



# Libero® SoC v2021.1

---

## SmartDebug User Guide for PolarFire® FPGAs

---

### Introduction

---

Design debug is a critical phase of FPGA design flow. Microchip's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug provides access to non-volatile memory (sNVM), SRAM, transceiver, µPROM, and probe capabilities. Microchip PolarFire® FPGA devices have built-in probe logic that greatly enhance the ability to debug logic elements within the device.

SmartDebug accesses the built-in probe points through the Active Probe and Live Probe features, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

**Note:** For SCB read operations, if APB DRI bus performs an SCB read operation while SmartDebug tries to read from SCB, the data may get corrupted. If the PolarFire controller initiates a polled read, it polls for SCB read done register. After it acknowledges the done operation, it takes several CPU cycles to transfer the data. If the APB DRI interface initiates an SCB read operation during the transfer, the stored data gets corrupted and SmartDebug may read corrupted data.

## Table of Contents

1. Getting Started with SmartDebug.....	4
1.1. Use Models.....	4
1.2. Creating a Standalone SmartDebug Project.....	4
1.3. Configuring a Generic Device.....	6
2. SmartDebug User Interface.....	7
2.1. Standalone SmartDebug User Interface.....	7
2.2. Programming Connectivity and Interface.....	8
2.3. View Device Status.....	12
3. Debugging.....	15
3.1. Debug FPGA Array.....	15
3.2. Hierarchical View.....	15
3.3. Netlist View.....	17
3.4. Live Probes.....	18
3.5. Active Probes.....	19
3.6. Probe Grouping (Active Probes Only).....	20
3.7. Memory Blocks.....	23
3.8. Probe Insertion (Post-Layout).....	32
3.9. FPGA Hardware Breakpoint Auto Instantiation.....	35
3.10. Event Counter.....	41
3.11. Frequency Monitor.....	44
3.12. User Clock Frequencies.....	49
3.13. Debug sNVM.....	50
3.14. Debug Transceiver.....	54
3.15. MSS Register Access.....	83
3.16. Debug IOD.....	86
3.17. Debug UPROM.....	88
3.18. Debug DDR IO Margin.....	92
3.19. Debug eNVM.....	106
4. SmartDebug Tcl Commands.....	110
5. Frequently Asked Questions.....	111
5.1. How do I monitor a static or pseudo-static signal?.....	111
5.2. How do I force a signal to a new value?.....	111
5.3. How do I count the transitions on a signal?.....	112
5.4. How do I monitor or measure a clock?.....	113
5.5. How do I perform simple SmartBERT tests?.....	115
5.6. How do I read LSRAM or USRAM content?.....	115
5.7. How do I change the content of LSRAM or USRAM?.....	116
5.8. How do I read the health check of the Transceiver?.....	117
6. Revision History.....	119
7. Microchip FPGA Technical Support.....	120
7.1. Customer Service.....	120

7.2. Customer Technical Support..... 120

7.3. Website..... 120

7.4. Outside the U.S..... 120

The Microchip Website..... 121

Product Change Notification Service..... 121

Customer Support..... 121

Microchip Devices Code Protection Feature..... 121

Legal Notice..... 122

Trademarks..... 122

Quality Management System..... 123

Worldwide Sales and Service..... 124

## 1. Getting Started with SmartDebug

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

For more information on how to use the debugger to gather the device status and to view the diagnostics, see [Get device status and view diagnostics](#).

### 1.1 Use Models

SmartDebug can be run in the following modes:

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

#### 1.1.1 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**.

#### 1.1.2 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 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.

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.

**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.

#### 1.1.3 Demo Mode

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

**Note:** SmartDebug demo mode is for demonstration purposes only, and does not provide the functionality of integrated mode or standalone mode. You cannot switch between demo mode and normal mode while SmartDebug is running.

### 1.2 Creating a Standalone SmartDebug Project

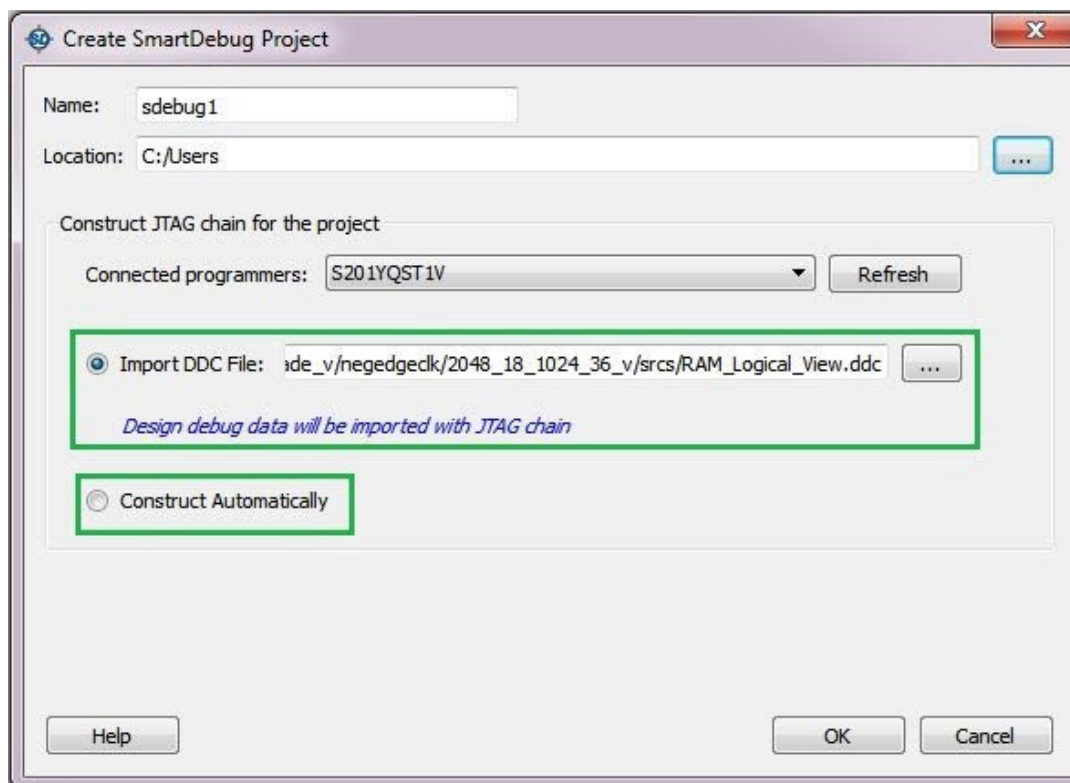
A standalone SmartDebug project can be configured in two ways:

- Import DDC files exported from Libero

- Construct Automatically

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

**Figure 1-1. Create SmartDebug Project Dialog Box**



### 1.2.1 Import DDC File (created from Libero)

When you select the **Import 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 Microchip 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**. Click **Export SmartDebug Data** to export a new DDC file and use it for project creation.

### 1.2.2 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.

### 1.2.3 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](#)

## **1.3 Configuring a Generic Device**

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

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

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

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

## 2. SmartDebug User Interface

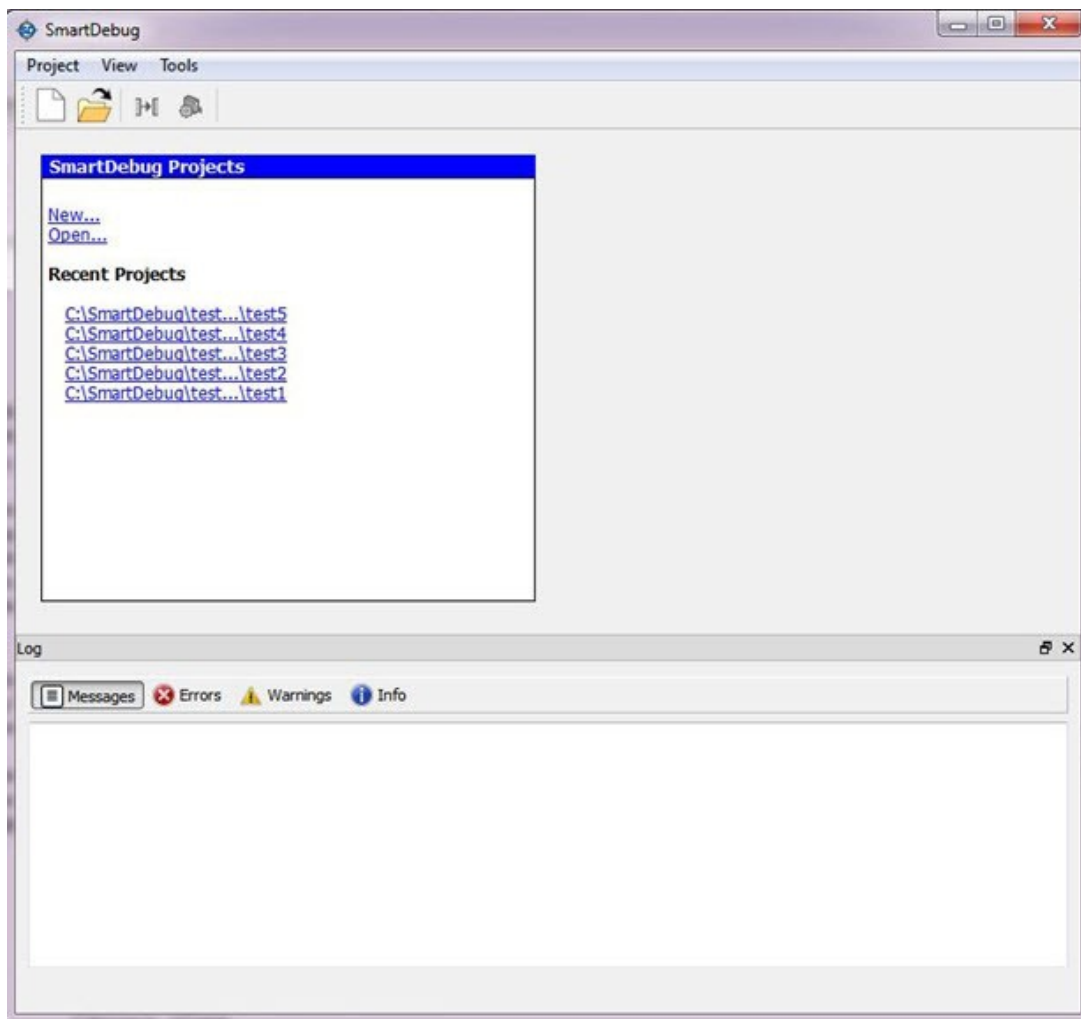
This topic introduces the basic elements and features of SmartDebug.

### 2.1 Standalone SmartDebug User Interface

You can launch the standalone SmartDebug from the Libero installation folder or from the FlashPRO installation folder on the following operating systems:

- Windows:
  - <Libero Installation folder>/Designer/bin/sdebug.exe
  - <FlashPRO Installation folder>/bin/sdebug.exe
- Linux:
  - <Libero Installation folder>/ bin/sdebug
  - <FlashPRO Installation folder>/bin/sdebug

Figure 2-1. Standalone SmartDebug Main Window



#### 2.1.1 Project Menu

The following table describes the menu options in the **Project** menu.

**Table 2-1. Project Menu**

Menu Option	Description
New Project	To create a new SmartDebug project.
Open Project	To open an existing debug project.
Execute Script	To execute SmartDebug-specific Tcl scripts.
Export Script File	To export SmartDebug-specific commands to a script file.
Recent Projects	To see a list of recent SmartDebug projects.

### 2.1.2 Log Window

The following table describes the various tabs on the **Log** view in the standalone SmartDebug window.

**Table 2-2. Log View Tabs**

Tab	Description
Messages	Displays standard output messages.
Errors	Displays error messages.
Warnings	Displays warning messages.
Info	Displays general information.

### 2.1.3 Tools Menu

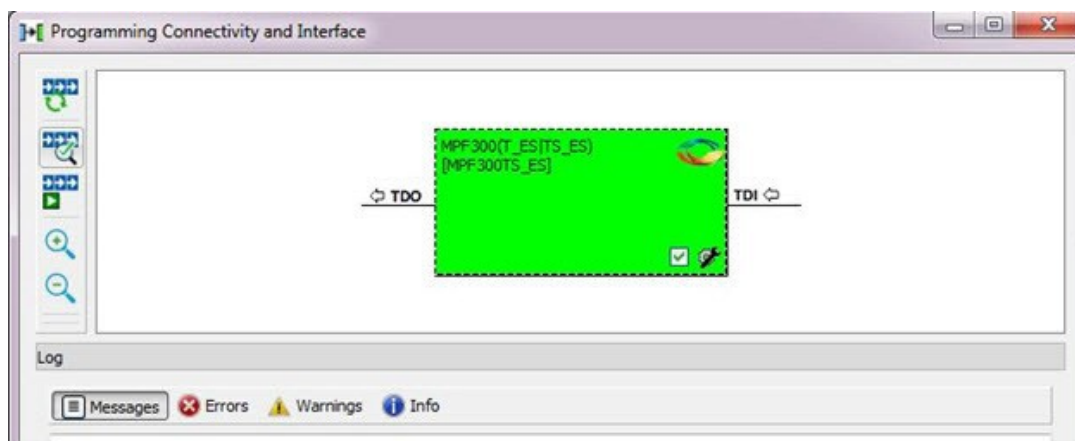
The Tools menu includes Programming Connectivity and Interface and Programmer Settings options, which are enabled after creating or opening a SmartDebug project.

For more information, see [2.2 Programming Connectivity and Interface](#).

## 2.2 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.

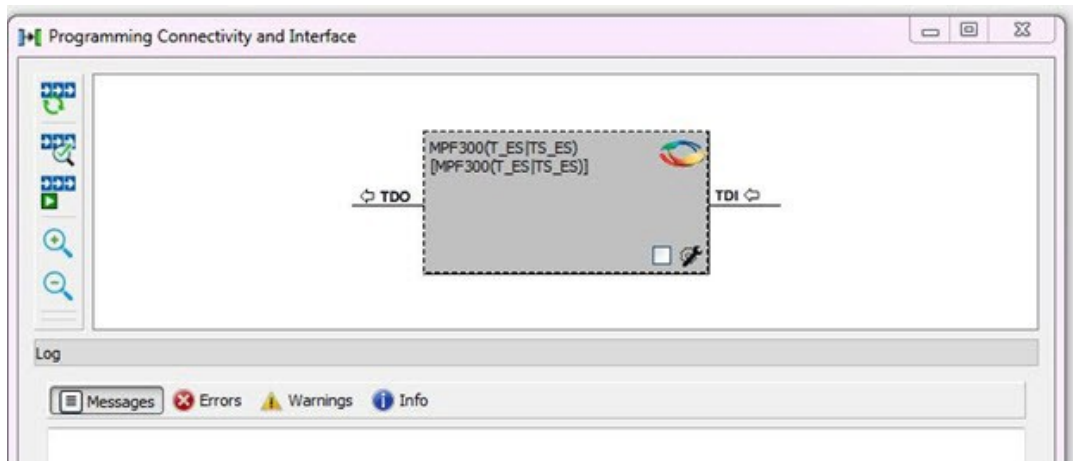
**Figure 2-2. Programming Connectivity and Interface Dialog Box – Project created using Import from DDC File**



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 2-3. Programming Connectivity and Interface window – Project created using Construct Automatically



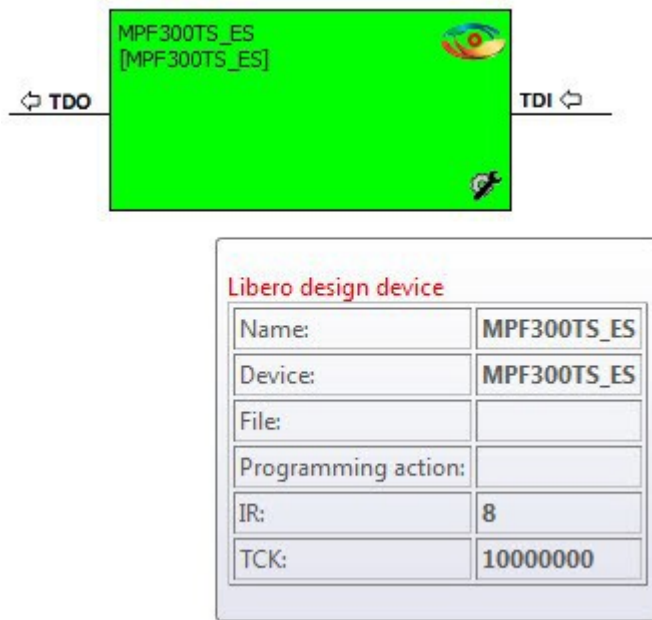
The Programming Connectivity and Interface dialog box includes the following actions:

- **Construct Chain Automatically:** Automatically construct the physical chain. Running Construct Chain Automatically in the Programming Connectivity and Interface removes all existing debug/programming data included using DDC/programming files. The project is the same as a new project created using the Construct Chain Automatically option.
- **Scan and Check Chain:** Scan the physical chain connected to the programmer and check if it matches the chain constructed in the scan chain block diagram.
- **Run Programming Action:** Option to program the device with the selected programming procedure. When two devices are connected in the chain, the programming actions are independent of the device.
- **Zoom In:** Zoom into the scan chain block diagram.
- **Zoom Out :** Zoom out of the scan chain block diagram.

### 2.2.1 Hover Information

The device tooltip displays the Libero design device information if you hover the cursor over a device in the scan chain block diagram

Figure 2-4. Libero Design Device Information



The following table describes each Libero design device information option.

**Table 2-3. Libero Design Device Information**

Option	Description
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 <b>Properties</b> ).
Device	Microchip 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.

### 2.2.2 Device Chain Details

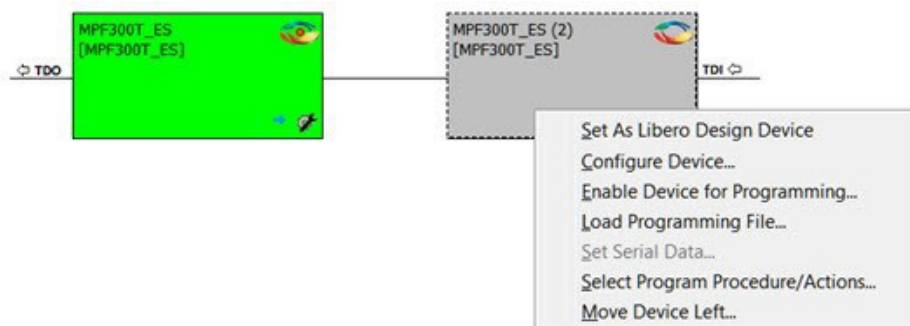
The device within the chain has the following details:

- 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.

### 2.2.3 Context Menu Options

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

**Figure 2-5. Programming Connectivity and Interface - Context Menu Options**



**Table 2-4. Programming Connectivity and Interface - Device Context Menu Options**

Option	Description
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. <ul style="list-style-type: none"> <li>• Family and Die: The device can be explicitly configured from the Family, Die drop-down.</li> <li>• Device Name: Editable field for providing user-specified name for the device.</li> </ul>

.....continued	
Option	Description
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. <ul style="list-style-type: none"> <li>• Actions: List of programming actions for your device.</li> <li>• Procedures: Advanced option; enables you to customize the list of recommended and optional procedures for the selected action.</li> </ul>
Import Debug Data from DDC File	Option to import debug data information from the DDC file. <b>Note:</b> 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).

## 2.2.4 Debug Context Save

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

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

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

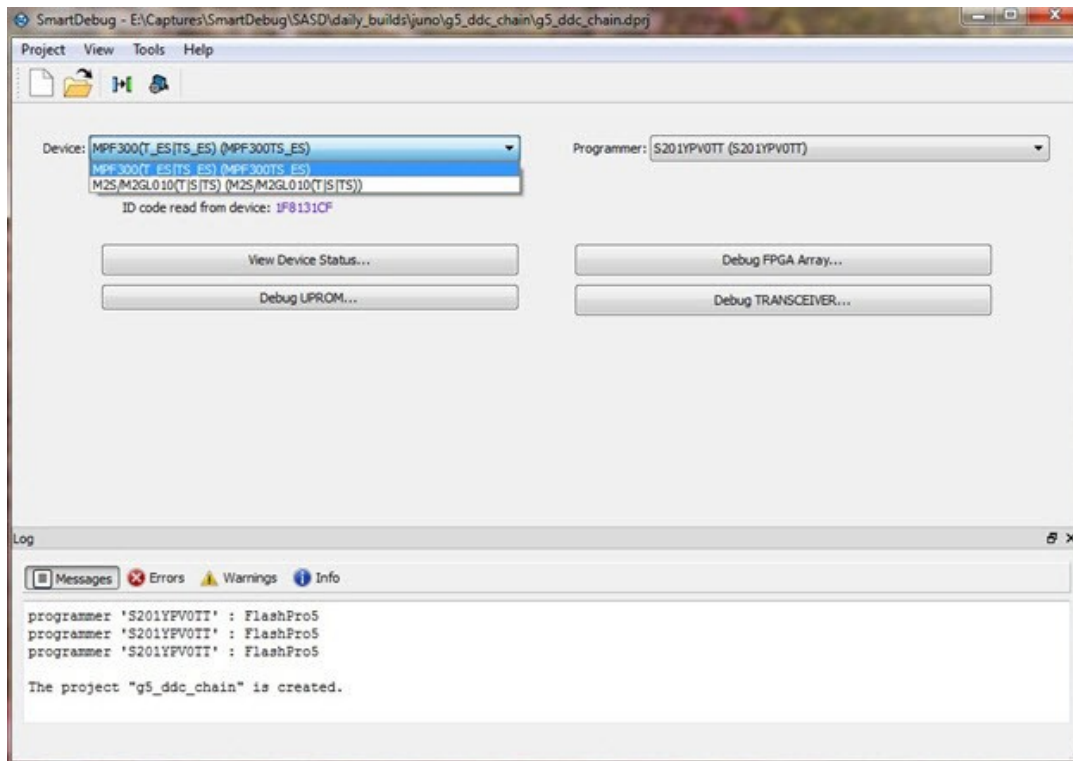
The debug context of the current session is reset for the following actions in Programming Connectivity and Interface:

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

## 2.2.5 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.

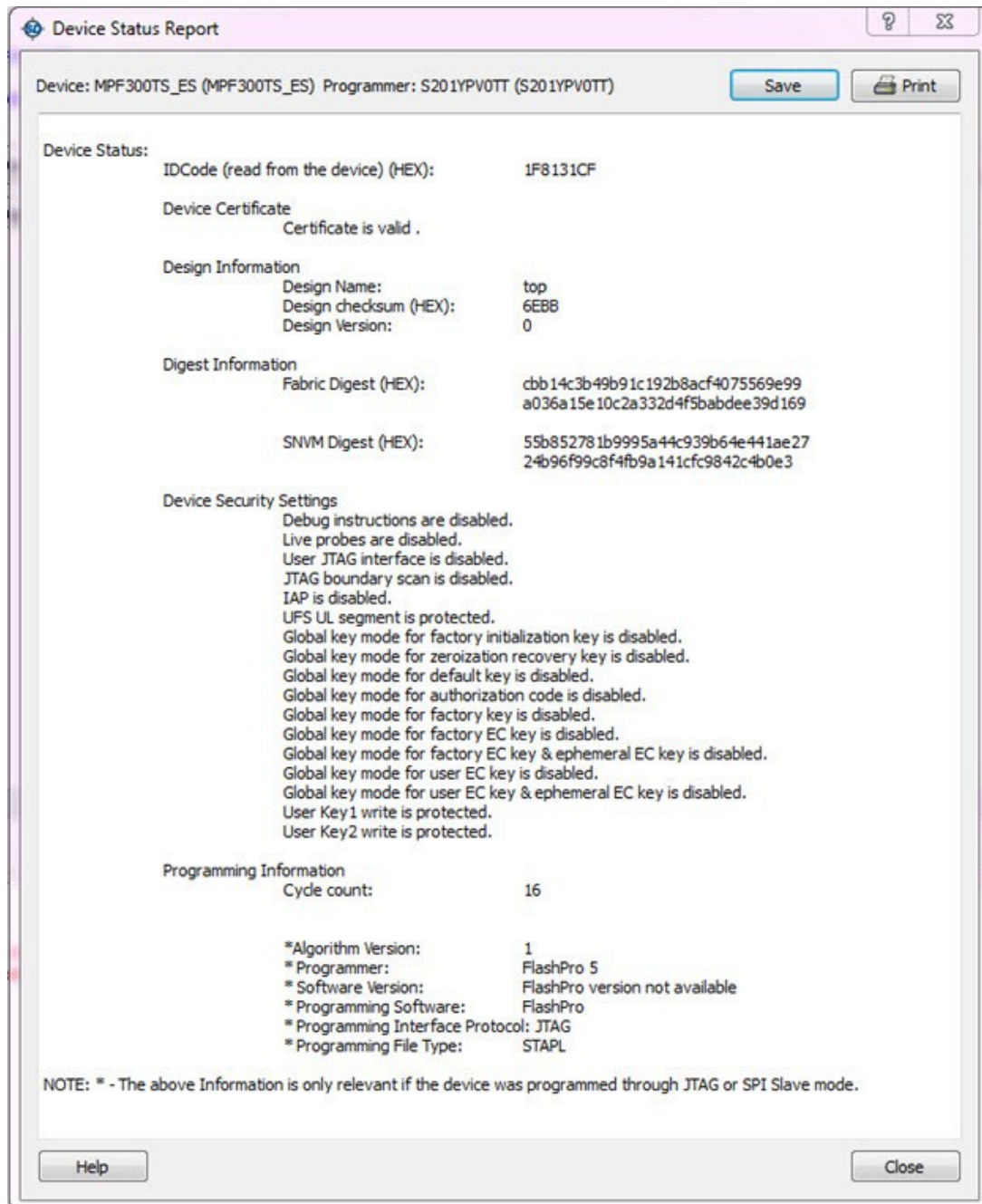
Figure 2-6. Selecting Devices for Debug



## 2.3 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 2-7. Device Status Report



The following table describes the device status report information.

Table 2-5. Device Status Report Information

Information	Description
IdCode	IDCode read from the device under debug.
Device Certificate	Device certificate displays Family and Die information if the 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.

.....continued	
Information	Description
Design Information	Design Information displays the following: <ul style="list-style-type: none"> <li>• Design Name</li> <li>• Design Checksum</li> <li>• Design Version</li> </ul>
Digest Information	Digest Information displays Fabric Digest, sNVM Digest (if applicable) computed from the device during programming. sNVM Digest is shown when sNVM is used in the design.
Device Security Settings	Device Security Settings displays information about your security settings, including live probes, JTAG boundary scan, global key modes, and user keys.
Programming Information	Programming Information displays the following: <ul style="list-style-type: none"> <li>• Cycle Count: The count is the number of times the device is programmed since it is out of factory reset. There is no limit to this count but a lower threshold would be around 2000 cycles.</li> <li>• Algorithm Version: The programming algorithm version number written to the device during programming.</li> <li>• Programmer: Details of the programmer hardware used during programming.</li> <li>• Software Version: Libero software version indicates the release version used for programming.</li> <li>• Programming Software: Software used for programming is FlashPro or DirectC or Non-Microchip software.</li> <li>• Programming Interface Protocol: Indicates the protocol followed for programming. For example, JTAG, SPI MASTER, and SPI SLAVE.</li> <li>• Programming File Type: Type of programming file used for programming the device. For example, STAPL, PPD, SVF, and IEEE532.</li> </ul>

## 3. Debugging

This topic introduces how to use the debugger to gather the device status and to view the diagnostics.

### 3.1 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) in either the [Hierarchical View](#) or the [Netlist View](#).

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](#)
- [Frequency Monitor](#)
- [User Clock Frequencies](#)

### 3.2 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.

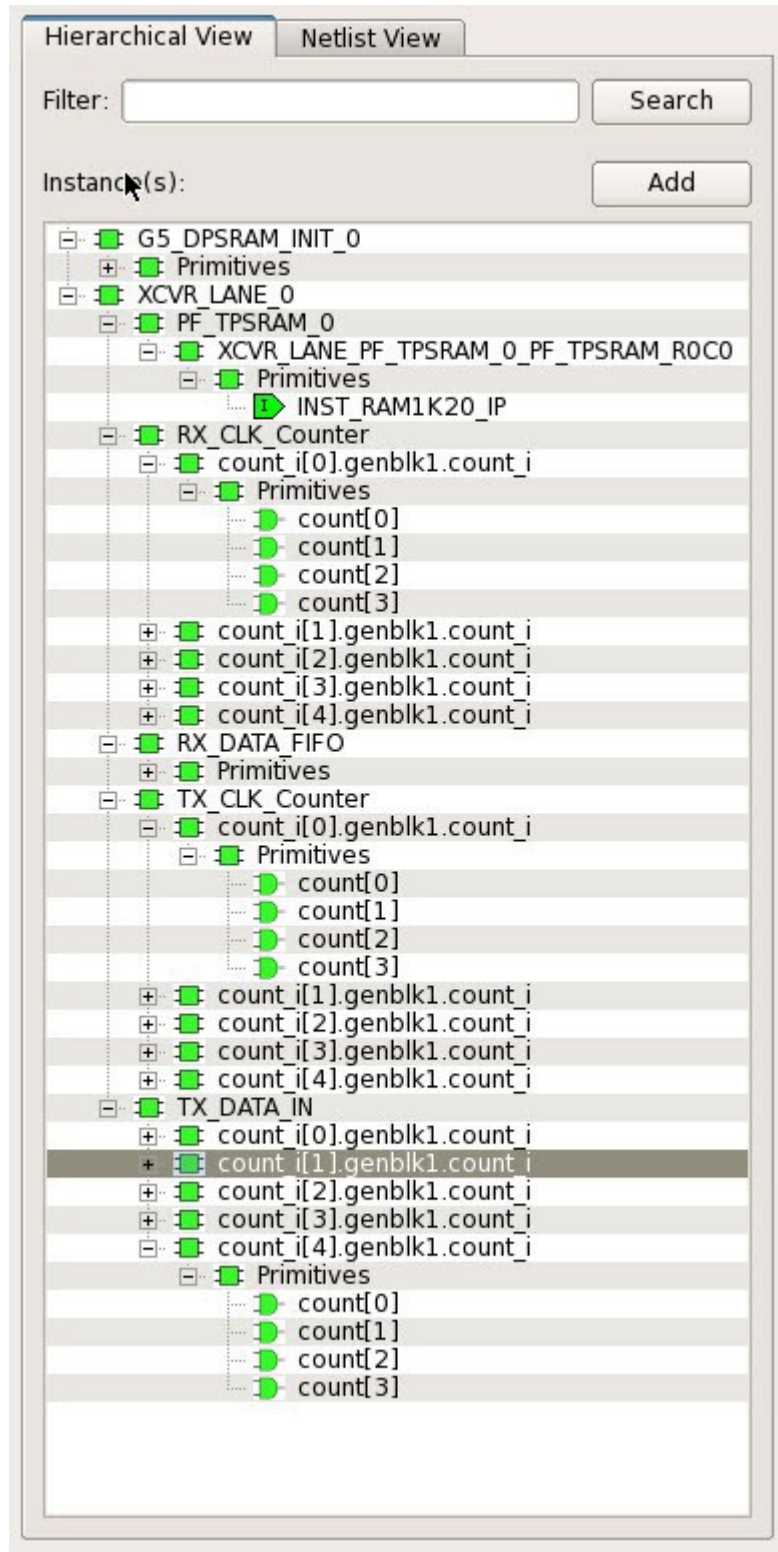
- **Filter:** In Live Probes, Active Probes, Memory Blocks, and the Probe Insertion UI, a search option is available in the Hierarchical View. You can use wildcard characters such as \* or ? in the search column for wildcard matching. Probe points of leaf level instances resulting from a search pattern 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.
- **Instance(s):** Displays the probe points available at the instance level.
- **Primitives:** Displays the lowest level of probe-able points in the hierarchy for the corresponding component such as 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 3-1. Hierarchical View



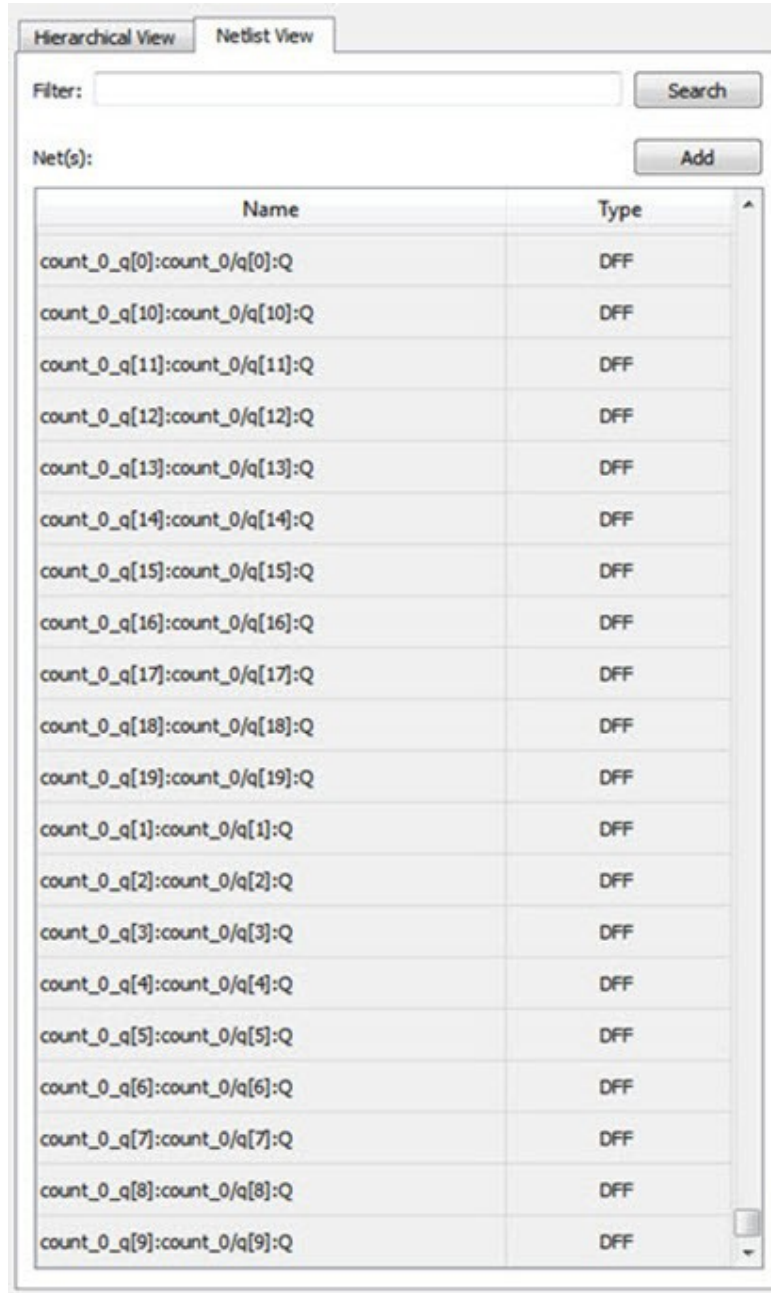


### 3.3 Netlist View

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

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.

**Figure 3-2. Netlist View**

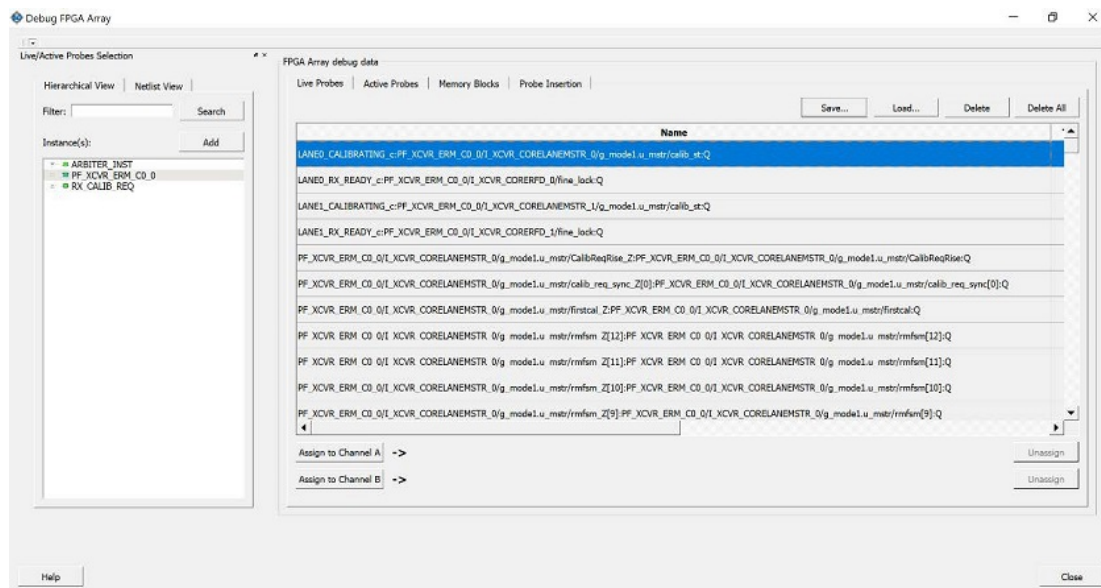


### 3.4 Live Probes

The same circuitry for programming the flash switches that has access to these points at the DFFs of every LE is re-purposed for debugging. SmartDebug controls these signal points with the JTAG interface to either allow asynchronous reads / writes to these DFFs or use an additional muxing circuit allowing two signal points to be re-routed to special pins called "Channels". These signal points are IO pads present on the FPGA boards that could be used to connect to the oscilloscope to monitor dynamic signals.

Live Probes is a design debug option that uses non-intrusive real time scoping of up to two probe points with no design changes. The Live Probes tab in the Debug FPGA Array dialog box displays a table with the probe names and pin types. There are two channels, and Live Probe can be assigned/unassigned independently.

Figure 3-3. Live Probes Tab in SmartDebug FPGA Array Dialog Box



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

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 Channel A and Channel B I/Os to monitor the desired probe points. Because there are only two channels, a maximum of two internal signals can be probed simultaneously.

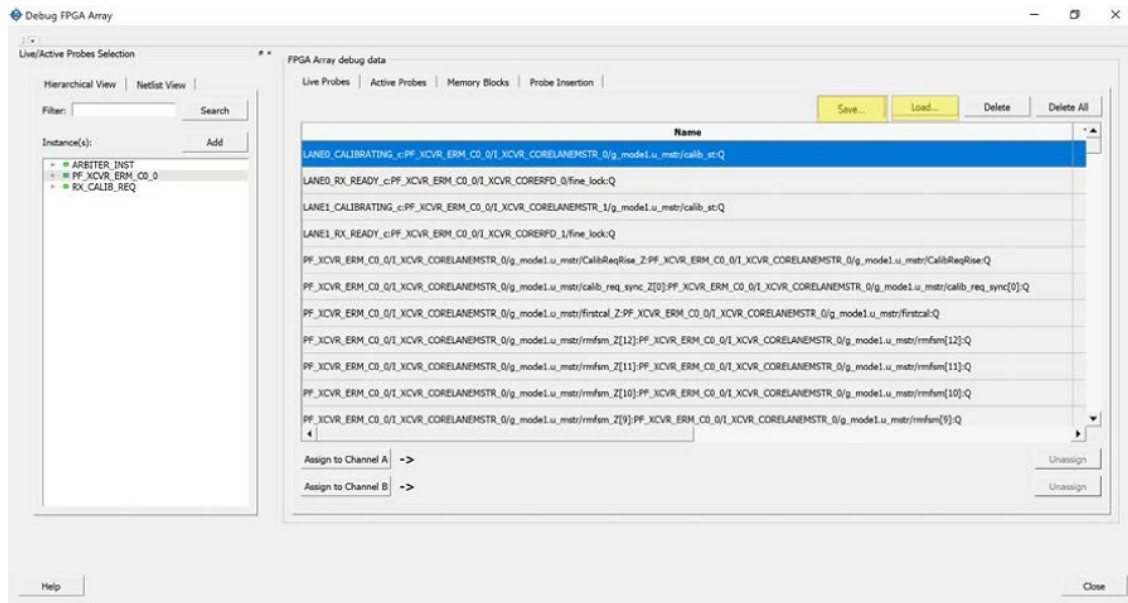
Click the **Unassign Channels** button to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

**Note:** Both probes can be assigned/unassigned independently.

The **Save** button saves the list of live probes currently shown in the SmartDebug Live Probe UI to file. The **Load** button loads the list of live probes from a file to SmartDebug Live Probe UI.

During save or load, check whether the appropriate signals saved or loaded match the signals in SmartDebug Live Probe UI and in the saved file.

Figure 3-4. Live Probes Tab - Save or Load in SmartDebug FPGA Array Dialog Box



#### Live Probes in Demo Mode

You can assign and unassign Live Probes Channel A and Channel B in the Demo Mode.

### 3.5 Active Probes

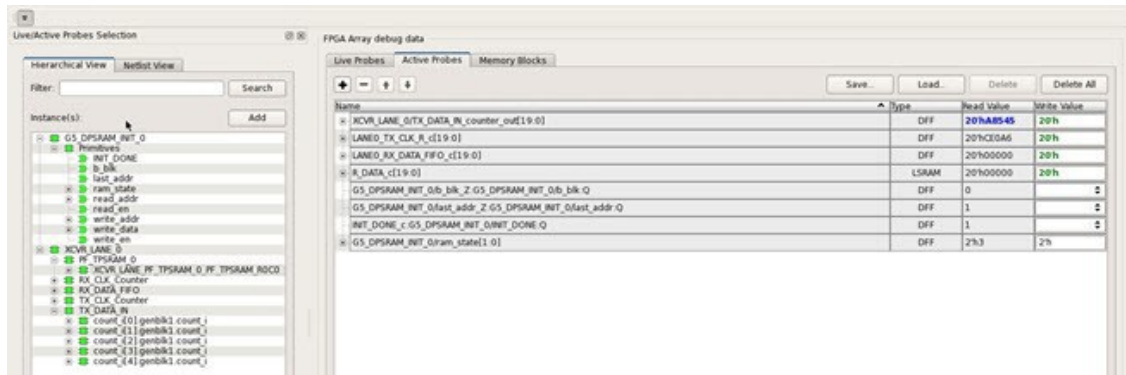
Active Probes is a design debug option to read and write to one or many probe points in the design through JTAG. Active probes makes use of the same circuitry used for programming the fabric. This feature is also non-intrusive and works asynchronously with the design clocks. In PolarFire, 18 DFFs are read at a time from the device that are physically placed adjacent to each other in the fabric. If you want to debug the design related to timing issue, then it is recommended to stop the design clocks. FHB feature can be used to stop the design clock and then probe all the DFFs in the design.

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.

When you select the desired probe, points appear in the **Active Probe** data chart and you can read and write multiple probes, see the following figure.

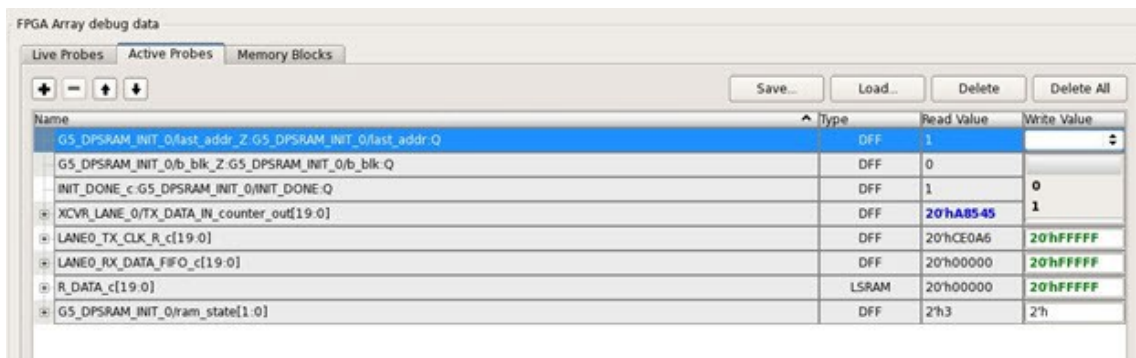
Figure 3-5. Active Probes Tab in SmartDebug FPGA Array Dialog Box



Use the following options in the **Write Value** column to modify the probe signal added to the SmartDebug FPGA Array debug data dialog box:

- 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 3-6. Active Probes Tab - Write Value Column Options



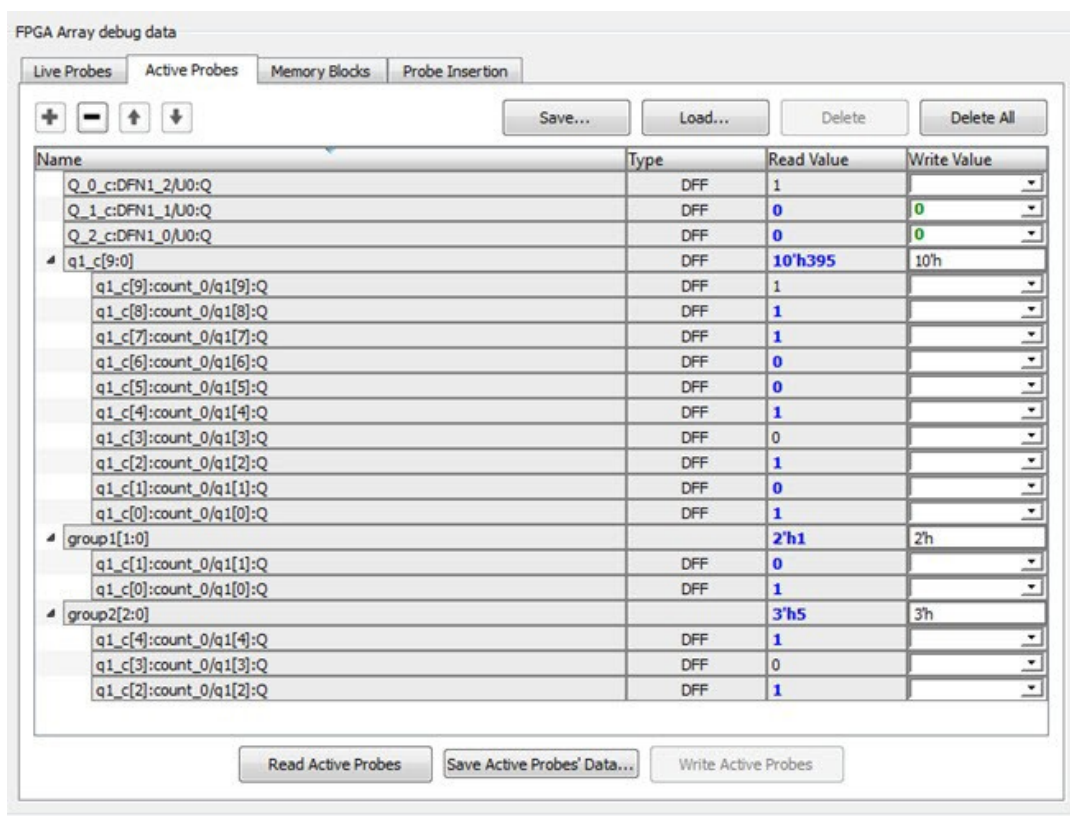
### Active Probes in Demo Mode

In the 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.

## 3.6 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.

Figure 3-7. Active Probes Tab



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

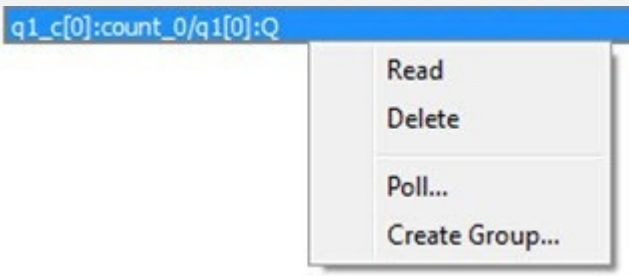
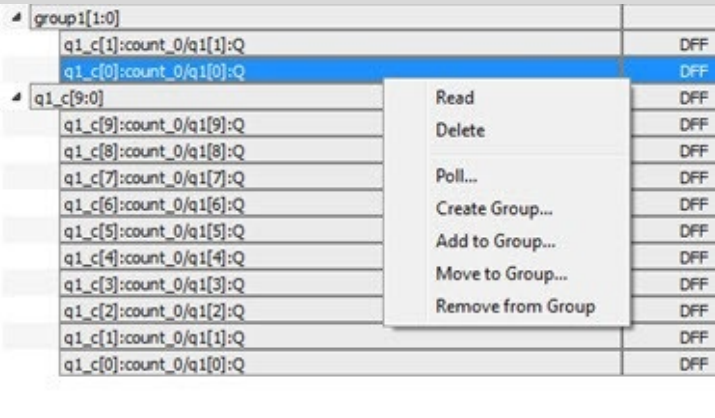
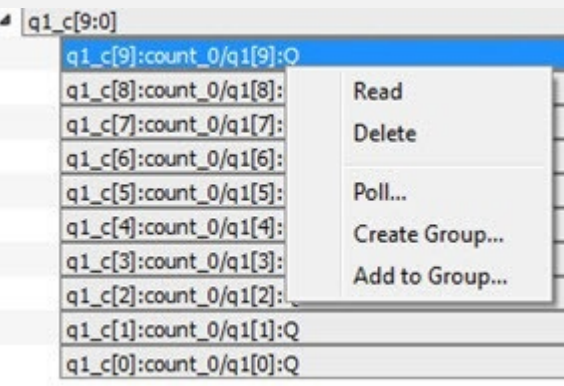
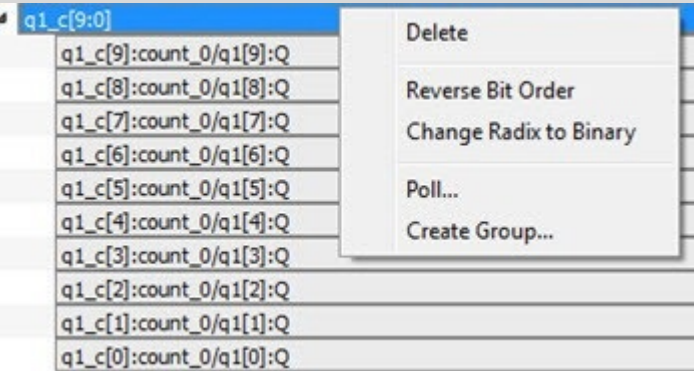
- Display bus name. An automatically generated bus name cannot be modified. Only custom bus names can be modified.
- Expand/collapse bus or probe group
- Move Up/Down the signal, bus, or probe group
- Save (Active Probes list)
- Load (already saved Active Probes list)
- Delete (applicable to a single 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
- 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.

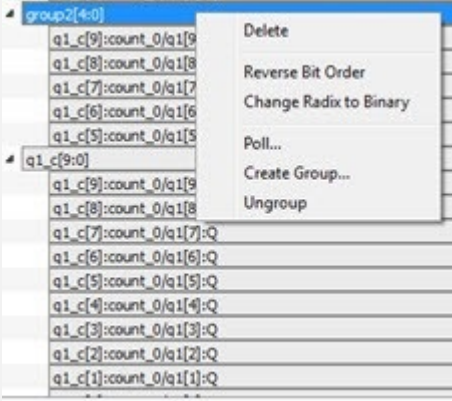
### 3.6.1 Context Menu of Probe Points Added to the Active Probes UI

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



Table 3-1. Probe Points Added to the Active Probes Tab - Context Menu

Situation	Options	User Interface
For individual signals that are not part of a probe group or bus	<ul style="list-style-type: none"> <li>Read</li> <li>Delete</li> <li>Poll</li> <li>Create Group</li> </ul>	
For individual signals in a probe group	<ul style="list-style-type: none"> <li>Read</li> <li>Delete</li> <li>Poll</li> <li>Create Group</li> <li>Add to Group</li> <li>Move to Group</li> <li>Remove from Group</li> </ul>	
For individual signals in a bus	<ul style="list-style-type: none"> <li>Read</li> <li>Delete</li> <li>Poll</li> <li>Create Group</li> <li>Add to Group</li> </ul>	
For a bus	<ul style="list-style-type: none"> <li>Delete</li> <li>Reverse Bit Order</li> <li>Change Radix to Binary</li> <li>Poll</li> <li>Create Group</li> </ul>	

.....continued		
Situation	Options	User Interface
For a probe group	<ul style="list-style-type: none"> <li>Delete</li> <li>Reverse Bit Order</li> <li>Change Radix to Binary</li> <li>Poll</li> <li>Create Group</li> <li>Ungroup</li> </ul>	

### 3.6.2 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.

**Note:** 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.

**Note:** 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

## 3.7 Memory Blocks

Memory debug accesses the RAMs present in the fabric. Large SRAM or micro SRAM can be accessed through JTAG. SmartDebug takes the access from the user interface via fabric control bus (FCB) to read and write into the locations. Once the read operation is performed, the interface is relinquished and given back to the user interface. During this operation, any data read may be outdated or unreliable and you may not be able to access the memory until the SmartDebug has finished its operation.

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.



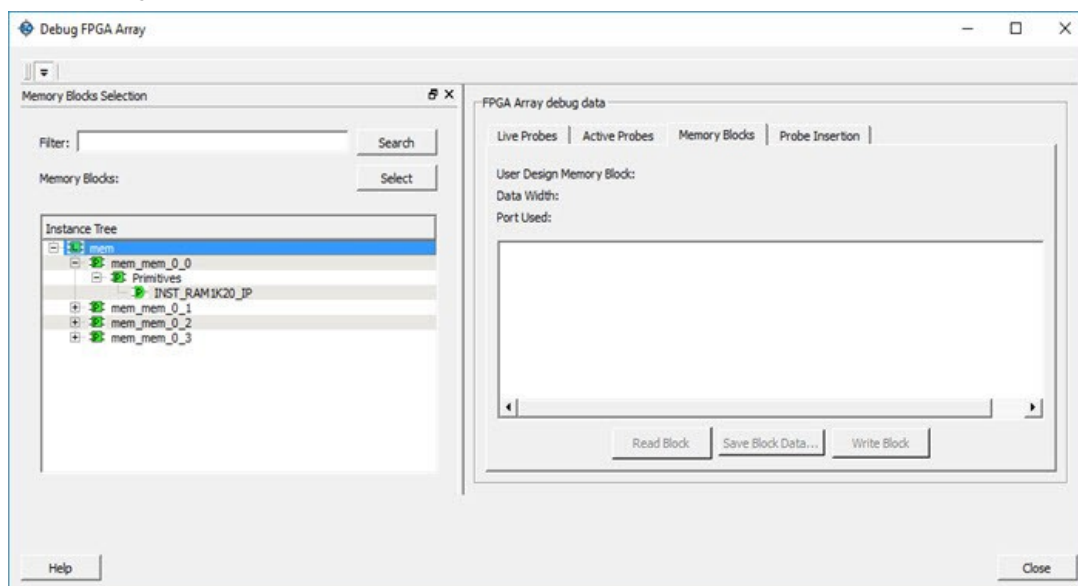
The following figure shows the hierarchical view of the Memory Blocks tab. You can view logical blocks and physical blocks. Logical blocks are shown with an L (  ), and physical blocks are shown with a P (  ).

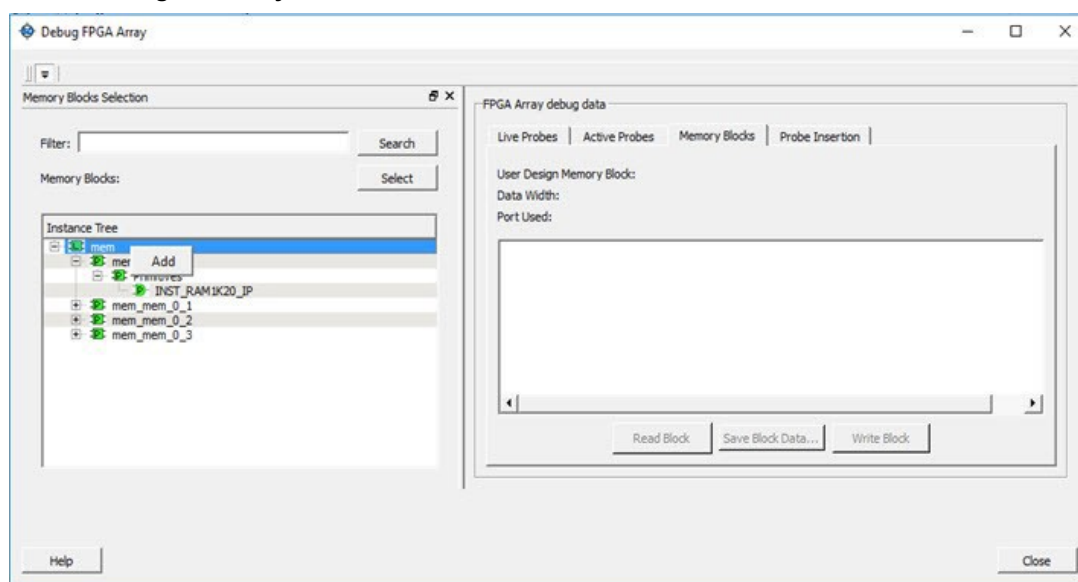
Figure 3-8. Memory Blocks Tab - Hierarchical View



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 3-9. 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.

### 3.7.1 Memory Block Fields

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

#### User Design Memory Block

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



### Data Width

If a block is logical, the depth and width is retrieved from each physical block, consolidated, and displayed. If the block is physical, the value of "Depth X Width" is 64 X 12 for uSRAM blocks, 16384x1, 8192 X 2, 4096 X 5, 2048 X 10, 1024 X 20 for LSRAM blocks, and 512 X 40(512x33 if Error Correcting Code is enabled where 512x7 is dedicated for ECC) for Two-Port RAM(TPSRAM) physical block.

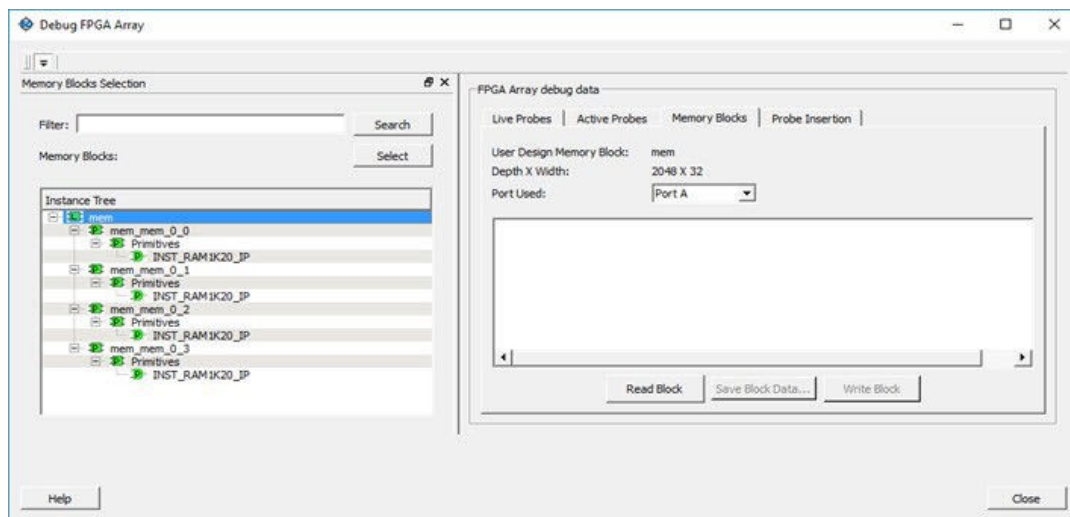
### Port Used

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 the 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.

### Memory Block Views

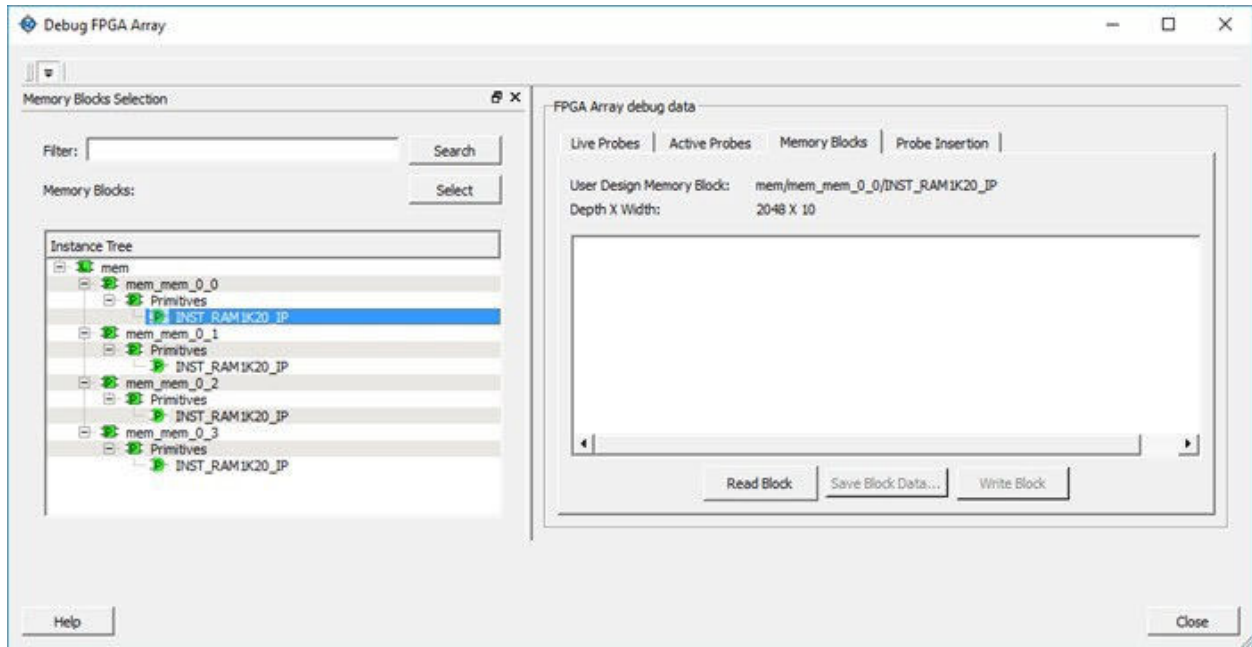
The following figure shows the Memory Blocks tab fields for a logical block view.

**Figure 3-10. Memory Blocks Tab Fields for Logical Block View**



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

Figure 3-11. Memory Blocks Tab Fields for Physical Block View



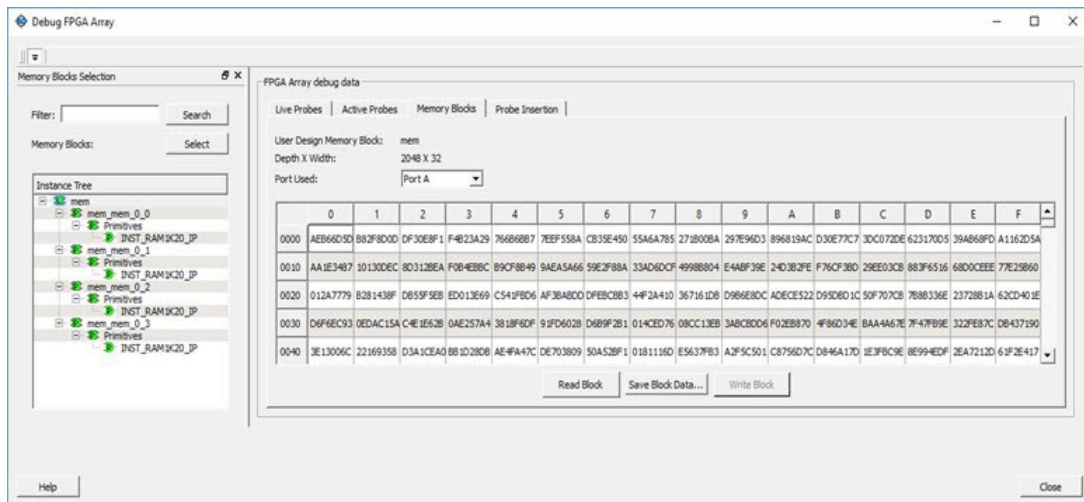
### 3.7.2 Read Block

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 3-12. Logical Block Read



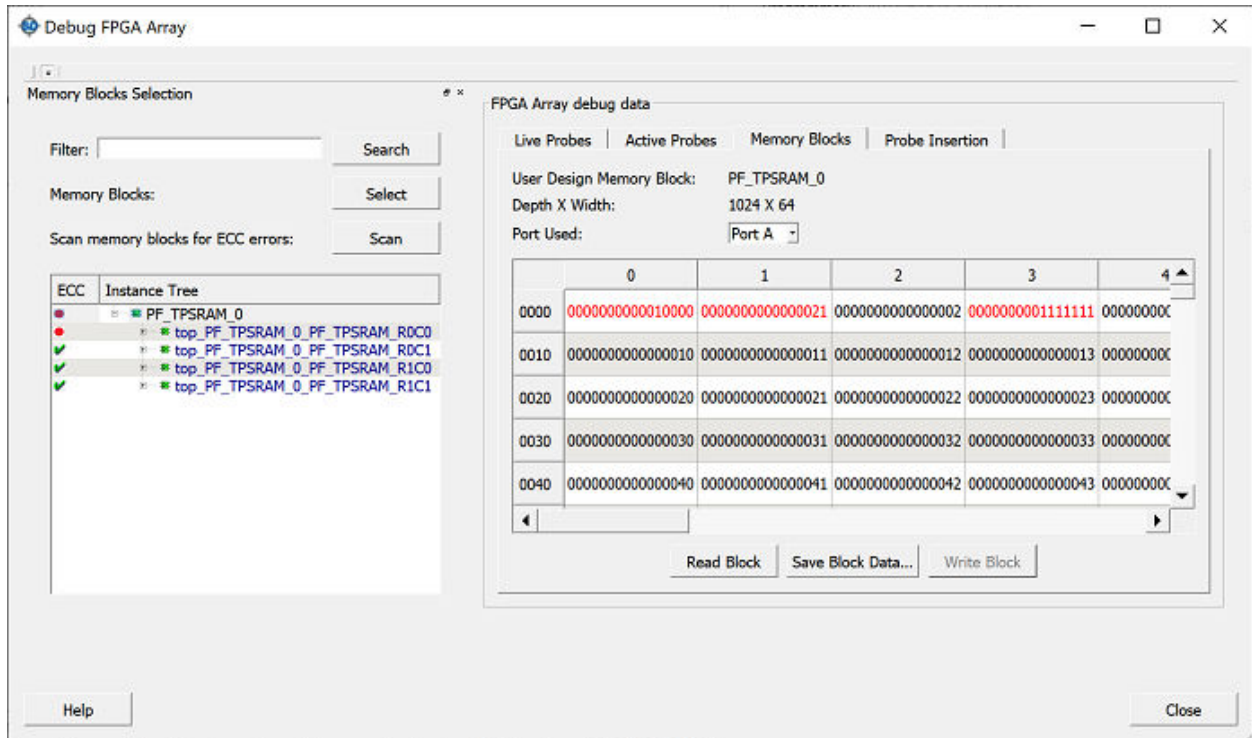
The data shown is in Hexadecimal format. In the above figure, data width is 32. Because each hexadecimal character has 4 bits of information, you can see 8 characters corresponding to 32 bits. Each row has 16 locations (shown in the column headers) which are numbered in hexadecimal from 0 to F.

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

### Logical Block Read for ECC Enabled Blocks

Two-Port RAMs support Error Correcting Code(ECC) that provides the capability of Single Error Correction and Double Error Detection (SEDED). Logical block view for ECC enabled blocks highlights the data in case any corruption is detected. All erroneous data is highlighted in red. Hover your cursor over the error to display a tooltip that shows the ECC error with the offset details. Once the logical block is read, all the physical block data is recorded for navigating to the respective physical block so that you can view the respective physical block without having to read from the device again.

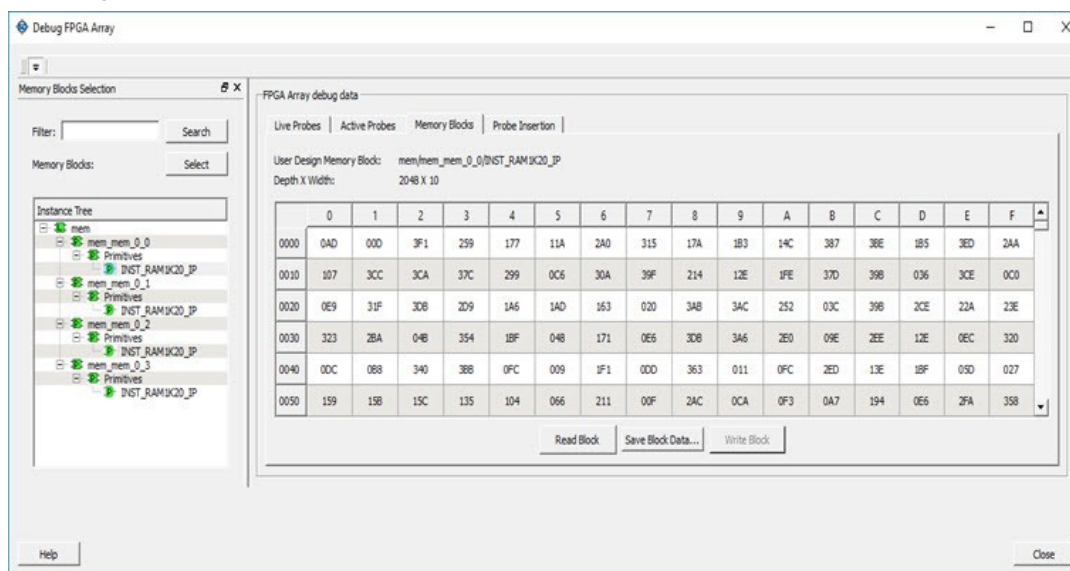
**Figure 3-13. Logical Block Read - ECC Enabled**



### Physical Block Read

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

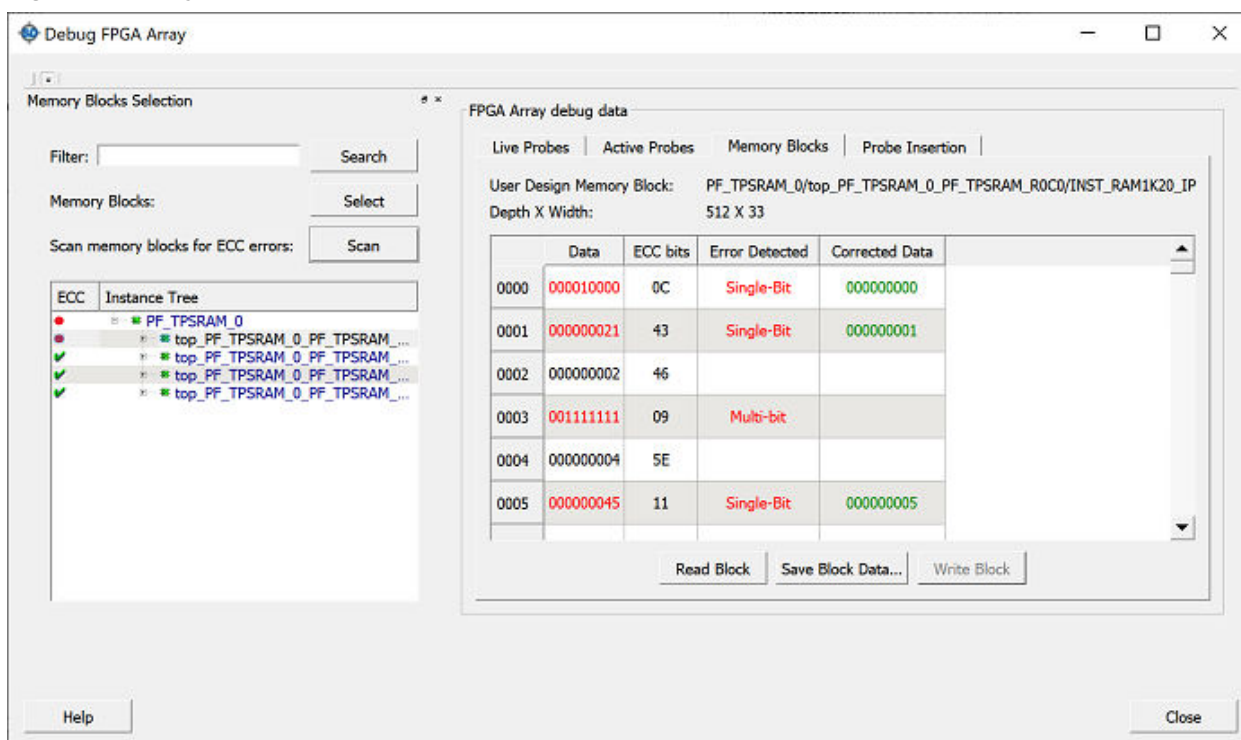
**Figure 3-14. Physical Block Read**



### Physical Block Read for ECC Enabled Blocks

Instead of data being shown in a matrix form, for ECC enabled blocks, the data internally is spread across Port A and Port B and hence they are concatenated to show the 33-bit data offset value in a single column. Similarly, ECC bits are concatenated from the two Ports and shown in the adjacent column.

**Figure 3-15. Physical Block Read - ECC Enabled**



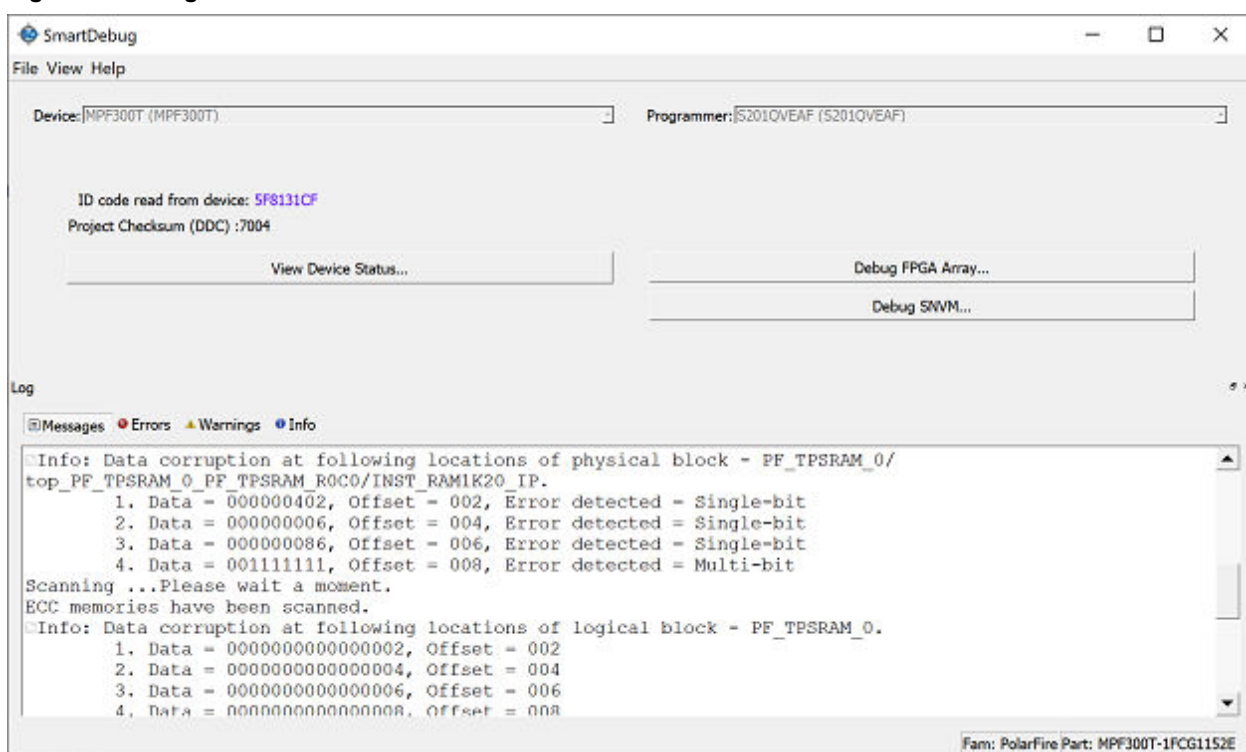
The following table describes the columns displayed in the physical block view.

Table 3-2. Physical Block View for ECC Enabled Blocks

Column	Description
Data	33-bit physical block data offset value.
ECC Bits	7-bit ECC value.
Error Detected	Displays identified error type. The error type value displayed can be either <b>Single-Bit</b> or <b>Multi-Bit</b> .
Corrected Data	If the error is a single-bit error, then the tool suggests the corrected data. The suggested data can be copied to the data cell in order to write in to the device. Right click on the corrected data to view an option to copy the data automatically to the <b>Data</b> cell.

Once the block is read, a log is generated and all the erroneous locations are listed. This log is also seen during a physical block read or when navigated from logical block view to physical. The same is true for navigation from physical to logical block view.

Figure 3-16. Log Info



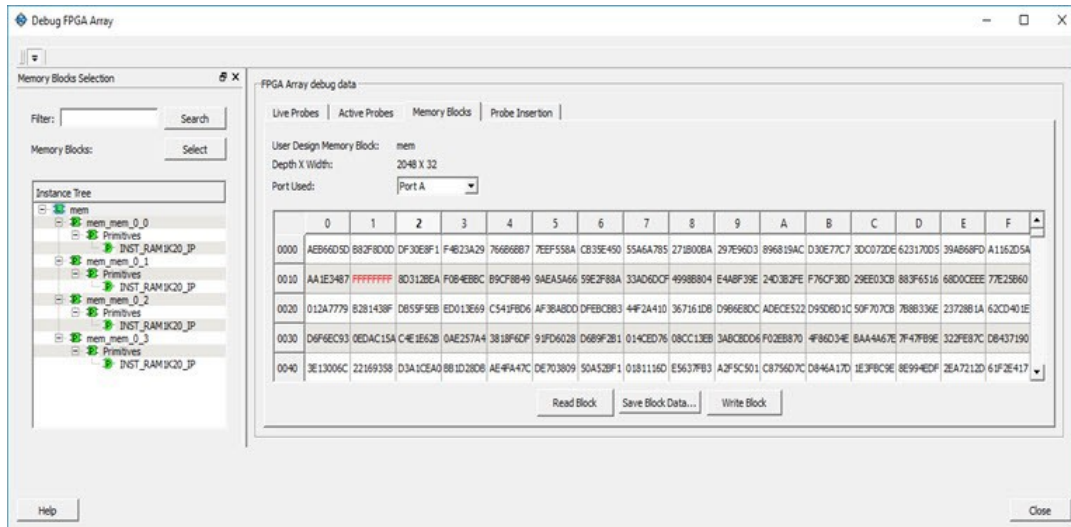
### 3.7.3 Write Block

#### 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 message showing the range of allowed values.



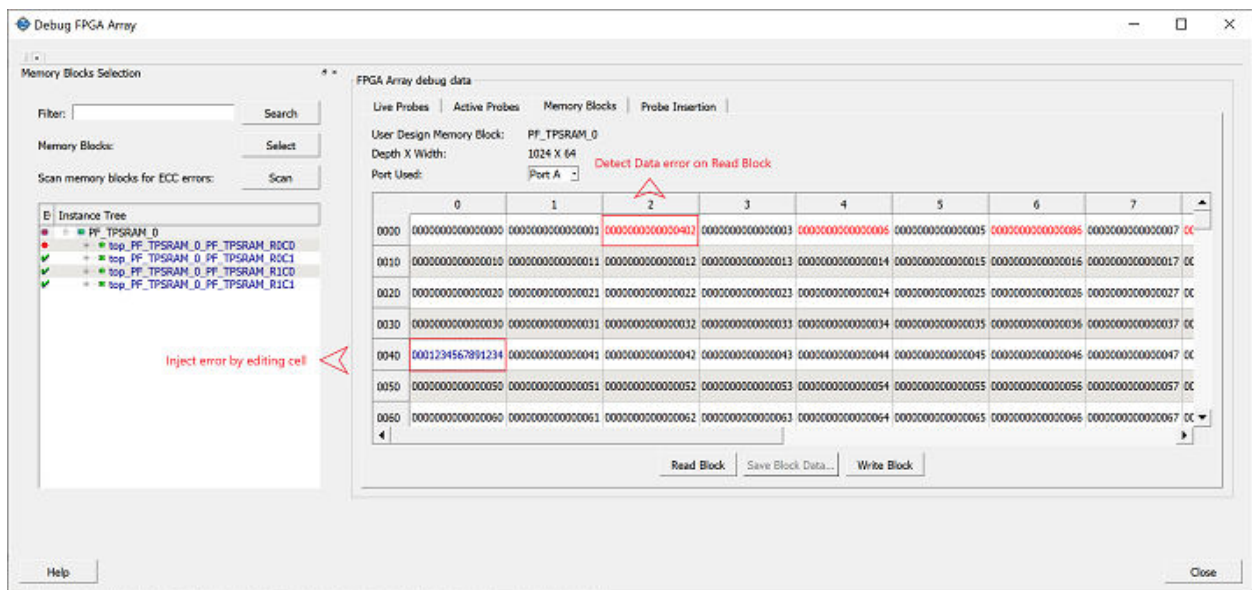
**Figure 3-17. Logical Block Write**



### Logical Block Write for ECC Enabled Blocks

You can inject error into the logical block by writing onto the location. To verify if data is corrupted, read on the logical block will highlight the corresponding location.

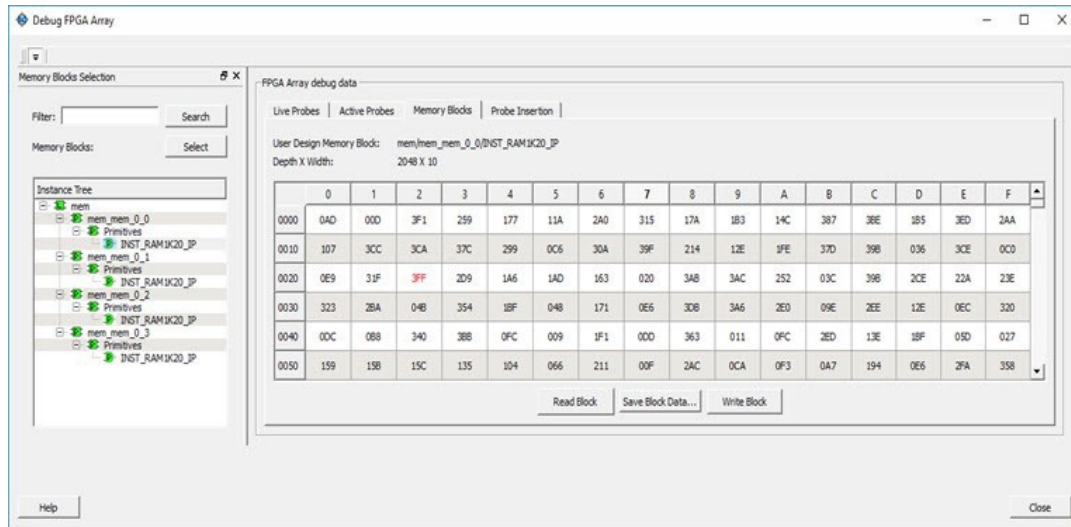
**Figure 3-18. Inject Error - Logical Block Write for ECC Enabled Blocks**



### Physical Block Write

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

**Figure 3-19. Physical Block Write**

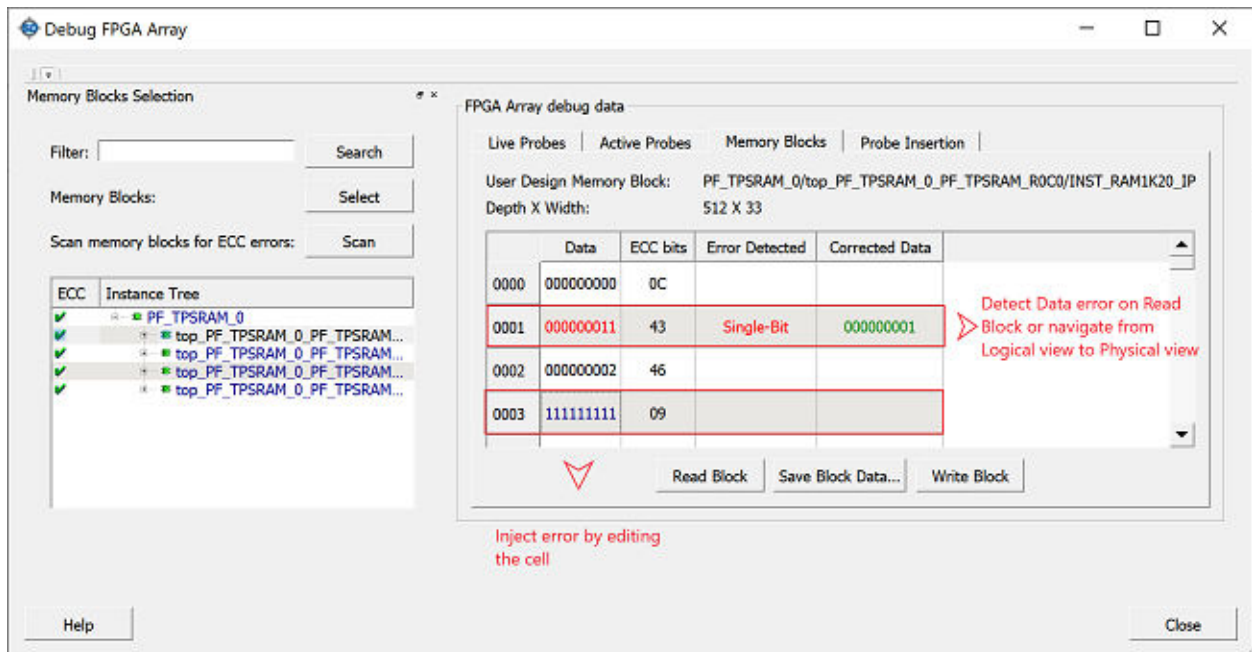


### Physical Block Write for ECC Enabled Blocks

You can inject error into the physical block by writing onto the location. To verify if data is corrupted, read on the physical block will display the erroneous data, location and suggest corrected data if it is a single bit.

**Note:** Once the data is corrupted(written) onto physical block location, the logical block is outdated. On navigating to the logical view, the locations of corrupted data will be updated only after **Read Block** is clicked.

**Figure 3-20. Inject Error - Physical Block Write for ECC Enabled Blocks**



### Limitation of Injecting Errors

The error injection is discretionary. Generally ECC algorithm has limitations as far as a Multi-bit detection is concerned. If the data corrupted is beyond 3 or 4-bits, then this might result in any one of the following scenarios:

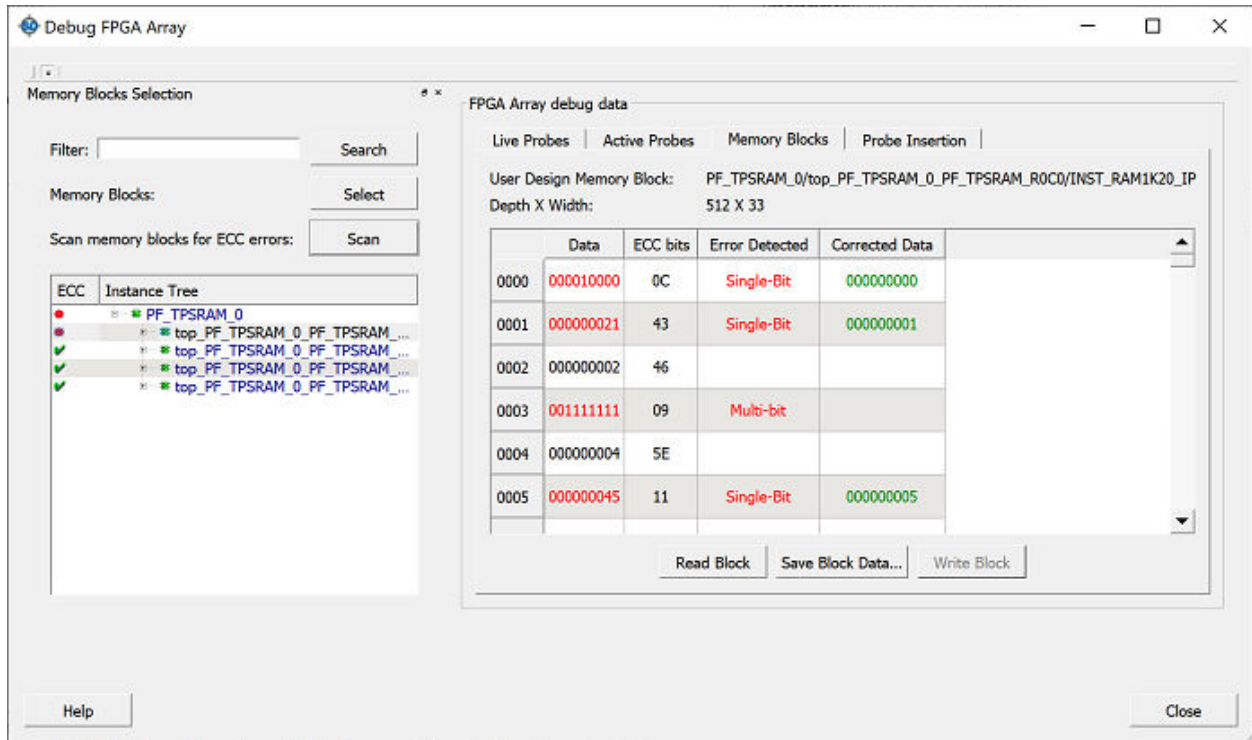
- An incorrect Single-bit corrected data is suggested
- Error might not be detected because the ECC calculated for the corrupted data might be the same ECC calculated originally.

### 3.7.4 Scan Memory

Memory Hierarchy in the **Debug FPGA Array** window is enhanced to display whether the ECC enabled memory blocks are corrupted or not by providing a scan option.

Click **Scan** to scan the ECC enabled memory blocks for errors. If there are no errors in a memory block, a green tick appears on to the left of memory block name. Otherwise, a red circle indicating the data is corrupted appears.

**Figure 3-21. Scan Memory Blocks for ECC Errors**



### 3.7.5 Unsupported Memory Blocks

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

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.

### 3.7.6 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.

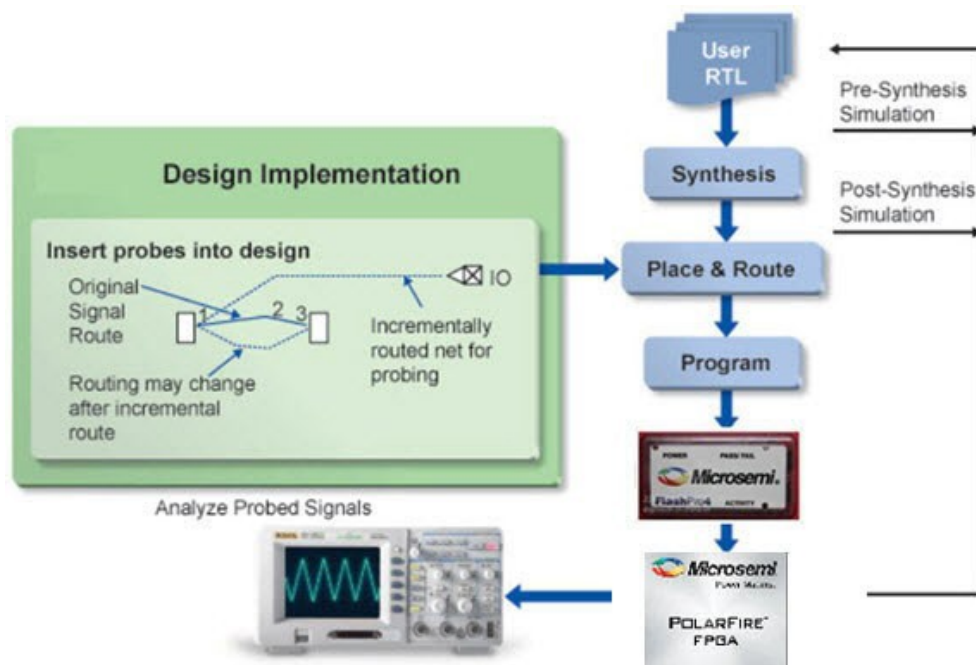
## 3.8 Probe Insertion (Post-Layout)

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 re-routed 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 3-22. Probe Insertion in the Design Process



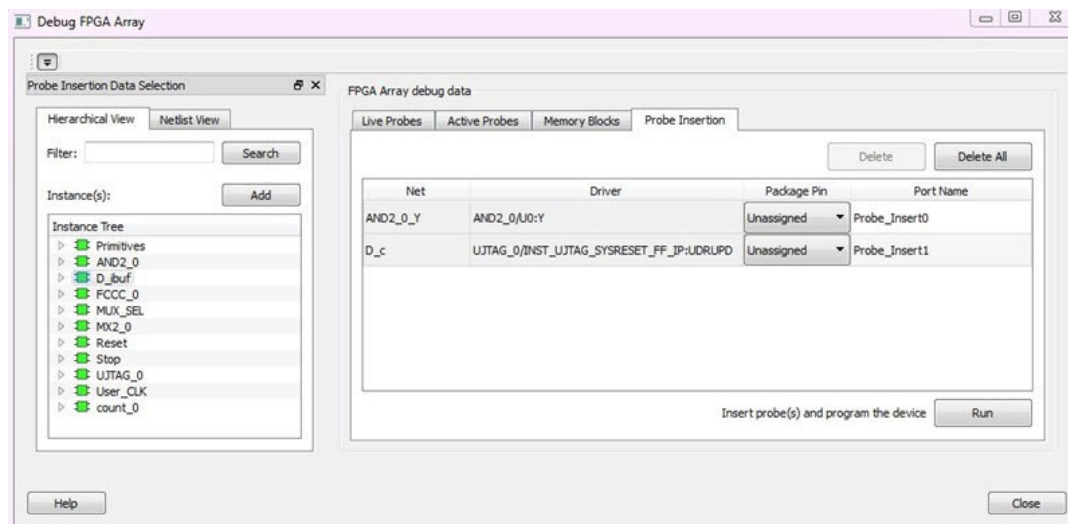
The Probe Insertion debug feature is complementary to Live Probes and Active Probes. Live Probes and Active Probes use a special dedicated probe circuitry.

### 3.8.1 Inserting Probe and Programming the Device

To insert probe(s) and program the device:

1. Double-click **SmartDebug Design** in the Design Flow window to open the SmartDebug window.  
**Note:** FlashPro Programmer must be connected for SmartDebug.
2. Select **Debug FPGA Array** and then select the Probe Insertion tab.

Figure 3-23. 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.

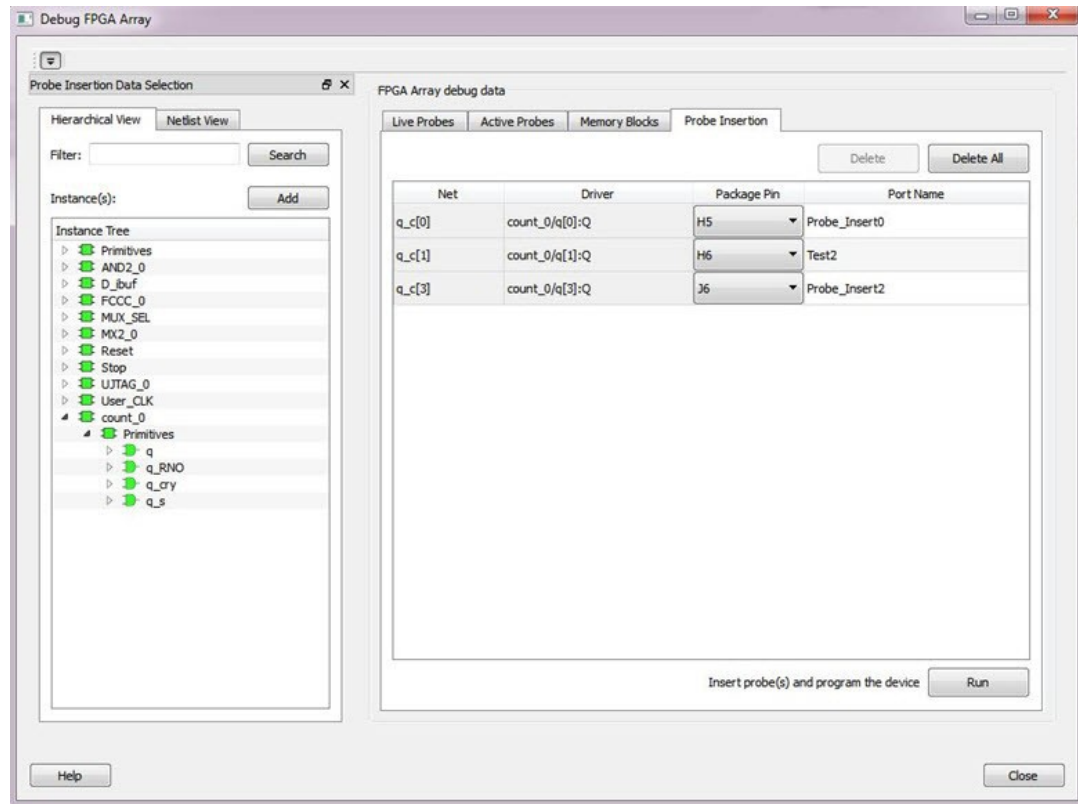
3. Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the **Filter** box.

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

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

- 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 3-24. Debug FPGA Array > Probe Insertion > Add Probe**



- 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.

### 3.8.2 Deleting a Probe

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.

### 3.8.3 Reverting to the Original Design

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

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

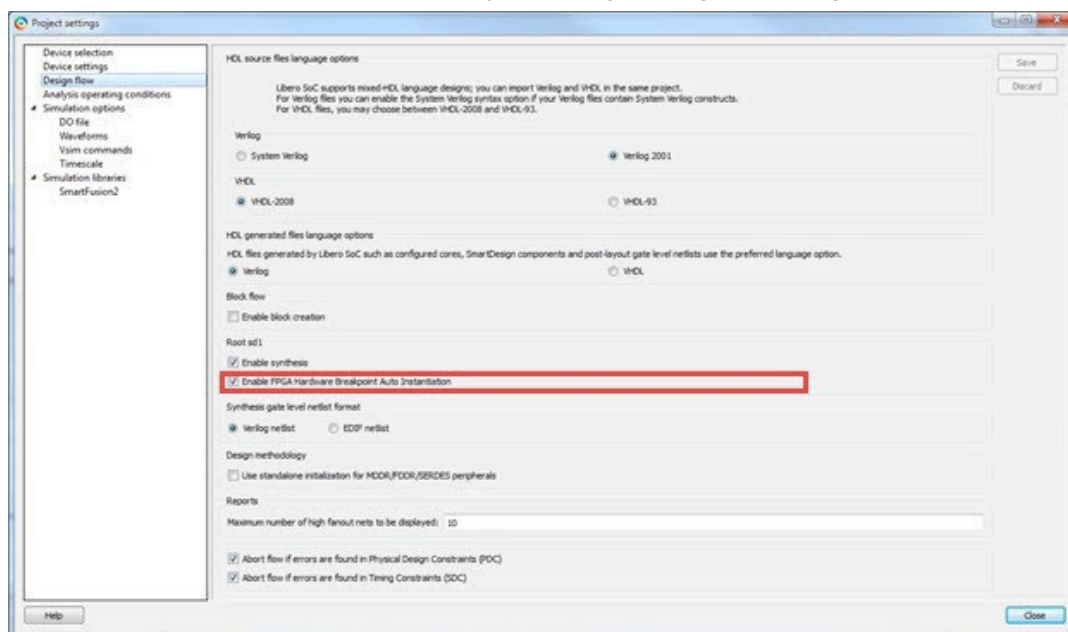
### 3.9 FPGA Hardware Breakpoint Auto Instantiation

The 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 FHB 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*).

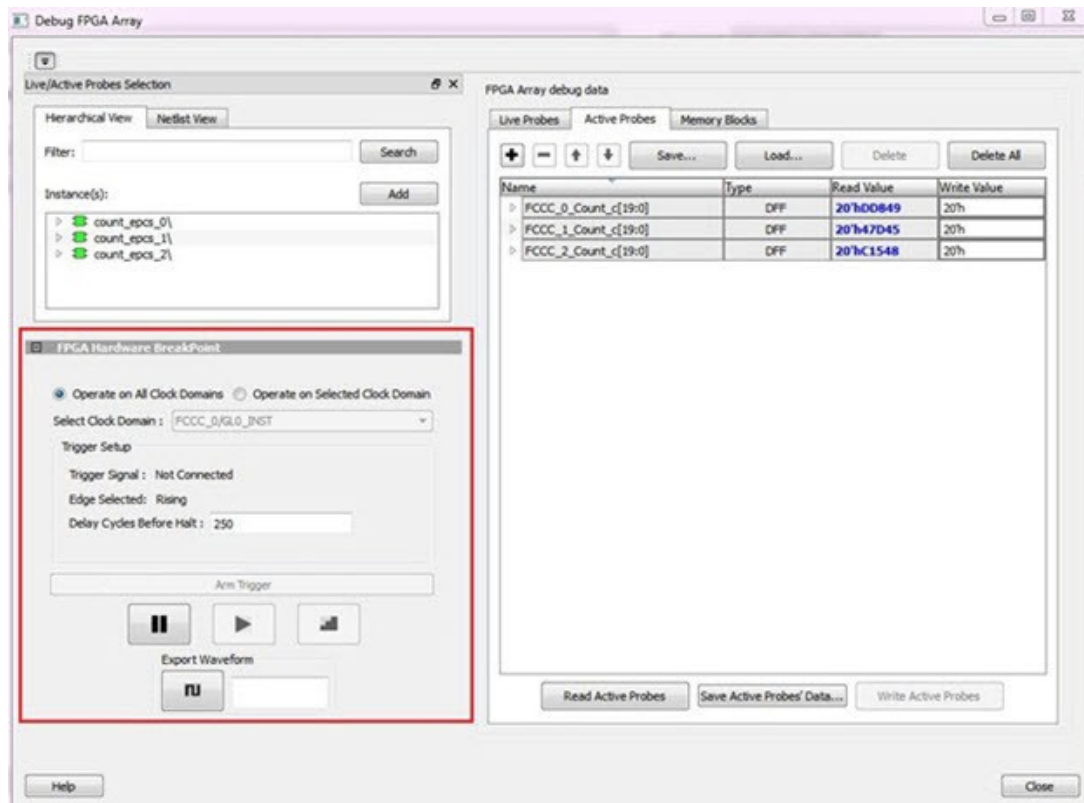
**Note:** FHB auto-instantiation can also be done in the “Import netlist as VM file” flow. See the following figure.

**Figure 3-25. Enable FHB Auto Instantiation in Project Settings Dialog Box: Design Flow Tab**






FHB controls appear in the Debug FPGA Array dialog box when there is an auto- instantiated FHB instance in the design as shown in the following figure.

Figure 3-26. FPGA Hardware Breakpoint (FHB) Controls



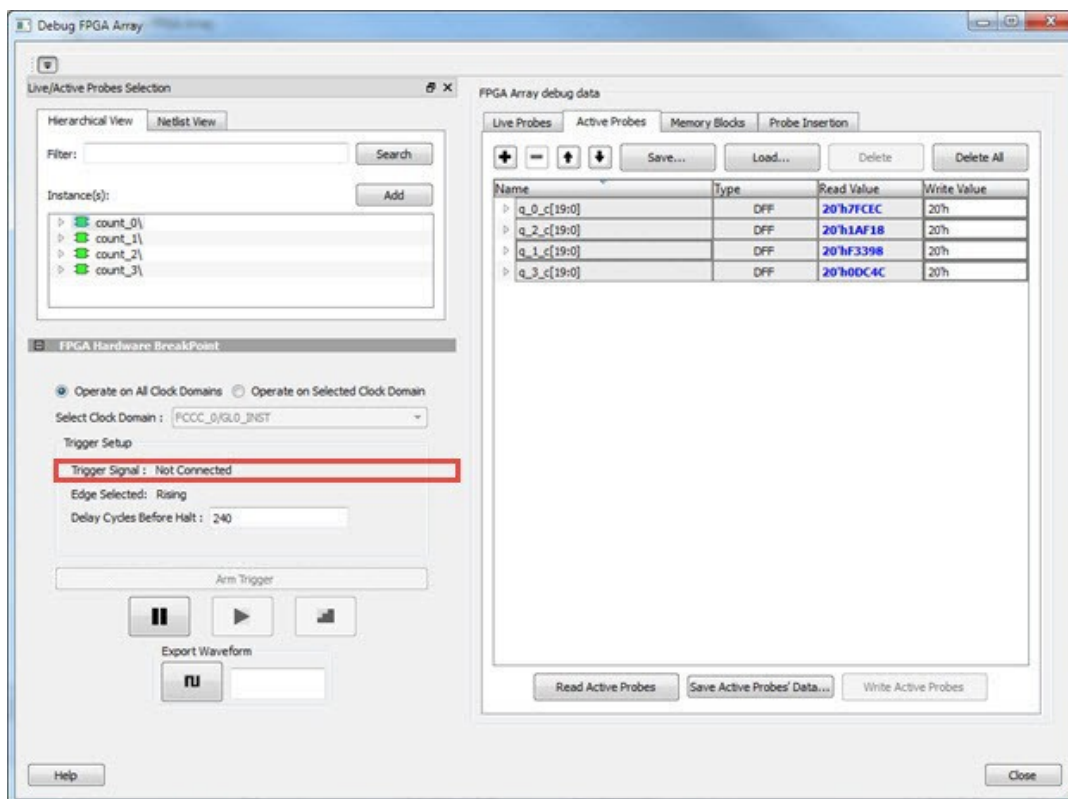
Choose **Operate on All Clock Domains** or **Operate on Selected Clock Domain** by selecting the appropriate radio button. Selecting either of these modes sets the FHB instances to the respective mode. Once you assign the Live Probe PROBE\_A connection and click **Arm Trigger**, the DUT halts on the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

When you choose **Operate on Selected Clock Domain** mode, the Select Clock Domain combo box is enabled,

and all available clock domains are listed. The Halt (Pause) , Play , and Step  buttons are associated for that clock domain. If you switch between clock domains in this mode, previous clock domain settings are not retained.

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

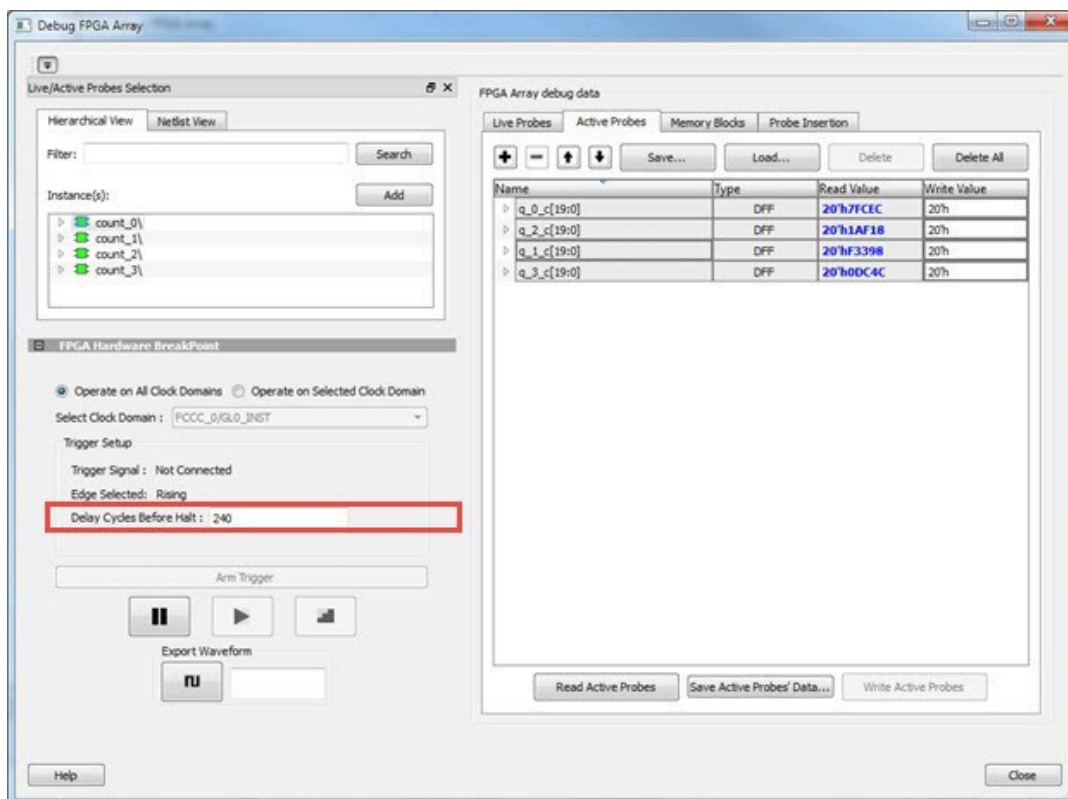
The Trigger Signal is shown as Not Connected until a live probe is assigned as shown in the following figure.



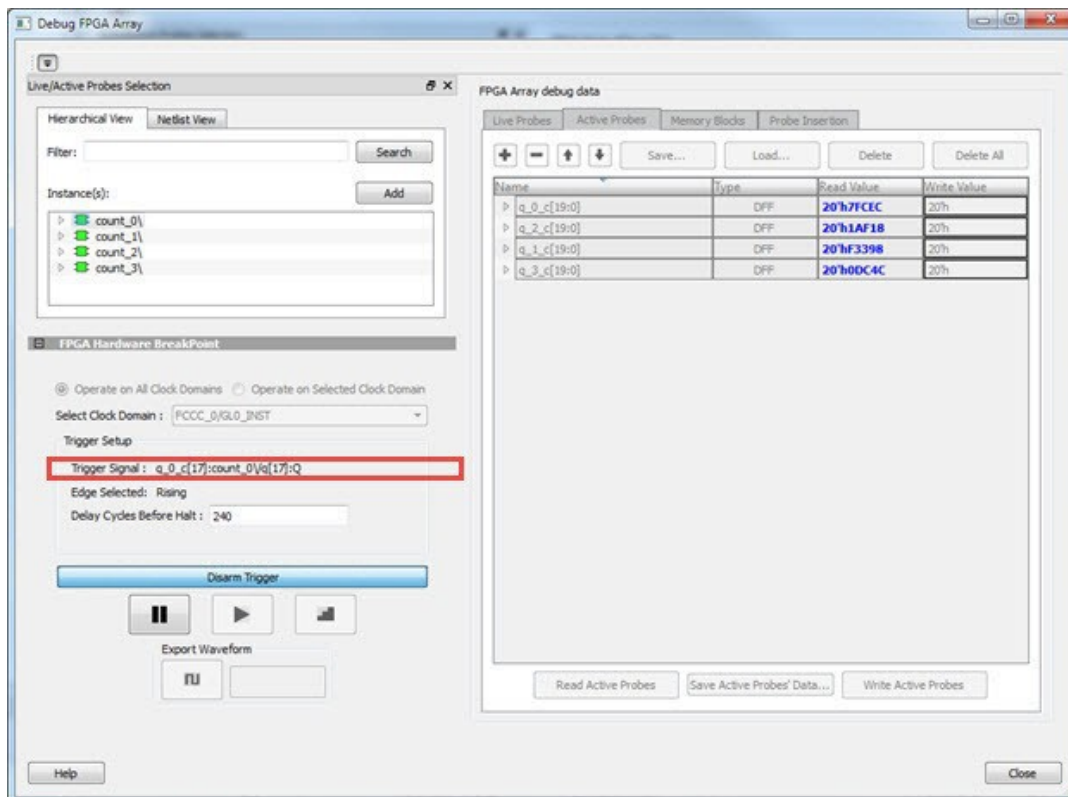
When a probe is assigned to Live Probe PROBE\_A, the Trigger Signal updates.

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 as shown in the following figure.



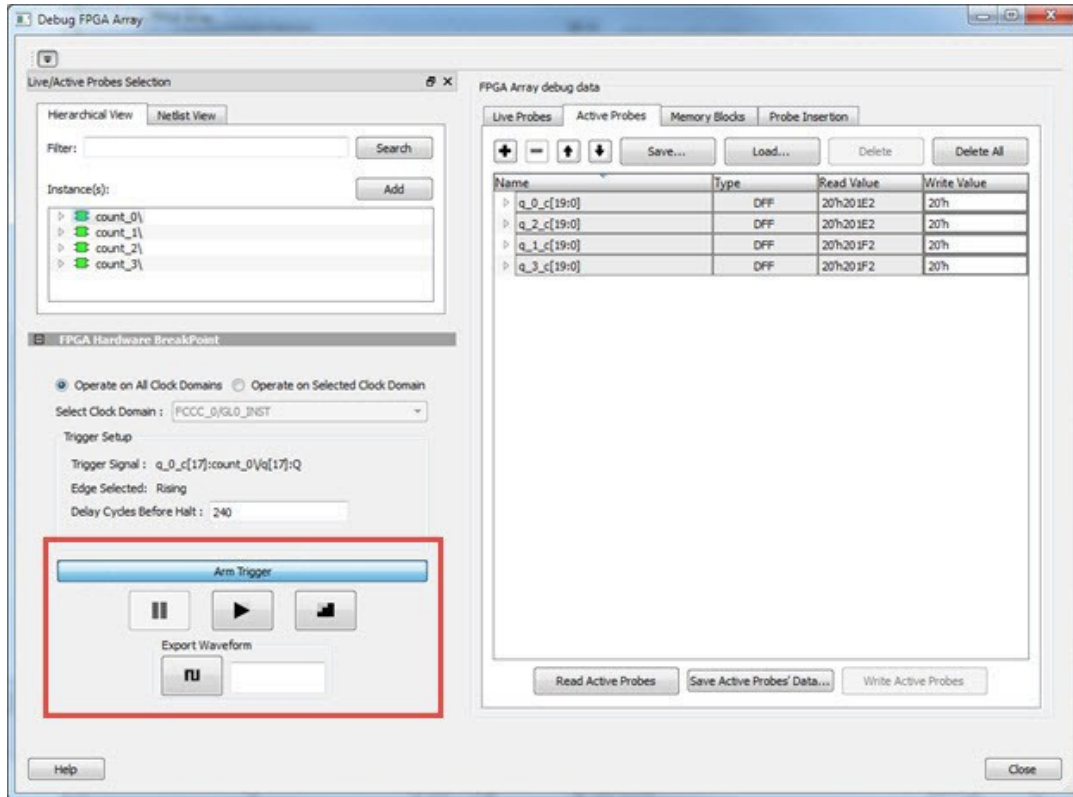
When a live probe connection is made and you click **Arm Trigger**, FHB functionality is disabled until the trigger is disarmed automatically or the design is force halted as shown in the following figure.



### 3.9.1 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 FHB controls as shown in the following figure.



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.

### 3.9.2 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)
```

Where FHB clock frequency is device specific. For PolarFire, the frequency is 160 MHz.



---

For more information, see [Assumptions and Limitations](#).

### 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 the **Operate on Selected Clock Domain** mode, the state of the **Halt** button is updated based on the state of the clock domain selected.
- In the **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 the `.vcd` file format.

## 3.9.3 Assumptions and Limitations

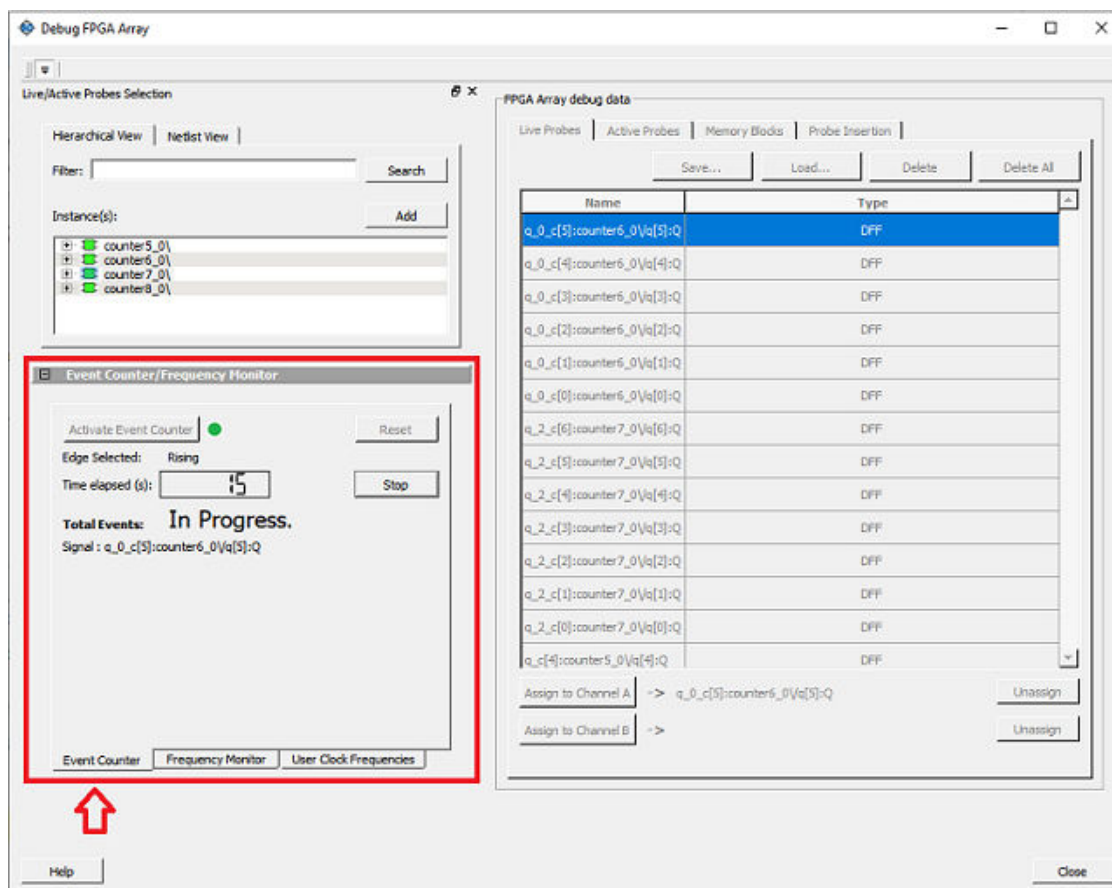
- If you select the auto instantiation option in Libero, you need to rerun Synthesis (if already run) to get the FHB related functionality.
- Supported for FCC driven gated clocks (GL0/GL1/GL2/GL3) only.
- CLKINT\_PRESERVE – FHB is not auto-instantiated if the user design contains this macro.
- Designs 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. 160 MHz).
  - If the DUT clock is running at or less than 160 MHz, the DUT clock will halt within one clock cycle (1 or less).
  - For frequencies higher than 160 MHz, the point at which the DUT halts cannot be guaranteed.



## 3.10 Event Counter

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

Figure 3-27. Event Counter Tab/UI

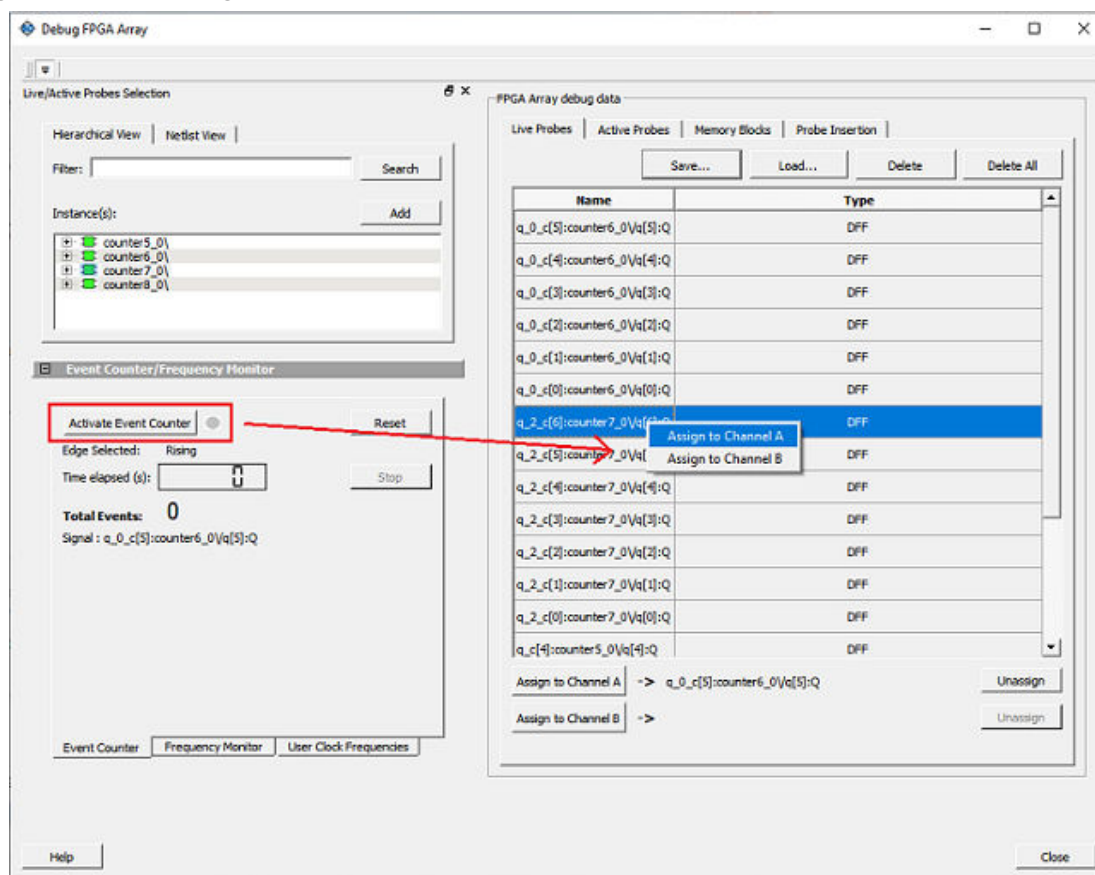


### 3.10.1 Activating the Event Counter

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

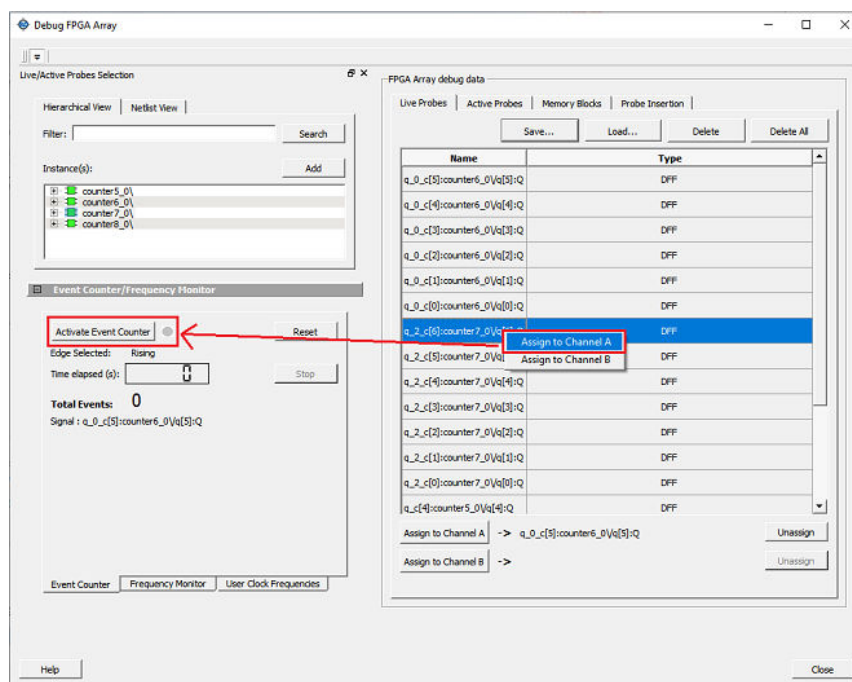
1. Click **Activate Event Counter** and then assign a signal to Live Probe Channel A.

Figure 3-28. Activating the Event Counter



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

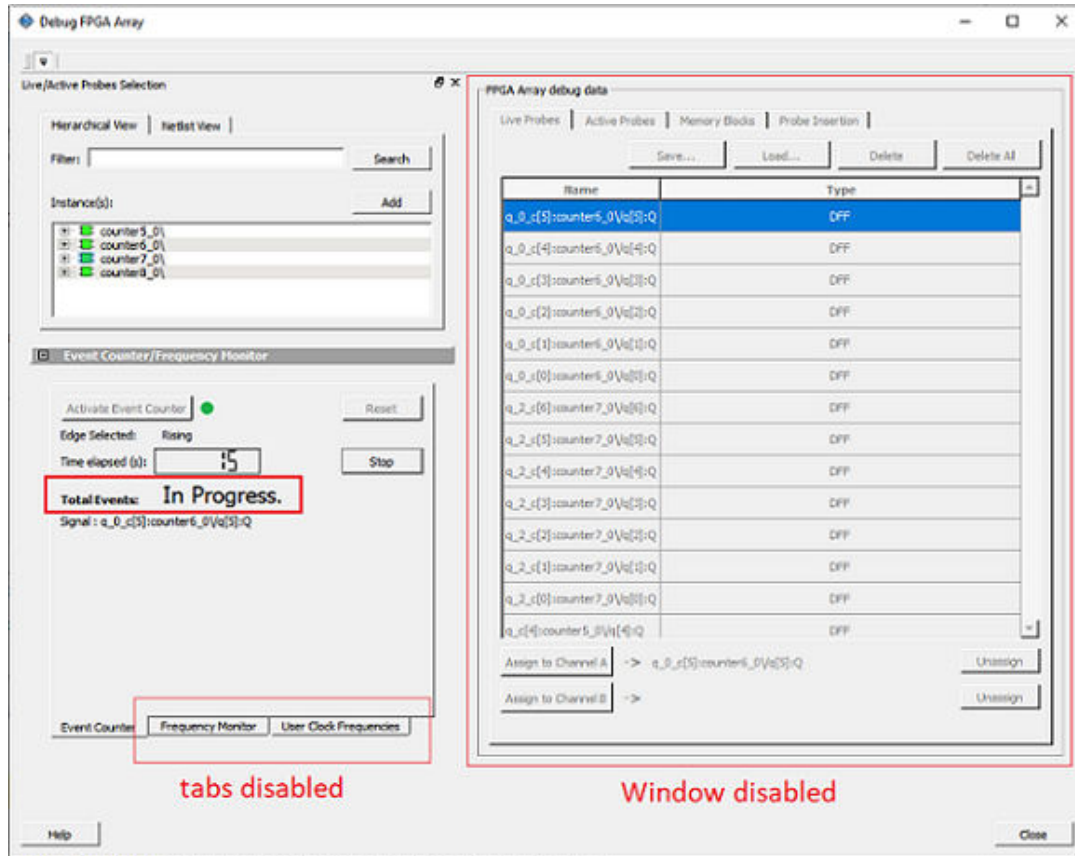
Figure 3-29. Activating the Event Counter - Assign Probe Channel



### 3.10.2 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.

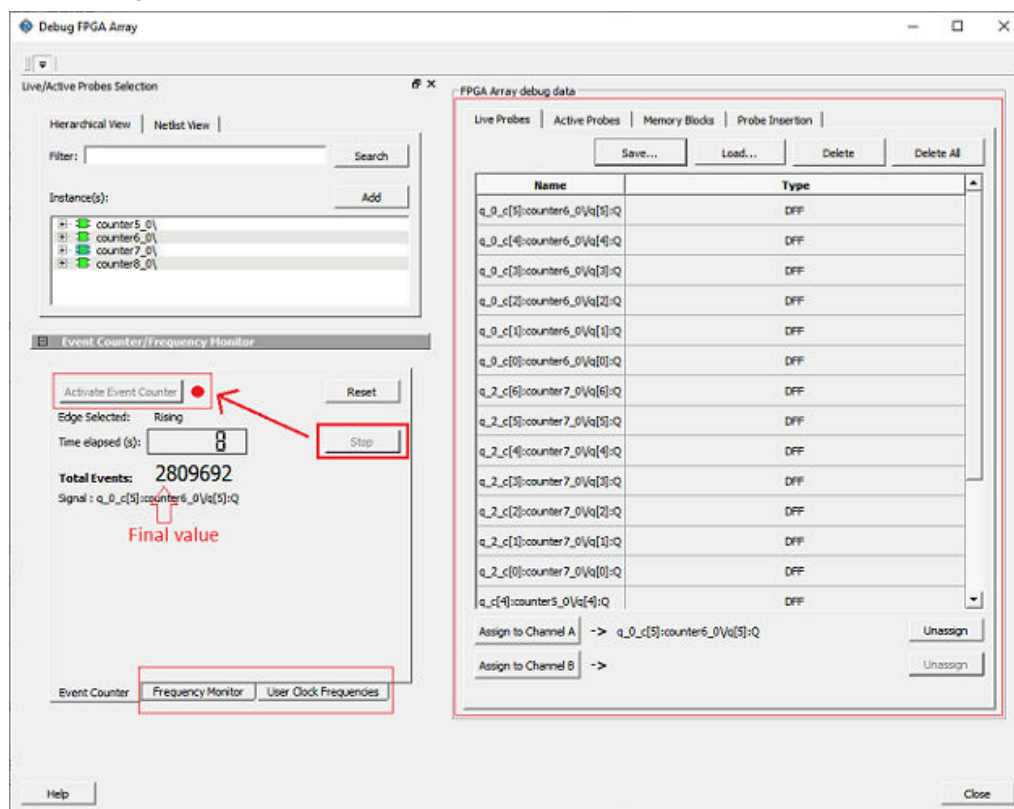
Figure 3-30. Running the Event Counter



### 3.10.3 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 indicate 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 3-31. 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.

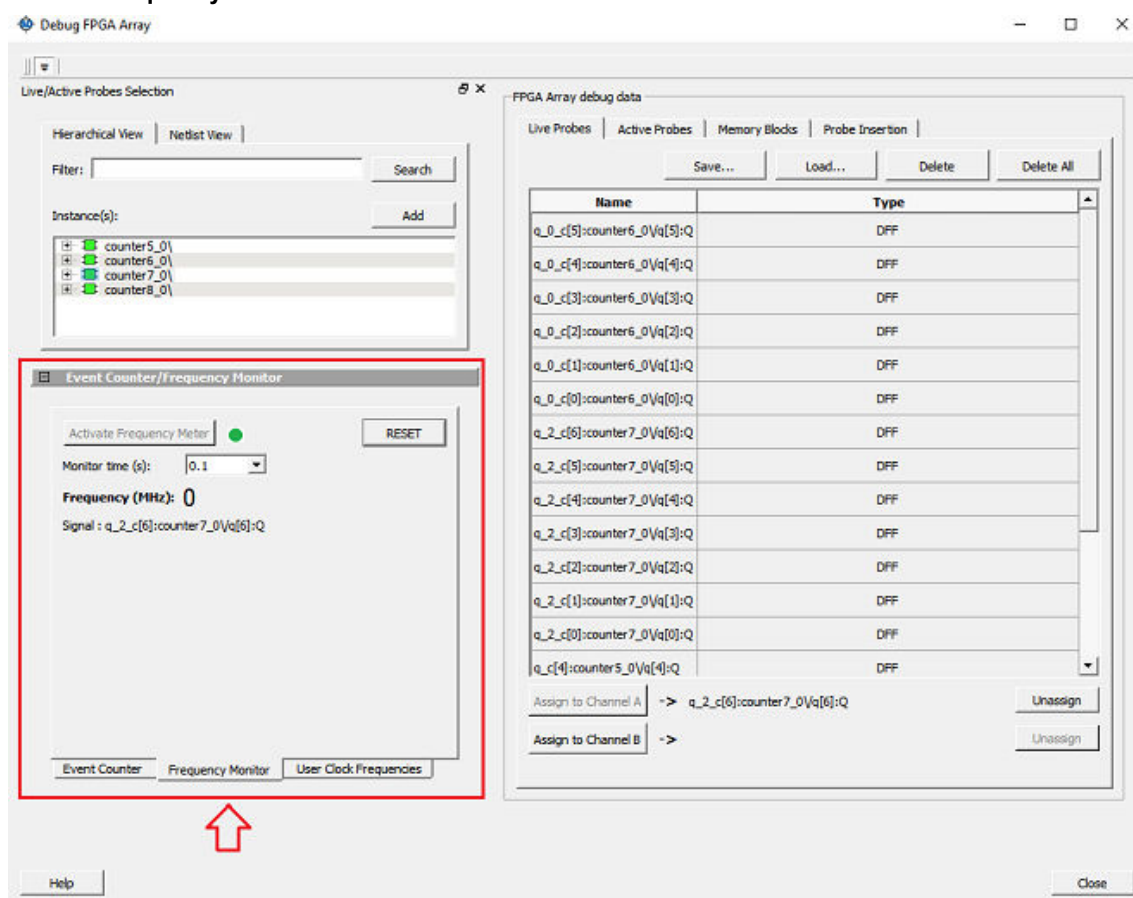
For more information, see [Frequency Monitor](#) and [User Clock Frequencies](#).

### 3.11 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 3-32. Frequency Monitor Tab/UI



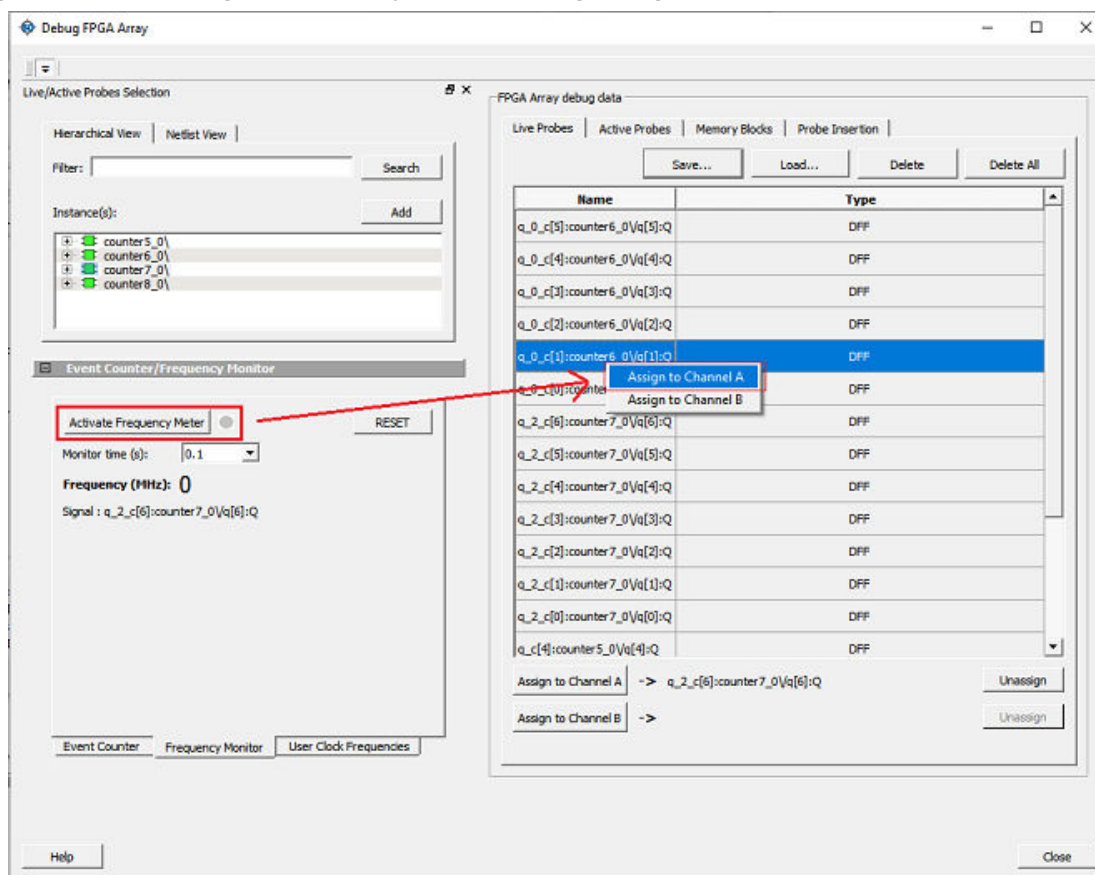
In the **Frequency Monitor** tab, you can activate the Frequency Monitor, change the monitor time (delay to calculate frequency), reset the monitor, and set the frequency in megahertz (MHz). Click the drop-down list to select monitor time value. During the frequency calculation, all tabs on the right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

### 3.11.1 Activating the Frequency Monitor

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

1. 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).

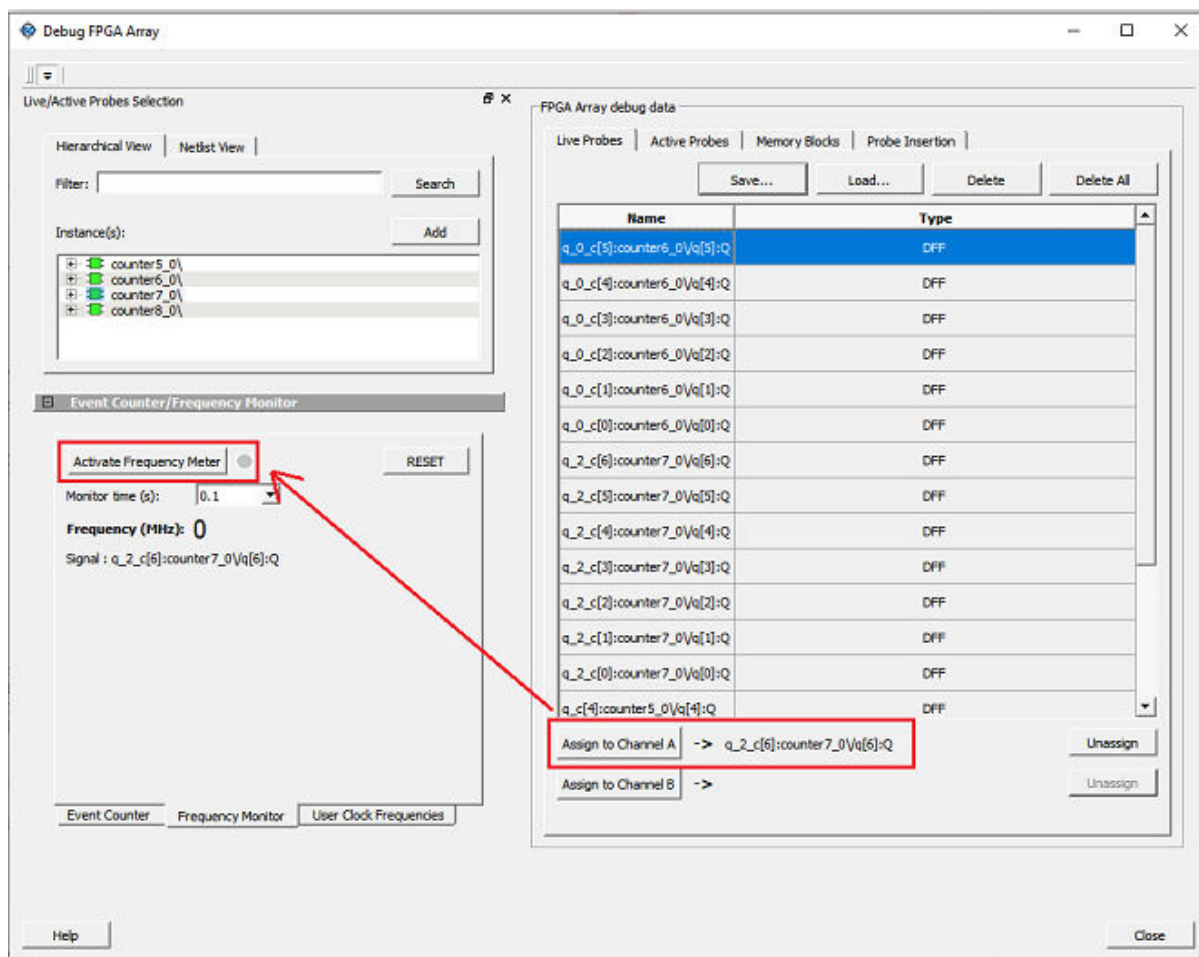
Figure 3-33. Activating the Frequency Monitor - Assign a Signal



2. 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 3-34. Activating the Frequency Monitor



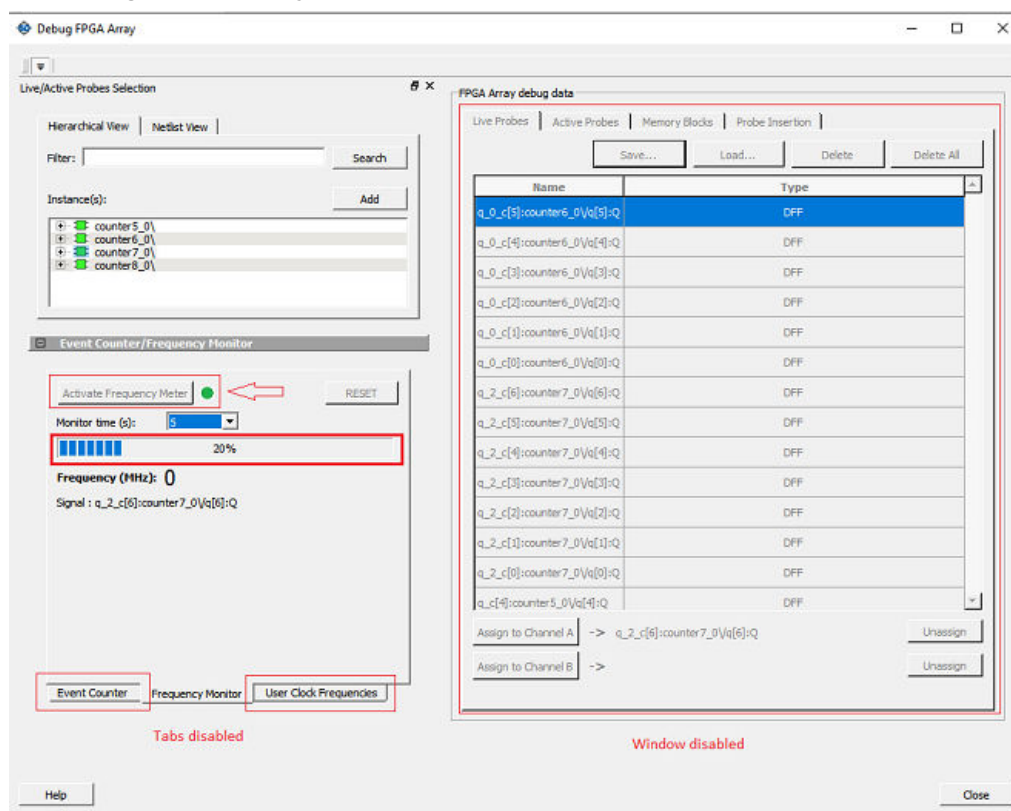
### 3.11.2 Running the Frequency Monitor

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 (see the following figure). The Reset button is also disabled during the run.

When a signal is assigned, the signal name appears next to Signal.



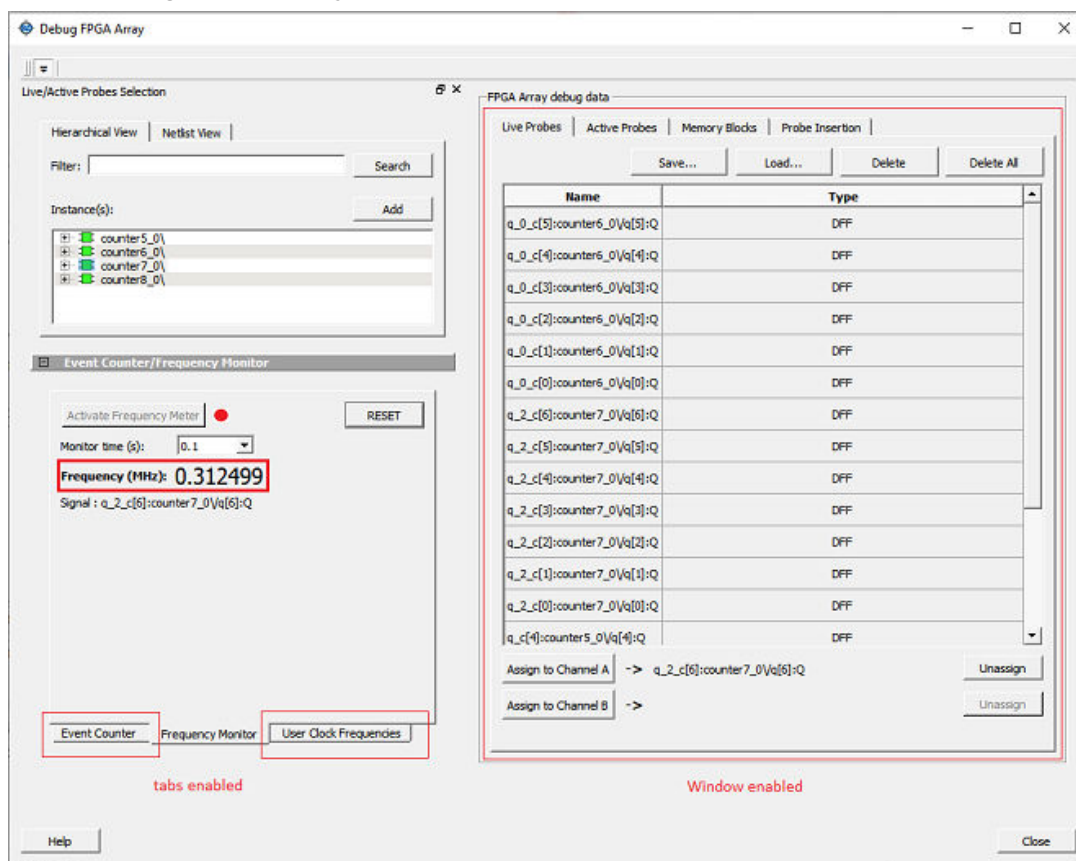
Figure 3-35. Running the Frequency Monitor



### 3.11.3 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 3-36. Stopping the Frequency Monitor

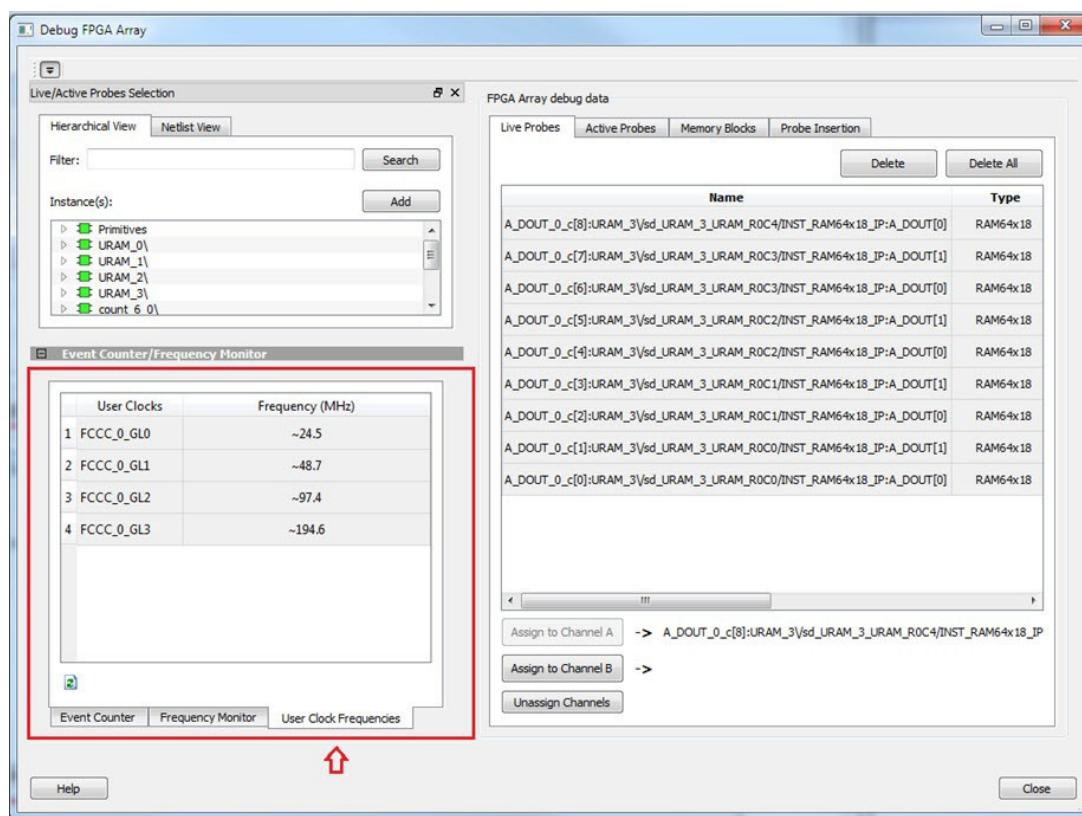


For more information, see [Frequency Monitor](#) and [User Clock Frequencies](#).

### 3.12 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 3-37. User Clock Frequencies Tab/UI



For related information, see [Event Counter](#) and [Frequency Monitor](#).

### 3.13 Debug sNVM

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

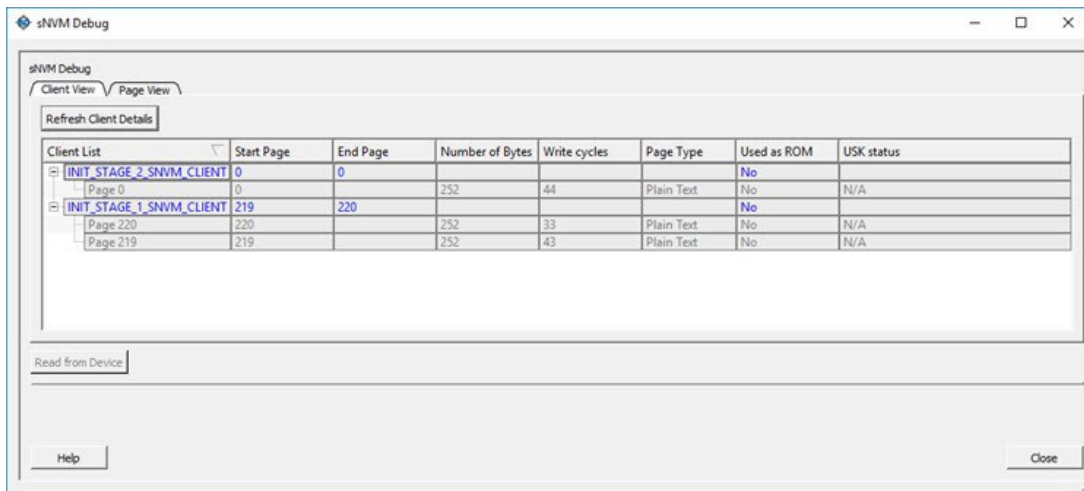
The sNVM Debug window has two tabs – [Client View](#) and [Page View](#).

#### 3.13.1 Client View

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

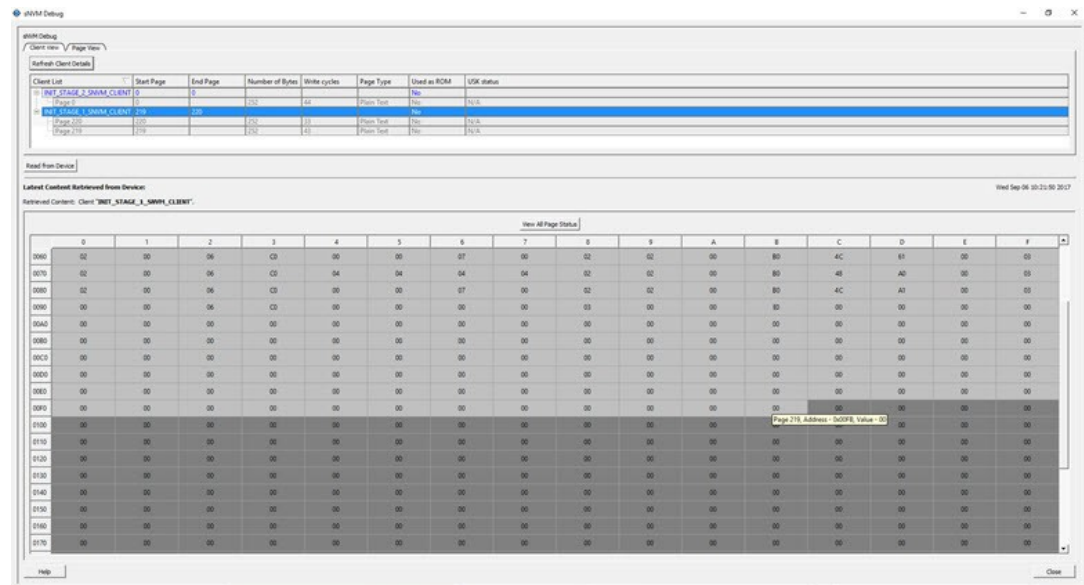
You can select only one client at a time. Pages inside the client cannot be selected. Start Page, End Page, and Number of Bytes are displayed for the selected client.

**Figure 3-38. Client View - Expanded List**



Click the **View All Page Status** button to see information for all pages in the client as shown in the following figure.

**Figure 3-39. Client View - Memory**

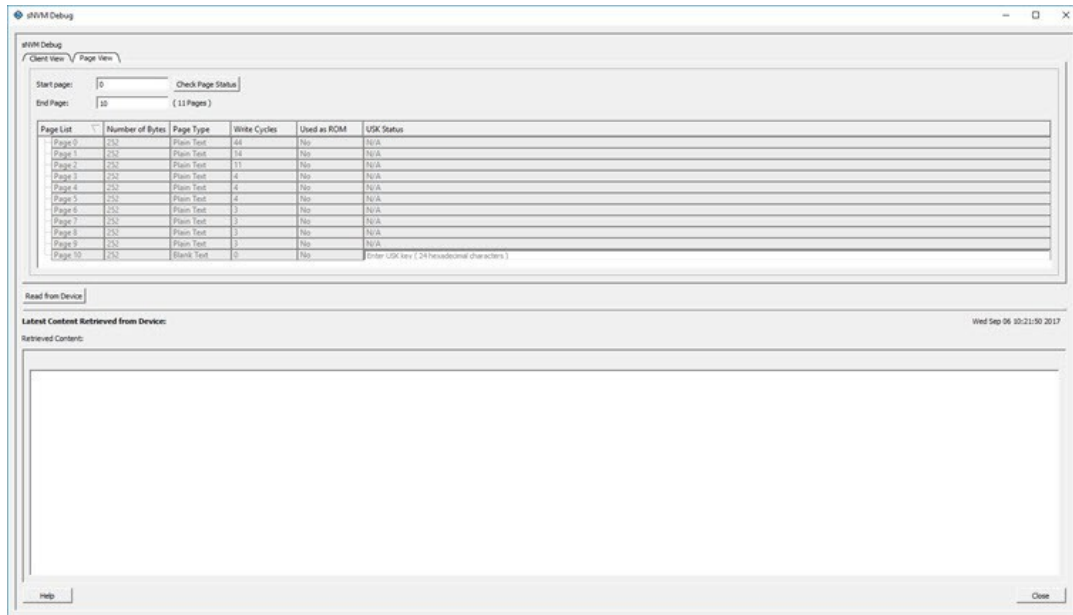


### 3.13.2 Page View

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

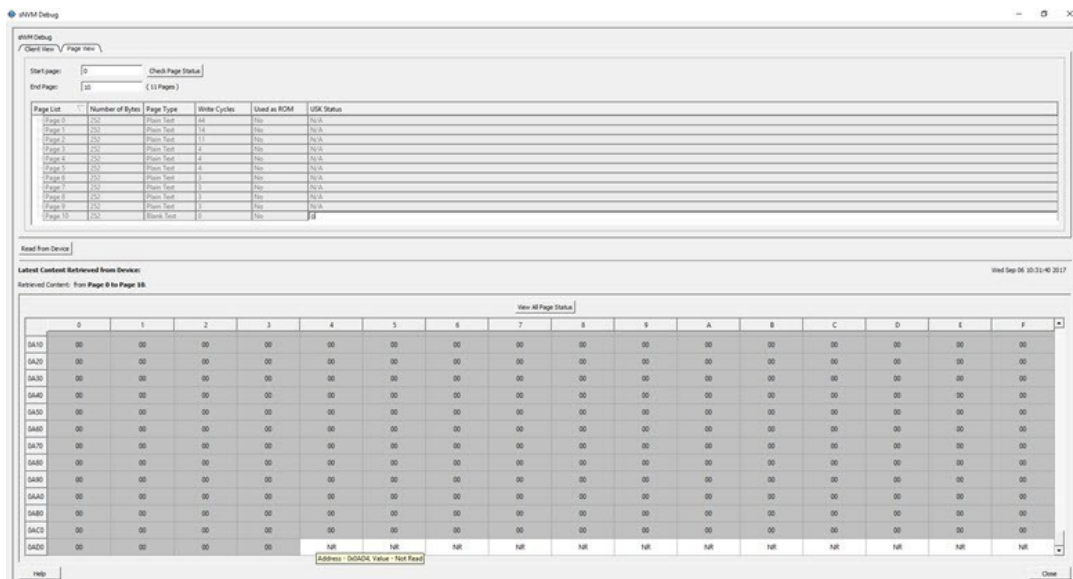
If a page is secured, the default USK is used by SmartDebug to get the page status. If successful, the USK automatically reads the page. If a different USK has been set using system services, use the option to enter the USK, as shown in the following figure.

**Figure 3-40. Page View - Enter USK highlighted**



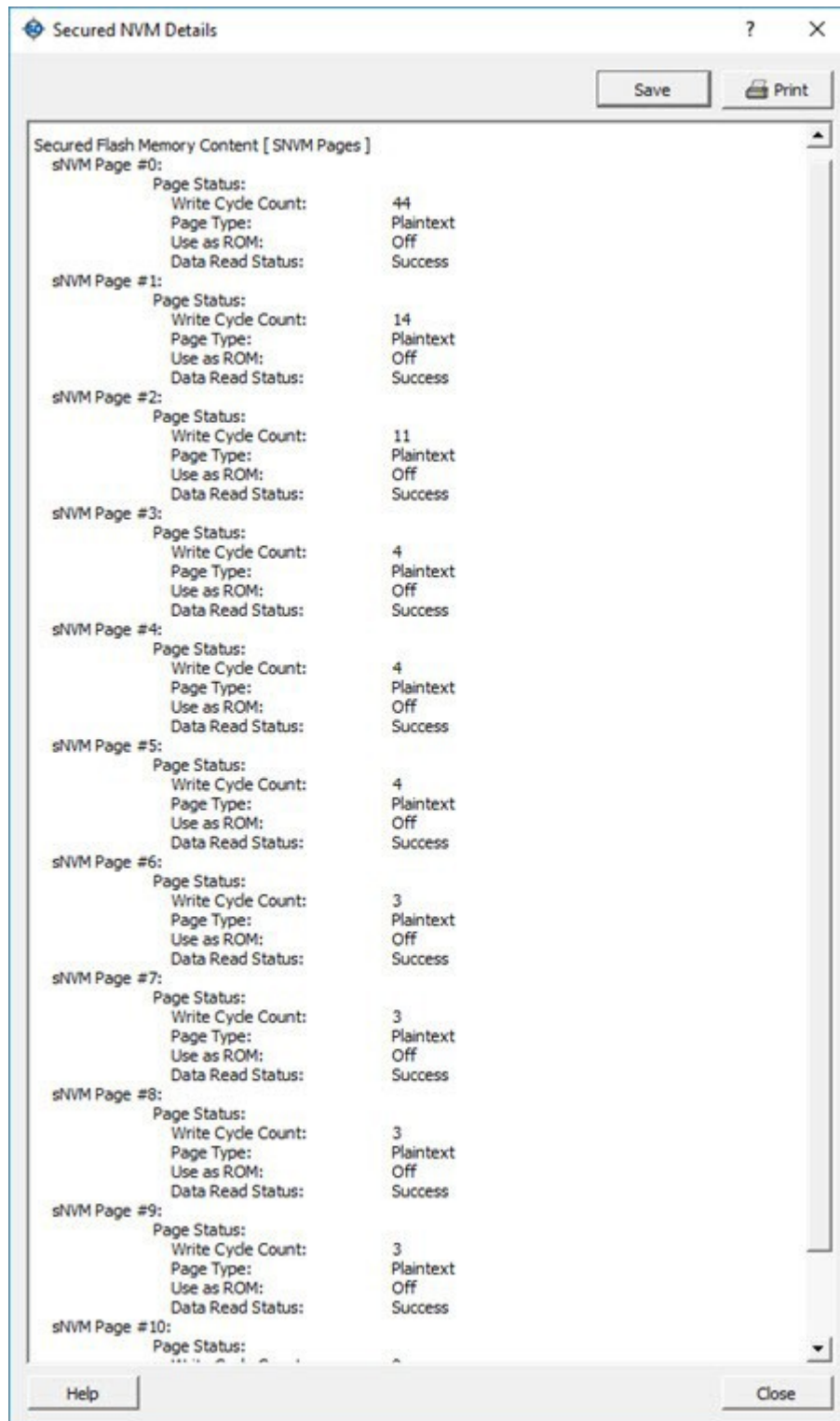
The following figure shows the specified page range.

**Figure 3-41. Page View - Page Range**



The following figure shows the page status.

Figure 3-42. View All Page Status



### 3.13.3 Read Operation

#### Client View

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

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

#### Page View

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

### 3.13.4 Runtime Operations

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

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

The above operations are not possible if the page is used as ROM. You can refresh page status in SmartDebug:

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

If the security key has been changed, SmartDebug prompts you to enter the USK manually. Enter the USK in the USK Status column (Client View tab and Page View tab). By default, the USK entered in the configurator as the USK client is used to authenticate the page.

### 3.13.5 Demo Mode

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

## 3.14 Debug Transceiver

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

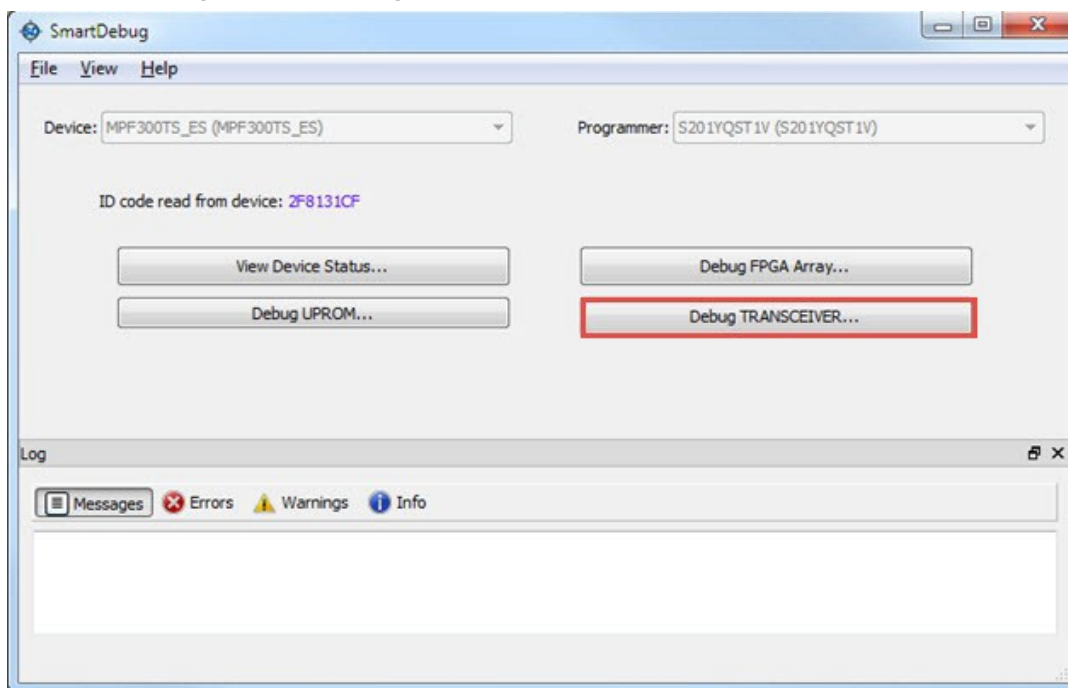
By default, lanes are configured in full duplex mode. Half duplex mode is also supported for each lane in the Quads. More information about half duplex mode lane configuration is provided in the following sections:

- [Configuration Report](#)
- [SmartBERT](#)
- [Loopback Modes](#)
- [Static Pattern Transmit](#)
- [Eye Monitor](#)
- [Register Access](#)
- [Signal Integrity](#)
- [Optimize Receiver](#)

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

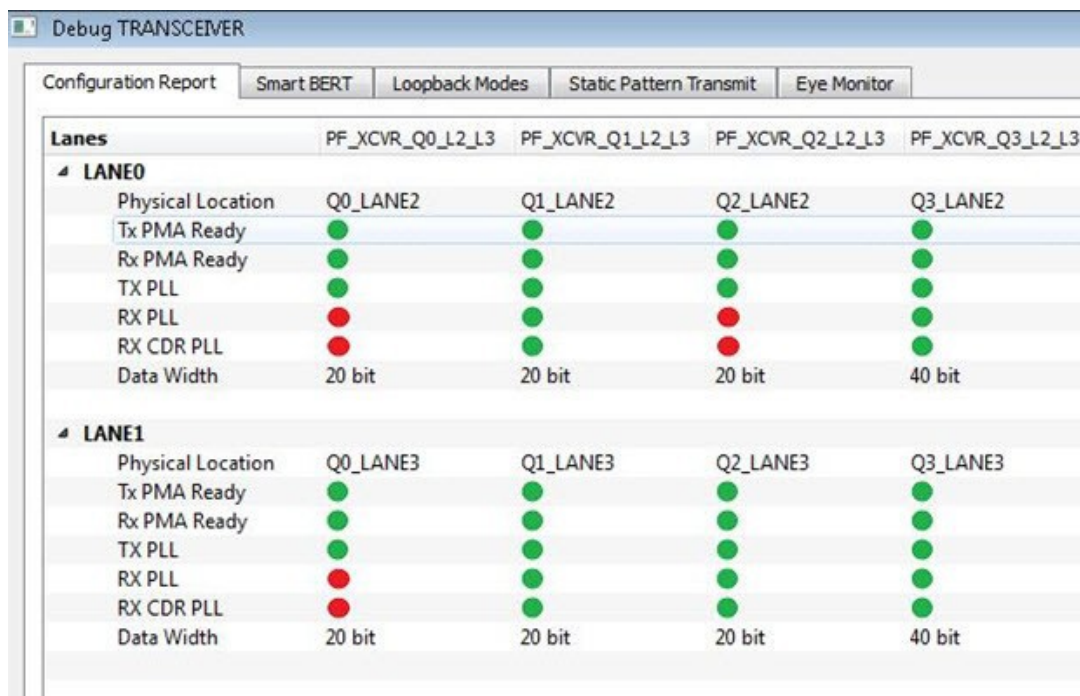


Figure 3-43. SmartDebug Window - Debug TRANSCEIVER



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

Figure 3-44. Debug Transceiver Dialog Box



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

- [Configuration Report](#) (shown by default when the dialog box opens)
- [SmartBERT](#)
- [Loopback Modes](#)
- [Static Pattern Transmit](#)

- [Eye Monitor](#)
- [Register Access](#)

### 3.14.1 Configuration Report

Configuration Report is the first tab in the **Debug TRANSCEIVER** dialog box, and is shown by default when the dialog box opens. The **Configuration Report** shows the physical location, status/health, and data width for all lanes of all the quads enabled in the system controller.

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

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

**Figure 3-45. Debug TRANSCEIVER - Configuration Report**

Lanes	Q0L0_0/PF_XCVR_C0_0/I_XCVR	Q0L1_TX_I2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR	Q0L1_TX_I2_RX_0/PF_XCVR_ERM_C2_RxOnly_0/I_XCVR	Q0L3_0/PF_XCVR_C0_0/I_XCVR
Physical Location	Q0_LANE0	Q0_LANE1	Q0_LANE2	Q0_LANE3
Tx PMA Ready	●	●	NA	●
Rx PMA Ready	●	NA	●	●
TX PLL	●	●	NA	●
RX PLL	●	NA	●	●
RX CDR PLL	●	NA	●	●
Data Width	40 bit	40 bit	40 bit	40 bit

Parameter information is shown in a tabular format, with lane numbers as rows and transceiver instance names as columns. The lane parameters are as follows:

- **Physical Location:** Physical block and lane location in the system controller.
- **Tx PMA Ready:** Indicates if the Tx of the lane is powered up and ready for transactions. Rx-only lane in half duplex mode is shown as "NA".
- **Rx PMA Ready:** Indicates if the Rx of the lane is powered up and ready for transactions. Tx-only lane in half duplex mode is shown as "NA".
- **TX PLL:** Indicates if the lane is locked onto TX PLL. Rx-only lane in half duplex mode is shown as "NA".
- **RX PLL:** Indicates if the lane is locked onto RX PLL. Tx-only lane in half duplex mode is shown as "NA".
- **RX CDR PLL:** Indicates if the lane is locked onto the incoming data. Tx-only lane in half duplex mode is shown as "NA".

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

### 3.14.2 Transceiver Hierarchy

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

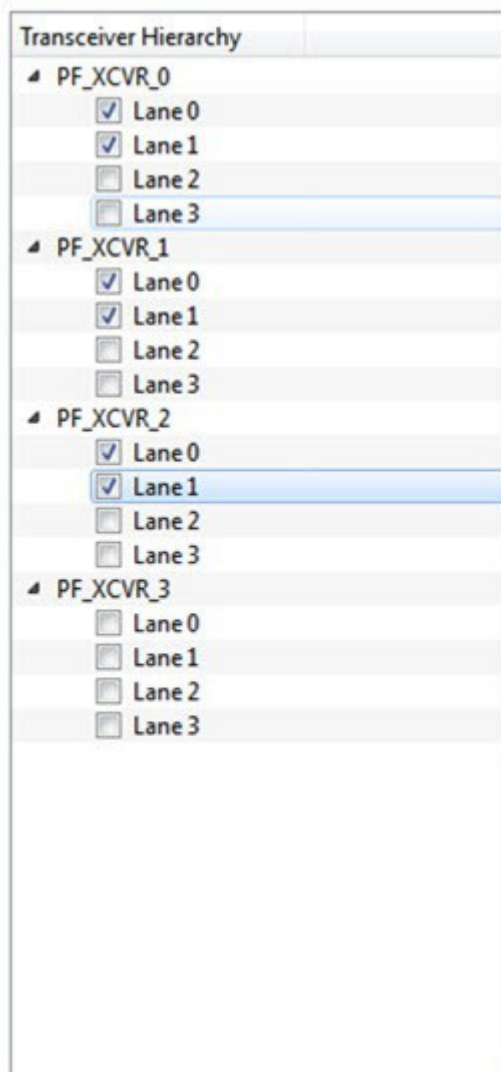
Transceiver Hierarchy view appears on the following tabs:

- [SmartBERT](#)

- [Loopback Modes](#)
- [Static Pattern Transmit](#)
- [Eye Monitor](#)

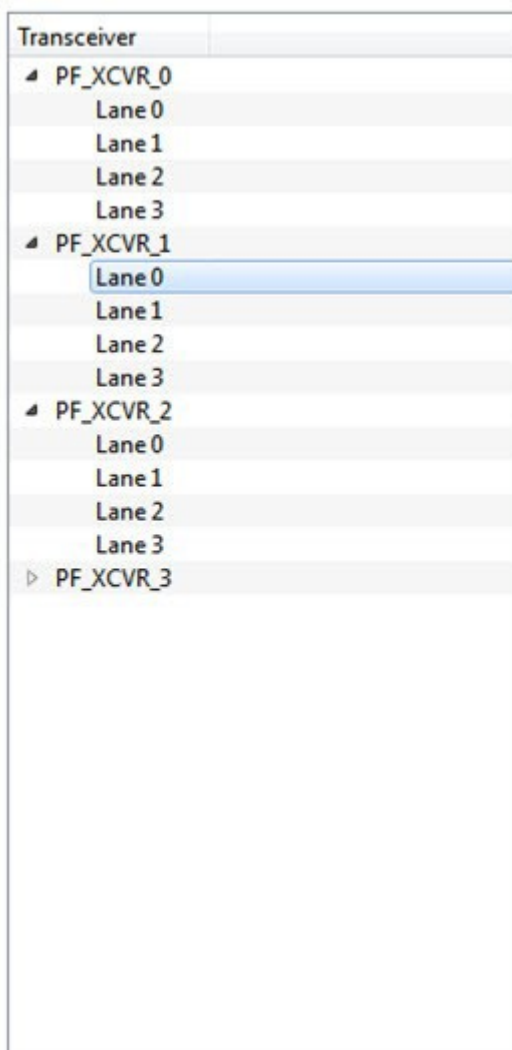
On the SmartBERT, Loopback Modes, and Static Pattern Transmit pages, check boxes allow multiple lanes to be selected for debug, as shown in the following figure.

**Figure 3-46. Transceiver Hierarchy Lane Selection Example - SmartBERT, Loopback Modes, Static Pattern Transmit Pages**



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

Figure 3-47. Transceiver Hierarchy Lane Selection Example - Eye Monitor Page

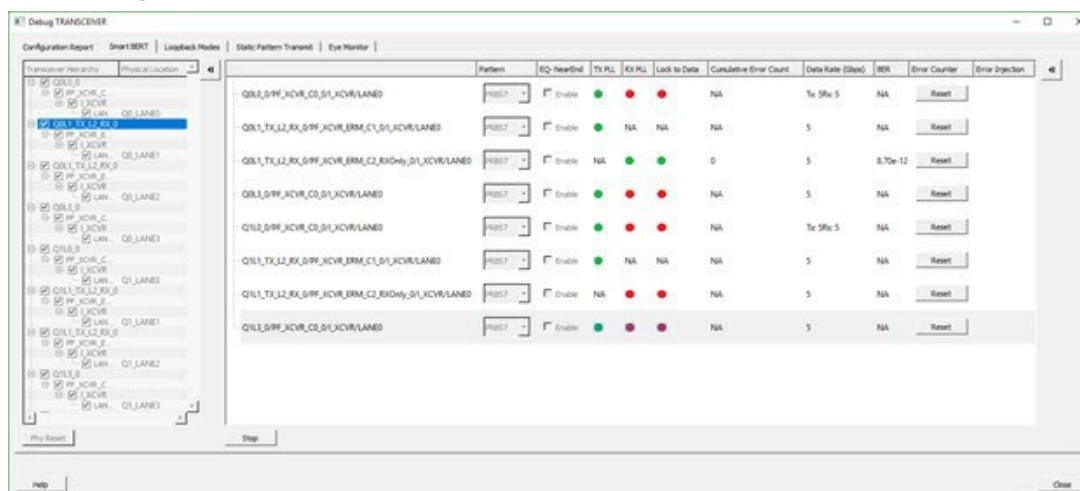


### 3.14.3 SmartBERT

You can select lanes in the Transceiver Hierarchy and use debug options to run SmartBERT tests on the SmartBERT page of the Debug TRANSCEIVER dialog box.

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

Figure 3-48. Debug TRANSCIEVER - SmartBERT



The following input options and outputs are represented as columns:

- **Pattern:** Input option. Select a PRBS pattern type from the drop-down list: PRBS7, PRBS9, PRBS15, PRBS23, or PRBS31. The default is PRBS7.
- **EQ-NearEnd:** Input option. When checked, enables EQ-NearEnd loopback from Lane Tx to Lane Rx. Disabled for half duplex mode.
- **TX PLL:** Indicates if lane is locked onto TX PLL when the SmartBERT test is in progress. Rx Only lane in half duplex mode is shown as "NA".
  - Gray: Indicates test is not in progress
  - Green: Indicates lane is locked onto TX PLL
  - Red: Indicates lane is not locked onto TX PLL
- **RX PLL:** Indicates if lane is locked onto RX PLL when the SmartBERT test is in progress. Tx Only lane in half duplex mode is shown as "NA".
  - Gray: Indicates test is not in progress
  - Green: Indicates lane is locked onto TX PLL
  - Red: Indicates lane is not locked onto TX PLL
- **Lock to Data:** Indicates if lane is locked onto incoming data / RX CDR PLL when the SmartBERT test is in progress. Tx Only lane in half duplex mode is shown as "NA".
  - Gray: Indicates test is not in progress
  - Green: Indicates lane is locked onto TX PLL
  - Red: Indicates lane is not locked onto TX PLL
- **Cumulative Error Count:** Displays the error count when the SmartBERT test is in progress.
- **Data Rate:** Data rates are shown according to the configured data rates for all duplex modes except for Independent TxRx, where both Tx data rate and Rx data rate are shown. See the figure above.
- **BER:** Calculates the Bit Error Rate (BER) from the cumulative error count and data rate and displays it in the column.
- **Error Counter Reset:** Resets the error counter and BER of the lane. A reset can be done at any time.

All output parameters are updated approximately once per second, with their values retrieved from the device. To add lanes, in the Transceiver Hierarchy, check the boxes next to the lanes to be added. To remove lanes, uncheck the boxes next to the lanes to be removed.

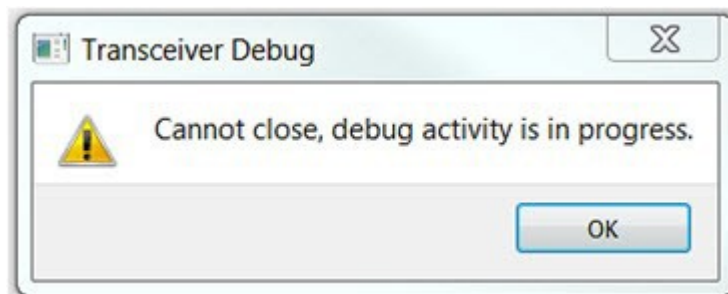
Select the desired options and click **Start** to start the Smart BERT test on all selected lanes. A popup message appears if a test cannot be started on one lane, multiple lanes, or all lanes. Tests will start normally on all unaffected lanes.

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

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

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

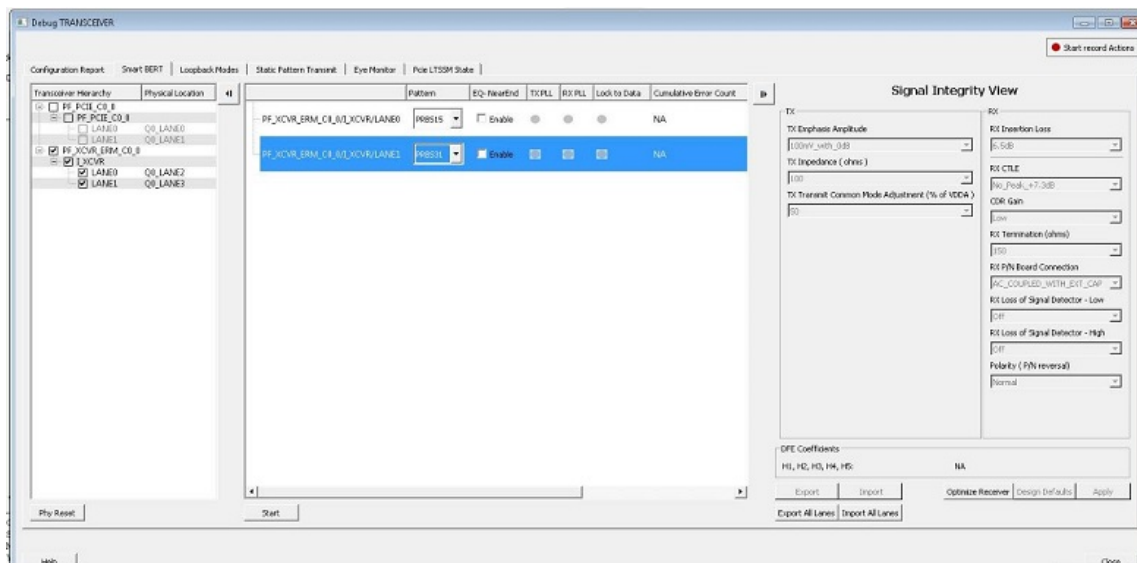
**Note:** You cannot close the SmartBERT window when a test is in progress. Attempting to do so will result in the following message:



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

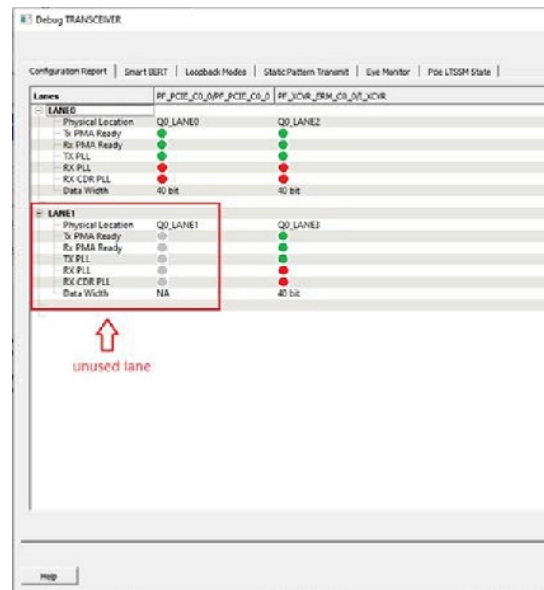
**Note:** SmartBERT tests are disabled for PCIe lanes.

**Figure 3-49. PCIe Lanes with SmartBERT Tests Disabled**



PCIe lanes x1 configuration exposes unused lanes that are disabled on all the tabs in the Debug TRANSCEIVER window.

**Figure 3-50. Configuration Report Showing Unused PCIe Lane**



### 3.14.3.1 SmartBERT IP

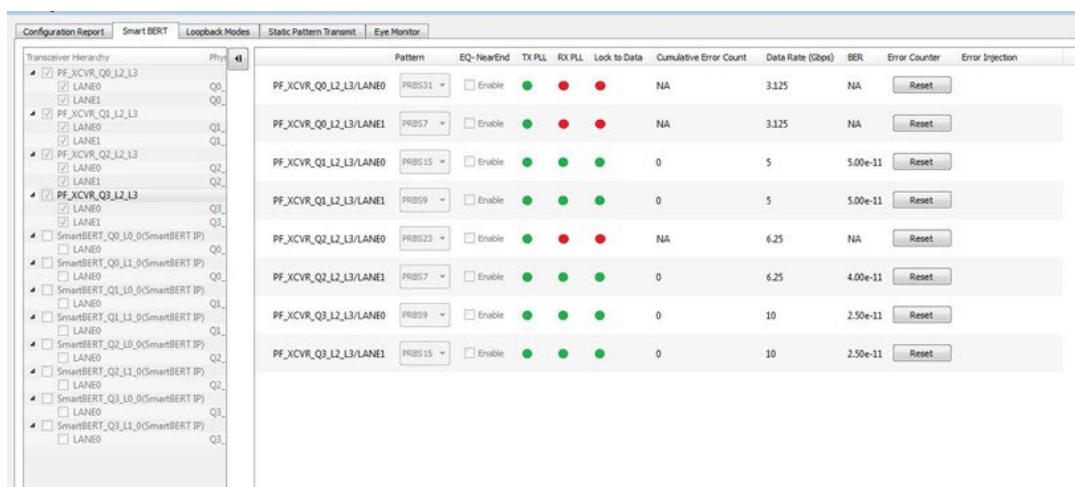
The CoreSmartBERT core provides a broad-based evaluation and demonstration platform for PolarFire transceivers (PF\_XCVR). Parameterizable to use different transceivers and clocking topologies, the SmartBERT core can also be customized to use different line rates and reference clock rates. Data pattern generators and checkers are included for each PF\_XCVR, giving several different Pseudo-random binary sequences PRBS (27, **223**, **215**, and **231**).

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

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

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

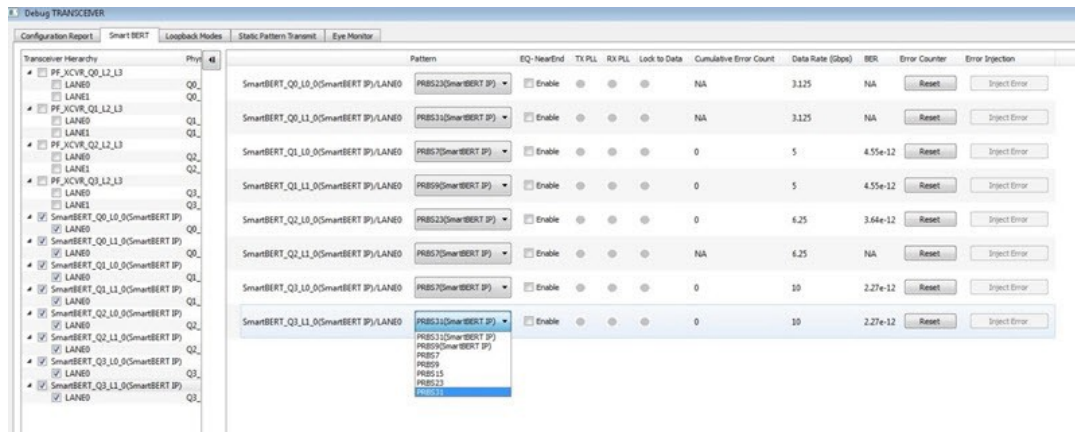
**Figure 3-51. SmartBERT IP - View all Lanes**



You can select patterns for the added lane(s) from a drop-down list as shown in the following figure.



**Figure 3-52. SmartBERT IP - Select a Pattern**



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

### Error Injection

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

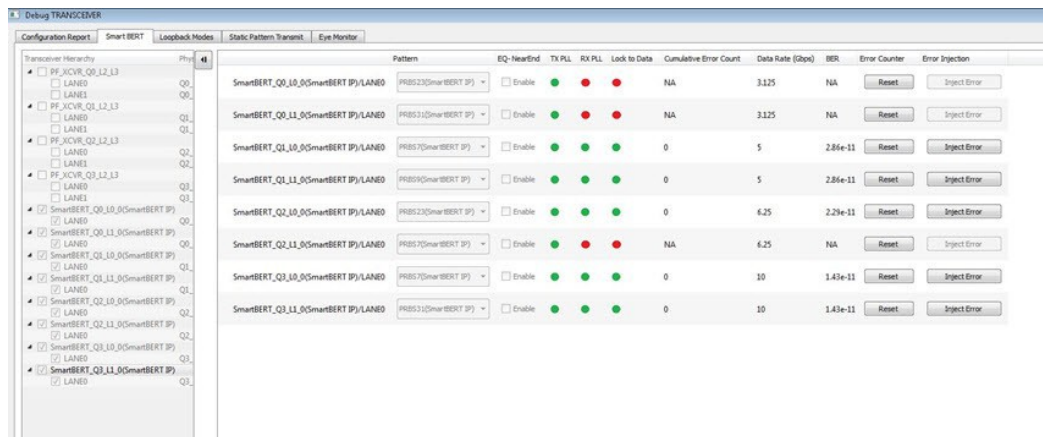
**Note:** This column does not appear for non-SmartBERT IP lanes, or if a non-configured PRBS pattern has been selected.

### Error Count

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

The following figure shows the Reset and Inject Error buttons.

**Figure 3-53. SmartBERT IP - Reset and Inject Error**

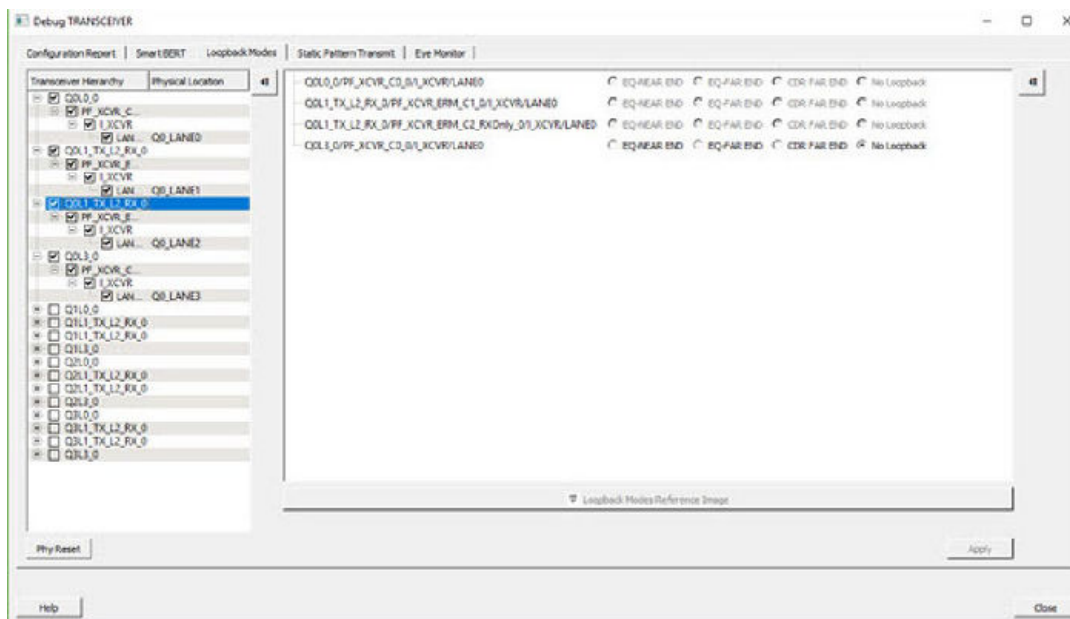


### 3.14.4 Loopback Modes

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

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

Figure 3-54. Debug TRANSCEIVER - Loopback Modes



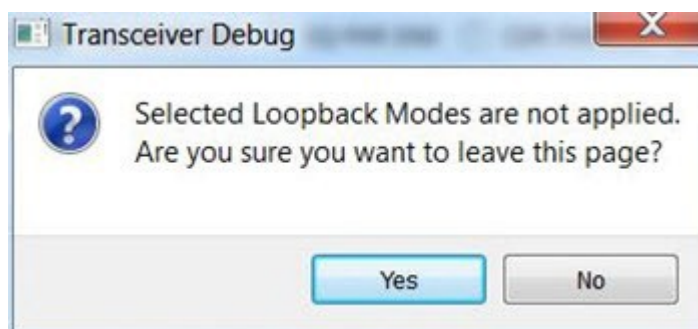
You can select the desired loopback type (EQ-NEAREND, EQ-FAREND, CDRFAREND, or No Loopback) for each lane.

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

- **EQ-NEAR END:** Set EQ-Near End loopback from Lane Tx to Lane Rx. This loopback mode is supported up to 10.3125 Gbps.
- **EQ-FAR END:** Set EQ-Far End loopback from Lane Tx to Lane Rx.
- **CDR FAR END:** Set CDR Far End loopback from Lane Rx to Lane Tx.
- **No Loopback:** Set this option to have no loopback between Lane Tx and Lane Rx. (For external loopback using PCB backplane or High Speed Loopback cables.)

Click **Apply** to enable the selected loopback mode on the lane(s).

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



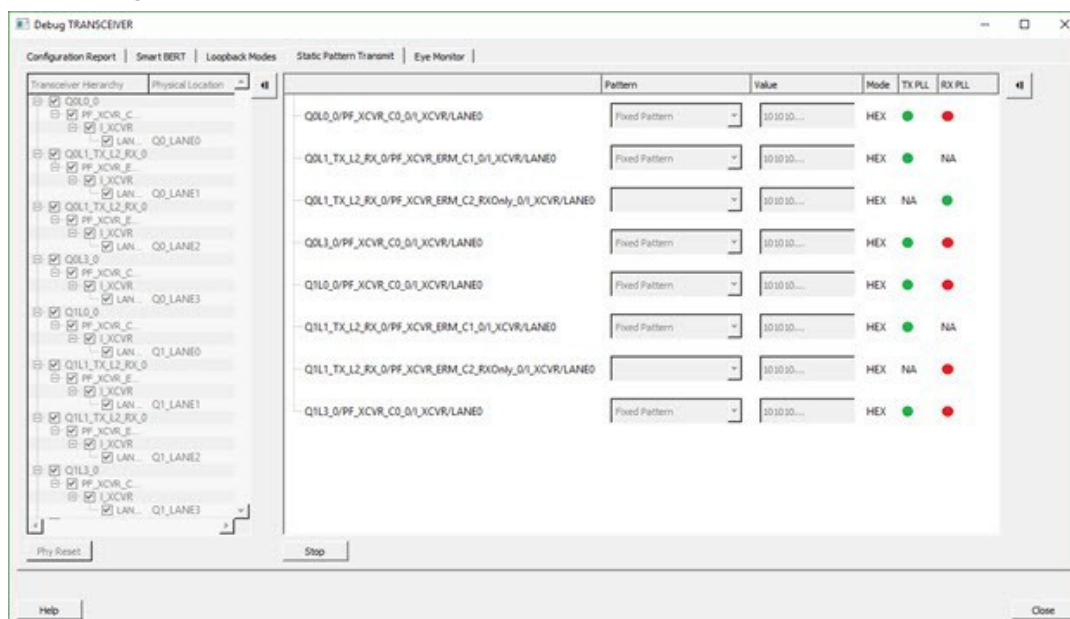
Click **Yes** to ignore the changed selections and move to another selected page. Click **No** to remain on the current page.

### 3.14.5 Static Pattern Transmit

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

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

Figure 3-55. Debug TRANSCEIVER – Static Pattern Transmit



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

- **Pattern:** Pattern selection is available/enabled for all modes except Rx Only, because it is applicable only for Tx. (and Tx is present in Full Duplex, Tx Only, and Independent TxRx)
  - Fixed Pattern is a 10101010... pattern. Length is equal to the data width of the Tx Lane.
  - Max Run Length Pattern is a 1111000... pattern. Length is equal to the data width of the Tx Lane, with half 1s and half 0s.
  - User Pattern is a user defined pattern in the value column. Length is equal to the data width.
- **Value:** Editor available only with the User Pattern type. For other pattern type selections, it is disabled.
  - Takes the input pattern to transmit from the Lane Tx of selected lanes.
  - Pattern type should be Hex numbers, and not larger than the data widthselected.
  - Internal validators dynamically check the pattern and indicate when an incorrect pattern is given as input.
- **Mode:** Currently, HEX mode is supported for pattern type.
- **TX PLL:** Indicates Lane lock onto TX PLL when Static Pattern Transmit is in progress.
  - *Gray:* Test is not in progress.
  - *Green:* Lane is locked onto TXPLL.
  - *Red:* Lane is not locked onto TXPLL.
- **RX PLL :** Indicates Lane lock onto RX PLL when Static Pattern Transmit is in progress.
  - *Gray:* Test is not in progress.
  - *Green:* Lane is locked onto RXPLL.
  - *Red:* Lane is not locked onto RXPLL.

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

### 3.14.6 Eye Monitor

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

To use the Eye Monitor feature, perform the following:

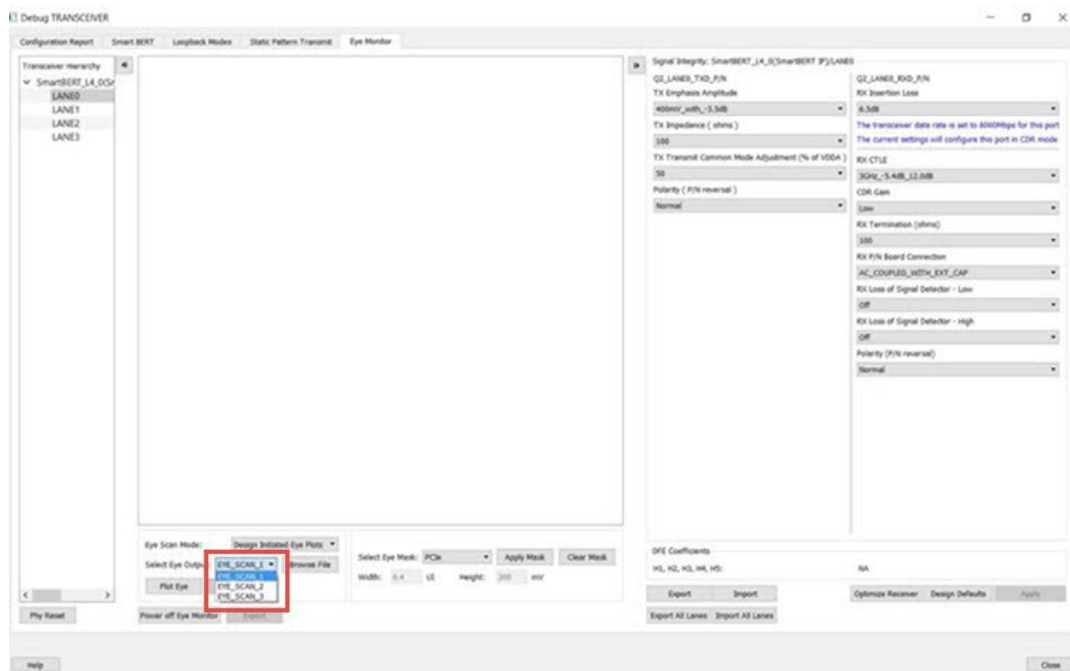
1. Invoke SmartDebug from Libero.
2. Click the **Eye Monitor** tab in the Debug TRANSCEIVER dialog box.

### 3.14.6.1 Select Eye Output

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

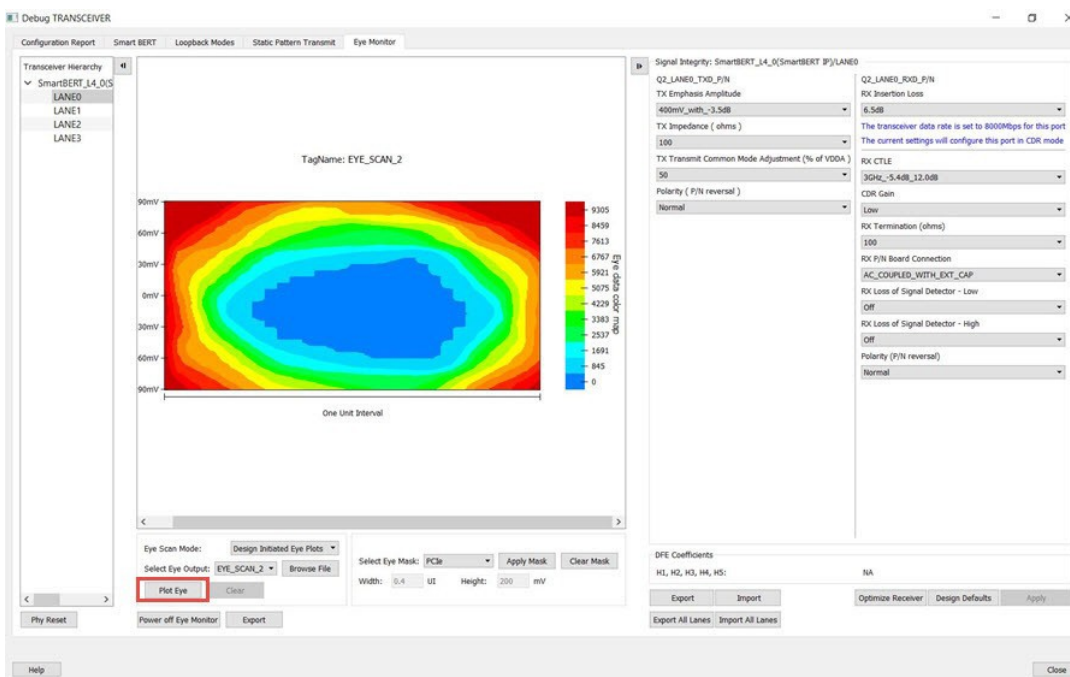
The drop-down list includes all eye outputs logged in the file, as shown in the following figure. If the loaded Design Initiated Eye Plot log file does not contain any eye output, it is disabled.

**Figure 3-56. Eye Monitor Page (Select Eye Output drop-down shown)**



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

**Figure 3-57. Eye Monitor Example**



**Note:** The TagName for the selected eye output is shown above the eye diagram. Ensure data transmission on Lane Rx for successful monitoring.

### 3.14.6.2 Eye Scan Mode

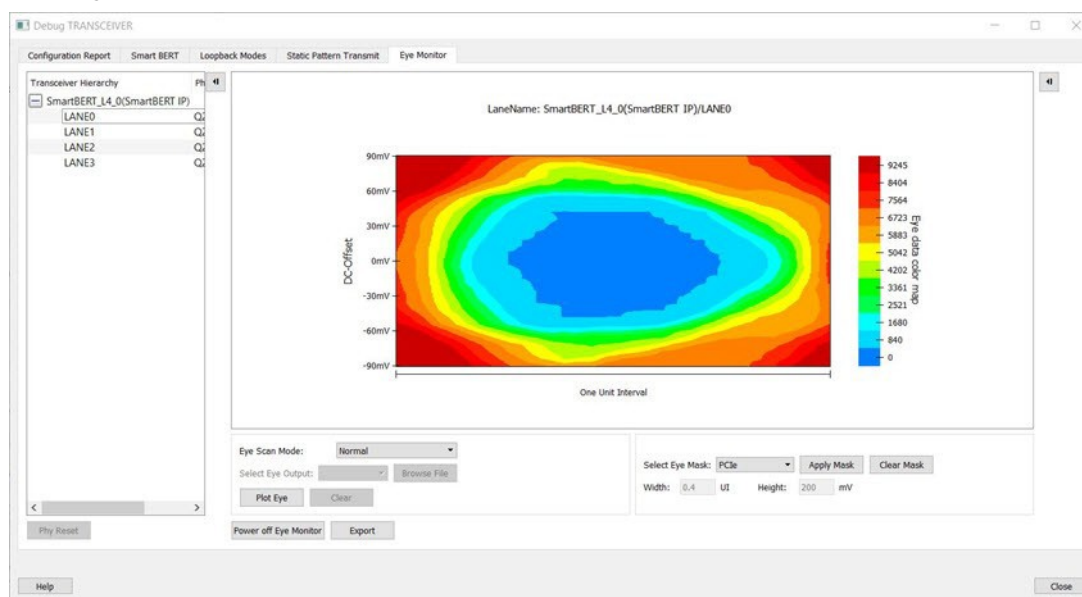
Eye Monitor can be run in two modes: **Normal Mode** or **Infinite Persistent Mode**. Choose the desired mode in the **Eye Scan Mode** drop-down.

#### 3.14.6.2.1 Normal Mode

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

In the Normal mode, **Plot Eye** performs single eye scanning and displays the Eye diagram as shown in the following example figure.

**Figure 3-58. Eye Scan Mode - Normal**

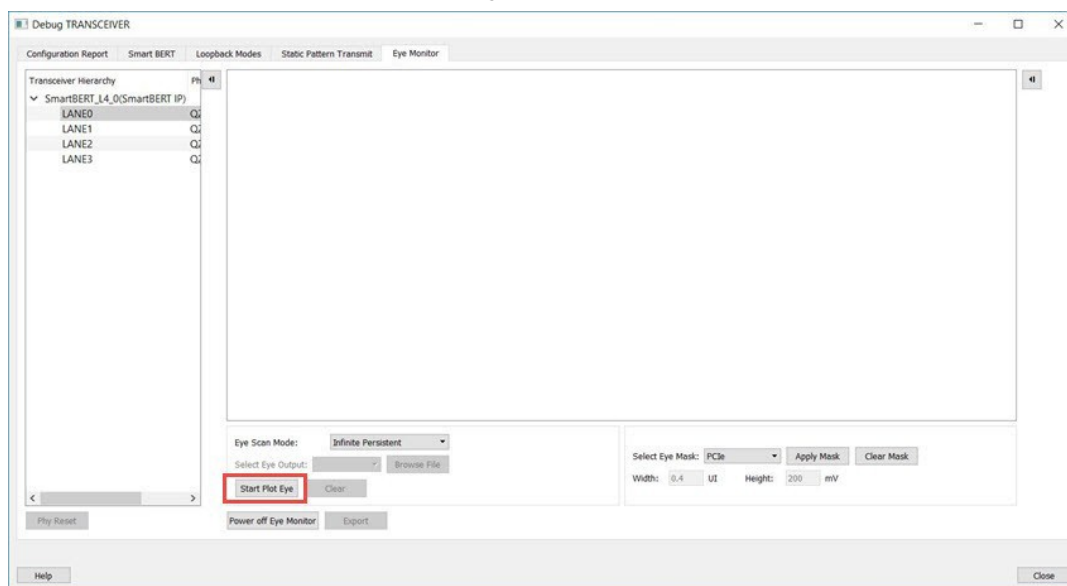


#### 3.14.6.2.2 Infinite Persistent Mode

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

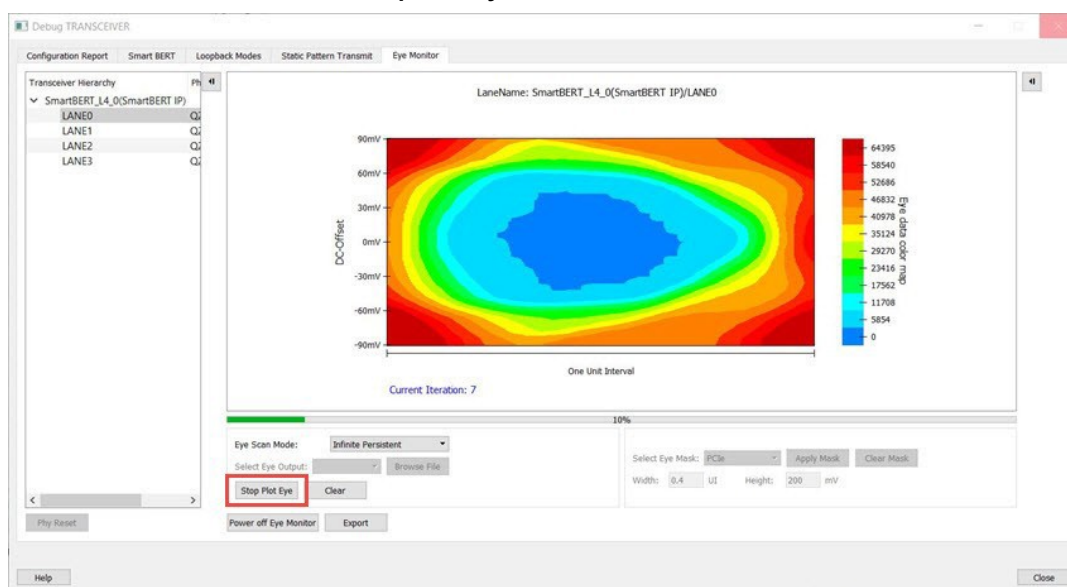
When the Infinite Persistent mode is selected, the **Plot Eye** button changes to **Start Plot Eye**. Click **Start Plot Eye** to start Infinite Persistent eye monitoring as shown in the following figure.

Figure 3-59. Infinite Persistent Mode - Start Plot Eye



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

Figure 3-60. Infinite Persistent Mode - Stop Plot Eye



### 3.14.6.2.3 Clear

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

### 3.14.6.2.4 Additional Eye Output Text Files

Data files are generated in Normal mode and Infinite Persistent mode. These files contain the eye data errors in the matrix format. The name of the file is `Plot_Eye*.txt`, where \* stands for numbering starting from 1.



The files are generated in the `designer` folder of the Libero project for integrated SmartDebug from Libero and in the `standalone` project folder for the standalone SmartDebug.

- In the Normal mode, one eye data error file is generated, as eye scanning is done only once.
- In Infinite Persistent mode, one file is generated per iteration.

Ensure to have sufficient space in the project location when running Infinite Persistent eye monitoring. The numbering used in the file naming continues to increment until the infinite persistent mode eye plot activity is in progress.

**Note:** When you close and restart SmartDebug, the file numbering begins again from 1. Be sure to save these files before starting Eye Monitoring again from a different SmartDebug session; otherwise, they will be overwritten.

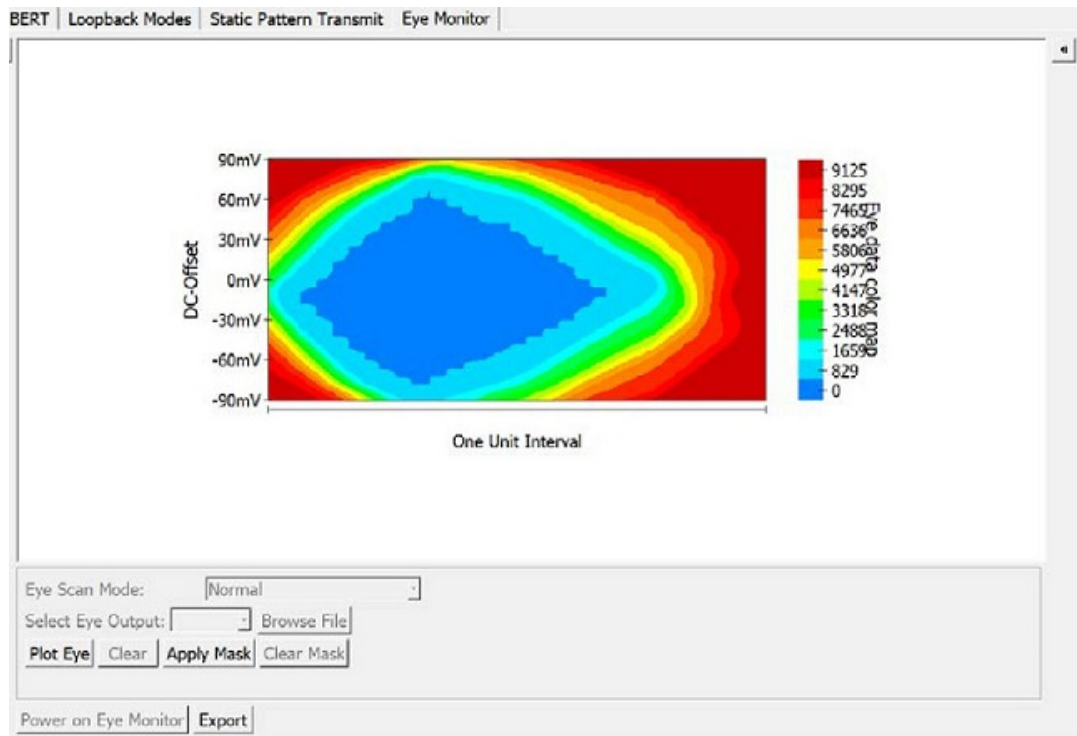
### 3.14.6.2.5 Error Handling

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

### 3.14.6.2.6 Eye Mask

The Eye mask feature has been added to both the normal and infinite persistent modes in the Libero SoC v12.5 release. Eye mask provides a guide to where the best eye opening with least errors can be seen. Both the **Apply Mask** and the **Clear Mask** buttons are disabled in the Default View. Click **Plot Eye** to enable the **Apply Mask** button.

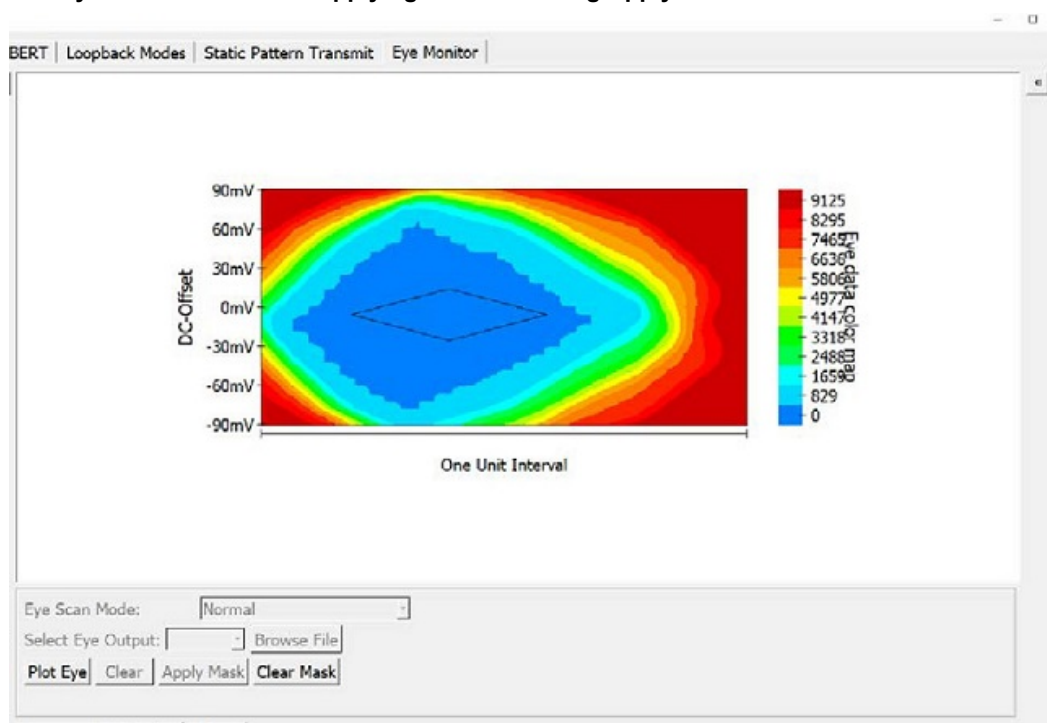
**Figure 3-61. Eye Monitor GUI After Clicking the Plot Eye Button**



After applying the mask, the **Clear Mask** button is enabled and the Eye Mask for the Eye Plot appears.



Figure 3-62. Eye Monitor GUI After Applying the Mask Using Apply Mask Button

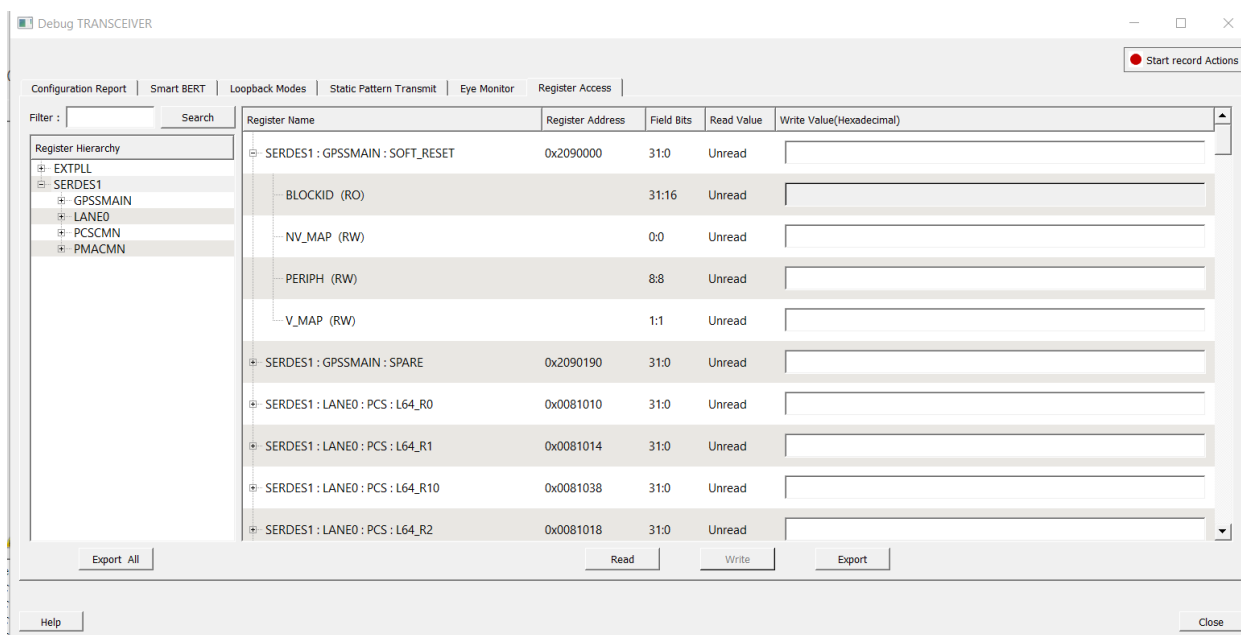


Click **Clear Mask** to clear the Eye Mask for the current Eye Plot and enable the **Apply Mask** button.

### 3.14.7 Register Access

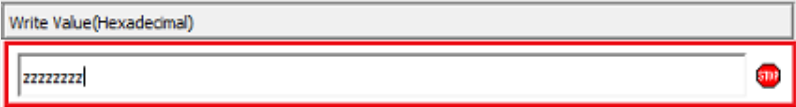
The Register Access page in the Debug TRANSCEIVER window allows you to perform register read, write, export, hide, and export all register operations. The exported register details are saved in a .CSV file.

Figure 3-63. Debug TRANSCEIVER - Register Access



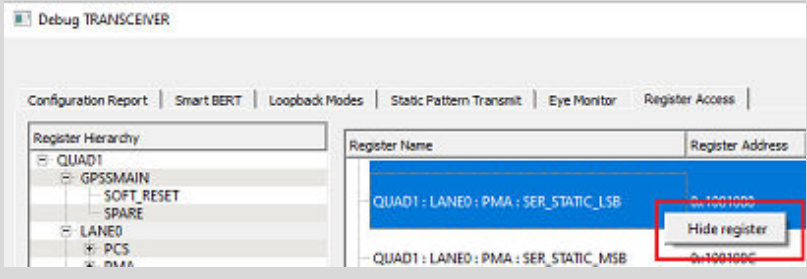
The following table describes the various interface elements on the page.

Table 3-3. Debug TRANSCEIVER - Register Access Options

Option	Description
Register Hierarchy	Lists the design specific registers in a collapsible tree view format.
Register Name	Displays the name of the register that can be looked up in the Register Hierarchy.
Register Address	Displays the physical address of a register. The address is shown in hexadecimal format. The address is calculated based on the Quad, lane number, register type, and offset.
Field Name	Displays the field name based on the register information read from the PF_XCVR.xml file.
Field Bits	Displays bit value based on the field offset and the width of the register.
Read Value	Displays the value read from the device. Click the <b>Read</b> button to retrieve the value from the device. By default, <b>unread</b> is displayed.
Write Value (Hexadecimal)	<p>Enter a hexadecimal register value. An error icon appears, if you enter an incorrect or invalid value.</p> 
Read	<p>Click to retrieve the register value from the device.</p> <p><b>Note:</b> The <b>Read</b> button is disabled when no register is selected in the <b>Register Hierarchy</b> tree view or when SmartDebug is running in the Demo mode.</p>
Write	<p>Click to write the user specified register value.</p> <p><b>Note:</b> The <b>Write</b> button is disabled when no value is specified in the <b>Write Value(Hexadecimal)</b> text box of the selected register or when SmartDebug is running in the Demo mode.</p>
Export	<p>Click to save the selected register details to a .csv file. When clicked, the <b>Register Access Export Action</b> dialog box appears. Specify the file name and the location of the file in the dialog box.</p> <p><b>Note:</b> The <b>Export</b> button is disabled when SmartDebug is running in the Demo mode.</p>
Export All	<p>Click to save all the register details to a .csv file. When clicked, the <b>Register Access Export All Action</b> dialog box appears. Specify the file name and the location of the file in the dialog box.</p> <p>The exported .csv file contains the register name, address, field name, field bits, read, and write values of registers. The first row of the file contains the header information. Register values are written from the second row onwards.</p>

.....continued

Option	Description
Hide register	Select register(s) and right-click to view this option. The <b>Hide registers</b> context menu option helps you remove the selected register(s) from the <b>Register Access</b> table.



### 3.14.8 Signal Integrity

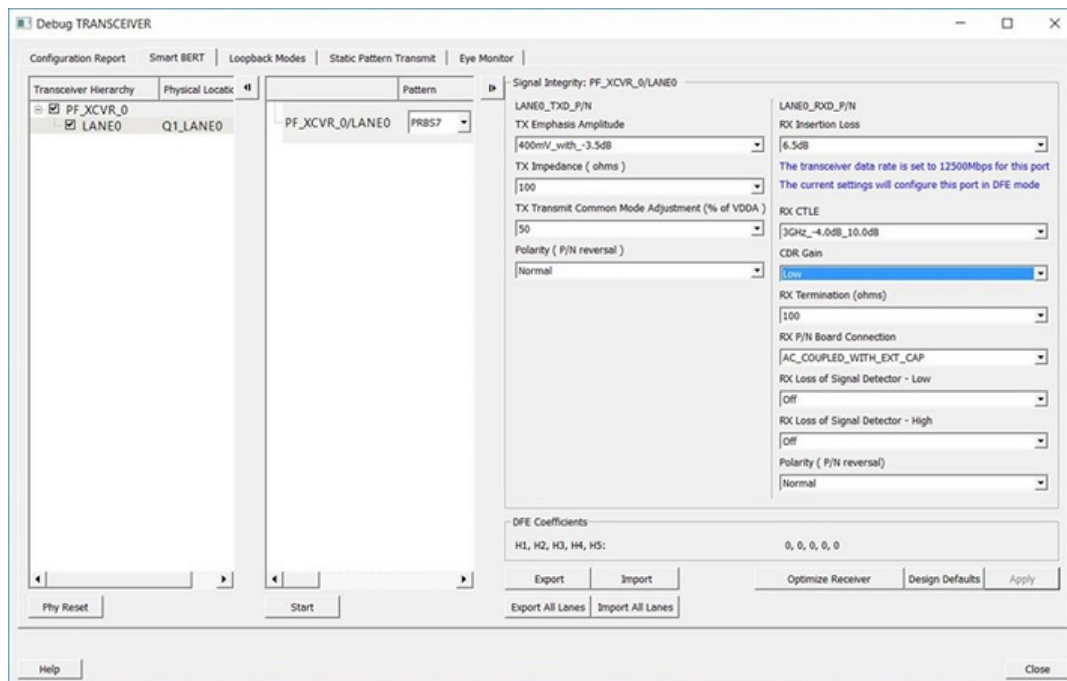
The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of .pdc files.

The Signal Integrity pane appears in the following SmartDebug pages:

- [SmartBERT](#)
- [Loopback Modes](#)
- [Static Pattern Transmit](#)
- [Eye Monitor](#)

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

**Figure 3-64. Debug TRANSCEIVER - Signal Integrity (Full Duplex Mode)**

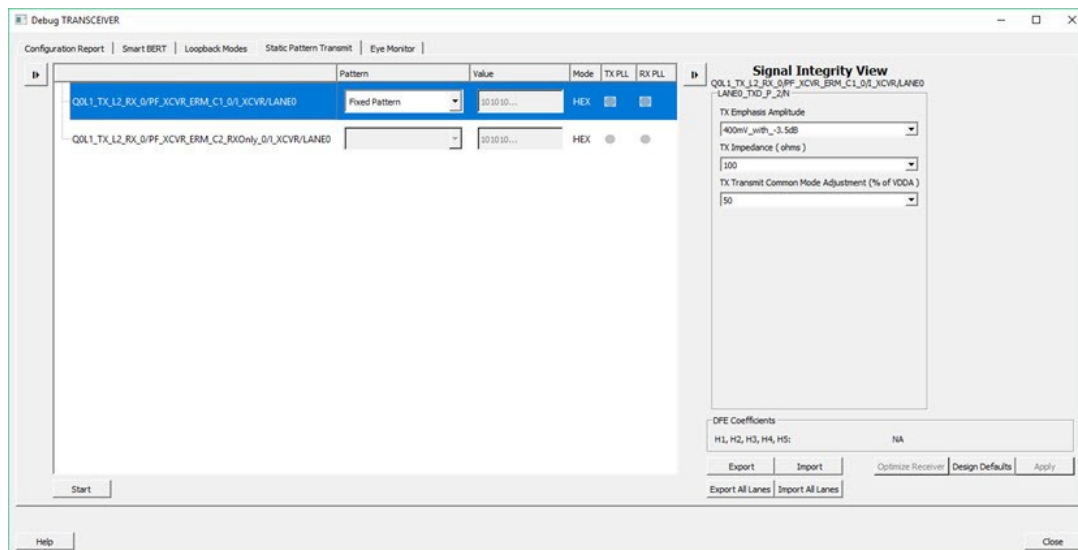


In full duplex mode, all parameters (Tx and Rx) are imported/exported.

The following figure shows Signal Integrity for Tx Only mode, Rx Only mode, and Independent TxRx mode.

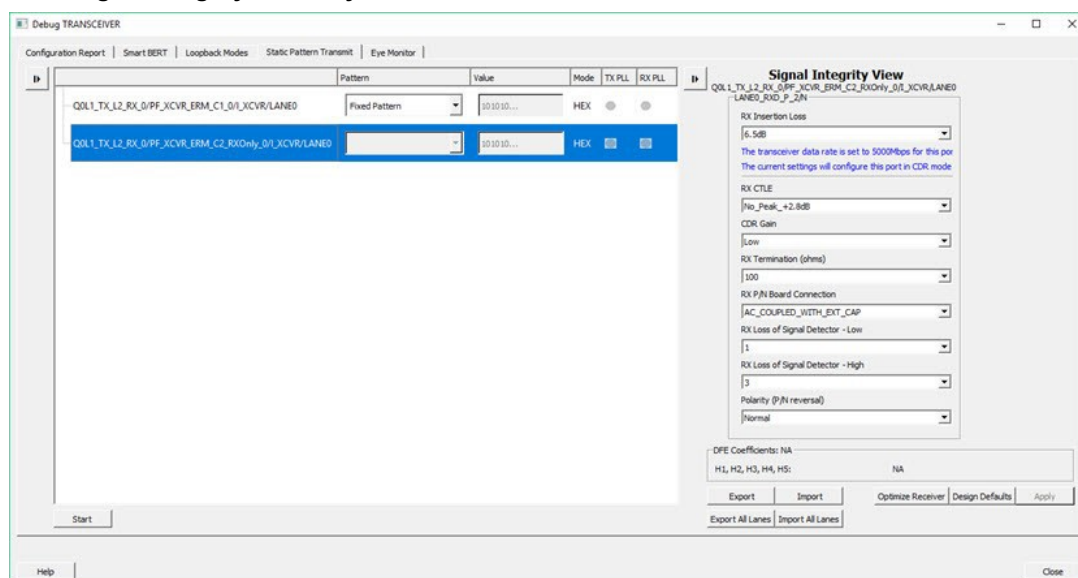
The following figure shows Signal Integrity for Tx Only mode. In this mode, only Tx parameters are imported/exported.

**Figure 3-65. Signal Integrity - Tx Only Mode**



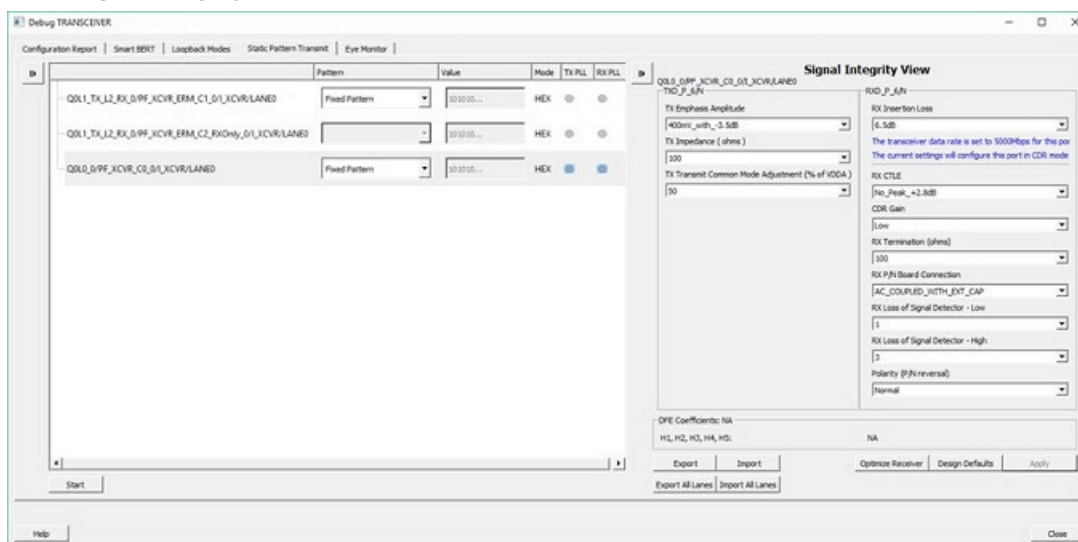
The following figure shows Signal Integrity for Rx Only mode. In this mode, only Rx parameters are imported/exported.

**Figure 3-66. Signal Integrity - Rx Only Mode**



The following figure shows Signal Integrity for independent TxRx mode. In this mode, Tx and Rx parameters are imported/exported.

**Figure 3-67. Signal Integrity - Independent TxRx Mode**



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

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

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

The **Polarity (P/N reversal)** parameter has been added. You can choose Normal or Inverted from the drop-down. Note that this parameter is not available for MPF300T\_ES (Rev C) or MPF300T\_XT (Rev E) devices.

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

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

If you change parameter options Tc1 and click another lane, move to another tab, or click Import, Import All, or Design Defaults without applying the changes, you will see *Some Signal Integrity options are modified. How do you wish to continue?* message.

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

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

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

### 3.14.8.1 Design Defaults

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

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

### 3.14.8.2 Export

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

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

#### 3.14.8.2.1 Export All Lanes

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

### 3.14.8.3 Import

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

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

#### 3.14.8.3.1 Import All Lanes

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

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

### 3.14.8.4 Signal Integrity and Calibration Report

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

```
SIGNAL INTEGRITY AND CALIBRATION REPORT
===== PF_XCVR_3/LANE0
Signal Integrity:
TX_EMPHASIS_AMPLITUDE=400mV_with_-3.5dB
TX_IMPEDANCE=150 TX_TRANSMIT_COMMON_MODE_ADJUSTMENT=50
TX_POLARITY=Normal
TXPLL_BANDWIDTH=LOW
RX_INSERTION_LOSS=6.5dB
RX_CTLE=3GHz_+5.5dB_2.1dB
RX_TERMINATION=100
RX_PN_BOARD_CONNECTION=AC_COUPLED_WITH_EXT_CAP
RX_LOSS_OF_SIGNAL_DETECTOR_LOW=PCIE
RX_LOSS_OF_SIGNAL_DETECTOR_HIGH=PCIE
RX_POLARITY=Normal
-
CTLE Coefficients:
CST1, RST1, CST2, RST2 = 3, 3, 1, 2
-
DFE Coefficients:
H1, H2, H3, H4, H5 = 0, -1, 2, 6, 3
=====
PF_XCVR_3/LANE0
Signal Integrity:
TX_EMPHASIS_AMPLITUDE=400mV_with_-3.5dB
TX_IMPEDANCE=100
TX_TRANSMIT_COMMON_MODE_ADJUSTMENT=80
TX_POLARITY=Normal
TXPLL_BANDWIDTH=LOW
RX_INSERTION_LOSS=17.0dB
RX_CTLE=3GHz_+5.5dB_2.1dB
RX_TERMINATION=100
RX_PN_BOARD_CONNECTION=AC_COUPLED_WITH_EXT_CAP
RX_LOSS_OF_SIGNAL_DETECTOR_LOW=PCIE
RX_LOSS_OF_SIGNAL_DETECTOR_HIGH=PCIE
RX_POLARITY=Normal
-
```

```
CTLE Coefficients:
CST1, RST1, CST2, RST2 = 3, 1, 2, 2
-
DFE Coefficients:
H1, H2, H3, H4, H5 = Not applicable for CDR configured lane
```

### 3.14.8.5 Signal Integrity Parameters in Half Duplex Modes

#### Tx Only XCVR Mode

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

#### Rx Only XCVR Mode

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

#### Independent TxRx XCVR Mode

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

### 3.14.8.6 Optimize Receiver

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

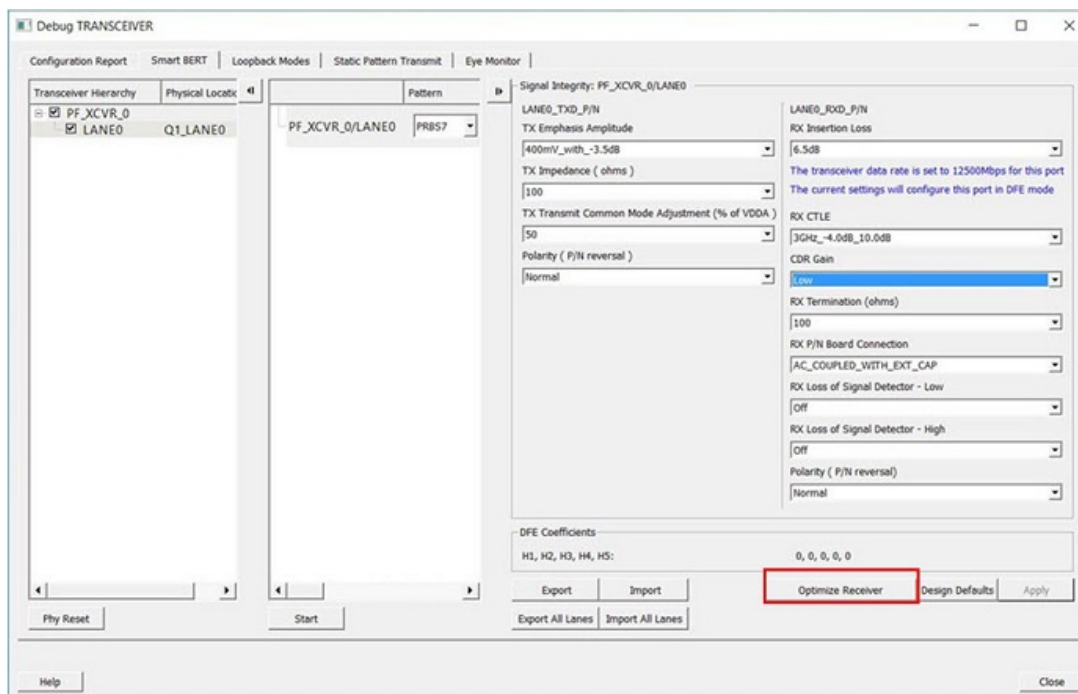
The Optimize Receiver function allows you to optimize the DFE coefficients and/or CTLE settings for the selected lanes, depending on receiver mode. For CDR mode receivers, CTLE settings can be optimized. For DFE mode receivers, CTLE settings and DFE coefficients can be optimized.

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

For information about how to use the optimized coefficients without SmartDebug, see the [PolarFire FPGA Transceiver User Guide](#).

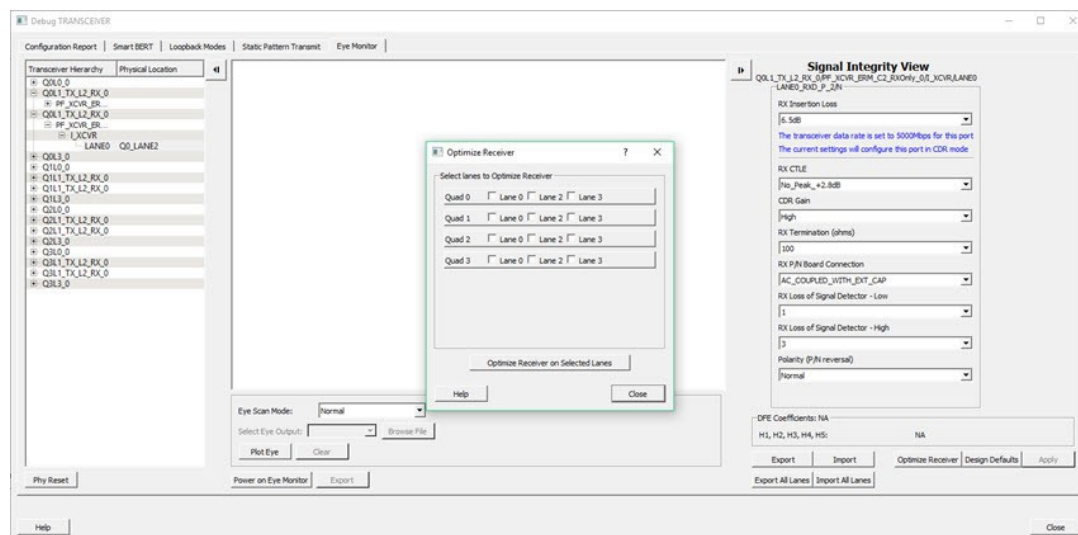


Figure 3-68. Debug TRANSCIEVER - Optimize Receiver



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

Figure 3-69. Optimize Receiver Dialog Box

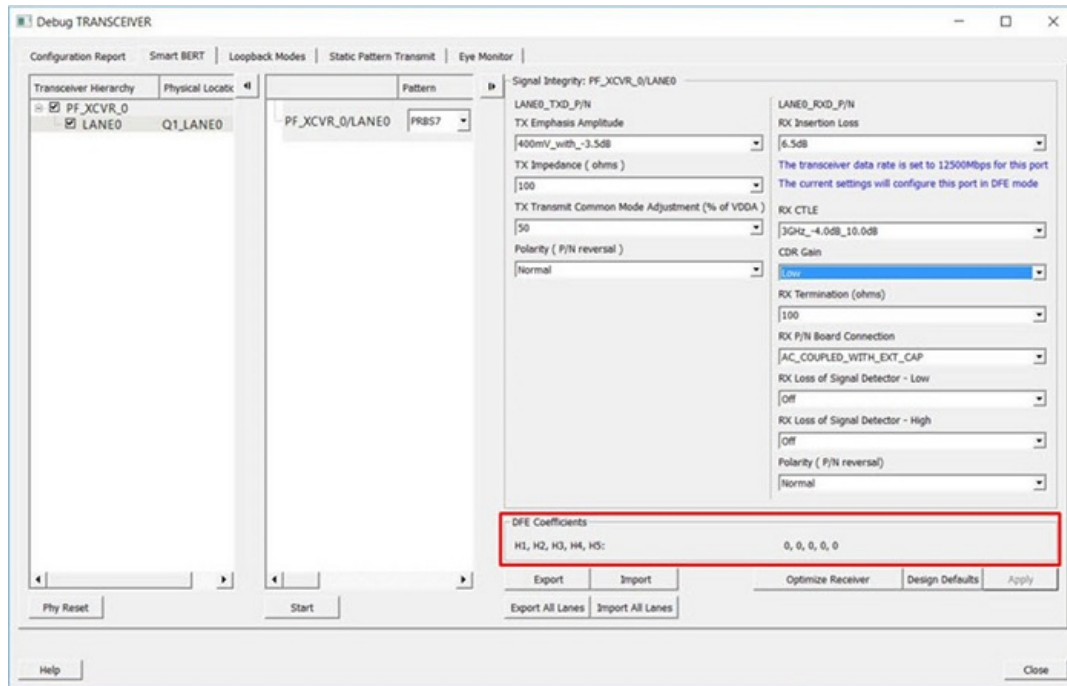


Select the lanes on which to run Optimize Receiver and click **Optimize Receiver on Selected Lanes**. You can select any combination of lanes including those configured in CDR or DFE. The hardware will perform CTLE calibration for CDR receivers and Full calibration for DFE receivers.

### 3.14.8.7 Display DFE Coefficient Values

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

Figure 3-70. Signal Integrity Pane - DFE Coefficients



- DFE coefficients for CDR lanes are displayed as NA.
- The DFE coefficients are read from the register fields as follows:

```
H1 = H1_MON
H2 = H2_MON
H3 = H3_MON
H4 = H4_MON
H5 = H5_MON
```

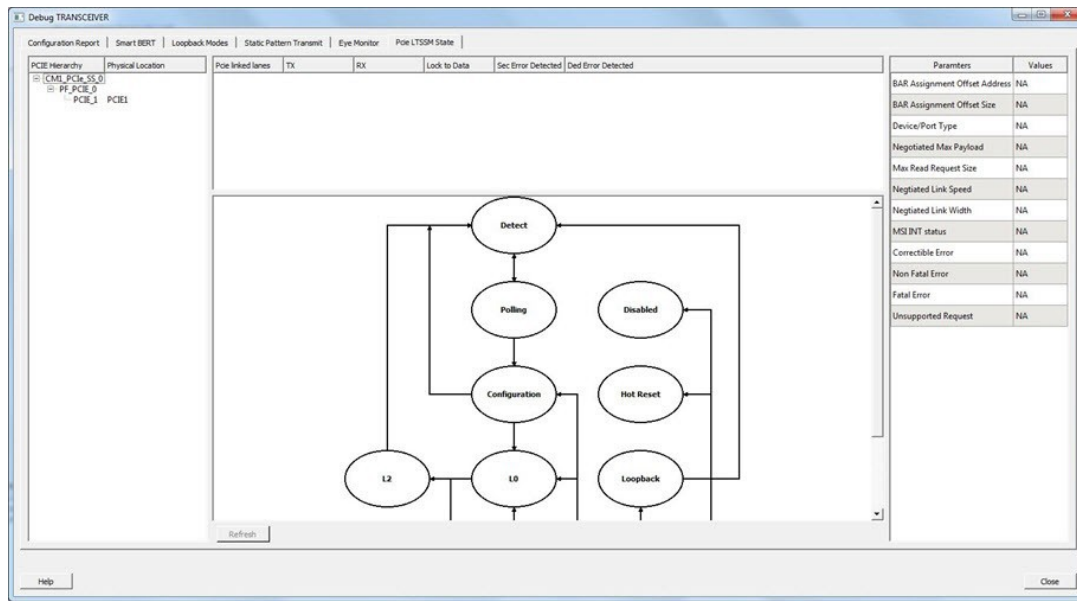
- DFE coefficients are read back from the device in the following scenarios:
  - When the lane is selected in the SmartBert, Loopback Modes, Static Pattern Transmit, and Eye Monitor pages.
  - When a test is started/stopped on selected lane in the SmartBert page.
  - When a test is started/stopped on selected lane in the Static Pattern Transmit page.
  - When Optimize Receiver is executed on the selected lane.

### 3.14.9 PCIE Debug

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

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

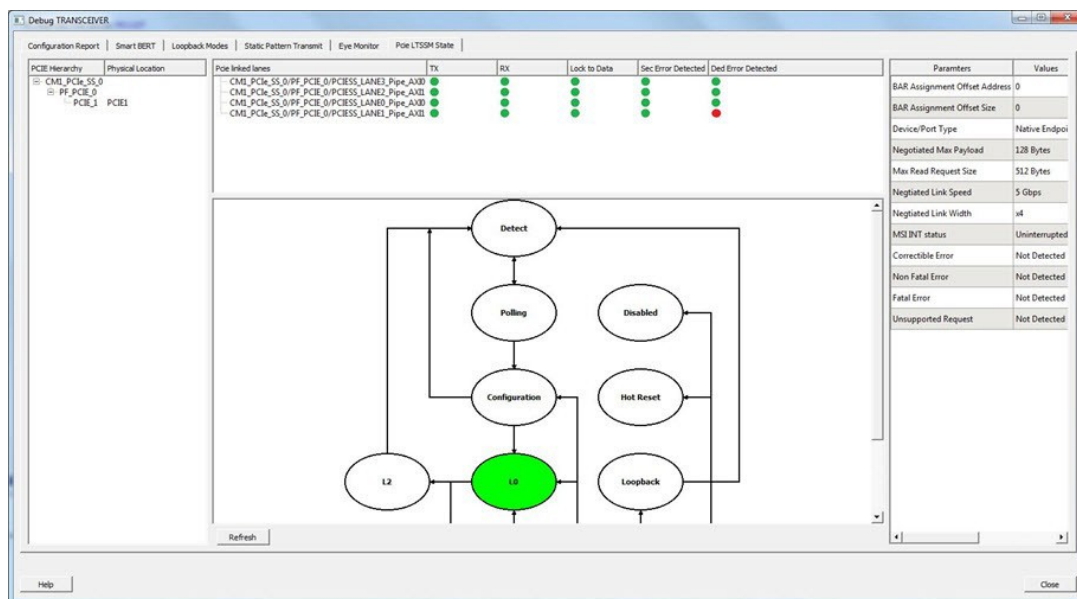
Figure 3-71. PCIe Design Hierarchy



### 3.14.9.1 Lane Status and Lane Link Error Status

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

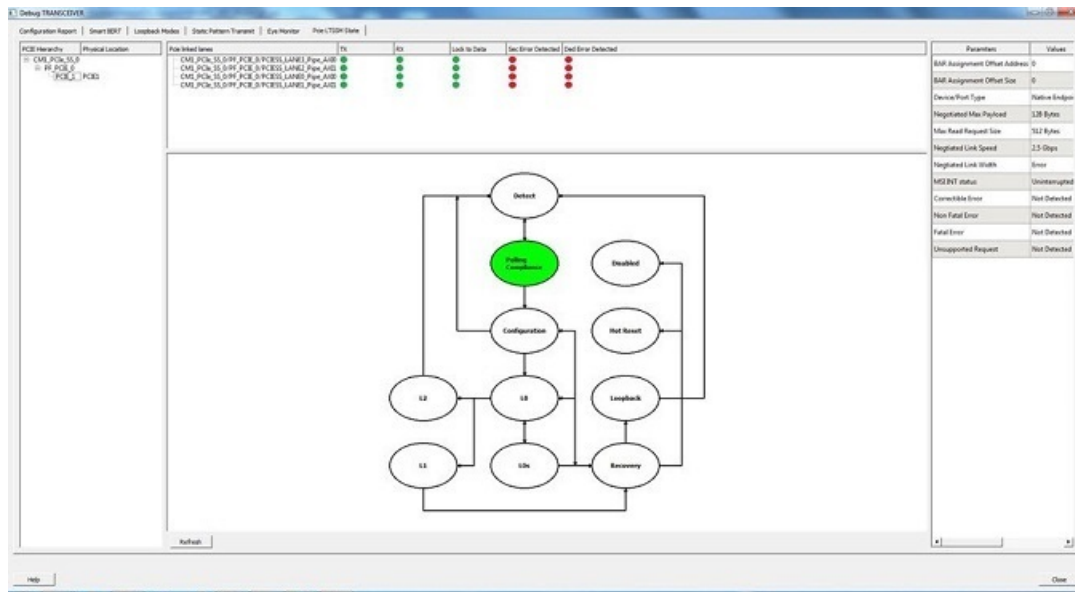
Figure 3-72. Lane Status and Lane Link Error Status



### 3.14.9.2 LTSSM State Machine

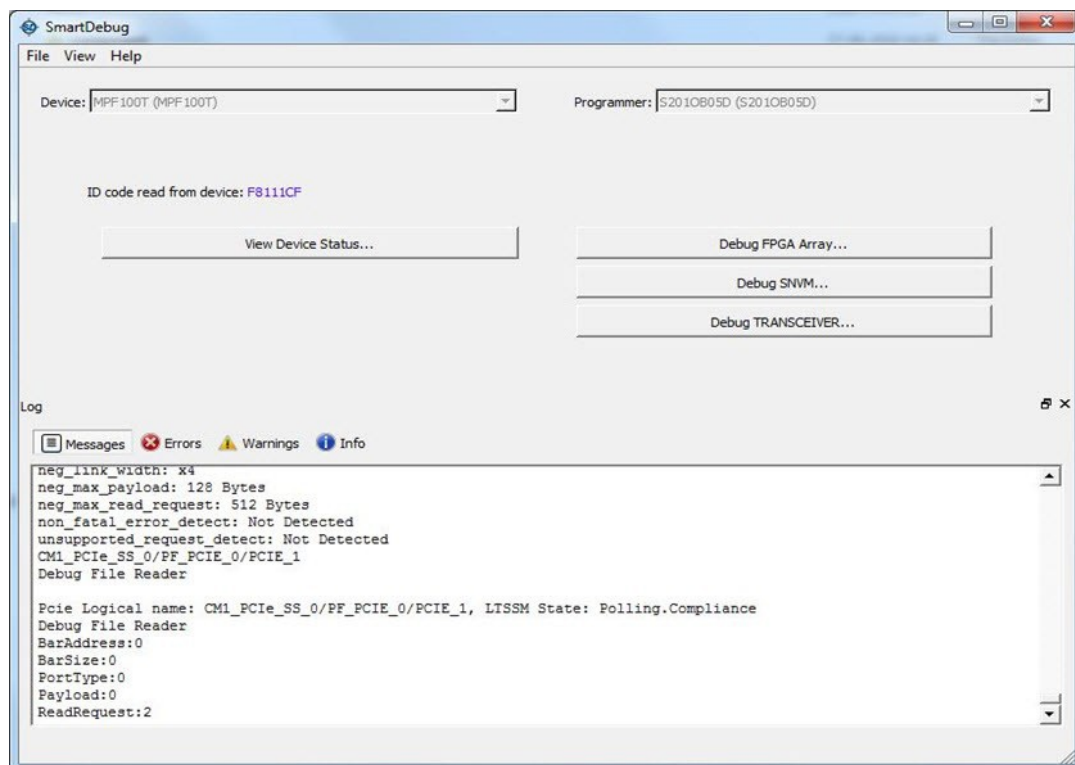
The PCIe LTSSM State page shows the LTSSM state machine to the right of PCIe Design Hierarchy and under the lane status, as shown in the following figure.

**Figure 3-73. LTSSM State Machine**



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

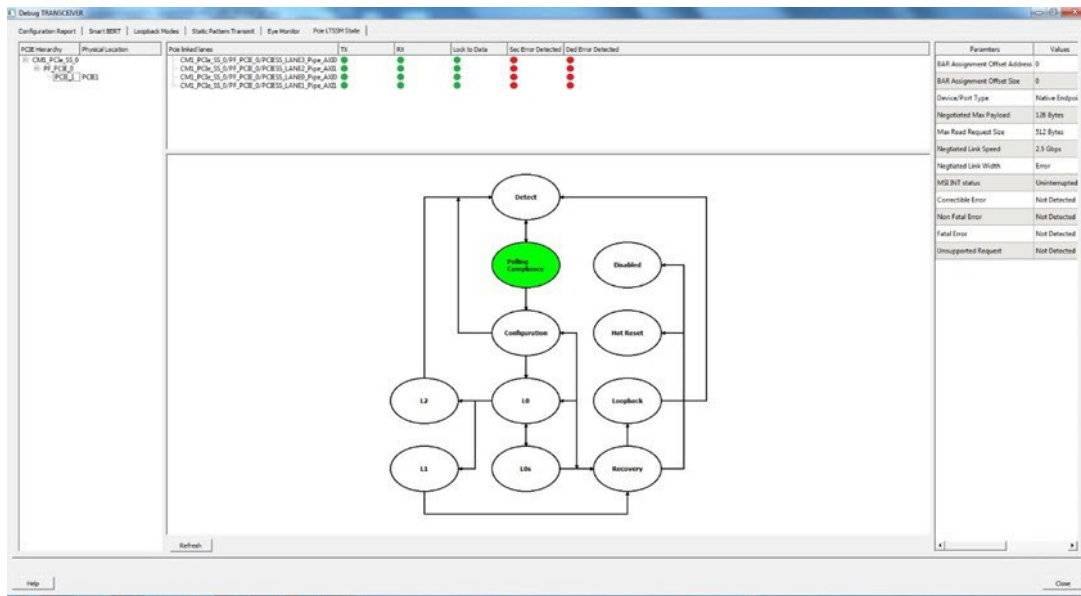
**Figure 3-74. Active State and Substate Information**



### 3.14.9.3 Config Space Parameter Information

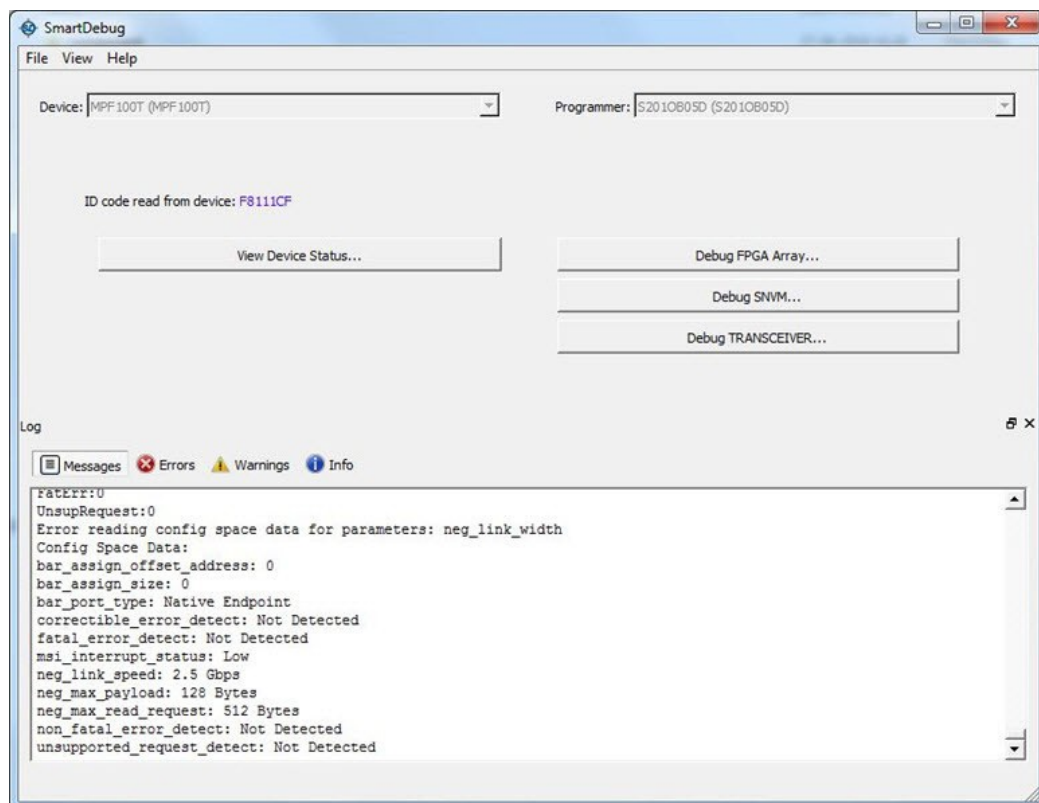
When you click on a PCIE instance in the PCIE Hierarchy, the config space parameter data is retrieved from the device and displayed to the right of the LTSSM state machine, as shown in the following example figure. When no PCIE instance is selected, or any level of hierarchy instance except for the PCIE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

**Figure 3-75. Config Space Parameter Information**



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

**Figure 3-76. Config Space Parameter Information in the SmartDebug Log Window**

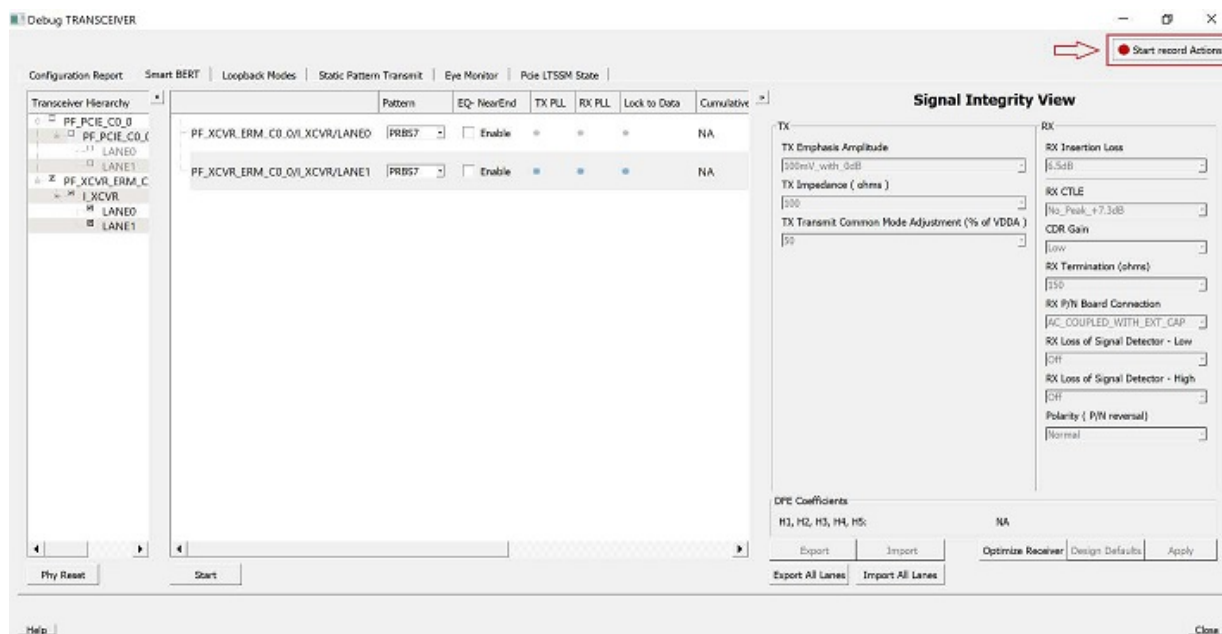


Click the **Refresh** button to update all PCIE data displayed in the GUI.

### 3.14.10 Record Actions

This option is used to record the register sequence of XCVR operations into a file. The Start record Actions option is shown at the top-right corner as an option to click to start recording.

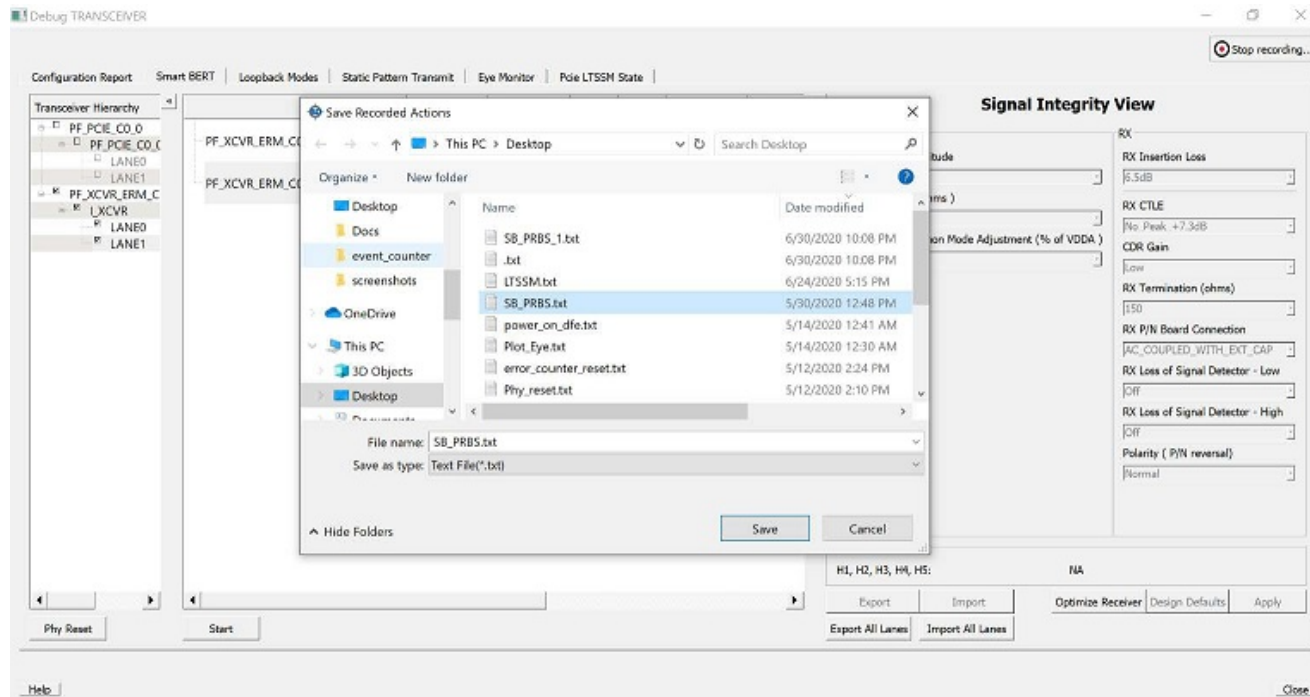
**Figure 3-77. Debug TRANSCEIVER Window Showing Start Record Actions Option**



This option is hidden in Demo Mode. When clicked on '**Start record Actions**', recording starts and the option changes to '**Stop recording...**'.

When clicked on '**Stop recording...**' to stop the recording, a window pop-up asks the user for the output to be saved to a text (.txt) file. After saving the file, the Debug TRANSCEIVER window goes to default state.

**Figure 3-78. Save Recorded Action Window Pop-Up to Save Text File After 'Stop recording...' is Clicked**



### 3.14.10.1 Recorded Content

The saved file is in plain text (.txt) format.

The first five lines contain the header. The header says that the following sequence is a read-modify-write operation. It also has the date and time of recording.



The operation sequence always starts with the name of the operation followed by the Quad and lane information if applicable.

The following snippet is an example demo file which records PRBS start sequence. Interpreting the text is as follows:

### Example 1

1. Start PRBS test on LANE2 QUAD0 using PRBS7 means:
  - 1.1. The lane selected on the SmartBERT tab is Q0\_ANE0 and the pattern selected is XCVR PMA PRBS7.
  - 1.2. The first set of lines has four subheadings which mean the following:
 

**Register Name:** Register name that can be looked up in the register map.

**Address:** The physical address of this register is 7 hexadecimal digits which is calculated based on the Quad, lane number, register type and offset.

**Write Mask:** Write mask is 8 hexadecimal digit number which indicates the mask value to be applied on the register read value.

**Write Value:** The write value is 8 hexadecimal digits number to be written after the mask is applied.

If the value is 0, the operation is basically read the register value and apply the mask on it and write in the register.
  - 1.3. End of start PRBS test sequence -> denotes that the sequence for that lane is completed and the next sequence is initiated.

Figure 3-79. Example 1

```
# ----- #
# Microchip Technology Inc. #
# The following sequence is a read-modify-write operation. #
# Tue Jun 30 22:08:59 2020 #
# ----- #

# -----
# Start PRBS test on LANE2 QUAD0 using PRBS7:
# -----
Register Name: SER_CTRL
Address:      0x1044070
Write Mask:   0xFFFFF7FF
Write Value:  0x0

Register Name: SERDES_RTL_CTRL
Address:      0x10440C0
Write Mask:   0xFFFFA1FF
Write Value:  0x2000

# End of start prbs test sequence.
```

The following are XCVR operations for which Record Actions is supported:

- SmartBERT tests
- Loopback tests
- Static Pattern tests
- Power ON/OFF Eye Monitor
- Transceiver PHY Reset
- Poll PCIe LTSSM State
- Read PCIe configuration space

The following are XCVR operations for which Record Actions is not supported:

- Eye Monitor



- Optimize Receiver
- Signal Integrity
- PRBS tests from SmartBERT IP
- Phy

### Example 2

LTSSM state on PCIe lane

Register Name and Address will be common to all the register sequence.

**Read Mask:** This means the sequence is only a read register operation. The mask value is the mask to extract the required field value. Here, the mask value is 0x1F which means reg[4:0] is the value to be read, hence the mask value.

**LTSSM STATE:** This is the result that is obtained after the mask is applied to the read value.

Figure 3-80. Example 2

```

1 # -----#
2 # Microchip Technology Inc. #
3 # The following sequence is a read-modify-write operation. #
4 # Wed Jun 24 17:15:48 2020 #
5 # -----#
6
7 # -----#
8 # LTSSM Status on PCIE1: #
9 # -----#
10 Register Name: LTSSM_STATE
11 Address: 0x300A05C
12 Read Mask: 0x1F
13 LTSSM STATE: 0x3
14
15 # End of LTSSM state operation.
16

```

## 3.15 MSS Register Access

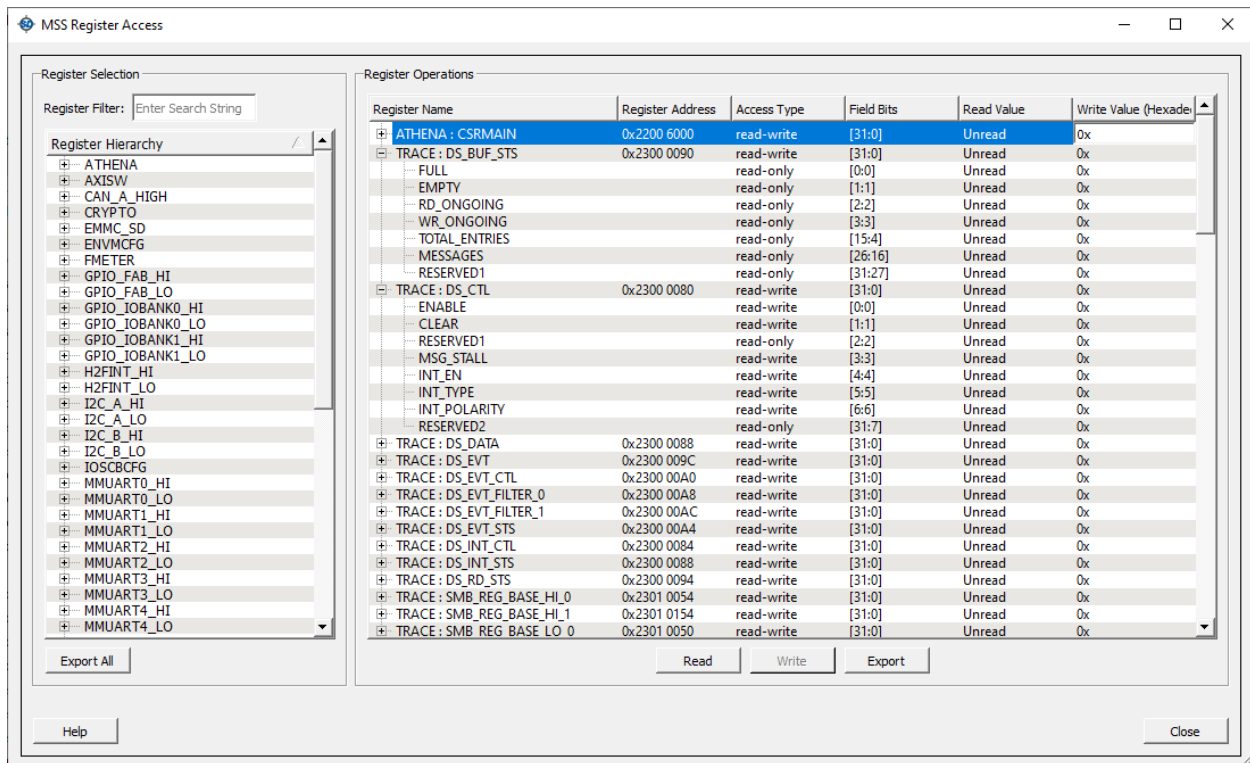
SmartDebug provides the **MSS Register Access** option to read and write to the device and export register read details to a .csv file. Click **MSS Register Access** on the main SmartDebug window to access this feature.

**Note:** The **MSS Register Access** button is visible and available for use only, if the MSS component is configured and used in the Libero design.

The MSS Register Access window comprises the **Register Selection** and **Register Operations** panes.

**Note:** Memory Protection Unit (MPU) in MSS component must be enabled or configured before accessing the MSS registers.

**Figure 3-81. MSS Register Access Window**



**Note:** Selected registers are retained even if you close and re-open the SmartDebug project in the standalone SmartDebug tool.

The following table describes the various options available on the **MSS Register Access** window.

Option	Description
<b>Register Selection Pane</b>	<p>Lists all the instances of the MSS block as described in the MSS register map.</p> <p>Peripherals are shown based on the configuration from the MSS component. Apart from peripherals, the following six instances of registers that are always shown in the pane as they are the configuration registers and are always accessible.</p> <ul style="list-style-type: none"> <li>AXISW</li> <li>MPUCFG</li> <li>ENVMCFG</li> <li>IOSCBCFG</li> <li>PFSOC_MSS_TOP_SCB_REGS</li> <li>PFSOC_MSS_TOP_SYSREG</li> </ul>
<b>Register Filter</b>	<p>Enables you search for instances/register by their names. Supports wildcard (*) searches.</p> <ul style="list-style-type: none"> <li>The search is real time.</li> <li>Enter asterisk ( *) to display all registers having instance names expanded.</li> <li>Names entered without asterisk will look for exact match.</li> </ul>
<b>Register Hierarchy</b>	<p>Displays Instance name and Register name in a two-level hierarchy-level format.</p> <ul style="list-style-type: none"> <li>You can select multiple names.</li> <li>Right-click and select 'Add' to add the registers to the operation pane.</li> <li>Drag and drop the register names to the Register Operation pane.</li> </ul>

.....continued

Option	Description
<b>Export All</b>	Exports information of all registers that are displayed in the selection pane to the specified .csv file. <ul style="list-style-type: none"> <li>The exported .csv file contains six sections: <ul style="list-style-type: none"> <li>Register Name</li> <li>Register Address (hexadecimal number)</li> <li>Register Field Name</li> <li>Register Field Range</li> <li>Register Field Value (hexadecimal number)</li> <li>Register Value (hexadecimal number)</li> <li>Comments (message to show if read register was successful or not)</li> </ul> </li> </ul>
<b>Register Operations</b>	Lists the registers selected for access operations such as read, write, and export. <p><b>Note:</b> Selected registers are retained even if you close and re-open the SmartDebug project in the standalone SmartDebug tool.</p> <ul style="list-style-type: none"> <li>The <b>Register Operations</b> pane has six columns - Name, address, access type, field bits, read value, and write value.</li> <li>The register has a sub-tree where all the register fields are listed.</li> <li>Select <b>Hide Register</b> from the context menu to remove the selected register. <b>Note:</b> You cannot remove a field inside the register.</li> <li>Write value column is populated with 0x to indicate that it supports hexadecimal values. <ul style="list-style-type: none"> <li>Click on a specific register row under write column to edit the value.</li> <li>The text entered is validated based on the range (field bits) of a field or a register. <b>Note:</b> Non-hexadecimal character will not be inserted.</li> </ul> </li> </ul>
<b>Read</b>	Read all selected registers by their register names and display the values in the hexadecimal format.
<b>Write</b>	Each cell in the table of selected registers can be accessed individually to write the values in hexadecimal format. <ul style="list-style-type: none"> <li>If there is a conflict in the values where full register write values and also field values of the same register, then a full register write is executed and field write will be ignored.</li> <li>Write to specific registers such as MPUCFG:PMPCFG_SCB_* will result in undesired SmartDebug tool behavior because these registers are responsible for configuring MPU block in MSS.</li> <li>You cannot edit the write field if the register has read-only access type.</li> <li>Write onto a field is a read-modify-write operation.</li> </ul>
<b>Export</b>	Reads the values of selected registers and saves them in the specified .csv file. The exported .csv file contains six sections <ul style="list-style-type: none"> <li>Register Name</li> <li>Register Address ( hexadecimal number )</li> <li>Register Field Name</li> <li>Register Field Range</li> <li>Register Field Value ( hexadecimal number )</li> <li>Register Value ( hexadecimal number )</li> </ul>

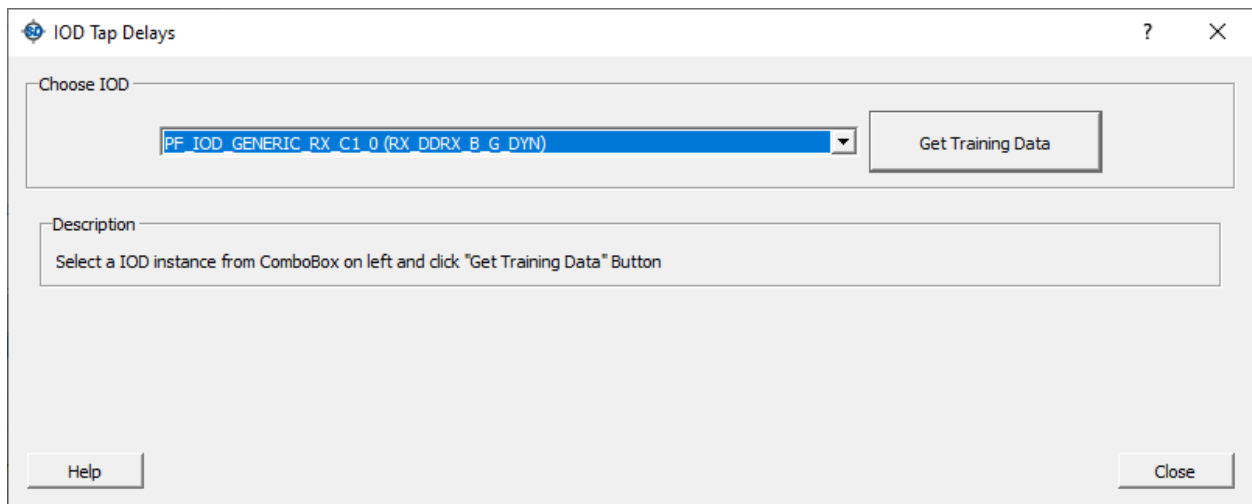
### 3.16 Debug IOD

SmartDebug provides the IOD Tap Delays feature for designs where the PF\_IOD\_GENERIC\_RX IP instance is configured as Dynamic or Fractional Dynamic mode in the design. Click the **Debug IOD** button on the main SmartDebug window to access this feature.

**Note:** The **Debug IOD** button is not available for designs where the IP in the modes mentioned is not used. The feature is supported by designs for PolarFire and PolarFire SoC family devices.

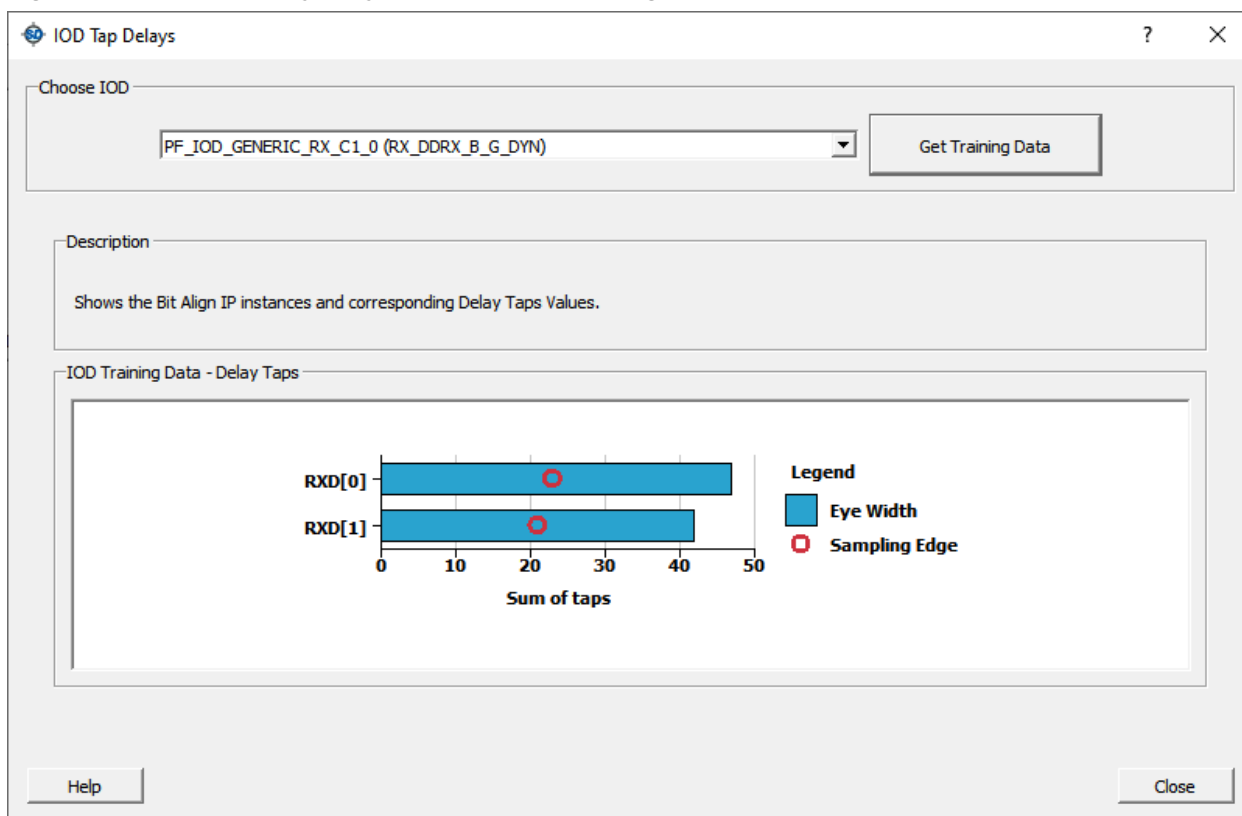
The **IOD Tap Delays** window lists the PF\_IOD\_GENERIC\_RX IP instances available in the design in the **Choose IOD** drop-down list. Select an IOD instance and click the **Get Training Data** button to view the Bit Align IP instances and corresponding Delay Tap values.

**Figure 3-82. IOD Tap Delays Window**



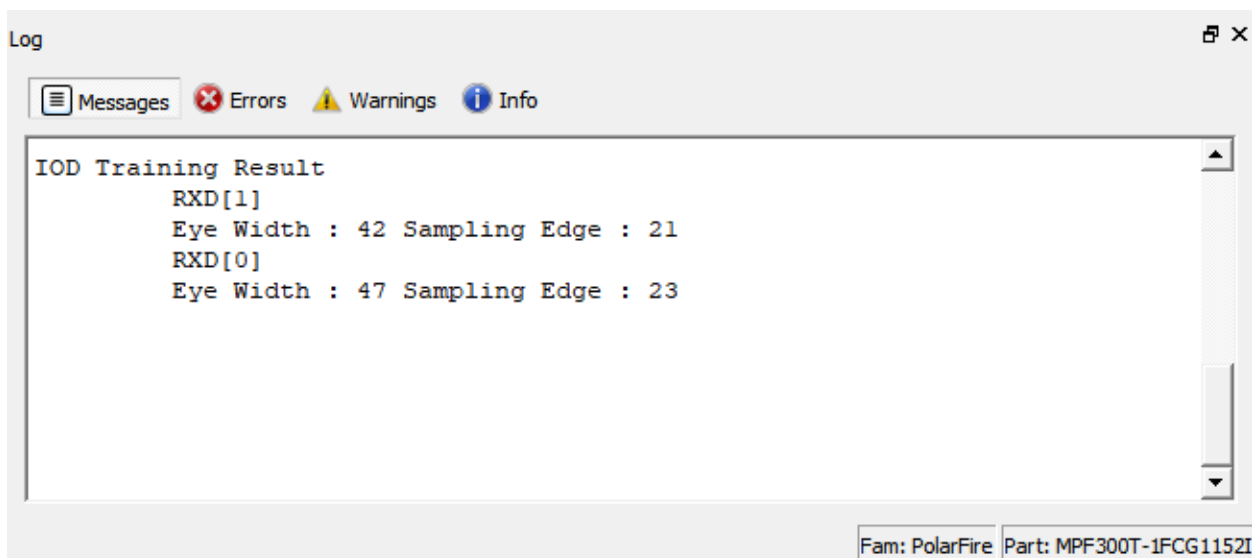
The following figure shows a design example with two instances of CORERXIODBITALIGN IP and the tap delay values read from those instances.

Figure 3-83. IOD Tap Delays - Eye Width and Sample Edge Data Representation



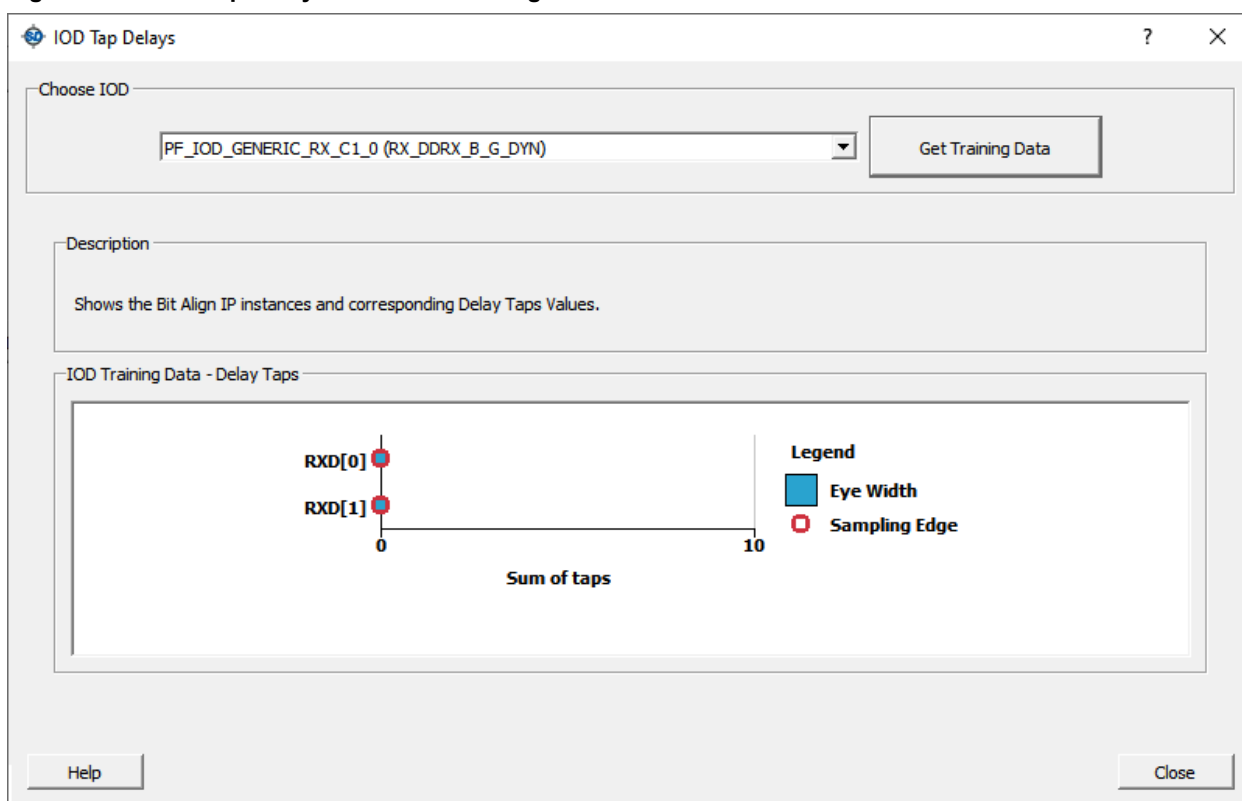
The following figure shows the SmartDebug **Log** window when the training is successful.

Figure 3-84. SmartDebug Log Window - Training Successful



The following figure shows the **IOD Tap Delays** window when the training fails.

Figure 3-85. IOD Tap Delays Window - Training Failed



The following figure shows the SmartDebug **Log** window when the training fails.

Figure 3-86. SmartDebug Log Window - Training Failed

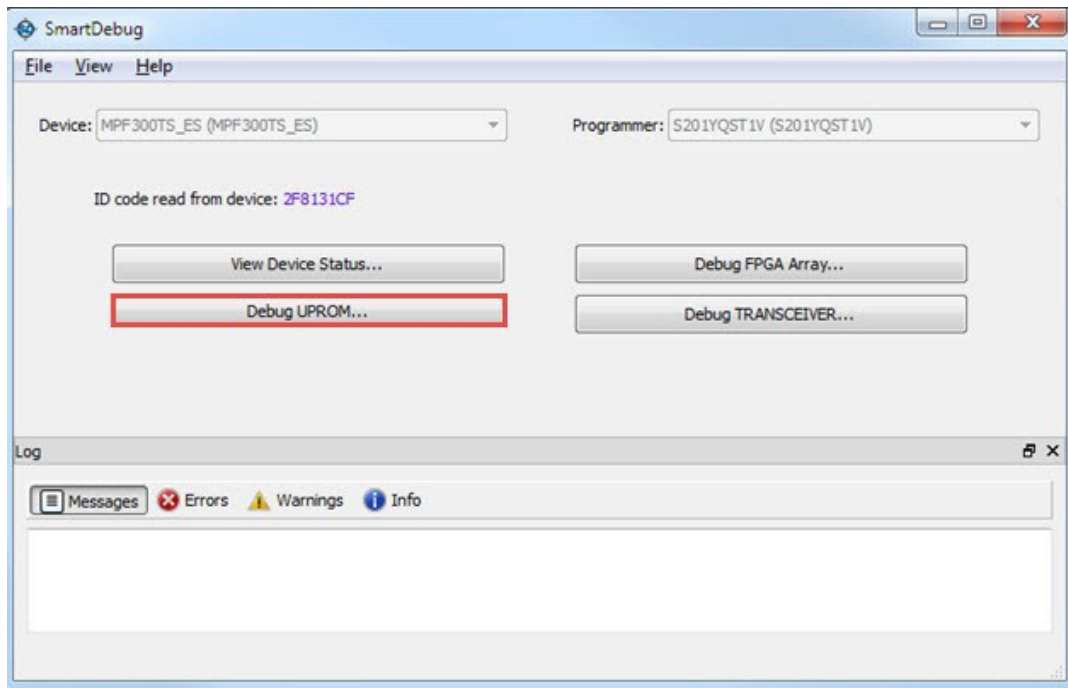


### 3.17 Debug UPROM

You can debug clients configured in a design and debug  $\mu$ PROM memory address information with the Debug UPROM feature.

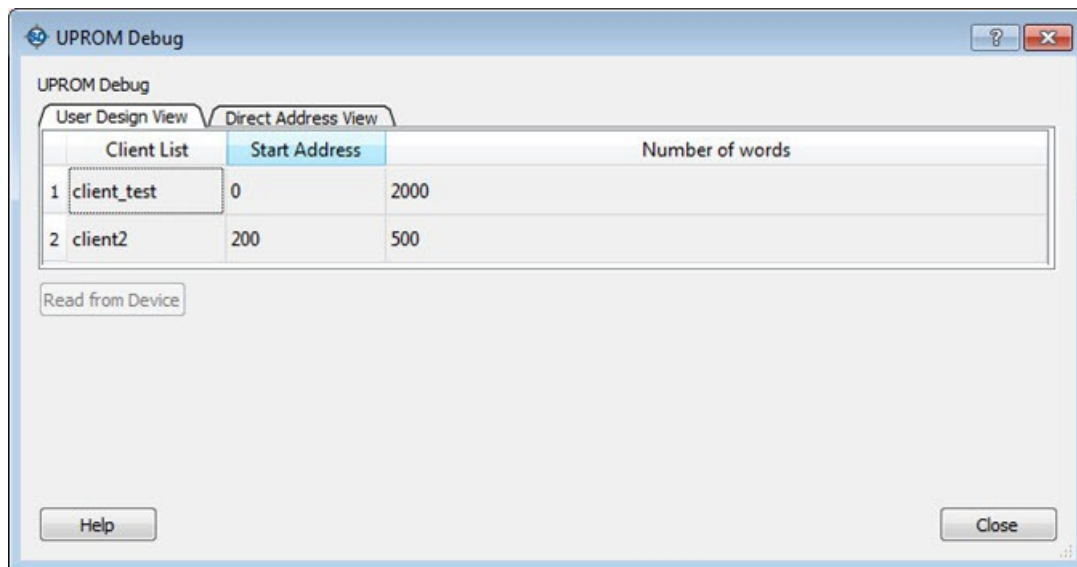
In the main SmartDebug window, click **Debug UPROM**.

**Figure 3-87. SmartDebug Window - Debug UPROM**



If a  $\mu$ PROM memory block is used in the Libero design, the **UPROM Debug** window appears.

**Figure 3-88. UPROM Debug Window**

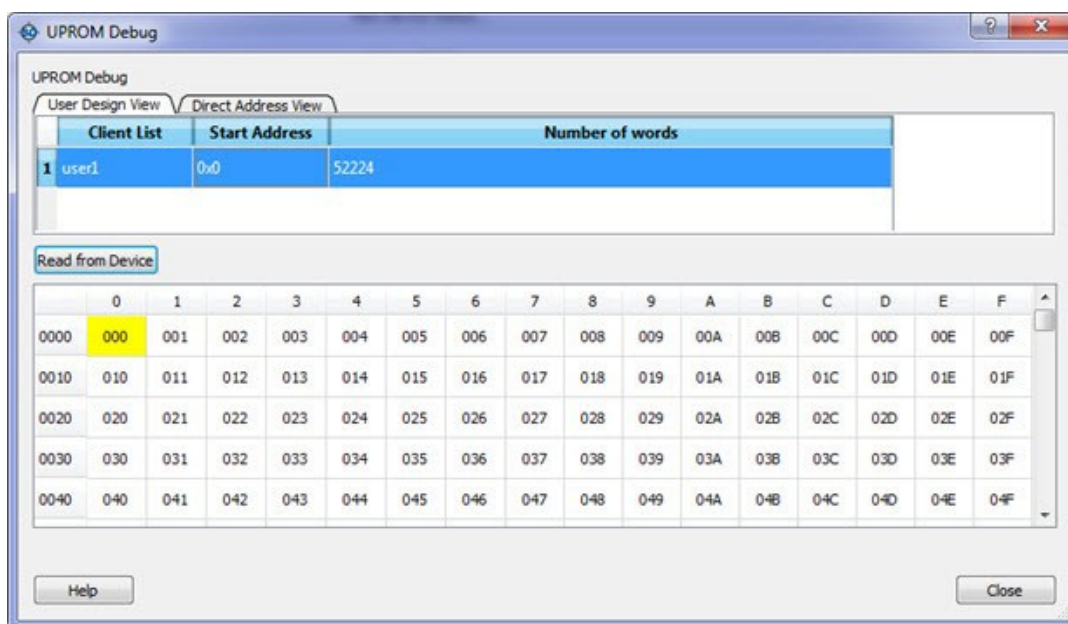


### 3.17.1 User Design View

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

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

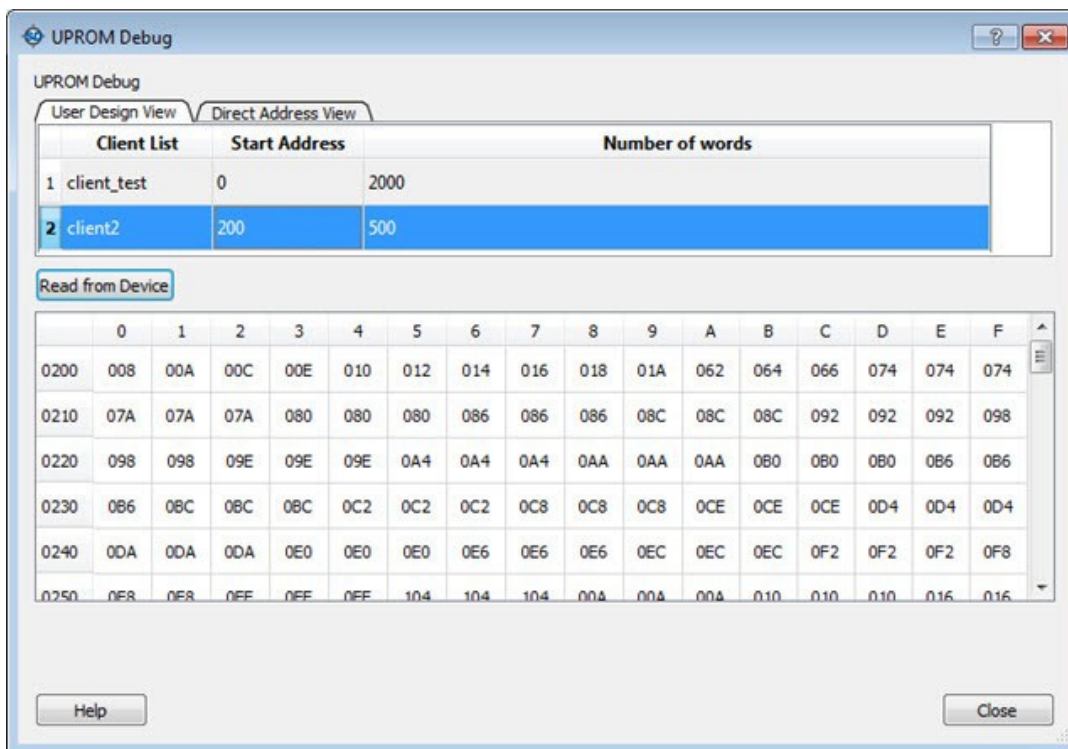




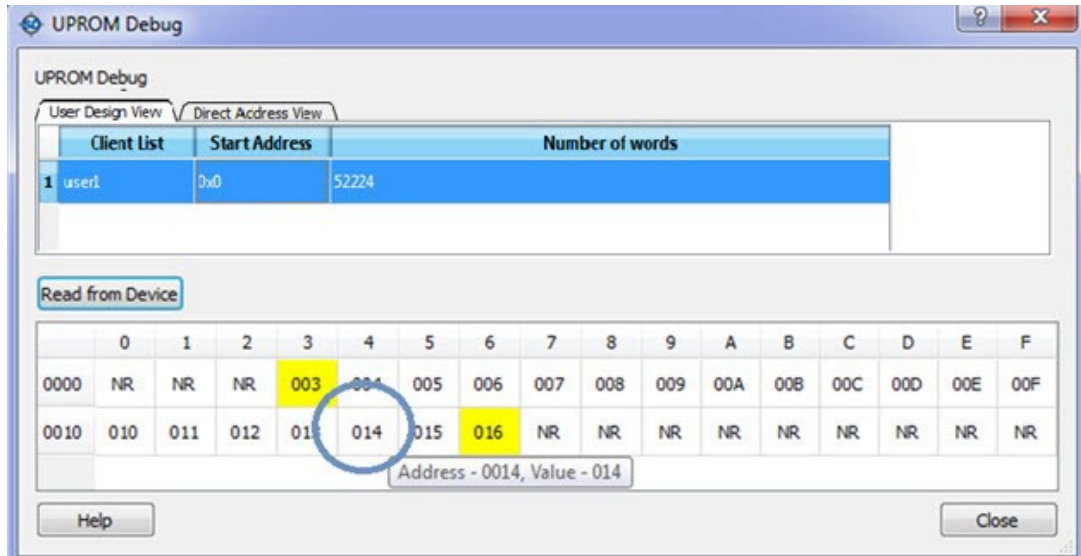
The Client address is associated with *Start Address* and *Number of 9-bit words*. Therefore, the table will contain as many locations as the number of 9-bit words.

In the example above, *Number of 9-bit words* is 52224, so 52224 words will be shown in the table. Column headers are numbered 0 to F in hexadecimal format, representing 16 words in a row.

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



Hover over a cell to see its address and value, as shown in the following figure.



### 3.17.2 Direct Address View

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

**Start Address:** Hexadecimal value (0 -9, A-F, upper/lower case)

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

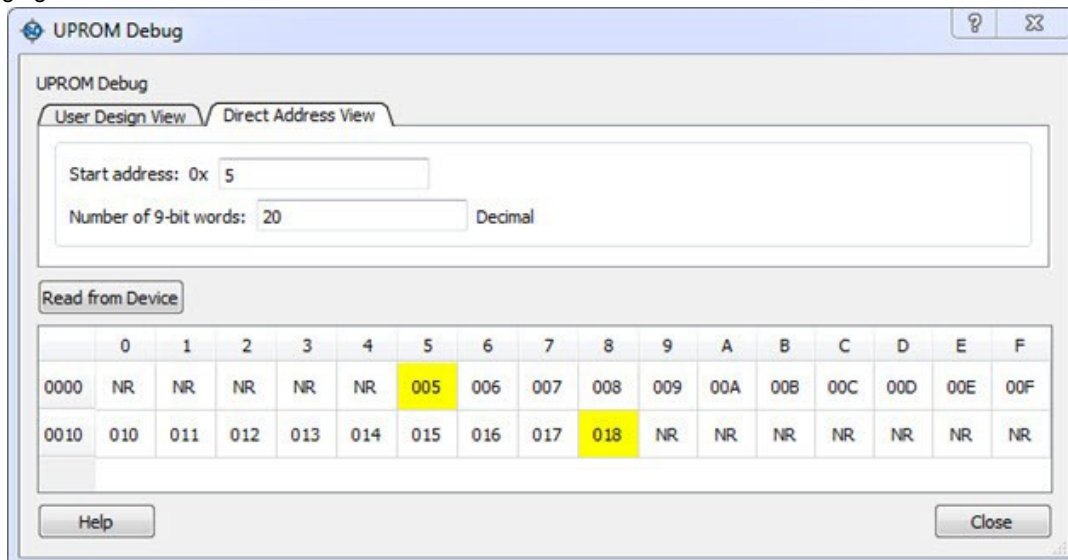
**Number of 9-bit words:** Positive integer value

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

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

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

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



### Notes

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

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

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

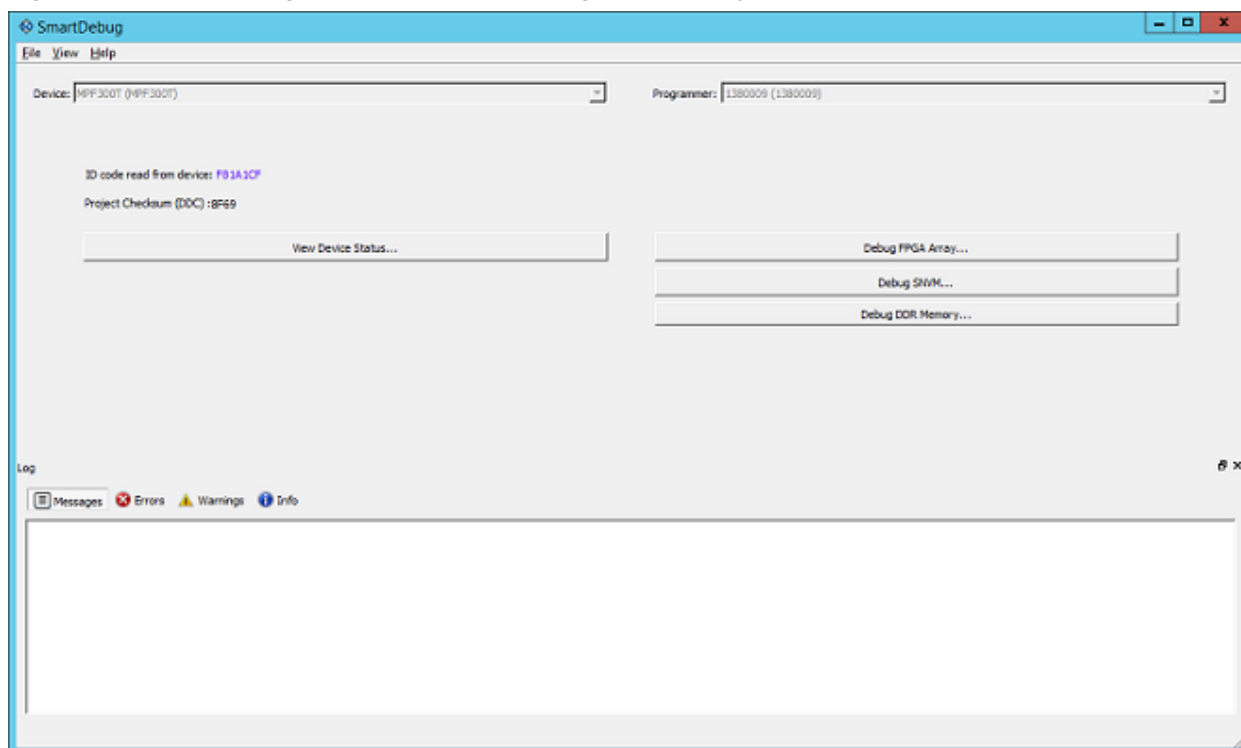
### 3.17.3 Demo Mode

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

### 3.18 Debug DDR IO Margin

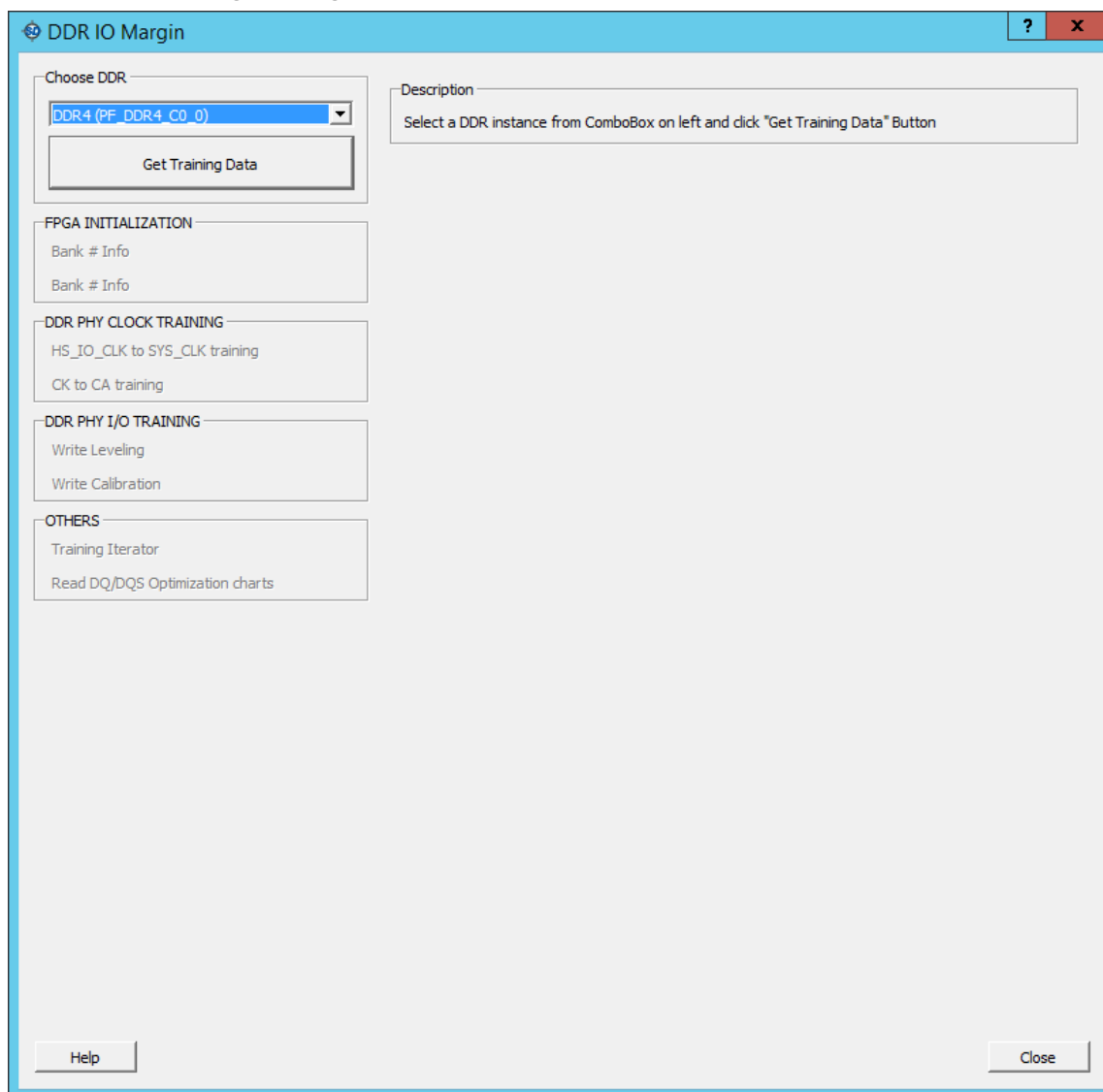
To access the Debug DDR IO Margin feature in SmartDebug, click **Debug DDR Memory...** in the main SmartDebug window. This option is available only for DDR3/DDR4/LPDDR3 memory configurations on PolarFire and PolarFire SoC devices. This option is not visible when DDR memory is not used in the design.

Figure 3-89. SmartDebug Main Window with Debug DDR Memory Option



This opens the Debug DDR IO Margin dialog box as shown in the following figure.

Figure 3-90. DDR IO Margin Dialog Box with Options Disabled



Initially, all options in the DDR IO Margin GUI are disabled. Users have to select which DDR instances training data they want to look at and click on the **Get Training Data** button. After clicking the **Get Training Data** button, the script to get the training data is run. It takes around two minutes and the DDR IO Margin dialog box displays the information based on the selected DDR Instance, as shown in the following figure.

Figure 3-91. DDR IO Margin Dialog Box Showing Information for First IO Bank

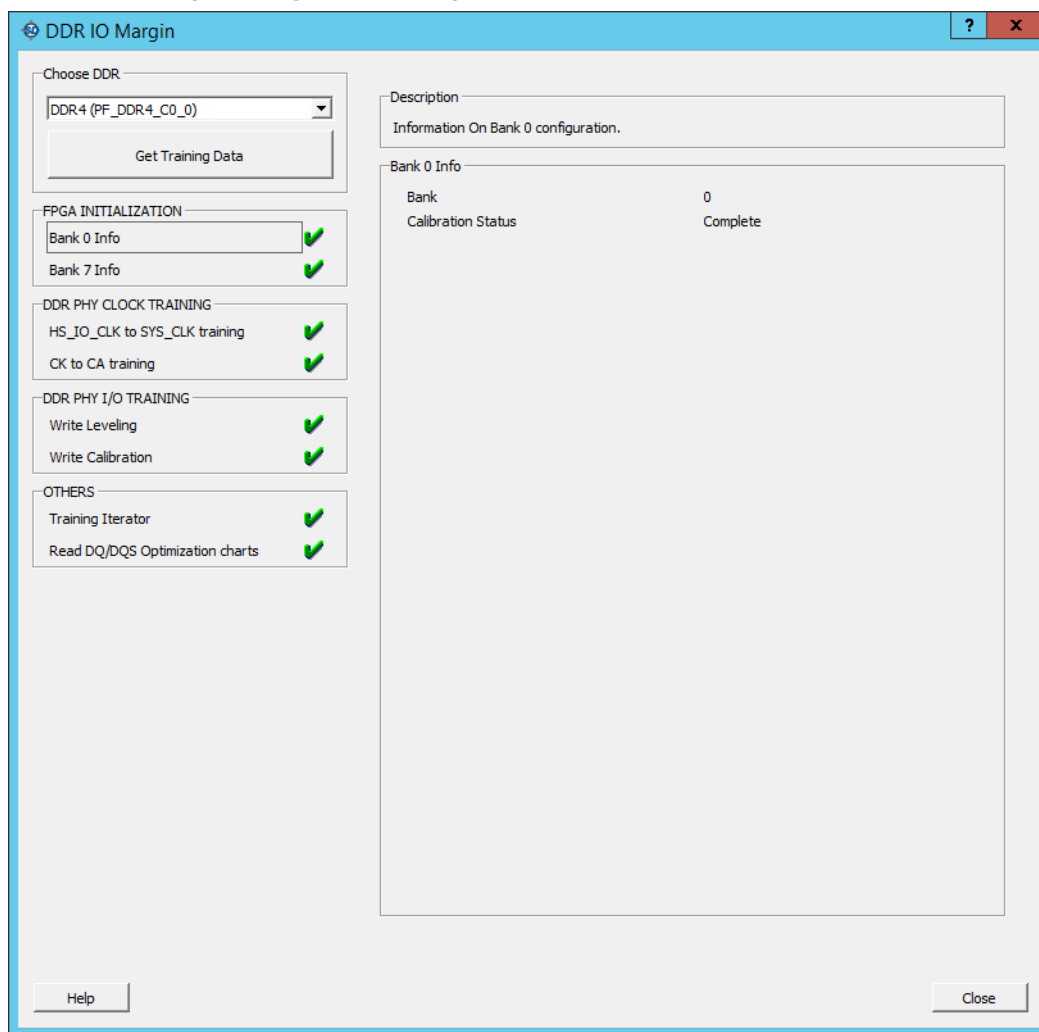


Figure 3-92. DDR IO Margin Dialog Box Showing Information for Second IO Bank

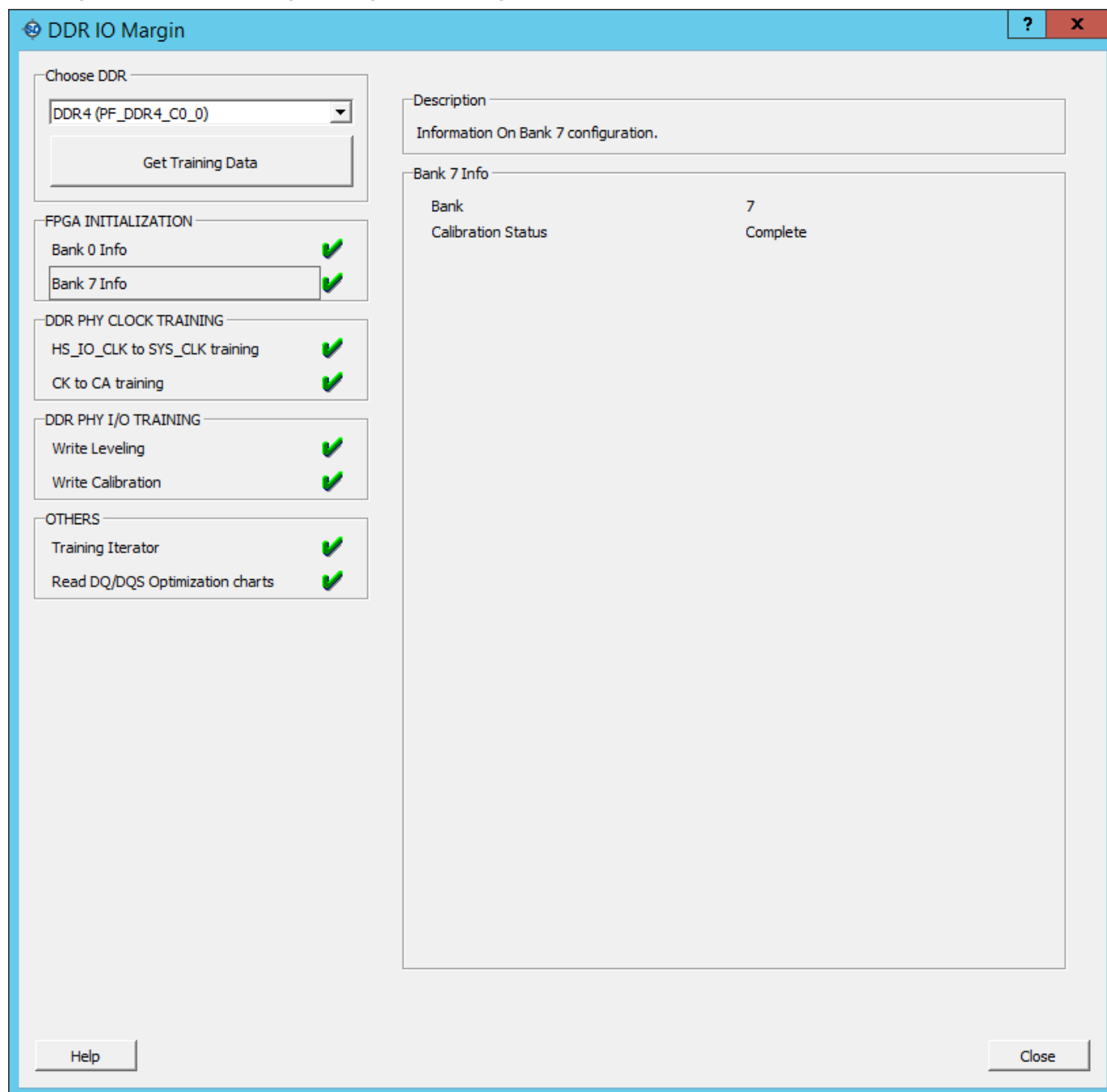




Figure 3-93. DDR IO Margin Dialog Box Showing HK\_IO\_CLK to SYS\_CLK Training

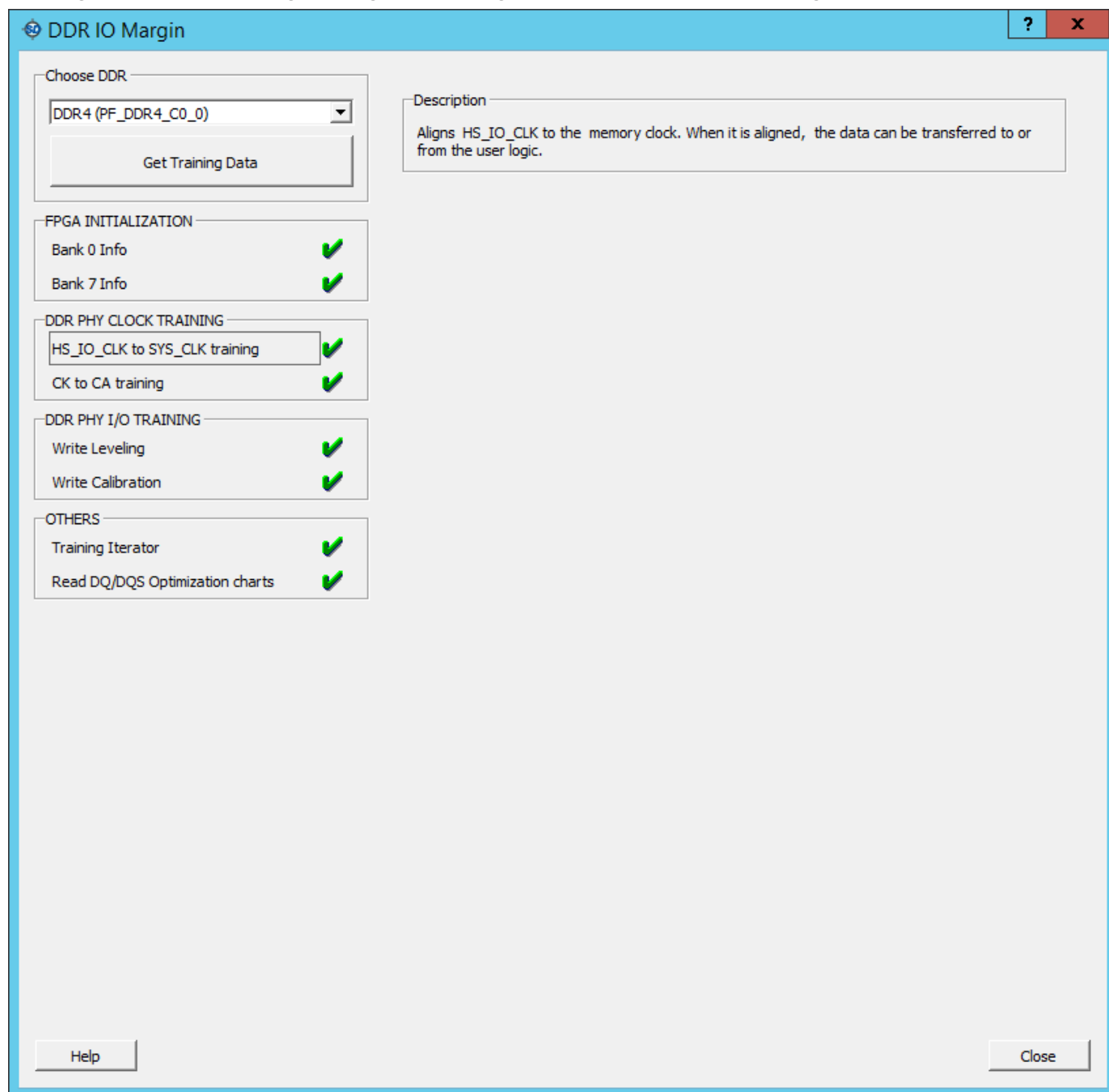


Figure 3-94. DDR IO Margin Dialog Box Showing CK to CA Training

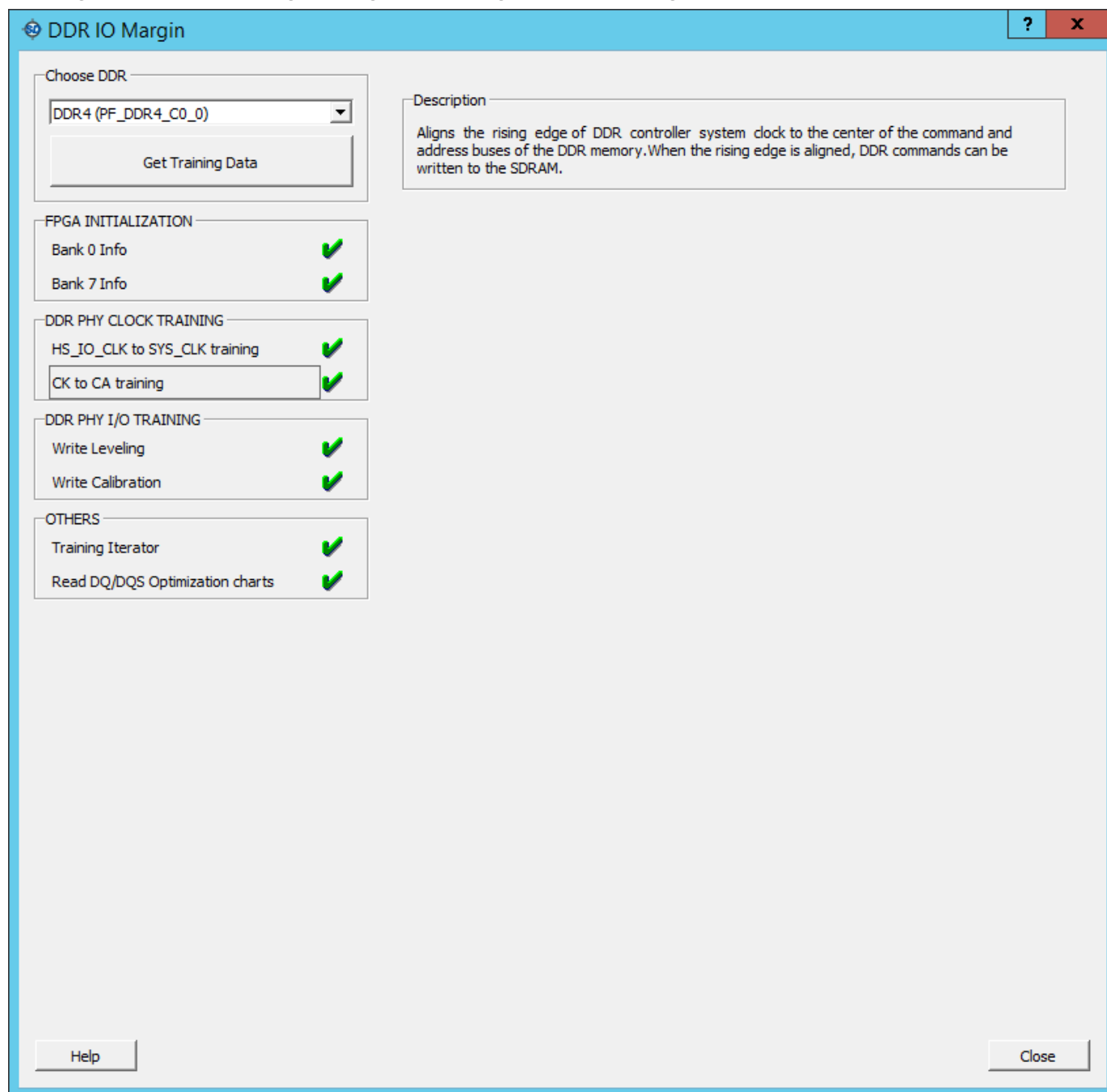


Figure 3-95. DDR IO Margin Dialog Box Showing Write Levelling Data

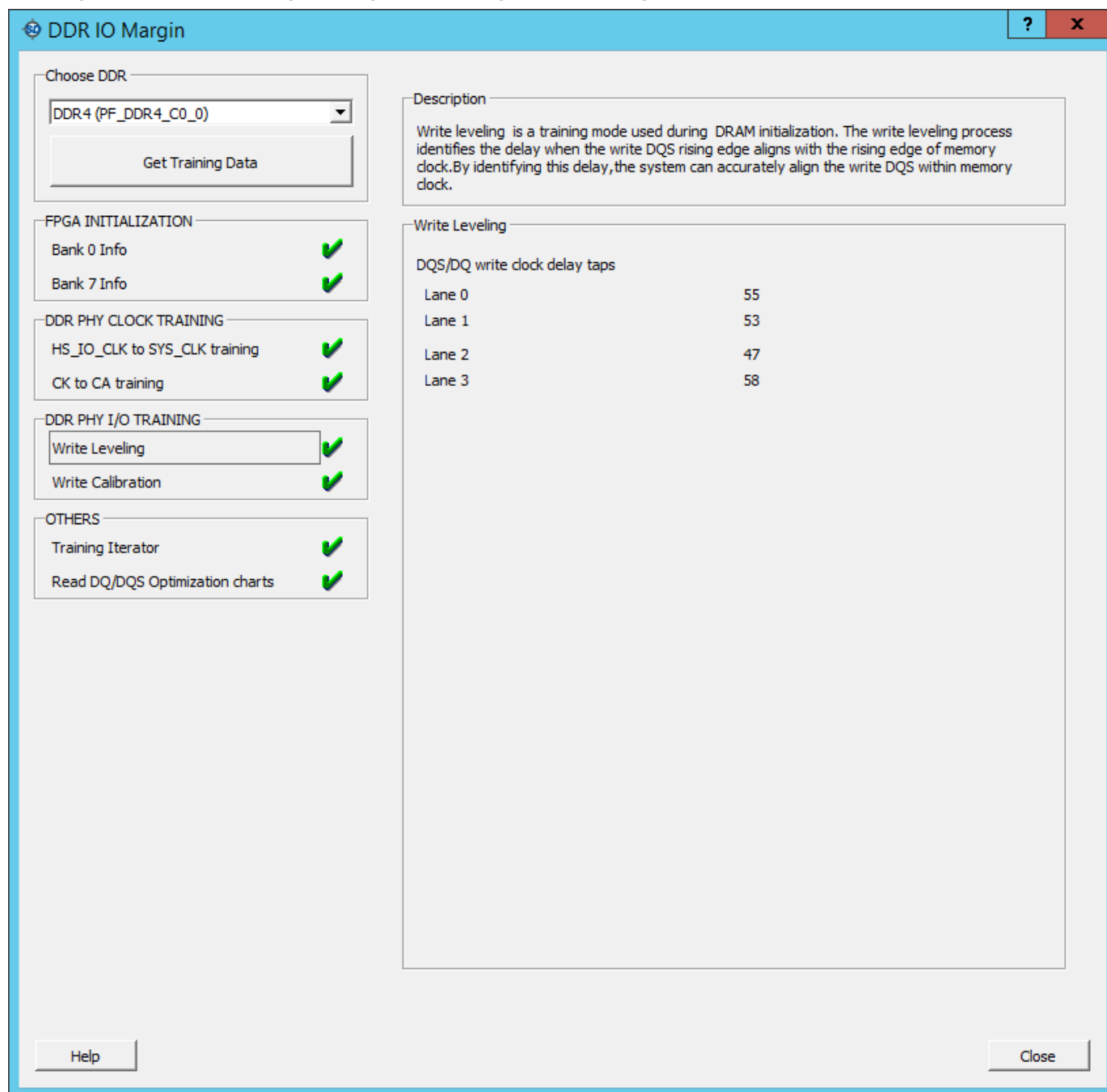


Figure 3-96. DDR IO Margin Dialog Box Showing Write Calibration Data

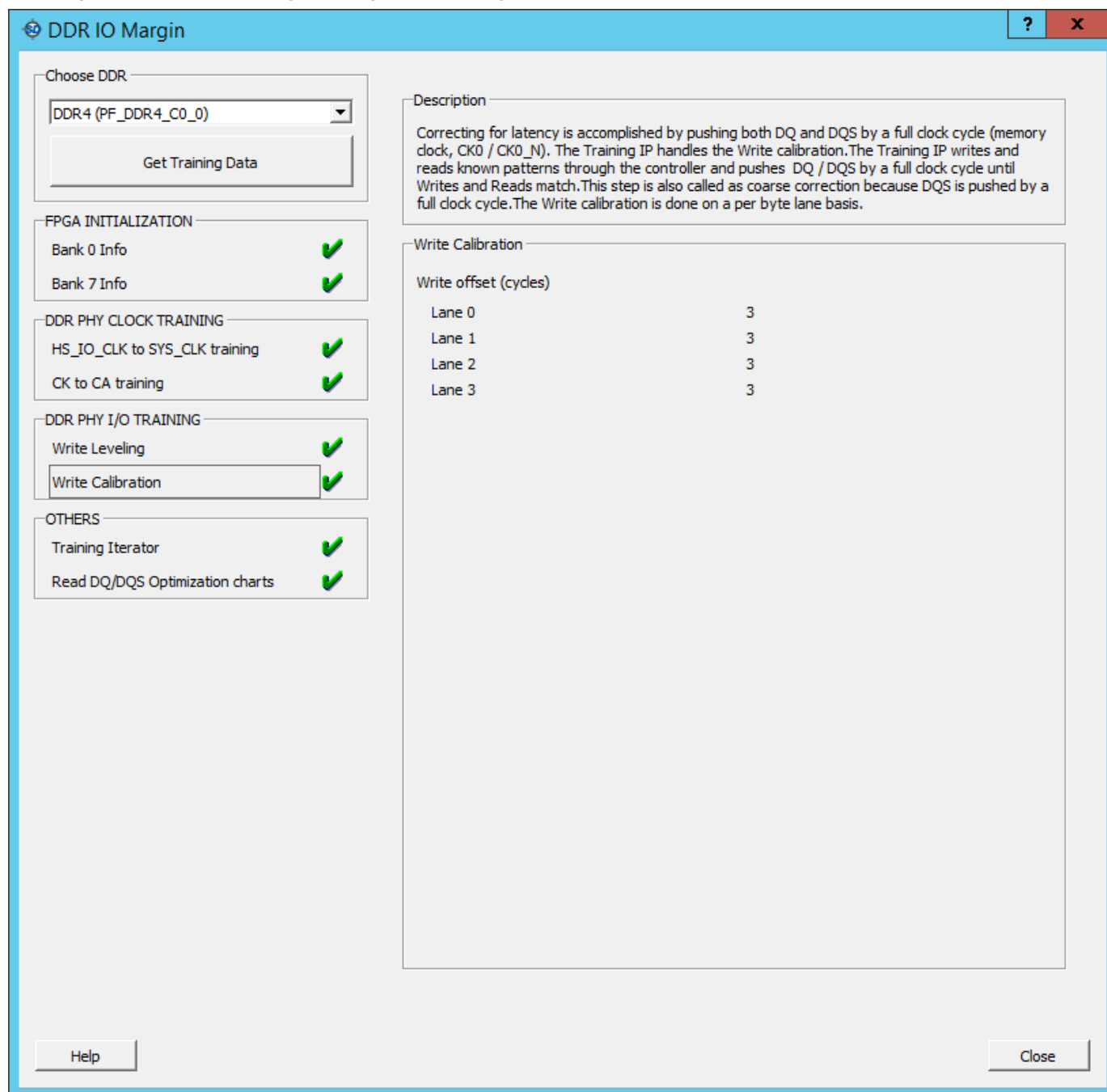


Figure 3-97. DDR IO Margin Dialog Box Showing Training Iterator Data

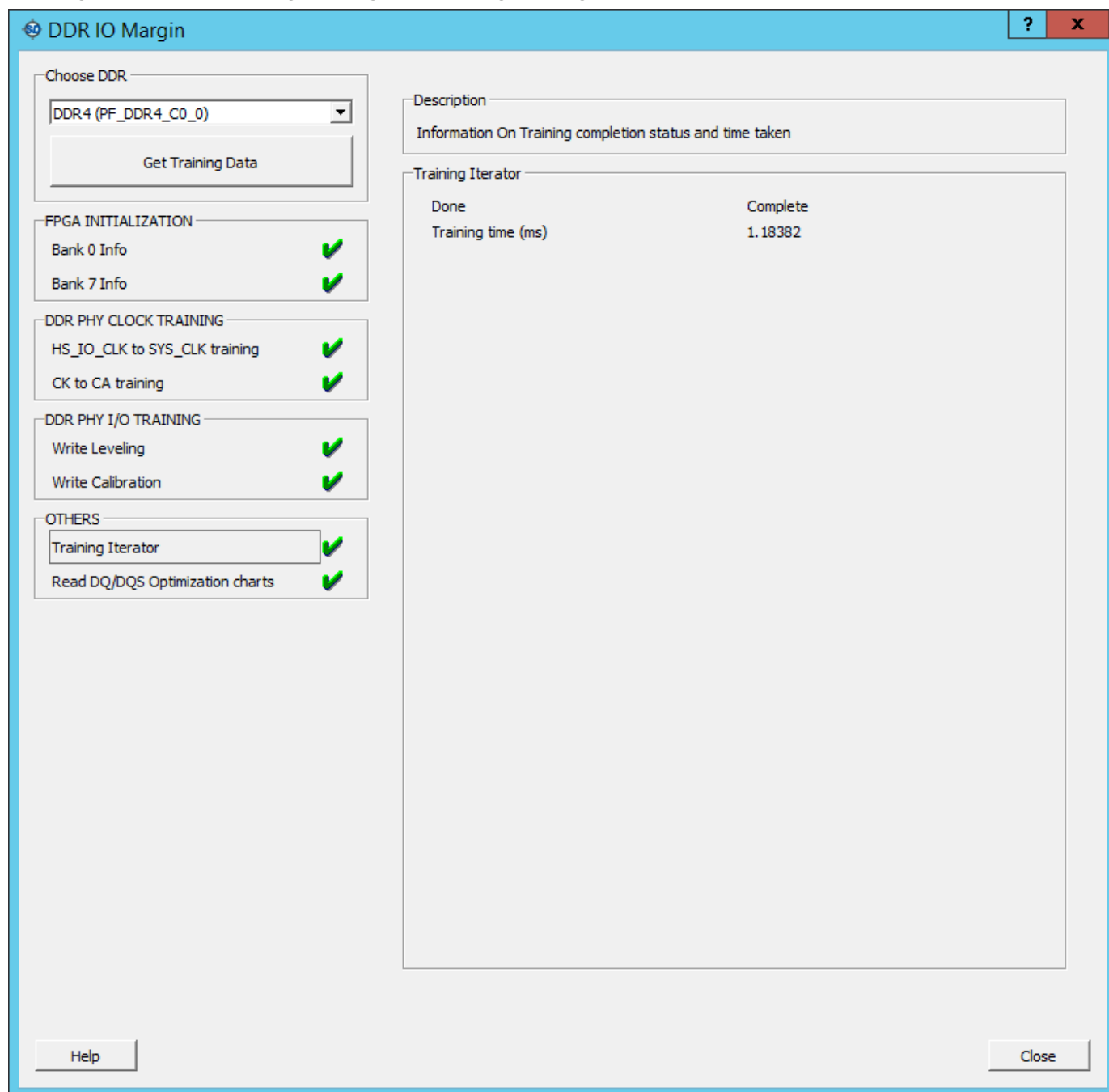
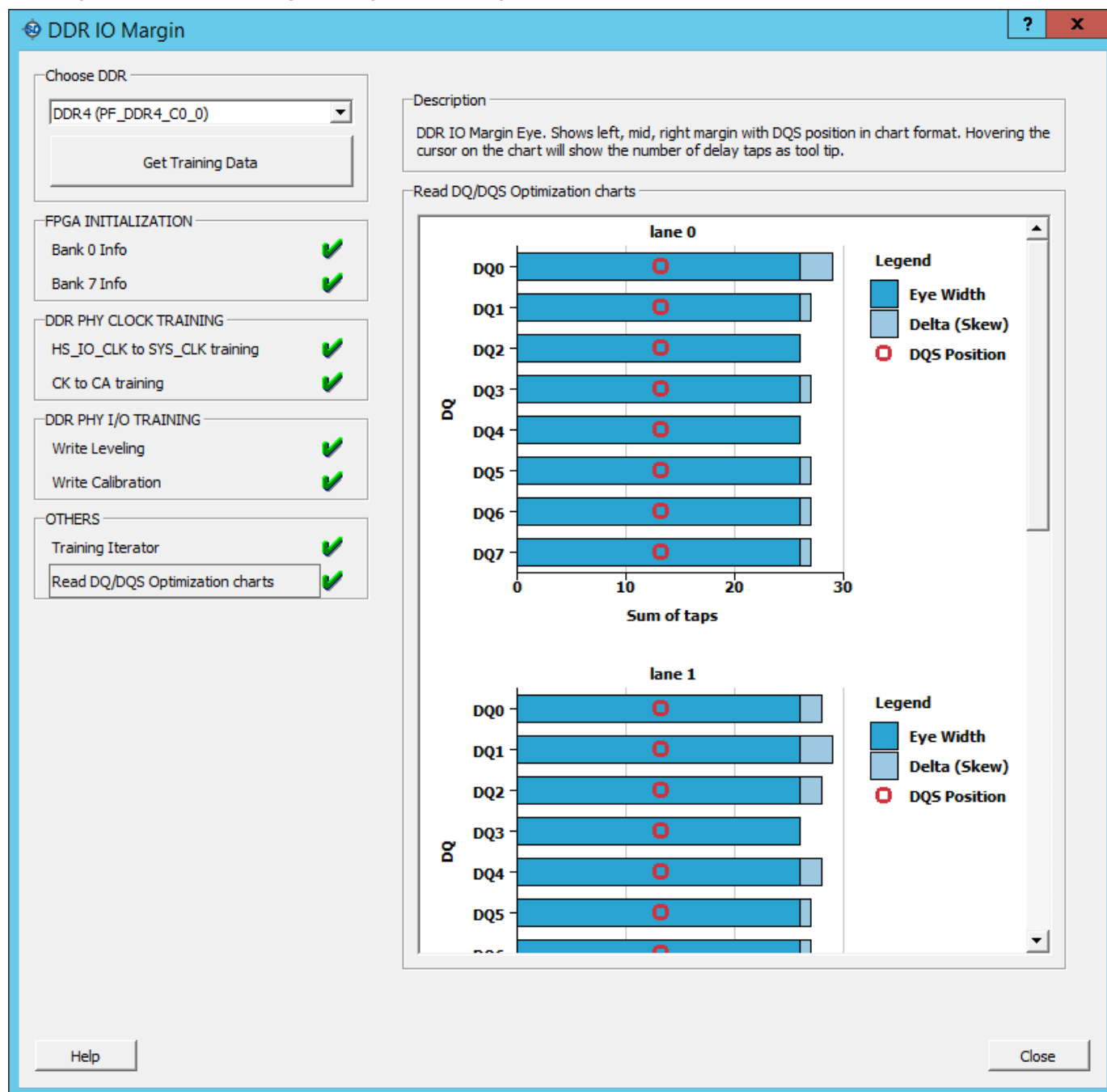


Figure 3-98. DDR IO Margin Dialog Box Showing Read DQ/DQS Optimization Charts



The tool informs the users about errors generated while getting training data as shown in the following figure.

Figure 3-99. DDR IO Margin Dialog Box Showing Error Generated when DDR PLL is Not Locked

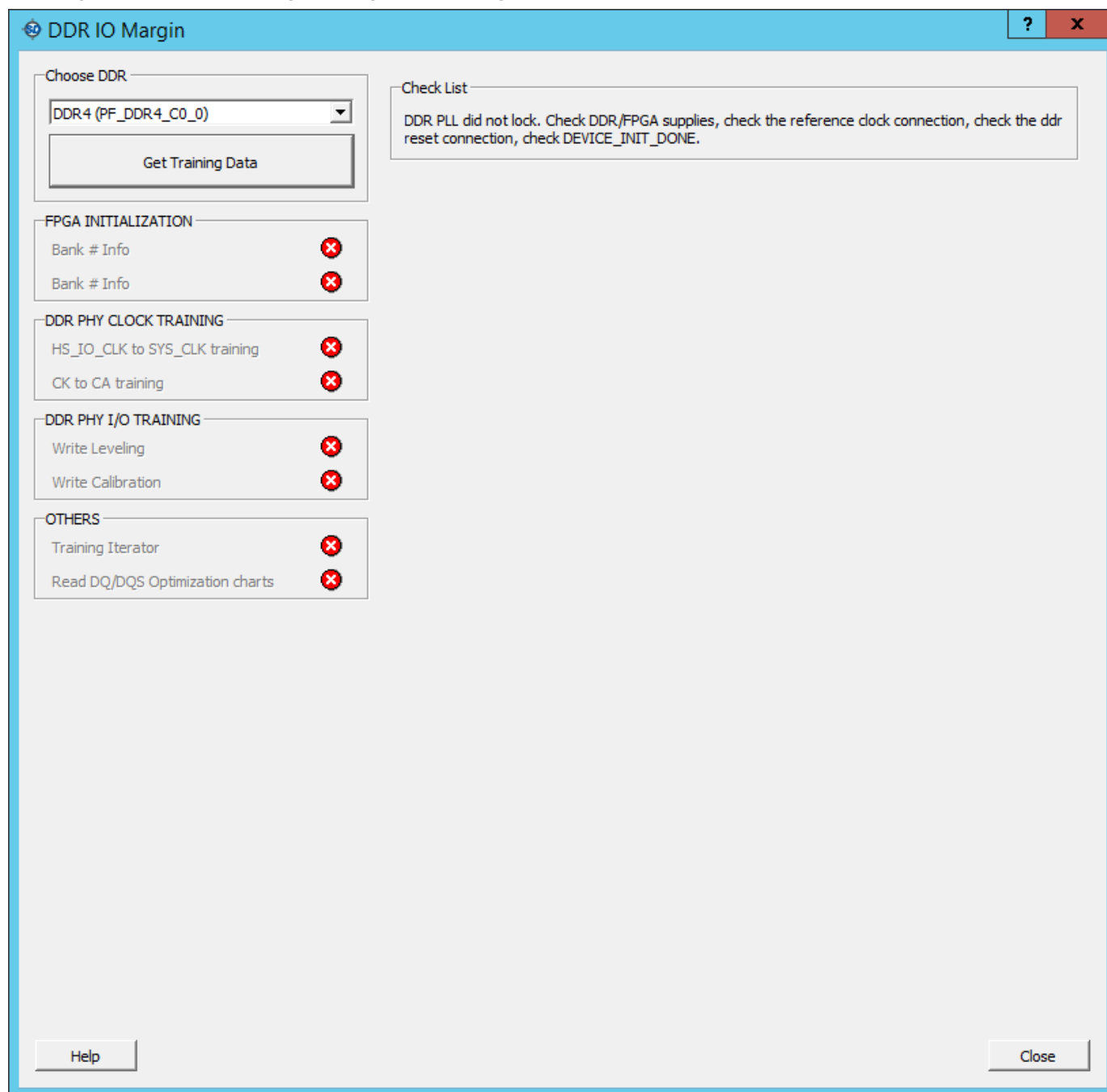




Figure 3-100. DDR IO Margin Dialog Box Showing Error Generated During Write Levelling Data

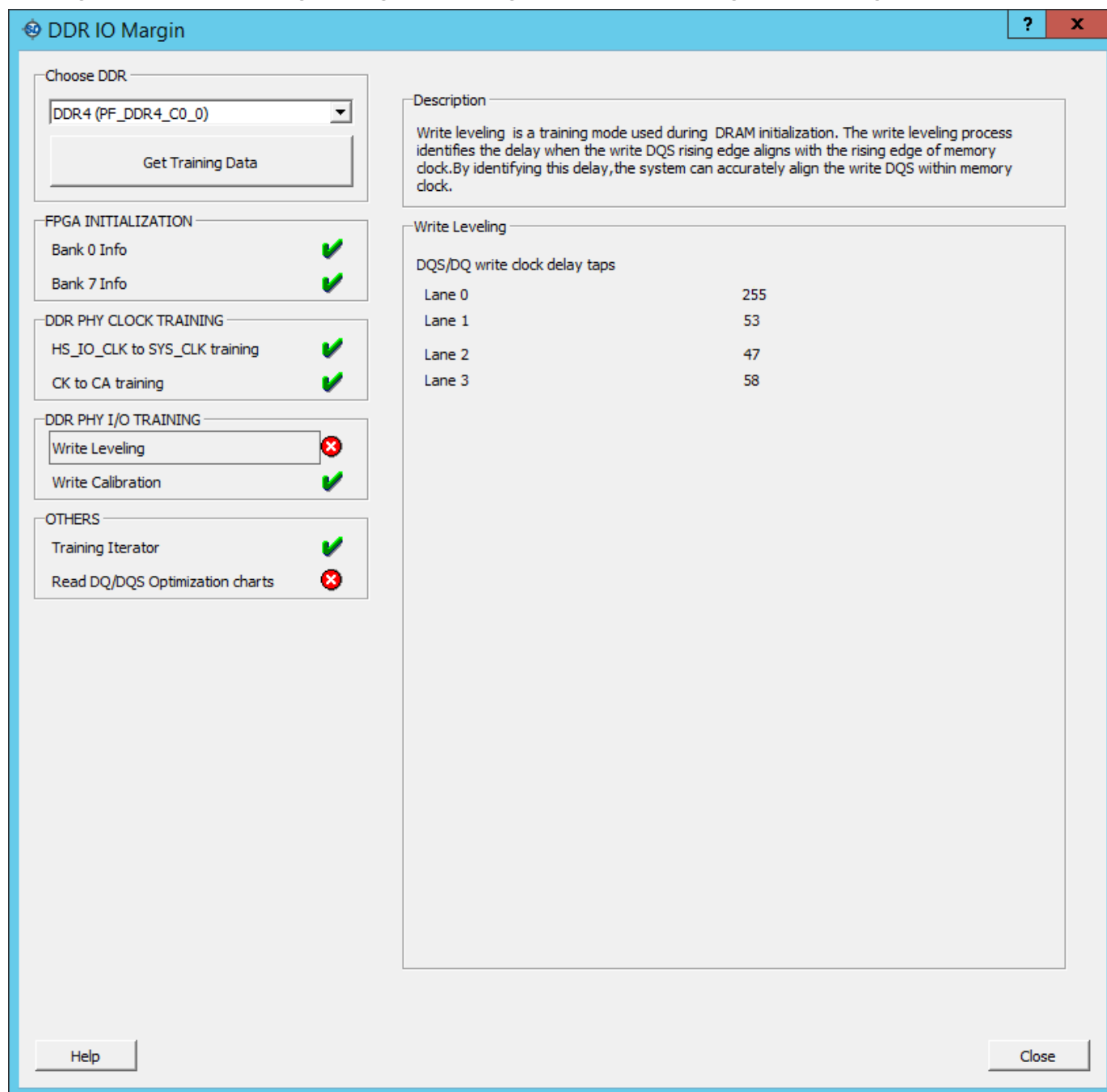


Figure 3-101. DDR IO Margin Dialog Box Showing Error Generated During Write Levelling Data

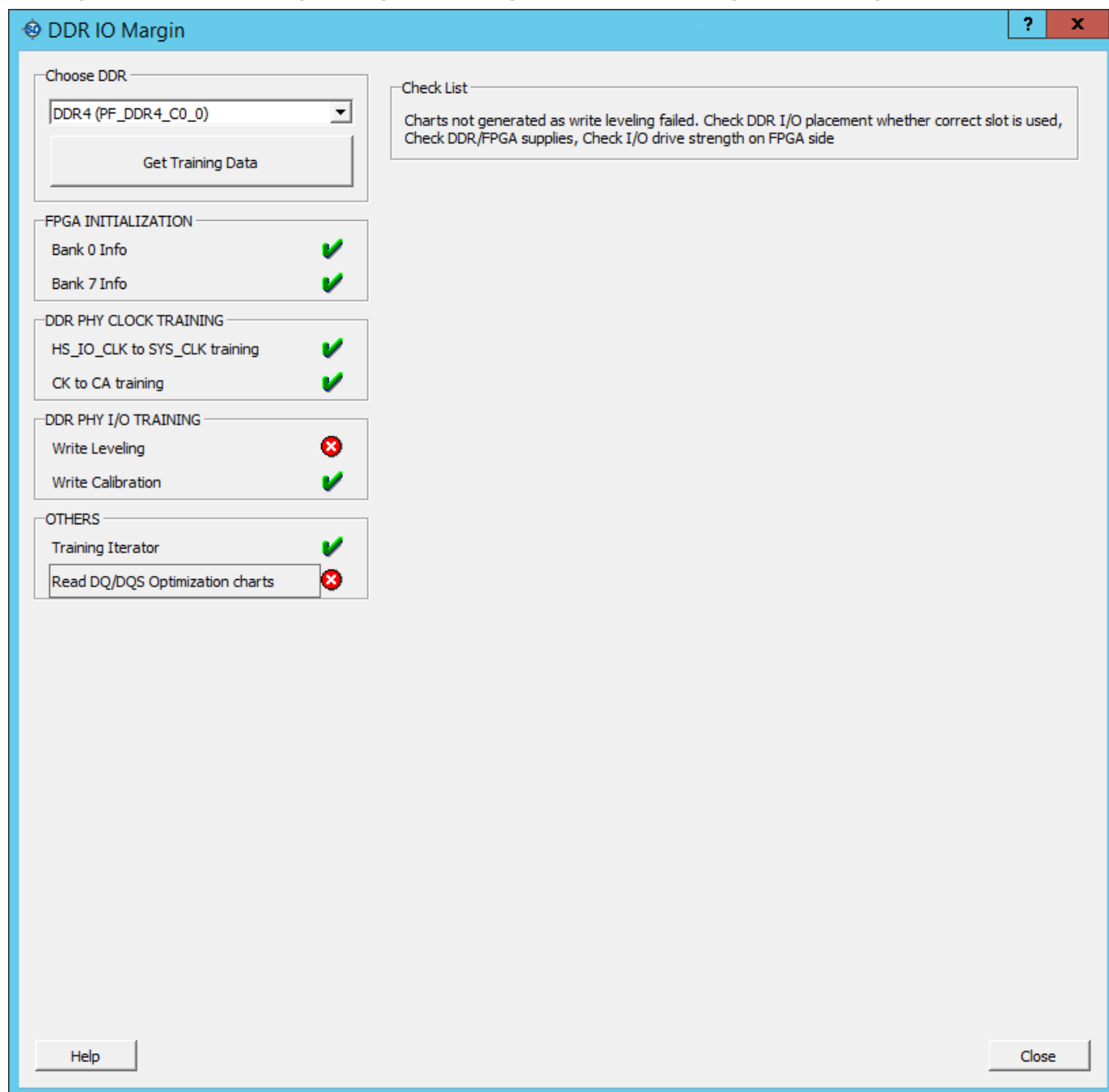
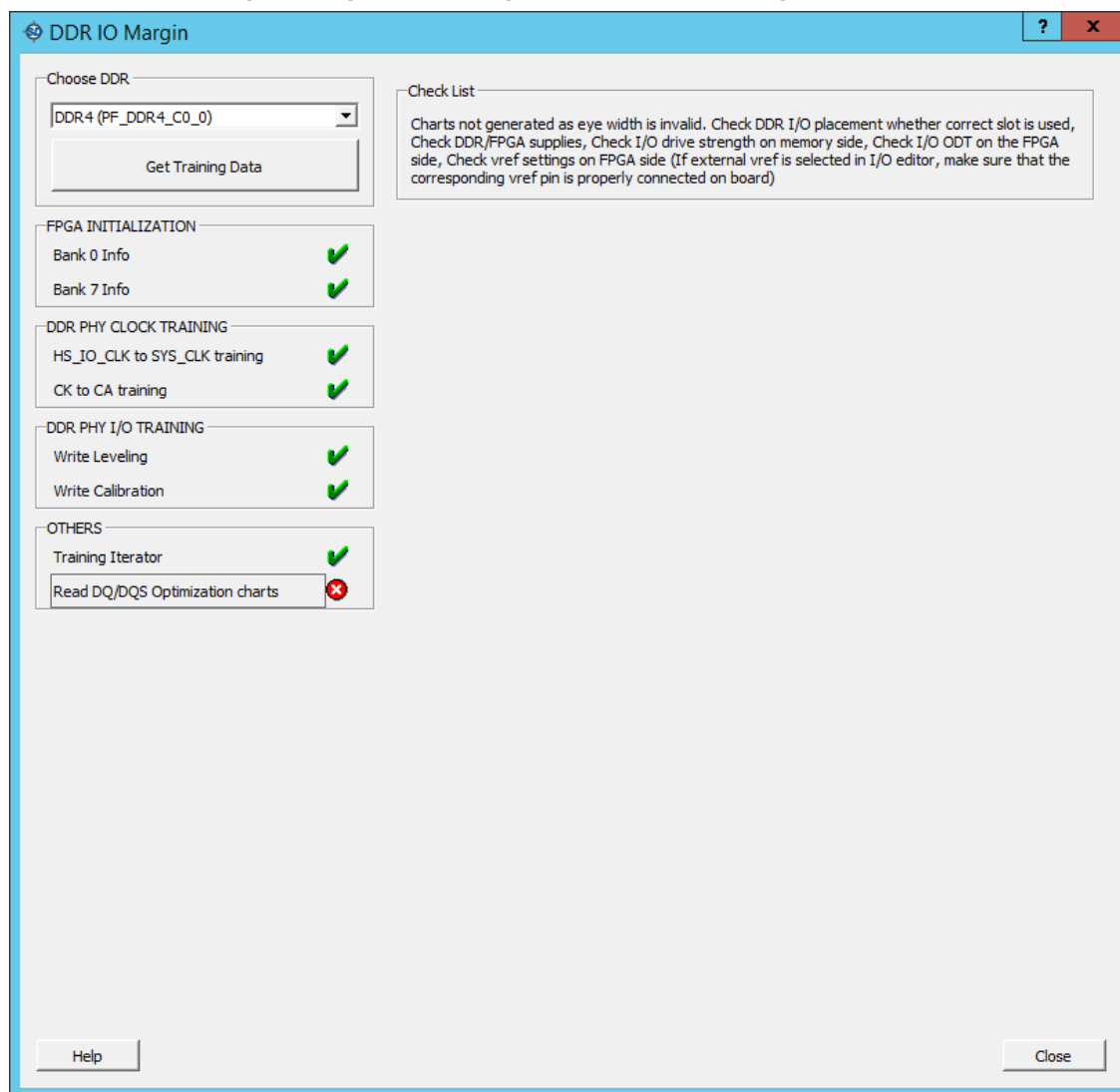


Figure 3-102. DDR IO Margin Dialog Box Showing Error Generated During Read DQ/DQS Optimization Charts



### 3.19 Debug eNVM

Debug eNVM is applicable for PolarFire SoC devices only. ENVM memory is present in the MSS core. This memory can be used to store user or application data of an SoC design. The total space that ENVM memory can hold is 128 Kbytes. ENVM is divided into four sectors - Sector 0, 1, 2, and 3. Libero tool can be used to configure the memory and place the data from a file or override the locations.

The configurator for ENVM memory has divided the memory into pages. A total of 512 pages each comprising 256 bytes of data. Multiple clients can be configured, and each client can hold data in multiple pages.

**Table 3-4. ENVM Memory Division with Page Size and MSS Address**

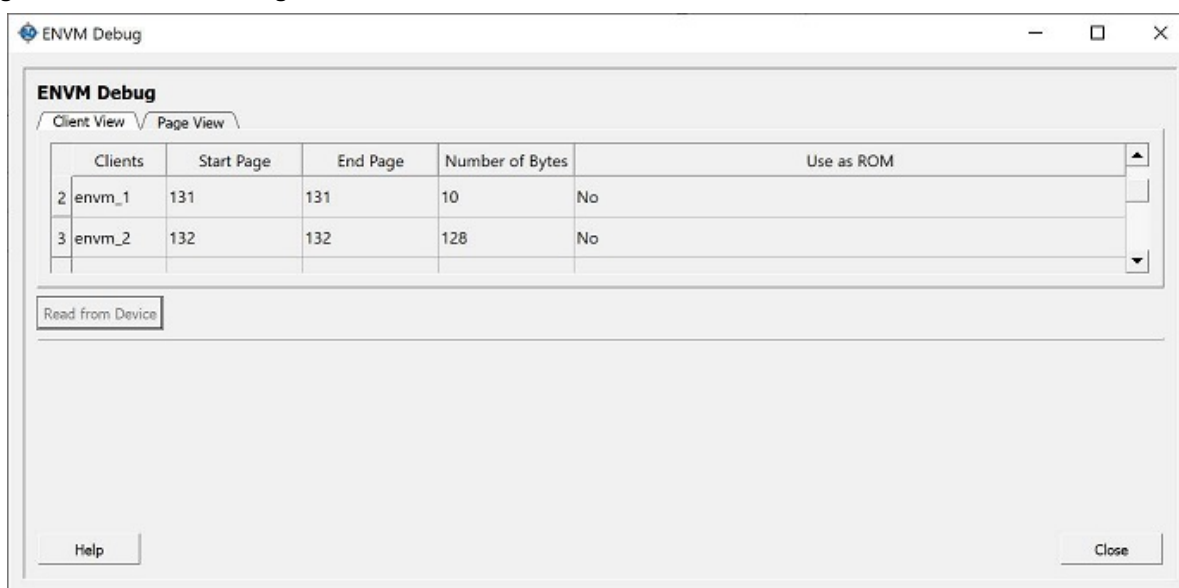
Size		Offset	Description	MSS Address
128 Kbytes	8K	0x00000000	Sector 2	0x20220000
	56K	0x00002000	Sector 0	0x20222000
	56K	0x00010000	Sector 1	0x20230000
	8K	0x0001E000	Sector 3	0x2023E000

There are two views present in the ENVM Debug window - [Client View](#) and [Page View](#).

### 3.19.1 Client View

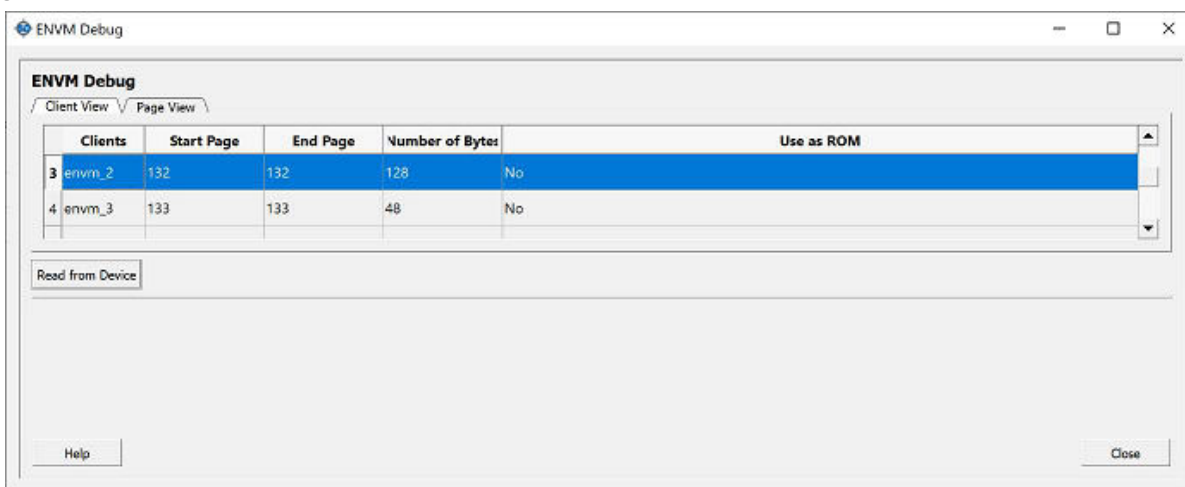
SmartDebug shows clients configured through the Configure Design Initialization Data and Memories tool in the Design Flow. Client view allows the user to select a client and read the data present from the device. Details like client name, client start page number, client end page number, number of bytes used and whether the client is used as a ROM are displayed in **ENVM Debug** window, see the following figure.

**Figure 3-103. ENVM Debug Window with Client View**



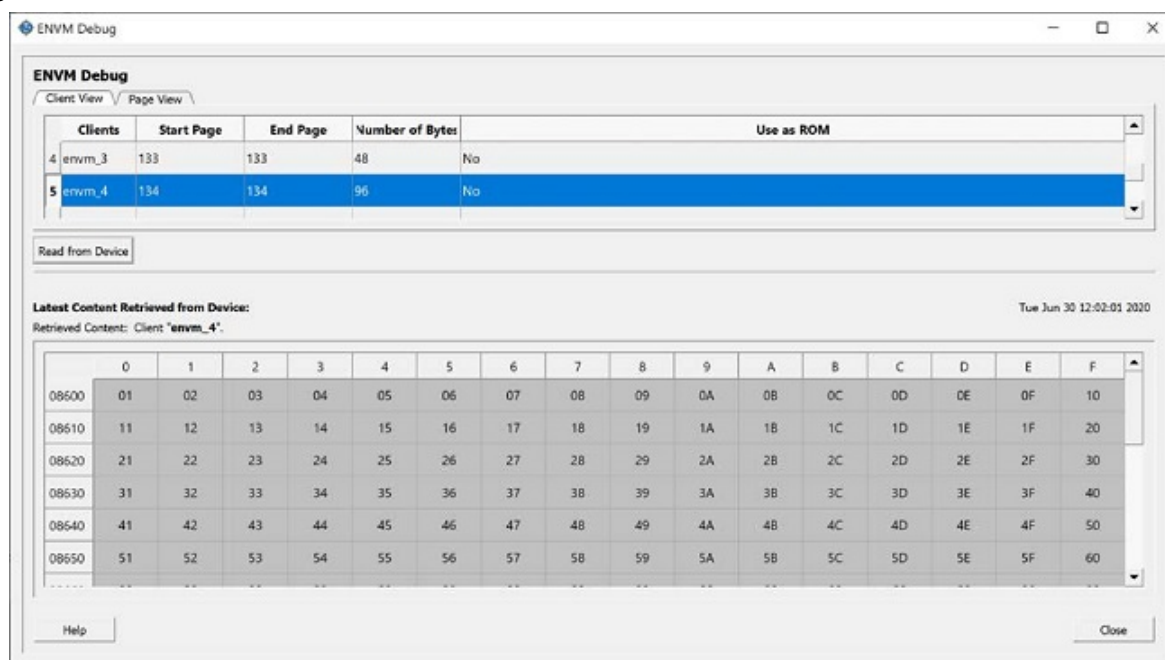
A single client can be selected at a time to view the content. Once a client is selected, the **Read from Device** button is enabled.

Figure 3-104. Read from Device on Selection



Click the **Read from Device** button to see the content displayed as a matrix of cells. The headers are added to show the MSS address offsets to each of the locations as well as on the row headers.

Figure 3-105. Client View Data



### 3.19.2 Page View

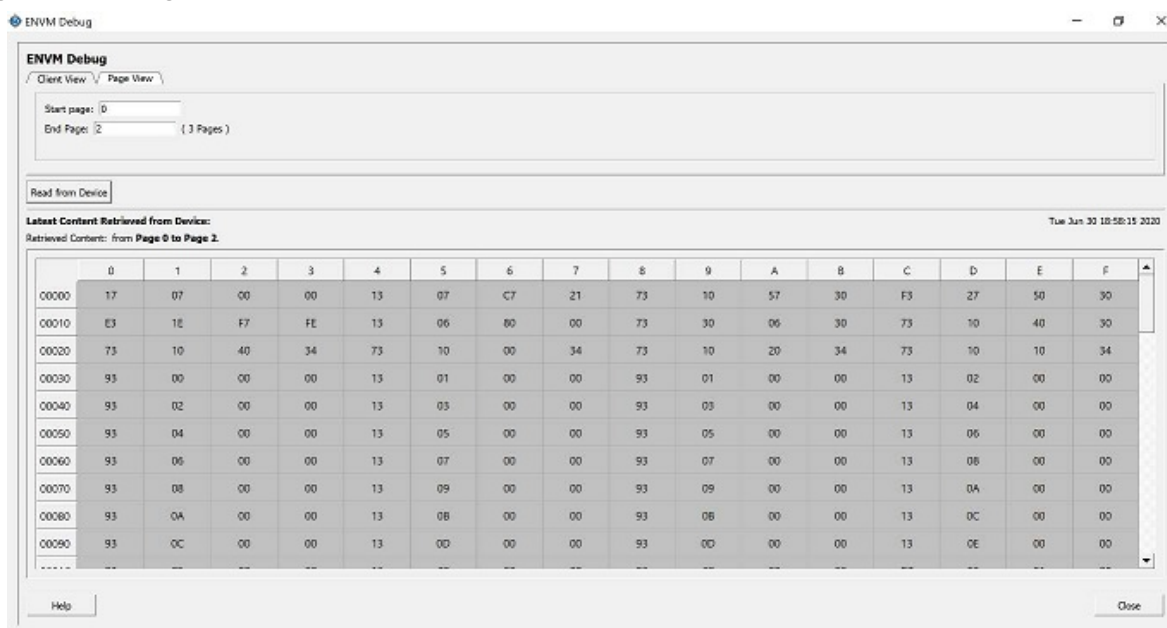
Page view can be seen when the **Page View** tab is selected in the **ENVM Debug** Window.

Start page and end page number options can be edited to input valid page numbers. Start page can be between 0 to 511. End Page can be between start page and 511.

Error messages will be displayed if incorrect entries are entered. **Read from Device** button will be enabled only after entering valid entries.

Click the **Read from Device** button to see the page content.

**Figure 3-106. Page View Data**



## **4. SmartDebug Tcl Commands**

For more information on the SmartDebug Tcl commands supported in PolarFire, see [PolarFire FPGA Tcl Commands Reference Guide](#).

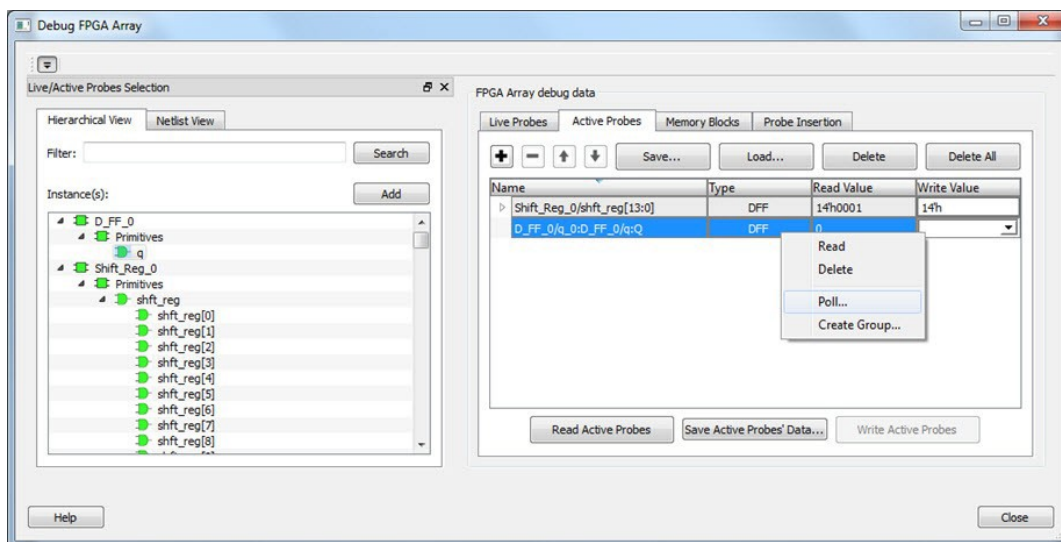


## 5. Frequently Asked Questions

### 5.1 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**.



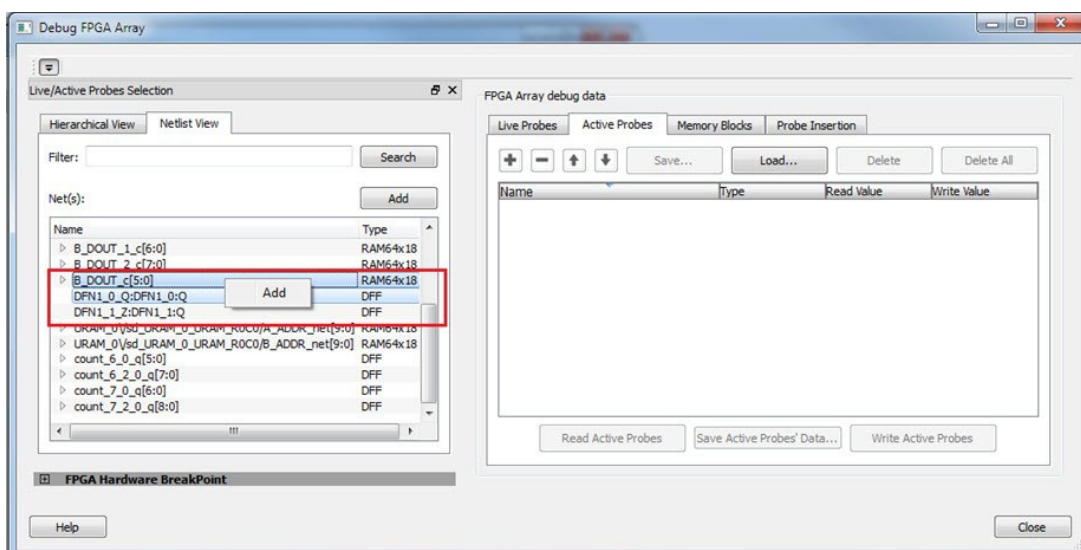
3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click **Start Polling**.



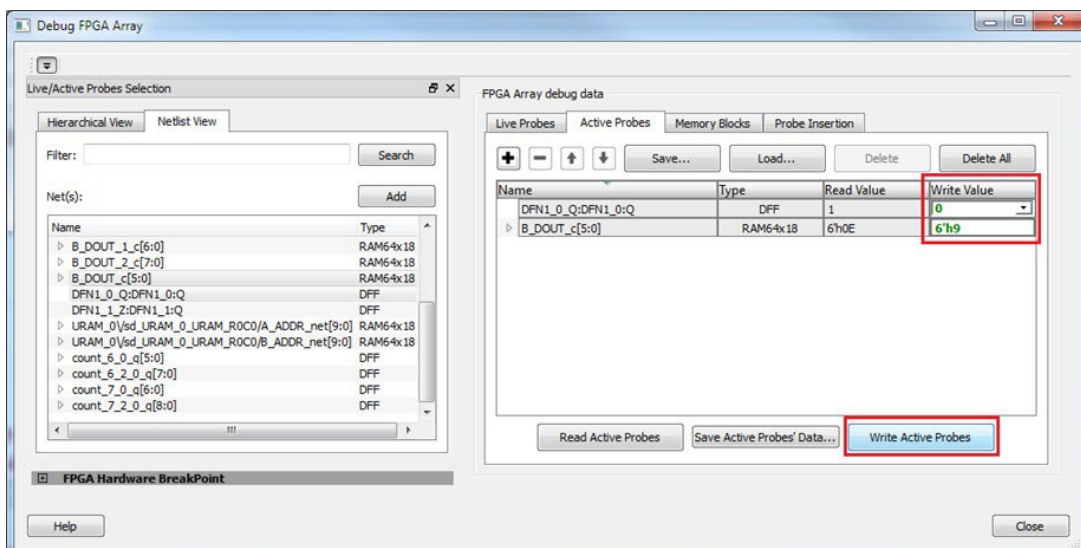
### 5.2 How do I force a signal to a new value?

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.



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

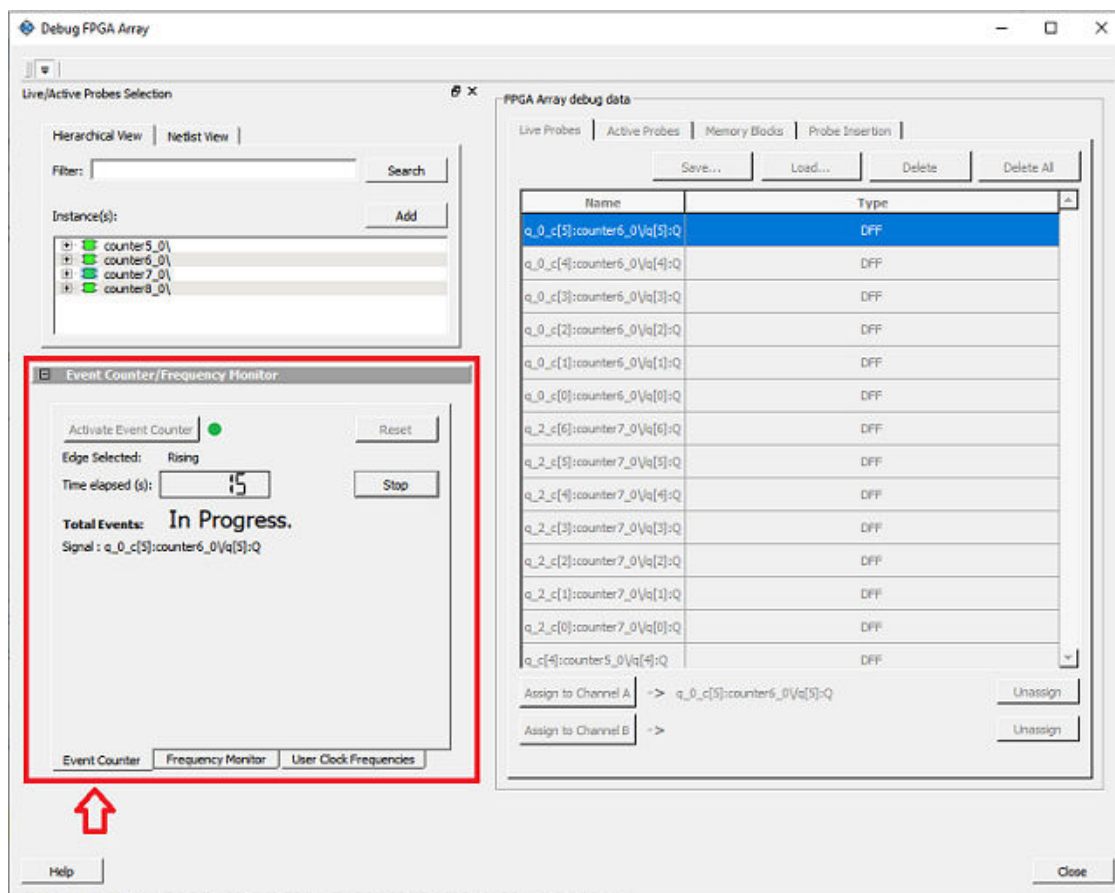


### 5.3 How do I count the transitions on a signal?

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. For more information, see [Event Counter](#).

To count the transitions on a signal:

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

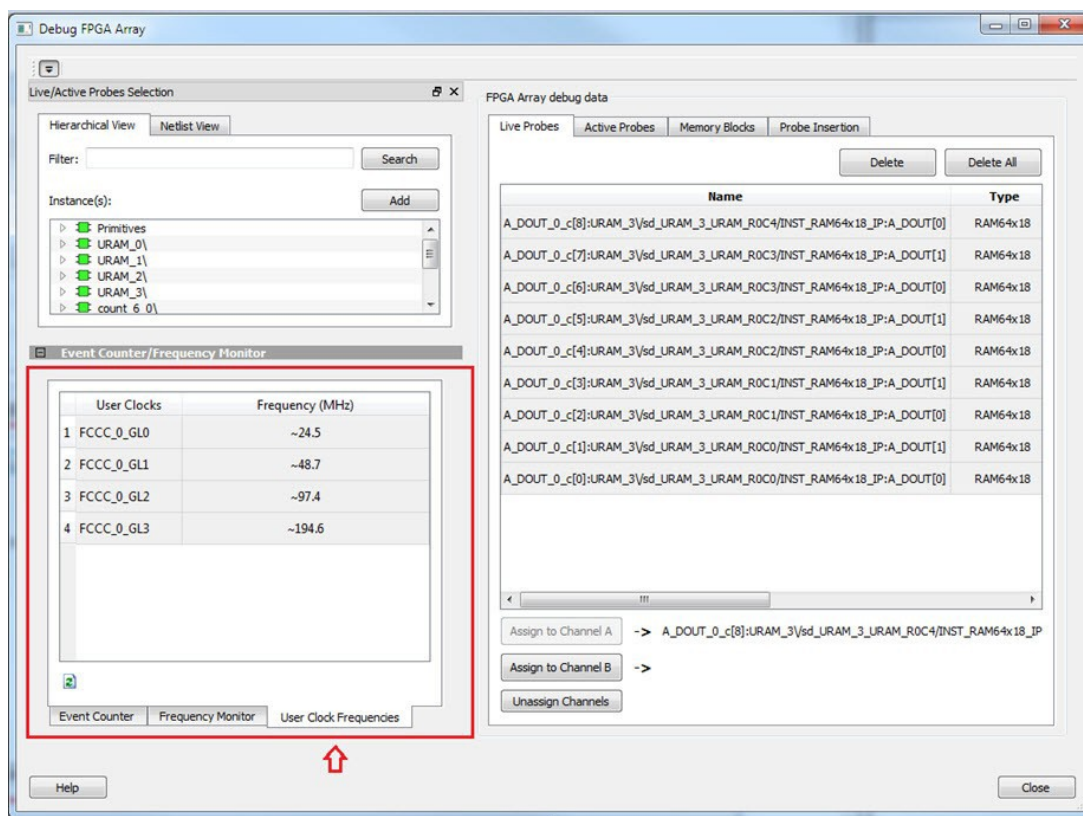


## 5.4 How do I monitor or measure a clock?

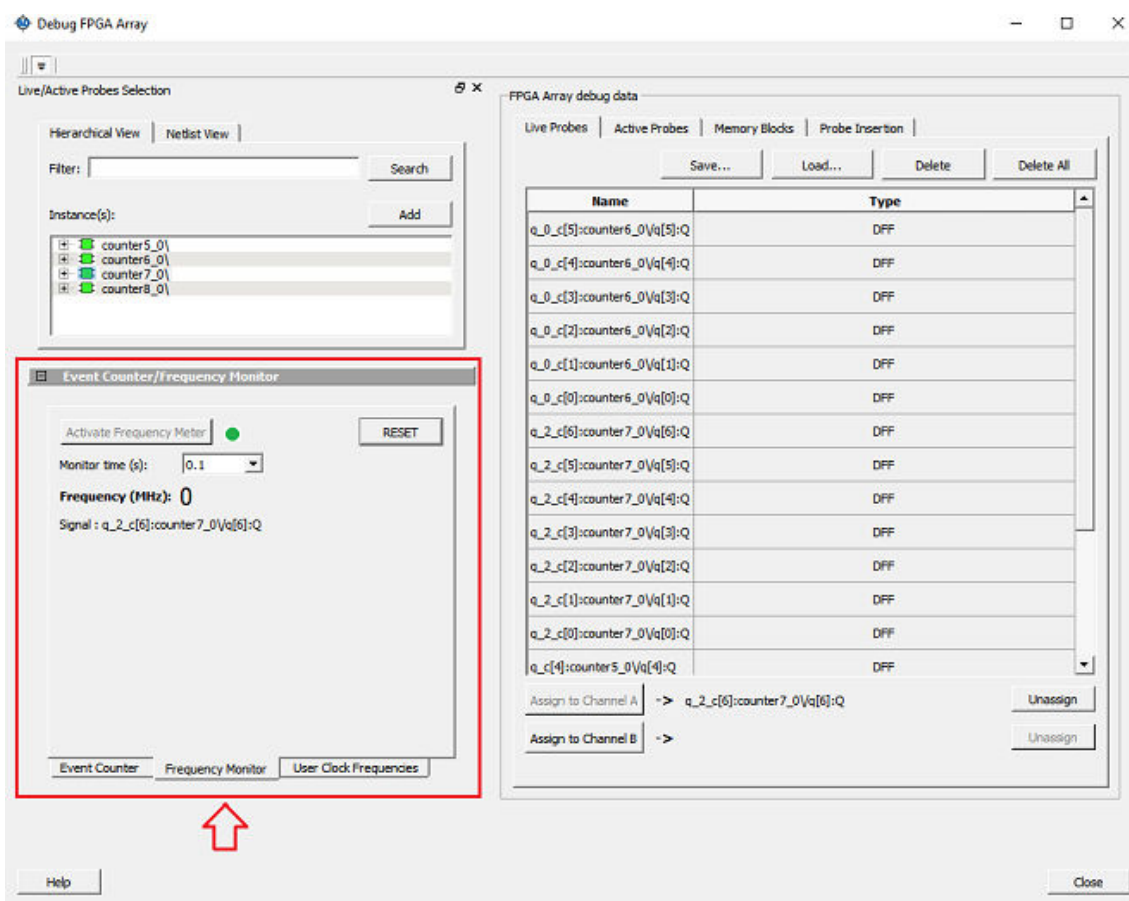
You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in the **Project Settings** dialog box.

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 following figure.



2. You can monitor frequencies of any probe points by:
  1. Assigning the desired signal to the Live Probe Channel A.
  2. Selecting the **Frequency Monitor** tab as shown in the following figure



3. Selecting the **Activate Frequency Meter** check-box.

## 5.5 How do I perform simple SmartBERT tests?

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

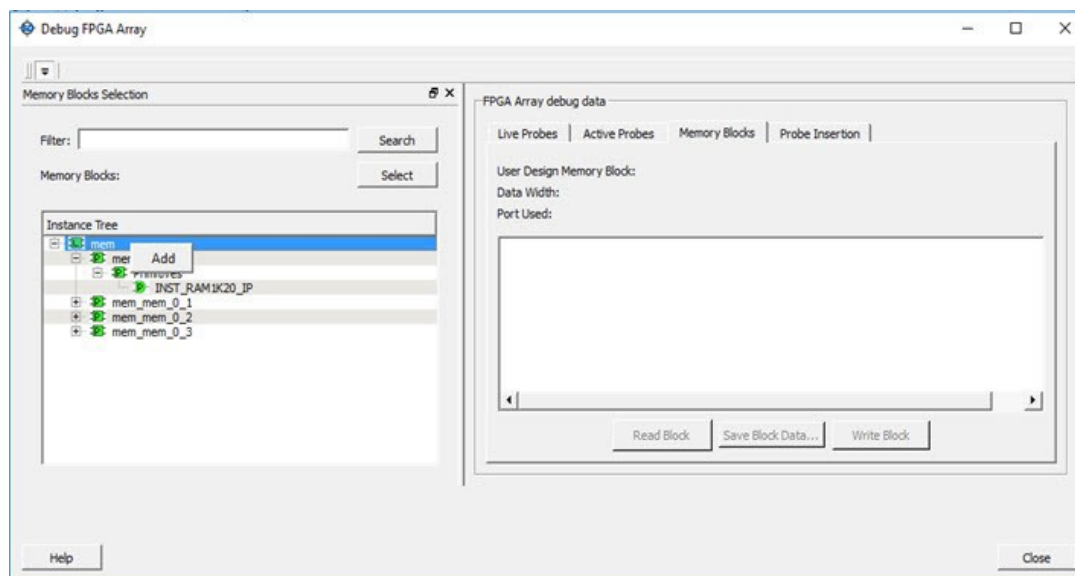
To perform a SmartBERT test, in the **SmartBERT** page of the **Debug Transceiver** dialog box, select to run a PRBS test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see [3.14.3 SmartBERT](#).

To perform a SmartBERT test, in the Smart BERT page of the Debug Transceiver dialog box, select your options and click **Start** to run a Smart BERT test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see [3.14.3 SmartBERT](#).

## 5.6 How do I read LSRAM or USRAM content?

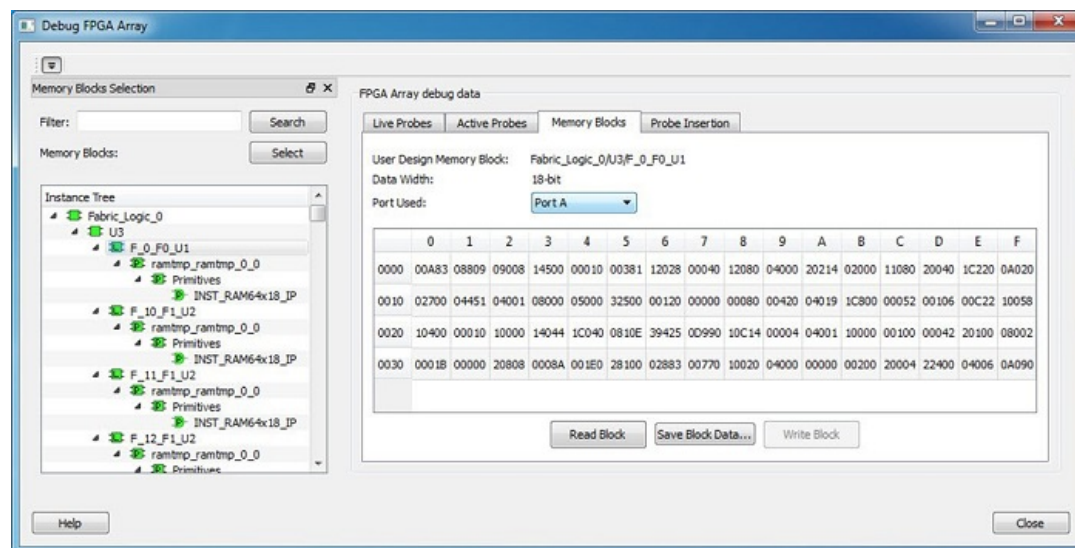
To read RAM content:

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:
  1. Click **Select**.
  2. Right-click and choose **Add**.
  3. Drag the block to the **Memory Blocks** tab.
4. Click **Read Block** to read the content of the block.



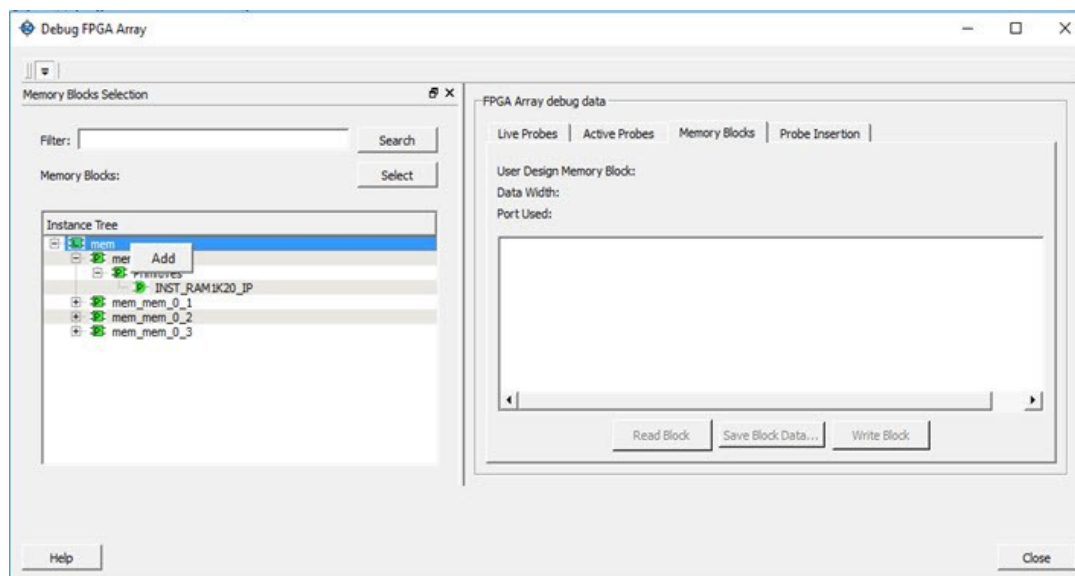
For more information, see [Memory Blocks](#).

## 5.7 How do I change the content of LSRAM or USRAM?

To change the content of LSRAM or USRAM:

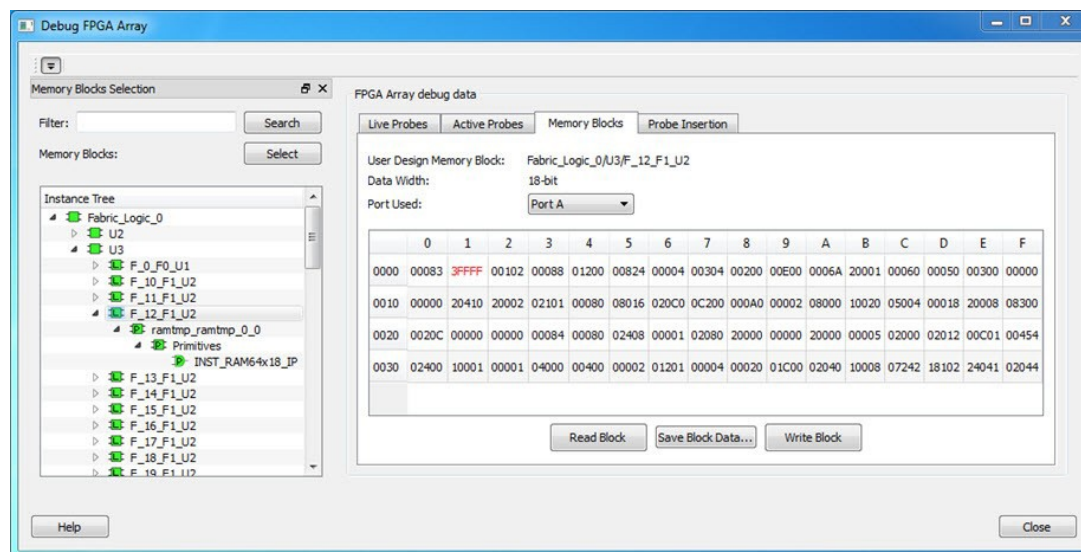
1. In the SmartDebug window, click **Debug FPGA Array**.
2. Click the **Memory Blocks** tab.
3. Select the memory block from the selection panel.





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:
  1. Click **Select**.
  2. Right-click and choose **Add**.
  3. 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.



For more information, see [Memory Blocks](#).

## 5.8 How do I read the health check of the Transceiver?

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



1. Review the **Configuration Report**, which returns Tx PMA Ready, Rx PMA Ready, TxPLL status, and RxPLL status. For the transceiver to function correctly, all four should be green. The Configuration Report can be found in the Debug TRANSCEIVER dialog box under Configuration Report. For more information, see [Debug Transceiver](#).

**Figure 5-1. Debug Transceiver**

Lanes	PF_XCVR_Q0_L2_L3	PF_XCVR_Q1_L2_L3	PF_XCVR_Q2_L2_L3	PF_XCVR_Q3_L2_L3
<b>LANE0</b>				
Physical Location	Q0_LANE2	Q1_LANE2	Q2_LANE2	Q3_LANE2
Tx PMA Ready	●	●	●	●
Rx PMA Ready	●	●	●	●
TX PLL	●	●	●	●
RX PLL	●	●	●	●
RX CDR PLL	●	●	●	●
Data Width	20 bit	20 bit	20 bit	40 bit
<b>LANE1</b>				
Physical Location	Q0_LANE3	Q1_LANE3	Q2_LANE3	Q3_LANE3
Tx PMA Ready	●	●	●	●
Rx PMA Ready	●	●	●	●
TX PLL	●	●	●	●
RX PLL	●	●	●	●
RX CDR PLL	●	●	●	●
Data Width	20 bit	20 bit	20 bit	40 bit

2. Run the SmartBERT Test, with EQ-NEAR END checked or with external loopback connection from Tx to Rx on selected lanes. This should result in 0 errors in the Cumulative Error Count column. For more information, see [SmartBERT](#).

## 6. Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision	Date	Description
B	04/2021	<ul style="list-style-type: none"> <li>Added the <a href="#">3.15 MSS Register Access</a> section.</li> <li>Added the <a href="#">3.16 Debug IOD</a> section.</li> <li>Updated the <a href="#">3.14.7 Register Access</a> section.</li> </ul>
A	11/2020	<ul style="list-style-type: none"> <li>Added read/write access support to the XCVR registers.</li> <li>Added enhancements for the Two-Port LSRAM configured in the ECC mode.</li> </ul>
8.0	06/2020	<ul style="list-style-type: none"> <li>Added a note about SCB Read operations.</li> <li>Added embedded FlashPro6 for supported programmers.</li> <li>Added load and save button information in the Live Probes section.</li> <li>Added a note about PCIe SMARTBERT tests and unused lanes.</li> <li>Added information about the new Eye Mask feature.</li> <li>Added information about Record Actions in the Debug TRANSCEIVER window.</li> <li>Added information about the new Debug DDR IO Margin feature.</li> <li>Added information about Debug eNVM.</li> </ul>

## 7. Microchip FPGA Technical Support

Microchip FPGA Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. This section provides information about contacting Microchip FPGA Products Group and using these support services.

### 7.1 Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

### 7.2 Customer Technical Support

Microchip FPGA Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microchip FPGA Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

You can communicate your technical questions through our Web portal and receive answers back by email, fax, or phone. Also, if you have design problems, you can upload your design files to receive assistance. We constantly monitor the cases created from the web portal throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

Technical support can be reached at [soc.microsemi.com/Portal/Default.aspx](https://soc.microsemi.com/Portal/Default.aspx).

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), log in at [soc.microsemi.com/Portal/Default.aspx](https://soc.microsemi.com/Portal/Default.aspx), go to the **My Cases** tab, and select **Yes** in the ITAR drop-down list when creating a new case. For a complete list of ITAR-regulated Microchip FPGAs, visit the ITAR web page.

You can track technical cases online by going to [My Cases](#).

### 7.3 Website

You can browse a variety of technical and non-technical information on the Microchip FPGA Products Group [home page](#), at [www.microsemi.com/soc](https://www.microsemi.com/soc).

### 7.4 Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support at (<https://soc.microsemi.com/Portal/Default.aspx>) or contact a local sales office.

Visit [About Us](#) for [sales office listings](#) and [corporate contacts](#).

## The Microchip Website

---

Microchip provides online support via our website at [www.microchip.com/](http://www.microchip.com/). This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Product Change Notification Service

---

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to [www.microchip.com/pcn](http://www.microchip.com/pcn) and follow the registration instructions.

## Customer Support

---

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: [www.microchip.com/support](http://www.microchip.com/support)

## Microchip Devices Code Protection Feature

---

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

## Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQL, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-7831-7

## **Quality Management System**

---

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

## Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<b>Corporate Office</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: <a href="http://www.microchip.com/support">www.microchip.com/support</a> Web Address: <a href="http://www.microchip.com">www.microchip.com</a>	<b>Australia - Sydney</b> Tel: 61-2-9868-6733 <b>China - Beijing</b> Tel: 86-10-8569-7000 <b>China - Chengdu</b> Tel: 86-28-8665-5511 <b>China - Chongqing</b> Tel: 86-23-8980-9588 <b>China - Dongguan</b> Tel: 86-769-8702-9880 <b>China - Guangzhou</b> Tel: 86-20-8755-8029 <b>China - Hangzhou</b> Tel: 86-571-8792-8115 <b>China - Hong Kong SAR</b> Tel: 852-2943-5100 <b>China - Nanjing</b> Tel: 86-25-8473-2460 <b>China - Qingdao</b> Tel: 86-532-8502-7355 <b>China - Shanghai</b> Tel: 86-21-3326-8000 <b>China - Shenyang</b> Tel: 86-24-2334-2829 <b>China - Shenzhen</b> Tel: 86-755-8864-2200 <b>China - Suzhou</b> Tel: 86-186-6233-1526 <b>China - Wuhan</b> Tel: 86-27-5980-5300 <b>China - Xian</b> Tel: 86-29-8833-7252 <b>China - Xiamen</b> Tel: 86-592-2388138 <b>China - Zhuhai</b> Tel: 86-756-3210040	<b>India - Bangalore</b> Tel: 91-80-3090-4444 <b>India - New Delhi</b> Tel: 91-11-4160-8631 <b>India - Pune</b> Tel: 91-20-4121-0141 <b>Japan - Osaka</b> Tel: 81-6-6152-7160 <b>Japan - Tokyo</b> Tel: 81-3-6880-3770 <b>Korea - Daegu</b> Tel: 82-53-744-4301 <b>Korea - Seoul</b> Tel: 82-2-554-7200 <b>Malaysia - Kuala Lumpur</b> Tel: 60-3-7651-7906 <b>Malaysia - Penang</b> Tel: 60-4-227-8870 <b>Philippines - Manila</b> Tel: 63-2-634-9065 <b>Singapore</b> Tel: 65-6334-8870 <b>Taiwan - Hsin Chu</b> Tel: 886-3-577-8366 <b>Taiwan - Kaohsiung</b> Tel: 886-7-213-7830 <b>Taiwan - Taipei</b> Tel: 886-2-2508-8600 <b>Thailand - Bangkok</b> Tel: 66-2-694-1351 <b>Vietnam - Ho Chi Minh</b> Tel: 84-28-5448-2100	<b>Austria - Wels</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 <b>Denmark - Copenhagen</b> Tel: 45-4485-5910 Fax: 45-4485-2829 <b>Finland - Espoo</b> Tel: 358-9-4520-820 <b>France - Paris</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 <b>Germany - Garching</b> Tel: 49-8931-9700 <b>Germany - Haan</b> Tel: 49-2129-3766400 <b>Germany - Heilbronn</b> Tel: 49-7131-72400 <b>Germany - Karlsruhe</b> Tel: 49-721-625370 <b>Germany - Munich</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 <b>Germany - Rosenheim</b> Tel: 49-8031-354-560 <b>Israel - Ra'anana</b> Tel: 972-9-744-7705 <b>Italy - Milan</b> Tel: 39-0331-742611 Fax: 39-0331-466781 <b>Italy - Padova</b> Tel: 39-049-7625286 <b>Netherlands - Drunen</b> Tel: 31-416-690399 Fax: 31-416-690340 <b>Norway - Trondheim</b> Tel: 47-72884388 <b>Poland - Warsaw</b> Tel: 48-22-3325737 <b>Romania - Bucharest</b> Tel: 40-21-407-87-50 <b>Spain - Madrid</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 <b>Sweden - Gothenberg</b> Tel: 46-31-704-60-40 <b>Sweden - Stockholm</b> Tel: 46-8-5090-4654 <b>UK - Wokingham</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820