SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools - Libero SoC v11.7

TU0530 Tutorial

Power Matters.™

Contents

Figures

Tables

1 Preface

1.1 About this Document

This tutorial describes the following topics:

- Launching SmartDebug from Libero: Accessing SmartDebug from Libero® System-on-Chip (SoC)
- View Device Status: Checking the device status
- View Flash Memory (eNVM) Content: Checking the flash memory (eNVM) content
- Debug FPGA Array: Debugging FPGA array (setting Live Probes, Active Probes, and reading and modifying fabric SRAM content)
- Probe Insertion: Post-Layout Probe Insertion
- SERDES Debug: Debugging SERDES blocks

1.2 Intended Audience

This tutorial is intended for:

- FPGA designers
- System-level designers

1.3 References

1.3.1 Microsemi Publications

Refer to the following web page for a complete and up-to-date listing of the SmartFusion2 device documentation: http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2

Refer to the following web page for a complete and up-to-date listing of the IGLOO2 device documentation: http://www.microsemi.com/products/fpga-soc/fpga/igloo2-fpga

- *SmartDebug for Software v11.7 User's Guide*
- *FPGA On-Chip Debug Tools*
- *IGLOO2 FPGA Evaluation Kit*
- *SmartFusion2 Security Evaluation Kit Board*
- *http://soc.microsemi.com/kb/article.aspx?id=SL5636*
- *UG0451: SmartFusion2 and IGLOO2 Programming User Guide*
- *UG0447: SmartFusion2 and IGLOO2 FPGA High Speed Serial Interfaces User Guide*

1.3.2 Others

Pasternack® PE39429-12 technical datasheet:

Pasternack Industries part number PE39429-12

1 SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools

1.1 Introduction

Design debug is a critical phase of the field programmable gate array (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) field programmable gate array (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.

1.2 Design Requirements

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware Requirements	
SMA Male-to-SMA Male Precision Cables, such as Pasternack Industries part number PE39429-12 (or equivalent)	Optionally recommended for evaluation board SERDES testing.
IGLOO2 Evaluation Kit or *SmartFusion2 Security Evaluation Kit (M2S090TS-FGG484)*	Rev D or later
Software Requirements	
Libero SoC software	v11.7
FlashPro4	v11.7

1.2.1 Reference Documents

For more information on using SmartDebug, see the *SmartDebug for Software v11.7 User's Guide*.

1.2.2 Project Files

Extract the *http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_tu0530_liberov11p7_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)

1.3 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 Figure 1.

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. 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. 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**

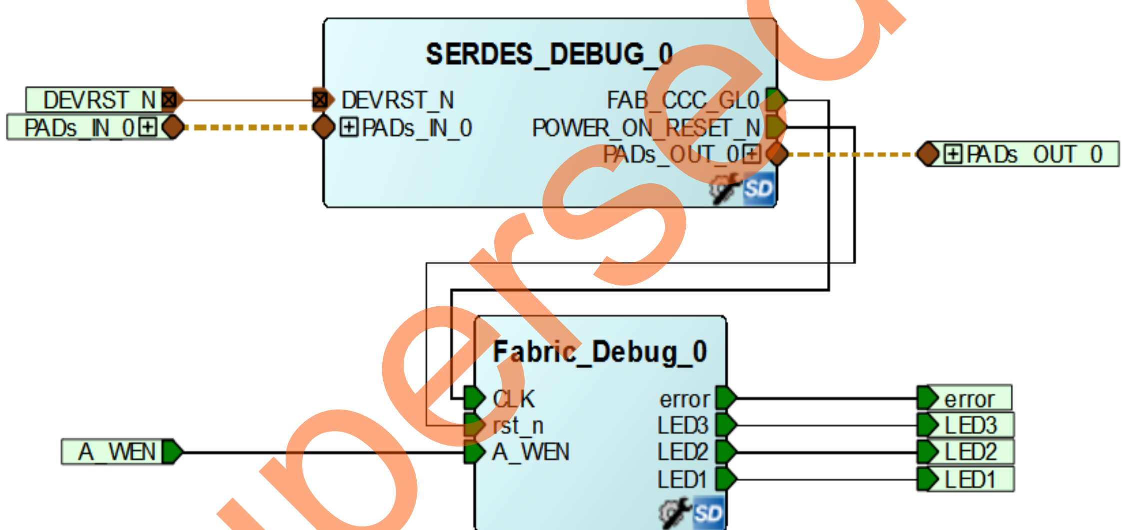


Figure 2 • **SERDES_Debug Overall Design Blocks (IGLOO2 Design Block)**

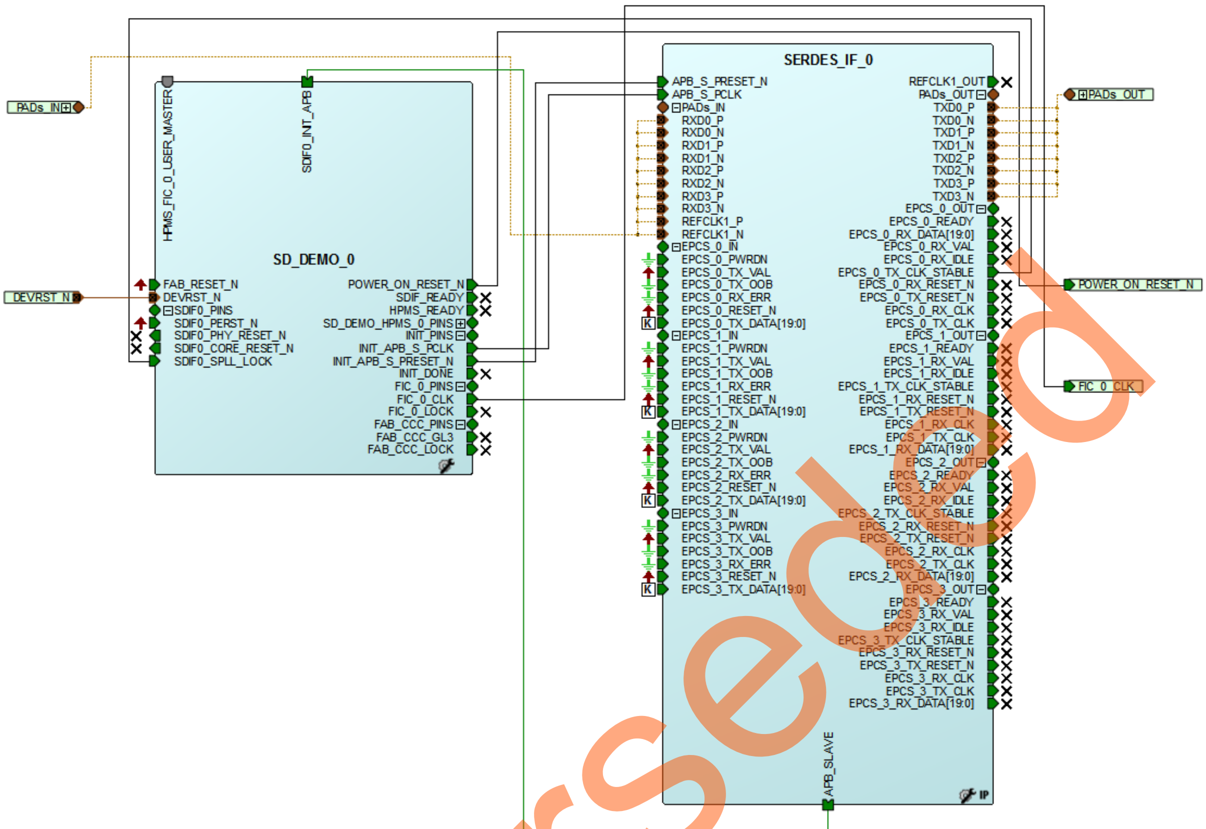
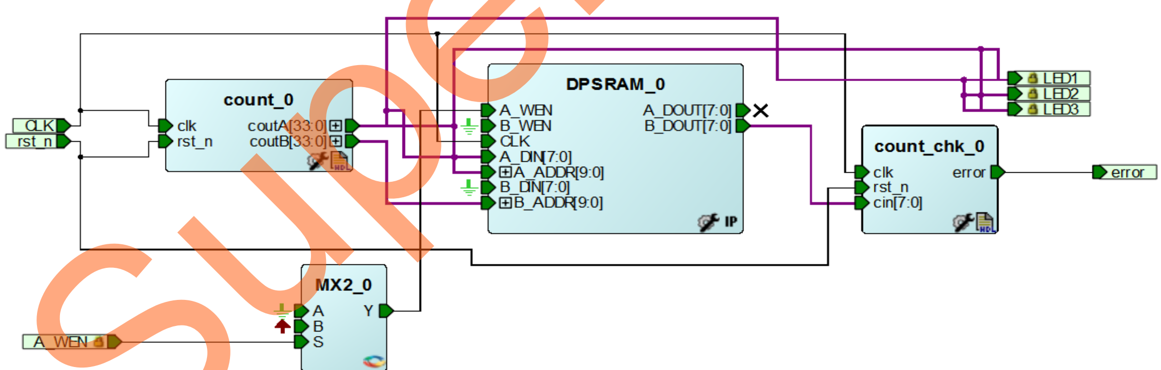


Figure 3 • **Fabric_Debug Overall Design Blocks (IGLOO2 Design Block)**



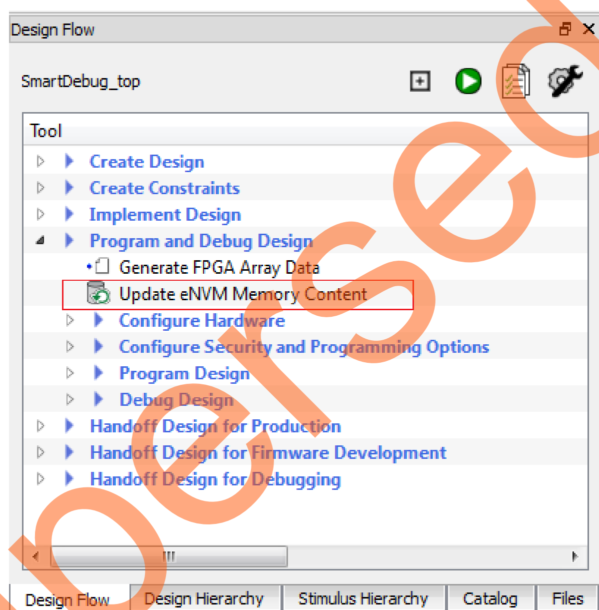
A_WEN is used to pause the write operation to the SRAM while demonstrating the SmartDebug write to the SRAM capability. It is assigned to SW2 on the board. When SW2 is pressed, the write operation from the counter pauses and does not overwrite the SmartDebug write into the SRAM.

1.4 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, refer to the *IGLOO2 FPGA Evaluation Kit Board* or *SmartFusion2 Security Evaluation Kit Board*.
4. Launch Libero SoC v11.7.
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, refer to the "Project Files" section on page 7.
7. Based on the location where the project files are extracted, update the paths in the eNVM data clients as follows:
   a. On the **Design Flow** window, double-click **Update eNVM Memory Content** as shown in Figure 4.

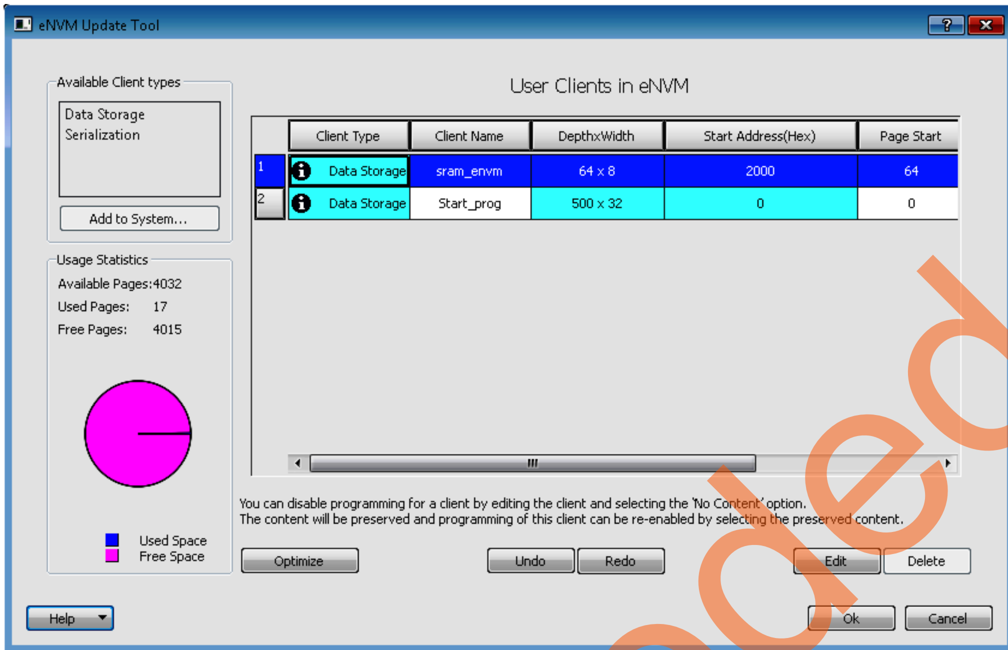
Figure 4 • **Update eNVM Memory Content in Design Flow Window**



The **eNVM Update Tool** window is displayed, as shown in Figure 5.

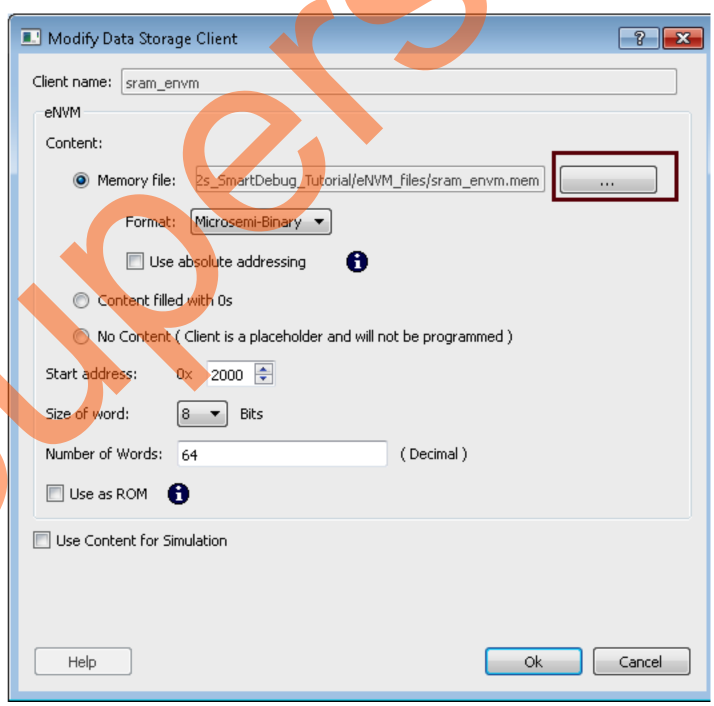
b. Double-click the **sram_envm** client.

Figure 5 • **eNVM Update Tool Window**



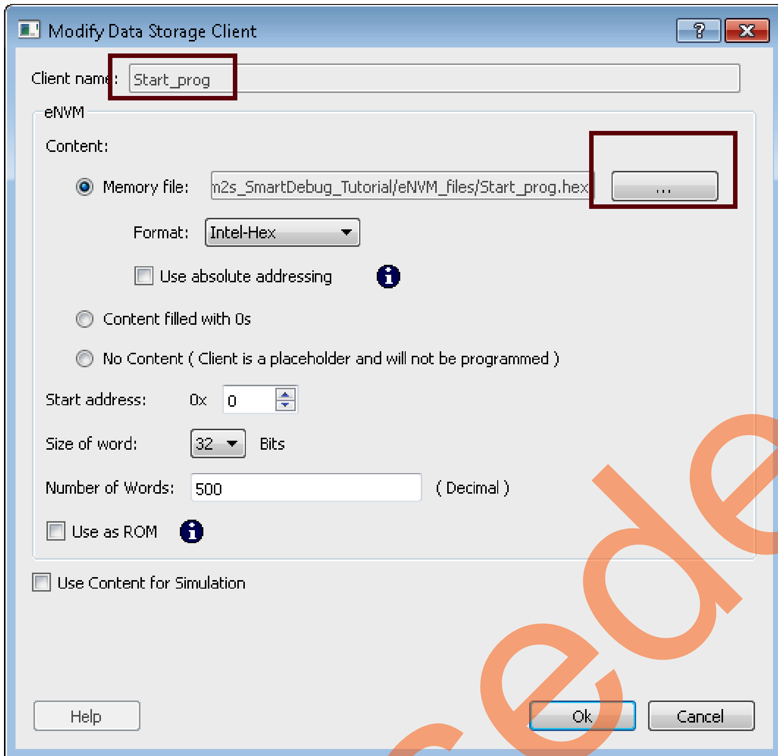
c. In the **Modify Data Storage Client** window, browse to the location of the `sram_envm.mem` file located in the eNVM_files folder included in the project as shown in Figure 6.

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



d. If the SmartFusion2 Security Evaluation Kit is used, double-click the **Start_prog** client (see Figure 5). 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 Figure 7.

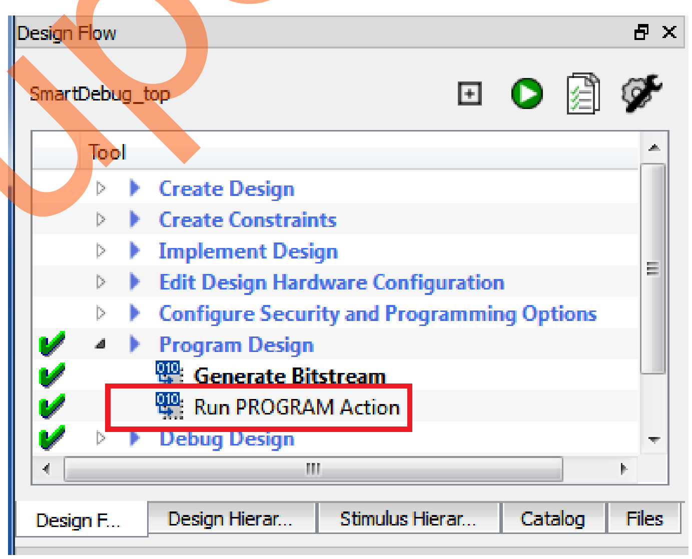
Figure 7 • Modify Data Storage Client Window - Specifying Start_prog.hex File



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 that 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, refer to *http://soc.microsemi.com/kb/article.aspx?id=SL5636*

8. In the **Design Flow** window, select **Run PROGRAM Action,** as shown in Figure 8. This programs the design into the device.

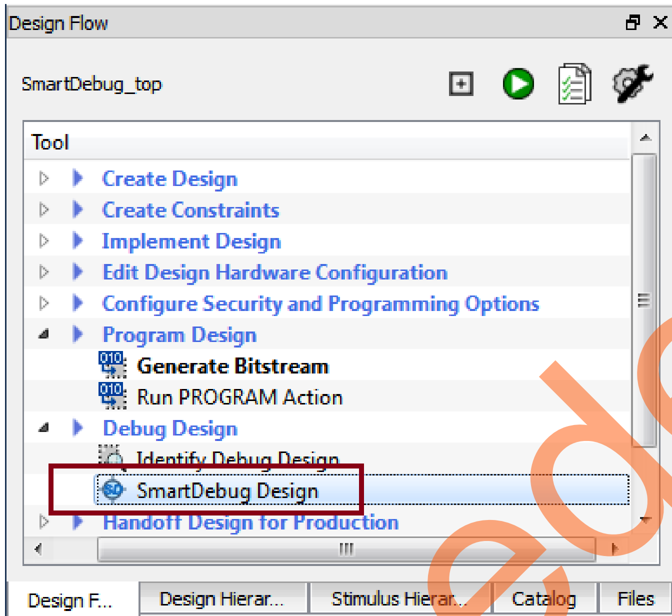
Figure 8 • Programming the Device



1.5 Launching SmartDebug from Libero

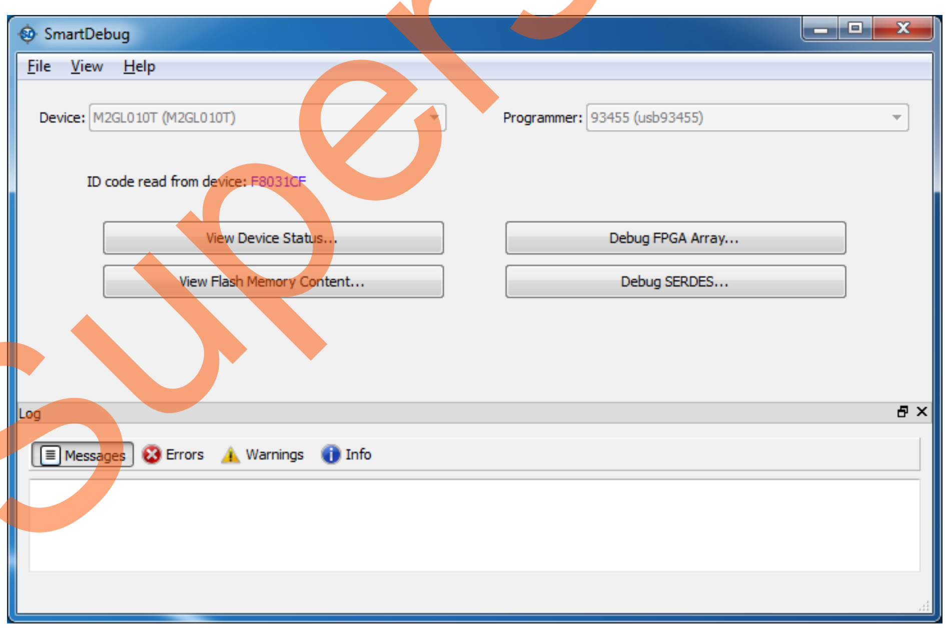
On the **Design Flow** window, double-click **SmartDebug Design**, as shown in Figure 9.

Figure 9 • **Launching SmartDebug Design Tools**



The **SmartDebug** window is displayed, as shown in Figure 10.

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). PROBE_A and PROBE_B are two dedicated pins on the device. You can connect an oscilloscope to the probe pins and monitor the signals status.

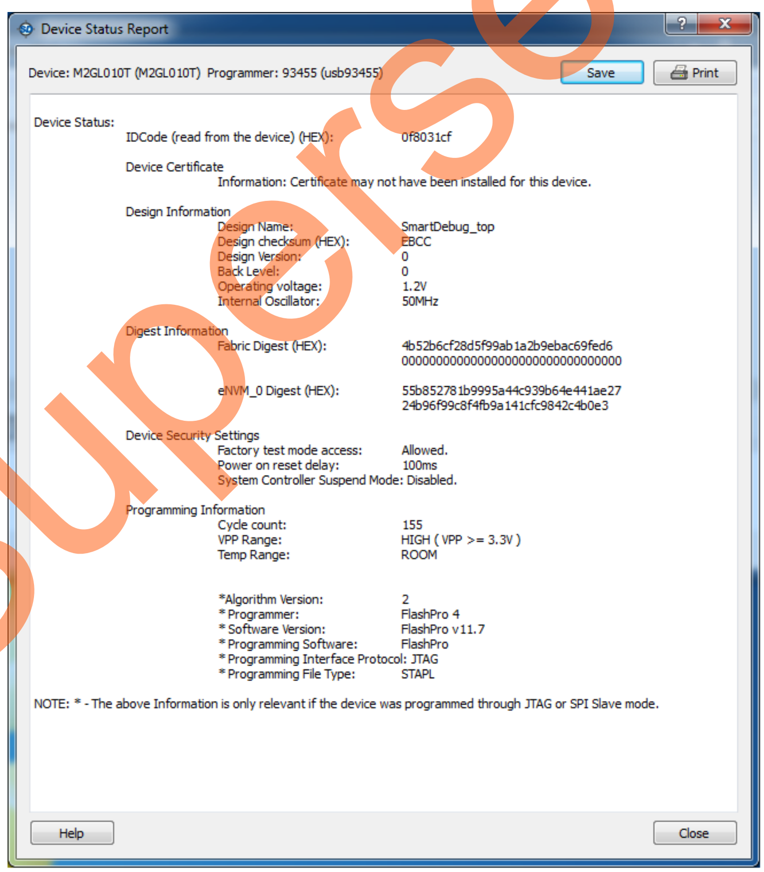
- **Active Probes**: Active probes allow dynamic asynchronous read and write to a flip-flop or probe point. This enables you to quickly observe the output of the logic internally, or to quickly experiment on how the logic is affected by writing to a probe point.
- **SRAM and eNVM Debug Capabilities**: 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 makes 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.

1.6 Debugging the Design

1.6.1 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. Figure 11 shows a sample of the device status information.

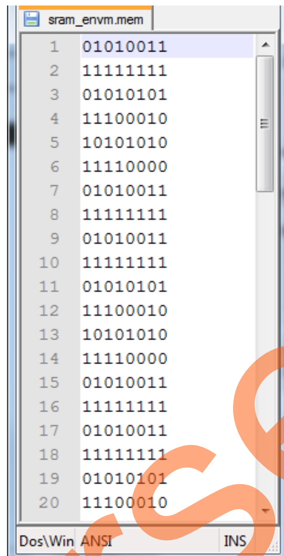
Figure 11 • **Device Status Report Sample**



1.6.2 View Flash Memory (eNVM) Content

The View Flash Memory Content can be accessed from the **SmartDebug** window, as shown in Figure 10. This option provides the capabilities to retrieve the eNVM content from the device using the Memories pages 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 for demonstration purposes. Figure 12 shows an excerpt of the data storage client content that was defined in the eNVM.

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

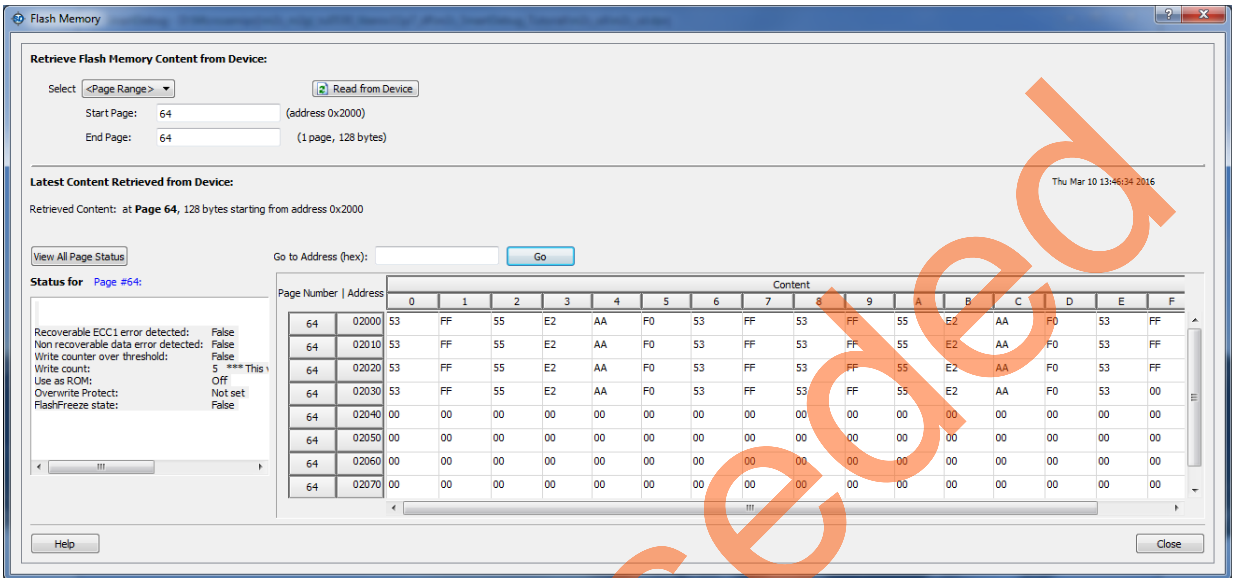


The content of eNVM is retrieved from the device, displayed, and is equivalent to what is shown in Figure 12.

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). The **Flash Memory** window is displayed, as shown in Figure 13.
2. Enter the **Start Page** and **End Page** as **64. Page 64 is used here for demonstration purpose.**
3. Click **Read from Device.** The content related to page 64 is displayed.

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



eNVM UI shows the status of the page selected eNVM page. 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, eNVM UI shows the status of each page that you select to view in the eNVM debug page.

1.6.3 Debug FPGA Array

The 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** UI includes both a **Hierarchical** and **Netlist** view to easily 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.

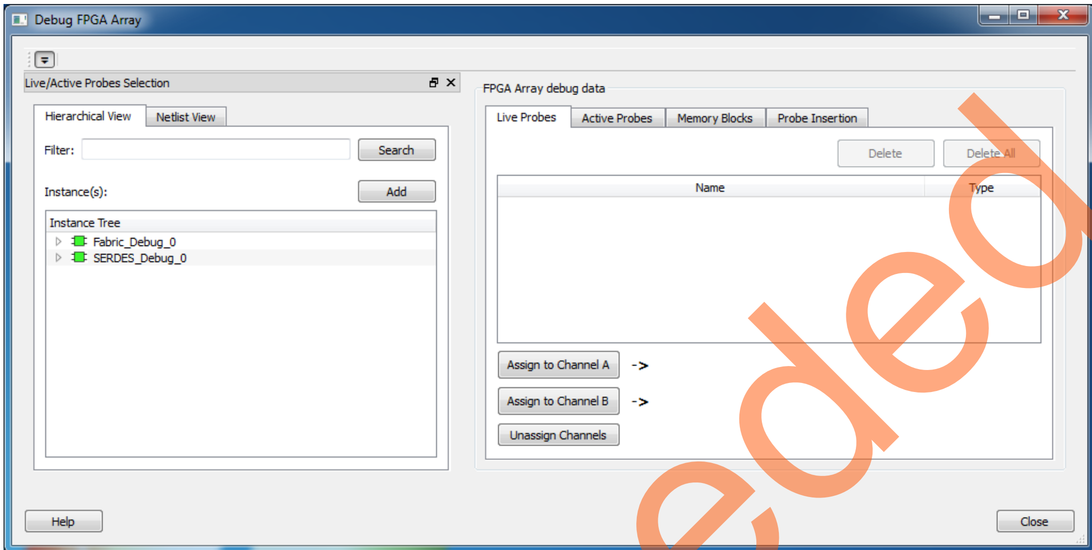
You can expand the hierarchy tree to see the lower level logic. Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree. The probe points are added by selecting any instance or the leaf level instance in the Hierarchical View. Adding an 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.

The Debug FPGA Array can be accessed from the SmartDebug window, as shown in Figure 10. On the **SmartDebug** window, click **Debug FPGA Array** to display the **Debug FPGA Array** window, as shown in Figure 14. The Debug FPGA Array window has a left and right pane. 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.

Figure 14 • **Debug FPGA Array Window**



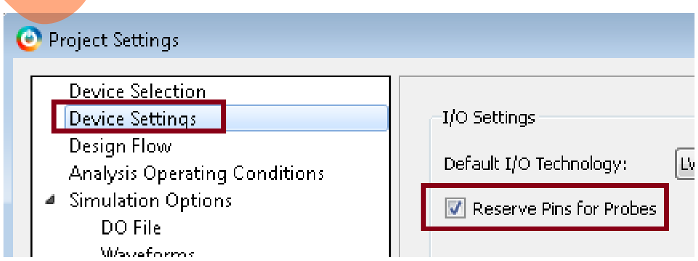
The following steps describe how to use the Live Probes, Active Probes, and the Memory Block debugging features:

1.6.3.1 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 Figure 15.

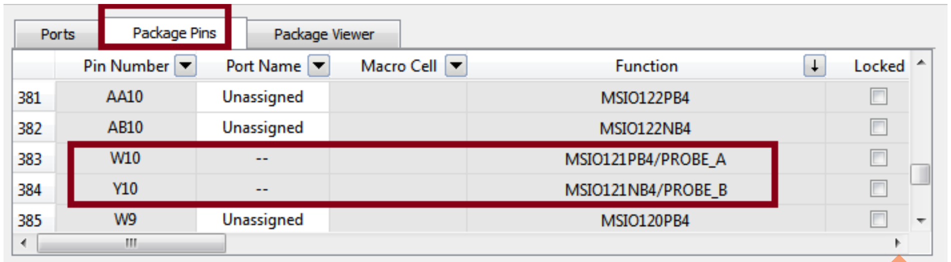
Figure 15 • **Reserving Probe Pin for Probes**



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 Figure 16.

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 Figure 16.

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



	Pin Number	Port Name	Macro Cell	Function	Locked
381	AA10	Unassigned		MSIO122PB4	
382	AB10	Unassigned		MSIO122NB4	
383	W10	--		MSIO121PB4/PROBE_A	
384	Y10	--		MSIO121NB4/PROBE_B	
385	W9	Unassigned		MSIO120PB4	

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.

Figure 17 shows an example of setting two probe points: coutA[23]:Q and coutA[24]:Q to be probed on ChannelA and ChannelB 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 ChannelA (PROBE_A) or ChannelB (PROBE_B) as follows:

1. Select the **point** to be probed, as shown in Figure 17.
2. From the **Netlist View,** select the Net to be probed**,** and click Add to that Net to the FPGA Array debug data.
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

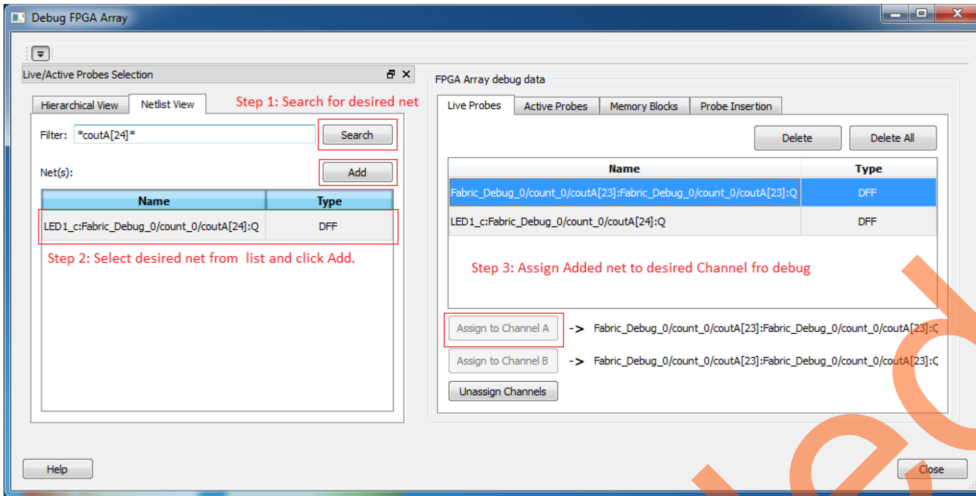
PROBE_B:

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

After setting the channels, SmartDebug configures the ChannelA and ChannelB I/Os to monitor the desired probe points. On the SmartFusion and IGLOO2 Evaluation Kit Rev D boards, the 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 to also discontinue the live probe function during debug.

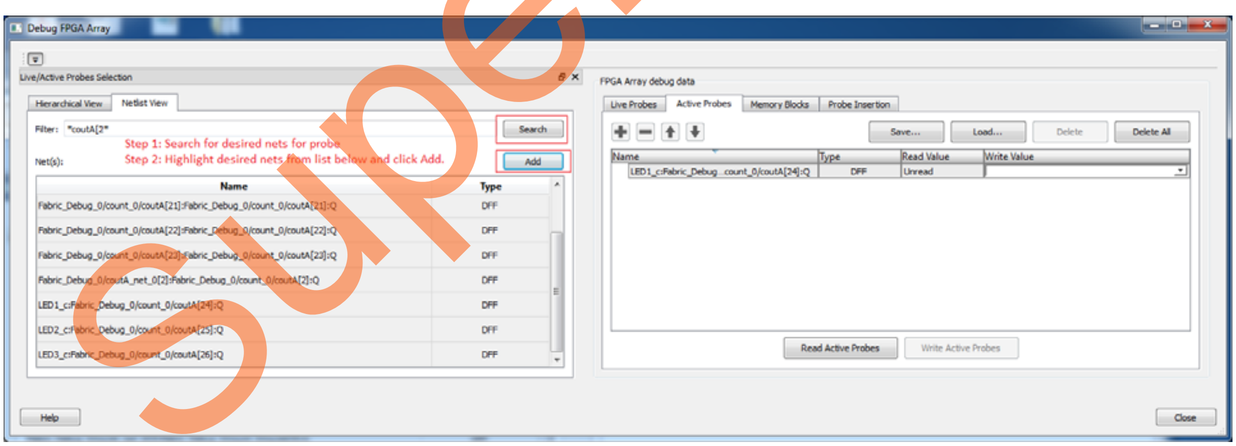
1.6.3.2 Active Probes

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.

1.6.3.2.1 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 Figure 18.

Figure 18 • 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.

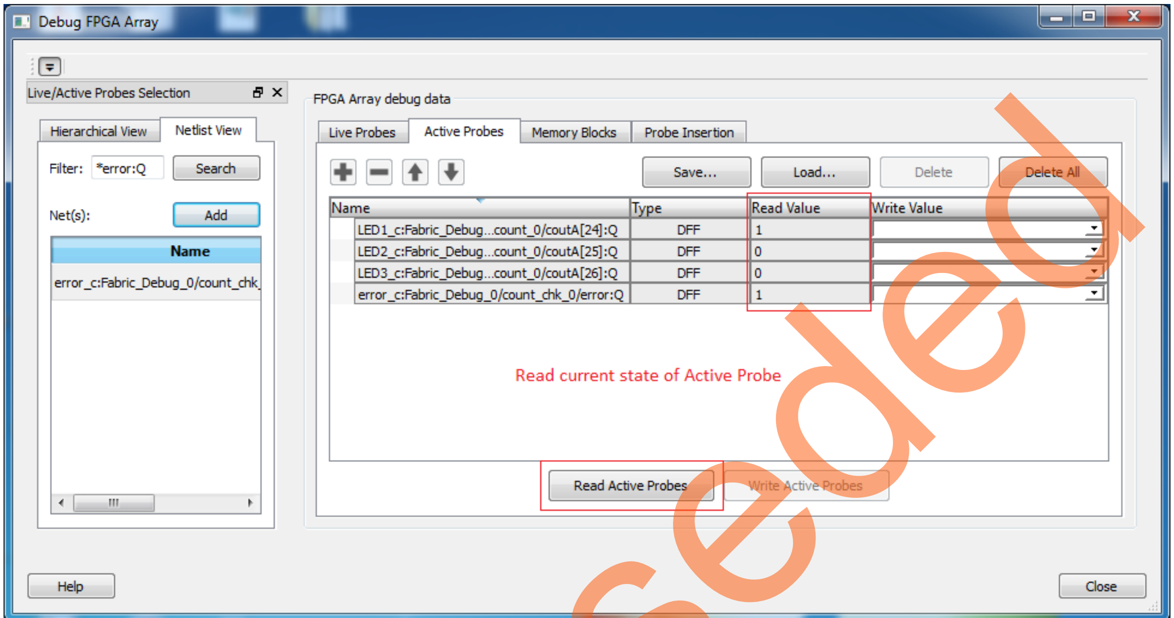
To find these points in the list of available probe points, use the Filter control, as shown in Figure 19.

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 Figure 19.

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 19 • **Selecting Desired Points to Read and Reading the Values**

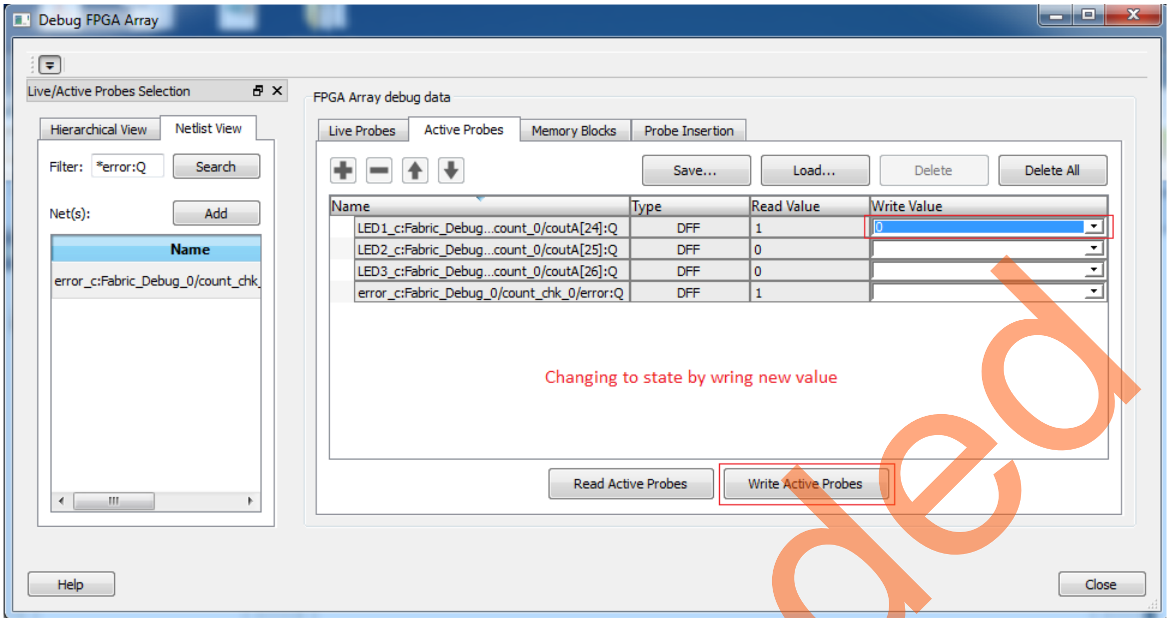


Note: The “+” and “-” icons expand or collapse the bus or group present in the Active Probes UI. In Figure 19, there is no bus or group, as a result, these buttons are not enabled.

1.6.3.3 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. Figure 20 shows the results of a first write of the design.

Figure 20 • **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, and to save them to a file. Use the Load button to load a saved active probe setup. This allows the user 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 Figure 20.

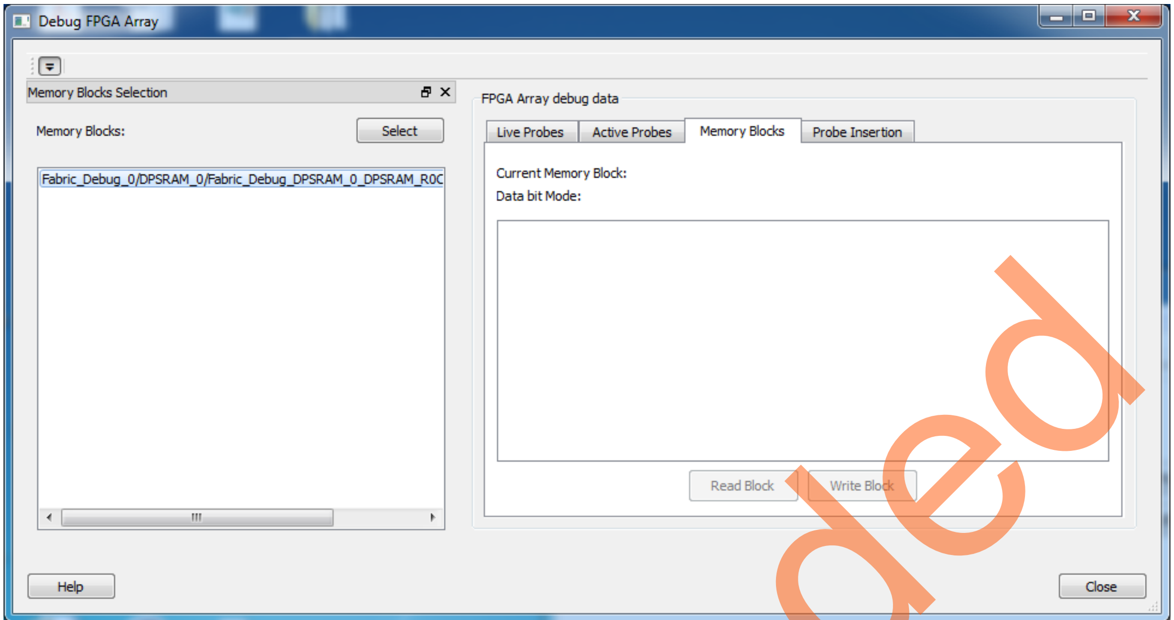
Also, the error signal must be **High (LED H5 is OFF),** indicating that there are no errors in the counting pattern.

The 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.

1.6.3.4 Fabric SRAM Memory Debug

To view the content of the Large SRAM in this design, click the **Memory Blocks** tab, as shown in Figure 21.

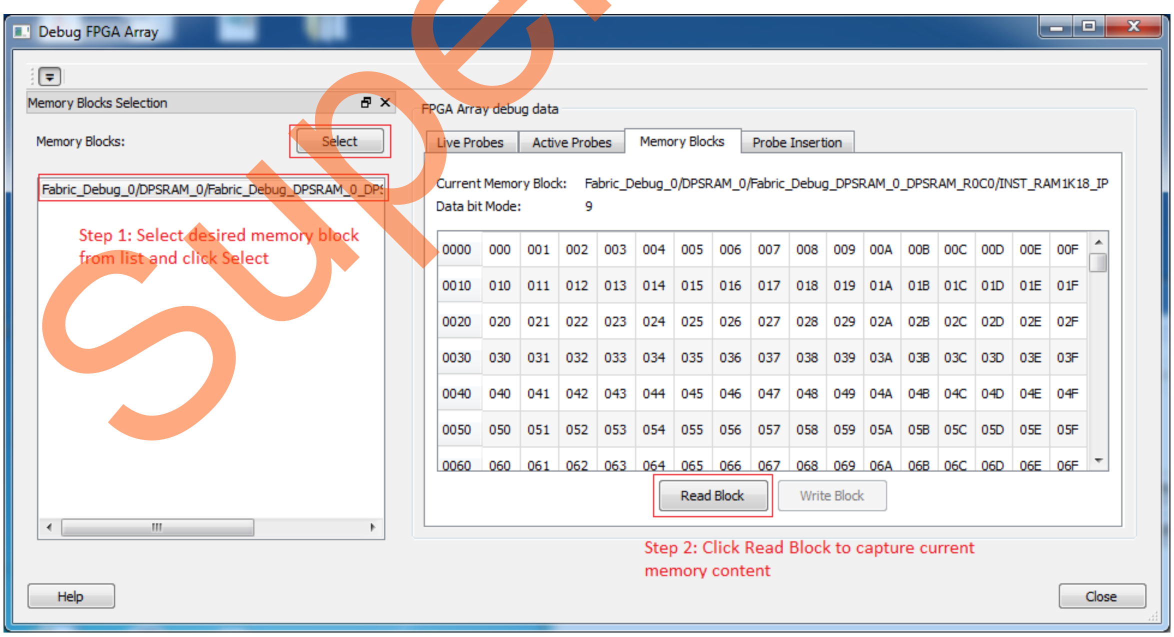
Figure 21 • **Memory Blocks Tab**



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 this left side panel window. Highlight 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 on **Read Block**.

The contents of the DPSRAM_0 is displayed, as shown in Figure 22. See the counting pattern that is loaded into the RAM.

Figure 22 • **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.

1.6.4 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.

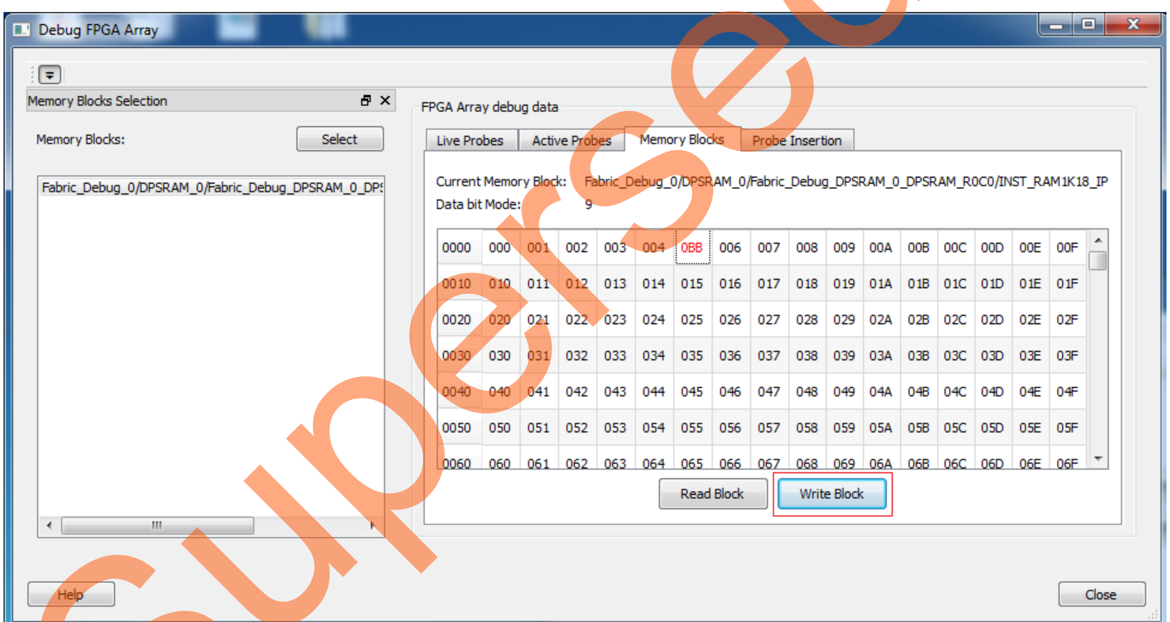
The following steps describe how to modify the RAM content and force an error:

1. Read the memory content, as shown in Figure 22.
2. Select an entry and double-click. Each entry is 9-bits wide.
3. Modify the value from the current value to break the count pattern. An example is shown in Figure 23.

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 Switch2 (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 23 • **Modifying DPSRAM Contents**



1.6.5 Probe Insertion

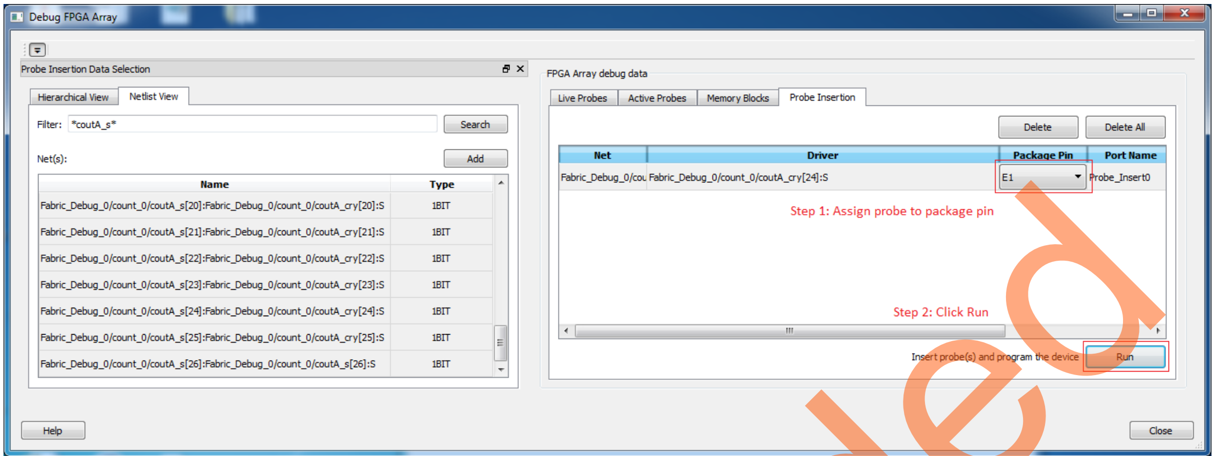
Probe insertion is a post-layout process that enables you to insert probes into the design and bring signals out to the FPGA package pins to evaluate and debug the design. Probe insertion enables you to select internal nets anywhere in the design, connect those nets to unused pins, and then run layout incrementally to manage the physical connection to the pin. Nets are selected and assigned using the SmartDebug Probe Insertion feature. For more information, refer to the *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 Figure 24.
2. In the **Filter** field, enter the ***couta*** signal, as shown in Figure 24, 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 Figure 24.

Figure 24 • **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 stand-alone SmartDebug.

1.6.6 SERDES Debug

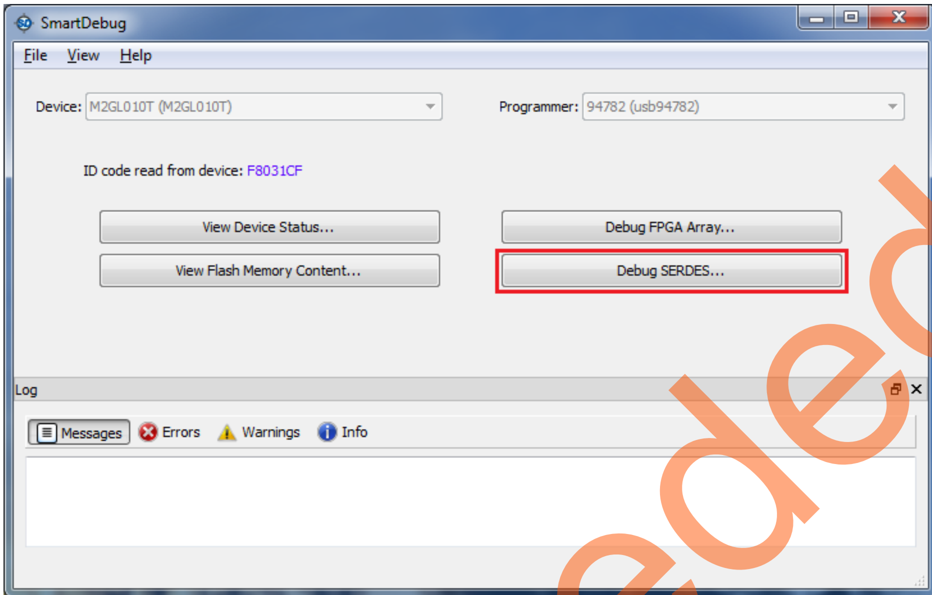
This SmartDebug SERDES tutorial helps the FPGA and board designers to perform SERDES real-time signal integrity testing and tuning in a system that:

- 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 Figure 25.

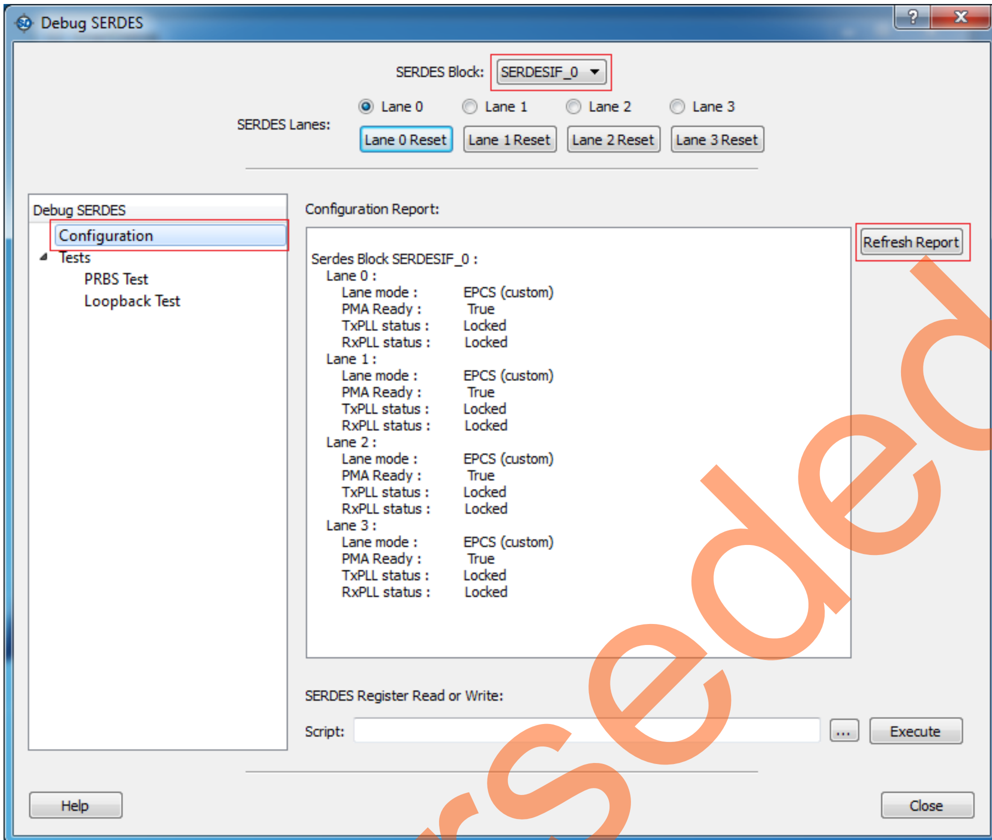
Figure 25 • **Debug SERDES Operation Selection**



The Configuration tab is displayed as shown in Figure 26. 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.

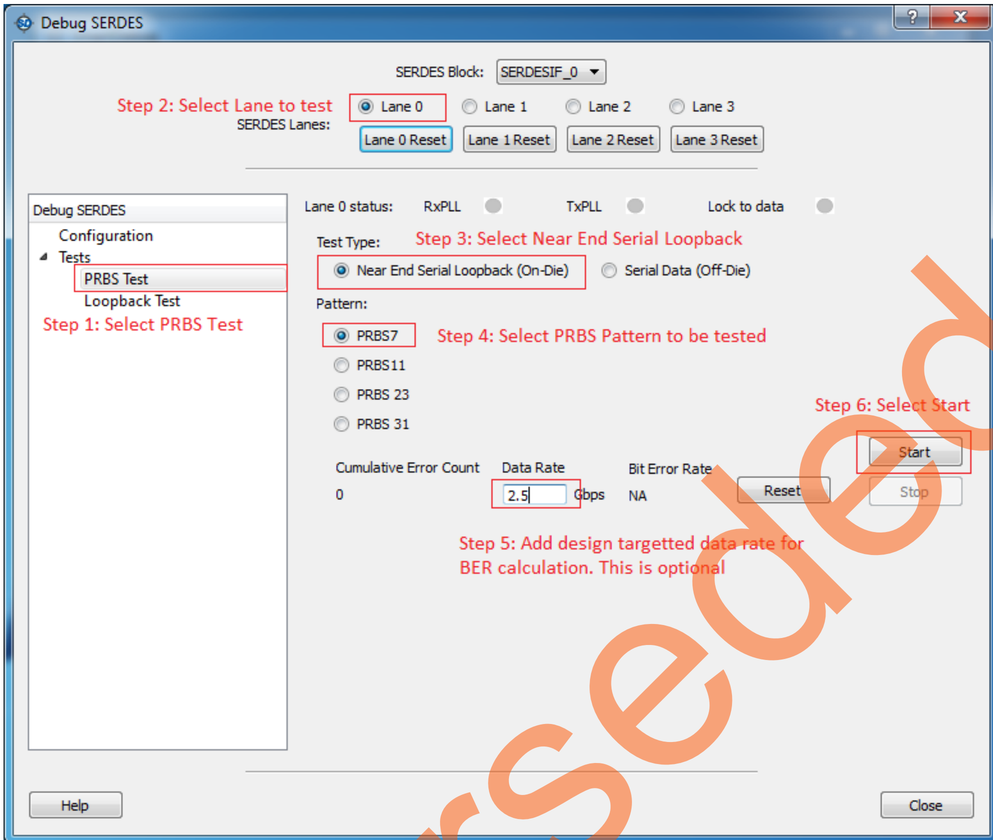
2. Click **Refresh Report,** to update the data.

Figure 26 • **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 Figure 29.
This test generates and checks PRBS7 data without going off-chip. The green LEDs indicate the lock status of TxPLL and RxCDR for the selected lane.

Figure 27 • **SERDES Test Tab**



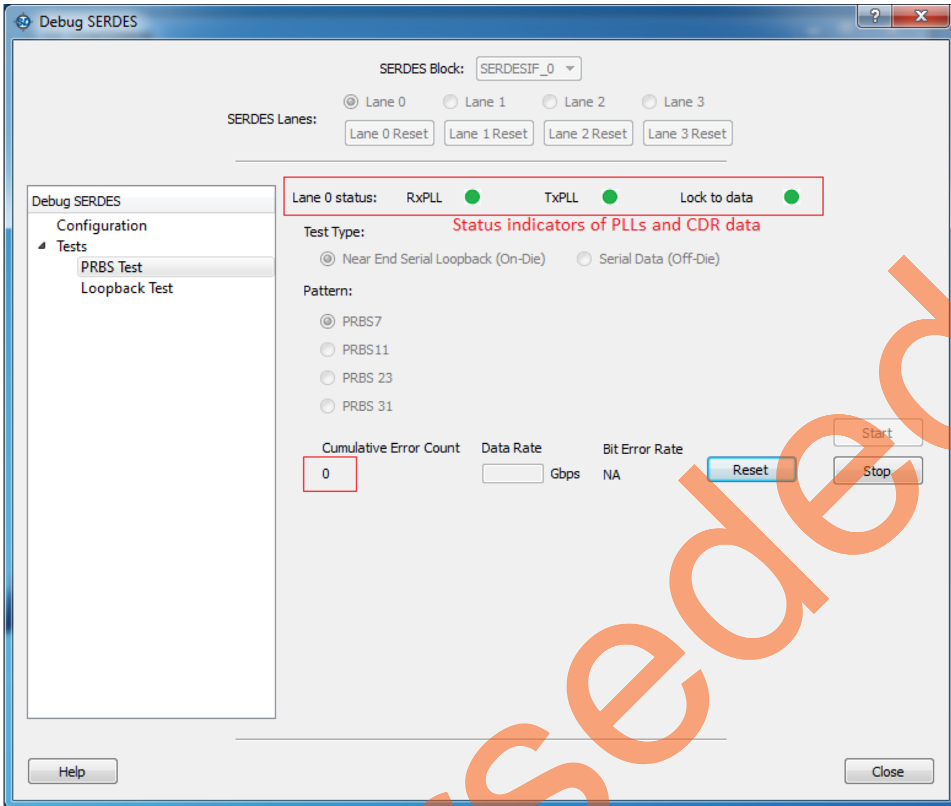
In this example setup, the data stream is expected to have zero errors as the datapath does not go off-chip while using the Near-end Serial Loopback as shown in Figure 27.

The Data Rate value is entered optionally 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.

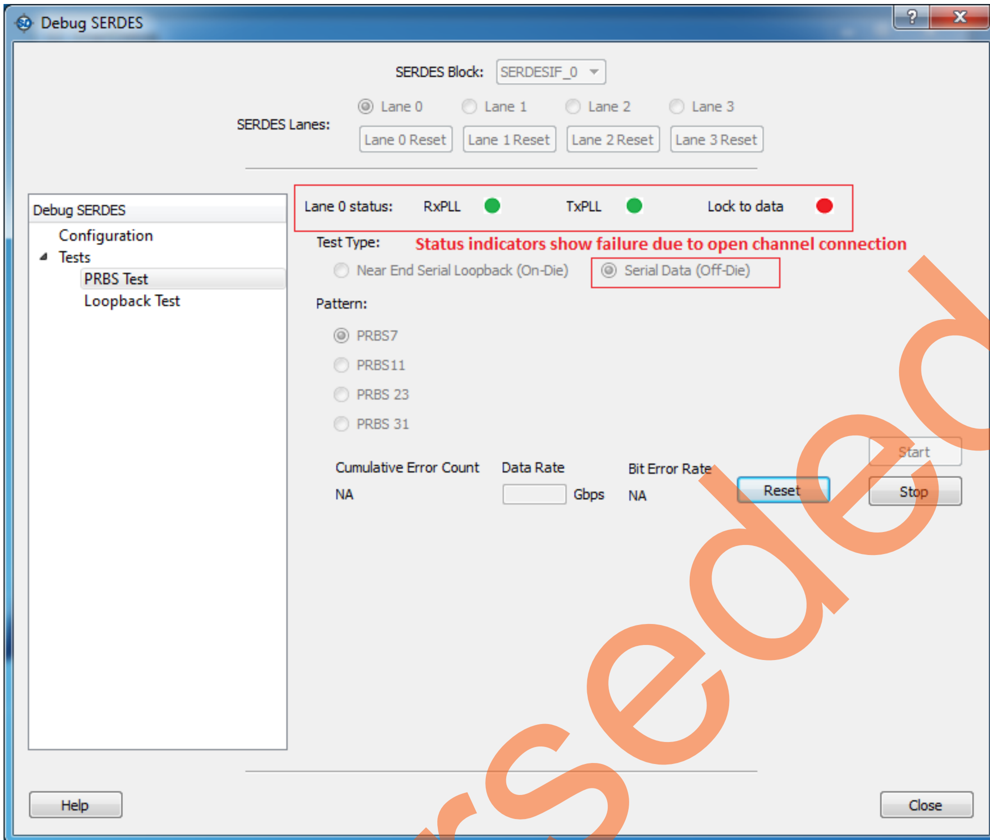
Figure 28 • **SERDES Link Status**



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

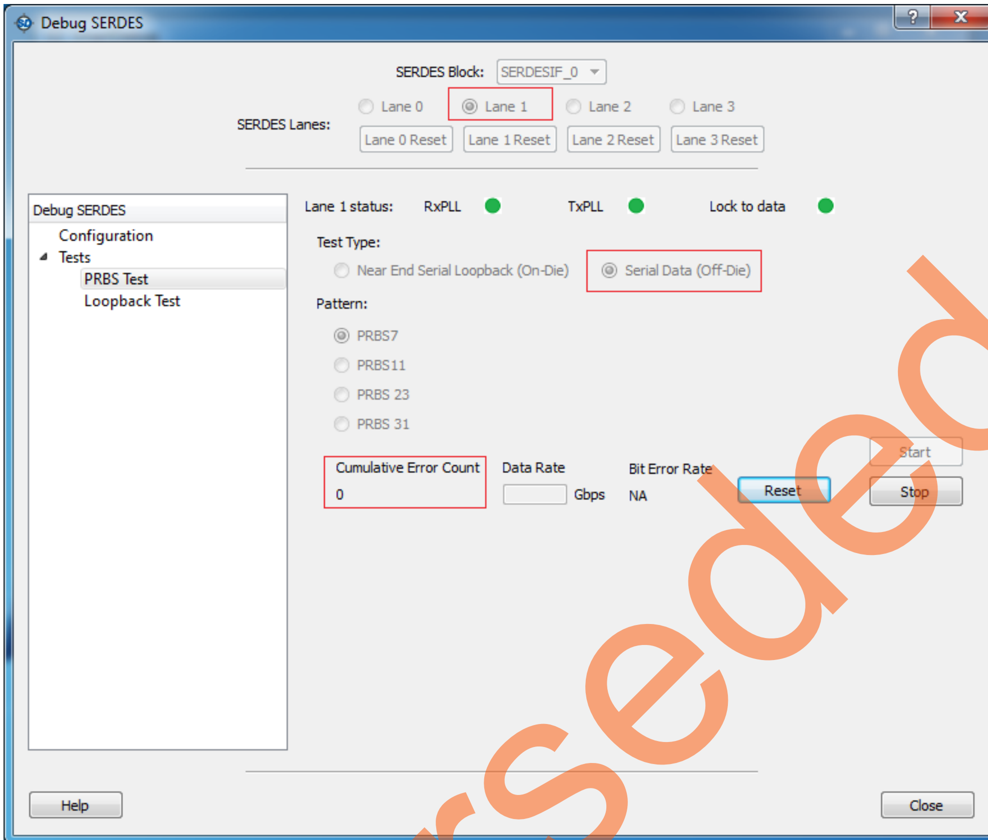
4. Click **Stop** and select **Serial Data (Off-Die)**. Click **Start** and observe that the **Lock to data** status indicator is red. This indicates that the data is no longer looping between Tx and Rx and Lane 0 is not looped together on the PCB.

Figure 29 • Sending Serial Tx Data Off-Die



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 Figure 32.
   This test generates and checks PRBS7 data going off-chip and folded back on the PCB to the receiver.

Figure 30 • **Lane 1 Transmitting Data Through On-Board Loopback**



6. The Tx and Rx channels of Lane 2 can be interconnected in a loopback configuration using coaxial cables. In this example, as shown in Figure 33 and Figure 34, after connecting a pair of high-quality 50 Ω SMA cables to the SMA connections on the Evaluation Kit board, SERDES debug can be used to send data off-board and check for errors. Select **Lane 2**, **Serial Data (Off-Die)**, **PRBS7**, and click **Start**.

Figure 31 • **External Cable Loopback**

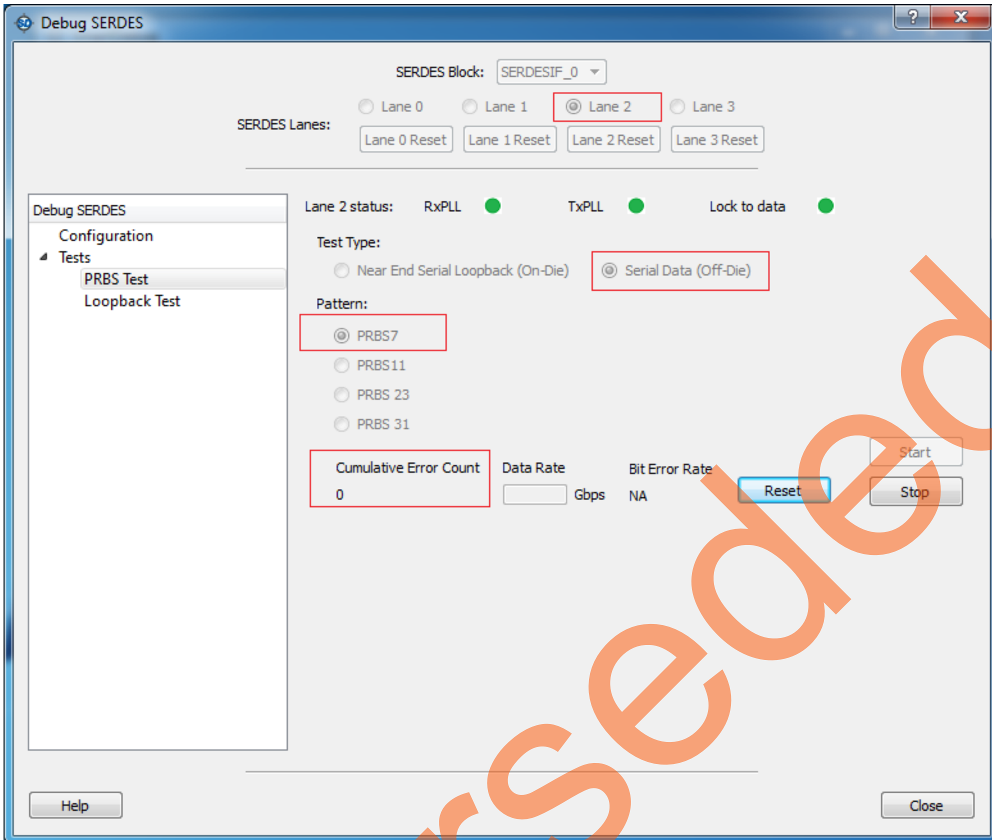
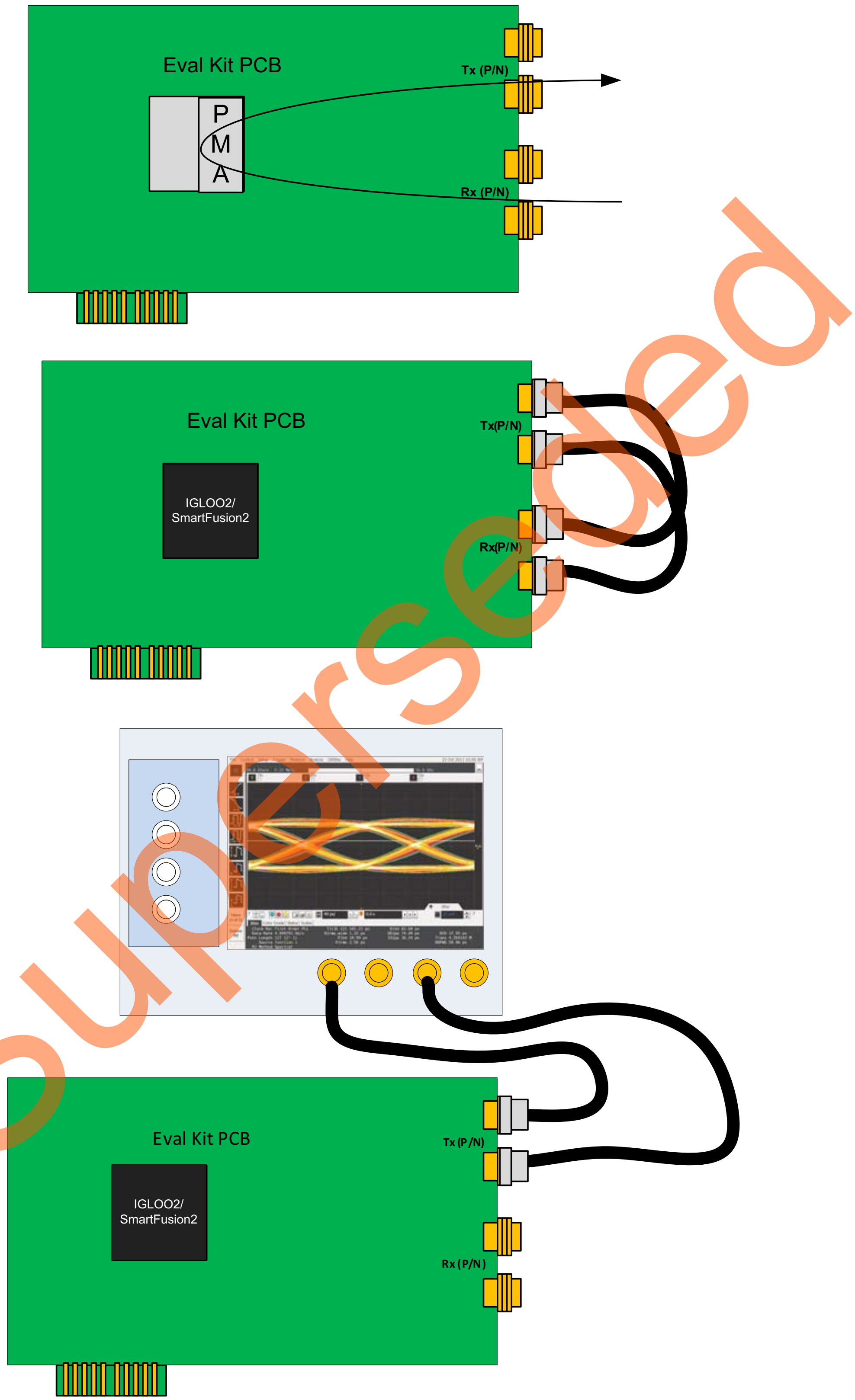


Figure 32 • **Evaluation Kit Board with External Coax Loopback Setup**



7. 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. In the example shown in Figure 35, when the Lane 2 test is started without any means to connect the Tx and Rx together, **Lock to data** status goes red 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 33 • **Lane 2 Transmitting Data Off-Board**

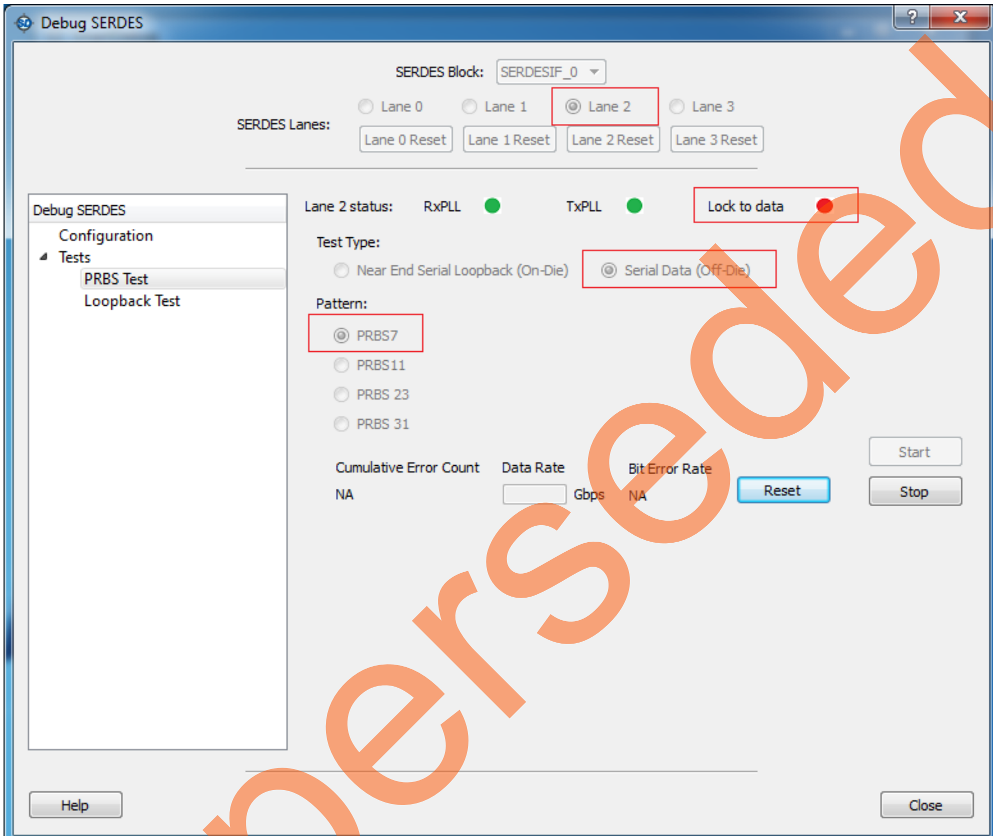
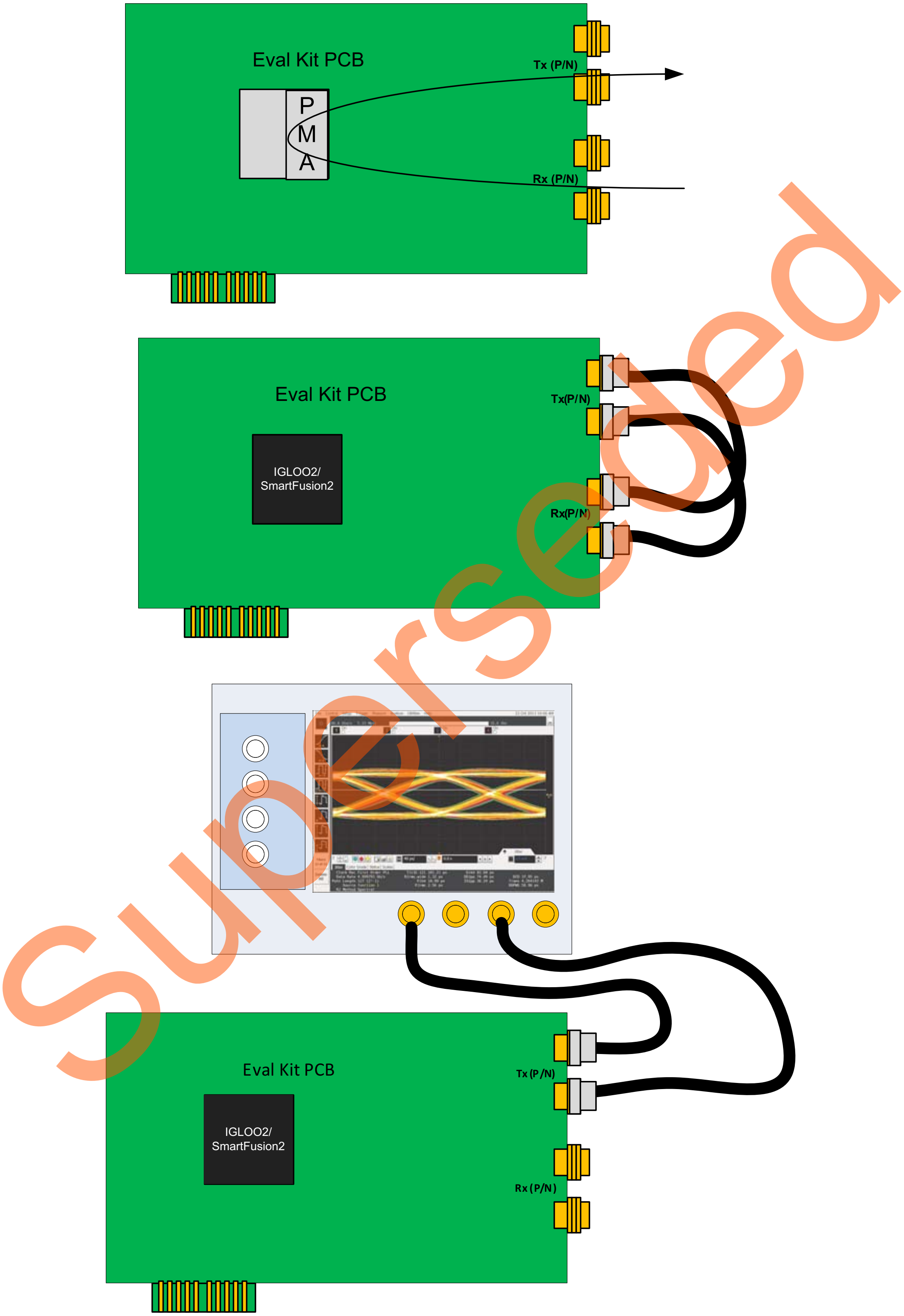


Figure 34 • Connecting Lane 2 to the Test Equipment



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

Bit error rate (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.

1.6.7 Far-End Loop Back 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).

In the example shown in Figure 37 and Figure 38, 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.

This is performed by selecting SERDES Lane 2, PCS Far End PMS Rx to Tx Loopback Test Type, and clicking Start.

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 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 35 • **PCS Far-End Rx to Tx Loopback**

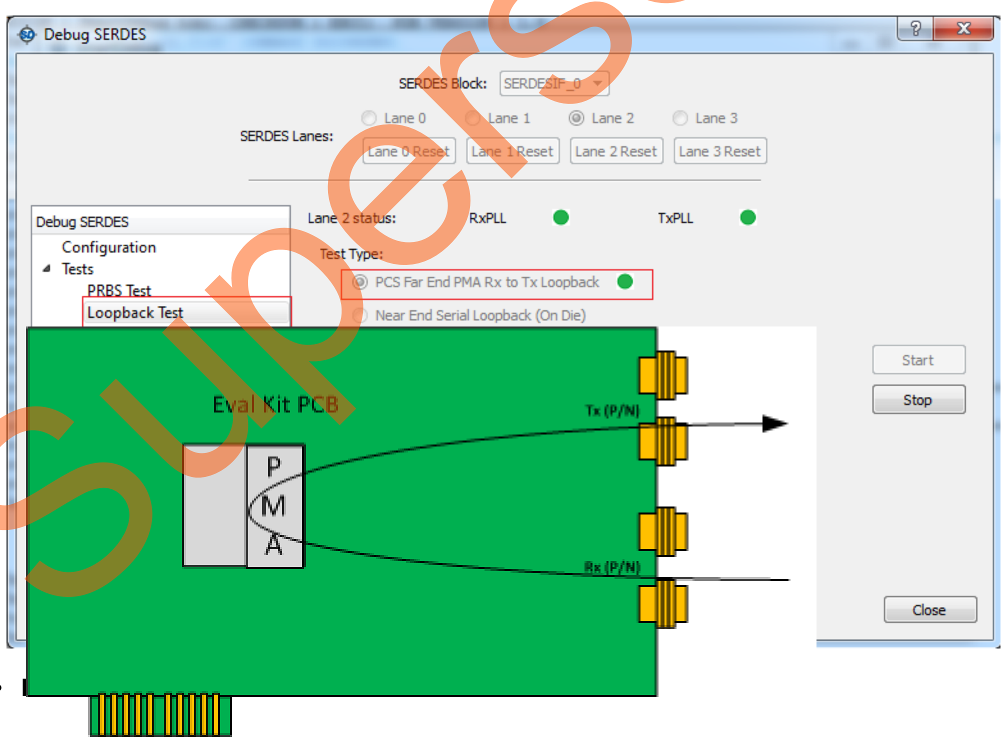


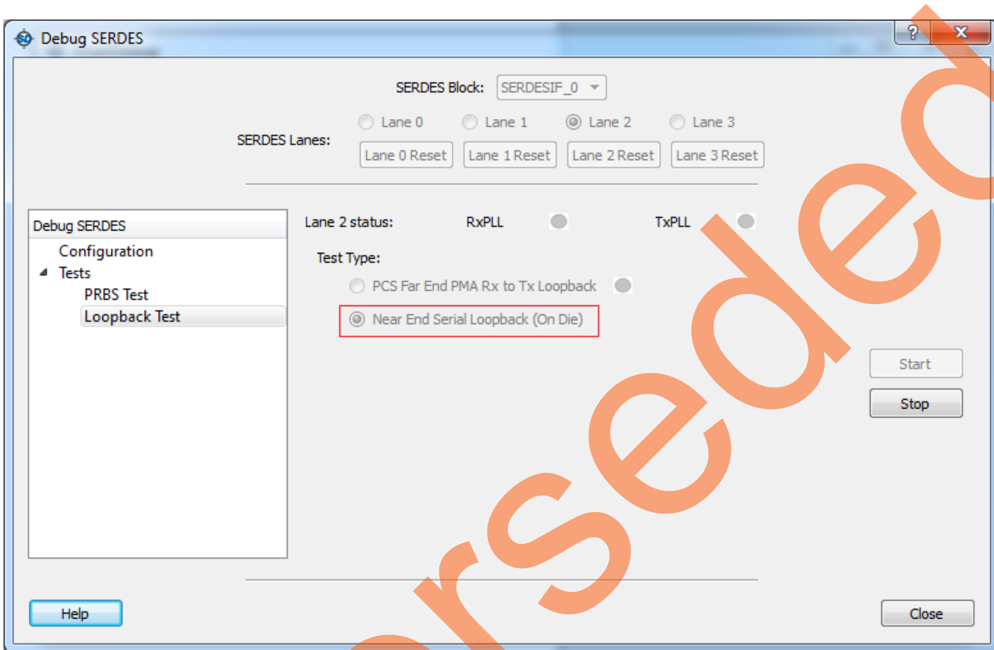
Figure 36 •

1.6.8 Near-End Serial Loopback

The Near-end Serial Loopback Test feature, as shown in Figure 37, 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 37 • **Loopback Test Feature**



1.6.9 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 8 hex character (with optional 0x prefix) example:
0x4002200C

RegisterValue is 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, refer to the *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
```

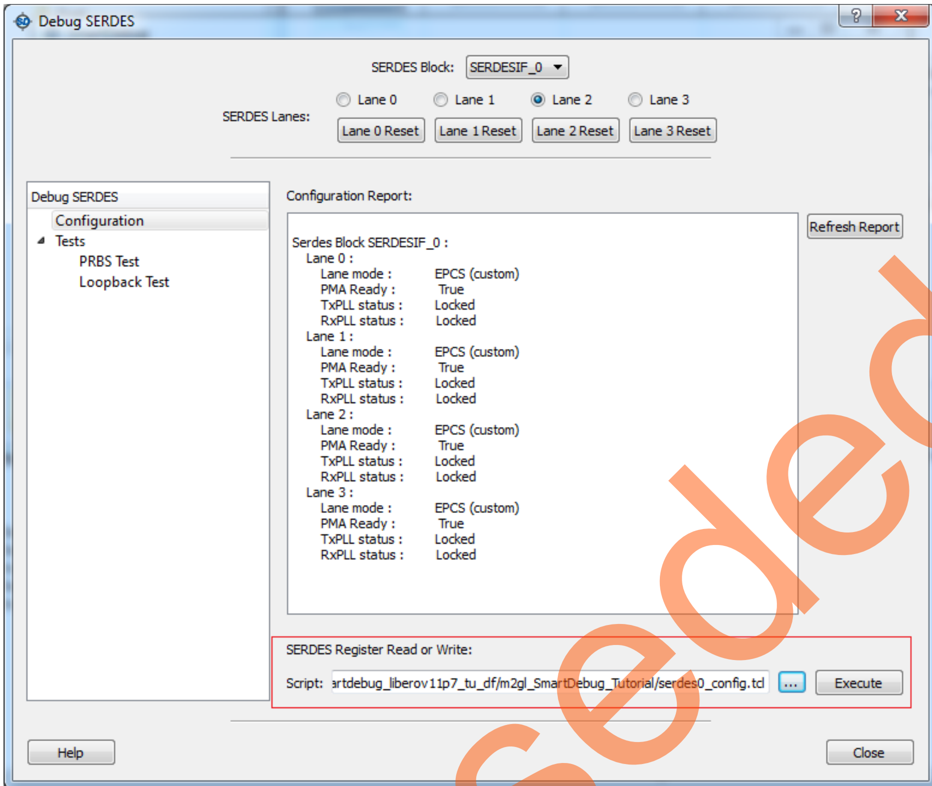
The value of the CONFIG_PHY_MODE_1 register must be known in the example shown above. This register contains the value of the CONFIG_REG_LANE_SEL, which defines the lanes accessed in the design. In this example, reading the CONFIG_PHY_MODE_1 register and passing its value and the associated offset targets 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.

Tcl commands and syntax are found in the SmartFusion2 and IGLOO2 FPGAs Tcl for SoC – Tcl Documentation.

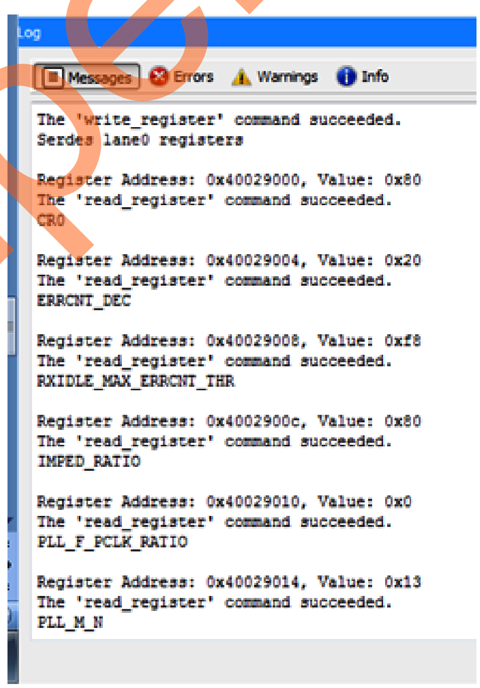
From the **Debug SERDES Configuration** tab, an executable Tcl script can be imported. Browse to the Tcl script file and click **Execute**, as shown in Figure 39. The script contains commands to write or read registers using a flattened top for most address mapping.

Figure 38 • **Tcl Script Execution User Interface**



After execution of the Tcl SERDES access, the log of the access is displayed in the SmartDebug Console Log pane, as shown in Figure 40.

Figure 39 • **SERDES Access Log**



Refer to "Appendix" on page 45 for more Tcl examples.

1.6.10 Executing SERDES Debug from SmartDebug Tcl

1.6.10.1 PRBS

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 (see the SmartDebug User Guide, for details).

-**start:** To start PRBS test.

-**stop**: To stop PRBS test.

-**reset_counter**: To reset the PRBS error count value to 0.

-**read_counter**: To 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
```

1.6.10.2 Loopback

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: To start loopback test.

stop: To 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:

1. **plesio** (PCS Far End PMA Rx to Tx Loopback)
2. **parallel**
3. **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
```

1.7 Stand-Alone SmartDebug

SmartDebug is offered as a stand-alone utility. This allows SmartDebug to be used on other host computer systems without the full installation of the Libero SoC Software. The stand-alone SmartDebug flow requires the user to create standalone SmartDebug project.

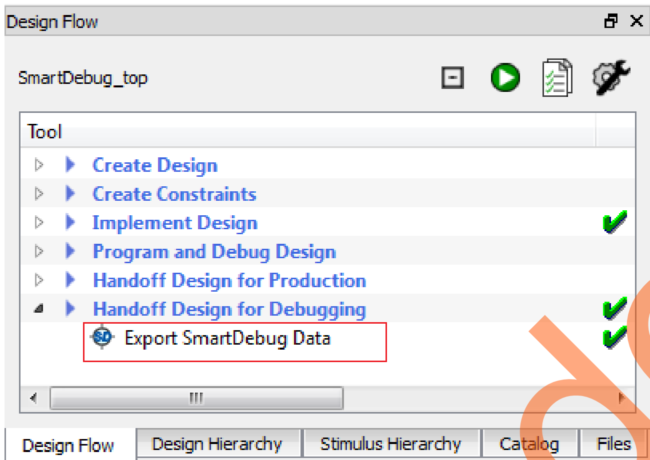
There are two ways to create a stand-alone SmartDebug project:

- Import DDC files from Libero
- Construct 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 prior to creating the DDC file.
2. Double-click **Export SmartDebug Data** to generate the DDC file from the Libero SoC design flow.

Figure 40 • **Export SmartDebug Data**



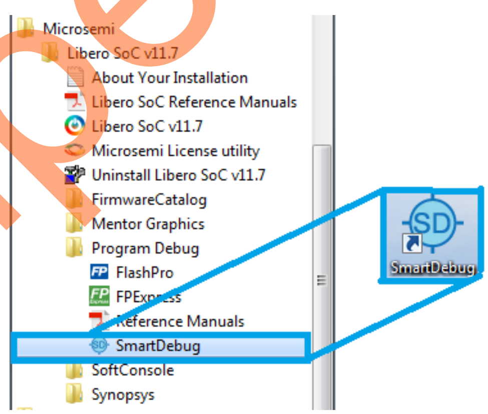
The exported DDC file is located at: *<project path>\designer\Project\export\Project_top.ddc*

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

The following procedure describes how to create a SmartDebug project:

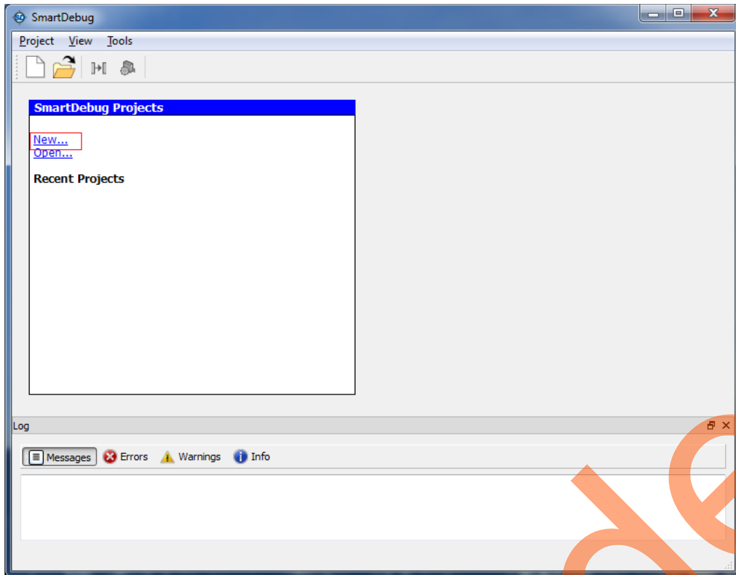
1. Connect a FlashPro programmer to a valid hardware device.
2. Start SmartDebug stand-alone utility. The Standalone SmartDebug is found in the Program Debug program group of the Microsemi Libero SoC v11.7 program group, as shown in Figure 41. You can also find it as an icon on your desktop.

Figure 41 • **Starting Standalone SmartDebug**



3. Click **New**.

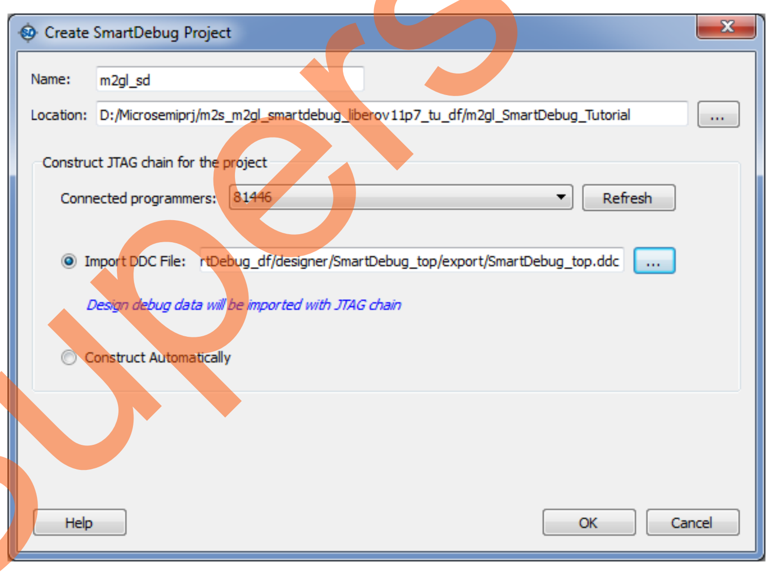
Figure 42 • **New SmartDebug Project**



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

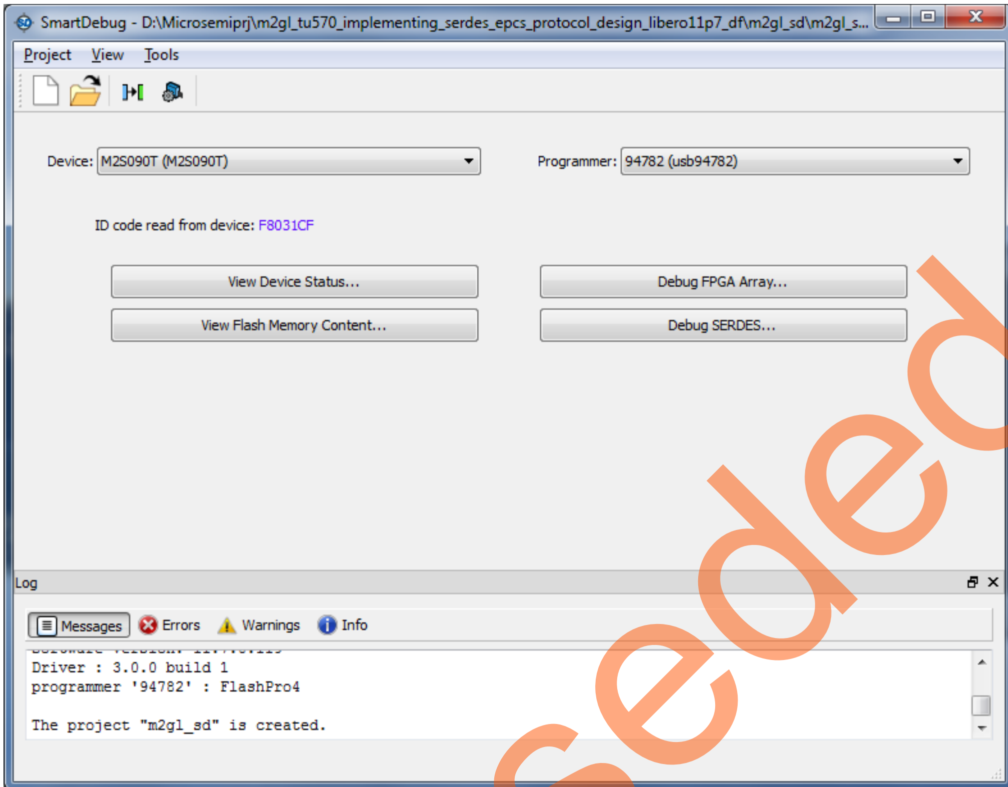
4. Enter the name of the SmartDebug project.
5. Browse to the desired directory location and select the DDC file related to the Libero SoC project.
6. Click **OK**.

Figure 43 • **Create SmartDebug Project**



The SmartDebug project is created. At this point SmartDebug works exactly the same as when it is launched from Libero. See the prior sections for operation. The Standalone SmartDebug UI is shown in Figure 44.

Figure 44 • **Standalone SmartDebug UI**



1.8 Conclusion

This tutorial demonstrated the capabilities of SmartDebug. SmartDebug provides the capabilities to observe and analyze many embedded device features. Live Probe gives real-time access to device test points and internal logic states can be easily 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. Adjustments and tuning the PMA analog settings for optimal link performance is easily accomplished to match the design to the system. Using the SmartDebug utility with the Evaluation Kit board provides designers a good understanding of its features and capabilities.

2 Appendix

2.1 Tcl Script Examples

2.1.1 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"
```

2.1.2 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"
```

2.1.3 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 Revision History

The following table shows the important changes made in this document for each revision.

Date	Version	Page
Revision 8 (April 2016)	Updated the document for Libero v11.7 software release (SAR 75567).	NA
Revision 7 (October 2015)	Updated the document for Libero v11.6 software release (SAR 68374).	NA
Revision 6 (January 2015)	Updated the document for Libero v11.5 software release (SAR 62936).	NA
Revision 5 (October 2014)	Updated the document for SERDES core change (SAR 61612).	NA
Revision 4 (September 2014)	Updated the document for Libero v11.4 software release (SAR 59069).	NA
	Updated the document for M2S025 Evaluation Kit board details (SAR 59069).	NA
	Updated the document for M2GL010 Evaluation Kit board details (SAR 59069).	NA
Revision 3 (April 2014)	Added Note in "Specifying Live Probe Points in Libero" section(SAR 56593).	17
Revision 2 (March 2014)	Updated the software version from 11.2 SP1 to 11.3 (SAR 56012).	NA
	Updated design files using the latest 11.3 SERDES core (SAR 56012).	NA
Revision 1 (January 2014)	Initial release.	NA

4 Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

4.1 Customer Service

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

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

4.2 Customer Technical Support Center

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

4.3 Technical Support

For Microsemi SoC Products Support, visit
http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

4.4 Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/products/fpga-soc/fpga-and-soc.

4.5 Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

4.5.1 Email

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

The technical support email address is soc_tech@microsemi.com.

4.5.2 My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

4.5.3 Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

4.6 ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace 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; 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, Calif, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

50200530-8/4.16