

AC468
Application Note
PolarFire FPGA Transceiver Decision Feedback
Equalization



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

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

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.

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 Transceiver Decision Feedback Equalization	2
2.1	DFE Calibration	2
2.2	Design Requirements	3
2.3	Prerequisites	3
2.4	Demo Design	4
2.4.1	Design Implementation	5
2.5	Port Description	11
2.6	Clocking Structure	11
2.7	Reset Structure	12
3	Libero Design Flow	13
3.1	Synthesize	14
3.2	Resource Utilization	14
3.3	Place and Route	14
3.4	Verify Timing	15
3.5	Design and Memory Initialization	15
3.6	Generate Bitstream	15
3.7	Run PROGRAM Action	16
4	Running the Demo	18
5	Appendix 1: Programming the Device Using FlashPro Express	20
6	Appendix 2: Running the TCL Script	23

Figures

Figure 1	DFE Block Diagram	4
Figure 2	Top Level Libero Design	5
Figure 3	CCC Configuration	5
Figure 4	CoreABC Configuration	6
Figure 5	CoreABC Program	7
Figure 6	PF_XCVR_DRI Configuration	8
Figure 7	PolarFire Transceiver Reconfiguration GUI	9
Figure 8	Transmit PLL Configurator	10
Figure 9	Clocking Structure	11
Figure 10	Reset Structure	12
Figure 11	Libero Design Flow Options	13
Figure 12	I/O Editor-Transceiver View	14
Figure 13	Design Flow	15
Figure 14	Generate Design Initialization Data	15
Figure 15	Board Setup	17
Figure 16	Programming the Device	17
Figure 17	Launching SmartDebug Design	18
Figure 18	SmartDebug Window Debug Options	18
Figure 19	Eye Plot	19
Figure 20	FlashPro Express Job Project	20
Figure 21	New Job Project from FlashPro Express Job	21
Figure 22	Programming the Device	21
Figure 23	FlashPro Express—RUN PASSED	22

Tables

Table 1	Design Requirements	3
Table 2	Port Description	11
Table 3	Resource Utilization	14
Table 4	Jumper Settings for PolarFire Device Programming	16

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 23.

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 v12.0.

1.4 Revision 3.0

Updated the document for Libero SoC PolarFire v2.3.

1.5 Revision 2.0

Updated the document for Libero SoC PolarFire v2.2.

1.6 Revision 1.0

The first publication of this document.

2 PolarFire FPGA Transceiver Decision Feedback Equalization

The PolarFire® FPGA family includes multiple embedded low-power, performance-optimized transceivers. Each transceiver has both the Physical Medium Attachment (PMA), protocol 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.

At the receiver front end, Continuous Time Linear Equalization (CTLE) with optional auto calibration compensates the High-Frequency losses to improve the received signal integrity. However, for lossy channels, the CTLE technique amplifies the high-frequency noise along with the data. This can be overcome by using Decision feedback equalization which mitigates lane noise caused due to Inter Symbol Interference (ISI) or cross-talk without amplifying the high-frequency noise within the data.

CTLE technique is sufficient to interpret data up to 8 Gbps for short reach applications. Beyond this, Decision Feedback Equalizer (DFE) is capable of equalizing channel response.

For more information on CTLE, DFE, and Transceiver insertion loss, see [UG0677: PolarFire FPGA Transceiver User Guide](#).

An optionally enabled 5-tap DFE is available to equalize the lane response in conjunction with the CTLE.

The DFE-based operation uses current bit information to cancel ISI for the following bit through a feedback mechanism, allowing the following bits to be correctly sampled. Using taps to delay and by multiplying the symbols, the DFE effectively cancels out interference on the analog signal.

The operation is nonlinear, allowing it to overcome the notch response that the CTLE does not perform. The DFE also includes an automatic calibration that finds the best possible tuning to match the transceiver lane to the system channel. DFE mode supports serial data transmission rates from 3 Gbps to 12.7 Gbps.

This demo design demonstrates the simple procedure to perform run time DFE calibration using a Dynamic Reconfiguration Interface (DRI). It also shows how to plot eye diagrams using the SmartDebug tool and verify signal integrity in DFE mode.

DFE equalization enables PolarFire transceivers to be efficient for systems running at approximately 10 Gbps or above where channel complexity is higher.

2.1 DFE Calibration

DFE Calibration is carried out by a robust descent algorithm that has been optimized to avoid local minima, achieve predictable results, enable low area, and operate at high clock speeds. It adjusts the feedback coefficients (H1 - H5) by trial-and-error in response to the eye-area.

The algorithm operates on one dimension (a single coefficient) at a time. It takes a step of size 1 in the positive and then the negative direction that is H1+1 and H1-1. If the area improves on either step, it continues to take another step in the same direction. If both directions yield a lower area, it continues to the next coefficient with the same step size. After failing to improve the area on all coefficients, it will increase the step size and continue. If the area is improved the step size immediately reduces to 1.

These demo designs can be programmed using either of the following options:

- **Using the.job file:** To program the device using the .job file provided with the design files, see [Appendix 1: Programming the Device Using FlashPro Express](#), page 20.
- **Using Libero SoC:** To program the device using Libero SoC, see [Table 4](#), page 16.

2.2 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 (MPF300-EVAL-KIT)	Rev D or later
2 SMA-to-SMA cables with 10 Gbps support (not provided with the kit)	
Host PC	
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® SoC Design Suite	

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.

2.3 Prerequisites

Before you begin:

1. For demo design files download link:
http://soc.microsemi.com/download/rsc/?f=mpf_ac468_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>
 The latest versions of ModelSim and Synplify Pro are included in the Libero SoC installation package.

2.4 Demo Design

The following steps describe the data flow in the demo design and ensure the transceiver CDR is locked.

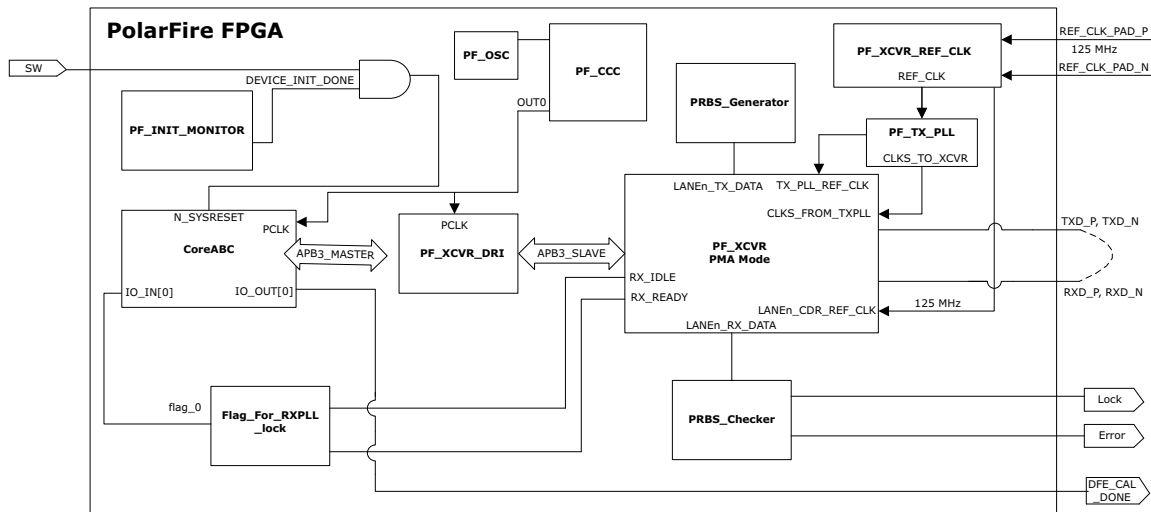
1. The reference design uses a transceiver interface (PF_XCVR) configured in native PMA mode running at 10.3125 Gbps data rate, 40-bit PCS fabric interface, and using 125 MHz reference clock.
2. The PRBS_Generator module generates a PRBS-7 pattern and forwards the data to the Transceiver TX end.
3. The differential serial data of TX and RX is looped back using onboard SMA-to-SMA cables
4. This data is then received by the PRBS_checker module which checks for data match. If matched the Lock is asserted, otherwise, an error signal is generated.

The following steps describe how DFE calibration is triggered.

1. When the CDR is locked, valid RX_IDLE and RX_READY signals are sent to the Flag_for_RXPLL_lock_0 module. Note that, RX_IDLE is low during data transmission and RX_READY is asserted when CDR is locked to the incoming serial data.
2. Flag_for_RXPLL_lock_0 module determines proper transceiver RX PLL lock using the condition (NOT RX_IDLE) AND RX_READY.
3. When the preceding condition is true, the Flag_o is asserted and sent to the CoreABC module.
4. This initiates the DFE calibration sequence using the CoreABC interface.
5. CoreABC acts as APB3 master and dynamically performs Read/Write to transceiver registers using PF_XCVR_DRI interface. The transceiver interface is connected as APB3 slave to the PF_XCVR_DRI interface.
6. When the DFE calibration sequence is successful, the DFE_CAL_DONE flag is asserted.

The following figure shows the block diagram of the reference design.

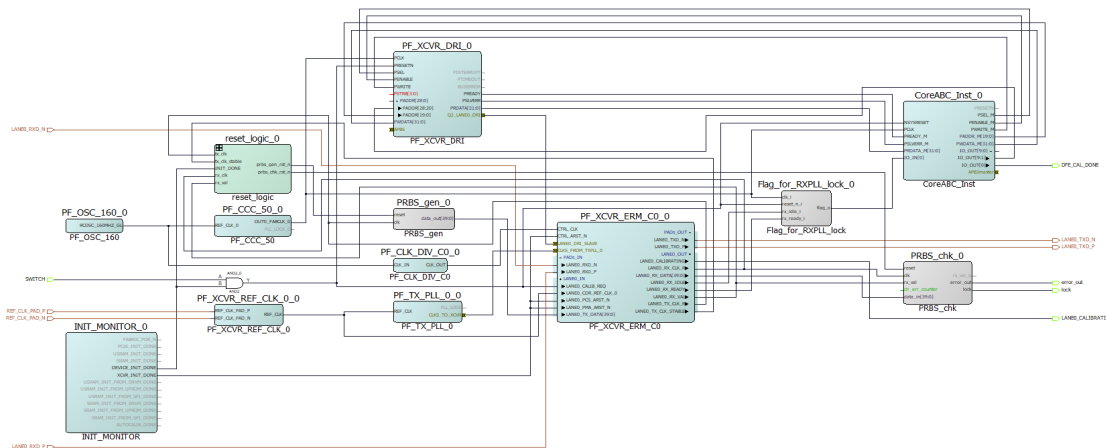
Figure 1 • DFE Block Diagram



2.4.1 Design Implementation

The following figure shows the top-level Libero design of the PolarFire Transceiver DFE design.

Figure 2 • Top Level Libero Design



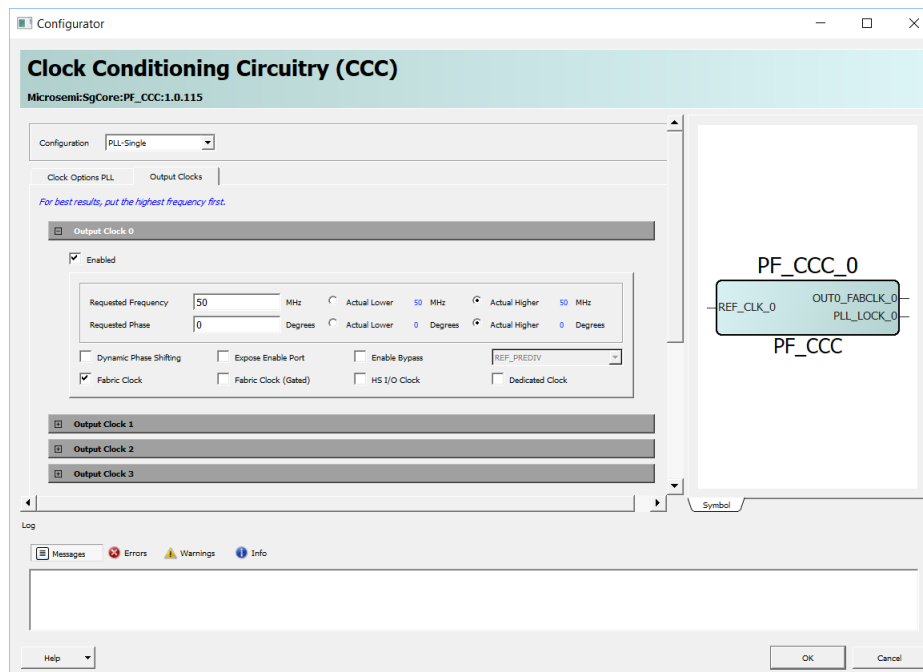
The following sections describe the IP cores used in the design and their configurations.

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

2.4.1.1 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. In this design, the clock is configured to 50 MHz.

Figure 3 • CCC Configuration

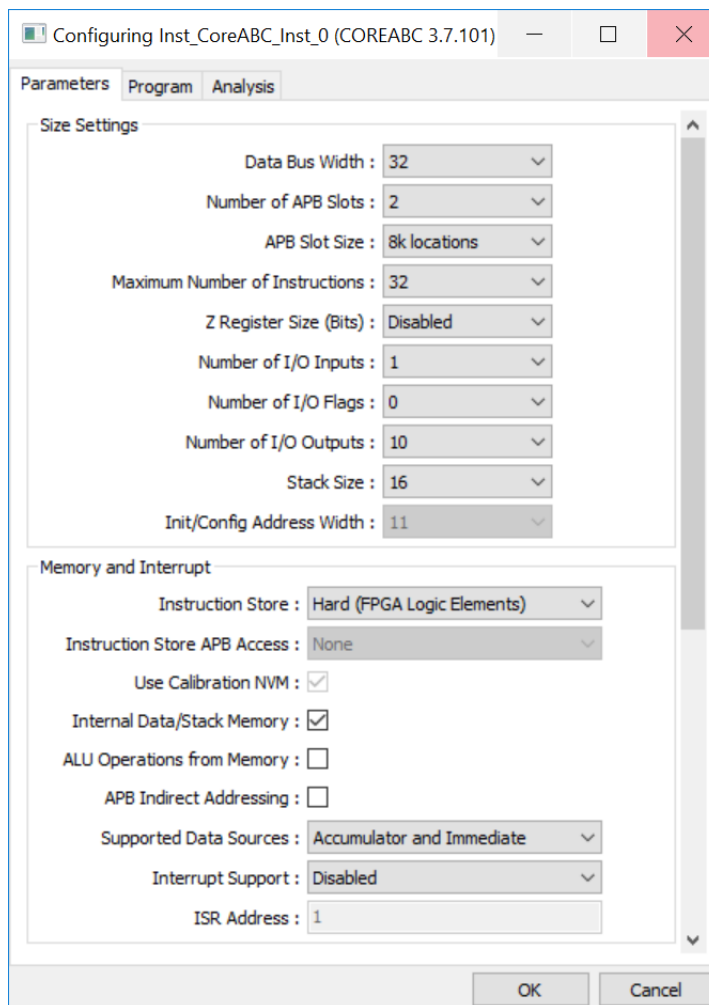


2.4.1.2 CoreABC

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

CoreABC in this reference design is connected to the DRI as an APB3 master. The APB3 slave of DRI is connected to Transceiver. The CoreABC initiates the DFE calibration sequence and dynamically performs Read /Write operation on the Transceiver register. The number of APB slots, APB slot size and the maximum number of instructions are configured depending on the number of peripherals and address size used. The following figure shows the parameter configuration of CoreABC interface.

Figure 4 • CoreABC Configuration



Configuring Inst_CoreABC_Inst_0 (COREABC 3.7.101)

Parameters Program Analysis

Size Settings

- Data Bus Width : 32
- Number of APB Slots : 2
- APB Slot Size : 8k locations
- Maximum Number of Instructions : 32
- Z Register Size (Bits) : Disabled
- Number of I/O Inputs : 1
- Number of I/O Flags : 0
- Number of I/O Outputs : 10
- Stack Size : 16
- Init/Config Address Width : 11

Memory and Interrupt

- Instruction Store : Hard (FPGA Logic Elements)
- Instruction Store APB Access : None
- Use Calibration NVM : ☒
- Internal Data/Stack Memory : ☒
- ALU Operations from Memory : ☐
- APB Indirect Addressing : ☐
- Supported Data Sources : Accumulator and Immediate
- Interrupt Support : Disabled
- ISR Address : 1

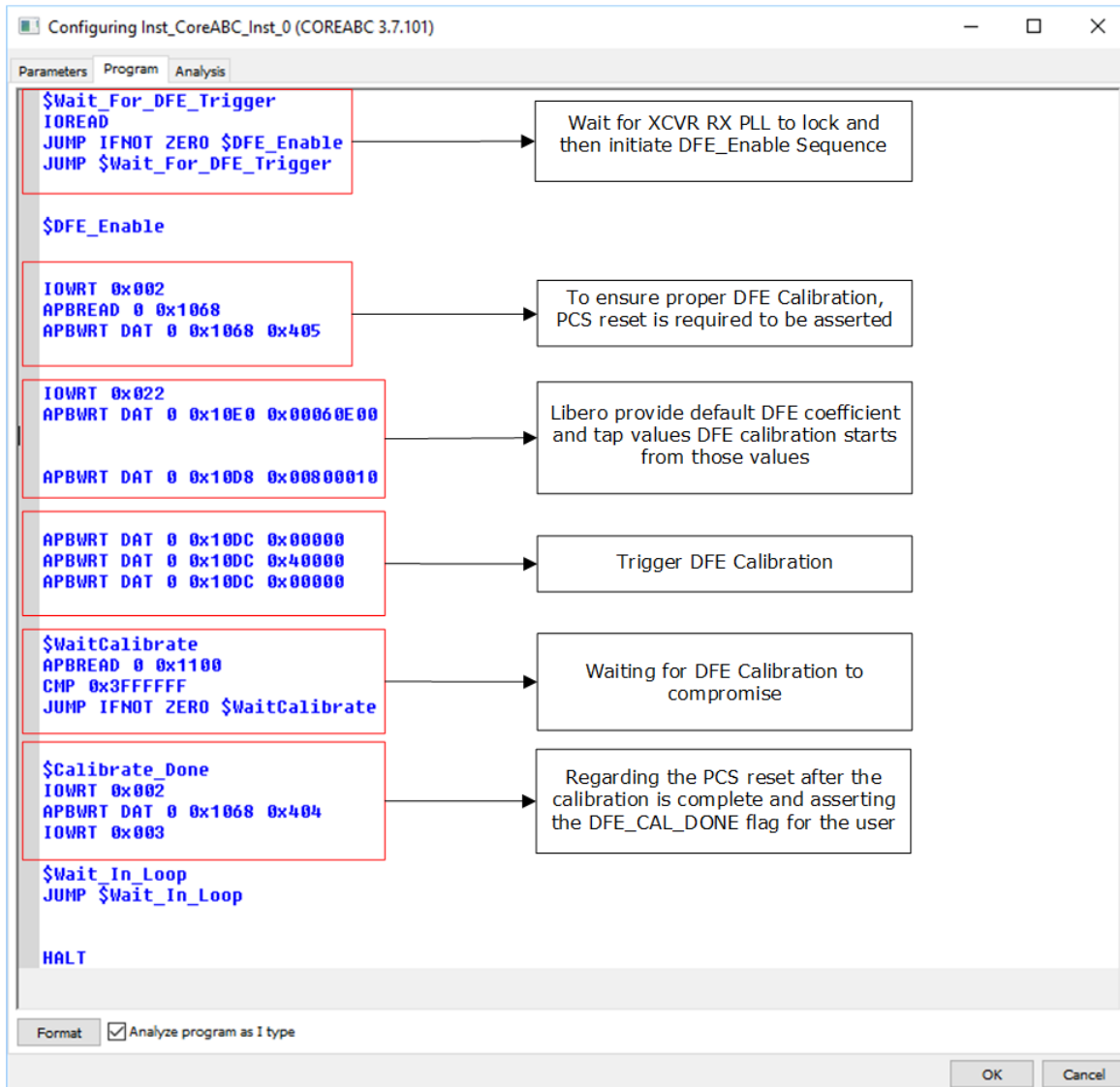
OK Cancel

2.4.1.2.1 CoreABC Program

The following figure illustrates the register settings required for performing DFE calibration.

Note: The register address changes depending on the transceiver Quad and Lanes used. In this demo, Q2 Lane0 is used. For more information, see [Device Register Map](#).

Figure 5 • CoreABC Program

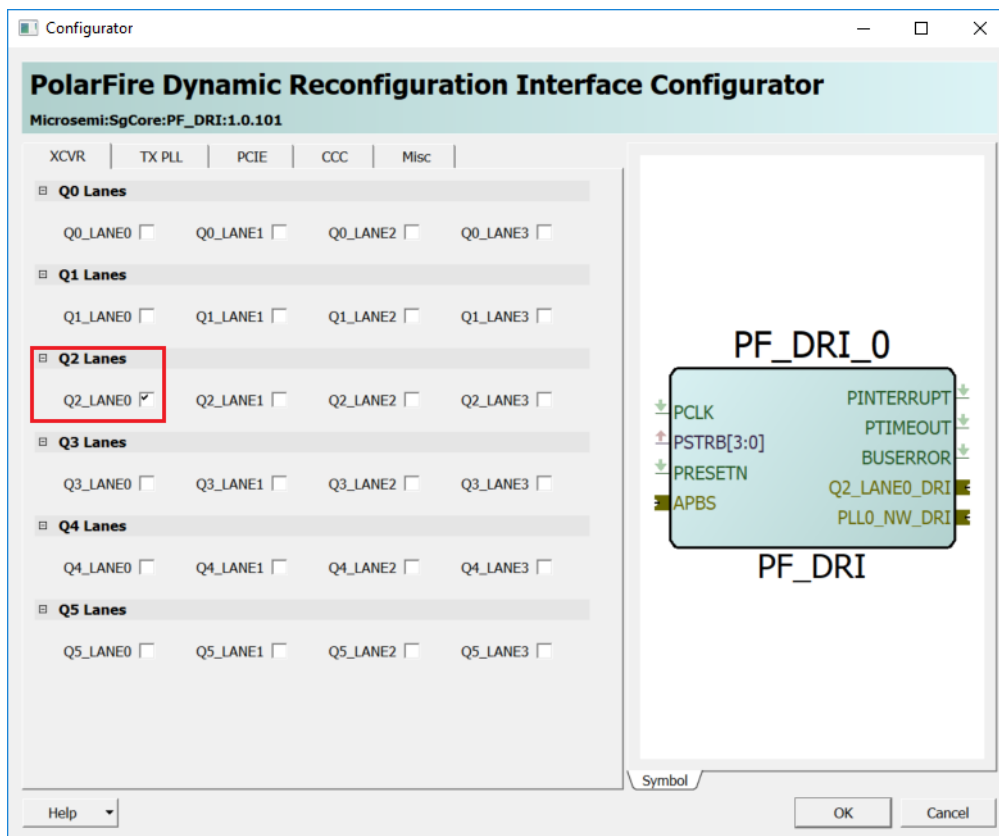


2.4.1.3 Dynamic Reconfiguration Interface

DRI performs the run time configuration of transceiver PMA/PCS, PCIe, CCC, and Transmit PLLs after initialization.

In this demo design, DRI is used for performing 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.

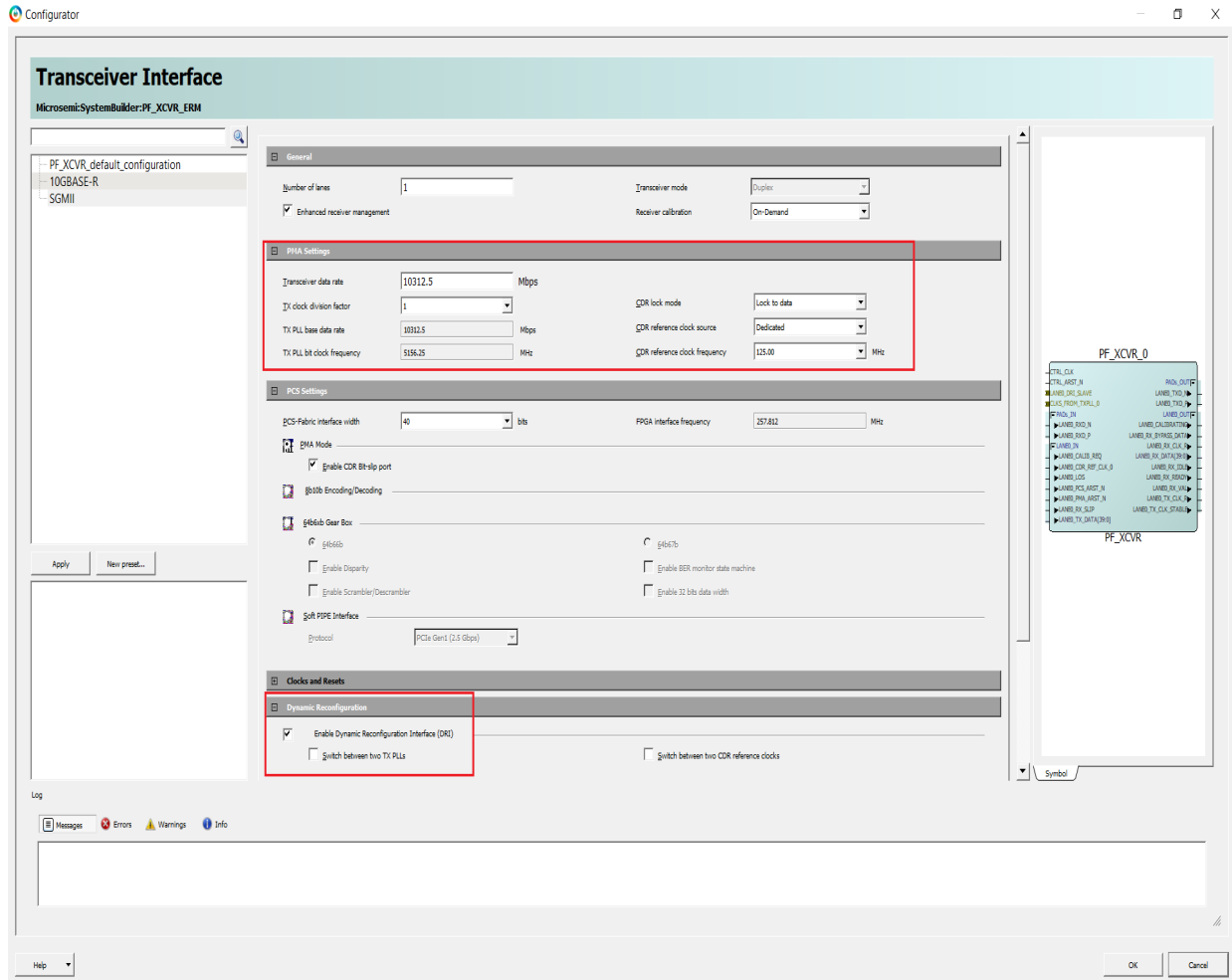
Figure 6 • PF_XCVR_DRI Configuration



2.4.1.4 Transceiver Interface Reconfiguration

The PolarFire transceiver interface configurator is set to 10.3125 Gbps, 40-bit PCS-Fabric interface width and native PMA mode. The ERM is enabled and receiver calibration is done on-demand. The following figure shows the PolarFire Transceiver Interface configurator settings and how to enable DRI.

Figure 7 • PolarFire Transceiver Reconfiguration GUI



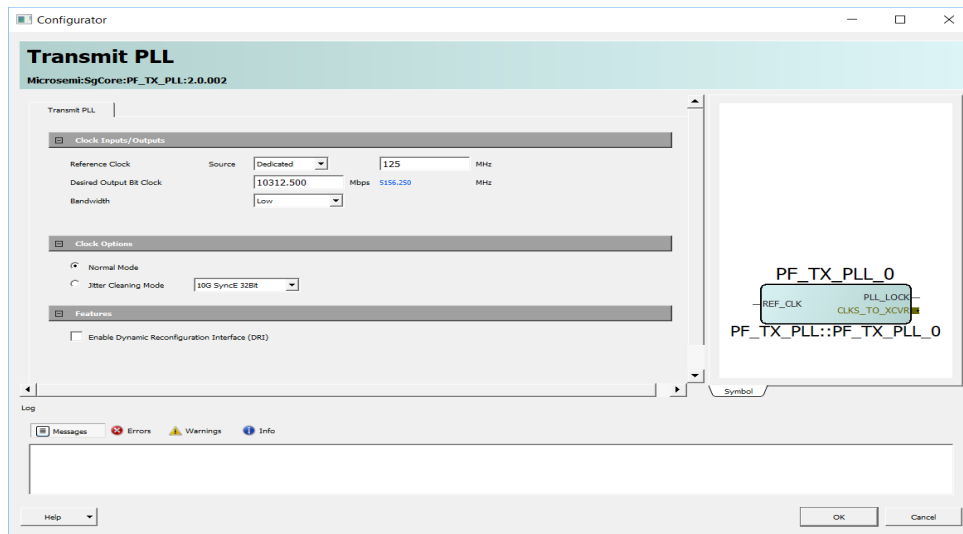
2.4.1.5 PolarFire Transceiver Reference Clock

The transceiver reference clock can be configured either as a differential clock or as a two single-ended REFCLKs. This design requires a single REFCLK. The REFCLK source the transceivers, and global clock network in this design. The reference clock 0 is configured as a differential reference clock.

2.4.1.6 Transmit PLL

The transmit PLL reference clock and desired output clock are set to 125 MHz and 5 MHz, respectively as shown in the following figure. The PolarFire transceiver is a half-rate architecture that is the internal high-speed path that uses both edges of the clock to keep the clock rates down. The clock thus runs at half of the data rate, thereby consuming less dynamic power.

Figure 8 • Transmit PLL Configurator



2.4.1.7 Flag_for_RXPLL_Lock

The `Flag_for_RXPLL_Lock` looks for signal activity on the transceiver. This is done by looking for `RX_READY` going HIGH and `RX_IDLE` going LOW (`RX_READY & ! RX_IDLE`). The flag output is used as an input to the CoreABC to start DFE.

2.4.1.8 PRBS Generator and Checker

The generator implements the PRBS polynomial and generates a continuous sequence of PRBS7 patterns of 40 bits each. The PRBS checker receives data from the transceiver to generate PRBS data locally, the two are then compared for data integrity. A lock signal is asserted if there is a data match otherwise an error signal is asserted to the user.

2.5 Port Description

The following table lists the key signals for this design.

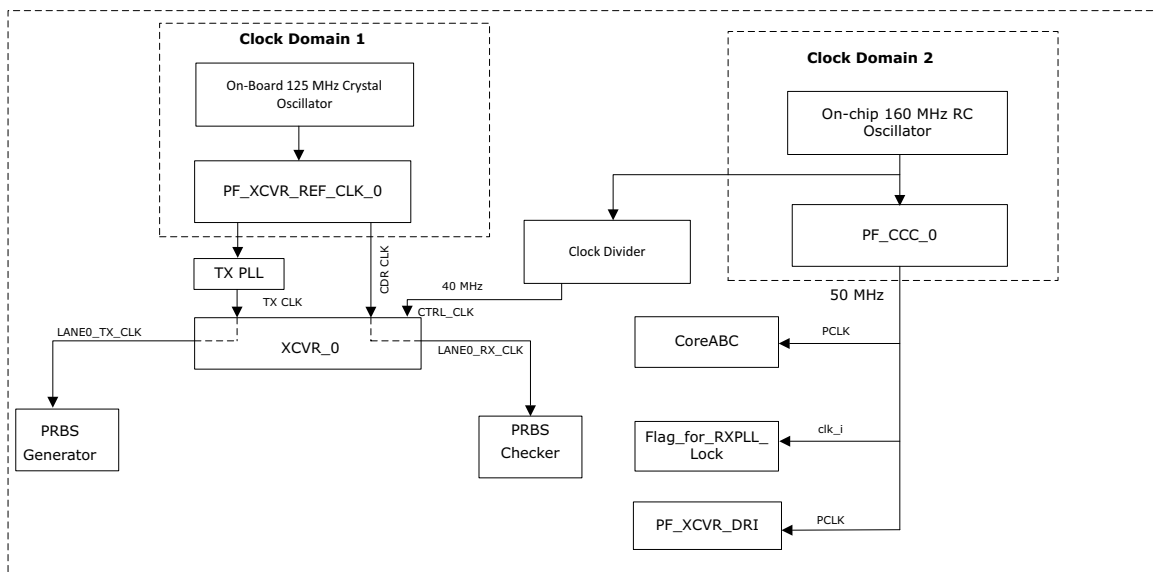
Table 2 • Port Description

Signal	Direction	Description
REF_CLK_PAD_P and REF_CLK_PAD_N	Input	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
SWITCH	Input	DIP switch setting to initiate DFE calibration Trigger for DFE calibration can also be generated by user design using the condition (RX_READY & ! RX_IDLE).
LANE0_TXD_N	Output	Transceiver transmitter differential output
LANE0_TXD_P	Output	Error flag generated from PRBS checker module when there is data mismatch
error_out	Output	Dynamic CCC OUT3 Fabric Clock
Lock	Output	Lock signal flag generated from PRBS checker module when there is data match

2.6 Clocking Structure

In this reference 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 in this design. The on-chip 160 MHz RC oscillator drives the CoreABC, Flag_for_RXPLL_lock, PF_XCVR_DRI block, and the ERM module of XCVR. The following figure shows the clocking structure in this reference design.

Figure 9 • Clocking Structure



2.7 Reset Structure

In this reference design, the reset signal of the PRBS generator and PRBS checker, are issued using the Reset_logic module. Reset_sync_tx_0 (CoreReset_PF) module releases active low reset of data generator block when TX_CLK_STABLE from PF_XCVR interface and DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted.

Similarly, Reset_sync_rx_0 (CoreReset_PF) module releases active low reset of data checker when RX_READY from PF_XCVR interface, DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted.

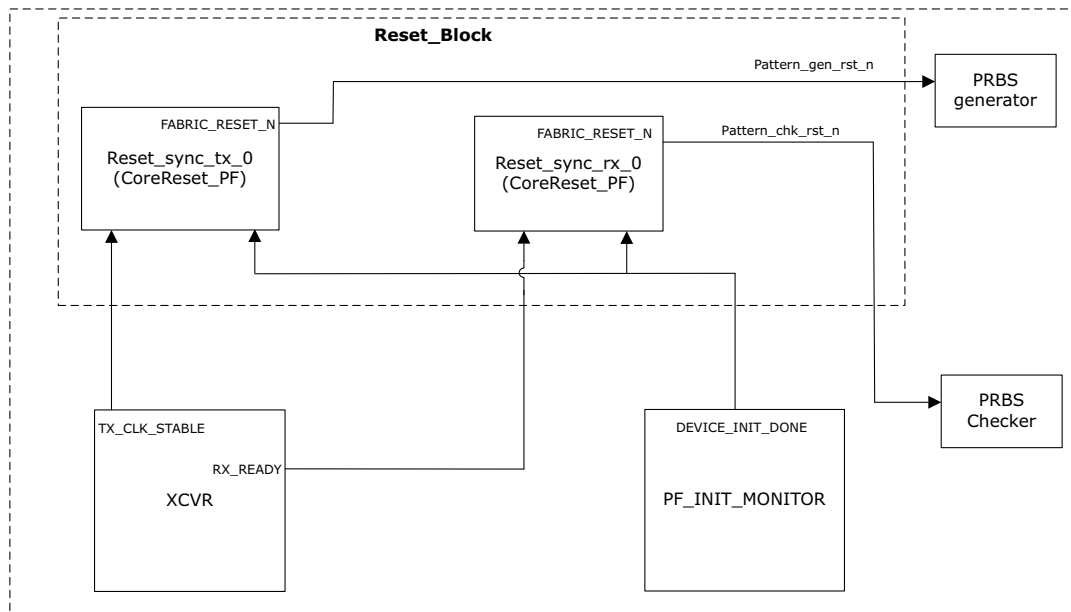
The previous setup ensures that the PRBS generation and PRBS checker does not start until the TX_clock_stable and RX_READY are asserted respectively.

DEVICE_INIT_DONE signal is asserted when the device initialization is complete. For more information about device initialization, see [UG0725: PolarFire FPGA Device Power-Up and Resets User Guide](#).

For more information on CoreReset_PF IP core, see [CoreReset_PF handbook](#) from the Libero catalog.

The following figure shows the reset structure in this reference design.

Figure 10 • Reset Structure



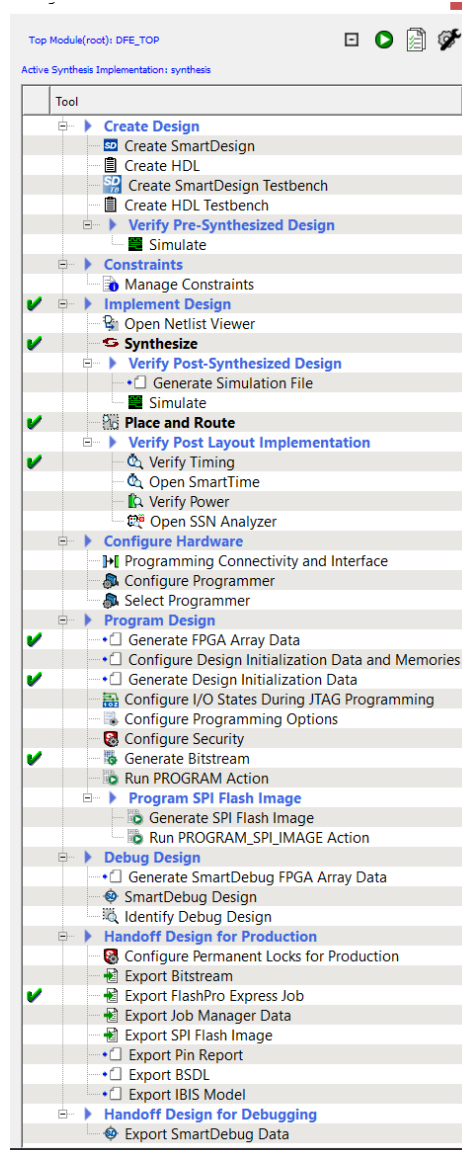
3 Libero Design Flow

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

- Synthesize, page 14
- Resource Utilization, page 14
- Place and Route, page 14
- Verify Timing, page 15
- Design and Memory Initialization, page 15
- Generate Bitstream, page 15
- Run PROGRAM Action, page 16

The following figure shows these options in the **Design Flow** tab.

Figure 11 • Libero Design Flow Options



3.1 Synthesize

To synthesize the design, perform the following steps:

1. In the **Design Flow** tab, double-click **Synthesize**.
When the synthesis is successful, a green tick mark appears as shown in Figure 11, page 13.
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 DFE design after synthesis.

Note: These values may vary slightly for different Libero runs, settings, and seed values.

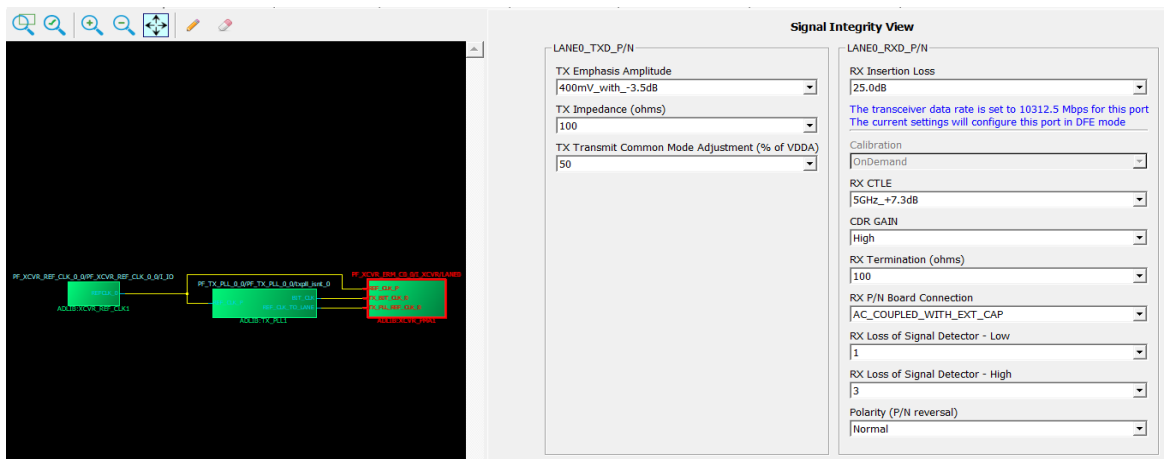
Table 3 • Resource Utilization

Type	Used	Total	Percentage
4LUT	1148	299544	0.38
DFF	697	299544	0.23
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 DFE design, the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor as shown in the following figure.

Figure 12 • I/O Editor-Transceiver View

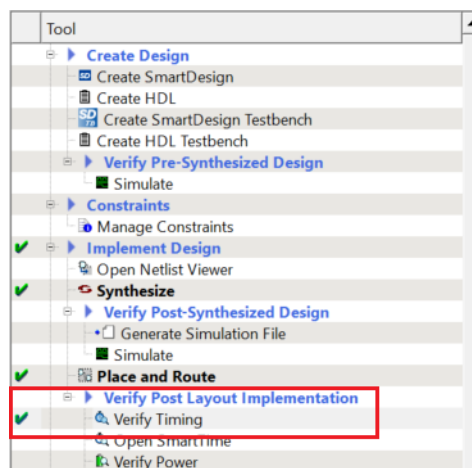


3.4 Verify Timing

To verify timing, perform the following steps:

1. From the **Design Flow** tab, double-click **Verify Timing**.
When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 11, page 13.
2. Right-click **Verify Timing** and select **View Report** to view the verify timing report and log files in the **Reports** tab.

Figure 13 • 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, see *UG0725: PolarFire FPGA Device Power-up and Resets User Guide*.

Figure 14 • 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. From the **Design Flow** tab, double-click **Generate Bitstream**. When the bitstream is successfully generated, a green tick mark appears as shown in Figure 11, page 13

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.

Follow these steps to program the PolarFire device:

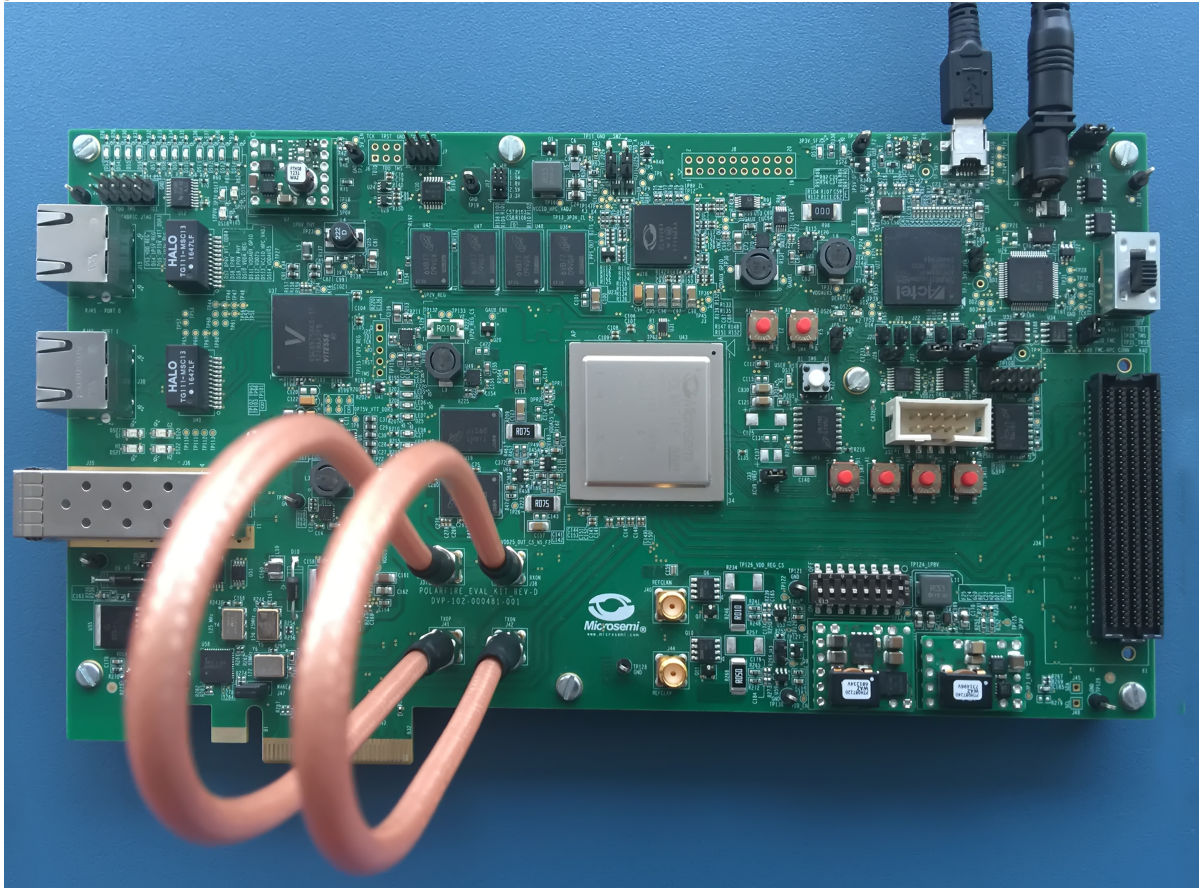
1. Ensure that the following jumper settings are set on the board.

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 1 and 2 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 board setup.

The following figure shows the board setup after these connections are made.

Figure 15 • 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 11, page 13.

Figure 16 • Programming the Device

4 Running the Demo

This section describes how to optimize DFE coefficients, and check the result on board.

Prerequisites for the procedure:

- The PolarFire Evaluation board is connected.
- The PolarFire FPGA is programmed with the DFE design.

To run the demo, perform the following steps:

1. After the device is programmed, change **SW11 DIP1** from 0 to 1. This brings the CoreABC interface out of reset and starts DFE calibration.

Note: Trigger for DFE calibration can also be generated by monitoring the condition (RX_READY & ! RX_IDLE)

2. Observe if LED4 is OFF, and LED10 and LED5 are ON. This signifies that run time DFE calibration is complete and there is no bit error. Where LED4 and LED10 represent bit error status and LED5 represents DFE calibration status.

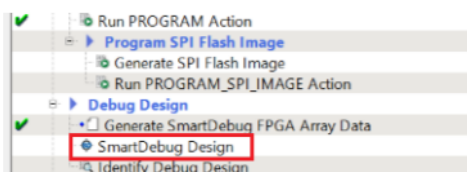
Eye Monitor enables visualizing the eye diagram present within the receiver. This feature plots the receive eye after the CTLE and DFE functions.

For plotting the Eye Diagram, follow the procedure enlisted.

On the Design Flow window, perform the following steps:

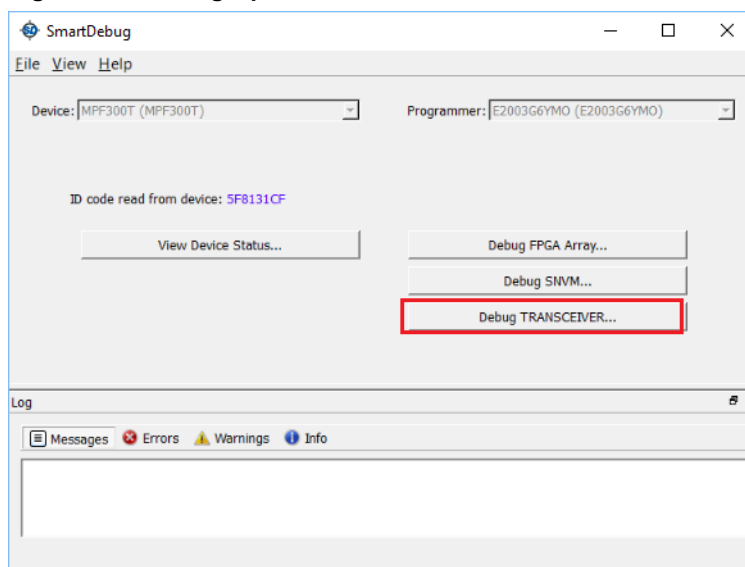
1. Select **Generate SmartDebug FPGA Array Data** to generate data for SmartDebug Design. Once the data is generated, a green tick mark is seen on the left side of the option indicating that the data generation is successful.
2. Open **SmartDebug Design**.

Figure 17 • Launching SmartDebug Design



3. Click **Debug TRANCEIVER** to open the SmartDebug window as shown in following figure.

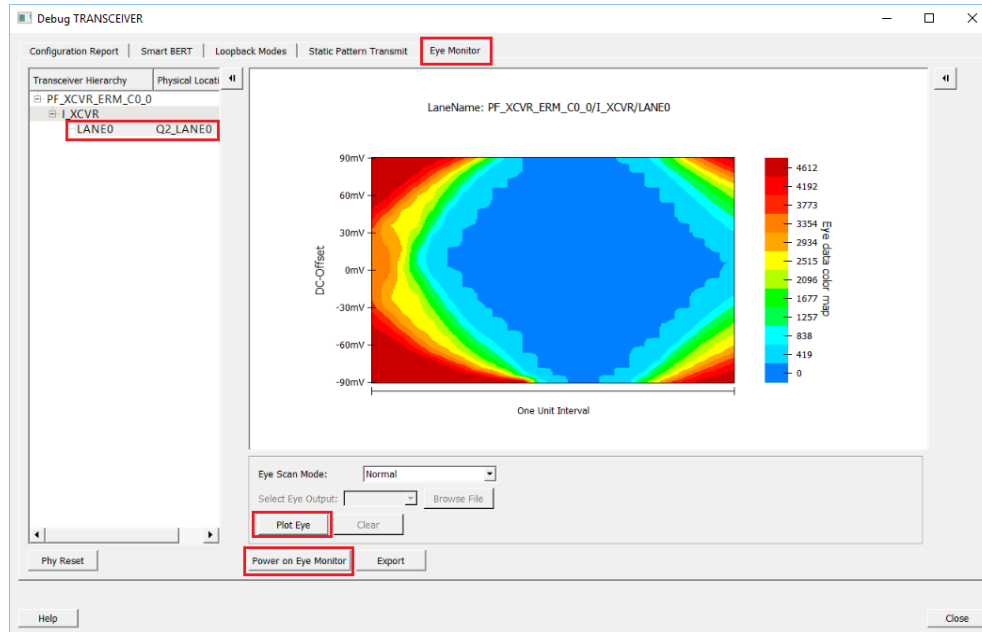
Figure 18 • SmartDebug Window Debug Options



4. Go to **EYE Monitor tab** and Select **LANE0**.
5. Click **Power on Eye Monitor**.
6. Click **Plot Eye** to plot the eye.

The following figure shows the Eye Plot.

Figure 19 • Eye Plot



5 Appendix 1: Programming the Device Using FlashPro Express

This chapter describes how to program the PolarFire device with the Job programming file using a FlashPro programmer. The default location of the Job file are located at following location:

`mpf_ac468_df\Programming_Job\PF_XCVR_DFE.job`

To program the PolarFire device using FlashPro Express, complete the following steps:

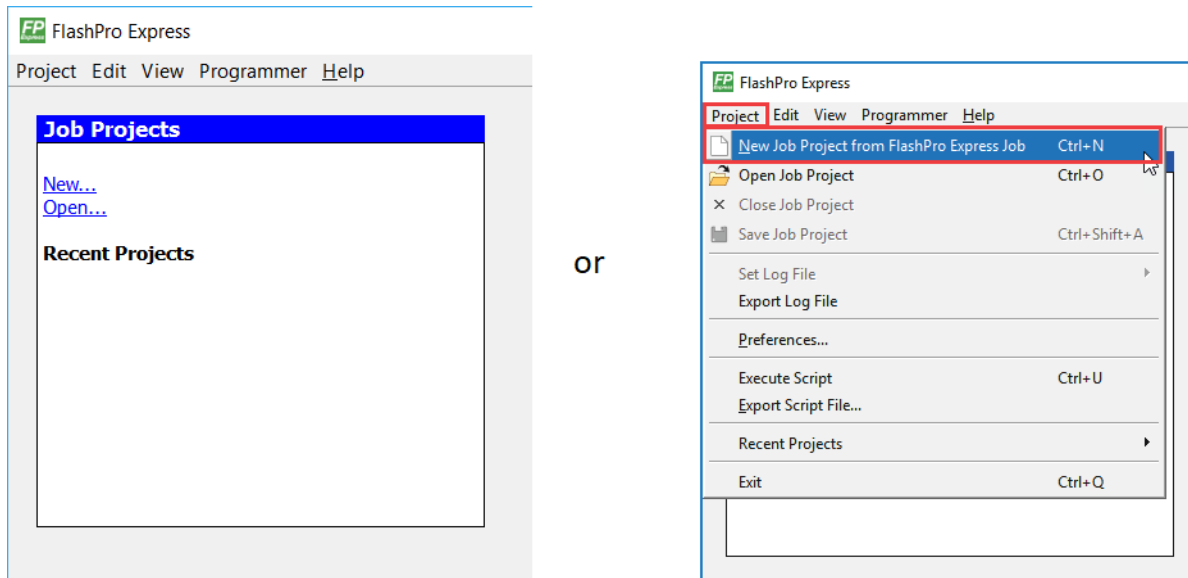
1. Ensure that the jumper settings on the board are the same as listed in Table 4, page 16.

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

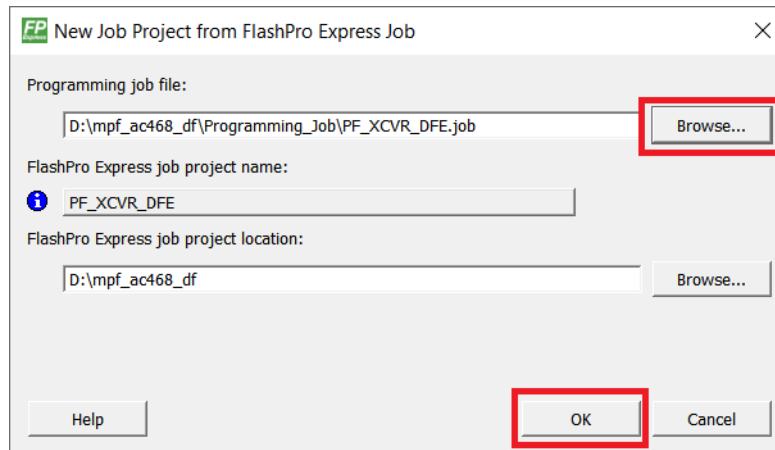
2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the Host PC to the **J5** (FTDI port) on the board.
4. Power on the board using the **SW3** slide switch.
5. On the host PC, launch the FlashPro Express software.
6. To create a new job, click **New** or

In the Project menu, select **New Job Project from FlashPro Express Job** as shown in the following figure.

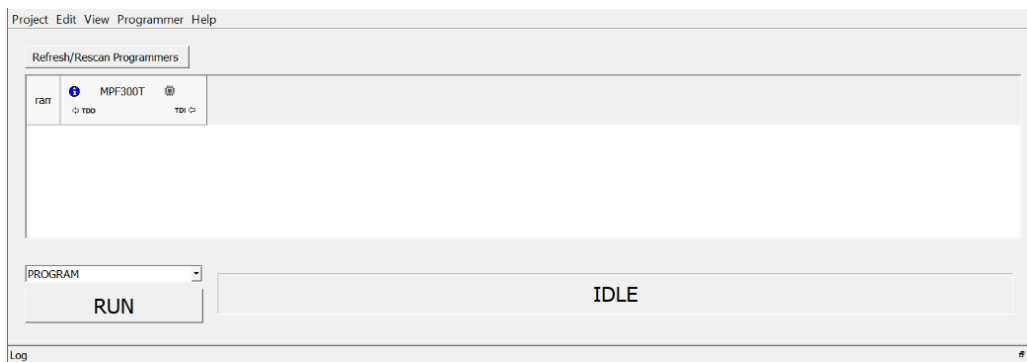
Figure 20 • FlashPro Express Job Project



7. Enter the following files in the New Job Project from the 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_ac468_df\Programming_Job\PF_XCVR_DFE.job`.
 - FlashPro Express job project location: Click **Browse** and navigate to the location where you want to save the project.

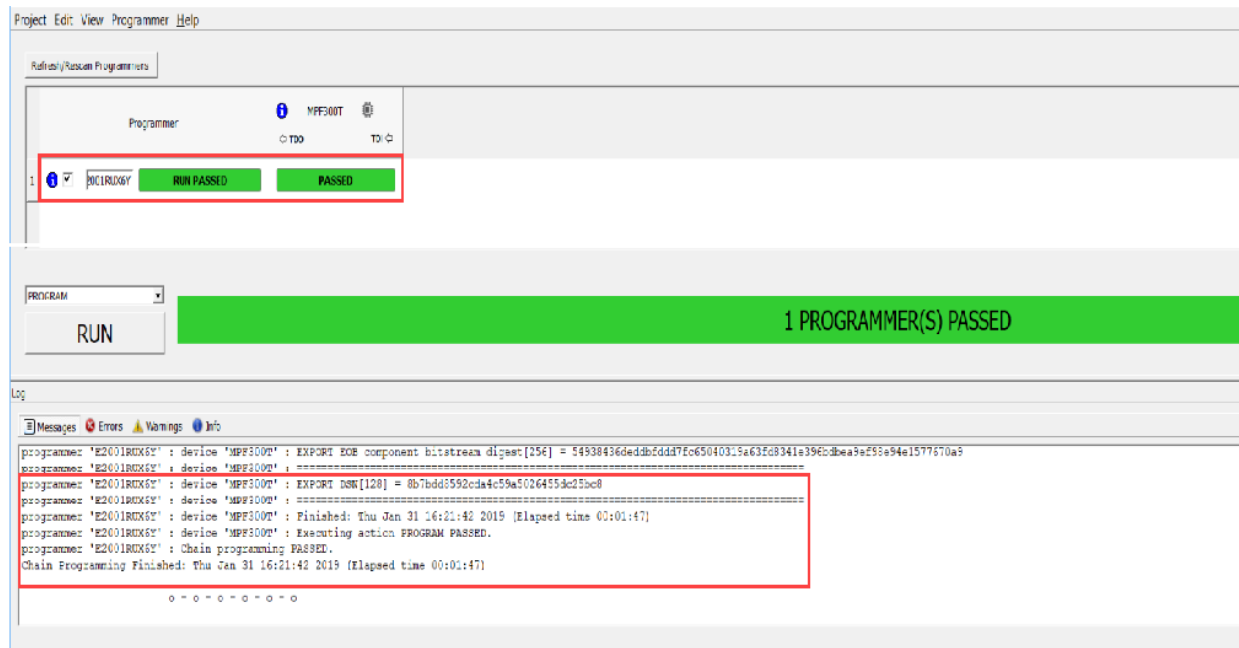
Figure 21 • New Job Project from FlashPro Express Job

8. Click **OK**. The required programming file is selected and ready to be programmed in the device.
9. 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 22 • Programming the Device

10. Click **RUN**. When the device is programmed successfully, a RUN PASSED status is displayed as shown in the following figure.

Figure 23 • FlashPro Express—RUN PASSED



11. 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_ac468_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.