AC475
Application Note
PolarFire FPGA Dynamic Reconfiguration Interface

Microsemi

a Microchip company

Microsemi 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
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

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

51900475. 6.0 2/21

Contents

1 Revision History . . . 1
1.1 Revision 6.0 . . . 1
1.2 Revision 5.0 . . . 1
1.3 Revision 4.0 . . . 1
1.4 Revision 3.0 . . . 1
1.5 Revision 2.0 . . . 1
1.6 Revision 1.0 . . . 1

2 PolarFire FPGA Dynamic Reconfiguration Interface . . . 2
2.1 Design Requirements . . . 2
2.2 Prerequisites . . . 3
2.3 Demo Design . . . 3
2.3.1 Design Implementation . . . 4
2.3.2 IP Configuration . . . 4
2.4 Port Description . . . 9
2.5 Simulating the Design . . . 10
2.6 Simulation Flow . . . 11
2.7 Clocking Structure . . . 13
2.8 Reset Structure . . . 13

3 Libero Design Flow . . . 14
3.1 Synthesize . . . 15
3.2 Resource Utilization . . . 15
3.3 Place and Route . . . 15
3.4 Verify Timing . . . 16
3.5 Design and Memory Initialization . . . 16
3.6 Generate Bitstream . . . 16
3.7 Run PROGRAM Action . . . 17

4 Running the Demo . . . 19
4.1 Power Monitor . . . 21

5 Appendix 1: Programming the Device Using FlashPro Express . . . 25

6 Appendix 2: Running the TCL Script . . . 27

Figures

Figure 1 DRI Block Diagram . . . 3
Figure 2 Top Level Libero Design . . . 4
Figure 3 CoreABC Configuration . . . 5
Figure 4 CoreABC Program . . . 6
Figure 5 CoreABC Program-Continued . . . 7
Figure 6 PF_DRI Configuration . . . 8
Figure 7 PF_DRI CCC Configuration . . . 8
Figure 8 PolarFire Transceiver Reconfiguration GUI . . . 9
Figure 9 Simulation Runtime . . . 10
Figure 10 Simulating the Design . . . 10
Figure 11 Top Level Simulation . . . 11
Figure 12 2.5G Transceiver Simulation . . . 12
Figure 13 CCC Output Frequency Simulation . . . 12
Figure 14 Clocking Structure . . . 13
Figure 15 Reset Structure . . . 13
Figure 16 Libero Design Flow Options . . . 14
Figure 17 I/O Editor-Transceiver View . . . 15
Figure 18 Design Flow . . . 16
Figure 19 Generate Design Initialization Data . . . 16
Figure 20 Board Setup . . . 18
Figure 21 COM Port Setup . . . 21
Figure 22 Power Usage at 1.25G . . . 22
Figure 23 Power Usage at 2.5G . . . 23
Figure 24 Power Usage at 5G . . . 24
Figure 25 FlashPro Express Job Project . . . 25
Figure 26 New Job Project from FlashPro Express Job . . . 26
Figure 27 Programming the Device . . . 26
Figure 28 FlashPro Express—RUN PASSED . . . 26

Tables

Table 1 Design Requirements 2
Table 2 Port Description 9
Table 3 Resource Utilization 15
Table 4 Jumper Settings for PolarFire Device Programming 17
Table 5 Output Status-LED 20
Table 6 Output Status-Probe Test Points 21
Table 7 Transceiver Power 24

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 6.0

Added Appendix 2: Running the TCL Script, page 27.

1.2 Revision 5.0

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

- Updated the document for Libero SoC v12.2.
- Removed the references to Libero version numbers.

1.3 Revision 4.0

Updated the document for Libero SoC v12.1 release.

1.4 Revision 3.0

Updated the document for Libero SoC v12.0 release.

1.5 Revision 2.0

Updated the document for Libero SoC PolarFire v2.3 release.

1.6 Revision 1.0

The first publication of this document.

2 PolarFire FPGA Dynamic Reconfiguration Interface

The PolarFire® FPGA family includes multiple embedded low-power and performance-optimized transceivers. Each transceiver has both the Physical Medium Attachment (PMA), Physical Coding Sub-layer (PCS) logic, and interfaces to the FPGA fabric.

The transceiver has a multi-lane architecture with each lane natively supporting serial data transmission rates from 250 Mbps to 12.7 Gbps.

This document describes how to perform the dynamic reconfiguration of Clock Conditioning Circuit (CCC) and transceivers in a PolarFire FPGA by changing the output clock frequency in a glitch-free way.

Each CCC and transceiver has a Dynamic Reconfiguration Interface (DRI), which can be enabled to configure its parameters without reprogramming the device. The volatile configuration registers control CCC and transceiver reconfiguration that are loaded with values from the flash configuration bits at power-up. An APB3 bus master must be interfaced to the CCC and transceiver using a DRI macro for dynamic configuration. The APB3 bus master is required to dynamically modify the CCC and transceiver configuration register values as per the design needs.

Any of the configuration registers can be accessed dynamically using APB3 interface. This document does not discuss all of the Fabric CCC and transceiver registers that can be dynamically configured. It describes how to dynamically change output clock frequency in CCC and how to dynamically change the transceivers data rate.

2.1 Design Requirements

The following table lists the resources required to run the demo.

Table 1 • **Design Requirements**

Requirement	Version
Operating System	64-bit Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit (MPF300T-EVAL-KIT)	Rev D or later
2 SMA-to-SMA cables with 10 Gbps support (not provided with the kit)	
Oscilloscope	
Software	
FlashPro Express	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
Libero® System-on-Chip (SoC)	

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you begin:

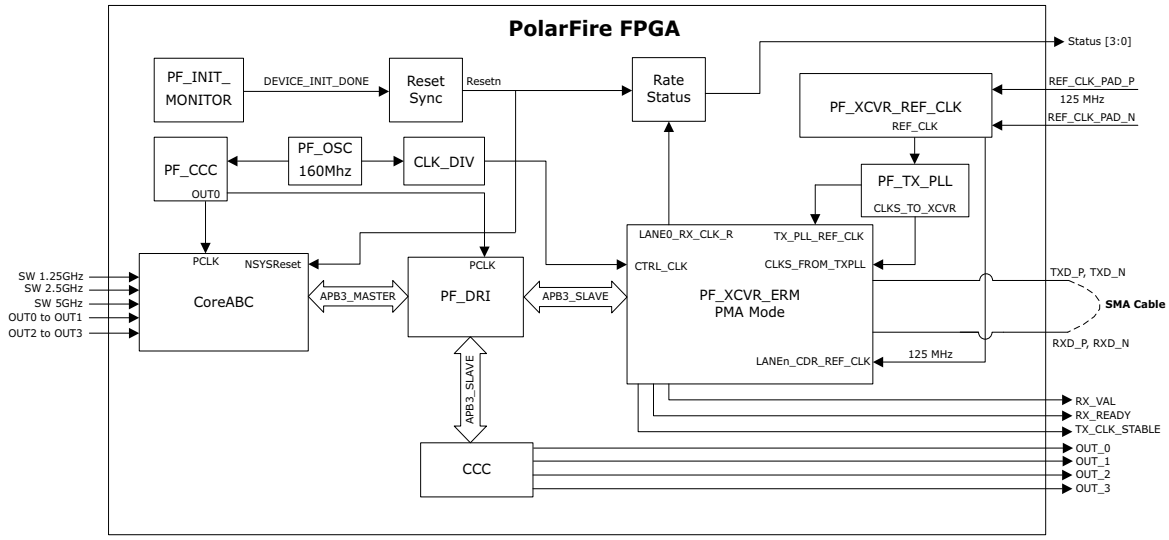
1. Download the demo design files from the following location:
   http://soc.microsemi.com/download/rsc/?f=mpf_ac475_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads
3. Download the transceiver register map from the following location:
   https://www.microsemi.com/document-portal/doc_download/1243471-polarfire-device-register-map

2.3 Demo Design

The following steps describe the data flow in the demo design:

1. OSC_160 MHz provides 160 MHz clock source to PF_CCC_50 block.
2. The PF_CCC_50 block provides 50 MHz clock for the fabric.
3. The 50 MHz fabric clock drives Reset-Synchronizer, CoreABC, and PF_DRI modules.
4. The transceiver (PF_XVCR) block instantiates the transceiver in 8b10b mode. This block receives clock from the REF_CLK signal of PF_XCVR_REF_CLK_0. The PF_TX_PLL_0 block also derives its reference clock from REF_CLK of PF_XCVR_REF_CLK_0.
5. The TX and RX lanes of the transceiver are externally looped back using SMA cables.
6. After CoreABC instruction is executed, the RX_VAL and RX_READY outputs should be monitored for link status.
7. The Rate_Status block indicates the rate at which the transceiver and the Dynamic CCC are configured.

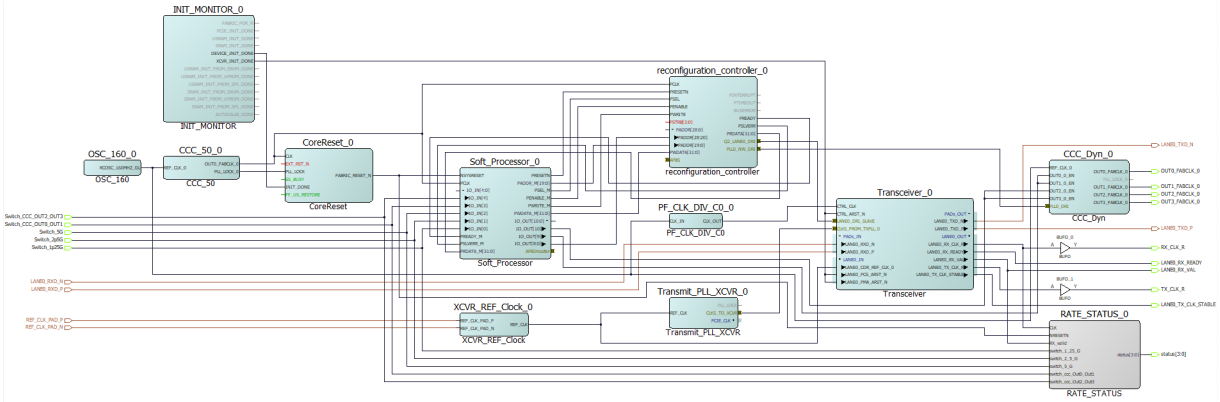
Figure 1 • **DRI Block Diagram**



2.3.1 Design Implementation

The following figure shows the top-level Libero design of the PolarFire Transceiver Dynamic Reconfiguration Interface design.

Figure 2 • **Top Level Libero Design**



Note: For the latest SmartDesign, see the design files provided.

2.3.2 IP Configuration

The following sections describe the IP cores used in the design and their configurations.

Note: The IP cores which are not described in the following section keep the default configuration.

2.3.2.1 PF_CCC_0 Configuration

The PF_CCC block provides a clock for CoreABC and Dynamic Configuration Interface. The input for the CCC is from 160 MHz on-chip RC oscillator. The output clock of CCC is configured at 50 MHz.

2.3.2.2 Oscillator Configuration

The RC Oscillator runs on two configurations; 160 MHz RC Oscillator and 2 MHz RC Oscillator. The Enable RCOSC_160 MHz to Global configuration is selected.

2.3.2.3 PF_CCC_Dyn_0

This CCC is dynamically reconfigured using DRI interface. The input for the CCC is from 160 MHz on-chip oscillator. Four Output clocks are configured at 100 MHz, 50 MHz, 100 MHz, and 50 MHz and their respective Output enables are used.

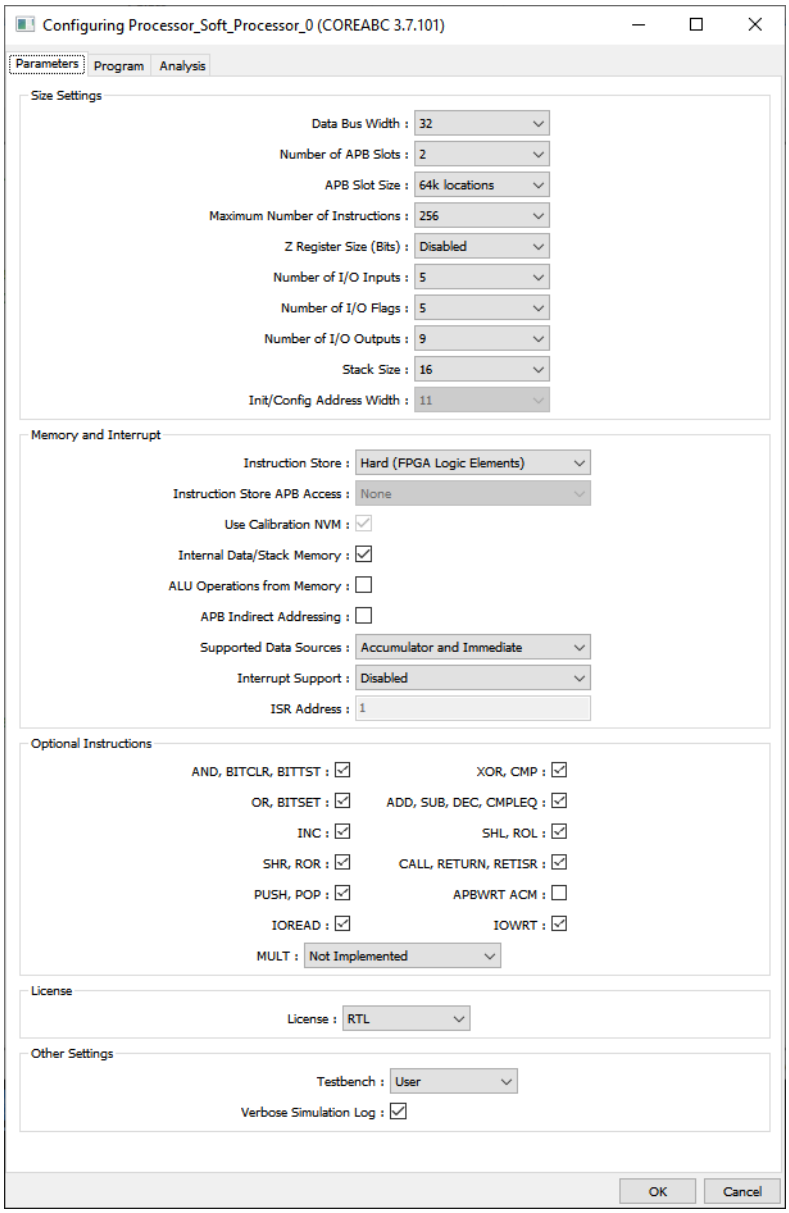
The Output enables are used to switch the Output clock frequencies in glitch free way.

2.3.2.4 CoreABC

The CoreABC is a programmable soft-controller targeted for implementing Advanced Microcontroller Bus Architecture (AMBA) based designs.

CoreABC in this design is connected to DRI as an APB3 master. The APB3 slaves of DRI are connected to the transceiver and CCC. CoreABC initiates the instruction sequence and dynamically performs Read/Write operation on Transceiver and CCC registers. The number of APB slots, APB slot size, and maximum number of instructions are configured depending on the number of peripherals and address size used. The CoreABC uses the 20-bit address bus and the DRI uses 29-bit address bus. The remaining 9-bit address bus can be connected using IOs from CoreABC as shown in Figure 2.The following figure shows the parameter configuration of CoreABC interface.

Figure 3 • **CoreABC Configuration**



2.3.2.4.1 CoreABC Program

The following figure illustrates the register settings required for performing DRI instructions. The DIP switches are connected to the inputs of CoreABC and are used to control the switching between the rates.

Figure 4 • **CoreABC Program**

```
$Rate_select

    IOWRT 0x600
    IOREAD
    NOP
    NOP
    NOP
    JUMP IF INPUT0 $Rate_change_1_25_G
    NOP
    NOP
    JUMP IF INPUT1 $Rate_change_2_5_G
    NOP
    NOP
    JUMP IF INPUT2 $Rate_change_5_G
    NOP
    NOP
    JUMP IF INPUT3 $Clock_change_out0_out1
    NOP
    NOP
    JUMP IF INPUT4 $Clock_change_out2_out3
JUMP $Rate_select

$Rate_change_1_25_G

IOWRT 0x611
//SER_RSTPD
APBWRT DAT 0 0x1078 0x00000003
//DES_RSTPD
APBWRT DAT 0 0x104C 0x00000032

//TXPLL_DIV_1
    APBWRT DAT 1 0x0010 0x0140000
//TXPLL_DIV_2
    APBWRT DAT 1 0x0014 0x1000000
//SER_CLK_CTRL
    APBWRT DAT 0 0x1074 0x0000073
//DES_RXPLL_DIV
    APBWRT DAT 0 0x1040 0x000420A

JUMP $Reset_release

$Rate_change_2_5_G

IOWRT 0x611
//SER_RSTPD
APBWRT DAT 0 0x1078 0x00000003
//DES_RSTPD
APBWRT DAT 0 0x104C 0x00000032

//TXPLL_DIV_1
    APBWRT DAT 1 0x0010 0x0140000
//TXPLL_DIV_2
    APBWRT DAT 1 0x0014 0x1000000
//SER_CLK_CTRL
    APBWRT DAT 0 0x1074 0x0000071
//DES_RXPLL_DIV
    APBWRT DAT 0 0x1040 0x0002214

JUMP $Reset_release
```

Read Input DIP Swtiches

XCVR Configuration Setting for 1.25 G

XCVR Configuration Setting for 2.5 G

Figure 5 • **CoreABC Program-Continued**

```
$Rate_change_5_G

IOWRT 0x611
//SER_RSTPD
APBWRT DAT 0 0x1078 0x00000003
//DES_RSTPD
APBWRT DAT 0 0x104C 0x00000032

//TXPLL_DIV_1
APBWRT DAT 1 0x0010 0x0140000
//TXPLL_DIV_2
APBWRT DAT 1 0x0014 0x1000000
//SER_CLK_CTRL
APBWRT DAT 0 0x1074 0x0000070
//DES_RXPLL_DIV
APBWRT DAT 0 0x1040 0x0000228

JUMP $Reset_release
```

XCVR Configuration Setting for 5 G

```
$Reset_release

//SER_RSTPD
APBWRT DAT 0 0x1078 0x00000001
//DES_RSTPD
APBWRT DAT 0 0x104C 0x00000010
IOWRT 0x601
//LRST
APBWRT DAT 0 0x1068 0x00000D0D
NOP
//LRST
APBWRT DAT 0 0x1068 0x00000404


JUMP $Rate_Change_Done
```

Release PMA and PCS Reset

```
$Clock_change_out0_out1

IOWRT 0x481
//PLL_OUT0_AND_OUT1_DIVIDER
APBWRT DAT 0 0x0010 0x08001800
IOWRT 0x681

JUMP $Rate_Change_Done

$Clock_change_out2_out3

IOWRT 0x281
//PLL_OUT2_AND_OUT3_DIVIDER
APBWRT DAT 0 0x0014 0x08001800
IOWRT 0x681

JUMP $Rate_Change_Done
```

CCC Configuration for OUT0, OUT1, OUT2, and OUT3

```
$Rate_Change_Done

NOP
NOP

JUMP $Rate_select

HALT
```

Jump Back to I/O Read

2.3.2.4.2 Instruction Flow for Transceiver Rate Change

The transceiver reconfiguration occurs in the following sequence:

1. Assert Reset for the serializer
2. Assert Reset for the deserializer
3. Change TX_PLL_DIV_1 register
4. Change TX_PLL_DIV_2
5. Change SER_CLK_CNTRL
6. Change DES_RXPLL_DIV
7. De-Assert reset for the serializer
8. De-assert reset for the deserializer
9. Assert and de-assert the soft PCS reset

Note: For more information about transceiver configuration register, refer to *PolarFire Device Register Map*.

2.3.2.4.3 Instruction for CCC Frequency Change

In this demo design, only the configuration of CCC post dividers is changed dynamically. To change the configuration of CCC post dividers, the output clocks must be stopped using corresponding OUT#_EN signals. The CoreABC instructions are used to perform the following steps:

1. Change PLL_DIV_0_1
2. Change PLL_DIV_2_3

2.3.2.5 Dynamic Reconfiguration Interface

DRI performs the run time configuration of transceiver PMA/PCS, PCIe, CCC, and Transmit PLLs after initialization. DRI ports are routed through hardwired connections.

In this demo design, DRI is used to perform the Run-time calibration of transceiver and CCC. Q2_LANE0 is enabled to expose slave to the Transceiver interface and PLL0_NW is enabled to expose slave to the CCC interface as shown in the following figure.

Figure 6 • **PF_DRI Configuration**

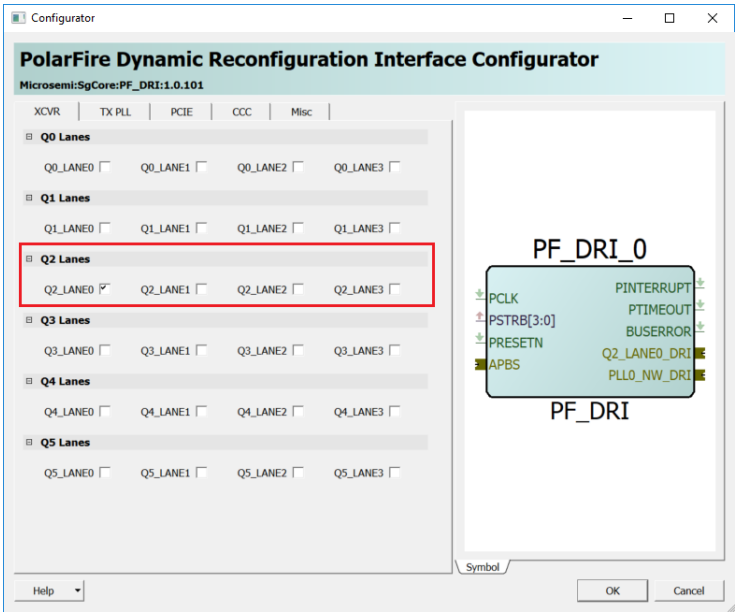
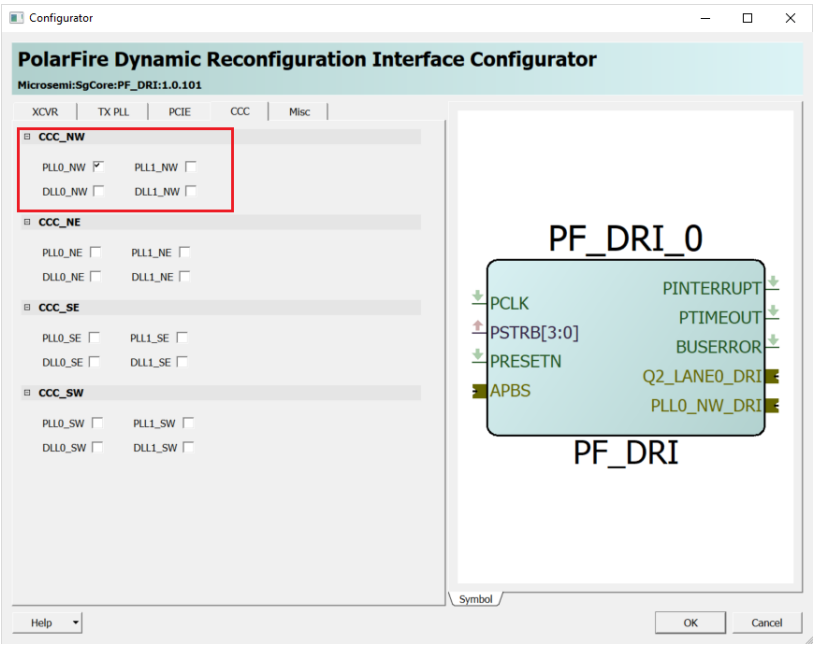


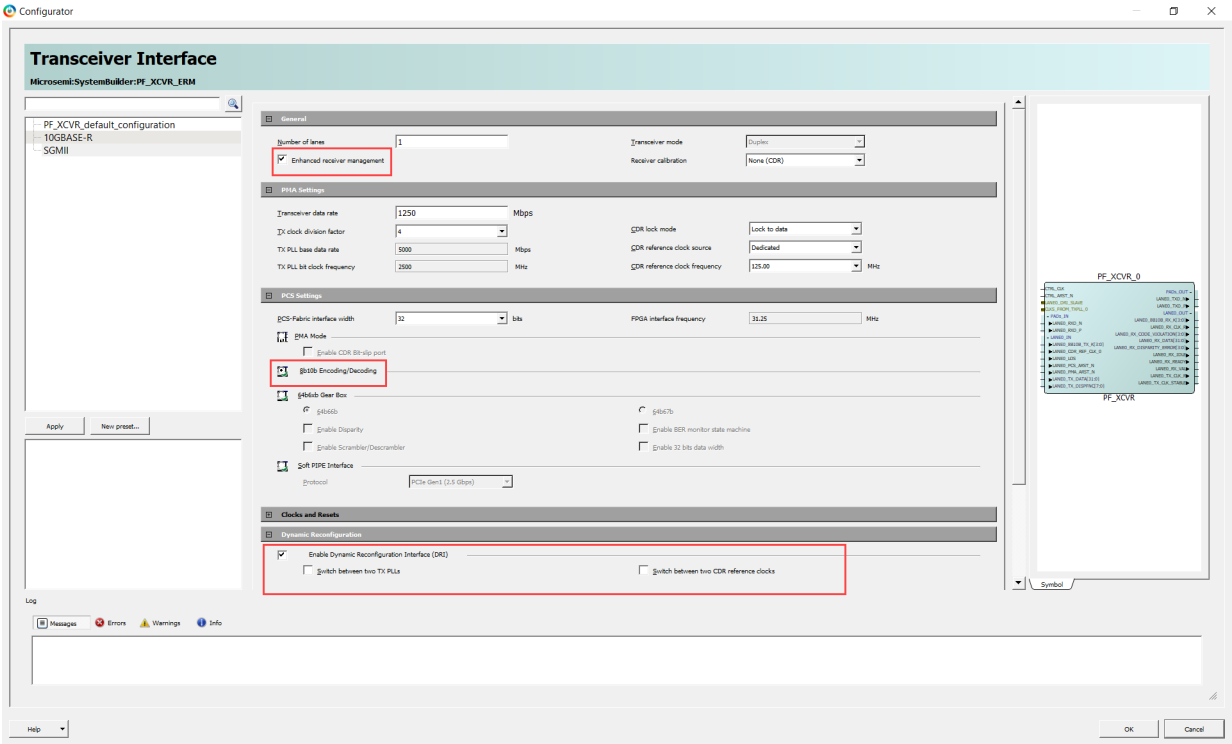
Figure 7 • **PF_DRI CCC Configuration**



2.3.2.6 Transceiver Interface Reconfiguration

The PolarFire transceiver interface configurator is set to 1.25 Gbps, 32-bit PCS-Fabric interface width. The following figure shows the PolarFire Transceiver Interface configurator settings and how to enable DRI.

Figure 8 • **PolarFire Transceiver Reconfiguration GUI**



2.4 Port Description

The following table lists the key signals for this design.

Table 2 • **Port Description**

Signal	Direction	Description
REF_CLK_PAD_P and REF_CLK_PAD_N		Differential reference clock is generated from the on-board 125 MHz oscillator
LANE0_RXD_N	Input	Transceiver receiver differential input
LANE0_RXD_P	Input	Transceiver receiver differential input
LANE0_TXD_N	Output	Transceiver transmitter differential output
LANE0_TXD_P	Output	Transceiver transmitter differential output
LANE0_RX_READY	Output	Asserted when the CDR is phase-locked to the incoming data transitions and the de-serializer is powered-up.
LANE0_TX_CLK_STABLE	Output	Transmit transceiver/PCS lane ready flag. This flag is asserted when the transmit PLL is locked to the reference clock.
LANE0_RX_VAL	Output	RX_VAL indicates that the XCVR data path is initialized.
RX_CLK_R	Output	Global or regional receive clock to the fabric
TX_CLK_R	Output	Global or regional transmit clock to the fabric
OUT0_FABCLK_0	Output	Dynamic CCC OUT0 Fabric Clock

Table 2 • **Port Description *(continued)***

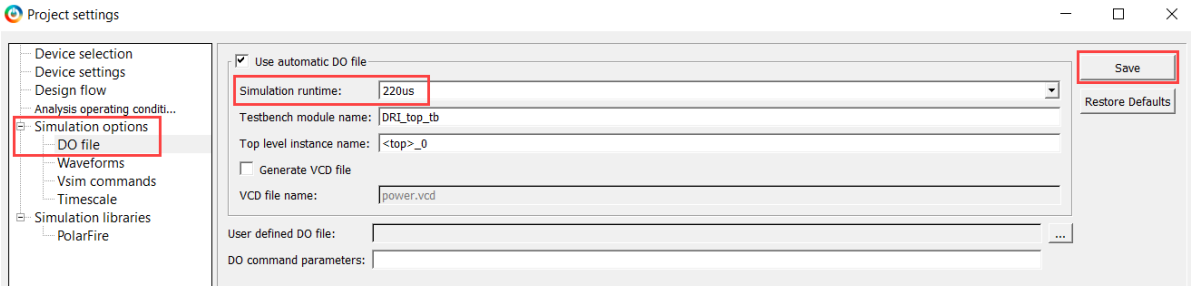
Signal	Direction	Description
OUT1_FABCLK_0	Output	Dynamic CCC OUT1 Fabric Clock
OUT2_FABCLK_0	Output	Dynamic CCC OUT2 Fabric Clock
OUT3_FABCLK_0	Output	Dynamic CCC OUT3 Fabric Clock

2.5 Simulating the Design

For simulation, perform the following the steps:

1. Start Libero SoC.
2. In the Project toolbar, click **Open Project**.
3. Navigate to the Libero_Design folder, select `DRI_XCVR_CCC.prjx`, and click **Open**.
4. The PolarFire transceiver project opens in Libero SoC.
5. Click **Project** > **Project settings** > **Simulation options** > **DO file**. Set the **Simulation runtime** to 220us and click **Save** as shown in the following figure.

Figure 9 • **Simulation Runtime**



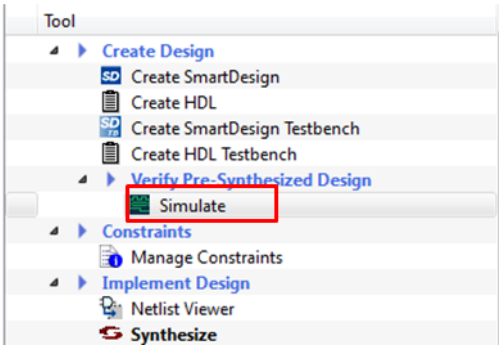
6. Navigate to the **Design Hierarchy** tab and double-click the top level component.

The SmartDesign page opens on the right pane and displays the high-level design. Now, you can view all of the design blocks and IP cores instantiated in the design.

Note: If not already installed, download the PF_XCVR_REF_CLK, PF_TX_PLL, PF_CCC, PF_XCVR, CoreABC, and PF_DRI cores under **Libero SoC > Catalog**.

In the **Design Flow** tab, under **Verify Pre-Synthesized Design,** double-click **Simulate** as shown in the following figure. The ModelSim tool takes around ten minutes to complete the simulation.

Figure 10 • **Simulating the Design**



2.6 Simulation Flow

The following steps describe the simulation flow:

1. At the start, the transceiver is kept at reset.
2. Once the device initialization is done, the transceiver is brought out of reset.
3. Constant K28.5 character is provided to transceiver.
4. The transmitter lanes are connected to receiver lanes internally in the testbench stimulus.
5. The transceiver is reconfigured at 1.25G by providing a pulse on Switch_1_25_G signal.
6. The transceiver is reconfigured at 2.5G by providing a pulse on Switch_2_5_G signal. Refer to Figure 12, page 12.
7. The transceiver is reconfigured at 5G by providing a pulse on Switch_5_G signal.
8. The OUT0 and OUT1 of Dynamic CCC are reconfigured by providing a pulse on Switch_CCC_OUT0_OUT1. Refer to Figure 13, page 12.
9. The OUT2 and OUT3 of Dynamic CCC are reconfigured by providing a pulse on Switch_CCC_OUT2_OUT3. Refer to Figure 13, page 12.
10. The status signal denotes which reconfiguration is used currently.

Note: When the pulse is provided on Switch_1_25_G, Switch_2_5_G, and Switch_5_G signals, the RX_valid is de-asserted and will be asserted once the reconfiguration is completed.

Figure 11 • **Top Level Simulation**

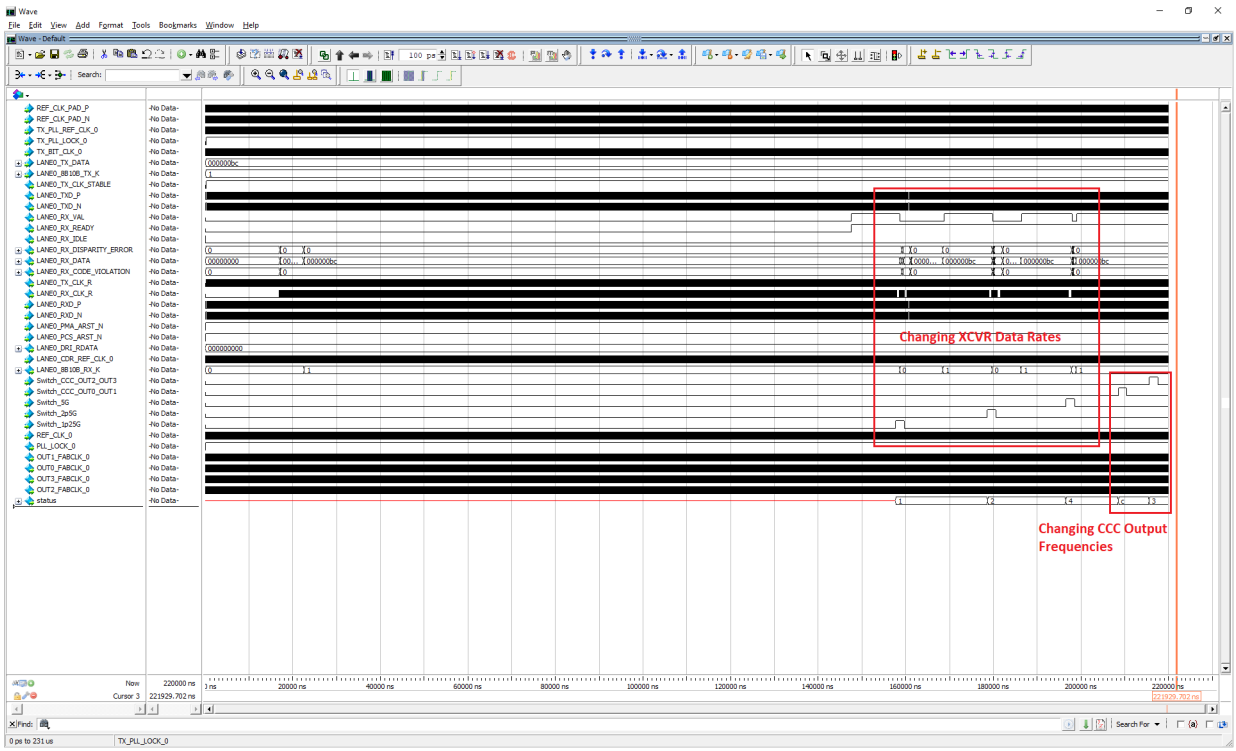


Figure 12 • **2.5G Transceiver Simulation**

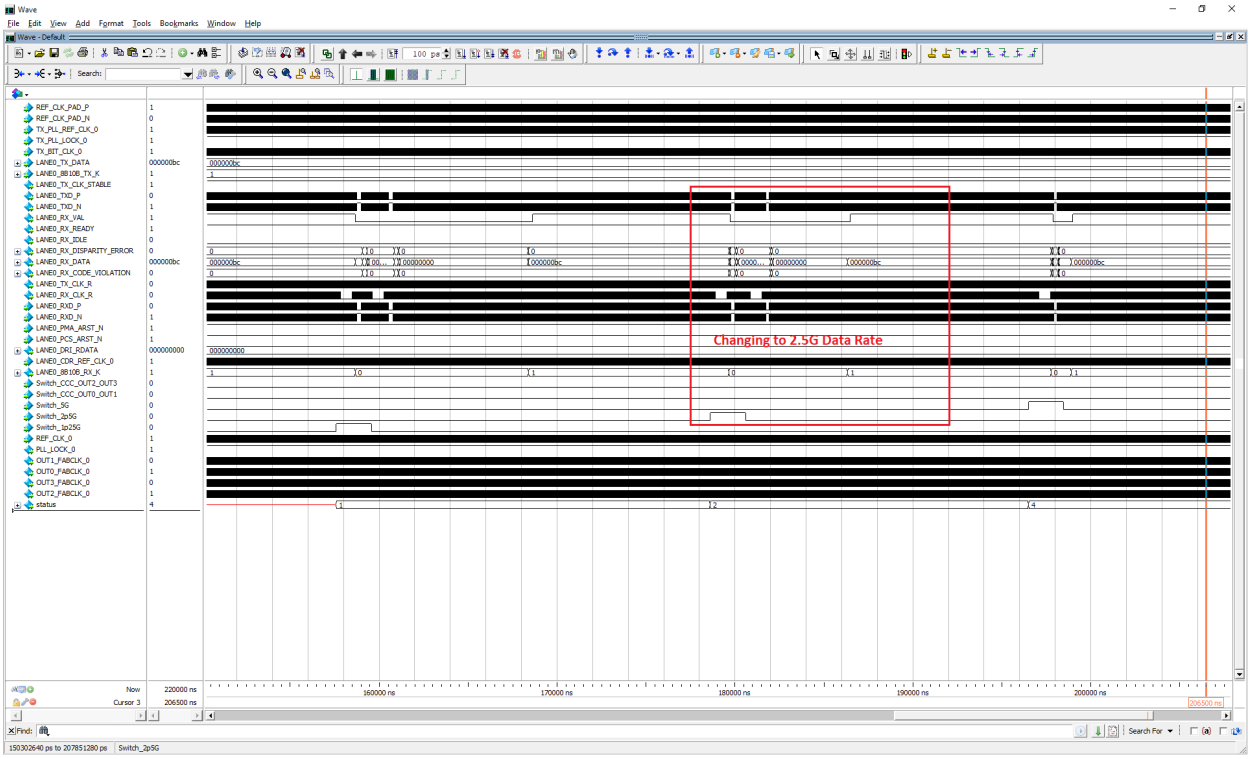
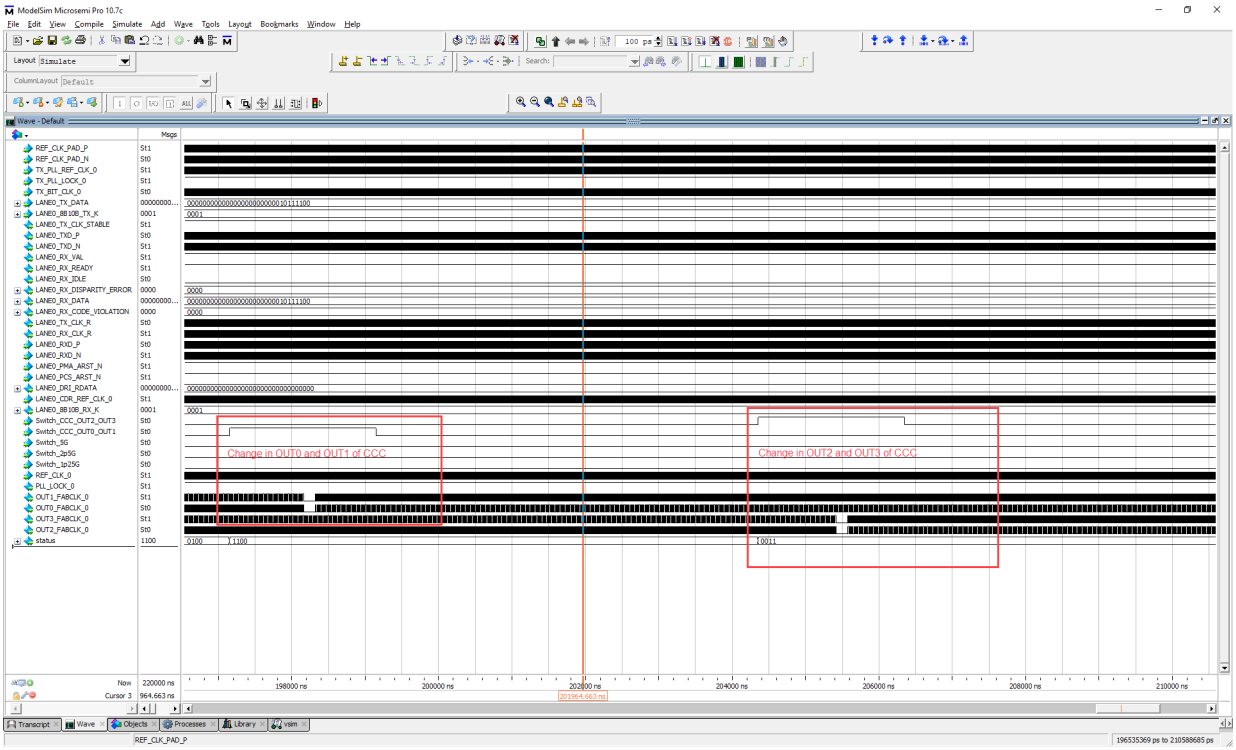


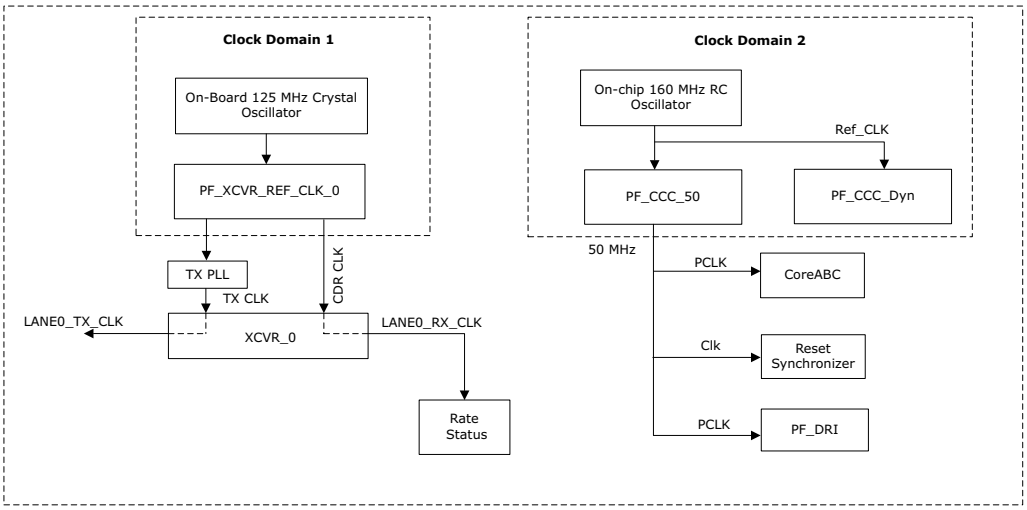
Figure 13 • **CCC Output Frequency Simulation**



2.7 Clocking Structure

In this demo design, there are two clock-domains. The on-board 125 MHz crystal oscillator drives the XCVR reference clock. The XCVR REFCLK source the transceivers and global clock network. The on-chip 160 MHz RC oscillator drives the CoreABC and PF_DRI block. The following figure shows the clocking structure of the design.

Figure 14 • **Clocking Structure**



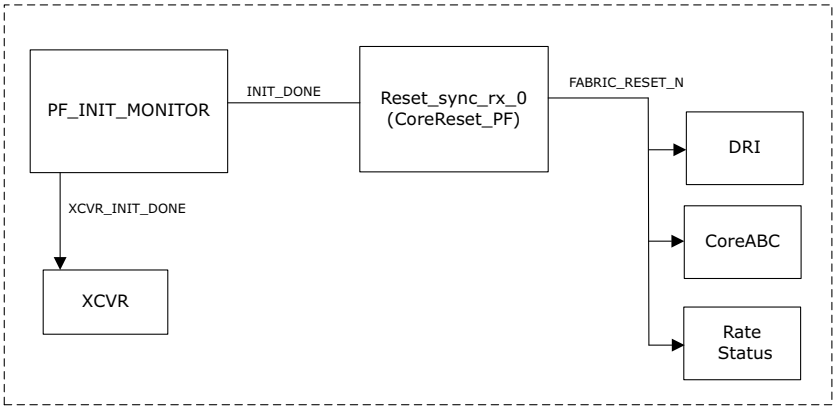
2.8 Reset Structure

When PLL_LOCK output from CCC_50 block and DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted, the Reset_Synchronizer_0 (CoreReset_PF) module releases active low reset of Soft_Processor_0(CoreABC), reconfiguration_controller_0 (PF_DRI), and the Rate_status_0 modules.

DEVICE_INIT_DONE signal is asserted when the device initialization is complete. For more information about device initialization, refer to *UG0725: PolarFire FPGA Device Power-Up and Resets User Guide*. For more information on CoreReset_PF IP core, refer to *CoreReset_PF handbook* from the Libero catalog.

The following figure shows the reset structure of the design.

Figure 15 • **Reset Structure**



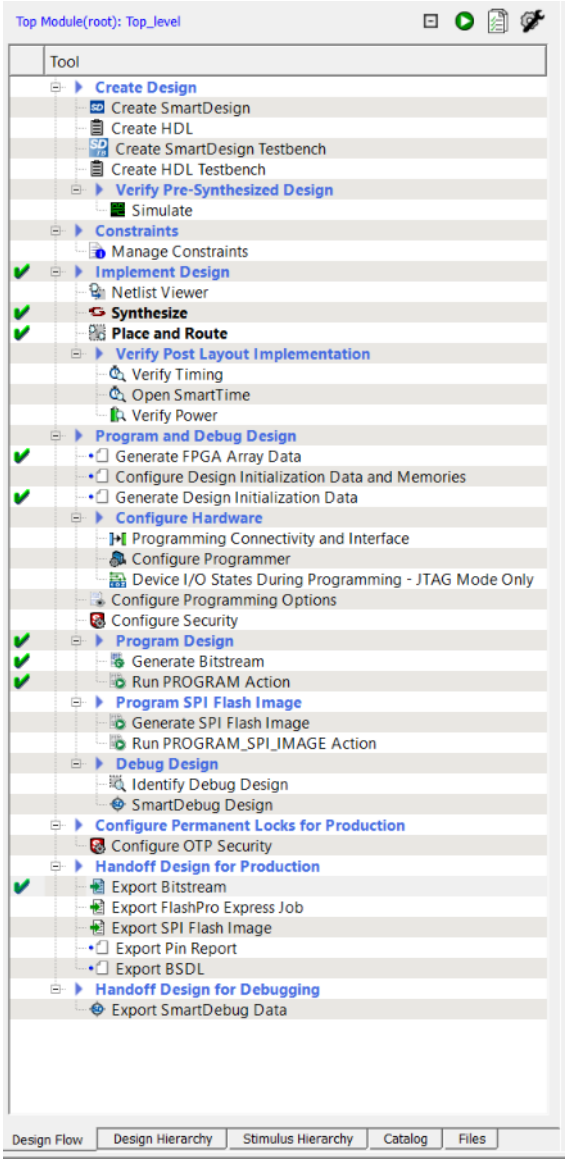
3 Libero Design Flow

The Libero design flow involves running the following processes in the Libero SoC:

- Synthesize, page 15
- Resource Utilization, page 15
- Place and Route, page 15
- Verify Timing, page 16
- Design and Memory Initialization, page 16
- Generate Bitstream, page 16
- Run PROGRAM Action, page 17

The following figure shows the options for the preceding processes in the **Design Flow** tab.

Figure 16 • **Libero Design Flow Options**



3.1 Synthesize

To synthesize the design, perform the following steps:

1. Double-click **Synthesize** from the **Design Flow** tab.
   When the synthesis is successful, a green tick mark appears as shown in Figure 16, page 14.
2. Right-click **Synthesize** and select **View Report** to view the synthesis report and log files in the **Reports** tab.

3.2 Resource Utilization

The following table lists the resource utilization of the DRI design after synthesis. These values may vary slightly for different Libero runs, settings, and seed values.

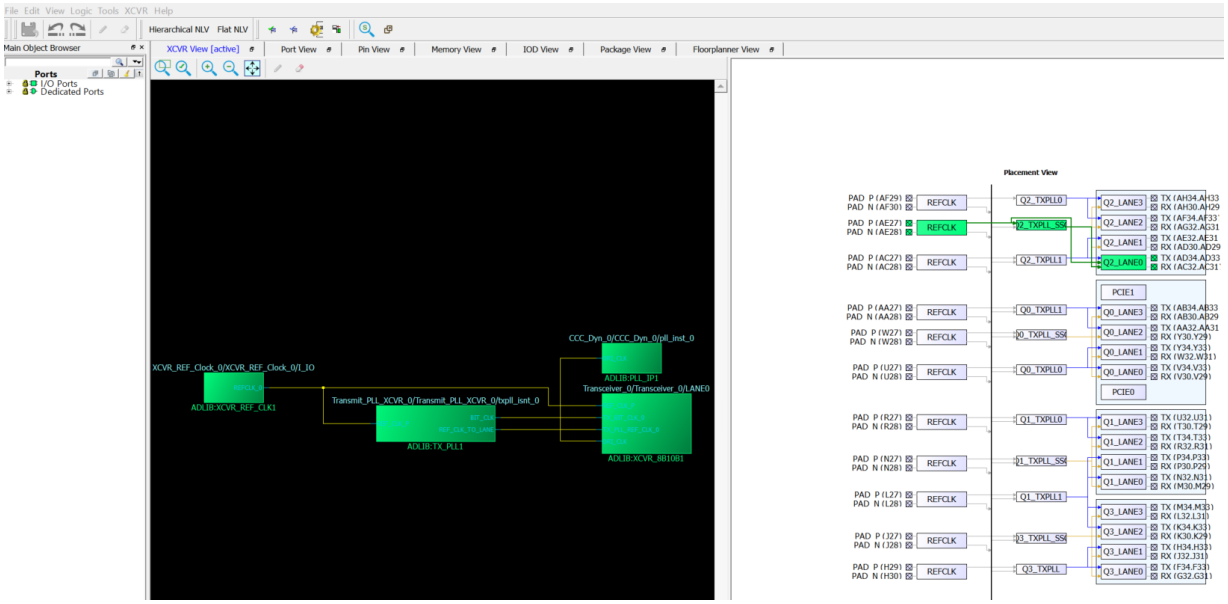
Table 3 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	959	299544	0.32
DFF	468	299544	0.16
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

3.3 Place and Route

For DRI design, the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. For SMA Transceiver loopback Lane0 of Quad2 is used, as shown in the following figure.

Figure 17 • **I/O Editor-Transceiver View**

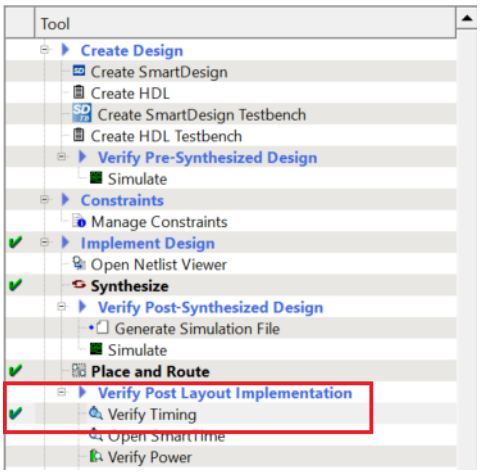


3.4 Verify Timing

To verify timing, perform the following steps:

1. Double-click **Verify Timing** from the **Design Flow** tab.
   When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 16, page 14.
2. Right-click **Verify Timing** and select **View Report** to view the verify timing report and log files in the **Reports** tab.

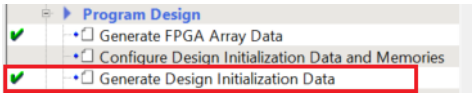
Figure 18 • **Design Flow**



3.5 Design and Memory Initialization

This option is used to create the XCVR initialization client, which is used in the demo design. When the PolarFire device powers up, the transceiver block is initialized by the initialization client generated during the Configure Design Initialization Data and Memories stage in the design flow. For more information about device power-up, refer to *UG0725: PolarFire FPGA Device Power-up and Resets User Guide*.

Figure 19 • **Generate Design Initialization Data**



3.6 Generate Bitstream

To generate the bitstream, perform the following steps:

1. Right-click **Generate Bitstream** and select Configure Options... to select the bitstream components—Custom security, Fabric, and sNVM.
2. Double-click **Generate Bitstream** from the **Design Flow** tab. When the bitstream is successfully generated, a green tick mark appears as shown in Figure 16, page 14

Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.

3.7 Run PROGRAM Action

After generating the bitstream, the PolarFire device must be programmed with the system services design.

To program the PolarFire device, perform the following steps:

1. Ensure that the following jumper settings are set on the board.

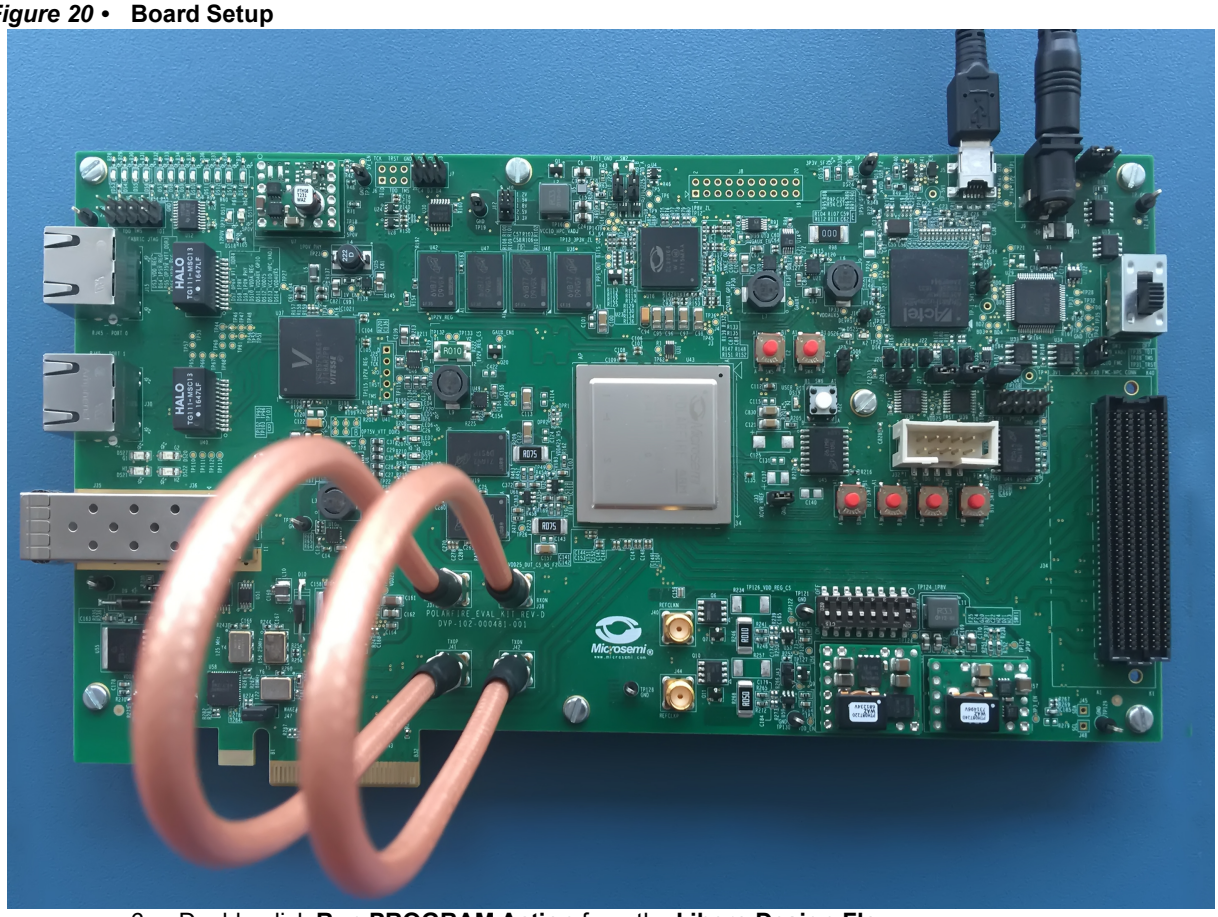
Table 4 • **Jumper Settings for PolarFire Device Programming**

Jumper	Description
J18, J19, J20, J21, and J22	Short pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Short pin 2 and 3 for programming through the onboard FlashPro5
J26	Short pin 1 and 2 for programming through the FTDI SPI
J27	Short pin 1 and 2 for programming through the FTDI SPI
J4	Short pin 1 and 2 for manual power switching using SW3
J12	Short pin 3 and 4 for 2.5 V
J46	• Short pin 1 and 2 for routing 125 MHz differential clock oscillator output to the line side • Open pin 1 and 2 for routing 122.88 MHz differential clock oscillator output to the line side

2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the host PC to the **J5** (FTDI port) on the board.
4. Power on the board using the **SW3** slide switch.
5. Connect TXN to RXN and TXP to RXP using the 2 SMA to SMA cables as shown in the following figure.

The following figure shows the PolarFire Evaluation kit board setup. For information about PolarFire Evaluation kit, refer to *UG0747: PolarFire FPGA Evaluation Kit User Guide*.

Figure 20 • **Board Setup**



6. Double-click **Run PROGRAM Action** from the **Libero Design Flow**.

The device is successfully programmed and the onboard LEDs 4, 5, 6, and 7 glow. A green tick mark appears next to **Run PROGRAM Action** as shown in Figure 16, page 14.

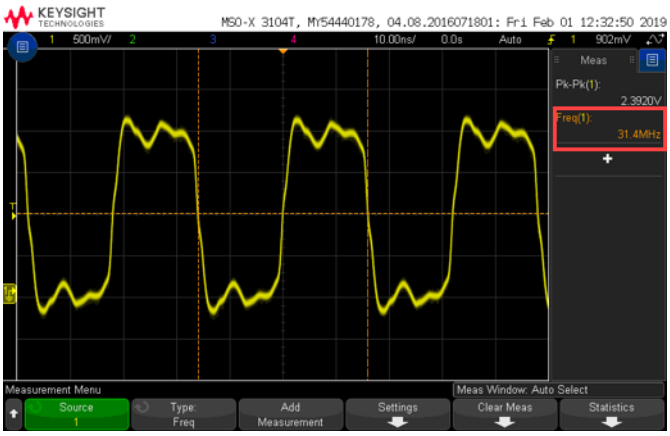
4 Running the Demo

This section describes how to run the DRI demo, check the result on board, and observe the frequency on oscilloscope.

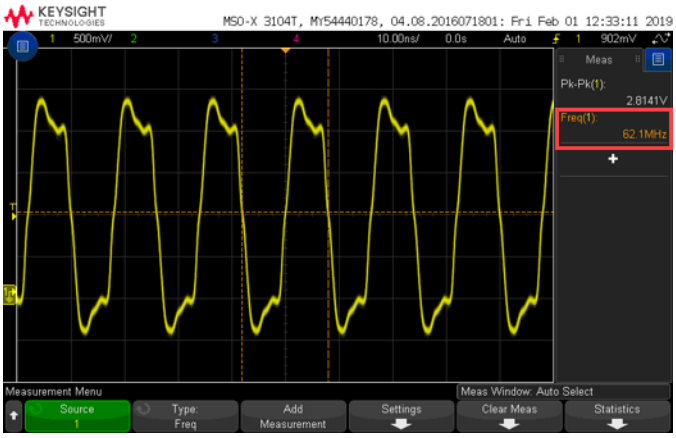
To run the demo, perform the following steps:

1. Setup the PolarFire Evaluation board as explained in Run PROGRAM Action, page 17. This will program the evaluation kit with DRI design and make sure all DIP switches of SW11 are ON. Execute the following tests:

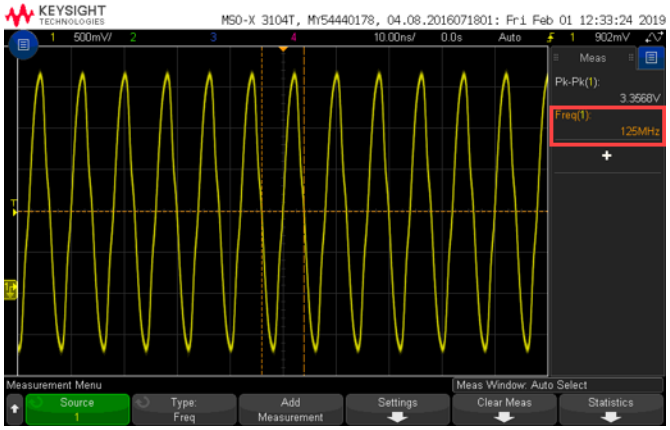
   TEST 1: Switch OFF **SW11 DIP1** and change it back to ON to configure the transceiver in 1.25G at 31.25 MHz of TX and RX Clock frequency.



   TEST 2: Switch OFF **SW11 DIP2** and change it back to ON to configure the transceiver in 2.5G mode at 62.5 MHz of TX and RX Clock frequency.



TEST 3: Switch OFF **SW11 DIP3** and change it back to ON to configure the transceiver in 5G mode at 125 MHz of TX and RX Clock frequency.



TEST 4: Switch OFF **DIP5** and change it back to ON to configure OUT0 and OUT1 frequencies of Dynamic CCC.

TEST 5: Switch **OFF** DIP6 and change it back to ON to configure OUT2 and OUT3 frequencies of Dynamic CCC.

Note: The DIP switches are active-low on the Evaluation Kit. If more than one DIP switch is high, DIP1 will have higher precedence over DIP2 and the XCVR will be configured for 1.2G and so on.

The following table lists the status of LED's as per selected inputs.

Table 5 • **Output Status-LED**

Test No.	Input/Output	LED4	LED5	LED7	LED8	LED9	LED10	LED11
		TX_CLK_STABLE	RX_READY	RX_VAL	Status			
TEST 1	DIP1-1.25G	ON	ON	ON	ON	OFF	OFF	OFF
TEST 2	DIP2-2.5G	ON	ON	ON	OFF	ON	OFF	OFF
TEST 3	DIP3-5G	ON	ON	ON	OFF	OFF	ON	OFF
TEST 4	DIP5-CCC OUT0-OUT1	ON	ON	ON	OFF	OFF	ON	ON
TEST 5	DIP6-CCC OUT2-OUT3	ON	ON	ON	ON	ON	OFF	OFF

Probe internal signals of the design from the PolarFire FPGA through test points to get the output frequency. Table 6 lists the output signals and probing points to observe the clock changes on an oscilloscope.

Note: For more information about the Jumper locations on the board, refer to the silkscreen in *UG0747:PolarFire FPGA Evaluation Kit User Guide*.

Table 6 • **Output Status-Probe Test Points**

Signal	Header	Probe Test Point
CC-Out2	J7	Pin- 4
CC-Out3		Pin- 6
CC-Out0	J8	Pin- 5
CC-Out1		Pin- 3
RX_CLK_R	J8	Pin- 9
TX_CLK_R	J8	Pin- 13

4.1 Power Monitor

PolarFire Evaluation Kit board comes with power monitoring solution, implemented using the on-board SmartFusion A2F 200 device and the PowerMonitor application. The PowerMonitor application connects to the power monitoring program running on the A2F 200 device to measure power. For more information about PowerMonitor, refer to *UG0747: PolarFire FPGA Evaluation Kit User Guide*.

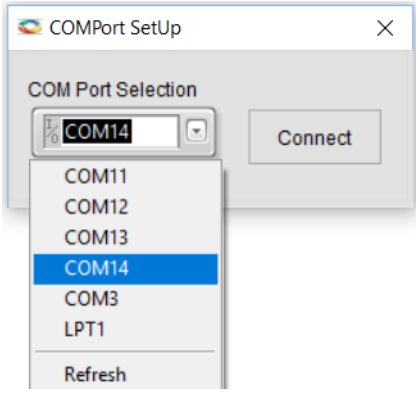
On the host PC, download the Microsemi PowerMonitor application from the following location and follow the instructions in the installation wizard to install the PowerMonitor application.

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

To measure the power, perform the following steps:

1. On the host PC desktop, click **Start** and select **PowerMonitor**.
2. In the COMPort SetUp dialog box, select the highest COM port from the drop-down and click **Connect** as shown in the following figure.

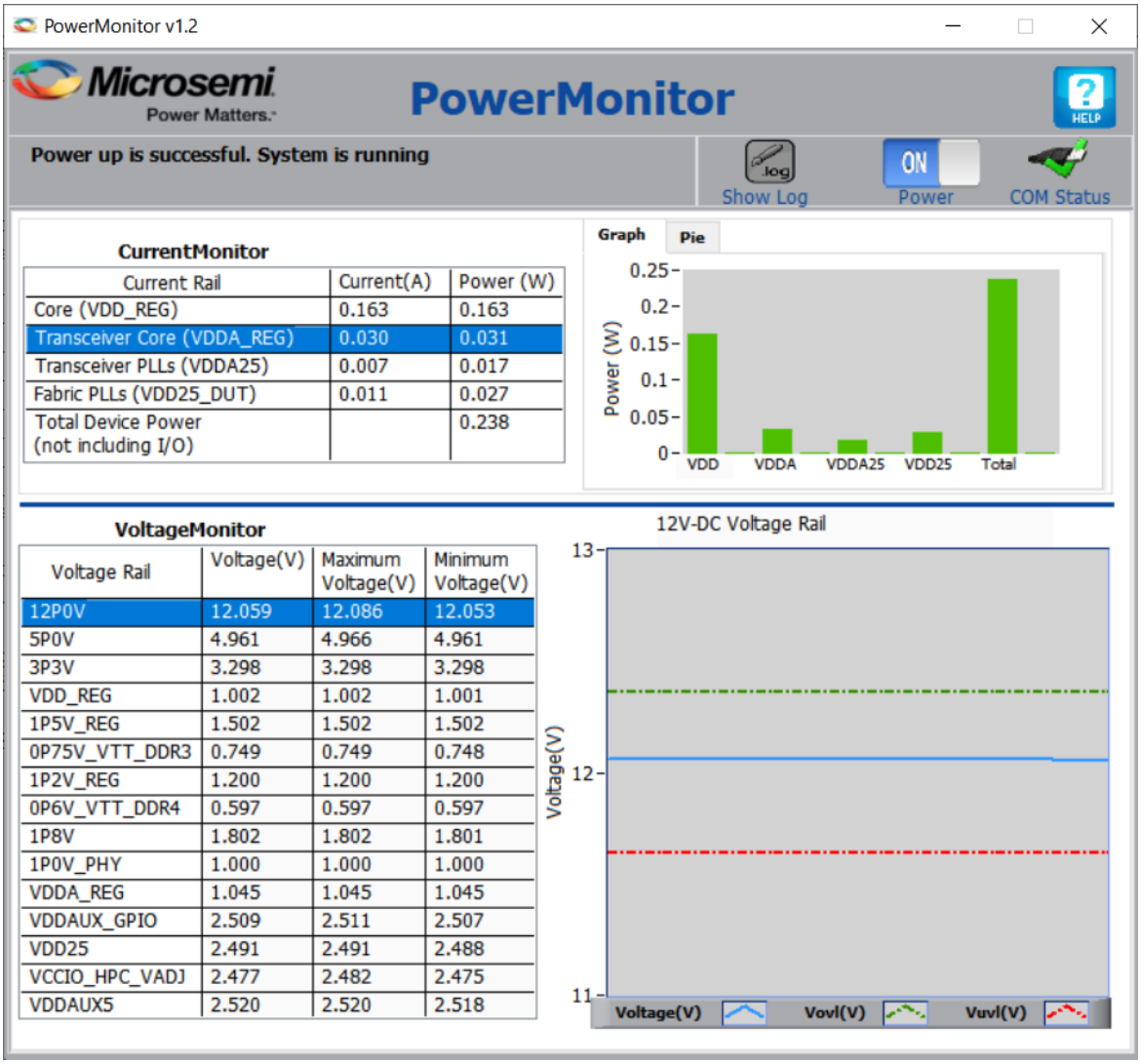
Figure 21 • **COM Port Setup**



The PowerMonitor application successfully connects to the board and starts displaying the Core Fabric (VDD) power, Fabric PLL (VDD25) power, Transceiver Core (VDDA) power, and Transceiver PLL (VDDA25) power.

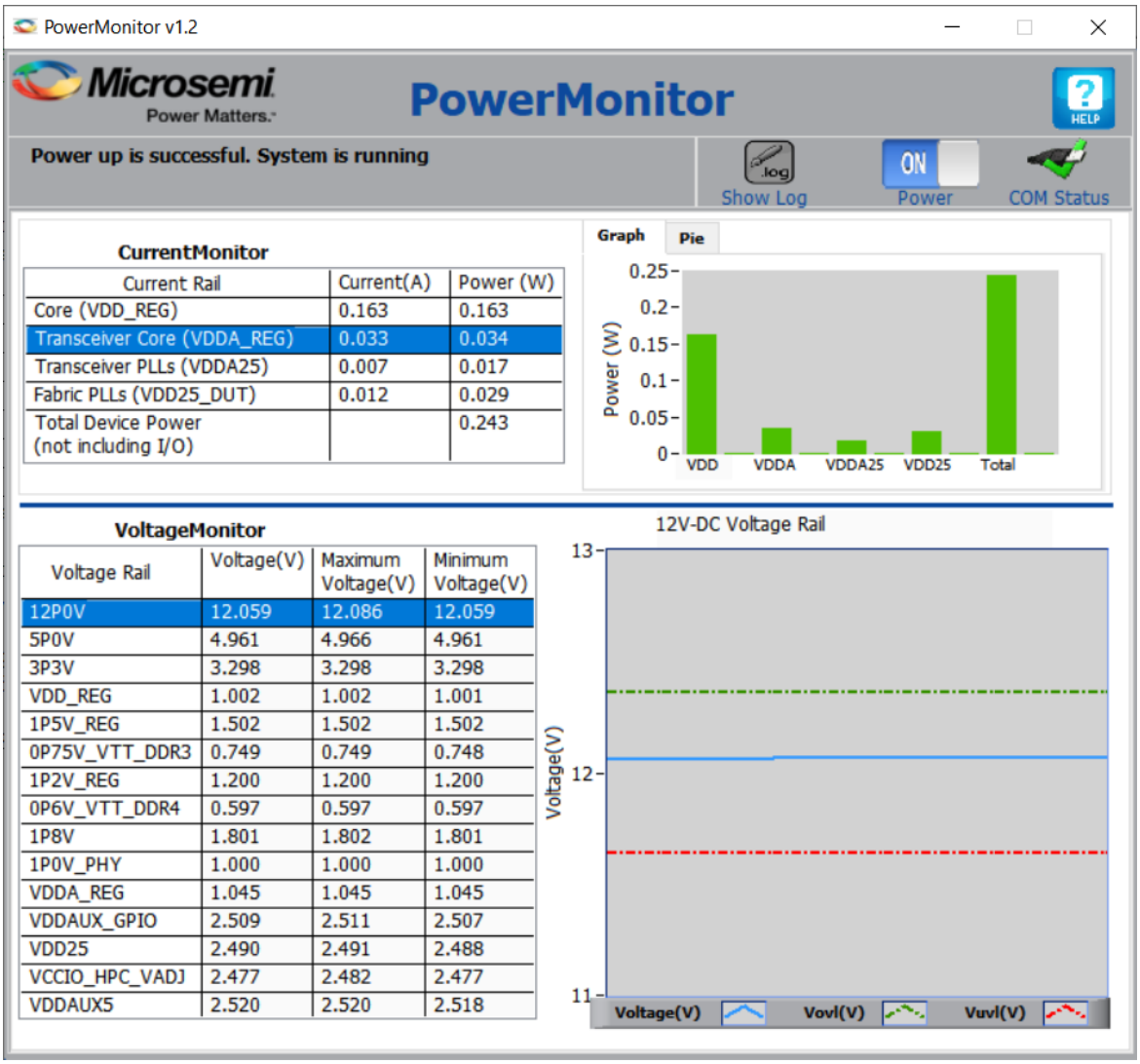
The following figure shows the total power consumed by the device is at 1.25G.

Figure 22 • **Power Usage at 1.25G**



The following figure shows the total power consumed by the device is at 2.5G.

Figure 23 • **Power Usage at 2.5G**



The following figure shows the total power consumed by the device is at 5G.

Figure 24 • **Power Usage at 5G**

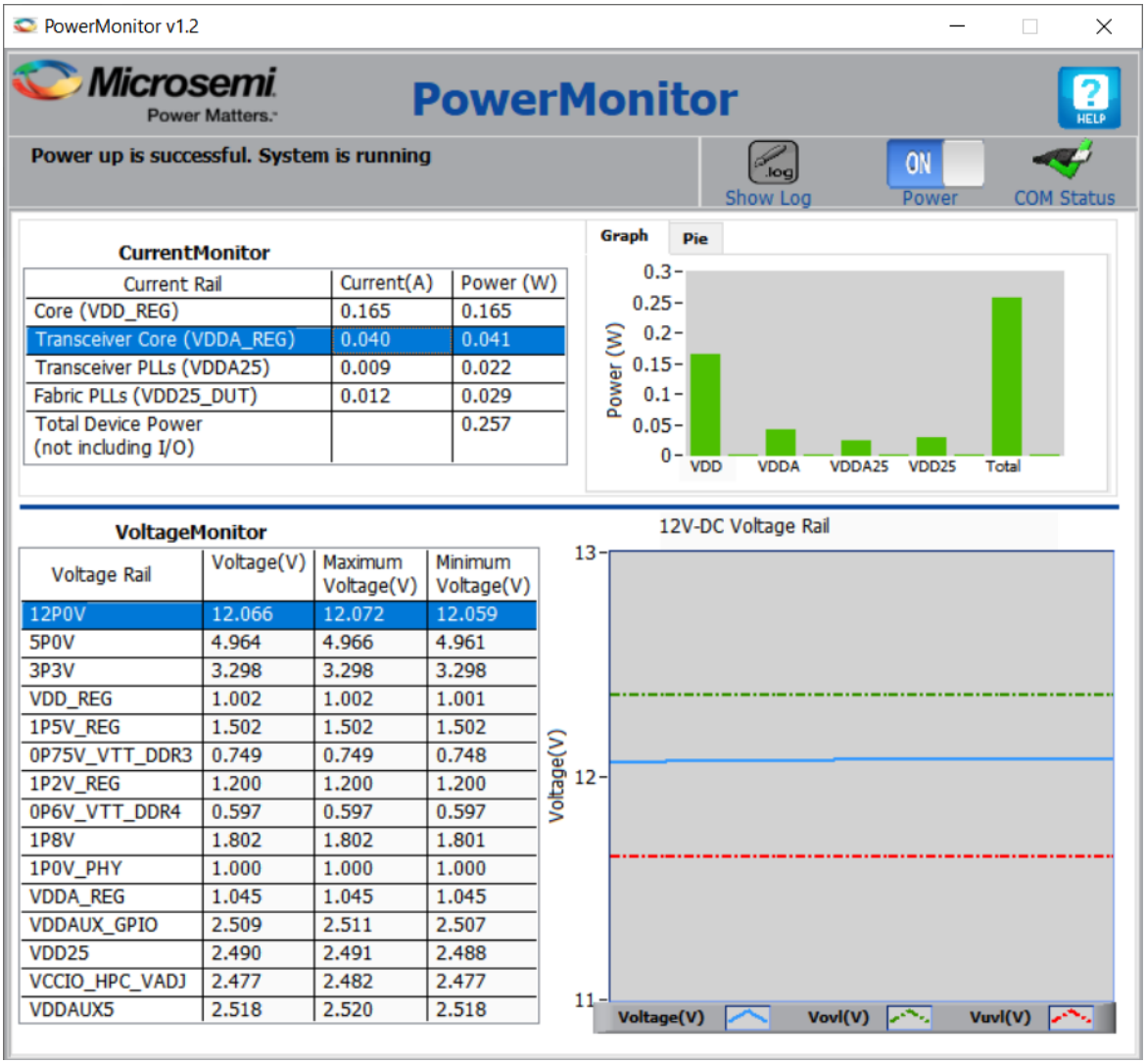


Table 7 • **Transceiver Power**

S.No	Lane Rate	Transceiver Power (W)	Total Power (W)
1	1.25G	0.031	0.238
2	2.5G	0.034	0.243
3	5G	0.041	0.257

This concludes Dynamic Reconfiguration of XCVR and CCC.

5 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the PolarFire device with the .job programming file using FlashPro Express. The .job file is available at the following design files folder location.

```
mpf_ac475_df\Programming_Job
```

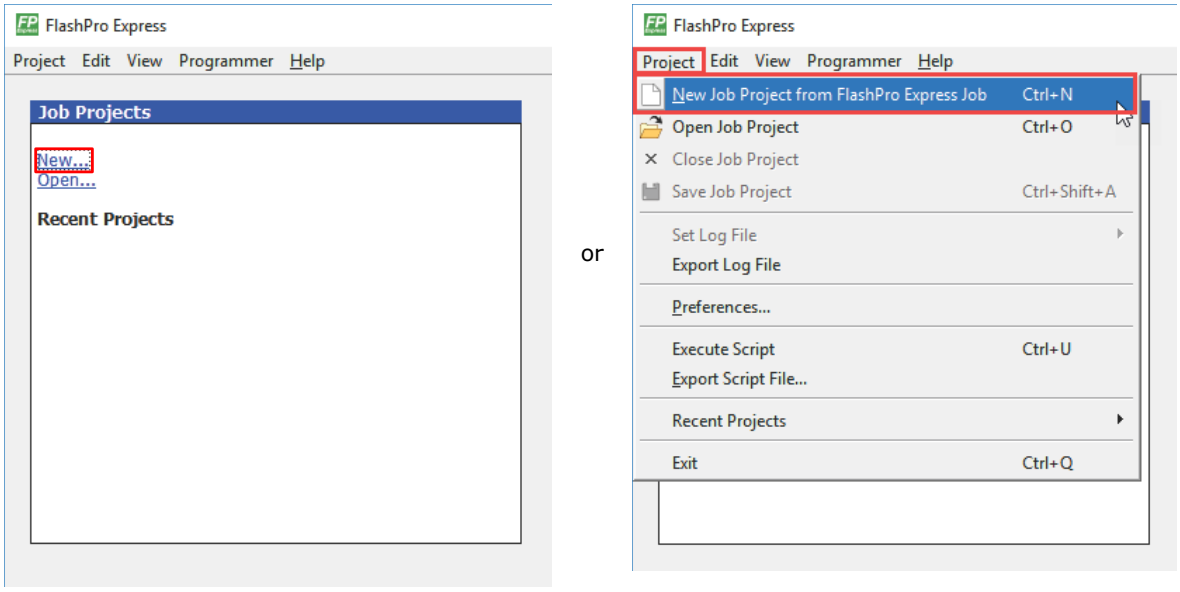
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as listed in Table 4, page 17.

Note: The power supply switch must be switched OFF while making the jumper connections.

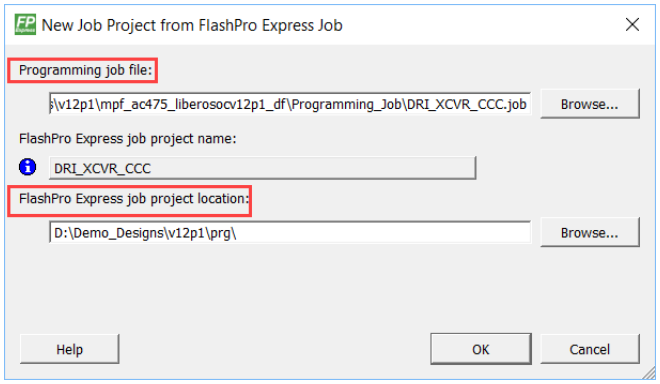
2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the Host PC to the **J5** (FTDI port) on the board.
4. Power on the board using the **SW3** slide switch.
5. Connect TXN to RXN and TXP to RXP using 2 SMA to SMA cables.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 25 • **FlashPro Express Job Project**



8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
   - **Programming job file**: Click **Browse**, navigate to the location where the .job file is located, and select the file. The default location is: *<download_folder>\mpf_ac475_df\Programming_Job.*
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

Figure 26 • **New Job Project from FlashPro Express Job**

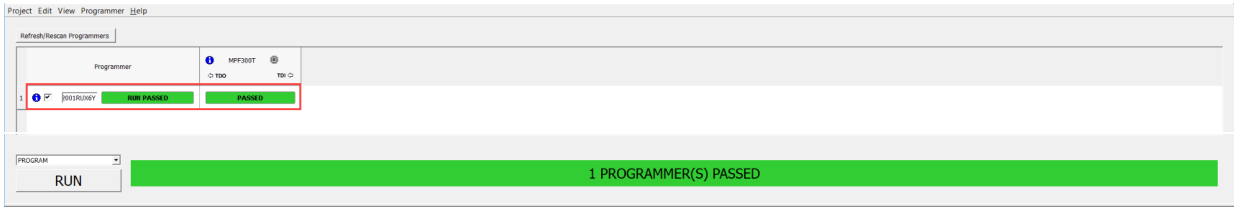


9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**.

Figure 27 • **Programming the Device**

11. Click **RUN.** When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure and the on-board LEDs 4, 5, 6, and 7 glow. Refer to Running the Demo, page 19 to run the TVS demo.

Figure 28 • **FlashPro Express—RUN PASSED**



12. Close **FlashPro Express** or in the **Project** tab, click **Exit**.

6 Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to **mpf_ac475_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.