This name string specifies the name of the new constraint set.

Example

```
The following example creates a new constraint set 'test'
with name style 'logic':
ldc_create_constraint_set -name_style logic test
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_create_group

Commands to define a single identifier that refers to a group of objects.

Syntax

```
ldc_create_group -name <group_name>
[-bbox {height  width}] <objects>
```

Arguments

- **-name group_name:** This value specifies the group name.
- **-bbox {height width}:** This value specifies the maximum number of rows (R) and columns (C), of the group’s bounding box. This option is not applicable to port (optional).

- **object:** Objects named in the group, either port, instance, pin or net.

Example

```
ldc_create_group -name group1 [get_ports {a*}]
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_create_macro

Commands to define an instance in a design as a macro.

Syntax

```
ldc_create_macro [-name <macro_name>]
[-use_pio <ports_list>] <instance>
```

Arguments

- **-name macro_name:** This value specifies the macro name (optional).
- **-use_pio ports_list:** This value specifies macro ports that are directly connected outside of the design (optional).
- **instance:** The instance which is defined as a macro.

Example

```
ldc_create_macro -name macro_top -use_pio {din*}
[get_cells u_top]
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_create_region

Commands to define a rectangular area.

Syntax

```
ldc_create_region -name <region_name> (-anchor <anchor>|
      -site <site>) -width <width> -height <height>
      [-buffer_left <left>] [-buffer_right <right>]
      [-buffer_top <top>] [-buffer_bottom <bottom>]
      [-exclusive] [-routing_exclusive]
```

Arguments

- **-name region_name:** This value specifies the region name.
- **-anchor anchor:** This value specifies anchor.
- **-site site:** This value specifies the site name. Option '-anchor' and '-site' are mutually exclusive.
- **-width width:** This value specifies the width of the placement region in columns.
- **-height height:** This value specifies the height of the placement region in rows.
- **–buffer_left left:** This value specifies the left buffer zone of the routing region (optional).
- **-buffer_right right:** This value specifies the right buffer zone of the routing region (optional).
- **–buffer_top top:** This value specifies the top buffer zone of the routing region (optional).
- **–buffer_bottom bottom:** This value specifies the bottom buffer zone of the routing region. Options '–buffer_left','-buffer_right', '-buffer_top' and '-buffer_bottom' are only for creating macro region. Without these options routing region is identical to placement region (optional).
- **-exclusive:** This option indicates that the current buffer cannot be used to place and route other logic during the Reuse stage (optional).
- **-routing_exclusive:** This option indicates that the current buffer cannot be used to route other logic during the Reuse stage. Options '–exclusive' and '-routing_exclusive' are only for creating macro region (optional).

Example

```
ldc_create_region -name top_region -site R17C2D
-width 23 -height 39
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_create_vref

Commands to define a voltage reference.

Syntax

```
ldc_create_vref -name <vref_name> -site <site_name>
```

Arguments

- **-name vref_name:** This value specifies voltage reference name.
- **-site site_name:** This value specifies PIO site of the target device.

Example

```
ldc_create_vref -name VREF1_BANK_3 -site N21
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_current_constraint_set

Commands to switch the constraint set by the name.

Syntax

```
ldc_current_constraint_set [<name>]
```

Arguments

- **name:** This option specifies the constraint set which is going to be active. If no constraint set is specified, current active constraint set is print out (optional).

Example 1

```
The following example switches the constraint set 'test'
to be active:
ldc_current_constraint_set test
```

Example 2

```
The following example gets the active constraint set:
ldc_current_constraint_set
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_define_attribute

Commands to define LSE attribute.

Syntax

```
ldc_define_attribute -attr <attr_name> -value
<attr_value> -object_type <type> -object <objects>
[-disable] [-comment <comment>]
```

Arguments

- **-attr attr_name:** This value indicates LSE attributes name, such as 'loc'.

- **-value attr_value:** This value indicates LSE attribute value such as '2'.
- **-object_type type:** This value indicates the type of object to which the attribute belongs, such as 'port'.
- **-object objects:** This value indicates the specific object to which the attribute applies, such as 'clk'.
- **-disable:** This option indicates that the command is disabled (optional).
- **-comment:** This option specifies comment of the command (optional).

Example

```
ldc_define_attribute -attr loc -value 2 -object_type
port -object {clk}
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_define_global_attribute

Commands to define global LSE attribute

Syntax

```
ldc_define_global_attribute -attr <attr_name> -value
    <attr_value> [-disable] [-comment <comment>]
```

Arguments

- **-attr attr_name:** This value indicates LSE attributes name, such as 'syn_ramstyle'.
- **-value attr_value:** This value indicates LSE attribute value such as 'registers'.
- **-disable:** This option indicates that the command is disabled (optional).
- **-comment:** This option specifies comment of the command (optional).

Example

```
ldc_define_global_attribute -attr syn_ramstyle
-value registers
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_delete_constraint

Commands to clear constraints by given IDs.

Syntax

```
ldc_delete_constraint [-all] [<id>] [<id>...]
```

Arguments

- **-all**: This option indicates that the command will delete all constraints (optional).
- **-id**: This value specifies the id of constraint which is going to be deleted (optional).

Example

The following example deletes constraint with index 3:

```
ldc_delete_constraint 3
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_delete_constraint_set

Commands to delete the constraint set by the name.

Syntax

```
ldc_delete_constraint_set <name>
```

Arguments

- **name:** This value specifies the name of the constraint set which is going to be deleted. Active constraint set can't be deleted.

Example

```
The following example deletes constraint set 'test':
ldc_delete_constraint_set test
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_get_groups

Commands to return group objects.

Syntax

```
ldc_get_groups [-regexp] [-nocase] [<patterns>]
```

Arguments

- **-regexp:** This option indicates that patterns are full regular expressions (optional).
- **-nocase:** This option indicates that matching is case-insensitive (optional).
- **patterns:** This value specifies the patterns against which group names are matched. If no pattern is specified, set '*' as default (optional).

Example

```
ldc_get_groups -nocase -regexp group\[0-9]+
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_list_constraint

Commands to list constraints from database.

Syntax

```
ldc_list_constraint [-all|-a]
```

Arguments

- **-all|-a**: This option indicates that the command will get all the constraints including disabled or incorrect constraints (optional).

Example

The following example lists enabled and correct constraints from database:

```
ldc_list_constraint
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_list_constraint_sets

Commands to list all existing constraint sets.

Syntax

```
ldc_list_constraint_sets
```

Example

The following example lists all existing constraint sets in the current view:

```
ldc_list_constraint_sets
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_prohibit

Commands to prohibit the use of a site or all sites in a region.

Syntax

```
ldc_prohibit (-site <site> | -region <region>)
```

Arguments

- **-site site:** This value specifies site name.
- **-region region:** This value specifies region name. Option '-site' and '-region' are mutually exclusive.

Examples

```
ldc_prohibit -site AB
ldc_prohibit -region regionA
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_read

Commands to read constraint file.

Syntax

```
ldc_read <file>
```

Arguments

- **file:** This value specifies the constraint file which is going to be read.

Example

```
The following example reads constraint file test.sdc:
ldc_read test.sdc
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_set_attribute

Commands to set constraint attributes to the objects.

Syntax

```
ldc_set_attribute <key-value list> [objects]
```

Arguments

- **key-value list:** This value is a list of constraint attributes.
- **objects:** This value specifies object(s), to which constraint attributes are set. If no object is specified, attributes are set to design (optional).

Example

```
ldc_set_attribute GSR_NET=TRUE [get_nets {my_gsr}]
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_set_location

Commands to set object location.

Syntax

```
ldc_set_location [-site <site_name>] [-bank <bank_num>]
    [-region <region_name>] <object>
```

Arguments

- **-site site_name:** This value specifies name of the site to place object (optional).
- **-bank bank_num:** This value specifies number of the bank to place object (optional).
- **-region region_name:** This value specifies name of the user defined region to place object. One option among '-site' ,'-bank' and '-region' is needed and they are mutually exclusive (optional).
- **object:** Object to be located.

Examples

```
ldc_set_location -site U6 [get_ports clk]
ldc_set_location -region region0 [ldc_get_groups {group1}]
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_set_port

Commands to set port constraint attributes.

Syntax

```
ldc_set_port (-iobuf [-vref <vref_name>]| -sso)
<key-value list> [ports]
```

Arguments

- **-iobuf:** This option indicates that IOBUF attributes are set.
- **-vref vref_name:** This value specifies voltage reference name. It must be bound to -iobuf (optional).
- **-sso:** This option indicates that SSO attributes are set. Option '-iobuf' and '-sso' are mutually exclusive.
- **key-value list:** This value is a list of attributes. Attributes must be enclosed with curly braces.
- **ports:** This value specifies ports to be set. If no port is specified, set constraint to all ports (optional).

Example

```
ldc_create_vref -name VERF2 -site R11
ldc_set_port -iobuf -vref VERF2 {IO_TYPE=HSTL15_I
VREF=VREF2_LOAD} [get_ports clk_third]
```

See Also

- “Radiant Software Physical Constraints Tcl Commands” on page 1594

ldc_set_sysconfig

Commands to set sysconfig attributes.

Syntax

```
ldc_set_sysconfig <key-value list>
```

Arguments

- **key-value list:** This value is a list of SYSCONFIG attributes. Attributes must be enclosed with curly braces.

Example

```
ldc_set_sysconfig {JTAG_PORT=ENABLE PROGRAMN_PORT=ENABLE
    MCCLK_FREQ=56.2 DONE_OD=ON}
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_set_vcc

Commands to set the voltage and/or derate for the bank, core or dphy.

Syntax

```
ldc_set_vcc (-bank <bank_number>|-core|-dphy DPHY0/DPHY1)
    [-derate <derate>] [<voltage>]
```

Arguments

- **-bank bank_number:** This value specifies the bank to be set with voltage or voltage derating percent.
- **-core:** This option indicates that voltage or voltage derating percent is set to core.
- **-dphy DPHY0/DPHY1:** This option indicates that voltage or voltage derating percent is set to dphy. Value of option '-dphy' can only be 'DPHY0' or 'DPHY1'. Option '-bank', '-core' and '-dphy' are mutually exclusive.
- **-derate:** This value specifies voltage derating percent (optional).
- **voltage:** This value can be any compatible voltage supply to that bank or the core or dphy; for example, 2.5, 3.3. At least one option between '-derate' and 'voltage' is needed (optional).

Examples

```
ldc_set_vcc -bank 1 3.3
ldc_set_vcc -dphy DPHY0 -derate -3
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

ldc_write

Commands to write constraint file.

Syntax

```
ldc_write [-overwrite|w] [-sdc] [-exclude_sdc]
[-include_error] <file>
```

Arguments

- **-overwrite|w:** This option indicates that the command will overwrite file of the same name (optional).
- **-sdc:** This option indicates that only SDC timing constraints and Synthesis attributes will be written to file (optional).

- **-exclude_sdc:** This option indicates that SDC timing constraints and Synthesis attributes won't be written to file. Option '-sdc' and '-exclude_sdc' are mutually exclusive (optional).
- **-include_error:** This option indicates that incorrect constraints will also be written to file (optional).
- **file:** This value specifies the constraint file which is going to be written.

Example

```
ldc_write -sdc -include_error test.sdc
```

See Also

- "Radiant Software Physical Constraints Tcl Commands" on page 1594

Synthesis Tcl Command

The **syn** command is used to run the synthesis process. You can change from project flow to a non-project flow by using the **syn_run** command.

Radiant Software Synthesis Tcl Command Description

The following table provides a listing of all valid Radiant software synthesis-related Tcl command options and describes option functionality.

Table 23: Synthesis Tcl Command

Command	Description
syn_run	Run synthesis tool.

syn_run

Run synthesis tool.

Syntax

```
syn_run
```

Arguments

Example

```
syn_run
```

See Also

- "Synthesis Tcl Command" on page 1606

Design Object Tcl Commands

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.

The **get_info** commands are used for constraints to return a single object, or a collection of objects. You can get this command by using **help {get_*}**.

Radiant Software Design Object Tcl Command Descriptions

The following table provides a listing of all valid Radiant software design object-related Tcl command options and describes option functionality.

In the example below, <*target name*> is a regular expression that matches one object only.

Clock Object

SDC Collection	Description
get_clocks	Return clock objects.
all_clocks	Return a list of all clocks in the current design.

This section displays the usage information of the commands:

get_clocks

```
% help get_clocks
get_clocks -  Commands to get clocks
  Usage:
    get_clocks [-regexp] [-nocase] <patterns>
```

all_clocks

```
% help all_clocks
all_clocks -  Commands to get a list of all clocks in the
current design
  Usage:
    all_clocks
```

Radiant Software Clock Object Tcl Commands Example

This section illustrates and describes a few samples of Radiant clock object Tcl commands.

Example 1

<*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.

```
get_clocks <target name>
[get_clocks clock_fast]

[all_clocks]
```

Port Object

SDC Collection	Description
get_port_info	Return port object information.
get_ports	Return port objects.
all_inputs	Return a list of all input ports in the current design.
all_outputs	Return a list of all output ports in the current design.

get_port_info

Commands to return port info.

Syntax

```
get_port_info [-name] [-is_inout] [-is_input] [-is_output]
<port_object>
```

Arguments

- **-name**: This option indicates that the command will output the name of the specified port (optional).
- **-is_inout**: This option indicates that the command will output if the specified port is an inout port (optional).
- **-is_input**: This option indicates that the command will output if the specified port is an input port (optional).
- **-is_output**: This option indicates that the command will output if the specified port is an output port (optional).
- **port_object**: This value specifies one port object whose information will be output.

Example:

The following example outputs whether port '-seg_out[0]' is an output port:

```
get_port_info -is_output [get_ports {seg_out[0]}]
```

See Also

- "Port Object" on page 1608

get_ports

Commands to get a list of ports in the current design.

Syntax

```
get_ports [-regexp] [-nocase] [<patterns>]
```

Arguments

- **-regexp:** This option indicates that patterns are full regular expressions (Optional).
- **-nocase:** This option indicates that matching is case-insensitive (Optional).
- **patterns:** This value specifies the patterns against which net names are matched (Optional). If no pattern is specified, set '*' as default.

Example

```
get_ports -nocase -regexp clk\[0-9]+
```

See Also

- "Port Object" on page 1608

all_inputs

Commands to get a list of all input ports in the current design.

Syntax

```
all_inputs
```

Example

```
The following example returns all input ports in the current
design:
all_inputs
```

See Also

- "Port Object" on page 1608

all_outputs

Commands to get a list of all output ports in the current design.

Syntax

```
all_outputs
```

Example

```
The following example returns all output ports in the current
design:
all_outputs
```

See Also

- "Port Object" on page 1608

Cell Object

SDC Collection	Description
get_cell_info	Return cell information.
get_cells	Return instance objects.

This section displays the usage information of the commands:

get_cell_info

```
% help get_cell_info
get_cell_info - Commands to return cell info
  Usage:
    get_cell_info [-name] [-inpins] [-outpins] [-pins]
<cell_object>
```

get_cells

```
% help get_cells
get_cells -  Commands to get instances
  Usage:
    get_cells [-hierarchical] [-regexp] [-nocase] [-of_objects
<objects>] <patterns>
```

Radiant Software Cell Object Tcl Commands Example

This section illustrates and describes a few samples of Radiant cell object Tcl commands.

Example 1

The command returns collection of a design instance that matches *<target name>*.

```
[get_cells <target name>
```

```
[get_cells {reg6 reg7}]
```

Net Object

SDC Collection	Description
get_net_info	Return net information.
get_nets	Return net objects.

get_net_info

Commands to return net info.

Syntax

```
get_net_info [-name] [-pin] <net_object>
```

Arguments

- **-name**: This option indicates that the command will output the name of the specified net (optional).
- **-pin**: This option indicates that the command will list pin names of specified net (optional).
- **-net_object**: This value specifies one net object whose information will be listed.

Example:

The following example lists pin names of net 'CLK':

```
get_net_info -pin [get_nets CLK]
```

See Also

- “Net Object” on page 1611

get_nets

Commands to get a list of nets in the current design.

Syntax

```
get_nets [-hierarchical] [-regexp] [-nocase] [-of_objects
<objects>] [<patterns>]
```

Arguments

- **-hierarchical:** This option indicates that the command will get nets from all levels of the design hierarchy (Optional). Without this argument, the command will only get nets from the top of the design hierarchy.
- **-regexp:** This option indicates that patterns are full regular expressions (Optional).
- **-nocase:** This option indicates that matching is case-insensitive (Optional) .
- **-of_objects objects:** This option indicates that the command will get nets of these objects(PINs or INSTANCEs) (Optional). When this argument is effective, other arguments won't work.
- **patterns:** This value specifies the patterns against which net names are matched (Optional)**.** If no pattern is specified, set '*' as default.

Example

```
get_nets -hierarchical -nocase -regexp clk\[0-9]+
get_nets -of_objects [get_pins {i*/*}]
```

See Also

- “Net Object” on page 1611

Pin Object

SDC Collection	Description
get_pin_info	Return pin objects.
get_pins	Return pin information.

get_pin_info

Commands to return pin info.

Syntax

```
get_pin_info [-name] [-is_clock] [-is_input] [-is_output]
[-net] [-parent_cell] <pin_object>
```

Arguments

- **-name**: This option indicates that the command will output the name of the specified pin (optional).
- **-is_clock**: This option indicates that the command will output if the specified pin is a clock pin (optional).
- **-is_input**: This option indicates that the command will output if the specified pin is an input pin (optional).

- **-is_output**: This option indicates that the command will output if the specified pin is an output pin (optional).
- **-net**: This option indicates that the command will output the net name of the specified pin (optional).
- **-parent_cell**: This option indicates that the command will output the instance name of the specified pin (optional).
- **pin_object**: This value specifies one pin object whose information will be output.

Example:

The following example outputs instance name of pin 'fifo_inst/state_0':

```
get_pin_info -parent_cell [get_pins fifo_inst/state_0]
```

See Also

- "Pin Object" on page 1612

get_pins

Commands to get a list of pins in the current design.

Syntax

```
get_pins [-hierarchical] [-regexp] [-nocase]
[-of_objects <objects>] [<patterns>]
```

Arguments

- **-hierarchical:** This option indicates that the command will get pins from all levels of the design hierarchy. Without this argument, the command will only get pins from the top of the design hierarchy (optional).
- **-regexp:** This option indicates that patterns are full regular expressions (optional).
- **-nocase:** This option indicates that matching is case-insensitive (optional).
- **-of_objects objects:** This option indicates that the command will get pins of these objects (NETs or INSTANCEs). When this argument is effective, other arguments won't work (optional).
- **patterns:** This value specifies the patterns against which net names are matched. If no pattern is specified, set '*' as default (optional).

Example

```
get_pins -hierarchical -nocase -regexp clk\[0-9]+
get_pins -of_objects [get_nets {clk*}]
```

See Also

- "Pin Object" on page 1612

Wildcard Support

Multiple wildcard (*) is supported. In addition, 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.

Note

1. Wildcard * can be used in any position of object name such as a*, *b, a*b etc.
2. Wildcard * cannot cross hierarchical separator /. For example get_cells sub1/* only returns sub1/sub2 and does not return sub1/sub2/sub1 or sub1/sub2/sub1/sub2.
3. When -hierarchical is set, it will search all levels of the design hierarchy starting at the current instance.

System Tcl Commands

The **sys** commands are used to show Radiant tool and system settings information. You can get these commands by using **help {sys_*}**.

Radiant Software System Tcl Command Descriptions

The following table provides a listing of all valid Radiant software system-related Tcl command options and describes option functionality.

Table 24: System Tcl Commands

Command	Description
sys_install_path	Return Radiant executable path.
sys_install_version	Return Radiant version.
sys_list_attribute	List system attributes.
ldc_set_attribute	Set system attributes.

sys_install_path

Return Radiant executable path

Syntax

```
sys_install_path
```

Arguments

Example

```
sys_install_path
```

See Also

```
sys_install_version
```

- "System Tcl Commands" on page 1614

sys_install_version

Return Radiant version

Syntax

```
sys_install_version
```

Arguments

Example

```
sys_install_version
```

See Also

```
sys_install_path
```

- "System Tcl Commands" on page 1614

sys_list_attribute

List system attributes.

Syntax

```
sys_list_attribute
```

Arguments

Example

```
sys_list_attribute
```

See Also

```
sys_set_attribute
```

- "System Tcl Commands" on page 1614

sys_set_attribute

Set system attributes.

Syntax

```
sys_set_attribute [-echo <value>] [-reg <value>]
[-gui <value>] [-msg <value>]
```

Arguments

- **-echo <value>:** value must be: {on,off} Allows you to enable or disable the command summary line. (Optional)
- **-reg <value>:** value must be: {on,off}. Allows you to output RAT_REG_RECORD_ON or RAT_REG_RECORD_OFF. (Optional)
- **-gui <value>:** value must be: {on,off}. The run is called from gui if its value is on. (Optional)
- **-msg <filename>:** Set message setting file to <filename>. (Optional)

Example

```
sys_set_attribute  -echo on -reg off
```

See Also

```
sys_list_attribute
```

- "System Tcl Commands" on page 1614

Radiant Software Project Tcl Commands

The Radiant software 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.

Radiant Software Project Tcl Command Descriptions

The following table provides a listing of all valid Radiant software project-related Tcl command options and describes option functionality.

Table 25: Radiant Software Project Tcl Commands

Command	Description
prj_activate_impl	Activate implementation in the current project.
prj_add_source	Add sources to the current project.

Table 25: Radiant Software Project Tcl Commands

Command	Description
prj_archive	Archive the current project.
prj_clean_impl	Clean up the implementation result files in the current project.
prj_clone_impl	Clone an existing implementation.
prj_close	Close the current project. Any unsaved changes are discarded.
prj_copy_strategy	Create a new strategy by copying from an existing strategy.
prj_create	Create a project.
prj_create_impl	Create a new implementation in the current project.
prj_create_strategy	Create a new strategy with default settings.
prj_device	List the device.
prj_disable_source	Disable the design sources from the current project.
prj_enable_source	Enable the design sources from the current project.
prj_get_impl_path	Return active or given implementation folder path.
prj_get_strategy_value	Get value of a strategy item.
prj_import_strategy	Import an existing strategy file.
prj_list_source	List source files in current project.
prj_list_strategy	List value of strategy items.
prj_open	Open a project file.
prj_open_milestone	Open project milestone netlist files.
prj_remove_impl	Delete the specified implementation from the current project.
prj_remove_source	Delete design source from the current project.
prj_remove_strategy	Delete an existing strategy.
prj_run_bitstream	Run bitstream process.
prj_run_map	Run map process.
prj_run_par	Run place & route process.
prj_run_synthesis	Run synthesis process.
prj_save	Save current project.
prj_saveas	Save the current project as a new project with specified name and directory.

Table 25: Radiant Software Project Tcl Commands

Command	Description
prj_set_device	Set the device.
prj_set_impl_opt	List, set or remove implementation options in the current project.
prj_set_opt	List, set or remove a project option.
prj_set_postscript	List or set user Tcl script after running milestone.
prj_set_prescript	List or set user Tcl script before running milestone.
des_set_reference_udb	Set reference UDB for incremental run or disable it.
prj_set_source_format	Set a source format.
prj_set_source_opt	List, set or remove a source option.
prj_set_strategy	Associate the strategy with the specified implementation.
prj_set_strategy_value	Set value to a strategy item.
prj_set_top_module	Set top module in the current implementation.

This section displays the usage information of the commands:

prj_activate_impl

Activate implementation in the current project.

Syntax

```
prj_activate_impl <impl name>
```

Arguments

- **<impl name>**: Specify implementation name.

Example:

```
prj_activate_impl "myimpl"
```

See also

- "Radiant Software Project Tcl Commands" on page 1616

prj_add_source

Add sources to the current project

Syntax

```
prj_add_source [-impl <implement name>]
[-simulate_only|-synthesis_only][-work <VHDL lib name>]
[[-opt <name=value>] ...] [-exclude] <src file>...
```

Arguments

- **-impl <implement name>**: Specify the implementation name. If not specified, use the default implementation (optional).
- **-simulate_only|-synthesis_only**: Indicate new added source can be simulation or synthesis only (optional).
- **-work <VHDL lib name>**: Indicate VHDL library name. Default library name is "work" (optional).
- **-opt <name=value>**: Specify one or multiple option value pair of source file (optional).
- **-exclude**: Indicate the source to be excluded from the project (optional).
- **<src file>**: Specify one or multiple source file paths.

Example

- prj_add_source -synthesis_only "D:/my_design/counter/source/count_attr.v"
- prj_add_source -work mywork D:/my_design/counter/source/my.vhd

See also

- "Radiant Software Project Tcl Commands" on page 1616

prj_archive

Archive the current project or extract the archive file.

Syntax

```
prj_archive [-includeAll] <archive_file>
: Archive the current project into the archive_file
prj_archive -extract -dir <destination directory>
<archive_file>
: Extract the archive file and load the project
```

Arguments

- **-includeAll:** Indicate to include all files under project folder into archive file (optional).
- **<archive_file>:** Specify the archive file (.zip) path.
- **-extract:** Indicate to extract the specified archive file.
- **-dir <destination directory>:** Specify the destination directory to extract archive file. This directory should exist in disk.

Example

```
prj_archive -includeAll "D:/my_desgin/counter/counter.zip"
prj_archive -dir D:/my_design/newcounter -extract
D:/my_design/counter/counter.zip
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_clean_impl

Clean up the implementation result files in the current project.

Syntax

```
des_list_net [-clock] [-fanout <n>] [-lsort <method>]
[-lsize <n>] [<nets>]
```

Arguments

- **-clock**: Lists clock nets only (optional).
- **-fanout <n>**: Lists nets with # of fanouts >= n (optional).
- **-lsort <method>**: Choose sorting method: 0 - id; 1 - all pins; 2 - clk pins; 3 - global control pins; 4 - normal pins (optional). Defaults to sorting by id.
- **-lsize <size>**: Limits the number of listed nets to size. Defaults to showing all nets (optional).
- **<nets>**: Only lists net names that match the pattern specified in <nets> (optional).

Example:

```
des_list_net -fanout 10 -lsort 2 -lsize 10 *D0*
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_clone_impl

Clone an existing implementation.

Syntax

```
prj_clone_impl <new impl name> [-dir <new impl directory>]
[-copyRef] [-impl <original impl name>]
```

Arguments

- **-name <new impl name>:** Specify the new implementation name.

- **-dir <new impl directory>:** Specify the new implementation directory (optional).
- **-copyRef:** Indicate to copy all source files from original to new implementation (optional).
- **-impl <original impl name>:** Specify the original implementation name. If not specified, use the default implementation (optional).

Example:

```
prj_clone_impl newimpl -copyRef -impl myimpl
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_close

Close the current open project.

Syntax

```
prj_close
```

Arguments

Example:

```
prj_close
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_copy_strategy

Create a new strategy by copying from an existing strategy.

Syntax

```
prj_copy_strategy -from <source strategy name> -name
<new strategy name> -file <strategy file name>
```

Arguments

- **-from <source strategy name>:** Specify the source strategy name.
- **-name <new strategy name>:** Specify new strategy name.
- **-file <strategy file name>:** Specify strategy file (.sty) path. This file is used to save new strategy setting.

Example:

```
prj_copy_strategy -from Strategy1 -name mystrategy -file
d:/my_desgin/counter/mystrategy.sty
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_create

Create a new project.

Syntax

```
prj_create -name <project name> [-dir <project directory>]
[-dev <device name>] [-performance <performance grade>]
[-impl <initial implementation name>] [-impl_dir <initial
implementation directory>][-synthesis <synthesis tool name>]
```

Arguments

- **-name <project name>:** Specify the project name.
- **-dir <project directory>:** Specify the project directory (optional).
- **-dev <device name>**: Specify the device part name.
- **-performance <performance grade>:** Specify the device performance grade.
- **-impl <initial implementation name>:** Initial implement name, default name is 'impl_1' (optional).
- **-impl_dir <initial implementation directory>:** Implement directory name, default directory name is same as name implementation name (optional).
- **-synthesis <synthesis tool name>:** Synthesis tool type. The valid synthesis name is "synplify" and "lse".

Example:

```
prj_create -name "counter" -impl "impl_1" -dev LIFCL-17-9BG256C
-performance "9_High-Performance_1.0V" -synthesis "synplify"

prj_create -name myproj -dev LIFCL-17-9BG256C -performance
"9_High-Performance_1.0V" -impl myimpl -impl_dir impldir -
synthesis lse
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_create_impl

Create a new implementation in the current project.

Syntax

```
prj_create_impl <new impl name> [-dir <implementation
directory>][-strategy <default strategy name>]
[-synthesis <synthesis tool name>]
```

Arguments

- **-name <new impl name>:** Specify the implementation name.
- **-dir <implementation directory>:** Specify the implementation directory. It's a subdirectory name under project path.
- **-strategy <default strategy name>:** Specify the strategy name (optional).
- **-synthesis <synthesis tool name>:** Synthesis tool type. The valid synthesis name is “synplify” and “lse”.

Example:

```
prj_create_impl myimpl -strategy Strategy1
prj_create_impl myimpl -dir myimpldir -synthesis "synplify"
```

See Also

- “Radiant Software Project Tcl Commands” on page 1616

prj_create_strategy

Create a new strategy with default setting

Syntax

```
prj_create_strategy -name <new strategy name> -file
<strategy file name>
```

Arguments

- **-name <new strategy name>:** Specify new strategy name.
- **-file <strategy file name>:** Specify strategy file (.sty) path. This file is used to save new strategy setting.

Example:

```
prj_create_strategy -name mystrategy -file
d:/my_desgin/counter/mystrategy.sty
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_device

List the device information of the current project

Syntax

```
prj_device [-family|-device|-package|-performance|
-operation|-part]
```

Arguments

- **-family:** Return the family name (optional).
- **-device:** Return the device name (optional).
- **-package:** Return the package name (optional).
- **-performance:** Return the performance grade (optional).
- **-operation:** Return the device operation (optional).
- **-part:** Return the device part.

Example:

```
prj_device -family -device
prj_device -part
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_disable_source

Disable the design sources from the current project.

Syntax

```
prj_disable_source [-impl <implement name>] <src file> ...
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **<src file>:** Specify one or multiple source files to be disabled.

Example:

```
prj_disable_source "D:/my_design/counter/source/count_attr.v"
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_enable_source

Enable the design sources from the current project.

Syntax

```
prj_enable_source [-impl <implement name>] <src file> ...
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **<src file>:** Specify one or multiple source files to be enabled.

Example:

```
prj_enable_source "D:/my_design/counter/source/count_attr.v"
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_get_impl_path

Return active or given implementation folder path

Syntax

```
prj_get_impl_path [-impl <implement>]
```

Arguments

- **-impl <implement name>:** Specify implementation name. If not specified, use the default implementation.

Example:

```
prj_get_impl_path
prj_get_impl_path -impl myimpl
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_get_strategy_value

Get the value of a strategy item.

Syntax

```
prj_get_strategy_value [-strategy <strategy name>]
<option name>
```

Arguments

- **-strategy <strategy name>:** Specify the strategy name. If not specified, use the default strategy (optional).
- **<option name>:** Specify the strategy option name.

Example:

```
prj_get_strategy_value -strategy Strategy1 map_ignore_sdc_error
prj_get_strategy_value par_disable_autohldtiming
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_import_strategy

Import an existing strategy file.

Syntax

```
prj_import_strategy -name <new strategy name> -file
<strategy file name>
```

Arguments

- **-name <new strategy name>:** Specify new strategy name.
- **-file <strategy file name>:** Specify import strategy (.sty) path.

Example:

```
prj_import_strategy -name mystrategy -file d:/my_desgin/
counter/strategy1.sty
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_list_source

List source files in current project.

Syntax

```
prj_list_source [-o <output_file>]
```

Arguments

- **-o <output_file>:** Specify the output file name. If not specified, list source files in stdout (optional).

Example:

```
prj_list_source
prj_list_source -o D:/my_design/counter/source.txt
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_list_strategy

List value of strategy items

Syntax

```
prj_list_strategy [-strategy <strategy name>] <pattern>
```

Arguments

- **-strategy <strategy name>:** Specify the strategy name. If not specified, use the default strategy (optional).
- **<pattern>:** Specify the strategy option name. Wildcard is supported.

Example:

```
prj_list_strategy -strategy Strategy1 map_*
prj_list_strategy par_*
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_open

Open a project file.

Syntax

```
prj_open <project file>
```

Arguments

- **<project name>:** Specify project file (.rdf) path.

Example:

```
prj_open "d:/my_desgin/counter/counter.rdf"
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_open_milestone

Commands to open project milestone netlist files.

Syntax

```
prj_open_milestone -postsyn | -postmap | -postpar
```

Arguments

- **-postsyn:** Open post-syn logical netlist (optional).
- **-postmap:** Open post-map physical netlist (optional).
- **-postpar:** Open post-par physical netlist.

Option '-postsyn' '-postmap' and '-postpar' are mutually exclusive.

Example:

```
prj_open_milestone -postsyn
prj_open_milestone -postmap
prj_open_milestone -postpar
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_remove_impl

Delete the specified implementation from the current project.

Syntax

```
prj_remove_impl <impl name>
```

Arguments

- **<impl name>:** Specify the implementation name.

Example:

```
prj_remove_impl "myimpl"
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_remove_source

Delete design source from the current project.

Syntax

```
prj_remove_source [-impl <implement name>]
-all: Remove all the design sources in project
prj_remove_source [-impl <implement name>] <src file> ...
: Remove the specified sources from the current project
```

Arguments

- **impl <implement name>:** Specify the implementation name. If not specified, use the default implementation.
- **-all:** Indicate to remove all sources from the current implementation.
- **<src file>**: Specify one or multiple source files to be removed. This option is mutually exclusive with "-all".

Example:

```
prj_remove_source -all
prj_remove_source "D:/my_design/counter/source/count_attr.v"
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_remove_strategy

Delete an existing strategy

Syntax

```
prj_remove_strategy <strategy name>
```

Arguments

- **<strategy name>:** Specify the strategy name to be removed.

Example:

```
prj_remove_strategy mystrategy
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_run_bitstream

Run bitstream process.

Syntax

```
prj_run_par [-forceAll|forceOne]
```

Arguments

- **-forceAll:** Indicate to force to run all tasks (optional).
- **-forceOne**: Indicate to force to run current task (optional).

Example:

```
prj_run_bitstream
prj_run_bitstream -forceAll
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_run_map

Run map process.

Syntax

```
prj_run_map [-forceAll|forceOne]
```

Arguments

- **-forceAll:** Indicate to force to run all tasks (optional).
- **-forceOne**: Indicate to force to run current task (optional).

Example:

```
prj_run_map
prj_run_map -forceAll
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_run_par

Run place and route process.

Syntax

```
prj_run_par [-forceAll|forceOne]
```

Arguments

- **-forceAll:** Indicate to force to run all tasks (optional).
- **-forceOne**: Indicate to force to run current task (optional).

Example:

```
prj_run_par
prj_run_par -forceAll
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_run_synthesis

Run synthesis process

Syntax

```
prj_run_synthesis [-forceAll|forceOne]
```

Arguments

- **-forceAll:** Indicate to force to run all tasks (optional).
- **-forceOne**: Indicate to force to run current task (optional).

Example:

```
prj_run_synthesis
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_save

Save the current project.

Syntax

```
prj_save [<project file>]
```

Arguments

- **<project file>:** Specify the Radiant project file (.rdf) path to be saved. If not specified, save to current opening project file (optional).

Example:

```
prj_save
prj_save d:/my_desgin/counter/mycount.rdf
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_saveas

Save the current project as a new project with specified name and directory and change current workaround of the new project.

Syntax

```
prj_saveas -name <new project name>
-dir <new project directory> [-copy_gen]
```

Arguments

- **-name <new project name>:** Specify the new Radiant project name.
- **-dir <new project directory>:** Specify the new Radiant project file directory.
- **-copy_gen:** Indicate to copy intermediate database files, report files, and other Radiant-generated files into a new project.

Example:

```
prj_saveas -name newcounter -dir
d:/my_desgin/newcounter -copy_gen
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_device

Set the device of current project

Syntax

```
prj_set_device [-family <family name>] [-device <device name>]
[-package <package name>] [-performance <performance grade>]
[-operation <operation>] [-part <part name>]
```

Arguments

- **-family <family name>:** Specify the family name (optional).
- **-device <device name>:** Specify the device name (optional).
- **-package <package name>:** Specify the package name (optional).
- **-performance <performance grade>:** Specify the performance grade (optional).
- **-operation <operation>:** Specify the device operation (optional).
- **-part:** Specify the device part. If '-part' option is specified, other device options are ignored.

Example:

```
prj_set_device -family LIFCL -device LIFCL-17 -package CABGA256
-performance 8_High-Performance_1.0V  -operation COM
prj_set_device -part LIFCL-17-8BG256C
```

See Also

- “Radiant Software Project Tcl Commands” on page 1616

prj_set_impl_opt

List, set, or remove implementation options in the current project.

Syntax

```
prj_set_impl_opt [-impl <implement name>]:
List all the options in the specified implementation
prj_set_impl_opt [-impl <implement name>] <option name> [option
value list]: List or set the implementation's option value
prj_set_impl_opt [-impl <implement name>] -append <option name>
<option value>: Append a value to the specified option value
prj_set_impl_opt [-impl <implement name>] -rem <option name>...
: Remove the the options in the implementation"
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation.
- **<option value>:** Specify the option value.
- **<option value list>:** Specify one or multiple option values. If not specified, it lists the option value of given option name.

- **-append <option name> <option value>:** Append the specified value to current option value (optional).
- **-rem <option name>:** Specify one or multiple option names to be removed.

Example:

```
# list all the options of the implementation
prj_set_impl_opt -impl impl_1
# set the option myoption value to "value1"
prj_set_impl_opt myoption value1
# list the option myoption value
prj_set_impl_opt myoption
# append "value2" to the option myoption
prj_set_impl_opt -append myoption value2
# remove the option
prj_set_impl_opt -rem myoption
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_opt

List, set or remove a project option

Syntax

```
prj_set_opt: List all the options in the current project
prj_set_opt <option name> [<option value list>]: List or
set the option value
prj_set_opt -append <option name> <option value>: Append
a value to the specified option value
prj_set_opt -rem <option name>: Remove the options of
the current project"
```

Arguments

- **<option name>:** Specify the project option name to be listed, set or removed.
- **<option value>:** Specify the option value.
- **<option value list>:** Specify the one or multiple option values. If not specified, it lists the option value instead (optional).
- **-append <option name> <option value>:** Append the specified value to current option value (optional).
- **-rem <option name>:** Specify one or multiple option names to be removed.

Example:

```
# list all the options in the current project
prj_set_opt
```

```
# set the option myoption value to "value1"
prj_set_opt myoption value1
# list the option myoption value
prj_set_opt myoption
# append "value2" to the option myoption
prj_set_opt -append myoption value2
# remove the option
prj_set_opt -rem myoption
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_postscript

List or set user Tcl script after running the milestone.

Syntax

```
prj_set_postscript [-impl <implement name>]
<milestone name> <script_file>
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **- <milestone name>:** Milestone name should be 'syn', 'map', 'par', 'export'.
- **<script_file>:** User Tcl script file path.

Example:

```
prj_set_postscript map my_map.tcl
prj_set_postscript -impl myimpl par my_par.tcl
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_prescript

List or set user Tcl script before running the milestone.

Syntax

```
prj_set_prescript [-impl <implement name>]
<milestone name> <script_file>
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **- <milestone name>:** Milestone name should be 'syn', 'map', 'par', 'export'.
- **<script_file>:** User Tcl script file path.

Example:

```
prj_set_prescript map my_map.tcl
prj_set_prescript -impl myimpl par my_par.tcl
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_source_format

Set a source format.

Syntax

```
prj_set_source_format -src <source name>
[-impl <implement name>] <format>: Currently,
only Verilog or VHDL is supported"
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **-src <source name>:** Specify the source file path to be set.
- **<format>:** Indicate source file format. Only Verilog or VHDL is supported.

Example:

```
prj_set_source_format -src "D:/my_design/counter/source/
count_attr.v" Verilog
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_source_opt

List, set, or remove a source option.

Syntax

```
prj_set_source_opt -src <source name> [-impl <implement name>]
```

```
: List all the options in the specified source
prj_set_source_opt -src <source name> [-impl <implement name>]
<option name> [option value list]
: List or set the source's option value
prj_set_source_opt -src <source name> [-impl <implement name>]
-append <option name> <option value>
: Append a value to the specified option value
prj_set_source_opt -src <source name> [-impl <implement name>]
-rem <option name>...: Remove the options of the source
```

Arguments

- **-src <source name>:** Specify the source file path to be set.
- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **<option value>:** Specify the option value.
- **<option value list>:** Specify the one or multiple option values. If not specified, it lists the option value instead (optional).
- **-append <option name> <option value>:** Append the specified value to current option value (optional).
- **-rem <option name>:** Specify one or multiple option names to be removed.

Example:

```
# list all the options of the source file
prj_set_source_opt -src "D:/my_design/counter/source/
count_attr.v"
# set the option myoption value to "value1"
prj_set_source_opt  -src "D:/my_design/counter/source/
count_attr.v" myoption value1
# list the option myoption value
prj_set_source_opt -src "D:/my_design/counter/source/
count_attr.v" myoption
# append "value2" to the option myoption
prj_set_source_opt -src "D:/my_design/counter/source/
count_attr.v" -append myoption value2
# remove the option
prj_set_source_opt -src "D:/my_design/counter/source/
count_attr.v" -rem myoption
```

See Also

- “Radiant Software Project Tcl Commands” on page 1616

prj_set_strategy

Associate the strategy with the specified implementation.

Syntax

```
prj_set_strategy [-impl <implementation name>] <strategy name>
```

Arguments

- **-impl <implement name>:** Specify the implementation name. If not specified, use the default implementation (optional).
- **<strategy name>:** Specify strategy name associated with implementation.

Example:

```
prj_set_strategy -impl myimpl mystrategy
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_strategy_value

Set value to a strategy item

Syntax

```
prj_set_strategy_value [-strategy <strategy name>] <option
name=option value> ...
```

Arguments

- **-strategy <strategy name>:** Specify the strategy name. If not specified, use the default strategy (optional).
- **<option name=option value>:** Specify one or multiple strategy option key value pair.

Example:

```
prj_set_strategy_value -strategy Strategy1
map_ignore_sdc_error=True map_infer_gsr=True
prj_set_strategy_value -strategy Strategy1
par_disable_autohldtiming=True par_place_iterator=50
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_top_module

Set top module in the current implementation.

Syntax

```
prj_set_top_module <top module>
```

Arguments

- **<top module>:** Specify new top module name.

Example:

```
prj_set_top_module counter
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

prj_set_reference_udb

Set reference UDB for incremental run or disable it.

Syntax

```
prj_set_reference_udb {-clean | -lastrun | -postsyn
<postsyn udb> -postmap <posmap udb> -postpar <postpar udb>
[-pdc <pdc>]}
```

Arguments

- **-clean:** Disable incremental run. This option is mutually exclusive with others.
- **-last run:** Incremental run based on previous run result. This option is mutually exclusive with others.
- **-postsyn <postsyn udb>:** Specify postsyn reference udb (.udb) file path. This option is mutually exclusive with '-clean' & '-lastrun'.
- **-postmap <postsyn udb>:** Specify postmap reference udb (.udb) file path. This option is mutually exclusive with '-clean' & '-lastrun'.
- **-postpar <postsyn udb>:** Specify postpar reference udb (.udb) file path. This option is mutually exclusive with '-clean' & '-lastrun'.
- **-pdc <pdc>:** Specify constraint file for map. This option is mutually exclusive with '-clean' & '-lastrun' (optional).

Example:

```
prj_set_reference_udb -lastrun
prj_set_reference_udb -postsyn ref_udb/postsy.udb -postmap
ref_udb/postmap.udb -postpar ref_udb/postpar.udb -pdc ref_udb/
user.pdc
prj_set_reference_udb -clean
```

See Also

- "Radiant Software Project Tcl Commands" on page 1616

Device Tcl Commands

The **dev** commands are used re used to access or report device information. You can get these commands by using **help {dev_*}**.

The following table provides a listing of all valid Radiant software device-related Tcl command options and describes option functionality.

Table 26: Device Tcl Commands

Command	Description
dev_list_device	List device.
dev_list_family	List family.
dev_list_operation	List operation.
dev_list_package	List package.
dev_list_performance	List performance grade.

dev_list_device

List device

Syntax

```
dev_list_device -family <family> [-p <pattern>]
```

Arguments

- **-family <family>:** Specify the device family name.
- **-p <pattern>:** Specify the device name. Wildcard is supported. If not specified, return all device names under given family. (Optional)

Example

```
dev_list_device -family LIFCL
dev_list_device -family LIFCL -p *17
```

See Also

- "Device Tcl Commands" on page 1640

dev_list_family

List family

Syntax

```
dev_list_family [-p <pattern>]
```

Arguments

- **-p <pattern>:** Specify the family name. Wild card is supported. If not specified, return all device family names. (Optional)

Example

```
dev_list_family
dev_list_family -p LIF*
```

See Also

- "Device Tcl Commands" on page 1640

dev_list_operation

List operation

Syntax

```
dev_list_operation -family <family> -device <device>
-package <package> -performance <performance>
```

Arguments

- **-family <family>:** Specify the device family name.
- **-device <device>:** Specify the device name.
- **-package <package>:** Specify the package name.
- **-performance <performance>:** Specify the performance grade.

Example

```
dev_list_operation -family LIFCL -device LIFCL-17 -package
CABGA256 -performance 9_High-Performance_1.0V
```

See Also

- "Device Tcl Commands" on page 1640

dev_list_package

List package

Syntax

```
dev_list_package -family <family> -device <device>
[-p <pattern>]
```

Arguments

- **-family <family>:** Specify the device family name.
- **-device <device>:** Specify the device name.
- **-p <pattern>:** Specify the package name. Wild card is supported. If not specified, return all package names under given family & device. (Optional)

Example

```
dev_list_package -family LIFCL -device LIFCL-17
dev_list_package -family LIFCL -device LIFCL-17 -p *256
```

See Also

- "Device Tcl Commands" on page 1640

dev_list_performance

List performance grade

Syntax

```
dev_list_performance -family <family> -device <device>
-package <package>
```

Arguments

- **-family <family>:** Specify the device family name.
- **-device <device>:** Specify the device name.
- **-package <package>:** Specify the package name.

Example

```
dev_list_performance -family LIFCL -device LIFCL-17
-package CABGA256
```

See Also

- "Device Tcl Commands" on page 1640

Simulation Libraries Compilation Tcl Commands

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 Radiant software Simulation Libraries Compilation-related Tcl command options and describes option functionality.

Note

Running cmpl_libs may take a long time and may cause the Radiant software to hang.

- It is recommended to run cmpl_libs using the Radiant Tcl Console (**Start Menu > Lattice Radiant Software > Accessories > TCL Console**).

or,

- Run cmpl_libs.tcl using the command line console. Refer to Running cmpl_libs.tcl from the Command Line.

Table 27: Simulation Libraries Compilation Tcl Command

Command	Description
cmpl_libs	Compile simulation library.
sim_generate_script	Generate a simulation script for use with ModelSim or QuestaSim.

This section displays the usage information of the commands:

cmpl_libs

The **-sim_path** argument specifies the path to the simulation tool executable (binary) folder. This option is mandatory. Currently, only QuestaSim simulators from Mentor, Xcelium simulator from Cadence and Active-HDL simulator from Aldec 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.
- To execute this script error free, Siemens Questa 2020.4 or a later version should be used for compilation.

The -sim_vendor argument specifies simulation vendor. This argument is optional. If you do not use this argument, Mentor is chosen by default, and the libraries are compiled for QuestaSim. If you are using QuestaSim choose mentor, Xcelium choose cadence, Active-HDL choose aldec for -sim_vendor.

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.

Note

- This argument is recommended if you did not change the current folder from the Radiant software startup (binary) folder, or if the current folder is write-protected.
- 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.

```
% cmpl_libs
Usage: cmpl_libs -sim_path <sim_path> [-sim_vendor
{mentor<default>|cadence|aldec}] [-device
{ice40up|lifcl|lfcpnx|lfd2nx|lfmxo5-25|lfmxo5-15d|lfmxo5-
100t|lfmxo5-55t|lavat|ut24c|ut24cp|all<default>}] [-target_path
<target_path>]
```

sim_generate_script

The sim_generate_script command is used to generate the QuestaSim qrun argument (.f) template file from the current Radiant project settings, including RTL, test bench, and other files. If there is no open Radiant project, this command will fail to run.

This command does not support modifying the original file list of the Radiant project. Simulation Wizard does not call this command to generate the qrun argument file. If you want to modify simulation files, you can edit the qrun argument file manually.

Note

The sim_generate_script command does not launch QuestaSim as the Simulation Wizard. However, as long as QuestaSim executes the command correctly, it can generate any type of file as an extension of the script.

sim_generate_script

Generate a simulation script for use with ModelSim or QuestaSim

Syntax

```
sim_generate_script -top <top module> [-qrun <true|false>]
[-stage <rtl|postsyn|postroute|timing>] [-inst <instance name>]
[-debug] [-run <timesteps><time_units>|0]
[-t <multiplier><time_unit>] [-no_gui] [-w] -o <do file>
```

Arguments

- **-top**: top module for simulation.
- **-qrun**: invoke QuestaSim with Qrun (optional).
- **-stage**: valid types are 'rtl', 'postsyn', 'postroute', 'timing'. 'rtl' is default (optional).
- **-inst**: specify instance for the associated SDF file (optional).
- **-debug**: apply full debug mode to the simulator (optional).
- **-run**: specify the number of timesteps for the simulation to run (optional). valid <time_unit> must be one of the fs, ps, ns, us, ms, sec. '0' means 'run -all'.
- **-t**: specify the simulator time resolution (optional). <multiplier> is optional. <time_unit> must be one of the fs, ps, ns, us, ms, sec.
- **-no_gui**: runs scripted batch simulations (optional).
- **-w**: allow to overwrite existed simulation do file or arguments file (optional).
- **-o**: specify simulation do file path or arguments file path.

Example

```
sim_generate_script -top test -run 0 -w -o test.f
sim_generate_script -top count -stage rtl -o count.f
```

See Also

- “Simulation Libraries Compilation Tcl Commands” on page 1643

Simulation Libraries Compilation Tcl Command Example

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 QuestaSim is specified by -sim_path.

```
cmpl_libs -sim_path C:/questasim64_10.4e/win64 -target_path c:/
mti_libs
```

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 Radiant software Reveal Inserter-related Tcl command options and describes option functionality.

Table 28: Reveal Inserter Tcl Commands

Command	Description
rvl_add_controller	Add Controller core.
rvl_add_core	Add Inserter core.
rvl_add_te	Add Inserter trigger expression.
rvl_add_trace	Add Inserter trace signals.
rvl_add_tu	Add Inserter trigger unit.
rvl_close_project	Close Inserter project.
rvl_del_controller	Delete Controller core.
rvl_del_core	Delete Inserter core.
rvl_del_te	Delete Inserter trigger expression.
rvl_del_trace	Delete Inserter trace signal.
rvl_del_tu	Delete inserter trigger unit.
rvl_group_trace	Group Inserter trace signals.
rvl_list_core	List Inserter core.
rvl_list_te	List Inserter trigger expression.
rvl_list_trace	List inserter trace signals.
rvl_list_tu	List Inserter trigger unit.
rvl_move_trace	Move Inserter trace signals.
rvl_new_project	Create inserter project.
rvl_open_project	Open Inserter project.
rvl_rename_core	Rename Inserter core.
rvl_rename_te	Rename Inserter trigger expression.
rvl_rename_trace	Rename Inserter trace bus.
rva_rename_tu	Rename Inserter trigger unit.
rvl_run_project	Run Inserter project.
rvl_save_project	Save Inserter project.
rvl_set_controller	List or set Controller options.

Table 28: Reveal Inserter Tcl Commands

Command	Description
rvl_set_core	Set Inserter core.
rvl_set_te	Set Inserter trigger expression.
rvl_set_traceoptn	List or set Inserter trace options.
rvl_set_trigoptn	List or set Inserter trigger options.
rvl_set_tu	Set Inserter trigger unit.
rvl_ungroup_trace	Ungroup Inserter trace signals.

rvl_add_controller

Commands to add the Controller Core into current project.

Syntax

```
rvl_add_core <core name>
```

Arguments

- **None**

Example:

```
rvl_add_controller
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_add_core

Commands to add add a new core in current project.

Syntax

```
rvl_add_controller
```

Arguments

- **<core name>:** Specify the core name.

Example:

```
rvl_add_core top_LA1
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_add_te

Commands to add a new trigger expression to a debug core in current project.

Syntax

```
rvl_add_te [-core <core name>] [-ram EBR|Slice] [-name <new TE
name>] [-expression <expreesion string>]
[-max_seq_depth <max depth>] [-max_event_count
<max event count>]
```

Arguments

- **-core <core name>:** Specify a core name (optional).
- **-ram EBR|Slice:** Specify the RAM type (optional).
- **-name <new TE name>:** Specify the TE name (optional).
- **-expression <expreesion string>:** Set Expression. A default trigger expression name will be created if it is omitted in command.
- **-max_seq_depth <max depth>:** Set the Max Sequence Depth (optional).
- **-max_event_count <max event count>:** Set the Max Event Counter (optional).

Example:

```
rvl_new_project -overwrite newRvl -stage presyn
rvl_set_te -expression tu1 TE3
rvl_set_te -max_seq_depth 8 TE3
rvl_set_te -max_event_count 32 TE3
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_add_trace

Commands to add signals in a debug core in current project.

Syntax

```
rvl_add_trace [-core <core name>] [-insert_at <position>]
<signals list>

You can specify an existing trace signal/bus name or a position
number in a trace bus as the inserting position.
```

Arguments

- **-core <core name>**: Specify a core name (optional)
- **-insert_at <position>**: Specify the inserting position (optional)
- **<signals list>**: Specify signals.

Example:

```
rvl_add_trace -insert_at leds {clk2}
rvl_add_trace -insert_at leds {clkoe2 clkout1}
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_add_tu

Commands to add a new trigger unit to a debug core in current project.

Syntax

```
rvl_add_tu [-core <core name>] [-radix bin|oct|dec|hex]
[-name <new TU name>] <TU definition>
```

Arguments

- **-airflow <airflow type>:** This option specifies airflow type
- **-core <core name>**: Specify a core name (optional).
- **-radix bin|oct|dec|hex**: Specify the Radix (optional).
- **-name <new TU name>**: Specify the TU name (optional).
- **TU definition**: TU definition format: "{signal list} Operator Value" (optional).

  Operator must be "==", "!=", ">", ">=", "<", "<=", ".RE."(rising edge), ".FE."(falling edge) and ".SC."(serial compare).

  A default trigger unit name will be created if it's omitted in command.

Example:

```
rvl_add_tu -radix bin
rvl_add_tu -radix bin -name
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_close_project

Commands to close the current reveal insert project.

Syntax

```
rvl_close_project [-force]
```

Arguments

- **-force**: Force to close (optional).

Example:

```
rvl_close_project -force
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_del_controller

Commands to remove the Controller Core from current project.

Syntax

```
rvl_del_controller
```

Arguments

- None

Example:

```
rvl_del_controller
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_del_core

Commands to remove an existing core from current project.

Syntax

```
rvl_del_core <core name>
```

Arguments

- **-<core name>**: Specify the core name

Example:

```
rvl_del_core top_LA1
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_del_te

Commands to delete an existing trigger expression in a debug core in current project.

Syntax

```
rvl_del_te [-core <core name>] [TE name]
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-name <TE name>:** Specify the TE name (optional).

Example:

```
rvl_del_te TE3
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_del_trace

Commands to delete trace signals in a debug core in current project

Syntax

```
rvl_del_trace [-core <core name>] <signals list>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-<signals list>**: Specify signals.

Example:

```
rvl_del_trace {clk2}
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_del_tu

Commands to delete an existing trigger unit in a debug core in current project.

Syntax

```
rvl_del_tu [-core <core name>] <TU name>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **<TU name>:** Specify the TU name (optional).

Example:

```
rvl_del_tu tuNew
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_group_trace

Commands to group specified trace signals in a debug core in current project into a bus.

Syntax

```
rvl_group_trace [-core <core name>] -bus <bus name> <signals
list>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-bus <bus name>**: Specify a bus name.
- **-<signals list>**: Specify signals.

Example:

```
rvl_group_trace -bus BusNew {clkout1 clkoe2}
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_list_core

Commands to list the default core in current project.

Syntax

```
rvl_list_core
```

Arguments

- **None**

Example:

```
rvl_list_core
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_list_te

Commands to list all trigger expressions in a debug core in current project.

Syntax

```
rvl_list_te [-core <core name>]
```

Arguments

- **-core <core name>**: Specify a core name (optional).

Example:

```
rvl_list_te
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_list_trace

Commands to list all trace signals in a debug core in current project

Syntax

```
rvl_list_trace [-core <core name>]
```

Arguments

- **-core <core name>**: Specify a core name (optional).

Example:

```
rvl_list_trace
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_list_tu

Commands to list all trigger unit in a debug core in current project.

Syntax

```
rvl_list_tu [-core <core name>]
```

Arguments

- **-core <core name>**: Specify a core name (optional).

Example:

```
rvl_list_tu
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_move_trace

Commands to move and rearrange the order of signals in a debug core in current project.

Syntax

```
rvl_move_trace [-core <core name>] [-move_to <position>]
<signals list>
```

You can specify an existing trace signal/bus name or a position number in a trace bus as the new position.

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-move_to <position>:** Specify the new position (optional).
- **<signals list>**: Specify signals.

Example:

```
rvl_move_trace -move_to clkoe2 {clkout1}
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_new_project

Commands to create a new Reveal Inserter project.

Syntax

```
rvl_new_project [-overwrite <new rvl file name>]
[-stage presyn|postsyn|postpar]
```

Arguments

- **-overwrite <new rvl file name>**: Specify project name (optional).
- **-stage presyn|postsyn|postpar:** Specify the Debug Stage.

Example:

```
rvl_new_project -overwrite newRvl -stage presyn
rvl_new_project -stage presyn
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_open_project

Commands to open a Reveal Inserter project file.

Syntax

```
rvl_open_project <reveal project file>
```

Arguments

- **<reveal project file>**: Specify the Reveal Inserter project file.

Example:

```
rvl_open_project newRvl.rvl
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_rename_core

Commands to rename an existing core in current project.

Syntax

```
rvl_rename_core <core name> <new name>
```

Arguments

- **<core name>:** Specify existing core name.
- **<new name>:** Specify the new core name.

Example:

```
rvl_rename_core top_LA1 new_LA
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_rename_te

Commands to rename an existing trigger expression in a debug core in current project.

Syntax

```
rvl_rename_te [-core <core name>] <old name> <new name>
```

Arguments

- **-overwrite <new rvl file name>**: Specify project name (optional).
- **<old name>:** Specify the old name.
- **<new name>**: Specify the new name.

Example:

```
rvl_rename_te TE2 TENew
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_rename_trace

Commands to change the name of a trace bus in a debug core in current project.

Syntax

```
rvl_rename_trace [-core <core name>] -bus <bus name> <new bus
name>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-bus <bus name>:** Specify the existing bus.
- **<new bus name>**: Specify the new bus name.

Example:

```
rvl_rename_trace -bus leds newBus
```

See Also

- “Reveal Inserter Tcl Commands” on page 1646

rvl_rename_tu

Commands to rename an existing trigger unit in a debug core in current project.

Syntax

```
rvl_rename_tu [-core <core name>] <old name> <new name>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **<old name>:** Specify the old name.
- **<new name>**: Specify the new name.

Example:

```
rvl_rename_tu tu1 tu1_new
```

See Also

- “Reveal Inserter Tcl Commands” on page 1646

rvl_run_project

Commands to run inserting debug core task or DRC checking on the current reveal insert project.

Syntax

```
rvl_run_project [-save] [-saveAs <file_name>] [-overwrite] [-
drc] [-insert_core <core_name>]...
```

Arguments

- **-save**: Save the project before run command (optional).
- **-saveAs:** Save as a different file before run command (optional).
- **-overwrite**: Overwrite the existing file if the saved as to file exists already.
- **-drc:** Run DRC checking only (optional).
- **-insert_core <core_name>**: Specify the core to be inserted.

  All cores will be inserted if none is specified.

Example:

```
rvl_run_project
rvl_run_project -drc
rvl_run_project -save
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_save_project

Commands to save the current reveal insert project.

Syntax

```
rvl_save_project [-overwrite <reveal project file>]
```

Arguments

- **-overwrite <reveal project file>**: Specify project name (optional).

Example:

```
rvl_save_project newRvl.rvl
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_set_controller

Commands to list or set options of Controller items in current project.

Syntax

```
rvl_set_controller [-item LED|Switch|Register|PLL1|PLL2|...]
      [-set_opt [Insert=on|off] [Width=1|2|3|...|32]]
      [-set_sig [SWn=signal] [LEDn=signal] [Clock=clk_signal]
[Clock_enable=en_signal]
      [Wr_Rdn=wr_rdn_signal] [Address=addr_bus]
[WData=wdata_bus] [RData=rdata_bus]]
```

Arguments

- **-item LED|Switch|Register|PLL1|PLL2|...**: Specify item to be set (optional).
- **-set_opt [Insert=on|off] [Width=1|2|3|...|32]**
  - **Insert=on|off**: Insert the item which specified by -item (optional).
  - **Width=1|2|3|...|32**: Set width for LED and Switch (optional).
- **-set_sig [SWn=signal] [LEDn=signal] [Clock=clk_signal] [Clock_enable=en_signal] [Wr_Rdn=wr_rdn_signal] [Address=addr_bus] [WData=wdata_bus] [RData=rdata_bus]**
  - **SWn=signal**: Specify the signal where n=1 to Width for Switch (optional).
  - L**EDn=signal**: Specify the signal where n=1 to Width for LED (optional).
  - **Clock=clk_signal**: Specify the signal for Register (optional).
  - **Clock_enable=en_signal**: Specify the signal for Clock_enable of Register (optional).
  - **Wr_Rdn=wr_rdn_signal for Register**: Specify the signal for Wr_Rdn of Register.
  - **Address=addr_bus**: Specify the Address for Wr_Rdn of Register.
  - **WData=wdata_bus**: Specify the WData for Register (optional).
  - **RData=rdata_bus**: Specify the RData for Register (optional).

Example:

```
rvl_set_controller -item PLL1 -set_opt {insert=on}
rvl_set_controller -item Switch -del SW2
rvl_set_controller -item Switch -set_sig {SW0=clkoe1}
```

See Also

- “Reveal Inserter Tcl Commands” on page 1646

rvl_set_core

Commands to set a core as the default core in current project.

Syntax

```
rvl_set_core [<core name>]
```

Arguments

- **<core name>**: Specify the core name (optional).

Example:

```
rvl_set_core
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_set_te

Commands to change an existing trigger expression in a debug core in current project.

Syntax

```
rvl_set_te [-core <core name>] [-ram <EBR|Slice>]
[-expression <expreesion string>][-max_seq_depth <max depth>]
[-max_event_count <max event count>] <TE name>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-ram <EBR|Slice>**: Specify the RAM type (optional).
- **-expression <expreesion string>**: Set Expression (optional).
- **-max_seq_depth <max depth>**: Set the Max Sequence Depth (optional).
- **-max_event_count <max event count>**: Set the Max Event Counter (optional).
- **-name <TE name>**: Specify the TE name (optional).

Example:

```
rvl_set_te -ram EBR TE3
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_set_traceoptn

Commands to list or set trace options of a debug core in current project.

Syntax

```
rvl_set_traceoptn [-core <core name>]
        [SampleClk = <signal name>]
        [SampleEnable = on|off]
        [SampleEnableSig = <signal name>]
        [SampleEnablePri = Active_Low|Active_High]
        [BufferDepth = 16|32|...|65536]
        [Implementation = EBR|DRAM]
        [Timestamp = on|off]
        [TimestampBits = <bits>]
        [CaptureMode = single|multiple]
        [MinimumSamplesPerTrigger = 8|16|32|64|128]
        [IncludeTrigger = on|off]
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **SampleClk = <signal name>**: Specify the Sample Clock (optional).
- **SampleEnable = on|off**: Enable the Sample (optional).
- **SampleEnableSig = <signal name>**: Specify the signal for Sample (optional). (depend on SampleEnable is on)
- **SampleEnablePri = Active_Low|Active_High**: Specify the mode of Sample (optional). (depend on SampleEnable is on)
- **BufferDepth = 16|32|...|65536**: Specify the Buffer Depth (optional).
- **Implementation = EBR|DRAM**: Specify the mode of Implementation (optional).
- **Timestamp = on|off**: Enable the Timestamp (optional).
- **TimestampBits = <bits>**: Specify the Bits of Timestamp, which must be in the range from (BufferDepth bits + 1) to 63 (optional). (depend on CaptureMode is multiple and BufferDepth)
- **CaptureMode = single|multiple**: Specify the Capture Mode (optional).
- **MinimumSamplesPerTrigger = 8|16|32|64|128**: Specify the minimum samples per trigger (optional). (depend on CaptureMode is multiple and BufferDepth)
- **IncludeTrigger = on|off**: Enable to include trigger signals in trace data (optional).

Example:

```
rvl_set_traceoptn
rvl_set_traceoptn CaptureMode=multiple
MinimumSamplesPerTrigger=16
rvl_set_traceoptn SampleEnableSig=dn31
rvl_set_traceoptn TimestampBits=63
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_set_trigoptn

Commands to list or set trigger options of a debug core in current project.

Syntax

```
rvl_set_trigoptn [-core <core name>]
        [DefaultRadix = bin|oct|dec|hex]
        [EventCounter = on|off]
        [CounterValue = 2|4|8|16|...|65536]
        [TriggerOut = on|off]
        [OutNetType = IO|NET|BOTH]
        [OutNetName = <net name>]
        [OutNetPri = Active_Low|Active_High]
        [OutNetMPW = <pulse number>]
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **DefaultRadix = bin|oct|dec|hex**: Specify the default Radix (optional).
- **EventCounter = on|off**: Enable the Event Counter (optional).
- **CounterValue = 2|4|8|16|...|65536**: Specify the value of Counter (optional). (depend on EventCounter is on)
- **TriggerOut = on|off**: Enable the Trigger Out (optional).
- **OutNetType = IO|NET|BOTH**: Specify the Net Type (optional). (depend on TriggerOut is on)
- **OutNetName = <net name>**: Specify the Net Name (optional). (depend on TriggerOut is on)
- **OutNetPri = Active_Low|Active_High**: Specify the Polarity mode (optional). (depend on TriggerOut is on)
- **OutNetMPW = <pulse number>**: Specify the Net Type (optional). (depend on TriggerOut is on)

Example:

```
rvl_set_trigoptn
rvl_set_trigoptn TriggerOut=on
rvl_set_trigoptn OutNetMPW=1
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

rvl_set_tu

Commands to set a trigger unit in a debug core in current project.

Syntax

```
rvl_set_tu [-core <core name>] [-radix bin|oct|dec|hex]
-name <TU name> [-add_sig <signal list>] [-del_sig
<signal list>] [-set_sig <signal list>]
[-expr <TU definition>] [-op <operator>] [-val <value>]
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **-radix bin|oct|dec|hex**: Specify the Radix (optional).
- **-name <TU name>**: Specify the TU name (optional).
- **-add_sig <signal list>**: Add signals (optional).
- **-del_sig <signal list>**: Delete signals (optional).
- **-set_sig <signal list>**: Set signals (optional).
- **-expr <TU definition>**:Define the TU Expression (optional). TU definition format: "{signal list} Operator Value"
- **-op <operator>**: Define the TU Operator (optional). Operator must be "==", "!=", ">", ">=", "<", "<=", ".RE."(rising edge), ".FE."(falling edge) and ".SC."(serial compare)
- **-val <value>**: Set value for Operator (optional).

Example:

```
rvl_set_tu -name fff -radix Oct
rvl_set_tu -name fff -op .RE.
rvl_set_tu -name fff -val 2
```

See Also

- “Reveal Inserter Tcl Commands” on page 1646

rvl_ungroup_trace

Commands to ungroup trace signals in a trace bus in a debug core in current project.

Syntax

```
rvl_ungroup_trace [-core <core name>] <bus name>
```

Arguments

- **-core <core name>**: Specify a core name (optional).
- **<bus name>**: Specify a bus name.

Example:

```
rvl_ungroup_trace BusNew
```

See Also

- "Reveal Inserter Tcl Commands" on page 1646

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 Radiant software Reveal Analyzer-related Tcl command options and describes option functionality.

Table 29: Reveal Analyzer Tcl Commands

Command	Description
rva_add_token	Add Analyzer token.
rva_add_tokenset	Add Analyzer token set.
rva_close_controller	Close Controller connection.
rva_close_project	Close Analyzer project.
rva_del_token	Delete Analyzer token.
rva_del_tokenset	Delete Analyzer token set.
rva_export_project	Export Analyzer waveform.
rva_export_tokenset	Export Analyzer token set.
rva_get_trace	List Analyzer trace signals.
rva_import_tokenset	Import Analyser token set.
rva_manualtrig	Manual Trigger to capture data.
rva_new_project	Create Analyzer project.
rva_open_controller	Open Controller connection.
rva_open_project	Open Analyzer project file.
rva_read_controller	Read data from 32-bit address in hex.
rva_rename_te	Rename Analyzer trigger expression.
rva_rename_tu	Rename Analyzer trigger unit.
rva_run	Run Analyzer project.
rva_run_controller	Run Controller command.
rva_save_project	Save the current Analyzer project.

Table 29: Reveal Analyzer Tcl Commands

Command	Description
rva_set_controller	Set Controller options.
rva_set_core	Set Analyzer core.
rva_set_project	Set Analyzer options.
rva_set_te	Set Analyzer trigger expression.
rva_set_token	Set Analyzer token.
rva_set_tokenset	Set Analyzer token set.
rvl_set_trigoptn	Set Analyzer trigger options.
rva_set_tu	Set Analyzer trigger unit.
rva_stop	Stop Analyzer project without capturing data.
rva_target_controller	Set Controller core as target.
rva_write_controller	Write Controller data to 32-bit address in hex.

rva_add_token

Adds a token with new name and value in a specific token set.

Syntax

```
rva_add_token <tokenset name> <name=value>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_add_tokenset

Adds a token set.

Syntax

```
rva_add_tokenset [-tokenset <tokenset name>]
[-bits <token bits>] [-token <name=value>]
```

Arguments

- **tokenset**: Sets the token set name
- **-bits**: Sets the token set bits
- **-token**: Add extra token

  If no arguments are specified, add a token set with default name and bits.

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_close_controller

Closes Controller connection to Lattice device after read/write has finished.

Syntax

```
rva_close_controller
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_close_project

Closes the current Reveal Analyzer project.

Syntax

```
rva_close_controller
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_del_token

Deletes a specific token in a token set.

Syntax

```
rva_del_token <tokenset name> <token name>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_del_tokenset

Deletes the specific token set.

Syntax

```
rva_del_tokenset <tokenset name>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_export_project

Exports waveform in VCD or TEXT from the current Reveal Analyzer project.

Syntax

```
rva_export_project <-vcd|-txt> <file name> [option]
```

Arguments

- **-vcd**: <file name> [module <title>]
- **-txt**: <file name> [-siglist <signal list>]

  By default, the module title is "<unknown>" and all signals are exported.

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_export_tokenset

Exports all token set to a specific file.

Syntax

```
rva_export_tokenset <file name>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_get_trace

Lists all trace signals in Analyzer core.

Syntax

```
rva_get_trace
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_import_tokenset

Imports and merges all token set from a specific file.

Syntax

```
rva_import_tokenset <file name>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_manualtrig

Manually triggers Reveal Analyzer to capture data.

Syntax

```
rva_manualtrig
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_new_project

Creates a new Reveal Analyzer project.

Syntax

```
rva_new_project -rva <file name> -rvl <file name> -dev <val> -
cable <val> -port <val> [-xcf <file name>]
```

Arguments

- **-rva**: Set the new rva file name
- **-rvl**: Set the associated rvl file name
- **-dev**: Set the device name
- **-cable**: Set type of cable as USB or USB2
- **-port**: Set logical port of cable as integer
- **-xcf**: Optional xcf file name for multiple device chain

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_open_controller

Open Controller connection to Lattice device before read/write began.

Syntax

```
rva_open_controller
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_open_project

Opens Controller connection to Lattice device before read/write begins.

Syntax

```
rva_open_project <rva file name>
```

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_read_controller

Reads data from 32bit address in hex.

Syntax

```
rva_read_controller -addr <addr32>
```

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_rename_te

Renames an existing trigger expression in a debug core in current project.

Syntax

```
rva_rename_te <name> <new name>
```

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_rename_tu

Renames an existing trigger unit in a debug core in the current project.

Syntax

```
rva_rename_tu <name> <new name>
```

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_run

Runs Reveal Analyzer until trigger condition to capture data.

Syntax

```
rva_run
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_run_controller

Runs command for Virtual LED, Virtual Switch, User Memory, and Hard IP.

Runs command for User Control Register and User Status Register.

Syntax

```
rva_run_controller -read_led|-read_switch|-write_switch
<data>|-dump_memfile <mem_file>|-load_memfile <mem_file>|
-read_ip <ipname>|-write_ip <ipname>|-eye_monitor <ipname>|
-read_status <regname>|-read_control <regname>|-write_control
<regname> -data <data>|-toggle_control <regname>|
-dump_status <reg_file>|-dump_control <reg_file>|-load_control
<reg_file>
```

Arguments

- **read_led**: Reads data from Virtual LED
- **read_switch**: Reads data from Virtual Switch
- **write_switch**: Writes data to Virtual Switch
- **dump_memfile**: Dumps data from User Memory to mem_file
- **load_memfile**: Loads data from mem_file to User Memory
- **read_ip**: Reads data from Hard IP register of ipname
- **write_ip**: Writes data to Hard IP register of ipname
- **eye_monitor**: Runs Eye-Opening Monitor for PCS channel of ipname
- **read_status**: Reads data from User Status Register of regname
- **read_control**: Reads data from User Control Register of regname
- **write_control**: Writes data to User Control Register of regname
- **toggle_control**: Toggles data bit in User Control Register of regname
- **dump_status**: Dumps data from User Status Register to reg_file
- **dump_control**: Dumps data from User Control Register to reg_file

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_save_project

Saves the current Reveal Analyzer project.

Syntax

```
rva_save_project [<new rva file name>]
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_controller

Sets the options for Controller core.

Syntax

```
rva_set_controller [-option <value>]
```

- **cable_type**: Sets type of cable as USB or USB2.
- **cable_port**: Sets logical port of cable as integer.

  If no arguments are specified, return list of options and values.

```
rva_set_controller -hard_ip <ipname> [-option <value>]
```

- **hard_ip**: Sets options of DPHY1, DPHY2, I2CFIFO1, SGMIICDR1, SGMIICDR2, PCIELL1, PCSCH1 to PCSCH8

  If no option/value specified, then return list of options and values.

Arguments

- **DPHY1, DPHY2 options and values**:
  - **high_speed_select**: Sets value as "1.5Gbps and below", "2.5Gbps"
  - **bypass_pll_clock**: Sets value as Enabled, Disabled
  - **pll_cn_counter**: Sets value as integer from 0 to 31
  - **pll_cm_counter**: Sets value as integer from 0 to 255
  - **pll_co_counter**: Sets value as integer from 0 to 7
- **I2CFIFO1 options and values**:
  - **generic_call_response**: Sets value as Enabled, Disabled
  - **sda_input_delay**: Sets value as Enabled, Disabled
  - **sda_output_delay**: Sets value as Enabled, Disabled
  - **fifo_mode_select**: Sets value as "Register Mode", "FIFO Mode"
  - **fifo_clock_stretching**: Sets value as Enabled, Disabled
  - **fifo_almost_full_value**: Sets value as integer from 0 to 31
  - **fifo_sleep_clock**: Sets value as Enabled, Disabled

- **slave_address_match_wakeup**: Sets value as Enabled, Disabled
- **fifo_almost_full_wakeup**: Sets value as Enabled, Disabled
- **master_read_complete_wakeup**: Sets value as Enabled, Disabled

- **SGMIICDR1, SGMIICDR2 options and values**:
  - **cdr_loss_of_lock**: Sets value as "Lock=+/-1000ppm x2 Unlock=+/-1500ppm x2", "Lock=+/-2000ppm x2 Unlock=+/-2500ppm x2", "Lock=+/-4000ppm Unlock=+/-7000ppm", "Lock=+/-300ppm Unlock=+/-450ppm"
  - **loopback_control**: Sets value as Disabled, "Bit Clock to IO", "Serial Data to IO"
- **PCIELL1 options and values**:
  - **swing_control**: Sets value as full, reduced
- **PCSCH1 to PCSCH8 options and values**:
  - **tx_differential_amplitude**: Sets value as 1000mV, 900mV, 800mV, 700mV, 600mV, 500mV, 400mV, 300mV, 200mV, 100mV, 0mV
  - **tx_output_termination**: Sets value as "100 ohm", "150 ohm"
  - **tx_postcursor_ratio**: Sets value as integer from 0 to 63 of 128th
  - **tx_precursor_ratio**: Sets value as integer from 0 to 63 of 128th
  - **tx_invert_polarity**: Sets value as on, off
  - **rx_input_termination**: Set value as "100 ohm", "150 ohm"
  - **rx_equalization**: Sets value as "RL2plus 8GT/s", "RL2plus 5GT/s", "RL2plus 2.5GT/s", "SS_LMS 8GT/s", "SS_LMS 5GT/s", "SS_LMS 2.5GT/s"
  - **rx_invert_polarity**: Sets value as on, off
  - **loopback_mode**: Set value as Disabled, "PMA Near-End Serial Loopback", "PMA Far-End Parallel Loopback", "8B/10B PCS Near-End Parallel Loopback", "8B/10B PCS Far-End Parallel Loopback", "64B/66B PCS Loopback Path A", "64B/66B PCS Loopback Path B", "64B/66B PCS Loopback Path C"
  - **reset_tx_mpcs**: Sets value as on, off
  - **reset_rx_mpcs**: Sets value as on, off
  - **reset_tx_pll**: Sets value as on, off
  - **reset_rx_pll**: Sets value as on, off
  - **force_tx_driver**: Sets value as on, off
  - **force_rx_driver**: Sets value as on, off
  - **eye_quality**: Sets quality of eye diagram as high, normal, low

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_set_core

Sets a core as the default core in the current project.

Syntax

```
rva_set_core [-name <name>] [-run <on|off>]
```

Arguments

- **name**: Selects core. This is needed for other actions.
- **run**: Turns the run option on or off for core.

  If no arguments are specified, return list of core.

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_project

Sets the current Reveal Analyzer project options.

Syntax

```
rva_set_project [-frequency <val>] [-period <val>] [-tckdelay
<val>] [-cabletype <val>] [-cableport <val>]
```

Arguments

- **frequency**: Sets the frequency value for sample clockin MHz
- **period**: Sets the period value for sample clock in ns or ps
- **tckdelay**: Sets the TCK Divider Setting upto 30
- **cabletype**: Sets the type of cable. Values must be "LATTICE", "USB", "USB2"
- **cableport**: Sets the port number as integer >=0

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_te

Changes an existing trigger expression in a debug core in current project.

Syntax

```
rva_set_te [-name <name>] [-expression <expression list>] [-
enable <on|off>]
```

Arguments

- **name**: Selects TE. If no options are specified, return options and values for the selected TE.
- **expression**: Sets TE expression
- **enable**: Enables/disables TE

  If no arguments are specified, return list of TE.

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_token

Selects a specific token in a specific token set.

Syntax

```
rva_set_token <tokenset name> <token name> -name <new token
name> -value <new token value>
```

Arguments

- **name**: Sets token name
- **value**: Sets token value

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_tokenset

Selects a specific token set.

Syntax

```
rva_set_tokenset  <tokenset name> [-name <new token set name>]
[-bits <new token bits>]
```

Arguments

- **name**: Renames a token set
- **bits**: Sets number of bits in token

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_set_trigoptn

Sets trigger options of Analyzer core in the current project.

Syntax

```
rva_set_trigoptn  [-teall <AND|OR>] [-finalcounter <on|off>]
[-finalcountervalue <val>][-samples <val>]
[-numtriggers <val>] [-position <pre|center|post|val>]
```

Arguments

- **teall**: Sets AND ALL or OR ALL for all TE
- **finalcounter**: Turns final trigger counter on/off
- **finalcountervalue**: Sets final trigger counter value
- **samples**: Sets number of samples to capture
- **numtriggers**: Sets number of triggers to capture
- **position**: Sets trigger position to pre-selected or user value

  If no arguments specified, then return list of options.

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_set_tu

Sets a trigger unit in a debug core in current project.

Syntax

```
rva_set_tu  [-name <name>] [-operator {== | != | > | >= | < |
<= | "rising edge" | "falling edge"}][-value <value>] [-radix
{bin | oct | dec | hex | <token>}]
```

Arguments

- **name**: Selects TU. If no options are specified, return options and value for the selected TU.
- **operator**: Sets TU comparison operator
- **value**: Sets TU value
- **radix**: Sets TU radix

  If no arguments are specified, return list of TU.

See Also

- “Reveal Analyzer Tcl Commands” on page 1664

rva_stop

Stops Reveal Analyzer without capturing data.

Syntax

```
rva_stop
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_target_controller

Sets Controller core as target before read/write begins.

Syntax

```
rva_target_controller
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

rva_write_controller

Writes data to 32bit address in hex.

Syntax

```
rva_write_controller -addr <addr32> -data <data>
```

See Also

- "Reveal Analyzer Tcl Commands" on page 1664

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 Radiant software Power Calculator-related Tcl command options and describes option functionality.

Table 30: Power Calculator Tcl Commands

Command	Description
pwc_add_inactiveimpl	Add inactive implementation name.
pwc_add_ipblock	Add IP Block row.
pwc_calculate	Run power calculation for current project.
pwc_close_project	Close the current project.
pwc_gen_csvreport	Generate CSV report and write to file.
pwc_gen_htmlreport	Generate HTML report and write to file.
pwc_gen_report	Generate text report and write to file.
pwc_new_project	Create a new project.
pwc_open_project	Open a project file.
pwc_remove_ipblock	Remove IP Block row.
pwc_save_project	Save the current project.
pwc_set_af	Set activity factor.
pwc_set_afpervcd	Open vcd file and set frequency and activity factor.
pwc_set_ambienttemp	Set ambient temperature value.
pwc_set_device	Set device information.
pwc_set_estimation	Set estimated routing option.
pwc_set_freq	Set frequency.
pwc_set_ipblock	Set IP Block row.
pwc_set_processtype	Set device power process type.
pwc_set_supply	Set multiplication factor and voltage of named power supply.
pwc_set_thetaja	Set thermal profile.
pwc_set_processtype	Set device power process type.
pwc_set_supply	Set multiplication factor and voltage of named power supply.
pwc_set_thetaja	Set thermal profile.
pwc_set_userenteredtj	Set user entered Tj value.

pwc_add_inactiveimpl

Commands to add inactive implementation name.

Syntax

```
pwc_add_inactiveimpl [<implementation name>]
```

Arguments

- **<implementation name>**: This option specifies implementation name.

Example:

```
pwc_add_inactiveimpl impl1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_add_ipblock

Commands to add IP Block row.

Syntax

```
pwc_add_ipblock -iptype <iptype name>
```

Arguments

- **iptype <iptype name>**: This option specifies IP type.

Example:

```
pwc_add_ipblock -iptype Logic
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_calculate

Commands to run power calculation for current project.

Syntax

```
pwc_calculate
```

Example:

```
pwc_calculate
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_close_project

Commands to close the current project.

Syntax

```
pwc_close_project
```

Example:

```
pwc_close_project
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_gen_csvreport

Commands to generate CSV report and write to file.

Syntax

```
pwc_gen_csvreport <report file>
```

Arguments

- **<report file>**: This option specifies the report file name.

Example:

```
pwc_gen_csvreport test.csv
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_gen_htmlreport

Commands to generate HTML report and write to file.

Syntax

```
pwc_gen_htmlreport <report file>
```

Arguments

- **<report file>**: This option specifies the report file name.

Example:

```
pwc_gen_htmlreport test.html
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_gen_report

Commands to generate text report and write to file.

Syntax

```
pwc_gen_report <report file>
```

Arguments

- **<report file>**: This option specifies the report file name.

Example:

```
pwc_gen_report test.html
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_new_project

Commands to create a new project.

Syntax

```
pwc_new_project <project file>
```

Arguments

- **<project file>**: This option specifies new project file name.

Example:

```
pwc_new_project test.pcf
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_open_project

Commands to open a project file.

Syntax

```
pwc_open_project <project file>
```

Arguments

- **<project file>**: This option specifies the project file name.

Example:

```
pwc_open_project test.pcf
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_remove_ipblock

Commands to remove IP Block row.

Syntax

```
pwc_remove_ipblock -iptype <iptype name> -matchkeys {<key1>
<value1>}+
```

Arguments

- **iptype <iptype name>**: This option specifies IP type.
- **matchkeys {<key1> <value1>}+**: This option specifies match keys.

Example:

```
pwc_remove_ipblock -iptype Logic -matchkeys Clk clk_1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_save_project

Commands to save the current project.

Syntax

```
pwc_save_project <project file>
```

Arguments

- **<project_file>**: This option specifies the project file name.

Example:

```
pwc_save_project test.pcf
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_af

Commands to set activity factor.

Syntax

```
pwc_set_af <activity factor> [-keepClocksAF]|<activity factor>
-iptype <iptype name> -clk <clock name>
```

Arguments

- **<activity factor>**: This option specifies default activity factor.
- **-keepClocksAF**: This option specifies activity factor except clocks tab (optional).
- **-iptype <iptype name>**: This option is to set activity factor for given IP.
- **-clk <clock name>**: This option is to set activity factor for given IP clock name.

Example:

```
pwc_set_af 20
pwc_set_af 30 -keepClocksAF
pwc_set_af 40 -iptype Logic -clk clk_1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_afpervcd

Commands to open vcd file and set frequency and activity factor.

Syntax

```
pwc_set_afpervcd -vcd <vcd file> [-module <module name>]
```

Arguments

- **-vcd <vcd file>**: This option specifies the vcd file name.
- **-module <module name>**: This option specifies the module name (optional).

Example:

```
pwc_set_afpervcd -vcf test.vcd
pwc_set_afpervcd -vcf test.vcd -module abc
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_ambienttemp

Commands to set ambient temperature value.

Syntax

```
pwc_set_ambienttemp <ambient temperature>
```

Arguments

- **<ambient temperature>**: This option specifies implementation name (optional).

Example:

```
pwc_set_ambienttemp 30
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_device

Commands to set device information.

Syntax

```
pwc_set_device -family <family name>|-device <device name>|-
package <package name>|-speed <speed name>|-operating
<operating name>|-part <part name>
```

Arguments

- **-family <family name>**: This option specifies family name.
- **-device <device name>**: This option specifies device name.
- **-package <package name>**: This option specifies package name.
- **-speed <speed name>**: This option specifies speed name.
- **-operating <operating name>**: This option specifies operating name.
- **-part <part name>**: This option specifies part name.

Example:

```
pwc_set_device -family LIFCL
pwc_set_device -device LIFCL-17
pwc_set_device -package QFN72
pwc_set_device -speed 8_Low-Power_1.0V
pwc_set_device -operating Commercial
pwc_set_device -part LIFCL-17-8SG72Cpwc_add_inactiveimpl impl1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_estimation

Commands to set estimated routing option.

Syntax

```
pwc_set_estimation <estimated routing option>
```

Arguments

- <estimated routing option>: This option specifies estimated routing type.

Example:

```
pwc_set_estimation Low
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_freq

Commands to set frequency.

Syntax

```
pwc_set_freq <frequency>|<clock> <frequency>|-freq <frequency>
-usefreqtwr 0|1 -freqtwropt min|pref|trace
```

Arguments

- **<frequency>**: This option specifies default frequency.
- **<clock> <frequency>**: This option specifies clock frequency.
- **-freq <frequency>**: This option specifies frequency.
- **-usefreqtwr 0|1**: This option specifies if it is to set frequency by importing timing report or not.
- **-freqtwropt min|pref|trace**: This option specifies the selection for importing timing report.

Example:

```
pwc_add_inactiveimpl impl1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_ipblock

Commands to set IP Block row.

Syntax

```
pwc_set_ipblock -iptype <iptype name> -matchkeys {<key1>
<value1>}+ -setkey <key> <value>
```

Arguments

- -iptype <iptype name>: This option specifies ip type.
- -matchkeys {<key1> <value1>}+: This option specifies match keys.
- -setkey <key> <value>: This option specifies key value.

Example:

```
pwc_set_ipblock -iptype Logic -matchkeys Clk clk_1 -setkey LUT
5000
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_junctiontemp

Commands to set junction temperature value.

Syntax

```
pwc_set_junctiontemp <junction temperature>
```

Arguments

- **<junction temperature>**: This option specifies the junction temperature value.

Example:

```
pwc_set_junctiontemp 60
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_powerscreen

Commands to set power screen type.

Syntax

```
pwc_set_powerscreen <power screen>
```

Arguments

- **<power screen>**: This option specifies power screen value.

Example:

```
pwc_set_powerscreen P20
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_processtype

Commands to set device power process type.

Syntax

```
pwc_set_processtype <process type>
```

Arguments

- **<process type>** This option specifies process type.

Example:

```
pwc_set_processtype Typical
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_supply

Commands to set multiplication factor and voltage of named power supply.

Syntax

```
pwc_set_supply -type <type> -voltage <voltage> -dpm <dpm>
```

Arguments

- -type <type>: This option specifies supply type.
- -voltage <voltage>: This option specifies voltage value.
- -dpm <dpm>: This option specifies dpm value.

Example:

```
pwc_set_supply -type Vcc -voltage 0.8 -dpm 1.5
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_thetaja

Commands to set thermal profile.

Syntax

```
pwc_set_thetaja <theta JA>|-board <board type> -heatsink
<heatsink value>|-board <board type> -heatsink <heatsink type>
-airflow <airflow type>
```

Arguments

- **<theta JA>**: This option specifies user defined theta JA.
- **-board <board type>**: This option specifies board type.
- **-heatsink <heatsink value>**: This option specifies custom profile heat sink value.
- **-heatsink <heatsink type>**: This option specifies heat sink type.
- **-airflow <airflow type>**: This option specifies airflow type.

Example:

```
pwc_set_thetaja 20
pwc_set_thetaja -board Medium Board -heatsink 10
pwc_set_thetaja -board Medium Board -heatsink No Heat Sink -
airflow 0 LFM
```

See Also

- "Power Calculator Tcl Commands" on page 1676

pwc_set_userenteredtj

Commands to set user entered Tj value.

Syntax

```
pwc_set_userenteredtj 1|0
```

Arguments

- **1**: This option specifies user entered Tj.

- **0**: This option specifies auto-computed Tj.

Example:

```
pwc_set_userenteredtj 1
```

See Also

- "Power Calculator Tcl Commands" on page 1676

Engineering Change Order Tcl Commands

This section provides Engineering Change Order (ECO) extended Tcl command syntax, command options, and usage examples.

ECO Tcl Command Descriptions

The following table provides a listing of all valid Radiant software ECO-related Tcl command options and describes option functionality.

Table 31: ECO Tcl Commands

Command	Description
eco_config_block	Config general IP.
eco_config_memory	Update memory initial value.
eco_config_sysio	Config sysio setting.
eco_save_design	Save design in memory to disk.

eco_config_block

Configures general IP.

Syntax

```
eco_config_block -comp <comp_name> (<key=value>)..
```

Arguments

- **-comp <comp_name>**: Specifies the IP instance name.
- **(<key=value>)...**: Specifies attribute key and value.

Example:

```
eco_config_ip -comp DPHY1 {"Speed Select"="2.5Gbps"}
```

See Also

- "Engineering Change Order Tcl Commands" on page 1688

eco_config_memory

Updates memory initial value.

Syntax

```
eco_config_memory -mem_id <memory_id> (-init_file <mem_file> -
format HEX|BIN|ADDR) | -all_0 | -all_1
```

Arguments

- **-mem_id**: Specifies the memory ID.
- **-init_file**: Specifies the memory file path.
- **-format**: Specifies memory file format. The valid format is HEX, BIN or ADDR.
- **-all_0**: Config memory all data to zero.
- **-all_1**: Config memory all data to 1. Option '-init_file' '-all_0' and '-all_1' are mutually exclusive.

Example:

```
eco_config_memory -mem_id {top} -all_0
eco_config_memory -mem_id {ram_dq} -all_1
eco_config_mem -mem_id {mem} -init_file {D:/mem/init_hex.mem} -
format HEX
eco_config_memory -mem_id
{large_ram_inst.lscc_lram_inst.genblk1.LRAM_0.LRAM_0.LRAM_inst}
-init_file {D:/memory/lram_eco/mem.bin} -format BIN
```

See Also

- "Engineering Change Order Tcl Commands" on page 1688

eco_config_sysio

Config sysio setting.

Syntax

```
eco_config_sysio -comp <comp_name> (<key=value>)...
```

Arguments

- **-comp <comp_name>**: Specifies the instance name.
- **(<key=value>)...**: Specifies one or more sysio key and value settings.

Example:

```
eco_config_sysio -comp {c[0]}
{diffdrive=NA;diffresistor=OFF;diff_invert=NA;drive=8;
early_io=ON;glitchfilter=OFF;hysteresis=NA;opendrain=OFF;
pullmode=NONE;slewrate=SLOW;termination=OFF;vref=OFF}
eco_config_sysio -comp {c[1]}
{diffdrive=NA;diffresistor=OFF;diff_invert=NA;drive=50RS}
```

See Also

- "Engineering Change Order Tcl Commands" on page 1688

eco_save_design

Saves current design to disk.

Syntax

```
eco_save_design [-udb <udb_file>]
```

Arguments

- **-udb <udb_file>**: Specifies the output udb file (optional).

Example:

```
eco_save_design -udb eco.udb
eco_save_design
```

See Also

- "Engineering Change Order Tcl Commands" on page 1688

IP Version Update Tcl Commands

This section provides the IP upgrade version Tcl command syntax, command options, and usage examples.

IP Version Update Command Descriptions

The following table provides a listing of all valid Radiant software IP version update-related Tcl command options and describes option functionality.

Table 32: IP Version Update Commands

Command	Description
ip_catalog_list	List supported IPs in local.
ip_upgrade	Update all .ipx resource files with the new IP version in the Radiant project.

ip_catalog_list

Return a list of IP, local IP contains the IP from Radiant install package, IP server and customer's packaged IP, server IP contains the IP in server. Only get local IP by defalut.

Syntax

```
ip_catalog_list [-name <IP wild name>] [-library <IP wild
library>][-vendor <IP wild vendor>][-version <IP wild version>]
[-server][-supported][-installed]
```

Arguments

- **-server:** Get IP list in server (optional).
- **-supported:** Only list supported local IPs (optional).
- **-installed:** Only list installed local IPs (optional).
- **-latest:** Only list latest version (optional).
- **-name <IP wild name>:** List IP with specified name (optional).
- **-library <IP wild library>:** List IP with specified library (optional).
- **-vendor <IP wild vendor>:** List IP with specified vendor (optional).
- **-version <IP wild version>:** List IP with specified version (optional).

Example:

```
ip_catalog_list -server -name I2C
ip_catalog_list -installed
```

See Also

- "IP Version Update Tcl Commands" on page 1690

ip_upgrade

Upgrade and regenerate the IP in a project with a more recent version.

Syntax

```
ip_upgrade [-all] [-ipx <ipx file path>] [-force_update]
[-version <version number>]
```

Arguments

- **-all:** Upgrade all IPs to the latest version that can be found locally (optional).
- **-ipx <ipx file path>:** Upgrade one IP to the latest version that can be found locally (optional).
- **-version <version number>:** Upgrade one IP to the specified version that can be found locally (optional).
- **-force_update:** Upgrade the ipx file with latest version regardless of whether it is compatible or not, which may cause generation errors (optional).

Example:

```
ip_upgrade -ipx D:/radiant_design/counter/adder1.ipx -
force_update
ip_upgrade -ipx D:/radiant_design/counter/adder1.ipx -version
1.3.0
ip_upgrade -all -force_update
```

- "IP Version Update Tcl Commands" on page 1690

Message Control Tcl Commands

This section provides Message Control extended Tcl command syntax, command options, and usage examples.

Message Control Tcl Command Descriptions

The following table provides a listing of all valid Radiant software message control-related Tcl command options and describes option functionality.

Table 33: Message Control Tcl Commands

Command	Description
msg_clean	Clean promoted, disabled and suppressed message(s) from the current project.
msg_clean_setting	Clean the setting of promoted, disabled and suppressed message(s) from the current project.
msg_demote	Reset message promotion type from the current project.
msg_disable	Disable the message from the current project.
msg_list_status	List all promoted, disabled or suppressed messages from the current project.

Table 33: Message Control Tcl Commands

Command	Description
msg_load	Load the setting of all promoted, disabled and suppressed messages from the file.
msg_promote	Set message promotion type from the current project.
msg_save	Save the setting of all promoted, disabled and suppressed messages to the file.
msg_suppress	Suppress the message from the current project.

msg_clean

Commands to clean promoted, disabled and suppressed message(s) from the current project.

Syntax

```
msg_clean <message Id>|-all
```

Arguments

- **<message Id>**: This option specifies implementation name (optional).
- **-all**: Clean the setting of all promoted, disabled and suppressed messages.

Example:

```
msg_clean 1026001
msg_clean -all
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_clean_setting

Commands to clean the setting of promoted, disabled and suppressed message(s) from the current project.

Syntax

```
msg_clean_setting <message Id>|-warning|-info|-all
```

Arguments

- **<message Id>**: This option specifies the message id.
- **-warning**: Clean the setting for warning messages.
- **-info**: Clean the setting for info messages.

- **-all**: Clean the setting for all the messages.

Example:

```
msg_clean_setting 1026001
msg_clean_setting -warning
msg_clean_setting -info
msg_clean_setting -all
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_demote

Commands to reset message promotion type from the current project.

Syntax

```
msg_demote <message Id>
```

Arguments

- **<message Id>**: This option specifies the message id.

Example:

```
msg_demote 1026001
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_disable

Commands to disable the message from the current project.

Syntax

```
msg_disable <message Id>|-warning|-info
```

Arguments

- **<message Id>**: This option specifies the message id.
- **-warning**: Disable all warning messages.
- **-info**: Disable all info messages..

Example:

```
msg_disable 1026001
msg_disable -warning
msg_disable -info
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_list_status

Commands to list all promoted, disabled or suppressed messages from the current project.

Syntax

```
msg_list_status [-promoted|-disabled|-suppressed]
```

Arguments

- **-promoted**: List all promoted messages (optional).
- **-disabled**: List all disabled messages (optional).
- **-suppressed**: List all suppressed messages (optional).

Example:

```
msg_list_status
msg_list_status -promoted

msg_list_status -disabled
msg_list_status -suppressed
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_load

Commands to load the setting of all promoted, disabled and suppressed messages from the file, default is promote.xml.

Syntax

```
msg_load [<message xml or pmt file>]
```

Arguments

- **<message xml or pmt file>**: This option specifies the message xml or pmt file (optional).

Example:

```
msg_load
msg_load promote.xml
msg_load promote.pmt
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_promote

Commands to set message promotion type from the current project. Use msg_save command after promoting a message to ensure the message promotion settings are saved.

Syntax

```
msg_promote <message Id> [-type 0|1|2|3]
```

Arguments

- <message Id>: This option specifies the message id.
- -type 0|1|2|3: This option the message promotion type, 0 is error, 1 is critical, 2 is warning, and 3 is info (optional).

Example:

```
msg_promote 1026001
msg_promote 1026001 -type 2
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_save

Commands to save the setting of all promoted, disabled and suppressed messages to the file, default is promote.xml.

Syntax

```
msg_save [<message xml or pmt file>]
```

Arguments

- **<message xml or pmt file>**: This option specifies the message xml or pmt file (optional).

Example:

```
msg_save
msg_save promote.xml
msg_save promote.pmt
```

See Also

- "Message Control Tcl Commands" on page 1692

msg_suppress

Commands to suppress the message from the current project, set maximum display count.

Syntax

```
msg_suppress <message Id> -cnt 0|<count>
```

Arguments

- **<message Id>**: This option specifies the message id.
- **-cnt 0|<count>**: This option specifies maximum display count, 0 is disabled, and 'count' means how many times the message will only display.

Example:

```
msg_suppress 1026001 -cnt 10
```

See Also

- “Message Control Tcl Commands” on page 1692

Radiantc TCL Commands

Radiant has two design flow modes: GUI and pure TCL mode. In pure TCL mode, you can use all TCL commands. However, in GUI mode (including the TCL section within it), Radiant oversees the project process and status without retaining the design in memory. Consequently, any TCL commands that require access to design data are unavailable. The list of these commands is provided below.

- Bitstream Generation Tcl Commands
- Design Tcl Commands
- Device Tcl Commands
- Technology Mapping Tcl Commands
- Placement Tcl Commands
- Routing Tcl Commands
- Timing Analysis Tcl Commands

Bitstream Generation Tcl Commands

The **bit** command is used for bitstream generation.

Bitstream Generation Tcl Command Descriptions

The following table provides a listing of all valid Radiant software bitstream generation-related Tcl command options and describes option functionality.

Table 34: Bitstream Generation Tcl Commands

Command	Description
bit_generate	Generate a bitstream file.
bit_list_option	List bit generation options.
bit_set_option	Set bit generation options.

bit_generate

Generate bitstream file.

Syntax

```
bit_generate [-w] <filename>
```

Arguments

- **-w**: Overwrite existing bitstream file if <filename> exists.
- **<filename>**: Specify output bitstream file. Do not include '.bit' as it is appended (required).

Example:

```
bit_generate -w mydesign
```

See Also

- “Bitstream Generation Tcl Commands” on page 1697

bit_list_option

List bitgen options.

Syntax

```
bit_list_option
```

Example:

```
bit_list_option
```

See Also

- “Bitstream Generation Tcl Commands” on page 1697

bit_set_option

Set bitgen options.

Syntax

```
bit_set_option {<key> <value> ...}
```

Example:

```
bit_list_option
bit_set_option {enable_early_io_wakeup true ip_evaluation
false}
```

See Also

- "Bitstream Generation Tcl Commands" on page 1697

Design Tcl Commands

The **des** commands are used to read or write the Radiant unified design database (.udb) file and list and report netlist objects such as instance, net, or port. You can get these commands by using **help {des_*}**.

The following table provides a listing of all valid Radiant software design-related Tcl command options and describes option functionality.

Table 35: Design Tcl Commands

Command	Description
des_get_util_data	Return the required data provided/used in device/ design.
des_list_attribute	List design attributes.
des_list_instance	List instance.
des_list_net	List all nets.
des_list_port	List port.
des_list_reference_udb	List reference udb(s) for increment flow.
des_read_udb	Read in a .udb file.
des_reload_milestone	Reload milestone result.
des_report	Report design info.
des_report_flow_status	Report design flow status.
des_report_instance	Report instance information.
des_report_net	Report a net.
des_report_operating_condition	Report operating condition.

Table 35: Design Tcl Commands

des_report_port	Report port information.
des_set_attribute	Set design attributes.
des_set_operating_condition	Change operating condition.
des_set_reference_udb	Set reference udb(s) for increment flow.
des_write_udb	Write the design into a .udb file.

des_get_util_data

Returns the required resource provided/used in device/design.

Syntax

```
des_get_util_data [-device][-lut][-reg][-slice][-dpram]
[-ripplelogic][-ebr][-dsp][-pio][-pll][-gsr][-osc]
```

Arguments

- **-device**: Returns the resource data provided in the device (optional).
- **-lut**: Returns number of LUTs (optional).
- **-reg**: Returns number of REGs (optional).
- **-slice**: Return number of Slices (optional).
- **-dpram**: Returns number of DPRAMs (optional).
- **-ripplelogic**: Return number of slices used in carry chains (optional).
- **-ebr**: Return number of EBRs (optional).
- **-dsp**: Return number of DSPs (optional).
- **-pio**: Return number of PIOs (optional).
- **-pll**: Return number of PLLs (optional).
- **-gsr**: Return number of GSRs (optional).
- **-osc**: Return number of oscilators (optional).

Example:

```
des_get_util_data -pio
Return # of PIO instances used in the design.

des_get_util_data -device -pio
Return the # of PIOs provides in the device.
```

If the return value is -1, it means that the selected data type is not available in the device, or the data type has not been supported.

See Also

- "Design Tcl Commands" on page 1699

des_list_attribute

Reports design attributes.

Syntax

```
des_list_attribute
```

Example:

```
des_list_attribute
```

See Also

- "Design Tcl Commands" on page 1699

des_list_instance

Commands to list specified instances.

Syntax

```
des_list_instance [<instances>]
```

Arguments

- **<instances>**: Only lists instances that match the pattern specified in <instances> (optional). If no pattern is specified, all instances will be listed.

Example:

```
des_list_instance *CWDATA_c*
```

See Also

- "Design Tcl Commands" on page 1699

des_list_net

Commands to list specified nets.

Syntax

```
des_list_net [-clock] [-fanout <n>] [-lsort <method>]
[-lsize <n>] [<nets>]
```

Arguments

- **-clock**: Lists clock nets only (optional).
- **-fanout <n>**: Lists nets with # of fanouts >= n (optional).
- **-lsort <method>**: Choose sorting method: 0 - id; 1 - all pins; 2 - clk pins; 3 - global control pins; 4 - normal pins (optional). Defaults to sorting by id.
- **-lsize <size>**: Limits the number of listed nets to size. Defaults to showing all nets (optional).
- **<nets>**: Only lists net names that match the pattern specified in <nets> (optional).

Example:

```
des_list_net -fanout 10 -lsort 2 -lsize 10 *D0*
```

This command lists the first 10 nets that start with D0, have a fanout of at least 10, sorted by clk pins first.

See Also

- "Design Tcl Commands" on page 1699

des_list_port

Commands to list specified ports.

Syntax

```
des_list_port [<ports>]
```

Arguments

- **<ports>**: Only lists instances that match the pattern specified in <ports> (optional). If no pattern is specified, all ports will be listed.

Example:

```
des_list_port *CWDATA_c*
```

See Also

- "Design Tcl Commands" on page 1699

des_list_reference_udb

Lists reference udb(s) for incremental flow, if there's any.

Syntax

```
des_list_reference_udb
```

Example:

```
des_list_reference_udb
```

See Also

- "Design Tcl Commands" on page 1699

des_read_udb

Commands to read database from udb file.

Syntax

```
des_read_udb [-logview <logical udb file>]
[-phyview <physical udb file>] [<udb file>]
```

Arguments

- **-logview <logical udb file>**: Specifies the logical udb file name (optional).
- **-phyview <physical udb file>**: Specifies the physical udb file name (optional).
- **<udb file>**: Specifies the udb file name (optional).

  -logview, -phyview, and <udb file> options are mutually exclusive.

  If either -logview, -phyview, or both are specified, you cannot specify a udb file.

Example:

```
des_read_udb -logview logic.udb -phyview phy.udb
des_read_udb post_par.udb
```

See Also

- "Design Tcl Commands" on page 1699

des_reload_milestone

Reloads milestone result.

Syntax

```
des_reload_milestone [-force] [-inc] [-postsyn|-postmap|
-postplc|-postrte|-postpar] <udb_name>
```

Arguments

- **-force**: Forcibly opens the milestone (optional).
- **-inc**: Loads the milestone result incrementally (optional).
- **-postsyn**: Opens a post-synthesis milestone (optional).
- -**postmap**: Opens a post-map milestone (optional).
- **-postplc**: Opens a post-placement milestone (optional).
- **-postrte**: Opens a post-routing milestone (optional).
- **-postpar**: Opens a post-par milestone (optional).
- **<udb_name>**: Milestone udb to open (required).

Example:

```
des_reload_milestone -postsyn -inc mydb_syn.udb
```

See Also

- “Design Tcl Commands” on page 1699

des_report

Reports design information.

Syntax

```
des_report
```

Example:

```
des_report
```

See Also

- “Design Tcl Commands” on page 1699

des_report_flow_status

Reports design flow status.

Syntax

```
des_report_flow_status
```

Example:

```
des_report_flow_status
```

See Also

- “Design Tcl Commands” on page 1699

des_report_instance

Commands to report specified instances.

Syntax

```
des_report_instance [<instances>]
```

Arguments

- **<instances>**: Only reports instances that match the pattern specified in <instances> (optional).

Example:

```
des_report_instance *CWDATA_c*
```

See Also

- “Design Tcl Commands” on page 1699

des_report_net

Commands to report specified nets.

Syntax

```
des_list_net [-file <filename>] [<nets>]
```

Arguments

- **-file <filename>**: Outputs results into <filename> (optional).
- **<nets>**: Only reports nets that match the pattern specified in <nets> (optional).

Example:

```
des_report_net -file mynet.txt *CWDATA_c*
```

See Also

- “Design Tcl Commands” on page 1699

des_report_operating_condition

Reports operating condition information.

Syntax

```
des_report_operating_condition
```

Example:

```
des_report_operating_condition
```

See Also

- "Design Tcl Commands" on page 1699

des_report_port

Commands to report specified ports.

Syntax

```
des_report_port [<port>]
```

Arguments

- **<ports>**: Only reports ports that match the pattern specified in <ports> (optional). If no pattern is specified, all ports are reported.

Example:

```
des_report_port *CWDATA_c*
```

See Also

- "Design Tcl Commands" on page 1699

des_set_attribute

Changes design attributes.

Syntax

```
des_set_attribute [-impl_dir <name>]
```

Arguments

- **-impl_dir <name>**: Sets implementation directory to <name> (optional). Defaults to current directory otherwise.

Example:

```
des_set_attribute -impl_dir impl_1
```

See Also

- "Design Tcl Commands" on page 1699

des_set_operating_condition

Changes operating condition of device.

Syntax

```
des_set_operating_condition [-temperature <temp>]
[-speed_grade <speed_grade>]
```

Arguments

- **-temperature <temp>**: Set device temperature to <temp> degrees Celsius (optional). Defaults to -1.
- **-speed_grade <speed_grade>**: Sets device speed grade to specified setting <speed_grade> (optional)

Example:

```
des_set_operating_condition -temp 100 -speed_grade 9_High-
Performance_1.0V
```

Sets temperature to 100 Celcius with High Performance speed grade.

See Also

- "Design Tcl Commands" on page 1699

des_set_reference_udb

Sets reference udb(s) for incremental flow.

Syntax

```
des_set_reference_udb [-postsyn <postsyn_udb>]
[-postmap <postmap_udb>][-postpar <postpar_udb>]
[-pdc <pdc_file>] [-clean]
```

Arguments

- -postsyn <postsyn_udb>: Set post-synthesis reference udb file to <postsyn_udb> (optional).
- -postmap <postmap_udb>: Set post-map reference udb file to <postmap_udb> (optional).
- -postpar <postpar_udb>: Set post-par reference udb file to <postpar_udb> (optional).

- -pdc <pdc_file>: Set reference pdc file to <pdc_file> (optional).
- -clean: Reset all reference udb settings (optional).

Example:

```
des_set_reference_udb -postsyn mydb_syn.udb -postpar
mydb_par.udb -pdc mypdc.pdc
Set post-synthesis reference udb to mydb_syn.udb,
post-par udb to mydb_par.udb and pdc file to mypdc.pdc.

des_set_reference_udb -clean
Removes any reference udb settings.
```

See Also

- "Design Tcl Commands" on page 1699

des_write_udb

Commands to write udb database file.

Syntax

```
des_write_udb [-w] [-logview] [-phyview] [<udb file>]
```

Arguments

- **-w**: Overwrite existing file (optional).
- **-logview:** Output logical view (optional).
- **-phyview**: Output physical view (optional).
- **<udb file>**: This option specifies the udb file name (optional).

  -logview, -phyview, and <udb file> options are mutually exclusive.

Example:

```
des_write_udb -logview logic.udb -phyview phy.udb
des_write_udb -w post_par.udb
```

See Also

- "Design Tcl Commands" on page 1699

Device Tcl Commands

The **dev** commands are used re used to access or report device information. You can get these commands by using **help {dev_*}**.

The following table provides a listing of all valid Radiant software device-related Tcl command options and describes option functionality.

Table 36: Device Tcl Commands

Command	Description
dev_report	Report the device info.

dev_report

Report device information.

Syntax

```
dev_report
```

Arguments

Example

```
dev_report
```

See Also

- "Device Tcl Commands" on page 1708

Technology Mapping Tcl Commands

The **map** commands are used to map or transfer a design from a logical view to a physical view.

Radiant Software Technology Mapping Tcl Command Descriptions

The following table provides a listing of all valid Radiant software mapping-related Tcl command options and describes option functionality.

Table 37: Technology Mapping Tcl Commands

Command	Description
map_list_option	List map options.
map_run	Run map tool.
map_set_option	Set map options.

map_list_option

List map options.

Syntax
```
map_list_option
```

Arguments

Example
```
map_list_option
```

See Also
- "Technology Mapping Tcl Commands" on page 1709

map_run
Run the map tool.

Syntax
```
map_run
```

Arguments

Example
```
map_run
```

See Also
- "Technology Mapping Tcl Commands" on page 1709

map_set_option
Set the map options.

Syntax
```
map_set_option {<key> <value> ...}
```

Arguments

Example
```
map_list_option
map_set_option {infer_gsr true pdc_file myfile.pdc}
```

See Also
- "Technology Mapping Tcl Commands" on page 1709

Place & Route Tcl Commands

The **par** commands are used to run the place & route process. You can run place & route results from project flow to a non-project flow by using the **par_run** command.

Radiant Software Place & Route Tcl Command Descriptions

The following table provides a listing of all valid Radiant software place & route-related Tcl command options and describes option functionality.

Table 38: Place & Route Tcl Command

Command	Description
par_list_option	List placement routing options.
par_run	Run placement routing.
par_set_option	Set placement routing options.

par_list_option

Lists placement and routing options.

Syntax

```
par_list_option
```

Example:

```
par_set_option
```

See also

- “Place & Route Tcl Commands” on page 1711

par_run

Runs placement and routing.

Syntax

```
par_run
```

Example:

```
par_run
```

See also

- “Place & Route Tcl Commands” on page 1711

par_set_option

Sets placement and routing options.

Syntax

```
par_set_option {<key> <value> ...}
```

Example:

```
par_set_option {number_of_host_machine_cores 8
path_based_placement on}
```

See also

- "Place & Route Tcl Commands" on page 1711

Placement Tcl Commands

The **plc** commands are used for automatic/interactive placement.

Placement Tcl Command Descriptions

The following table provides a listing of all valid Radiant software placement-related Tcl command options and describes option functionality.

Table 39: Placement Tcl Commands

Command	Description
plc_list_option	List placement options.
plc_output_instance_location	Output instance locations in LDC format.
plc_report	Report placement information.
plc_run	Run placement tool.
plc_set_option	Set placement options.

plc_list_option

Lists placement options.

Syntax

```
plc_list_option
```

Example:

```
plc_list_option
```

See Also

- "Placement Tcl Commands" on page 1712

plc_output_instance_location

Outputs instance locations in LDC format.

Syntax

```
plc_output_instance_location [-all] [-pio] [-dsp] [-ebr] [-
slice] [-all_constraint] -file <filename> [<pattern>]
```

Arguments

- **-all**: Selects all instance types to output (optional).
- **-pio**: Selects PIO instances to output (optional).
- **-dsp**: Select DSP instances to output (optional).
- **-ebr**: Select EBR instances to output (optional).
- **-slice**: Select slices to output (optional).
- **-all_constraint**: Select all constraint instances to output (optional).
- **-file <filename>**: Specify output file to write to (required).
- **-<pattern>**: Only output instances that match <pattern> (optional).

Example:

```
plc_output_instance_location -dsp -file newfile.ldc CW*
```

This example outputs DSP instances that start with CW into a new constraint file called **newfile.ldc**.

See Also

- "Placement Tcl Commands" on page 1712

plc_report

Reports placement information such as sites, comps, bounding box, and estimated wiring segments.

Syntax

```
plc_report
```

Example:

```
plc_report
```

See Also

- "Placement Tcl Commands" on page 1712

plc_run

Runs placement tool.

Syntax

```
plc_run
```

Example:

```
plc_run
```

See Also

- "Placement Tcl Commands" on page 1712

plc_set_option

Sets placement options.

Syntax

```
plc_set_option {<key> <value> ...}
```

Example:

```
plc_list_option
plc_set_option {disable_timing_driven true path_based_placement
on}
```

See Also

- "Placement Tcl Commands" on page 1712

Routing Tcl Commands

The **rte** commands are used for automatic/interactive routing.

Routing Tcl Command Descriptions

The following table provides a listing of all valid Radiant software routing-related Tcl command options and describes option functionality.

Table 40: Routing Tcl Commands

Command	Description
rte_list_option	List routing options.
rte_report	Report routing results.
rte_run	Run the routing tool.
rte_set_option	Set routing options.

rte_list_option

List routing options.

Syntax

```
rte_list_option
```

Example:

```
rte_list_option
```

See Also

- “Routing Tcl Commands” on page 1714

rte_report

Report routing results.

Syntax

```
rte_report
```

Arguments

- **-net**: Report the nets (optional).

Example:

```
rte_report
```

See Also

- “Routing Tcl Commands” on page 1714

rte_run

Run the routing tool.

Syntax

```
rte_run
```

Example:

```
rte_run
```

See Also

- "Routing Tcl Commands" on page 1714

rte_set_option

Set routing options.

Syntax

```
rte_set_option {<key value> ... }
```

Example:

```
rte_list_option
rte_set_option {disable_timing_driven true}
```

See Also

- "Routing Tcl Commands" on page 1714

Timing Analysis Tcl Commands

The **sta** commands are used to run timing analysis interactively, with two major categories, namely **sta_get** and **sta_report**. The sta_get command returns the value, which can be used in other sta commands or printed out to the standard output. The sta_report command sends the information to the standard output.

You can get these commands by using **help {sta_*}**.

The following table provides a listing of all valid Radiant software timing analysis-related Tcl command options and describes option functionality.

Table 41: Timing Analysis Tcl Commands

Command	Description
sta_get_clocks	Get a list of clocks that satisfy the name.
sta_get_coverage	Get the timing coverage for connections.

Table 41: Timing Analysis Tcl Commands

sta_get_info	Get information about a clock or a terminal (port/pin).
sta_get_paths	Collect detailed paths for query.
sta_get_slack	Get the total negative, worst or terminal slack for the design.
sta_get_terms	Get a list of ports and pins that satisfy the name.
sta_path_info	Extract data from a given path.
sta_report_clocks	Report on a clock or clocks.
sta_report_constraints	List timing constraints.
sta_report_timing	Report detailed or summary timing.
sta_report_unconstrained	List unconstrained ports, pins, or nets.
sta_set_option	Change the style of the timing report.

Note

If you do not change the default PAR Timing Strategy (Speed Grade for Hold Analysis) and the Junction Temperature value in Device Constraint Editor, there may be differences between the Radiant GUI and the TCL results.

In the Radiant GUI Timing Analysis report, the Hold Analysis of STA uses Speed Grade "m" and Junction Temperature of "0" degrees. In the TCL timing analysis, on the other hand, the Hold Analysis of STA uses the speed grade of the device and the default junction temperature of 85 degrees.

sta_get_clocks

Commands to list requested clocks.

Syntax

```
sta_get_clocks [<name>]
```

Arguments

- **<name>:** Only list clock names that match the pattern specified in <name> (Required)

Example

```
sta_get_clocks clk *
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_get_coverage

Commands to list specified instances.

Syntax

```
sta_get_coverage
```

Example

```
sta_get_coverage
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_get_info

Commands to get information about clocks or terminals.

Syntax

```
sta_get_info [-clock <clock name>] [-waveform]
[-period][-duty_cycle] [-terminal <terminal name>]
[-isclockpin][-isinput] [-isoutput] [-net] [-cell]
[-termClocks]
```

Arguments

- **-clock <clock name>:** Name of clock whose info is requested <clock name> (optional)
- **-waveform:** requests waveform of clock (optional)
- **-period:** period of clock will be returned (optional)
- **-duty_cycle:** duty cycle will be returned (optional)
- **-terminal <terminal name>:** Only report on terminal that match the pattern specified in <terminal name> (optional)
- **-isclockpin:** Returns true if the terminal is a clock pin (optional)
- **-isinput:** Returns true if the terminal is an input (optional)
- **-isoutput:** Returns true if the terminal is an output (optional)
- **-net:** Net connected to terminal (optional)
- **-cell:** Parent cell of terminal (optional)
- **-termClocks:** User defined clocks that arrive at the terminal
  Option -clock is mutually exclusive with options -terminal, -isclockpin, -isinput, -isoutput, -net, -cell, and -termClocks. Option -terminal is mutually exclusive with options -waveform, -period, and -duty_cycle. (optional)

Example

```
sta_get_info -clock clk1 -duty_cycle
sta_get_info -terminal abc/LSR -net
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_get_paths

Change operating condition of device.

Syntax

```
sta_get_paths [-n <number>] [-endpoints]
[-from <ports/pins>]
[-from_clock <clocks>] [to <ports/pins>]
[-to_clock <clocks>] [-hold] [-summary]
```

Arguments

- **-n <number>:** The total number of paths reported is up to <number> paths (optional)
- **-endpoints:** Report the endpoints only (optional)
- **-from <ports/pins>:** The from pins of the report are given by <ports/pins> (optional)
- **-from_clock <clocks>:** The launch clock of the report are given by <ports/pins> (optional)
- **-to <ports/pins>:** The to pins of the report are given by <ports/pins> (optional)
- **-to_clock <clocks>:** The latch clocks of the report are given by <ports/pins> (optional)
- **-hold:** Report hold timing paths (optional)
- **-summary:** Report setup and hold timing errors and total negative slack at the current speed grade (optional)

Example

```
sta_get_paths -n 5 -from abcd/Q0
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_get_slack

Command to report slack of design or terminal.

Syntax

```
sta_get_slack [-tns] [-worst] [-hold] [-terminal <terminal>]
```

Arguments

- **-tns:** Get the total negative slack of the design (optional)
- **-worst:** Get the worst slack of the design (optional)
- **-hold:** Get the hold slack (optional)
- **-terminal <terminal>:** Get the slack of the terminal specified by <terminal> (optional)
  Option -terminal is mutually exclusive with the options -tns and -worst

Example

```
sta_get_slack -terminal abcd/e -hold
sta_get_slack -tns
```

See Also

- “Timing Analysis Tcl Commands” on page 1716

sta_get_terms

Command to get a list of terms.

Syntax

```
sta_get_terms [-ports | -pins] [-name <name of ports or pins>]
```

Arguments

- **-ports:** The name represents ports (optional)
- **-pins:** The name represents pins (optional)
- **-name:** name or ports or pins. Option '-ports and -pins are mutually exclusive. (optional)

Example

```
sta_get_terms -ports -name *
sta_get_terms -name */LSR
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_path_info

Get information on a given path.

Syntax

```
sta_path_info [-path <path>] [-slack] [-constraint]
[-skew] [-launch_clock] [-latch_clock] [-levels]
```

Arguments

- **-path:** A path object obtained with sta_get_paths. (required)
- **-slack:** Return the slack of the path. (optional)
- **-constraint:** Returns the value of the constraint that must be met for the given path (optional).
- **-skew:** Returns the clock skew (optional).
- **-launch_clock:** Return the clock launching the data at the start point (optional).
- **-latch_clock:** Return the clock capturing the data at the endpoint (optional).
- **-levels:** Return number of logic levels (optional).

Example

```
sta_path_info
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_report_clocks

Report design clocks.

Syntax

```
sta_report_clocks [-clocks <clocks>] [-file <file name>] [-app]
```

Arguments

- **-clocks <clocks>:** Report clocks that are indicated by <clocks> (optional)

- **-file <file name>:** Name of file that report is to be redirected to (optional)
- **-app:** Append report to end of file (optional)

Example

```
sta_report_clocks -clocks cl* -file abc -append
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_report_constraints

Command to list timing constraints.

Syntax

```
sta_report_constraints [-file <file name>] [-app]
```

Arguments

- **-file <file name>:** Name of file that report is to be redirected to <file name> (optional)
- **-app:** Append the report to an existing file (optional)

Example

```
sta_report_constraints
sta_report_constraints -file abc -app
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_report_timing

Change operating condition of device.

Syntax

```
sta_report_timing [-n <number>] [-endpoints]
[-from <ports/pins>] [-from_clock <clocks>]
[to <ports/pins>] [-to_clock <clocks>] [-hold]
[-summary] [-file <file name>] [-app]
```

Arguments

- **-n <number>:** The total number of paths reported is up to <number> paths (optional).
- **-endpoints:** Report the endpoints only (optional)
- **-from <ports/pins>:** The from pins of the report are given by <ports/pins> (optional)
- **-from_clock <clocks>:** The launch clock of the report are given by <ports/pins> (optional)
- **-to <ports/pins>:** The pins of the report are given by <ports/pins> (optional)
- **-to_clock <clocks>:** The latch clocks of the report are given by <ports/pins> (optional)
- **-hold:** Report hold timing paths (optional)
- **-summary:** Report setup and hold timing errors and total negative slack at the current speed grade (optional)
- **-file <file name>:** Redirect the report to the file <file name> (optional)
- **-app:** Append report to the end of the file (optional)

Example

```
sta_report_timing -n 5 -from abcd/Q0
```

See Also

- "Timing Analysis Tcl Commands" on page 1716

sta_report_unconstrained

Command to report unconstrained terminals.

Syntax

```
sta_report_unconstrained [-n <number>] [-start] [-end]
[-io] [-clock_nets] [-file <file name>] [-app]
```

Arguments

- **-n <number>:** Report up to the number specified in <number> (optional)
- **-start:** Report unconstrained timing start point pins (optional)
- **-end:** Report unconstrained timing endpoint pins (optional)
- **-io:** Report unconstrained IOs (optional)
- **-clock_nets:** Report clock nets without a clock definition (optional)
- **-file:** File to which the report is redirected (optional)
- **-app:** Append report at the end of the file (optional)

Example

```
sta_report_unconstrained -file abc -start -app
sta_report_unconstrained
```

See Also

- “Timing Analysis Tcl Commands” on page 1716

sta_set_option

Commands to read database from udb file.

Syntax

```
sta_set_option [-name <logical or physical>] [-worst_delay]
```

Arguments

- **-name <logical or physical>:** This option controls the name used in reports (optional)
- **-worst_delay:** This option activates mixed temperature delay data (optional)

Example

```
sta_set_option -name physical
sta_set_option -worst_delay
```

See Also

- “Timing Analysis Tcl Commands” on page 1716

Revision History

The following table gives the revision history for this document.

Date	Version	Description
12/20/2024	2024.2	Updates for Radiant 2024.2 software.
09/23/2024	2024.1.1	Updates for Radiant 2024.1.1 software.
06/28/2024	2024.1	Updates for Radiant 2024.1 software.
03/29/2024	2023.2.1	Updates for Radiant 2023.2.1 software.
11/27/2023	2023.2	Updates for Radiant 2023.2 software.
06/26/2023	2023.1	Updates for Radiant 2023.1 software.
03/13/2023	2022.1.1	Updates for Radiant 2022.1.1 software.
11/28/2022	2022.1	Updates for Radiant 2022.1 software.
06/23/2022	3.2	Updates for Radiant 3.2 software.
05/23/2022	3.1.1 MachXO5 -NX Device Update	Updates for Lattice Radiant 3.1.1 MachXO5-NX Device Update.
03/17/2022	3.1.1	Updates for Radiant 3.1.1 software.
12/13/2021	3.1	Updates for Radiant 3.1 software.
06/14/2021	3.0	Updates for Radiant 3.0 software.
03/31/2021	2.2.1	Updates for Radiant 2.2.1 software.
11/05/2020	2.2	Updates for Radiant 2.2 software.
06/12/2020	2.1	Updates for Radiant 2.1 software.
02/25/2020	2.0 SP1	Updates for Radiant 2.0 SP1 software.

Date	Version	Description
12/04/2019	2.0	Updates for Radiant 2.0 software.
03/29/2019	1.1	Updates for Radiant 1.1 software.
02/13/2018	1.0	Initial Release.