TU0530
Tutorial
SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools - Libero SoC v11.8 SP1

Microsemi
Power Matters.™

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

About Microsemi

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

50200530. 10.0 9/17

Contents

Figures

Tables

1 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.

1.1 Revision 10.0

The following is a summary of changes made in this revision:

- Updated a tcl script for SerDes and PCIe registers (see Example 4: Sample Script to Read SerDes and PCIe Registers, page 57).
- Updated the FPGA Hardware Breakpoint Auto Instantiation section for opening .vcd in any viewer (see FPGA Hardware Breakpoint Auto Instantiation, page 22).
- Updated about sorting options in the Debug Fabric SRAM Memory section (see Debug Fabric SRAM Memory, page 26).
- Added a section for displaying tooltip (see Displaying Tooltip, page 29).
- Added a section for Demo mode (see Demo Mode, page 50).

1.2 Revision 9.0

The document (published in April 2017) was updated for Libero v11.8 software release.

1.3 Revision 8.0

Updated the document for Libero v11.7 software release (SAR 75567).

1.4 Revision 7.0

Updated the document for Libero v11.6 software release (SAR 68374).

1.5 Revision 6.0

Updated the document for Libero v11.5 software release (SAR 62936).

1.6 Revision 5.0

Updated the document for SERDES core change (SAR 61612).

1.7 Revision 4.0

The following is a summary of the changes in revision 4.0 of this document.

- Updated the document for Libero v11.4 software release (SAR 59069).
- Updated the document for M2S025 Evaluation Kit board details (SAR 59069).
- Updated the document for M2GL010 Evaluation Kit board details (SAR 59069).

1.8 Revision 3.0

Added a note in Specifying Live Probe Points in Libero, page 12 (SAR 56593).

1.9 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Updated the software version from 11.2 SP1 to 11.3 (SAR 56012).
- Updated design files using the latest 11.3 SERDES core (SAR 56012).

1.10 Revision 1.0

Revision 1.0 was the first publication of this document.

2 SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools

Design debug is a critical phase of the FPGA design flow. Microsemi multiple design debug tools and features complement design simulations by allowing verification and troubleshooting at the hardware level. Microsemi SmartDebug tools help the designer to analyze the key elements of a flash design, such as the embedded non-volatile memory (eNVM) data, SRAM data, and probes capabilities. Microsemi SmartFusion®2 system-on-chip (SoC) FPGA and IGLOO®2 FPGA devices have built-in probe points that greatly enhance the ability to debug logic elements within the device. The enhanced debug features implemented in the SmartFusion2 and IGLOO2 devices give access to any logic element through live probe and active probe features, which enable designers to check the state of inputs and outputs in real-time, without any re-layout of the design.

2.1 Design Requirements

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
IGLOO2 Evaluation Kit (M2GL010T) - 12 V adapter - FlashPro4 - USB A to mini-B cable or *SmartFusion2 Security Evaluation Kit (M2S090TS)* - 12 V adapter - FlashPro4 - USB A to mini-B cable	Rev D or later
SMA Male-to-SMA Male Precision Cables, such as *Pasternack Industries part number PE39429-12* (or equivalent)	Optionally recommended for evaluation board SerDes testing
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® SoC software	v11.8 SP1
FlashPro programming software	v11.8 SP1

2.1.1 Reference Documents

For more information about SmartDebug, see the *SmartDebug for Libero v11.8 SP1 User Guide*.

2.1.2 Project Files

Extract the *http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_tu0530_liberov11p8SP1_df*

Libero SoC project along with the `Readme.txt` file and programming (`.stp`) file to a folder on the PC (for example: *C:\Microsemiprj*). Confirm that the following design files are extracted from the downloaded folder:

- m2gl_SmartDebug_Tutorial - For IGLOO2 Evaluation Kit (M2GL010T)
- m2s_SmartDebug_Tutorial - For SmartFusion2 Security Evaluation Kit (M2S090TS)

2.2 Design Overview

The design consists of two main blocks: the SerDes debug block (SERDES_Debug) and the fabric debug block (Fabric_Debug), as shown in the following figure.

The SERDES_Debug block is used to demonstrate the SmartDebug capabilities that can be used to perform SerDes real-time signal integrity testing and debugging. The design consists of a System Builder block (SD_DEMO) and an instance of SerDes Interface block (SERDES_IF), as shown in Figure 2, page 4. Within the System Builder, a data storage client is stored in the flash memory (eNVM). SmartDebug provides the capabilities to view the eNVM content by reading the content in real-time from the device.

The Fabric_Debug block demonstrates the way to use SmartDebug to perform FPGA array debugging. To demonstrate this, the Fabric_Debug uses a counter to load a counting pattern into the LSRAM instance (DPSRAM). The data stored is the same as the address. On the read side of the LSRAM, there is a count checker (count_chk) to ensure that the count progresses as expected. If there is an error, the output (error) is latched high, as shown in Figure 3, page 4. This Fabric_Debug block design is used to demonstrate the different silicon built-in capabilities, such as setting live probes to monitor an internal user-selected point on the device in real-time.

In addition, you can set active probes, which provide the capabilities for dynamic asynchronous read and write to a flip-flop or probe point. This enables you to quickly observe the output of the logic internally or to quickly experiment on how the logic is affected by writing to a probe point. Finally, the Fabric_Debug design block is used to demonstrate the SmartDebug capabilities, where you can read and modify the fabric SRAM content in real-time.

Figure 1 • **SmartDebug Top-Level Blocks**

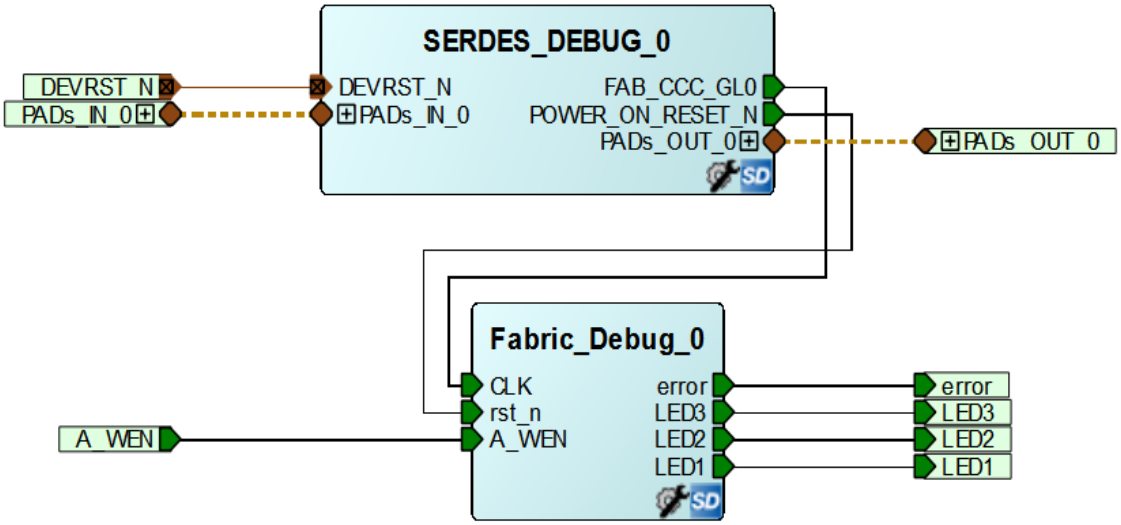


Figure 2 • **SERDES_Debug Overall Design Blocks (IGLOO2 Design Block)**

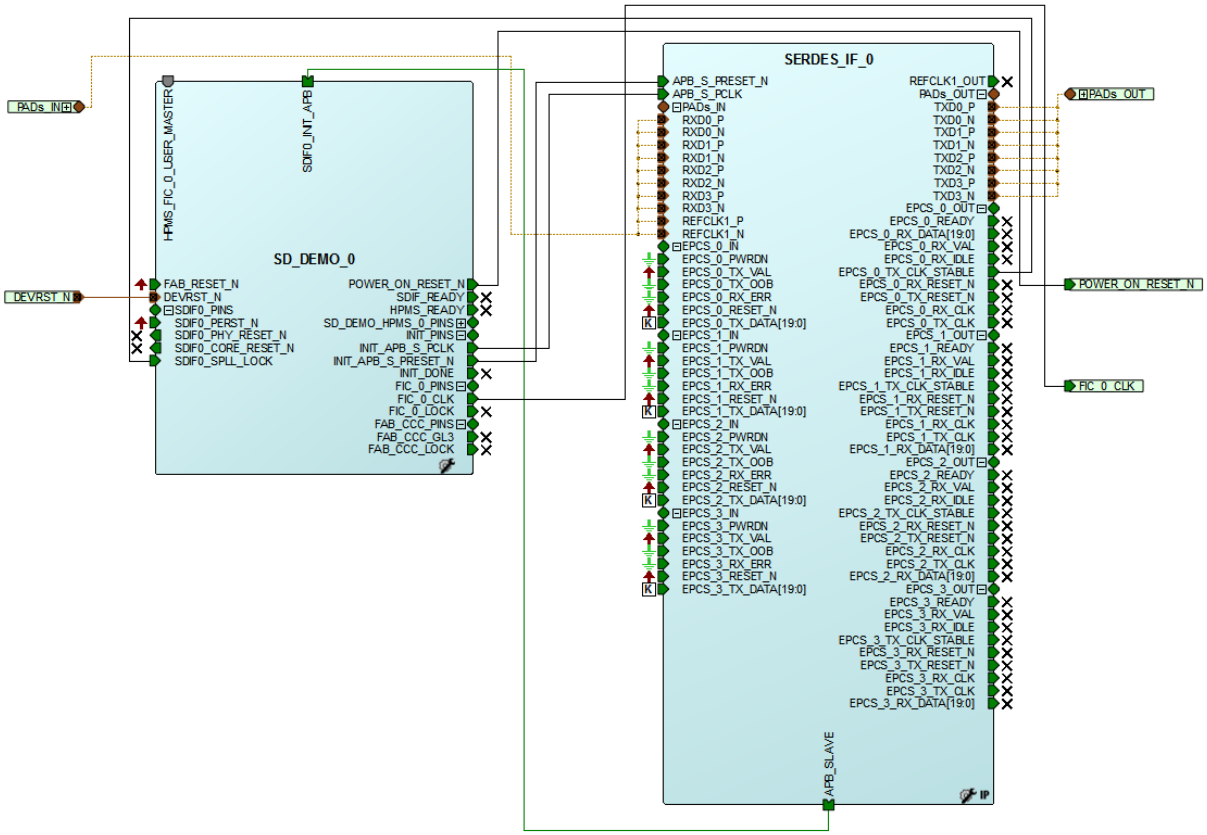
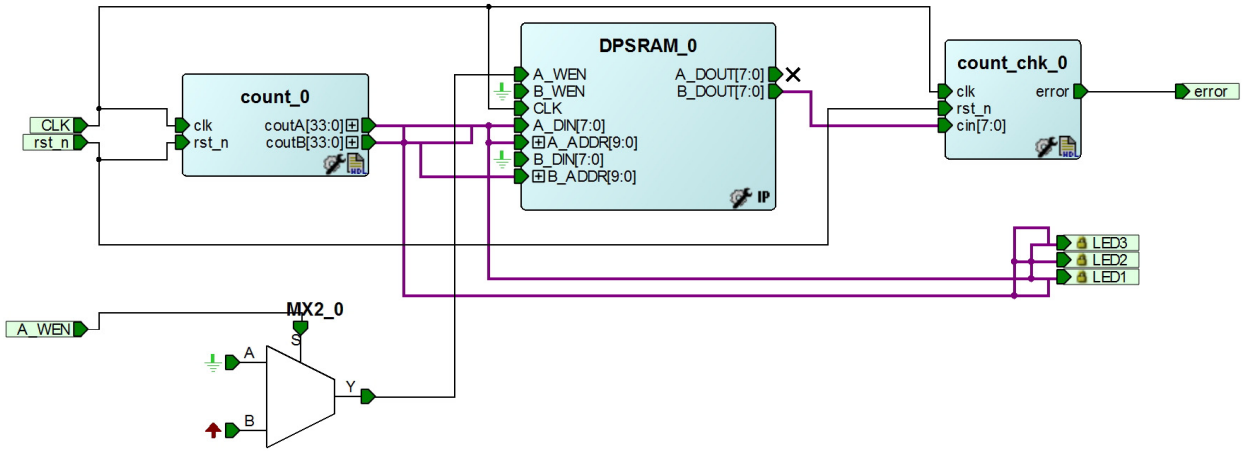


Figure 3 • **Fabric_Debug Overall Design Blocks (IGLOO2 Design Block)**



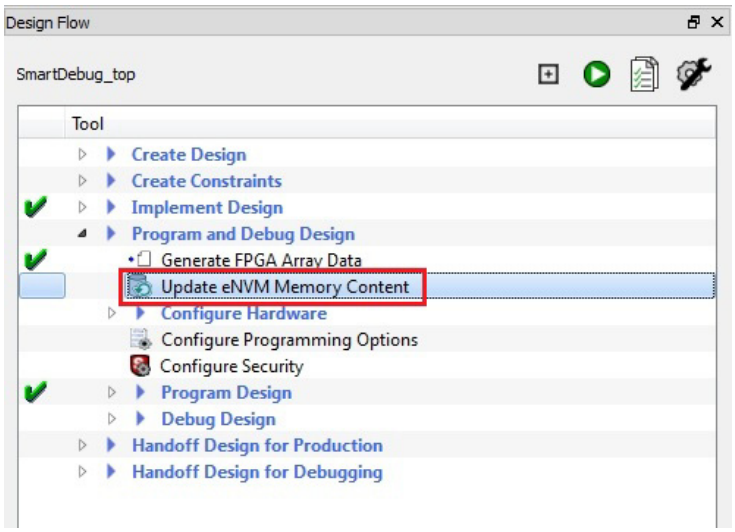
A_WEN is used to pause the write operation to the SRAM while demonstrating the SmartDebug write to the SRAM capability. It is assigned to SW2 on the board. When SW2 is pressed, the write operation from the counter pauses, and does not overwrite the SmartDebug write into the SRAM.

2.3 Programming the Device

The following steps describe how to program the IGLOO2 or SmartFusion2 Security Evaluation Kit board:

1. Connect the FlashPro4/5 programmer to the J5 connector on the IGLOO2 or SmartFusion2 Security Evaluation Kit.
2. Connect the power supply to the J6 connector.
3. Switch on the power supply, SW7. For more information, see the *IGLOO2 FPGA Evaluation Kit Board* or *SmartFusion2 Security Evaluation Kit Board*.
4. Launch Libero SoC v11.8 SP1.
5. From the **Project** menu, select **Open Project**.
6. Browse to the folder where the design files are extracted and open the appropriate design file (IGLOO2 or SmartFusion2). For more information, see Project Files, page 2.
   Based on the location where the project files are extracted, the paths in the eNVM data clients need to be updated.
7. On the **Design Flow** window, double-click **Update eNVM Memory Content** as shown in the following figure.

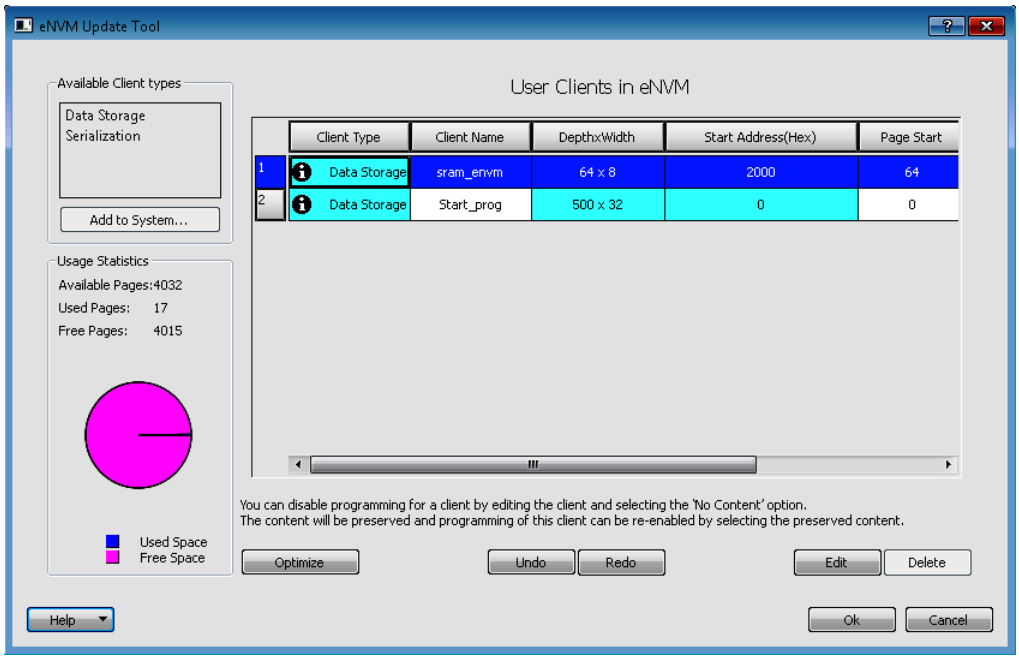
Figure 4 • **Update eNVM Memory Content in Design Flow Window**



The **eNVM Update Tool** window is displayed, as shown in the following figure.

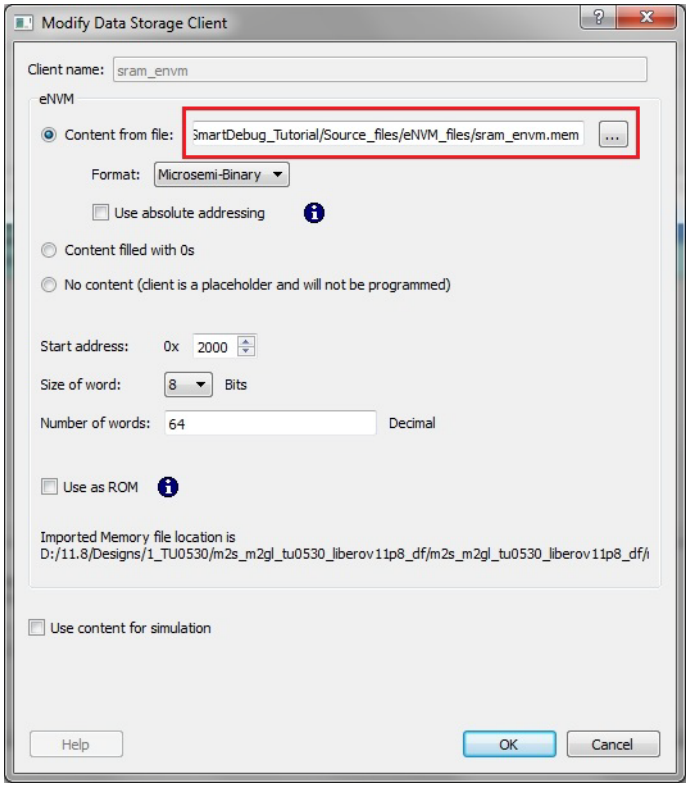
8. Double-click the **sram_envm** client.

Figure 5 • **eNVM Update Tool Window**



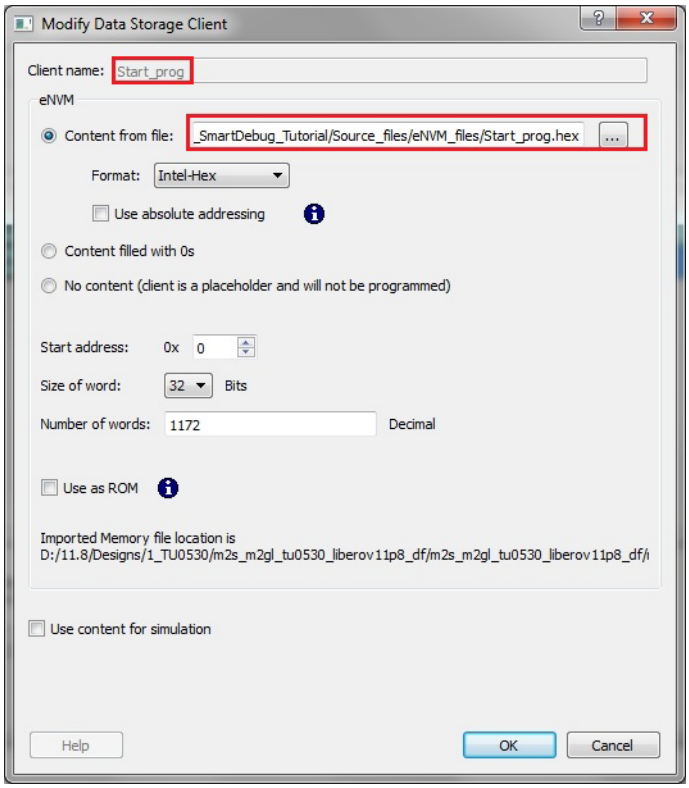
9. In the **Modify Data Storage Client** window, browse to the location of the `sram_envm.mem` file located in the eNVM_files folder included in the project, as shown in the following figure.

Figure 6 • **Modify Data Storage Client Window**



10. If the SmartFusion2 Security Evaluation Kit is used, double-click the **Start_prog** client (see Figure 5, page 6). In the **Modify Data Storage Client** window, browse to the location of the `Start_prog.hex` file located in the eNVM_files folder included in the project, as shown in the following figure.

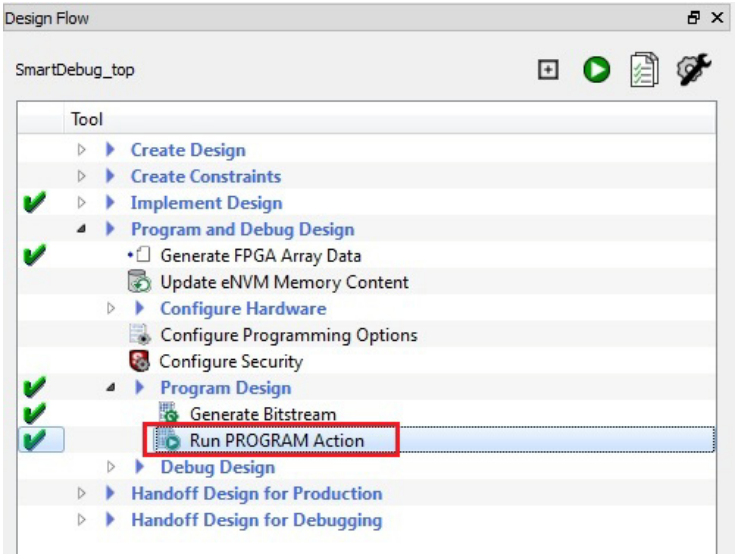
Figure 7 • **Modify Data Storage Client Window - Specifying Start_prog.hex File**



Note: The `.hex` file is a simple user boot code loop program that is programmed into address zero (eNVM address 0x60000000). This is to ensure there is a valid user boot code for the ARM Cortex-M3 to execute on power-up or at power-on reset. For more information, see *http://soc.microsemi.com/kb/article.aspx?id=SL5636*

11. In the **Design Flow** window, select **Run PROGRAM Action,** as shown in the following figure. This programs the design into the device.

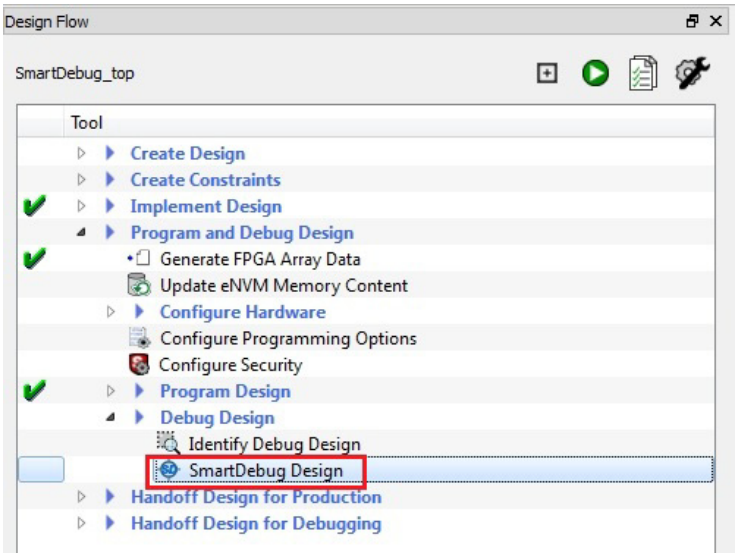
Figure 8 • **Programming the Device**



2.4 Launching SmartDebug from Libero

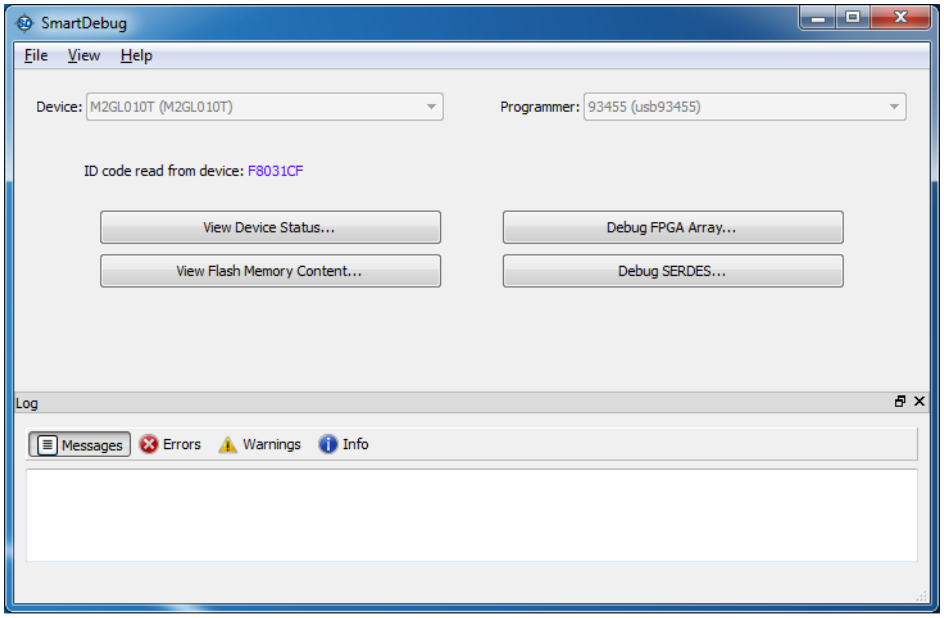
On the **Design Flow** window, double-click **SmartDebug Design**, as shown in the following figure.

Figure 9 • **Launching SmartDebug Design Tools**



The **SmartDebug** window is displayed, as shown in the following figure.

Figure 10 • **SmartDebug Window Debug Options**



SmartDebug tools provide the following features and capabilities:

- Live probes: Two dedicated probes can be configured to observe a probe point, which is any output of a register. After selecting the probe points, the probe data can be sent to two dedicated pins (PROBE_A and PROBE_B). You can connect an oscilloscope to the probe pins and monitor the signals status.
- Active probes: Active probes allow dynamic asynchronous read and write to a flip-flop or probe point. This enables you to quickly observe the output of the logic internally, or to quickly experiment on how the logic is affected by writing to a probe point.
- SRAM and eNVM debug capabilities: SmartDebug includes test capabilities that can access SRAM and eNVM to assist with checking the flash memory (eNVM) content and reading and modifying the fabric SRAM content.

- Probe Insertion: Probe insertion is a post-layout process that enables you to insert probes into the design and brings signals out to the FPGA package pins to evaluate and debug the design.
- SerDes debug capabilities: SerDes debug capabilities make debugging high-speed serial designs simple. The SmartDebug JTAG interface extends access to configure, control, and observe SerDes operations and is accessible in every SerDes design. The designs are implemented using the Libero System Builder to incorporate the SERDESIF block enabling SerDes access from the SmartDebug toolset. The SerDes Debug window displays real-time system and lane status information. SerDes configurations are supported with Tcl scripting, allowing access to the entire SerDes register map for real-time customized tuning.

2.5 Debugging the Design

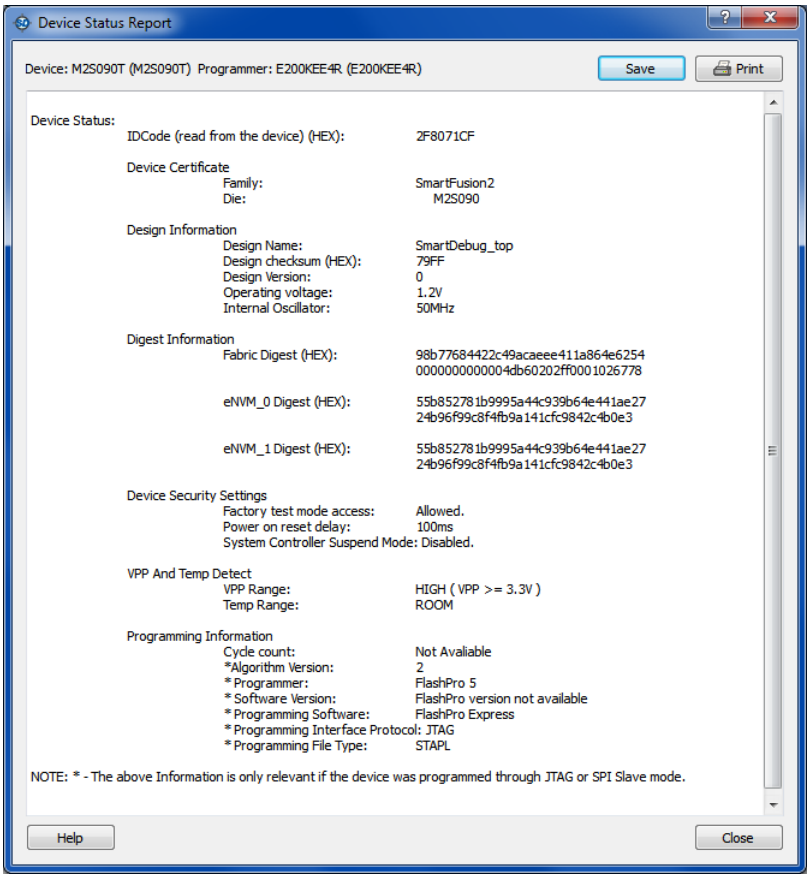
Debugging the device involves the following:

- View device status
- View flash memory (eNVM) content
- Debug FPGA array
- Debug SERDES

2.5.1 View Device Status

The View Device Status option provides the device status report. It summarizes the device information, programmer information, user information, factory serial number, and security information, if any are set. The following figure shows a sample of the device status information.

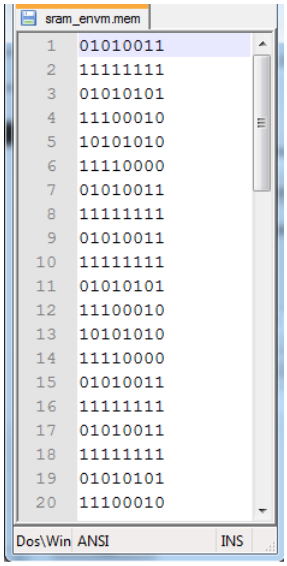
Figure 11 • **Device Status Report Sample**



2.5.2 View Flash Memory (eNVM) Content

The View Flash Memory Content option can be accessed from the **SmartDebug** window, as shown in Figure 10 on page 8. This option provides the capabilities to retrieve the eNVM content from the device using the Memories page of the System Builder under the SERDES_Debug block. To demonstrate how to retrieve the content of the eNVM, the data to be programmed into the eNVM is defined first. One way to perform this is by defining an eNVM data storage client using the eNVM configurator. The client can be stored into any page of the eNVM. eNVM page 64 is used here as an example. The following figure shows an excerpt of the data storage client content that was defined in the eNVM.

Figure 12 • **Memory File Content Saved into the eNVM**

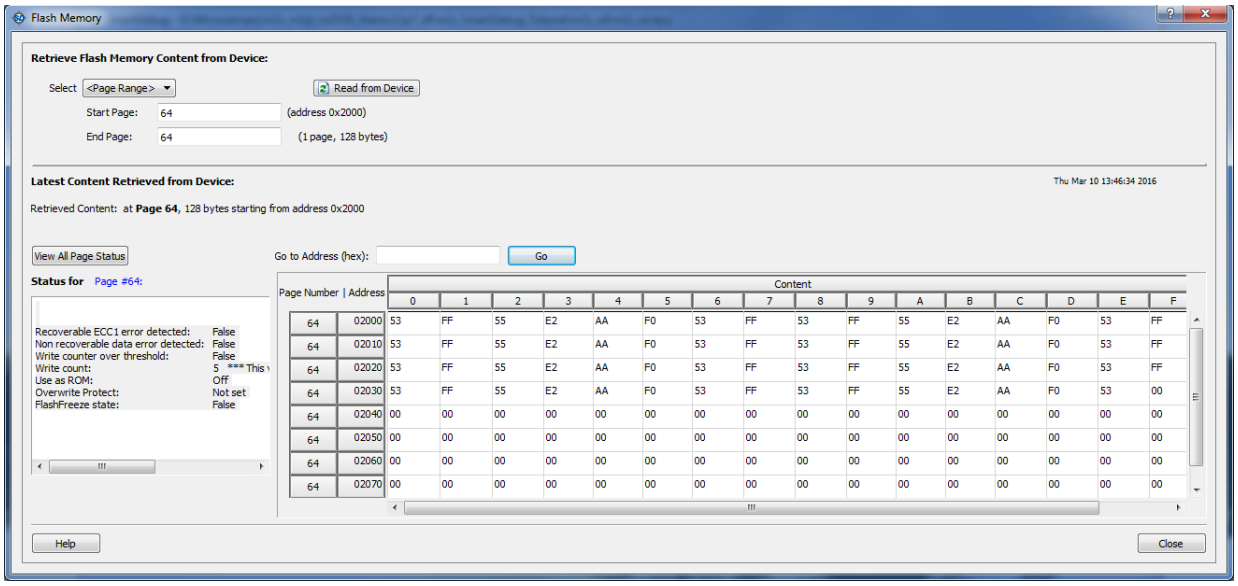


The content of eNVM is retrieved from the device, displayed, and is equivalent to what is shown in the preceding figure.

The following steps describe how the eNVM content can be read in real-time from the device:

1. On the **SmartDebug** window, click **View Flash Memory Content** (see Figure 10, page 8). The **Flash Memory** window is displayed, as shown in the following figure.
2. Enter the **Start Page** and **End Page** as **64.** Page 64 is used here as an example.
3. Click **Read from Device**. The content related to page 64 is displayed.

Figure 13 • **Flash Memory (eNVM) Content Read from the Device**



The eNVM window shows the content of the eNVM page selected. When you click the **View All Page Status** button, a dialog appears, and displays details such as:

- Number of images that have ECC errors.
- Number of overwrite threshold warnings.

After these details, the eNVM window shows the status of each page that you selected to view in the eNVM debug page.

2.5.3 Debug FPGA Array

SmartFusion2 and IGLOO2 devices have built-in probe points that enhance the ability to debug the logic within the device using the live probes and active probes features. The enhanced debug features implemented in the devices give access to any logic element and enable you to check the state of inputs and outputs in real-time, without re-layout of the design.

In addition to the ability to specify probe points, SmartDebug also provides the capability to read, modify, and write into the fabric SRAM block. The **Debug** window includes both **Hierarchical** and **Netlist** views that help you find test points. The **Hierarchical View** lets you view the instance-level hierarchy of the design programmed on the device. This view also lets you select the signals that are required to add to the **Live Probes**, **Active Probes**, and **Probe Insertion** tabs in the **Debug FPGA Array** dialog box.

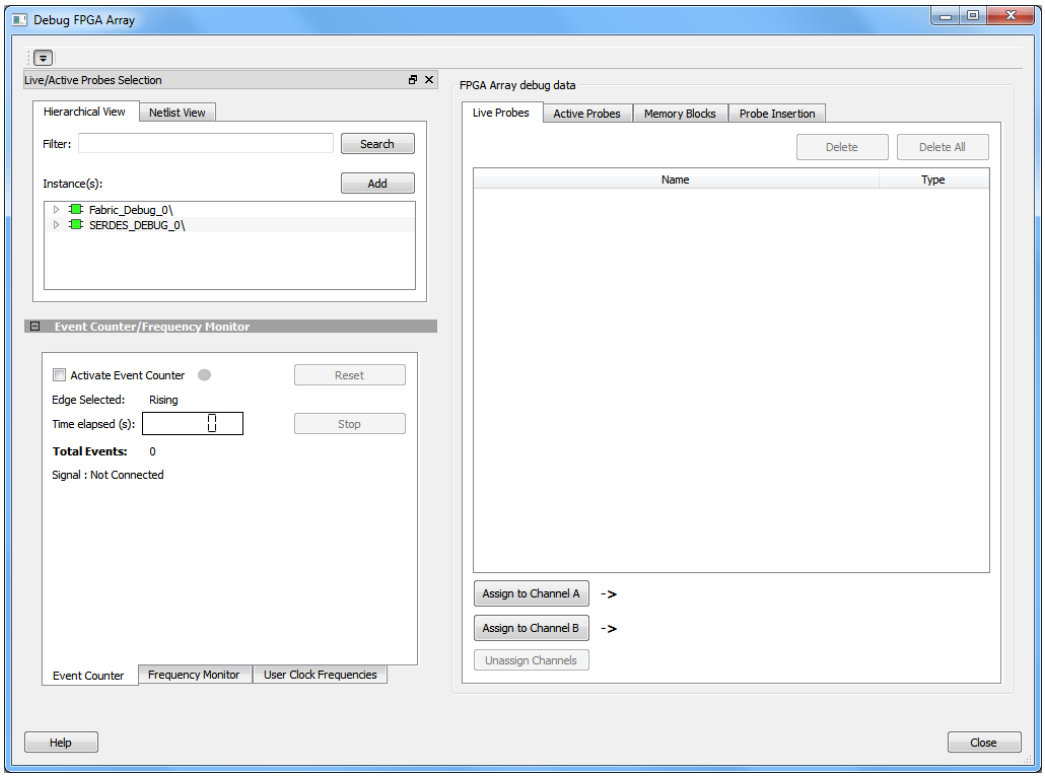
You can expand the hierarchy tree to see the lower level logic. Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree. The probe points are added by selecting any instance or the leaf level instance in the **Hierarchical View**. Adding an instance adds all the probe-able points available in the instance to live probes, active probes, and probe insertion.

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

This section demonstrates the abilities of setting live probes, active probes, and reading/writing from/to the fabric SRAM.

On the **SmartDebug** window, click **Debug FPGA Array**, as shown in Figure 10, page 8. The Debug FPGA Array window has a left and right pane, as shown in the following figure. The left pane has two tabs that allow you to toggle between the **Hierarchical View** and **Netlist View** debug points in the design. This information is read into SmartDebug from the Libero SoC design database. Event counter and frequency monitor are also provided in the left pane.

Figure 14 • **Debug FPGA Array Window**

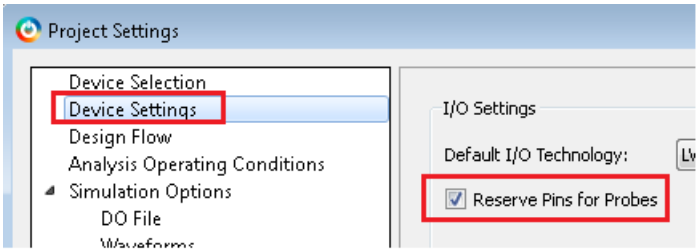


2.5.3.1 Specifying Live Probe Points in Libero

With live probe, two dedicated probes can be configured to observe a probe point, which is any output of a register. The probe data can be sent to the two dedicated probe pins (PROBE_A and PROBE_B). You can connect an oscilloscope to the probe pins and monitor the signals status. The probe points location can be changed without recompiling or reprogramming the design. The probes can capture data at a speed of up to 100 MHz.

The PROBE_A and PROBE_B pins are dedicated dual-purpose pins. These pins are regular I/Os, if not used by the live probes channels. These pins can be reserved for probing by selecting **Reserve Pins for Probes** in the **Project Settings** window, as shown in the following figure.

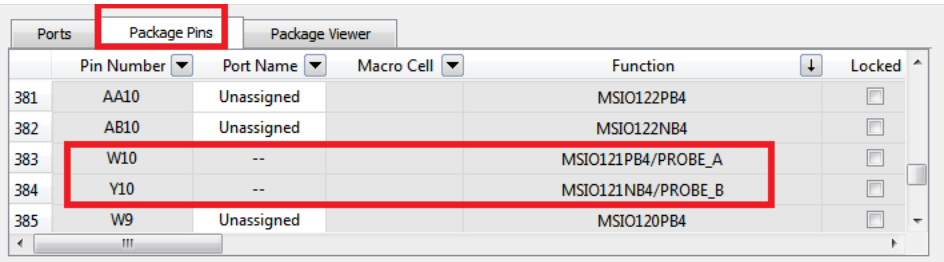
Figure 15 • **Reserving Probe Pin for Probes**



In addition, the probe pin on your package can be identified from the pin description document for that particular package. Another option is to check the **Function** column in the **Package Pins** tab of the **I/O Editor** in the Libero SoC software, as shown in the following figure.

Y10 and W10 are the two dedicated probe pins in M2GL010T and M2S090T in the 484 FBGA package, which can be used for probing, as shown in the following figure.

Figure 16 • **Identifying Probe Pins using Package Viewer Inside Libero I/O Editor**



Note: The probe pins, PROBE_A/PROBE_B, are not exposed and not accessible on the IGLOO2 Evaluation Kit Rev C board. These pins are accessible on the IGLOO2 Evaluation Kit board Rev D and SmartFusion2 M2S090TS Security Evaluation Kit Rev D on J29 and J30 jumpers.

The following figure shows an example of setting two probe points: coutA[23]:Q and coutA[24]:Q to be probed on Channel A and Channel B respectively.The **Live Probes** tab shows the probe point name and pin type (SRAM, Logic, or I/O). When a probe point is selected, it can be assigned to either Channel A (PROBE_A) or Channel B (PROBE_B) as follows:

1. Select the point to be probed, as shown in the following figure.
2. From the **Netlist View** tab, select the net to be probed**,** and click **Add**.
3. Click **Assign to Channel A** or **Assign to Channel B.**
4. Click **Close**.

A message is displayed in the log window of Libero SoC, showing the signals that are assigned to be probed, as follows:

Live probe has been set:

PROBE_A:

Channel A: Fabric_Debug_0/count_0/coutA[23]:Fabric_Debug_0/count_0/coutA[23]:Q

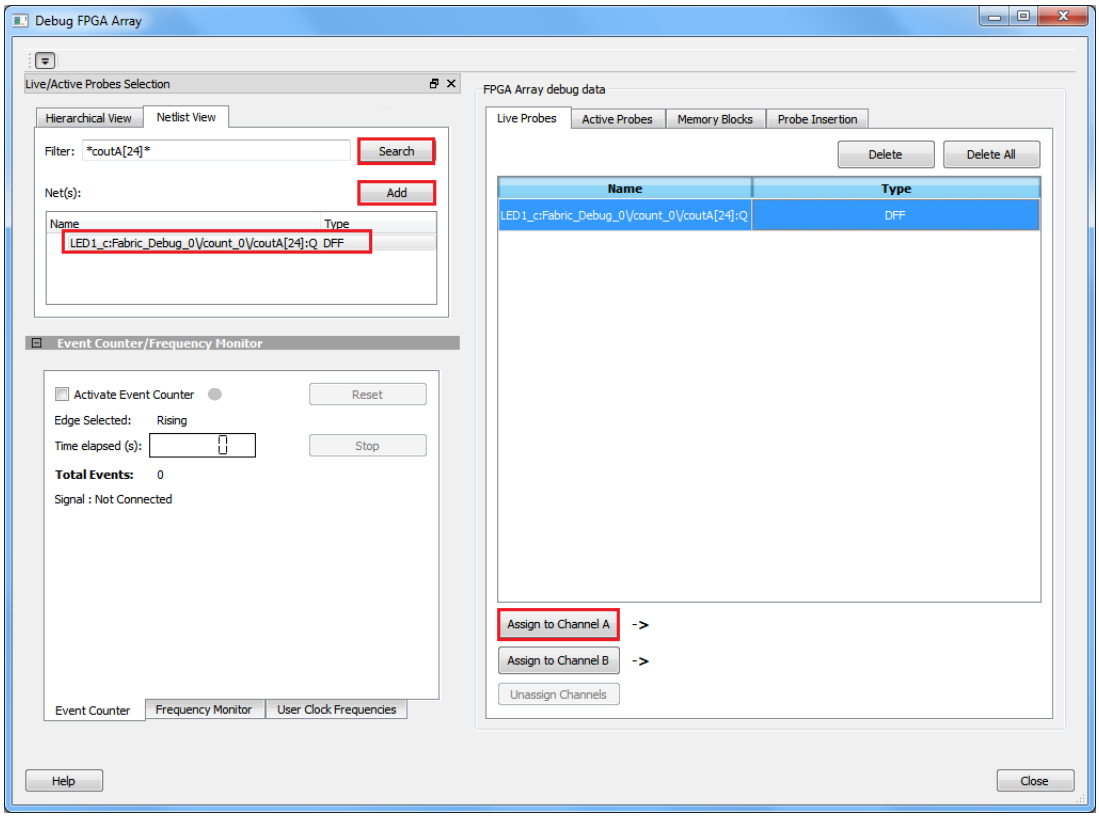
PROBE_B:

Channel B: LED1_c:Fabric_Debug_0/count_0/coutA[24]:Q.

After setting the channels, SmartDebug configures the Channel A and Channel B I/Os to monitor the desired probe points. On the SmartFusion and IGLOO2 Evaluation Kit Rev D boards, PROBE_A and PROBE_B pins are exposed on the J29 and J30 connectors. An oscilloscope can be connected to these probe points to monitor the signals that are assigned to be probed. The maximum number of simultaneous probes is two internal signals. A filter box is provided to filter out the net names.

Note: The active probes WRITE overwrites the settings of the live probe channels, if any.

Figure 17 • **Live Probes Channels Assignments**



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

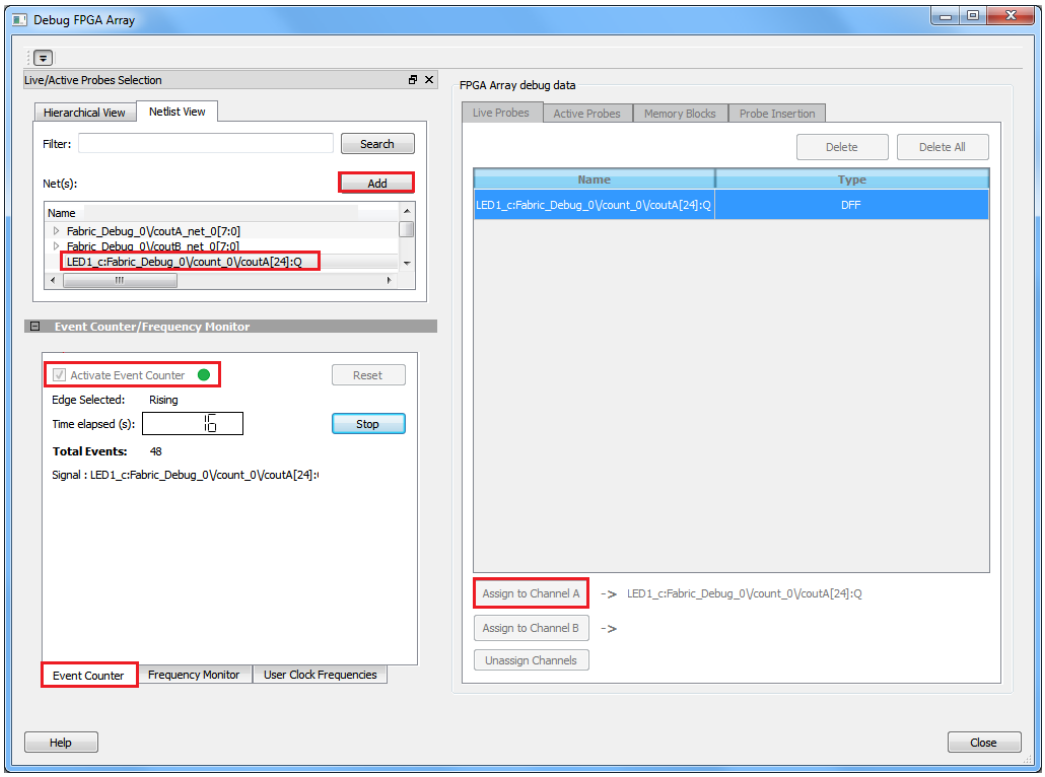
2.5.3.2 Using Event Counter for Live Probe Signals

The event counter counts the signals that are assigned to channel A using live probe. Live probe can track events from the MSS or the board. When the event counter is activated and a signal is assigned to channel A, the counter starts counting the rising edge transitions. The counter must be stopped to get the final signal transition count. During the count, you cannot assign another signal to channel A/channel B or go to any other tab on the window.

To use event counter for live probe signals, follow these steps:

1. From the **Netlist View** tab, select the net to be probed for event counter, and click **Add**, as shown in the following figure.
2. Select the **Activate Event Counter** check box.
3. Click **Assign to Channel A**.
   Total events and time elapsed are displayed.
4. To stop the event counter, click **Stop**.

Figure 18 • **Event Counter**



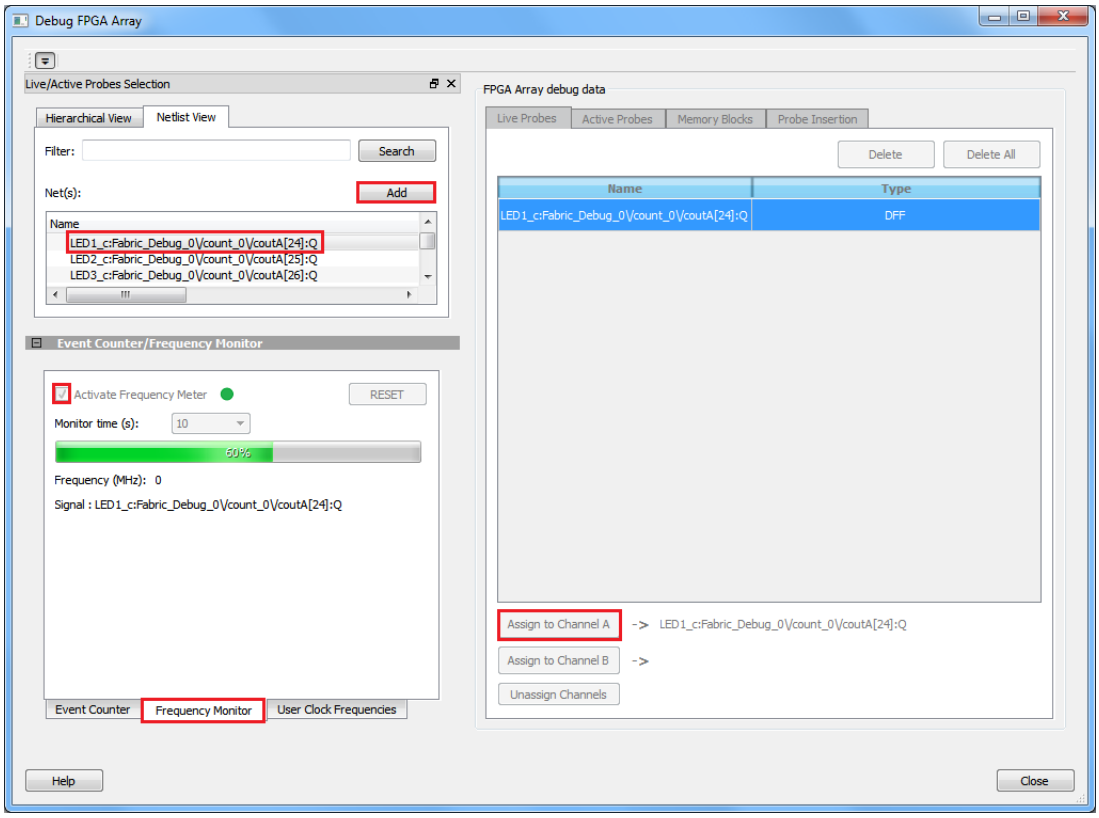
2.5.3.3 Using Frequency Monitor for Live Probe Signals

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 MHz.

To use frequency monitor for live probe signals, follow these steps:

1. From the **Netlist View** tab, select the net to be probed for frequency monitor, and click **Add**, as shown in the following figure.
2. Switch to **Frequency Monitor** tab, and select the **Activate Frequency Monitor** check box.
3. Click **Assign to Channel A**. Monitor time and progress are displayed.
4. To stop the frequency monitor, click **Stop**.

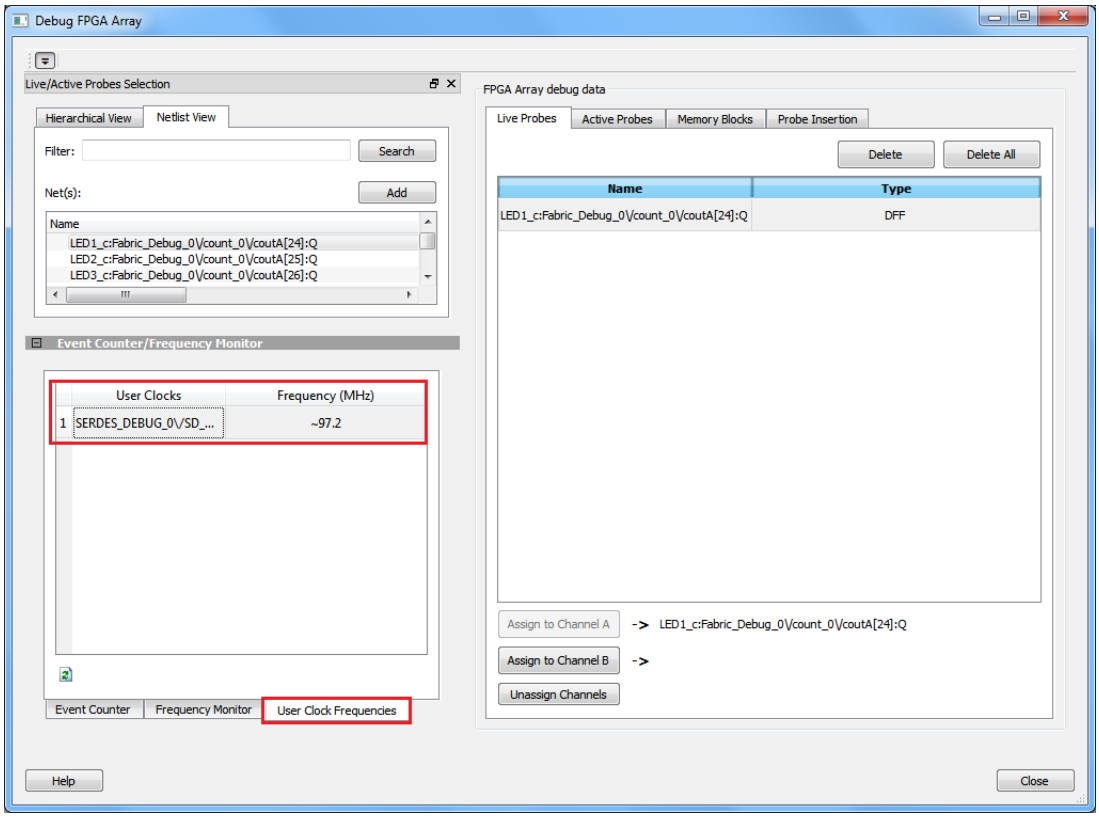
Figure 19 • **Frequency Monitor**



2.5.3.4 Monitoring User Clock Frequencies

The **User Clock Frequencies** tab shows the frequencies that were configured from the FCCC block (SERDES_DEBUG_0/SD_DEMO_0/CCC_0_GL0). The Refresh button can be used to recalculate frequencies if clocks are changed.

Figure 20 • **User Clock Frequencies**



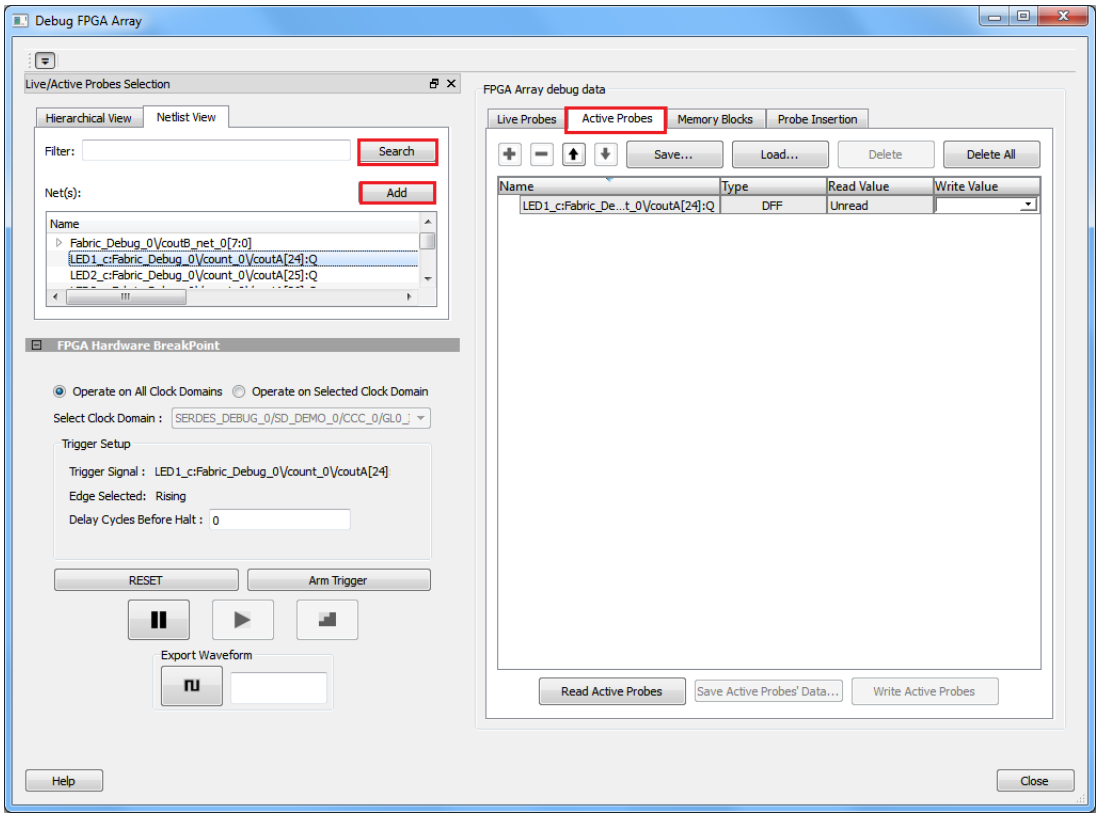
2.5.3.5 Active Probes

Active probe allows dynamic asynchronous read and write to a flip-flop or probe point. It enables you to observe the output of the logic internally or to experiment on how the logic is affected by writing to a probe point. The following steps describe how to select a specific set of probe pins by reading the current value and then writing different values.

2.5.3.5.1 Selecting Active Probes

1. On the **Debug FPGA Array** window, click the **Active Probes** tab.
2. Add search filter to find desired net to add probe, as shown in the following figure.

Figure 21 • **Selecting Active Probes From the Design**



A window that shows all the available probe points in the design opens. In this tutorial, the following points are monitored:

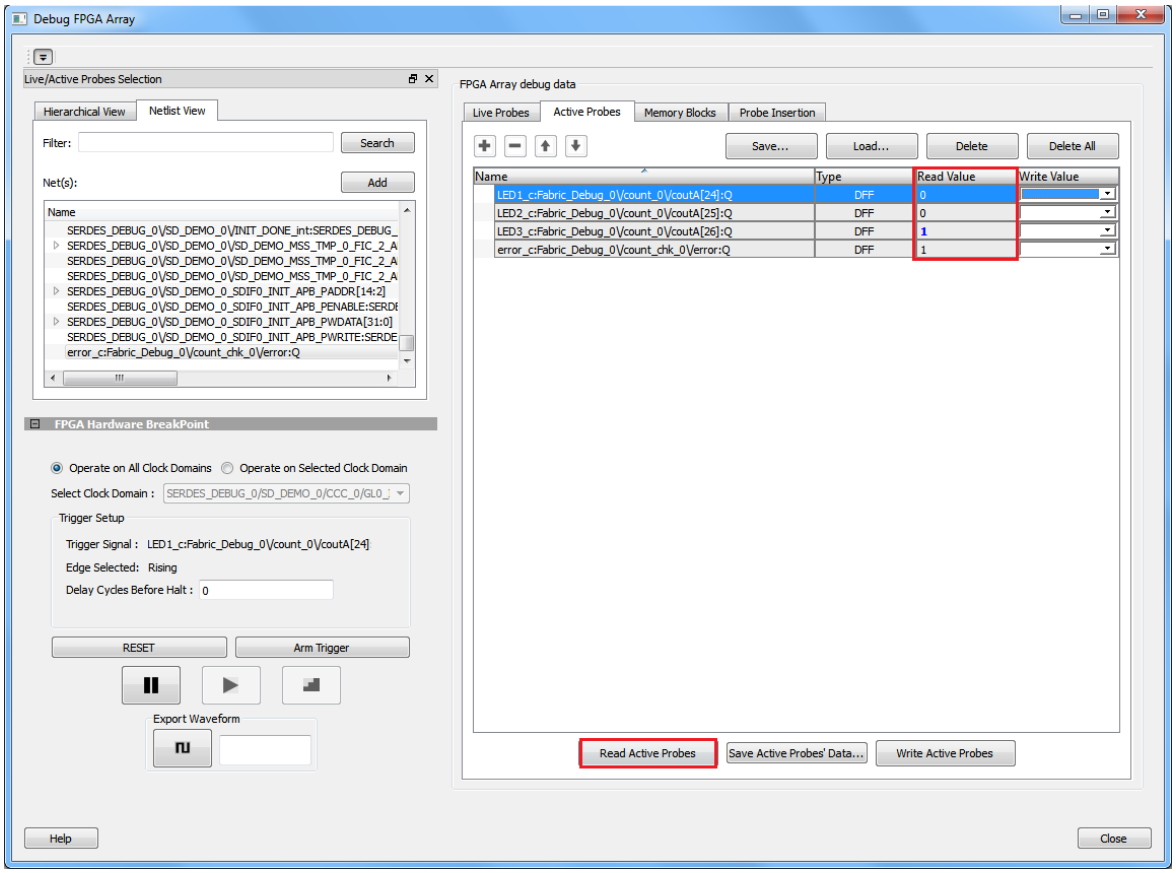
- Three bits of the counter output coutA—coutA[24]:Q, coutA[25]:Q, and coutA[26]:Q.
- The monitoring signal error, which is also connected to the LED (H5) on the board. If the LED is on, it indicates that the RAM count and the expected value mismatch.

Note: As active probe only deals with individual signals, the coutA bus segment is broken up into three separate probe lines.

3. Select the desired points and click **Add** to move to the **Selected Probe Points** window, and click **Read Active Probes,** as shown in the following figure.

Note: The coutA bus counts constantly. As a result, the value that you read may be different from the value shown. Also, the error signal must be high (LED H5 is off), which indicates that there are no errors in the counting pattern.

Figure 22 • **Selecting Desired Points to Read and Reading the Values**

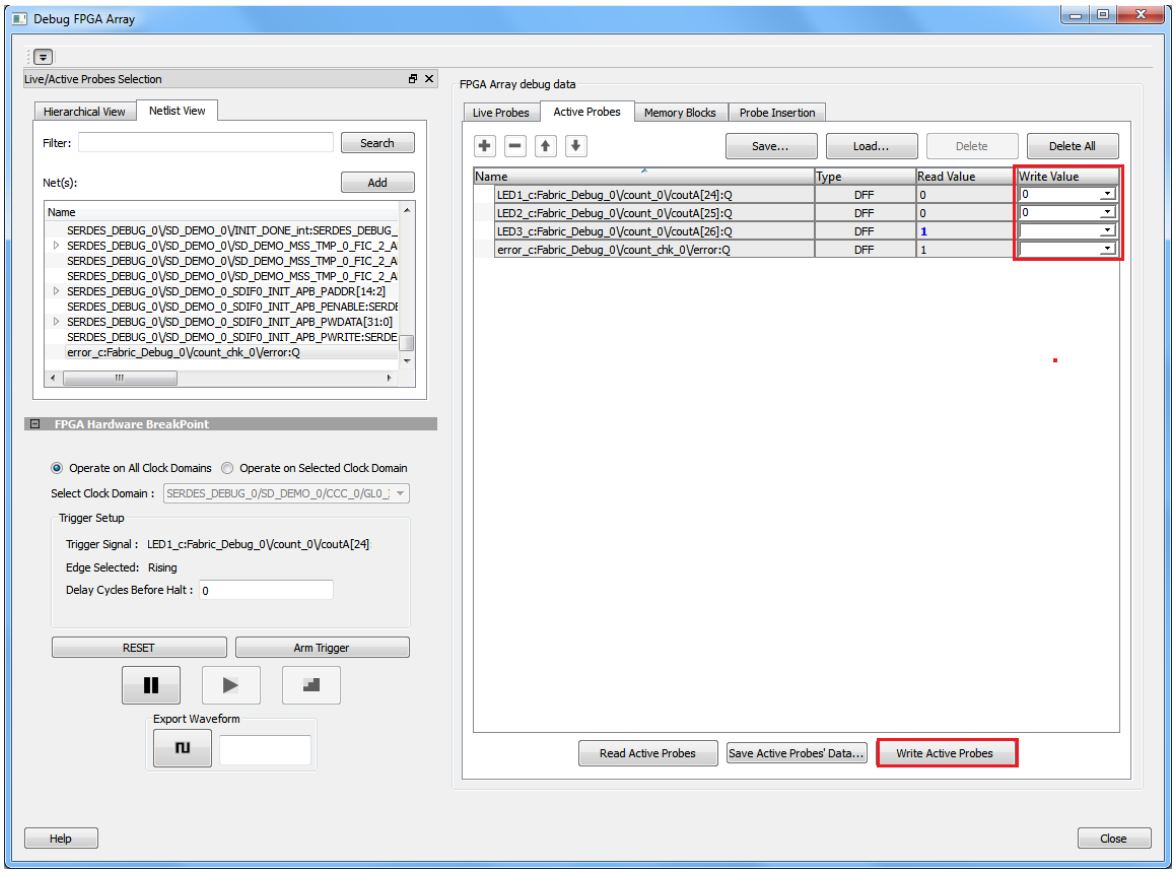


Note: The “+” and “-” icons expand or collapse the bus or group present in the **Active Probes** window. In the preceding figure, because there is no bus or group, these buttons are not enabled.

2.5.3.5.2 Writing Active Probes

The write operation is similar to the read operation. After specifying points you can define new write values that are applied to the device, when **Write Active Probes** is selected. The following figure shows the results of a first write of the design.

Figure 23 • **Active Probe Writing**



Note: To toggle the states between high and low, select a new **Write Value** from the drop-down menu, and click **Write Active Probe** to update the state.

Use the **Save** button to save a set of active probes. Use the **Load** button to load a saved active probe setup. This allows you to use various setups while debugging without needing to recreate the active probe settings.

The coutA bus counts constantly; therefore, the value that you read may be different from the value shown in the preceding figure.

Also, the error signal must be **High (LED H5 is OFF),** indicating that there are no errors in the counting pattern.

Note: Probe grouping feature is available with active probes. This feature is useful to manage large designs with many signals. This feature gathers multiple signals as a single entity. Probe nets with the same name are automatically grouped in a bus when they are added to the **Active Probes** tab. Create custom probe groups by manually selecting and adding probe nets of a different name into the group.

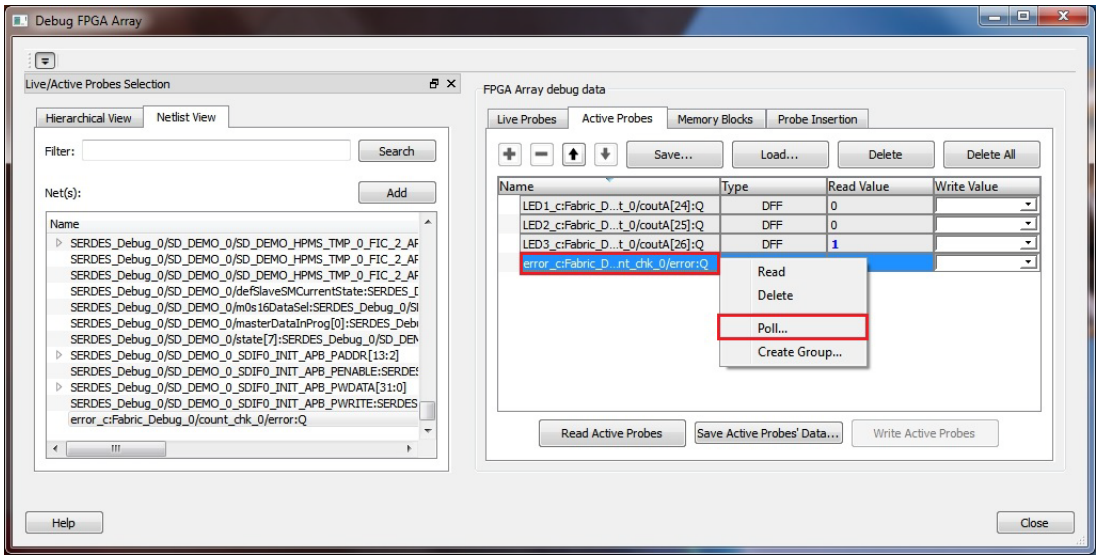
2.5.3.5.3 Pseudo Static Signal Polling

Pseudo static signal polling allows you to select active probe signals to continuously poll for a specified value. This feature is useful in probing signals that reach the intended state and stay in that state. The selected signal is polled once per second.

To use pseudo static signal polling, follow these steps:

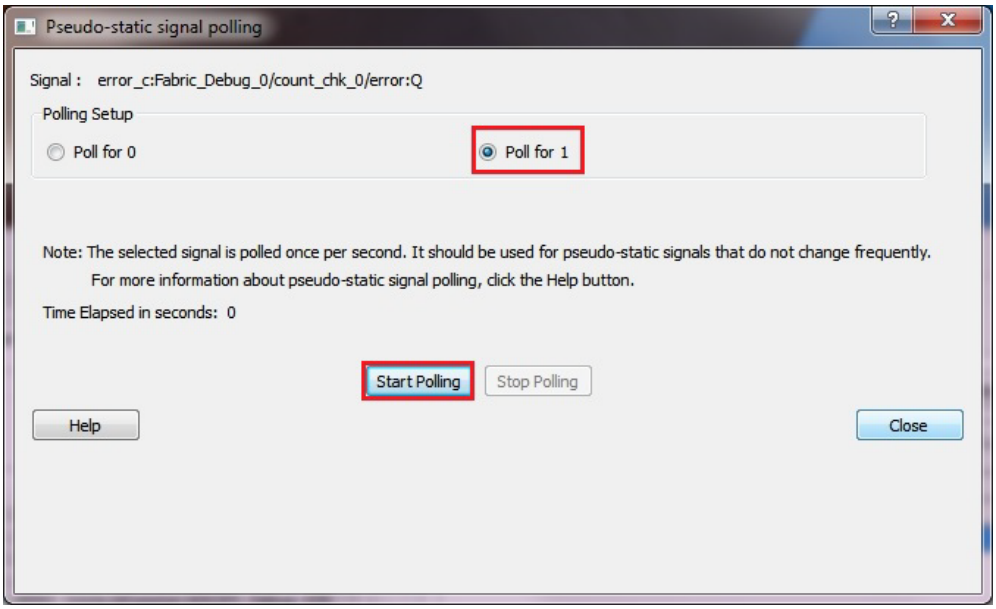
1. On the **Active Probes** tab, right-click **error_c:Fabric_Debug_0/count_chk_0/error:Q** and select **Poll**, as shown in the following figure.

Figure 24 • **Pseudo Static Signal Poll**



2. Select **Polling Setup** as **Poll for 1**, and click **Start Polling**.

Figure 25 • **Polling Setup Selection**



3. After the polled value is obtained, a status message is displayed, as shown in the following figure.

Figure 26 • **Status Message**

2.5.3.5.4 FPGA Hardware Breakpoint Auto Instantiation

FPGA hardware breakpoint (FHB) auto instantiation automatically instantiates an FHB instance per clock domain. The FHB instances gate the clock domain they are instantiated on. These instances can be used to halt the design through a live probe signal. The FHB controls in the SmartDebug window allow you to control the debugging cycle. The FHB controls have the following tabs:

- Event Counter—counts the signals that are assigned to Channel A or Channel B through the Live Probe feature.
- Frequency Monitor—calculates the frequency of any signal in the design that can be assigned to Live Probe Channel A or Channel B.
- User Clock Frequencies—shows the frequencies that are configured from the FCCC block.

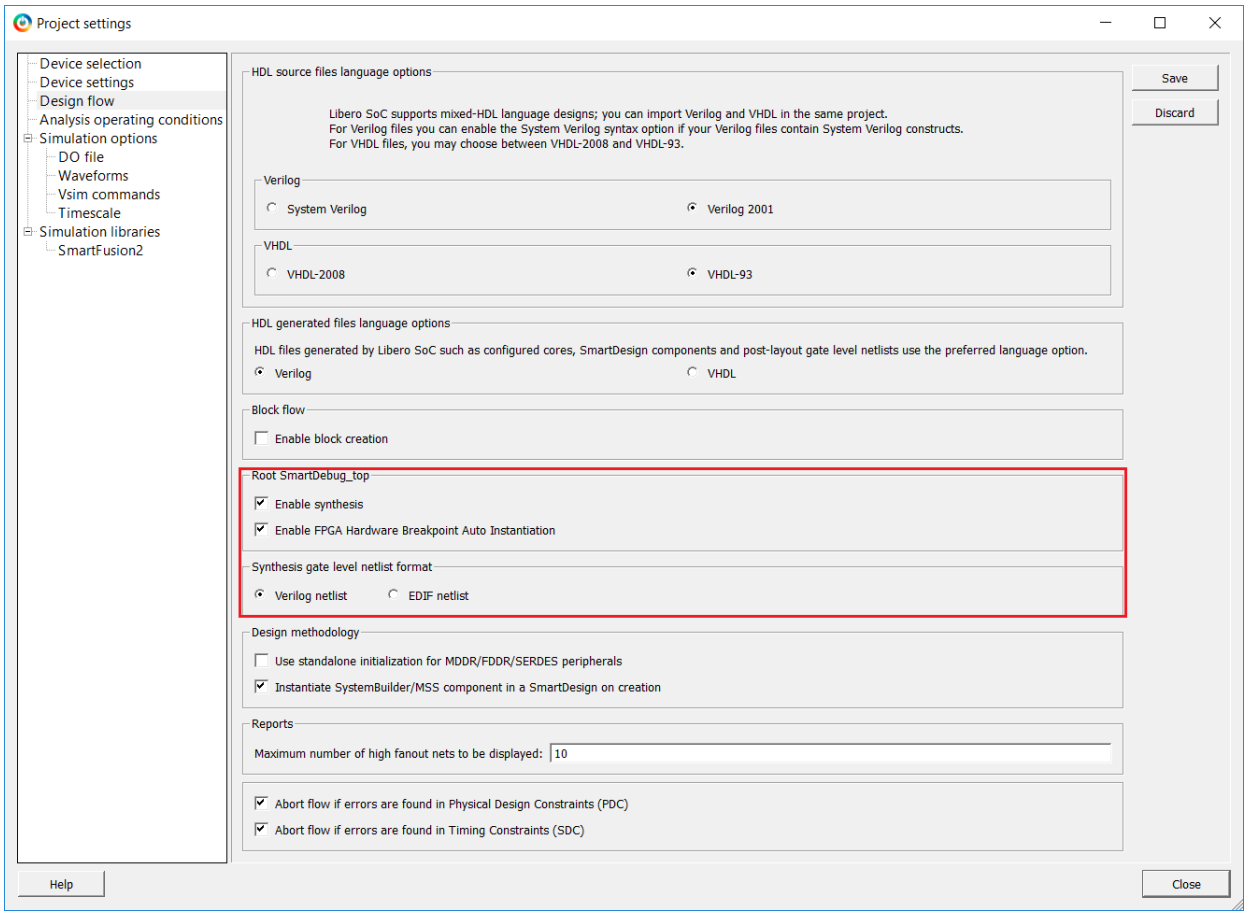
Note: FHB is available only for SmartFusion2 and IGLOO2 devices.

Enabling FHB

To enable FHB in Enhanced Constraint Flow, follow these steps:

1. Go to **Project** > **Project Settings** > **Design Flow**. The **Project settings** dialog box is displayed.
2. Click **Design flow** in the left pane. The design flow setting are displayed in the right pane.
3. Under **Root SmartDebug_top**, select the **Enable FPGA Hardware Breakpoint Auto Instantiation** check box.
4. Under **Synthesis gate level netlist format**, select **Verilog netlist**, as shown in the following figure.
5. Click Save and thenclick **Close** to close the dialog box.

Figure 27 • **Enable FHB**

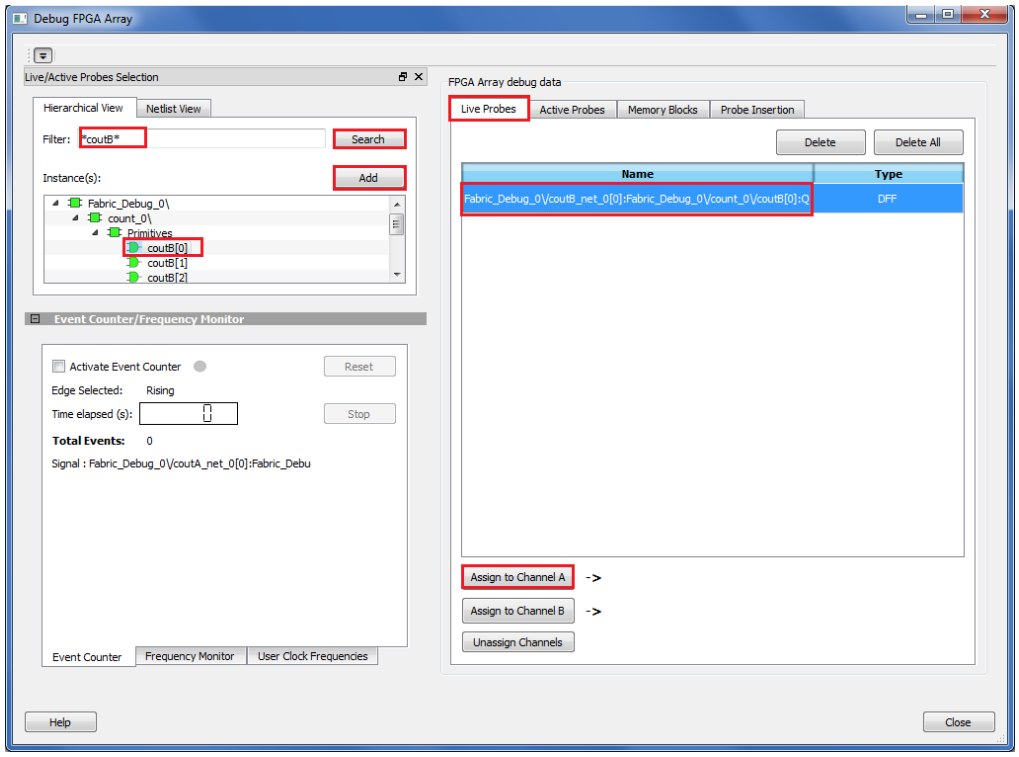


Using FHB

To use FHB auto instantiation, follow these steps:

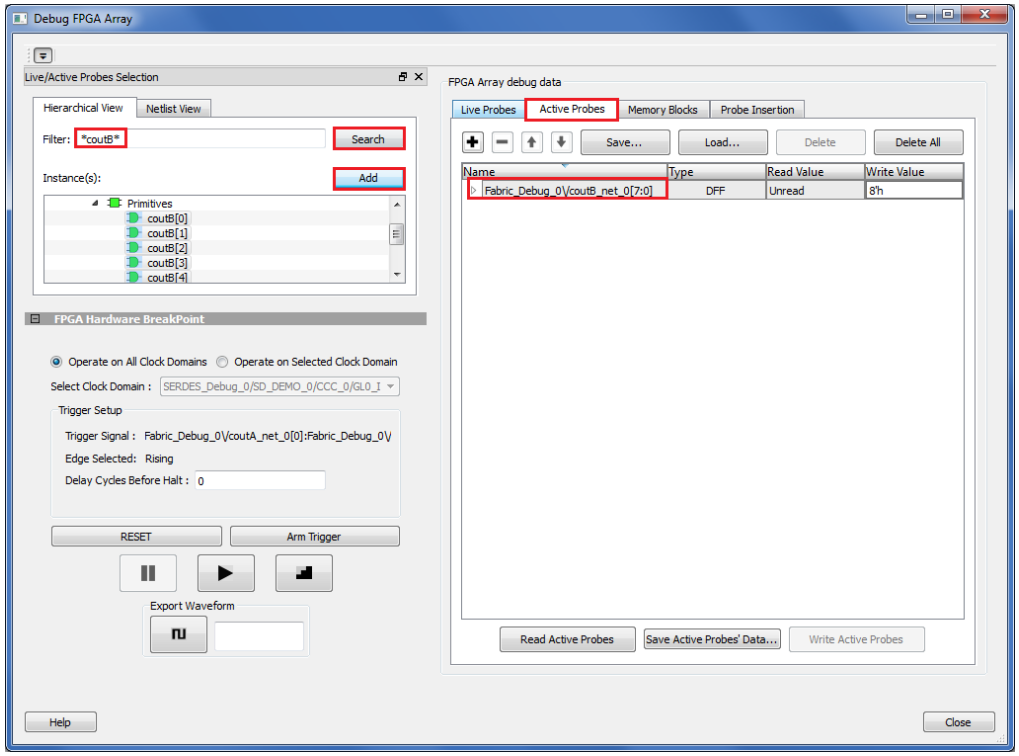
1. Go to the **Live Probes** tab, and search for *coutB*.
2. Add **coutB[0]** to live probes, as shown in the following figure. coutB[0] is used as the hardware break point trigger.
3. Click **Assign to Channel A**.

Figure 28 • **Add coutB[0] to Live Probes**



4. Switch to **Active Probes** tab, and search for *coutB*.
5. Add coutB[0] to coutB[7] to active probes, as shown in the following figure.

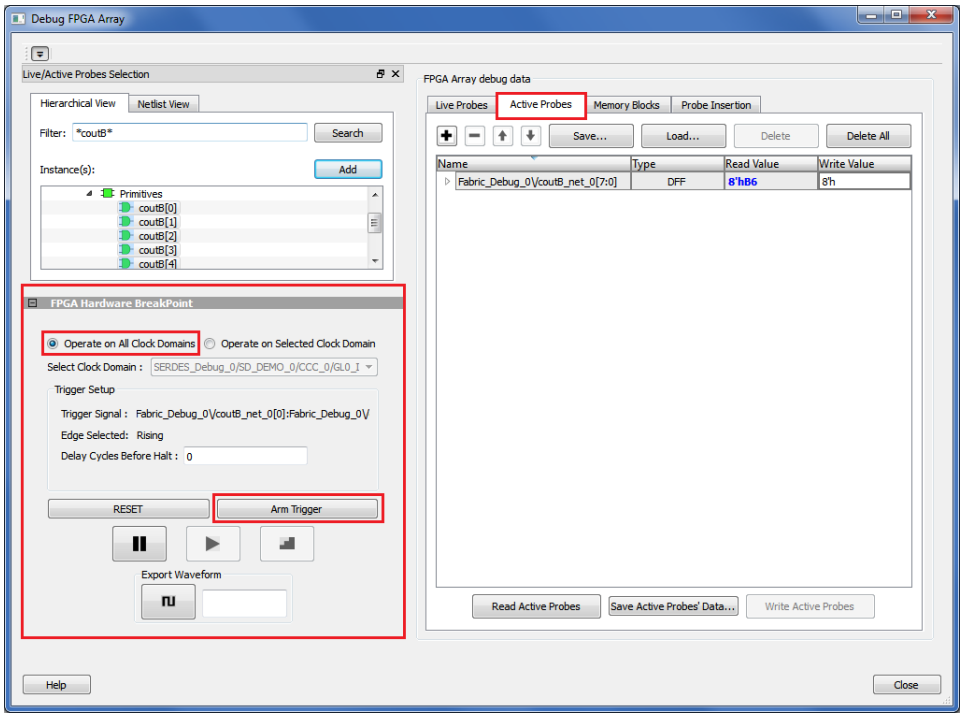
Figure 29 • **Add couB[0]–coutB[7] to Active Probes**



6. Select **Operate on All Clock Domains** or **Operate on Selected Clock Domain**, as shown in the following figure.

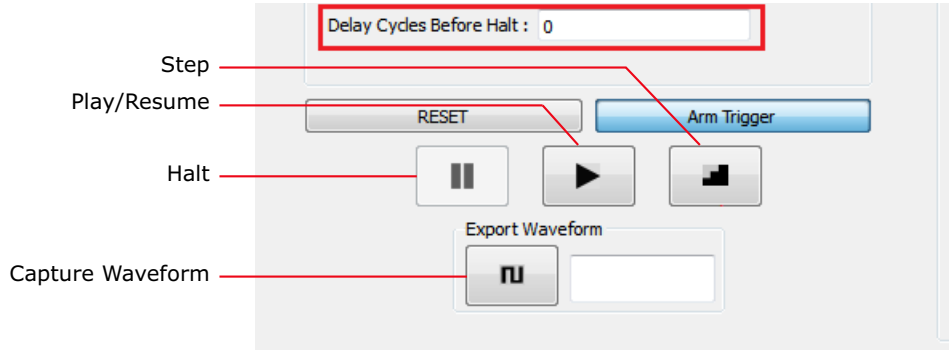
7. Click **Arm Trigger**. The counter halts on the next positive edge that occurs on the signal connected to Channel A (coutB[0]) in **Live Probes**.

Figure 30 • **Select Clock Domain**



8. Click **Reset**, to release the counter from halt state.

Figure 31 • **FHB Controls**



The following actions can done using FHB controls:

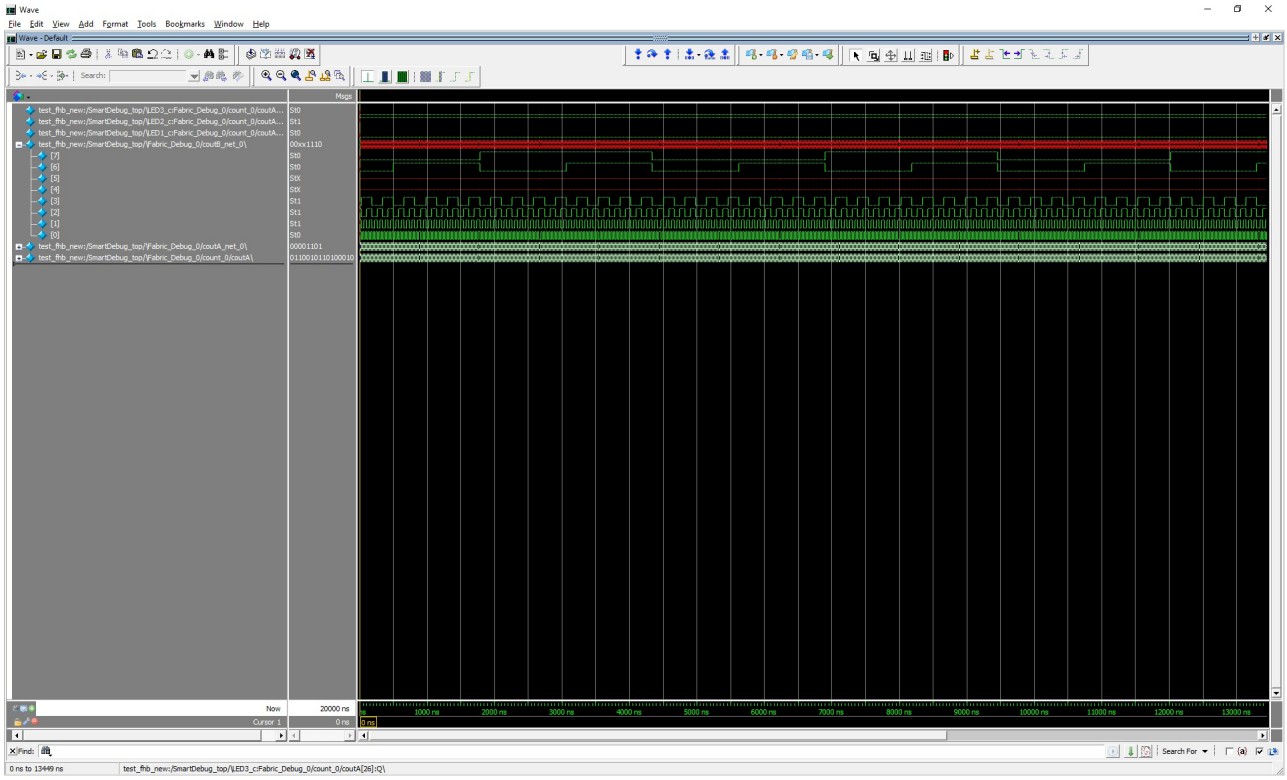
- Provide custom delay cycles before halt.
- Force a selected clock domain or all clock domains to halt without waiting for a trigger from a live probe signal, by clicking the **Halt** button.
- When the clock domain is in halted state (live probe halt or force halt), resume the clock domain by clicking the **Play/Resume** button.
- When the clock domain is in halted state (live probe halt or force halt), advance the clock domain by one clock cycle and hold the state of the clock domain by clicking the **Step** button.
- Save the waveform view of the selected active probes by specifying the number of clock cycles to capture in **Export Waveform** text box, and then clicking the **Capture Waveform** button. The waveform is saved as a `.vcd` file.
- View the saved waveforms by importing the `.vcd` file. The waveform file can be viewed in a waveform viewer that supports `.vcd` format.

Opening .VCD File in Modelsim

Active Probes in SmartDebug supports bus names that are added as slice of the bus in the design. For example, count_out_c[7:5,3:0], where count_out_c[4] is not included. This naming style is Microsemi-specific and is not supported in generic VCD viewers. To support generic VCD viewers, the probe name has been updated to count_out_c[7:0] where the 4th bit, count_out_c[4], is represented as x in the generated *.vcd file during FHB step function.

Note: Live Probe triggering occurs on the Positive Edge only.

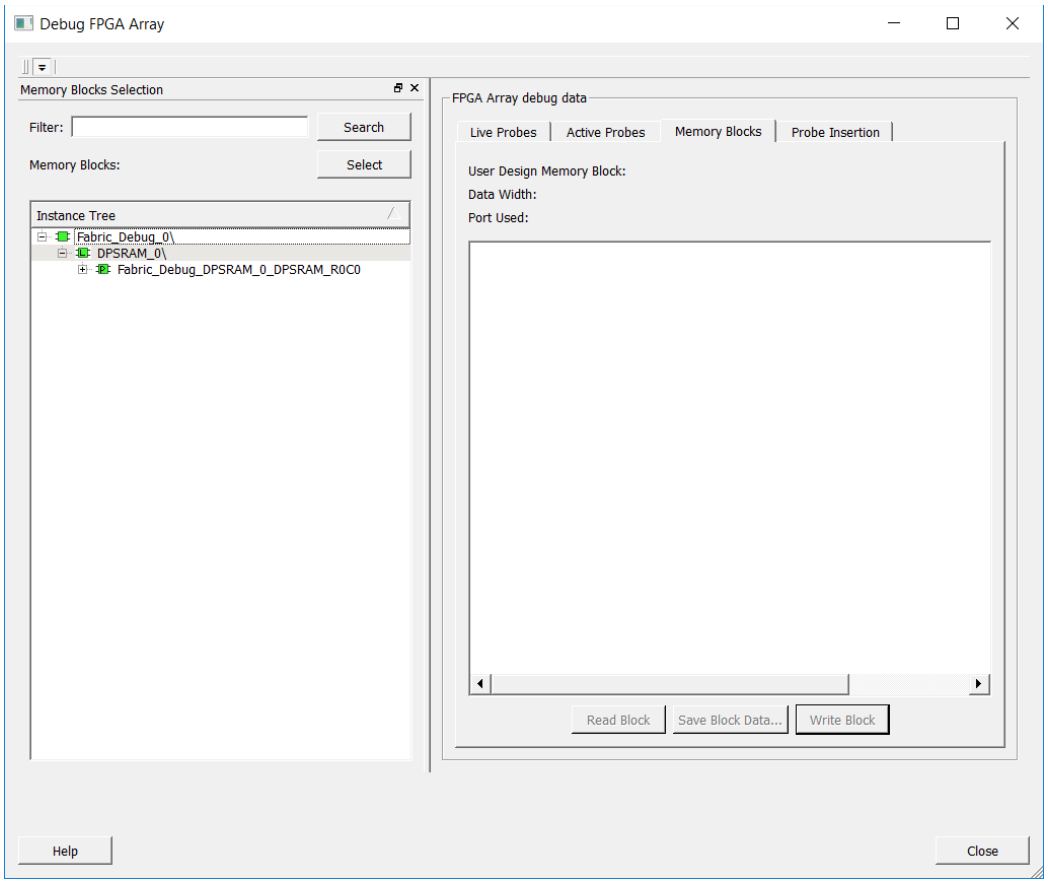
Figure 32 • **Opening a Non-Continuous Signal in Modelsim**



2.5.3.6 Debug Fabric SRAM Memory

To view the content of the large SRAM in this design, click the **Memory Blocks** tab, as shown in the following figure.The Memory Blocks sort options are introduced on the left pane of the window. The sorting can be done in ascending or descending order. By default, the memory blocks are listed in ascending order.

Figure 33 • **Memory Blocks Tab**

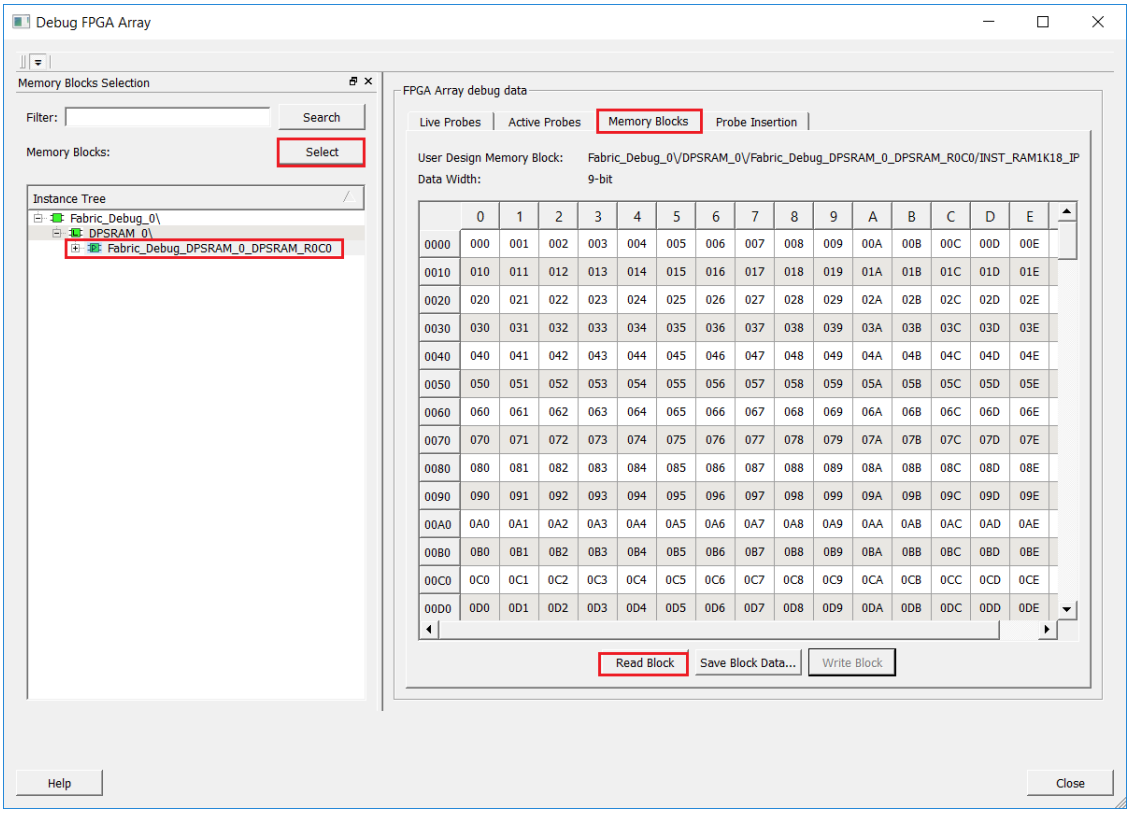


This design contains a single large SRAM, named DPSRAM_0, and it is the only one available on the left side window that shows the list of all the available RAM blocks in your design. If your design has more RAM blocks, the RAM blocks are shown on the left panel window. Select the memory block from the left panel, and double-click or click **Select** to make it the **Current Memory Block** to debug. After selecting the block, click **Read Block**.

Note: Physical blocks have P written on them, and logical blocks have L written on them.

The contents of the DPSRAM_0 is displayed, as shown in the following figure. See the counting pattern that is loaded into the RAM.

Figure 34 • **DPSRAM_0 Contents**

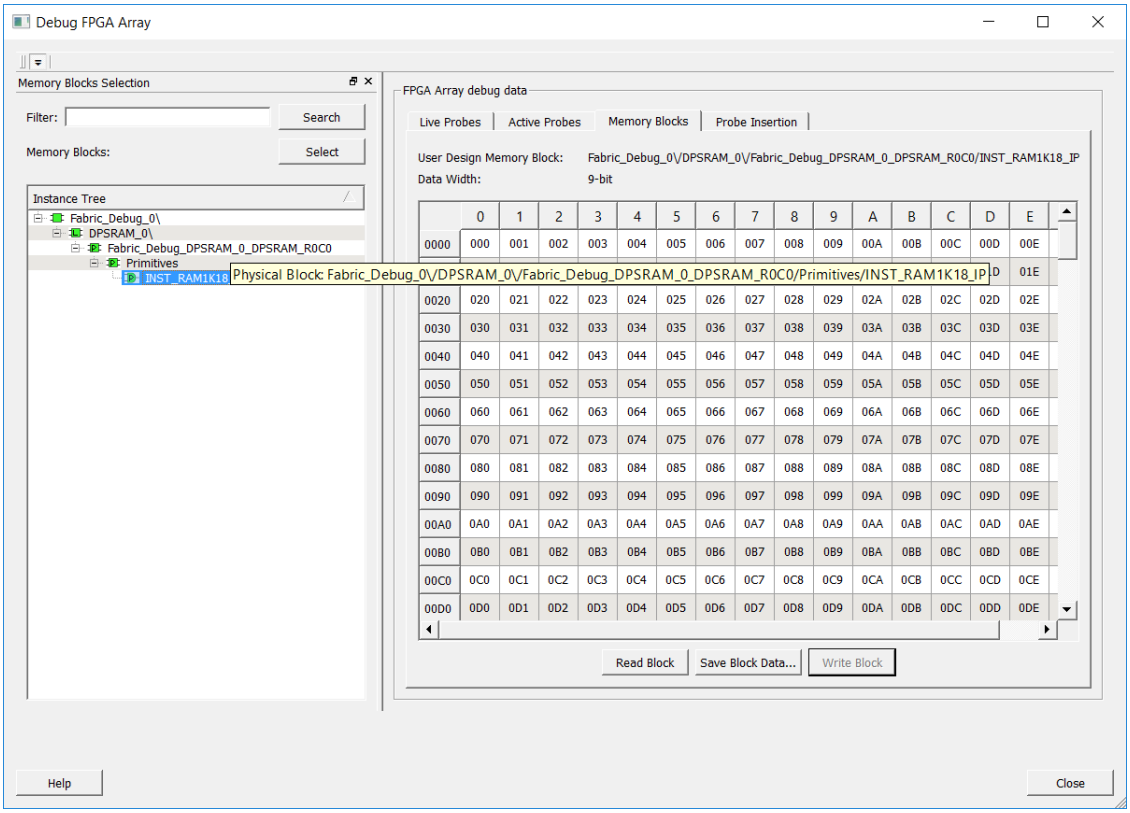


Note: The left pane displays all the memory blocks contained in the design. The right pane only displays the current memory block that is selected.

2.5.3.6.1 Displaying Tooltip

When you hover the memory block on the left pane, a tooltip is displayed about the memory block path. The following figure shows a tooltip about the selected physical memory on the left pane.

Figure 35 • **Displaying Tooltip for Memory Blocks**



2.5.3.6.2 Writing to Fabric SRAM Blocks

The design reads the contents of the DPSRAM and compares it with a synchronized counter in the checker, which looks for errors. If the content of the DPSRAM is modified, it breaks the count pattern and causes an error in the checker.

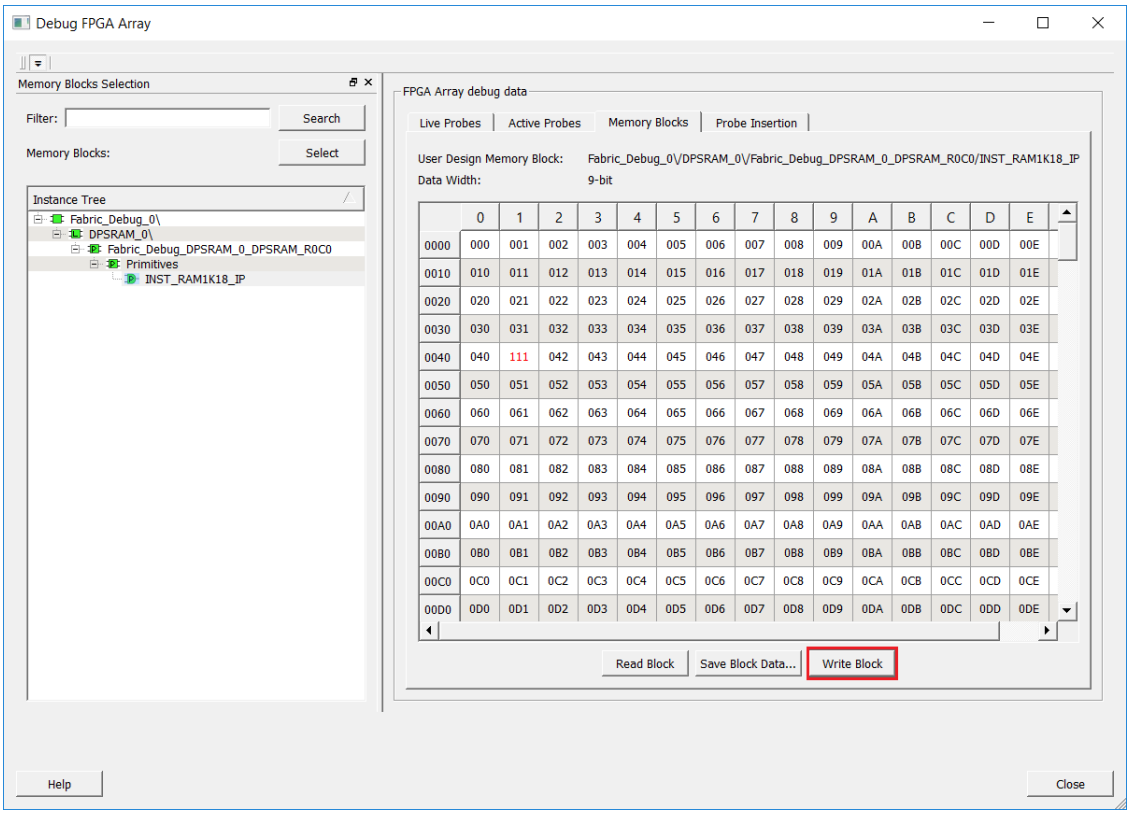
The following steps describe how to modify the RAM content and force an error:

1. Read the memory content, as shown in the preceding figure.
2. Select an entry and double-click. Each entry is 9-bits wide.
3. Modify the current value to break the count pattern. An example is shown in the following figure.

Note: The counter writes to the SRAM constantly. To prevent the overwrite of the changes that are forced into the SRAM, the writing is stopped by forcing A_WEN LOW through switch 2 (SW2). This drives a SELECT of a MUX that selects between high and low inputs. When SW2 is pressed, A_WEN becomes low, which prevents any write from the counter to the SRAM block.

4. Press and hold SW2, and click **Write Block** to write the modified value to the SRAM.
5. The error LED(H5) light turns on, indicating an error in the counting pattern.
6. Release SW2 to resume the write operation from the counter to the SRAM.
   This overwrites the error that was injected into the SRAM. The content of the SRAM can be rechecked by clicking **Read Block**.

Figure 36 • **Modifying DPSRAM Contents**



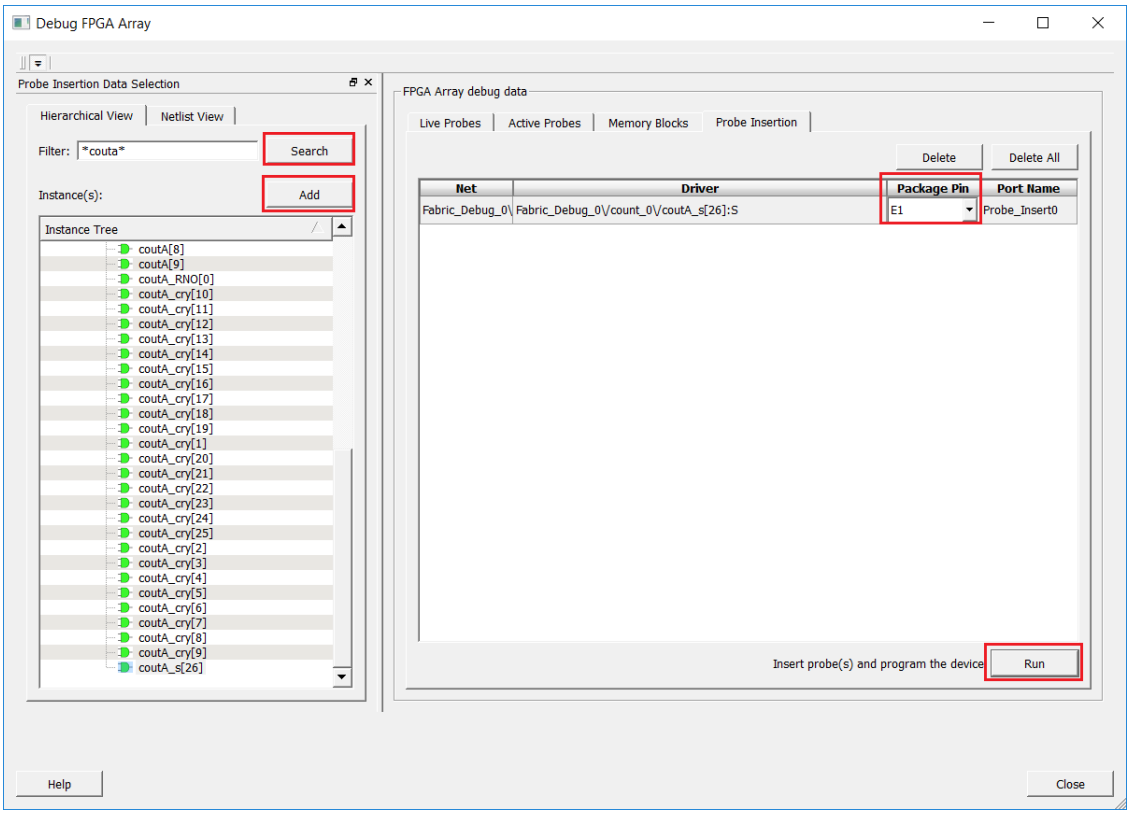
2.5.3.7 Probe Insertion

Probe insertion is a post-layout process that enables you to insert probes into the design and bring signals out to the FPGA package pins to evaluate and debug the design. Probe insertion enables you to select internal nets anywhere in the design, connect those nets to unused pins, and then run the layout incrementally to manage the physical connection to the pin. Nets are selected and assigned using the SmartDebug probe insertion feature. For more information, see *FPGA On-Chip Debug Tools.*

The following steps describe how to probe a net:

1. On the **Debug FPGA Array** window, click the **Probe Insertion** tab and click **Add Probe**, as shown in the following figure.
2. In the **Filter** field, enter the ***couta*** signal, as shown in the following figure, and click **Search**. Select signal, **Fabric_Debug_0/count_0/coutA_s[24]**. The goal is to add a probe by routing the counter bit 24 signal out into the E1 LED and check if it toggles.
3. Click **Add.**
4. In the **Probe Insertion** window, assign **Package Pin - E1**, as shown in the following figure.

Figure 37 • **Assigning Package Pin and Running the Flow**



5. Click **Run**.
   The place and route tool is run incrementally in the background re-routing the coutA_s[24] signal to package pin E1, which is the LED on the board. The LED E1 on the board starts toggling when the incremental place and route is completed, which indicates that the counter output bit[24] is routed to the E1 package pin.
6. Click **Close** to exit the **Debug FPGA Array** window.

Note: The **Probe Insertion** tab is not available with standalone SmartDebug.

2.5.4 Debug SERDES

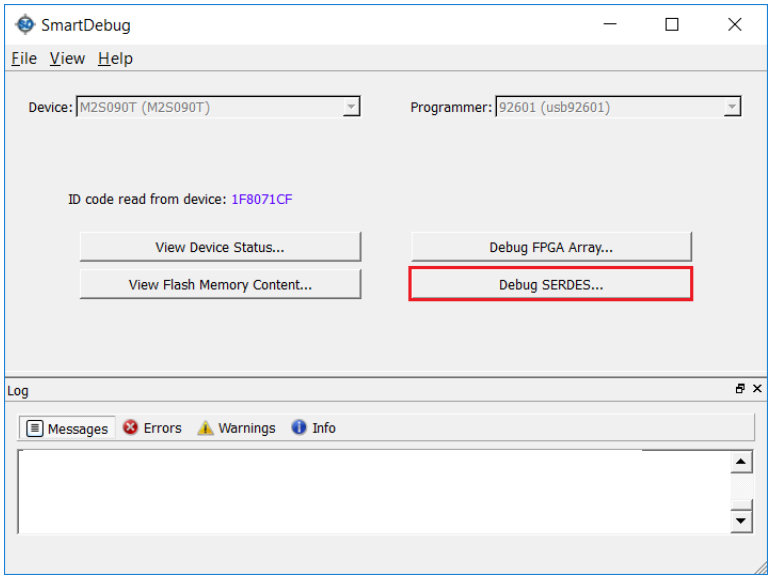
This SmartDebug SerDes tutorial helps the FPGA and board designers to perform SerDes real-time signal integrity testing and tuning in a system that:

- Provides real-time access to SERDESIF block control and status registers.
- Provides testing functions with pseudo-random binary sequence (PRBS) pattern generators and checkers.
- Runs link tests with various loop back options.
- Provides overview for tuning many combinations of physical medium attachment (PMA) analog settings to find the optimal set for a SerDes channel.

The following steps describe how to perform SerDes debug:

1. On the **SmartDebug** window, click **Debug SERDES**, as shown in the following figure.

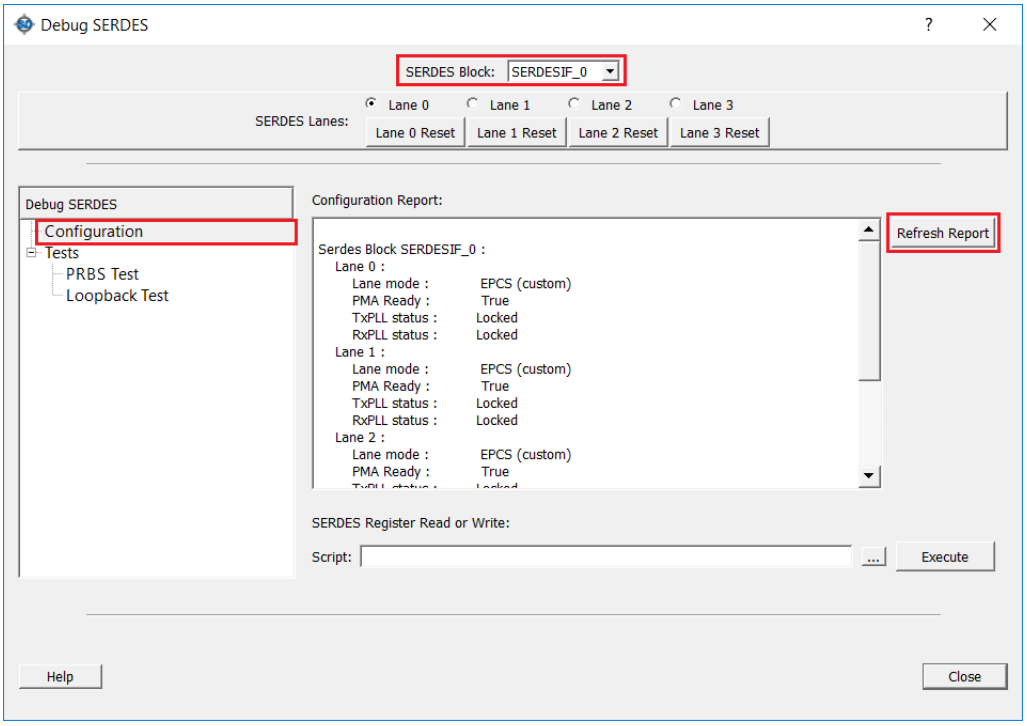
Figure 38 • **Debug SerDes Operation Selection**



The **Configuration** tab is displayed as shown in the following figure. The **Configuration** tab auto-identifies and populates SERDESIF and the lanes used in the design. The status of each lane and the programmed lane mode are displayed. This example demonstrates the use of SERDESIF_0 block and the lock status of TxPLL and RxCDR.

2. Click **Refresh Report,** to update the data.

Figure 39 • **SerDes Configuration Tab**

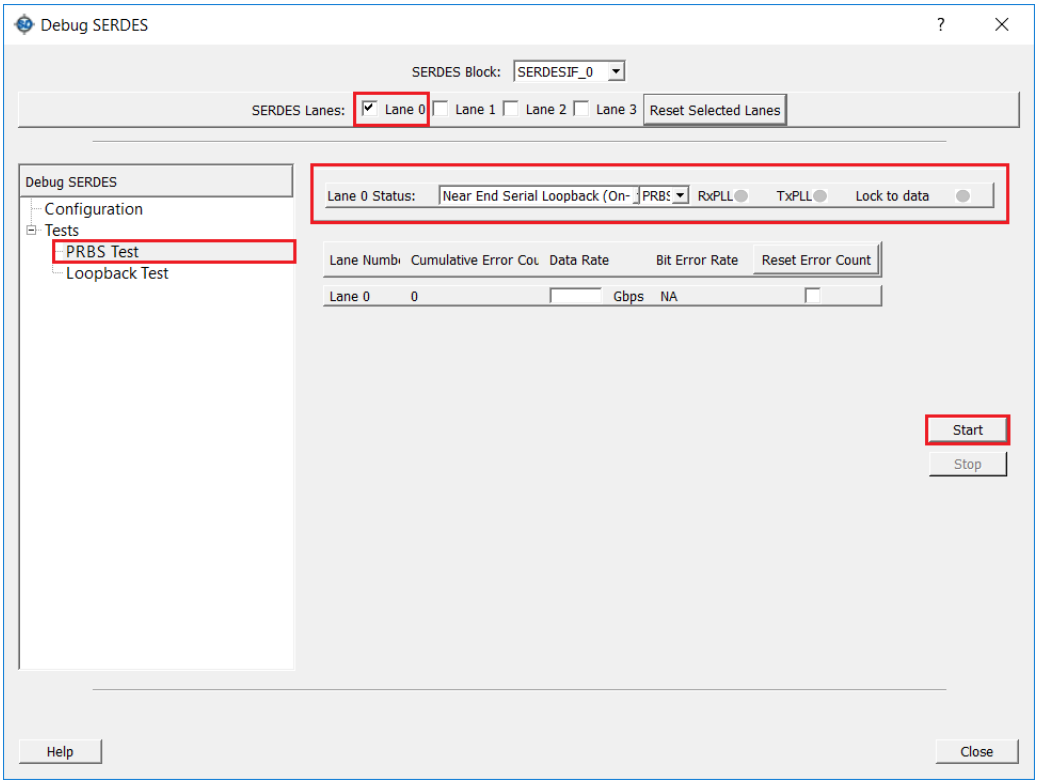


3. The **PRBS Test** tab provides several capabilities for each lane of the SerDes block. Based on the selected SerDes lane, information is provided for each channel. For example, select **Lane 0**, **Near-end Serial Loopback**, **PRBS7**, and click **Start**, as shown in the following figure.

This test generates and checks PRBS7 data without going off-chip.

Note: Lane 0 is the PCIe lane. This lane is connected to the PCIe edge fingers of the evaluation board.

Figure 40 • **SerDes Test Tab—Lane 0**



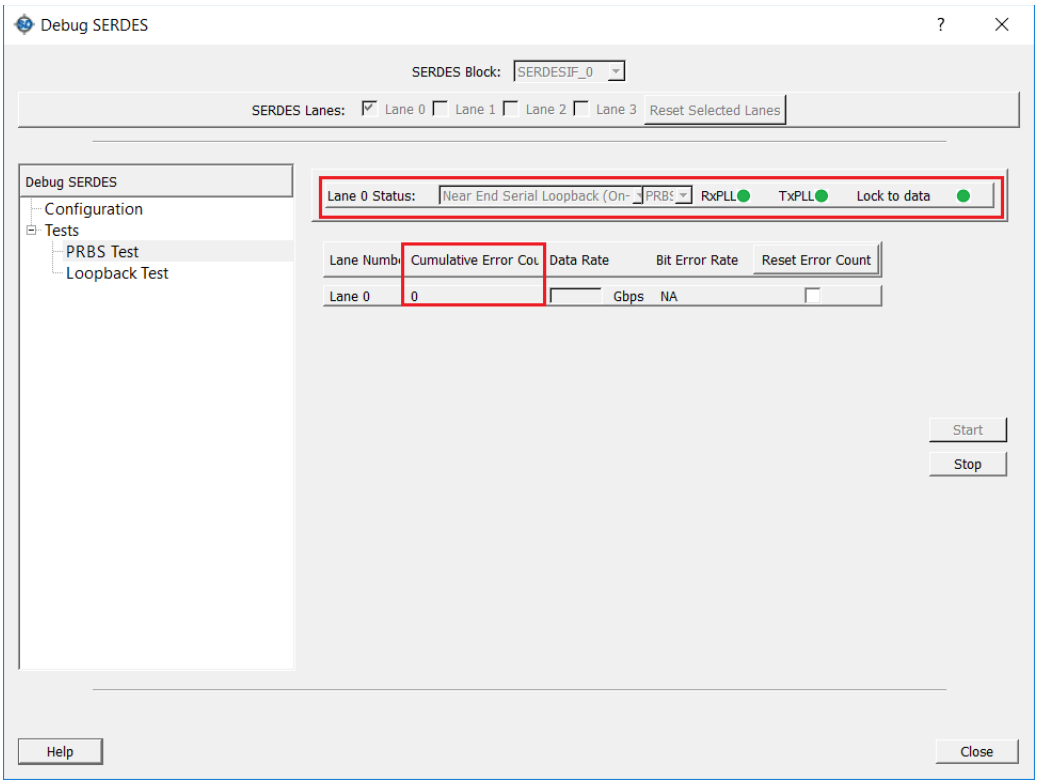
In this example setup, the data stream is expected to have zero errors as the datapath does not go off-chip while using the near-end serial loopback. The green LEDs indicate the lock status of TxPLL and RxCDR for the selected lane, as shown in the following figure.

The data rate value is entered based on the actual targeted data rate of the design. This rate is entered in Gbps and is used to derive the bit-error rate (BER). When the data rate is entered, the bit-error rate is calculated based on the following formula:

BER = (1 + Error count) / (data rate * seconds)

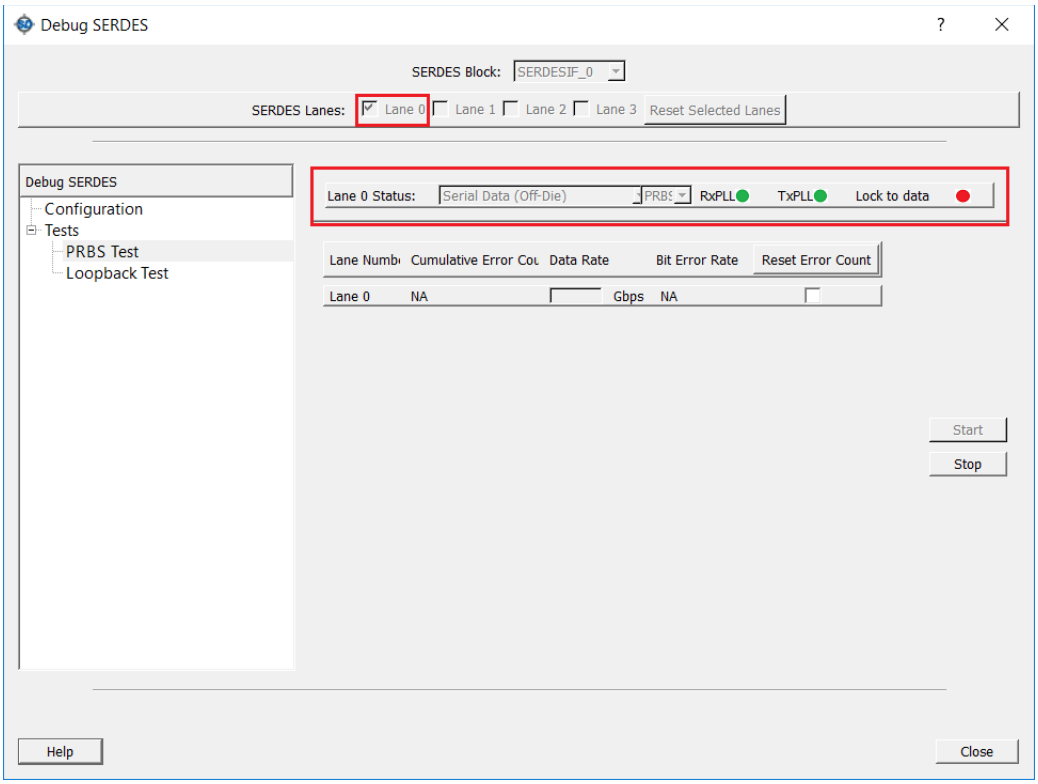
If the data rate is left blank, BER is not calculated. The **Reset** button clears the cumulative error count and the bit-error rate.

Figure 41 • **SerDes Link Status—Lane 0**



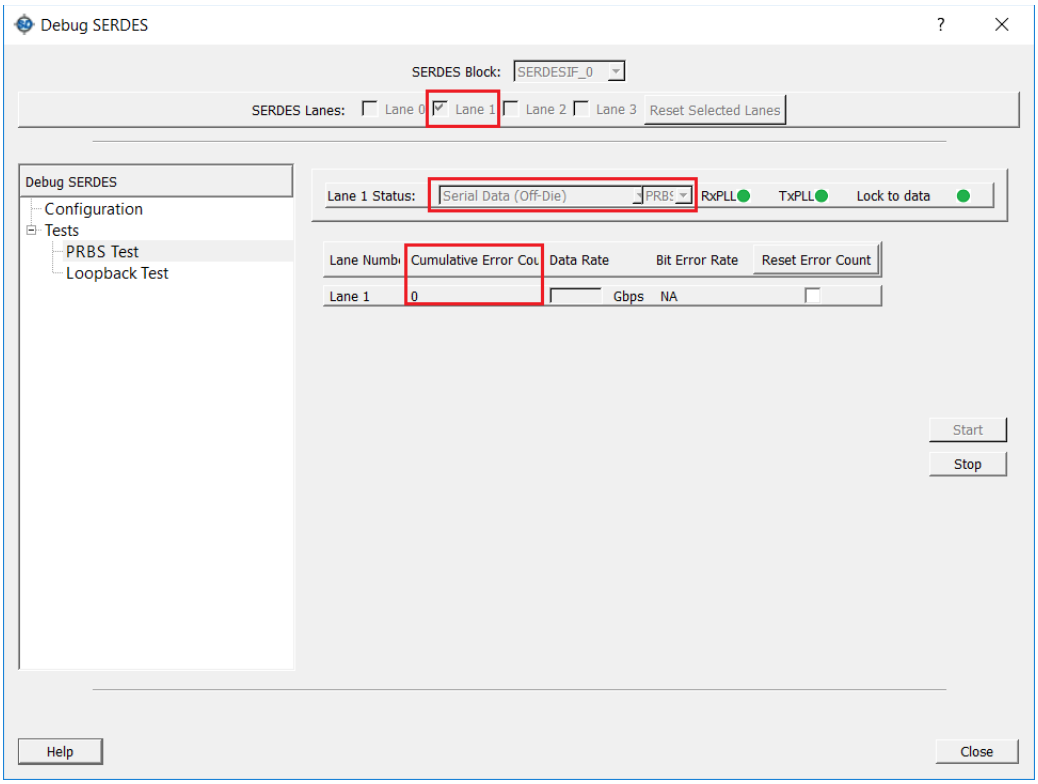
4. Click **Stop** and select **Serial Data (Off-Die)**. Click **Start** and observe that the **Lock to data** status indicator is red. This indicates that the data is no longer looped between Tx and Rx, and Lane 0 is not looped together on the PCB, as shown in the following figure.

Figure 42 • **Sending Serial Tx Data Off-Die—Lane 0**



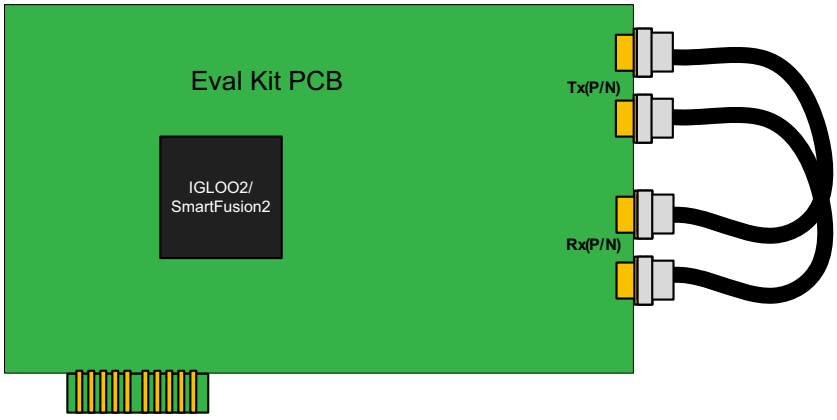
5. The Evaluation Kit board connects Lane 1 on the PCB to loop back Tx and Rx. This loopback demonstrates a complete path with data transmitted and received. For example, select **Lane 1**, **Serial Data (Off-Die)**, **PRBS7**, and click **Start**, as shown in the following figure.
   This test generates and checks PRBS7 data going off-chip and folded back on the PCB to the receiver.

Figure 43 • **Transmitting Data Through On-Board Loopback—Lane 1**



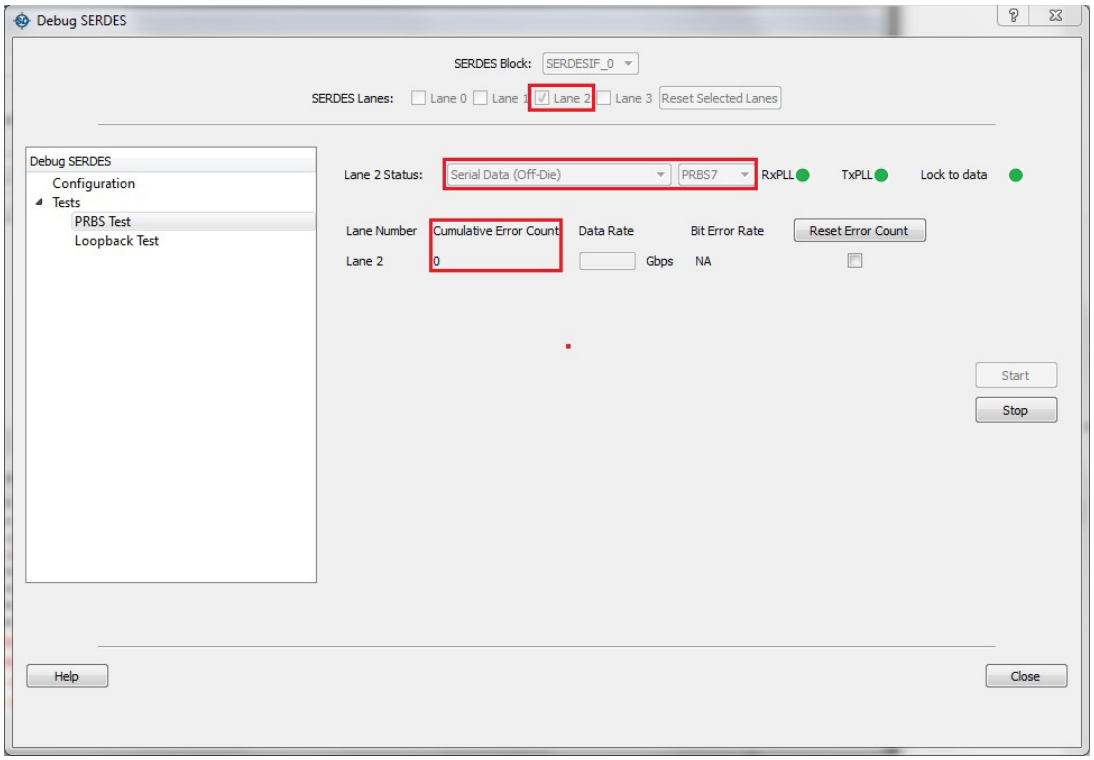
6. The Tx and Rx channels of Lane 2 can be interconnected in a loopback configuration using coaxial cables. After connecting a pair of high-quality 50 Ω SMA cables to the SMA connections on the Evaluation Kit board, as shown in the following figure, SerDes debug can be used to send data off-board and check for errors.

Figure 44 • **Evaluation Kit Board with External Coax Loopback Setup**



7. Select Lane 2, Serial Data (Off-Die), PRBS7, and click **Start**, as shown in the following figure.

Figure 45 • **External Cable Loopback**



8. The Lane 2 SMA test connections can be used for interconnecting with high-speed coaxial cables to test equipment or other test fixtures like test backplanes. When the Lane 2 test is started without any means to connect the Tx and Rx together, as shown in the following figure, **Lock to data** status goes red (see Figure 47, page 39) as the link is broken between the pattern generator and the checker. This setup sends a data pattern of the board for analysis on the test equipment.

Note: SMA Male-to-SMA male precision cables, such as *Pasternack Industries part number PE39429-12* (or equivalent), are recommended.

Figure 46 • **Connecting Lane 2 to the Test Equipment**

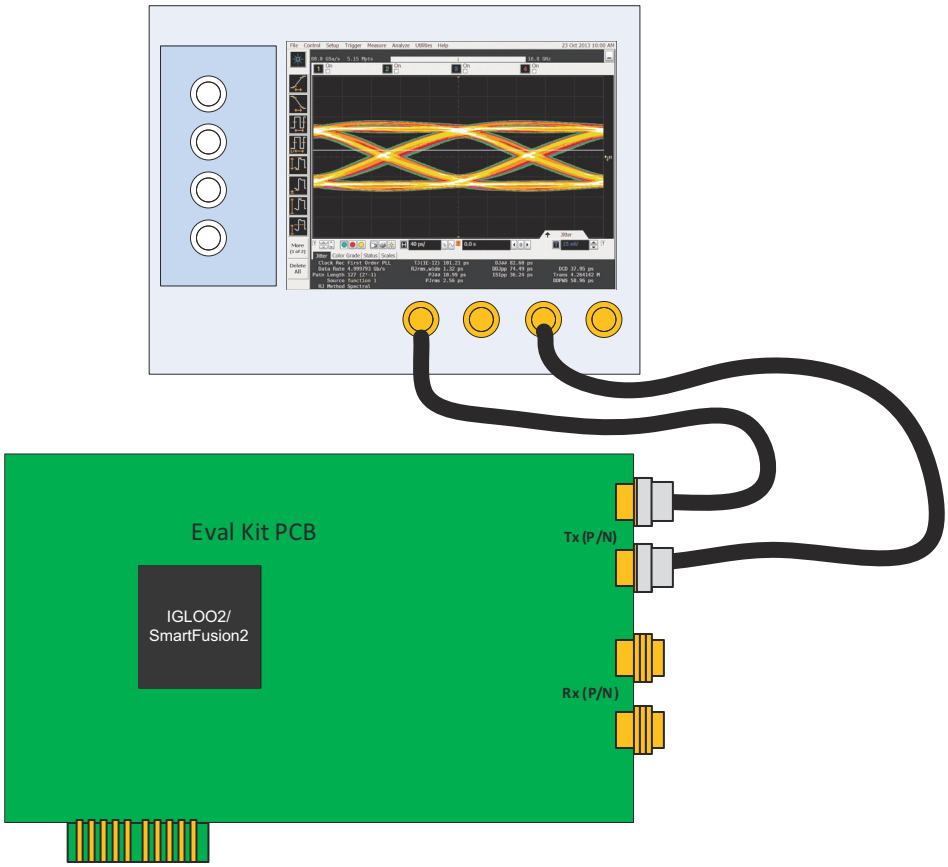
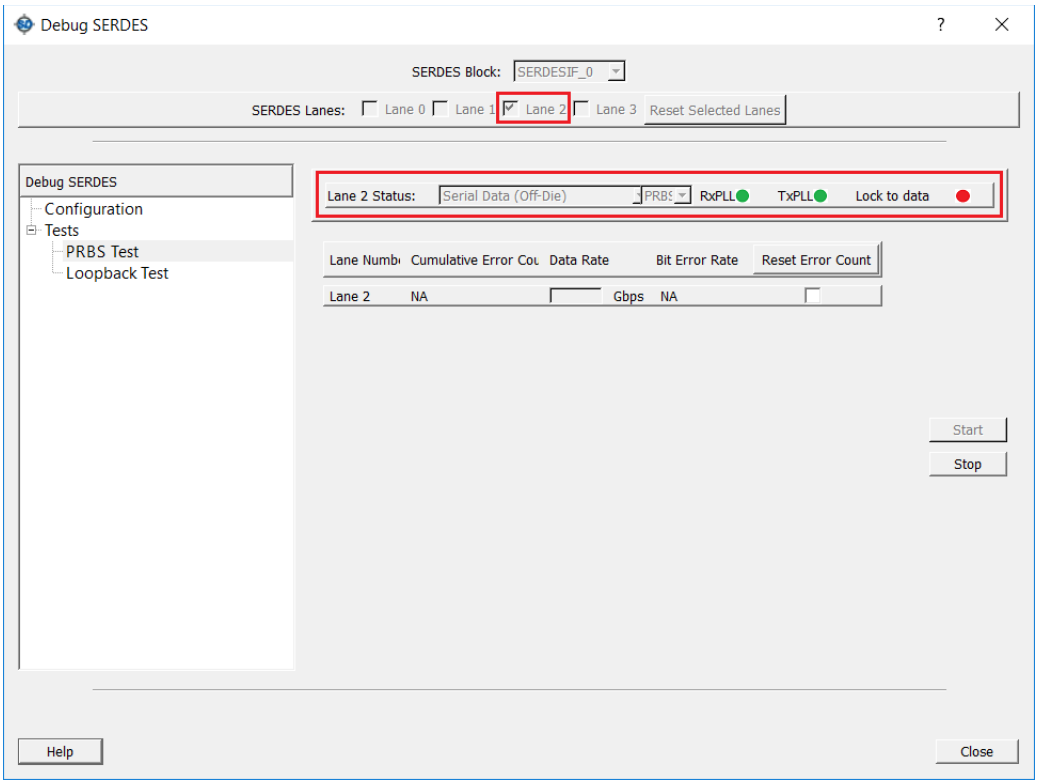


Figure 47 • **Transmitting Data Off-Board—Lane 2**

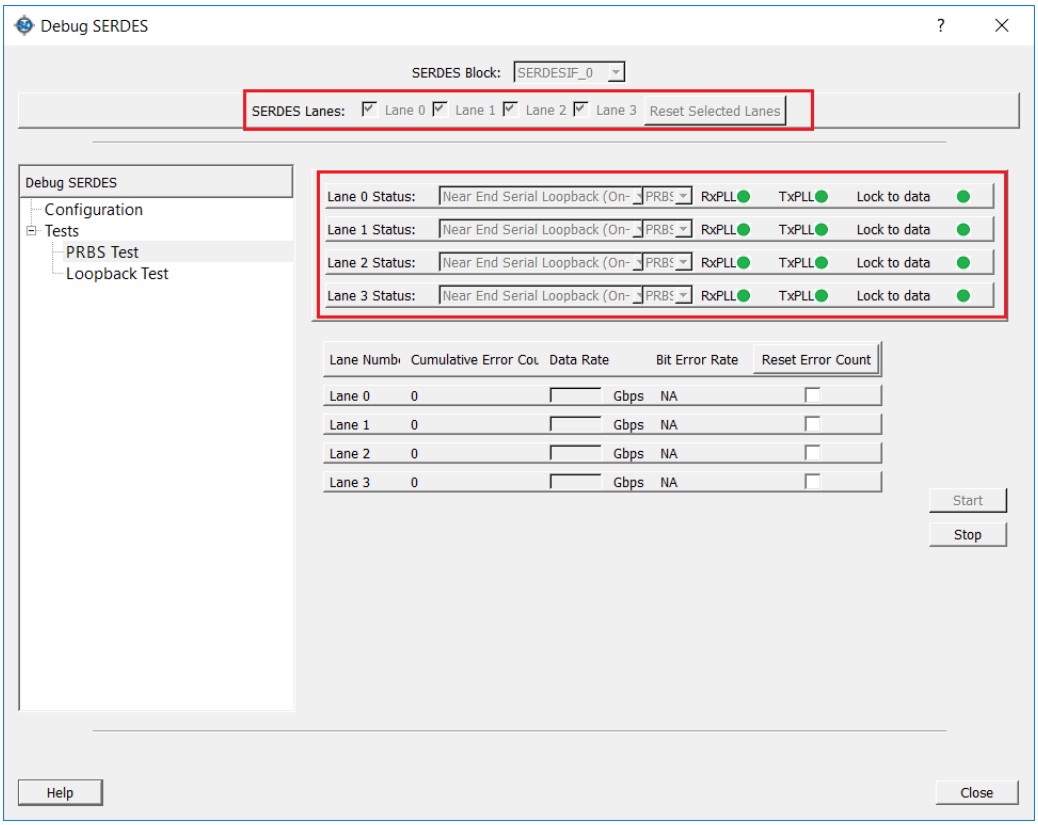


Several test patterns are available from the test pattern generator. PRBS7 is a very typical pattern for testing signal integrity in communication applications.

BER is the count of the number of errors over time to provide a level of confidence for a high-speed link. For a 2.5 Gbps test, it takes about three minutes with zero errors to achieve a BER of 10e-12. The SmartDebug SerDes provides an error counter that can be used for BER test.

Multiple lanes can be tested at a time by selecting the required lanes, as shown in the following figure.

Figure 48 • **Multiple Lane Testing**



2.5.4.1 Far-End Loopback Support

Far-end loopback is supported from the **Loopback Test** tab. From this tab, you can receive data from a far-end source and fold the received data (Rx) back out of the transmitter (Tx).

By using the Evaluation Kit board, data is received from a far-end transmit source, such as a device or test equipment. It is received into Lane 2 and looped back out of the transmitter, as shown in the following figure.

This is performed by selecting SerDes **Lane 2**, **PCS Far End PMS Rx to Tx Loopback** Test Type, and clicking **Start**, as shown in Figure 50, page 41.

Data entering the SMA connectors on Lane 2 of the Evaluation Kit board is observed coming off the board on the Tx SMA connectors.

Note: In this test, the IGLOO2 Evaluation Kit board or SmartFusion2 Evaluation Kit board must use the same SerDes reference clock as the far-end device that is sending and receiving the data. The datapath through the SERDEIF goes through the CDR and reclocks the data to the local REFCLK. This requires 0 ppm difference between the far-end clock source and the Evaluation Kit clock source. For this, use the SMA inputs (designators J17 and J21) of the board as the input for the SerDes REFCLK.

Figure 49 • **Far-End Loopback on the Evaluation Board**

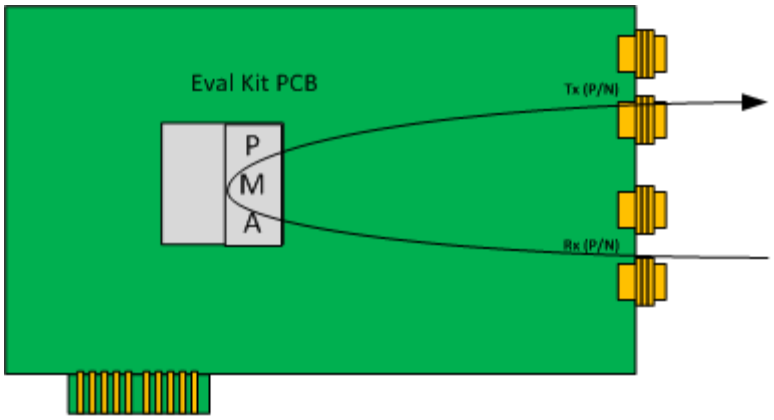
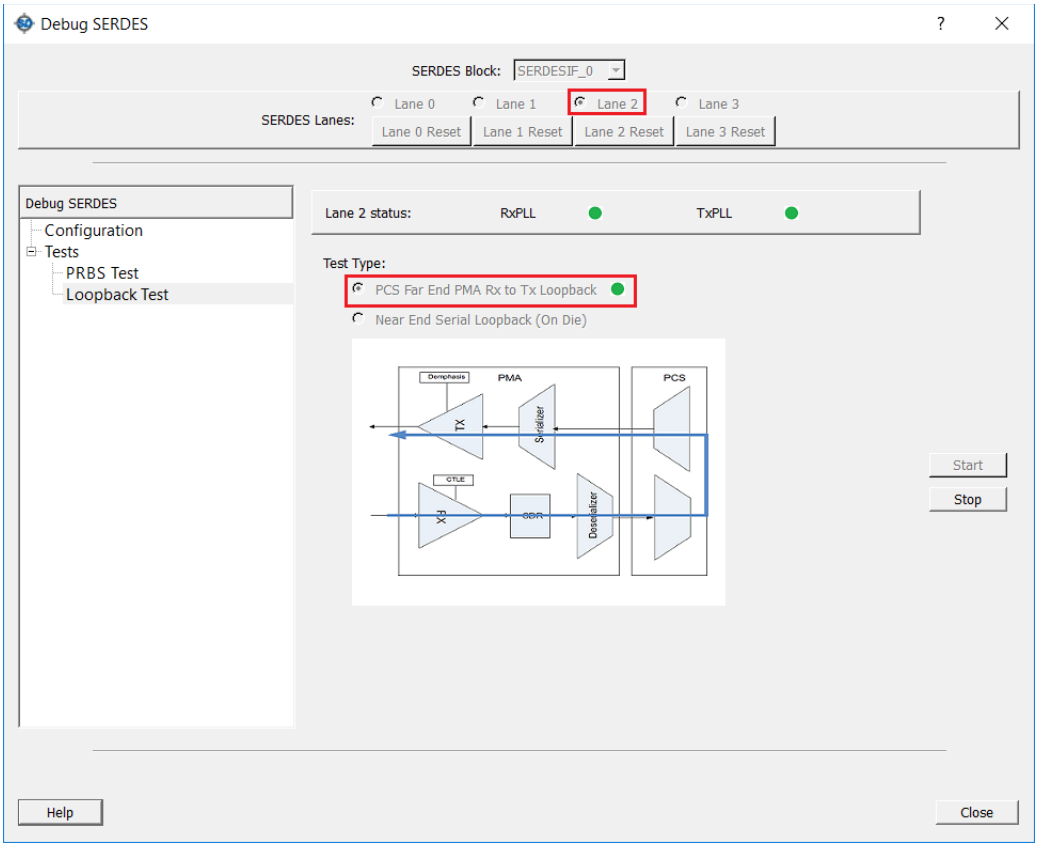


Figure 50 • **PCS Far-End Rx to Tx Loopback**

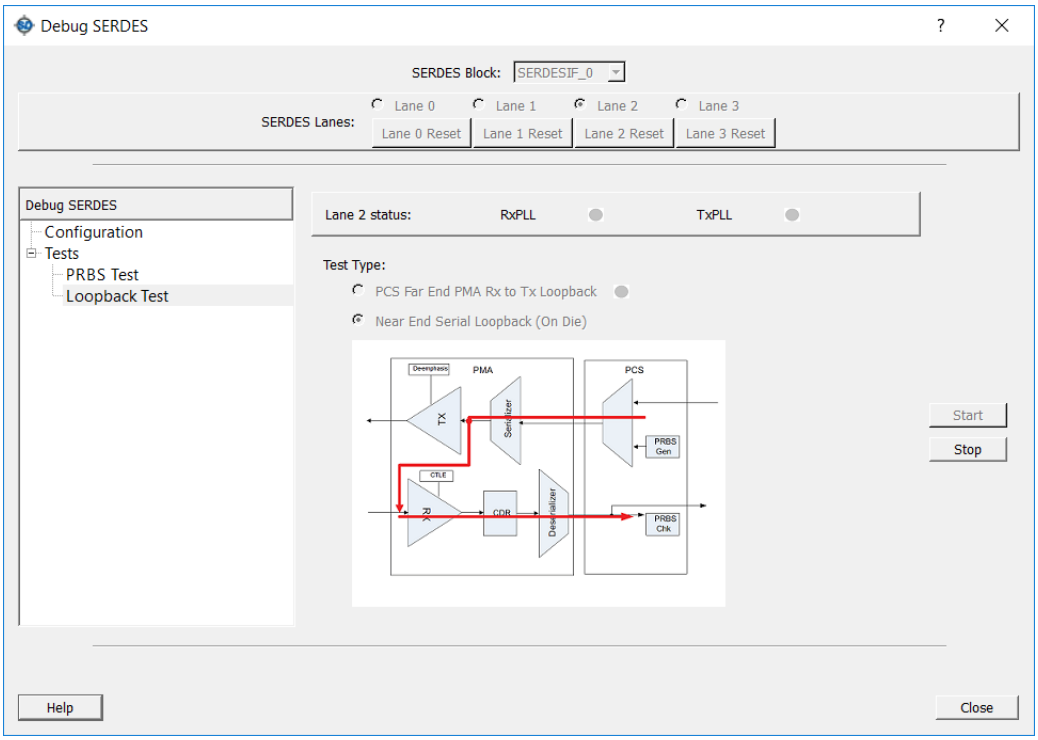


2.5.4.2 Near-End Serial Loopback

The near-end serial loopback test feature, as shown in the following figure, includes a loopback that folds back the TX to the RX. This feature requires the FPGA design to verify the correct operation of the looping data stream.

Use this feature to allow an FPGA application to send a data stream into the SerDes, and to bring the data stream back into the FPGA from the SerDes. This operation is done at the device pins without leaving the device. All data generation and checking is done in the FPGA application design.

Figure 51 • **Loopback Test Feature**



2.5.4.3 Tcl Support

The SerDes Debug tool set permits execution of Tcl scripts. This allows customized writes and reads of the entire SerDes register base. Tcl can be used to update or check status of the SerDes system, PCIe system, and SerDes lane registers.

Tcl command syntax is:

```
read_register  -addr  <RegisterAddress >
write_register -addr <RegisterAddress> -value <RegisterValue>
```

where, `RegisterAddress` is a 8 hex character (with optional 0x prefix) example: 0x4002200C

`RegisterValue` is a 1-8 hex character (with optional 0x prefix) example: 0x1, 0x1F

Example:

```
read_register -addr 0x4002200C
write_register -addr 0x4002E008 -value 0x3
```

Address for the SerDes blocks are as follows:

```
SERDESIF_0                    0x40028000 – 0x4002A3FF
SERDESIF_1                    0x4002C000 – 0x4002E3FF
SERDESIF_2                    0x40030000 – 0x400323FF
SERDESIF_3                    0x40034000 – 0x400363FF
```

Within each SerDes block, the memory map is as follows:

Name – Offset from the base address (example, for SERDESIF_0 the base address is 0x40028000).

PCIe Core register map	0x0000 – 0x0FFF
Lane 0 registers	0x1000 – 0x13FF
Lane 1 registers	0x1400 – 0x17FF
Lane 2 registers	0x1800 – 0x1BFF
Lane 3 registers	0x1C00 – 0x1FFF
SERDESIF system register map	0x2000 – 0x23FF

Example Tcl applications:

1. To access the Tx impedance ratio register for lane 2 in SERDESIF_1, the address is 0x4002C000 (SERDESIF_1 base) + 0x1800 (lane 2 offset) + 0x0C (register offset) = 0x4002D80C
2. To access the PRBS control register for lane 0 in SERDESIF_0, the address is 0x40028000 (SERDESIF_0 base) + 0x1000 (lane 0 offset) + 0x190 (register offset) = 0x40029190

For register map details, see *UG0447: IGLOO2 and SmartFusion2 High Speed Serial Interfaces User Guide*.

Read only the lanes that are programmed by the design. Also, read the PCIe registers only if any of the lanes have PCIe protocol.

Example:

The Tcl script below is used to alter the TX_PST (Transmit Post Emphasis) setting of Lane 0 of SERDESIF_0.

```
# Serdes block 0

# Set the config_phy_mode_1 value by separately running the following Tcl
command "" in separate script and write the value without '0x' prefix

set config_phy_mode_1 80f

# set config_phy_mode_1


scan $config_phy_mode_1 %x phyMode1Val


# set CONFIG_REG_LANE_SEL for this lane

set lane0PhyMode [expr { ($phyMode1Val & 255) | 256 }]

scan [format %x $lane0PhyMode] %s lane0PhyMode

write_register -addr 0x4002a028 -val $lane0PhyMode

puts "Serdes lane0 registers"


write_register -addr 0x40029028 -val 0x1a

puts "TX_PST_RATIO"

read_register -addr 0x40029028


#Reset the config_phy_mode_1 value to original value

write_register -addr 0x4002a028 -val $config_phy_mode_1
```

The value of the CONFIG_PHY_MODE_1 register must be known in the preceding example. This register contains the value of the CONFIG_REG_LANE_SEL, which defines the lanes accessed in the

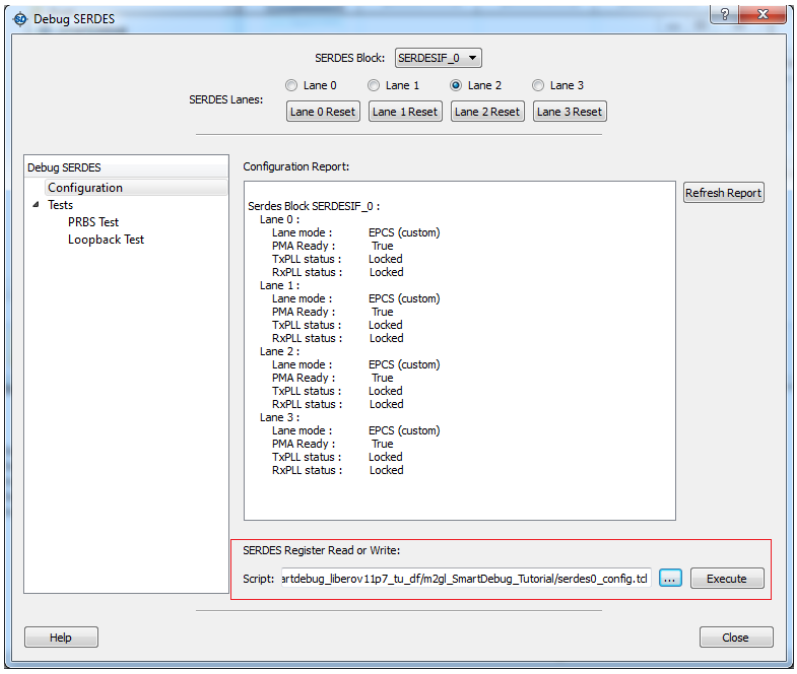
design. In this example, reading the CONFIG_PHY_MODE_1 register and passing its value and the associated offset targets the correct lane.

Note: Some SerDes PMA register settings are updated only after the assertion of a PHY_RESET or writing to the UPDATE_SETTINGS register.

For more information about the Tcl commands and syntax, see the *SmartFusion2, IGLOO2 and RTG4 Tcl for SoC Documentation*.

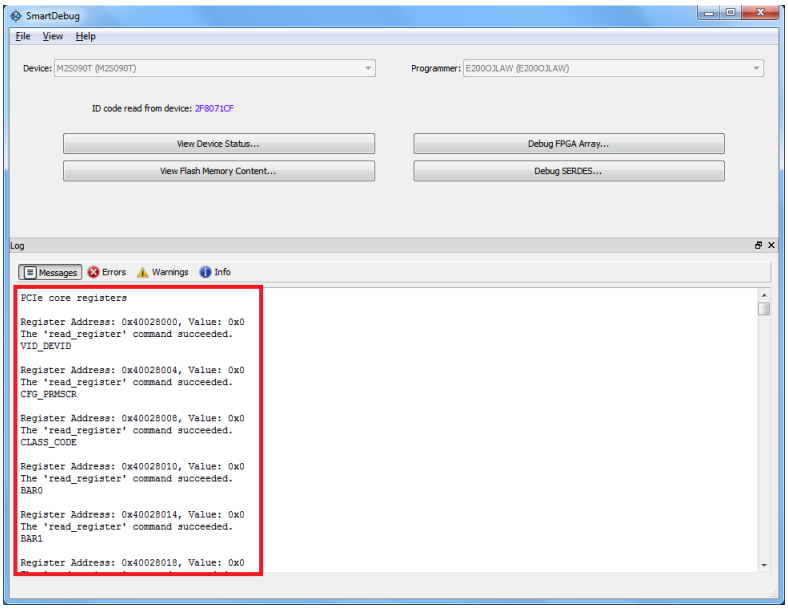
From the **Debug SERDES Configuration** tab, an executable Tcl script can be imported. Browse to the Tcl script file and click **Execute**, as shown in the following figure. The script contains commands to write or read registers using a flattened top for most address mapping.

Figure 52 • **Tcl Script Execution User Interface**



After execution of the Tcl SerDes access, the log is displayed in the SmartDebug console **Log** pane, as shown in the following figure.

Figure 53 • **SerDes Access Log**



For more Tcl examples, see Appendix: Tcl Script Examples, page 56.

2.5.4.3.1 Executing SerDes Debug from SmartDebug Tcl

PRBS and loopback tests can be performed using Tcl commands.

PRBS Test

User-level command: used in PRBS test to start, stop, reset the error counter, and read the error counter value.

```
prbs_test [-deviceName <device_name>]  -start -serdes <num> -lane <num> [-
near] -pattern <PatternType> [-value <PatternValue>]


prbs_test [-deviceName <device_name>]  -stop -serdes <num> -lane <num>


prbs_test [-deviceName <device_name>]  -reset_counter -serdes <num> -lane
<num>


prbs_test [-deviceName <device_name>] -read_counter -serdes <num> -lane <num>
```

- **deviceName** <*device_name*>: parameter is optional, if only one device is available in the current configuration or set for debug. For more information, see *SmartDebug for Software v11.7 User's Guide.*
- **start:** start PRBS test.
- **stop**: stop PRBS test.
- **reset_counter**: reset the PRBS error count value to 0.
- **read_counter**: read and print the error count value.
- **SerDes** <*num*>: SerDes block number. Must be between 0 and 4 and varies between dies.
- **lane** <*num*>: SerDes lane number. Must be between 0 and 4.
- **near**: corresponds to near-end (on-die) option for PRBS test. Not specifying this option implies off-die.
- **pattern** <*PatternType*>: the pattern sequence to be used for the PRBS test. It can be one of the following:
- **prbs7** or **prbs11** or **prbs23** or **prbs31**

- **custom**
- **user**
- **value** <*PatternValue*>: specifies the pattern type value for cases other than PRBS sequences. It can be one of the following:
  - If custom is selected above, then it must be one of all_zeros, all_ones, alternated, or dual_alternated.
  - If user is selected above, then it must be 20 hexadecimal characters.

Example:

```
prbs_test -start -serdes 1 -lane 0 -near -pattern prbs11
```

```
prbs_test -start -serdes 2 -lane 2 -pattern custom -value all_zeros
```

```
prbs_test -start -serdes 0 -lane 1 -near -pattern user -value
0x0123456789ABCDEF0123
```

Loopback Test

User level command: used to start and stop the loopback tests.

```
loopback_test [-deviceName <device_name>] -start -serdes <num> -lane <num> -
type <LoopbackType>
```

```
loopback_test [-deviceName <device_name>] -stop -serdes <num> -lane <num>
```

- **deviceName** <*device_name*>: parameter is optional, if only one device is available in the current configuration or set for debug (see the SmartDebug User Guide, for details).
- **start:** start loopback test.
- **stop**: stop loopback test.
- **serdes** <*num*>: SerDes block number. Must be between 0 and 4 and varies between dies.
- **lane** <*num*>: SerDes lane number. Must be between 0 and 4.
- **type** <*LoopbackType*>: specifies the loopback test type. Must be one of the following:
  - **plesio** (PCS Far End PMA Rx to Tx Loopback)
  - **parallel**
  - **meso** (PCS Far End PMA Rx to Tx Loopback)

Example:

```
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
```

Tcl scripting for SerDes SmartDebug can be used in batch mode without launching SmartDebug. The following is an example batch script:

```
open_project -project {D:/my_serdes_design/my_serdes.pro}
```

```
set_debug_device -name {M2S/M2GL050(T|S|TS)}
```

```
read_id_code
```

```
set_programming_file -name {M2S/M2GL050(T|S|TS)} -file {./
SERDES1_REFCLK1_EPCS_MODE_SF2_DEV_KIT/SERDES1_REFCLK1_EPCS_MODE/designer/
SERDES_LOOPBACK_top/export/SERDES_LOOPBACK_top.stp}
```

```
run_selected_actions
```

```
set_debug_device -name {M2S/M2GL050(T|S|TS)}
```

```
//Place serdes tcl commands after here
```

2.6 Standalone SmartDebug

SmartDebug is offered as a standalone utility. This allows SmartDebug to be used on other host computer systems without the full installation of the Libero SoC software. In standalone SmartDebug flow, a standalone SmartDebug project needs to be created.

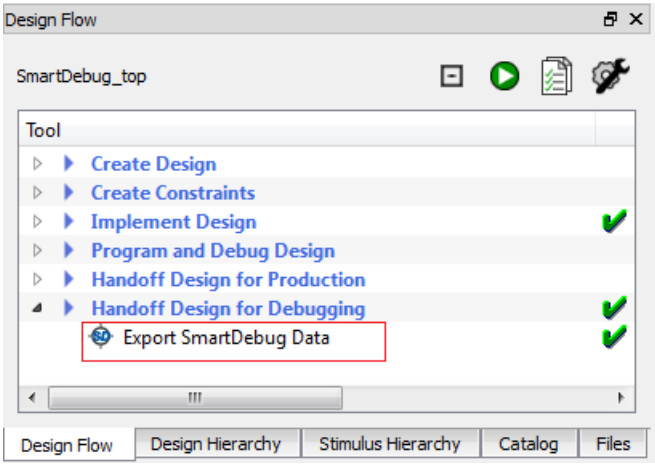
There are two ways to create a stand-alone SmartDebug project:

- Import DDC files from Libero
- Create automatically

The following procedure describes how to import a DDC file from Libero:

1. Enable the appropriate debug features such as probe insertion of the targeted devices in the Libero design project before creating the DDC file.
2. Double-click **Export SmartDebug Data** to generate the DDC file from the Libero SoC design flow.

Figure 54 • **Export SmartDebug Data**



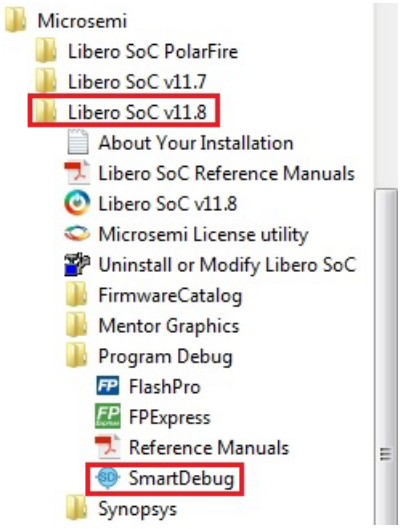
The exported DDC file is located at: *<project path>\designer\Project\export\Project_top.ddc*

The exported DDC file and programming file need to be transferred to a standalone PC for debug sessions.

The following procedure describes how to create a SmartDebug project:

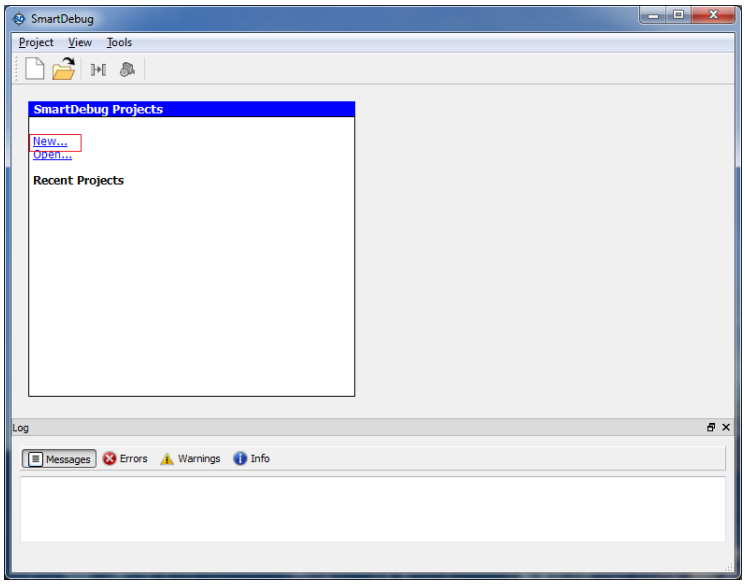
1. Connect a FlashPro programmer to a valid hardware device.
2. Start SmartDebug standalone utility. The standalone SmartDebug is found in the **Program Debug** program group of the **Microsemi Libero SoC v11.8** program group, as shown in the following figure. You can also find it as an icon on your desktop.

Figure 55 • **Starting Standalone SmartDebug**



3. Click **New**.

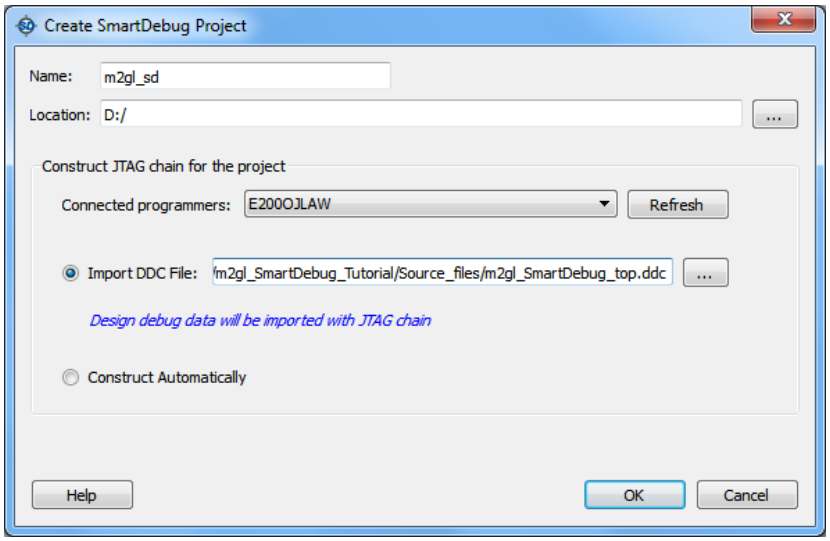
Figure 56 • **New SmartDebug Project**



The **Create SmartDebug Project** window is displayed.

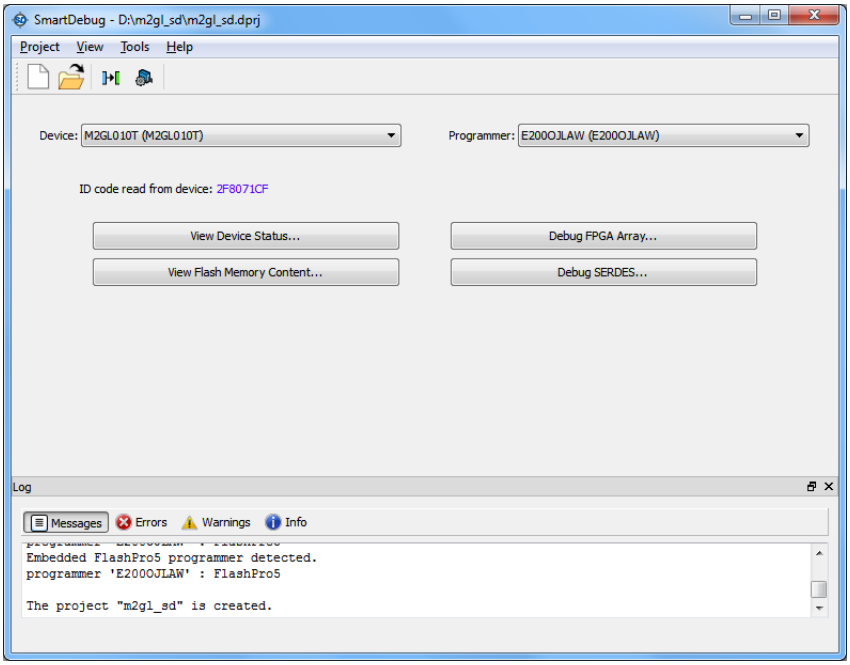
4. Enter the Name of the SmartDebug project.
5. Browse to the desired directory **Location** to save the SmartDebug project, and select the DDC file related to the Libero SoC project.
6. Click **OK**.

Figure 57 • **Create SmartDebug Project**



The SmartDebug project is created. At this point, SmartDebug works exactly the same as when it is launched from Libero. For information about debugging, see Debugging the Design, page 9. The standalone SmartDebug window is displayed as shown in the following figure.

Figure 58 • **Standalone SmartDebug UI**



2.7 Demo Mode

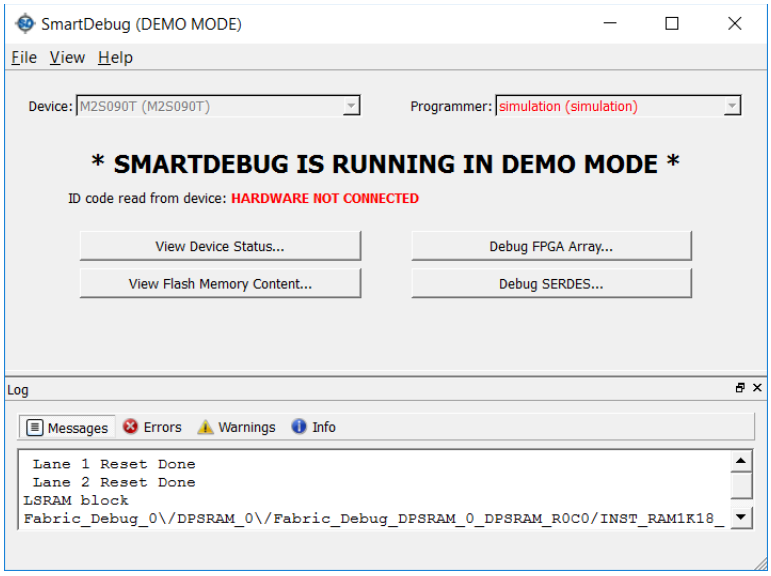
The Demo mode allows you to experience SmartDebug functionalities (Active Probe, Live Probe, Memory Blocks, and Debug SERDES) without having to connect a board to the system running SmartDebug.

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

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

The following figure is displayed when you open SmartDebug in Demo mode.

Figure 59 • **Launching SmartDebug in Demo Mode**



Demo mode supports all the functionalities that Integrated and Standalone modes support.

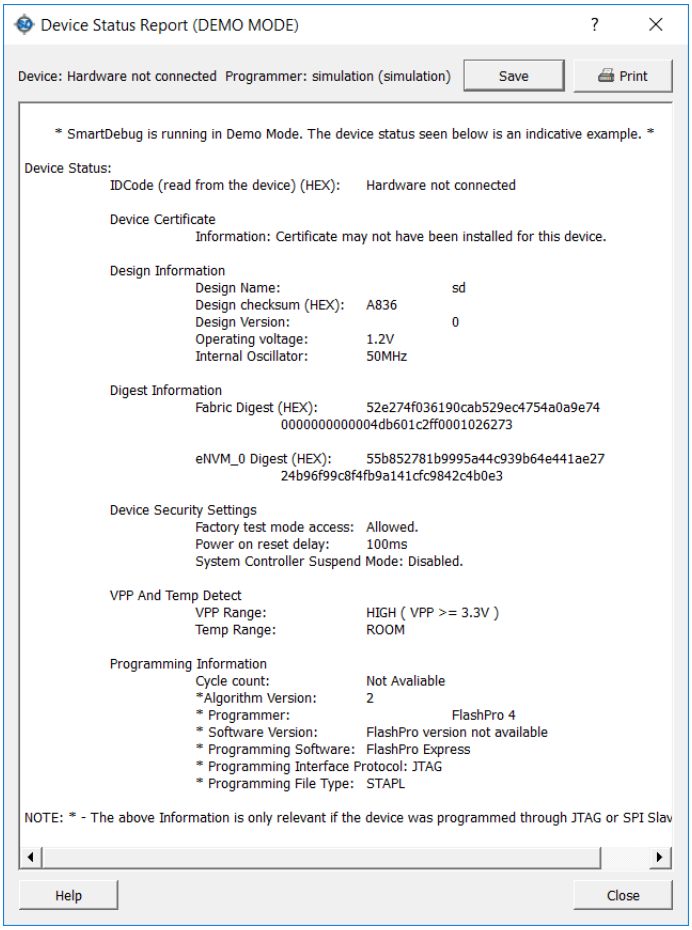
- View Device Status
- Debug FPGA Array
- View Flash Memory Content
- Debug SerDes

Note: View Flash Memory Content is not supported in Demo mode.

2.7.1 View Device Status

The following figure shows an example of the Device Status Report in Demo Mode.

Figure 60 • **Viewing Device Status**



2.7.2 Debug FPGA Array

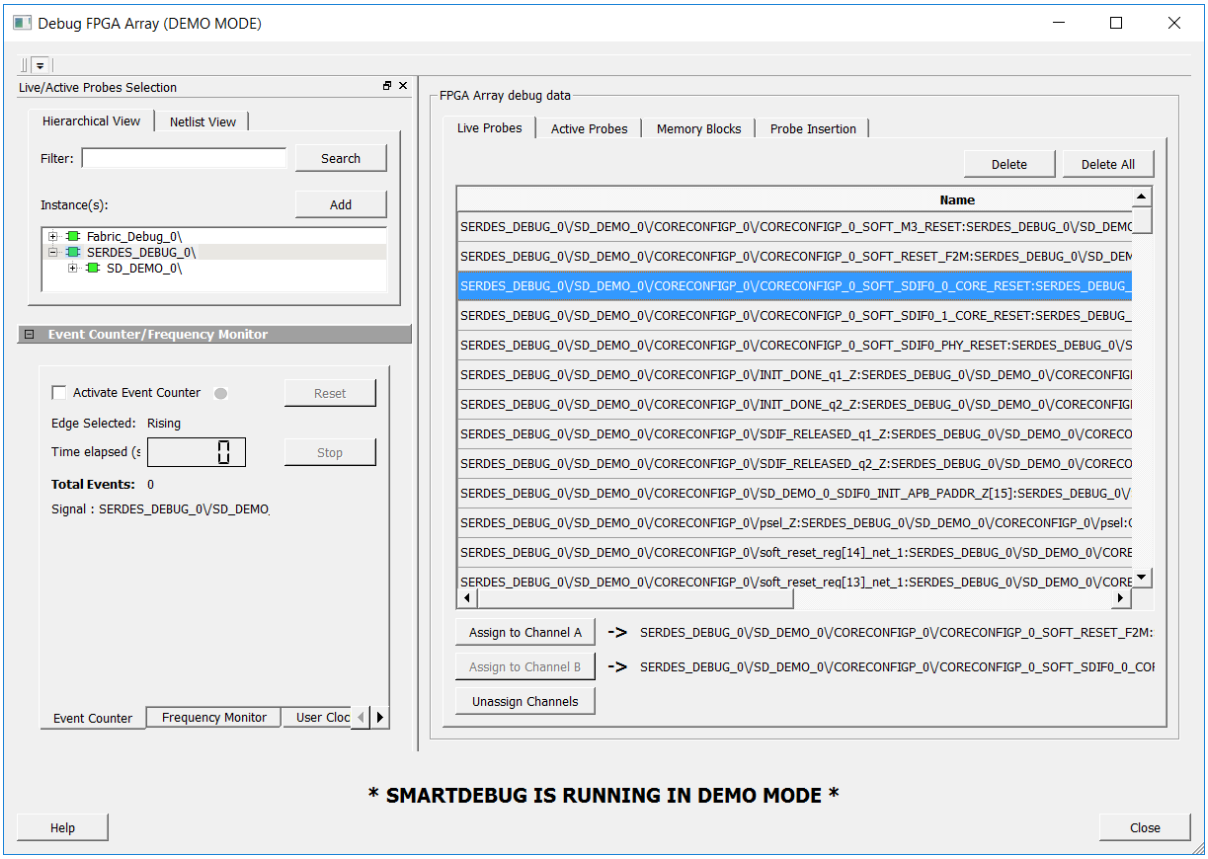
In the Debug FPGA Array dialog box, you can view your Live Probes, Active Probes, Memory Blocks, and Insert Probes (Probe Insertion).

Note: Insert Probes (Probe Insertion) is not supported in Demo mode.

2.7.2.1 Live Probes

You can assign and unassign Live Probes to Channel A and Channel B. The following figure shows an example of Live Probes assignment.

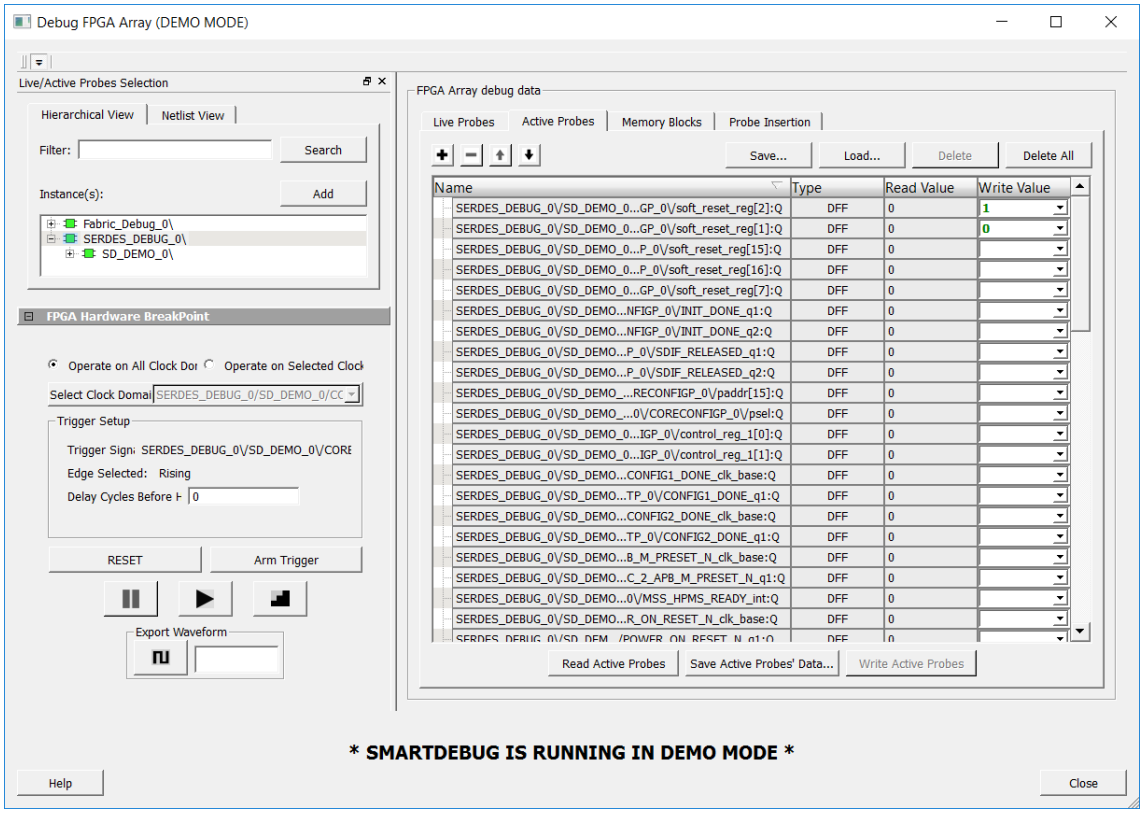
Figure 61 • **Assigning Live Probes to Channels**



2.7.2.2 Live Probes

In Demo mode, a temporary probe data file with details of current and previous values of probes is added in the active probes. The write values of probes are updated to this file and the GUI is updated with values from this file when you click Write Probes. The values are read when you click Read Probes. If there is no existing data for a probe in the file, the read values are all 0s and continue to update the values based on the changes.

Figure 62 • **Reading or Writing Live Probe Values**

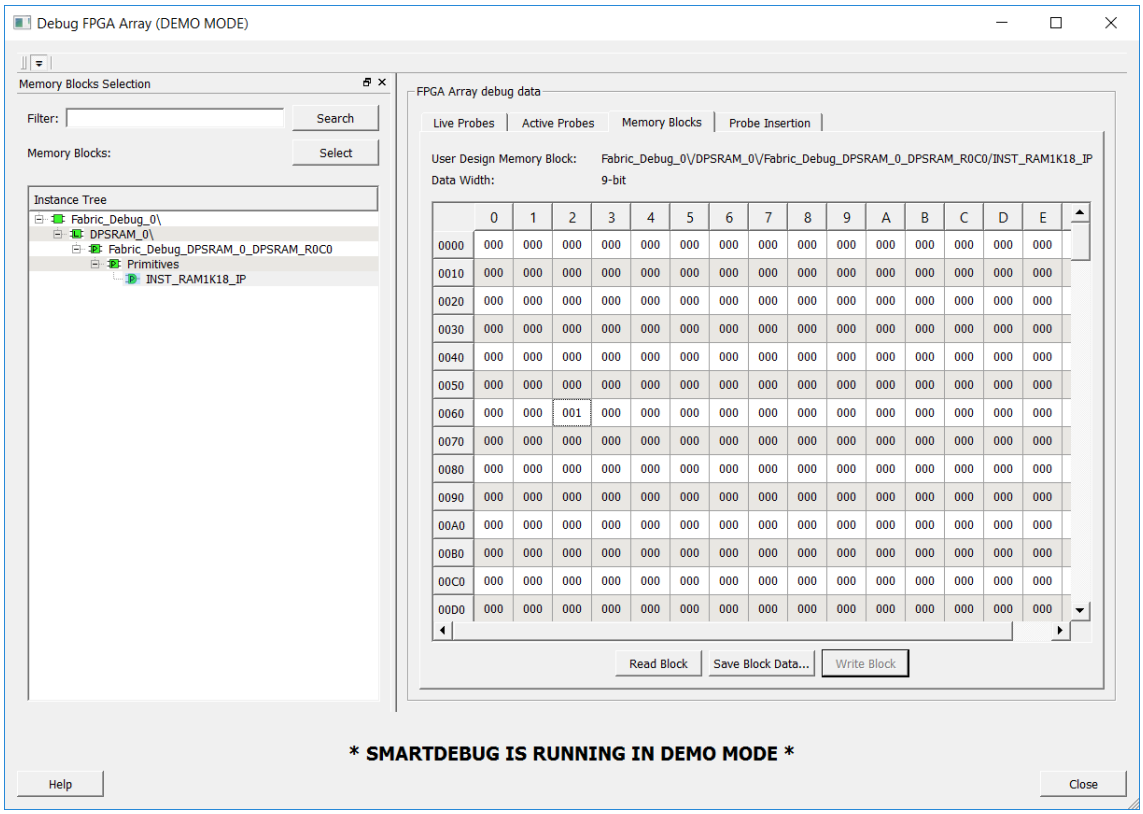


2.7.2.3 Memory Blocks

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.It supports both Physical and Logical views.

The following figure shows an example of memory blocks in Demo mode.

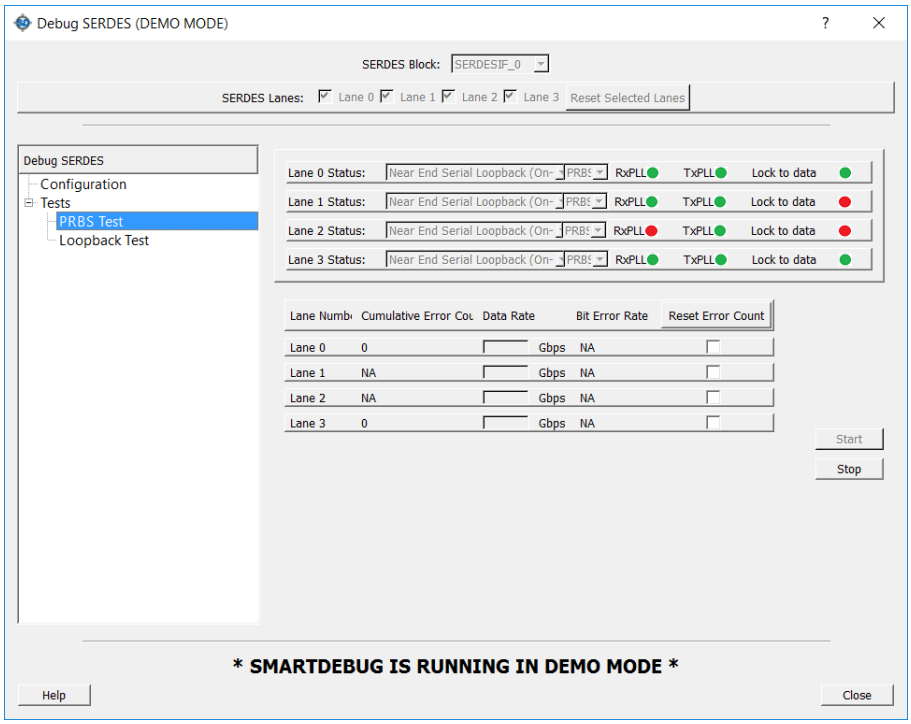
Figure 63 • **Memory Blocks in Demo Mode**



2.7.3 Debug SerDes

In Demo mode, you can run multi lane PRBS and loopback tests. SerDes demo mode does not create any temporary data file, and is provided for you to experience all the features of SerDes. All the channels are enabled in Demo mode, and in that, a couple of channels work properly and couple of channels show connectivity issues to display all possible GUI options.

Figure 64 • **Debugging SerDes**



2.8 Conclusion

This tutorial demonstrated capabilities of SmartDebug. SmartDebug provides capabilities to observe and analyze many embedded device features. Live probes give a real-time access to device test points, and internal logic states can be accessed using active probes. The SmartDebug SerDes utility assists FPGA and board designers to validate signal integrity of high-speed serial links in a system and improve board bring-up time. This is completed in real-time without any design modifications. The PMA analog settings can be tuned to optimize link performance and to match the design to the system.

3 Appendix: Tcl Script Examples

The following Tcl scripts can be used for debugging.

3.1 Example 1: Change M/N/F Registers for Lane1 and Lane2 of SERDESIF_0

```
# set CONFIG_REG_LANE_SEL
write_register -addr 0x4002a028 -val 20F
read_register -addr 0x4002a028

write_register -addr  0x40029410 -val 0x0
puts "PLL_F_PCLK_RATIO_Lane1"

write_register -addr  0x40029414 -val 0x13
puts "PLL_M_N_Lane1"

write_register -addr 0x40029600 -val 0x1
puts "UPDATE_SETTINGS_Lane1"

puts "Serdes lane1 registers"

# set CONFIG_REG_LANE_SEL
write_register -addr 0x4002a028 -val 40F

write_register -addr 0x40029810 -val 0x0
puts "PLL_F_PCLK_RATIO_Lane2"
write_register -addr 0x40029814 -val 0x13
puts "PLL_M_N_Lane2"

write_register -addr 0x40029a00 -val 0x1
puts "UPDATE_SETTINGS_Lane2"

puts "Serdes lane2 registers"
```

3.2 Example 2: Change RX LEQ Registers Lane2 of SERDESIF_0

```
# set CONFIG_REG_LANE_SEL
write_register -addr 0x4002a028 -val 40F

write_register -addr 0x4002981c -val 0x00
puts "RE_AMP_RATIO_Lane2"

write_register -addr 0x40029820 -val 0x00
puts "RE_CUT_RATIO_Lane2"

write_register -addr 0x40029a00 -val 0x1
puts "UPDATE_SETTINGS_Lane2"
```

3.3 Example 3: Change TX De-Emphasis Registers Lane2 of SERDESIF_0

```
# set CONFIG_REG_LANE_SEL
write_register -addr 0x4002a028 -val 40F

write_register -addr 0x40029828 -val 0xa
puts "TX_PST_RATIO_Lane2"

write_register -addr 0x4002982c -val 0x0
puts "TX_PRE_RATIO_Lane2"

write_register -addr 0x40029a00 -val 0x1
puts "UPDATE_SETTINGS_Lane2"
```

3.4 Example 4: Sample Script to Read SerDes and PCIe Registers

```
# Serdes block 0
# Set the following inputs according to your design/need. 0 means false, 1
means true.
# Set the config_phy_mode_1 value by separately running the following tcl
command "" in separate script and write the value without '0x' prefix
set config_phy_mode_1 80f
# set config_phy_mode_1
set dumpPcIecore0Registers 1
set dumpPcIecore1Registers 0
set dumpLane0Registers 1
```

```
set dumpLane1Registers 0
set dumpLane2Registers 1
set dumpLane3Registers 0
set dumpSerdesSystemRegisters 1
set dumpSerdesSystem1Registers 1


scan $config_phy_mode_1 %x phyMode1Val

if {$dumpPCIeCore0Registers == 1} {

puts "************************************"
puts "**PCIE Controller 0 Core Registers**"

read_register -addr 0x40028000
puts "VID_DEVID"
read_register -addr 0x40028004
puts "CFG_PRMSCR"
read_register -addr 0x40028008
puts "CLASS_CODE"
read_register -addr 0x40028010
puts "BAR0"
read_register -addr 0x40028014
puts "BAR1"
read_register -addr 0x40028018
puts "BAR2"
read_register -addr 0x4002801C
puts "BAR3"
read_register -addr 0x40028020
puts "BAR4"
read_register -addr 0x40028024
puts "BAR5"
read_register -addr 0x4002802C
puts "SUBSYSTEM_ID"
read_register -addr 0x40028030
puts "PCIE_DEVSCR"
read_register -addr 0x40028034
puts "PCIE_LINKSCR"
```

```
read_register -addr 0x40028038
puts "TC_VC_MAPPING"
read_register -addr 0x4002803C
puts "CAPTURED_BUS_DEVICE_NB"
read_register -addr 0x40028040
puts "MSI_CTRL_STATUS"
read_register -addr 0x40028044
puts "LTSSM"
read_register -addr 0x40028048
puts "POWER_MGT_CAPABILITY"
read_register -addr 0x4002804C
puts "CFG_PMSCR"
read_register -addr 0x40028050
puts "AER_ECRC_CAPABILITY"
read_register -addr 0x40028054
puts "VC1_CAPABILITY"
read_register -addr 0x40028058
puts "MAX_PAYLOAD_SIZE"
read_register -addr 0x4002805C
puts "CLKREQ"
read_register -addr 0x40028060
puts "ASPM_L0S_CAPABILITY"
read_register -addr 0x40028064
puts "ASPM_L1_CAPABILITY"
read_register -addr 0x40028068
puts "TIMEOUT_COMPLETION"
read_register -addr 0x40028070
puts "PM_DATA_SCALE_0"
read_register -addr 0x40028074
puts "PM_DATA_SCALE_1"
read_register -addr 0x40028078
puts "PM_DATA_SCALE_2"
read_register -addr 0x4002807C
puts "PM_DATA_SCALE_3"
read_register -addr 0x40028080
puts "MSI_0"
read_register -addr 0x40028084
puts "MSI_1"
```

```
read_register -addr 0x40028088
puts "MSI_2"
read_register -addr 0x4002808C
puts "MSI_3"
read_register -addr 0x40028090
puts "MSI_4"
read_register -addr 0x40028094
puts "MSI_5"
read_register -addr 0x40028098
puts "MSI_6"
read_register -addr 0x4002809C
puts "MSI_7"
read_register -addr 0x400280a0
puts "ERROR_COUNTER_0"
read_register -addr 0x400280a4
puts "ERROR_COUNTER_1"
read_register -addr 0x400280a8
puts "ERROR_COUNTER_2"
read_register -addr 0x400280aC
puts "ERROR_COUNTER_3"
read_register -addr 0x400280b0
puts "CREDIT_ALLOCATION_0"
read_register -addr 0x400280b4
puts "CREDIT_ALLOCATION_1"
read_register -addr 0x400280b8
puts "CREDIT_ALLOCATION_2"
read_register -addr 0x400280bC
puts "CREDIT_ALLOCATION_3"
read_register -addr 0x400280C0
puts "AXI_SLAVE_WINDOW_0"
read_register -addr 0x400280C4
puts "AXI_SLAVE_WINDOW_1"
read_register -addr 0x400280C8
puts "AXI_SLAVE_WINDOW_2"
read_register -addr 0x400280CC
puts "AXI_SLAVE_WINDOW_3"
read_register -addr 0x40028100
puts "AXI_MASTER_WINDOW_0"
```

```
read_register -addr 0x40028104
puts "AXI_MASTER_WINDOW_1"
read_register -addr 0x40028108
puts "AXI_MASTER_WINDOW_2"
read_register -addr 0x4002810C
puts "AXI_MASTER_WINDOW_3"
read_register -addr 0x40028140
puts "IMASK"
read_register -addr 0x40028144
puts "ISTATUS"
read_register -addr 0x40028148
puts "ICMD"
read_register -addr 0x4002814C
puts "IRSTATUS"
read_register -addr 0x40028150
puts "IMSIADDR"
read_register -addr 0x40028154
puts "SLOTCAP"
read_register -addr 0x40028158
puts "SLOTCSR"
read_register -addr 0x4002815C
puts "ROOTCSR"
read_register -addr 0x40028160
puts "CFG_CONTROL"
read_register -addr 0x40028164
puts "CFG_WRITE_DATA"
read_register -addr 0x40028168
puts "CFG_READ_DATA"
read_register -addr 0x4002816C
puts "INFO"
read_register -addr 0x40028170
puts "IO_CONTROL"
read_register -addr 0x40028174
puts "IO_ADDR"
read_register -addr 0x40028178
puts "IO_WRITE_DATA"
read_register -addr 0x4002817C
puts "IO_READ_DATA"
```

```
read_register -addr 0x40028180
puts "CFG_FBE"
read_register -addr 0x40028184
puts "PREFETCH_IO_WINDOW"
read_register -addr 0x40028204
puts "PCIE_CONFIG"
read_register -addr 0x40028230
puts "PCIE_DEV2SCR"
read_register -addr 0x40028234
puts "PCIE_LINK2SCR"
read_register -addr 0x40028260
puts "ASPM_L0S_GEN2"
read_register -addr 0x40028300
puts "K_CNT_CONFIG_0"
read_register -addr 0x40028304
puts "K_CNT_CONFIG_1"
read_register -addr 0x40028308
puts "K_CNT_CONFIG_2"
read_register -addr 0x4002830C
puts "K_CNT_CONFIG_3"
read_register -addr 0x40028310
puts "K_CNT_CONFIG_4"
read_register -addr 0x40028314
puts "K_CNT_CONFIG_5"
}

if {$dumpPCIeCore1Registers == 1} {
puts "**********************************"
puts "**PCIE Controller 1 Core Registers**"

read_register -addr  0x4002C000
puts "VID_DEVID"
read_register -addr  0x4002C004
puts "CFG_PRMSCR"
read_register -addr  0x4002C008
puts "CLASS_CODE"
read_register -addr  0x4002C010
puts "BAR0"
```

```
read_register -addr  0x4002C014
puts "BAR1"
read_register -addr  0x4002C018
puts "BAR2"
read_register -addr  0x4002C01C
puts "BAR3"
read_register -addr  0x4002C020
puts "BAR4"
read_register -addr  0x4002C024
puts "BAR5"
read_register -addr  0x4002C02C
puts "SUBSYSTEM_ID"
read_register -addr  0x4002C030
puts "PCIE_DEVSCR"
read_register -addr  0x4002C034
puts "PCIE_LINKSCR"
read_register -addr  0x4002C038
puts "TC_VC_MAPPING"
read_register -addr  0x4002C03C
puts "CAPTURED_BUS_DEVICE_NB"
read_register -addr  0x4002C040
puts "MSI_CTRL_STATUS"
read_register -addr  0x4002C044
puts "LTSSM"
read_register -addr  0x4002C048
puts "POWER_MGT_CAPABILITY"
read_register -addr  0x4002C04C
puts "CFG_PMSCR"
read_register -addr  0x4002C050
puts "AER_ECRC_CAPABILITY"
read_register -addr  0x4002C054
puts "VC1_CAPABILITY"
read_register -addr  0x4002C058
puts "MAX_PAYLOAD_SIZE"
read_register -addr  0x4002C05C
puts "CLKREQ"
read_register -addr  0x4002C060
puts "ASPM_L0S_CAPABILITY"
```

```
read_register -addr  0x4002C064
puts "ASPM_L1_CAPABILITY"
read_register -addr  0x4002C068
puts "TIMEOUT_COMPLETION"
read_register -addr  0x4002C070
puts "PM_DATA_SCALE_0"
read_register -addr  0x4002C074
puts "PM_DATA_SCALE_1"
read_register -addr  0x4002C078
puts "PM_DATA_SCALE_2"
read_register -addr  0x4002C07C
puts "PM_DATA_SCALE_3"
read_register -addr  0x4002C080
puts "MSI_0"
read_register -addr  0x4002C084
puts "MSI_1"
read_register -addr  0x4002C088
puts "MSI_2"
read_register -addr  0x4002C08C
puts "MSI_3"
read_register -addr  0x4002C090
puts "MSI_4"
read_register -addr  0x4002C094
puts "MSI_5"
read_register -addr  0x4002C098
puts "MSI_6"
read_register -addr  0x4002C09C
puts "MSI_7"
read_register -addr  0x4002C0a0
puts "ERROR_COUNTER_0"
read_register -addr  0x4002C0a4
puts "ERROR_COUNTER_1"
read_register -addr  0x4002C0a8
puts "ERROR_COUNTER_2"
read_register -addr  0x4002C0aC
puts "ERROR_COUNTER_3"
read_register -addr  0x4002C0b0
puts "CREDIT_ALLOCATION_0"
```

```
read_register -addr  0x4002C0b4
puts "CREDIT_ALLOCATION_1"
read_register -addr  0x4002C0b8
puts "CREDIT_ALLOCATION_2"
read_register -addr  0x4002C0bC
puts "CREDIT_ALLOCATION_3"
read_register -addr  0x4002C0C0
puts "AXI_SLAVE_WINDOW_0"
read_register -addr  0x4002C0C4
puts "AXI_SLAVE_WINDOW_1"
read_register -addr  0x4002C0C8
puts "AXI_SLAVE_WINDOW_2"
read_register -addr  0x4002C0CC
puts "AXI_SLAVE_WINDOW_3"
read_register -addr  0x4002C100
puts "AXI_MASTER_WINDOW_0"
read_register -addr  0x4002C104
puts "AXI_MASTER_WINDOW_1"
read_register -addr  0x4002C108
puts "AXI_MASTER_WINDOW_2"
read_register -addr  0x4002C10C
puts "AXI_MASTER_WINDOW_3"
read_register -addr  0x4002C140
puts "IMASK"
read_register -addr  0x4002C144
puts "ISTATUS"
read_register -addr  0x4002C148
puts "ICMD"
read_register -addr  0x4002C14C
puts "IRSTATUS"
read_register -addr  0x4002C150
puts "IMSIADDR"
read_register -addr  0x4002C154
puts "SLOTCAP"
read_register -addr  0x4002C158
puts "SLOTCSR"
read_register -addr  0x4002C15C
puts "ROOTCSR"
```

```
read_register -addr  0x4002C160
puts "CFG_CONTROL"
read_register -addr  0x4002C164
puts "CFG_WRITE_DATA"
read_register -addr  0x4002C168
puts "CFG_READ_DATA"
read_register -addr  0x4002C16C
puts "INFO"
read_register -addr  0x4002C170
puts "IO_CONTROL"
read_register -addr  0x4002C174
puts "IO_ADDR"
read_register -addr  0x4002C178
puts "IO_WRITE_DATA"
read_register -addr  0x4002C17C
puts "IO_READ_DATA"
read_register -addr  0x4002C180
puts "CFG_FBE"
read_register -addr  0x4002C184
puts "PREFETCH_IO_WINDOW"
read_register -addr  0x4002C204
puts "PCIE_CONFIG"
read_register -addr  0x4002C230
puts "PCIE_DEV2SCR"
read_register -addr  0x4002C234
puts "PCIE_LINK2SCR"
read_register -addr  0x4002C260
puts "ASPM_L0S_GEN2"
read_register -addr  0x4002C300
puts "K_CNT_CONFIG_0"
read_register -addr  0x4002C304
puts "K_CNT_CONFIG_1"
read_register -addr  0x4002C308
puts "K_CNT_CONFIG_2"
read_register -addr  0x4002C30C
puts "K_CNT_CONFIG_3"
read_register -addr  0x4002C310
puts "K_CNT_CONFIG_4"
```

```
read_register -addr  0x4002C314
puts "K_CNT_CONFIG_5"
}

if {$dumpSerdesSystem1Registers == 1} {
puts "***********************************"
puts "**Serdes system registers for PCI Controller 1**"

read_register -addr 0x4002e010
puts "CONFIG2_AXI_AHB_BRIDGE"
read_register -addr 0x4002e014
puts "CONFIG2_ECC_INTR_ENABLE"
read_register -addr 0x4002e018
puts "CONFIG2_TEST_IN"
read_register -addr 0x4002e01C
puts "TEST2_OUT_READ_ADDR"
read_register -addr 0x4002e020
puts "CONFIG2_PCIE_PM"
read_register -addr 0x4002e030
puts "CONFIG2_PCIE_0"
read_register -addr 0x4002e034
puts "CONFIG2_PCIE_1"
read_register -addr 0x4002e038
puts "CONFIG2_PCIE_2"
read_register -addr 0x4002e03C
puts "CONFIG2_PCIE_3"
read_register -addr 0x4002e040
puts "CONFIG2_BAR_SIZE_0_1"
read_register -addr 0x4002e044
puts "CONFIG2_BAR_SIZE_2_3"
read_register -addr 0x4002e048
puts "CONFIG2_BAR_SIZE_3_4"
read_register -addr 0x4002e054
puts "TEST2_OUT_READ_DATA"
read_register -addr 0x4002e05C
puts "SERDESIF2_INTR_STATUS"
read_register -addr 0x4002e084
puts "CONF2_AXI_MSTR_WNDW_0"
```

```
read_register -addr 0x4002e088
puts "CONF2_AXI_MSTR_WNDW_1"
read_register -addr 0x4002e08C
puts "CONF2_AXI_MSTR_WNDW_2"
read_register -addr 0x4002e090
puts "CONF2_AXI_MSTR_WNDW_3"
read_register -addr 0x4002e094
puts "CONF2_AXI_SLV_WNDW_0"
read_register -addr 0x4002e098
puts "CONF2_AXI_SLV_WNDW_1"
read_register -addr 0x4002e09C
puts "CONF2_AXI_SLV_WNDW_2"
read_register -addr 0x4002e0a0
puts "CONF2_AXI_SLV_WNDW_3"
read_register -addr 0x4002e0aC
puts "RESERVED2"
read_register -addr 0x4002e0b4
puts "ADVCONFIG2"
read_register -addr 0x4002e0b8
puts "ADVSTATUS2"
read_register -addr 0x4002e0bC
puts "ECC_ERR_INJECT2"
}

if {$dumpLane0Registers == 1} {
puts "**********************************"
# set CONFIG_REG_LANE_SEL for this lane
set lane0PhyMode [expr { ($phyMode1Val & 255) | 256 }]
sCan [format %x $lane0PhyMode] %s lane0PhyMode
write_register -addr 0x4002a028 -val $lane0PhyMode
puts "Serdes lane0 registers"

read_register -addr 0x40029000
puts "CR0"
read_register -addr 0x40029004
puts "ERRCNT_DEC"
read_register -addr 0x40029008
puts "RXIDLE_MAX_ERRCNT_THR"
```

```
read_register -addr 0x4002900C
puts "IMPED_RATIO"
read_register -addr 0x40029010
puts "PLL_F_PCLK_RATIO"
read_register -addr 0x40029014
puts "PLL_M_N"
read_register -addr 0x40029018
puts "CNT250NS_MAX"
read_register -addr 0x4002901C
puts "RE_AMP_RATIO"
read_register -addr 0x40029020
puts "RE_CUT_RATIO"
read_register -addr 0x40029024
puts "TX_AMP_RATIO"
read_register -addr 0x40029028
puts "TX_PST_RATIO"
read_register -addr 0x4002902C
puts "TX_PRE_RATIO"
read_register -addr 0x40029030
puts "ENDCALIB_MAX"
read_register -addr 0x40029034
puts "CALIB_STABILITY_COUNT"
read_register -addr 0x40029038
puts "POWER_DOWN"
read_register -addr 0x4002903C
puts "RX_OFFSET_COUNT"
read_register -addr 0x40029040
puts "PLL_F_PCLK_RATIO_5GBPS"
read_register -addr 0x40029044
puts "PLL_M_N_5GBPS"
read_register -addr 0x40029048
puts "CNT250NS_MAX_5GBPS"
read_register -addr 0x40029050
puts "TX_PST_RATIO_DEEMP0_FULL"
read_register -addr 0x40029054
puts "TX_PRE_RATIO_DEEMP0_FULL"
read_register -addr 0x40029058
puts "TX_PST_RATIO_DEEMP1_FULL"
```

```
read_register -addr 0x4002905C
puts "TX_PRE_RATIO_DEEMP1_FULL"
read_register -addr 0x40029060
puts "TX_AMP_RATIO_MARGIN0_FULL"
read_register -addr 0x40029064
puts "TX_AMP_RATIO_MARGIN1_FULL"
read_register -addr 0x40029068
puts "TX_AMP_RATIO_MARGIN2_FULL"
read_register -addr 0x4002906C
puts "TX_AMP_RATIO_MARGIN3_FULL"
read_register -addr 0x40029070
puts "TX_AMP_RATIO_MARGIN4_FULL"
read_register -addr 0x40029074
puts "TX_AMP_RATIO_MARGIN5_FULL"
read_register -addr 0x40029078
puts "TX_AMP_RATIO_MARGIN6_FULL"
read_register -addr 0x4002907C
puts "TX_AMP_RATIO_MARGIN7_FULL"
read_register -addr 0x40029080
puts "RE_AMP_RATIO_DEEMP0"
read_register -addr 0x40029084
puts "RE_CUT_RATIO_DEEMP0"
read_register -addr 0x40029088
puts "RE_AMP_RATIO_DEEMP1"
read_register -addr 0x4002908C
puts "RE_CUT_RATIO_DEEMP1"
read_register -addr 0x40029090
puts "TX_PST_RATIO_DEEMP0_HALF"
read_register -addr 0x40029094
puts "TX_PRE_RATIO_DEEMP0_HALF"
read_register -addr 0x40029098
puts "TX_PST_RATIO_DEEMP1_HALF"
read_register -addr 0x4002909C
puts "TX_PRE_RATIO_DEEMP1_HALF"
read_register -addr 0x400290a0
puts "TX_AMP_RATIO_MARGIN0_HALF"
read_register -addr 0x400290a4
puts "TX_AMP_RATIO_MARGIN1_HALF"
```

```
read_register -addr 0x400290a8
puts "TX_AMP_RATIO_MARGIN2_HALF"
read_register -addr 0x400290aC
puts "TX_AMP_RATIO_MARGIN3_HALF"
read_register -addr 0x400290b0
puts "TX_AMP_RATIO_MARGIN4_HALF"
read_register -addr 0x400290b4
puts "TX_AMP_RATIO_MARGIN5_HALF"
read_register -addr 0x400290b8
puts "TX_AMP_RATIO_MARGIN6_HALF"
read_register -addr 0x400290bC
puts "TX_AMP_RATIO_MARGIN7_HALF"
read_register -addr 0x400290C0
puts "PMA_STATUS"
read_register -addr 0x400290C4
puts "TX_SWEEP_CENTER"
read_register -addr 0x400290C8
puts "RX_SWEEP_CENTER"
read_register -addr 0x400290CC
puts "RE_SWEEP_CENTER"
read_register -addr 0x400290d0
puts "ATXDRR_7_0"
read_register -addr 0x400290d4
puts "ATXDRR_14_8"
read_register -addr 0x400290d8
puts "ATXDRP_DYN_7_0"
read_register -addr 0x400290dC
puts "ATXDRP_DYN_15_8"
read_register -addr 0x400290e0
puts "ATXDRP_DYN_20_16"
read_register -addr 0x400290e4
puts "ATXDRA_DYN_7_0"
read_register -addr 0x400290e8
puts "ATXDRA_DYN_15_8"
read_register -addr 0x400290eC
puts "ATXDRA_DYN_20_16"
read_register -addr 0x400290f0
puts "ATXDRT_DYN_7_0"
```

```
read_register -addr 0x400290f4
puts "ATXDRT_DYN_15_8"
read_register -addr 0x400290f8
puts "ATXDRT_DYN_20_16"
read_register -addr 0x400290fC
puts "ATXDRP_EI1_7_0"
read_register -addr 0x40029100
puts "ATXDRP_EI1_15_8"
read_register -addr 0x40029104
puts "ATXDRP_EI1_20_16"
read_register -addr 0x40029108
puts "ATXDRA_EI1_7_0"
read_register -addr 0x4002910C
puts "ATXDRA_EI1_15_8"
read_register -addr 0x40029110
puts "ATXDRA_EI1_20_16"
read_register -addr 0x40029114
puts "ATXDRT_EI1_7_0"
read_register -addr 0x40029118
puts "ATXDRT_EI1_15_8"
read_register -addr 0x4002911C
puts "ATXDRT_EI1_20_16"
read_register -addr 0x40029120
puts "ATXDRP_EI2_7_0"
read_register -addr 0x40029124
puts "ATXDRP_EI2_15_8"
read_register -addr 0x40029128
puts "ATXDRP_EI2_20_16"
read_register -addr 0x4002912C
puts "ATXDRA_EI2_7_0"
read_register -addr 0x40029130
puts "ATXDRA_EI2_15_8"
read_register -addr 0x40029134
puts "ATXDRA_EI2_20_16"
read_register -addr 0x40029138
puts "ATXDRT_EI2_7_0"
read_register -addr 0x4002913C
puts "ATXDRT_EI2_15_8"
```

```
read_register -addr 0x40029140
puts "ATXDRT_EI2_20_16"
read_register -addr 0x40029144
puts "OVERRIDE_CALIB"
read_register -addr 0x40029148
puts "FORCE_ATXDRR_7_0"
read_register -addr 0x4002914C
puts "FORCE_ATXDRR_15_8"
read_register -addr 0x40029150
puts "FORCE_ATXDRR_20_16"
read_register -addr 0x40029154
puts "FORCE_ATXDRP_7_0"
read_register -addr 0x40029158
puts "FORCE_ATXDRP_15_8"
read_register -addr 0x4002915C
puts "FORCE_ATXDRP_20_16"
read_register -addr 0x40029160
puts "FORCE_ATXDRA_7_0"
read_register -addr 0x40029164
puts "FORCE_ATXDRA_15_8"
read_register -addr 0x40029168
puts "FORCE_ATXDRA_20_16"
read_register -addr 0x4002916C
puts "FORCE_ATXDRT_7_0"
read_register -addr 0x40029170
puts "FORCE_ATXDRT_15_8"
read_register -addr 0x40029174
puts "FORCE_ATXDRT_20_16"
read_register -addr 0x40029178
puts "RXD_OFFSET_CALIB_RESULT"
read_register -addr 0x4002917C
puts "RXT_OFFSET_CALIB_RESULT"
read_register -addr 0x40029180
puts "SCHMITT_TRIG_CALIB_RESULT"
read_register -addr 0x40029184
puts "FORCE_RXD_OFFSET_CALIB"
read_register -addr 0x40029188
puts "FORCE_RXT_OFFSET_CALIB"
```

```
read_register -addr 0x4002918C
puts "FORCE_SCHMITT_TRIG_CALIB"
read_register -addr 0x40029190
puts "PRBS_CTRL"
read_register -addr 0x40029194
puts "PRBS_ERRCNT"
read_register -addr 0x40029198
puts "PHY_RESET_OVERRIDE"
read_register -addr 0x4002919C
puts "PHY_POWER_OVERRIDE"
read_register -addr 0x400291a0
puts "CUSTOM_PATTERN_7_0"
read_register -addr 0x400291a4
puts "CUSTOM_PATTERN_15_8"
read_register -addr 0x400291a8
puts "CUSTOM_PATTERN_23_16"
read_register -addr 0x400291aC
puts "CUSTOM_PATTERN_31_24"
read_register -addr 0x400291b0
puts "CUSTOM_PATTERN_39_32"
read_register -addr 0x400291b4
puts "CUSTOM_PATTERN_47_40"
read_register -addr 0x400291b8
puts "CUSTOM_PATTERN55_48"
read_register -addr 0x400291bC
puts "CUSTOM_PATTERN_63_56"
read_register -addr 0x400291C0
puts "CUSTOM_PATTERN_71_64"
read_register -addr 0x400291C4
puts "CUSTOM_PATTERN_79_72"
read_register -addr 0x400291C8
puts "CUSTOM_PATTERN_CTRL"
read_register -addr 0x400291CC
puts "CUSTOM_PATTERN_STATUS"
read_register -addr 0x400291d0
puts "PCS_LOOPBBACK_CTRL"
read_register -addr 0x400291d4
puts "GEN1_TX_PLL_CCP"
```

```
read_register -addr 0x400291d8
puts "GEN1_RX_PLL_CCP"
read_register -addr 0x400291dC
puts "GEN2_TX_PLL_CCP"
read_register -addr 0x400291e0
puts "GEN2_RX_PLL_CCP"
read_register -addr 0x400291e4
puts "CDR_PLL_MANUAL_CR"
read_register -addr 0x40029200
puts "UPDATE_SETTINGS"
read_register -addr 0x40029280
puts "PRBS_ERR_CYC_FIRST_7_0"
read_register -addr 0x40029284
puts "PRBS_ERR_CYC_FIRST_15_8"
read_register -addr 0x40029288
puts "PRBS_ERR_CYC_FIRST_23_16"
read_register -addr 0x4002928C
puts "PRBS_ERR_CYC_FIRST_31_24"
read_register -addr 0x40029290
puts "PRBS_ERR_CYC_FIRST_39_32"
read_register -addr 0x40029294
puts "PRBS_ERR_CYC_FIRST_47_40"
read_register -addr 0x40029298
puts "PRBS_ERR_CYC_FIRST_49_48"
read_register -addr 0x400292a0
puts "PRBS_ERR_CYC_LAST_7_0"
read_register -addr 0x400292a4
puts "PRBS_ERR_CYC_LAST_15_8"
read_register -addr 0x400292a8
puts "PRBS_ERR_CYC_LAST_23_16"
read_register -addr 0x400292aC
puts "PRBS_ERR_CYC_LAST_31_24"
read_register -addr 0x400292b0
puts "PRBS_ERR_CYC_LAST_39_32"
read_register -addr 0x400292b4
puts "PRBS_ERR_CYC_LAST_47_40"
read_register -addr 0x400292b8
puts "PRBS_ERR_CYC_LAST_49_48"
```

```
}

if {$dumpLane1Registers == 1} {

puts "***********************************"
# set CONFIG_REG_LANE_SEL for this lane
set lane1PhyMode [expr { ($phyMode1Val & 255) | 512 }]
sCan [format %x $lane1PhyMode] %s lane1PhyMode
write_register -addr 0x4002a028 -val $lane1PhyMode
puts "Serdes lane1 registers"

read_register -addr 0x40029400
puts "CR0"
read_register -addr 0x40029404
puts "ERRCNT_DEC"
read_register -addr 0x40029408
puts "RXIDLE_MAX_ERRCNT_THR"
read_register -addr 0x4002940C
puts "IMPED_RATIO"
read_register -addr 0x40029410
puts "PLL_F_PCLK_RATIO"
read_register -addr 0x40029414
puts "PLL_M_N"
read_register -addr 0x40029418
puts "CNT250NS_MAX"
read_register -addr 0x4002941C
puts "RE_AMP_RATIO"
read_register -addr 0x40029420
puts "RE_CUT_RATIO"
read_register -addr 0x40029424
puts "TX_AMP_RATIO"
read_register -addr 0x40029428
puts "TX_PST_RATIO"
read_register -addr 0x4002942C
puts "TX_PRE_RATIO"
read_register -addr 0x40029430
puts "ENDCALIB_MAX"
read_register -addr 0x40029434
```

```
puts "CALIB_STABILITY_COUNT"
read_register -addr 0x40029438
puts "POWER_DOWN"
read_register -addr 0x4002943C
puts "RX_OFFSET_COUNT"
read_register -addr 0x40029440
puts "PLL_F_PCLK_RATIO_5GBPS"
read_register -addr 0x40029444
puts "PLL_M_N_5GBPS"
read_register -addr 0x40029448
puts "CNT250NS_MAX_5GBPS"
read_register -addr 0x40029450
puts "TX_PST_RATIO_DEEMP0_FULL"
read_register -addr 0x40029454
puts "TX_PRE_RATIO_DEEMP0_FULL"
read_register -addr 0x40029458
puts "TX_PST_RATIO_DEEMP1_FULL"
read_register -addr 0x4002945C
puts "TX_PRE_RATIO_DEEMP1_FULL"
read_register -addr 0x40029460
puts "TX_AMP_RATIO_MARGIN0_FULL"
read_register -addr 0x40029464
puts "TX_AMP_RATIO_MARGIN1_FULL"
read_register -addr 0x40029468
puts "TX_AMP_RATIO_MARGIN2_FULL"
read_register -addr 0x4002946C
puts "TX_AMP_RATIO_MARGIN3_FULL"
read_register -addr 0x40029470
puts "TX_AMP_RATIO_MARGIN4_FULL"
read_register -addr 0x40029474
puts "TX_AMP_RATIO_MARGIN5_FULL"
read_register -addr 0x40029478
puts "TX_AMP_RATIO_MARGIN6_FULL"
read_register -addr 0x4002947C
puts "TX_AMP_RATIO_MARGIN7_FULL"
read_register -addr 0x40029480
puts "RE_AMP_RATIO_DEEMP0"
read_register -addr 0x40029484
```

```
puts "RE_CUT_RATIO_DEEMP0"
read_register -addr 0x40029488
puts "RE_AMP_RATIO_DEEMP1"
read_register -addr 0x4002948C
puts "RE_CUT_RATIO_DEEMP1"
read_register -addr 0x40029490
puts "TX_PST_RATIO_DEEMP0_HALF"
read_register -addr 0x40029494
puts "TX_PRE_RATIO_DEEMP0_HALF"
read_register -addr 0x40029498
puts "TX_PST_RATIO_DEEMP1_HALF"
read_register -addr 0x4002949C
puts "TX_PRE_RATIO_DEEMP1_HALF"
read_register -addr 0x400294a0
puts "TX_AMP_RATIO_MARGIN0_HALF"
read_register -addr 0x400294a4
puts "TX_AMP_RATIO_MARGIN1_HALF"
read_register -addr 0x400294a8
puts "TX_AMP_RATIO_MARGIN2_HALF"
read_register -addr 0x400294aC
puts "TX_AMP_RATIO_MARGIN3_HALF"
read_register -addr 0x400294b0
puts "TX_AMP_RATIO_MARGIN4_HALF"
read_register -addr 0x400294b4
puts "TX_AMP_RATIO_MARGIN5_HALF"
read_register -addr 0x400294b8
puts "TX_AMP_RATIO_MARGIN6_HALF"
read_register -addr 0x400294bC
puts "TX_AMP_RATIO_MARGIN7_HALF"
read_register -addr 0x400294C0
puts "PMA_STATUS"
read_register -addr 0x400294C4
puts "TX_SWEEP_CENTER"
read_register -addr 0x400294C8
puts "RX_SWEEP_CENTER"
read_register -addr 0x400294CC
puts "RE_SWEEP_CENTER"
read_register -addr 0x400294d0
```

```
puts "ATXDRR_7_0"
read_register -addr 0x400294d4
puts "ATXDRR_14_8"
read_register -addr 0x400294d8
puts "ATXDRP_DYN_7_0"
read_register -addr 0x400294dC
puts "ATXDRP_DYN_15_8"
read_register -addr 0x400294e0
puts "ATXDRP_DYN_20_16"
read_register -addr 0x400294e4
puts "ATXDRA_DYN_7_0"
read_register -addr 0x400294e8
puts "ATXDRA_DYN_15_8"
read_register -addr 0x400294eC
puts "ATXDRA_DYN_20_16"
read_register -addr 0x400294f0
puts "ATXDRT_DYN_7_0"
read_register -addr 0x400294f4
puts "ATXDRT_DYN_15_8"
read_register -addr 0x400294f8
puts "ATXDRT_DYN_20_16"
read_register -addr 0x400294fC
puts "ATXDRP_EI1_7_0"
read_register -addr 0x40029500
puts "ATXDRP_EI1_15_8"
read_register -addr 0x40029504
puts "ATXDRP_EI1_20_16"
read_register -addr 0x40029508
puts "ATXDRA_EI1_7_0"
read_register -addr 0x4002950C
puts "ATXDRA_EI1_15_8"
read_register -addr 0x40029510
puts "ATXDRA_EI1_20_16"
read_register -addr 0x40029514
puts "ATXDRT_EI1_7_0"
read_register -addr 0x40029518
puts "ATXDRT_EI1_15_8"
read_register -addr 0x4002951C
```

```
puts "ATXDRT_EI1_20_16"
read_register -addr 0x40029520
puts "ATXDRP_EI2_7_0"
read_register -addr 0x40029524
puts "ATXDRP_EI2_15_8"
read_register -addr 0x40029528
puts "ATXDRP_EI2_20_16"
read_register -addr 0x4002952C
puts "ATXDRA_EI2_7_0"
read_register -addr 0x40029530
puts "ATXDRA_EI2_15_8"
read_register -addr 0x40029534
puts "ATXDRA_EI2_20_16"
read_register -addr 0x40029538
puts "ATXDRT_EI2_7_0"
read_register -addr 0x4002953C
puts "ATXDRT_EI2_15_8"
read_register -addr 0x40029540
puts "ATXDRT_EI2_20_16"
read_register -addr 0x40029544
puts "OVERRIDE_CALIB"
read_register -addr 0x40029548
puts "FORCE_ATXDRR_7_0"
read_register -addr 0x4002954C
puts "FORCE_ATXDRR_15_8"
read_register -addr 0x40029550
puts "FORCE_ATXDRR_20_16"
read_register -addr 0x40029554
puts "FORCE_ATXDRP_7_0"
read_register -addr 0x40029558
puts "FORCE_ATXDRP_15_8"
read_register -addr 0x4002955C
puts "FORCE_ATXDRP_20_16"
read_register -addr 0x40029560
puts "FORCE_ATXDRA_7_0"
read_register -addr 0x40029564
puts "FORCE_ATXDRA_15_8"
read_register -addr 0x40029568
```

```
puts "FORCE_ATXDRA_20_16"
read_register -addr 0x4002956C
puts "FORCE_ATXDRT_7_0"
read_register -addr 0x40029570
puts "FORCE_ATXDRT_15_8"
read_register -addr 0x40029574
puts "FORCE_ATXDRT_20_16"
read_register -addr 0x40029578
puts "RXD_OFFSET_CALIB_RESULT"
read_register -addr 0x4002957C
puts "RXT_OFFSET_CALIB_RESULT"
read_register -addr 0x40029580
puts "SCHMITT_TRIG_CALIB_RESULT"
read_register -addr 0x40029584
puts "FORCE_RXD_OFFSET_CALIB"
read_register -addr 0x40029588
puts "FORCE_RXT_OFFSET_CALIB"
read_register -addr 0x4002958C
puts "FORCE_SCHMITT_TRIG_CALIB"
read_register -addr 0x40029590
puts "PRBS_CTRL"
read_register -addr 0x40029594
puts "PRBS_ERRCNT"
read_register -addr 0x40029598
puts "PHY_RESET_OVERRIDE"
read_register -addr 0x4002959C
puts "PHY_POWER_OVERRIDE"
read_register -addr 0x400295a0
puts "CUSTOM_PATTERN_7_0"
read_register -addr 0x400295a4
puts "CUSTOM_PATTERN_15_8"
read_register -addr 0x400295a8
puts "CUSTOM_PATTERN_23_16"
read_register -addr 0x400295aC
puts "CUSTOM_PATTERN_31_24"
read_register -addr 0x400295b0
puts "CUSTOM_PATTERN_39_32"
read_register -addr 0x400295b4
```

```
puts "CUSTOM_PATTERN_47_40"
read_register -addr 0x400295b8
puts "CUSTOM_PATTERN55_48"
read_register -addr 0x400295bC
puts "CUSTOM_PATTERN_63_56"
read_register -addr 0x400295C0
puts "CUSTOM_PATTERN_71_64"
read_register -addr 0x400295C4
puts "CUSTOM_PATTERN_79_72"
read_register -addr 0x400295C8
puts "CUSTOM_PATTERN_CTRL"
read_register -addr 0x400295CC
puts "CUSTOM_PATTERN_STATUS"
read_register -addr 0x400295d0
puts "PCS_LOOPBBACK_CTRL"
read_register -addr 0x400295d4
puts "GEN1_TX_PLL_CCP"
read_register -addr 0x400295d8
puts "GEN1_RX_PLL_CCP"
read_register -addr 0x400295dC
puts "GEN2_TX_PLL_CCP"
read_register -addr 0x400295e0
puts "GEN2_RX_PLL_CCP"
read_register -addr 0x400295e4
puts "CDR_PLL_MANUAL_CR"
read_register -addr 0x40029600
puts "UPDATE_SETTINGS"
read_register -addr 0x40029680
puts "PRBS_ERR_CYC_FIRST_7_0"
read_register -addr 0x40029684
puts "PRBS_ERR_CYC_FIRST_15_8"
read_register -addr 0x40029688
puts "PRBS_ERR_CYC_FIRST_23_16"
read_register -addr 0x4002968C
puts "PRBS_ERR_CYC_FIRST_31_24"
read_register -addr 0x40029690
puts "PRBS_ERR_CYC_FIRST_39_32"
read_register -addr 0x40029694
```

```
puts "PRBS_ERR_CYC_FIRST_47_40"
read_register -addr 0x40029698
puts "PRBS_ERR_CYC_FIRST_49_48"
read_register -addr 0x400296a0
puts "PRBS_ERR_CYC_LAST_7_0"
read_register -addr 0x400296a4
puts "PRBS_ERR_CYC_LAST_15_8"
read_register -addr 0x400296a8
puts "PRBS_ERR_CYC_LAST_23_16"
read_register -addr 0x400296aC
puts "PRBS_ERR_CYC_LAST_31_24"
read_register -addr 0x400296b0
puts "PRBS_ERR_CYC_LAST_39_32"
read_register -addr 0x400296b4
puts "PRBS_ERR_CYC_LAST_47_40"
read_register -addr 0x400296b8
puts "PRBS_ERR_CYC_LAST_49_48"
}

if {$dumpLane2Registers == 1} {

puts "***********************************"
# set CONFIG_REG_LANE_SEL for this lane
set lane2PhyMode [expr { ($phyMode1Val & 255) | 768 }]
sCan [format %x $lane2PhyMode] %s lane2PhyMode
write_register -addr 0x4002a028 -val $lane2PhyMode
puts "Serdes lane2 registers"

read_register -addr 0x40029800
puts "CR0"
read_register -addr 0x40029804
puts "ERRCNT_DEC"
read_register -addr 0x40029808
puts "RXIDLE_MAX_ERRCNT_THR"
read_register -addr 0x4002980C
puts "IMPED_RATIO"
read_register -addr 0x40029810
puts "PLL_F_PCLK_RATIO"
```

```
read_register -addr 0x40029814
puts "PLL_M_N"
read_register -addr 0x40029818
puts "CNT250NS_MAX"
read_register -addr 0x4002981C
puts "RE_AMP_RATIO"
read_register -addr 0x40029820
puts "RE_CUT_RATIO"
read_register -addr 0x40029824
puts "TX_AMP_RATIO"
read_register -addr 0x40029828
puts "TX_PST_RATIO"
read_register -addr 0x4002982C
puts "TX_PRE_RATIO"
read_register -addr 0x40029830
puts "ENDCALIB_MAX"
read_register -addr 0x40029834
puts "CALIB_STABILITY_COUNT"
read_register -addr 0x40029838
puts "POWER_DOWN"
read_register -addr 0x4002983C
puts "RX_OFFSET_COUNT"
read_register -addr 0x40029840
puts "PLL_F_PCLK_RATIO_5GBPS"
read_register -addr 0x40029844
puts "PLL_M_N_5GBPS"
read_register -addr 0x40029848
puts "CNT250NS_MAX_5GBPS"
read_register -addr 0x40029850
puts "TX_PST_RATIO_DEEMP0_FULL"
read_register -addr 0x40029854
puts "TX_PRE_RATIO_DEEMP0_FULL"
read_register -addr 0x40029858
puts "TX_PST_RATIO_DEEMP1_FULL"
read_register -addr 0x4002985C
puts "TX_PRE_RATIO_DEEMP1_FULL"
read_register -addr 0x40029860
puts "TX_AMP_RATIO_MARGIN0_FULL"
```

```
read_register -addr 0x40029864
puts "TX_AMP_RATIO_MARGIN1_FULL"
read_register -addr 0x40029868
puts "TX_AMP_RATIO_MARGIN2_FULL"
read_register -addr 0x4002986C
puts "TX_AMP_RATIO_MARGIN3_FULL"
read_register -addr 0x40029870
puts "TX_AMP_RATIO_MARGIN4_FULL"
read_register -addr 0x40029874
puts "TX_AMP_RATIO_MARGIN5_FULL"
read_register -addr 0x40029878
puts "TX_AMP_RATIO_MARGIN6_FULL"
read_register -addr 0x4002987C
puts "TX_AMP_RATIO_MARGIN7_FULL"
read_register -addr 0x40029880
puts "RE_AMP_RATIO_DEEMP0"
read_register -addr 0x40029884
puts "RE_CUT_RATIO_DEEMP0"
read_register -addr 0x40029888
puts "RE_AMP_RATIO_DEEMP1"
read_register -addr 0x4002988C
puts "RE_CUT_RATIO_DEEMP1"
read_register -addr 0x40029890
puts "TX_PST_RATIO_DEEMP0_HALF"
read_register -addr 0x40029894
puts "TX_PRE_RATIO_DEEMP0_HALF"
read_register -addr 0x40029898
puts "TX_PST_RATIO_DEEMP1_HALF"
read_register -addr 0x4002989C
puts "TX_PRE_RATIO_DEEMP1_HALF"
read_register -addr 0x400298a0
puts "TX_AMP_RATIO_MARGIN0_HALF"
read_register -addr 0x400298a4
puts "TX_AMP_RATIO_MARGIN1_HALF"
read_register -addr 0x400298a8
puts "TX_AMP_RATIO_MARGIN2_HALF"
read_register -addr 0x400298aC
puts "TX_AMP_RATIO_MARGIN3_HALF"
```

```
read_register -addr 0x400298b0
puts "TX_AMP_RATIO_MARGIN4_HALF"
read_register -addr 0x400298b4
puts "TX_AMP_RATIO_MARGIN5_HALF"
read_register -addr 0x400298b8
puts "TX_AMP_RATIO_MARGIN6_HALF"
read_register -addr 0x400298bC
puts "TX_AMP_RATIO_MARGIN7_HALF"
read_register -addr 0x400298C0
puts "PMA_STATUS"
read_register -addr 0x400298C4
puts "TX_SWEEP_CENTER"
read_register -addr 0x400298C8
puts "RX_SWEEP_CENTER"
read_register -addr 0x400298CC
puts "RE_SWEEP_CENTER"
read_register -addr 0x400298d0
puts "ATXDRR_7_0"
read_register -addr 0x400298d4
puts "ATXDRR_14_8"
read_register -addr 0x400298d8
puts "ATXDRP_DYN_7_0"
read_register -addr 0x400298dC
puts "ATXDRP_DYN_15_8"
read_register -addr 0x400298e0
puts "ATXDRP_DYN_20_16"
read_register -addr 0x400298e4
puts "ATXDRA_DYN_7_0"
read_register -addr 0x400298e8
puts "ATXDRA_DYN_15_8"
read_register -addr 0x400298eC
puts "ATXDRA_DYN_20_16"
read_register -addr 0x400298f0
puts "ATXDRT_DYN_7_0"
read_register -addr 0x400298f4
puts "ATXDRT_DYN_15_8"
read_register -addr 0x400298f8
puts "ATXDRT_DYN_20_16"
```

```
read_register -addr 0x400298fC
puts "ATXDRP_EI1_7_0"
read_register -addr 0x40029900
puts "ATXDRP_EI1_15_8"
read_register -addr 0x40029904
puts "ATXDRP_EI1_20_16"
read_register -addr 0x40029908
puts "ATXDRA_EI1_7_0"
read_register -addr 0x4002990C
puts "ATXDRA_EI1_15_8"
read_register -addr 0x40029910
puts "ATXDRA_EI1_20_16"
read_register -addr 0x40029914
puts "ATXDRT_EI1_7_0"
read_register -addr 0x40029918
puts "ATXDRT_EI1_15_8"
read_register -addr 0x4002991C
puts "ATXDRT_EI1_20_16"
read_register -addr 0x40029920
puts "ATXDRP_EI2_7_0"
read_register -addr 0x40029924
puts "ATXDRP_EI2_15_8"
read_register -addr 0x40029928
puts "ATXDRP_EI2_20_16"
read_register -addr 0x4002992C
puts "ATXDRA_EI2_7_0"
read_register -addr 0x40029930
puts "ATXDRA_EI2_15_8"
read_register -addr 0x40029934
puts "ATXDRA_EI2_20_16"
read_register -addr 0x40029938
puts "ATXDRT_EI2_7_0"
read_register -addr 0x4002993C
puts "ATXDRT_EI2_15_8"
read_register -addr 0x40029940
puts "ATXDRT_EI2_20_16"
read_register -addr 0x40029944
puts "OVERRIDE_CALIB"
```

```
read_register -addr 0x40029948
puts "FORCE_ATXDRR_7_0"
read_register -addr 0x4002994C
puts "FORCE_ATXDRR_15_8"
read_register -addr 0x40029950
puts "FORCE_ATXDRR_20_16"
read_register -addr 0x40029954
puts "FORCE_ATXDRP_7_0"
read_register -addr 0x40029958
puts "FORCE_ATXDRP_15_8"
read_register -addr 0x4002995C
puts "FORCE_ATXDRP_20_16"
read_register -addr 0x40029960
puts "FORCE_ATXDRA_7_0"
read_register -addr 0x40029964
puts "FORCE_ATXDRA_15_8"
read_register -addr 0x40029968
puts "FORCE_ATXDRA_20_16"
read_register -addr 0x4002996C
puts "FORCE_ATXDRT_7_0"
read_register -addr 0x40029970
puts "FORCE_ATXDRT_15_8"
read_register -addr 0x40029974
puts "FORCE_ATXDRT_20_16"
read_register -addr 0x40029978
puts "RXD_OFFSET_CALIB_RESULT"
read_register -addr 0x4002997C
puts "RXT_OFFSET_CALIB_RESULT"
read_register -addr 0x40029980
puts "SCHMITT_TRIG_CALIB_RESULT"
read_register -addr 0x40029984
puts "FORCE_RXD_OFFSET_CALIB"
read_register -addr 0x40029988
puts "FORCE_RXT_OFFSET_CALIB"
read_register -addr 0x4002998C
puts "FORCE_SCHMITT_TRIG_CALIB"
read_register -addr 0x40029990
puts "PRBS_CTRL"
```

```
read_register -addr 0x40029994
puts "PRBS_ERRCNT"
read_register -addr 0x40029998
puts "PHY_RESET_OVERRIDE"
read_register -addr 0x4002999C
puts "PHY_POWER_OVERRIDE"
read_register -addr 0x400299a0
puts "CUSTOM_PATTERN_7_0"
read_register -addr 0x400299a4
puts "CUSTOM_PATTERN_15_8"
read_register -addr 0x400299a8
puts "CUSTOM_PATTERN_23_16"
read_register -addr 0x400299aC
puts "CUSTOM_PATTERN_31_24"
read_register -addr 0x400299b0
puts "CUSTOM_PATTERN_39_32"
read_register -addr 0x400299b4
puts "CUSTOM_PATTERN_47_40"
read_register -addr 0x400299b8
puts "CUSTOM_PATTERN55_48"
read_register -addr 0x400299bC
puts "CUSTOM_PATTERN_63_56"
read_register -addr 0x400299C0
puts "CUSTOM_PATTERN_71_64"
read_register -addr 0x400299C4
puts "CUSTOM_PATTERN_79_72"
read_register -addr 0x400299C8
puts "CUSTOM_PATTERN_CTRL"
read_register -addr 0x400299CC
puts "CUSTOM_PATTERN_STATUS"
read_register -addr 0x400299d0
puts "PCS_LOOPBBACK_CTRL"
read_register -addr 0x400299d4
puts "GEN1_TX_PLL_CCP"
read_register -addr 0x400299d8
puts "GEN1_RX_PLL_CCP"
read_register -addr 0x400299dC
puts "GEN2_TX_PLL_CCP"
```

```
read_register -addr 0x400299e0
puts "GEN2_RX_PLL_CCP"
read_register -addr 0x400299e4
puts "CDR_PLL_MANUAL_CR"
read_register -addr 0x40029a00
puts "UPDATE_SETTINGS"
read_register -addr 0x40029a80
puts "PRBS_ERR_CYC_FIRST_7_0"
read_register -addr 0x40029a84
puts "PRBS_ERR_CYC_FIRST_15_8"
read_register -addr 0x40029a88
puts "PRBS_ERR_CYC_FIRST_23_16"
read_register -addr 0x40029a8C
puts "PRBS_ERR_CYC_FIRST_31_24"
read_register -addr 0x40029a90
puts "PRBS_ERR_CYC_FIRST_39_32"
read_register -addr 0x40029a94
puts "PRBS_ERR_CYC_FIRST_47_40"
read_register -addr 0x40029a98
puts "PRBS_ERR_CYC_FIRST_49_48"
read_register -addr 0x40029aa0
puts "PRBS_ERR_CYC_LAST_7_0"
read_register -addr 0x40029aa4
puts "PRBS_ERR_CYC_LAST_15_8"
read_register -addr 0x40029aa8
puts "PRBS_ERR_CYC_LAST_23_16"
read_register -addr 0x40029aaC
puts "PRBS_ERR_CYC_LAST_31_24"
read_register -addr 0x40029ab0
puts "PRBS_ERR_CYC_LAST_39_32"
read_register -addr 0x40029ab4
puts "PRBS_ERR_CYC_LAST_47_40"
read_register -addr 0x40029ab8
puts "PRBS_ERR_CYC_LAST_49_48"
}

if {$dumpLane3Registers == 1} {
```

```
puts "***********************************"
# set CONFIG_REG_LANE_SEL for this lane
set lane3PhyMode [expr { ($phyMode1Val & 255) | 1024 }]
sCan [format %x $lane3PhyMode] %s lane3PhyMode
write_register -addr 0x4002a028 -val $lane3PhyMode
puts "Serdes lane3 registers"

read_register -addr 0x40029C00
puts "CR0"
read_register -addr 0x40029C04
puts "ERRCNT_DEC"
read_register -addr 0x40029C08
puts "RXIDLE_MAX_ERRCNT_THR"
read_register -addr 0x40029C0C
puts "IMPED_RATIO"
read_register -addr 0x40029C10
puts "PLL_F_PCLK_RATIO"
read_register -addr 0x40029C14
puts "PLL_M_N"
read_register -addr 0x40029C18
puts "CNT250NS_MAX"
read_register -addr 0x40029C1C
puts "RE_AMP_RATIO"
read_register -addr 0x40029C20
puts "RE_CUT_RATIO"
read_register -addr 0x40029C24
puts "TX_AMP_RATIO"
read_register -addr 0x40029C28
puts "TX_PST_RATIO"
read_register -addr 0x40029C2C
puts "TX_PRE_RATIO"
read_register -addr 0x40029C30
puts "ENDCALIB_MAX"
read_register -addr 0x40029C34
puts "CALIB_STABILITY_COUNT"
read_register -addr 0x40029C38
puts "POWER_DOWN"
read_register -addr 0x40029C3C
```

```
puts "RX_OFFSET_COUNT"
read_register -addr 0x40029C40
puts "PLL_F_PCLK_RATIO_5GBPS"
read_register -addr 0x40029C44
puts "PLL_M_N_5GBPS"
read_register -addr 0x40029C48
puts "CNT250NS_MAX_5GBPS"
read_register -addr 0x40029C50
puts "TX_PST_RATIO_DEEMP0_FULL"
read_register -addr 0x40029C54
puts "TX_PRE_RATIO_DEEMP0_FULL"
read_register -addr 0x40029C58
puts "TX_PST_RATIO_DEEMP1_FULL"
read_register -addr 0x40029C5C
puts "TX_PRE_RATIO_DEEMP1_FULL"
read_register -addr 0x40029C60
puts "TX_AMP_RATIO_MARGIN0_FULL"
read_register -addr 0x40029C64
puts "TX_AMP_RATIO_MARGIN1_FULL"
read_register -addr 0x40029C68
puts "TX_AMP_RATIO_MARGIN2_FULL"
read_register -addr 0x40029C6C
puts "TX_AMP_RATIO_MARGIN3_FULL"
read_register -addr 0x40029C70
puts "TX_AMP_RATIO_MARGIN4_FULL"
read_register -addr 0x40029C74
puts "TX_AMP_RATIO_MARGIN5_FULL"
read_register -addr 0x40029C78
puts "TX_AMP_RATIO_MARGIN6_FULL"
read_register -addr 0x40029C7C
puts "TX_AMP_RATIO_MARGIN7_FULL"
read_register -addr 0x40029C80
puts "RE_AMP_RATIO_DEEMP0"
read_register -addr 0x40029C84
puts "RE_CUT_RATIO_DEEMP0"
read_register -addr 0x40029C88
puts "RE_AMP_RATIO_DEEMP1"
read_register -addr 0x40029C8C
```

```
puts "RE_CUT_RATIO_DEEMP1"
read_register -addr 0x40029C90
puts "TX_PST_RATIO_DEEMP0_HALF"
read_register -addr 0x40029C94
puts "TX_PRE_RATIO_DEEMP0_HALF"
read_register -addr 0x40029C98
puts "TX_PST_RATIO_DEEMP1_HALF"
read_register -addr 0x40029C9C
puts "TX_PRE_RATIO_DEEMP1_HALF"
read_register -addr 0x40029Ca0
puts "TX_AMP_RATIO_MARGIN0_HALF"
read_register -addr 0x40029Ca4
puts "TX_AMP_RATIO_MARGIN1_HALF"
read_register -addr 0x40029Ca8
puts "TX_AMP_RATIO_MARGIN2_HALF"
read_register -addr 0x40029CaC
puts "TX_AMP_RATIO_MARGIN3_HALF"
read_register -addr 0x40029Cb0
puts "TX_AMP_RATIO_MARGIN4_HALF"
read_register -addr 0x40029Cb4
puts "TX_AMP_RATIO_MARGIN5_HALF"
read_register -addr 0x40029Cb8
puts "TX_AMP_RATIO_MARGIN6_HALF"
read_register -addr 0x40029CbC
puts "TX_AMP_RATIO_MARGIN7_HALF"
read_register -addr 0x40029CC0
puts "PMA_STATUS"
read_register -addr 0x40029CC4
puts "TX_SWEEP_CENTER"
read_register -addr 0x40029CC8
puts "RX_SWEEP_CENTER"
read_register -addr 0x40029CCC
puts "RE_SWEEP_CENTER"
read_register -addr 0x40029Cd0
puts "ATXDRR_7_0"
read_register -addr 0x40029Cd4
puts "ATXDRR_14_8"
read_register -addr 0x40029Cd8
```

```
puts "ATXDRP_DYN_7_0"
read_register -addr 0x40029CdC
puts "ATXDRP_DYN_15_8"
read_register -addr 0x40029Ce0
puts "ATXDRP_DYN_20_16"
read_register -addr 0x40029Ce4
puts "ATXDRA_DYN_7_0"
read_register -addr 0x40029Ce8
puts "ATXDRA_DYN_15_8"
read_register -addr 0x40029CeC
puts "ATXDRA_DYN_20_16"
read_register -addr 0x40029Cf0
puts "ATXDRT_DYN_7_0"
read_register -addr 0x40029Cf4
puts "ATXDRT_DYN_15_8"
read_register -addr 0x40029Cf8
puts "ATXDRT_DYN_20_16"
read_register -addr 0x40029CfC
puts "ATXDRP_EI1_7_0"
read_register -addr 0x40029d00
puts "ATXDRP_EI1_15_8"
read_register -addr 0x40029d04
puts "ATXDRP_EI1_20_16"
read_register -addr 0x40029d08
puts "ATXDRA_EI1_7_0"
read_register -addr 0x40029d0C
puts "ATXDRA_EI1_15_8"
read_register -addr 0x40029d10
puts "ATXDRA_EI1_20_16"
read_register -addr 0x40029d14
puts "ATXDRT_EI1_7_0"
read_register -addr 0x40029d18
puts "ATXDRT_EI1_15_8"
read_register -addr 0x40029d1C
puts "ATXDRT_EI1_20_16"
read_register -addr 0x40029d20
puts "ATXDRP_EI2_7_0"
read_register -addr 0x40029d24
```

```
puts "ATXDRP_EI2_15_8"
read_register -addr 0x40029d28
puts "ATXDRP_EI2_20_16"
read_register -addr 0x40029d2C
puts "ATXDRA_EI2_7_0"
read_register -addr 0x40029d30
puts "ATXDRA_EI2_15_8"
read_register -addr 0x40029d34
puts "ATXDRA_EI2_20_16"
read_register -addr 0x40029d38
puts "ATXDRT_EI2_7_0"
read_register -addr 0x40029d3C
puts "ATXDRT_EI2_15_8"
read_register -addr 0x40029d40
puts "ATXDRT_EI2_20_16"
read_register -addr 0x40029d44
puts "OVERRIDE_CALIB"
read_register -addr 0x40029d48
puts "FORCE_ATXDRR_7_0"
read_register -addr 0x40029d4C
puts "FORCE_ATXDRR_15_8"
read_register -addr 0x40029d50
puts "FORCE_ATXDRR_20_16"
read_register -addr 0x40029d54
puts "FORCE_ATXDRP_7_0"
read_register -addr 0x40029d58
puts "FORCE_ATXDRP_15_8"
read_register -addr 0x40029d5C
puts "FORCE_ATXDRP_20_16"
read_register -addr 0x40029d60
puts "FORCE_ATXDRA_7_0"
read_register -addr 0x40029d64
puts "FORCE_ATXDRA_15_8"
read_register -addr 0x40029d68
puts "FORCE_ATXDRA_20_16"
read_register -addr 0x40029d6C
puts "FORCE_ATXDRT_7_0"
read_register -addr 0x40029d70
```

```
puts "FORCE_ATXDRT_15_8"
read_register -addr 0x40029d74
puts "FORCE_ATXDRT_20_16"
read_register -addr 0x40029d78
puts "RXD_OFFSET_CALIB_RESULT"
read_register -addr 0x40029d7C
puts "RXT_OFFSET_CALIB_RESULT"
read_register -addr 0x40029d80
puts "SCHMITT_TRIG_CALIB_RESULT"
read_register -addr 0x40029d84
puts "FORCE_RXD_OFFSET_CALIB"
read_register -addr 0x40029d88
puts "FORCE_RXT_OFFSET_CALIB"
read_register -addr 0x40029d8C
puts "FORCE_SCHMITT_TRIG_CALIB"
read_register -addr 0x40029d90
puts "PRBS_CTRL"
read_register -addr 0x40029d94
puts "PRBS_ERRCNT"
read_register -addr 0x40029d98
puts "PHY_RESET_OVERRIDE"
read_register -addr 0x40029d9C
puts "PHY_POWER_OVERRIDE"
read_register -addr 0x40029da0
puts "CUSTOM_PATTERN_7_0"
read_register -addr 0x40029da4
puts "CUSTOM_PATTERN_15_8"
read_register -addr 0x40029da8
puts "CUSTOM_PATTERN_23_16"
read_register -addr 0x40029daC
puts "CUSTOM_PATTERN_31_24"
read_register -addr 0x40029db0
puts "CUSTOM_PATTERN_39_32"
read_register -addr 0x40029db4
puts "CUSTOM_PATTERN_47_40"
read_register -addr 0x40029db8
puts "CUSTOM_PATTERN55_48"
read_register -addr 0x40029dbC
```

```
puts "CUSTOM_PATTERN_63_56"
read_register -addr 0x40029dC0
puts "CUSTOM_PATTERN_71_64"
read_register -addr 0x40029dC4
puts "CUSTOM_PATTERN_79_72"
read_register -addr 0x40029dC8
puts "CUSTOM_PATTERN_CTRL"
read_register -addr 0x40029dCC
puts "CUSTOM_PATTERN_STATUS"
read_register -addr 0x40029dd0
puts "PCS_LOOPBBACK_CTRL"
read_register -addr 0x40029dd4
puts "GEN1_TX_PLL_CCP"
read_register -addr 0x40029dd8
puts "GEN1_RX_PLL_CCP"
read_register -addr 0x40029ddC
puts "GEN2_TX_PLL_CCP"
read_register -addr 0x40029de0
puts "GEN2_RX_PLL_CCP"
read_register -addr 0x40029de4
puts "CDR_PLL_MANUAL_CR"
read_register -addr 0x40029e00
puts "UPDATE_SETTINGS"
read_register -addr 0x40029e80
puts "PRBS_ERR_CYC_FIRST_7_0"
read_register -addr 0x40029e84
puts "PRBS_ERR_CYC_FIRST_15_8"
read_register -addr 0x40029e88
puts "PRBS_ERR_CYC_FIRST_23_16"
read_register -addr 0x40029e8C
puts "PRBS_ERR_CYC_FIRST_31_24"
read_register -addr 0x40029e90
puts "PRBS_ERR_CYC_FIRST_39_32"
read_register -addr 0x40029e94
puts "PRBS_ERR_CYC_FIRST_47_40"
read_register -addr 0x40029e98
puts "PRBS_ERR_CYC_FIRST_49_48"
read_register -addr 0x40029ea0
```

```
puts "PRBS_ERR_CYC_LAST_7_0"
read_register -addr 0x40029ea4
puts "PRBS_ERR_CYC_LAST_15_8"
read_register -addr 0x40029ea8
puts "PRBS_ERR_CYC_LAST_23_16"
read_register -addr 0x40029eaC
puts "PRBS_ERR_CYC_LAST_31_24"
read_register -addr 0x40029eb0
puts "PRBS_ERR_CYC_LAST_39_32"
read_register -addr 0x40029eb4
puts "PRBS_ERR_CYC_LAST_47_40"
read_register -addr 0x40029eb8
puts "PRBS_ERR_CYC_LAST_49_48"
}

if {$dumpSerdesSystemRegisters == 1} {

puts "************************************"
puts "Serdes system registers for PCI Controller 0"

read_register -addr 0x4002a000
puts "SER_PLL_CONFIG_LOW"
read_register -addr 0x4002a004
puts "SER_PLL_CONFIG_HIGH"
read_register -addr 0x4002a008
puts "SERDESIF_SOFT_RESET"
read_register -addr 0x4002a00C
puts "SER_INTERRUPT_ENABLE"
read_register -addr 0x4002a010
puts "CONFIG_AXI_AHB_BRIDGE"
read_register -addr 0x4002a014
puts "CONFIG_ECC_INTR_ENABLE"
read_register -addr 0x4002a018
puts "CONFIG_TEST_IN"
read_register -addr 0x4002a01C
puts "TEST_OUT_READ_ADDR"
read_register -addr 0x4002a020
puts "CONFIG_PCIE_PM"
```

```
read_register -addr 0x4002a024
puts "CONFIG_PHY_MODE_0"
read_register -addr 0x4002a028
puts "CONFIG_PHY_MODE_1"
read_register -addr 0x4002a02C
puts "CONFIG_PHY_MODE_2"
read_register -addr 0x4002a030
puts "CONFIG_PCIE_0"
read_register -addr 0x4002a034
puts "CONFIG_PCIE_1"
read_register -addr 0x4002a038
puts "CONFIG_PCIE_2"
read_register -addr 0x4002a03C
puts "CONFIG_PCIE_3"
read_register -addr 0x4002a040
puts "CONFIG_BAR_SIZE_0_1"
read_register -addr 0x4002a044
puts "CONFIG_BAR_SIZE_2_3"
read_register -addr 0x4002a048
puts "CONFIG_BAR_SIZE_3_4"
read_register -addr 0x4002a04C
puts "SER_CLK_STATUS"
read_register -addr 0x4002a050
puts "SER_CLK_CALIB_STATUS"
read_register -addr 0x4002a054
puts "TEST_OUT_READ_DATA"
read_register -addr 0x4002a058
puts "SER_INTERRUPT"
read_register -addr 0x4002a05C
puts "SERDESIF_INTR_STATUS"
read_register -addr 0x4002a060
puts "SER_CLK_CALIB_CONFIG"
read_register -addr 0x4002a064
puts "REFCLK_SEL"
read_register -addr 0x4002a068
puts "PCLK_SEL"
read_register -addr 0x4002a06C
puts "EPCS_RSTN_SEL"
```

```
read_register -addr 0x4002a070
puts "CHIP_ENABLES"
read_register -addr 0x4002a074
puts "SERDES_TEST_OUT"
read_register -addr 0x4002a078
puts "SERDES_FATC_RESET"
read_register -addr 0x4002a07C
puts "RC_OSC_SPLL_REFCLK_SEL"
read_register -addr 0x4002a080
puts "SPREAD_SPECTRUM_CLK"
read_register -addr 0x4002a084
puts "CONF_AXI_MSTR_WNDW_0"
read_register -addr 0x4002a088
puts "CONF_AXI_MSTR_WNDW_1"
read_register -addr 0x4002a08C
puts "CONF_AXI_MSTR_WNDW_2"
read_register -addr 0x4002a090
puts "CONF_AXI_MSTR_WNDW_3"
read_register -addr 0x4002a094
puts "CONF_AXI_SLV_WNDW_0"
read_register -addr 0x4002a098
puts "CONF_AXI_SLV_WNDW_1"
read_register -addr 0x4002a09C
puts "CONF_AXI_SLV_WNDW_2"
read_register -addr 0x4002a0a0
puts "CONF_AXI_SLV_WNDW_3"
read_register -addr 0x4002a0a4
puts "DESKEW_CONFIG"
read_register -addr 0x4002a0a8
puts "DEBUG_MODE_KEY"
read_register -addr 0x4002a0b0
puts "EXTRA_BITS"
read_register -addr 0x4002a0b4
puts "ADVCFG"
read_register -addr 0x4002a0b8
puts "ADVSTATUS"
read_register -addr 0x4002a0bC
puts "ECC_ERR_INJECT"
```

```
read_register -addr 0x4002a0C8
puts "ENHANCEMENT"


}

#Reset the Config_phy_mode_1 value to original value
write_register -addr 0x4002a028 -val $Config_phy_mode_1
```