SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools

Libero SoC v11.4SP1 Tutorial

Microsemi

Table of Contents

SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools

Introduction

Design debug is a critical phase of the FPGA design flow. Microsemi®'s multiple design debug tools and features compliment design simulations by allowing verification and troubleshooting at the hardware level. Having successfully passed functional and post-layout simulations, Microsemi's design debug tools can help provide the designer with a pre-system level implementation early warning of other design issues. Microsemi's design debug tools can provide the peace of mind, that intended design goals and functionality are maintained by performing various analysis in the actual programmed FPGA. Microsemi design debug focuses the designer on analysis of the key elements of a flash design, such as the embedded non-volatile memory (eNVM) data, SRAM data, and probes capabilities. Microsemi SmartFusion®2 and IGLOO®2 field programmable gate array (FPGA) devices have built-in probe points that greatly enhance the ability to debug logic elements within the device. The enhanced debug features implemented into the SmartFusion2 and IGLOO2 devices give access to any logic element and enable designers to check the state of inputs and outputs in real-time, without any re-layout of the design through Live Probe and Active Probe features:

- 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 then be sent to two dedicated pins (PROBE_A and PROBE_B), then to an oscilloscope, or even redirected back to the FPGA fabric to drive a software logic analyzer.
- Active Probe allows dynamic asynchronous read and write to a flip-flop or probe point. This will enable a user to quickly observe the output of the logic internally, or to quickly experiment on how the logic will be affected by writing to a probe point.
- SmartDebug includes SERDES control and test capabilities that can also access SRAM and eNVM to assist with debugging high speed serial designs, with no extra steps. The SmartDebug JTAG interface extends access to configure, control, and observe SERDES operations and is accessible in every SERDES design. Users simply implement their design with the Libero® System Builder to incorporate the SERDESIF block enabling SERDES access from the SmartDebug toolset.

This quickly enables designers to explore configuration options without going through FPGA recompilation or making changes to the board. The SERDES Debug GUI 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.

Upon completing this tutorial you will be familiar with the following:

- Accessing SmartDebug from Libero SoC on a design
- Checking the device status
- Checking the flash memory (eNVM) content
- Debugging FPGA array (setting Live Probes, Active Probes, and reading/modifying fabric SRAM content)
- Debugging SERDES designs

Tutorial 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 Evaluation Kit*	Rev C or later
Software Requirements	
Libero® System-on-Chip (SoC)	v11.4SP1
FlashPro programming software	v11.4SP1
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	-
Set the following SmartDebug definition variable before launching Libero SoC v11.4SP1: • SMART_DEBUG_DISABLE_JTAG_RESET = 1	Refer to the following KB for more information: *http://soc.microsemi.com/kb/article.aspx?id=KI8956*

Associated Project Files

Extract the

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_smartdebug_liberov11p4sp1_tu_df

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

- m2gl_SmartDebug_Tutorial—For IGLOO2 Evaluation Kit (M2GL010T)
- m2s_SmartDebug_Tutorial—For SmartFusion2 Evaluation Kit (M2S090T)

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 on page 6. 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 real-time from the device.

The Fabric_Debug block demonstrates the way to use SmartDebug to do 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 is progressing as expected. If there is an error, the output (error) is latched high, as shown in Figure 3 on page 6. This Fabric_Debug block design is used to demonstrate the different silicon built-in capabilities, such as setting Live Probes to monitor in real-time an internal user-selected point on the device. In addition, users can set Active Probes which provides the

capabilities for dynamic asynchronous read and write to a flip-flop or probe point. This will enable users to quickly observe the output of the logic internally or to quickly experiment on how the logic will be affected by writing to a probe point. Finally, the Fabric_Debug design block will be used to demonstrate the SmartDebug capabilities where users can read and modify in real-time the fabric SRAM content.

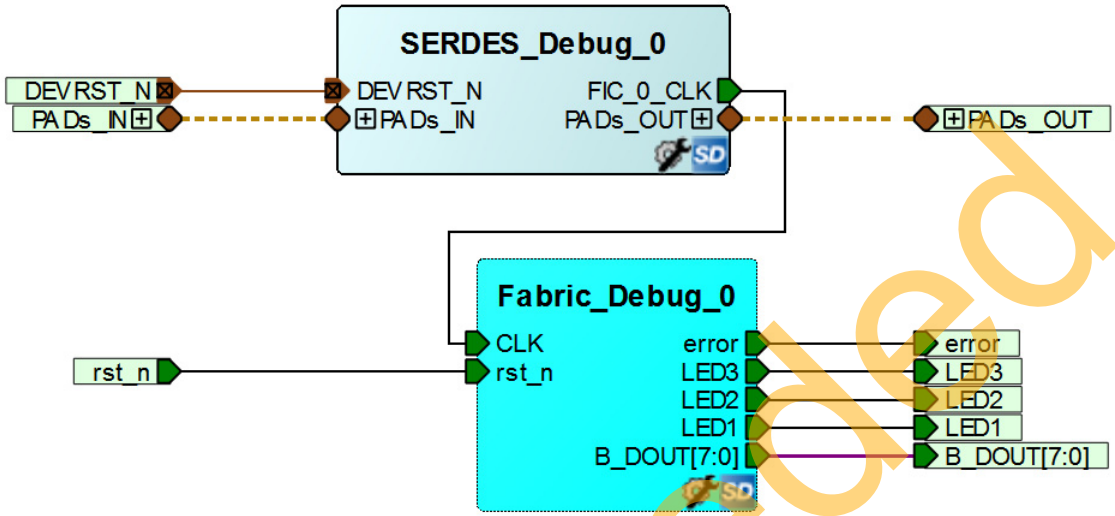

***Figure 1* • SmartDebug Top Level Blocks**

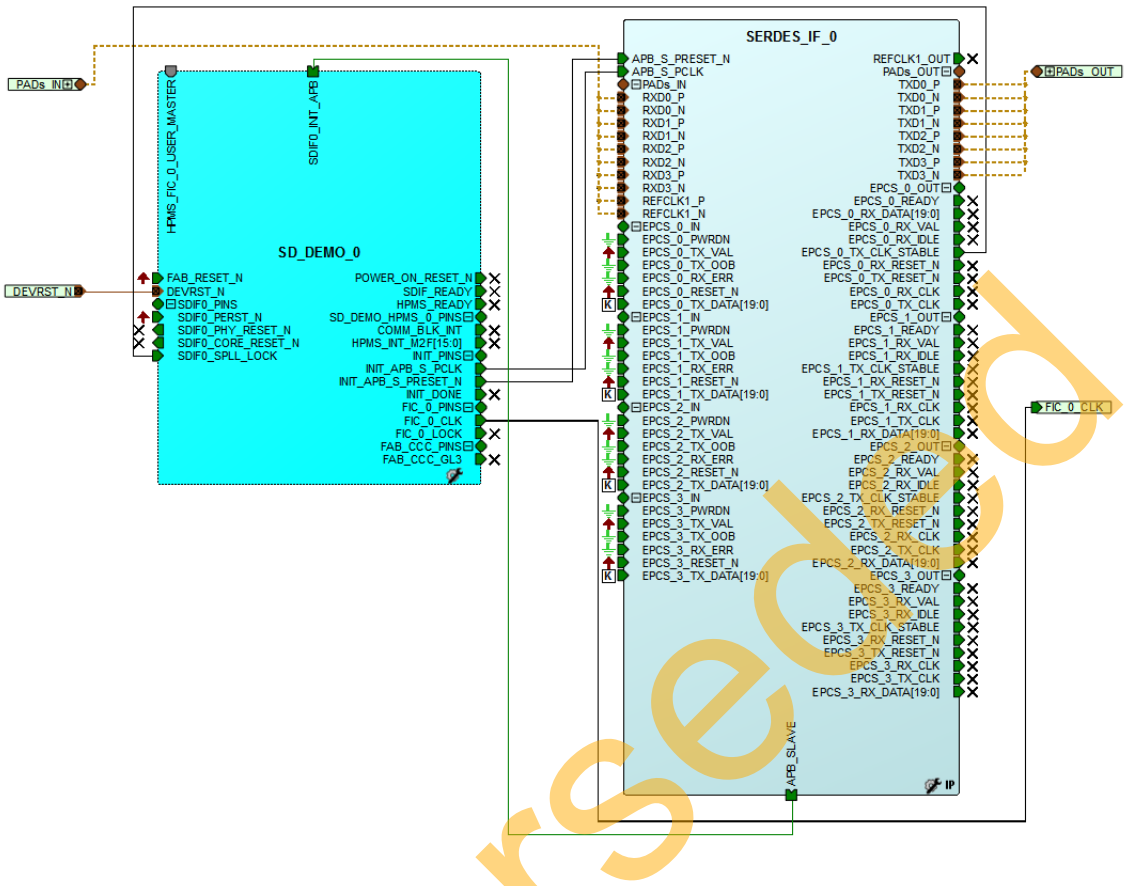

***Figure 2* • SERDES_Debug Overall Design Blocks (M2GL Design Block)**

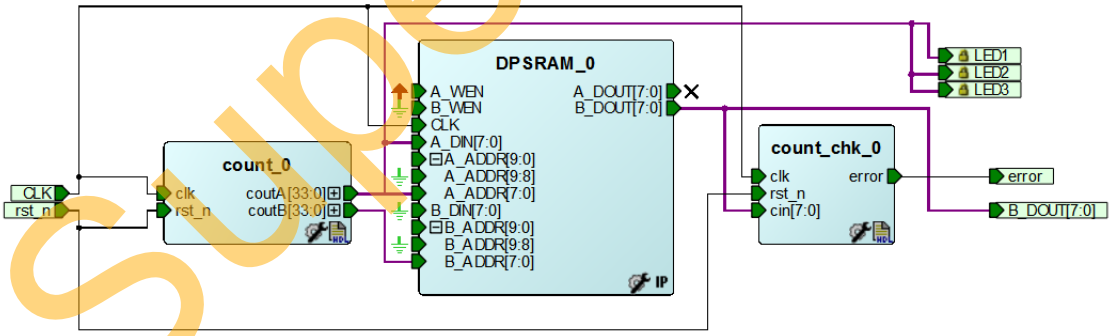

***Figure 3* • Fabric_Debug Overall Design Blocks**

Programming the Device

This section lists the step-by-step instructions on how to program the IGLOO2 or SmartFusion2 Evaluation Board:

1. Connect the **FlashPro4 programmer** to the **J5** connector on the IGLOO2 or SmartFusion2 Evaluation Kit.
2. Connect the **power** supply to the **J6** connector

3. Switch the power supply (**SW7**) to the **ON** position. Refer to the IGLOO2 FPGA Evaluation Kit Board or SmartFusion2 Evaluation Kit Board for more details.
4. Launch Libero SoC v11.4SP1.
5. From the **Project** menu, select **Open Project**. Browse to the folder where the design files were extracted. Refer to the "Associated Project Files" section on page 4. Depending on the selected Evaluation Kit, select the folder and click **Open**.
6. In the **Design Flow** window, select **Run PROGRAM Action**. This will program the design into the device, as shown in Figure 4.

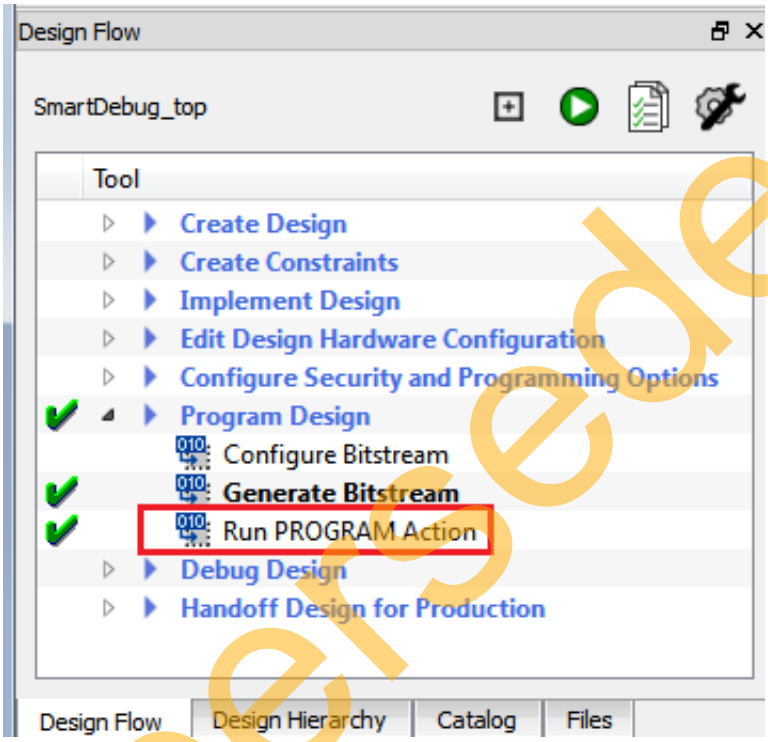

Figure 4 • **Programming the Device**

Launching SmartDebug

Launch SmartDebug by selecting the **SmartDebug Design** option from the **Design Flow** window, as shown in Figure 5.

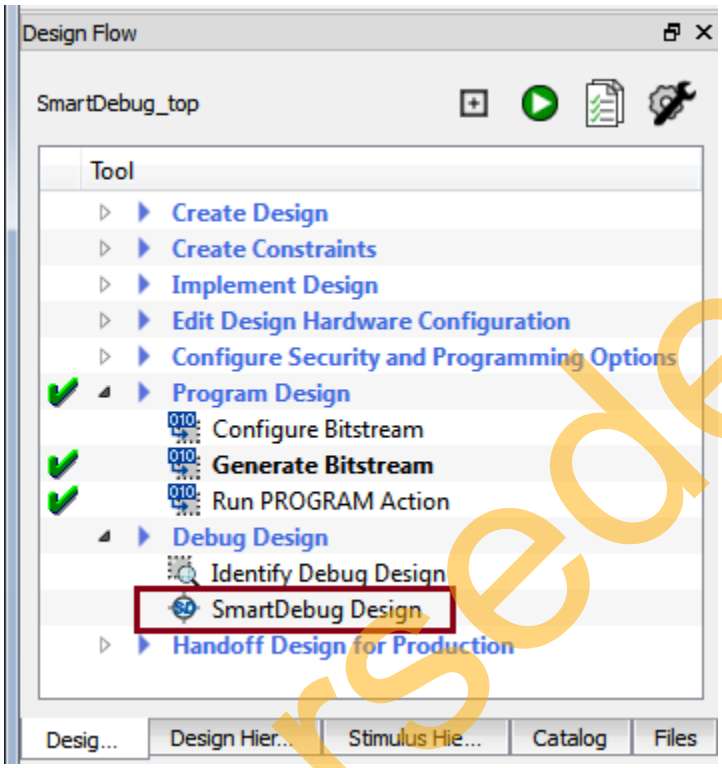

Figure 5 • **Launching SmartDebug Design Tools**

This will open the **SmartDebug** window, as shown in Figure 6.

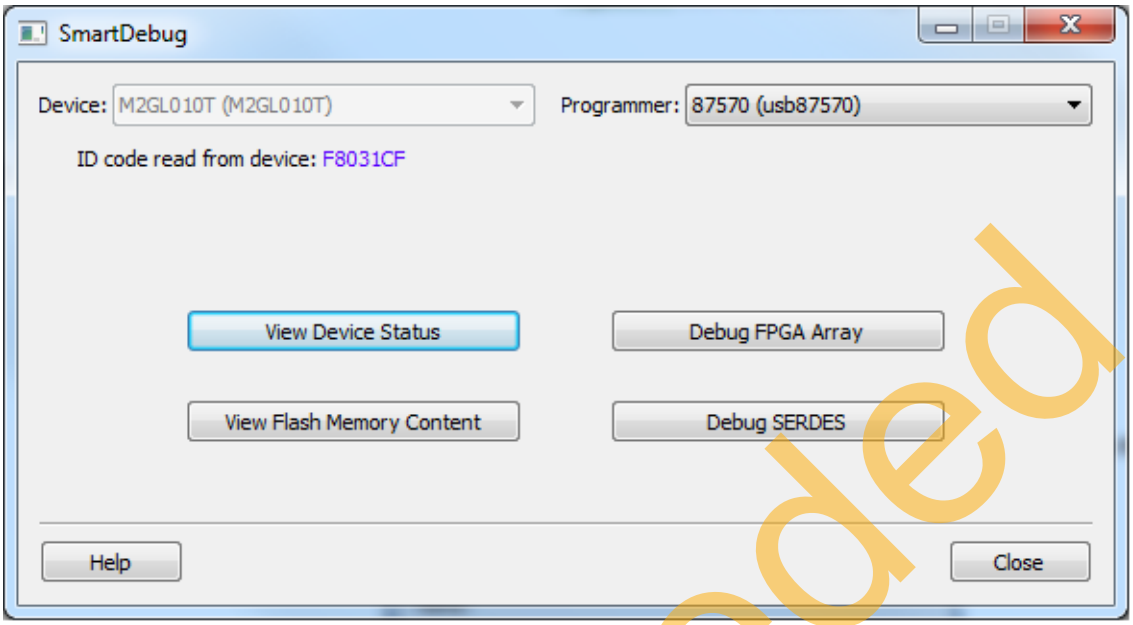

***Figure 6* • SmartDebug Window Debug Options**

Debugging the Design

View Device Status

The View Device Status option provides the device status report. It is a summary of your device information, programmer information, user information, factory serial number, and security information if any are set. Figure 7 shows a sample of the device status information.

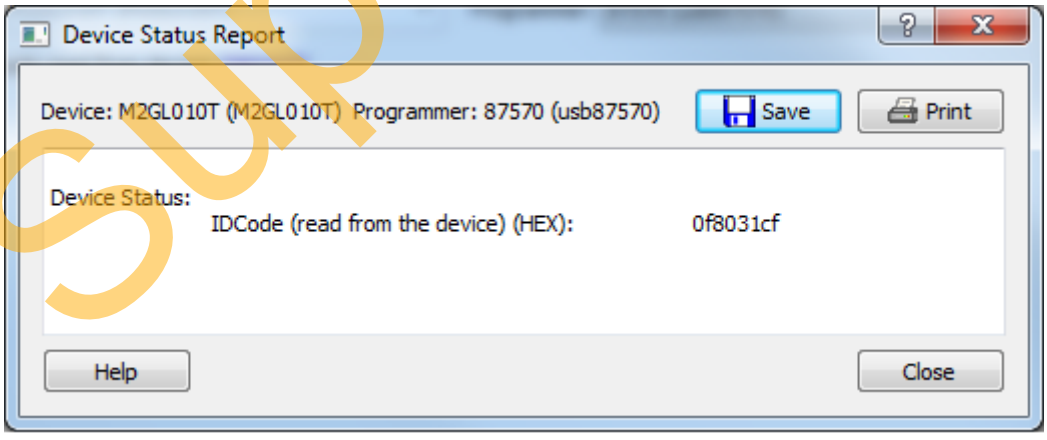

***Figure 7* • Device Status Report Sample**

View Flash Memory (eNVM) Content

The View Flash Memory Content can be accessed from the **SmartDebug** window, as previously shown in Figure 6 on page 9. 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 read back the content of the eNVM, the data to be programmed into the eNVM was defined first. One way to do this is by defining an eNVM data storage client using the eNVM configurator. The client can be stored into any page of the eNVM. Page 64 was used here for demonstration purposes. Figure 8 shows an excerpt of the data storage client content that was defined in the eNVM.

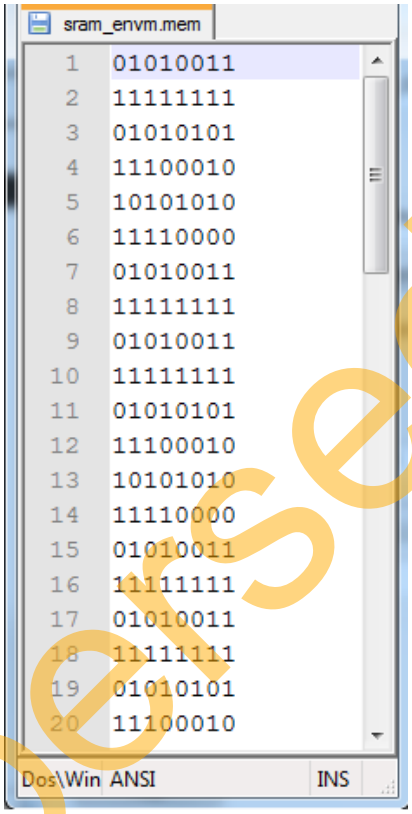

***Figure 8* • Memory File Content Saved into the eNVM**

The content of eNVM is retrieved from the device, displayed, and is equivalent to what is shown in Figure 8.

The eNVM content can be read in real-time from the device as follows:

1. Using the SmartDebug window, select the **View Flash Memory Content** option. The Flash Memory window opens, as shown in Figure 9.
2. Since the data storage client is stored into page 64, specify the **Start Page** and **End Page** as page number **64.** Page 64 is used here for demonstration purposes.
3. Select the **Read from Device** option.

Note: When targeting the M2S090 device, SmartDebug does not support Flash Memory content read at present.

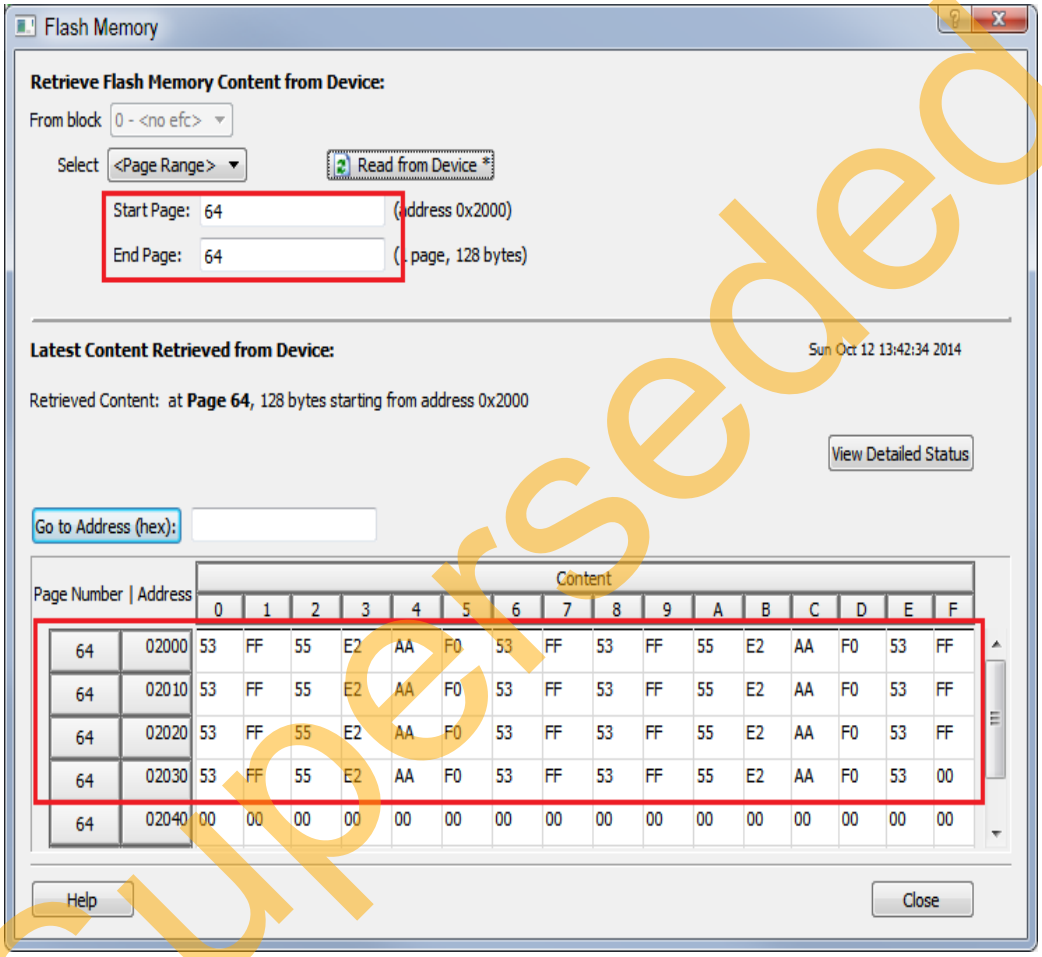

Figure 9 • **Flash Memory (eNVM) Content Read from the Device**

Debug FPGA Array

SmartFusion2 and IGLOO2 devices have built-in probe points that greatly enhance the ability to debug logic elements within the device through the Live Probes and Active Probes features. The enhanced debug features implemented into the devices give access to any logic element and enable users 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. This step 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 previously shown in Figure 6 on page 9. Selecting the Debug FPGA Array option opens the **Debug FPGA Array** window, as shown below in Figure 10.

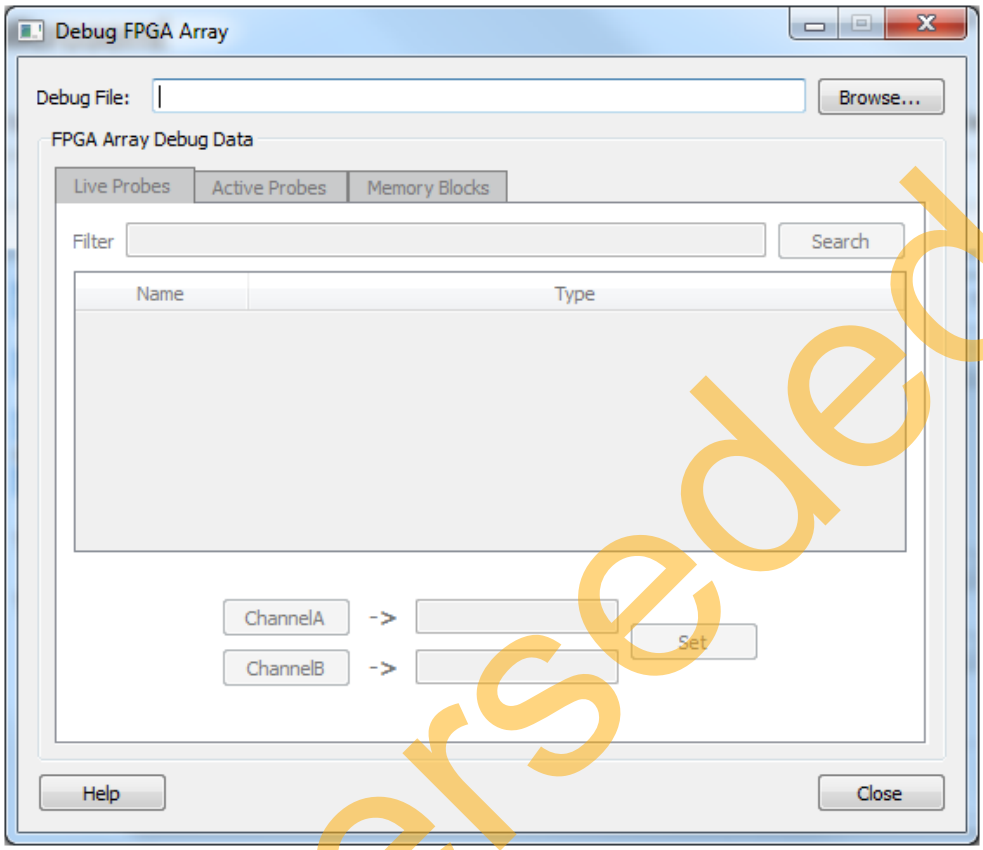

***Figure 10* • Debug FPAG Array Window**

Libero SoC generates the Debug File, *<projectName>_debug.txt,* during Place and Route and stores the file into the *<project path>\designer* folder. The Debug File contains information used by SmartDebug mainly for mapping the user design names to their respective physical addresses on the device. It also contains other information used during the debug process.

Before starting the FPGA Array Debug, the Debug File must be specified.

Select the **Browse** button and then select the **SmartDebug_top_debug.txt** file: *Debug File = <project root>\designer\SmartDebug_top_debug.txt.*

Once the file has been selected, the window will populate the **Live Probes** tab with the available debug points, as shown in Figure 11.

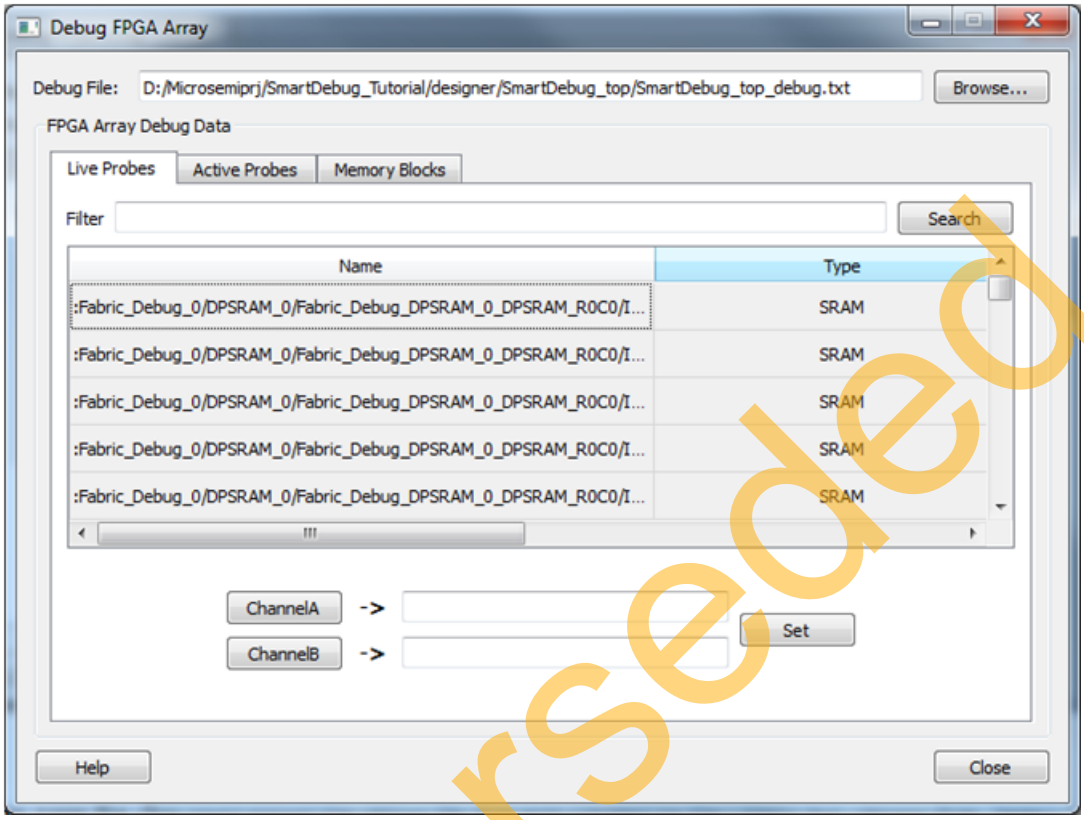

Figure 11 • **Specify Debug File Location**

In the next few steps, this tutorial demonstrates how to use the Live Probes, Active Probes, and the Memory Block debugging features.

1. Specifying Live Probe Points

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 then be sent to the two dedicated probe pins (PROBE_A and PROBE_B), then to the oscilloscope or even redirected back to the FPGA fabric. The probe points location can be changed without having to recompile or reprogram the design. The probes can capture data at a speed of up to 100MHz.

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. The pins can be reserved for probing by selecting the option **Reserve Pins for Probes** in the **Project Settings** window, as shown in Figure 12.

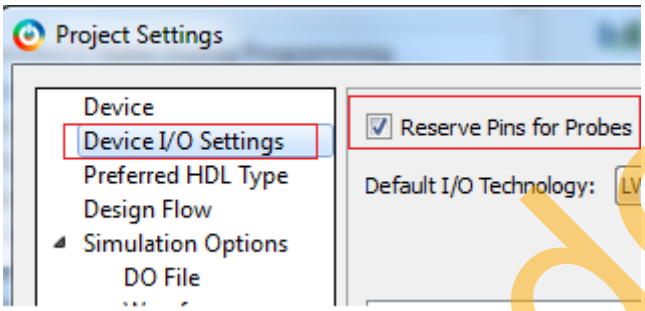

***Figure 12* • Reserving Probe Pin for Probes**

Furthermore, you can identify the probe pin on your package by looking at the pin description document for that particular package. Another option is to check the Function column in the Package Viewer of the **I/O Editor** in the Libero SoC software, as shown in Figure 13.

M2GL010T or M2S090T has two dedicated 484 FBGA pin numbers (Y10 and W10), which can be used for probing, as shown in Figure 13.

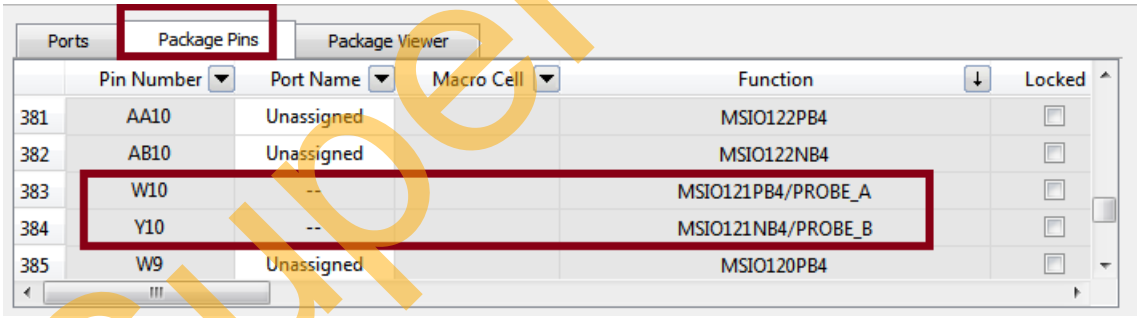

***Figure 13* • Identifying Probe Pins using Package Viewer Inside I/O Editor**

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 M2S090T Evaluation Kit Rev Don J29 jumpers.

The Live Probes tab (Figure 14) shows the probe point name and pin type (SRAM, Logic, or I/O). Once a point is selected it can be assigned to either ChannelA (PROBE_A) or ChannelB (PROBE_B) as follows:

1. Select the **point** that you want to probe
2. Select the **channel** on which you want to probe the selected point
3. Select **Set**

Figure 14 shows an example of setting two probe points: coutA[23]:Q and coutA[24]:Q to be probed on ChannelA and ChannelB respectively.

A message will be printed in the Log window of the Libero SoC indicating which signals were 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 the channels have been set, SmartDebug configures the ChannelA and ChannelB I/Os to monitor the desired probe points. On the IGLOO2 Evaluation Kit Rev D Board, the PROBE_A and PROBE_B pins are exposed on the J29 connector. You can connect an oscilloscope to these probe points and monitor the signals that are assigned to be probed. The maximum number of simultaneous probes is two internal signals. There is also a Filter box to filter through the Net Names. As you begin typing in the Filter box, the Net Name table only shows results for the queried names.

Note: The Active Probes WRITE will overwrite the settings of the Live Probe channels (if any).

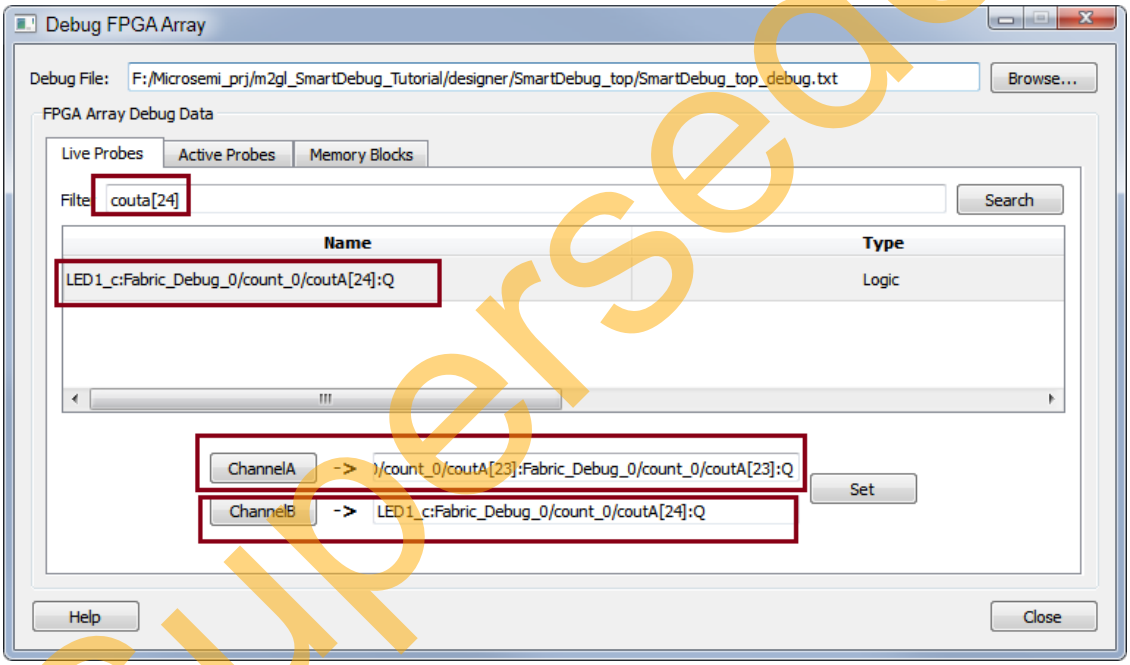

***Figure 14* • Live Probes Channels Assignments**

2. Active Probes

Active Probe allows dynamic asynchronous read and write to a flip-flop or probe point. This will enable a user to quickly observe the output of the logic internally or to quickly experiment on how the logic will be affected by writing to a probe point. The following steps will demonstrate how to select a specific set of probe pins reading the current value and then writing different values.

Selecting Active Probes

1. Select the **Active Probes** tab from the **Debug FPGA Array** window.
2. Once inside the Active Probes tab, click on the **Select Active Probes** button to define the internal points to monitor, as shown in Figure 15.

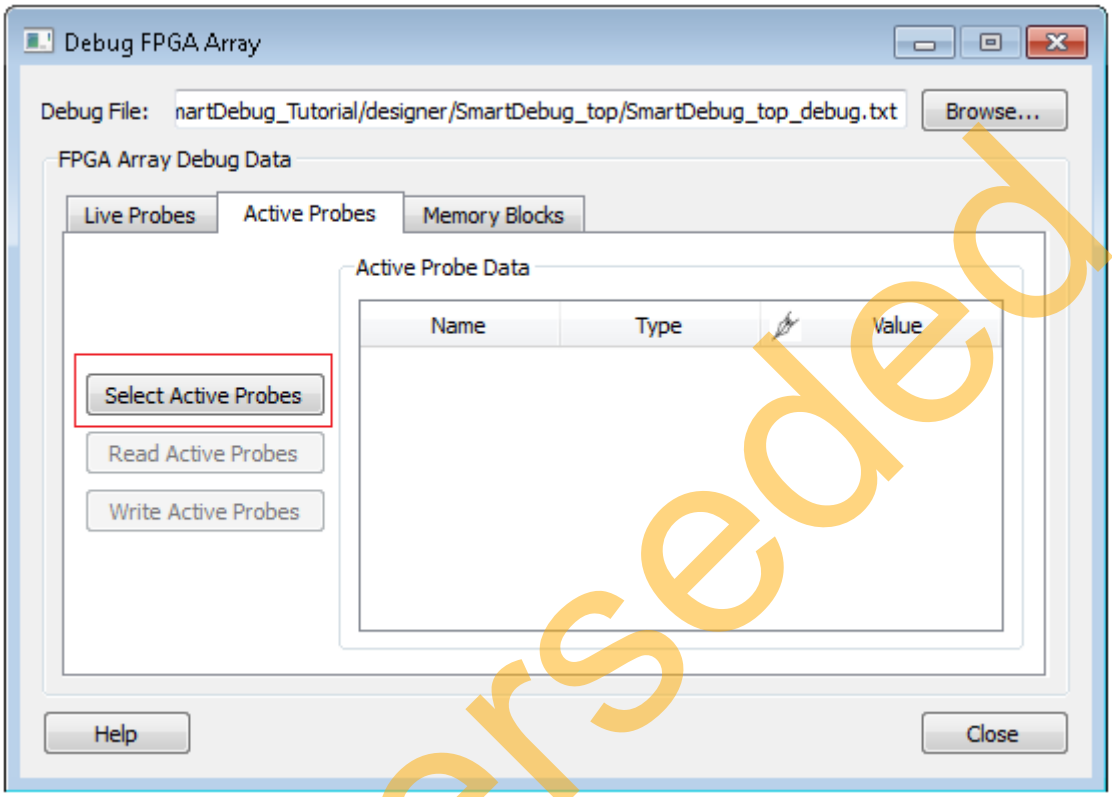

***Figure 15* • Selecting Active Probes From the Design**

3. Select Active Probes opens a window that shows all the available probe points in the design. For this tutorial we are going to monitor the following points:

- Three bits of the counter output coutA.
- The monitoring signal “error”, which is also connected to the LED (H5) on the board. If the LED is ON, it indicates that it does not have any error in the count.
- An internal register “sync”

To find these points in the list of available probe points use the Filter control, as shown in Figure 16 on page 17.

Note: Since Active Probe only deals with individual signals, the coutA bus segment will be broken up into three separate probe lines.

4. Select the desired points and click on **Add** to move to the **Selected Probe Points** window and click **OK,** as shown in Figure 16.

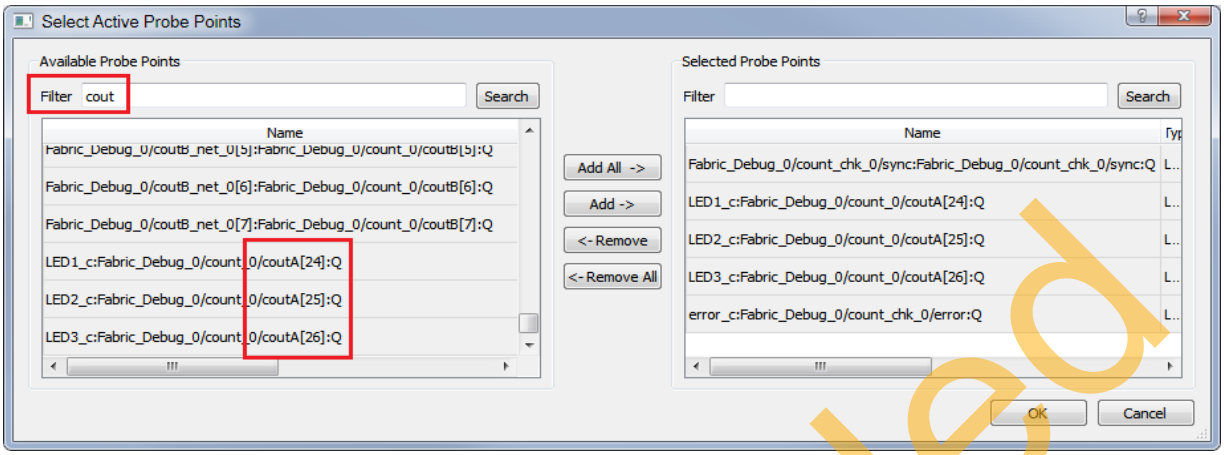

Figure 16 • **Selecting Desired Points to Read**

Reading Active Probes

Once all of the probes points have been specified, select **Read Active Probes** to gather the current values of the internal signals. Figure 17 shows the results similar to a first read of the design.

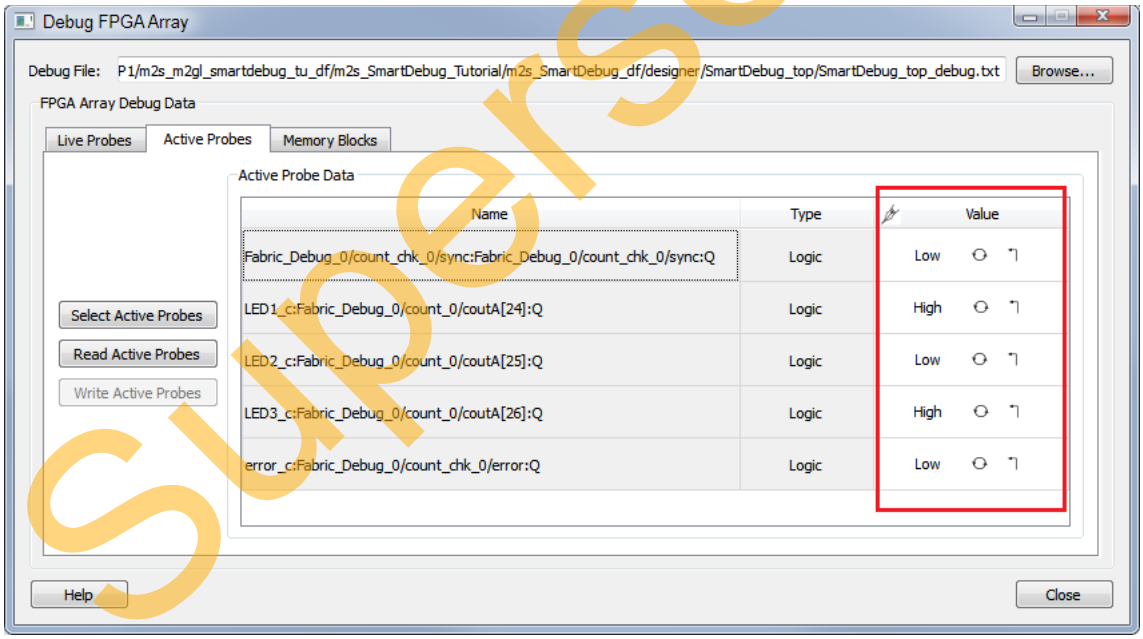

Figure 17 • **Active Probe Readings**

Note: To toggle the states between High and Low, press , and to reset the value, press .

The coutA bus will be constantly counting therefore, the value may be different than what is seen on the target platform. The value of sync should be High indicating that the checker on the read side of the DPSRAM has latched onto the count pattern.

Also, the error signal should be **Low** indicating that there have been no errors in the counting pattern.

When the design is held in reset the sync signal goes Low and waits for a specific pattern from the DPSRAM to sync up the counters. To watch this happen, hold **SW2**(rst_n) down and select **Read Active**

Probes. The sync signal will be Low as the design is now held in reset. Release **SW2** and read again, the sync is now High.

3. Fabric SRAM Memory Debug

To view the contents of the Large SRAM in this design select the **Memory Blocks** tab, as shown in Figure 18.

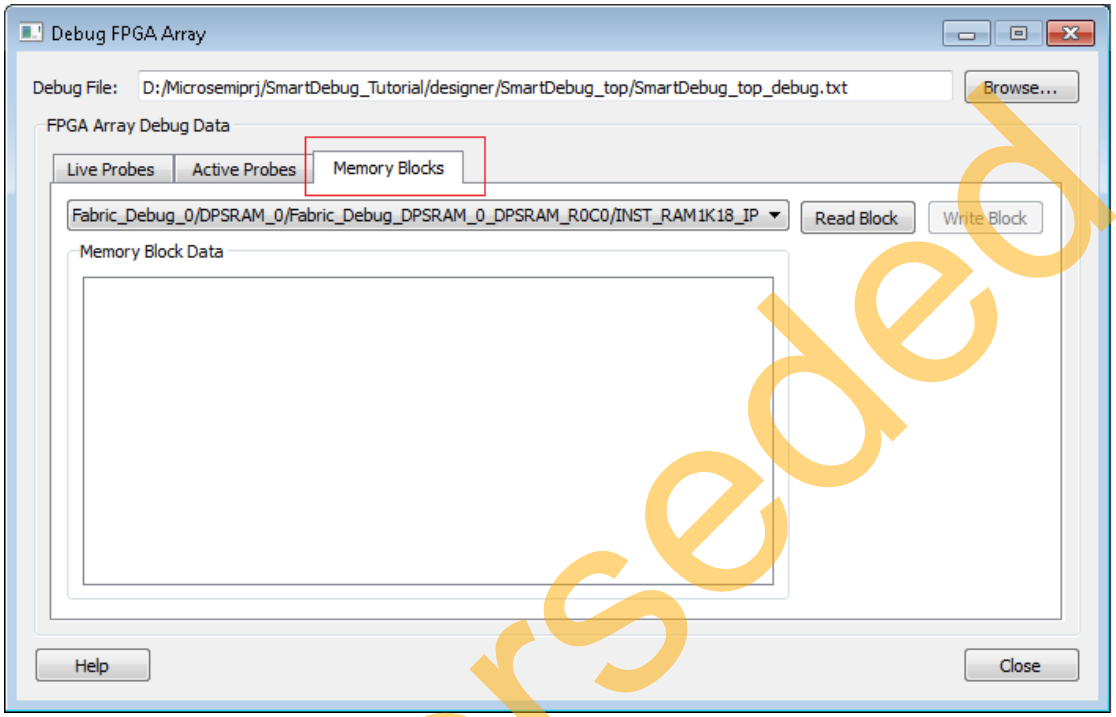

***Figure 18* • Memory Blocks Tab**

This design contains a single Large SRAM, named DPSRAM_0, and it is the only one available in the drop-down list. Select this block and select **Read Block**.

The contents of the DPSRAM_0 will be displayed, as shown in Figure 19. See the counting pattern that is loaded into the RAM.

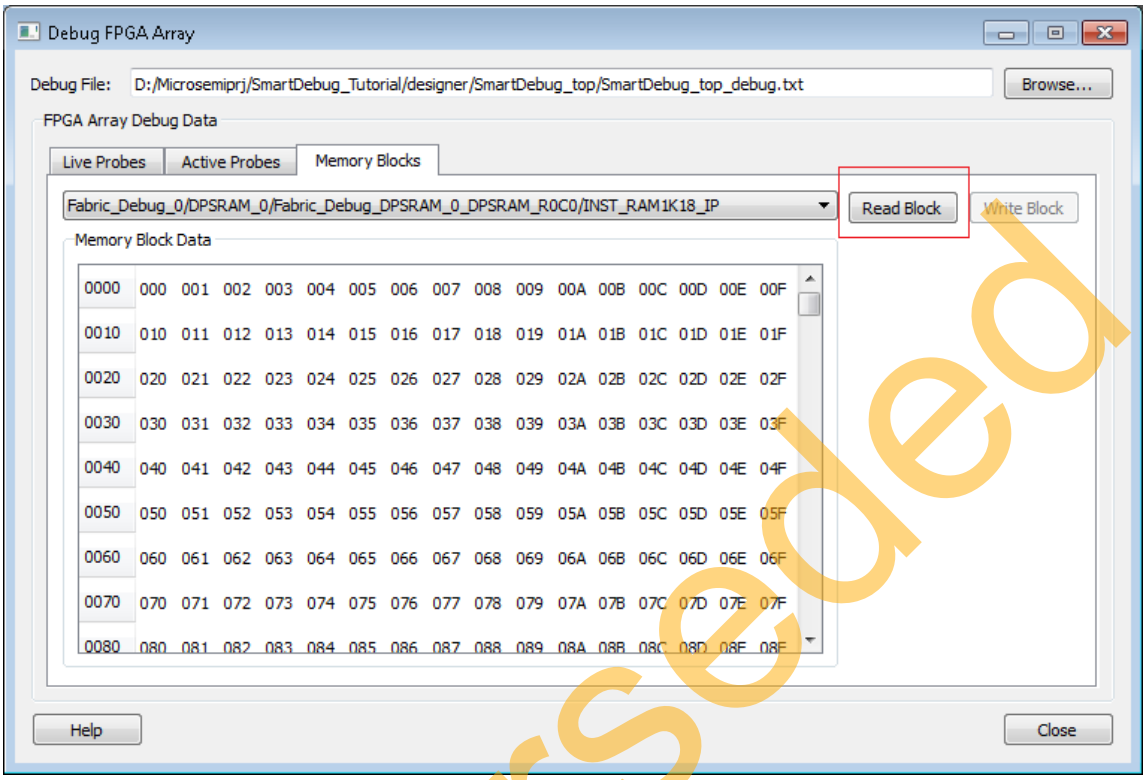

***Figure 19* • DPSRAM_0 Contents**

Forcing a Design Modification

The design reads the contents of the DPSRAM and compares it to a synchronized counter in the checker which is looking for errors. If the contents of the DPSRAM is modified it will break the count pattern and cause an error in the checker.

Modify the contents of the DPSRAM and force an error as follows:

1. Read the memory content, as shown in Figure 19
2. In the Active Probe tab, read the probes
3. Power cycle the board (turn off and on the board power)

Note: Since SmartDebug is accessing the SRAM at the same time the counter is writing to the SRAM (due to a known issue) the error LED will go off. To work around this current issue, turn off and on the board power before proceeding. There is no need to restart SmartDebug.

4. Once the board comes back up, read the active probe again. There will be an error, which can be ignored.
5. Read the active probes the second time. On the second read there will be no errors and the LED should come on indicating no errors.
6. Go to the **Memory Blocks** tab, select an entry and double-click. Each entry is 9-bits wide.

7. Modify the value from the current value to break the count pattern, as shown in Figure 20.
8. Select **Write Block** to write the modified value to the SRAM.
9. The error LED light should go off, indicating an error in the counting pattern.

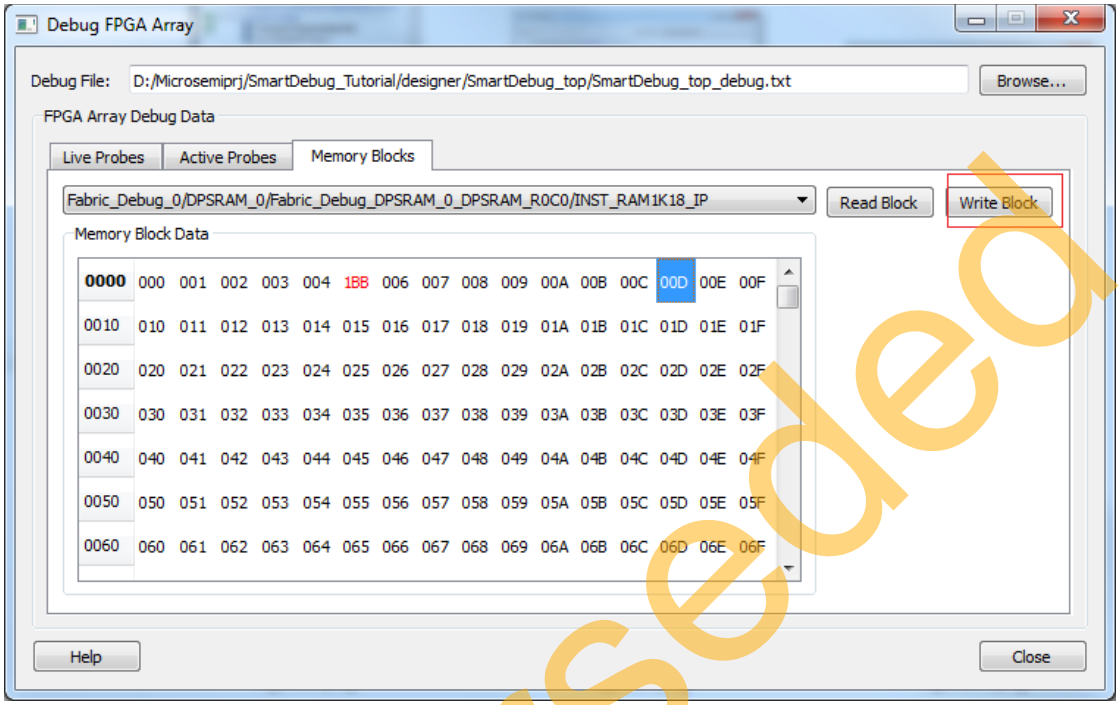

***Figure 20* • Modifying DPSRAM Contents**

In the **Active Probes** tab, perform a read and you can see that the error signal is now High, refer to Figure 21.

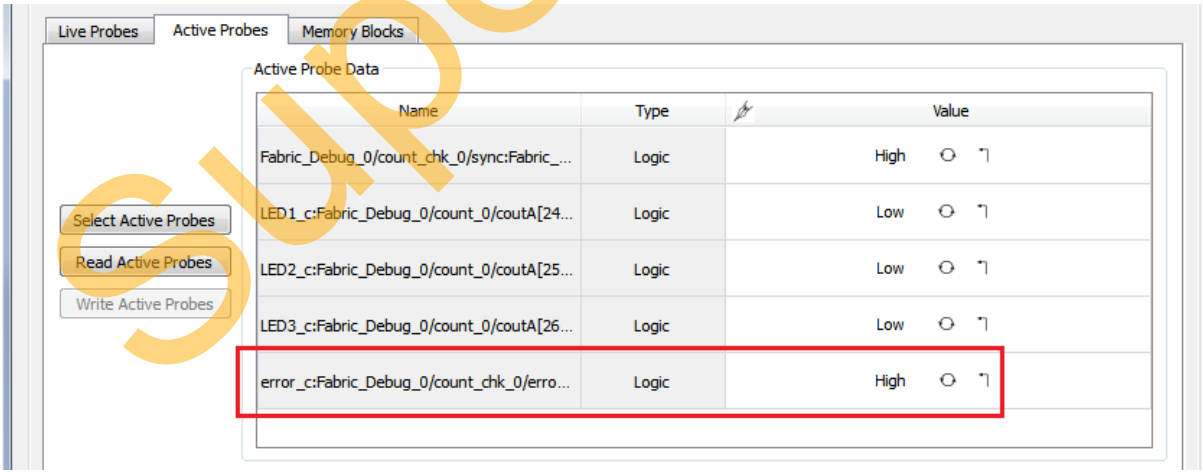

***Figure 21* • High Error Signal after Forcing an Error**

SERDES Debug

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

- Real-time access to SERDESIF Block control and status registers
- Provide testing functions with pseudo-random binary sequence (PRBS) or constant pattern generators and checkers
- Run link tests with various loop back options
- Provide overview for tuning many combinations of physical medium attachment (PMA) analog settings to find the optimal set for a particular SERDES channel

1. Select **Debug SERDES** from the SmartDebug Graphical User Interface, as depicted in Figure 22.

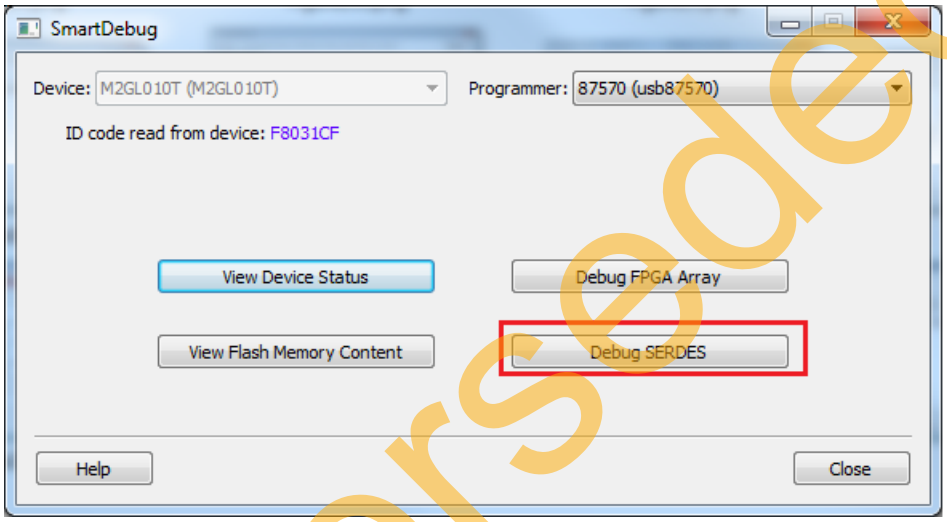

***Figure 22* • Debug SERDES Operation Selection**

2. The **Configuration** tab will auto-identify and populate which SERDESIF is used in the design and the lanes and how they are programmed and powered-up. The status of each lane is shown as well as the programmed lane mode. This example demonstrates the use of SERDESIF_0 block, as well as the lock status of the TXPLL and RXCDR. This window can be updated through the **Refresh Report** button, refer to Figure 23.

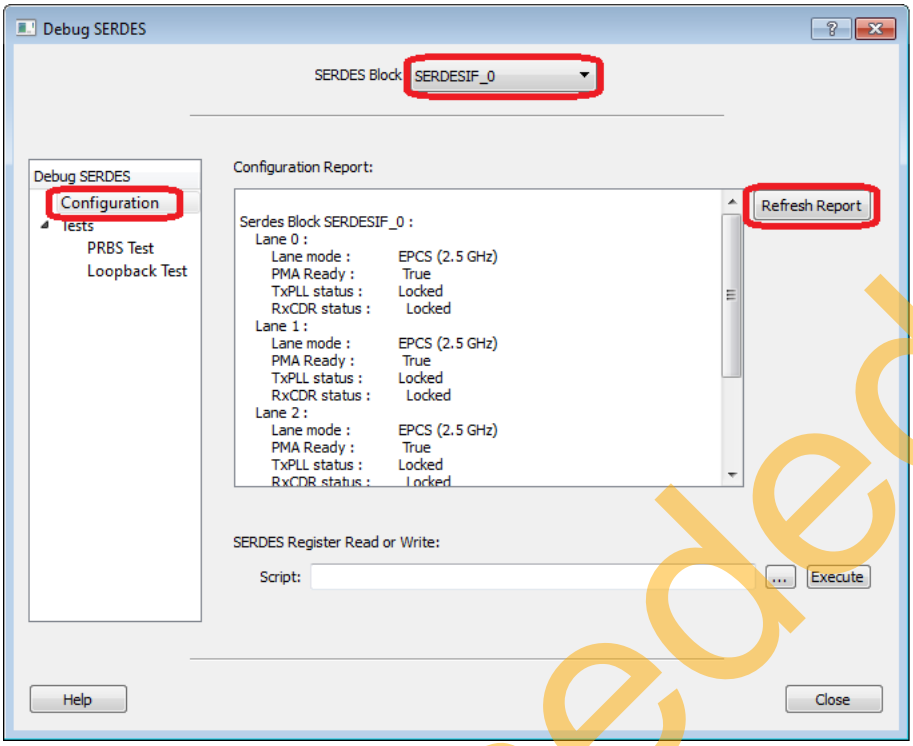

Figure 23 • **SERDES Configuration Tab**

3. The **PRBS Test** tab provides several capabilities for each Lane of the SERDES Block. The information is provided per-channel, based on the SERDES Lane selected within the GUI. For example, select **Lane 0**, select the **Near-end Serial Loopback test type**, and select **PRBS7 Pattern**. This test will generate and check PRBS7 data without going off-chip, as shown in Figure 24.

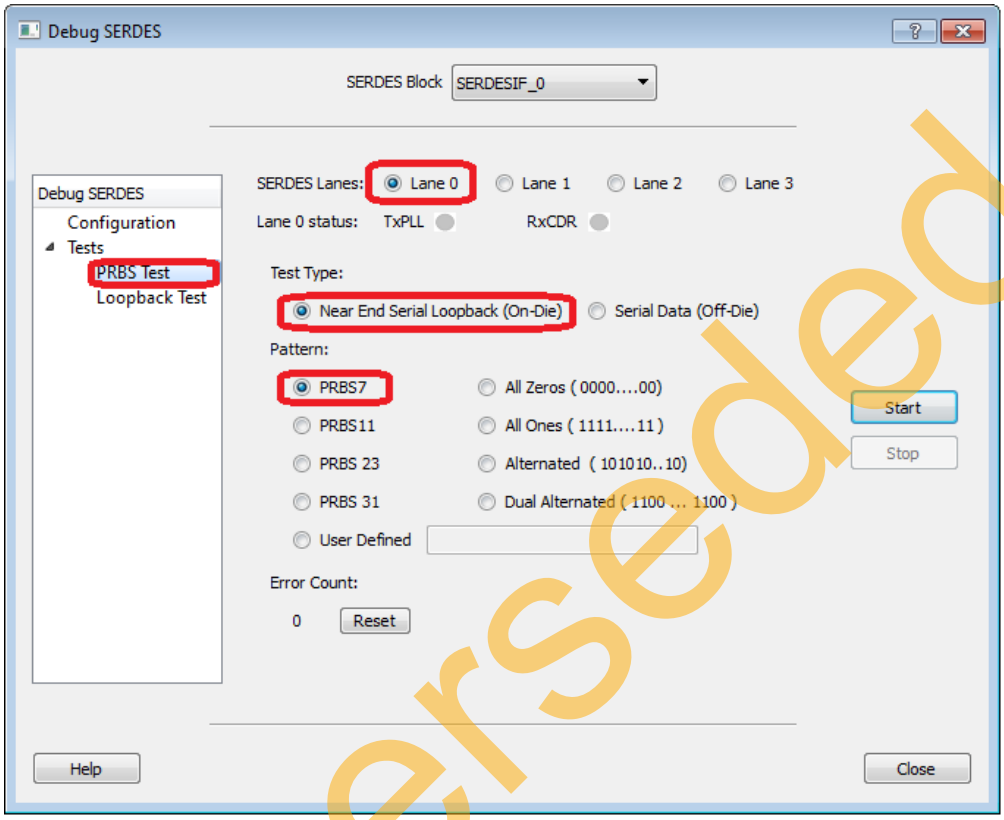

Figure 24 • **SERDES Test Tab**

As shown in Figure 25, the Lane 0 status is indicated after Starting the test. The green LEDs indicate the lock status of the TXPLL and RXCDR for the selected Lane.

In this example setup, the datastream is expected to see zero errors as the datapath does not go off-chip while using the Near-end Serial Loopback. The Error Count will increment up to 255. The Reset button will clear the count and the counter will continue to count while the test continues to run.

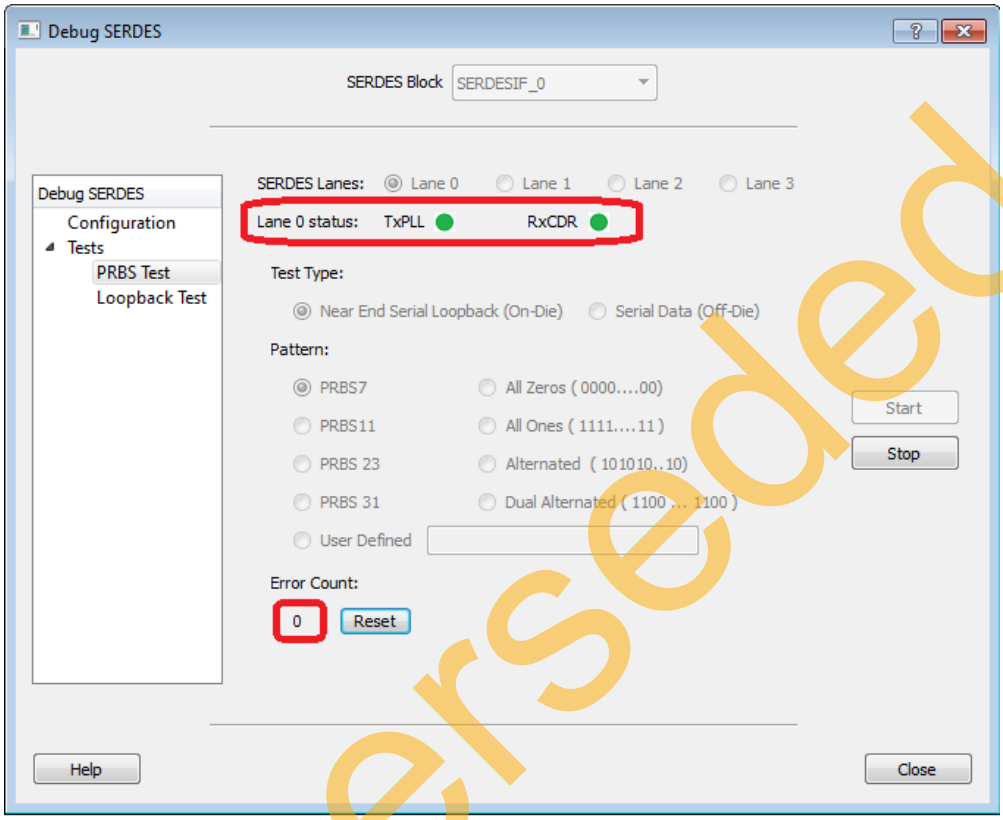

Figure 25 • **SERDES Link Status**

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

4. Stop the test and change the Test Type to **Serial Data (Off-Die)**. Restart and observe the Lane 0 error counter. This is due to the fact that the data is no longer looping between Tx and Rx. Lane 0 is not looped together on the PCB. In this case, the error count will increment up to the maximum 255, as shown in Figure 26.

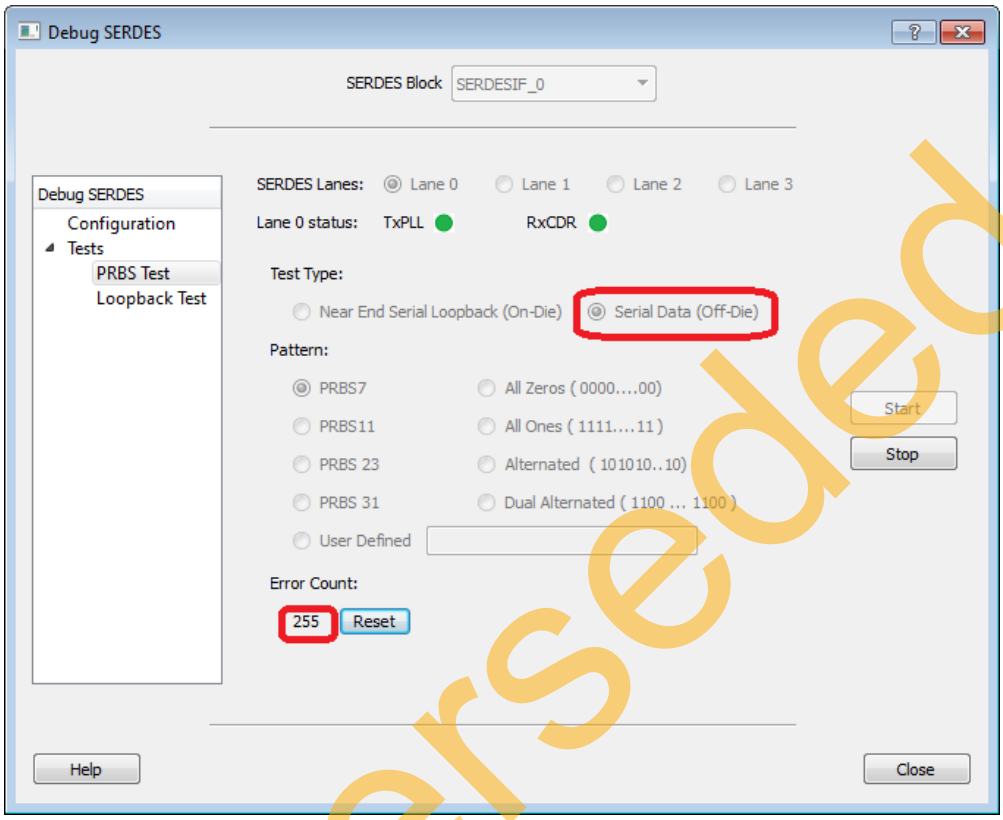

Figure 26 • **Sending Serial Tx Data Off-Die**

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 being transmitted and received intact. The example demonstrates this, select **Lane 1**, select the **Serial Data test type**, and select **PRBS7 Pattern**. This test will generate and check PRBS7 data going off-chip and folded back on the PCB to the receiver, refer to Figure 27.

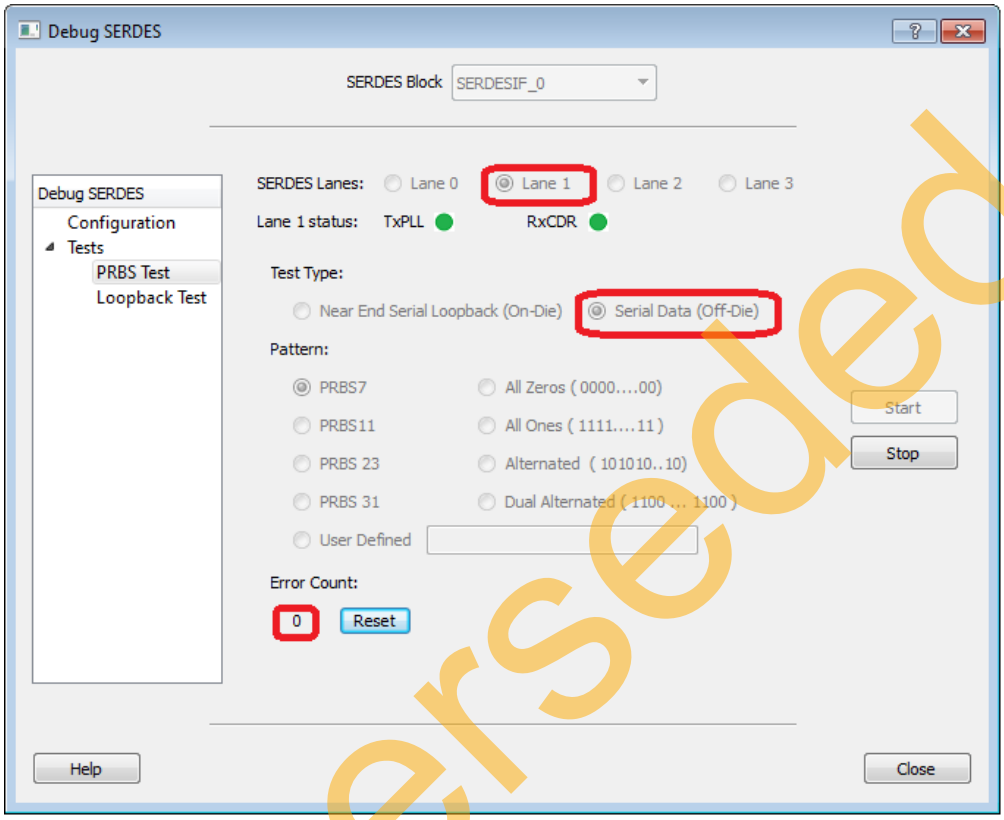

Figure 27 • **Lane 1 Transmitting Data Through On-Board Loopback**

6. The Tx and Rx channels of Lane 2 can be interconnected in a loop-back configuration using coax cables. In this example, as shown in Figure 28 and Figure 29, after connecting a pair of high-quality 50-Ohm SMA cables to the SMA connections on the Evaluation Kit board, the SERDES debug can be used to send data off-board and check for errors. This requires selection of Lane 2 and Serial Data (Off-Die). A PRBS7 pattern is chosen in the example test.

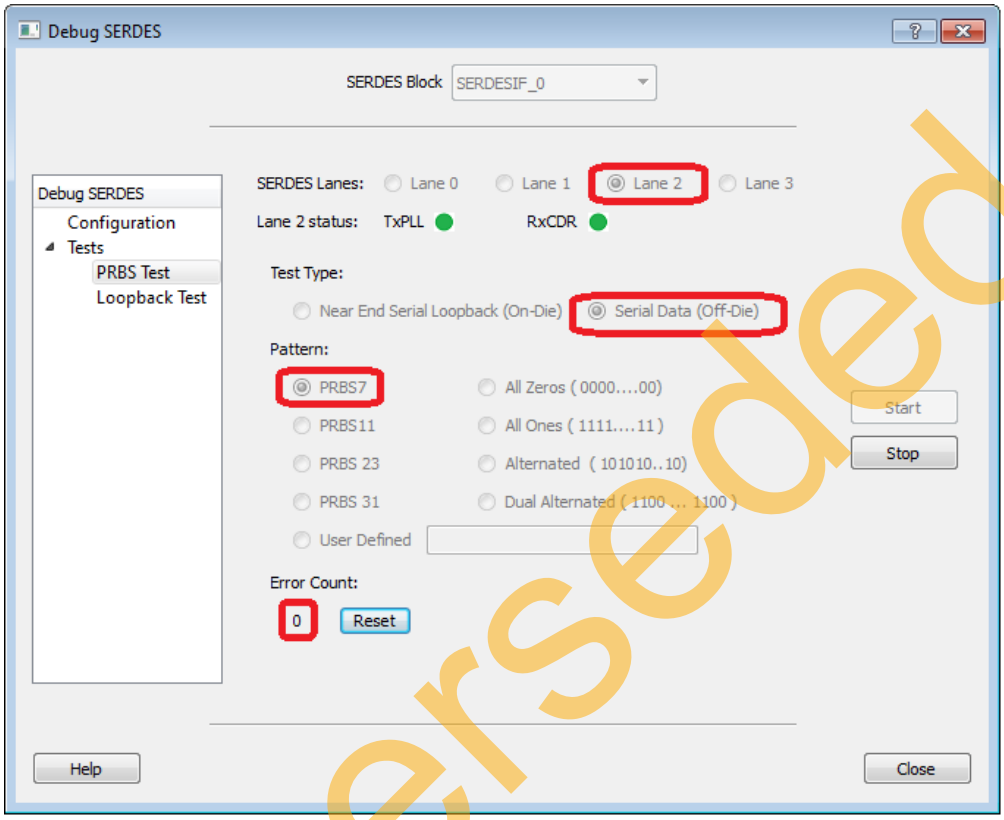

***Figure 28* • External Cable Loopback**

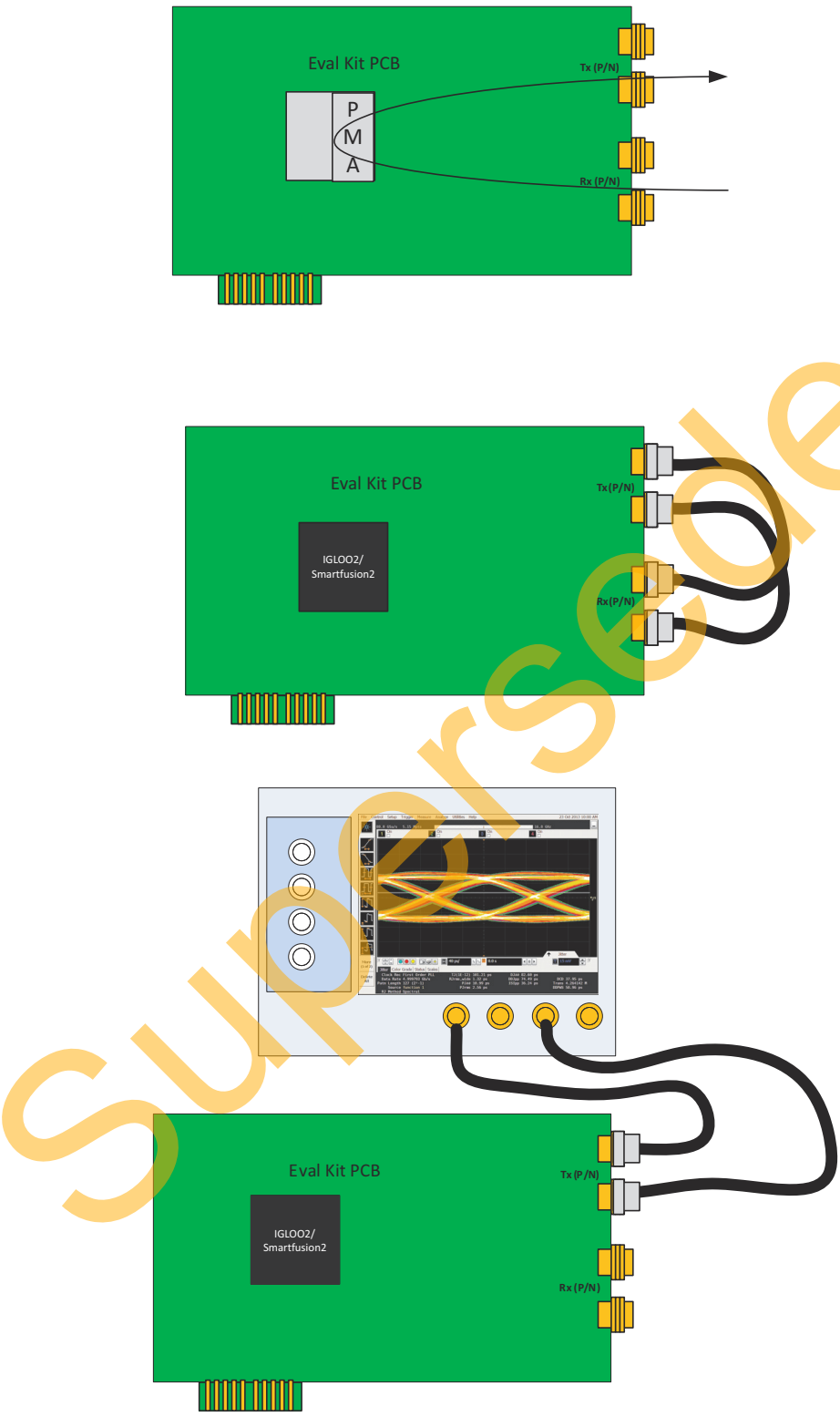

Figure 29 • **Evaluation Kit Board with External Coax Loopback Setup**

7. The Lane 2 SMA test connections can be used for interconnecting with high-speed coax cables to test equipment or other test fixtures like test backplanes. In the example shown in Figure 30, when the Lane 2 test is started without any means to connect the Tx and Rx together, the Error Count will increment; as the link is broken between the pattern generator and the checker. This setup will send a data pattern of the board for analysis on the test equipment, such as a high speed oscilloscope does

Note: SMA Male-to-SMA Male Precision Cables, such as Pasternack Industries part number PE39429-12 (or equivalent), are recommended.

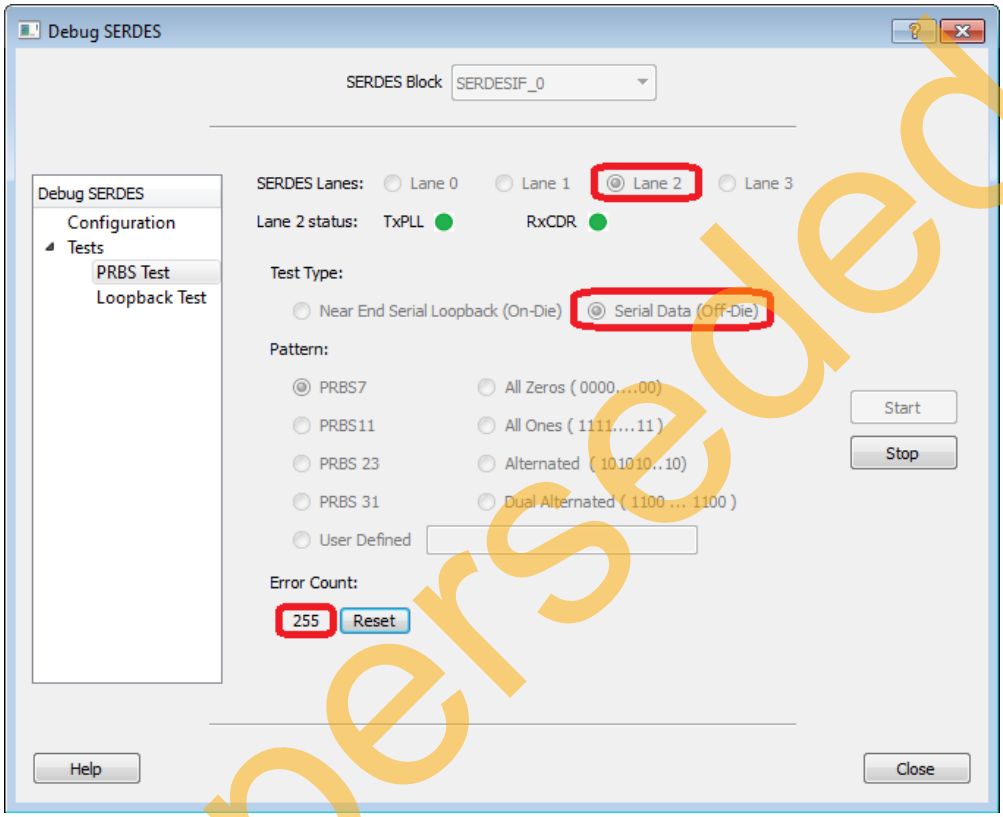

***Figure 30* • Lane 2 Transmitting Data Off-Board**

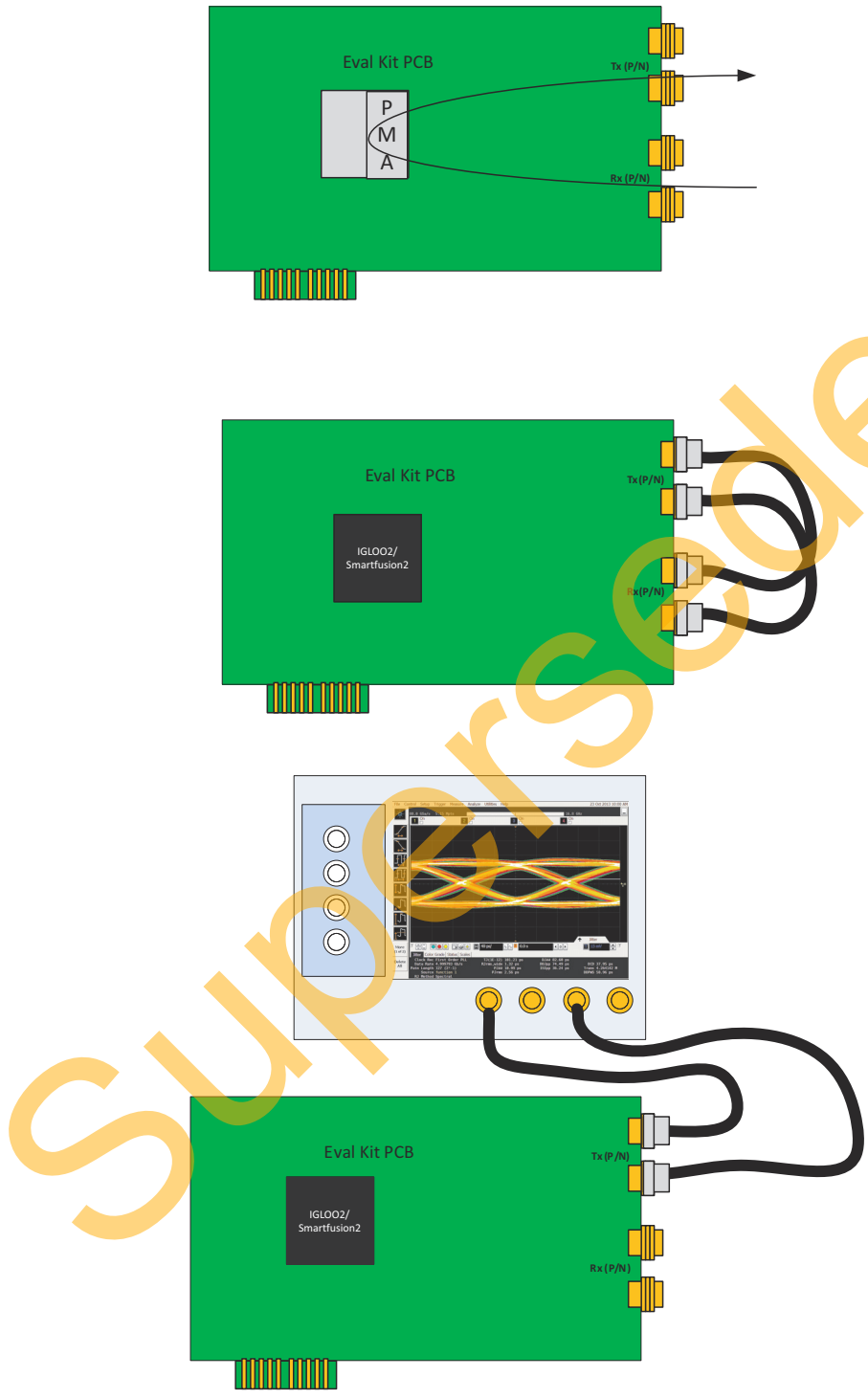

Figure 31 • **Connecting Lane 2 to the Test Equipment**

Several test patterns are available from the test pattern generator. They include several PRBS and constant patterns. Not all patterns are suited for all applications. For instance, all ones or all zeros will not be useful with AC-coupled channels; as the DC-offset is removed producing a signal that is at-ground. If using the generator to send data to the test equipment, some test equipment cannot tolerate the long run-lengths of some of the PRBS patterns. PRBS7 is a very typical pattern for testing signal integrity in communication applications.

Bit Error Rate (BER) is simply counting the number of errors over time to provide a level of confidence of 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 GUI provides an error counter allowing the user to do any BER test.

An online calculator can determine how long to run a pattern test based on the target BER.

Far-End Loop Back Support

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

In the example below, Figure 32 on page 32 and Figure 33 on page 32, by using the Evaluation Board traffic is received from a far-end transmit source, such as another device or test equipment. It is received into Lane 2 and looped back out the transmitter.

This is accomplished by selecting SERDES Lane 2, selecting the PCS Far End PMS Rx to Tx Test Type, and Start to complete the setup.

Traffic entering the SMA connectors on Lane 2 of the Evaluation board will be observed coming off the board on the Tx SMA connectors.

Note: In this test, the IGLOO2 Evaluation board must use the same SERDES reference clock as the far-end. The data path through the SERDEIF goes through the CDR and reclocks the data to the local REFCLK. This requires 0ppm difference between the far-end clock source and the Eval-Kit clock source. For this, use the SMA inputs [designators J17 & J21] of the board rather than the local on-board oscillator, as the input of the SERDES REFCLK.

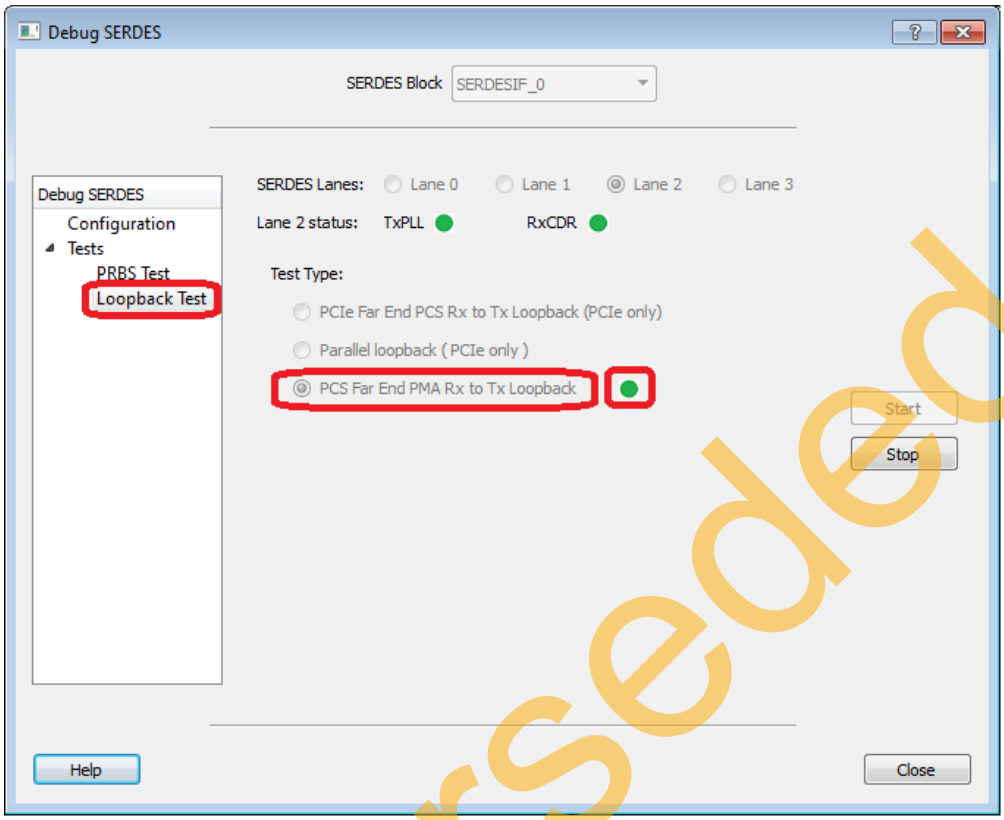

***Figure 32* • PCS Far-End Rx to Tx Loopback**

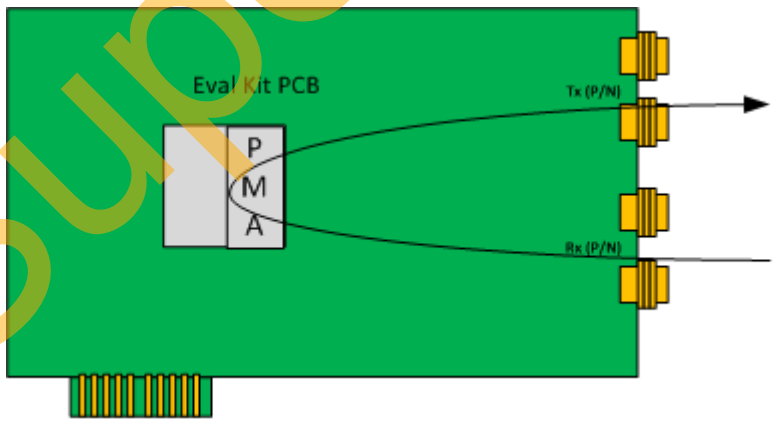

***Figure 33* • Far-End Loopback on the Evaluation Board**

Tcl Support

The SERDES Debug tool set permits execution of Tcl scripts. This scripting capability 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 will be 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 will be 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 will be 0x40028000 (SERDESIF_0 base) + 0x1000 (lane 0 offset) + 0x190 (register offset) = 0x40029190

Reference the *IGLOO2 High-Speed Serial Users Guide or SmartFusion2 High Speed Serial User Guide* for register map details.

Attempt only to read the lanes which 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 which lanes are accessed in the design. In this example, simply reading the CONFIG_PHY_MODE_1 register and passing its value and the associated offset will target the correct lane.

Note: Some SERDES PMA register settings will only be updated after assertion of a PHY_RESET or writing to the UPDATE_SETTINGS register.

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

From the Configuration Tab GUI, there is a dialogue box to import an executable Tcl script. The script will contain commands to write/read registers in using a flattened top for most address mapping. Simply browse to the Tcl script file and Execute, refer to Figure 34.

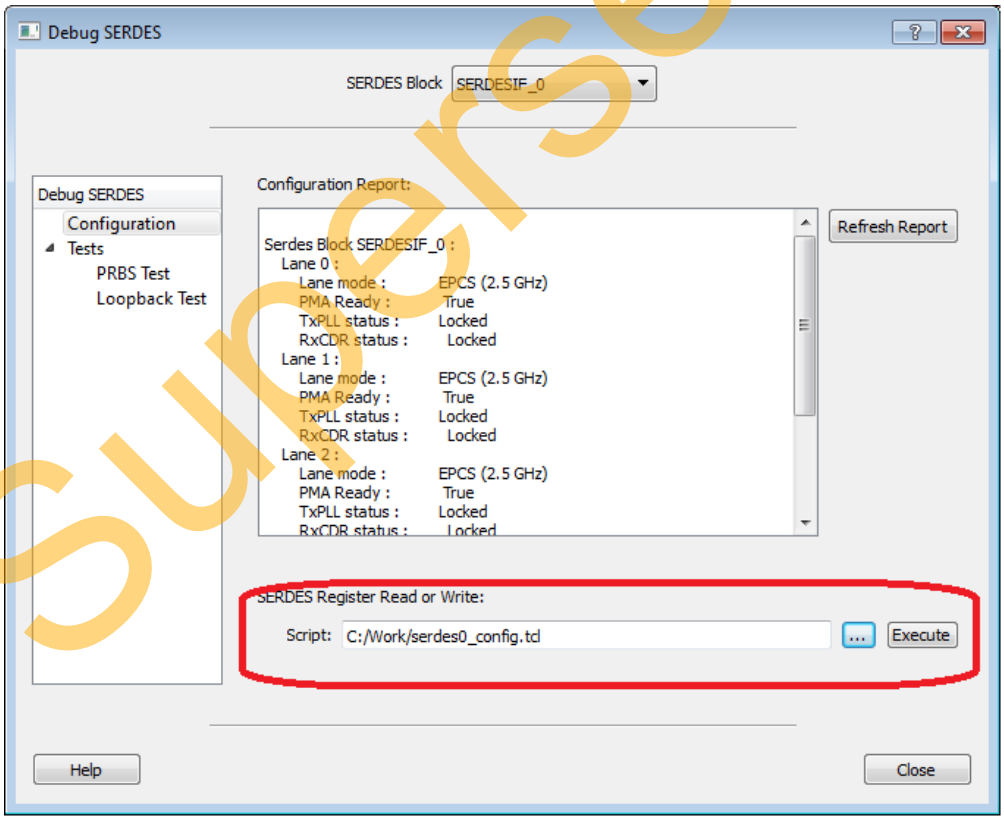

Figure 34 • **Tcl Script Execution User Interface**

Upon execution of the Tcl SERDES access, log of the access is displayed in the Libero SoC Console Log pane, as shown in Figure 35.

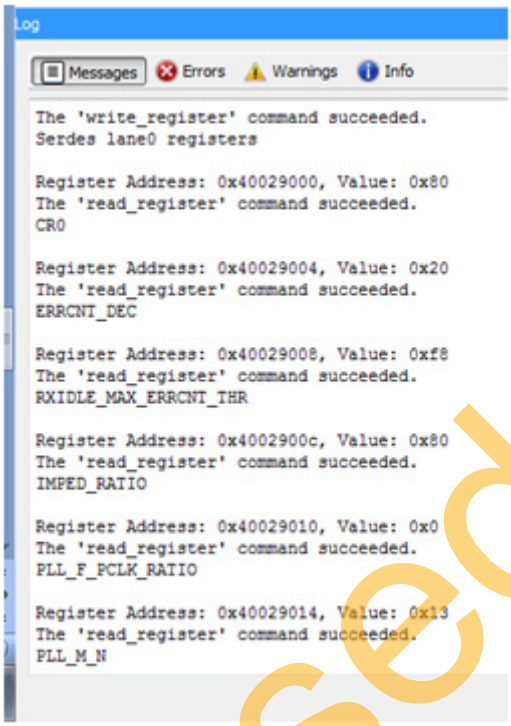

Figure 35 • **SERDES Access Log**

Refer to the Appendix for more Tcl examples.

Executing SERDES Debug from SmartDebug Tcl

PRBS:

```
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>
```

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

-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. Should be between 0 and 4 and varies between dies.

-**lane** <*num*>: SERDES lane number. Should be between 0 and 4.

-**near**: Corresponds to near-end (on-die) option for PRBS test. Not specifying implies off-die.

-**pattern** <*PatternType*>: The pattern sequence to use for PRBS test. It can be one of the following:

1. **prbs7** or **prbs11** or **prbs23** or **prbs31**
2. **custom**
3. **user**

-**value** <*PatternValue*>: Specifies the pattern type value for cases other than PRBS* sequences. It can be one of the following:

1. If **custom** is selected above, then it should be one of **all_zeros, all_ones, alternated, or dual_alternated**.
2. If **user** is selected above, then it should 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:

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

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

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

- **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. Should be between 0 and 4 and varies between dies.
- **lane** <*num*>: SERDES lane number. Should be between 0 and 4.
- **type** <*LoopbackType*>: Specifies the loopback test type. Should 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 from the GUI. Below 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_LOO
PBACK_top/export/SERDES_LOOPBACK_top.stp}
run_selected_actions
set_debug_device -name {M2S/M2GL050(T|S|TS)}
//Place serdes tcl commands after here
```

Conclusion

This tutorial demonstrated the capabilities of SmartDebug. SmartDebug provides the capabilities to observe and analyze many embedded device features. LiveProbe gives real-time access to device test points. While internal logic states can be easily accessed using ActiveProbes. 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.

A – Appendix

Tcl Script Examples

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"
```

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"
```

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"
```

B – List of Changes

Date	Version	Changes
October 2014	5	Updated the document for SERDES core change (SAR 61612).
September 2014	4	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).
April 2014	3	Added Note in "1. Specifying Live Probe Points" section (SAR 56593).
March 2014	2	Updated the software version from 11.2SP1 to 11.3 (SAR 56012).
		Updated design files using the latest 11.3 SERDES core (SAR 56012).
January 2014	1	Initial release.

C – 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.

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

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.

Technical Support

Visit the Customer Support website for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC home page.

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.

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:

Technical support email address: soc_tech@microsemi.com

My Cases

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

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email at: soc_tech@microsemi.com or contact a local Sales office listing at Sales.Support@Microsemi.com.

ITAR Technical Support

Contact technical support at: soc_tech_itar@microsemi.com for RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR). 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 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

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and 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; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

50200530-5/10.14