UG0773
User Guide
PolarFire FPGA SmartDebug
Software v12.2

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. **View the online help included with software to enable all linked content.**

Microsemi
a Microchip company

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com.**

5-02-00773-6/08.19

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email:
sales.support@microsemi.com
www.microsemi.com

©2019 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Table of Contents

Table of Contents 2

Welcome to SmartDebug 3
Introduction to SmartDebug 3
Getting Started with SmartDebug 4
Using SmartDebug 4
Create Standalone SmartDebug Project 4

SmartDebug User Interface 7
Standalone SmartDebug User Interface 7
Programming Connectivity and Interface 8
View Device Status 12

Debugging 15
Debug FPGA Array 15
Hierarchical View 15
Netlist View 17
Live Probes 19
Active Probes 19
Probe Grouping (Active Probes Only) 20
Memory Blocks 24
Probe Insertion (Post-Layout) 29
Event Counter 32
Frequency Monitor 35
FPGA Hardware Breakpoint Auto Instantiation 40
User Clock Frequencies 47
Debug sNVM 47
Debug Transceiver 51
Debug uPROM 75

SmartDebug Tcl Commands 80

Frequently Asked Questions 81
How do I monitor a static or pseudo-static signal? 81
How do I force a signal to a new value? 81
How do I count the transitions on a signal? 82
How do I monitor or measure a clock? 83
How do I perform simple SmartBERT tests? 85
How do I read LSRAM or USRAM content? 85
How do I change the content of LSRAM or USRAM? 86
How do I read the health check of the Transceiver? 88

Welcome to SmartDebug

Introduction to SmartDebug

Design debug is a critical phase of FPGA design flow. Microsemi's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug provides access to non-volatile memory (sNVM), SRAM, transceiver, uPROM, and probe capabilities. Microsemi PolarFire FPGA devices have built-in probe logic that greatly enhance the ability to debug logic elements within the device. SmartDebug accesses the built-in probe points through the Active Probe and Live Probe features, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

Use Models

SmartDebug can be run in the following modes:

- Integrated mode from the Libero Design Flow
- Standalone mode
- Demo mode

Integrated Mode

When run in integrated mode from Libero, SmartDebug can access all design and programming hardware information. No extra setup step is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window, expand **Debug Design** and double-click **SmartDebug Design**.

Standalone Mode

SmartDebug can be installed separately in the setup containing FlashPro Express and Job Manager. This provides a lean installation that includes all the programming and debug tools to be installed in a lab environment for debug. In this mode, SmartDebug is launched outside of the Libero Design Flow. When launched in standalone mode, you must to go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all debug features in the supported devices.

Note: In standalone mode, the Probe Insertion feature is not available in FPGA Array Debug, as it requires incremental routing to connect the user net to the specified I/O.

Demo Mode

Demo mode allows you to experience SmartDebug features (Active Probe, Live Probe, Memory Blocks, Transceiver, Debug sNVM, Debug uPROM) without connecting a board to the system running SmartDebug.

Note: SmartDebug demo mode is for demonstration purposes only, and does not provide the functionality of integrated mode or standalone mode.

Note: You cannot switch between demo mode and normal mode while SmartDebug is running.

Standalone Mode Use Model Overview

In the main use model for standalone SmartDebug, the DDC file must be generated from Libero and imported into a SmartDebug project to obtain full access to the device debug features. Alternatively, SmartDebug can be used without a DDC file with a limited feature set.

Supported Families, Programmers, and Operating Systems

Programming and Debug: PolarFire

Programmers: FlashPro, FlashPro3, FlashPro4, FlashPro5, FlashPro6

Operating Systems: Windows 7, Windows 10, RHEL 6.x, RHEL 7.x, Cent OS 6, and Cent OS 7

Getting Started with SmartDebug

This topic introduces the basic elements and features of SmartDebug. If you are already familiar with the user interface, proceed to the Solutions to Common Issues Using SmartDebug or Frequently Asked Questions sections.

SmartDebug enables you to use JTAG to interrogate and view embedded silicon features and device status .

See Using SmartDebug for an overview of the use flow.

You can use the debugger to:

- Get device status and view diagnostics

Using SmartDebug

The most common flow for SmartDebug is:

1. Create your design. You must have a FlashPro programmer connected to use SmartDebug.
2. Expand **Debug Design** and double-click **Smart Debug Design** in the Design Flow window. SmartDebug opens for your target device.
3. Click **View Device Status** to view the device status report and check for issues.
4. Examine individual silicon features, such as FPGA debug.

Create Standalone SmartDebug Project

A Standalone SmartDebug project can be configured in two ways:

- Import DDC files exported from Libero
- Construct Automatically

From the SmartDebug main window, click **Project** and choose **New Project**. The Create SmartDebug Project dialog box opens.

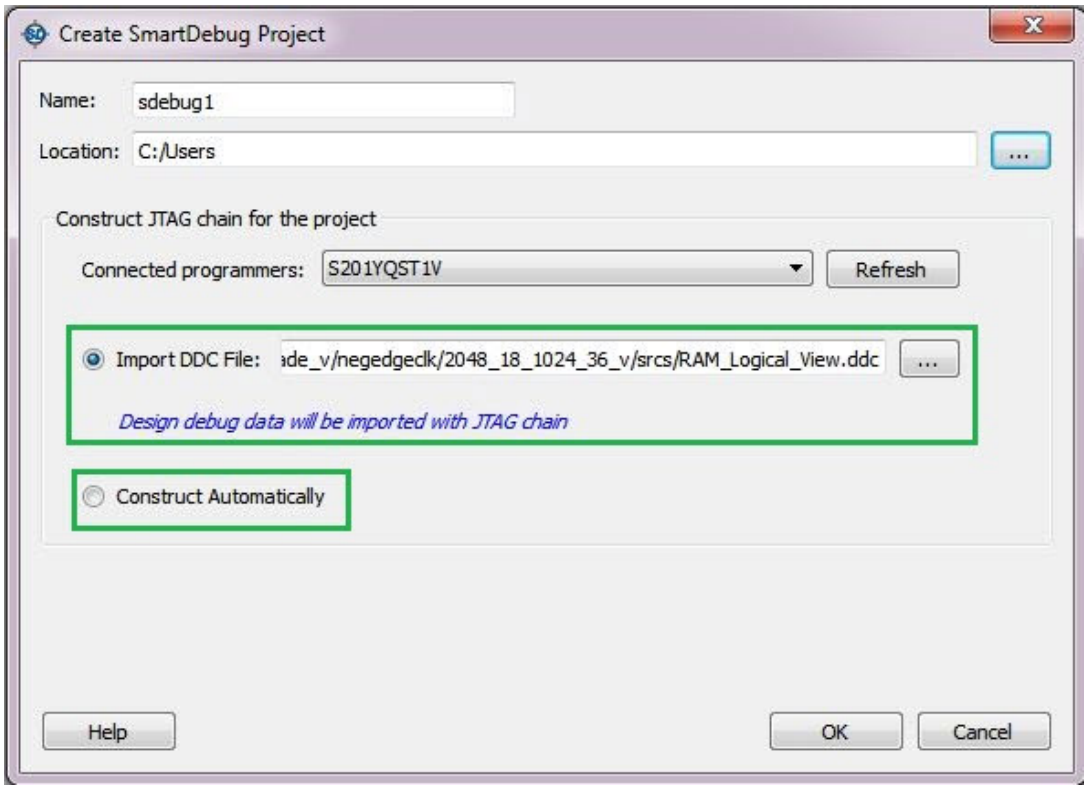


Figure 1 · Create SmartDebug Project Dialog Box

Import from DDC File (created from Libero)

When you select the **Import from DDC File** option in the Create SmartDebug Project dialog box, the Design Debug Data of the target device and all hardware and JTAG chain information present in the DDC file exported in Libero are automatically inherited by the SmartDebug project. The programming file information loaded onto other Microsemi devices in the chain is also transferred to the SmartDebug project.

Debug data is imported from the DDC file (created through Export SmartDebug Data in Libero) into the debug project, and the devices are configured using data from the DDC file.

If the DDC version and software version are not compatible, project creation is not allowed, and you must run **Generate SmartDebug FPGA Array Data**. Then click **Export SmartDebug Data** to export a new DDC file and use it for project creation.

Construct Automatically

When you select the **Construct Automatically** option, a debug project is created with all the devices connected in the chain for the selected programmer. This is equivalent to Construct Chain Automatically in FlashPro.

Configuring a Generic Device

For Microsemi devices having the same JTAG IDCODE (i.e., multiple derivatives of the same Die), the device type must be configured for SmartDebug to enable relevant features for debug. The device can be configured by loading the programming file, by manually selecting the device using Configure Device, or by importing DDC files through Programming Connectivity and Interface. When the device is configured, all debug options are shown.

For debug projects created using Construct Automatically, you can use the following options to debug the devices:

- Load the programming file – Right-click the device in Programming Connectivity and Interface.
- Import Debug Data from DDC file – Right-click the device in Programming Connectivity and Interface.

The appropriate debug features of the targeted devices are enabled after the programming file or DDC file is imported.

Connected FlashPRO Programmers

The drop-down lists all FlashPro programmers connected to the device. Select the programmer connected to the chain with the debug device. At least one programmer must be connected to create a standalone SmartDebug project.

Before a debugging session or after a design change, program the device through Programming Connectivity and Interface.

See Also

Programming Connectivity and Interface

View Device Status

SmartDebug User Interface

Standalone SmartDebug User Interface

You can start standalone SmartDebug from the Libero installation folder or from the FlashPRO installation folder.

Windows:

<Libero Installation folder>/Designer/bin/sdebug.exe

<FlashPRO Installation folder>/bin/sdebug.exe

Linux:

<Libero Installation folder>/ bin/sdebug

<FlashPRO Installation folder>/bin/sdebug

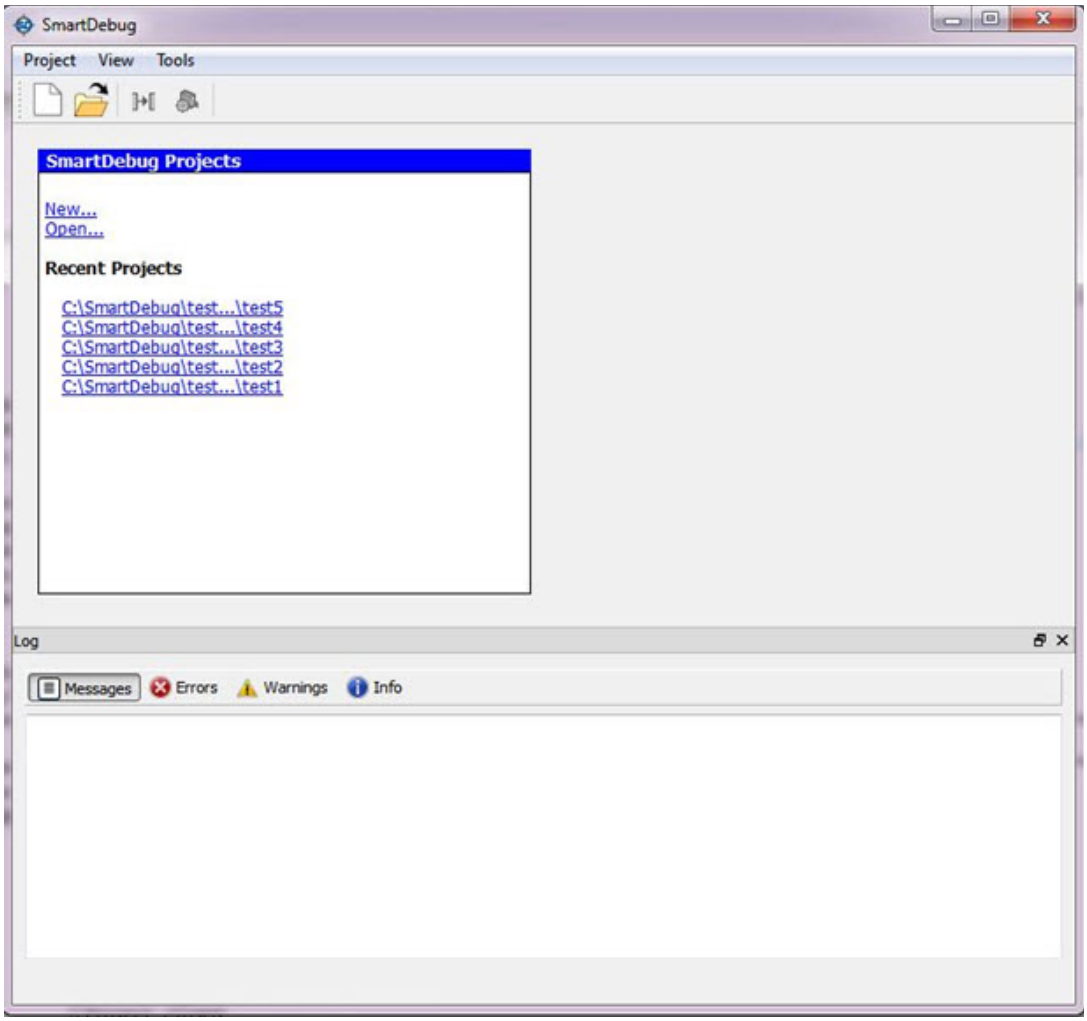


Figure 2 · Standalone SmartDebug Main Window

Project Menu

The Project menu allows you to do the following:

- Create new SmartDebug projects (**Project > New Project**)
- Open existing debug projects (**Project > Open Project**)

- Execute SmartDebug-specific Tcl scripts (**Project > Execute Script**)
- Export SmartDebug-specific commands to a script file (**Project > Export Script File**)
- See a list of recent SmartDebug projects (**Project > Recent Projects**).

Log Window

SmartDebug displays the Log window by default when it is invoked. To suppress the Log window display, click the View menu and toggle **View Log**.

The Log window has four tabs:

Messages – displays standard output messages

Errors – displays error messages

Warnings – displays warning messages

Info – displays general information

Tools Menu

The Tools menu includes Programming Connectivity and Interface and Programmer Settings options, which are enabled after creating or opening a SmartDebug project.

Programming Connectivity and Interface

To open the Programming Connectivity and Interface dialog box, from the standalone SmartDebug Tools menu, choose **Programming Connectivity and Interface**. The Programming Connectivity and Interface dialog box displays the physical chain from TDI to TDO.

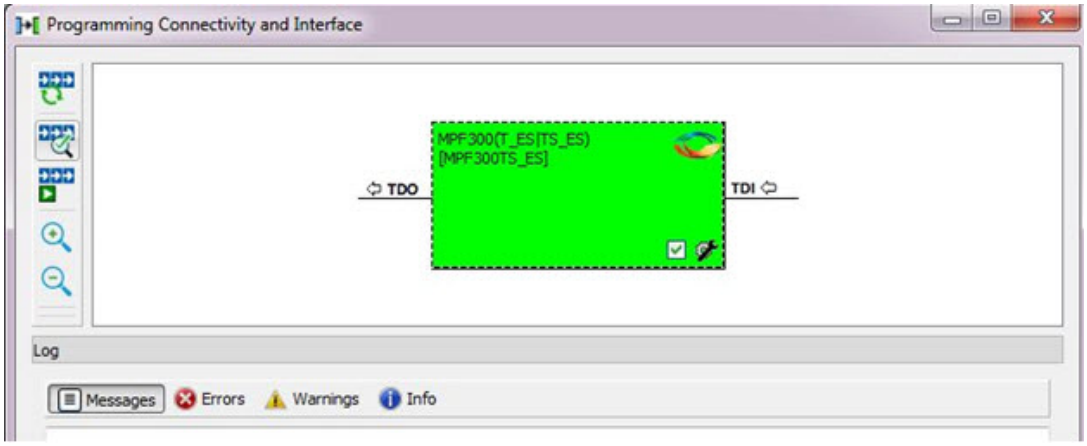


Figure 3 · Programming Connectivity and Interface Dialog Box – Project created using Import from DDC File

All devices in the chain are disabled by default when a standalone SmartDebug project is created using the **Construct Automatically** option in the Create SmartDebug Project dialog box.

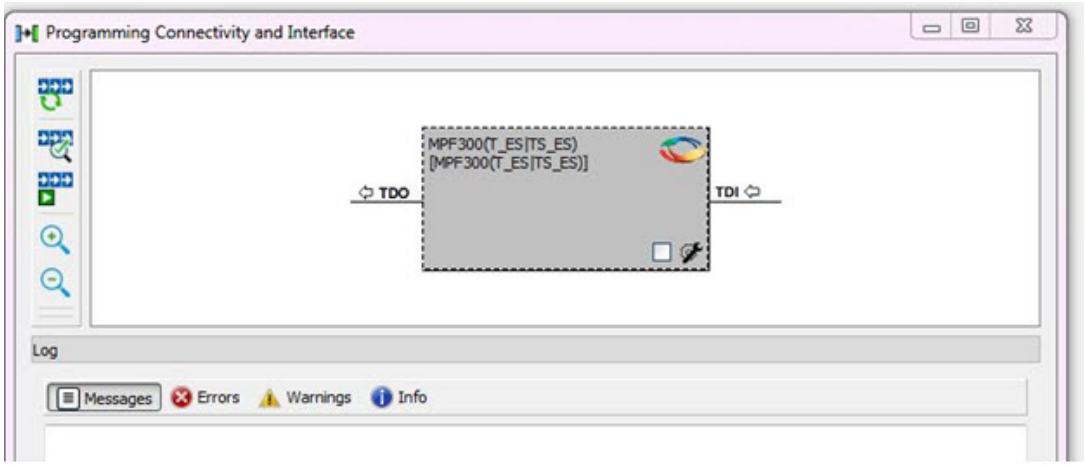


Figure 4 · Programming Connectivity and Interface window – Project created using Construct Automatically

The Programming Connectivity and Interface dialog box includes the following actions:

- **Construct Chain Automatically** - Automatically construct the physical chain.

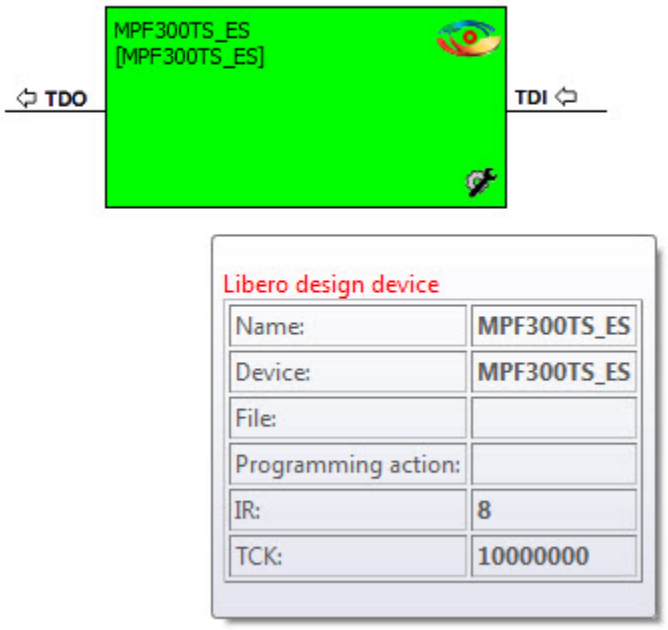
  Running Construct Chain Automatically in the Programming Connectivity and Interface removes all existing debug/programming data included using DDC/programming files. The project is the same as a new project created using the Construct Chain Automatically option.
- **Scan and Check Chain** – Scan the physical chain connected to the programmer and check if it matches the chain constructed in the scan chain block diagram.
- **Run Programming Action** – Option to program the device with the selected programming procedure.

  When two devices are connected in the chain, the programming actions are independent of the device.
- **Zoom In** – Zoom into the scan chain block diagram.
- **Zoom Out** – Zoom out of the scan chain block diagram.

Hover Information

The device tooltip displays the following information if you hover your cursor over a device in the scan chain block diagram:

- **Name:** User-specified device name. This field indicates the unique name specified by the user in the Device Name field in Configure Device (right-click **Properties**).
- **Device:** Microsemi device name.
- **Programming File:** Programming file name.
- **Programming action:** The programming action selected for the device in the chain when a programming file is loaded.
- **IR:** Device instruction length.
- **TCK:** Maximum clock frequency in MHz to program a specific device; standalone SmartDebug uses this information to ensure that the programmer operates at a frequency lower than the slowest device in the chain.



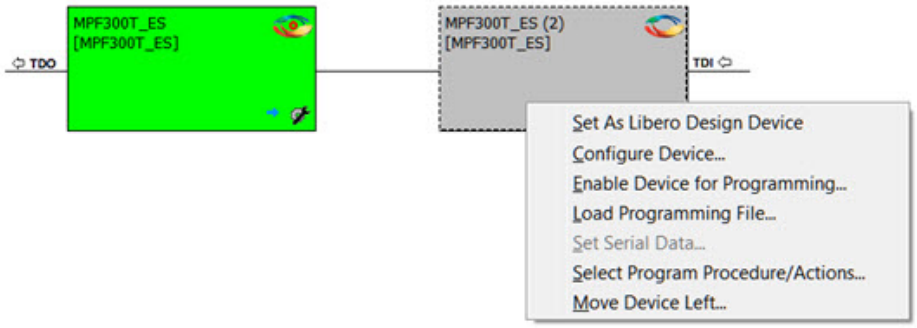
Device Chain Details

The device within the chain has the following details:

- User-specified device name
- Device name
- Programming file name
- Programming action – Select **Enable Device for Programming** to enable the device for programming. Enabled devices are green, and disabled devices are grayed out.

Right-click Properties

The following options are available when you right-click a device in the Programming Connectivity and Interface dialog box.



Set as Libero Design Device - The user needs to set Libero design device when there are multiple identical Libero design devices in the chain.

Configure Device - Ability to reconfigure the device.

- **Family and Die**: The device can be explicitly configured from the Family, Die drop-down.
- **Device Name**: Editable field for providing user-specified name for the device.

Enable Device for Programming - Select to enable the device for programming. Enabled devices are shown in green, and disabled devices are grayed out.

Load Programming File - Load the programming file for the selected device.

Select Programming Procedure/Actions- Option to select programming action/procedures for the devices connected in the chain.

- **Actions**: List of programming actions for your device.
- **Procedures**: Advanced option; enables you to customize the list of recommended and optional procedures for the selected action.

Import Debug Data from DDC File - Option to import debug data information from the DDC file.

Note: This option is supported when SmartDebug is invoked in standalone mode.

The DDC file selected for import into device must be created for a compatible device. When the DDC file is imported successfully, all current device debug data is removed and replaced with debug data from the imported DDC file.

The JTAG Chain configuration from the imported DDC file is ignored in this option.

If a programming file is already loaded into the device prior to importing debug data from the DDC file, the programming file content is replaced with the content of the DDC file (if programming file information is included in the DDC file).

Debug Context Save

Debug context refers to the user selections in debug options such as Debug FPGA Array, Debug Transceiver, and View Flash Memory Content. In standalone SmartDebug, the debug context of the current session is saved or reset depending on the user actions in Programming Connectivity and Interface.

The debug context of the current session is retained for the following actions in Programming Connectivity and Interface:

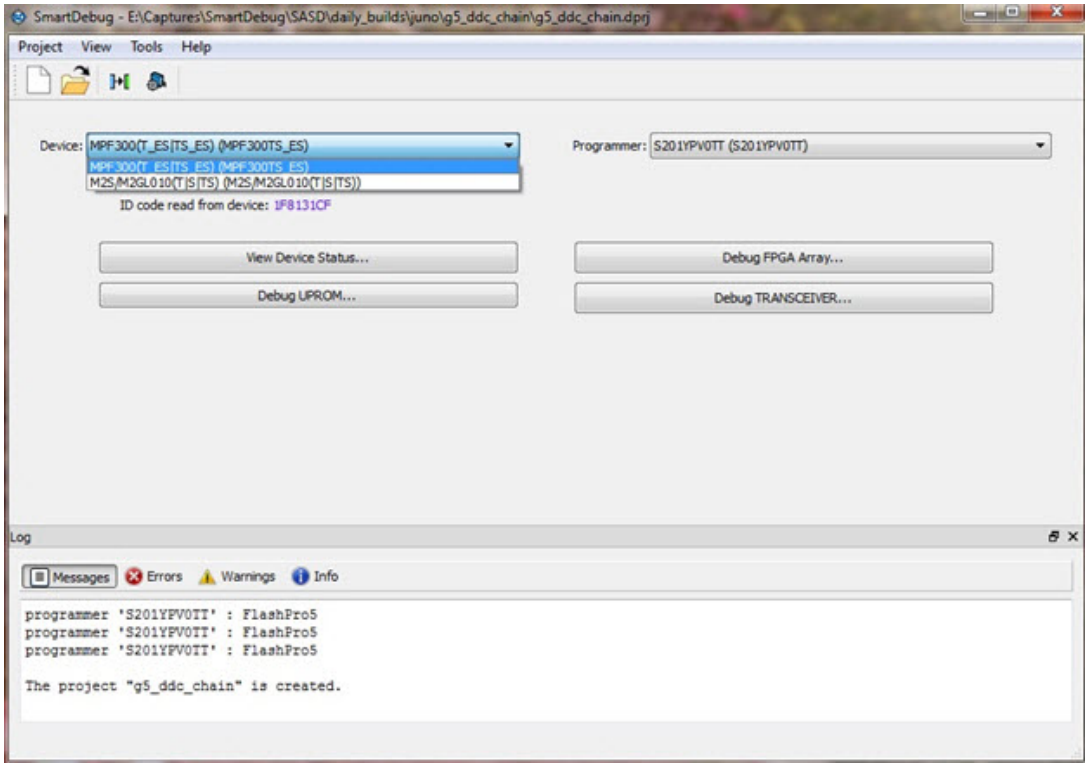
- Enable Device for Programming
- Select Programming Procedure/Actions
- Scan and Check Chain
- Run Programming Action

The debug context of the current session is reset for the following actions in Programming Connectivity and Interface:

- Auto Construct – Clears all the existing debug data. You need to reimport the debug data from DDC file.
- Import Debug Data from DDC file
- Configure Device – Renaming the device in the chain
- Configure Device – Family/Die change
- Load Programming File

Selecting Devices for Debug

Standalone SmartDebug provides an option to select the devices connected in the JTAG chain for debug. The device debug context is not saved when another debug device is selected.



View Device Status

Click **View Device Status** in the standalone SmartDebug main window to display the Device Status Report. The Device Status Report is a complete summary of IDCode, device certificate, design information, programming information, digest, and device security information. Use this dialog box to save or print your information for future reference.

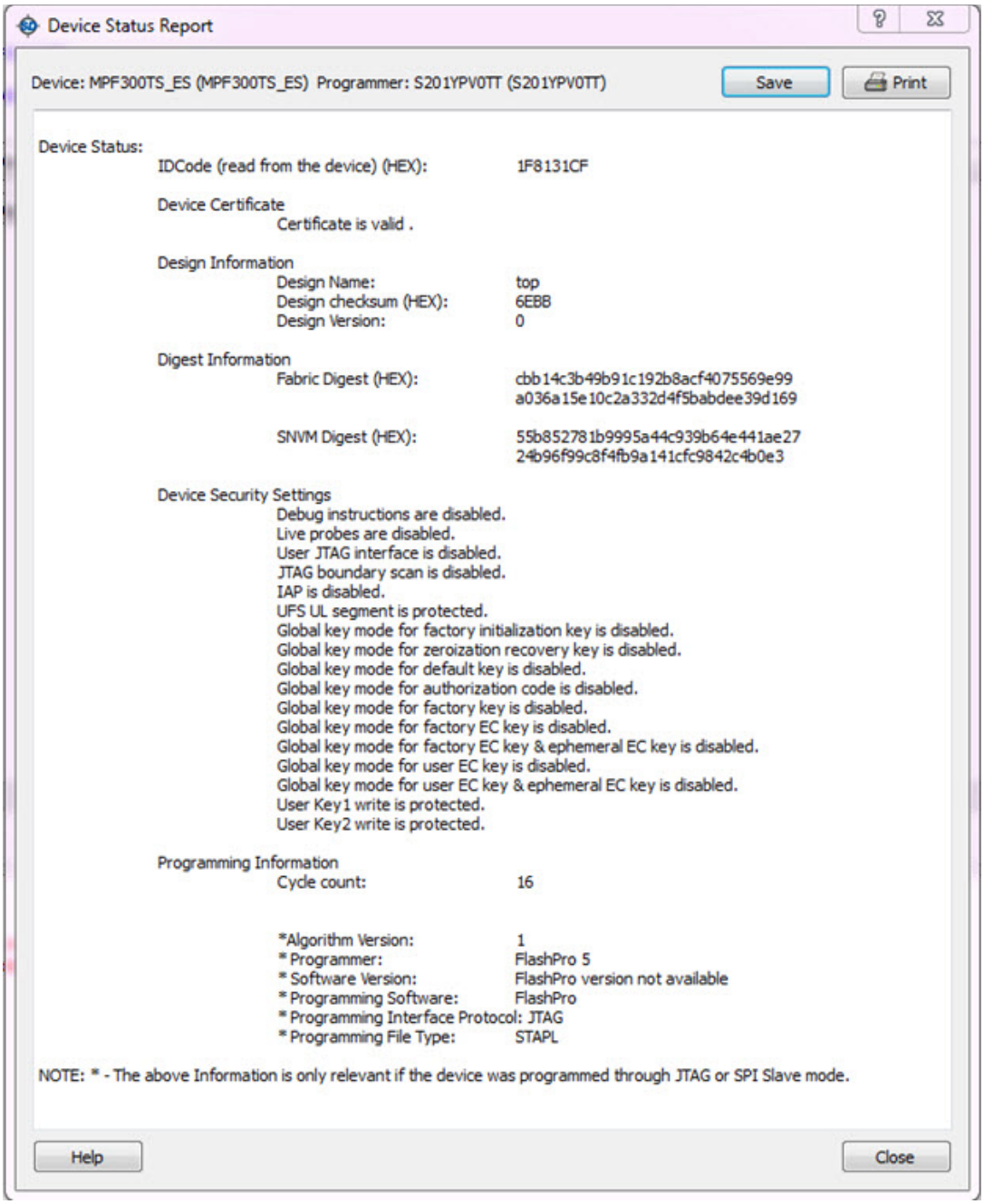


Figure 5 · Device Status Report

IdCode

IDCode read from the device under debug.

Device Certificate

Device certificate displays Family and Die information if device certificate is installed on the device.

If the device certificate is not installed on the device, a message indicating that the device certificate may not have been installed is shown.

Design Information

Design Information displays the following:

- Design Name
- Design Checksum
- Design Version

Digest Information

Digest Information displays Fabric Digest, sNVM Digest (if applicable) computed from the device during programming. sNVM Digest is shown when sNVM is used in the design.

Device Security Settings

Device Security Settings displays information about your security settings, including live probes, JTAG boundary scan, global key modes, and user keys.

Programming Information

Programming Information displays the following:

- Cycle Count
- Algorithm Version
- Programmer
- Software Version
- Programming Software
- Programming Interface Protocol
- Programming File Type

Debugging

Debug FPGA Array

In the Debug FPGA Array dialog box, you can view your Live Probes, Active Probes, Memory Blocks, and Insert Probes (Probe Insertion).

The Debug FPGA Array dialog box includes the following four tabs:

- Live Probes
- Active Probes
- Memory Blocks
- Probe Insertion

It also includes the FPGA Hardware Breakpoint (FHB) controls, consisting of the following tabs:

- "Event Counter" on page 32
- "Frequency Monitor" on page 35
- "User Clock Frequencies" on page 47

Hierarchical View

The Hierarchical View lets you view the instance level hierarchy of the design programmed on the device and select the signals to add to the Live Probes, Active Probes, and Probe Insertion tabs in the Debug FPGA Array dialog box. Logical and physical Memory Blocks can also be selected.

- **Instance** – Displays the probe points available at the instance level.
- **Primitives** – Displays the lowest level of probeable points in the hierarchy for the corresponding component —i.e., leaf cells (hard macros on the device).

You can expand the hierarchy tree to see lower level logic.

Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree.

The probe points can be added by selecting any instance or the leaf level instance in the Hierarchical View. Adding an instance adds all the probe able points available in the instance to Live Probes, Active Probes, and Probe Insertion.

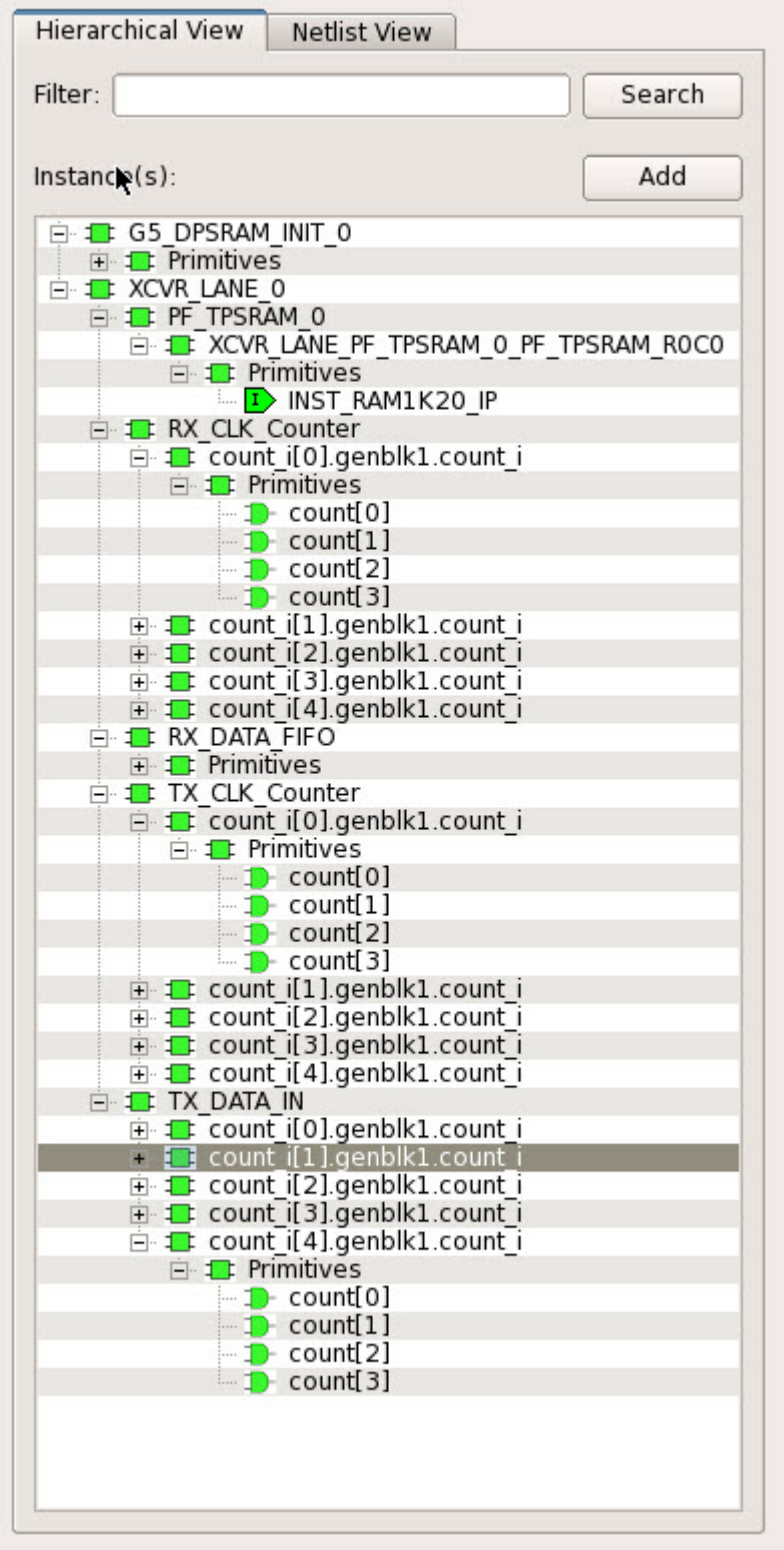


Figure 6 · Hierarchical View

Search

In Live Probes, Active Probes, Memory Blocks, and the Probe Insertion UI, a search option is available in the Hierarchical View. You can use wildcard characters such as * or ? in the search column for wildcard matching.

Probe points of leaf level instances resulting from a search pattern can only be added to Live Probes, Active Probes, and the Probe Insertion UI. You cannot add instances of search results in the Hierarchical View.

Netlist View

The Netlist View displays a flattened net view of all the probe-able points present in the design, along with the associated cell type.

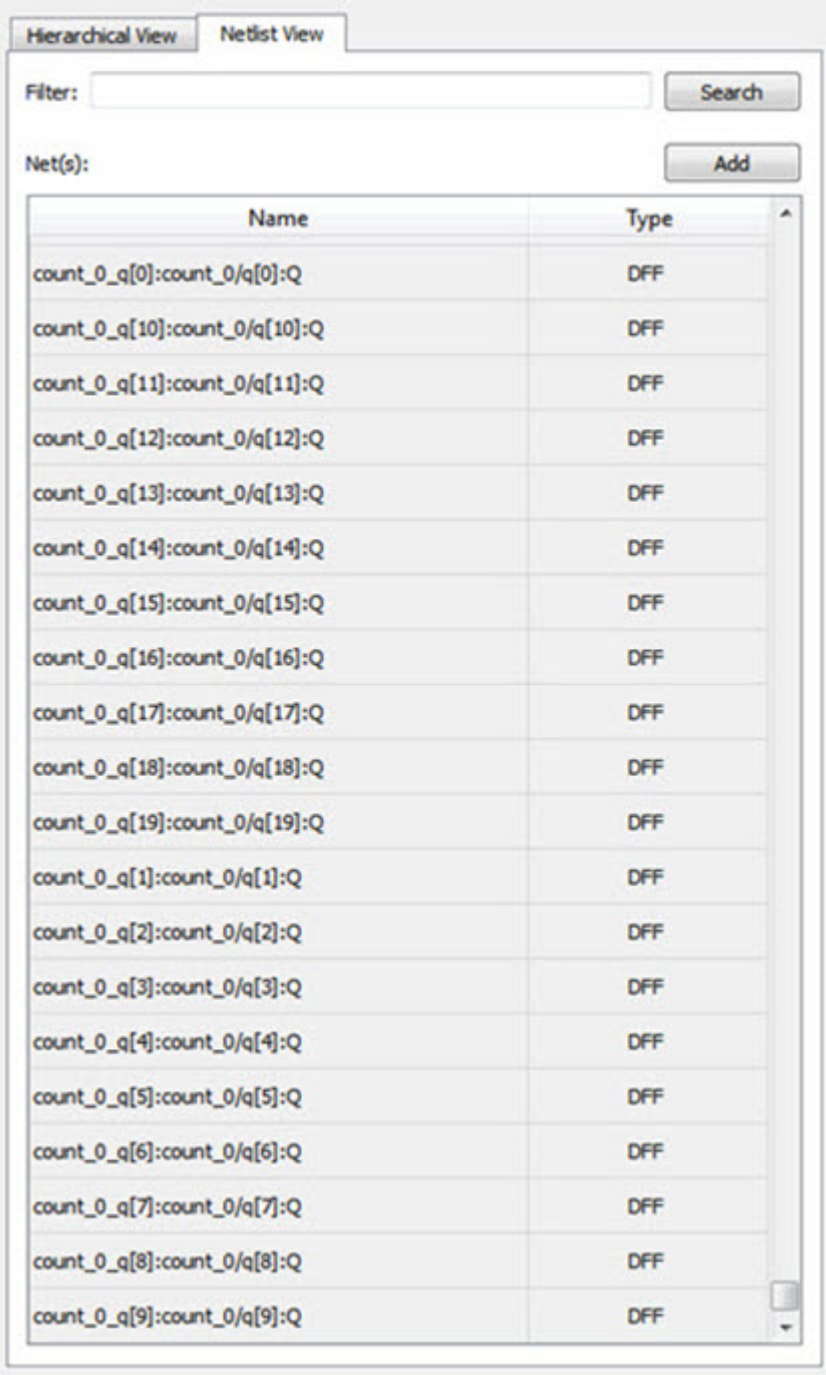


Figure 7 · Netlist View

Search

A search option is available in the Netlist View for Live Probes, Active Probes, and Probe Insertion. You can use wildcard characters such as * or ? in the search column for wildcard matching.

Live Probes

Live Probes is a design debug option that uses non-intrusive real time scoping of up to two probe points with no design changes.

The Live Probes tab in the Debug FPGA Array dialog box displays a table with the probe names and pin types.

There are two channels, and Live Probe can be assigned/unassigned independently.

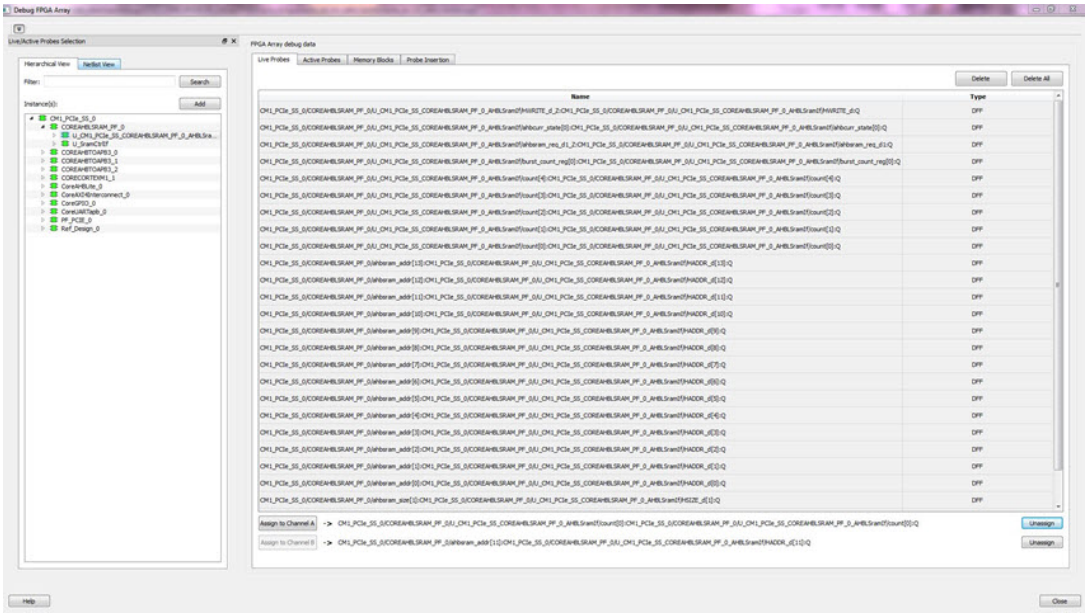


Figure 8 · Live Probes Tab in SmartDebug FPGA Array Dialog Box

Two probe channels (ChannelA and ChannelB) are available. When a probe name is selected, it can be assigned to either ChannelA or ChannelB.

You can assign a probe to a channel by doing either of the following:

- Right-click a probe in the table and choose **Assign to Channel A** or **Assign to Channel B**.
- Click the **Assign to Channel A** or **Assign to Channel B** button to assign the probe selected in the table to the channel. The buttons are located below the table.

When the assignment is complete, the probe name appears to the right of the button for that channel, and SmartDebug configures the ChannelA and ChannelB I/Os to monitor the desired probe points. Because there are only two channels, a maximum of two internal signals can be probed simultaneously.

Click the **Unassign Channels** button to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

Note: Both probes can be assigned/unassigned independently.

Live Probes in Demo Mode

You can assign and unassign Live Probes ChannelA and ChannelB in Demo Mode.

Active Probes

Active Probes is a design debug option to read and write to one or many probe points in the design through JTAG.

In the left pane of the Active Probes tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are listed with the Name and Type (which is the physical location of the flip-flop) in the Netlist View.

Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the Filter box.

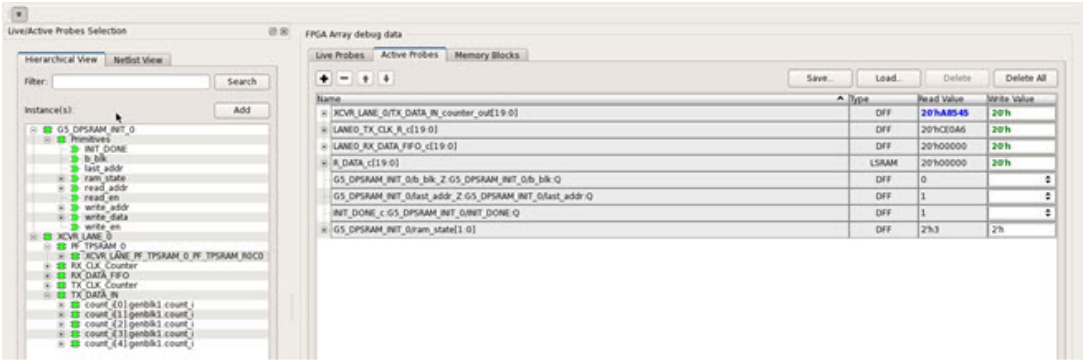


Figure 9 · Active Probes Tab in SmartDebug FPGA Array Dialog Box

When you have selected the desired probe, points appear in the Active Probe Data chart and you can read and write multiple probes (as shown in the figure below).

You can use the following options in the Write Value column to modify the probe signal added to the UI:

- Drop-down menu with values ‘0’ and ‘1’ for individual probe signals
- Editable field to enter data in hex or binary for a probe group or a bus

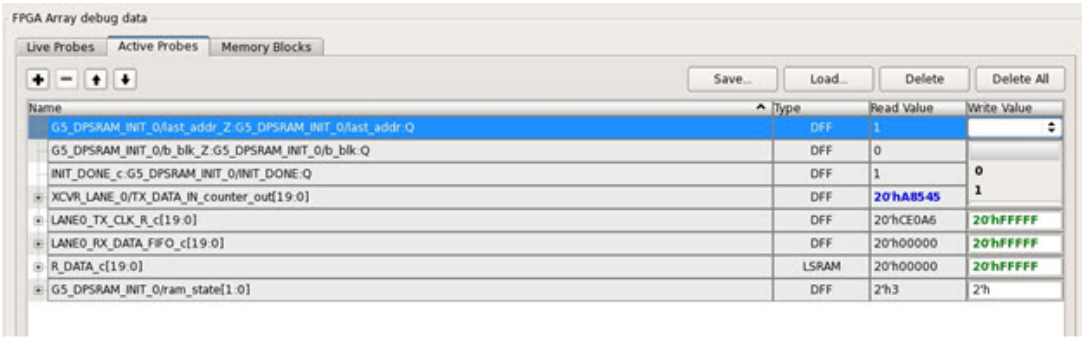


Figure 10 · Active Probes Tab - Write Value Column Options

Active Probes in Demo Mode

In demo mode, a temporary probe data file with details of current and previous values of probes added in the active probes tab is created in the designer folder. The write values of probes are updated to this file, and the GUI is updated with values from this file when you click Write Active Probes. Data is read from this file when you click Read Active Probes. If there is no existing data for a probe in the file, the read value displays all 0s. The value is updated based on your changes.

Probe Grouping (Active Probes Only)

During the debug cycle of the design, designers often want to examine the different signals. In large designs, there can be many signals to manage. The Probe Grouping feature assists in comprehending multiple signals as a single entity. This feature is applicable to Active Probes only. Probe nets with the same name are automatically grouped in a bus when they are added to the Active Probes tab. Custom probe groups can also be created by manually selecting probe nets of a different name and adding them into the group.

The Active Probes tab provides the following options for probe points that are added from the Hierarchical View/Netlist View:

- Display bus name. An automatically generated bus name cannot be modified. Only custom bus names can be modified.
- Expand/collapse bus or probe group

- Move Up/Down the signal, bus, or probe group
- Save (Active Probes list)
- Load (already saved Active Probes list)
- Delete (applicable to a single probe point added to the Active Probes tab
- Delete All (deletes all probe points added to the Active Probes tab)
- In addition, the context (right-click) menu provides the following operations:
    - Create Group, Add/Move signals to Group, Remove signals from Group,
    - Ungroup
    - Reverse bit order, Change Radix for a bus or probe group
    - Read, Write, or Delete the signal or bus or probe group

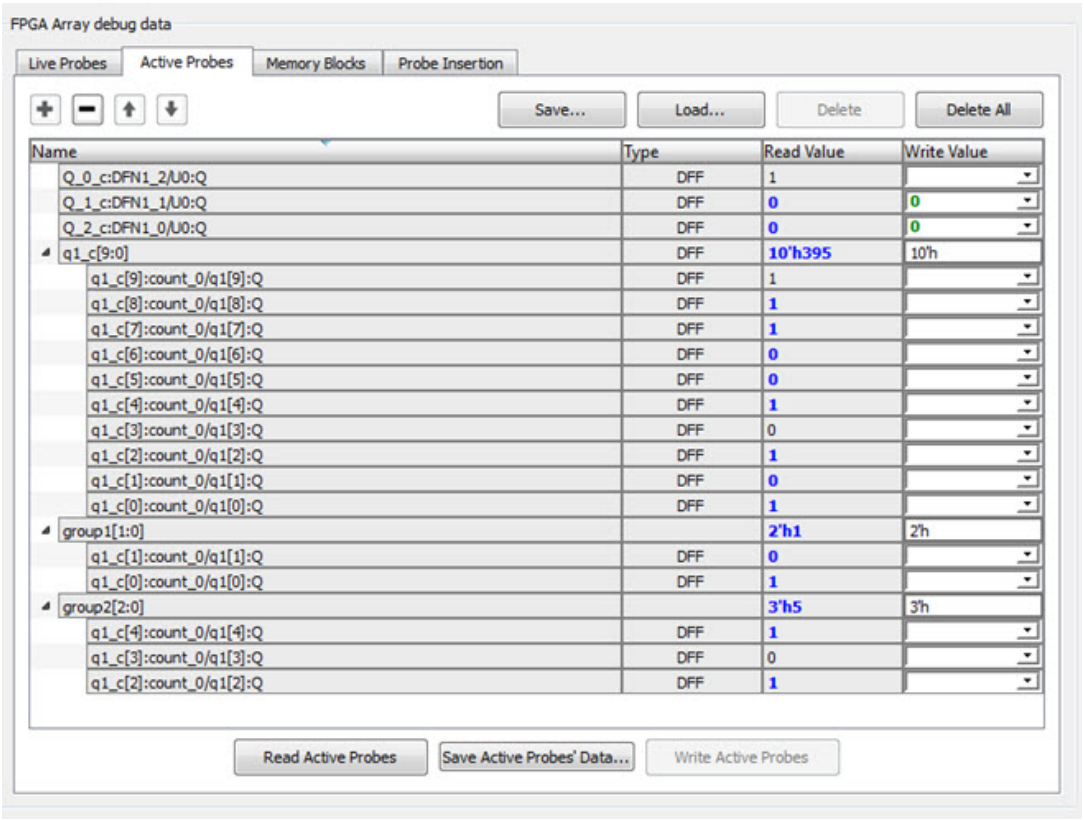


Figure 11 · Active Probes Tab

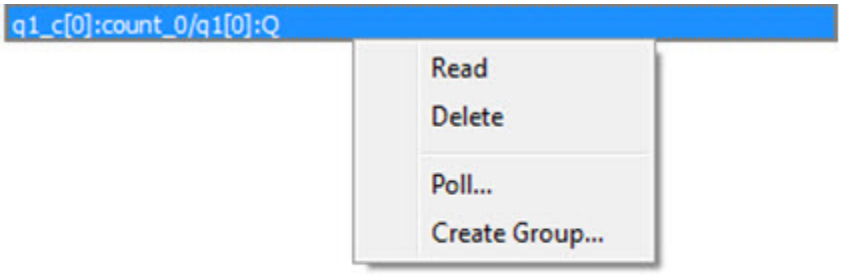
- Green entries in the “Write Value” column indicate that the operation was successful.
- Blue entries in the “Read Value” column indicate values that have changed since the last read.

Context Menu of Probe Points Added to the Active Probes UI

When you right-click a signal or bus, you will see the following menu options:

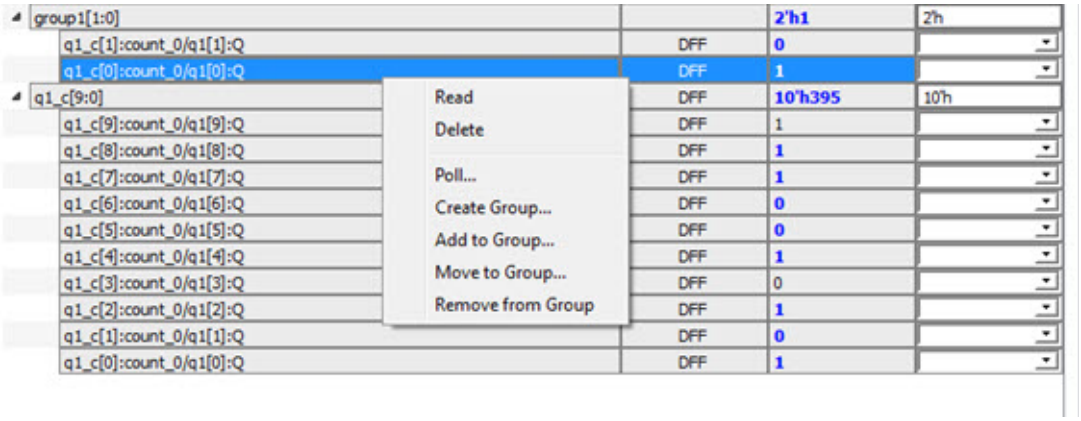
For individual signals that are not part of a probe group or bus:

- Read
- Delete
- Poll
- Create Group



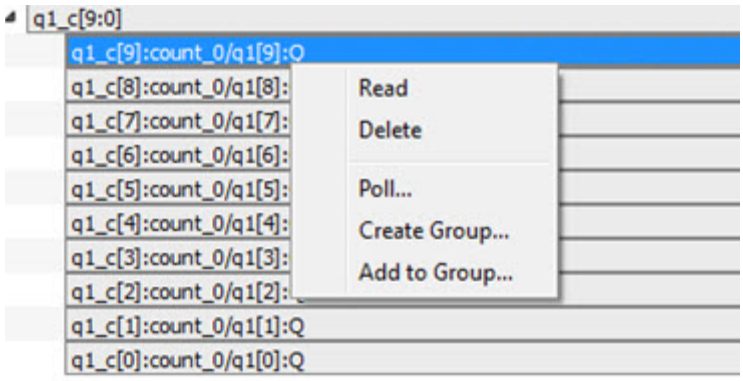
For individual signals in a probe group:

- Read
- Delete
- Poll
- Create Group
- Add to Group
- Move to Group
- Remove from Group



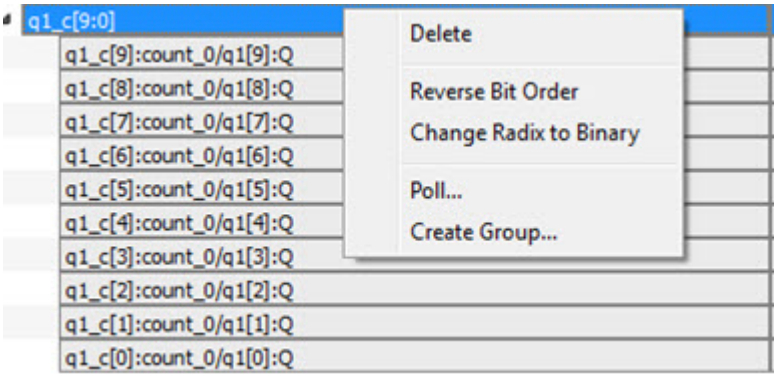
For individual signals in a bus:

- Read
- Delete
- Poll
- Create Group
- Add to Group



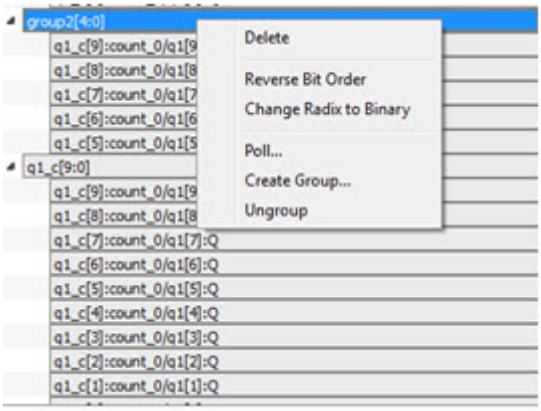
For a bus:

- Delete
- Reverse Bit Order
- Change Radix to Binary
- Poll
- Create Group



For a probe group:

- Delete
- Reverse Bit Order
- Change Radix to Binary
- Poll
- Create Group
- Ungroup



Differences Between a Bus and a Probe Group

A bus is created automatically by grouping selected probe nets with the same name into a bus. A bus *cannot* be ungrouped.

A Probe Group is a custom group created by adding a group of signals in the Active Probes tab into the group. The members of a Probe Group are not associated by their names. A Probe Group *can* be ungrouped.

In addition, certain operations are also restricted to the member of a bus, whereas they are allowed in a probe group.

The following operations are not allowed in a bus:

• **Move to Group**: Moving a signal to a probe group

• **Remove from Group**: Removing a signal from a probe group

Memory Blocks

The Memory Blocks tab in the Debug FPGA Array dialog box shows the hierarchical view of all memory blocks in the design. The depth and width of blocks shown in the logical view are determined by the user in SmartDesign, RTL, or IP cores using memory blocks.

Notes:

- RAM is not accessible to the user when SmartDebug is accessing RAM blocks.
- RAM is not accessible to the user during a read or write operation.
    - During a single location write, the RAM block is not accessible. If multiple locations are written, the RAM block is accessed and released for each write.
    - When each write is completed, access returns to the user, so the access time is a single write operation time.

The example figure that follows shows the hierarchical view of the Memory Blocks tab. You can view logical blocks and physical blocks. Logical blocks are shown with an **L** ( ), and physical blocks are shown with a **P** ( ).

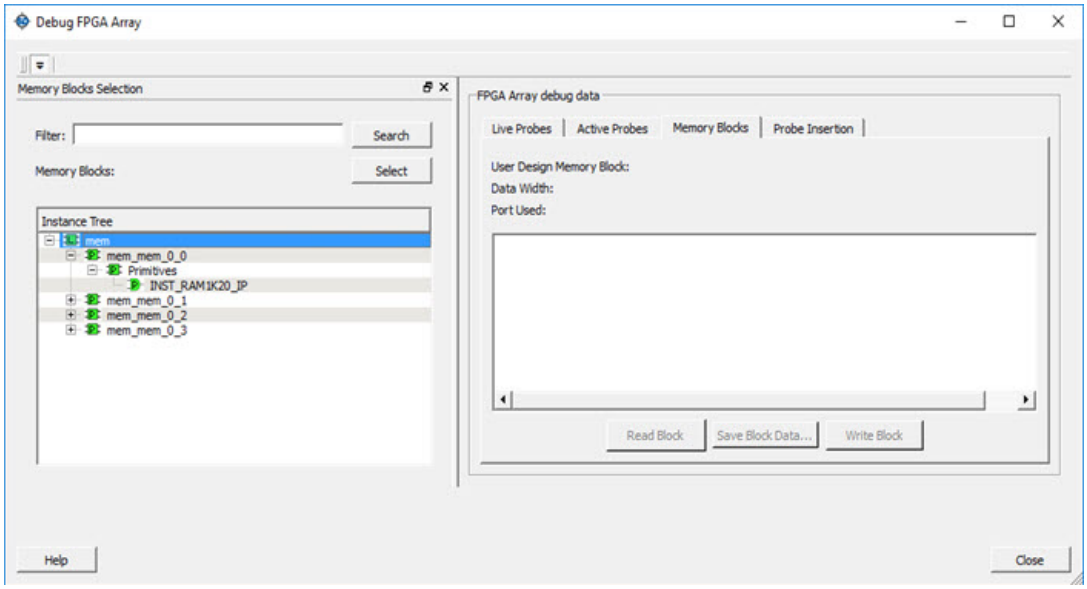


Figure 12 · Memory Blocks Tab - Hierarchical View

You can only select one block at a time. You can select and add blocks in the following ways:

- Right-click the name of a memory block and click **Add** as shown in the following figure.

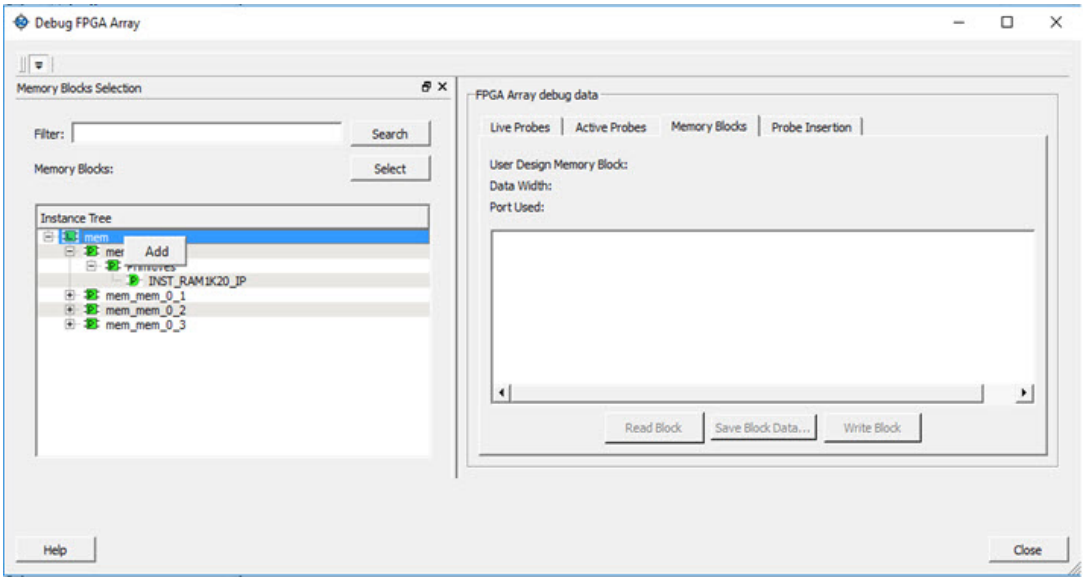


Figure 13 · Adding a Memory Block

- Click on a name in the list and then click **Select**.
- Select a name, drag it to the right, and drop it into the Memory Blocks tab.
- Enter a memory block name in the Filter box and click **Search** or press **Enter**. Wildcard search is supported.

Note: Only memory blocks with an **L** or **P** icon can be selected in the hierarchical view.

Memory Block Fields

The following memory block fields appear in the Memory Blocks tab.

User Design Memory Block

The selected block name appears on the right side. If the block selected is logical, the name from top of the block is shown.

Data Width

If a block is logical, the depth and width is retrieved from each physical block, consolidated, and displayed. If the block is physical, the value of "Depth X Width" is 64 X 12 for uSRAM blocks, 16384 X 1, 8192 X 2, 4096 X 5, 2048 X 10, and 1024 X 20 for LSRAM blocks.

Note: LSRAM physical block configuration of 40-bits is not supported in SmartDebug. This will be available in a future release.

Port Used

This field is displayed only in the logical block view. Because configurators can have asymmetric ports, memory location can have different widths. The port shown can either be Port A or Port B. For TPSRAM, where both ports are used for reading, Port A is used. This field is hidden for physical blocks, as the values shown will be irrespective of read ports.

The following figure shows the Memory Blocks tab fields for a logical block view.

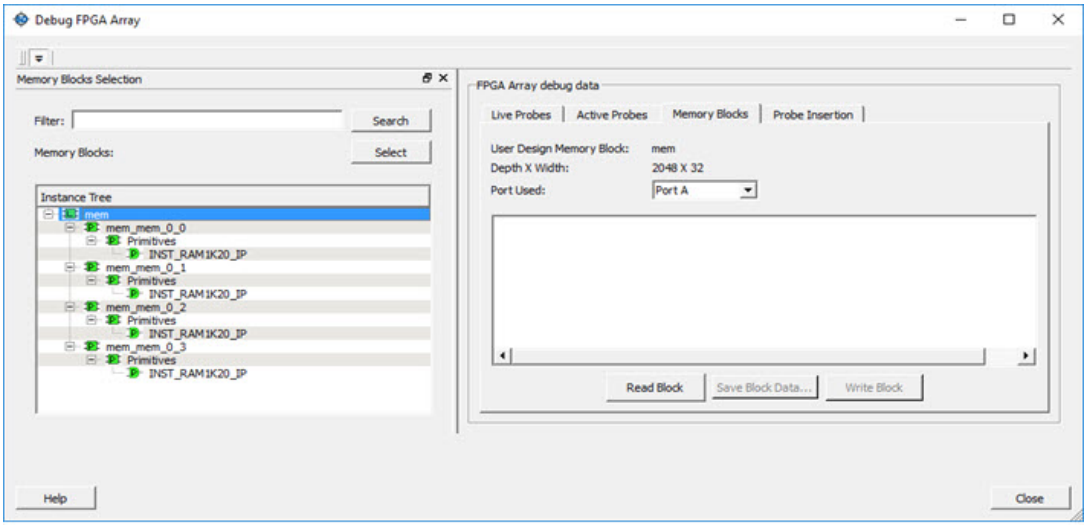


Figure 14 · Memory Blocks Tab Fields for Logical Block View

The following figure shows the Memory Blocks tab fields for a physical block view.

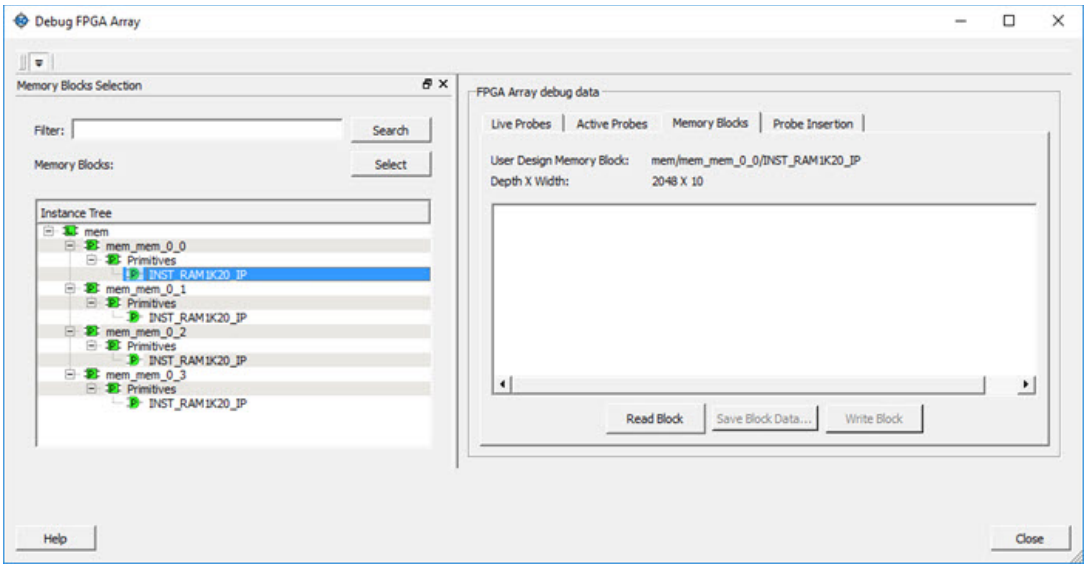


Figure 15 · Memory Blocks Tab Fields for Physical Block View

Read Block

Memory blocks can be read once they are selected. If the block name appears on the right-hand side, the Read Block button is enabled. Click **Read Block** to read the memory block.

Logical Block Read

A logical block shows three fields. User Design Memory Block and Depth X Width are read only fields, and the Port Used field has options. If the design uses both ports, Port A and Port B are shown under options. If only one port is used, only that port is shown.

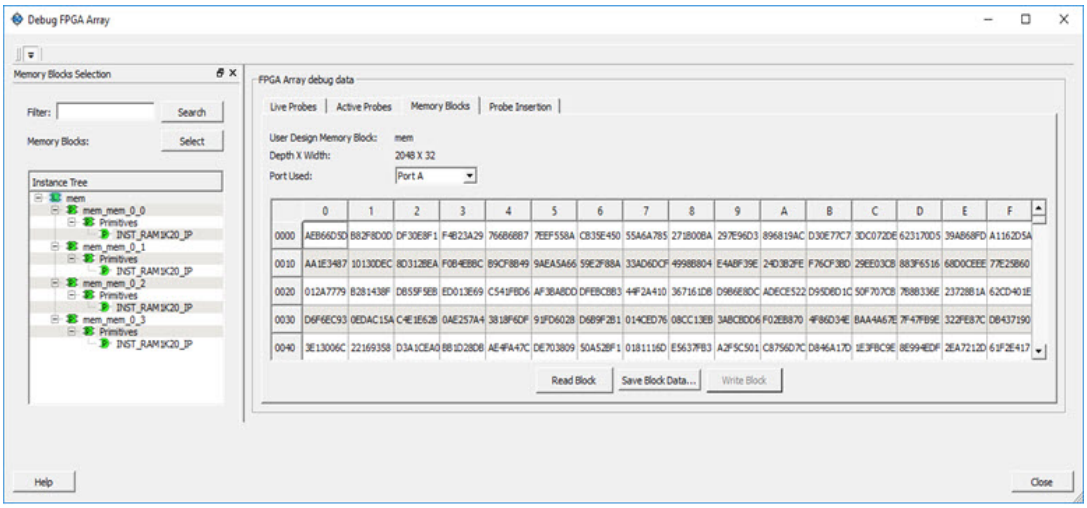


Figure 16 · Logical Block Read

The data shown is in Hexadecimal format. In the example figure above, data width is 32. Because each hexadecimal character has 4 bits of information, you can see 8 characters corresponding to 32 bits. Each row has 16 locations (shown in the column headers) which are numbered in hexadecimal from 0 to F.

Note: For all logical blocks that cannot be inferred from physical blocks, the corresponding icon does not contain a letter.

Physical Block Read

When a Physical block is selected, only the User Design Memory Block and Depth X Width fields are shown.

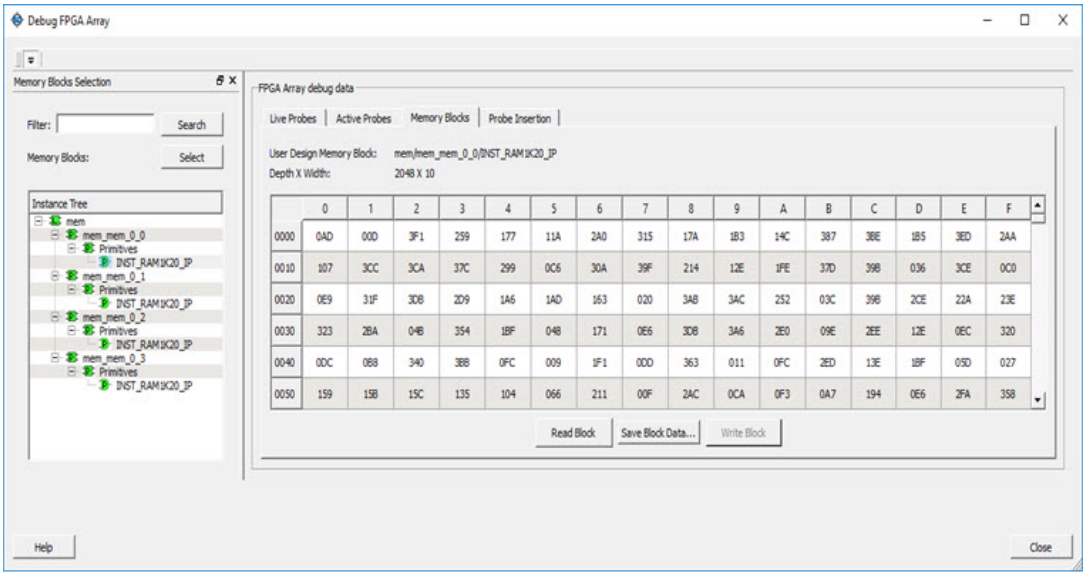


Figure 17 · Physical Block Read

Write Block

Logical Block Write

A memory block write can be done on each location individually. A logical block shows each location of width. The written format is hexadecimal numbers from 0 to F. Width is shown in bits, and values are shown in hexadecimal format. If an entered value exceeds the maximum value, SmartDebug displays a pop-up message showing the range of allowed values.

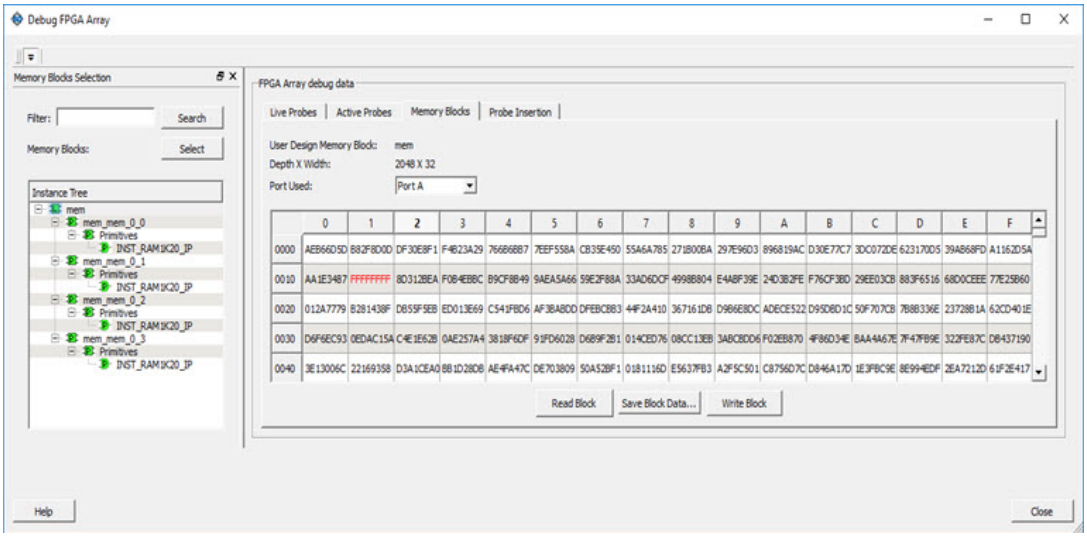


Figure 18 · Logical Block Write

Physical Block Write

Physical blocks have a fixed width of 129 bits for uSRAM and the maximum value that can be written in hexadecimal format is FFF. Similarly, for LSRAM blocks, a range of values are possible (1, 2, 5, 10, and 20) and the maximum values can be 1, 3, 1F, 3FF, and FFFFF, respectively. If an entered value exceeds the limit, SmartDebug displays a popup message showing the range of values that can be entered.

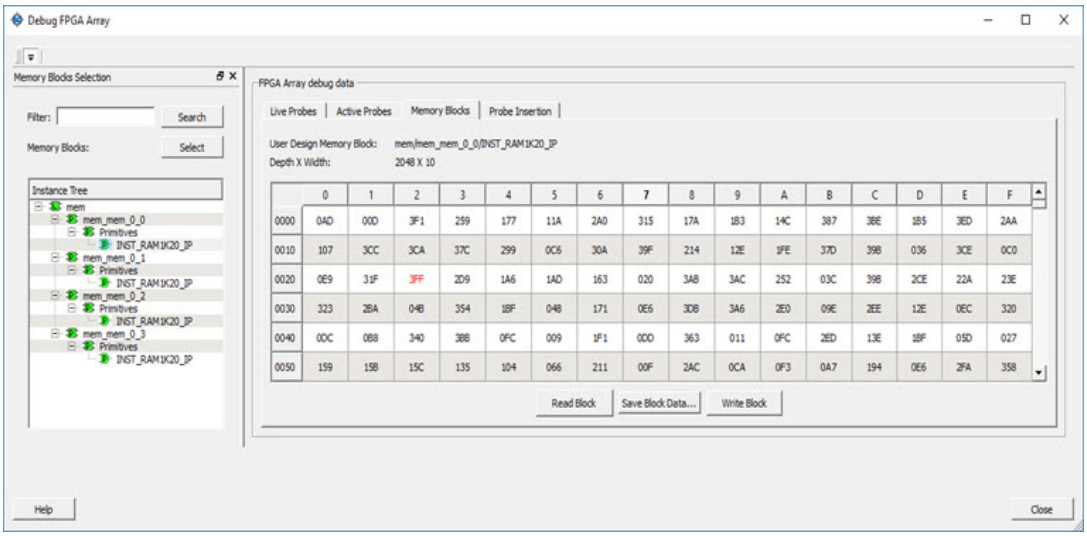


Figure 19 · Physical Block Write

Unsupported Memory Blocks

If RTL is used to configure memory blocks, it is recommended that you follow RAM block inference guidelines provided by Microsemi.

SmartDebug may or may not be able to support logical view for memory blocks that are inferred using RTL coding not specified in the above document.

Memory Blocks in Demo Mode

A temporary memory data file is created in the designer folder for each type of RAM selected. All memory data of all instances of USRAM, LSRAM, and other RAM types is written to their respective data files. The default value of all memory locations is shown as 0s, and is updated based on your changes.

Both physical block view and logical block view are supported.

Probe Insertion (Post-Layout)

Introduction

Probe insertion is a post-layout debug process that enables internal nets in the FPGA design to be routed to unused I/Os. Nets are selected and assigned to probes using the Probe Insertion window in SmartDebug. The rerouted design can then be programmed into the FPGA, where an external logic analyzer or oscilloscope can be used to view the activity of the probed signal.

Note: This feature is not available in standalone mode because of the need to run incremental routing.

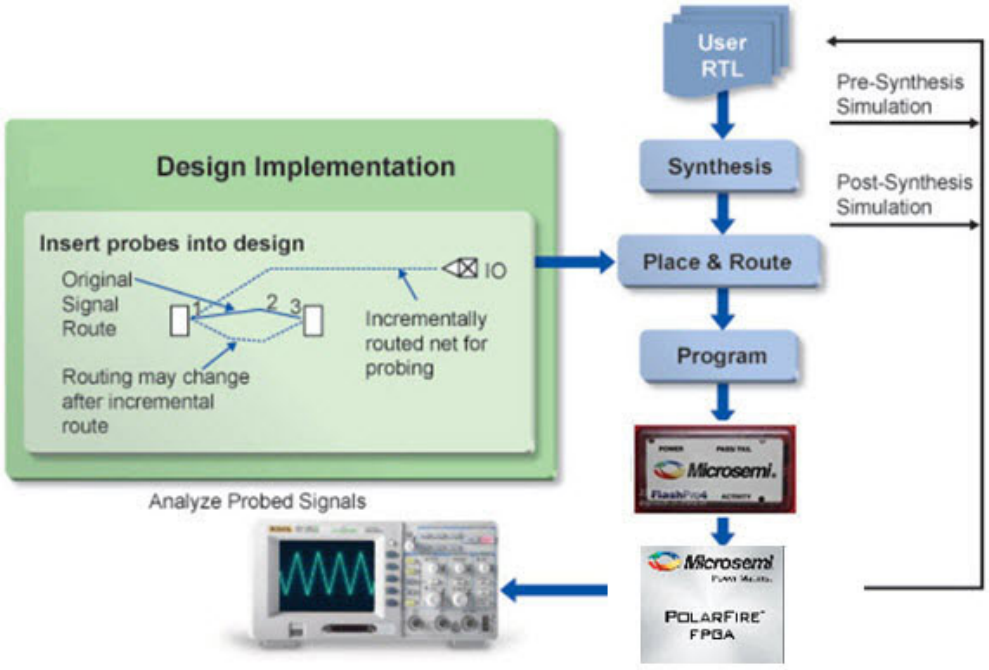


Figure 20 · Probe Insertion in the Design Process

The Probe Insertion debug feature is complementary to Live Probes and Active Probes. Live Probes and Active Probes use a special dedicated probe circuitry.

Probe Insertion

1. Double-click **SmartDebug Design** in the Design Flow window to open the SmartDebug main window.

   Note: FlashPro Programmer must be connected for SmartDebug.

2. Select **Debug FPGA Array** and then select the Probe Insertion tab.

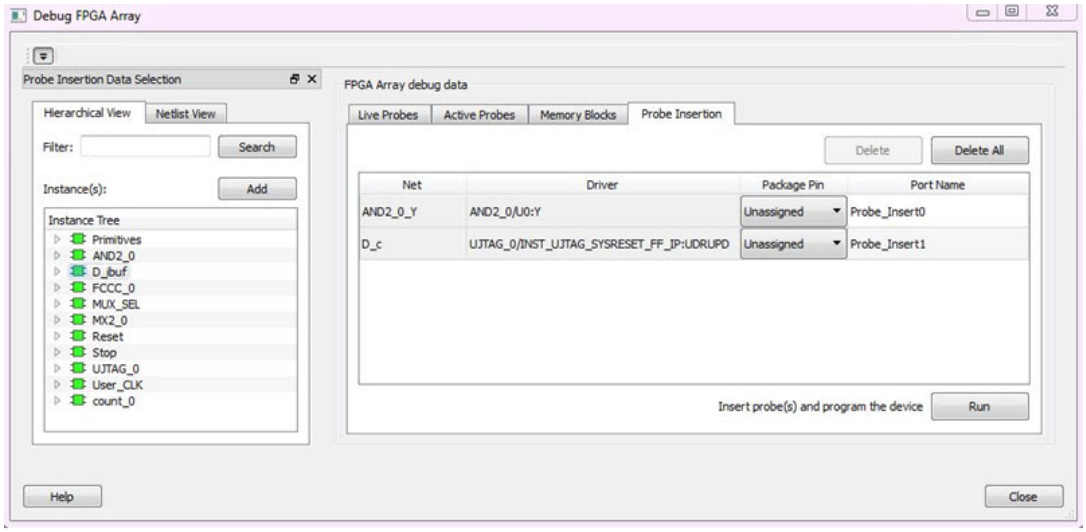


Figure 21 · Probe Insertion Tab

In the left pane of the Probe Insertion tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are shown with the Name and Type in the Netlist View.

3. Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the Filter box.

Each entry has a Net and Driver name which identifies that probe point.

The selected net(s) appear in the Probes table in the Probe Insertion tab, as shown in the figure below. SmartDebug automatically generates the Port Name for the probe. You can change the Port Name from the default if desired.

4. Assign a package pin to the probe using the drop-down list in the Package Pin column. You can assign the probe to any unused package pin (spare I/O).

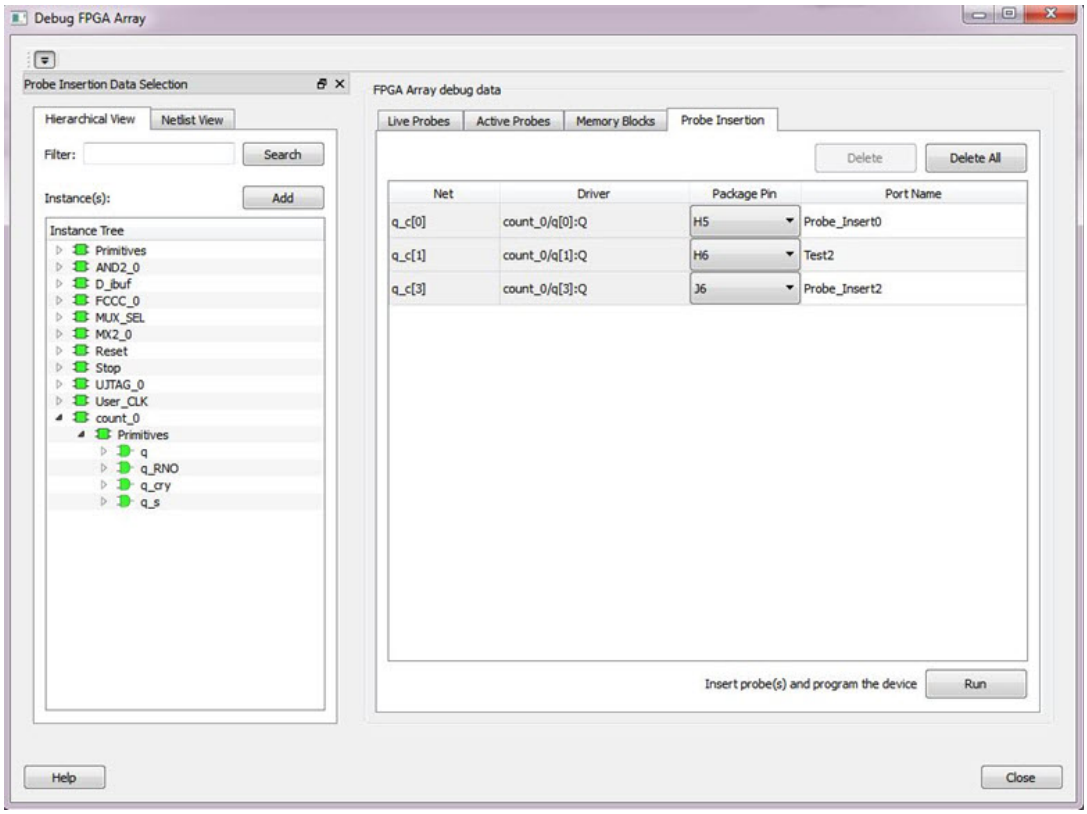


Figure 22 · Debug FPGA Array > Probe Insertion > Add Probe

5. Click **Run**.

This triggers Place and Route in incremental mode, and the selected probe nets are routed to the selected package pin. After incremental Place and Route, Libero automatically reprograms the device with the added probes.

The log window shows the status of the Probe Insertion run.

Probe Deletion

To delete a probe, select the probe and click **Delete**. To delete all probes, click **Delete All**.

Note: Deleting probes from the probes list without clicking **Run** does not automatically remove the probes from the design.

Reverting to the Original Design

To revert to the original design after you have finished debugging:

1. In SmartDebug, click **Delete All** to delete all probes.
2. Click **Run**.
3. Wait until the action has completed by monitoring the activity indicator (spinning blue circle). Action is completed when the activity indicator disappears.
4. Close SmartDebug.

Event Counter

The Event Counter counts the signals that are assigned to Channel A through the Live Probe feature. This feature can track events from the board. When the Event Counter is activated, and a signal is assigned to Channel A, the counter starts counting the rising edge transitions. The counter must be stopped to get the final signal transition count. During the count, you cannot assign another signal to Channel A/Channel B or go to any other tab on the window.

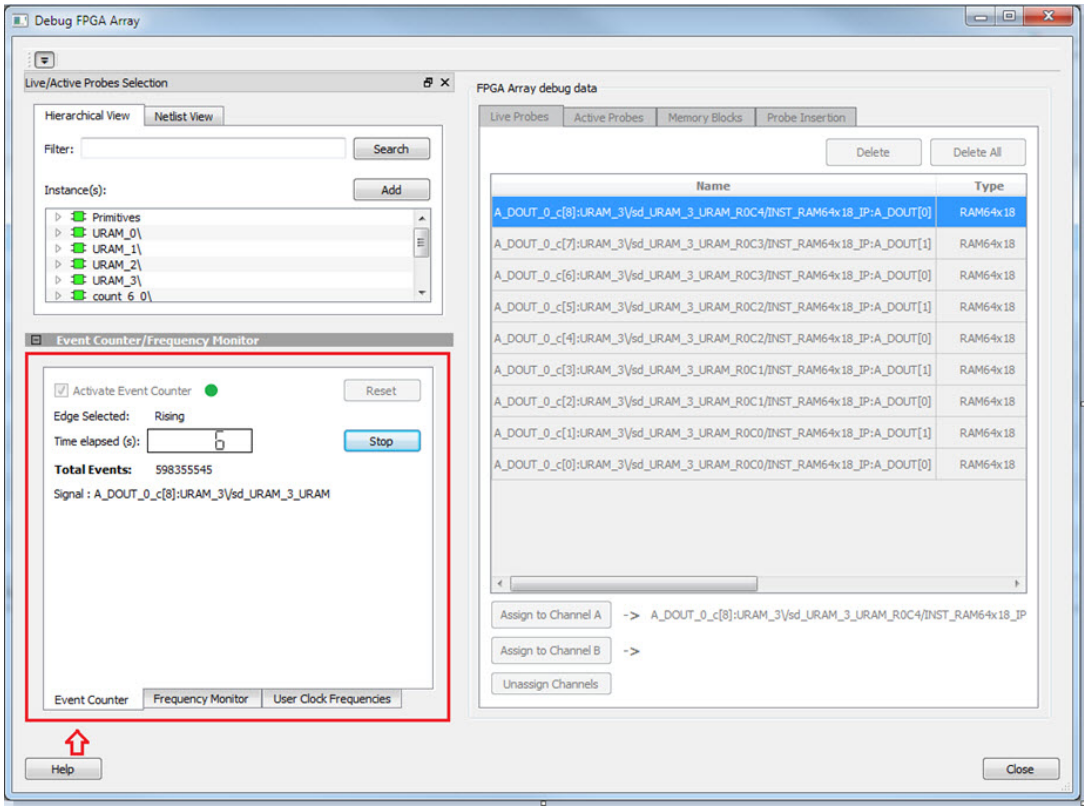


Figure 23 · Event Counter Tab/UI

Activating the Event Counter

You can activate the Event Counter in either of the following two ways:

- Click **Activate Event Counter** and then assign a signal to Live Probe Channel A.

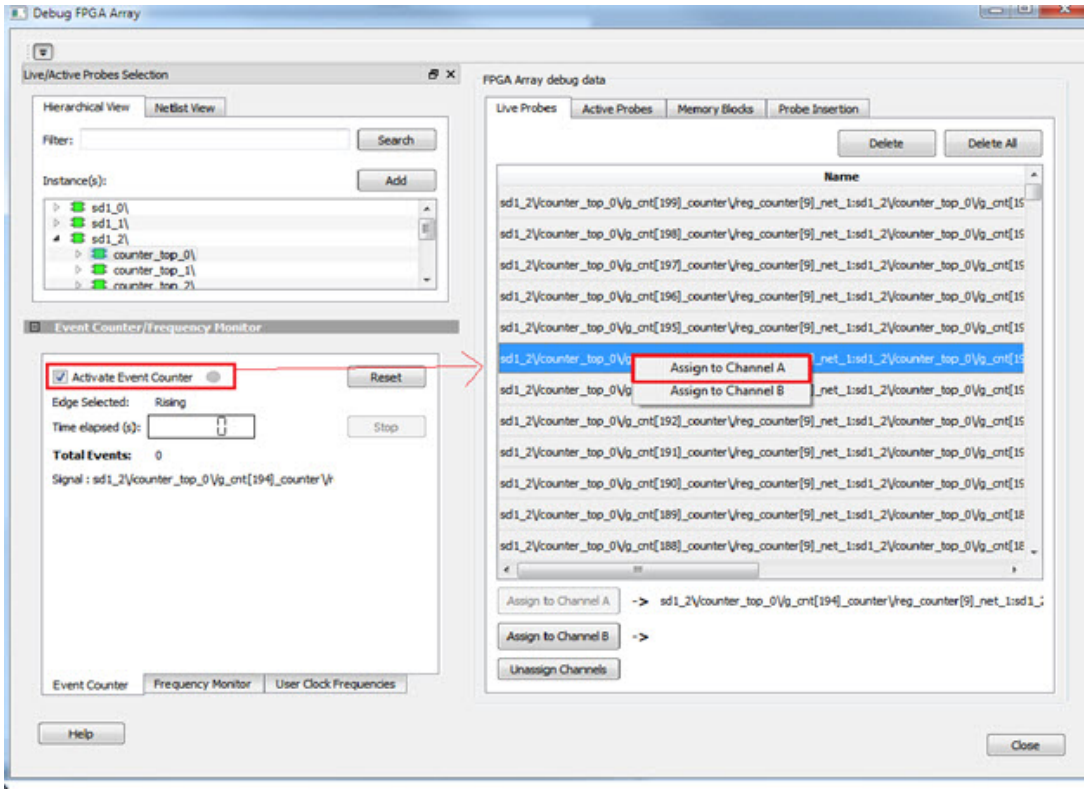


Figure 24 · Activating the Event Counter

- Assign a signal to Probe Channel A and then click **Activate Event Counter**.

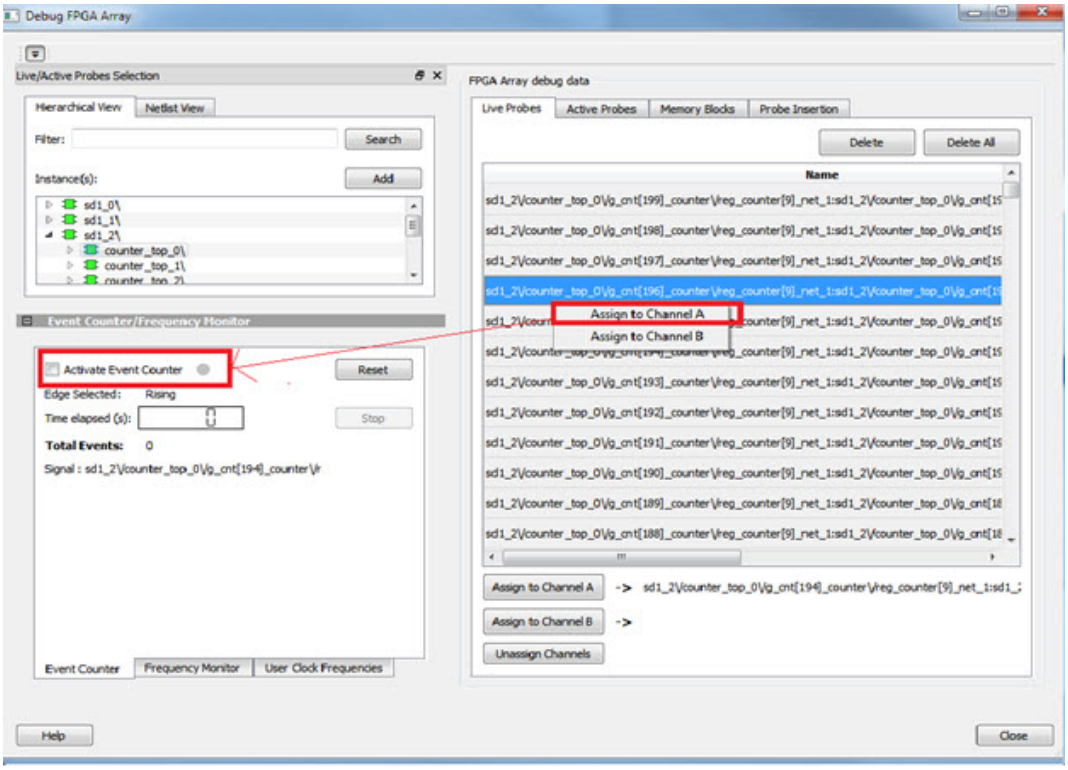


Figure 25 · Activating the Event Counter - Assign Probe Channel

Running the Event Counter

Event Counter automatically runs the counter, which is indicated by a green LED. The counts are updated every second, and are shown next to Total Events. FPGA Array debug data and the control tabs in the Event Counter panel are disabled while Event Counter is running. When a signal is assigned, the signal name appears next to Signal.

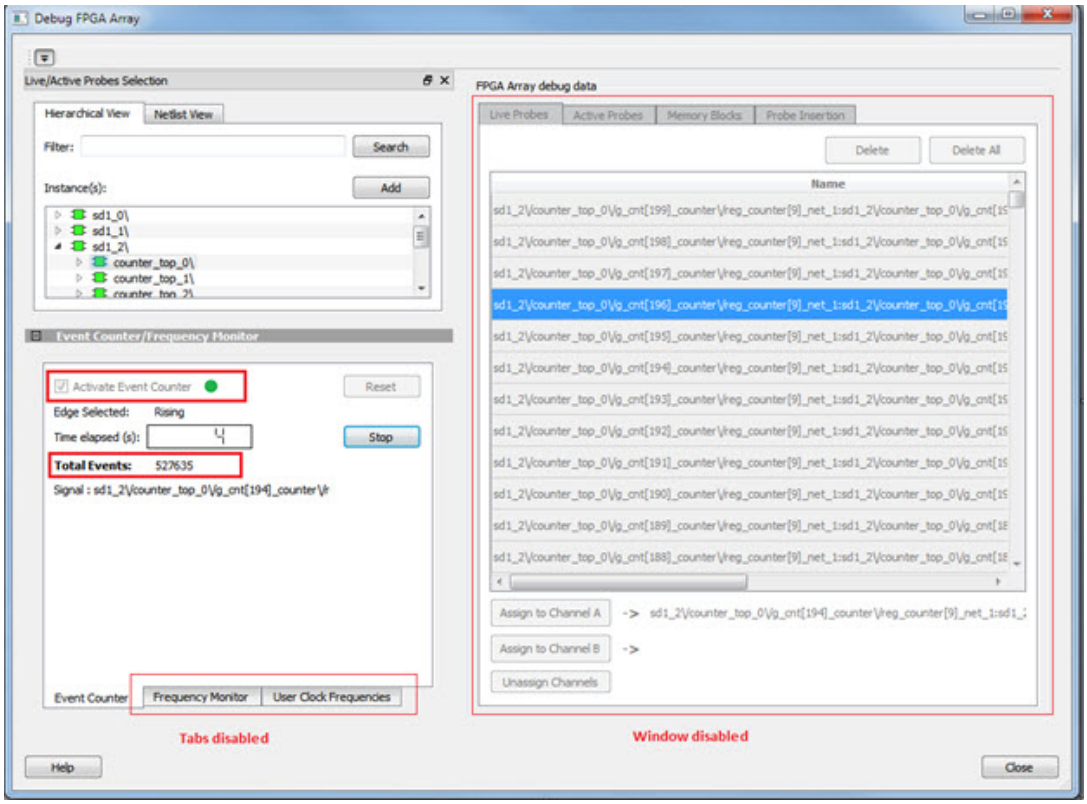


Figure 26 · Running the Event Counter

Stopping the Event Counter

The only button enabled when Event Counter is running is the “Stop” button. Click button to stop counting. A red LED is shown to indicates the Event Counter has stopped. FPGA Array debug data and the control tabs in the Event Counter panel are enabled when Event Counter is not running.

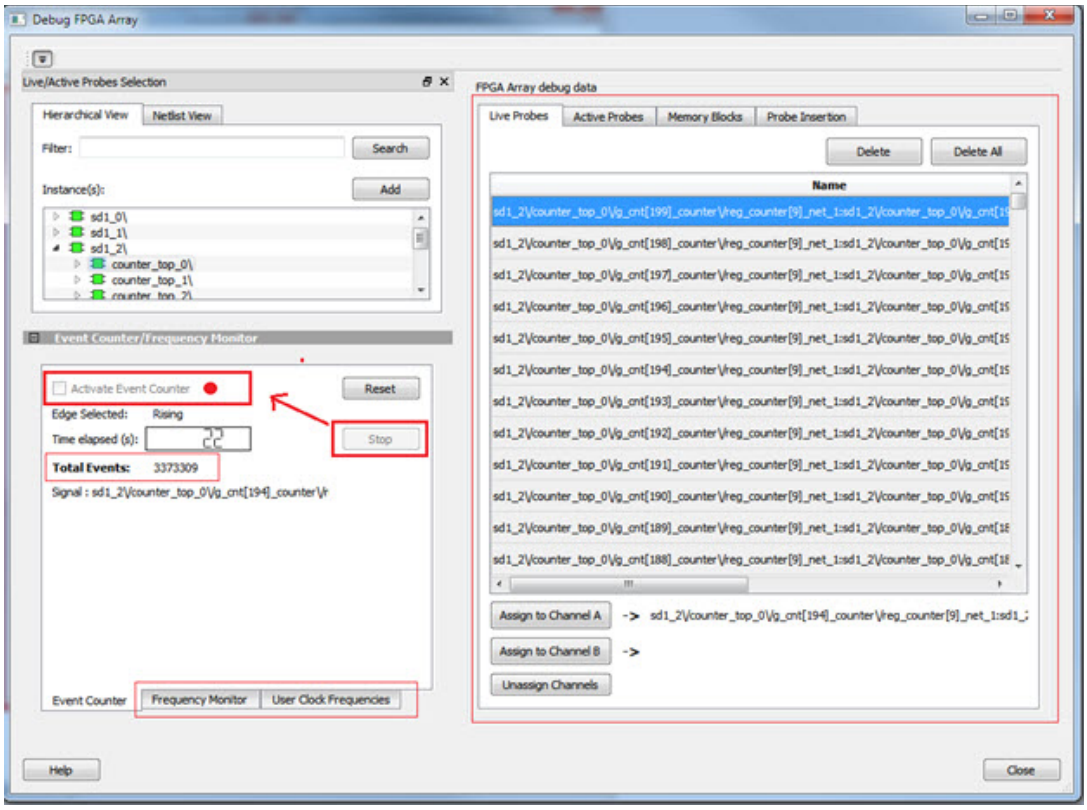


Figure 27 · Stopping the Event Counter

Note: When a DC signal (signal tied to logic ‘0’) is assigned to Live Probe Channel A, or if there are no transitions on the signal assigned to Live Probe Channel A with initial state ‘0’, the Event Counter value is updated as ‘1’ when the counter is stopped. This is a limitation of the FHB IP, and will be fixed in upcoming releases.

See Also

"Frequency Monitor" on page 35

"User Clock Frequencies" on page 47

Frequency Monitor

The Frequency Monitor calculates the frequency of any signal in the design that can be assigned to Live Probe channel A. The Frequency Monitor must be activated before or after the signal is assigned to Live Probe Channel A. You can enter the time to monitor the signal. The accuracy of results increases as the monitor time increases. The unit of measurement is displayed in Megahertz (MHz). During the run, progress is displayed in the pane.

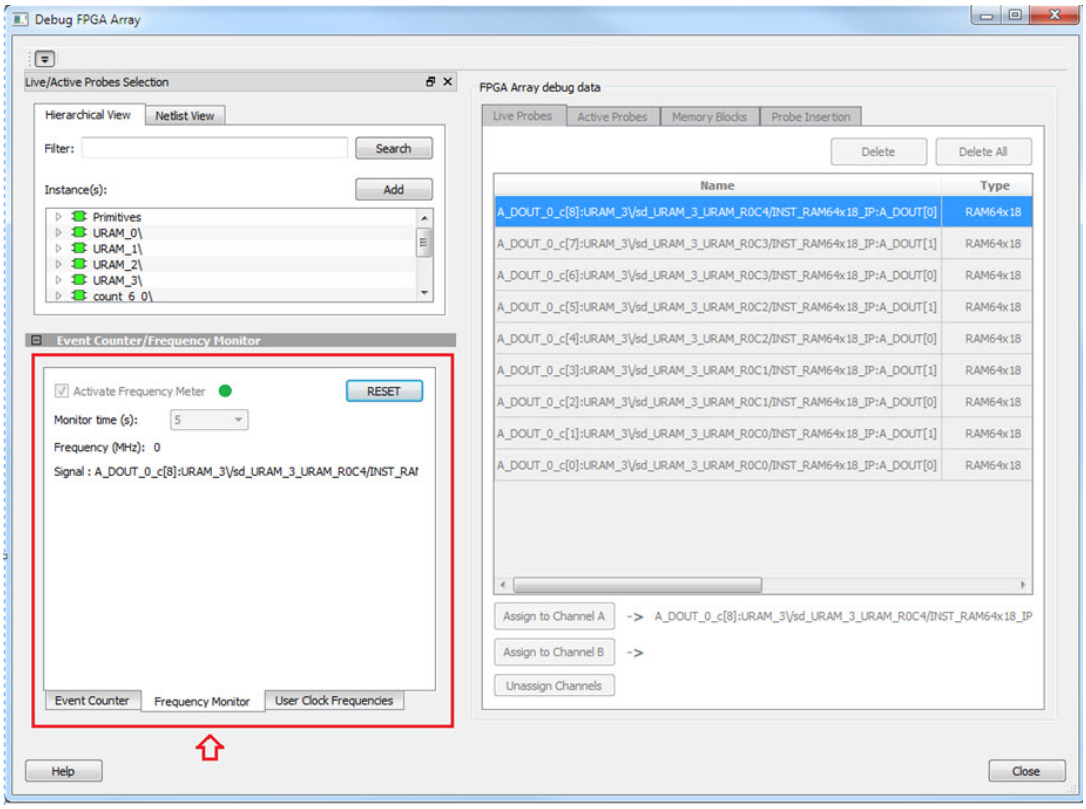


Figure 28 · Frequency Monitor Tab/UI

In the Frequency Monitor tab, you can activate the Frequency Monitor, change the monitor time (delay to calculate frequency), reset the monitor, and set the frequency in megahertz (MHz). Click the drop-down list to select monitor time value. During the frequency calculation, all tabs on the right side of the window are disabled, as well as the tabs in the FPGA Hardware Breakpoint (FHB) pane.

Activating the Frequency Monitor

You can activate the Frequency Monitor in either of the following two ways:

- Click **Activate Frequency Monitor**, and then click the **Live Probe** tab and assign a signal to Channel A (Channel B is not configured for spatial debug operations).

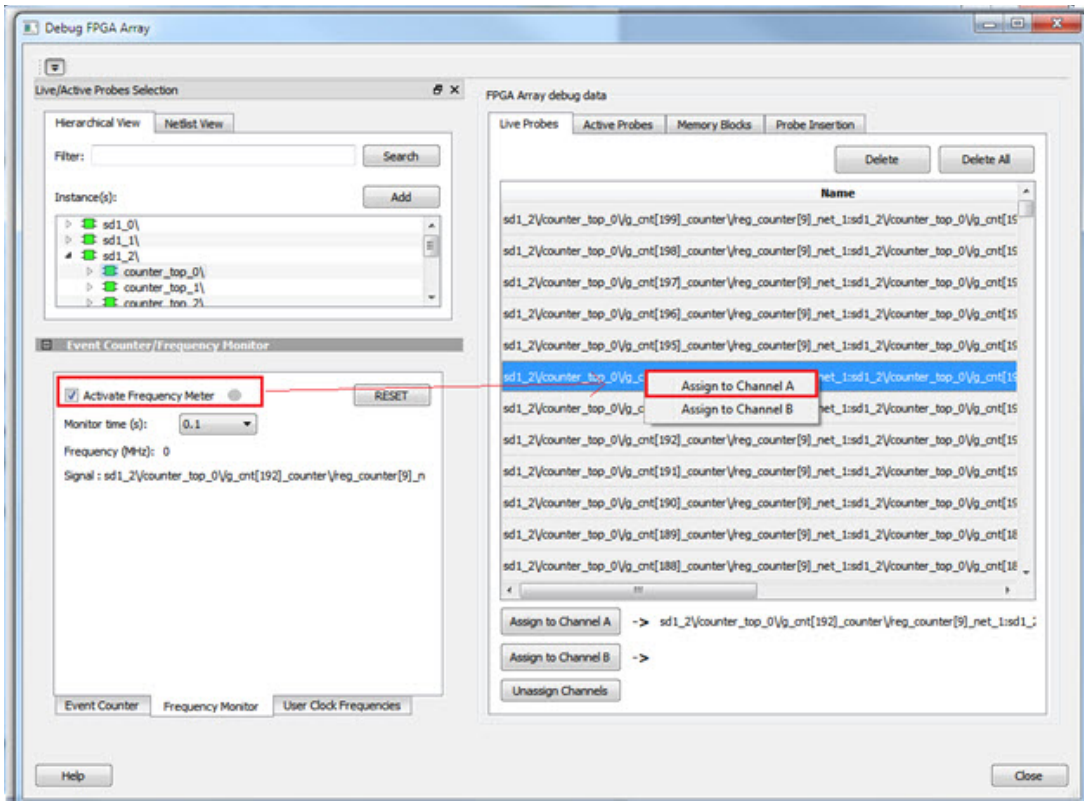


Figure 29 · Activating the Frequency Monitor - Assign a Signal

- Click the **Live Probe** tab and assign a signal to Channel A, and then click the **Frequency Monitor** tab and check the Activate Frequency Monitor checkbox.

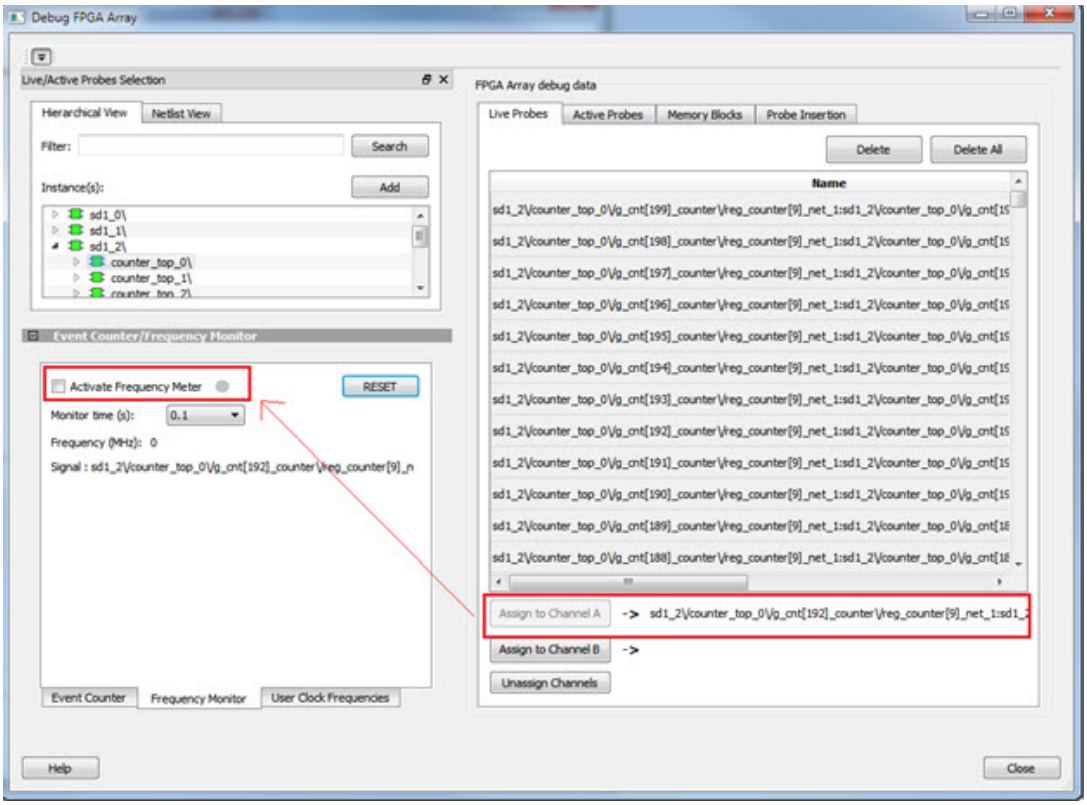


Figure 30 · Activating the Frequency Monitor

Running the Frequency Monitor

The Frequency Monitor runs automatically, and is indicated by a green LED. While it is running, FPGA Array debug data and the control tabs in the panel are disabled. A progress bar shows the monitor time progress when it is 1 second and above (as shown in the following figure). The Reset button is also disabled during the run. When a signal is assigned, the signal name appears next to Signal.

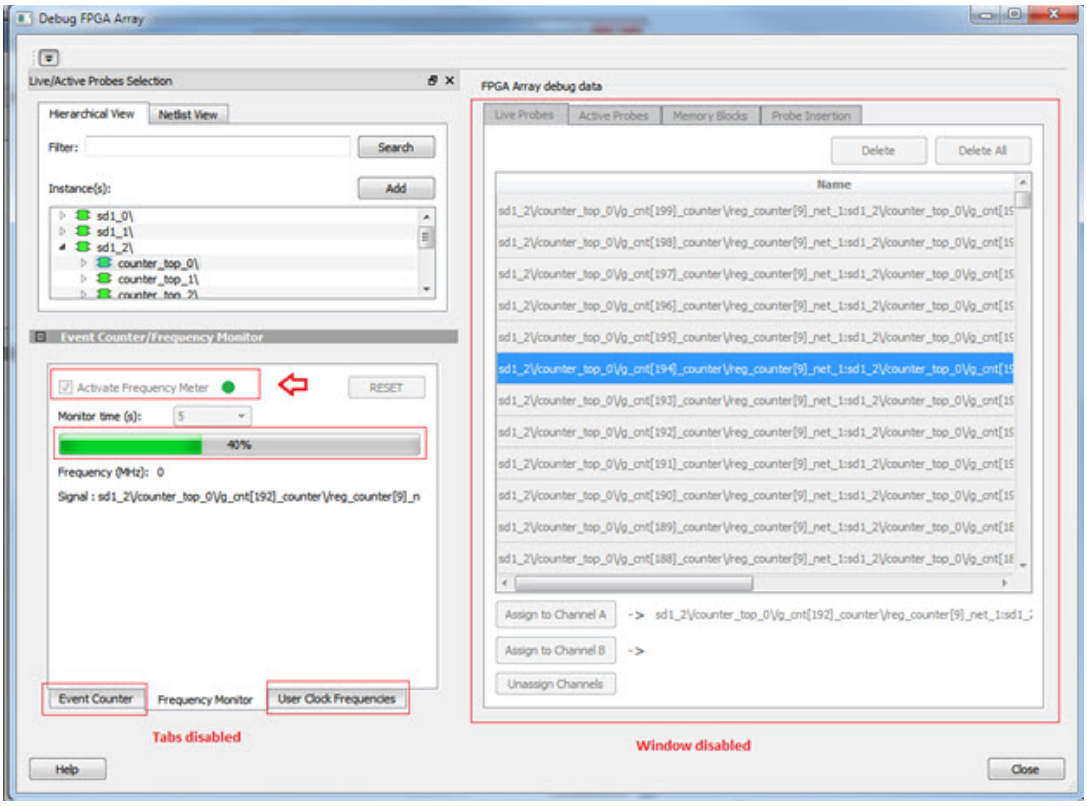


Figure 31 · Running the Frequency Monitor

Stopping the Frequency Monitor

The Frequency Monitor stops when the specified monitor time has elapsed. This is indicated by a red LED. The result appears next to Frequency. The window and the tabs on the control panel are enabled. The Reset button is also enabled to reset the Frequency to 0 to start over the next iteration. The progress bar is hidden when the Frequency Monitor stops.

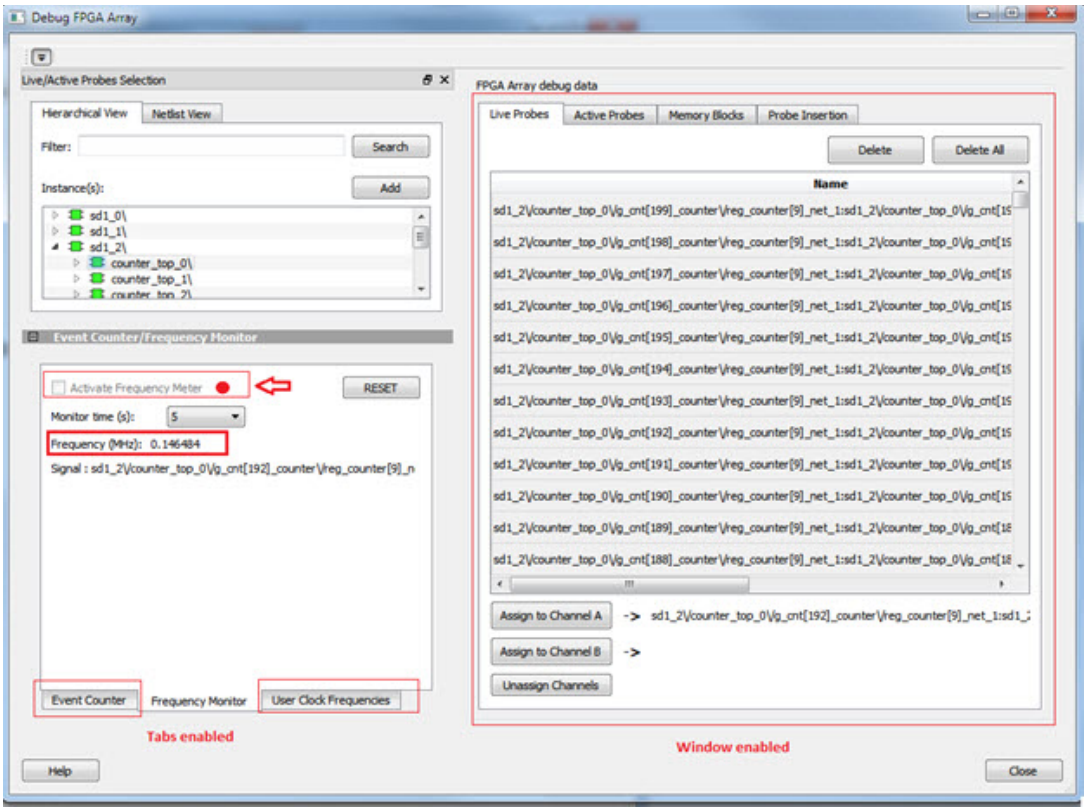


Figure 32 · Stopping the Frequency Monitor

See Also

"Event Counter" on page 32

"User Clock Frequencies" on page 47

FPGA Hardware Breakpoint Auto Instantiation

The FPGA Hardware Breakpoint (FHB) Auto Instantiation feature automatically instantiates an FHB instance per clock domain that is using gated clocks (GL0/GL1/GL2/GL3) from an FCCC instance. The FHB instances gate the clock domain they are instantiated on. These instances can be used to force halt the design or halt the design through a live probe signal. Once a selected clock domain or all clock domains are halted, you can play or step on the clock domains, either selectively or all at once. The FPGA Hardware Breakpoint controls in the SmartDebug UI allow you to control the debugging cycle.

To enable this option, select the Enable FHB Auto Instantiation check box in the Design flow tab of the Project Settings dialog box (**Libero > File > Project Settings**). See the example figure that follows.

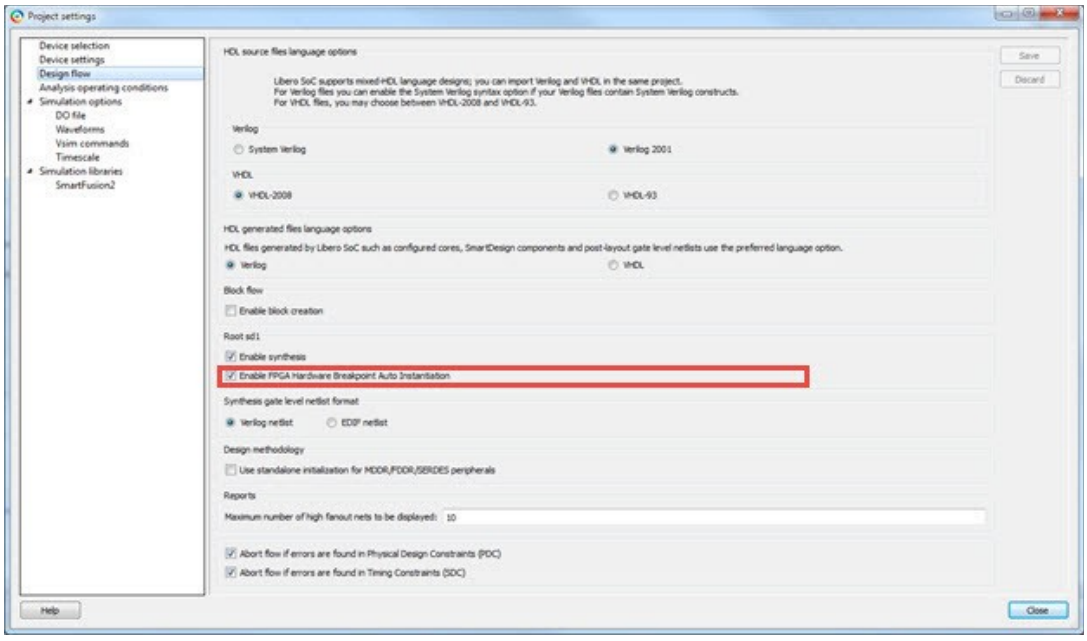


Figure 33 · Enable FHB Auto Instantiation in Project Settings Dialog Box: Design flow Tab

FPGA Hardware Breakpoint (FHB) controls appear in the Debug FPGA Array dialog box when there is an auto-instantiated FHB instance in the design. See the example figure that follows.

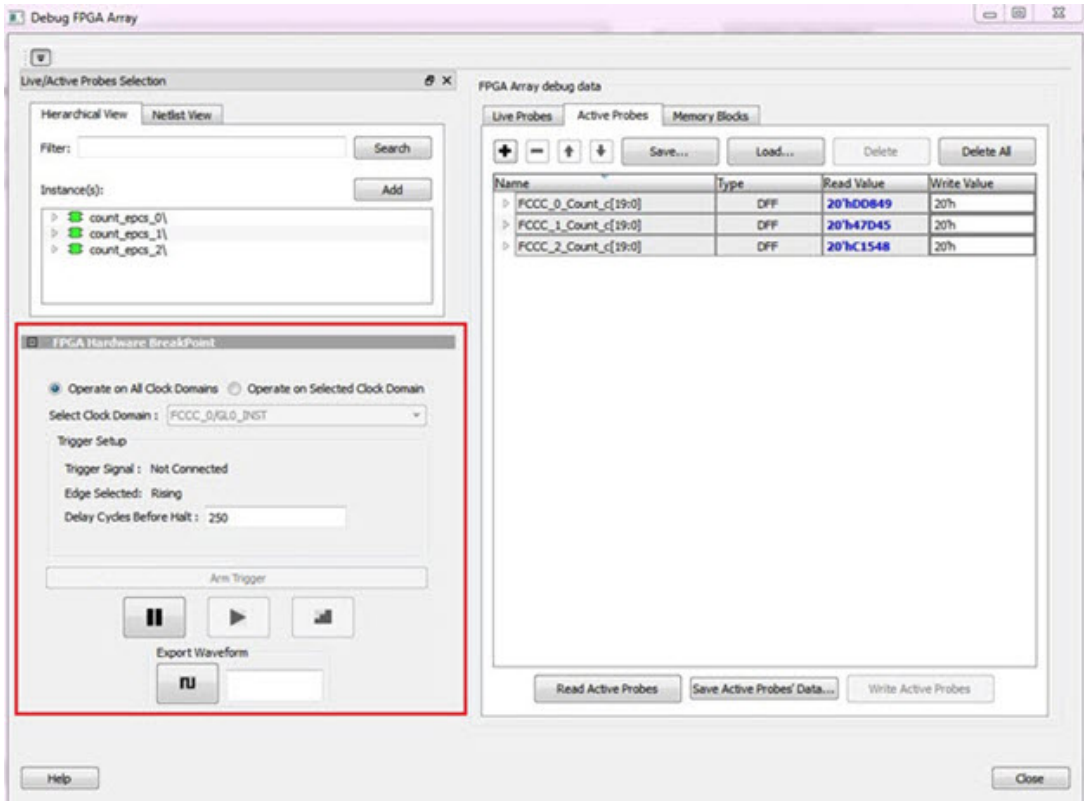


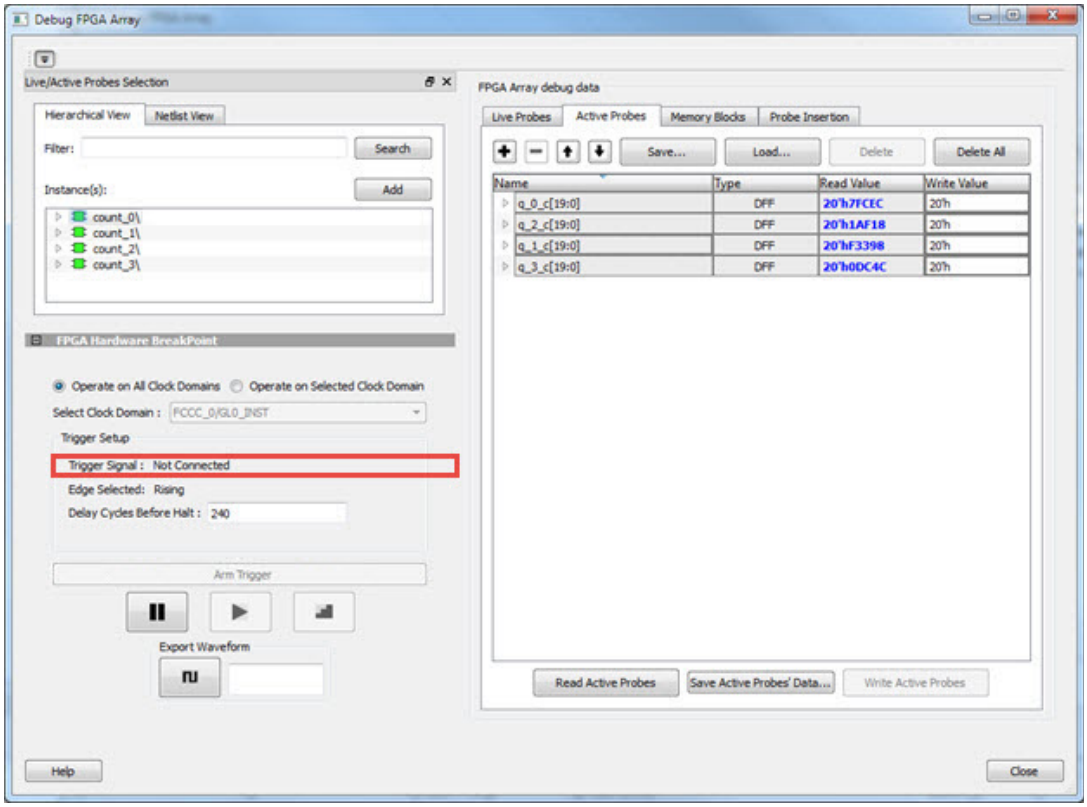
Figure 34 · FPGA Hardware Breakpoint (FHB) Controls

You can choose **Operate on All Clock Domains** or **Operate on Selected Clock Domain** by selecting the appropriate radio button. Selecting either of these modes sets the FHB instances to the respective mode. Once you assign the Live Probe PROBE_A connection and click **Arm Trigger**, the DUT halts on the next positive edge that occurs on the signal connected to Live Probe PROBE_A.

When you choose **Operate on Selected Clock Domain** mode, the Select Clock Domain combo box is enabled, and all available clock domains are listed. The Halt (Pause) , Play , and Step  buttons are associated for that clock domain. If you switch between clock domains in this mode, previous clock domain settings are not retained.

When you choose **Operate on All Clock Domains** mode, the Select Clock Domain combo box is disabled. The Halt, Play, and Step buttons are associated for all clock domains.

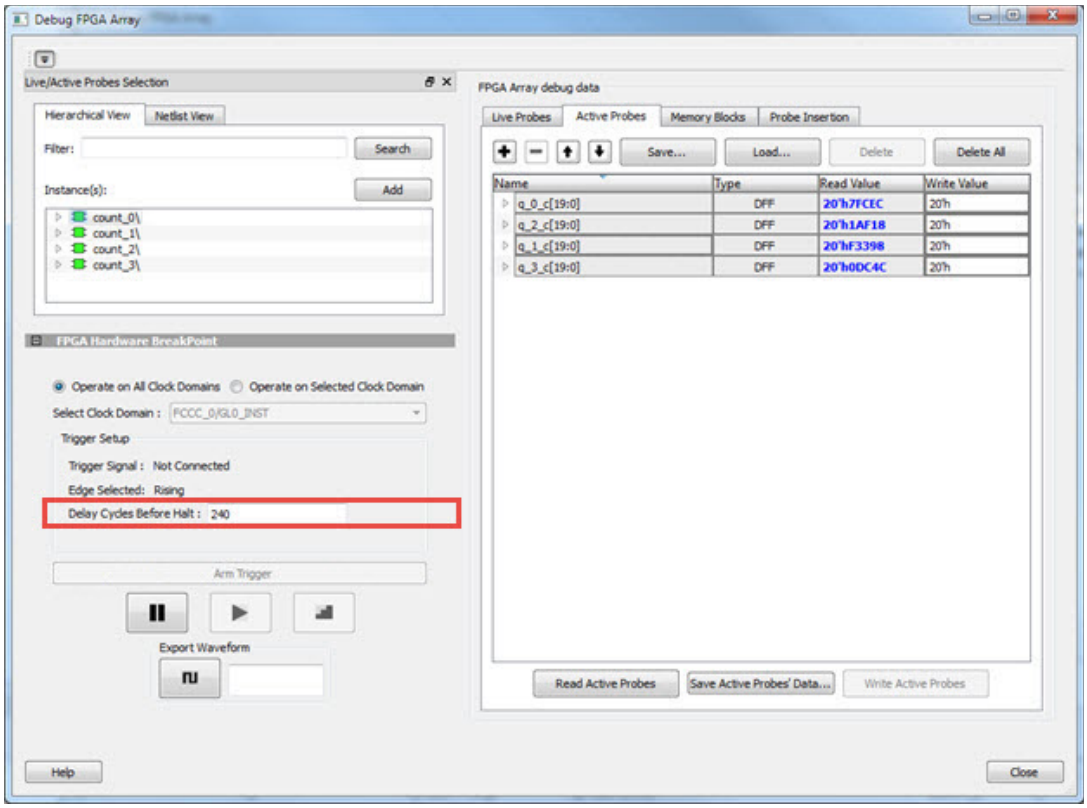
The Trigger Signal is shown as Not Connected until a live probe is assigned. See the example figure that follows.



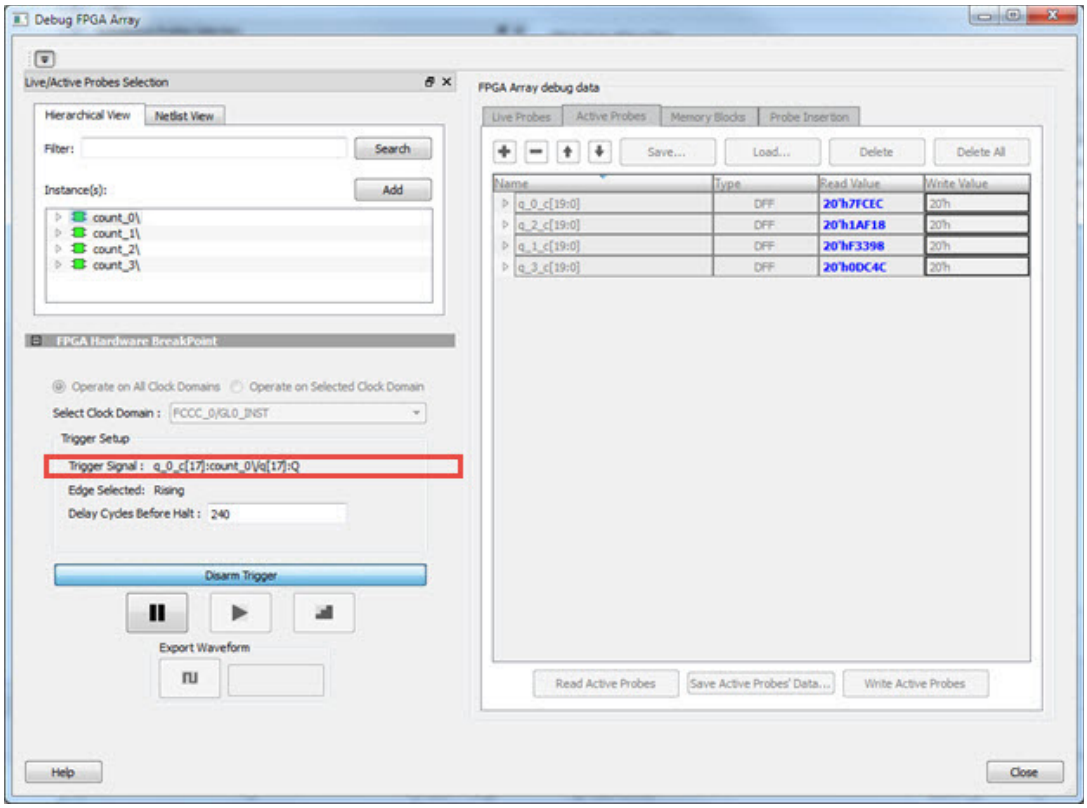
When a probe is assigned to Live Probe PROBE_A, the Trigger Signal updates.

If you require a certain number of clock cycles before halting the clock domain after triggering, a value between 0 and 255 must be entered for Delay Cycles Before Halt before you click **Arm Trigger**. This sets the FHBs to trigger after the specified delay from the rising edge trigger.

Delay is not applied to a forced Halt. See the example figure that follows.



When a live probe connection is made and you click **Arm Trigger**, FPGA Hardware Breakpoint functionality is disabled until the trigger is disarmed automatically or the design is force halted.

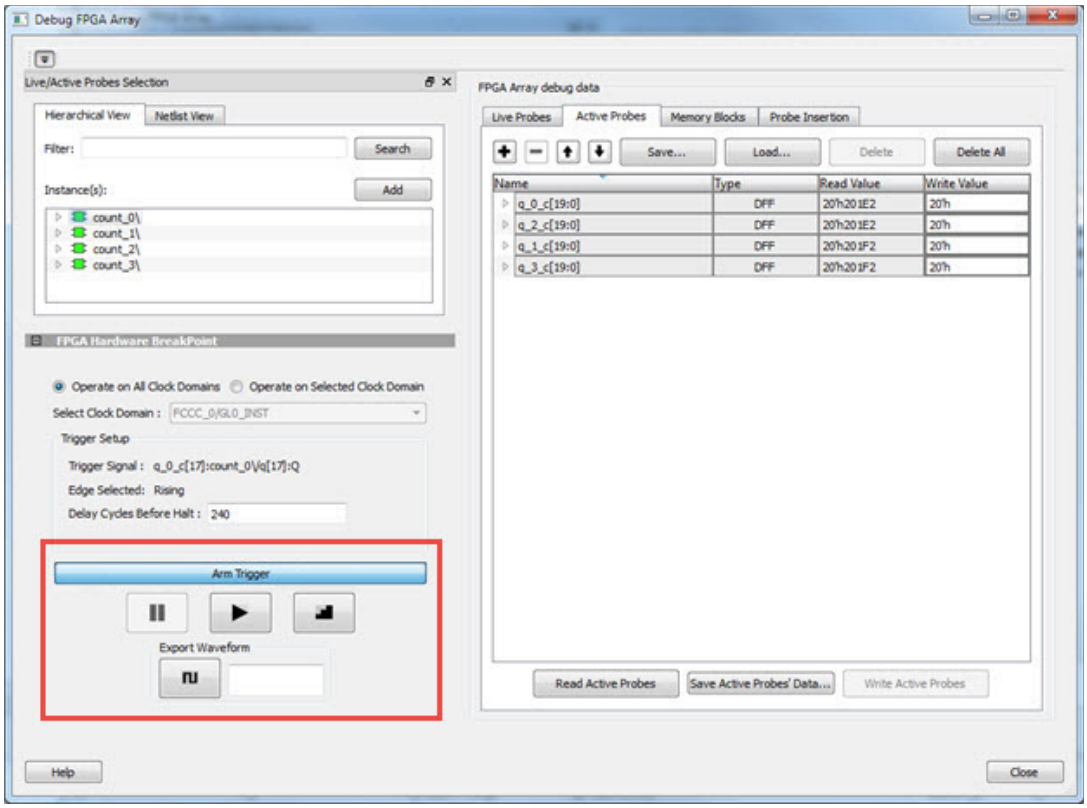


Trigger Input

You can use the trigger input signal if you want an event in the DUT to trigger the FHB IP (for example, a particular state in the FSM or counter value, and so on) when this signal is asserted. If the trigger signal is already asserted (or HIGH) at the time of arming the FHB, the DUT is halted immediately.

Force Halt/Play/Step is done using the FPGA Hardware Breakpoint controls (see the example figure that follows). Once the clock domain is halted, you can either force Play the clock domain or Step the clock domain by 1 clock cycle.

You can save the waveform view of the selected active probes using Export Waveform by specifying the number of clock cycles to capture. The waveform is saved to a .vcd file.



FPGA Hardware Breakpoint Operations

Live Probe Halt

You can halt a selected clock domain or all clock domains in Live Probe Halt mode based on the mode selection (**Operate on All Clock Domains** or **Operate on Selected Clock Domain**).

Assign a signal to Live Probe PROBE_A in the **Live Probes** tab of the UI, and then click the **Active Probe** tab to see the FPGA Hardware Breakpoint controls.

Click **Arm Trigger** to arm the FHBs to look for a trigger on the signal connected to Live Probe PROBE_A.

Once the trigger occurs, the clock domains are halted.

Note: If only one clock domain is halted, other clock domains continue to run, and you should anticipate results accordingly.

Note: Live Probe Halt can be delayed for a maximum of 255 clock cycles.

The actual delay realized on hardware is calculated by the following equation:

```
Actual delay cycles on hardware =
#Delay clock cycles before halt mentioned in smartdebug * (DUT clock frequency/FHB clock frequency)
```

FHB clock frequency is device specific:

PolarFire: 160MHz

See Assumptions and Limitations for more information.

Force Halt

You can force halt a selected clock domain or all clock domains based on mode selection without having to wait for a trigger from a live probe signal. Click the **Halt** button in the FPGA Hardware Breakpoint (FHB) controls.

In **Operate on Selected Clock Domain** mode, the state of the Halt button is updated based on the state of the clock domain selected.

In **Operate on all Clock Domains** mode, the Halt button is disabled only when all clock domains are halted. Each clock domain is halted sequentially in the order shown in the Select Clock Domain combo box.

Note: If only one clock domain is halted, other clock domains continue to run, and you should anticipate results accordingly.

Play

Once the clock domain is in a halted state (live probe halt or force halt), you can click **Play** in the FPGA Hardware Breakpoint controls. This resumes the clock domain from the halted state.

In **Operate on all Clock Domains** mode, each clock domain runs sequentially in the order shown in the Select Clock Domain combo box.

Step

Once the clock domain is in a halted state (live probe halt or force halt), you can click the **Step** button in the FPGA Hardware Breakpoint controls. This advances the clock domain by one clock cycle and holds the state of the clock domain.

In **Operate on All Clock Domains** mode, each clock domain steps sequentially in the order shown in the Select Clock Domain combo box.

Waveform Capture

You can save the waveform view of the selected active probes using Export Waveform by specifying the number of clock cycles to capture in text box and then clicking **Capture Waveform** . The waveform is saved to a .vcd file.

You can view the waveforms by importing the .vcd file. The waveform file can be viewed in any waveform viewer that supports vcd format.

Assumptions and Limitations

- If you select the auto instantiation option in Libero, you need to rerun Synthesis (if already run) to get the FHB related functionality.
- Supported for FCC driven gated clocks (GL0/GL1/GL2/GL3) only.
- CLKINT_PRESERVE – FHB is not auto-instantiated if the user design contains this macro.
- Designs that have Encrypted IPs are not supported.
- EDIF using constraints flow is not supported.
- Live Probe triggering occurs on the Positive Edge only.
- For imported verilog netlist files (.vm files), you must rerun synthesis to get FHB-related functionality. If synthesis is disabled and the netlist is compiled directly, FHB functionality is not inferred.
- If only one clock domain is halted during operations, other clock domains continue to run, and you should anticipate results accordingly.
- FHB performance can only be characterized against the clock which it is running at (i.e. 160MHz).
    - If the DUT clock is running at or less than 160MHz, the DUT clock will halt within one clock cycle (1 or less).
    - For frequencies higher than 160MHz, the point at which the DUT halts cannot be guaranteed.

User Clock Frequencies

The User Clock Frequencies tab shows the frequencies that have been configured from the FCCC block. If assigned, live probe channels are temporarily unassigned, and reassigned after user clock frequencies have been calculated. The Refresh button recalculates frequencies if clocks have been changed.

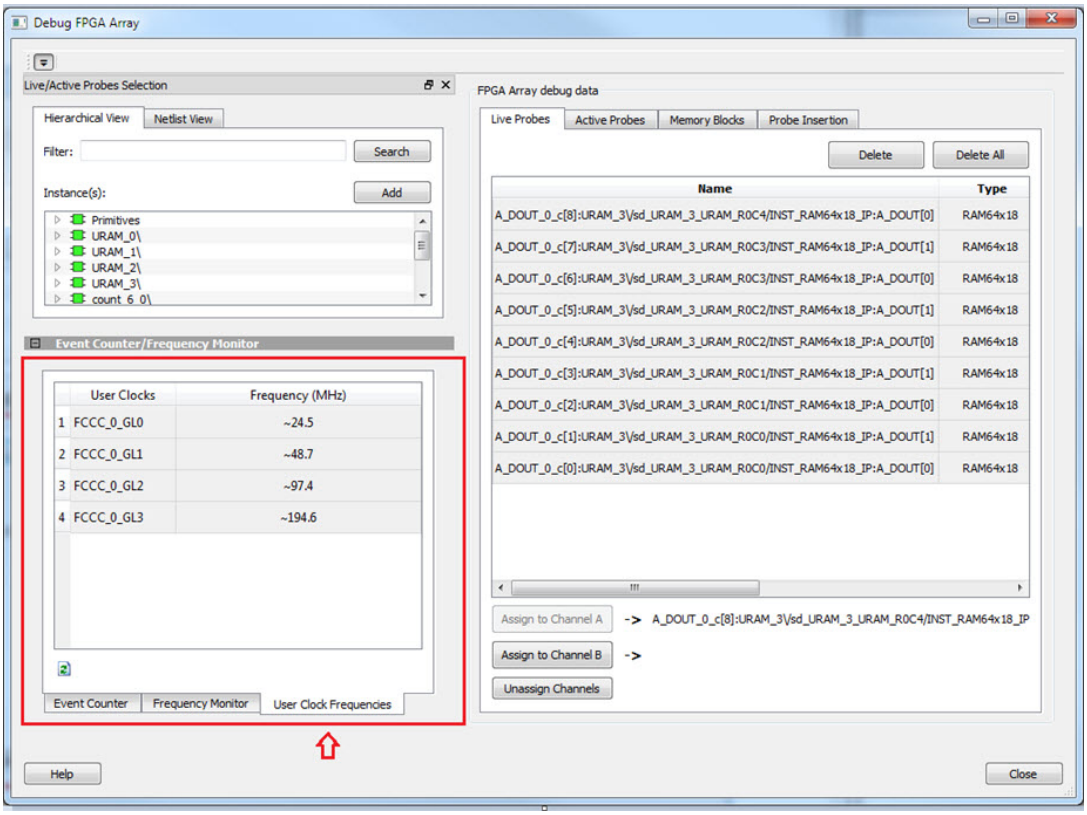


Figure 35 · User Clock Frequencies Tab/UI

See Also

"Event Counter" on page 32

"Frequency Monitor" on page 35

Debug sNVM

The sNVM block stores User data and UIC data. This data is stored as clients and can be configured in the Libero design. The USK (User Secret Key) security key secures pages within the memory. Authenticated data can be plain text or encrypted text, and non-authenticated data is plain text. SmartDebug helps the user read the page content of the sNVM block.

The sNVM Debug window has two tabs – Client View and Page View.

Client View

When you open the sNVM window, two tabs are visible. Client information appears in the Client View tab when it is configured in the Libero design. Select a client to expand the table and see pages and page status inside the client. Click the **Read From Device** button to view the memory content.

You can select only one client at a time. Pages inside the client cannot be selected.

Start Page, End Page, and Number of Bytes are displayed for the selected client.

Click the **View All Page Status** button to see information for all pages in the client. See the following example figures.

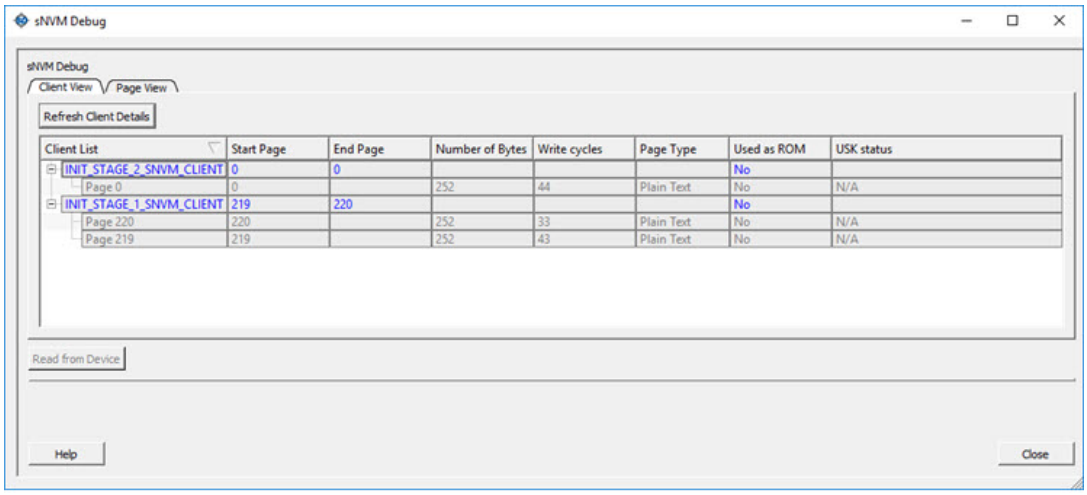


Figure 36 · Client View - expanded list

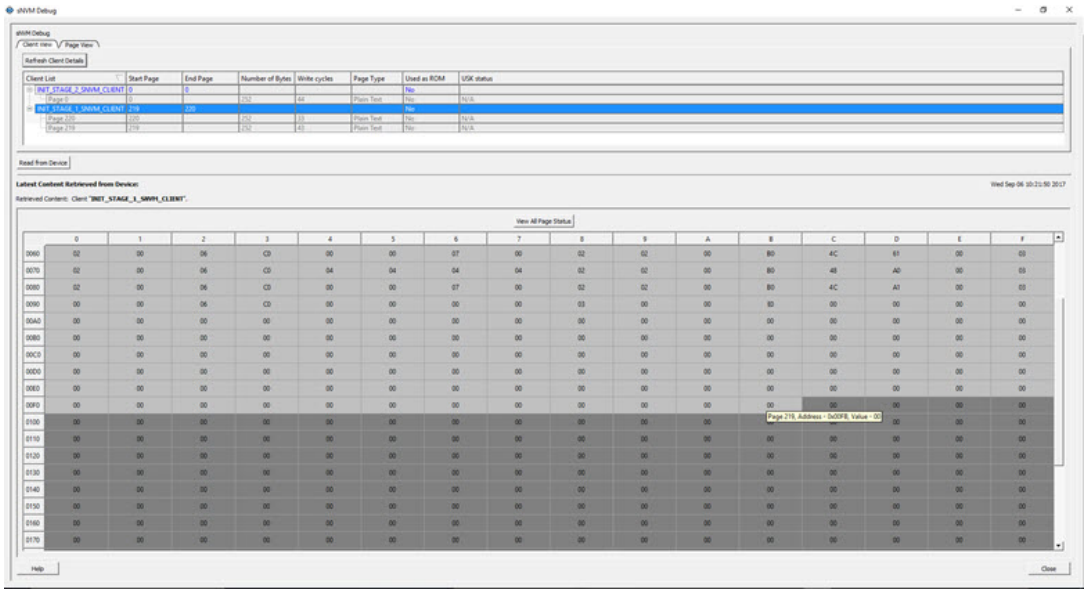


Figure 37 · Client View - Memory

Page View

Page View is used to read a range of pages where start and end page have been specified.

If a page is secured, the default USK is used by SmartDebug to get the page status. If successful, the USK automatically reads the page. If a different USK has been set using system services, use the option to enter the USK, as shown in the example figure below.

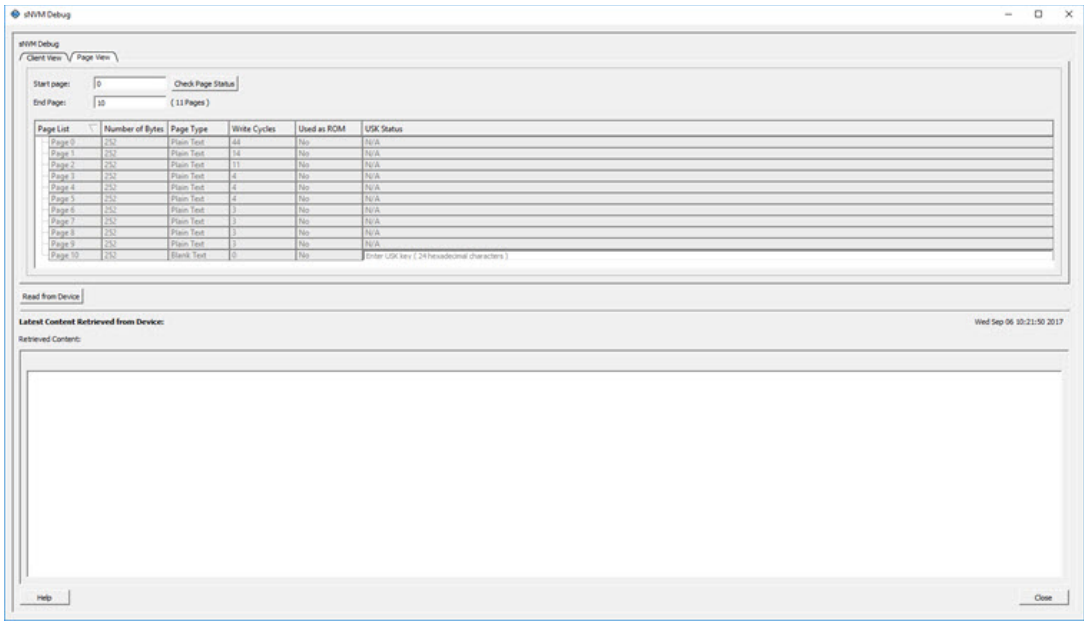


Figure 38 · Page View - Enter USK highlighted

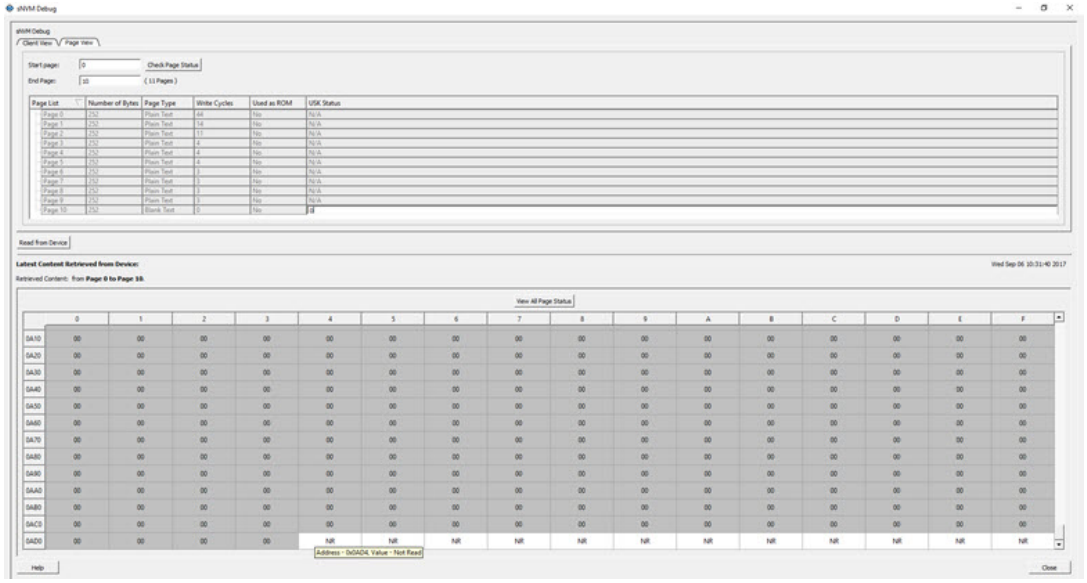


Figure 39 · Memory View - Page Range

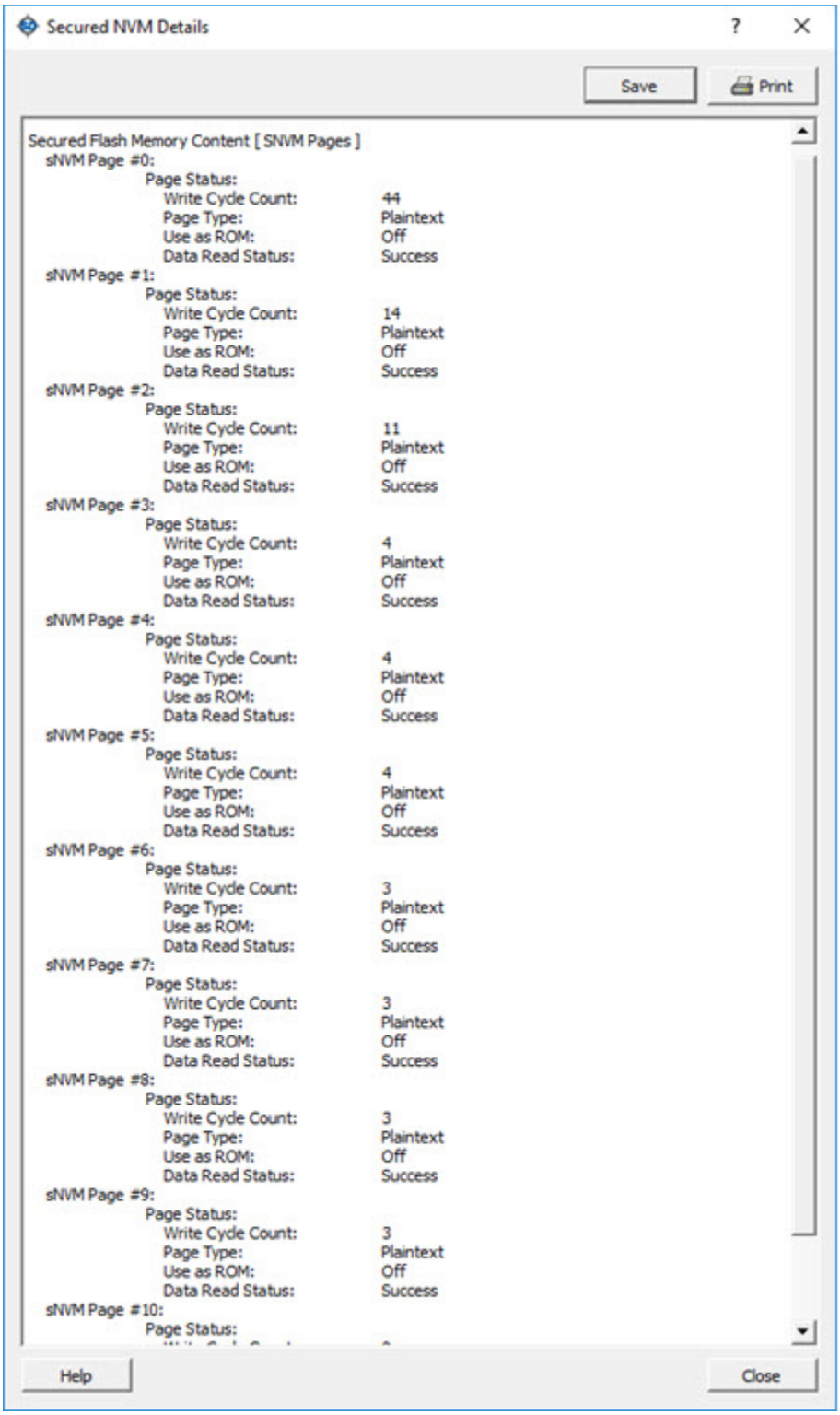


Figure 40 · View All Page Status

Read Operation

Client View

The Client View displays all the clients that are configured in the design. When a client is expanded, a table listing all pages is displayed.

When a client is selected, the Read from Device button is enabled. Click **Read from Device** to read the content of the client. A client can have one or more pages. Refresh Client Details option is given to the user to refresh the table. Click **Refresh Client Details** to update the information in SmartDebug and refresh the table. This is helpful when a client configuration is changed using system services.

Page View

When valid parameters are entered and **Check Page Status** is clicked, a table of all pages is shown with page status information. Pages in the table are read-only and cannot be selected. The page range included in Start Page and End Page is validated, and the Read from Device button is enabled. Click **Read from Device** to read the content.

Runtime Operations

After a design is programmed into the device, you can do the following:

- Change the content of a page
- Authenticate a page
- Change the security key of each configured page

The above operations are not possible if the page is used as ROM.

You can refresh page status in SmartDebug:

- Click the **Refresh Client Details** button in the Client View tab to refresh the client view table and update it with the latest changes.
- Click the **Check Page Status** button in the Page View tab to refresh the pages in the table.

If the security key has been changed, SmartDebug prompts you to enter the USK manually.

Enter the USK in the USK Status column (Client View tab and Page View tab).

By default, the USK entered in the configurator as the USK client is used to authenticate the page.

Demo Mode

Debug sNVM is supported in Demo Mode. The Client View and Page View are supported. Data from device initialization and configurators is shown in the Client View and User Design View.

Debug Transceiver

Debug Transceiver

The Debug Transceiver feature in SmartDebug checks the lane functionality and health for different settings of the lane parameters.

By default, lanes are configured in full duplex mode. Half duplex mode is also supported for each lane in the Quads. More information about half duplex mode lane configuration is provided in the following sections:

- Configuration Report
- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor
- Signal Integrity

- Optimize Receiver

To access the Debug Transceiver feature in SmartDebug, click **Debug Transceiver** in the main SmartDebug window.

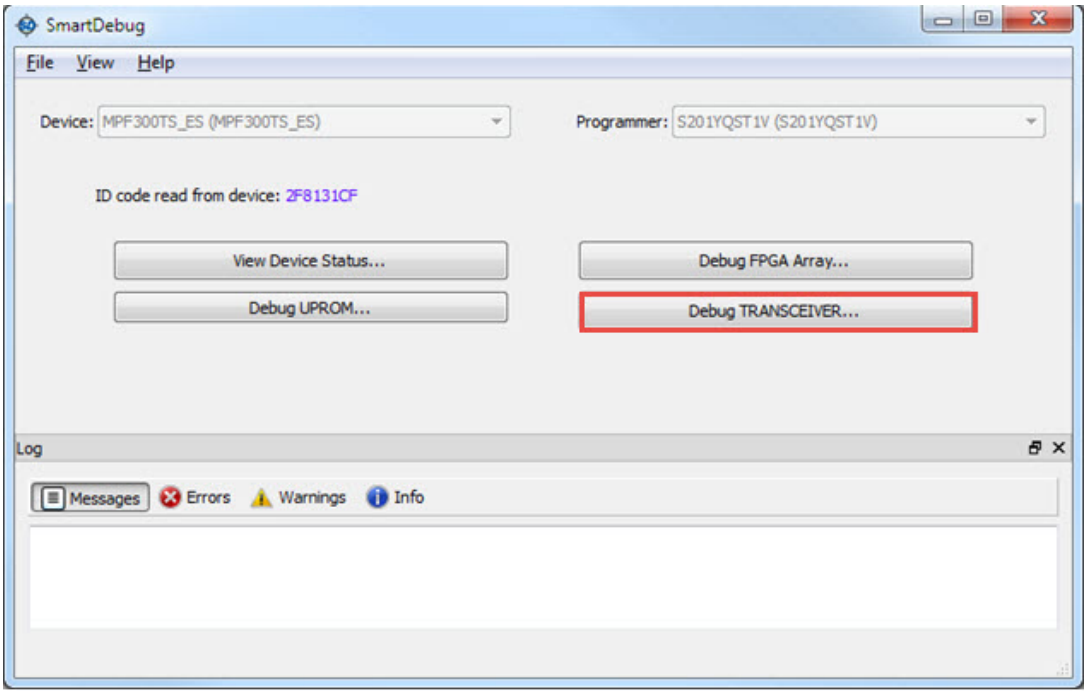


Figure 41 · SmartDebug Window - Debug TRANSCEIVER

This opens the Debug TRANSCEIVER dialog box, which is shown in the following example.

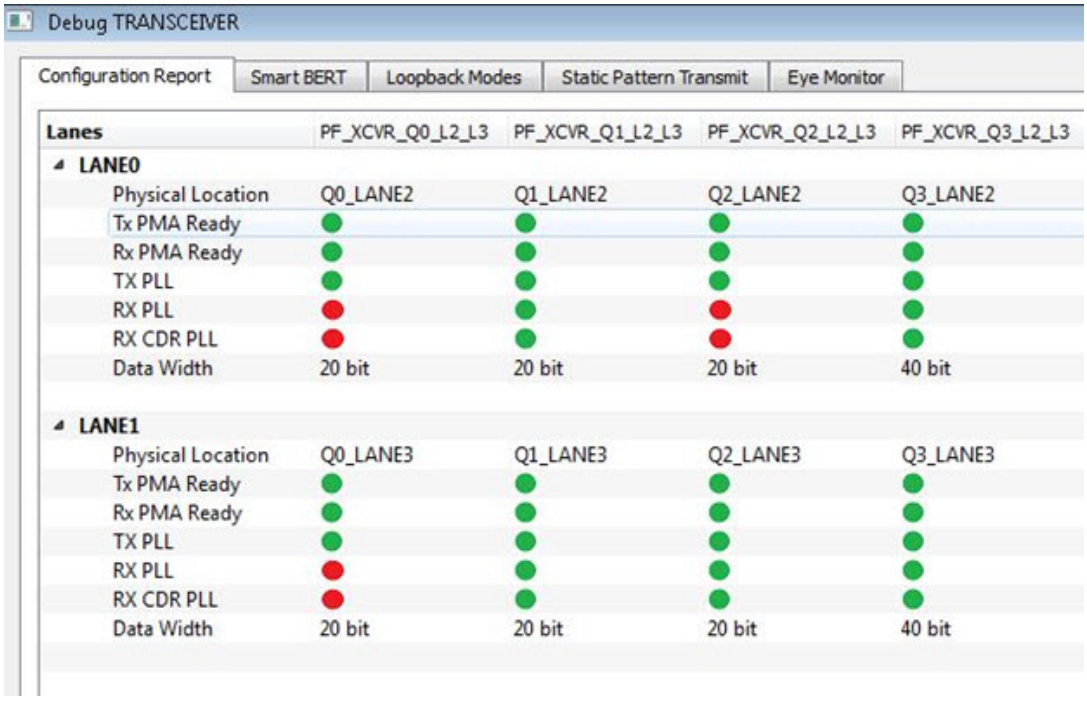


Figure 42 · Debug Transceiver Dialog Box

Debug Transceiver has five distinct debug features, which are represented by tabs in the Debug TRANSCEIVER dialog box:

- Configuration Report (shown by default when the dialog box opens)
- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor

Configuration Report

The Configuration Report is the first tab in the Debug TRANSCEIVER dialog box, and is shown by default when the dialog box opens. The Configuration Report shows the lane status/health properties of all lanes of Quads in the design.

Click the **Refresh** button to refresh the information.

Note: The report refreshes automatically when you navigate from another tab.

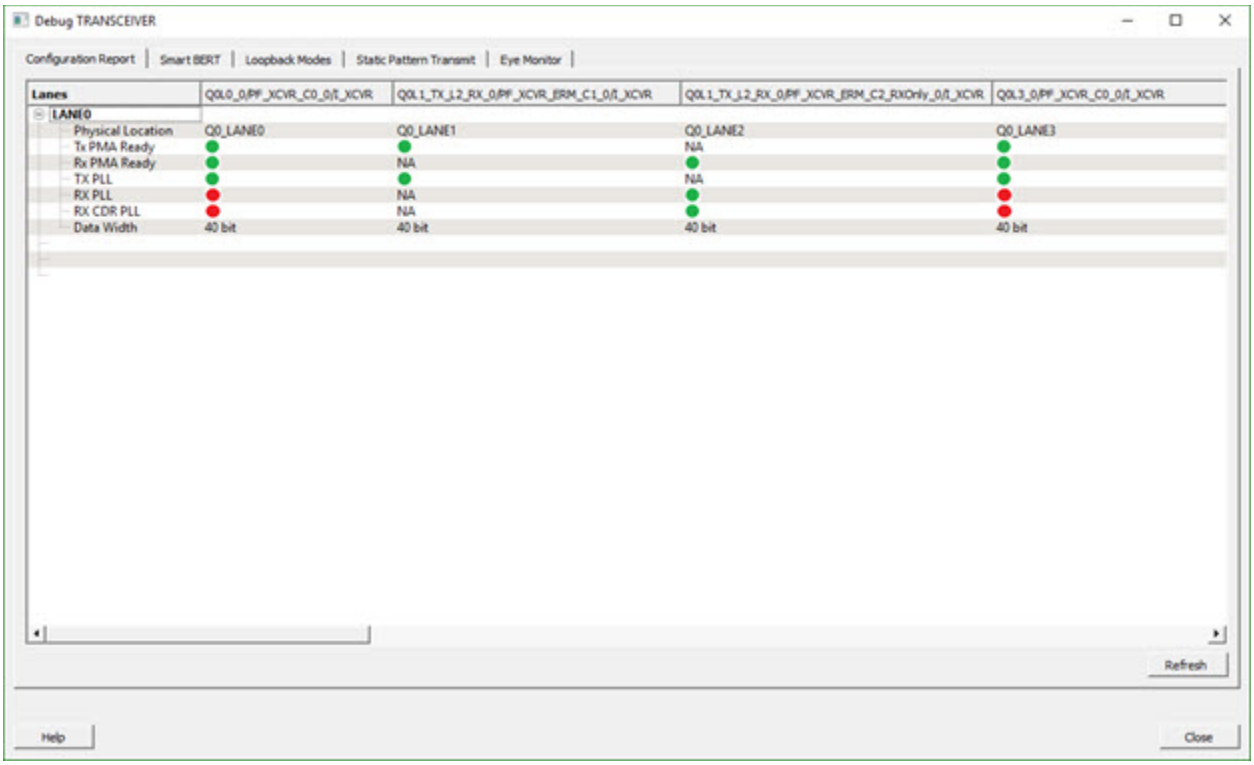


Figure 43 · Debug TRANSCEIVER - Configuration Report

The Configuration Report shows the physical location, status/health, and data width for all lanes of all the quads enabled in the system controller.

Parameter information is shown in table format, with lane numbers as rows and transceiver instance names as columns.

The lane parameters are as follows:

Physical Location - Physical block and lane location in the system controller.

Tx PMA Ready - Indicates if the Tx of the lane is powered up and ready for transactions. Rx-only lane in half duplex mode is shown as "NA".

Rx PMA Ready - Indicates if the Rx of the lane is powered up and ready for transactions. Tx-only lane in half duplex mode is shown as "NA".

TX PLL - Indicates if the lane is locked onto TX PLL. Rx-only lane in half duplex mode is shown as "NA".

RX PLL - Indicates if the lane is locked onto RX PLL. Tx-only lane in half duplex mode is shown as "NA".

RX CDR PLL - Indicates if the lane is locked onto the incoming data. Tx-only lane in half duplex mode is shown as "NA".

For the parameters above, green indicates true and red indicates false.

Notes:

Click the **Refresh** button to update the lane status.

The report refreshes automatically when you navigate from another tab.

Transceiver Hierarchy

Transceiver Hierarchy is a lane hierarchy with all the lanes instantiated in the design shown with respect to top level instance.

Transceiver Hierarchy is shown in the following tabs: "SmartBERT" on page 55, "Loopback Modes" on page 59, "Static Pattern Transmit" on page 60, and "Eye Monitor" on page 61.

In the SmartBERT, Loopback Modes, and Static Pattern Transmit pages, check boxes allow multiple lanes to be selected for debug, as shown in the following example.

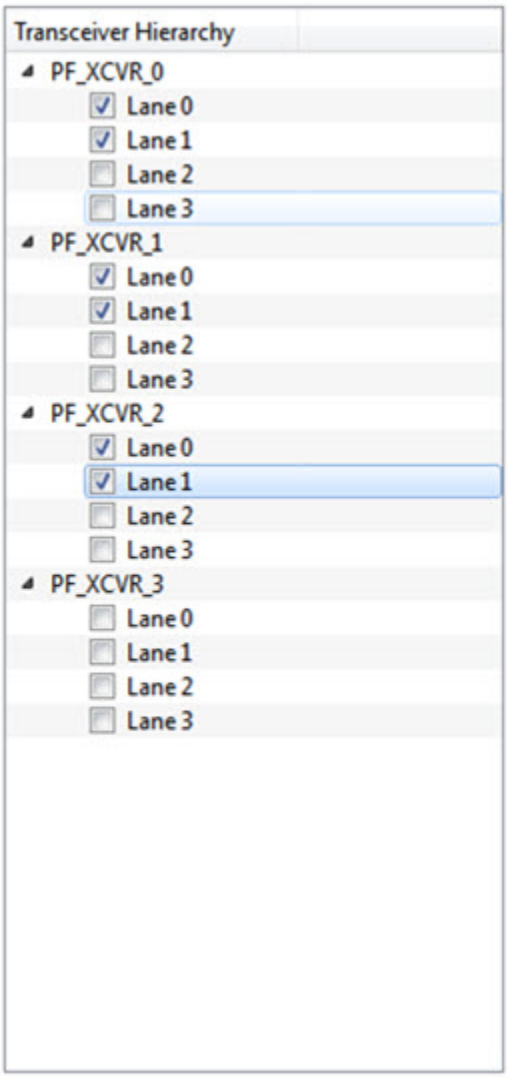


Figure 44 · Transceiver Hierarchy Lane Selection Example - SmartBERT, Loopback Modes, Static Pattern Transmit Pages

In the Eye Monitor page, eye monitoring is done one lane at a time, as shown in the following example.

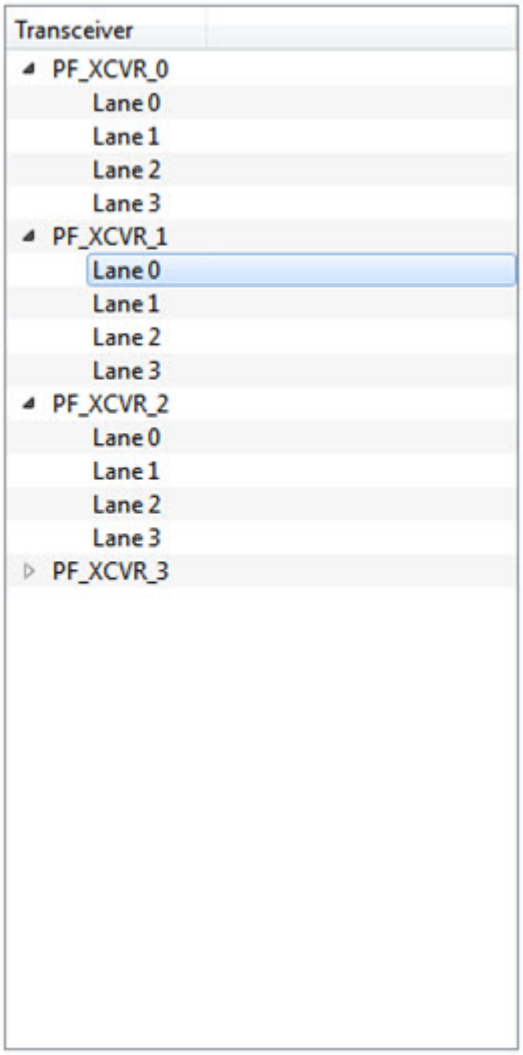


Figure 45 · Transceiver Hierarchy Lane Selection Example - Eye Monitor Page

SmartBERT

In the SmartBERT page of the Debug TRANSCEIVER dialog box, you can select lanes in the Transceiver Hierarchy and use debug options to run Smart BERT tests.

Click the **SmartBERT** tab in the Debug TRANSCEIVER dialog box to open the SmartBERT page.

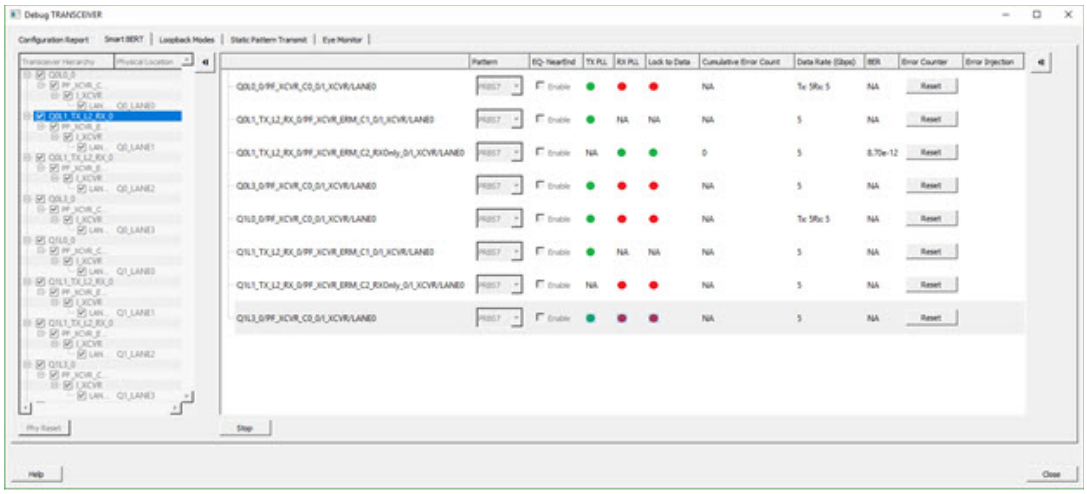


Figure 46 · Debug TRANSCEIVER - SmartBERT

The following input options and outputs are represented as columns:

Pattern – Input option. Select a PRBS pattern type from the drop-down list: PRBS7, PRBS9, PRBS15, PRBS23, or PRBS31. The default is PRBS7.

EQ-NearEnd – Input option. When checked, enables EQ-NearEnd loopback from Lane Tx to Lane Rx. Disabled for half duplex mode.

TX PLL – Indicates if lane is locked onto TX PLL when the SmartBERT test is in progress. Rx Only lane in half duplex mode is shown as "NA".

Gray – Indicates test is not in progress

Green – Indicates lane is locked onto TX PLL

Red – Indicates lane is not locked onto TX PLL

RX PLL – Indicates if lane is locked onto RX PLL when the SmartBERT test is in progress. Tx Only lane in half duplex mode is shown as "NA".

Gray – Indicates test is not in progress

Green – Indicates lane is locked onto TX PLL

Red – Indicates lane is not locked onto TX PLL

Lock to Data – Indicates if lane is locked onto incoming data / RX CDR PLL when the SmartBERT test is in progress. Tx Only lane in half duplex mode is shown as "NA".

Gray – Indicates test is not in progress

Green – Indicates lane is locked onto TX PLL

Red – Indicates lane is not locked onto TX PLL

.**Cumulative Error Count** – Displays the error count when the SmartBERT test is in progress.

Data Rate – Data rates are shown according to the configured data rates for all duplex modes except for Independent TxRx, where both Tx data rate and Rx data rate are shown. See the example figure above.

BER – Calculates the Bit Error Rate (BER) from the cumulative error count and data rate and displays it in the column.

Error Counter Reset – Resets the error counter and BER of the lane. A reset can be done at any time.

All output parameters are updated approximately once per second, with their values retrieved from the device.

To add lanes, in the Transceiver Hierarchy, check the boxes next to the lanes to be added. To remove lanes, uncheck the boxes next to the lanes to be removed.

Select the desired options and click **Start** to start the Smart BERT test on all selected lanes.

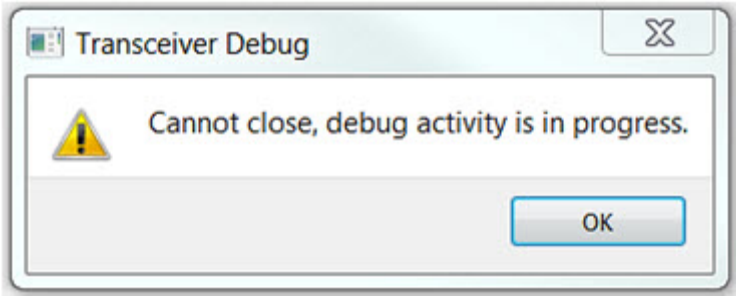
Note: A popup message appears if a test cannot be started on one lane, multiple lanes, or all lanes. Tests will start normally on all unaffected lanes.

Click the **Phy Reset** button to do a Phy reset on all checked lanes in the Transceiver Hierarchy. This button is disabled when a PRBS test is in progress.

Edit the signal integrity option of any lane by selecting the lane in the PRBS tree and modifying the option in the Signal Integrity group box.

Note: You can navigate to other tabs when a SmartBERT test is in progress, but you cannot perform any debug activity except to use Plot Eye for any lane on the Eye Monitor page.

Note: You cannot close the SmartBERT window when a test is in progress. Attempting to do so will result in the following message:



Click the **Stop** button to stop the SmartBERT test on all lanes simultaneously.

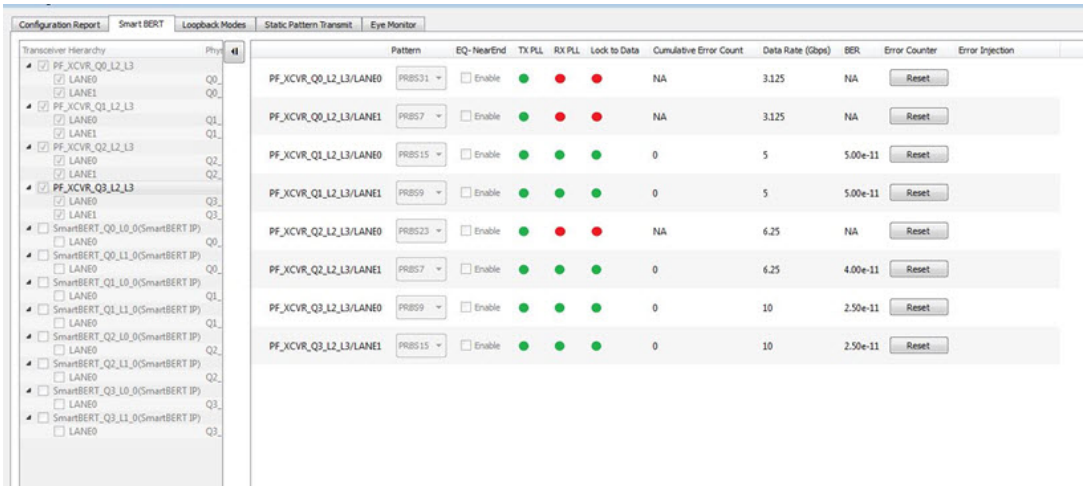
SmartBERT IP

The CoreSmartBERT core provides a broad-based evaluation and demonstration platform for PolarFire transceivers (PF_XCVR). Parameterizable to use different transceivers and clocking topologies, the SmartBERT core can also be customized to use different line rates and reference clock rates. Data pattern generators and checkers are included for each PF_XCVR, giving several different Pseudo-random binary sequences PRBS (2**7**,2**23**, 2**15** and 2**31**).

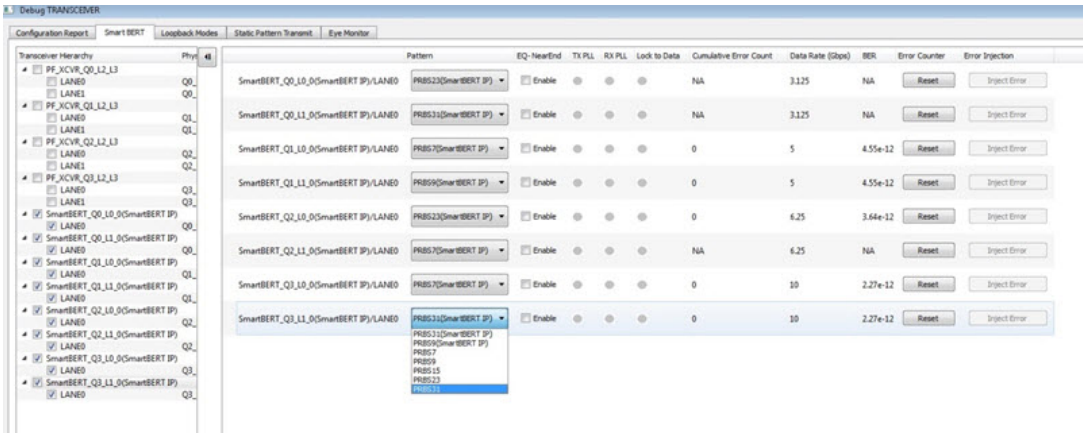
Each SmartBERT IP can have four lanes configured. Each Lane can have the pattern type PRBS7, PRBS9, PRBS23, or PRBS31 configured.

SmartDebug identifies the lanes that are used by the SmartBERT IP and distinguishes them by adding "_IP" to the SmartBERT IP instance name in the Transceiver Hierarchy. See the following example.

You can expand a SmartBERT IP instance to see all the lanes. Check the checkbox next to a lane to add it to the SmartBERT IP page and include the lane in a PRBS test. If the box is unchecked, it will not be added. See the following example.



You can select patterns for the added lane(s) from a drop-down list. See the following example.



After the lane(s) have been added and the patterns(s) selected, click **Start** to enable the transmitter and receiver for the added lanes and patterns.

Error Injection

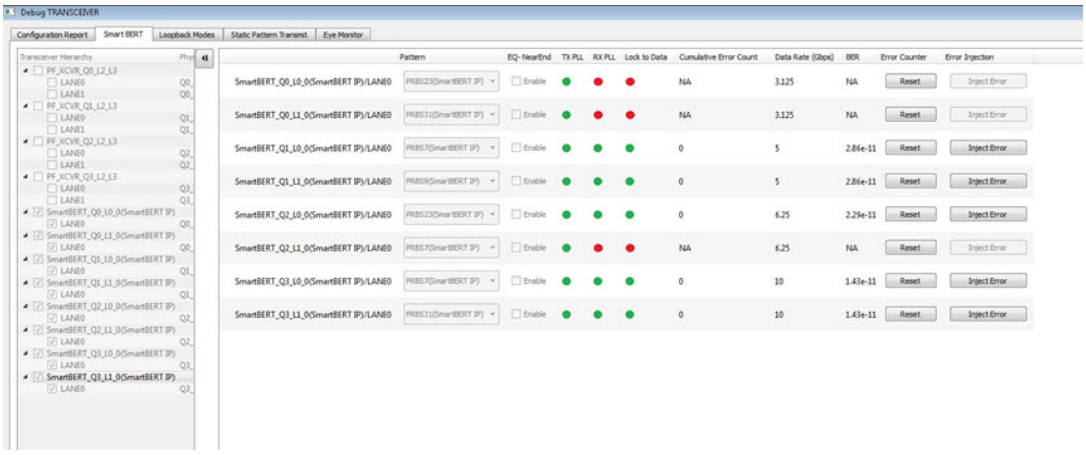
When SmartBERT IP lanes are added, you will see the Error Injection column and Error Inject button. Errors can be injected by clicking the **Error Inject** button when a PRBS test is running. This feature tests whether the error is identified by the pattern checker.

Note: This column does not appear for non-SmartBERT IP lanes, or if a non-configured PRBS pattern has been selected.

Error Count

Error Count is shown when a lane is added and a PRBS pattern is run. The error count can be cleared by clicking the **Reset** button under the Error Counter column.

The following example shows the Reset and Inject Error buttons.



Loopback Modes

The Loopback Modes page in the Debug TRANSCEIVER dialog box allows you to select lanes from the Transceiver Hierarchy and use Loopback Mode debug options.

Click the **Loopback Modes** tab in the Debug TRANSCEIVER dialog box.

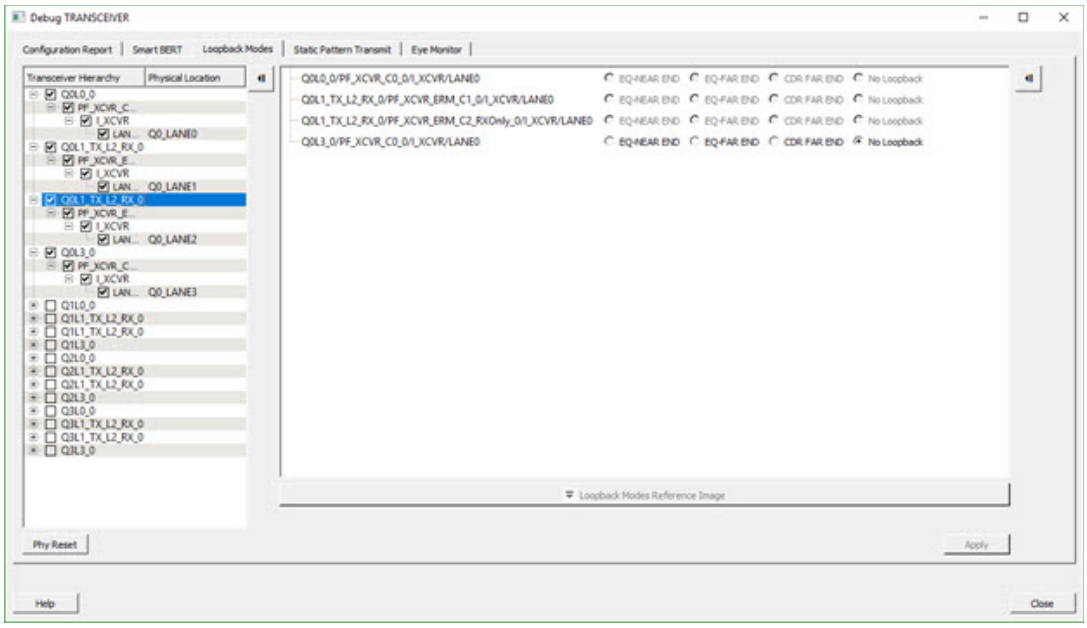


Figure 47 · Debug TRANSCEIVER - Loopback Modes

You can select the desired loopback type (EQ-NEAREND, EQ-FAREND, CDRFAREND, or No Loopback) for each lane.

Note: These loopback types (EQ-NEAREND, EQ-FAREND, CDRFAREND, No Loopback) are enabled only for full duplex modes and are disabled for the three half duplex modes. See the example figure above.

EQ-NEAR END – Set EQ-Near End loopback from Lane Tx to Lane Rx. This loopback mode is supported up to 10.3125 Gbps.

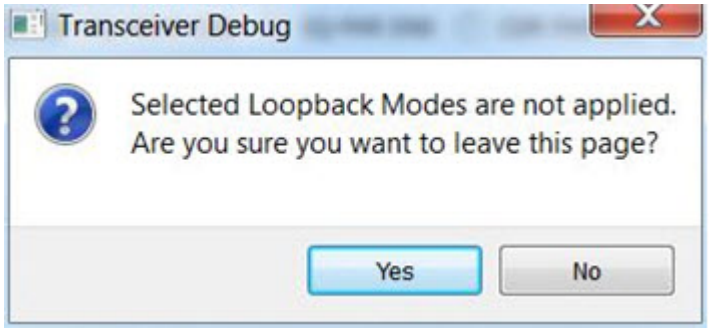
EQ-FAR END – Set EQ-Far End loopback from Lane Tx to Lane Rx.

CDR FAR END – Set CDR Far End loopback from Lane Rx to Lane Tx.

No Loopback – Set this option to have no loopback between Lane Tx and Lane Rx. (For external loopback using PCB backplane or High Speed Loopback cables.)

When you have selected the desired options, click **Apply** to enable the selected loopback mode on the lane(s).

Note: If you proceed to another tab without applying your changes to loopback modes, the following popup message appears:



Click **Yes** to ignore the changed selections and move to another selected page.

Click **No** to remain on the current page.

Static Pattern Transmit

In the Static Pattern Transmit page of the Debug TRANSCEIVER dialog box, you can select lanes from the Transceiver Hierarchy and use Static Pattern Transmit debug options.

Click the **Static Pattern Transmit** tab in the Debug TRANSCEIVER dialog box to open the Static Pattern Transmit page.

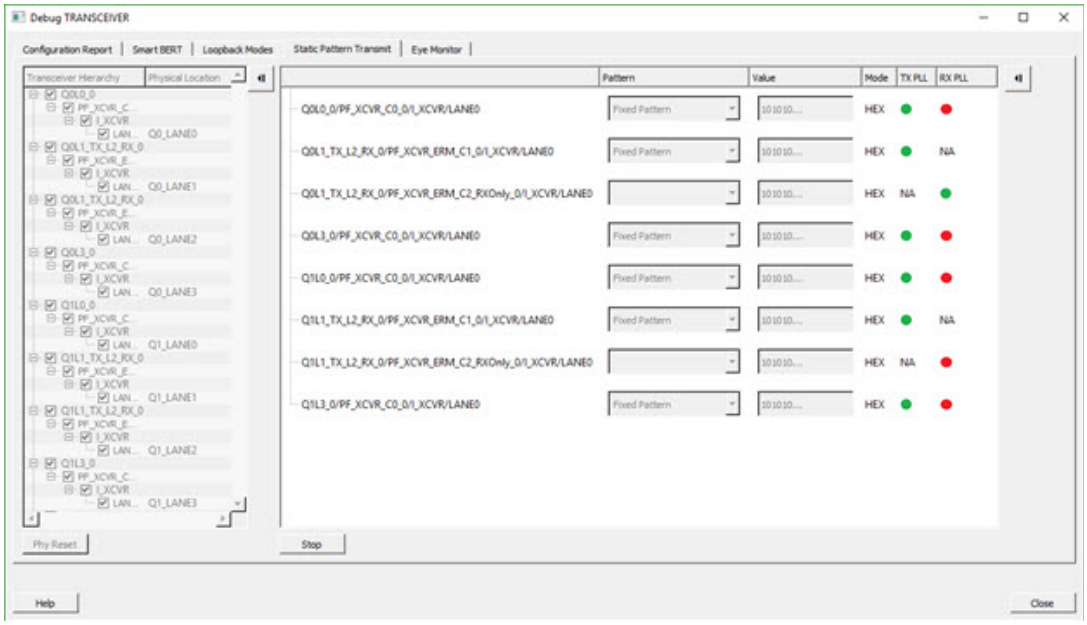


Figure 48 · Debug TRANSCEIVER – Static Pattern Transmit

When a lane is added from the Transceiver Hierarchy, the following debugging options can be selected:

Pattern – Pattern selection is available/enabled for all modes except Rx Only, since it is applicable only for Tx (and Tx is present in Full Duplex, Tx Only, and Independent TxRx)

- Fixed Pattern is a 10101010... pattern. Length is equal to the data width of the Tx Lane.
- Max Run Length Pattern is a 1111000... pattern. Length is equal to the data width of the Tx Lane, with half 1s and half 0s.
- User Pattern is a user defined pattern in the value column. Length is equal to the data width.

Value – Editor available only with the User Pattern pattern type. For other pattern type selections, it is disabled.

- Takes the input pattern to transmit from the Lane Tx of selected lanes.
- Pattern type should be Hex numbers, and not larger than the data width selected.
- Internal validators dynamically check the pattern and indicate when an incorrect pattern is given as input.

Mode – Currently, HEX mode is supported for pattern type.

TX PLL – Indicates Lane lock onto TX PLL when Static Pattern Transmit is in progress

- *Gray* – Test is not in progress
- *Green* – Lane is locked onto TXPLL
- *Red* – Lane is not locked onto TXPLL

RX PLL – Indicates Lane lock onto RX PLL when Static Pattern Transmit is in progress

- *Gray* – Test is not in progress
- *Green* – Lane is locked onto RXPLL
- *Red* – Lane is not locked onto RXPLL

Click **Start** to start Static Pattern Transmit on selected lanes.

Click **Stop** to stop Static Pattern Transmit test on selected lanes.

Eye Monitor

You can determine signal integrity with the Eye Monitor feature. It allows you to create an eye diagram to measure signal quality. Eye Monitoring estimates the horizontal eye-opening at the receiver serial data sampling point and helps you select an optimum data sampling point at the receiver.

To use the Eye Monitor feature, do the following:

1. Invoke SmartDebug from Libero.
2. Click the **Eye Monitor** tab in the Debug TRANSCEIVER dialog box.

Select Eye Output

The Select Eye Output drop-down is enabled when an Eye Plot log file is browsed and loaded in the Eye Monitor page. Click **Browse File** to load the Eye Plot output files.

The drop-down list includes all eye outputs logged in the file, as shown in the example figure below. If the loaded Design Initiated Eye Plot log file does not contain any eye output, it is disabled.

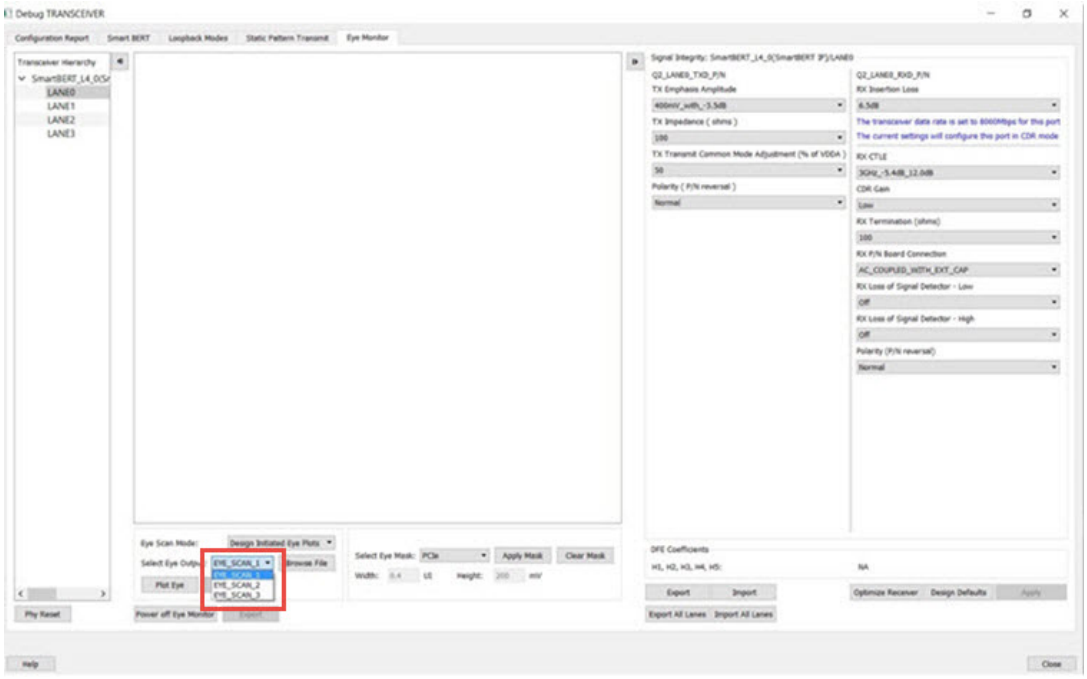


Figure 49 · Eye Monitor Page (Select Eye Output drop-down shown)

After selecting Eye output from the Select Eye Output drop-down, click **Plot Eye** to start eye monitoring for the lane. The Eye diagram displays, as shown in the following example.

Note that the TagName for the selected eye output is shown above the eye diagram.

Note: Ensure data transmission on Lane Rx for successful monitoring.

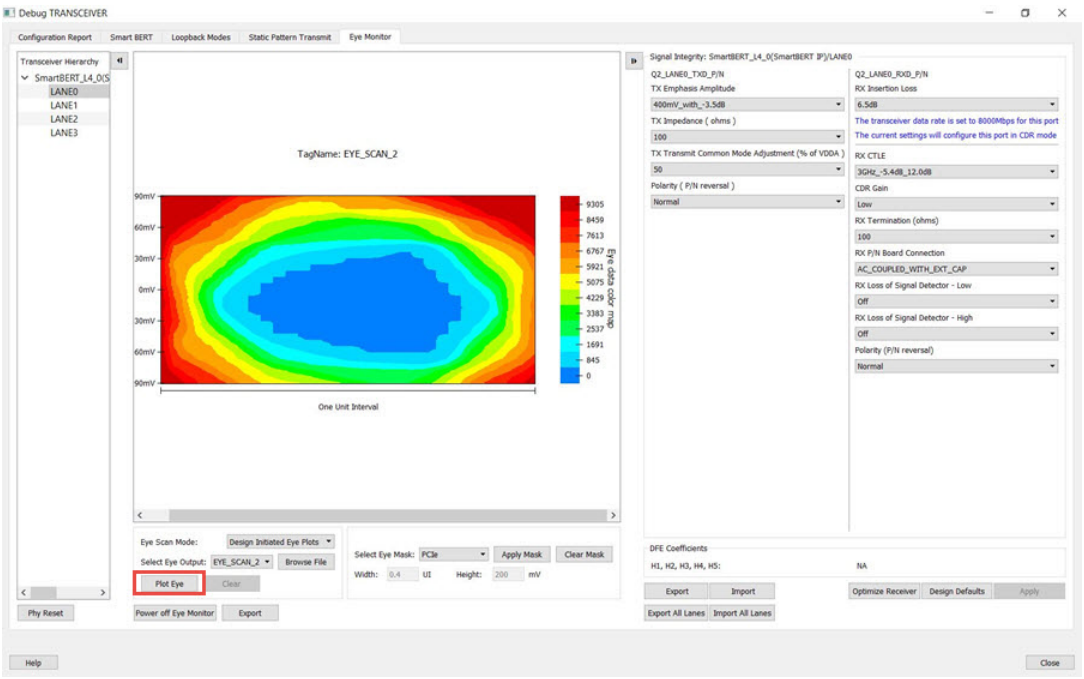


Figure 50 · Eye Monitor Example

Eye Scan Mode

Eye Monitor can be run in two modes: Normal or Infinite Persistent. Choose the desired mode in the Eye Scan Mode drop-down.

Normal Mode

Note: This feature is not available for Tx Only mode. In this mode, all Eye Monitor buttons and the Optimize Receiver button are disabled (grayed out).

In Normal mode, Plot Eye performs single eye scanning and displays the Eye Diagram in the GUI. See the following example figure.

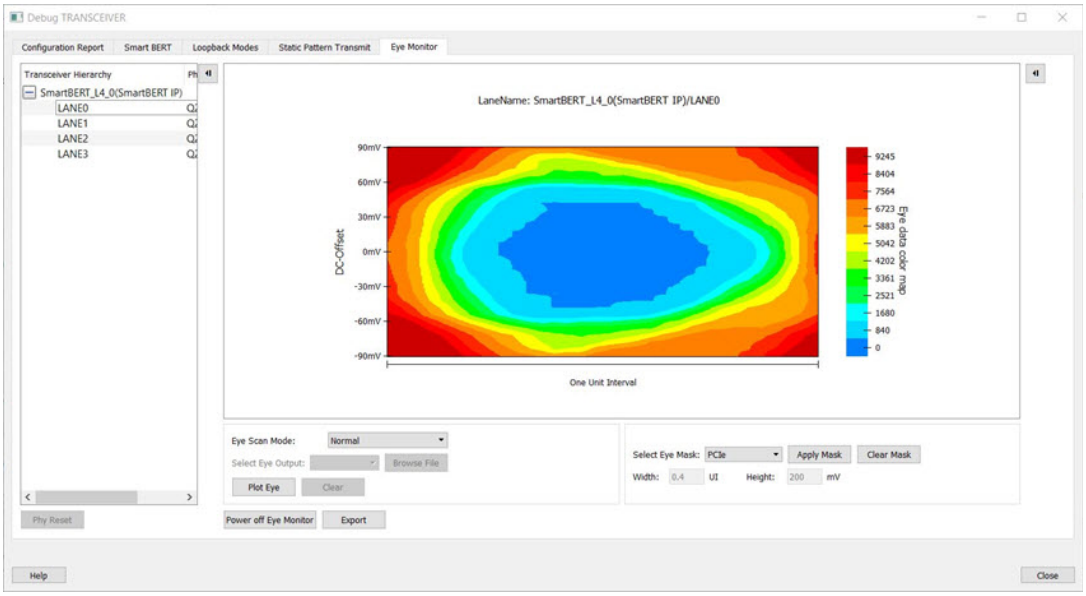


Figure 51 · Eye Scan Mode - Normal

Infinite Persistent Mode

Note: This feature is not available for Tx Only mode.

When Infinite Persistent mode is selected, the Plot Eye button changes to Start Plot Eye. Click **Start Plot Eye** to start Infinite Persistent eye monitoring. See the following example figure.

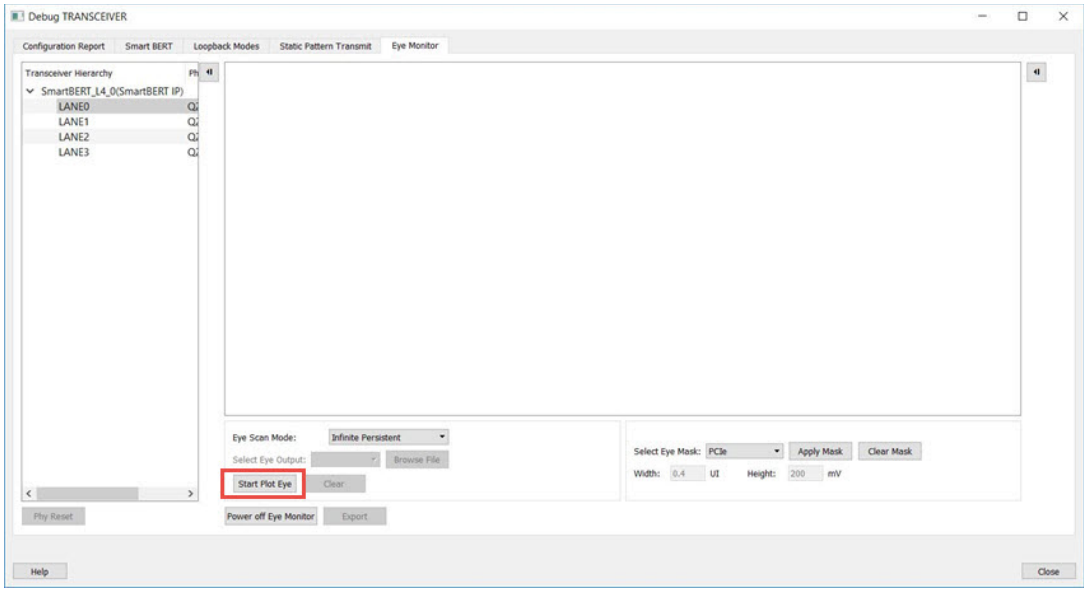


Figure 52 · Infinite Persistent Mode - Start Plot Eye

The **Start Plot Eye** button changes to **Stop Plot Eye** and the infinite scanning and cumulation process begins. In every iteration, the eye is cumulated with all previous eyes to make a single “cumulative eye”. This cumulative eye is displayed with a color scheme in the GUI, as shown in the following example figure. The completed iteration number and the cumulative BER is updated and displayed after every iteration, along with the cumulative eye. To stop cumulative eye monitoring, click **Stop Plot Eye**. The process halts after the current iteration completes.

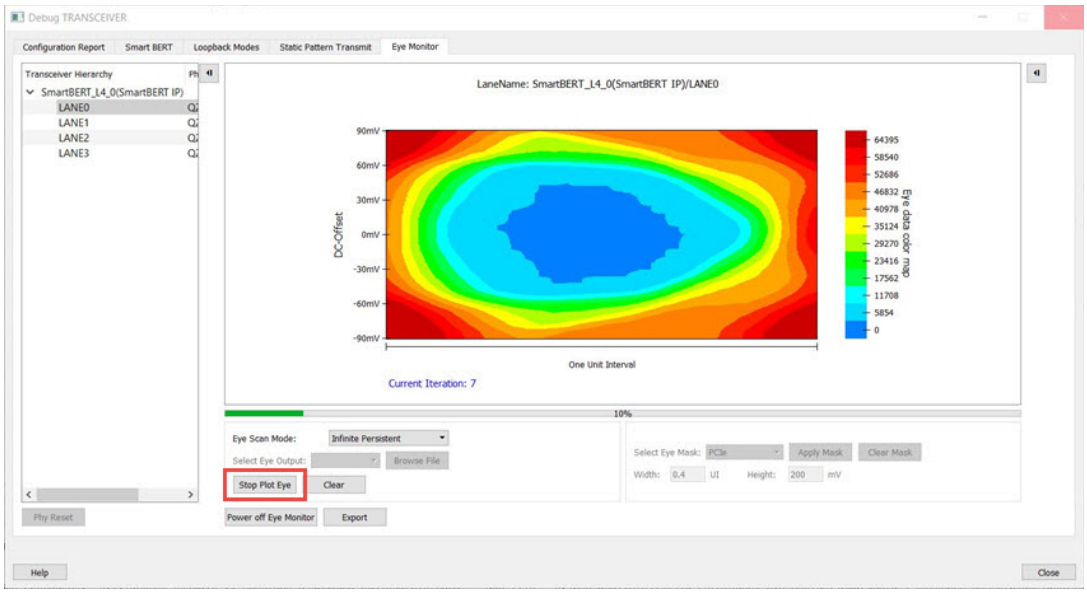


Figure 53 · Infinite Persistent Mode - Stop Plot Eye

Clear

The **Clear** button is enabled for Infinite Persistent eye scan mode and is disabled for Normal eye scan mode. At any time during Infinite Persistent eye monitoring, clicking the **Clear** button clears the cumulative eye computation and then and then starts a new cumulative eye computation. Note that Current Iteration count does not reset; only the cumulative eye is cleared.

Additional Eye Output Text Files

Data files are generated in Normal mode and Infinite Persistent mode. These files contain the Eye Data Errors in matrix format with the name “Plot_Eye*.txt”, where * stands for numbering starting from 1. The files are generated in the “designer” folder of the Libero project for integrated SmartDebug from Libero and in the “standalone project folder” for standalone SmartDebug. In Normal mode, one file is generated, as eye scanning is done only once. In Infinite Persistent mode, one file is generated per iteration. You must have sufficient space in the project location when running Infinite Persistent eye monitoring. The numbering used in the file naming continues to increment until the infinite persistent mode eye plot activity is in progress. When you close and restart SmartDebug, the file numbering begins again from 1. Be sure to save these files before starting Eye Monitoring again from a different SmartDebug session; otherwise, they will be overwritten.

Error Handling

Eye Scanning can be performed successfully only if there is data traffic on the Lane Rx when Eye Monitoring is in progress. In Normal Mode, when an Eye Scan fails, a popup message is displayed. In Infinite Persistent mode, when an Eye Scan fails in any interation, a popup message is displayed and Eye scanning terminates.

Signal Integrity

The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of .pdc files.

The Signal Integrity pane appears in the following SmartDebug pages:

- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor

When you open Debug Transceiver in SmartDebug and click the SmartBERT, Loopback Modes, Static Pattern Transmit, or Eye Monitor tab, all parameters in the Signal Integrity pane are shown as Undefined. Only the Export All Lanes and Import All Lanes buttons are enabled. See the following example.

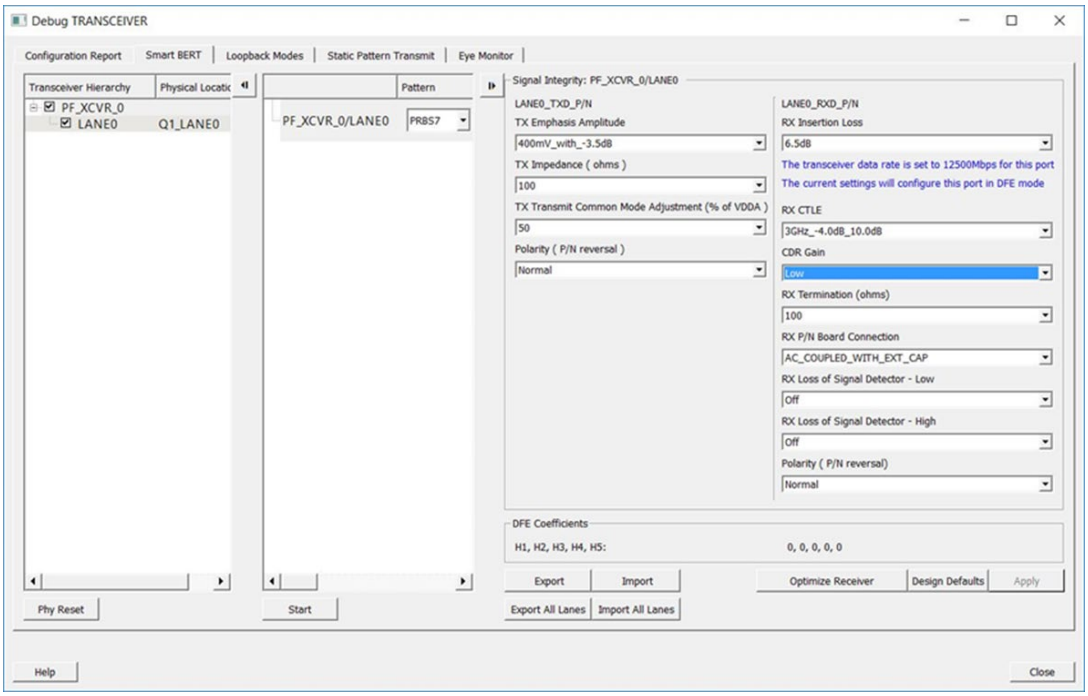


Figure 54 · Debug TRANSCEIVER - Signal Integrity (Full Duplex Mode)

In full duplex mode, all parameters (Tx and Rx) are imported/exported.

The following example figures show Signal Integrity for Tx Only mode, Rx Only mode, and Independent TxRx mode.

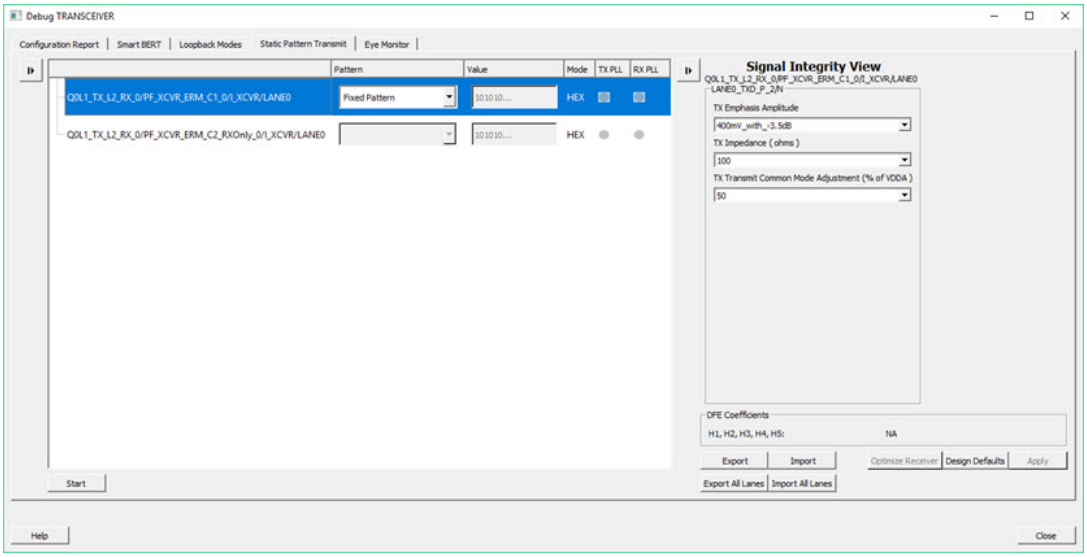


Figure 55 · Signal Integrity - Tx Only Mode

In this mode, only Tx parameters are imported/exported.

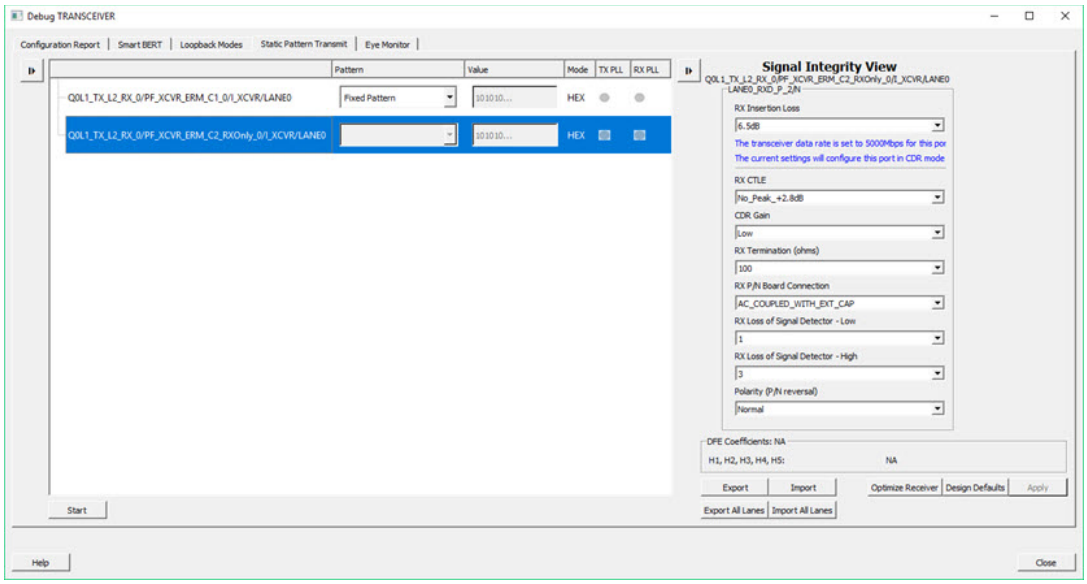


Figure 56 · Signal Integrity - Rx Only Mode

In this mode, only Rx parameters are imported/exported.

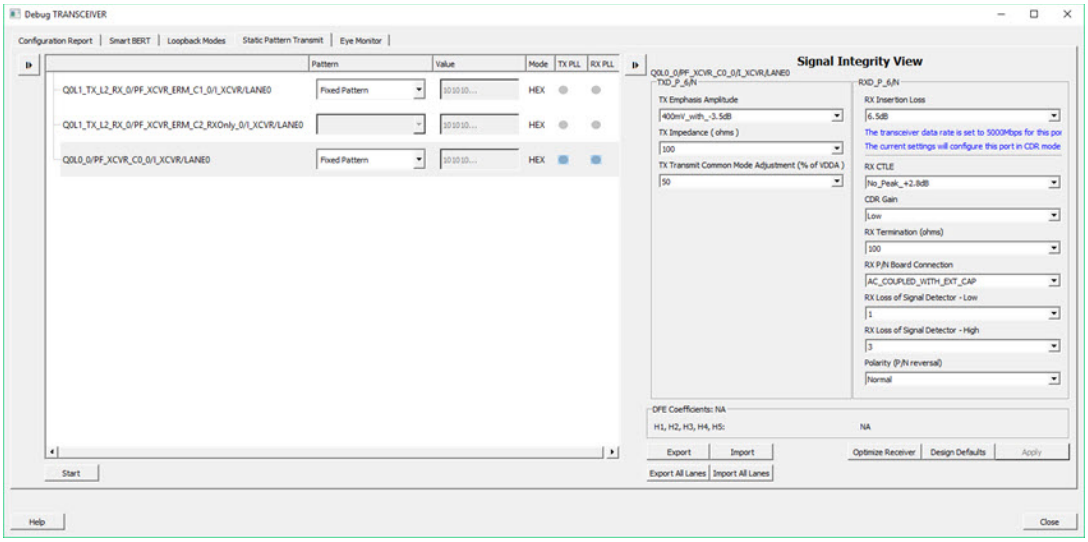


Figure 57 · Signal Integrity - Independent TxRx Mode

In this mode, Tx and Rx parameters are imported/exported.

When a lane is selected in the SmartBERT, Loopback Modes, Static Pattern Transmit, or Eye Monitor pages, the corresponding Signal Integrity parameters (configured in the I/O Editor or changed in SmartDebug) are enabled and shown in the Signal Integrity pane.

The selected lane instance name is displayed in the Signal Integrity group box, and the Export, Import, and Design Defaults buttons are enabled.

You can select options for each parameter from the drop-down for that parameter. Click **Apply** to set the selected transceiver instance with the selected options.

The *Polarity (P/N reversal)* parameter has been added. You can choose Normal or Inverted from the drop-down. Note that this parameter is not available for MPF300T_ES (Rev C) or MPF300T_XT (Rev E) devices.

The *CDR Gain* parameter has been added for MPF300T, MPF100T, MPF200T, MPF500T devices, and you can select the High or Low option from the drop-down. This parameter is supported for Export, Export All, Import, Import All, Design Defaults, and Apply flows of Signal Integrity. Note that this parameter is not available for MPF300T_ES (Rev C) or MPF300T_XT (Rev E) devices.

Note: The Apply button is enabled when you make a selection for any parameter.

If you change parameter options and click another lane, move to another tab, or click Import, Import All, or Design Defaults without applying the changes, you will see the following message stating "*Some Signal Integrity options are modified. How do you wish to continue?*"

Click **Apply** to apply the changes or **Discard** to discard the changes.

Note: If you change any register setting related to Signal Integrity, power cycle the board, or do a user reset of the device, the XCVR lane signal integrity state may be different than what is shown in the SmartDebug Signal Integrity tab.

To ensure that the Signal Integrity in SmartDebug is in sync with the Signal Integrity state of the device, click the **Design Defaults** button in SmartDebug. This will set the SI in the Signal Integrity tab to the design constraints (SI parameter values chosen from the IO Editor).

Design Defaults

Clicking the **Design Defaults** button loads the Signal Integrity parameter options for the selected lane instance. These are the signal integrity settings that were selected in the Libero design flow run and reside in the STPL file. Design Default parameter options are applied to the device and updated in Modified Constraints.

Note: Modified Constraints is a list of I/O constraints set on the TXP/N and RXP/N lane ports. For a selected lane, this set is created in the SmartDebug session and is updated when a Signal Integrity parameter option is modified and applied or an external PDC file is imported.

Export

Clicking the **Export** button exports the current selected parameter options and other physical information for the selected lane instance to an external PDC file. A pop-up box prompts you to choose the location where you want the .pdc file to be exported.

The exported content will be in the form of two set_io commands, one for the TXP port and one for the RXP port of the selected lane instance.

Export All Lanes

Clicking the **Export All Lanes** button exports the current selected parameter options and other physical information for all lane instances in the design to an external PDC file. A pop-up box prompts you to choose the location where you want the .pdc file to be exported.

Import

Clicking the **Import** button imports Signal Integrity parameter options and other physical information for the selected lane from an external PDC file.

The Signal Integrity parameter options are applied to the device and updated in Modified Constraints.

Import All Lanes

Clicking the Import All Lanes button imports Signal Integrity parameter options and other physical information for all lanes from an external PDC file.

The Signal Integrity parameter options are applied to the device and updated in Modified Constraints.

Signal Integrity and Calibration Report

You can generate and extract an additional report containing Signal Integrity parameters and options, CTLE register settings, and DFE coefficients by clicking **Export** or **Export all** in SmartDebug. Click **Export** to export the report only for the selected lane. Click **Export all** to export a report for all the lanes. This report is a text file that contains the Signal Integrity parameters and options, CTLE register values {CST1, RST1, CST2, RST2}, and DFE coefficient values {H1, H2, H3, H4, H5}. The exported file has a .txt extension with the same name as the .pdc file, and is exported in the same location. DFE Coefficients are exported only for DFE configured lanes. See the following example report.

```
SIGNAL INTEGRITY AND CALIBRATION REPORT
================================================================================
PF_XCVR_3/LANE0

Signal Integrity:
TX_EMPHASIS_AMPLITUDE=400mV_with_-3.5dB
TX_IMPEDANCE=150
TX_TRANSMIT_COMMON_MODE_ADJUSTMENT=50
TX_POLARITY=Normal
TXPLL_BANDWIDTH=LOW
RX_INSERTION_LOSS=6.5dB
RX_CTLE=3GHz_+5.5dB_2.1dB
RX_TERMINATION=100
RX_PN_BOARD_CONNECTION=AC_COUPLED_WITH_EXT_CAP
RX_LOSS_OF_SIGNAL_DETECTOR_LOW=PCIE
RX_LOSS_OF_SIGNAL_DETECTOR_HIGH=PCIE
RX_POLARITY=Normal
--------------------------------------------------------------------------------
CTLE Coefficients:
CST1, RST1, CST2, RST2 = 3, 3, 1, 2
--------------------------------------------------------------------------------
DFE Coefficients:
H1, H2, H3, H4, H5 = 0, -1, 2, 6, 3
================================================================================
======

PF_XCVR_3/LANE0
Signal Integrity:
TX_EMPHASIS_AMPLITUDE=400mV_with_-3.5dB
TX_IMPEDANCE=100
TX_TRANSMIT_COMMON_MODE_ADJUSTMENT=80
TX_POLARITY=Normal
TXPLL_BANDWIDTH=LOW
RX_INSERTION_LOSS=17.0dB
RX_CTLE=3GHz_+5.5dB_2.1dB
RX_TERMINATION=100
RX_PN_BOARD_CONNECTION=AC_COUPLED_WITH_EXT_CAP
RX_LOSS_OF_SIGNAL_DETECTOR_LOW=PCIE
RX_LOSS_OF_SIGNAL_DETECTOR_HIGH=PCIE
```

RX_POLARITY=Normal

CTLE Coefficients:

CST1, RST1, CST2, RST2 = 3, 1, 2, 2

DFE Coefficients:

H1, H2, H3, H4, H5 = Not applicable for CDR configured lane

Signal Integrity Parameters in Half Duplex Modes

Tx Only XCVR Mode

- The Signal Integrity view shows only Tx parameters.
- Lane information is also shown in the Signal Integrity view.
- Optimize Receiver is disabled.
- The Export option exports only Tx parameters.
- The Import option imports only Tx parameters. If Rx parameters are present, the tool errors out.

Rx Only XCVR Mode

- The Signal Integrity view shows only Rx parameters.
- Lane information is also shown in the Signal Integrity view as a header.
- Optimize Receiver is enabled.
- The Export option exports only Rx parameters.
- The Import option imports only Rx parameters. If Tx parameters are present, the tool errors out.

Independent TxRx XCVR Mode

This mode is displayed as full duplex mode (no changes).

Optimize Receiver

Note: This feature is available for MPF300T, MPF100T, MPF200T, and MPF500T devices.

The Optimize Receiver function allows you to optimize the DFE coefficients and/or CTLE settings for the selected lanes, depending on receiver mode. For CDR mode receivers, CTLE settings can be optimized. For DFE mode receivers, CTLE settings and DFE coefficients can be optimized.

For DFE coefficients, the optimize function runs through an algorithm for each lane and sets the best available coefficients for each selected lane for the current temperature, voltage, and data pattern conditions. After the optimization is complete, the transceiver lanes are set to these coefficients for the user to continue debugging.

For information about how to use the optimized coefficients without SmartDebug, see the PolarFire FPGA Transceiver User Guide.

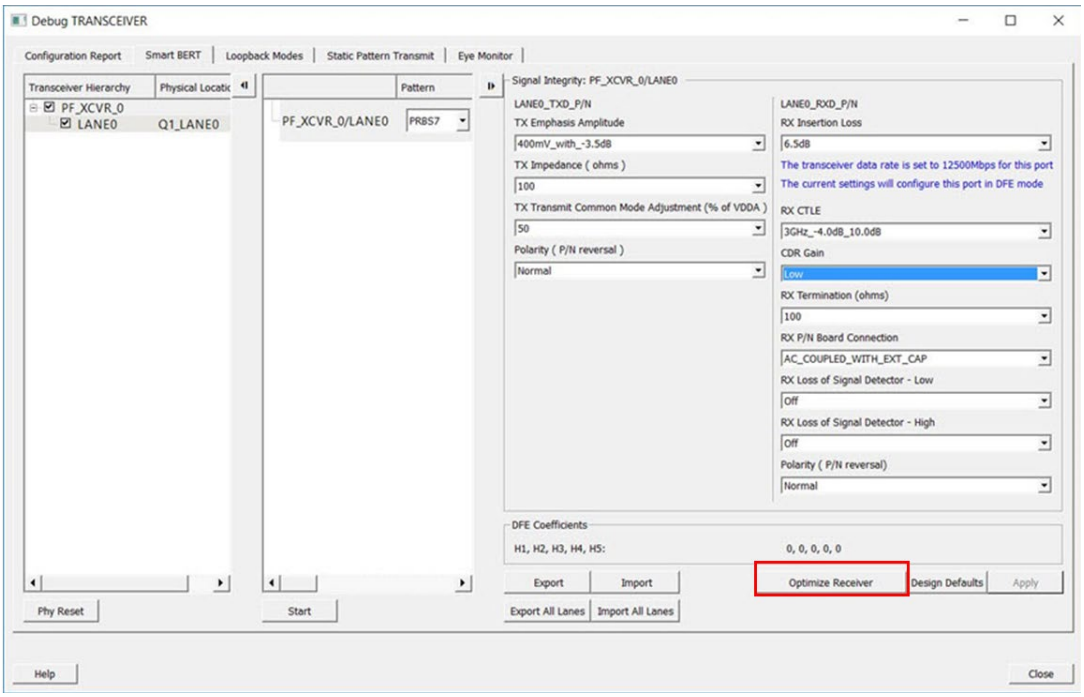


Figure 58 · Debug TRANSCEIVER - Optimize Receiver

Click **Optimize Receiver** to open the Optimize Receiver dialog box. Full duplex, Rx Only, and Independent TxRx will be shown (Tx Only lanes will not be shown).

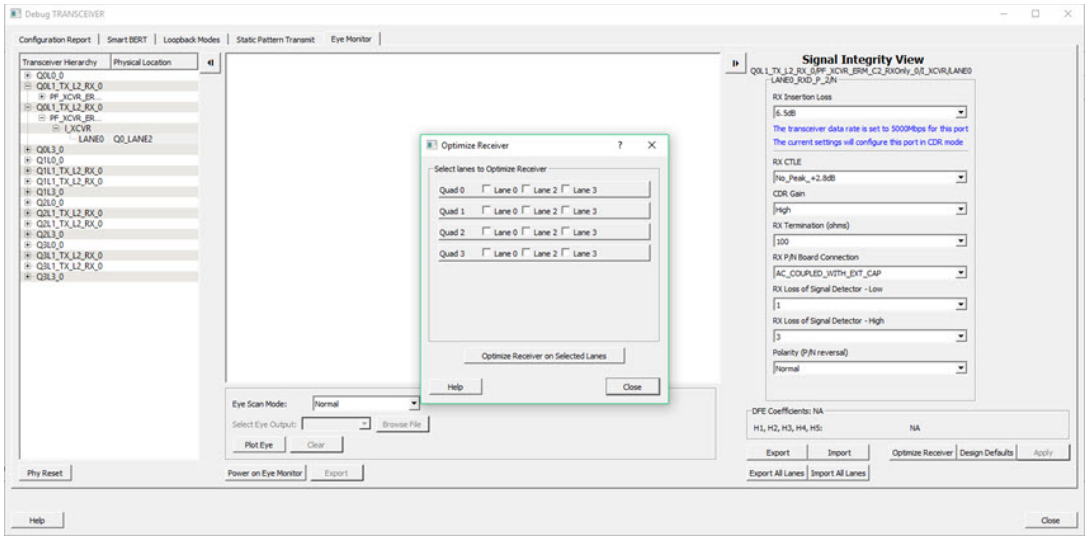


Figure 59 · Optimize Receiver Dialog Box

Select the lanes on which to run Optimize Receiver and click **Optimize Receiver on Selected Lanes**. You can select any combination of lanes including those configured in CDR or DFE. The hardware will perform CTLE calibration for CDR receivers and Full calibration for DFE receivers.

Display DFE Coefficient values

The DFE coefficients H1, H2, H3, H4, and H5 are displayed as non-editable in the Signal Integrity pane for any lane configured in DFE mode. See the following example figure.

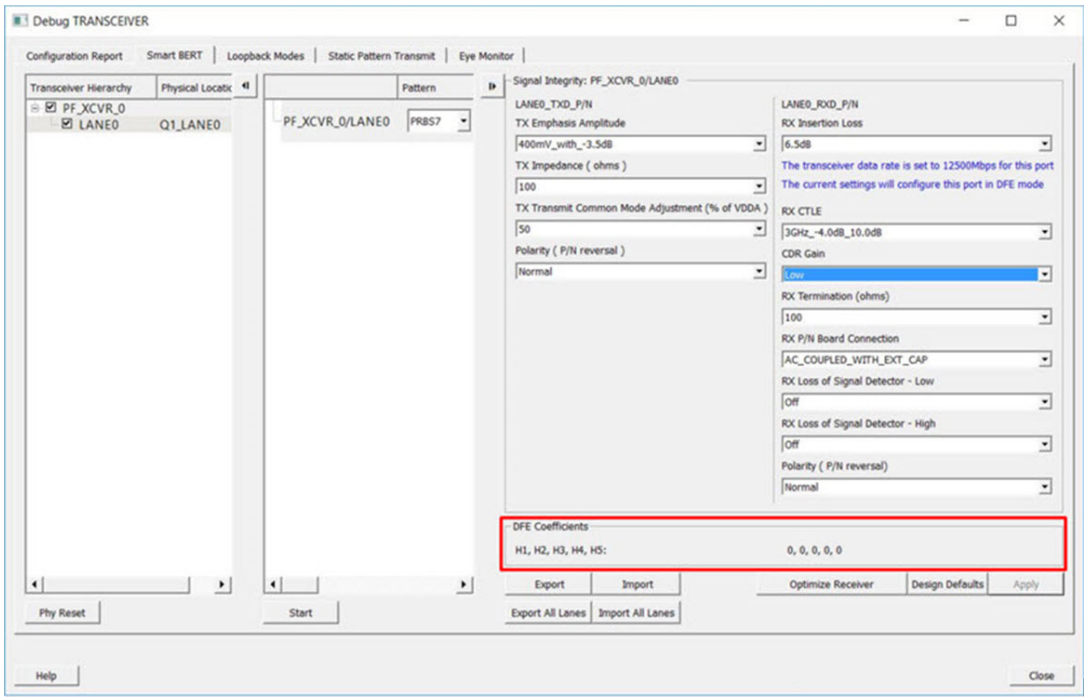


Figure 60 · Signal Integrity Pane - DFE Coefficients

DFE coefficients for CDR lanes are displayed as NA.

The DFE coefficients are read from the register fields as follows:

H1 = H1_MON

H2 = H2_MON

H3 = H3_MON

H4 = H4_MON

H5 = H5_MON

DFE coefficients are read back from the device in the following scenarios:

- When the lane is selected in the SmartBert, Loopback Modes, Static Pattern Transmit, and Eye Monitor pages.
- When a test is started/stopped on selected lane in the SmartBert page
- When a test is started/stopped on selected lane in the Static Pattern Transmit page
- When Optimize Receiver is executed on the selected lane.

PCIE Debug

The PCIE LTSSM State page is available in Debug TRANSCEIVER when PCIE has been instantiated in a design.

The PCIE LTSSM State page shows the PCIE Design Hierarchy. It contains the tree hierarchy of the PCIE instance in the design, as shown in the following example figure. The physical location of the PCIE instance is shown beside the PCIE instance in second column.

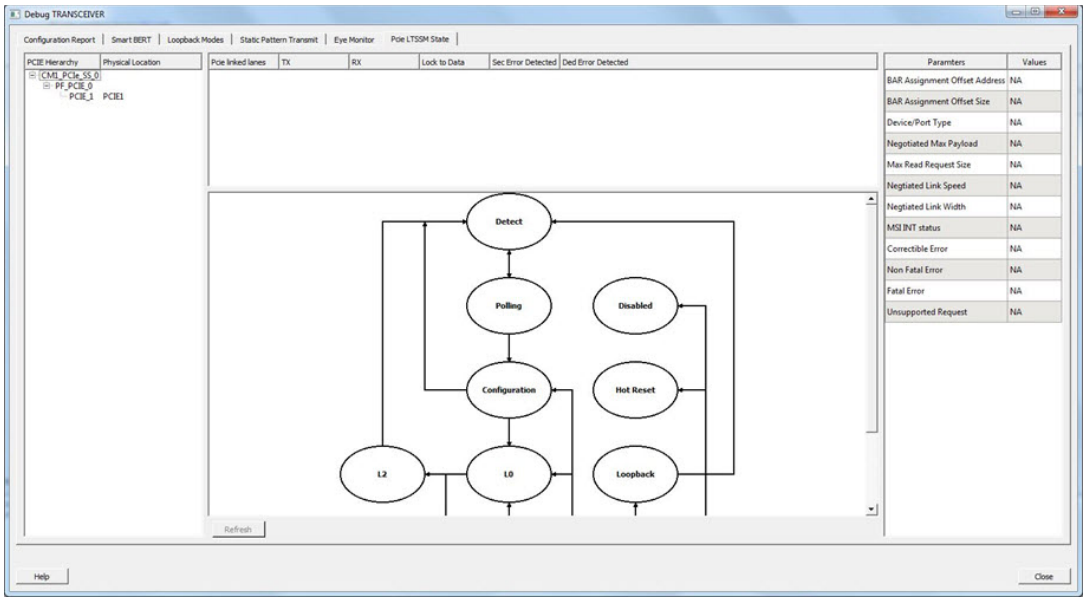


Figure 61 · PCIE Design Hierarchy

Lane Status and Lane Link Error Status

When a PCIE instance is selected (without selecting any other hierarchy level above), the Lane status, SEC Error status, and DED Error status is shown to the right of the PCIE Hierarchy. The logical names of the lanes are also shown. See the following example figure.

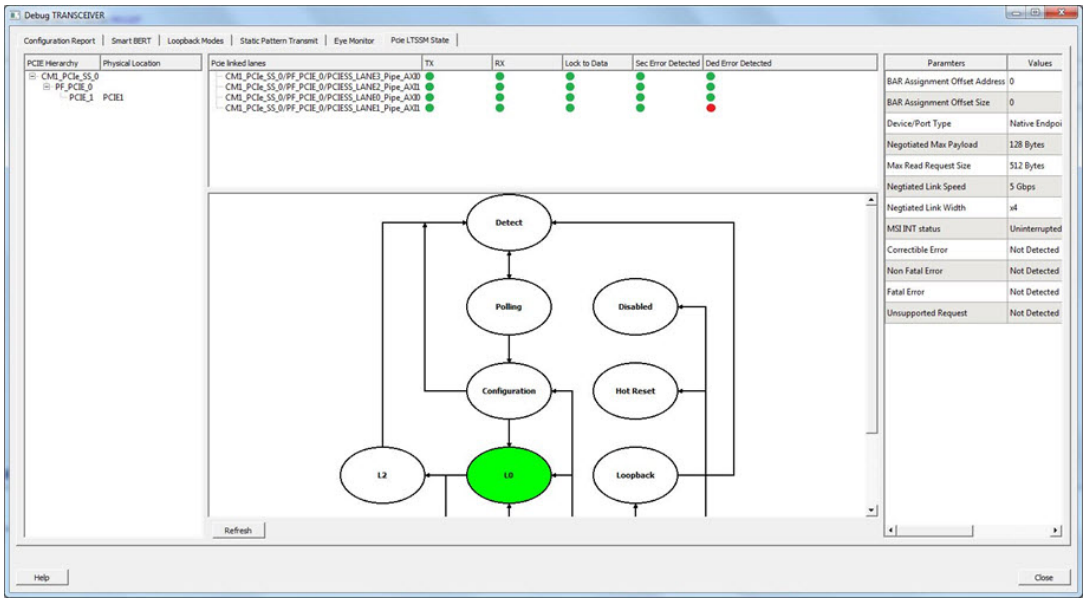


Figure 62 · Lane Status and Lane Link Error Status

LTSSM State Machine

The PCIE LTSSM State page shows the LTSSM state machine to the right of PCIE Design Hierarchy and under the lane status, as shown in the following example figure.

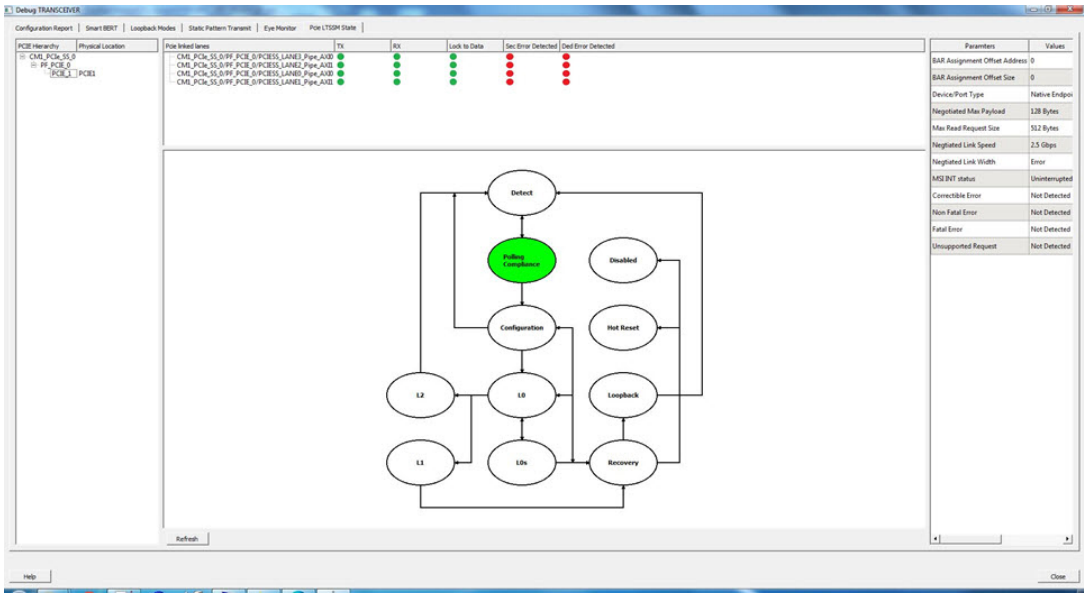


Figure 63 · LTSSM State Machine

When you click a PCIE instance in the PCIE Hierarchy, the active LTSSM state is retrieved from register LTSSM_STATE:PL_LTSSM_OUT, the active state is highlighted in the state machine, and the substate information is shown. The LTSSM state output is also logged in the SmartDebug log window. See the following example figure.

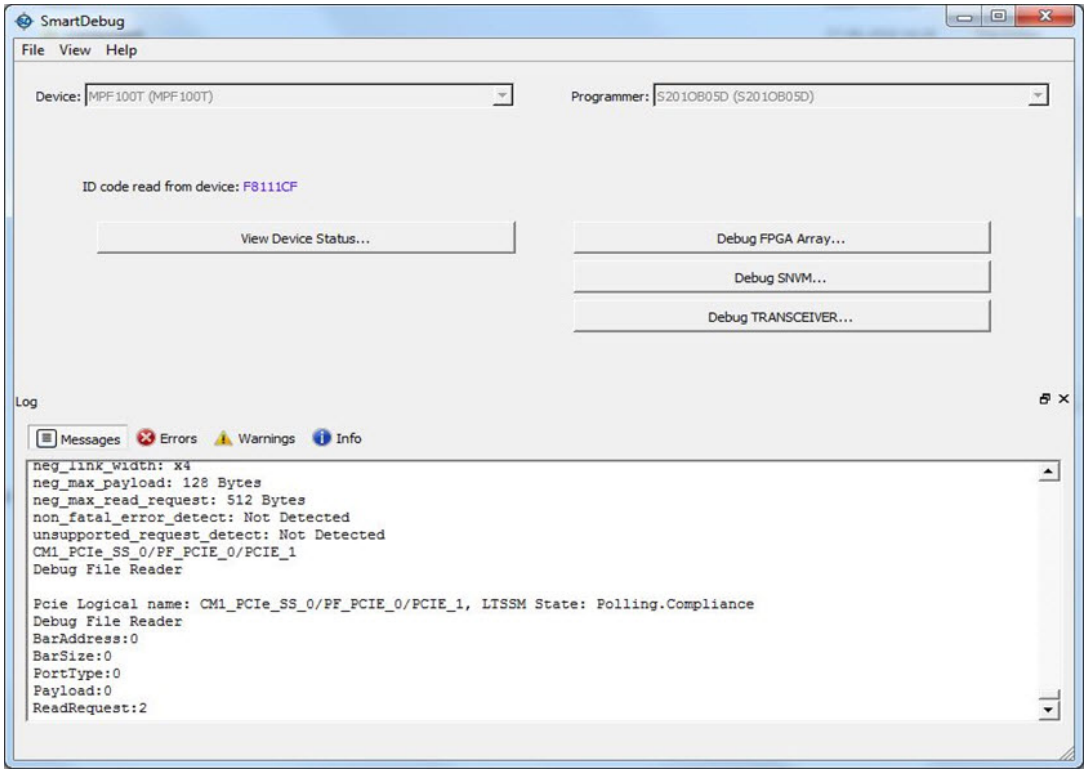


Figure 64 · Active State and Substate Information

Config Space Parameter Information

When you click on a PCIE instance in the PCIE Hierarchy, the config space parameter data is retrieved from the device and displayed to the right of the LTSSM state machine, as shown in the following example figure. When no PCIE instance is selected, or any level of hierarchy instance except for the PCIE instance is selected, the parameter values are displayed as “NA”. If there is an error retrieving parameter data, the value displayed is “Error” in the GUI.

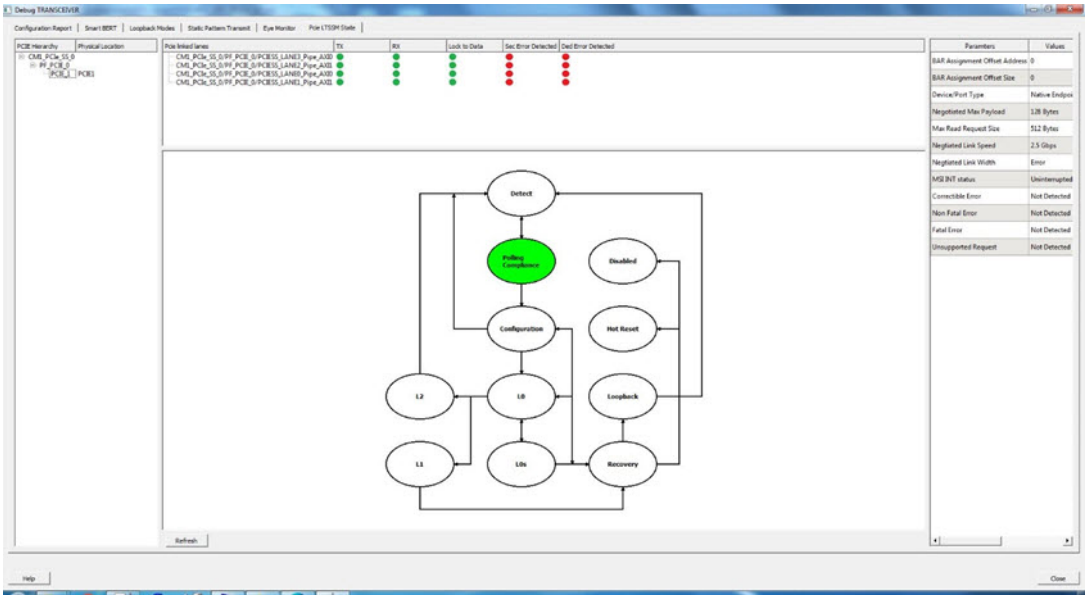


Figure 65 · Config Space Parameter Information

The list of config parameters and values is also logged in the SmartDebug log window, as shown in the following example figure.

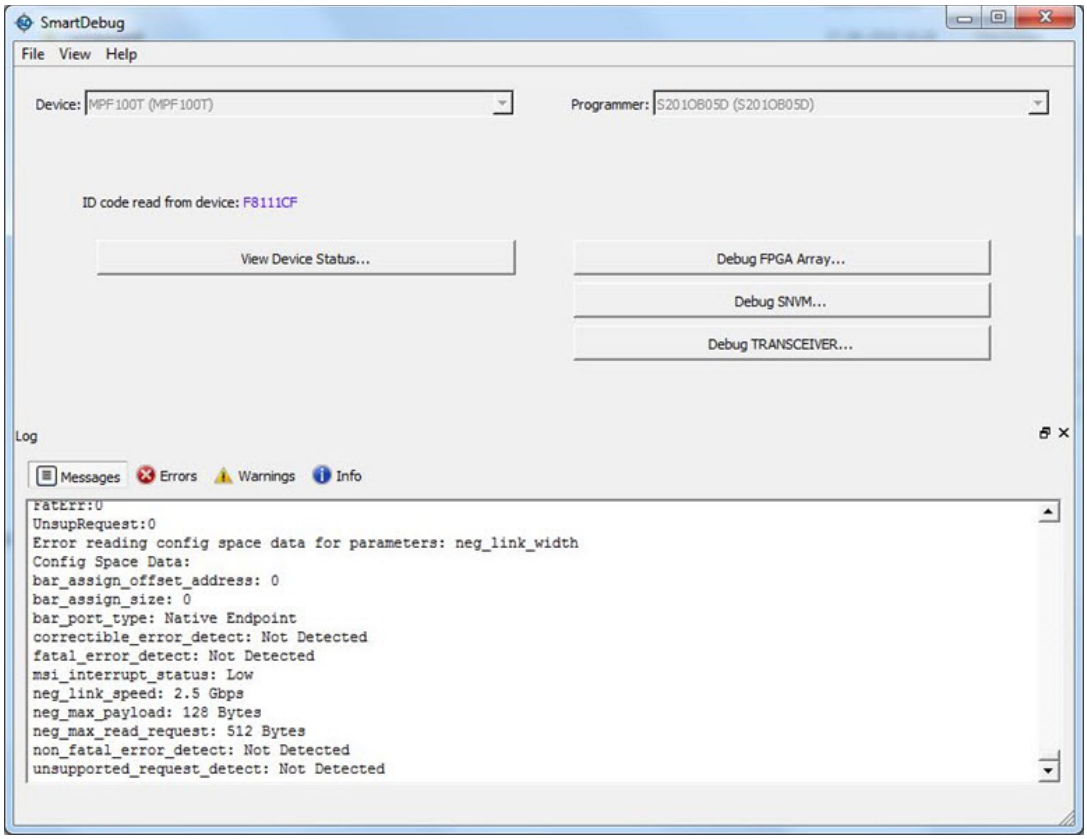


Figure 66 · Config Space Parameter Information in the SmartDebug Log Window

Refresh Button

Click the **Refresh** button to update all PCIE data displayed in the GUI.

Debug uPROM

You can debug clients configured in a design and debug µPROM memory address information with the Debug µPROM feature.

In the main SmartDebug window, click **Debug uPROM**.

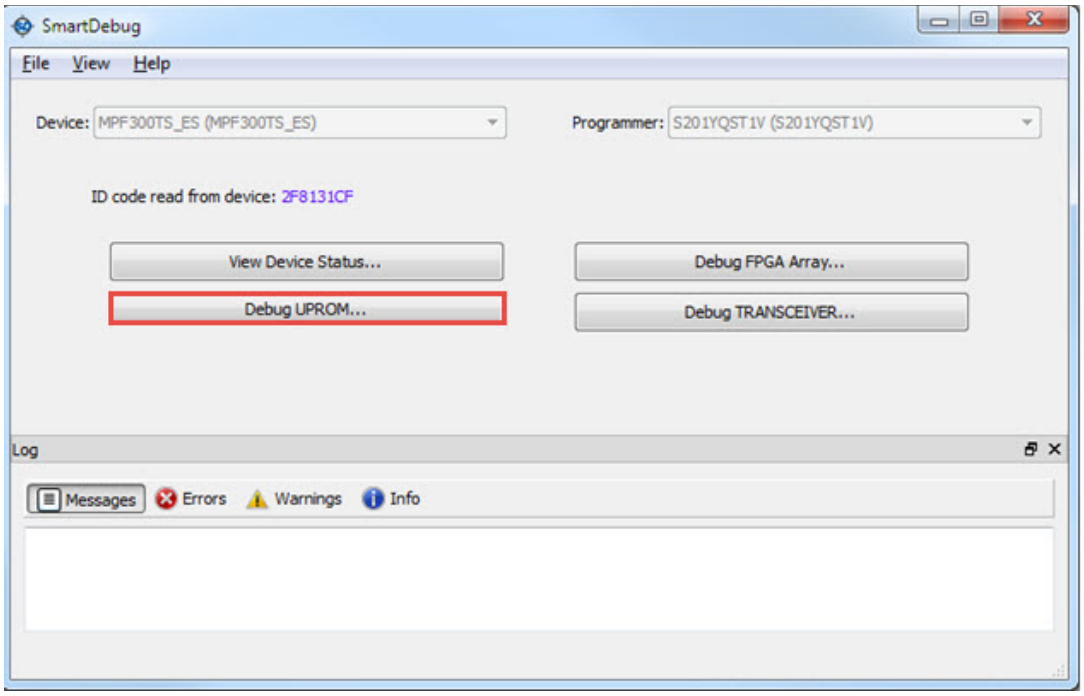


Figure 67 · SmartDebug Window - Debug uPROM

If a µPROM memory block is used in the Libero design, the µPROM Debug window appears.

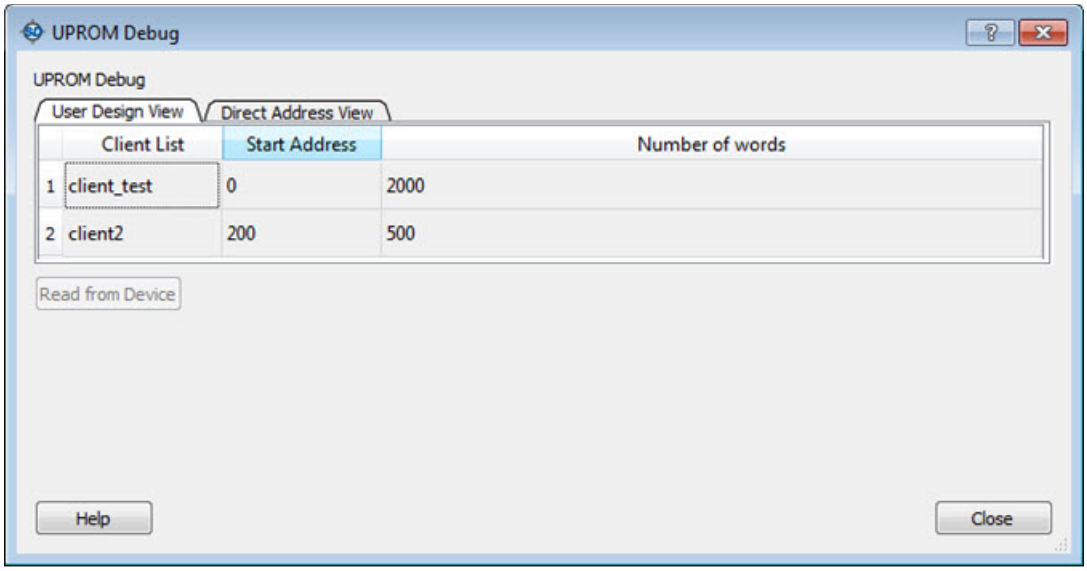
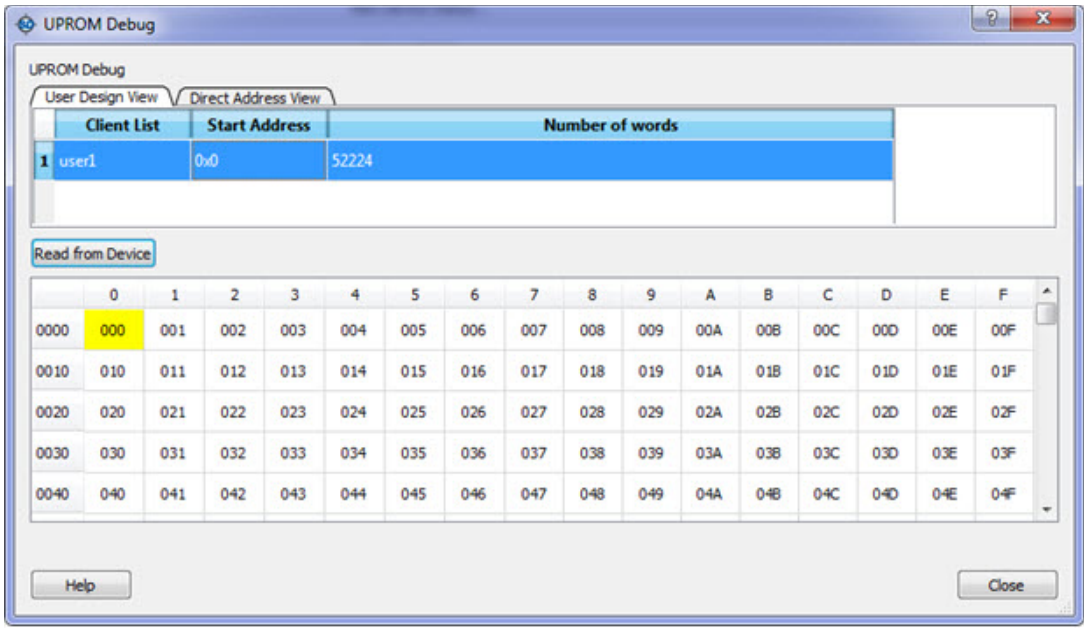


Figure 68 · µPROM Debug Window

User Design View

The User Design View tab in the µPROM Debug window lists all clients configured in the design. Selecting a client in the list enables the **Read from Device** button.

Clicking the **Read from Device** button displays a table showing the data in the location at the selected client address. See the following example.

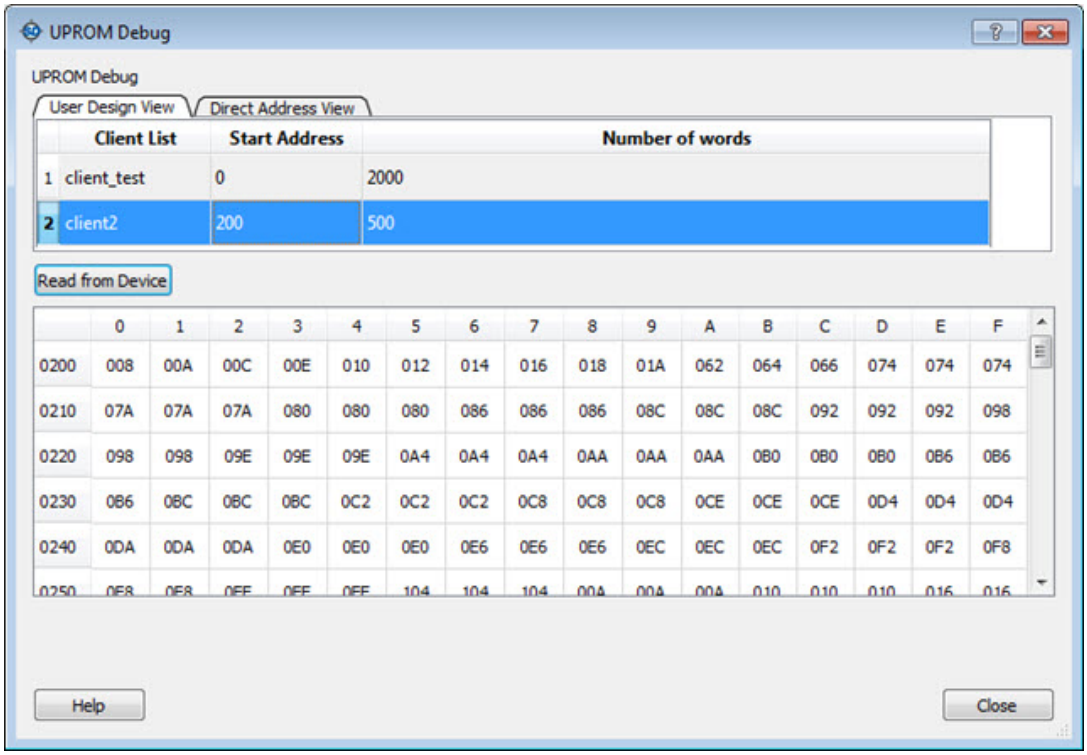


The Client address is associated with *Start Address* and *Number of 9-bit words*. Therefore, the table will contain as many locations as the number of 9-bit words.

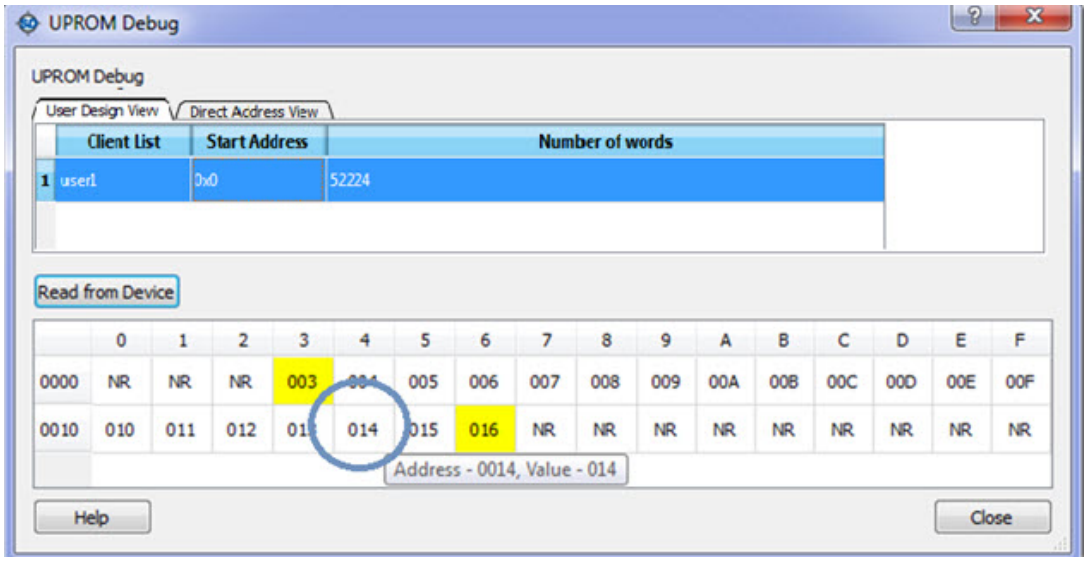
In the example above, *Number of 9-bit words* is 52224, so 52224 words will be shown in the table.

Column headers are numbered 0 to F in hexadecimal format, representing 16 words in a row.

Row addresses begin with a word address associated with *Start Address*. For example, if the *Start Address* is 0x15 (hex), the starting row has an address of 0x0010.



You can hover over a cell to see its address and value, as shown in the following example.



Direct Address View

The Direct Address View tab in the µPROM Debug window provides access to µPROM memory. You can read a part of a client or more than one client by specifying the *Start Address* and *Number of 9-bit words*.

Start Address - hexadecimal value (0 -9, A-F, upper/lower case)

Values are validated and errors are indicated by a red "STOP" icon ( ). The error message displays when you hover over the icon.

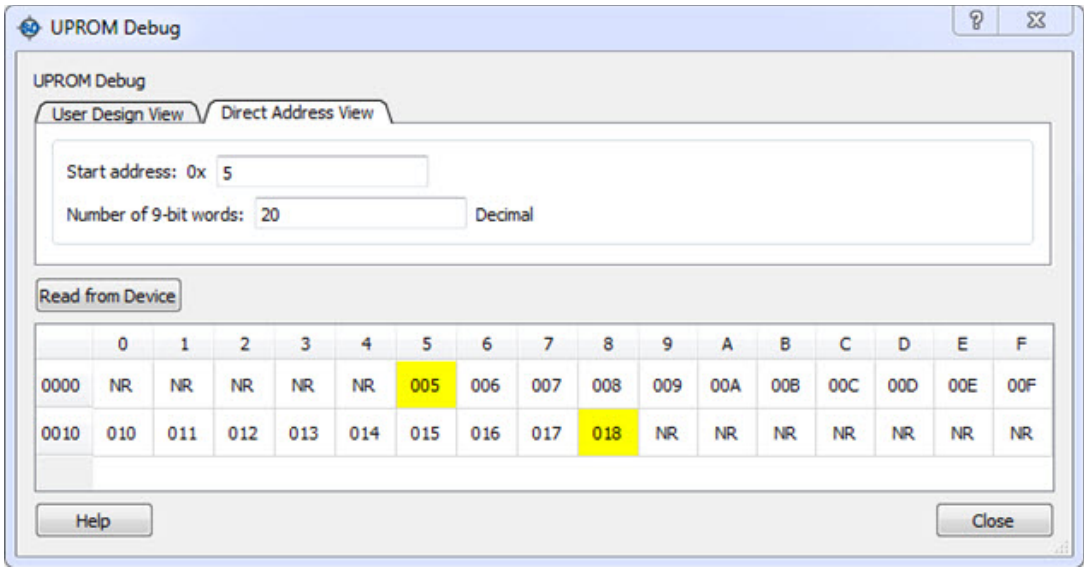
Number of 9-bit words - positive integer value

Values are validated and errors are indicated by a red "STOP" icon ( ). The error message displays when you hover over the icon.

Read from Device - Disabled until valid values are entered in the fields.

Invalid or blank values are indicated by a red "STOP" icon ( ). The error message displays when you hover over the icon.

Note: If the word falls within the 16 words that are placed in a row, the start location and the end location are highlighted in the row to show the starting point of the data. All preceding locations show 'NR' (Not Read). See the following example.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	NR	NR	NR	NR	NR	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0010	010	011	012	013	014	015	016	017	018	NR	NR	NR	NR	NR	NR	NR

Notes:

When one field is entered, both fields are validated to enable the Read from Device button.

If fields change after enabling Read from Device, values are validated again and Read from Device may be disabled if invalid values are entered.

If the µPROM Debug window is closed and reopened, the session is retained. The µPROM Debug session is lost only if the main SmartDebug window is closed.

Demo Mode

Debug µPROM is supported in Demo Mode. The User Design View and Direct Address View are supported, and data from device initialization and configurators is shown.

SmartDebug Tcl Commands

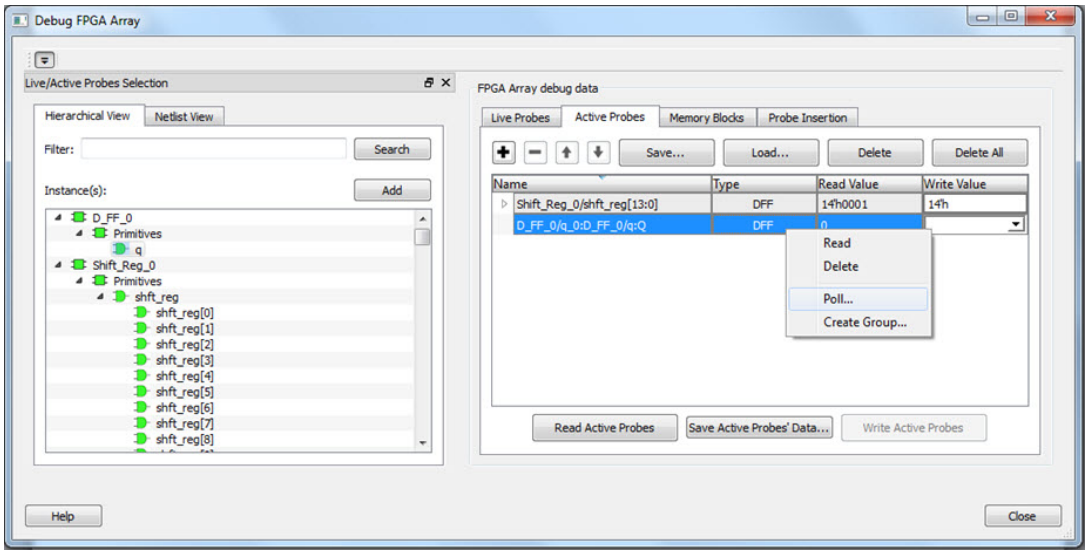
See the PolarFire FPGA Tcl Commands Reference Guide for detailed information about the SmartDebug Tcl Commands supported in PolarFire.

Frequently Asked Questions

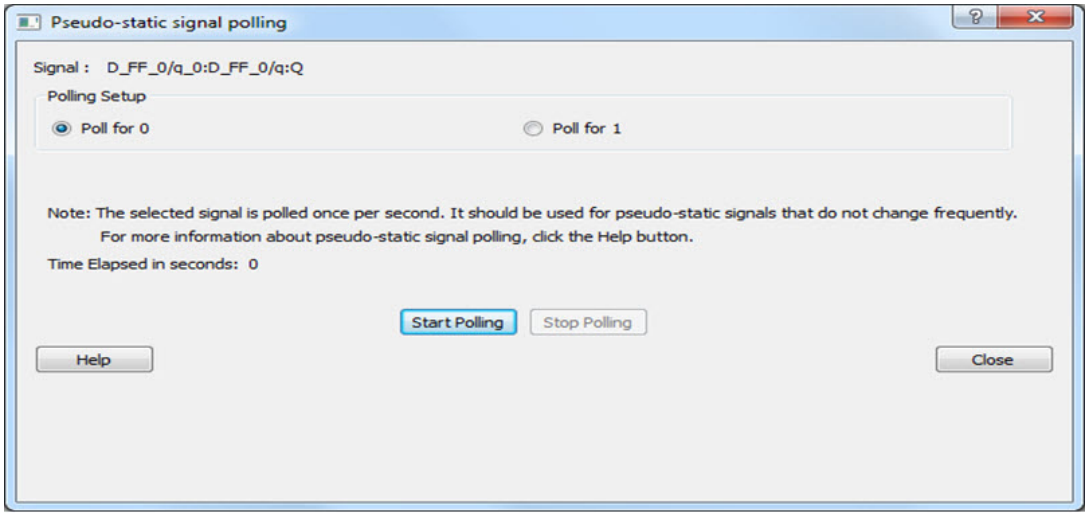
How do I monitor a static or pseudo-static signal?

To monitor a static or pseudo-static signal:

1. Add the signal to the **Active Probes** tab.
2. Select the signal in the **Active Probes** tab, right-click, and choose **Poll...**.



3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click **Start Polling**.

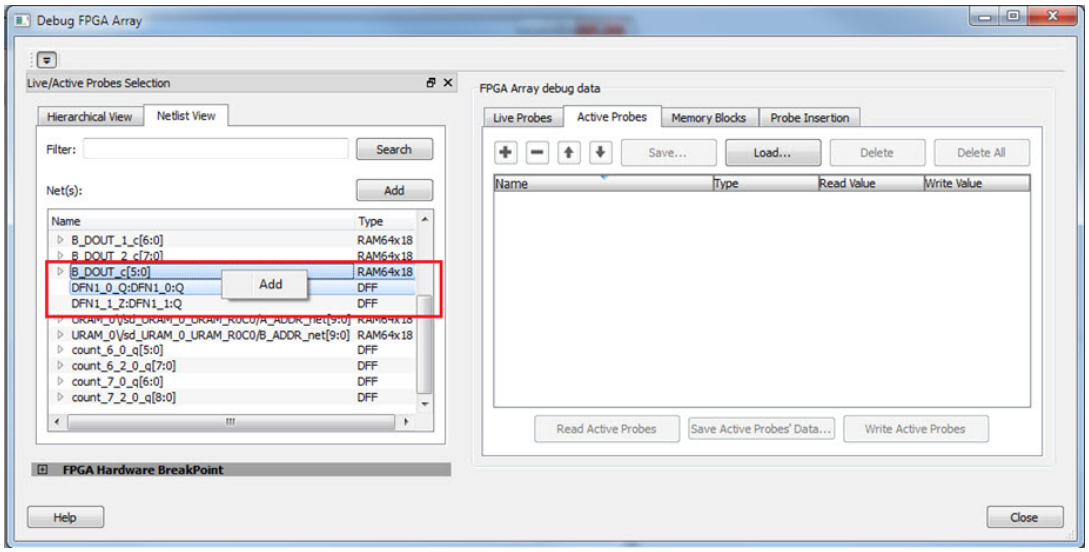


How do I force a signal to a new value?

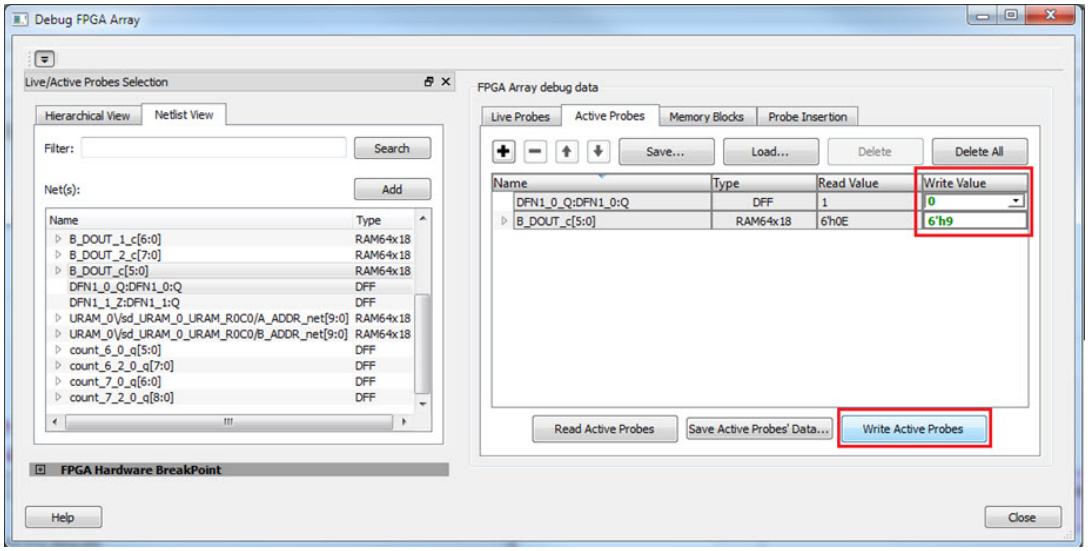
To force a signal to a new value:

1. In the SmartDebug window, click **Debug FPGA Array**.
2. Click the **Active Probes** tab.

3. Select the signal from the selection panel and add it to Active Probes tab.



1. Click **Read Active Probe** to read the value.
2. In the Write Value column, enter the value to write to the signal and then click **Write Active Probes**.

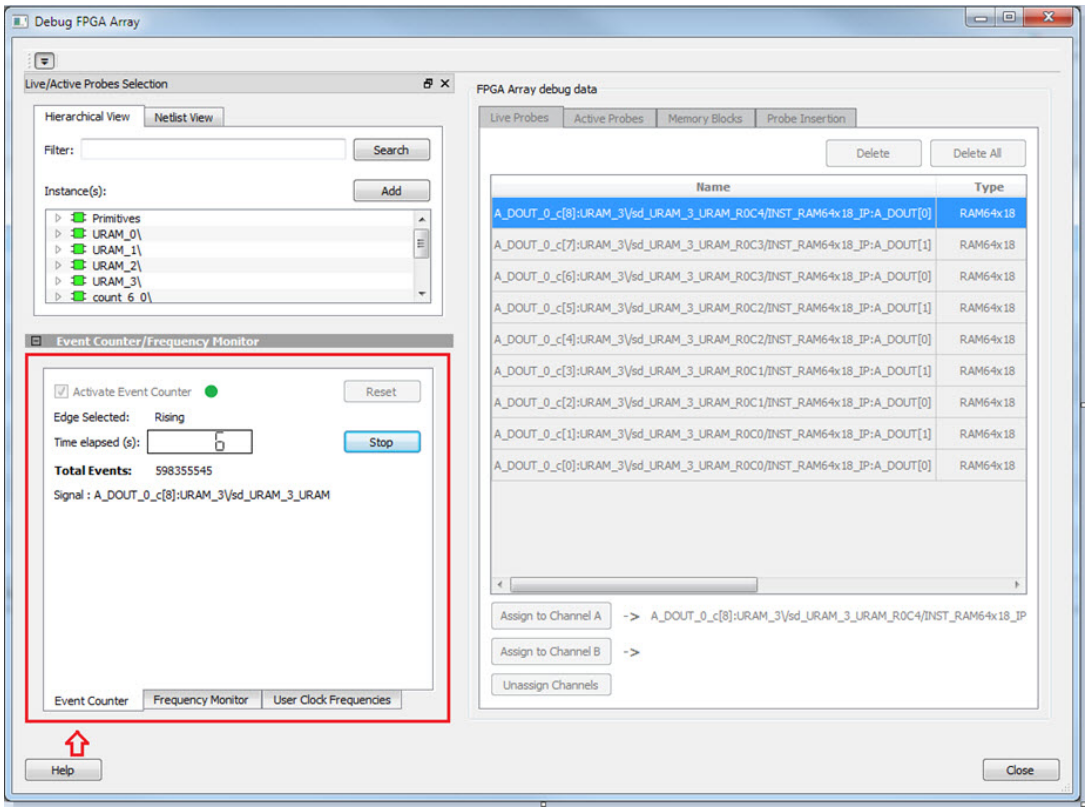


How do I count the transitions on a signal?

If FHB IP is auto-instantiated in the design, you can use the Event Counter in the **Live Probes** tab to count the transitions on a signal.

To count the transitions on a signal:

1. Assign the desired signal to Live Probe Channel A.
2. Click the **Event Counter** tab and check the Activate Event Counter checkbox.



See Also

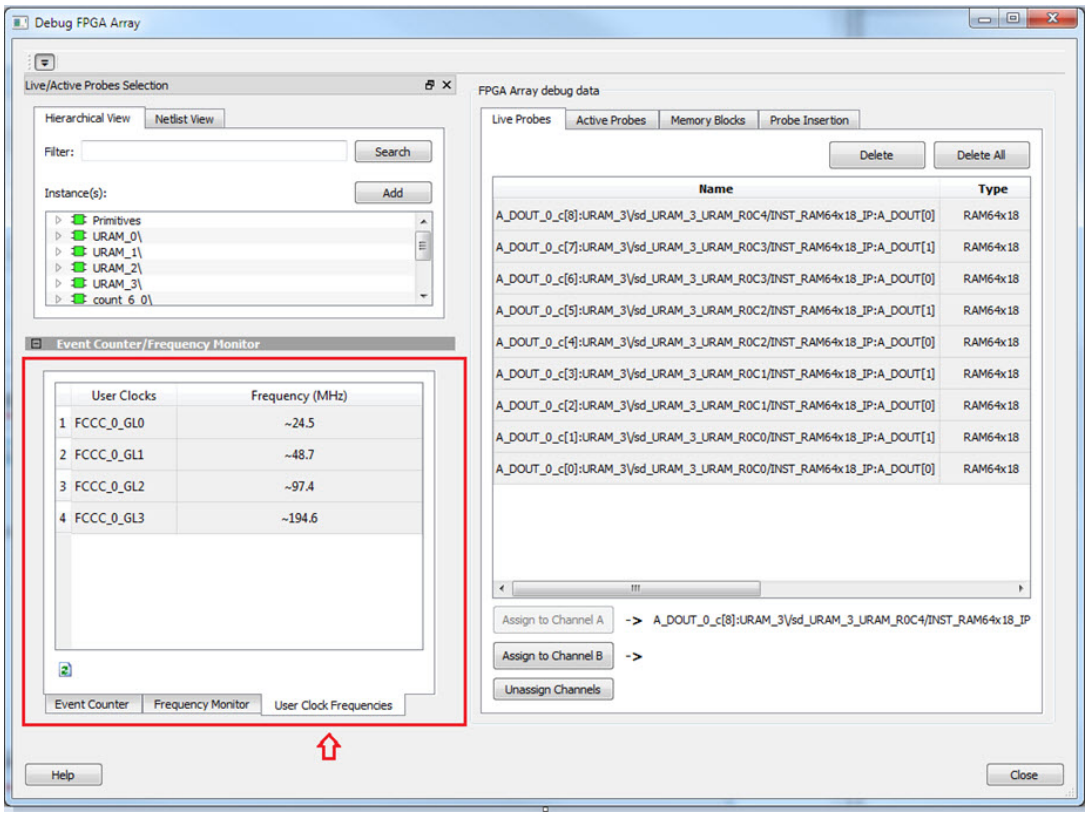
"Event Counter" on page 32

How do I monitor or measure a clock?

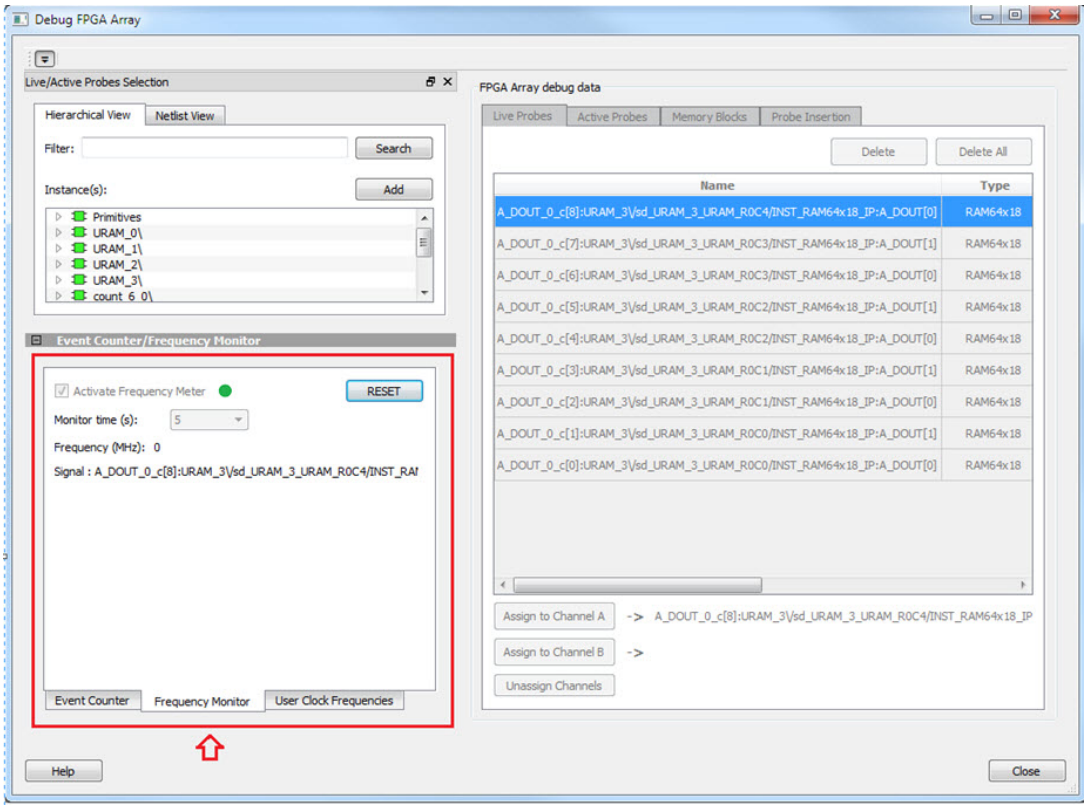
You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in Project Settings dialog box.

In the **Live Probe** tab, SmartDebug allows you to:

1. Measure all the FABCCC GL clocks by clicking the **User Clock Frequencies** tab, as shown in the figure below.



2. Monitor frequencies of any probe points by:
    a. Assigning the desired signal to Live Probe Channel A.
    b. Selecting the **Frequency Monitor** tab as shown in the following figure and checking the Activate Frequency Meter checkbox.



How do I perform simple SmartBERT tests?

You can perform SmartBERT tests using the Debug Transceiver option in SmartDebug.

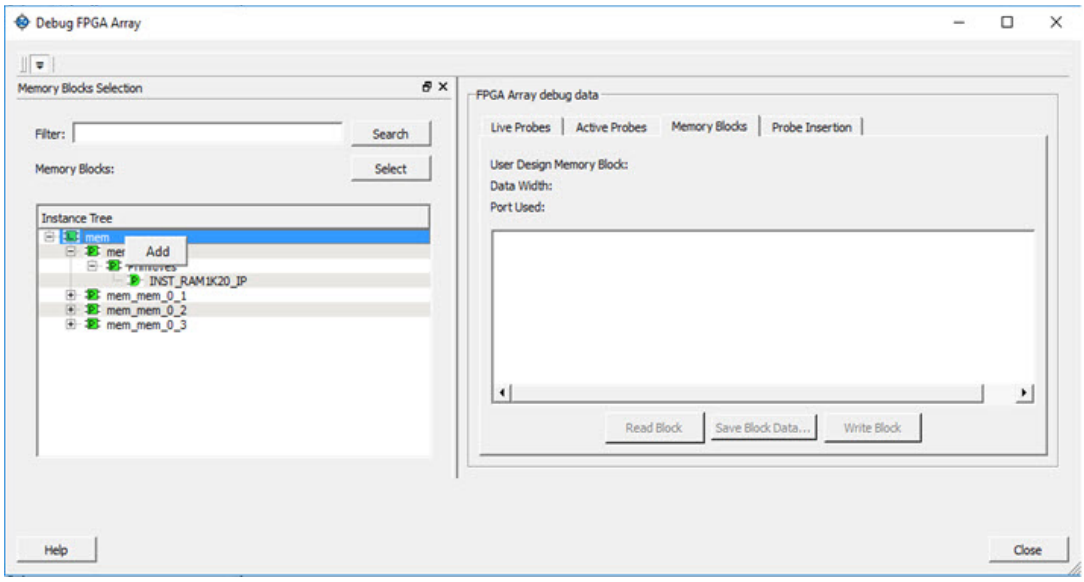
To perform a SmartBERT test, in the SmartBERT page of the Debug Transceiver dialog box, select to run a PRBS test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see "SmartBERT" on page 55 .

To perform a SmartBERT test, in the Smart BERT page of the Debug Transceiver dialog box, select your options and click **Start** to run a Smart BERT test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see "SmartBERT" on page 55.

How do I read LSRAM or USRAM content?

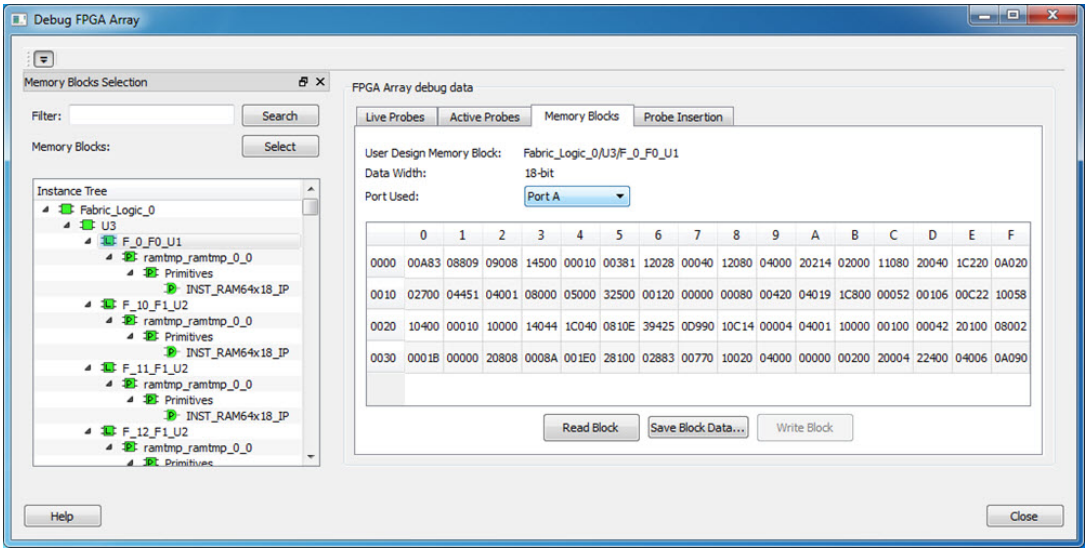
To read RAM content:

1. In the Debug FPGA Array dialog box, click the **Memory Blocks** tab.
2. Select the memory block to be read from the selection panel on the left of the window.



An "L" in the icon next to the block name indicates that it is a logical block, and a "P" in the icon indicates that it is a physical block. A logical block displays three fields in the Memory Blocks tab: User Design Memory Blocks, Data Width, and Port Used. A physical block displays two fields in the Memory Blocks tab: User Design Memory Block and Data Width.

3. Add the block in one of the following ways:
    a. Click **Select**.
    b. Right-click and choose **Add**.
    c. Drag the block to the **Memory Blocks** tab.
4. Click **Read Block** to read the content of the block.



See Also

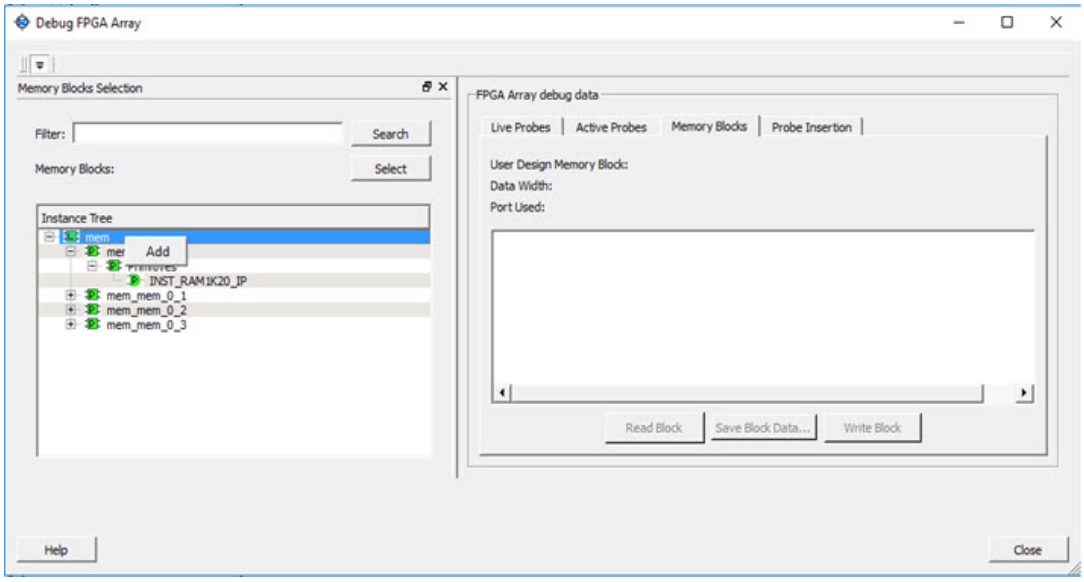
"Memory Blocks " on page 24

How do I change the content of LSRAM or USRAM?

To change the content of LSRAM or USRAM:

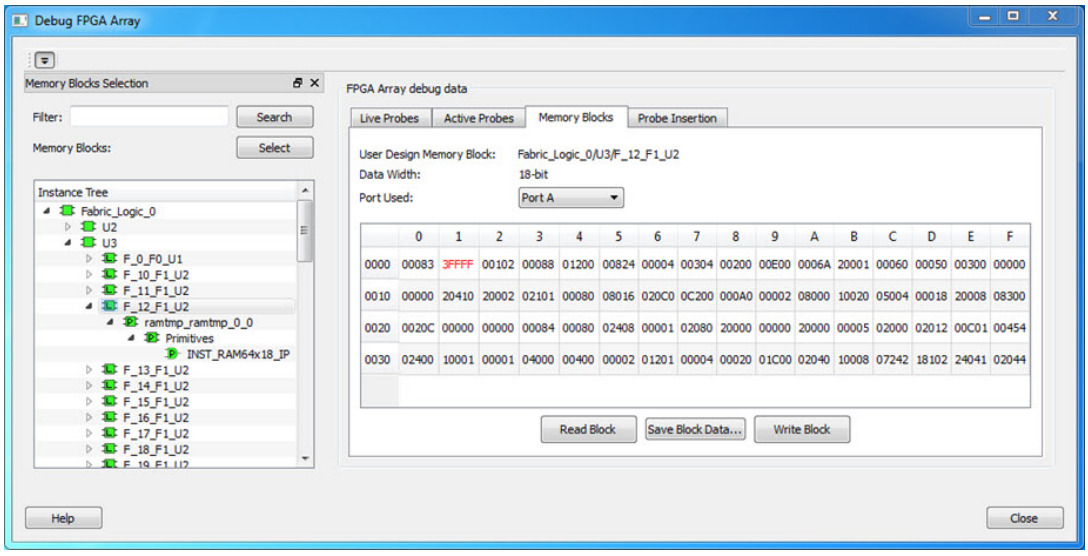
1. In the SmartDebug window, click **Debug FPGA Array**.

2. Click the **Memory Blocks** tab.
3. Select the memory block from the selection panel on the left of the window.



An "L" in the icon next to the block name indicates that it is a logical block, and a "P" in the icon indicates that it is a physical block. A logical block displays three fields in the Memory Blocks tab: User Design Memory Blocks, Data Width, and Port Used. A physical block displays two fields in the Memory Blocks tab: User Design Memory Block and Data Width.

4. Add the memory block in one of the following ways:
    a. Click **Select**.
    b. Right-click and choose **Add**.
    c. Drag the block to the **Memory Blocks** tab.
5. Click **Read Block**. The memory content matrix is displayed.
6. Select the memory cell value that you want to change and update the value.
7. Click **Write Block** to write to the device.



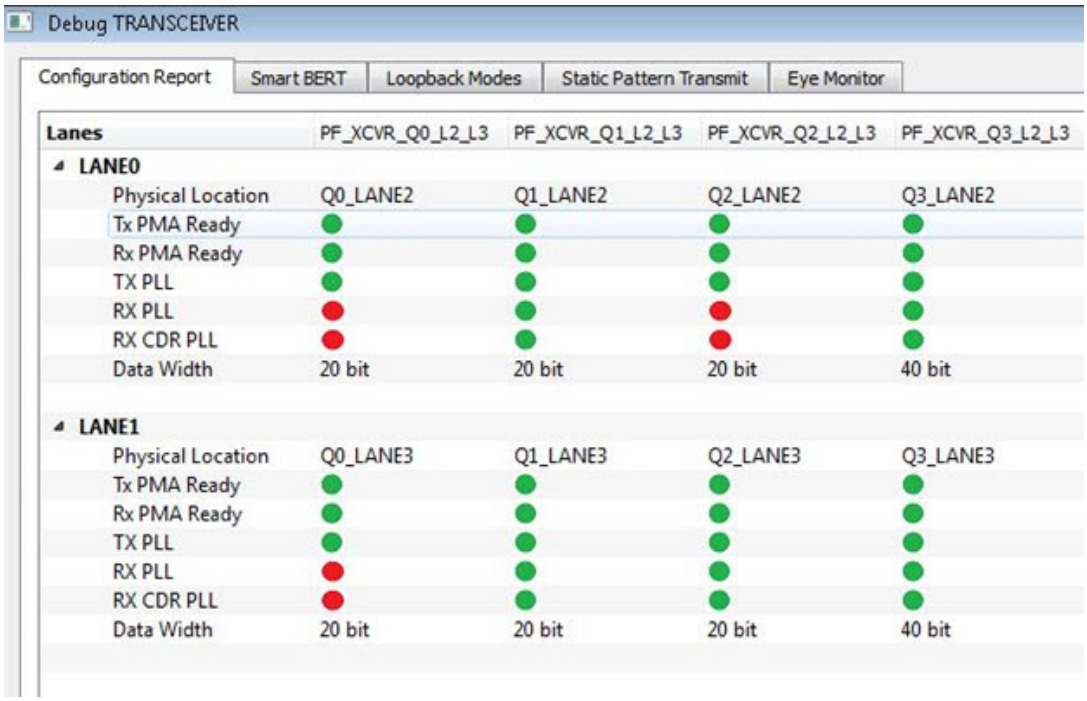
See Also

"Memory Blocks " on page 24

How do I read the health check of the Transceiver?

You can read the transceiver health check using the following Debug Transceiver options:

1. Review the **Configuration Report**, which returns Tx PMA Ready, Rx PMA Ready, TxPLL status, and RxPLL status. For the transceiver to function correctly, all four should be green. The Configuration Report can be found in the Debug TRANSCEIVER dialog box under Configuration Report. See "Debug Transceiver" on page 51.



2. Run the SmartBERT Test, with EQ-NEAR END checked or with external loopback connection from Tx to Rx on selected lanes. This should result in 0 errors in the Cumulative Error Count column. See "SmartBERT" on page 55.