

Reveal User Guide for Radiant Software



October 8, 2026

Copyright

Copyright © 2026 Lattice Semiconductor Corporation. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Lattice Semiconductor Corporation (“Lattice”).

Trademarks

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

Disclaimers

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

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

Type Conventions Used in This Document

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

Contents

Chapter 1	Introduction	7
Chapter 2	Reveal Inserter	9
	Using JTAG Debugger	9
	Debug Flows	10
	Input and Output Files	16
	Limitations	17
	Getting Started	19
	Starting Reveal Inserter	19
	Creating a New Reveal Inserter Project	20
	Opening an Existing Reveal Inserter Project	22
	Using the Reveal Controller Simulation Model	22
	Managing the Cores in a Project	23
	Renaming a Core	24
	Removing a Core	24
	Viewing Signals in the Design Tree Pane	24
	Searching for Signals	25
	Setting Up the Trace Signals	26
	Selecting the Debug Logic Core	26
	Selecting the Trace Signals	26
	Viewing Trace Signals and Buses	27
	Grouping Trace Signals into a Bus	28
	Ungrouping Trace Signals in a Bus	28
	Removing Signals and Buses from the Trace Data Pane	28
	Renaming a Bus	28
	Setting Required Sample Parameters	29
	Power-on Reset (POR) Debug	30
	Setting Sample Options	30
	Setting Up the Trigger Signals	32
	Triggering	32
	Adding Trigger Units	41
	Renaming Trigger Units	41

Setting Up Trigger Units	42
Removing Trigger Units	44
Adding Trigger Expressions	44
Renaming Trigger Expressions	45
Setting Up Trigger Expressions	45
Removing Trigger Expressions	47
Setting Up Virtual Switch/LED Settings	48
Configuring User Memory Setup	48
Configuring User Control Register Setup	50
Configuring User Status Register Setup	52
Configuring Hard IP Setup	53
Checking the Debug Logic Settings	54
Saving a Project	55
Inserting the Debug Logic Cores	56
Removing Debug Logic from the Design	56
Closing a Project	57
Exiting Reveal Inserter	57
Performing Logic Analysis with Reveal Analyzer	58
Using JTAGhub	58
JTAGhub Input and Output Ports	62
JTAGhub Ecosystem	64
JTAGhub Usage and Design Examples	65
JTAGhub Addressing Scheme	66
JTAGhub CORES	66
Setting Parameters and Connectivity	67
JTAG BYPASS Instruction in SOFTJTAG Chain	68
Using JTAGMON	70
JTAGMON Input and Output Ports	70
JTAGMON Design Flow	71
User Interface Descriptions	72
Chapter 3 Reveal Analyzer	74
About Reveal Analyzer	75
Reveal On-Chip Debug Design Flow	75
Inputs	76
Outputs	76
Inserting the Debug Logic	77
Mapping, Placing, and Routing the Design	77
Generating a Bitstream File	77
Connecting to the Evaluation Board	78
Downloading a Design onto the Device	78
Starting Reveal Analyzer	79
Starting with a New File	80
Starting with an Existing File	81
Changing the Cable Connection	82
Selecting a Reveal Analyzer Core	82
Setting Up the Trace Signals	83
Setting the Trace Bus Radix	83
Adding Time Stamps to Trace Samples	83

Setting Up the Trigger Signals	84
Renaming Trigger Units	84
Setting Up Trigger Units	84
Renaming Trigger Expressions	86
Setting Up Trigger Expressions	86
Setting Trigger Options	87
Creating Token Sets	88
Debugging with Reveal Controller	89
Performing Logic Analysis	105
Data Capture with Sample Enable	106
Common Error Conditions	106
Stopping a Logic Analysis	106
Using Manual Triggering	107
Viewing Waveforms	107
Viewing Logic Analysis	107
Adjusting the Waveform Display	108
Panning	108
Zooming In and Out	108
Incremental Display for Multiple Trigger Capture	110
Specifying the Clock Period	111
Placing, Moving, and Locating Cursors	112
Counting Samples	113
Exporting Waveform Data	113
Saving a Project	114
Exiting Reveal Analyzer	114
User Interface Descriptions	115
LA Trigger Tab	115
LA Waveform Tab	120
Viewing and Saving Waveforms	121
Using the Memory Controller Debug	122
Chapter 4 Best Practices	126
General Guidelines	127
Best Practices for Post-Synthesis Flow	128
Guidelines when Updating User Design in Post-Synthesis Debug Flow	128
Guidelines when Changing the Synthesis Tool	129
Revision History	130

Introduction

The Radiant software contains the Reveal Inserter and the Reveal Analyzer tools used for design debugging. This guide describes Reveal Inserter and Reveal Analyzer.

One of the most common activities in debugging is logic analysis. To do this, use Reveal Inserter and Reveal Analyzer. You can use both with all supported FPGA devices.

Reveal continuously monitors signals within the FPGA for specific conditions, which can range from simple to quite complex. When the trigger condition occurs, Reveal can save signal values preceding, during, and following the event for analysis, including a waveform presentation. The data can be saved to a value change dump file (.vcd), which can be used with tools such as QuestaSim, or to an ASCII tabular format that can be used with tools such as Excel.

Before running Reveal Analyzer, use Reveal Inserter to add Reveal modules to your design. In these modules, specify the signals to monitor, define the trigger conditions, and other options. Reveal supports multiple logic analyzer cores using hard/soft JTAG interface. You can have up to 15 modules, typically one for each clock region of interest. When the modules are set up, regenerate the bitstream data file to program the FPGA.

A feature is enabling an added module called Reveal Controller. This controller module enables:

- ▶ Access to the control and status registers of the hard IPs such as I2CFIFO, PLL, PCIe, CDR, DPHY, and USB23 via the LMMI (Lattice Memory Mapped Interface) slave interface.
- ▶ Virtual switches and LEDs emulating on-board switches and LEDs to control and monitor a user design. Up to 32 switches and 32 LEDs are supported.

- ▶ Read/write access to a bank of user registers and/or initialize memory post-configuration.

The main purpose is narrowing down to problem areas during debug cycles using a divide and conquer method into many small functional blocks to control and monitor the status of each block.

Before starting a test run, set up Reveal Analyzer. This includes setting a number of options, including modifying trigger conditions and customizing the waveform display. You can save these settings for later use. During and after a test run, view the incoming data in Reveal's LA Waveform view. You can also save the data to a .vcd or .txt file to analyze with other tools.

Note:

Reveal supports IEEE-P1735 encryption. If this encryption is applied to a design, the design tree will allow only the visible ports and signals that are not encrypted to be inserted by the Reveal Inserter for triggering purposes.

Reveal 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 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 Analyzer, you must use Reveal Inserter to allow debug access.

Reveal 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 Analyzer displays the data captured on the silicon according to the settings that you specify.

Reveal 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 Analyzer, Reveal Analyzer includes a demonstration design that you can run without the evaluation board so that you can learn how to use the tool.

Reveal Analyzer requires the Programmer 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.

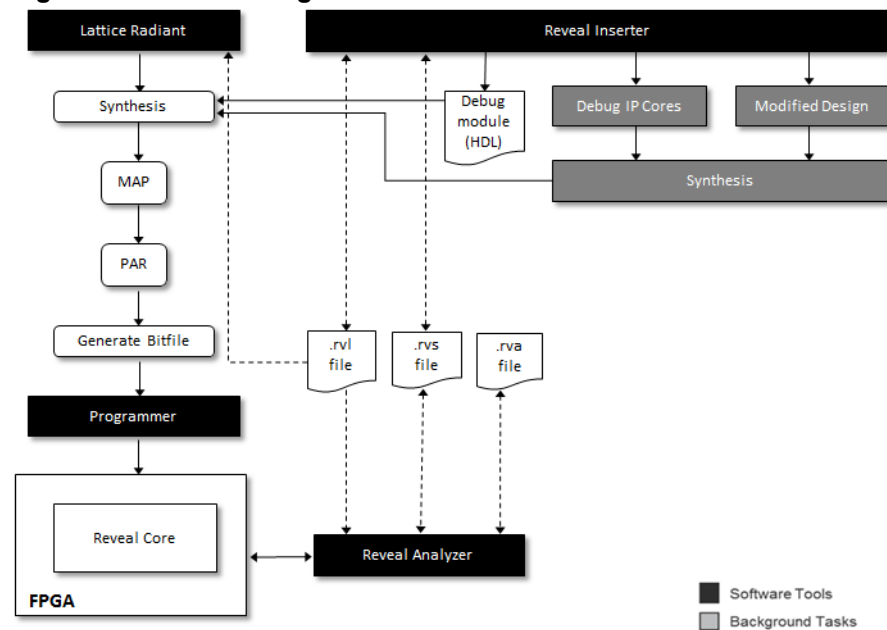
About Reveal Analyzer

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

Reveal On-Chip Debug Design Flow

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

Figure 13: Reveal Design Flow



Before accessing Reveal 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 78](#) for information on this procedure.

You do not need to install a cable and a power supply if you want to run the demonstration design that comes with Reveal Analyzer so that you can learn how to use the tool.

You must also have a design project 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 Process view.

5. Map, place, and route the design.
6. Generate the bitstream data (.bin file).
7. Set up a cable connection.
8. Download the design onto the device by using Programmer.
9. Start Reveal Analyzer.
10. Create a new Reveal 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 to perform the logic analysis, and wait for the design to trigger and download the trace information into Reveal 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 Analyzer requires the following as input:

- ▶ A design project
- ▶ A Reveal Analyzer settings (.rvs) file, which is output by Reveal Inserter or Reveal 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 Analyzer.

A Reveal Analyzer project (.rva) file, which is the project file output by Reveal Analyzer in a previous session. It contains the information used by Reveal Analyzer, such as window settings, waveform trace signal positions, radixes, markers, and signal colors.

Outputs

Reveal Analyzer generates the following files:

- ▶ A Reveal 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 Analyzer trace (.trc) file, which contains the waveform information acquired from previous runs of Reveal Analyzer. When you first open Reveal Analyzer, the waveform displays this information until

you press the Run button. If the debug signals have been changed from a previous Reveal 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 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.

Mapping, Placing, and Routing the Design

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

To map, place, and route the design:

1. Double-click the **Map Design** process in the Process view.
2. Double-click the **Place & Route Design** process in the Process view.

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

Generating a Bitstream File

Now you generate a bitstream (.bin) file, as appropriate, to download into the device.

To generate a bitstream file:

- ▶ Double-click the **Export Files** process in the Process view.

This process creates a .bin file that is ready for downloading into the device.

Connecting to the Evaluation Board

Reveal 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 Programmer.

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 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 the Radiant software, choose **Tools > Programmer**.
6. The Cable Setup Window view is default open.
7. Click **Detect Cable** button.
8. Attach the JTAG connector cable to the appropriate JTAG programming header of the evaluation board. See the device evaluation board documentation for details.

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

Downloading a Design onto the Device

To download a design onto the device, use Programmer. This process creates a scan chain configuration (.xcf) file. Reveal Analyzer derives the information in the downloaded design directly from the device on the board.

To download the design onto the device:

1. In Programmer, select **File > New**.
A new programmer project window appears, enter Filename.
2. Click on Detect Cable button.
3. If there are multiple cables, select one from the Multiple Cables Detected dialog drop down list.
4. Click the ... button under File Name section.

5. Select the `<design_name>.bin` file, and click **Open**.
6. Double click In the Operation box, a new Device Properties dialog appears. Select **Fast Configuration**, in the Operation drop down list.
7. Click **OK** to close the Device Properties dialog box.
8. Choose **Run > Program Device**, or click the **Program Device** button on the toolbar.

After a few moments, the download and programming activity will end. A green PASS button appears in the Status field.
9. Select **File > Save Project As** to save the configuration setup as an .xcf file.
10. The .xcf file must reside in the design project directory for Reveal Analyzer to use it. In the File Name box in the dialog box that appears, type in `<design_name>.xcf`, and click **Save**.
11. Choose **File > Exit** in Programmer.

See the Programmer Help for detailed instructions on the downloading process.

Starting Reveal Analyzer

Before starting Reveal Analyzer you need to decide if you want to work with a new Reveal Analyzer (.rva) file or an existing one. The .rva file defines the Reveal Analyzer project and contains data about the display of signals in the LA Waveform view. You may want to start Reveal Analyzer with a new file to set up a new test. Start with an existing file to rerun a test, to set up a new test based on existing settings, or to just view the waveforms from an earlier test. (See ["Starting with an Existing File" on page 81.](#))

How you start Reveal Analyzer also depends on whether you are using it integrated with the Radiant software or using the stand-alone version, and on your operating system.

Starting with a New File

Before you can start Reveal Analyzer with a new .rva file, you need to be connected to your evaluation board with a download cable and have the board's power turned on.

To start Reveal Analyzer with a new file:

1. Issue the start command:
 - ▶ For integrated with the Radiant software, go to the Radiant software main window and choose **Tools** >  **Reveal Analyzer**.
 - ▶ For stand-alone in Windows, go to the Windows Start menu and choose **Programs** > **Lattice Radiant Reveal** > **Lattice Reveal Logic Analyzer**.
 - ▶ For stand-alone in Linux, go to a command line and enter the following:

```
<Reveal install path>/bin/linux64/revealrva
```

2. If Reveal Analyzer opens with an existing file, choose **File** > **Save** <file> **As**.

The Save Reveal Analyzer File dialog box opens. Change the filename and click **Save**. You now have a new .rva file ready to work with.

3. (Stand-alone only) In the Reveal Analyzer Startup Wizard dialog box, browse to the implementation directory. This is where the Reveal Inserter project (.rvl) file should be and where the .rva file will be created.
4. In the Reveal Analyzer Startup Wizard dialog box, select **Create a new file** (at the upper-left of the dialog box).

The dialog box presents a few rows of boxes that need to be filled in.

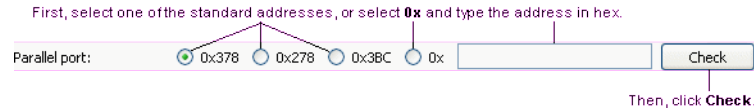
5. In the next row, type in the base name of the file. The extension is added automatically.
6. To the right of this row is a drop-down menu. Choose the type of cable that your board is connected to.

Another row in the dialog box changes to select the port.

7. Select the specific port. The method depends on the port type:
 - ▶ If USB, click **Detect**. Then choose from the active ports found. The following figure shows the second row after choosing a USB type.



- ▶ If parallel, select the port address. If it's not one of the standard addresses given, select **0x** and type in the hexadecimal address. Then click **Check** to verify that the connection is working. The following figure shows the second row after choosing a parallel type.



8. Click **Browse** in the RVL source row to find the Reveal Inserter project (.rvl) file.
9. To add the new .rva file to the File List view, select **Import file into current implementation**. (Not available in stand-alone.) The .rva file works the same either way.
10. Click **OK**.

Starting with an Existing File

If you want to start with an existing file, you just need to have that .rva file in the design project. You need to be connected to the evaluation board only if you want to run a test and capture data.

To start Reveal Analyzer with an existing file:

1. Issue the start command. To start:
 - ▶ In the Radiant software main window, choose **Tools** >  **Reveal Analyzer**.
 - ▶ The stand-alone Reveal Analyzer in Windows, go to the Windows Start menu and choose **Programs** > **Lattice Radiant Reveal** > **Lattice Reveal Logic Analyzer**.
 - ▶ The stand-alone Reveal Analyzer in Linux, enter the following on a command line:

`<Reveal install path>/bin/linux64/revealrva`

If Reveal Analyzer opens with the .rva file you want to use, you're ready to go. Otherwise continue with the following steps.

2. (Stand-alone only) In the Reveal Analyzer Startup Wizard dialog box, browse to the implementation directory. This is where the Reveal Inserter project (.rvl) file and where the .rva file should be.
3. In the Reveal Analyzer Startup Wizard dialog box, select **Open an existing file** (in the lower part of the dialog box).
4. In the "File name" box, choose one of the available .rva files.
5. If the file you want is not in the menu, click **Browse** and browse to the desired .rva file.
6. To add the new .rva file to the File List view, select **Import file into current implementation**. (Not available in stand-alone.) The .rva file works the same either way.
7. Click **OK**.

If the connection to your evaluation board has changed, either in the cable type or the computer port used, you need to tell Reveal Analyzer about the new connection. See [“Changing the Cable Connection” on page 82](#).

Changing the Cable Connection

If you need to change how your evaluation board is connected to your computer, go ahead and make the change. Then go through the following procedure to change the Reveal Analyzer project.

To change the cable setting in a Reveal Analyzer project:

1. Make sure your evaluation board is connected and that its power is on.
2. If Reveal Analyzer is not already open, start it as described in [“Starting with an Existing File” on page 81](#).
3. Choose **Design > Cable Connection Manager**.
The Cable Connection Manager dialog box opens.
4. In the dialog box, choose the cable type.
The second row in the dialog box changes to select the specific port.
5. Select the specific port. The method depends on the port type:
 - ▶ If USB, click **Detect**. Then choose from the active ports found.
 - ▶ If parallel, select the port address. If it's not one of the standard addresses given, select **0x** and type in the hexadecimal address. Then click **Check** to verify that the connection is working.
6. To change the clock speed of the cable connection, adjust the value of TCK Low Pulse Width Delay.
7. Click **OK**.

Selecting a Reveal Analyzer Core

After you have created a project, each Reveal core in the design will have a Reveal Analyzer window available to set triggers and view captured data.

To display a Reveal Analyzer core:

- ▶ Choose the only core from the check box. Only one core is supported.

Setting Up the Trace Signals

Although you can add trace signals only in Reveal Inserter, you can set radices for them by using the LA Waveform tab.

Setting the Trace Bus Radix

You can set the radix of a trace bus displayed in the LA Waveform tab. You can choose a binary, octal, decimal, or hexadecimal radix. You can also use any token set whose bit width matches the bus.

To set the bus radix of a signal bus:

1. In the LA Waveform tab, select one or more buses.
To select one bus, click on it. To select more than one, Control-click on each one. To select all buses in a range, click on one end of the range and Shift-click the other end. If you want to change all the signals in the waveform to the same radix, you do not need to select anything.
2. Right-click in one of the selected waveforms and choose **Set Bus Radix**. Be careful to click in the same row as one of your selections, or you will change the selection.
The Set Bus Radix dialog box opens.
3. In the drop-down menu, choose the radix or token set.
4. In the Range drop-down menu, choose **Selected signals** or, if you want to change all the signals in the waveform to the same radix, choose **All signals**.
5. Click **OK**.

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.

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

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 LA Trigger 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 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 clearing 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:

- ▶ Click in the appropriate box in the Name column of the Trigger Unit section of the LA Trigger 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 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 Analyzer.
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 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 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:

- ▶ Click in the appropriate box in the Name column of the Trigger Expression section of the LA Trigger 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 Analyzer. You can also enable or disable trigger expressions in Reveal Analyzer. However, you cannot change the sequence depth, the maximum sequence depth, or the maximum event counter of the trigger expressions in Reveal 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 [“Triggering” on page 32](#) for information on these operators.

Reveal 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 [“Triggering” on page 32](#) 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 Analyzer. 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 Analyzer.
5. In the POR Debug section, if you wish to connect POR when Analyzer opens, click the **Connect POR when Analyzer opens** check box.
6. 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

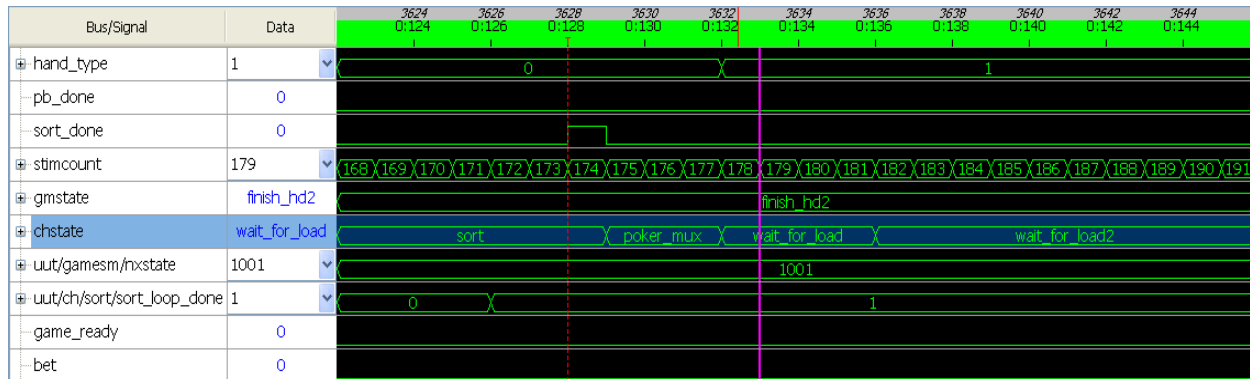
You can set a number of options to control the triggering.

1. **To set trigger options in Reveal Analyzer:**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 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.
2. 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.
3. In the **Trigger Position** field, select **Pre-selected Position** or **User-selected Position**.

Creating Token Sets

You can create sets of “tokens,” or text labels, for values that might appear on trace buses. You can create tokens such as ONE, TWO, THREE, or Reset, Boot, Load. Tokens can make reading the waveforms in Reveal Analyzer easier and can highlight the occurrence of key values. See the following figure for an example. The row for the chstate bus uses tokens.

Figure 14: LA Waveform View Using Tokens



To create or modify a token set:

1. Choose **Design > Token Set Manager**.

The Token Manager dialog box opens. If the Reveal project already has token sets defined, they are listed in the dialog box.

2. If you want to use token sets that were previously saved to a separate file, right-click in the dialog box and choose **Import**. In the Import Tokens dialog box, browse to the token (.rvt) file and click **Open**.

The token sets in the .rvt file are added to the list in Token Manager.

3. To create a new token set, click **Add Set**.

A new token set is started with default values. But it has no tokens defined yet.

4. To change the size of the token values, double-click the value in the Num. of Bits column and type in the new width, in bits. The width can be up to 256. The width must be the same as the bus that the token set will be used with.

The Num. of Bits value can only be changed when the token set is empty. If there are any tokens, you will get an error message.

5. To create a new token, select a token set. Then click **Add Token**.

A new token is created with default values. Repeat for as many new tokens needed.

6. You can modify token sets by doing any of the following:

- ▶ To change the name of a token or token set, double-click the name and type a new name. The name can consist of letters, numbers, and underscores (_). It must start with a letter.

- ▶ To change the value of a token, double-click the value and type in a new value. Token values must be prefixed by one of the radix indicators shown in the following table:

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

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

- ▶ To remove a token or token set, select it. Then click **Remove**.
7. You can save the collection of token sets showing in the dialog box to a separate file for use in another project. To save the token sets, right-click and choose **Export**. In the Export Tokens dialog box, browse to the desired location and type in the name of the new token (.rvt) file. Click **Save**.
 8. When you are done, click **Close** to close the dialog box. The token sets are automatically applied to the current Reveal project.

Debugging with Reveal Controller

Reveal Controller is another divide-and-conquer mechanism for you to emulate an otherwise unavailable environment for power debug. For example, your evaluation board would only have a limited number of LEDs or switches but the virtual environment enables up to 32 bits. Register memory mapping and dumping of values is also easily manifested while visibility into Hard IPs is also enabled.

Note

Reveal Controller is not supported in iCE40 UltraPlus.

Virtual LED Switch Console

Once Reveal Analyzer is set up and started, you are presented with three tabs, the first of which is a virtual LED/Switch console to emulate the board.

The top section, Virtual LED shows all the LEDs that were defined in the insertion stage. Once the board is running, you can see the LEDs flash in red and green as they would in the real hardware environment.

The lower section shows the virtual switch in which you can either enter the data as a hex value or can set the virtual dip switches.

Select Direct Mode to see the data and switches in real time instead of waiting until the Apply button is clicked.

Note

Switches are of type in/out. When defining switches in RTL, use the WIRE definition otherwise multiple input driver errors will be encountered later in the flow.

Assigning Colors to Virtual Switch/LED Signals

To assign colors to virtual switch/LED signals:

1. In the Switch List and LED List tables, click the color of the signal.
2. This opens the Select Color dialog box. Choose the color to assign to the signal and click **OK**.

The colors you assign are also reflected in the virtual LED/Switch console in Reveal Analyzer.

User Memory Analysis

The second tab is for User Memory Analysis. The title indicates the range of the User Memory map. Analysis follows no particular steps or order.

- ▶ Default Data: A value that will initialize all memory locations.
- ▶ Write Address/Data: Address and data in hex to be written.
- ▶ Read Address/Data: Enter address. The data is read back when 'Read' is selected.
- ▶ Memory File: You can dump from/to a range of memory addresses to a .mem file. A user can also **Load MemFile** to load a pre-configured memory file.

Running User Control Register

1. In the **User Control Register** tab, select a label from the **Name** column, then click the **Rd** button.

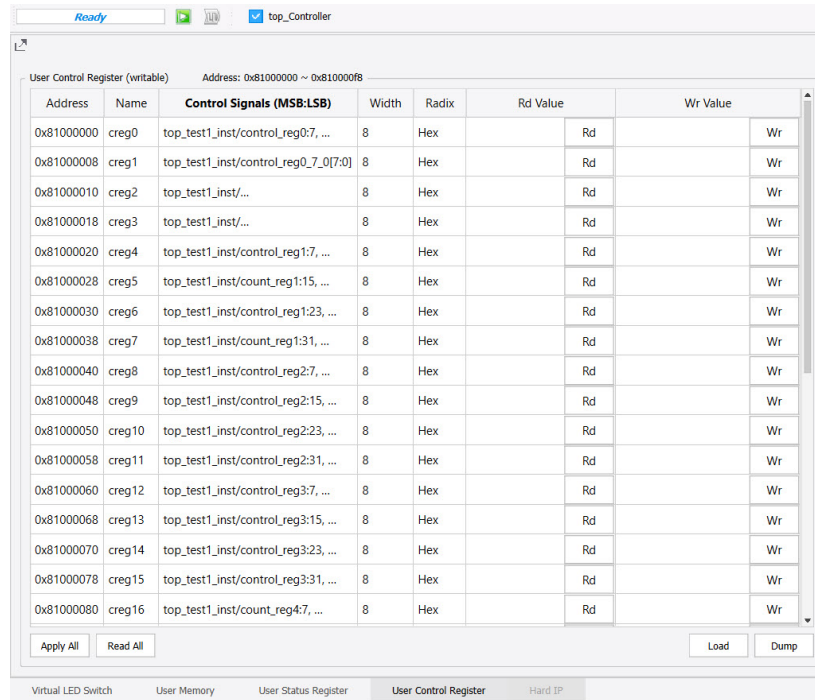
User Control Register reads the value of the corresponding address location and executes the following TCL command:

```
rva_run_controller -read_control "label"
```

2. Select a label from the Name column.
3. In the **WR Value** column, do the following:
 - a. Enter desired value.
 - b. Click the **Wr** button.

User Control Register executes the following TCL command:

```
rva_run_controller -write_control "label" -data value
```



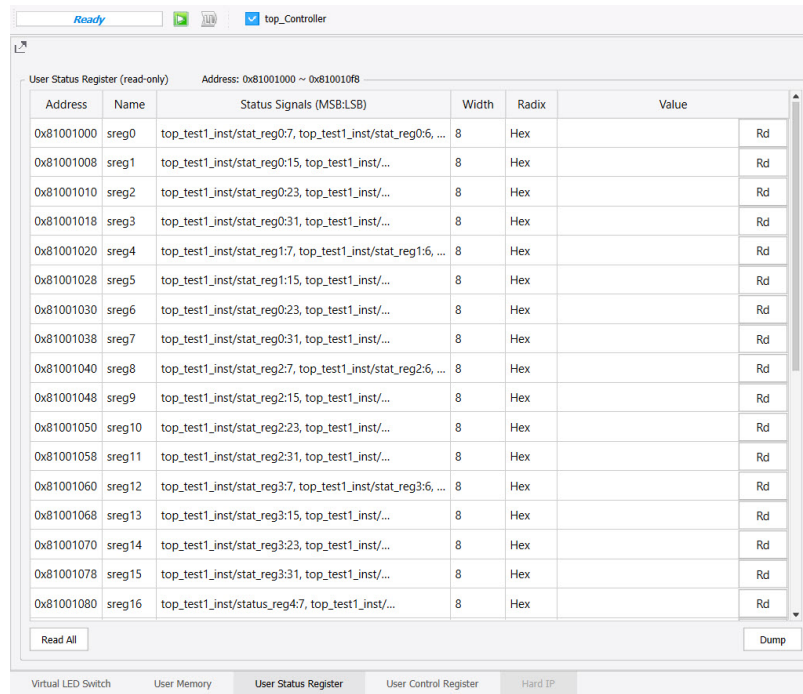
4. Double-click a row in the **Control Signals** column to display the list of signals with their corresponding value.
5. In the User Control Register tab, click the **Apply All** button to apply the value of all address locations assigned by the Name column and execute multiple TCL commands sequentially.
6. Click the **Read All** button to read the value of all address locations assigned by the Name column and execute multiple TCL commands sequentially.
7. Click the **Load** button to load the value from the register file to all address locations of Control Register.
8. Click the **Dump** button to save the value of all address locations to a register file.

Running User Status Register

1. In the **User Status Register** tab, select a label from the **Name** column, then click the **Rd** button.

User Status Register reads the value of the corresponding address location and executes the following TCL command:

```
rva_run_controller -read_status "label"
```



- On the bottom left corner, click the **Read All** button.

The Read All button reads the value of all address locations assigned by the Name column and executes multiple TCL commands sequentially.

- On the bottom right corner, click the **Dump** button.

The Dump button saves the value of all address locations to a register file.

- Double-click a row in the **Status Signals** column to display the list of signals with their corresponding value.

Hard IP Debug

All the IP selected for analysis in Reveal Inserter are displayed here.

Main operations are reading/writing to memory locations.

Note

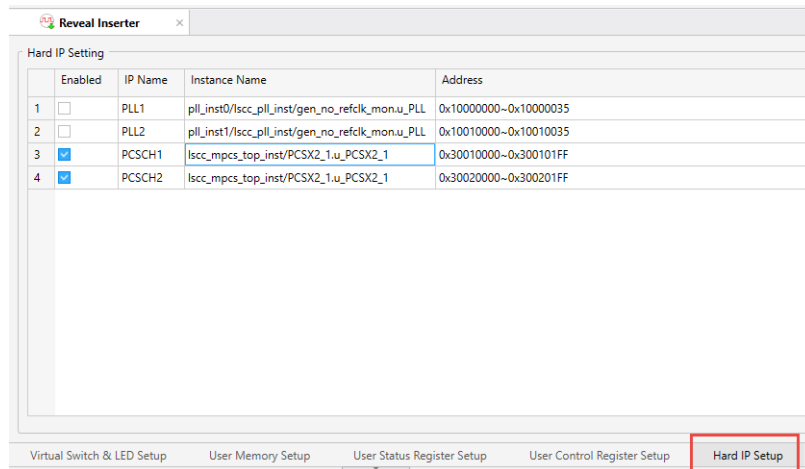
The detailed settings of each IP is beyond the scope of this help. Links will be provided to Application Notes to explain its usage.

Debugging Nexus PCS/SerDes

Radiant supports PCS/SerDes debugging for devices built on the Nexus platform such as CertusPro-NX.

To debug Nexus PCS/SerDes:

1. Use the Reveal Inserter tool to add controller cores to the design. These cores allow writing and reading the control and status registers and change the parameters for PCS/Serdes operations.
2. If your design has a hard IP, the Hard IP Setup tab displays all your hard IPs. Under the Enable column, click the check box of an IP to enable the debug function.



3. Generate the updated bitstream with the inserted debug cores and program the device.
4. Open Reveal Analyzer/Controller and connect to the programmed device through USB/JTAG.

5. In the Hard IP tab, you will see the instantiated hard IP blocks on the Nexus device. Scroll to the IP core to debug..

PCSX2_1

Address (0x30010000 ~ 0x300101FF)

Instance Hierarchy: lsc_mpcs_top_inst/PCSX2_1.u_PCSX2_1

Eye-Opening Monitor...

Transmit Settings

Differential Amplitude: 0mV

Output Termination: 100 ohm

Tx Post-Cursor Ratio: 0

Tx Pre-Cursor Ratio: 0

Invert Tx Data Polarity: off

Tx PLL Loss of Lock Status: unlocked

Receive Settings

Input Termination: 100 ohm

Rx EQ: SS_LMS 2.5GT/s

Invert Rx Data Polarity: off

CDR Loss of Lock Status: unlocked

Data Loss of Lock Status: unlocked

Loopback Mode Settings Note: The "Reload" function will not update th

Loopback Mode: Disabled

Serdes/PCS Reset

MPCS Tx Reset: off

MPCS Rx Reset: off

PMA Tx Reset: off

PMA Rx Reset: off

PMA Tx HiZ: off

PMA Rx HiZ: off

Apply Reload

Write Address: 0x30010000

Write Data: 0x0 Write

Read Address: 0x30010000

Read Data: Read

Dump From (Offset): 0x00000000 Range: Dump

Dump MemFile: ... Dump

Load MemFile: ... Load

Virtual LED Switch User Memory User Status Register User Control Register **Hard IP**

6. Set the control parameters and click the **Apply** button.
7. Run your design.
8. Analyze the captured data. You can modify settings if allowed.

Click the **Eye Opening Monitor** button to view a graphical eye diagram that details the signal eye width and height. See [“Running Reveal Eye-Opening Monitor” on page 102](#).

Debugging Avant with SerDes Toolkit

In designs using Avant MPPHY module, Radiant supports SerDes debugging using the SerDes toolkit. This feature allows you to evaluate and debug the performance of transceivers. It generates and checks data patterns to measure the bit error rate (BER), which helps in identifying signal integrity issues. In Reveal Inserter, set up hard IP by clicking the Hard IP Setup tab and then select functions by enabling check boxes.

You can test and validate transceiver links. You can verify that the transceivers are functioning correctly and can handle the data rates required in your design. By analyzing the BER, you can identify and diagnose issues such as signal degradation, noise, or incorrect settings. You can also optimize transceiver configurations by adjusting various transceiver parameters.

SerDes toolkit features:

- ▶ Real-time access to transceiver settings. This allows you to monitor and modify the settings of the transceiver on-the-fly while the system is running.
 - ▶ Adjust Parameters: Use different transceiver and clocking topologies. Change settings such as pre-emphasis, equalization, and data rate in real-time to see how they affect the signal quality.
 - ▶ Optimize Performance: Fine-tune the transceiver settings to achieve the best possible performance for your specific board configuration.
 - ▶ Troubleshoot Issues: Quickly identify and resolve issues by making adjustments and immediately observing the impact on the link quality.
- ▶ Sweep testing. Automate the process of testing various settings to find the optimal configuration. The AutoSweep feature with interval-stepping capability allows quick testing of many combinations of physical medium attachment (PMA) analog settings to find the optimal setting for a particular transceiver link.

Note

You can capture Eye Diagram data while auto-sweeping the PMA settings.

- ▶ Data pattern generators and checkers select transceiver.
- ▶ Multiple different PRBS (Pseudo-Random Binary Sequence) and clock patterns transmitted over the channels.
- ▶ Channel manager interface monitor and track status while testing multiple channels simultaneously.
- ▶ Running link tests between multiple devices across one or more boards.
- ▶ Save and Load capability that enables comparison of different sets of signal integrity results including the eye diagram. This capability enables customers to compare the signal integrity results for different PMA settings.

After the device is programmed, SerDes Toolbox, TX, and RX settings are displayed in the Hard IP tab of the Reveal Controller interface.

Start Page | Reports | Reveal Analyzer/Controller | top_Controller

Ready

MPPCIEX8AQ1

MPPX4 Address (0x30000000 ~ 0x3000FFFF)

Instance Hierarchy: inst_m

Assigned Quad: Quad-A

SerDes Toolbox

	Lane0	Lane1	Lane2	Lane3
SerDes Toolbox	Read	Read	Read	Read
Loopback Mode	Disabled	Disabled	Disabled	Disabled
TX PRBS Pattern	Disabled	Disabled	Disabled	Disabled
RX PRBS Pattern	Disabled	Disabled	Disabled	Disabled
PAT0	0	0	0	0
Restart	Restart	Restart	Restart	Restart
Inject 1-bit Error	Inject	Inject	Inject	Inject
PRBS Error Count	0	0	0	0

TX Settings

	Lane0	Lane1	Lane2	Lane3
TX	Read	Read	Read	Read
PLL Lock Status	Unlocked	Unlocked	Unlocked	Unlocked
TX Data Polarity	Non-invert	Non-invert	Non-invert	Non-invert
TX Data Width	8-bit	8-bit	8-bit	8-bit
TX Data Enable	Off	Off	Off	Off
TX Equal Pre-cursor	0	0	0	0
TX Equal Main-cursor	0	0	0	0
TX Equal Post-cursor	0	0	0	0
TX Termination	54-ohm	54-ohm	54-ohm	54-ohm
TX Diff Amplitude Enable	Off	Off	Off	Off
TX Diff Amplitude (iboot)	0	0	0	0
TX Diff Amplitude (vboost)	0	0	0	0

RX Settings

	Lane0	Lane1	Lane2	Lane3
RX	Read	Read	Read	Read
CDR Lock Status	Unlocked	Unlocked	Unlocked	Unlocked

Virtual LED Switch | User Memory | User Status Register | User Control Register | Hard IP

Clicking the **Read** button of a specific lane shows the required settings from the device and populates the values for the lane. Inactive lanes are grayed out.

Table 2: SerDes Toolbox Settings

Settings	Display/Values	Description
Loopback Mode	DISABLED (Default)	This is a testing configuration where the transmitted signal is looped back to the receiver without leaving the device.
	Near-End Serial	In Near-End Serial, the transmitted signal is looped back at the near end of the communication link, close to the transmitter.
	Parallel	In Parallel, the transmitted data is looped back in parallel.
		Multiple data lanes are used simultaneously for the loopback test.
TX PRBS Pattern	DISABLED (Default)	This is the PRBS (Pseudo-Random Binary Sequence) pattern generated and sent by the transmitter side of the communication link.
	PRBS31	
	PRBS23_18	
	PRBS23_21	
	PRBS16	The PRBS pattern is used to test the integrity of the link by sending a known sequence of bits and checking if the received sequence matches the expected pattern.
	PRBS15	▶ PRBS<BITS> PRBS patterns with more number of bits (long pattern) are particularly useful for jitter measurement.
	PRBS11	▶ FIXED WORD is a single unchanging predefined sequence.
	PRBS9	▶ DC balanced refers to a pattern that includes an equal number of ones and zeros. This balance helps keep the DC level steady and reduces signal degradation, which is particularly important when the signal passes through AC-coupling circuits.
	PRBS7	
	FIXED WORD (PAT0)	
	DC BALANCED (PAT0, ~PAT0)	
	FIXED PATTERN (000, PAT0, 3FF, ~PAT0)	▶ FIXED PATTERN is a predefined sequence that is a combination of multiple fixed words or sequences.
		When selected, the Restart button takes on the value.
RX PRBS Pattern	DISABLED (Default)	This is the PRBS pattern generated and sent by the receiver side of the communication link.
	PRBS31	
	PRBS23_18	
	PRBS23_21	
	PRBS16	The PRBS pattern is used to test the integrity of the link by sending a known sequence of bits and checking if the received sequence matches the expected pattern.
	PRBS15	▶ PRBS<BITS> PRBS patterns with more number of bits (long pattern) are particularly useful for jitter measurement.
	PRBS11	▶ FIXED WORD is a single unchanging predefined sequence.
	PRBS9	▶ DC balanced refers to a pattern that includes an equal number of ones and zeros. This balance helps keep the DC level steady and reduces signal degradation, which is particularly important when the signal passes through AC-coupling circuits.
	PRBS7	
	FIXED WORD (PAT0)	
	DC BALANCED (PAT0, ~PAT0)	
	FIXED PATTERN (000, PAT0, 3FF, ~PAT0)	▶ FIXED PATTERN is a predefined sequence that is a combination of multiple fixed words or sequences.
		When selected, the Restart button takes on the value.
PAT0		Enter the pattern:
		▶ 8 bits for 16-bit data width
		▶ 10 bits for 20-bit data-width

Table 2: SerDes Toolbox Settings

Settings	Display/Values	Description
Restart	Restart button	Clicking this button clears the TX and RX pattern settings and sets the selected Patterns and sync.
Inject 1-bit Error	Inject 1-bit Error button	Clicking this button inserts a 1-bit error in LSB of TX word.
PRBS Error Count	PRBS Error Count button	Clicking this button reads the error count and displays it in the cell.

Table 3: TX Settings

Settings	Display/Values	Description
PLL Lock Status	Green = Locked Red = Unlocked	When the PLL (Phase-Locked Loop) is locked, the output signal stays in sync with the input signal, maintaining a steady phase relationship. If the PLL becomes "unlocked," it indicates that synchronization failed, and the output signal is no longer aligned with the input signal.
TX Data Polarity	Non-inverted = Unchecked Inverted = Checked	Non-Inverted Data Polarity means that the data signal is transmitted without any change in its original form. The logical '1' (high) and logical '0' (low) levels of the signal are maintained. Inverted Data Polarity means that the data signal is flipped or transmitted in an inverted form. In this case, the logical '1' becomes '0' and the logical '0' becomes '1'.
TX Data Width	8-bit 10-bit 16-bit 20-bit 32-bit 40-bit	TX data width is the number of bits that can be transmitted simultaneously over a channel.
TX Data Enable	Off On	When the TX Data Enable signal is On, the data on the TX line is valid and ready for transmission. When the signal is Off, the data on the TX line is not valid, and the transmission should be ignored.
TX Equal Pre-cursor	0.0 to 6.75 with 0.25 each step	The adjustment of the signal before the main data pulse to compensate for distortions and signal degradation caused by the transmission channel. Pre-conditioning the signal with pre-cursor equalization reduces inter-symbol interference (ISI) and enhances overall signal quality at the receiver's end.
TX Equal Main-cursor	0 to 24	The primary coefficient (main cursor) is used as the key value in determining the total equalization. The main cursor coefficient represents the central, most significant part of the transmitted signal. It is the primary reference point in the signal that should have the highest amplitude.

Table 3: TX Settings

Settings	Display/Values	Description
TX Equal Post-cursor	0.0 to 8.75 with 0.25 each step	The adjustment of the signal after the main data pulse to compensate for distortions and signal degradation caused by the transmission channel. Pre-conditioning the signal with post-cursor equalization reduces inter-symbol interference (ISI) and enhances overall signal quality at the receiver's end.
TX Termination	54-ohms 52-ohms 50-ohms 48-ohms 46-ohms 44-ohms 42-ohms 40-ohms	The termination resistor value on the transmission side of the communication link. This resistor matches the transmission line impedance and minimizes signal reflections.
TX Diff Amplitude Enable	Off On	When the TX Diff Amplitude Enable setting is turned On, the transmitter can generate a differential signal with the appropriate amplitude so that the signal can be accurately interpreted by the receiver, even in the presence of noise.
TX Diff Amplitude (iboot)	0 to 15	The strength of the amplitude boost applied to the differential signal. This help helps the transmitted signal to be accurately received despite noise or interference.
TX Diff Amplitude (vboost)	0 to 7	

Table 4: RX Settings

Settings	Display/Values	Description
CDR Lock Satus	Green = Locked Red = Unlocked	When the CDR (Clock and Data Recovery) the data is correctly sampled and the clock is properly aligned with the data transitions. If the CDR is unlocked, data errors may occur due to misalignment between the clock and data signals.
RX Signal Status	Green = Valid Red = Invalid	A valid RX signal status indicates that the received data is correctly aligned with the clock signal. An invalid RX signal means that the received data is not correct or cannot be properly interpreted. This may be due to signal integrity issues, clock misalignment, protocol errors, or data corruption.

Table 4: RX Settings

Settings	Display/Values	Description
RX Loss of Signal Threshold	60mVpp	This value is the minimum signal level required to maintain reliable data transmission. When the signal falls below this threshold, the system detects a loss of signal.
	80mVpp	
	100mVpp	
	120mVpp	
	140mVpp	
	160mVpp	
	180mVpp	
RX Data Polarity	Non-inverted = Unchecked	Non-Inverted Data Polarity means that the data signal is received without any change in its original form. The logical '1' (high) and logical '0' (low) levels of the signal are maintained. Inverted Data Polarity means that the data signal is flipped or received in an inverted form. In this case, the logical '1' becomes '0' and the logical '0' becomes '1'.
	Inverted = Checked	
RX Data Width	8-bit	RX data width is the number of bits that can be received simultaneously over a channel.
	10-bit	
	16-bit	
	20-bit	
	32-bit	
	40-bit	
RX Equal Afe Rate	0 to 7	The "adaptive equalization filter rate" setting that adjusts the equalization parameters dynamically to optimize signal integrity and minimize bit errors.
RX Equal Atten Level	0 to 7	The receiver equalization attenuation level setting adjusts the attenuation applied to the received signal to optimize signal integrity and minimize bit errors.
RX Equal Ctle Pole	0 to 3	This value indicates the pole location of the Continuous-Time Linear Equalizer (CTLE) in the receiver path.
RX Equal Ctle Boost	0 to 31	This value refers to the CTLE boost in the receiver path. Boosting the CTLE increases its gain, especially at higher frequencies.
RX Termination	54-ohms	The termination resistor value on the receiver side of the communication link. This resistor matches the transmission line impedance and minimizes signal reflections
	52-ohms	
	50-ohms	
	48-ohms	
	46-ohms	
	44-ohms	
	42-ohms	
	40-ohms	
RX Termination Acdc	DC (Default) AC	This indicates whether the receiver termination is set to an alternating current (AC) or a direct current (DC) coupling mode.

Table 4: RX Settings

Settings	Display/Values	Description
RX Equal Vga1 Gain	0 to 7	The value indicates the amplification level of the signal received by VGA2.
RX Equal Vga2 Gain	0 to 7	The value indicates the amplification level of the signal received by VGA2.

Clicking the Eye Opening Monitor button provides a graphical eye diagram that details the signal eye width and height.

See [“Running Reveal Eye-Opening Monitor” on page 102](#) for more information.

Running Reveal Eye-Opening Monitor

The Reveal Eye-Opening Monitor can be launched from the Reveal Analyzer/Controller. You need to be connected to a board to be able to run the Eye-Opening Monitor.

To launch the Eye-Opening Monitor:

1. In the Radiant software main window choose **Tools > Reveal Analyzer/Controller**

In the drop-down menu, choose **top_Controller** to display the **Hard IP** tab.

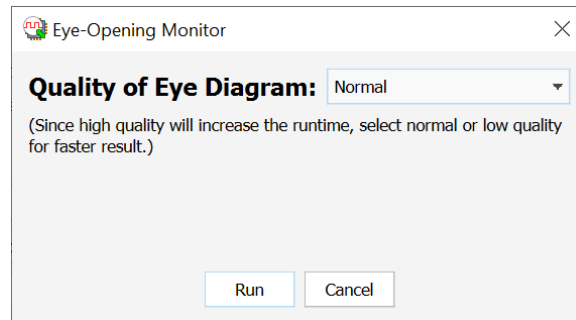
2. Click the **Hard IP** tab.

If debugging a non-Avant device, scroll down to the PCS channels to locate the **Eye-Opening Monitor** button.

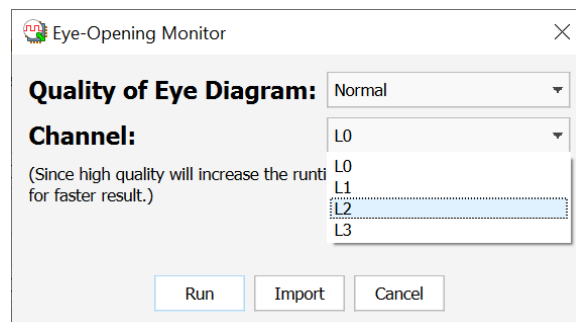
If debugging an Avant device, the **Eye-Opening Monitor** button is located at the upper-right of the current IP instance.

3. Click the **Eye-Opening Monitor** button. The Eye-Opening Monitor dialog box is displayed.

The figure below shows the Eye-Opening Monitor dialog box when debugging a non-Avant device.



The figure below shows the Eye-Opening Monitor dialog box when debugging an Avant device. The Channel and Import options are added.



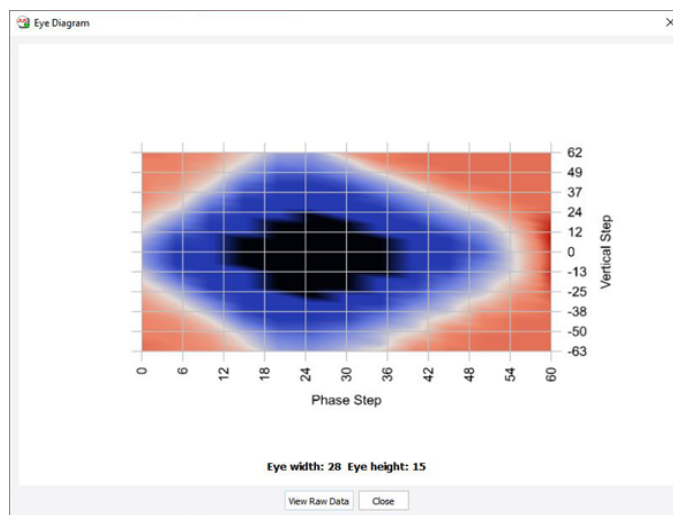
4. In the Eye-Opening Monitor dialog box, select the quality of the eye diagram from the drop-down menu.

It is recommended to select normal or low quality for a faster result, and select high quality for the final result. The default is normal quality.

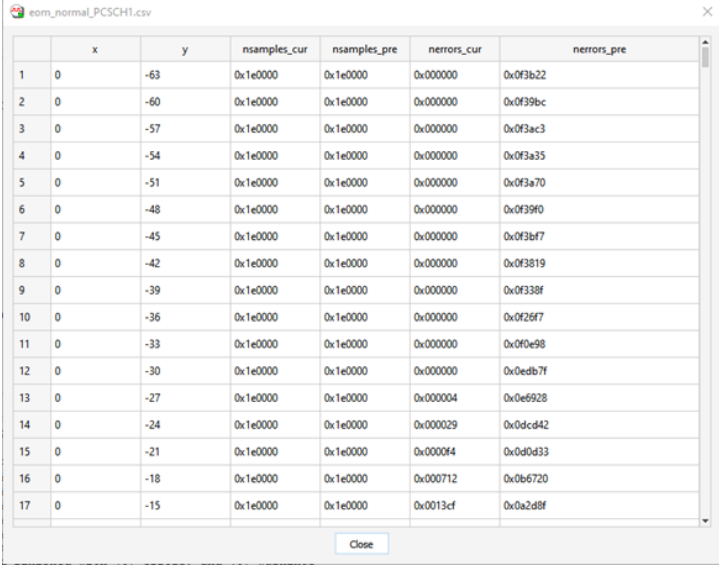
If you are debugging Avant SerDes multiple channels, select the channel to debug in the Channel drop-down menu.

5. Click the **Run** button to start the EOM process. You can stop the process at any time by clicking the **Stop** button.
6. After the EOM has finished running, it will display the eye diagram for the current channel. The figure below shows the Eye Diagram when debugging non- Avant devices.

Figure 15: Example of Eye Diagram for Non-Avant Devices



At the bottom of the eye diagram for non-Avant devices, you can click the **View Raw Data** button to open the raw data file containing the number of samples and errors used in calculating the density of the eye diagram.



	x	y	nsamples_cur	nsamples_pre	nerrors_cur	nerrors_pre
1	0	-63	0x1e0000	0x1e0000	0x000000	0x0f3b22
2	0	-60	0x1e0000	0x1e0000	0x000000	0x0f39bc
3	0	-57	0x1e0000	0x1e0000	0x000000	0x0f3ac3
4	0	-54	0x1e0000	0x1e0000	0x000000	0x0f3a35
5	0	-51	0x1e0000	0x1e0000	0x000000	0x0f3a70
6	0	-48	0x1e0000	0x1e0000	0x000000	0x0f39f0
7	0	-45	0x1e0000	0x1e0000	0x000000	0x0f3bf7
8	0	-42	0x1e0000	0x1e0000	0x000000	0x0f3819
9	0	-39	0x1e0000	0x1e0000	0x000000	0x0f38f8
10	0	-36	0x1e0000	0x1e0000	0x000000	0x0f26f7
11	0	-33	0x1e0000	0x1e0000	0x000000	0x0f0e98
12	0	-30	0x1e0000	0x1e0000	0x000000	0x0edb7f
13	0	-27	0x1e0000	0x1e0000	0x000004	0x0e9928
14	0	-24	0x1e0000	0x1e0000	0x000029	0x0dcd42
15	0	-21	0x1e0000	0x1e0000	0x0000f4	0x0d0d33
16	0	-18	0x1e0000	0x1e0000	0x000712	0x0b6720
17	0	-15	0x1e0000	0x1e0000	0x0013cf	0x0a2d8f

- ▶ The raw data comes from the eom register.
- ▶ The numbers from nsample_cur, nsample_pre, nerrors_cur, and nerrors_pre columns calculate the result.
- ▶ The numbers are collected by reading the eom register of the PCS IP.

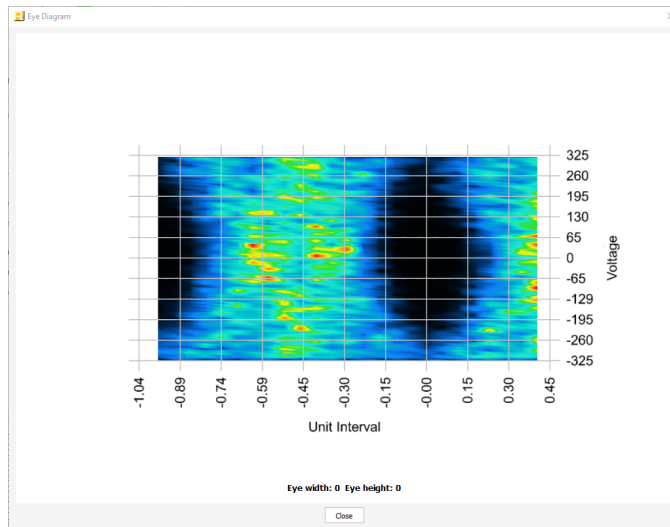
If you change the Reveal Controller options, you can re-run the EOM to display the new eye diagram and view the new raw data until the desired result is achieved.

Importing an Existing EOM Result

If you are debugging Avant SerDes, each time you run the EOM, the result is automatically saved as a .txt file in the project folder. The file name provides information on the EOM settings. For example, the file name *eom_normal_MPPHYX4Q1_L2.txt* tells you that the EOM result was generated using normal quality for the MPPHYX4Q1 instance with L2 or lane 2 as the selected channel.

To import an existing EOM result:

1. Click the **Import** button in the Eye-Opening Monitor dialog box.
2. The Select Existing EOM File dialog box is displayed. Selecting AVANT EOM (*.txt) displays only the EOM results for debugging the Avant device.
3. Select the .txt file and click **Open**. The Eye Diagram is displayed. The figure below shows the Eye Diagram when debugging Avant devices.

Figure 16: Example of Eye Diagram for Avant Devices

Performing Logic Analysis

After you have configured trigger settings in the LA Trigger tab, you can perform a logic analysis on a single core.

To perform logic analysis:

1. In the `<design_name>_LA<core_number>` check box in the toolbar, select the core on which to perform logic analysis.
2. Click  on the Reveal Analyzer toolbar.

The Run button changes into the Stop  button and the status bar next to the button shows the progress.

Reveal Analyzer first configures the 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 LA Waveform tab.

If the trigger condition is not met, Reveal Analyzer will continue running. In that case, you can use manual triggering, described in [“Using Manual Triggering”](#) on page 107.

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 discontinuous, 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 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 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

Please refer to the Reveal Troubleshooting guide for more information.

Stopping a Logic Analysis

You can stop a logic analysis while it is running.

To stop a logic analysis:

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

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

Viewing Waveforms

After running your logic analysis, you can view the trace buffer data in waveform format in the LA Waveform tab. Whenever the trace stops, Reveal Analyzer reads the trace samples back from the trace memory and automatically updates the signal waveforms.

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

Viewing Logic Analysis

To view a logic analysis:

1. Click the **LA Waveform** tab.
2. Choose a module from the drop-down menu in the Reveal Analyzer toolbar.

Adjusting the Waveform Display

You can adjust the waveform display by panning and zooming. You can also adjust the colors.

Panning

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

To pan the waveform display:

1. Right-click in the waveform and choose **Pan Mode**.
2. 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 Analyzer LA Waveform tab to increase or decrease the displayed time interval.

To zoom in on a waveform:

- ▶ Choose **View > Zoom In**, click , or right-click on the waveform and choose **Zoom > Zoom In**.

To zoom in on a specified area:

1. Right-click the waveform and choose **Zoom Mode**.
The pointer changes to a cross.
2. Hold down the left mouse button and drag the pointer across the area you want to zoom in on.
A shaded area appears on the waveform display.
3. Release the mouse button.
The shaded area expands to fill the display.

To zoom out on a waveform:

- ▶ Choose **View > Zoom Out**, click , or right-click on the waveform and choose **Zoom > Zoom Out**.

To show the entire waveform in the window:

- ▶ Choose **View >  Zoom Fit**.

To zoom to the trigger point:

- ▶ Right-click in the waveform and choose **Zoom > Zoom Trigger**

To zoom to the start of the display:

- ▶ Right-click in the waveform and choose **Zoom > Zoom Start**

To zoom to the end of the display:

- ▶ Right-click in the waveform and choose **Zoom > Zoom End**

Setting a Trace Bus Radix

You can set the radix of a trace bus displayed in the LA Waveform tab. You can choose a binary, octal, decimal, or hexadecimal radix. You can also use any token set whose bit width matches the bus.

To set the bus radix of a signal or bus:

1. In the LA Waveform tab, click in the Data cell of the signal or bus.
A menu appears showing the different radices and any token sets that fit.
2. Choose the desired radix or token set.

To set the bus radix of multiple signals and buses:

This method can set several signals to the same radix but cannot use tokens.

1. In the LA Waveform tab, select one or more buses.
To select one bus, click on it. To select more than one, Control-click on each one. To select all buses in a range, click on one end of the range and Shift-click the other end. If you want to change all the signals in the waveform to the same radix, you do not need to select anything.
2. Right-click in one of the selected waveforms and choose **Set Bus Radix**.
Be careful to click in the same row as one of your selections, or you will change the selection.
The Set Bus Radix dialog box opens.
3. In the drop-down menu, choose the radix.
4. In the Range drop-down menu, choose **Selected signals** or, if you want to change all the signals in the waveform to the same radix, choose **All signals**.
5. Click **OK**.

Changing LA Waveform Colors

You can change the colors used by the waveform.

To change the colors:

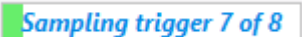
1. Open the Options dialog box. Depending on which version of Reveal Analyzer you're using, do one of the following:
 - ▶ If integrated with the Radiant software, choose **Tools > Options**. Then, in the Options dialog box, choose **Color** in the left panel and then click on the **Reveal Analyzer** tab.
 - ▶ If stand-alone, choose **Design > Options**.
2. Click on the color sample for the desired part of the LA Waveform view. The Select Color dialog box opens.
3. Select a color.
4. In the Select Color dialog box, click **OK**.
5. To see the effect of the change, click **Apply**.
6. Change other colors if desired.
7. Click **OK**.

Incremental Display for Multiple Trigger Capture

In previous Radiant releases, waveforms for multiple triggers are displayed only after all the triggers are captured. With the incremental display feature, you can start analyzing partial results right after the first trigger is completed. Indicators allow you to monitor capture progress and decide when to load additional or specific trigger frames.

The incremental display behavior is described as follows:

1. First Trigger – When the first trigger condition is met, trace data is captured. The captured waveform is immediately displayed in the LA Waveform tab.
2. Subsequent Triggers – Additional trigger events are captured in the background. The waveform display does not automatically switch to later trigger frames or segments to preserve your current analysis view.

In the Reveal Analyzer toolbar, the capture status field indicates the current trigger being sampled out of the total number of triggers. For example,  .

- ▶ In the Reveal Analyzer toolbar, clicking the **View Latest Triggers** button  displays all the available trigger frames. This button becomes active when new trigger data is available.
3. Manual Selection – You can manually select and view additional trigger frames through the following options:

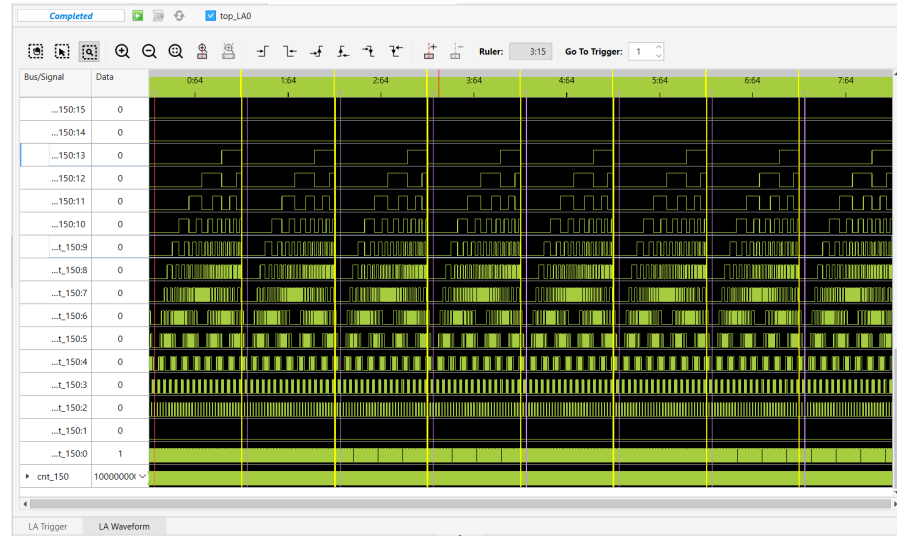
- ▶ In the Waveform toolbar, indicating a trigger number in **Go to Trigger** loads the specific trigger to view.

The waveform display remains on the currently selected trigger frame until a new frame is selected.

4. Capture Completion

After the final trigger event, data capture stops. All trigger frames are available for analysis.

Figure 17: Completed Multiple Trigger Capture



Specifying the Clock Period

You can specify a clock period for your logic analysis. Setting the frequency enables you to determine the location of your cursors, as well as the distance between them. Frequency is determined by dividing 1 by the period.

To set the clock frequency:

1. Right-click the waveform and choose **Set Clock Period**.
2. In the Specify Clock Period 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. Place the cursor in either the Period or Frequency text box and type in the desired value. The other text box fills in automatically.

Only integers are allowed. If you try to specify a frequency that would require a non-integer period, the period is truncated to an integer and the frequency is automatically adjusted. For example, typing 150 in the Frequency text box gives you a period of 6 and a frequency of 166.

4. Click **OK**.

Placing, Moving, and Locating Cursors

The LA Waveform view comes with three types of “cursors” to highlight moments in the waveform. The cursors are vertical lines cutting through all the signals at the leading edge of a clock cycle. See Figure 19 on page 120. The three types are:

- ▶ **Trigger.** A purple line with a “T” at the top, trigger cursors are automatically placed at the moment of each final trigger event. If the module used a sample enable signal and the exact moment of the trigger is unknown, the waveform shows a trigger cursor five clock cycles before the sample enable signal turned inactive and sampling stopped.
- ▶ **Active.** A red line appears wherever you click in the waveform. The Data column shows the values of the signals and buses at the moment highlighted by the active cursor.
- ▶ **User.** A blue line can be placed anywhere you want. Use these cursors to mark moments of interest. You can also use these cursors to maneuver about a long waveform with the Go to Cursor command.

Most cursor functions require that the LA Waveform view be in Select mode: right-click in the LA Waveform view and choose **Select Mode**.

To create a user cursor:

1. Click in the desired clock cycle.
The active cursor appears. Make sure it is where you want the user cursor to be.
2. Right-click and choose **Add Cursor**.

To move a user cursor:

1. Zoom in so you can easily see and click in individual samples.
2. Click in the desired location.
The active cursor appears. Make sure it is where you want the user cursor to be.
3. Carefully click in the sample to the right of the user cursor.
You must click on or to the right of the user cursor. Otherwise you are just moving the active cursor to a neighboring sample.
The user cursor and the active cursor exchange locations.

To jump to a user cursor:

- ▶ Right-click in the waveform and choose **Go to Cursor > <cursor>**.
Cursors are identified by the sample index as shown in the green bar at the top of the waveform display.

To remove a user cursor:

1. Click on or near the cursor.

The active cursor appears. Make sure it is on or next to the user cursor you want to remove.

2. Right-click and choose **Remove Cursor**.

To remove all user cursors:

- ▶ Right-click in the waveform and choose **Clear All Cursor**.

Counting Samples

You can easily count the number of samples in a range on the display.

To count samples:

- ▶ Click where you want to start counting and drag to the end of the range.

While you're dragging, the LA Waveform view shows two red lines and the number of samples between the lines.

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 data:

1. If you want the data to include an approximate measure of time instead of a simple count of clock cycles, right-click the waveform and choose **Set Clock Period**. See [“Specifying the Clock Period” on page 111](#).
2. If you want to export only some of the signals, select them in the waveform. You can only export whole buses. If you select only some of the signals in a bus, you get the whole bus.
3. Right-click in the waveform and choose **Export Waveform**.
The Export Waveform dialog box opens.
4. Browse to the location where you want to export the file.
5. Type in a name in the **File name** box.
6. Choose a file type.
7. If you are exporting only some of the signals, choose **Selected signals** in the Range box.

8. If you are exporting to .vcd, type in a module name. This will form the title in the .vcd file. If you leave the field empty, the module name will be "<unknown>".
9. Click **Save**.

Saving a Project

You can save the trigger settings and waveform setup settings in a Reveal 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 Analyzer project:

- ▶ Choose **File** >  **Save** <file>.

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

To save the project file with a different name:

1. Choose **File** > **Save** <file> **As**.
The Save Reveal Analyzer File dialog box appears.
2. Browse to the directory in which you want to save the project.
3. In the File name box, type the file name.
4. Click **Save**.

Exiting Reveal Analyzer

To exit Reveal Analyzer:

- ▶ Choose **File** > **Close**.

User Interface Descriptions

The Reveal Analyzer window consists of the LA Trigger, LA Waveform.

LA Trigger Tab

The LA Trigger 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.

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:

Name Specifies the name of the trigger unit. See [“Trigger Expression and Trigger Unit Naming Conventions” on page 120](#) 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 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 five 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 18: 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 65,536. The maximum is 65,536. 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.

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. Since the trigger occurred before enough samples were captured to fill the buffer, both the position is different and the total number of captured samples will be less than the set value. This condition is more likely to occur using the post-trigger setting. To avoid this, do not use a trigger condition that occurs immediately or very frequently.

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.

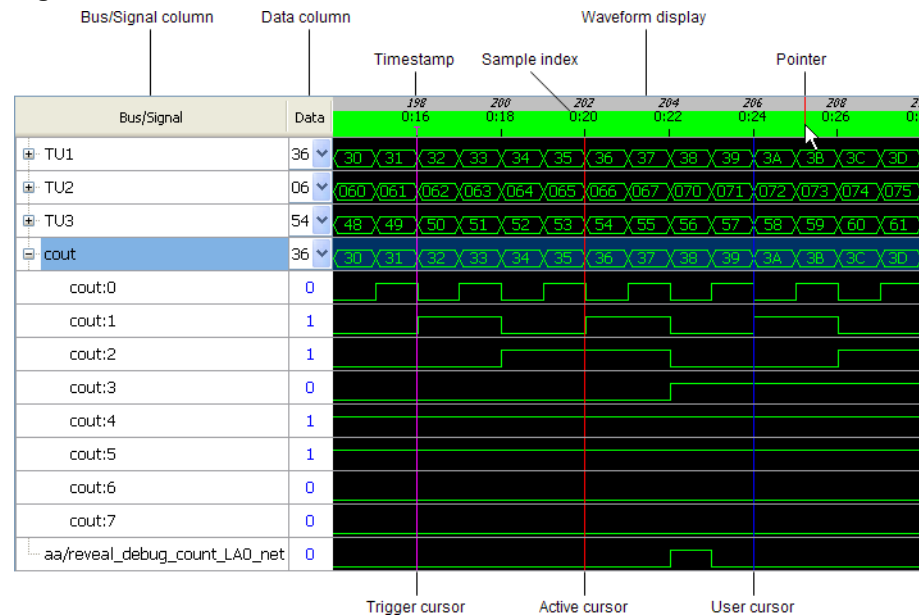
Trigger Expression and Trigger Unit Naming Conventions

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.

LA Waveform Tab

Waveforms are presented in a grid layout as shown in Figure 19 along with several features to help you find and analyze the data.

Figure 19: Elements of the LA Waveform View



Bus/Signal Column Displays the names of the trace buses and signals in the selected module.

Note:

When you copy and paste a duplicate or incorrect signal, you can delete it using either of the following keyboard shortcuts:

- ▶ Shift + Delete
- ▶ Ctrl + X

Data Column Displays the value of the bus or signal at the active cursor (a solid, red line that you can set in the waveform display). Buses also have a drop-down menu for setting the radix used in the Data column and in the waveform display. The menu includes token sets whose bit width matches the bus. See [“Setting the Trace Bus Radix” on page 83](#).

Waveform Display Displays the trace data in waveform format. When there is room, bus values are included in the display using the radix set in the Data column. You can zoom in and out, pan, and jump to various points.

The waveform display includes several other elements to help you read the display and analyze the data:

- ▶ **Timestamp.** The gray bar at the top of the display shows “timestamps” of the trace frames (actually, a simple count of the clock cycles). Timestamps are shown only if the Timestamp trace option was selected for the module in Reveal Inserter. See [“Adding Time Stamps to Trace Samples” on page 83](#).
- ▶ **Sample Index.** The green bar near the top of the display shows a count of triggers and trace samples within each trigger's data set. The sample indexes have the form *<trigger>:<sample>*. For example, 0:2 indicates the first trigger and the third trace sample for that trigger (the counts are zero-based). 2:10 indicates the third trigger and the eleventh trace sample for that trigger.
- ▶ **Pointer.** A red line that cuts across the timestamp and sample index bars, the pointer follows the horizontal movement of the mouse pointer across the waveform display. Use the pointer to see where you are in time as you examine the waveform.
- ▶ **Cursors.** Vertical lines cutting through all the signals, cursors mark moments in the waveform. See [“Placing, Moving, and Locating Cursors” on page 112](#).

Viewing and Saving Waveforms

After capturing data, you can view the trace data in waveform format in the LA Waveform tab. Whenever the trace stops, Reveal Analyzer reads the trace samples and automatically updates the signal waveforms.

To view a waveform:

1. Click the **LA Waveform** tab.
2. Choose a module from the pulldown menu in the Reveal Analyzer tool bar.

Sometimes the waveform includes fewer clock cycles than you expect; in particular, fewer before the trigger. This happens if the trigger occurs so quickly after the test starts that there are not enough clock cycles before the trigger to fill that part of the trace buffer.

You can also save the captured waveform as an RVA file and reload it when you reopen Reveal Analyzer. The saved waveform carries most of the

customization that you applied such as color changes, bus grouping, cursors, and others (except for zoom in/out information.)

To save a waveform:

1. Choose **File > Save <file> As**.
The Save Reveal Analyzer File dialog box appears.
2. Browse to the directory in which to save the project.
Make sure to save the .rva in the same directory as .rvl project file.
3. In the File name box, type the file name.
4. Click **Save**.

Using the Memory Controller Debug

The memory controller debug feature for Avant devices provides an interactive and targeted method for diagnosing and resolving DDR interface issues or stress-testing your design.

- ▶ Initiate various types of memory training routines such as write leveling, read gate training, and VREF training.
- ▶ Write directly to DDR mode registers and fine-tuning memory parameters such as drive strength, termination, and on-die VREF settings.
- ▶ Immediately assess the impact of your changes through the Margin Report and VREF Report. These are visual and quantitative feedback on signal margins and voltage reference calibration.
- ▶ Stress-test your design, even if not currently experiencing issues. You can create traffic patterns based on your use cases. Test different types of memory access patterns and see how the memory controller handles them.

The following section describes the memory controller debug feature. For complete information on the IP, see [FPGA-IPUG-02195, DDR Memory PHY Module](#).

Enabling the Memory Controller Debug

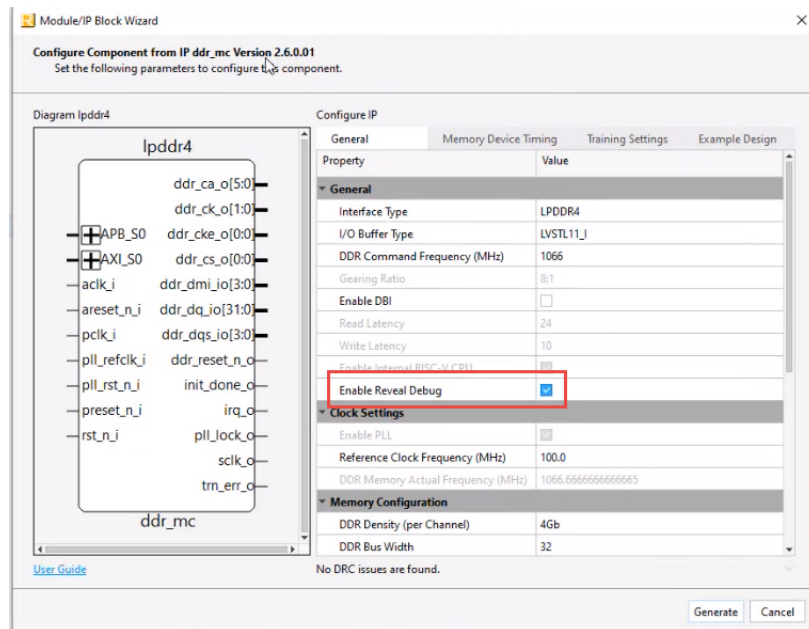
Enabling the memory controller debug is a two-step process:

- ▶ Enable Reveal debug during IP configuration
- ▶ Enable memory controller debug in Reveal Inserter

To enable Reveal debug during IP configuration:

1. In the Module/IP tree, select the component to generate.
2. Configure your settings in the Module/IP Block Wizard.

Check the **Enable Reveal Debug** option.



3. Click **Generate**.

To automatically import the .ipx file into your project when the component is generated, select **Insert to project** in the Check Generating Result dialog box.

4. Click **Finish**.

To enable the memory controller debug in Reveal Inserter:

1. Open Reveal Inserter. select **RTL (Pre-Synthesis)** in Debug Stage.
2. Click **Finish**.
3. In Datasets, click **Add Controller** to add a Controller core.
4. Click the **Hard IP Setup** tab.
5. Under Hard IP Setting, enable the IP by selecting its check box on the Enabled column.
6. Click the  **Insert Debug** button.
7. In the Insert Debug to Design dialog box, select the modules to insert.
8. Select **Activate Reveal file in design project**.

If the .rvl file is not active in the design project, the Reveal modules will not be included during synthesis.

9. Click **OK**.
10. In the Save Reveal Project dialog box, indicate the file name and click **Save**.

The active .rvl file is listed in the File List view under Debug Files.

11. Click **Export Files** to run the design flow and insert the Reveal controller.

Downloading the Bitstream and Starting a Reveal Analyzer Project

If the correct setup process is followed and Reveal inserter is properly configured, download the bitstream and start a new project in Reveal Analyzer/Controller.

Connect the board with the Avant device to your computer. Using Radiant Programmer, download the bitstream to the board. Refer to [“Programming the FPGA” on page 453](#).

After downloading the bitstream and making sure that there are no errors, open the Reveal Analyzer/Controller to create a new project.

Running Calibration

To open the memory controller debug interface and run calibration:

1. In Reveal Analyzer/Controller, choose **top_Controller** in the drop-down menu.
2. Click the **Hard IP** tab.

You can separate the tab by clicking the detach icon at the upper right corner.

3. The **Memory Interface Calibration** tab opens.

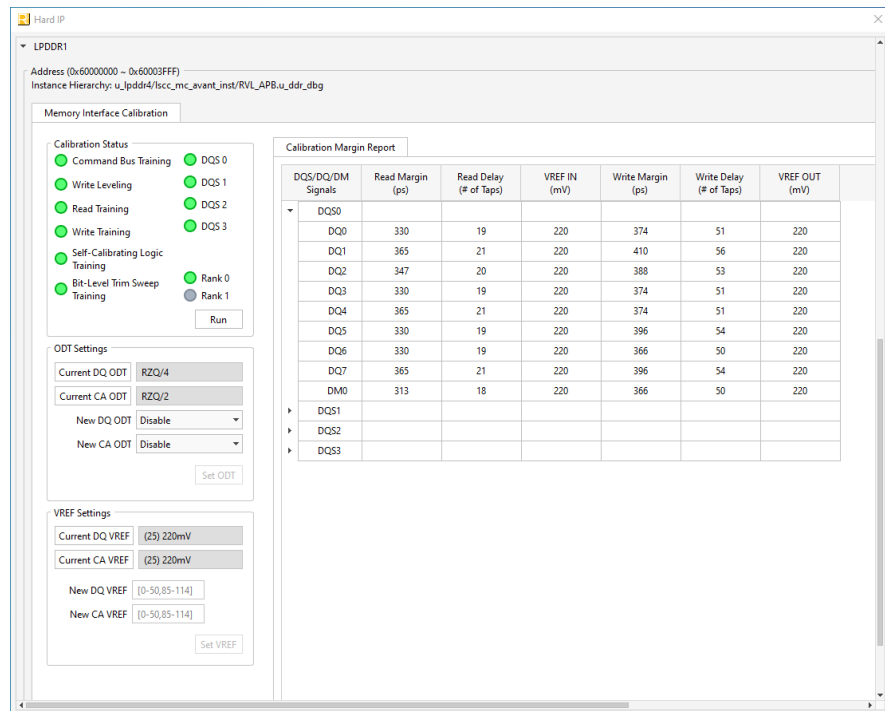
You can view the status of various memory controller debug tuning tests such as command bus training, write leveling, read training, and write training. You can also read and write from the mode registers of their memory controller IP.

4. Click the **Run** button.

This shows the calibration status, calibration margin report, current ODT, and VREF settings.

The Calibration Margin Report tab shows the margin for various DQS/DQ, command/control, and address signals.

For detailed information on the available options, refer to [FPGA-IPUG-02195, DDR Memory PHY Module](#).



When you run the calibration, a .csv file of the results is automatically saved in the directory of your .rva file.

Note

This feature is repeatable with TCL commands, with or without the GUI.

Best Practices

This section presents a set of best practices for leveraging the Lattice Radiant Reveal debug tool in FPGA development workflows. Reveal offers visibility into internal signal behavior, enabling precise timing analysis, logic verification, and system-level debugging. Designed for engineers familiar with RTL design and synthesis, this best practices guide focuses on optimizing trace configurations, minimizing resource overhead, and integrating Reveal into iterative development cycles. By applying these practices, you can enhance debugging efficiency, accelerate root cause analysis, and improve overall design robustness.

This documents includes the following sections:

- ▶ [“General Guidelines” on page 127](#)
- ▶ [“Best Practices for Post-Synthesis Flow” on page 128](#)

This best practices guide is a **work in progress**. As Reveal continues to evolve and new use cases emerge, additional recommendations, clarifications, and examples will be incorporated. Users are encouraged to refer to the latest version of this document and provide feedback to help improve its accuracy and relevance.

General Guidelines

To maximize the effectiveness of the Reveal debug tool and ensure accurate signal capture and analysis, follow the general usage guidelines in this section.

▶ **Prefer Registered Signals for Trigger and Trace**

Always select registered signals (such as signals driven by flip-flops for example) for both trigger and trace. Avoid combinatorial signals, as they are prone to glitches and may produce unreliable or misleading trace data due to their asynchronous nature.

▶ **Use `syn_rvl_debug` Attributes in RTL**

Annotate the registered signals intended for debugging with the `syn_rvl_debug` attribute directly in the RTL code. This highlights the signals and ensures that they are preserved through synthesis and made available for Reveal debug configuration.

▶ **Debug at the Post-Synthesis Stage**

Perform debugging at the post-synthesis stage to ensure that the signals selected for tracing reflect the actual synthesized netlist. This stage provides a more accurate representation of the design and avoids discrepancies between RTL and implementation.

▶ **Select the Sample Clock First in Inserter GUI**

In the Reveal Inserter GUI, begin by selecting the sample clock. This clock determines the timing domain for signal sampling and must be chosen before selecting any signals.

▶ **Group Signals by Clock Domain**

After selecting the sample clock, choose only the signals that belong to the same clock domain for a single analyzer core. Mixing signals from different domains within the same core can lead to incorrect timing analysis and data corruption.

▶ **Use Separate Analyzer Cores for Different Clocks**

For designs with multiple clock domains, instantiate separate analyzer cores for each domain. This ensures proper synchronization and avoids clock domain crossing issues during trace capture.

▶ **Review the Trace Report for CDC Issues**

After instrumentation, carefully examine the trace report for any clock domain crossing (CDC) warnings or errors. These issues can compromise the integrity of captured data and should be resolved before proceeding with debug.

▶ **Minimize Trigger Unit Complexity in Tight Timing Designs**

If your design is close to failing timing constraints, reduce the number of signals in the trigger unit. Each additional signal can increase logic depth and routing complexity, potentially pushing the design over timing limits.

Best Practices for Post-Synthesis Flow

The post-synthesis debug flow in Reveal provides a more accurate and implementation-aware environment for signal tracing and analysis. At this stage, the design has undergone synthesis, allowing you to work with actual netlist data rather than abstract RTL representations. This section outlines key strategies for configuring Reveal effectively during post-synthesis.

Guidelines when Updating User Design in Post-Synthesis Debug Flow

▶ **Disable All Active Reveal Projects (*.rvl)**

Before making any changes to the design, deactivate all active Reveal projects. This prevents conflicts between the existing debug configuration and the updated design, and ensures that Reveal does not attempt to instrument outdated netlist data.

▶ **Synthesize the User Design After Design Update**

Once the design has been modified, run synthesis to generate an updated netlist. This step is crucial because Reveal operates on post-synthesis data, and any changes to the RTL must be reflected in the synthesized output.

▶ **Create a New Reveal Project at the Post-Synthesis Stage**

After synthesis, create a new Reveal project and select the post-synthesis debug stage. This ensures that the tool uses the latest netlist and allows you to configure signal tracing based on the updated design.

▶ **Activate Existing Post-Synthesis Reveal Project (If Applicable)**

If a Reveal project already exists for the post-synthesis stage and matches the updated design, it can be reactivated. This saves time by reusing previous configurations, provided they are still valid for the current netlist.

▶ **Run Synthesis to Generate Reveal Core Only**

After configuring the Reveal project, run synthesis again, but this time it will synthesize only the Reveal analyzer core. This step integrates the debug logic into the design without re-synthesizing the entire user logic, streamlining the process.

Guidelines when Changing the Synthesis Tool

▶ **Disable All Active Reveal Projects (*.rvl)**

Before switching synthesis tools, deactivate any active Reveal projects. This prevents conflicts between the previous synthesis output and the new tool's netlist structure. Reveal projects are tightly coupled with the synthesis tool used, so disabling them ensures a clean transition.

▶ **Synthesize the User Design After Selecting a Different Synthesis Tool**

Once the new synthesis tool is selected, re-run synthesis on the user design. This step generates a fresh netlist compatible with the selected tool and ensures that Reveal can correctly interpret the design structure for instrumentation.

▶ **Create a New Reveal Project at the Post-Synthesis Stage**

After synthesis is complete, create a new Reveal project and select the post-synthesis debug stage. This ensures that the Reveal configuration aligns with the netlist generated by the new synthesis tool, allowing accurate signal tracing and core insertion.

Revision History

The following table provides the revision history for this document.

Date	Version	Description
10/8/2026	2026.1.1	Updated to reflect changes in Radiant 2026.1.1 software.
06/26/2026	2026.1	Updated to reflect changes in Radiant 2026.1 software.
03/27/2026	2025.2.1	Updated to reflect changes in Radiant 2025.2.1 software.
12/11/2025	2025.2	Updated to reflect changes in Radiant 2025.2 software.
09/26/2025	2025.1.1	Updated to reflect changes in Radiant 2025.1.1 software.
06/26/2025	2025.1	Updated to reflect changes in Radiant 2025.1 software.
03/31/2025	2024.2.1	Updated to reflect changes in Radiant 2024.2.1 software.
12/20/2024	2024.2	Updated to reflect changes in Radiant 2024.2 software.
06/28/2024	2024.1	Updated to reflect changes in Radiant 2024.1 software.
03/29/2024	2023.2.1	Updated to reflect changes in Radiant 2023.2.1 software.
11/27/2023	2023.2	Updated to reflect changes in Radiant 2023.2 software.
06/5/2023	2023.1	Updated to reflect changes in Radiant 2023.1 software.
02/27/2023	2022.1.1	Updated to reflect changes in Radiant 2022.1.1 software.
11/7/2022	2022.1	Updated to reflect changes in Radiant 2022.1 software.
06/6/2022	3.2	Updated to reflect changes in Radiant 3.2 software.
12/7/2021	3.1	Updated to reflect changes in Radiant 3.1 software.
05/1/2020	2.1	Updated to reflect changes in Radiant 2.1 software.
10/23/2019	2.0	Initial Release. Content was previously Appendix A of "Radiant Software User Guide."