Future Electronics – Microsemi
PolarFire Avalanche Board

SmartDebug / SerDes Demo

This demo is based on the 3rd hands-on laboratory from our PolarFire Avalanche seminar series held in 2018.

Overview

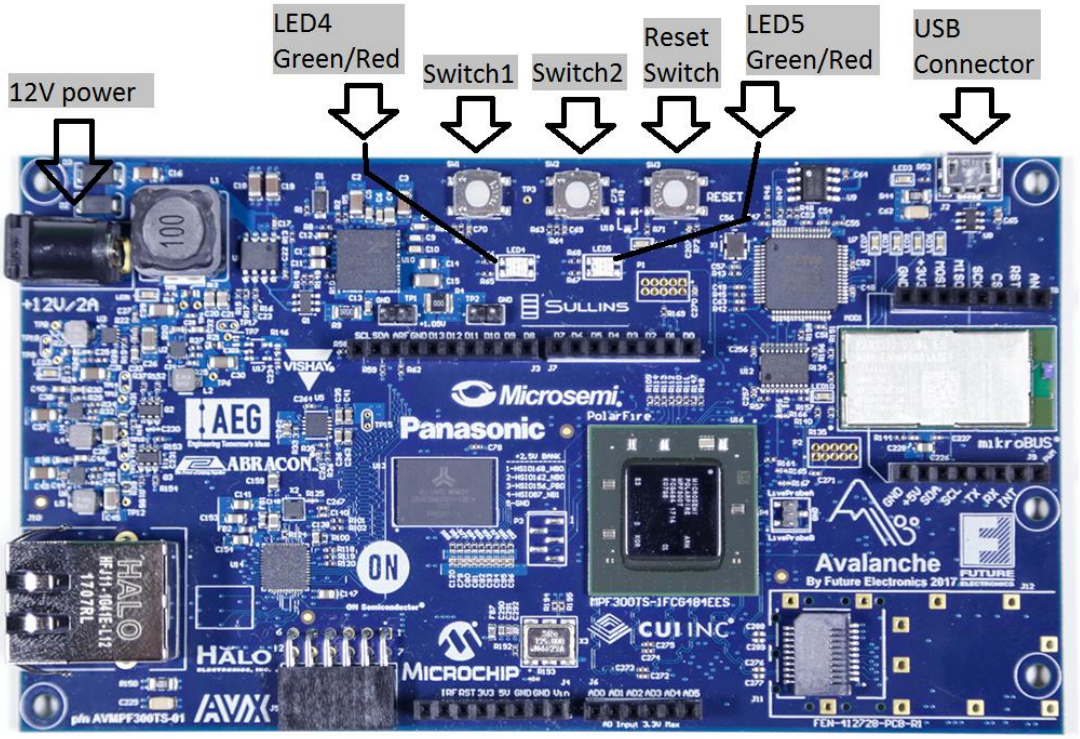
SmartDebug enables debugging a design by allowing verification and troubleshooting at the hardware level, provides access to probe points, non-volatile memory (NVM), fabric memory and transceivers.

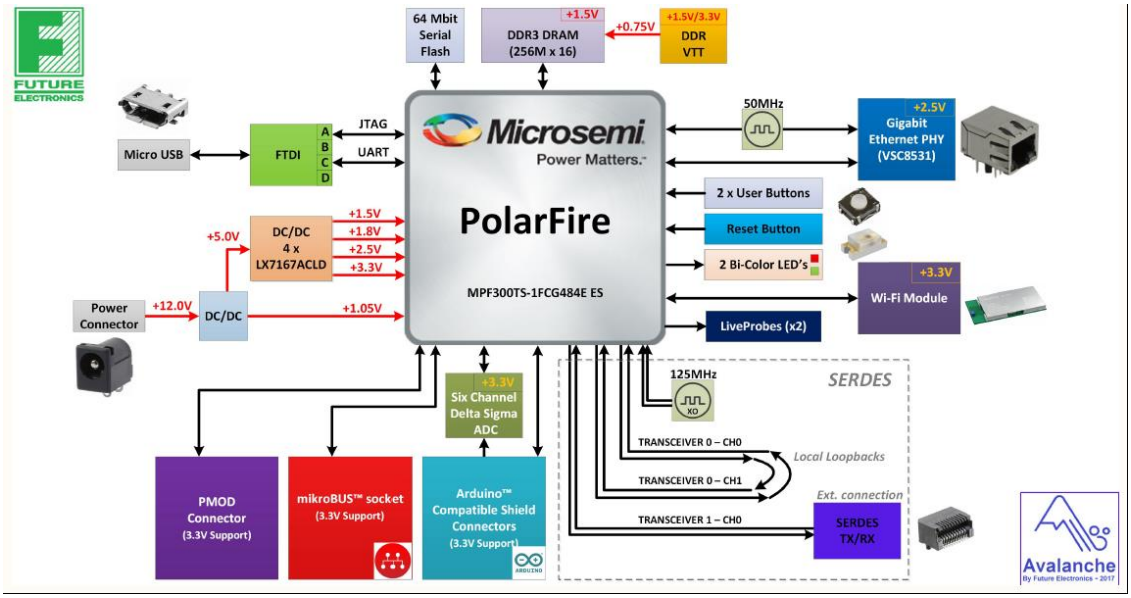
This design example implements SERDES Loop back to show the SMART Debug implementation with an eye diagram for test purposes.

The Avalanche Board

SW1, SW2, Green LED and Red LED located on top board edge

USB connector (for power and programming located on top right of board edge)

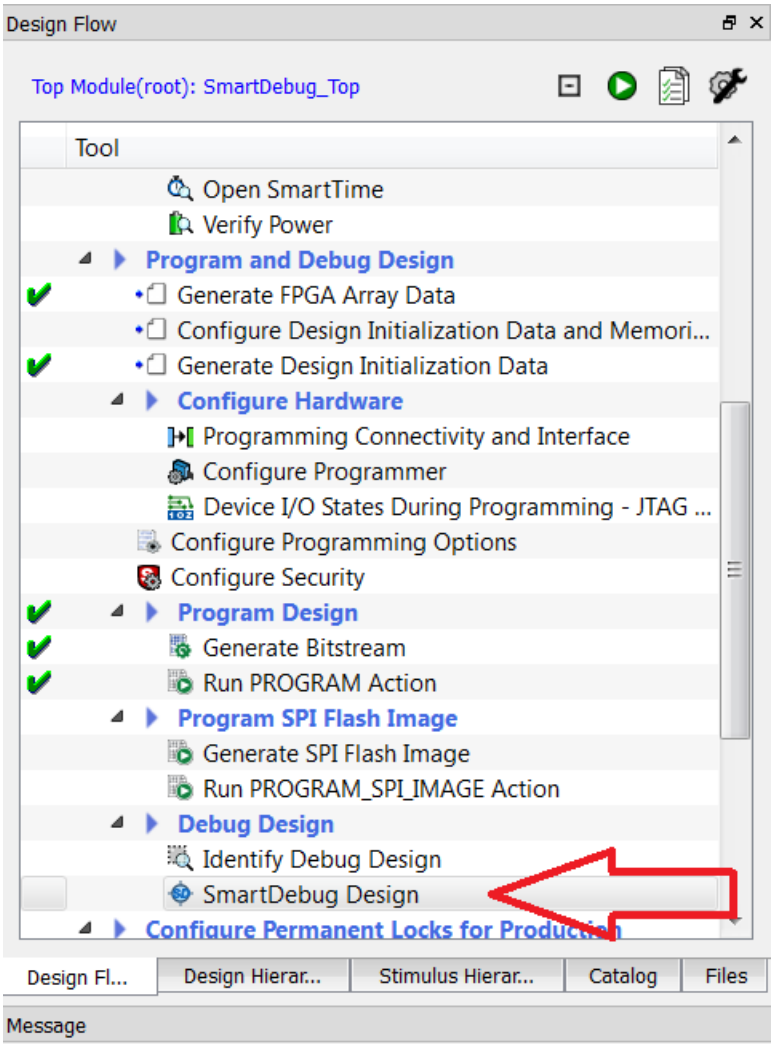




See appendix A for a block diagram of the FPGA design.

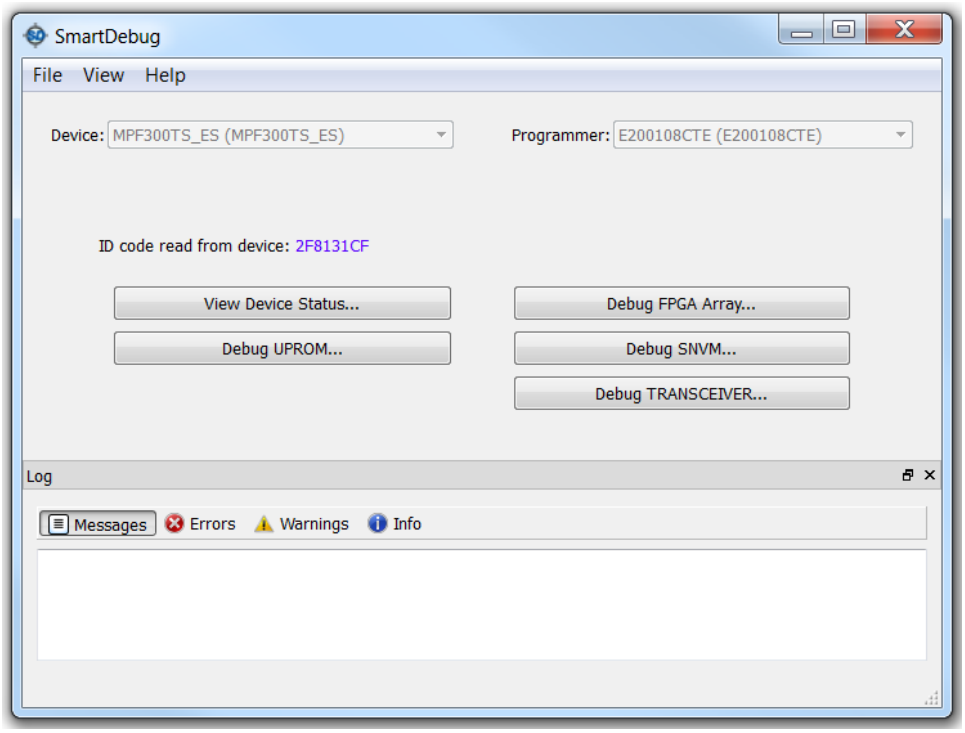
SmartDebug GUI

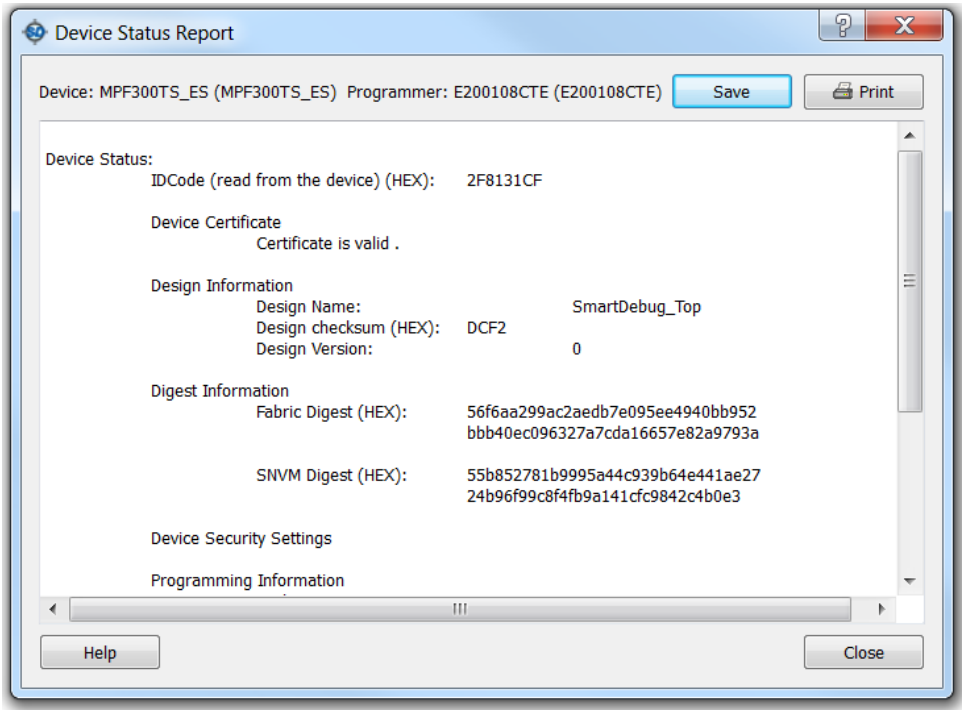
To invoke SmartDebug in the Integrated mode, expand Debug Design and double-click SmartDebug Design.



Device Status

Click on ‘View Device Status’ for a complete summary of ID Code, device certificate, design information, programming information, digest, and device security information, as shown in the following figure.

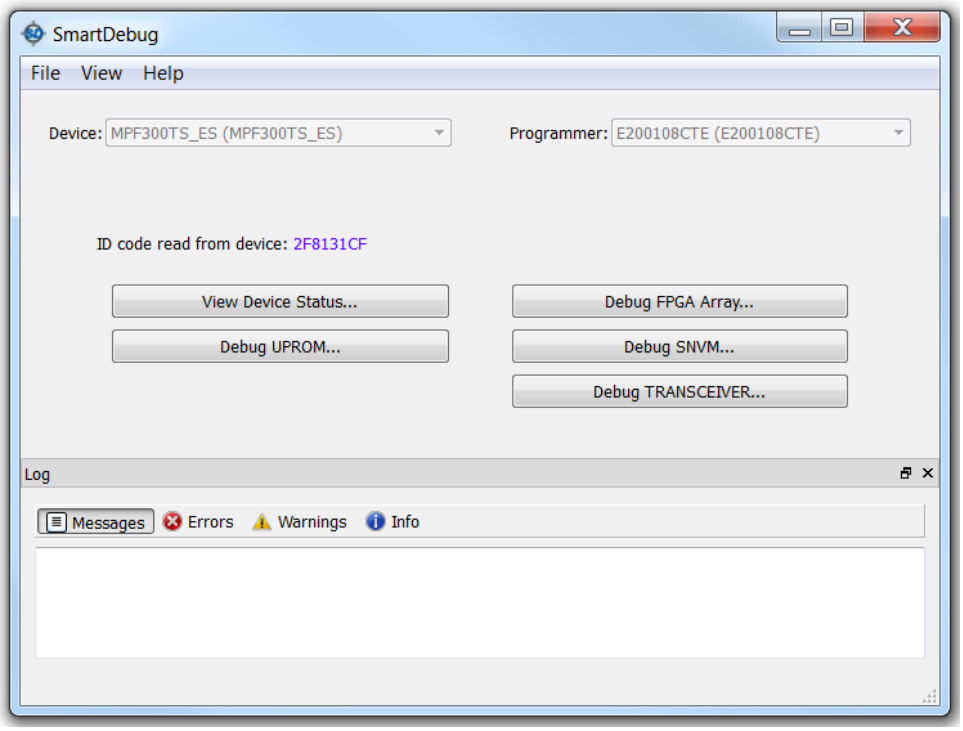




Debug FPGA Array

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.

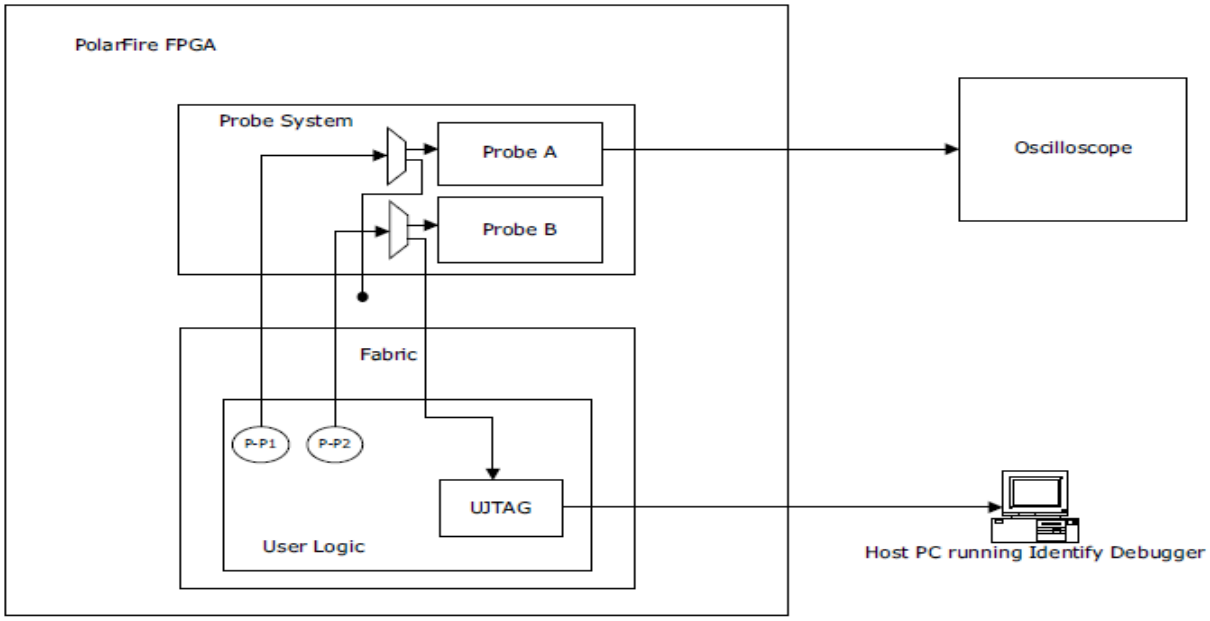
Click on ‘Debug FPGA Array’.



Live Probes

Live Probes enables the monitoring of two internal signals at a time in the design without having to rerun place and route.

PolarFire devices have two dedicated live probe channels (for example, pin A10 and A11 of PolarFire MPF300TS device in the FG484 package). 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.

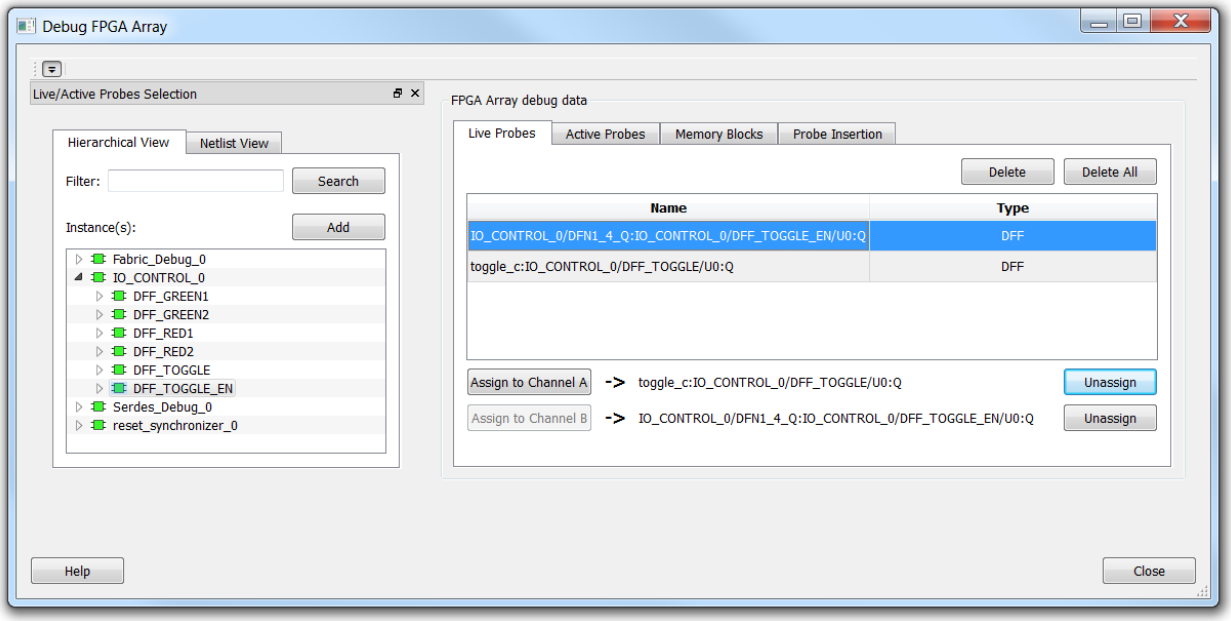


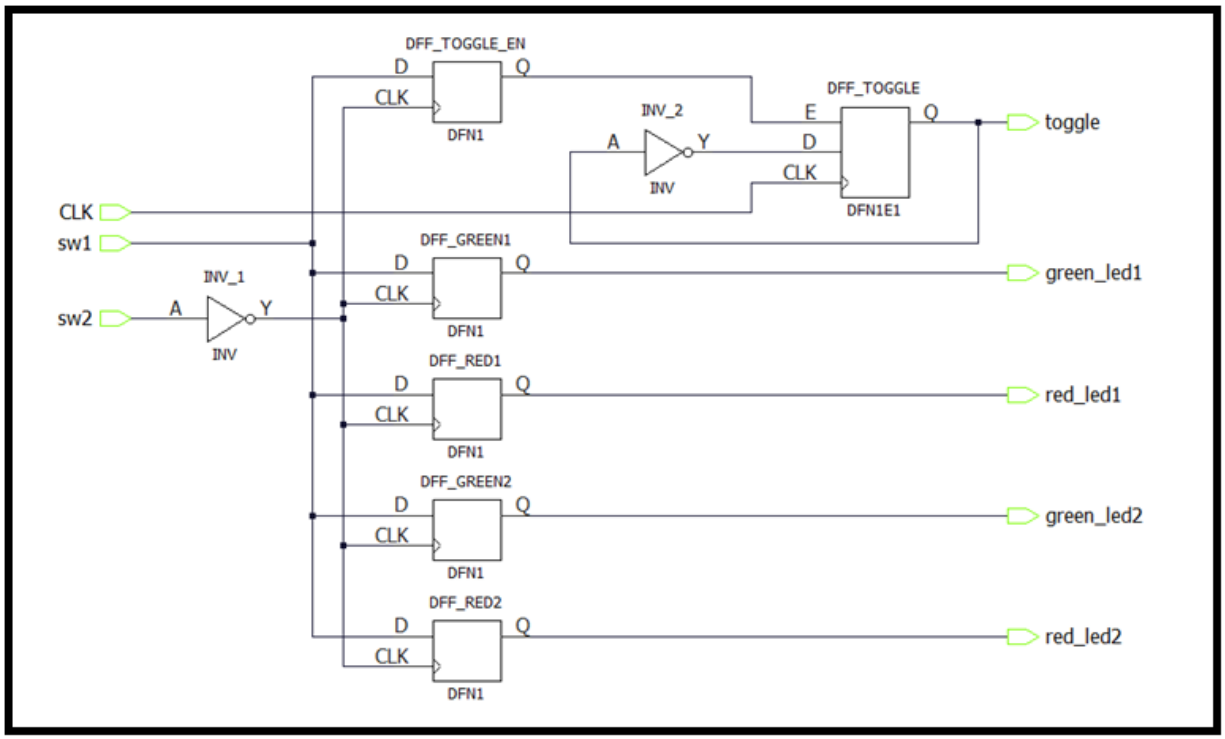
Live Probes, SmartDebug in conjunction with Oscilloscope block diagram

Debug FPGA Array defaults to the Live Probes tab.

Double click on DFF_TOGGLE and DFF_TOGGLE_EN in the left column to add to the Live Probes windows. Highlight DFF_TOGGLE_EN and then click on Assign to Channel A. Highlight DFF_TOGGLE and then click on Assign to Channel B.

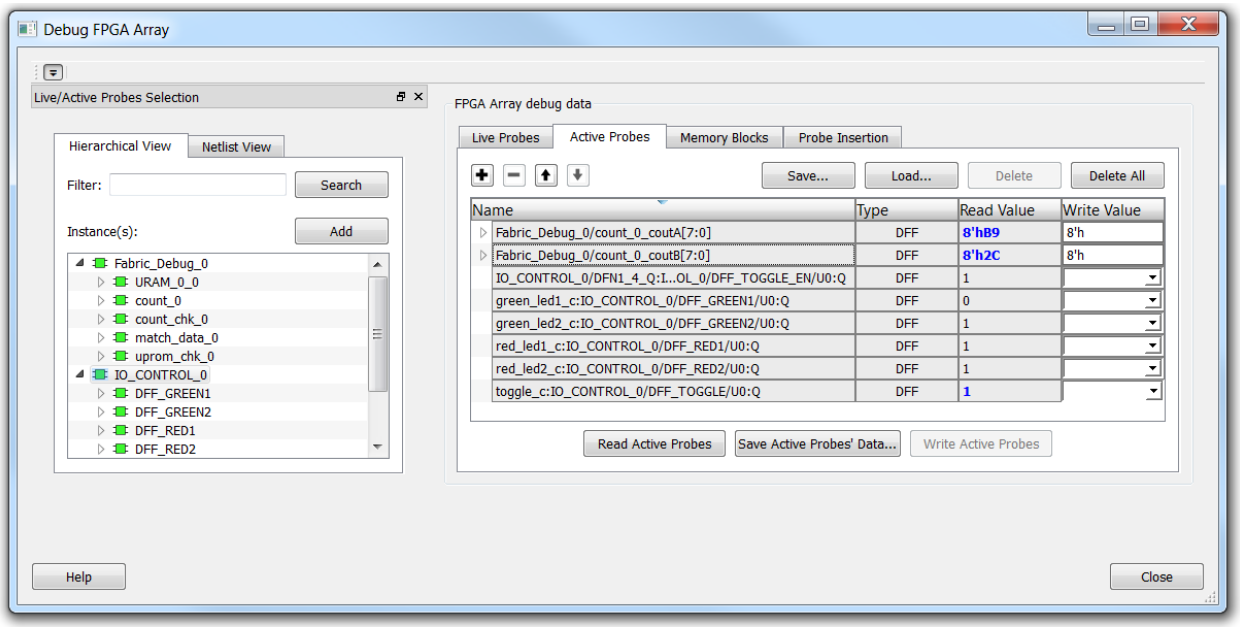
The two Live Probe pins are now connected to these signals.





Active Probes

Active Probes enable you to read or change the values of probe points in a design through JTAG.



Click on the Active Probes tab.

In the left column, double-click on IO_CONTROL. Click once on Fabric_Debug_0 to open it and then double-click on count_0.

Click on Read Active Probes and you will see the read values.

On the board LED4 and LED5 each have a Green and Red LED. You can turn them on and off by entering a 1 or 0 in the Write Value box and Pressing Write Active Probes (1 = off, 0 = on).

Repeatedly click Read Active Probes. You will see TOGGLE change values because the TOGGLE DFF has a free running clock. If you enter 0 in the TOGGLE_EN Write Value and click Write Active Probe, it will disable the Toggle Clock. Press Read Active Probes to verify.

Repeatedly click Read Active Probes to see the count_0_coutA and count_0_coutB values changing.

.

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.

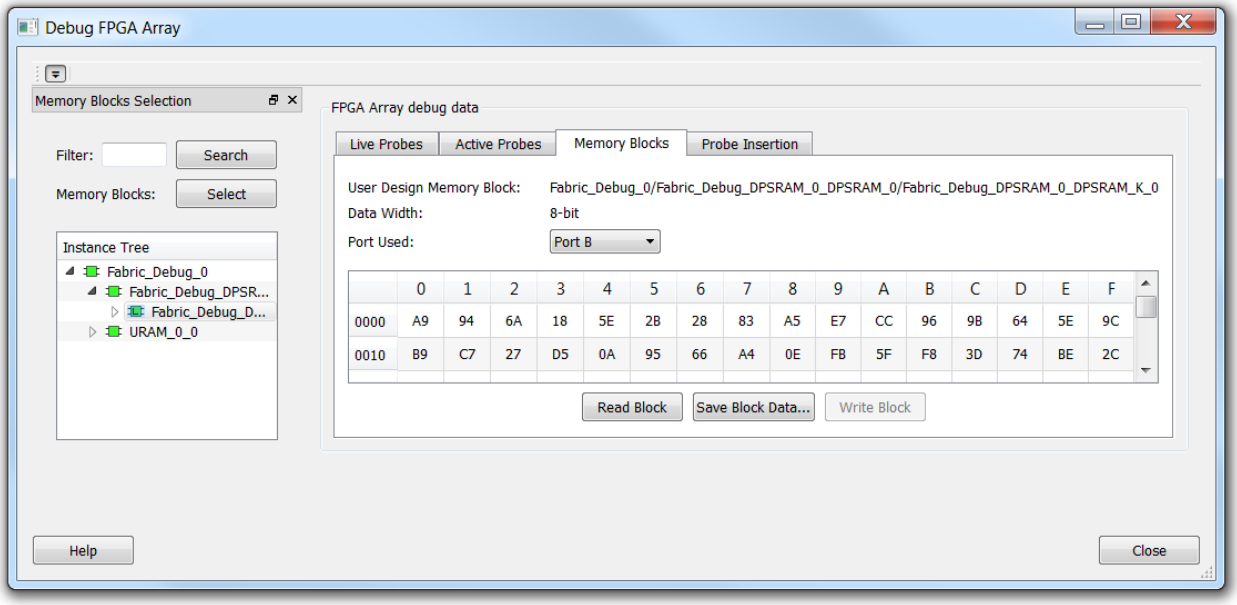
Click on the Memory tab.

In the left column click once on Fabric_Debug_0.

Click once on Fabric_Debug_DPSRAM_0_DPSRAM_0.

Double-click on Fabric_Debug_DPSRAM_0_DPSRAM_K_0.

This should add the DPSRAM to the Memory Blocks window.



Click on Read Block.

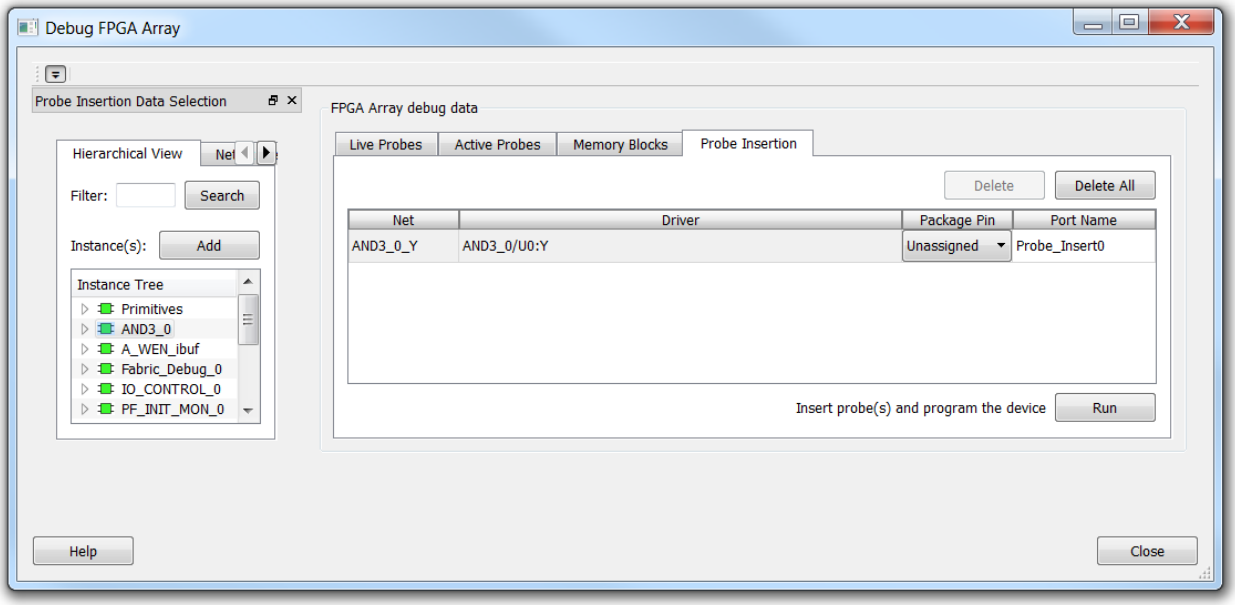
Double-click in one of the memory blocks and change the value then click on Write Block.

Probe Insertion

Probe Insertion is the only SmartDebug feature that requires the FPGA to be reprogrammed. A partial recompile is invoked from SmartDebug when using Probe Insertion.

Probe Insertion enables connection of internal ports in the design to used or unused pins.

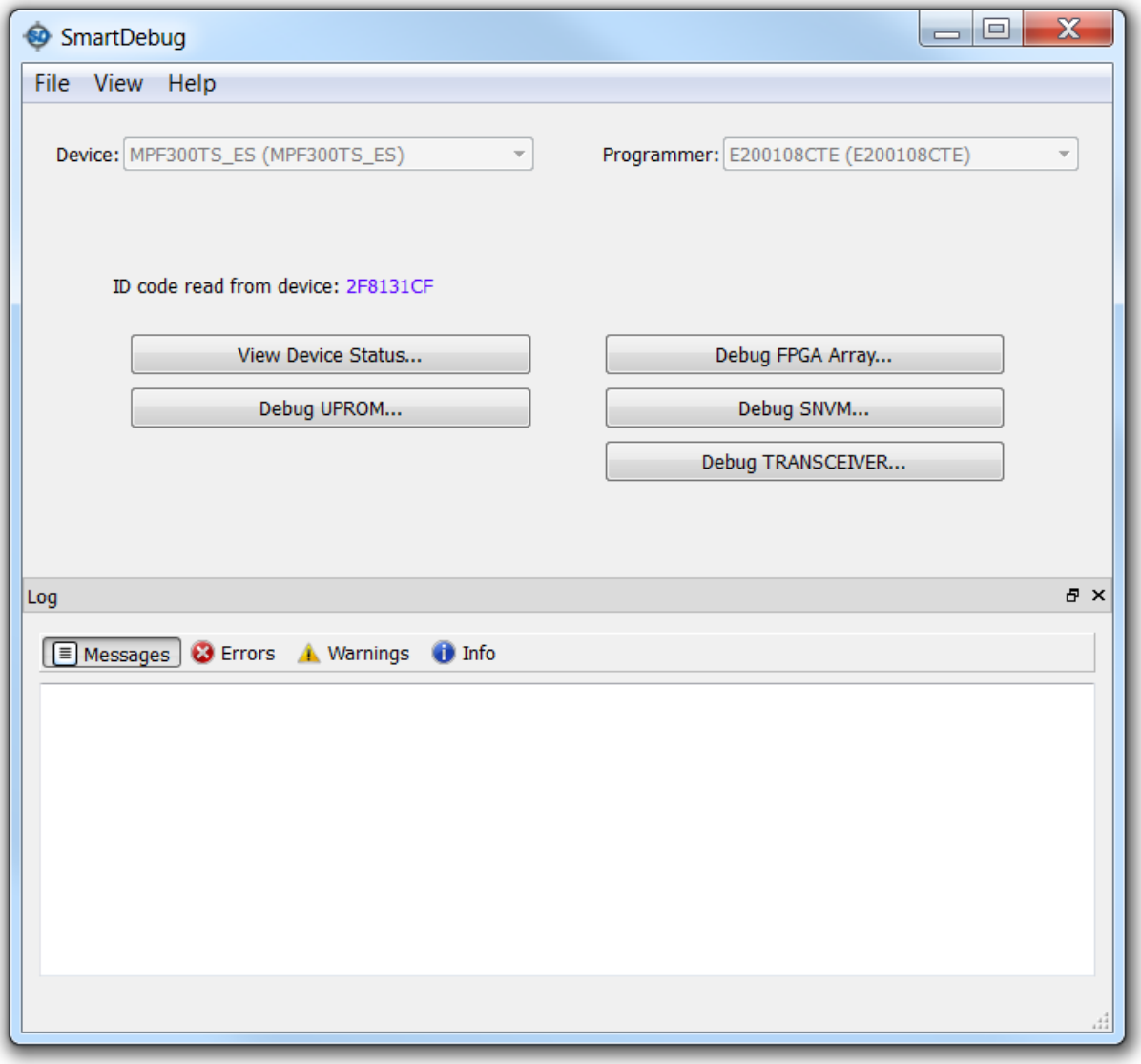
Do not click run during the Workshop because this feature requires a significant amount of time to complete.



Close the Debug FPGA Array window.

Debug µPROM

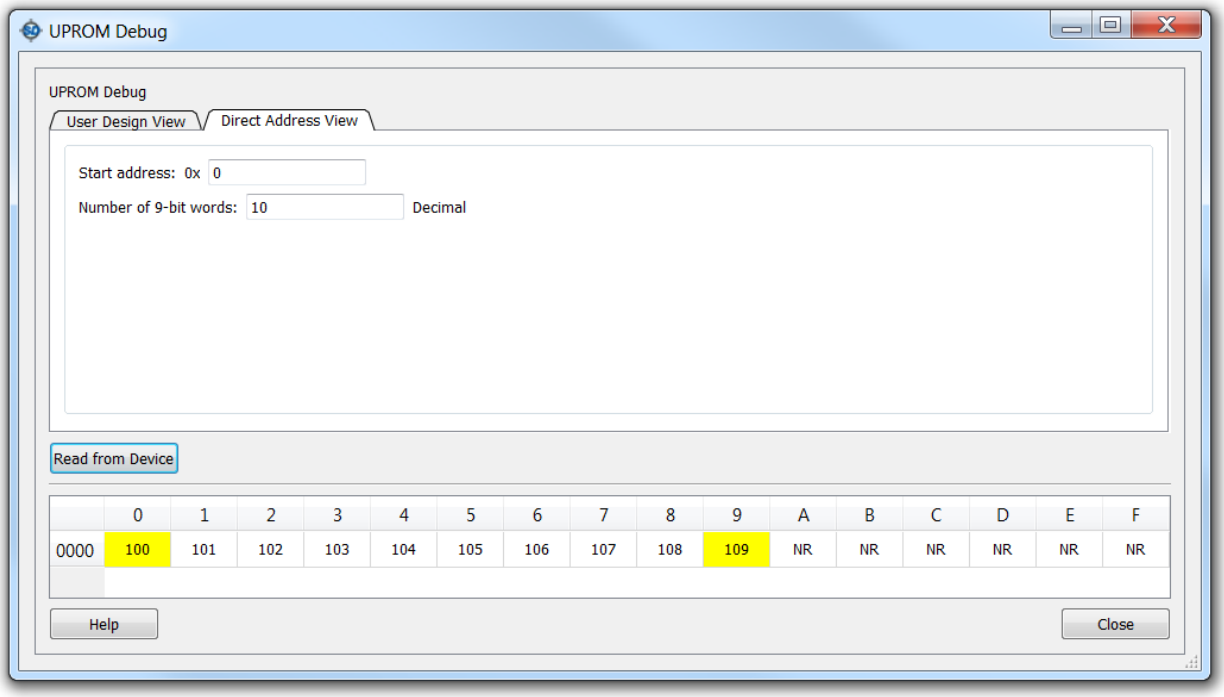
SmartDebug enables debugging µPROM and reading its µPROM contents. The clients added in the design can be debugged using the SmartDebug Debug uPROM feature.



Click on Debug UPROM.

In the UPROM Debug window click on the Direct Address View tab.

Enter zero in the Start Address box and 10 in the Number of 9-bit words box.



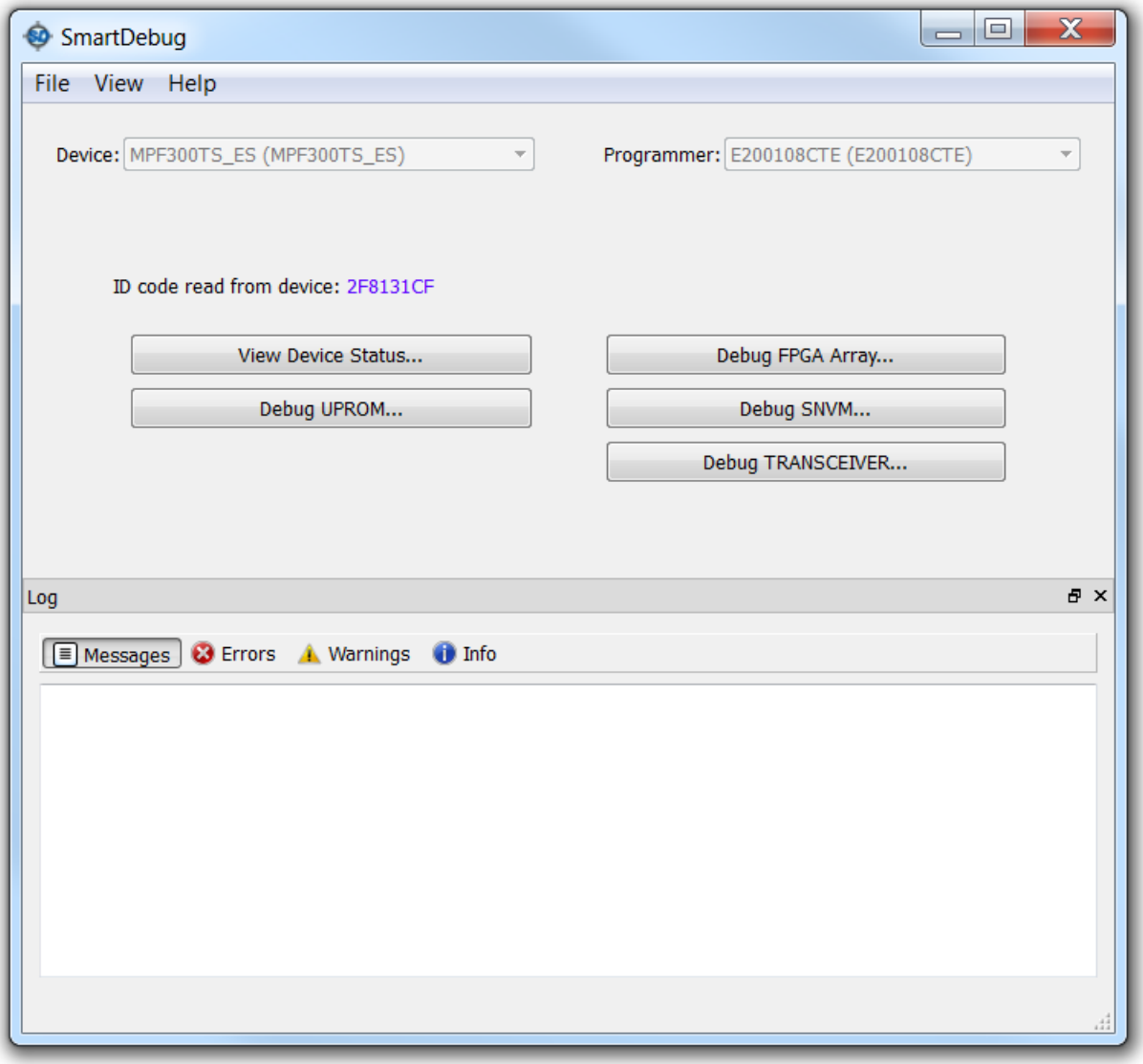
Click on Read from Device

You should see 10 UPROM memory locations.

sNVM Debug

sNVM Debug enables reading from and writing to the sNVM during debug. It can be done by reading each page or reading multiple pages based on the authentication. Debug Pass Key is required to carry out sNVM_DEBUG instruction.This feature supports debugging of plain text non-authenticated, and authenticated plain text, and clientscipher authenticated.

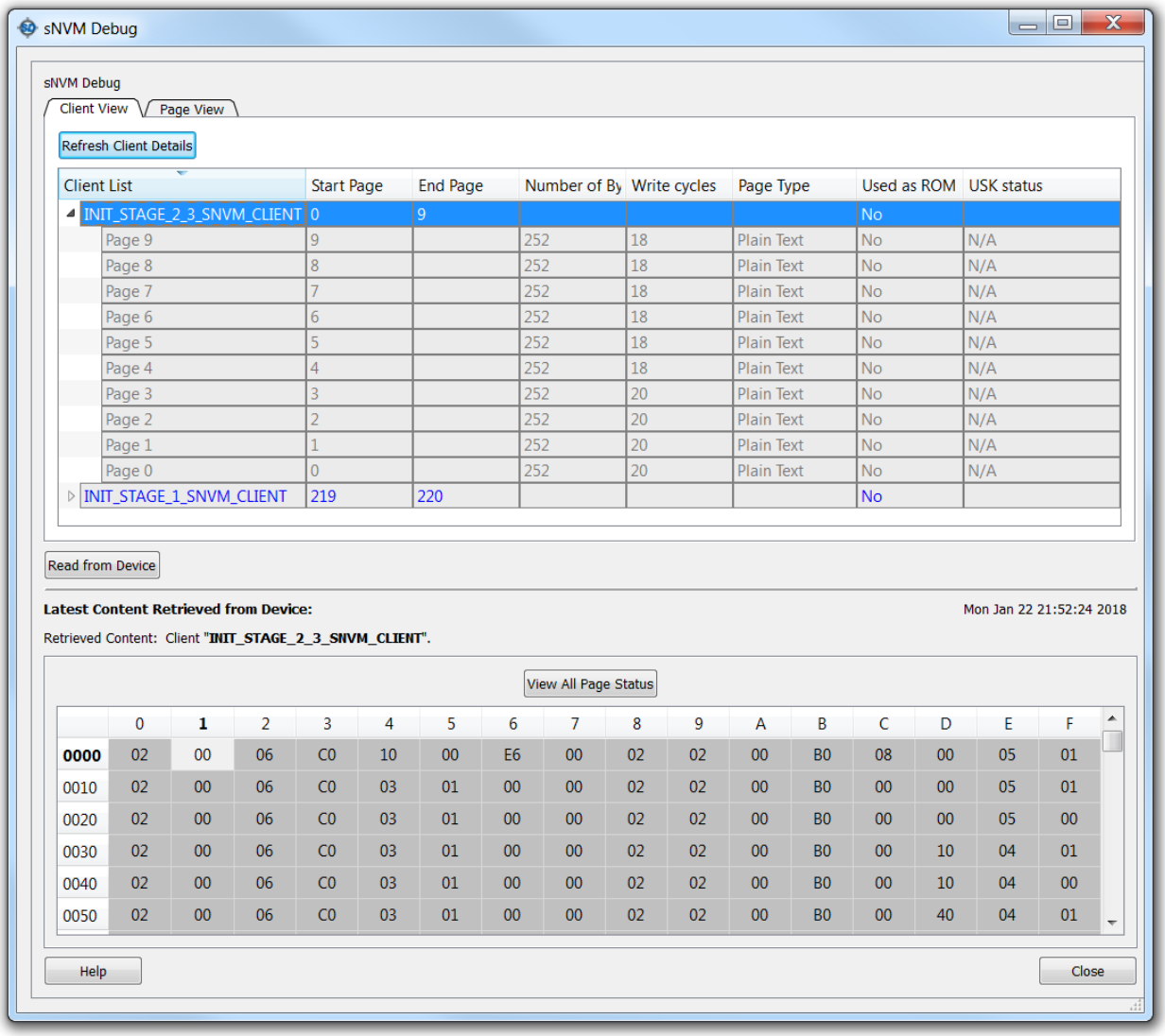
Click Debug sNVM in the SmartDebug window. The sNVM Debug window is displayed.



Click the Client View tab. The client view details are listed. It shows Client Names, Start Page, Number of Bytes, Write Cycles, Page Type, Used as ROM, and USK Status.

Double-click on a client to open it.

Click on Read from Device.



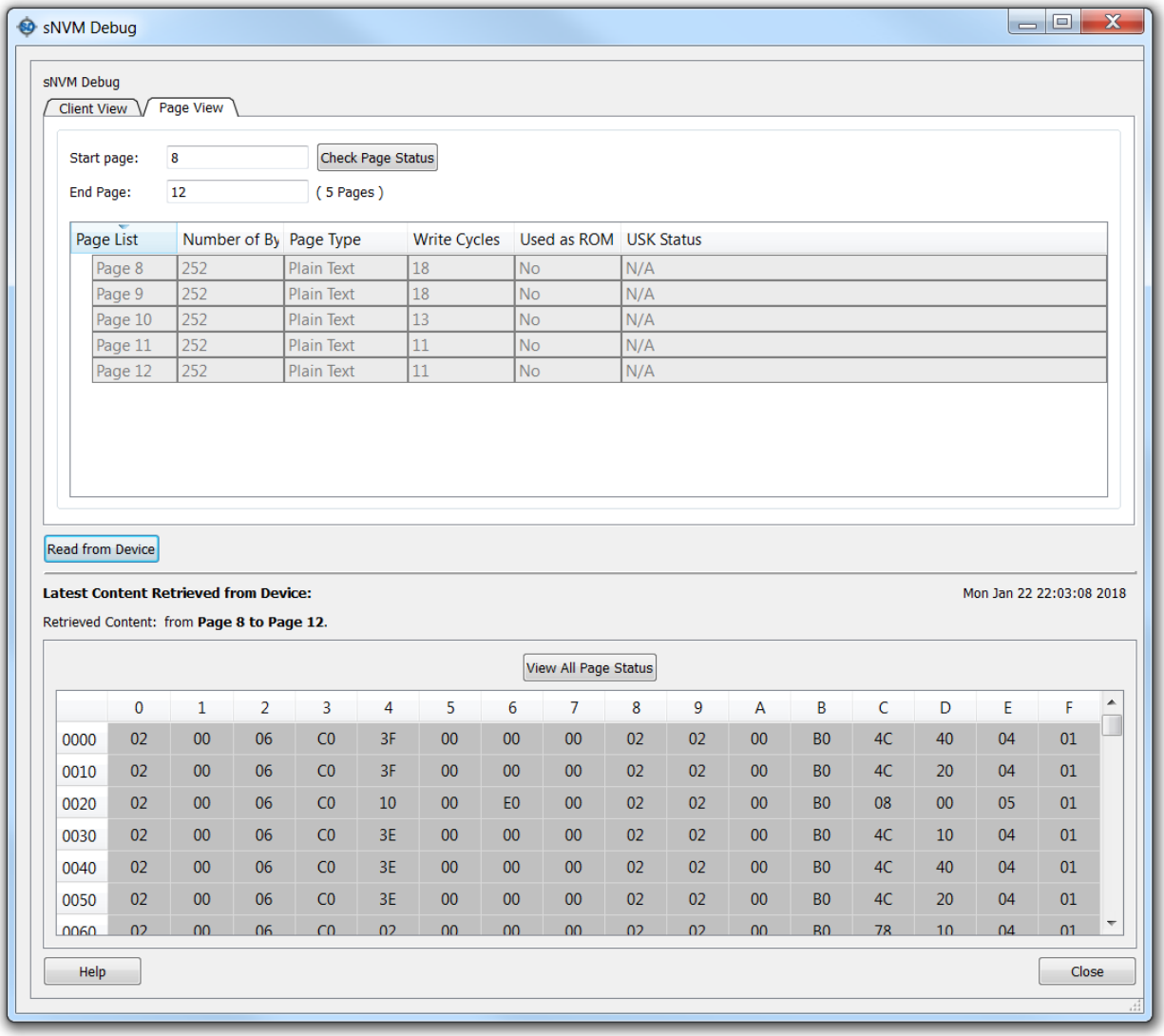
Page View

Click on the Page View tab.

Enter 8 in the Start page box and 12 in the End Page box.

Click Check Page Status.

Click Read from Device.



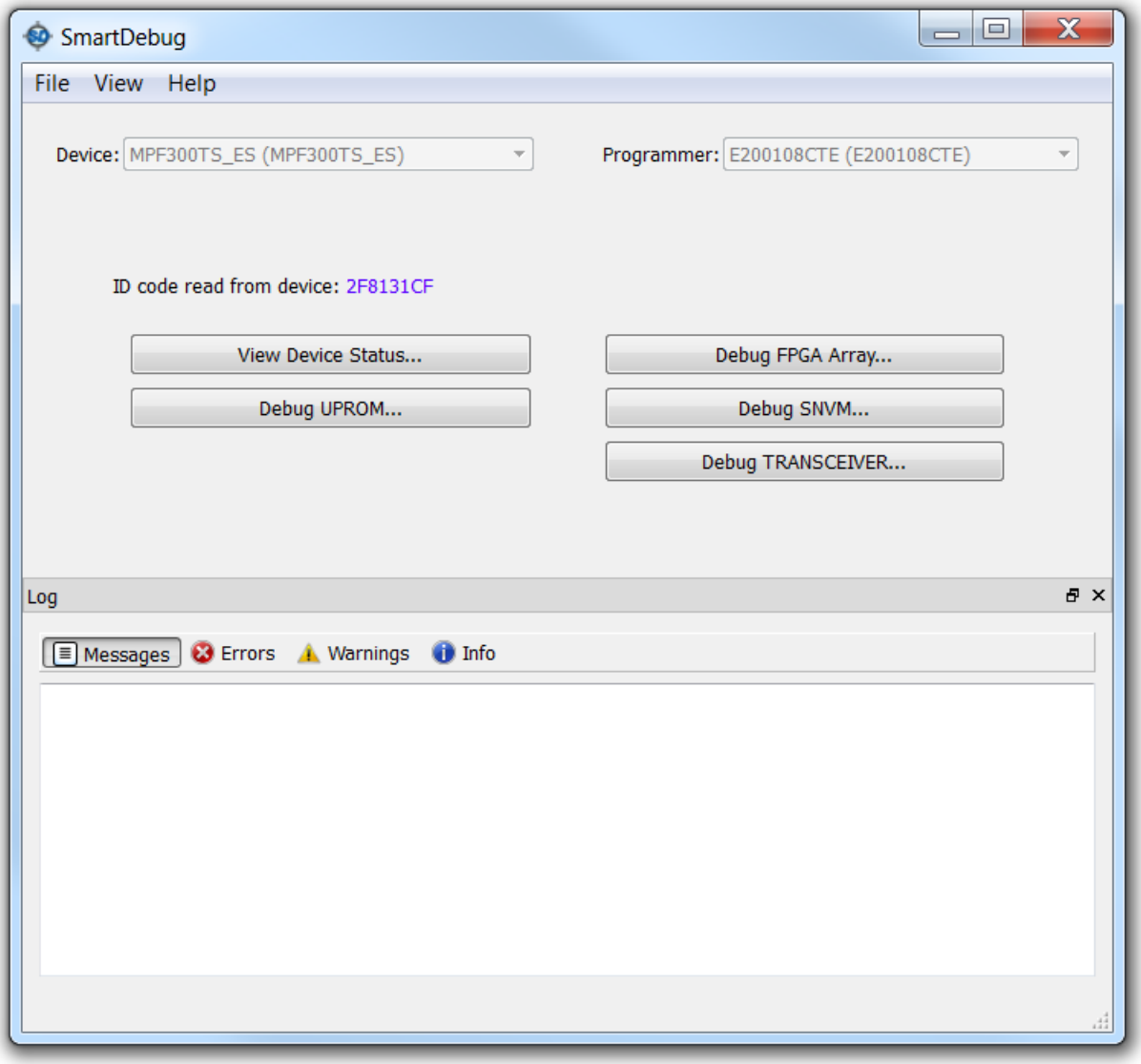
Close the sNVM window.

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



Click Debug TRANSCEIVER

SmartBERT (BERT = Bit Error Rate Test)

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. Near-end loopback can be performed using one of these PRBS patterns. Bit Error Rate (BER) displays the BER for the PRBS test in progress.

Click on the SmartBERT tab.

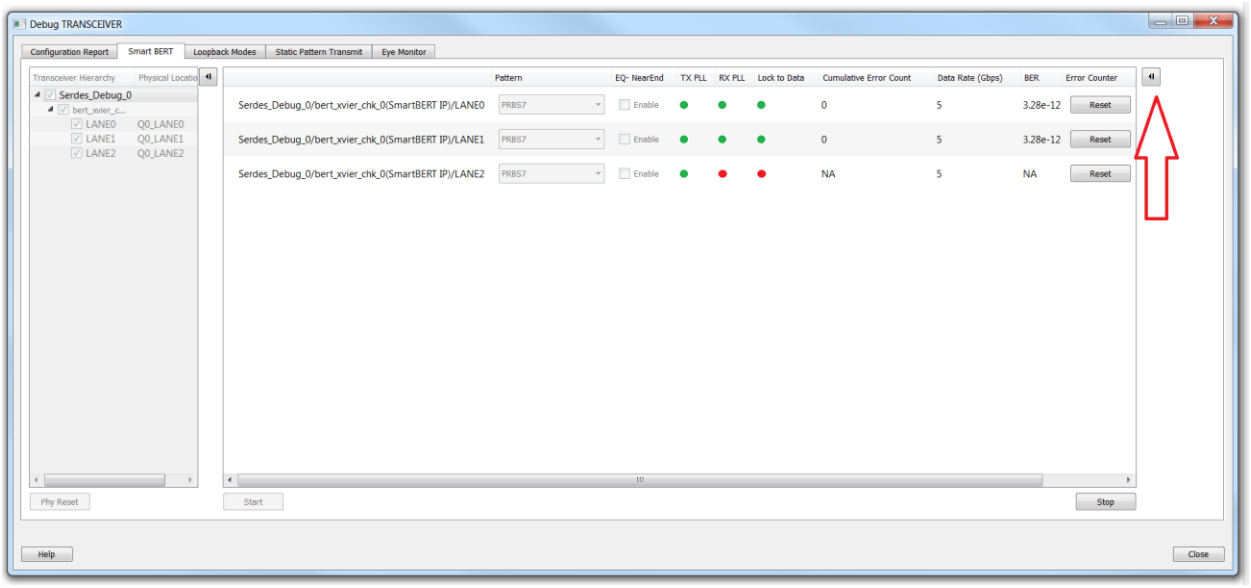
Click the <| (red arrow) to close the Signal Integrity window.

Click Serdes_Debug_0 in the left window. This will enable all 3 Serdes Lanes.

In the Pattern column, select PRBS7 for all 3 lanes.

Click start at the bottom of the window.

You should see the following:



Lanes0 and Lane1 are looped together on the board. The RX PLL column should show Green for both Lane 0 and 1. Lane 2 RX PLL should show red because Lane 2 is not connected to anything at this time (It is connected to the SFP connector).

Notice the EQ-NearEnd column. This could be used to loop back a Lane to itself internally. Don't enable it for this test.

Click Stop at the bottom of the window.

Error Injection

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.

Error Count

Error Count is displayed when the lane is added and PRBS pattern is run. Click Reset to clear the error count under Error Counter.

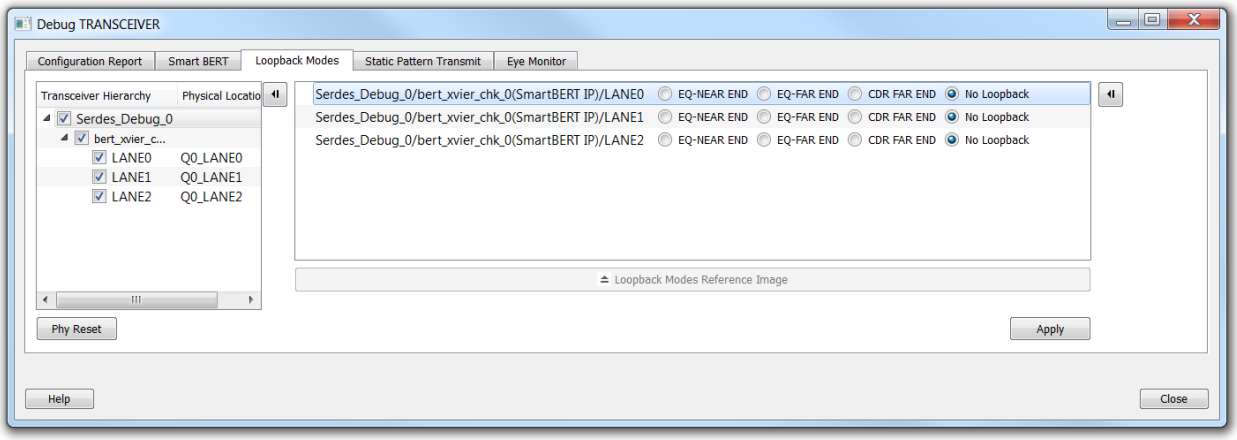
Loopback Modes

Loopback modes help you perform the following types of loopback tests:

- EQ-Near End Looback—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 Looback supports data transmission rates of up to 10.315 Gbps.
- EQ-Far End Looback—serialized data from Rx is looped back to Tx in PMA. This is called far-end serial loopback. EQ-Far End Looback supports data transmission rates of up to 1.25 Gbps.
- CDR-Far End Looback—de-serialized data from PCS Rx channel is looped back to Tx.
- No-Loop Back—data is not looped internally.

Click on Loopback Modes

Observe this window but don't make any changes.



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

Static Pattern tab

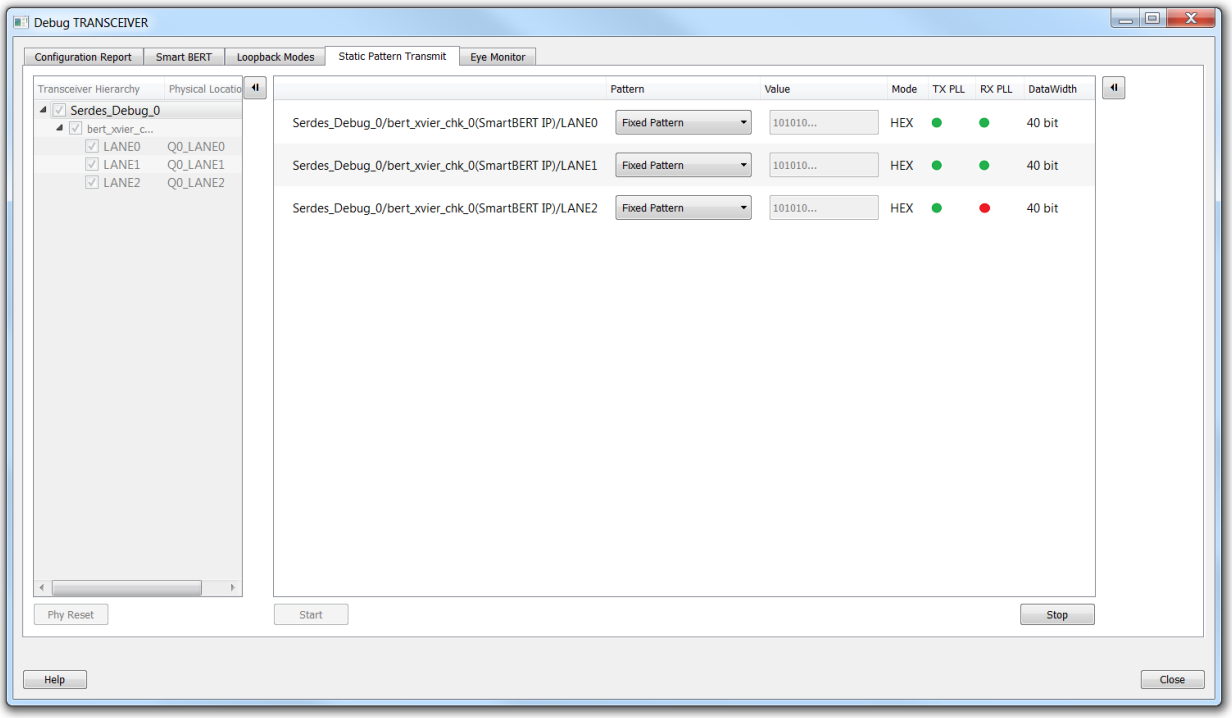
- TX-PLL—indicates lane lock onto TX PLL when a static pattern is transmitted.
- RX-PLL—indicates RX PLL lock when a static pattern is transmitted.
- Data Width—displays the data width configured for a transceiver lane.

Click on the Static Pattern Transmit tab.

Click on Serdes_Debug_0 in the left window to enable all 3 lanes.

Click on Start at the bottom of the window.

You should see the following:



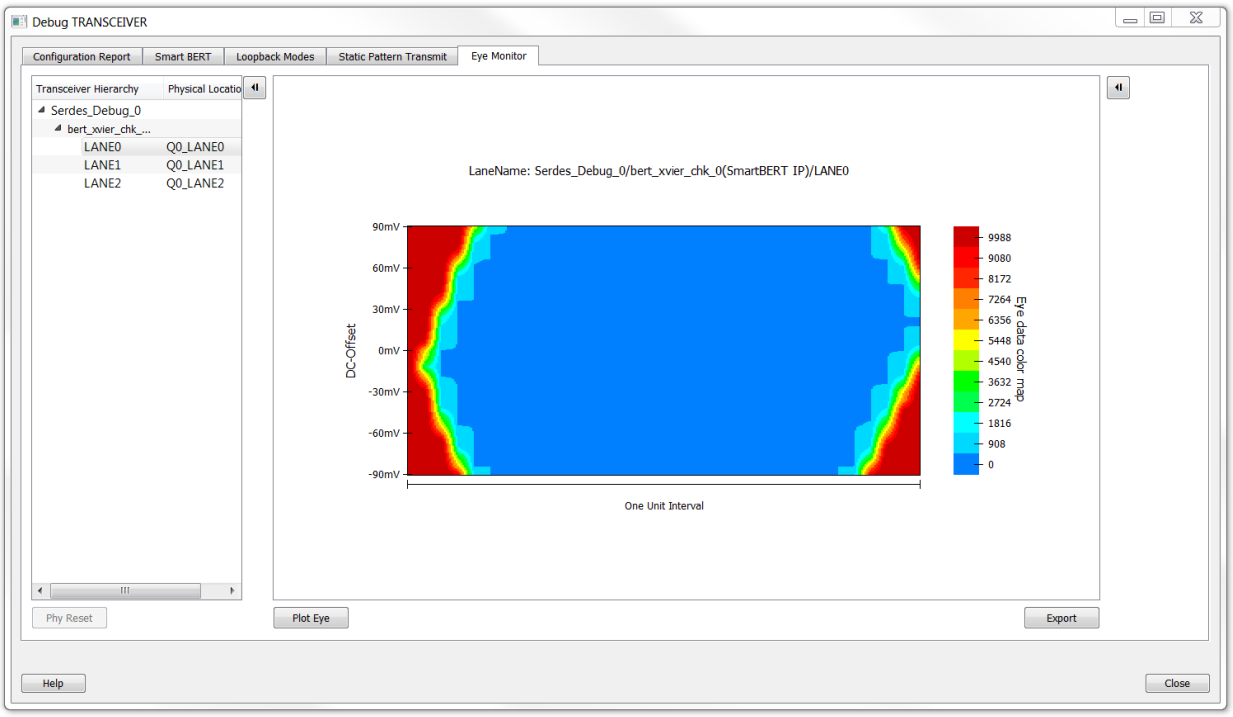
While the Static Pattern Transmit is running, click on the Eye Monitor tab.

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.

Expand Serdes_Debug_0 in the left window and click on Lane0.

Click on Plot Eye at the bottom of the window.

You should see the following:

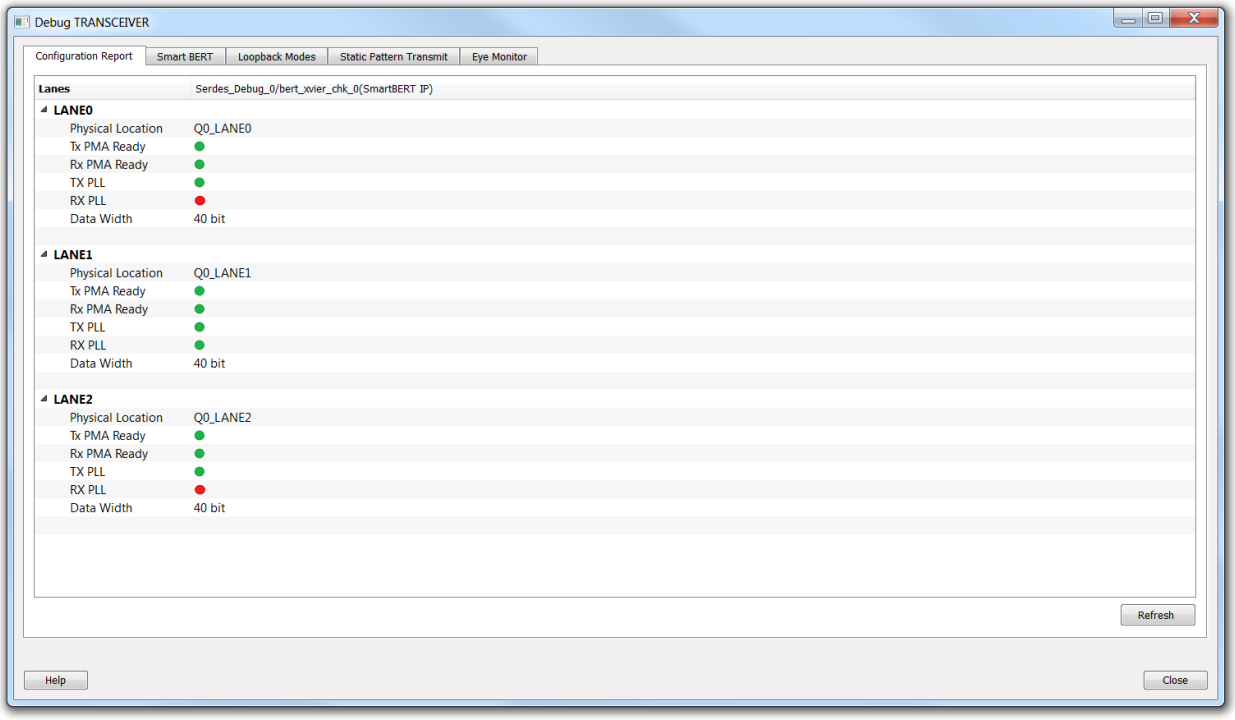


Configuration Report

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.

Click on the Configuration Report tab.



You must go back to the Static Pattern Transmit tab and click on Stop at the bottom of the page before you can close the Debug Transceiver window.

Signal Integrity

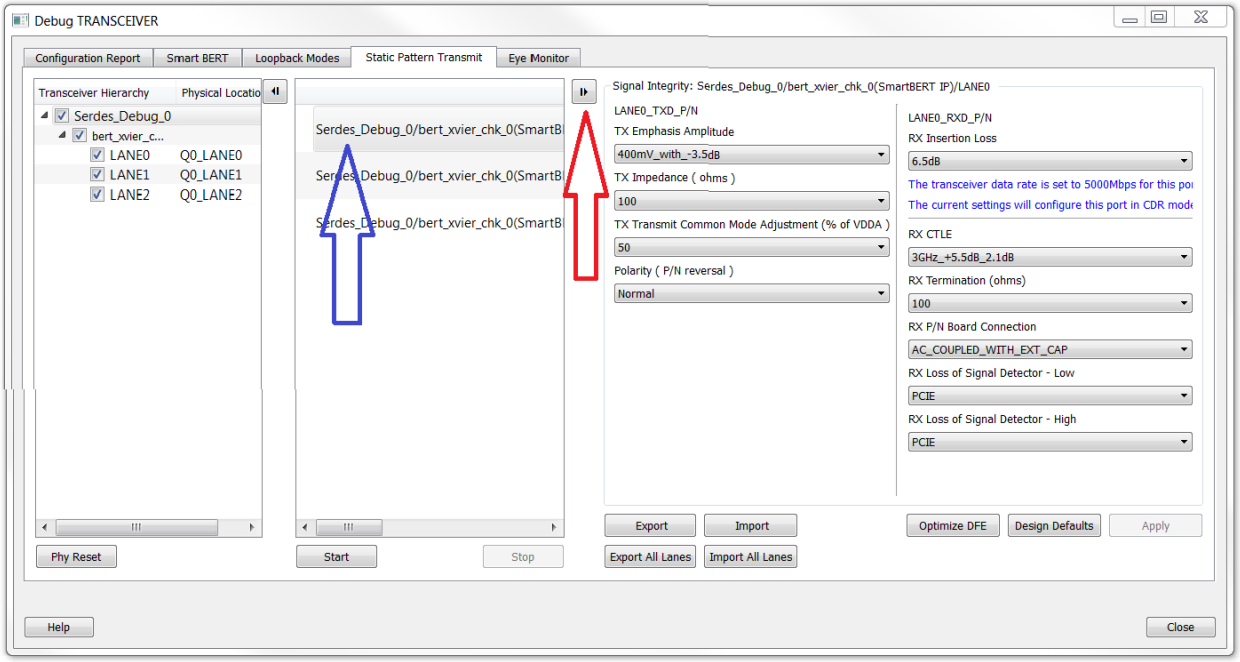
The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of .pdc files.

The Signal Integrity pane appears in the following SmartDebug pages:

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

Click on the Static Pattern Transmit tab.

Click on the <| (red arrow) to expand the Signal Integrity window.



Click on the Serdes_Debug_0/bert_xier… (blue arrow) to make the Signal Integrity window active to Lane0.

The parameters can be used to “tune” the Serdes Lanes.

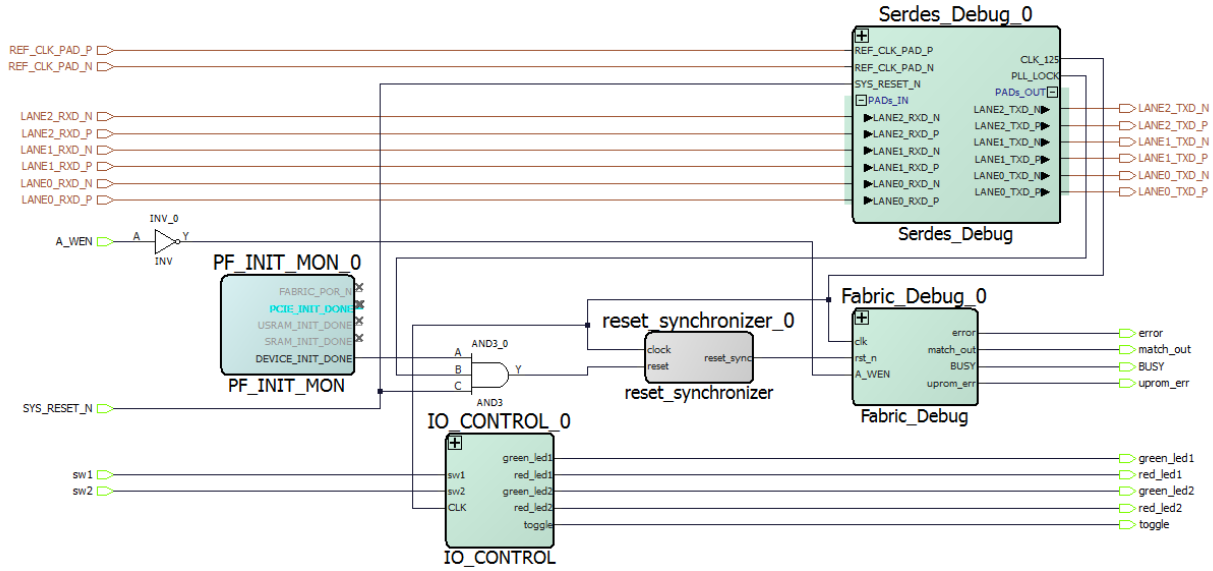
Once the optimal settings are determined, these setting can be exported and used to configure the Serdes in the FPGA.

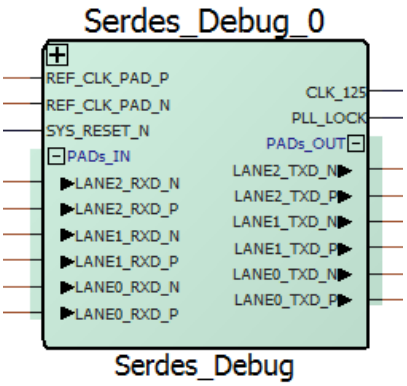
Without SmartDebug, you would make these setting in the FPGA, run place and route, program the FPGA and test the Serdes Lanes. Each iteration would require repeating this process. SmartDebug allows you to minimize or eliminate these iterations.

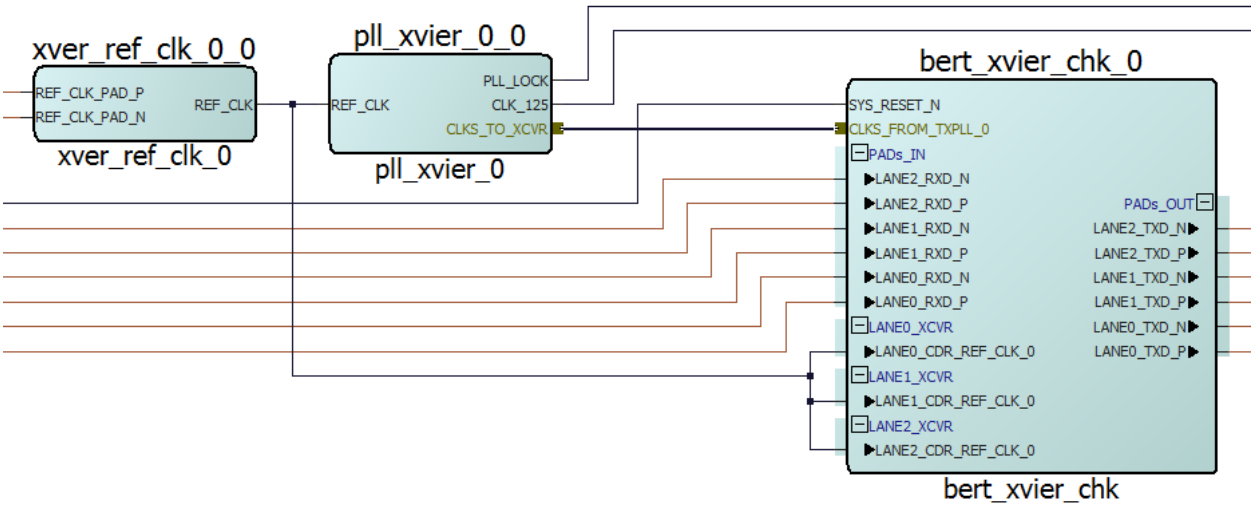
Congratulations!!! Lab 3 is now complete.

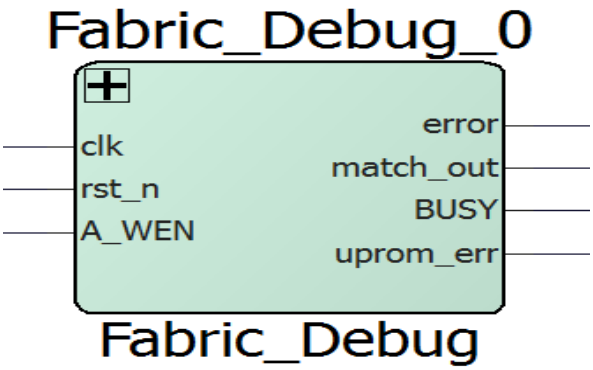
Appendix A:

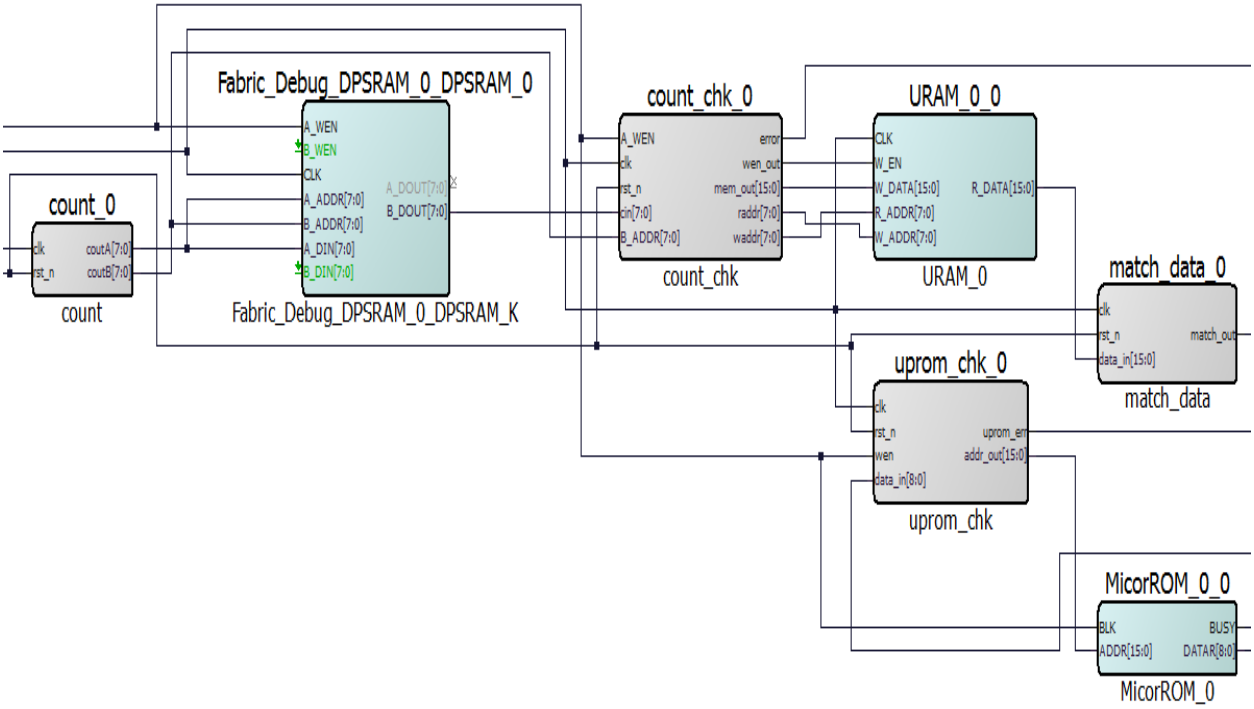
Tutorial Design Block diagram

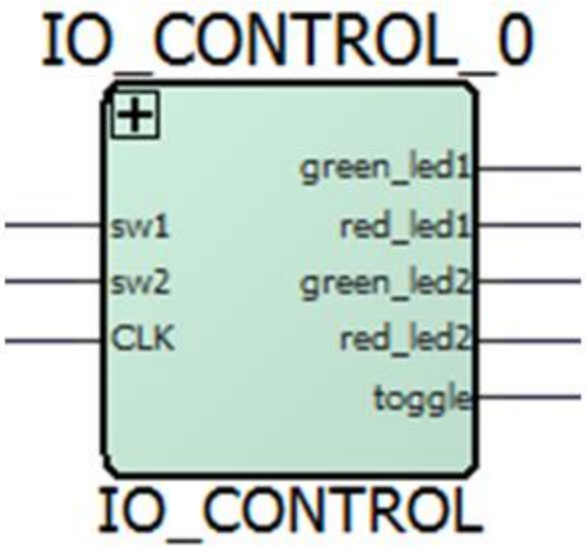


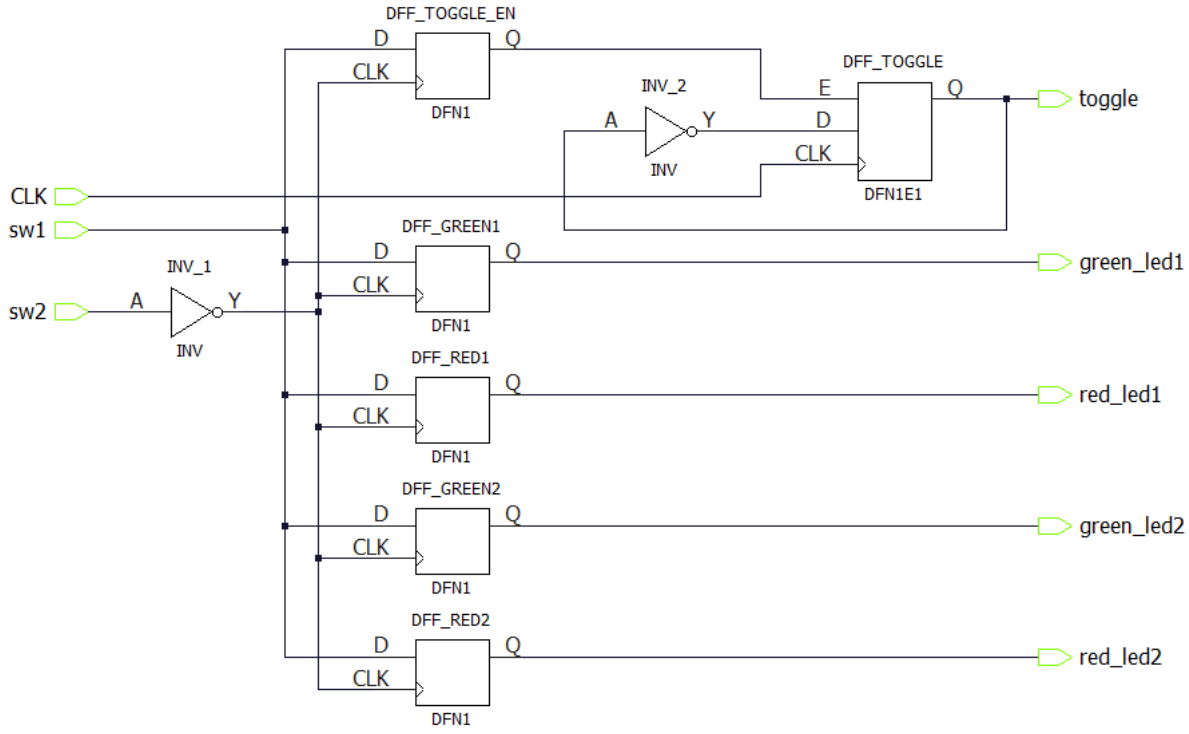

Serdes_Debug_0
REF_CLK_PAD_P
REF_CLK_PAD_N
SYS_RESET_N
PADs_IN
LANE2_RXD_N
LANE2_RXD_P
LANE1_RXD_N
LANE1_RXD_P
LANE0_RXD_N
LANE0_RXD_P
CLK_125
PLL_LOCK
PADs_OUT
LANE2_TXD_N
LANE2_TXD_P
LANE1_TXD_N
LANE1_TXD_P
LANE0_TXD_N
LANE0_TXD_P
Serdes_Debug

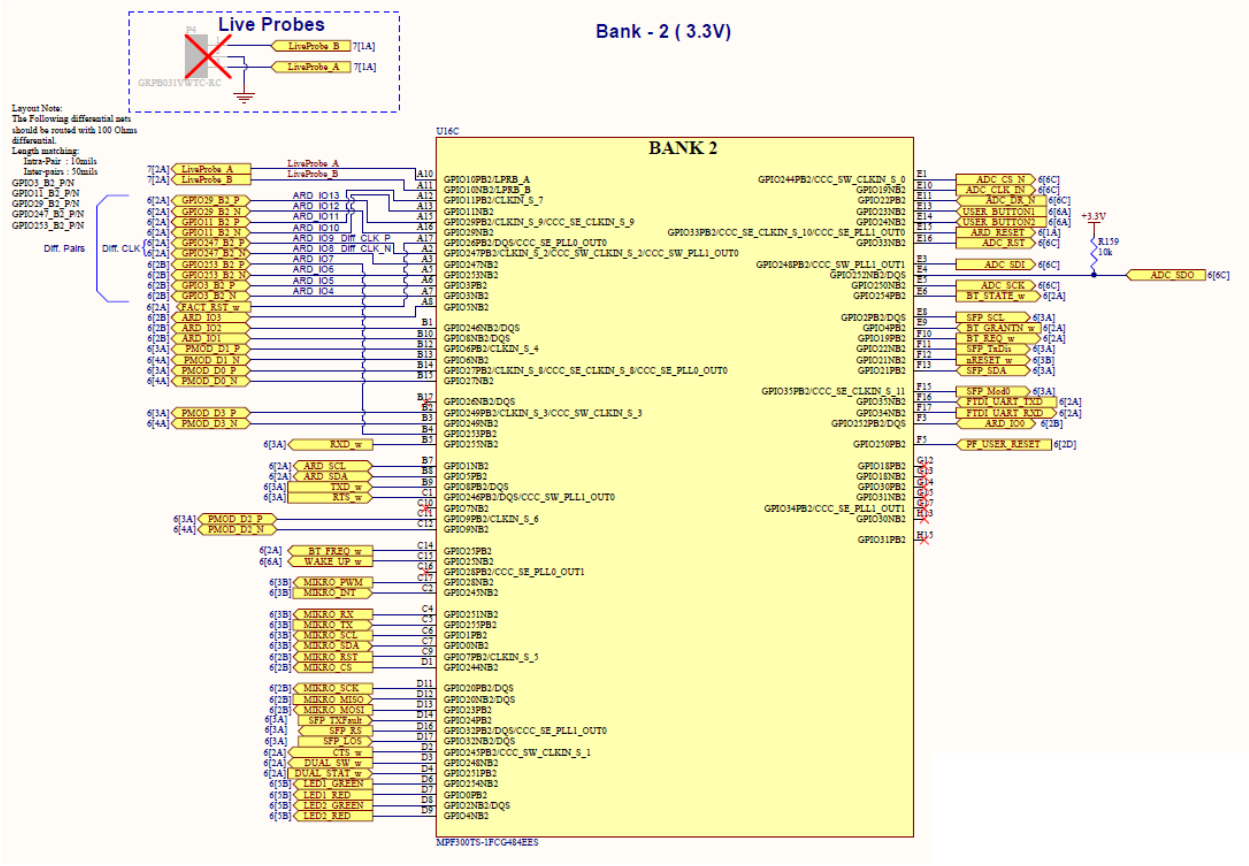

xver_ref_clk_0_0
REF_CLK_PAD_P
REF_CLK_PAD_N
REF_CLK
xver_ref_clk_0
pll_xvier_0_0
REF_CLK
PLL_LOCK
CLK_125
CLKS_TO_XCVR
pll_xvier_0
bert_xvier_chk_0
SYS_RESET_N
CLKS_FROM_TXPLL_0
PADs_IN
LANE2_RXD_N
LANE2_RXD_P
LANE1_RXD_N
LANE1_RXD_P
LANE0_RXD_N
LANE0_RXD_P
LANE0_XCVR
LANE0_CDR_REF_CLK_0
LANE1_XCVR
LANE1_CDR_REF_CLK_0
LANE2_XCVR
LANE2_CDR_REF_CLK_0
PADs_OUT
LANE2_TXD_N
LANE2_TXD_P
LANE1_TXD_N
LANE1_TXD_P
LANE0_TXD_N
LANE0_TXD_P
bert_xvier_chk


Fabric_Debug_0
clk
rst_n
A_WEN
error
match_out
BUSY
uprom_err
Fabric_Debug


Fabric_Debug_DPSRAM_0_DPSRAM_0
count_chk_0
URAM_0_0
match_data_0
count_0
uprom_chk_0
MicorROM_0_0
count
Fabric_Debug_DPSRAM_0_DPSRAM_K
count_chk
URAM_0
match_data
uprom_chk
MicorROM_0


IO_CONTROL_0
green_led1
sw1
red_led1
sw2
green_led2
CLK
red_led2
toggle
IO_CONTROL


DFF_TOGGLE_EN
D
Q
CLK
DFN1
INV_2
A
Y
INV
DFF_TOGGLE
E
Q
D
CLK
DFN1E1
toggle
CLK
sw1
DFF_GREEN1
green_led1
INV_1
sw2
A
Y
INV
DFF_RED1
red_led1
DFF_GREEN2
green_led2
DFF_RED2
red_led2


Live Probes
LiveProbe_B 7[1A]
LiveProbe_A 7[1A]
Bank - 2 (3.3V)
Layout Note:
The Following differential nets
should be routed with 100 Ohms
differantial.
Length matching:
Intra-Pair : 10mils
Inter-pairs : 50mils
Diff. Pairs
Diff. CLK
U16C
BANK 2
MPF300TS-1FCG484EES
+3.3V
R159
10k
ADC_SDO 6[6C]