Reveal User Guide

Lattice Semiconductor Corporation
5555 NE Moore Court
Hillsboro, OR 97124
(503) 268-8000

February 2010

Copyright

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, CleanClock, E^2CMOS, Extreme Performance, FlashBAK, FlexiClock, flexiFlash, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeECP3, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MACO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, ProcessorPM, PURESPEED, Reveal, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TransFR, UltraMOS, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP, Bringing the Best Together, and More of the Best are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

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 SEMICONDUCTOR CORPORATION (LSC) 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 LSC 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.

LSC may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. LSC makes no commitment to update this documentation. LSC 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. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

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 Introduction 1

Chapter 2 Reveal Inserter 3

About Reveal Inserter **3**
- Reveal On-Chip Debug Design Flow **4**
- Inputs **6**
- Outputs **6**
- Limitations **6**

Getting Started **8**
- Before Starting Reveal Inserter **8**
- Starting Reveal Inserter **8**
- Creating a New Reveal Inserter Project **9**
- Opening an Existing Reveal Inserter Project **10**

Managing the Cores in a Project **10**
- Adding a Core **10**
- Renaming a Core **11**
- Removing a Core **11**

Viewing Signals in the Design Hierarchy Pane **11**

Searching for Signals **12**

Setting Up the Trace Signals **13**
- Selecting the Debug Logic Core **13**
- Selecting the Trace Signals **14**
- Viewing Trace Signals and Buses **14**
- Grouping Trace Signals into a Bus **15**
- Ungrouping Trace Signals in a Bus **15**
- Removing Signals and Buses from the Trace Data Pane **15**
- Renaming a Bus **15**
- Setting Required Sample Parameters **16**
- Setting Sample Options **16**

Setting Up the Trigger Signals **18**
- Triggering **19**

Adding Trigger Units **26**
Renaming Trigger Units **27**
Setting Up Trigger Units **27**
Removing Trigger Units **29**
Adding Trigger Expressions **29**
Renaming Trigger Expressions **30**
Setting Up Trigger Expressions **30**
Removing Trigger Expressions **33**
Using Tokens **33**
Checking the Debug Logic Settings **39**
Saving a Project **40**
Inserting the Debug Logic Cores **40**
Removing Debug Logic from the Design **41**
Closing a Project **42**
Deleting a Project **42**
Exiting Reveal Inserter **42**
Building the Database **42**
Mapping, Placing, and Routing the Design **43**
Generating a Bitstream or JEDEC File **43**
Connecting to the Evaluation Board **43**
Downloading Design onto the Device **43**
Performing Logic Analysis with Reveal Logic Analyzer **43**
User Interface Descriptions **44**
Trace Signal Setup Tab **47**
Trigger Signal Setup Tab **50**
Dialog Boxes **55**
Toolbar **62**
Menus **63**

Chapter 3 Reveal Logic Analyzer 69

About Reveal Logic Analyzer **70**
Reveal On-Chip Debug Design Flow **70**
Inputs **71**
Outputs **72**
Inserting the Debug Logic **73**
Building the Database **73**
Mapping, Placing, and Routing the Design **73**
Generating a Bitstream or JEDEC File **73**
Connecting to the Evaluation Board **74**
Downloading a Design onto the Device **74**
Starting Reveal Logic Analyzer **76**
Starting Reveal Logic Analyzer from Project Navigator **76**
Starting Reveal Logic Analyzer as a Stand-Alone Tool **77**
Running a Demonstration Design **77**
Creating a New Reveal Logic Analyzer Project **77**
Programming and Debugging Devices in a Daisy Chain **79**
Opening an Existing Reveal Logic Analyzer Project **80**
Opening an Existing Project **80**

Opening a Recently Opened Project **80**
Selecting a Reveal Logic Analyzer Core Window **81**
Setting Up the Trace Signals **81**
Grouping Trace Signals into a Bus **81**
Ungrouping Trace Signals from a Bus **82**
Setting the Trace Bus Radix **82**
Renaming Trace Buses **84**
Setting Bus Bit Order **84**
Adding Time Stamps to Trace Samples **85**
Making Trace Signals Invisible **85**
Making Trace Signals Visible **86**
Locating a Trace Signal **86**
Setting Trace Signal Colors **86**
Moving Signals in the Waveform View Tab **86**
Setting Up the Trigger Signals **87**
Renaming Trigger Units **87**
Setting Up Trigger Units **87**
Renaming Trigger Expressions **88**
Setting Up Trigger Expressions **89**
Setting Trigger Options **90**
Performing Logic Analysis **90**
Data Capture with Sample Enable **91**
Common Error Conditions **91**
Performing Logic Analysis on Multiple Devices **92**
Stopping a Logic Analysis **93**
Using Manual Triggering **93**
Viewing Waveforms **94**
Viewing Logic Analyses **94**
Viewing Cascading or Tiled Logic Analyses **94**
Splitting a Waveform **94**
Reversing a Waveform Split **94**
Adjusting the Waveform Display **95**
Panning **95**
Zooming In and Out **95**
Specifying the Clock Frequency **96**
Placing, Moving, and Locating Markers **96**
Finding Signals **98**
Finding Data **99**
Finding Data in a Signal **99**
Finding Data in a Bus **99**
Printing the Waveform **100**
Exporting Waveform Data **100**
Saving a Project **101**
Closing a Project **102**
Exiting Reveal Logic Analyzer **102**
User Interface Descriptions **102**
Trigger Signal Setup Tab **103**
Waveform View Tab **109**
Dialog Boxes **110**

Toolbar **115**
Status Bar **118**
Menus **119**

Chapter 4 Reveal On-Chip Debug Tutorial 125

Introduction **125**
Learning Objectives **125**
Time to Complete This Tutorial **126**
System Requirements **126**
Accessing Online Help **126**
About the Tutorial Data Flow **126**

Task 1: Creating an RTL-Type ispLEVER Project **127**

Task 2: Generating and Adding the Reveal Core **133**

Setting Up the Trigger Signals **136**
Setting Up the Trigger Units **136**
Setting Up the Trigger Expressions **138**
Inserting the Debug Logic **138**
Building the Design Files **140**
Mapping, Placing, and Routing the Design **141**
Generating a Bitstream **141**

Task 3: Programming the Evaluation Board **142**
Connecting to the Evaluation Board **142**
Downloading the Program **143**

Task 4: Performing Logic Analysis **146**
Creating a New Reveal Logic Analyzer Project **146**
Running the Logic Analyzer **148**
Using Single Trigger Capture Mode **150**

Summary **150**

Glossary **151**

Recommended Reference Materials **152**

1

Introduction

The Reveal software included in ispLEVER is a next-generation FPGA on-chip debug tool. It offers several key usability benefits including the following features:

- Integration with the Project Navigator design flow
- One-button operation to insert debug logic into the design
- Simple flow for modifying the original design or the debug configuration
- Advanced triggering capabilities for more flexible dynamic triggers
- Improved logic analyzer waveform usability

This user guide contains all the necessary information for getting started with the Reveal software. It contains three primary sections: the Reveal Inserter, the Reveal Logic Analyzer, and a tutorial.

The Reveal Inserter section contains information on how to add debug information to your design. It also contains detailed information on how to use and set up the triggering architecture used in Reveal. The triggering architecture, which is based on trigger units and trigger expressions, has some differences from other systems but offers increased capabilities and flexibility for an internal logic analyzer.

The Reveal Logic Analyzer section contains information on how to use and connect the software to the design running on the target hardware.

Finally, the tutorial section takes a very simple design through several steps to show you how to use the basics of the Reveal software flow.

2

Reveal Inserter

Reveal Inserter enables you to select which design signals to use for debug tracing or triggering, then generate a core on the basis of these signals and their use. After generating the required core, it generates a modified design with the necessary debug connections and links it to the signals. Reveal Inserter supports VHDL, Verilog, and EDIF flows for debug insertion. Once the design has been modified for debug, it is mapped, placed, and routed with the normal design flow in ispLEVER.

After you generate the bitstream or JEDEC file, Reveal Logic Analyzer helps you debug your FPGA circuitry by giving you access to internal nodes inside the device so you can observe their behavior. It enables you to set and change various values and combinations of trigger signals. Once the specified trigger condition is reached, the data values of the trace signals are saved in the trace buffer. After the data is captured, it is transferred from the FPGA through the JTAG ports to the PC.

About Reveal Inserter

This section introduces some of the key features of Reveal Inserter: the devices that it supports, the steps in its design flow, its inputs, its outputs, and its limitations.

You can use Reveal Inserter with all FPGAs and with MachXO devices of 1200 or more LUTs.

You can run Reveal Inserter on both the Windows and Linux operating systems on the PC. For information on installing ispLEVER, see the *ispLEVER Installation Notice for Windows* or the *ispLEVER Installation Notice for Linux* for the current release.

Reveal On-Chip Debug Design Flow

On Windows, you can use either of the following two flows to generate and integrate debug logic cores into your design. Both flows enable you to specify parameters to customize the Reveal cores. On Linux, you must use the post-synthesis EDIF flow.

- In the pre-synthesis RTL design flow, trace and trigger signal selection is based on a Verilog or VHDL design before synthesis. The Reveal software is designed to enable you to pick the desired signals for tracing and triggering directly from the design before having to generate any debug cores. Once you select them, the Reveal software generates the necessary debug cores and provides information to the implementation flow in ispLEVER on how to connect the debug cores to the design. During implementation, the Build Database step generates a modified design with the necessary debug connections for the debug cores and assembles the design. As part of this process, temporary RTL files are generated in the background but are not part of the user flow. Any design modifications should only be made to the original RTL source code. Any debug changes should only be made in the Reveal Inserter software.
- In the post-synthesis EDIF flow, trace and trigger signal selection is based on an EDIF netlist read by Reveal Inserter software. Once the design is read into Reveal Inserter, the debug flow for EDIF is the same as the flow for an RTL design. The Reveal software generates the necessary debug cores and provides information to the implementation flow in ispLEVER on how to connect the debug cores to the design. During implementation, the Build Database step generates a modified design with the necessary debug connections for the debug cores and assembles the design. Any design modifications should only be made to the original EDIF netlist (or source used to generate the EDIF netlist). Any debug changes should only be made in the Reveal Inserter software.

Note

Interactive synthesis is not compatible with the Reveal debugging flow. When you use Reveal, the interactive synthesis option is not available.

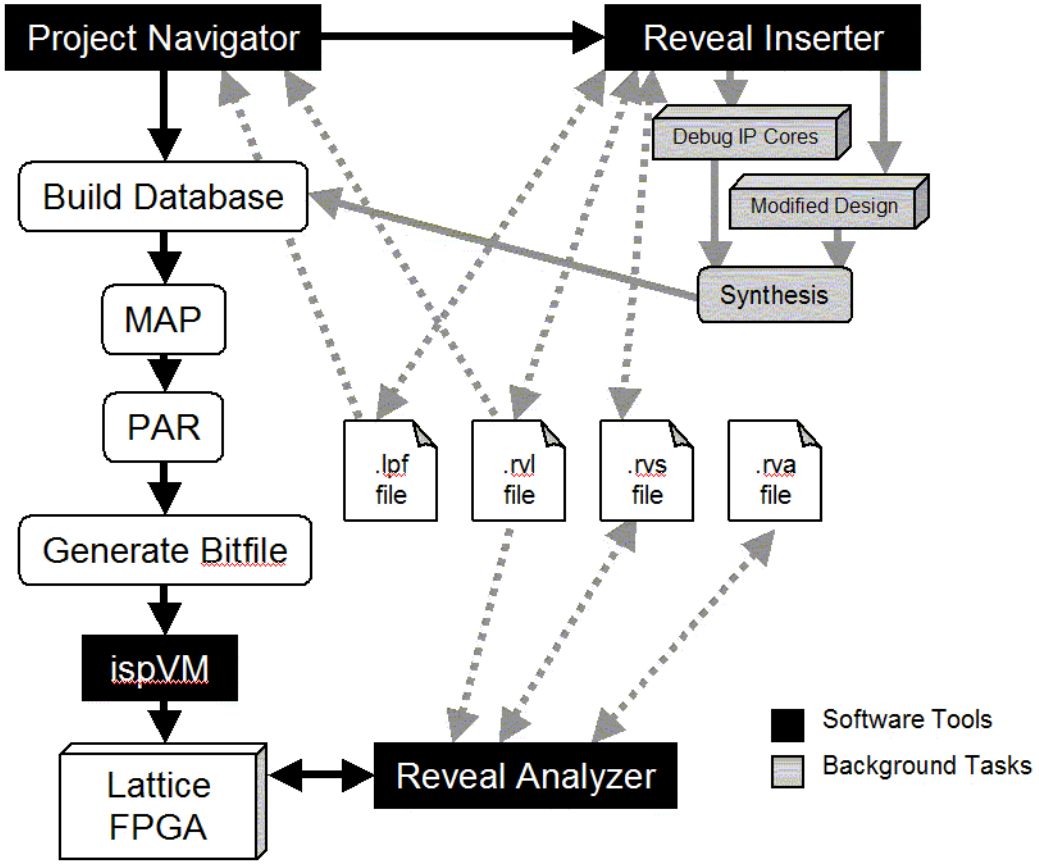
The Reveal Inserter design flow is shown in the following figure.

To generate and insert debug logic cores, follow these general steps:

1. Start Reveal Inserter.
2. Create a new Reveal Inserter project or open an existing Reveal Inserter project.
3. Add new cores to the project, if needed.
4. For each core, set up the trace signals in the Trace Signal Setup tab.
5. For each core, set up the trigger signals in the Trigger Signal Setup tab.
6. Insert the debug logic.

   This process generates and synthesizes the necessary debug logic.

Figure 1: Reveal Inserter Design Flow



The generated .rvl is automatically imported into ispLEVER if you enabled the Import Reveal file to ispLEVER Project option in the Insert Debug to Design dialog box.

7. Build the database in the ispLEVER Project Navigator.

   This process creates two or more .ngo files.

   When Reveal is used for your design, the Build Database process performs two extra steps:

   - It builds a version of the design that contains the necessary connections for the debug logic to the signals that you are tracing or triggering.
   - It adds logic for communicating with the JTAG pins for Reveal Logic Analyzer. Newer software and IP from Lattice Semiconductor support a mechanism to allow multiple elements to share the JTAG pins. However, if the design was built with earlier software or IP and contains a JTAG primitive, this logic conflicts with the Reveal flow and results in an error in the Build Database process.

8. Map, place, and route the design.
9. Generate the bitstream data or JEDEC file.

   If you want to perform logic analysis with Reveal Logic Analyzer, continue with these steps:

10. Set up the cable connection with ispVM.
11. Download the design onto the device.
12. Start Reveal Logic Analyzer and perform logic analysis with it.

Step 13 is documented in the Reveal Logic Analyzer online Help.

Inputs

The inputs to Reveal Inserter in the RTL flow are the following:

- A VHDL or Verilog file (or multiple VHDL or Verilog files)
- A preference (.lpf) file, if it already exists

The inputs to Reveal Inserter in the EDIF flow are the following:

- EDIF file or files
- A preference (.lpf) file, if it already exists

Outputs

Reveal Inserter generates the following files in the RTL flow:

- Reveal Inserter project (.rvl) file, which contains the signal connections for each core and some settings for the debugging logic, such as maximum sequence depth and maximum event counter. The information in this file is statically set in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.
- Reveal Inserter settings (.rvs) file, which contains settings that can be dynamically changed without regenerating the debug logic. This information includes trigger units, comparison types, values, and trigger expressions. The information in this file is dynamically set in either Reveal Logic Analyzer or in both Reveal Logic Analyzer and Reveal Inserter.
- Reveal Inserter parameter (.rvp) file, which contains information needed for debug logic generation.

Reveal Inserter also modifies the logical preference (.lpf) file:

- Timing settings are modified for the debug logic.
- RVL_ALIAS preferences are added. RVL_ALIAS maps clock names generated by Reveal Inserter to the clock names used in the original design. So you do not need to change your preferences that refer to those clock signals.

Limitations

Reveal Inserter has the following limitations in the current release.

Unsupported VHDL and Verilog Features in Reveal Inserter

The following features that are valid in the VHDL and Verilog languages are not supported in Reveal Inserter when you use the RTL flow:

- Array types of two dimensions or more are not shown in the port or node section.

- Undeclared wires attached to instantiated component instances are not shown in the hierarchical design tree. You must declare these wires explicitly if you want to trace or trigger with them.
- Variables used in conditional statements like if-then-else statements are not available for tracing and triggering.
- Variables used in selection statements like the case statement are not available for tracing and triggering.
- If function calls are used in the array declaration, the actual size of the array is unknown to Reveal Inserter.
- Entity and architecture of the same design cannot be in different files.
- In Verilog, you must explicitly declare variables at the very beginning of a module body to avoid obtaining different results from various synthesis tools.
- In VHDL, you must declare synthesis attributes within an entity, not within an architecture, to avoid obtaining different results from various synthesis tools.

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.

Constants

You can use signals that are set to constant values as trace or trigger signals. Although it is not recommended that you use these signals as trace or trigger signals, Reveal Inserter will not issue an error message if you do.

You cannot use a signal for the sample clock that has been set to a constant value. The value of the sample clock must be able to change. However, Reveal Inserter will not generate an error message if you use a signal set to a constant value for the sample clock. It does not know whether a signal has been set to a constant value before synthesis and can only avoid displaying those that were originally declared as constants in the hierarchy tree.

Dangling or Unconnected Nets

You cannot use dangling or unconnected nets for tracing or triggering. These nets are unavailable in the hierarchy tree, so you cannot select them. You also cannot use them for sample clocks or sample enables. They are removed or attached to GND or VCC during synthesis. Reveal Inserter issues an error message if you attempt to use them for tracing or triggering.

Getting Started

After you create a project in Project Navigator, you can start Reveal Inserter and create a Reveal project. Or open an existing Reveal project for modification.

Before Starting Reveal Inserter

Reveal Inserter assumes that your design has been implemented before you insert the debug logic. You should ensure that you have a design that can be synthesized before you try to insert the debug logic. When you invoke Reveal Inserter, it verifies that the design has been synthesized and issues a warning message if it has not.

Starting Reveal Inserter

Reveal Inserter is started from Project Navigator. Open the desired Project Navigator project to have access to the tools.

To start Reveal Inserter:

1. Do one of the following:
   - In Project Navigator, choose **Tools > Reveal Inserter.**
   - In the Project Navigator toolbar, click the Reveal Inserter  button.
2. If the Welcome to Reveal dialog box opens, click **OK**. To avoid seeing it again, select **Do not show again**.
3. If the Reveal Inserter Startup Wizard opens, choose a Reveal project, if there is a choice. Click **OK**.

When Reveal Inserter opens, it shows the Reveal project or, if there are no existing projects, Reveal Inserter creates one.

When Reveal Inserter opens a design, it must parse and statically elaborate it. In some cases, code opened in Project Navigator and successfully synthesized with some synthesis tools may be flagged as having an error when Reveal Inserter tries to open the design. In these cases, Reveal Inserter is interpreting the HDL code more strictly than the chosen synthesis tool. It is likely that the code would not synthesize with a different synthesis tool or would have other compliance issues.

To correct this problem, see the reveal_error.log file in the project directory. This file contains information and error messages that enable you to see any problems found in the design.

Creating a New Reveal Inserter Project

After you create a project in Project Navigator, you can create a project in Reveal Inserter.

Note

If you are going to use Reveal Inserter on the Linux platform, you must install a stand-alone synthesis tool, such as Synopsys® Synplify Pro®, before you create a Reveal project.

After you install the stand-alone synthesis, you set the SYNPLIFY and LM_LICENSE_FILE environment variables. The SYNPLIFY variable is set to the installation path of the synthesis tool. The LM_LICENSE_FILE variable is set to the path of the license file of the synthesis tool.

In addition, your Linux system must meet the minimum system requirements outlined in the *ispLEVER Installation Guide for Linux*.

Only the EDIF flow is supported for Linux.

To create a new Reveal Inserter project:

1. Choose **File > New Project** in Reveal Inserter window or click  in the toolbar.

   The Create Project dialog box appears.

   In the Project Name box, the <*project_name*>.rvl path and file name should already appear.

2. If it does not, click ... to browse to the project, and type <*project_name*>.rvl.

3. In the Signal Type box, select the type of flow from the drop-down menu. The following flows are available on Windows:

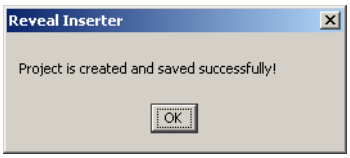
   - Pre-Synthesis RTL is available if you have a VHDL, Verilog, Schematic/Verilog, or Schematic/VHDL project for Windows.
   - Post-Synthesis EDIF is available only for an EDIF or a mixed HDL project.

   Only the post-synthesis EDIF flow is available on Linux.

4. Click **OK**.

5. In the following confirmation dialog box, click **OK**.

Figure 2: Project Creation Confirmation Dialog Box



If you want to use a different set of trace and trigger signals after you create the first Reveal Inserter project, you can create a new Reveal Inserter project for the same Project Navigator project.

Opening an Existing Reveal Inserter Project

To open an existing project, you must have available a Reveal Inserter project (.rvl) file and a Reveal Inserter settings (.rvs) file from a previous Reveal Inserter session.

To open an existing Reveal Inserter project:

1. Choose **File > Open Project** in the Reveal Inserter window or click  in the toolbar.
2. In the Select File dialog box, browse to the .rvl file of interest, enter its name in the File Name box, and click **Open**.

If the desired .rvl file has been recently opened, you can open it directly from the File menu.

To open a recently opened .rvl file directly from the File menu:

- Choose **File > Recent Project > *<filename>*** from the list of the four most recently opened files near the bottom of the File menu.

Managing the Cores in a Project

Each Reveal Inserter project can include up to 15 debug logic cores. Each core has its own settings for the debug logic, such as trace signals, trigger signals, sample clock, sample enable, and trigger output signal. These settings are called a dataset. In many cases, a single core 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 multple cores, one for each clock region where the clock is used as the sample clock for the core.

Adding a Core

When you open a new project, Reveal Inserter automatically adds the first debug logic core to the first dataset and gives it a name of <*top_module*>_LA<*number*>, where *top_module* is the name of the top module in the Reveal Inserter project, and *number* is a sequential number. For example, the third core added to the “counter” project would be named counter_LA3. The core name is case insensitive—for example, “core_LA0” is the same as “core_la0.”

All Reveal cores are listed in the Debug Datasets pane in the Reveal Inserter window.

To add a core to a dataset:

- Choose **Datasets > Add New Core** or right-click on **Datasets** in the Debug Datasets pane and choose **Add New Core**.

Reveal Inserter now creates a new core. The added cores are displayed in the Debug Datasets pane.

Renaming a Core

You can rename a debug logic core if you want to change its initial name.

To rename a core or cores in a project:

1. Highlight the name of the core in the Debug Datasets pane, and choose **Datasets > Rename Core**, or right-click on the name of the core and choose **Rename** from the pop-up menu.
2. In the Input dialog box, type the new name of the core in the Rename Core box.
3. Click **OK** in the dialog box.

During the renaming process, Reveal Inserter verifies that:

- The core name begins with a letter and consists of letters, numbers, and underscores (_).
- The core name is not the same as that of any other core.
- The core name is not the same as that of any module or instance in the design.

Removing a Core

You can also remove a debug logic core.

To remove a core or cores from a project:

- Select the core in the Debug Datasets pane, and choose **Datasets > Remove Core**, or right-click on the name of the core and choose **Remove** from the pop-up menu.

Viewing Signals in the Design Hierarchy Pane

In the Design Hierarchy pane of Reveal Inserter window, you can display the hierarchy of the whole design, including the ports and nodes in the top module and submodules, so that you can choose the signals to use for data tracing and triggering.

From the Design Hierarchy pane, you can drag a signal to the upper half of the Trace Signal Setup tab to set it as a trace signal or drag it to the lower half of the tab to set it as a sample clock signal or a sample enable signal.

In the Design Hierarchy pane, the names of trace, trigger, and control signals are in bold font if they are currently being used. If you select a signal in the hierarchy, the Signal Information tab displays information about how it is used.

To view all signals in the design hierarchy:

- Choose **View > Expand All Design Hierarchy**, or right-click on the design name in the Design Hierarchy pane and choose **Expand All** from the pop-up menu.

To view the buses, ports, top-level signals, and top level of the hierarchy:

- Choose **View > Collapse All Design Hierarchy**, or right-click on the design name in the Design Hierarchy pane and choose **Collapse All** from the pop-up menu.

You can also view signals and buses in the Trace Data pane of the Trace Signal Setup tab.

Searching for Signals

You can search for a signal or signals and set the selected signals as trace signals, trigger unit signals, sample clock signals, or sample enable signals. You can search for signal names or patterns of characters.

To search for a signal:

1. In the Signal Search box beneath the Design Hierarchy pane, enter the name of the signal or pattern to find. You can set a filter by using the case-insensitive alphanumeric characters and wildcards shown in the following table.
2. Click **OK**.

   If Reveal Inserter finds only one signal, it highlights it in the Design Hierarchy pane.

   If Reveal Inserter finds multiple signals, it opens the Search Signals dialog box to list all the signals found.
3. If you are searching for multiple signals, select the desired signals in the Search Signals dialog box, and click **OK**.
   - Shift-click to select contiguous signals.
   - Control-click to select non-contiguous signals.

   The selected signals are now highlighted in the Design Hierarchy pane.

From the Design Hierarchy pane, you can drag signals to the Trace Data pane, the Sample Clock box, and the Sample Enable box in the Trace Signal Setup tab. You can also drag signals to the Signals (MSB:LSB) box in the Trigger Unit section of the Trigger Signal Setup tab.

Although the buses are displayed as "*busname*[*n*:*m*]" in the Design Hierarchy pane, Reveal Inserter ignores the string after the bus name when it searches for buses. For example, if the design contains a bus called a[0:2], you can search for it by a pattern such as "a" or "a*," but you cannot use a pattern such as "a[*."

If a bus is named xyz, a search for xyz highlights the entire bus. A search for xyz* brings up the Search Signals dialog box and all the individual signals in the xyz bus.

The following wildcards are supported in searches:

Wildcard Character	Characters to Replace	Example
?	Any single character	?a? where “a” is the middle character in a three-character string
*	Any sequence of characters	*a* where the string contains the “a” character
^	The beginning of a string	^a^ where the string begins with “a”
$	The end of a string	*a$ where the string ends with “a”
X{*n*}	*X* exactly *n* times	a{2}* where the string begins with “aa”
X{*n*,}	*X* at least *n* times	a{2,}* where the string begins with “aa,” “aaa,” “aa...a”
X{n, m}	*X* at least *n* but not more than *m* times	a{2, 4}* where the string begins with “aa,” “aaa,” or “aaaa”
[abc]	“a,” “b,” or “c”	[abc]* where the string begins with “a,” “b,” or “c”
[^abc]	Any character except “a,” “b,” or “c”	[^abc]* where the string does not begin with “a,” “b,” or “c”
[a-d]*	Any character in the range of “a” through “d”	[a-d]* where the string begins with any character in the range of “a” through “d”

Setting Up the Trace Signals

The first step in performing a logic analysis is to specify how the data from the trace bus will be captured. Use the Trace Signal Setup tab in the Reveal Inserter window to choose the signals from which to collect sample data in the selected core.

Selecting the Debug Logic Core

Before you configure the trace signals, select the debug logic core to configure in the Debug Datasets pane.

Selecting the Trace Signals

You can use either of two methods to select trace signals: dragging and dropping or using a search engine to find them. You can select up to 512 trace signals in each core.

To select trace signals by dragging and dropping:

- Select the desired signals in the Design Hierarchy pane and drag them to the Trace Data pane in the Trace Signal Setup tab.

To select trace signals by using a search engine:

1. In the Signal Search box beneath the Design Hierarchy pane, enter the name or pattern of the signal to find. You can set a filter by using case-insensitive alphanumeric characters and wildcards. See “Searching for Signals” on page 12 for information about the wildcards that you can use.
2. Click **OK**.

   If Reveal Inserter finds only one signal, it highlights it in the Design Hierarchy pane.

   If Reveal Inserter finds multiple signals, it opens the Search Signals dialog box to list all the signals found.
3. If you are searching for multiple signals, select the desired signals in the Search Signals dialog box, and click **OK**.

   The signals are now selected in the Design Hierarchy pane.
4. Drag them to Trace Data pane in the Trace Signal Setup tab.

Viewing Trace Signals and Buses

In the Trace Data pane in the Trace Signal Setup tab, you can display the signals in buses or remove them from view.

To display all the signals in all the buses:

- Right-click in the Trace Data pane, and choose **Expand All** from the pop-up menu.

To hide all the signals in all the buses:

- Right-click in the Trace Data pane, and choose **Collapse All** from the pop-up menu.

To display all the signals in an individual bus:

- Right-click on the bus and choose **Expand** from the pop-up menu.

To hide all the signals in an individual bus:

- Right-click on the bus and choose **Collapse** from the pop-up menu.

Grouping Trace Signals into a Bus

You can group trace signals, buses, or both into a bus.

To group signals or buses into a bus or to add signals or buses to a bus:

1. In the Trace Data pane of the Trace Signal Setup tab, select the signals, buses, or both to be grouped.
2. Choose **Trace > Group Signals/Buses**, or right-click and choose **Group** from the pop-up menu.

Ungrouping Trace Signals in a Bus

You can ungroup the signals or buses in a bus.

To ungroup the signals, buses, or both in a bus:

1. In the Trace Data pane in the Trace Signal Setup tab, select the signals, buses, or both to be ungrouped from the bus.
2. Choose **Trace > UnGroup Signals/Buses**, or right-click and choose **UnGroup** from the pop-up menu.

Removing Signals and Buses from the Trace Data Pane

You can remove signals from the Trace Data pane in the Trace Signal Setup tab.

To remove a signal or a bus from the Trace Data pane:

1. In the Trace Signal Setup tab, select the signals to be removed from the Trace Data pane.
2. Choose **Trace > Remove Signals/Buses**, or right-click and choose **Remove** from the pop-up menu. You can also press the Delete key.

Renaming a Bus

You can rename a bus.

To rename a bus:

1. In the Trace Data pane of the Trace Signal Setup tab, select the bus.
2. Choose **Trace > Rename Bus**, or right-click and choose **Rename** from the pop-up menu.
3. In the Input dialog box, enter the new name of the bus.
4. Click **OK**.

Setting Required Sample Parameters

For each core, you must set the certain sample parameters for the trace signals.

To set the required sample parameters:

1. In the **Sample Clock** box in the Trace Signal Setup tab, type the name of the clock signal or drag the clock signal from the design tree shown in the Design Hierarchy pane.

   Note

   On the board, make sure that the minimum sample clock frequency is at least three times (preferably four times) that of the JTAG clock. If the sample clock speed is too slow, you will be unable to complete logic analysis with Reveal Logic Analyzer.

2. In the **Buffer Depth** box, specify the size of the trace memory buffer.

   This parameter defines the number of trace bus samples that a core can capture. It can be set to a minimum of 16 or to powers of 2 from 16 to 65536. The buffer size is determined by the type of hardware.

3. In the **Implementation** box, specify how the debug logic is to be implemented in the FPGA. You can choose one of the following:

   - EBR – Implements the debugging logic as embedded block RAM (EBR). This setting is the default.
   - DistRAM – Implements the debugging logic as distributed RAM.

4. In the **Data Capture Mode** box, select Single Trigger Capture or Multiple Trigger Capture. Single Trigger Capture is enabled by default.

5. If you choose Multiple Trigger Capture, you must also set the **Minimum samples per trigger** option, which specifies the minimum number of data samples to collect per trigger. The minimum is either 8 or 1/256 of the total buffer depth, whichever is greater. The maximum number of samples depends on the design.

Setting Sample Options

In addition to the required parameters, you can set options for the data sample.

Using a Sample Enable

A sample enable is an optional signal used to capture data only when the sample enable is active, either high or low. If you do not specify a sample enable signal, trace data is collected on every sample clock after the trigger.

You may want to use a sample enable in cases where you need to capture a lot of data, but the data is only important during certain times, not whenever the sample clock is running. In these cases, the sample enable is a “gate” that allows you to turn the capturing of data on and off. An example is a design that contains many different sections, but some sections only work during certain clock phases. You typically use a master clock and generate different

signals for the phases. You could use one of the phases as the sample enable.

To set the sample enable:

- In the **Sample Enable** checkbox, indicate whether a sample enable signal is to be used. If you want to use a sample enable:
  a. Select the checkbox to indicate that a sample enable signal will be used. The checkbox is deselected by default.
  b. Enter the name of the sample enable signal in the box beneath the checkbox, or drag the signal from the Design Hierarchy pane.
  c. In the box to the right of the signal name box, select either **Active High**, which means that trace data is captured when the sample enable is high and the sample clock occurs, or **Active Low**, which means that trace data is captured when the sample enable is low and the sample clock occurs. Active High is the default.

Each sample shown in the trace buffer is only captured when the sample enable is active and there is a sample clock. Data samples can be discontiguous, unlike those in a normal data capture.

Additionally, it is possible that the actual trigger condition may occur when the sample enable is not active. This causes two changes from a normal data capture:

- The actual data values for the trigger condition may not be visible, because the data cannot be captured when the sample enable is inactive.
- Reveal Logic Analyzer cannot accurately calculate the trigger point, since the trigger point may have occurred when the sample enable is inactive. Normally a trigger point is shown as a single marker on the clock on which the trigger occurred. If a sample enable is used, a trigger region that spans 5 clock cycles is shown instead. Reveal Logic Analyzer can guarantee that the trigger occurred in this region, but it cannot determine during which clock cycle the trigger occurred.

The sample enable is a very useful feature, but it takes more understanding than a normal data capture.

Adding Trigger Signals to Trace Signals

You can add trigger signals to the trace signals so that the data from the trigger signals is included in the trace data. Tracing trigger signals increases the amount of logic used by the trace buffer.

To add the trigger signals to the trace signals:

- Select the **Include trigger signals in trace data** option. This option is turned off by default.

Adding Time Stamps to Trace Samples

In Reveal Inserter, you can optionally specify a sample clock count value to be stored with each trace sample to indicate the sample count clock value at which the sample was captured. This count is extra data (bits) captured into the trace buffer that increase the trace buffer's width. This time stamp enables you to see how many sample clock intervals have elapsed between data captures when you use a sample enable. It is useful in some cases when it is necessary to know if you captured the right data. A time stamp is also useful when you try to synchronize data between multiple cores, off-chip data, or both. For example, if you trigger two cores at the same time, you can use the time stamps on the trace samples to calculate how the data between the cores compares.

To add time stamps to the trace samples:

1. Select the **Timestamp** box in the Trace Signal Setup tab.
2. In the drop-down menu in the Bits box next to the Timestamp box, select the amount of trace memory storage needed by the time stamp, in bits.

   The number of bits for the timestamp is the number of bits in the maximum count of the timestamp. But each bit is equivalent to adding another signal to be traced, so the amount of trace memory needed is therefore much larger. The minimum number of bits that appears in the drop-down menu is obtained by multiplying the value in the Buffer Depth box by 2 and converting the result to an exponential value. For example, if the value in the Buffer Depth box is 256, the minimum number of bits in the Bits drop-down menu is calculated as follows:

   256 X 2 = 512

   $512 = 2^9$

   So the minimum number of bits available in the Bits menu in this case is 9.

   The maximum number of bits available in the Bits menu is always 63.

Setting Up the Trigger Signals

The Reveal software has some similarities to and some differences from external logic analyzers. An external logic analyzer typically offers up to a few dozen signals or channels and megabits worth of capture data depth. Internal or embedded logic analyzers have different constraints. An internal logic analyzer can offer thousands of signal connections, since no extra pins are required to connect to the signal. But the resources inside an FPGA force a limitation on the amount of data that can be captured, typically constrained to several thousand bits. This difference drives different requirements. An internal logic analyzer requires the ability to accurately pinpoint the desired event in order to capture a smaller amount of data around that precise event. The capabilities in the Reveal software are designed specifically for the triggering requirements of an internal logic analyzer.

Triggering

With the Reveal software, it is easy to set up simple triggering conditions, as well as extremely complex triggers. 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.

Some of Reveal's triggering features are static and some are dynamic. Static features can only be changed in Reveal Inserter and require the design to be re-implemented by synthesis, map, place, and route. Although you can set most of the dynamic features in Reveal Inserter, you can change all dynamic features when Reveal Logic Analyzer is running, and you do not have to re-implement the design.

Trigger Units

The trigger unit is used to compare a number of input 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.

You can change the signals in a trigger unit only in Reveal Inserter. Changing the input signals requires the design to be re-implemented.

You can specify up to 16 trigger units for each debug core. A common technique is to group associated input signals into a trigger unit. For example, you might use a trigger unit for the address bus in a design, another for the data bus, and another for the control signals.

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 has the edge.

For example, consider a trigger unit defined as cout[3:0], rising edge, 1110. This trigger unit will be true only when cout[3], cout[2], or cout[1] have a rising edge. What happens on cout[0] does not matter.

- 0000 > 1110

  True because cout[3], cout[2], and cout[1] rose.

- 0000 > 1111

  True for the same reason. It does not matter whether cout[0] rises or not.

- 0000 > 0100

  True because a rising edge on any of the specified bits is sufficient.

- 1000 > 1000

  False because cout[3] did not rise. It just stayed high.

Serial Compare The “serial compare” operator checks for a series of values on a single signal. For example, if a trigger unit’s specified value is 1011, the “serial compare” operator looks for a 1 on the first clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only after those four conditions are met in those four clock cycles is the trigger unit true.

Serial compare is available only when a single signal is listed in the trigger unit’s signal list. The radix is automatically binary.

You can only set the serial compare operator in Reveal Inserter. You cannot change it or select it in Reveal Analyzer as you can the other operators.

Trigger Expressions

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 Logic 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 Logic 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 21 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 21 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 21 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 21 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 Logic Analyzer.

Here is an example of a trigger expression with a THEN operator:

```
TU1 THEN TU2
```

This trigger expression is interpreted as “wait for TU1 to be true, then wait for TU2 to be true.”

If the same example were written with a NEXT operator:

```
TU1 NEXT TU2
```

it is interpreted as “wait for TU1 to be true, then wait *one clock cycle* for TU2 to be true.” If TU2 is not true in the next clock cycle, the sequence fails and starts over, waiting for TU1 again.

The next trigger expression:

```
TU1 THEN TU2 #2
```

is interpreted as “wait for TU1 to be true, then wait for TU2 to be true for two sample clocks.” TU2 may be true on consecutive or non-consecutive sample clocks and still meet this condition.

The following statement:

```
TU1 ##5 THEN TU2
```

means that TU1 must occur for five consecutive sample clocks before TU2 is evaluated. If there are any extra delays between any of the five occurrences of TU1, the sequence fails and starts over.

The next expression:

```
(TU1 & TU2)#2 THEN TU3
```

means “wait for the second occurrence of TU1 and TU2 to be true, then wait for TU3.”

The last expression:

```
TU1 THEN (1)#200
```

means “wait for TU1 to be true, then wait for 200 sample clocks.” This expression is useful if you know that an event occurs a certain time after a condition.

You can only use one count (# or ##) operator per sequence. For example, the following statement is not valid, because it uses two counts in a sequence:

```
TU1 #5 & TU2 #2
```

Multiple count values are allowed for a single trigger expression, but only one per sequence. For two count operators to be valid in a trigger expression, the expression must contain at least one THEN or NEXT operator, as in the following example:

```
(TU1 & TU2) #5 THEN TU2 #2
```

This expression means “wait for TU1 and TU2 to be true for five sample clocks, then wait for TU2 to be true for two sample clocks.”

Also, the count operator must be applied to the entire sequence expression, as indicated by parentheses in the expression just given. The following is not allowed:

```
TU1 #5 & TU2 THEN TU2 #2
```

The count (#) operator cannot be used as part of a sequence following a NEXT operator. A consecutive count (##) operator may be used after a NEXT operator. The following is not allowed:

```
TU1 NEXT TU2 #2
```

The count (# or ##) operators can only be used in one of two areas:

- Immediately after a trigger unit or parentheses(). However, if the trigger unit is combined with another trigger unit without parentheses, a # cannot be used.
- After a closing parenthesis.

Precedence The symbols used in trigger expression syntax take the following precedence:

- Because it inserts a sequence, the THEN and NEXT operators always take the highest precedence in trigger expressions.
- Between THEN or NEXT statements, the order is defined by parentheses that you insert. For example, the following trigger expression:

  ```
  TU1 & (TU2|TU3)
  ```

  means “wait for either TU1 and TU2 or TU1 and TU3 to be true.”

  If you do not place any parentheses in the trigger expression, precedence is left to right until a THEN or NEXT statement is reached.

  For example, the following trigger expression:

  ```
  TU1 & TU2|TU3
  ```

  is interpreted as “wait for TU1 & TU2 to be true or wait for TU3 to be true.”
- The precedence of the ^ operator is same as that of the & operator and the | operator.
- The logic negation operator (!) has a higher precedence than the ^ operator, & operator, or | operator, for example:

  ```
  !TU1 & TU2
  ```

  means “not TU1 and TU2.”
- The # and ## operators have the same precedence as the ^ operator, & operator, or | operator. However, they can only be used in one of two areas:
  - Immediately after a trigger unit or trigger units combined in parentheses. However, if the trigger unit is combined with another trigger unit without parentheses, a # or ## operator cannot be used.

    Here is an example of correct syntax using the count (#) operator:

    ```
    TU1 #2 THEN TU3
    ```

    This statement means “wait for TU1 to be true for two sample clocks, then wait for TU3.”

    However, the following syntax is incorrect, because the count operator is applied to multiple trigger units combined without parentheses:

    ```
    TU1 & TU2#2 THEN TU3
    ```
  - After a closing parenthesis. Use parentheses to combine multiple trigger units and then apply a count, as in the following example:

    ```
    (TU1 & TU2)#2 THEN TU3
    ```

    This statement means “wait for the combination of TU1 and TU2 to be true for two sample clocks, then wait for TU3.”

Following is a series of examples that demonstrate the flexibility of trigger expressions.

Example 1: Simplest Trigger Expression Following is the simplest trigger expression:

TU1

This trigger expression is true, causing a trigger to occur when the TU1 trigger unit is matched. The value and operator for the trigger unit is defined in the trigger unit, not in the trigger expression.

Example 2: Combinatorial Trigger Expression An example of a combinatorial trigger expression is as follows:

TU1 & TU2 | TU3

This trigger expression is true when (TU1 and TU2) or TU3 are matched. If no precedence ordering is specified, the order is left to right.

Example 3: Combinatorial Trigger Expression with Precedence Ordering In the following example of a combinatorial trigger expression, precedence makes a difference:

TU1 & (TU2 | TU3)

This trigger expression gives different results than the previous one. In this case, the trigger expression is true if (TU1 and TU2) or (TU1 and TU3) are matched.

Example 4: Simple Sequential Trigger Expression Following is an example of a simple sequential trigger expression:

TU1 THEN TU2

This trigger expression looks for a match of TU1, then waits for a match on TU2 a minimum of one sample clock later. Since this expression uses a THEN statement, it is considered to have multiple sequences. The first sequence is “TU1,” since it must be matched first. The second sequence is “TU2,” because it is only checked for a match after the first sequence has been found. The “sequence depth” is therefore 2.

The sequence depth is an important concept to understand for trigger expressions. Since the debug logic is inserted into the design, logic must be used to support the required sequence depth. Matching the depth to the entered expression can be used to minimize the logic. However, if you try to define a trigger expression that has a greater sequence depth than is available in the FPGA, an error will prevent the trigger expression from running. The dynamic capabilities of the trigger expression can therefore be limited. To allow more flexibility, you can specify the maximum sequence depth when you set up the debug logic in Reveal Inserter. You can reserve more room for the trigger expression than is required for the trigger expression currently entered. If you specify multiple trigger expressions, each trigger expression can have its own maximum sequence depth.

Example 5: Mixed Combinatorial and Sequential Trigger Expression
Here is an example showing how you can mix combinatorial and sequential elements in a trigger expression:

TU1 & TU2 THEN TU3 THEN TU4 | TU5

This trigger expression only generates a trigger if (TU1 AND TU2) match, then TU3 matches, then (TU4 or TU5) match. You can set precedence for any sequence, but not across sequences. The expression (TU1 & TU2) | TU3 THEN TU4 is correct. The expression (TU1 & TU2 THEN TU3) | TU4 is invalid and is not allowed.

Example 6: Sequential Trigger Expression with Sequence Counts The next trigger expression shows two new features, the sequence count and a true operator to count sample clocks:

(TU1 & TU2)#2 THEN TU3 THEN TU4#5 THEN (1)#200

This trigger expression means wait for (TU1 and TU2) to be true two times, then wait for TU3 to be true, then wait for TU4 to be true five times, then wait 200 sample clocks. The count (# followed by number) operator can only be applied to a whole sequence, not part of a sequence. When the count operator is used in a sequence, the count may or may not be contiguous. The always true operator (1) can be used to wait or delay for a number of contiguous sample clocks. It is useful if you knew that an event that you wanted to capture occurred a certain time after a condition but you did not know the state of the trigger signals at that time.

However, there is a limitation on the maximum size of the counter. This depends on how much hardware is reserved for the sequence counter. When you define a trigger expression, the Max Event Counter setting in the Trigger Expression section of Reveal Inserter and Reveal Logic Analyzer specifies how large a count value is allowed in the trigger expression. Each trigger expression can have a unique Max Event Counter setting.

Trigger Expression and Trigger Unit Naming Conventions

You can rename trigger units and trigger expressions. The names can be a mixture of lower-case or upper-case letters, underscores, and digits from 0 through 9. The first character must be either an underscore or a letter. The names can be any length.

Final 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.

Multiple Core Support for Triggers

Each core in a design has its own triggers and traced signals. When a design is “triggered” (meaning that the triggers are enabled and active so that the debug logic is running and looking for the trigger condition), you can specify which individual cores are enabled.

Simultaneous Trigger Activation in Multiple Cores Since each core is typically a different clock region, you can specify whether the triggering is enabled for each core when triggering for the device is activated.

For each core, you can indicate if the trigger or triggers should be enabled when triggering for the device is activated.

Cross-Triggering To support triggers based on multiple sample clocks, cross-triggering is available between different debug cores.

- Reveal provides an optional trigger-out signal in the triggering section for every core.
- If a design has multiple cores, trigger-out signals from other cores are listed as an available signal for triggers in another core. To use a trigger-out signal as an input to another core, you must specify it as a “net” or “both” type. The I/O type is only used for connecting the trigger-out to an external I/O. Trigger-out signals are listed in the Trigger Out Signals from Reveal Core(s) window pane at the bottom left of the Reveal Inserter window.

Adding Trigger Units

You can add trigger units only in Reveal Inserter. You cannot add them in Reveal Logic Analyzer. You can change some of the trigger conditions defined in Reveal Inserter in Reveal Logic Analyzer during hardware debugging.

All trigger units are automatically available for use in all trigger expressions defined.

You can add up to 16 trigger units per core. Each trigger unit consists of the following:

- Trigger unit name (label)
- Signals in the trigger unit
- Comparison function
- Radix of the trigger unit value
- Value of the trigger unit

To add a trigger unit:

1. If you want the buses in the new trigger units that you will add to have a certain radix by default, set that radix in the **Trigger Radix** box in the Trigger Unit section of the Trigger Signal Setup tab before you add any trigger units.

Changing the trigger radix value does not affect any trigger units that were created before you made the change.

2. To add a new trigger unit, click **Add** in the Trigger Unit section of the Trigger Signal Setup tab.

   A line now appears in the Trigger Unit section, with a default trigger unit named TU<*number*>, where *number* is a sequential number. The first trigger unit is named TU1 by default.

Renaming Trigger Units

You can rename a trigger unit.

To rename a trigger unit:

- Double-click in the appropriate box in the Name column of the Trigger Unit section of the Trigger Signal Setup tab, backspace over the existing name, and type in the new name.

Setting Up Trigger Units

All signals must be defined for a trigger unit in Reveal Inserter. You cannot change them in Reveal Logic Analyzer.

To set up a trigger unit:

1. If you want to change the default name of the trigger unit, backspace over the default name in the Name box in the Trigger Unit section of the Trigger Signal Setup tab and type the new name.
2. Specify the signals in the trigger unit:
   a. Double-click in the box in the **Signals (MSB:LSB)** column.

      The Select Signals dialog box appears.

   b. In the Select Signals box of the dialog box, highlight the signal or signals that you want to use in the trigger unit, and click **>** to move them to the box on the right. (Shift-click to select multiple signals.)

      Each trigger unit can have up to 256 signals. Since there are 16 allowable trigger units, each core can have a maximum of 4096 trigger signals.

   c. If you want to change the order of a signal in the list of signals, highlight its name and click  to move it up one line or  to move it down one line.

      The order of the signals affects how the comparison is performed.

   d. Click **OK**.

   As an alternative to this procedure, you can drag and drop signals from the Design Hierarchy pane to the Signals (MSB:LSB) box in a trigger unit.

   If you want to select certain signals by using a search engine:

   a. In the Signal Search box beneath the Design Hierarchy pane, enter the name or pattern of the signal to find. You can set a filter by using case-insensitive alphanumeric characters and wildcards. See

"Searching for Signals" on page 12 for information about the wildcards that you can use.

b. Click **OK**.

If Reveal Inserter finds only one signal, it highlights it in the Design Hierarchy pane.

If Reveal Inserter finds multiple signals, it opens the Search Signals dialog box to list all the signals found.

c. If you are searching for multiple signals, select the desired signals in the Search Signals dialog box, and click **OK**.

The signals are now selected in the Design Hierarchy pane.

d. Drag them to Signals (MSB:LSB) box in the Trigger Unit section of the Trigger Signal Setup tab.

If you move the cursor over a trigger-unit line in the Signals box, the software displays a complete list of the signals in that trigger unit.

3. In the **Operator** column, set the comparators for the trigger condition. You can choose from the following states:

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- Rising edge – compares on the rising edge of the clock
- Falling edge – compares on the falling edge of the clock
- Serial compare – compares until the trigger condition is met. For example, if the trigger condition is 10011, the serial compare option looks for a 1 on the first clock, a 0 on the next clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only if those five conditions are met in those five clock cycles will the serial compare output be active.

  The serial comparator is available only when a single signal is listed in the Trigger Unit signal list. If you choose this option, you must choose Binary in the Radix box.

  You can only set the serial compare operator in Reveal Inserter. You cannot change it as you can other operators in Reveal Logic Analyzer.

The default comparator is == (equal to).

For more information on the effect of the "Rising edge" and "Falling edge" operators, see "Rising-Edge and Falling-Edge Operators" on page 19.

Both the operator type and the trigger unit value can be changed in Reveal Logic Analyzer during hardware debugging.

4. In the **Radix** column, set the radix of the trigger unit value given in the Value box by selecting a radix from the drop-down menu. You can choose one of the following:
   - Binary. This is the default. You must choose Binary if you selected “Serial compare” as a comparator.
   - Octal
   - Decimal
   - Hexadecimal
   - <*token_set_name*>. To select <*token_set_name*>, you must have created token sets. See “Using Tokens” on page 33 for instructions on creating token sets.
5. In the **Value** column, enter the comparison value.

   This value is the pattern of highs and lows that you want on the trigger unit that will initiate collection of the trace data. The default is binary, unless you selected <*token_set_name*> in the Radix column.

   If you selected <*token_set_name*> in the Radix column, a drop-down menu opens in the Value column. This menu lists all the tokens that you entered in the Token Manager dialog box for the chosen token set. Select any name. The only token sets available for a given bus must match the bit width of the bus. Other token sets will not be listed as choices for that bus.

   You can use “x” for a don’t-care value in the Value column if you selected Binary, Octal or Hexadecimal in the Radix column and if you selected the ==, !=, or serial compare operators in the Operator column.

Removing Trigger Units

You can remove trigger units in Reveal Inserter, but you cannot remove them in Reveal Logic Analyzer.

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**.

Adding 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 Logic Analyzer. You can add up to 16 trigger expressions.

You can add trigger expressions only in Reveal Inserter. You cannot add them in Reveal Logic 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**.

  A line appears with the default trigger expression called TE<*number*>, where <*number*> is a sequential number. The first trigger expression is named TE1 by default. You can rename the trigger expression by backspacing over the name and typing a new name.

Renaming Trigger Expressions

You can rename a trigger expression.

To rename a trigger expression:

- Double-click in the appropriate box in the Name column of the Trigger Expression section of the Trigger Signal Setup tab, backspace over the existing name, and type in the new name.

Setting Up Trigger Expressions

You set up the initial trigger expressions in Reveal Inserter, but you can change them and their names in Reveal Logic Analyzer. You can also enable or disable trigger expressions in Reveal Logic Analyzer. However, you cannot change the sequence depth, the maximum sequence depth, or the maximum event counter of the trigger expressions in Reveal Logic Analyzer.

To set up a trigger expression:

1. If you want to change the default name of the trigger expression, backspace over the default name in the **Name** box in the Trigger Expression section of the Trigger Signal Setup tab and type the new name.

   You can also change the name of a trigger expression in Reveal Logic Analyzer.

2. In the **Expression** box, enter the names of the trigger units and the operators that you want to use to connect them.

   You can use the following operators to connect trigger units:

   - & (AND) – Combines trigger units using an & 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 21 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 21 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 21 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 21 for information on counters.

For more information on the precedence of these symbols in trigger expression syntax, see “Precedence” on page 23.

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 Logic Analyzer during hardware debugging.

3. From the drop-down menu in the **Ram Type** box, specify how the trigger expression is to be implemented in the debug logic. You can choose one of the following:
   - EBR – Implements the trigger expression as embedded block RAM (EBR). Reveal Inserter calculates the appropriate number of EBRs. By default, the trigger expression is implemented as EBR.
   - *<number>* Slices – Implements the trigger expression as slices. Reveal Inserter calculates the appropriate number of slices.

   The **Sequence Depth** box is read-only, so you do not need to enter data in this box.
4. From the drop-down menu in the **Max Sequence Depth** box, specify the maximum number of sequences, or trigger units connected by THEN operators, that can be used in a trigger expression.

   You can choose 1, 2, 4, 8, or 16. Reveal supports up to 16 maximum sequence levels.

   If the number in the Sequence Depth box is higher than that set in the Max Sequence Depth box, the number in the Max Sequence Depth box appears in red to indicate an error.

   The Max Sequence Depth value is set statically in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.

5. From the drop-down menu in the **Max Event Counter** box, specify the maximum size of the count in the trigger expression (the count is how many times a sequence must occur before a THEN statement). You can choose 1 and powers of 2 from 2 to 65,536. The maximum is 65,536. The default is 1. If the largest counter value used in the trigger expression is larger than that set in the Max Event Counter box, the number in the Max Event Counter box appears in red.

   You cannot change the Max Event Counter setting in Reveal Logic Analyzer. You can only change it in Reveal Inserter.

   You can also add a counter to the output of the final trigger from all the trigger expressions. This counter adds an option to the final trigger output that combines all the trigger expressions. It is similar to the AND All and OR All options in the Analyzer.

6. To add a counter to the output of the final trigger, do the following:

   a. Select the **Enable Final Trigger Counter** checkbox in the lower left portion of the Trigger Signal Setup tab.

   b. From the drop-down menu in the **Event Counter Value** box, select the maximum size of the count of all the trigger expression outputs combined. You can choose powers of 2 between 2 and 65536.

   Leaving the Enable Final Trigger Counter option unselected is equivalent to setting the counter to a value of 1.

   You can change the value of this parameter in Reveal Logic Analyzer, but you cannot make the value bigger, so be sure to reserve enough space for the count that you think you will need.

7. If you want create a trigger output signal, do the following in the **Trigger Out** section:

   a. Select the **Enable Trigger Out** option.

   b. If you want to create a net, type in the name of the signal that you want to use as the trigger output signal in the **Net** box.

      The default name of the trigger output signal is reveal_debug_<*design_name*>_<*core_name*>_net. An example is reveal_debug_counter1_LA0_net.

   c. In the drop-down menu next to the Net box, select one of the following:

      - NET – Creates a net signal that can be connected to the input of a trigger unit of another core. This setting is the default.
      - IO – Creates an I/O signal that can trigger outside the chip.
      - BOTH – Creates a signal that can both connect to the input of a trigger unit of another core and trigger outside the chip.

   d. In the Polarity box, select the polarity of the trigger output signal from the drop-down menu, either **Active High** or **Active Low**.

   e. 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.

Once you create a net as a trigger output signal, its name appears in the Trigger Output Signal from Reveal Core(s) box beneath the Design Hierarchy pane.

Removing Trigger Expressions

You can disable a trigger expression from being used by deselecting the checkbox to the left of the trigger expression name in Reveal Logic Analyzer, but you can remove a trigger expression only in Reveal Inserter.

To remove a trigger expression:

1. Click in any box in the line representing the expression that you want to remove.
2. Click **Remove**.

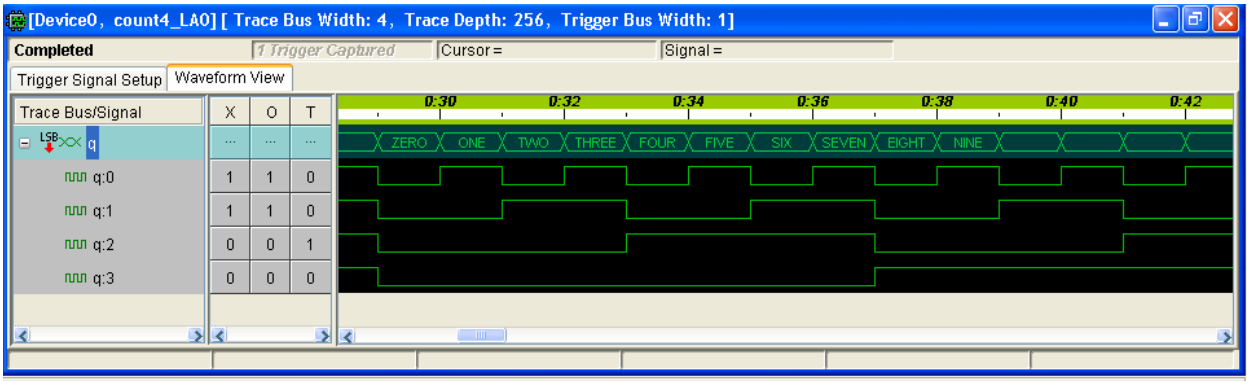
Using Tokens

Reveal Inserter supports multiple token radixes so that you can use different radixes for different buses in Reveal Logic Analyzer. For example, you might want to use one token radix for a state machine bus and another radix for a data bus.

Tokens

Tokens are labels that you can assign to trigger unit values in Reveal Inserter. They can be displayed on the waveforms in the Waveform View tab of Reveal Logic Analyzer. These labels identify all trigger unit values the same way for clarity, without regard to the radix notation in which the values appear. For example, the waveforms in the following figure are labeled in decimal notation (decimal radix):

Figure 3: Waveforms Labeled with Decimal Notation



Here are some examples of token values showing the label on the left and the radix values on the right:

```
ZERO=d'00
ONE=d'01
TWO=b'10
THREE=d'3
FOUR=b'100
```

```
FIVE=x'5
SIX=b'110
SEVEN=b'111
EIGHT=h'8
NINE=d'9
```

In these examples, Reveal Logic Analyzer assigns all buses with a value equal to 0 the label "ZERO," all buses with a value equal to 1 the label "ONE," all buses with a value equal to 2 the label "TWO," and so forth. For example, a bus consisting of four signals in states 0, 1, 0, 1 has a value of 0101 in binary notation, which is equal to 5, and a value of 5 in decimal, octal, and hexadecimal notation. The bus would therefore be assigned the label "FIVE."

Token values must be prefixed by one of the radix indicators shown in the following table and the ' symbol:

Radix	Prefix	Example
Binary	b'	b'110x0
Octal	o'	o'53
Decimal	d'	d'123
Hexadecimal	x'	x'0F2

If a value does not have a prefix, its radix is assumed to be binary. You can use a case-insensitive "x" in binary numbers as a don't-care value.

You should define a token in Reveal Inserter for each bus value that you wish to view. For example, if you want to assign a label to a bus with a value of 37, you must define the THIRTY-SEVEN label in Reveal Inserter. The example tokens just given do not assign labels to any buses with a value larger than 9, so these buses appear without labels whenever the bus has a value of 10 or greater.

Observe the following guidelines when creating tokens:

- Each token must have a name.
- The name of the token must be unique in the token set.
- The name of the token must consist of letters, numbers, and "_." It must start with a letter. Token names are case-insensitive.
- The value of the token must have a valid format and valid value.

Token Sets

As noted earlier, Reveal Inserter supports multiple token radixes so that you can use different radixes for different buses in Reveal Logic Analyzer. To this end, Reveal enables you to create sets of tokens through the Token Manager dialog box activated by the Datasets > Token Set Manager command in Reveal Inserter. You assign a name and a width, in bits, to each token set, then add tokens and token values to it. See “Creating a Token Set” on page 35 for detailed instructions. The name of each token set appears on the Bus Radix dialog box in the Waveform View tab in Reveal Logic Analyzer. All token sets are saved in the .rvs file.

Observe the following guidelines when creating token sets:

- A token set must have a name.
- The name of the token set must be unique.
- The name of the token set must consist of letters, numbers, and “_.” It must start with a letter. Token set names are case-insensitive.
- The width of the token set must be in integers from 0 through 9, inclusive.
- The width of the token set must be less than or equal to 256.

Token File

You can store the tokens and token sets that you created in the graphical user interface in a token (.rvt) file, which you can then import at a later time. This token file is in XML format and uses the same syntax as the token section in the .rvs file:

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<Tokenset>
<Token name=”xxx” width=”x”>
<entry key=”xx” value=”xx”/>
<entry key=”xx” value=”xx”/>
...
</Token>
...
</Tokenset>
```

See “Creating (Exporting) a Token File” on page 37 for instructions on creating token files, and “Importing a Token File” on page 38 for instructions on using a previously created token file.

Creating a Token Set

To create a token set:

1. Choose **Datasets > Token Set Manager**.
2. In the Token Sets pane of the Token Manager dialog box, click **Add** to add a token set.

   A token set with the default name of TokenSet1 is now added in the Name column to the Token Sets pane.

3. If you want to change the name of the token set, backspace over the default name in the Name column and enter the new name of the token set.
4. In the Max Token Bits column of the line, double-click and enter the width of the maximum number of tokens in that token set, in bits. Enter the exponent when the maximum number of tokens is written in powers of 2. For example, if the token set has 16 tokens, you would enter 4 in the Max Token Bits column, which is the exponent when 16 is written in exponential notation (2^4). The maximum value of 2 is 1 bit, of 4 is 2 bits, of 8 is 3 bits, of 16 is 4 bits, of 32 is 5 bits, and so forth.
5. Repeat steps 2 through 4 for each token set that you want to add.

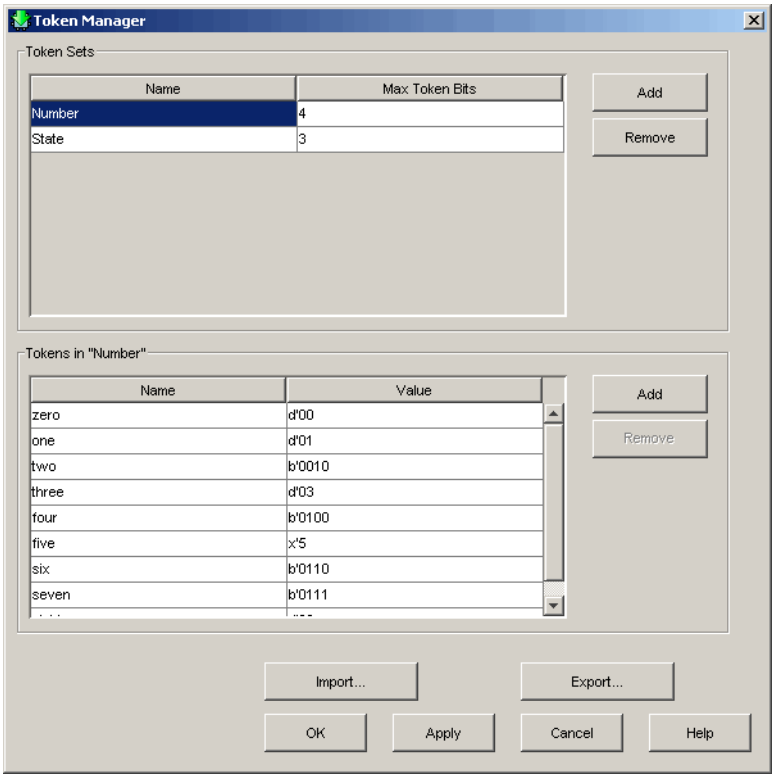
   Now you will add the tokens to each token set.
6. In the Token Sets pane, highlight the line containing the name of the first desired token set.
7. In the Tokens in <*token_set*> pane, click **Add**.

   A token with the default name of Token0 is now added in the Name column to the Tokens in <*token_set*> pane. By default, each token has no value.
8. If you want to change the name of the token, backspace over the default name and enter the new label for each anticipated bus value.
9. In the Value column of the token line, enter the anticipated bus value. Token values must be in the format shown in the previous section.
10. Repeat steps 7 through 10 to add the tokens for each token set.

    The number of tokens per set does not have to be in powers of 2. If the number of tokens is not in powers of 2, enter the exponent of the next largest number that is in powers of two in the Max Token Bits column. For example, if you enter 10 tokens in a set, you would enter the exponent of the next largest number that is in powers of 2, which is 16 (2^4). You would therefore enter 4 in the Max Token Bits column. The software automatically creates the 6 missing values as “not defined.”
11. Click **OK** to close the dialog box or **Apply** to keep the dialog box open.

The following figure shows a sample Token Manager dialog box. In this figure, the tokens are displayed for the “Number” token set.

Figure 4: Example Token Manager Dialog Box



Editing Existing Tokens

To edit existing tokens:

1. Choose **Datasets > Token Set Manager**.
2. In the Token Manager dialog box, use the **Add** and **Remove** keys to add and remove token sets and tokens. In addition, you can edit table cells directly to change token set names, token set bit widths, token names, and token values.
3. Click **OK** to close the dialog box or **Apply** to keep the dialog box open.

Creating (Exporting) a Token File

You can store the tokens and token sets that you created in the Reveal Inserter graphical user interface in a token (.rvt) file to re-import at a later time.

To create a token file:

1. Create the desired token sets and tokens.
2. Choose **Datasets > Token Set Manager** to open the Token Manager dialog box.
3. Click **Export**.

4. In the Export Tokens dialog box, type the name of the new token (.rvt) file in the File Name box, and select **Reveal token file (*.rvt)** in the File Types box. Click **Open**.

   Alternatively, if you want to overwrite an existing token (.rvt) file, browse to the file, then click **Open**.

5. In the Export Successful message box, click **OK**.

Reveal Inserter now writes all the tokens from the graphical user inteface to the token file.

Importing a Token File

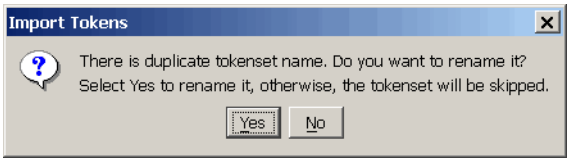
You may want to import the tokens and token sets from a token (.rvt) file that you created previously into the Reveal Inserter graphical user interface.

To import a token file:

1. Choose **Datasets > Token Set Manager** to open the Token Manager dialog box.
2. Click **Import**.
3. In the Import Tokens dialog box, browse to the token (.rvt) file to import or type the file name in the File Name box, then click **Open**.

   If the name of a token set in the imported file is the same as the name of a token set that already resides in the graphical user interface, the following message box comes up to ask you if you want to rename it.

Figure 5: Import Tokens Message Box



   Click **Yes** to rename it. By default, token sets are renamed <*original_name*>_01,<*original_name*>_02, <*original_name*>_03, and so forth. You cannot provide your own name for the token set. If you do not rename the token set, it will not be imported.

4. Click **OK**.

Reveal Inserter now places the tokens and token sets from the imported token (.rvt) file in the Reveal Inserter graphical user interface.

Checking the Debug Logic Settings

Reveal Inserter automatically checks the settings of the debug logic before saving the project or inserting the debug logic cores, but you may want to check them independently beforehand. With one DRC command (Datasets > Design Rule Check), you can verify the following:

- The core names begin with a letter and consist of letters, numbers, and underscores (_).
- A core name is not the same as that of any other core.
- The core name is not the same as that of any module already defined in the design.
- The number of cores is between 1 and 15.
- The number of trace signals is between 1 and 512.
- The number of trigger signals is between 1 and 4096.
- The number of trigger units and trigger expressions is between 1 and 16.
- The number of trigger signals in a trigger unit is between 1 and 256.
- A sample clock is specified.
- The sample clock signal is a 1-bit signal already defined in the design.
- A sample enable is specified.
- The sample enable signal is a 1-bit signal already defined in the design.
- The name of the trigger-out signal is given if this signal is enabled.
- The name of the trigger-out signal is not the same as any signal already defined in the design.
- The number of EBRs needed does not exceed the number available.
- The design includes an input signal.
- The syntax of the trigger expressions is correct.
- The trigger expression sequence is less than or equal to the maximum sequence.
- The trigger output signal is specified, if the Enable Trigger Out option is enabled.
- The trigger output signal is not the same as the name of any signal in the design.
- The values of the trigger unit are correct.
- The names of the trigger units and the trigger expressions conform to the guidelines given in the “Trigger Expression and Trigger Unit Naming Conventions” on page 25.
- The bit widths of the token values are the same as the bit widths of the trigger unit signals.

To check the logic debugging settings:

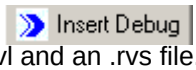
- Choose **Datasets > Design Rule Check** or click  in the toolbar.

The results of the check are displayed in the Message tab. The Message tab also displays the total resource utilization, as in the following example:

```
The number of EBRs needed is 2.
The number of DistRAM (logic/ROM/RAM) slices needed is 0.
```

Saving a Project

Once you set the debug options, save the project so that the project information is saved in an .rvl and an .rvs file. Reveal Inserter automatically performs a design rule check before it saves these files.

When you select Datasets > Insert Debug or click the 

button, Reveal Inserter saves the project information in an .rvl and an .rvs file.

Note

Reveal Inserter generates a “signature” or tracking mechanism each time that debug logic is inserted into the design. The signature is placed into the project file and into the debug logic. Reveal Logic Analyzer reads this signature to ensure that the FPGA has been programmed with the latest debug logic. Reveal Inserter generates a new signature every time the .rvl file is written, and Reveal Logic Analyzer checks this signature each time that it runs the design. If you save the project in Reveal Inserter without re-running the Project Navigator flow, Reveal Logic Analyzer issues an error message, even if the debug logic was not changed.

To save the project settings in the current directory:

- Choose **File > Save Project** or click  in the toolbar to save the project in .rvl and .rvs files in your current directory.

To save the project settings in another directory:

- Choose **File > Save As Project** to save the project in .rvl and .rvs files in a directory other than the current directory. In the Select Project dialog box, browse to the desired directory, enter the name of the .rvl file in the File Name box, select **.rvl** in the Files of Type box, and click **Save**.

Inserting the Debug Logic Cores

Once you set all the options in the tabs in Reveal Inserter window, you can insert the debug logic cores into the design.

To insert the debug logic cores into the design:

1. Choose **Datasets > Insert Debug** or click 

in the toolbar.
2. In the Insert Debug to Design dialog box, select the cores to insert.
3. In the box beneath “Please select design .lpf file you want to use,” enter the path and name of the .lpf file to use.

   You must specify the .lpf file, which contains the timing preferences, to avoid generating false errors in the timing between the JTAG clock and

the sample clocks. By default, Reveal Inserter uses the default .lpf file for the project.

4. If you want to import the .rvl file into the ispLEVER project, select **Import Reveal file to ispLEVER project**.

   Importing the .rvl file is the default, and the “Import Reveal file to ispLEVER project option” should not normally be left unchecked. If the .rvl file is not present in Project Navigator, Reveal Inserter does not insert the debug logic into the design. If the .rvl file is changed and it is not re-imported into Project Navigator, the design will not be implemented correctly.

   Importing the .rvl file enables ispLEVER to include the debug logic in the implementation. Only one .rvl file can be imported into an ispLEVER project. If you have previously imported an .rvl file, Project Navigator asks if it should replace the file. Reveal Inserter automatically saves the .rvl file before it inserts the debug logic.

5. Click **Insert**.

   This step automatically performs a design rule check, saves the debug options, generates and builds the debug logic cores, invokes the synthesis tool to synthesize the cores, and imports the .rvl file into Project Navigator. A progress bar in the lower right of the Reveal Inserter window shows when the insertion is complete. If Reveal Inserter encounters no errors, you should see the following message in the Resource Usage tab, indicating that the core insertion was successful:

   ```
   Reveal Project imported.
   ```

   The Resource Usage tab also displays the total resource utilization, as in the following example:

   ```
   The number of EBRs needed is 2.
   The number of DistRAM (logic/ROM/RAM) slices needed is 0.
   ```

   In addition, a message box opens, informing you that the insertion was successful.

6. Click **OK** in the message box.

You should now see the .rvl file listed in the Sources in Project pane of Project Navigator, if it was not listed there before.

Removing Debug Logic from the Design

You may want to remove the debug logic cores in pre-production versions of your device to free block RAM resources and LUT-based logic and to expand the design. If you want to remove the debug logic cores from your design, you must remove the .rvl file from Project Navigator. Otherwise, the cores will continue to be inserted.

To remove the debug logic cores from the design:

1. In Project Navigator, highlight the .rvl file and right-click.

2. Choose **Remove**.

The .rvl file is now removed from the design.

Closing a Project

To close a Reveal Inserter project:

- Choose **File > Close Project**.

Deleting a Project

To delete a Reveal Inserter project:

- Choose **File > Delete Project**.

Exiting Reveal Inserter

To exit Reveal Inserter:

- Choose **File > Exit**.

Building the Database

To build the design project database:

- Double-click the **Build Database** process in the ispLEVER Project Navigator.

During this process, Project Navigator invokes the synthesis tool.

Note

If your design contains an unlicensed IP block, the Hardtimer mechanism enables you to evaluate the IP. You can control this mechanism by highlighting the Build Database Process, selecting Properties, and setting the Hardware Evaluation option to Enable.

If you use the Reveal tools on a design that includes an unlicensed IP block, you cannot disable the Hardtimer mechanism. It is required to generate the bitstream data or JEDEC file for Reveal Logic Analyzer. If you set the Hardware Evaluation option to Disable, the Reveal flow overwrites the Hardtimer mechanism. However, the option automatically reverts to Disable when you exit the Reveal flow.

Mapping, Placing, and Routing the Design

To map, place, and route the design:

1. Double-click the **Map Design** process in the ispLEVER Project Navigator.
2. Double-click the **Place & Route Design** process in the ispLEVER Project Navigator.

All core clock pins must be located and driven by valid signals to ensure successful hardware debugging.

Generating a Bitstream or JEDEC File

To generate a bitstream:

- Double-click the **Generate Bitstream Data** or **Generate Data File (JEDEC)** process in Project Navigator.

  This process creates a .bit or .jed file for FPGAs that is ready for downloading into the device.

Connecting to the Evaluation Board

Reveal Logic Analyzer requires that a Lattice Semiconductor or USB download cable and a power supply be installed between your computer and evaluation board. Refer to the ispVM System online Help or the Reveal Logic Analyzer online Help for more information about setting up a cable connection.

Downloading Design onto the Device

For more information about downloading your design onto the device, refer to the ispVM System online Help or the Reveal Logic Analyzer online Help.

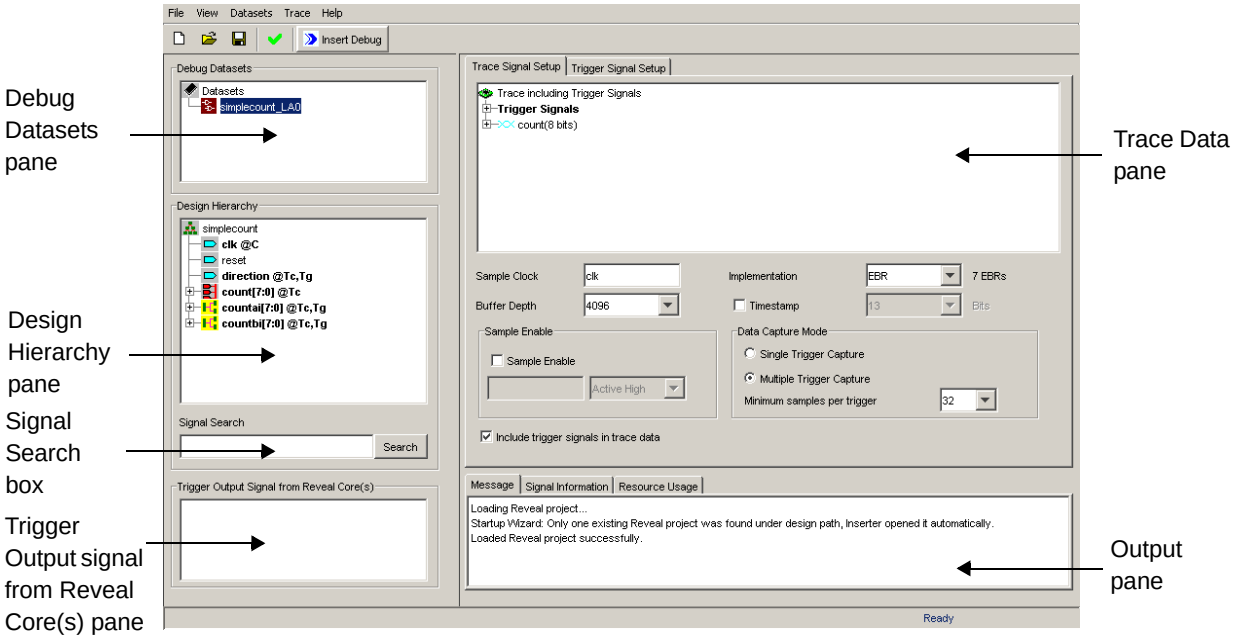
Performing Logic Analysis with Reveal Logic Analyzer

After you have created your design project database with ispLEVER software, generated a debug logic core with Reveal Inserter, mapped, placed, and routed your design, and downloaded the design to the evaluation board, you can perform a logic analysis with Reveal Logic Analyzer. Refer to the Reveal Logic Analyzer online Help for more information about performing logic analysis.

User Interface Descriptions

The Reveal Inserter window appears when you first select Tools > Reveal Inserter in Project Navigator or click on the icon.

Figure 6: Reveal Inserter Window



The Reveal Inserter window includes the following features:

Debug Datasets pane Lists the cores in the current dataset. You debug a design with Reveal Inserter debug logic, using a certain sample clock. If you want to debug a multi-clock design, you can create a core for each sample clock region. These cores are listed in the Debug Datasets pane.

Design Hierarchy pane Lists all the buses and signals in the design. The names of trace, trigger, and control signals are in bold font if they are currently being used.

One of the following strings appears after each signal name to indicate its use:

- @Tc indicates that the signal is a trace signal.
- @Tg indicates that the signal is a trigger signal.
- @C indicates that the signal is a control signal.

Similarly, one of the following strings appears after each bus name to indicate its use:

- @Tc indicates that all the signals in the bus are used only as trace signals.
- @Tg indicates that all the signals in the bus are used only as trigger signals.

- @Tc, Tg indicates that all the signals in the bus are used as trace signals and trigger signals. It also appears if all the signals are used as trigger signals and none of the signals in the bus are used as control signals and you selected the “Include trigger signals in trace data” option.
- @Mx indicates the following:
  - At least one signal in the bus is used as a control signal.
  - Some signals in the bus are used both as trigger signals and as other kinds of signals.
  - Some signals in the bus are used both as trace signals and as other kinds of signals, except that all the signals are used as trigger signals, none of the signals are used as control signals, and you selected the “Include trigger signals in trace data” option.

If you select or deselect the “Include trigger signals in trace data” option, the signal and bus names are immediately updated in the Design Hierarchy pane. If you set a signal as a trigger signal and select the “Include trigger signals in trace data” option, the use of the signal is displayed as Tc, Tg, even though you did not drag the signal name to the Trace Data pane.

If you select a signal in the hierarchy, the Signal Information tab at the bottom of the Reveal Inserter window displays information about how it is used.

You can enlarge the width of this pane to see longer signal names by dragging the splitter at the right edge of the pane.

Signal Search box Enables you to search for a signal or a group of signals. You can enter a signal name or pattern. You can set a filter by using the case-insensitive alphanumeric characters and wildcards described in “Searching for Signals” on page 12.

If Reveal Inserter finds only one signal, it highlights it in the Design Hierarchy pane. If it finds multiple signals, it opens the Search Signals dialog box to list all the signals found. When you click OK, the selected signals are highlighted in the Design Hierarchy pane. From the Design Hierarchy pane, you can drag signals to the Trace Data pane, the Sample Clock box, and the Sample Enable box in the Trace Signal Setup tab. You can also drag signals to the Signals (MSB:LSB) box in the Trigger Unit section of the Trigger Signals Setup tab.

Trigger Output Signal from Reveal Core(s) pane Displays the names of the trigger output signals defined in the Trigger Out box in the Trigger Signal Setup tab.

This field displays the trigger output signals for all but the first core and only those output signals for which NET or BOTH were chosen. From the Trigger Out Nets box, you can drag the signal names to the top half of the Trace Signal Setup tab.

Trace Signal Setup Activates the Trace Signal Setup tab.

Trigger Signal Setup Activates the Trigger Signal Setup tab.

Trace Data pane Displays the selected trace signals in the Trace Signal Setup tab.

Output pane Displays the information in the Message, Signal Information, and Resource Usage tabs.

Message tab Displays error and informational messages, such as the results of the design rule check and the core insertion. After Reveal Inserter checks the design rules and inserts the cores, the Message tab displays the total resource utilization, for example:

```
The number of EBRs needed is 2.
The number of DistRAM (logic/ROM/RAM) slices needed is 0.
```

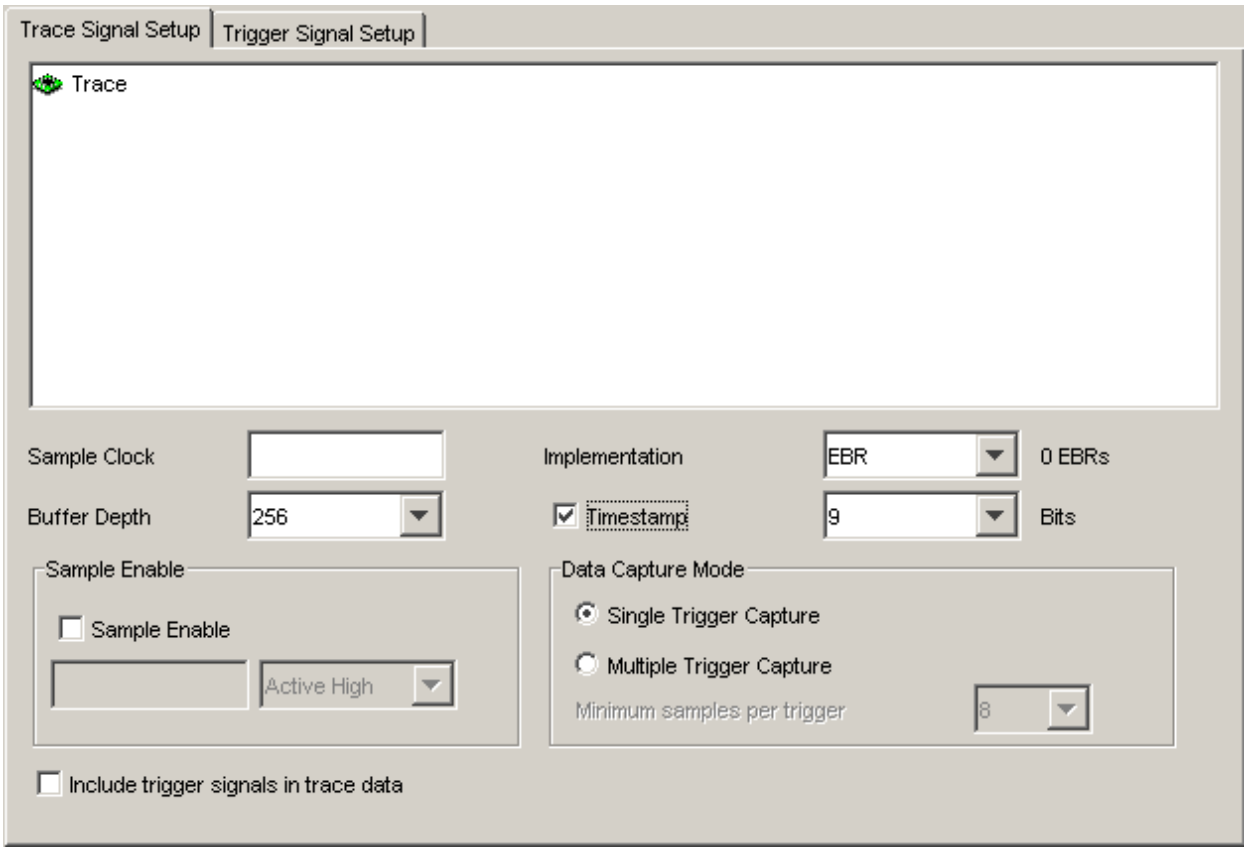
Signal Information tab Displays information on the signal selected in the Design Hierarchy pane, such as signal origin and destination, fanout, and attributes. For both the RTL and the EDIF flow, this tab lists the signal names, whether they are used as trace, trigger, or control signals, and whether they are inputs, outputs, or nets.

Resource Usage tab Displays information on the resources consumed by the pieces of the debugging hardware, such as the trigger logic and trace buffer. The debug resource usage information is not updated until after the Insert Debug operation has been completed.

Trace Signal Setup Tab

The Trace Signal Setup tab enables you to specify the signals whose states, or data values, will be monitored and captured in the trace buffer for debugging purposes.

Figure 7: Trace Signal Setup Tab



Trace Data pane Displays the names of the trace signals. In the Trace Data pane, you can do the following:

- Select multiple signals and buses and group them as a new bus.
- Drag and drop to group signals and buses into a bus and ungroup signals and buses from a bus.
- Use a pop-up menu to group and ungroup signals and buses.
- Use a pop-up menu to rename a bus.
- Drag a bus or signal to change its order. Grouped signals are grouped in LSB order.

Sample Clock Specifies the name of the clock that determines when the trace data is captured in the trace buffer. You can either type in the clock signal name or drag it from the Design Hierarchy pane.

Buffer Depth Specifies the size of the trace memory. This parameter defines the number of trace bus samples that a core can capture. It can be set

to a minimum of 16 or to powers of 2 from 16 to 65536. The buffer size is determined by the type of hardware.

Sample Enable Determines whether a sample enable signal is used. A sample enable is an optional signal used to capture data on the sample clock only when the sample enable signal is active. If you do not specify a sample enable signal, trace data is collected on every sample clock after the trigger. You can either type in the sample enable signal name or drag it from the Design Hierarchy pane. The Sample Enable checkbox is unselected by default.

If you select the checkbox, the boxes beneath it become available so that you can specify the name of the sample enable signal. You can set the sample enable signal to one of the following:

- Active High, which means that trace data is captured on the rising edge of the clock when the sample enable signal is high. Active High is the default.
- Active Low, which means that trace data is captured on the rising edge of the clock when the sample enable signal is low.

Each sample shown in the trace buffer is only captured when the sample enable is active and there is a sample clock. Data samples may be discontiguous, unlike those in a normal data capture.

It is also possible that the actual trigger condition may occur when the sample enable is not active, causing two changes from a normal data capture:

- The actual data values for the trigger condition may not be visible, because the data cannot be captured when the sample enable is inactive.
- Reveal Logic Analyzer cannot accurately calculate the trigger point, since the trigger point may have occurred when the sample enable was inactive. Normally a trigger point is shown as a single marker on the clock on which the trigger occurred. If a sample enable is used, a trigger region that spans 5 clock cycles is shown instead. Reveal Logic Analyzer can guarantee that the trigger occurred in this region, but it cannot determine during which clock cycle the trigger occurred.

Implementation Specifies how the trace buffer logic is to be implemented in the FPGA. You can choose one of the following in the drop-down menu:

- EBR – Implements the debugging logic as embedded block RAM (EBR). This setting is the default.
- DistRAM – Implements the debugging logic as distributed RAM.

Once you set this option, Reveal Inserter displays the total number of active cores.

Timestamp Enables you to specify a sample clock count value to be stored with each trace sample to indicate the sample count clock value at which the sample was captured. This time stamp enables you to see how many sample clock intervals have elapsed between data captures when you use a sample enable.

If you select the Timestamp option, you must also enter in the Bits box the amount of trace memory storage that the time stamp needs, in bits.

Bits Specifies the amount of trace memory storage needed by the time stamp, in bits.

The minimum value available in the drop-down menu in the Bits box is obtained by multiplying the value in the Buffer Depth box by 2 and converting the result to an exponential value. For example, if the value in the Buffer Depth box is 256, the minimum number of bits in the Bits drop-down menu is calculated as follows:

256 X 2 = 512

$512 = 2^9$

So the minimum number of bits available in the Bits menu in this case is 9.

The maximum number of bits available in the Bits menu is always 63.

Data Capture Mode Specifies how Reveal Logic Analyzer captures the trace data. You can select Single Trigger Capture or Multiple Trigger Capture. Single Trigger Capture is the default.

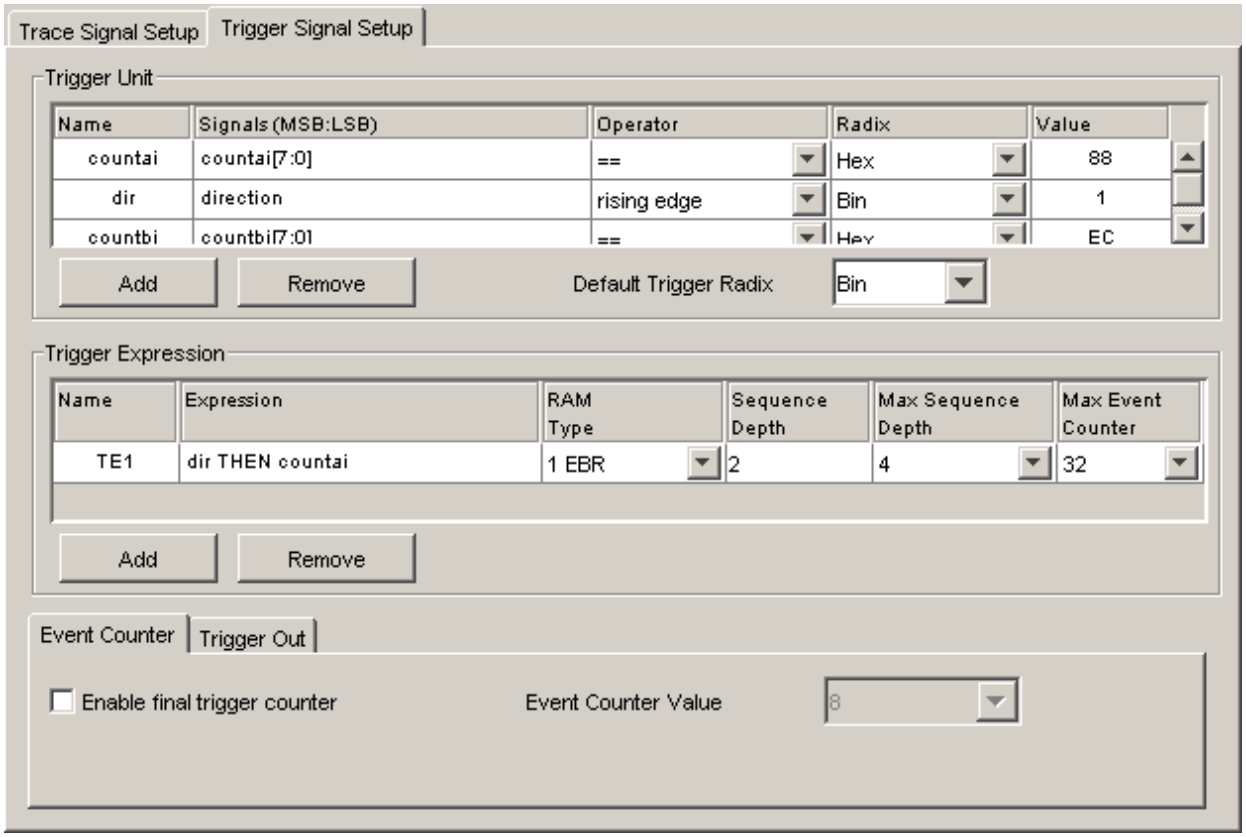
- Single Trigger Capture mode – The trace buffer captures the data around a single trigger event. The amount of data stored before and after the trigger is determined by the trigger position settings in Reveal Logic Analyzer. Selecting Device > Run Selected LAs enables the logic to start looking for the trigger, but it does not activate the trigger.
- Multiple Trigger Capture mode – The trace buffer can capture data associated with multiple trigger events. Each trigger and the associated data must be a minimum size. In Reveal Inserter, you specify the minimum number of samples captured per trigger. When you run Reveal Logic Analyzer, the number of samples per trigger can be increased up to the size of the trace buffer, but it cannot be decreased to less than the minimum number of samples specified in Reveal Inserter.
- Minimum samples per trigger – When you select Multiple Trigger Capture, you must also specify the minimum number of samples to collect per trigger. This setting of this option depends on the device and other trace options.

Include trigger signals in trace data Determines whether the trigger signals are added to the trace signals so that the data from the trigger signals is included in the trace data. Enabling this option uses more hardware resources to store the trigger signal data. This option is turned off by default.

Trigger Signal Setup Tab

The Trigger Signal Setup tab enables you to select the trigger signals and define the data values or pattern of data values that causes the collection of trace data to begin.

Figure 8: Trigger Signal Setup Tab



The Trigger Signal Setup tab includes the Trigger Unit, Trigger Expression, and Event Counter sections.

Trigger Unit

The parameters in the Trigger Unit section of the Trigger Signal Setup tab enable you to configure the trigger units, which are the basic trigger comparison mechanism in Reveal Inserter and Reveal Logic Analyzer. You select signals from the design and add them to a trigger unit. Trigger units allow comparison of the signal to a value that is entered during hardware debug. All trigger units are automatically available for use in all trigger expressions defined. You can include up to 16 trigger units in a core. See “Setting Up the Trigger Signals” on page 18 for detailed information about trigger units.

Each trigger unit consists of the following information:

Name Specifies the name of the trigger unit. See “Trigger Expression and Trigger Unit Naming Conventions” on page 25 for the guidelines governing trigger unit names.

Signals (MSB:LSB) Lists the signals in the trigger unit. Each trigger unit can have up to 256 trigger signals. Since there are 16 allowable trigger units, each core can have a maximum of 4096 trigger signals.

Operator Specifies the comparators that Reveal will use to compare the states of the trigger unit signals to the pattern of signal states that you set in the Trigger Signal Setup tab of Reveal Logic Analyzer. You can choose from the following states:

- == equal to. This comparator is the default.
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- Rising edge – Compares on the rising edge of the clock
- Falling edge – Compares on the falling edge of the clock
- Serial compare – Compares until the trigger condition is met. For example, if the trigger condition is 10011, the serial compare option looks for a 1 on the first clock, a 0 on the next clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only if those five conditions are met in those five clock cycles will the serial compare output be active.

  This comparator is available only when a single signal is listed in the Trigger Unit signal list. If you choose this option, you must choose Binary in the Radix box.

  You can only set the serial compare operator in Reveal Inserter. You cannot change it as you can other operators in Reveal Logic Analyzer.

Radix Specifies the radix of the trigger unit. It can be one of the following:

- Binary – This is the default. You must choose Binary if you selected “Serial compare” as a comparator.
- Octal
- Decimal
- Hexadecimal
- <*token_sets*> – If you have created token sets, the names of the token sets are listed in the drop-down menu. See “Creating a Token Set” on page 35 for instructions on creating token sets.

Value Specifies the comparison value. This value is the pattern of highs and lows that you want on the trigger unit that will initiate collection of the trace data. The default is binary, unless you selected <*token_set_name*> in the Radix column.

If you selected <*token_set_name*> in the Radix column, a drop-down menu opens in the Value column. This menu lists all the tokens that you entered in the Token Manager dialog box for the chosen token set. Select any name.

The only token sets available for a given bus must match the bit width of the bus. Other token sets will not be listed as choices for that bus.

You can use “x” for a don’t-care value in the Value column if you selected Binary, Octal or Hexadecimal in the Radix column and if you selected the ==, !=, or serial compare operators in the Operator column.

Add Adds a new trigger unit.

Remove Deletes the selected trigger unit.

Default Trigger Radix Specifies a default radix for the values of any new trigger units added. Changing the trigger radix value does not affect any trigger units that were created before you made the change.

Trigger Expression

The parameters in the Trigger Expression section of the Trigger Signal Setup tab enable you to configure the trigger expressions, which are combinatorial or sequential equations of trigger units or both that define when the collection of the trace data samples begins. You can add up to 16 trigger expressions. See “Setting Up the Trigger Signals” on page 18 for detailed information about trigger expressions.

Each trigger expression consists of the following information:

Name Specifies the name of the trigger expression. The default name is TE<*number*>, where <*number*> is a sequential number. See “Trigger Expression and Trigger Unit Naming Conventions” on page 25 for the guidelines governing trigger expression names.

Expression Specifies the trigger units and the operator or operators that indicate the relationship of one trigger unit to other trigger units. You can use the following operators to connect trigger units:

- & (AND) – Combines trigger units using an & 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 21 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 21 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 21 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 21 for information on counters.

For more information on the precedence of these symbols in trigger expression syntax, see “Precedence” on page 23.

RAM Type Specifies how the trigger expression is to be implemented in the debug logic. You can choose one of the following:

- EBR – Implements the trigger expression as embedded block RAM (EBR). Reveal Inserter calculates the appropriate number of EBRs. By default, the trigger expression is implemented as EBR.
- Slice – Implements the trigger expression as slices. Reveal Inserter calculates the appropriate number of slices.

For more information on trigger expressions, see “Trigger Expressions” on page 20.

Sequence Depth Specifies the number of sequences, which are sequential states connected by THEN operators, used in a trigger expression. Reveal supports up to 16 sequence levels. For example, in the following figure, TE1 consists of one sequence, since it has no THEN operator. It therefore has a sequence depth of 1. TE2 has two sequences, TU1|TU2 and TU3 & TU2, linked by a THEN operator, so its sequence depth is 2. TE3 has three sequences:

- TU1 & TU3 & TU2 followed by THEN
- TU1 followed by THEN
- TU3

TE3 therefore has a sequence depth of 3.

Figure 9: Trigger Expression Section

Name	Expression	RAM Type	Sequence Depth	Max Sequence Depth	Max Event Counter
TE1	TU1 & TU2	1 EBR	1	2	2
TE2	TU1 | TU2 THEN TU3 & TU2	1 EBR	2	2	2
TE3	TU1 & TU3 & TU2 THEN TU1 THEN TU3	1 EBR	3	4	4

The Sequence Depth option is read-only.

Max Sequence Depth Specifies the maximum number of sequences, which are trigger units connected by THEN operators, that can be used in a trigger expression. You can choose 1, 2, 4, 8, or 16. The maximum sequence depth is 16. If you set a number that is higher in the Sequence Depth box than that set in the Max Sequence Depth box, the number in the Max Sequence Depth box appears in red.

The Max Sequence Depth value is set statically in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.

Max Event Counter Determines the maximum size of the count in the trigger expression (the count is how many times a sequence must occur before a THEN statement). You can set this option to 1 or powers of 2 from 2 to 65,536. The maximum is 65,536. The default is 1. If the largest counter value used in the trigger expression is larger than that set in the Max Event Counter box, the number in the Max Event Counter box appears in red.

The Max Sequence Depth value is set statically in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.

Add Adds a line for a new trigger expression.

Remove Deletes the selected trigger expression.

Event Counter Tab

Enable Final Trigger Counter Determines whether a counter is added to the output of the final trigger. This counter is the combination of all the trigger expressions enabled when Reveal Logic Analyzer is run. This option is disabled by default.

Event Counter Value Specifies the maximum size of the count of the final trigger output. You can choose powers of two between 2 and 65536. In Reveal Inserter, this option has the effect of reserving space in the design for an extra counter added to the end of the triggering logic for the total number of triggers, both trigger units and trigger expressions. Leaving the Enable Final Trigger Counter option unselected is equivalent to setting the counter to a value of 1.

You can change the value of this parameter in Reveal Logic 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 Tab

Enable Trigger Out Creates a trigger output signal. You can create a net, an I/O, or both. If you want this signal to be an input to another core, select NET. If you want to trigger outside the chip, select IO. To specify both conditions, select BOTH. Selecting NET or BOTH activates the Net box so that you can enter the name of a trigger output signal.

Net Specifies the name of the trigger output signal enabled by the Enable Trigger Out parameter. The trigger output signal can be used to connect to the input signal of another core or to a general-purpose I/O pin in the FPGA, as an output from the chip, or both. For nets, the default name of the trigger output signal is reveal_debug_<*design_name*>_<*core_name*>_net, for example, reveal_debug_counter1_LA0_net.

You can set the trigger output signal to one of the following:

- NET – Creates a net signal that can be connected to the input of a trigger unit of another core. This setting is the default.
- IO – Creates an I/O signal that can trigger outside the chip.
- BOTH – Creates a signal that can both connect to the input of a trigger unit of another core and trigger outside the chip.

The Net option is only available if you select Enable Trigger Out.

Polarity Specifies whether the output trigger signal from outside the core is active high or active low. This option is only available if you select Enable Trigger Out.

Minimum Pulse Width Specifies the pulse width of the trigger out signal, as measured by the cycles of the sample clock. You can input any value of 0 or greater as the minimum pulse. This option is only available if you select Enable Trigger Out.

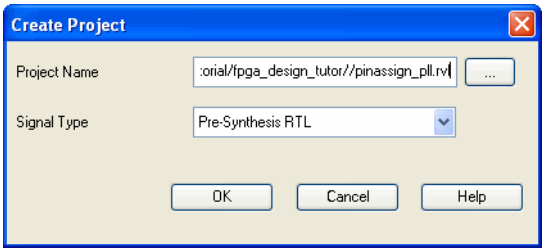
Dialog Boxes

The dialog boxes in Reveal Inserter graphical user interface are listed in alphabetical order.

Create Project Dialog Box

The Create Project dialog box is activated by the File > New Project command or the  button in the Reveal Inserter window.

Figure 10: Create Project Dialog Box



This dialog box contains the following parameters:

Project Name Specifies the name of the Reveal Inserter project. You can either type in the name of the project or click ... to browse to it. If you enter only a file name in this box, Reveal Inserter places the file in the design project directory.

Signal Type Specifies the type of input. The following flows are available on Windows:

- Pre-Synthesis RTL is available if you have a VHDL, Verilog, Schematic/Verilog, or Schematic/VHDL project.
- Post-Synthesis EDIF is available only for an EDIF or a mixed HDL project.

Only the EDIF flow is available on Linux.

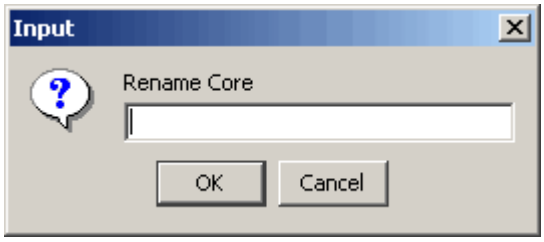
Cancel Closes the dialog box without applying any changes.

Help Displays the online Help topic for this dialog box.

Input Dialog Box

The Input dialog box is activated by any renaming command, such as Datasets > Rename Core or Trace > Rename Bus, or by right-clicking on the name of an item and choosing Rename from the pop-up menu.

Figure 11: Input Dialog Box



This dialog box contains the following parameters.

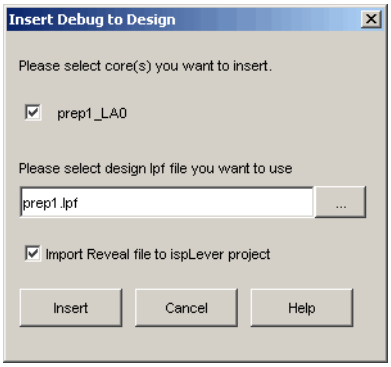
Rename *item_name* Specifies the new name for an item.

Cancel Closes the dialog box without applying any changes.

Insert Debug to Design Dialog Box

The Insert Debug to Design dialog box is activated by the Datasets > Insert Debug command or by clicking the Insert Debug button on the toolbar.

Figure 12: Insert Debug to Design Dialog Box



This dialog box contains the following parameters:

Please select core(s) you want to insert Enables you to select which cores to insert into the design.

Please select design .lpf file you want to use Specifies the path and name of the .lpf file to use. You must specify the .lpf file, which contains the timing preferences, to avoid generating false errors in the timing between the JTAG clock and the sample clocks. By default, this option uses the .lpf file for the project.

Import Reveal file to ispLEVER project Imports the .rvl file into the ispLEVER project. Importing the .rvl file enables ispLEVER to include the debug logic in the implementation. Only one .rvl file can be imported into an ispLEVER project. If you have previously imported an .rvl file, Project Navigator asks if it should replace the file. Reveal Inserter saves the .rvl file before it inserts the debug logic.

Insert Inserts the selected cores into the design.

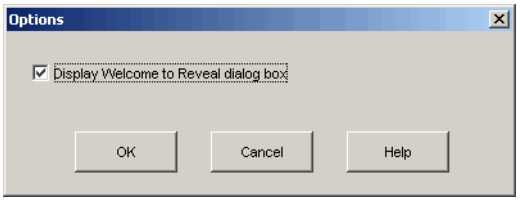
Cancel Closes the dialog box without applying any changes.

Help Displays the online Help topic for this dialog box.

Options Dialog Box

The Options dialog box is activated by the File > Options command.

Figure 13: Options Dialog Box



This dialog box contains the following parameters:

Display Welcome to Reveal dialog box Determines whether the Welcome to Reveal dialog box appears when you start Reveal Inserter. If you select this option, the Welcome to Reveal dialog box appears each time that you start Reveal Inserter. If you deselect this option, the Welcome to Reveal dialog box does not appear after the first time that you start Reveal Inserter. This option is selected by default.

To reinstate the appearance of the Welcome to Reveal dialog box after you have deselected the “Display Welcome to Reveal dialog box” option, select **File > Options** and in the Options dialog box, select **Display Welcome to Reveal dialog box**.

Cancel Closes the dialog box without applying any changes.

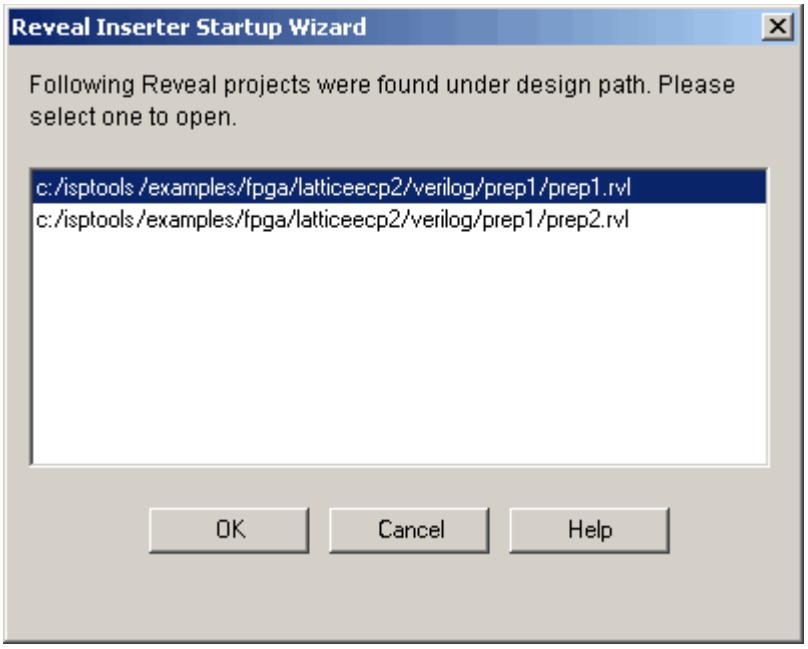
Help Displays the online Help topic for this dialog box.

Reveal Inserter Startup Wizard Dialog Box

The Reveal Inserter Startup Wizard dialog box is displayed in two forms. Both forms are activated by selecting Tools > Reveal Inserter in Project Navigator.

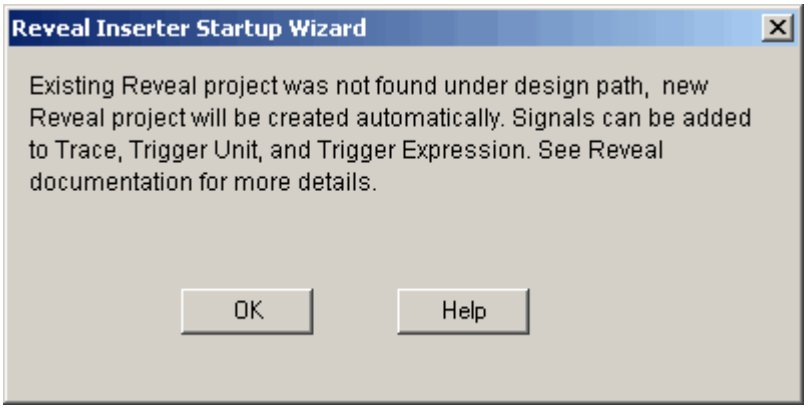
The following version of the dialog box is displayed only if Reveal Inserter finds two or more existing .rvl files in the current directory.

Figure 14: Reveal Inserter Startup Wizard Displaying .rvl Files



If Reveal Inserter does not find an .rvl file in the current directory, it opens the following version of the Startup Wizard dialog box.

Figure 15: Reveal Inserter Startup Wizard Dialog Box with No Existing .rvl File

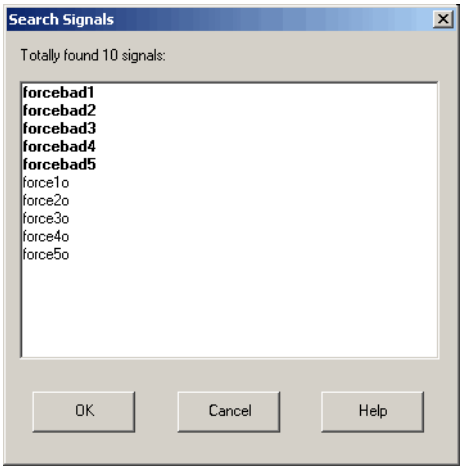


If Reveal Inserter finds a single .rvl in the current directory, it automatically opens it.

Search Signals Dialog Box

The Search Signals dialog box is activated by entering a signal name or pattern in the Signal Search box in the lower left portion of the main Reveal Inserter window and clicking Search to the right of the box.

Figure 16: Search Signals Dialog Box



This dialog box contains the following parameters:

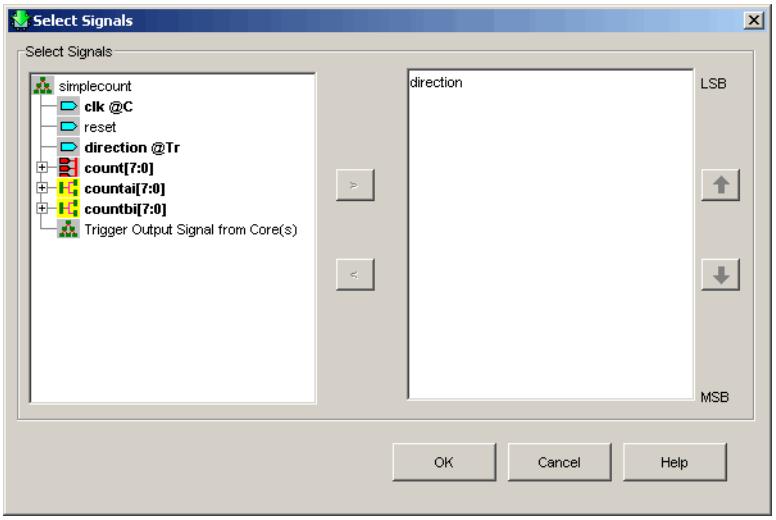
Cancel Closes the dialog box without applying any changes.

Help Displays the online Help topic for this dialog box.

Select Signals Dialog Box

The Select Signals dialog box is activated by double-clicking in the Signals (MSB:LSB) box of a trigger unit line. It enables you to select which signals to include in a trigger unit and how to order them. As an alternative, you can also drag and drop signals from the Design Hierarchy pane to the Trace Data pane in the Trace Signal Setup tab and to the Signals (MSB:LSB) box in the Trigger Signal Setup tab.

Figure 17: Select Signals Dialog Box



This dialog box contains the following parameters:

Select Signals Displays the hierarchy of the signals in the design. It also displays the output signal from the core.

Selected Signals (right pane) Displays the signals that you have selected to include in a trigger unit. You cannot select more than 256 signals.

> Moves a signal or signals from the Select Signals pane to the right pane.

< Moves a signal or signals from the right pane to the Selected Signals pane.

Moves the selected signal up one line. The order of the signals affects how the comparison is performed.

Moves the selected signal down one line. The order of the signals affects how the comparison is performed.

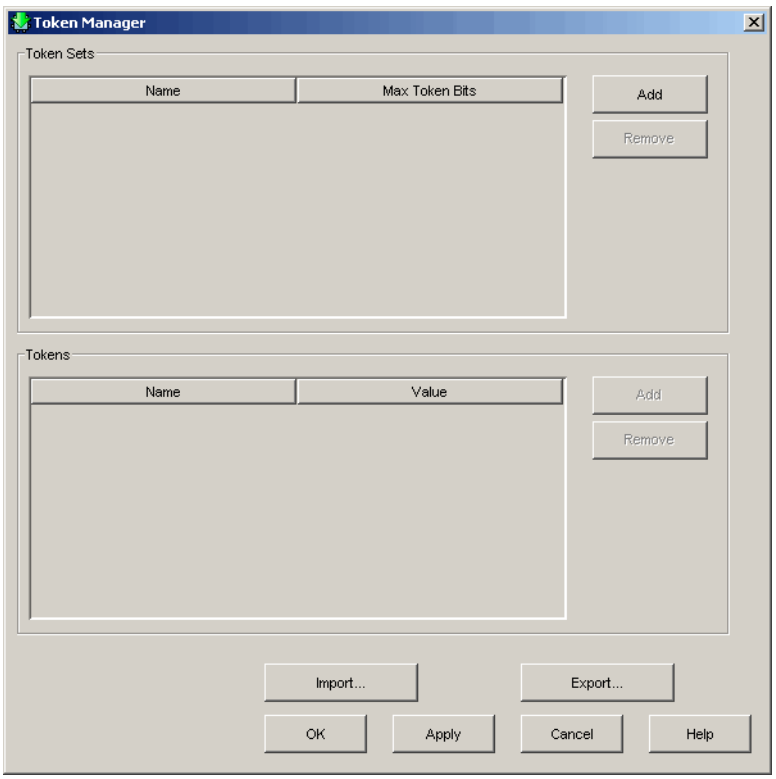
Cancel Closes the dialog box without applying any changes.

Help Displays the online Help topic for this dialog box.

Token Manager Dialog Box

The Token Manager dialog box is activated by the Datasets > Token Set Manager command. It enables you to define and manage token sets, which are groups of tokens that apply to specific buses in Reveal Logic Analyzer.

Figure 18: Token Manager Dialog Box



This dialog box contains the following parameters:

Token Sets Lists all the token sets that you have created for the design.

- Name – Specifies the name of the token set. This name must be unique. It is case-insensitive.
- Max Token Bits – Specifies the maximum bit size of the token value in the token set. The default value is 4.
- Add – Adds a new token set with a default name of TokenSet<*n*>, where <*n*> is a sequential number, starting with 1.
- Remove – Removes the selected token set.

Tokens in <*token_set_name*> Specifies the names and values of the tokens in the selected token set.

- Name – Lists the names of the tokens included in the selected token set. They are the labels of the anticipated bus values that will appear in the Waveform View tab of Reveal Logic Analyzer. The name of each token must be unique. Names are case-insensitive.
- Value – Specifies the anticipated bus values.

- Add – Adds a new token with a default name of Token<*n*>, where <*n*> is a sequential number, starting with 0.
- Remove – Removes the selected token.

You can edit the names and values of tokens directly in the table cell.

Import Opens the Import Tokens dialog box so that you can select a previously created token (.rvt) file, then places the file's tokens and token sets in the Reveal Inserter graphical user interface.

Export Opens the Export Tokens dialog box so that you can specify the name of the token (.rvt) file in which to save the tokens and token sets created in the graphical user interface. The token file is in XML format.

Apply Saves the token sets without closing the dialog box.

Cancel Closes the dialog box without applying any changes.

Help Displays the online Help topic for this dialog box.

Toolbar

The toolbar in the Reveal Inserter window consists of buttons that enable you to perform tasks without using menu commands. To hide or show the input bar, choose **View > Toolbar**.

Figure 19: Reveal Inserter Toolbar

Activates the Create Project dialog box so that you can create a new Reveal Inserter project.

The equivalent command is File > New Project.

Activates the Select Project dialog box so you can select the .rvl file of an existing Reveal Inserter project to open.

The equivalent command is File > Open Project.

Saves a Reveal Inserter project in an .rvl file in the current directory.

The equivalent command is File > Save Project.

Checks the Reveal Inserter logic debug settings to be sure that they are correct.

The equivalent command is Datasets > Design Rule Check.

Activates the About dialog box, which displays copyright information for Reveal Inserter.

The equivalent command is Help > About.

Inserts the selected cores into the design.

The equivalent command is Datasets > Insert Debug.

Menus

The Reveal Inserter graphical user interface includes the File menu, the View menu, the Datasets menu, the Trace menu, the Help menu, and several pop-up menus.

File Menu

The File Menu in the Reveal Inserter window consists of the following commands:

New Project Activates the Create Project dialog box so that you can create a new Reveal Inserter project. This command is equivalent to the  toolbar button.

Open Project Activates the Select Project dialog box so you can select the .rvl file of an existing Reveal Inserter project to open. This command is equivalent to the  toolbar button.

Close Project Closes an open Reveal Inserter project.

Save Project Saves Reveal Inserter project in a .rvl file in the current directory. This command is equivalent to the  toolbar button.

Save As Project Activates the Select Project dialog box so that you can save Reveal Inserter project in an .rvl file in a directory other than the current directory.

Delete Project Deletes a Reveal Inserter project.

Options Activates the Options dialog box, in which you can specify whether the Welcome to Reveal dialog box should appear when you first start Reveal Inserter. If you deselect the Display Welcome to Reveal dialog box option, Reveal Inserter suppresses the appearance of the Welcome to Reveal dialog

box in subsequent activations of the tool. If you select the Display Welcome to Reveal dialog box option, Reveal reinstates the appearance of the Welcome to Reveal dialog box when you start Reveal Inserter. By default, the option is selected.

Recent Project Displays a list of the four most recent projects that you opened.

Exit Closes Reveal Inserter.

View Menu

The View menu in Reveal Inserter window consists of the following commands:

Expand All Design Hierarchy Displays the hierarchy of the whole design when you right-click on the project name, including the ports and nodes in the top module and submodules. When you right-click on a bus name, it displays all the signals in the bus.

Collapse All Design Hierarchy Displays only the buses, ports, top-level signals, and top level of the hierarchy in the design when you right-click on the project name. When you right-click on a bus name, it hides the names of all the signals in the bus.

Datasets Menu

The Datasets menu in Reveal Inserter window consists of the following commands:

Add New Core Adds a new core to the project. Reveal Inserter creates a new core with a default name of <*top_module*>_LA<*number*>, where *top_module* is the name of the top module in the Reveal Inserter project, and *number* is a sequential number. For example, the third core added to the “counter“ project would be named counter_LA3. The core name is case-insensitive—for example, “core_LA0" is the same as “core_la0.” The added cores are displayed in the Debug Datasets pane.

Rename Core Activates the Input dialog box so that you can specify a new name for the designated core.

During the renaming process, Reveal Inserter verifies that:

- The core name begins with a letter and consists of letters, numbers, and underscores (_).
- The core name is not the same as that of any other core.
- The core name is not the same as that of any module or instance in the design.

Remove Core Removes the designated core or cores from the project.

Token Set Manager Activates the Token Manager dialog box so that you can define and manage token sets.

Design Rule Check Checks the accuracy of Reveal Inserter logic debugging settings, namely that:

- The core names begin with a letter and consist of letters, numbers, and underscores (_).
- A core name is not the same as that of any other core.
- The core name is not the same as that of any module already defined in the design.
- The number of cores is between 1 and 15.
- The number of trace signals is between 1 and 512.
- The number of trigger signals is between 1 and 4096.
- The number of trigger units and trigger expressions is between 1 and 16.
- The number of trigger signals in a trigger unit is between 1 and 256.
- A sample clock is specified.
- The sample clock signal is a 1-bit signal already defined in the design.
- A sample enable is specified.
- The sample enable signal is a 1-bit signal already defined in the design.
- The name of the trigger-out signal is given if this signal is enabled.
- The name of the trigger-out signal is not the same as any signal already defined in the design.
- The number of EBRs needed does not exceed the number available.
- The design includes an input signal.
- The syntax of the trigger expressions is correct.
- The trigger expression sequence is less than or equal to the maximum sequence.
- The trigger output signal is specified, if the Enable Trigger Out option is enabled.
- The trigger output signal is not the same as the name of any signal in the design.
- The values of the trigger unit are correct.
- The names of the trigger units and the trigger expressions conform to the guidelines given in the “Trigger Expression and Trigger Unit Naming Conventions” on page 25.
- The bit widths of the token values are the same as the bit widths of the trigger unit signals.

This command is equivalent to the  toolbar button.

Reveal Inserter automatically performs a check before saving the project or inserting the debug logic.

Insert Debug Activates the Insert Debug to Design dialog box so that you can select the cores to insert into the design, specify the name of the .lpf file

to use, and import the .rvl into ispLEVER. This command automatically performs a design-rule check before inserting the debug logic cores.

This command is equivalent to the 

toolbar button.

Trace Menu

The Trace menu in the Reveal Inserter window consists of the following commands:

Group Signals/Buses Groups signals, buses, or both into a bus signal.

UnGroup Signals/Buses Ungroups signals, buses, or both in a bus signal.

Remove Signals/Buses Removes signals, buses, or both from the Trace Signal Setup tab.

Rename Bus Activates the Input dialog box so that you can enter a new name for a bus.

Help Menu

The Help menu in Reveal Inserter window consists of the following commands:

Reveal Inserter Help Opens the online Help for Reveal Inserter.

ispLEVER Help Opens the online Help for all the design tools in ispLEVER.

About Opens the About dialog box, which displays copyright information for Reveal Inserter. This command is equivalent to the  toolbar button.

Debug Datasets Pop-Up Menu

The pop-up menu in the Debug Datasets pane is activated when you right-click on the name of a core or on “Datasets.”

Add New Core Adds a new core to the project.

Remove Removes the designated core or cores from the project.

Rename Activates the Input dialog box so that you can specify a new name for the designated core.

During the renaming process, Reveal Inserter verifies that:

- The core name begins with a letter and consists of letters, numbers, and underscores (_).
- The core name is not the same as that of any other core.
- The core name is not the same as that of any module or instance in the design.

Trace Signal Setup Pop-Up Menu

The popup menu in the Trace Signal Setup tab is activated when you right-click on a bus or a signal. It consists of the following commands:

Group Groups signals, buses, or both into a bus signal. This command is equivalent to the Trace > Group Signals/Buses command.

UnGroup Ungroups the signals in a bus.

Rename Activates the Input dialog box so that you can enter a new name for a bus. This command is equivalent to the Trace > Rename Bus command.

Remove Removes signals, buses, or both from the list of signals and buses in the Trace Signal Setup tab. It is equivalent to the Trace > Remove Signals/Buses command.

Expand All Displays the signals in all the buses in the Trace Data pane.

Collapse All Displays only the buses in the Trace Data pane.

Expand Displays the signals in the selected bus.

Collapse Displays only the selected bus.

When you right-click on a signal, only the Group, Remove, Expand All, and Collapse All commands are available.

Design Hierarchy Pop-Up Menu

The pop-up menu in the Design Hierarchy pane is activated when you right-click on a bus or the name of the project.

Expand All Displays the hierarchy of the whole design when you right-click on the project name, including the ports and nodes in the top module and submodules. When you right-click on a bus name, it displays all the signals in the bus.

Collapse All Displays only the buses in the design when you right-click on the project name. When you right-click on a bus name, it hides the names of all the signals in the bus.

Expand Displays the signals in the selected bus.

3

Reveal Logic Analyzer

Logic analyzers enable you to view signal information to debug design functionality. With external logic analyzers, you connect to pins on a board, set one or more trigger conditions, and sample and view collected data. Internal logic analyzers, such as Reveal Logic Analyzer, depend on additional logic placed into the design for triggering and tracing, then transferring the data to a PC, usually through a JTAG connection, for viewing and analysis.

Reveal Inserter handles the task of inserting debug logic into your design. Before using Reveal Logic Analyzer, you must use Reveal Inserter to allow debug access.

Reveal Logic Analyzer enables you to configure trigger settings and extract information from a programmed device through the JTAG ports. It interfaces directly to the Reveal cores in the design. You can set up triggers, select capture modes, and run or stop the triggers. Reveal Logic Analyzer displays the data captured on the silicon according to the settings that you specify.

Reveal Logic Analyzer's graphical user interface enables you to view the trace data of a signal or bus in a waveform viewer.

Although an evaluation board is normally required to run Reveal Logic Analyzer, Reveal Logic Analyzer includes a demonstration design that you can run without the evaluation board so that you can learn how to use the tool.

Reveal Logic Analyzer requires the ispVM System programming software to configure the specified device. The acquired data is displayed in the waveform viewer.

You can export waveform data to a value change dump (.vcd) file, which can be imported by such third-party tools as ModelSim or Active-HDL. You can also output a file in ASCII tabular format for exporting the data into other tools such as Excel.

About Reveal Logic Analyzer

This section introduces some of the key features of Reveal Logic Analyzer: the devices that it supports, the steps in its design flow, its inputs, and its outputs.

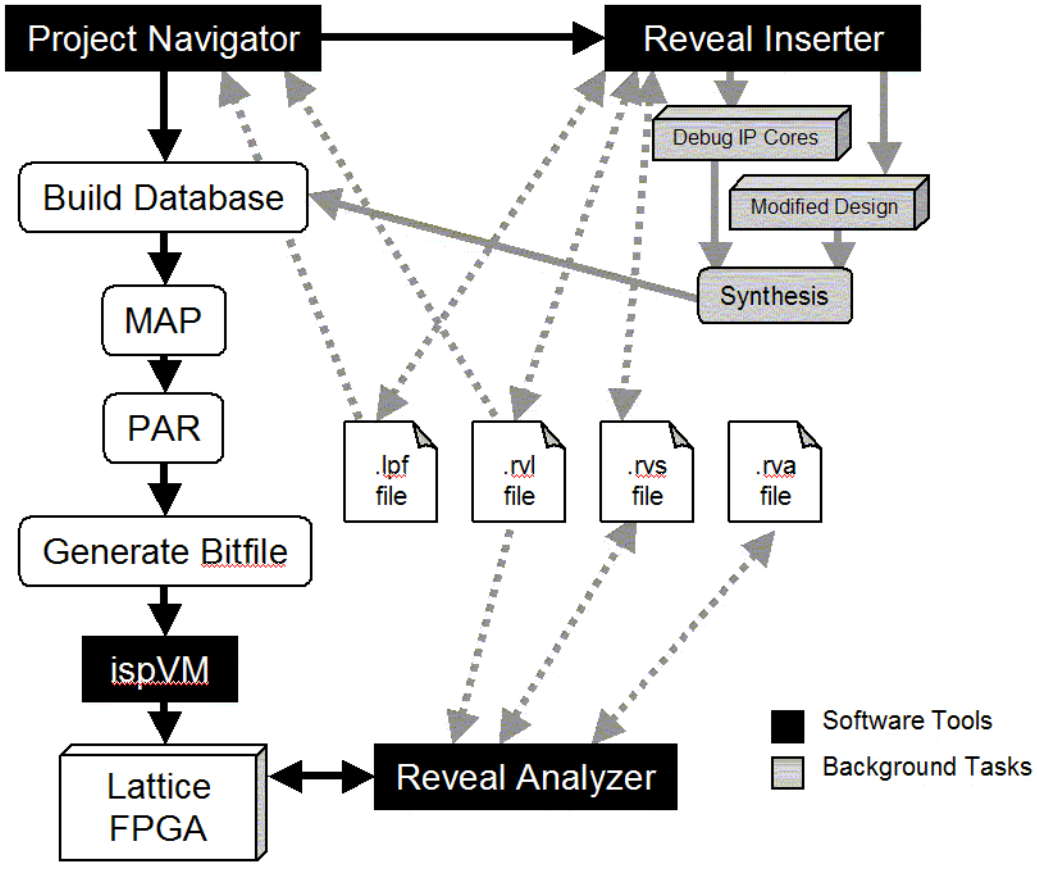
You can use Reveal Logic Analyzer with all FPGAs and with MachXO devices of 1200 or more LUTs.

You can run Reveal Logic Analyzer on both the Windows and Linux operating systems on the PC.

Reveal On-Chip Debug Design Flow

The following figure shows the Reveal insertion and logic analysis design flow.

Figure 20: Reveal Design Flow



Before accessing Reveal Logic Analyzer, you must install a Lattice Semiconductor or USB download cable and a power supply between your computer and the evaluation board. Refer to “Connecting to the Evaluation Board” on page 74 for information on this procedure. See the ispVM System Help for detailed information on these cables.

You do not need to install a cable and a power supply if you want to run the demonstration design that comes with Reveal Logic Analyzer so that you can learn how to use the tool. Click on **File > Open Demo** in Reveal Logic Analyzer to access this design.

You must also have a Project Navigator design that has had on-chip debug logic inserted by the Reveal Inserter software.

The general steps involved in performing a logic analysis are the following:

1. Start Reveal Inserter.
2. Configure the trace and trigger signal settings in Reveal Inserter.
3. Insert the debug logic with Reveal Inserter.
4. Build the database in the ispLEVER Project Navigator.
5. Map, place, and route the design.
6. Generate the bitstream data or JEDEC file.
7. Set up a cable connection.
8. Download the design onto the device by using ispVM System.
9. Start Reveal Logic Analyzer.
10. Create a new Reveal Logic Analyzer project or open an existing one.
11. Configure the trigger settings for each core in each device that you want to use to perform logic analysis of the design.
12. Click the Run button (or select Device > Run Selected LAs) to perform the logic analysis, and wait for the design to trigger and download the trace information into Reveal Logic Analyzer from the board.
13. View the resulting waveforms for each core.
14. Optionally, you can export the waveform data for each core in a value change dump (.vcd) file for use in third-party tools or in an ASCII-format text (.txt) file.

Inputs

Reveal Logic Analyzer requires the following as input:

- A design from Project Navigator
- A Reveal Logic Analyzer settings (.rvs) file, which is output by Reveal Inserter or Reveal Logic Analyzer. It contains all the dynamically changeable trigger settings, such as trigger unit operators, trigger unit values, and any trigger expressions.
- An existing Reveal Inserter project (.rvl file), which contains the connections for each core and all the static settings of the debugging

logic. The information in this file is statically set in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.

- A Reveal Logic Analyzer project (.rva) file, which is the project file output by Reveal Logic Analyzer in a previous session. It contains the information used by Reveal Logic Analyzer, such as window settings, waveform trace signal positions, radixes, markers, and signal colors.
- Optionally, a scan chain configuration (.xcf) file, which is generated by ispVM System for programming devices in a JTAG daisy chain. The .xcf file contains information about each device, the data files targeted, and the operations to be performed. The .xcf file must reside in the Project Navigator project directory for Reveal Logic Analyzer to use it.
- An .xcf file is required as input only if you will be programming devices in a JTAG daisy chain. It is not required if you will be programming a single device.

Note

Only one device can be debugged in a JTAG daisy chain.

Outputs

Reveal Logic Analyzer generates the following files:

- A Reveal Logic Analyzer project (.rva) file, which contains the information such as window settings, waveform trace signal positions, radixes, markers, and signal colors. This file is also an input file when you re-open a project that you previously saved.
- A Reveal Logic Analyzer trace (.trc) file, which contains the waveform information acquired from previous runs of Reveal Logic Analyzer. When you first open Reveal Logic Analyzer, the waveform displays this information until you press the Run button. If the debug signals have been changed from a previous Reveal Logic Analyzer run, incorrect information is displayed in the waveform when it is first opened. Once a run has been completed, the waveform contains valid information with the changed debug configuration.
- Optionally, a value change dump (.vcd) file, in which you can export waveform data for display in third-party tools such as ModelSim and Active-HDL. The .vcd file is an ASCII file containing header information, variable definitions, and variable value changes. Its format is specified by the IEEE 1364 standard.
- Optionally, an ASCII-format text (.txt) file, in which you can export waveform data for display in third-party tools such as ModelSim and Active-HDL. The .txt file is in a simple ASCII character-tab-delimited format. It includes a header line with the signal names, then each line contains the value for each signal, one line per each sample clock.

Inserting the Debug Logic

Before performing logic analysis with Reveal Logic Analyzer, you must use Reveal Inserter to generate the debug logic and insert it into your design. You must also set up the trace and trigger signals to be used in Reveal Inserter.

Building the Database

After you insert the debug logic in the Reveal Inserter, you return to the ispLEVER Project Navigator to build the database.

To build the design project database:

- Double-click the **Build Database** process in the ispLEVER Project Navigator.

During this process, Project Navigator invokes the synthesis tool.

Mapping, Placing, and Routing the Design

Once you build the database, you map, place, and route the design.

To map, place, and route the design:

1. Double-click the **Map Design** process in the ispLEVER Project Navigator.
2. Double-click the **Place & Route Design** process in the ispLEVER Project Navigator.

All core clock pins must be located and driven by valid signals for successful hardware debugging.

Generating a Bitstream or JEDEC File

Now you generate a bitstream or JEDEC file to download onto the device.

To generate a bitstream or JEDEC file:

- Double-click the **Generate Bitstream Data** or **Generate Data File (JEDEC)** process in Project Navigator.

  This process creates a .bit .jed file for FPGAs that is ready for downloading into the device.

Connecting to the Evaluation Board

Reveal Logic Analyzer requires that a Lattice Semiconductor parallel port cable or USB download cable and a power supply be installed between your computer and the evaluation board so that you can program the device with ispVM System.

To connect the evaluation board to your computer:

1. Install a driver for the download cable, if it has not been previously installed.
2. Reboot your computer, if the driver was not previously installed.
3. Attach the parallel port or USB ispDOWNLOAD cable to the parallel port or USB port of your system.
4. Plug in the AC adapter to a wall outlet, and plug the other end into the power jack provided on the evaluation board.

   Note

   You should follow the handling and power-up advice provided in the Lattice Semiconductor device evaluation board documentation when using the evaluation board.

5. In Project Navigator, select **Tools > ispVM System** or click the  button on the toolbar.
6. Select **Options > Cable and IO Port Setup**.
7. Click **Auto Detect**, then click **OK**.
8. Attach the JTAG connector ispDOWNLOAD cable to the appropriate JTAG programming header of the evaluation board. See the device evaluation board documentation for details.

Refer to the ispVM System Help for more information about your cable connection.

Downloading a Design onto the Device

To download a design onto the device, use ispVM System. This process creates a scan chain configuration (.xcf) file. An .xcf file is not necessary to use Reveal Logic Analyzer, unless you will be programming or debugging devices in a daisy chain (see “Programming and Debugging Devices in a Daisy Chain” on page 79 for more information). Reveal Logic Analyzer derives the information in the downloaded design directly from the device on the board.

To download the design onto the device:

1. If you do not already have ispVM System open, choose **Device > ispVM System** in Reveal Logic Analyzer, or click the  button on the Reveal Logic Analyzer toolbar.

2. In the ispVM System interface, select **File > New**.

   A new chain configuration window appears.

3. Choose **ispTools > Scan Chain** or click the Scan toolbar icon.

   ispVM detects the device that you are using and adds it to the list.

4. Select the first device in the New Scan Configuration Setup list.
5. In the popup box labeled Multi Match Device's ID List, select the appropriate device.
6. Highlight the selected device and choose **Edit Device** from the pop-up menu.
7. In the Device Information dialog box, click the **Select** button of the Device section to open the Select Device dialog box.
8. In the Select Device dialog box, do the following:
   a. In the Device Family box, select the appropriate device family.
   b. In the Device box, select the device, as appropriate for your revision of the standard evaluation board.
   c. In the Package box, select the appropriate package.
   d. Click **OK**.
9. Click the **Browse** button of the Data File section.
10. Select the <*design_name*>**.bit** or **.jed** file, and click **Open**.
11. In the Operation box, select **Fast Program**, if it is not already selected.
12. Click **OK** to close the Device Information dialog box.
13. Choose **Project > Download**, or click the **GO** button on the toolbar.

    After a few moments, the download and programming activity will end. A green PASS button appears in the New Scan Configuration Setup dialog box.

14. Select **File > Save As** to save the configuration setup as an .xcf file.

    The .xcf file must reside in the Project Navigator project directory for Reveal Logic Analyzer to use it.

    Note

    An .xcf file is not required unless you will be programming or debugging devices in a daisy chain. See "Programming and Debugging Devices in a Daisy Chain" on page 79 for more information.

15. In the File Name box in the dialog box that appears, type in <*design_name*>**.xcf**, and click **Save**.
16. Choose **File > Exit** in the LSC ispVM System window.

See the ispVM System Help for detailed instructions on the downloading process.

Starting Reveal Logic Analyzer

As noted earlier, before accessing Reveal Logic Analyzer, you must install a Lattice Semiconductor or USB download cable and a power supply between your computer and evaluation board. Refer to the ispVM System Help for detailed information on these cables.

You do not need to install a cable and a power supply if you want to run the demonstration design that comes with Reveal Logic Analyzer so that you can learn how to use the tool. Click on **File > Open Demo** to access this design.

You can start Reveal Logic Analyzer from Project Navigator or as a stand-alone tool.

Note

Reveal Inserter generates a “signature” or tracking mechanism each time that debug logic is inserted into the design. The signature is placed into the project file and into the debug logic. Reveal Logic Analyzer reads this signature to ensure that the FPGA has been programmed with the latest debug logic. Reveal Inserter generates a new signature every time the .rvl file is written, and Reveal Logic Analyzer checks this signature each time that it runs the design. If you save the project in Reveal Inserter without re-running the Project Navigator flow, Reveal Logic Analyzer issues an error message, even if the debug logic was not changed.

Starting Reveal Logic Analyzer from Project Navigator

To start Reveal Logic Analyzer:

You can start Reveal Logic Analyzer from the ispLEVER Project Navigator on both Windows and Linux.

1. On Linux only, type the following on the command line to activate Project Navigator:

   <*ispLEVER_install_path*>/ispcpld/bin/**ispgui**

2. Choose **Tools > Reveal Logic Analyzer** from the ispLEVER Project Navigator or click on the  icon.

   The Reveal Logic Analyzer Startup Wizard dialog box appears.

3. Click **OK**.

   The New Project dialog box now appears.

4. Click **Cancel**.

5. Follow the instructions in “Creating a New Reveal Logic Analyzer Project” on page 77 to select the cable type and create a Reveal Logic Analyzer project.

Starting Reveal Logic Analyzer as a Stand-Alone Tool

You can start Reveal Logic Analyzer as a stand-alone tool on both Windows and Linux.

To start Reveal Logic Analyzer as a stand-alone tool on Windows:

- In the Microsoft Windows Start menu, select **Programs > Lattice Semiconductor > Accessories > Reveal Logic Analyzer**.

  The Reveal Logic Analyzer window opens.

To start Reveal Logic Analyzer as a stand-alone tool on Linux:

- Type the following on the command line:

  <*ispLEVER_install_path*>/ispcpld/bin/**revealla**

  The Reveal Logic Analyzer window opens.

Running a Demonstration Design

Reveal Logic Analyzer provides you with a simple pre-existing counter design and simulation data so that you can learn how to use the tool without having your computer connected to an evaluation board.

To run the demonstration design:

1. Start Reveal Logic Analyzer.
2. Choose **File > Open Demo**.
3. Configure the trigger settings in the Trigger Signal Setup tab.
4. Choose **Device > Run Selected LAs** or click  .

The trace buffer data now appears in the Waveform View tab in waveform format.

- “Selecting a Reveal Logic Analyzer Core Window” on page 81

Creating a New Reveal Logic Analyzer Project

In order to create a new Reveal Logic Analyzer project, you must first have available a Project Navigator design directory with a Reveal project (.rvl) file generated by Reveal Inserter.

To open a new Reveal Logic Analyzer project:

1. Choose **Device > Connection Setup**.

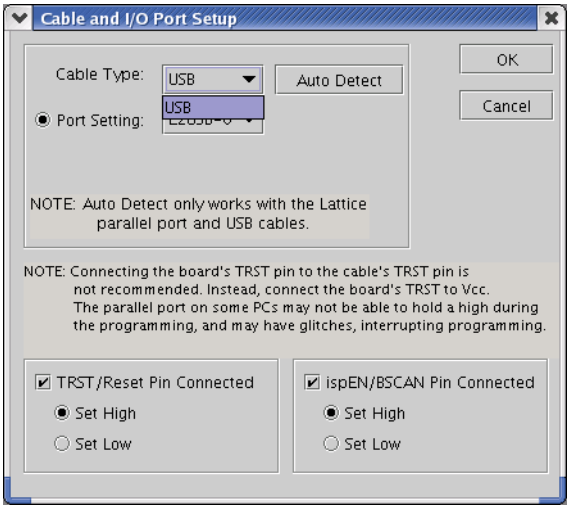
   The Cable and I/O Port Setup dialog box appears.

2. In the pulldown menu in the Cable Type box, choose the cable type or click **Auto Detect**.

Note

Only the USB cable is available on Linux in this release, as shown in Figure 21.

Figure 21: Cable and I/O Port Setup Dialog Box in Linux



3. Deselect the **TRST/Reset Pin Connected** option and the **ispEN/BSCAN Pin Connected** option.
4. Click **OK**.
5. In the Reveal Logic Analyzer window, choose **File > New** to open the New Project dialog box.
6. In the Project Name box, enter the name of the project.
7. In the Project Directory box, browse to the directory in which you want to save the Reveal Logic Analyzer project.

   It is recommended that you save the Reveal Logic Analyzer project files in the project directory for the design into which the debug logic has been inserted.
8. If you will be programming devices in a JTAG daisy chain, choose the name of the .xcf file from the drop-down menu in the **XCF File** box. See “Programming and Debugging Devices in a Daisy Chain” on page 79.

   The .xcf file must reside in the Project Navigator project directory for Reveal Logic Analyzer to use it.

Note

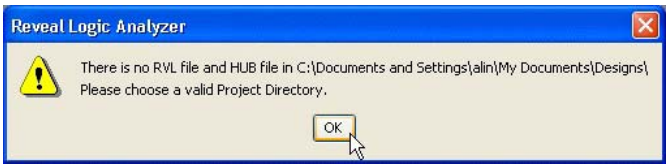
Only one device can be debugged in a JTAG daisy chain.

9. Click **Next**.

   Reveal Logic Analyzer verifies that the project directory includes at least one .rvl file and one .hub file. If the project directory does not include

these files, it issues the warning message shown in Figure 22. You can then choose a valid project directory.

Figure 22: .rvl file and .hub File Warning Dialog Box

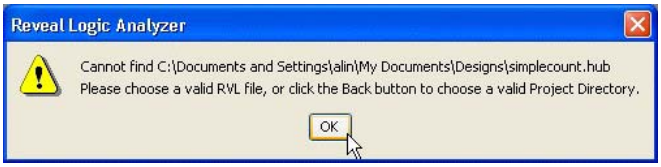


When the tool has verified that the project directory includes at least one .rvl file and one .hub file, the Device Information dialog box appears.

10. From the drop-down menu in the Inserter File (.rvl) box, select the name of the Reveal Inserter project file.
11. Click **Finish**.

    On the basis of the settings in the .rvl file, Reveal Logic Analyzer now verifies that the project directory contains the correct design .hub file and creates the new project. If the project directory does not include this file, it issues the warning message shown in Figure 23. You can then either choose a valid .rvl file or click the **Back** button to choose a valid project directory.

Figure 23: .rvl File Warning Dialog Box



Programming and Debugging Devices in a Daisy Chain

In ispVM System, you can program multiple devices in a JTAG daisy chain configuration at the same time. An .xcf file is required if you will be programming devices in a daisy chain at the same time. The .xcf file must reside in the Project Navigator project directory for Reveal Logic Analyzer to use it.

Reveal Logic Analyzer supports the debugging of a device in a JTAG daisy chain, but it is subject to the following limitations:

- You can only debug one device at a time in a daisy chain.
- Only one of the bitstreams in a daisy chain can include a Reveal core.

To debug a device with a Reveal core in a daisy chain in Reveal Logic Analyzer:

- In the Device Information dialog box in ispVM, set the Operation box for the devices without the Reveal core to **Bypass**. Set the Operation box for the device with the Reveal core to **Fast Program**.
- In the XCF File box in the New Project dialog box in Reveal Logic Analyzer, choose the name of the .xcf file to be used for debugging from the drop-down menu.

Opening an Existing Reveal Logic Analyzer Project

To open an existing project, you must have available a Reveal Logic Analyzer project (.rva) file from a previous Reveal Logic Analyzer session.

Opening an Existing Project

You can open an existing in Reveal Logic Analyzer by using the File menu to open a dialog box.

To open an existing Reveal Logic Analyzer project:

1. Choose **File > Open**.
2. In the **File Name** field in the Open Project dialog box, select the .rva file to open.
3. Click **Open**.

Opening a Recently Opened Project

If the desired .rva file has been recently opened, you can open it directly from the File menu.

To open a recently opened .rva file directly from the File menu:

- Choose **File > Recent Files > *filename*** from the list of the four most recently opened files near the bottom of the File menu.

Selecting a Reveal Logic Analyzer Core Window

After you have created a project, each Reveal core in the design will have a Reveal Logic Analyzer window available to set triggers and view captured data. To select a Reveal Logic Analyzer window, click on the appropriate core in the Datasets window pane in the upper left area.

Note

The darker gray background in certain fields in the Trigger Signal Setup tab indicates that you can change these values only in Reveal Inserter. You cannot change them in Reveal Logic Analyzer.

To display a Reveal Logic Analyzer core window:

- Choose **Window > Show LA Window > Device<*number*> > Device<*number*>_LA<*number*>**.

The LA window appears with the Trigger Signal Setup and Waveform View tabs.

Setting Up the Trace Signals

Although you can add trace signals only in Reveal Inserter, you can set their order, group and ungroup them, set radixes for them, set colors, hide extraneous signals, or make hidden signals visible by using the Waveform View tab.

Grouping Trace Signals into a Bus

You can group trace signals into buses, and you can use two methods of nesting buses.

To group trace signals into a bus:

1. In the Waveform View tab, shift-click the left mouse button to select the signals in the waveform.
2. Choose **Bus/Signal > Group Into Bus**, or right-click, and in the pop-up menu click **Group Into Bus**.

To nest a bus, you can either:

- Select the bus and drag it to a location under the desired bus.
- Select signals already grouped into a bus to form a bus nested under that bus.

Ungrouping Trace Signals from a Bus

To ungroup trace signals from a bus:

1. Highlight the grouped signals in the waveform in Reveal Logic Analyzer's Waveform View tab.
2. Click **Bus/Signal > Ungroup From Bus** or right-click, and in the pop-up menu, click **Ungroup From Bus**.

Setting the Trace Bus Radix

You can set the radix of a trace bus or buses displayed in the Trace Bus/ Signal pane of the Waveform View tab. You can choose a binary, octal, decimal, or hexadecimal radix. You can also use token sets to set the bus radix.

Setting the Trace Bus Radix by Using the Menu

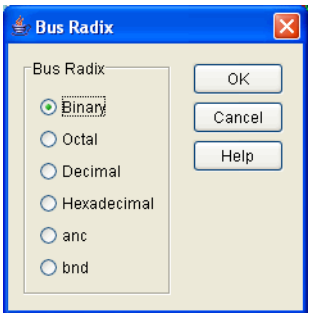
You can set the bus radix of a signal bus or multiple signal buses by using menu commands in the Reveal Logic Analyzer window.

To set the bus radix of a signal bus by using menu commands:

1. Highlight the bus in the Trace Bus/Signal pane of the Waveform View tab.
2. Right-click and choose **Set Bus Radix** from the drop-down menu to display the Bus Radix dialog box.
3. Choose **Binary**, **Octal**, **Decimal**, or **Hexadecimal**, or <*token_set*>.

   If you have defined a token set and the token bit size is the same as the bus width, user-defined token sets will be displayed so that you can choose a token set, if desired. In the following figure, "anc" and "bnd" are token sets.

Figure 24: Token Sets in the Bus Radix Dialog Box



4. Click **OK**.

To set the trace bus radix of multiple signal buses by using menu commands:

1. Select the buses for which to set the radix:
   a. To select multiple contiguous buses, select the first bus with the left mouse button, press **Shift**, then select the last bus with the left mouse

button. All the buses in between the first and last bus are now highlighted.

b. To select multiple non-contiguous buses, press **Control** and select the individual buses with the left mouse button. The individual buses are now highlighted.

You can also select signals, but you must select at least one bus to open the Set Bus Radix dialog box. The Set Bus Radix command is disabled if you select only signals.

2. Right-click and choose **Set Bus Radix** from the drop-down menu to display the Bus Radix dialog box.
3. Choose **Binary**, **Octal**, **Decimal**, or **Hexadecimal**, or <*token_set*>.

   The default is Binary.
4. Click **OK**.

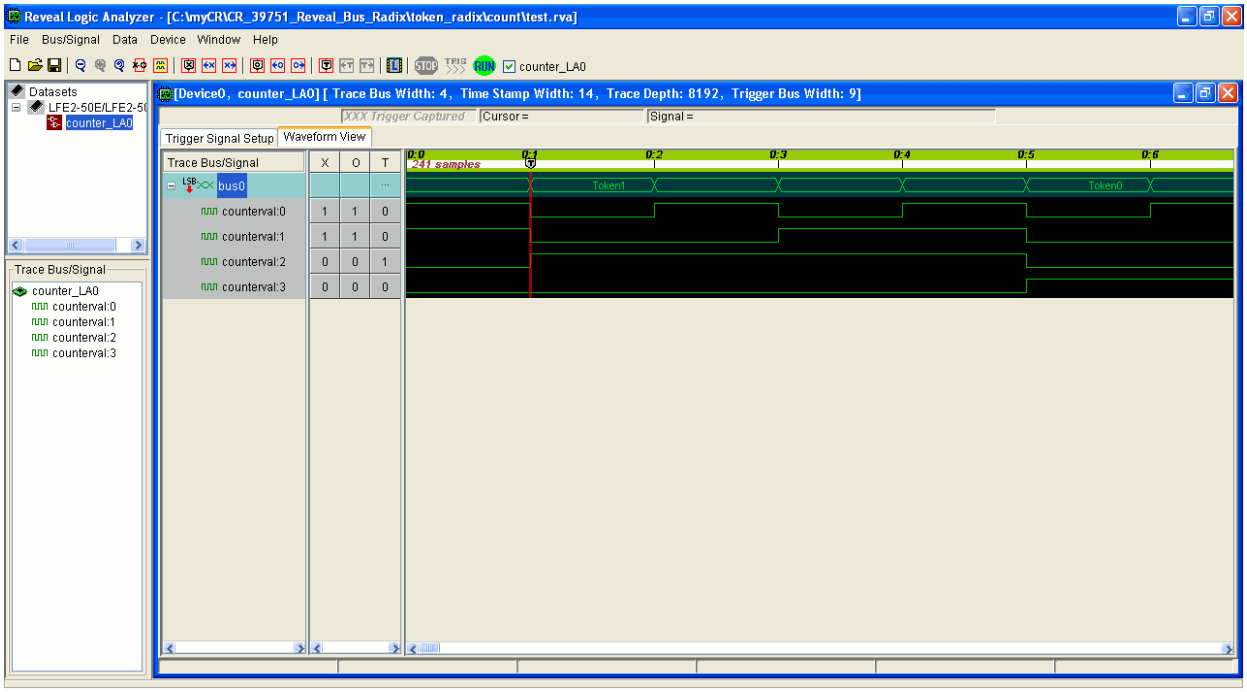
Setting the Trace Bus Radix by Using Token Sets

See the Using Tokens section of the Reveal Inserter online Help for information about setting a trace bus radix by using token sets.

Note

If you do not define all the tokens possible in a token set in Reveal Inserter, the waveforms in Reveal Logic Analyzer will only show the values for the defined tokens. For example, if you specify only two tokens, Token0 and Token1, out of the sixteen tokens possible in a 4-bit token set, only the values for Token0 and Token1 will be shown on the resulting waveforms, as shown in the following figure.

Figure 25: Bus Values for Incomplete Token Set



Renaming Trace Buses

You can rename a trace bus.

To rename a trace bus:

1. In the Trace Bus/Signal pane in the Waveform View tab, right-click on the trace bus that you want to rename and choose **Rename** from the pop-up menu.
2. Backspace over the existing name and type in the new name.

Setting Bus Bit Order

In the Trace Bus/Signal pane of the Waveform View tab, there is a notation to the left of each bus name indicating which bit in the bus is the most significant bit (MSB) or the least significant bit (LSB). You can change this order.

To set the bus bit order:

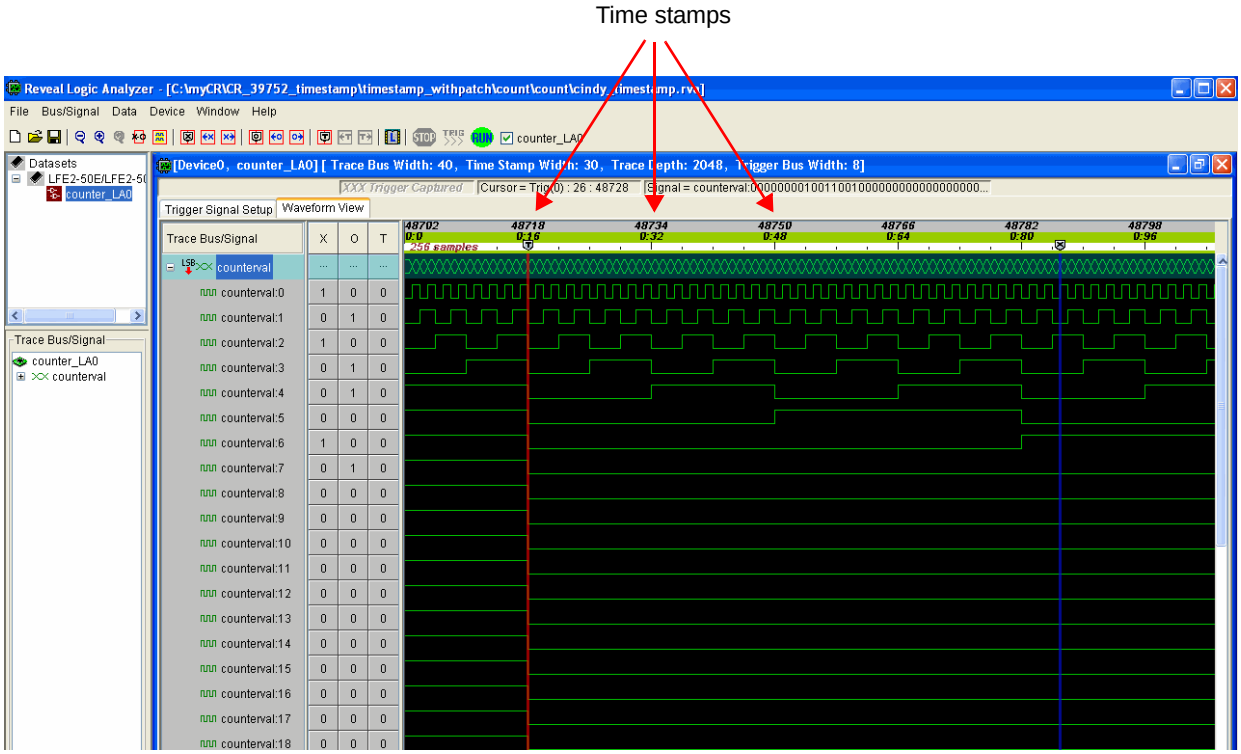
1. In the Trace Bus/Signal pane of the waveform display in the Waveform View tab, right-click on the bus of interest in the Waveform View tab and choose **Set MSB/LSB** from the pop-up menu.
2. From the pop-up menu, choose **MSB** to make a bit the most significant bit or **LSB** to make a bit the least significant bit.

Adding Time Stamps to Trace Samples

In the Reveal Inserter, you can optionally specify a sample clock count value to be stored with each trace sample to indicate the sample count clock value at which the sample was captured. This count is extra data (bits) captured into the trace buffer that increase the trace buffer's width. This time stamp enables you to see how many sample clock intervals have elapsed between data captures when you use a sample enable. It is useful in some cases when it is necessary to know if you captured the right data. A time stamp is also useful when you try to synchronize data between multiple cores, off-chip data, or both. For example, if you trigger two cores at the same time, you can use the time stamps on the trace samples to calculate how the data between the cores compares.

The following figure gives examples of time stamps.

Figure 26: Time Stamps in the Waveform Window



See Adding Time Stamps to Trace Samples in the Reveal Inserter online Help for information on adding time stamps to trace samples in Reveal Inserter.

Making Trace Signals Invisible

To make trace signals invisible:

1. Highlight the signal in the Trace Bus/Signal pane of the Waveform View tab and right-click.
2. In the pop-up menu, select **Hide**.

Making Trace Signals Visible

To make trace signals visible:

- In the Trace Bus/Signals pane of the Waveform View tab, choose **Bus/Signal > Unhide All** or right-click and choose **Bus/Signal > Unhide All**.

Locating a Trace Signal

You can locate a trace signal in the Waveform View display.

To locate a trace signal:

- In the Trace Bus/Signal pane in the extreme left of the Reveal Logic Analyzer window (not the pane of the same name in the Waveform View tab), select the trace signal of interest, right-click, and choose **Locate in Waveform View** from the pop-up menu.

Reveal Logic Analyzer now highlights the selected signal or bus in turquoise in the Waveform View tab. If a signal is hidden in the Waveform View tab, this command makes it visible.

Setting Trace Signal Colors

You can set colors of the individual trace signals in the Waveform View tab to make it easier to view the waveform of a specific signal.

To set the color of a trace signal:

1. In the Trace Bus/Signals pane of the Waveform View tab, highlight the signal whose color you want to change.
2. Right-click, and in the pop-up menu, choose **Set Colors** to display the Colors dialog box.
3. To set the foreground color, click the **Foreground Use** button, then double-click the palette button to display the Color Chooser dialog box.
4. To set the background color, click the **Background Use** button, then double-click the palette button to display the Color Chooser dialog box.
5. In the Color Chooser dialog box, choose the desired color for either the foreground or background.
6. Click **OK**.

Moving Signals in the Waveform View Tab

You can move signals in the Trace Bus/Signal pane of the Waveform View tab.

To move a signal in the Trace Bus/Signal pane of the Waveform View tab:

1. In the Trace Bus/Signal pane of the Waveform View tab, hold down the left mouse button and highlight the signal that you want to move.
2. Drag and drop the signal to the desired location on the list of signals.

Setting Up the Trigger Signals

Before you perform logic analysis, you must define the conditions under which the trigger will start or stop the collection of data on the trace signals specified in Reveal Inserter. You must define these triggers for each core. Use the Trigger Signal Setup tab of the Reveal Inserter window to specify the trigger units and trigger expressions that start the collection of the sample data for the selected core. In Reveal Logic Analyzer, you cannot add or remove new trigger units or trigger expressions, but you can change the values and operators in the trigger units and trigger expressions. In addition, you can disable a trigger expression from being used by deselecting the checkbox to the left of the trigger expression name. You must make sure in Reveal Inserter that all signals that you might want to trigger on are included in the trigger units. In addition, you may want to create several trigger expressions ahead of time.

Renaming Trigger Units

You can rename a trigger unit.

To rename a trigger unit:

- Double-click in the appropriate box in the Name column of the Trigger Unit section of the Trigger Signal Setup tab, backspace over the existing name, and type in the new name.

Setting Up Trigger Units

All signals for a trigger unit must be defined in Reveal Inserter. You cannot change them in Reveal Logic Analyzer.

To set up a trigger unit:

1. If you want to change the default name of the trigger unit, backspace over the default name in the Name column and type the new name.
2. If you want to add, change, or remove the signals in the Signals (MSB:LSB) column, you must add, change, or remove them in Reveal Inserter. You cannot add, change, or remove signals in Reveal Logic Analyzer.

   If you move the cursor over a trigger-unit line in the Signals (MSB:LSB) box, the software displays a complete list of the signals in that trigger unit.
3. In the **Operator** column, set the comparators for the trigger condition. You can choose from the following states:
   - == equal to
   - != not equal to
   - > greater than
   - >= greater than or equal to
   - < less than
   - <= less than or equal to

- Rising edge – compares on the rising edge of the clock
- Falling edge – compares on the falling edge of the clock
- Serial compare – compares until the trigger condition is met. For example, if the trigger condition is 10011, the serial compare option looks for a 1 on the first clock, a 0 on the next clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only if those five conditions are met in those five clock cycles will the serial compare output be active.

  Other operators cannot be changed to a serial compare, and a serial compare cannot be changed to another operator in Reveal Logic Analyzer. These can only be changed in Reveal Inserter.

  The serial comparator is available only when a single signal is listed in the Trigger Unit signal list. If you choose this option, you must choose Binary in the Radix box.

The default comparator is == (equal to).

4. In the **Radix** column, select a radix from the drop-down menu to set the radix of the trigger bus value given in the Value box. You can choose one of the following:
   - Binary. This is the default. You must choose Binary if you selected “Serial compare” as a comparator.
   - Octal
   - Decimal
   - Hexadecimal
   - <*token_set_name*>. To select <*token_set_name*>, you must have created token sets in Reveal Inserter. See the Reveal Inserter online Help for information on this procedure.
5. In the **Value** column, enter the comparison value.

   This value is the pattern of highs and lows that you want on the trigger unit that will initiate collection of the trace data. The default is binary, unless you selected <*token_set_name*> in the Radix column.

   If you selected <*token_set_name*> in the Radix column, a drop-down menu opens in the Value column. This menu lists all the tokens that you entered in the Token Manager dialog box for the chosen token set. Select any name.

   You can use ”x” for a don’t-care value in the Value column if you selected Binary, Octal or Hexadecimal in the Radix column and if you selected the ==, !=, or serial compare operators in the Operator column.

Renaming Trigger Expressions

You can rename a trigger expression.

To rename a trigger expression:

- Double-click in the appropriate box in the Name column of the Trigger Expression section of the Trigger Signal Setup tab, backspace over the existing name, and type in the new name.

Setting Up Trigger Expressions

You must set up the initial trigger expressions in Reveal Inserter, but you can change their names and some values in Reveal Logic Analyzer. You can also enable or disable trigger expressions in Reveal Logic Analyzer. However, you cannot change the sequence depth, the maximum sequence depth, or the maximum event counter of the trigger expressions in Reveal Logic Analyzer.

To set up a trigger expression:

1. To enable a trigger expression, click the checkbox in the **Enable** column.
2. In the **Expression** box, enter the names of the trigger units that you want to use and the operators that you want to use to connect them.

   You can use the following operators to connect trigger units:

   - & (AND)
   - | (OR)
   - ^ (XOR)
   - ! (NOT)
   - Parentheses
   - THEN
   - NEXT
   - # (count)
   - ## (consecutive count)

   See “Trigger Expression” on page 105 for information on these operators.

   Reveal Logic Analyzer checks the syntax and displays the syntax in red font if it is erroneous.

   The setting in the **Sequence Depth** box is set by the software, so it is read-only. See “Trigger Expression” on page 105 for more information on this parameter.
3. If you want to change the setting in the **Max Sequence Depth** box, you must change it in Reveal Inserter; you cannot change it in Reveal Logic Analyzer. See the Reveal Inserter online Help for details. The number of sequences in the Trigger Expression box cannot exceed the number specified in the Max Sequence Depth box.
4. If you want to change the setting in the **Max Event Counter** box, you must change it in Reveal Inserter; you cannot change it in Reveal Logic Analyzer. See the Reveal Inserter online Help for details.
5. Specify whether the final trigger occurs when one or all of the conditions specified by the trigger expressions is met before trace data is captured:

   - AND All indicates that the conditions specified by all the trigger expressions must be met before the trace data is captured.
   - OR All indicates that the conditions of one of the trigger expressions must be met before the trace data is captured. This option is the default.

Only trigger expressions whose checkboxes are enabled are included in the AND or OR.

Setting Trigger Options

You can set a number of options to control the triggering.

To set trigger options in Reveal Logic Analyzer:

1. If you want to add a counter to the output of the final trigger, do the following:
   a. Select the **Final Event Counter** checkbox in the lower left portion of the Trigger Signal Setup tab.
   b. In the drop-down menu next to Event Counter, select the number of times that the conditions specified by AND All or OR (whichever you have selected) must occur before trace data capture begins. The lowest value of the counter is 1 time, and the maximum value is the value set in Reveal Inserter. The default value is 1 time. Leaving this option unselected is equivalent to setting the counter to a value of 1 time.
2. In the **Samples Per Trigger** box, select the number of samples to collect per trigger. The minimum is 16. The maximum is the trace buffer depth chosen in Reveal Inserter when the core is generated. The values available in the Samples Per Trigger box also change according to the number of triggers. If the number of triggers is set higher than 1, the samples per trigger multiplied by the selected number of triggers cannot exceed the trace buffer depth. Reveal Logic Analyzer adjusts the Samples Per Trigger value, if necessary. The default number of samples per trigger is the sample buffer depth. For example, if the sample buffer depth is 2048, the default sample per trigger is 2048.
3. In the **Number of Triggers** box, select the trace buffer depth divided by the samples per trigger. The trace buffer depth is specified by the setting of the Buffer Depth parameter in the Trace Signal Setup tab in Reveal Inserter. The default is 1.
4. In the **Trigger Position** field, select **Pre-selected Position** or **User-selected Position**. See “Trigger Position” on page 108 for information on these options.

Performing Logic Analysis

After you have configured trigger settings in the Trigger Signal Setup tab, you can perform a logic analysis using up to 15 cores, depending on the number of cores you created in Reveal Inserter, and view the trace buffer data in waveform format in the Waveform View tab.

To perform logic analysis:

1. In the <*design_name*>_LA<*core_number*> check box in the toolbar, select the cores on which to perform logic analysis.
2. Choose **Device > Run Selected LAs** or click  on the toolbar.

Reveal Logic Analyzer first configures the cores selected for the correct trigger condition, then waits for the trigger conditions to occur. Once the specified trigger has occurred, the data is downloaded to the PC. The resulting waveforms appear in the Waveform View tab.

If a core in your design contains more than eight trigger units, you may notice a delay of 30 seconds or more in triggering while Reveal Logic Analyzer configures the core or cores. For example, for a core with 16 trigger units, Reveal Logic Analyzer may take as long as five minutes to display the waveforms.

If the trigger condition is not met, Reveal Logic Analyzer will continue running. In that case, you can use manual triggering, described in “Using Manual Triggering” on page 93.

Data Capture with Sample Enable

Triggers occur at every sample clock edge when the condition is met. Trace data is also captured on the sample clock edge. If a sample enable is used, each sample shown in the trace buffer is only captured when the sample enable is active and there is a sample clock. Data samples can be discontiguous, unlike those in a normal data capture.

It is also possible that the actual trigger condition may occur when the sample enable is not active, causing two changes from a normal data capture:

- The actual data values for the trigger condition may not be visible, because the data cannot be captured when the sample enable is inactive.
- Reveal Logic Analyzer cannot accurately calculate the trigger point, since the trigger point may have occurred when the sample enable was inactive. Normally a trigger point is shown as a single marker on the clock on which the trigger occurred. If a sample enable is used, a trigger region that spans five clock cycles is shown instead. Reveal Logic Analyzer can guarantee that the trigger occurred in this region, but it cannot determine during which clock cycle the trigger occurred.

The sample enable is a very useful feature, but it takes more understanding than a normal data capture.

Common Error Conditions

Reveal Logic Analyzer may fail to generate waveforms and instead issue an error message because of problems with the inserted core or cores.

Clock Causes Core to Malfunction

If Reveal Logic Analyzer issues the following error message, the core may not be functional because of a clock problem:

```
Incorrect HUB ID. Check the clock or cable setup.
```

In this case, be sure the clock is running on the target hardware.

Signature in .rvl File Does Not Match Signature in Bitstream

If you have defined a token set and Reveal Logic Analyzer issues an error message similar to the following, the core may be functional, but the signature in the .rvl file does not match the signature in the bitstream:

```
Incorrect core signature 1234 from core(core1), expected
signature is 1235.
```

This problem can be caused by saving the Reveal Inserter project file, even if you have made no changes, without re-implementing the design.

If you receive this message, regenerate the bitstream or JEDEC file, or download another file whose signature matches that of the .rvl file.

Sample Clock Runs Too Slowly

The following core signature error message indicates that the sample clock is running too slowly when you perform a logic analysis on a circuit containing a Reveal core:

```
Core response 0
```

On the board, make sure that the minimum sample clock frequency is at least three times (and preferably four times) that of the JTAG clock. If the sample clock speed is too slow, you will be unable to complete logic analysis with Reveal Logic Analyzer.

Performing Logic Analysis on Multiple Devices

You can perform logic analysis on one device in a scan chain of multiple devices on a board or multiple boards. The maximum number of boards is specified by ispVM System.

When you perform logic analysis on a device in a scan chain, you must generate a scan chain configuration (.xcf) file. See the ispVM online Help for instructions on generating this file. You must also do the following:

1. Select the **Device > Connection Setup** command to activate the Cable and I/O Port Setup dialog box.
2. Turn on the **TRST/Reset Pin Connected** option, then turn on either the **Set High, Then Low** or **Set Low, Then High** option.
3. Turn on the **ispEN/BSSCAN Pin Connected** option, then turn on either the **Set High, Then Low** or **Set Low, Then High** option.
4. Click **OK**.

Stopping a Logic Analysis

You can stop a logic analysis while it is running.

To stop a logic analysis:

- Choose **Device > Stop** or click  .

This command only stops the logic analysis on the current core in the active window. You must stop each core separately.

Using Manual Triggering

If you set up a trigger but triggering fails to occur or you want to trigger manually instead of triggering when a signal condition occurs, you can use manual triggering to capture data. The captured data may then help you find out why triggering did not occur as you originally intended.

When you select manual triggering, Reveal Logic 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 press the Stop () button to stop the logic analysis. The buffer starts downloading the data.

To use manual triggering:

Note

A logic analysis must be running before you can use the Manual Trigger command.

1. After you start the logic analysis with the  button, select **Device > Manual Trigger** or click .

   This command only applies to the logic analysis on the current core in the active window. You must stop each core separately.

2. When you have captured the desired number of triggers in multiple trigger capture mode, click  .

Viewing Waveforms

After running your logic analysis, you can view the trace buffer data in waveform format in the Waveform View tab. Whenever the trace stops, Reveal Logic Analyzer reads the trace samples back from the trace memory and automatically updates the signal waveforms. You can vary the display by using commands in the Window menu. For example, you can view multiple logic analyses in a cascade or a tile display.

If you perform a logic analysis, exit Reveal Logic Analyzer, and then reopen it, the old data is displayed in the waveform until you perform a new logic analysis.

Viewing Logic Analyses

To view a single logic analysis or multiple logic analyses:

- Choose **Window > Single** or **Window > Multiple**.

Viewing Cascading or Tiled Logic Analyses

To view multiple logic analyses in a cascade or tile display:

- Choose **Window > Cascade** or **Window > Tile**.

Splitting a Waveform

You may want to split a waveform window to compare two signals closely.

To compare two different sections of the same waveform:

- Choose **Data > Split** > **Horizontal Split** to split a waveform horizontally into two identical waveform sub-windows.
- Choose **Data > Split** > **Vertical Split** to split a waveform vertically into two identical waveform sub-windows.

Reversing a Waveform Split

You may want to merge a split waveform window after signal comparison.

To reverse the horizontal or vertical split of a waveform or both:

- Choose **Data > Split > Unsplit**.

Adjusting the Waveform Display

You can adjust the waveform display by panning and zooming.

Panning

You can move the waveform display in the Waveform View tab so that you can view any part of it.

To pan the waveform display:

- Press and drag the left mouse button to the left or the right.

Zooming In and Out

You can zoom in and out on a waveform in the Reveal Logic Analyzer Waveform View tab to increase or decrease the displayed time interval.

To zoom in on a waveform:

- Choose **Data > Zoom > Zoom In**, click  , or right-click on the waveform and choose **Zoom > zoom in**.

  Alternatively, you can press and drag the right mouse button to the right. Two vertical dashed lines appear. Reveal Logic Analyzer zooms the area between the two lines. The smaller the area between the two lines, the greater the magnification.

To zoom out on a waveform:

- Choose **Data > Zoom > Zoom Out**, click  , or right-click on the waveform and choose **Zoom > zoom out**.

  Alternatively, you can press and drag the right mouse button to the left.

  Two vertical dashed lines appear. Reveal Logic Analyzer zooms the area between the two lines. The smaller the area between the two lines, the greater the compression.

To undo the last zoom on a waveform and revert to the previously zoomed view:

- Choose **Data > Zoom > Zoom Previous** or click  .

To zoom the area between the X and O markers on the waveform view:

- Choose **Data > Zoom > Zoom to X–O** or click on  .

To fit the waveform (zoom in or out) so that both X and O markers are visible, regardless of their position on the waveform:

- Choose **Data > Zoom > Fit Window** or right-click and choose **Zoom > zoom > zoom fit**.

To expand the waveforms to their maximum size:

- Right-click and choose **Zoom > zoom full**.

Specifying the Clock Frequency

You can specify a clock frequency for your logic analysis. Setting the frequency enables you to determine the location of your X and O markers, as well as the distance between them, in the status bar of the Waveform View tab. Frequency is determined by dividing 1 by the period.

To set the clock frequency:

1. Choose **Data > Set Clock Frequency**.
2. In the Specify Clock Frequency dialog box, choose either picoseconds or nanoseconds for the period interval selector. Click the box next to Period to select picoseconds (ps) or nanoseconds (ns).
3. Type the desired period in the **Period** box.

The resulting frequency, in megahertz, is displayed in the **Frequency** box.

Note

If you type in a frequency in megahertz in the Frequency box, the resulting period is displayed in the Period box.

Placing, Moving, and Locating Markers

In the Waveform View tab, you can place, move, and locate vertical dotted lines called markers, which you can use to indicate an area of interest to you on the waveform.

You can place two markers, X and O, on the waveforms generated by Reveal Logic Analyzer. You can move these markers to any point in the waveform and view the state of the signal in the X and O columns of the Waveform View tab. You can also move the signal display to the left and right of the X and O markers.

Reveal Logic Analyzer automatically places a T marker to indicate the position of the trigger on the waveform. You can view the state of the signal at the point marked by the T marker in the T column on the Waveform View tab. When you use Single Trigger Capture mode in Reveal Inserter, you can locate the T marker, but you cannot place it or move it. When you use Multiple Trigger Capture mode in Reveal Inserter, you can move the active T marker to the next or previous trigger region. Use [<-T] and [T->] to move the T marker in this mode.

If you use a sample enable for a core, two T markers appear on the waveform, with blue highlighting in between them. The exact trigger position cannot be determined, so Reveal Logic Analyzer displays the possible trigger range (five samples) as blue highlighting instead of a trigger position.

You cannot delete markers.

Selecting X and O Marker Snap Location

You can specify whether the X and O markers snap to the edge or the center of a time interval.

To set the snap location of the X and O markers:

- Choose **Data > Snap To Edge** to snap X or O markers to the edge of a time interval. This setting is the default.
- Choose **Data > Snap To Center** to snap X or O markers to the center of a time interval.

Placing X and O Markers

To place X and O markers:

1. Click the right mouse button at the desired location on the waveform.
2. Choose **Place X Marker** or **Place O Marker** from the pop-up menu.

The marker is placed at the point where you clicked.

Locating Markers

To locate markers:

- Choose **Data > Go to X Marker, Data > Go to O Marker, or Data > Go To Trigger** to locate the X marker, O marker, or T marker, respectively.

Displaying X and O Markers in One Window

To fit the waveform (zoom in or out) so that both X and O markers are visible, regardless of their position on the waveform:

- Choose **Data > Fit Window**.

Moving X and O Markers

To move X and O markers:

- On the waveform, click and hold down the right mouse button on the appropriate marker and drag it left or right to the desired location on the waveform.

The state of each signal is displayed on the X and O column in the Waveform View tab.

Moving the Signal Display to the Left or Right of the X or O Marker

You can move the signal display one time interval to the left or right of an X or O marker.

To move the signal display to the left or right of the X or O marker:

1. To move the signal display one time interval to the left of the X marker, click  .
2. To move the signal display one time interval to the right of the X marker, click  .
3. To move the signal display one time interval to the left of the O marker, click  .
4. To move the signal display one time interval to the right of the O marker, click  .

The state of each signal is displayed in the X and O column of the Waveform View tab.

Moving the Active T Marker

If the Multiple Trigger Capture option is checked in the Data Capture Mode section of the Trigger Signal Setup tab, you can move the active T marker to the next or previous trigger region. The trigger marker itself cannot be moved, but when multiple trigger regions are present, you can select which trigger is the active trigger marker.

- To move the T marker to the next group of samples to the left of its present location, click  .
- To move the T marker to the next group of samples to the right of its present location, click  .

The state of each signal is displayed in the T column of the Waveform View tab.

Finding Signals

You can use either full or partial text strings to find signals in Reveal Logic Analyzer's Waveform View tab.

To find a signal in Reveal Logic Analyzer's Waveform View tab:

1. Choose **Bus/Signal > Find Signal** to display the Find Signal dialog box.
2. In the **Find What** box, type the full or partial text string of the signals that you wish to find.
3. Click the **Up** or **Down** button for the direction of your search.
4. Click **Find Next**.

   The first signal matching your text string will be highlighted.
5. Click **Find Next** again to find the next signal that matches your text string.

Finding Data

You can find data in a bus or a signal by using the search criteria in the Find Data dialog box.

Finding Data in a Signal

To find data in a signal:

1. Choose **Data > Find Data**.
2. In the Find Data dialog box, select the search criterion in the **Find** box. You can select Low, High, Rising Edge, Falling Edge, or Both Edges.
3. In the **Start At** box, select the place where you want the search engine to begin the search: Beginning of Data, Trigger, Last Found, X Marker, or Y Marker.
4. If you want to add a marker at the location where the data is found, turn on either the **Place X Marker** and **Place O Marker** button.
5. Click **Next** to find the next data point matching your search criteria. Click **Previous** to find the previous data point matching your criteria.

Finding Data in a Bus

To find data in a bus:

1. Choose **Data > Find Data**.
2. In the Find Data dialog box, select the search criteria in the **Find** box. You can select =, Not =, In Range, or Not in Range.
3. If you selected = or Not = in the Find box, enter the value to search for in the Value box, in hexadecimal. If you select Not =, you must repeatedly select **Next** to find all the values.
4. If you entered In Range or Not in Range in the Find box, enter the range of values to search for in the **Lower Limit** and **Upper Limit** boxes, in hexadecimal. You must repeatedly select **Next** to find all the values.
5. In the **Start At** box, select the place where you want the search engine to begin the search: Beginning of Data, Trigger, Last Found, X Marker, O Marker.
6. If you want to add a marker at the location where the data is found, select either the **Place X Marker** and **Place O Marker** button.
7. Click **Next** to find the next data point matching your search criteria. Click **Previous** to find the previous data point matching your criteria.

Printing the Waveform

You can print out a copy of the logic analysis waveform data.

To print out waveform data:

1. You can optionally use the **File > Print Preview** command to view the waveforms in the Print Preview dialog box before printing.
2. To print the waveforms, choose **File > Print**.
3. In the Print dialog box, select a printer and any desired options in the associated printer option tabs.
4. Click **Print**.

Exporting Waveform Data

You can export waveform data to a value change dump (.vcd) file, which can be imported by such third-party tools as ModelSim or Active-HDL, or to an ASCII-format text (.txt) file. You must have performed a logic analysis, implemented a trigger on hardware, and captured data that is shown in the waveform display before you can export data.

To export waveform data to a .vcd file:

1. Choose **File > Export > Export VCD**.
2. In the Export Waveform to VCD File dialog box:
   a. Browse to the location where you want to export the file.
   b. Type in a name in the **File Name** box.
   c. Set Files of Type to **Value Change Dump Files (*.vcd)**.
   d. Click **Save**.

To export waveform data to an ASCII-format .txt file:

1. Choose **File > Export > Export TEXT**.
2. In the Export Waveform to ASCII Text File dialog box:
   a. Browse to the location where you want to export the file.
   b. Type in a name in the **File Name** box.
   c. Set Files of Type to **ASCII Text Files(*.txt)**.
   d. Click **Save**.

Saving a Project

You can save the trigger settings and waveform setup settings in a Reveal Logic Analyzer project (.rva) file that you can use as an input file in the future. You can also save an existing .rva file in a file with a different name.

To save a Reveal Logic Analyzer project:

- Choose **File > Save**.

The project data is now output into an .rva file.

To save the project file with a different name:

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

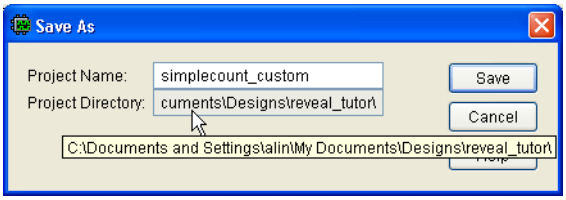
   The Save As dialog box appears.

2. In the Project Name box, enter name of the .rva file in which you want to save the project.

3. In the Project Directory box, enter the name of the directory in which you want to save the project.

   The Project Directory box displays the read-only name of the current project directory. If the name is too long to display, you can move the mouse cursor over it to display the complete name as a tool tip, as shown in Figure 27.

Figure 27: Replace Project Warning Dialog Box

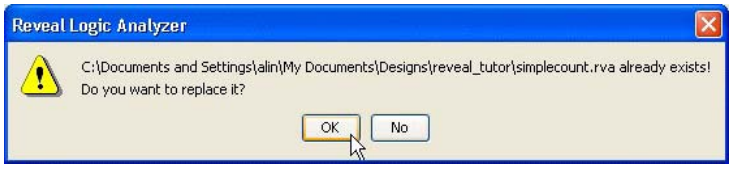


4. Click **Save**.

   If you try to save the project under a name that already exists in the current directory, the warning box shown in Figure 28 appears.

5. Click **OK** in the warning box to replace the project's contents.

Figure 28: Replace Project Warning Dialog Box



Closing a Project

To close an open Reveal Logic Analyzer project:

- Choose **File** > **Close**.

Exiting Reveal Logic Analyzer

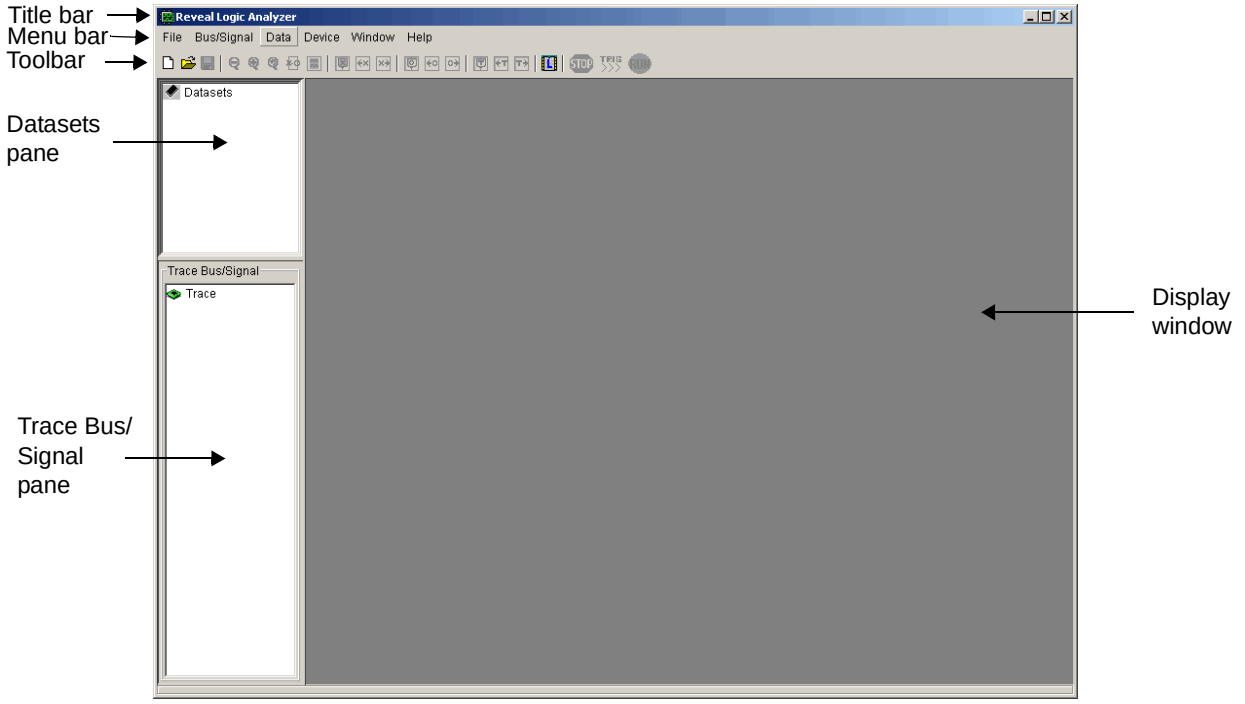
To exit Reveal Logic Analyzer:

- Choose **File** > **Exit**.

User Interface Descriptions

The Reveal Logic Analyzer window appears when you first start the tool.

Figure 29: Reveal Logic Analyzer Window



Title Bar Displays the Reveal Logic Analyzer name and the current project name.

Menu Bar Displays the menus that list the commands available in Reveal Logic Analyzer.

Toolbar Displays shortcut buttons to allow you to perform tasks identical to the commands on the menus.

Datasets pane Lists the datasets in the design and the cores in the datasets.

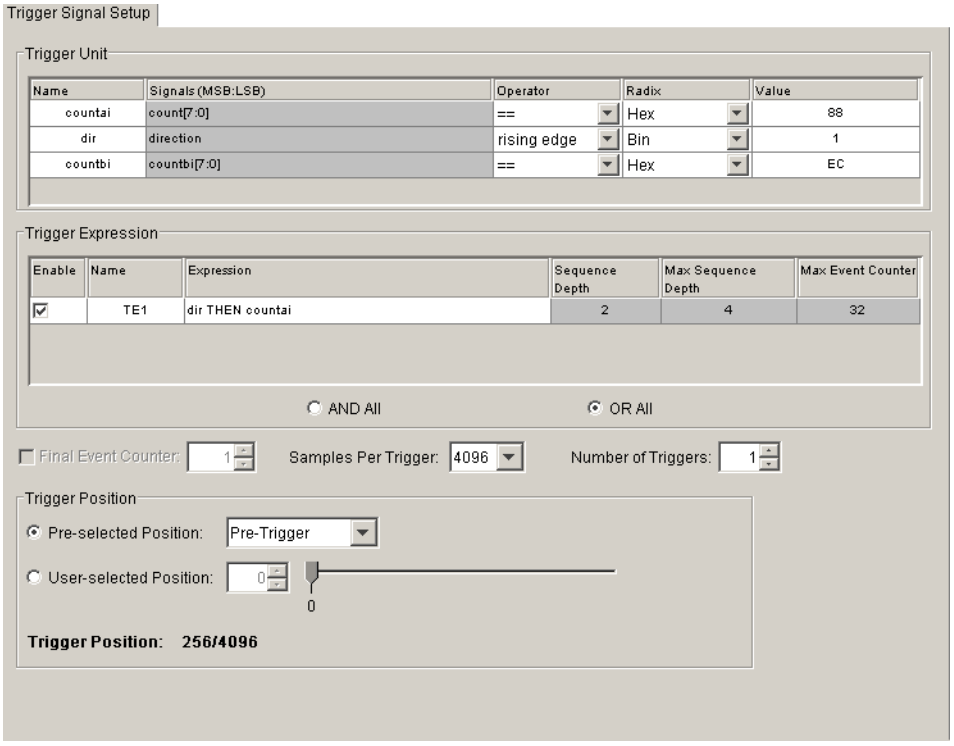
Trace Bus/Signal pane Displays all the trace and trigger-as-trace signals in the active core.

Display window Displays an LA window with the Trigger Signal Setup tab and the Waveform View tab for each core. Multiple windows are displayed for multiple cores.

Trigger Signal Setup Tab

The Trigger Signal Setup tab enables you to select the trigger signals and define the data values or pattern of data values that cause trace data collection to begin.

Figure 30: Trigger Signal Setup Tab



Trigger Unit

The parameters in the Trigger Unit section enable you to configure the trigger units, which are the basic trigger comparison mechanism in Reveal Inserter and Reveal Logic Analyzer. Trigger units allow comparison of the signal to a value that is entered during hardware debug. You can include up to 16 trigger units in a core. Each trigger unit consists of the following information:

Note

The darker gray background in certain fields in the Trigger Signal Setup tab indicates that you can change these values only in Reveal Inserter. You cannot change them in Reveal Logic Analyzer.

Name Specifies the name of the trigger unit. See “Trigger Expression and Trigger Unit Naming Conventions” on page 109 for the guidelines governing trigger unit names. The default name is TU<*number*>, where <*number*> is a sequential number. The first trigger unit is named TU1 by default.

Signals Lists the signals in the trigger unit. You can select up to 4096 trigger signals. You can specify these signals only in Reveal Inserter.

Operator Specifies the comparators that Reveal will use to compare the states of the trigger bus signals to the pattern of signal states that you set in the Trigger Signal Setup tab of Reveal Logic Analyzer. You can choose from the following states:

- == equal to. This comparator is the default.
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- Rising edge – Compares on the rising edge of the clock
- Falling edge – Compares on the falling edge of the clock
- Serial compare – Compares until the trigger condition is met. For example, if the trigger condition is 10011, the serial compare option looks for a 1 on the first clock, a 0 on the next clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only if those five conditions are met in those five clock cycles will the serial compare output be active.

  Other operators cannot be changed to a serial compare, and a serial compare cannot be changed to another operator in Reveal Logic Analyzer. These can only be changed in Reveal Inserter.

  The serial comparator is available only when a single signal is listed in the Trigger Unit signal list. If you choose this option, you must choose Binary in the Radix box.

Radix Specifies the radix of the trigger bus. It can be one of the following:

- Binary. This is the default. You must choose Binary if you selected “Serial compare” as a comparator.
- Octal
- Decimal
- Hexadecimal
- <*token_set_names*>. To select <*token_set_name*>, you must have created token sets in Reveal Inserter. See the Reveal Inserter online Help for information on this procedure.

Value Specifies the comparison value. This value is the pattern of highs and lows that you want on the trigger unit that will initiate collection of the trace data. The default is binary.

If you selected a <*token_set_name*> in the Radix column, a drop-down menu opens in the Value column. This menu lists all the tokens that you entered in the Token Manager dialog box for the chosen token set. Select any name.

You can use “x” for a don’t-care value in the Value column if you selected Binary, Octal or Hexadecimal in the Radix column and if you selected the ==, !=, or serial compare operators in the Operator column.

Trigger Expression

The parameters in the Trigger Expression section of the Trigger Signal Setup tab enable you to configure the trigger expressions, which are combinatorial, sequential, or both combinatorial and sequential equations of trigger units that define when the collection of the trace data samples begins. You can add up to 16 trigger expressions. Each trigger expression consists of the following information:

Enable Determines whether the trigger expression is active when the triggering is enabled by the Run button.

Name Specifies the name of the trigger expression. The default name is TE<*number*>, where <*number*> is a sequential number. The first trigger expression is named TE1 by default.

Expression Specifies the trigger units and the operator or operators that indicate the relationship of one trigger unit to other trigger units. You can use the following operators to connect trigger units:

- & (AND) – Combines trigger units using an & 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.

- 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.
- # (count) – Inserts a counter into a sequence. Sequences are groups of combinatorial operations connected by THEN operators. The counter counts how many times a sequence must occur before a THEN statement. The maximum value of this count is determined by the Max Event Counter value. It must be specified in Reveal Inserter and cannot be changed in Reveal Logic Analyzer.

  Here are some examples.

  The following statement:

  ```
  TU1 #5 THEN TU2
  ```

  means that TU1 must be true for five consecutive or non-consecutive sample clocks before TU2 is evaluated. The counts do not have to be sequential. TU1 does not have to be true five times in a row to satisfy this condition. It only has to be true fives times to meet this condition.

  The next expression:

  ```
  (TU1 & TU2)#2 THEN TU3
  ```

  means “wait for the second occurrence of TU1 and TU2 being 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 (#) operator per sequence. For example, the following statement is not valid, because it uses two counts in a sequence:

  ```
  TU1 #5 & TU2 #2
  ```

- ## (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.

Sequence Depth Specifies the number of sequences, which are groups of combinatorial operations connected by THEN operators, used in a trigger expression. Reveal supports up to 16 sequence levels.

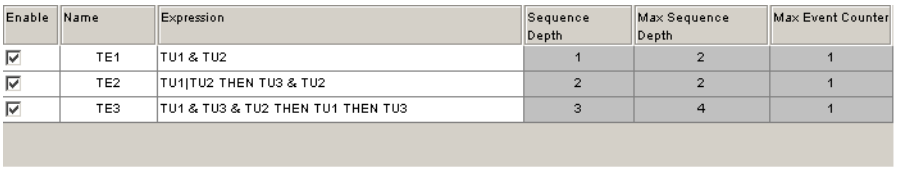
For example, in the following figure, TE1 consists of one sequence, since it has no THEN operator. It therefore has a sequence depth of 1. TE2 has two

sequences, TU1|TU2 and TU3 & TU2, linked by a THEN operator, so its sequence depth is 2. TE3 has three sequences:

- TU1 & TU3 & TU2 followed by THEN
- TU1 followed by THEN
- TU3

TE3 therefore has a sequence depth of 3.

Figure 31: Trigger Expression Sequences



Enable	Name	Expression	Sequence Depth	Max Sequence Depth	Max Event Counter
☑	TE1	TU1 & TU2	1	2	1
☑	TE2	TU1|TU2 THEN TU3 & TU2	2	2	1
☑	TE3	TU1 & TU3 & TU2 THEN TU1 THEN TU3	3	4	1

Max Sequence Depth Specifies the maximum number of sequences, or trigger units connected by THEN operators, that can be used in a trigger expression. You can set this option only in Reveal Inserter.

Max Event Counter Determines the maximum size of the count in the trigger expression (the count is how many times a sequence must occur before a THEN statement). You can set this option from 1 to 64,000. The maximum is 64,000. The default is 1. You can set this option only in Reveal Inserter.

AND All Indicates that the conditions specified by all the trigger expressions must be met before the trace data is captured.

OR All Indicates that the conditions of one of the trigger expressions must be met before the trace data is captured. This option is the default.

Final Event Counter Specifies the number of times that AND All or OR all must be met before triggering occurs and trace data is captured. In effect, it adds an extra counter to the final trigger logic. The lowest value of the counter is 2 times, and the maximum value is 65536 times. The default value is 8 times. Leaving this option unselected is equivalent to setting the counter to a value of 1 time. This option is unselected by default.

Samples Per Trigger Specifies the number of samples to collect per trigger. The minimum is 16. The maximum is the trace buffer depth chosen in Reveal Inserter when the core is generated. The values available in the Samples Per Trigger box also change according to the number of triggers. If the number of triggers is set higher than 1, the samples per trigger multiplied by the selected number of triggers cannot exceed the trace buffer depth. Reveal Logic Analyzer adjusts the Samples Per Trigger value, if necessary. The default number of samples per trigger is the sample buffer depth. For example, if the sample buffer depth is 2048, the default sample per trigger is 2048.

Number of Triggers Specifies the trace buffer depth divided by the samples per trigger. The trace buffer depth is specified by the setting of the

Buffer Depth parameter in the Trace Signal Setup tab in Reveal Inserter. The default is 1.

Trigger Position

Pre-selected Position Specifies the position of the trigger point in the data stream. For example, 32/512 means the trigger point is the 32nd sample in the trace memory that has a total depth of 512 samples (from sample 0 to sample 511). You can use one of the following samples in the Position box:

- Pre-Trigger – Sets the trigger point at 1/16 of the total number of samples in the trace memory. For example, when the trace memory has a total number of 512 samples, 1/16 of these would be 32. The 32 setting means that 32 data samples are collected before the trigger occurs. The Pre-Trigger setting is helpful if you are mostly interested in the data states that occurred after the trigger event. Using the example just given, only 32 samples of data (from sample 0 to sample 31) that occur before the trigger are stored, but 480 samples of data (from sample 32 to sample 511) that occur after the trigger are stored.
- Center-Trigger – Sets the trigger point at 50 percent (1/2) of the total number of samples in the trace memory. For example, when the trace memory has a total number of 512 samples, 50 percent of these would be 256, so the trigger point would be set at 256. The Center-Trigger setting provides equal amounts of trace data before and after the trigger event. Using the example just given, 256 samples of data (from sample 0 to sample 255) that occur before the trigger are stored, and 256 samples of data (from sample 256 to sample 511) that occur after the trigger are stored.
- Post-Trigger – Sets the trigger point at 15/16 of the total number of samples in the trace memory. For example, when the trace memory has a total number of 512 samples, 15/16 of these would be 480. The Post-Trigger setting is helpful if you are mostly interested in the data states that occurred before the trigger event. Using the example just given, 480 samples of data (from sample 0 to sample 479) that occur before the trigger are stored, but only 31 samples of data that occur after the trigger are stored.

Note

The actual trigger position in the captured samples may not match the trigger position that you set in the Trigger Signal Setup tab. For example, if the trigger condition occurs before the trigger position set in the Trigger Signal Setup tab, the actual trigger position is earlier than that specified in the Trigger Signal Setup tab.

User-Selected Position Enables you to choose the trigger point from certain points selected by the tool.

Trigger Position Shows the position of the trigger point in relation to the number of samples per trigger. It is in read-only notation at the very bottom of the Trigger Position section. For example, if you selected the Center-Trigger setting for the Pre-selected Position option and you selected 1024 samples in the Samples Per Trigger box, the Trigger Position field would display a trigger point equal to half the samples, 512/1024.

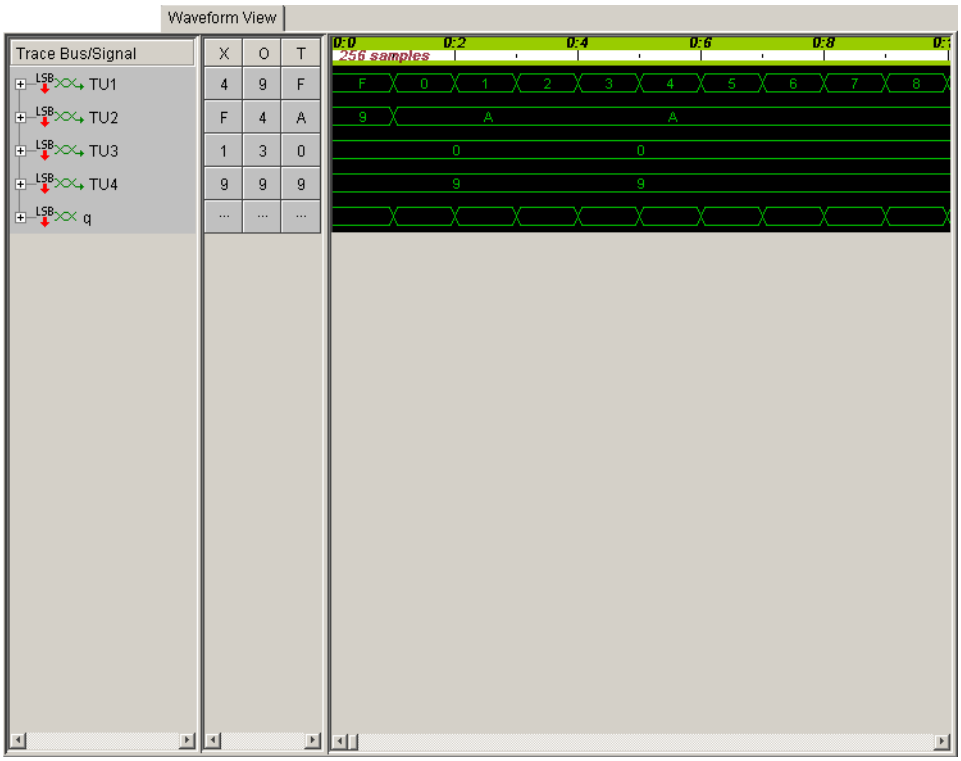
Trigger Expression and Trigger Unit Naming Conventions

You can rename trigger units and trigger expressions. The names can be a mixture of lower-case or upper-case letters, underscores, and digits from 0 through 9. The first character must be either an underscore or a letter. The names can be any length.

Waveform View Tab

The Waveform View tab, shown in the following figure, enables you to view the contents of the trace buffer in waveform format.

Figure 32: Waveform View Tab



Trace Bus/Signal column Displays the names of the trace buses and signals and trigger-as-trace signals in the selected core.

Marker Column Displays the state of each signal on the waveform marked with the X marker, O marker, and T (trigger) marker:

- X – Displays the state of each signal in the time interval marked with the X marker.
- O – Displays the state of each signal in the time interval marked with the O marker.
- T – Displays the state of each signal in the time interval marked with the T marker.

Waveform window Displays the trace buffer data in waveform format.

The numbers at the top, such as 0:2, 0:4, and 0:6 in the illustration just shown, are index numbers that represent the number of the trigger event followed by the number of the trace sample or trace frame. For example, the illustration shows waveforms for 256 samples collected from four trigger events. 0:6 means the sixth sample of the first trigger event, and 0:254 means the 254th sample of the first trigger event. Similarly, 2:36 means the 36th sample in the third trigger event, and 3:184 means the 184th sample in the fourth trigger event.

Markers

In the Waveform View tab are three vertical lines called markers, which you can use to indicate an area of interest to you on the waveform. Each marker displays in the XOT column the state of each signal that it marks:

- The X marker displays the state of each signal in the X column. It is a blue line.
- The O marker displays the state of each signal in the O column. It is a purple line.
- The T marker displays the state of each signal in the T column. It is a red line.

You can place and locate the X and O markers and move them to any point in the waveform. You can view the state of the signal in the X and O columns.

Reveal Logic Analyzer automatically places a T marker to indicate the position of the trigger on the waveform. You can view the state of the signal at the point marked by the T marker in the T column. When you are using the Single Trigger Capture option in Reveal Inserter, you can locate the T marker, but you cannot place it or move it. When you are using the Multiple Trigger Capture option in Reveal Inserter, you can move the active T marker to the next or previous trigger region. Use [<-T] and [T->] to move the active T marker in this mode.

If you use a sample enable for a core, two T markers appear on the waveform, with blue highlighting in between them. The exact trigger position cannot be determined, so Reveal Logic Analyzer displays the possible trigger range (five samples) as blue highlighting instead of a trigger position.

You cannot delete markers.

Dialog Boxes

The dialog boxes in Reveal Logic Analyzer are listed here in alphabetical order.

Bus Radix Dialog Box

The Bus Radix dialog box is activated by the Set Bus Radix command on the drop-down menu that appears when you highlight a bus in the Waveform View tab and right-click the mouse button. It enables you to specify a bus radix.

This dialog box contains the following options:

Binary Specifies a binary bus radix.

Octal Specifies an octal bus radix.

Decimal Specifies a decimal bus radix.

Hexadecimal Specifies a hexadecimal bus radix.

<token_set> Specifies that a token set be used to specify the radix. If the token bit size is the same as the bus width, user-defined token sets will be displayed on this dialog box so that you can choose a token set, if desired. For example, in the illustration shown, “anc” and “bnd” are token sets.

OK Applies the changes that you have made and closes the dialog box.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

Cable and I/O Port Setup Dialog Box

The Cable and I/O Port Setup dialog box is activated by the Device > Connection Setup command in Reveal Logic Analyzer. It specifies the parameters that govern the connection of the download cable between the evaluation board and your computer.

This dialog box contains the following options:

Cable Type Specifies the download cable type.

Auto Detect Automatically detects the port address of the parallel or USB port to which the download cable is connected.

Port Setting Specifies the download port address. Select the port from the list or click Auto Detect.

TCK Low Pulse Width Delay Specifies the width of the TCK low pulse in multiples of the normal width: 1 is normal, 2 is twice normal, and so on. Extending the pulse width may help if you have trouble communicating with a target board that has a long JTAG chain.

Custom Port Specifies the address of the custom parallel port to which the download cable is connected. Use hexadecimal format to type the port name.

TRST/Reset Pin Connected Specifies whether the TRST pin connected to the download cable is active high or active low.

ispEN/BSCAN Pin Connected Specifies whether the ispEN/BSCAN pin connected to the download cable is active high or active low.

OK Applies any changes that you have made and closes the dialog box.

Cancel Closes the dialog box without applying any changes.

Colors Dialog Box

The Colors dialog box is activated by the Set Colors command on the drop-down menu that appears when you highlight a signal in the Waveform View tab and right-click the mouse button. It enables you to specify signal colors.

This dialog box contains the following options:

Foreground Specifies the foreground color for the specified signal. Clicking the button next to the box opens the Color Chooser dialog box. Clicking the button next to System Default uses the default colors.

Background Specifies the background color for the specified signal. Clicking the button next to the box opens the Color Chooser dialog box. Clicking the button next to System Default uses the default colors.

OK Applies any changes that you have made and closes the dialog box.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

Device Information Dialog Box

The Device Information dialog box is activated by the Device > Device Information command. It displays information about the device that is under analysis.

This dialog box contains the following options:

Name Displays the name of the device.

ID Displays the device identification.

Inserter File (.rvl) Displays a list of the available Reveal Inserter project (.rvl) files generated by Reveal Inserter in your project directory.

Back Re-opens the New Project dialog box.

Cancel Closes the dialog box without applying any changes.

Finish Closes the dialog box.

Help Displays the help topic for this dialog box.

Find Data Dialog Box

The Find Data dialog box is activated by the Data > Find Data command. It enables you to find specific data in a waveform and to place an X or O marker at that location.

This dialog box contains the following options:

Bus/Signal Name Specifies the name of the bus or signal for which to search.

Find Specifies the type of data for which to search. You can select one of the following for signals:

- Low – Finds where the signal is at 0.
- High – Finds where the signal is at 1.
- Rising Edge – Finds where the signal transitions from 0 to 1.
- Falling Edge – Finds where the signal transitions from 1 to 0.
- Both Edges – Finds where the signal transitions from 0 to 1 and from 1 to 0.

You can select one of the following for buses:

- = Finds where the bus value is equal to the specified value, in hexadecimal. You must repeatedly select Next to find all the values.
- Not = Finds where the bus value is not equal to the specified value, in hexadecimal. You must repeatedly select Next to find all the values.
- In Range – Finds where the bus value is in the range specified by Lower Limit and Upper Limit, in hexadecimal. You must repeatedly select Next to find all the values in the range.
- Not in Range – Finds where the bus value is not in the range specified by Lower Limit and Upper Limit, in hexadecimal. You must repeatedly select Next to find all the values.

Value (HEX) Enables you to set a search value for buses, in hexadecimal, when you specify = or Not = in the Find field.

Lower Limit (HEX), Upper Limit (HEX) Specify the limits of the range, in hexadecimal, when you select a bus in the Bus/Signal Name field and the In Range or Not in Range option in the Find field.

Start At Specifies where to start the search. You can start at one of the following locations:

- Beginning of Data – Starts the search at the beginning of the waveform data.
- Trigger – Starts the search from the location of the trigger.
- Last Found – Starts the search from the location of the last result found.
- X Marker – Starts the search from the location of the X marker.
- O Marker – Starts the search from the location of the O marker.

When Found Places an X marker or an O marker at the location of the search result.

Next Finds the next bus or signal to meet the search criteria.

Previous Finds the previous bus or signal that met the search criteria.

Close Closes the dialog box.

Help Displays the help topic for this dialog box.

Find Signal Dialog Box

The Find Signal dialog box is activated by the Bus/Signal > Find Signal command. It enables you to find a signal in a bus.

This dialog box contains the following options:

Find What Specifies the signal name or the string to find in the signal name.

Direction Directs the search either up or down in the list of signals.

Find Next Finds the next signal containing the specified string.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

New Project Dialog Box

The New Project dialog box is activated by the File > New command. It enables you to create a new Reveal Logic Analyzer project.

This dialog box contains the following options:

Project Name Specifies the name of the project.

Project Directory Specifies the location of the Reveal project directory.

XCF File Displays a list of the available scan chain configuration (.xcf) files. This file is required only when you program devices in a JTAG daisy chain. The .xcf file must reside in the Project Navigator project directory for Reveal Logic Analyzer to use it.

Browse Enables you to navigate to the location of the Reveal project directory.

Next Opens the Device Information dialog box.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

Print Preview Dialog Box

The Print Preview dialog box is activated by the File > Print Preview command. It enables you to view the image that will be printed before you actually print it.

This dialog box contains the following options:

Close Closes the dialog box.

Print Prints the view shown in the dialog box.

Help Displays the help topic for this dialog box.

Reveal Logic Analyzer Startup Wizard Dialog Box

The Reveal Logic Analyzer Startup Wizard dialog box is activated by the Tools > Reveal Logic Analyzer command in ispLEVER. It informs you that a new Reveal Logic Analyzer project will be created, since Reveal Logic Analyzer did not find an existing project.

This dialog box contains the following options:

OK Activates the New Project dialog box so that you can create a new Reveal Logic Analyzer project.

Help Displays the help topic for this dialog box.

Save As Dialog Box

The Save As dialog box is activated by the File > Save As command. It enables you to save a Reveal Logic Anayzer project to a different file.

This dialog box contains the following options:

Save Saves the Reveal Logic Analyzer project to a different .rva file.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

Specify Clock Frequency Dialog Box

The Specify Clock Frequency dialog box is activated by the Data > Set Clock Frequency command. It enables you to specify the frequency of the clock governing the signal.

This dialog box contains the following options:

Period Specifies the time period to display in the waveform view, either in nanoseconds (ns) or picoseconds (ps). Select the time period to display by clicking on the box at the right.

Frequency Specifies the clock frequency, in megahertz (MHz).

OK Applies any changes that you have made and closes the dialog box.

Cancel Closes the dialog box without applying any changes.

Help Displays the help topic for this dialog box.

Toolbar

The toolbar consists of buttons that enable you to perform tasks without using menu commands.

Figure 33: Reveal Logic Analyzer Toolbar

Opens the New dialog box so you can create a new Reveal Logic Analyzer project.

The equivalent command is File > New.

Opens the Open dialog box so you can open an existing Reveal Logic Analyzer project (.rva) file.

The equivalent command is File > Open.

Saves the Reveal Logic Analyzer project (.rva) file.

The equivalent command is File > Save.

Compresses the waveforms in the Waveform View tab.

The equivalent command is Data > Zoom > Zoom Out.

Expands the waveforms in the Waveform View tab.

The equivalent command is Data > Zoom > Zoom In.

Reverts to the view on the screen before the last view and magnifies it.

The equivalent command is Data > Zoom > Zoom Previous.

Zooms in to the X marker, O marker, or the first trigger position.

The equivalent command is Data > Zoom > Zoom to X-O.

Sizes the waveform to fit the window.

The equivalent command is Data > Zoom > Fit Window.

Goes to an X marker that you can place anywhere on the waveform.

The equivalent command is Data > Go To > Go To X.

Moves the X marker one time interval to the left.

Moves the X marker one time interval to the right.

Goes to a 0 marker that you can place anywhere on the waveform.

Data > Go To > Go To O

Moves the O marker one time interval to the left.

Moves the O marker one time interval to the right.

Marks the location with a T where a trigger condition matches. This marker cannot be moved.

The equivalent command is Data > Go To > Go To T.

Moves the signal display left.

Moves the signal display right.

Starts the ispVM software.

The equivalent command is Device > ispVM System.

Stops a logic analysis.

The equivalent command is Device > Stop.

Enables manual triggering. This button forces a trigger and fills the buffer with data at that time. In single trigger capture mode, it fills the buffer and stops. In multiple trigger capture mode, it captures one trigger and data. This command is available only after you start running the logic analysis.

The equivalent command is Device > Manual Trigger.

Performs a logic analysis.

The equivalent command is Device > Run Selected LAs.

Lists the cores available for logic analysis.

Status Bar

The status bar appears at the top of the LA window for each core and displays the following information.

Figure 34: Status Bar

Status indicator Displays five states:

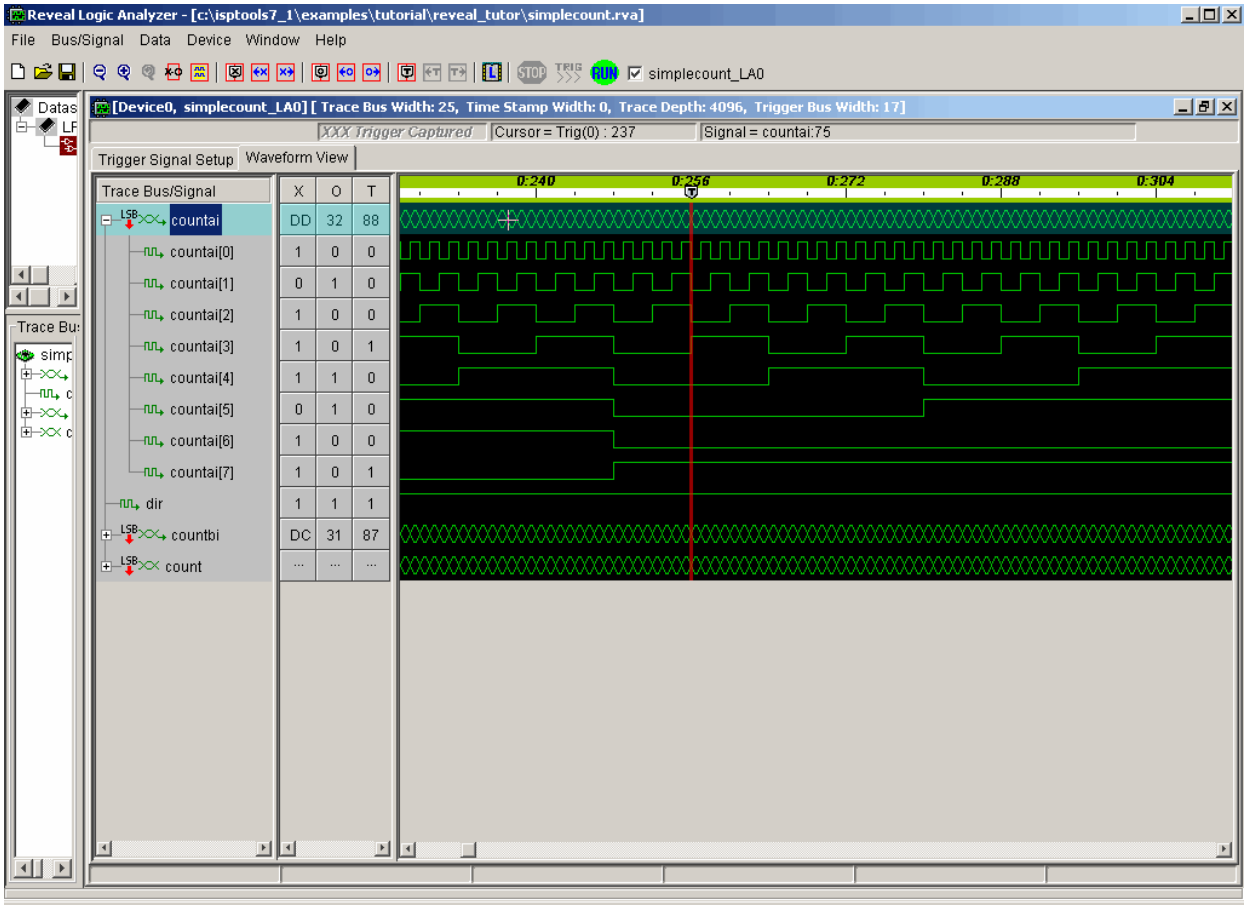
- Configuration – Indicates that the Device > Run Selected LAs command has been invoked.
- Running – Indicates that Reveal Logic Analyzer is running and waiting for triggers.
- <*Number*> Trigger Captured – Indicates the number of trigger units used to capture trace data.
- Uploading – Indicates that the Logic Analyzer is reading the trace buffer.
- Completed – Indicates that Reveal Logic Analyzer has displayed the signal waveforms.

Trigger indicator Displays the number of trigger events captured.

Sample cursor position Displays which sample the cursor is positioned over.

Signal cursor position Displays which signal the cursor is positioned over. When you position the cursor over a signal in a bus, the bus value is displayed, as shown in the following figure.

Figure 35: Bus Value Shown in Signal Information Box



Menus

Reveal Logic Analyzer's graphical user interface includes the File menu, the Bus/Signal menu, the Data menu, the Device menu, the Window menu, the Help menu, and pop-up menus.

File Menu

The File menu consists of the following commands:

New Opens the New Project dialog box so you can create a new Reveal Logic Analyzer project. This command is equivalent to the toolbar button.

Open Opens the Open Project dialog box so you can select an existing Reveal Logic Analyzer Project (.rva) file. This command is equivalent to the toolbar button.

Open Demo Provides simulation data so that you can learn how to use Reveal Logic Analyzer.

Close Closes a Reveal Logic Analyzer project (.rva) file.

Save Saves the output of Reveal Logic Analyzer in a Reveal Logic Analyzer project (.rva) file. This command is equivalent to the  toolbar button.

Save As Opens the Save dialog box so you can save your Reveal Logic Analyzer project (.rva) file with a different file name.

Print Opens the Print dialog box so you can print Reveal Logic Analyzer logic analysis waveforms.

Print Preview Opens the Print Preview dialog box to display what the printer will print.

Export Exports waveform data to a value change dump (.vcd) file or an ASCII-format .txt file.

Recent Files Displays the names of the four most recently opened Reveal Logic Analyzer project (.rva) files.

Exit Exits the Reveal Logic Analyzer software.

Bus/Signal Menu

The Bus/Signal menu consists of the following commands:

Group Into Bus Groups trace signals into a bus signal.

Ungroup From Bus Ungroups trace signals grouped into a bus signal.

Find Signal Displays the Find Signal dialog box so that you can find a signal.

Unhide All Renders all hidden signals visible.

Data Menu

The Data menu consists of the following commands:

Find Data Displays the Find Data dialog box.

Set Clock Frequency Displays the Specify Clock Frequency dialog box so that you can specify the clock period or the frequency.

Zoom > Zoom In Magnifies the waveforms in the Waveform View tab. This command is equivalent to the  toolbar button.

Zoom > Zoom Out Compresses the waveforms in the Waveform View tab. This command is equivalent to the  toolbar button.

Zoom > Zoom To X-0 Zooms the area between the X marker and the O marker in the Waveform View tab. This command is equivalent to the  toolbar button.

Zoom > Fit Window Sizes the waveform to fit the window in the Waveform View tab. This command is equivalent to the  toolbar button.

Zoom > Zoom Previous Reverts to the zoomed view before the last zoomed view in the Waveform View tab. This command is equivalent to the  toolbar button.

Go To > Go To X Marker Goes to a marker that you can place anywhere on the waveform in the Waveform View tab. This command is equivalent to the  toolbar button.

Go To > Go To 0 Marker Goes to a marker that you can place anywhere on the waveform in the Waveform View tab. This command is equivalent to the  toolbar button.

Go To > Go to Trigger Marks the location in the Waveform View tab with a T where the trigger condition matches the condition set in the Trigger Setup tab. The T marker cannot be moved. This command is equivalent to the  toolbar button.

Split > Horizontal Split Horizontally splits a waveform into two identical waveform sub-windows in the Waveform View tab to make it easier to compare different portions of the waveform.

Split > Vertical Split Vertically splits a waveform in the Waveform View tab into two identical waveform sub-windows across signal or bus assignment and waveform sub-windows to make it easier to compare different signals of the waveform.

Split > Unsplit Reverses the horizontal or vertical split of the waveform or both in the Waveform View tab.

Snap to Edge Snaps the X or O marker to the edge of the time interval in the Waveform View tab. This setting is the default.

Snap to Center Snaps the X or O marker to the center of the time interval in the Waveform View tab.

Device Menu

The Device menu consists of the following commands:

Connection Setup Opens the Cable and I/O Port Setup dialog box.

ispVM System Starts ispVM System. This command is equivalent to the  toolbar icon.

Device Information Opens the Device Information dialog box.

Run Selected LAs Performs a logic analysis of the cores that you selected in the <*design_name*>_LA<*core_number*> checkbox.

Manual Trigger Enables manual triggering. This command forces a trigger and fills the buffer with data at that time. In single-trigger capture mode, it fills the buffer and stops. In multiple-trigger capture mode, it captures one trigger and data. This command is available only after you start running the logic analysis. It is equivalent to the  toolbar button.

Stop Terminates a logic analysis. This command is available only after you start running the logic analysis. It is equivalent to the  toolbar button.

Window Menu

The Window menu consists of the following commands:

Show LA Window > Device > Device LA Displays the Device and Device LA submenus.

Single Displays a single logic analysis window.

Multiple Displays multiple logic analysis windows.

Cascade Arranges multiple opened logic analysis windows so that they overlap.

Tile Arranges multiple opened logic analysis windows so that they are contiguous.

Help Menu

The Help menu consists of the following commands:

Reveal Logic Analyzer Help Opens the online Help for Reveal Logic Analyzer.

ispLEVER Help Opens the online Help for all the design tools in ispLEVER.

About Opens the About Reveal Logic Analyzer dialog box, which shows the version of Reveal Logic Analyzer being used.

Trace Bus/Signal Pane Pop-Up Menu

The following pop-up menu appears when you highlight a bus or a signal in the Waveform View tab in an LA window and click the right mouse button.

Locate in Waveform View Highlights the selected signal or bus between two horizontal lines in the Waveform View tab.

Waveform View Tab Pop-Up Menu

The following pop-up menu appears when you highlight a bus or a signal in the Waveform View tab in an LA window and click the right mouse button.

Group Into Bus Groups signals into a bus signal.

Ungroup From Bus Ungroups signals grouped into a bus signal.

Hide Hides a signal from view.

Unhide All Makes all signals and buses visible in the Waveform View tab.

Set Bus Radix Activates the Bus Radix dialog box so you can specify a radix for a bus.

Rename Enables you to rename a bus.

Set Colors Activates the Colors dialog box so you can change the color of a signal.

Flip signal order Enables you to list the signals in a bus in order of the left-most, or most significant, bit (MSB) or the right-most, or least significant, bit (LSB).

Set MSB/LSB Reverses the order of the bits in a bus:

- MSB – Makes a bit the most significant bit (MSB).
- LSB – Makes a bit the least significant bit (LSB).

Waveform View Tab Marker Pop-Up Menu

The following pop-up menu appears when you click the right mouse button on a waveform in the Waveform View tab.

Place X Marker Places an X marker on the waveform.

Place O Marker Places an O marker on the waveform.

Zoom Enables you to perform the following types of zooming:

- Zoom in – Magnifies the waveforms in the window.
- Zoom out – Shrinks the waveforms in the window.
- Zoom fit – Sizes the waveform to fit the window.
- Zoom full – Magnifies the waveforms in the window to their maximum size.
-

4

Reveal On-Chip Debug Tutorial

Introduction

This tutorial describes how the Reveal Inserter and the Reveal Logic Analyzer in ispLEVER work together to add internal on-chip debug logic to your design as part of your functional debug strategy. The Reveal Logic Analyzer enables you to evaluate the state of the internal signals within your device with minimal impact to scarce logic resources and I/Os. “Software-based” logic analyzers like Reveal provide a more economical and convenient way to examine actual device behavior than traditional external logic analyzers.

This tutorial also covers the basics of programming a LatticeECP2 LFEC20 device using the ispVM programming environment. For a more complete presentation on this topic, see the online Help for ispVM.

Learning Objectives

When you have completed this tutorial, you should be able to do the following:

- Understand the basic design flow, processes, and data files involved in incorporating a Reveal core into your design.
- Understand two distinct design flows possible to integrate a Reveal core: RTL instantiation or EDIF netlist.
- Define the basic trigger logic and trace buffer dimensions for an internal logic analysis.
- Link the internal Reveal logic analysis core interface to trigger and trace signals.
- Understand what additional resources are required to add internal logic analysis cores to your design.
- Program the standard LatticeECP2 evaluation board.

- Examine the internal signals using the Reveal Logic Analyzer waveform display.

Time to Complete This Tutorial

The time to complete this tutorial is about 60 minutes.

System Requirements

The following software is required to complete this tutorial:

- ispLEVER Starter or ispLEVER software with device support for the device used with your Lattice Semiconductor evaluation board
- Active licenses for Synopsys Synplify or Mentor Graphics Precision RTL Synthesis with VHDL support
- ispVM

This tutorial requires one of the following FPGA boards to examine the run-time operation of the tutorial design:

- LatticeECP2 Standard Evaluation Board (revision B)
- LatticeECP2 Advanced Evaluation Board (revision C)
- LatticeXP Standard Evaluation Board
- LatticeXP Advanced Evaluation Board

In addition, this tutorial requires the following hardware to power and program an FPGA board:

- ispDOWNLOAD cable
- AC adapter

Accessing Online Help

You can find online help information on any tool included in the tutorial at any time by pressing the F1 key.

About the Tutorial Data Flow

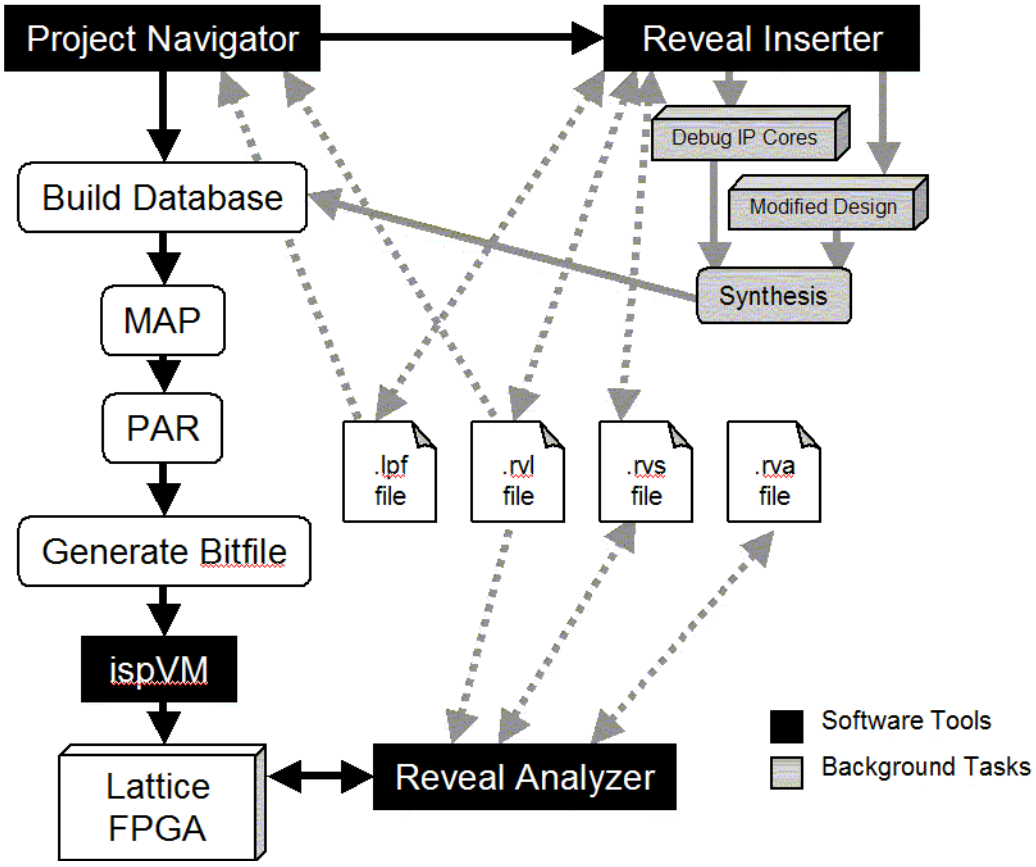
Two flows are available in Reveal for inserting debug capabilities into your design: the RTL flow and the EDIF netlist flow. This tutorial demonstrates the EDIF netlist flow. The primary distinction between this flow and the RTL flow is that the EDIF flow displays signals in the Reveal Inserter software as post-synthesis net names. The RTL flow displays signal names in their original form. For more information on these two flows, see the online Help for the Reveal Inserter and the Reveal Logic Analyzer.

The flow chart shown in Figure 36 illustrates the sequence of ispLEVER tools used to specify debug options, implement the design, program the FPGA, and analyze the results in the tutorial design. It begins with a predefined RTL description of the tutorial design, simplecount.vhd. The Reveal Inserter is used to define the original design and the debug options. A new version of the design (vhdlhdesign.ngo), including a reference to the Reveal core added, is

processed through the normal Map Design, Place & Route Design, and Generate Bitstream Data processes to create a programming file.

The flow chart in Figure 36 shows the sequence of tools to use to program and run the Reveal core on a LatticeECP2 evaluation board.

Figure 36: Tutorial Design Flow for Debugging with Reveal



Task 1: Creating an RTL-Type ispLEVER Project

In this task, you will create an RTL-type project for the LatticeECP2 standard evaluation board device, add the tutorial design source file, import predefined constraints for location and timing preferences, and use the Build Database process to create a Lattice Semiconductor logical object file (.ngo) in preparation for adding a Reveal core.

Although the tutorial tasks and figures illustrate the LatticeECP2 standard evaluation board, the procedures and preferences supplied with the tutorial are compatible with LatticeECP2 advanced build of the FPGA board and the standard and advanced builds of the LatticeXP FPGA board.

The first step is to create a new project in Project Navigator.

To create a new project:

1. Start ispLEVER, if it is not already running.

   Note

   If you are running this tutorial on Linux, type the following on the command line to activate Project Navigator:

   <*ispLEVER_install_path*>/ispcpld/bin/**ispgui**

2. In Project Navigator, select **File > New Project** to open the Project Wizard dialog box.

   Note

   If you are going to use Reveal Inserter on the Linux platform, you must install a stand-alone synthesis tool, such as Synopsys® Synplify Pro®, before you create a Reveal project.

   After you installed the stand-alone synthesis, you must set the SYNPLIFY and LM_LICENSE_FILE environment variables. The SYNPLIFY variable is set to the installation path of the synthesis tool. The LM_LICENSE_FILE variable is set to the path of the license file of the synthesis tool.

   In addition, your Linux system must meet the minimum system requirements outlined in the *ispLEVER Installation Guide for Linux*.

   Only the EDIF flow is supported for Linux.

3. In the Project Wizard dialog box, shown in Figure 37, select or specify the following:

   a. In the Project Name box, enter **simplecount**.

   b. In the Location box, enter the following directory:

      <*install_path*>\examples\Tutorial**reveal_tutor**

   c. In the Design Entry Type box, select **VHDL**.

   d. In the Synthesis Tools box, select **Synplify**.

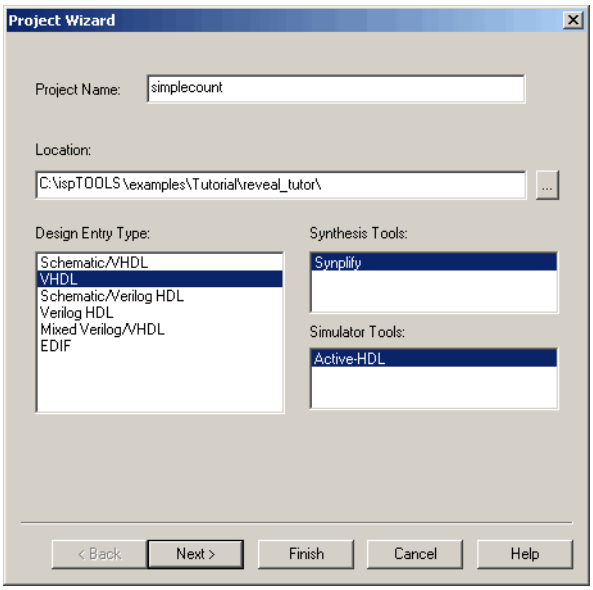
   e. In the Simulator Tools box, select **Active-HDL**.

   f. Click **Next**.

   Note:

   If you want to preserve the original tutorial design files, save the reveal_tutor directory to another location on your computer before proceeding. You must also copy the simplecount.lpf file to this new location.

Figure 37: Project Wizard Dialog Box



4. In the Project Wizard - Select Device dialog box, shown in Figure 38, do the following:

   a. In the Family box, choose **LatticeECP2** or the appropriate device family for your build of the FPGA board.

   b. In the Device box, choose **LFE2-50E** or the appropriate device for your build of the FPGA board.

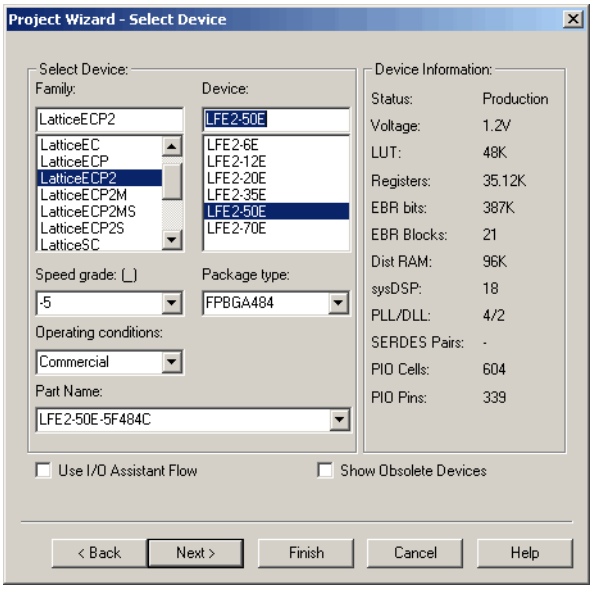
   c. In the Speed grade box, choose **-5**.

   d. In the Package Type box, choose **FBPGA484** or the appropriate package for your build of the FPGA board.

   e. In the Operating Conditions box, choose **Commercial**.

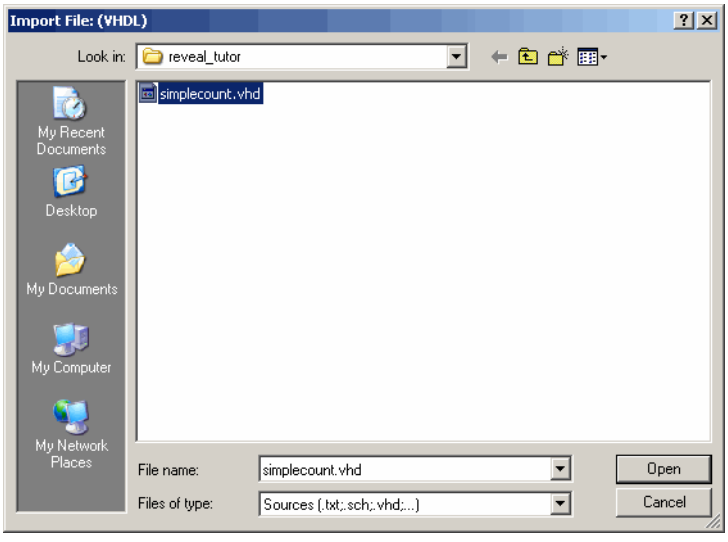
f. Click **Next** to open the Project Wizard - Add Source dialog box.

Figure 38: Project Wizard – Select Device Dialog Box



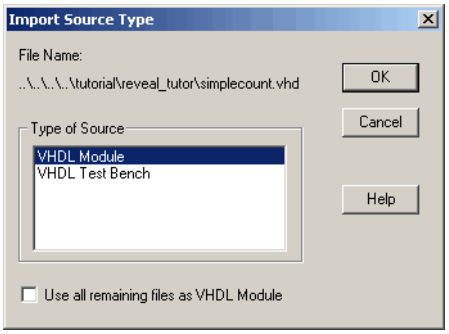
5. In the Project Wizard - Add Source dialog box, click **Add Source** to activate the Import File (VHDL) dialog box.
6. In the File Name box in the Import File (VHDL) dialog box, shown in Figure 39, select **simplecount.vhd** and click **Open**.

Figure 39: Import File (VHDL) Dialog Box



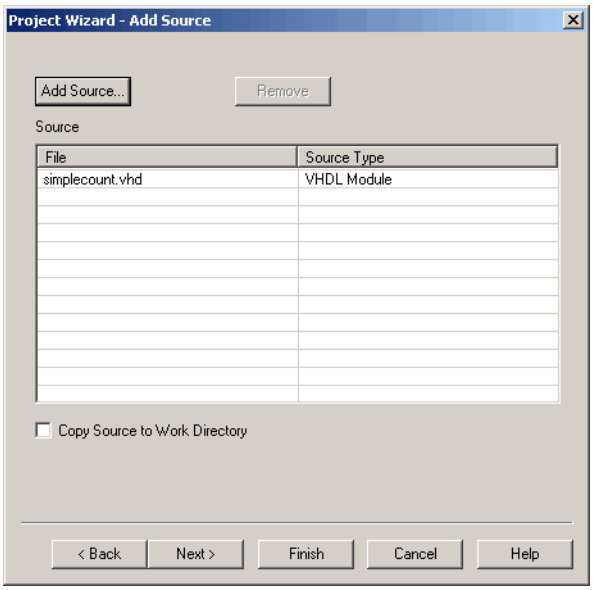
7. In the Import Source Type dialog box, select **VHDL Module**, and click **OK**.

Figure 40: Import Source Type Dialog Box



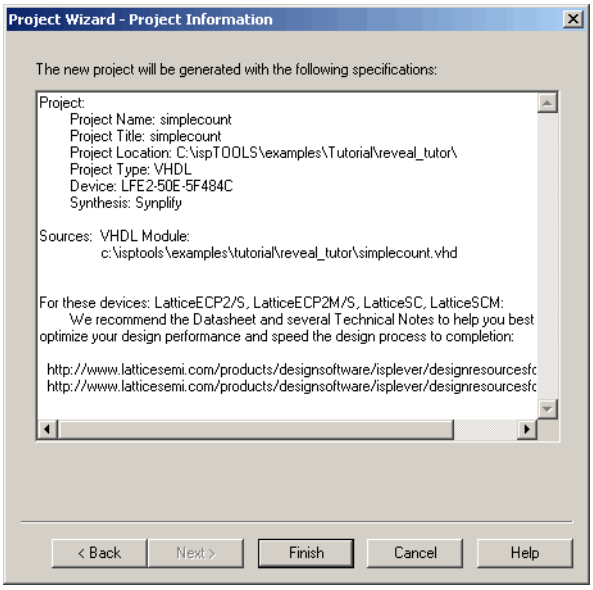
8. In the Project Wizard - Add Source dialog box, shown in Figure 41, click **Next**.

Figure 41: Project Wizard – Add Source Dialog Box



9. In the Project Wizard - Project Information box, shown in Figure 42, click **Finish**.

Figure 42: Project Wizard – Project Information Dialog Box

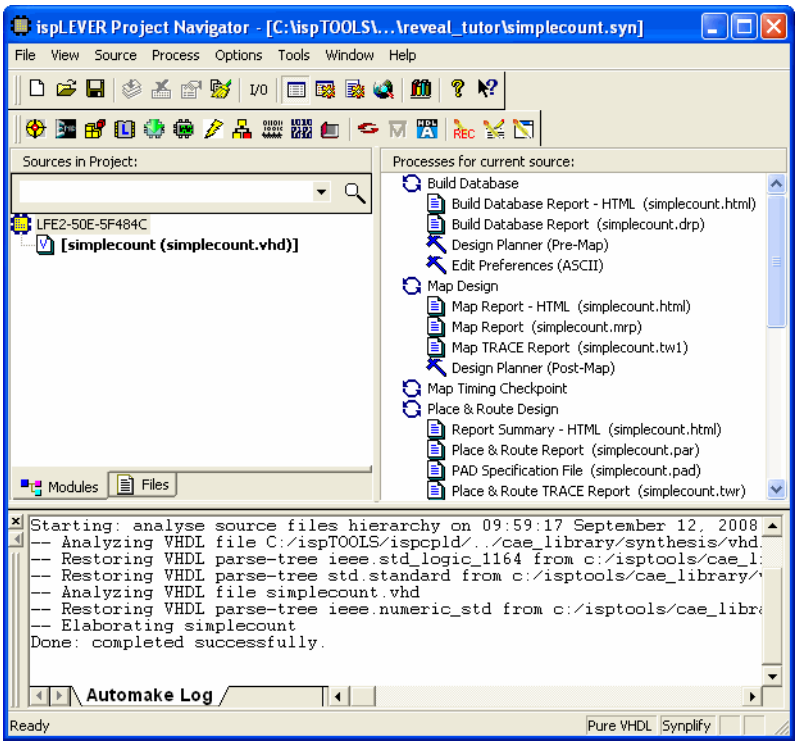


The new VHDL project is added to the Sources in Project window, as shown in Figure 43.

Note:

Click on the part name to see the contents of the Processes for Current Source window.

Figure 43: New Project in ispLEVER Window



Task 2: Generating and Adding the Reveal Core

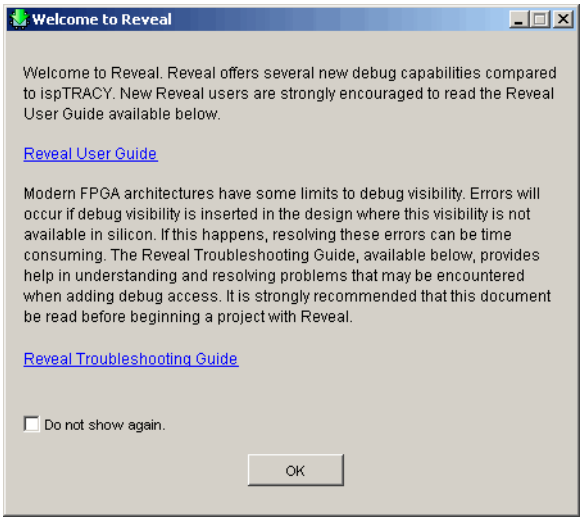
In this task, you will use the Reveal Inserter to configure a Reveal core based on triggering conditions and the desired trace buffer. The primary output of the Reveal Inserter is a modified version of your design with one or more cores instantiated and the core logic ready for mapping, placement, and routing.

To generate and add a Reveal core:

1. In Project Navigator, choose **Tools > Reveal Inserter** or click the button on the Project Navigator tool bar.

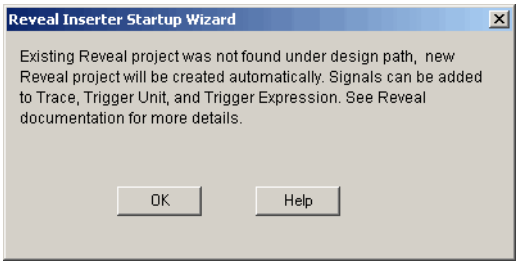
2. In the Welcome to Reveal dialog box, shown in Figure 44, click **OK**.

Figure 44: Welcome to Reveal Dialog Box



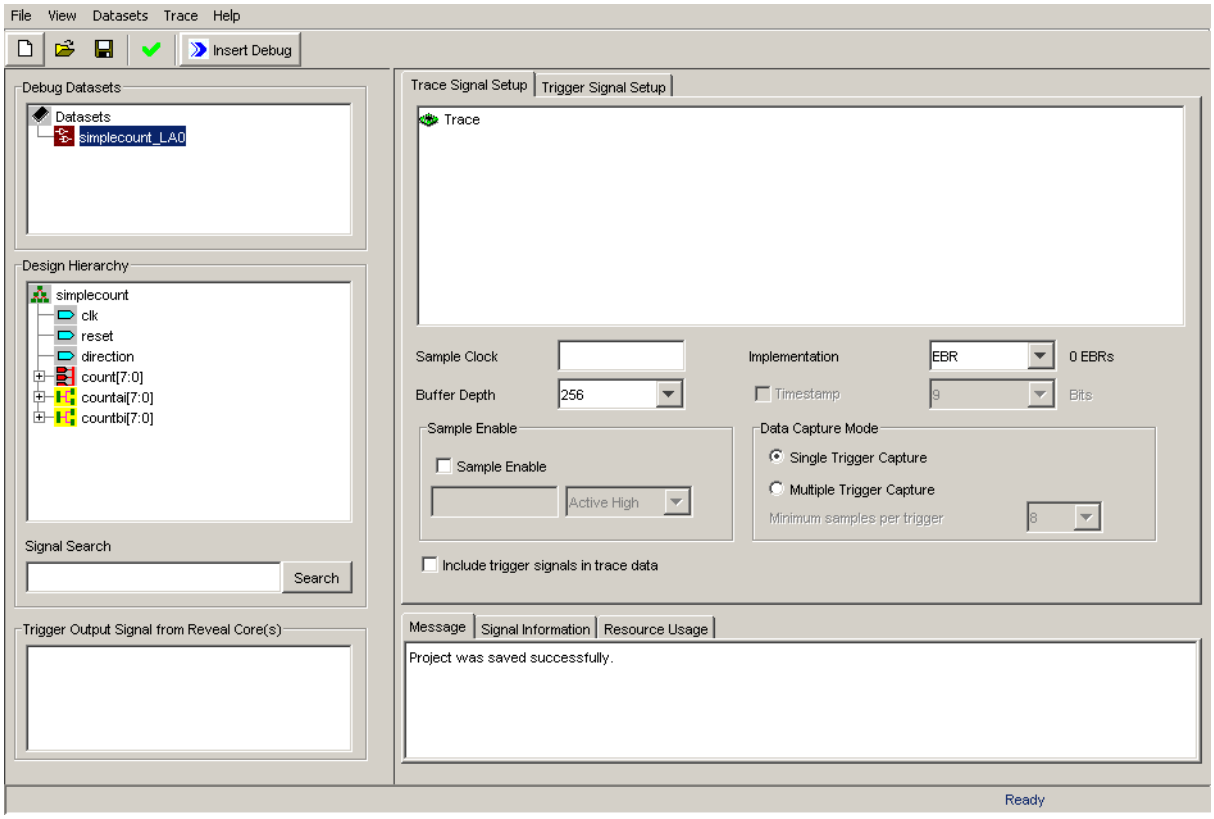
3. In the Reveal Inserter Startup Wizard dialog box, shown in Figure 45, click **OK**.

Figure 45: Reveal Inserter Startup Wizard Dialog Box



The main window of the Reveal Inserter opens, as shown in Figure 46.

Figure 46: Reveal Inserter Main Window

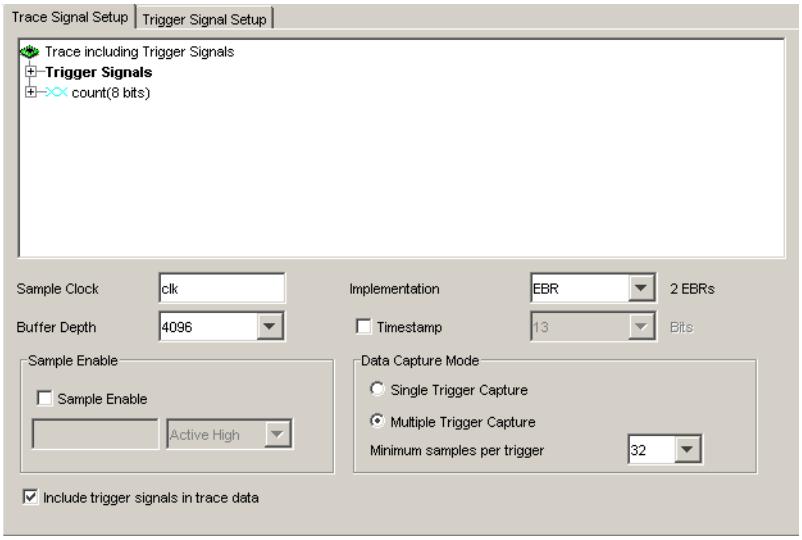


4. Click on the **Trace Signal Setup** tab, if it is not already selected.
5. From the Design Hierarchy pane, drag the count[7:0] bus to the Trace Data pane.
6. Select the **Include trigger signals in trace data** option.

   The bus and the trigger signals now appear in the Trace Data pane, and the name of the bus now appears in bold font in the Design Hierarchy pane.
7. Drag the **clk** signal from the Design Hierarchy pane to the Sample Clock box, or type **clk** in the Sample Clock box.
8. From the pulldown menu in the Buffer Depth box, select **4096**.
9. Set Data Capture Mode to **Multiple Trigger Capture** and Minimum Samples Per Trigger to **32**.

The Trace Signal Setup tab should now resemble the illustration in Figure 47.

Figure 47: Trace Signal Setup Tab



Setting Up the Trigger Signals

Setting Up the Trigger Units

You will set up the trigger units in the Trigger Unit section of the tab.

To set up the trigger units:

1. Click on the **Trigger Signal Setup** tab.

   One line appears in the Trigger Unit section of the tab with a default name of TU1.

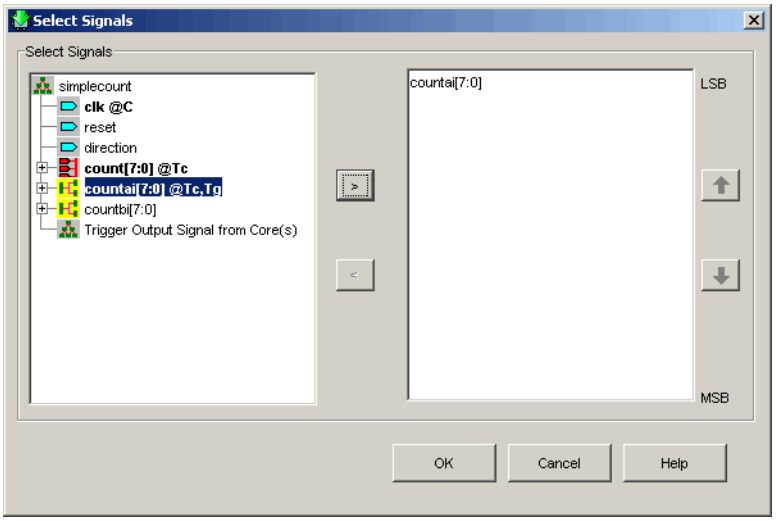
2. Double-click the TU1 name in the Name box, backspace over “TU1,” and type **countai**.

3. Double-click in the Signals (MSB:LSB) box to activate the Select Signals dialog box.

4. In the Select Signals dialog box, select the **countai[7:0]** bus and click **>**.

The countai[7:0] bus now appears in the right-hand pane of the dialog box, as shown in Figure 48.

Figure 48: Select Signals Dialog Box



5. Click **OK** in the dialog box.
6. In the Operator box of the trigger unit, use the default of ==.
7. In the Radix box, select **Hex** from the drop-down menu.
8. In the Value box, double-click, backspace, and type **88**.
9. Click **Add** to add a second trigger unit.
10. In the Name box, double-click TU2, backspace over “TU2,” and type **dir**.
11. Double-click in the Signals (MSB:LSB) box.
12. In the Select Signals dialog box, select **direction**, click **>**, and click **OK**.
13. In the Operator box, select **rising edge** from the drop-down menu.
14. In the Radix box, select the default of **Bin**.
15. In the Value box, double-click, backspace, and type **1**.
16. Click **Add** to add a third trigger unit.
17. In the Name box, double-click TU3, backspace over “TU3,” and type **countbi**.
18. Drag the countbi[7:0] bus from the Design Hierarchy pane to the Signals (MSB:LSB) box.
19. In the Operator box, select **==** from the drop-down menu.
20. In the Radix box, select **Hex**.
21. In the Value box, double-click, backspace, and type **EC**.

Setting Up the Trigger Expressions

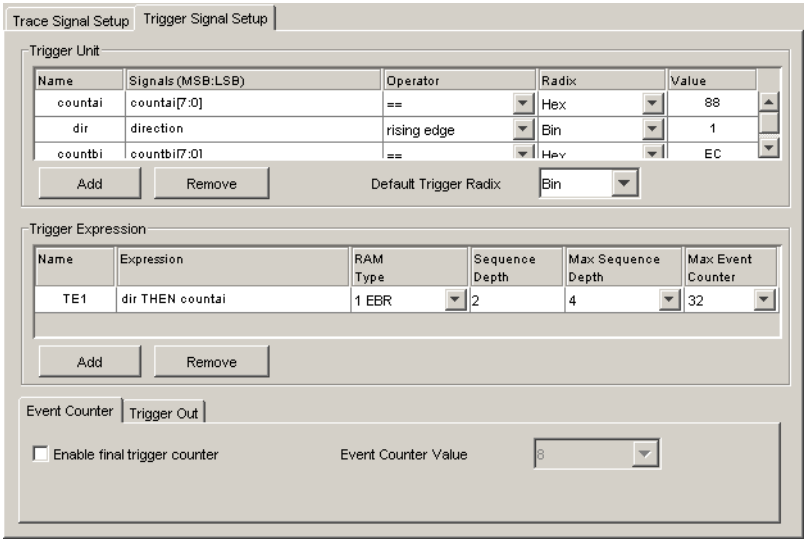
Now you will set up the trigger expressions in the Trigger Expression section of the tab.

To set up the trigger expressions:

1. In the Name box in the Trigger Expressions section, use the default name of TE1.
2. In the Expression box, select the countai and dir trigger units by typing **dir THEN countai**.
3. In the RAM Type box, select **1 EBR** from the drop-down menu.
4. In the Sequence Depth box, make sure a value of **2** appears.
5. In the Max Sequence Depth box, select **4** from the drop-down menu.
6. In the Max Event Counter box, select **32** from the drop-down menu.

   The Trigger Signal Setup tab should now resemble the illustration shown in Figure 49.

Figure 49: Trigger Signal Setup Tab



Inserting the Debug Logic

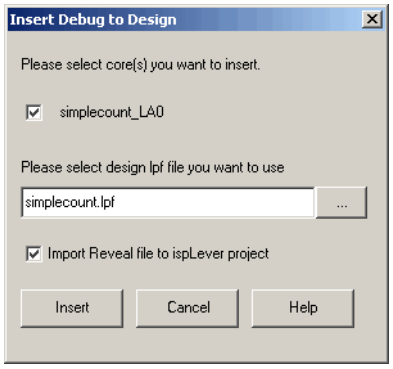
To insert the debug logic:

1. Choose **Datasets > Insert Debug** or click 

.

2. In the Insert Debug to Design dialog box, shown in Figure 50, be sure that the **Import Reveal File to ispLEVER Project** option is selected.

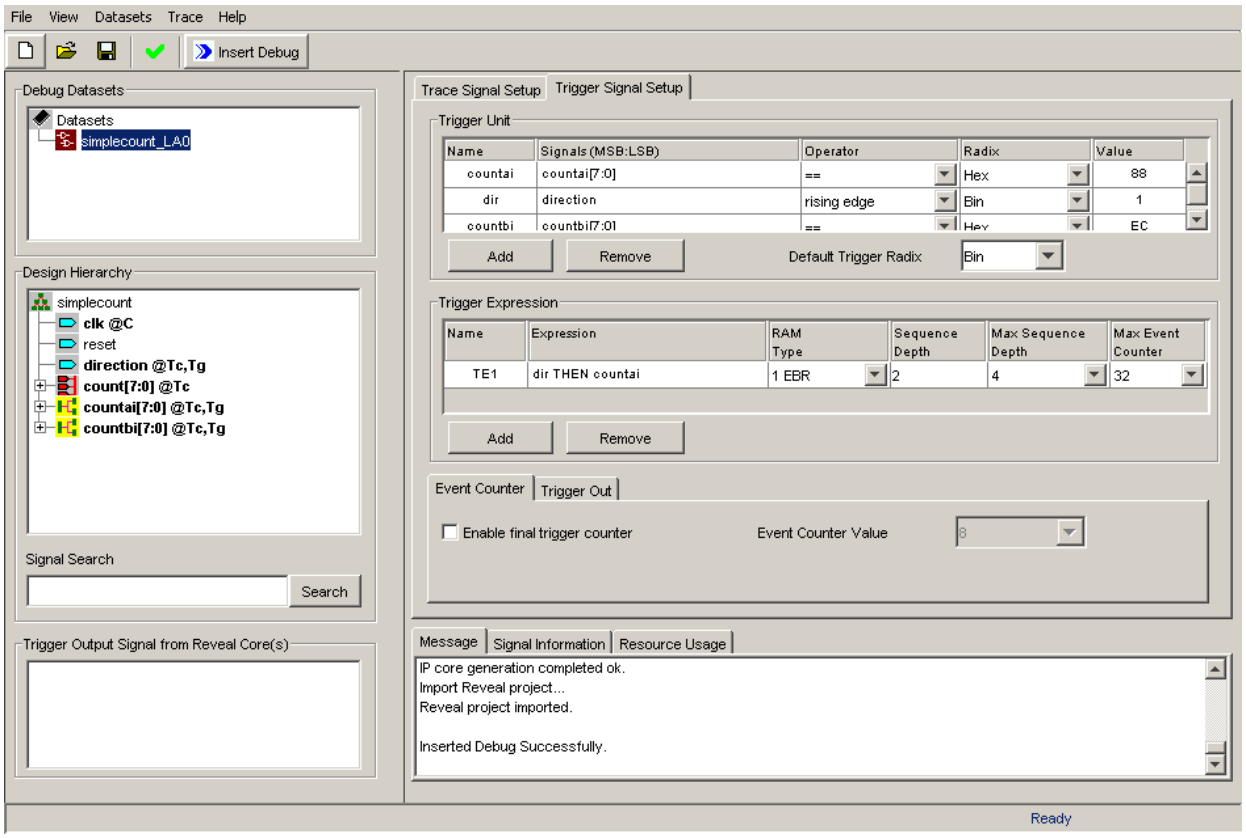
Figure 50: Insert Debug to Design Dialog Box



3. Click **Insert**.

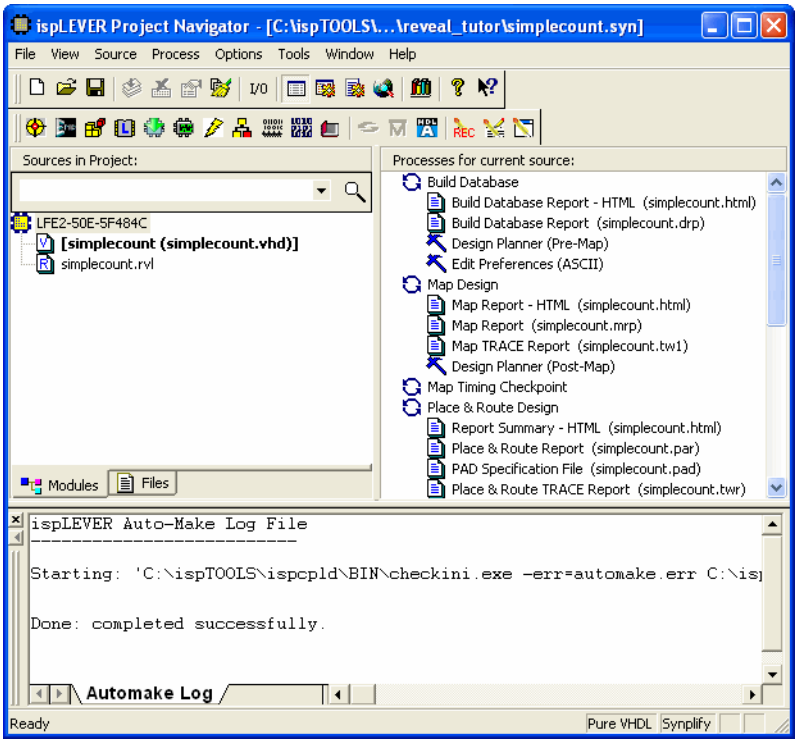
   The Reveal Inserter now invokes the synthesis tool, adds the debug logic, and imports the Reveal project (.rvl) file into ispLEVER. When the process is complete, it displays the messages shown in the Message tab in Figure 51.

Figure 51: Message Tab



The Reveal (.rvl) file is now added to the Sources in Project pane in Project Navigator, as shown in Figure 52.

Figure 52: .rvl File in Project Navigator



4. In the Reveal Inserter window, choose **File > Exit**.

Building the Design Files

In this task, you will build the design files.

To build the design files:

1. In Project Navigator, highlight the device name (**LFE2-50E-5F484C**) in the Sources in Project window.
2. Double-click on **Build Database**.

   Information and warning messages appear in the Automake Log tab of the output panel of Project Navigator. You can ignore these warnings in this tutorial.

   The Build Database process translates the synthesis output in EDIF to an NGD logical design database. Information and warning messages appear in the Automake Log tab of the output panel of Project Navigator. You can ignore these warnings in this tutorial.
3. In the ispLEVER Process message box, click **OK**.

Mapping, Placing, and Routing the Design

Next, you map, place, and route the design before generating a bitstream.

To map, place, and route the design:

1. Double-click the **Map Design** process in the ispLEVER Project Navigator.
2. In the ispLEVER Process message box, click **OK**.
3. Double-click the **Place & Route Design** process in the ispLEVER Project Navigator.

All core clock and reset pins must be located and driven by valid signals for successful hardware debugging.

Generating a Bitstream

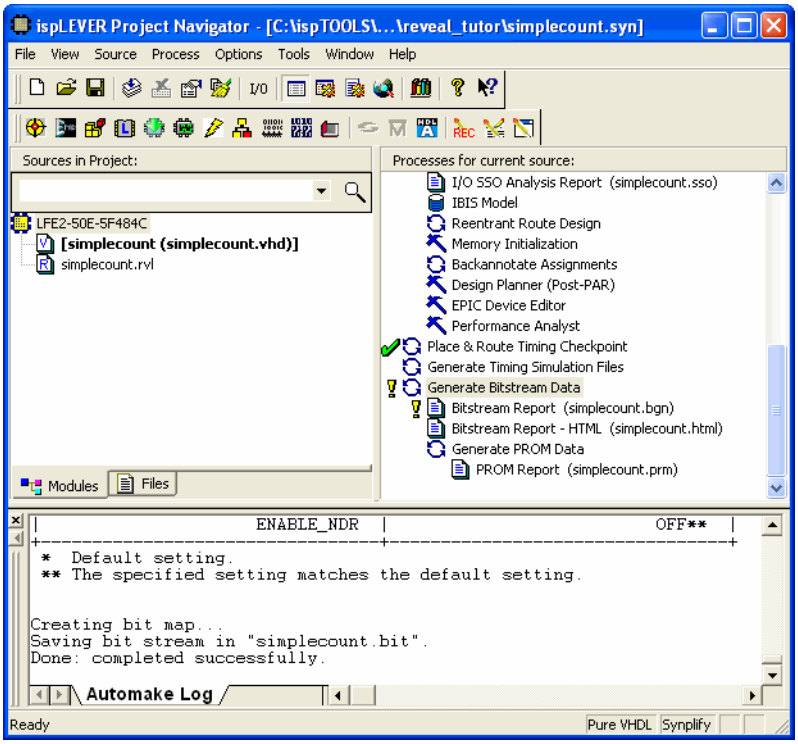
Now you are ready to generate the bitstream (.bit) file.

To generate a bitstream (.bit) file for the evaluation board:

1. Double-click the **Generate Bitstream Data** process in the ispLEVER Project Navigator.
2. In the ispLEVER Process message box, click **OK**.

   IspLEVER creates a programming file, simplecount.bit, that is ready for downloading into the device. Figure 53 shows Project Navigator after bitstream generation.

Figure 53: Project Navigator After Bitstream Generation



Task 3: Programming the Evaluation Board

Using the guidelines of the appropriate board user's guide, such as the *LatticeECP2 Standard Evaluation Board User's Guide*, you will connect the ispDOWNLOAD cable between the JTAG header of the board and the parallel port of your system. You will run the ispVM programming environment to download the bitstream created in the last task to program the device.

The simplecount.lpf file supplied with the tutorial contains location preferences for the following evaluation boards:

- LatticeECP2 standard evaluation board (revision B)
- LatticeECP2 advanced evaluation board (revision C)

The simplecount_xp.lpf file supplied with the tutorial contains location preferences for the following evaluation boards:

- LatticeXP standard evaluation board
- LatticeXP advanced evaluation board

Connecting to the Evaluation Board

A standard LatticeECP2 evaluation board is shown in Figure 54.

Figure 54: Standard LatticeECP2 Evaluation Board

To connect to the LatticeECP2 evaluation board:

1. Install a driver for the download cable.
2. Reboot your computer.
3. Attach the parallel port or USB ispDOWNLOAD cable to the parallel port or USB port of your system.
4. In Project Navigator, select **Tools > ispVM System** or click the button on the toolbar.
5. In ispVM System, select **Options > Cable and IO Port Setup**.
6. Click **Auto Detect**, then click **OK**.

7. Attach the 1 x 8 JTAG connector ispDOWNLOAD cable to JP1 (1x10) of the JTAG programming header of the evaluation board. Justify the alignment of pin 1 (VCC) of the header to the VCC lead of the cable.
8. Align the on-board oscillator so that pad V1 is driven. The 16-pin socket will allow connection to PLL clock pin V1 when the bottom of the oscillator is aligned to socket pins 8 and 9.
9. Plug in the AC adapter to a wall outlet, and plug the other end into the power jack at J31.

Note

You should follow the handling and power-up advice provided in the *LatticeECP2 Standard Evaluation Board (Revision B) User's Guide* when using the evaluation board.

Downloading the Program

This task illustrates the procedure for downloading the programming bitstream to a LatticeECP2 standard evaluation board. The procedure is similar when you use the LatticeECP2 advanced or the LatticeXP standard or advanced builds of the FPGA board. Substitute the appropriate device for your FPGA board in the following steps.

To download the program to the LatticeECP2 evaluation board:

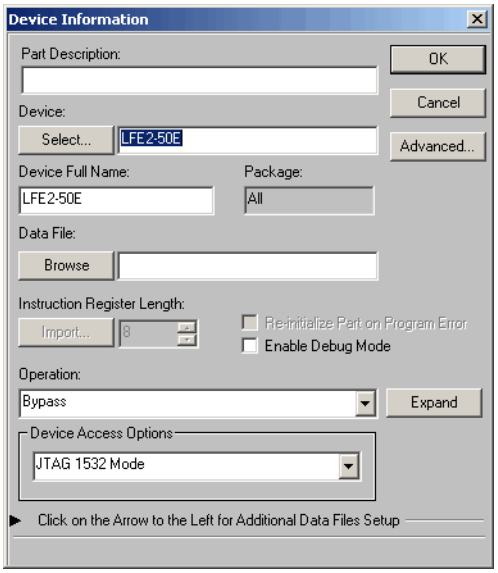
1. In the ispVM System interface, select **File > New**.

   A new chain configuration window appears.
2. Choose **ispTools > Scan Chain** or click the Scan toolbar icon.

   IspVM detects a single LFE2-50E or LFE2-50SE device in the chain and adds it to the list.
3. Click on the LFE2-50E/LFE2-50SE line of the Device List column in the New Scan Configuration Setup dialog box.
4. In the Multi Match Device's ID List box, select **LFE2-50E**.
5. Double-click in the File Name-IR Length box next to the device.

The Device Information dialog box appears, as shown in Figure 55.

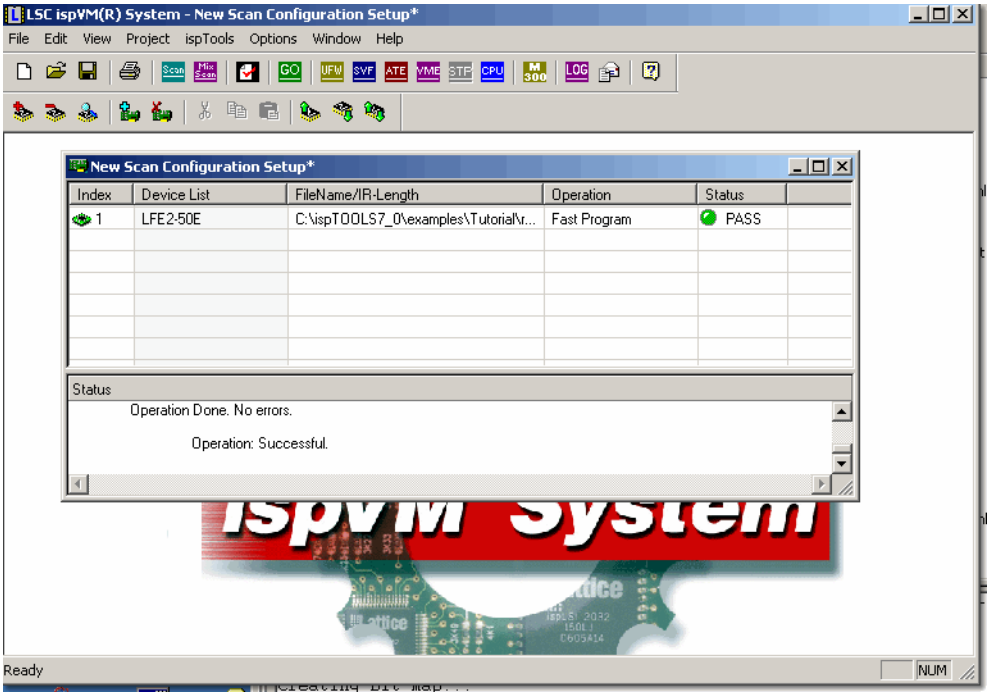
Figure 55: Device Information Dialog Box



6. Click the **Select** button of the Device section to open the Select Device dialog box.
7. Select the following:
   - Device Family: **LatticeECP2**
   - Device: **LFE2-50E**, as appropriate for your revision of the LatticeECP2 standard evaluation board
   - Package: **484-ball fpBGA**
8. Click **OK**.
9. Click the **Browse** button of the Data File section.
10. Select the **simplecount.bit** file, and click **Open**.
11. In the Operation box, select **Fast Program**, if it is not already selected.
12. Click **OK** to close the Device Information dialog box.
13. Choose **Project > Download**, or click the **GO** button on the toolbar.

After a few moments, the download and programming activity will end. A green PASS button appears in the New Scan Configuration Setup dialog box, as shown in Figure 56.

Figure 56: New Scan Configuration Setup Dialog Box



14. Select **File > Save As** to save the configuration setup as an .xcf file.
15. In the File Name box in the dialog box that appears, type **simplecount.xcf**, make sure that **Configuration File (*.xcf)** is selected in the Save as Type box, and click **Save**.
16. Choose **File > Exit** in the LSC ispVM System window.

Task 4: Performing Logic Analysis

In this task, you will use the Reveal Logic Analyzer to set up trigger conditions and view trace buffer data from the on-chip Reveal core operating within the device on the LatticeECP2 standard evaluation board. The trigger setup influences under what specific conditions and how the Reveal core trace signal states are displayed in the Reveal Logic Analyzer's graphical user interface. In this task, you will explore just a few of the many ways to trigger and trace the system.

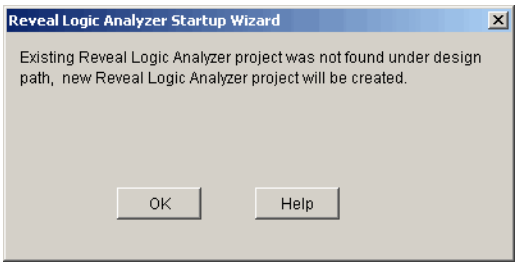
Creating a New Reveal Logic Analyzer Project

To create a new Reveal Logic Analyzer project:

1. Since the reset is active high, push down the switch on the board that is connected to the reset signal for the design. On the LatticeECP2 board, this switch is the rightmost switch in the row of switches.
2. In Project Navigator, choose **Tools > Reveal Logic Analyzer** or click the  button on the toolbar.

   The Reveal Logic Analyzer Startup Wizard dialog box appears, as shown in Figure 57.

Figure 57: Reveal Logic Analyzer Startup Wizard Dialog Box



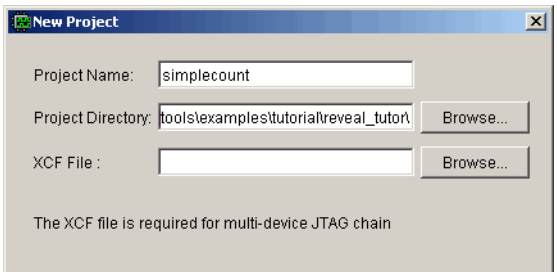
3. Click **OK**.

   The New Project dialog box now appears.
4. Specify the following in the New Project dialog box, shown in Figure 58:
   a. In the Project Name box, enter **simplecount**.
   b. In the Project Directory box, browse to or enter the following path:

      <*install_path*>\examples\tutorial**reveal_tutor**

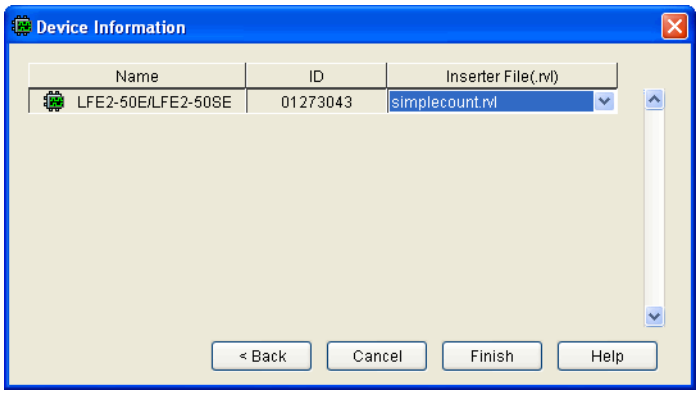
c. Click the **Next** button.

Figure 58: New Project Dialog Box



5. In the Device Information dialog box, shown in Figure 59, make sure the the Inserter File (.rvl) column shows **simplecount.rvl**.

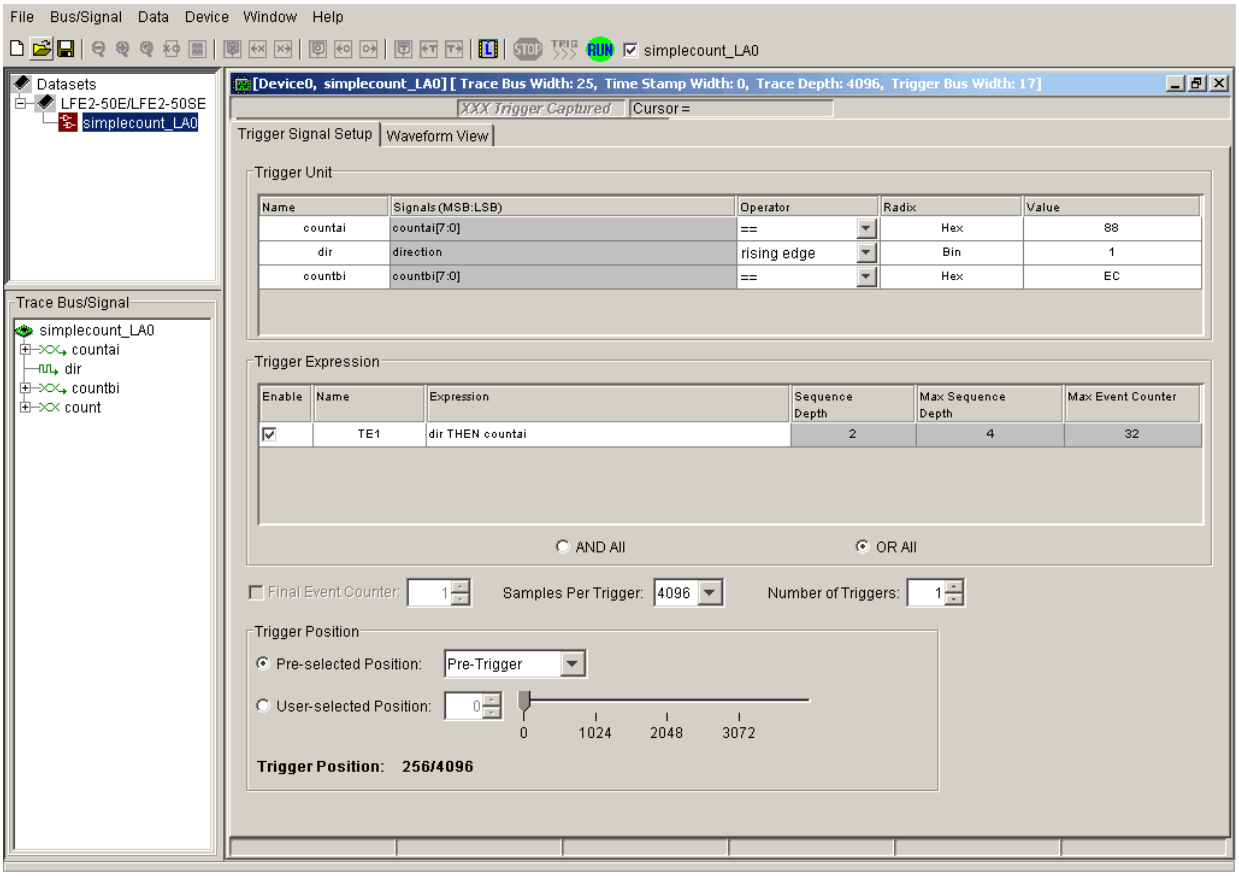
Figure 59: Device Information Dialog Box



6. In the Device Information dialog box, click **Finish**.

The Reveal Logic Analyzer main window now appears with the Trigger Signal Setup tab selected, as shown in Figure 60. It contains the same trigger units and trigger expressions that you set up in the Reveal Inserter.

Figure 60: Initial Trigger Signal Setup Tab in Reveal Logic Analyzer Main Window



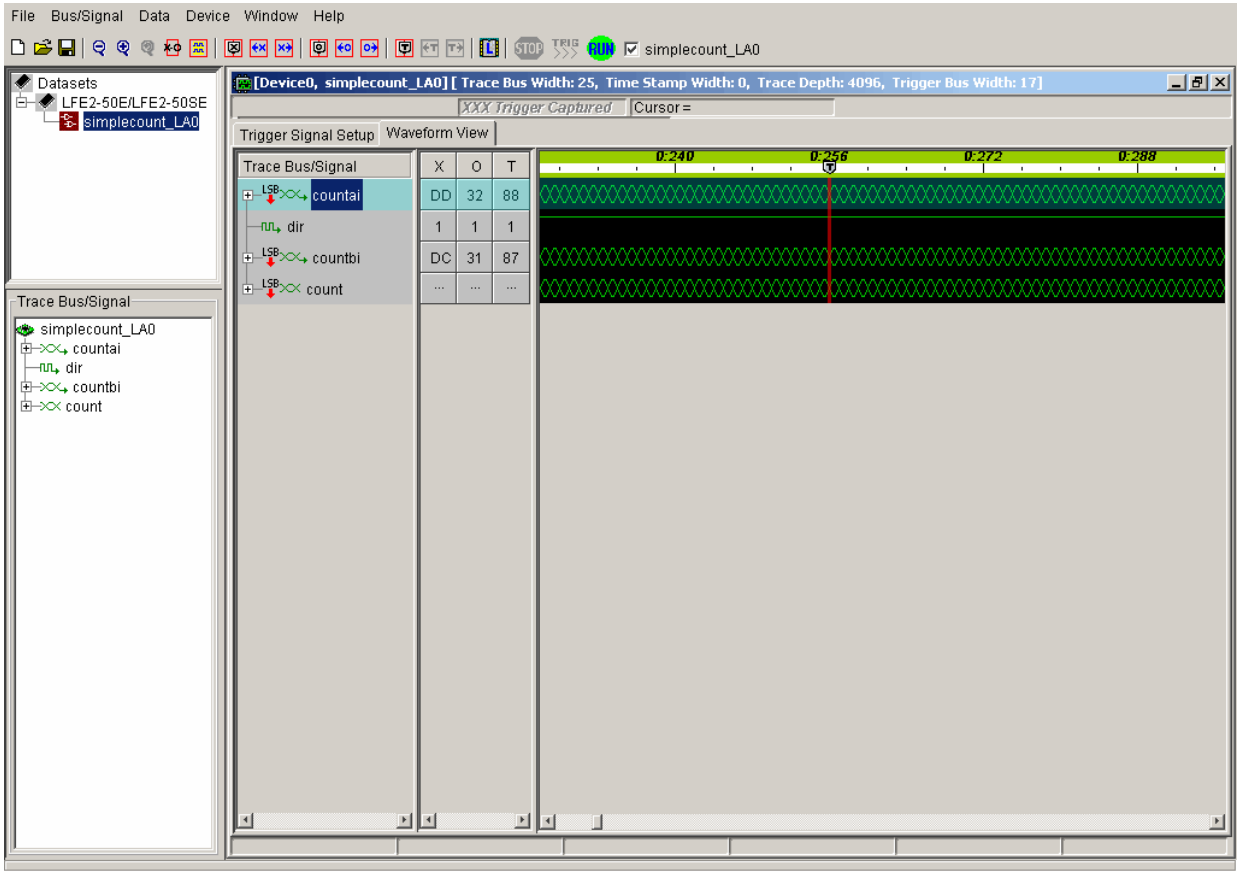
Running the Logic Analyzer

To run the Reveal Logic Analyzer:

1. Choose **Device > Run Selected LAs** or click RUN.
2. On the row of eight switches on the blue signal box on your board, push down the seventh switch from the left to set the direction signal low. Then pull the same switch up to bring the signal level high, generating a rising edge that will cause the “dir” trigger unit to be true. The trigger expression can now evaluate the next trigger unit and generate a trigger for data to be captured.
3. Click the **Waveform View** tab.

You now see the waveforms displayed, as shown in Figure 61.

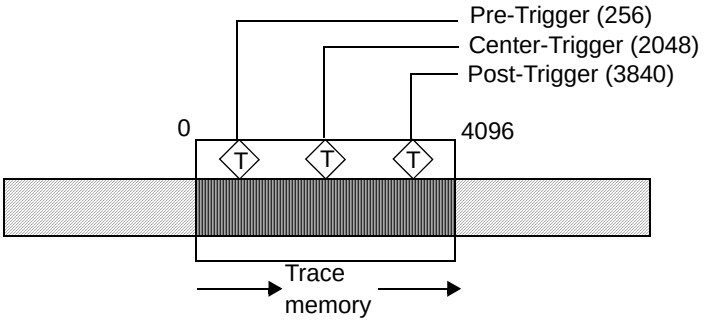
Figure 61: Initial Waveform Display



Using Single Trigger Capture Mode

Single Trigger Capture is an option in the Trace Signal Setup tab of the Reveal Inserter. The way it works is illustrated in Figure 62. The entire trace buffer is available to capture events around a single trigger point. You can use the entire buffer or a smaller amount if you only want to capture a smaller number of samples. You can use the Position option to determine the relative amount of past or future samples that you would like to examine relative to the trigger position.

Figure 62: Determining Number of Samples



The Pre-Trigger setting is helpful if you care mostly about data states that occurred after the trigger. With this setting, only 256 samples of data that occur before the trigger are stored; however, 3840 samples of data that occur after the trigger are stored. The Post-Trigger setting provides the most trace data on states that occurred before the trigger event, and the Center-Trigger setting provides equal amounts of trace data.

Summary

You have completed the “Reveal On-Chip Debug Tutorial.” In this tutorial, you have learned how to do the following:

- Specify the signals to use for tracing your design and for generating trigger events, and insert the debug logic into your design using the Reveal Inserter.
- Generate a bitstream file for the target device on the LatticeECP2 standard evaluation board.
- Program a standard LatticeECP2 standard evaluation board with ispVM.
- Run the Reveal Logic Analyzer application to examine the sample captures in the internal trace memory array.

Glossary

Following are the terms and concepts that you should understand to use this tutorial effectively.

.bit file A .bit file is a binary-format configuration file containing the default outputs of the bit generation process. It is used to program a Lattice Semiconductor FPGA device.

logic analysis core A logic analysis, or debugging, core is a precharacterized piece of logic (IP) specifically designed for logic analysis.

.ncd file An .ncd file is a binary-format FPGA post-map physical design database file generated by the Map Design process of Project Navigator. The .ncd file includes mapping information and, potentially, placement and routing information.

.ngd file An .ngd file is a binary-format FPGA pre-map logical design database file generated by the NGDBuild phase of the Build Database process in Project Navigator. The .ngd file represents the logical design information of the ASCII EDIF netlist.

.ngo file An .ngo file is a binary-format FPGA pre-map logical design database file generated by the EDIF-to-NGD translation phase in the Build Database process in Project Navigator. It contains all of the data in the input .ngd file, as well as information on the physical design produced by the mapping. The .ngo file typically represents one or more hierarchical branches of a larger logical design. In this tutorial, an .ngo file is produced for the Reveal logic analysis core.

.rva file An .rva file is the project file created by the Reveal Logic Analyzer. It contains the project name, core configuration trace and trigger signals, the trigger and event settings, and the Reveal chain acquisition data.

.rvt file The .rvt file is the Reveal token file, which stores the tokens and token sets that you created in the graphical user interface This token file is in XML format and uses the same syntax as the token section in the .rvs file.

slice A slice is an architectural element within an FPGA consisting of two LUT4 lookup tables that feed two registers (programmed to be in FF or latch mode), and some associated logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7, and LUT8. It also includes control logic to perform set/reset functions (programmable as synchronous or asynchronous), clock select, chip-select and wider RAM/ROM functions. The registers in the slice can be configured for positive or negative and edge or level clocks. There are four interconnected slices per PFU block. Each slice in a PFU is capable of four modes of operation: logic, ripple, RAM, and ROM. Each slice in the PFF is capable of all modes except RAM.

trace A trace is one or more signal states that are monitored for debugging purposes. The Reveal Logic Analyzer enables you to monitor signal states that occur before and after a user-defined trigger pattern. IPexpress enables you to define the width and depth of a trace memory to hold the signal states.

trigger A trigger is a specific state or range of states that cause a trace to be stored for review and debugging. The Reveal Logic Analyzer enables you to establish different scenarios for a trigger condition, given a set of trigger signals.

.xcf file An .xcf file is a scan chain configuration file generated by ispVM 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.

Recommended Reference Materials

The following reference materials are recommended to supplement this tutorial:

- *LatticeECP2 Family Handbook*
- *LatticeXP Family Handbook*
- LatticeECP2 Family Data Sheet
- LatticeXP Family Data Sheet
- *LatticeECP2 Standard Evaluation Board (Revision B) User's Guide*
- *LatticeECP2 Advanced Evaluation Board (Revision C) User's Guide*
- *LatticeXP Standard Evaluation Board User's Guide*
- *LatticeXP Advanced Evaluation Board User's Guide*
- ispLEVER online Help for the Reveal Inserter and the Reveal Logic Analyzer