TU0827
Tutorial
PolarFire FPGA Debugging Using Splash Kit

Microsemi®
Power Matters.™

Power Matters.™

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

About Microsemi

50200827. 1.0 4/18

Contents

Figures

Tables

1 Revision History

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

1.1 Revision 1.0

The first publication of this document.

2 PolarFire FPGA Debugging Using Splash Kit

Design debug is a critical phase of the FPGA design flow. Microsemi's SmartDebug enables debugging of PolarFire devices in real time, and it does not require any other internal logic analyzer (ILA).

SmartDebug supports the following features:

- Device Status View
- FPGA Array Debug
- sNVM Debug
- µPROM Debug
- Transceiver Debug

These features enable designers to check the state of inputs and outputs in real time, without any re-layout of the design.

The built-in probe points of the PolarFire device and the probe capabilities of SmartDebug enable the real-time debug features.

SmartDebug provides the following capabilities:

- **Live probes:** Two dedicated probes can be configured to observe a probe point. The probe point may be any output of a register. After selecting the probe points, the probe data can be sent to two dedicated pins (PROBE_A and PROBE_B). You can connect an oscilloscope to the probe pins and monitor the signal status.
- **Active probes:** It allows dynamic asynchronous read and write to a flip-flop or probe point. This enables user 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.
- **Debug memory:** SmartDebug provides the Memory Blocks tab to dynamically and asynchronously read from and write to a selected FPGA fabric SRAM block.
- **sNVM debug capabilities:** It enables reading each page or multiple pages from sNVM.
- **Probe insertion:** It is a post-layout process that enables you to insert probes into the design and gets the signals out to the FPGA package pins to evaluate and debug the design.
- **TRANSCEIVER debug capabilities:** It makes debugging of high-speed serial designs simple. The SmartDebug JTAG interface extends access to configure, control, and observe XCVR operations and is accessible in every TRANSCEIVER design. The designs are implemented using the Libero System Builder to incorporate the TRANSCEIVER block enabling XCVR access from the SmartDebug. The Debug TRANSCEIVER window displays real-time system and the lane status information. XCVR configurations are supported with TCL scripting, allowing access to the entire XCVR register map for real-time customized tuning.

This tutorial provides a demo design to demonstrate SmartDebug's capabilities, which are used to perform real-time signal integrity testing and debugging.

2.1 Design Requirements

The following table lists the hardware, software, and IP requirements for this demo design.

Table 1 • **Design Requirements**

Requirement	Version
Operating system	64-bit Windows 7 or 10
Hardware	
PolarFire Splash Kit (MPF300TS-1FCG484EES) – PolarFire Splash Board – 12 V/5 A power adapter and cord – USB 2.0 A to mini-B cable for UART and programming	Rev 2 or later

Table 1 • **Design Requirements *(continued)***

Requirement	Version
Software	
Libero® SoC PolarFire	v2.1
IP	
PF_INIT_MONITOR	2.0.101
CORERESET_PF	2.1.100
PF_XCVR_REF_CLK	1.0.103
PF_TX_PLL	1.0.109
PF_XCVR_REF_CLK	1.0.103
PF_CCC	1.0.112
CORESMARTBERT	2.0.106
PF_UPROM	1.0.108
PF_URAM	1.1.107
PF_DPSRAM	1.1.110

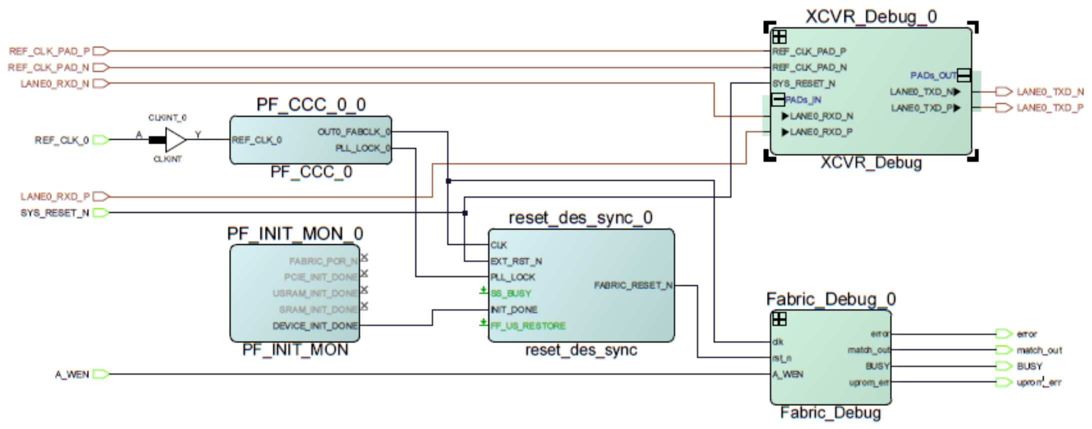
2.2 Prerequisites

Before you start:

1. Download the design files from the following link:
http://soc.microsemi.com/download/rsc/?f=mpf_tu0827_liberosocpolarfirev2p1_df
2. Download and install Libero SoC PolarFire v2.1 from the following location:
https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc-polarfire#downloads

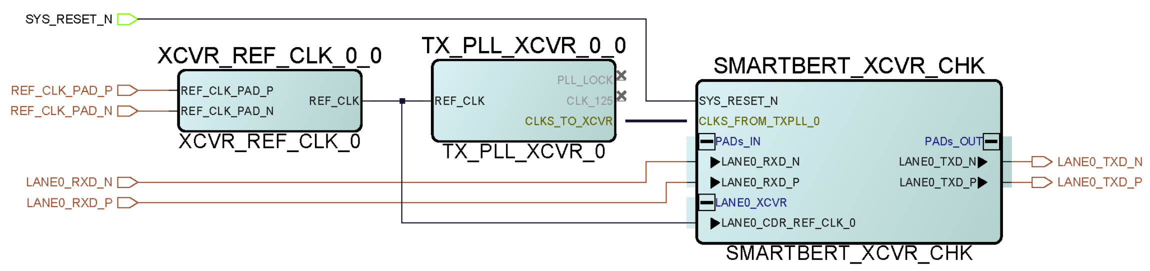
2.3 Demo Design

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



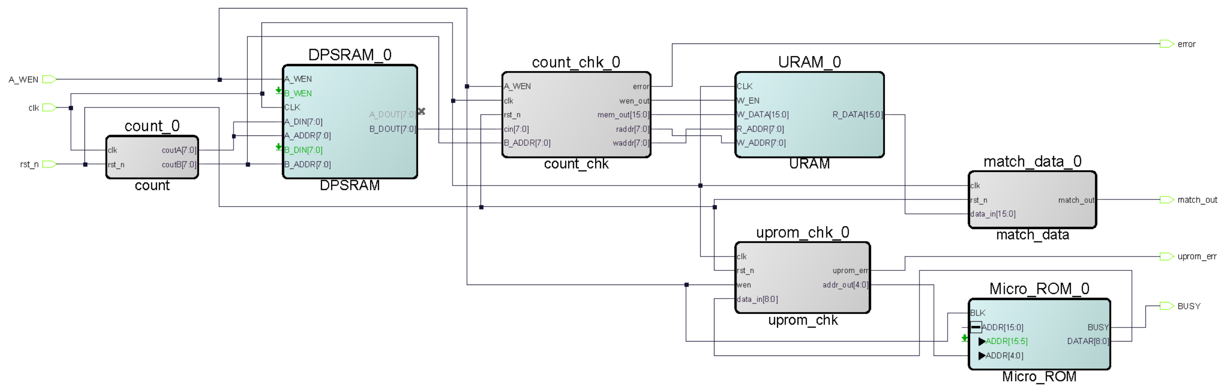
The design consists of five main blocks: the XCVR debug block (XCVR_Debug), the fabric debug block (Fabric_Debug), PF_INIT_MON block, PF_CCC, and Reset_des_sync_0 as shown in Figure 1, page 3.

Figure 2 • **XCVR_Debug Overall Design Blocks**



XCVR_Debug: The XCVR_Debug block demonstrates SmartDebug's real-time signal integrity (SI) testing and debugging capabilities to test and debug the PolarFire transceiver. The XCVR_Debug block consists of a CoreSmartBERT core along with TX_PLL and XCVR_REF_CLK macros. It implements the PolarFire transceiver in PMA mode.

Figure 3 • **Fabric_Debug Overall Design Blocks**



Fabric_Debug: The Fabric_Debug block demonstrates several FPGA fabric debug features of SmartDebug. First, it demonstrates SmartDebug's FPGA array debugging capabilities using a counter that loads a counting pattern into the LSRAM instance (DPSRAM). The data value of the LSRAM block is the same as the address value of the block. On the read side of the LSRAM, a count checker (count_chk) ensures that the count progresses as expected. If there is an error, the output (error) is latched high. Second, the Fabric_Debug block demonstrates the debug μPROM feature of SmartDebug using a μPROM instance. Third, the Fabric_Debug block demonstrates how to set live probes to monitor an internal user-selected point on the device in real time, and how to set active probes for dynamic asynchronous read and write to a flip-flop or probe point. These features help to quickly observe the output of the logic internally or quickly experiment to determine how the logic is affected by writes to a probe point. Lastly, the Fabric_Debug block demonstrates SmartDebug's capabilities to read and modify fabric SRAM content in real-time.

The PF_CCC block generates 125 MHz clock. Fabric_Debug logic works on this clock.

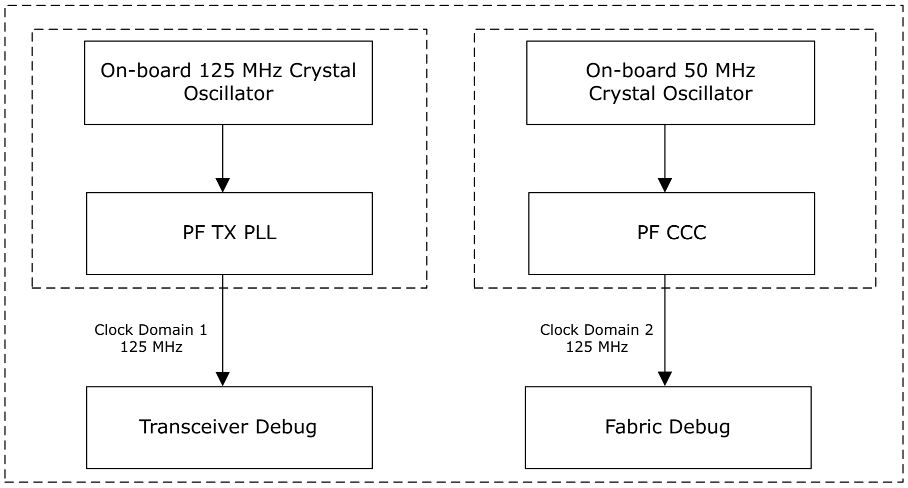
The PF_INIT_MON block checks the status of device initialization. When the initialization of SRAM and μPROM is completed, the IP asserts DEVICE_INIT_DONE signal. This signal is tied with an external reset and PLL lock.

The reset_des_sync_0 block is an instantiation of CoreRESET_PF IP. It synchronizes the de-assertion of asynchronous reset.

2.4 Clocking Structure

The reference design has two clock domains. As shown in the following illustration, clock domain 1, used for transceiver debug, runs at 125 MHz, and clock domain 2, used for fabric debug, runs at 125 MHz.

Figure 4 • **Clocking Structure**



2.5 Programming the Device

Before programming the device, SmartBERT probe related constraints need to be generated. The SmartDebug reads and writes to probe points associated with the SmartBERT IP for debugging. JTAG write to some probe points are not working as expected. This is a known issue. A software workaround is provided to determine the working probe points. The constraints need to be updated by following the steps mentioned in Appendix: Known Issues, page 20.

The following steps describe how to program the device on a PolarFire Splash Kit.

1. Ensure that the following jumper settings are followed.

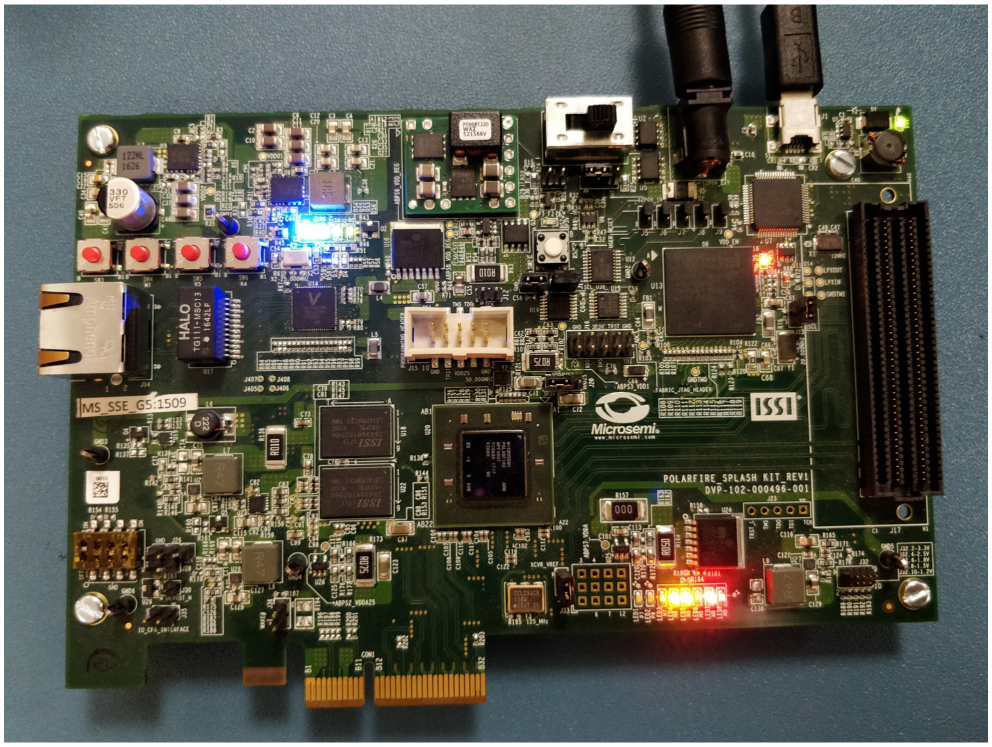
Note: Power-down the board before making the jumper connections.

Table 2 • **Jumper Settings**

Jumper	Description
J11	Close pin 1 and 2 for programming through FTDI chip
J5, J6, J7, J8, J9	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI
J10	Open pin 1 and 2 for programming through the FTDI SPI
J4	Short pin 1 and 2 for manual power switching using SW1
J3	Open pin 1 and 2 for 1.0 V

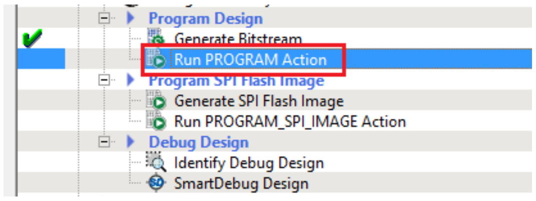
2. Connect the power supply cable to the **J2** connector on the board.
3. Connect the USB cable from the Host PC to the **J1** (FTDI port) on the board.
4. Power on the board using the **SW1** slide switch.

Figure 5 • **Board Setup**



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

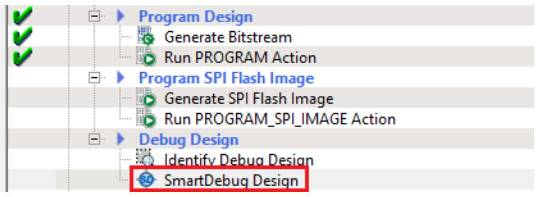
Figure 6 • **Programming the Device**



2.6 Launching SmartDebug from Libero

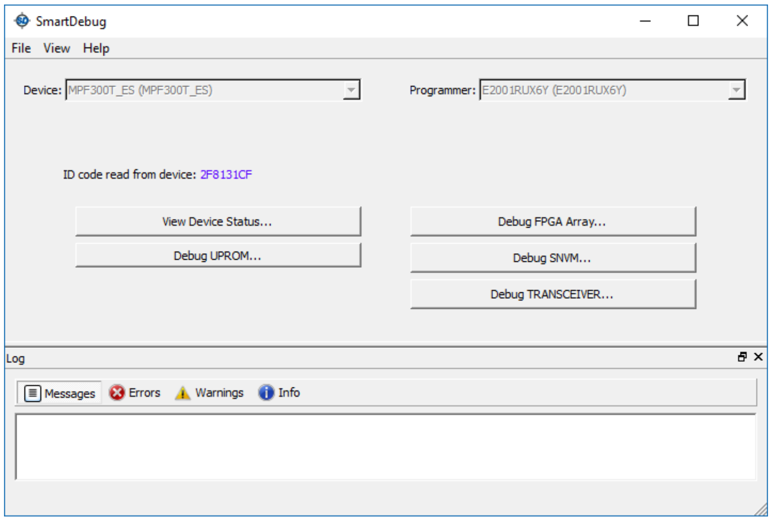
On the **Design Flow** window, double-click **SmartDebug Design**, as shown in the following figure.

Figure 7 • **Launching SmartDebug Design**



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

Figure 8 • **SmartDebug Window Debug Options**



2.7 Debugging the Design

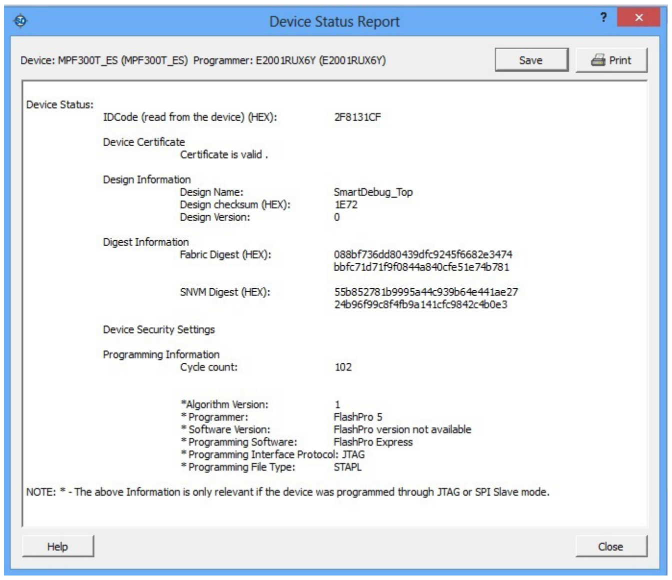
Debugging the device involves the following:

- View Device Status, page 7
- Debug FPGA Array, page 8
- Debug µPROM, page 12
- sNVM Debug, page 12
- Debug TRANSCEIVER, page 14

2.7.1 View Device Status

The View Device Status option provides the device status report. It summarizes the device information, programmer information, design information, factory serial number, and security information, if any are set. To view the device status report, click **View Device Status** in the **SmartDebug** window. The following figure shows a sample of the device status information.

Figure 9 • **Device Status Report Sample**



```
Device: MPF300T_ES (MPF300T_ES)  Programmer: E2001RUX6Y (E2001RUX6Y)

Device Status:
        IDCode (read from the device) (HEX):        2F8131CF

        Device Certificate
                Certificate is valid .

        Design Information
                Design Name:                SmartDebug_Top
                Design checksum (HEX):      1E72
                Design Version:             0

        Digest Information
                Fabric Digest (HEX):        088bf736dd80439dfc9245f6682e3474
                                            bbfc71d71f9f0844a840cfe51e74b781

                SNVM Digest (HEX):          55b852781b9995a44c939b64e441ae27
                                            24b96f99c8f4fb9a141cfc9842c4b0e3

        Device Security Settings

        Programming Information
                Cycle count:                102

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

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

2.7.2 Debug FPGA Array

The Debug FPGA Array provides an interface to probe the user logic implemented in the logic elements (LEs) of the FPGA using active and live probes, read-write access to the fabric flip-flops, and read-write access to the memories implemented using LSRAMs/URAMs. Probe insertion allows assignment of the internal signals to the assigned or unassigned pins. These signals can be monitored using the oscilloscope in real-time.The Debug FPGA Array supports the following four features:

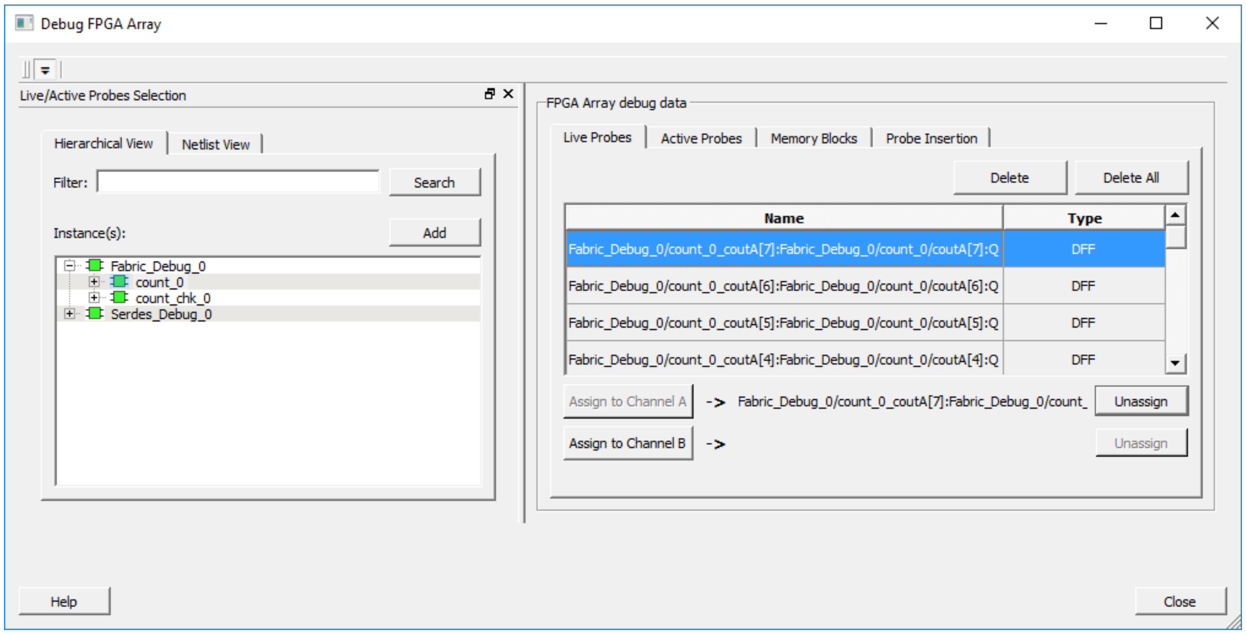
- Live Probes
- Active Probes
- Memory Blocks
- Probe Insertion

2.7.2.1 Live Probes

Live Probes enables the monitoring of two internal signals at a time in the design without having to repeat place and route. PolarFire devices have two dedicated live probe channels (for example, pin H6 and G6 of PolarFire MPF300TS device).

To use Live Probes, reserve pins using **Reserve Pins for Probes** under **Constraints Manager** in Libero SoC PolarFire. If you do not reserve pins for live probes, the live probe I/O's function as GPIOs and are used for routing nets in the design. The following figure shows the **Live Probes** tab.

Figure 10 • **Debug FPGA Array—Live Probes**



Name	Type
Fabric_Debug_0/count_0_coutA[7]:Fabric_Debug_0/count_0/coutA[7]:Q	DFF
Fabric_Debug_0/count_0_coutA[6]:Fabric_Debug_0/count_0/coutA[6]:Q	DFF
Fabric_Debug_0/count_0_coutA[5]:Fabric_Debug_0/count_0/coutA[5]:Q	DFF
Fabric_Debug_0/count_0_coutA[4]:Fabric_Debug_0/count_0/coutA[4]:Q	DFF

Assign to Channel A -> Fabric_Debug_0/count_0_coutA[7]:Fabric_Debug_0/count_

Assign to Channel B ->

2.7.2.2 Active Probes

Active Probes enables to read or change the values of probe points in a design through JTAG. Active Probes dynamically and asynchronously read or write to any logic element register bit. The probe points of a design are selected using active probes. Active probes are useful for a quick observation of an internal signal. All of the probe points for the design are displayed in **Hierarchical View** and **Netlist View** in the left pane of the **Active Probes** tab.

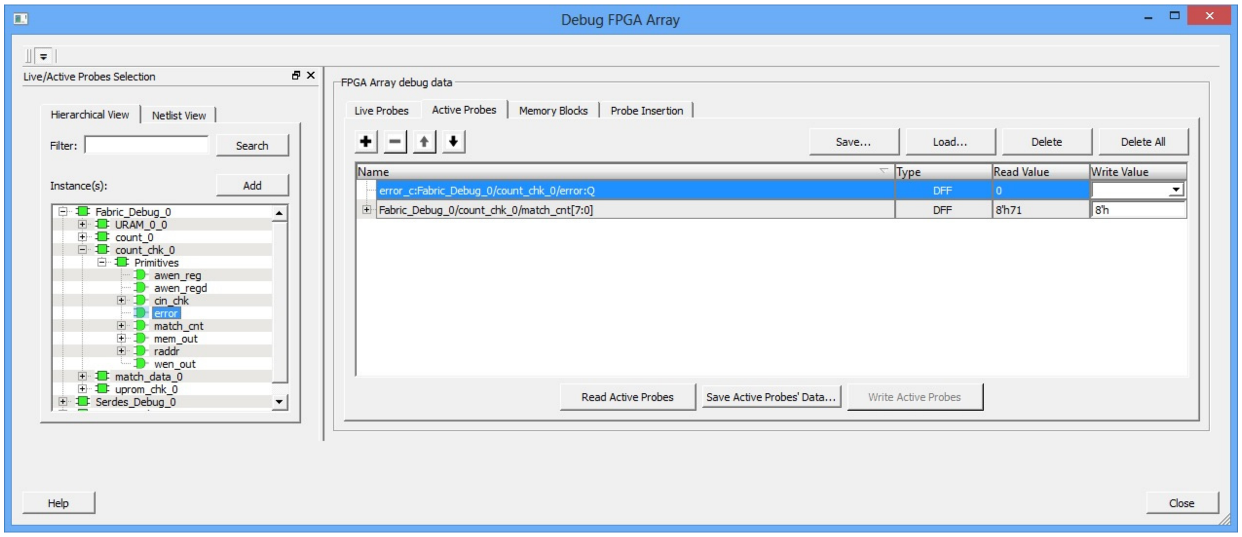
- **Hierarchical View**: Available probe points are listed in hierarchical order.
- **Netlist View**: Available probe points are listed with the Name and Type, which are physical locations of flip-flops.

To add probe points to a list:

1. Select the **Active Probes** tab in the right pane. The probe signals are displayed in the left pane.
2. Select the probe points that you want to add from the Hierarchical View or Netlist View in the left pane.

3. Right-click the selected points and click **Add** to add them to the **Active Probes**. You can also add the selected probe points by clicking Add in the top-right corner of the left pane. The probes signals can be filtered with the **Filter** option.
4. Click **Read Active Probes** to read the content of the registers added to the window.

Figure 11 • **Debug FPGA Array—Active Probes**

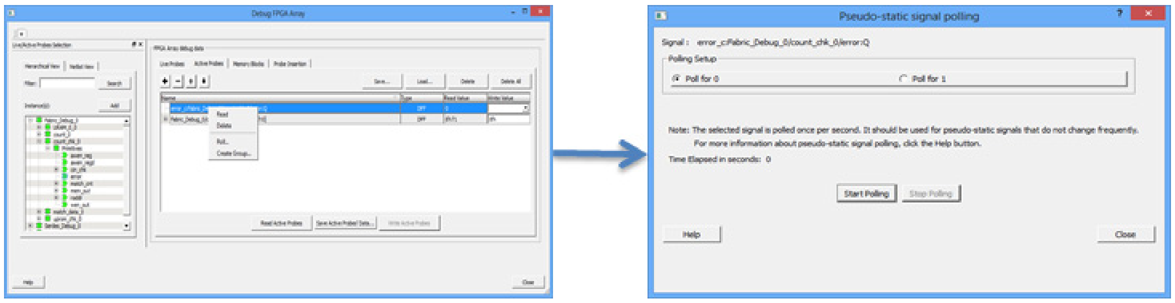


Name	Type	Read Value	Write Value
error_c:Fabric_Debug_0/count_chk_0/error:Q	DFF	0	
Fabric_Debug_0/count_chk_0/match_cnt[7:0]	DFF	8'h71	8'h

5. To use pseudo static signal polling, on the **Active Probes** tab, right-click any probe point and select **Poll**, as shown in the following figure.

Static signal polling is used to check whether the logical bit value is changed to expected polled value.

Figure 12 • **Pseudo-static Signal Polling**



Signal : error_c:Fabric_Debug_0/count_chk_0/error:Q

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

Time Elapsed in seconds: 0

2.7.2.3 Memory Blocks

SmartDebug provides the Memory Blocks tab to dynamically and asynchronously read from and write to a selected FPGA fabric SRAM block. Memory blocks are categorized into two views:

- Physical View—shows the actual memory view of the RAM in FPGA
- Logical View—shows a logical representation of RAM block

Using the **Memory Blocks** tab, you can select the required memory block to:

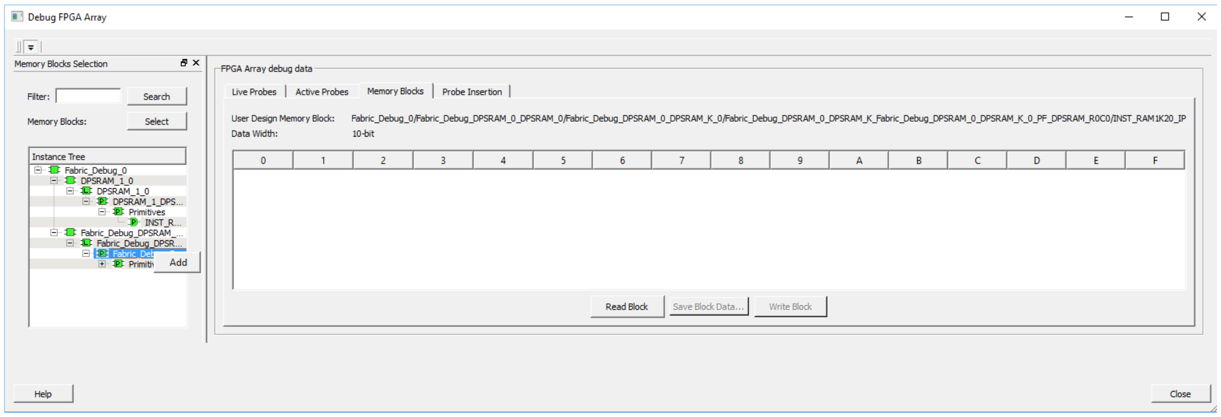
- Read
- Capture a snapshot of the memory
- Modify memory values, and then write the values back to that block

To read and write memory blocks:

1. Select the **Memory Blocks** tab in the right pane of the SmartDebug window.
2. View the memory blocks in the left pane in the **Hierarchical View**.
3. Select the memory block in the left pane and click **select** in the top-right corner of the pane.
4. Right-click the selected memory block and click **Add**.

The following figure shows the **Memory Blocks** tab in **Debug FPGA Array** window.

Figure 13 • **Debug FPGA Array—Memory Blocks**

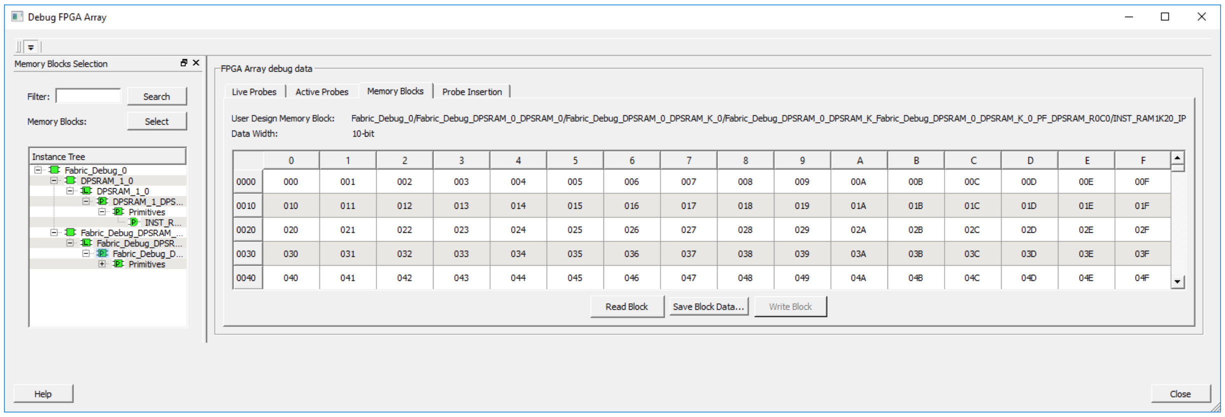


User Design Memory Block: Fabric_Debug_0/Fabric_Debug_DPSRAM_0_DPSRAM_0/Fabric_Debug_DPSRAM_0_DPSRAM_K_0/Fabric_Debug_DPSRAM_0_DPSRAM_K_Fabric_Debug_DPSRAM_0_DPSRAM_K_0_PF_DPSRAM_R0C0/INST_RAM1K20_IP
Data Width: 10-bit

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

5. Click **Read Block**. The specified memory block is read as shown in the following figure.

Figure 14 • **Memory Blocks—Read Block**



User Design Memory Block: Fabric_Debug_0/Fabric_Debug_DPSRAM_0_DPSRAM_0/Fabric_Debug_DPSRAM_0_DPSRAM_K_0/Fabric_Debug_DPSRAM_0_DPSRAM_K_Fabric_Debug_DPSRAM_0_DPSRAM_K_0_PF_DPSRAM_R0C0/INST_RAM1K20_IP
Data Width: 10-bit

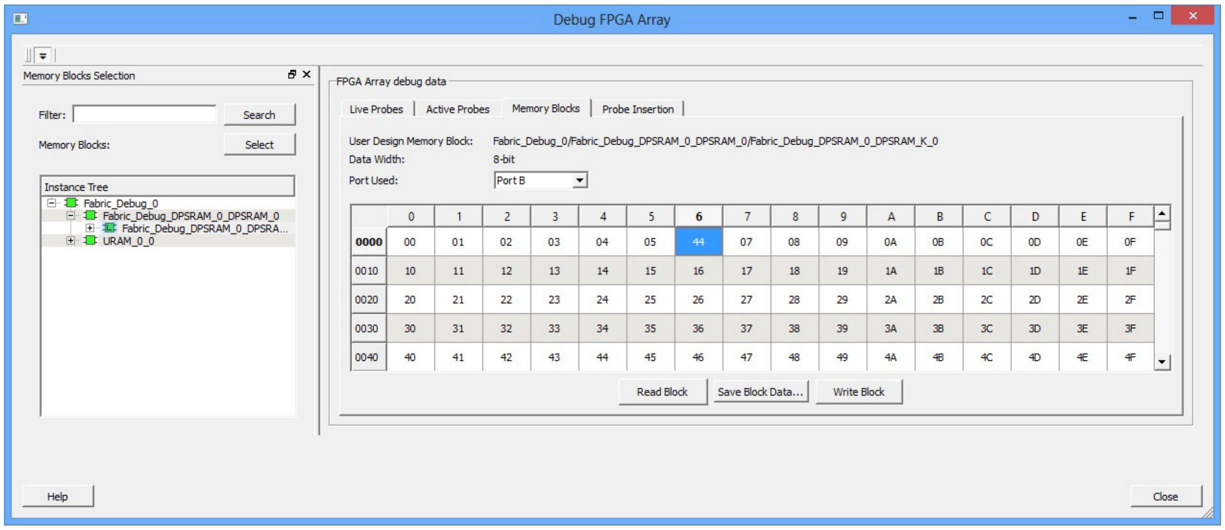
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0010	010	011	012	013	014	015	016	017	018	019	01A	01B	01C	01D	01E	01F
0020	020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F
0030	030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E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

6. Enter a hexadecimal value in the memory block locations and click **Write Block** to write content into memory.

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 signal value to low through DIP1 (first switch of SW8). This drives a SELECT of a MUX that selects between high and low inputs. When DIP1 is asserted, A_WEN becomes low, which prevents any write from the counter to the SRAM block.

7. Switch On DIP1, enter a hexadecimal value in the memory block location(s) and click **Write Block** to write the modified value to the SRAM, as shown in the following figure.

Figure 15 • **Memory Blocks—Write Block**



Debug FPGA Array

Memory Blocks Selection

Filter:

Memory Blocks:

Instance Tree
- Fabric_Debug_0
  - Fabric_Debug_DPSRAM_0_DPSRAM_0
    - Fabric_Debug_DPSRAM_0_DPSRA...
  - URAM_0_0

FPGA Array debug data

User Design Memory Block: Fabric_Debug_0/Fabric_Debug_DPSRAM_0_DPSRAM_0/Fabric_Debug_DPSRAM_0_DPSRAM_K_0
Data Width: 8-bit
Port Used: Port B

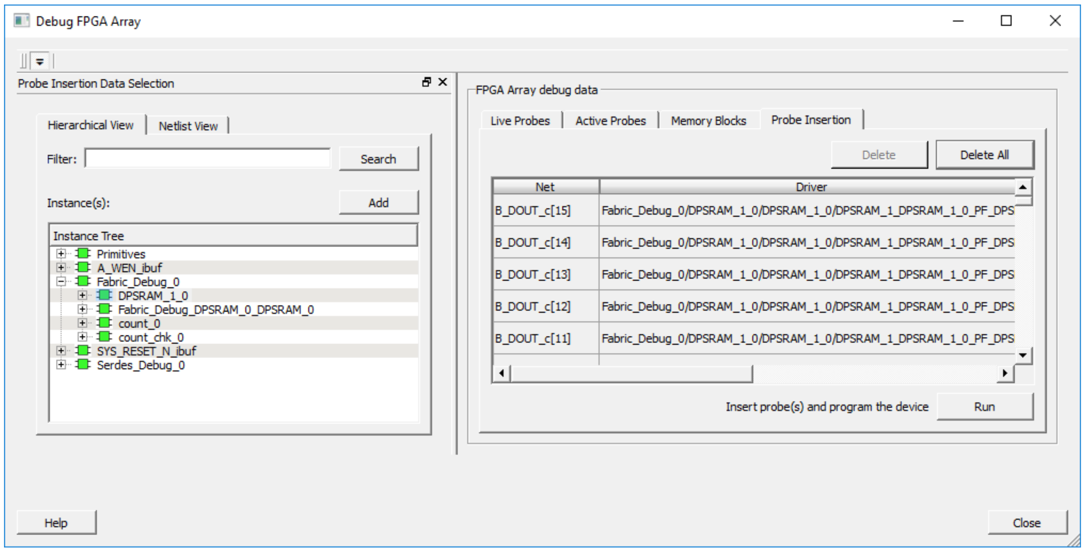
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	00	01	02	03	04	05	44	07	08	09	0A	0B	0C	0D	0E	0F
0010	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
0020	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
0030	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
0040	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

8. The error LED(P8) light turns on, indicating an error in the counting pattern.
9. Go to **Active Probes** tab, read the value of **error** signal, it should show '1'. To use static signal polling, right-click **error_c:Fabric_Debug_0/count_chk_0/error:Q** and select **Poll** (**Poll for 0)**, as shown in Figure 12, page 9.
10. Move DIP1 to off state to resume the write operation from the counter to the SRAM. This overwrites the error that was injected into the SRAM. Check the status of LED, it must turn off. Hit the **Poll for 0**, **User value match** message should appear on the polling window. Close the **Pseudo-static signal polling** window.
11. The content of the SRAM can be rechecked by clicking **Read Block** in the **Memory Blocks** tab.

2.7.2.4 Probe Insertion

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

Figure 16 • **Debug FPGA Array—Probe Insertion**



Debug FPGA Array

Probe Insertion Data Selection

Filter:

Instance(s):

Instance Tree
- Primitives
- A_WEN_ibuf
- Fabric_Debug_0
  - DPSRAM_1_0
  - Fabric_Debug_DPSRAM_0_DPSRAM_0
  - count_0
  - count_chk_0
- SYS_RESET_N_ibuf
- Serdes_Debug_0

FPGA Array debug data

Net	Driver
B_DOUT_c[15]	Fabric_Debug_0/DPSRAM_1_0/DPSRAM_1_0/DPSRAM_1_DPSRAM_1_0_PF_DPS
B_DOUT_c[14]	Fabric_Debug_0/DPSRAM_1_0/DPSRAM_1_0/DPSRAM_1_DPSRAM_1_0_PF_DPS
B_DOUT_c[13]	Fabric_Debug_0/DPSRAM_1_0/DPSRAM_1_0/DPSRAM_1_DPSRAM_1_0_PF_DPS
B_DOUT_c[12]	Fabric_Debug_0/DPSRAM_1_0/DPSRAM_1_0/DPSRAM_1_DPSRAM_1_0_PF_DPS
B_DOUT_c[11]	Fabric_Debug_0/DPSRAM_1_0/DPSRAM_1_0/DPSRAM_1_DPSRAM_1_0_PF_DPS

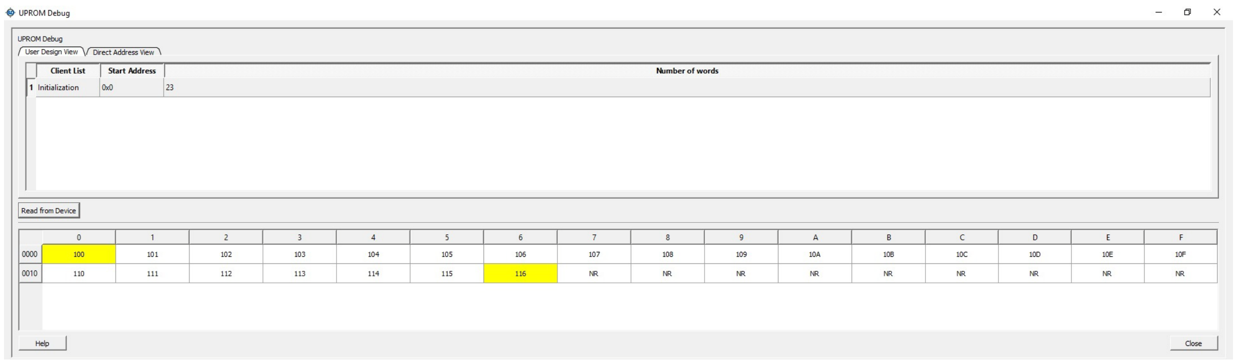
Insert probe(s) and program the device

2.7.3 Debug µPROM

SmartDebug enables debugging µPROM and reading its µPROM contents. The clients added in the design can be debugged using the SmartDebug Debug µPROM feature.

1. Click **Debug µPROM** in the **SmartDebug** window. The **µPROM Debug** window is displayed as shown in the following figure.
2. Select **Initialization** in the **User Design View** tab and then click **Read from Device** to read the µPROM content. Check whether the content provided in uprom.mem file (part of design stimulus files) matches with the data read from µPROM.

Figure 17 • **µPROM Debug**



	Client List	Start Address	Number of words
1	Initialization	0x0	23

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
0010	110	111	112	113	114	115	116	NR	NR	NR	NR	NR	NR	NR	NR	NR

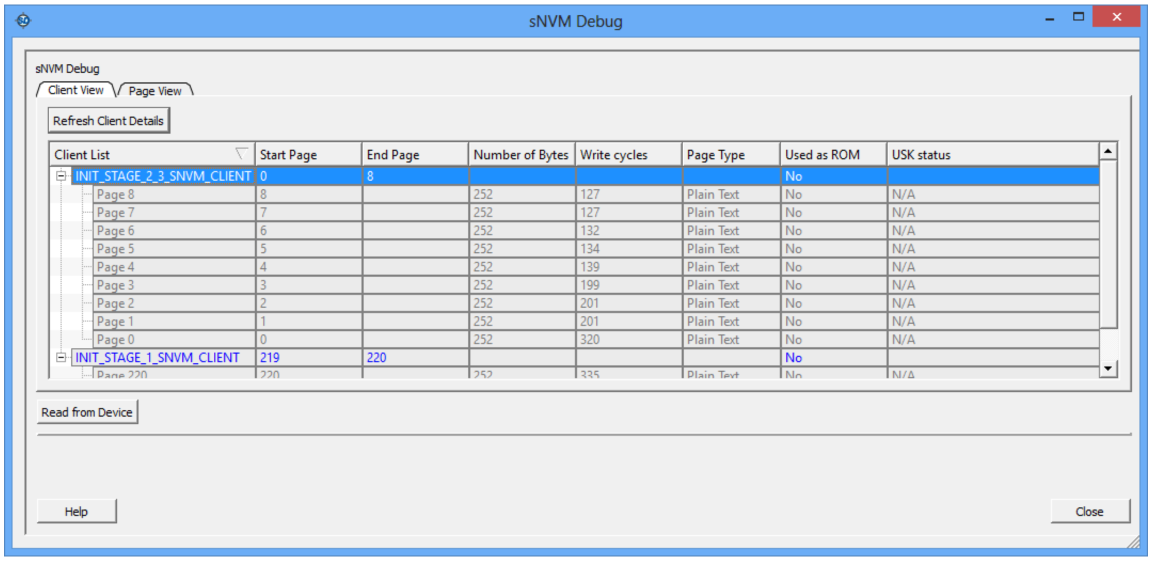
Note: PolarFire devices have a single user programmable read only memory (µPROM) row located at the bottom of the fabric, providing up to 459 Kb of non-volatile, read-only memory. The address bus is 16 bits wide, and the read data bus is 9-bit wide. µPROM is used to store the configuration data, which is used by Fabric logic to process.

2.7.4 sNVM Debug

sNVM Debug feature enables reading from the sNVM during debug. Debug Pass Key is required to carry out SNVM_DEBUG instruction. This feature supports debugging of non-authenticated plain text, authenticated plain text, and clients cipher authenticated.

1. Click **Debug SNVM** in the **SmartDebug** window.
2. Click the **Client View** tab. The client view details are listed—Client Names, Start Page, Number of Bytes, Write Cycles, Page Type, Used as ROM, and USK Status.
3. Select a client from the list in the **Client View** and click **Read from Device** as shown in the following figure.

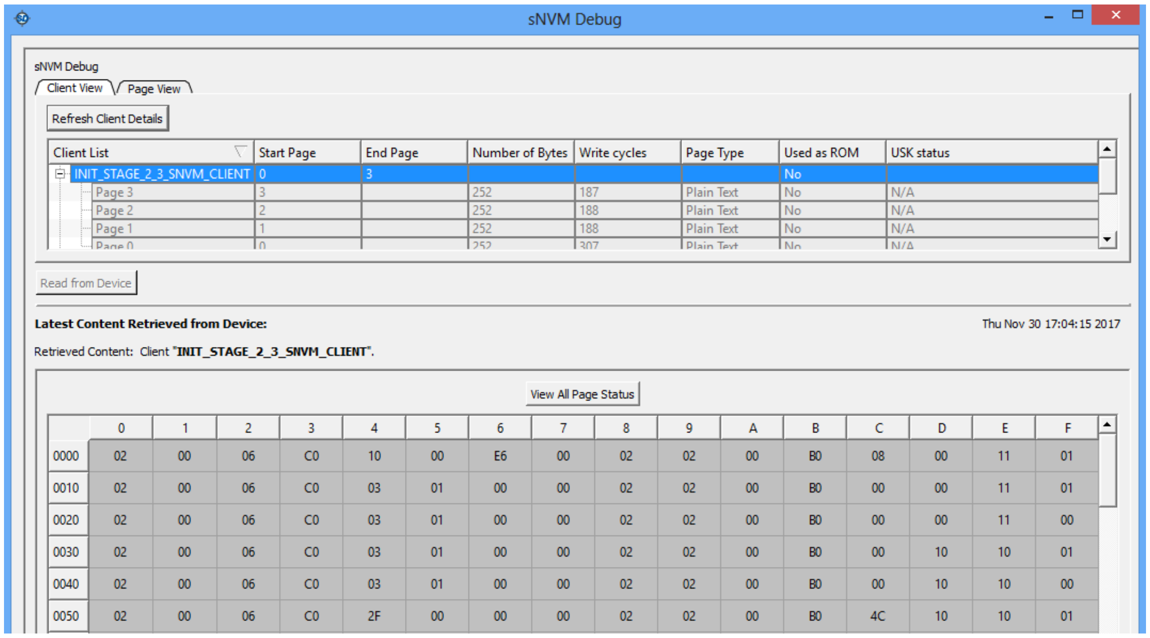
Figure 18 • **sNVM Debug**



Client List	Start Page	End Page	Number of Bytes	Write cycles	Page Type	Used as ROM	USK status
INIT_STAGE_2_3_SNVM_CLIENT	0	8				No	
Page 8	8		252	127	Plain Text	No	N/A
Page 7	7		252	127	Plain Text	No	N/A
Page 6	6		252	132	Plain Text	No	N/A
Page 5	5		252	134	Plain Text	No	N/A
Page 4	4		252	139	Plain Text	No	N/A
Page 3	3		252	199	Plain Text	No	N/A
Page 2	2		252	201	Plain Text	No	N/A
Page 1	1		252	201	Plain Text	No	N/A
Page 0	0		252	320	Plain Text	No	N/A
INIT_STAGE_1_SNVM_CLIENT	219	220				No	
Page 220	220		252	335	Plain Text	No	N/A

The following figure shows the **Client View** window.

Figure 19 • **sNVM Debug—Client View**



sNVM Debug

Client View | Page View

Refresh Client Details

Client List	Start Page	End Page	Number of Bytes	Write cycles	Page Type	Used as ROM	USK status
INIT_STAGE_2_3_SNVM_CLIENT	0	3				No	
Page 3	3		252	187	Plain Text	No	N/A
Page 2	2		252	188	Plain Text	No	N/A
Page 1	1		252	188	Plain Text	No	N/A
Page 0	0		252	307	Plain Text	No	N/A

Read from Device

Latest Content Retrieved from Device: Thu Nov 30 17:04:15 2017

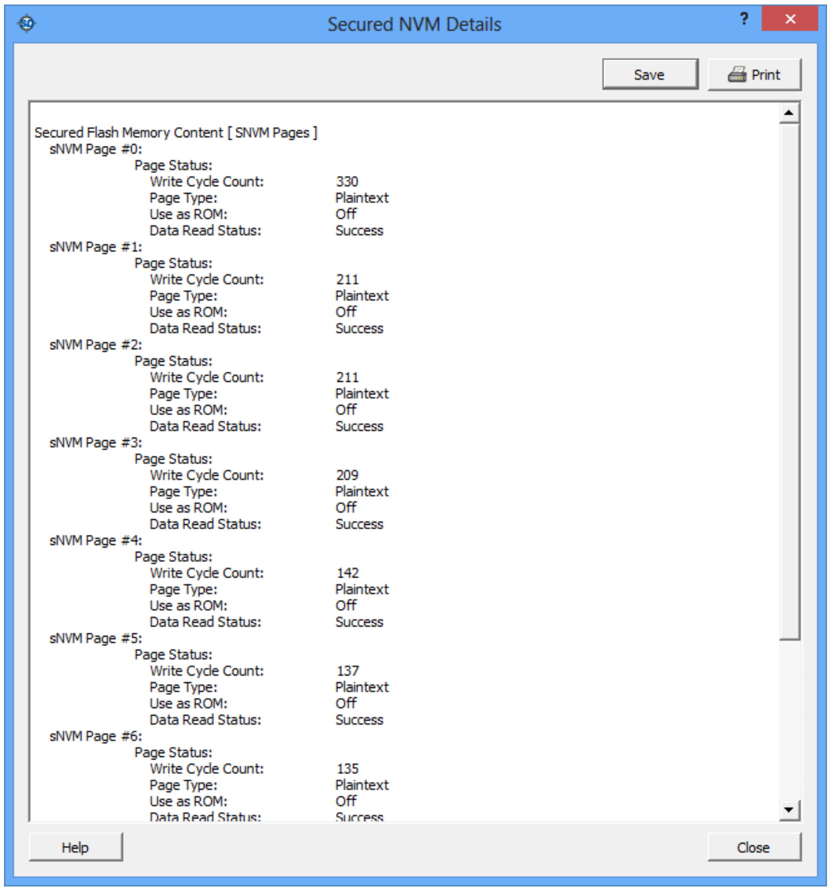
Retrieved Content: Client "**INIT_STAGE_2_3_SNVM_CLIENT**".

View All Page Status

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	02	00	06	C0	10	00	E6	00	02	02	00	B0	08	00	11	01
0010	02	00	06	C0	03	01	00	00	02	02	00	B0	00	00	11	01
0020	02	00	06	C0	03	01	00	00	02	02	00	B0	00	00	11	00
0030	02	00	06	C0	03	01	00	00	02	02	00	B0	00	10	10	01
0040	02	00	06	C0	03	01	00	00	02	02	00	B0	00	10	10	00
0050	02	00	06	C0	2F	00	00	00	02	02	00	B0	4C	10	10	01

4. Click **View All Page Status** to view the page status such as Write Cycle Count, Page Type, Use as ROM, and Data Read Status as shown in the following figure.

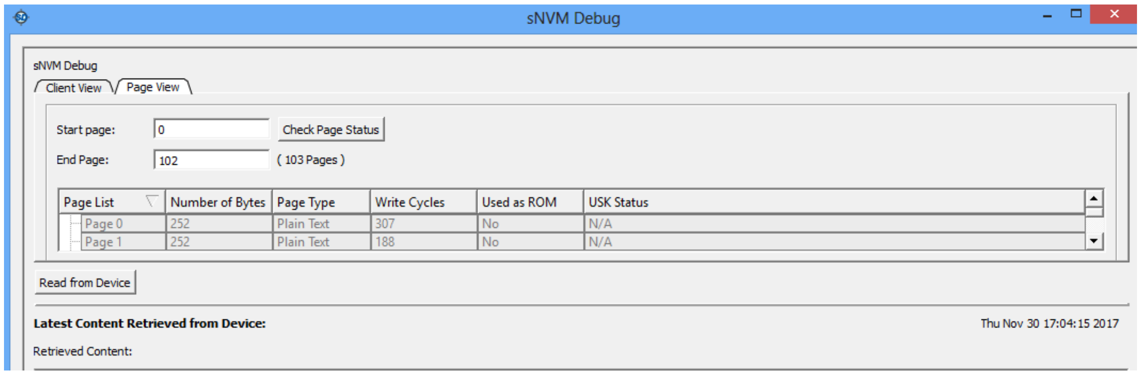
Figure 20 • **Secured NVM Details**



```
Secured Flash Memory Content [ SNVM Pages ]
   sNVM Page #0:
            Page Status:
               Write Cycle Count:       330
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #1:
            Page Status:
               Write Cycle Count:       211
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #2:
            Page Status:
               Write Cycle Count:       211
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #3:
            Page Status:
               Write Cycle Count:       209
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #4:
            Page Status:
               Write Cycle Count:       142
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #5:
            Page Status:
               Write Cycle Count:       137
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
   sNVM Page #6:
            Page Status:
               Write Cycle Count:       135
               Page Type:               Plaintext
               Use as ROM:              Off
               Data Read Status:        Success
```

5. Click the **Page View** tab in the **sNVM Debug** window, Page view displays the client details of the required pages. You can read pages from 0-220 in the page view.
6. Enter the page number that you want to read in the **Start Page** and **Number of Bytes** in the respective boxes.
7. Click **Check Page Status**. The page status information is displayed as shown in the following figure.

Figure 21 • **sNVM Debug—Page View**



sNVM Debug

Client View | Page View

Start page: 0 — Check Page Status

End Page: 102 (103 Pages)

Page List	Number of Bytes	Page Type	Write Cycles	Used as ROM	USK Status
Page 0	252	Plain Text	307	No	N/A
Page 1	252	Plain Text	188	No	N/A

Read from Device

Latest Content Retrieved from Device: Thu Nov 30 17:04:15 2017

Retrieved Content:

2.7.5 Debug TRANSCEIVER

SmartDebug enables transceiver debugging, which includes checking lane functionality and health for different settings of lane parameters. To access the debug transceiver feature, select **Debug TRANSCEIVER** in the **SmartDebug** window. Debug Transceiver supports the following features:

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

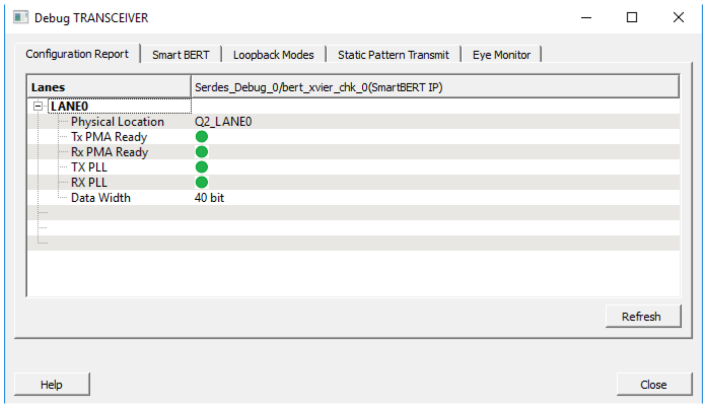
2.7.5.1 Configuration Report

The Configuration Report feature creates a report that shows the physical location, Tx and Rx PLL lock status, and data width of all enabled transceiver lanes. This report includes the following lane parameters:

- **Physical Location**: Physical location of the transceiver lanes in the system.
- **Tx PMA Ready**: Tx lane of the transceiver is powered up and ready for transactions.
- **Rx PMA Ready**: Rx lane is powered up and ready for transactions.
- **TX PLL**: TX PLL of the transceiver is locked.
- **RX PLL**: RX PLL of the transceiver is locked.
- **Data Width**: Configured data width of the corresponding lanes in the transceiver.

The following figure shows **Configuration Report** tab.

Figure 22 • **Configuration Report**



Debug TRANSCEIVER

Configuration Report | Smart BERT | Loopback Modes | Static Pattern Transmit | Eye Monitor

Lanes	Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)
LANE0	
Physical Location	Q2_LANE0
Tx PMA Ready	●
Rx PMA Ready	●
TX PLL	●
RX PLL	●
Data Width	40 bit

Refresh

Help | Close

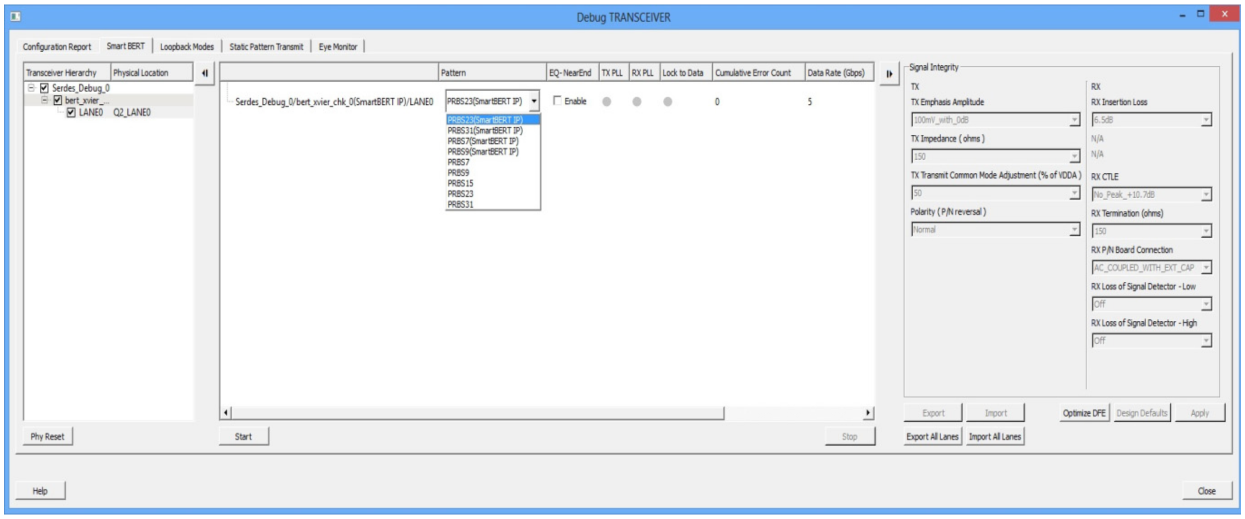
2.7.5.2 SmartBERT

SmartBERT enables you to run diagnostic tests on the transceiver lanes. SmartBERT uses the PRBS generator and checker functionality available in each transceiver lane to determine the bit error rate (BER) of a lane. The various PRBS patterns supported are PRBS7, PRBS9, PRBS15, PRBS23, and PRBS31. Near-end loopback can be performed using one of these PRBS patterns.

To run SmartBERT in Debug TRANSCEIVER, follow these steps:

1. Select the **SmartBERT** tab in the **Debug TRANSCEIVER** window.
2. Select the **Pattern** from the drop-down list.
3. Select the **EQ-NearEnd** check box to enable internal loop back, (this step can be ignored if external loop back is enabled).
4. Click **Start**. It enables both transmitter and the receiver for a particular lane and for a particular PRBS pattern. The following figure shows the status of the TXPLL, RXPLL, Lock to Data, Data rate, and the BER.

Figure 23 • **Debug TRANSCEIVER—Smart BERT**



Transceiver Hierarchy	Physical Location
Serdes_Debug_0	
bert_xvier_...	
LANE0	Q2_LANE0

	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)
Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	Enable	●	●	●	0	5

PRBS23(SmartBERT IP)
PRBS31(SmartBERT IP)
PRBS7(SmartBERT IP)
PRBS9(SmartBERT IP)
PRBS7
PRBS9
PRBS15
PRBS23
PRBS31

Signal Integrity

TX
- TX Emphasis Amplitude: 100mV_with_0dB
- TX Impedance (ohms): 150
- TX Transmit Common Mode Adjustment (% of VDDA): 50
- Polarity (P/N reversal): Normal

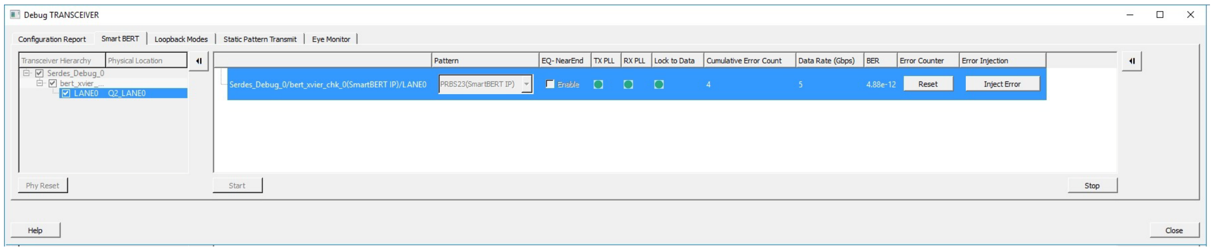
RX
- RX Insertion Loss: 6.5dB
- N/A
- N/A
- RX CTLE: No_Peak_+10.7dB
- RX Termination (ohms): 150
- RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
- RX Loss of Signal Detector - Low: Off
- RX Loss of Signal Detector - High: Off

When a SmartBERT IP lane is added, the **Error Injection** column is displayed in the in the right pane. The error injection feature is provided to inject an error while running a PRBS pattern. This feature is unavailable if regular lanes are added. Also, this feature is disabled for a SmartBERT IP lane that has a non-configured PRBS pattern selected.

5. Click **Reset** to clear the error count under **Error Counter**. Error Count is displayed when the lane is added.

The following figure shows the **Smart BERT** tab in the **Debug TRANSCEIVER** window.

Figure 24 • **Smart BERT—Error Counter**



Transceiver Hierarchy	Physical Location
Serdes_Debug_0	
bert_xvier_...	
LANE0	Q2_LANE0

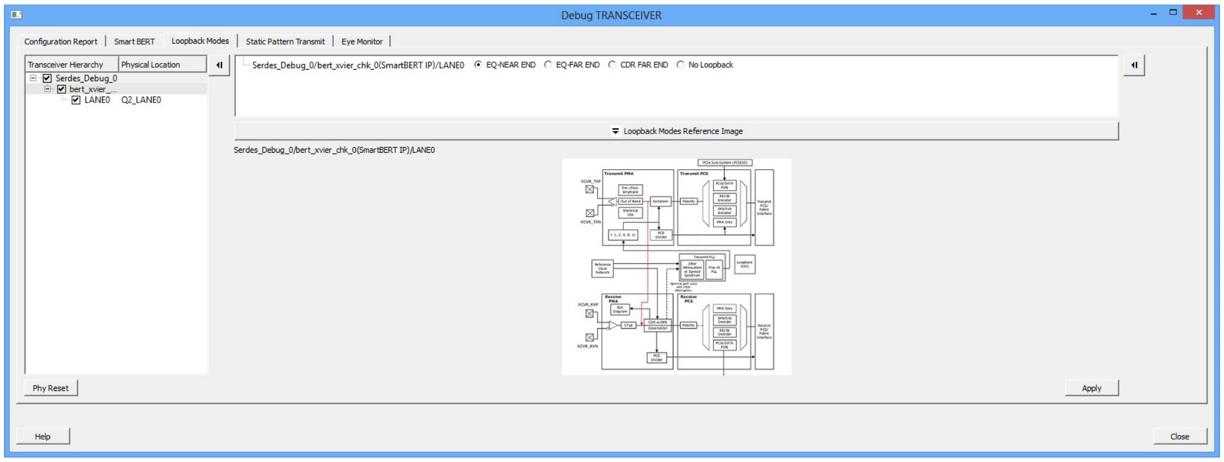
	Pattern	EQ-NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)	BER	Error Counter	Error Injection
Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	Enable	🟩	🟩	🟩	4	5	4.88e-12	Reset	Inject Error

2.7.5.3 Loopback Modes

Loopback modes perform the following types of loopback tests:

- EQ-Near End Loopback: Serialized data from PMA is looped from Tx to Rx internally before the transmit buffer. This is called near-end serial loopback. EQ-Near End loopback supports data transmission rates of up to 10.315 Gbps.
- EQ-Far End Loopback: Serialized data from Rx is looped back to Tx in PMA. This is called far-end serial loopback. EQ-Far End loopback supports data transmission rates of up to 1.25 Gbps.
- CDR-Far End Loopback: De-serialized data from PCS Rx channel is looped back to Tx.
- No Loopback: Data is not looped internally.

Figure 25 • **Debug TRANSCEIVER—Loopback Modes**


Debug TRANSCEIVER

Configuration Report | Smart BERT | Loopback Modes | Static Pattern Transmit | Eye Monitor

Transceiver Hierarchy	Physical Location
☑ Serdes_Debug_0	
☑ bert_xvier_...	
☑ LANE0	Q2_LANE0

Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0 [✓ EQ-NEAR END] EQ-FAR END CDR FAR END No Loopback

Loopback Modes Reference Image

Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0

2.7.5.4 Static Pattern Transmit

Static Pattern Transmit enables the selection of pattern to be transmitted on a specific transceiver (Tx) lane. The following patterns are supported:

- Fixed pattern
- Max run length pattern
- User pattern

The user pattern is defined in the value column. It must be hex numbers and not greater than the configured data width.

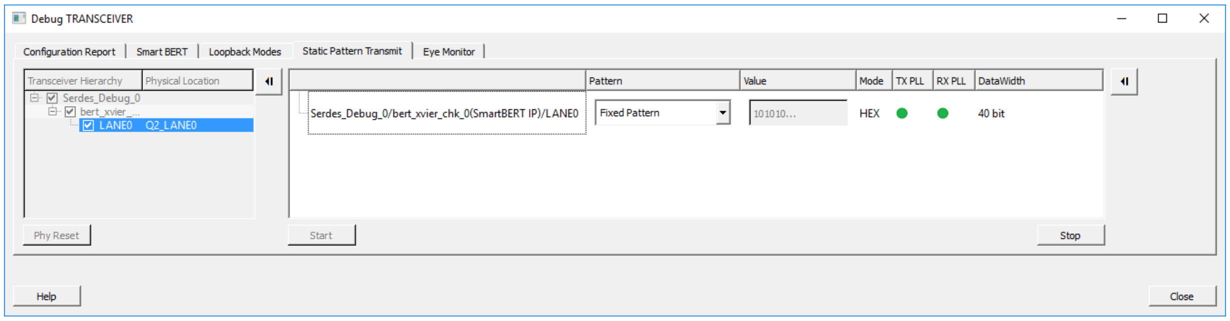
TX-PLL indicates lane lock onto TX PLL when a static pattern is transmitted. RX-PLL indicates RX PLL lock when a static pattern is transmitted. Data Width displays the data width configured for a transceiver lane.

To view static pattern transmit:

1. Select the **Static Pattern Transmit** tab.
2. Select the **Transceiver Hierarchy** in the left pane of the window. The selected lane data is displayed in the right pane. Select a pattern from the **Pattern** drop-down list.
3. Click **Start**. The static pattern for the selected lanes is transmitted.
4. Click **Stop**. The static pattern transmission is stopped for the selected lanes.

The following figure shows the **Static Pattern Transmit** tab.

Figure 26 • **Static Pattern Transmit**



Transceiver Hierarchy	Physical Location
Serdes_Debug_0	
bert_xvier_...	
LANE0	Q2_LANE0

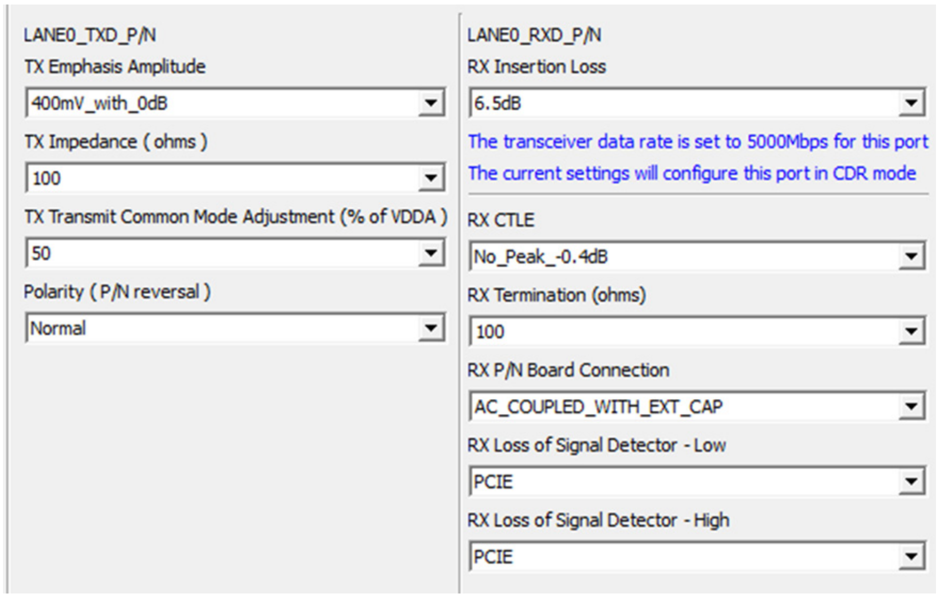
	Pattern	Value	Mode	TX PLL	RX PLL	DataWidth
Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0	Fixed Pattern	101010...	HEX	●	●	40 bit

2.7.5.5 Eye Monitor

Eye Monitor enables visualizing the eye diagram present within the receiver. This feature plots the receive eye after the CTLE and DFE functions. The diagram representation provides vertical and horizontal measurements of the eye and BER performance measurements. Whenever PRBS/static pattern transmission is in progress, click the **Eye Monitor** tab in the **Debug TRANSCEIVER** window to see the eye monitor representation within the receiver.

The following figure shows the recommended SI settings for the demo design. These settings are for short reach and less lossy cables.

Figure 27 • **Recommended Settings for Eye Monitor**



LANE0_TXD_P/N

TX Emphasis Amplitude: 400mV_with_0dB

TX Impedance (ohms): 100

TX Transmit Common Mode Adjustment (% of VDDA): 50

Polarity (P/N reversal): Normal

LANE0_RXD_P/N

RX Insertion Loss: 6.5dB

The transceiver data rate is set to 5000Mbps for this port

The current settings will configure this port in CDR mode

RX CTLE: No_Peak_-0.4dB

RX Termination (ohms): 100

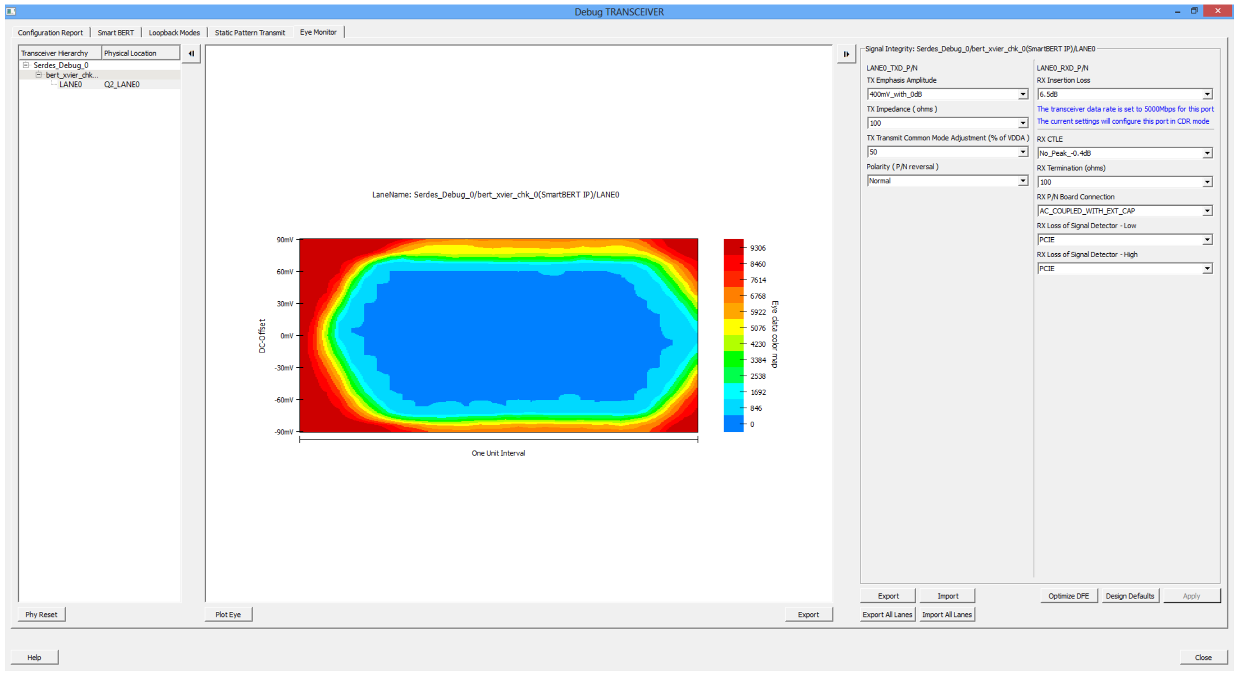
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP

RX Loss of Signal Detector - Low: PCIE

RX Loss of Signal Detector - High: PCIE

The following figure shows the **Eye Monitor** tab.

Figure 28 • **Debug TRANSCEIVER—Eye Monitor**



Signal Integrity: Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0

LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_0dB
TX Impedance (ohms): 100
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 5000Mbps for this port
The current settings will configure this port in CDR mode
RX CTLE: No_Peak_-0.4dB
RX Termination (ohms): 100
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: PCIE
RX Loss of Signal Detector - High: PCIE

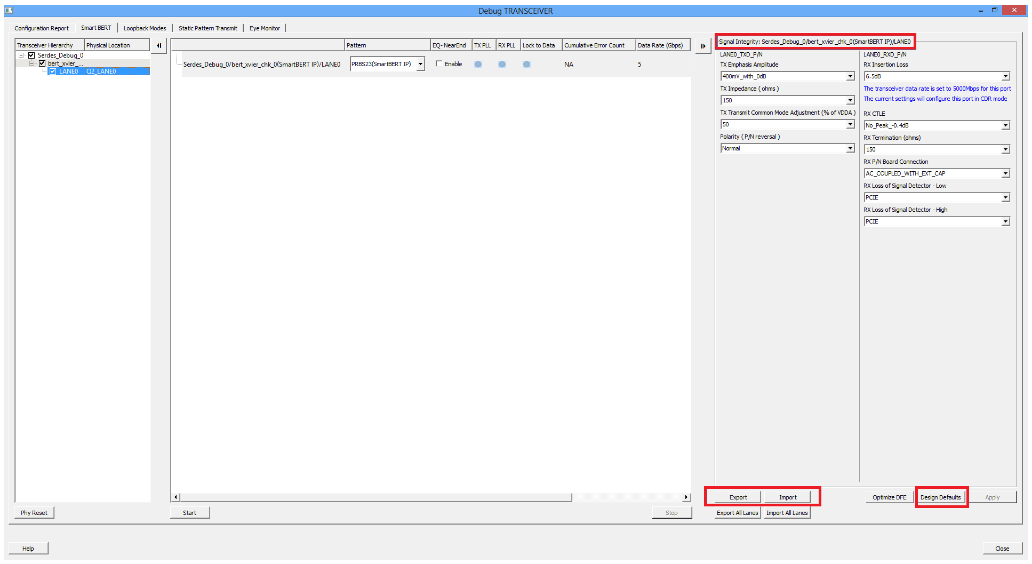
2.7.5.6 Signal Integrity

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

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

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

Figure 29 • **Signal Integrity**



	Pattern	EQ- NearEnd	TX PLL	RX PLL	Lock to Data	Cumulative Error Count	Data Rate (Gbps)
Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0	PRBS23(SmartBERT IP)	Enable	●	●	●	NA	5

Signal Integrity: Serdes_Debug_0/bert_xvier_chk_0(SmartBERT IP)/LANE0

LANE0_TXD_P/N
TX Emphasis Amplitude: 400mV_with_0dB
TX Impedance (ohms): 150
TX Transmit Common Mode Adjustment (% of VDDA): 50
Polarity (P/N reversal): Normal

LANE0_RXD_P/N
RX Insertion Loss: 6.5dB
The transceiver data rate is set to 5000Mbps for this port
The current settings will configure this port in CDR mode
RX CTLE: No_Peak_-0.4dB
RX Termination (ohms): 150
RX P/N Board Connection: AC_COUPLED_WITH_EXT_CAP
RX Loss of Signal Detector - Low: PCIE
RX Loss of Signal Detector - High: PCIE

2.7.5.6.1 Design Defaults

Click **Design Defaults** to load the signal integrity parameter options for the selected lane instance. These are the signal integrity settings selected in the Libero design flow and reside in the STAPL file. Design default parameter options are applied to the device and updated in Modified Constraints.

2.7.5.6.2 Export

Click **Export** to export the selected parameter options and other physical information to an external PDC file. A popup box prompts to choose the location where you want the .pdc file to be exported.

The exported content is in two set_io commands form—TXP and RXP ports of the selected lane instance.

2.8 Conclusion

This tutorial demonstrated capabilities of SmartDebug to observe and analyze many embedded device features. Live probes give a real-time access to device test points, and internal logic states can be accessed using active probes. The SmartDebug TRANSCEIVER 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 can be done in real-time without any design modifications. The PMA analog settings can be tuned to optimize link performance and to match the design to the system.

3 Appendix: Known Issues

This chapter lists known issues related to SmartDebug hardware design debug and provides workarounds for each of the issues.

3.1 Probe Points Write Issue

The SmartDebug reads and writes to the probe points associated with the SmartBERT IP for debugging. There is a known issue where JTAG writes to some probe points do not work. The following procedure provides a workaround to ensure that this issue does not impact the functionality of the SmartBERT IP. The workaround involves generating a constraint file that ensures the design is placed only in probe points that work.

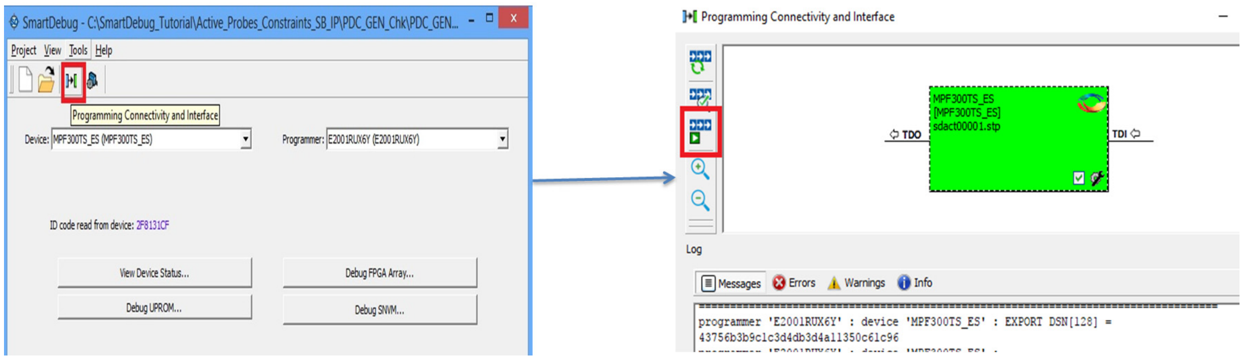
Note: This procedure be followed before running Place and Route in the Libero design flow.

The following files are provided in a Active_Probes_Constraints_SB_IP folder.

- DDC file (Full_Fabric_FF.ddc – a test design)
- TCL script (Execute_probes.tcl – execute from SmartDebug)
- Input file (Input_File.txt – edit before executing TCL)
- Reference_Files folder (internal use), which contains the following files:
  - sd.reference.pdc
  - SmartBERT_IP_Quad0.fp.pdc
  - SmartBERT_IP_Quad1.fp.pdc
  - SmartBERT_IP_Quad2.fp.pdc
  - SmartBERT_IP_Quad3.fp.pdc

1. Go to Active_Probes_Constraints_SB_IP. folder
2. Open the Standalone SmartDebug.
3. Click **Project** > **New Project** to create a new project.
4. Import DDC Full_Fabric_FF.ddc file. Click **OK**.
5. Program the design using the **Programming Connectivity and Interface** window as shown in the following figure. Close the window when the design has been programmed.

Figure 30 • **Programming Connectivity and Interface**


SmartDebug - C:\SmartDebug_Tutorial\Active_Probes_Constraints_SB_IP\PDC_GEN_Chk\PDC_GEN...
Project View Tools Help
Programming Connectivity and Interface
View Device Status...
Debug FPGA Array...
Debug UPROM...
Debug SNVM...
Programming Connectivity and Interface
MPF300TS_ES
[MPF300TS_ES]
sdact00001.stp
TDO
TDI
Log
Messages
Errors
Warnings
Info

Device: MPF300TS_ES (MPF300TS_ES)

Programmer: E2001RUX6Y (E2001RUX6Y)

ID code read from device: 2F8131CF

```
================================================================================
programmer 'E2001RUX6Y' : device 'MPF300TS_ES' : EXPORT DSN[128] =
43756b3b9c1c3d4db3d4a11350c61c96
```

6. Open `Input_Files.txt` in text editor. Enter Quad and number of lanes that are configured for CoreSmartBERT IP in the design.
7. Go back to the SmartDebug window and click **Project** > **Execute script** and enter the TCL script file path (Execute_probes.tcl).
8. Click **Run** to execute. This may take approximately three minutes to complete.
9. Close the Standalone SmartDebug.
   The output of the TCL execution is a PDC (*.pdc) file/files that can be used in the Libero flow.
10. Go to the current directory and locate the Output_PDC_Files folder.
    Generated PDC file contains a list of registers used in CoreSmartBERT IP.
11. Import the PDC files into the design.

12. Replace the top-level name for each constraint mentioned in the PDC file with the hierarchy name of the IP in the design.
13. If multiple hierarchy levels are present, include all levels in the space specified in the PDC file.

For example, if design has CoreSmartBERT IP with component name SmartBERT_IP_0, replace “<Enter_module_path_here>” with “SmartBERT_IP_0” for each location constraint in the PDC file.

14. Run the Libero flow till Run PROGRAM Action.

Note: For this demo design, replace <Enter_module_path_here> with XCVR_Debug_0/SmartBert_xcvr_chk_0. Use the updated pdc file instead of the file provided in the design (place_pll.pdc).

3.2 Data Traffic Errors on XCVR Lanes in CDR Mode

While plotting the eye using eye monitor, errors are introduced in data traffic on transceiver lanes configured to use the CDR receiver path. The errors are introduced when DFE and EM blocks are turned off during normal operation to save power. This issue does not impact the functionality. The cumulative error count and BER values can be ignored when plotting the eye. A software update will be provided in future Libero releases to fix the issue.

4 Appendix: References

This section lists documents that provide more information about the SmartDebug and IP cores used in the reference design.

- For more information about SmartDebug, see *UG0743: PolarFire FPGA Debugging User Guide* and *UG0773: PolarFire SmartDebug User Guide*.
- For more information about PolarFire transceiver blocks, see *UG0677: PolarFire FPGA Transceiver User Guide*.
- Fore more information about PF_CCC, see *UG0684: PolarFire FPGA Clocking Resources User Guide*.
- For more information about Libero, see the *Microsemi Libero SoC PolarFire* web page.
- For more information about PolarFire FPGA Splash Kit, see *UG0786: PolarFire FPGA Splash Kit User Guide*.
- For more information about PF_UPROM, PF_URAM, and PF_DPSRAM, see Libero catalog.