UG0638
User Guide
SmartFusion2, IGLOO2, RTG4 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 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

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.**

5-02-00638-5/08.19

Table of Contents

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 (eNVM), SRAM, SERDES, and probe capabilities. Microsemi SmartFusion2 System-on-chip (SoC) field programmable gate array (FPGA), IGLOO2 FPGA, and RTG4 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, SERDES) 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: SmartFusion2, IGLOO2, and RTG4

Programmers: FlashPro3, FlashPro4, FlashPro5, FlashPro6

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

Supported Tools

The following table lists device family support for SmartDebug tools.

SmartDebug Support per Device Family	SmartFusion2	IGLOO2	RTG4
Live Probes	X	X	X
Active Probes	X	X	X
Memory Debug	X	X	X
Probe Insertion (available only through Libero flow)	X	X	X
View Flash Memory Content	X	X	
Debug SERDES	X	X	X
FPGA Hardware Breakpoint (Needs FHB Auto Instantiation)	X	X	X
Event Counter (Needs FHB Auto Instantiation)	X	X	X
Frequency Monitor (Needs FHB Auto Instantiation)	X	X	X

Note: "X" indicates the tool is supported.

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 (FlashROM, Security Settings, Embedded Flash Memory (NVM)).

See Using SmartDebug for an overview of the use flow.

You can use the debugger to:

- Get device status and view diagnostics
- Use the Embedded Flash Memory Debug GUI to read out and compare your content with your original files

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.

Running SmartDebug in Demo Mode

Demo mode allows you to experience SmartDebug features (Active Probe, Live Probe, Memory Blocks, SERDES) 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.

If programming hardware is not detected when you invoke SmartDebug, you will see the following.

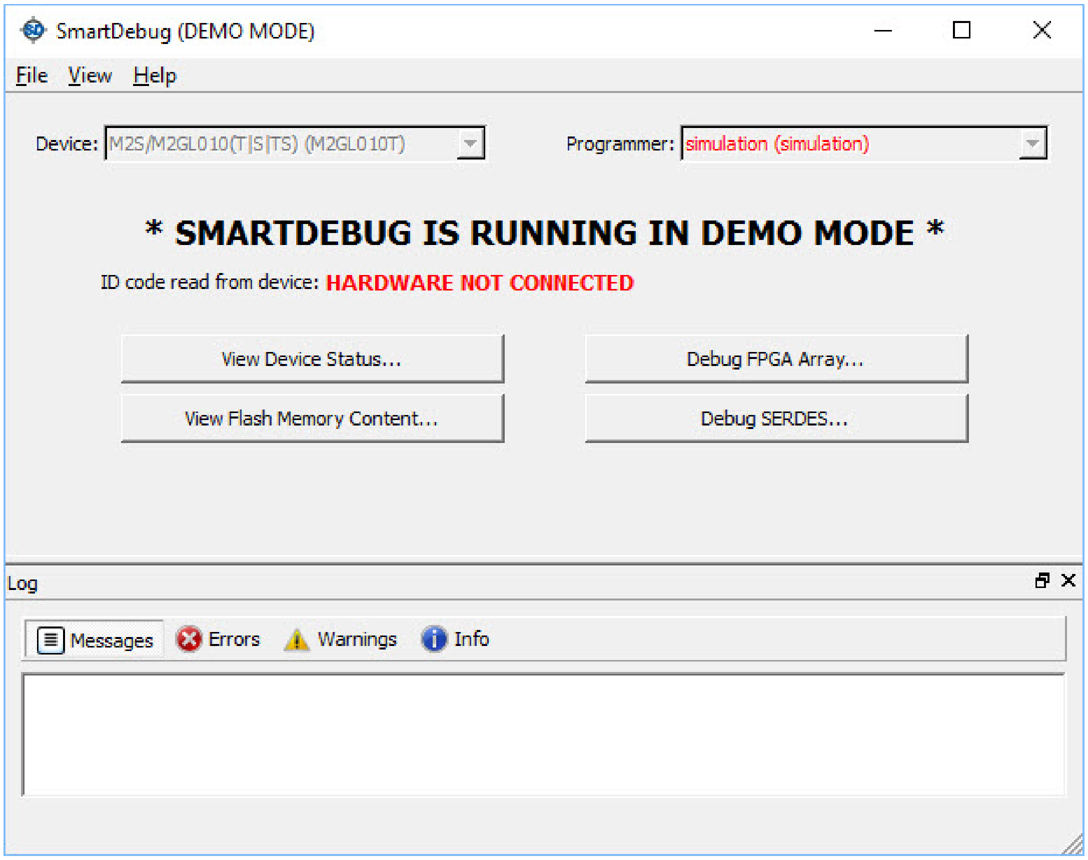


Figure 1 · SmartDebug in Demo Mode Example

See Also

[Active Probes]
[Live Probes]
[Memory Blocks]
[Debug SERDES - Loopback Test]
[Debug SERDES - PRBS Test]

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.

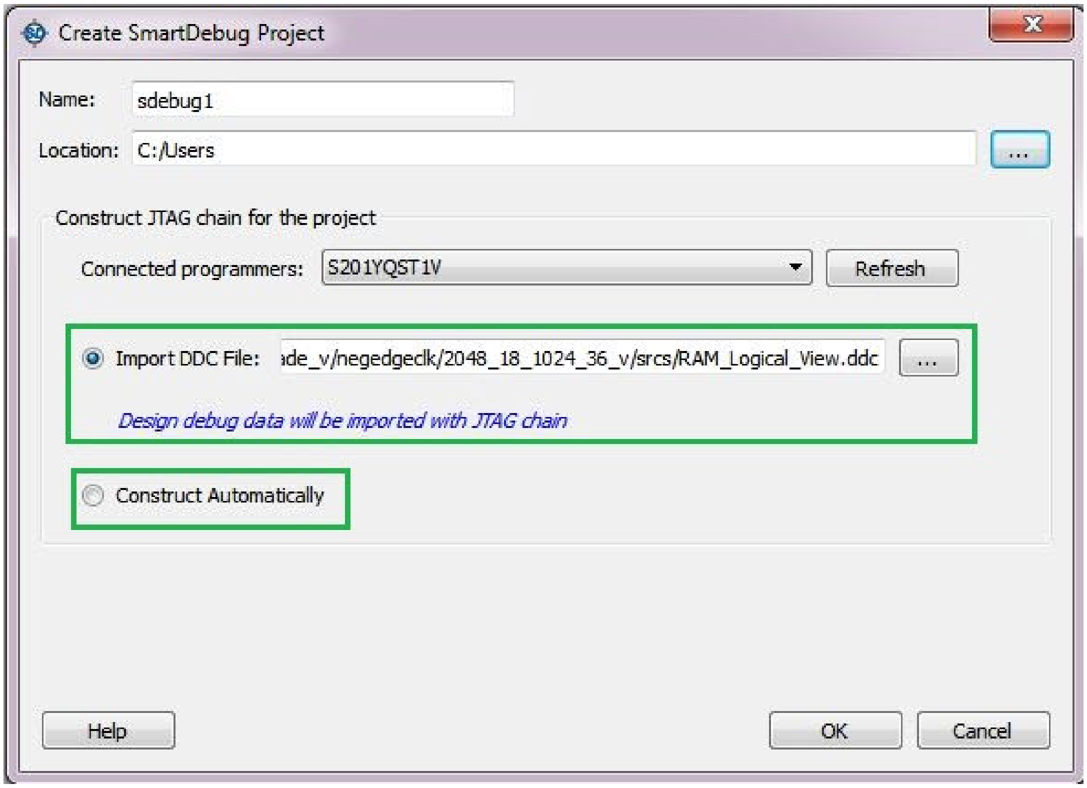


Figure 2 · Create SmartDebug Project Dialog Box

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—for example, M2S090T, M2S090TS, and so on), 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

Export SmartDebug Data (from Libero)

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

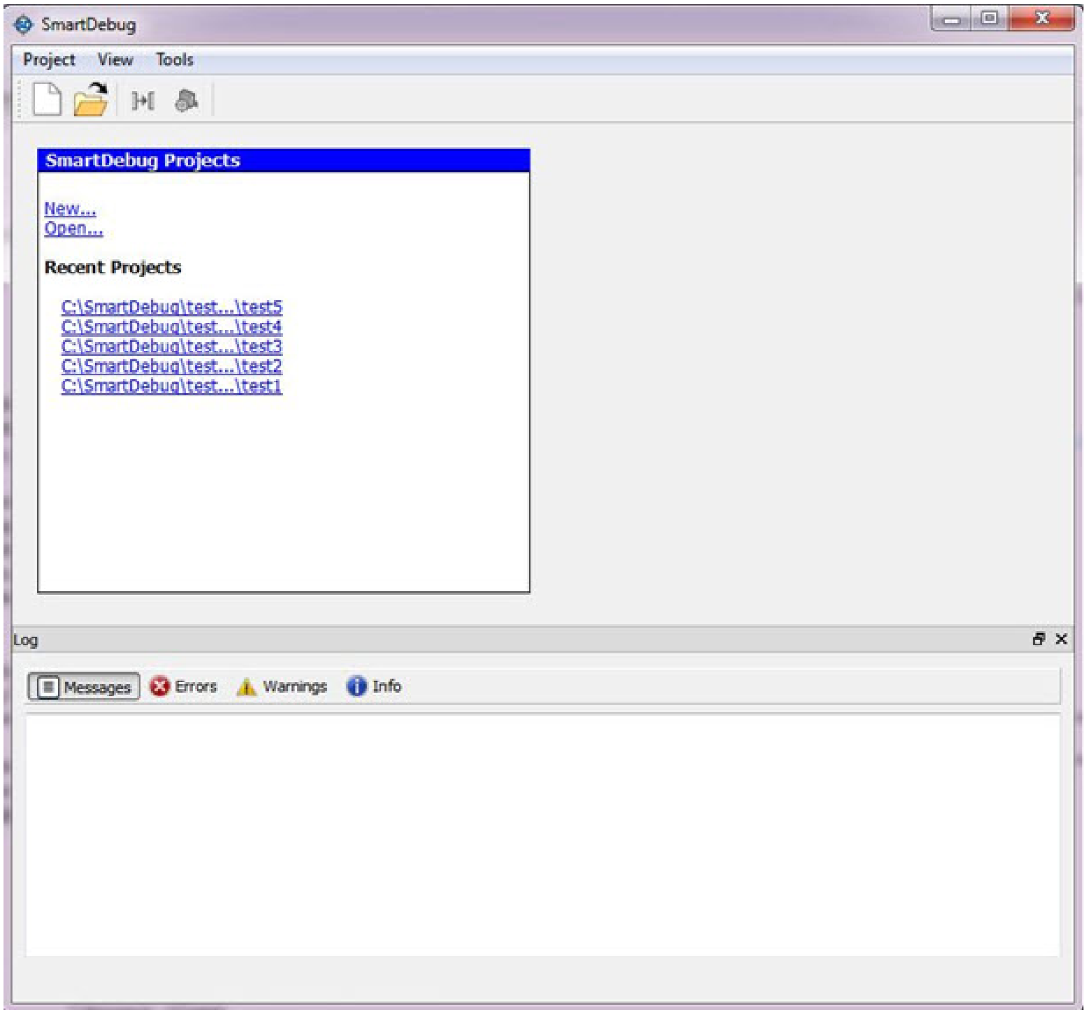


Figure 3 · Standalone SmartDebug Main Window

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.

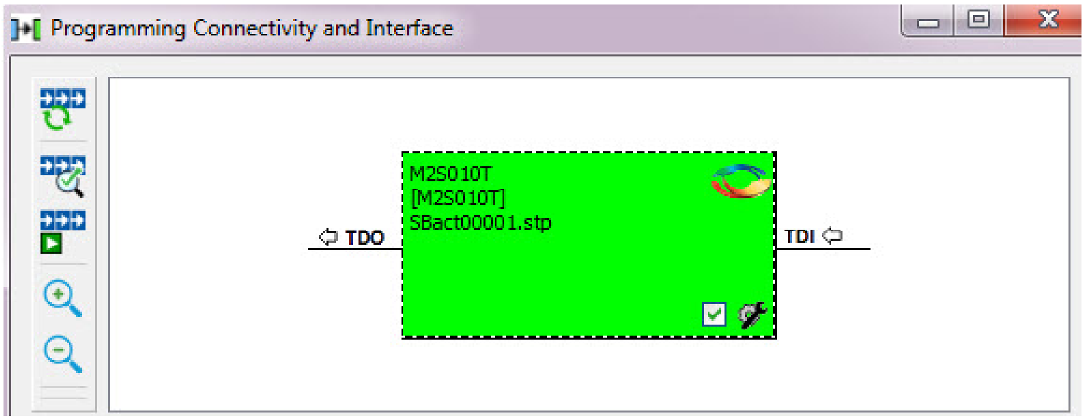


Figure 4 · Programming Connectivity and Interface Dialog Box – Project created using Import from DDC File

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.

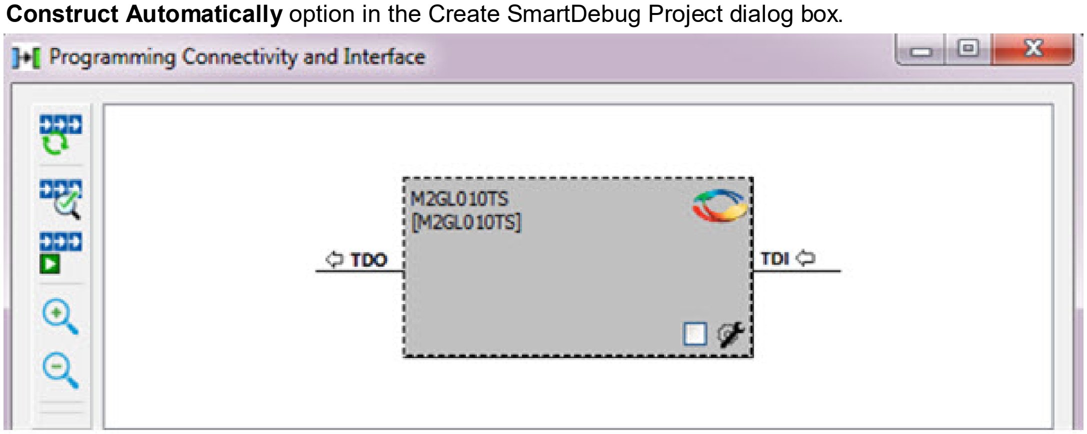


Figure 5 · Programming Connectivity and Interface window – Project created using Construct Automatically

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. For example, if M2S090 and M2GL010 devices are connected in the chain, and the M2S090 device is to be programmed and t

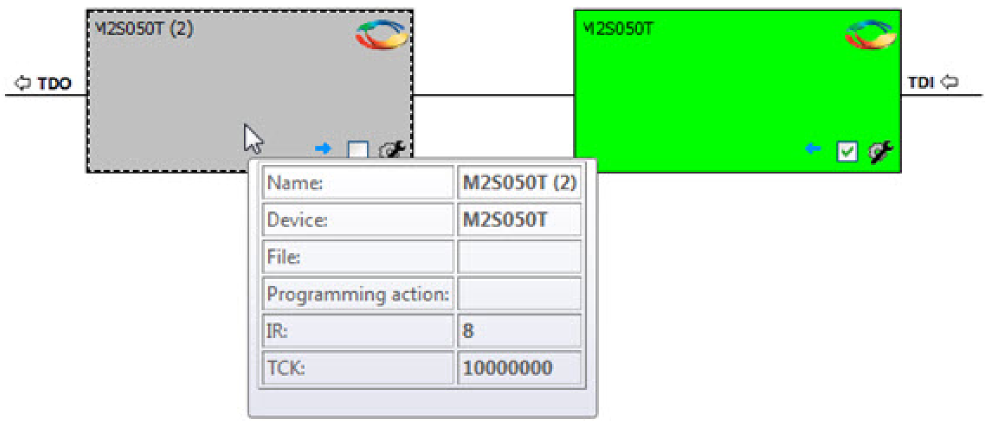
  he M2GL010 device is to be erased, both actions can be done at the same time using the Run Programming Action option.

- **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.



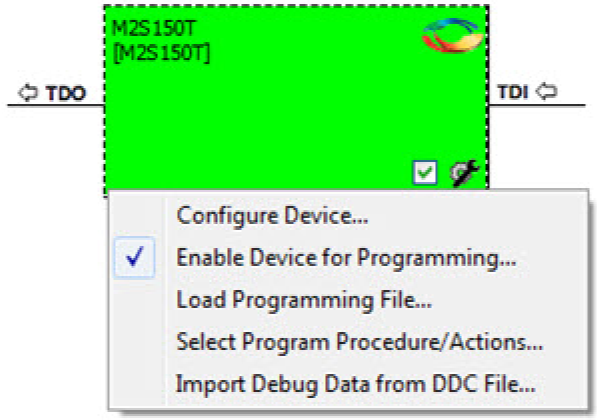
Device Chain Details

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 SERDES, 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

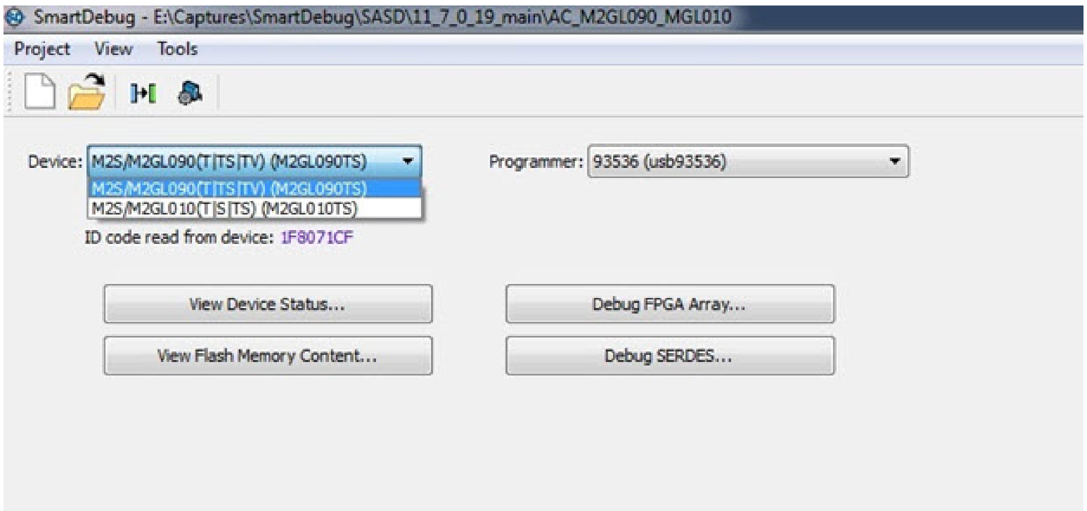
The debug context of the current session is reset for the following actions in Programming Connectivity and Interface:

- 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.

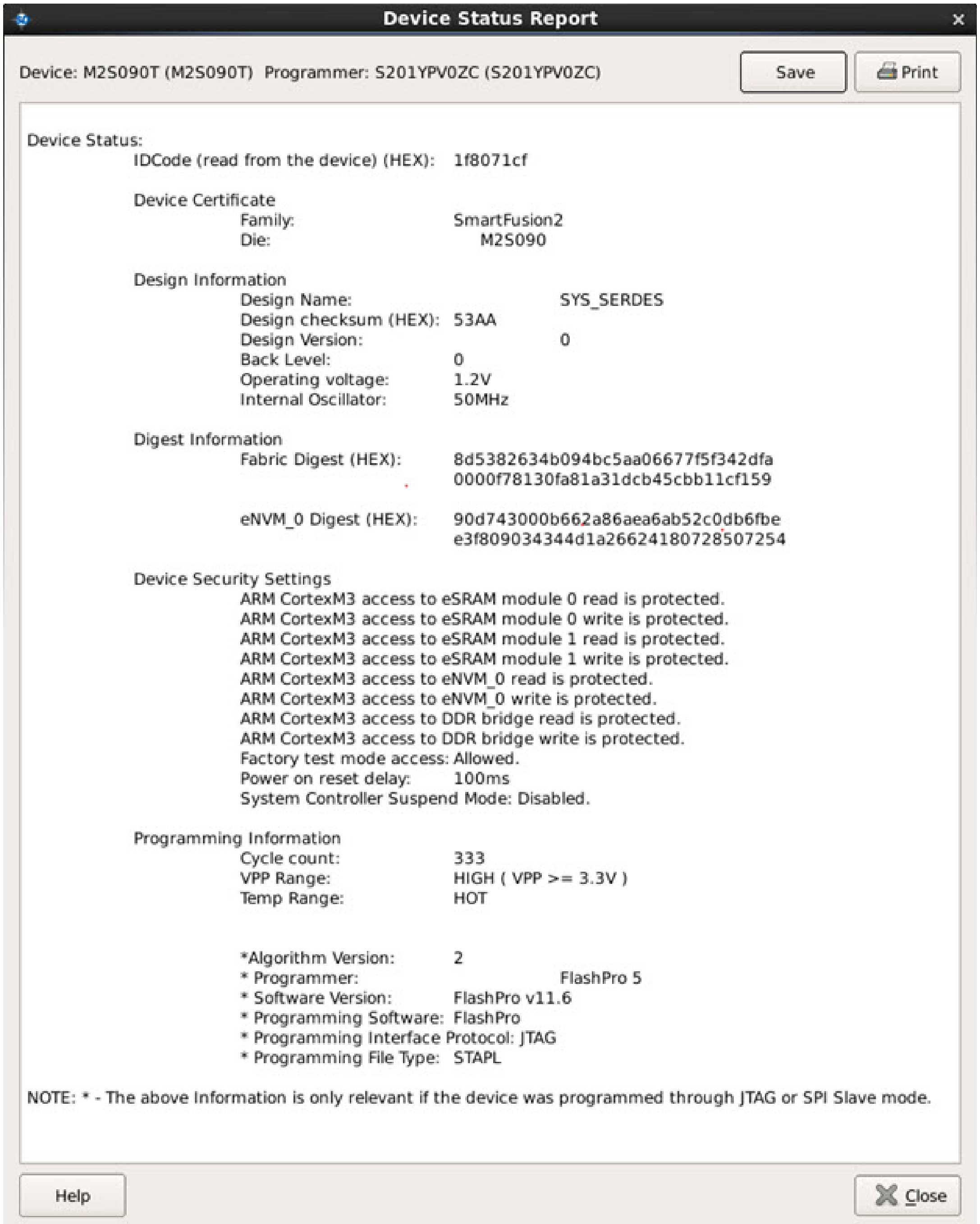


Figure 6 · Device Status Report

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
- Back Level (SmartFusion2 and IGLOO2 only)
- Operating Voltage (SmartFusion2 and IGLOO2 only)
- Internal Oscillator (SmartFusion2 and IGLOO2 only)

Digest Information

Digest Information displays Fabric Digest, eNVM_0 Digest and eNVM_1 Digest (for M2S090 and M2S150 devices only) computed from the device during programming. eNVM Digest is shown when eNVM is used in the design.

Device Security Settings

Note: For RTG4 devices, only Lock Bit information is displayed.

Device Security Settings indicate the following:

- Factory test mode access
- Power on reset delay
- System Controller Suspend Mode

In addition, if custom security options are used, Device Security Settings indicate:

- User Lock segment is protected
- User Pass Key 1/2 encrypted programming is enforced for the FPGA Array
- User Pass Key 1/2 encrypted programming is enforced for the eNVM_0 and eNVM_1
- SmartDebug write access to Active Probe and AHB mem space
- SmartDebug read access to Active Probe, Live Probe & AHB mem space
- UJTAG access to fabric

Programming Information

Programming Information displays the following:

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

Embedded Flash Memory (NVM) Content Dialog Box (SmartFusion2 and IGLOO2 Only)

The NVM content dialog box is divided into two sections:

- View content of Flash Memory pages (as shown in the figure below)
- Check page status and identify if a page is corrupted or if the write count limit has exceeded the 10-year retention threshold

Choose the eNVM page contents to be viewed by specifying the page range (i.e., start page and the end page) and click **Read from Device** to view the values.

You must click **Read from Device** each time you specify a new page range to update the view.

Specify a page range if you wish to examine a specific set of pages. In the Retrieved Data View, you can enter an Address value (such as 0010) in the Go to Address field and click the corresponding button to go directly to that address. Page Status information appears to the right.

Contents of Page Status

- ECC1 detected and corrected
- ECC2 detected
- Write count of the page
- If write count has exceeded the threshold
- If the page is used as ROM (first page lock)
- Overwrite protect (second page lock)
- Flash Freeze state (deep power down)

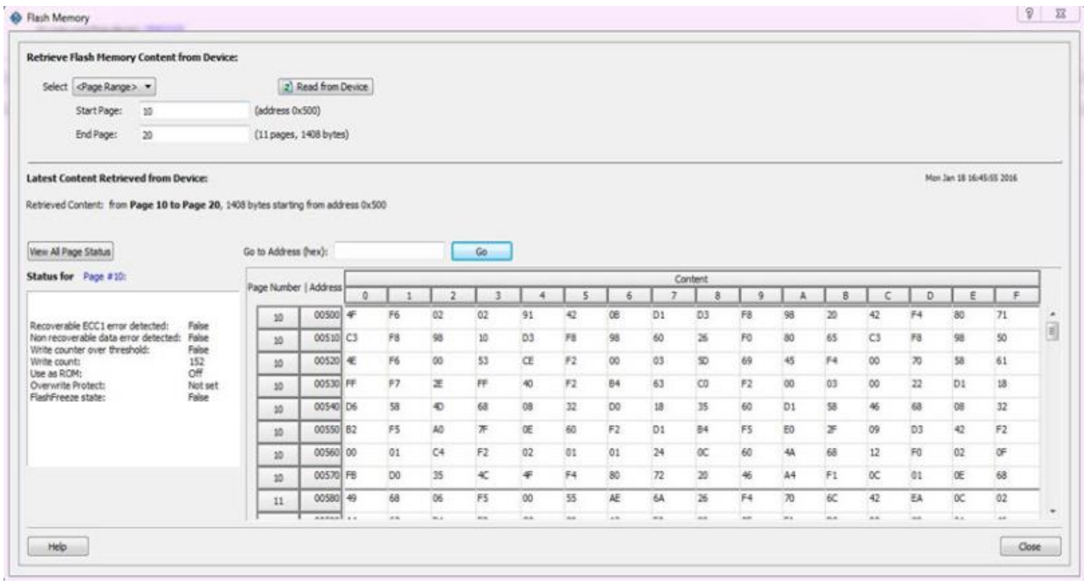


Figure 7 · Flash Memory Dialog Box for a SmartFusion2 Device (SmartDebug)

The page status gets updated when you:

- Click Page Range
- Click a particular cell in the retrieved eNVM content table
- Scroll pages from the keyboard using the Up and Down arrow keys
- Click Go to Address (hex)

The retrieved data table displays the content of the page range selection. If content cannot be read (for example, pages are read-protected, but security has been erased or access to eNVM private sectors), Read from Device reports an error.

Click **View Detailed Status** for a detailed report on the page range you have selected.

For example, if you want to view a report on pages 1-3, set the Start Page to 1, set the End Page to 3, and click **Read from Device**. Then click **View Detailed Status**. The figure below is an example of the data for a specific page range.

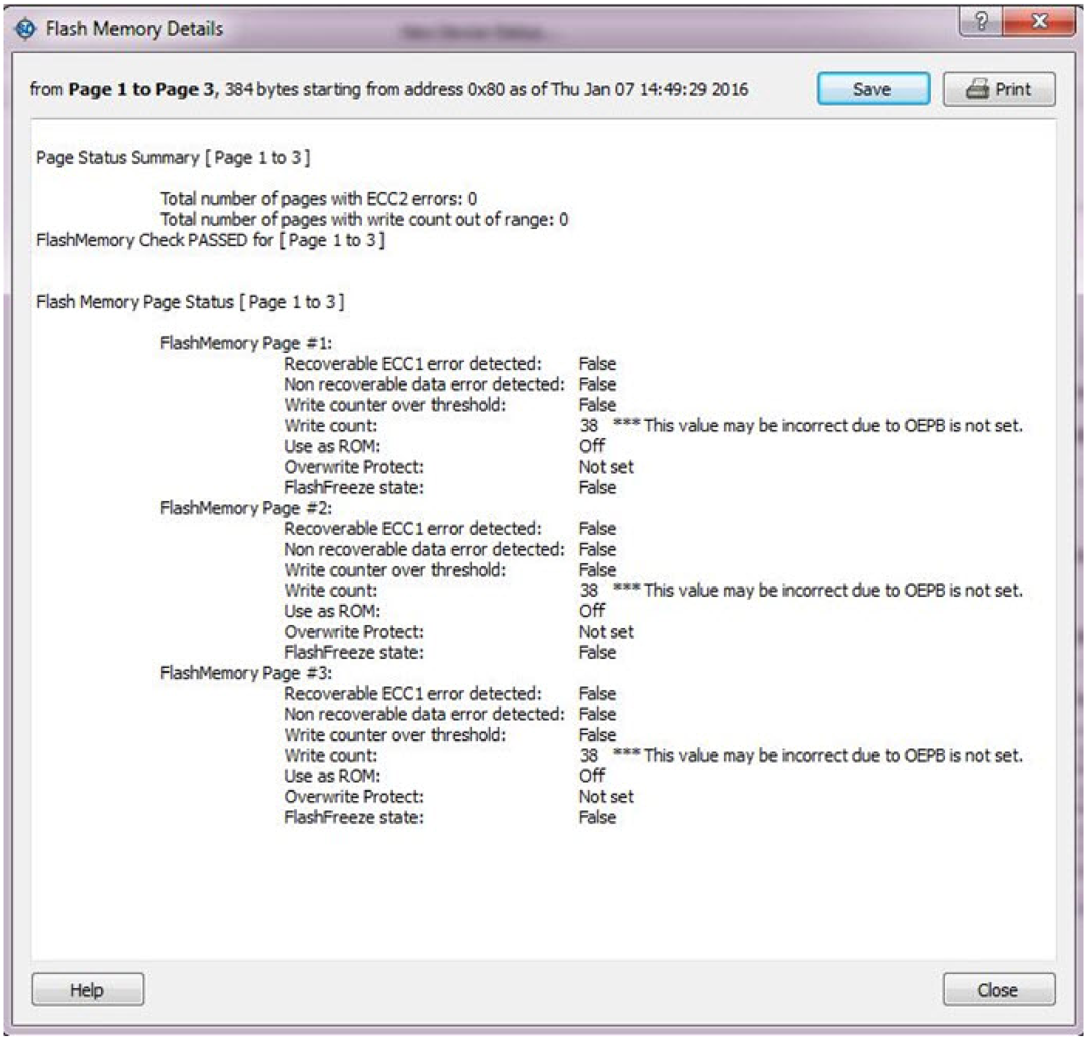


Figure 8 · Flash Memory Details Dialog Box (SmartDebug)

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 38
- "Frequency Monitor" on page 41
- "User Clock Frequencies" on page 52

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.

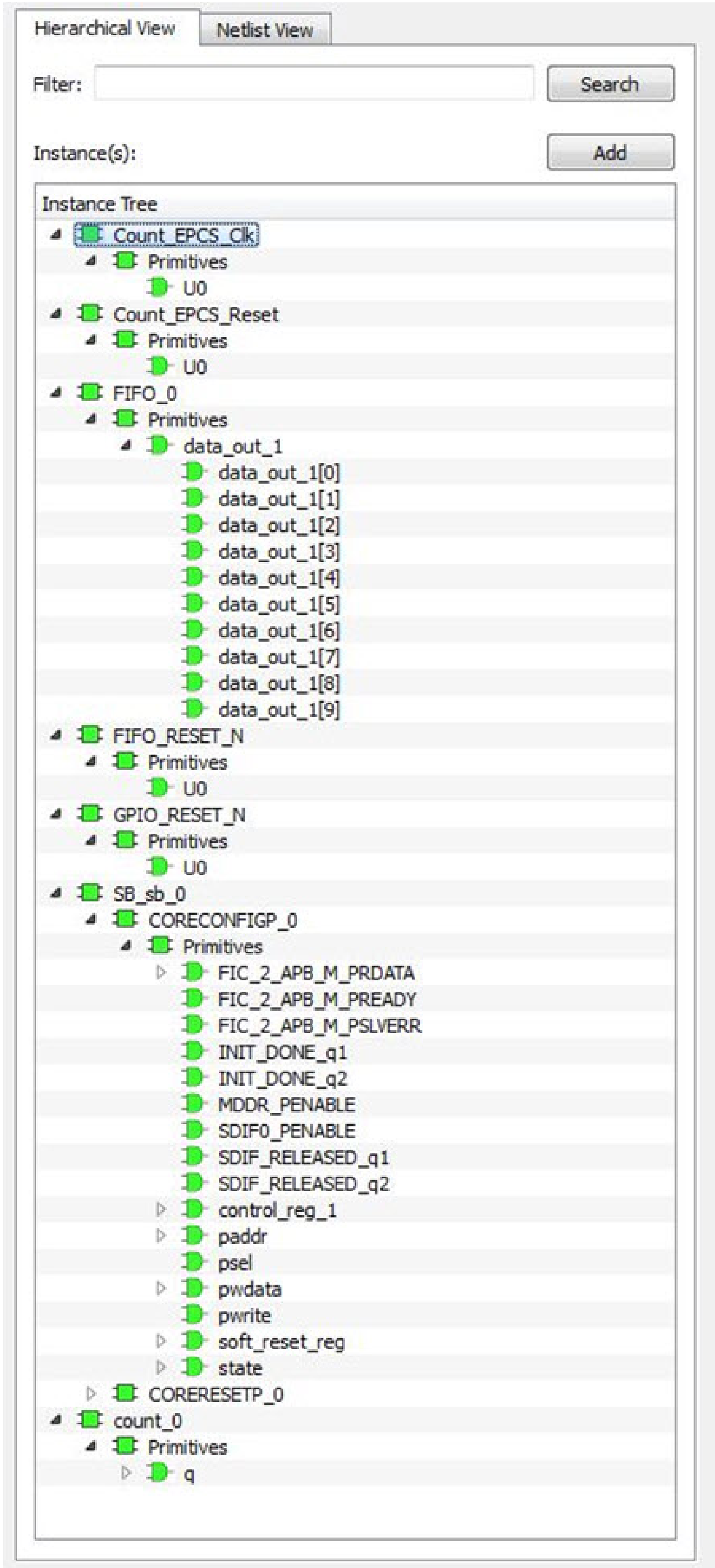


Figure 9 · Hierarchical View

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.

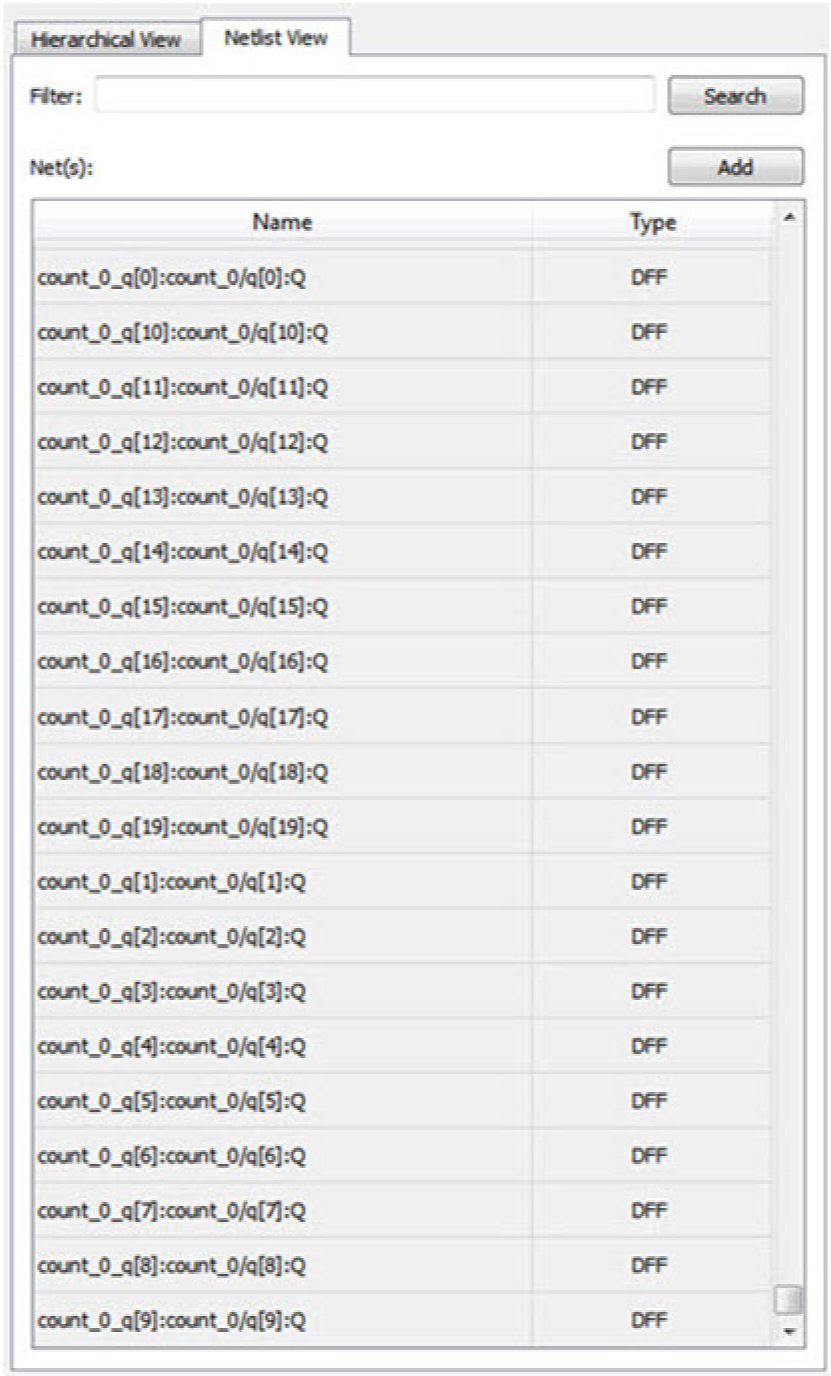


Figure 10 · Netlist View

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.

Note: SmartFusion2 and IGLOO2 support two probe channels, and RTG4 supports one probe channel.

SmartFusion2 and IGLOO2

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: At least one channel must be set; if you want to use both probes, they must be set at the same time.

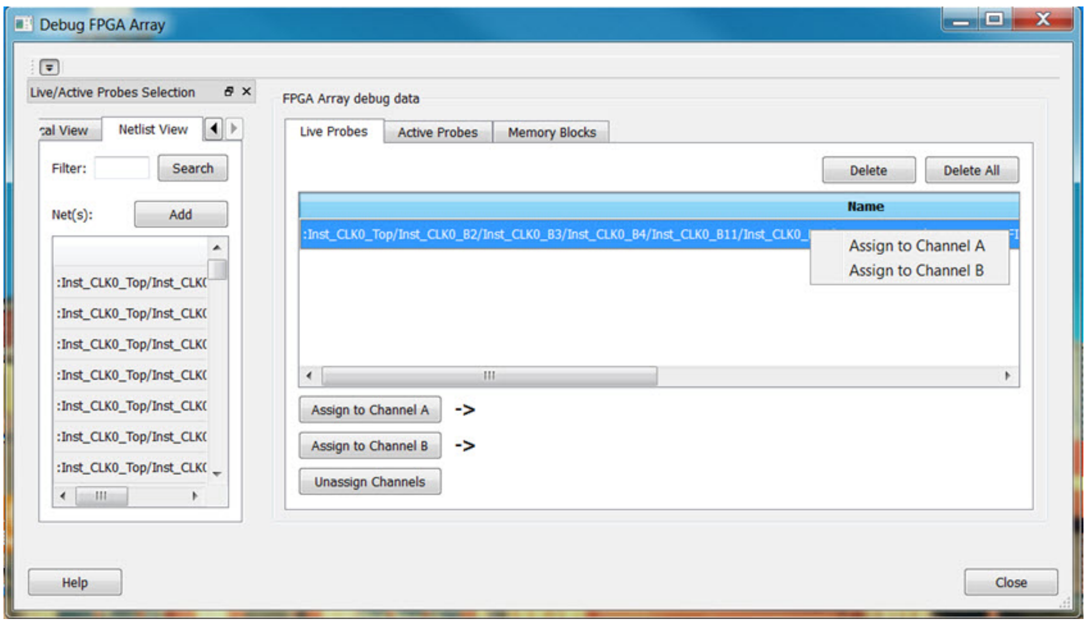


Figure 11 · Live Probes Tab (SmartFusion2 and IGLOO2) in SmartDebug FPGA Array Dialog Box

RTG4

One probe channel (Probe Read Data Pin) is available for RTG4 for debug. When a probe name is selected, it can be assigned to the Probe Channel (Probe Read Data Pin).

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 Probe Read Data Pin**.
- Click the **Assign to Probe Read Data Pin** button to assign the probe selected in the table to the channel. The button is located below the table.

Click the **Unassign probe read data pin** button to clear the live probe name to the right of the channel button and discontinue the live probe function during debug.

The Active Probes READ/WRITE overwrites the settings of Live Probe channels (if any).

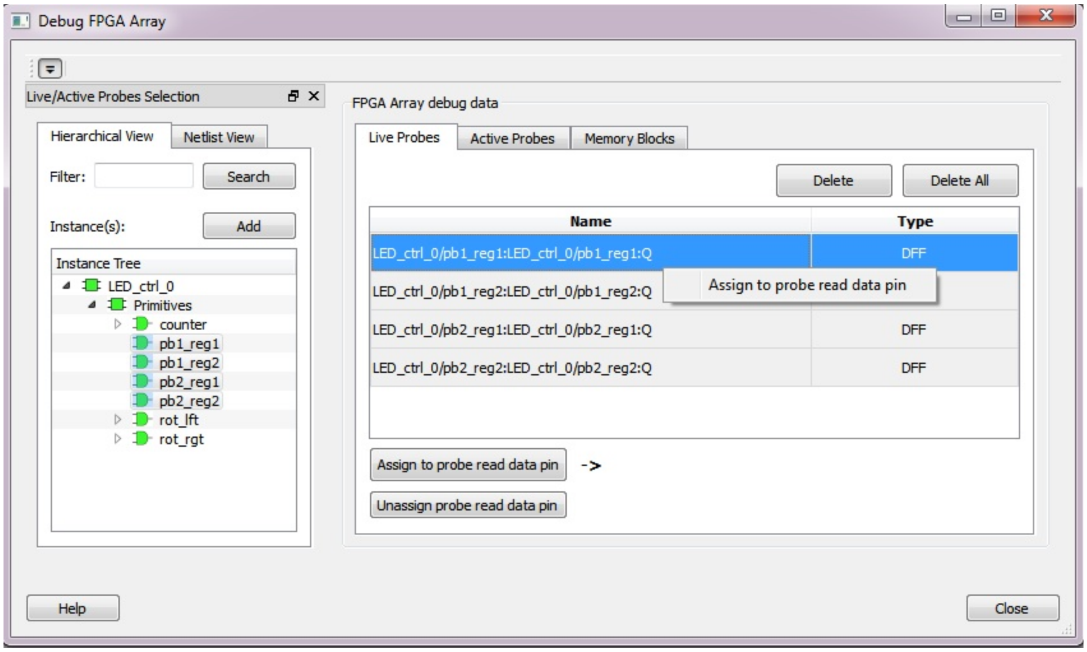


Figure 12 · Live Probes Tab (RTG4) in SmartDebug FPGA Array Dialog Box

Supported Families

SmartFusion2

IGLOO2

RTG4

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.

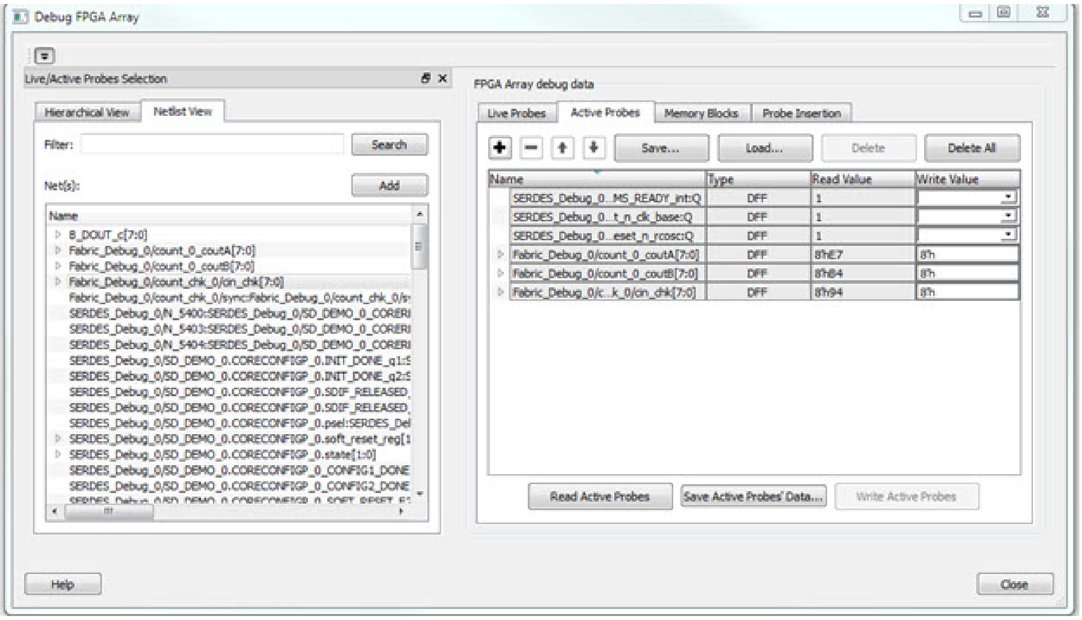


Figure 13 · Active Probes Tab in SmartDebug FPGA Array Dialog Box

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

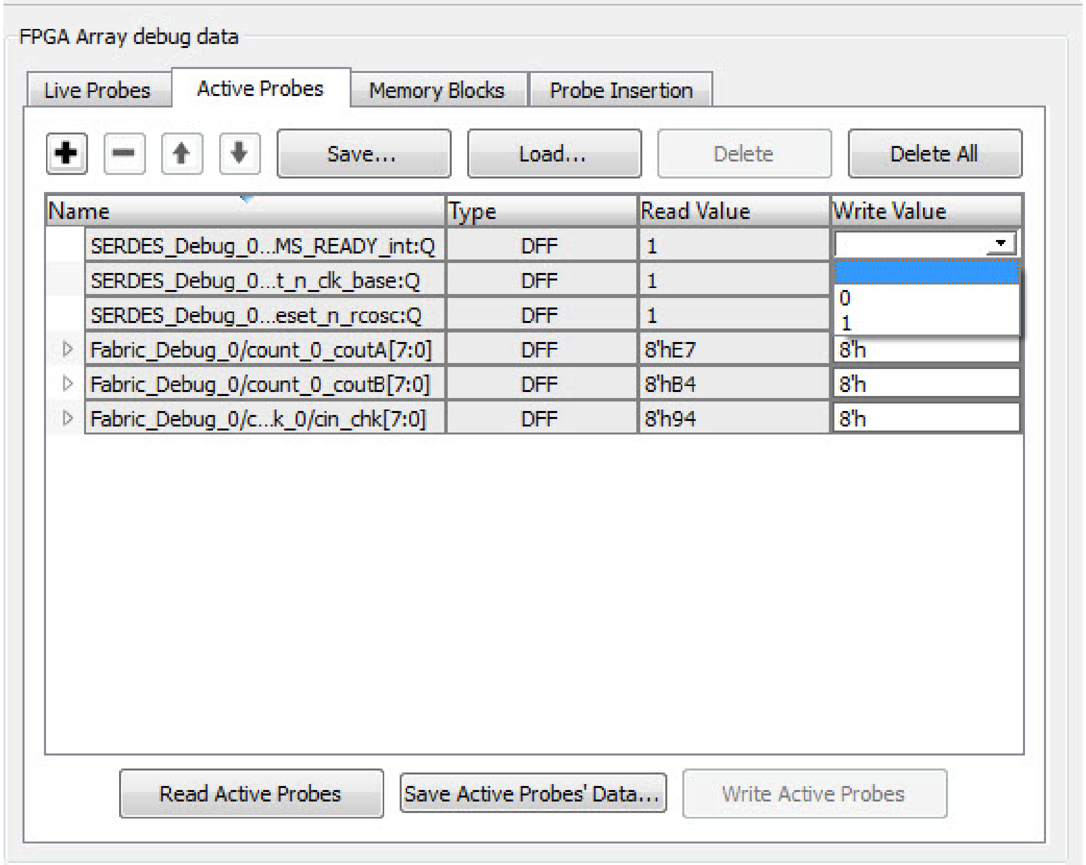


Figure 14 · Active Probes Tab - Write Value Column Options

Supported Families

SmartFusion2
IGLOO2
RTG4

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

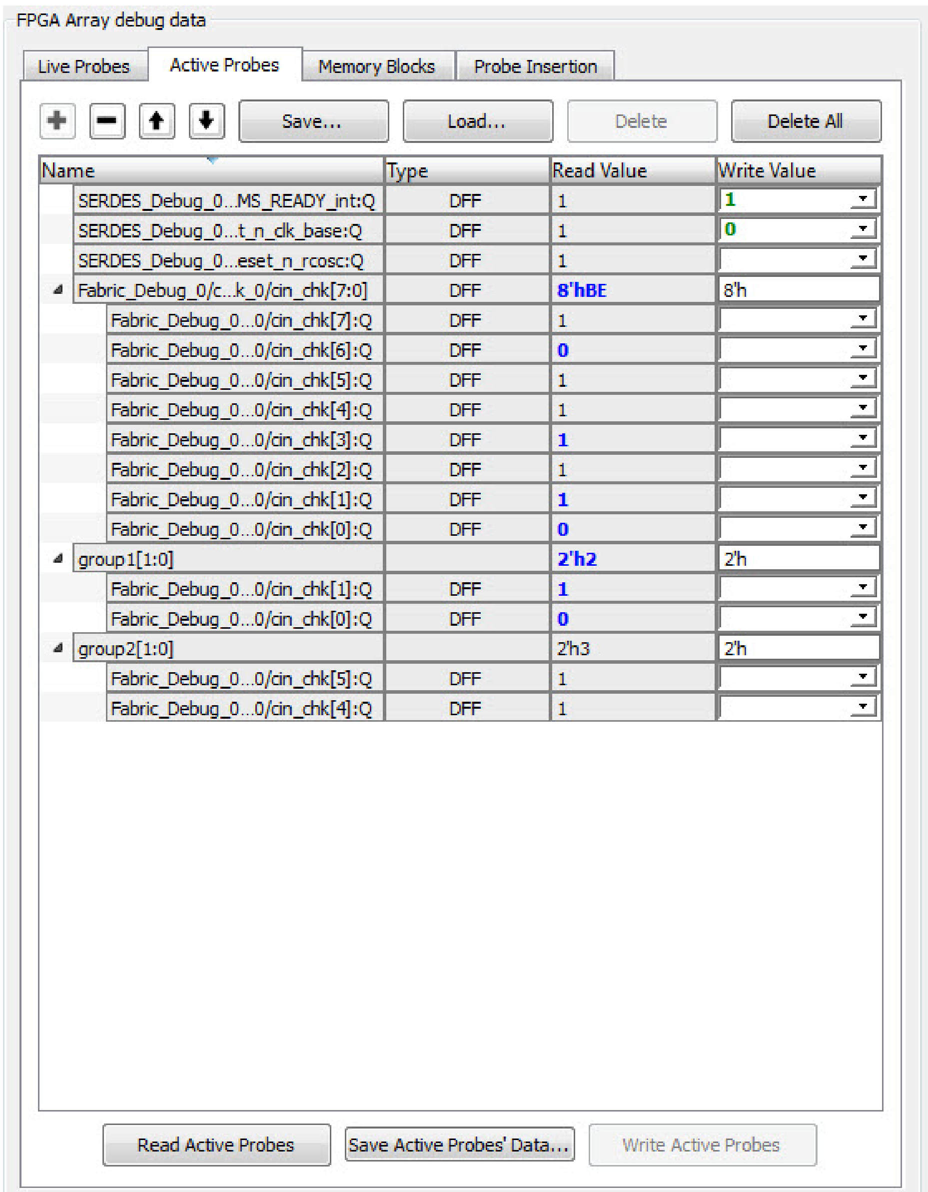


Figure 15 · Active Probes Tab

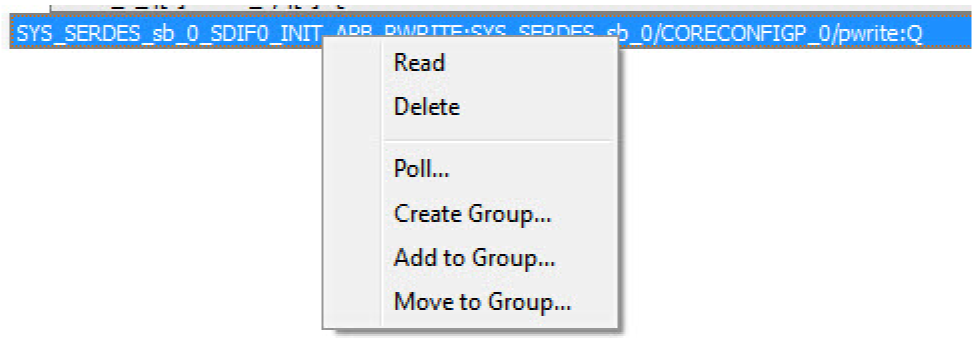
- Green entries in the “Write Value” column indicate that the operation was successful.
- Blue entries in the “Read Value” column indicate values that have changed since the last read.

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:

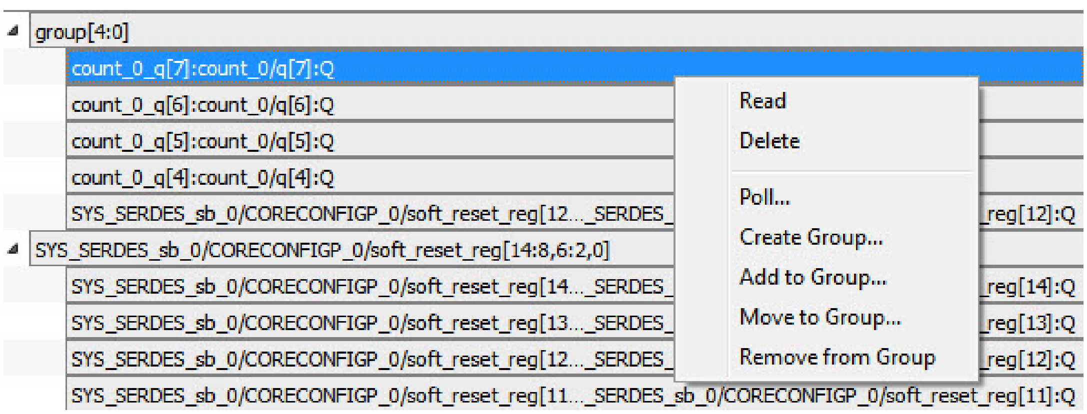
For individual signals that are not part of a probe group or bus:

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



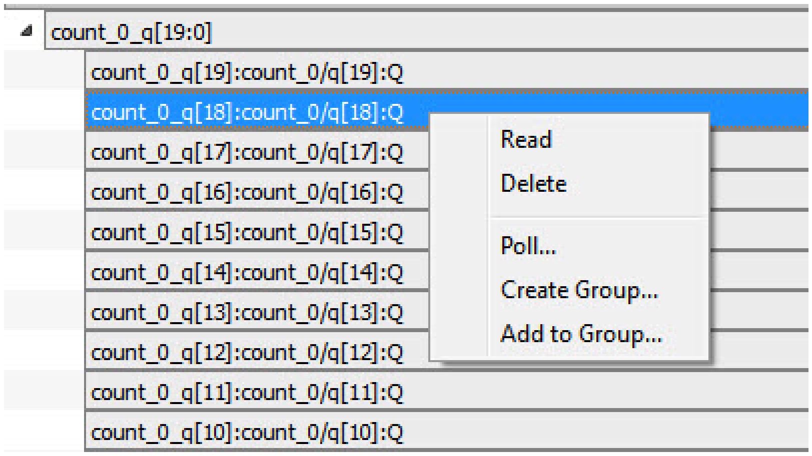
For individual signals in a probe group:

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



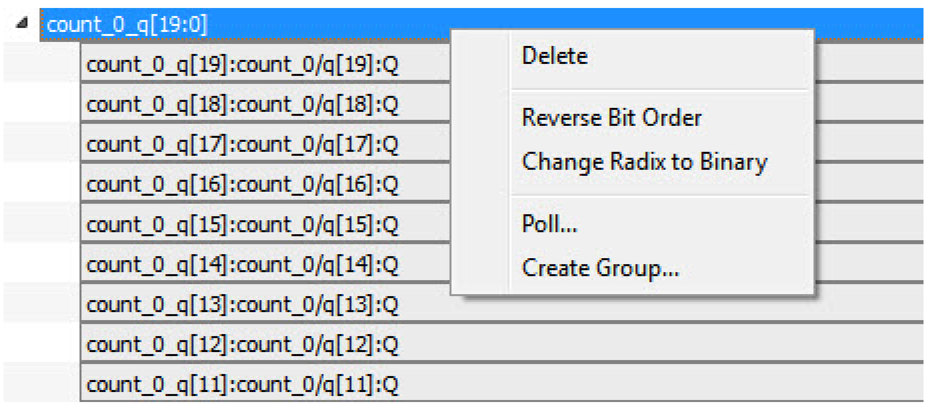
For individual signals in a bus:

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



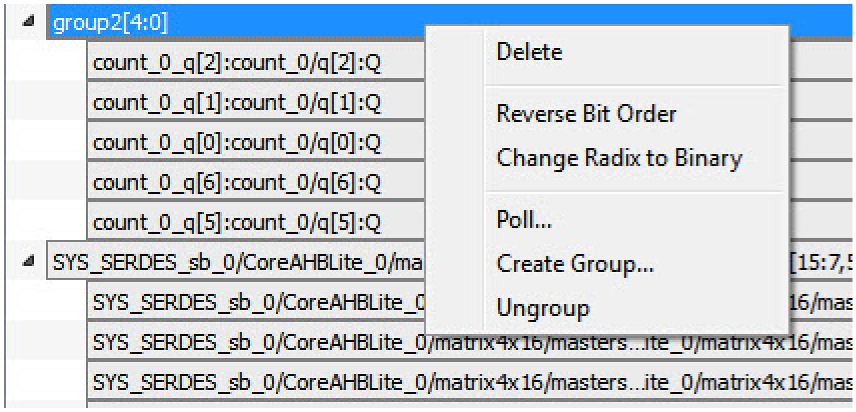
For a bus:

- 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** ().

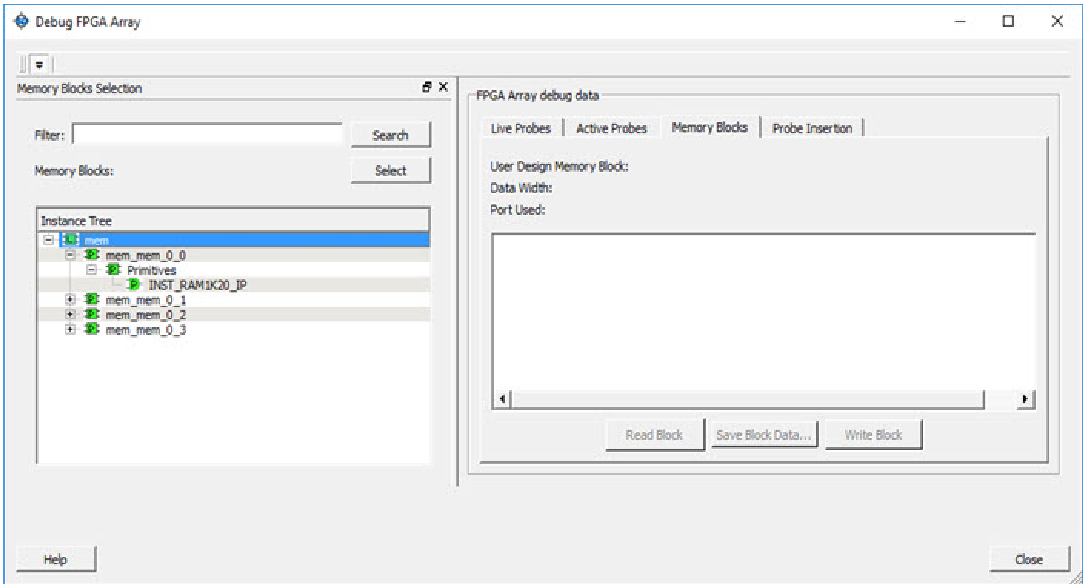


Figure 16 · Memory Blocks Tab - Hierarchical View

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.

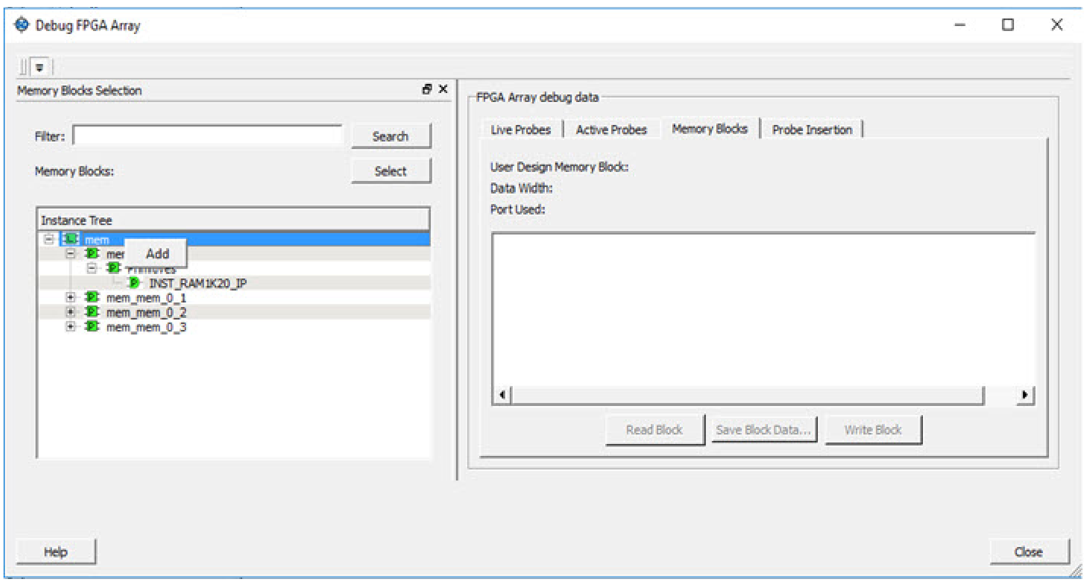


Figure 17 · Adding a Memory Block

- 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 width is 9-bits, and the depth is 128 for uSRAM blocks and 2048 for LSRAM blocks.

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.

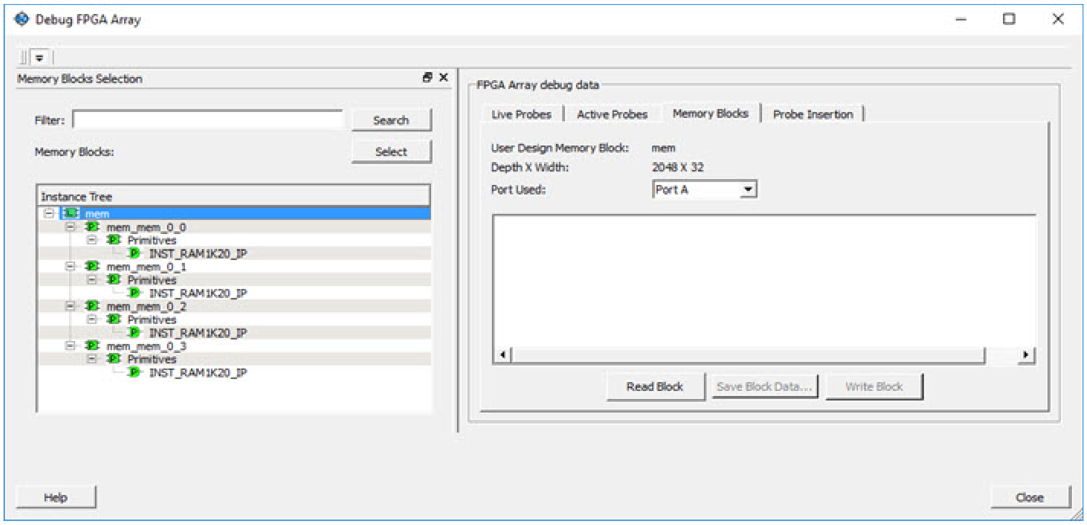


Figure 18 · Memory Blocks Tab Fields for Logical Block View

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

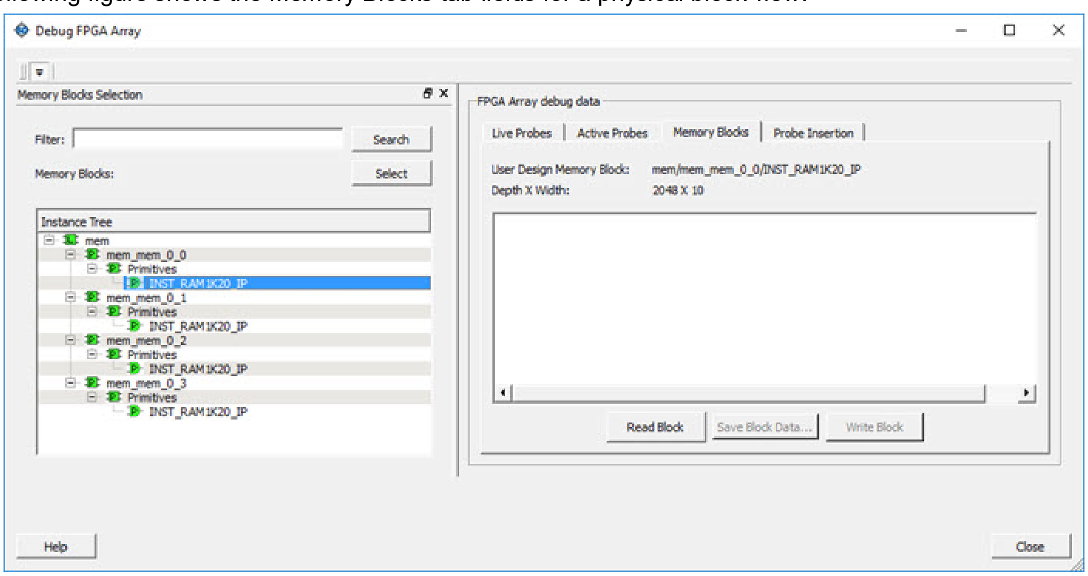


Figure 19 · Memory Blocks Tab Fields for Physical Block View

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.

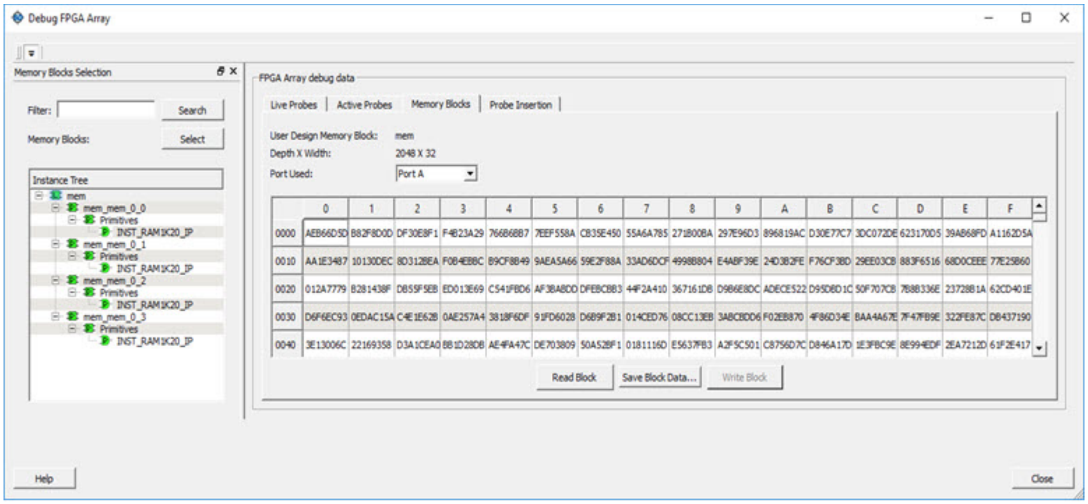


Figure 20 · Logical Block Read

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.

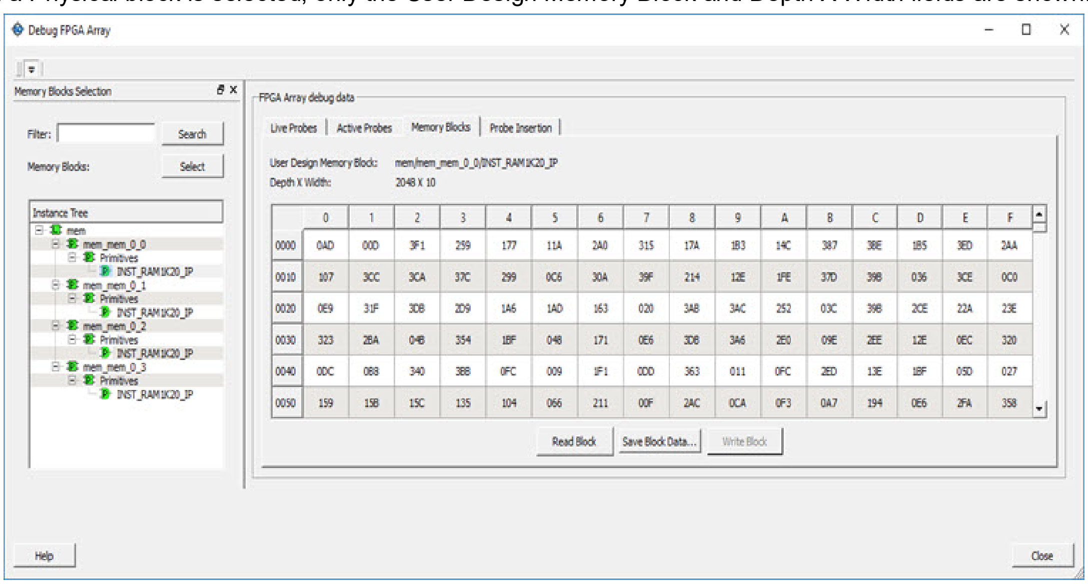


Figure 21 · Physical Block Read

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.

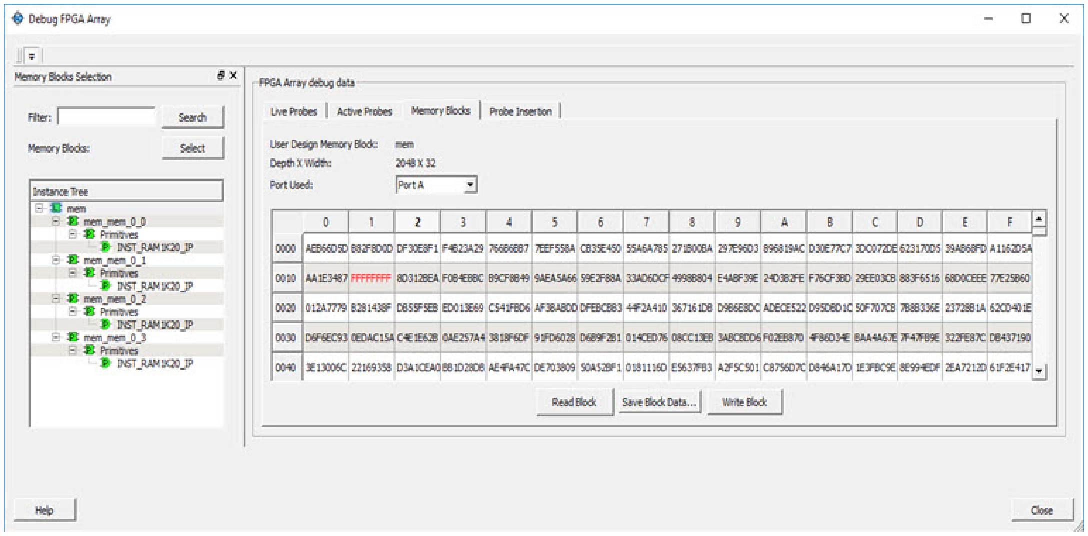


Figure 22 · Logical Block Write

Physical Block Write

Physical blocks have a fixed width of 9 bits. The maximum value that can be written in hexadecimal format is 1FF. If an entered value exceeds the limit, SmartDebug displays a popup message showing the range of values that can be entered.

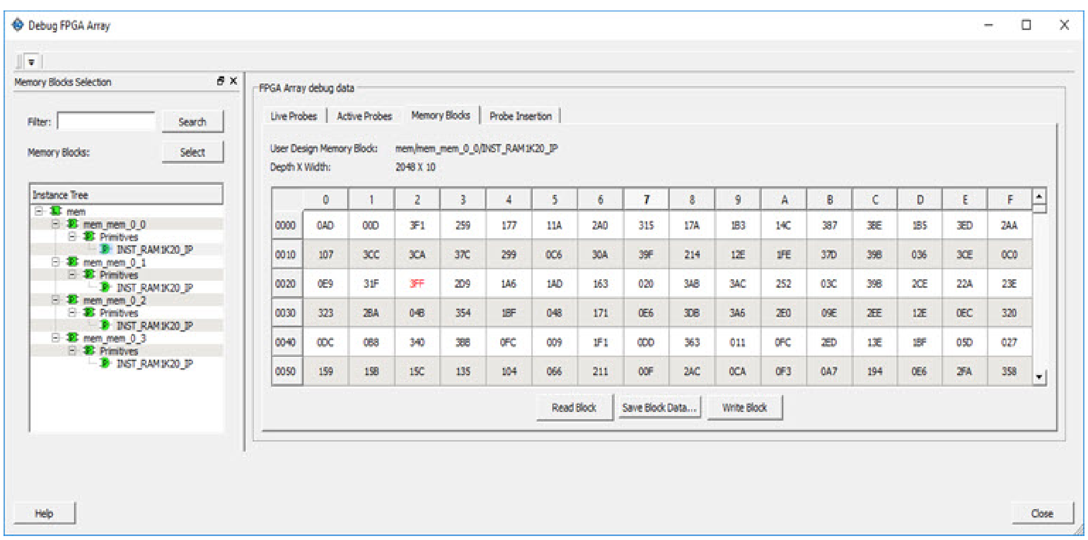


Figure 23 · Physical Block Write

Unsupported Memory Blocks

If RTL is used to configure memory blocks, it is recommended that you follow RAM block inference guidelines provided by Microsemi. See Inferring Microsemi SmartFusion2 RAM Blocks for more information.

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.

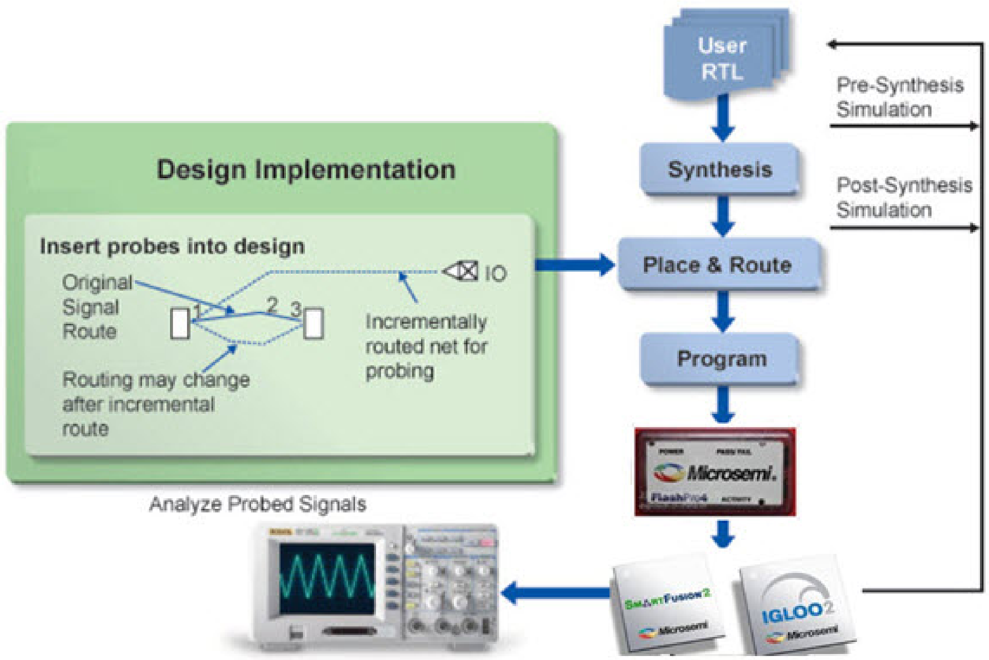


Figure 24 · Probe Insertion in the Design Process

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.

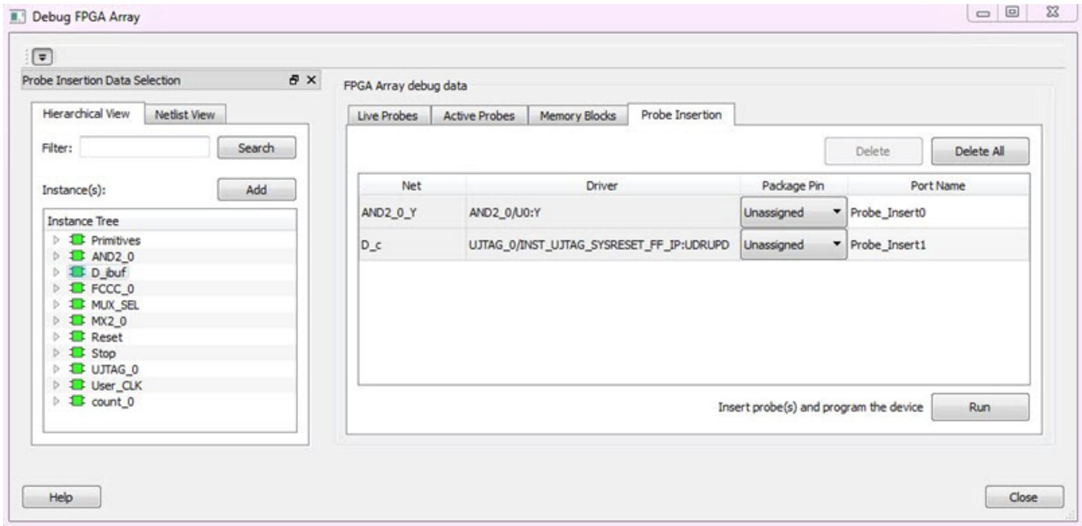


Figure 25 · Probe Insertion Tab

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).

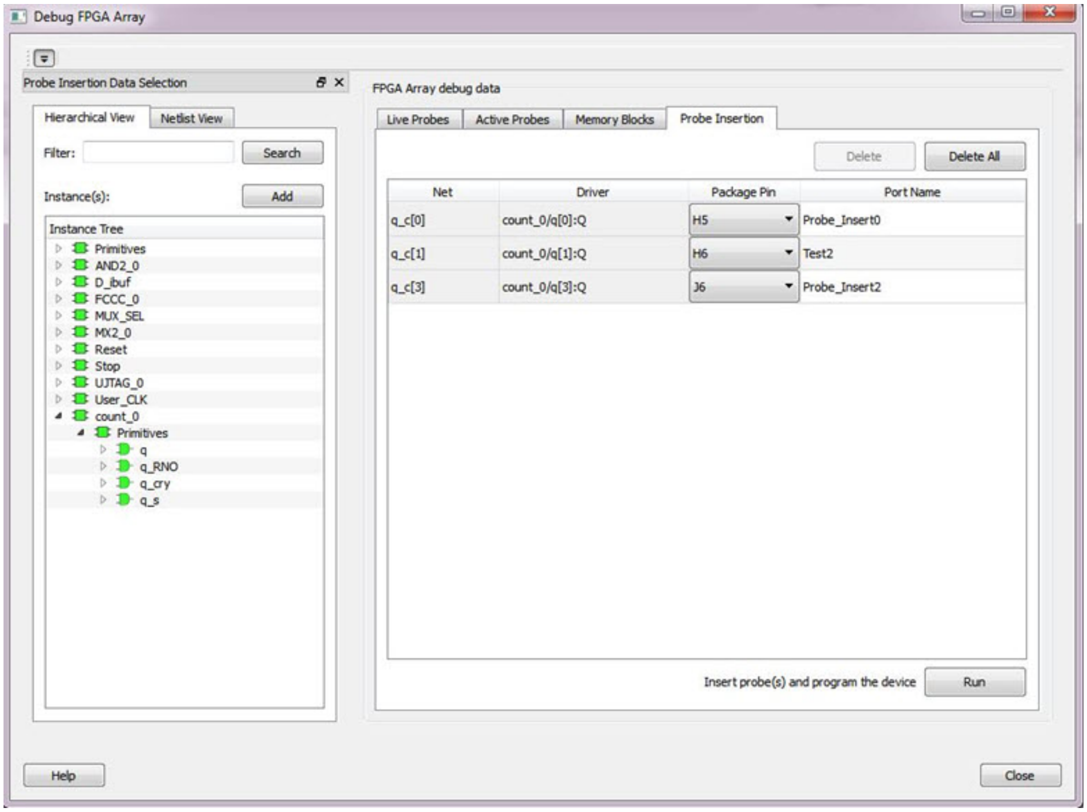


Figure 26 · Debug FPGA Array > Probe Insertion > Add Probe

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 MSS or 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.

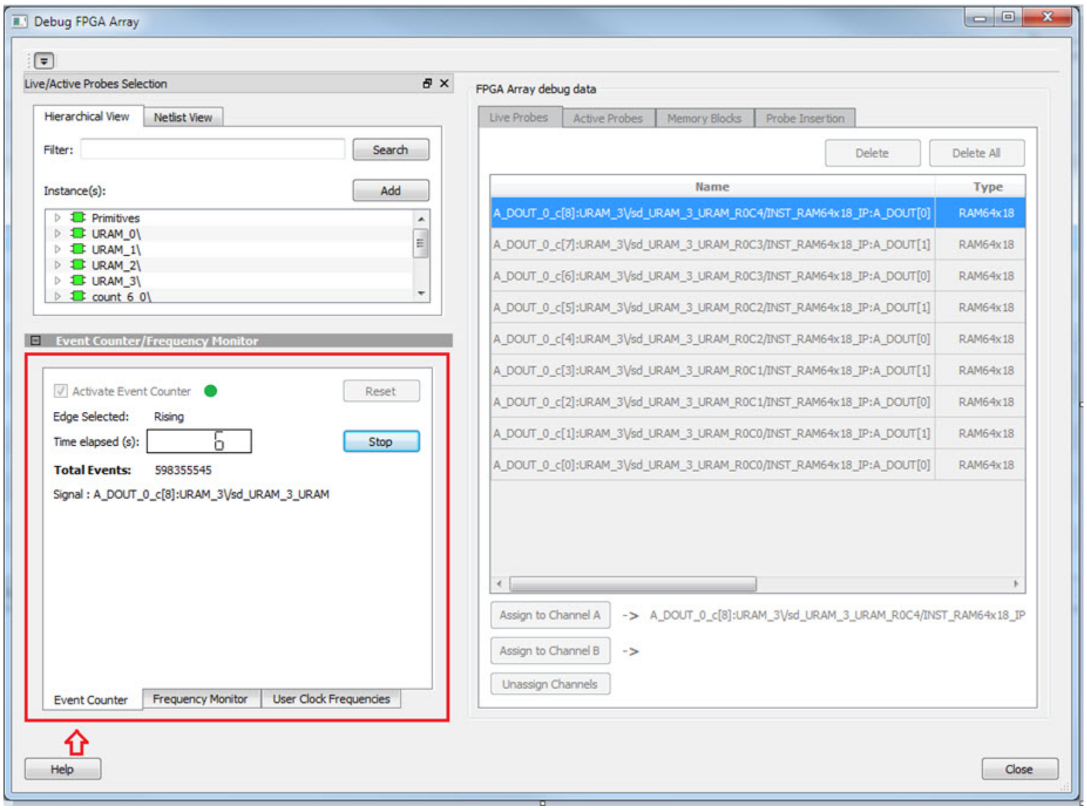


Figure 27 · Event Counter Tab/UI

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.

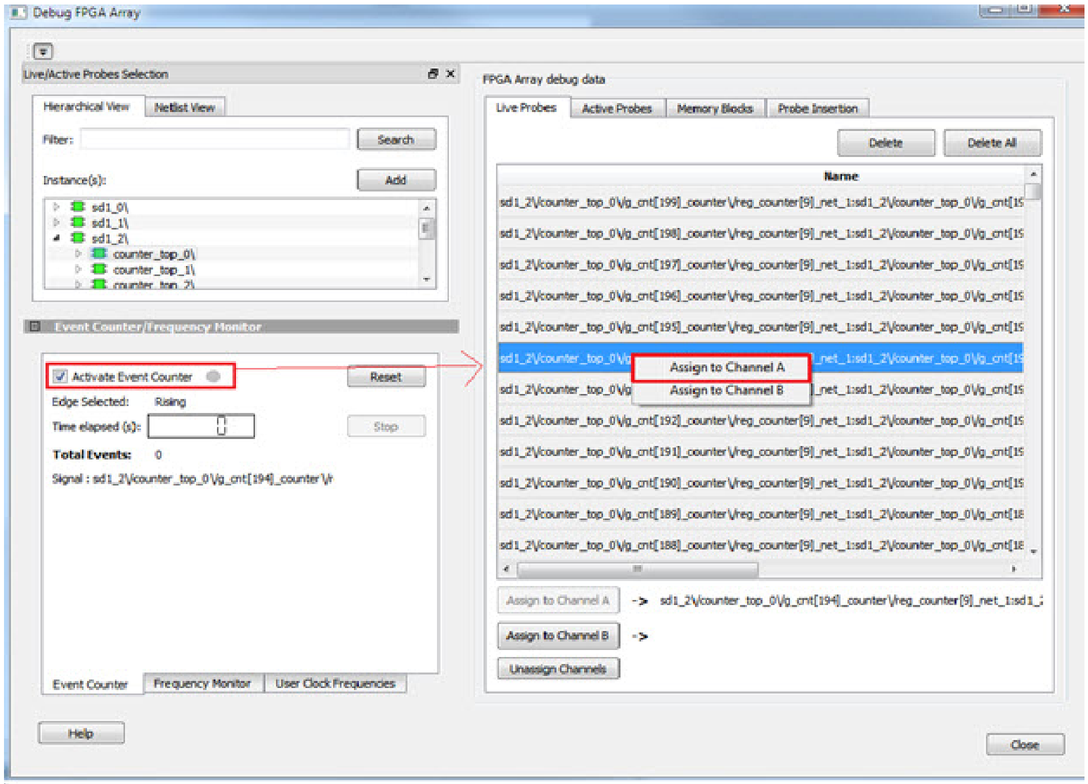


Figure 28 · Activating the Event Counter

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

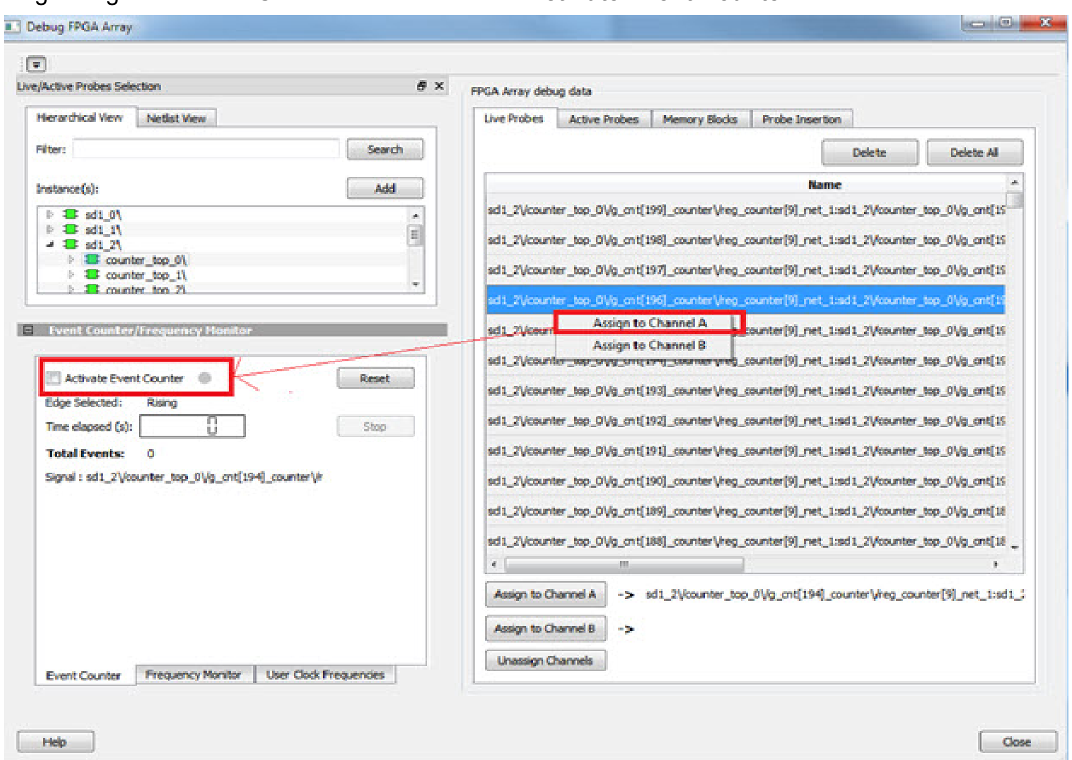


Figure 29 · Activating the Event Counter - Assign Probe Channel

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.

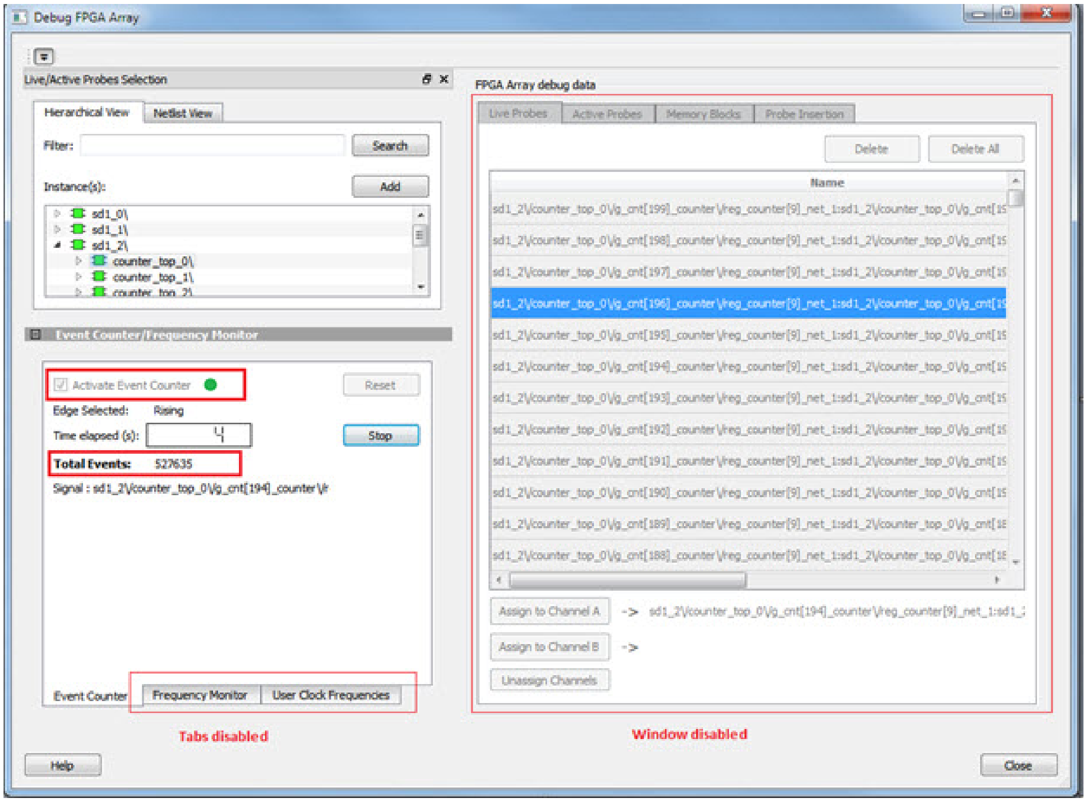


Figure 30 · Running the Event Counter

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.

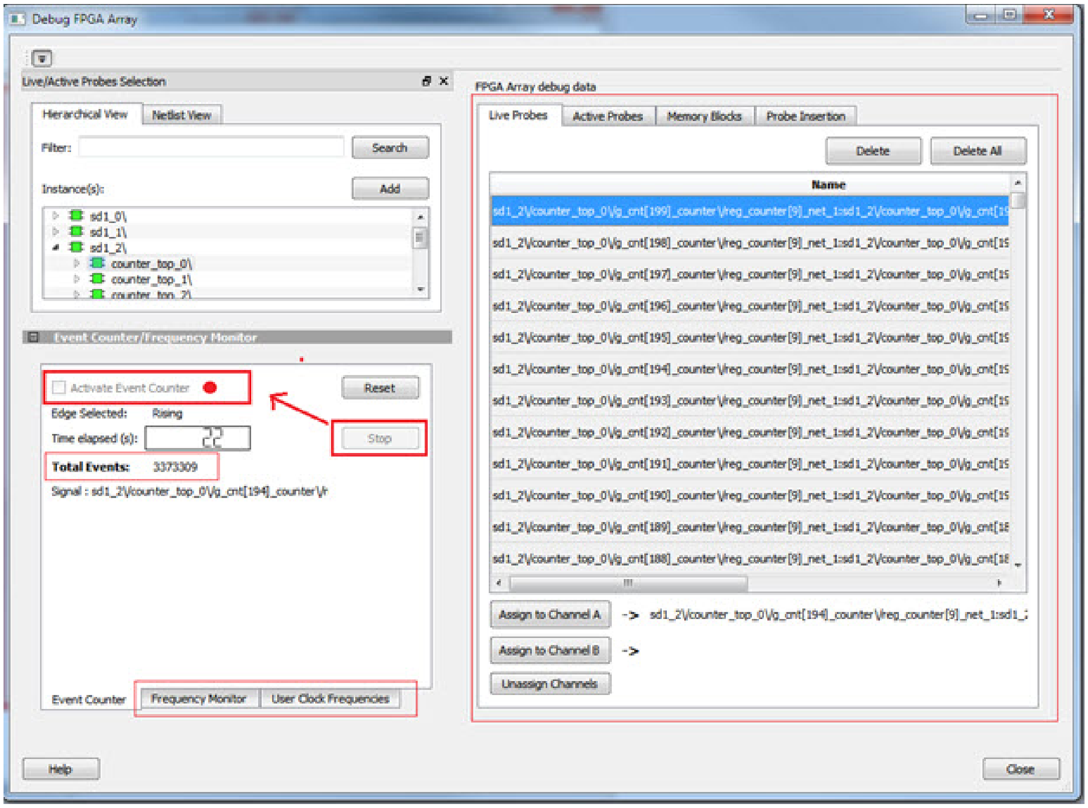


Figure 31 · Stopping the Event Counter

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 41

"User Clock Frequencies" on page 52

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.

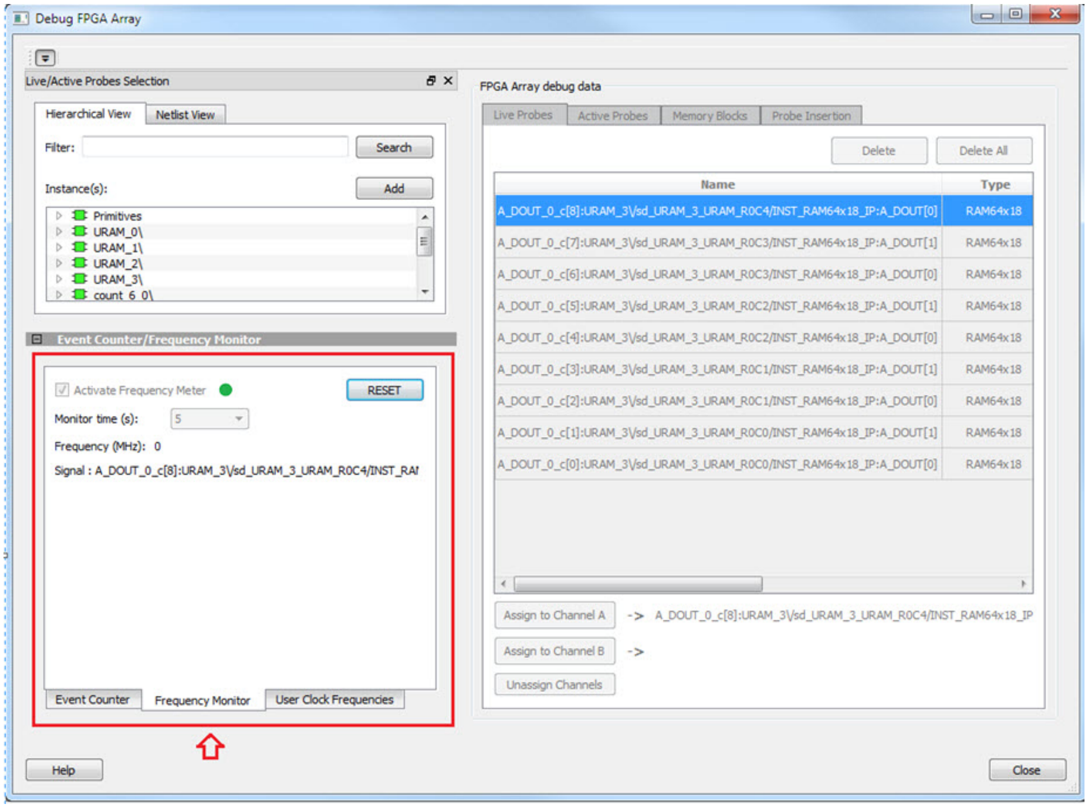


Figure 32 · Frequency Monitor Tab/UI

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).

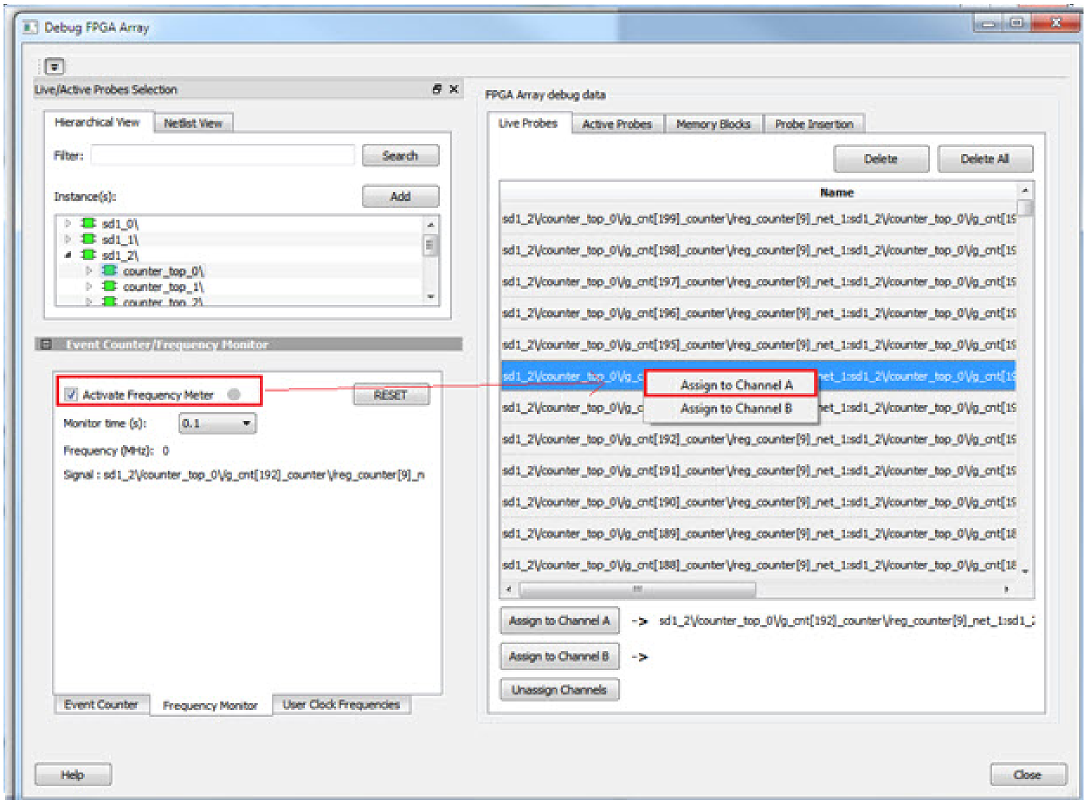


Figure 33 · Activating the Frequency Monitor - Assign a Signal

- 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.

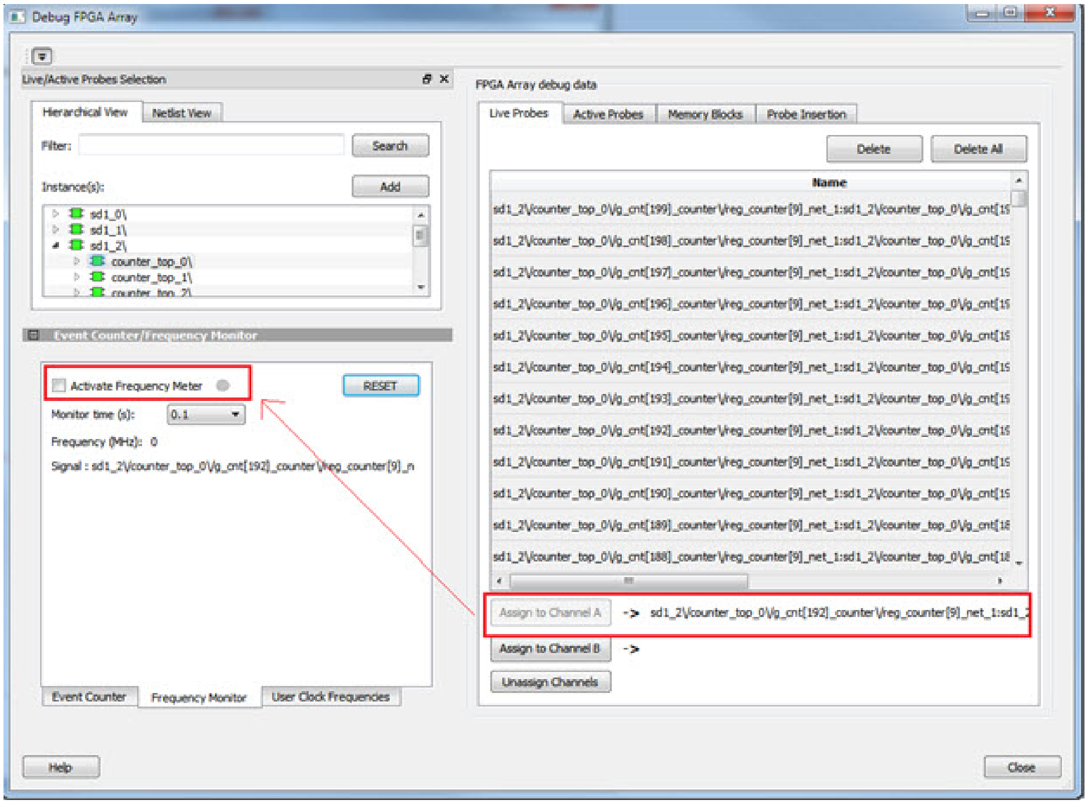


Figure 34 · Activating the Frequency Monitor

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.

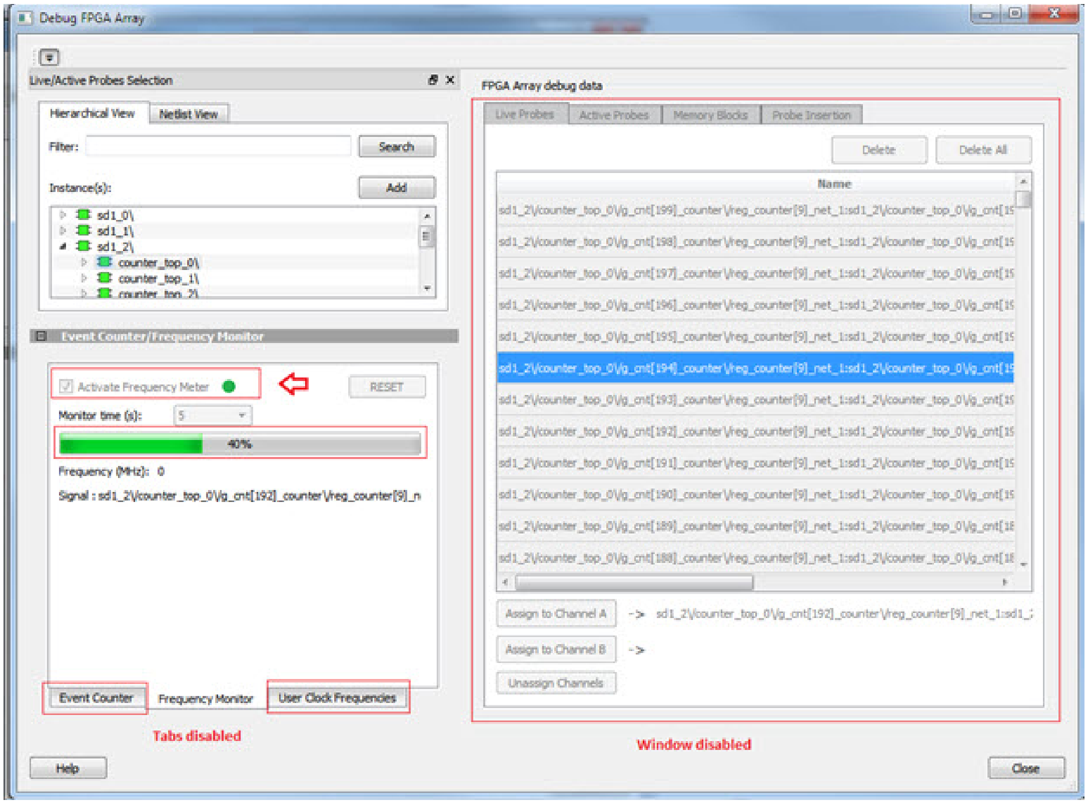


Figure 35 · Running the Frequency Monitor

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.

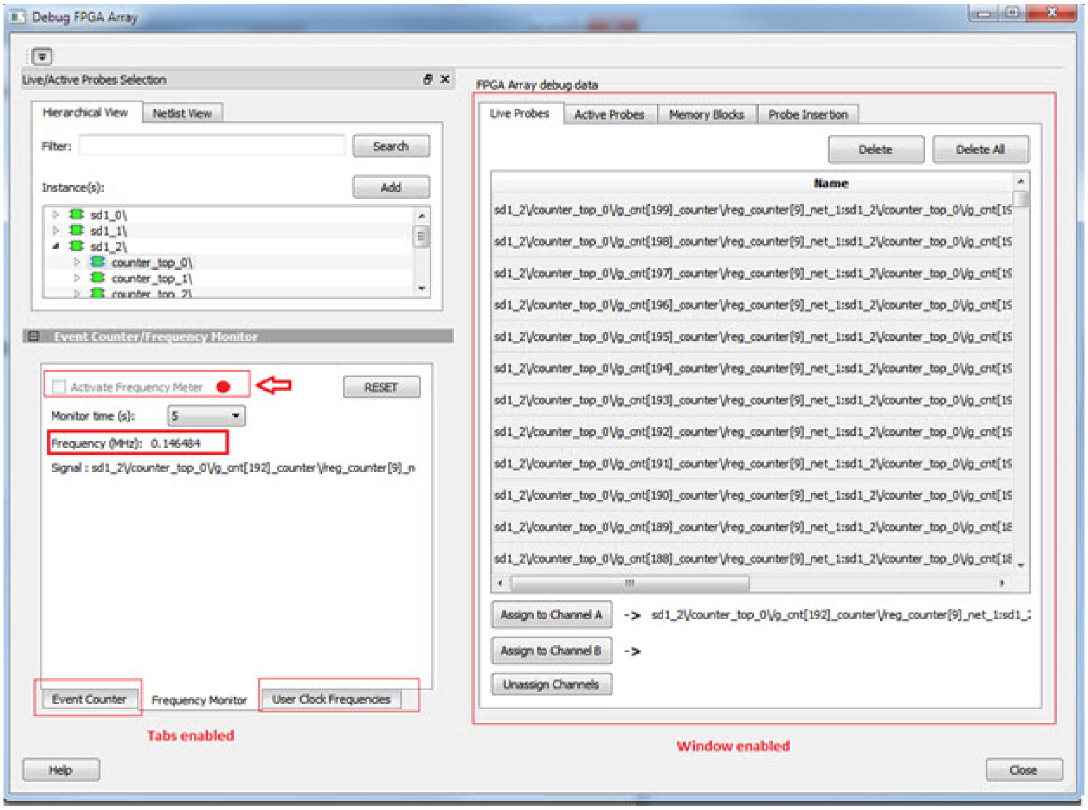


Figure 36 · Stopping the Frequency Monitor

See Also

"Event Counter" on page 38

"User Clock Frequencies" on page 52

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.

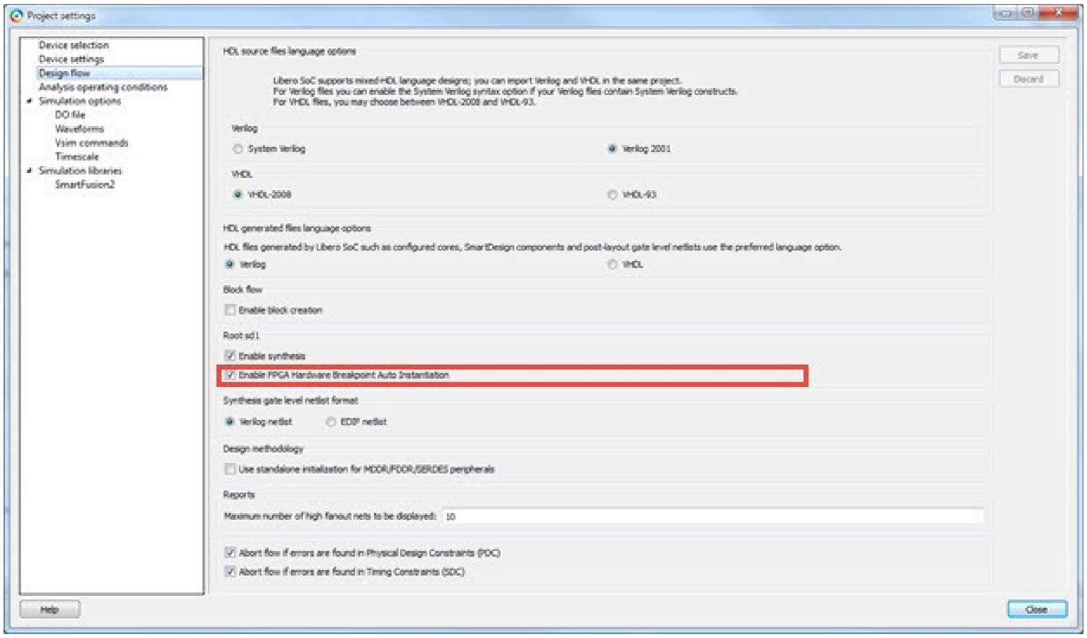


Figure 37 · Enable FHB Auto Instantiation in Project Settings Dialog Box: Design flow Tab

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.

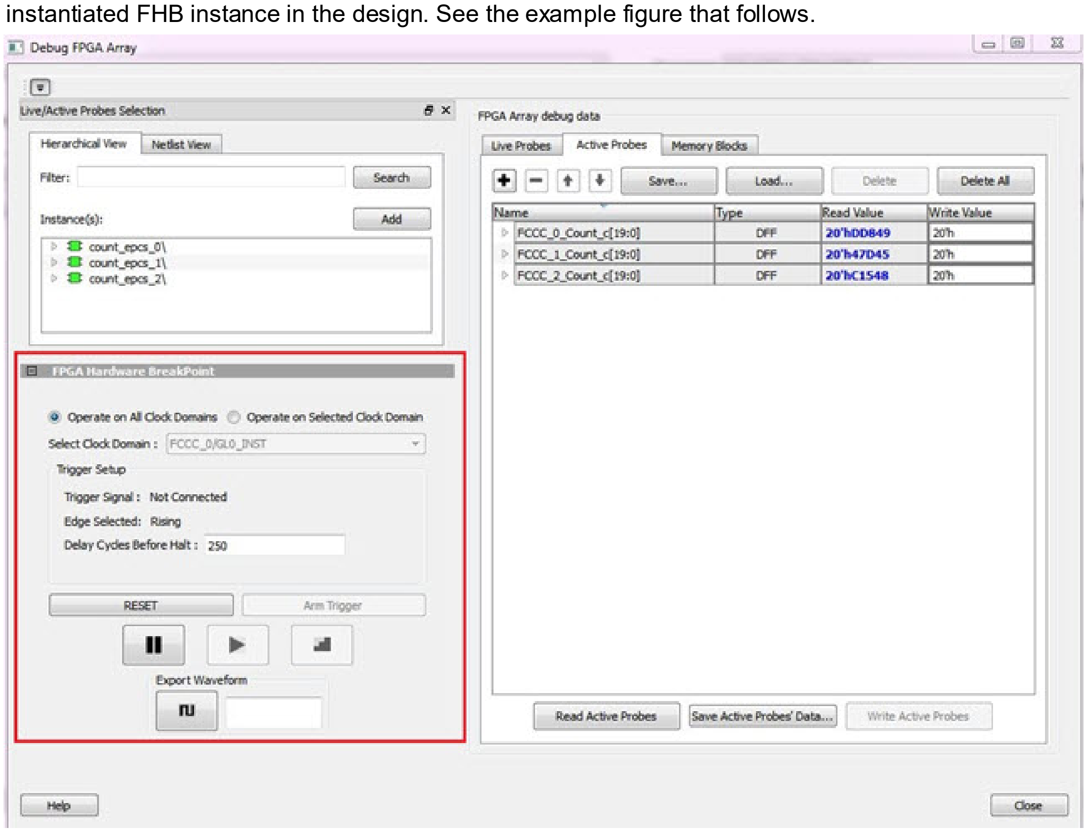
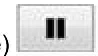
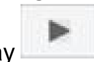


Figure 38 · FPGA Hardware Breakpoint (FHB) Controls

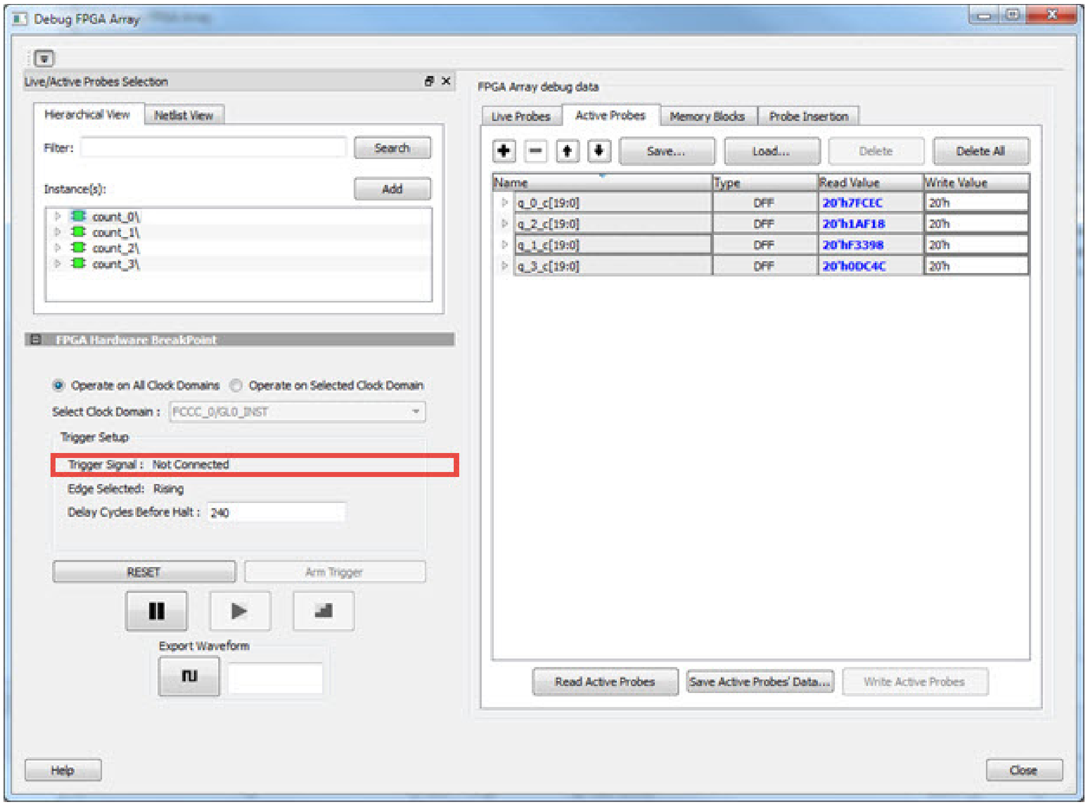
You can choose **Operate on All Clock Domains** or **Operate on Selected Clock Domain** by selecting the appropriate radio button. Selecting either of these modes sets the FHB instances to the respective mode. Once you assign the Live Probe PROBE_A connection and click **Arm Trigger**, the DUT halts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

Note: The **Operate on Selected Clock Domain** mode is not supported for RTG4 devices.

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.

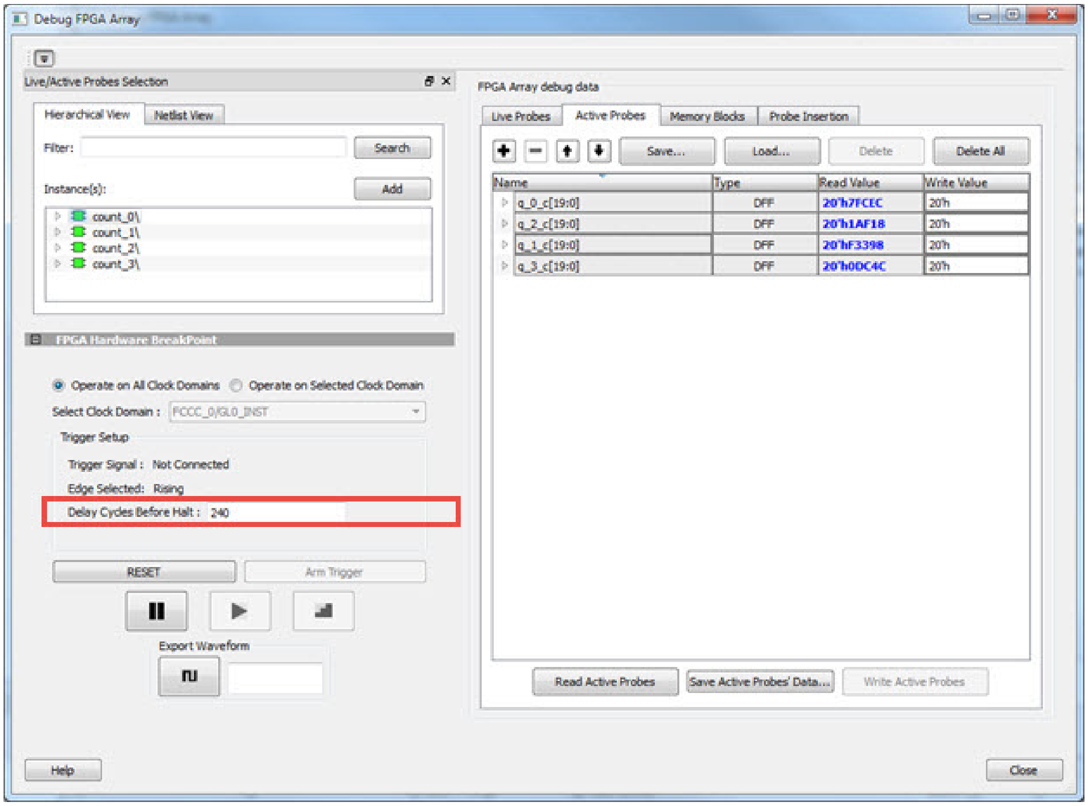
The Trigger Signal is shown as Not Connected until a live probe is assigned. See the example figure that follows.



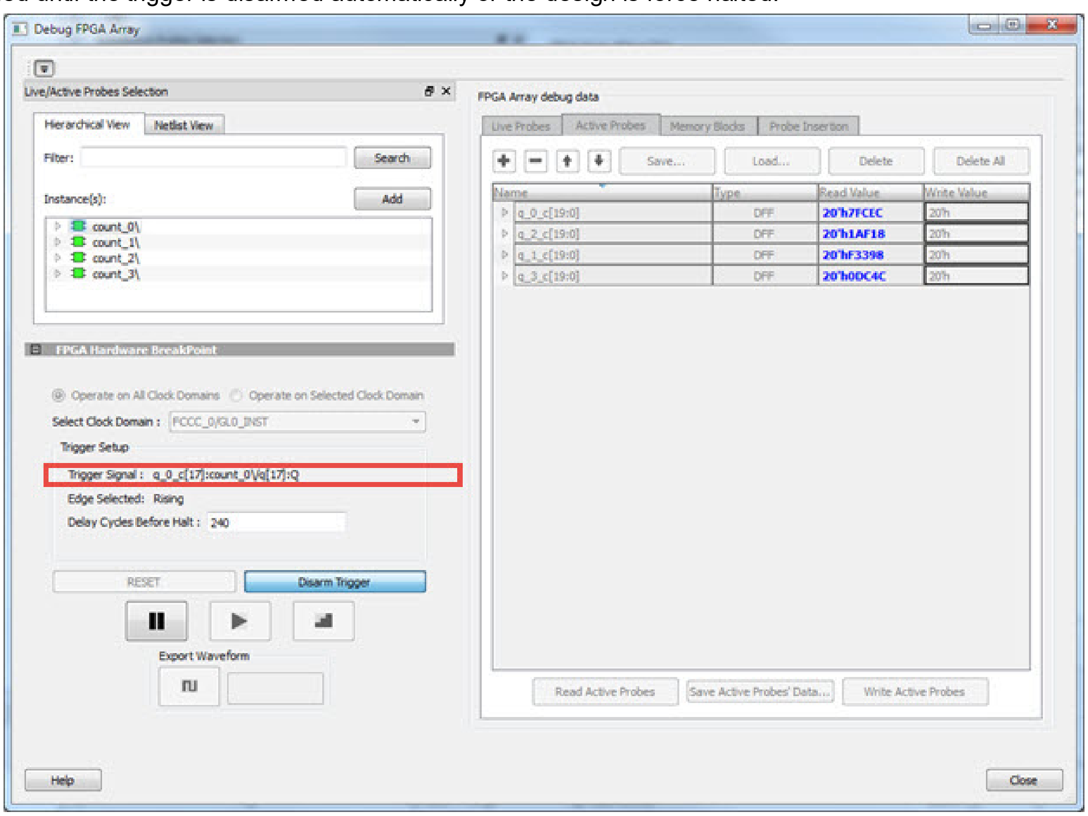
When a probe is assigned to Live Probe PROBE_A, the Trigger Signal updates.

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.

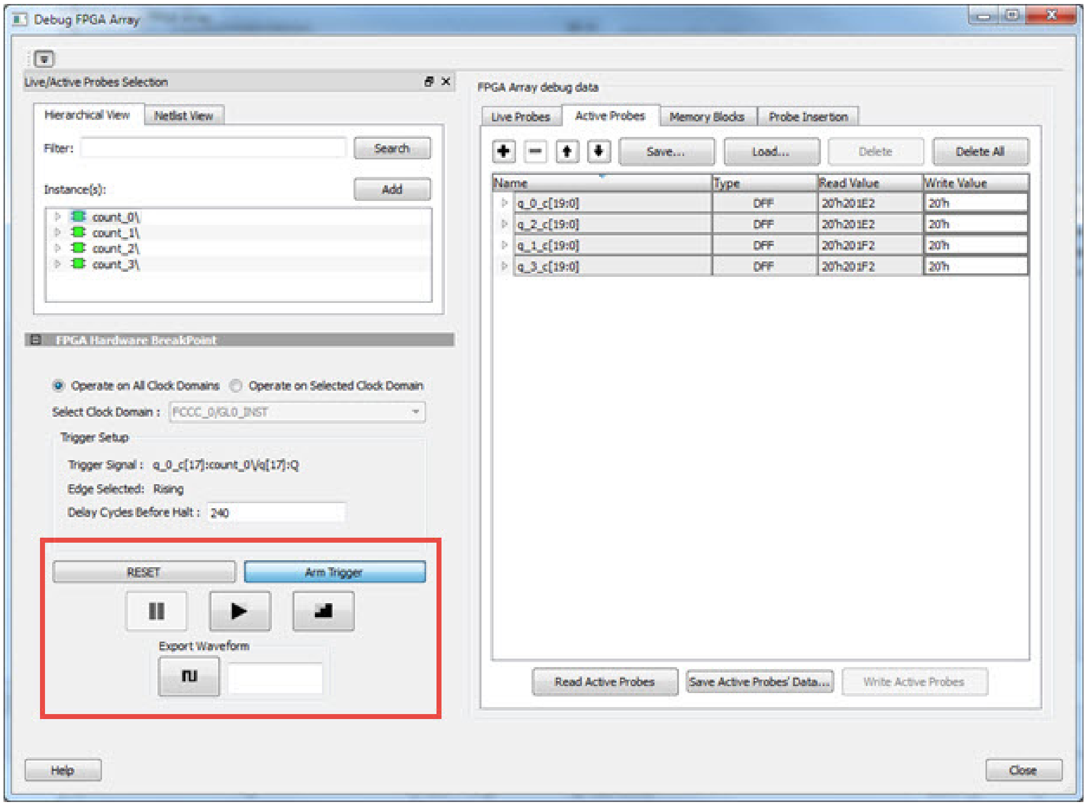


Trigger Input

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:

SmartFusion2: 50MHz

IGLOO2: 50MHz

RTG4: 100MHz

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.

Reset

You can reset a selected clock domain or all clock domains (based on the mode selection) by clicking **RESET** at any time. This resets the FHBs on clock domains and instructs FHB muxes not to look for a trigger.

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. 50MHz).
  - If the DUT clock is running at or less than 50MHz, the DUT clock will halt within one clock cycle (1 or less).
  - For frequencies higher than 50MHz, 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.

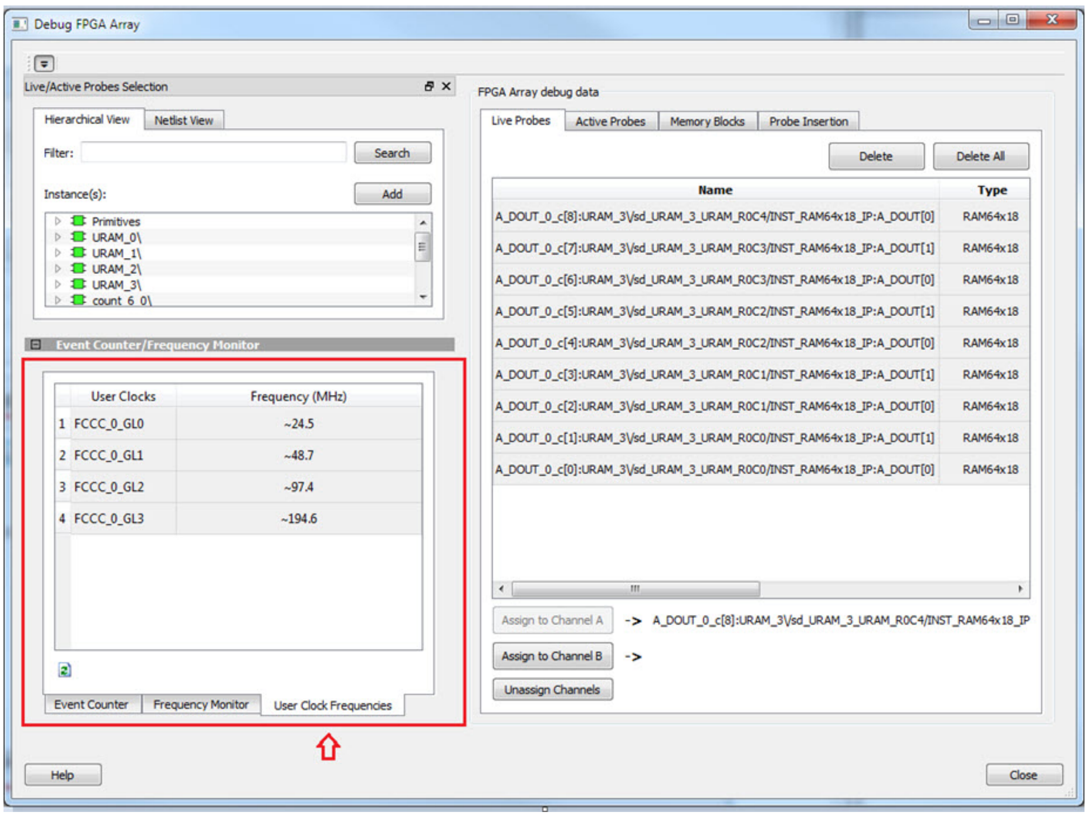


Figure 39 · User Clock Frequencies Tab/UI

See Also

"Event Counter" on page 38
"Frequency Monitor" on page 41
[UG0449- SmartFusion2 and IGLOO2 Clocking Resources User Guide](#)
[UG0586- RTG4 FPGA Clocking Resources User Guide](#)

Pseudo Static Signal Polling

With Active Probes you can check the current state of any probe in the design. However, in most cases, you will not able to time the active probe read to capture its intended value. For these cases, you can use Pseudo Static Signal Polling, in which the SmartDebug software polls the signal at intervals of one second to check if the probe has the intended value. This feature is useful in probing signals which reach the intended state and stay in that state.

From the Active Probes tab in the Debug FPGA Array dialog box, right-click a signal, bus, or group and choose **Poll...**. See the example figure that follows.

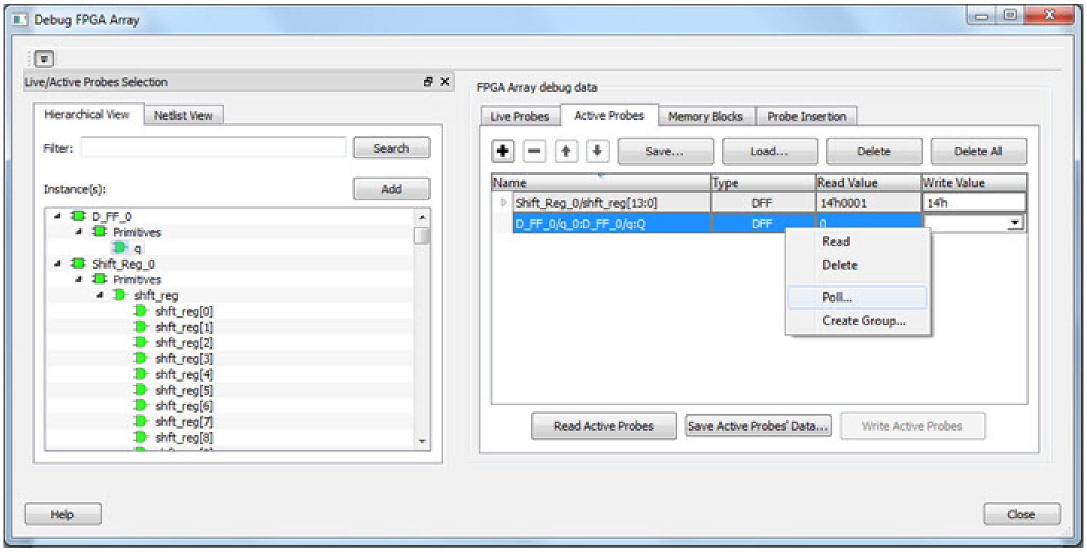


Figure 40 · Debug FPGA Array Dialog Box - Poll Option

The Pseudo-static signal polling dialog box opens.

Scalar Signal Polling

Polling Setup

To poll scalar signals, select **Poll for 0** or **Poll for 1**.

The selected signal is polled once per second. It should be used for pseudo-static signals that do not change frequently. The elapsed time is shown next to **Time Elapsed in seconds**.

To begin polling, click **Start Polling**. See the following example figure.

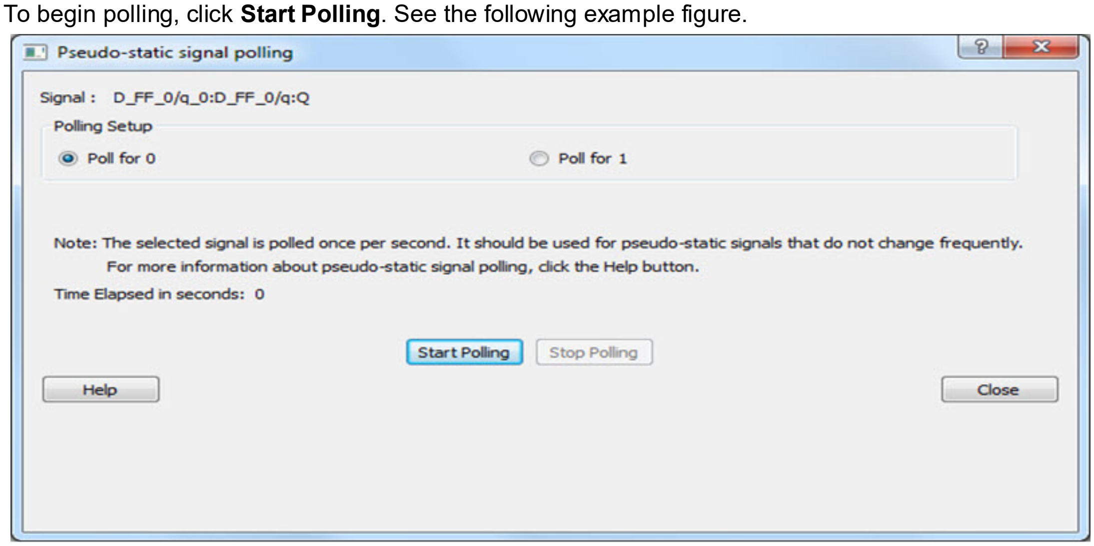


Figure 41 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Start Polling

To end polling, click **Stop Polling**. See the following example figure.

Figure 42 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Stop Polling

Note: You cannot change the poll value or close the polling dialog box while polling is in progress.

The elapsed time is updated in seconds until the polled value is found. When the polled value is found, **User value matched** is displayed in green in the dialog box. See the following example figure.

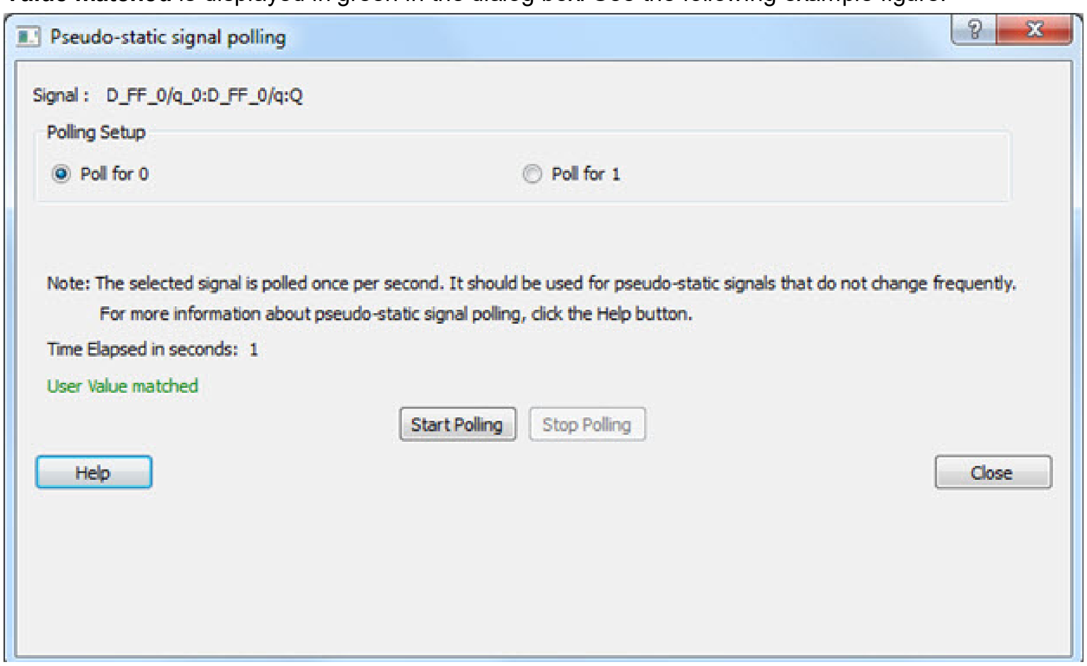


Figure 43 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - User Value matched

Vector Signal Polling

To poll vector signals, enter a value in the text box. The entered value is checked and validated. If an invalid value is entered, start polling is disabled, and an example displays showing the required format. See the following example figures.

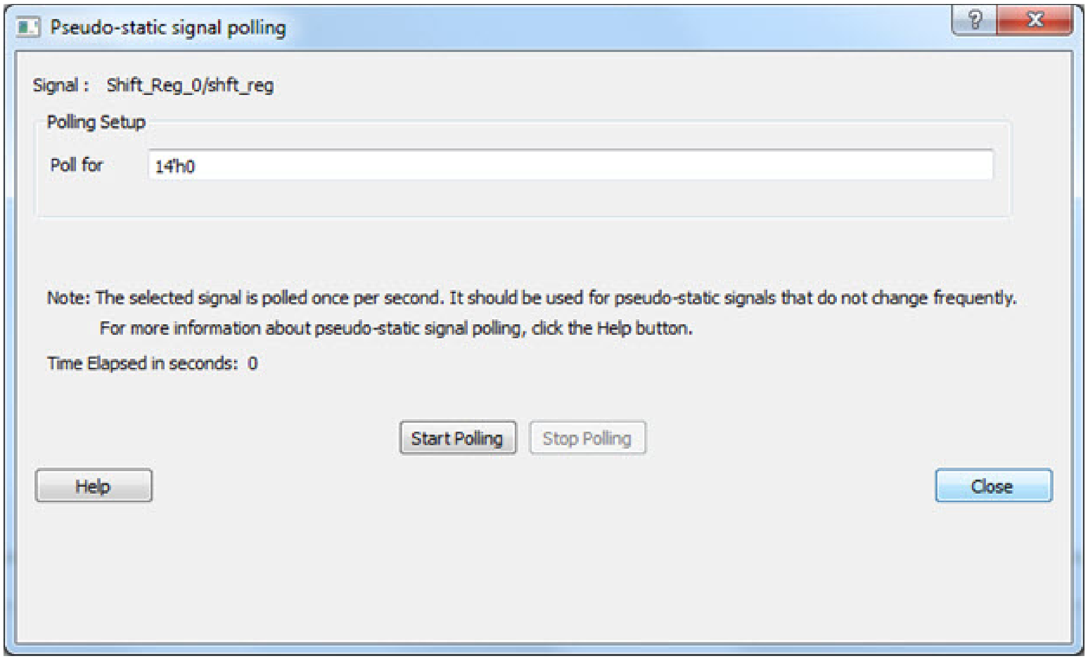


Figure 44 · Pseudo-static signal polling Dialog Box (Vector Signal Polling)

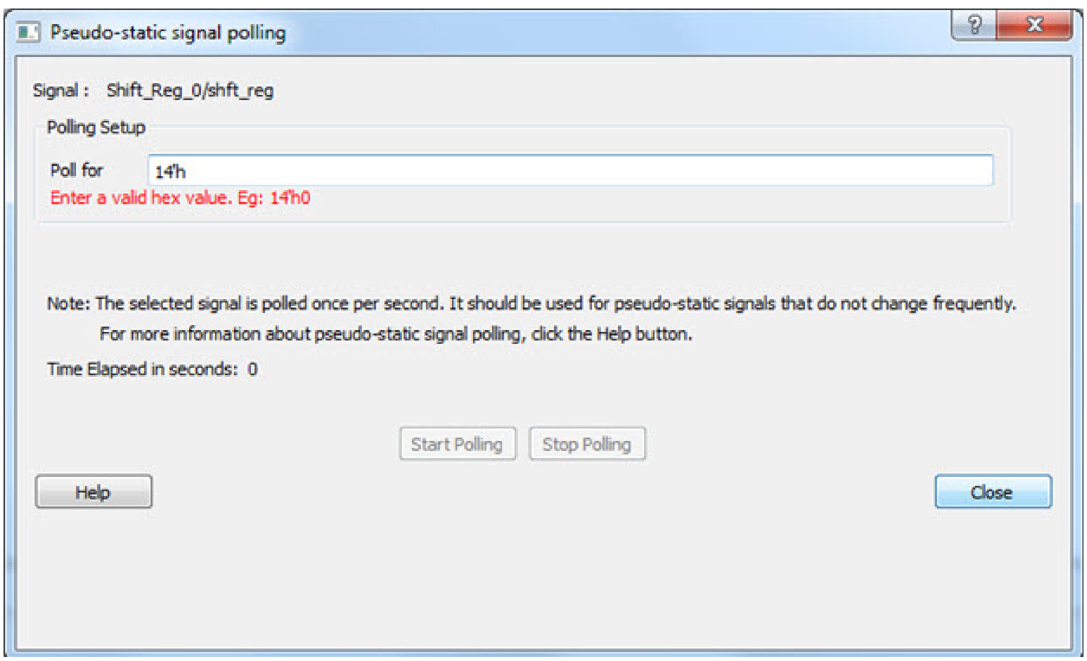


Figure 45 · Pseudo-static signal polling Dialog Box (Vector Signal Polling) -- After Validation

When you enter a valid value and click **Start Polling** is clicked, polling begins.

To end polling, click **Stop Polling**.

Note: You cannot change the poll value or close the polling dialog box while polling is in progress.

The elapsed time is updated in seconds until the polled value is found. When the polled value is found, **User value matched** is displayed in green in the dialog box.

Debug SERDES (SmartFusion2, IGLOO2, and RTG4)

You can examine and debug the SERDES blocks in your design in the Debug SERDES dialog box (shown in the figure below).

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

Debug SERDES Configuration is explained below. See the PRBS Test and Loopback Test topics for information specific to those procedures.

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

Debug SERDES - Configuration

Configuration Report

The Configuration Report output depends on the options you select in your PRBS Test and Loopback Tests. The default report lists the following for each Lane in your SERDES block:

Lane mode - Indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

PMA Ready - Indicates whether PMA has completed its internal calibration sequence for the specific lane and whether the PMA is operational. See the SmartFusion2 or IGLOO2 High Speed Serial Interfaces User Guide on the Microsemi website for details.

TxPll status - Indicates the loss-of-lock status for the TXPLL is asserted and remains asserted until the PLL reacquires lock.

RxPLL status - Indicates the CDR PLL frequency is not grossly out of range of with incoming data stream.

Click **Refresh Report** to update the contents of your SERDES Configuration Report. Changes to the specified SERDES register programming can be read back to the report.

SERDES Register Read or Write

Script - Runs Read/Write commands to access the SERDES control/status register map using a script. Enter the full pathname for the script location or click the browse button to navigate to your script file. Click **Execute** to run the script.

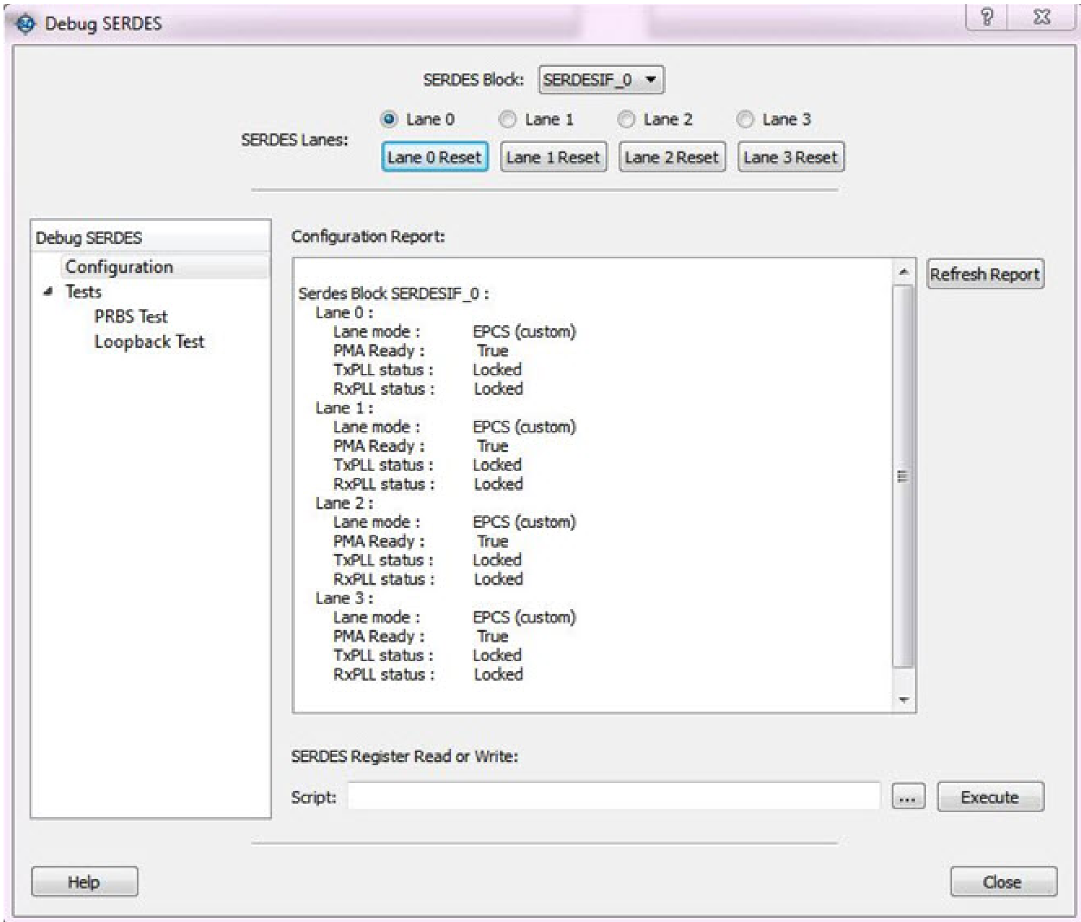


Figure 46 · Debug SERDES - Configuration

Note: The PCIe and XAUI protocols only support PRBS7. The EPCS protocol supports PRBS7/11/23/31.

Debug SERDES – Loopback Test

Loopback data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through predefined tests.

See the PRBS Test topic for more information about the PRBS test options.

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

SERDES Lanes

Select the **Lane** and **Lane Status** on which to run the Loopback test. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

Test Type

PCS Far End PMA RX to TX Loopback- This loopback brings data into the device and deserializes and serializes the data before sending it off-chip. This loopback requires 0PPM clock variation between the TX and RX SERDES clocks.

See the SmartFusion2 or IGLOO2 High Speed Serial Interfaces User's Guide on the Microsemi website for details.

Near End Loopback (On Die) - To enable, select the Near End Loopback (On Die) option and click **Start**. Click **Stop** to disable. Using this option allows you to send and receive user data without sending traffic off-chip. You can test design functionality without introducing other issues on the PCB.

See the SmartFusion2 or IGLOO2 High Speed Serial Interfaces User's Guide on the Microsemi website for details.

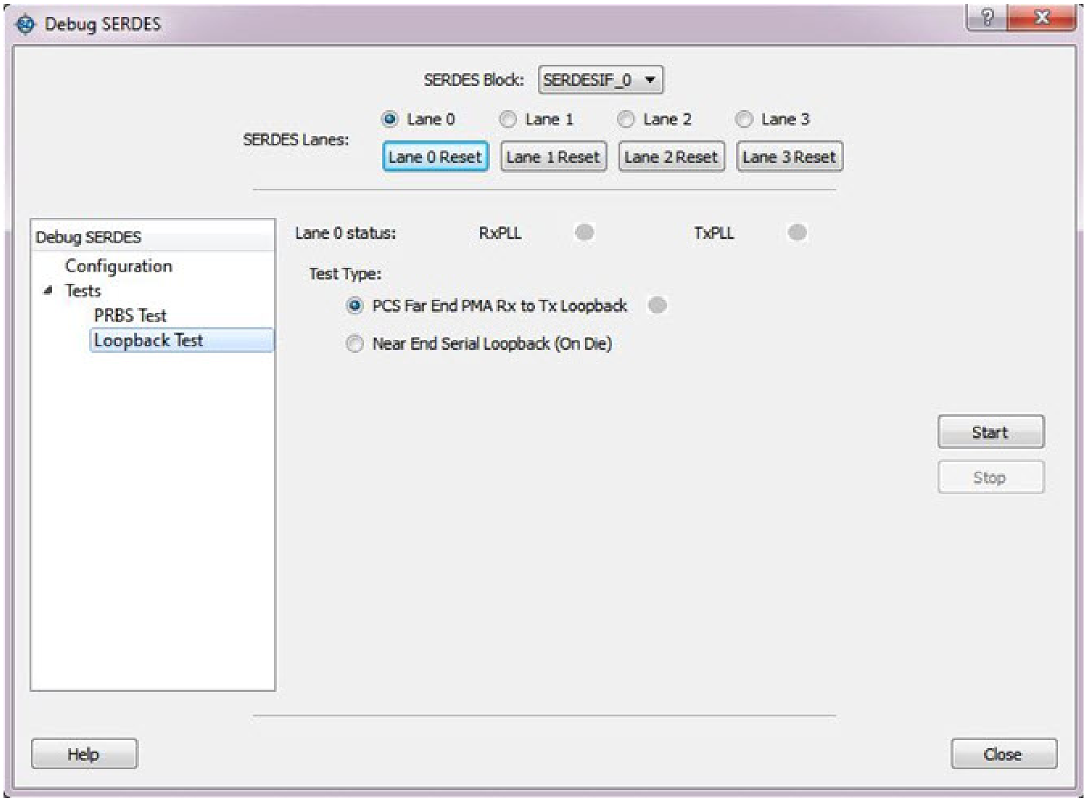
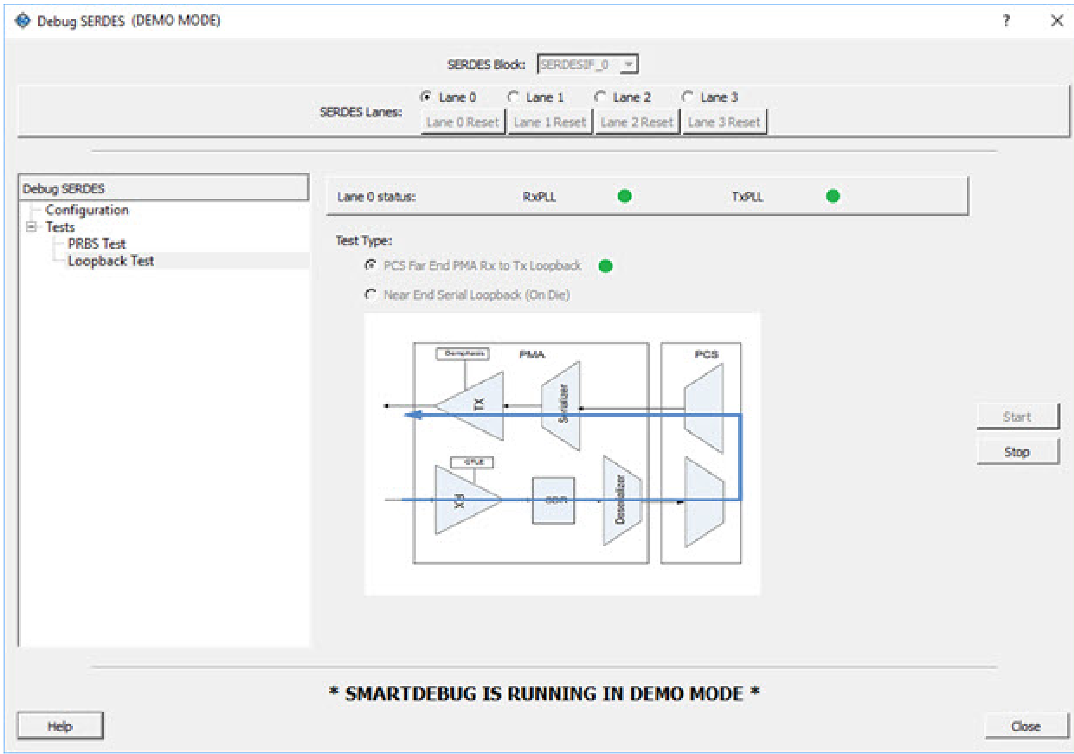


Figure 47 · Debug SERDES - Loopback Test

Running Loopback Tests in Demo Mode

You can run Loopback tests in demo mode. The SERDES demo mode is provided to demonstrate the GUI features of SERDES. All channels are enabled. Properly working channels and channels with connectivity issues are shown so you can see the available GUI options. See the following example figure.



Debug SERDES – PRBS Test

PRBS data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through several predefined patterns.

View Loopback Test settings in the Debug SERDES - Loopback Test topic.

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

SERDES Lanes

Check the box or boxes to select the lane(s) on which to run the PRBS test. Then select the Lane Status, test type, and pattern for each lane you have selected. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register. See the examples below.

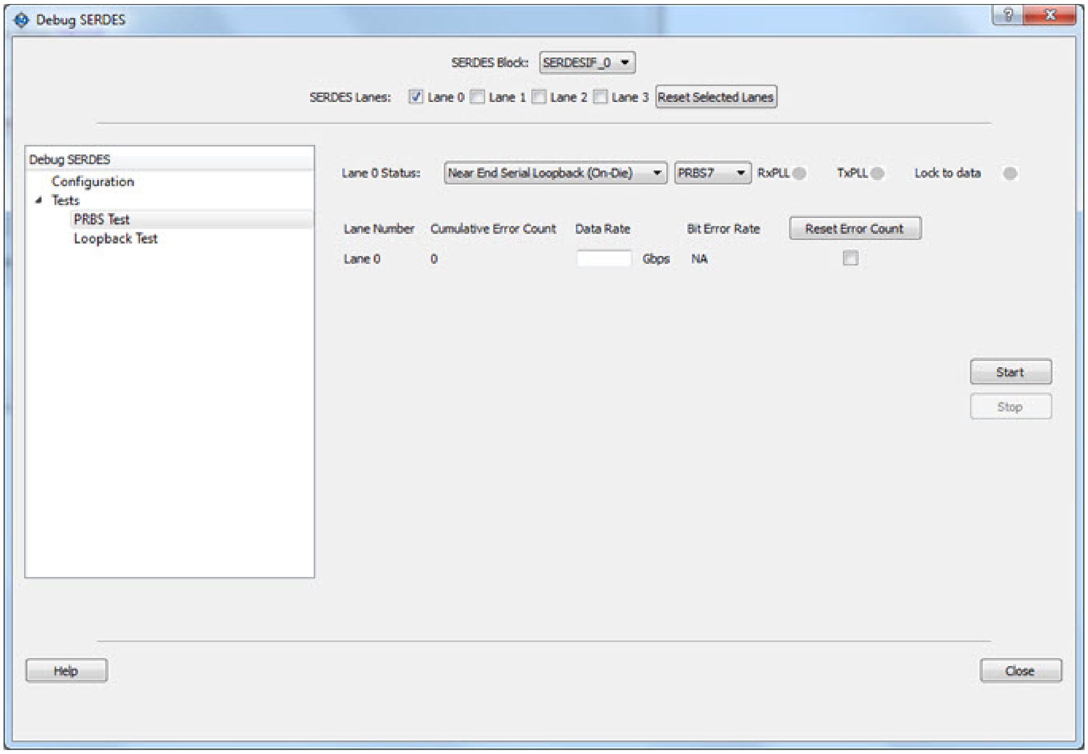


Figure 48 · SERDES Lanes - Single Lane Selected

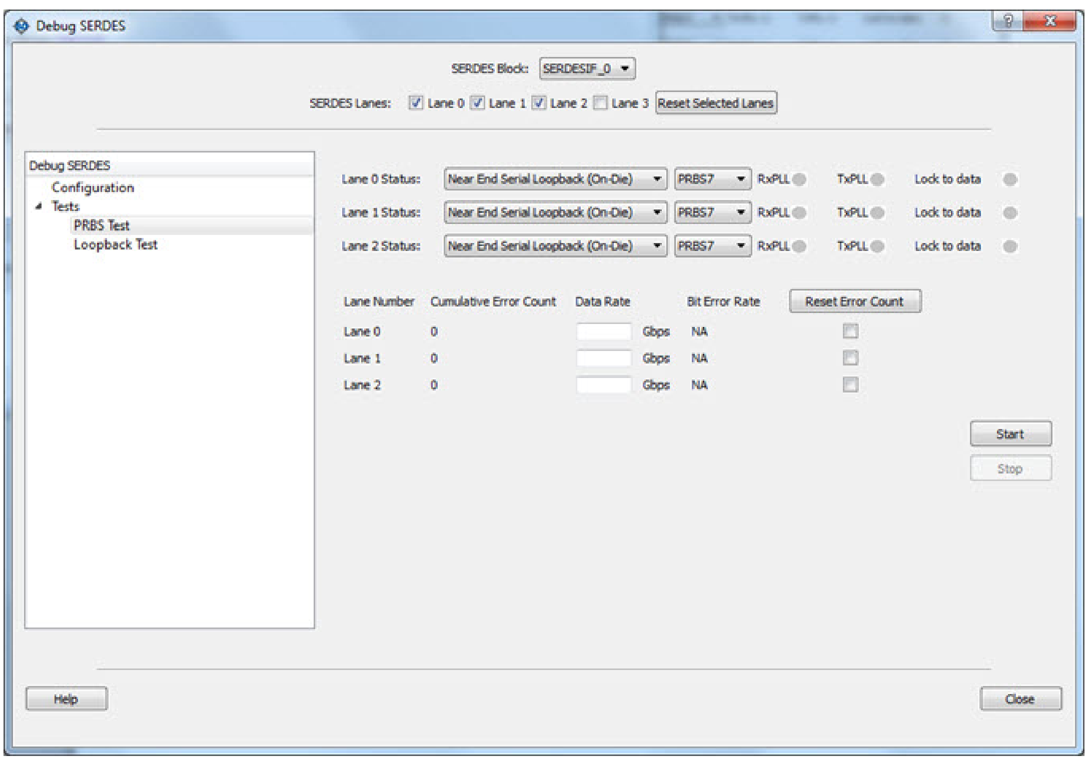


Figure 49 · SERDES Lanes - Multiple Lanes Selected

Test Type

Near End Serial Loopback (On-Die) enables a self-test of the device. The serial data stream is sent internally from the SERDES TX output and folded back onto the SERDES RX input.

Serial Data (Off-Die) is the normal system operation where the data stream is sent off chip from the TX output and must be connected to the RX input via a cable or other type of electrical interconnection.

If more than one SERDES Lane has been selected, the test type can be selected per lane. In the following example, Near End Serial Loopback (On-Die) has been selected for Lane 0 and Lane 3, and Serial Data (Off-Die) has been selected for Lane 1 and Lane 2.

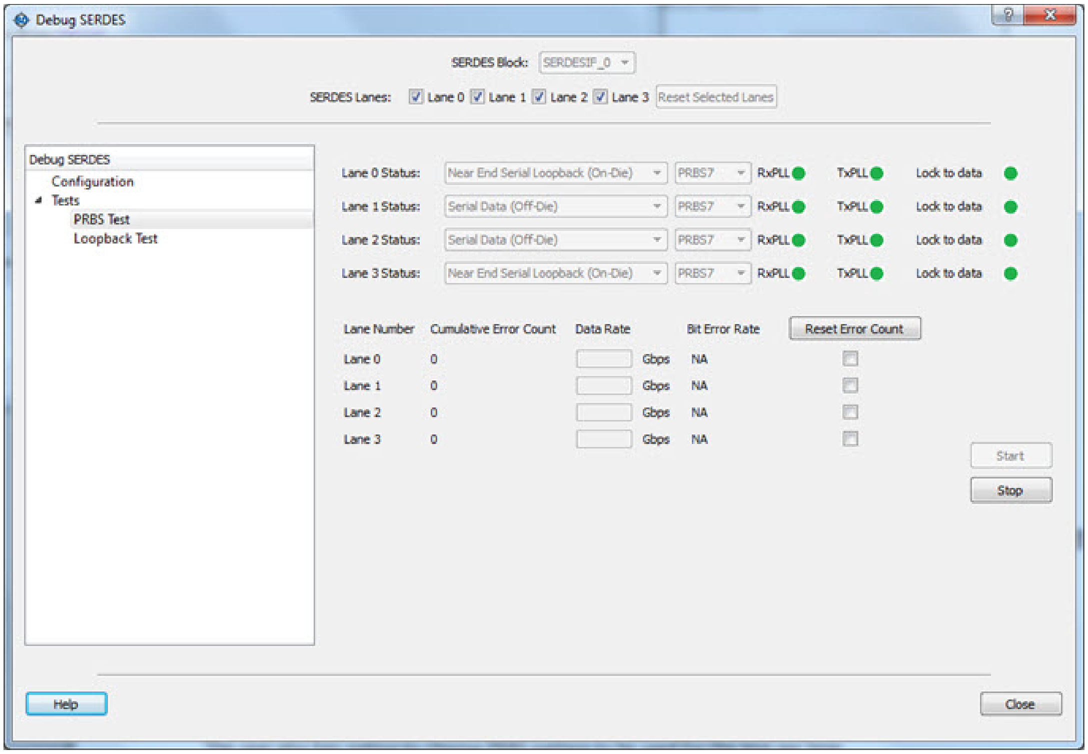


Figure 50 · Test Type Example

Pattern

The SERDESIF includes an embedded test pattern generator and checker used to perform serial diagnostics on the serial channel, as shown in the table below. If more than one lane is selected, the PRBS pattern can be selected per lane.

Pattern	Type
PRBS7	Pseudo-Random data stream of 2^7 polynomial sequences
PRBS11	Pseudo-Random data stream of 2^11 polynomial sequences
PRBS23	Pseudo-Random data stream of 2^23 polynomial sequences
PRBS31	Pseudo-Random data stream of 2^31 polynomial sequences

Cumulative Error Count

Lists the number of cumulative errors after running your PRBS test. To reset the error count to zero, select the lane(s) and click **Reset**. By default, Cumulative Error Count = 0, the Data Rate text box is blank, and Bit Error Rate = NA.

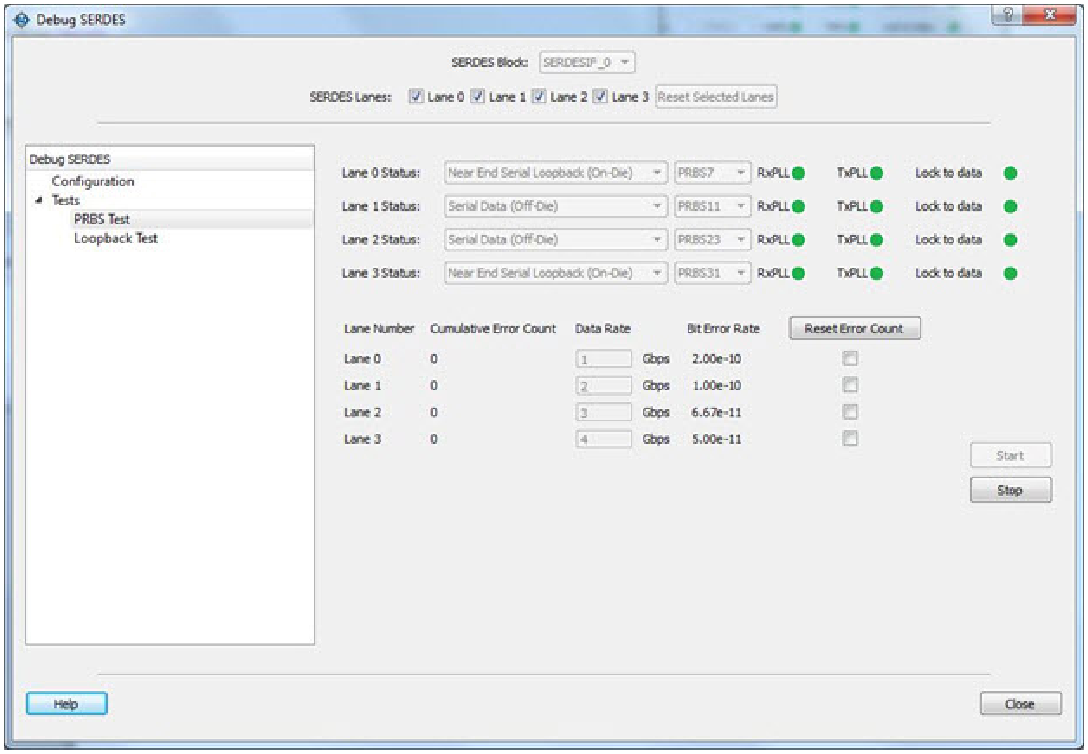


Figure 51 · Debug SERDES - PRBS Test

Note: If the design uses SERDES PCIe, PRBS7 is the only available option for PRBS tests.

Bit Error Rate

The Bit Error Rate is displayed per lane. If you did not specify a Data Rate, the Bit Error Rate displays the default NA. When the PRBS test is started, the Cumulative Error Count and Bit Error Rate are updated every second. You can select specific lanes and click **Reset Error Count** to clear the Cumulative Error Count and Bit Error Rate fields of the selected lanes.

In the example below, the Bit Error Rate is displayed for all lanes.

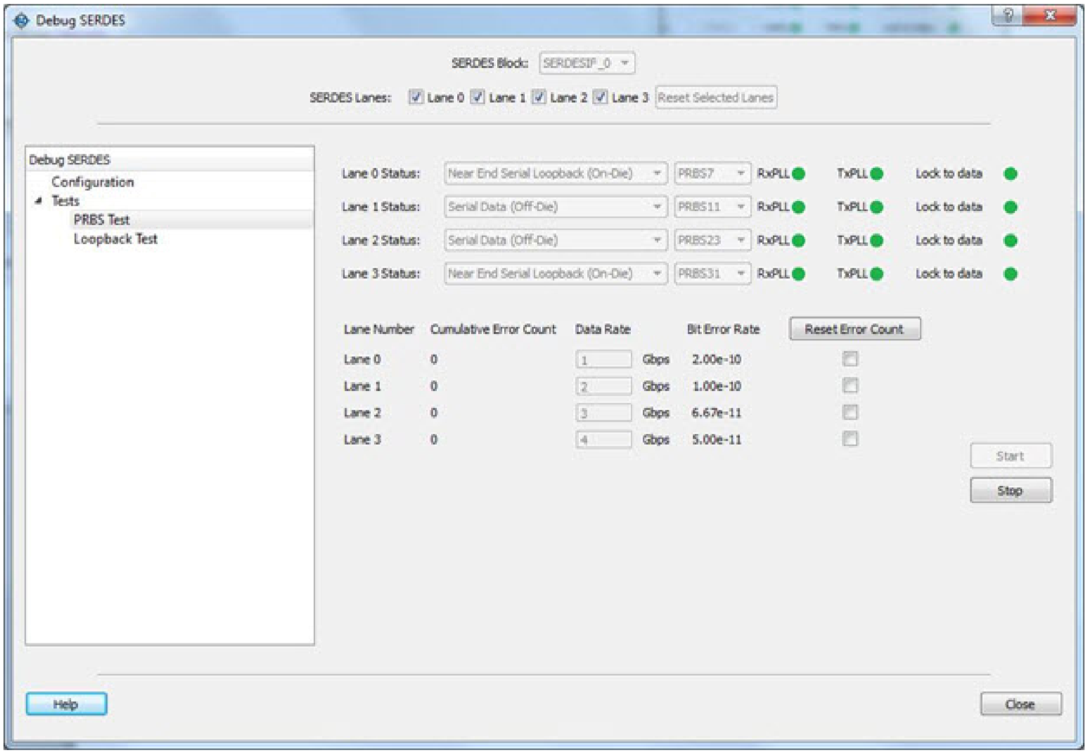


Figure 52 · Bit Error Rate Example - All Lanes

In the example below, Lane 1 and Lane 2 are selected and **Reset Error Count** is clicked.

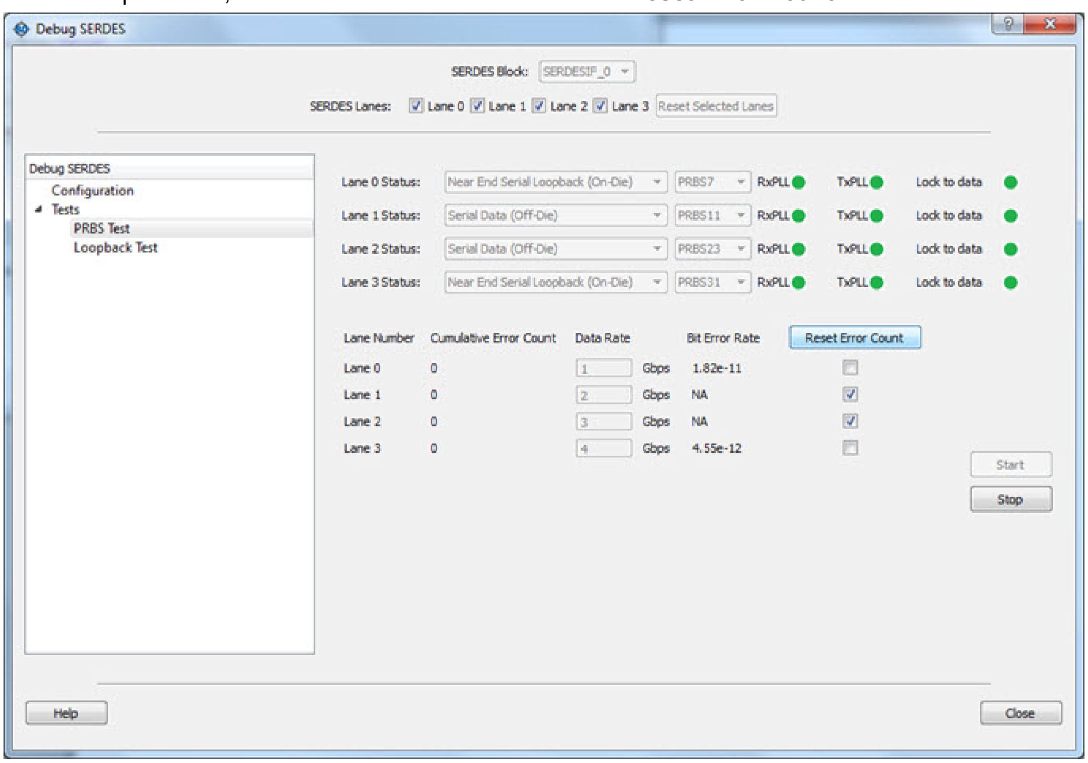
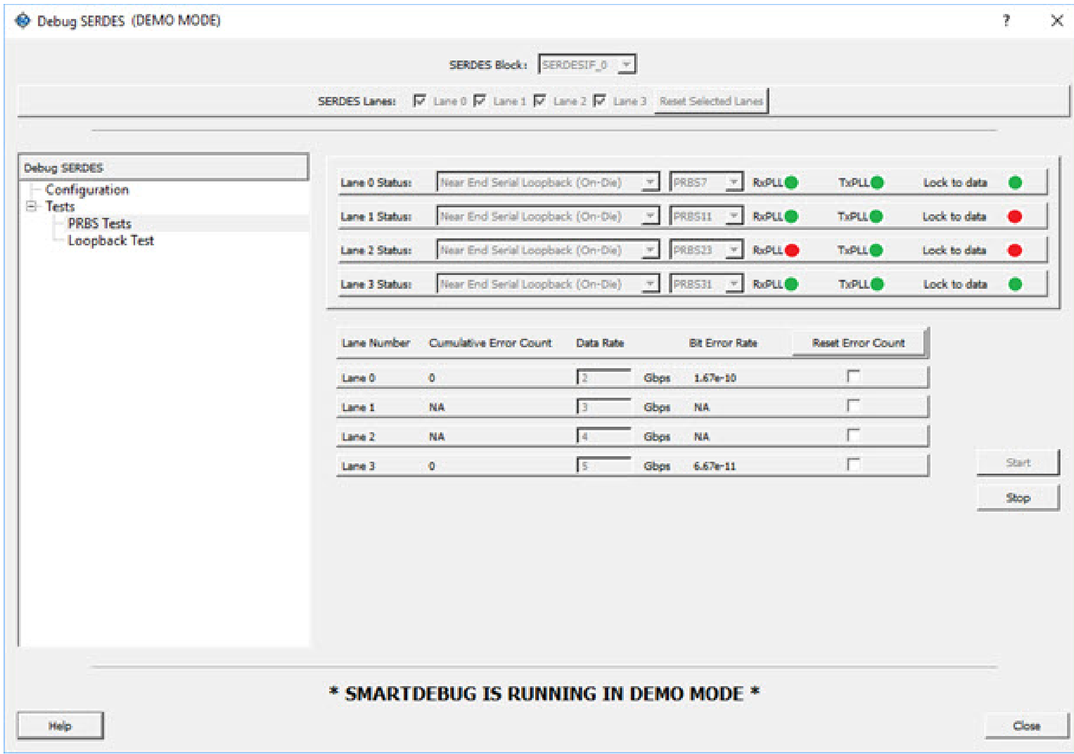


Figure 53 · Reset Error Count Example

Running PRBS Tests in Demo Mode

You can run Multi Lane PRBS tests in demo mode. The SERDES demo mode is provided to demonstrate the GUI features of SERDES. All channels are enabled. Properly working channels and channels with connectivity issues are shown so you can see the available GUI options. See the following example figure.



Notes:

The formula for calculating the BER is as follows:

BER = (#bit errors+1)/#bits sent

#bits sent = Elapsed time/bit period

When clicked on Start:

- The BER is updated every second for the entered data rate and errors observed.
- If no data rate is entered by the user, the BER is set to the default NA.

When clicked on Stop:

- The BER resets to default.

When clicked on Reset:

- The BER resets to default.
- If no test is in progress, the BER remains in the default value.
- If the PRBS test is in progress, the BER calculation restarts.

Debug SERDES – PHY Reset

SERDES PMA registers (for example, TX_AMP_RATIO) modified using a TCL script from the Configuration tab require a soft reset for the new values to be updated. Lane Reset for individual lanes achieves this functionality. Depending on the SERDES lanes used in the design, the corresponding Lane Reset buttons are enabled.

Lane Reset Behavior for SERDES Protocols Used in the Design

- EPCS: Reset is independent for individual lanes. Reset to Lane X (where X = 0,1,2,3) resets the Xth lane.
- PCIe: Reset to Lane X (where X = 0,1,2,3) resets all lanes present in the PCIe link and PCIe controller.

For more information about soft reset, refer to the SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide.

SmartDebug Tcl Support

Refer to the Libero SoC Tcl Commands Reference Guide for information about the Tcl commands supported by SmartDebug.

Frequently Asked Questions

Embedded Flash Memory (NVM) - Failure when Programming/Verifying

If the Embedded Flash Memory failed verification when executing the PROGRAM_NVM, VERIFY_NVM or PROGRAM_NVM_ACTIVE_ARRAY action, the failing page may be corrupted. To confirm and address this issue:

1. In the **Inspect Device** window click **View Flash Memory Content**.
2. Select the Flash Memory block and client (or page range) to retrieve from the device.
3. Click **Read from Device**; the retrieved data appears in the lower part of the window.
4. Click **View Detailed Status**.

   Note: Note: You can use the check_flash_memory and read_flash_memory Tcl commands to perform diagnostics similar to the commands outlined above.

5. To reset the corrupted NVM pages, either re-program the pages with your original data or 'zero-out' the pages by using the Tcl command recover_flash_memory.

If the Embedded Flash Memory failed verification when executing a VERIFY_NVM or VERIFY_NVM_ACTIVE_ARRAY action, the failure may be due to the change of content in your design. To confirm this, repeat steps 1-3 above.

Note: NVM corruption is still possible when writing from user design. Check NVM status for confirmation.

Analog System Not Working as Expected

If the Analog System is not working correctly, it may be due the following:

1. System supply issue. To troubleshoot:

- Physically verify that all the supplies are properly connected to the device and they are at the proper level. Then confirm by running the Device Status.
- Physically verify that the relevant channels are correctly connected to the device.

2. Analog system is not properly configured. You can confirm this by examining the Analog System.

ADC Not Sampling the Correct Value

If the ADC is sampling all zero values then the wrong analog pin may be connected to the system, or the analog pin is disconnected. If that is not the case and the ADC is not sampling the correct value, it may be due to the following:

1. System supply issues - Run the device status to confirm.
2. Analog system is not configured at all - To confirm, read out the ACM configuration and verify if the ACM content is all zero.
3. Analog system is not configured correctly - To confirm, read out the ACM configuration and verify that the configuration is as expected .

   Once analog block configuration has been confirmed, you can use the `sample_analog_channel` Tcl command for debug sampling of the analog channel with user-supplied sampling parameters.

   If you have access to your Analog System Builder settings project (<Libero IDE project>/Smartgen/AnalogBlock), you may use the compare function provided by the tool.

How do I unlock the device security so I can debug?

You must provide the PDB file with a User Pass Key in order to unlock the device and continue debugging.

If you do not have a PDB with User Pass Key, you can create a PDB file in FlashPro (if you know the Pass Key value).

How do I export a report?

You can export three reports from the SmartDebug GUI: Device Status, Client Detailed Status from the NVM, or the Compare Client Content report from the NVM. Each of those reports can be saved and printed.

For more information about Tcl commands supported by SmartDebug, see SmartDebug Tcl Commands.

How do I generate diagnostic reports for my target device?

A set of diagnostic reports can be generated for your target device depending on which silicon feature you are debugging. A set of Tcl commands are available to export those reports. The following is a summary of those Tcl commands based on the silicon features.

When using the –file parameter, ensure that you use a different file name for each command so you do not overwrite the report content. If you do not specify the –file option in the Tcl, the output results will be directed to the FlashPro log window.

For the overall device:

`read_device_status`

`read_id_code`

For FlashROM:

`compare_flashrom_client`

`read_flashrom`

For Embedded Flash Memory (NVM):

`compare_memory_client`

`check_flash_memory`

`read_flash_memory`

For Analog Block:

`read_analog_block_config`

`compare_analog_config`

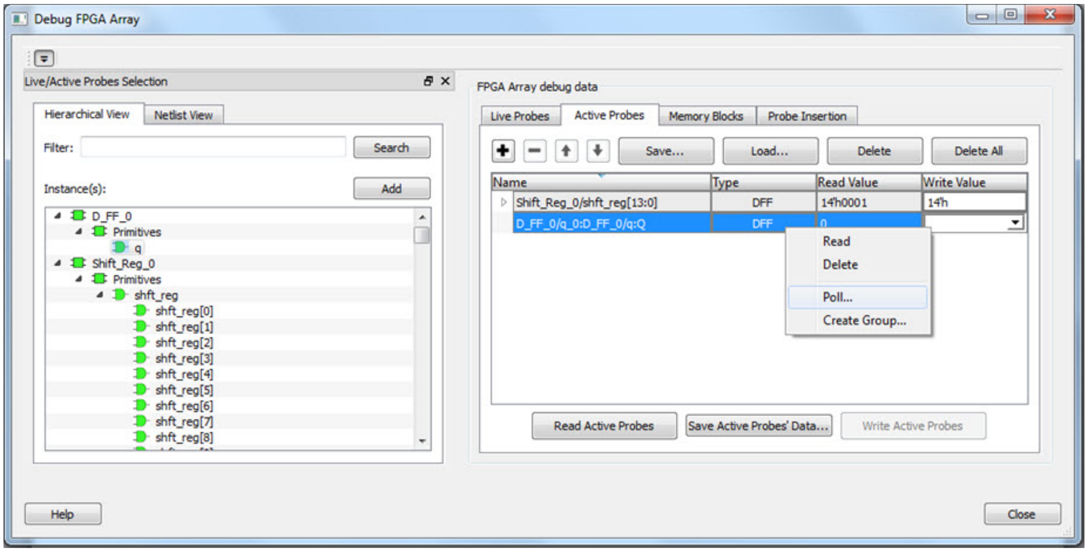
`sample_analog_channel`

To execute the Tcl command, from the **File** menu choose **Run Script**.

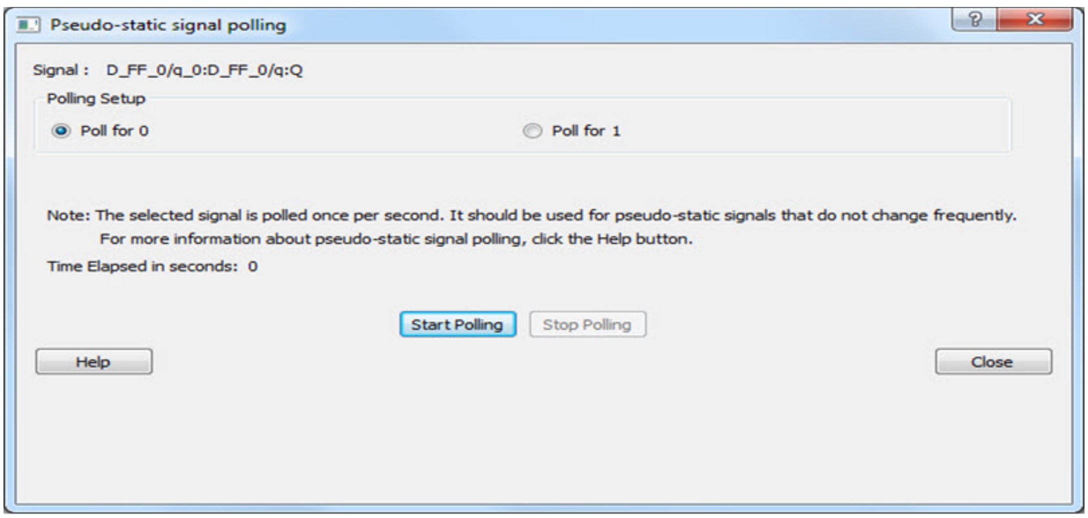
How do I monitor a static or pseudo-static signal?

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...**.



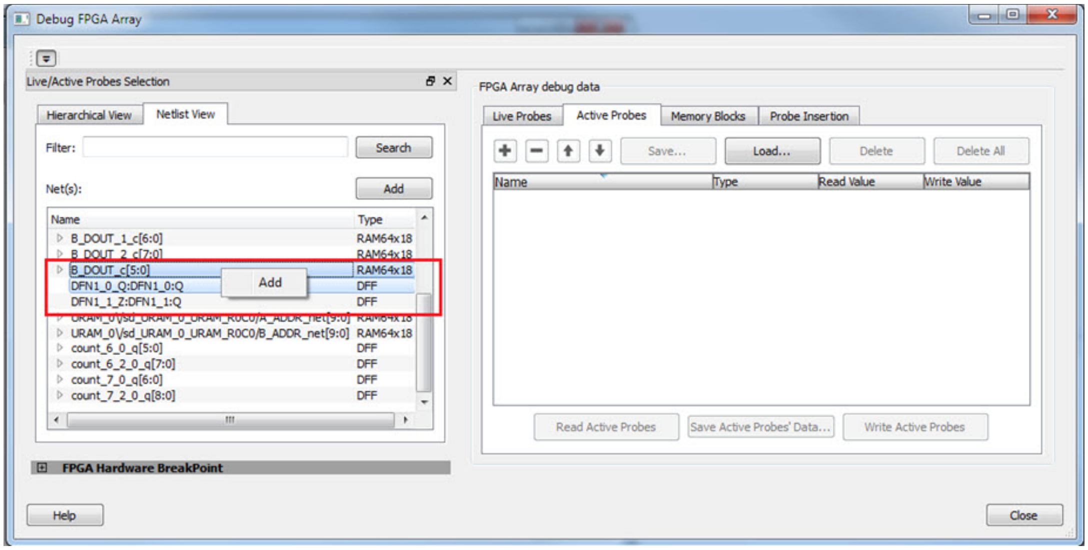
3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click **Start Polling**.



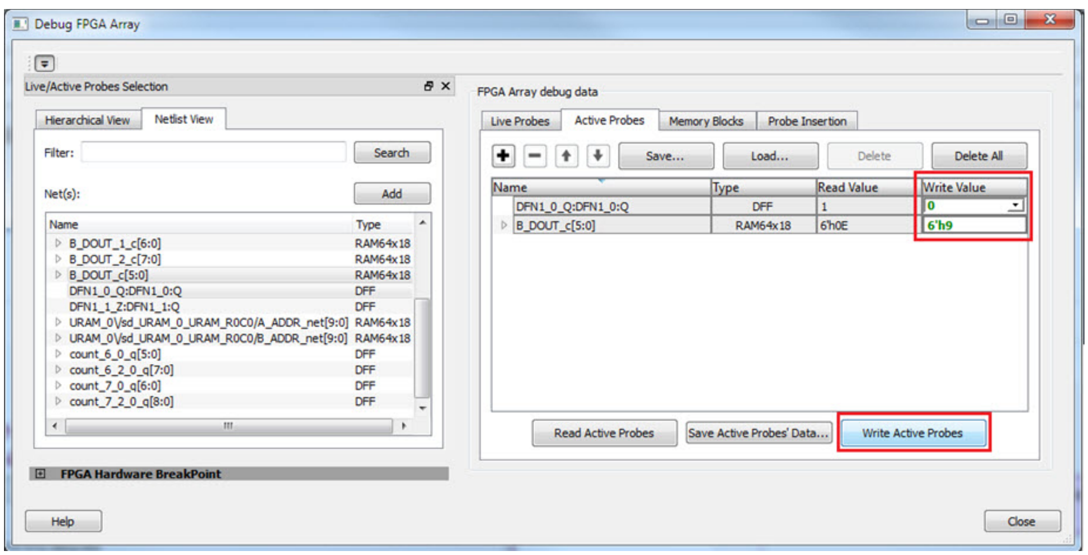
How do I force a signal to a new value?

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**.

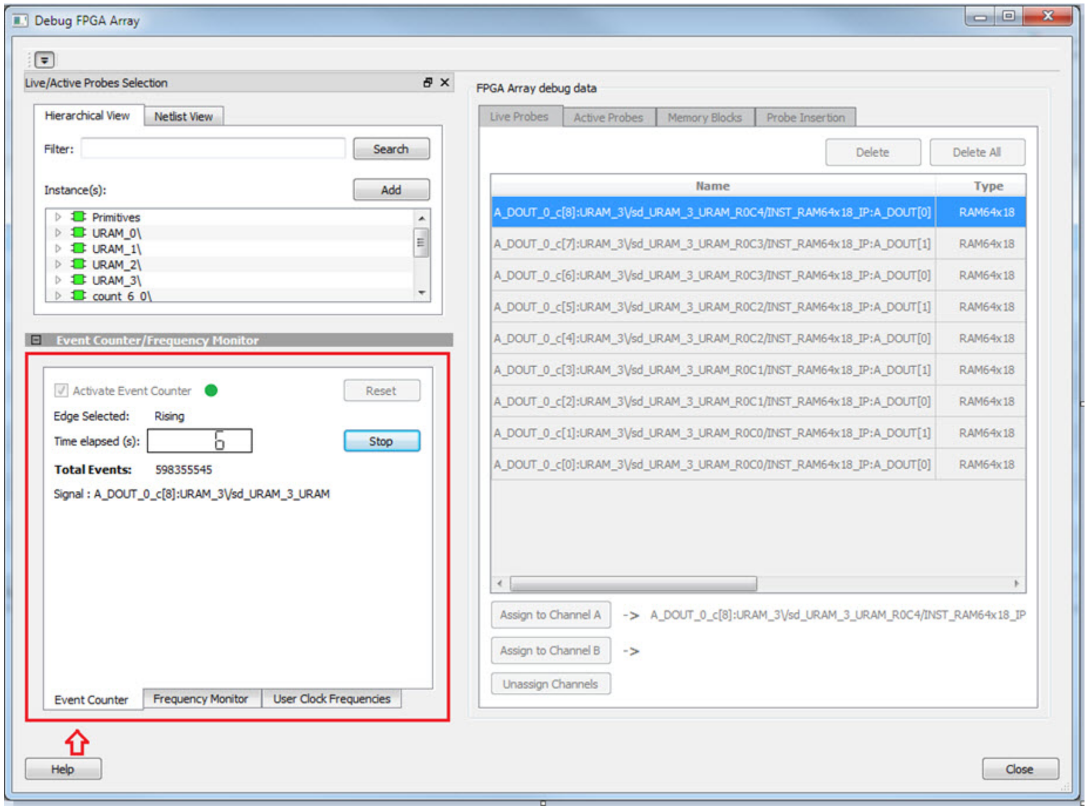


How do I count the transitions on a signal?

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

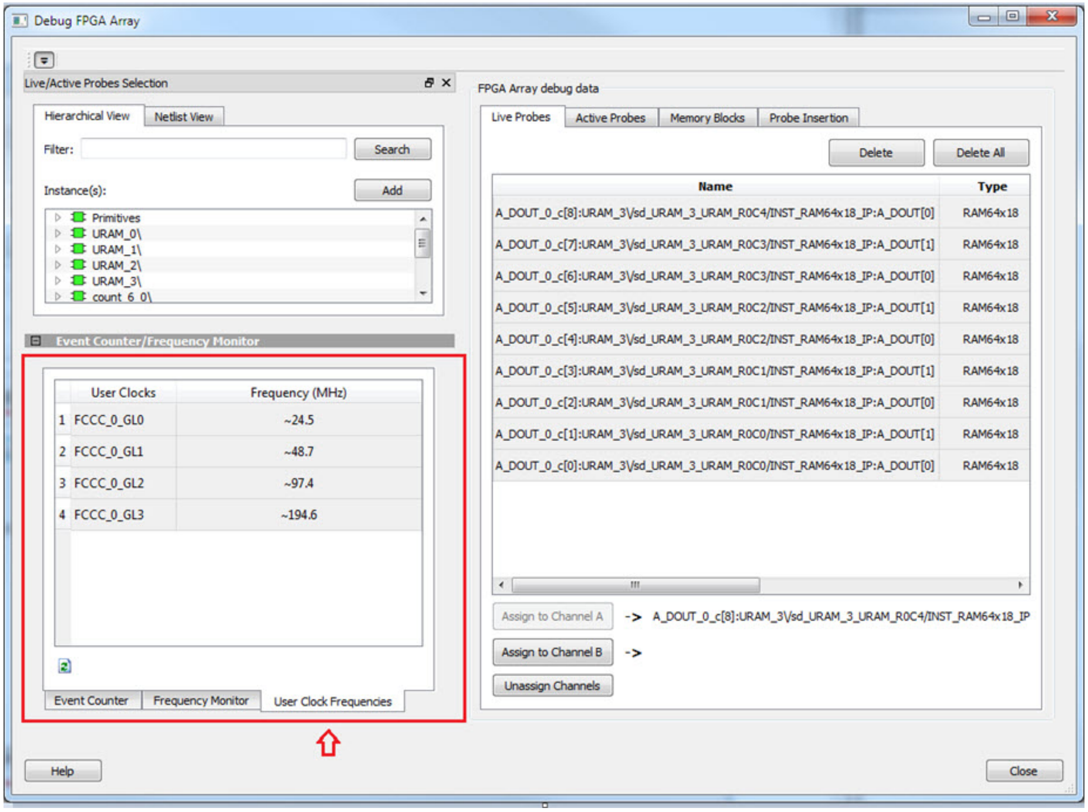
"Event Counter" on page 38

How do I monitor or measure a clock?

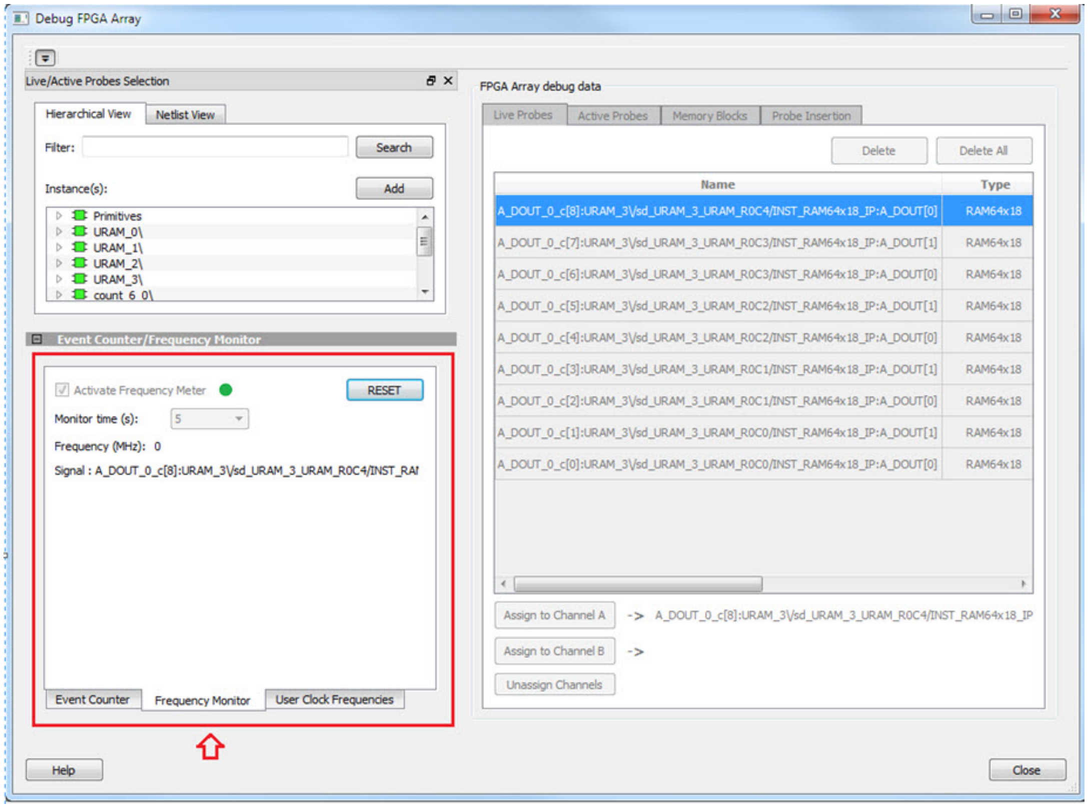
You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in Project Settings dialog box.

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 PRBS and loopback tests?

You can perform PRBS and loopback tests using the Debug SERDES option in SmartDebug.

To perform a PRBS test, in the Debug SERDES dialog box, select **PRBS Test** to run a PRBS test on-die or off-die For more information, see "Debug SERDES – PRBS Test" on page 59.

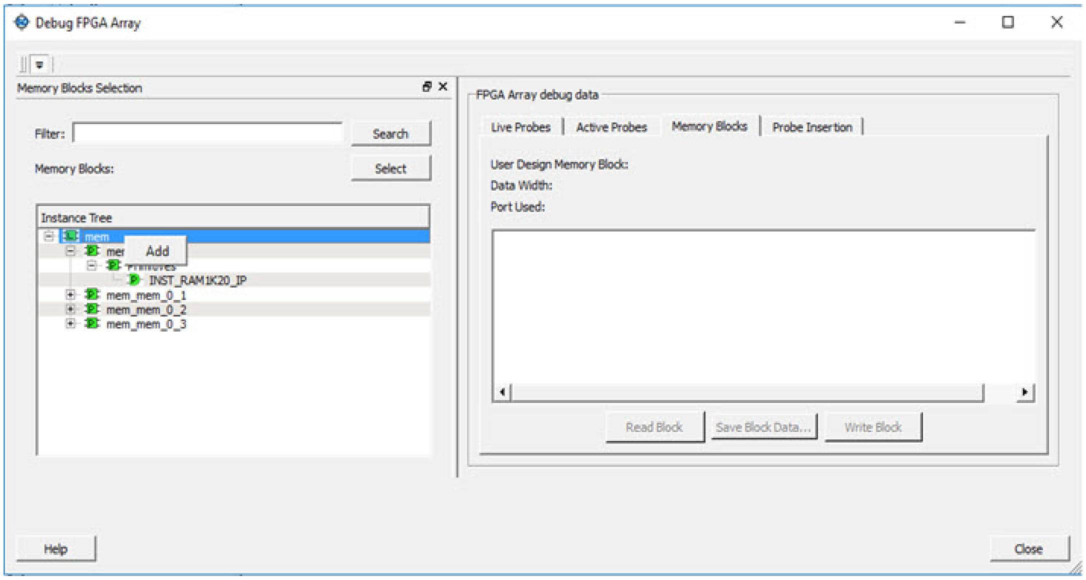
To perform a PRBS test, in the Debug SERDES dialog box, select PRBS Test to run a PRBS test on-die or off-die. For more information, see "Debug SERDES – PRBS Test" on page 59.

To perform a loopback test, in the Debug SERDES dialog box, select **Loopback Test** to run a near end serial loopback /far end PMA Rx to Tx loopback test. For more information, see "Debug SERDES – Loopback Test" on page 57.

How do I read LSRAM or USRAM content?

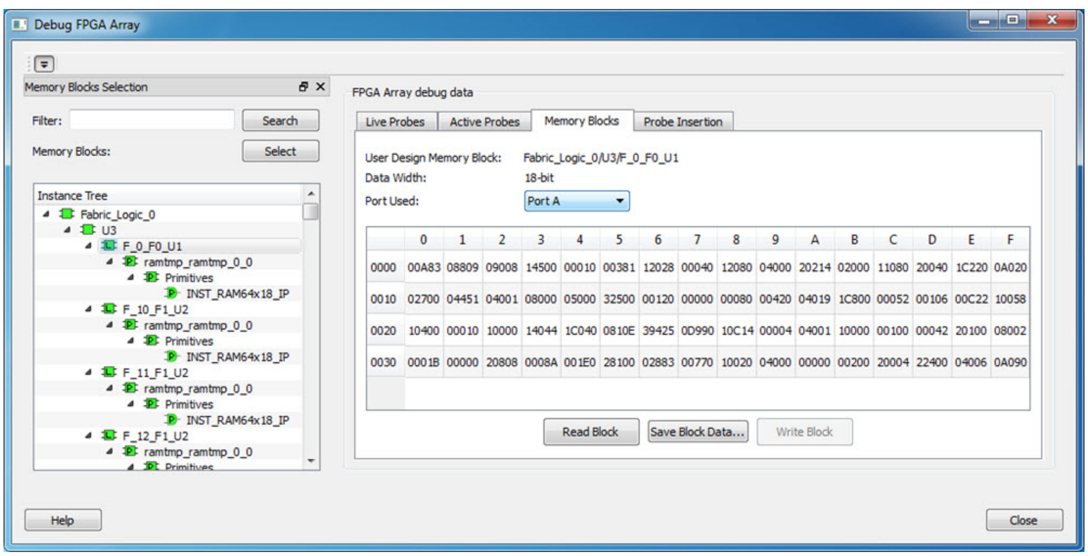
To read RAM content:

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

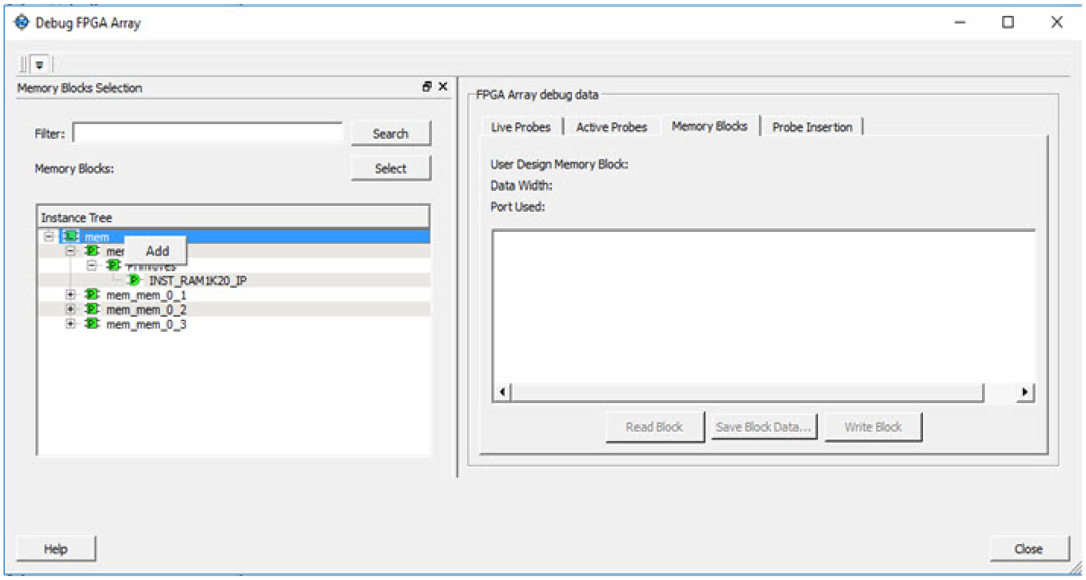
"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30

How do I change the content of LSRAM or USRAM?

To change the content of LSRAM or USRAM:

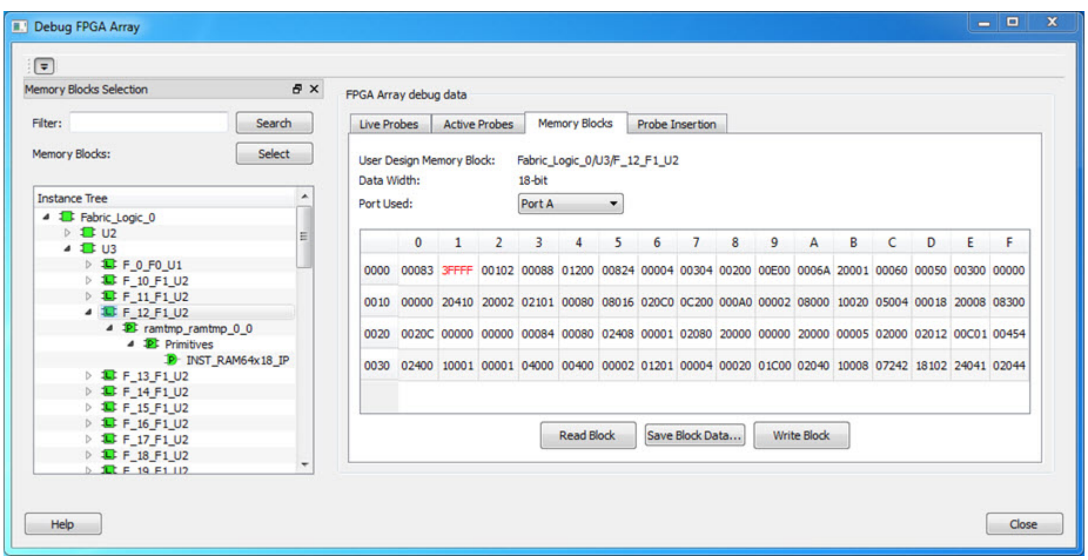
1. In the SmartDebug window, click **Debug FPGA Array**.

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.



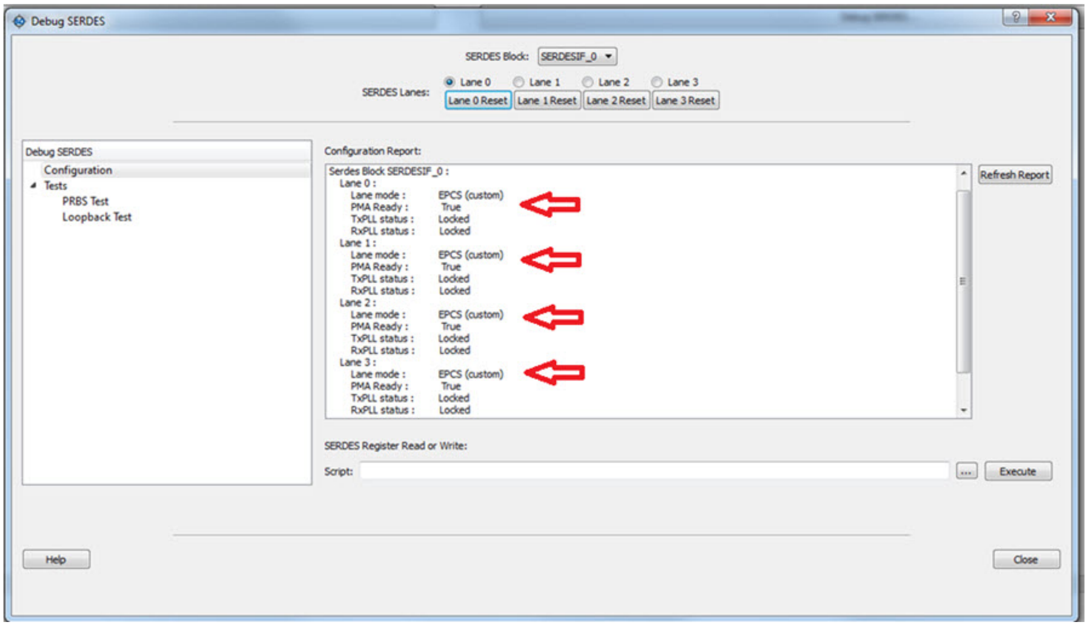
See Also

"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30

How do I read the health check of the SERDES?

You can read the SERDES health check using the following Debug SERDES options:

1. Review the **Configuration Report**, which returns PMA Ready, TxPLL status, and RxPLL status. For SERDES to function correctly, PMA ready should be true, and TxPLL and RxPLL status should be locked. The Configuration Report can be found in the Debug SERDES dialog box under Configuration. See Debug SERDES (SmartFusion2, IGLOO2, and RTG4).



2. Run the **PRBS Test**, which is a Near End Serial Loopback tests on selected lanes. This should result in 0 errors in the Cumulative Error Count column. See "Debug SERDES – PRBS Test" on page 59.

Where can I find files to compare my contents/settings?

FlashROM

You can compare the FlashROM content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

Embedded Flash Memory (NVM)

You can compare the Embedded Flash Memory content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

What is a UFC file? What is an EFC file?

UFC is the User FlashROM Configuration file, generated by the FlashROM configurator; it contains the partition information set by the user. It also contains the user-selected data for region types with static data.

However, for AUTO_INC and READ_FROM_FILE, regions the UFC file contains only:

- Start value, end value, and step size for AUTO_INC regions, and
- File directory for READ_FROM_FILE regions

EFC is the Embedded Flash Configuration file, generated by the Flash Memory Builder in the Project Manager Catalog; it contains the partition information and data set by the user.

Both UFC and EFC information is embedded in the PDB when you generate the PDB file.

Is my FPGA fabric enabled?

When your FPGA fabric is programmed, you will see the following statement under Device State in the Device Status report:

```
FPGA Array Status: Programmed and Enabled
```

If the FPGA fabric is not programmed, the Device State shows:

```
FPGA Array Status: Not Enabled
```

Is my Embedded Flash Memory (NVM) programmed?

To figure out if your NVM is programmed, read out and view the NVM content or perform verification with the PDB file.

To examine the NVM content, see the FlashROM Memory Content Dialog Box.

How do I display Embedded Flash Memory (NVM) content in the Client partition?

You must load your PDB into your FlashPro project in order to view the Embedded Flash Memory content in the Client partition. To view NVM content in the client partition:

1. Load your PDB into your FlashPro project.
2. Click **Inspect Device**.
3. Click **View Flash Memory Content**.
4. Choose a block from the drop-down menu.
5. Select a client.
6. Click **Read from Device**. The Embedded Flash Memory content from the device appears in the Flash Memory dialog box.

How do I know if I have Embedded Flash Memory (NVM) corruption?

When Embedded Flash Memory is corrupted, checking Embedded Flash Memory may return with any or all of the following page status:

- ECC1/ECC2 failure
- Page write count exceeds the 10-year retention threshold
- Page write count is invalid
- Page protection is set illegally (set when it should not be)

See the How do I interpret data in the Flash Memory (NVM) Status Report? topic for details.

If your Embedded Flash Memory is corrupted, you can recover by reprogramming with original design data. Alternatively, you can 'zero-out' the pages by using the Tcl command `recover_flash_memory`.

Why does Embedded Flash Memory (NVM) corruption happen?

Embedded Flash Memory corruption occurs when Embedded Flash Memory programming is interrupted due to:

- Supply brownout; monitor power supplies for brownout conditions. For SmartFusion monitor the VCC_ENVM/VCC_ROSC voltage levels; for Fusion, monitor VCC_NVM/VCC_OSC.
- Reset signal is not properly tied off in your design. Check the Embedded Memory reset signal.

How do I recover from Embedded Flash Memory corruption?

Reprogram with original design data or 'zero-out' the pages by using the Tcl command `recover_flash_memory`.

What is a JTAG IR-Capture value?

JTAG IR-Capture value contains private and public device status values. The public status value in the value read is ISC_DONE, which indicates if the FPGA Array is programmed and enabled.

The ISC_DONE signal is implemented as part of IEEE 1532 specification.

What does the ECC1/ECC2 error mean?

ECC is the Error Correction Code embedded in each Flash Memory page.

ECC1 – One bit error and correctable.

ECC2 – Two or more errors found, and not correctable.

What happens if invalid firmware is loaded into eNVM in SmartFusion2 devices?

When invalid firmware is loaded into eNVM in SmartFusion2 devices, Cortex-M3 will not be able to boot and issues reset to MSS continuously. eNVM content using View Flash Memory content will read zeroes in SmartDebug.

To verify that your FlashROM is programmed, read out and view the FlashROM content or perform verification with the PDB file by selecting the VERIFY or VERIFY_FROM action in FlashPro.

Can I compare serialization data?

To compare the serialization data, you can read out the FlashROM content and visually check data in the serialization region. Note that a serialization region can be an AUTO_INC or READ_FROM_FILE region.

For serialization data in the AUTO_INC region, check to make sure that the data is within the specified range for that region.

For READ_FROM_FILE region, you can search for a match in the source data file.

Can I tell what security options are programmed in my device?

To determine the programmed security settings, run the Device Status option from the Inspect Device dialog and examine the Security Section in the report.

This section lists the security status of the FlashROM, FPGA Array and Flash Memory blocks.

How do I interpret data in the Device Status report?

The Device Status Report generated from the FlashPro SmartDebug Feature contains the following sections:

- IDCode (see below)
- User Information
- Device State
- Factory Data
- Security Settings

How do I interpret data in the Flash Memory (NVM) Status Report?

The Embedded Flash Memory (NVM) Status Report generated from the FlashPro SmartDebug feature consists of the page status of each NVM page. For example:

```
Flash Memory Content [ Page 34 to 34 ]
FlashMemory Page #34:
Status Register(HEX): 00090000
Status ECC2 check: Pass
```

Data ECC2 Check: Pass

```
Write Count: Pass (2304 writes)
Total number of pages with status ECC2 errors: 0
Total number of pages with data ECC2 errors: 0
Total number of pages with write count out of range: 0
FlashMemory Check PASSED for [ Page 34 to 34 ]
The 'check_flash_memory' command succeeded.
The Execute Script command succeeded.
```

Table 1 · Embedded Flash Memory Status Report Description

Flash Memory Status Info	Description
Status Register (HEX)	Raw page status register captured from device
Status ECC2 Check	Check for ECC2 issue in the page status
Data ECC2 Check	Check for ECC2 issue in the page data
Write Count	Check if the page-write count is within the expected range. The expected write count is greater than or equal to: 6,384 - SmartFusion devices 2,288 - Fusion devices Note: Write count, if corrupted, cannot be reset to a valid value within the customer flow;invalid write count will not prevent device from being programmed with the FlashPro tool. The write count on all good eNVM pages is set to be 2288 instead of 0 in the manufacturing flow. The starting count of the eNVM is 2288. Each time the page is programmed or erased the count increments by one. There is a Threshold that is set to 12288, which equals to 3 * 4096. Since the threshold can only be set in multiples of 4096 (2^12), to set a 10,000 limit, the Threshold is set to 12288 and the start count is set to 2288; and thus the eNVM has a 10k write cycle limit. After the write count exceeds the threshold, the STATUS bit goes to 11 when attempting to erase/program the page.

Device Status Report

IDCode

The IDCode section shows the raw IDCode read from the device. For example, in the Device Status report for an AFS600 device, you will find the following statement:

```
IDCode (HEX): 233261cf
```

The IDCode is compliant to IEEE 1149.1. The following table lists the IDCode bit assignments:

Table 2 · IDCode Bit Assignments

Bit Field (little endian)	Example Bit Value for AFS600 (HEX)	Description
Bit [31-28] (4 bits)	2	Silicon Revision
Bit [27-12] (16 bits)	3326	Device ID
Bit [11-0] (12 bits)	1cf	IEEE 1149.1 Manufacturer ID for Microsemi

User Info

The User Information section reports the information read from the User ROW (UROW) of IGLOO, ProASIC3, SmartFusion and Fusion devices. The User Row includes user design information as well as troubleshooting information, including:

- Design name (10 characters max)
- Design check sum (16-bit CRC)
- Last programming setup used to program/erase any of the silicon features.
- FPGA Array / Fabric programming cycle count

For example:

```
User Information:
UROW data (HEX): 603a04e0a1c2860e59384af926fe389f
Programming Method: STAPL
Programmer: FlashPro3
Programmer Software: FlashPro vX.X
Design Name: ABCBASICTO
Design Check Sum: 603A
Algorithm Version: 19
Array Prog. Cycle Count: 19
```

Table 3 · Device Status Report User Info Description

Category	Field	Description
User Row Data	(Example) UROW data (HEX): 603a04e0a1c2860e59384af926fe389f	Raw data from User Row (UROW)

Category	Field	Description
Programming Troubleshooting Info	(Example) Programming Method: STAPL Programmer: FlashPro3 Programmer Software: FlashPro v8.6 Algorithm Version: 19	Known programming setup used. This includes: Programming method/file, programmer and software. It also includes programming Algorithm version used.
Design Info	(Example) Design Name: ABCASICTO Design Check Sum: 603A	Design name (limited to 10 characters) and check sum. Design check sum is a 16-bit CRC calculated from the fabric (FPGA Array) datastream generated for programming. If encrypted datastream is generated selected, the encrypted datastream is used for calculating the check sum.

Device State

The device state section contains:

- IR-Capture register value, and
- The FPGA status

The IR-Capture is the value captured by the IEEE1149.1 instruction register when going through the IR-Capture state of the IEEE 1149.1 state machine. It contains information reflecting some of the states of the devices that is useful for troubleshooting.

One of the bits in the value captured is the ISC_DONE value, specified by IEEE 1532 standard. When the value is ‘1’ it means that the FPGA array/fabric is programmed and enabled. This is available for IGLOO, ProASIC3, SmartFusion and Fusion devices.

For example:

```
Device State:
IRCapture Register (HEX): 55
FPGA Array Status: Programmed and enabled
```

For a blank device:

```
Device State:
IRCapture Register (HEX): 51
FPGA Array Status: Not enabled
```

Factory Data

The Factory Data section lists the Factory Serial Number (FSN).

Each of the IGLOO, ProASIC3, SmartFusion and Fusion devices has a unique 48-bit FSN.

Security

The security section shows the security options for the FPGA Array, FlashROM and Flash Memory (NVM) block that you programmed into the device.

For example, using a Fusion AFS600 device:

```
Security:
```

```
Security Register (HEX): 0000000088c01b
FlashROM
Write/Erase protection: Off
Read protection: Off
Encrypted programming: Off
FPGA Array
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: Off
FlashMemory Block 0
Write protection: On
Read protection: On
Encrypted programming: Off
FlashMemory Block 1
Write protection: On
Read protection: On
Encrypted programming: Off
```

Table 4 · Device Status Report - Security Description

Security Status Info	Description
Security Register (HEX)	Raw data captured from the device's security status register
Write/Erase Protection	Write protection is applicable to FlashROM, FPGA Array (Fabric)and Flash Memory (NVM) blocks. When On, the Silicon feature is write/erase protected by user passkey.
Read Protection	Read protection is applicable to FlashROM and Flash Memory (NVM) blocks. When On, the Silicon feature is read protected by user passkey.
Verify Protection	Verify Protection is only applicable to FPGA Array (Fabric) only. When On, the FPGA Array require user passkey for verification. Reading back from the FPGA Array (Fabric) is not supported. Verification is accomplished by sending in the expected data for verification.
Encrypted Programming	Encrypted Programming is supported for FlashROM, FPGA Array (Fabric) and Flash Memory (NVM) blocks. When On, the silicon feature is enable for encrypted programmed. This allows field design update with encrypted datastream so the user design is protected.

Encrypted Programming

To allow encrypted programming of the features, the target feature cannot be Write/Erase protected by user passkey.

The security settings of each silicon feature when they are enabled for encrypted programming are listed below.

FPGA Array (Fabric)

```
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: On
```

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FPGA Array (Fabric). This setting allows the FPGA Array (Fabric) to be programmed and verified with an encrypted datastream.

FlashROM

```
Write/Erase protection: Off
Read protection: On
Encrypted programming: On
```

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FlashROM. This setting allows the FlashROM to be programmed and verified with an encrypted datastream.

FlashROM always allows verification. If encrypted programming is set, verification has to be performed with encrypted datastream.

Designer and FlashPro automatically set the FlashROM to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

Flash Memory (NVM) Block

```
Write/Erase protection: Off
Read protection: On
Encrypted programming: On
```

The above setting is set automatically set by Designer or FlashPro when you select to enable encrypted programming of the Flash Memory (NVM) block. This setting allows the Flash Memory (NVM) block to be programmed with an encrypted datastream.

The Flash Memory (NVM) block does not support verification with encrypted datastream.

Designer and FlashPro automatically set the Flash Memory (NVM) block to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.