

Introduction (Ask a Question)

Design debug is a critical phase of the FPGA design flow. Microchip's SmartDebug tool complements design simulation by enabling verification and troubleshooting at the hardware level.

Using SmartDebug, you can debug Microchip FPGA arrays and SerDes via their respective Joint Test Action Group (JTAG) port(s) without requiring an internal logic analyzer. SmartDebug can also capture FPGA device status, MSS register access, and flash and DDR memory content.



Attention: SmartDebug works only with JTAG ports and does not support SPI ports. SmartDebug is specifically designed for debugging devices connected in a JTAG chain.

SmartDebug uses dedicated and specialized probe points built into the FPGA fabric to accelerate and simplify the debug process significantly. It also allows you to select or change different probe point on-the-fly without additional overhead, saving significant recompile time.

SmartDebug can be accessed within the Libero[®] design flow or as a standalone software application.



Important: This document is updated frequently. The latest version of this document is available at this location: [Libero SoC Design Suite Documentation](#).

Supported Device Families (Ask a Question)

The following table lists the family of devices that SmartDebug supports. This guide covers all these device families. However, some information in this guide might apply to certain device families only. In this case, such information is clearly identified.

Table 1. Device Families Supported by SmartDebug

Device Family	Description
PolarFire [®]	PolarFire FPGAs deliver the industry's lowest power at mid-range densities with exceptional security and reliability.
RT PolarFire	RT PolarFire is Microchip's family of next-generation radiation-tolerant FPGAs that enable higher computing and connectivity throughput at 40 to 50% lower power than competing SRAM FPGAs.
PolarFire SoC	PolarFire SoC is the first SoC FPGA with a deterministic, coherent RISC-V CPU cluster, and a deterministic L2 memory subsystem enabling Linux [®] and real-time applications.
RT PolarFire [®] SoC	RT PolarFire SoC FPGA is the industry's first embedded, real-time, Linux [®] -capable, RISC-V-based Microprocessor Subsystem (MSS) on the flight-proven, radiation-tolerant PolarFire FPGA fabric.
SmartFusion [®] 2	SmartFusion 2 addresses fundamental requirements for advanced security, high reliability, and low power in critical industrial, military, aviation, communications, and medical applications.
IGLOO [®] 2	IGLOO 2 is a low-power mixed-signal programmable solution.
RTG4 [™]	RTG4 is Microchip's family of radiation-tolerant FPGAs.

Note: For SCB read operations with PolarFire devices, if the APB DRI bus performs an SCB read operation while SmartDebug is trying to read from the 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 becomes corrupt and SmartDebug may read corrupt data.

Supported Tools [\(Ask a Question\)](#)

The following table lists the device family support for SmartDebug tools. A check mark indicates that the tool is supported.

Table 2. Device Family Support for SmartDebug Tools

SmartDebug Support per Device Family	PolarFire [®] and RT PolarFire	PolarFire SoC and RT PolarFire [®] SoC	SmartFusion [®] 2	IGLOO [®] 2	RTG4 [™] FPGAs
Active Probes	✓	✓	✓	✓	✓
Fabric DDR	✓	✓	×	×	×
MSS DDR	×	✓	×	×	×
Debug IOD	✓	✓	×	×	×
Debug SNVM	✓	✓	×	×	×
Debug Transceiver/SerDes	✓	✓	✓	✓	✓
PCIe Debug	✓	✓	×	×	×
Debug uPROM	✓	✓	×	×	×
Event Counter (needs FHB Auto Instantiation)	✓	✓	✓	✓	✓
FPGA Hardware Breakpoint (needs FHB Auto Instantiation)	✓	✓	✓	✓	✓
Frequency Monitor (needs FHB Auto Instantiation)	✓	✓	✓	✓	×
Live Probes	✓	✓	✓	✓	✓
Memory Debug	✓	✓	✓	✓	✓
MSS Register Access	×	✓	×	×	×
Probe Insertion (available only through Libero [®] flow)	✓	✓	✓	✓	✓
TVS Monitor	✓	✓	×	×	×
View Flash Memory Content—eNVM Debug	×	✓	✓	✓	×



Important: The contents of eNVM cannot be debugged in Boot mode 0. Therefore, before using SmartDebug, Debug eNVM requires a device to be in an Active Boot mode (that is, a Boot mode other than 0) and have a valid embedded software application running that enables the MPU.

Table of Contents

Introduction.....	1
Supported Device Families.....	1
Supported Tools.....	2
1. Getting Started with SmartDebug.....	4
1.1. Use Models.....	4
1.2. Configuring a Generic Device.....	6
1.3. Debugging Devices Connected in a JTAG Chain.....	6
2. SmartDebug User Interface.....	10
2.1. Standalone SmartDebug User Interface.....	10
2.2. Programming Connectivity and Interface.....	12
2.3. View Device Status.....	16
2.4. Rescan Programmer.....	18
3. Debugging.....	20
3.1. Common Debug Elements.....	20
3.2. PolarFire and PolarFire SoC Debug Elements.....	59
3.3. SmartFusion 2, IGLOO 2, and RTG4 Debug Elements.....	148
4. Frequently Asked Questions.....	159
4.1. SmartDebug FAQs for PolarFire.....	159
4.2. SmartDebug FAQs for SmartFusion 2, IGLOO 2, and RTG4.....	166
5. SmartDebug Tcl Commands.....	179
5.1. Smart Debug Tcl Commands.....	179
6. Revision History.....	263
Microchip FPGA Support.....	268
Microchip Information.....	268
Trademarks.....	268
Legal Notice.....	268
Microchip Devices Code Protection Feature.....	269

1. Getting Started with SmartDebug [\(Ask a Question\)](#)

SmartDebug allows you to interrogate and view embedded silicon features and device status. The following procedure describes the most common steps for using SmartDebug.

Note: To use SmartDebug, connect a FlashPro programmer from a PC to the target device.

1. Create your design.
2. Expand **Debug Design** and double click **SmartDebug 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.

1.1. Use Models [\(Ask a Question\)](#)

SmartDebug can be run in the following modes:

- [Integrated mode](#)
- [Standalone mode](#)
- [Demo mode](#)

1.1.1. Integrated Mode [\(Ask a Question\)](#)

Running SmartDebug in Integrated mode allows you to access all design and programming hardware information. No additional setup is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To run SmartDebug in Integrated mode, expand **Debug Design** in the Libero Design Flow window, and then double click **SmartDebug Design**. Unlike Standalone mode, you do not have to create a project to use SmartDebug in Integrated mode.

1.1.2. Standalone Mode [\(Ask a Question\)](#)

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 create a project 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** because it requires incremental routing to connect the user net to the specified I/O.

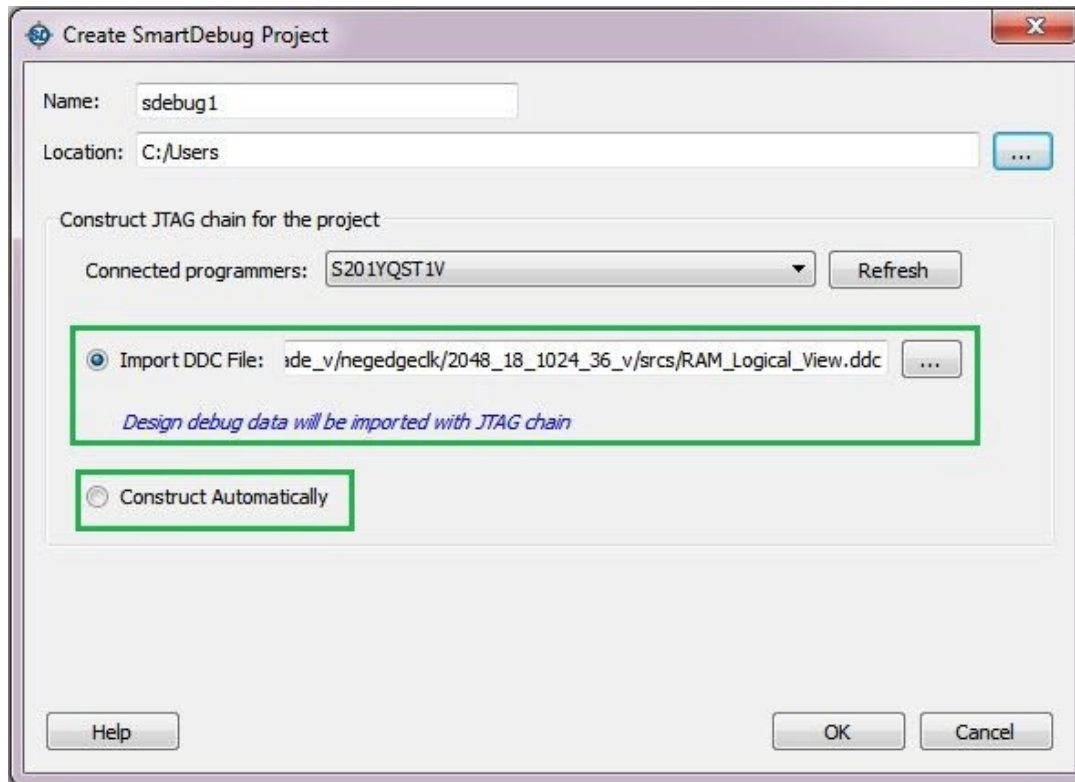
1.1.2.1. Creating a Standalone SmartDebug Project [\(Ask a Question\)](#)

Creating standalone SmartDebug projects starts with the Create SmartDebug Project dialog box. This dialog box provides two options for creating standalone SmartDebug projects:

- [Import DDC File](#) creates a standalone object from DDC files created in Libero.
- [Construct Automatically](#) creates a debug project with all the devices connected in the chain for the selected programmer.

To display the Create SmartDebug Project dialog box, from the SmartDebug main window, click **Project** and choose **New Project**.

Figure 1-1. Create SmartDebug Project Dialog Box



1.1.2.1.1. Import DDC File Created from Libero [\(Ask a Question\)](#)

If you select the **Import DDC File** option in the Create SmartDebug Project dialog box, the SmartDebug project inherits:

- The Design Debug Data of the target device and all hardware and JTAG chain information in the DDC file exported in Libero.
- The programming file information loaded onto other Microchip devices in the chain.

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, you cannot create the project. In this case, PolarFire users must run **Generate SmartDebug FPGA Array Data** under **Debug Design** in Design Flow. All users must then click **Export SmartDebug Data** under **Handle Design for Debugging** in Design Flow to export a new DDC file for use with project creation.

1.1.2.1.2. Construct Automatically [\(Ask a Question\)](#)

Selecting the **Construct Automatically** option in the Create SmartDebug Project dialog box creates a debug project with all the devices connected in the chain for the selected programmer. This is equivalent to **Construct Chain Automatically** in FlashPro.

1.1.2.1.3. Connected Programmers Drop-Down List [\(Ask a Question\)](#)

The **Connected Programmers** drop-down list in the Create SmartDebug Project dialog box shows 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 using **Programming Connectivity and Interface** in Design Flow.

1.1.3. Demo Mode [\(Ask a Question\)](#)

Demo mode provides access to the following SmartDebug features without requiring you to connect a board to the system:

- Active Probe
- Live Probe
- Memory Blocks
- Transceiver
- Debug sNVM
- Debug UPROM
- Fabric DDR
- Debug IOD

To run SmartDebug in Demo mode, start SmartDebug in either Integrated or Standalone mode when the hardware is not connected.

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 any other mode while SmartDebug is running in Demo mode.

1.2. Configuring a Generic Device [\(Ask a Question\)](#)

For Microchip devices that have the same JTAG IDCODE (that is, multiple derivatives of the same Die), configure the device type for SmartDebug to enable relevant features for debug. To configure the device, load the programming file by selecting the device using **Configure Device** in Design Flow, or import DDC files using **Programming Connectivity and Interface** in Design Flow. When the device is configured, all debug options are shown.

For debug projects created using **Construct Automatically**, 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.

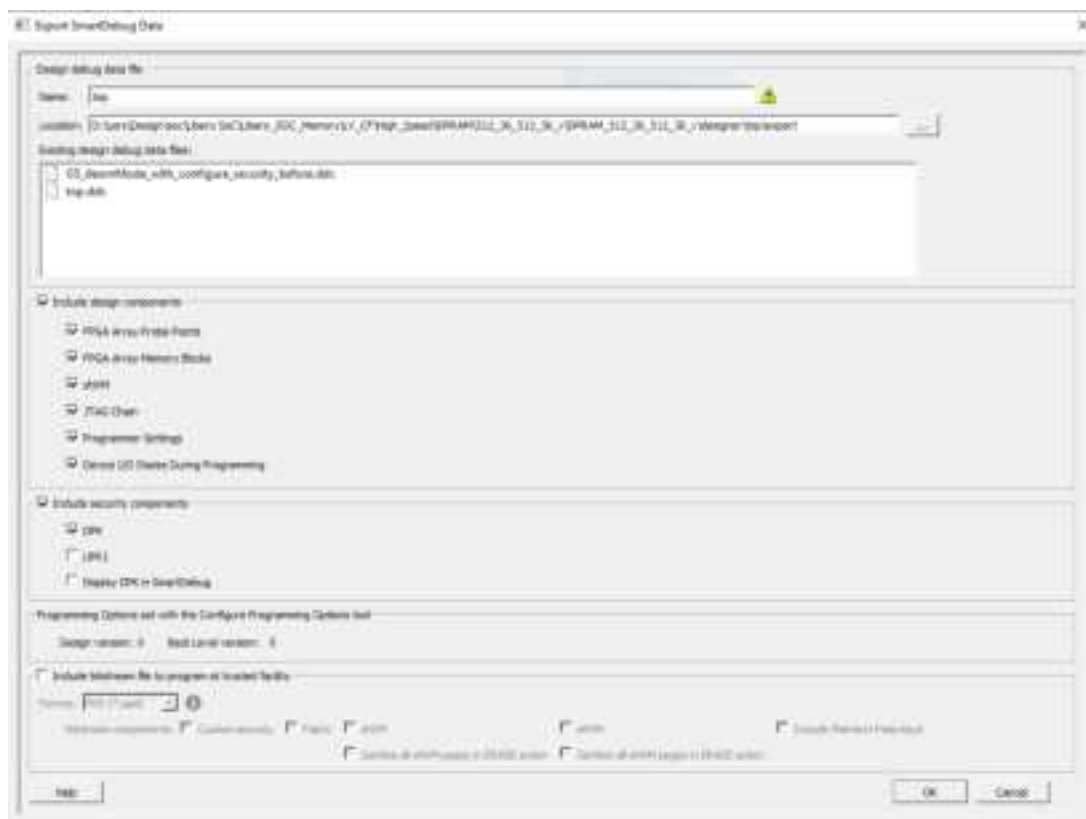
The appropriate debug features of the targeted devices are enabled after the programming file or DDC file is imported.

1.3. Debugging Devices Connected in a JTAG Chain [\(Ask a Question\)](#)

To debug the devices that are connected in JTAG chain, use the following procedure.

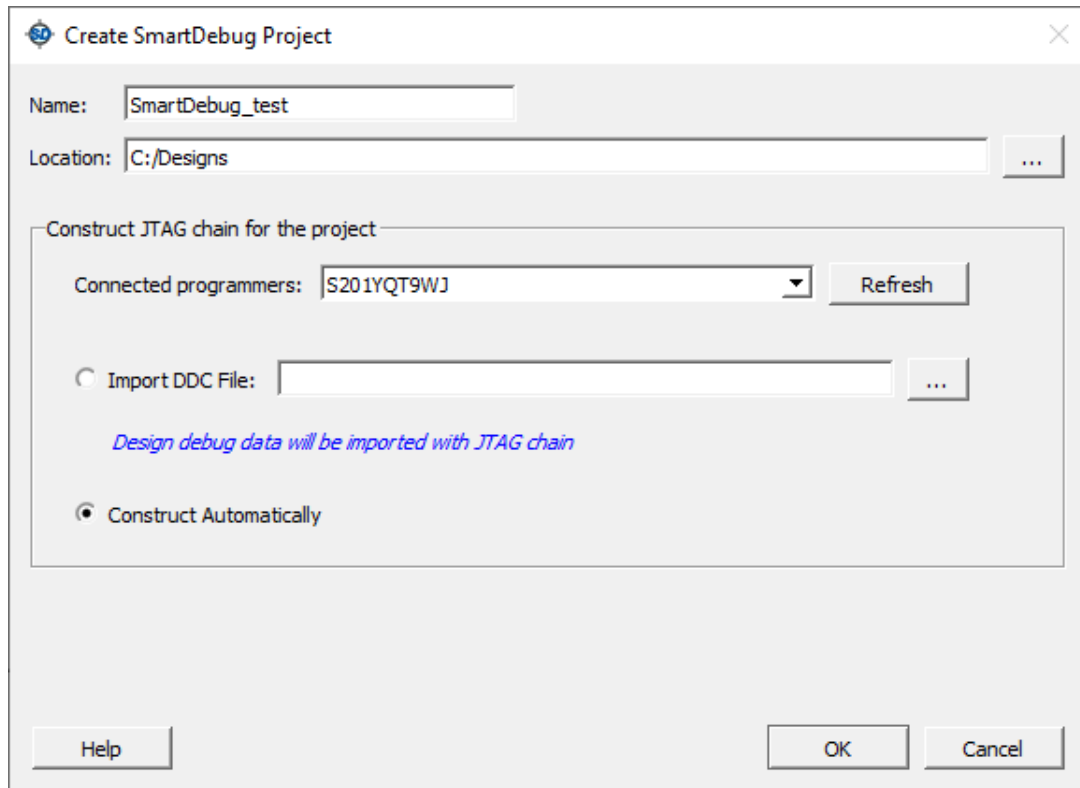
1. In Libero, under **Handoff Design for Debugging**, click **Export SmartDebug Data**.
2. When the Export SmartDebug Data dialog box appears, select the parameters shown in the following figure.

Figure 1-2. Export SmartDebug Data Dialog Box



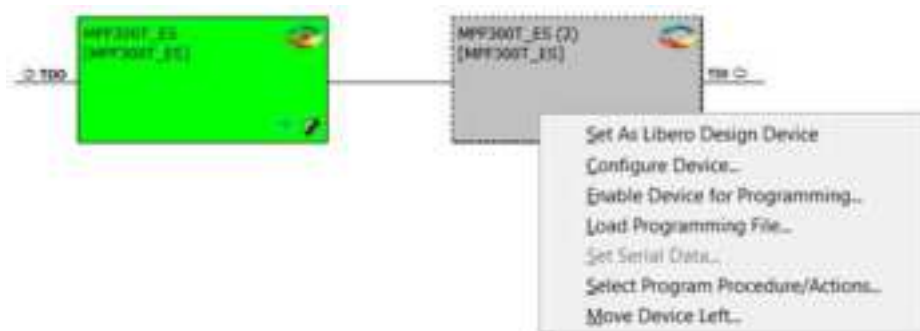
3. Start SmartDebug in stand-alone mode, and then click **Construct Automatically** in the Create SmartDebug Project dialog box to create a debug project with all the devices connected in the chain for the selected programmer.

Figure 1-3. Create SmartDebug Project Dialog Box



4. In Design Flow, open **Programming Connectivity and Interface**, enable the devices for programming, and load the DDC file exported in step 2. After the DDC file loads successfully for all the devices, program the devices.

Figure 1-4. Enable Device for Programming



5. When you finish programming, close the **Programming Connectivity and Interface**. The **Device** drop-down list in SmartDebug shows the devices connected in a JTAG chain.
6. Use the **Device** drop-down list to select the device you want to debug.


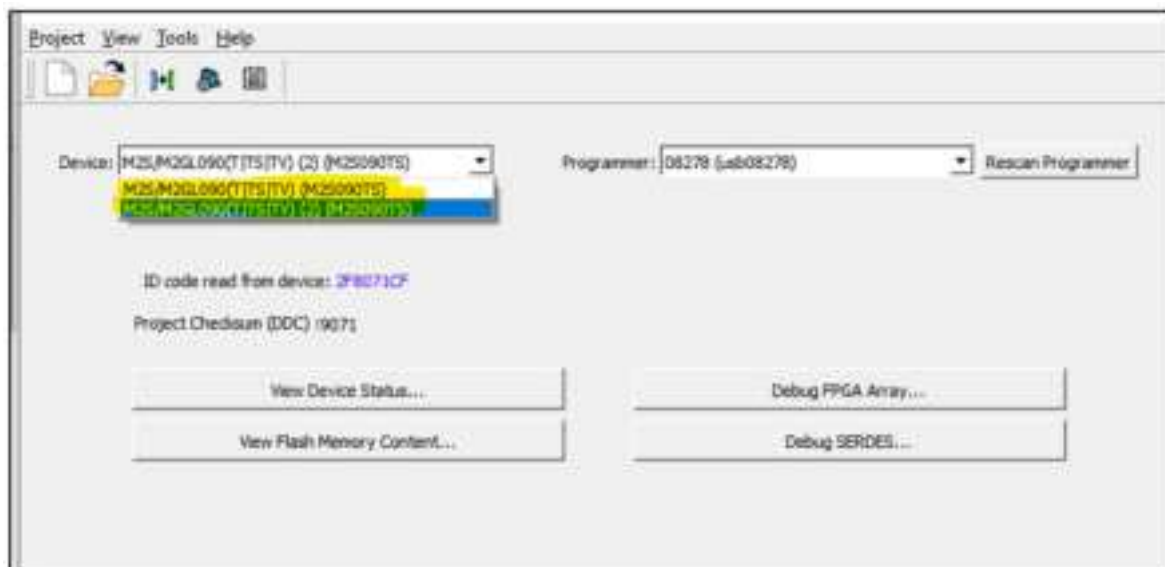
 **Important:** You can debug only one device at a time.

Figure 1-5. Device Drop-Down List



2. SmartDebug User Interface [\(Ask a Question\)](#)

This topic introduces the basic elements and features of SmartDebug.

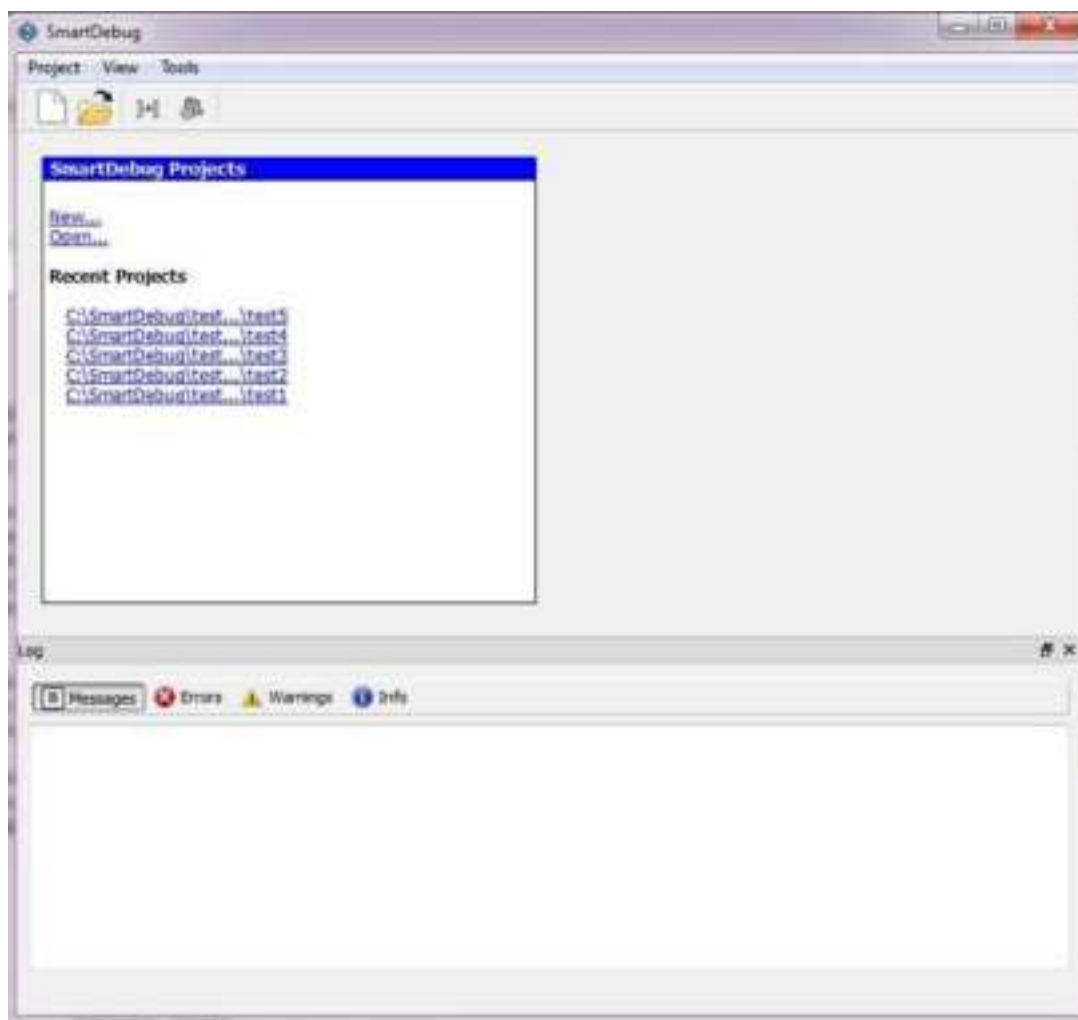
2.1. Standalone SmartDebug User Interface [\(Ask a Question\)](#)

You can launch the standalone SmartDebug from the Libero installation folder or from the FlashPro installation folder on the operating systems listed in the following table.

Table 2-1. Launching Standalone SmartDebug

Operating System	Launch Instructions
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 [\(Ask a Question\)](#)

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

Table 2-2. Project Menu

Menu Option	Description
New Project	Creates a new SmartDebug project.
Open Project	Opens an existing debug project.
Execute Script	Executes SmartDebug-specific Tcl scripts.
Export Script File	Exports SmartDebug-specific commands to a script file.
Recent Projects	Lists recent SmartDebug projects.

2.1.2. Log Window [\(Ask a Question\)](#)

The following table describes the tabs on the **Log** view in the standalone SmartDebug window. This view appears when you start SmartDebug. To hide it, click **View > View Log**.

Table 2-3. Log View Tabs

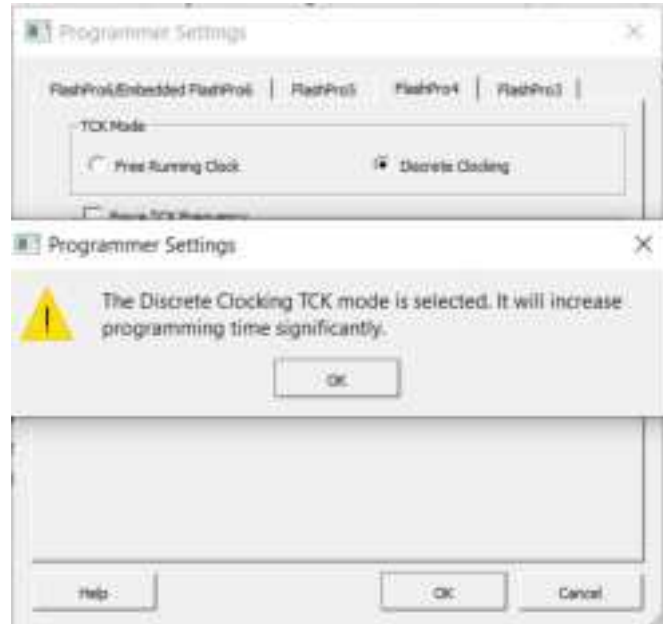
Tab	Description
Messages	Displays standard output messages.
Errors	Displays error messages.
Warnings	Displays warning messages.
Info	Displays general information.

2.1.3. Tools Menu [\(Ask a Question\)](#)

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

Attention: When the **Discrete Clocking** mode is selected, a warning pop-up appears, which states that using the Discrete Clocking TCK mode will increase the programming time significantly, as shown in the following figure.

Figure 2-2. Programmer Settings Warning Pop-Up



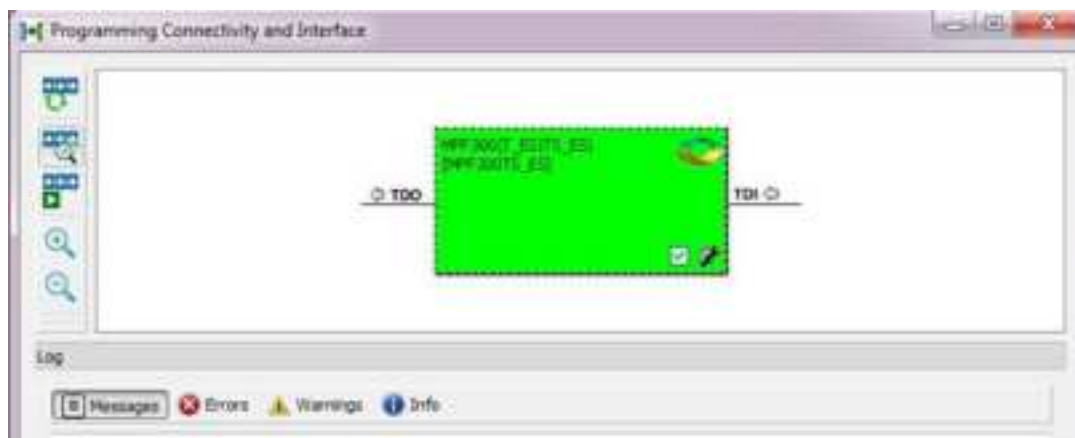
After the pop-up is closed by the user, a warning icon with a message tooltip appears next to the **Discrete Clocking** radio button. The icon and the tooltip appear when the programmer settings dialog is re-opened.

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

2.2. Programming Connectivity and Interface [\(Ask a Question\)](#)

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.

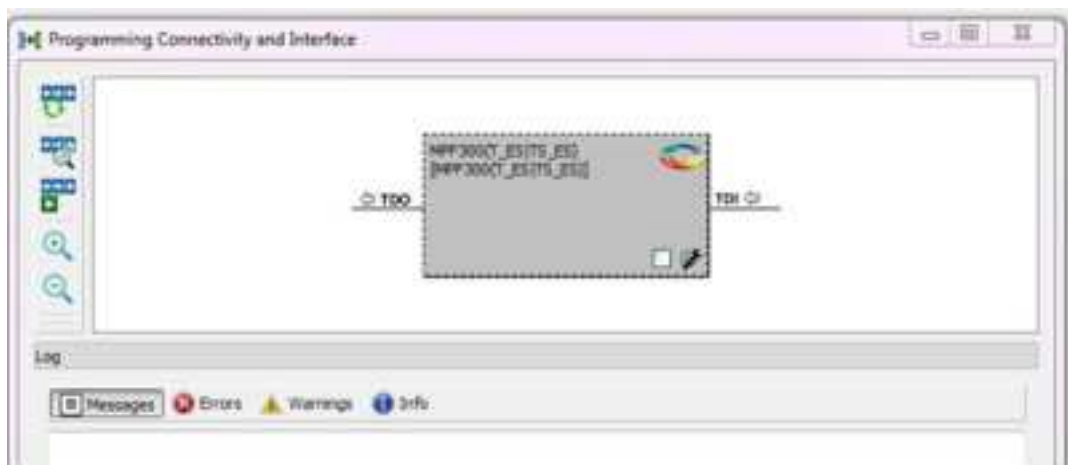
Figure 2-3. Programming Connectivity and Interface Dialog Box – Project Created Using Import from DDC File



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

Note: SmartDebug displays a pop-up window if it detects an outdated FlashPro6 programmer. You can then choose to update the programmer design.

Figure 2-4. Programming Connectivity and Interface Window – Project Created Using Construct Automatically



The following table describes the actions available in the Programming Connectivity and Interface dialog box.

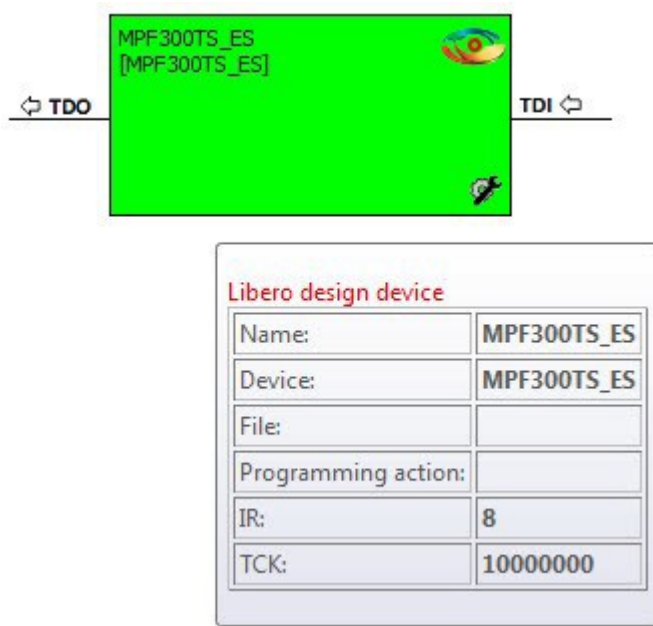
Table 2-4. Programming and Connectivity Interface Dialog Box Options

Action	Description
Construct Chain Automatically	Construct the physical chain automatically. 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	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 [\(Ask a Question\)](#)

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

Figure 2-5. Libero Design Device Information



The following table describes each Libero design device information option.

Table 2-5. 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 Properties).
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 [\(Ask a Question\)](#)

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 [\(Ask a Question\)](#)

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

Figure 2-6. Programming Connectivity and Interface - Context Menu Options

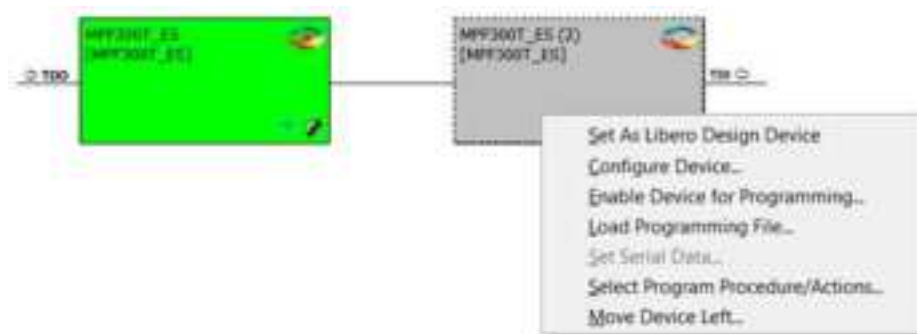


Table 2-6. Programming Connectivity and Interface - Device Context Menu Options

Option	Description
Set as Libero Design Device	The user must set the 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"> Family and Die: The device can be explicitly configured from the Family, Die drop-down. Device Name: Editable field for providing user-specified name for the device.
Enable Device for Programming	Select to enable the device for programming. Enabled devices are shown in green, and disabled devices are grayed out.
Load Programming File	Load the programming file for the selected device.
Select Programming Procedure/Actions	Option to select programming action/procedures for the devices connected in the chain. <ul style="list-style-type: none"> Actions: List of programming actions for your device. Procedures: Advanced option; enables you to customize the list of recommended and optional procedures for the selected action.
Import Debug Data from DDC File	Option to import debug data information from the DDC file. <p>Note: This option is supported when SmartDebug is invoked in Standalone mode. The DDC file selected for import into device must be created for a compatible device. When the DDC file is imported successfully, all current device debug data is removed and replaced with debug data from the imported DDC file.</p> <p>The JTAG Chain configuration from the imported DDC file is ignored in this option.</p> <p>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).</p>

2.2.4. Debug Context Save [\(Ask a Question\)](#)

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 [\(Ask a Question\)](#)

Standalone SmartDebug provides an option to select the devices connected in the JTAG chain for debug. Click the **Device** drop-down list to select the device. The device debug context is not saved when another debug device is selected.

2.3. View Device Status [\(Ask a Question\)](#)

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 2-7. Device Status Report Information

Information	Description
IDCode	IDCode read from the device under debug.
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 indicates that the device certificate may not have been installed.

Table 2-7. Device Status Report Information (continued)

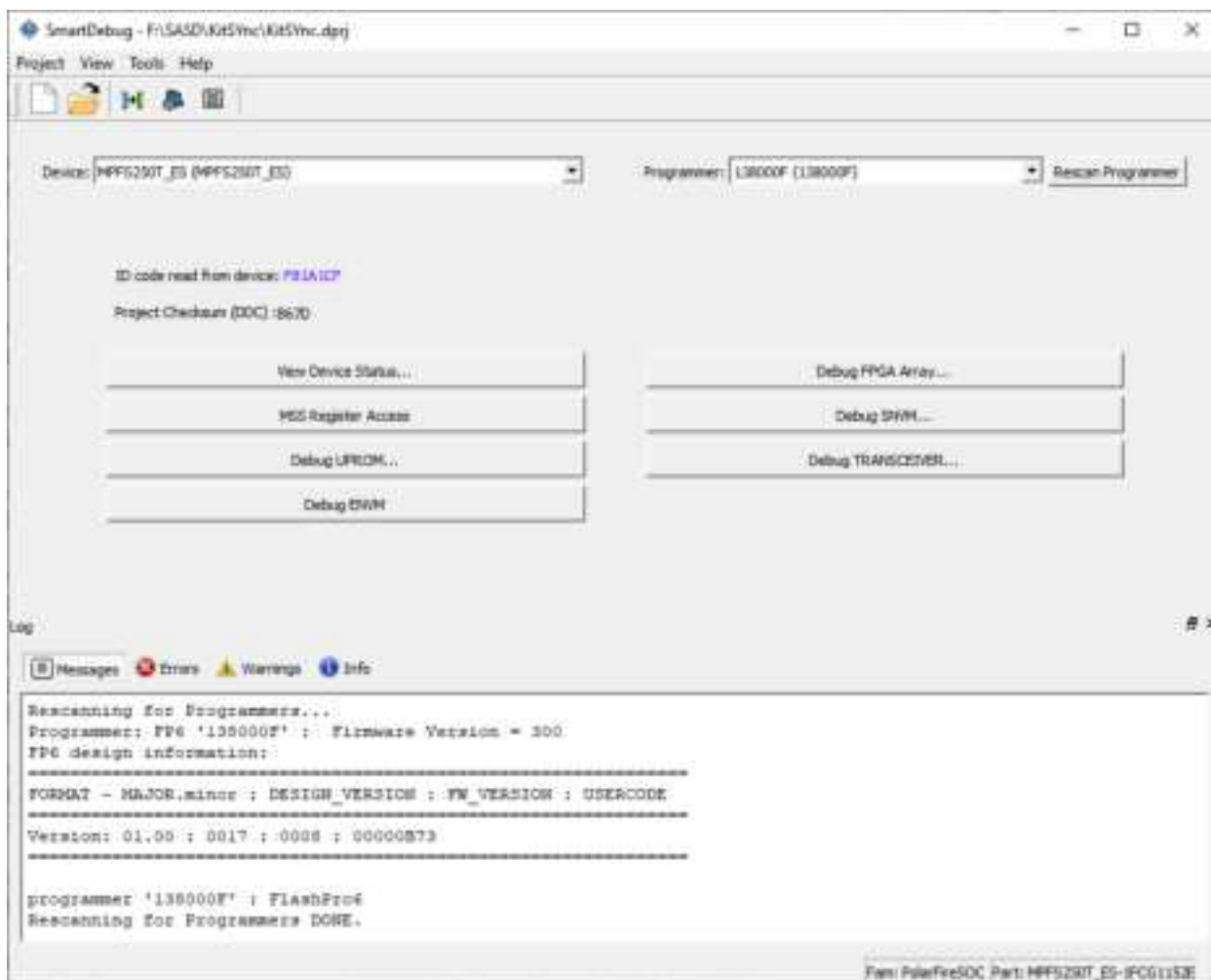
Information	Description
Design Information	Displays the following: <ul style="list-style-type: none"> • Design Name • Design Checksum • Design Version
Digest Information	Displays Fabric Digest, sNVM Digest (if applicable) computed from the device during programming. sNVM Digest is shown when sNVM is used in the design.
Device Security Settings	Displays information about your security settings, including live probes, JTAG boundary scan, global key modes, and user keys.
Programming Information	Displays the following: <ul style="list-style-type: none"> • Cycle Count: Number of times the device has been programmed since it has been out of factory reset. There is no limit to this count, but a lower threshold is around 2000 cycles. • Algorithm Version: Programming algorithm version number written to the device during programming. • Programmer: Details of the programmer hardware used during programming. • Software Version: Libero software version indicates the release version used for programming. • Programming Software: Software used for programming is FlashPro or DirectC or Non-Microchip software. • Programming Interface Protocol: Indicates the protocol followed for programming. For example, JTAG, SPI MASTER, and SPI SLAVE. • Programming File Type: Type of programming file used for programming the device. For example, STAPL, PPD, SVF, and IEEE® 532.

2.4. Rescan Programmer [\(Ask a Question\)](#)

The **Rescan Programmer** button when clicked rescans for the programmer attached to the computer. You can also click the button to rescan for programmers when they are switched or changed. The information gathered by the utility is displayed as log messages.

Note: The **Rescan Programmer** button is disabled in the Demo mode.

Figure 2-8. Rescan Programmer



An error is displayed when the programmer is detached from the computer.

Figure 2-9. Log Message When the Programmer is Detached

```
Rescanning for Programmers...
Rescanning for Programmers DONE.
Error: No programmers found. Please check the programmer connection to the computer and ensure the drivers are properly installed.
```

3. Debugging [\(Ask a Question\)](#)

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

3.1. Common Debug Elements [\(Ask a Question\)](#)

The following sections describe the debug elements common to PolarFire, SmartFusion 2, IGLOO 2, and RTG4 devices.

3.1.1. Debug FPGA Array [\(Ask a Question\)](#)

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](#)

3.1.2. Hierarchical View [\(Ask a Question\)](#)

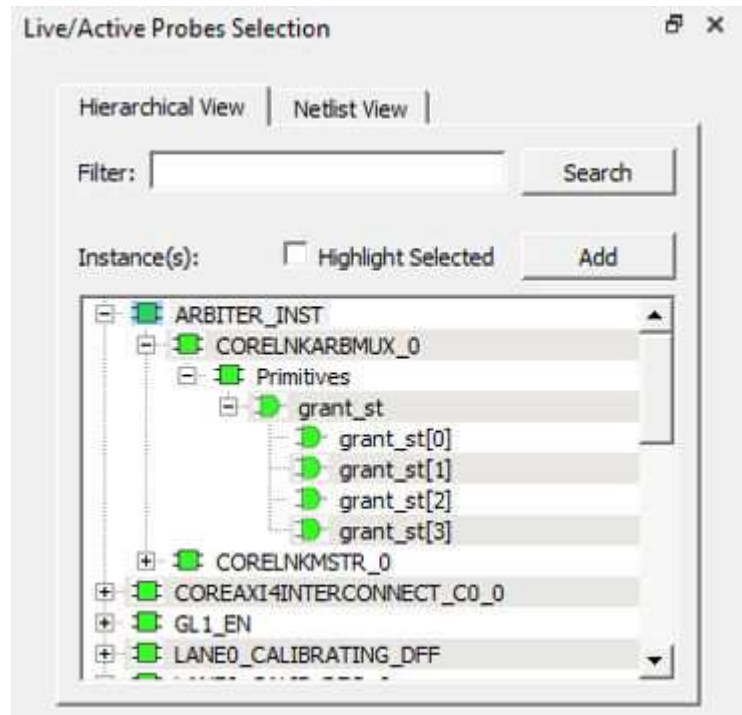
The Hierarchical View lets you view the instance level hierarchy of the design programmed on the device and select the signals to add to the Live Probes, Active Probes, and Probe Insertion tabs in the Debug FPGA Array dialog box. Logical and physical Memory Blocks can also be selected.

- **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).
- **Highlight Selected:** Check to highlight selected active, live, or probe insertion probes in their respective tabs within the Hierarchical View. By default, this check box is checked.

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

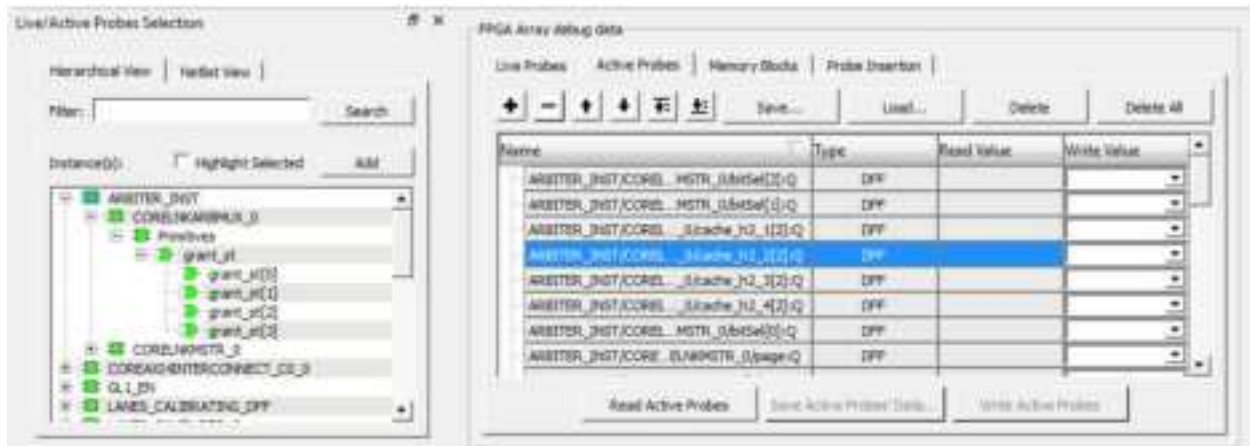


Use the **Highlight Selected** check box to show selected active, live, or probe insertion probes in the Hierarchical view.

Figure 3-2. Example of Showing Selected Probes (Highlight Selected Check Box is Checked)



Figure 3-3. Example of Not Showing Selected Probes (Highlight Selected Check Box is Not Checked)

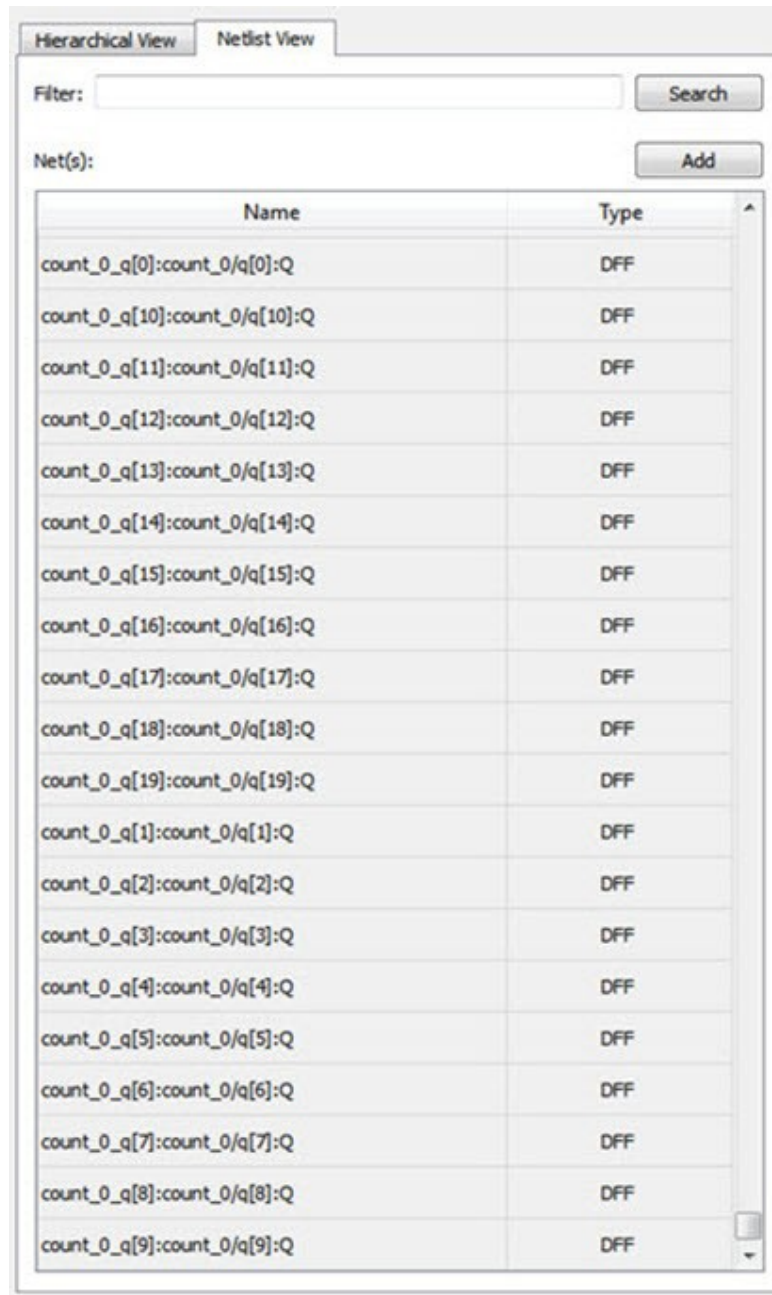


3.1.3. Netlist View [\(Ask a Question\)](#)

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 3-4. Netlist View



Name	Type
count_0_q[0]:count_0/q[0]:Q	DFF
count_0_q[10]:count_0/q[10]:Q	DFF
count_0_q[11]:count_0/q[11]:Q	DFF
count_0_q[12]:count_0/q[12]:Q	DFF
count_0_q[13]:count_0/q[13]:Q	DFF
count_0_q[14]:count_0/q[14]:Q	DFF
count_0_q[15]:count_0/q[15]:Q	DFF
count_0_q[16]:count_0/q[16]:Q	DFF
count_0_q[17]:count_0/q[17]:Q	DFF
count_0_q[18]:count_0/q[18]:Q	DFF
count_0_q[19]:count_0/q[19]:Q	DFF
count_0_q[1]:count_0/q[1]:Q	DFF
count_0_q[2]:count_0/q[2]:Q	DFF
count_0_q[3]:count_0/q[3]:Q	DFF
count_0_q[4]:count_0/q[4]:Q	DFF
count_0_q[5]:count_0/q[5]:Q	DFF
count_0_q[6]:count_0/q[6]:Q	DFF
count_0_q[7]:count_0/q[7]:Q	DFF
count_0_q[8]:count_0/q[8]:Q	DFF
count_0_q[9]:count_0/q[9]:Q	DFF

3.1.4. Live Probes [\(Ask a Question\)](#)

The same circuitry for programming the flash switches that have 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 from and writes to these DFFs, or to use an additional muxing circuit that allows two signal points to be rerouted to special pins called "channels." These signal points are I/O 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 or unassigned independently.



Attention: The **Live Probe** tab is disabled and a tooltip is shown if the user design has live probe I/Os enabled as input.

Two probe channels are available for PolarFire, SmartFusion 2, and IGLOO 2 devices. When a probe name is selected, it can be assigned to any channel. Both probes can be assigned or unassigned independently.

To assign a probe to a channel, either:

- Right click a probe in the table and choose **Assign to Channel** for PolarFire and PolarFire SoC or **Assign to probe read data pin** for RTG4.
- Click **Assign to Channel** for PolarFire and PolarFire SoC or **Assign to probe read data pin** for RTG4 to assign the probe selected in the table to the channel.

When the assignment is complete, the probe name appears to the right of the button for that channel, and SmartDebug configures the channel 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 **Unassign Channels** for PolarFire and PolarFire SoC or **Unassign probe read data pin** for RTG4 to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

Note: RTG4 devices support one probe channel.

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.

Note: Sequential elements and outputs related to LSRAM, USRAM, and MATH are supported. Combinational logic and registers related to I/O pads are not supported.

Live Probes in Demo Mode

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

3.1.5. Active Probes [\(Ask a Question\)](#)

The active probes design debug option reads and writes to one or many probe points in the design through JTAG. This option uses the same circuitry that is used for programming the fabric. It 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. The 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, as shown in the following figure.



Attention: Assign to Channel A and Unassign from Channel A RMB options are hidden in the Active Probes table, if the user design has live probe I/Os enabled as input.

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

Note: Sequential elements and outputs related to LSRAM, USRAM and MATH are supported. Combinational logic and registers related to I/O pads are not supported.

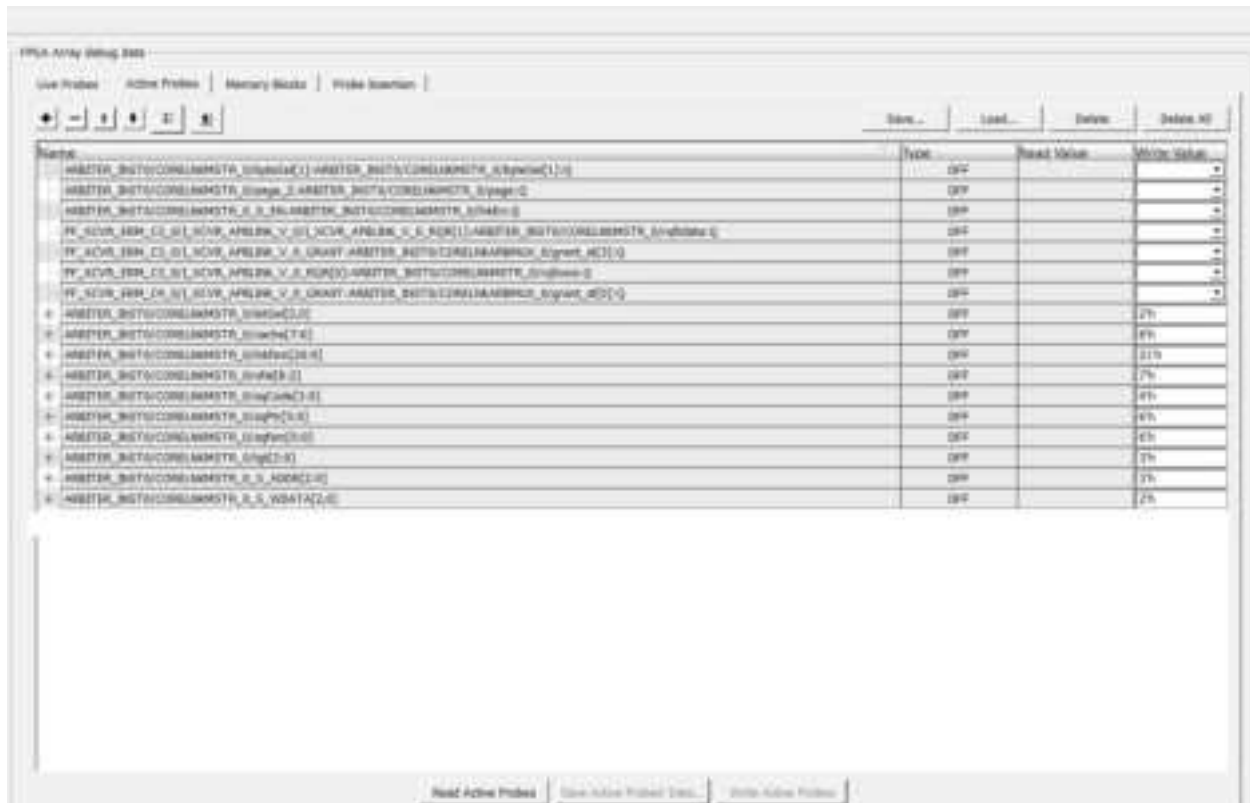
Active Probes in Demo Mode

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.

3.1.6. Probe Grouping (Active Probes Only) [\(Ask a Question\)](#)

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 3-6. 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)

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.

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

3.1.6.1. Context Menu of Probe Points Added to the Active Probes UI [\(Ask a Question\)](#)

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

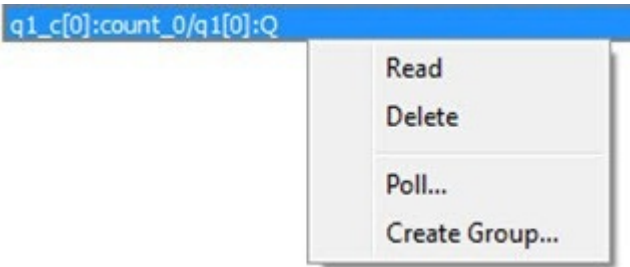
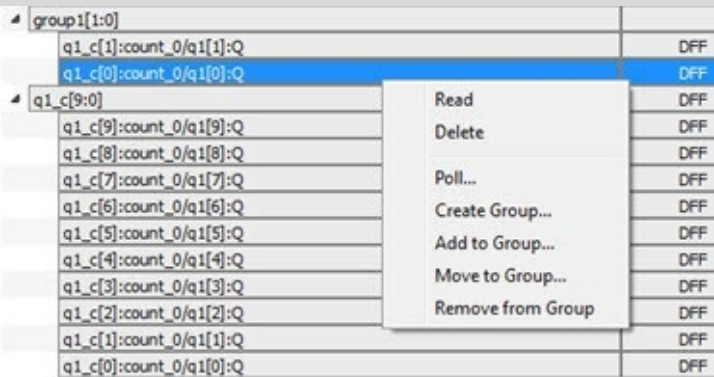
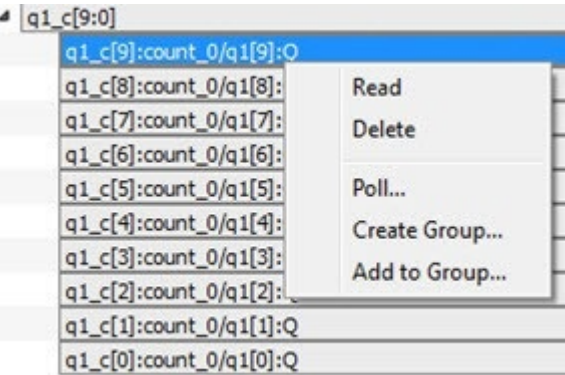
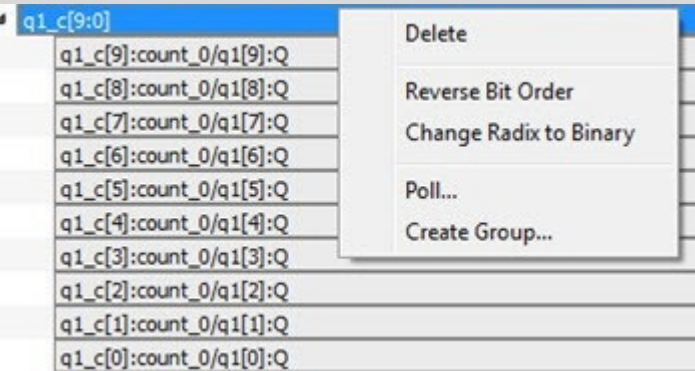
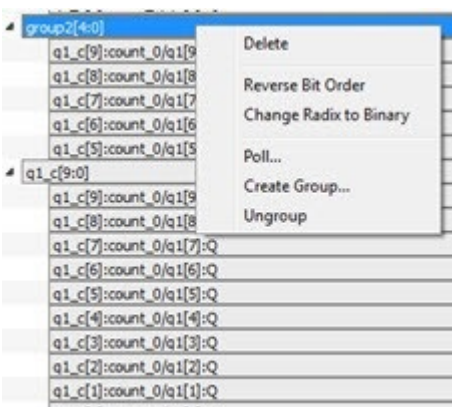
Situation	Options	User Interface
For individual signals that are not part of a probe group or bus	<ul style="list-style-type: none"> Read Delete Poll Create Group 	
For individual signals in a probe group	<ul style="list-style-type: none"> Read Delete Poll Create Group Add to Group Move to Group Remove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

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

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

3.1.6.2. Differences Between a Bus and a Probe Group [\(Ask a Question\)](#)

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

3.1.7. Memory Blocks [\(Ask a Question\)](#)

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. You determine the depth and width of blocks shown in the logical view in SmartDesign, RTL, or IP cores using memory blocks.

Notes:

- You cannot access RAM when SmartDebug accesses RAM blocks.
- You cannot access RAM is not accessible 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.



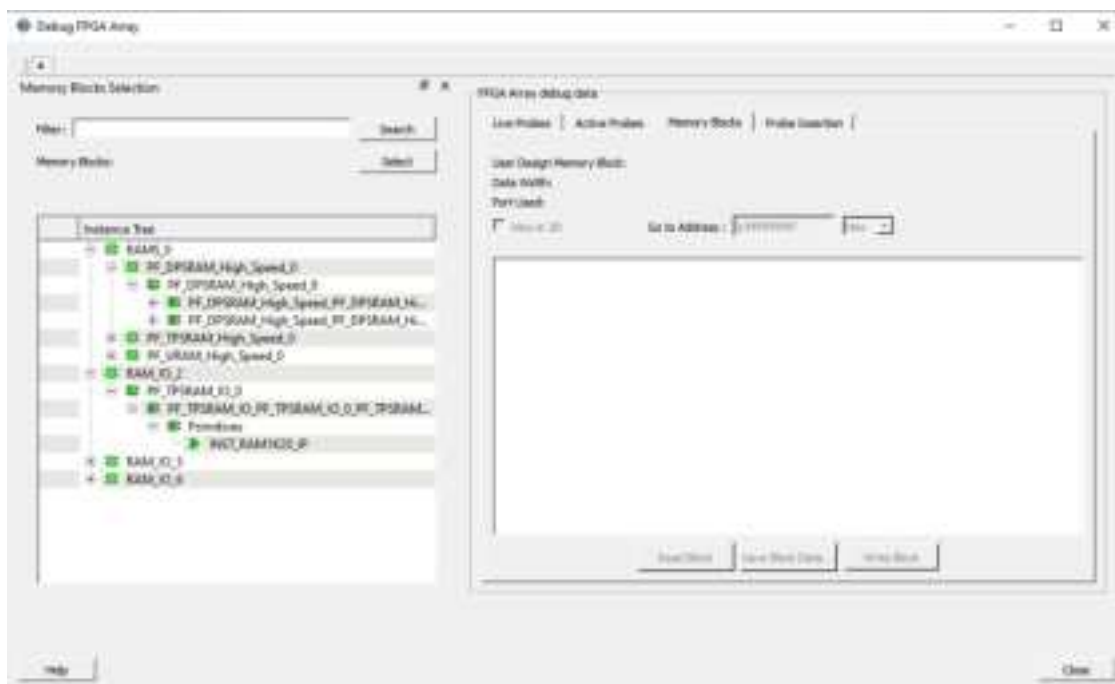
The following figure shows the hierarchical view of the **Memory Blocks** tab. You can view logical blocks and physical blocks. Logical blocks are shown with an **L** (), and physical blocks are shown with a **P** ().

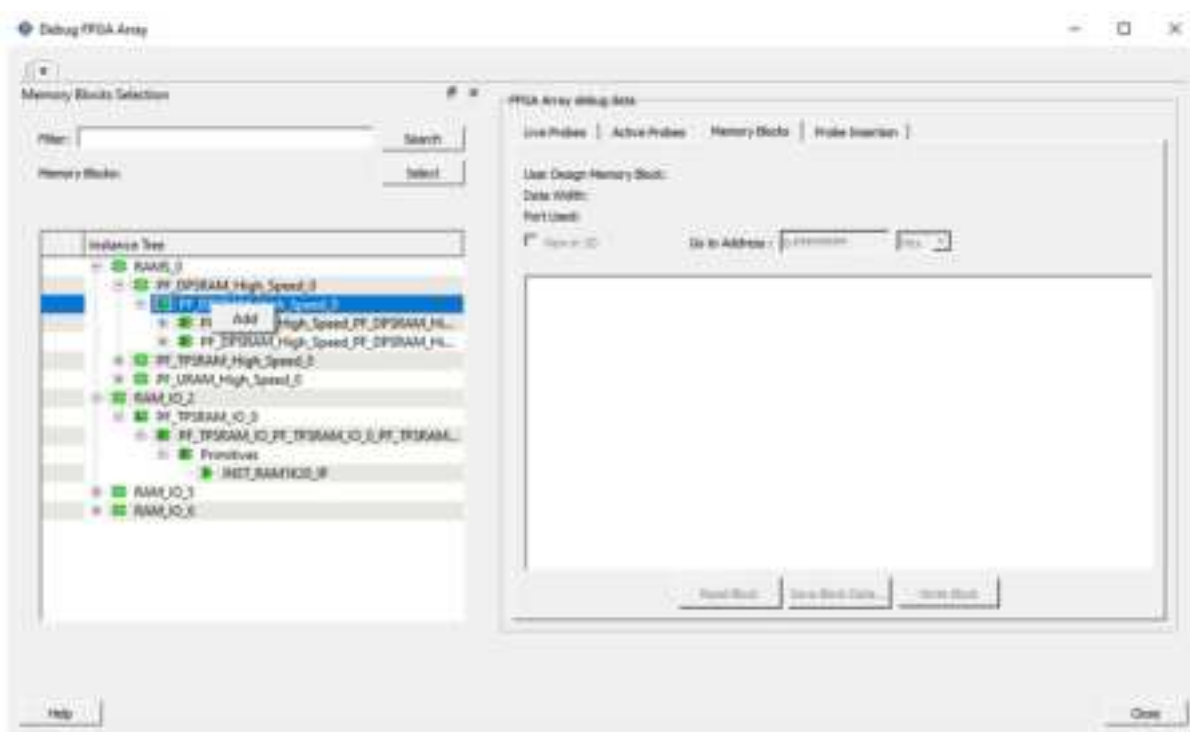
Figure 3-7. Memory Blocks Tab - Hierarchical View



You can select only 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 3-8. Adding a Memory Block



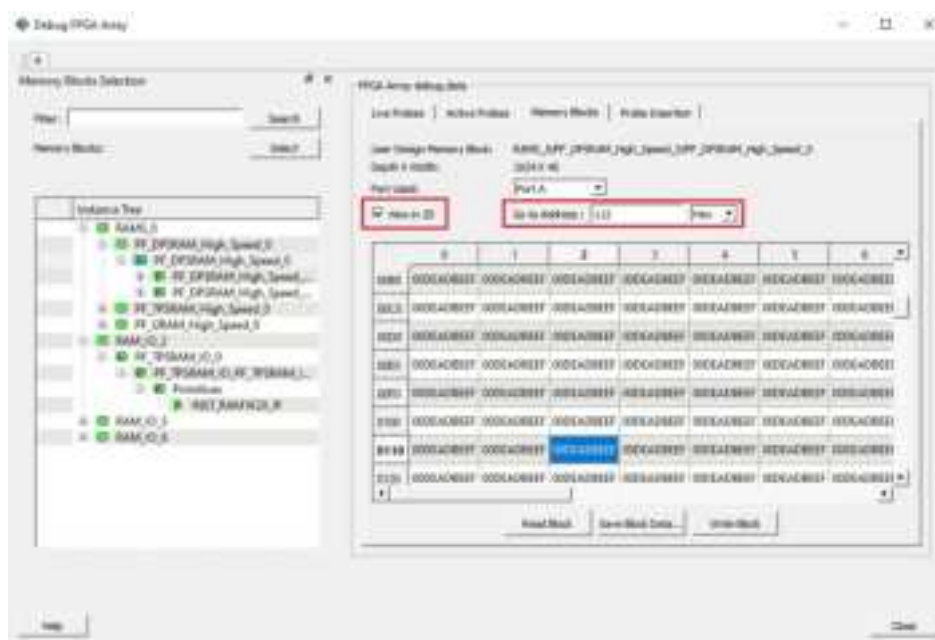
- Click a name in the list and then click **Select**.
- Click a name, drag it to the right, and drop it into the **Memory Blocks** tab.

- 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 a **P** icon can be selected in the hierarchical view.

A **View in 2D** check box allows you to toggle between a 1-dimensional or 2-dimensional view of the memory block (default is 2 dimensional). The **Go to Address** option to the right of the check box allows you to navigate to an offset address location in either hexadecimal or decimal format.

Figure 3-9. View in 2D and Go to Address Options



3.1.7.1. Memory Block Fields [\(Ask a Question\)](#)

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 the 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.

Values vary with the device family. For PolarFire devices, if the block is physical, the value of "Depth × Width" is 64 × 12 for μ SRAM blocks, 16384 × 1, 8192 × 2, 4096 × 5, 2048 × 10, 1024 × 20 for LSRAM blocks, and 512 × 40 (512 × 33 if Error Correcting Code is enabled where 512 × 7 is dedicated for ECC) for Two-Port RAM(TPSRAM) physical block. Corresponding values for other Microchip devices can be found in their respective datasheets.

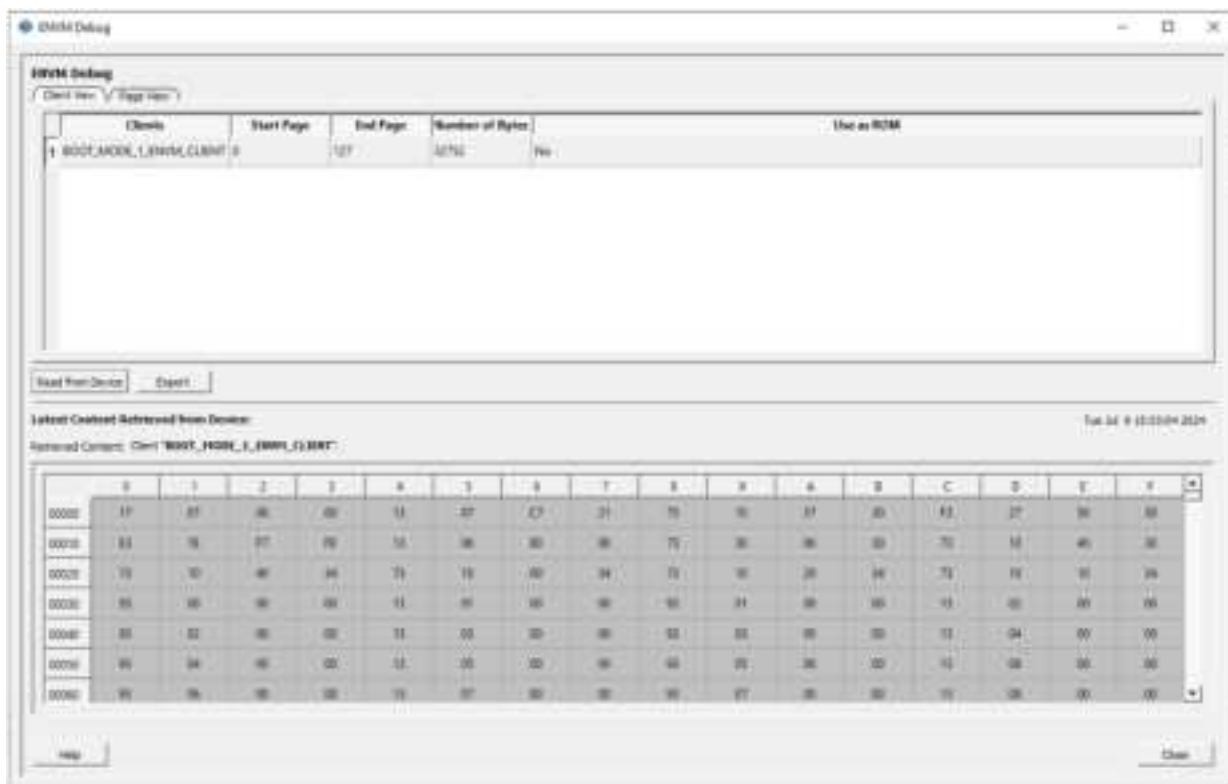
Port Used

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

Memory Block Views

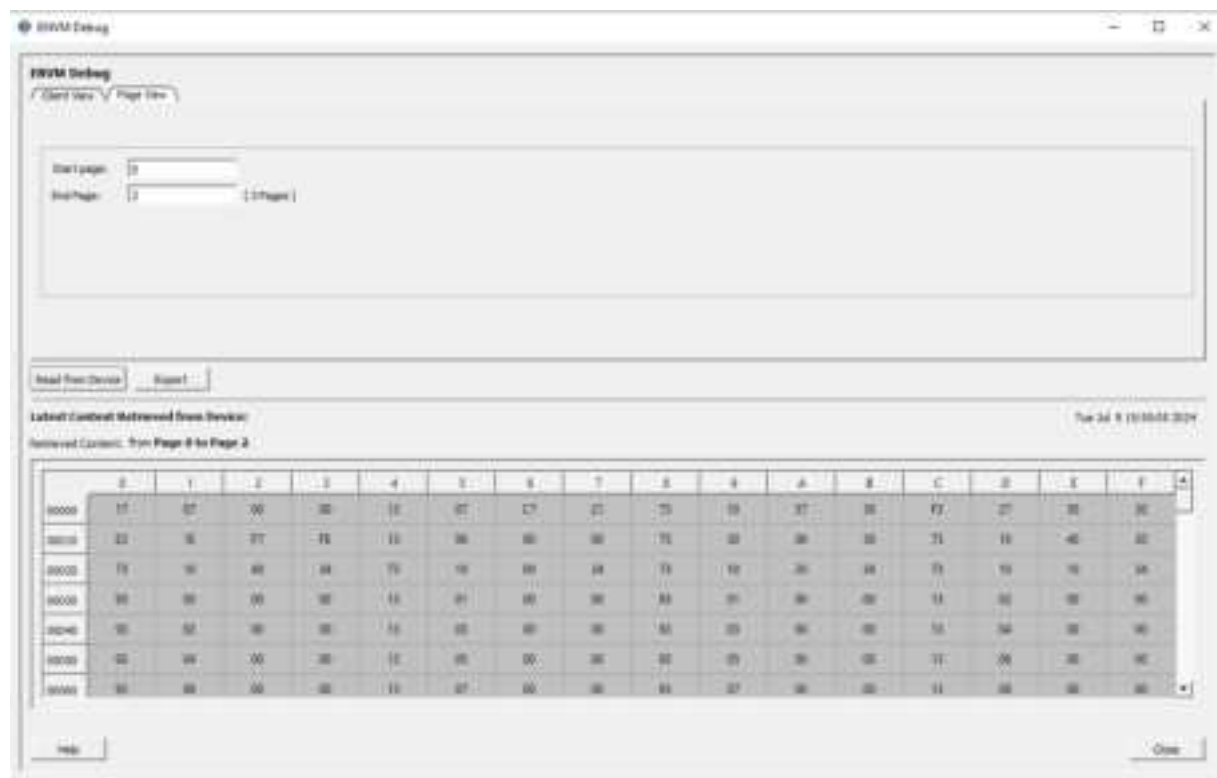
The following figure shows the **Memory Blocks** tab fields for a logical block view.

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



3.1.7.2. Read Block [\(Ask a Question\)](#)

Memory blocks can be read once they are selected. If the block name appears on the right 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 (1D View)

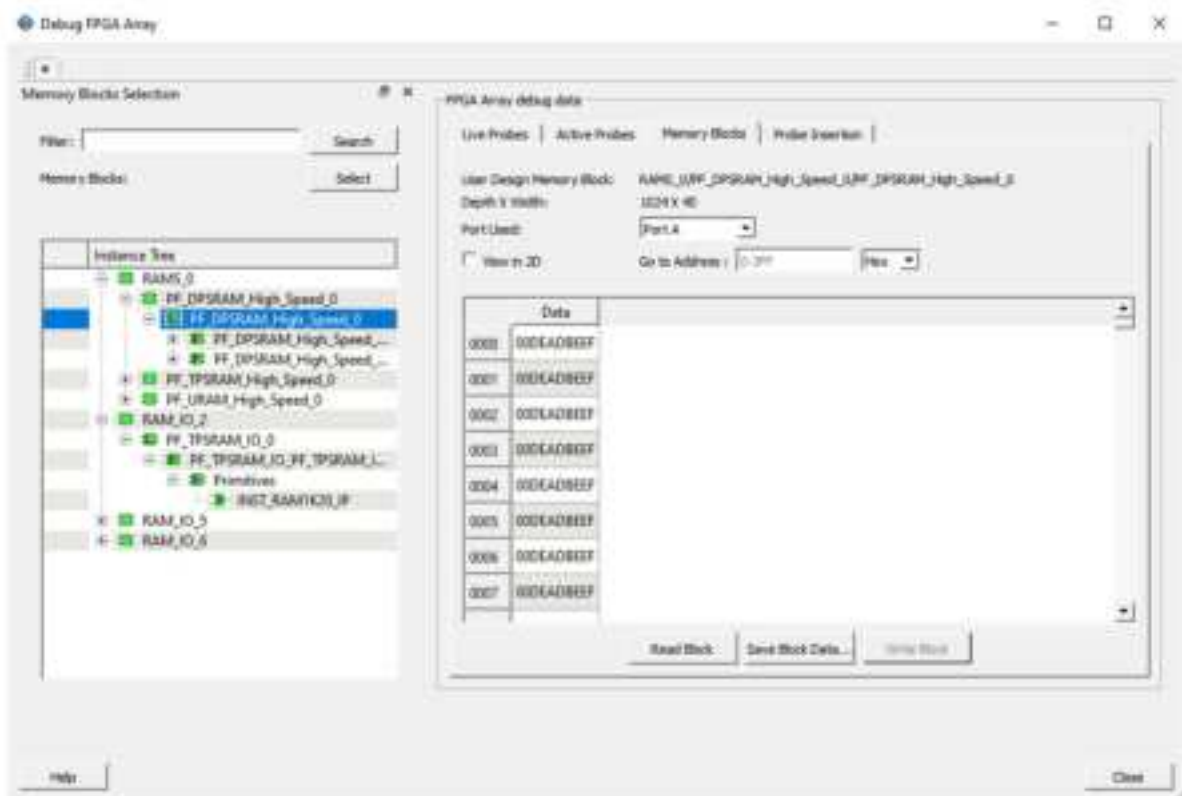
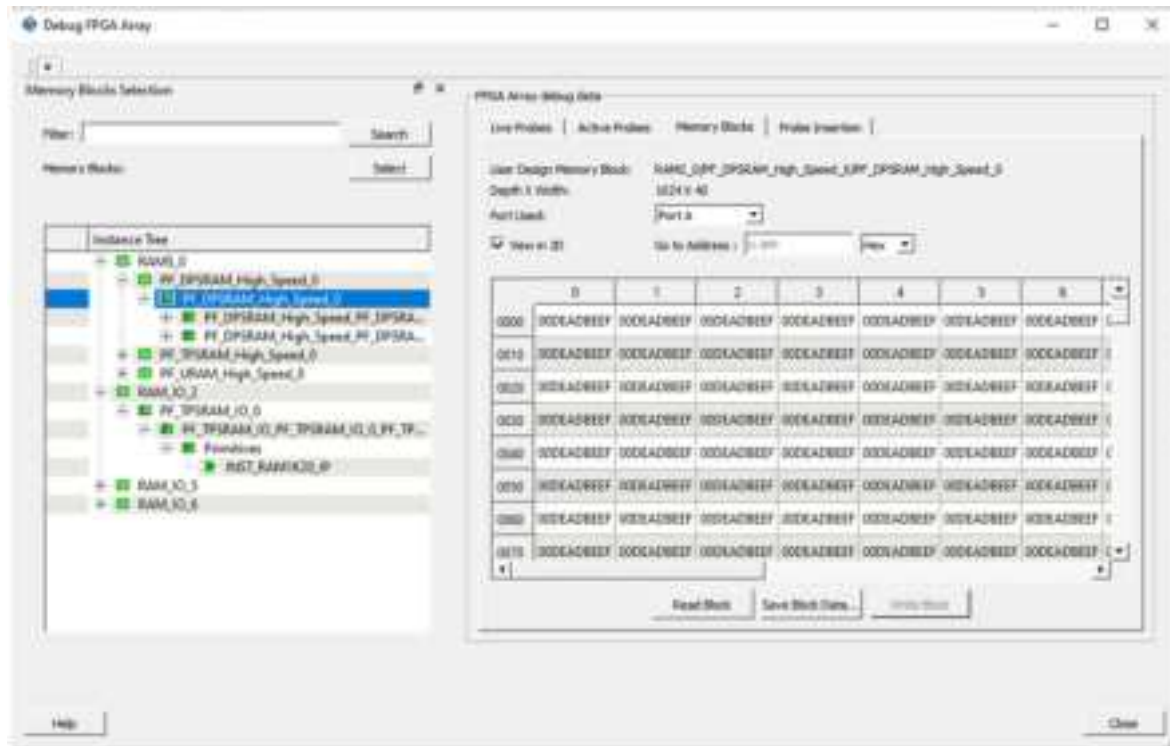


Figure 3-13. Logical Block Read (2D View)



The data shown is in hexadecimal format. In the preceding figure, data width is 40. Because each hexadecimal character has 4 bits of information, you can see 10 characters corresponding to 40 bits. Each row has 16 locations (shown in the column headers), which are numbered in hexadecimal from 0 to F.

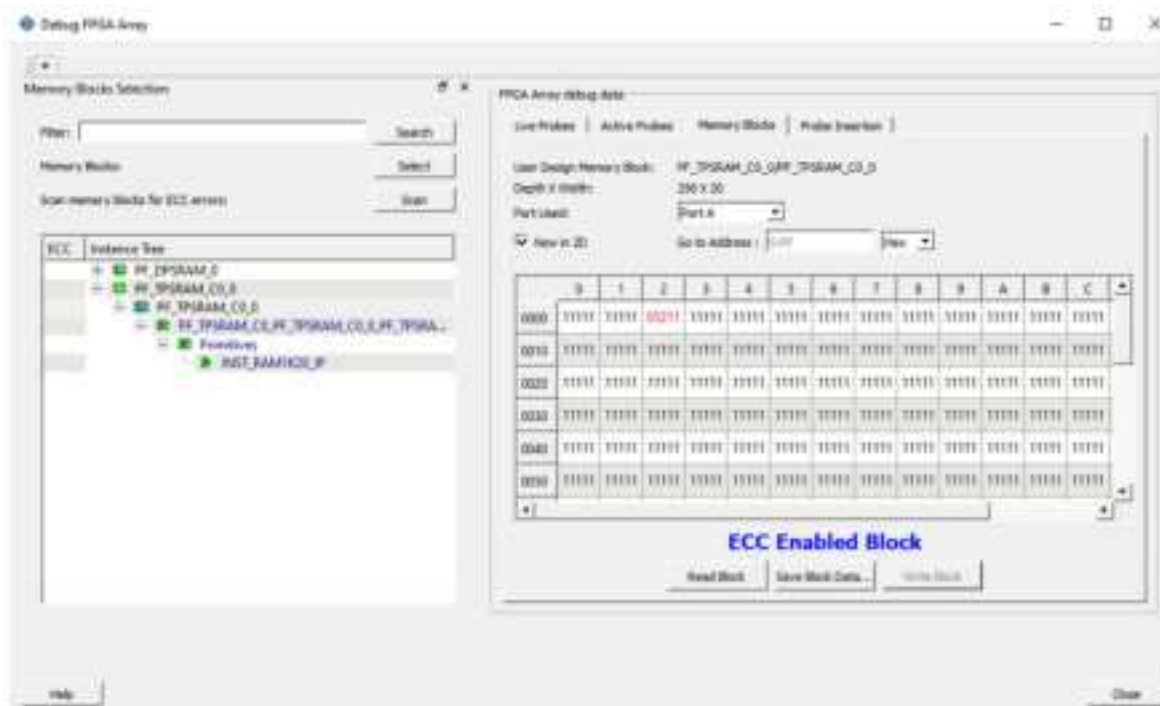
Notes: For all logical blocks that cannot be inferred from physical blocks, the corresponding icon does not contain a letter. The following are the scenarios where logical block cannot be constructed:

- If the inference flow (RTL – synthesis) is used in the design, the inference guidelines provided by Synopsys have to be followed.
- Because few IPs deviate from inference guidelines, no logical reconstruction is supported.
- If your design has output pins of RAM (A_DOUT and B_DOUT) that have been partially promoted to top, physical blocks corresponding to remaining pins may be optimized. As a result, reconstruction may not be done.

Logical Block Read for ECC-Enabled Blocks (PolarFire, PolarFire SoC, and RTG4)

Two-Port RAMs on PolarFire, PolarFire SoC, and RTG4 devices support Error Correcting Code (ECC) that provides Single Error Correction and Double Error Detection (SECD). The 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. After the logical block is read, all the physical block data is recorded for navigating to the respective physical block, so you can view the respective physical block without having to read from the device again.

Figure 3-14. Logical Block Read - ECC Enabled



Physical Block Read

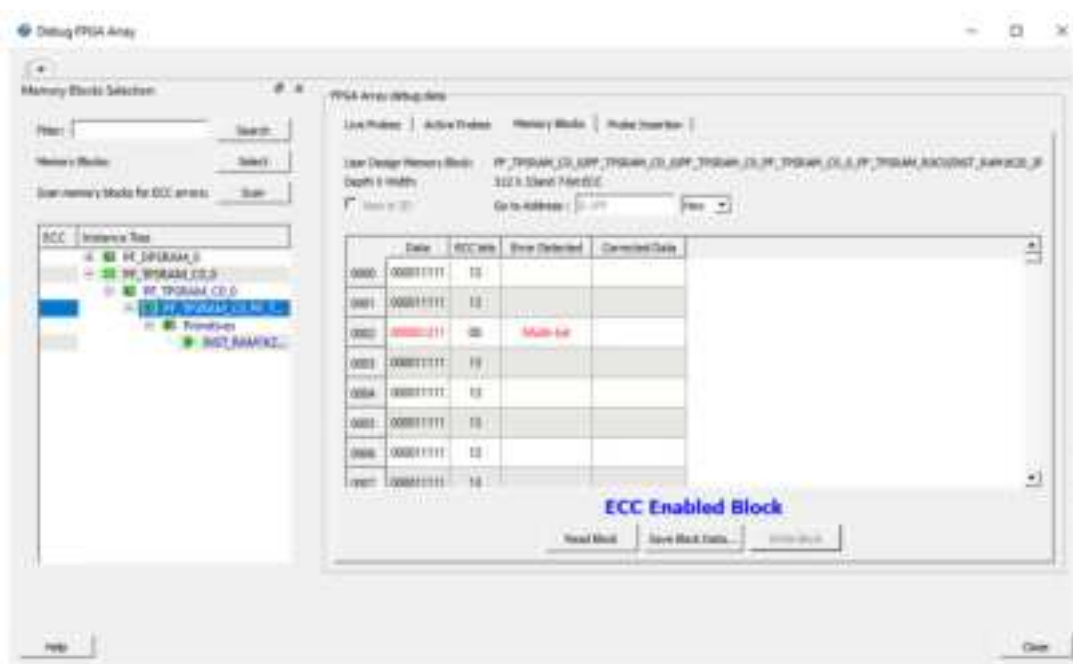
When a physical block is selected, only the **User Design Memory Block** and **Depth X Width** fields are shown.

Figure 3-15. Physical Block Read (1D View)



Figure 3-16. Physical Block Read (2D View)**Physical Block Read for ECC-Enabled Blocks**

Instead of data being shown in a matrix form, ECC-enabled blocks are shown with data internally spread across Port A and Port B. As a result, 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.

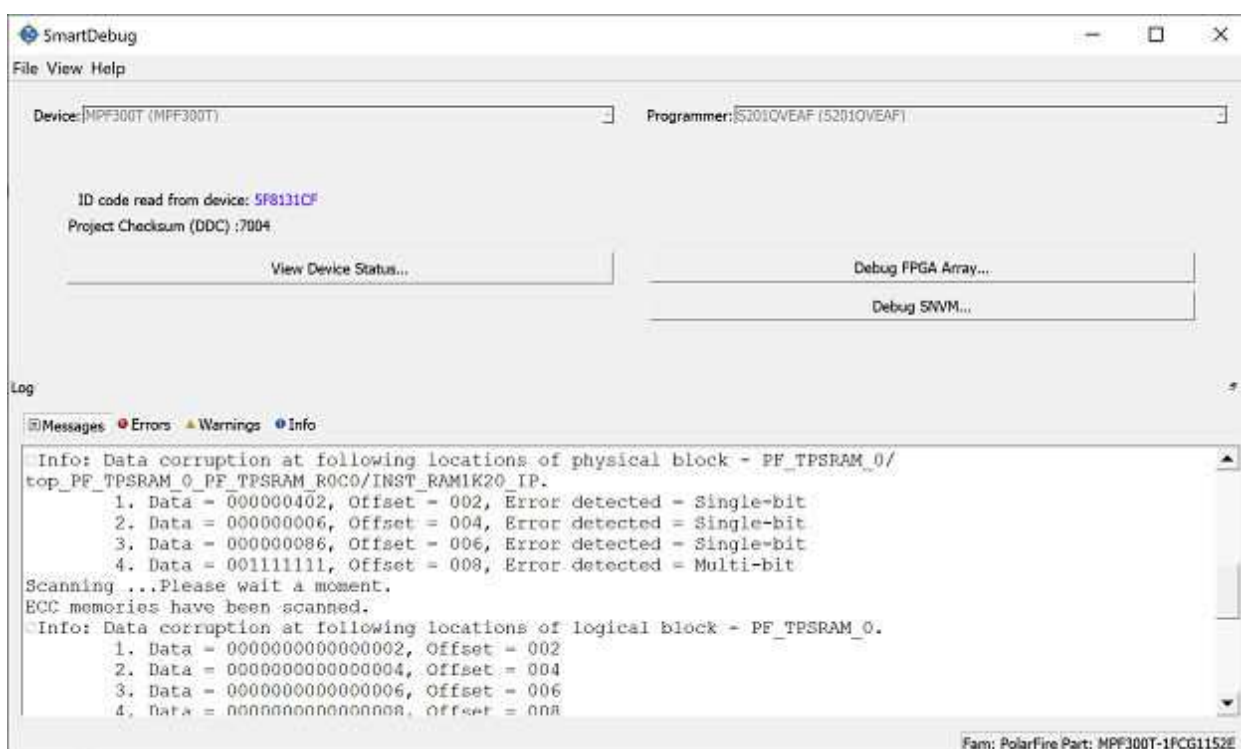
Figure 3-17. Physical Block Read - ECC Enabled

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	PolarFire and PolarFire SoC devices: 33-bit physical block data offset value. RTG4 devices: 18-bit physical block data offset value.
ECC Bits	PolarFire and PolarFire SoC devices: 7-bit ECC value. RTG4 devices: 6-bit ECC value.
Error Detected	Displays identified error type. The error type value displayed can be either Single-Bit or Multi-Bit .
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 into the device. Right click on the corrected data to view an option to copy the data automatically to the Data cell.

After 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 when navigating from physical to logical block view.

Figure 3-18. Log Info

3.1.7.3. Write Block [\(Ask a Question\)](#)

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.

The screenshot displays the Xilinx Vivado IDE interface. On the left, the 'Memory Block Selection' window is open, showing a tree view of memory blocks. The 'RAM_0' block is selected, and its sub-components are listed, including 'RAM_0_0', 'RAM_0_1', 'RAM_0_2', 'RAM_0_3', 'RAM_0_4', 'RAM_0_5', and 'RAM_0_6'. The 'RAM_0_0' block is highlighted.

On the right, the 'FPGA Array Debug Data' window is open. It shows the 'Live Probes' tab, which displays the 'User Design Memory Block' as 'RAM_0_0' and 'RAM_0_1'. The 'Depth x Width' is '32K x 40'. The 'Port Used' is 'Port A'. The 'View in 2D' button is active. The 'Go to Address' field is set to '0x00000000'. The 'Hex' button is active.

The 'FPGA Array Debug Data' window also displays a table of memory addresses and their corresponding values. The table has 7 columns, labeled 0 through 6. The rows show memory addresses from 0x00000000 to 0x00000006. The value for address 0x00000000 is 0x00000000, which is highlighted with a red box.

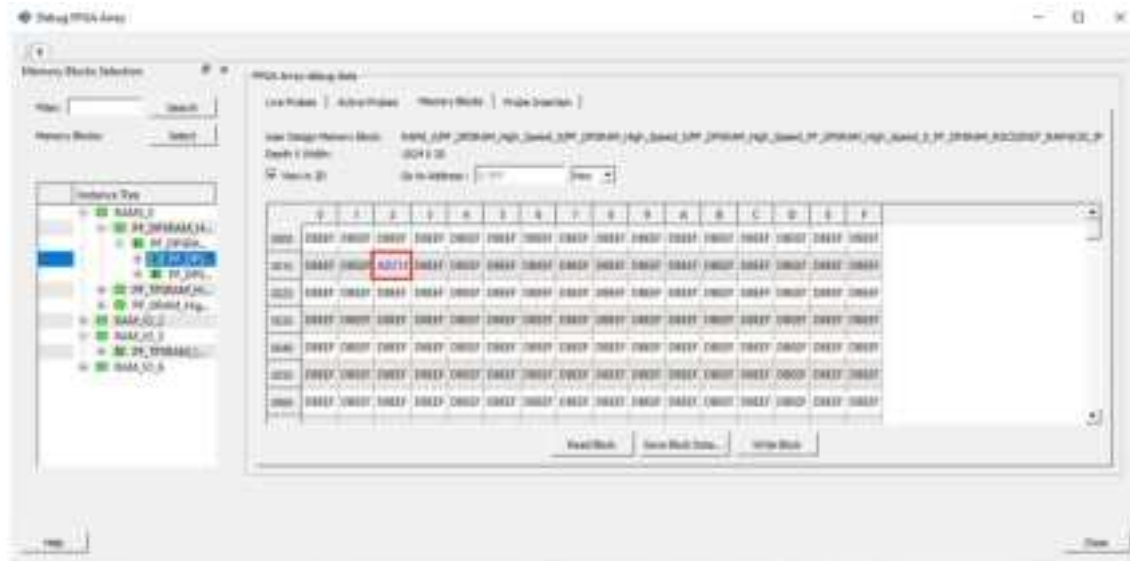
	0	1	2	3	4	5	6
0x00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000002	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000003	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000004	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000005	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0x00000006	00000000	00000000	00000000	00000000	00000000	00000000	00000000

At the bottom of the 'FPGA Array Debug Data' window, there are three buttons: 'Read Block', 'Save Block Data...', and 'Write Block'.

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

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, a message shows the range of permitted values that can be entered.

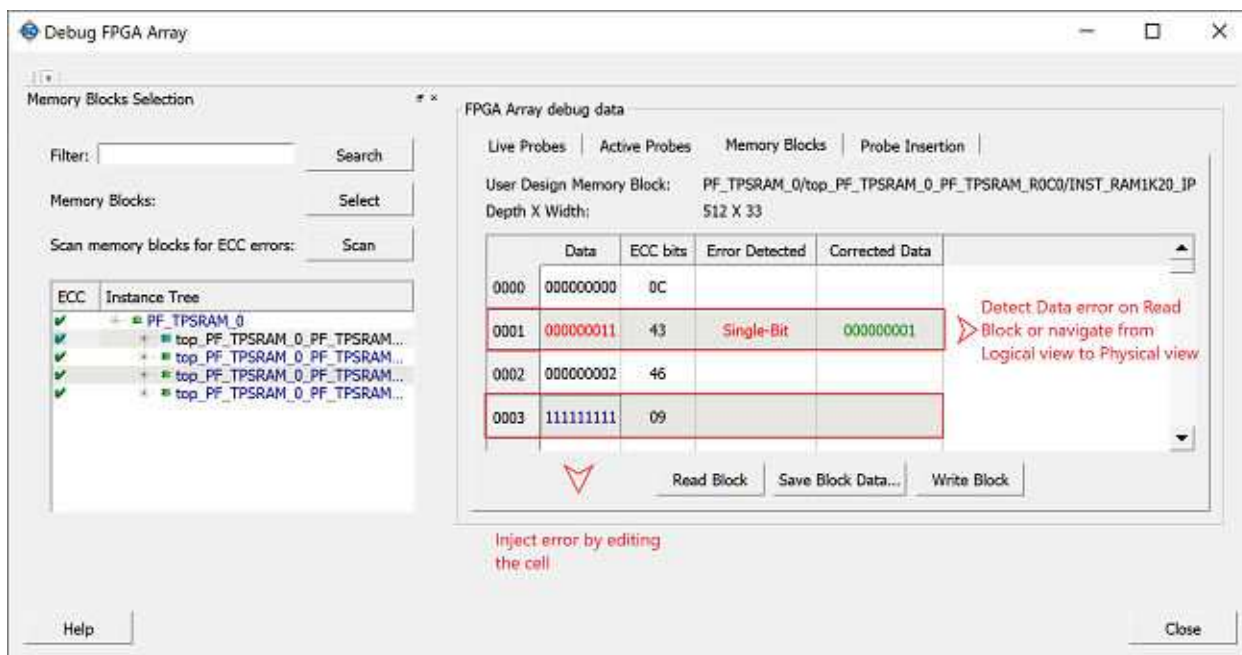
Figure 3-21. Physical Block Write



Physical Block Write for ECC-Enabled Blocks

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

Note: If corrupted data is written onto a physical block location, the logical block is outdated. Navigating to the logical view updates the locations of corrupted data after you click **Read Block**.

Figure 3-22. Inject Error - Physical Block Write for ECC-Enabled Blocks

Limitation of Injecting Errors

Notes: Error injection is discretionary. Generally, the ECC algorithm has limitations when it comes to Multi-bit detection. Data corrupted beyond 3 or 4 bits might result in 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 that ECC calculated originally.

3.1.7.4. Scan Memory [\(Ask a Question\)](#)

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 indicates that the data is corrupted.

Figure 3-23. Scan Memory Blocks for ECC Errors



3.1.7.5. Unsupported Memory Blocks [\(Ask a Question\)](#)

If RTL is used to configure memory blocks, it is recommended that you follow the Microchip RAM block inference guidelines in the application note for your Microchip device (see Related Application Notes below).

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

Related Application Notes

- PolarFire: [Inferring Microchip PolarFire RAM Blocks Application Note](#)
- RTG4: [Inferring Microchip RTG4 RAM Blocks Application Note](#)
- SmartFusion 2: [Inferring Microchip SmartFusion 2 RAM Blocks Application Note](#)

3.1.7.6. Memory Blocks in Demo Mode [\(Ask a Question\)](#)

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.

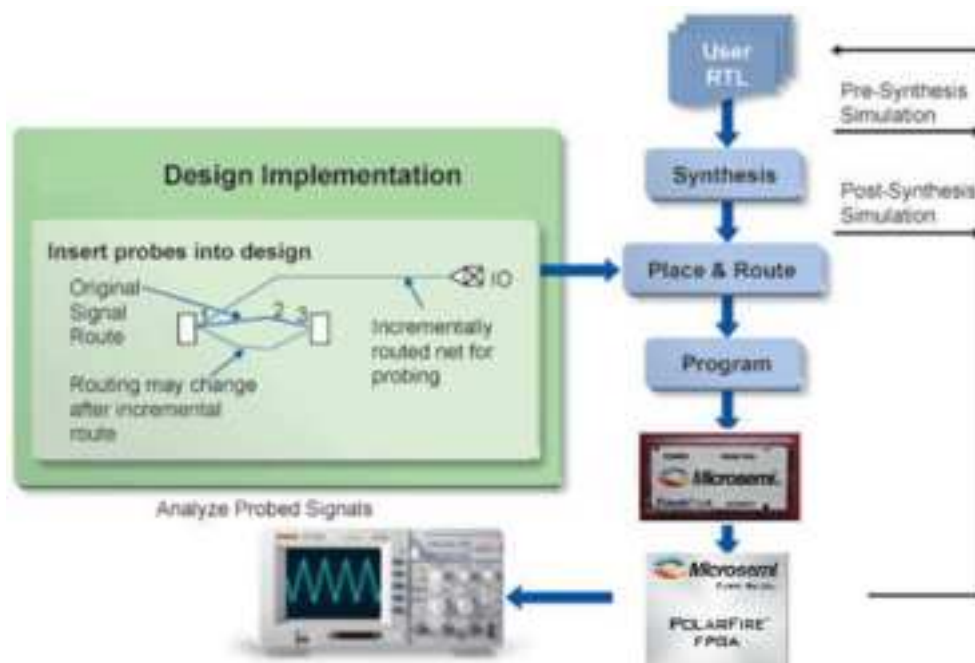
Both physical block view and logical block view are supported.

3.1.8. Probe Insertion (Post-Layout) [\(Ask a Question\)](#)

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

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

Figure 3-24. Probe Insertion in the Design Process



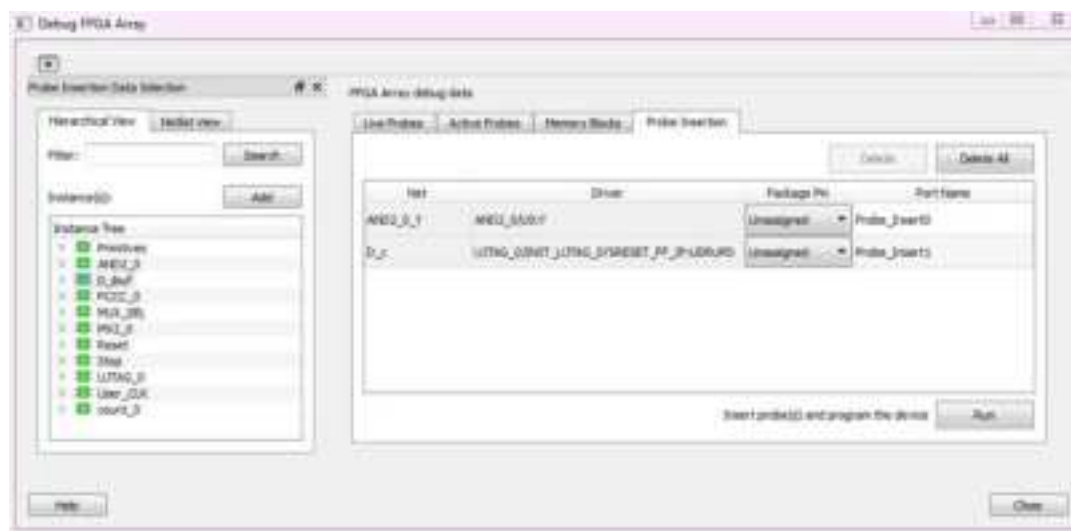
The Probe Insertion debug feature is complementary to Live Probes and Active Probes. Live Probes and Active Probes use a special dedicated probe circuitry.

3.1.8.1. Inserting Probe and Programming the Device [\(Ask a Question\)](#)

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 3-25. Probe Insertion Tab



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

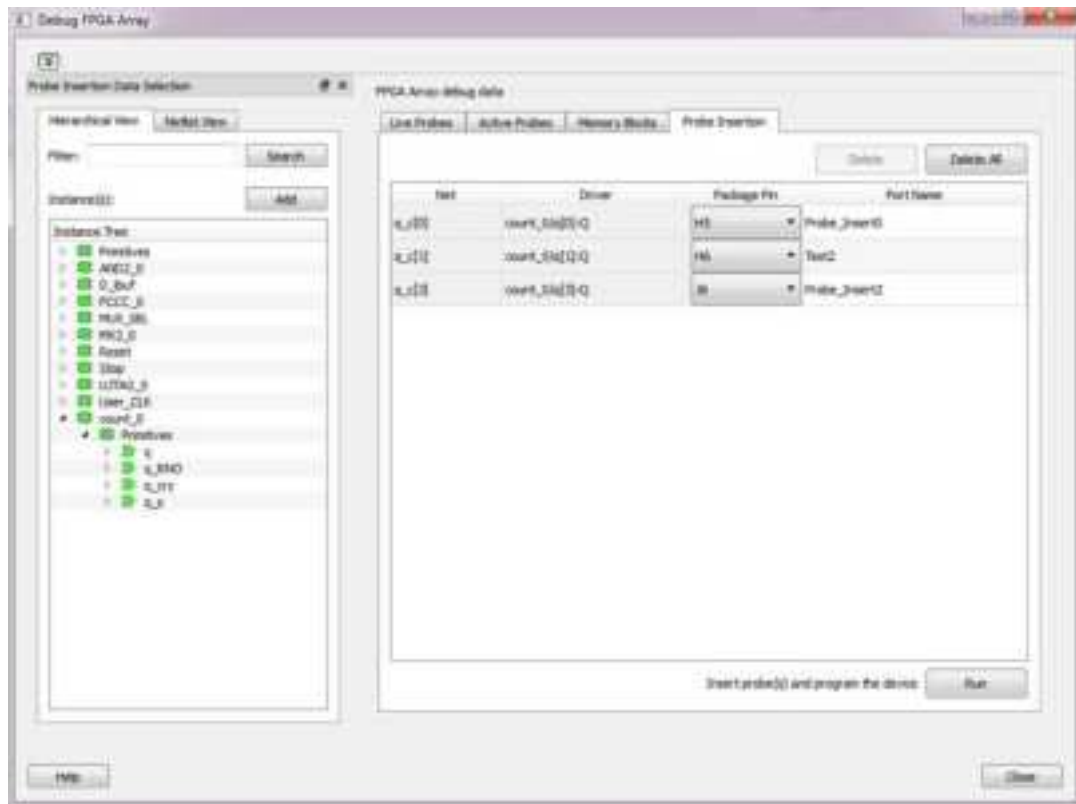
3. Select probe points from the Hierarchical View or Netlist View, right click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the **Filter** box.

Each entry has a Net and Driver name 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 default Port Name if desired.

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

Figure 3-26. Debug FPGA Array > Probe Insertion > Add Probe



5. Click **Run**.

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

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

3.1.8.2. Deleting a Probe [\(Ask a Question\)](#)

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.

3.1.8.3. Reverting to the Original Design [\(Ask a Question\)](#)

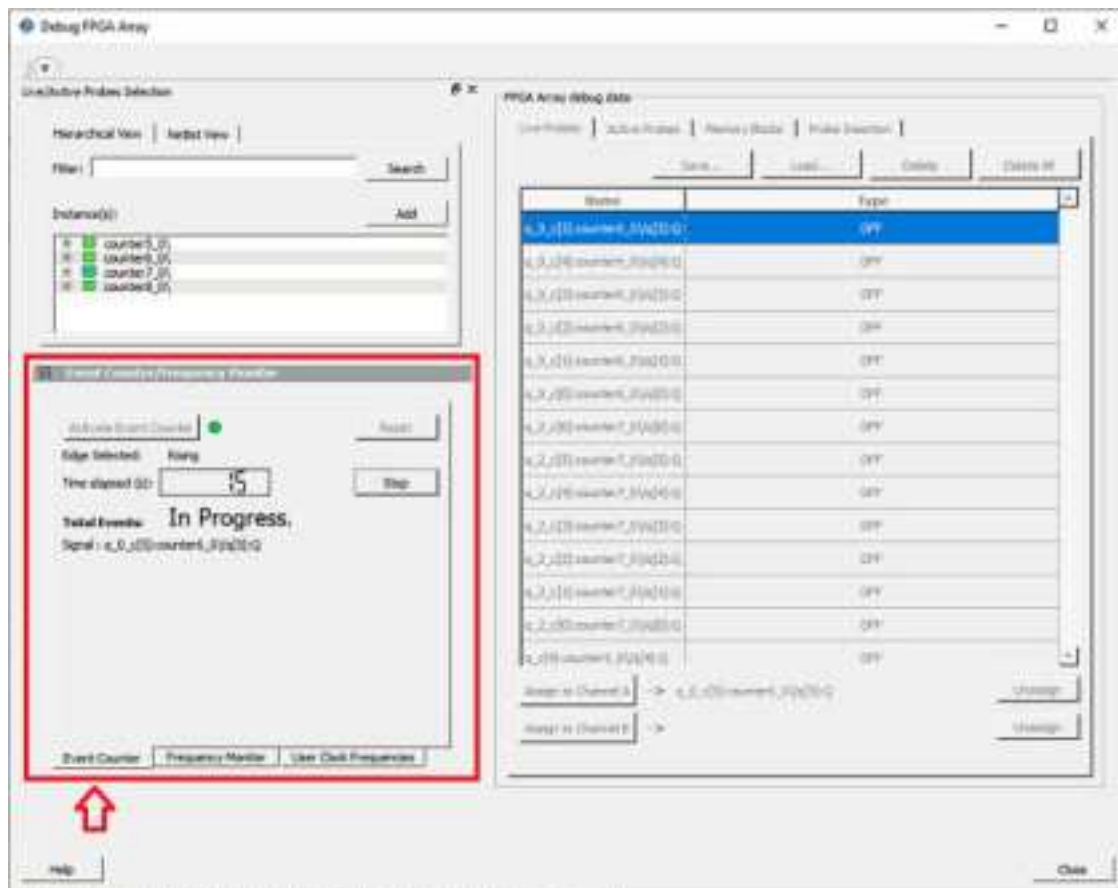
To revert to the original design after you have finished debugging:

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

3.1.9. Event Counter [\(Ask a Question\)](#)

Enabling FPGA Hardware Breakpoint enables Event Counter in the SmartDebug tool (see section [FPGA Hardware Breakpoint Auto Instantiation](#)). This feature is used for any pseudo-static signal that does not toggle often. It counts the number of positive-edge transitions. Activate Event Counter is enabled when any signal or probe point from the selected list is assigned to Channel A via Live Probe. If you click **Activate Event Counter**, the count begins and the time elapsed is updated. Counting continues until the event counter is stopped using the **Stop** option. When Event counter is active, other tabs and features, such as **Active Probes**, **Memory Blocks**, and **Probe Insertion** are disabled.

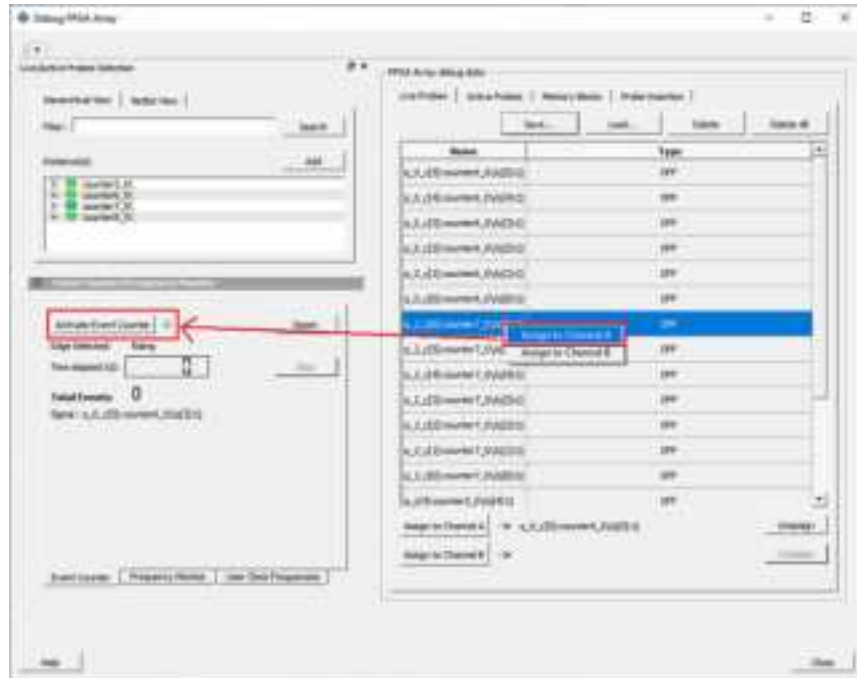
Figure 3-27. Event Counter Tab/UI



3.1.9.1. Activating the Event Counter [\(Ask a Question\)](#)

To activate the Event Counter, assign a signal to Probe Channel A, and then click **Activate Event Counter**.

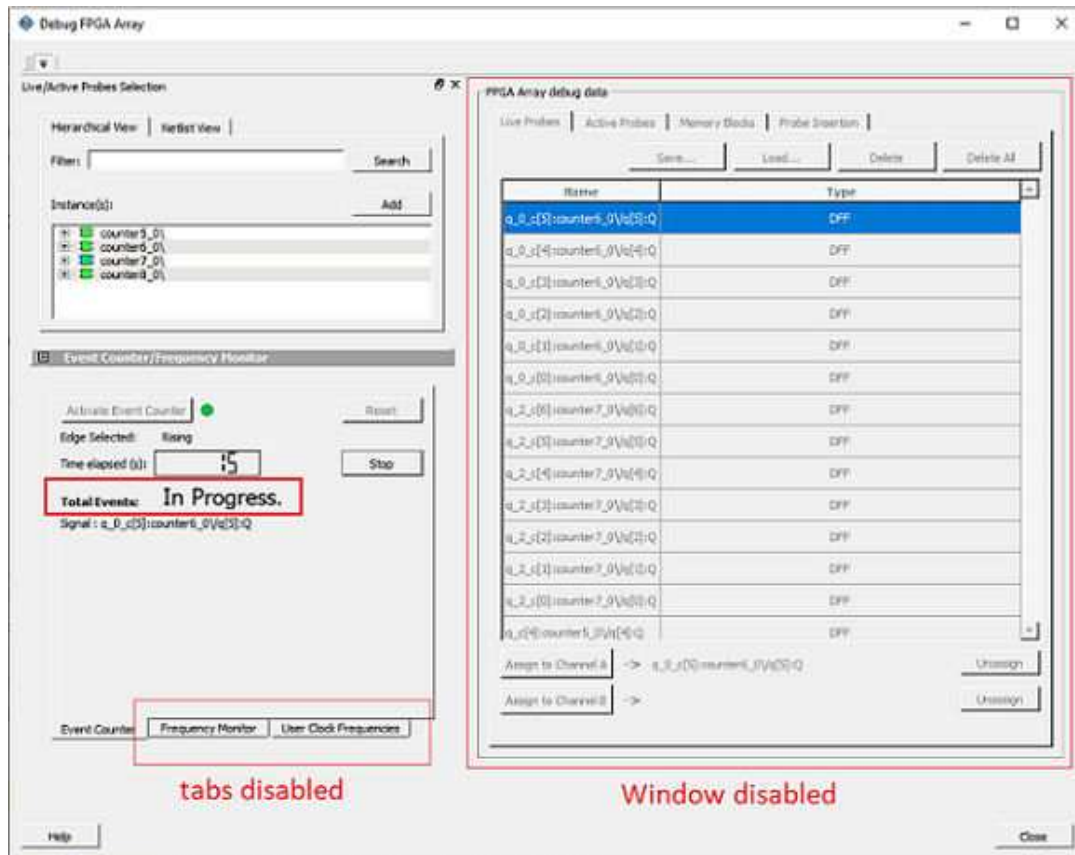
Figure 3-28. Activating the Event Counter - Assign Probe Channel



3.1.9.2. Running the Event Counter [\(Ask a Question\)](#)

When the Event Counter is active and in progress, a green LED appears next to the button as shown in the following figure. Time elapsed is shown in seconds. The **Stop** button is enabled to stop the counter. FPGA Array debug data and the control tabs in the Event Counter panel are disabled while running the Event Counter.

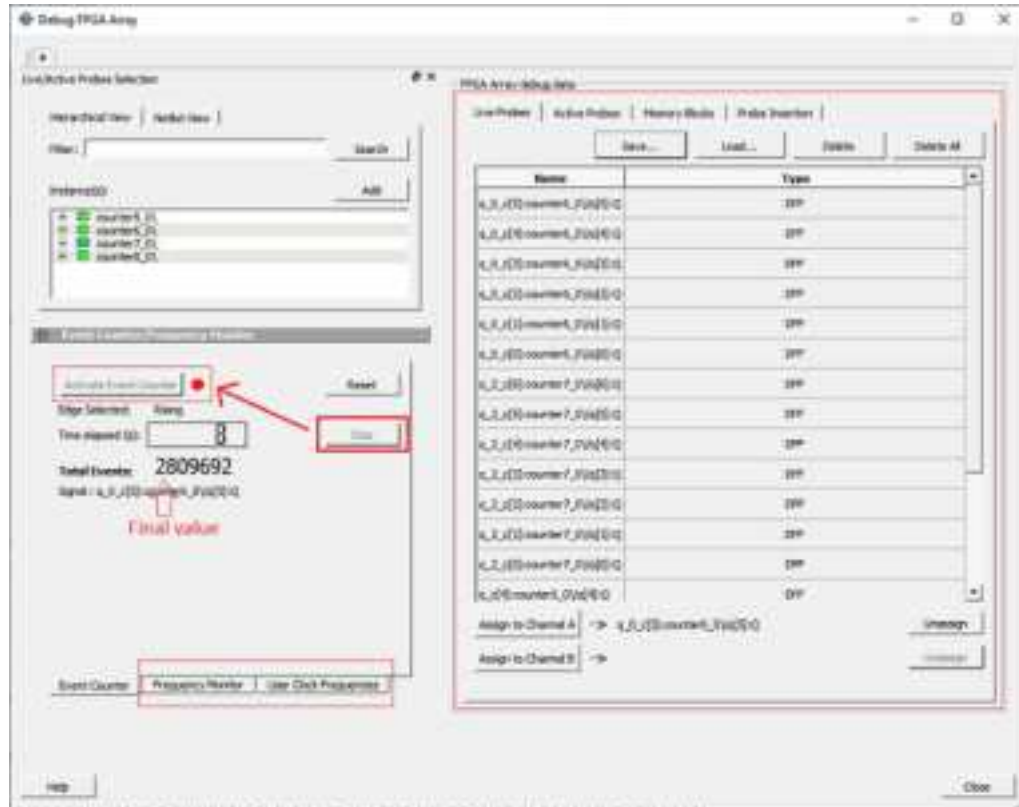
Figure 3-29. Running the Event Counter



3.1.9.3. Stopping the Event Counter [\(Ask a Question\)](#)

To stop the Event Counter from counting, click the **Stop** button. A red LED appears when the Event Counter has stopped. FPGA Array debug data and the control tabs in the Event Counter panel are enabled when the Event Counter is not running.

Figure 3-30. Stopping the Event Counter



Note: If 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 stops. This is a limitation of the FHB IP, and will be fixed in upcoming releases.

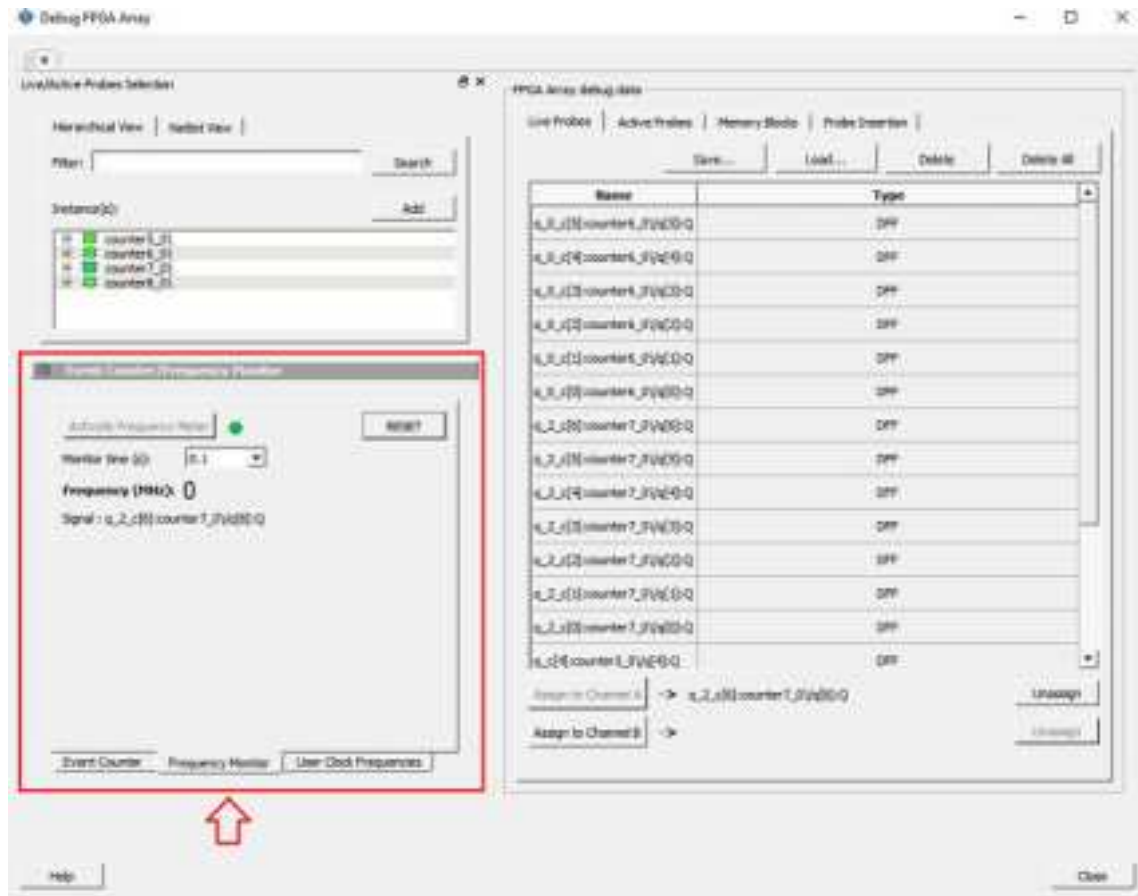
For more information, see [Frequency Monitor](#) and [User Clock Frequencies](#).

3.1.10. Frequency Monitor [\(Ask a Question\)](#)

Enabling FPGA Hardware Breakpoint enables Frequency Monitor in the SmartDebug tool (see section [FPGA Hardware Breakpoint Auto Instantiation](#)). The Frequency Monitor calculates the frequency of any signal in the design that can be assigned to Live Probe channel A. The Frequency Monitor is enabled when a probe point is assigned to Channel A via Live Probe.

The drop-down menu allows you to select the time to monitor the signal before the frequency is calculated. 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 3-31. Frequency Monitor Tab/UI

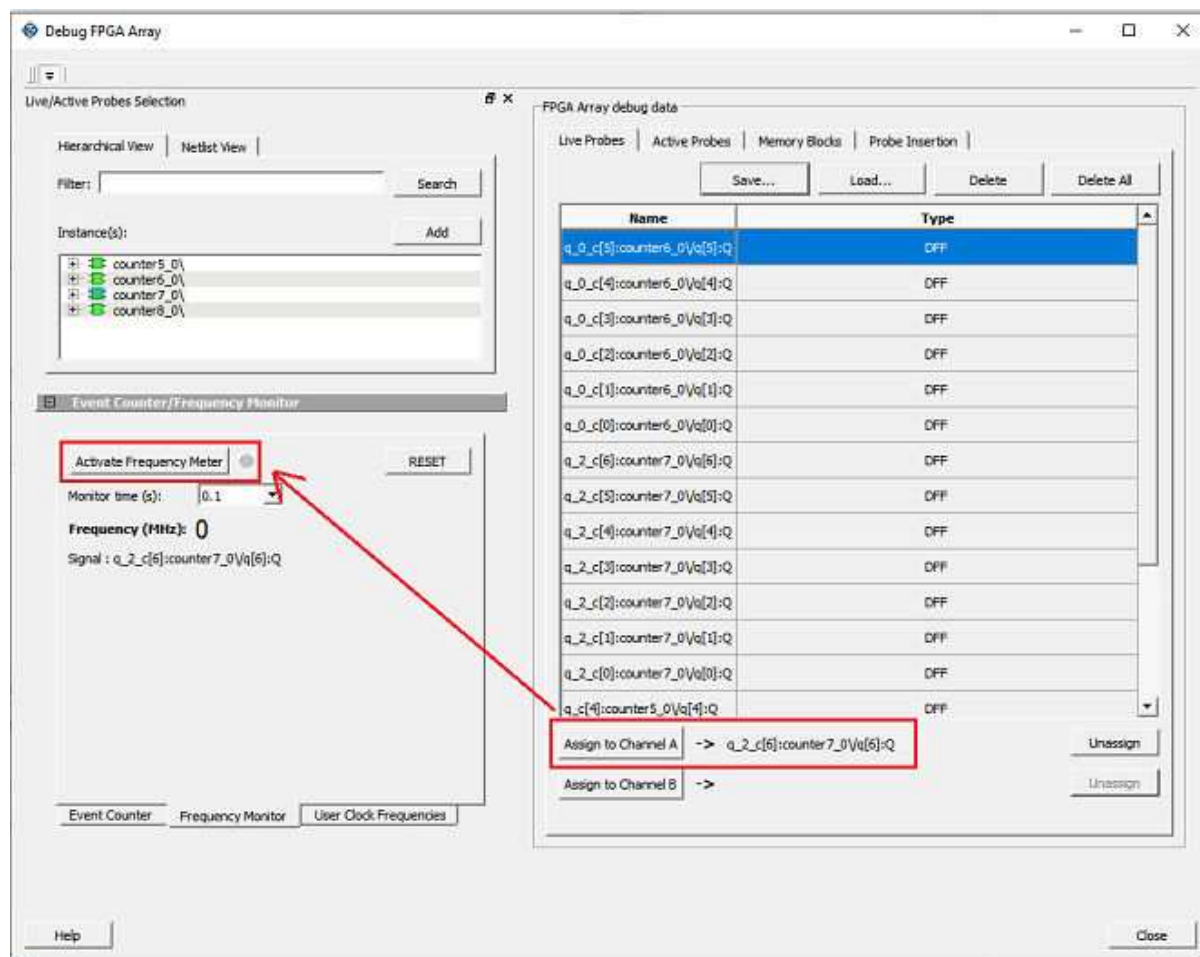


In the **Frequency Monitor** tab, you can activate the Frequency Monitor, change the monitor time (delay to calculate frequency), reset the monitor, and set the frequency in megahertz (MHz). Click the drop-down list to select monitor time value. During the frequency calculation, all tabs on the right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

3.1.10.1. Activating the Frequency Monitor [\(Ask a Question\)](#)

Assign a signal to Channel A, click the **Live Probe** tab, click the **Frequency Monitor** tab, and then click **Activate Frequency Monitor**.

- **Figure 3-32. Activating the Frequency Monitor**

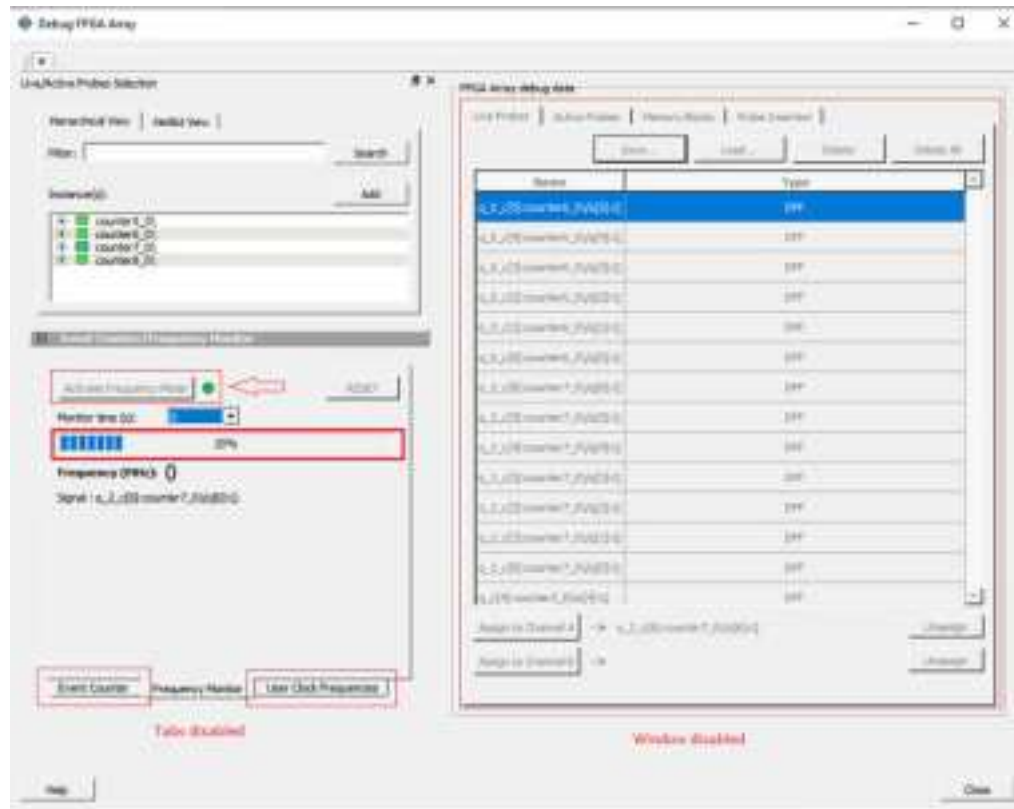


3.1.10.2. Running the Frequency Monitor [\(Ask a Question\)](#)

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**.

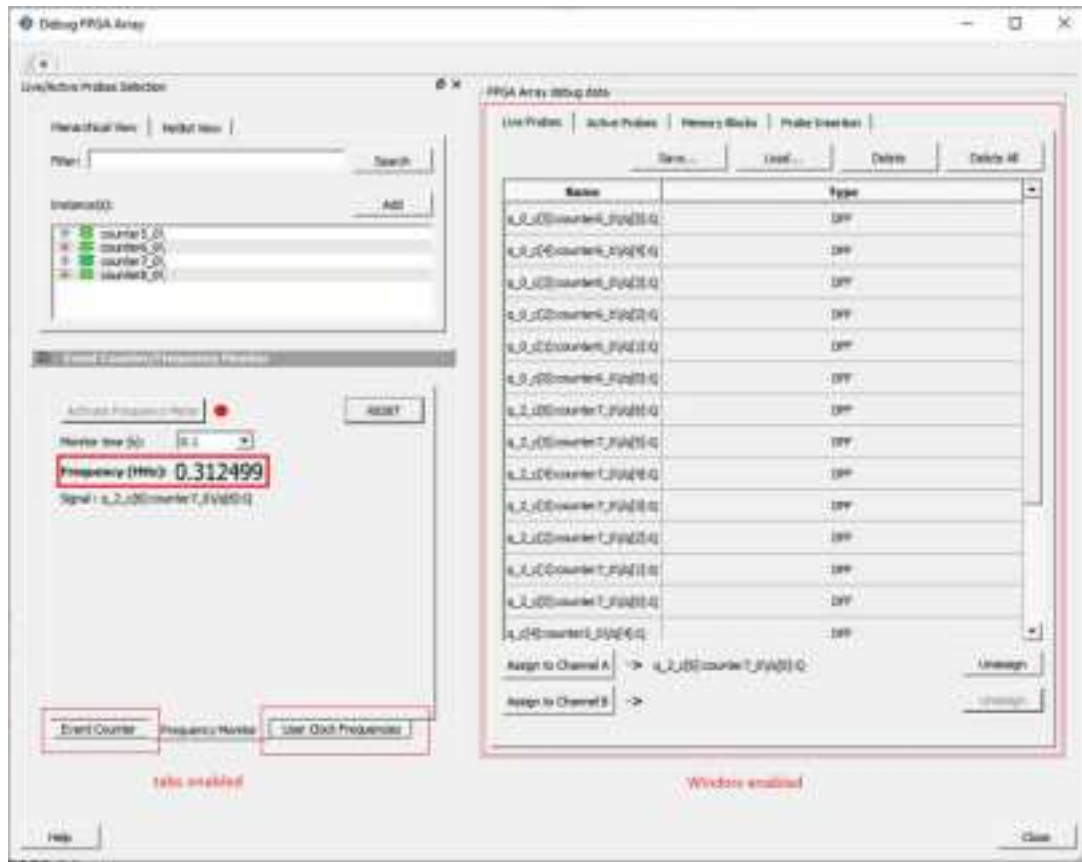
Figure 3-33. Running the Frequency Monitor



3.1.10.3. Stopping the Frequency Monitor [\(Ask a Question\)](#)

The Frequency Monitor stops when the specified monitor time elapses. 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 3-34. Stopping the Frequency Monitor



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

3.1.11. FPGA Hardware Breakpoint Auto Instantiation [\(Ask a Question\)](#)

The FPGA Hardware Breakpoints (FHB) auto-instantiation feature automatically instantiates an FHB instance per clock domain that uses clocks (GL0, GL1, GL2, and GL3) from an FCCC instance. In PolarFire and PolarFire SoC, the instantiation is not limited to FCCC, it can be done on any of the four CLKINT types, such as CLKINT, GCLKINT, RCLKINT, and RGCLKINT. Enabling FHB auto-instantiation also enables the Event Counter and Frequency Monitor. 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. After halting a selected clock domain or all clock domains, you can play or step on the clock domains. The FHB controls in the SmartDebug UI allow you to control the debugging cycle.

To enable this option:

1. Launch **Libero**.
2. Select **Project > Project Settings**. The **Project settings** dialog box appears.
3. Select **Design flow** from the options available on the left pane of the dialog box.
4. Select the **Enable FPGA Hardware Breakpoints Auto Instantiation** check box from the **Root top** options that appear on the right pane of the dialog box.
5. In PolarFire and PolarFire SoC, you must create an NDC constraints file in Constraint Manager and enter the following command in the file to instantiate an FHB IP: `auto_instantiate_fhb -inst_name {<clkint name here>}`. To instantiate FHB in all clock domains in the design, those many commands must be entered in the NDC file. This is auto-generated for other family devices.

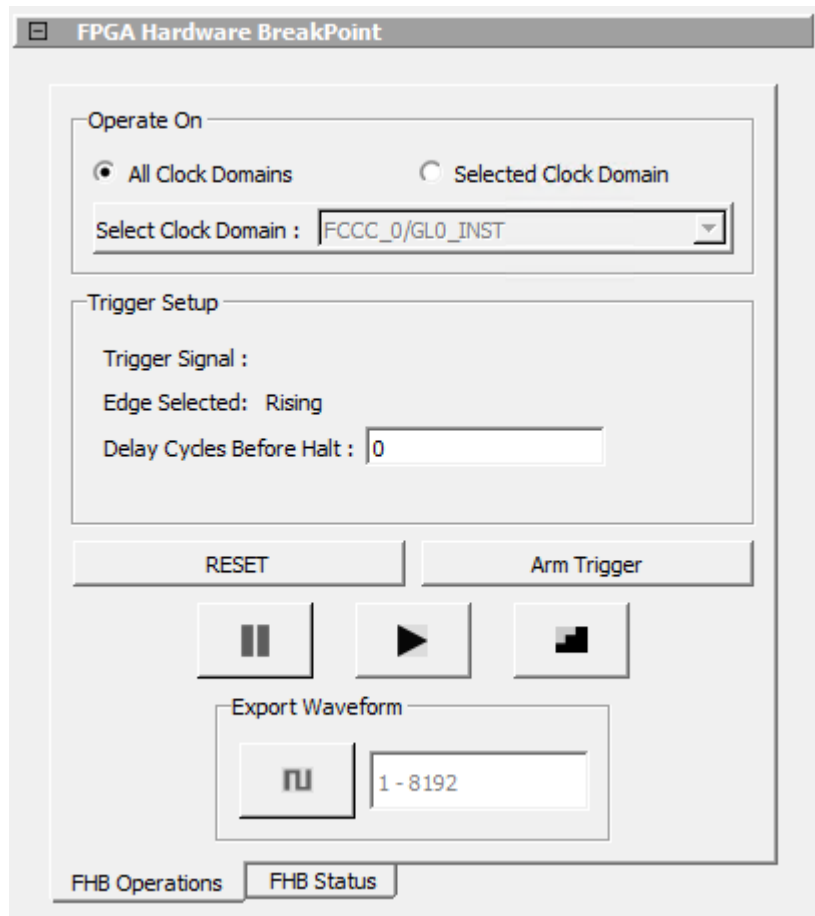
Notes: Observe the following:

- FHB auto-instantiation can also be done in the “Import netlist as VM file” flow.
- In PolarFire and PolarFire SoC, FHB can also be instantiated in the designs that have secured or encrypted components.
- The Spatial Debug Widget and FHB feature are hidden, and an INFO message is printed in the log if the user design has live probe I/Os enabled as input.

3.1.11.1. FHB Operations [\(Ask a Question\)](#)




If there is an auto-instantiated FHB instance in the design, FHB controls appear in the **FPGA Operations** tab.

Figure 3-35. FPGA Hardware BreakPoint - FHB Controls



Selecting a Clock Domain Mode

Select **All Clock Domains** or **Selected Clock Domain** to set the FHB instances to the appropriate mode.

- If you select **All Clock Domains**, use the Halt (Pause) , Play , and Step  buttons to navigate through all clock domains.
- If you select **Selected Clock Domain**, select a clock domain from the **Select Clock Domain** drop-down. Use the Halt (Pause), Play, and Step buttons to navigate through the selected clock domain. If you switch clock domains, previous clock domain settings are not retained.

Specifying a Trigger

After you assign the Live Probe PROBE_A connection, determine whether a certain number of clock cycles is required before halting the clock domain after triggering.

- If clock cycles are not required, click **Arm Trigger** to stop the DUT on the next positive edge that occurs on the signal connected to Live Probe PROBE_A.
- If clock cycles are required, enter a value between 0 and 255 for **Delay Cycles Before Halt**, and then click **Arm Trigger**. This setting configures the FHBs to trigger after the specified delay from the rising edge trigger. The delay is not applied to a forced Halt.

Observe the following guidelines:

- The Trigger Signal appears as **Not Connected** until a live probe is assigned.
- When a probe is assigned to Live Probe PROBE_A, the Trigger Signal updates.
- Clicking **Arm Trigger** when a live probe connection is made disables FHB functions until the trigger is disarmed automatically or the design is force halted.

Live Probe Halt

To halt a selected clock domain or all clock domains (based on your clock mode selection), assign a signal to Live Probe PROBE_A in the **Live Probes** tab in the Debug FPGA Array dialog box, and then click the **Active Probe** tab to see the FPGA Hardware Breakpoint controls.

To arm the FHBs to look for a trigger on the signal connected to Live Probe PROBE_A, click **Arm Trigger**. After the trigger occurs, the clock domains are halted. If only one clock domain is halted, other clock domains continue to run, and you should anticipate results accordingly.

Note: You can delay Live Probe Halt up to 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

To force halt a selected clock domain or all clock domains (based on your clock 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 Button

When the clock domain is in a halted state (live probe halt or force halt), clicking **Play** in the FPGA Hardware Breakpoint controls 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 **Select Clock Domain**.

Step Button

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. Clicking this button advances the clock domain by one clock cycle and holds the state of the clock domain. In **All Clock Domains** mode, each clock domain steps sequentially in the order shown in **Select Clock Domain**.

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. The valid range for capture depth is 1 to 8K (8192).

To view the waveforms, import the `.vcd` file. You can then view the waveform file using a waveform viewer that supports the `.vcd` file format.

Trigger Input

To have an event in the DUT trigger the FHB IP (for example, a particular state in the FSM or counter value) when this signal is asserted, use the trigger input signal. If the trigger signal is already asserted (HIGH) when the FHB is armed, the DUT halts immediately.

Force Halt, Play, and Step operations are performed using the FHB controls. After the clock domain is halted, you can either force Play the clock domain or Step the clock domain by 1 clock cycle.

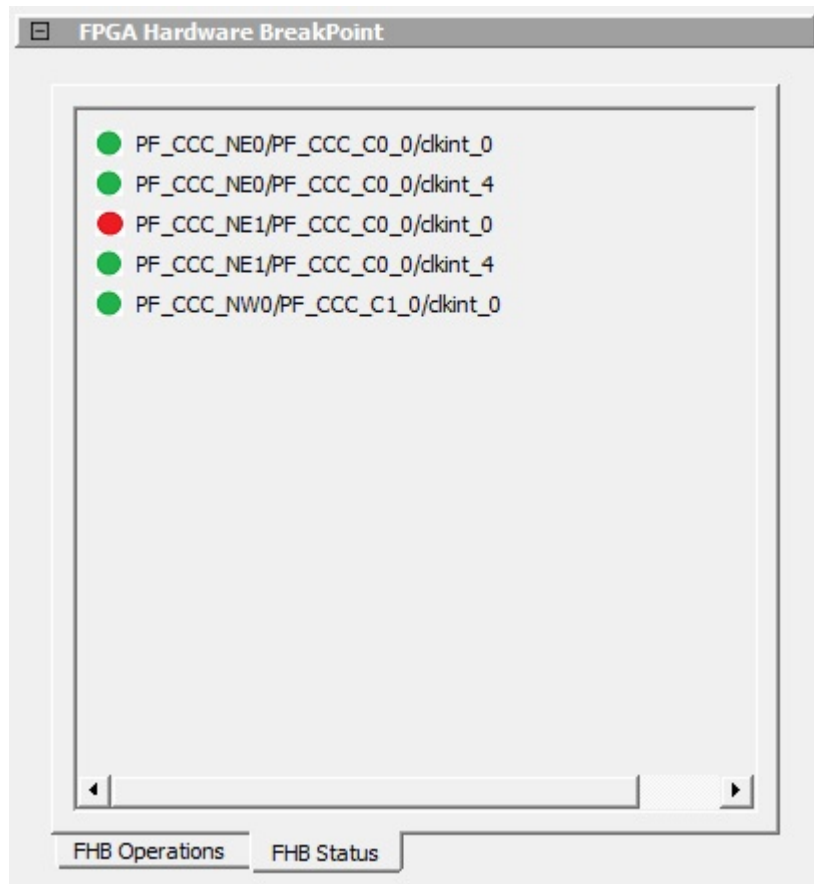
To save the waveform view of the selected active probes, use Export Waveform by specifying the number of clock cycles to capture. The waveform is saved to a `.vcd` file.

3.1.11.2. FHB Status [\(Ask a Question\)](#)

You can now view the status of all the FHB clock domains at the same time using the **FHB Status** tab on the **FPGA Hardware Breakpoint** widget. There is no Tcl command equivalent to this function.

The status of all FHB clocks in a sample design is shown in the following figure.

Figure 3-36. FPGA Hardware BreakPoint - FHB Status



3.1.11.3. Assumptions and Limitations [\(Ask a Question\)](#)

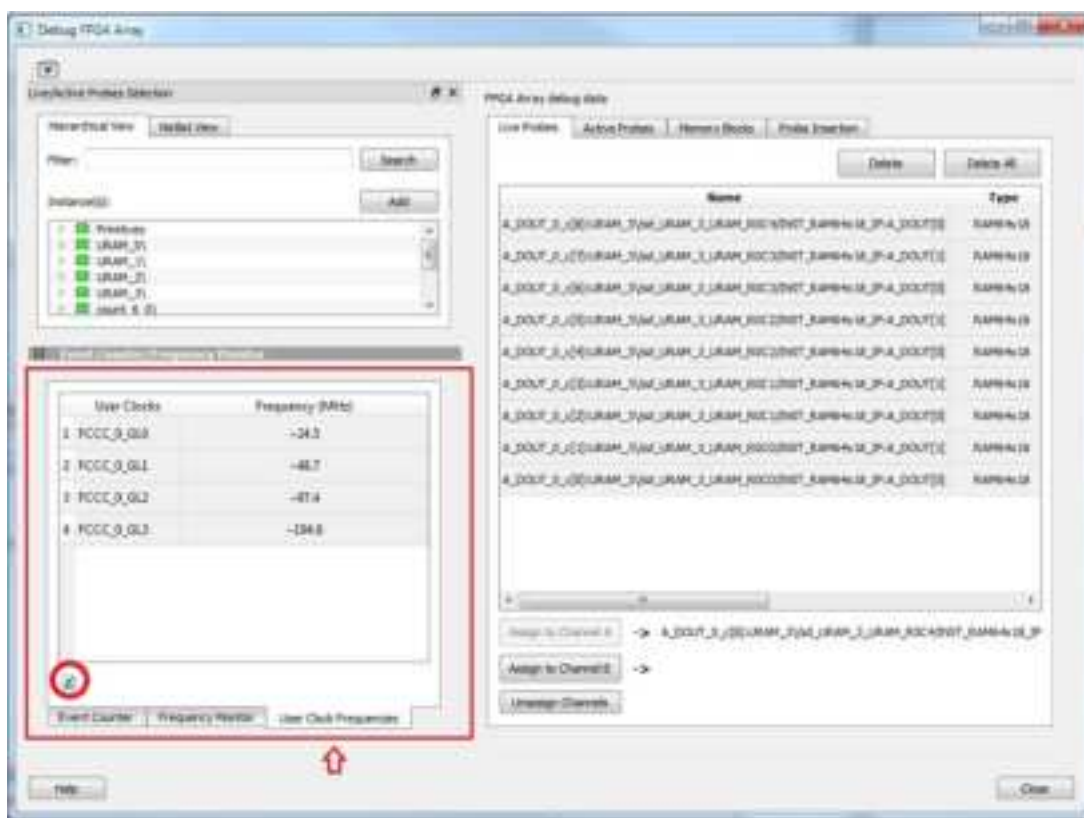
- If you select the auto instantiation option in Libero and ran Synthesis, you must rerun Synthesis for the FHB-related functionality.
- In SmartFusion 2, IGLOO 2, and RTG4 devices, FHB is inserted on the Global clock outputs from CCC only.
- CLKINT_PRESERVE – FHB is not auto-instantiated if the user design contains this macro.
- In SmartFusion 2, IGLOO 2, and RTG4 devices, FHB instantiation is not supported in designs that have encrypted IP cores.
- EDIF using constraints flow is not supported.
- Live Probe triggering occurs on the Positive Edge only.
- In SmartFusion 2, IGLOO 2, and RTG4 devices, 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 halts 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 (that is, 160 MHz).
 - If the DUT clock is running at or less than 160 MHz, the DUT clock halts within one clock cycle (1 or less).
 - For frequencies higher than 160 MHz, the point at which the DUT halts cannot be ensured.

- FHB cannot be instantiated if a design has system controller Suspend Mode enabled.
- For all device families, FHB cannot be added to designs with System Controller Suspend Mode set to ON.
- For PolarFire and PolarFire SoC, the FHB auto-generates the FHB SDC file, obviating the need for you to create this file. However, the file does not appear in the Constraint Manager and is not available for you to manage. For other product families, the FHB SDC file is auto-generated and auto-associated by the FHB flow, and you can manage the SDC file through the Constraint Manager.

3.1.12. User Clock Frequencies [\(Ask a Question\)](#)

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 (circled in the following figure) recalculates frequencies if clocks have been changed.

Figure 3-37. User Clock Frequencies Tab/UI



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

3.1.13. Pseudo Static Signal Polling [\(Ask a Question\)](#)

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

From the **Active Probes** tab in the Debug FPGA Array dialog box, right click a signal, bus, or group and select **Poll** to open the Pseudo-static signal polling dialog box as shown in the following figure.

Figure 3-38. Debug FPGA Array Dialog Box - Poll Option



3.1.13.1. Scalar Signal Polling [\(Ask a Question\)](#)

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

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

To begin polling, click **Start Polling** as shown in the following figure.

Figure 3-39. Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Start Polling



To end polling, click **Stop Polling** as shown in the following figure.

Figure 3-40. Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Stop Polling



Note: You cannot change the poll value or close the polling dialog box while polling is in progress. The elapsed time is updated in seconds until the polled value is found. When the polled value is found, **User value matched** is displayed in green in the dialog box as shown in the following figure.

Figure 3-41. Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - User Value Matched



3.1.13.2. Vector Signal Polling [\(Ask a Question\)](#)

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

Figure 3-42. Pseudo-static signal polling Dialog Box (Vector Signal Polling)



Figure 3-43. Pseudo-static signal polling Dialog Box (Vector Signal Polling) - After Validation



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

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

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

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

3.2. PolarFire and PolarFire SoC Debug Elements [\(Ask a Question\)](#)

The following sections describe the debug elements for PolarFire and PolarFire SoC.

3.2.1. Debug sNVM [\(Ask a Question\)](#)

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.

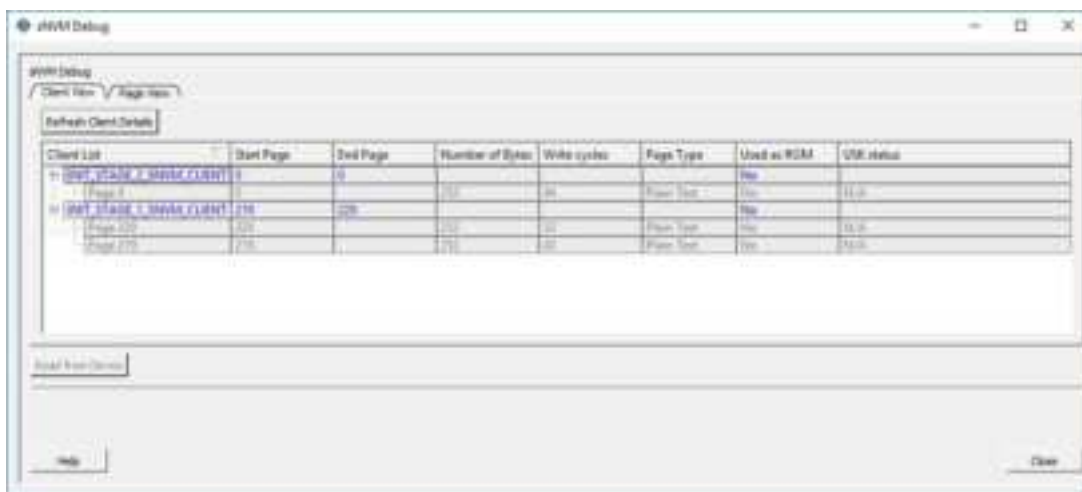
The sNVM Debug window has two tabs – [Client View](#) and [Page View](#).

3.2.1.1. Client View [\(Ask a Question\)](#)

When the sNVM window opens, two tabs appear. 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.

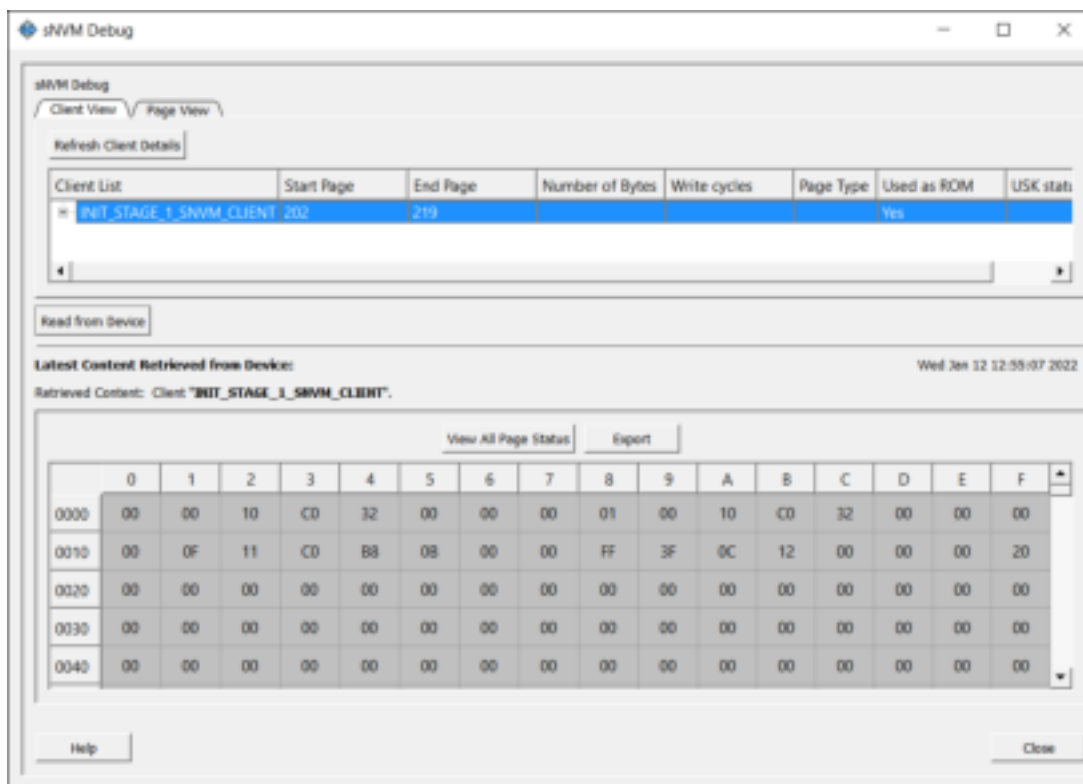
You can select only one client at a time. Pages inside the client cannot be selected. The Start Page, End Page, and Number of Bytes are displayed for the selected client.

Figure 3-44. 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. Click the **Export** button to export the sNVM data to a text file.

Figure 3-45. Client View - Memory

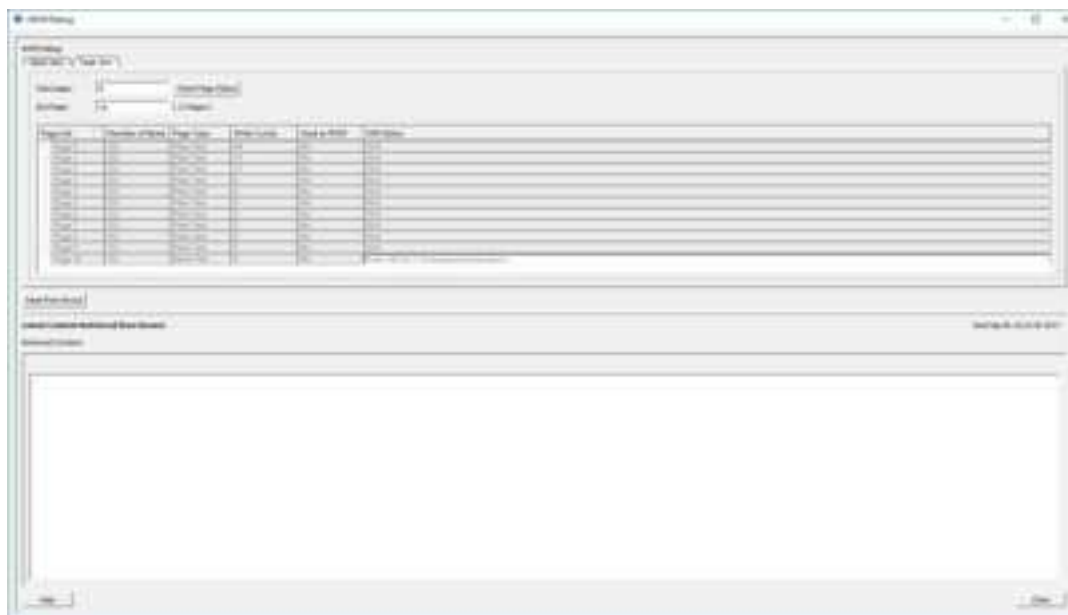


3.2.1.2. Page View [\(Ask a Question\)](#)

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

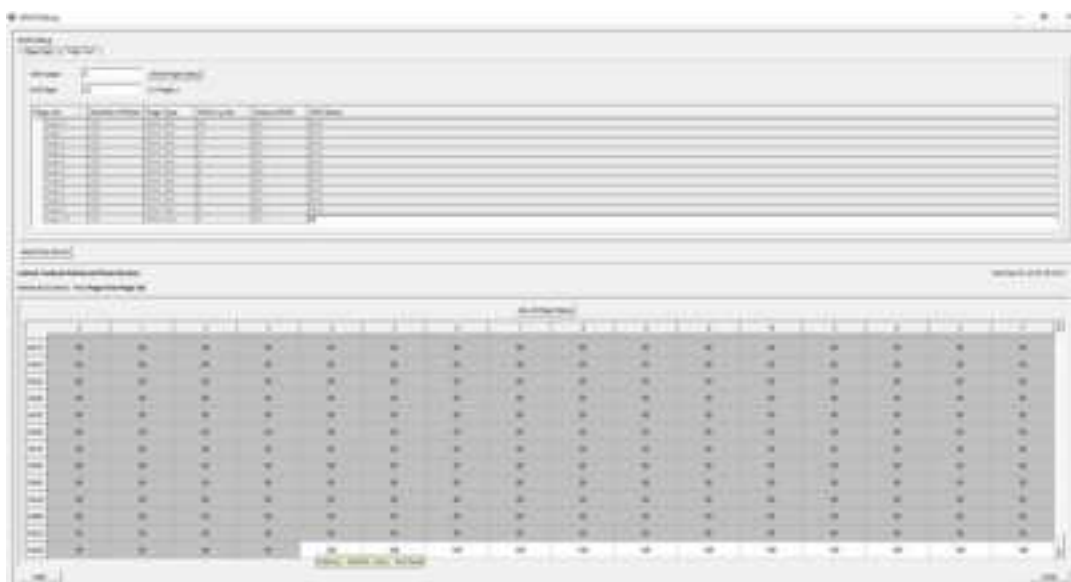
If a page is secured, the default USK is used by SmartDebug to get the page status. If successful, the USK automatically reads the page. If a different USK has been set using system services, use the option to enter the USK, as shown in the following figure.

Figure 3-46. Page View - Enter USK highlighted



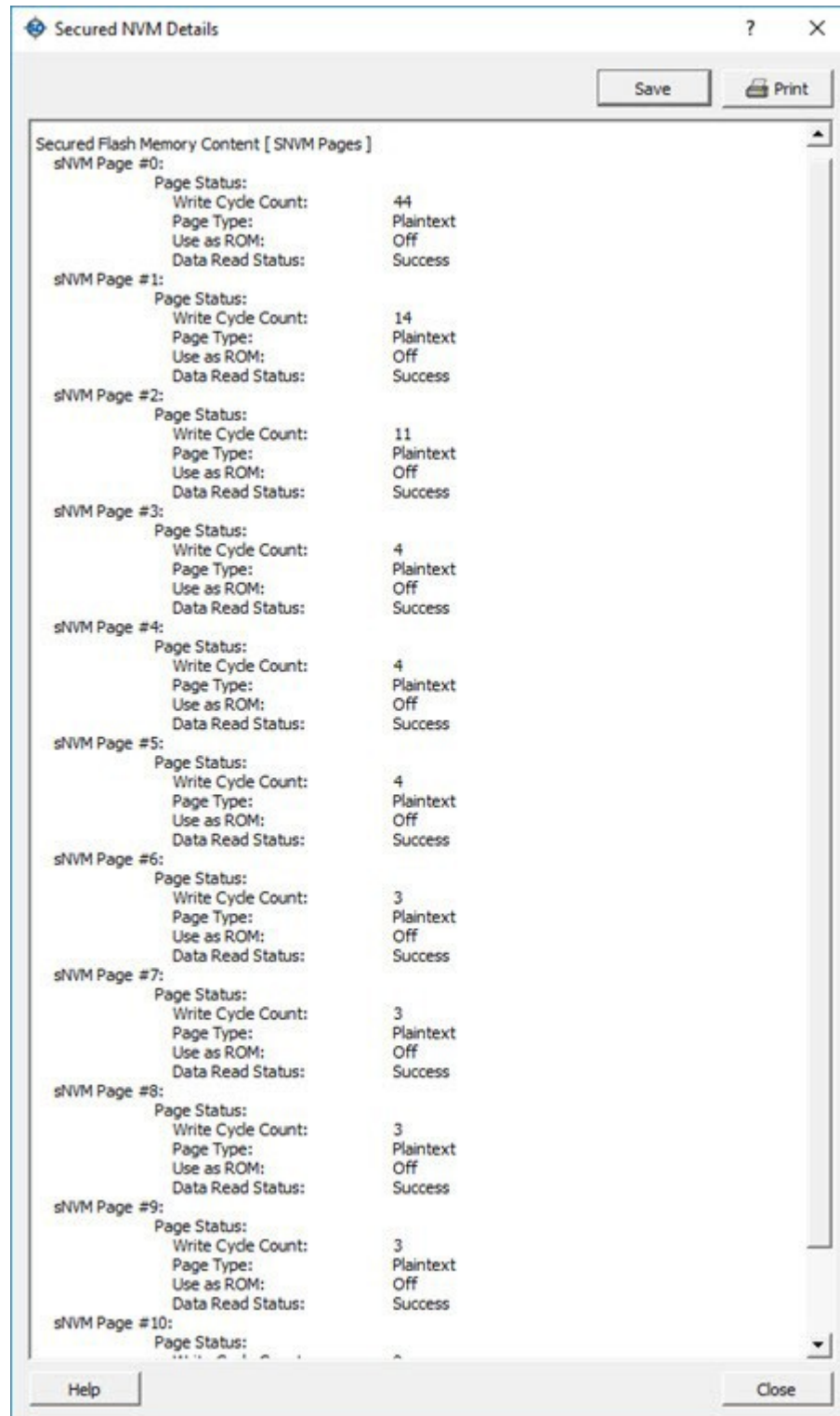
The following figure shows the specified page range.

Figure 3-47. Page View - Page Range



The following figure shows the page status.

Figure 3-48. View All Page Status



3.2.1.3. Read Operation [\(Ask a Question\)](#)

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

Entering valid parameters and clicking **Check Page Status** displays a table of all pages 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.

3.2.1.4. Runtime Operations [\(Ask a Question\)](#)

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 preceding 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.

3.2.1.5. Demo Mode [\(Ask a Question\)](#)

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.

3.2.2. Debug Transceiver [\(Ask a Question\)](#)

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. This opens the Debug TRANSCEIVER dialog box.

3.2.2.1. Configuration Report [\(Ask a Question\)](#)

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 3-49. Debug TRANSCEIVER - Configuration Report

Lanes	Q0L0_0_PPF_XCVR_C0_0/R_XCVR	Q0L1_TX_L2_RX_0_PPF_XCVR_BRM_C1_0/R_XCVR	Q0L1_TX_L2_RX_0_PPF_XCVR_BRM_C2_RXonly_0/R_XCVR	Q0L3_0_PPF_XCVR_C0_0/R_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 preceding parameters, green indicates true and red indicates false.

3.2.2.2. Transceiver Hierarchy [\(Ask a Question\)](#)

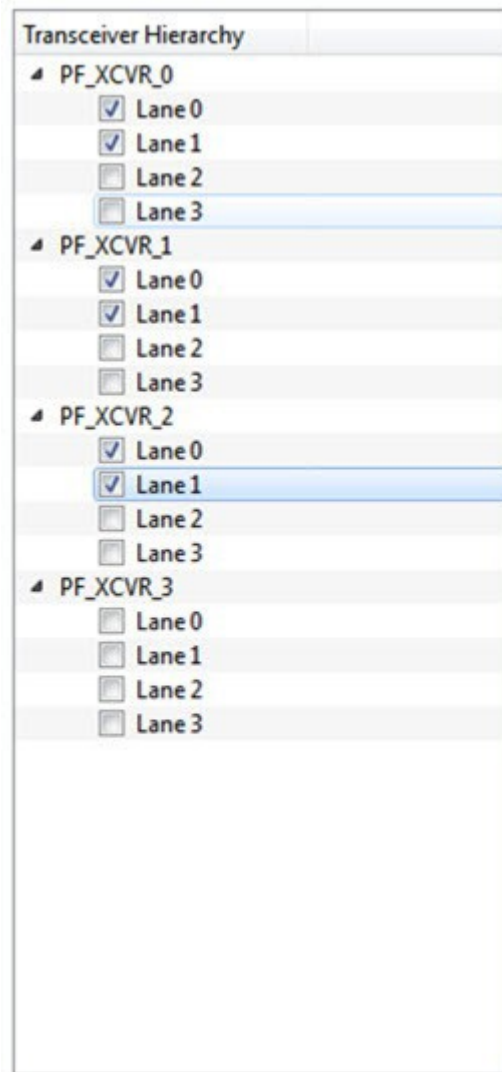
Transceiver Hierarchy view is a lane hierarchy with all the lanes instantiated in the design shown with respect to top level instance.

Transceiver Hierarchy view appears on the following tabs:

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

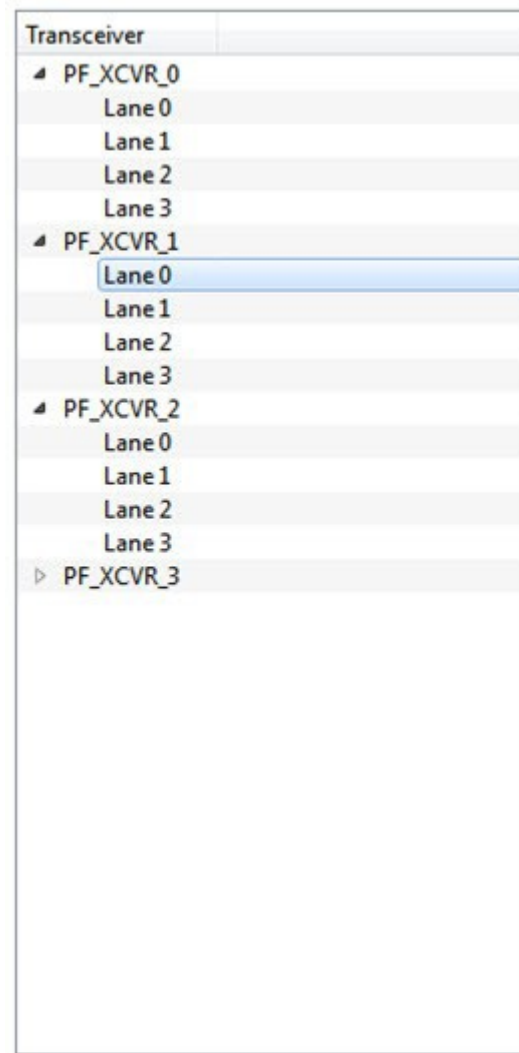
On the SmartBERT, Loopback Modes, and Static Pattern Transmit pages, check boxes allow multiple lanes to be selected for debug, as shown in the following figure.

Figure 3-50. 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 3-51. Transceiver Hierarchy Lane Selection Example - Eye Monitor Page

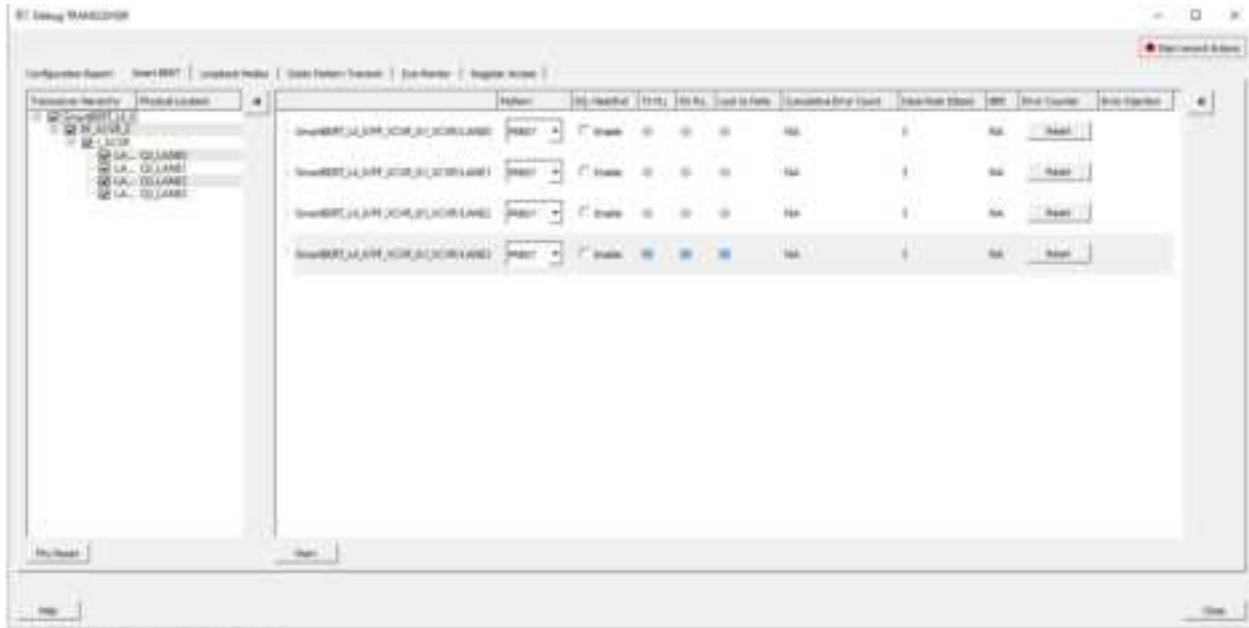


3.2.2.3. SmartBERT [\(Ask a Question\)](#)

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 3-52. Debug TRANSCEIVER - SmartBERT



The following input options and outputs are represented as columns:

- **Pattern:** Input option. Select a PRBS pattern type from the drop-down list: PRBS7, PRBS9, PRBS15, PRBS23, or PRBS31. The default is PRBS7.
- **EQ-NearEnd:** Input option. When checked, enables EQ-NearEnd loopback from Lane Tx to Lane Rx. Disabled for half duplex mode.
- **TX PLL:** Indicates whether a 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 preceding figure.
- **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.

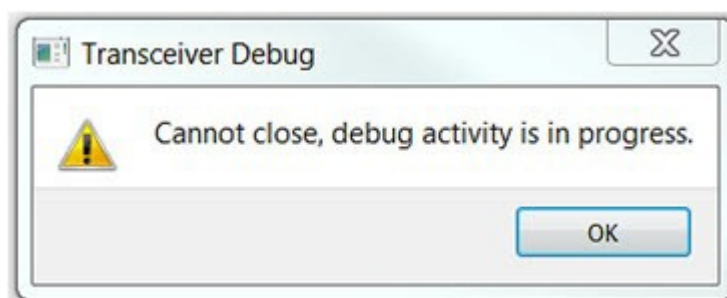
Select the desired options and click **Start** to start the Smart BERT test on all selected lanes. A popup message appears if a test cannot be started on one lane, multiple lanes, or all lanes. Tests will start normally on all unaffected lanes.

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.

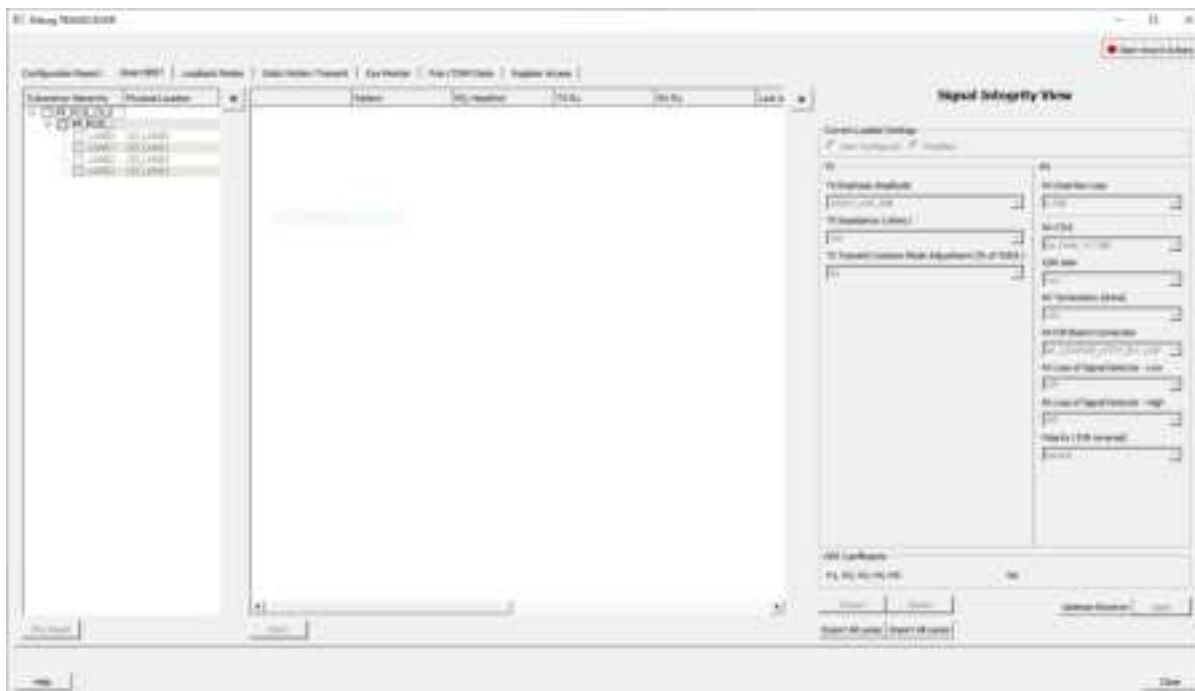
Note: You cannot close the SmartBERT window when a test is in progress. Attempting to do so will result in the following message:



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

Note: SmartBERT tests are disabled for PCIe® lanes.

Figure 3-53. 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 3-54. Configuration Report Showing Unused PCIe Lane



3.2.2.3.1. SmartBERT IP [\(Ask a Question\)](#)

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 check box next to a lane to add the lane 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 3-55. 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 3-56. 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.

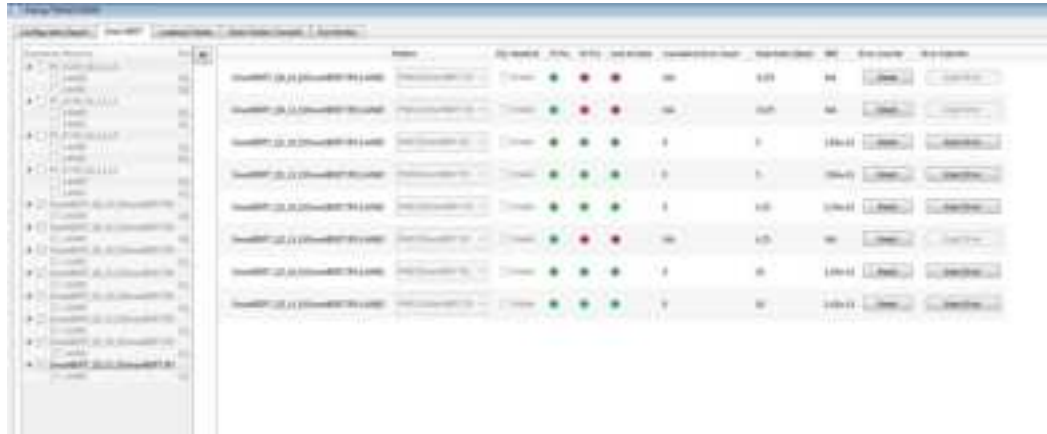
Note: This column does not appear for non-SmartBERT IP lanes, or if a non-configured PRBS pattern has been selected.

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 3-57. SmartBERT IP - Reset and Inject Error

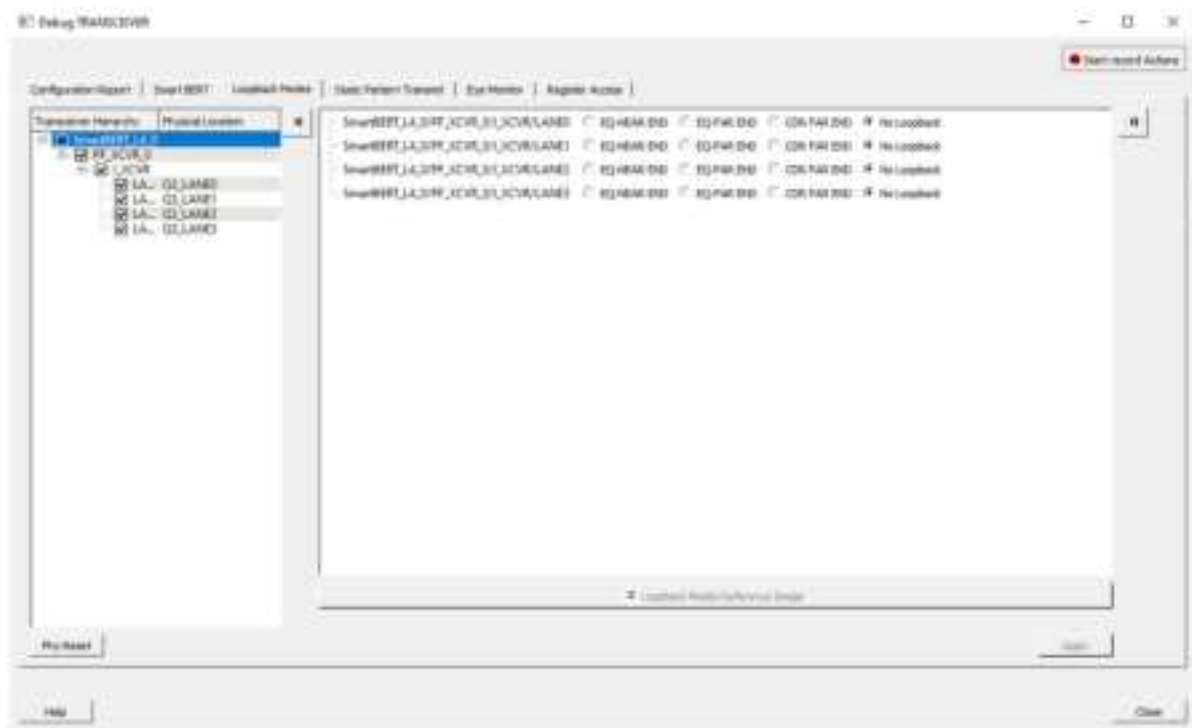


3.2.2.4. Loopback Modes [\(Ask a Question\)](#)

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 3-58. Debug TRANSCEIVER - Loopback Modes



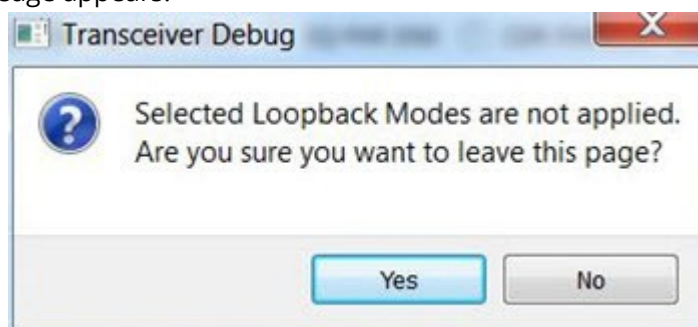
You can select the desired loopback type (EQ-NEAREND, EQ-FAREND, CDR-FAREND, or No Loopback) for each lane.

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 preceding figure.

- **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:



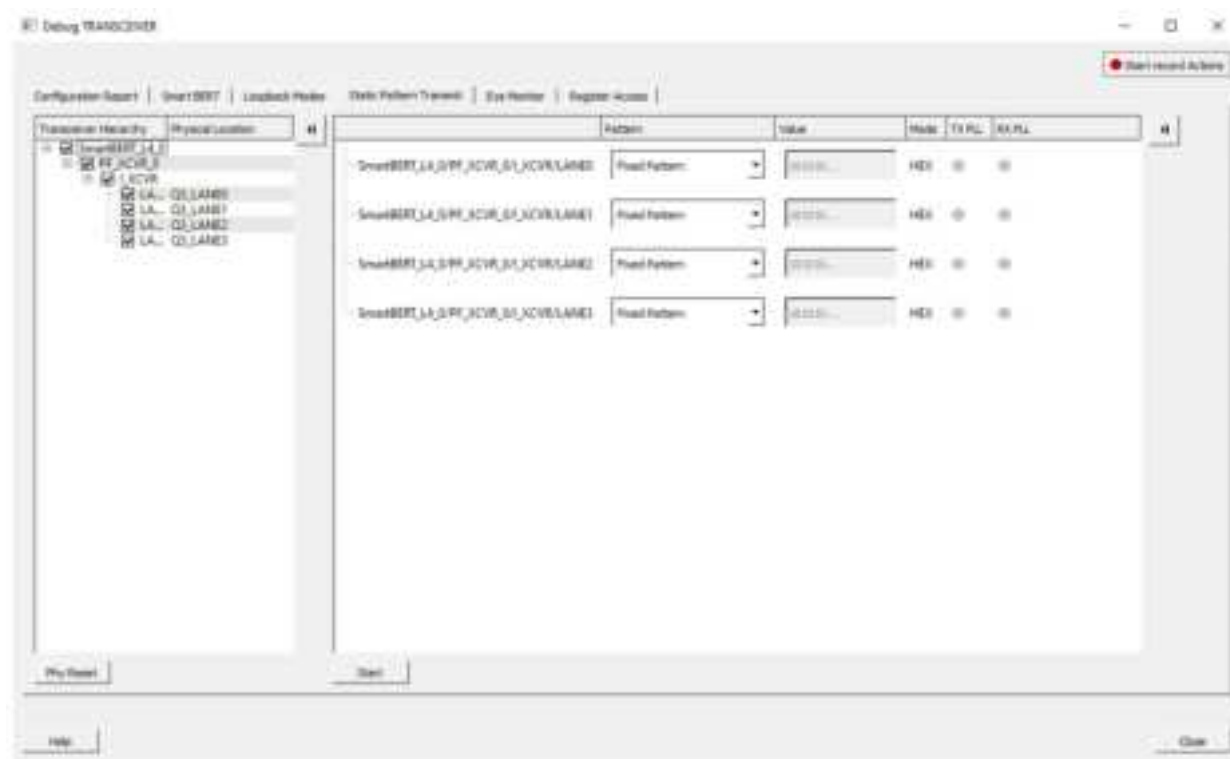
Click **Yes** to ignore the changed selections and move to another selected page. Click **No** to remain on the current page.

3.2.2.5. Static Pattern Transmit [\(Ask a Question\)](#)

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 3-59. 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 width selected.
 - Internal validators dynamically check the pattern and indicate when an incorrect pattern is given as input.
- **Mode:** Currently, HEX mode is supported for pattern type.
- **TX PLL:** Indicates Lane lock onto TX PLL when Static Pattern Transmit is in progress.
 - *Gray:* Test is not in progress.
 - *Green:* Lane is locked onto TXPLL.
 - *Red:* Lane is not locked onto TXPLL.
- **RX PLL :** Indicates Lane lock onto RX PLL when Static Pattern Transmit is in progress.
 - *Gray:* Test is not in progress.
 - *Green:* Lane is locked onto RXPLL.
 - *Red:* Lane is not locked onto RXPLL.

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

3.2.2.6. Eye Monitor [\(Ask a Question\)](#)

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. Start SmartDebug from Libero.
2. Click the **Eye Monitor** tab in the Debug TRANSCIEVER dialog box.

3.2.2.6.1. Eye Scan Mode [\(Ask a Question\)](#)

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



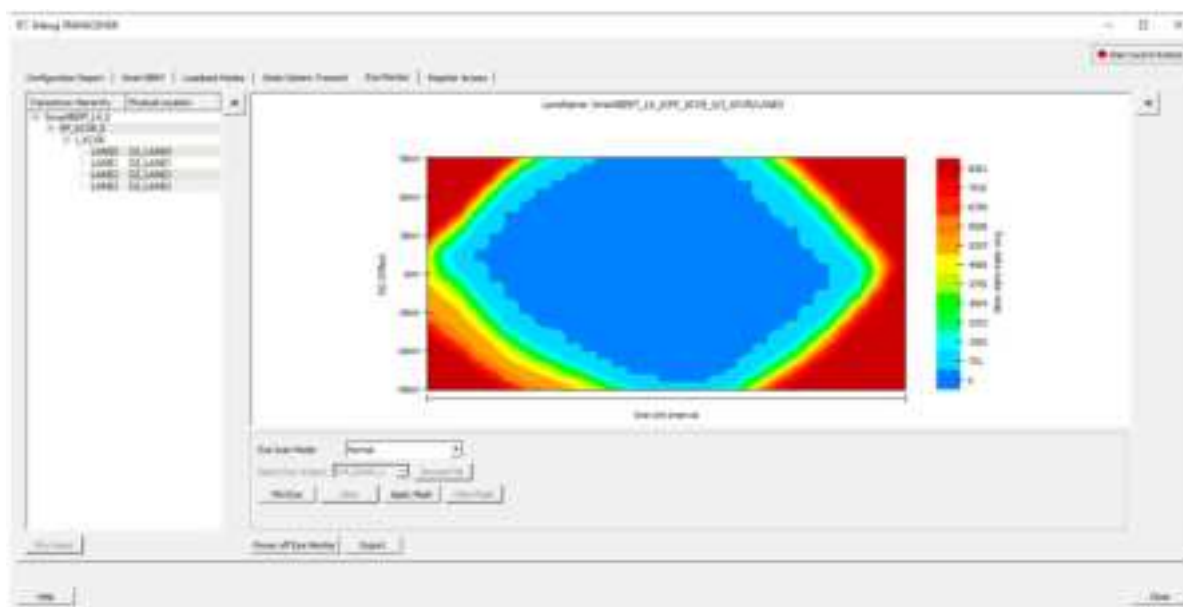
Important: Eye plot is disabled for lanes with a data-rate less than 3 Gbps.

Normal Mode [\(Ask a Question\)](#)

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 3-60. Eye Scan Mode - Normal



Infinite Persistent Mode [\(Ask a Question\)](#)

When the Infinite Persistent mode is selected, the **Plot Eye** button changes to **Start Plot Eye**.

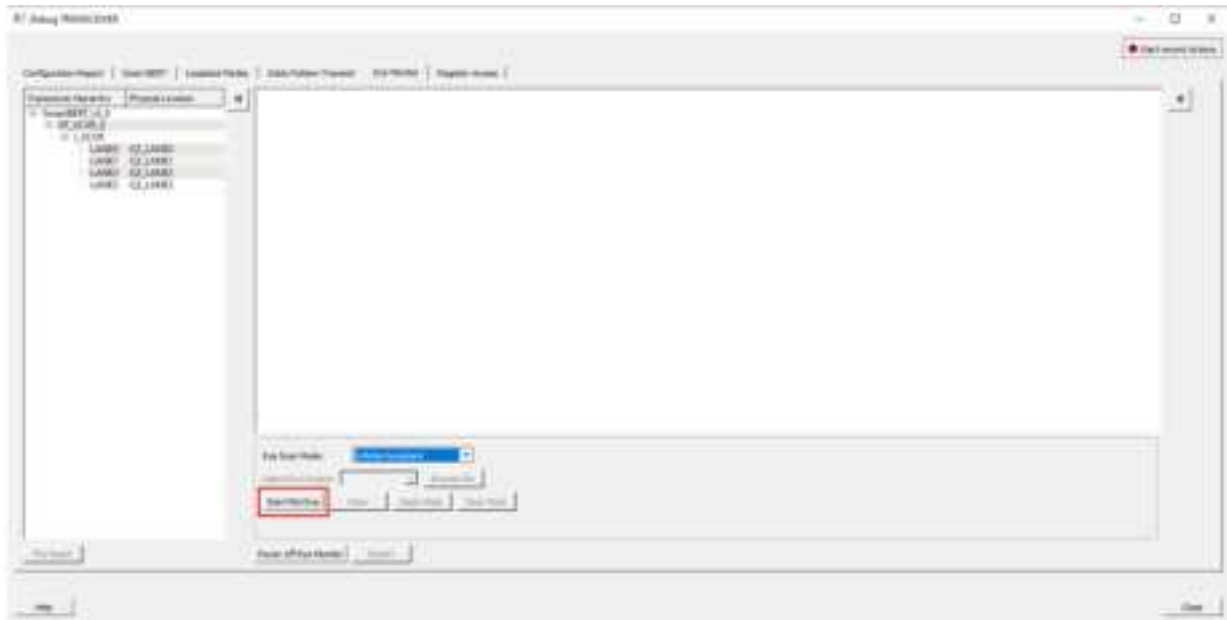


Important:

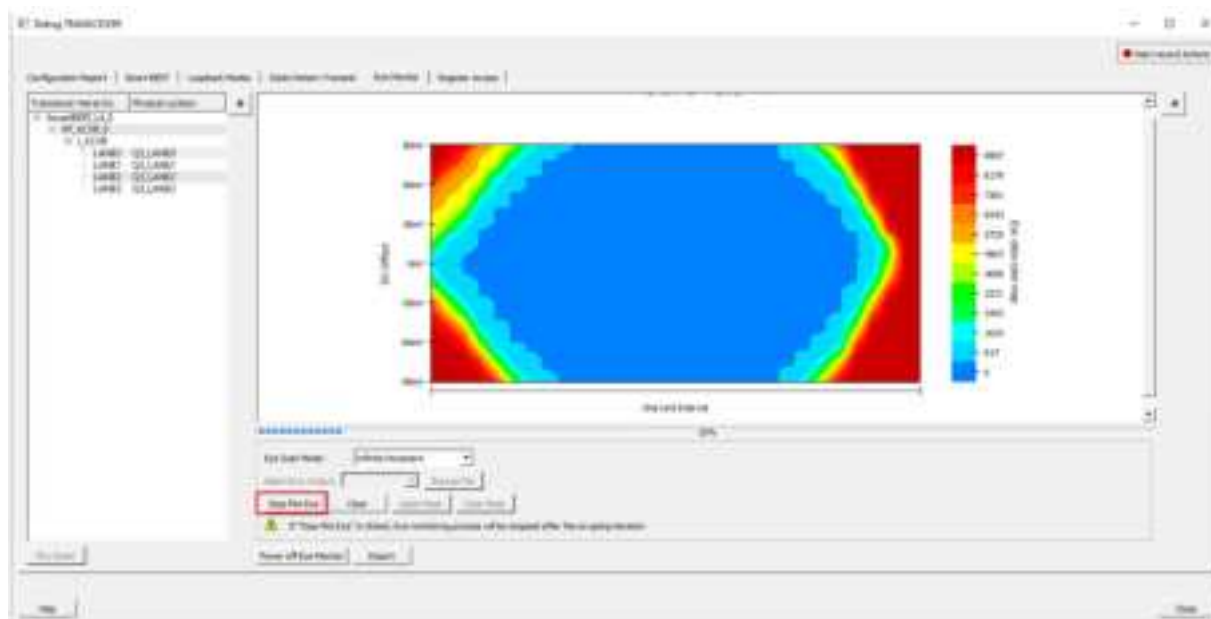
- This feature is not available for Tx Only mode
- Eye plot is disabled for lanes with a data-rate less than 3 Gbps

Click **Start Plot Eye** to start Infinite Persistent eye monitoring as shown in the following figure.

Figure 3-61. 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 appears with a color scheme in the GUI, as shown in the following figure. The completed iteration number, and the cumulative BER is updated and appears 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 3-62. Infinite Persistent Mode - Stop Plot Eye**Clear** [\(Ask a Question\)](#)

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: The Current Iteration count does not reset. Only the cumulative eye is cleared.

Additional Eye Output Text Files [\(Ask a Question\)](#)

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.

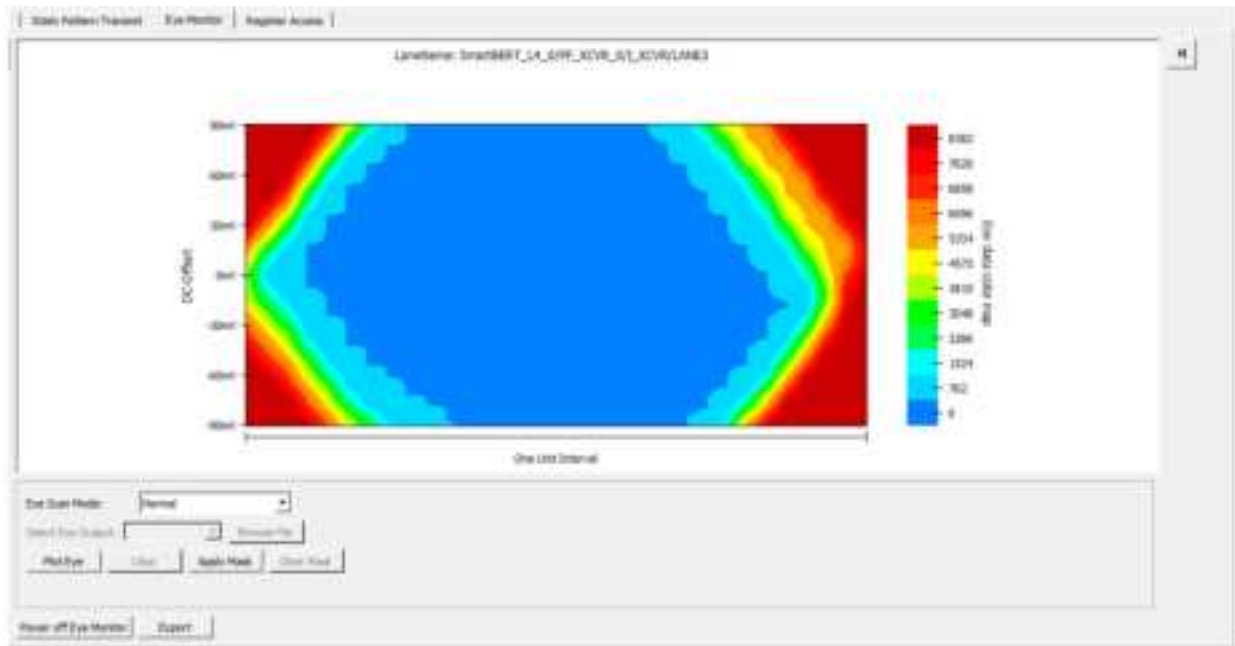
Error Handling [\(Ask a Question\)](#)

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.

Eye Mask [\(Ask a Question\)](#)

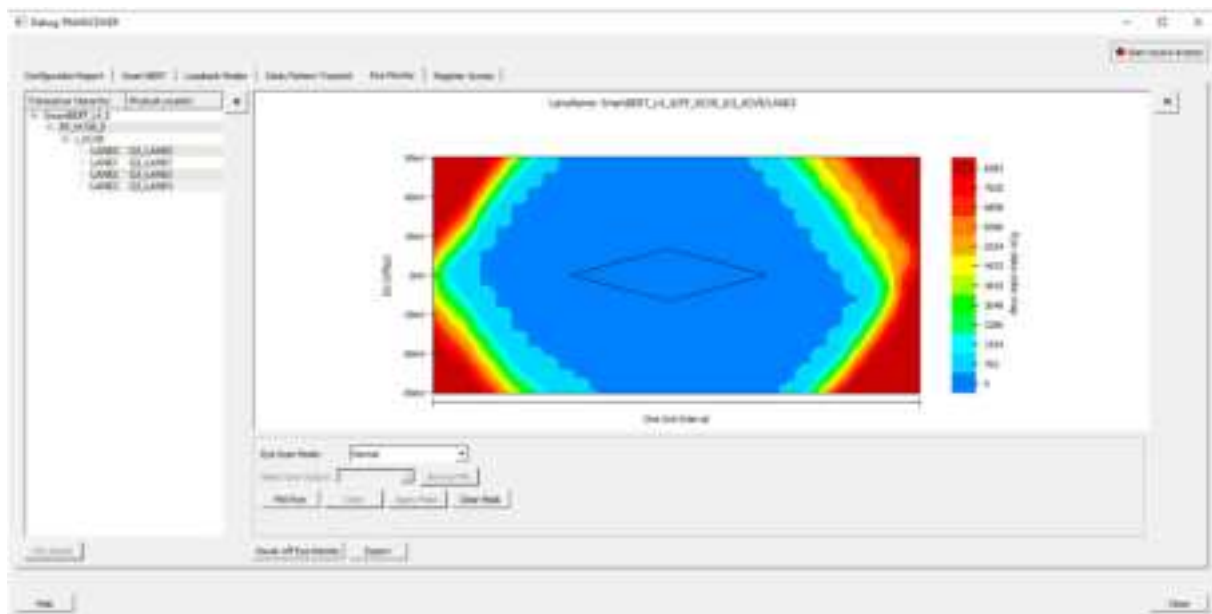
The Eye mask feature has been added to both the normal and infinite persistent modes in the Libero SoC v12.5 release. Eye mask provides a guide to where the best eye opening with least errors can be seen. Both the **Apply Mask** and the **Clear Mask** buttons are disabled in the Default View. Click **Plot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

Figure 3-63. 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 3-64. 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.

3.2.2.6.2. Design Initiated Eye Plots [\(Ask a Question\)](#)

After you run Eye Monitor in an embedded application such as IAR Embedded Workbench using the PolarFire Transceiver driver, you can display the associated eye plots. When the eye monitor firmware is run, the terminal log is stored in a text file located in the root folder. This file is compatible with SmartDebug to view the eye plots.

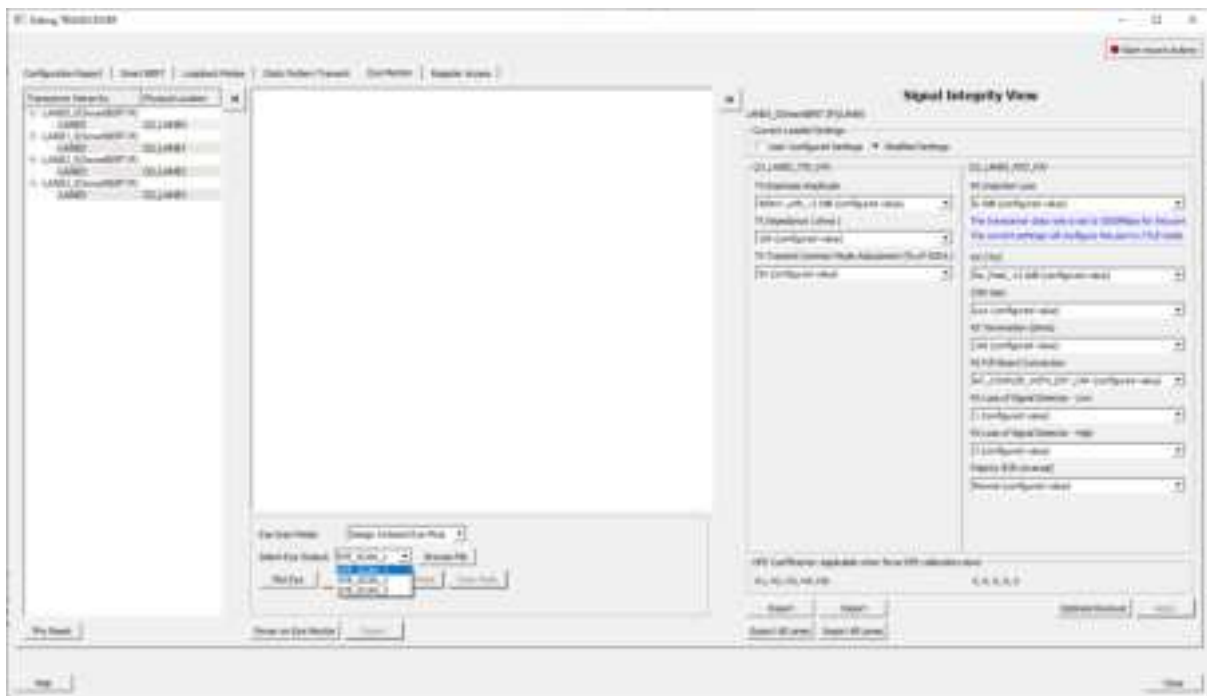


Attention: Do not edit the Eye Plot log in any way before you import it into SmartDebug. Otherwise, an error message appears and the log will not be imported.

To access design-initiated eye plots, perform the following steps:

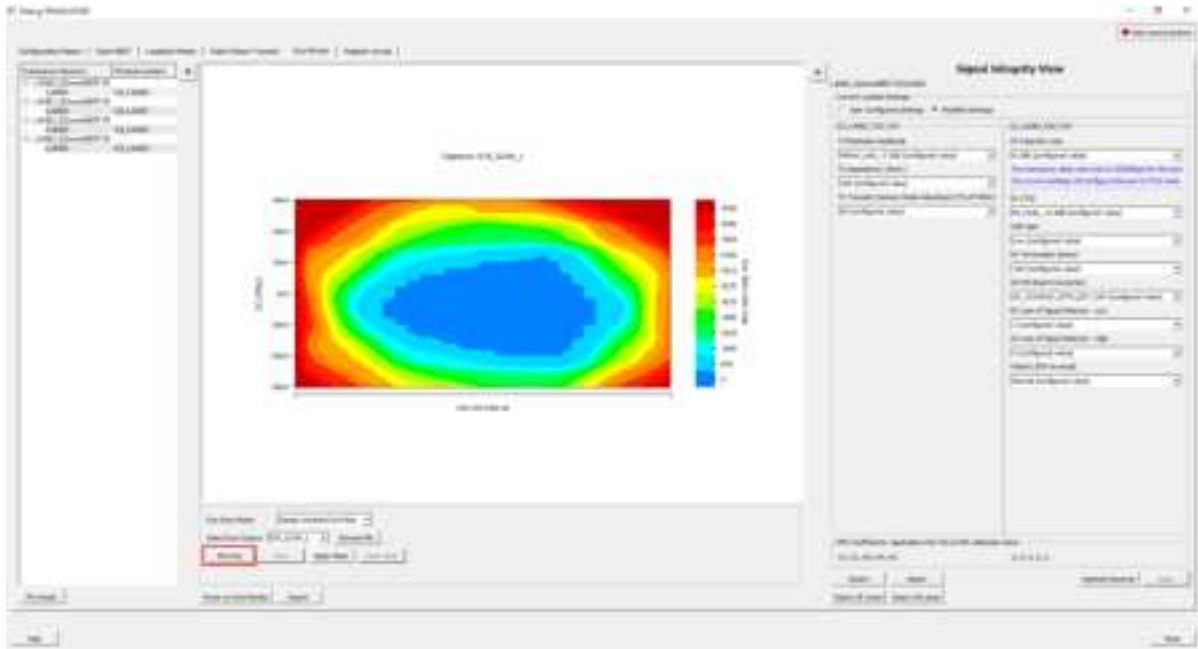
1. Select **Eye Scan Mode Design Initiated Eye Plot**. The **Browse** option becomes enabled.
2. Select the eye plot log files from the dialog box. The **Select Eye Output** drop-down menu is enabled.
3. In the **Select Eye Output** drop-down menu, select a plot by its tag-name. The Plot Eye option is enabled.
4. Click **Plot Eye**. The eye diagram loads on the canvas.

Figure 3-65. Design Initiated Eye Plot Selection



To start eye monitoring for the lane, select Eye output from the Select Eye Output drop-down, and click **Plot Eye**. The Eye diagram appears, as shown in the following figure.

Figure 3-66. Viewing the Eye Scan



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

3.2.2.6.3. Power On/Off Eye Monitor [\(Ask a Question\)](#)

Switching Off Power to the Eye Monitor Core [\(Ask a Question\)](#)

To save power, switch off the power to the eye monitor core.

The default option is provided after reading the status from the device. To power off the eye monitor, click **Power Off Eye Monitor** (the button toggles to **Power on Eye Monitor**). Clicking the button again powers on the eye monitor.



Attention: Eye Plot is not affected if the eye monitor is powered off, because the debug tool internally powers on the eye monitor core and restores the state when the plot is done.

3.2.2.7. Register Access [\(Ask a Question\)](#)

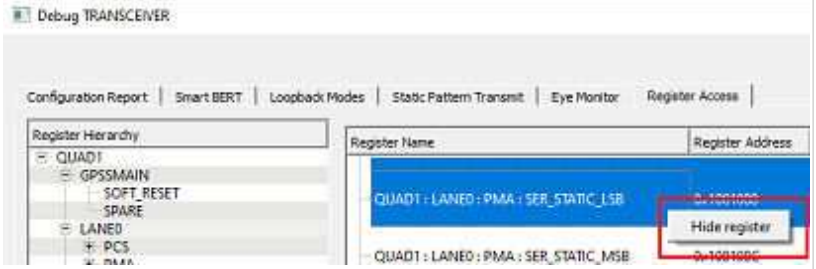
The Register Access page in the Debug TRANSCEIVER window allows you to perform register read, write, export, hide, and export all register operations. The list at the top of the page allows you to show all registers or show only the registers whose read value has been changed. By default, **Show All Registers** is selected. The exported register details are saved in a .CSV file.

Figure 3-67. Debug TRANSCEIVER - Register Access



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.
Access Type	Specifies the register access type. This column also indicates the field access types when a register is expanded.
Field Name	Displays the field name based on the register information read from the <code>PF_XCVR.xml</code> file.
Field Bits	Displays bit value based on the field offset and the width of the register.
Prev Read Value	Displays the previously read value. By default, unread is displayed.
Curr Read Value	Displays a register's current value.
Read Value	Displays the value read from the device. Click the Read button to retrieve the value from the device. By default, unread is displayed.
Write Value (Hexadecimal)	Enter a hexadecimal register value.
Import	Click to load the registers from a <code>.csv</code> file that is exported. This option loads the register list in the selected pane. Any registers selected earlier will be prompted to delete or cancel the load operation.
Delete All	Click to delete all selected registers.
Read	Click to retrieve the register value from the device. Note: The Read button is disabled when no register is selected in the Register Hierarchy tree view or when SmartDebug is running in the Demo mode.
Write	Click to write the user specified register value. Note: The Write button is disabled when no value is specified in the Write Value(Hexadecimal) text box of the selected register or when SmartDebug is running in the Demo mode.

Table 3-3. Debug TRANSCEIVER - Register Access Options (continued)

Option	Description
Export	Click to save the selected register details to a .csv file. When clicked, the Register Access Export Action dialog box appears. Specify the file name and the location of the file in the dialog box. Note: The Export button is disabled when SmartDebug is running in the Demo mode.
Export All	Click to save all the register details to a .csv file. When clicked, the Register Access Export All Action dialog box appears. Specify the file name and the location of the file in the dialog box. 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.
Hide register	Select register(s) and right click to view this option. The Hide registers context menu option helps you remove the selected register(s) from the Register Access table. 

3.2.2.8. Signal Integrity ([Ask a Question](#))

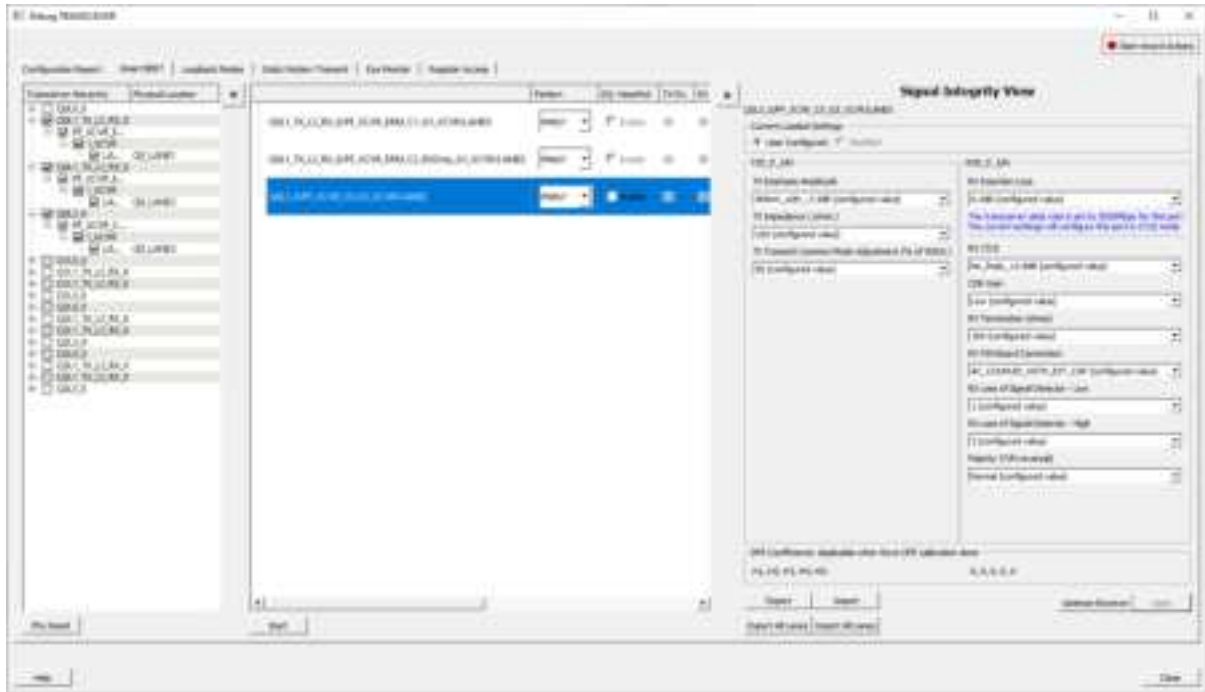
The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of .pdc files.

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.

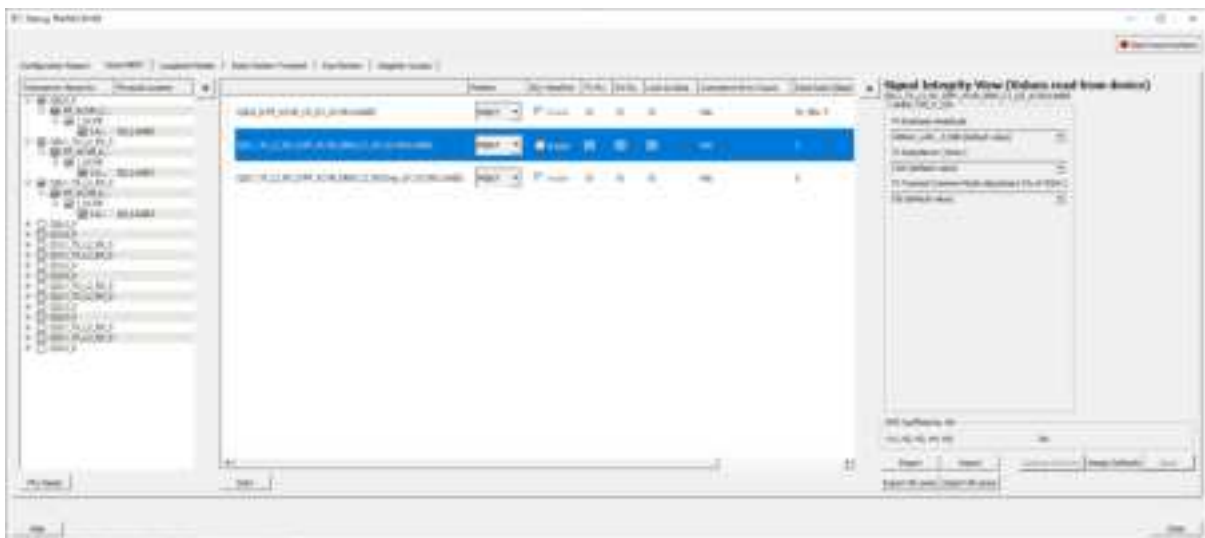
Figure 3-68. Debug TRANSCEIVER - Signal Integrity (Full Duplex Mode - Independent TxRx mode)



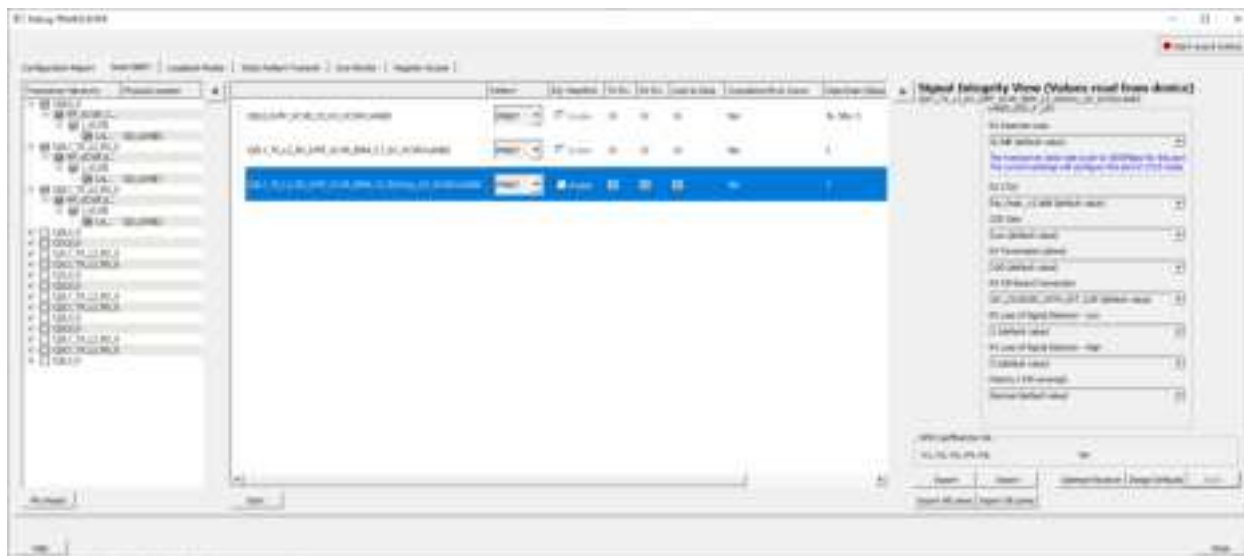
In full duplex mode, all parameters (Tx and Rx) are imported/exported.

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

Figure 3-69. Signal Integrity - Tx Only Mode



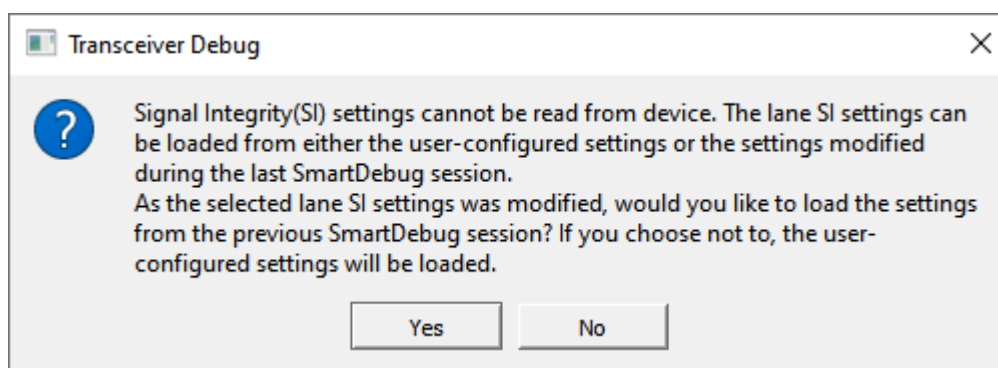
The following figure shows Signal Integrity for Rx Only mode. In this mode, only Rx parameters are imported and exported.

Figure 3-70. Signal Integrity - Rx Only 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.



Attention: If the Signal Integrity Parameters were modified in the previous debug session, the following popup message prompts users to select either modified settings or user-configured settings.



Click **Yes** to load the settings from the previous SmartDebug session or click **No** to load user-configured settings.

The selected lane instance name is displayed in the **Signal Integrity** box, and the **Export** and **Import** 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 **Current Loaded Settings Group** box shows the currently selected settings. You can change the selection to either **User Configured Settings*** or **Modified Settings***, and then click **Apply** to apply the settings on the device.

- **User Configured Settings.** Clicking this radio button loads the Signal Integrity Parameters that were selected in the Libero Design flow run and present in the bitstream.
- **Modified Settings.** Selecting the **Modified Settings** radio box loads the Signal Integrity parameters that were modified in the previous debug sessions.

The **Polarity** (P/N reversal) parameter has been added. You can choose **Normal** or **Inverted** from the drop-down menu. 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, and 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, and Apply flows of Signal Integrity. 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 **Tcl** and click another lane, move to another tab, or click **Import**, **Import All** without applying the changes, the following message appears: *Some Signal Integrity options are modified. How do you wish to continue?*

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



Important: If you change any register setting related to Signal Integrity, power cycle the board, or perform 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 **User Configured** settings from the **Current Loaded Settings Group Box** in SmartDebug, and then click **Apply**. This sets the SI in the **Signal Integrity** tab to the design constraints (SI parameter values chosen from the I/O Editor).

3.2.2.8.1. Export [\(Ask a Question\)](#)

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

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

Export All Lanes [\(Ask a Question\)](#)

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.

3.2.2.8.2. Import [\(Ask a Question\)](#)

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

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

Import All Lanes [\(Ask a Question\)](#)

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.

3.2.2.8.3. Signal Integrity and Calibration Report [\(Ask a Question\)](#)

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
```

3.2.2.8.4. Signal Integrity Parameters in Half Duplex Modes [\(Ask a Question\)](#)

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).

3.2.2.8.5. Optimize Receiver [\(Ask a Question\)](#)

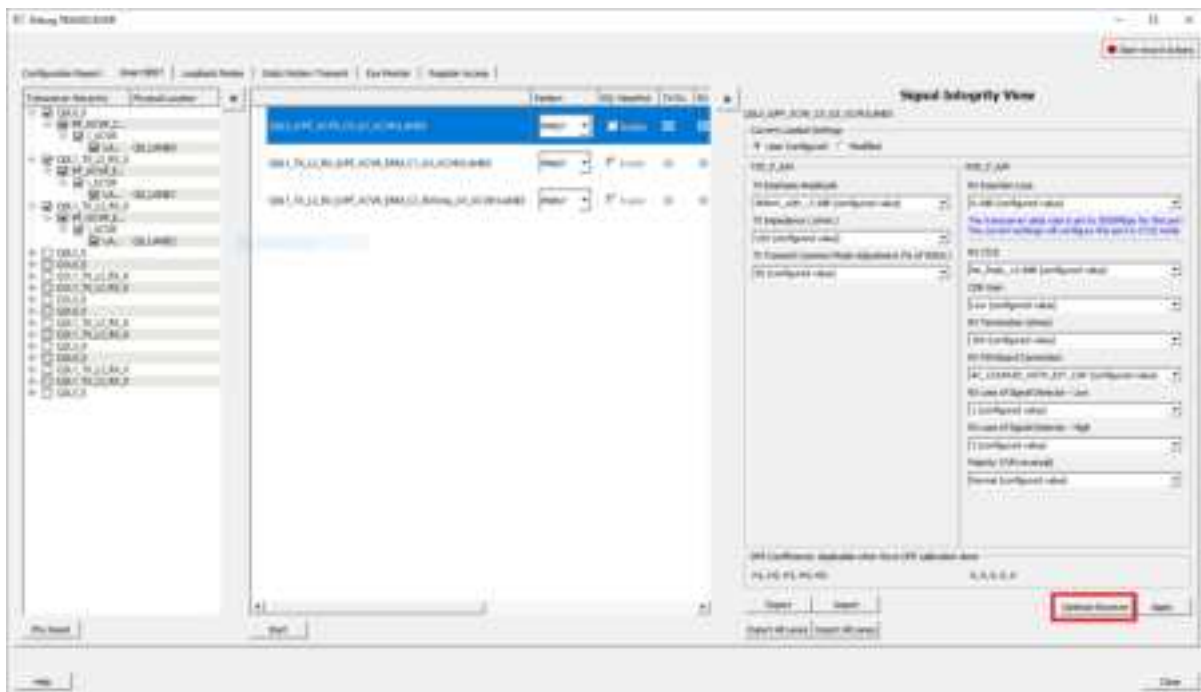
Note: This feature is available for MPF300T, MPF100T, MPF200T, and MPF500T devices.

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 and/or DFE coefficients 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.

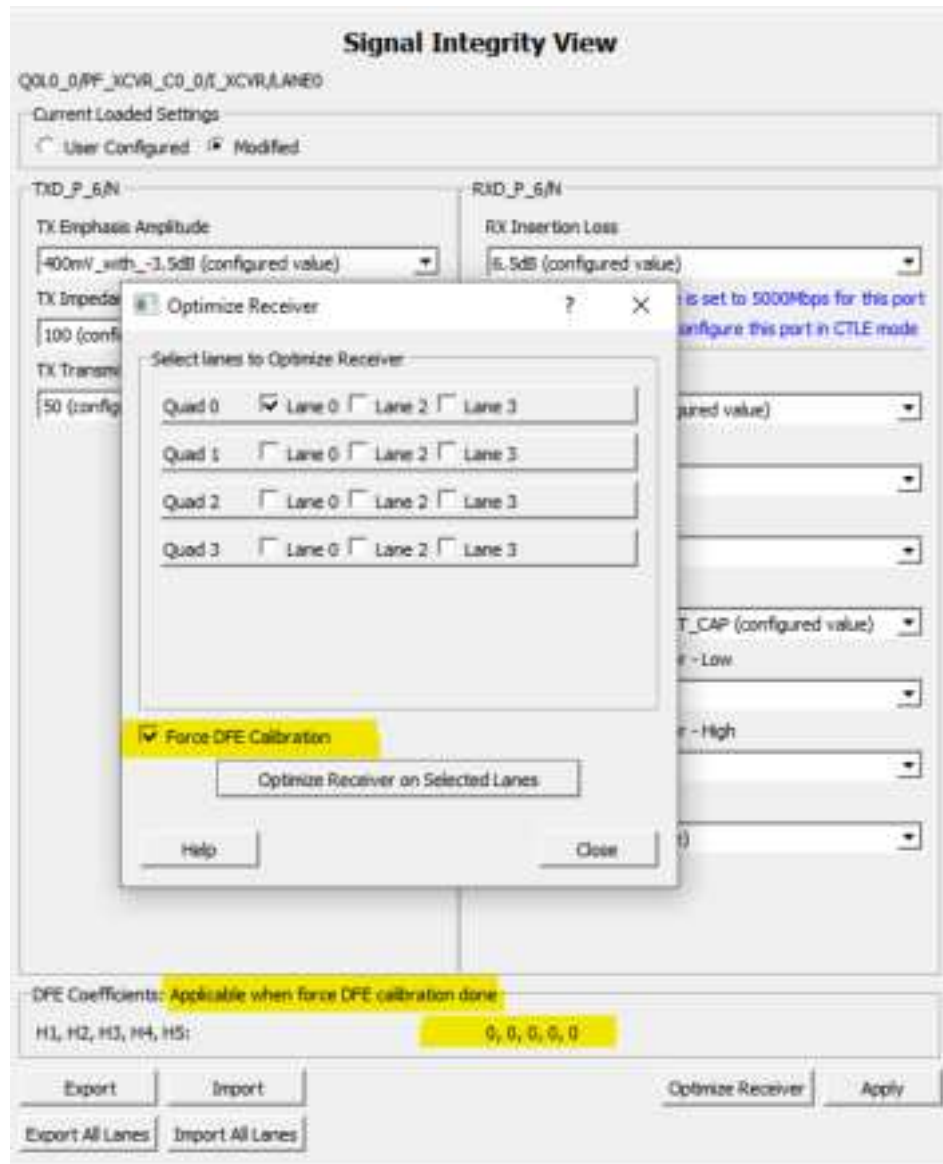
For information about how to use the optimized coefficients without SmartDebug, see the [PolarFire Family Transceiver User Guide](#)

Figure 3-71. Debug TRANSCEIVER—Optimize Receiver



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

Figure 3-72. Optimize Receiver Dialog Box

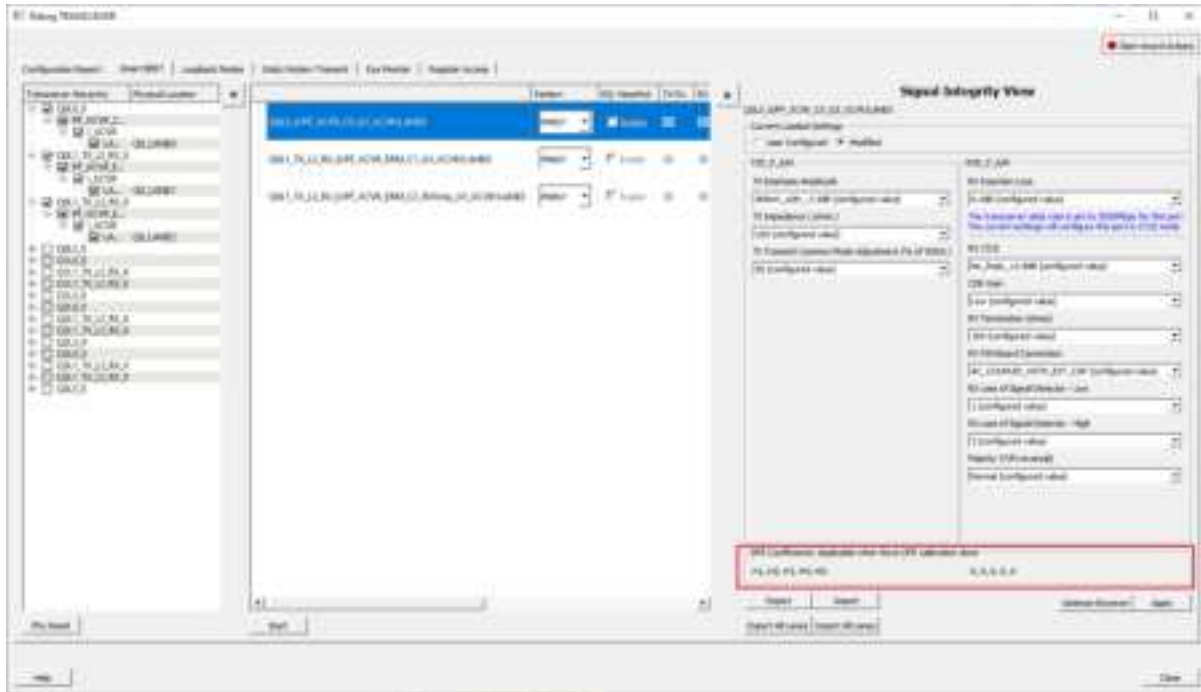


Select the lanes on which to run Optimize Receiver and click **Optimize Receiver on Selected Lanes**. You can select any combination of lanes including those configured in CDR or DFE. The hardware performs Full calibration for DFE mode receivers and may perform CTLE calibration only or CTLE and DFE calibration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

3.2.2.8.6. Display DFE Coefficient Values [\(Ask a Question\)](#)

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 or CDR mode. See the following figure.

Figure 3-73. Signal Integrity Pane—DFE Coefficients



- DFE coefficients are shown for CDR mode receivers if force DFE calibration is selected. This can be done from the check box provided inside the Optimize Receiver dialog box (shown in section [Optimize Receiver](#)).
- 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.

3.2.2.8.7. Exporting DFE Coefficients [\(Ask a Question\)](#)

Optimized DFE coefficient values are saved in a *.txt file when you click **Export**. The text file is saved in the same location where the physical design constraint (*.pdc) file is saved when the **Export** or **Export All** operation is performed.

During the export process, a Physical Design Constraints (*.pdc) file is generated, which includes the `set_io` command along with the DFE Coefficient parameters such as `RX_DFE_COEFFICIENT_H1`.

 **Important:** Coefficients in the Physical Design Constraints file are initialized to their default settings. To update these values, move the optimized DFE coefficient values from the Signal Integrity and Calibration report (contained in the *.txt file) to the *.pdc file, ensuring they are arranged in the correct sequence, from H1 to H5.

3.2.2.9. PCIe Debug [\(Ask a Question\)](#)

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.


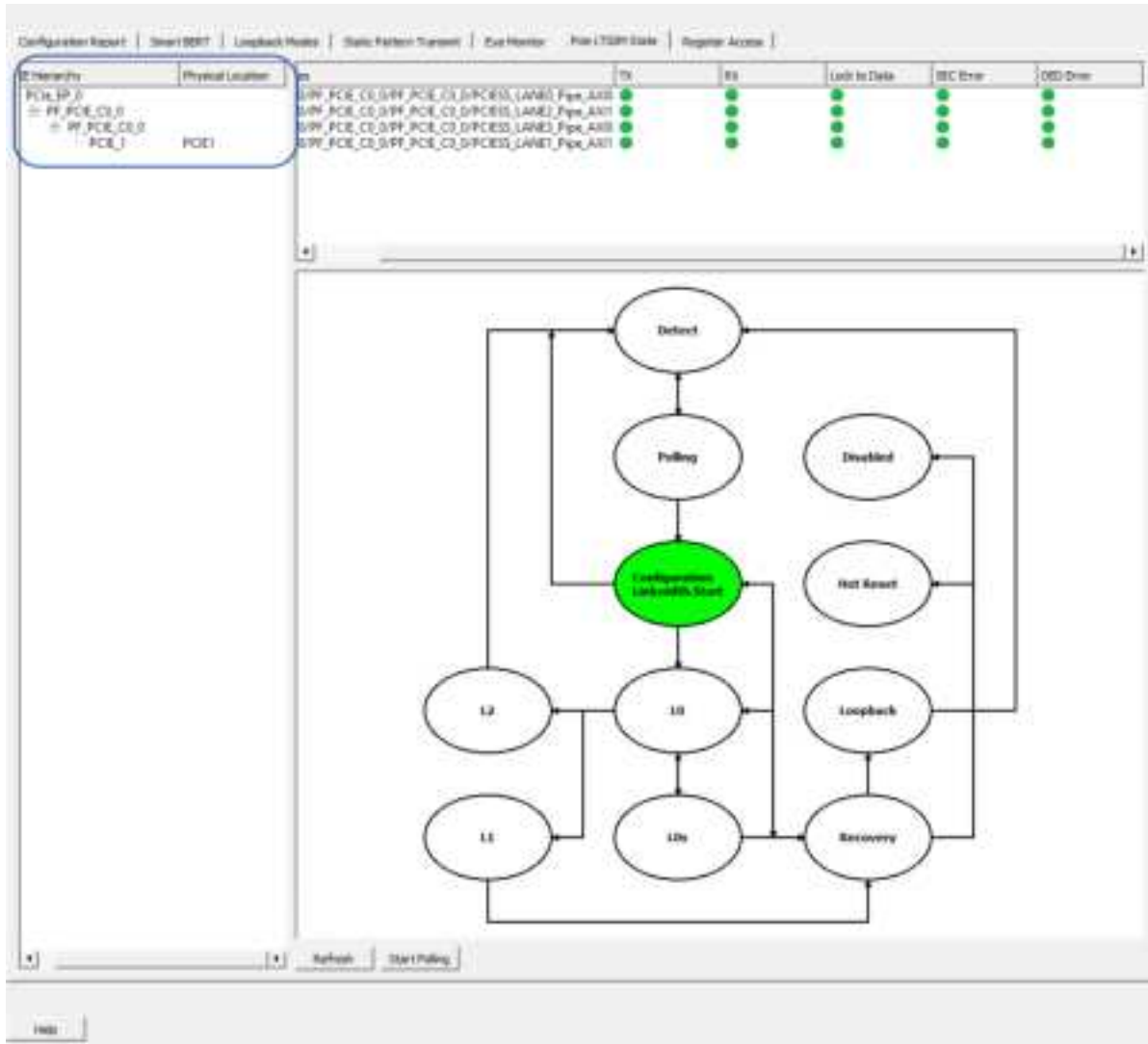
 **Attention:** All parameters shown on the PCIe LTSSM State page are read-only and cannot be changed.

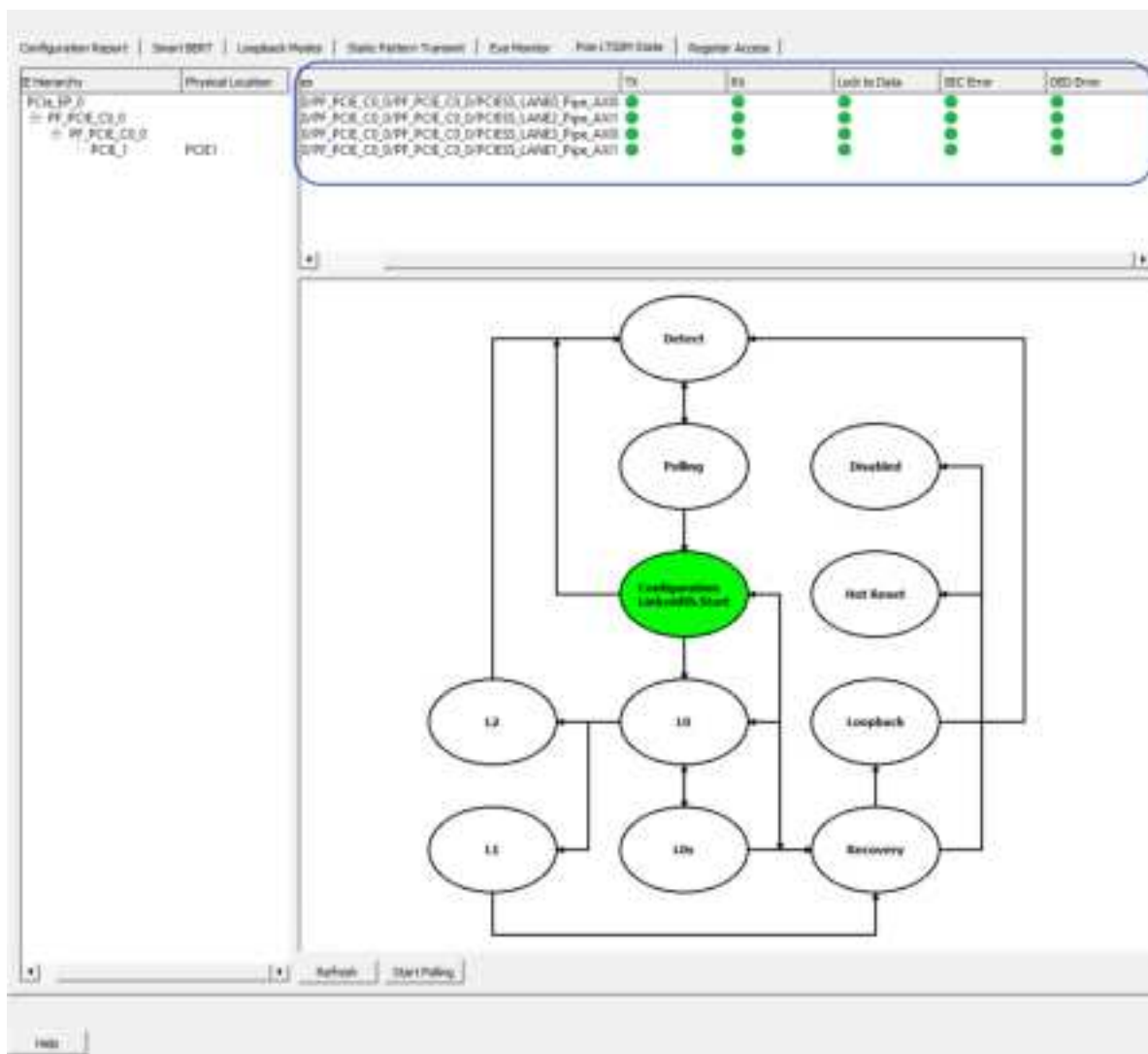
Figure 3-74. PCIe Design Hierarchy



3.2.2.9.1. Lane Status and Lane Link Error Status [\(Ask a Question\)](#)

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.

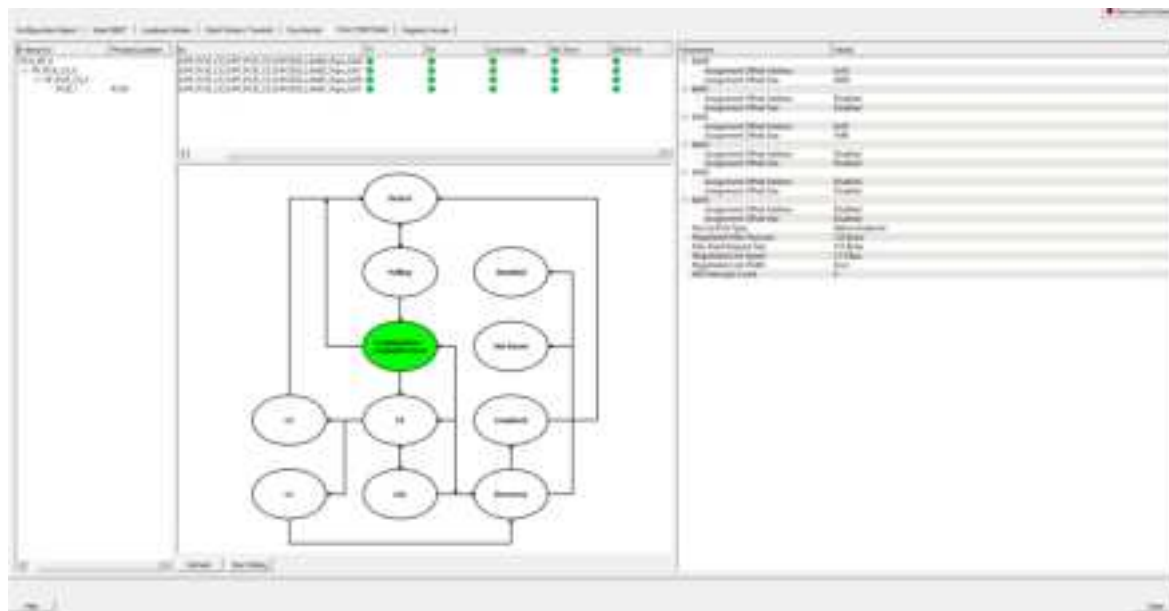
Figure 3-75. Lane Status and Lane Link Error Status



3.2.2.9.2. LTSSM State Machine [\(Ask a Question\)](#)

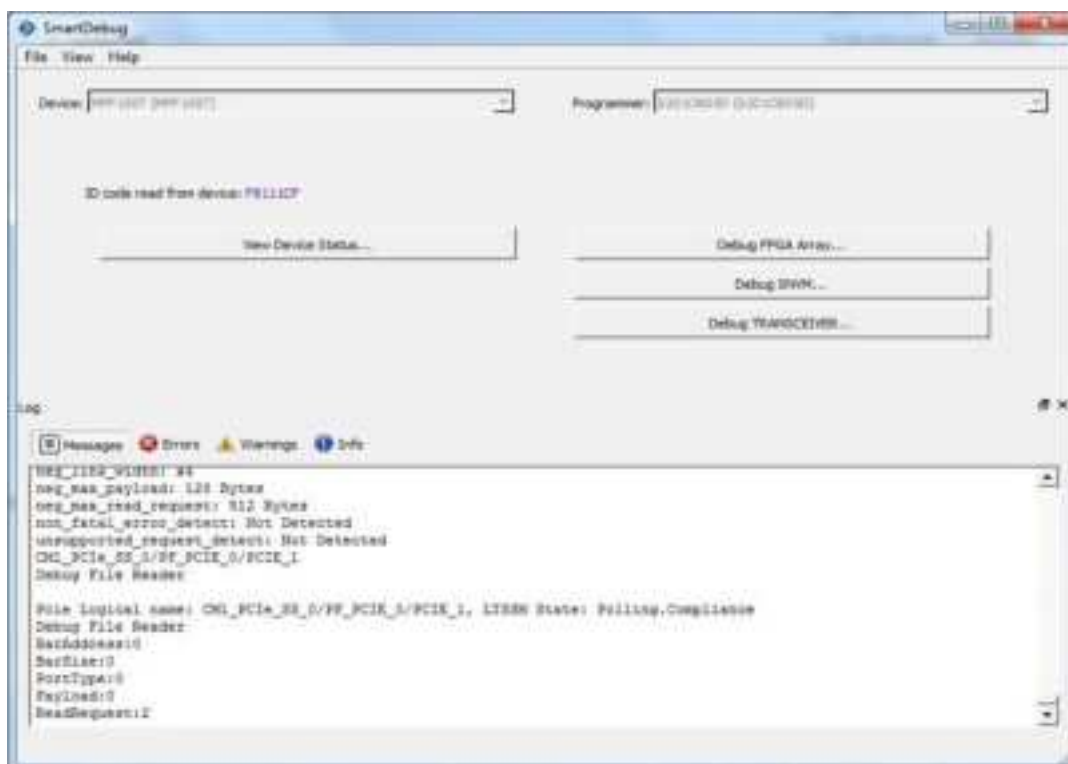
The Pcie LTSSM State page shows the LTSSM state machine to the right of PCIe Design Hierarchy and under the lane status, as shown in the following figure.

Figure 3-76. 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 3-77. Active State and Substate Information



3.2.2.9.3. Config Space Parameter Information [\(Ask a Question\)](#)

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 3-78. Config Space Parameter Information

Parameters	Values
BAR0	
Assignment Offset Address	0x00
Assignment Offset Size	64KB
BAR1	
Assignment Offset Address	Disabled
Assignment Offset Size	Disabled
BAR2	
Assignment Offset Address	0x00
Assignment Offset Size	1MB
BAR3	
Assignment Offset Address	Disabled
Assignment Offset Size	Disabled
BAR4	
Assignment Offset Address	Disabled
Assignment Offset Size	Disabled
BAR5	
Assignment Offset Address	Disabled
Assignment Offset Size	Disabled
Device/Port Type	Native Endpoint
Negotiated Max Payload	128 Bytes
Max Read Request Size	512 Bytes
Negotiated Link Speed	2.5 Gbps
Negotiated Link Width	Error
MSI Interrupt Count	4

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

Figure 3-79. Config Space Parameter Information in the SmartDebug Log Window

```

[2025-01-01 10:00:00] INFO: PCIe Config Space Data for Parameters: BAR0, BAR1, BAR2, BAR3, BAR4, BAR5, Device/Port Type, Negotiated Max Payload, Max Read Request Size, Negotiated Link Speed, Negotiated Link Width, MSI Interrupt Count
[2025-01-01 10:00:00] INFO: Config Space Data
[2025-01-01 10:00:00] INFO: BAR0: Assignment Offset Address: 0x00, Assignment Offset Size: 64KB
[2025-01-01 10:00:00] INFO: BAR1: Assignment Offset Address: Disabled, Assignment Offset Size: Disabled
[2025-01-01 10:00:00] INFO: BAR2: Assignment Offset Address: 0x00, Assignment Offset Size: 1MB
[2025-01-01 10:00:00] INFO: BAR3: Assignment Offset Address: Disabled, Assignment Offset Size: Disabled
[2025-01-01 10:00:00] INFO: BAR4: Assignment Offset Address: Disabled, Assignment Offset Size: Disabled
[2025-01-01 10:00:00] INFO: BAR5: Assignment Offset Address: Disabled, Assignment Offset Size: Disabled
[2025-01-01 10:00:00] INFO: Device/Port Type: Native Endpoint
[2025-01-01 10:00:00] INFO: Negotiated Max Payload: 128 Bytes
[2025-01-01 10:00:00] INFO: Max Read Request Size: 512 Bytes
[2025-01-01 10:00:00] INFO: Negotiated Link Speed: 2.5 Gbps
[2025-01-01 10:00:00] INFO: Negotiated Link Width: Error
[2025-01-01 10:00:00] INFO: MSI Interrupt Count: 4

```

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

The following table describes the config space parameters.

Table 3-4. Config Space Parameter Descriptions

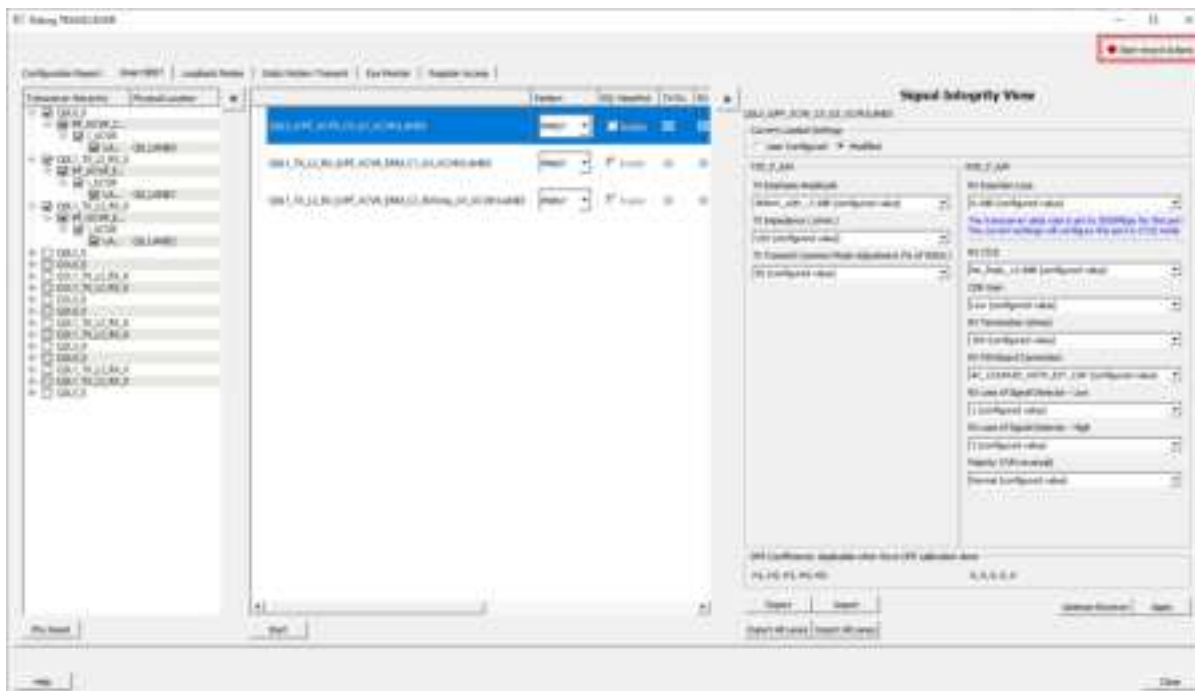
PCIe Config Space Parameters		Description
1	BAR0	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.
2	BAR1	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.

Table 3-4. Config Space Parameter Descriptions (continued)

PCIe Config Space Parameters		Description
3	BAR2	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.
4	BAR3	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.
5	BAR4	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.
6	BAR5	—
	Assignment Address Offset	Defines the starting address of the address translation space.
	Assignment Offset Size	Defines the address translation space size.
7	Device/Port Type	PCIe Port Type
8	Negotiated Max PayLoad	Negotiated maximum Payload of the PCIe link
9	Max Read Request Size	Negotiated maximum read request size of the PCIe link
10	Negotiated Link Speed	Negotiated link speed of the PCIe link. Supported values are: <ul style="list-style-type: none"> • 2.5 Gbps • 5.0 Gbps • 8.0 Gbps
11	Negotiated Link Width	Negotiated link width of the PCIe link. Supported values are: <ul style="list-style-type: none"> • x1 • x2 • x4 • x8 • x16
12	MSI Interrupt Count	Number of Message Signaled Interrupts (MSI) messages. This parameter is not applicable for Rootport. Range: 1–128

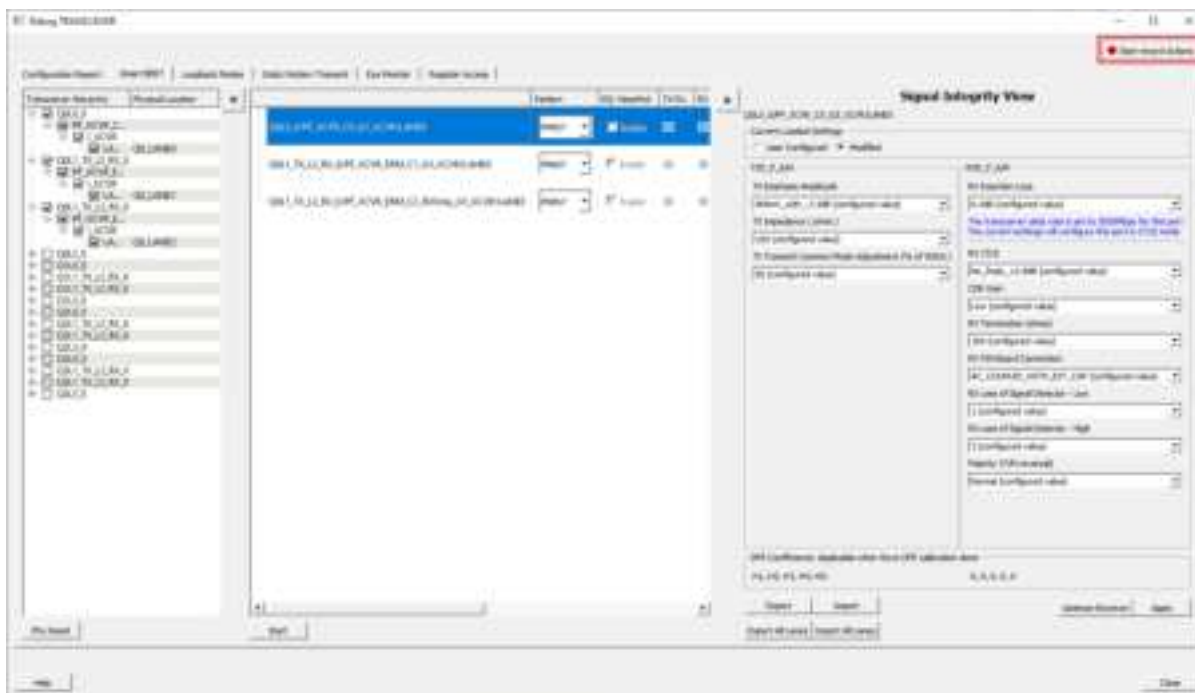
3.2.2.10. Record Actions ([Ask a Question](#))

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 3-80. Debug TRANSCEIVER Window Showing Start Record Actions Option

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

When you click **Stop recording...** to stop the recording, a window prompts you to save the output a text (.txt) file. After saving the file, the Debug TRANSCEIVER window reverts to default state.

Figure 3-81. Save Recorded Action Window Pop-Up to Save Text File After 'Stop recording...' is Clicked

3.2.2.10.1. Recorded Content [\(Ask a Question\)](#)

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:
 - a. The lane selected on the **SmartBERT** tab is Q0_ANE0 and the pattern selected is XCVR PMA PRBS7.
 - b. The first set of lines has the following four subheadings:

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

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

Write Mask: Write mask is an 8 hexadecimal-digit number that 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.
 - c. End of start PRBS test sequence denotes that the sequence for that lane is completed and the next sequence is initiated.

Figure 3-82. 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
- Signal Integrity
- 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
- PRBS tests from SmartBERT IP

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 3-83. 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

```

3.2.3. MSS Register Access (PolarFire SoC) [\(Ask a Question\)](#)

For PolarFire SoC devices, 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.

The screenshot shows the HSI Register Access tool with the 'Register Operations' tab selected. The tool displays a list of registers with their names, addresses, sizes, and values. The 'Register Name' column lists various registers like AD_CONVERT, AD_CONVERT_0, AD_CONVERT_1, etc. The 'Register Address' column shows the hexadecimal address for each register. The 'Register Size' column shows the size in bytes. The 'Register Value' column shows the current value of the register. The 'Write Value (Hexadecimal)' column shows the value to be written to the register. The tool also includes a 'Register Selection' tab and a 'Register Filter' field.

Register Name	Register Address	Register Type	Read Size	Write Size	Write Value (Hexadecimal)
AD_CONVERT	0x00000000	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_0	0x00000001	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_1	0x00000002	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_2	0x00000003	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_3	0x00000004	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_4	0x00000005	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_5	0x00000006	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_6	0x00000007	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_7	0x00000008	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_8	0x00000009	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_9	0x0000000A	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_10	0x0000000B	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_11	0x0000000C	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_12	0x0000000D	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_13	0x0000000E	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_14	0x0000000F	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_15	0x00000010	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_16	0x00000011	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_17	0x00000012	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_18	0x00000013	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_19	0x00000014	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_20	0x00000015	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_21	0x00000016	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_22	0x00000017	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_23	0x00000018	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_24	0x00000019	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_25	0x0000001A	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_26	0x0000001B	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_27	0x0000001C	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_28	0x0000001D	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_29	0x0000001E	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_30	0x0000001F	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_31	0x00000020	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_32	0x00000021	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_33	0x00000022	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_34	0x00000023	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_35	0x00000024	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_36	0x00000025	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_37	0x00000026	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_38	0x00000027	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_39	0x00000028	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_40	0x00000029	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_41	0x0000002A	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_42	0x0000002B	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_43	0x0000002C	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_44	0x0000002D	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_45	0x0000002E	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_46	0x0000002F	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_47	0x00000030	read-write	32-bit	32-bit	0x00000000
AD_CONVERT_48	0x00000031				

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

Option	Description
Register Selection	<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 are always shown in the pane as they are the configuration registers and are always accessible.</p> <ul style="list-style-type: none"> • AXISW • MPUCFG • ENVMCFG • IOSCBCFG • PFSOC_MSS_TOP_SCB_REGS • PFSOC_MSS_TOP_SYSREG
Register Filter	<p>Allows you to search for instances/register by their names. Supports wildcard (*) searches.</p> <ul style="list-style-type: none"> • The search is real time. • Enter an asterisk (*) to display all registers having instance names expanded. • Names entered without asterisk look for exact match.
Register Hierarchy	<p>Displays Instance name and Register name in a two-level hierarchy-level format.</p> <ul style="list-style-type: none"> • You can select multiple names. • Right click and select 'Add' to add the registers to the operation pane. • Drag and drop the register names to the Register Operation pane.

MSS Register Access (PolarFire SoC) (continued)

Option	Description
Export All	Exports information of all registers that are displayed in the selection pane to the specified .csv file. <ul style="list-style-type: none"> The exported .csv file contains six sections: <ul style="list-style-type: none"> Register Name Register Address (hexadecimal number) Register Field Name Register Field Range Register Field Value (hexadecimal number) Register Value (hexadecimal number) Comments (message to show if read register was successful or not)
Register Operations	Lists the registers selected for access operations such as read, write, and export. <p>Note: 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"> The Register Operations pane has six columns - Name, address, access type, field bits, read value, and write value. The register has a sub-tree where all the register fields are listed. Select Hide Register from the context menu to remove the selected register. <p>Note: You cannot remove a field inside the register.</p> Write value column is populated with 0x to indicate that it supports hexadecimal values. <ul style="list-style-type: none"> Click a specific register row under the write column to edit the value. The text entered is validated based on the range (field bits) of a field or a register. <p>Note: Non-hexadecimal character will not be inserted.</p>
Import	Click to load the registers from a .csv file that is exported. This option loads the register list in the selected pane. Any registers selected earlier will be prompted to delete or cancel the load operation..
Delete All	Click to delete all selected registers.
Read	Read all selected registers by their register names and display the values in the hexadecimal format.
Write	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"> 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 is ignored. 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. You cannot edit the write field if the register has read-only access type. Write onto a field is a read-modify-write operation.
Export	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"> Register Name Register Address (hexadecimal number) Register Field Name Register Field Range Register Field Value (hexadecimal number) Register Value (hexadecimal number)

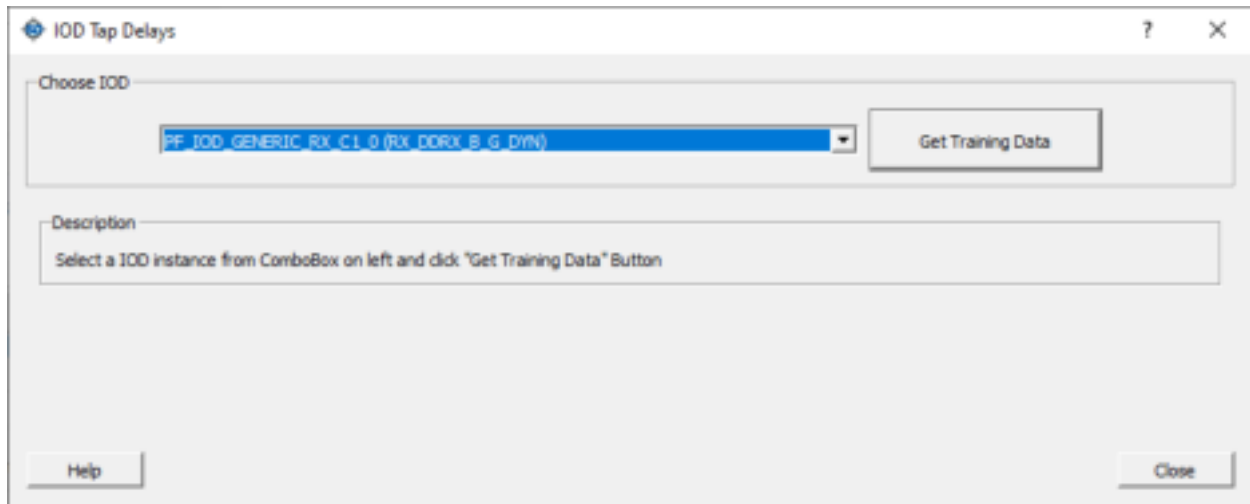
3.2.4. Debug IOD ([Ask a Question](#))

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.

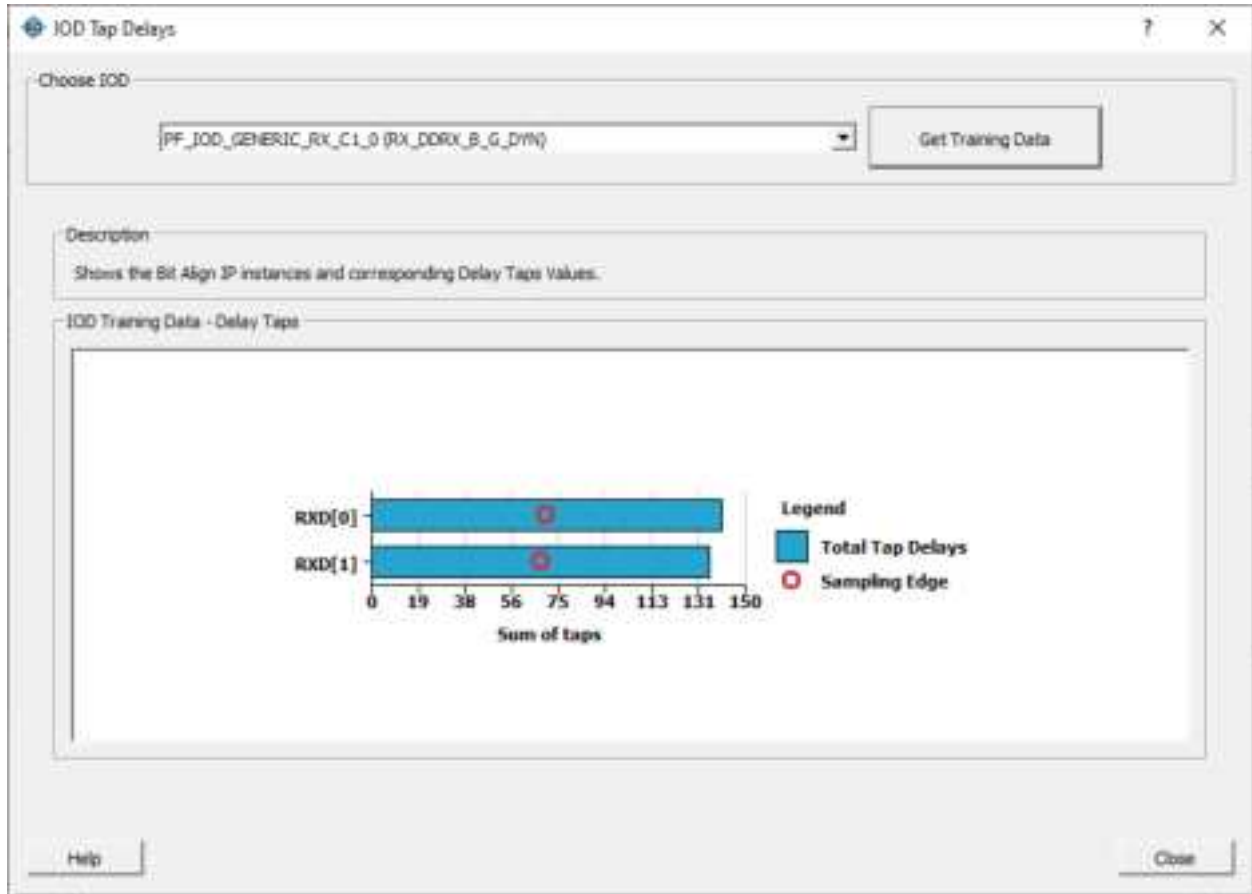
Figure 3-85. IOD Tap Delays Window



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

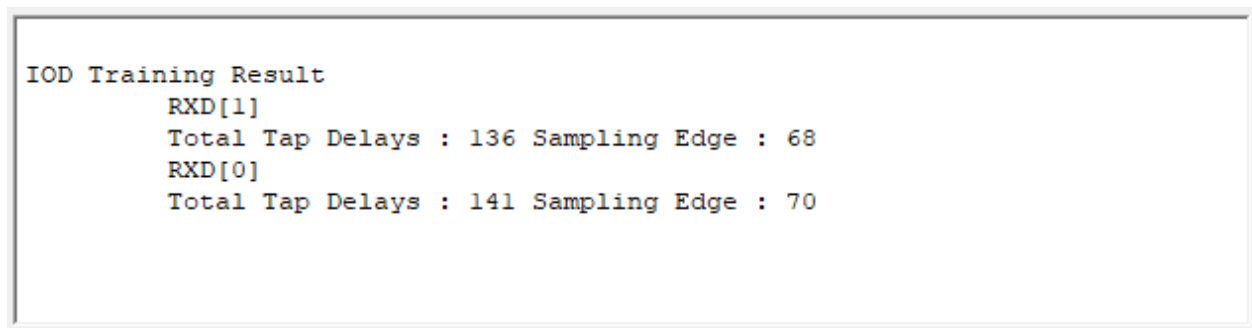
➔ Important: CORERXIIODBITALIGN IP must have the output debug pins either connected or promoted to the top for SmartDebug to detect and identify the debug signals.

Figure 3-86. IOD Tap Delays—Eye Width and Sample Edge Data Representation



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

Figure 3-87. SmartDebug Log Window—Training Successful



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

Figure 3-88. IOD Tap Delays Window—Training Failed



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

Figure 3-89. SmartDebug Log Window—Training Failed



3.2.5. Debug UPROM [\(Ask a Question\)](#)

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

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

If a μ PROM memory block is used in the Liberio design, the **UPROM Debug** window appears.

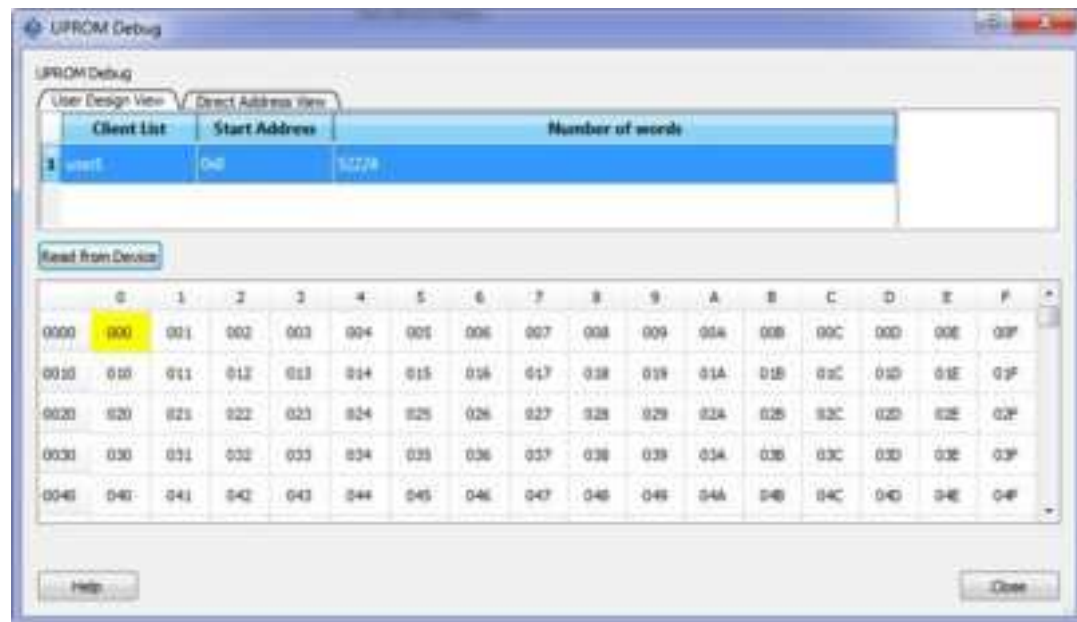
Figure 3-90. UPROM Debug Window



3.2.5.1. User Design View [\(Ask a Question\)](#)

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

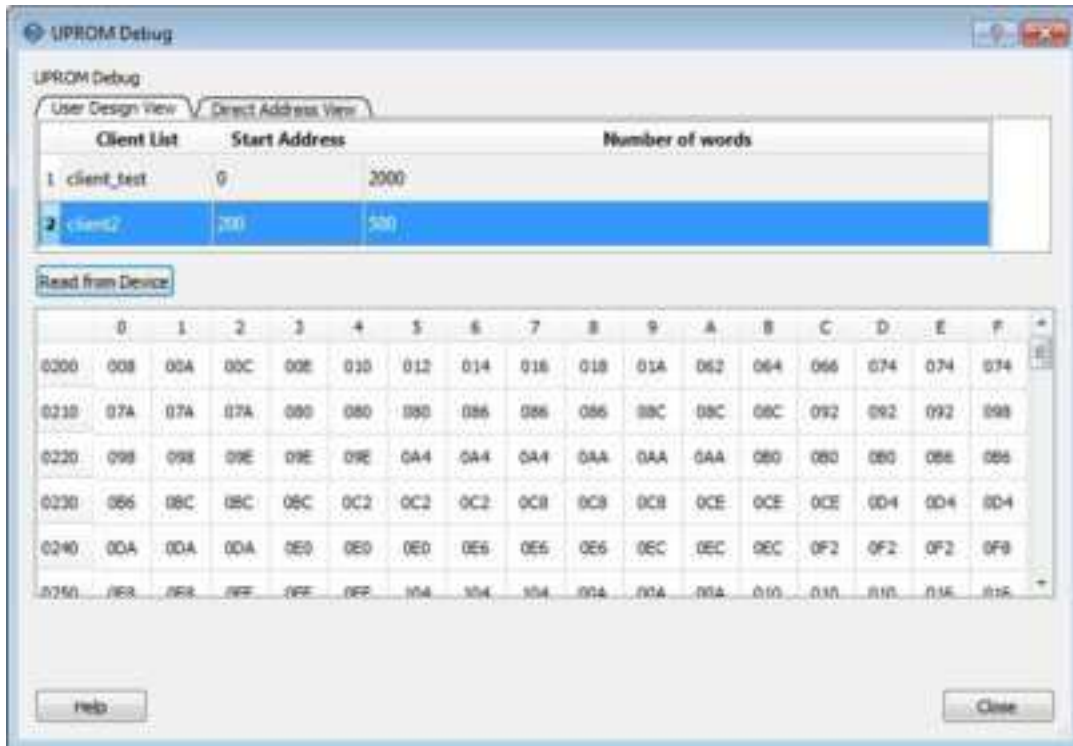
Click the **Read from Device** button to display a table showing the data in the location at the selected client address. See the following figure.



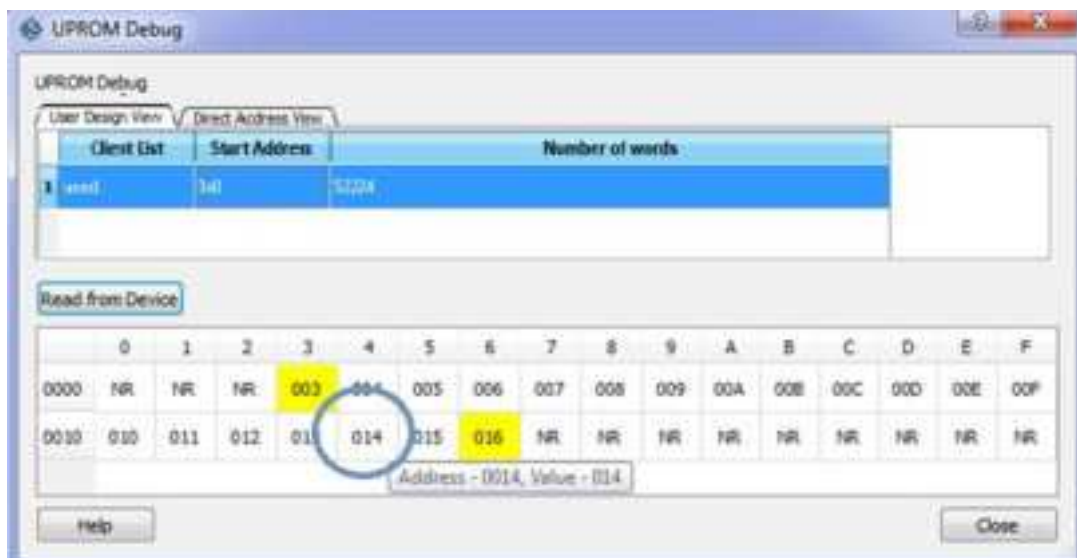
The Client address is associated with *Start Address* and *Number of 9-bit words*. Therefore, the table will contain as many locations as the number of 9-bit words.

In the preceding example, *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 your cursor over a cell to see the cell's address and value, as shown in the following figure.



3.2.5.2. Direct Address View [\(Ask a Question\)](#)

The **Direct Address View** tab in the μ PROM Debug window provides access to μ 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 your cursor over the icon.

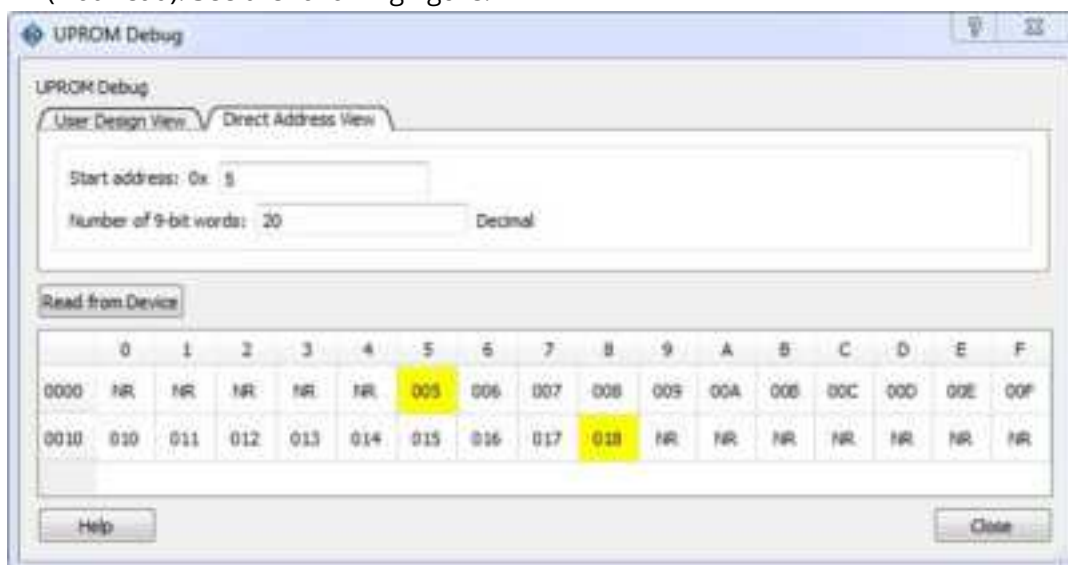
Number of 9-bit words: Positive integer value

Values are validated and errors are indicated by a red "**STOP**". The error message displays when you hover your cursor 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**". The error message displays when you hover your cursor 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: Observe the following:

- When one field is entered, both fields are validated and the **Read from Device** button is enabled.
- 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 **μ PROM Debug** window is closed and reopened, the session is retained. The μ PROM Debug session is lost only if the main **SmartDebug** window is closed.

3.2.5.3. Demo Mode [\(Ask a Question\)](#)

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

3.2.6. Debug Fabric DDR IO Margin [\(Ask a Question\)](#)

To access the Debug Fabric DDR IO margin feature in SmartDebug, click **Debug Fabric 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.

FPGA memory controller sub-systems train the DDR interface channel for optimal signal integrity in both the time and voltage domains. The memory controller goes through various training phases to adjust and compensate for unbalanced loading. You must be familiar with the start-up and training phases used with external memory interfaces. The Fabric DDR IO Margin feature assists users to analyze and understand the interface results of the training algorithms. For more information on the memory controller training and optimization, see the [PolarFire® FPGA and PolarFire® SoC Memory Controller User Guide](#).

Figure 3-91. SmartDebug Main Window with Debug Fabric DDR IO Margin Option



When you click **Debug Fabric DDR Memory...** in the main SmartDebug window, the following **Fabric DDR IO Margin** dialog box opens, as shown in the following figure.

Figure 3-92. Fabric DDR IO Margin Dialog Box with Options Disabled



Initially, all options in the **Fabric DDR IO Margin** dialog box are disabled. To view the training data for specific DDR instances, select the desired instance and click the **Get Training Data** button. Clicking this button runs a script that fetches the training data, which takes approximately two minutes. During this time, the **Fabric DDR IO Margin** dialog box displays information based on the selected DDR instance, as shown in the following figures. **First stage, Overall Training Status** is marked as **Passed** if all subsequent stages are completed successfully. If any stage fails, the status is marked as **Failed**.

Figure 3-93. Fabric DDR IO Margin Dialog Box Showing Information for First IO Bank

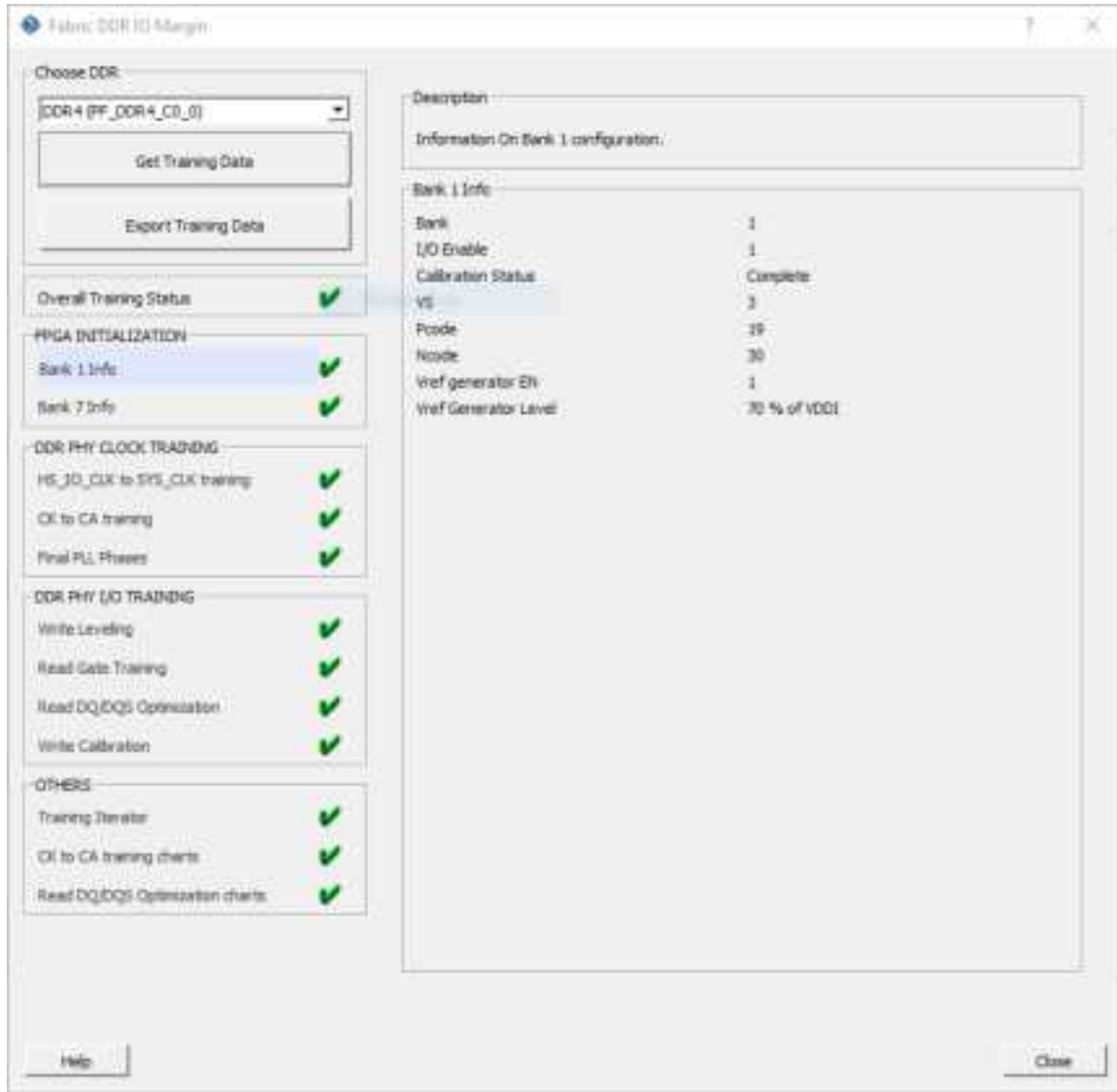


Figure 3-94. Fabric DDR IO Margin Dialog Box Showing Information for Second IO Bank

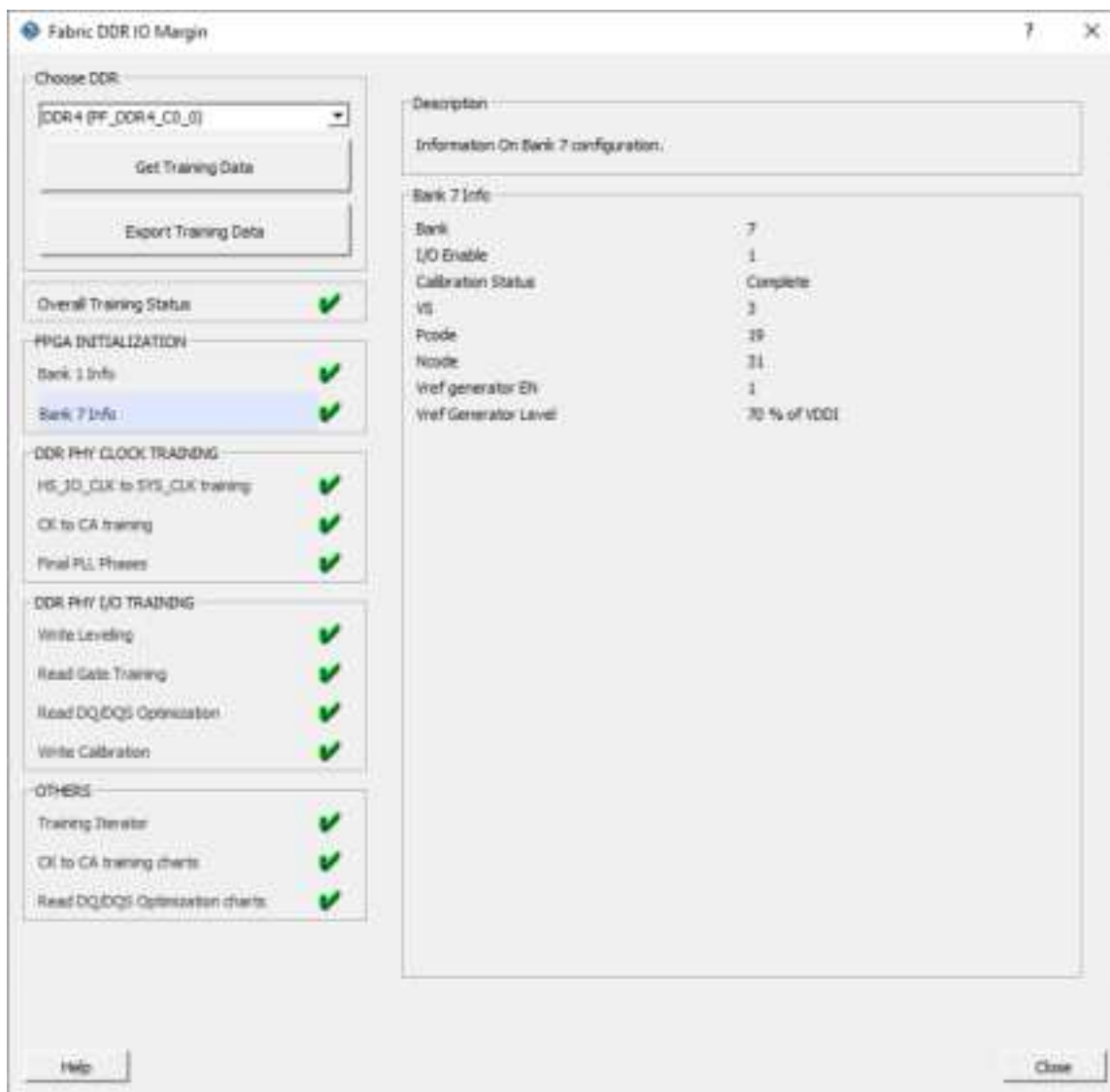


Figure 3-95. Fabric DDR IO Margin Dialog Box Showing HK_IO_CLK-to-SYS_CLK Training

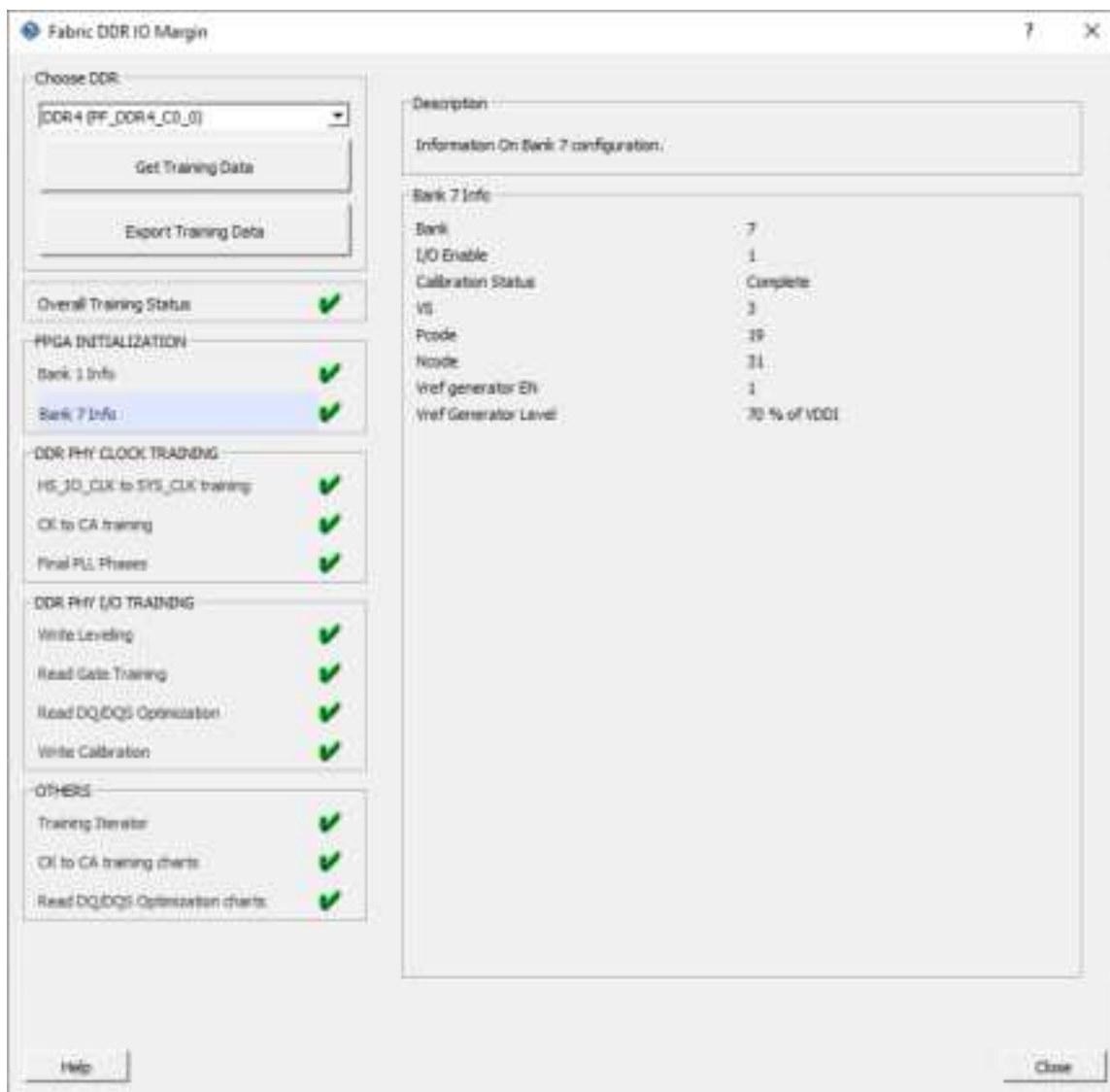


Figure 3-96. Fabric DDR IO Margin Dialog Box Showing CK-to-CA Training

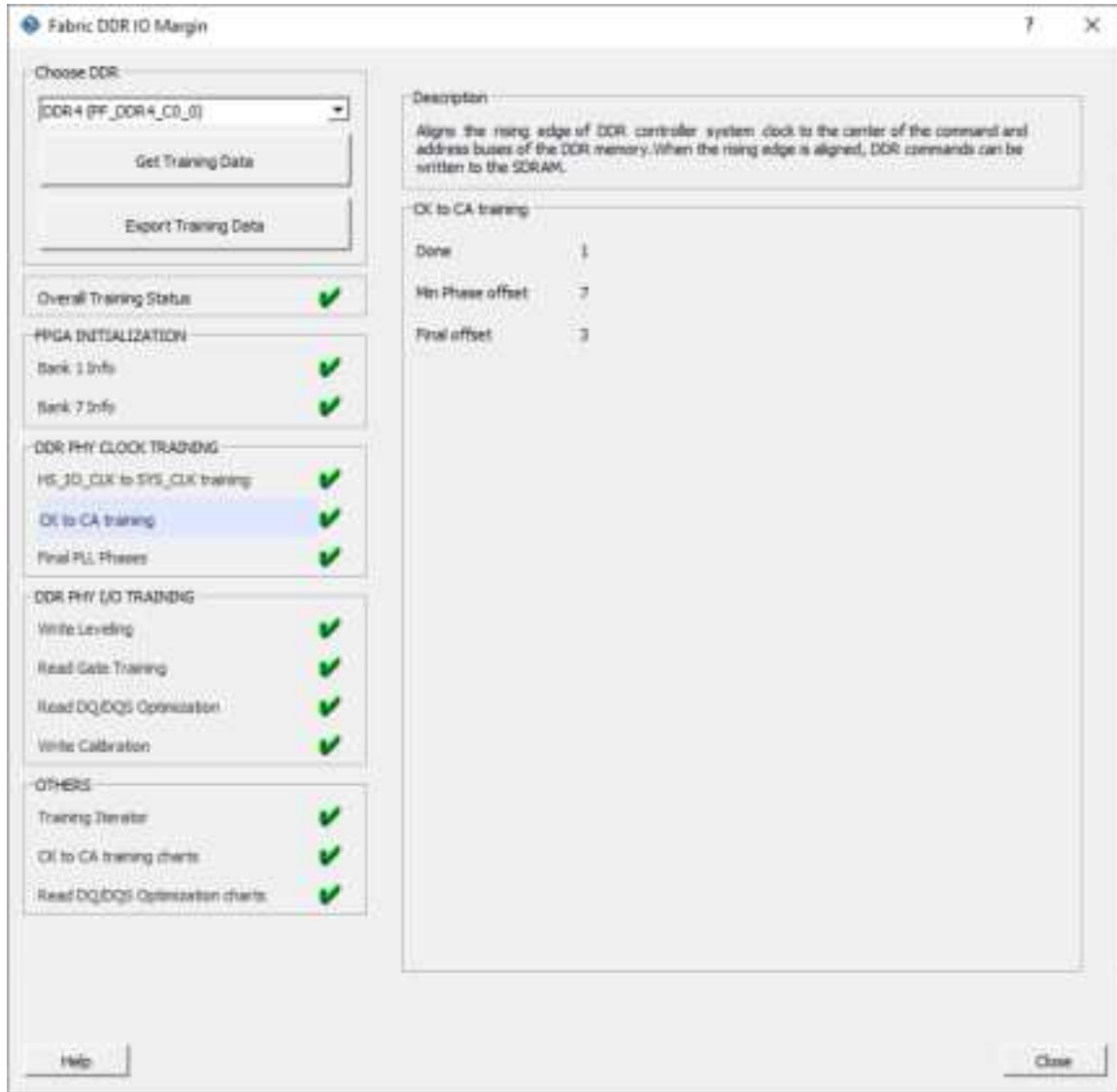


Figure 3-97. Fabric DDR IO Margin Dialog Box Showing Final PLL Phases

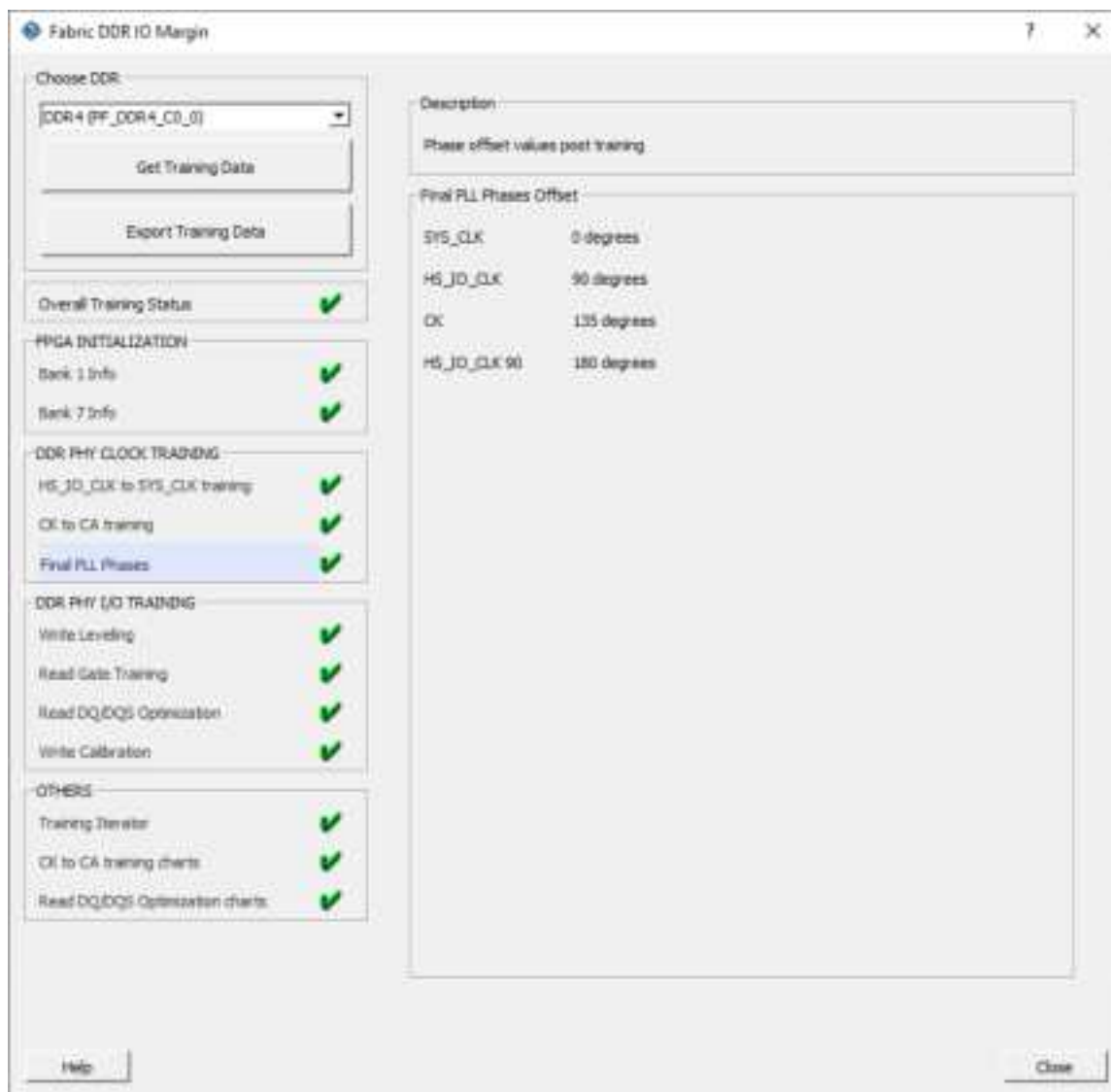


Figure 3-98. Fabric DDR IO Margin Dialog Box Showing Write Leveling Data

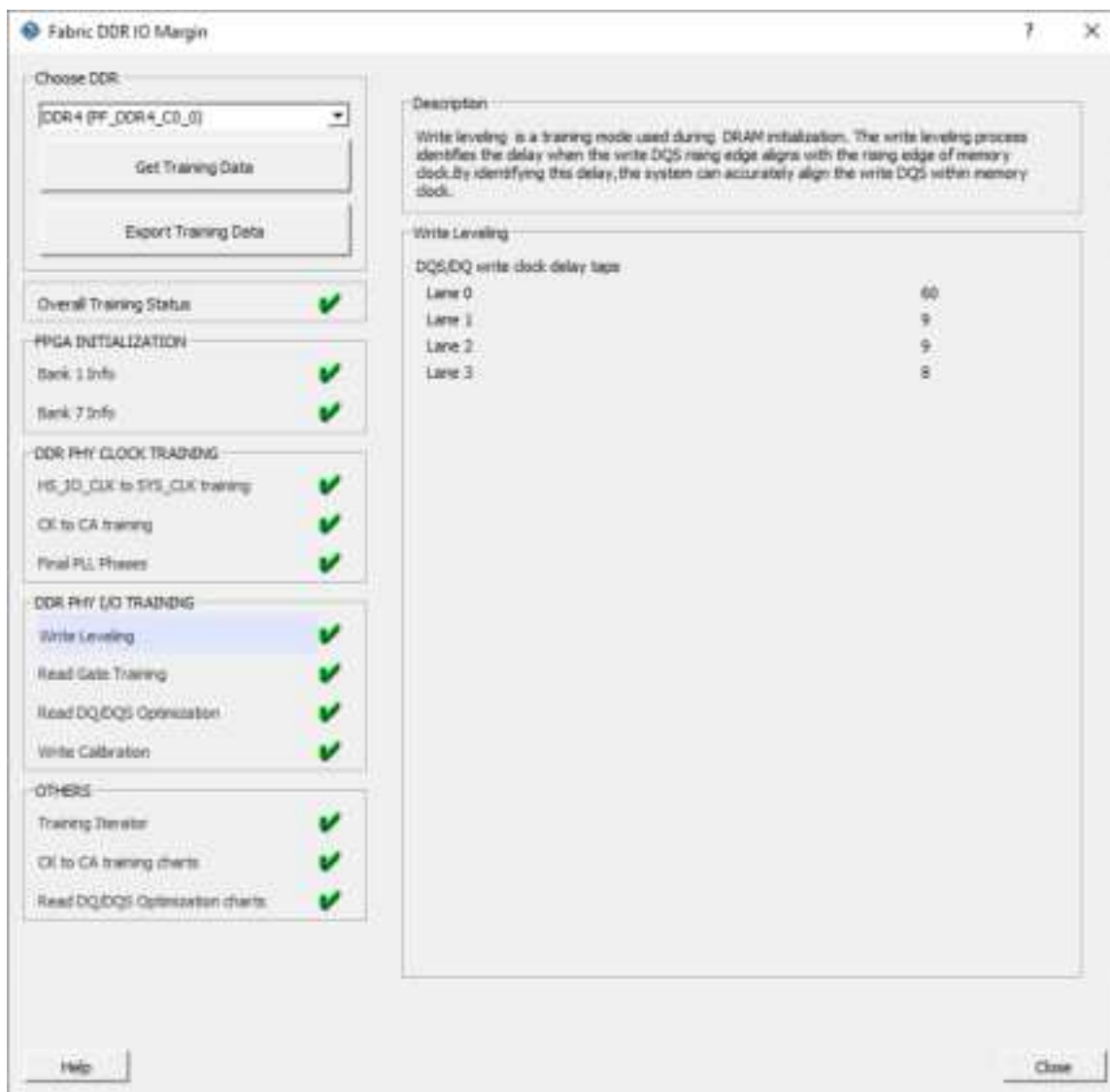


Figure 3-99. Fabric DDR IO Margin Dialog Box Showing Read Gate Training Data

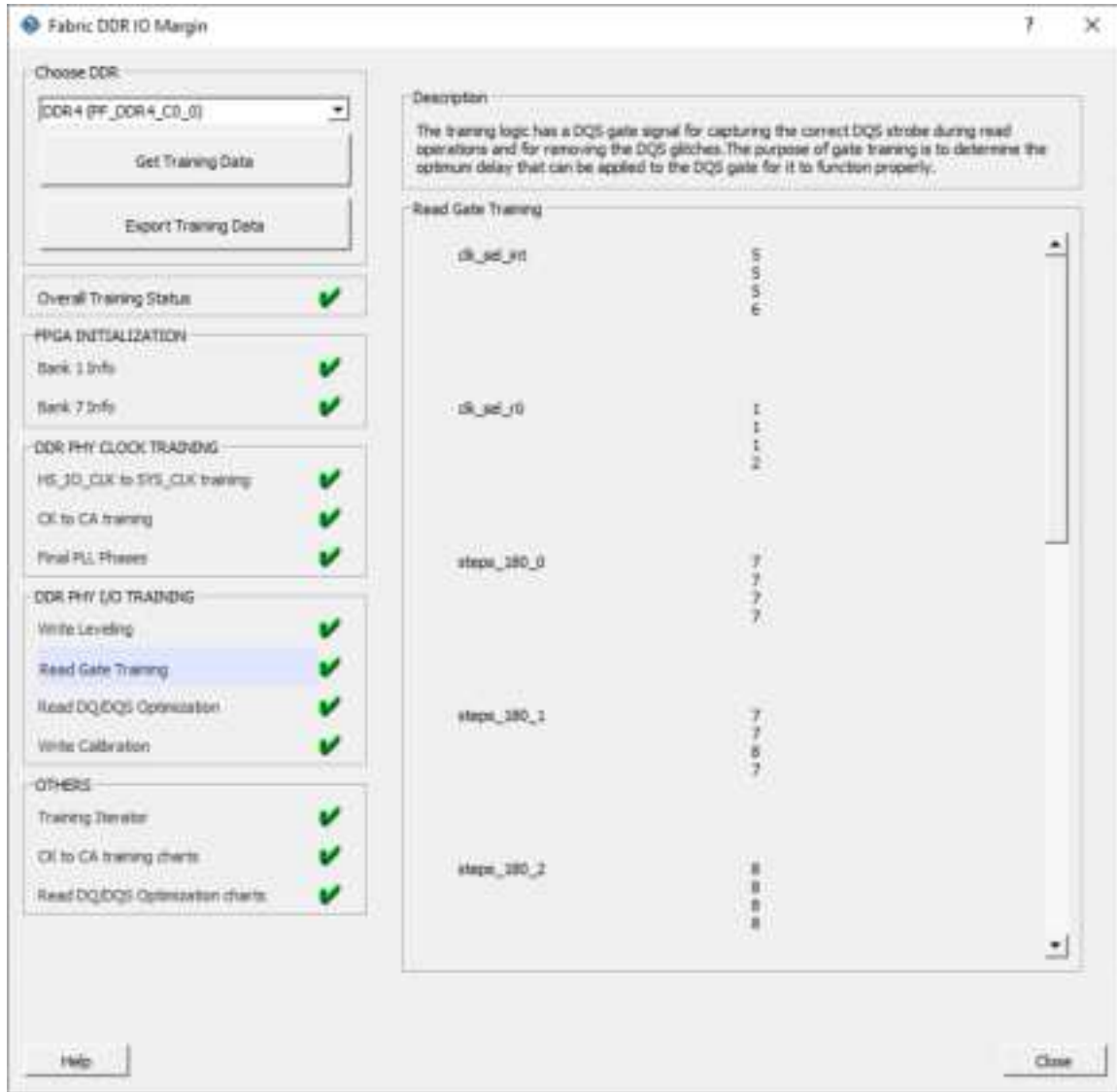


Figure 3-100. Fabric DDR IO Margin Dialog Box Showing Read DQ/DQS Optimization Data

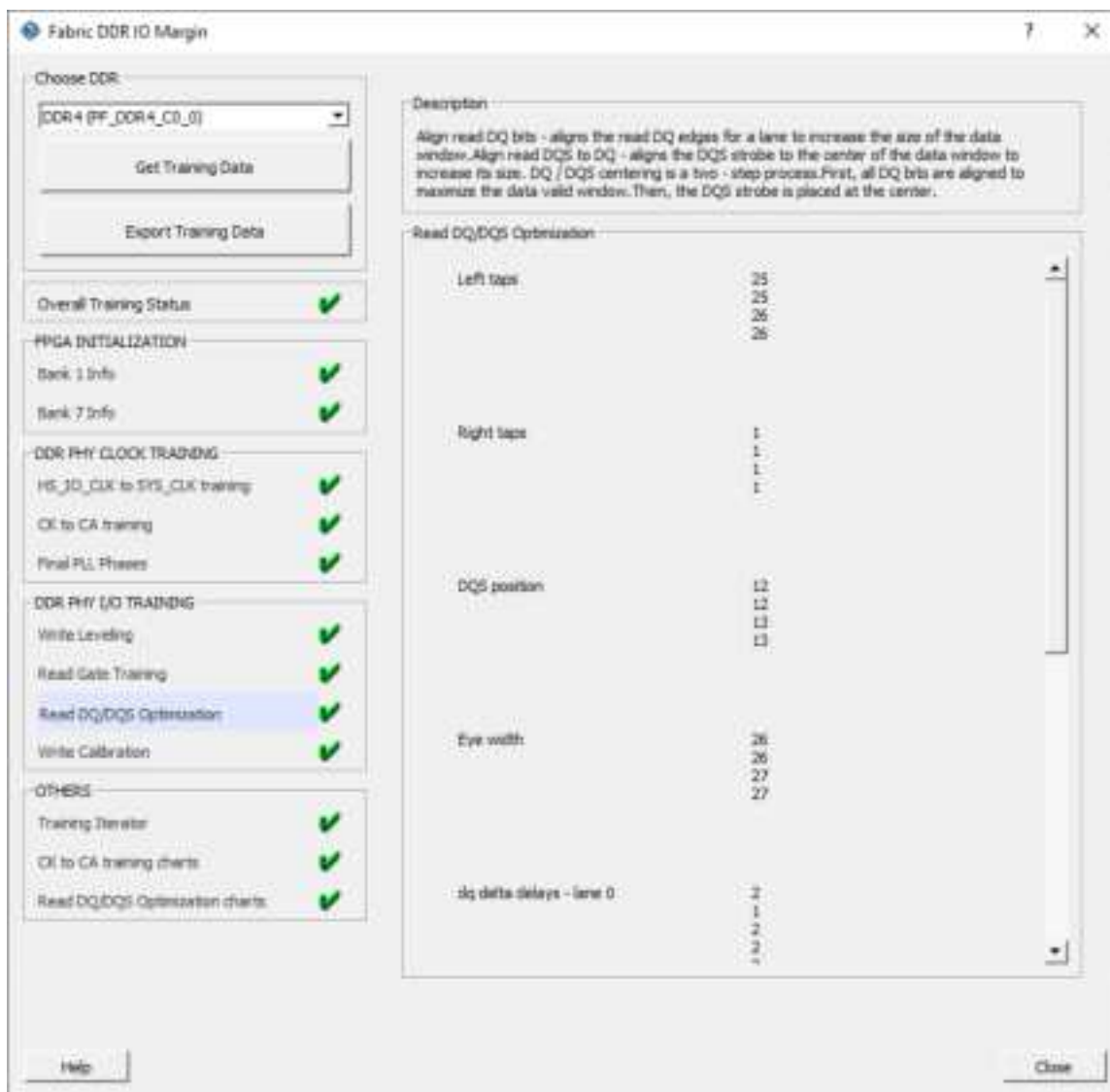


Figure 3-101. Fabric DDR IO Margin Dialog Box Showing Write Calibration Data

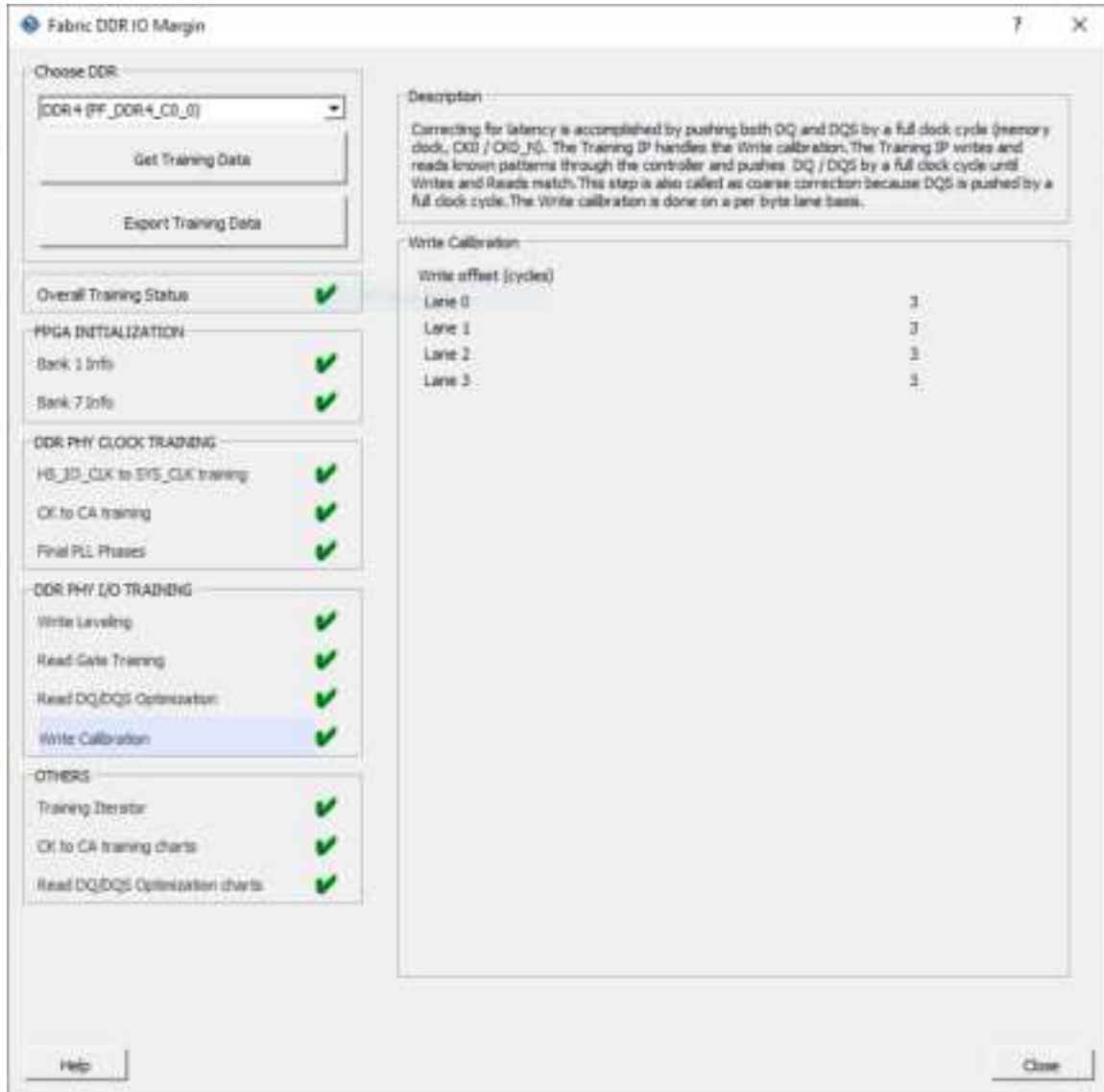


Figure 3-102. Fabric DDR IO Margin Dialog Box Showing Training Iterator Data

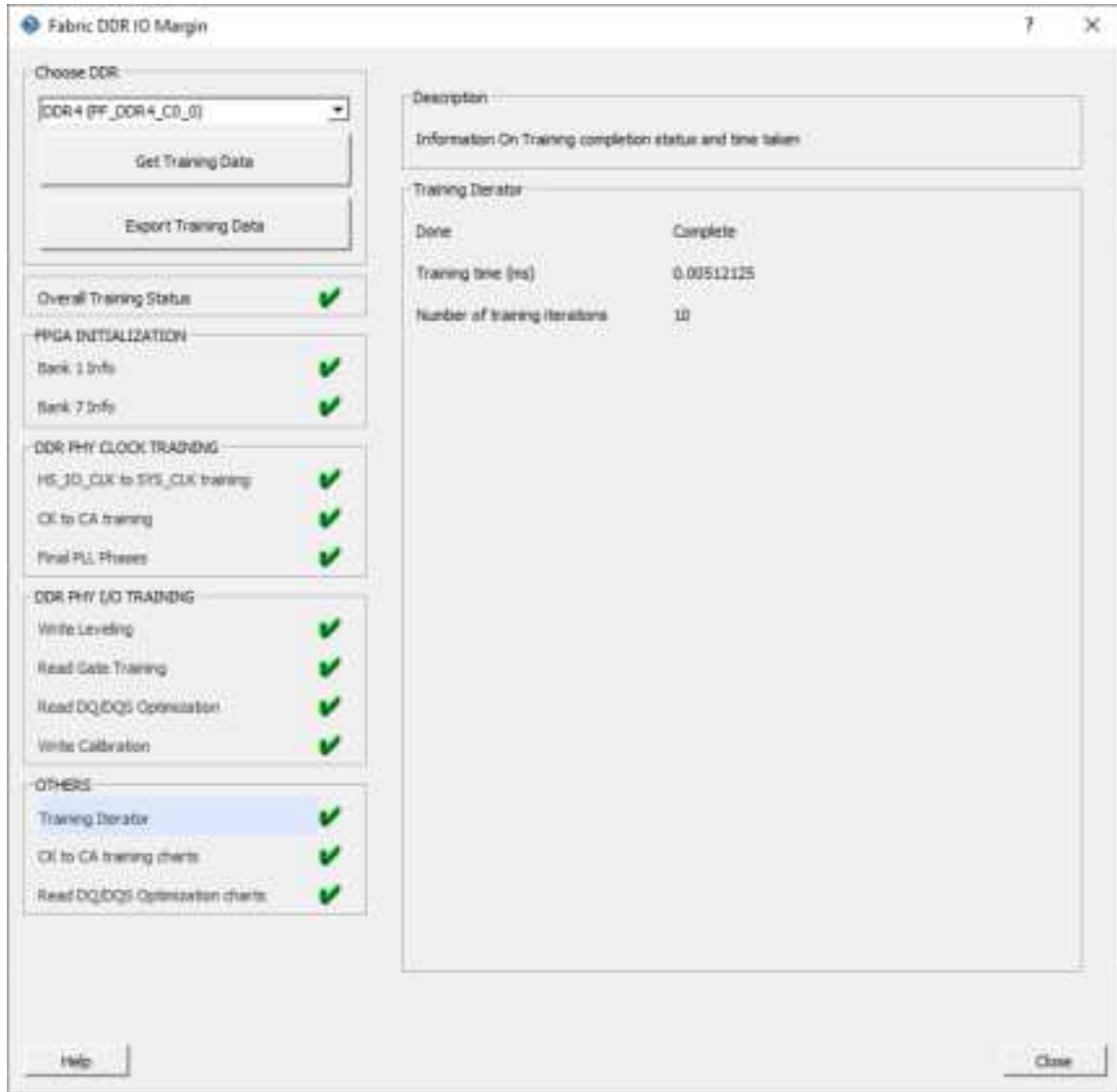


Figure 3-103. Fabric DDR IO Margin Dialog Box showing CK-to-CA Training Charts

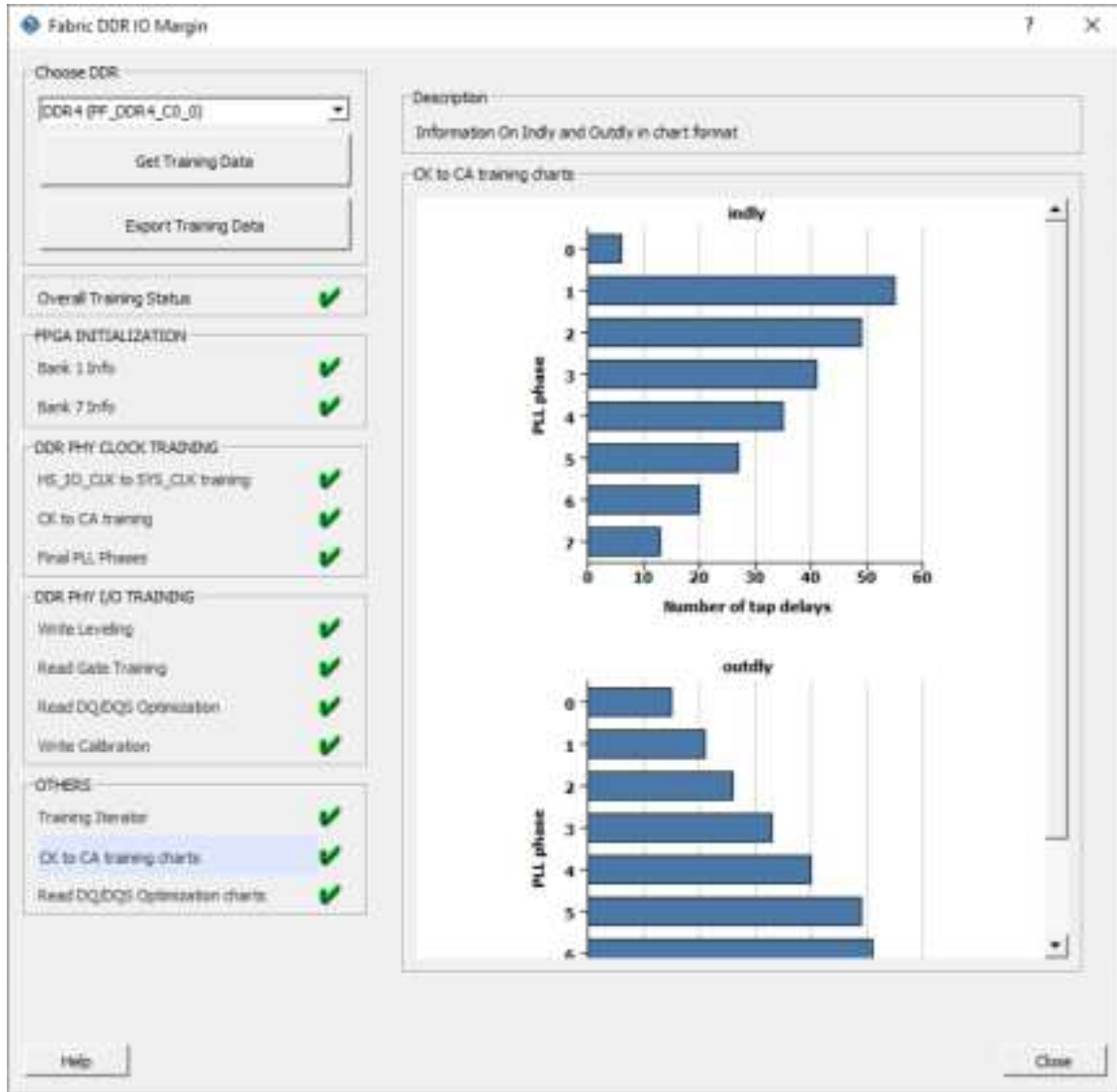
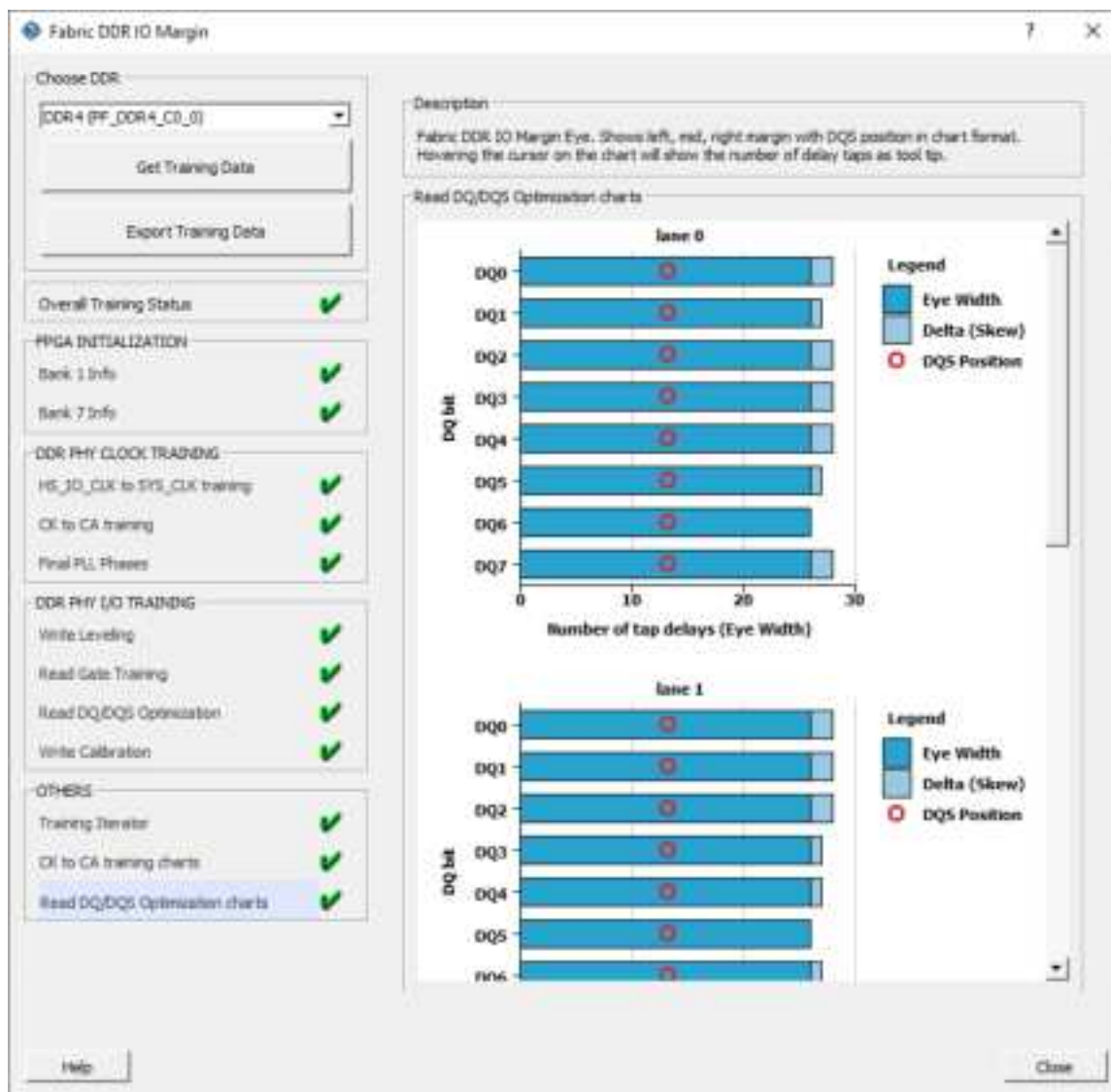


Figure 3-104. Fabric DDR IO Margin Dialog Box Showing Read DQ/DQS Optimization Charts

SmartDebug informs users about errors generated while getting training data, as shown in the following figures and tables. Additional error information can be displayed as tooltips by hovering over the elements in the **FPGA INITIALIZATION**, **DDR PHY CLOCK TRAINING**, **DDR PHY I/O TRAINING**, and **OTHERS** categories. The following table outlines the stages, reasons for failure, and suggested actions to address and eliminate these failures.

Table 3-5. DDR Stages, Reasons for Failure, and Recommended Actions

Stages	Reason(s) for Failure	Suggested Actions to Remove Failure
FPGA Initialization	Memory controller is held in reset.	Make sure the controller is not held in reset, check SYS_RESET_N.
	PLL_REF_CLK is undriven.	Make sure the controller input clock, PLL_REF_CLK, is toggling.
HS_IO_CLK-to-SYS_CLK Training	PLL_REF_CLK is undriven.	<ul style="list-style-type: none"> Make sure the controller input clock, PLL_REF_CLK, is toggling. Check PLL_LOCK and SYS_CLK for valid signals.

Table 3-5. DDR Stages, Reasons for Failure, and Recommended Actions (continued)

Stages	Reason(s) for Failure	Suggested Actions to Remove Failure
CK-to-CA Training	<ul style="list-style-type: none"> Signal-integrity issues occur on CA/CK due to mismatched drive and termination settings. CK/CA default offset is not optimal for the system. 	<p>Memory chip - mode registers. Configure the output drive strength and/or the ODT values using the Memory Initialization tab in the DDRx configurator.</p> <p>FPGA - IO attribute editor. Configure the output drive strength and/or the ODT values using the IO attribute editor.</p>
Write Leveling	Signal-integrity issues occur on DQ/DQS.	<p>Memory chip - mode registers. Configure the output drive strength and/or the ODT values using the Memory Initialization tab in the DDRx configurator.</p> <p>FPGA - IO attribute editor. Configure the output drive strength and/or the ODT values using the IO attribute editor.</p>
Read DQ/DQS Optimization	Signal-integrity issues occur on DQ/DQS.	<p>Memory chip - mode registers. Configure the output drive strength and/or the ODT values using the Memory Initialization tab in the DDRx configurator.</p> <p>FPGA - IO attribute editor. Configure the output drive strength and/or the ODT values using the IO attribute editor.</p>
Write Calibration	Signal-integrity issues occur on DQ/DQS.	<p>Memory chip - mode registers. Configure the output drive strength and/or the ODT values using the Memory Initialization tab in the DDRx configurator.</p> <p>FPGA - IO attribute editor. Configure the output drive strength and/or the ODT values using the IO attribute editor.</p>
Training Iterator	Training was not successful after 10 attempts.	See the above actions for specific failure reasons.

Figure 3-105. Example of Tooltip



Figure 3-106. Fabric DDR IO Margin Dialog Box Showing Error Generated when DDR PLL is not Locked

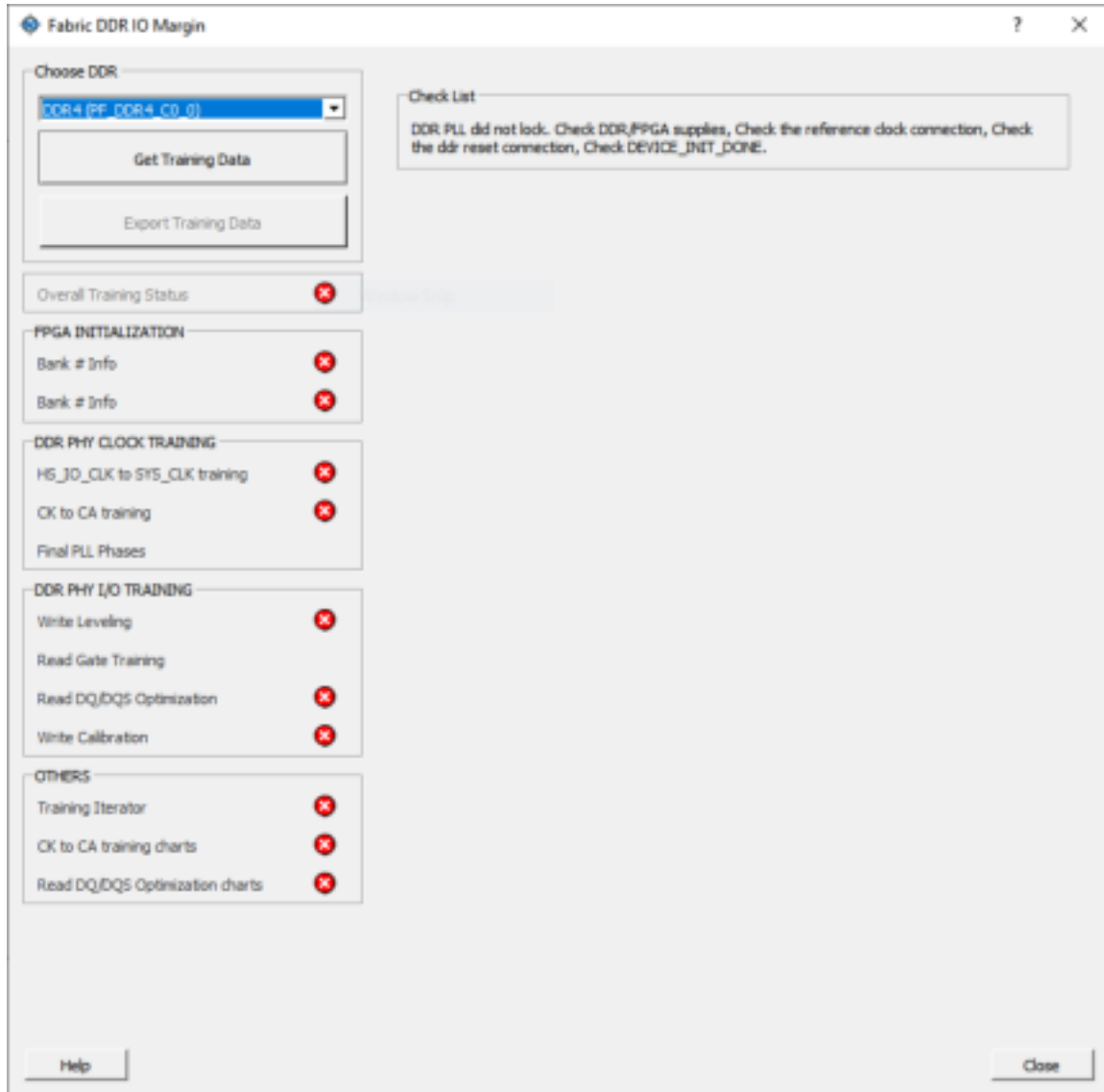


Figure 3-107. Fabric DDR IO Margin Dialog Box Showing Error Generated During FPGA Initialization

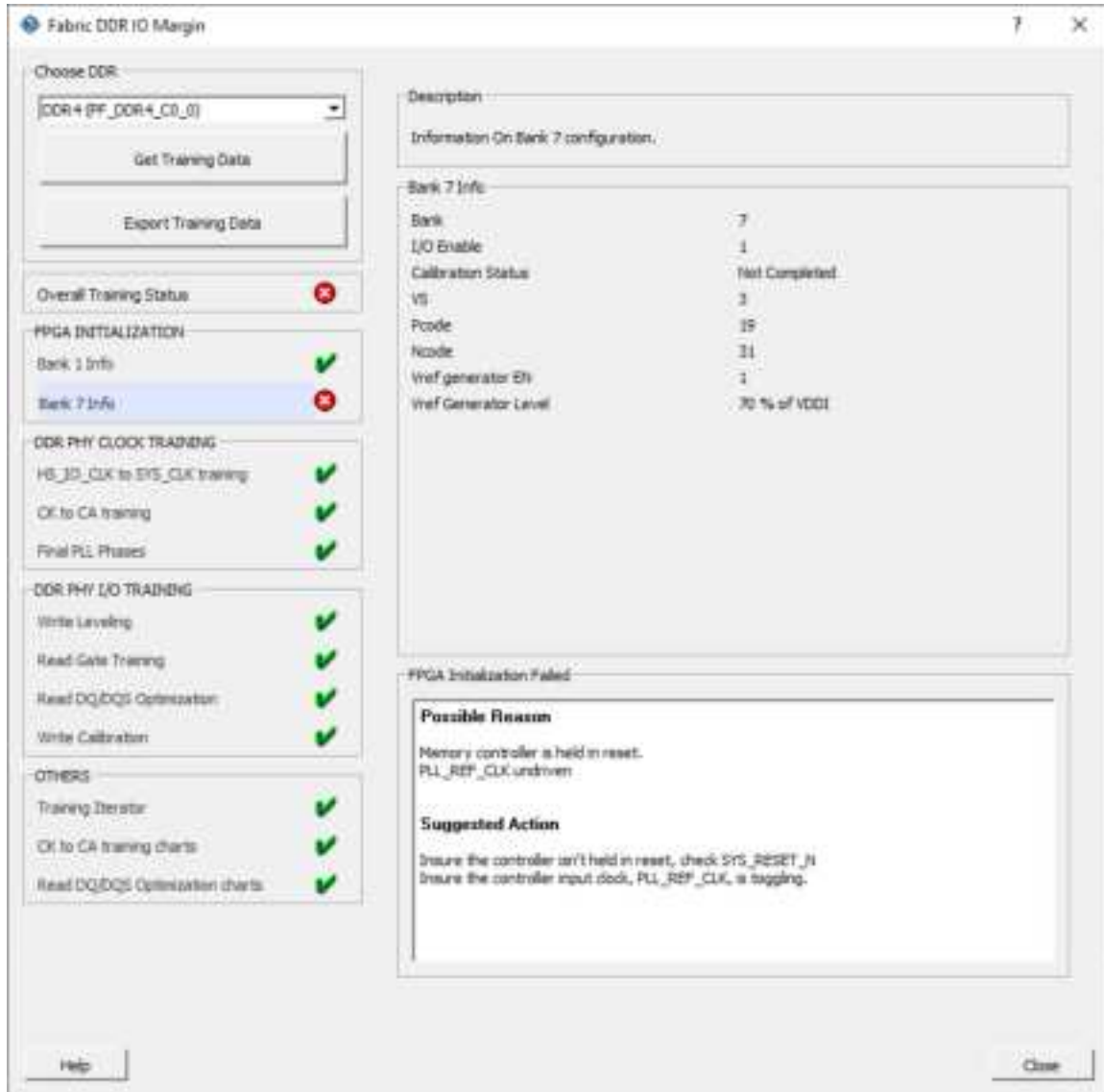


Figure 3-108. Fabric DDR IO Margin Dialog Box Showing Error Generated During HS_IO_CLK-to-SYS_CLK Training

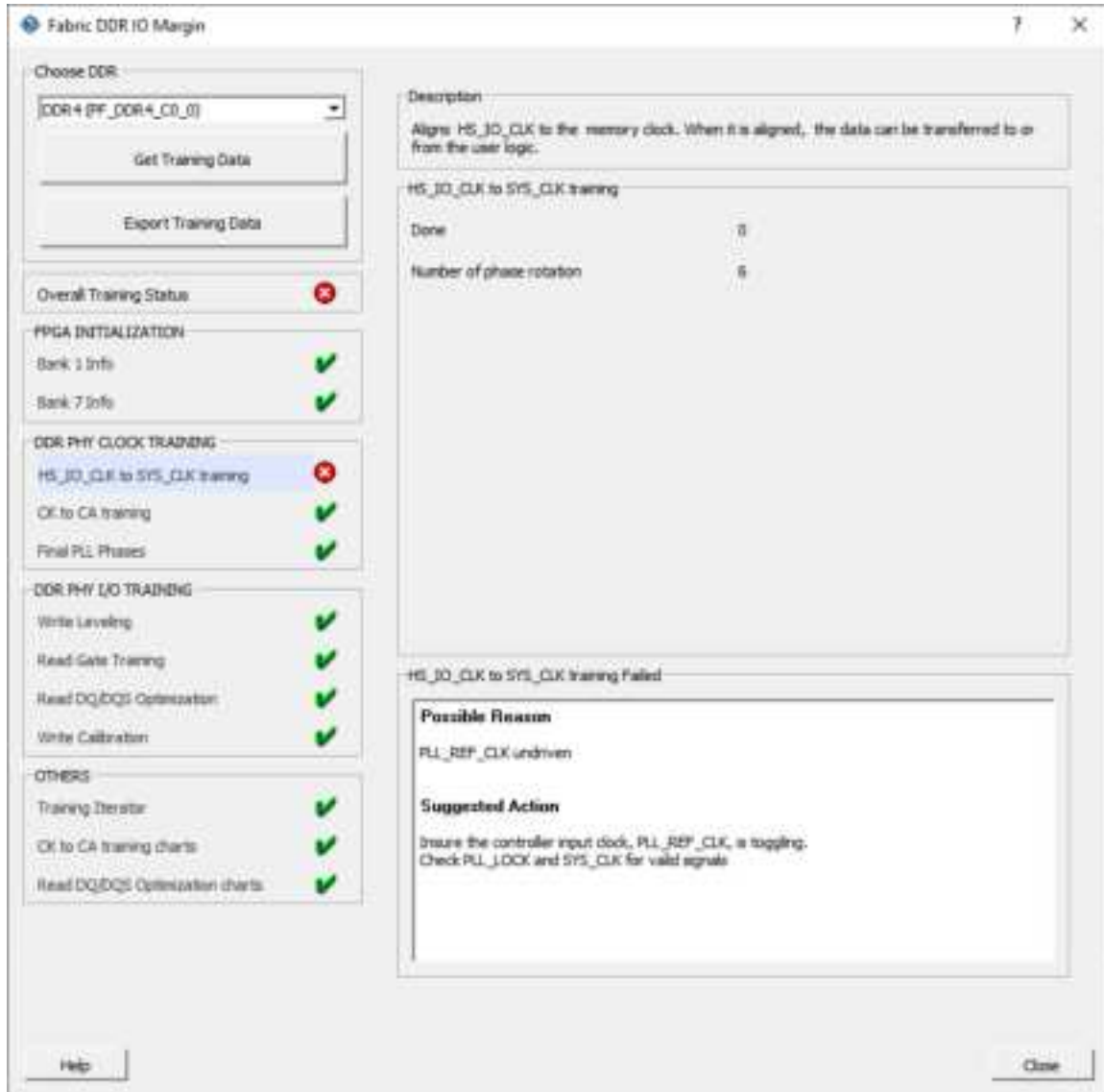


Figure 3-109. Fabric DDR IO Margin Dialog Box Showing Error Generated During CK-to-CA Training



Figure 3-110. Fabric DDR IO Margin Dialog Box Showing Error Generated During Write Leveling Data



Figure 3-111. Fabric DDR IO Margin Dialog Box Showing Error Generated During Read DQ/DQS Optimization



Figure 3-112. Fabric DDR IO Margin Dialog Box Showing Error Generated During Write Calibration

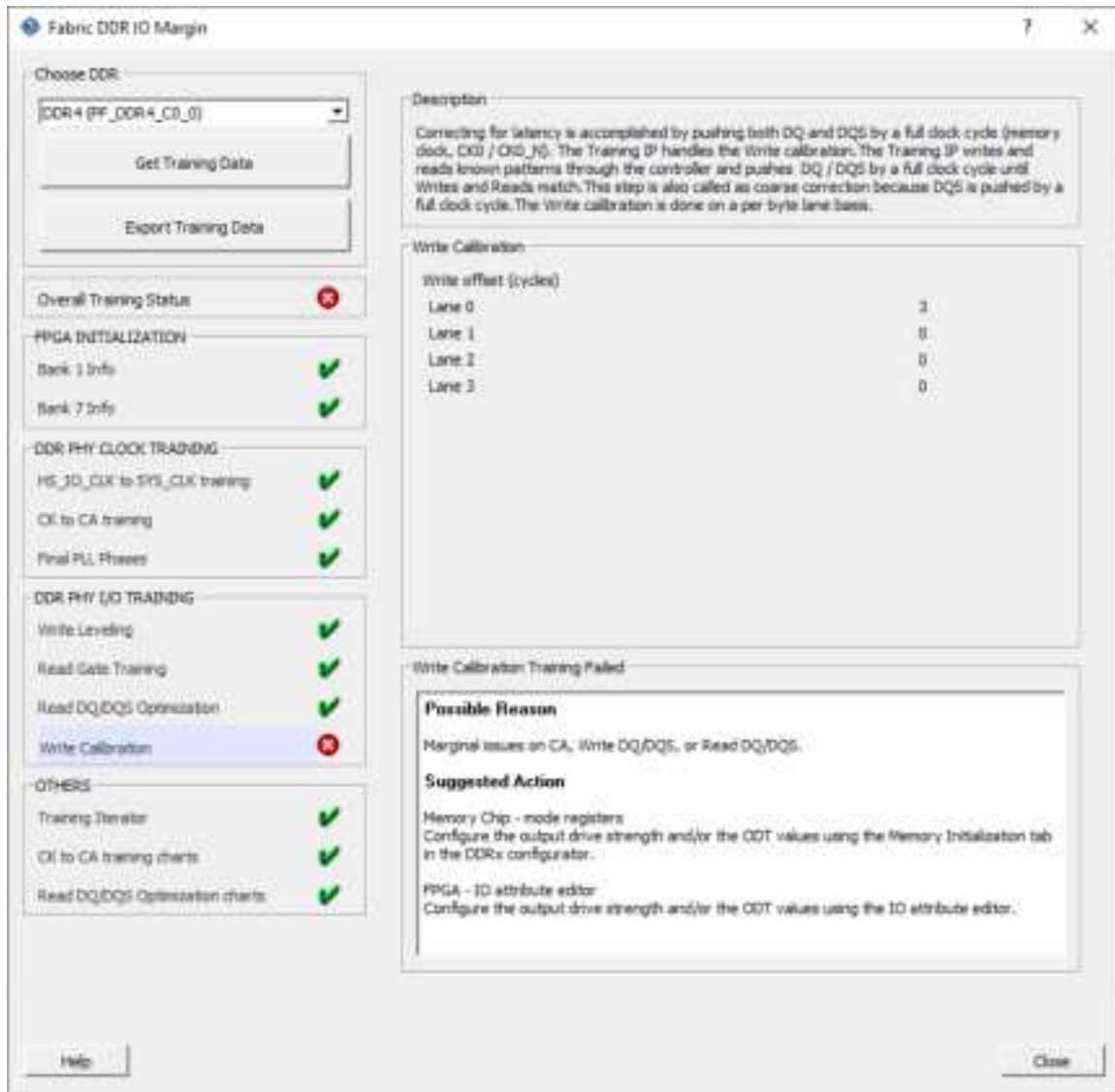
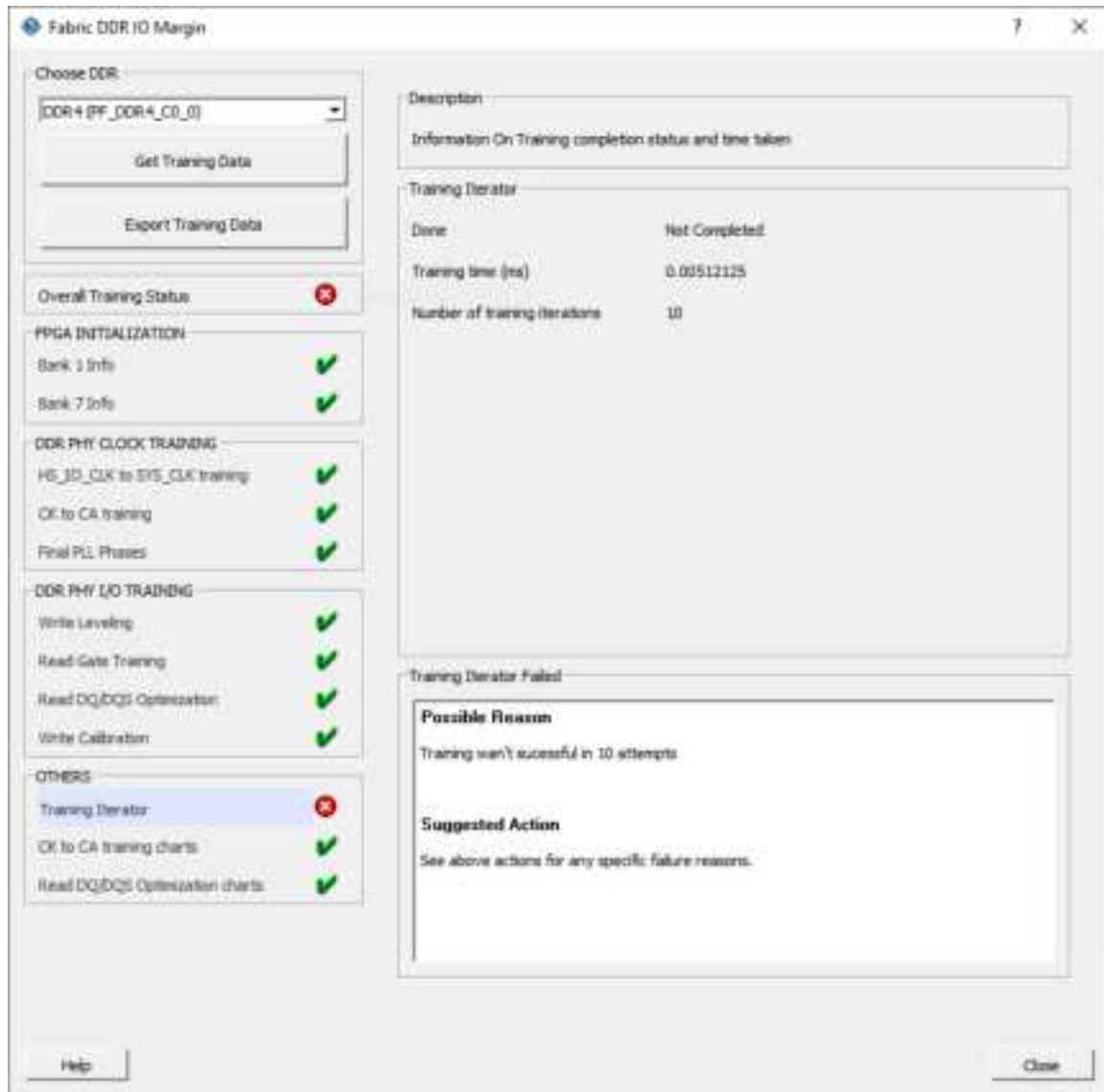


Figure 3-113. Fabric DDR IO Margin Dialog Box Showing Error Generated During Training Iterator



3.2.7. Debug MSS DDR IO Margin [\(Ask a Question\)](#)

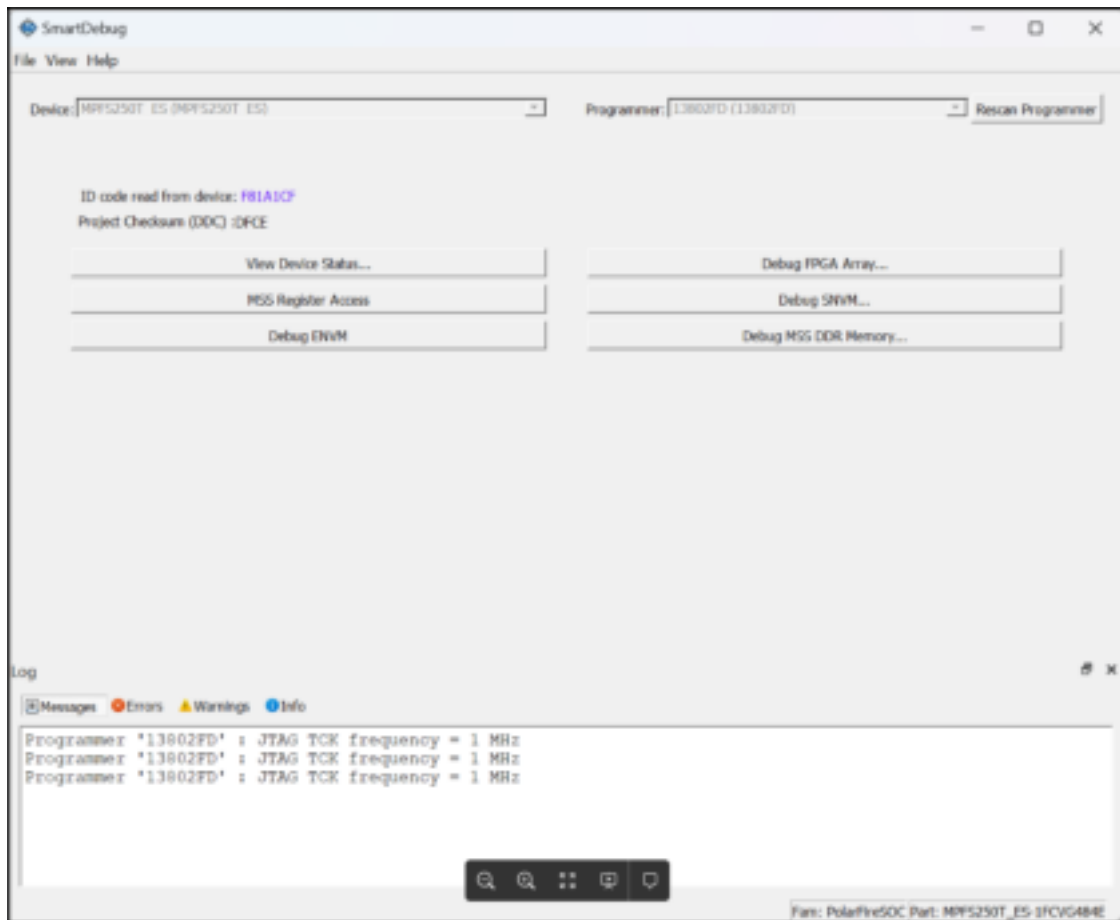
The MSS DDR-PHY is an integral part of PolarFire SOC memory subsystem that runs the initialization and training sequence. The training steps include:

- Clock alignment
- Command/Address training
- Memory initialization
- Write leveling
- Read gate leveling
- DQ/DQS optimization
- Write calibration

SmartDebug shows the training sequence results and supplies a probable reason for failure and action required to ensure that the training passes.

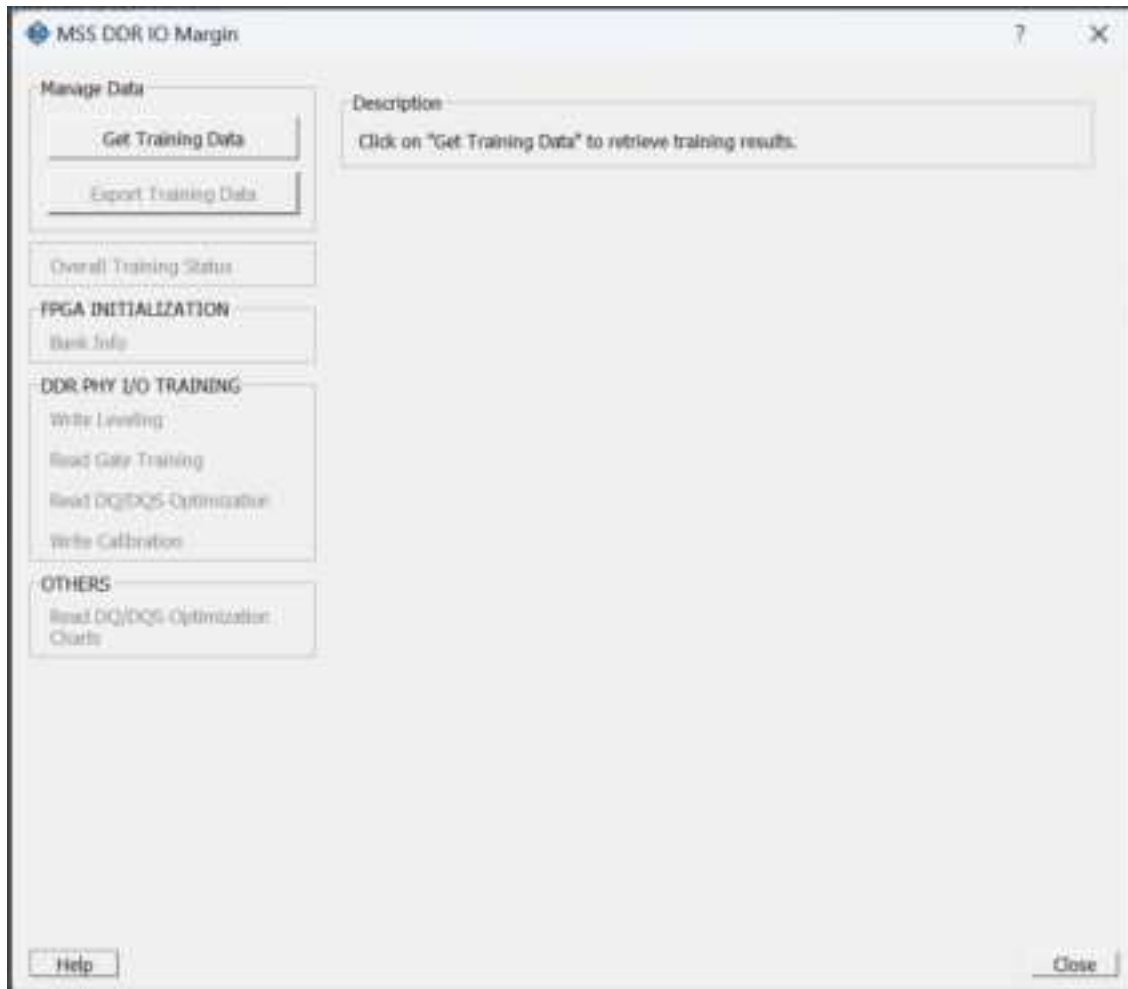
To access the Debug MSS DDR IO Margin feature in SmartDebug, click **Debug MSS DDR Memory...** in the main SmartDebug window. This option is not shown when MSS DDR memory is not used in the design.

Figure 3-114. SmartDebug Main Window with Debug MSS DDR Memory Option



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

Figure 3-115. MSS DDR IO Margin Dialog Box with Options Disabled



Initially, all options in the **Debug MSS DDR IO Margin** dialog box are disabled. To access the training data, click the **Get Training Data** button. Clicking this button prompts SmartDebug to read the device registers and retrieve the training data. During this time, the **MSS DDR IO Margin** dialog box shows information similar to that shown in the following figures. First stage, **Overall Training Status** is marked as **Passed** if all subsequent stages are completed successfully. If any stage fails, the status is marked as **Failed**.

Figure 3-116. MSS DDR IO Margin Dialog Box Showing Overall Training Status

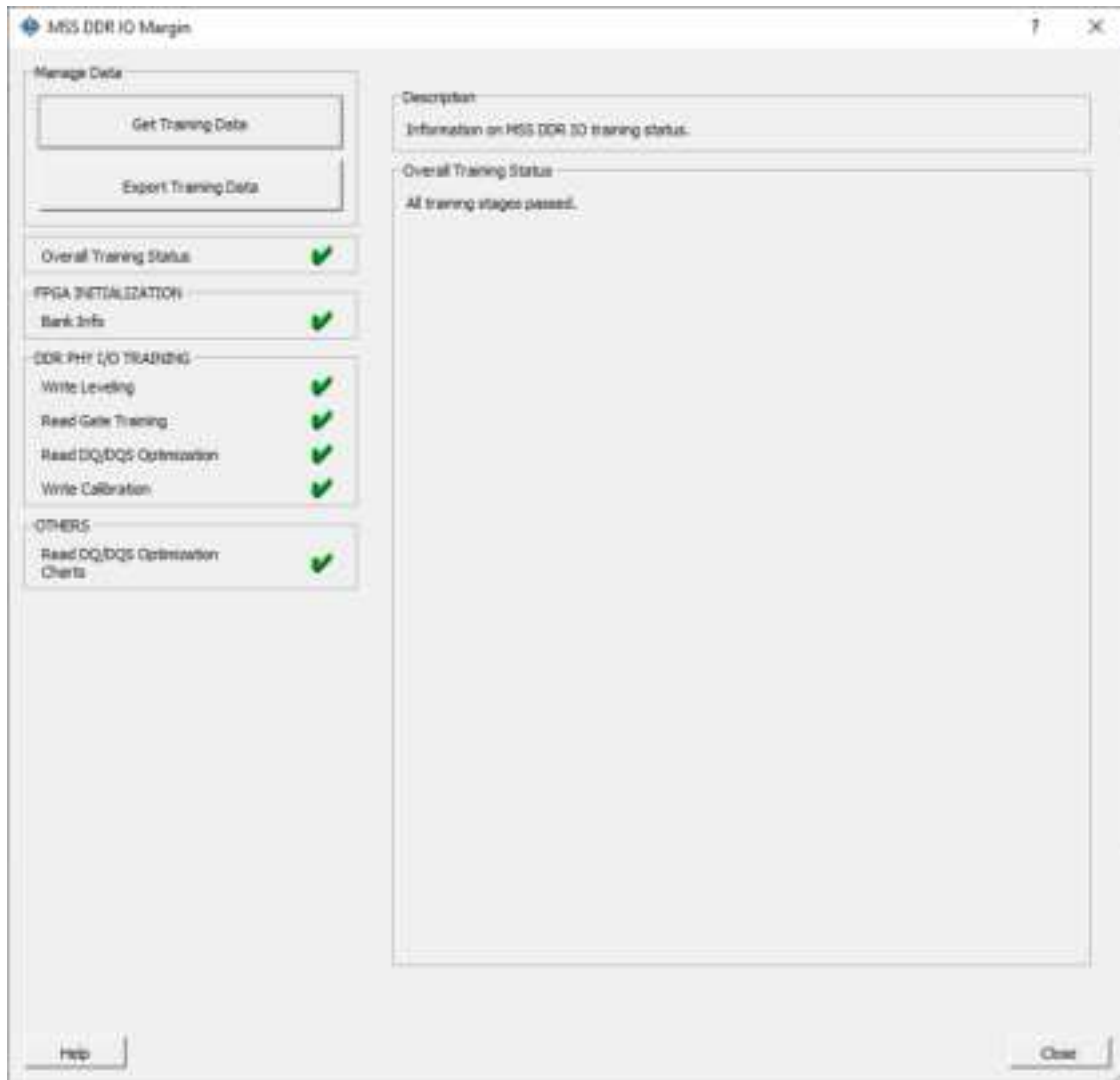


Figure 3-117. MSS DDR IO Margin Dialog Box Showing Bank Info

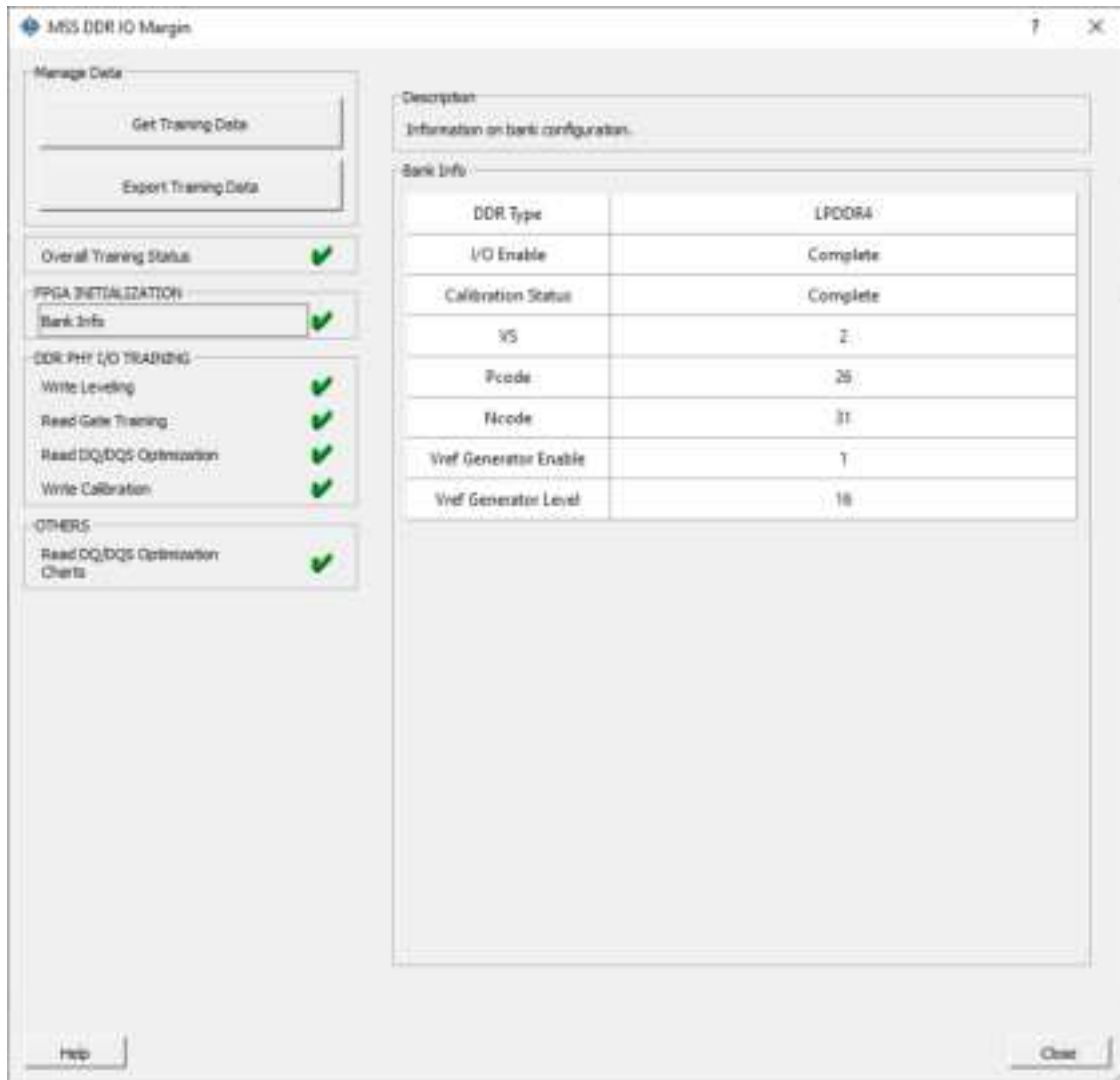


Figure 3-118. MSS DDR IO Margin Dialog Box Showing Write Leveling Data

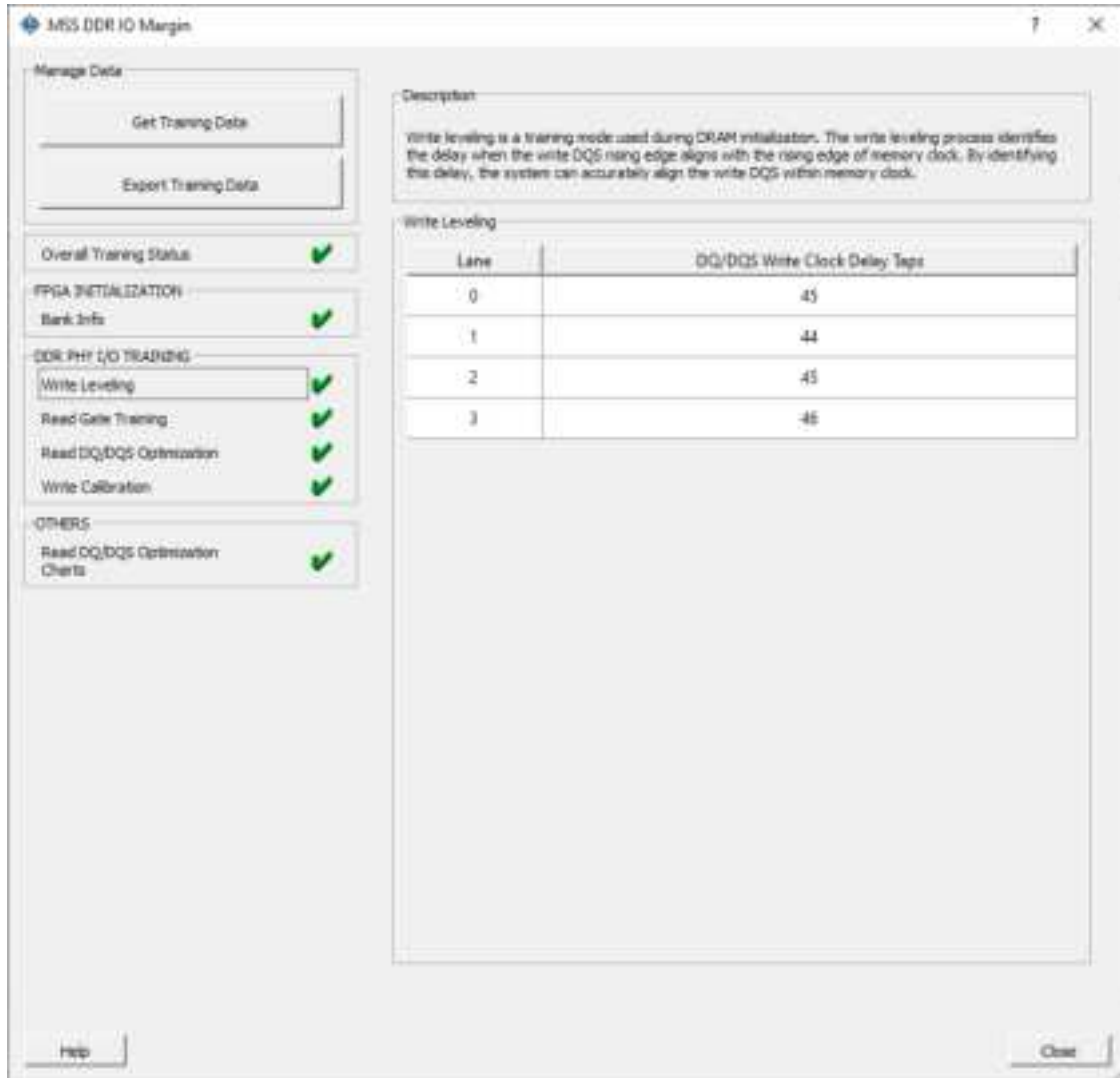


Figure 3-119. MSS DDR IO Margin Dialog Box Showing Read Gate Training Data



Figure 3-120. MSS DDR IO Margin Dialog Box Showing DQ/DQS Optimization Data

MSS DDR IO Margin

Manage Data

Get Training Data

Export Training Data

Overall Training Status ✓

FPGA INITIALIZATION

Bank Info ✓

DDR PHY I/O TRAINING

Write Leveling ✓

Read Gain Training ✓

Read DQ/DQS Optimization ✓

Write Calibration ✓

OTHERS

Read DQ/DQS Optimization Charts ✓

Description

Align read DQ bits - aligns the read DQ edges for a lane to increase the size of the data window. Align read DQS to DQ - aligns the DQS strobe to the center of the data window to increase its size. DQ / DQS centering is a two-step process. First, all DQ bits are aligned to recenter the data valid window. Then, the DQS strobe is placed at the center.

Read DQ/DQS Optimization

Lane	Left Taps	Right Taps	DQS Position	Eye Width
0	4	10	7	23
1	5	10	6	23
2	0	17	9	17
3	5	10	6	24

DQ Delta Delay

	Lane 0	Lane 1	Lane 2	Lane 3
DQ Bit 0	0	1	1	1
DQ Bit 1	1	0	1	2
DQ Bit 2	0	2	0	0
DQ Bit 3	1	0	2	1
DQ Bit 4	1	3	0	1
DQ Bit 5	2	2	1	2
DQ Bit 6	1	1	1	1
DQ Bit 7	2	0	1	0

help Close

Figure 3-121. MSS DDR IO Margin Dialog Box Showing Write Calibration Data



SmartDebug informs you about errors generated while getting training data, as shown in the following figures. Additional error details can be viewed as tooltips by hovering over the cross icons and in the **Group Box** at the bottom of the page. These details include potential reasons for the failure and suggested Actions in **FPGA INITIALIZATION**, **PHY I/O TRAINING**, and **OTHERS** categories, as shown in the following example.

Figure 3-122. MSS DDR IO Margin Dialog Box Showing DQ/DQS Optimization Charts



Figure 3-123. MSS DDR IO Margin Dialog Box Shows Error during Bank Info and ToolTip

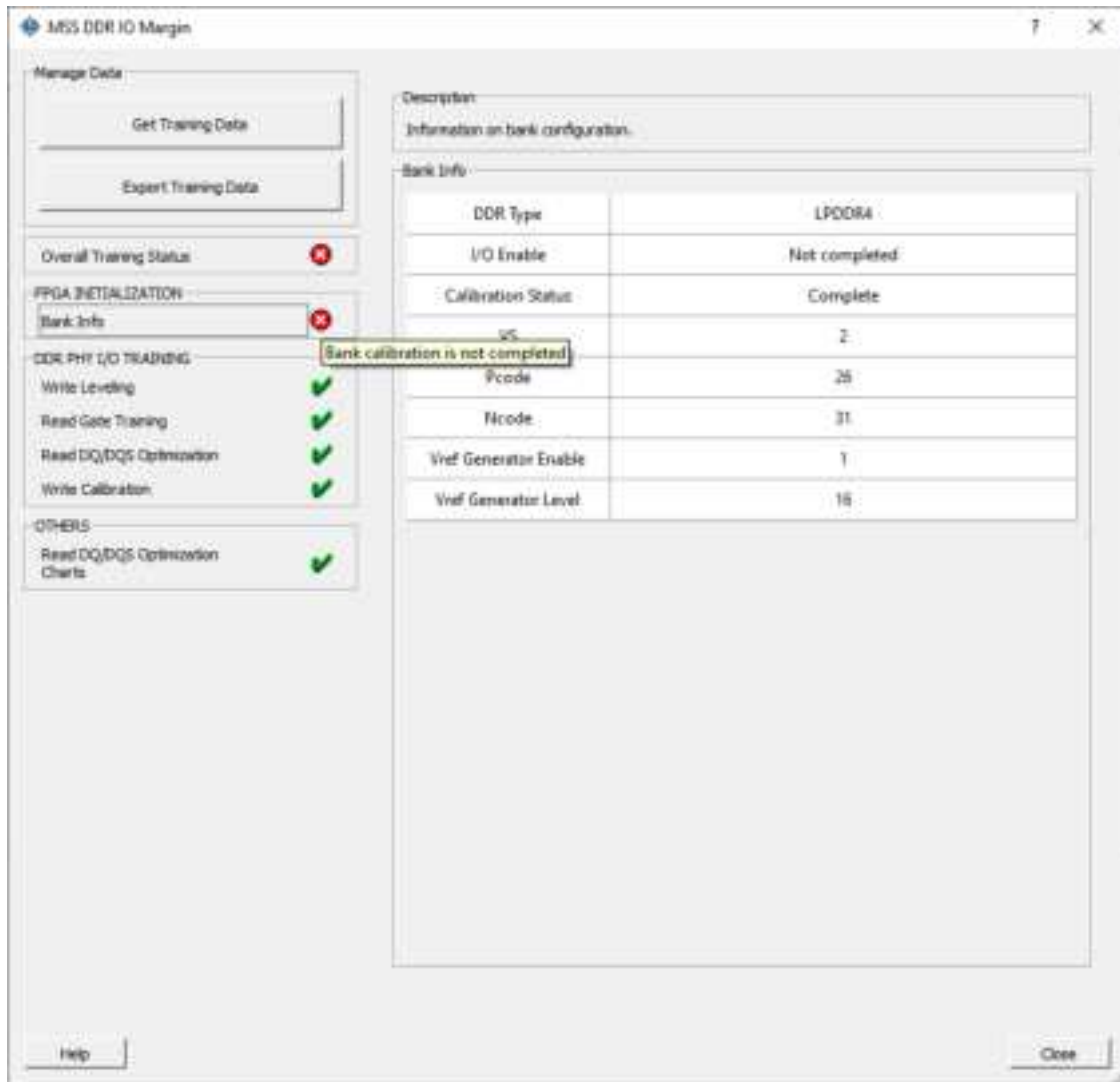


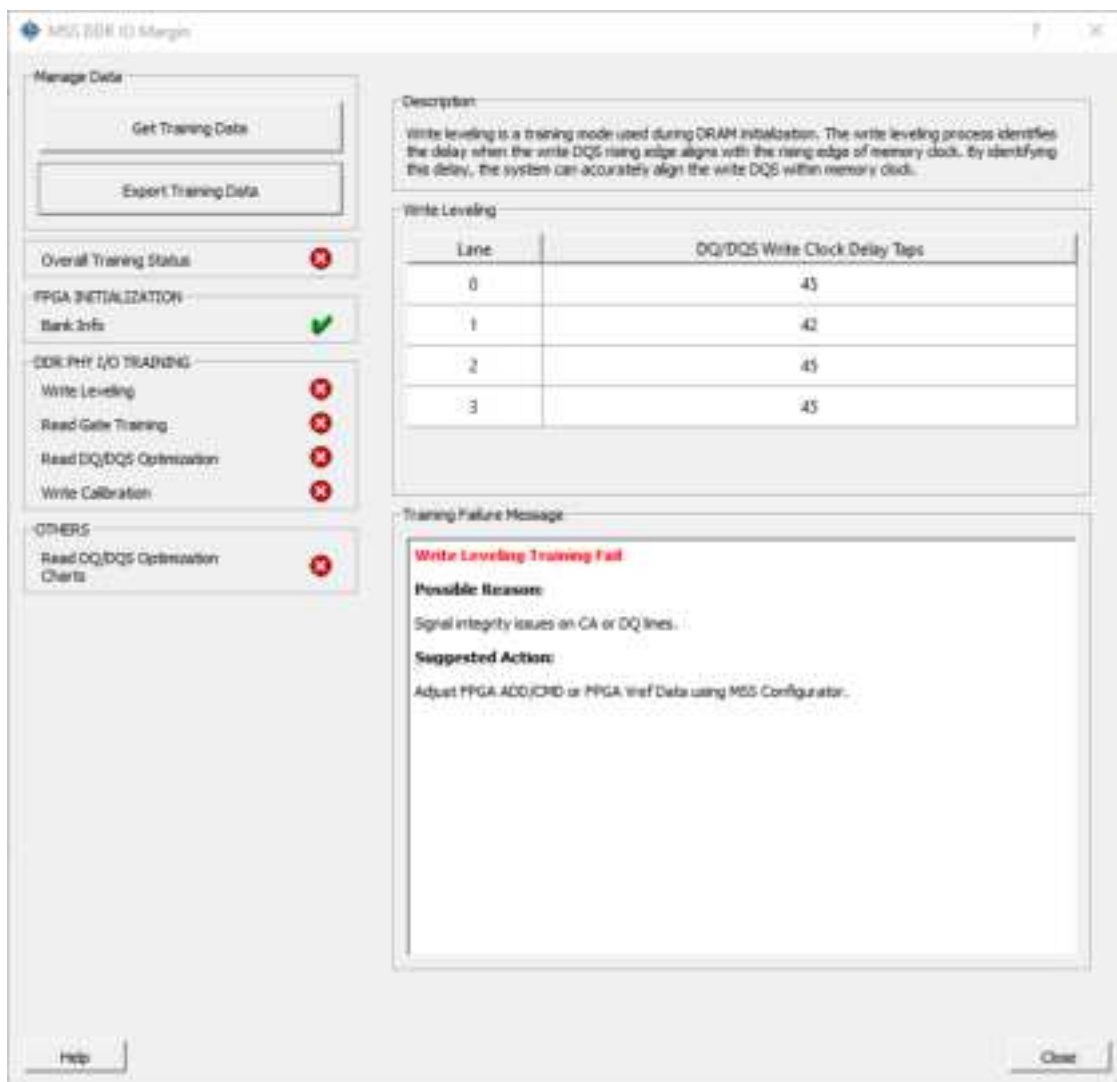
Figure 3-124. MSS DDR IO Margin Dialog Box Showing Training Failure Message during Write Leveling

Figure 3-125. MSS DDR IO Margin Dialog Box Showing Training Failure Message during Read Gate Training Data

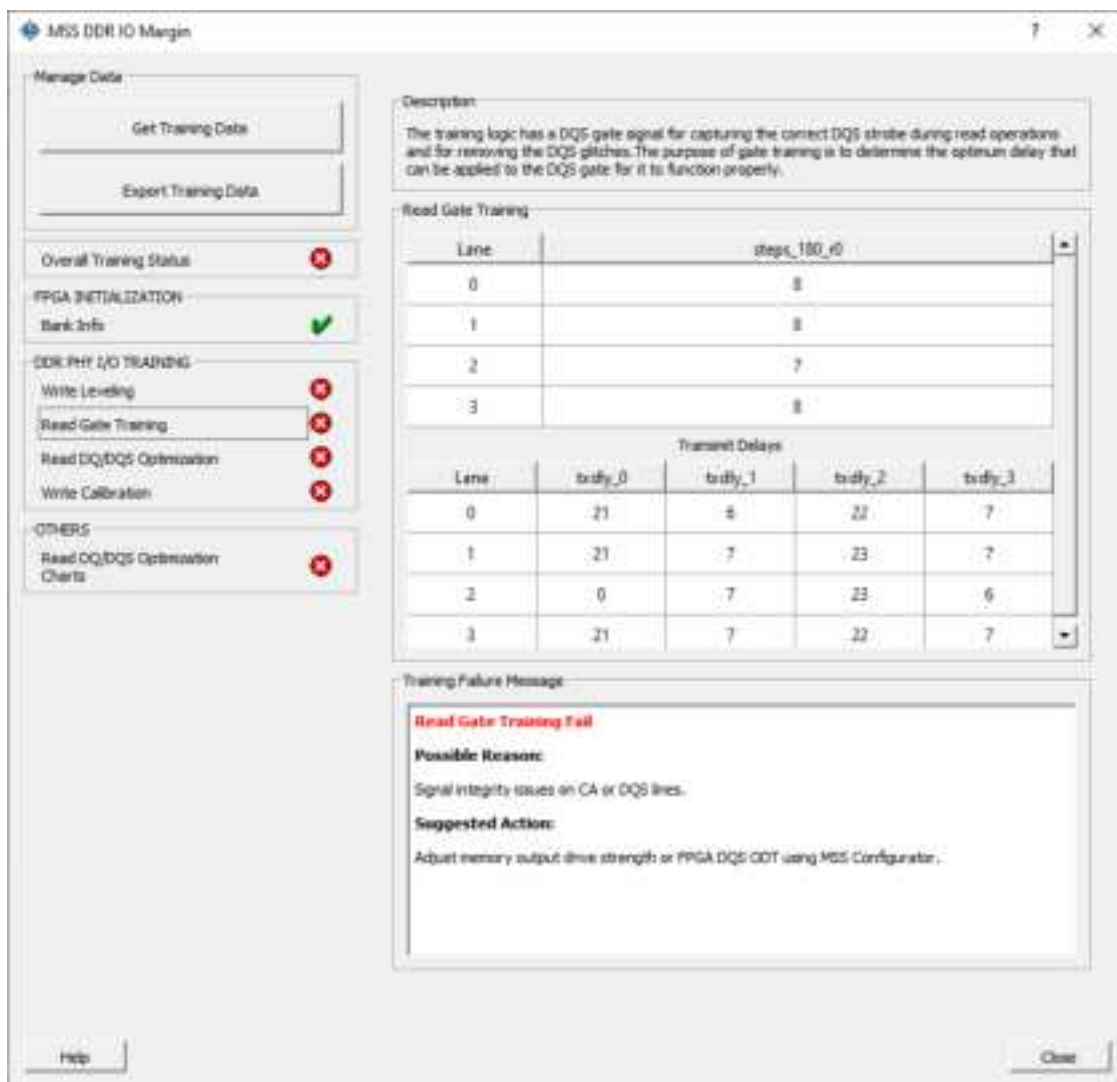
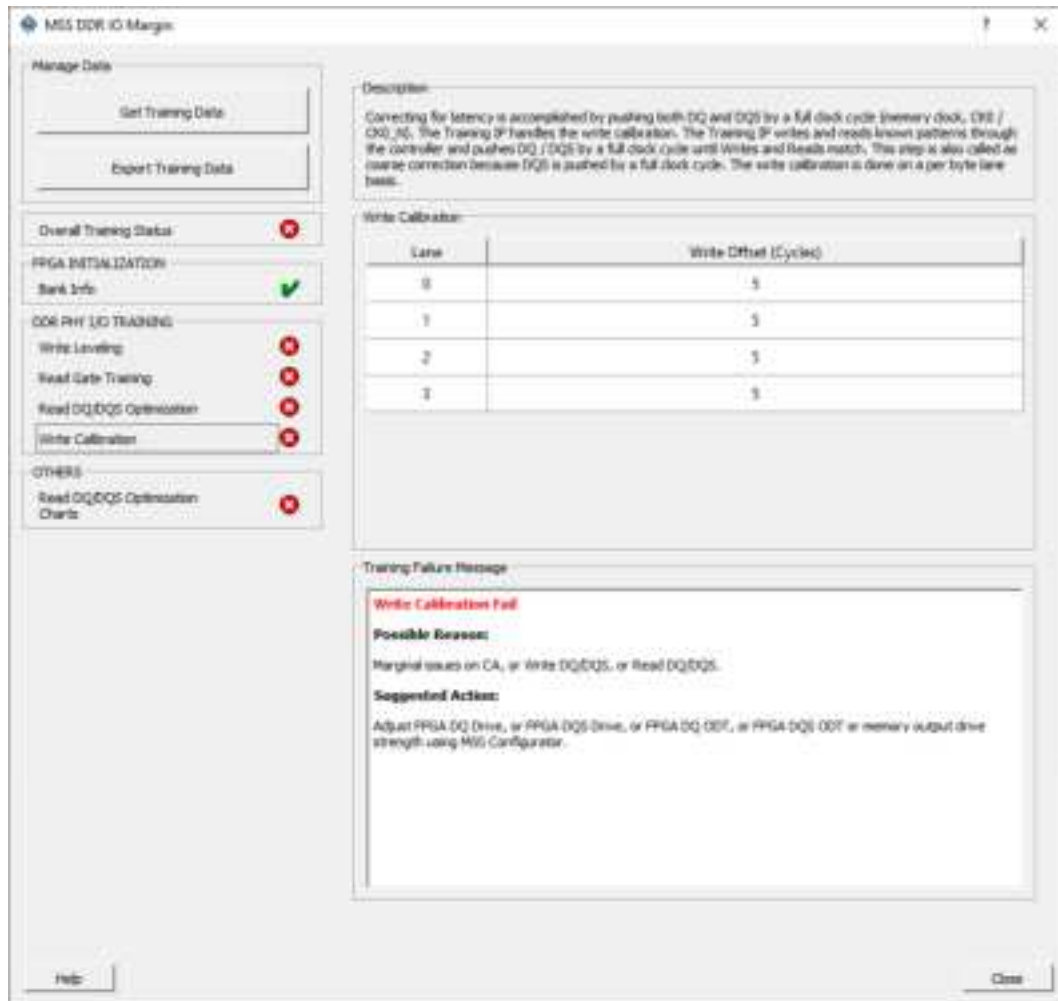


Figure 3-126. MSS DDR IO Margin Dialog Box Showing Training Failure Message during Write Calibration

3.2.8. Debug eNVM (PolarFire SoC) [\(Ask a Question\)](#)

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.

➔ Important: The contents of eNVM cannot be debugged in boot mode 0. Therefore, before using SmartDebug, Debug eNVM requires a device to be in an active boot mode (that is, a boot mode other than 0) and have a valid embedded software application running that enables the MPU.

Table 3-6. 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](#).

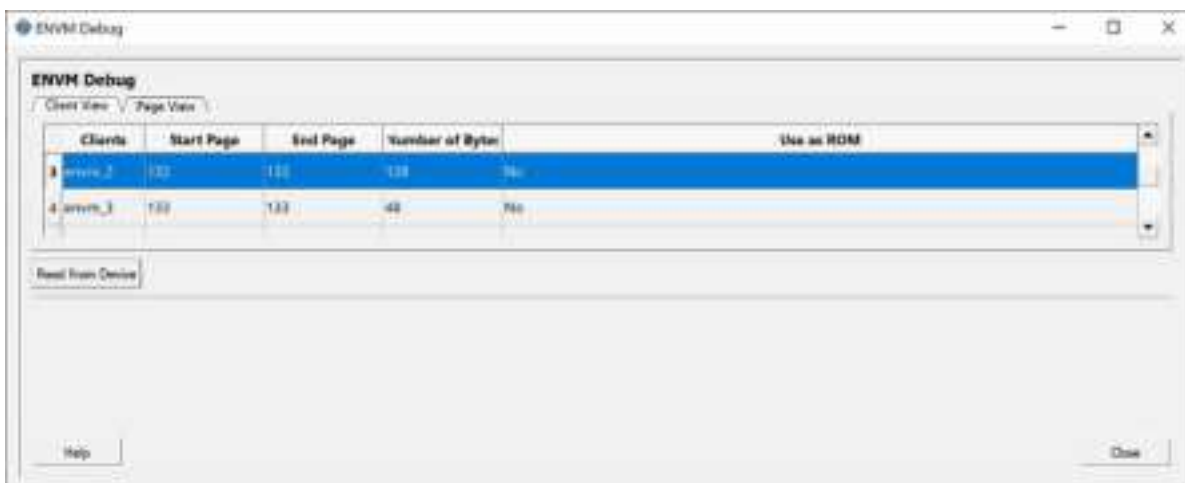
3.2.8.1. Client View [\(Ask a Question\)](#)

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. The **ENVM Debug** window shows the client name, client start page number, client end page number, number of bytes used and whether the client is used as a ROM (see the following figure).

Figure 3-127. ENVM Debug Window with Client View

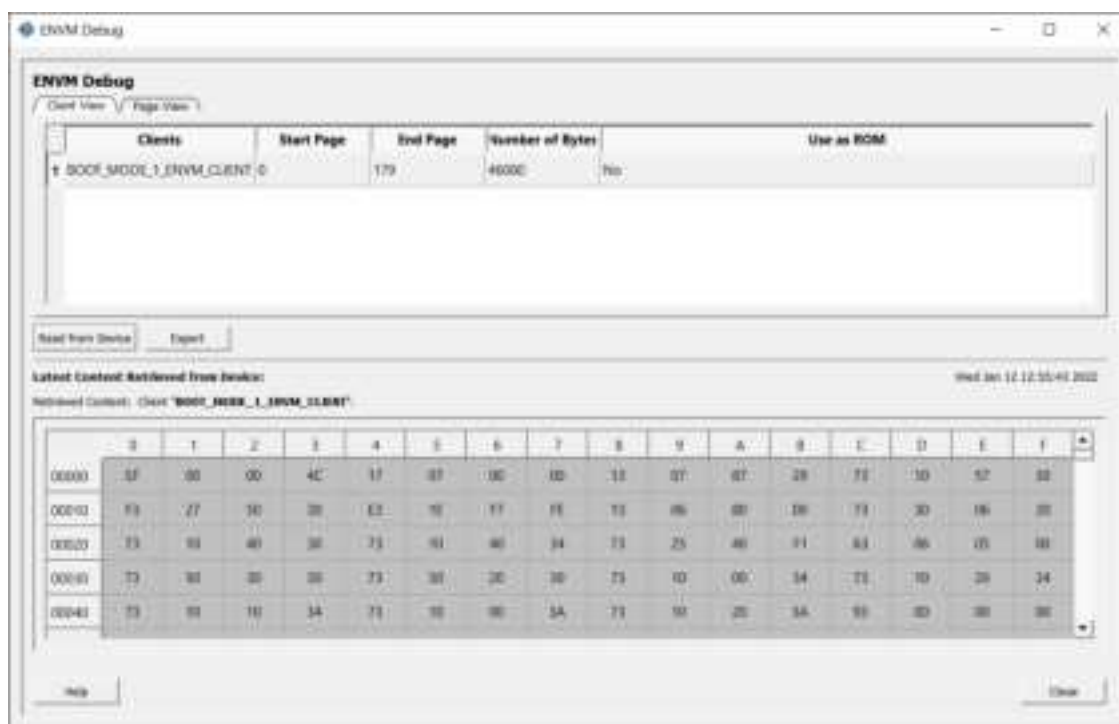
A single client can be selected at a time to view the content. Once a client is selected, the **Read from Device** button is enabled.

Figure 3-128. 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. Click the **Export** button to export the eNVM data to a text file.

Figure 3-129. Client View Data



3.2.8.2. Page View [\(Ask a Question\)](#)

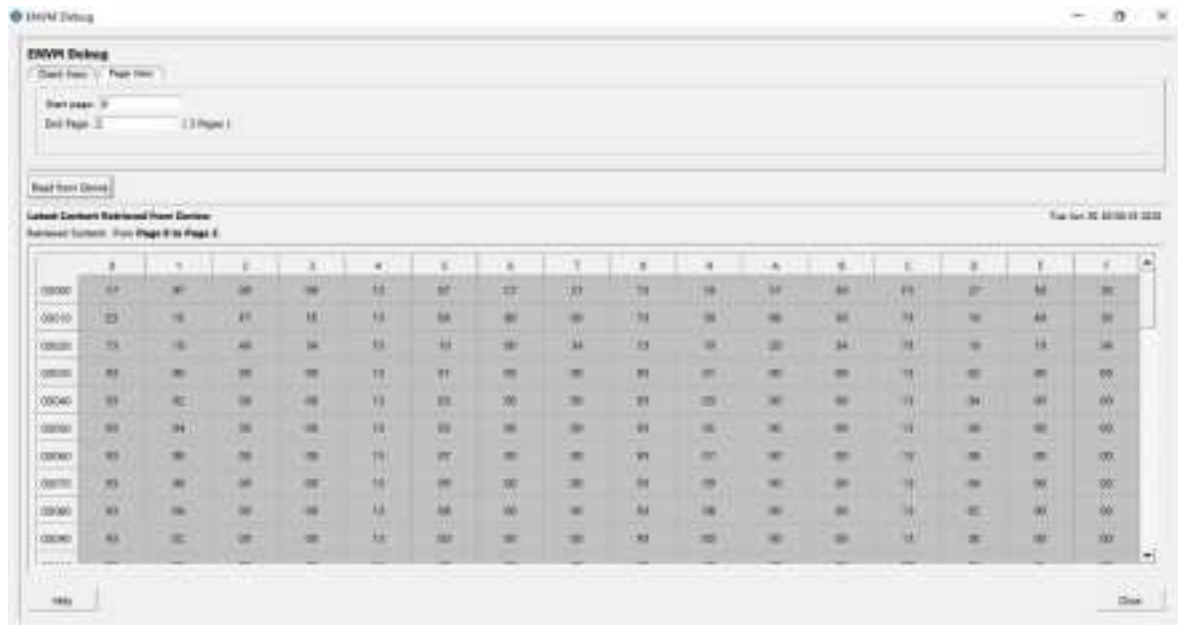
Page view can be seen when the **Page View** tab is selected in the **ENVM Debug** Window.

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 3-130. Page View Data



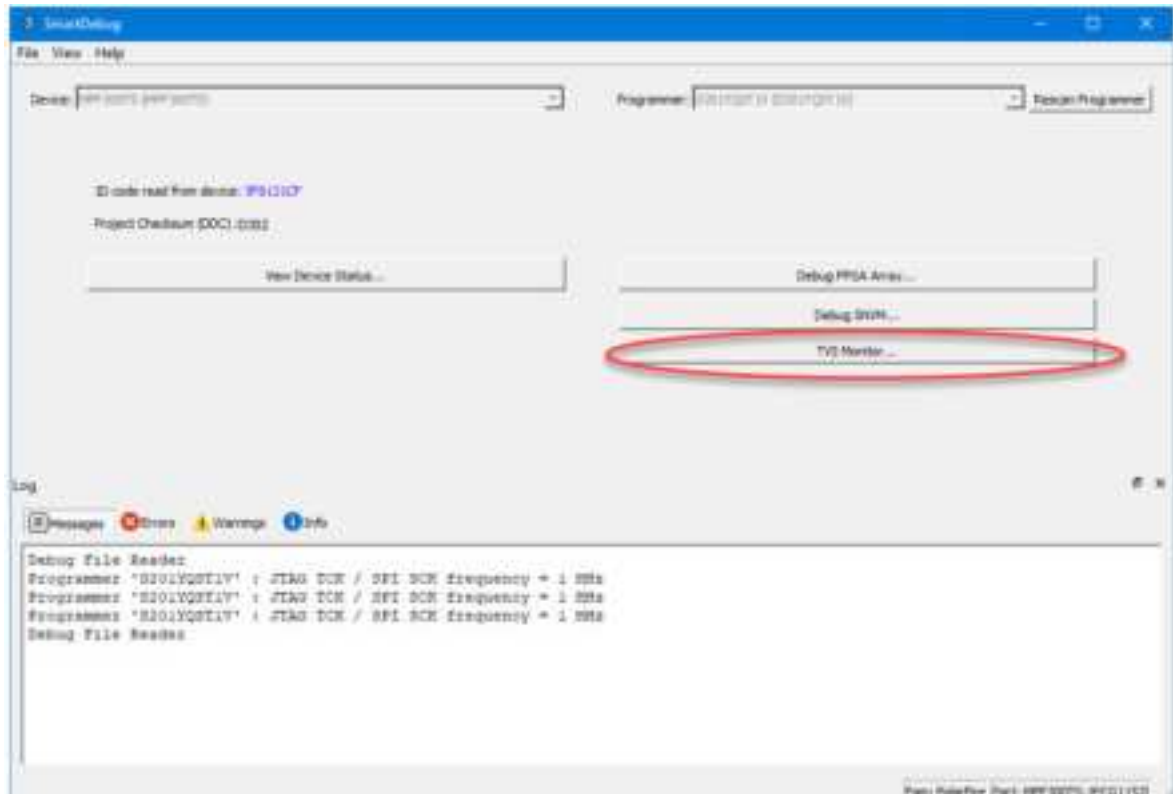
3.2.9. Measuring Die Temperature and Voltages [\(Ask a Question\)](#)

SmartDebug provides a TVS Sensor Monitor feature that measures the die temperature and voltages on a hardware device.

To use this feature:

1. In the Smart Debug main window, click **TVS Monitor**. The TVS Monitor dialog box appears.
Note: The **TVS Monitor** button appears in the window only if the TVS IP core is used in the Libero design.

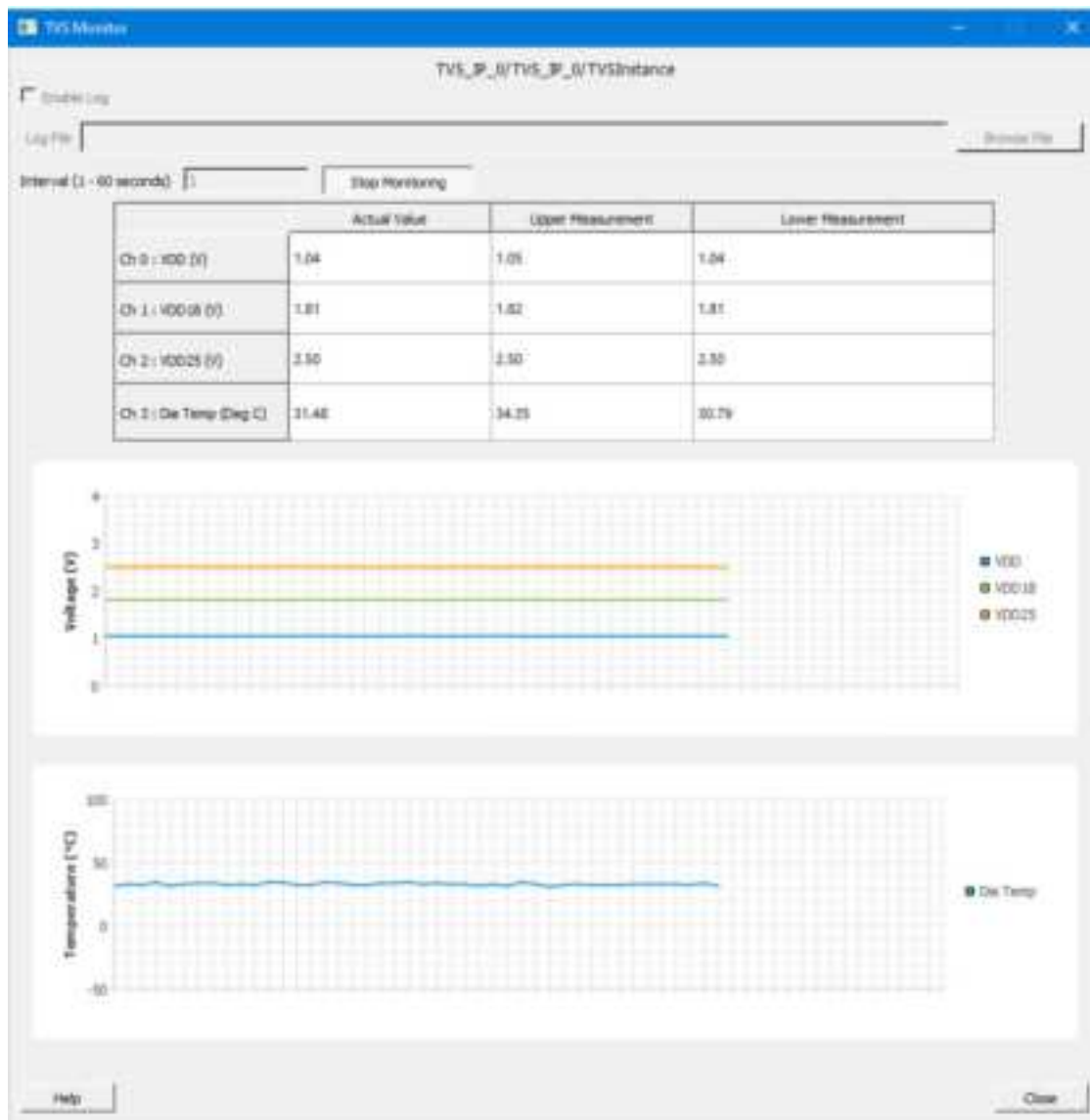
Figure 3-131. SmartDebug Main Window



2. To save the monitoring information in a .csv-formatted file, check **Log**, and then use the **Log File** field and **Browse** button to select a location for the file.
3. By default, TVS Sensor Monitor reads channel values at 1-second intervals. To use a different time interval, change the **Interval** value. Valid ranges are 1 to 60 seconds.
4. Click **Start Monitoring**. TVS Sensor Monitor uses a 4-channel ADC to read all four channel values at the specified time interval and displays the information in a table and graphs (see the example and descriptions below).
5. To stop monitoring, click the **Stop Monitoring** button.

The following figure shows an example of the table and graphs that TVS Sensor Monitor uses to display the channel and die information being read.

Figure 3-132. Example of TVS Sensor Monitor Reading Channel Values



The table in the TVS Monitor dialog box shows the following information.

Table 3-7. Table in the TVS Monitor

Row	Description
0	VDD (V)
1	VDD18 (V)
2	VDD25 (V)
3	Die temperature shown in Celsius

The graphs in the TVS Monitor dialog box show the actual values from column 1 in the table above the graphs. The plotted values are rotational. Initially, 60 period values are plotted for each channel. After 60 points are shown on each channel, older values are replaced by new values.

Table 3-8. Graphs in the TVS Monitor

Column	Description
Upper Measurement	Channel maximum value read from the device.
Lower Measurement	Channel minimum value read from the device.
Voltage graph	Actual values of channels 0, 1, and 2.
Temperature graph	Die temperature.

If you output the information to a log file, the data appears in the following format:

```
TimeStamp, VDD (V), VDD18 (V), VDD25 (V), Die Temperature (Deg C)
20:58:24, 1.04488, 1.80988, 2.49988, 35.7875
20:58:25, 1.04887, 1.80587, 2.49988, 35.7875
20:58:26, 1.04488, 1.80988, 2.50388, 35.7875
```

Note: You can read TVS values using the TCL command `tv_monitor`. If you run TVS Monitor from the user interface, however, it does not record the TCL command. As a result, using the **Export Script File** option in the **Project** menu does not export this command to a script file.

3.3. SmartFusion 2, IGLOO 2, and RTG4 Debug Elements [\(Ask a Question\)](#)

The following sections describe the debug elements for SmartFusion 2, IGLOO 2, and RTG4.

3.3.1. Debug SerDes [\(Ask a Question\)](#)

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

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

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

3.3.1.1. Debug SerDes - Configuration [\(Ask a Question\)](#)

3.3.1.1.1. Configuration Report [\(Ask a Question\)](#)

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

Table 3-9.

Report Element	Description
Lane mode	Programmed mode on a SerDes lane as defined by the SerDes system register.
PMA Ready	Shows whether PMA completed its internal calibration sequence for the specific lane and whether the PMA is operational. For details, see the UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide .
TxPLL status	Loss-of-lock status for the TXPLL is asserted and remains asserted until the PLL reacquires lock.
RxPLL status	Shows whether the CDR PLL frequency is not grossly out of range of with incoming data stream.
Refresh Report	Click to update the contents of your SerDes Configuration Report. Changes to the specified SerDes register programming can be read back to the report.

3.3.1.1.2. SerDes Register Read or Write [\(Ask a Question\)](#)

You can provide a script to read/write commands to access the SerDes control/status register map. Enter the script file path in the **Script** text box or click the **Browse** button to navigate to your script file. Click **Execute** to run the script.

Attention: SmartDebug TCL scripts that write to the `UPDATE_SETTINGS` register may not work in a few scenarios, if default values on some of the registers are changed by the user while configuring

the SerDes using the configuration GUI. This is applicable to RTG4, SmartFusion 2, and IGLOO 2, when configuring SerDes in XAUI, EPCS, or PCIe mode.

Background: When the user configures a `SERDES_with_initialization` block in the Libero GUI, the user has the option to adjust register values such as `RE_AMP_RATIO`, `RE_CUT_RATIO`, `TX_AMP_RATIO`, `TX_PST_RATIO`, and `TX_PRE_RATIO` to non-default values. When configuration is done, Libero generates an HDL file that implements the `SERDES_with_initialization` module. This module contains SerDes IP, CoreABC, and CoreAPB3 (a bridge connection between CoreABC and SerDes IP). CoreABC is a programmable state machine/microcontroller that initializes the SerDes IP to function according to what the user selects in the SerDes configuration GUI. This initialization runs at power-up or at system reset. The user's configuration is contained in a set of simple programming instructions inside CoreABC.

The Issue: If a user configures a `SERDES_with_initialization` block in the Libero GUI and changes default values on some of the registers, the CoreABC programming instructions will contain writes to the `SERDES_UPDATE_SETTINGS` register. If the user subsequently uses SmartDebug Tcl scripts to modify SerDes registers (for example, to tune SerDes lane performance), the user might find that his Tcl script does not update the SerDes registers when the `UPDATE_SETTINGS` register is written.

Recommendation: When the user configures the `SERDES_with_initialization` block, do not alter the default register settings for registers such as `RE_AMP_RATIO`, `RE_CUT_RATIO`, `TX_AMP_RATIO`, `TX_PST_RATIO`, `TX_PRE_RATIO`, and so on. Leaving register values as default will prevent any issue during SmartDebug TCL script SerDes register access.

Workaround: If the user needs to configure a `SERDES_with_initialization` block with non-default register values and will subsequently use SmartDebug Tcl scripts to adjust SerDes register values, the user will also have to add the SmartDebug command `serdes_lane_reset` to his Tcl script. The `serdes_lane_reset` command is added after writes to the `UPDATE_SETTINGS` register command. Adding the `serdes_lane_reset` command to the Tcl script will allow the expected register updates to occur.

Helpful hint for SmartDebug TCL scripts that access SerDes registers: Setting `CONFIG_PHY_MODE_1` in SmartDebug TCL script is not needed, as it is done by the tool during `serdes_read_register` or `serdes_write_register` command.

Helpful hint for SerDes configuration: Setting `SYSTEM_CONFIG_PHY_MODE_1` for each lane is required in CoreABC code or any other code that will directly configure the SerDes IP via the SerDes IP APB bus.

Figure 3-133. Debug SerDes - Configuration



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

3.3.2. Debug SerDes – Loopback Test [\(Ask a Question\)](#)

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

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

3.3.2.1. SerDes Lanes [\(Ask a Question\)](#)

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

3.3.2.1.1. Test Type [\(Ask a Question\)](#)

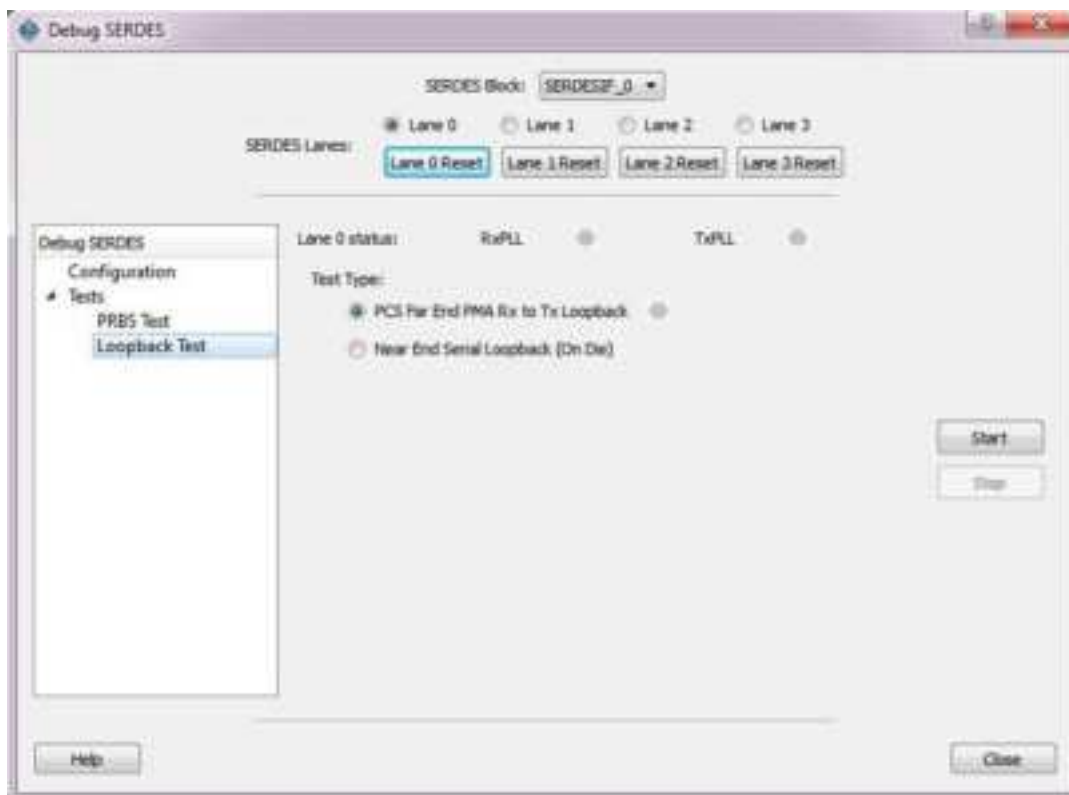
The following table describes the test type options. For details, see the [UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide](#).

Table 3-10. Test Type Options

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

Table 3-10. Test Type Options (continued)

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

Figure 3-134. Debug SerDes - Loopback Test

3.3.2.2. Running Loopback Tests in Demo Mode [\(Ask a Question\)](#)

You can run Loopback tests in demo mode. The Debug SerDes demo mode is provided to graphically demonstrate the SerDes features. By default, all channels are enabled. As shown in the following figure, the mode displays working channels and channels with connectivity issues to help you see the available options.



3.3.3. Debug SerDes – PRBS Test [\(Ask a Question\)](#)

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

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

3.3.3.1. SerDes Lanes [\(Ask a Question\)](#)

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

Figure 3-135. SerDes Lanes - Single Lane Selected

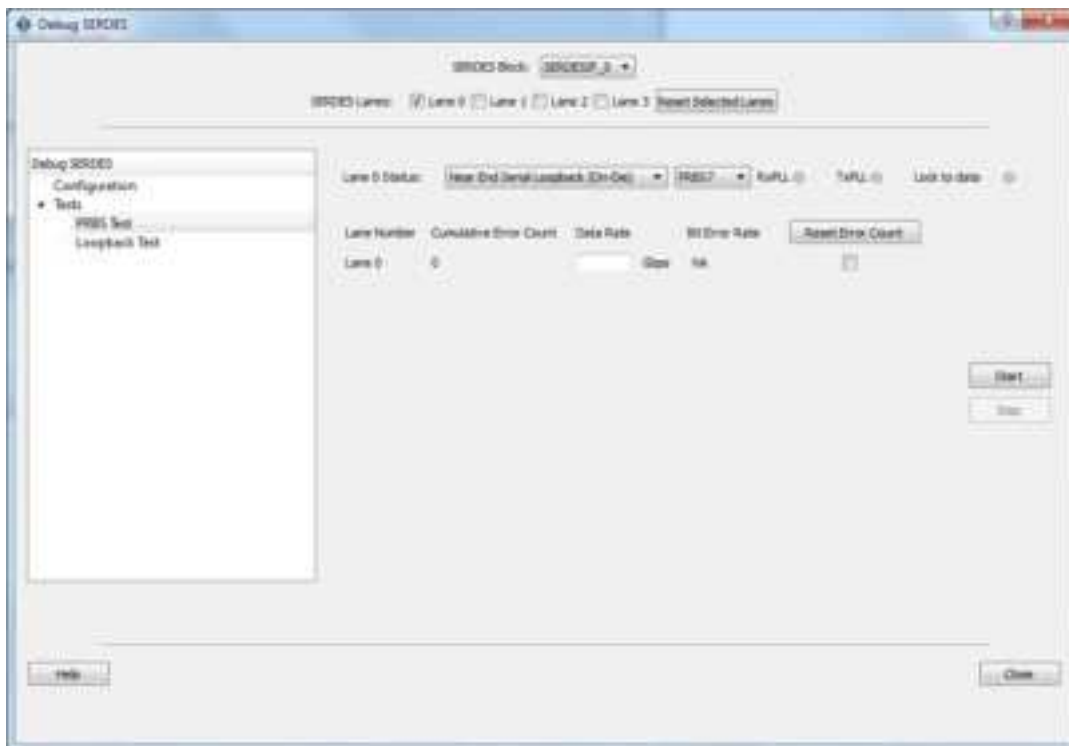
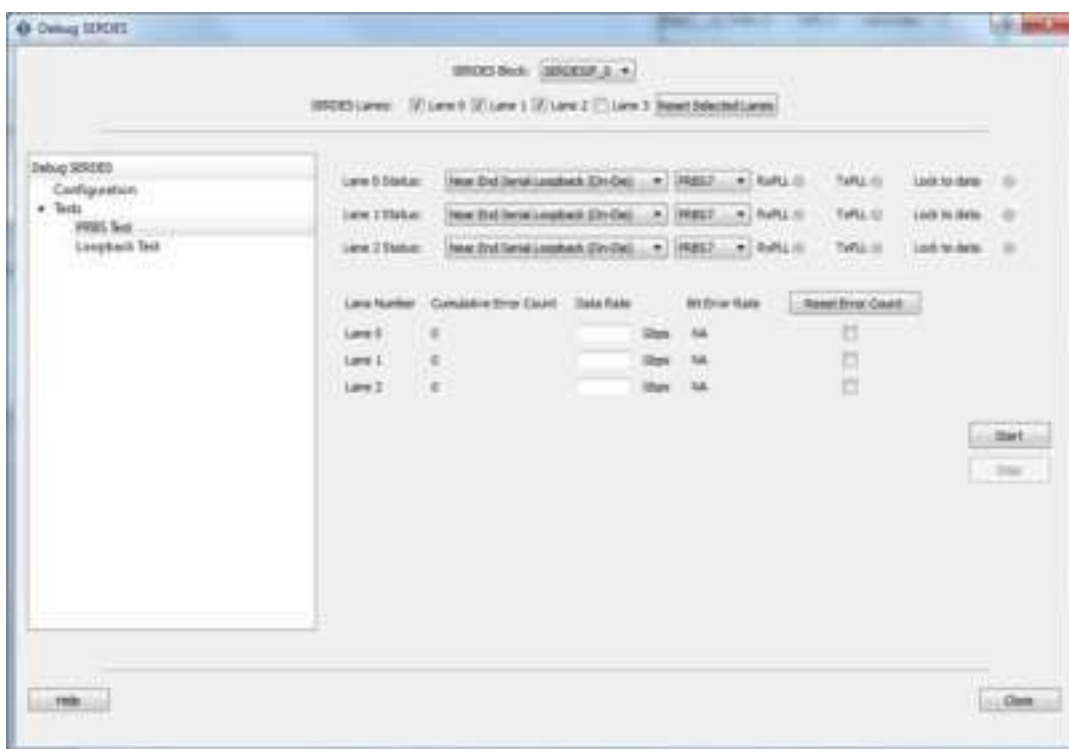


Figure 3-136. SerDes Lanes - Multiple Lanes Selected



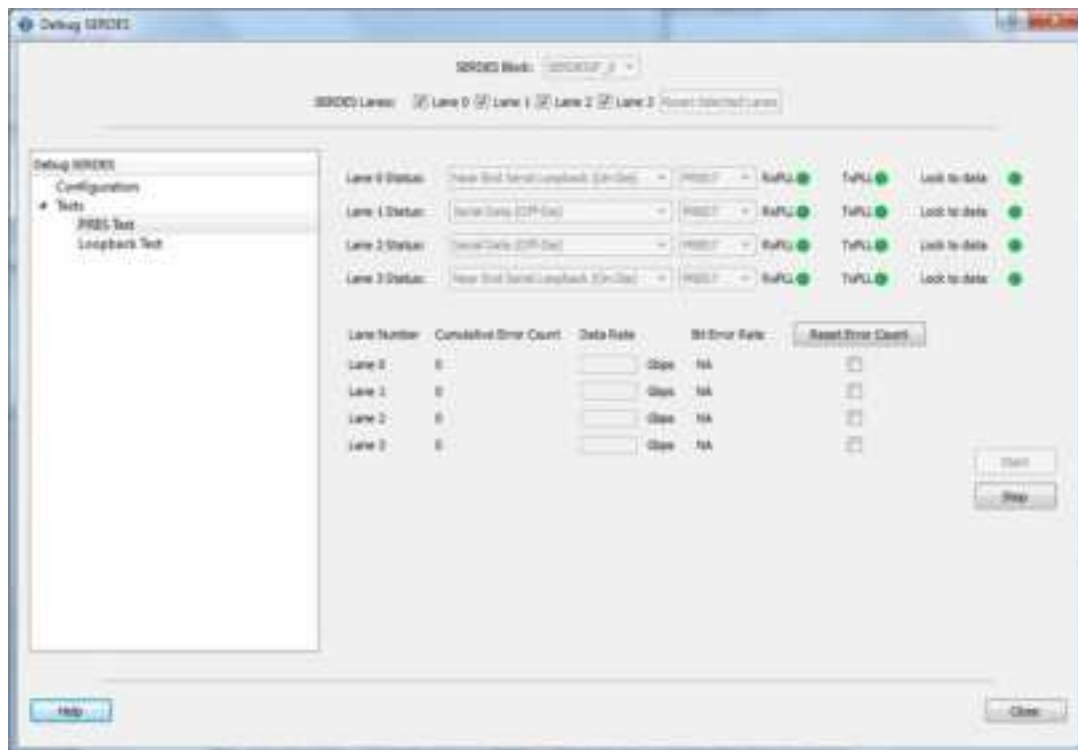
3.3.3.2. Test Type [\(Ask a Question\)](#)

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

Table 3-11. Test Type Options

Option	Description
Near End Serial Loopback (On-Die)	Enables a self-test of the device. The serial data stream is sent internally from the SerDes TX output and folded back onto the SerDes RX input.
Serial Data (Off-Die)	Normal system operation where the data stream is sent off-chip from the TX output and must be connected to the RX input via a cable or other type of electrical interconnection.

Figure 3-137. Test Type Example



3.3.3.3. Pattern [\(Ask a Question\)](#)

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

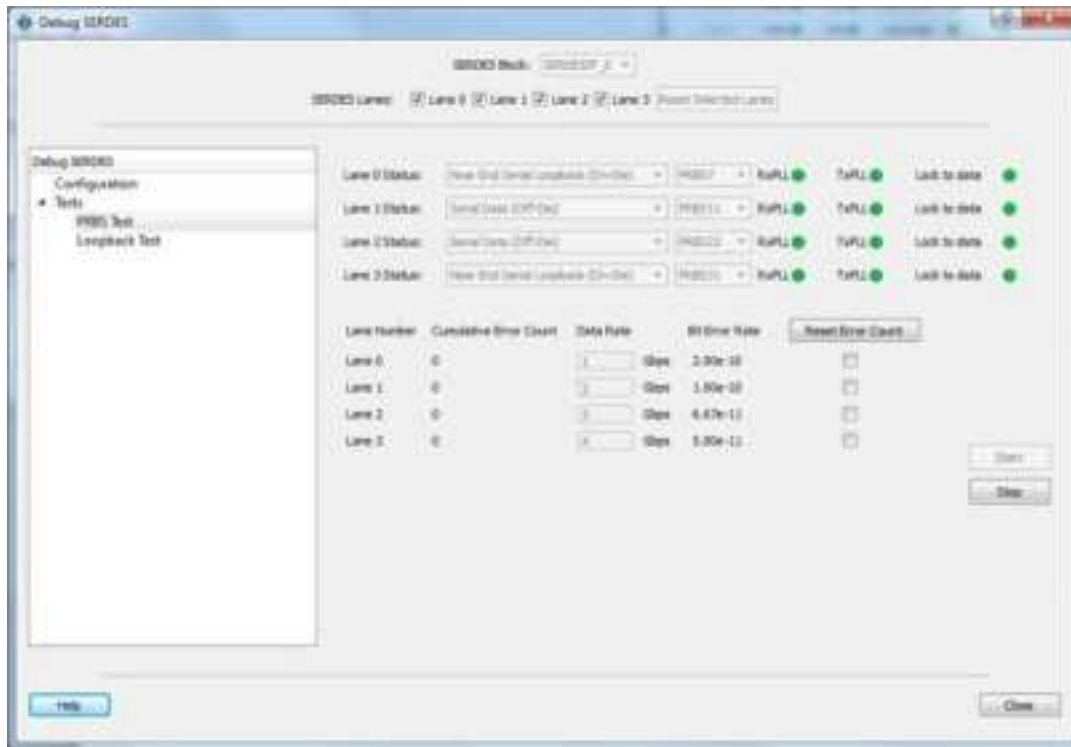
Table 3-12. Diagnostic Pattern

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

3.3.3.4. Cumulative Error Count [\(Ask a Question\)](#)

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

Figure 3-138. Debug SerDes - PRBS Test



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

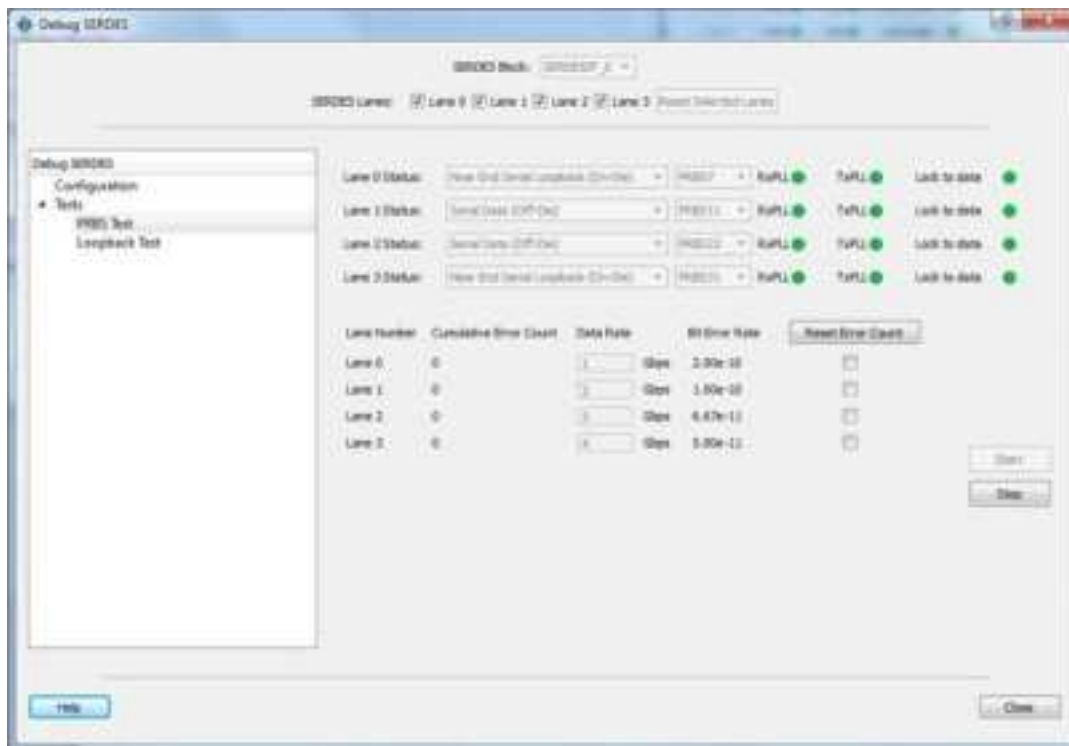
3.3.3.5. Bit Error Rate [\(Ask a Question\)](#)

The Bit Error Rate is displayed per lane. If you did not specify a Data Rate, the Bit Error Rate displays the default NA. When the PRBS test is started, the Cumulative Error Count and Bit Error Rate are updated every second.

You can select specific lanes and click **Reset Error Count** to clear the Cumulative Error Count and Bit Error Rate fields of the selected lanes.

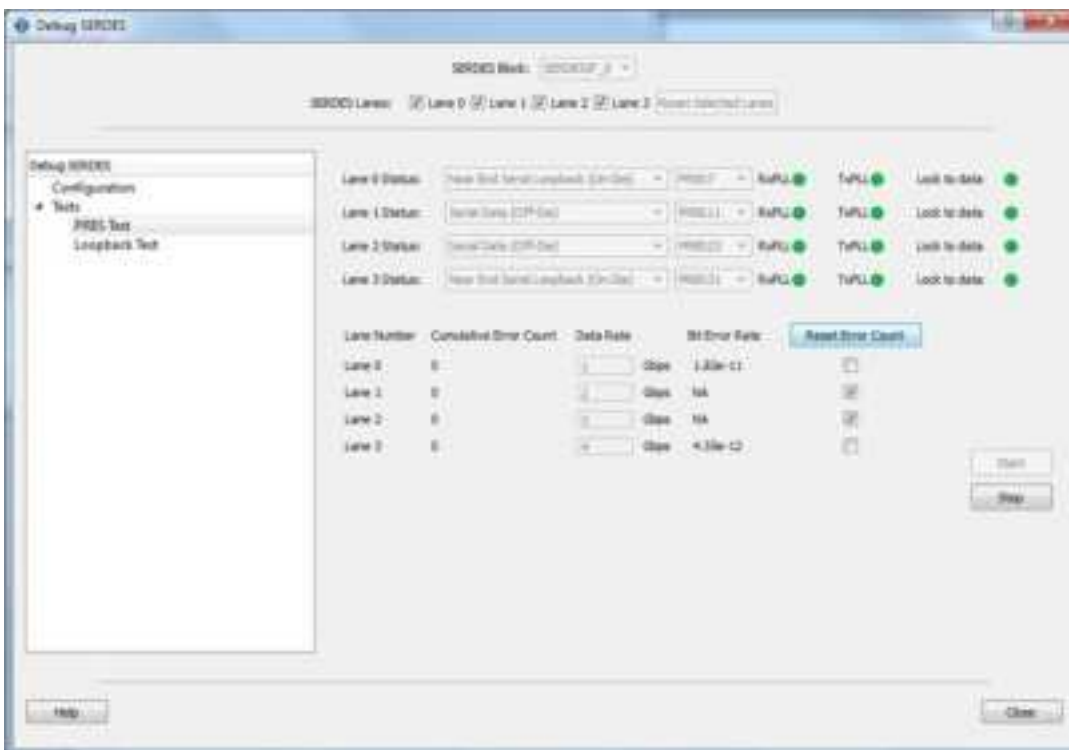
In the following example, the Bit Error Rate is displayed for all lanes.

Figure 3-139. Bit Error Rate Example - All Lanes



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

Figure 3-140. Reset Error Count Example



3.3.3.6. Running PRBS Tests in Demo Mode [\(Ask a Question\)](#)

You can run Multi-Lane PRBS tests in demo mode. The Debug SerDes demo mode is provided to graphically demonstrate the SerDes features. By default, all channels are enabled. As shown in the following figure, the mode displays working channels and channels with connectivity issues to help you see the available options.



Notes

- The formula for calculating the BER is as follows:

$$BER = (\#bit\ errors + 1) / \#bits\ sent \quad \#bits\ sent = Elapsed\ time / bit\ period$$

- When you click the **Start** button, the BER is updated every second for the entered data rate and errors are observed. If you do not enter any data rate, the BER is set to the default NA.
- When you click the **Stop** button, the BER resets to default.
- When you click the **Reset** button, the BER resets to default.
- If no test is in progress, the BER remains in the default value.
- If the PRBS test is in progress, the BER calculation restarts.

3.3.4. Debug SerDes – PHY Reset [\(Ask a Question\)](#)

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

3.3.4.1. Lane Reset Behavior for SerDes Protocols Used in the Design [\(Ask a Question\)](#)

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

For more information about soft reset, see the [UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide](#).

4. Frequently Asked Questions [\(Ask a Question\)](#)

The following topics describe frequently asked questions about SmartDebug.

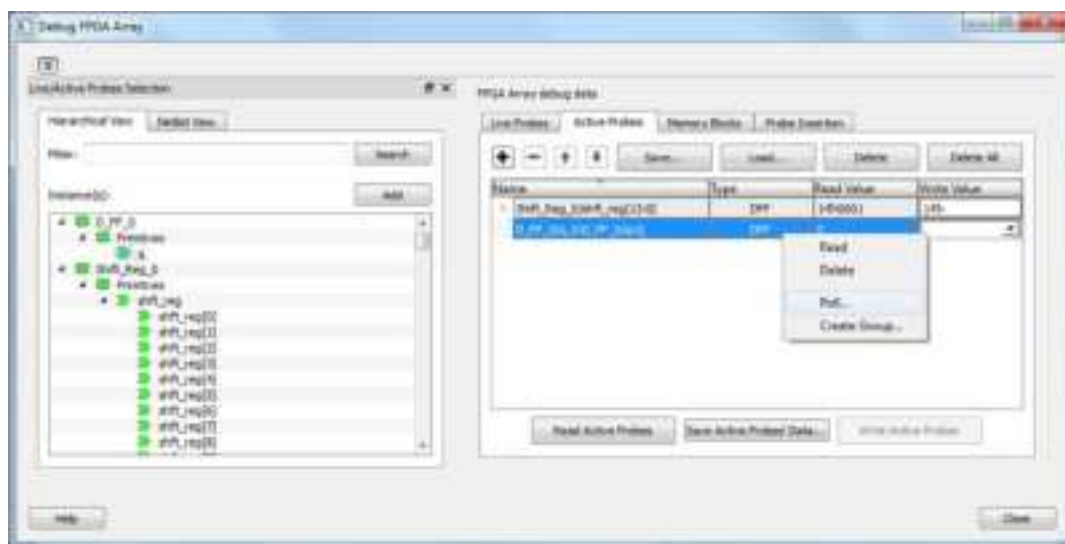
4.1. SmartDebug FAQs for PolarFire [\(Ask a Question\)](#)

The following topics contains frequently asked questions related to using SmartDebug with PolarFire.

4.1.1. How Do I Monitor a Static or Pseudo-static Signal? [\(Ask a Question\)](#)

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**.



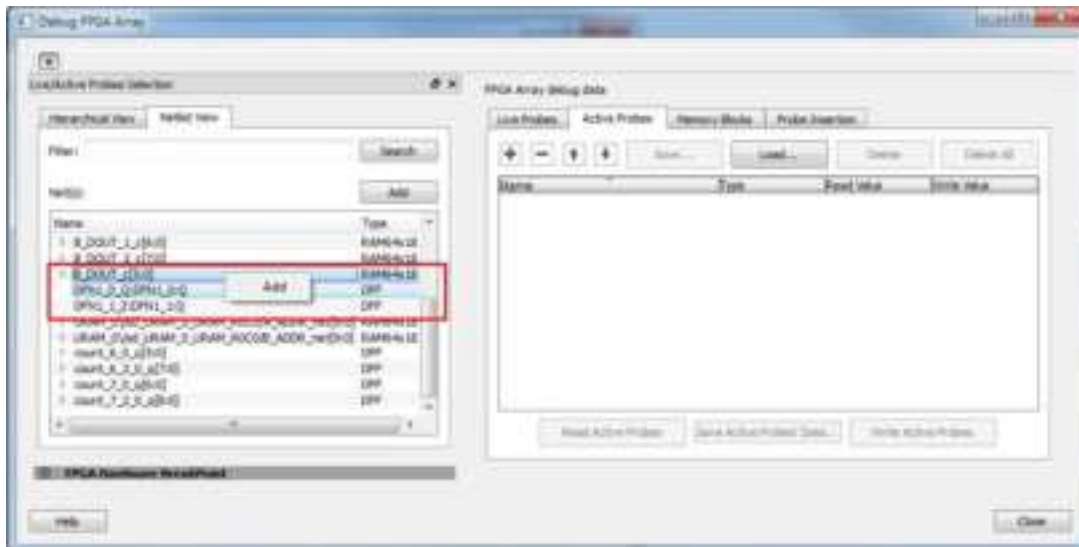
3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click **Start Polling**.



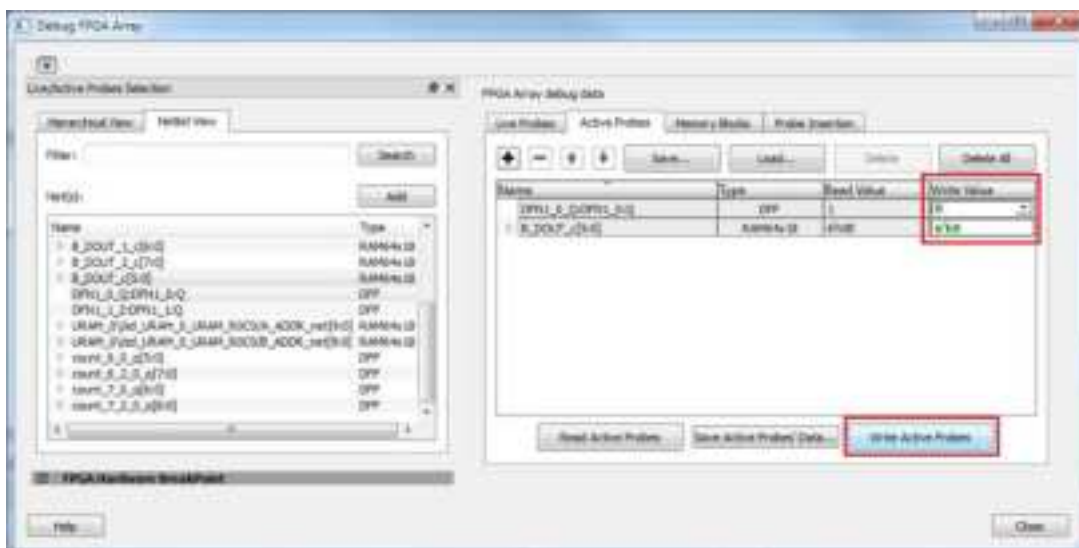
4.1.2. How Do I Force a Signal to a New Value? [\(Ask a Question\)](#)

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**.

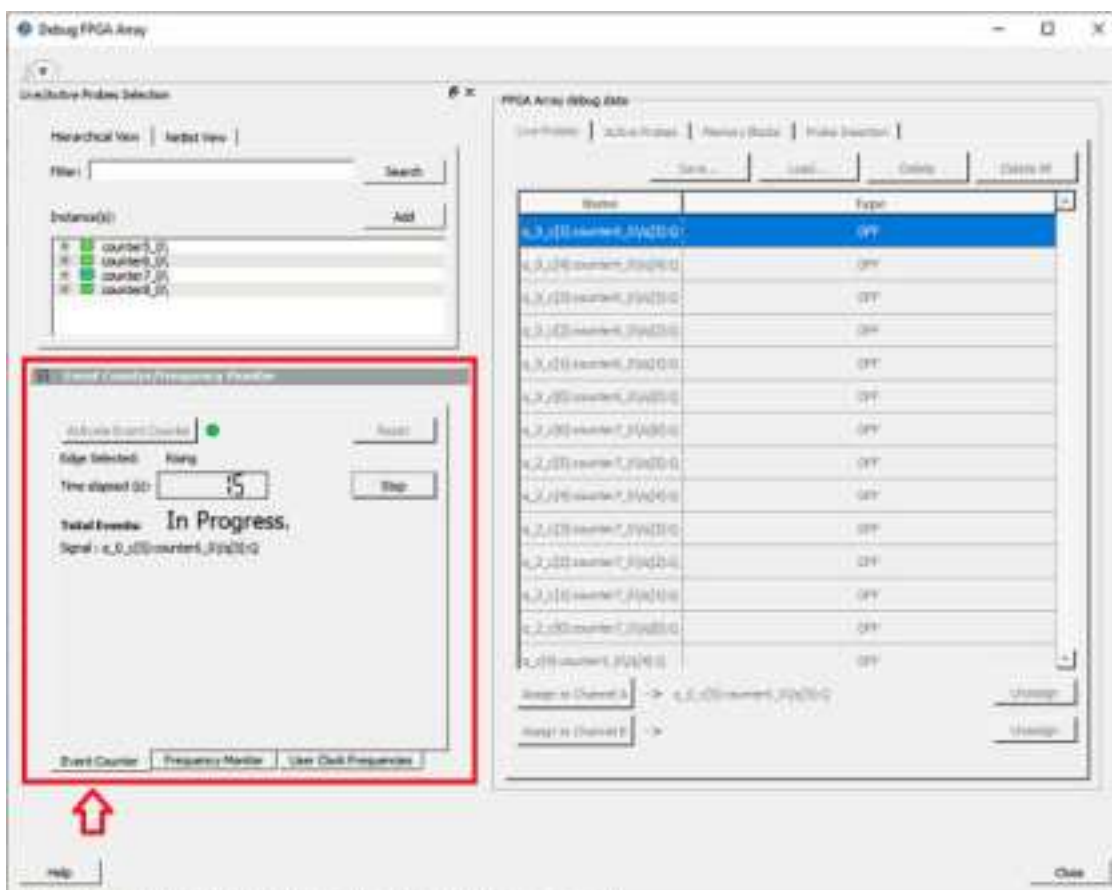


4.1.3. How Do I Count the Transitions on a Signal? [\(Ask a Question\)](#)

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.

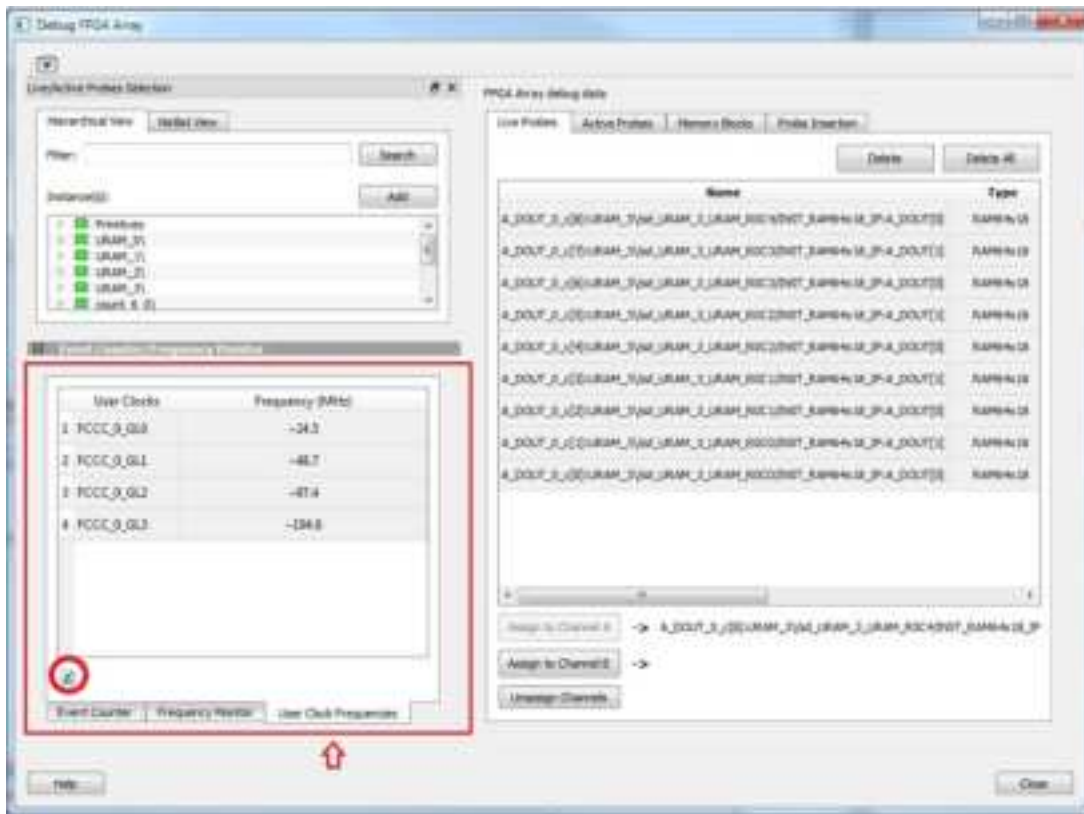


4.1.4. How Do I Monitor or Measure a Clock? [\(Ask a Question\)](#)

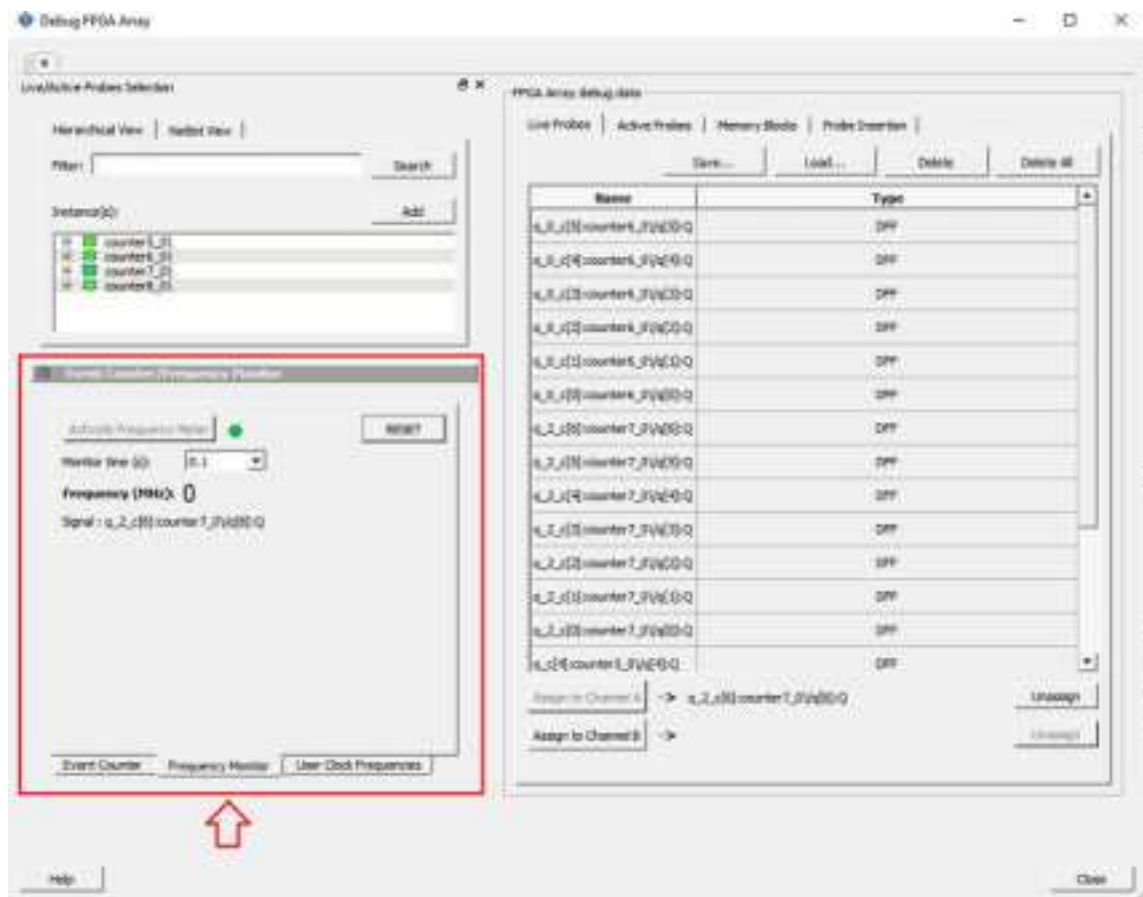
You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in the **Project Settings** dialog box.

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.

4.1.5. How Do I Perform Simple SmartBERT Tests? [\(Ask a Question\)](#)

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

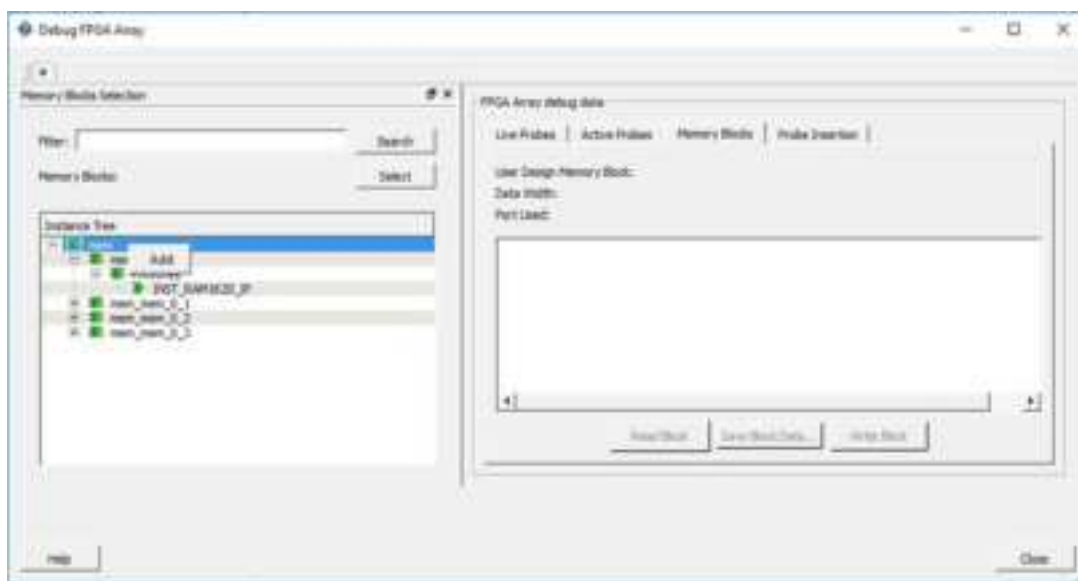
To perform a SmartBERT test, in the **SmartBERT** page of the **Debug Transceiver** dialog box, select to run a PRBS test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see [SmartBERT](#).

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

4.1.6. How Do I Read LSRAM or USRAM Content? [\(Ask a Question\)](#)

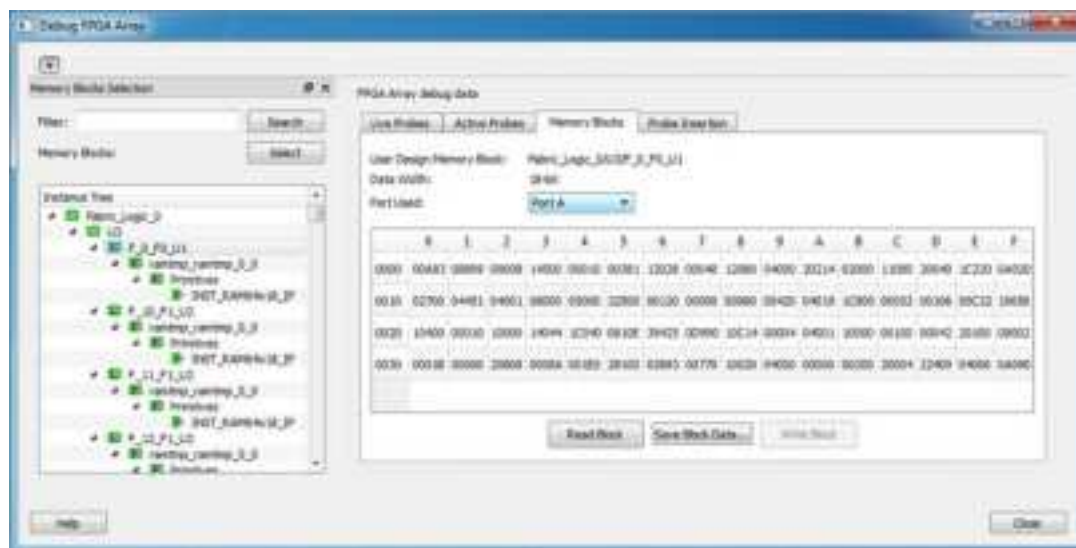
To read RAM content:

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



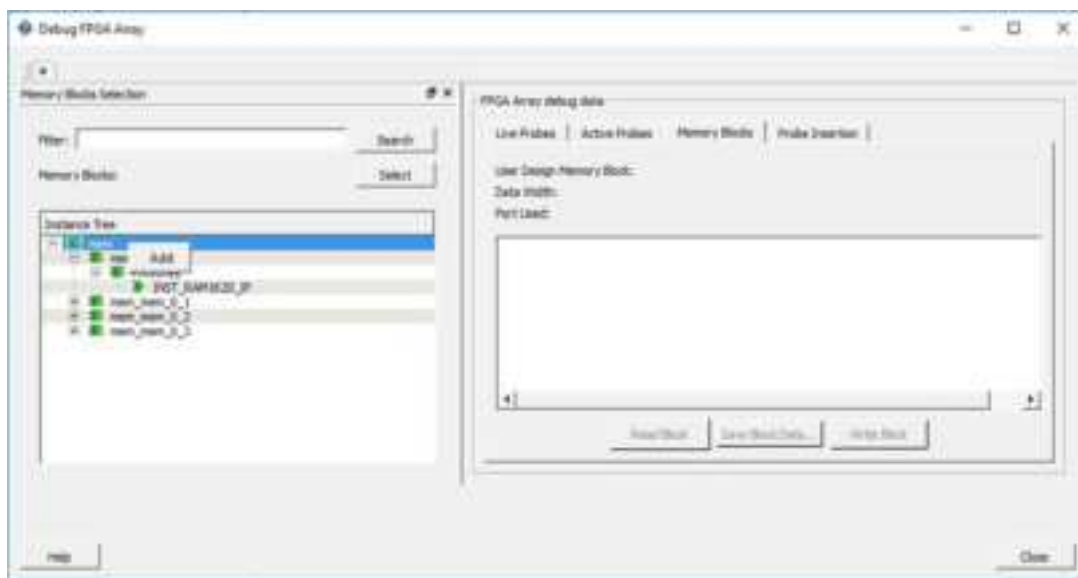
For more information, see [Memory Blocks](#).

4.1.7. How Do I Change the Content of LSRAM or USRAM? [\(Ask a Question\)](#)

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:
 - Click **Select**.
 - Right click and choose **Add**.
 - Drag the block to the **Memory Blocks** tab.
5. Click **Read Block**. The memory content matrix appears.
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](#).

4.1.8. How Do I Read the Health Check of the Transceiver? [\(Ask a Question\)](#)

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 4-1. Debug Transceiver

Lanes	PF_XCVR_Q0_L2_L3	PF_XCVR_Q1_L2_L3	PF_XCVR_Q2_L2_L3	PF_XCVR_Q3_L2_L3
LANE0				
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
LANE1				
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](#).

4.2. SmartDebug FAQs for SmartFusion 2, IGLOO 2, and RTG4 [\(Ask a Question\)](#)

The following topics contains frequently asked questions related to using SmartDebug with SmartFusion 2, IGLOO 2, and RTG4.

4.2.1. Embedded Flash Memory (NVM) - Failure when Programming/Verifying [\(Ask a Question\)](#)

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

1. In the **Inspect Device** window click **View Flash Memory Content**.
2. Select the Flash Memory block and client (or page range) to retrieve from the device:
 - a. Click **Read from Device**; the retrieved data appears in the lower part of the window.
 - b. Click **View Detailed Status**.
Note: You can use the `check_flash_memory` and `read_flash_memory` Tcl commands to perform diagnostics similar to the preceding commands.
 - c. To reset the corrupted NVM pages, either re-program the pages with your original data or 'zero-out' the pages by using the Tcl command `recover_flash_memory`.

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

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

4.2.2. Analog System Not Working as Expected [\(Ask a Question\)](#)

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

- System supply issue. To troubleshoot:
 - a. Physically verify that all the supplies are properly connected to the device and they are at the proper level. Then confirm by running the Device Status.
 - b. Physically verify that the relevant channels are correctly connected to the device.
- Analog system is not properly configured. You can confirm this by examining the Analog System.

4.2.3. ADC Not Sampling the Correct Value [\(Ask a Question\)](#)

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

- System supply issues: Run the device status to confirm.
- Analog system is not configured at all: To confirm, read out the ACM configuration and verify if the ACM content is all zero.
- Analog system is not configured correctly: To confirm:
 - a. Read out the ACM configuration and verify that the configuration is as expected.
 - b. Once analog block configuration has been confirmed, you can use the `sample_analog_channel` Tcl command for debug sampling of the analog channel with user-supplied sampling parameters.
 - c. If you have access to your Analog System Builder settings project (`<Libero IDE project>/Smartgen/AnalogBlock`), you may use the compare function provided by the tool.

4.2.4. How Do I Unlock the Device Security So That I Can Debug? [\(Ask a Question\)](#)

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

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

4.2.5. How Do I Export a Report? [\(Ask a Question\)](#)

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

For more information about Tcl commands supported by SmartDebug, see the [Tcl Commands Reference Guide](#).

4.2.6. How Do I Generate Diagnostic Reports for My Target Device? [\(Ask a Question\)](#)

A set of diagnostic reports can be generated for your target device depending on which silicon feature you are debugging. A set of Tcl commands are available to export those reports.

Note: Select **File > Run Script** to execute a Tcl command.

The following is a summary of the Tcl commands based on the silicon features.

For the Overall Device:

- read_device_status
- read_id_code

For FlashROM:

- compare_flashrom_client
- read_flashrom

For Embedded Flash Memory (NVM):

- compare_memory_client
- check_flash_memory
- read_flash_memory

For Analog Block:

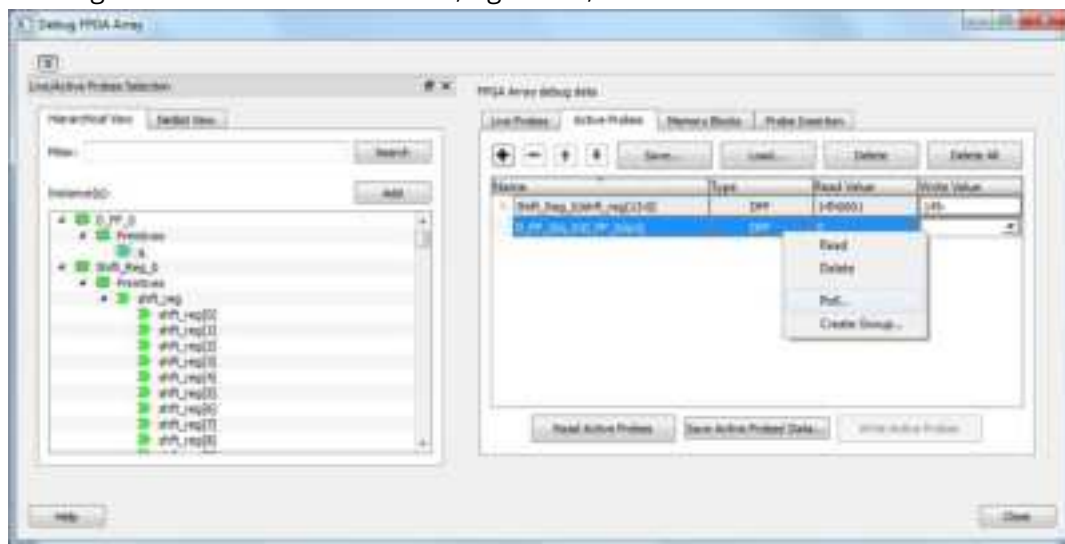
- read_analog_block_config
- compare_analog_config
- sample_analog_channel

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

4.2.7. How Do I Monitor a Static or Pseudo-static Signal? [\(Ask a Question\)](#)

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...**



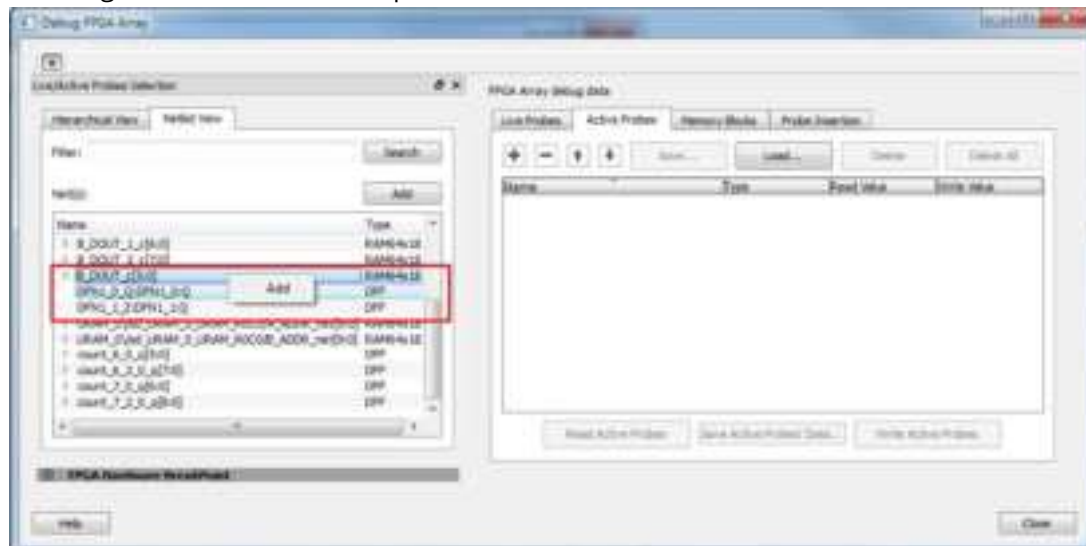
3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click **Start Polling**.



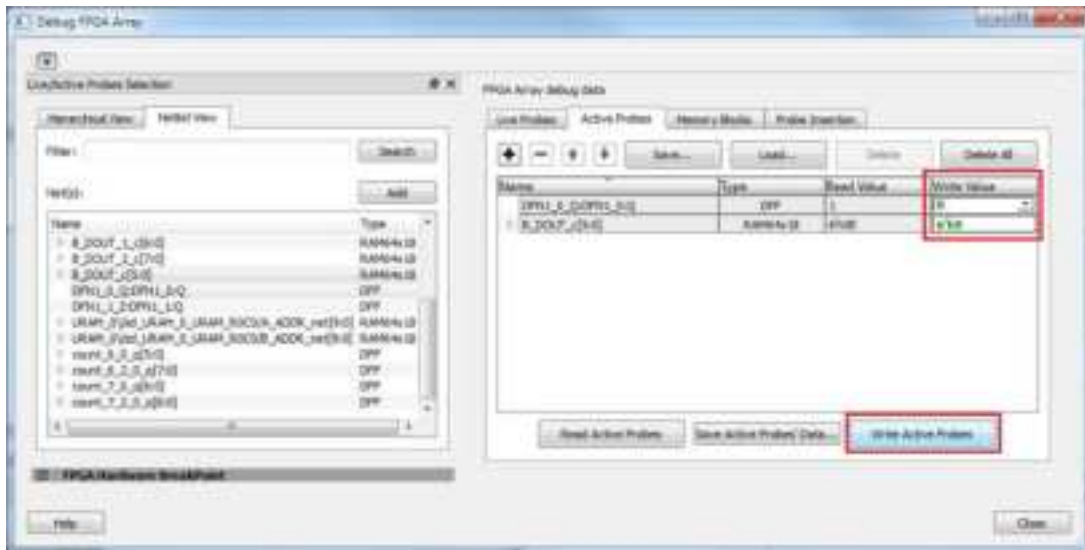
4.2.8. How Do I Force a Signal to a New Value? [\(Ask a Question\)](#)

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**.

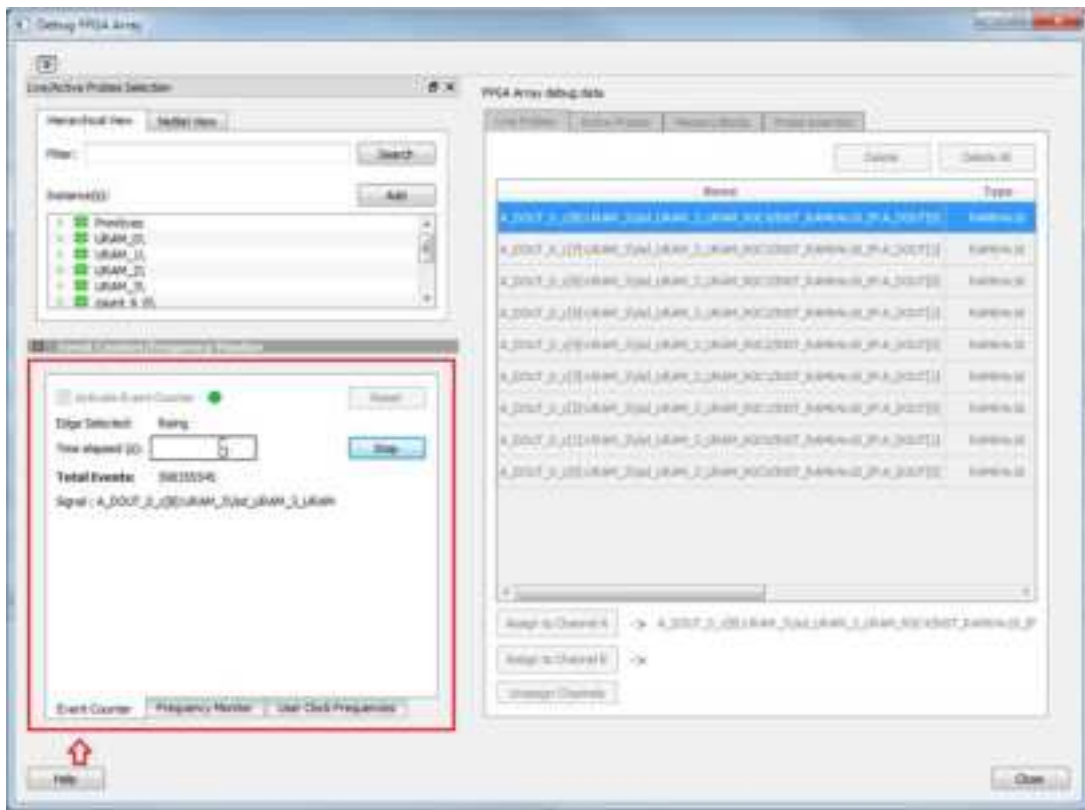


4.2.9. How Do I Count the Transitions on a Signal? [\(Ask a Question\)](#)

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

To count the transitions on a signal:

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

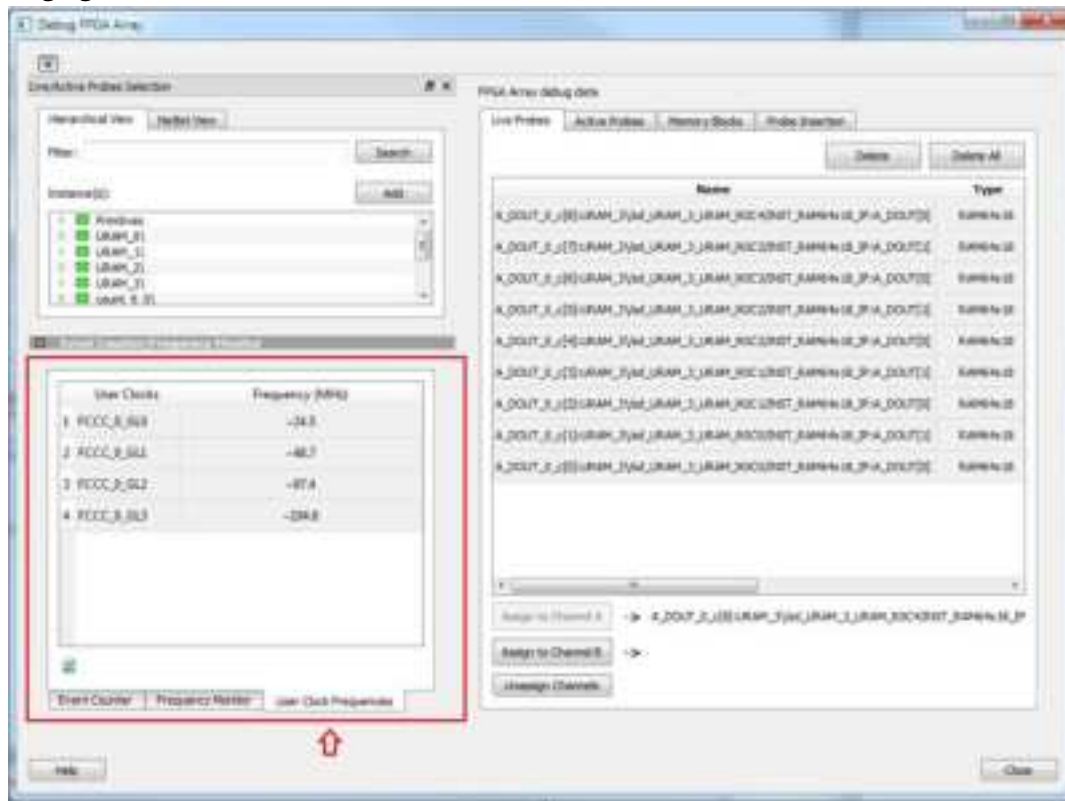


4.2.10. How Do I Monitor or Measure a Clock? [\(Ask a Question\)](#)

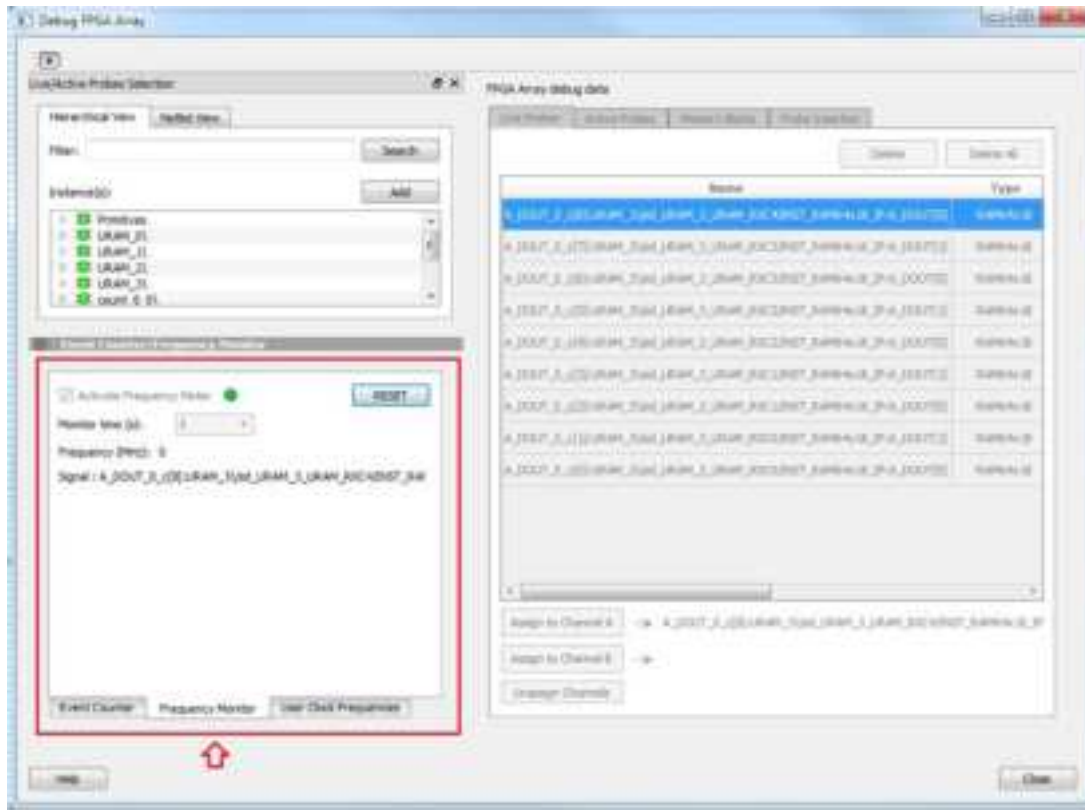
You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in Project Settings dialog box.

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

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



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



4.2.11. How Do I Perform Simple PRBS and Loopback Tests? [\(Ask a Question\)](#)

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

To perform a PRBS test, in the Debug SerDes dialog box, select **PRBS Test** to run a PRBS test on-die or off-die. For more information, see [Debug SerDes – PRBS Test](#).

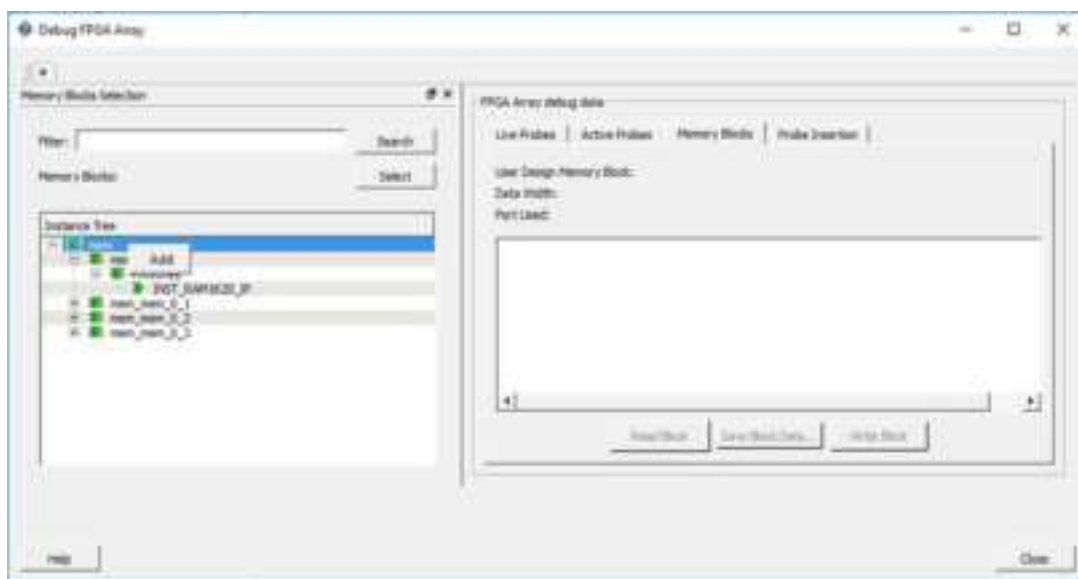
To perform a PRBS test, in the Debug SerDes dialog box, select PRBS Test to run a PRBS test on-die or off-die. For more information, see [Debug SerDes – PRBS Test](#).

To perform a loopback test, in the Debug SerDes dialog box, select **Loopback Test** to run a near end serial loopback /far end PMA Rx to Tx loopback test. For more information, see [Debug SerDes – Loopback Test](#).

4.2.12. How Do I Read LSRAM or USRAM Content? [\(Ask a Question\)](#)

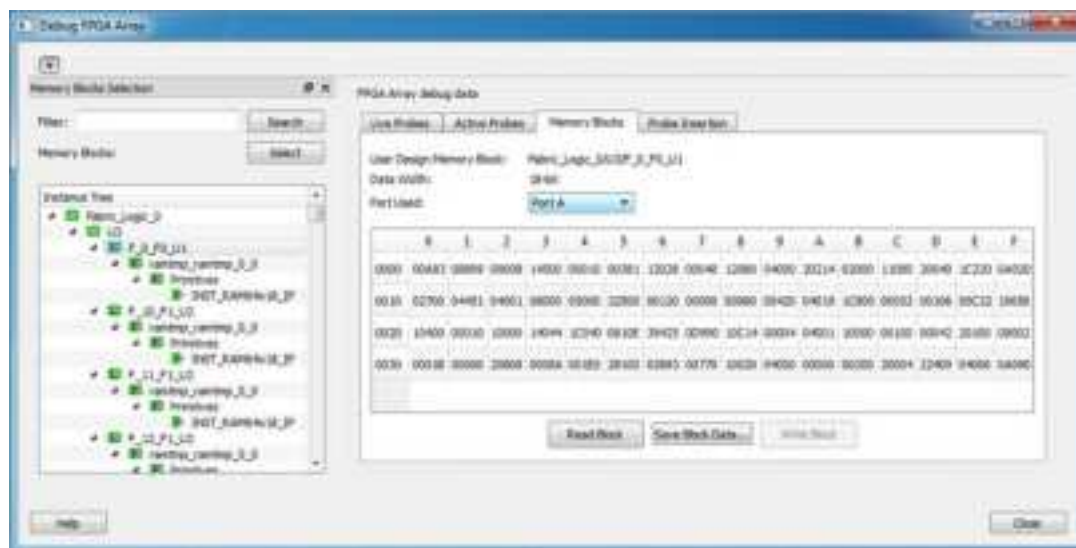
To read RAM content:

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.

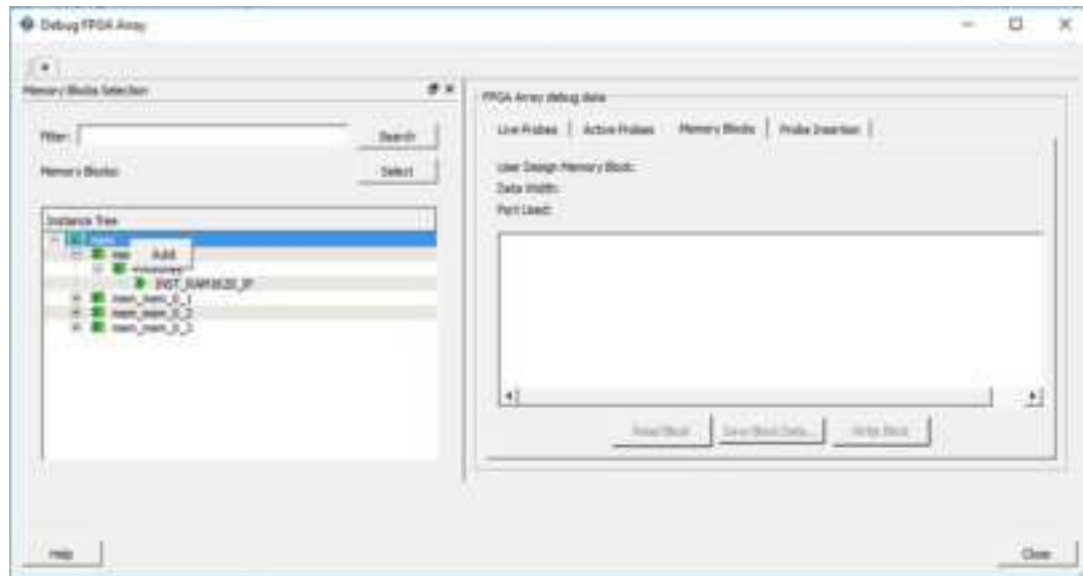
3. Add the block in one of the following ways:
 - Click **Select**.
 - Right click and choose **Add**.
 - Drag the block to the **Memory Blocks** tab.
4. Click **Read Block** to read the content of the block.



4.2.13. How Do I Change the Content of LSRAM or USRAM? [\(Ask a Question\)](#)

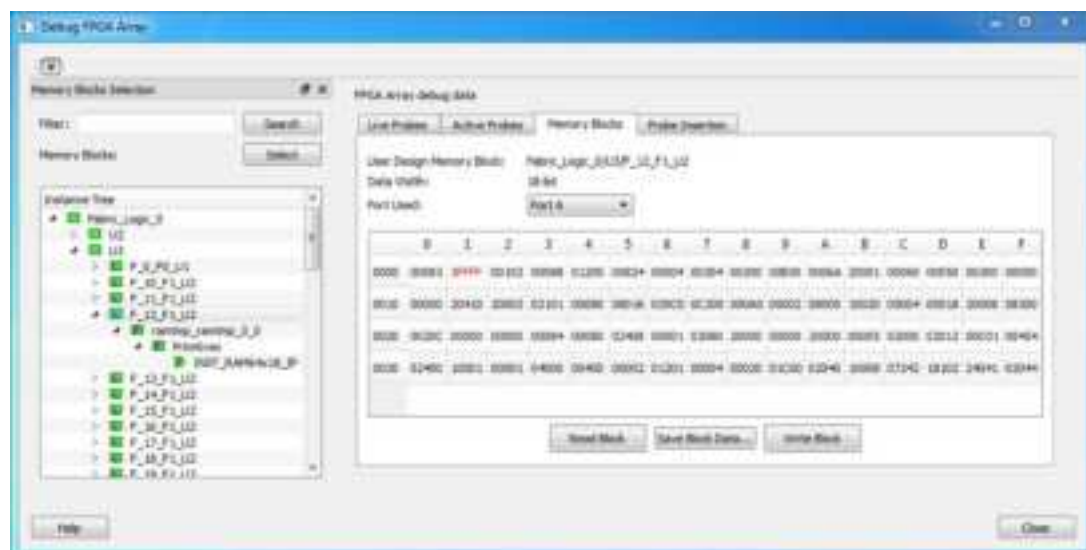
To change the content of LSRAM or USRAM:

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



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

4. Add the memory block in one of the following ways:
 - Click **Select**.
 - Right click and choose **Add**.
 - 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.



4.2.14. How Do I Read the Health Check of the SerDes? [\(Ask a Question\)](#)

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

1. Review the **Configuration Report**, which returns PMA Ready, TxPLL status, and RxPLL status. For SerDes to function correctly, PMA ready should be true, and TxPLL and RxPLL status should be locked. The Configuration Report can be found in the Debug SerDes dialog box under Configuration. See [Debug SerDes \(SmartFusion 2, IGLOO 2, and RTG4\)](#).



2. Run the **PRBS Test**, which is a Near End Serial Loopback tests on selected lanes. This should result in 0 errors in the Cumulative Error Count column. See [Debug SerDes – PRBS Test](#).

4.2.15. Where Can I Find Files to Compare My Contents/Settings? [\(Ask a Question\)](#)

FlashROM

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

Embedded Flash Memory (NVM)

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

4.2.16. What is a UFC File? What is an EFC File? [\(Ask a Question\)](#)

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

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

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

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

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

4.2.17. Is My FPGA Fabric Enabled? [\(Ask a Question\)](#)

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

FPGA Array Status: Programmed and Enabled

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

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

4.2.18. Is My Embedded Flash Memory (NVM) Programmed? [\(Ask a Question\)](#)

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

To examine the NVM content, see the **FlashROM Memory Content** dialog box.

4.2.19. How Do I Display Embedded Flash Memory (NVM) Content in the Client Partition? [\(Ask a Question\)](#)

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

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

4.2.20. How Do I Know Whether I Have Embedded Flash Memory (NVM) Corruption? [\(Ask a Question\)](#)

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

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

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

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

4.2.21. Why Does Embedded Flash Memory (NVM) Corruption Happen? [\(Ask a Question\)](#)

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

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

4.2.22. How do I Recover from Embedded Flash Memory Corruption? [\(Ask a Question\)](#)

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

4.2.23. What is a JTAG IR-Capture Value? [\(Ask a Question\)](#)

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

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

4.2.24. What Does the ECC1/ECC2 Error Mean? [\(Ask a Question\)](#)

ECC is the Error Correction Code embedded in each Flash Memory page. ECC1 – One bit error and correctable.

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

4.2.25. What Happens if Invalid Firmware is Loaded into eNVM in SmartFusion 2 Devices? [\(Ask a Question\)](#)

When invalid firmware is loaded into eNVM in SmartFusion 2 devices, Arm® Cortex®-M3 will not be able to boot and issues reset to MSS continuously. eNVM content using View Flash Memory content will read zeroes in SmartDebug.

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

4.2.26. Can I Compare Serialization Data? [\(Ask a Question\)](#)

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

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

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

4.2.27. Can I Tell What Security Options are Programmed in My Device? [\(Ask a Question\)](#)

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

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

4.2.28. How Do I Interpret Data in the Device Status Report? [\(Ask a Question\)](#)

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

- IDCode
- User Information
- Device State
- Factory Data
- Security Settings

4.2.29. How Do I Interpret Data in the Flash Memory (NVM) Status Report? [\(Ask a Question\)](#)

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

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

4.2.29.1. Data ECC2 Check: Pass [\(Ask a Question\)](#)

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

Table 4-1. Embedded Flash Memory Status Report Description

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

5. SmartDebug Tcl Commands [\(Ask a Question\)](#)

This section describes the SmartDebug Tcl Commands for the PolarFire, PolarFire SoC, SmartFusion 2, IGLOO 2, and RTG4 device families.

5.1. Smart Debug Tcl Commands [\(Ask a Question\)](#)

5.1.1. add_probe_insertion_point [\(Ask a Question\)](#)

Description

This Tcl command adds a probe point to be connected to user-specified I/Os for probe insertion flow. This command will fail if any of the parameters are missing.

Note:

Probe Insertion feature disabled in the SmartDebug Demo and Standalone modes.

```
add_probe_insertion_point -net {net_name} \
                          -driver {driver_name} \
                          -pin {pin_name} \
                          -port {port_name}
```

Arguments

Parameter	Type	Description
net	string	Specify name of the existing net which is added in probe insertion list. This parameter is mandatory.
driver	string	Specify driver name of the net. This parameter is mandatory.
pin	string	Specify Package pin name(that is, I/O to which the net will be routed during probe insertion). This parameter is mandatory.
port	string	Specify user-specified name for the probe insertion. This parameter is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Port name is already used.
None	Not a valid pin or already used pin.
None	Parameter 'param_name' is not defined. Valid command formatting is 'add_probe_insertion_point [-net "net_name"] \ [-driver "driver_name"] \ [-pin "pin_name"] \ [-port "port_name"]'.
None	Probe insertion operations are not supported in Standalone SmartDebug.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example adds a probe point to the probe insertion list:

```
add_probe_insertion_point -net {sw} \
    -driver {sw_buf/IN:Y} \
    -pin {Unassigned} \
    -port {Probe_Insert0}
```

See Also

- `remove_probe_insertion_point`
- `program_probe_insertion`

5.1.2. `add_to_probe_group` [\(Ask a Question\)](#)

Description

This Tcl command adds the specified probe points to the specified probe group.

This command will fail if any of the options are incorrect.

```
add_to_probe_group -name {Probe name} -group {Group name}
```

Arguments

Parameter	Type	Description
name	string	Specifies one or more probes to add.
group	string	Specifies name of the probe group.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'group' has illegal value.
None	Required parameter 'group' is missing.
None	Parameter 'name' has illegal value.
None	Required parameter 'name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'add_to_probe_group [-name "name"]+ \-group "group name"'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example adds {DFN1_0_Q:DFN1_0/U0:Q} instance to the {probe_group}.

```
add_to_probe_group -name {DFN1_0_Q:DFN1_0/U0:Q} -group {probe_group}
```

See Also

- `create_probe_group`

- remove_from_probe_group
- move_to_probe_group

5.1.3. check_flash_memory [\(Ask a Question\)](#)

Description

The command performs diagnostics of the page status and data information as follows: • Page Status – includes ECC2 check of the page status information, write count • Page Data - ECC2 check

```
check_flash_memory [-name { device_name }] \
                  [-startpage { integer_value }] \
                  [-endpage { integer_value }] \
                  [-access { all | status | data }] \
                  [-show { summary | pages }] \
                  [-file { filename }]
```

Arguments

Parameter	Type	Description
name	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
startpage	integer	Startpage value must be an integer. You must specify a -endpage along with this argument.
endpage	integer	Endpage value must be an integer. You must specify a -startpage along with this argument.
access	string	You must set -startpage and -endpage before use. Specifies what NVM information to check: page status, data or both. All: Shows the number of pages with corruption status, data corruption and out-of-range write count (default). Status: Shows the number of pages with corruption status and the number of pages with out-of-range write count. Data: Shows only the number of pages with data corruption.
show	string	This is an optional argument. You must set -startpage and -endpage before use. Specifies output level, as explained in the table below. Summary: Displays the summary for all checked pages (default). Pages: Displays the check results for each checked page.
file	string	This is an optional argument. You must set -startpage and -endpage before use. Specifies name of output file for memory check.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'show' has illegal value.
None	Parameter 'file' has illegal value.
None	Parameter 'endpage' has illegal value.
None	Missing '-endpage' argument for page range. Specify a page range with both a -startpage and an -endpage argument.
None	Parameter 'startpage' has illegal value.
None	Missing '-startpage' argument for page range. Specify a page range with both a -startpage and an -endpage argument.

check_flash_memory (continued)

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'check_flash_memory [-deviceName "device name"] [-block "integer value"] [-client "client name"] [-startpage "integer value"] [-endpage "integer value"] [-access "all status data"] [-show "summary pages"] [-file "filename"]'.
None	Invalid value for -show: 'show_value'. Value should be 'summary' or 'pages'.
None	endpage: Invalid argument value: 'endpage_value' (expecting integer value).
None	startpage: Invalid argument value: 'startpage_value' (expecting integer value).
None	Invalid value for -access: 'access_value'. Value should be 'all' or 'status' or 'data'.
None	Missing specification for Flash Memory area. Use one of: -client [-block]or -startpage -endpage -block.

Supported Families

SmartFusion 2

IGLOO 2*

RTG4*

Example

This example checks flash memory form pages 0 to 1 and saves their pages to check_flash_memory.txt file:

```
check_flash_memory -startpage 0 -endpage 1 \
  -file {check_flash_memory.txt} \
  -show {pages}
```

See Also

- read_flash_memory

5.1.4. close_project ([Ask a Question](#))**Description**

This Tcl command closes a SmartDebug project.

```
close_project
```

Arguments

Parameter	Type	Description
None	None	None

Return Type	Description
None	None

Error Codes

Error Code	Description
None	None

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This command closes the SmartDebug project:

```
close_project
```

See Also

- `new_project`
- `open_project`

5.1.5. `complete_debug_job` [\(Ask a Question\)](#)

Description

Completes the current open job and generates a job status container including cryptographically signed Job Ticket end certifiers and Certificates of Conformance (if enabled) of the programmed devices. It archives ticket data from the HSM database. The resultant Job Status container can be imported into Job Manager and validated using U-HSM. If the job status file is not specified, the information is printed in the log window and no Job Status container is created for subsequent verification.

The HSM Job can be completed only if the number of devices in each HSM ticket has been exhausted. If devices remain, the job can only be terminated by using the `-terminate` option.

Note: This command fails if there are devices left in any HSM ticket and the terminate option is not used.

```
complete_debug_job [-job_status_file "job status file"] [-terminate]
```

Arguments

Parameter	Type	Description
<code>job_status_file</code>	string	Full path to the output Job Status container, which contains End-Job Certifier and CofCs. If not specified, information is printed in the log window.
<code>terminate</code>	flag	Terminates the HSM job even if there are devices left in any HSM ticket. This parameter is optional if the number of devices in all tickets has been exhausted.

Error Codes

Error Code	Description
None	Fpeng error: Chain manager is not set.
None	Parameter 'param_name' is not defined. Valid command formatting is 'complete_debug_job [-job_status_file "job status file"] \ [-terminate]'.

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

Example

This example terminates an HSM job.

```
complete_debug_job -terminate -job_status_file {./MyJobStatusEnd}
```

5.1.6. **create_probe_group** [\(Ask a Question\)](#)

Description

This Tcl command creates a new probe group.

```
create_probe_group -name {Group name}
```

Arguments

Parameter	Type	Description
name	string	Specifies the name of the new probe group.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'create_probe_group -name "group name"'. Parameter 'name' has illegal value.
None	Parameter 'name' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example creates new probe group named "my_new_grp" :

```
create_probe_group -name my_new_grp
```

5.1.7. **debug_dds** [\(Ask a Question\)](#)

Description

This Tcl command retrieves/gets the training data from the Training IP and displays the status of different stages of training along with the eye width chart.

```
debug_dds [-dds_type type_of_dds] \
  [-data_width width] \
  [-slot_memory_slot] \
  [-inst_path ddr instance_path_from_top] \
  [-frequency frequency]

debug_dds [-deviceName "device name"] \
  -dds_type "DDR Type" \
  -data_width "integer value" \
  -slot "DDR Slot" -inst_path \
  "Instance Path from Top" \
  -frequency "decimal value"
```

Arguments

Parameter	Type	Description
ddr_type	string	Specifies DDR type. Supported DDR types are: DDR4/DDR3/LPDDR4.
data_width	integer	Specify data width. Supported data widths are 16, 32, and 64.
slot	string	Slot that is used for the memory (for example, NORTH_NE/ NORTH_NW).
inst_path	string	The instance path is from top module.
frequency	double	Specifies frequency in MHz.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'ddr_type' is missing.
None	Required parameter 'data_width' is missing.
None	data_width: Invalid argument value: 'data' (expecting integer value).
None	frequency: Invalid argument value: expecting decimal value.
None	DDR Debug: Valid values for "-ddr_type" parameter are DDR3, DDR4, LPDDR3. Provided value is a Error: .
None	DDR Debug: Valid values for "-ddr_width" parameter for DDR3 are 16, 32 and 64.
None	Parameter 'param_name' is not defined. Valid command formatting is 'debug_ddr [-deviceName "device name"] -ddr_type "DDR Type" -data_width "integer value" -slot "DDR Slot" -inst_path "Instace Path from Top" -frequency "decimal value" '.

Supported Families

PolarFire

PolarFire SoC

Example

This example gets the training data from the Training IP and displays the status of different stages of training along with the eye width chart:

```
debug_ddr -ddr_type {DDR4} -data_width 32 -slot {NORTH_NE} \  
-inst_path {PF_DDR4_C0_0} -frequency 800.00
```

See Also

- ddr_read
- ddr_write

5.1.8. debug_mss_ddr [\(Ask a Question\)](#)

Description

This command retrieves the training data from the memory subsystem, which executes the initialization and training sequence, and displays the status of various training stages.

```
debug_mss_ddr [-deviceName "device name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specifies the device name.

Return Type	Description
None	None

Supported Families

PolarFire SoC

Example

This example gets the training data from the memory subsystem and displays the status of different stages of training:


```
debug_mss_ddr
```

5.1.9. debug_iod [\(Ask a Question\)](#)

Description

This Tcl command gets the training data from the CORERXIODBITALIGN IP and displays Eye Width and Sampling Edge.

```
debug_iod [-deviceName "device name"] \
  -iod_type {RX_DDRX_B_G_DYN/RX_DDRX_B_R_DYN/RX_DDRX_B_G_FDYN} \
  -inst_path {PF_IOD_GENERIC_RX instance path from Top}
```

 **Important:** CORERXIODBITALIGN IP must have the output debug pins either connected or promoted to the top for SmartDebug to detect and identify the debug signals. The following pins must be configured for CORERXIODBITALIGN for IOD training to succeed and to be able to perform tasks such as reading delay taps, left taps, right taps, and bit align errors:

- RX_BIT_ALIGN_LEFT_WIN
- RX_BIT_ALIGN_RGHT_WIN
- BIT_ALGN_ERR
- DEM_BIT_ALGN_TAPDLY
- BIT_ALGN_DONE
- BIT_ALGN_START

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
iod_type	string	Specify iod type. Valid types are: RX_DDRX_B_G_DYN, RX_DDRX_B_R_DYN and RX_DDRX_B_G_FDYN.
inst_path	string	PF_IOD_GENERIC_RX instance path from Top.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'iod_type' is missing.
None	Parameter 'iod_type' has illegal value.
None	Required parameter 'inst_path' is missing.
None	Parameter 'inst_path' has illegal value.
None	IOD Debug: Provide the path of IOD instance for top level module in the design valid for "-inst_path" parameter.

Supported Families

PolarFire

PolarFire SoC

Example

Get training data from {PF_IOD_GENERIC_RX_C1_0} instance.

```
debug_iod -iod_type {RX_DDRX_B_G_DYN} -inst_path {PF_IOD_GENERIC_RX_C1_0}
```

See Also

- debug_dds

5.1.10. ddr_read [\(Ask a Question\)](#)

Description

This tcl command reads the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_read -deviceName "device name" \
        -block {DDR name} \
        -name {register name}
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
block	string	Specify block name: fddr mddr east_fddr west_fddr. <ul style="list-style-type: none"> • Specifies which DDR configurator is used in the Libero design. • SmartFusion 2 and IGLOO 2 - fddr and mddr 56. • RTG4 - east_fddr and west_fddr.
name	string	<ul style="list-style-type: none"> • Specifies which configuration registers need to be read. • A complete list of registers is available in the DDR Interfaces User Guides for the respective families.

Return Type	Description
Returns 16-bit hexadecimal value.	The result of the command in the example below will be: Register Name: DDRC_DYN_REFRESH_1_CR Value: 0x1234 "ddr_read" command succeeded.

Error Codes

Error Code	Description
None	Required parameter 'block' is missing.

ddr_read (continued)	
Error Code	Description
None	Parameter 'block' has illegal value.
None	Required parameter 'name' is missing.
None	Parameter 'name' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'ddr_read [-deviceName "device name"] -block "DDR Block Name" -name "DDR Register Name"'.

Supported Families

SmartFusion 2

IGLOO 2

RTG4

Example

Read DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion 2 or IGLOO 2 device:

```
ddr_read -block fddr -name DDRC_DYN_REFRESH_1_CR
```

See Also

- ddr_write

5.1.11. ddr_write [\(Ask a Question\)](#)

Description

This tcl command writes the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_write [-deviceName "device name"] \
          -block {ddr name} \
          -name {register name} \
          -value {hexadecimal value}
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
block	string	Specify block name: fddr mddr east_fddr west_fddr. <ul style="list-style-type: none"> • Specifies which DDR configurator is used in the Libero design. • SmartFusion 2 and IGLOO 2 - fddr and mddr • RTG4 - east_fddr and west_fddr.
name	string	<ul style="list-style-type: none"> • Specifies which configuration registers need to be read. • A complete list of registers is available in the DDR Interfaces User Guides for the respective families.
value	hexadecimal	<ul style="list-style-type: none"> • Specifies the value to be written into the specified register of a given block. • Hex_value in the form of "0x12FA".

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'value' has illegal value.
None	Required parameter 'value' is missing.
None	Parameter 'name' has illegal value.
None	Required parameter 'name' is missing.
None	Parameter 'block' has illegal value.
None	Required parameter 'block' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'ddr_write [-deviceName "device name"] -block "DDR Block Name" -name "DDR Register Name" -value "DDR register value"'.

Supported Families

SmartFusion 2

IGLOO 2

RTG4*

Example

Write a 16-bit value DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion 2 or IGLOO 2 device:

```
ddr_write -block fddr -name DDRC_DYN_REFRESH_1_CR -value 0x123f
```

See Also

- ddr_read

5.1.12. delete_active_probe [\(Ask a Question\)](#)

Description

This Tcl command deletes either all or the selected active probes.

Note:

You cannot delete an individual probe from the Probe Bus.

```
delete_active_probe -deviceName "device name" -all | -name {probe name}
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
all	None	Deletes all active probe names.
name	string	Deletes the selected probe names.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'name' has illegal value.

Supported Families

PolarFire
PolarFire SoC
SmartFusion 2
IGLOO 2
RTG4

Example

1. This example deletes all active probe names.

```
delete_active_probe -all
```

2. This example deletes the selected "out[5]:out[5]:Q" and "my_grp1.out[1]:out[1]:Q" active probe names;

```
delete_active_probe -name out[5]:out[5]:Q \  
-name my_grp1.out[1]:out[1]:Q
```

3. This example deletes the group, bus and their members.

```
delete_active_probe -name my_grp1 \  
-name my_busT
```

See Also

- select_active_probe
- create_probe_group

5.1.13. load_live_probe_list* [\(Ask a Question\)](#)

Description

This Tcl command loads the list of live probes from the file(*.txt).

```
load_live_probe_list [-deviceName "device name"] -file "filename"
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
file	string	Specify path and the name of input file(*.txt). This parameter is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'file' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'load_live_probe_list [-deviceName "device name"] \ -file "filename"'.

Supported Families

PolarFire
PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example loads M3T device live probes list from live_probe_list.txt file. Text file which has the probes list saved from previous SmartDebug save action.

```
save_live_probe_list -file {./live_probe_list.txt} load_live_probe_list  
-deviceName {M3T} -file {./live_probe_list.txt}
```

See Also

- save_live_probe_list

5.1.14. eye_monitor_power [\(Ask a Question\)](#)

Description

This tcl command switches on and off power eye monitor.

```
eye_monitor_power [-deviceName "device name"] \  
-switch {on | off} \  
-lane {physical lane name}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
switch	string	This argument specifies if the eye monitor is on or off.
lane	string	Specify the physical lane instance name.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Eye Monitor Power On/Off: Lane name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Parameter 'lane' has illegal value.
None	Parameter 'switch' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'eye_monitor_power [-deviceName "device name"] [-switch "Eye Monitor Power"] [-lane "Physical Lane Name"]'.

Supported Families

PolarFire

PolarFire SoC

Example

This example turns on the eye monitor.

```
eye_monitor_power -switch {on} -lane {Q0_LANE0}
```

This example turns off the eye monitor.

```
eye_monitor_power -switch {off} -lane {Q0_LANE0}
```

5.1.15. event_counter [\(Ask a Question\)](#)

Description

This Tcl command runs on signals that are assigned to Channel A through the Live Probe feature and displays the total events.

It is run after setting the live probe signal to channel A. The user specifies the duration to run the event_counter command.

```
event_counter -run | -stop -after {duration in seconds}
```

Arguments

Parameter	Type	Description
run	none	Run event counter.
stop	none	Stop event counter. This parameter must be specified with the -after parameter.
after	integer	Specify duration in seconds to stop event_counter. This argument is required when -stop argument is specified.

Return Type	Description
string	Displays the total events with value-property format.

Error Codes

Error Code	Description
None	Missing argument. Must specify '-run' or '-stop'.
None	Must specify time by using the argument '-after'.
None	after: Invalid argument value: 'value' (expecting integer value).
None	No signal assigned to channel A.
None	Parameter 'param_name' is not defined. Valid command formatting is 'event_counter [-deviceName "device name"] [-run "TRUE FALSE"] [-stop "TRUE FALSE"] [-after "integer value"]'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example assigns 'Q_c:DFN1_0:Q' signal to Channel A, runs event counter with the 5 delay seconds to stop:

```
set_live_probe -probeA {Q_c:DFN1_0:Q}
event_counter -run
event_counter -stop -after 5
```

See Also

- fhb_control
- run_frequency_monitor
- set_live_probe

5.1.16. execute_dfe_calibration [\(Ask a Question\)](#)

Description

This Tcl command executes calibration. There are two types of calibration DFE (decision feedback equalizer) and CTLE (continuous time linear equalizer).

```
execute_dfe_calibration [-deviceName "device name"] \
    -lane {physical location name} \
    -full_calibration {0 | 1}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specify the physical lane instance name.
full_calibration	boolean	This parameter specifies what kind of calibration you want to execute. <ul style="list-style-type: none"> • 1 - execute full calibration both DFE and CTLE Calibrations. • 0 - execute DFE Calibration.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Execute DFE Calibration: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name
None	Parameter 'lane' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'execute_dfe_calibration [-deviceName "device name"] [-lane "Physical Lane Name"] [-full_calibration "TRUE FALSE"]'.

Supported Families

PolarFire

PolarFire SoC

Example

This example executes the dfe calibration for lane "Q0_Lane0".

```
execute_dfe_calibration -lane {Q0_Lane0} -full_calibration 1
```

5.1.17. **export_dds_training_data** [\(Ask a Question\)](#)

Description

This Tcl command exports the training data that is read from the device into a simple text file, which helps users to compare data between multiple runs by exporting the training data.

```
export_dds_training_data [-file "file name"]
```

Arguments

Parameter	Type	Description
fileName	string	File name where the exported data will be saved.

Supported Families

PolarFire

PolarFire SoC

Example

This example exports the DDR training data to D:\exportedData.txt.

```
export_dds_training_data -file D:\exportedData.txt
```

5.1.18. **export_mss_dds_training_data** [\(Ask a Question\)](#)

Description

This Tcl command exports the training data read from device to the specified file.

```
export_mss_dds_training_data [-file "file name"]
```

Arguments

Parameter	Type	Description
file	string	Specifies the file name with file path to export the MSS DDR I/O margin training results.

Supported Families

PolarFire SoC

Example

This example exports the MSS DDR training data to D:\exportedData.txt.

```
export_mss_dds_training_data -file D:/exportedData.txt
```

5.1.19. **fhb_control** [\(Ask a Question\)](#)

Description

This Tcl command provides FPGA Hardware Breakpoint (FHB) capability for SmartDebug.

```
fhb_control [-deviceName "device name"] -halt \
    -clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] -run \
    -clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] -step {number of steps} \
    -clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] -reset \
```

```
-clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] -arm_trigger \
-trigger_signal {probe point signal to trigger the HALT operation} \
-trigger_edge_select {rising} -clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] \
[-capture_waveform "integer value"] \
[-vcd_file "Waveform File"]

fhb_control [-deviceName "device name"] \
[-clock_domain_status] \
-clock_domain {clock domain name(s)/all}

fhb_control [-deviceName "device name"] \
[-capture_waveform "integer value"] \
[-vcd_file "Waveform File"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
clock_domain	string	Specifies clock domain names to halt run step reset disarm . Can be single or multiple clock domains, halted in order specified by user.
halt	none	Specifies to halt the clock.
run	none	Specifies to run the clock.
step	integer	Specifies to step the clock "number of steps" times. Minimum value is 1.
reset	none	Specifies to reset FHB configuration for the specified clock domain.
arm	none	Specifies to arm FHB configuration for the specified clock domain.
trigger_signal	string	probe point signal to trigger the HALT operation} Set the trigger signal to arm the FHBs.
trigger_edge_select	string	Specifies the trigger signal edge to arm the FHBs. FHBs will be armed on rising edge of trigger signal.
delay	integer	Specifies the value between 0 to 255 of delay cycles before halt.
clock_domain_status	none	Specifies to read and display status of specified clock domain(s). Can be single or multiple clock domains.
disarm	none	Specifies to disarm FHB configuration for the specified clock domain.
capture_waveform	integer	Specifies to capture waveform of all the added signals to active probes in the specified clock domain for "number of steps".
vcd_file	string	Target file to save the data and see the waveform.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Fhb control: One of the following parameters should be set:-halt, -run, -step, -arm, -disarm, -clock_domain_status, -reset and -capture_waveform
None	Fhb control: Clock Domain all not found in the design.
None	step: Invalid argument value: 'step_value' (expecting integer value).
None	Fhb control: Minimum value of -step should be 1.
None	Fhb control: -trigger_edge_select parameter value can only be rising.

fhb_control (continued)

Error Code	Description
None	capture_waveform: Invalid argument value: 'capture_waveform value' (expecting integer value).
None	Fhb control: Minimum value of -capture_waveform should be 1.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

```
fhb_control -halt -clock_domain {"FCCC_0/GL0_INST" "FCCC_0/GL1_INST" }
fhb_control -run -clock_domain {"FCCC_0/GL0_INST" "FCCC_0/GL1_INST" }
fhb_control -step -clock_domain {"FCCC_0/GL0_INST" "FCCC_0/GL1_INST" }
fhb_control -reset -clock_domain {"FCCC_0/GL0_INST" "FCCC_0/GL1_INST" }
fhb_control -arm_trigger -trigger_signal {q_0_c[14]:count_1_q[14]:Q} \
    -trigger_edge_select {rising} -delay 0 \
    -clock_domain {"FCCC_0/GL0_INST"}
fhb_control -disarm_trigger -trigger_signal {q_0_c[14]:count_1_q[14]:Q} \
    -trigger_edge_select {rising} -delay 0 \
    -clock_domain {"FCCC_0/GL0_INST"}
fhb_control -capture_waveform {10} \
    -vcd_file {D:/wvf_location/waveform.vcd}
fhb_control -clock_domain_status \
    -clock_domain { "FCCC_0/GL0_INST" "FCCC_0/GL1_INST" "FCCC_0/GL2_INST" }
```

See Also

- event_counter
- run_frequency_monitor

5.1.20. get_programmer_info ([Ask a Question](#))**Description**

This Tcl command lists the IDs of all FlashPro programmers connected to the computer. Command will fail if programmers are not connected.

```
get_programmer_info
```

Arguments

Parameter	Type	Description
None	None	None

Return Type	Description
List of IDs	Returns the list of IDs of all FlashPro programmers connected to the computer.

Error Codes

Error Code	Description
None	None

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

RTG4

Example

Get the list of all connected programmers IDs:

```
set IDs [get_programmer_info];
puts "IDs of connected programmers: $IDs";
```

See Also

- read_device_status
- read_id_code

5.1.21. get_user_clock_frequencies [\(Ask a Question\)](#)**Description**

This Tcl command calculates the user clock frequencies.

Note:

Before this commands usage user has to enable FHB(FPGA Hardware Breakpoint) controller from Libero project settings.

```
get_user_clock_frequencies [-deviceName "device name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.

Return Type	Description
String	Displays user clock frequency in megahertz (MHz).

Error Codes

Error Code	Description
None	Fhb control: The design does not have FHB enabled.
None	Parameter 'param_name' is not defined. Valid command formatting is. 'get_user_clock_frequencies [-deviceName "device name"] '.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

Get 'M4' device user clock frequensy:

```
get_user_clock_frequencies -deviceName "M4"
```

Output:

User clock frequency - clocking_0VPPF_out0 value = 100.07 MHz

User clock frequency - clocking_0VPPF_out1 value = 132.02 MHz

See Also

- event_counter
- run_frequency_monitor

5.1.22. import_ddc_file [\(Ask a Question\)](#)

Description

This is a Standalone SmartDebug command. Enables you to import DDC file (created through Export SmartDebug Data in Libero) into the debug project.

```
import_ddc_file -import_ddc "DDC file" -device_name "device name"
```

Arguments

Parameter	Type	Description
import_ddc	string	Specify path to the DDC file. This parameter is mandatory.
device_name	string	Specify the device name. This parameter is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'import_ddc' is missing.
None	Required parameter 'device_name' is missing.
None	Failed to import DDC file '*.ddc'. There is no device 'device_name' in the current JTAG chain.
None	Parameter 'param_name' is not defined. Valid command formatting is 'import_ddc_file -import_ddc "DDC file" -device_name "device name"'

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

RTG4

Example

This example importes DDC file to the debug project:

```
import_ddc_file -import_ddc {./src/top.ddc} -device_name {MPF250T_ES}
```

See Also

- new_project

5.1.23. load_active_probe_list [\(Ask a Question\)](#)

Description

This Tcl command loads the list of probes from the file.

```
load_active_probe_list [-deviceName "device name"] \  
-file "path to the file"
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
file	string	The input file location.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Required parameter 'file' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'load_active_probe_list [-deviceName "device name"] -file "filename"'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example loads the active probe list from “./my_probes.txt” file.

```
load_active_probe_list -file "./my_probes.txt"
```

See Also

- delete_active_probe
- read_active_probe
- save_active_probe_list
- select_active_probe
- write_active_probe

5.1.24. load_SI_design_defaults [\(Ask a Question\)](#)

Description

This Tcl command loads the Signal Integrity parameter options for the selected lane instance.

```
load_SI_design_defaults [-deviceName "device name"] \  
[-lane "Lane Instance Name"] \  
[-all_lanes "TRUE | FALSE"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specify the physical lane instance name.
all_lanes	boolean	If you want to load design defaults for all lanes, then give "TRUE" to the argument, else "FALSE".

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Signal Integrity: Must not specify both '-lane' and '-all_lanes' command arguments.
None	Signal Integrity: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Parameter 'lane' has illegal value.
None	Signal Integrity: Must specify one of '-lane' or '-all_lanes' command arguments.
None	Parameter 'param_name' is not defined. Valid command formatting is 'load_SI_design_defaults [-deviceName "device name"] [-lane "Lane Instance Name"] [-all_lanes "TRUE FALSE"]'.

Supported Families

PolarFire

PoarFire SoC

Example

This example loads design defaults for lane "Q0_LANE0"

```
load_SI_design_defaults -lane {Q0_LANE0}
```

See Also

- signal_integrity_write
- signal_integrity_import
- signal_integrity_export

5.1.25. loopback_mode [\(Ask a Question\)](#)

Description

This Tcl command applies loopback mode to a specified lane.

```
loopback_mode -lane {Physical Lane name} -apply -type {loopback_type}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specify the physical location of the lane.

loopback_mode (continued)

Parameter	Type	Description
apply	none	Apply specified loopback to specified lane.
type	string	Specify the loopback type to apply. Type valid values are: EQ-NearEnd, EQ-FarEnd, CDRFarEnd and NoLpbk.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Loopback Mode: Must specify '-apply' argument.
None	Loopback Mode: Transceiver physical Lane Name must be specified.
None	Parameter 'type' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'loopback_mode [-deviceName "device name"] [-lane "Physical Lane Name"] [-apply "TRUE FALSE"] [-type "Loopback type"]'.

Supported Families

PolarFire

PolarFire SoC

Example

This examples applies EQ-FarEnd | EQ-NearEnd | CDRFarEnd | NoLpbk loopback mode to a "Q0_LANE0" lane.

```
loopback_mode -lane {Q0_LANE0} -apply -type {EQ-FarEnd}
loopback_mode -lane {Q0_LANE0} -apply -type {EQ-NearEnd}
loopback_mode -lane {Q0_LANE0} -apply -type {CDRFarEnd}
loopback_mode -lane {Q0_LANE0} -apply -type {NoLpbk}
```

See Also

- [loopback_test](#)
- [smartbert_test](#)
- [prbs_test](#)

5.1.26. loopback_test [\(Ask a Question\)](#)

Description

This Tcl command used to start and stop the loopback tests. Loopback data stream patterns are generated and checked by the internal SerDes block. These are used to self-test signal integrity of the device. You can switch the device through predefined tests.

Note:

loopback_test is renamed as loopback_mode in G5.

```
loopback_test [-deviceName "device name"] [-start] \
    -serdes "integer value" -lane "integer value" \
    -type "Loopback Type"
loopback_test [-deviceName "device name"] [-stop] \
    -serdes "integer value" -lane "integer value"
```

Arguments

Parameter	Type	Description
deviceName	string	Specifies device name. This parameter is optional if only one device is available in the current configuration or set for debug.
start	none	Starts the loopback test.
stop	none	Stops the loopback test.
SerDes	integer	Specifies SerDes block number. Must be between 0 and 4 and varies between dies.
lane	integer	Specifies SerDes lane number. Must be between 0 and 3.
type	string	Specifies the loopback test type. Loopback test types are: Must be meso (PCS Far End PMA RX to TX Loopback), plesio and parallel.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'loopback_test [-deviceName "device name"] [-start "TRUE FALSE"] [-stop "TRUE FALSE"] -serdes "integer value" -lane "integer value" [-type "Loopback type"]'.
None	Required parameter 'serdes' is missing.
None	Required parameter 'lane' is missing.
None	serdes: Invalid argument value: 'serdes_value' (expecting integer value).
None	lane: Invalid argument value: 'lane_value' (expecting integer value).
None	Loopback test: IDCode verify failed.
None	Loopback test: Invalid loopback type specified.

Supported Families

IGLOO 2

RTG4

Example

Start and stop loopback tests.

```
loopback_test -start -serdes 1 -lane 1 -type meso
loopback_test -start -serdes 0 -lane 0 -type plesio
loopback_test -start -serdes 1 -lane 2 -type parallel
loopback_test -stop -serdes 1 -lane 2
```

See Also

- loopback_mode
- prbs_test
- smartbert_test

5.1.27. move_to_probe_group [\(Ask a Question\)](#)

Description

This Tcl command moves the specified probe points to the specified probe group.

Note:

Probe points related to a bus cannot be moved to another group.

```
move_to_probe_group -name {probe name} -group {group name}
```

Arguments

Parameter	Type	Description
name	string	Specifies one or more probes to move.
group	string	Specifies name of the probe group.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'group' has illegal value.
None	Required parameter 'group' is missing.
None	Parameter 'name' has illegal value.
None	Required parameter 'name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'move_to_probe_group [-name "name"]+ \-group "group name"'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example moves {out[5]:out[5]:Q} and {grp1.out[3]:out[3]:Q} probes to the {my_grp2}:

```
move_to_probe_group -name {out[5]:out[5]:Q} \
                    -name {grp1.out[3]:out[3]:Q} \
                    -group {my_grp2}
```

See Also

- create_probe_group
- add_to_probe_group

5.1.28. mss_add_register [\(Ask a Question\)](#)

Description

This tcl command records whenever registers are selected in the tool.

```
mss_add_register -reg_name {register name} -reset {0|1}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
reg_name	string	Specifies name of the register.
reset	boolean	When set to 1, all the previously selected registers will be cleared from the list and the new ones will be added. If 0, then adds the register to the old list.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Enter register names and reset value. Params are "reg_name" and "reset".
None	Parameter 'reg_name' has illegal value.
None	register is not found in the valid list provided in pfsoc_regmap.htm file.
None	Parameter 'reset' has illegal value.
None	reset: Invalid argument value: " (expecting TRUE, 1, true, FALSE, 0 or false).
None	Parameter 'param_name' is not defined. Valid command formatting is 'mss_add_register [-deviceName "device name"] \ [-reg_name "Add Register Names"]* \ [-reset "TRUE FALSE"]'.

Supported Families

PolarFire SoC

Example

This example adds the following registers into mss register list.

```
mss_add_register \
-reg_name {ATHENA:CSRMAIN} \
-reg_name {ATHENA:CSRMERRS} \
-reg_name {ATHENA:CSRMERRT0} \
-reg_name {ATHENA:CSRMERRT1} \
-reg_name {ATHENA:CSRMERRV} \
-reset 0
```

See Also

- mss_read_register
- mss_write_register
- mss_export_register

5.1.29. mss_read_register [\(Ask a Question\)](#)

Description

This tcl command reads all selected registers by their register names and displays the values. The values are in hexadecimal format.

```
mss_read_register [-deviceName "device name"] \
[-reg_name {register name}] \
[-axiQos "integer value"] \
[-axiProt "integer value"] \
[-axiCache "integer value"] \
```

```
[-axiLock "integer value"] \
[-silent "TRUE | FALSE"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
reg_name	string	This is an optional argument, that specifies the name of the register according to the hierarchy seen in the UI separated by colon.
axiQos	integer	This is an optional parameter that specifies the value of the attribute QoS on Axi interface.
axiCache	integer	This is an optional parameter that specifies the value of the attribute Cache on Axi interface.
axiProt	integer	This is an optional parameter that specifies the value of the attribute Protocol on Axi interface.
axiLock	integer	This is an optional parameter that specifies the value of the attribute Lock on Axi interface.

Return Type	Description
Hexadecimal	Reads all selected registers by their register names and displays the values.

Error Codes

Error Code	Description
None	Register is not found in the valid list provided in pfsoc_regmap.htm file.
None	Parameter 'reg_name' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'mss_read_register [-deviceName "device name"] \ [-reg_name "Register Name"]* \ [-axiQos "integer value"] \ [-axiProt "integer value"] \ [-axiCache "integer value"] \ [-axiLock "integer value"] \ [-silent "TRUE FALSE"]'.

Supported Families

PolarFire SoC

Example

Read register with parameters. Read with {ATHENA:CSRMAIN} and {ATHENA:CSRMERRS} registers:

```
mss_read_register \
  -reg_name {ATHENA:CSRMAIN} \
  -reg_name {ATHENA:CSRMERRS}
```

Register read without parameters (reads all the selected registers that are added using mss_add_register command):

```
mss_read_register
```

See Also

- mss_write_register
- mss_export_register
- mss_add_register

5.1.30. mss_write_register [\(Ask a Question\)](#)

Description

This tcl command writes value to the selected registers.

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.

```
mss_write_register [-deviceName "device name"] \[-reg_name "Register Name"]* \[-value "Register write value"]* \[-axiQos "integer value"] \[-axiProt "integer value"] \[-axiCache "integer value"] \[-axiLock "integer value"] \[-silent "TRUE | FALSE"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
reg_name	string	Name of the register according to the hierarchy seen in the UI separated by colon. It can also contain register field separated by colon.
value	hexadecimal	Hexadecimal value - ranges from 1-bit to 64-bits.
axiQos	integer	This is an optional parameter that specifies the value of the attribute QoS on Axi interface.
axiCache	integer	This is an optional parameter that specifies the value of the attribute Cache on Axi interface.
axiProt	integer	This is an optional parameter that specifies the value of the attribute Protocol on Axi interface.
axiLock	integer	This is an optional parameter that specifies the value of the attribute Lock on Axi interface.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Can not write into a read-only register or a field.
None	Invalid register name specified.
None	Parameter 'reg_name' has illegal value.
None	register is not found in the valid list provided in pfsoc_regmap.htm file.
None	Invalid register value specified.
None	Parameter 'value' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'mss_write_register [-deviceName "device name"] \[-reg_name "Register Name"]* \[-value "Register write value"]* \[-axiQos "integer value"] \[-axiProt "integer value"] \[-axiCache "integer value"] \[-axiLock "integer value"] \[-silent "TRUE FALSE"]'

Supported Families

PolarFire SoC

Example

This example writes the values into the registers.

```
mss_write_register \
  -reg_name {MMUART0_LO:RBR} -value {0x123} \
  -reg_name {MMUART0_LO:IER} -value {0xFFFFFFFF} \
  -reg_name {MMUART0_LO:IIR:IIR} -value {0x3}
```

See Also

- [mss_read_register](#)
- [mss_export_register](#)
- [mss_add_register](#)

5.1.31. mss_import_register [\(Ask a Question\)](#)

Description

This Tcl command imports the register list from a *.csv file generated by register export operation.

```
mss_import_register \
  -file_name {absolute or relative path to the *.csv file} \
  [-deviceName "device name"]
```

Arguments

Parameter	Type	Description
file_name	string	Specifies the absolute and relative path to the *.csv file.
deviceName	string	Specify the device name. This parameter is optional, if only one device is available in the current configuration.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Register Access: Must specify '-file_name'.
None	Required parameter 'file_name' is missing.
None	Parameter 'file_name' has illegal value.
None	Register Access: file specified for import must have .csv extension.
None	Unable to write to the file: /prj_path/imported_file.csv
None	Parameter 'param_name' is not defined. Valid command formatting is 'mss_import_register [-deviceName "device name"] -file_name "filename" '.

Supported Families

PolarFire SoC

Example

This example imports the register list from the { ./MssRegisters_SmartDebug.csv } file.

```
mss_import_register -file_name {./MssRegisters_SmartDebug.csv}
```

See Also

- [mss_add_register](#)

- [mss_read_register](#)
- [mss_write_register](#)

5.1.32. **mss_export_register** [\(Ask a Question\)](#)

Description

This command reads the selected registers by `mss_add_register` command and save the content to a *.csv file.

```
mss_export_register -file_name {path the file} [-all]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
file_name	string	Specifies the path to the file, where the command saves the exported data.
all	none	This is an optional parameter that is used to export all registers shown in the hierarchy.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Register Access: Must specify '-file_name'.
None	Parameter 'file_name' has illegal value.
None	Register Access: file specified for Export must have .csv extension.
None	Cannot export register information to the file: List of selected registers is empty.
None	Parameter 'param_name' is not defined. Valid command formatting is 'mss_export_register [-deviceName "device name"] [-file_name "File Name"] [-all "TRUE FALSE"]'.

Supported Families

PolarFire SoC

Example

This example exports all mss registers into {./mss_exp.csv} file.

```
mss_export_register -file_name {./mss_exp.csv} -all
```

See Also

- [mss_add_register](#)
- [mss_read_register](#)
- [mss_write_register](#)

5.1.33. new_project [\(Ask a Question\)](#)

Description

This Tcl command creates a SmartDebug project that enables the user to debug the design. Either DDC can be used to create a project or construct automatically with DDC.

```
new_project [-location {project location}] \
  [-name {name of the new SmarDebug project}] \
  [-import_ddc {path to the DDC file}] \
  [-auto_construct {"TRUE"|"FALSE"}] \
  [-set_programmer {set debug programmer}]
```

Arguments

Parameter	Type	Description
location	string	Specify the location of the project where user wants to create the project. Must not be an existing directory. This parameter is optional. If -location option is omitted, the tool creates a new project in the current directory.
name	string	Specify name of the new project. This parameter is optional. This parameter is optional. If this option is omitted, the tool creates a new project with 'untitled_project' name.
import_ddc	string	Specify the path to the DDC(Design Debug Data Container) file exported from Libero to be imported. Set empty parameter value if -auto_construct is 1.
auto_construct	boolean	Valid values are:TRUE or 1, FALSE or 0(default). Specify 1 or TRUE if you want to create new project importing DDC file otherwise specify 0 or FALSE. This parameter is optional.
set_programmer	string	Set ID code of the programmer. This parameter is optional.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'new_project [-location "project folder"] [-name "name"] [-import_ddc "DDC file"] [-auto_construct "TRUE FALSE"] [-set_programmer "set debug programmer"].

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

RTG4

Example

Create new project using Standalone SmartDebug:

```
new_project -location {/exprj} \
  -name {exprj} \
  -import_ddc {./src/top.ddc} \
  -auto_construct 0 \
  -set_programmer {AF01QVEAF}
```

See Also

- `import_ddc_file`

5.1.34. open_project ([Ask a Question](#))**Description**

This Tcl command opens an existing SmartDebug project (*.dprj).

```
open_project -project {relative or absolute path and name of the project file}
```

Arguments

Parameter	Type	Description
project	string	Specify relative or absolute path to the *.dprj file. This parameter is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'project' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'open_project -project "file"'

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

RTG4

Example

This command opens the 'SDPrj.dprj' project from the SDProject directory:

```
open_project -project {./SDProject/SDPrj.dprj}
```

See Also

- `new_project`

5.1.35. optimize_dfe ([Ask a Question](#))**Description**

This Tcl command supports the Optimize DFE (decision feedback equalizer) feature in SmartDebug.

```
optimize_dfe [-deviceName "device name"] \  
             -dfe_algorithm {type of dfe algorithm} \  
             -lane {lane(s) configured in the design}
```

Arguments

Parameter	Type	Description
deviceName	script	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.

optimize_dfe (continued)

Parameter	Type	Description
dfe_algorithm	script	This command executes Dfe Algorithm with type of dfe algorithm and lanes as input. Algorithm selection has two options: software_based -executes DfeSs.tcl script xcvr_based -executes internal Dfe Auto Calibration. This argument is mandatory.
lane	script	List of lane(s) configured in the design. This argument is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'optimize_dfe [-deviceName "device name"] -lane "[Physical Lane Name]+" -dfe_algorithm "Dfe Algorithm Selection".
None	Parameter 'deviceName' has illegal value.
None	Parameter 'lane' has illegal value.
None	Required parameter 'lane' is missing.
None	Required parameter 'dfe_algorithm' is missing.
None	Parameter 'dfe_algorithm' has illegal value.
None	Optimize DFE: dfe_algorithm has invalid option. Possible options: software_based, xcvr_based.
None	Execute DFE Calibration: Execute DFE calibration failed.
None	Optimize DFE: Transceiver Physical Lanes Q1_LANE0 are configured in CDR Mode.XCVR_BASED Dfe Algorithm is valid for DFE configured lanes only.

Supported Families

PolarFire

PolarFire SoC*

Example

This example optimizes dfe for lane "Q2_LANE0" using software_based algorithm.

```
optimize_dfe -lane {"Q2_LANE0"} -dfe_algorithm {software_based}
```

This example optimizes dfe for lane "Q2_LANE0" using xcvr_based algorithm.


```
optimize_dfe -lane {"Q2_LANE0"} -dfe_algorithm {xcvr_based}
```

This example optimizes dfe for lane "Q2_LANE0" and "Q0_LANE0" using xcvr_based algorithm.

```
optimize_dfe -lane {"Q2_LANE0" "Q0_LANE0"} -dfe_algorithm {xcvr_based}
```

5.1.36. optimize_receiver ([Ask a Question](#))**Description**

This Tcl command allows you to optimize the DFE (decision feedback equalizer) coefficients and/or CTLE (continuous time linear equalizer) settings for the selected lanes, depending on the receiver mode. For CDR mode receivers, the CTLE settings and/or DFE coefficients can be optimized. For DFE mode receivers, the CTLE settings and DFE coefficients can be optimized.

 **Important:** This feature is available for MPF300T, MPF100T, MPF200T, MPF500T, MPFS250T, MPFS025T, and MPFS095T devices.

```
optimize_receiver [-deviceName "device name"] -lane {physical lane name} [-force_dfe_calibration {1/0/TRUE/FALSE}]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration or a device has already been selected using the set_debug_device command.
lane	string	Specifies the physical lane instance name.
force_dfe_calibration	boolean	Optional parameter to run DFE calibration on CDR mode receivers. When set to false, it disables running DFE calibration and DFE coefficients are reset.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Optimize RECEIVER: The DFE Calibration process on lane Q1_LANE0 failed and the Calibration process timed out.
None	Optimize RECEIVER: Transceiver Physical Lane LANE_NAME not found in the design
None	Required parameter 'lane' is missing.
None	Parameter 'lane' is missing or has an invalid value.
None	Parameter 'lane' has an illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'optimize_receiver [-deviceName "device name"] -lane "[Physical Lane Name]+"'.

Supported Families

PolarFire

PolarFire SoC

Example

This example optimizes receiver for the "Q0_LANE0" lane:

```
optimize_receiver -lane {Q0_LANE0} - force_dfe_calibration 1
```

See Also

- optimize_dfe

5.1.37. pcie_config_space [\(Ask a Question\)](#)

Description

This Tcl command displays the value of the entered parameter in the SmartDebug log window and returns the register:field value to the Tcl.

```
pcie_config_space -pcie_logical_name {Pcie Logical Name} \
  -paramNameList {Pcie Parameter Name}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
pcie_logical_name	string	Complete logical hierarchy of the PCIe block whose status is to be read from the device. This parameter is mandatory.
paramNameList	string	Parameter name to read from the device. This parameter is optional.

Return Type	Description
string	Returns register:field value.

Error Codes

Error Code	Description
None	Parameter 'arg_name' is not defined. Valid command formatting is 'pcie_config_space [-deviceName "device name"] -pcie_logical_name "Pcie Logical Name" [-paramNameList "[Pcie Parameter Name]+"] [-allparams "TRUE FALSE"]'.
None	Required parameter 'pcie_logical_name' is missing.
None	Parameter 'pcie_block_name' is not defined. Valid command formatting is 'pcie_config_space [-deviceName "device name"] -pcie_logical_name "Pcie Logical Name" [-paramNameList "[Pcie Parameter Name]+"] [-allparams "TRUE FALSE"]'.
None	Parameter 'param_name' is not defined. Valid command formatting is 'pcie_config_space [-deviceName "device name"] -pcie_logical_name "Pcie Logical Name" [-paramNameList "[Pcie Parameter Name]+"] [-allparams "TRUE FALSE"]'.

Supported Families

PolarFire

PolarFire SoC

Example

Output Display in SmartDebug window: 512 bytes

Return value to the tcl script: 0x2

```
pcie_config_space -pcie_block_name {sb_0/CM1_Subsystem/my_pcie_0} \
    -param_name {neg_max_payload}
```

See Also

- pcie_ltssm_status

5.1.38. pcie_ltssm_status [\(Ask a Question\)](#)

Description

This Tcl command displays the current LTSSM state from the PLDA core in the SmartDebug log window and returns the register:field value to the Tcl.

```
pcie_ltssm_status -pcie_logical_name {PCIe logical name}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

pcie_ltssm_status (continued)

Parameter	Type	Description
pcie_logical_name	string	Complete logical hierarchy of the PCIe block whose status is to be read from the device. This parameter is mandatory.

Return Type	Description
string	Returns register:field value.

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'pcie_ltssm_status [-deviceName "device name"] -pcie_logical_name "Pcie Logical Name"'. None
None	Required parameter 'pcie_logical_name' is missing.
None	Parameter 'pcie_block_name' is not defined. Valid command formatting is 'pcie_ltssm_status [-deviceName "device name"] -pcie_logical_name "Pcie Logical Name"'. None

Supported Families

PolarFire

PolarFire SoC

Example

Output Display in SmartDebug window: Configuration.Linkwidth.start Return value to the tcl script:

```
pcie_ltssm_status -pcie_block_name {sb_0/CM1_Subsystem/my_pcie_0}
```

See Also

- pcie_config_space

5.1.39. plot_eye [\(Ask a Question\)](#)

Description

This Tcl command is used to plot eye and export eye plots.

```
plot_eye [-deviceName "device name"] \
        -lane "Physical Lane Name" \
        [-file "filename"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specify the lane instance name.
file	string	Specify the path to the location where the file is to be exported.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'lane' is missing.
None	Parameter 'file' has illegal value.
None	Plot Eye: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Parameter 'lane' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'plot_eye [-deviceName "device name"] -lane "Physical Lane Name" [-file "filename"] .

Supported Families

PolarFire

PolarFire SoC

Example

This example plots eye for lane {Q2_LANE0} and saves it into {./export.txt} file.

```
plot_eye -lane {Q2_LANE0} -file {./export.txt}
```

5.1.40. prbs_test [\(Ask a Question\)](#)

Description

This Tcl command used in PRBS test to start, stop, reset the error counter and read the error counter value. PRBS data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through several predefined patterns.

Note:

prbs_test is renamed as smartbert_test in G5.

```
prbs_test [-deviceName device_name ] -start -serdes "integer value" \
-lane "integer value" [-near] -pattern "PatternType"
prbs_test [-deviceName device_name ] -stop -serdes "integer value" \
-lane "integer value"
prbs_test [-deviceName device_name ] -reset_counter \
-serdes "integer value" -lane "integer value"
prbs_test [-deviceName device_name ] -read_counter \
-serdes "integer value" -lane "integer value"
```

Arguments

Parameter	Type	Description
deviceName	string	Specifies device name. This parameter is optional if only one device is available in the current configuration or set for debug.
start	none	Starts the prbs test.
stop	none	Stops the prbs test.
reset_counter	none	Resets the prbs error count value to 0.
read_counter	none	Reads and prints the error count value.
SerDes	integer	SerDes block number. Must be between 0 and 4 and varies between dies.
lane	integer	SerDes lane number. Must be between 0 and 4.
near	none	Corresponds to near-end (on-die) option for prbs test. Not specifying implies off-die.

prbs_test (continued)

Parameter	Type	Description
pattern	string	The pattern sequence to use for PRBS test. It can be one of the following: prbs7, prbs11, prbs23, or prbs31.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'prbs_test [-deviceName "device name"] [-start "TRUE FALSE"] [-stop "TRUE FALSE"] [-reset_counter "TRUE FALSE"] [-read_counter "TRUE FALSE"] [-pattern "Pattern type"] -serdes "integer value" -lane "integer value" [-near "TRUE FALSE"]
None	Required parameter 'serdes' is missing.
None	serdes: Invalid argument value: 'serdes_value' (expecting integer value).
None	Required parameter 'lane' is missing.

Supported Families

SmartFusion 2

IGLOO 2

RTG4

Example

The following example starts PRBS test with the "prbs11" pattern:

```
prbs_test -start -serdes 1 -lane 0 -near -pattern "prbs11"
```

See Also

- smartbert_test
- loopback_mode
- loopback_test

5.1.41. process_job_request [\(Ask a Question\)](#)

Description

Processes a job request received from Job Manager. It is part of the Job Ticket generation process.

Note: The HSM server name must be specified using the HSM_SERVER_NAME DEF variable.

```
process_job_request -request_file "job request file" -reply_file "job reply file" [-  
  overwrite_reply "TRUE | FALSE"]
```

Arguments

Parameter	Type	Description
request_file	string	Specifies the full file name of the job request file. This argument is mandatory.
reply_file	string	Specifies the full job name of the job reply file.

process_job_request (continued)

Parameter	Type	Description
overwrite_reply	string	Optional parameter. <ul style="list-style-type: none"> 1: overwrites any pre-existing reply_file. 0: prevents overwriting of any pre-existing reply_file.

Error Codes

Error Code	Description
None	Required parameter "request_file" is missing.
None	None Required parameter "reply_file" is missing.
None	Parameter "request_file" : the file "/request_file_name" does not exist.
None	overwrite_reply: Invalid argument value: "value" (expecting TRUE, 1, true, FALSE, 0 or false).
None	Cannot get HSM.
None	Parameter "param_name" is not defined. Valid command formatting is "process_job_request -request_file "job request file" \ -reply_file "job reply file" \ [-overwrite_reply "TRUE FALSE"]".

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

Example

This example processes a job request.

```
process_job_request \
    -request_file {D:/flashpro_files/jobmgr_project/cm_request.req} \
    -reply_file {D:/flashpro_files/jobmgr_project/cm_reply.rep} \
    -overwrite_reply {TRUE}
```

5.1.42. program_probe_insertion [\(Ask a Question\)](#)

Description

This Tcl command runs the probe insertion flow on the selected nets. 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.

Note:

Probe Insertion feature disabled in the SmartDebug Demo and Standalone modes.

```
program_probe_insertion
```

Arguments

Parameter	Type	Description
None	None	None

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Probe insertion operations are not supported in Standalone SmartDebug.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example runs the probe insertion flow:

```
program_probe_insertion
```

See Also

- `add_probe_insertion_point`
- `remove_probe_insertion_point`

5.1.43. `read_active_probe` [\(Ask a Question\)](#)

Description

This Tcl command reads active probe values from the device. The target probe points are selected by the `select_active_probe` command.

Note:

When the user tries to read at least one signal from the bus/group, the complete bus or group is read. The user is presented with the latest value for all the signals in the bus/group.

```
read_active_probe [-deviceName device_name ] \
                  [-name probe_name ] \
                  [-group_name bus_name | group_name ] \
                  [-value_type b|h] \
                  [-file file_path ]
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
name	string	Instead of all probes, read only the probes specified. The probe name should be prefixed with bus or group name if the probe is in the bus or group.
group_name	string	Instead of all probes, reads only the specified buses or groups specified here.
value_type	string	Optional parameter, used when the read value is stored into a variable as a string. b = binary h = hex
file	string	Optional. If specified, redirects output with probe point values read from the device to the specified file.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Parameter 'value_type' has illegal value.
None	Parameter 'name' has illegal value.
None	Parameter 'group_name' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example reads active probes of {group1}.

```
read_active_probe -group_name {group1}
```

See Also

- `select_active_probe`
- `write_active_probe`
- `delete_active_probe`

5.1.44. `read_envm_memory` [\(Ask a Question\)](#)

Description

This is a PolarFire SoC specific tcl command to read the ENVM memory from the device. It reads from the client configured in Libero or a page range can be given as inputs. The output will be in a matrix form displayed byte-wise and several rows with page number information.

Client name is optional in the command. However, if client name is specified, then it is validated against its start page and end page from the design.

```
read_envm_memory [-deviceName "device name"] \
  [-client "client name"] \
  -startpage "integer value" \
  -endpage "integer value" \
  [-fileName "envm data file name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
client	string	Specifies the client name.
startpage	string	Specifies the start page that is integer value.
endpage	string	Specifies the end page that is integer value.
fileName	string	Specifies the file name path where the data will be saved.

Return Type	Description
String	The output will be in a matrix form displayed byte-wise and several rows with page number information.

Error Codes

Error Code	Description
None	Required parameter 'startpage' is missing.
None	Required parameter 'endpage' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'read_envm_memory [-deviceName "device name"] [-client "client name"] -startpage "integer value" -endpage "integer value" [-fileName "envm data file name"]'

Supported Families

PolarFire SoC*

Example

This example reads eNVM memory from 0 to 205 pages.

```
read_envm_memory -startpage "0" -endpage "205"
```

5.1.45. read_id_code [\(Ask a Question\)](#)

Description

This Tcl command reads the ID code of a device. Each device has a unique ID, thereby executing this command returns a hexadecimal value.

Note:

Being able to read the IDCODE is an indication that the JTAG interface is working correctly.

```
read_id_code [-deviceName "device name"]
```

Arguments

Parameter	Type	Description
deviceName	none	Specify device name. This parameter is optional if only one device is available in the current configuration.

Return Type	Description
Hexadecimal number	Returns the hexadecimal ID code of the DUT/product.

Error Codes

Error Code	Description
None	None

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following command reads the IDCODE from the current configuration:

```
read_id_code
```

See Also

- `set_debug_device`
- `read_device_status`

5.1.46. read_device_status ([Ask a Question](#))**Description**

This Tcl command displays a summary of the device. Device status like ID code, design information, digest information, security and programmer information can be know using this command. Returns a log that can be saved to a file or printed.

```
read_device_status [-deviceName "device name"] [-file "filename"]
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
file	string	Specify path and the name of file where device status will be saved. This parameter is optional.

Return Type	Description
String	Displays the device information report with the value-property format.

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'read_device_status [-deviceName "device name"] [-file "filename"]'
None	Unable to read device information for the selected device: IDCode verify failed..

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example saves the details of the device in log_file:

```
read_device_status -file {./log_file}
```

See Also

- `read_id_code`

5.1.47. read_lsram [\(Ask a Question\)](#)

Description

This tcl command reads a specified block of large SRAM from the device.

```
read_lsram [-deviceName "device name"] \
            [-name "LSRAM block name"] \
            [-logicalBlockName "USRAM user defined block name"] \
            [-port "LSRAM port name"] \
            [-fileName "Data file name"] \
            [-file "Data file name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
name	string	Specifies the name for the target block.
logicalBlockName	string	Specifies the name for the user defined memory block.
port	string	Specifies the port for the memory block selected. Can be either Port A or Port B.
fileName	string	Optional; specifies the output file name for the data read from the device.
file	string	Optional; specifies the output file name for the data read from the device.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Parameter 'fileName' has illegal value.
None	Port port_name is an invalid Port name.
None	Parameter 'port' has illegal value.
None	RAM port name must be specified.
None	LSRAM block cannot be read. Use physical block option to read.
None	Parameter 'logicalBlockName' has illegal value.
None	Missing argument. Must specify '-name' or '-logicalBlockName'.
None	Parameter 'deviceName' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is read_lsram [-deviceName "device name"] [-name "LSRAM block name"] [-logicalBlockName "USRAM user defined block name"] [-port "LSRAM port name"] [-fileName "Data file name"] [-file "Data file name"].
None	LSRAM block name failed to read: Target block not found in debug file.
None	Parameter 'name' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

Reads the LSRAM Block Fabric_Logic_0/U2/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM1K20_IP from the PolarFire device and writes it to the file output.txt.

```
read_lsram \
-name {Fabric_Logic_0/U2/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM1K20_IP} \
-fileName {output.txt}
```

This example reads the uSRAM logical Block {Fabric_Logic_0/U3/F_0_F0_U1} from {Port A}.

```
read_lsram -logicalBlockName {Fabric_Logic_0/U2/F_0_F0_U1} -port {Port A}
```

See Also

- write_lsram

5.1.48. read_usram [\(Ask a Question\)](#)

Description

This tcl command reads a uSRAM block from the device.

```
read_usram [-deviceName "device name"] \
[-name "USRAM block name"] \
[-logicalBlockName "USRAM user defined block name"] \
[-port "USRAM port name"] \
[-file "Data file name"] \
[-fileName "Data file name"]
```

Physical block

```
read_usram -name {RAMS_LSRAM_URAM_0/PF_DPSRAM_C0_0/PF_DPSRAM_C0_0/
PF_DPSRAM_C0_PF_DPSRAM_C0_0_PF_DPSRAM_R0C0/INST_RAM1K20_IP}
```

Logical block

```
read_usram -logicalBlockName {RAMS_LSRAM_URAM_0/PF_URAM_C0_0/PF_URAM_C0_0} -port {Port A}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
name	string	Specifies the name for the target block.
logicalBlockName	string	Specifies the name of the user defined memory block.
port	string	Specifies the port of the memory block selected. Can be either Port A or Port B.
file	string	This parameter is optional. Specifies the output file name for the data read from the device.
fileName	string	This parameter is optional. Specifies the output file name for the data read from the device.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Missing argument. Must specify '-name' or '-logicalBlockName'.
None	Error: Parameter 'param_name' is not defined. Valid command formatting is 'read_usram [-deviceName "device name"] [-name "USRAM block name"] [-logicalBlockName "USRAM user defined block name"] [-port "USRAM port name"] [-file "Data file name"] [-fileName "Data file name"]'.
None	Parameter 'name' has illegal value.
None	Error reading USRAM block value from the device: Target block not found in debug file.
None	Parameter 'file' has illegal value.
None	Port_name is an invalid Port name.
None	Parameter 'port' has illegal value.
None	RAM port name must be specified.
None	LSRAM block cannot be read. Use physical block option to read.
None	Parameter 'logicalBlockName' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

Reads the uSRAM Block Fabric_Logic_0/U3/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM64x12_IP from the PolarFire device and writes it to the file sram_block_output.txt.

```
read_usram \
-name {Fabric_Logic_0/U3/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM64x12_IP} \
-fileName {output.txt}
```

This example reads the uSRAM logical Block {Fabric_Logic_0/U3/F_0_F0_U1} from {Port A}.

```
read_usram -logicalBlockName {Fabric_Logic_0/U3/F_0_F0_U1} -port {Port A}
```

See Also

- write_usram

5.1.49. read_snvm_memory [\(Ask a Question\)](#)

Description

This Tcl command reads client or page(s) in the sNVM (Secure Non volatile memory) memory from the device and returns a plain text (status and data of page).

Note:

If you have more than one client configured in Libero and use a client name with inappropriate -startpage, -endpage or -uskKey option values, then the command will fail.

```
read_snvm_memory [-deviceName "device name"] \
[-client "client name"] \
-startpage "integer value" \
-endpage "integer value" \
```

```
[ -fileName "snvm data file name" ] \
-uskKey "usk key"
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
client	string	Name of the client configured in Libero.
startpage	integer	Start page number configured in Libero.
endpage	integer	End page number.
fileName	string	Name of output file for memory read.
uskKey	hexadecimal	User Secret Key security key configured for the client in hexadecimal format

Return Type	Description
String	Displays page status and data of sNVM memory with plain text format.

Error Codes

Error Code	Description
None	Number of usk keys entered do not match with page range mentioned.
None	The start page value is incorrect for the client entered.
None	The end page value is incorrect for the client entered.
None	Invalid usk key specified. Input should be either '0' or 24 hexadecimal characters.
None	Parameter 'param_name' is not defined. Valid command formatting is 'read_snvm_memory [-deviceName "device name"] [-client "client name"] -startpage "integer value" -endpage "integer value" [-fileName "snvm data file name"] -uskKey "usk key"'.
None	Parameter 'uskKey' has illegal value.
None	Required parameter 'uskKey' is missing.
None	Parameter 'fileName' has illegal value.
None	Parameter 'endpage' has illegal value.
None	Parameter 'startpage' has illegal value.
None	Parameter 'client' has illegal value.
None	Client name was not found.
None	Parameter 'deviceName' has illegal value.
None	Parameter 'startpage' must be a positive integer value.
None	startpage: Invalid argument value: 'value' (expecting integer value).
None	endpage: Invalid argument value: 'value' (expecting integer value).
None	Parameter 'endpage' must be a positive integer value.

Supported Families

PolarFire

PolarFire SoC

Example

Read "0" startpages and "2" endpages from sNMV memory from the device and save into snvm.txt file:

```
read_snvm_memory -startpage "0" -endpage "2" \
                -fileName "snvm.txt" -uskKey {0:0:0}
```

5.1.50. read_uprom_memory [\(Ask a Question\)](#)

Description

This Tcl command reads a uPROM memory block from the device.

Note:

If you have more than one clients configured in Libero and If you use client name with inappropriate -startaddress and -words then the command will fail.

```
read_uprom_memory [-deviceName "device name"] \
                  [-client "client name"] \
                  -startaddress "start address" \
                  -words "integer value" \
                  [-fileName "Data file name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
client	string	Specify the name of client for memory read. Clients are configured in Libero tool under "device and memory initialisation". Those are a bunch of pages which can be used by the design to load data. Each client can have a different purpose.
startaddress	hexadecimal	Specifies the start address of the uPROM memory block.
words	integer	Specifies the number of 9-bit words.
fileName	string	Name of output file for memory read.

Return Type	Description
String	Displays page status and data of uPROM memory with plain text format..

Error Codes

Error Code	Description
None	Parameter 'deviceName' has illegal value.
None	Parameter 'client' has illegal value.
None	Parameter 'startaddress' has illegal value.
None	Required parameter 'startaddress' is missing.
None	Parameter 'words' has illegal value.
None	Required parameter 'words' is missing.
None	Parameter 'fileName' has illegal value.
None	Invalid argument value: ' -word {1.0} ' (expecting integer value).
None	Parameter 'param_name' is not defined. Valid command formatting is 'read_uprom_memory [-deviceName "device name"] [-client "client name"] -startaddress "start address" -words "integer value" [-fileName "Data file name"]'.
None	The startaddress value is incorrect for the client entered.

read_uprom_memory (continued)

Error Code	Description
None	The word number is incorrect for the client entered.
None	Parameter 'words' must be a positive integer value.

Supported Families

PolarFire

PolarFire SoC

Example

1. This example reads 10 9-bit words from uPROM memory '0xA' address :

```
read_uprom_memory -startaddress {0xA} -words {10}
```

2. This example reads 9-bit words from uPROM memory '0xA' address :

```
read_uprom_memory -client {client1} \
  -startaddress "2" \
  -words "2" \
  -fileName "./data.txt"
```

5.1.51. read_flash_memory [\(Ask a Question\)](#)

Description

The command reads information from the NVM(Non volatile memory) modules. There are two types of information that can be read:

- Page Status – includes ECC1, ECC2, status, write count, access protection.
- Page Data

```
read_flash_memory [-deviceName { device_name }] \
  [-startpage { integer_value }] \
  [-endpage { integer_value }] \
  [-access { all | status | data }] \
  [-file { filename }]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
startpage	integer	Startpage is for page range. The value must be an integer. You must specify a -endpage and -block along with this argument.
endpage	integer	Endpage is for page range. The value must be an integer. You must specify a -startpage and -block along with this argument.
access	string	Specifies what eNVM information to check: page status, data or both. By default "all".
file	string	Name of output file for memory read.

Return Type	Description
String	Displays page status and data of Flash Memory Content with plain text format.

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Parameter 'access' has illegal value.
None	Parameter 'endpage' has illegal value.
None	Parameter 'startpage' has illegal value.
None	Parameter 'deviceName' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'read_flash_memory [-deviceName "device name"] [-block "integer value"] [-client "client name"] [-startpage "integer value"] [-endpage "integer value"] [-access "all status data raw"] [-file "filename"]'

Supported Families

SmartFusion 2

IGLOO 2*

RTG4*

Example

This example checks eNVM data information from 0 to 2 pages.

```
read_flash_memory -startpage 0 -endpage 2 -access {data} \
                  -file {flash_memory}
```

See Also

- [check_flash_memory](#)

5.1.52. record_actions [\(Ask a Question\)](#)

Description

This sequence can be used to access registers from an external processor to perform the same actions done in SmartDebug, to provide the register sequence for each of the actions performed in the XCVR Debug Window.

Note:

This command is valid only when the XCVR block is presented in Libero Design.

```
record_actions -start_recording | -stop_recording -file {file name}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
start_recording	none	Specifies the moment of start recording.
stop_recording	none	Specifies the moment of stop recording.
file	string	Specify path and the name of output *.txt file. This parameter is mandatory when <code>stop_recording</code> is specified.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Start Record parameter already set. Recording is in progress.
None	Record action parameters are absent. Either -start_recording or -stop_recording should be passed as a parameter.
None	Both parameters cannot be passed. Either -start_recording or -stop_recording should be passed as a parameter.
None	Error: Parameter 'param_name' is not defined. Valid command formatting is 'record_actions [-deviceName "device name"] [-start_recording "TRUE FALSE"] [-stop_recording "TRUE FALSE"] [-file "file name"]'.

Supported Families

PolarFire

PolarFire SoC

Example

This example starts recording, then stops it and saves the recorded data in the {./actions} file.

```
record_actions -start_recording
record_actions -stop_recording -file {./actions.txt}
```

5.1.53. remove_from_probe_group [\(Ask a Question\)](#)

Description

This Tcl command removes the specified probe points from the group. That is, the removed probe points won't be associated with any probe group.

This command will fail if the specified value of the parameter is incorrect.

Note:

Probes cannot be removed from the bus.

```
remove_from_probe_group -name {group_name.probe_point_name}
```

Arguments

Parameter	Type	Description
name	string	Specifies one or more probe points to remove from the probe group.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'name' has illegal value.
None	Required parameter 'name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'remove_from_probe_group [-name "name"]+'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example removes DFN1_0_Q:DFN1_0/U0:Q instace from new_group.

```
remove_from_probe_group -name new_group.DFN1_0_Q:DFN1_0/U0:Q
```

See Also

- create_probe_group
- add_to_probe_group
- move_to_probe_group

5.1.54. remove_probe_insertion_point [\(Ask a Question\)](#)

Description

This Tcl command removes probe point from probe insertion list. The command will fail if the net name or driver are not specified or are incorrect.

Notes:

- Deleting probes from the probes list without clicking 'Run' does not automatically remove the probes from the design.
- Probe Insertion feature disabled in the SmartDebug Demo and Standalone modes.

```
remove_probe_insertion_point -net {net_name} -driver {driver_name}
```

Arguments

Parameter	Type	Description
net	string	Specify name of the existing net which is added in probe insertion list. This parameter is mandatory.
driver	string	Specify driver name. This parameter is mandatory.

Return Type	Description
None	Probe insertion operations are not supported in Standalone SmartDebug.

Error Codes

Error Code	Description
None	No probe point with net: "net_name" and driver: "driver_name" is added to be removed.
None	Parameter 'param_name' is not defined. Valid command formatting is 'remove_probe_insertion_point [-net "net_name"] [-driver "driver_name"] '.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example removes probe from the probe insertion list:

```
remove_probe_insertion_point -net {count_c[0]} -driver {Counter_out[0]:Q}
```

See Also

- `add_probe_insertion_point`
- `program_probe_insertion`

5.1.55. `rescan_programmer` [\(Ask a Question\)](#)

Description

This Tcl command rescans for programmer connected to the host via the USB port.

Notes:

- This command does not have any arguments.
- In the demo mode, this command returns "simulation".

```
rescan_programmer [-deviceName "device name"]
```

Arguments

Return Type	Description
String	Displays the programmer ID.

Error Codes

Error Code	Description
None	No programmers found. Check the programmer connection to the computer and ensure the drivers are properly installed.
None	Parameter 'param_name' is not defined. Valid command formatting is 'rescan_programmer [-deviceName "device name"]'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example rescans the programmer connection.

```
rescan_programmer
```

This example rescans and retrieves the programmer name.

```
set programmer_id [rescan_programmer]
puts $programmer_id
```

5.1.56. run_frequency_monitor [\(Ask a Question\)](#)

Description

This tcl command calculates the frequency of any signal in the design that can be assigned to Live Probe channel A and displays the Frequency. The Frequency unit of measurement is in Megahertz (MHz).

It is run after setting the live probe signal to channel A.

```
run_frequency_monitor [-deviceName "device name"] \
    -signal "signal name" \
    -time "time in seconds to delay before calculate"
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
signal	string	Specifies the signal name assigned to Live Probe channel A. This parameter must be specified with the -time parameter.
time	integer or double	Specifies the duration in seconds to run frequency monitor. The value can be 0.1, 1, 5, 8, or 10.

Return Type	Description
String	Displays the Frequency with value-property format.

Error Codes

Error Code	Description
None	No recognized device 'device_name' is available for debugging.
None	Parameters are missing.
None	Parameter '-signal' is missing.
None	Parameter '-time' is missing.
None	Invalid monitor time specified. The values can be either 0.1, 1, 5, 8 or 10.
None	Parameter 'param_name' is not defined. Valid command formatting is 'run_frequency_monitor [-deviceName "device name"] [-signal "signal"] [-time "time"]'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

```
set_live_probe -probeA {Q_c:DFN1_0:Q} -probeB {}
run_frequency_monitor -signal {Q_c:DFN1_0:Q} -time {0.1}
```

See Also

- fhb_control
- set_live_probe
- event_counter

5.1.57. save_active_probe_list ([Ask a Question](#))**Description**

This Tcl command saves the list of active probes to a file.

```
save_active_probe_list [-deviceName "device name"] \  
-file "path to the file"
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
file	string	The output file location.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'file' has illegal value.
None	Required parameter 'file' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'save_active_probe_list [-deviceName "device name"] -file "filename"'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example saves the active probe list in “./my_probes.txt” file.

```
save_active_probe_list -file “./my_probes.txt”
```

See Also

- delete_active_probe
- load_active_probe_list
- read_active_probe
- select_active_probe
- write_active_probe

5.1.58. save_live_probe_list ([Ask a Question](#))**Description**

This Tcl command saves the list of live probes to a file(*.txt).

```
save_live_probe_list [-deviceName "device name" -file "filename"
```

Arguments

Parameter	Type	Description
deviceName	string	Specify device name. This parameter is optional if only one device is available in the current configuration.
file	string	Specify path and the name of output file(*.txt). This parameter is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'file' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'save_live_probe_list [-deviceName "device name"] \ -file "filename"'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example saves live probes list to live_probe_list.txt file:

```
save_live_probe_list -file {./live_probe_list.txt}
```

5.1.59. scan_ecc_memories [\(Ask a Question\)](#)

Description

This Tcl command scans and reports any errors detected in the PolarFire, PolarFire SoC, RTG4, RT PolarFire, and RT PolarFire SoC TPSRAM blocks that have ECC enabled.

```
scan_ecc_memories [-deviceName "device name"] \
                  [-fileName "Output file name"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
fileName	string	Specify the name of the file where output is redirected. This argument is mandatory.

Return Type	Description
string	Reports memory blocks where data corruption has been detected.

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'scan_ecc_memories [-deviceName "device name"] [-fileName "Output file name"]'.
None	Parameter 'fileName' has illegal value.
None	Unable to write to the file.

Supported Families

PolarFire

PolarFire SoC

RTG4

Example

This example scans TPSRAM blocks and saves the report into the {./output.txt} file. Log is provided with names of logical and physical blocks if corruption in data is detected. If no corruption is detected, then an appropriate message is provided.

```
scan_ecc_memories -filename {./output.txt}
```

5.1.60. select_active_probe [\(Ask a Question\)](#)

Description

This Tcl command manages the current selection of active probe points to be used by active probe READ operations. This command extends or replaces your current selection with the probe points found using the search pattern.

```
select_active_probe [-name probe_name_pattern ] \
                   [-reset true|false ]
```

Arguments

Parameter	Type	Description
name	string	Specifies the name of the probe. Optionally, search pattern string can specify one or multiple probe points. The pattern search characters "*" and "?" also can be specified to filter out the probe names.
reset	boolean	This optional parameter resets all previously selected probe points. If name is not specified, empties out current selection.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'reset' has illegal value.
None	Parameter 'name' has illegal value.
None	Cannot select active probe: Specified probe point(s) not found.
None	Parameter 'param_name' is not defined. Valid command formatting is 'select_active_probe [-name "name"]* \[-reset "TRUE FALSE"]'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

```
Select_active_probe -name out[5]:out[5]:Q
Select_active_probe -name out.out[1]:out[1]:Q \
                    -name out.out[3]:out[3]:Q \
                    -name out.out[5]:out[5]:Q
```

See Also

- write_active_probe
- read_active_probe

5.1.61. serdes_lane_reset [\(Ask a Question\)](#)

Description

This Tcl command resets lane in EPCS and PCI Lane modes. The result is shown in the log window/console.

```
serdes_lane_reset -serdes "integer value" -lane "integer value"
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
serdes	integer	Specifies SerDes block number. It must be between 0 and 4 varies between dies. It must be one of the SerDes blocks used in the design.
lane	integer	Specifies SerDes lane number. It must be between 0 and 3. It must be one of the lanes enabled for the block in the design.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'serdes' is missing.
None	Parameter 'serdes' has illegal value.
None	Required parameter 'lane' is missing.
None	Parameter 'lane' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'serdes_lane_reset [-deviceName "device name"] -serdes "integer value" -lane "integer value".

Supported Families

SmartFusion 2

IGLOO 2

RTG4

Example

This example resets Lane 0, for specified SerDes block:

```
serdes_lane_reset -serdes 0 -lane 0
```

See Also

- `serdes_read_register`
- `serdes_write_register`

5.1.62. serdes_read_register ([Ask a Question](#))**Description**

This tcl command reads the SerDes register value and displays the result in the log window/console.

```
serdes_read_register -serdes "integer value" \
                    [-lane "integer value"] \
                    -name "Serdes Register Name"
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
serdes	integer	Specify SerDes block number. Must be between 0 and varies between dies.
lane	integer	Specify SerDes lane number. Must be between 0 and 3. The lane number must be specified when the lane register is used. Otherwise, the command will fail. When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.
name	string	Specify name of the SerDes register.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	SerDes block number must be specified and must be one of the following.
None	Parameter 'serdes' has illegal value.
None	serdes: Invalid argument value: " (expecting integer value).
None	Reg_name is either an invalid or unsupported SerDes register.
None	Parameter 'lane' has illegal value.
None	lane: Invalid argument value: " (expecting integer value).
None	Parameter 'name' has illegal value.
None	0 is either an invalid or unsupported SerDes register.
None	Parameter 'param_name' is not defined. Valid command formatting is 'serdes_read_register [-deviceName "device name"] [-serdes "integer value"] [-lane "integer value"] [-name "Serdes Register Name"] '.

Supported Families

SmartFusion 2

IGLOO 2

RTG4

Example

This example reads {SYSTEM_SER_PLL_CONFIG_HIGH} register value of the SerDes 0.

```
serdes_read_register -serdes 0 -name {SYSTEM_SER_PLL_CONFIG_HIGH}
```

See Also

- `serdes_lane_reset`
- `serdes_write_register`

5.1.63. `serdes_write_register` [\(Ask a Question\)](#)

Description

This tcl command writes the value to the SerDes register. Displays the result in the log window/console.

```
serdes_write_register [-serdes "integer value"] \  
                     [-lane "integer value"] \  
                     -name "Serdes register name" \  
                     -value "Serdes register value"
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
serdes	integer	SerDes block number. Must be between 0 and 5 and varies between dies.
lane	integer	SerDes lane number. Must be between 0 and 3. The lane number should be specified when the lane register is used. Otherwise, the command will fail. When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.
name	string	Name of the SerDes register.
value	hexadecimal	Specify the value in hexadecimal format.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'name' is missing.
None	Parameter 'serdes' has illegal value.
None	Parameter 'lane' has illegal value.
None	Parameter 'name' has illegal value.
None	Required parameter 'value' is missing.

serdes_write_register (continued)

Error Code	Description
None	Parameter 'value' has illegal value.
None	'Reg_name' is either an invalid or unsupported SerDes register.
None	SerDes lane number should not be specified for system register.
None	Parameter 'parm_name' is not defined. Valid command formatting is 'serdes_write_register [-deviceName "device name"] [-serdes "integer value"] [-lane "integer value"] -name "Serdes register name" -value "Serdes register value" '

Supported Families

SmartFusion 2

IGLOO 2

RTG4

Example

This example writes {0x5533} value to the {SYSTEM_SER_PLL_CONFIG_HIGH} SerDes register:

```
serdes_write_register -serdes 0 \
    -name {SYSTEM_SER_PLL_CONFIG_HIGH} \
    -value {0x5533}
```

See Also

- `serdes_lane_reset`
- `serdes_read_register`

5.1.64. set_debug_device ([Ask a Question](#))**Description**

Sets the device name in the tool.

```
set_debug_device -name "device name"
```

Arguments

Parameter	Type	Description
name	string	Specify the name of the device. This argument is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'set_debug_device -name "device name"'
None	Required parameter 'name' is missing.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example sets MPF250T_ES as the device name:

```
set_debug_device -name "MPF250T_ES"
```

5.1.65. set_debug_programmer [\(Ask a Question\)](#)

Description

Identifies the programmer you want to use for debugging (if you have more than one).

```
set_debug_programmer -name { programmer_name }
```

Arguments

Parameter	Type	Description
name	string	Specify the programmer. This argument is mandatory.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Unable to select programmer 'prog_name' for debug.
None	Required parameter 'name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'set_debug_programmer -name "programmer name"'
None	Parameter 'name' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example selects the programmer 10841:

```
set_debug_programmer -name {10841}
```

5.1.66. set_hsm_params [\(Ask a Question\)](#)

Description

Saves HSM parameters for the SmartDebug application. These parameters remain in effect until overridden by another invocation of this command.

```
set_hsm_params -hsm_server_name "hsm_server"
```

Arguments

Parameter	Type	Description
-hsm_server_name	string	Specifies the name or IP address of the HSM server computer.

Error Codes

Error Code	Description
None	Required parameter 'hsm_server_name' is missing.
None	Could not get server information. Check server address and connection and try again.
None	HSM server name cannot be empty.
None	hsm_type_u: Invalid argument value: " (expecting TRUE, 1, true, FALSE, 0 or false).
None	Warning:Deprecated 'hsm_type_u' parameter is used.
None	FTP login password must be specified along with the user name.

Supported Families

PolarFire

SmartFusion 2

IGLOO 2

Example

This example sets the IP address of the server name computer.

```
set_hsm_params - hsm_server_name {10.241.140.224}
```

5.1.67. set_live_probe [\(Ask a Question\)](#)

Description

This Tcl command assigns channels A and/or B to the specified probe point(s). At least one probe point must be specified. Only exact probe name is allowed (that is, no search pattern that may return multiple points).

Note:

For RTG4, only one probe channel (Probe Read Data Pin) is available: A

```
set_live_probe [-deviceName "device name" ] \
               [-probeA "probe point A" ] \
               [-probeB "probe point B" ]
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration or set for debug. This parameter is optional.
probeA	string	Specifies target probe point for the probe channel A. This parameter is optional.
probeB	string	Specifies target probe point for the probe channel B. This parameter is optional.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'set_live_probe [-deviceName "device name"] [-probeA "probe point A"] [-probeB "probe point B"] '.
None	Parameter 'probeA' has illegal value.

set_live_probe (continued)

Error Code	Description
None	Cannot set live probes: Probe A is not found.
None	Cannot set live probes: IDCode verify failed.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Exception

- The array must be programmed and active.
- Active probe read or write operation will affect current settings of Live probe since they use the same probe circuitry inside the device.
- Setting only one Live probe channel affects the other one, so if both channels need to be set, they must be set from the same call to set_live_probe.
- Security locks may disable this function.
- To be available for Live probe, ProbeA and ProbeB I/Os must be reserved for Live probe respectively.

Example

The following example sets Live probe channel A to the probe point A12 on device sf2.

```
set_live_probe [-deviceName sf2] [-probeA A12]
```

See Also

- unset_live_probe

5.1.68. signal_integrity_export ([Ask a Question](#))**Description**

This Tcl command exports the current selected parameter options and other physical information for the selected lane/all lanes instance to an external PDC file.

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.

```
signal_integrity_export -lane {physical lane name} \
    -pdc_file_name {path to the *.pdc file}
signal_integrity_export -pdc_file_name {path to the *.pdc file} \
    -all_lanes
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specifies the physical location of the lane. You must specify either 'lane' or 'all_lanes' parameter.
pdc_file_name	string	The path of the pdc file to be saved

signal_integrity_export (continued)

Parameter	Type	Description
all_lanes	none	Specifies all physical location of the lanes. You must specify either 'lane' or 'all_lanes' parameter.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_names' is not defined. Valid command formatting is 'signal_integrity_export [-deviceName "device name"] [-lane "Lane Instance Name"] [-pdc_file_name "PDC File Name"] [-all_lanes "TRUE FALSE"]'.
None	Signal Integrity: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Parameter 'pdc_file_name' has illegal value.
None	Signal Integrity: Must not specify both '-lane' and '-all_lanes' command arguments.

Supported Families

PolarFire

PolarFire SoC

Example

The following example exports the current selected parameter options and other physical information for the "Q0_LANE0" lane instance to an ./SI_Q0LANE0.pdc:

```
signal_integrity_export -lane {Q0_LANE0} \
    -pdc_file_name {./SI_Q0LANE0.pdc}
```

See Also

- signal_integrity_import
- signal_integrity_write
- load_SI_design_defaults

5.1.69. signal_integrity_import [\(Ask a Question\)](#)

Description

This Tcl command imports Signal Integrity parameter options and other physical information for the selected lane/all lanes from an external PDC file.

```
signal_integrity_import -lane {physical lane name} \
    -pdc_file_name {path to the *.pdc file}
signal_integrity_import -all_lanes \
    -pdc_file_name {path to the *.pdc file}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specifies the physical location of the lane. Must specify either '-lane' or '-all_lanes' command arguments.
pdc_file_name	string	The path of the pdc file to be saved.

signal_integrity_import (continued)

Parameter	Type	Description
all_lanes	none	Specifies all physical location of the lanes. Must specify either '-lane' or '-all_lanes' command arguments.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'signal_integrity_import [-deviceName "device name"] [-lane "Lane Instance Name"] [-all_lanes "TRUE FALSE"] [-pdc_file_name "PDC File Name"]'.
None	Signal Integrity: Must specify '-pdc_file_name'.
None	Signal Integrity: Import from *.pdc failed. Signal Integrity Constraints of lane not available in the file.
None	Signal Integrity: Unable to Import from *.pdc file.
None	Signal Integrity: Must specify one of '-lane' or '-all_lanes' command arguments.
None	Signal Integrity: Must not specify both '-lane' and '-all_lanes' command arguments.
None	Signal Integrity: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.

Supported Families

PolarFire

PolarFire SoC

Example

The following example imports Signal Integrity parameter options and other physical information for the "Q0_LANE0" lane from ./SI_Q0LANE0.pdc:

```
signal_integrity_import -lane {Q0_LANE0} \
                        -pdc_file_name {./SI_Q0LANE0.pdc}
```

See Also

- signal_integrity_export
- signal_integrity_write
- load_SI_design_defaults

5.1.70. signal_integrity_write [\(Ask a Question\)](#)

Description

This Tcl command writes parameter to a specified lane.

```
signal_integrity_write \
[-deviceName "device name"] \
[-lane "Lane Instance Name"] \
[-TX_EMPHASIS_AMPLITUDE "TX Transmit Emphasis and Amplitude"] \
[-TX_IMPEDANCE "TX Impedance"] \
[-TX_POLARITY "TX Impedance"] \
[-TX_TRANSMIT_COMMON_MODE_ADJUSTMENT "TX Transmit Common Mode Adjust"] \
[-RX_TERMINATION "RX Termination"] \
[-RX_LOSS_OF_SIGNAL_DETECTOR_LOW "RX Loss of Signal Detector Low "] \
[-RX_LOSS_OF_SIGNAL_DETECTOR_HIGH "RX Loss of Signal Detector High "] \
[-RX_PN_BOARD_CONNECTION "RX Board Connection "] \
[-RX_POLARITY "Polarity RX "] \
[-RX_CTLE "RX CTLE"] \
```

```
[-RX_INSERTION_LOSS "RX Insertion Loss"] \  
[-RX_CDR_GAIN "RX CDR Gain"]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration.
lane	string	Specifies the physical location of the lane.
TX_EMPHASIS_AMPLITUDE	string	Specifies TX Emphasis Amplitude.
TX_IMPEDANCE	integer	Specifies TX Impedance(ohms) value. Possible values are: 100, 150, 85 180.
TX_TRANSMIT_COMMON_MODE_ADJUSTMENT	integer	Specifies TX Transmit Common Mode Adjustment (% of VDDA). Possible values are: 50, 60, 70, and 80.
RX_INSERTION_LOSS	string	Specifies RX Insertion Loss. Possible values are: 6.5dB, 17.0dB and 25.0dB.
RX_CTLE	string	RX CTLE value.
RX_CDR_GAIN	string	Specifies CDR Gain value. It can be "Low" or "High".
RX_TERMINATION	integer	Specifies RX Termination(ohms). Possible values are: 85, 100 and 150.
RX_PN_BOARD_CONNECTION	string	Specifies RX P/N Board Connection. Possible values are "AC_COUPLED_WITH_EXT_CAP" or "DC_COUPLED".
RX_LOSS_OF_SIGNAL_DETECTOR_LOW	string/ integer	Specifies RX Loss Signal Detector value. Possible values are: Off, PCIE, SATA, BMR and 1, 2, 3, 4, 5, 6 and 7.
RX_LOSS_OF_SIGNAL_DETECTOR_HIGH	string/ integer	Specifies RX Loss Signal Detector value. Possible values are: Off, PCIE, SATA, BMR and 1, 2, 3, 4, 5, 6 and 7.
RX_POLARITY	string	Specifies Polarity (P/N reversal) value, it can be "Normal" or "Inverted".

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'signal_integrity_write [-deviceName "device name"] [-lane "Lane Instance Name"] [-TX_EMPHASIS_AMPLITUDE "TX Transmit Emphasis and Amplitude"] [-TX_IMPEDANCE "TX Impedance"] [-TX_POLARITY "TX Impedance"] [-TX_TRANSMIT_COMMON_MODE_ADJUSTMENT "TX Transmit Common Mode Adjust"] [-RX_TERMINATION "RX Termination"] [-RX_LOSS_OF_SIGNAL_DETECTOR_LOW "RX Loss of Signal Detector Low "] [-RX_LOSS_OF_SIGNAL_DETECTOR_HIGH "RX Loss of Signal Detector High "] [-RX_PN_BOARD_CONNECTION "RX Board Connection "] [-RX_POLARITY "Polarity RX "] [-RX_CTLE "RX CTLE"] [-RX_INSERTION_LOSS "RX Insertion Loss"] [-RX_CDR_GAIN "RX CDR Gain"]'.
None	Parameter 'RX_CDR_GAIN' has illegal value.
None	Parameter 'RX_INSERTION_LOSS' has illegal value.
None	Parameter 'RX_CTLE' has illegal value.
None	Signal Integrity: RX_INSERTION value is invalid. It must be 6.5dB, 17.0dB or 25.0dB.
None	Parameter 'RX_POLARITY' has illegal value.
None	Parameter 'RX_PN_BOARD_CONNECTION' has illegal value.
None	Parameter 'RX_LOSS_OF_SIGNAL_DETECTOR_HIGH' has illegal value.

signal_integrity_write (continued)

Error Code	Description
None	Parameter 'RX_LOSS_OF_SIGNAL_DETECTOR_LOW' has illegal value.
None	Parameter 'RX_TERMINATION' has illegal value.
None	Parameter 'TX_TRANSMIT_COMMON_MODE_ADJUSTMENT' has illegal value.
None	Parameter 'TX_IMPEDANCE' has illegal value.
None	Parameter 'TX_EMPHASIS_AMPLITUDE' has illegal value.
None	Signal Integrity: Must specify one of '-TX_EMPHASIS_AMPLITUDE', '-TX_IMPEDANCE', '-TX_POLARITY', '-TX_TRANSMIT_COMMON_MODE_ADJUSTMENT', '-RX_TERMINATION', '-RX_LOSS_OF_SIGNAL_DETECTOR_LOW', '-RX_LOSS_OF_SIGNAL_DETECTOR_HIGH', 'RX_PN_BOARD_CONNECTION', '-RX_POLARITY', '-RX_CTLE', '-RX_INSERTION_LOSS' or '-RX_CDR_GAIN' arguments.
None	Parameter 'lane' has illegal value.

Supported Families

PolarFire

PolarFire SoC

Example

Write signal integrity on "Q2_LANE0" lane with the possible values of parameters:

```
signal_integrity_write \
-lane {Q2_LANE0} \
-TX_EMPHASIS_AMPLITUDE {400mV_with_-3.5dB} \
-TX_IMPEDANCE {100} \
-TX_TRANSMIT_COMMON_MODE_ADJUSTMENT {50} \
-RX_TERMINATION {100} \
-RX_LOSS_OF_SIGNAL_DETECTOR_LOW {1} \
-RX_LOSS_OF_SIGNAL_DETECTOR_HIGH {3} \
-RX_PN_BOARD_CONNECTION {AC_COUPLED_WITH_EXT_CAP} \
-RX_POLARITY {Normal} \
-RX_CTLE {No_Peak_+2.8dB} \
-RX_INSERTION_LOSS {6.5dB} \
-RX_CDR_GAIN {High}
```

See Also

- signal_integrity_import
- signal_integrity_export
- load_SI_design_defaults

5.1.71. smartbert_test [\(Ask a Question\)](#)

Description

This Tcl command is used for the following:

- Start a Smart BERT test - Start a test with a specified PRBS patterns on a specified SmartBERT lane.
- Stop a Smart BERT test - Stop SmartBERT/PRBS test on a specified lane.
- Reset error count - Reset counter of a lane during selected pattern test.
- Inject error - Inject error into a SmartBERT IP lane.

```
smartbert_test -start -pattern {pattern name} \
               -lane {Physical Location} \
               [-smartbert_ip {TRUE | FALSE}] \
               [-EQ-NearEndLoopback]
smartbert_test -reset_counter -lane {Physical Location}
```

```
smartbert_test -lane {Physical Location} [-inject_error {TRUE | FALSE}]
smartbert_test -stop -lane {Physical Location}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
lane	string	Specify the physical location of the lane.
start	none	Start the Smart BERT test.
pattern	string	Specify the pattern type of the Smart BERT test. Valid values of pattern type are: PRBS7, PRBS9, PRBS15, PRBS23 and PRBS31.
smartbert_ip	boolean	This parameter applicable to the lane configured through SmartBERT IP.
EQ-NearEndLoopback	none	Enable EQ-Near End Loopback on specified lane.
reset_counter	none	Reset lane error counter on hardware and cumulative error count on the UI.
inject_error	boolean	Specifies to inject error into a SmartBERT IP. Valid values are: TRUE or FALSE.
stop	none	Stop the smart BERT test.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	SmartBert test: Must specify one of '-start', '-stop', '-reset_counter' or '-read_counter' arguments.
None	SmartBert test: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	PRBS test: Invalid pattern type specified.
None	SmartBert test: Is not a G5 Device.
None	Parameter 'param_name' is not defined. Valid command formatting is 'smartbert_test [-deviceName "device name"] [-smartbert_ip "TRUE FALSE"] [-start "TRUE FALSE"] [-stop "TRUE FALSE"] [-reset_counter "TRUE FALSE"] [-read_counter "TRUE FALSE"] [-pattern "Pattern type"] [-lane "Physical Lane Name"] [-EQ-NearEndLoopback "TRUE FALSE"] [-inject_error "TRUE FALSE"]'.

Supported Families

PolarFire

PolarFire SoC

Example

The following example starts Smart BERT test with a "prbs7" PRBS patterns on a "Q0_LANE0" SmartBERT lane:

```
# Transceiver lane without SmartBert IP without EQ-NearEndLoopback
smartbert_test -start -pattern {prbs7} -lane {Q0_LANE0}

# Transceiver SmartBERT IP lane
smartbert_test -start -smartbert_ip "TRUE" \
```

```
-pattern {prbs7} -lane {Q0_LANE0} \
-EQ-NearEndLoopback "TRUE"
```

The following example resets counter of a "Q0_LANE0" lane during selected pattern test.

```
smartbert_test -reset_counter -lane {Q0_LANE0}
```

The following stops Smart BERT/PRBS test on a "Q0_LANE0" SmartBERT lane:

```
smartbert_test -stop -lane {Q0_LANE0}
```

The following example injects error into a "Q0_LANE0" SmartBERT IP lane:

```
smartbert_test -lane {Q0_LANE0} -inject_error {TRUE}
```

See Also

- prbs_test
- loopback_mode
- loopback_test

5.1.72. static_pattern_transmit [\(Ask a Question\)](#)

Description

This Tcl command starts and stops a Static Pattern Transmit on selected lanes.

```
static_pattern_transmit -start -lane {Transceiver Physical Lane Name} \
-pattern {pattern_type} \
-value {user_pattern_value}
static_pattern_transmit -stop -lane {Transceiver Physical Lane Name} \
-pattern {empty} -value {empty}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the <code>set_debug_device</code> command.
lane	string	Specify Transceiver physical Lane Name.
start	none	Start the Static Pattern Transmit.
stop	none	Stop the Static Pattern Transmit.
pattern	string	Specify "pattern_type" of Static Pattern Transmit. "pattern_type" valid values are: <ul style="list-style-type: none"> • fixed - Fixed Pattern is a 10101010... pattern. Length is equal to the data width of the Tx Lane. • maxrunlength - 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. • custom - User Pattern is a user defined pattern in the value column. Length is equal to the data width.
value	hexadecimal	Specify user_pattern_value in hex if pattern_type selected is custom. Takes the input pattern to transmit from the Lane Tx of selected lanes. Internal validators dynamically check the pattern and indicate when an incorrect pattern is given as input.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'static_pattern_transmit [-deviceName "device name"] [-start "TRUE FALSE"] [-stop "TRUE FALSE"] [-lane "Physical Lane Name"] [-pattern "Pattern type"] [-value "User pattern Value"]'.
None	Static Pattern Transmit: Must specify one of '-start', '-stop' arguments.
None	Static Pattern Transmit: Transceiver physical Lane Name must be specified.
None	Static Pattern Transmit: Must specify pattern type argument.
None	Static Pattern Transmit: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Static Pattern Transmit: Invalid static pattern type specified.
None	Static Pattern Transmit: Pattern Length exceeds the expected size.

Supported Families

PolarFire

PolarFire SoC

Example

The following examples starts/stops fixed/maxrunlength Static Pattern transmit on "Q0_LANE0" / "Q0_LANE1" lane:

```
static_pattern_transmit -start -lane {Q0_LANE0} \
                        -pattern {fixed} -value {}
static_pattern_transmit -stop -lane {Q0_LANE0}

static_pattern_transmit -start -lane {Q0_LANE1}
                        -pattern {maxrunlength} -value {}
static_pattern_transmit -stop -lane {Q0_LANE1}
```

The following examples starts/stops fcustom Static Pattern transmit on "Q2_LANE2" lane with "1010111" user pattern value:

```
static_pattern_transmit -start -lane {Q2_LANE2} \
                        -pattern {custom} -value {1010111}
static_pattern_transmit -stop -lane {Q2_LANE2}
```

5.1.73. transceiver_lane_reset [\(Ask a Question\)](#)

Description

This tcl command resets the transceiver lane.

```
transceiver_lane_reset [-deviceName "device name"] \
                        -lane {physical location}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
lane	string	Specify the physical lane instance name.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Phy Reset: Lane Name not found in the list of assigned physical lanes in Libero. Provide the correct lane name.
None	Parameter 'lane' has illegal value.
None	Phy Reset: Transceiver physical Lane Name must be specified.
None	Parameter 'param_name' is not defined. Valid command formatting is 'transceiver_lane_reset [-deviceName "device name"] [-lane "Physical Lane Name"]'.
None	Parameter 'deviceName' has illegal value.

Supported Families

PolarFire

PolarFire SoC

Example

This tcl example resets the lane {Q0_LANE0}.

```
transceiver_lane_reset -lane {Q0_LANE0}
```

5.1.74. tvs_monitor [\(Ask a Question\)](#)

Description

This Tcl command reads the Temperature and Voltage Sensor (TVS) values from device and saves the values in a .csv user specified extension file.



Important: This command is valid only when the TVS IP is presented in Libero Design.

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
start	none	Starts the TVS monitor.
stop	none	Stops the TVS monitor.
interval	integer	Specifies the duration in Seconds. The allowed limit is 1 sec to 60 sec. This parameter is optional. The default interval is 1 sec.
file	string	Specify the name of the file where output is redirected. This argument is mandatory.

Error Codes

Error Code	Description
None	TVS Monitor feature is not available for this design.
None	Failed to read TVS Channels values.
None	Parameters are absent. Either -start or -stop should be passed as a parameter.

Supported Families

PolarFire

PolarFire SoC

Example

This tcl example reads the TVS.

```
tvs_monitor -start -file {./temp.csv}
tvs_monitor -start -interval 10 -file {./temp.csv}
tvs_monitor -stop
```

5.1.75. ungroup [\(Ask a Question\)](#)

Description

This Tcl command disassociates the probes as a group.

```
ungroup -name {group name}
```

Arguments

Parameter	Type	Description
name	string	Specifies name of the group.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	None

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

Ungroup 'my_group' probe groups:

```
ungroup -name "my_group"
```

See Also

- [create_probe_group](#)
- [add_to_probe_group](#)

5.1.76. unset_live_probe [\(Ask a Question\)](#)

Description

This Tcl command discontinues the debug function and clears both live probe channels (Channel A and Channel B). An all zeros value is shown for both channels in the oscilloscope.

Note:

For RTG4, only one probe channel (Probe Read Data Pin) is available.

```
unset_live_probe [-deviceName "device name"] \
                 [-probeA "TRUE | FALSE"] \
                 [-probeB "TRUE | FALSE"] '
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
probeA	boolean	Specify 1 or TRUE for unset live probe on Channel A, otherwise specify 0 or FALSE.
probeB	boolean	Specify 1 or TRUE for unset live probe on Channel B, otherwise specify 0 or FALSE.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Cannot unset live probes: Mention the name of the channel to unset.
None	Parameter 'param_name' is not defined. Valid command formatting is 'unset_live_probe [-deviceName "device name"] [-probeA "TRUE FALSE"] [-probeB "TRUE FALSE"]'.
None	probeA: Invalid argument value: (expecting TRUE, 1, true, FALSE, 0 or false).
None	probeB: Invalid argument value: (expecting TRUE, 1, true, FALSE, 0 or false).
None	Parameter 'deviceName' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

The following example unsets both live probe channels (Channel A and Channel B) from the device sf2:

```
unset_live_probes -probeA 1 -probeB 1 [-deviceName {sf2}]
```

See Also

- set_live_probe

5.1.77. update_fp6_programmers [\(Ask a Question\)](#)**Description**

This Tcl command updates all the FlashPro6 programmers that require update. This command takes no parameters. To execute the Tcl Command in Libero SoC, add the command before `run_tool`. To

execute the Tcl command in FlashPro Express tool or SmartDebug tool, add the Tcl command before `run_selected_actions`.

```
update_fp6_programmers
```

Arguments

Parameter	Type	Description
None	None	None

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'update_fp6_programmers'.

Supported Families

PolarFire®

PolarFire SoC

Example

This example updates all the FlashPro6 programmers that require update.

```
update_fp6_programmers
```

5.1.78. write_active_probe [\(Ask a Question\)](#)

Description

This Tcl command sets the target probe point on the device to the specified value. The target probe point name must be specified.

```
write_active_probe [-deviceName device_name ] \
    -name probe_name \
    -value true|false \
    -group_name group_bus_name \
    -group_value "hex-value" | "binary-value"
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration.
name	string	Specifies the name for the target probe point. Cannot be a search pattern.
value	boolean	Specifies values to be written. True = High, False = Low.
group_name	string	Specify the group or bus name to write to complete group or bus.
group_value	string	Specify the value for the complete group or bus. Hex-value format : "<size>'h<value>" Binary-value format: "<size>'b<value>"

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'group_value' has illegal value.
None	Active probe value must be specified.
None	Parameter 'group_name' has illegal value.
None	Parameter 'value' has illegal value.
None	Parameter 'name' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'write_active_probe [-deviceName "device name"] \[-name "Probe point name"]* \[-value "TRUE FALSE"]* \[-group_name "Group or Bus Name"]* \[-group_value "Group or Bus value"]* \[-silent "TRUE FALSE"]'.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example writes to a single probe.

```
write_active_probe -name out[5]:out[5]:Q -value true
```

This example writes to a probe in the group:

```
write_active_probe -name grpl.out[3]:out[3]:Q -value "low"
```

This example writes the value to complete group:

```
write_active_probe -group_name grpl -group_value "8'hF0"
```

This example writes multiple probes at the same time:

```
write_active_probe -group_name out \
                  -group_value "8'b11110000" \
                  -name out[2]:out[2]:Q \
                  -value true
```

See Also

- [select_active_probe](#)
- [read_active_probe](#)
- [save_active_probe_list](#)
- [load_active_probe_list](#)

5.1.79. write_lsram [\(Ask a Question\)](#)

Description

This tcl command writes a word into the specified large TPSRAM location.

TPSRAM block has aspect ratio of 512x40 (ECC disabled) and 512x33 (ECC enabled). SmartDebug enhanced the physical block view to read and write as 40-bit and 33-bit data. The write value is more than the size of integer and hence provided a new parameter -tpsramValue to accommodate the changes

Write onto TPSRAM physical block → 40-bit wide or 33-bit wide for PF and 18-bit wide or 36-bit wide for RTG4

Physical block

```
write_lsram [-deviceName "device name"] \
            -name {physical block name} \
            -offset {offset value} \
            -value {integer value} \
            [-tpsramValue "TPSRAM physical block word value"]
```

Logical block

```
write_lsram [-deviceName "device name"] \
            -logicalBlockName {block name} \
            -port {port name} \
            -offset {offset value} \
            -logicalValue {hexadecimal value}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
name	string	Specifies the name for the target block.
logicalBlockName	string	Specifies the name of the user defined memory block.
port	string	Specifies the port of the memory block selected. Can be either Port A or Port B.
offset	integer	Offset (address) of the target word within the memory block.
logicalValue	hexadecimal	Specifies the hexadecimal value to be written to the memory block. Size of the value is equal to the width of the output port selected.
value	integer	Word to be written to the target location. Depending on the configuration of memory blocks, the width can be 1, 2, 5, 10, or 20 bits. This is an integer, which minimum value is 0 and may go up to depending on the size of each location}
tpsramValue	integer	integer value, minimum value is 0 to $(2^N - 1)$ where N is number of bits configured. PolarFire , PolarFire Soc and RTG4 only.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'write_lsram [-deviceName "device name"] [-name "LSRAM block name"] [-logicalBlockName "LSRAM user defined block name"] [-port "LSRAM port name"] [-offset "integer value"] [-logicalValue "LSRAM block word value"] [-value "integer value"] [-tpsramValue "TPSRAM physical block word value"]'.
None	Parameter 'name' has illegal value
None	Missing argument. Must specify '-name' or '-logicalBlockName'.
None	Parameter 'logicalValue' has illegal value.
None	Error write LSRAM block PF_DPSRAM_C0_0/PF_DPSRAM_C0_0: Target memory block should first be read before write..
None	Parameter 'logicalBlockName' has illegal value.
None	LSRAM block cannot be read. Use physical block option to read.

write_lsram (continued)

Error Code	Description
None	RAM port name must be specified.
None	Parameter 'port' has illegal value.
None	Port port_name is an invalid Port name.
None	Parameter 'file' has illegal value Parameter 'tpsramValue' has illegal value.
None	Parameter 'value' has illegal value.
None	value: Invalid argument value: 'value' (expecting integer value).
None	Parameter 'offset' has illegal value.
None	offset: Invalid argument value: 'value' (expecting integer value).
None	Active probe value must be specified.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

This example writes a value of 69905 to the physical block of device PolarFire in the "PF_DPSRAM_C0_0/INST_RAM1K20_IP" with an offset of 3:

```
write_lsram -name {PF_DPSRAM_C0_0/INST_RAM1K20_IP} \
  -offset 3 -value 69905
```

```
write_lsram -logicalBlockName {PF_DPSRAM_C0_0/PF_DPSRAM_C0_0} \
  -port {Port B} -offset {1} -logicalValue {0xA} \
  -tpsramValue 300
```

See Also

- read_lsram

5.1.80. write_usram [\(Ask a Question\)](#)

Description

This tcl command writes a 12-bit word into the specified uSRAM location.

```
write_usram [-deviceName "device name"] \
  [-name "USRAM block name"] \
  [-logicalBlockName "USRAM user defined block name"] \
  [-port "USRAM port name"] \
  [-offset "integer value"] \
  [-logicalValue "USRAM block word value"] \
  [-value "integer value"]
```

Physical block

```
write_usram -name block_name \
  -offset offset_value \
  -value integer_value
```

Logical block

```
write_usram -logicalBlockName block_name \
  -port port_name \
  -offset offset_value \
  -logicalValue hexadecimal_value
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
name	string	Specifies the name for the target block.
logicalBlockName	string	Specifies the name of the user defined memory block.
port	string	Specifies the port of the memory block selected. Can be either Port A or Port B.
offset	integer	Offset (address) of the target word within the memory block.
logicalValue	integer	Specifies the hexadecimal value to be written to the memory block. Size of the value is equal to the width of the output port selected.
value	integer	12- bit value to be written.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'logicalValue' has illegal value.
None	offset: Invalid argument value: 'offset_value' (expecting integer value).
None	Parameter 'offset' has illegal value.
None	Active probe value must be specified.
None	Port_name is an invalid Port name.
None	Parameter 'port' has illegal value.
None	LSRAM port name must be specified.
None	LSRAM block word cannot be written. Use physical block word to write.
None	Missing argument. Must specify '-name' or '-logicalBlockName'.
None	Parameter 'value' has illegal value.
None	Parameter 'name' has illegal value.

Supported Families

PolarFire

PolarFire SoC

SmartFusion 2

IGLOO 2

RTG4

Example

Writes a value of 0x291 to the device PolarFire in the Fabric_Logic_0/U3/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM64x12_IP with an offset of 0.

```
write_lsram \
-name {Fabric_Logic_0/U3/F_0_F0_U1/ramtmp_ramtmp_0_0/INST_RAM64x12_IP} \
```

```
-offset 0 \
-value 291
```

```
write_usram -logicalBlockName {Fabric_Logic_0/U3/F_0_F0_U1} -port {Port A} -offset 1
-logicalValue {00FFF}
```

See Also

- [read_usram](#)

5.1.81. xcvr_add_register [\(Ask a Question\)](#)**Description**

This Tcl command adds transceiver registers details to the SmartDebug register access interface.

```
xcvr_add_register [-deviceName {device name} \
                  [-reg_name {Add Register Names}]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required, if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
reg_name	string	Name of the register.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	reg_name register is not added to Register access list.
None	Parameter 'reg_name' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'xcvr_add_register [-deviceName "device name"] [-reg_name "Add Register Names"]'.
None	Register Access: Must specify '-reg_name

Supported Families

PolarFire

PolarFire SoC

Example

This example adds all the registers under the {SERDES1} component.

```
xcvr_add_register -reg_name {SERDES1}
```

See Also

- [xcvr_export_register](#)
- [xcvr_read_register](#)
- [xcvr_write_register](#)

5.1.82. **xcsr_export_register** [\(Ask a Question\)](#)

Description

This Tcl command exports previously added transceiver registers details to a *.csv file.

```
xcsr_export_register [-deviceName {device name}] \  
                    [-file_name {file Name}] [-all]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
file_name	string	Path of the export file.
all	none	Specify to export all transceiver registers details to *.csv file.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'param_name' is not defined. Valid command formatting is 'xcsr_export_register [-deviceName "device name"] [-file_name "File Name"] [-all "TRUE FALSE"]'.
None	Parameter 'file_name' has illegal value.
None	Register Access: file specified for Export must have .csv extension.
None	Register Access: Must specify '-file_name'.

Supported Families

PolarFire

PolarFire SoC

Example

Export previously added transceiver registers details to a .csv file:

```
xcsr_export_register -file_name {register_export.csv}
```

See Also

- [xcsr_add_register](#)
- [xcsr_read_register](#)
- [xcsr_write_register](#)

5.1.83. **xcsr_import_register** [\(Ask a Question\)](#)

Description

This Tcl command imports exported transceiver registers details from a *.csv file.

```
xcsr_import_register \  
                    -file_name {absolute or relative path to the *.csv file name}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
file_name	string	Path of the exported transceiver file.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Required parameter 'file_name' is missing.
None	Parameter 'file_name' has illegal value.
None	Register Access: file specified for Import must have .csv extension.
None	Register Access: Must specify '-file_name'.
None	Parameter 'param_name' is not defined. Valid command formatting is 'xcvr_import_register -file_name "File Name"'.

Supported Families

PolarFire

PolarFire SoC

Example

Import previously exported transceiver registers details from a .csv file:

```
xcvr_import_register -file_name {register_list.csv}
```

5.1.84. xcvr_read_register [\(Ask a Question\)](#)

Description

This Tcl command reads SCB registers and their field values. Read value is in hex format. This command is used in SmartDebug Signal Integrity.

```
xcvr_read_register [-deviceName {device name}] \
  -inst_name {instance name} \
  -reg_name [<reg_name> | <reg_name:field_name>]
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
inst_name	string	Specify the lane instance name used in the design.
reg_name	string	Specify the <reg_name> for register name or <reg_name>:<field_name> for the register's field.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'reg_name' has illegal value.
None	Required parameter 'reg_name' is missing.
None	Parameter 'inst_name' has illegal value.
None	Required parameter 'inst_name' is missing.
None	Parameter 'param_name' is not defined. Valid command formatting is 'xcvr_read_register [-deviceName "device name"] -inst_name "Instance Name" -reg_name "Transceiver Register Name" '.

Supported Families

PolarFire

PolarFire SoC

Example

Reading pcslane's 32-bit register LNTV_R0:

```
xcvr_read_register -inst_name {CM1_PCIE_SS_0/PF_PCIE_0/LANE1} \
                  -reg_name {LNTV_R0}
```

See Also

- [xcvr_add_register](#)
- [xcvr_export_register](#)
- [xcvr_write_register](#)

5.1.85. [xcvr_write_register](#) [\(Ask a Question\)](#)

Description

This Tcl command writes SCB registers and their field values. Write value is in hex format. This command is used in SmartDebug Signal Integrity.

```
xcvr_write_register [-deviceName {device name}] \
                   [-inst_name {Instance name}] \
                   -reg_name {Transceiver register name} \
                   -value {Transceiver register value}
```

Arguments

Parameter	Type	Description
deviceName	string	Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
inst_name	string	Specify the lane instance name used in the design.
reg_name	string	Specify the <reg_name> for register name or <reg_name>:<field_name> for the register's field.
value	integer	Specify the value in hex format.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Parameter 'value' has illegal value.
None	Required parameter 'value' is missing.
None	Parameter 'reg_name' has illegal value.
None	Required parameter 'reg_name' is missing.
None	Parameter 'inst_name' has illegal value.
None	Parameter 'param_name' is not defined. Valid command formatting is 'xcsr_write_register [-deviceName "device name"] [-inst_name "Instance name"] [-broadcast "TRUE FALSE"] -reg_name "Transceiver register name" -value "Transceiver register value".
None	Must specify either '-inst_name' or '-broadcast' parameter.

Supported Families

PolarFire

PolarFire SoC

Example

Writing pcscmn's 32-bit register GSSCLK_CTRL

```
xcsr_write_register -inst_name {CM1_PCIE_SS_0/PF_PCIE_0/LANE1} \
                    -reg_name {GSSCLK_CTRL} \
                    -value 0xffffffff
```

See Also

- xcsr_add_register
- xcsr_export_register
- xcsr_read_register

6. Revision History [\(Ask a Question\)](#)

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
P	05/2025	<p>The following list of changes is made in revision P of the document:</p> <ul style="list-style-type: none"> • In section Introduction, added a note that SmartDebug works only with JTAG ports. • In section Supported Device Families, added a row in Table 1 for RT PolarFire® SoC. • In section Supported Tools, revised Table 2, as follows: <ul style="list-style-type: none"> – Added Fabric DDR and MSS DDR, added PCIe Debug. – Merged SerDes and Debug Transceiver onto one row called Debug Transceiver/SerDes. – Changed the column from PolarFire SoC to PolarFire SoC and RT PolarFire SoC. • In section FHB Operations, updated Figure 3-35. Also, under Waveform Capture, added the valid range for capture depth. • In section SmartBERT, updated Figure 3-53. • In section Design Initiated Eye Plots, updated Figure 3-65 and Figure 3-66. • In section Signal Integrity, updated and renamed Figure 3-68. Added a note about Signal Integrity Parameters being modified and added a figure that shows the popup that appears if the Signal Integrity Parameters were modified in the previous debug session. Described the behavior of the Current Loaded Settings Group box. Removed instances of Design Default. • Removed the Design Defaults topic. • In section Optimize Receiver, updated Figure 3-71 and Figure 3-72. • In section Display DFE Coefficient Values, updated Figure 3-73. • In section PCIe Debug, added a note about parameters shown on the PCIe LTSSM State page being read-only. Also, updated Figure 3-74. • In section Lane Status and Lane Link Error Status, updated Figure 3-75. • In section LTSSM State Machine, updated Figure 3-76. • In section Config Space Parameter Information, added Table 3-4. Also, updated Figure 3-78 and Figure 3-79. • In section Record Actions, updated Figure 3-80 and Figure 3-81. • Updated the section Debug Fabric DDR IO Margin. • Added section Debug MSS DDR IO Margin. • In section scan_ecc_memories, added RT PolarFire to the family of devices mentioned in the description. • In section set_live_probe, removed RT PolarFire from the Supported Families. • In various sections, changed “Debug DDR” to “Fabric DDR.”

Revision History (continued)		
Revision	Date	Description
N	08/2024	<p>The following list of changes is made in revision N of the document:</p> <ul style="list-style-type: none"> In section Supported Tools, added TVS Monitor as a supported tool by PolarFire, RT PolarFire, and PolarFire SoC. In section Design Initiated Eye Plots, added a procedure for accessing design-initiated eye plots. Added section Switching Off Power to the Eye Monitor Core. Added section Exporting DFE Coefficients.
M	02/2024	<p>The following list of changes is made in revision M of the document:</p> <ul style="list-style-type: none"> Added a note in Eye Scan Mode and Infinite Persistent Mode sections about eye plot being disabled for lanes with a data-rate less than 3 Gbps.
L	02/2024	<p>The following list of changes is made in revision L of the document:</p> <ul style="list-style-type: none"> Updated the figure of the Export SmartDebug Data dialog box in the Debugging Devices Connected in a JTAG Chain section.
K	08/2023	<p>The following list of changes is made in revision K of the document:</p> <ul style="list-style-type: none"> Section Memory Blocks: Added description and screen shot of the View in 2D check box and the Go to Address option. Section Memory Block Fields: updated screen shots. Section Read Block: added new 1D and 2D screen shots; updated all other screen shots. Section Write Block through section Scan Memory: updated screen shots.
J	04/2023	<p>The following list of changes is made in revision J of the document:</p> <ul style="list-style-type: none"> Section Live Probes: Added a note about the Live Probe tab is disabled and a tooltip is shown if the user design has live probe I/Os enabled as input. Section Active Probes: Added a note about the Assign to Channel A and Unassign from Channel A RMB options are hidden in the Active Probes table if the user design has live probe I/Os enabled as input. Section FPGA Hardware Breakpoint Auto Instantiation: Added an observation about the Spatial Debug Widget and FHB feature are hidden and an INFO message is printed in the log if the user design has live probe I/Os enabled as input. Section Assumptions and Limitations: Revised bullet to say that in SmartFusion® 2, IGLOO® 2, and RTG4™ devices, 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. Section SerDes Register Read or Write: Added Attention, Background, The Issue, Recommendation, and Workaround, along with two helpful hints.

Revision History (continued)

Revision	Date	Description
H	12/2022	<p>The following list of changes is made in revision H of the document:</p> <ul style="list-style-type: none"> Added the topic Debugging Devices Connected in a JTAG Chain Added the topic Measuring Die Temperature and Voltages Added new Tcl commands to the following sections <ul style="list-style-type: none"> complete_debug_job process_job_request set_hsm_params Changed content in the following sections: <ul style="list-style-type: none"> Supported Tools: Removed the check mark from the PolarFire and RT PolarFire column for View Flash Memory Content—eNVM Debug and added a note that the contents of eNVM cannot be debugged in boot mode 0. Live Probes: Updated screen shots and the description of the Assign to Channel button. Event Counter: Revised the entire first paragraph. Activating the Event Counter: There is only one way to activate the Event Counter. Running the Event Counter: Revised the first two sentences in the first paragraph. Stopping the Event Counter: A red LED shows when the Event Counter is stopped. Frequency Monitor: Enabling FPGA Hardware Breakpoint enables Frequency Monitor in the SmartDebug tool and that Frequency Monitor is enabled when a probe point is assigned to Channel A via Live Probe. Also, mentioned that time to monitor the signal before the frequency is calculated can be chosen from the drop-down menu. Activating the Frequency Monitor: There is only one way to activate the Frequency Monitor. Running the Frequency Monitor: Mentioned that a green LED shows when the Frequency Monitor is running. Stopping the Frequency Monitor: A red LED shows when the Frequency Monitor is stopped. FPGA Hardware Breakpoint Auto Instantiation: Removed reference to gated clocks. Revised sentence to read that once a selected clock domain or all clock domains are halted, you can play or step on the clock domains. Added step 5 to the procedure. Assumptions and Limitations: Added that the FHB cannot be added to designs with System Controller Suspend Mode set to ON (all families) and that the FHB SDC file is no longer user-facing in PolarFire and PolarFire SoC. Register Access: Described the drop-down list at the top of the page, updated the figure for Debug TRANSCEIVER - Register Access, and added descriptions for Prev Read Value and Curr Read Value. Recorded Content: Added Signal Integrity to the list of supported Recorded Actions and removed Signal Integrity and PHY from the list of Recorded Actions that are not supported. Debug IOD: Added a note about the pins that must be configured for CORERXIODBITALIGN for IOD training to succeed and to be able to perform tasks such as reading delay taps, left taps, right taps, and bit align errors. Debug eNVM (PolarFire SoC): Added the same note added to section Supported Tools that contents of eNVM cannot be

Revision History (continued)		
Revision	Date	Description
G	09/2022	<p>The following list of changes is made in revision G of the document:</p> <ul style="list-style-type: none"> Updated Important note to reflect MSS DDR subsystem in section Supported Tools.
F	08/2022	<p>The following list of changes is made in revision F of the document:</p> <ul style="list-style-type: none"> In section Supported Tools, added a note about Debug DDR not being supported for the PolarFire SoC MSSDDR. In section Tools Menu, added a note about a warning pop-up that appears when Discrete Clocking mode is selected In section Optimize Receiver: <ul style="list-style-type: none"> Added “and/or DFE coefficients” to the description. Reworded sentence to read: For CDR mode receivers, CTLE settings and/or DFE coefficients can be optimized. Reworded sentence to read: The hardware will perform Full calibration for DFE mode receivers and may perform CTLE calibration only or CTLE and DFE calibration on CDR mode receivers. Revised Figure 3-72 showing the Optimize Receiver dialog box. In section Display DFE Coefficient Values: <ul style="list-style-type: none"> Added CDR to the sentence: 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 or CDR mode. Revised the first bullet to state that DFE coefficients will be shown for CDR mode receivers if force DFE calibration is selected. In section Debug IOD and section debug_iod, added a note that “CORERXIODBITALIGN IP” must have the output debug pins either connected or promoted to the top for SmartDebug to detect and identify the debug signals. In section Debug Fabric DDR IO Margin, mentioned that error information can be displayed as tooltips.
E	04/2022	<p>The following list of changes is made in revision E of the document:</p> <ul style="list-style-type: none"> In section Programming Connectivity and Interface, mentioned that SmartDebug can detect an outdated FlashPro6 programmer and allow the user to update the programmer design. Revised sections Client View and Client View to show and describe the Export button. Added the new section update_fp6_programmers. Updated the Introduction and sections Integrated Mode, Standalone Mode, and Demo Mode. In section Unsupported Memory Blocks, added links to related application notes. In section User Clock Frequencies, the Refresh button is circled in the figure.

Revision History (continued)		
Revision	Date	Description
D	12/2021	<p>The following list of changes is made in revision D of the document:</p> <ul style="list-style-type: none"> Consolidated PolarFire content with content for SmartFusion 2, IGLOO 2, and RTG4 FPGAs. Described the Highlight Selected check box in section Hierarchical View. Updated screen shots in sections Normal Mode, Infinite Persistent Mode, Eye Mask, Signal Integrity, and Debug Fabric DDR IO Margin. Added the new SmartDebug Tcl command export_dds_training_data.
C	08/2021	<p>The following list of changes is made in revision C of the document:</p> <ul style="list-style-type: none"> Added Rescan Programmer. Added FHB Status. Updated MSS Register Access (PolarFire SoC) to add Import and Delete All button details. Added SmartDebug Tcl Commands.
B	04/2021	<p>The following list of changes is made in revision B of the document:</p> <ul style="list-style-type: none"> Added MSS Register Access (PolarFire SoC). Added Debug IOD. Updated Register Access.
A	11/2020	<p>The following list of changes is made in revision A of the document:</p> <ul style="list-style-type: none"> Added read/write access support to the XCVR registers. Added enhancements for the Two-Port LSRAM configured in the ECC mode.
8.0	06/2020	<p>The following list of changes is made in revision 8.0 of the document:</p> <ul style="list-style-type: none"> Added a note about SCB Read operations. Added embedded FlashPro6 for supported programmers. Added load and save button information in the Live Probes section. Added a note about PCIe SMARTBERT tests and unused lanes. Added information about the new Eye Mask feature. Added information about Record Actions in the Debug TRANSCEIVER window. Added information about the new Debug DDR IO Margin feature. Added information about Debug eNVM.

Microchip FPGA 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. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

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**

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at <https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>.

ISBN: 979-8-3371-1232-9

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications 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. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

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.

Microchip Devices Code Protection Feature

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

- 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, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- 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. Microchip is committed to continuously improving the code protection features of our products.