Libero® SoC v2021.2

SmartDebug User Guide for PolarFire® FPGAs

Introduction

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

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

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

Table of Contents

1. Getting Started with SmartDebug

SmartDebug enables you to use JTAG to interrogate and view embedded silicon features and device status. The most common flow for SmartDebug is:

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

For more information on how to use the debugger to gather the device status and to view the diagnostics, see Get device status and view diagnostics.

1.1 Use Models

SmartDebug can be run in the following modes:

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

1.1.1 Integrated Mode

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

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

1.1.2 Standalone Mode

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

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

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

1.1.3 Demo Mode

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

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

1.2 Creating a Standalone SmartDebug Project

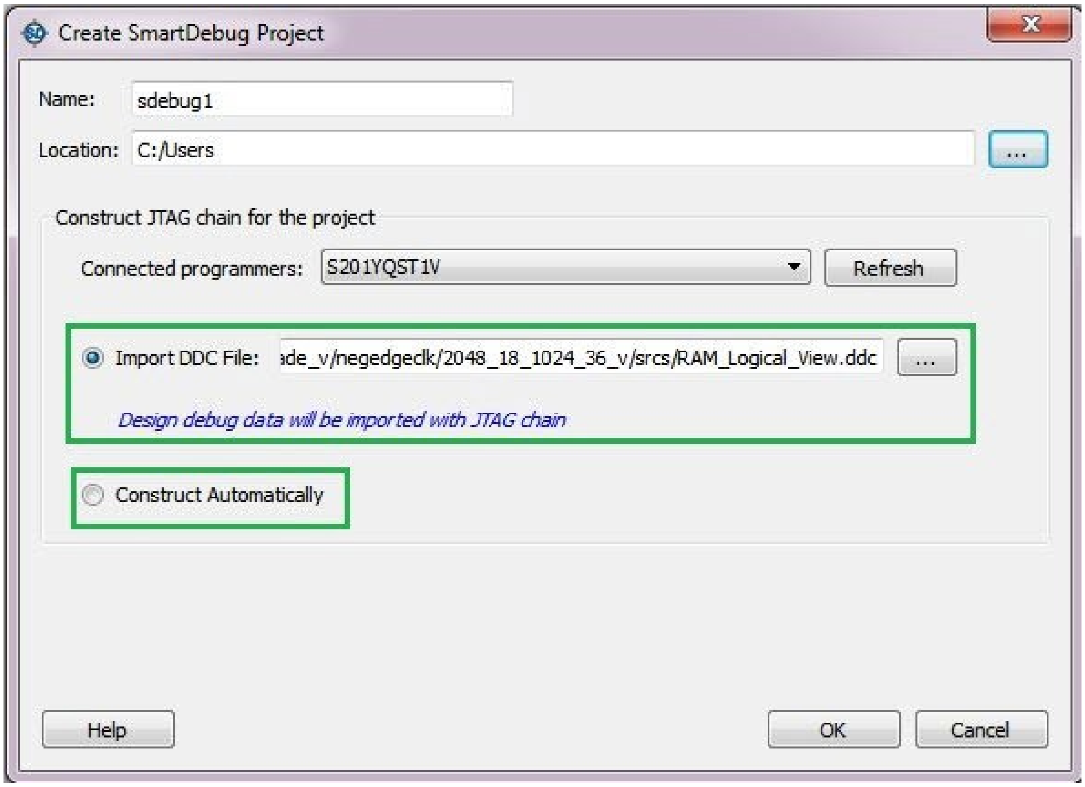
A standalone SmartDebug project can be configured in two ways:

- Import DDC files exported from Libero

- Construct Automatically

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

Figure 1-1. Create SmartDebug Project Dialog Box



Create SmartDebug Project

Name: sdebug1

Location: C:/Users

Construct JTAG chain for the project

Connected programmers: S201YQST1V

● Import DDC File: ade_v/negedgeclk/2048_18_1024_36_v/srcs/RAM_Logical_View.ddc

Design debug data will be imported with JTAG chain

○ Construct Automatically

1.2.1 Import DDC File (created from Libero)

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

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

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

1.2.2 Construct Automatically

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

1.2.3 Connected FlashPRO Programmers

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

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

See Also

- Programming Connectivity and Interface
- View Device Status

1.3 Configuring a Generic Device

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

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

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

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

2. SmartDebug User Interface

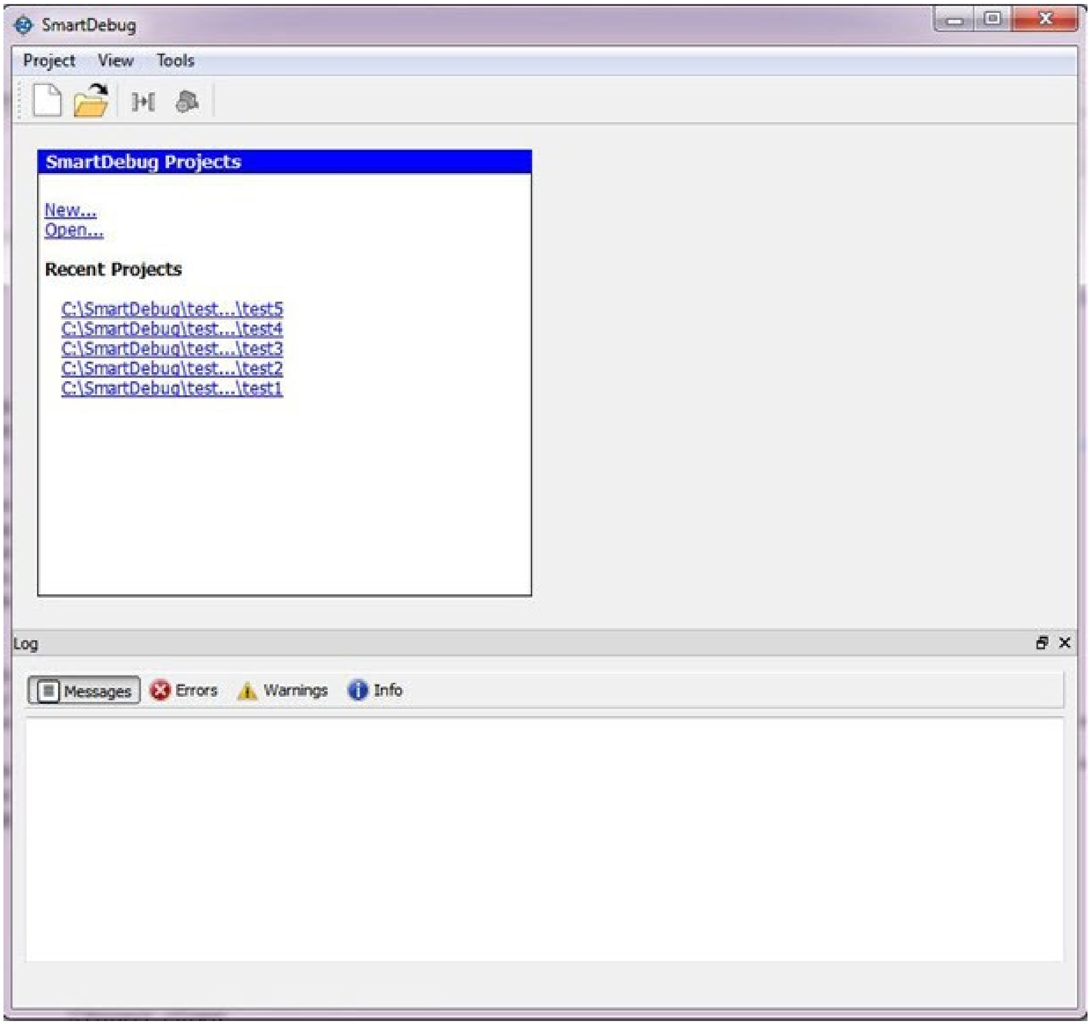
This topic introduces the basic elements and features of SmartDebug.

2.1 Standalone SmartDebug User Interface

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

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

Figure 2-1. Standalone SmartDebug Main Window



SmartDebug
Project
View
Tools
SmartDebug Projects
New...
Open...
Recent Projects
C:\SmartDebug\test...\test5
C:\SmartDebug\test...\test4
C:\SmartDebug\test...\test3
C:\SmartDebug\test...\test2
C:\SmartDebug\test...\test1
Log
Messages
Errors
Warnings
Info

2.1.1 Project Menu

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

Table 2-1. Project Menu

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

2.1.2 Log Window

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

Table 2-2. Log View Tabs

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

2.1.3 Tools Menu

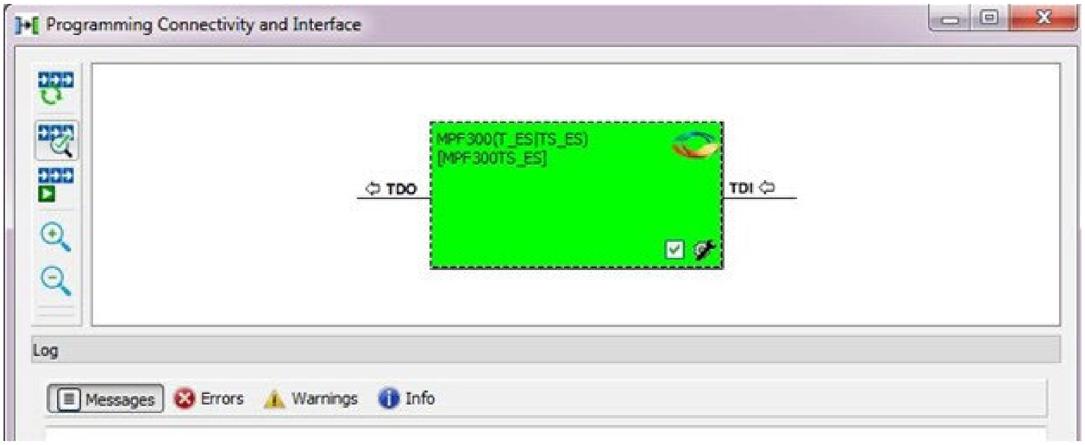
The Tools menu includes Programming Connectivity and Interface and Programmer Settings options, which are enabled after creating or opening a SmartDebug project.

For more information, see 2.2 Programming Connectivity and Interface.

2.2 Programming Connectivity and Interface

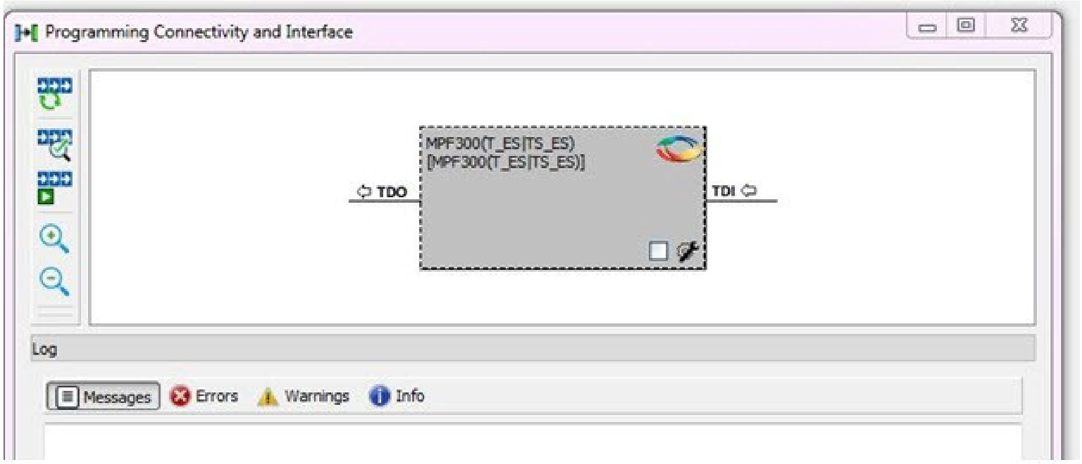
To open the Programming Connectivity and Interface dialog box, from the standalone SmartDebug Tools menu, choose **Programming Connectivity and Interface**. The Programming Connectivity and Interface dialog box displays the physical chain from TDI to TDO.

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



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

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



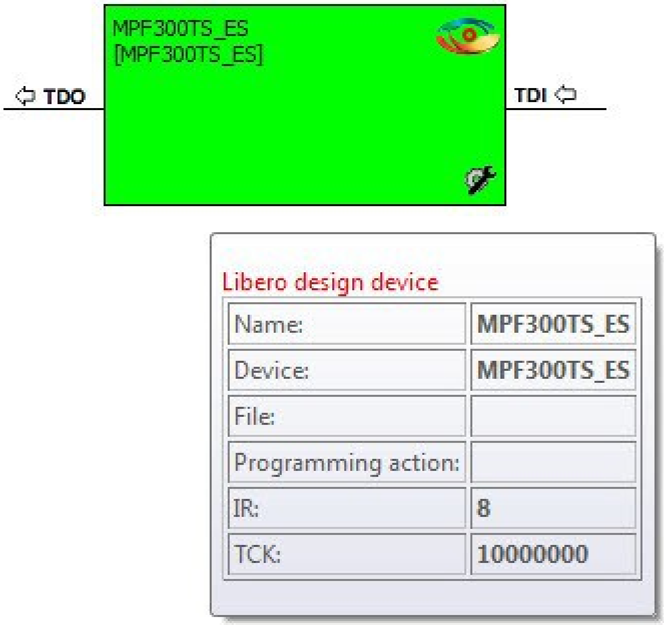
The Programming Connectivity and Interface dialog box includes the following actions:

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

2.2.1 Hover Information

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

Figure 2-4. Libero Design Device Information



Libero design device

Name:	MPF300TS_ES
Device:	MPF300TS_ES
File:	
Programming action:	
IR:	8
TCK:	10000000

The following table describes each Libero design device information option.

Table 2-3. Libero Design Device Information

Option	Description
Name	User-specified device name. This field indicates the unique name specified by the user in the Device Name field in Configure Device (right-click **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

The device within the chain has the following details:

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

2.2.3 Context Menu Options

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

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

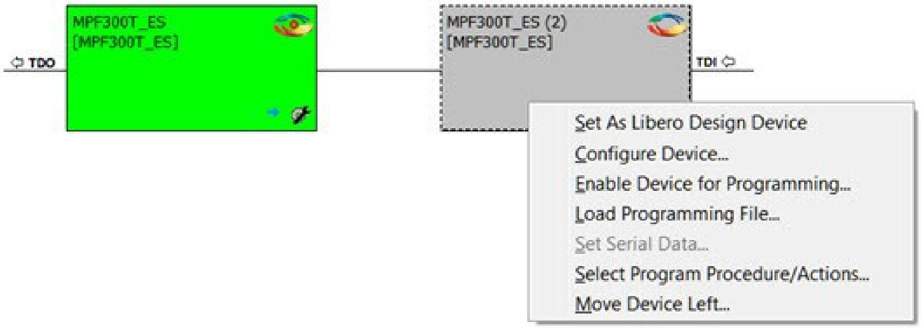


Table 2-4. Programming Connectivity and Interface - Device Context Menu Options

Option	Description
Set as Libero Design Device	The user needs to set Libero design device when there are multiple identical Libero design devices in the chain.
Configure Device	Ability to reconfigure the device. • 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.

...........continued

Option	Description
Enable Device for Programming	Select to enable the device for programming. Enabled devices are shown in green, and disabled devices are grayed out.
Load Programming File	Load the programming file for the selected device.
Select Programming Procedure/Actions	Option to select programming action/procedures for the devices connected in the chain. • Actions: List of programming actions for your device. • Procedures: Advanced option; enables you to customize the list of recommended and optional procedures for the selected action.
Import Debug Data from DDC File	Option to import debug data information from the DDC file. **Note:** This option is supported when SmartDebug is invoked in standalone mode. The DDC file selected for import into device must be created for a compatible device. When the DDC file is imported successfully, all current device debug data is removed and replaced with debug data from the imported DDC file. The JTAG Chain configuration from the imported DDC file is ignored in this option. If a programming file is already loaded into the device prior to importing debug data from the DDC file, the programming file content is replaced with the content of the DDC file (if programming file information is included in the DDC file).

2.2.4 Debug Context Save

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

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

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

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

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

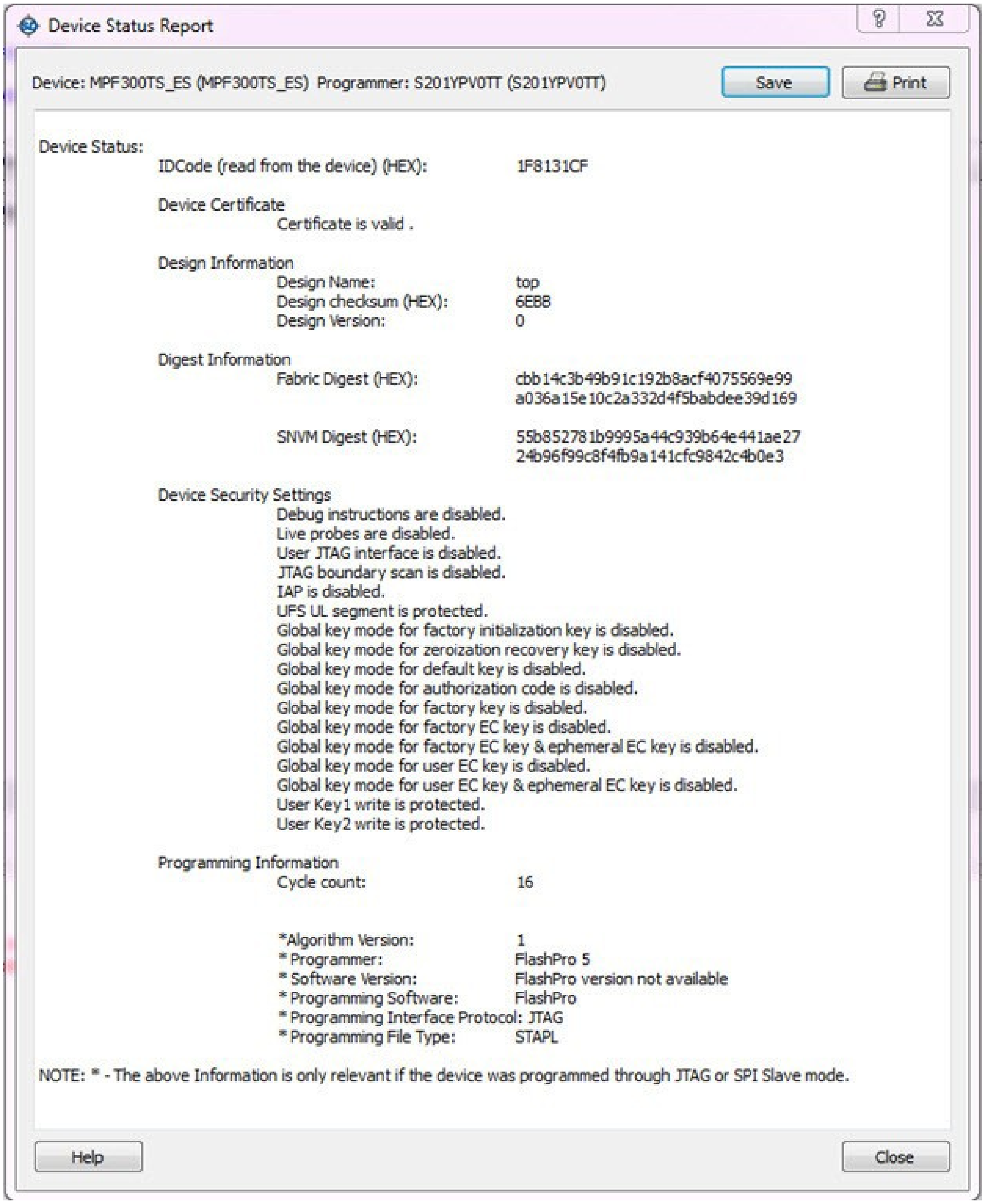
2.2.5 Selecting Devices for Debug

Standalone SmartDebug provides an option to select the devices connected in the JTAG chain for debug. 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

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 2-6. Device Status Report



```
Device: MPF300TS_ES (MPF300TS_ES)  Programmer: S201YPV0TT (S201YPV0TT)

Device Status:
          IDCode (read from the device) (HEX):            1F8131CF

          Device Certificate
                    Certificate is valid .

          Design Information
                    Design Name:                          top
                    Design checksum (HEX):                6EBB
                    Design Version:                       0

          Digest Information
                    Fabric Digest (HEX):                  cbb14c3b49b91c192b8acf4075569e99
                                                          a036a15e10c2a332d4f5babdee39d169

                    SNVM Digest (HEX):                    55b852781b9995a44c939b64e441ae27
                                                          24b96f99c8f4fb9a141cfc9842c4b0e3

          Device Security Settings
                    Debug instructions are disabled.
                    Live probes are disabled.
                    User JTAG interface is disabled.
                    JTAG boundary scan is disabled.
                    IAP is disabled.
                    UFS UL segment is protected.
                    Global key mode for factory initialization key is disabled.
                    Global key mode for zeroization recovery key is disabled.
                    Global key mode for default key is disabled.
                    Global key mode for authorization code is disabled.
                    Global key mode for factory key is disabled.
                    Global key mode for factory EC key is disabled.
                    Global key mode for factory EC key & ephemeral EC key is disabled.
                    Global key mode for user EC key is disabled.
                    Global key mode for user EC key & ephemeral EC key is disabled.
                    User Key1 write is protected.
                    User Key2 write is protected.

          Programming Information
                    Cycle count:                          16


                    *Algorithm Version:                   1
                    * Programmer:                         FlashPro 5
                    * Software Version:                   FlashPro version not available
                    * Programming Software:               FlashPro
                    * Programming Interface Protocol:     JTAG
                    * Programming File Type:              STAPL

NOTE: * - The above Information is only relevant if the device was programmed through JTAG or SPI Slave mode.
```

The following table describes the device status report information.

Table 2-5. Device Status Report Information

Information	Description
IdCode	IDCode read from the device under debug.
Device Certificate	Device certificate displays Family and Die information if the device certificate is installed on the device. If the device certificate is not installed on the device, a message indicating that the device certificate may not have been installed is shown.

...........continued

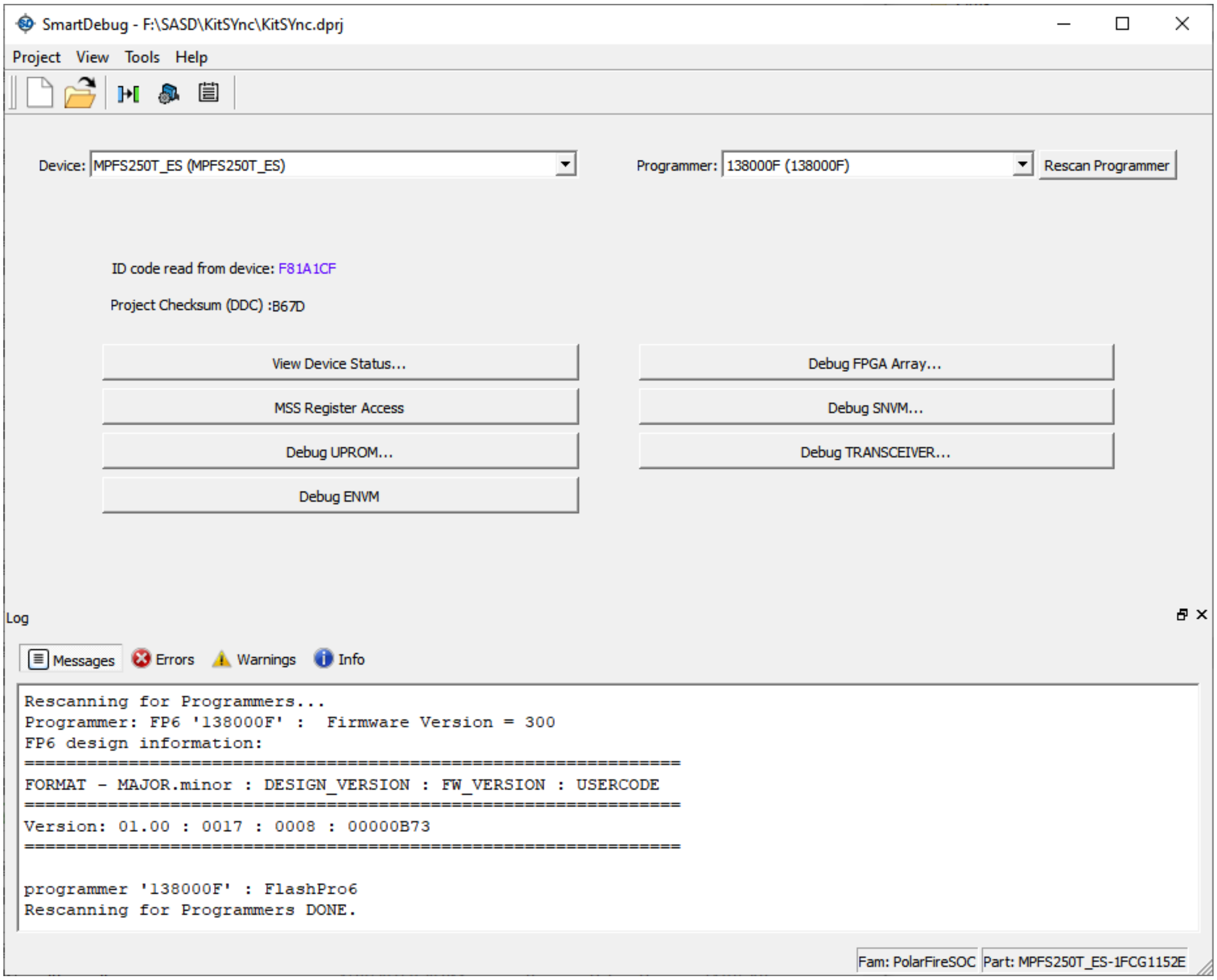
Information	Description
Design Information	Design Information displays the following: • Design Name • Design Checksum • Design Version
Digest Information	Digest Information displays Fabric Digest, sNVM Digest (if applicable) computed from the device during programming. sNVM Digest is shown when sNVM is used in the design.
Device Security Settings	Device Security Settings displays information about your security settings, including live probes, JTAG boundary scan, global key modes, and user keys.
Programming Information	Programming Information displays the following: • Cycle Count: The count is the number of times the device is programmed since it is out of factory reset. There is no limit to this count but a lower threshold would be around 2000 cycles. • Algorithm Version: The 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 IEEE532.

2.4 Rescan Programmer

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 2-7. Rescan Programmer



Device: MPFS250T_ES (MPFS250T_ES)

Programmer: 138000F (138000F)

ID code read from device: F81A1CF

Project Checksum (DDC) :B67D

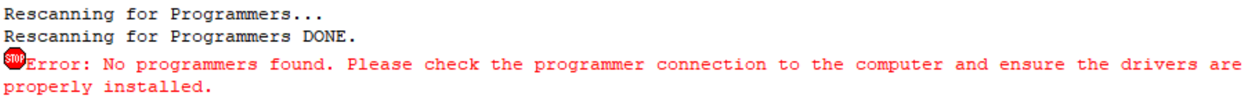
```
Rescanning for Programmers...
Programmer: FP6 '138000F' :  Firmware Version = 300
FP6 design information:
===============================================================
FORMAT - MAJOR.minor : DESIGN_VERSION : FW_VERSION : USERCODE
===============================================================
Version: 01.00 : 0017 : 0008 : 00000B73
===============================================================

programmer '138000F' : FlashPro6
Rescanning for Programmers DONE.
```

Fam: PolarFireSOC Part: MPFS250T_ES-1FCG1152E

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

Figure 2-8. 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

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

3.1 Debug FPGA Array

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

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

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

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

- Event Counter
- Frequency Monitor
- User Clock Frequencies

3.2 Hierarchical View

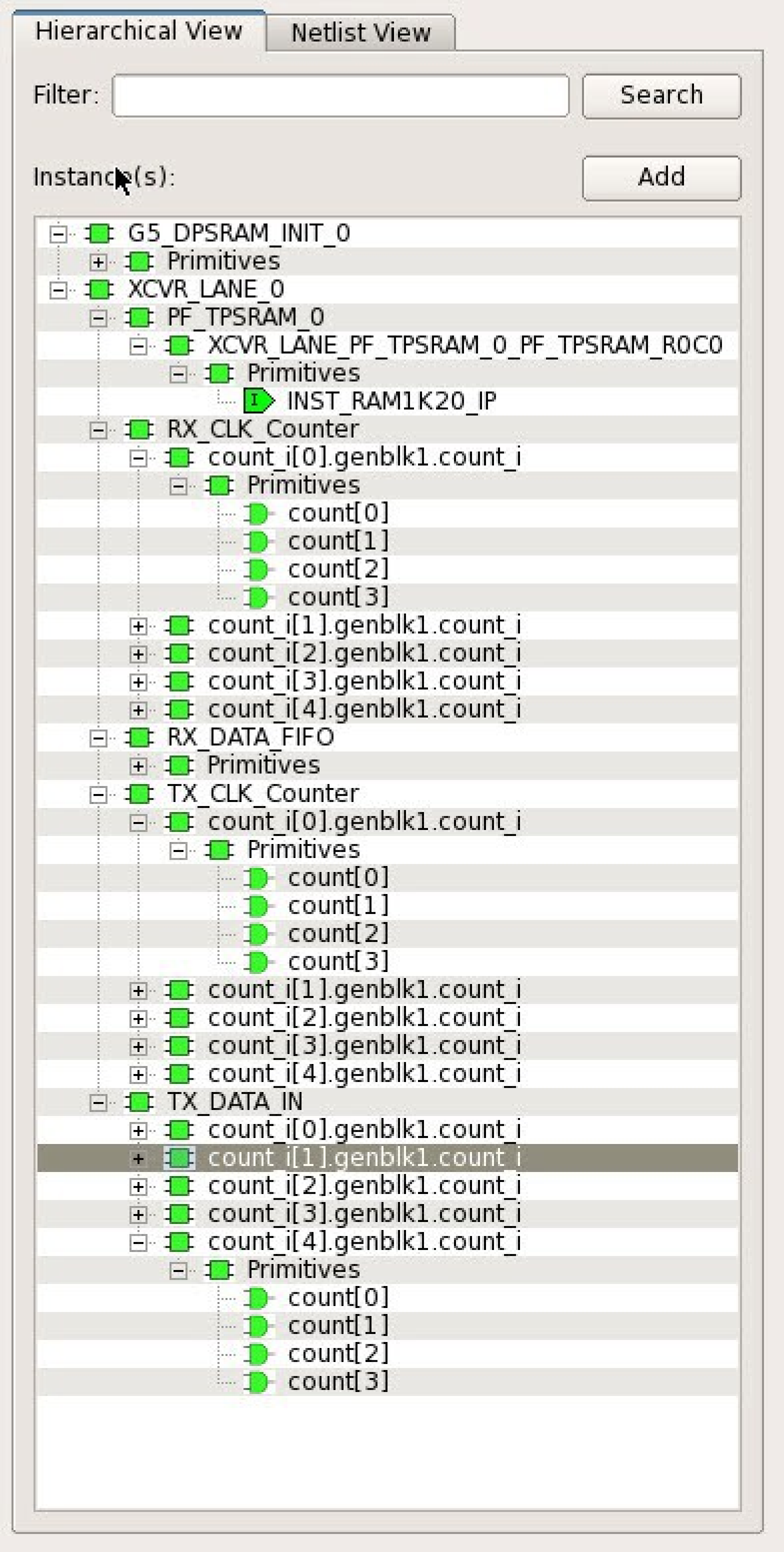
The Hierarchical View lets you view the instance level hierarchy of the design programmed on the device and select the signals to add to the Live Probes, Active Probes, and Probe Insertion tabs in the Debug FPGA Array dialog box. Logical and physical Memory Blocks can also be selected.

- **Filter**: In Live Probes, Active Probes, Memory Blocks, and the Probe Insertion UI, a search option is available in the Hierarchical View. You can use wildcard characters such as * or ? in the search column for wildcard matching. Probe points of leaf level instances resulting from a search pattern can only be added to Live Probes, Active Probes, and the Probe Insertion UI. You cannot add instances of search results in the Hierarchical View.
- **Instance(s)**: Displays the probe points available at the instance level.
- **Primitives**: Displays the lowest level of probe-able points in the hierarchy for the corresponding component such as leaf cells (hard macros on the device).

You can expand the hierarchy tree to see lower level logic. Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree.

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

Figure 3-1. Hierarchical View



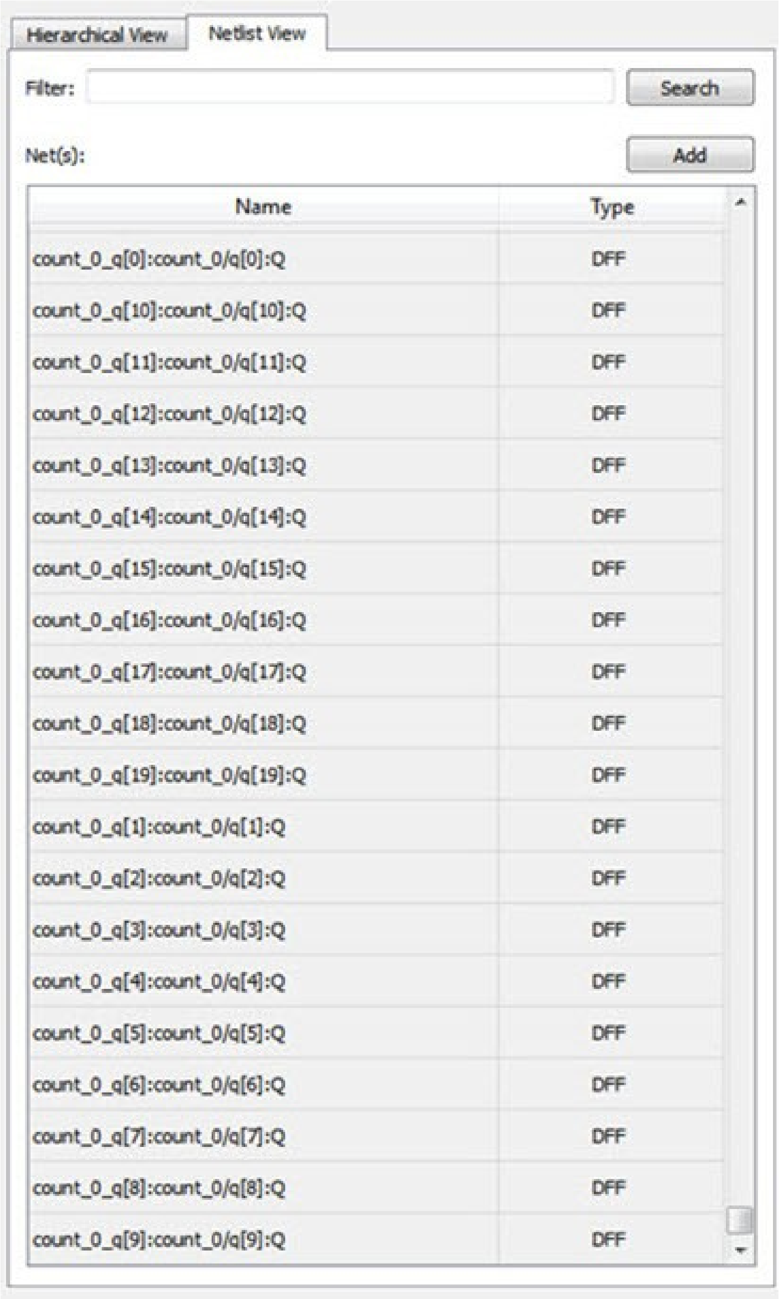
```
⊟ G5_DPSRAM_INIT_0
    ⊞ Primitives
⊟ XCVR_LANE_0
    ⊟ PF_TPSRAM_0
        ⊟ XCVR_LANE_PF_TPSRAM_0_PF_TPSRAM_R0C0
            ⊟ Primitives
                INST_RAM1K20_IP
    ⊟ RX_CLK_Counter
        ⊟ count_i[0].genblk1.count_i
            ⊟ Primitives
                count[0]
                count[1]
                count[2]
                count[3]
        ⊞ count_i[1].genblk1.count_i
        ⊞ count_i[2].genblk1.count_i
        ⊞ count_i[3].genblk1.count_i
        ⊞ count_i[4].genblk1.count_i
    ⊟ RX_DATA_FIFO
        ⊞ Primitives
    ⊟ TX_CLK_Counter
        ⊟ count_i[0].genblk1.count_i
            ⊟ Primitives
                count[0]
                count[1]
                count[2]
                count[3]
        ⊞ count_i[1].genblk1.count_i
        ⊞ count_i[2].genblk1.count_i
        ⊞ count_i[3].genblk1.count_i
        ⊞ count_i[4].genblk1.count_i
    ⊟ TX_DATA_IN
        ⊞ count_i[0].genblk1.count_i
        ⊞ count_i[1].genblk1.count_i
        ⊞ count_i[2].genblk1.count_i
        ⊞ count_i[3].genblk1.count_i
        ⊟ count_i[4].genblk1.count_i
            ⊟ Primitives
                count[0]
                count[1]
                count[2]
                count[3]
```

3.3 Netlist View

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

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

Figure 3-2. Netlist View



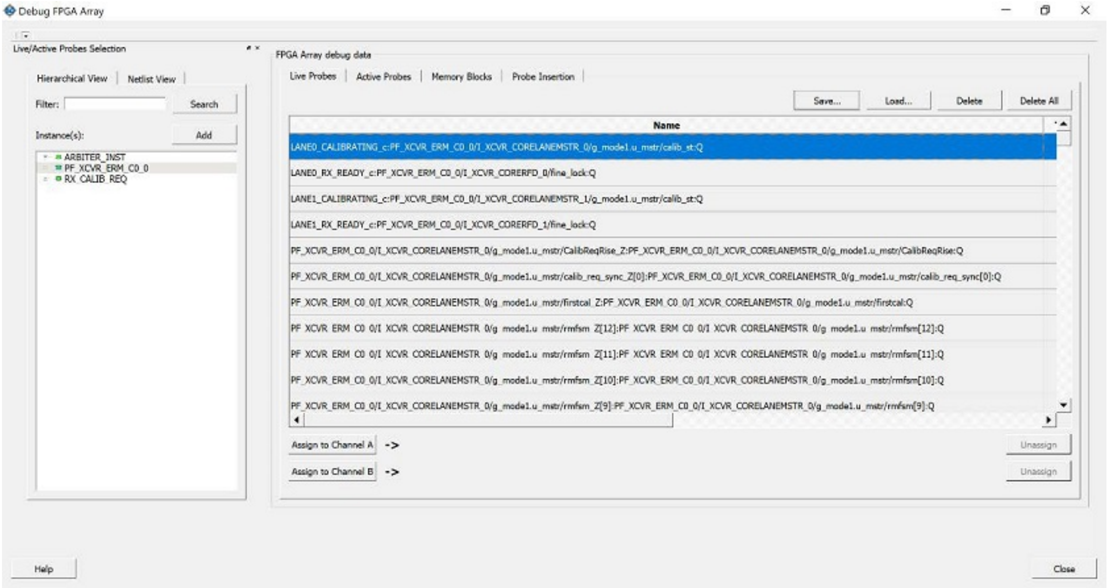
Name	Type
count_0_q[0]:count_0/q[0]:Q	DFF
count_0_q[10]:count_0/q[10]:Q	DFF
count_0_q[11]:count_0/q[11]:Q	DFF
count_0_q[12]:count_0/q[12]:Q	DFF
count_0_q[13]:count_0/q[13]:Q	DFF
count_0_q[14]:count_0/q[14]:Q	DFF
count_0_q[15]:count_0/q[15]:Q	DFF
count_0_q[16]:count_0/q[16]:Q	DFF
count_0_q[17]:count_0/q[17]:Q	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

3.4 Live Probes

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

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

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



Name
LANE0_CALIBRATING_c:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_st:Q
LANE0_RX_READY_c:PF_XCVR_ERM_C0_0/I_XCVR_CORERFD_0/fine_lock:Q
LANE1_CALIBRATING_c:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_1/g_mode1.u_mstr/calib_st:Q
LANE1_RX_READY_c:PF_XCVR_ERM_C0_0/I_XCVR_CORERFD_1/fine_lock:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/CalibReqRise_Z:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/CalibReqRise:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_req_sync_Z[0]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_req_sync[0]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/firstcal_Z:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/firstcal:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[12]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[12]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[11]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[11]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[10]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[10]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[9]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[9]:Q

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

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

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

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

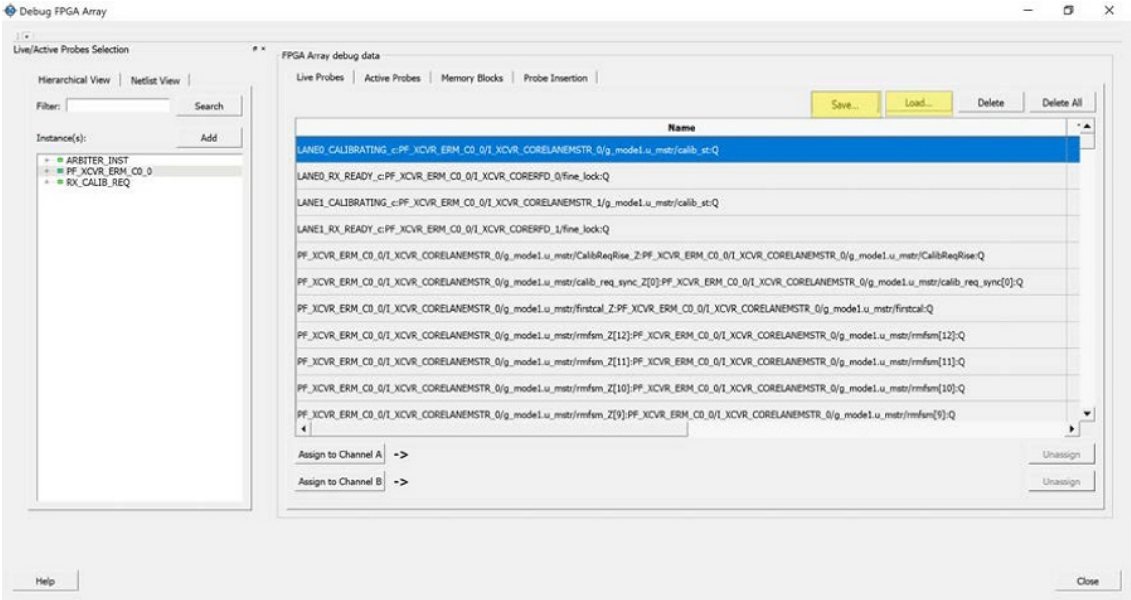
Click the **Unassign Channels** button to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

Note: Both probes can be assigned/unassigned independently.

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

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

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



Name
LANE0_CALIBRATING_c:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_st:Q
LANE0_RX_READY_c:PF_XCVR_ERM_C0_0/I_XCVR_CORERFD_0/fine_lock:Q
LANE1_CALIBRATING_c:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_1/g_mode1.u_mstr/calib_st:Q
LANE1_RX_READY_c:PF_XCVR_ERM_C0_0/I_XCVR_CORERFD_1/fine_lock:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/CalibReqRise_Z:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/CalibReqRise:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_req_sync_Z[0]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/calib_req_sync[0]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/firstcal_Z:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/firstcal:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[12]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[12]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[11]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[11]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[10]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[10]:Q
PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm_Z[9]:PF_XCVR_ERM_C0_0/I_XCVR_CORELANEMSTR_0/g_mode1.u_mstr/rmfsm[9]:Q

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

3.5 Active Probes

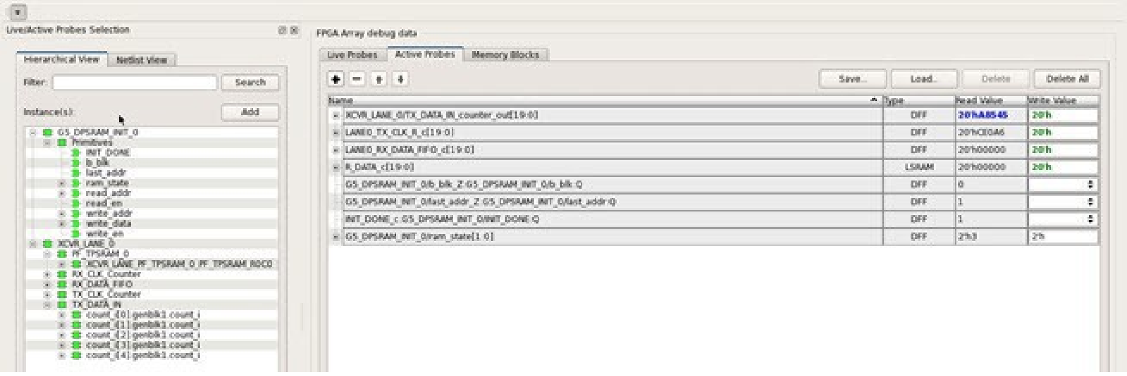
Active Probes is a design debug option to read and write to one or many probe points in the design through JTAG. Active probes makes use of the same circuitry used for programming the fabric. This feature is also non-intrusive and works asynchronously with the design clocks. In PolarFire, 18 DFFs are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

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

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

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

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

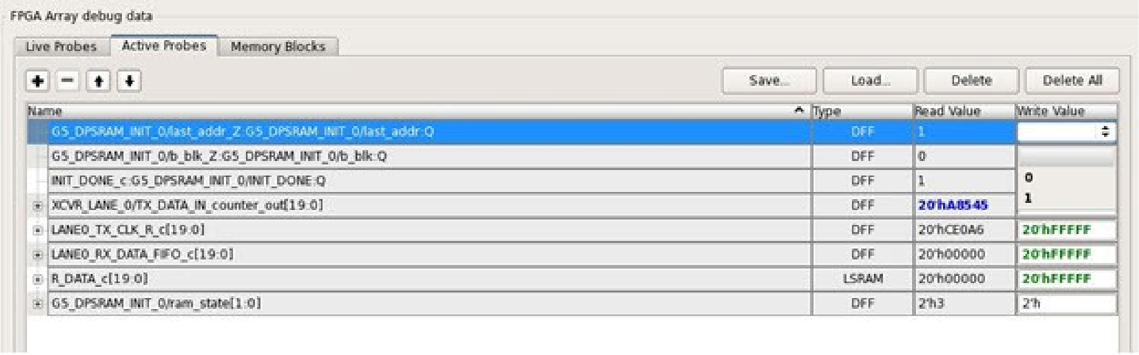


Name	Type	Read Value	Write Value
XCVR_LANE_0/TX_DATA_IN_counter_out[19:0]	DFF	20'hA8545	20'h
LANE0_TX_CLK_R_c[19:0]	DFF	20'hCE0A6	20'h
LANE0_RX_DATA_FIFO_c[19:0]	DFF	20'h00000	20'h
R_DATA_c[19:0]	LSRAM	20'h00000	20'h
G5_DPSRAM_INIT_0/b_blk_Z:G5_DPSRAM_INIT_0/b_blk:Q	DFF	0	
G5_DPSRAM_INIT_0/last_addr_Z:G5_DPSRAM_INIT_0/last_addr:Q	DFF	1	
INIT_DONE_c:G5_DPSRAM_INIT_0/INIT_DONE:Q	DFF	1	
G5_DPSRAM_INIT_0/ram_state[1:0]	DFF	2'h3	2'h

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

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

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



Name	Type	Read Value	Write Value
G5_DPSRAM_INIT_0/last_addr_Z:G5_DPSRAM_INIT_0/last_addr:Q	DFF	1	
G5_DPSRAM_INIT_0/b_blk_Z:G5_DPSRAM_INIT_0/b_blk:Q	DFF	0	
INIT_DONE_c:G5_DPSRAM_INIT_0/INIT_DONE:Q	DFF	1	
XCVR_LANE_0/TX_DATA_IN_counter_out[19:0]	DFF	20'hA8545	
LANE0_TX_CLK_R_c[19:0]	DFF	20'hCE0A6	20'hFFFFF
LANE0_RX_DATA_FIFO_c[19:0]	DFF	20'h00000	20'hFFFFF
R_DATA_c[19:0]	LSRAM	20'h00000	20'hFFFFF
G5_DPSRAM_INIT_0/ram_state[1:0]	DFF	2'h3	2'h

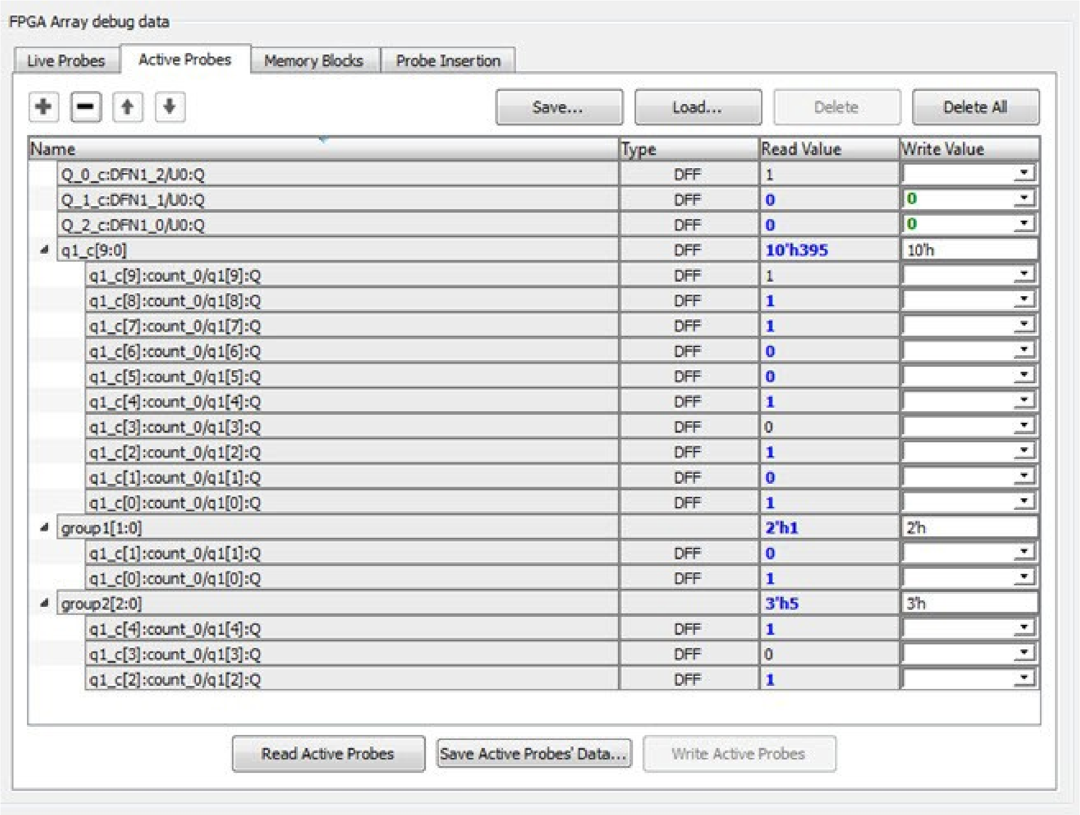
Active Probes in Demo Mode

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

3.6 Probe Grouping (Active Probes Only)

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

Figure 3-7. Active Probes Tab



Name	Type	Read Value	Write Value
Q_0_c:DFN1_2/U0:Q	DFF	1	
Q_1_c:DFN1_1/U0:Q	DFF	0	0
Q_2_c:DFN1_0/U0:Q	DFF	0	0
q1_c[9:0]	DFF	10'h395	10'h
q1_c[9]:count_0/q1[9]:Q	DFF	1	
q1_c[8]:count_0/q1[8]:Q	DFF	1	
q1_c[7]:count_0/q1[7]:Q	DFF	1	
q1_c[6]:count_0/q1[6]:Q	DFF	0	
q1_c[5]:count_0/q1[5]:Q	DFF	0	
q1_c[4]:count_0/q1[4]:Q	DFF	1	
q1_c[3]:count_0/q1[3]:Q	DFF	0	
q1_c[2]:count_0/q1[2]:Q	DFF	1	
q1_c[1]:count_0/q1[1]:Q	DFF	0	
q1_c[0]:count_0/q1[0]:Q	DFF	1	
group1[1:0]		2'h1	2'h
q1_c[1]:count_0/q1[1]:Q	DFF	0	
q1_c[0]:count_0/q1[0]:Q	DFF	1	
group2[2:0]		3'h5	3'h
q1_c[4]:count_0/q1[4]:Q	DFF	1	
q1_c[3]:count_0/q1[3]:Q	DFF	0	
q1_c[2]:count_0/q1[2]:Q	DFF	1	

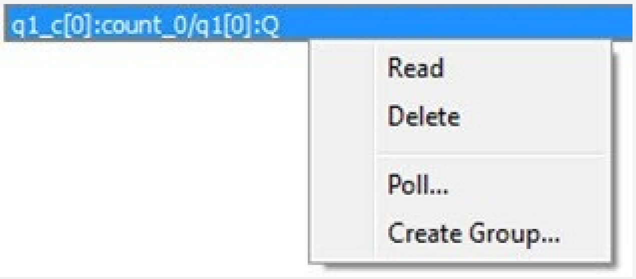
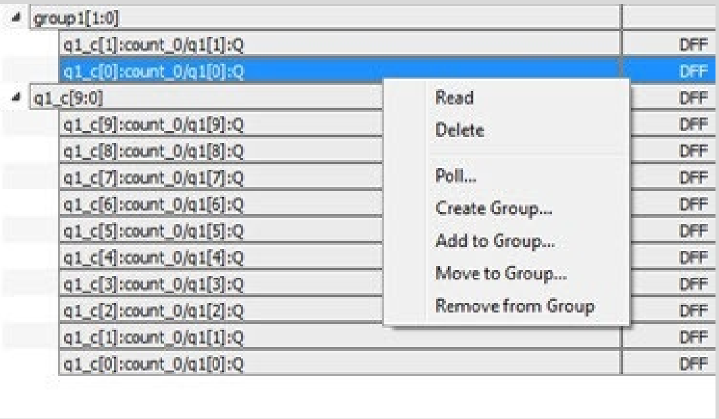
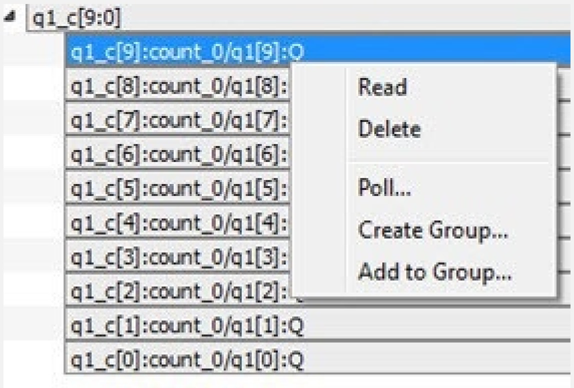
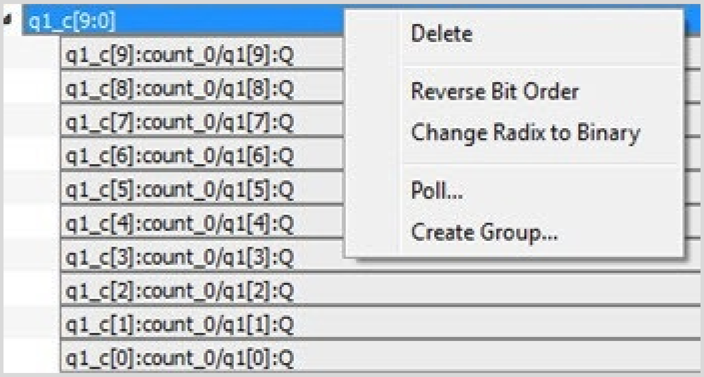
The Active Probes tab provides the following options for probe points that are added from the Hierarchical View/ Netlist View:

- Display bus name. An automatically generated bus name cannot be modified. Only custom bus names can be modified.
- Expand/collapse bus or probe group
- Move Up/Down the signal, bus, or probe group
- Save (Active Probes list)
- Load (already saved Active Probes list)
- Delete (applicable to a single probe point added to the Active Probes tab
- Delete All (deletes all probe points added to the Active Probes tab)
- In addition, the context (right-click) menu provides the following operations:
  - Create Group, Add/Move signals to Group, Remove signals from Group,
  - Ungroup
  - Reverse bit order, Change Radix for a bus or probe group
  - Read, Write, or Delete the signal or bus or probe group
- Green entries in the “Write Value” column indicate that the operation was successful.
- Blue entries in the “Read Value” column indicate values that have changed since the last read.

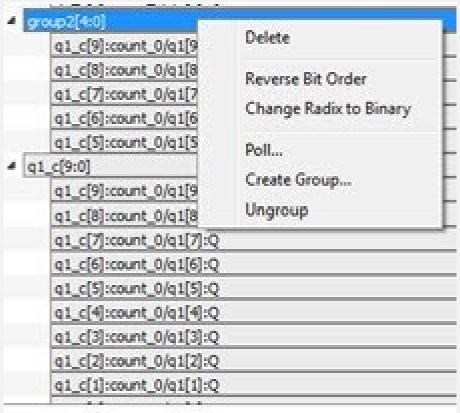
3.6.1 Context Menu of Probe Points Added to the Active Probes UI

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

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

Situation	Options	User Interface
For individual signals that are not part of a probe group or bus	• Read • Delete • Poll • Create Group	 q1_c[0]:count_0/q1[0]:Q Read Delete Poll... Create Group...
For individual signals in a probe group	• Read • Delete • Poll • Create Group • Add to Group • Move to Group • Remove from Group	 group1[1:0] q1_c[1]:count_0/q1[1]:Q DFF q1_c[0]:count_0/q1[0]:Q DFF q1_c[9:0] DFF q1_c[9]:count_0/q1[9]:Q DFF q1_c[8]:count_0/q1[8]:Q DFF q1_c[7]:count_0/q1[7]:Q DFF q1_c[6]:count_0/q1[6]:Q DFF q1_c[5]:count_0/q1[5]:Q DFF q1_c[4]:count_0/q1[4]:Q DFF q1_c[3]:count_0/q1[3]:Q DFF q1_c[2]:count_0/q1[2]:Q DFF q1_c[1]:count_0/q1[1]:Q DFF q1_c[0]:count_0/q1[0]:Q DFF Read Delete Poll... Create Group... Add to Group... Move to Group... Remove from Group
For individual signals in a bus	• Read • Delete • Poll • Create Group • Add to Group	 q1_c[9:0] q1_c[9]:count_0/q1[9]:Q q1_c[8]:count_0/q1[8]: q1_c[7]:count_0/q1[7]: q1_c[6]:count_0/q1[6]: q1_c[5]:count_0/q1[5]: q1_c[4]:count_0/q1[4]: q1_c[3]:count_0/q1[3]: q1_c[2]:count_0/q1[2]: q1_c[1]:count_0/q1[1]:Q q1_c[0]:count_0/q1[0]:Q Read Delete Poll... Create Group... Add to Group...
For a bus	• Delete • Reverse Bit Order • Change Radix to Binary • Poll • Create Group	 q1_c[9:0] q1_c[9]:count_0/q1[9]:Q q1_c[8]:count_0/q1[8]:Q q1_c[7]:count_0/q1[7]:Q q1_c[6]:count_0/q1[6]:Q q1_c[5]:count_0/q1[5]:Q q1_c[4]:count_0/q1[4]:Q q1_c[3]:count_0/q1[3]:Q q1_c[2]:count_0/q1[2]:Q q1_c[1]:count_0/q1[1]:Q q1_c[0]:count_0/q1[0]:Q Delete Reverse Bit Order Change Radix to Binary Poll... Create Group...

...........continued

Situation	Options	User Interface
For a probe group	• Delete • Reverse Bit Order • Change Radix to Binary • Poll • Create Group • Ungroup	 group2[4:0] q1_c[9]:count_0/q1[9 q1_c[8]:count_0/q1[8 q1_c[7]:count_0/q1[7 q1_c[6]:count_0/q1[6 q1_c[5]:count_0/q1[5 q1_c[9:0] q1_c[9]:count_0/q1[9 q1_c[8]:count_0/q1[8 q1_c[7]:count_0/q1[7]:Q q1_c[6]:count_0/q1[6]:Q q1_c[5]:count_0/q1[5]:Q q1_c[4]:count_0/q1[4]:Q q1_c[3]:count_0/q1[3]:Q q1_c[2]:count_0/q1[2]:Q q1_c[1]:count_0/q1[1]:Q Delete Reverse Bit Order Change Radix to Binary Poll... Create Group... Ungroup

3.6.2 Differences Between a Bus and a Probe Group

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

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

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

The following operations are not allowed in a bus.

- **Move to Group**: Moving a signal to a probe group
- **Remove from Group**: Removing a signal from a probe group

3.7 Memory Blocks

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

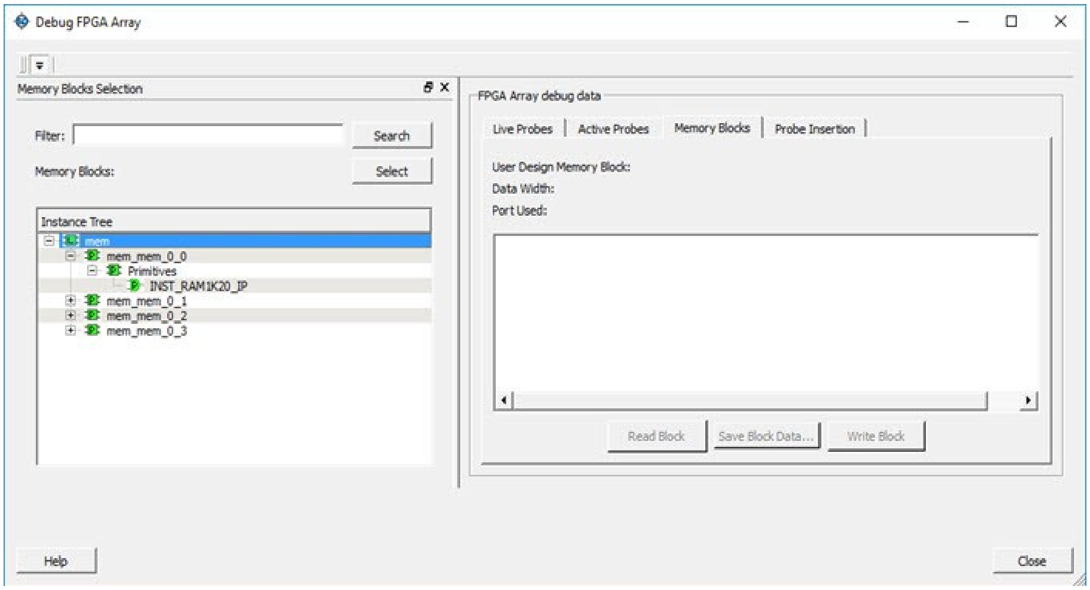
The Memory Blocks tab in the Debug FPGA Array dialog box shows the hierarchical view of all memory blocks in the design. The depth and width of blocks shown in the logical view are determined by the user in SmartDesign, RTL, or IP cores using memory blocks.

Notes:

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

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

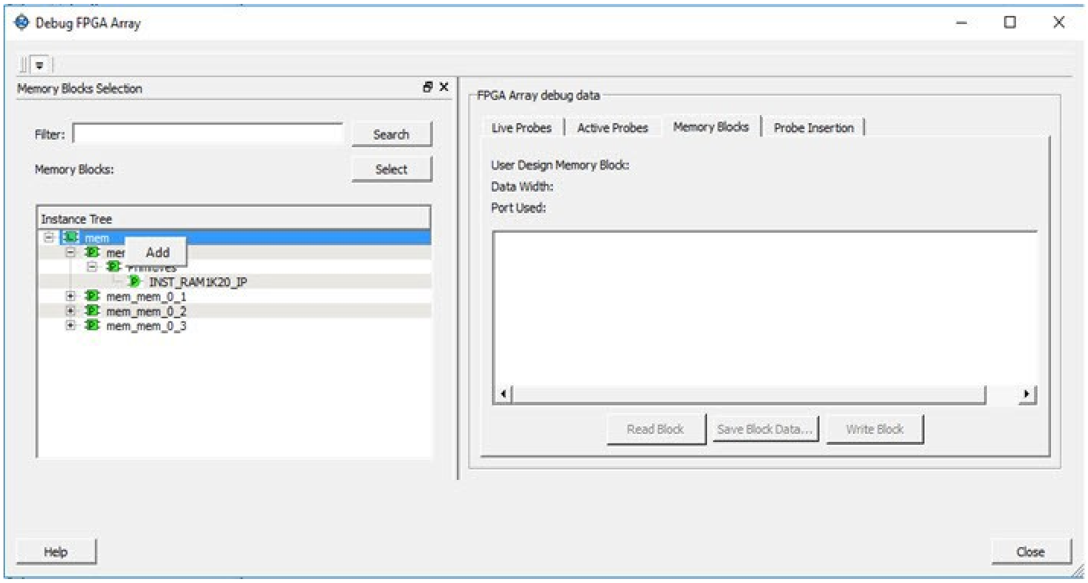
Figure 3-8. Memory Blocks Tab - Hierarchical View



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

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

  Figure 3-9. Adding a Memory Block



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

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

3.7.1 Memory Block Fields

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

User Design Memory Block

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

Data Width

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

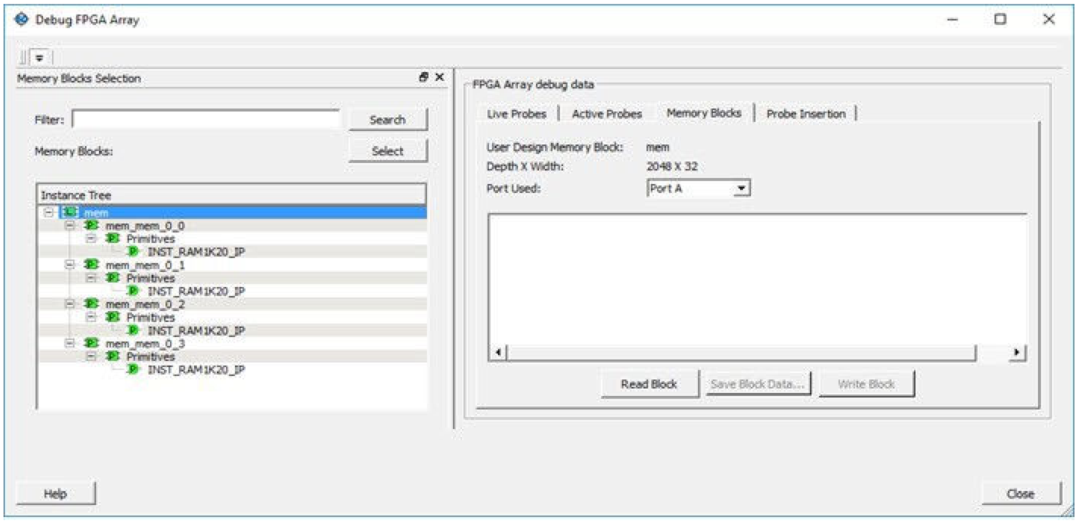
Port Used

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

Memory Block Views

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

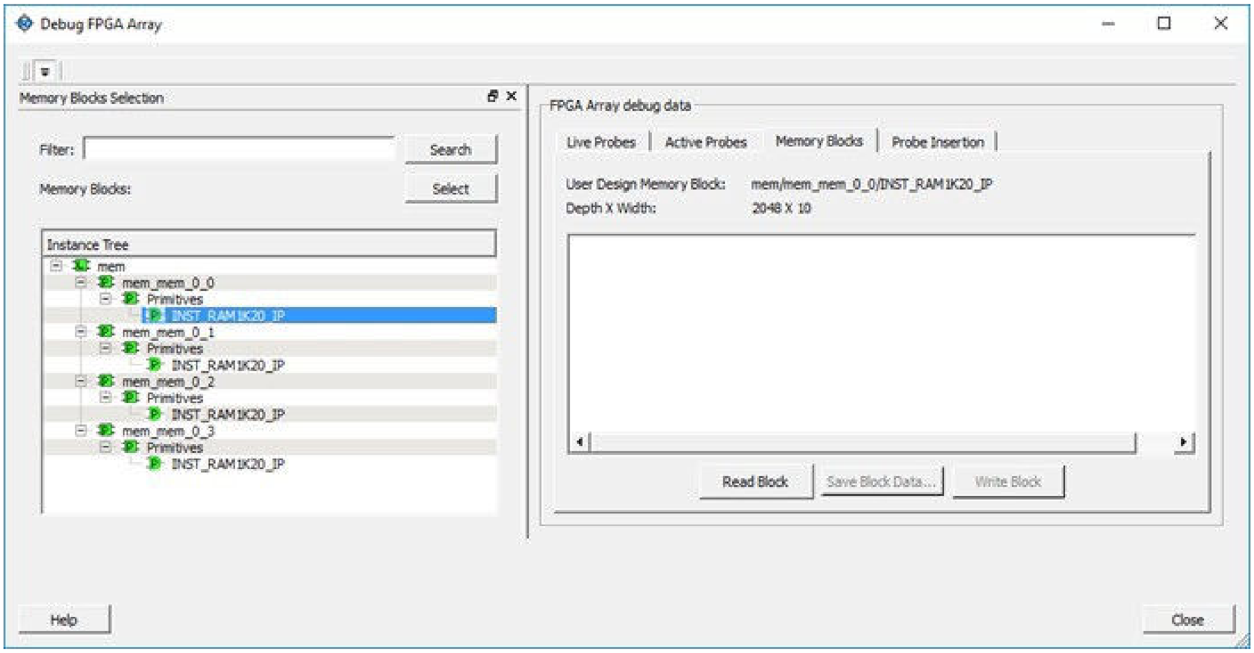
Figure 3-10. Memory Blocks Tab Fields for Logical Block View



User Design Memory Block: mem
Depth X Width: 2048 X 32
Port Used: Port A

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



User Design Memory Block: mem/mem_mem_0_0/INST_RAM1K20_IP
Depth X Width: 2048 X 10

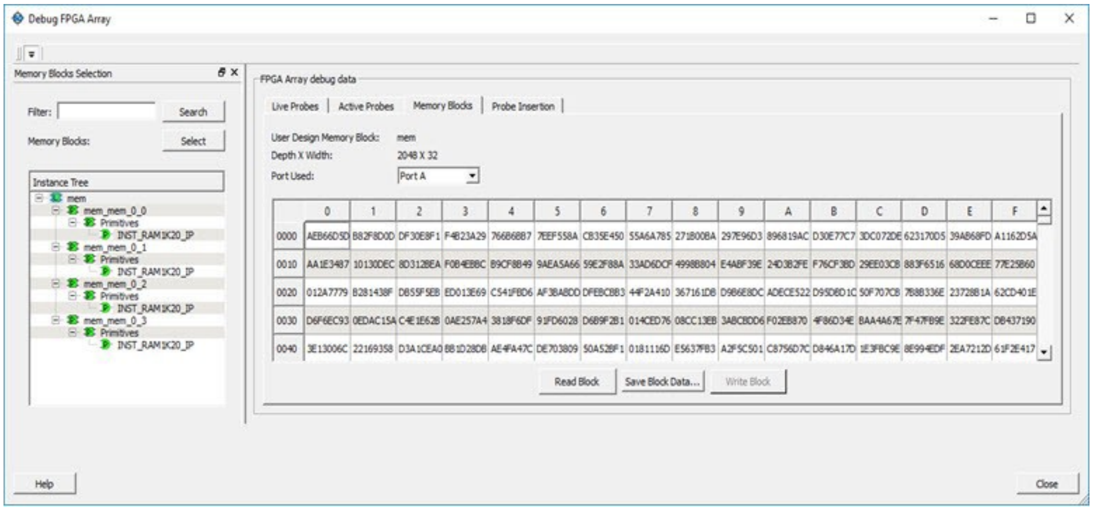
3.7.2 Read Block

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

Logical Block Read

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

Figure 3-12. Logical Block Read



User Design Memory Block: mem
Depth X Width: 2048 X 32
Port Used: Port A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	AEB66D5D	B82F8D0D	DF30E8F1	F4B23A29	766B6BB7	7EEF558A	CB35E450	55A6A785	271B00BA	297E96D3	896819AC	D30E77C7	3DC072DE	623170D5	39AB68FD	A1162D5A
0010	AA1E3487	10130DEC	8D312BEA	F0B4EBBC	B9CF8B49	9AEA5A66	59E2F88A	33AD6DCF	4998B804	E4ABF39E	24D3B2FE	F76CF38D	29EE03CB	883F6516	68D0CEEE	77E25B60
0020	012A7779	B281438F	DB55F5EB	ED013E69	C541FBD6	AF3BABDD	DFEBCBB3	44F2A410	367161DB	D9B6E8DC	ADECE522	D95D8D1C	50F707CB	7B8B336E	23728B1A	62CD401E
0030	D6F6EC93	0EDAC15A	C4E1E62B	0AE257A4	3818F6DF	91FD6028	D6B9F2B1	014CED76	08CC13EB	3ABCBDD6	F02EB870	4F86D34E	BAA4A67E	7F47FB9E	322FE87C	DB437190
0040	3E13006C	22169358	D3A1CEA0	8B1D28DB	AE4FA47C	DE703809	50A528F1	0181116D	E5637FB3	A2F5C501	C8756D7C	D846A17D	1E3FBC9E	8E994EDF	2EA7212D	61F2E417

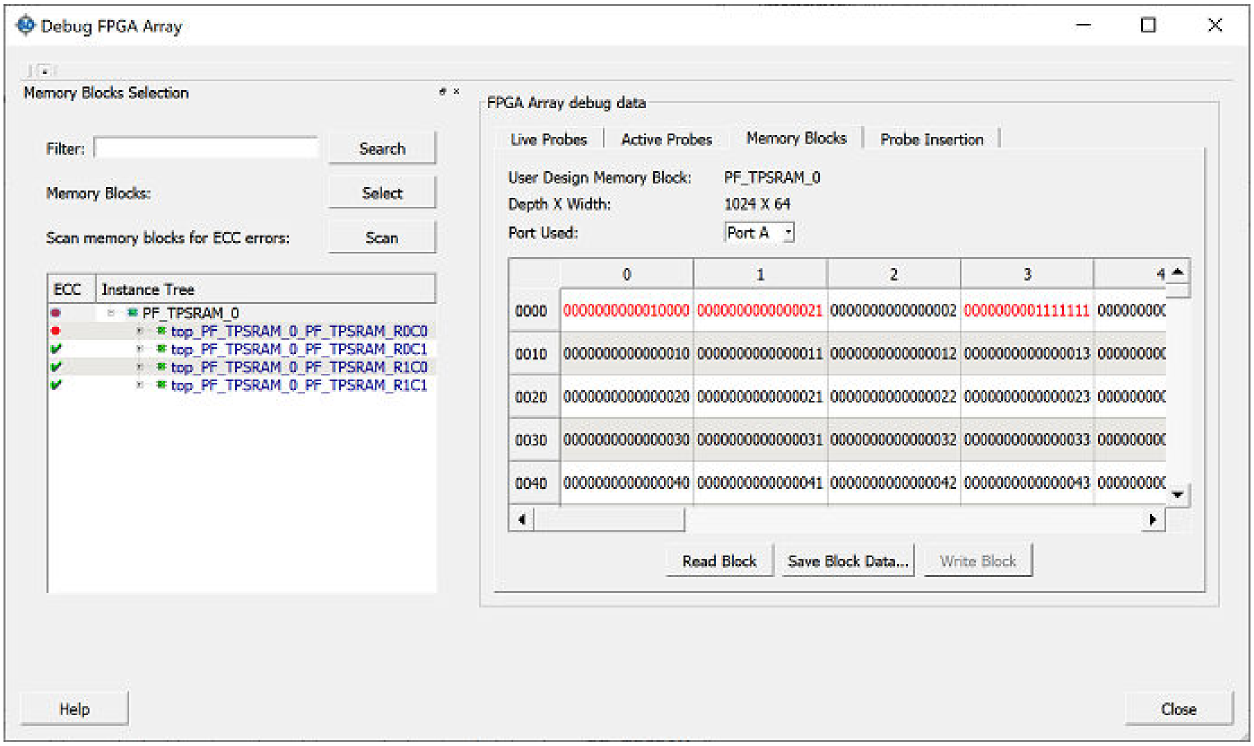
The data shown is in Hexadecimal format. In the above figure, data width is 32. Because each hexadecimal character has 4 bits of information, you can see 8 characters corresponding to 32 bits. Each row has 16 locations (shown in the column headers) which are numbered in hexadecimal from 0 to F.

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

Logical Block Read for ECC Enabled Blocks

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

Figure 3-13. Logical Block Read - ECC Enabled



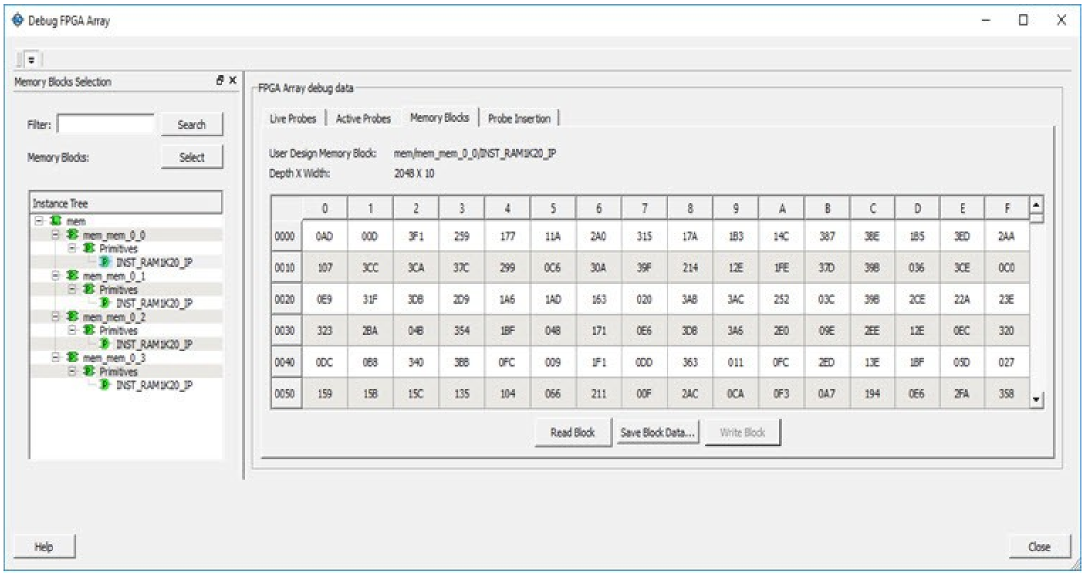
User Design Memory Block: PF_TPSRAM_0
Depth X Width: 1024 X 64
Port Used: Port A

	0	1	2	3	4
0000	0000000000010000	0000000000000021	0000000000000002	0000000001111111	00000000
0010	0000000000000010	0000000000000011	0000000000000012	0000000000000013	00000000
0020	0000000000000020	0000000000000021	0000000000000022	0000000000000023	00000000
0030	0000000000000030	0000000000000031	0000000000000032	0000000000000033	00000000
0040	0000000000000040	0000000000000041	0000000000000042	0000000000000043	00000000

Physical Block Read

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

Figure 3-14. Physical Block Read



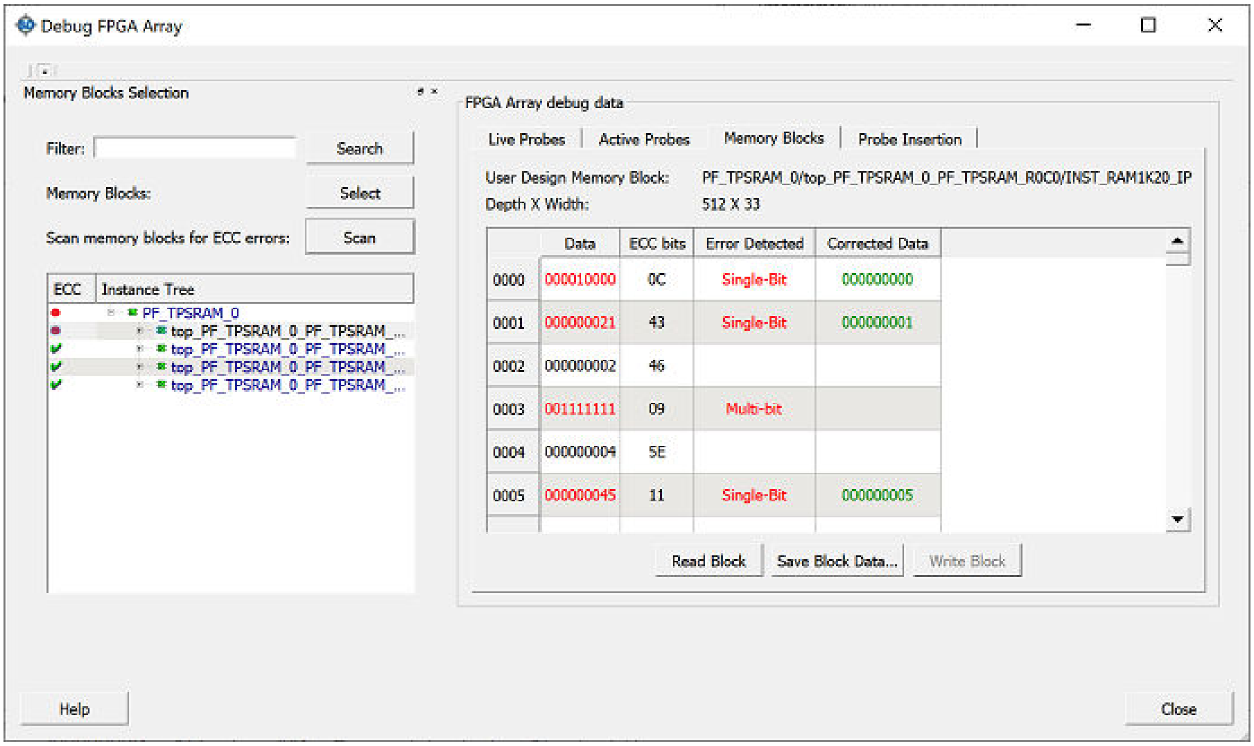
User Design Memory Block: mem/mem_mem_0_0/INST_RAM1K20_IP
Depth X Width: 2048 X 10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0AD	00D	3F1	259	177	11A	2A0	315	17A	1B3	14C	387	38E	1B5	3ED	2AA
0010	107	3CC	3CA	37C	299	0C6	30A	39F	214	12E	1FE	37D	39B	036	3CE	0C0
0020	0E9	31F	3DB	2D9	1A6	1AD	163	020	3A8	3AC	252	03C	39B	2CE	22A	23E
0030	323	2BA	04B	354	1BF	048	171	0E6	3D8	3A6	2E0	09E	2EE	12E	0EC	320
0040	0DC	0B8	340	3BB	0FC	009	1F1	0DD	363	011	0FC	2ED	13E	1BF	05D	027
0050	159	15B	15C	135	104	066	211	00F	2AC	0CA	0F3	0A7	194	0E6	2FA	358

Physical Block Read for ECC Enabled Blocks

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

Figure 3-15. Physical Block Read - ECC Enabled



User Design Memory Block: PF_TPSRAM_0/top_PF_TPSRAM_0_PF_TPSRAM_R0C0/INST_RAM1K20_IP
Depth X Width: 512 X 33

	Data	ECC bits	Error Detected	Corrected Data
0000	000010000	0C	Single-Bit	000000000
0001	000000021	43	Single-Bit	000000001
0002	000000002	46		
0003	001111111	09	Multi-bit	
0004	000000004	5E		
0005	000000045	11	Single-Bit	000000005

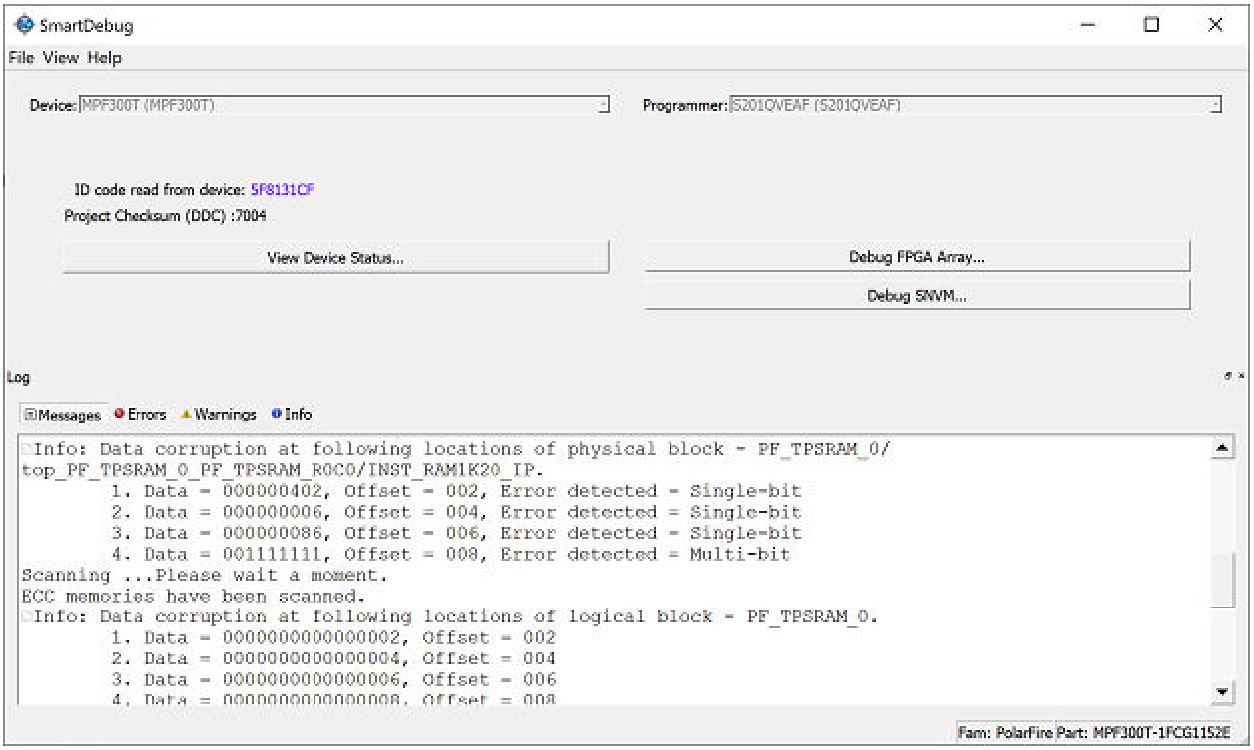
The following table describes the columns displayed in the physical block view.

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

Column	Description
Data	33-bit physical block data offset value.
ECC Bits	7-bit ECC value.
Error Detected	Displays identified error type. The error type value displayed can be either **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 in to the device. Right click on the corrected data to view an option to copy the data automatically to the **Data** cell.

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

Figure 3-16. Log Info



Device: MPF300T (MPF300T)

Programmer: S201QVEAF (S201QVEAF)

ID code read from device: 5F8131CF

Project Checksum (DDC) :7004

```
Info: Data corruption at following locations of physical block - PF_TPSRAM_0/
top_PF_TPSRAM_0_PF_TPSRAM_R0C0/INST_RAM1K20_IP.
        1. Data = 000000402, Offset = 002, Error detected = Single-bit
        2. Data = 000000006, Offset = 004, Error detected = Single-bit
        3. Data = 000000086, Offset = 006, Error detected = Single-bit
        4. Data = 001111111, Offset = 008, Error detected = Multi-bit
Scanning ...Please wait a moment.
ECC memories have been scanned.
Info: Data corruption at following locations of logical block - PF_TPSRAM_0.
        1. Data = 0000000000000002, Offset = 002
        2. Data = 0000000000000004, Offset = 004
        3. Data = 0000000000000006, Offset = 006
        4. Data = 0000000000000008, Offset = 008
```

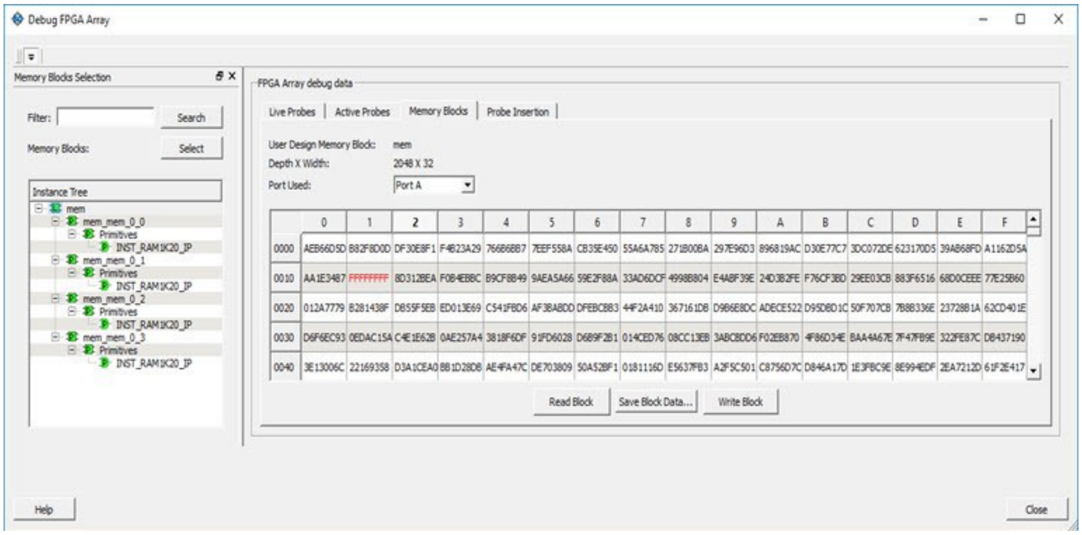
Fam: PolarFire Part: MPF300T-1FCG1152E

3.7.3 Write Block

Logical Block Write

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

Figure 3-17. Logical Block Write



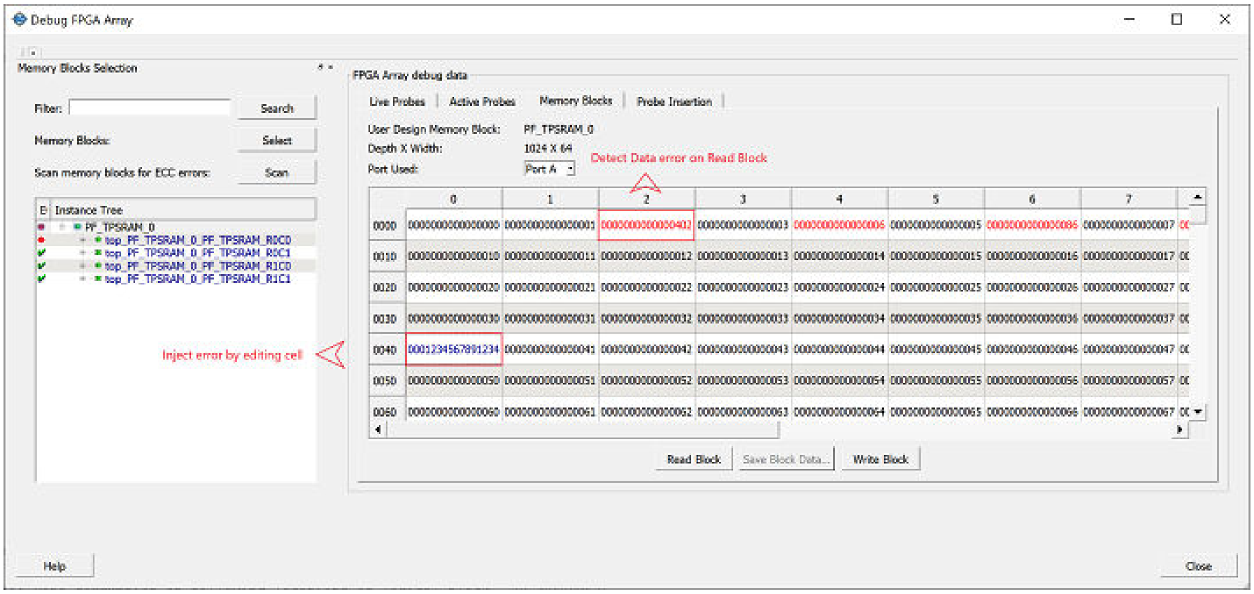
User Design Memory Block: mem
Depth X Width: 2048 X 32
Port Used: Port A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	AEB66D5D	B82F8D0D	DF30E8F1	F4B23A29	766B6BB7	7EEF558A	CB35E450	55A6A785	271B00BA	297E96D3	896819AC	D30E77C7	3DC072DE	623170D5	39AB68FD	A1162D5A
0010	AA1E3487	FFFFFFFF	8D312BEA	F0B4EBBC	B9CF8B49	9AEA5A66	59E2F88A	33AD6DCF	4998B804	E4ABF39E	24D3B2FE	F76CF3BD	29EE03CB	883F6516	68D0CEEE	77E25B60
0020	012A7779	B281438F	DB55F5EB	ED013E69	C541FBD6	AF3BABDD	DFEBCBB3	44F2A410	367161DB	D9B6E8DC	ADECE522	D95D8D1C	50F707CB	7B8B336E	23728B1A	62CD401E
0030	D6F6EC93	0EDAC15A	C4E1E62B	0AE257A4	3818F6DF	91FD6028	D6B9F2B1	014CED76	08CC13EB	3ABC8DD6	F02EB870	4F86D34E	BAA4A67E	7F47FB9E	322FE87C	DB437190
0040	3E13006C	22169358	D3A1CEA0	8B1D28D8	AE4FA47C	DE703809	50A52BF1	0181116D	E5637FB3	A2F5C501	C8756D7C	D846A17D	1E3FBC9E	8E994EDF	2EA7212D	61F2E417

Logical Block Write for ECC Enabled Blocks

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

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



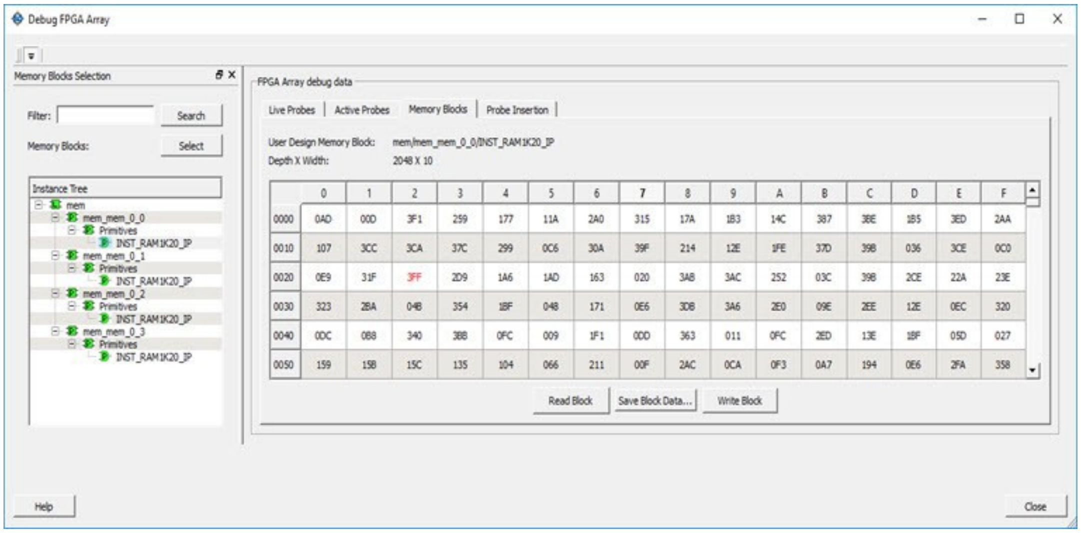
User Design Memory Block: PF_TPSRAM_0
Depth X Width: 1024 X 64
Port Used: Port A

	0	1	2	3	4	5	6	7
0000	0000000000000000	0000000000000001	0000000000000402	0000000000000003	0000000000000006	0000000000000005	0000000000000086	0000000000000007
0010	0000000000000010	0000000000000011	0000000000000012	0000000000000013	0000000000000014	0000000000000015	0000000000000016	0000000000000017
0020	0000000000000020	0000000000000021	0000000000000022	0000000000000023	0000000000000024	0000000000000025	0000000000000026	0000000000000027
0030	0000000000000030	0000000000000031	0000000000000032	0000000000000033	0000000000000034	0000000000000035	0000000000000036	0000000000000037
0040	0001234567891234	0000000000000041	0000000000000042	0000000000000043	0000000000000044	0000000000000045	0000000000000046	0000000000000047
0050	0000000000000050	0000000000000051	0000000000000052	0000000000000053	0000000000000054	0000000000000055	0000000000000056	0000000000000057
0060	0000000000000060	0000000000000061	0000000000000062	0000000000000063	0000000000000064	0000000000000065	0000000000000066	0000000000000067

Physical Block Write

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

Figure 3-19. Physical Block Write



User Design Memory Block: mem/mem_mem_0_0/INST_RAM1K20_IP
Depth X Width: 2048 X 10

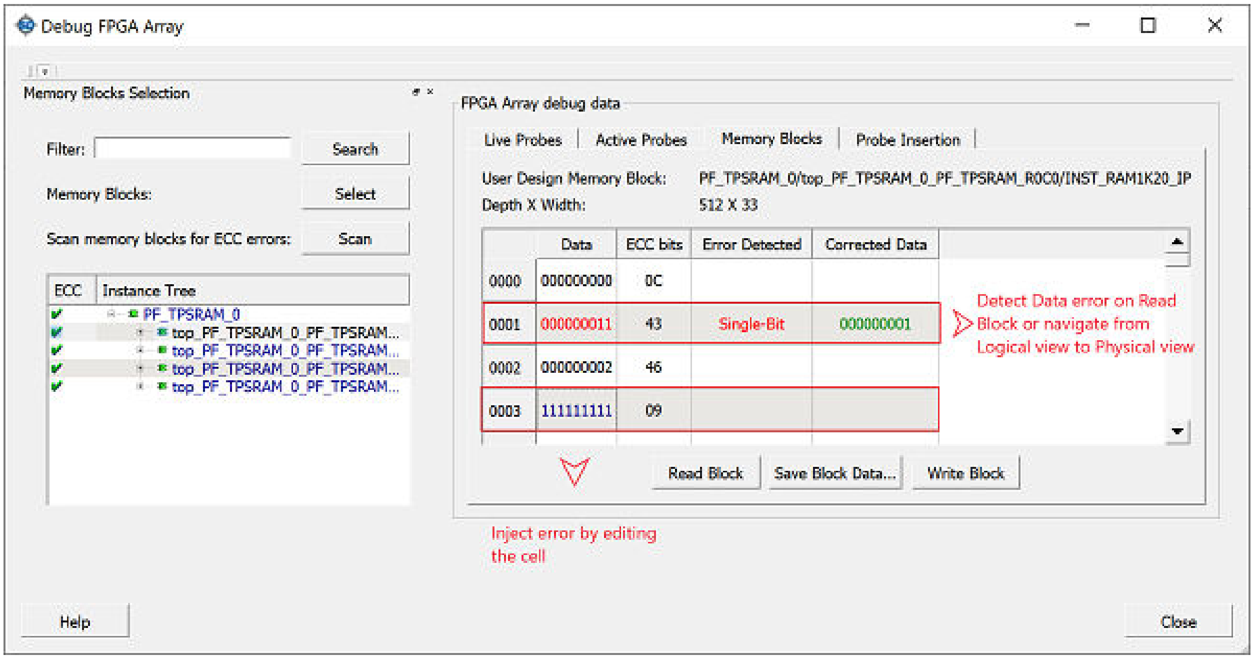
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0AD	00D	3F1	259	177	11A	2A0	315	17A	1B3	14C	387	3BE	1B5	3ED	2AA
0010	107	3CC	3CA	37C	299	0C6	30A	39F	214	12E	1FE	37D	39B	036	3CE	0C0
0020	0E9	31F	3FF	2D9	1A6	1AD	163	020	3AB	3AC	252	03C	39B	2CE	22A	23E
0030	323	2BA	04B	354	1BF	048	171	0E6	3DB	3A6	2E0	09E	2EE	12E	0EC	320
0040	0DC	0B8	340	3BB	0FC	009	1F1	0DD	363	011	0FC	2ED	13E	1BF	05D	027
0050	159	15B	15C	135	104	066	211	00F	2AC	0CA	0F3	0A7	194	0E6	2FA	358

Physical Block Write for ECC Enabled Blocks

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

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

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



User Design Memory Block: PF_TPSRAM_0/top_PF_TPSRAM_0_PF_TPSRAM_R0C0/INST_RAM1K20_IP
Depth X Width: 512 X 33

	Data	ECC bits	Error Detected	Corrected Data
0000	000000000	0C		
0001	000000011	43	Single-Bit	000000001
0002	000000002	46		
0003	111111111	09		

Detect Data error on Read Block or navigate from Logical view to Physical view

Inject error by editing the cell

Limitation of Injecting Errors

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

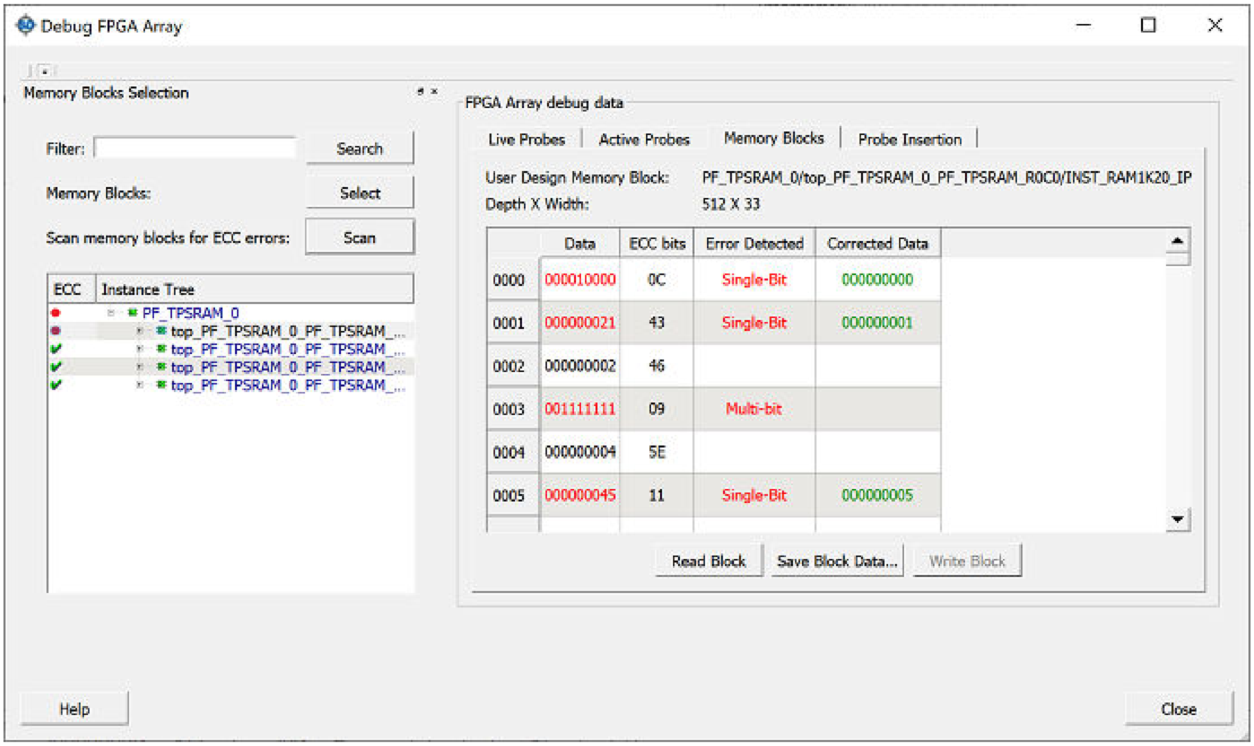
- An incorrect Single-bit corrected data is suggested.
- Error might not be detected because the ECC calculated for the corrupted data might be the same ECC calculated originally.

3.7.4 Scan Memory

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

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

Figure 3-21. Scan Memory Blocks for ECC Errors



Filter:

Memory Blocks:

Scan memory blocks for ECC errors:

ECC	Instance Tree
●	PF_TPSRAM_0
●	top_PF_TPSRAM_0_PF_TPSRAM_...
✓	top_PF_TPSRAM_0_PF_TPSRAM_...
✓	top_PF_TPSRAM_0_PF_TPSRAM_...
✓	top_PF_TPSRAM_0_PF_TPSRAM_...

User Design Memory Block: PF_TPSRAM_0/top_PF_TPSRAM_0_PF_TPSRAM_R0C0/INST_RAM1K20_IP
Depth X Width: 512 X 33

	Data	ECC bits	Error Detected	Corrected Data
0000	000010000	0C	Single-Bit	000000000
0001	000000021	43	Single-Bit	000000001
0002	000000002	46		
0003	001111111	09	Multi-bit	
0004	000000004	5E		
0005	000000045	11	Single-Bit	000000005

3.7.5 Unsupported Memory Blocks

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

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

3.7.6 Memory Blocks in Demo Mode

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

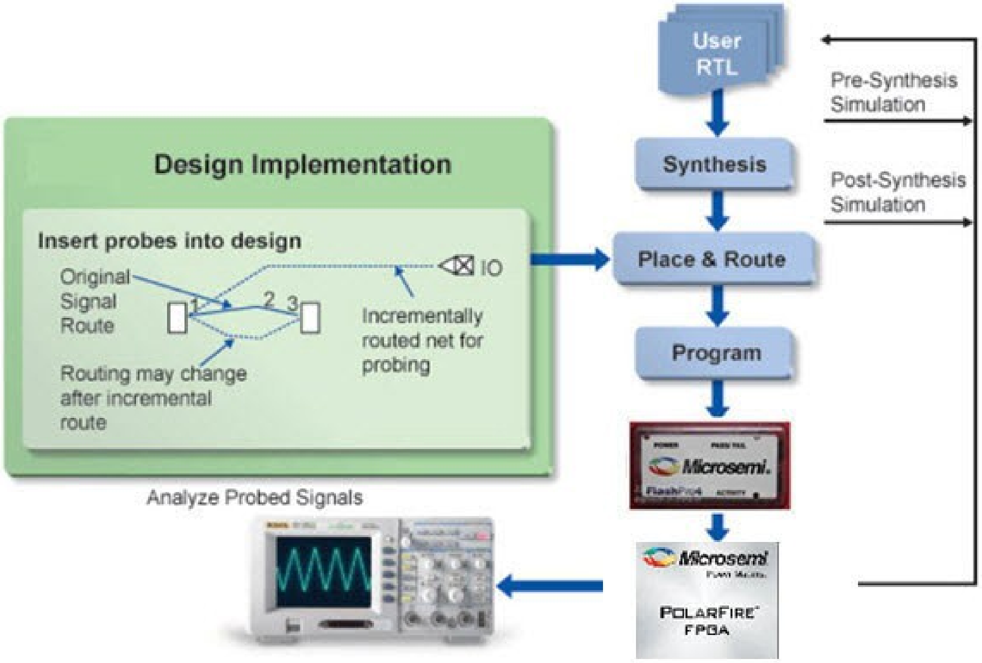
Both physical block view and logical block view are supported.

3.8 Probe Insertion (Post-Layout)

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

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

Figure 3-22. Probe Insertion in the Design Process



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

3.8.1 Inserting Probe and Programming the Device

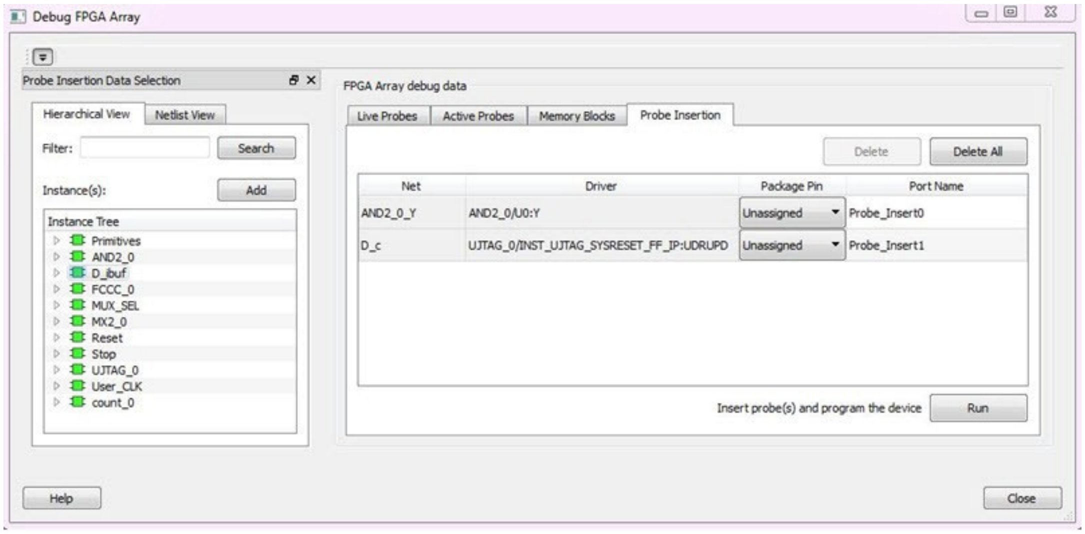
To insert probe(s) and program the device:

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

   Note: FlashPro Programmer must be connected for SmartDebug.

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

   Figure 3-23. Probe Insertion Tab



Net	Driver	Package Pin	Port Name
AND2_0_Y	AND2_0/U0:Y	Unassigned	Probe_Insert0
D_c	UJTAG_0/INST_UJTAG_SYSRESET_FF_IP:UDRUPD	Unassigned	Probe_Insert1

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

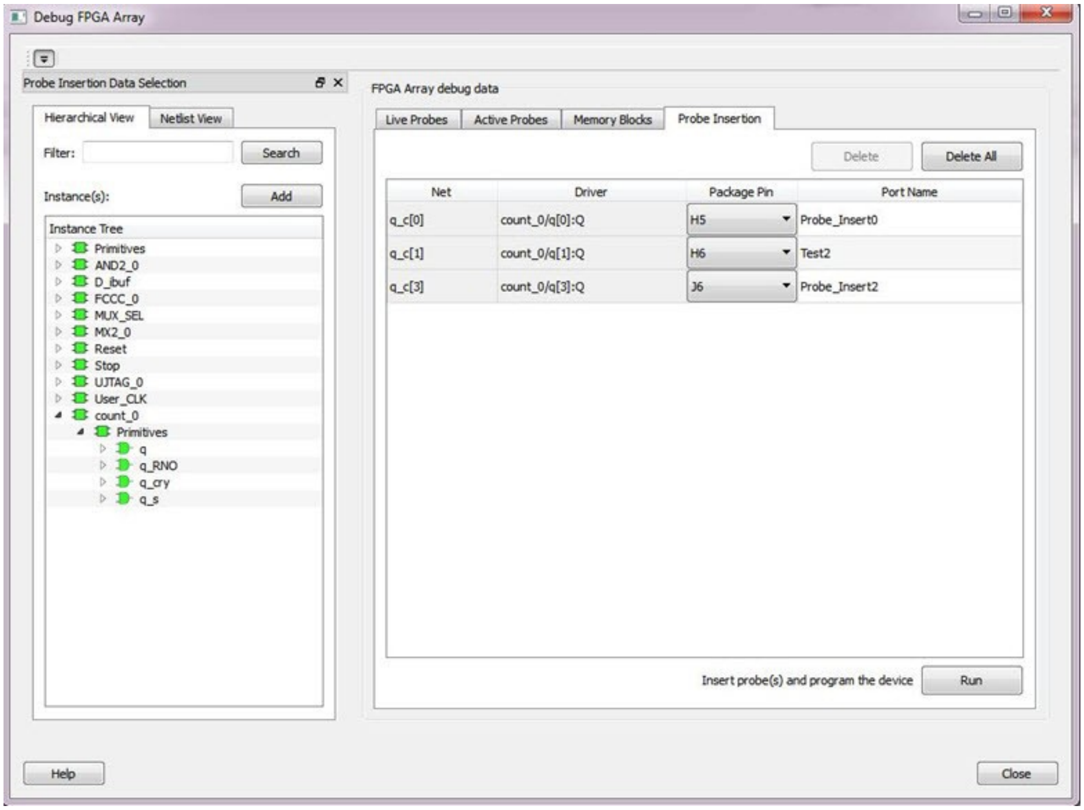
3. Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the **Filter** box.

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

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

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



Net	Driver	Package Pin	Port Name
q_c[0]	count_0/q[0]:Q	H5	Probe_Insert0
q_c[1]	count_0/q[1]:Q	H6	Test2
q_c[3]	count_0/q[3]:Q	J6	Probe_Insert2

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.

3.8.2 Deleting a Probe

To delete a probe, select the probe and click **Delete**. To delete all probes, click **Delete All**.
Note: Deleting probes from the probes list without clicking **Run** does not automatically remove the probes from the design.

3.8.3 Reverting to the Original Design

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

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.

3.9 FPGA Hardware Breakpoint Auto Instantiation

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

To enable this option:

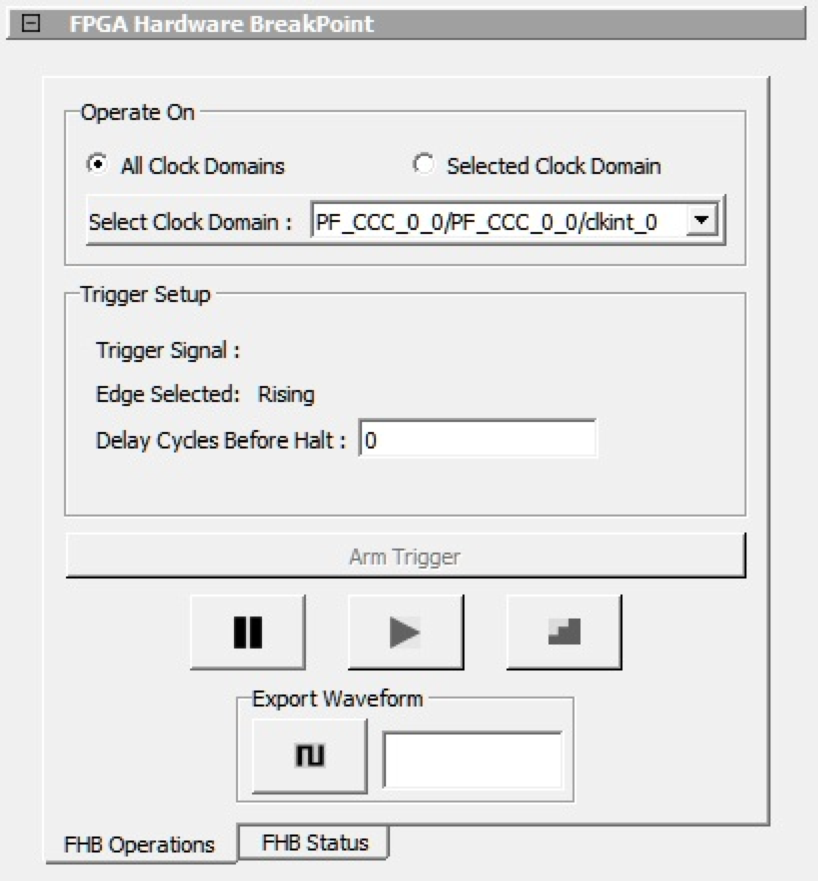
1. Launch Libero.
2. Select **Project** > **Project Settings**. The **Project settings** dialog box appears.
3. Select **Design flow** from the options available on the left pane of the dialog box.
4. Select the **Enable FPGA Hardware Breakpoints Auto Instantiation** check box from the **Root top** options that appear on the right pane of the dialog box.

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

3.9.1 FHB Operations

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

Figure 3-25. FPGA Hardware Breakpoint (FHB) Controls



FPGA Hardware BreakPoint

Operate On

● All Clock Domains ○ Selected Clock Domain

Select Clock Domain : PF_CCC_0_0/PF_CCC_0_0/clkint_0

Trigger Setup

Trigger Signal :

Edge Selected: Rising

Delay Cycles Before Halt : 0

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

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

Note: The Trigger Signal is shown as Not Connected until a live probe is assigned. When a probe is assigned to Live Probe PROBE_A, the Trigger Signal updates.

If you require a certain number of clock cycles before halting the clock domain after triggering, a value between 0 and 255 must be entered for Delay Cycles Before Halt before you click **Arm Trigger**. This sets the FHBs to trigger after the specified delay from the rising edge trigger.
Note: Delay is not applied to a forced Halt.

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

Live Probe Halt

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

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

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

Note: Live Probe Halt can be delayed for a maximum of 255 clock cycles. The actual delay realized on hardware is calculated by the following equation:

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

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

For more information, see Assumptions and Limitations .

Force Halt

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

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

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

Play

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

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

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

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

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

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

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

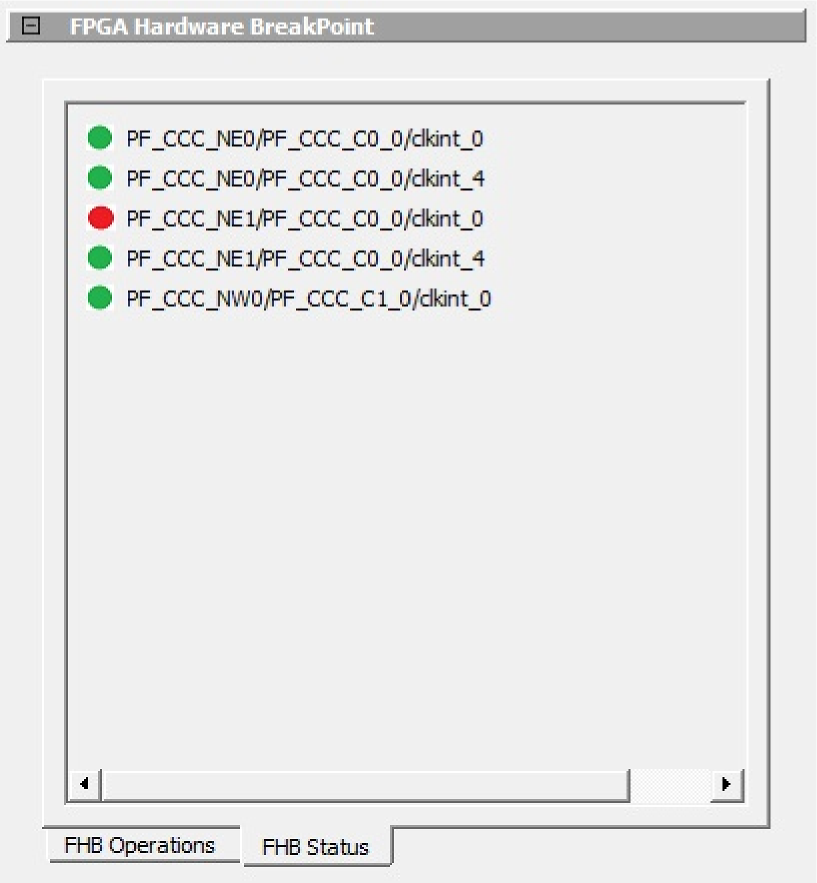
Force Halt/Play/Step is done using the FHB controls. Once the clock domain is halted, you can either force Play the clock domain or Step the clock domain by 1 clock cycle. You can save the waveform view of the selected active probes using Export Waveform by specifying the number of clock cycles to capture. The waveform is saved to a `.vcd` file.

3.9.2 FHB Status

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 functionality.

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

Figure 3-26. FPGA Hardware Breakpoint Clock Status



🟢 PF_CCC_NE0/PF_CCC_C0_0/clkint_0
🟢 PF_CCC_NE0/PF_CCC_C0_0/clkint_4
🔴 PF_CCC_NE1/PF_CCC_C0_0/clkint_0
🟢 PF_CCC_NE1/PF_CCC_C0_0/clkint_4
🟢 PF_CCC_NW0/PF_CCC_C1_0/clkint_0

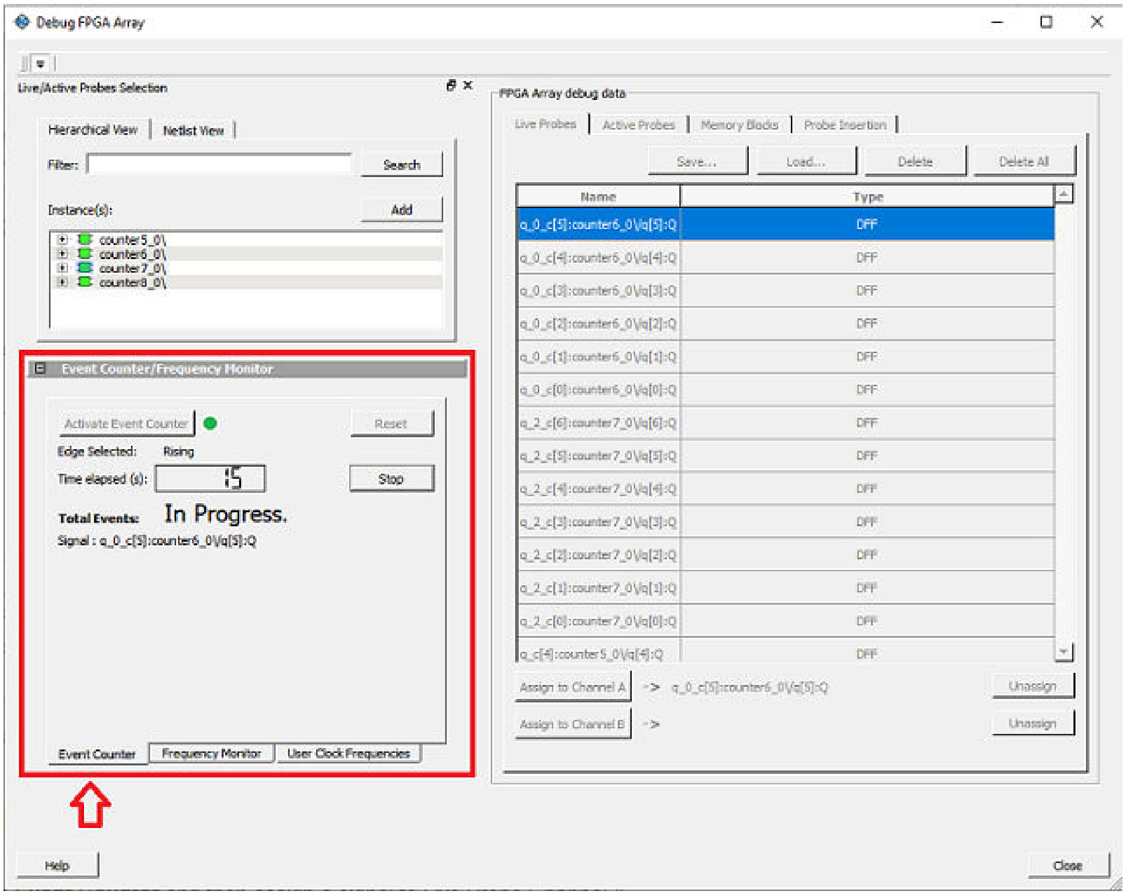
3.9.3 Assumptions and Limitations

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

3.10 Event Counter

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

Figure 3-27. Event Counter Tab/UI



Filter:

Instance(s):

Edge Selected: Rising

Time elapsed (s): 15

Total Events: In Progress.

Signal : q_0_c[5]:counter6_0\/q[5]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q

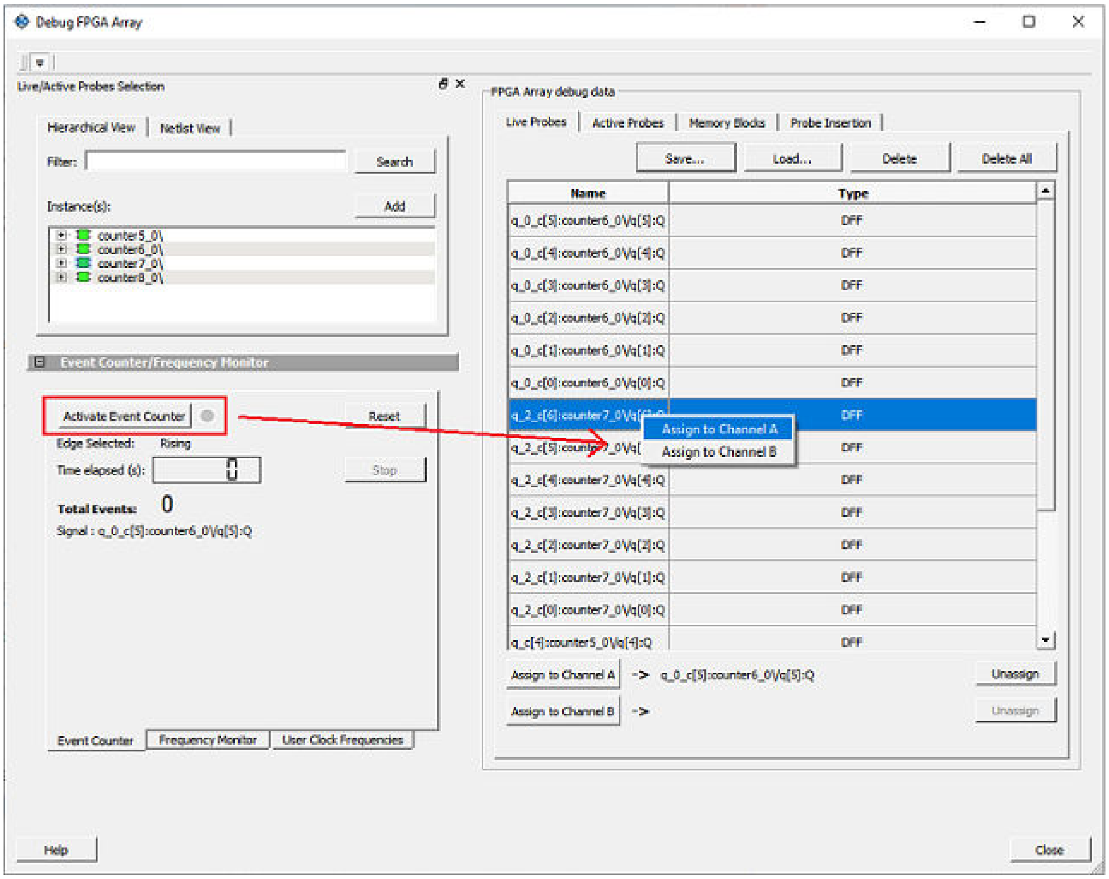
Assign to Channel B ->

3.10.1 Activating the Event Counter

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

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

Figure 3-28. Activating the Event Counter



Edge Selected: Rising

Time elapsed (s): 0

Total Events: 0

Signal : q_0_c[5]:counter6_0\/q[5]:Q

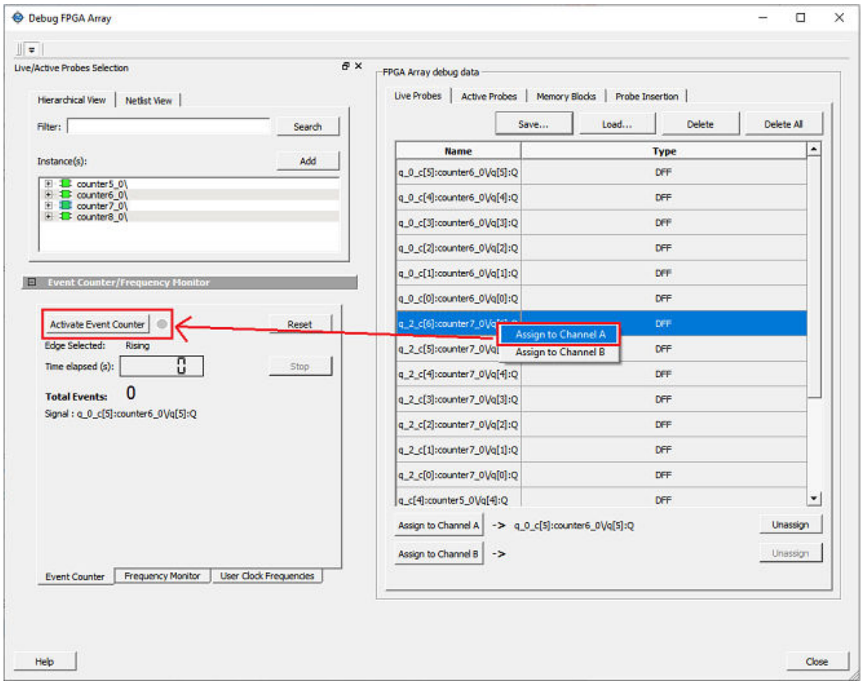
Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q

Assign to Channel B ->

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

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



Edge Selected: Rising

Time elapsed (s): 0

Total Events: 0

Signal : q_0_c[5]:counter6_0\/q[5]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

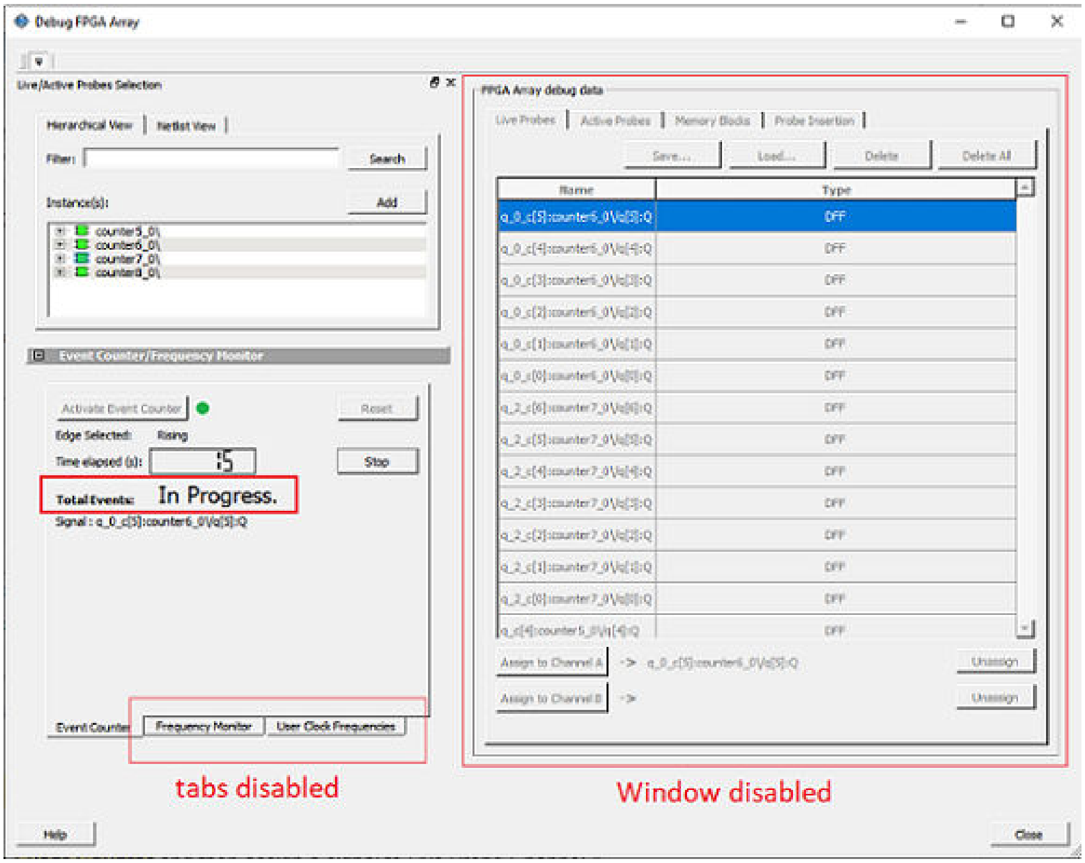
Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q

Assign to Channel B ->

3.10.2 Running the Event Counter

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

Figure 3-30. Running the Event Counter



Edge Selected: Rising
Time elapsed (s): 5
Total Events: In Progress.
Signal : q_0_c[5]:counter6_0\/q[5]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q
Assign to Channel B ->

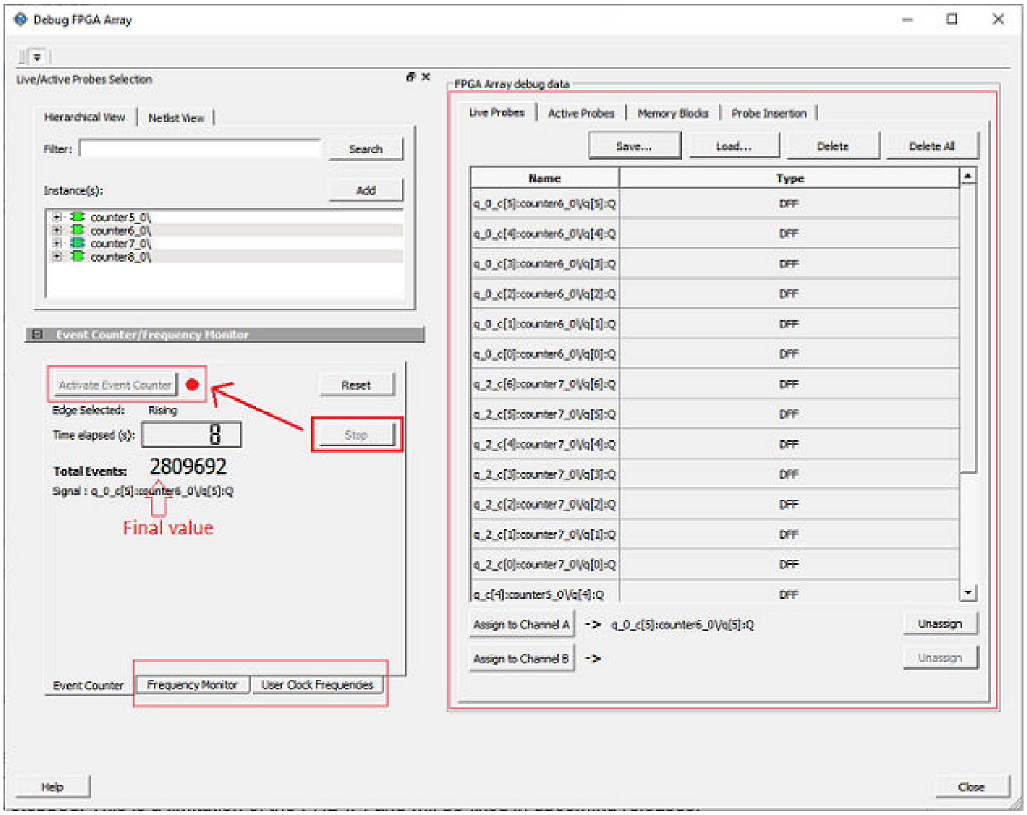
tabs disabled

Window disabled

3.10.3 Stopping the Event Counter

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

Figure 3-31. Stopping the Event Counter



Edge Selected: Rising

Time elapsed (s): 8

Total Events: 2809692

Signal : q_0_c[5]:counter6_0\/q[5]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q

Assign to Channel B ->

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

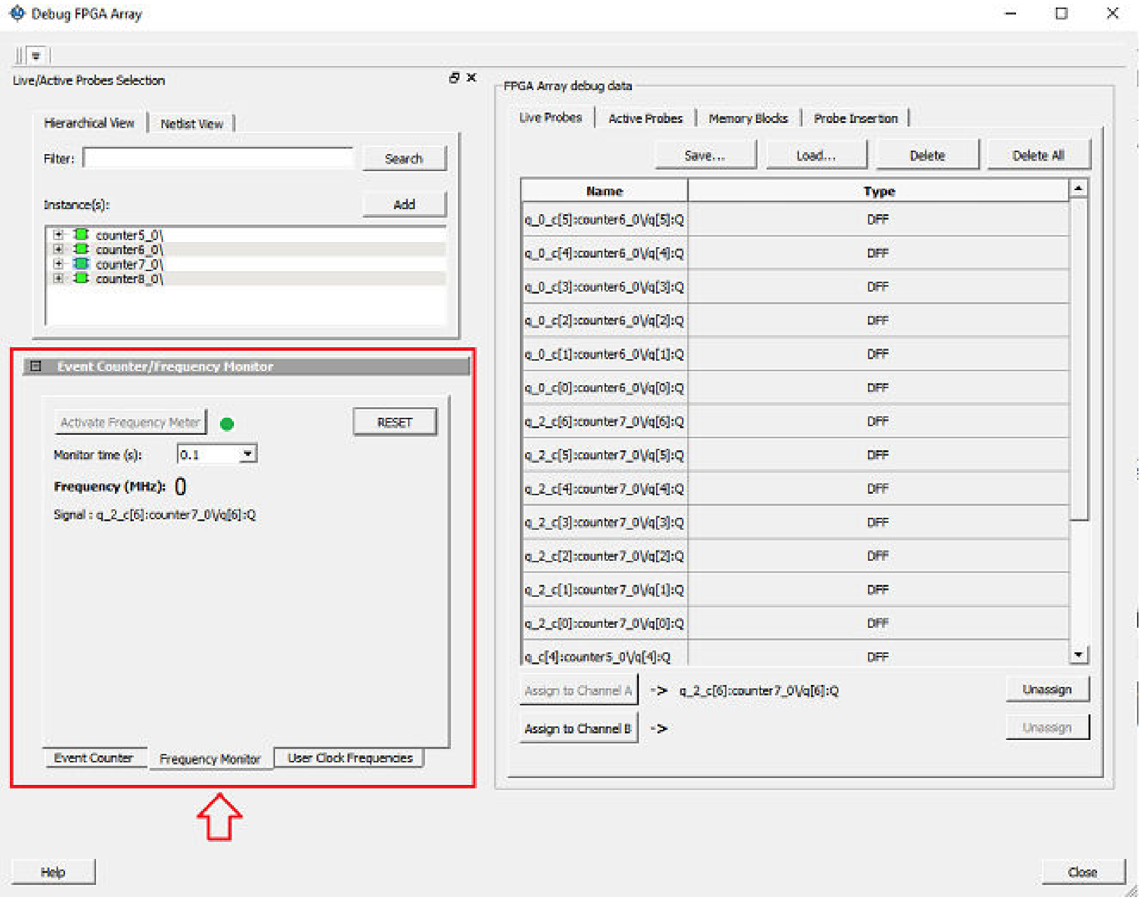
For more information, see Frequency Monitor and User Clock Frequencies.

3.11 Frequency Monitor

The Frequency Monitor calculates the frequency of any signal in the design that can be assigned to Live Probe channel A. The Frequency Monitor must be activated before or after the signal is assigned to Live Probe Channel A.

You can enter the time to monitor the signal. The accuracy of results increases as the monitor time increases. The unit of measurement is displayed in Megahertz (MHz). During the run, progress is displayed in the pane.

Figure 3-32. Frequency Monitor Tab/UI



Monitor time (s): 0.1

Frequency (MHz): 0

Signal : q_2_c[6]:counter7_0\/q[6]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_2_c[6]:counter7_0\/q[6]:Q

Assign to Channel B ->

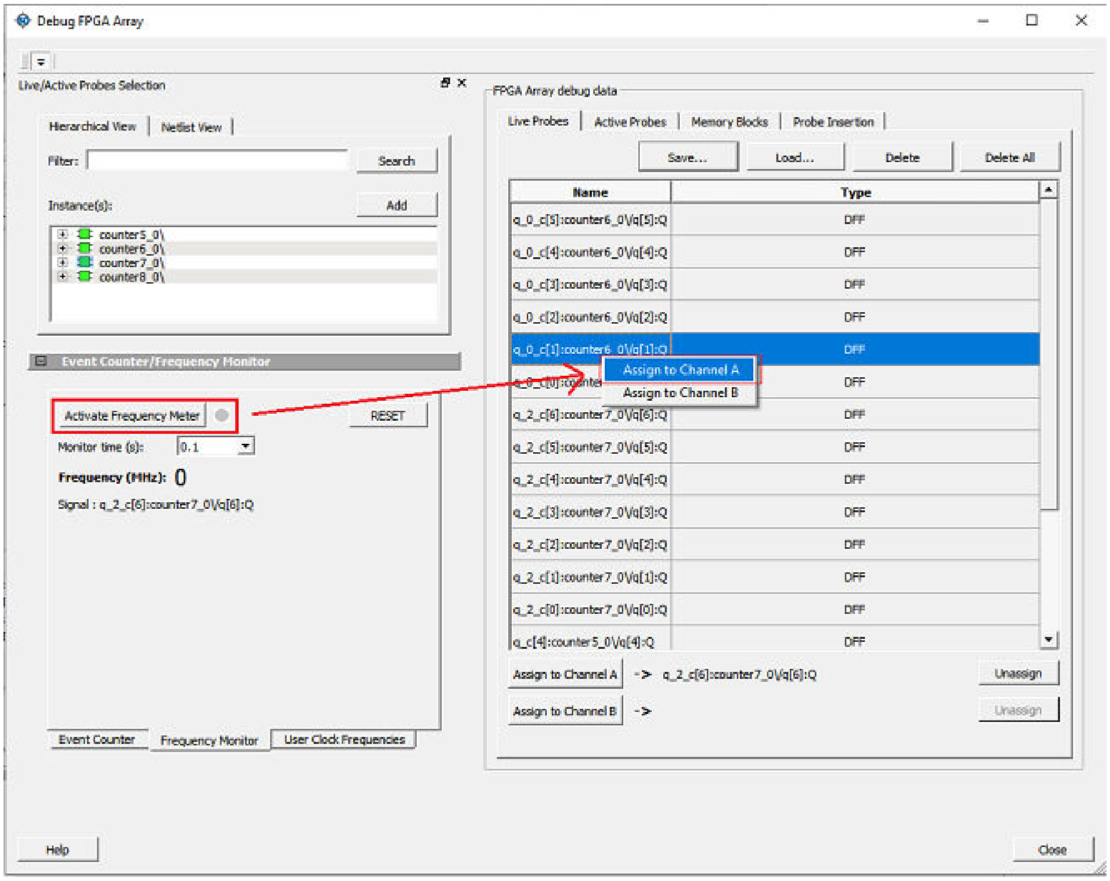
In the **Frequency Monitor** tab, you can activate the Frequency Monitor, change the monitor time (delay to calculate frequency), reset the monitor, and set the frequency in megahertz (MHz). Click the drop-down list to select monitor time value. During the frequency calculation, all tabs on the right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

3.11.1 Activating the Frequency Monitor

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

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

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



Debug FPGA Array

Live/Active Probes Selection

Hierarchical View | Netlist View

Filter: Search

Instance(s): Add

- counter5_0\
- counter6_0\
- counter7_0\
- counter8_0\

Event Counter/Frequency Monitor

Activate Frequency Meter ● RESET

Monitor time (s): 0.1

Frequency (MHz): 0

Signal : q_2_c[6]:counter7_0\q[6]:Q

Event Counter | Frequency Monitor | User Clock Frequencies

FPGA Array debug data

Live Probes | Active Probes | Memory Blocks | Probe Insertion

Save... Load... Delete Delete All

Name	Type
q_0_c[5]:counter6_0\q[5]:Q	DFF
q_0_c[4]:counter6_0\q[4]:Q	DFF
q_0_c[3]:counter6_0\q[3]:Q	DFF
q_0_c[2]:counter6_0\q[2]:Q	DFF
q_0_c[1]:counter6_0\q[1]:Q	DFF
q_0_c[0]:counte	DFF
q_2_c[6]:counter7_0\q[6]:Q	DFF
q_2_c[5]:counter7_0\q[5]:Q	DFF
q_2_c[4]:counter7_0\q[4]:Q	DFF
q_2_c[3]:counter7_0\q[3]:Q	DFF
q_2_c[2]:counter7_0\q[2]:Q	DFF
q_2_c[1]:counter7_0\q[1]:Q	DFF
q_2_c[0]:counter7_0\q[0]:Q	DFF
q_c[4]:counter5_0\q[4]:Q	DFF

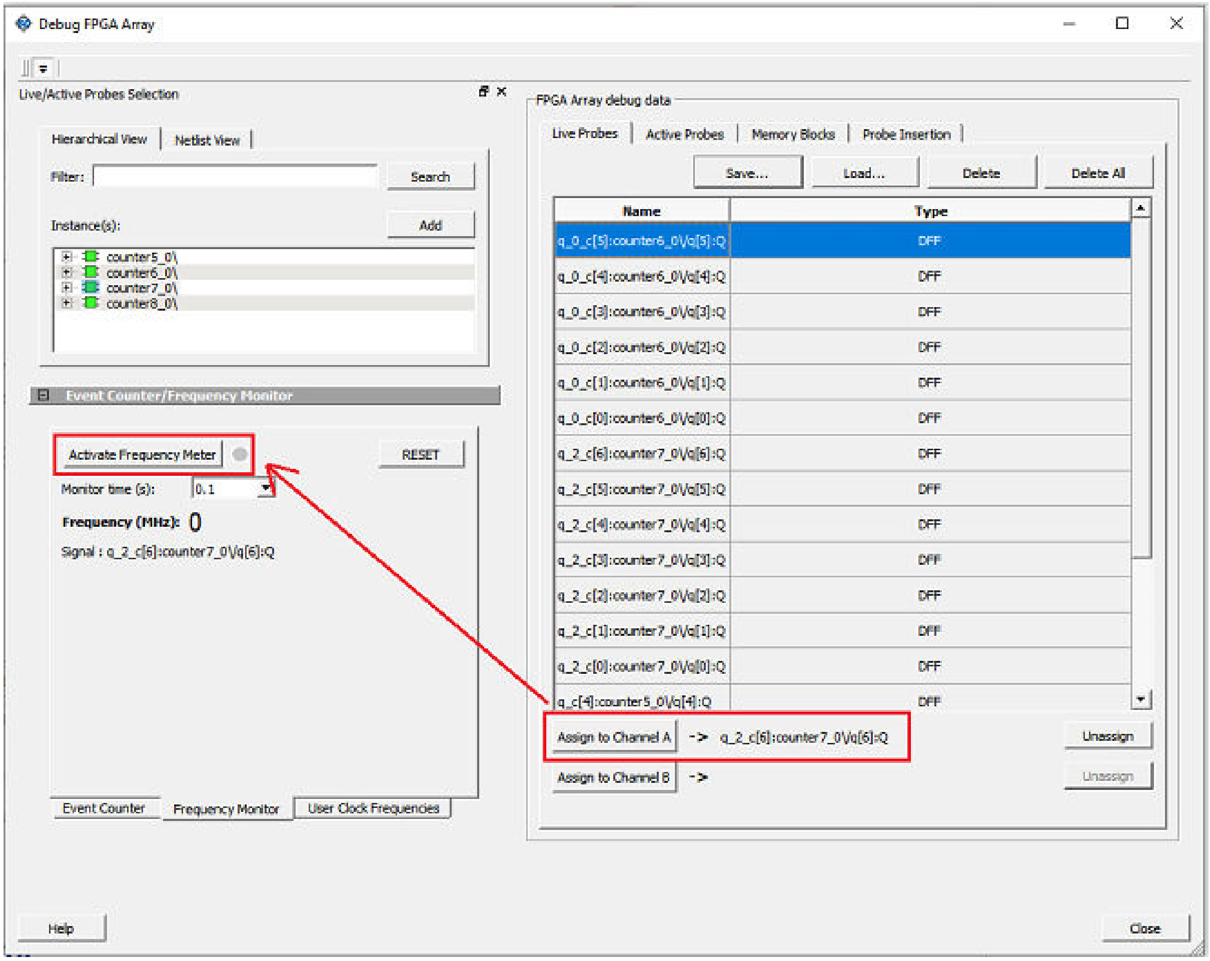
Assign to Channel A -> q_2_c[6]:counter7_0\q[6]:Q Unassign

Assign to Channel B -> Unassign

Help Close

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

Figure 3-34. Activating the Frequency Monitor



Filter:

Instance(s):
- counter5_0\
- counter6_0\
- counter7_0\
- counter8_0\

Monitor time (s): 0.1

Frequency (MHz): 0

Signal : q_2_c[6]:counter7_0\/q[6]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_2_c[6]:counter7_0\/q[6]:Q

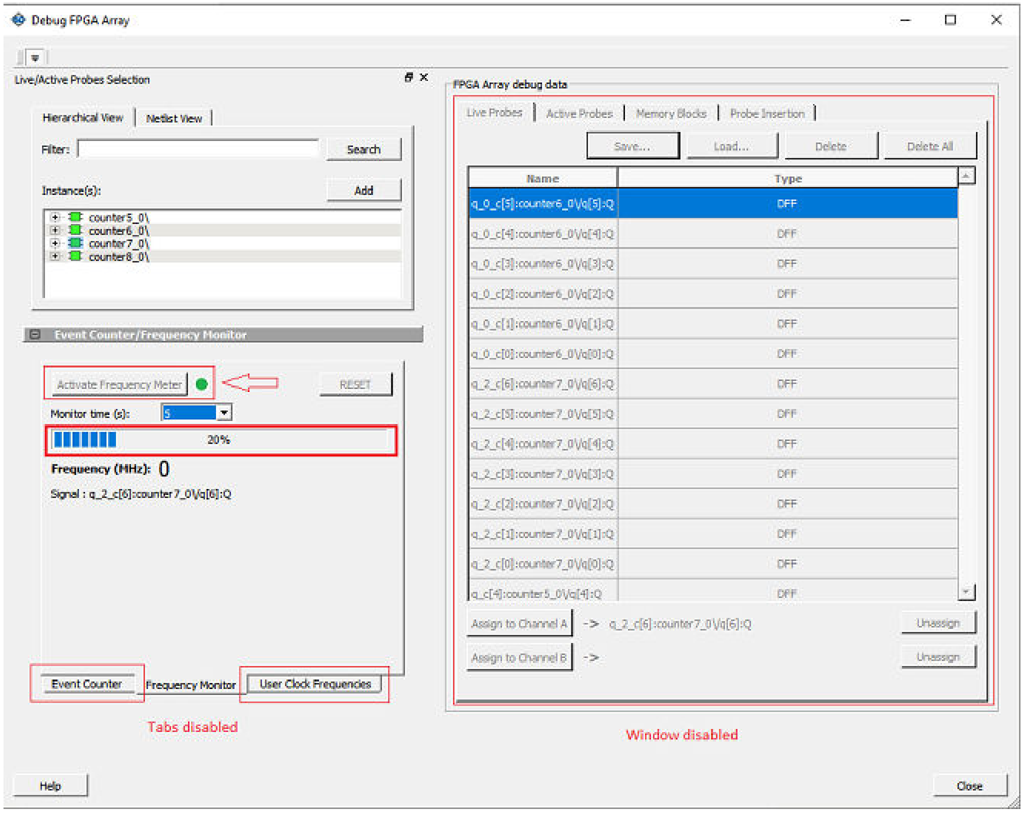
Assign to Channel B ->

3.11.2 Running the Frequency Monitor

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

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

Figure 3-35. Running the Frequency Monitor



Monitor time (s): 5

20%

Frequency (MHz): 0

Signal : q_2_c[6]:counter7_0\/q[6]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_2_c[6]:counter7_0\/q[6]:Q

Assign to Channel B ->

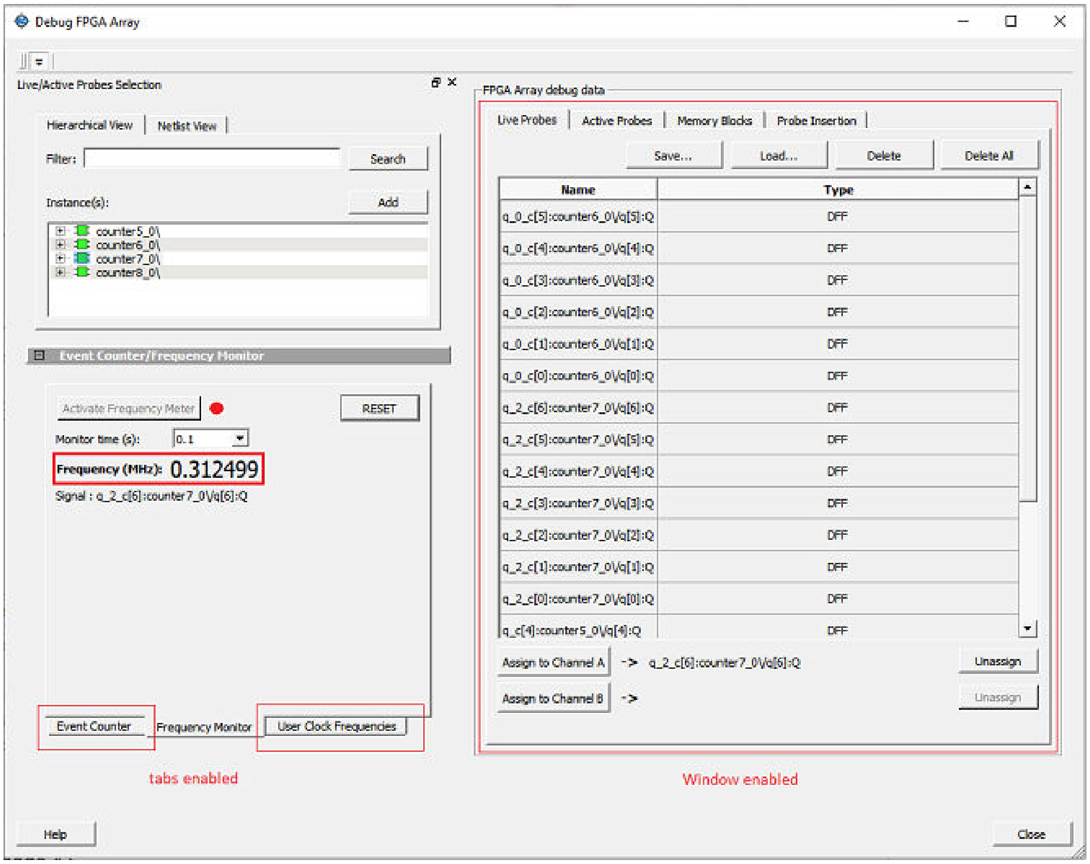
Tabs disabled

Window disabled

3.11.3 Stopping the Frequency Monitor

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

Figure 3-36. Stopping the Frequency Monitor



Monitor time (s): 0.1

Frequency (MHz): 0.312499

Signal : q_2_c[6]:counter7_0\/q[6]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_2_c[6]:counter7_0\/q[6]:Q

Assign to Channel B ->

tabs enabled

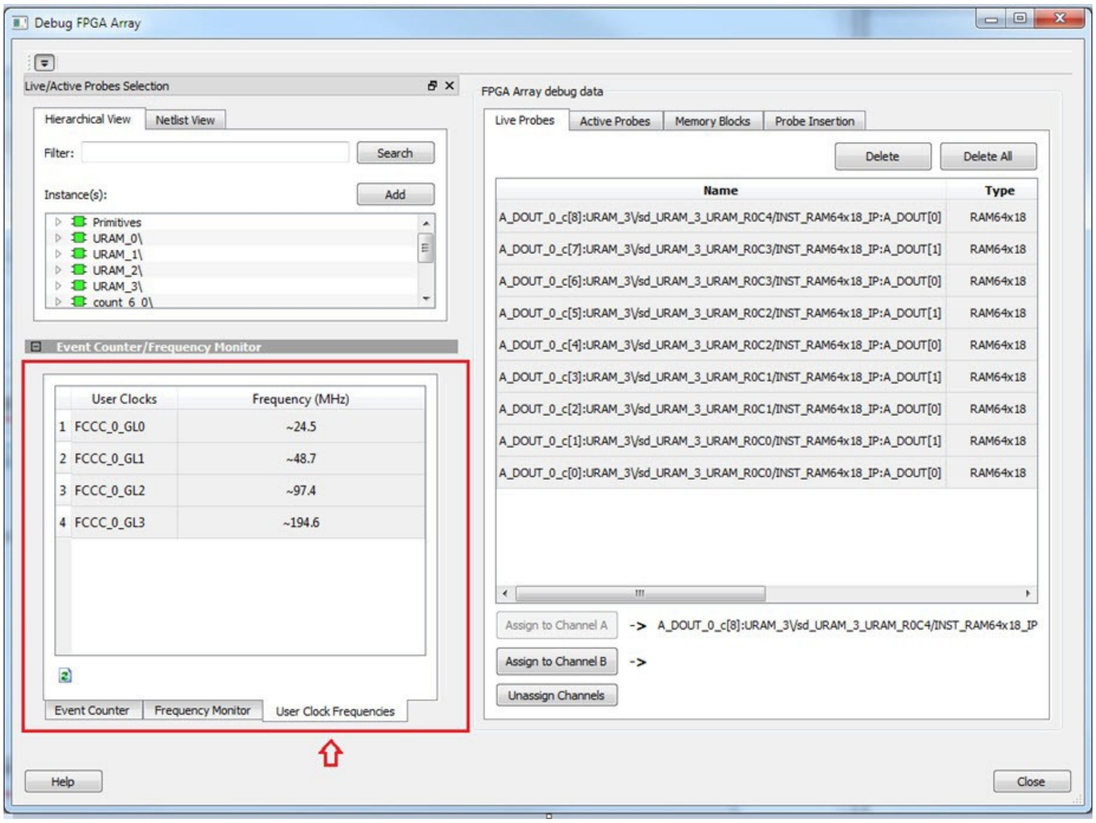
Window enabled

For more information, see Frequency Monitor and User Clock Frequencies.

3.12 User Clock Frequencies

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

Figure 3-37. User Clock Frequencies Tab/UI



	User Clocks	Frequency (MHz)
1	FCCC_0_GL0	~24.5
2	FCCC_0_GL1	~48.7
3	FCCC_0_GL2	~97.4
4	FCCC_0_GL3	~194.6

Name	Type
A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[7]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[4]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[1]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[0]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18

Assign to Channel A -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP

Assign to Channel B ->

For related information, see Event Counter and Frequency Monitor.

3.13 Debug sNVM

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

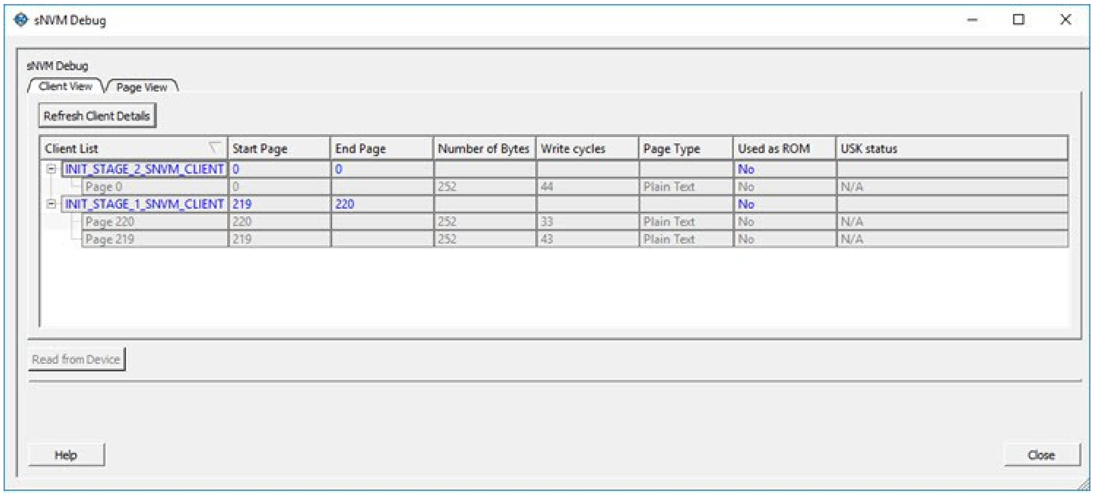
The sNVM Debug window has two tabs – Client View and Page View.

3.13.1 Client View

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

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

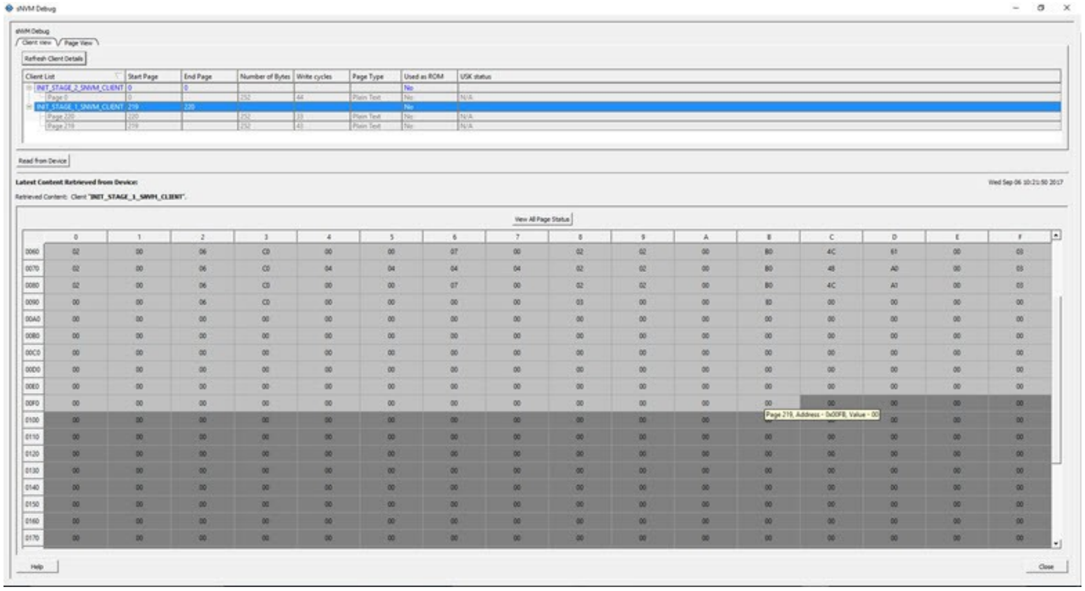
Figure 3-38. Client View - Expanded List



Client List	Start Page	End Page	Number of Bytes	Write cycles	Page Type	Used as ROM	USK status
⊟ INIT_STAGE_2_SNVM_CLIENT	0	0				No	
Page 0	0		252	44	Plain Text	No	N/A
⊟ INIT_STAGE_1_SNVM_CLIENT	219	220				No	
Page 220	220		252	33	Plain Text	No	N/A
Page 219	219		252	43	Plain Text	No	N/A

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

Figure 3-39. Client View - Memory



Client List	Start Page	End Page	Number of Bytes	Write cycles	Page Type	Used as ROM	USK status
⊟ INIT_STAGE_2_SNVM_CLIENT	0	0				No	
Page 0	0		252	44	Plain Text	No	N/A
⊟ INIT_STAGE_1_SNVM_CLIENT	219	220				No	
Page 220	220		252	33	Plain Text	No	N/A
Page 219	219		252	43	Plain Text	No	N/A

Latest Content Retrieved from Device: Wed Sep 06 10:21:50 2017

Retrieved Content: Client "INIT_STAGE_1_SNVM_CLIENT".

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0060	02	00	06	C0	00	00	07	00	02	02	00	B0	4C	61	00	03
0070	02	00	06	C0	04	04	04	04	02	02	00	B0	48	A0	00	03
0080	02	00	06	C0	00	00	07	00	02	02	00	B0	4C	A1	00	03
0090	00	00	06	C0	00	00	00	00	03	00	00	80	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

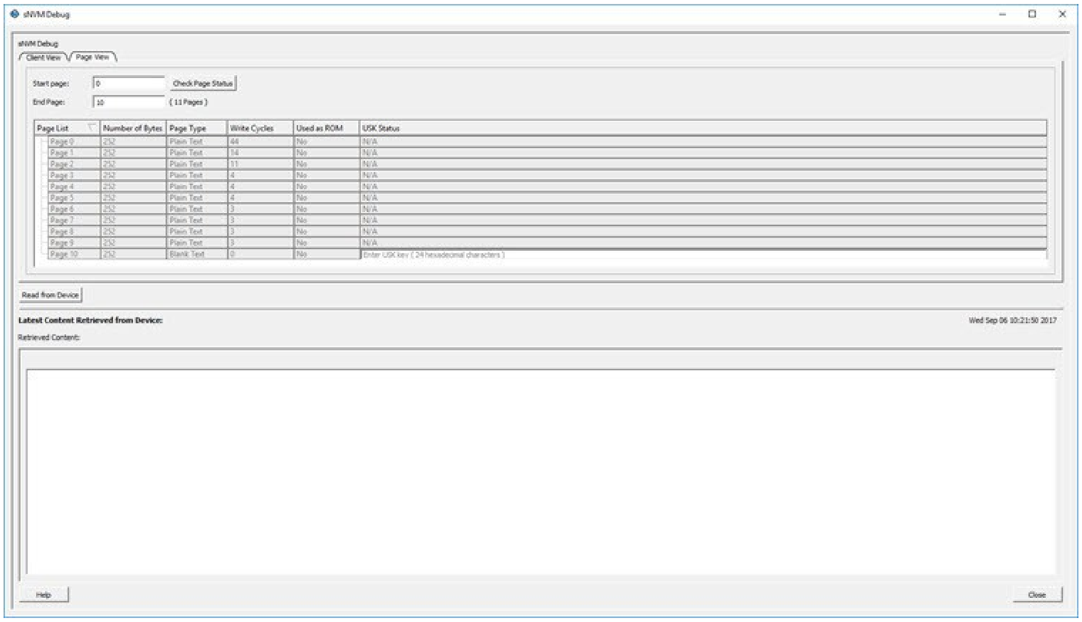
Page 219, Address - 0x00FB, Value - 00

3.13.2 Page View

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

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

Figure 3-40. Page View - Enter USK highlighted



Start page: 0

End Page: 10 (11 Pages)

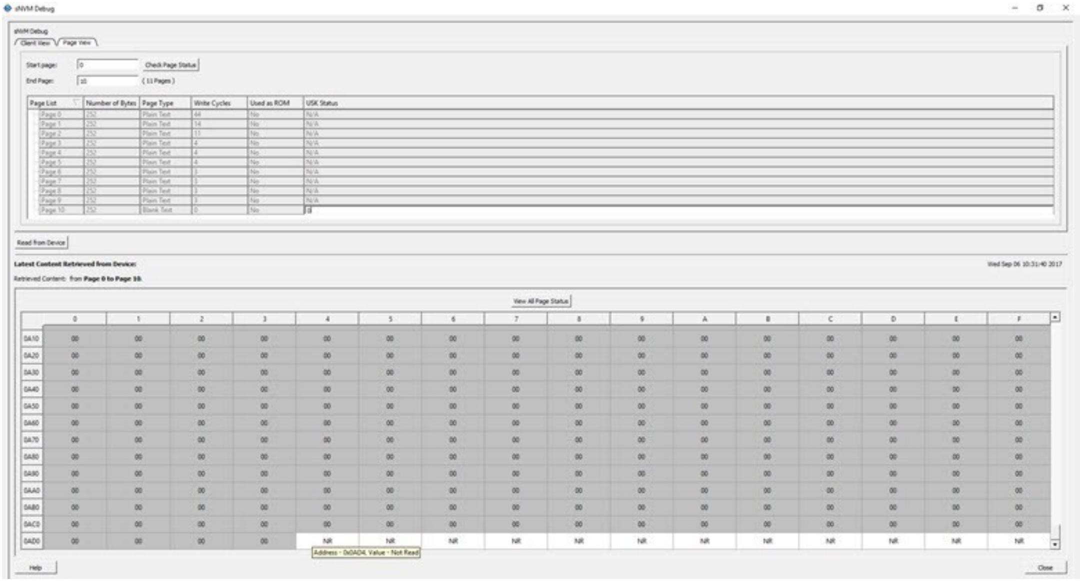
Page List	Number of Bytes	Page Type	Write Cycles	Used as ROM	USK Status
Page 0	252	Plain Text	44	No	N/A
Page 1	252	Plain Text	14	No	N/A
Page 2	252	Plain Text	11	No	N/A
Page 3	252	Plain Text	4	No	N/A
Page 4	252	Plain Text	4	No	N/A
Page 5	252	Plain Text	4	No	N/A
Page 6	252	Plain Text	3	No	N/A
Page 7	252	Plain Text	3	No	N/A
Page 8	252	Plain Text	3	No	N/A
Page 9	252	Plain Text	3	No	N/A
Page 10	252	Blank Text	0	No	Enter USK key (24 hexadecimal characters)

Latest Content Retrieved from Device: Wed Sep 06 10:21:50 2017

Retrieved Content:

The following figure shows the specified page range.

Figure 3-41. Page View - Page Range



Start page: 0

End Page: 10 (11 Pages)

Page List	Number of Bytes	Page Type	Write Cycles	Used as ROM	USK Status
Page 0	252	Plain Text	44	No	N/A
Page 1	252	Plain Text	14	No	N/A
Page 2	252	Plain Text	11	No	N/A
Page 3	252	Plain Text	4	No	N/A
Page 4	252	Plain Text	4	No	N/A
Page 5	252	Plain Text	4	No	N/A
Page 6	252	Plain Text	3	No	N/A
Page 7	252	Plain Text	3	No	N/A
Page 8	252	Plain Text	3	No	N/A
Page 9	252	Plain Text	3	No	N/A
Page 10	252	Blank Text	0	No	0

Latest Content Retrieved from Device: Wed Sep 06 10:31:40 2017

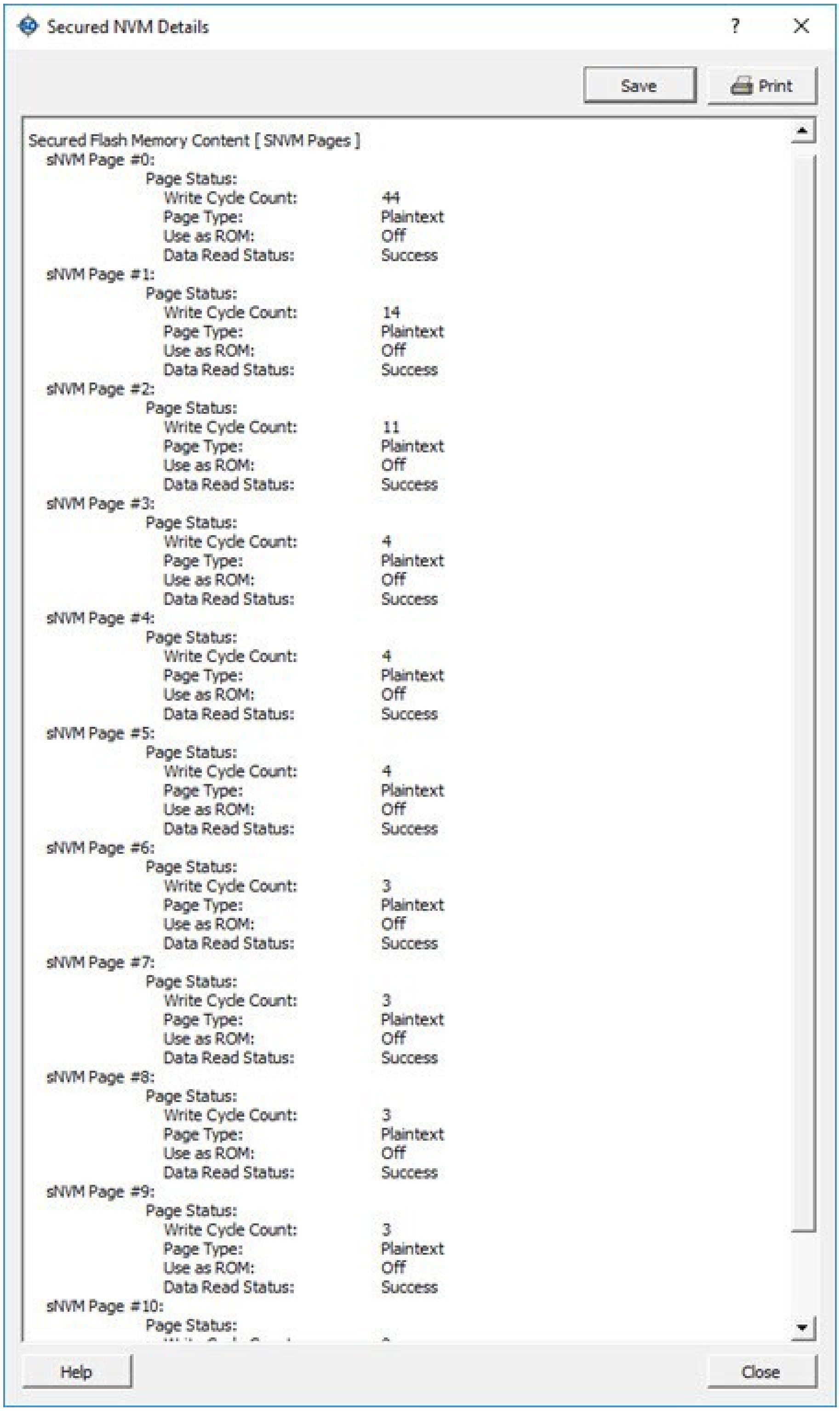
Retrieved Content: from **Page 0 to Page 10.**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0A10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0AA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0AB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0AC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0AD0	00	00	00	00	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR

Address - 0x0AD4, Value - Not Read

The following figure shows the page status.

Figure 3-42. View All Page Status



```
Secured Flash Memory Content [ SNVM Pages ]
   sNVM Page #0:
                Page Status:
                   Write Cycle Count:          44
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #1:
                Page Status:
                   Write Cycle Count:          14
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #2:
                Page Status:
                   Write Cycle Count:          11
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #3:
                Page Status:
                   Write Cycle Count:          4
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #4:
                Page Status:
                   Write Cycle Count:          4
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #5:
                Page Status:
                   Write Cycle Count:          4
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #6:
                Page Status:
                   Write Cycle Count:          3
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #7:
                Page Status:
                   Write Cycle Count:          3
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #8:
                Page Status:
                   Write Cycle Count:          3
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #9:
                Page Status:
                   Write Cycle Count:          3
                   Page Type:                  Plaintext
                   Use as ROM:                 Off
                   Data Read Status:           Success
   sNVM Page #10:
                Page Status:
```

3.13.3 Read Operation

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

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

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

3.13.4 Runtime Operations

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

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

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

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

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

3.13.5 Demo Mode

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

3.14 Debug Transceiver

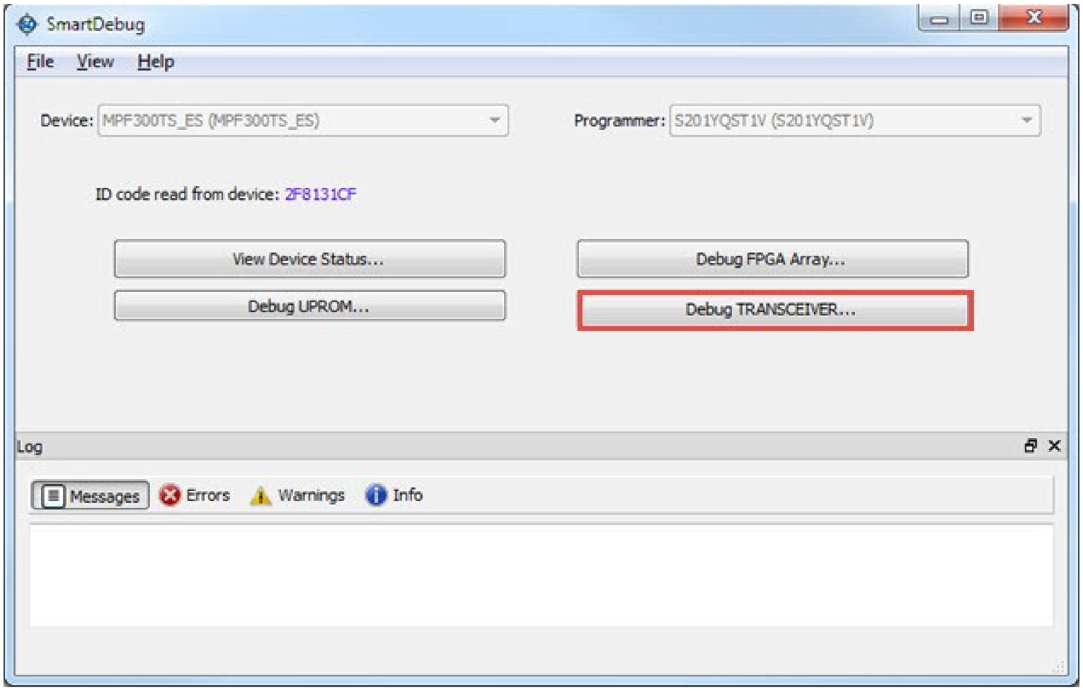
The Debug Transceiver feature in SmartDebug checks the lane functionality and health for different settings of the lane parameters.

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

- Configuration Report
- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor
- Register Access
- Signal Integrity
- Optimize Receiver

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

Figure 3-43. SmartDebug Window - Debug TRANSCEIVER



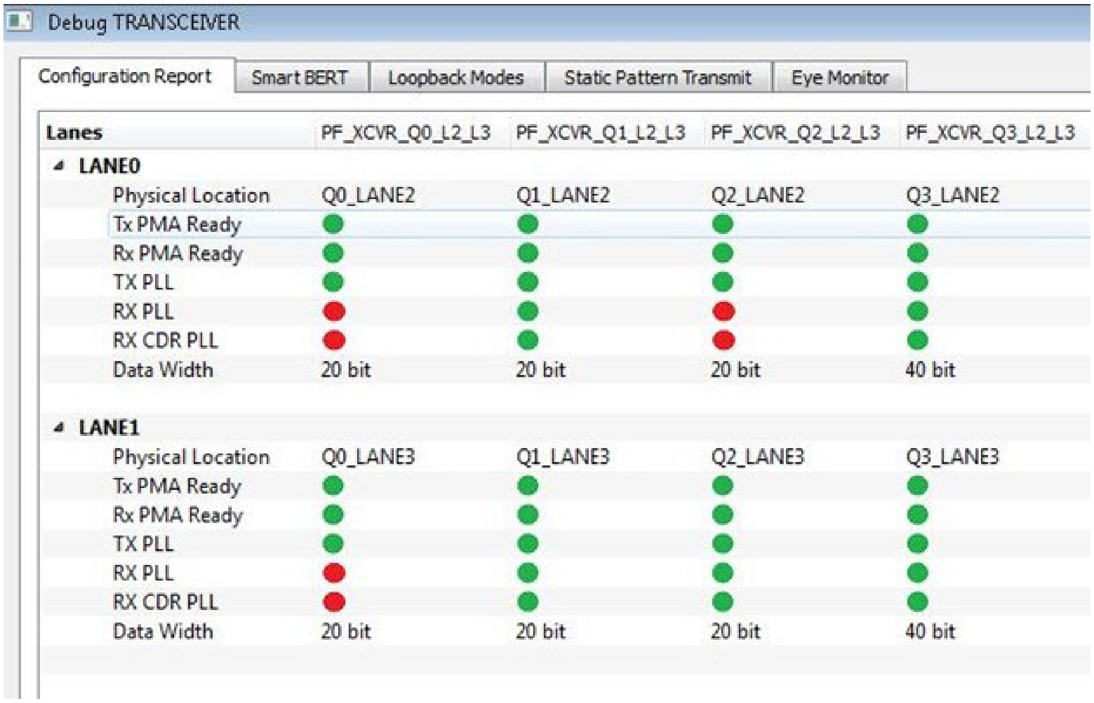
Device: MPF300TS_ES (MPF300TS_ES)

Programmer: S201YQST1V (S201YQST1V)

ID code read from device: 2F8131CF

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

Figure 3-44. Debug Transceiver Dialog Box



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

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

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

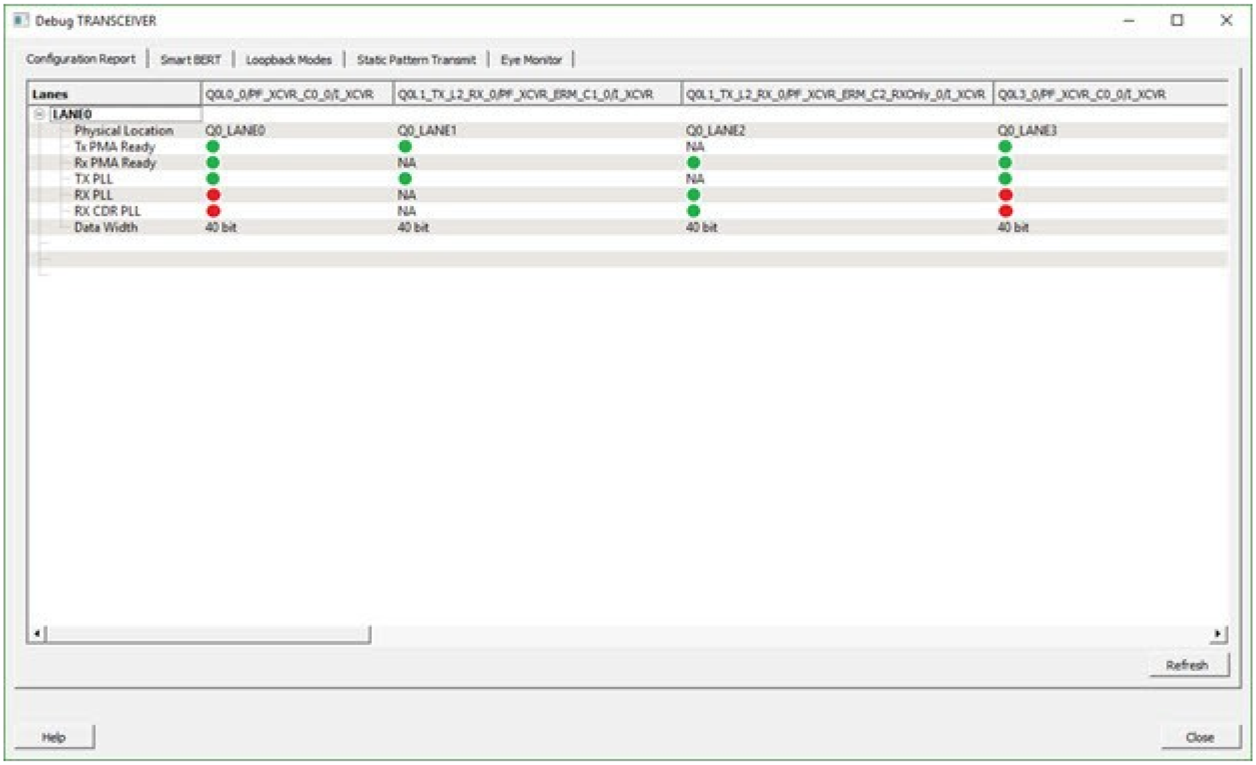
- Eye Monitor
- Register Access

3.14.1 Configuration Report

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

Click the **Refresh** button to refresh the information.
Note: The report refreshes automatically when you navigate from another tab.

Figure 3-45. Debug TRANSCEIVER - Configuration Report


Debug TRANSCEIVER
Configuration Report | Smart BERT | Loopback Modes | Static Pattern Transmit | Eye Monitor
Refresh
Help
Close

Lanes	Q0L0_0/PF_XCVR_C0_0/I_XCVR	Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR	Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR	Q0L3_0/PF_XCVR_C0_0/I_XCVR
LANE0				
Physical Location	Q0_LANE0	Q0_LANE1	Q0_LANE2	Q0_LANE3
Tx PMA Ready	🟢	🟢	NA	🟢
Rx PMA Ready	🟢	NA	🟢	🟢
TX PLL	🟢	🟢	NA	🟢
RX PLL	🔴	NA	🟢	🔴
RX CDR PLL	🔴	NA	🟢	🔴
Data Width	40 bit	40 bit	40 bit	40 bit

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

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

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

3.14.2 Transceiver Hierarchy

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

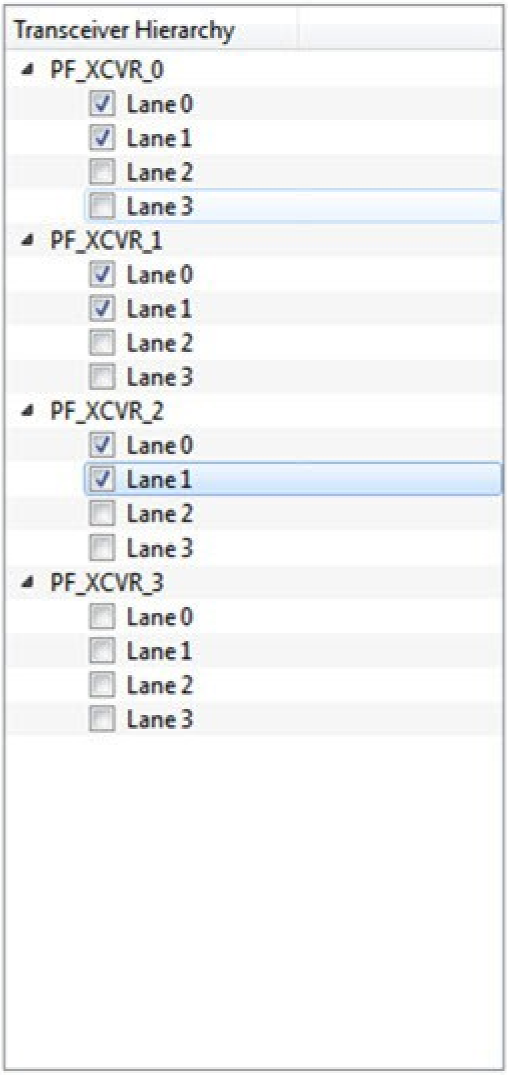
Transceiver Hierarchy view appears on the following tabs:

- SmartBERT

- Loopback Modes
- Static Pattern Transmit
- Eye Monitor

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

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

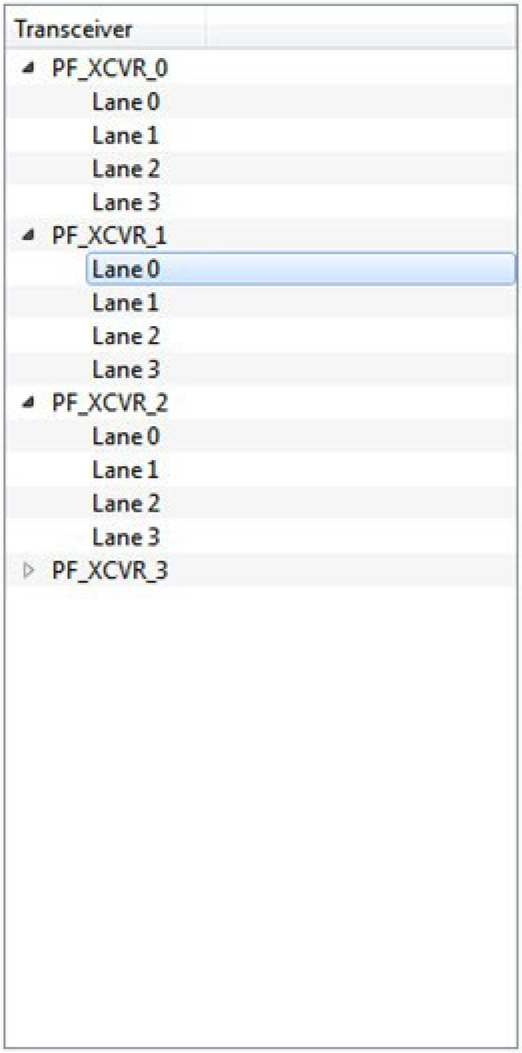


Transceiver Hierarchy

- PF_XCVR_0
  - ☑ Lane 0
  - ☑ Lane 1
  - ☐ Lane 2
  - ☐ Lane 3
- PF_XCVR_1
  - ☑ Lane 0
  - ☑ Lane 1
  - ☐ Lane 2
  - ☐ Lane 3
- PF_XCVR_2
  - ☑ Lane 0
  - ☑ Lane 1
  - ☐ Lane 2
  - ☐ Lane 3
- PF_XCVR_3
  - ☐ Lane 0
  - ☐ Lane 1
  - ☐ Lane 2
  - ☐ Lane 3

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

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



Transceiver

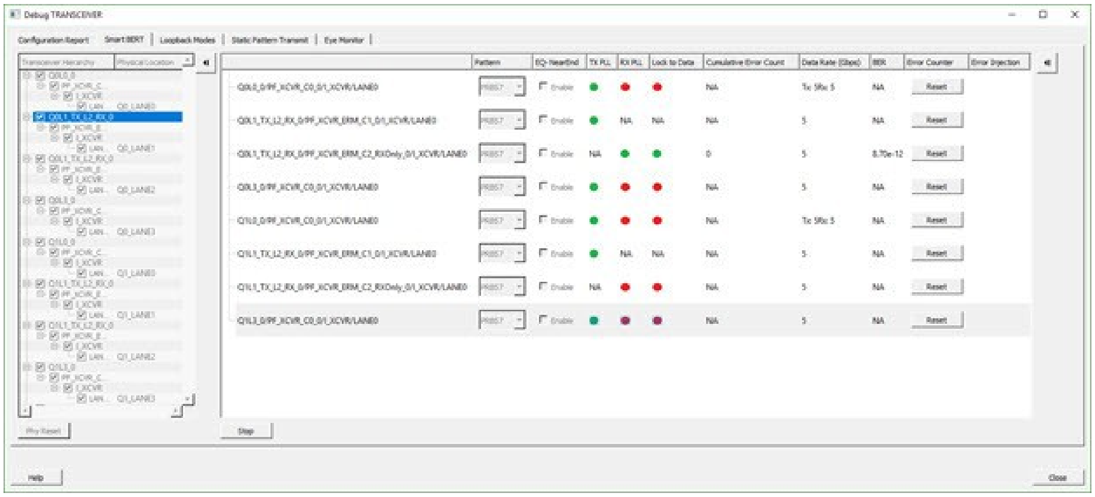
- PF_XCVR_0
  - Lane 0
  - Lane 1
  - Lane 2
  - Lane 3
- PF_XCVR_1
  - Lane 0
  - Lane 1
  - Lane 2
  - Lane 3
- PF_XCVR_2
  - Lane 0
  - Lane 1
  - Lane 2
  - Lane 3
- PF_XCVR_3

3.14.3 SmartBERT

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

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

Figure 3-48. Debug TRANSCEIVER - SmartBERT



☑ Q0L0_0
☑ PF_XCVR_C...
☑ I_XCVR
☑ LAN... Q0_LANE0
☑ Q0L1_TX_L2_RX_0
☑ PF_XCVR_E...
☑ I_XCVR
☑ LAN... Q0_LANE1
☑ Q0L1_TX_L2_RX_0
☑ PF_XCVR_E...
☑ I_XCVR
☑ LAN... Q0_LANE2
☑ Q0L3_0
☑ PF_XCVR_C...
☑ I_XCVR
☑ LAN... Q0_LANE3
☑ Q1L0_0
☑ PF_XCVR_C...
☑ I_XCVR
☑ LAN... Q1_LANE0
☑ Q1L1_TX_L2_RX_0
☑ PF_XCVR_E...
☑ I_XCVR
☑ LAN... Q1_LANE1
☑ Q1L1_TX_L2_RX_0
☑ PF_XCVR_E...
☑ I_XCVR
☑ LAN... Q1_LANE2
☑ Q1L3_0
☑ PF_XCVR_C...
☑ I_XCVR
☑ LAN... Q1_LANE3

	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)	BER	Error Counter	Error Injection
Q0L0_0/PF_XCVR_C0_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	🔴	🔴	NA	Tx: 5Rx: 5	NA	Reset	
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	NA	NA	NA	5	NA	Reset	
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0	PRBS7	□ Enable	NA	🟢	🟢	0	5	8.70e-12	Reset	
Q0L3_0/PF_XCVR_C0_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	🔴	🔴	NA	5	NA	Reset	
Q1L0_0/PF_XCVR_C0_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	🔴	🔴	NA	Tx: 5Rx: 5	NA	Reset	
Q1L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	NA	NA	NA	5	NA	Reset	
Q1L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0	PRBS7	□ Enable	NA	🔴	🔴	NA	5	NA	Reset	
Q1L3_0/PF_XCVR_C0_0/I_XCVR/LANE0	PRBS7	□ Enable	🟢	🔴	🔴	NA	5	NA	Reset	

The following input options and outputs are represented as columns:

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

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

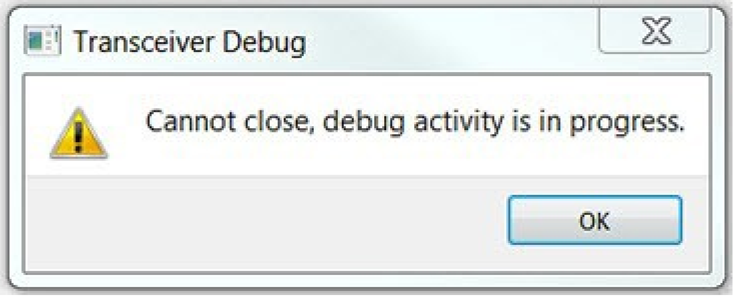
Select the desired options and click **Start** to start the Smart BERT test on all selected lanes. A popup message appears if a test cannot be started on one lane, multiple lanes, or all lanes. Tests will start normally on all unaffected lanes.

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

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

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

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



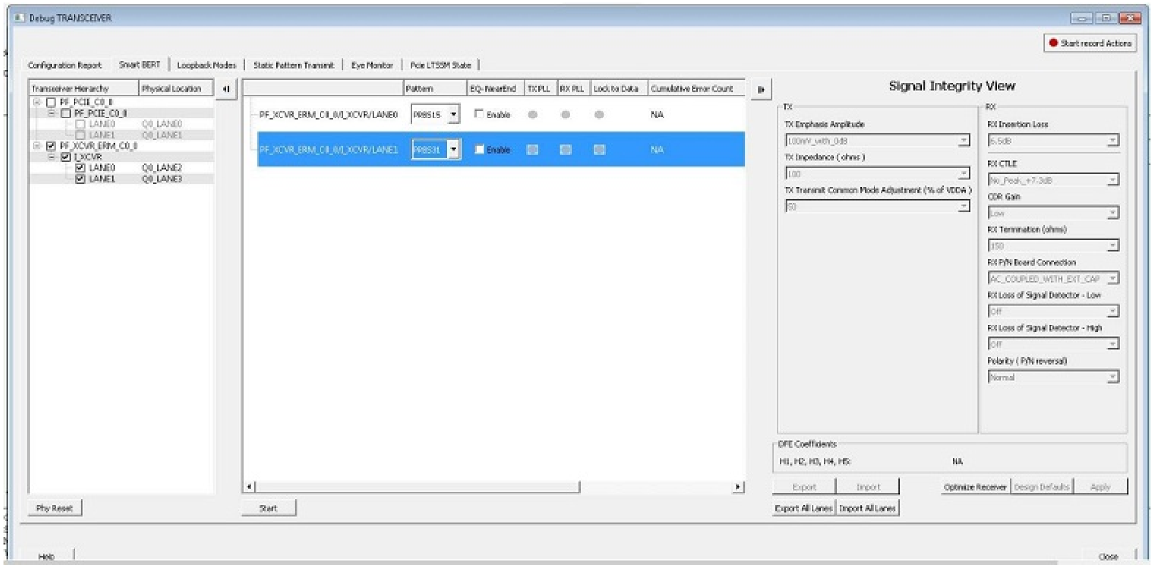
Transceiver Debug

Cannot close, debug activity is in progress.

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

Note: SmartBERT tests are disabled for PCIe lanes.

Figure 3-49. PCIe Lanes with SmartBERT Tests Disabled



Transceiver Hierarchy	Physical Location
□ PF_PCIE_C0_0	
□ PF_PCIE_C0_0	
□ LANE0	Q0_LANE0
□ LANE1	Q0_LANE1
☑ PF_XCVR_ERM_C0_0	
☑ I_XCVR	
☑ LANE0	Q0_LANE2
☑ LANE1	Q0_LANE3

	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count
PF_XCVR_ERM_C0_0/I_XCVR/LANE0	PRBS15	□ Enable	●	●	●	NA
PF_XCVR_ERM_C0_0/I_XCVR/LANE1	PRBS31	□ Enable	■	■	■	NA

Signal Integrity View

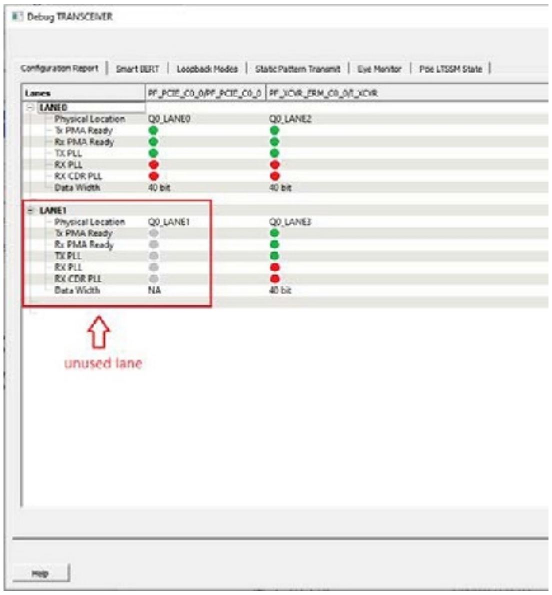
TX
TX Emphasis Amplitude: 100mV_with_0dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50

RX
RX Insertion Loss: 6.5dB
RX CTLE: No_Peak_+7.3dB
CDR Gain: Low
RX Termination (ohms): 150
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

DFE Coefficients
H1, H2, H3, H4, H5: NA

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

Figure 3-50. Configuration Report Showing Unused PCIe Lane



Lanes	PF_PCIE_C0_0/PF_PCIE_C0_0	PF_XCVR_ERM_C0_0/I_XCVR
LANE0		
Physical Location	Q0_LANE0	Q0_LANE2
Tx PMA Ready	🟢	🟢
Rx PMA Ready	🟢	🟢
TX PLL	🟢	🟢
RX PLL	🔴	🔴
RX CDR PLL	🔴	🔴
Data Width	40 bit	40 bit
LANE1		
Physical Location	Q0_LANE1	Q0_LANE3
Tx PMA Ready	⚪	🟢
Rx PMA Ready	⚪	🟢
TX PLL	⚪	🟢
RX PLL	⚪	🔴
RX CDR PLL	⚪	🔴
Data Width	NA	40 bit

3.14.3.1 SmartBERT IP

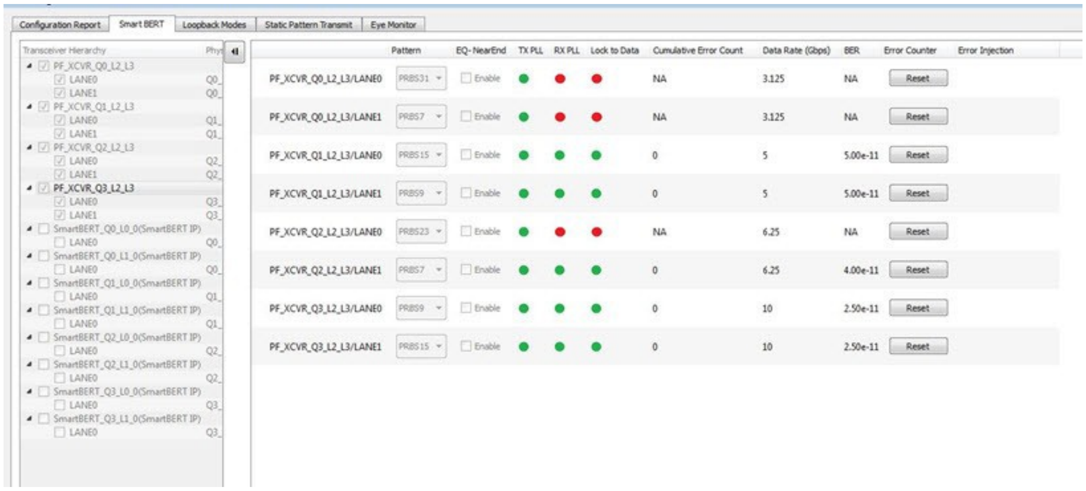
The CoreSmartBERT core provides a broad-based evaluation and demonstration platform for PolarFire transceivers (PF_XCVR). Parameterizable to use different transceivers and clocking topologies, the SmartBERT core can also be customized to use different line rates and reference clock rates. Data pattern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

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

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

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

Figure 3-51. SmartBERT IP - View all Lanes

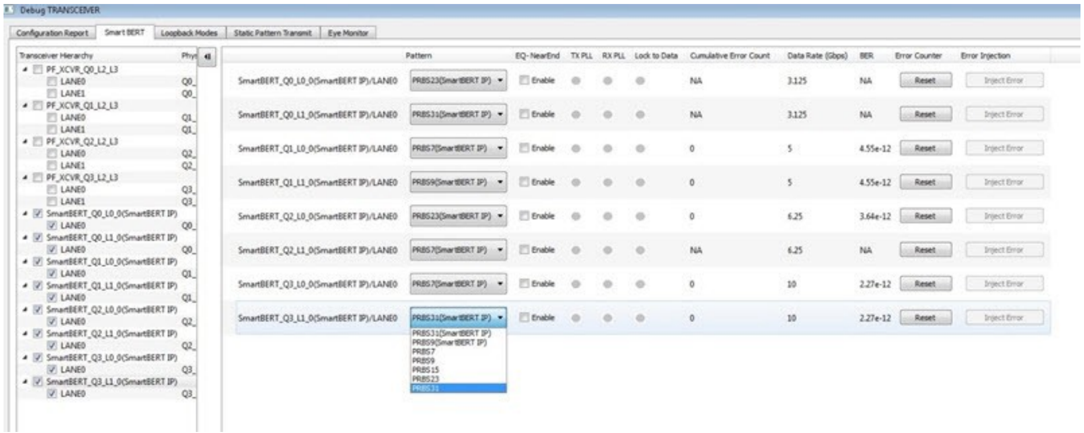


Transceiver Hierarchy	Phy
☑ PF_XCVR_Q0_L2_L3	
☑ LANE0	Q0_
☑ LANE1	Q0_
☑ PF_XCVR_Q1_L2_L3	
☑ LANE0	Q1_
☑ LANE1	Q1_
☑ PF_XCVR_Q2_L2_L3	
☑ LANE0	Q2_
☑ LANE1	Q2_
☑ PF_XCVR_Q3_L2_L3	
☑ LANE0	Q3_
☑ LANE1	Q3_
☐ SmartBERT_Q0_L0_0(SmartBERT IP)	
☐ LANE0	Q0_
☐ SmartBERT_Q0_L1_0(SmartBERT IP)	
☐ LANE0	Q0_
☐ SmartBERT_Q1_L0_0(SmartBERT IP)	
☐ LANE0	Q1_
☐ SmartBERT_Q1_L1_0(SmartBERT IP)	
☐ LANE0	Q1_
☐ SmartBERT_Q2_L0_0(SmartBERT IP)	
☐ LANE0	Q2_
☐ SmartBERT_Q2_L1_0(SmartBERT IP)	
☐ LANE0	Q2_
☐ SmartBERT_Q3_L0_0(SmartBERT IP)	
☐ LANE0	Q3_
☐ SmartBERT_Q3_L1_0(SmartBERT IP)	
☐ LANE0	Q3_

	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)	BER	Error Counter	Error Injection
PF_XCVR_Q0_L2_L3/LANE0	PRBS31	☐ Enable	🟢	🔴	🔴	NA	3.125	NA	Reset	
PF_XCVR_Q0_L2_L3/LANE1	PRBS7	☐ Enable	🟢	🔴	🔴	NA	3.125	NA	Reset	
PF_XCVR_Q1_L2_L3/LANE0	PRBS15	☐ Enable	🟢	🟢	🟢	0	5	5.00e-11	Reset	
PF_XCVR_Q1_L2_L3/LANE1	PRBS9	☐ Enable	🟢	🟢	🟢	0	5	5.00e-11	Reset	
PF_XCVR_Q2_L2_L3/LANE0	PRBS23	☐ Enable	🟢	🔴	🔴	NA	6.25	NA	Reset	
PF_XCVR_Q2_L2_L3/LANE1	PRBS7	☐ Enable	🟢	🟢	🟢	0	6.25	4.00e-11	Reset	
PF_XCVR_Q3_L2_L3/LANE0	PRBS9	☐ Enable	🟢	🟢	🟢	0	10	2.50e-11	Reset	
PF_XCVR_Q3_L2_L3/LANE1	PRBS15	☐ Enable	🟢	🟢	🟢	0	10	2.50e-11	Reset	

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

Figure 3-52. SmartBERT IP - Select a Pattern



	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)	BER	Error Counter	Error Injection
SmartBERT_Q0_L0_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	□ Enable	●	●	●	NA	3.125	NA	Reset	Inject Error
SmartBERT_Q0_L1_0(SmartBERT IP)/LANE0	PRBS31(SmartBERT IP)	□ Enable	●	●	●	NA	3.125	NA	Reset	Inject Error
SmartBERT_Q1_L0_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	●	●	●	0	5	4.55e-12	Reset	Inject Error
SmartBERT_Q1_L1_0(SmartBERT IP)/LANE0	PRBS9(SmartBERT IP)	□ Enable	●	●	●	0	5	4.55e-12	Reset	Inject Error
SmartBERT_Q2_L0_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	□ Enable	●	●	●	0	6.25	3.64e-12	Reset	Inject Error
SmartBERT_Q2_L1_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	●	●	●	NA	6.25	NA	Reset	Inject Error
SmartBERT_Q3_L0_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	●	●	●	0	10	2.27e-12	Reset	Inject Error
SmartBERT_Q3_L1_0(SmartBERT IP)/LANE0	PRBS31(SmartBERT IP)	□ Enable	●	●	●	0	10	2.27e-12	Reset	Inject Error

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

Error Injection

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

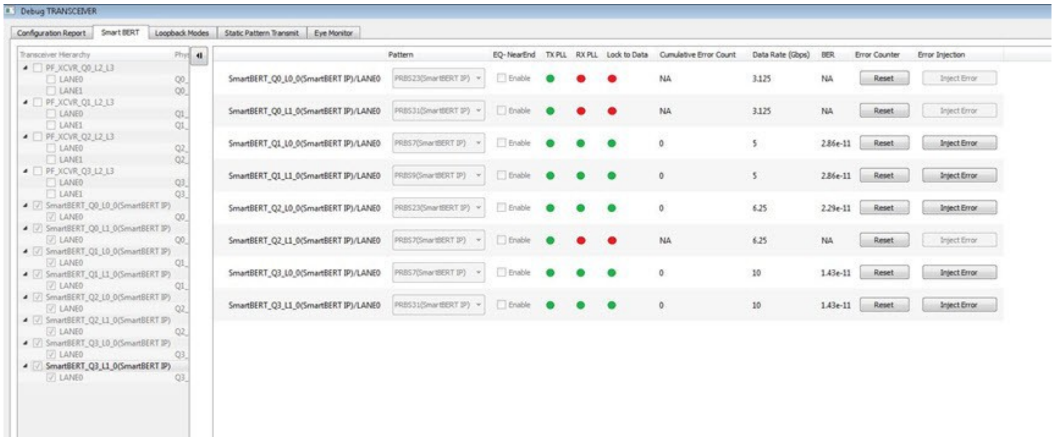
Note: This column does not appear for non-SmartBERT IP lanes, or if a non-configured PRBS pattern has been selected.

Error Count

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

The following figure shows the Reset and Inject Error buttons.

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



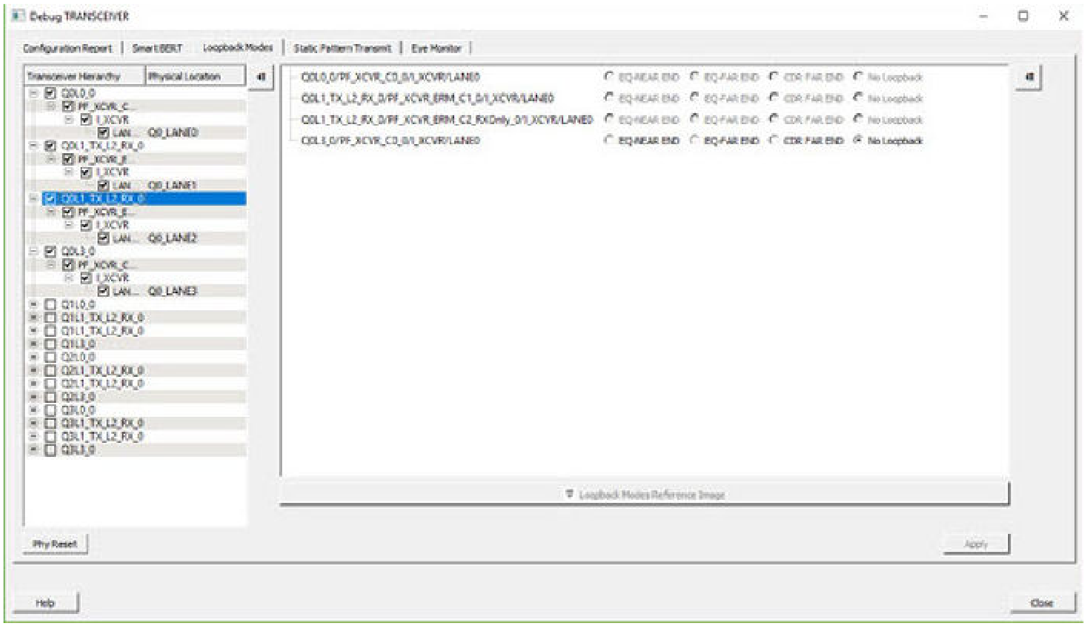
	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)	BER	Error Counter	Error Injection
SmartBERT_Q0_L0_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	□ Enable	🟢	🔴	🔴	NA	3.125	NA	Reset	Inject Error
SmartBERT_Q0_L1_0(SmartBERT IP)/LANE0	PRBS31(SmartBERT IP)	□ Enable	🟢	🔴	🔴	NA	3.125	NA	Reset	Inject Error
SmartBERT_Q1_L0_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	🟢	🟢	🟢	0	5	2.86e-11	Reset	Inject Error
SmartBERT_Q1_L1_0(SmartBERT IP)/LANE0	PRBS9(SmartBERT IP)	□ Enable	🟢	🟢	🟢	0	5	2.86e-11	Reset	Inject Error
SmartBERT_Q2_L0_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	□ Enable	🟢	🟢	🟢	0	6.25	2.29e-11	Reset	Inject Error
SmartBERT_Q2_L1_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	🟢	🔴	🔴	NA	6.25	NA	Reset	Inject Error
SmartBERT_Q3_L0_0(SmartBERT IP)/LANE0	PRBS7(SmartBERT IP)	□ Enable	🟢	🟢	🟢	0	10	1.43e-11	Reset	Inject Error
SmartBERT_Q3_L1_0(SmartBERT IP)/LANE0	PRBS31(SmartBERT IP)	□ Enable	🟢	🟢	🟢	0	10	1.43e-11	Reset	Inject Error

3.14.4 Loopback Modes

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

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

Figure 3-54. Debug TRANSCEIVER - Loopback Modes



Q0L0_0/PF_XCVR_C0_0/I_XCVR/LANE0 ○ EQ-NEAR END ○ EQ-FAR END ○ CDR FAR END ○ No Loopback
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0 ○ EQ-NEAR END ○ EQ-FAR END ○ CDR FAR END ○ No Loopback
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0 ○ EQ-NEAR END ○ EQ-FAR END ○ CDR FAR END ○ No Loopback
Q0L3_0/PF_XCVR_C3_0/I_XCVR/LANE0 ○ EQ-NEAR END ○ EQ-FAR END ○ CDR FAR END ● No Loopback

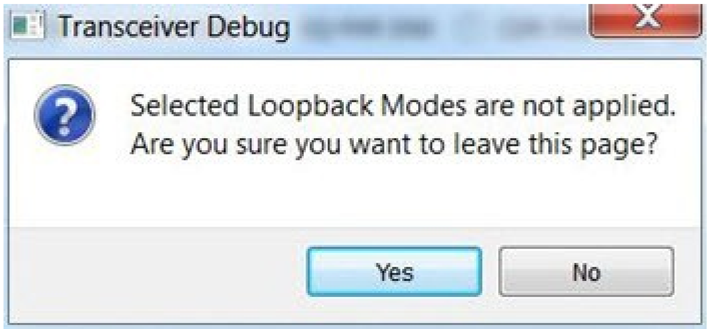
You can select the desired loopback type (EQ-NEAREND, EQ-FAREND, CDRFAREND, or No Loopback) for each lane.

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

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

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

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



Selected Loopback Modes are not applied.
Are you sure you want to leave this page?

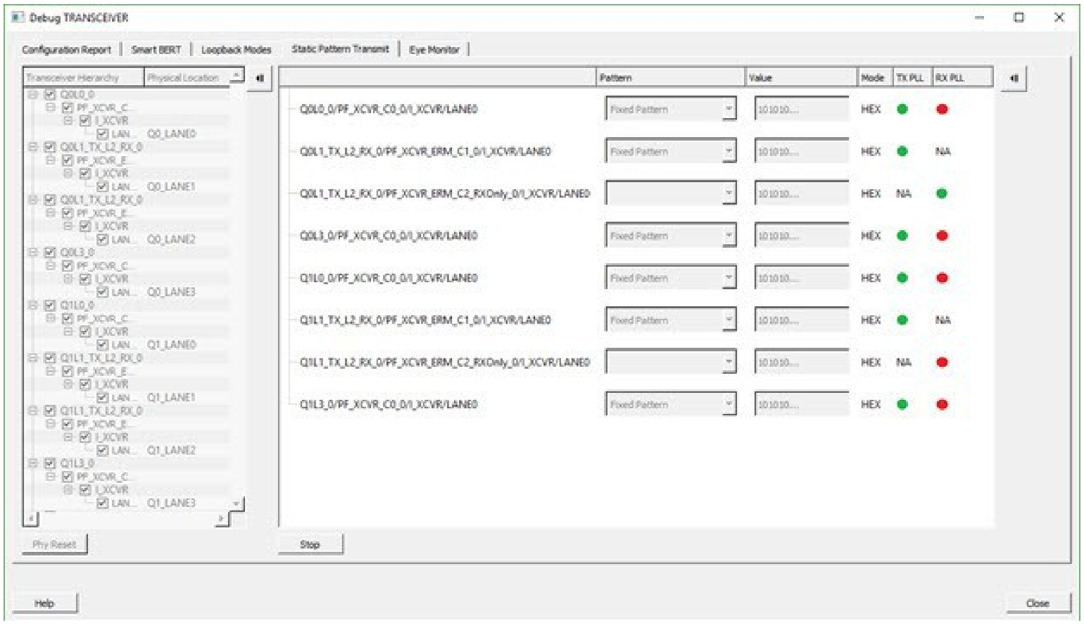
Click **Yes** to ignore the changed selections and move to another selected page. Click **No** to remain on the current page.

3.14.5 Static Pattern Transmit

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

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

Figure 3-55. Debug TRANSCEIVER – Static Pattern Transmit



	Pattern	Value	Mode	TX PLL	RX PLL
Q0L0_0/PF_XCVR_C0_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	🔴
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	NA
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0		101010...	HEX	NA	🟢
Q0L3_0/PF_XCVR_C0_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	🔴
Q1L0_0/PF_XCVR_C0_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	🔴
Q1L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	NA
Q1L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0		101010...	HEX	NA	🔴
Q1L3_0/PF_XCVR_C0_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	🟢	🔴

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

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

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

3.14.6 Eye Monitor

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

To use the Eye Monitor feature, perform the following:

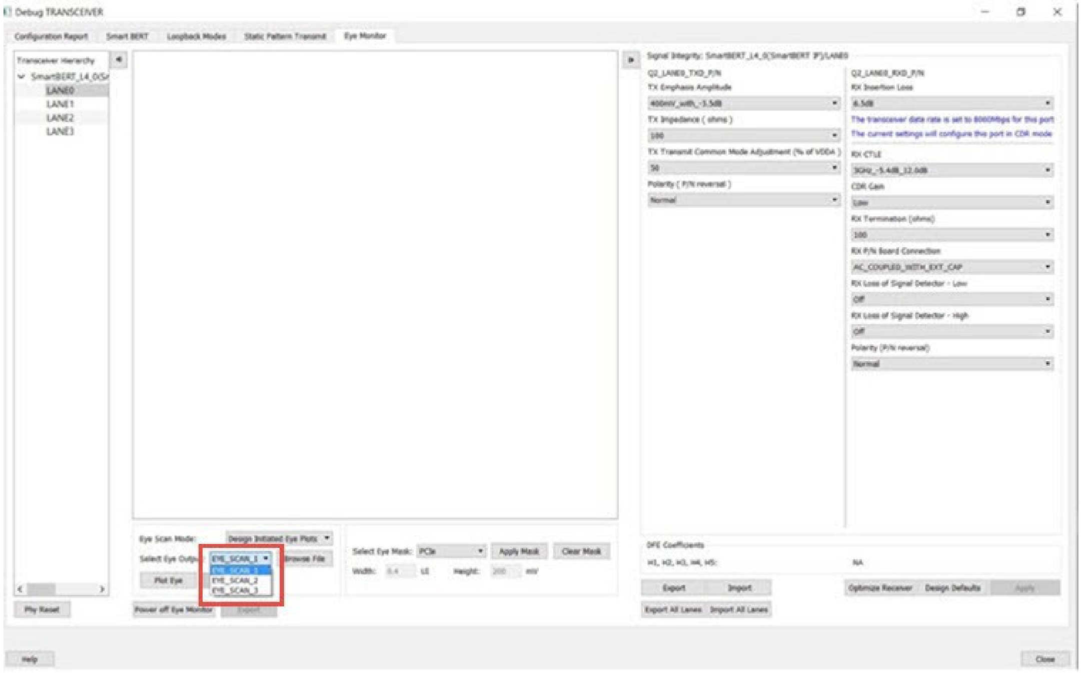
1. Invoke SmartDebug from Libero.
2. Click the **Eye Monitor** tab in the Debug TRANSCEIVER dialog box.

3.14.6.1 Select Eye Output

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

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

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



Signal Integrity: SmartBERT_L4_0(SmartBERT IP)/LANE0

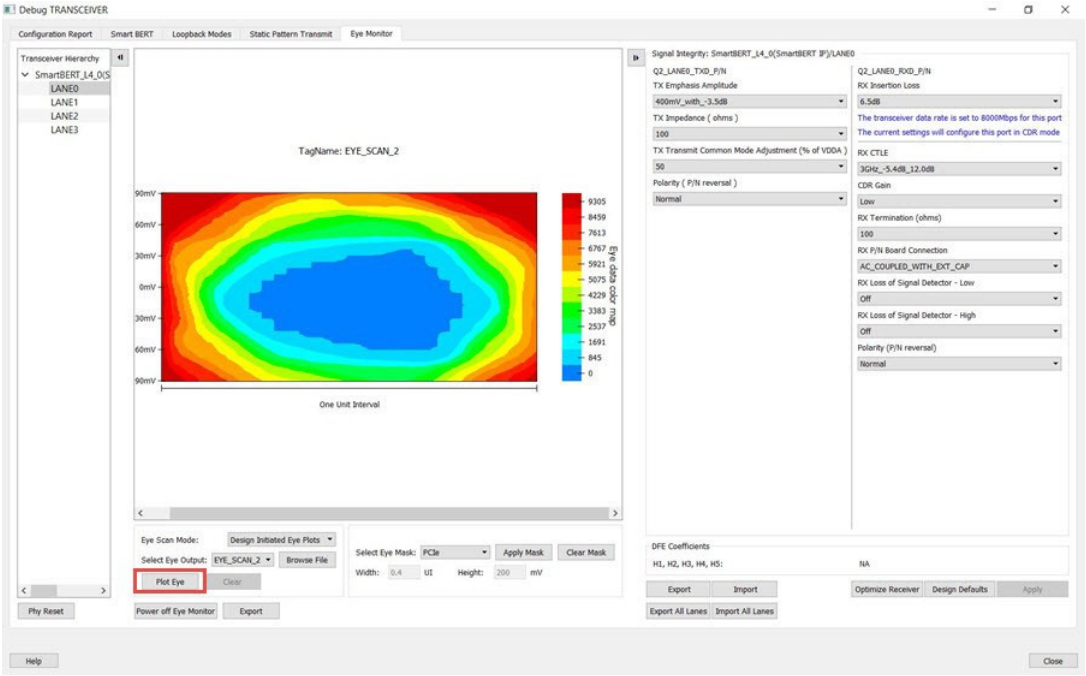
Q2_LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

Q2_LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 8000Mbps for this port
The current settings will configure this port in CDR mode
RX CTLE: 3GHz_-5.4dB_12.0dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

Eye Scan Mode: Design Initiated Eye Plots
Select Eye Output: EYE_SCAN_1
EYE_SCAN_1
EYE_SCAN_2
EYE_SCAN_3
Select Eye Mask: PCIe
Width: 0.4 UI Height: 200 mV
DFE Coefficients
H1, H2, H3, H4, H5: NA

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

Figure 3-57. Eye Monitor Example



Signal Integrity: SmartBERT_L4_0(SmartBERT IP)/LANE0

Q2_LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

Q2_LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 8000Mbps for this port
The current settings will configure this port in CDR mode
RX CTLE: 3GHz_-5.4dB_12.0dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

Eye Scan Mode: Design Initiated Eye Plots
Select Eye Output: EYE_SCAN_2
Select Eye Mask: PCIe
Width: 0.4 UI Height: 200 mV
DFE Coefficients
H1, H2, H3, H4, H5: NA

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

3.14.6.2 Eye Scan Mode

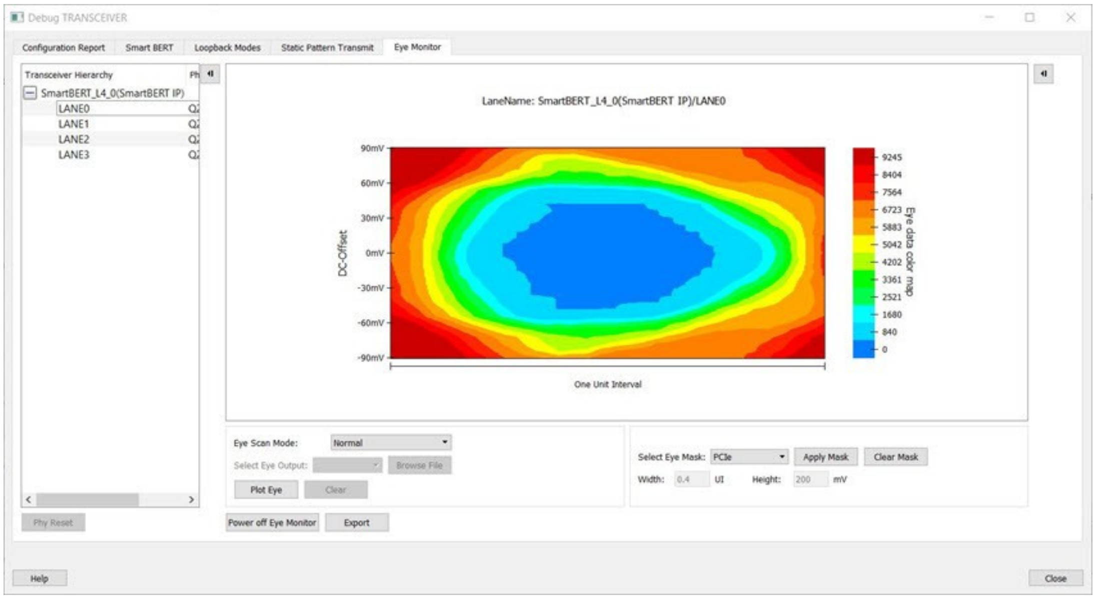
Eye Monitor can be run in two modes: Normal Mode or Infinite Persistent Mode. Choose the desired mode in the **Eye Scan Mode** drop-down.

3.14.6.2.1 Normal Mode

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

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

Figure 3-58. Eye Scan Mode - Normal



LaneName: SmartBERT_L4_0(SmartBERT IP)/LANE0

Eye Scan Mode: Normal

Select Eye Output:

Select Eye Mask: PCIe

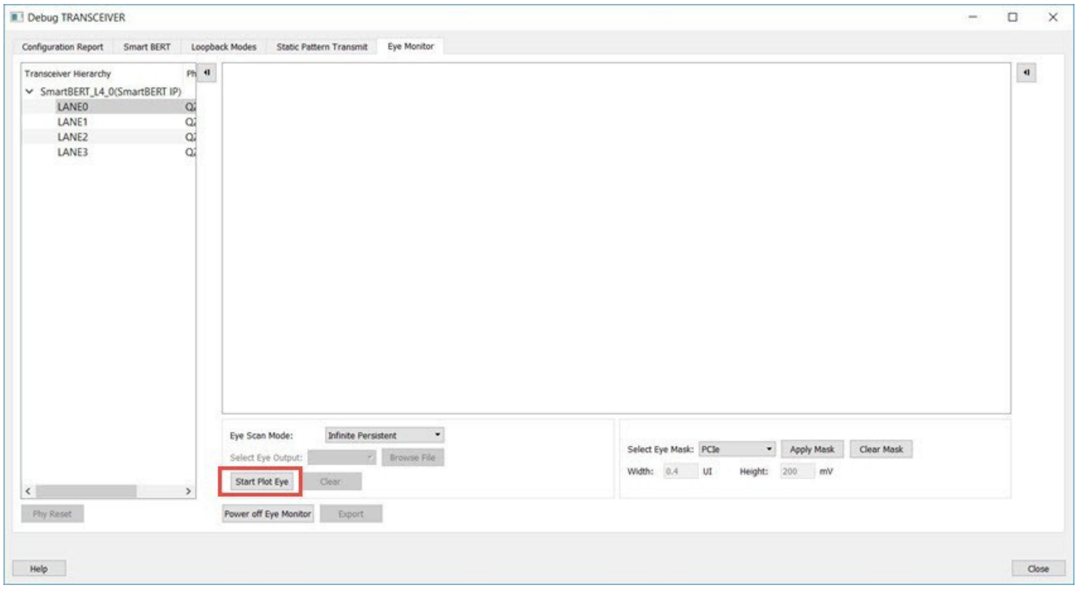
Width: 0.4 UI Height: 200 mV

3.14.6.2.2 Infinite Persistent Mode

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

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

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



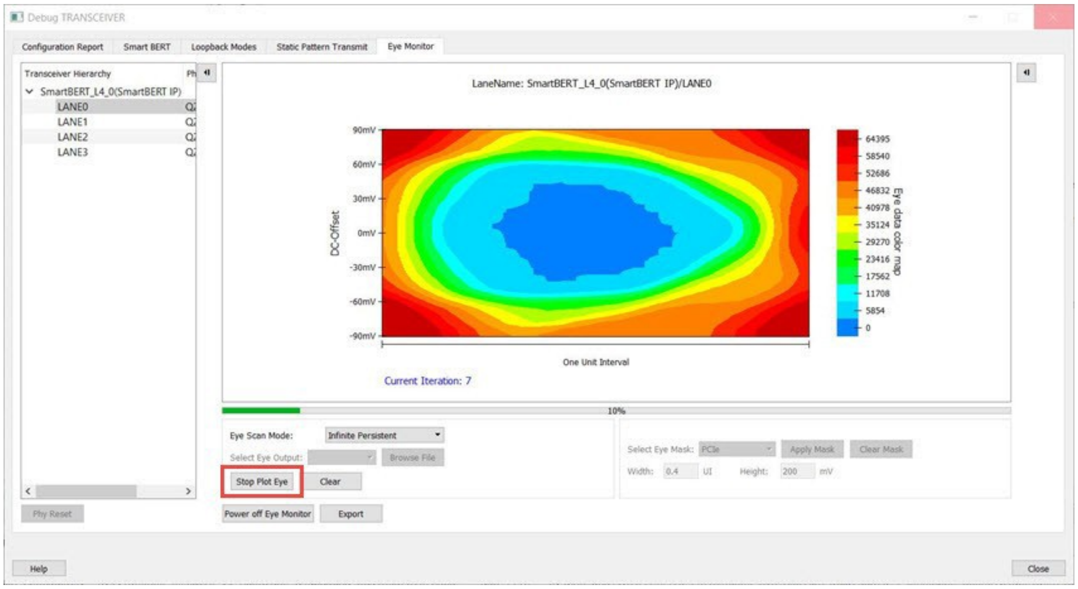
Eye Scan Mode: Infinite Persistent

Select Eye Mask: PCIe

Width: 0.4 UI Height: 200 mV

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

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



LaneName: SmartBERT_L4_0(SmartBERT IP)/LANE0

Current Iteration: 7

Eye Scan Mode: Infinite Persistent

Select Eye Mask: PCIe

Width: 0.4 UI Height: 200 mV

3.14.6.2.3 Clear

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

3.14.6.2.4 Additional Eye Output Text Files

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

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

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

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

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

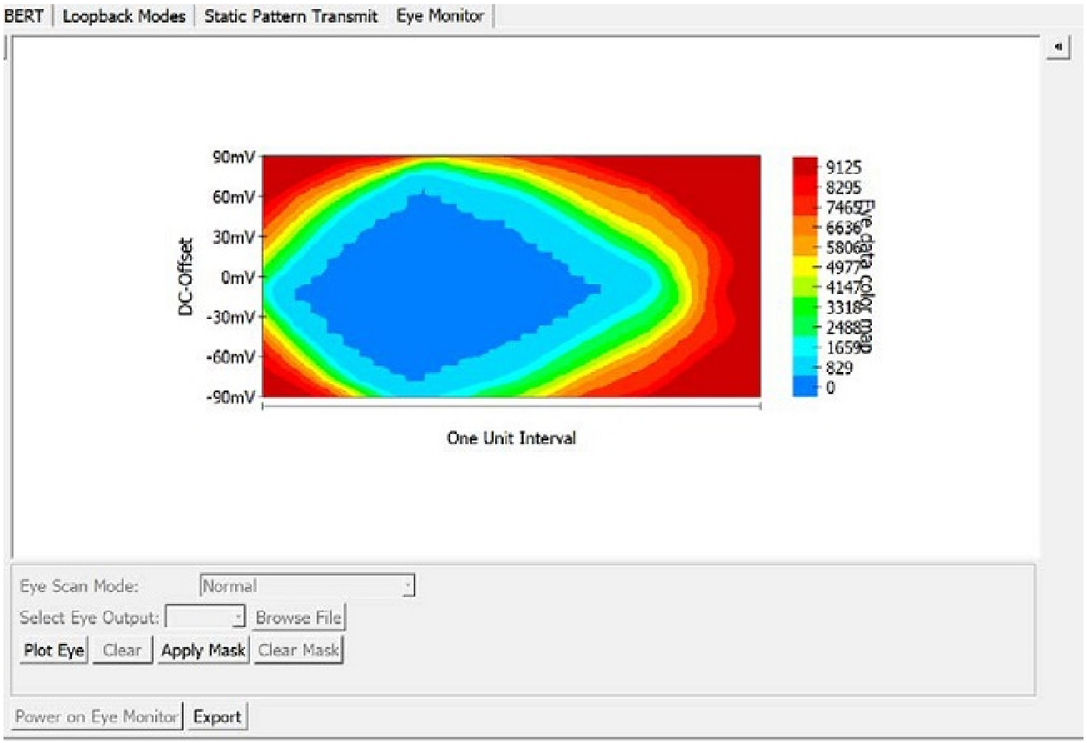
3.14.6.2.5 Error Handling

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

3.14.6.2.6 Eye Mask

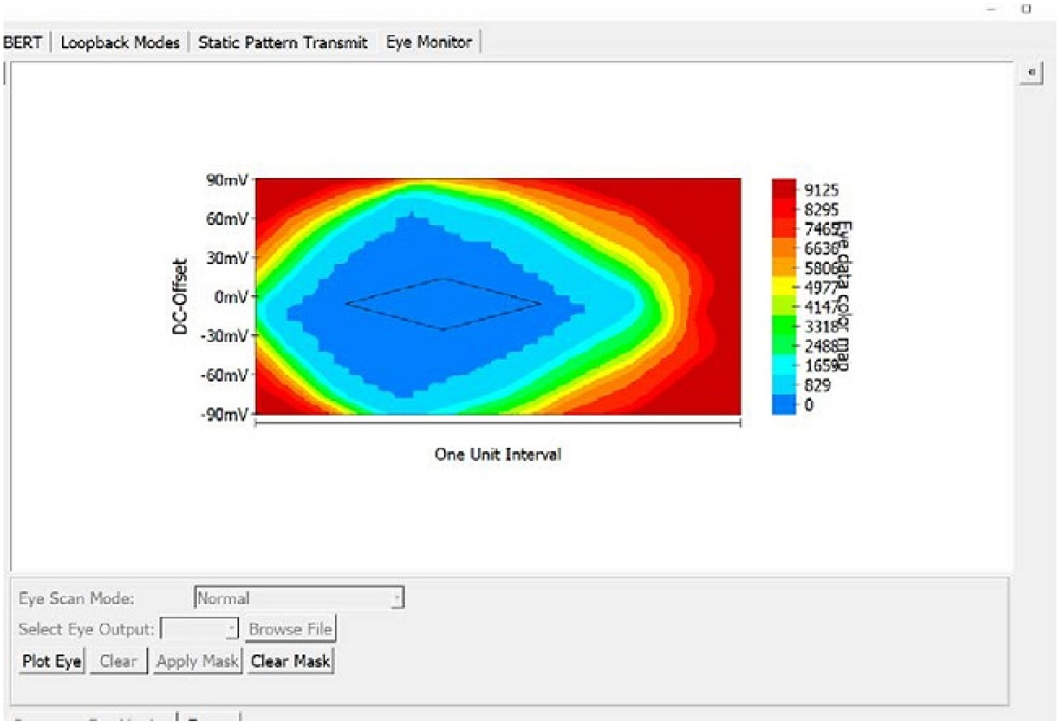
The Eye mask feature has been added to both the normal and infinite persistent modes in the Libero SoC v12.5 release. Eye mask provides a guide to where the best eye opening with least errors can be seen. Both the **Apply Mask** and the **Clear Mask** buttons are disabled in the Default View. Click **Plot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

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



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

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



Eye Scan Mode: Normal

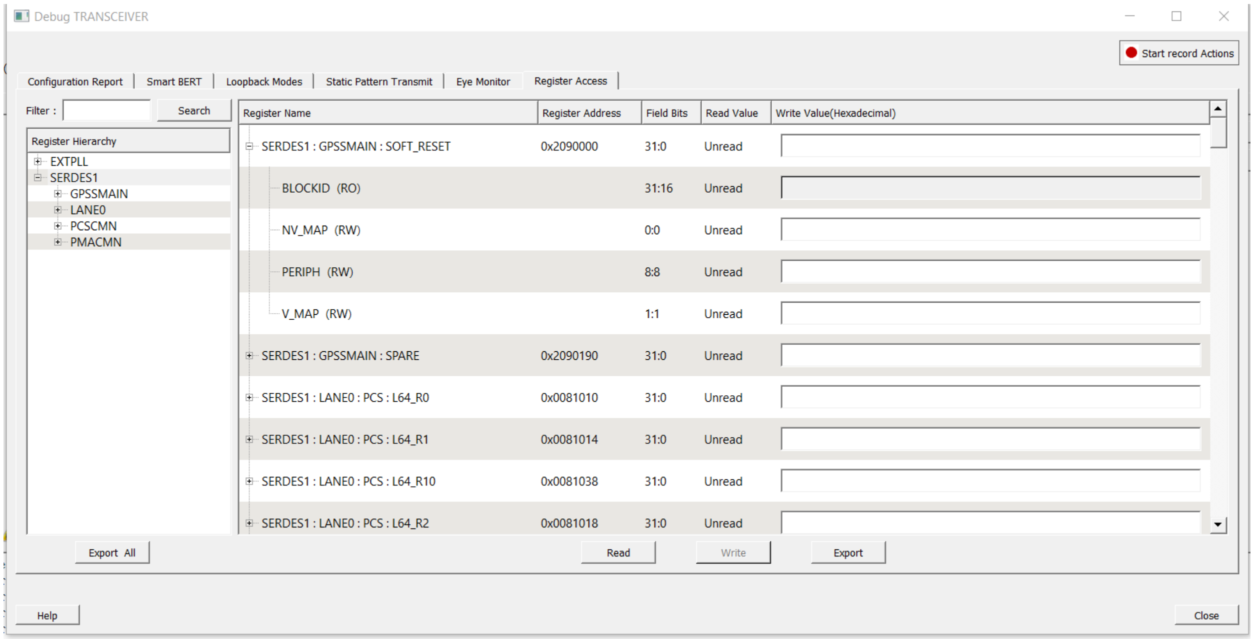
Select Eye Output:

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

3.14.7 Register Access

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

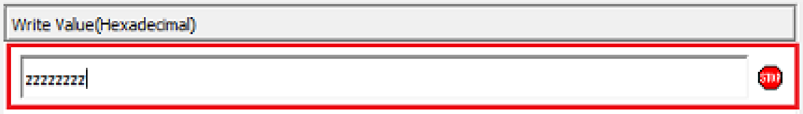
Figure 3-63. Debug TRANSCEIVER - Register Access



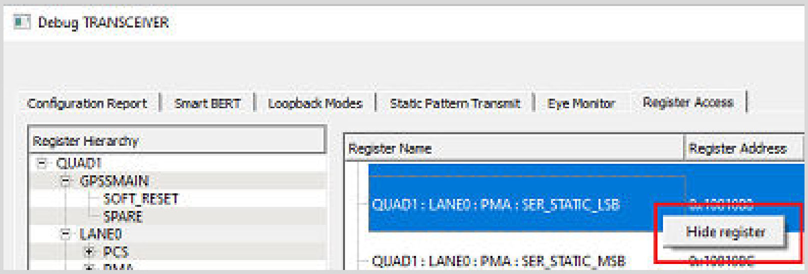
Register Name	Register Address	Field Bits	Read Value	Write Value(Hexadecimal)
SERDES1 : GPSSMAIN : SOFT_RESET	0x2090000	31:0	Unread	
BLOCKID (RO)		31:16	Unread	
NV_MAP (RW)		0:0	Unread	
PERIPH (RW)		8:8	Unread	
V_MAP (RW)		1:1	Unread	
SERDES1 : GPSSMAIN : SPARE	0x2090190	31:0	Unread	
SERDES1 : LANE0 : PCS : L64_R0	0x0081010	31:0	Unread	
SERDES1 : LANE0 : PCS : L64_R1	0x0081014	31:0	Unread	
SERDES1 : LANE0 : PCS : L64_R10	0x0081038	31:0	Unread	
SERDES1 : LANE0 : PCS : L64_R2	0x0081018	31:0	Unread	

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

Table 3-3. Debug TRANSCEIVER - Register Access Options

Option	Description
Register Hierarchy	Lists the design specific registers in a collapsible tree view format.
Register Name	Displays the name of the register that can be looked up in the Register Hierarchy.
Register Address	Displays the physical address of a register. The address is shown in hexadecimal format. The address is calculated based on the Quad, lane number, register type, and offset.
Field Name	Displays the field name based on the register information read from the `PF_XCVR.xml` file.
Field Bits	Displays bit value based on the field offset and the width of the register.
Read Value	Displays the value read from the device. Click the **Read** button to retrieve the value from the device. By default, **unread** is displayed.
Write Value (Hexadecimal)	Enter a hexadecimal register value. An error icon appears if you enter an incorrect or invalid value.  Write Value(Hexadecimal) zzzzzzzz
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.
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.

...........continued

Option	Description
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.  Debug TRANSCEIVER Configuration Report Smart BERT Loopback Modes Static Pattern Transmit Eye Monitor Register Access Register Hierarchy QUAD1 GPSSMAIN SOFT_RESET SPARE LANE0 PCS PMA Register Name Register Address QUAD1 : LANE0 : PMA : SER_STATIC_LSB 0x1001000 Hide register QUAD1 : LANE0 : PMA : SER_STATIC_MSB 0x100100C

3.14.8 Signal Integrity

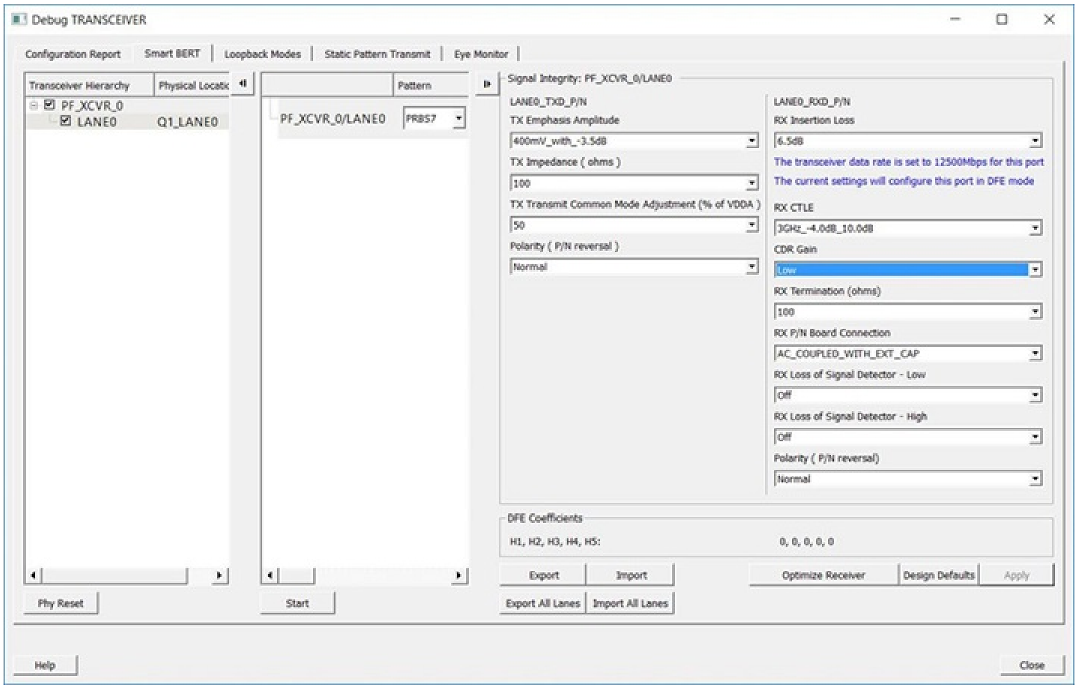
The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of `.pdc` files.

The Signal Integrity pane appears in the following SmartDebug pages:

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

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

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



Signal Integrity: PF_XCVR_0/LANE0

LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 12500Mbps for this port
The current settings will configure this port in DFE mode
RX CTLE: 3GHz_-4.0dB_10.0dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

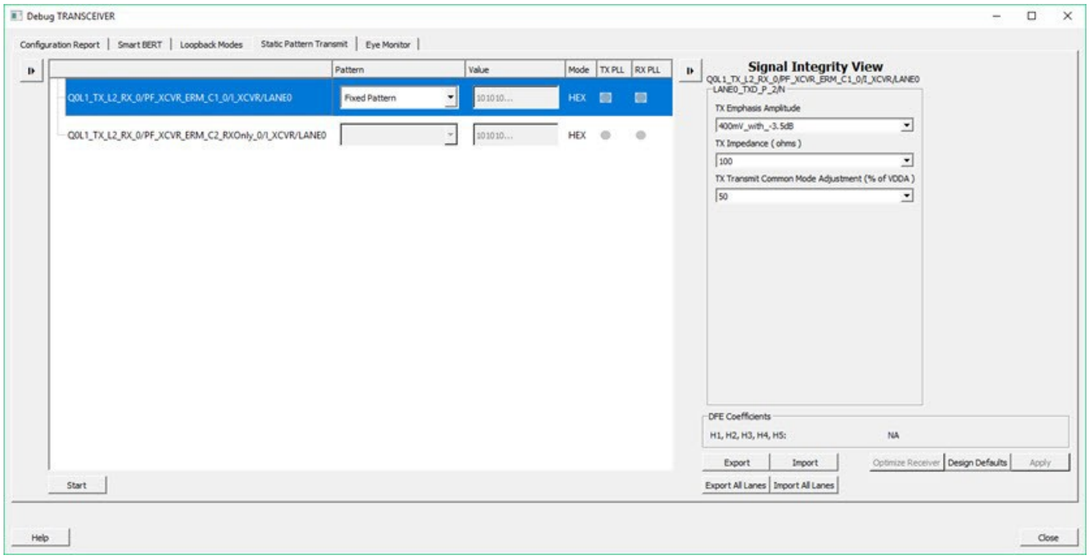
DFE Coefficients
H1, H2, H3, H4, H5: 0, 0, 0, 0, 0

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

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

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

Figure 3-65. Signal Integrity - Tx Only Mode



	Pattern	Value	Mode	TX PLL	RX PLL
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	■	■
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0		101010...	HEX	●	●

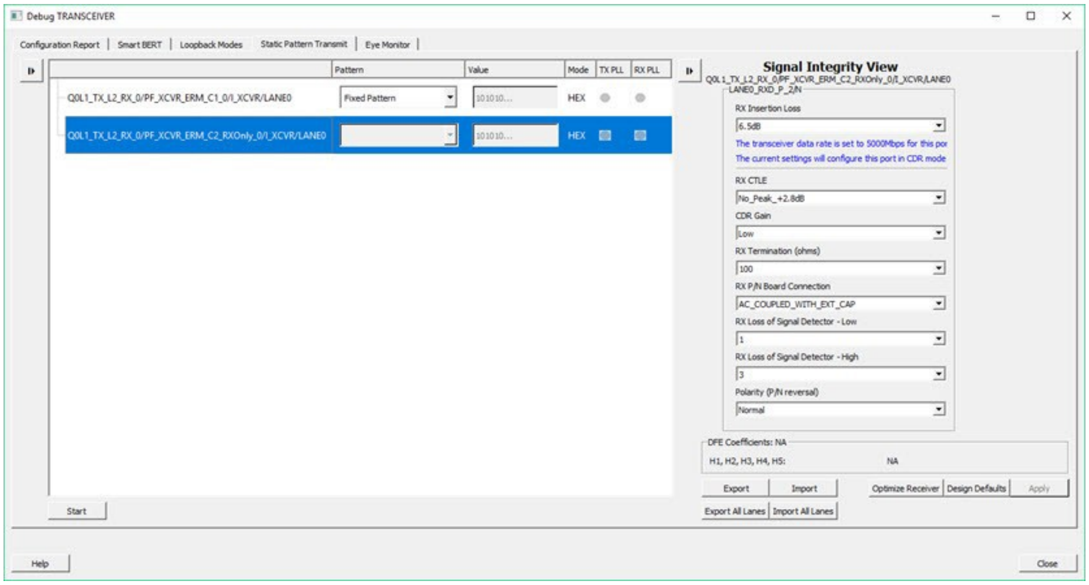
Signal Integrity View
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0
LANE0_TXD_P_2/N

TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50

DFE Coefficients
H1, H2, H3, H4, H5: NA

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

Figure 3-66. Signal Integrity - Rx Only Mode



	Pattern	Value	Mode	TX PLL	RX PLL
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	●	●
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0		101010...	HEX	■	■

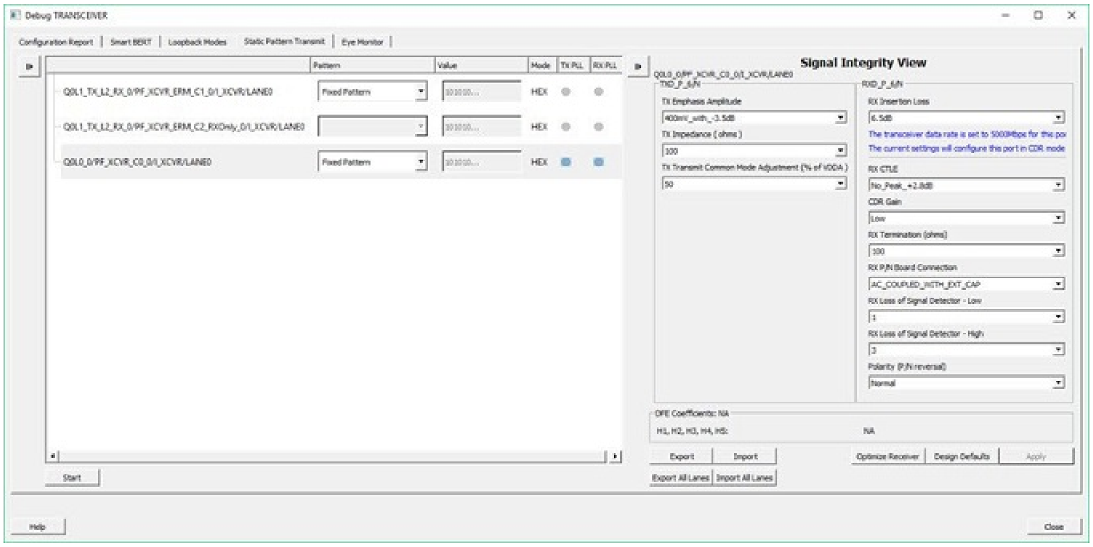
Signal Integrity View
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0
LANE0_RXD_P_2/N

RX Insertion Loss: 6.5dB
The transceiver data rate is set to 5000Mbps for this por
The current settings will configure this port in CDR mode
RX CTLE: No_Peak_+2.8dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: 1
RX Loss of Signal Detector - High: 3
Polarity (P/N reversal): Normal

DFE Coefficients: NA
H1, H2, H3, H4, H5: NA

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

Figure 3-67. Signal Integrity - Independent TxRx Mode



	Pattern	Value	Mode	TX PLL	RX PLL
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C1_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	●	●
Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0		101010...	HEX	●	●
Q0L0_0/PF_XCVR_C0_0/I_XCVR/LANE0	Fixed Pattern	101010...	HEX	●	●

Q0L0_0/PF_XCVR_C0_0/I_XCVR/LANE0

TXD_P_&/N

TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50

RXD_P_&/N

RX Insertion Loss: 6.5dB
The transceiver data rate is set to 5000Mbps for this po
The current settings will configure this port in CDR mode
RX CTLE: No_Peak_+2.8dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: 1
RX Loss of Signal Detector - High: 3
Polarity (P/N reversal): Normal

DFE Coefficients: NA
H1, H2, H3, H4, H5: NA

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

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

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

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

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

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

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

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

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

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

3.14.8.1 Design Defaults

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

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

3.14.8.2 Export

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

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

3.14.8.2.1 Export All Lanes

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

3.14.8.3 Import

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

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

3.14.8.3.1 Import All Lanes

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

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

3.14.8.4 Signal Integrity and Calibration Report

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

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

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

3.14.8.5 Signal Integrity Parameters in Half Duplex Modes

Tx Only XCVR Mode

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

Rx Only XCVR Mode

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

Independent TxRx XCVR Mode

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

3.14.8.6 Optimize Receiver

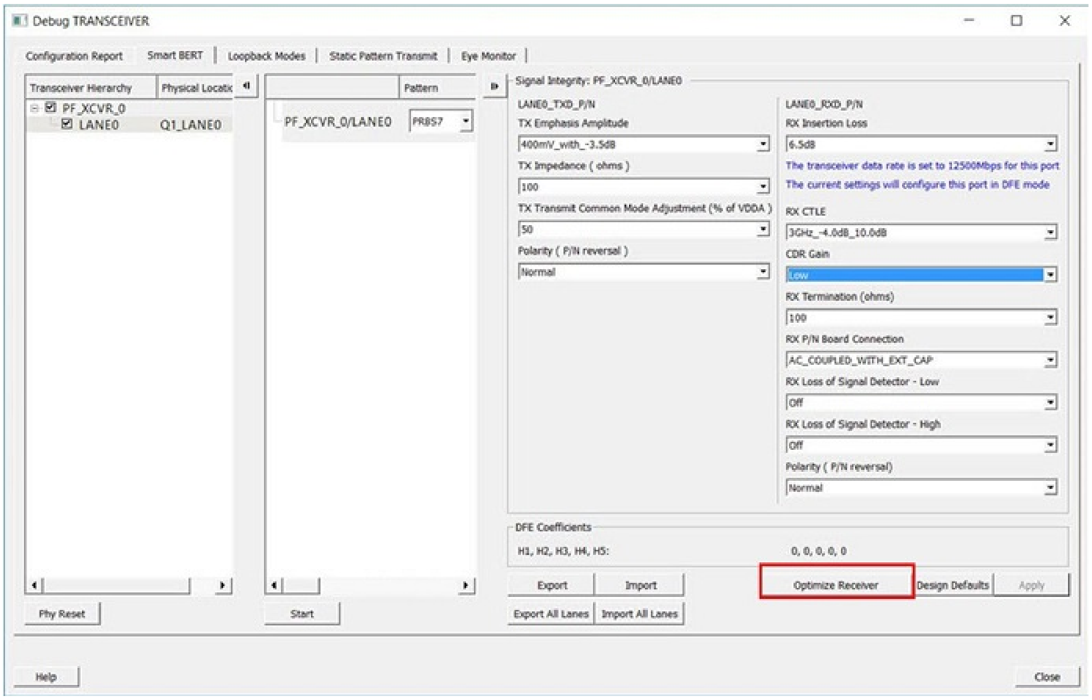
Note: This feature is available for MPF300T, MPF100T, MPF200T, and MPF500T devices.

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

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

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

Figure 3-68. Debug TRANSCEIVER - Optimize Receiver



☑ PF_XCVR_0
☑ LANE0 Q1_LANE0

PF_XCVR_0/LANE0 PRBS7

Signal Integrity: PF_XCVR_0/LANE0

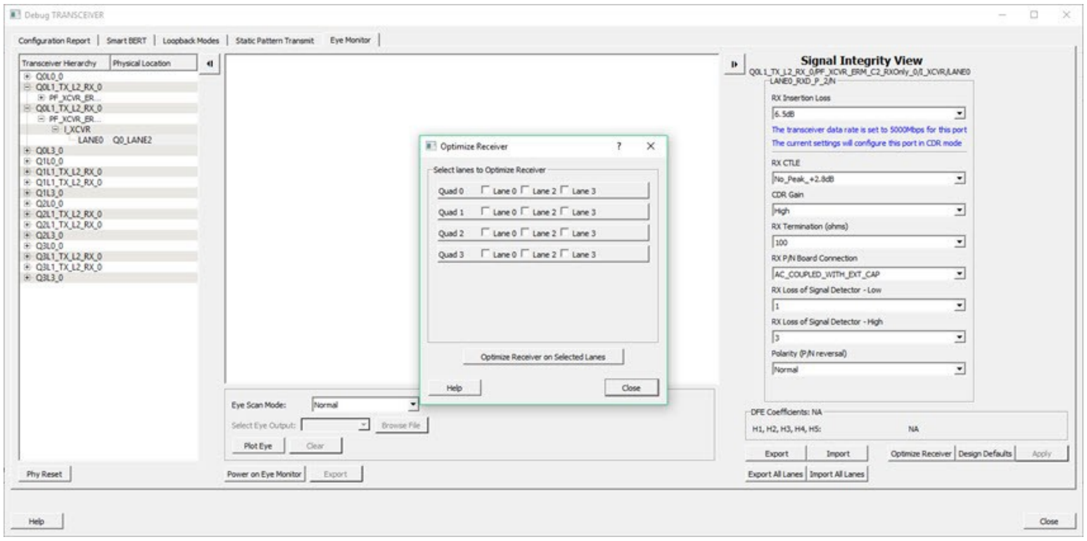
LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 12500Mbps for this port
The current settings will configure this port in DFE mode
RX CTLE: 3GHz_-4.0dB_10.0dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

DFE Coefficients
H1, H2, H3, H4, H5: 0, 0, 0, 0, 0

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

Figure 3-69. Optimize Receiver Dialog Box



Q0L0_0
Q0L1_TX_L2_RX_0
PF_XCVR_ER...
Q0L1_TX_L2_RX_0
PF_XCVR_ER...
I_XCVR
LANE0 Q0_LANE2
Q0L3_0
Q1L0_0
Q1L1_TX_L2_RX_0
Q1L1_TX_L2_RX_0
Q1L3_0
Q2L0_0
Q2L1_TX_L2_RX_0
Q2L1_TX_L2_RX_0
Q2L3_0
Q3L0_0
Q3L1_TX_L2_RX_0
Q3L1_TX_L2_RX_0
Q3L3_0

Quad 0 ☐ Lane 0 ☐ Lane 2 ☐ Lane 3
Quad 1 ☐ Lane 0 ☐ Lane 2 ☐ Lane 3
Quad 2 ☐ Lane 0 ☐ Lane 2 ☐ Lane 3
Quad 3 ☐ Lane 0 ☐ Lane 2 ☐ Lane 3

Eye Scan Mode: Normal

Q0L1_TX_L2_RX_0/PF_XCVR_ERM_C2_RXOnly_0/I_XCVR/LANE0
LANE0_RXD_P_2/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 5000Mbps for this port
The current settings will configure this port in CDR mode
RX CTLE: No_Peak_+2.8dB
CDR Gain: High
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: 1
RX Loss of Signal Detector - High: 3
Polarity (P/N reversal): Normal

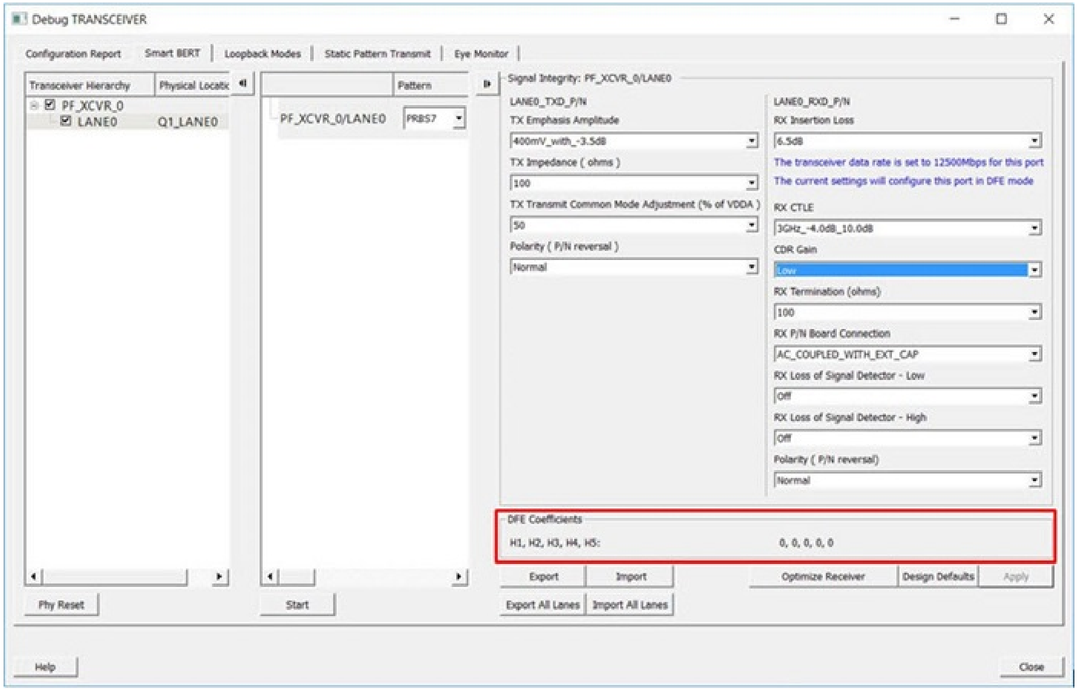
DFE Coefficients: NA
H1, H2, H3, H4, H5: NA

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

3.14.8.7 Display DFE Coefficient Values

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

Figure 3-70. Signal Integrity Pane - DFE Coefficients



☑ PF_XCVR_0
☑ LANE0 Q1_LANE0

PF_XCVR_0/LANE0 PRBS7

Signal Integrity: PF_XCVR_0/LANE0

LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_-3.5dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 12500Mbps for this port
The current settings will configure this port in DFE mode
RX CTLE: 3GHz_-4.0dB_10.0dB
CDR Gain: Low
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

DFE Coefficients
H1, H2, H3, H4, H5: 0, 0, 0, 0, 0

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

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

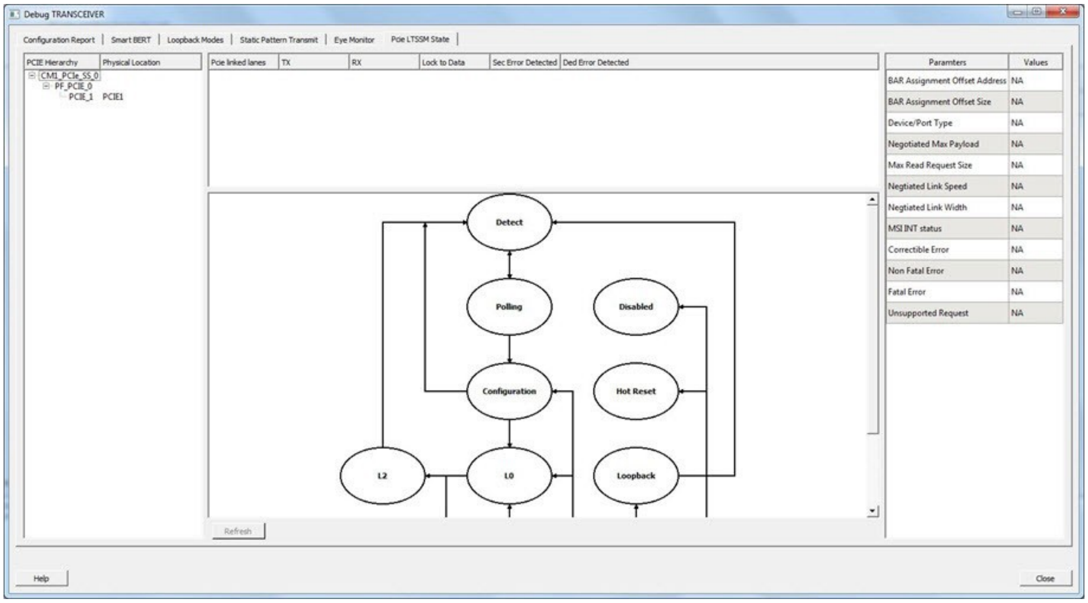
- DFE coefficients are read back from the device in the following scenarios:
  - When the lane is selected in the SmartBert, Loopback Modes, Static Pattern Transmit, and Eye Monitor pages.
  - When a test is started/stopped on selected lane in the SmartBert page.
  - When a test is started/stopped on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

3.14.9 PCIE Debug

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

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

Figure 3-71. PCIE Design Hierarchy



PCIE Hierarchy	Physical Location
CM1_PCIe_SS_0	
PF_PCIE_0	
PCIE_1	PCIE1

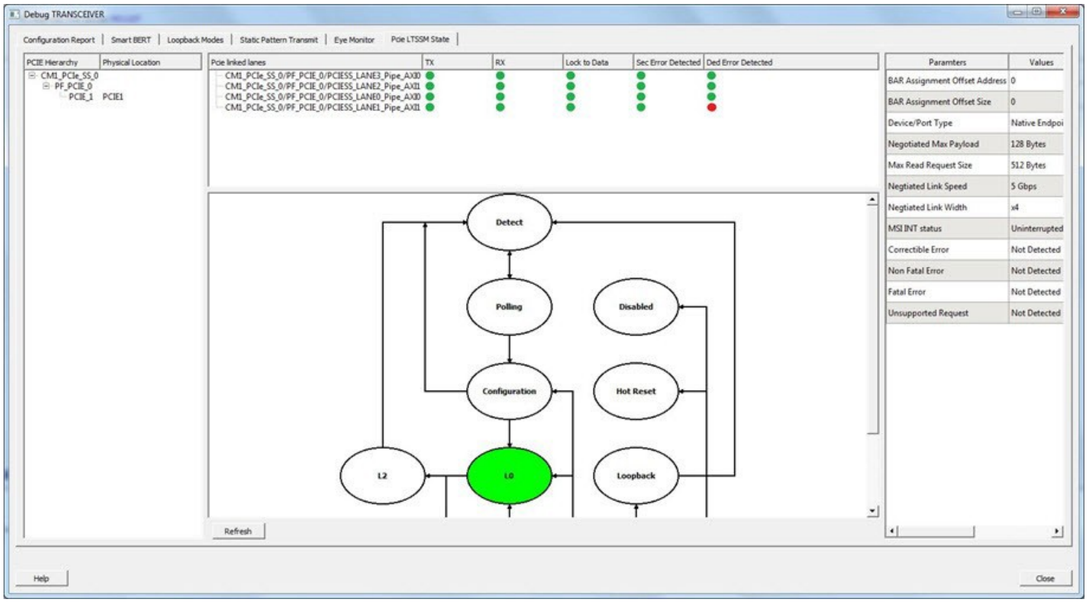
Pcie linked lanes	TX	RX	Lock to Data	Sec Error Detected	Ded Error Detected

Paramters	Values
BAR Assignment Offset Address	NA
BAR Assignment Offset Size	NA
Device/Port Type	NA
Negotiated Max Payload	NA
Max Read Request Size	NA
Negtiated Link Speed	NA
Negtiated Link Width	NA
MSI INT status	NA
Correctible Error	NA
Non Fatal Error	NA
Fatal Error	NA
Unsupported Request	NA

3.14.9.1 Lane Status and Lane Link Error Status

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

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



PCIE Hierarchy	Physical Location
CM1_PCIe_SS_0	
PF_PCIE_0	
PCIE_1	PCIE1

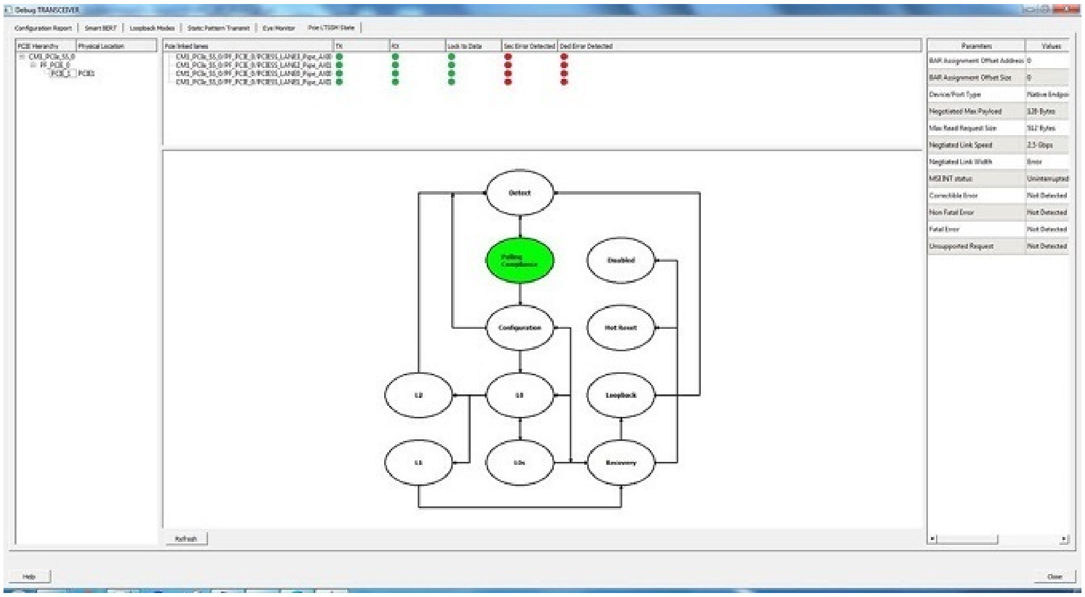
Pcie linked lanes	TX	RX	Lock to Data	Sec Error Detected	Ded Error Detected
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE3_Pipe_AXI0	🟢	🟢	🟢	🟢	🟢
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE2_Pipe_AXI1	🟢	🟢	🟢	🟢	🟢
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE0_Pipe_AXI0	🟢	🟢	🟢	🟢	🟢
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE1_Pipe_AXI1	🟢	🟢	🟢	🟢	🔴

Paramters	Values
BAR Assignment Offset Address	0
BAR Assignment Offset Size	0
Device/Port Type	Native Endpoi
Negotiated Max Payload	128 Bytes
Max Read Request Size	512 Bytes
Negtiated Link Speed	5 Gbps
Negtiated Link Width	x4
MSI INT status	Uninterrupted
Correctible Error	Not Detected
Non Fatal Error	Not Detected
Fatal Error	Not Detected
Unsupported Request	Not Detected

3.14.9.2 LTSSM State Machine

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

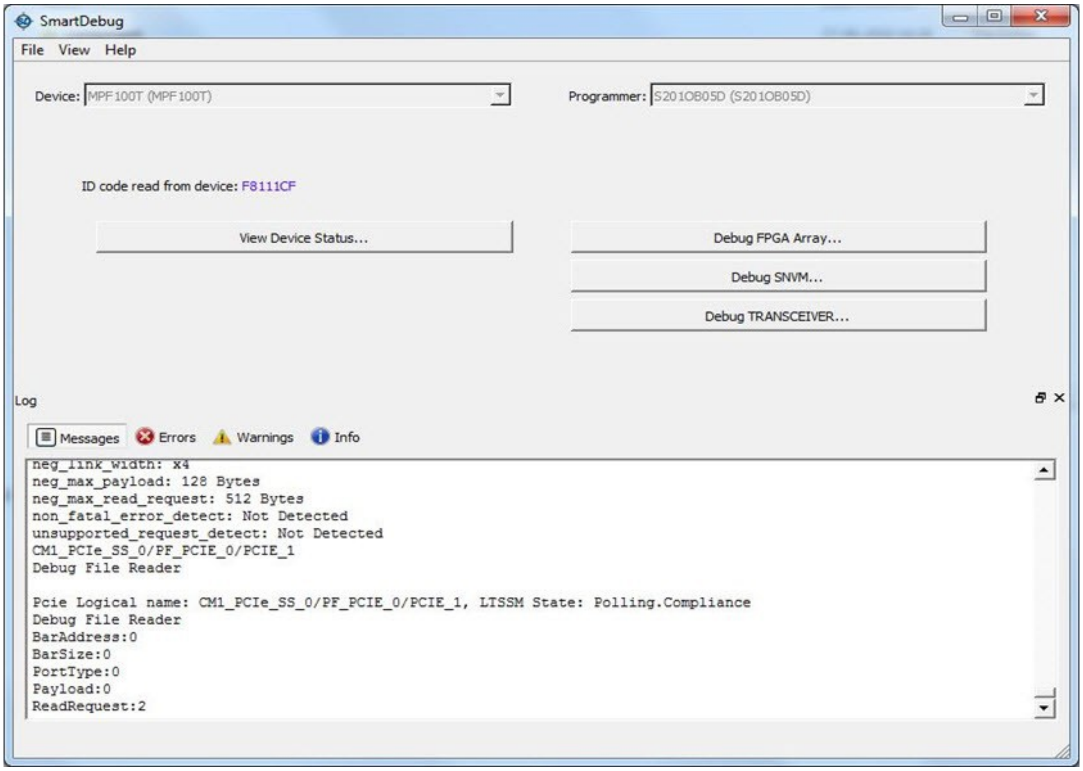
Figure 3-73. LTSSM State Machine



Parameters	Values
BAR Assignment Offset Address	0
BAR Assignment Offset Size	0
Device/Port Type	Native Endpoi
Negotiated Max Payload	128 Bytes
Max Read Request Size	512 Bytes
Negotiated Link Speed	2.5 Gbps
Negotiated Link Width	Error
MSI INT status	Uninterrupted
Correctible Error	Not Detected
Non Fatal Error	Not Detected
Fatal Error	Not Detected
Unsupported Request	Not Detected

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

Figure 3-74. Active State and Substate Information



ID code read from device: F8111CF

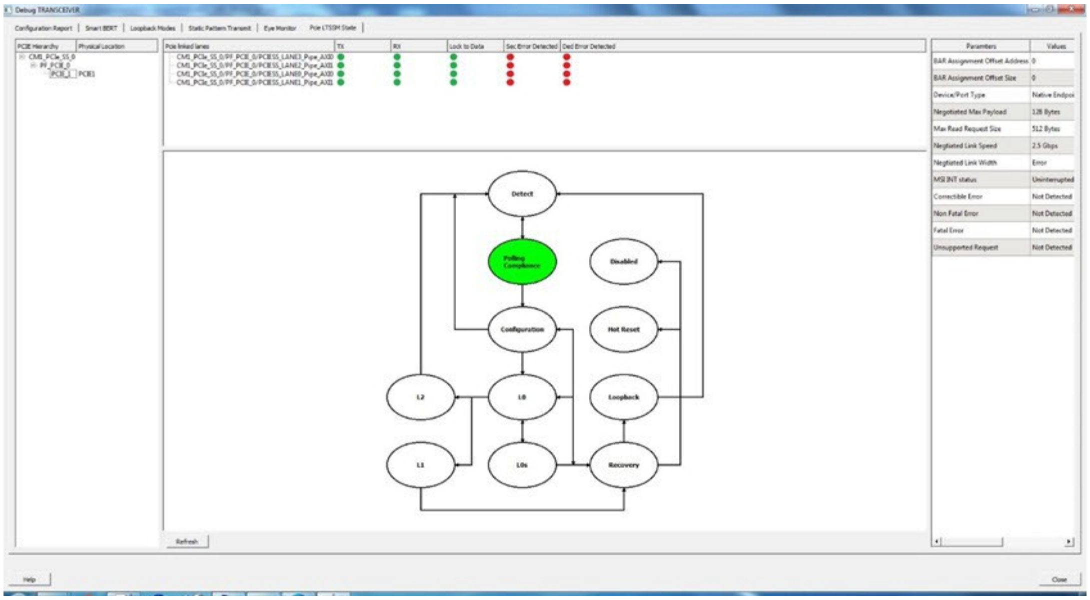
```
neg_link_width: x4
neg_max_payload: 128 Bytes
neg_max_read_request: 512 Bytes
non_fatal_error_detect: Not Detected
unsupported_request_detect: Not Detected
CM1_PCIe_SS_0/PF_PCIE_0/PCIE_1
Debug File Reader

Pcie Logical name: CM1_PCIe_SS_0/PF_PCIE_0/PCIE_1, LTSSM State: Polling.Compliance
Debug File Reader
BarAddress:0
BarSize:0
PortType:0
Payload:0
ReadRequest:2
```

3.14.9.3 Config Space Parameter Information

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

Figure 3-75. Config Space Parameter Information

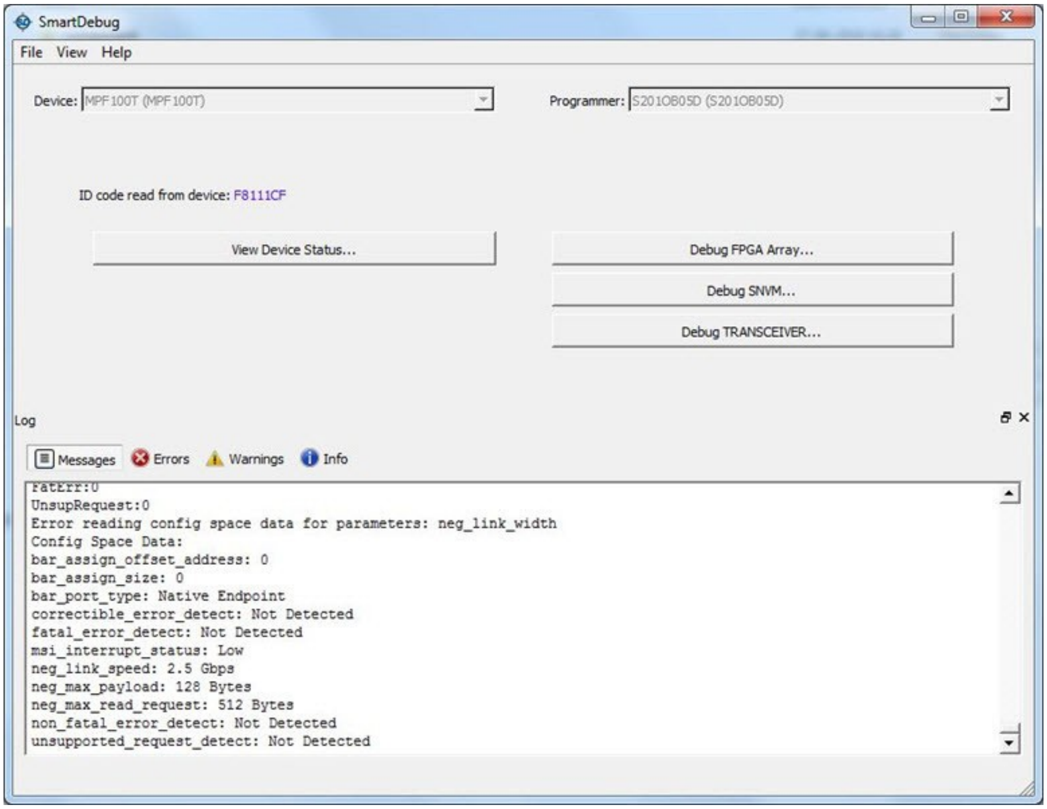


Pcie linked lanes	TX	RX	Lock to Data	Sec Error Detected	Ded Error Detected
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE3_Pipe_AXI0	●	●	●	●	●
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE2_Pipe_AXI1	●	●	●	●	●
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE0_Pipe_AXI0	●	●	●	●	●
CM1_PCIe_SS_0/PF_PCIE_0/PCIESS_LANE1_Pipe_AXI1	●	●	●	●	●

Parameters	Values
BAR Assignment Offset Address	0
BAR Assignment Offset Size	0
Device/Port Type	Native Endpoi
Negotiated Max Payload	128 Bytes
Max Read Request Size	512 Bytes
Negtiated Link Speed	2.5 Gbps
Negtiated Link Width	Error
MSI INT status	Uninterrupted
Correctible Error	Not Detected
Non Fatal Error	Not Detected
Fatal Error	Not Detected
Unsupported Request	Not Detected

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

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



Device: MPF100T (MPF100T)

Programmer: S201OB05D (S201OB05D)

ID code read from device: F8111CF

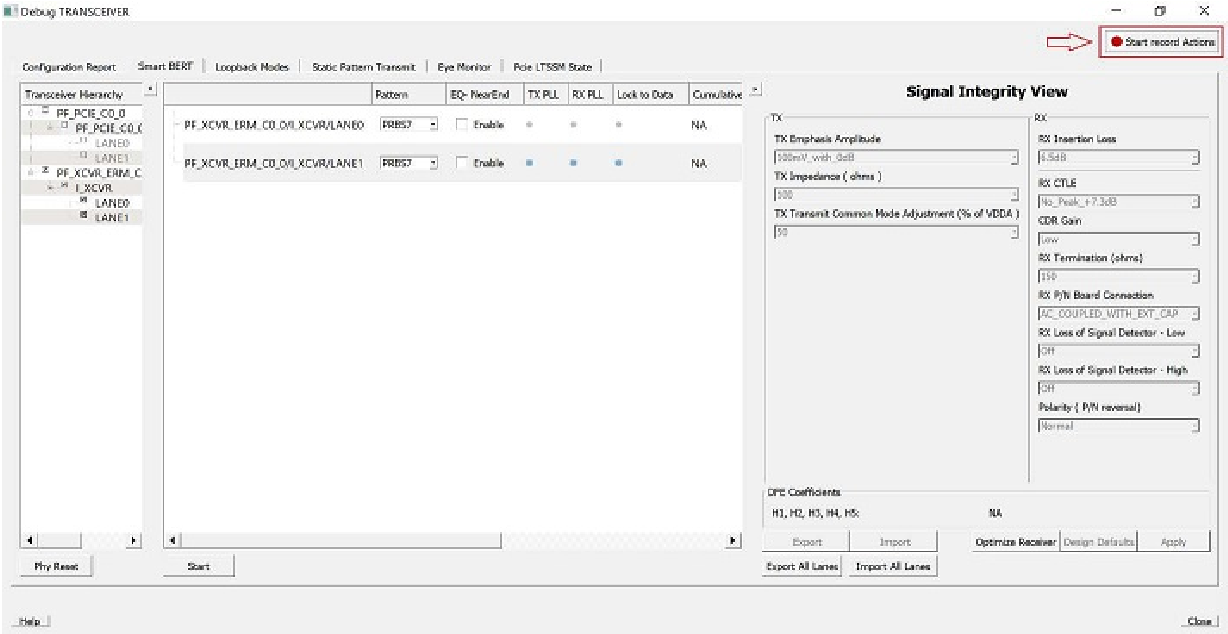
```
FatErr:0
UnsupRequest:0
Error reading config space data for parameters: neg_link_width
Config Space Data:
bar_assign_offset_address: 0
bar_assign_size: 0
bar_port_type: Native Endpoint
correctible_error_detect: Not Detected
fatal_error_detect: Not Detected
msi_interrupt_status: Low
neg_link_speed: 2.5 Gbps
neg_max_payload: 128 Bytes
neg_max_read_request: 512 Bytes
non_fatal_error_detect: Not Detected
unsupported_request_detect: Not Detected
```

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

3.14.10 Record Actions

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

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



PF_XCVR_ERM_C0_0/I_XCVR/LANE0 PRBS7 □ Enable NA
PF_XCVR_ERM_C0_0/I_XCVR/LANE1 PRBS7 □ Enable NA

TX
TX Emphasis Amplitude: 300mV_with_0dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50

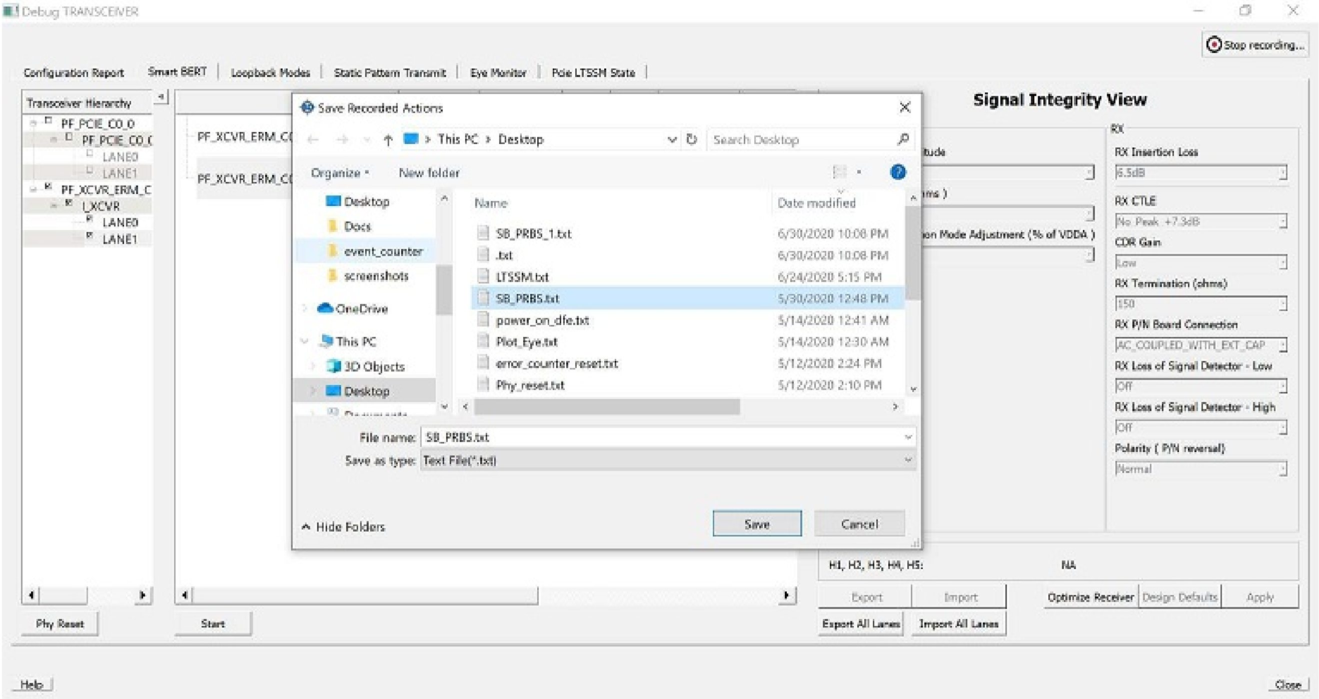
RX
RX Insertion Loss: 6.5dB
RX CTLE: No_Peak_+7.3dB
CDR Gain: Low
RX Termination (ohms): 150
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal

DFE Coefficients
H1, H2, H3, H4, H5: NA

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

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

Figure 3-78. Save Recorded Action Window Pop-Up to Save Text File After ‘Stop recording…’ is Clicked



Name	Date modified
SB_PRBS_1.txt	6/30/2020 10:08 PM
.txt	6/30/2020 10:08 PM
LTSSM.txt	6/24/2020 5:15 PM
SB_PRBS.txt	5/30/2020 12:48 PM
power_on_dfe.txt	5/14/2020 12:41 AM
Plot_Eye.txt	5/14/2020 12:30 AM
error_counter_reset.txt	5/12/2020 2:24 PM
Phy_reset.txt	5/12/2020 2:10 PM

File name: SB_PRBS.txt
Save as type: Text File(*.txt)

RX Insertion Loss: 6.5dB
RX CTLE: No_Peak_+7.3dB
CDR Gain: Low
RX Termination (ohms): 150
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: Off
RX Loss of Signal Detector - High: Off
Polarity (P/N reversal): Normal
H1, H2, H3, H4, H5: NA

3.14.10.1 Recorded Content

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

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

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

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

Example 1

1. Start PRBS test on LANE2 QUAD0 using PRBS7 means:
   1.1. The lane selected on the SmartBERT tab is Q0_ANE0 and the pattern selected is XCVR PMA PRBS7.
   1.2. The first set of lines has four subheadings which mean the following:
   Register Name: Register name that can be looked up in the register map.

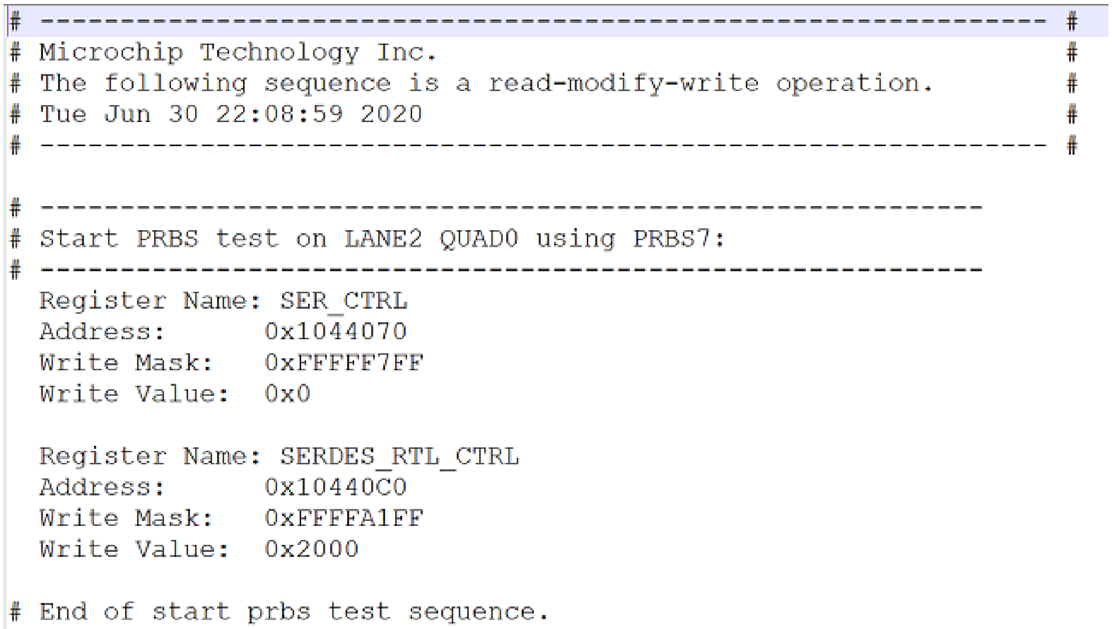
   Address: The physical address of this register is 7 hexadecimal digits which is calculated based on the Quad, lane number, register type and offset.

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

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

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

Figure 3-79. Example 1



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

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

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

# End of start prbs test sequence.
```

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

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

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

- Eye Monitor

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

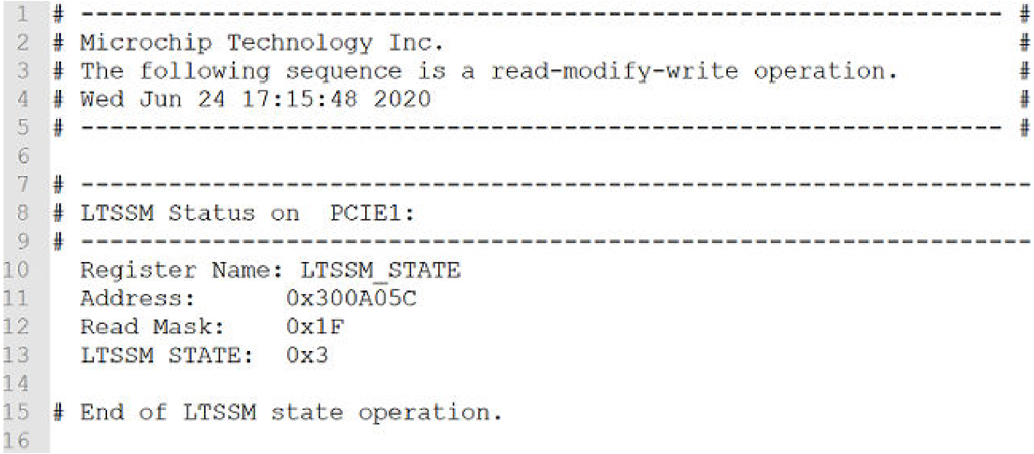
Example 2
LTSSM state on PCIe lane

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

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

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

Figure 3-80. Example 2



```
# ---------------------------------------------------------------- #
# Microchip Technology Inc.                                        #
# The following sequence is a read-modify-write operation.         #
# Wed Jun 24 17:15:48 2020                                         #
# ---------------------------------------------------------------- #

# -----------------------------------------------------------------
# LTSSM Status on  PCIE1:
# -----------------------------------------------------------------
  Register Name: LTSSM_STATE
  Address:       0x300A05C
  Read Mask:     0x1F
  LTSSM STATE:   0x3

# End of LTSSM state operation.
```

3.15 MSS Register Access

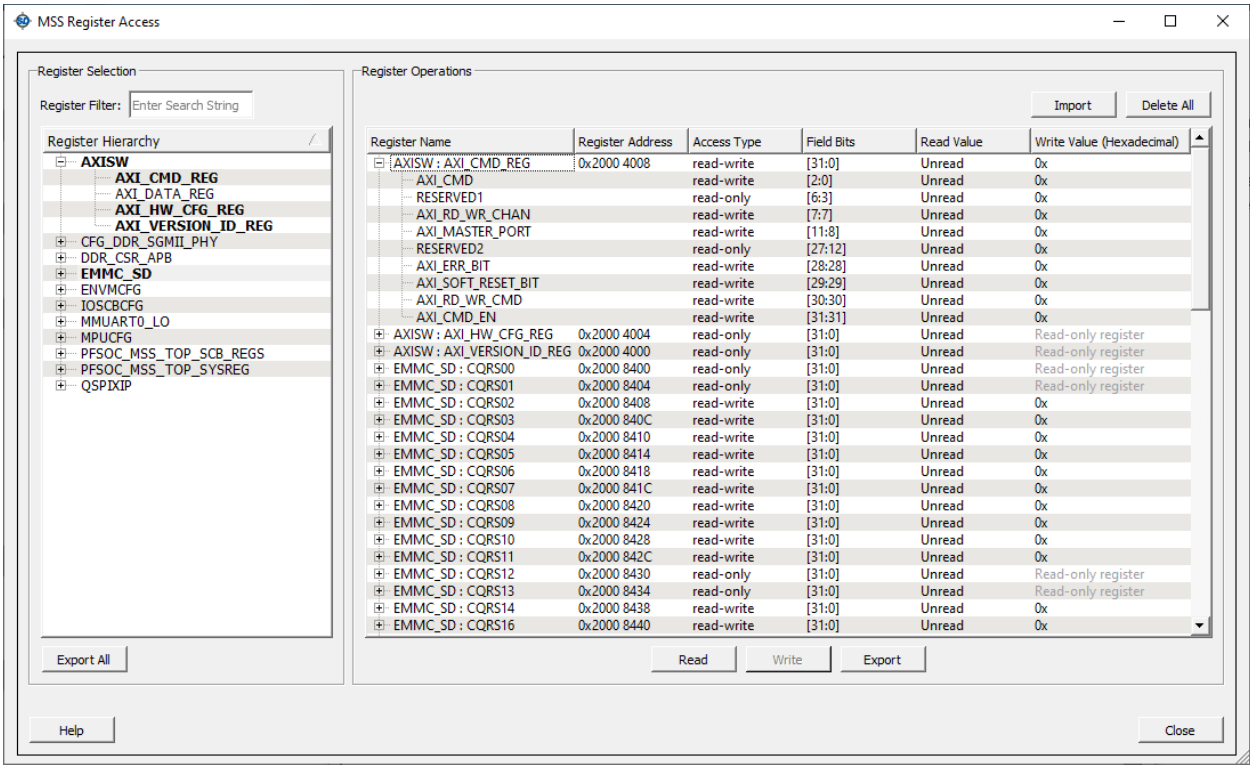
SmartDebug provides the **MSS Register Access** option to read and write to the device and export register read details to a `.csv` file. Click **MSS Register Access** on the main SmartDebug window to access this feature.

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

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

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

Figure 3-81. MSS Register Access Window



Register Filter: Enter Search String

Register Hierarchy

- **AXISW**
  - **AXI_CMD_REG**
  - AXI_DATA_REG
  - **AXI_HW_CFG_REG**
  - **AXI_VERSION_ID_REG**
- CFG_DDR_SGMII_PHY
- DDR_CSR_APB
- **EMMC_SD**
- ENVMCFG
- IOSCBCFG
- MMUART0_LO
- MPUCFG
- PFSOC_MSS_TOP_SCB_REGS
- PFSOC_MSS_TOP_SYSREG
- QSPIXIP

Register Name	Register Address	Access Type	Field Bits	Read Value	Write Value (Hexadecimal)
AXISW : AXI_CMD_REG	0x2000 4008	read-write	[31:0]	Unread	0x
AXI_CMD		read-write	[2:0]	Unread	0x
RESERVED1		read-only	[6:3]	Unread	0x
AXI_RD_WR_CHAN		read-write	[7:7]	Unread	0x
AXI_MASTER_PORT		read-write	[11:8]	Unread	0x
RESERVED2		read-only	[27:12]	Unread	0x
AXI_ERR_BIT		read-write	[28:28]	Unread	0x
AXI_SOFT_RESET_BIT		read-write	[29:29]	Unread	0x
AXI_RD_WR_CMD		read-write	[30:30]	Unread	0x
AXI_CMD_EN		read-write	[31:31]	Unread	0x
AXISW : AXI_HW_CFG_REG	0x2000 4004	read-only	[31:0]	Unread	Read-only register
AXISW : AXI_VERSION_ID_REG	0x2000 4000	read-only	[31:0]	Unread	Read-only register
EMMC_SD : CQRS00	0x2000 8400	read-only	[31:0]	Unread	Read-only register
EMMC_SD : CQRS01	0x2000 8404	read-only	[31:0]	Unread	Read-only register
EMMC_SD : CQRS02	0x2000 8408	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS03	0x2000 840C	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS04	0x2000 8410	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS05	0x2000 8414	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS06	0x2000 8418	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS07	0x2000 841C	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS08	0x2000 8420	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS09	0x2000 8424	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS10	0x2000 8428	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS11	0x2000 842C	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS12	0x2000 8430	read-only	[31:0]	Unread	Read-only register
EMMC_SD : CQRS13	0x2000 8434	read-only	[31:0]	Unread	Read-only register
EMMC_SD : CQRS14	0x2000 8438	read-write	[31:0]	Unread	0x
EMMC_SD : CQRS16	0x2000 8440	read-write	[31:0]	Unread	0x

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

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

Option	Description
Register Selection	Lists all the instances of the MSS block as described in the MSS register map. 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. • AXISW • MPUCFG • ENVMCFG • IOSCBCFG • PFSOC_MSS_TOP_SCB_REGS • PFSOC_MSS_TOP_SYSREG
Register Filter	Enables you search for instances/register by their names. Supports wildcard (*) searches. • The search is real time. • Enter asterisk (*) to display all registers having instance names expanded. • Names entered without asterisk look for exact match.
Register Hierarchy	Displays Instance name and Register name in a two-level hierarchy-level format. • 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.

...........continued

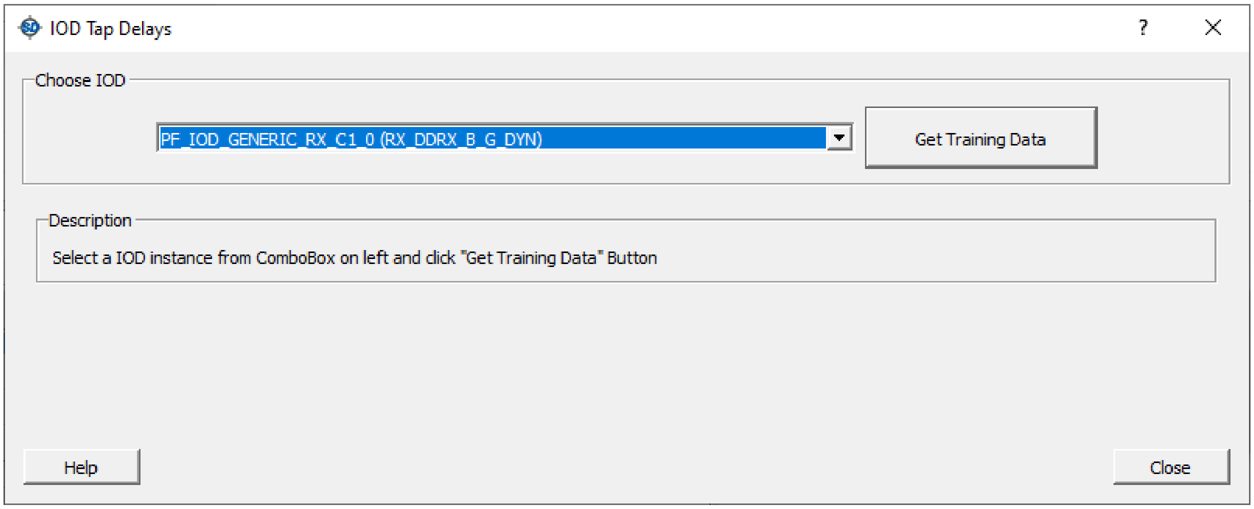
Option	Description
Export All	Exports information of all registers that are displayed in the selection pane to the specified `.csv` file. • The exported `.csv` file contains six sections: – Register Name – Register Address (hexadecimal number) – Register Field Name – Register Field Range – Register Field 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. **Note:** Selected registers are retained even if you close and re-open the SmartDebug project in the stand-alone SmartDebug tool. • 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. **Note:** You cannot remove a field inside the register. • Write value column is populated with `0x` to indicate that it supports hexadecimal values. – Click on a specific register row under write column to edit the value. – The text entered is validated based on the range (field bits) of a field or a register. **Note:** Non-hexadecimal character will not be inserted.
Import	Import the register list from a *.csv file generated by the **Export** or the **Export All** register export operation.
Delete All	Deletes all the selected registers displayed listed under the **Register Operations** pane.
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. • 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 • Register Name • Register Address (hexadecimal number) • Register Field Name • Register Field Range • Register Field Value (hexadecimal number) • Register Value (hexadecimal number)

3.16 Debug IOD

SmartDebug provides the IOD Tap Delays feature for designs where the `PF_IOD_GENERIC_RX` IP instance is configured as Dynamic or Fractional Dynamic mode in the design. Click the **Debug IOD** button on the main SmartDebug window to access this feature.
Note: The **Debug IOD** button is not available for designs where the IP in the modes mentioned is not used. The feature is supported by designs for PolarFire and PolarFire SoC family devices.

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

Figure 3-82. IOD Tap Delays Window

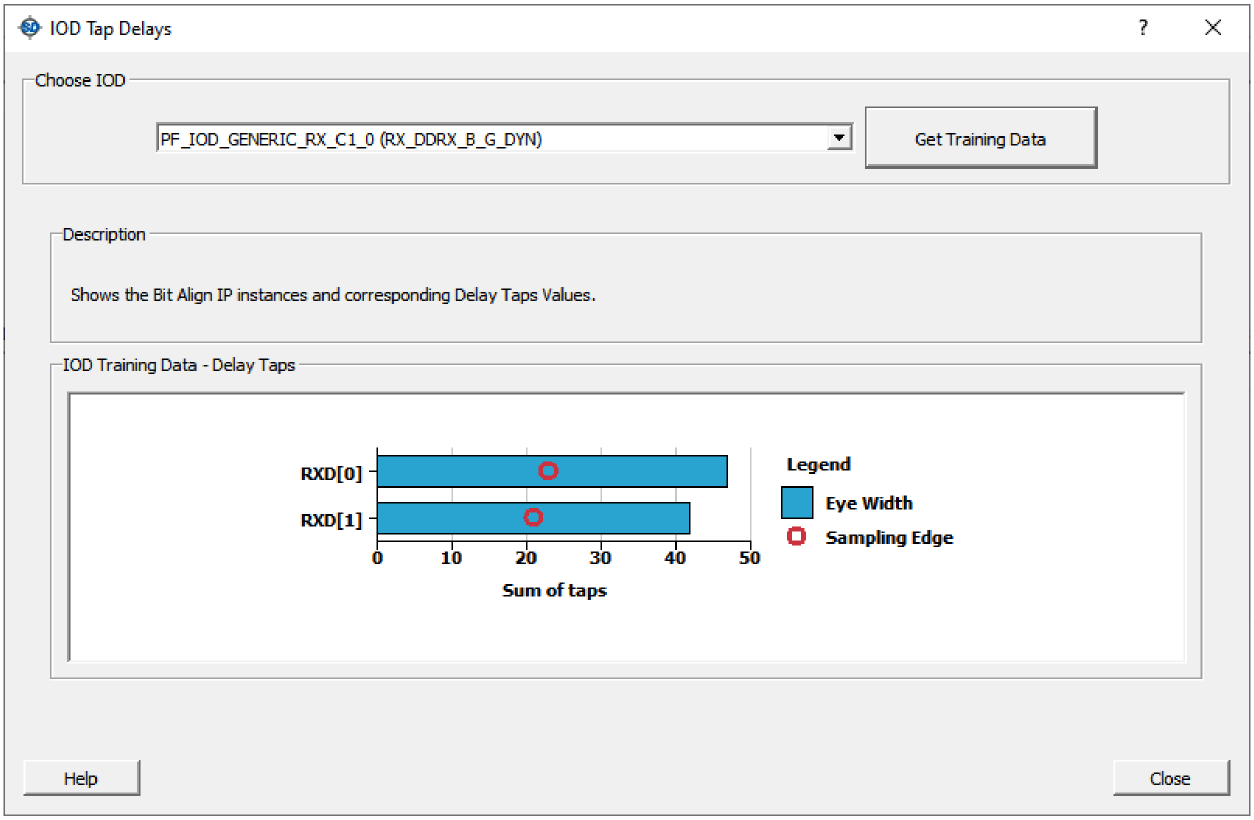


Choose IOD: PF_IOD_GENERIC_RX_C1_0 (RX_DDRX_B_G_DYN)

Description: Select a IOD instance from ComboBox on left and click "Get Training Data" Button

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

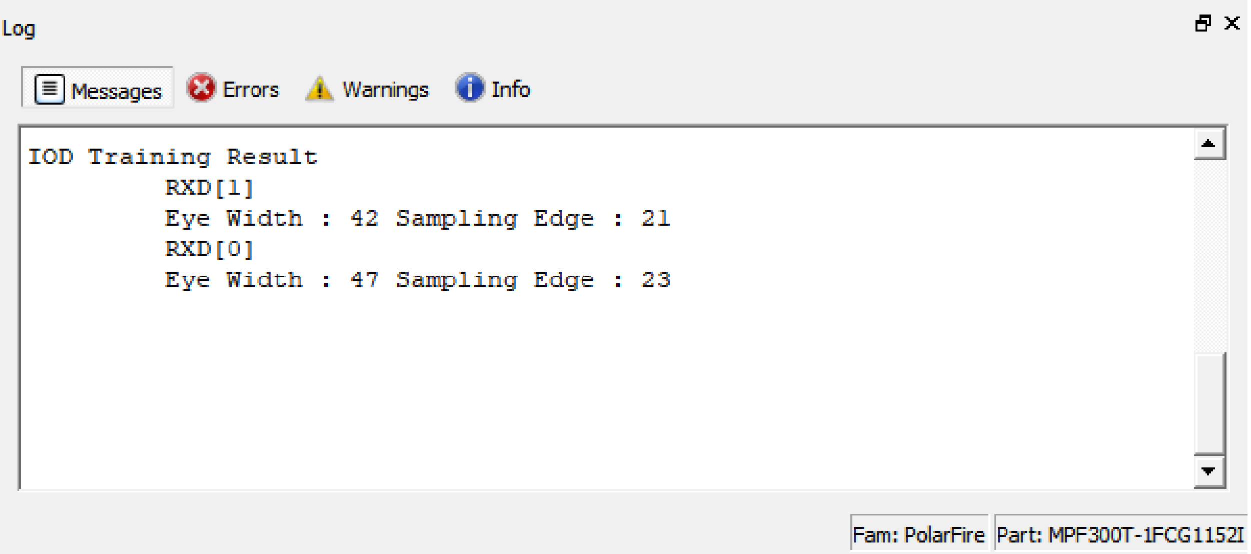
Figure 3-83. IOD Tap Delays - Eye Width and Sample Edge Data Representation



Shows the Bit Align IP instances and corresponding Delay Taps Values.

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

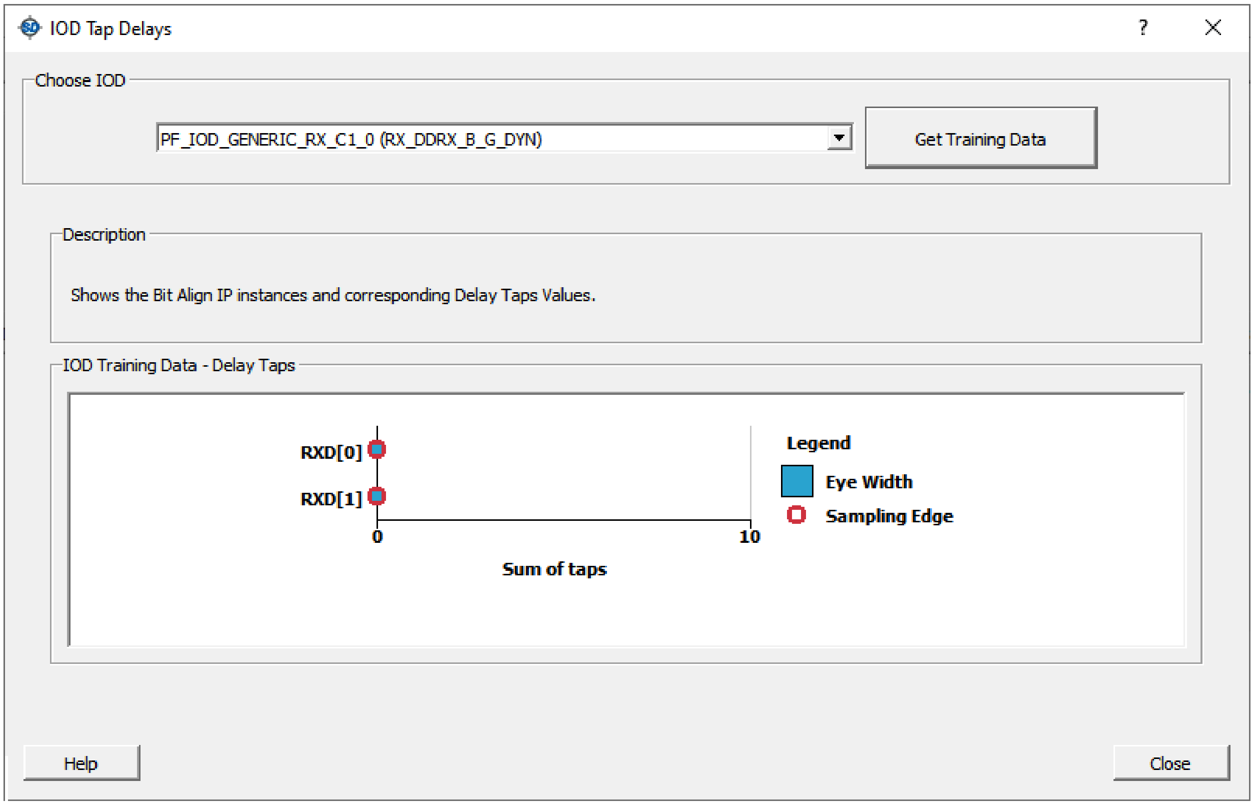
Figure 3-84. SmartDebug Log Window - Training Successful



```
IOD Training Result
         RXD[1]
         Eye Width : 42 Sampling Edge : 21
         RXD[0]
         Eye Width : 47 Sampling Edge : 23
```

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

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



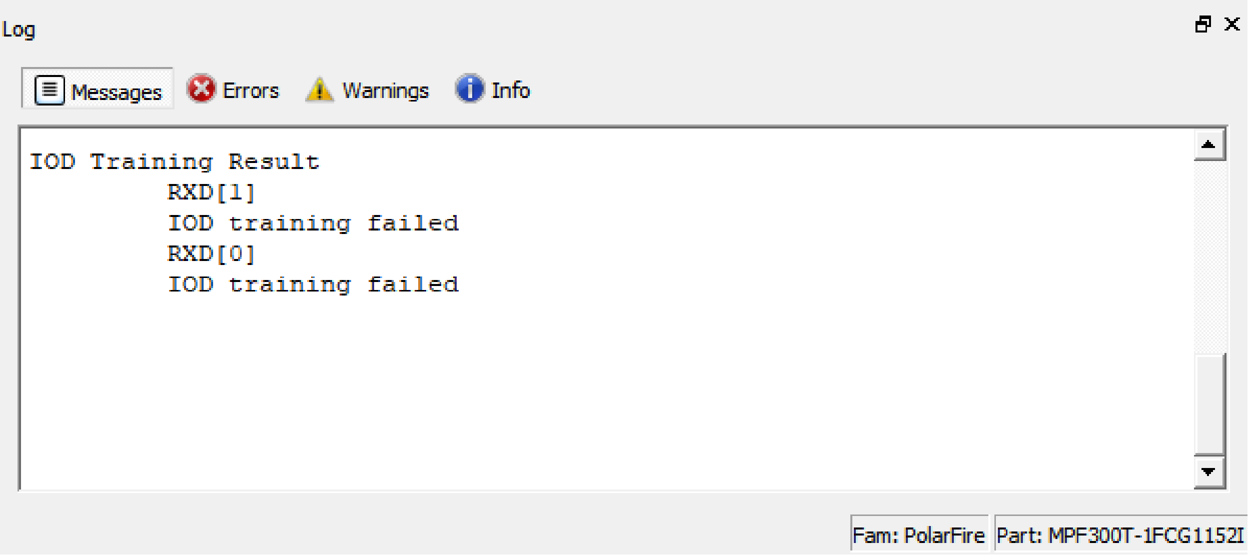
PF_IOD_GENERIC_RX_C1_0 (RX_DDRX_B_G_DYN)

Description

Shows the Bit Align IP instances and corresponding Delay Taps Values.

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

Figure 3-86. SmartDebug Log Window - Training Failed



```
IOD Training Result
         RXD[1]
         IOD training failed
         RXD[0]
         IOD training failed
```

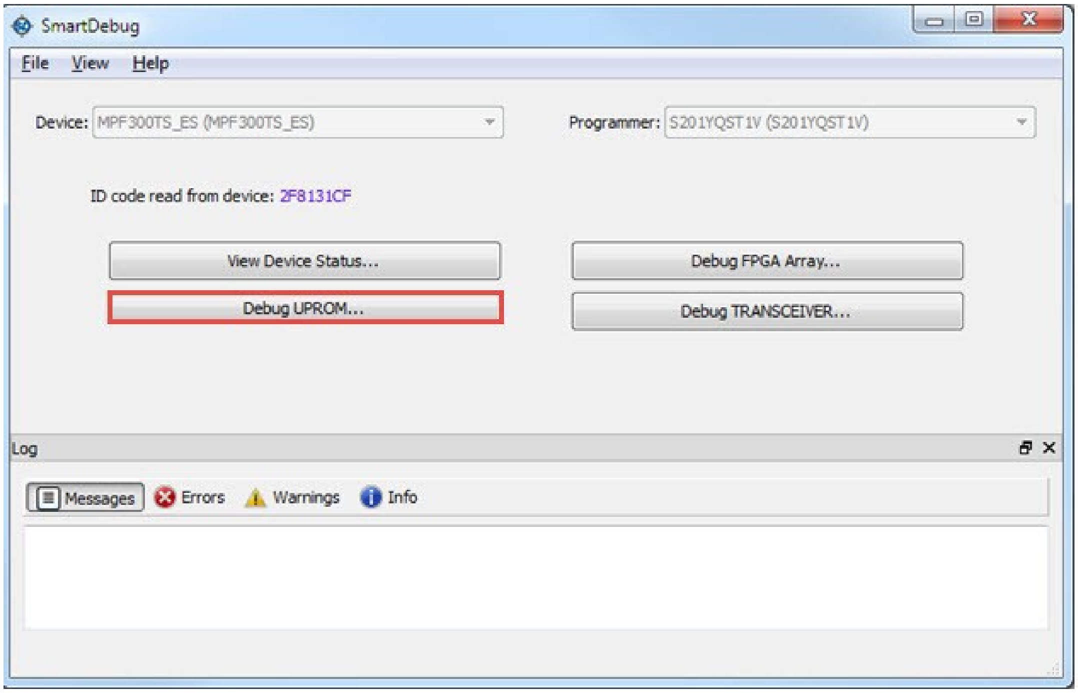
Fam: PolarFire Part: MPF300T-1FCG1152I

3.17 Debug UPROM

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

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

Figure 3-87. SmartDebug Window - Debug UPROM



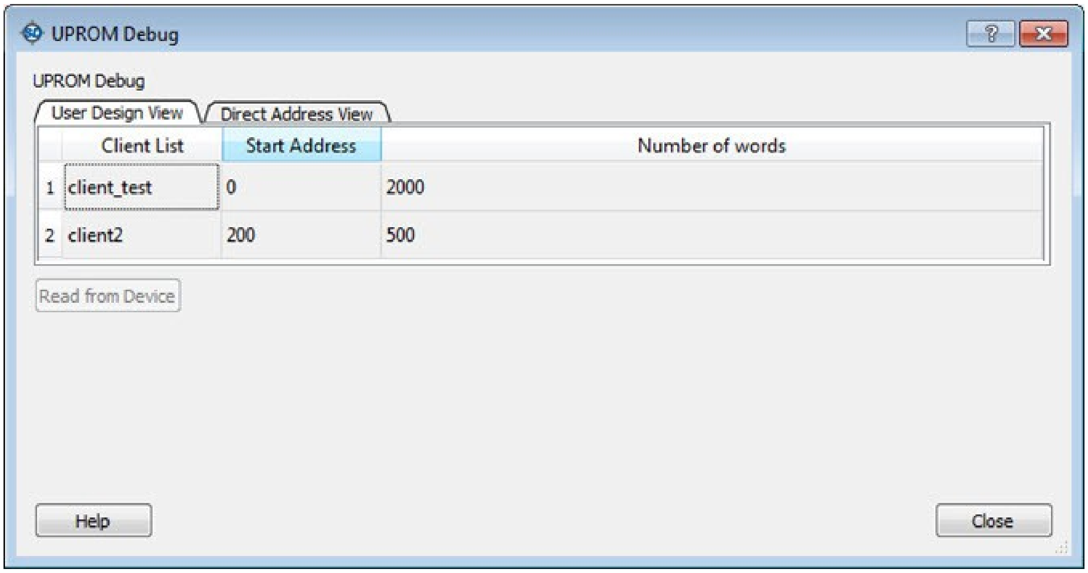
Device: MPF300TS_ES (MPF300TS_ES)

Programmer: S201YQST1V (S201YQST1V)

ID code read from device: 2F8131CF

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

Figure 3-88. UPROM Debug Window

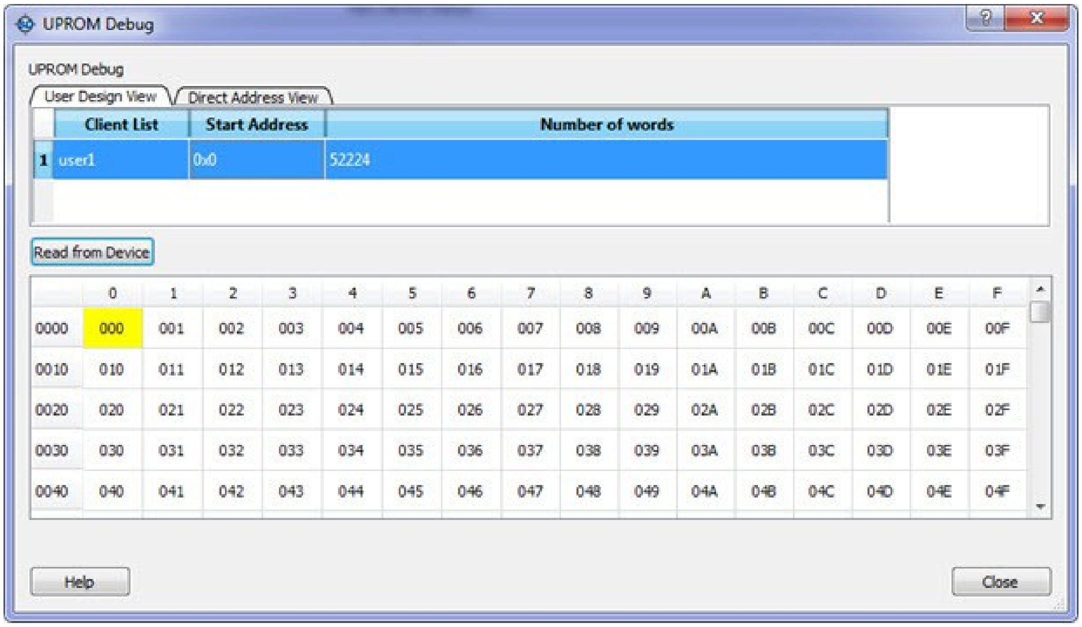


	Client List	Start Address	Number of words
1	client_test	0	2000
2	client2	200	500

3.17.1 User Design View

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

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



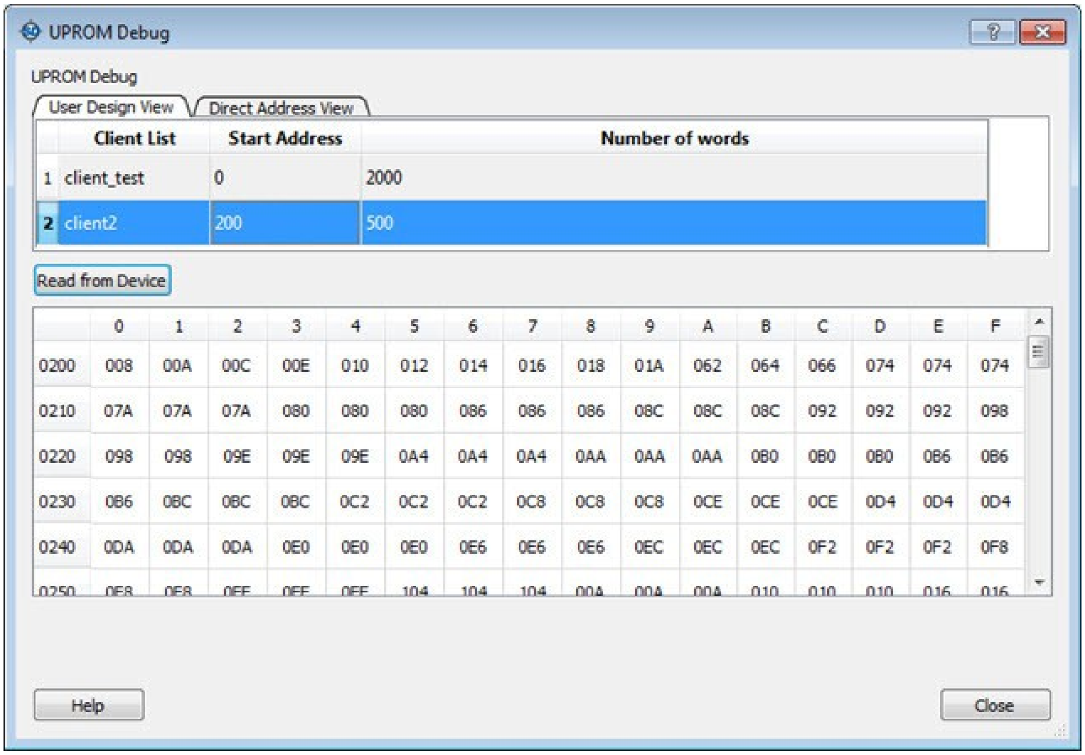
	Client List	Start Address	Number of words
1	user1	0x0	52224

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0010	010	011	012	013	014	015	016	017	018	019	01A	01B	01C	01D	01E	01F
0020	020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F
0030	030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E	03F
0040	040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F

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

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

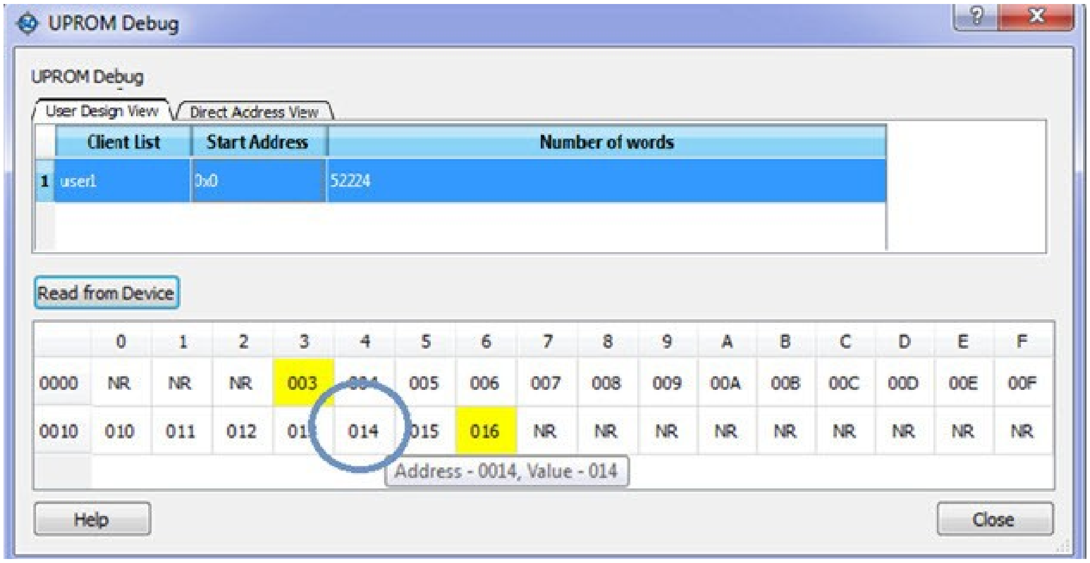
Row addresses begin with a word address associated with *Start Address*. For example, if the *Start Address* is 0x15 (hex), the starting row has an address of 0x0010.



	Client List	Start Address	Number of words
1	client_test	0	2000
2	client2	200	500

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0200	008	00A	00C	00E	010	012	014	016	018	01A	062	064	066	074	074	074
0210	07A	07A	07A	080	080	080	086	086	086	08C	08C	08C	092	092	092	098
0220	098	098	09E	09E	09E	0A4	0A4	0A4	0AA	0AA	0AA	0B0	0B0	0B0	0B6	0B6
0230	0B6	0BC	0BC	0BC	0C2	0C2	0C2	0C8	0C8	0C8	0CE	0CE	0CE	0D4	0D4	0D4
0240	0DA	0DA	0DA	0E0	0E0	0E0	0E6	0E6	0E6	0EC	0EC	0EC	0F2	0F2	0F2	0F8
0250	0E8	0E8	0EE	0EE	0EE	104	104	104	00A	00A	00A	010	010	010	016	016

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



	Client List	Start Address	Number of words
1	user1	0x0	52224

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	NR	NR	NR	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0010	010	011	012	013	014	015	016	NR	NR	NR	NR	NR	NR	NR	NR	NR

Address - 0014, Value - 014

3.17.2 Direct Address View

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

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

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

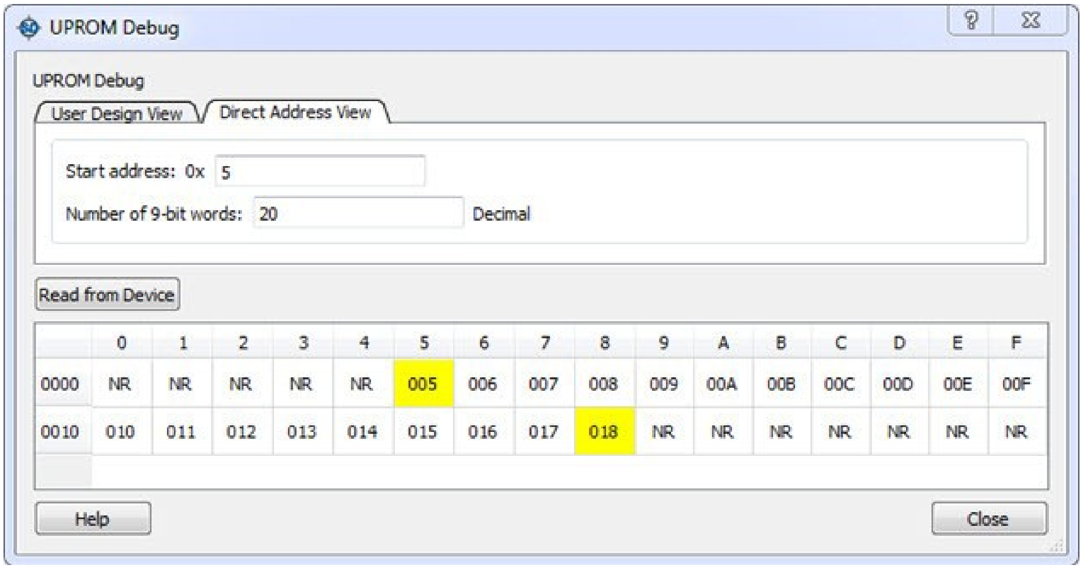
Number of 9-bit words: Positive integer value

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

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

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

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



Start address: 0x 5

Number of 9-bit words: 20 Decimal

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

Notes

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

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

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

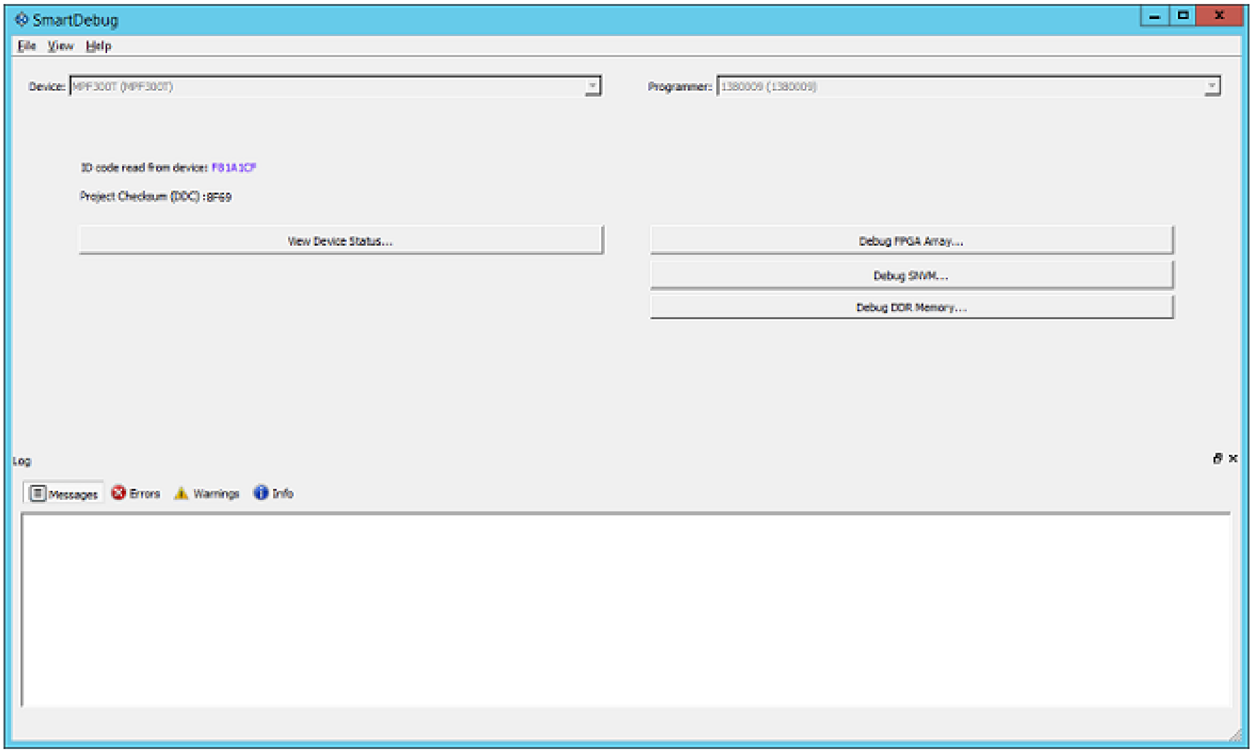
3.17.3 Demo Mode

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

3.18 Debug DDR IO Margin

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

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



Device: MPF300T (MPF300T)

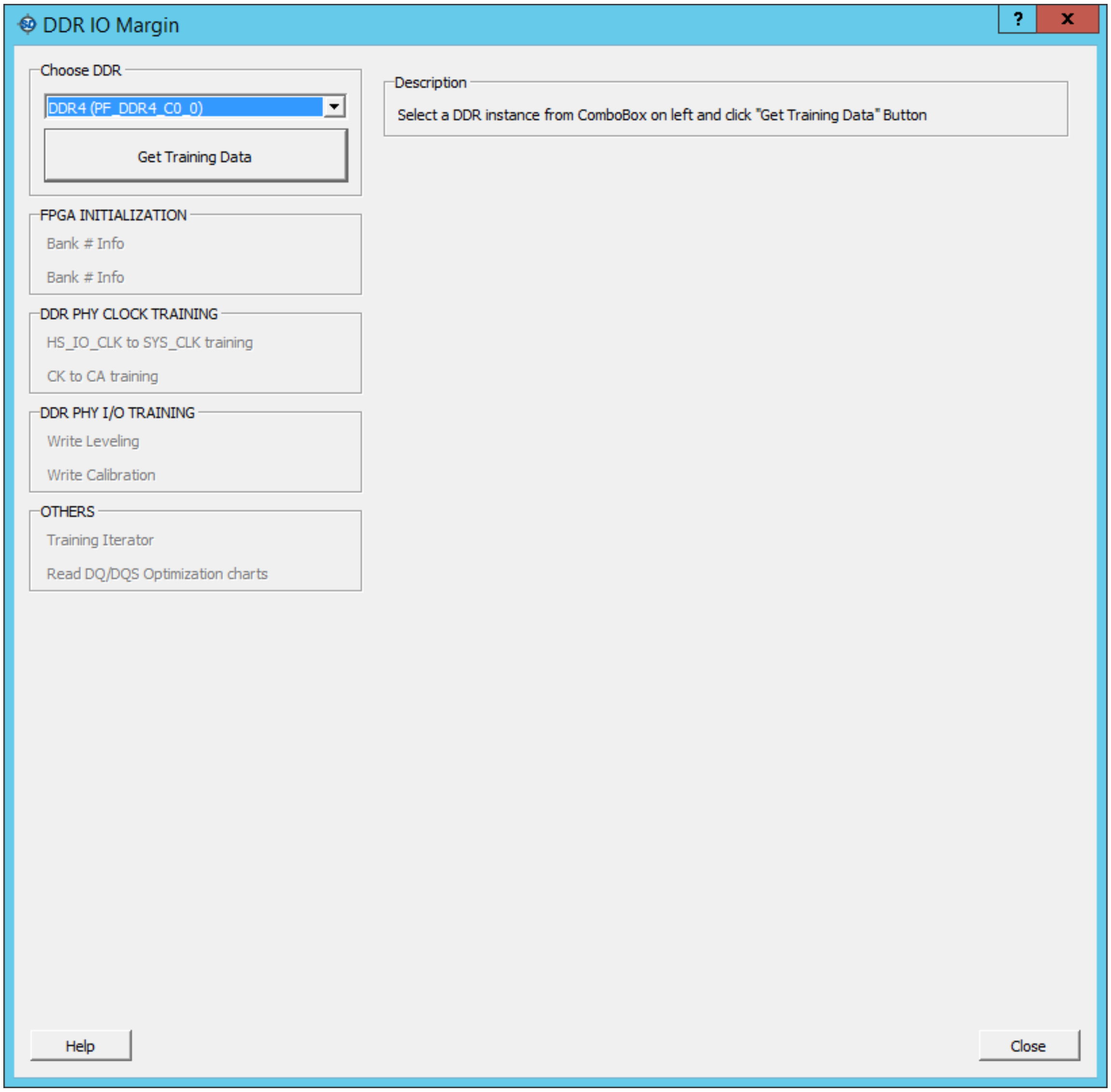
Programmer: 1380009 (1380009)

ID code read from device: F81A1CF

Project Checksum (DDC) :8F69

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

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

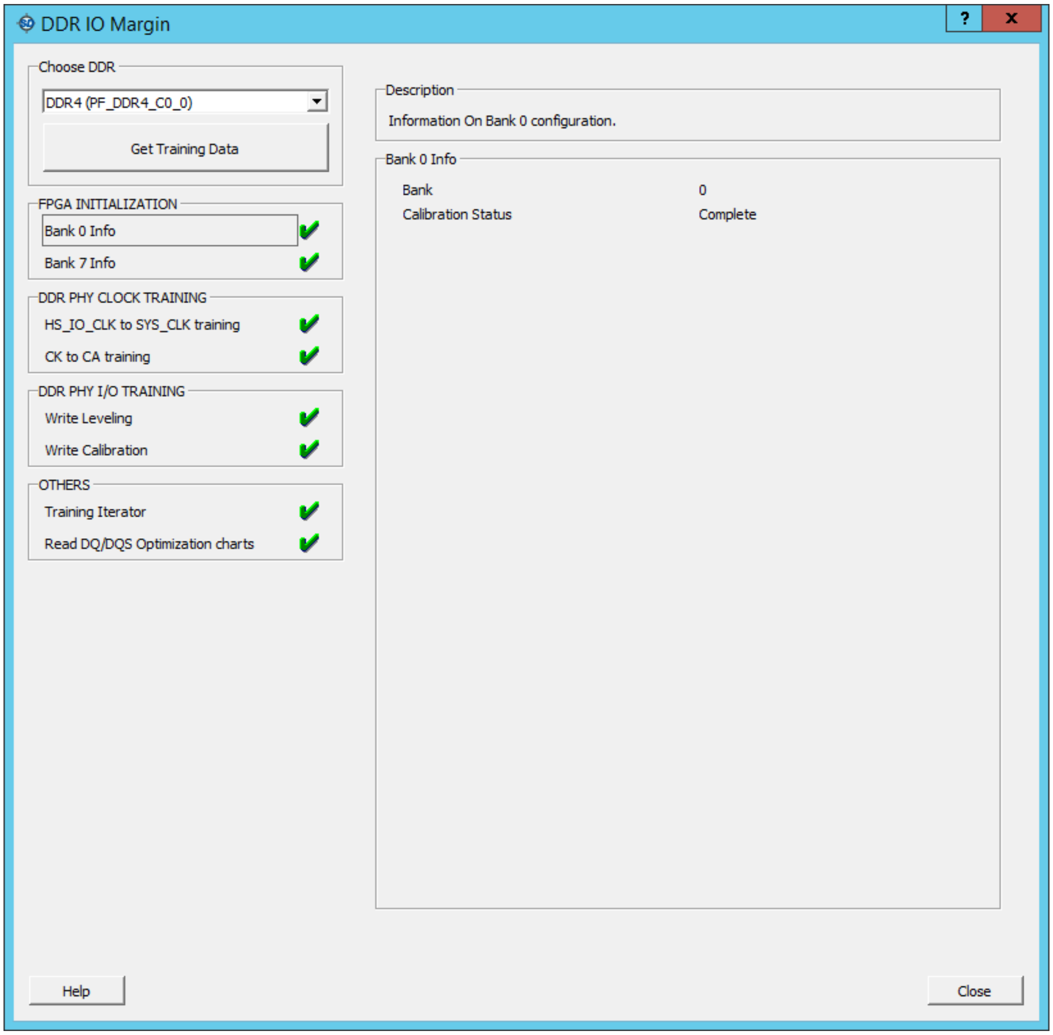


Choose DDR: DDR4 (PF_DDR4_C0_0)

Description: Select a DDR instance from ComboBox on left and click "Get Training Data" Button

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

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



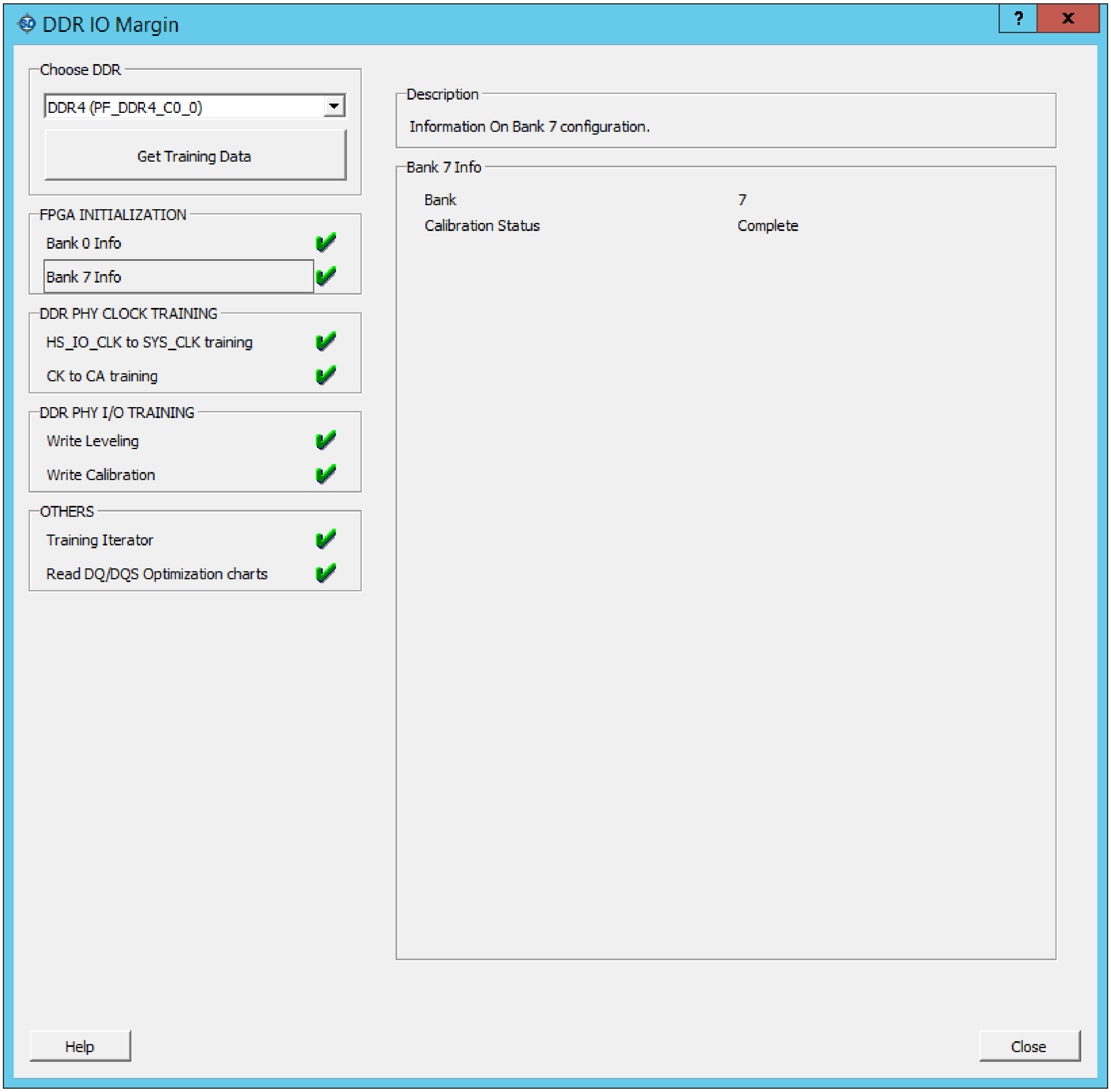
Choose DDR: DDR4 (PF_DDR4_C0_0)

Description: Information On Bank 0 configuration.

Bank 0 Info

Bank	0
Calibration Status	Complete

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



DDR4 (PF_DDR4_C0_0)

Bank 0 Info ✓
Bank 7 Info ✓

HS_IO_CLK to SYS_CLK training ✓
CK to CA training ✓

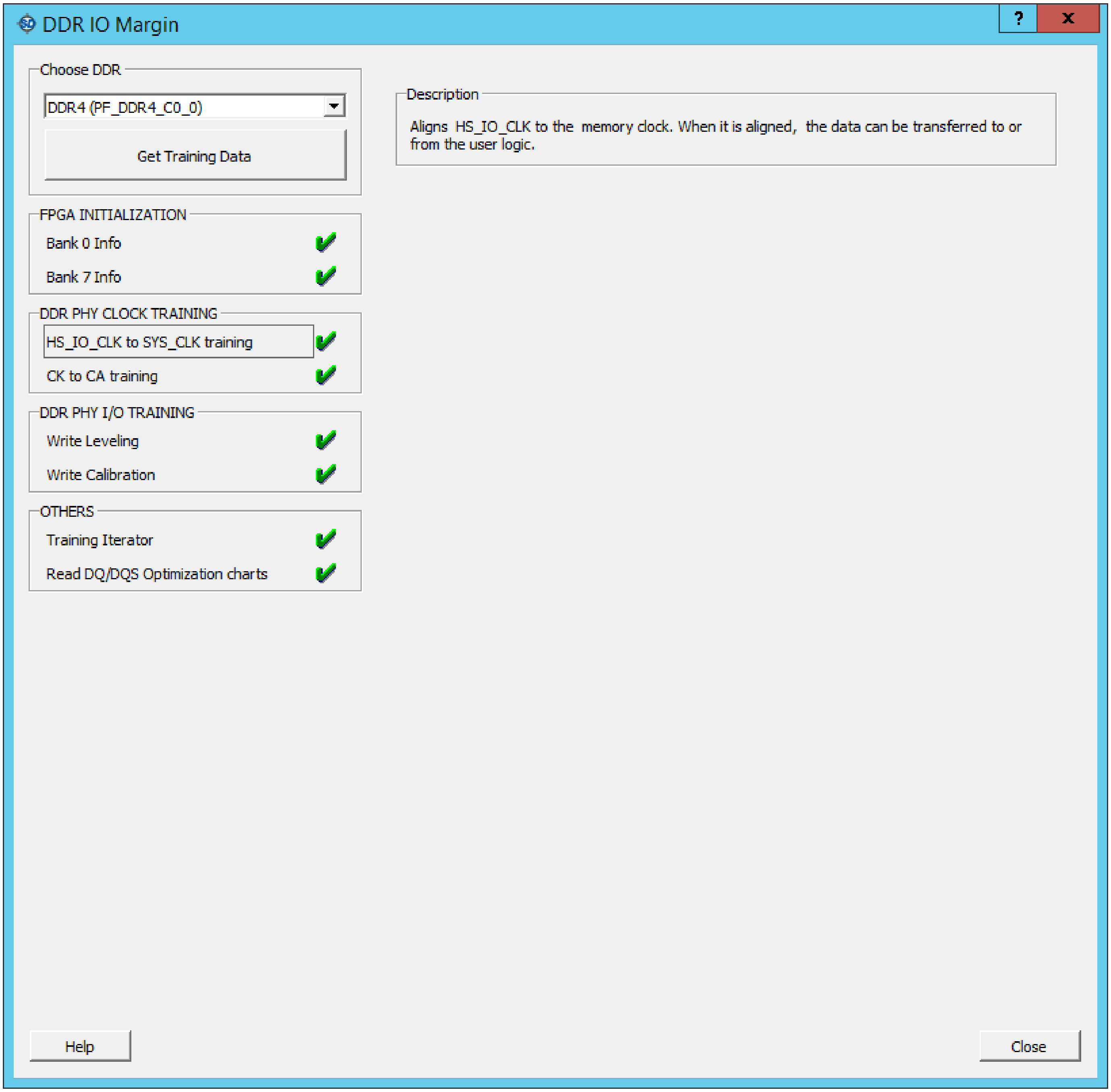
Write Leveling ✓
Write Calibration ✓

Training Iterator ✓
Read DQ/DQS Optimization charts ✓

Information On Bank 7 configuration.

Bank	7
Calibration Status	Complete

Figure 3-93. DDR IO Margin Dialog Box Showing HK_IO_CLK to SYS_CLK Training



DDR IO Margin

Choose DDR: DDR4 (PF_DDR4_C0_0)

Description

Aligns HS_IO_CLK to the memory clock. When it is aligned, the data can be transferred to or from the user logic.

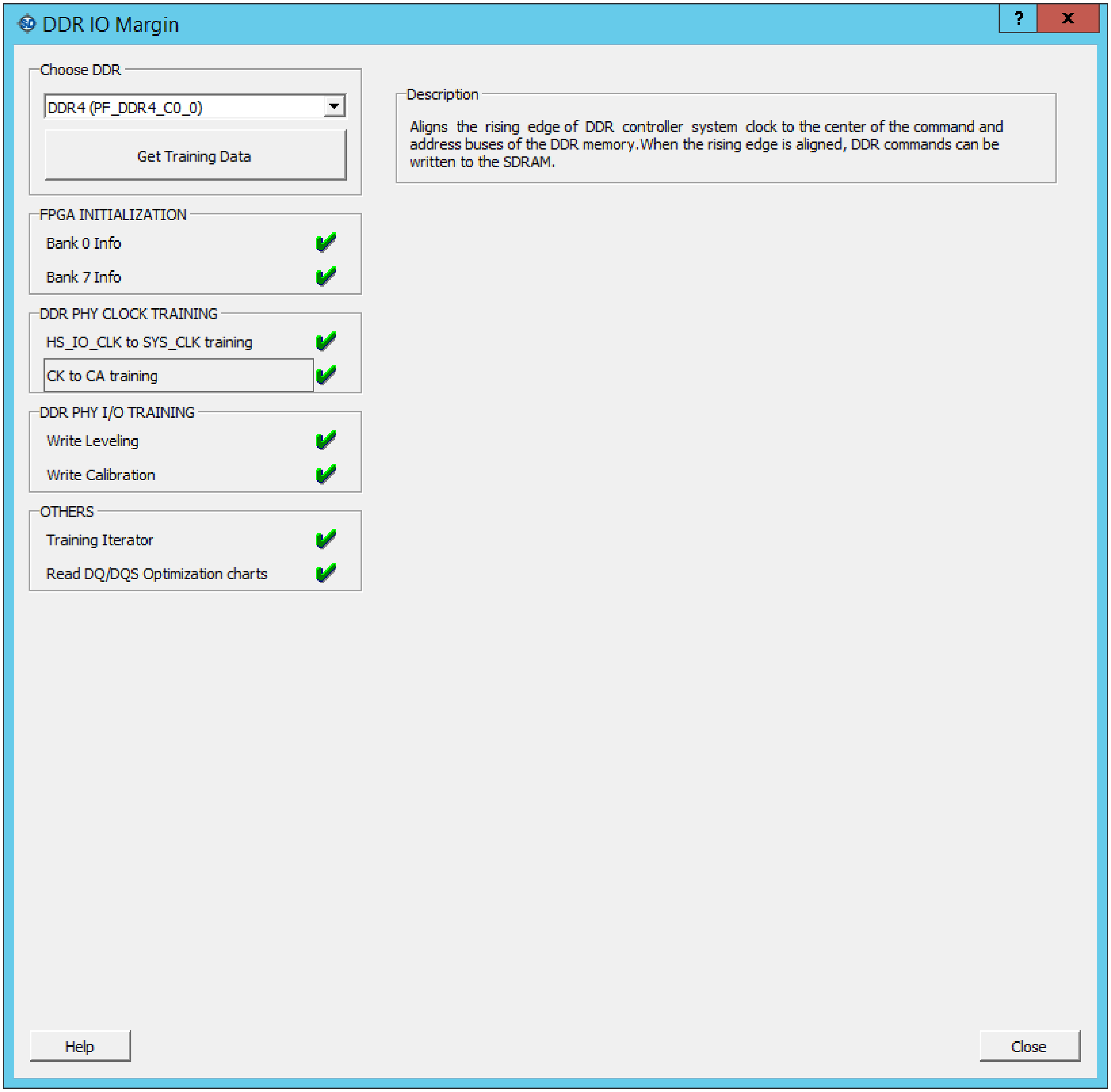
FPGA INITIALIZATION
- Bank 0 Info ✓
- Bank 7 Info ✓

DDR PHY CLOCK TRAINING
- HS_IO_CLK to SYS_CLK training ✓
- CK to CA training ✓

DDR PHY I/O TRAINING
- Write Leveling ✓
- Write Calibration ✓

OTHERS
- Training Iterator ✓
- Read DQ/DQS Optimization charts ✓

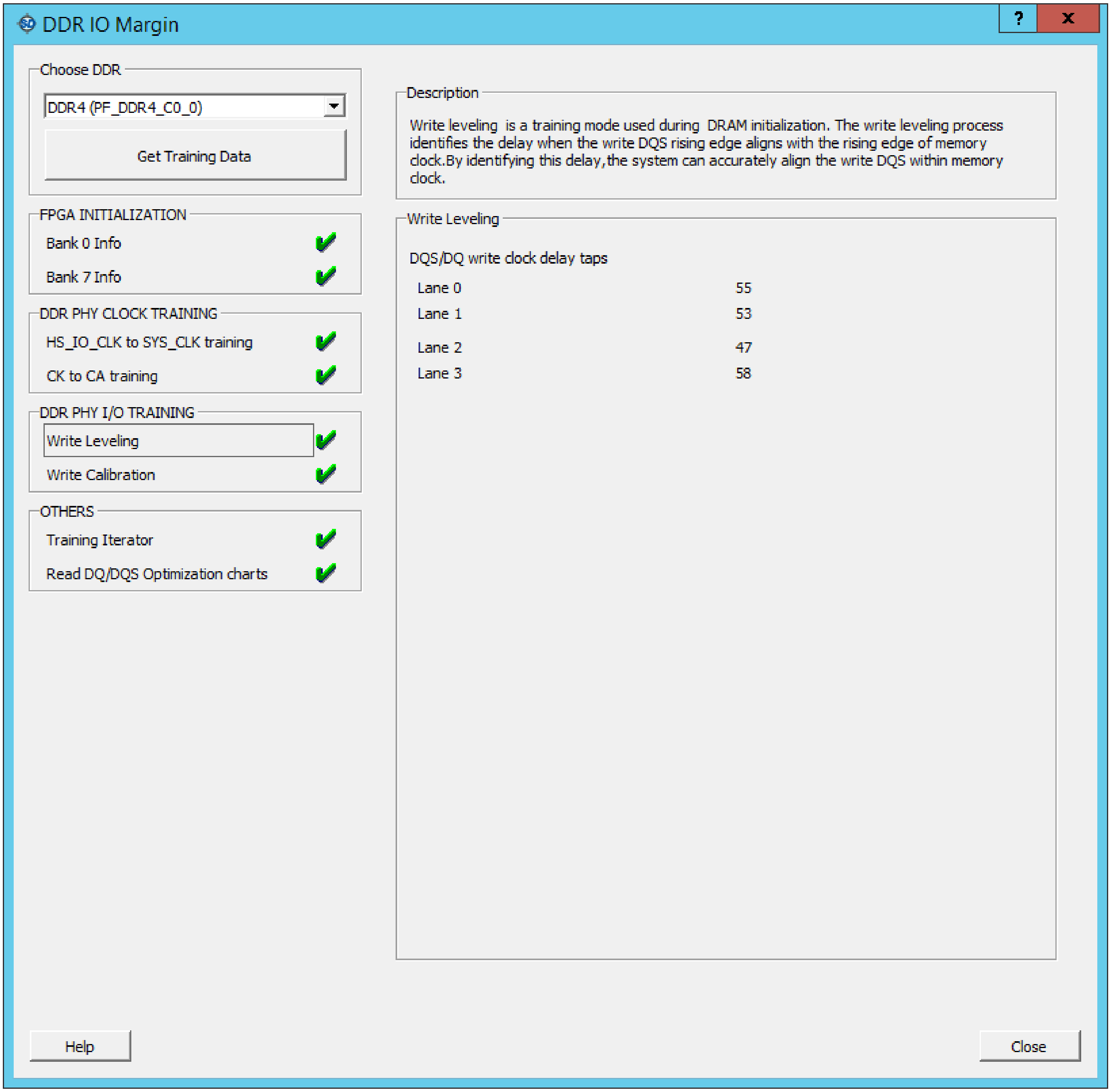
Figure 3-94. DDR IO Margin Dialog Box Showing CK to CA Training



Choose DDR: DDR4 (PF_DDR4_C0_0)

Description: Aligns the rising edge of DDR controller system clock to the center of the command and address buses of the DDR memory.When the rising edge is aligned, DDR commands can be written to the SDRAM.

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

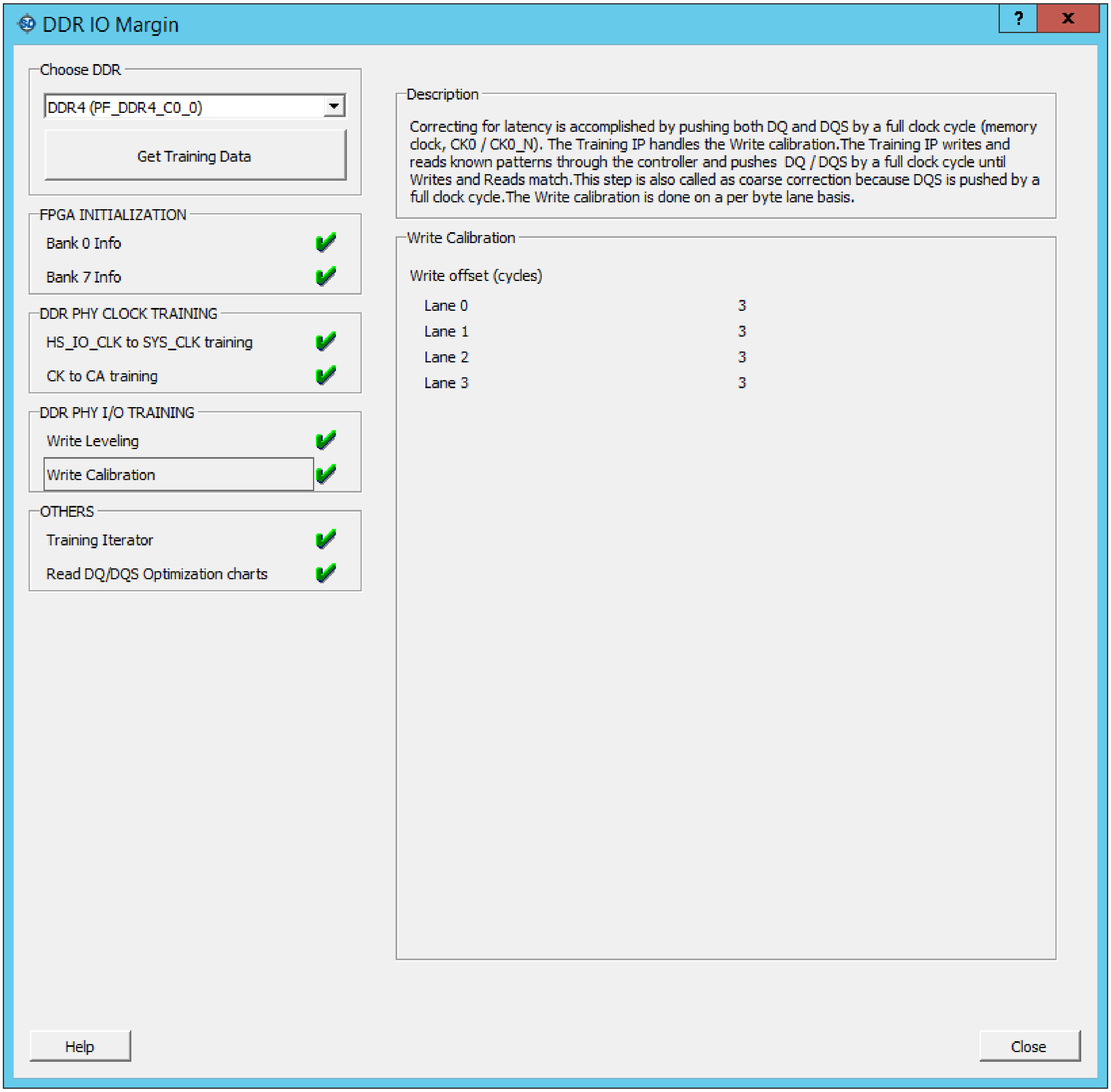


Write leveling is a training mode used during DRAM initialization. The write leveling process identifies the delay when the write DQS rising edge aligns with the rising edge of memory clock.By identifying this delay,the system can accurately align the write DQS within memory clock.

DQS/DQ write clock delay taps

Lane 0	55
Lane 1	53
Lane 2	47
Lane 3	58

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



Description

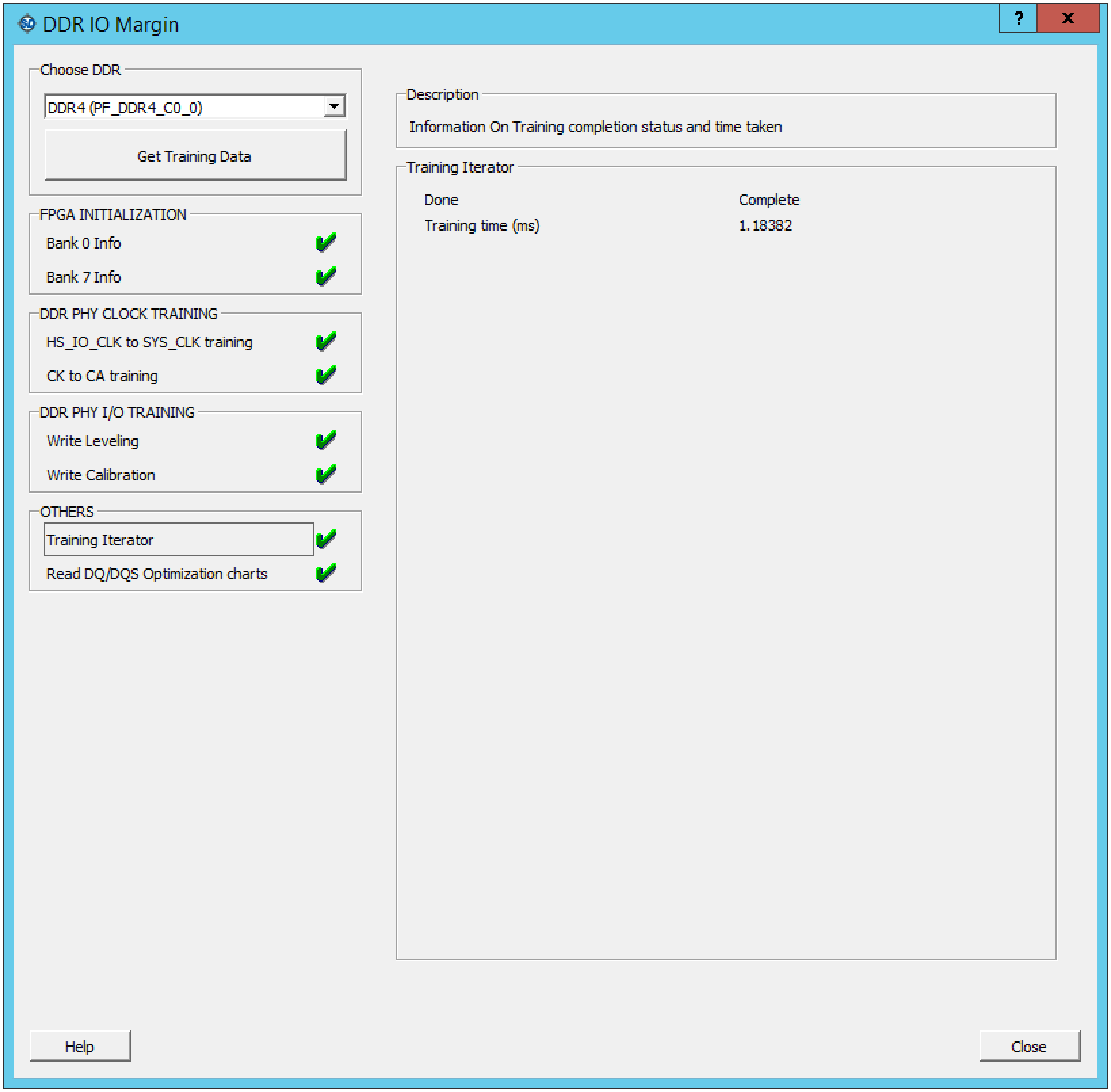
Correcting for latency is accomplished by pushing both DQ and DQS by a full clock cycle (memory clock, CK0 / CK0_N). The Training IP handles the Write calibration.The Training IP writes and reads known patterns through the controller and pushes DQ / DQS by a full clock cycle until Writes and Reads match.This step is also called as coarse correction because DQS is pushed by a full clock cycle.The Write calibration is done on a per byte lane basis.

Write Calibration

Write offset (cycles)

Lane 0	3
Lane 1	3
Lane 2	3
Lane 3	3

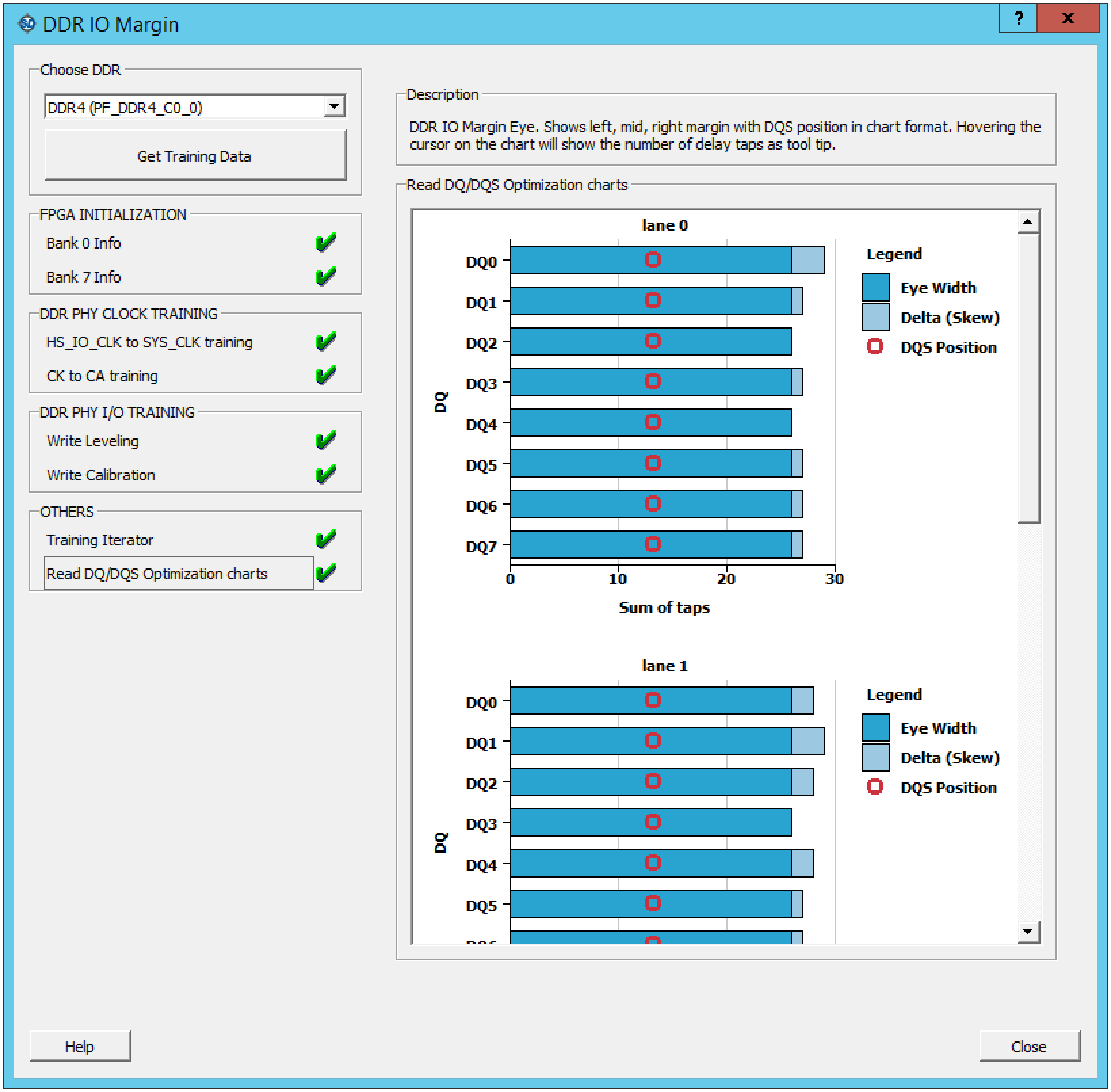
Figure 3-97. DDR IO Margin Dialog Box Showing Training Iterator Data



Information On Training completion status and time taken

Done	Complete
Training time (ms)	1.18382

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



DDR IO Margin

Choose DDR

DDR4 (PF_DDR4_C0_0)

FPGA INITIALIZATION

Bank 0 Info ✓

Bank 7 Info ✓

DDR PHY CLOCK TRAINING

HS_IO_CLK to SYS_CLK training ✓

CK to CA training ✓

DDR PHY I/O TRAINING

Write Leveling ✓

Write Calibration ✓

OTHERS

Training Iterator ✓

Read DQ/DQS Optimization charts ✓

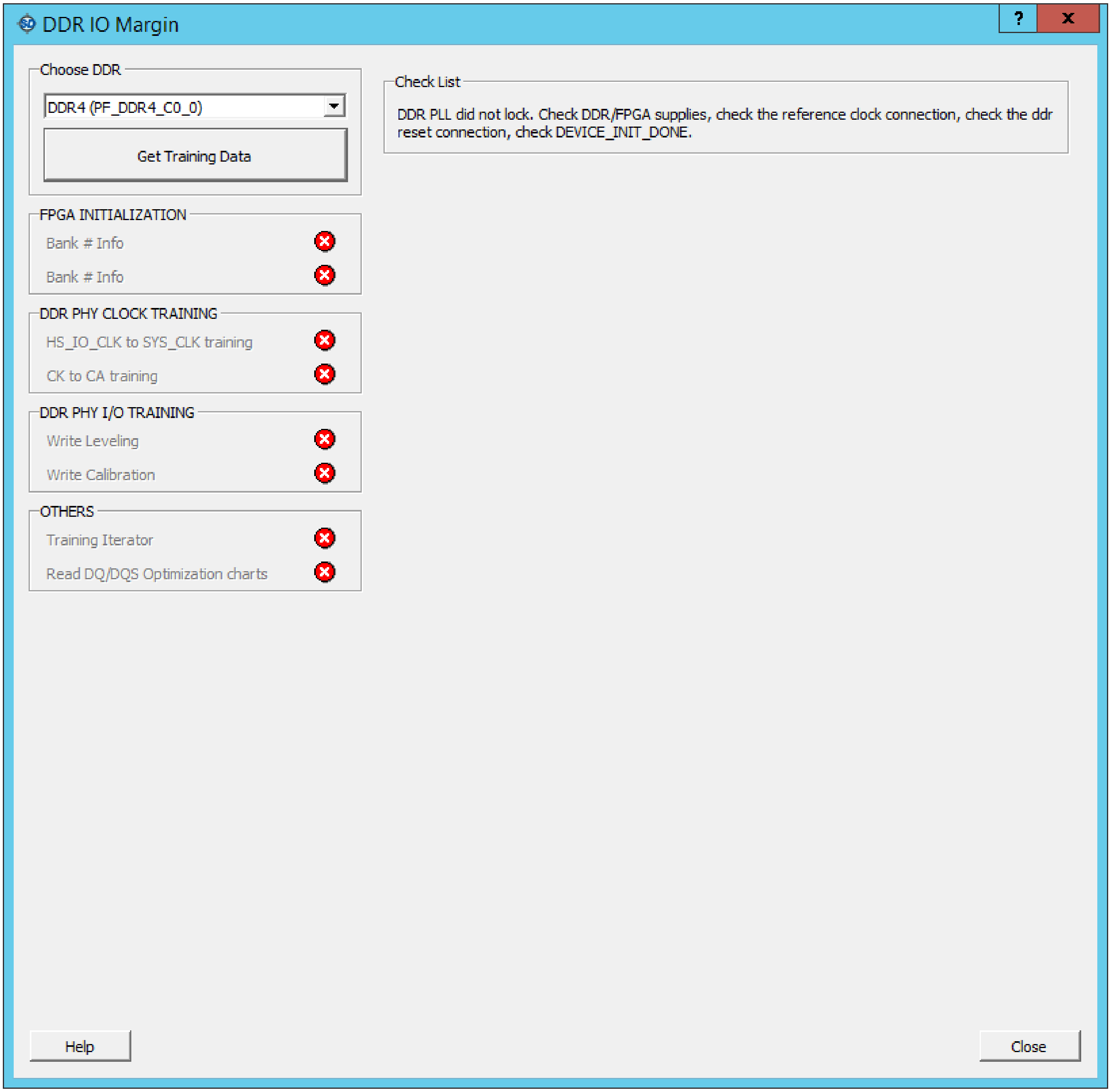
Description

DDR IO Margin Eye. Shows left, mid, right margin with DQS position in chart format. Hovering the cursor on the chart will show the number of delay taps as tool tip.

Read DQ/DQS Optimization charts

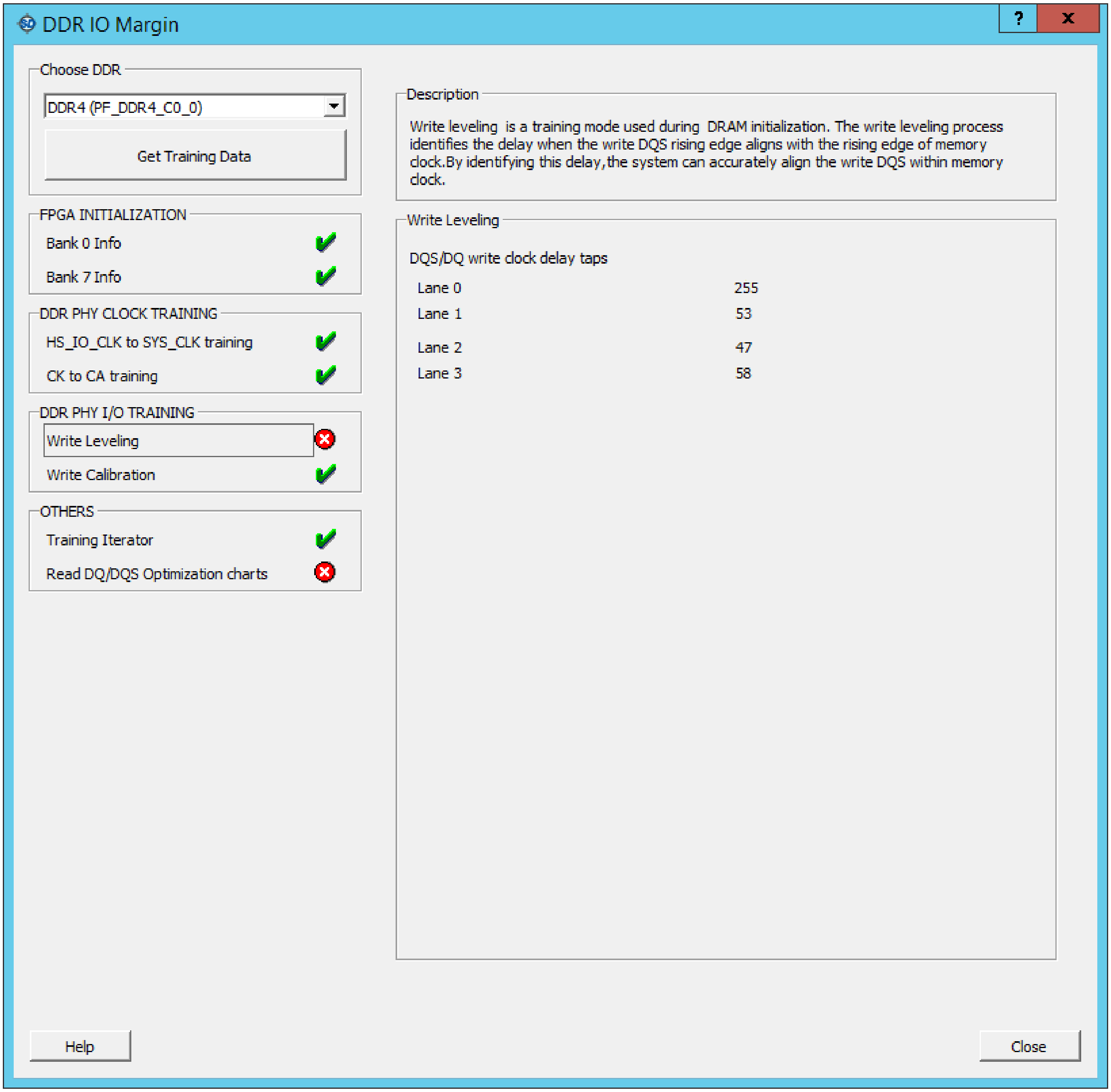
The tool informs the users about errors generated while getting training data as shown in the following figure.

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



DDR PLL did not lock. Check DDR/FPGA supplies, check the reference clock connection, check the ddr reset connection, check DEVICE_INIT_DONE.

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



Description

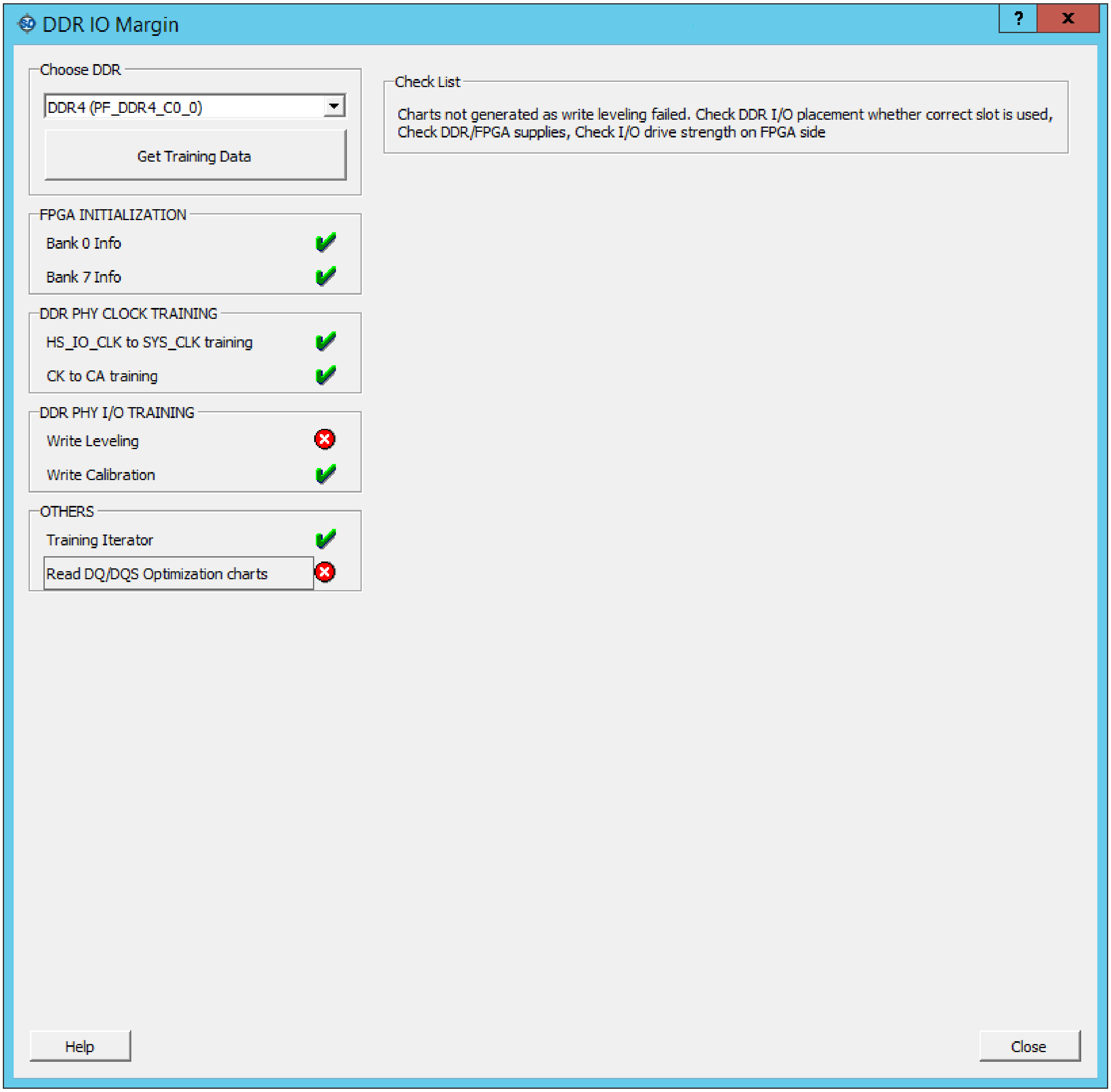
Write leveling is a training mode used during DRAM initialization. The write leveling process identifies the delay when the write DQS rising edge aligns with the rising edge of memory clock.By identifying this delay,the system can accurately align the write DQS within memory clock.

Write Leveling

DQS/DQ write clock delay taps

Lane 0	255
Lane 1	53
Lane 2	47
Lane 3	58

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



DDR IO Margin

Choose DDR

DDR4 (PF_DDR4_C0_0)

Check List

Charts not generated as write leveling failed. Check DDR I/O placement whether correct slot is used, Check DDR/FPGA supplies, Check I/O drive strength on FPGA side

FPGA INITIALIZATION

Bank 0 Info ✓

Bank 7 Info ✓

DDR PHY CLOCK TRAINING

HS_IO_CLK to SYS_CLK training ✓

CK to CA training ✓

DDR PHY I/O TRAINING

Write Leveling ☒

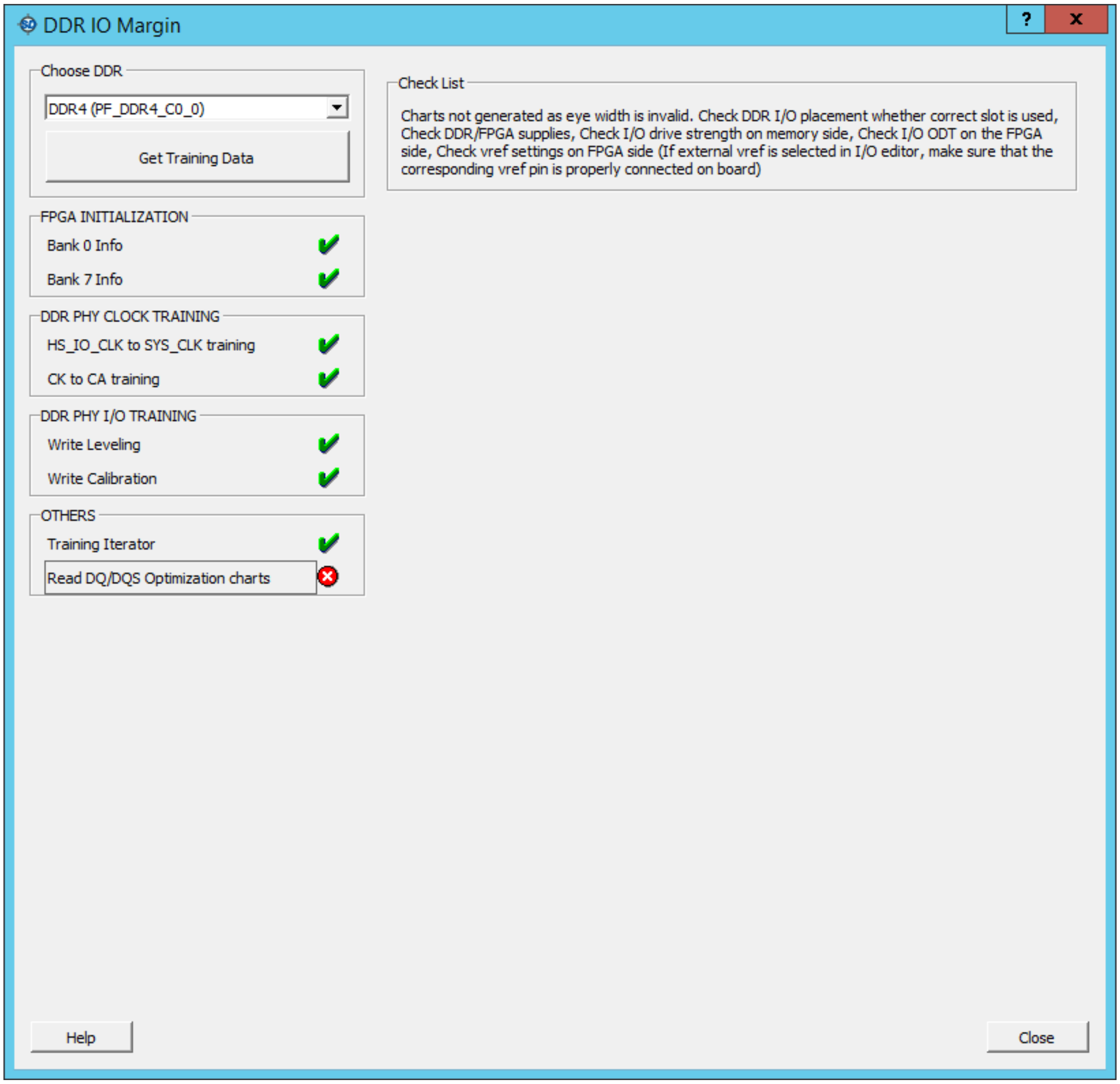
Write Calibration ✓

OTHERS

Training Iterator ✓

Read DQ/DQS Optimization charts ☒

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



Charts not generated as eye width is invalid. Check DDR I/O placement whether correct slot is used, Check DDR/FPGA supplies, Check I/O drive strength on memory side, Check I/O ODT on the FPGA side, Check vref settings on FPGA side (If external vref is selected in I/O editor, make sure that the corresponding vref pin is properly connected on board)

3.19 Debug eNVM

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

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

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

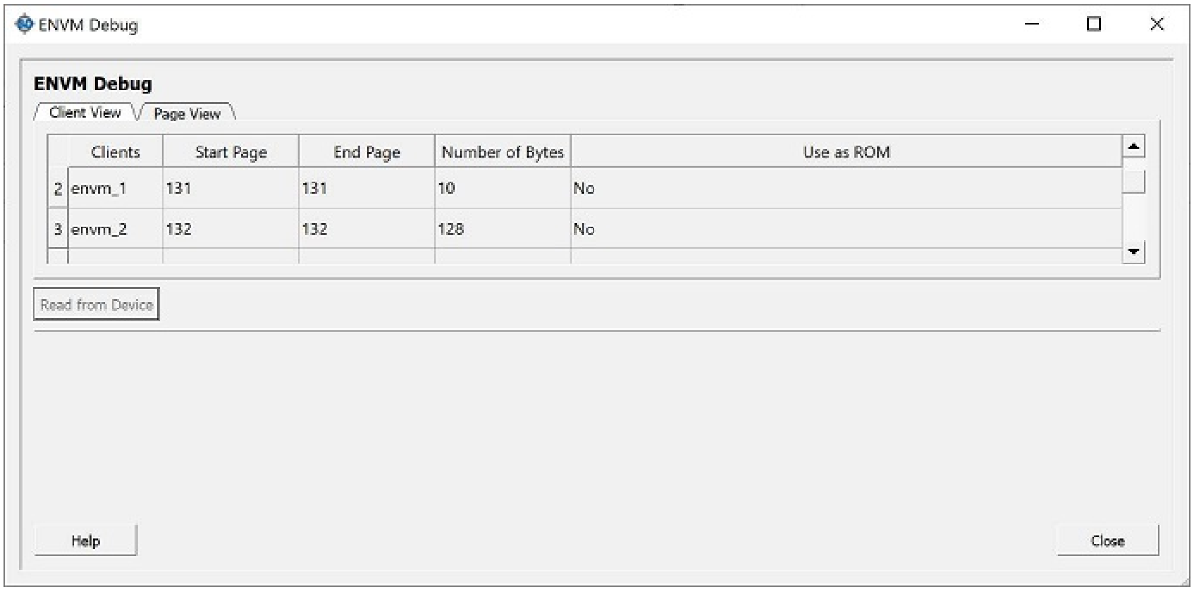
Size		Offset	Description	MSS Address
128 Kbytes	8K	0x00000000	Sector 2	0x20220000
	56K	0x00002000	Sector 0	0x20222000
	56K	0x00010000	Sector 1	0x20230000
	8K	0x0001E000	Sector 3	0x2023E000

There are two views present in the ENVM Debug window - Client View and Page View.

3.19.1 Client View

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

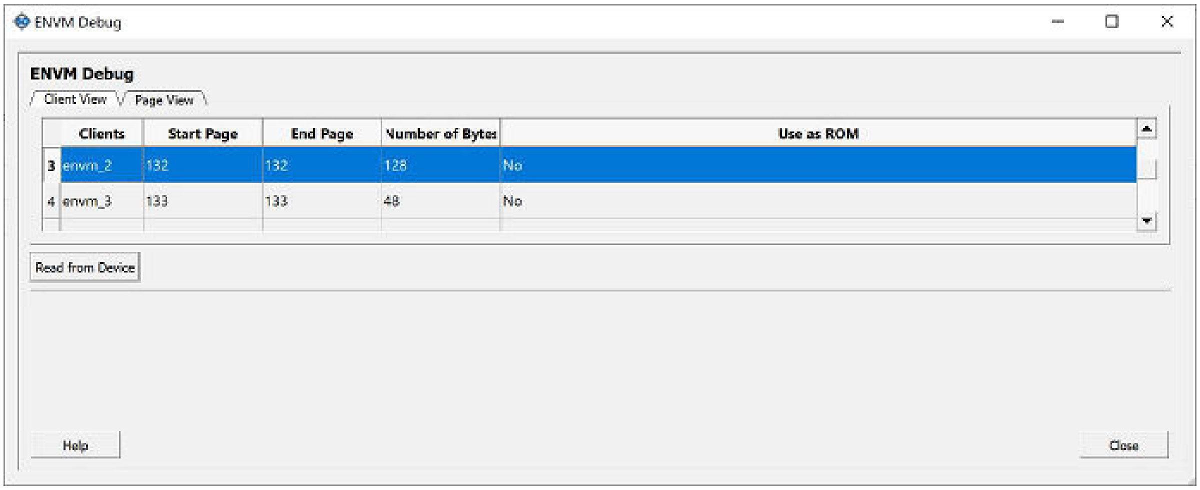
Figure 3-103. ENVM Debug Window with Client View



	Clients	Start Page	End Page	Number of Bytes	Use as ROM
2	envm_1	131	131	10	No
3	envm_2	132	132	128	No

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

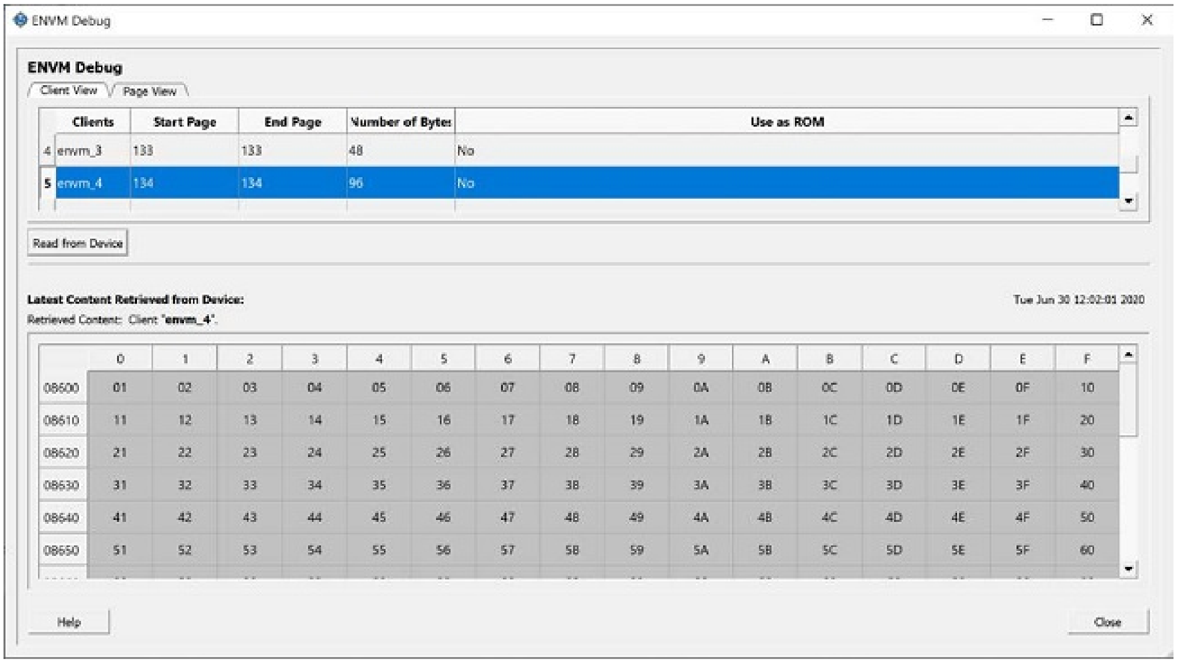
Figure 3-104. Read from Device on Selection



	Clients	Start Page	End Page	Number of Bytes	Use as ROM
3	envm_2	132	132	128	No
4	envm_3	133	133	48	No

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

Figure 3-105. Client View Data



	Clients	Start Page	End Page	Number of Bytes	Use as ROM
4	envm_3	133	133	48	No
5	envm_4	134	134	96	No

Latest Content Retrieved from Device: Tue Jun 30 12:02:01 2020

Retrieved Content: Client "**envm_4**".

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
08600	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
08610	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
08620	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30
08630	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40
08640	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
08650	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60

3.19.2 Page View

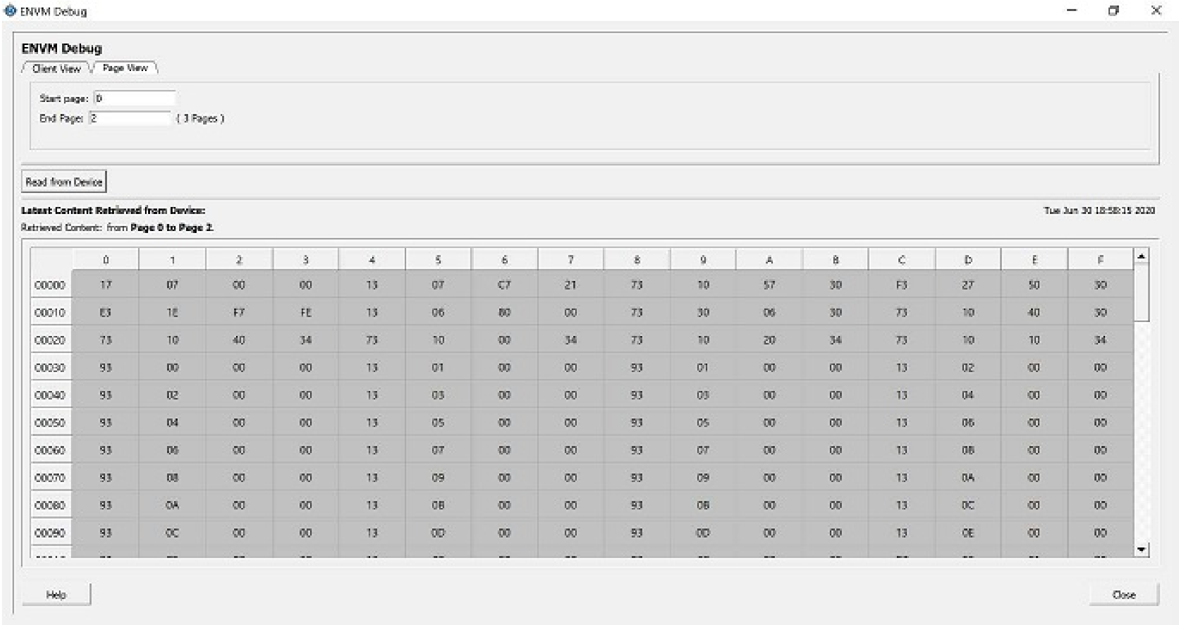
Page view can be seen when the **Page View** tab is selected in the **ENVM Debug** Window.

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

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

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

Figure 3-106. Page View Data



Start page: 0

End Page: 2 (3 Pages)

Latest Content Retrieved from Device: Tue Jun 30 18:58:15 2020

Retrieved Content: from **Page 0 to Page 2.**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000	17	07	00	00	13	07	C7	21	73	10	57	30	F3	27	50	30
00010	E3	1E	F7	FE	13	06	80	00	73	30	06	30	73	10	40	30
00020	73	10	40	34	73	10	00	34	73	10	20	34	73	10	10	34
00030	93	00	00	00	13	01	00	00	93	01	00	00	13	02	00	00
00040	93	02	00	00	13	03	00	00	93	03	00	00	13	04	00	00
00050	93	04	00	00	13	05	00	00	93	05	00	00	13	06	00	00
00060	93	06	00	00	13	07	00	00	93	07	00	00	13	08	00	00
00070	93	08	00	00	13	09	00	00	93	09	00	00	13	0A	00	00
00080	93	0A	00	00	13	0B	00	00	93	0B	00	00	13	0C	00	00
00090	93	0C	00	00	13	0D	00	00	93	0D	00	00	13	0E	00	00

4. SmartDebug Tcl Commands

This section describes the SmartDebug Tcl Commands for the PolarFire, PolarFire SoC, SmartFusion2, IGLOO2 and RTG4 device families.

4.1 add_probe_insertion_point

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(i.e. 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"]'.

Supported Families

PolarFire
SmartFusion2

RTG4
IGLOO2
PolarFire SoC

Example
This example adds probe point to 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

4.2 add_to_probe_group

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

This command will fail if any of the optins 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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.3 check_flash_memory

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.

...........continued

Parameter	Type	Description
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). **Ststus:** 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.
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'.

...........continued

Error Code	Description
None	Missing specification for Flash Memory area. Use one of:-client [-block]or-startpage -endpage -block.

Supported Families

SmartFusion2
RTG4*
IGLOO2*

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

4.4 close_project

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
SmartFusion2
RTG4
IGLOO2

PolarFire SoC

Example

This command closes the SmartDebug project:

```
close_project
```

See Also

- new_project
- open_project

4.5 create_probe_group

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"'.
None	Parameter 'name' has illegal value.

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
This example creates new probe group named "my_new_grp" :

```
create_probe_group -name my_new_grp
```

4.6 debug_ddr

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_ddr [-ddr_type type_of_ddr] \
          [-data_width width] \
          [-slot memory_slot] \
          [-inst_path ddr_instance_path_from_top] \
          [-frequency frequency]

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"
```

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 (e.g., 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: .

...........continued

Error Code	Description
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

4.7 debug_iod

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}
```

Arguments

Parameter	Type	Description
deviceName	sring	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	sring	Specifie iod type. Valid types are: RX_DDRX_B_G_DYN, RX_DDRX_B_R_DYN and RX_DDRX_B_G_FDYN.
inst_path	sring	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 instace 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_ddr

4.8 ddr_read

Description

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

```
ddr_read -deviceName "deive 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. • Specifies which DDR configurator is used in the Libero design. • SmartFusion2 and IGLOO2 - fddr and mddr 56. • RTG4 - east_fddr and west_fddr.
name	string	• 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.
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 Resgister Name"'.

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
Read DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

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

See Also

- ddr_write

4.9 ddr_write

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 blcok name: fddr | mddr | east_fddr | west_fddr. • Specifies which DDR configurator is used in the Libero design. • SmartFusion2 and IGLOO2 - fddr and mddr • RTG4 - east_fddr and west_fddr.
name	string	• 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	• 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 Resgister Name" -value "DDR register value"'.

Supported Families

PolarFire
SmartFusion2
RTG4*
IGLOO2
PolarFire SoC

Example

Write a 16-bit value DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

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

See Also

- ddr_read

4.10 delete_active_probe

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.11 load_live_probe_list*

http://bugzilla.microsemi.net/show_bug.cgi?id=118844

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.12 eye_monitor_power

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}
```

4.13 event_counter

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
SmartFusion2

RTG4
IGLOO2
PolarFire SoC

Example
The following example assigns 'Q_c:DFN1_0:Q' signal to Channel A, runts 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

4.14 execute_dfe_calibration

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. • 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

...........continued

Error Code	Description
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
```

4.15 fhb_control

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.

...........continued

Parameter	Type	Description
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	Sepcifies 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.
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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.16 get_programmer_info

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
SmartFusion2
RTG4
IGLOO2

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

4.17 get_user_clock_frequencies

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

SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
Get 'M4' device user clock frequensy:

```
get_user_clock_frequencies -deviceName "M4"
```

Output:

```
User clock frequency - clocking_0\/PF_out0 value = 100.07 MHz
User clock frequency - clocking_0\/PF_out1 value = 132.02 MHz
```

See Also
- event_counter
- run_frequency_monitor

4.18 import_ddc_file

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
SmartFusion2
RTG4
IGLOO2

Example

This example importes DDC file to the debug project:

```
import_ddc_file -import_ddc {./src/top.ddc} -device_name {SAA300ST}
```

See Also

- new_project

4.19 load_active_probe_list

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
SmartFusion2
RTG4

IGLOO2
PolarFire SoC

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

4.20 load_SI_design_defaults

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.

...........continued

Error Code	Description
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

4.21 loopback_mode

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
lane	string	Specify the physical location of the lane.
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.

...........continued

Error Code	Description
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

4.22 loopback_test

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.

...........continued

Parameter	Type	Description
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

SmartFusion2
RTG4
IGLOO2

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

4.23 move_to_probe_group

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.24 mss_add_register

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
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

4.25 mss_read_register

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 "interger 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

4.26 mss_write_register

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 -reg_name {register_name} -value {hexadecimal_value}
```

Arguments

Parameter	Type	Description
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.

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.

...........continued

Error Code	Description
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

4.27 mss_import_register

Description

This Tcl command imports the register list from a `*.csv` file generated by register export operation.

```
mss_import_register \
           -file_name {aboslute 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

4.28 mss_export_register

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
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

4.29 new_project

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	Specifiy 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	Specifiy 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.

...........continued

Parameter	Type	Description
import_ddc	string	Specifiy 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
SmartFusion2
RTG4
IGLOO2

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

4.30 open_project

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
SmartFusion2
RTG4
IGLOO2

Example

This command opens the 'SDPrj.dprj' project from the SDProject directory:

```
open_project -project {./SDProject/SDPrj.dprj}
```

See Also

- new_project

4.31 optimize_dfe

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 set_debug_device command.

...........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 falied.
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 oftimizes dfe for lane "Q2_LANE0" using software_based algorithm.

```
optimize_dfe -lane {"Q2_LANE0"} -dfe_algorithm {software_based}
```

This example oftimizes dfe for lane "Q2_LANE0" using xcvr_based algorithm.

```
optimize_dfe -lane {"Q2_LANE0"} -dfe_algorithm {xcvr_based}
```

This example oftimizes dfe for lane "Q2_LANE0" and “Q0_LANE0” using xcvr_based algorithm.

```
optimize_dfe -lane {"Q2_LANE0" “Q0_LANE0”} -dfe_algorithm {xcvr_based}
```

4.32 optimize_receiver

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 receiver mode. For CDR mode receivers, CTLE settings can be optimized. For DFE mode receivers, CTLE settings and DFE coefficients can be optimized.

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

```
optimize_receiver [-deviceName "device name"] -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.
lane	string	Specify the physical lane instance name.

Return Type	Description
None	None

Error Codes

Error Code	Description
None	Optimize RECEIVER: DFE Calibration process on lane Q1_LANE0 failed 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 invalid value.
None	Parameter 'lane' has 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}
```

See Also

- optimize_dfe

4.33 pcie_config_space

Description

This Tcl command displays the value of the entered parameter in the SmartDebug log window and return thes register:field value to the Tcl.

```
pcie_config_space -pcie_logical_name {Pcie Logical Name} \
                  -param_name {Pcie Parameter Name}
```

Arguments

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.
param_name	string	Parameter name to read from the device. This parameter is mandatory.

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

4.34 pcie_ltssm_status

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_block_name {pcie_block_name}
```

Arguments

Parameter	Type	Description
pcie_block_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	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"'.

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

4.35 plot_eye

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}
```

4.36 prbs_test

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.
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"] '

...........continued

Error Code	Description
None	Required parameter 'serdes' is missing.
None	serdes: Invalid argument value: 'serdes_value' (expecting integer value).
None	Required parameter 'lane' is missing.

Supported Families

SmartFusion2
RTG4
IGLOO2

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

4.37 program_probe_insertion

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	None

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example

This example runs the probe insertion flow:

```
program_probe_insertion
```

See Also

- add_probe_insertion_point
- remove_probe_insertion_point

4.38 read_active_probe

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

...........continued

Parameter	Type	Description
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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.39 read_envm_memory

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	Specifie the client name.
startpage	string	Specifie the start page that is integer value.
endpage	string	Specifie the end page that is integer value.
fileName	string	Specifie 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"
```

4.40 read_id_code

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example

The following command reads the IDCODE from the current configuration:

```
read_id_code
```

See Also

- set_debug_device
- read_device_status

4.41 read_device_status

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.

...........continued

Parameter	Type	Description
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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example

The following example saves the details of the device in log_file:

```
read_device_status -file {./log_file}
```

See Also

- read_id_code

4.42 read_lsram

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 phyical 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
SmartFusion2

RTG4
IGLOO2
PolarFire SoC

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

4.43 read_usram

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"]

Phisical 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 parametr is ptional. Specifies the output file name for the data read from the device.

...........continued

Parameter	Type	Description
fileName	string	This parametr is ptional. 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 phyical block option to read.
None	Parameter 'logicalBlockName' has illegal value.

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.44 read_snvm_memory*

http://bugzilla.microsemi.net/show_bug.cgi?id=118937

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 clients configured in Libero and If you 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		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.

...........continued

Error Code	Description
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 memeory from the device and save into svnm.txt file:

```
read_snvm_memory -startpage "0" -endpage "2" \
                 -fileName "snvm.txt" -uskKey {0:0:0}
```

4.45 read_uprom_memory

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	Specifie 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.
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"
```

4.46 read_flash_memory

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 Memnory Contet with plain text format.

Error Codes

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

...........continued

Error Code	Description
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

SmartFusion2
IGLOO2*
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

4.47 record_actions

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
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.

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 ba passed as a parameter.
None	Both parameters cannot be passed. Either -start_recording or -stop_recording should ba 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}
```

4.48 remove_from_probe_group

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
SmartFusion2
RTG4
IGLOO2
PoarlFire SoC

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

4.49 remove_probe_insertion_point

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	None

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.50 rescan_programmer

Description

This Tcl command rescans for programmer connected to the PC via the USB port.

Notes:

- This command does not have any arguments.
- In 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. Please 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
SmartFusion2
RTG4
IGLOO2
PoarlFire SoC

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
```

4.51 run_frequency_monitor

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.52 save_active_probe_list

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.53 save_live_probe_list

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
The following example saves live probes list to live_probe_list.txt file:

```
save_live_probe_list -file {./live_probe_list.txt}
```

4.54 scan_ecc_memories

Description
This Tcl command scans and reports any errors detected in the PolarFire, PolarFire SoC, RTG4 and RTPF 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}
```

4.55 select_active_probe

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 [-deviceName device_name ] \
                    [-name probe_name_pattern ] \
                    [-reset true|false ]
```

Arguments

Parameter	Type	Description
deviceName	string	Parameter is optional if only one device is available in the current configuration..
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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.56 serdes_lane_reset

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
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

SmartFusion2
RTG4
IGLOO2

Example

This exmaple resets Lane 0, for specified SERDES block:

```
serdes_lane_reset -serdes 0 -lane 0
```

See Also

- serdes_read_regster
- serdes_write_regster

4.57 serdes_read_register

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
serdes	integer	Specify SERDES block number. Must be between 0 and 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

SmartFusion2

RTG4
IGLOO2

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

4.58 serdes_write_register

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
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.

...........continued

Error Code	Description
None	Parameter 'name' has illegal value.
None	Required parameter 'value' is missing.
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

SmartFusion2
RTG4
IGLOO2

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

4.59 set_debug_programmer

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"'
NoneThe following example selects the programmer 10841	Parameter 'name' has illegal value.

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
The following example selects the programmer 10841:

```
set_debug_programmer -name {10841}
```

4.60 set_live_probe

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 (i.e. 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.
None	Cannot set live probes: Probe A is not found.
None	Cannot set live probes: IDCode verify failed.

Supported Families

PolarFire
SmartFusion2
RTG4
IGLOO2
PolarFire SoC
RTPF*

Exception

- The array must be programmed and active
- Active probe read or write operation will affect current settings of Live probe since they use 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
- In order to be available for Live probe, ProbeA and ProbeB I/O's 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

4.61 signal_integrity_export

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 {phisical 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
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
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

4.62 signal_integrity_import

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 {phisical 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
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.
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

4.63 signal_integrity_write

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, 17.0bdB and 25.0dB.
RX_CTLE	string	RX CTLE value.
RX_CDR_GAIN	string	Specifies CDR Gain value. It can be "Low" or "High".

...........continued

Parameter	Type	Description
RX_TERMINATION	integer	Specifies RX Termination(ohms). Possible values are: 85, 100 and150.
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.
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.

...........continued

Error Code	Description
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

4.64 smartbert_test

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
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

4.65 static_pattern_transmit

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
lane	string	Specify Transceiver physical Lane Name.
start	none	Start the Static Pattern Transmit.
stop	none	Stop the Static Pattern Transmit.

...........continued

Parameter	Type	Description
pattern	string	Specify "pattern_type" of Static Pattern Transmit. "pattern_type" valid values are: • 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_LANE1}
```

4.66 transceiver_lane_reset

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}
```

4.67 ungroup

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

Example
Ungroup 'my_group' probe groups:

```
ungroup -name "my_group"
```

See Also

- create_probe_group
- add_to_probe_group

4.68 unset_live_probe

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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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

4.69 write_active_probe

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.

...........continued

Error Code	Description
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
SmartFusion2
RTG4
IGLOO2
PolarFire SoC

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 grp1.out[3]:out[3]:Q -value "low"
```

This example writes the value to complete group:

```
write_active_probe -group_name grp1 –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

4.70 write_lsram

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 -name {physical block name} \
            -offset {offset value} \
            -value {integer value} \
            [-tpsramValue "TPSRAM physical block word value"]
Logical block
write_lsram -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 "USRAM 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

...........continued

Error Code	Description
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 phyical block option to read.
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
SmartFusion2
RTG4
IGLOO2
PolarFie SoC

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

4.71 write_usram

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.

...........continued

Error Code	Description
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 phyical 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
SmartFusion2
RTG4
IGLOO2
PolarFie SoC

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

4.72 xcvr_add_register

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

4.73 xcvr_export_register

Description

This Tcl command exports previously added transceiver registers details to a *.csv file.

```
xcvr_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'xcvr_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:

```
xcvr_export_register -file_name {register_export.csv}
```

See Also

- xcvr_add_register
- xcvr_read_register
- xcvr_write_register

4.74 xcvr_import_register

Description

This Tcl command imports exported transceiver registers details from a *.csv file.

```
xcvr_import_register \
            -file_name {absoulte or relative path to the *.csv file name}
```

Arguments

Parameter	Type	Description
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}
```

4.75 xcvr_read_register

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 {instanse 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.

...........continued

Parameter	Type	Description
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

4.76 xcvr_write_register

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'xcvr_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

```
xcvr_write_register -inst_name {CM1_PCIe_SS_0/PF_PCIE_0/LANE1} \
                    -reg_name {GSSCLK_CTRL} \
                    -value 0xffffffff
```

See Also

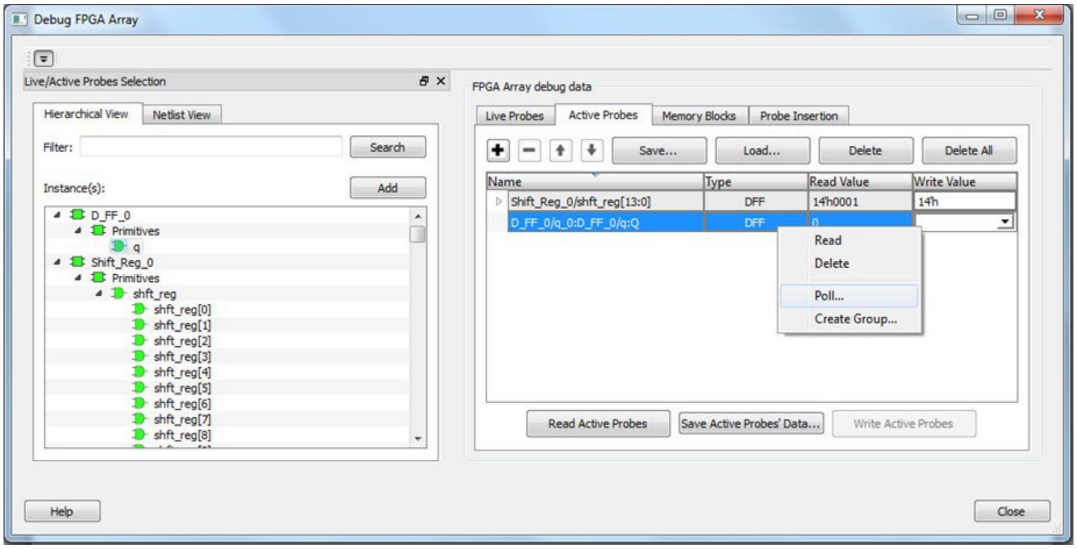
- xcvr_add_register
- xcvr_export_register
- xcvr_read_register

5. Frequently Asked Questions

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

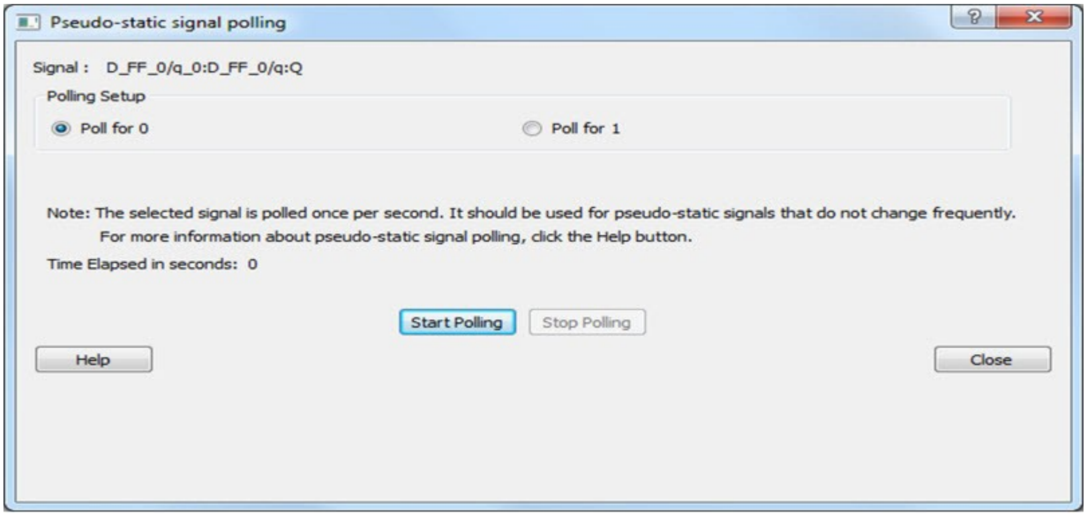
To monitor a static or pseudo-static signal:

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



Name	Type	Read Value	Write Value
Shift_Reg_0/shft_reg[13:0]	DFF	14'h0001	14'h
D_FF_0/q_0:D_FF_0/q:Q	DFF	0	

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



Signal : D_FF_0/q_0:D_FF_0/q:Q

● Poll for 0 ○ Poll for 1

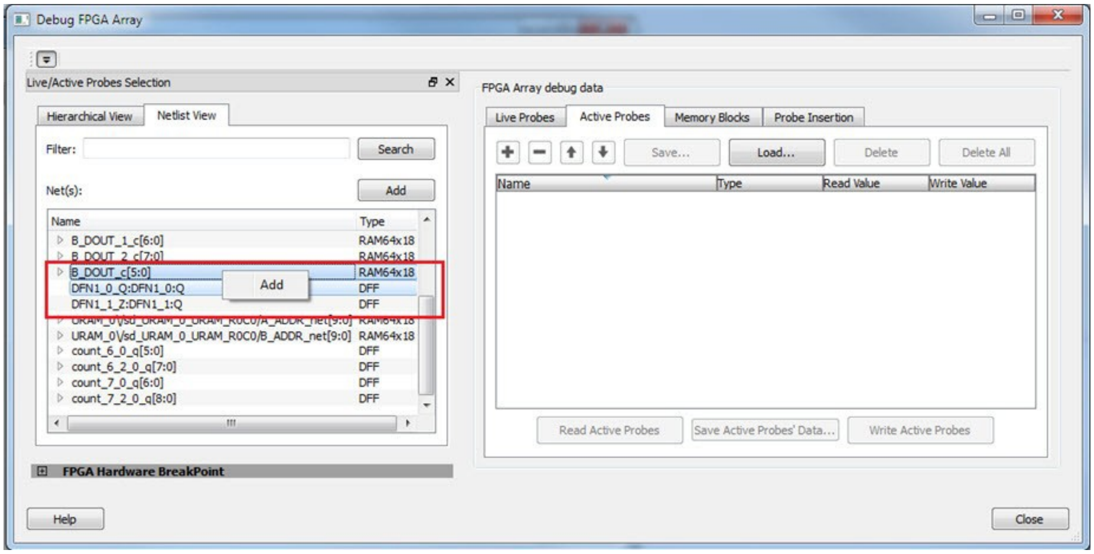
Note: The selected signal is polled once per second. It should be used for pseudo-static signals that do not change frequently.
For more information about pseudo-static signal polling, click the Help button.

Time Elapsed in seconds: 0

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

To force a signal to a new value:

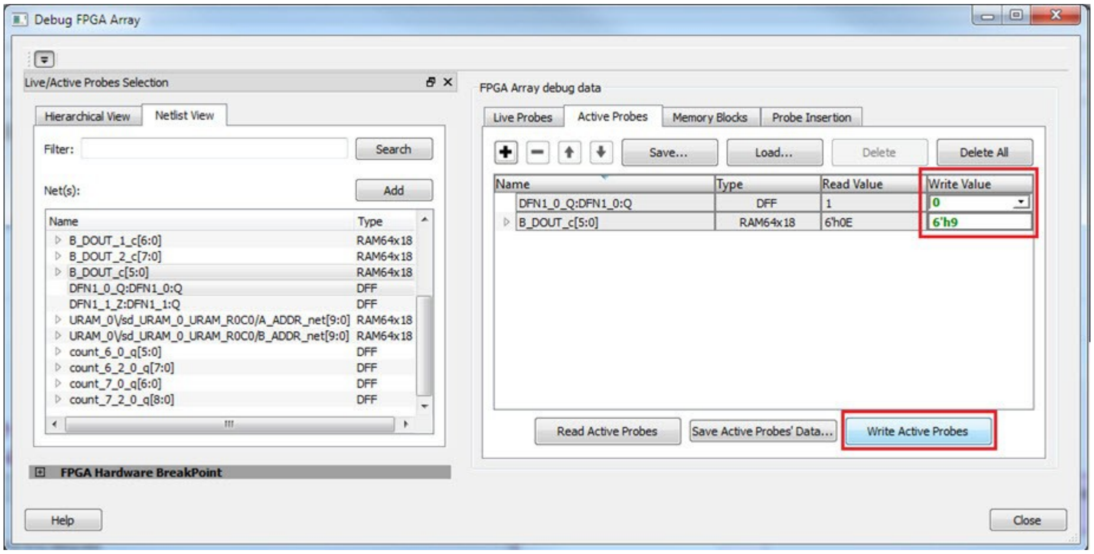
1. In the SmartDebug window, click **Debug FPGA Array**.
2. Click the **Active Probes** tab.
3. Select the signal from the selection panel and add it to Active Probes tab.



Name	Type
B_DOUT_1_c[6:0]	RAM64x18
B_DOUT_2_c[7:0]	RAM64x18
B_DOUT_c[5:0]	RAM64x18
DFN1_0_Q:DFN1_0:Q	DFF
DFN1_1_Z:DFN1_1:Q	DFF
URAM_0\/sd_URAM_0_URAM_R0C0/A_ADDR_net[9:0]	RAM64x18
URAM_0\/sd_URAM_0_URAM_R0C0/B_ADDR_net[9:0]	RAM64x18
count_6_0_q[5:0]	DFF
count_6_2_0_q[7:0]	DFF
count_7_0_q[6:0]	DFF
count_7_2_0_q[8:0]	DFF

Name	Type	Read Value	Write Value

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**.



Name	Type
B_DOUT_1_c[6:0]	RAM64x18
B_DOUT_2_c[7:0]	RAM64x18
B_DOUT_c[5:0]	RAM64x18
DFN1_0_Q:DFN1_0:Q	DFF
DFN1_1_Z:DFN1_1:Q	DFF
URAM_0\/sd_URAM_0_URAM_R0C0/A_ADDR_net[9:0]	RAM64x18
URAM_0\/sd_URAM_0_URAM_R0C0/B_ADDR_net[9:0]	RAM64x18
count_6_0_q[5:0]	DFF
count_6_2_0_q[7:0]	DFF
count_7_0_q[6:0]	DFF
count_7_2_0_q[8:0]	DFF

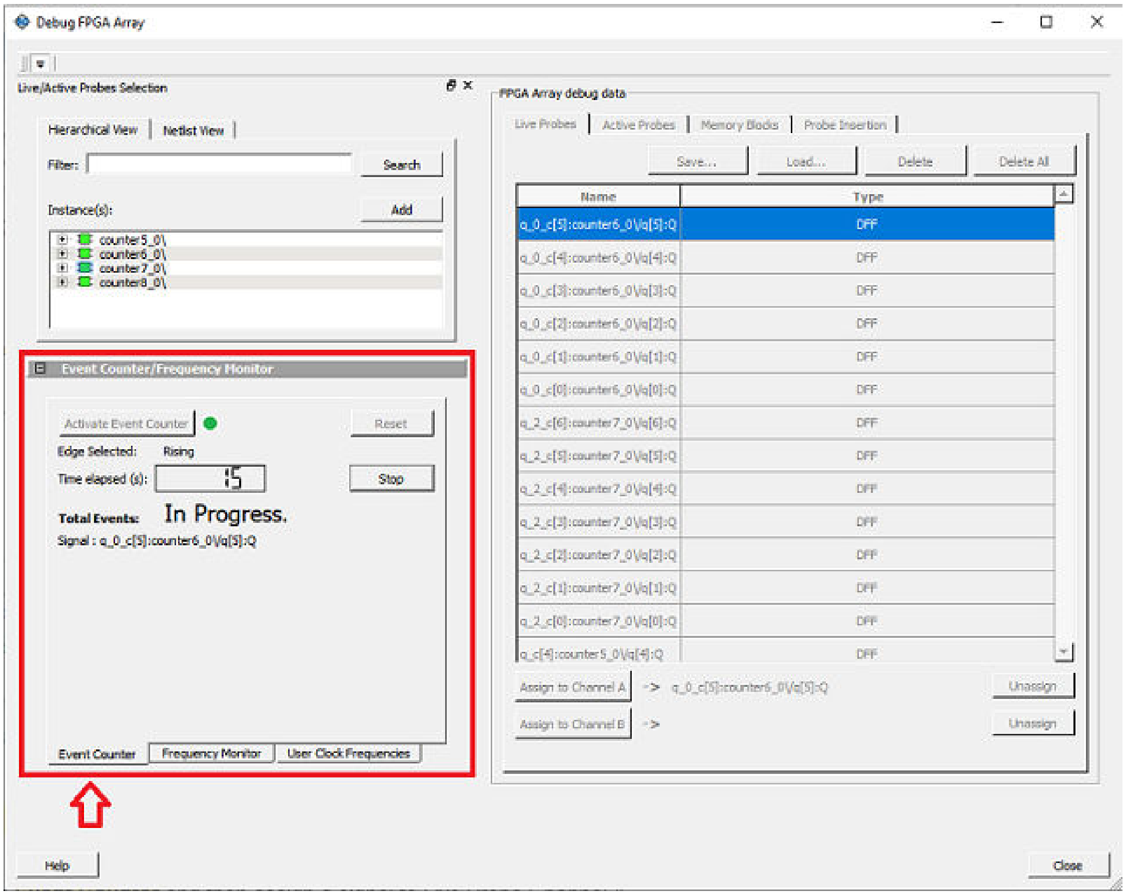
Name	Type	Read Value	Write Value
DFN1_0_Q:DFN1_0:Q	DFF	1	0
B_DOUT_c[5:0]	RAM64x18	6'h0E	6'h9

5.3 How do I count the transitions on a signal?

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

To count the transitions on a signal:

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



Edge Selected: Rising

Time elapsed (s): 15

Total Events: In Progress.

Signal : q_0_c[5]:counter6_0\/q[5]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_0_c[5]:counter6_0\/q[5]:Q

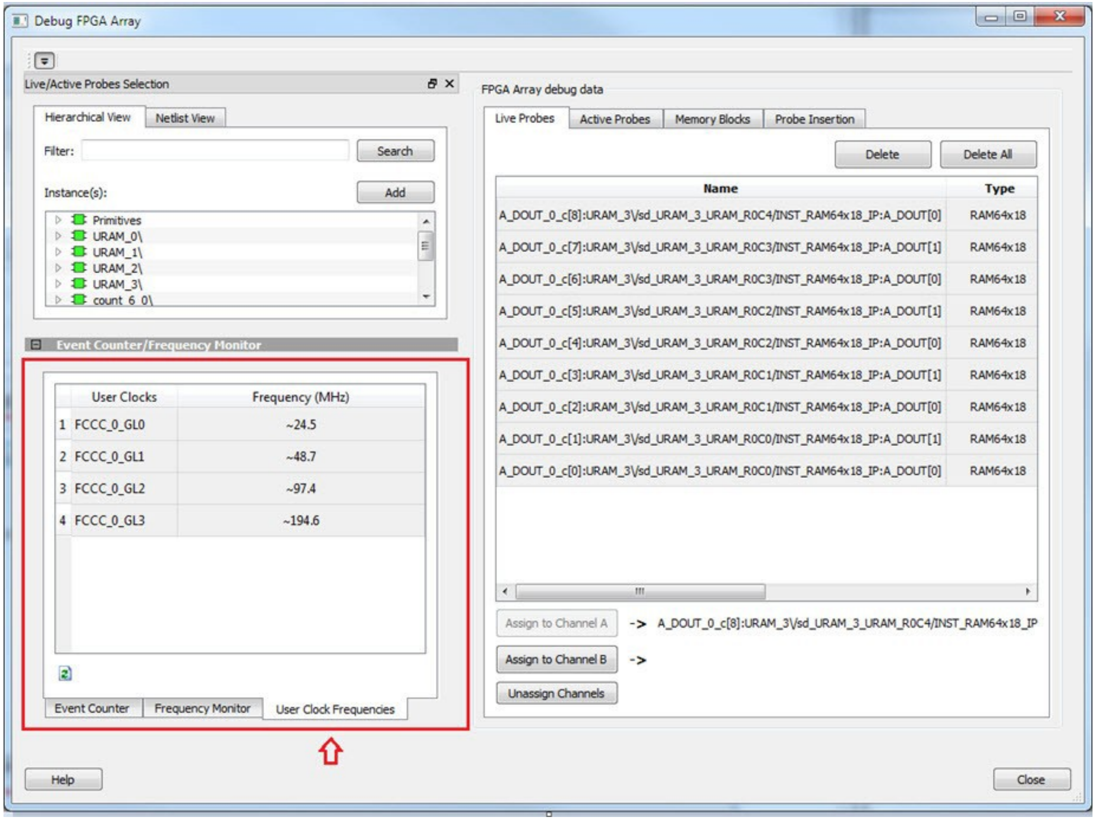
Assign to Channel B ->

5.4 How do I monitor or measure a clock?

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

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

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



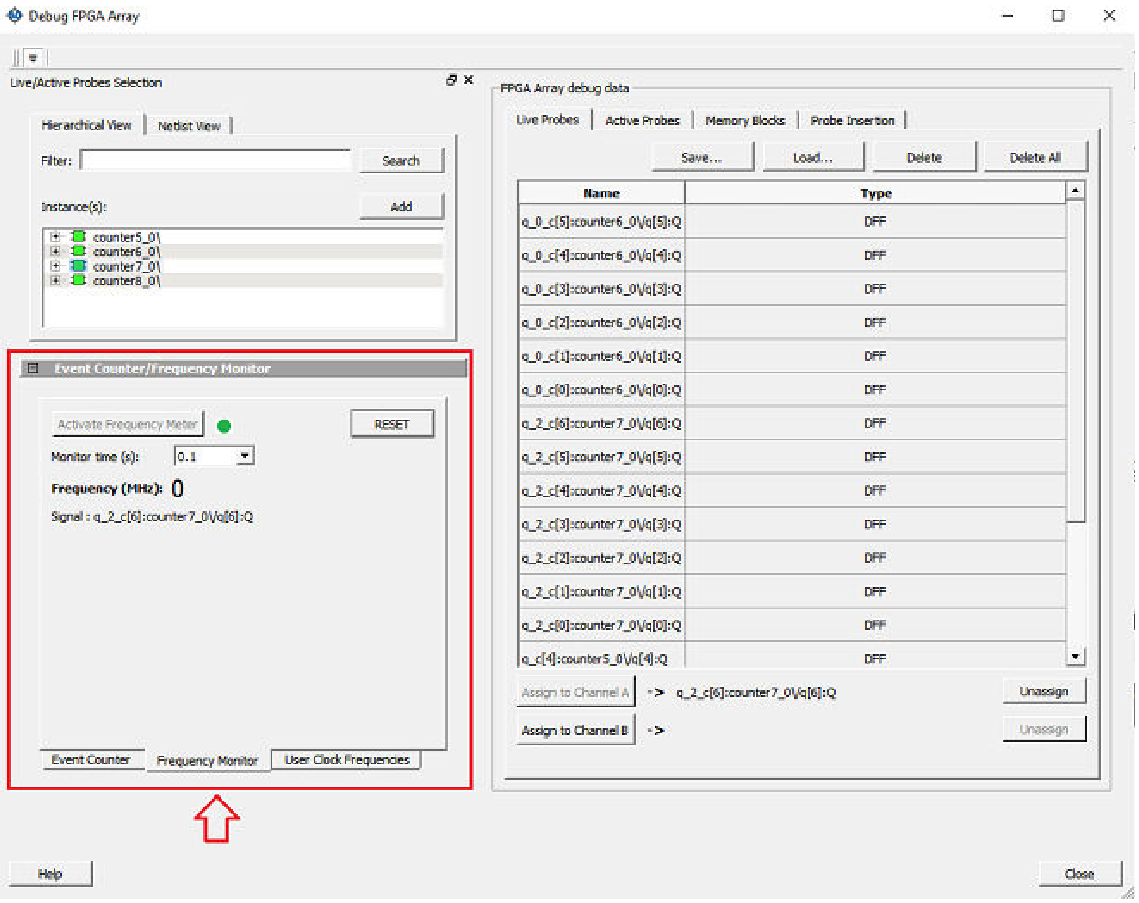
	User Clocks	Frequency (MHz)
1	FCCC_0_GL0	~24.5
2	FCCC_0_GL1	~48.7
3	FCCC_0_GL2	~97.4
4	FCCC_0_GL3	~194.6

Name	Type
A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[7]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[4]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
A_DOUT_0_c[1]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
A_DOUT_0_c[0]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18

Assign to Channel A -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP

Assign to Channel B ->

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



Filter:

Instance(s):

- counter5_0\
- counter6_0\
- counter7_0\
- counter8_0\

Monitor time (s): 0.1

Frequency (MHz): 0

Signal : q_2_c[6]:counter7_0\/q[6]:Q

Name	Type
q_0_c[5]:counter6_0\/q[5]:Q	DFF
q_0_c[4]:counter6_0\/q[4]:Q	DFF
q_0_c[3]:counter6_0\/q[3]:Q	DFF
q_0_c[2]:counter6_0\/q[2]:Q	DFF
q_0_c[1]:counter6_0\/q[1]:Q	DFF
q_0_c[0]:counter6_0\/q[0]:Q	DFF
q_2_c[6]:counter7_0\/q[6]:Q	DFF
q_2_c[5]:counter7_0\/q[5]:Q	DFF
q_2_c[4]:counter7_0\/q[4]:Q	DFF
q_2_c[3]:counter7_0\/q[3]:Q	DFF
q_2_c[2]:counter7_0\/q[2]:Q	DFF
q_2_c[1]:counter7_0\/q[1]:Q	DFF
q_2_c[0]:counter7_0\/q[0]:Q	DFF
q_c[4]:counter5_0\/q[4]:Q	DFF

Assign to Channel A -> q_2_c[6]:counter7_0\/q[6]:Q

Assign to Channel B ->

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

5.5 How do I perform simple SmartBERT tests?

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

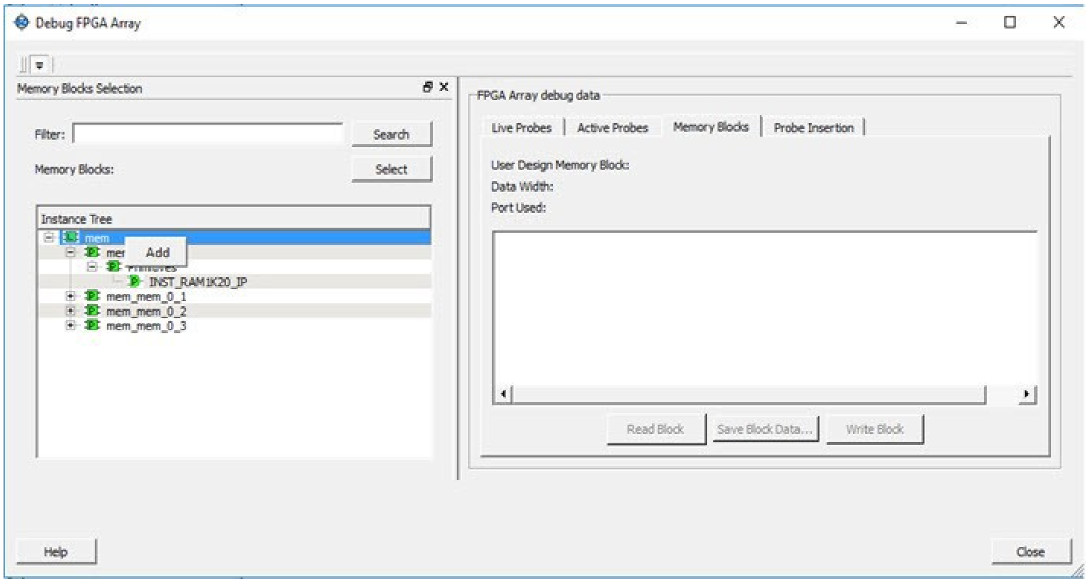
To perform a SmartBERT test, in the **SmartBERT** page of the **Debug Transceiver** dialog box, select to run a PRBS test on-die or off-die with EQ-NEAREND checked or unchecked. For more information, see 3.14.3 SmartBERT.

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

5.6 How do I read LSRAM or USRAM content?

To read RAM content:

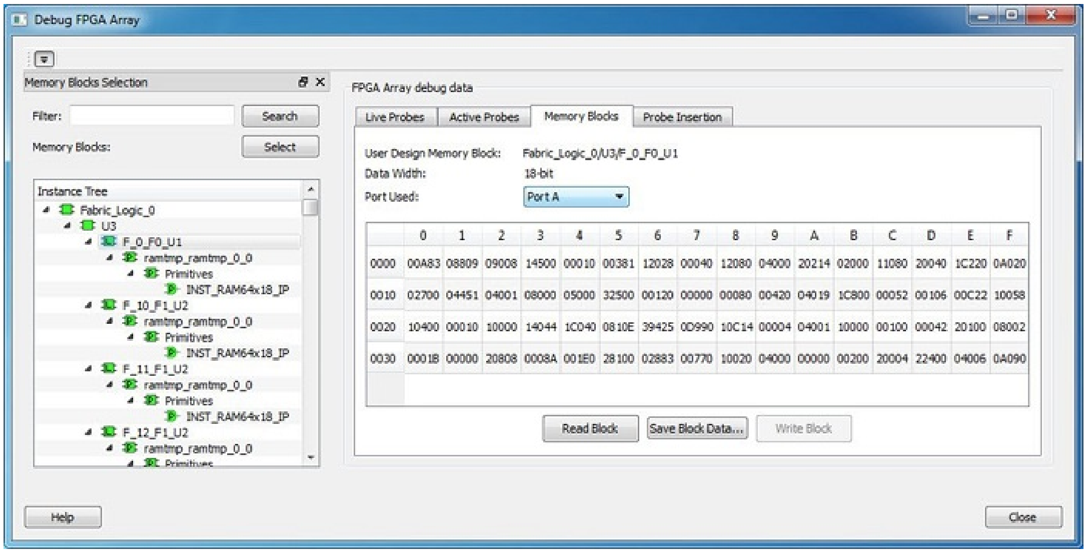
1. In the Debug FPGA Array dialog box, click the **Memory Blocks** tab.
2. Select the memory block to be read from the selection panel on the left of the window.



User Design Memory Block:
Data Width:
Port Used:

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

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



User Design Memory Block: Fabric_Logic_0/U3/F_0_F0_U1
Data Width: 18-bit
Port Used: Port A

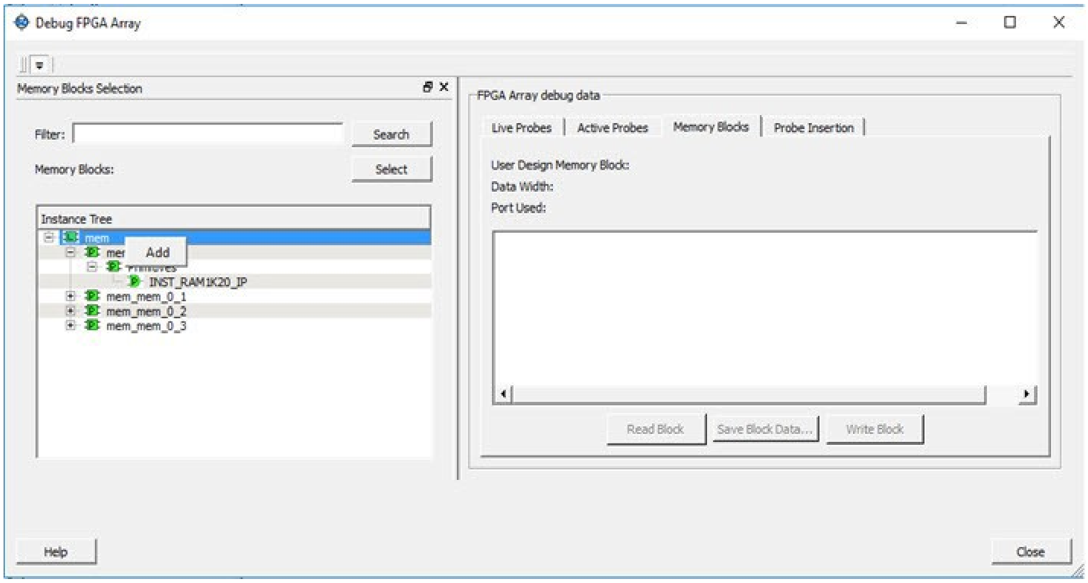
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	00A83	08809	09008	14500	00010	00381	12028	00040	12080	04000	20214	02000	11080	20040	1C220	0A020
0010	02700	04451	04001	08000	05000	32500	00120	00000	00080	00420	04019	1C800	00052	00106	00C22	10058
0020	10400	00010	10000	14044	1C040	0810E	39425	0D990	10C14	00004	04001	10000	00100	00042	20100	08002
0030	0001B	00000	20808	0008A	001E0	28100	02883	00770	10020	04000	00000	00200	20004	22400	04006	0A090

For more information, see Memory Blocks .

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

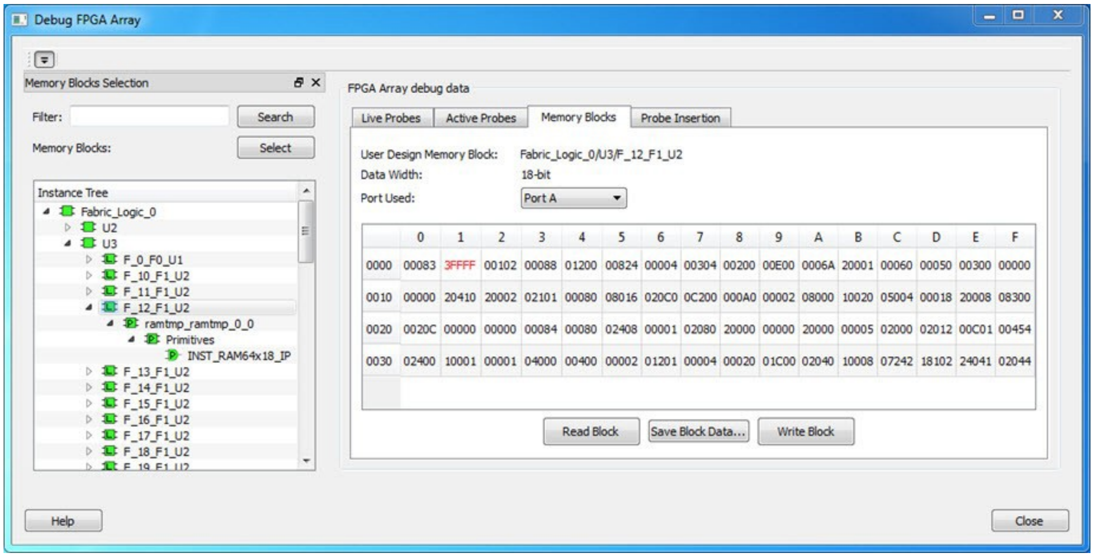
To change the content of LSRAM or USRAM:

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



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

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



User Design Memory Block: Fabric_Logic_0/U3/F_12_F1_U2
Data Width: 18-bit
Port Used: Port A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	00083	3FFFF	00102	00088	01200	00824	00004	00304	00200	00E00	0006A	20001	00060	00050	00300	00000
0010	00000	20410	20002	02101	00080	08016	020C0	0C200	000A0	00002	08000	10020	05004	00018	20008	08300
0020	0020C	00000	00000	00084	00080	02408	00001	02080	20000	00000	20000	00005	02000	02012	00C01	00454
0030	02400	10001	00001	04000	00400	00002	01201	00004	00020	01C00	02040	10008	07242	18102	24041	02044

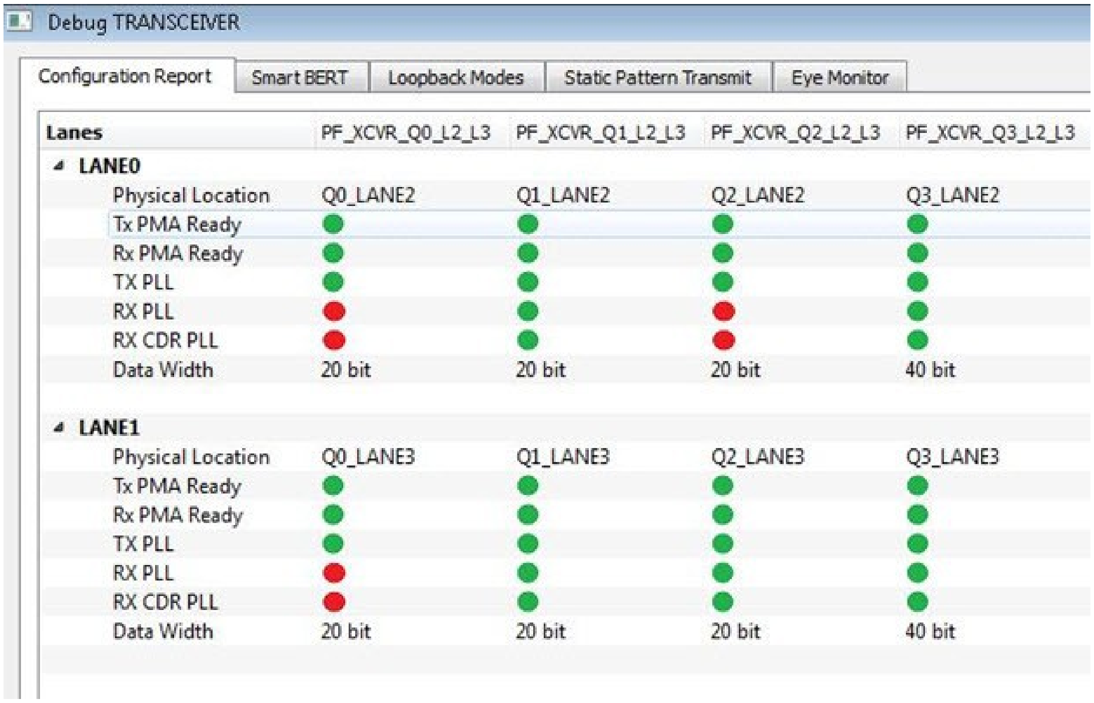
For more information, see Memory Blocks .

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

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

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

   Figure 5-1. Debug Transceiver



Lanes	PF_XCVR_Q0_L2_L3	PF_XCVR_Q1_L2_L3	PF_XCVR_Q2_L2_L3	PF_XCVR_Q3_L2_L3
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.

6. Revision History

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

Revision	Date	Description
C	08/2021	• Added 2.4 Rescan Programmer. • Added 3.9.2 FHB Status. • Updated 3.15 MSS Register Access to add **Import** and **Delete All** button details. • Added 4. SmartDebug Tcl Commands.
B	04/2021	• Added 3.15 MSS Register Access. • Added 3.16 Debug IOD. • Updated 3.14.7 Register Access.
A	11/2020	• 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	• 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.

7. Microchip FPGA Technical Support

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

7.1 Customer Service

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

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

7.2 Customer Technical Support

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

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

Technical support can be reached at soc.microsemi.com/Portal/Default.aspx.

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

You can track technical cases online by going to My Cases.

7.3 Website

You can browse a variety of technical and non-technical information on the Microchip FPGA Products Group home page, at www.microsemi.com/soc.

7.4 Outside the U.S.

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

Visit About Us for sales office listings and corporate contacts.

The Microchip Website

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

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

Product Change Notification Service

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

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

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

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

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

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

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

Legal Notice

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

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

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

Trademarks

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

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

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

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

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

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

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

ISBN: 978-1-5224-8569-8

Quality Management System

For information regarding Microchip’s Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra’anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820