DG0843
Demo Guide
PolarFire FPGA CPRI

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.

50200843. 4.0 3/21

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 4.0

Added Appendix 2: Running the TCL Script, page 28.

1.2 Revision 3.0

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 2.0

Updated the document for Libero SoC v12.0.

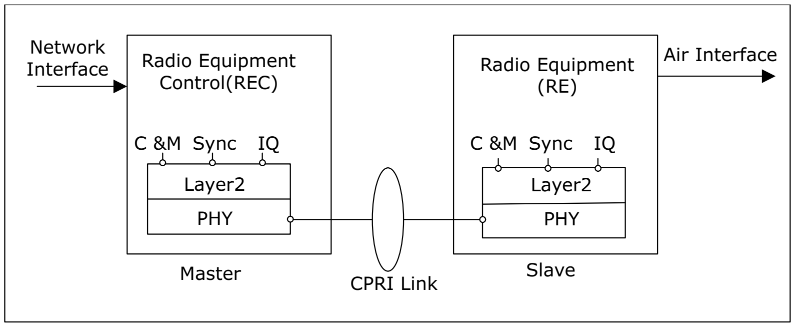
1.4 Revision 1.0

The first publication of this document.

2 Common Public Radio Interface Demo

The Common Public Radio Interface (CPRI) is an industry standard which defines the publicly available specification for the key internal interface of radio base stations between the Radio Equipment Control (REC) and the Radio Equipment (RE) as shown in following Figure 1, page 2. Microsemi provides the CPRI slave IP core that implements the transmitter and receiver interfaces of the CPRI standard. These IP cores are easy to integrate with CPRI-based data converters to develop high-bandwidth applications. Microsemi CPRI IP supports from line rate1 (614.4 Mbps) to line rate 7 (9830.4 Mbps). The transceiver supports link rate from 250 Mbps to 12.5 Gbps per lane. The transceiver supports up to line rate 9 for CPRI protocol. The transceiver (PF_XCVR) module integrates several functional blocks to support multiple high-speed serial protocols within the FPGA.

Figure 1 • **Radio Base Station System**



This demo design is created using the PolarFire high-speed transceiver blocks and CPRI Slave IP core. The design operates in the loopback mode by sending the CPRI Master data to the CPRI Slave IP core through the transceiver lanes, which are looped back on the board. This loopback setup facilitates a standalone CPRI interface demo that does not require CPRI testers and other devices to validate the design.

The demo design shows CPRI loopback using transceiver on the Evaluation board. The CPRI Slave IP is configured at:

- Line Rate 5: 4.9152 Gbps
- Number of Antenna Carriers: 4
- The Transceiver is configured in 8b10b mode running at 4.9152 Gbps data rate.
- 32-bit PCS fabric interface using a 122.88 MHz reference clock.

2.1 Design Requirements

The following table lists the hardware and software requirements for running the demo.

Table 1 • **Design Requirements**

Requirement	Version
Operating system	Windows 7, 8.1, 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT)	Rev D or later
Software	
Libero SoC	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	

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:

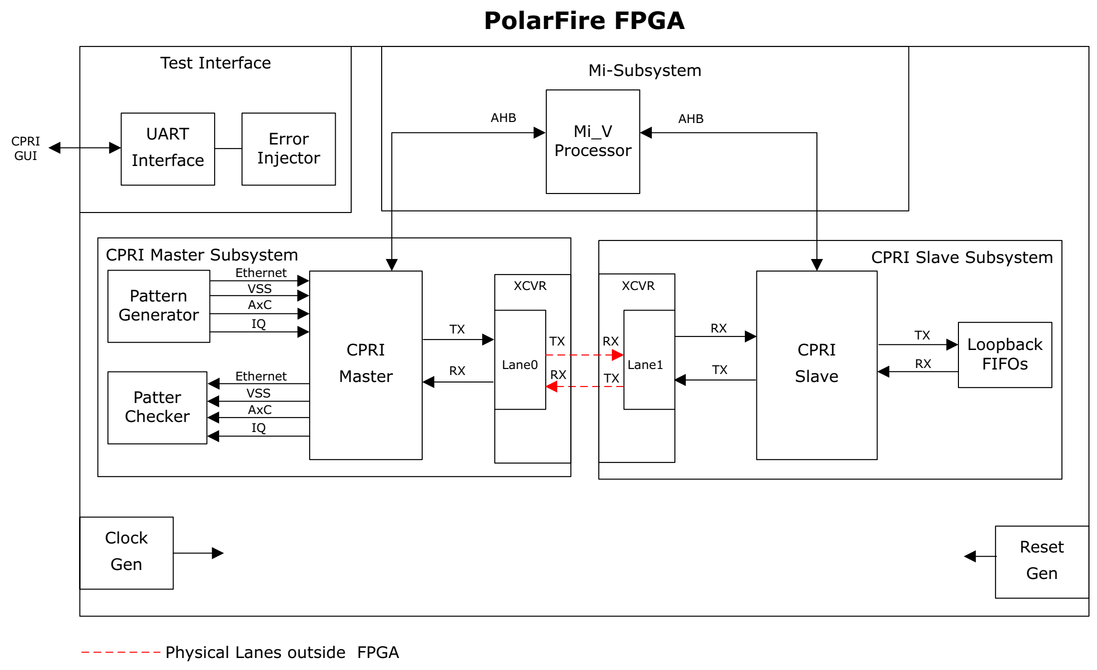
1. For demo design files download link:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0843_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads
3. Download and install the SoftConsole on the host PC from the following location:
   https://www.microsemi.com/product-directory/design-tools/4879-softconsole#downloads

2.3 Demo Design

The PolarFire CPRI Loopback demo design is developed for demonstrating CPRI IP in Slave mode. The pattern generator in this demo design generates CPRI protocol data such as IQ data, VSS data, Ethernet data, and Antenna carrier control data which is provided to the CPRI master module. The CPRI master module frames the data according to the CPRI protocol. The generated frames from CPRI master are loopbacked to CPRI slave IP using the XCVR. The CPRI slave IP unpacks the incoming data into IQ data, VSS data, Ethernet data, and Antenna carrier control data. This data is then loopbacked from CPRI slave IP to CPRI master module, therefore demonstrates Full duplex transmission of CPRI IP in Slave mode.

The following figure shows the top-level block diagram of the CPRI demo design.

Figure 2 • **CPRI Demo Design Block Diagram**



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

1. The Mi_V_module configures the registers of the CPRI Master module and CPRI slave IP blocks.
2. The reference design uses a transceiver interface (PF_XCVR_ERM) configured in 8b10b mode running at 4.9152 Gbps data rate, 32-bit PCS fabric interface, and 122.88 MHz reference clock.
3. The CPRI master module and CPRI IP are configured at Line Rate 5: 4.9152 Gbps with 4 Antenna Carriers.
4. The CPRI master module receives the IQ data from the TX IQ Data Generator and control information such as Ethernet data, Vendor specific data and antenna carrier control from the respective pattern generators.
5. CPRI master module then frames the incoming data and transmits a 32-bit CPRI frame to the transceiver.
6. The differential serial data of TX and RX is looped back using an on-board loopback.
7. The CPRI slave IP receives the incoming frames and segregates the incoming data according to the CPRI protocol.
8. The segregated IQ data, Vendor specific data, Ethernet data, and Antenna carrier control information are written to the respective loop back FIFO’s.
9. CPRI slave IP then reads the data from loopback FIFO's frames the data and transmits the a 32-bit CPRI frame to CPRI master module through the transceiver.
10. The CPRI master module receives the incoming frames, segregates and sends the incoming data to respective pattern checkers.
11. Incoming control information is compared with the control data for Vendor specific, Fast Ethernet, IQ data and Antenna carrier control data. When the data is matched, the respective lock signals are asserted.
12. The IQ data and control information such as Ethernet, Vendor specific and antenna carrier control lock signals from CPRI Slave IP and Master module along with transceiver RX_Valid and RX_ready for both the lanes are sent to the UART_interface block.
13. The UART_interface block forwards these status signal and locks information on its TX interface to the GUI for display.

2.3.1 Design Implementation

Each CPRI link connects two ports that have asymmetrical functions and roles: a master and a slave. The CPRI Demo design shows the CPRI point to point link between CPRI REC Master and CPRI RE Slave.

The Top-Level design includes the following SmartDesign components:

- CPRI Master Subsystem, page 5
- CPRI Slave Subsystem, page 7
- Transceiver Subsystem, page 7
- Mi-V Subsystem, page 8
- Test Interface, page 9

2.3.1.1 CPRI Master Subsystem

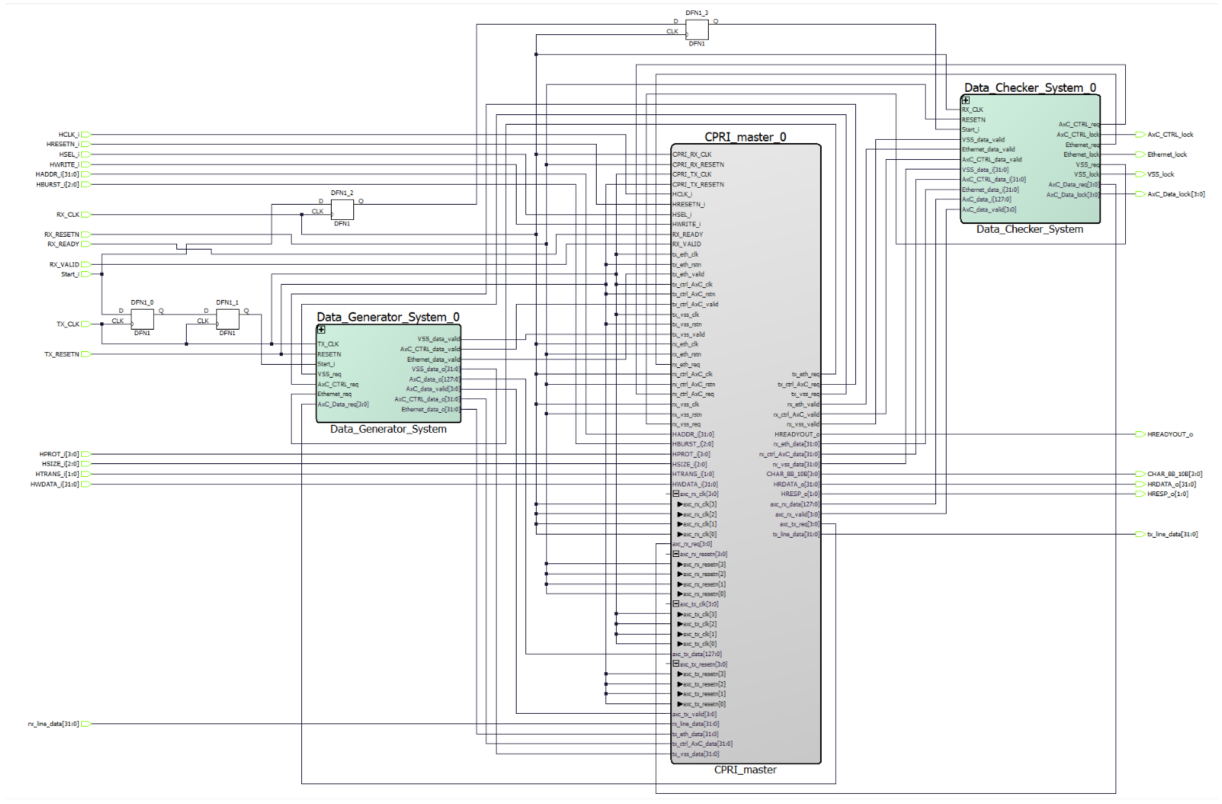
This sections gives the brief description about the CPRI REC Master.

The CPRI REC Master subsystem consists of the following modules.

- CPRI Master
- Pattern data generator
- Pattern checker modules

The following figure shows the implementation of CPRI master subsystem SmartDesign.

Figure 3 • **CPRI Master Subsystem SmartDesign**



2.3.1.1.1 CPRI Master

When the CPRI master is configured, it receives IQ data, Vendor specific data, Antenna carrier control data and Ethernet data from the respective pattern generator blocks. It then frames incoming data into the CPRI frame format at the rate of 4.9152 Gbps and transmits it to the serial lines through the XCVR interface.

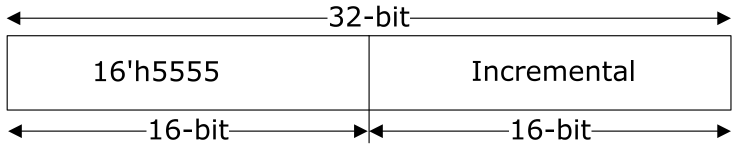
On the receive side, the CPRI master receives the data from the serial lines through the XCVR interface in the CPRI frame format, it then unpacks the data and sends the received IQ, VSS, and Ethernet data to the respective pattern checker block. For more information about register configuration, see Register Configuration, page 11.

2.3.1.1.2 Pattern Data Generator

The pattern generator generates the following data:

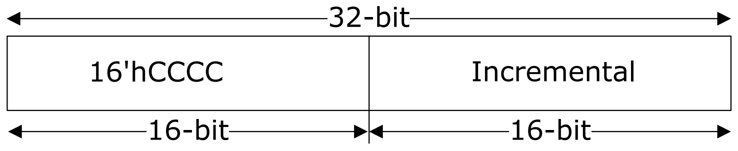
- 32-bit Ethernet data. The MSB 16-bit are constant and LSB 16-bits are incremental data.

Figure 4 • **Ethernet Data Pattern**



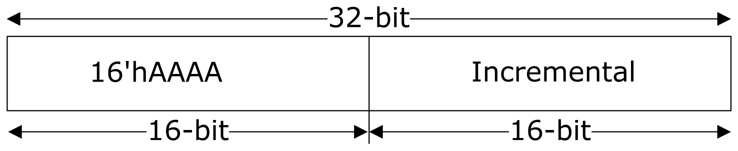
- 32-bit VSS data. The MSB 16-bit are constant and the LSB 16-bits are incremental data.

Figure 5 • **VSS Data Pattern**



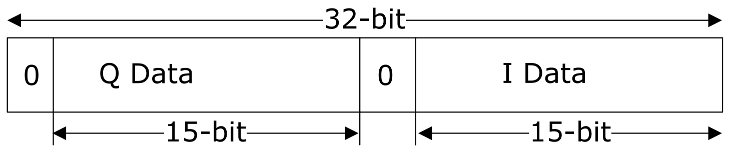
- 32-bit Antenna Carrier Control data. The MSB 16-bit are constant and the LSB 16-bits are incremental data.

Figure 6 • **Antenna Carrier Data Pattern**



- In this demo four antenna carriers are used, each antenna generates 32-bit data. The IQ data samples of each Antenna carrier control are fixed at 15-bits. The user-interface width is fixed at 32-bit, with zero appending on MSB. 15-bit IQ data samples are bit-stuffed in the 32-bit user interface for each antenna carrier.

Figure 7 • **IQ Data Pattern**



2.3.1.1.3 Pattern Checker

The pattern checker has individual pattern checker modules for VSS, Ethernet, Antenna carrier control, and IQ data, which checks for incremental, data and if the received data is matched, then the respective lock signals are asserted.

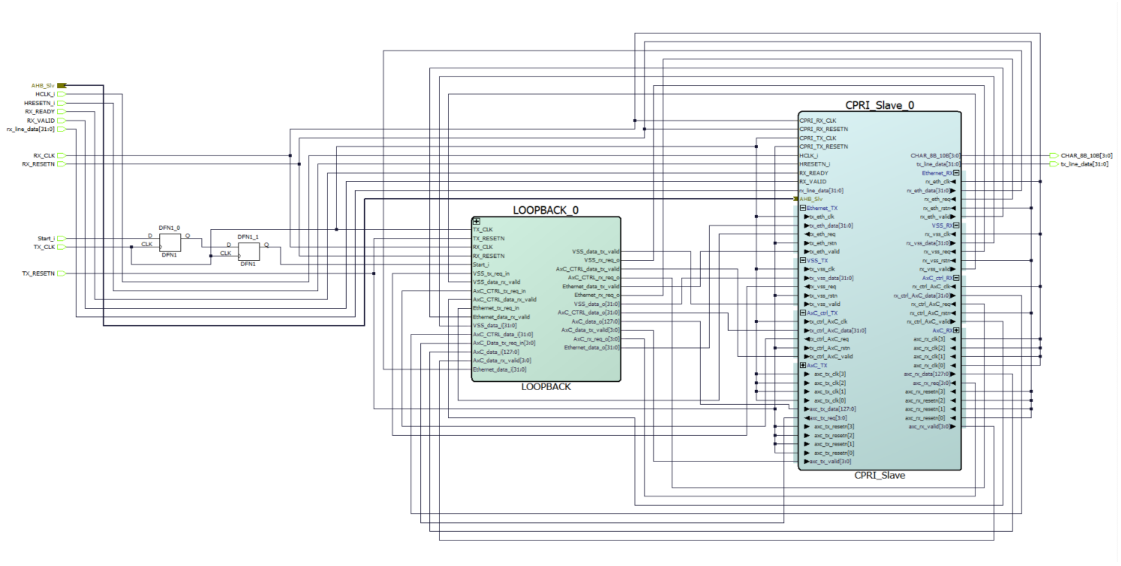
2.3.1.2 CPRI Slave Subsystem

The CPRI RE slave subsystem consists of the following modules:

- CPRI Slave
- Loopback FIFO

The CPRI Slave Subsystem SmartDesign implements CPRI Slave IP interface along with Ethernet, VSS, Antenna carrier control, and IQ data Loopback FIFO's, as shown in following figure.

Figure 8 • **CPRI Slave System**



2.3.1.2.1 CPRI Slave

CPRI slave IP receives the data from the serial lines through the XCVR interface in the CPRI frame format. It then unpacks the data and sends the received IQ, VSS, Antenna carrier control, and Ethernet data to the respective Loopback FIFO's. On the transmit side, CPRI slave IP receives IQ, VSS, Antenna carrier control, and Ethernet data from the respective Loopback FIFO's and then frames incoming data into the CPRI frame format at the rate of 4.9152 Gbps and transmits it to the serial lines through transceiver interface. For more information on CPRI register configuration, see Register Configuration, page 11.

For more information about CPRI IP, see **Libero Catalog** -> **Solution-Wireless**->***Microsemi CPRI User Guide***.

2.3.1.2.2 Loopback FIFO

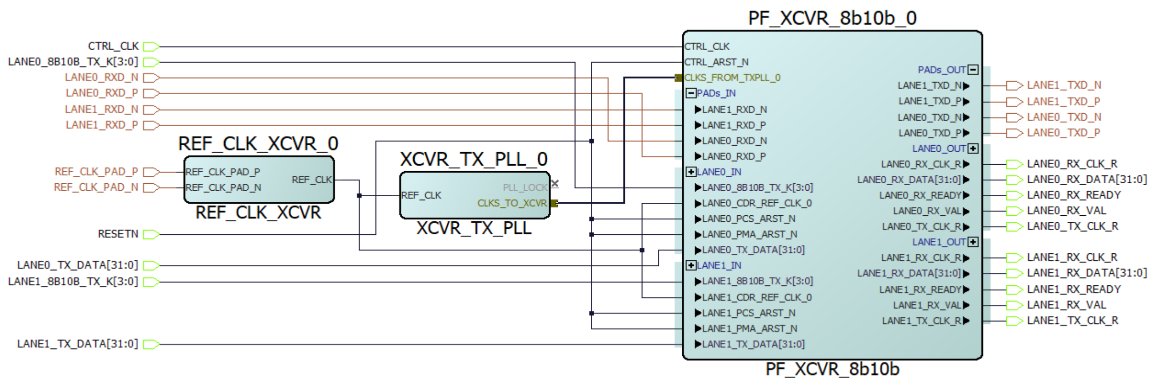
The Loopback FIFO module consists of individual loopback FIFO modules for VSS, Ethernet, Antenna carrier control, and IQ data, which loopback's the deframed data from the RX interface of CPRI IP to TX interface of CPRI IP.

2.3.1.3 Transceiver Subsystem

The transceiver is configured for:

- Two lanes at 122.88 MHz clock- each lane carries 32-bit IQ data
- 32-bit PCS interface
- Data rate of 4.9152 Gbps
- Lane0 is configured for Master and Lane1 is configured for Slave.

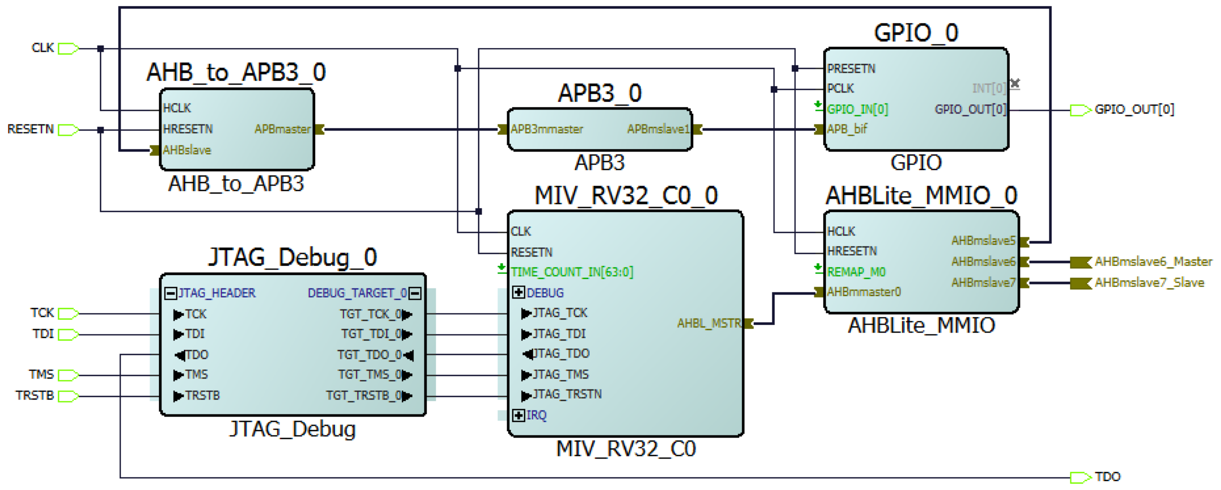
Figure 9 • **Transceiver Subsystem**



2.3.1.4 Mi-V Subsystem

The Mi_V_Subsystem SmartDesign operates at 100 MHz. It implements an AHB interface and GPIO interface. The AHB interface is used to access the CPRI configuration registers, and the GPIO interface is used to indicate configuration done status, as shown in the following figure.

Figure 10 • **Mi-V Subsystem**



For more information about how to build the Mi-V subsystem, see *TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial*.

The address map of Mi-V processor is described in the following table.

Table 2 • **System Memory Map**

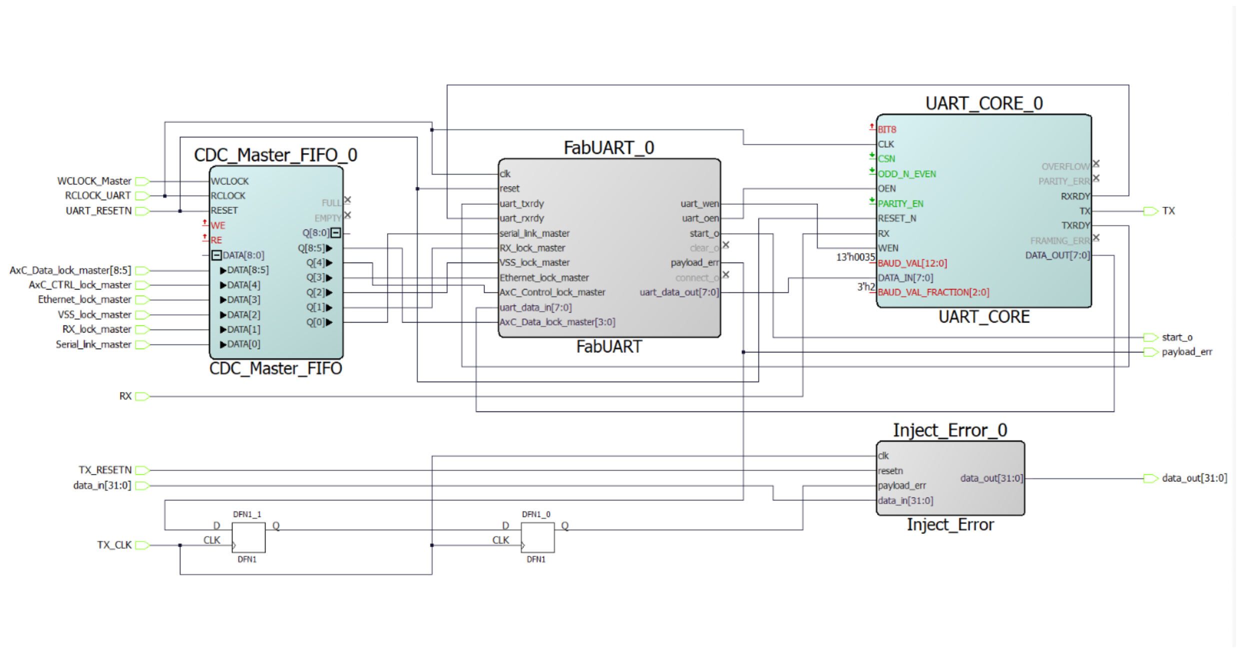
Component	Memory Map	Description
CoreGPIO	0x60051000	This bus interface is used to access the GPIO's through APB interface.
CPRI Slave IP	0x60060000	This bus interface is used to access CPRI Slave IP configuration registers through the AHB interface.
CPRI Master module	0x60070000	This bus interface is used to access CPRI Master configuration registers through the AHB interface.

2.3.1.5 Test Interface

The test interface system contains the UART interface and Inject error module.

The following figure shows the test interface SmartDesign implementations.

Figure 11 • **Test Interface SmartDesign**



2.3.1.5.1 UART Interface

The UART_Interface SmartDesign interfaces Fabric UART logic with the GUI, which displays the lock signal.

2.3.1.5.2 Inject Error

The Inject error module induces an error in the CPRI frame which propagates through the serial interface. The presence of an error is validated using a pattern checker module.

2.4 Port Description

The following table describes the input and output ports of the design.

Table 3 • **Port Description**

Signal	Direction	Description
RESETN	Input	External reset
REF_CLK_PAD_P_0 and REF_CLK_PAD_N_01	Input	This is the differential reference clock generated from the on-board 122.88 MHz oscillator.
LANE0_RXD_P and LANE0_RXD_N	Input	Transceiver Receiver differential input of Lane0
LANE1_RXD_P and LANE1_RXD_N	Input	Transceiver Receiver differential input Lane1
RX	Input	This is the input signal received by the UART interface from the GUI.
LANE0_TXD_P and LANE0_TXD_N	Output	Transceiver Receiver differential output of Lane0
LANE1_TXD_P and LANE1_TXD_N	Output	Transceiver Receiver differential output of Lane1
Config_done	Output	Indicates CPRI master and slave register configuration is completed by Mi-V processor
VSS_lock	Output	Indicates the Received Vendor specific data is correct
Ethernet_lock	Output	Indicates the Received Ethernet data is correct
AxC_Control_lock	Output	Indicates the received Antenna Control data is correct
AxC_Data_lock	Output	Indicates the received Antenna IQ data is correct
TX	Output	This is the output data received by the GUI from the UART interface.

2.5 Register Configuration

The CPRI Master module and CPRI Slave IP are connected to the Mi-V processor using an AHB interface. The base address of CPRI master is `0x60070000` and the base address of CPRI Slave IP is `0x60060000`.

Note: It is recommended to follow the order of register configuration as described in the Table 4, page 11.

The following table describes register configuration and these registers are configured in the SoftConsole project: `mpf_dg0843_df\SoftConsole\CPRI_config\main.c`

Table 4 • **Register Configuration**

Offset	Value	Description
0x04	0x1F	TX Interface Sequence Number Register Line Rate 5: TX_SEQ_NUM: 0x1f
0x08	0x1F	RX Interface Sequence Number Register Line Rate 5: RX_SEQ_NUM: 0x1f
0x0C	0xBC	TX Framer K character Register TX_Z_0_0_SYNC: 0xBC which is K28.5 character
0x10	0xBC	RX Framer K character Register RX_Z_0_0_SYNC: 0xBC which is K28.5 character
0x14	0x50	TX Framer D character Register TX_Z_0_1_SYNC: 0x50 which is D16.2 character
0x18	0x50	RX Framer D character Register RX_Z_0_1_SYNC: 0x50 which is D16.2 character
0x1C	0x50	TX Framer D character Register TX_Z_0_Y_SYNC: 0x50 which is D16.2 character
0x20	0x50	RX Framer D character Register RX_Z_0_Y_SYNC: 0x50 which is D16.2 character
0x2C	0x14	TX Fast Ethernet pointer register TX_Z_194_0: 0x14 the Fast Ethernet data to be written from this location
0x24	0x01	TX Protocol Version Register TX_Z_2_0_SYNC: 0x01, the current version of the IP supports only protocol version 1
0x30	0x14	RX Fast Ethernet pointer register RX_Z_194_0: 0x14 the Fast Ethernet data to be written from this location
0x100	0xF4240	Start Timer Register Time out value for L1 Sync state machine
0x104	0x1FFF	HFN Sync Counter Register Line 5: HFN_COUNT: 1FFF This value indicates the duration of one CPRI Hyperframe for the selected Line rate
0x00	0xF	CPRI Control Register Enables IQ mapper, demapper, and L1 sync state machine and CPRI IP functionality

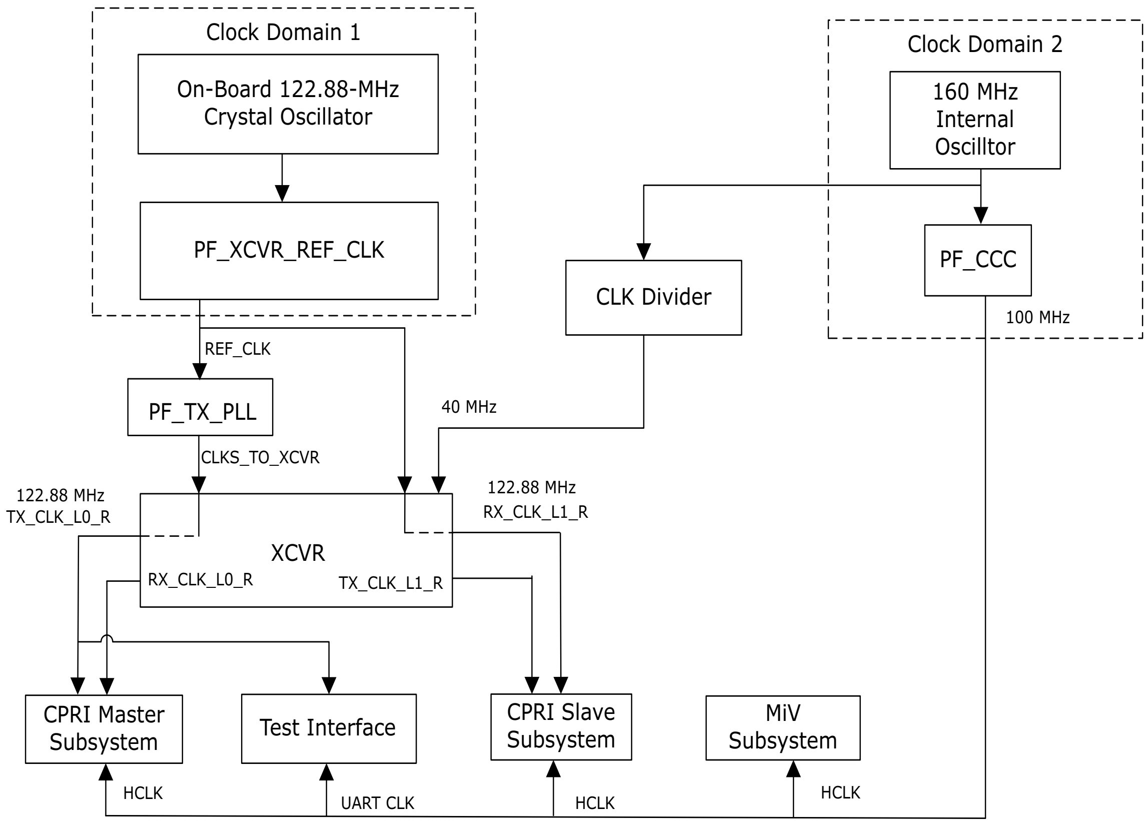
2.6 Clocking Structure

In this demo design, there are two clock domains. The on-board 122.88 MHz crystal oscillator drives the XCVR reference clock in 8b10b mode. This generates Lane0/1 RX clock and Lane0/1 TX clock. CPRI Master system, CPRI Slave Subsystem, and HDL modules use TX and RX clock (122.88 MHz).

The on-chip 160 MHz RC oscillator drives the CCC which generates 100 MHz clock. The UART_Interface, Mi-V system and the AHB interface of the CPRI Master Subsystem and CPRI Slave Subsystem use 100 MHz clock. The clock divider generates 40 MHz clock for the XCVR_ERM.

The following figure shows the clocking structure in the reference design.

Figure 12 • **Clocking Structure**



The following tables describes the clocks used in the demo design.

Table 5 • **Clocks**

Clock Name	Source	Frequency
Mi-V Clock	CCC_0	100 MHz
RX_CLK_L0_R	Transceiver RX recovered clock (Lane0)	122.88 MHz
RX_CLK_L1_R	Transceiver RX recovered clock (Lane1)	122.88 MHz
TX_CLK_L0_R	Transceiver TX PLL clock (Lane0)	122.88 MHz
TX_CLK_L1_R	Transceiver TX PLL clock (Lane1)	122.88 MHz
CTRL_CLK	CLK Divider	40 MHz

2.7 Reset Structure

In the demo design, the reset signal is generated using the Reset_Block module. CoreReset_FF (CoreReset_PF) module releases active low reset signal of TEST_INTERFACE, Mi-V subsystem and CPRI Master and Slave subsystem, when PLL_lock output from PF_CCC block, RESETN (External active low signal) and DEVICE_INIT_DONE signal from INIT_MONITOR block are asserted.

The CoreReset_L0_TX (CoreReset_PF) module releases an active low reset signal of the CPRI Master subsystem and Test Interface when RESETN (External active low signal) and DEVICE_INIT_DONE signal from INIT_MONITOR block are asserted.

The CoreReset_L0_RX (CoreReset_PF) module releases an active low reset signals of the CPRI Master subsystem when RESETN (External active low signal) and DEVICE_INIT_DONE signal from INIT_MONITOR block are asserted.

The CoreReset_L1_TX (CoreReset_PF) module releases an active low reset signal of the CPRI Slave subsystem when RESETN (External active low signal) and DEVICE_INIT_DONE signal from INIT_MONITOR block are asserted.

The CoreReset_L1_RX (CoreReset_PF) module releases an active low reset signal of the CPRI Slaven subsystem when RESETN (External active low signal) and DEVICE_INIT_DONE signal from INIT_MONITOR block is asserted.

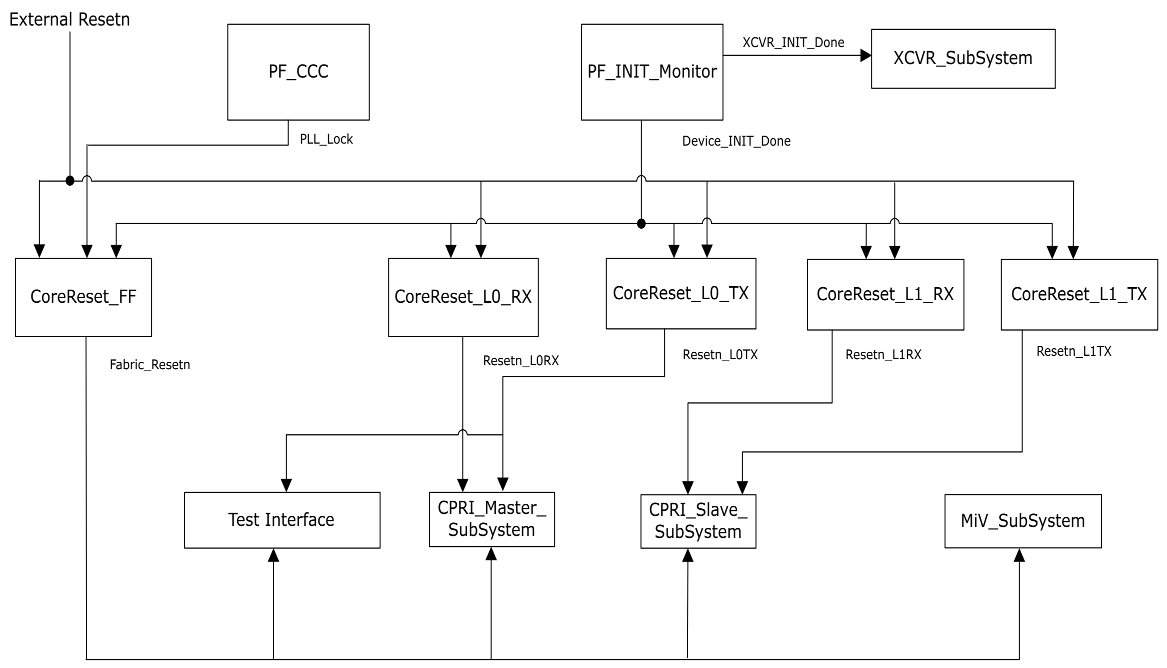
The INIT_MONITOR releases the active low signal of XCVR_subsystem reset signals (PMA_ARST_N and PCS_ARST_N) when XCVR_INIT_DONE signal from INIT_MONITOR block is asserted.

DEVICE_INIT_DONE and XCVR_INIT_DONE signals are 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 the demo design.

Figure 13 • **Reset Structure**



3 Libero Design Flow

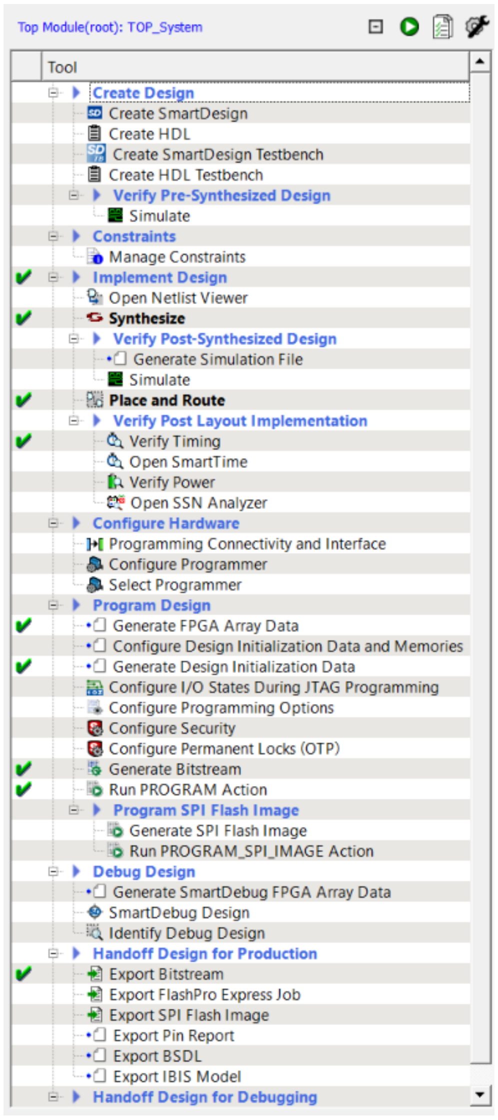
This chapter describes the Libero design flow, which involves the following processes:

- Synthesis, page 16
- Place and Route, page 17
- Verify Timing, page 18
- Design and Memory Initialization, page 18
- Generate Bitstream, page 20
- Run Program Action, page 21

Note: To initialize the TCM in PolarFire using the system controller, a local parameter **l_cfg_hard_tcm0_en**, in the `miv_rv32_opsrv_cfg_pkg.v` file should be changed to 1’b1 prior to synthesis. See the 2.7 TCM section in the *MIV_RV32 Handbook*.

The following figure shows these options in the Design Flow window.

Figure 14 • **Libero Design Flow Options**



3.1 Synthesis

To synthesis the design, perform the following step:

1. On the **Design Flow** window, and double-click **Synthesize**.

When the synthesis is successful, a green tick mark appears next to **Synthesize**, as shown in Figure 14, page 15.

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

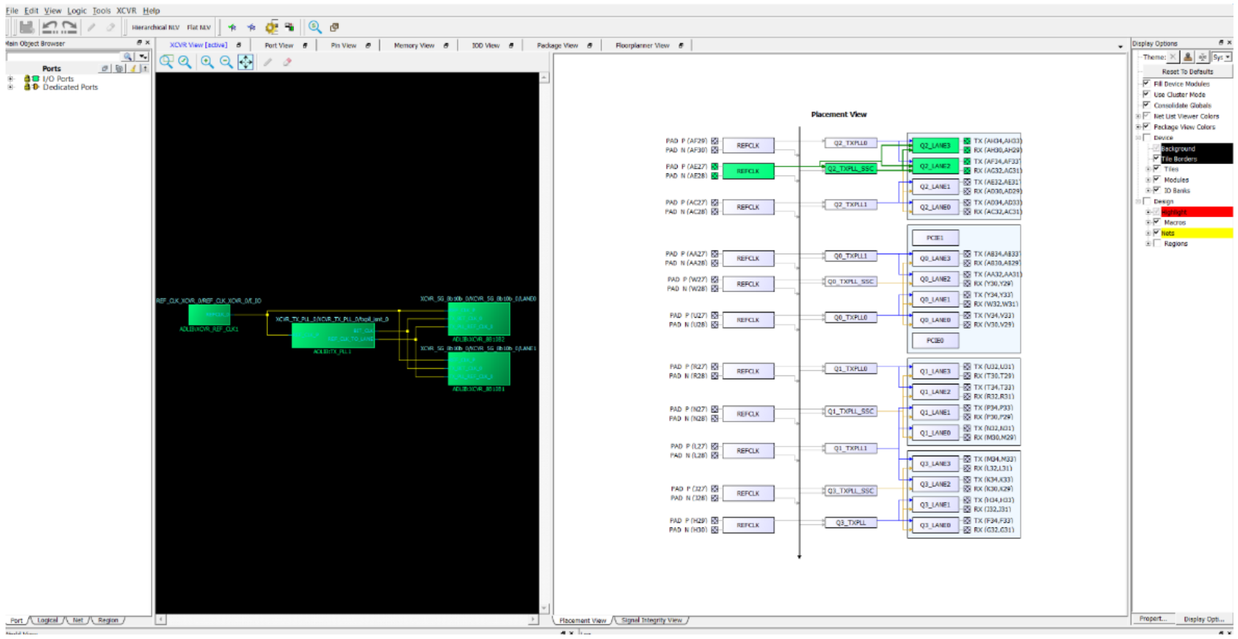
Table 6 • **Resource Utilization—Synthesis**

Type	Used	Total	Percentage
4LUT	31871	299544	10.64
DFF	20214	299544	6.75
I/O	0	1536	0.00
User I/O	11	512	2.15
-Single-ended I/O	11	512	2.15
-Differential-ended I/O	0	256	0.00
µSRAM	12	2772	0.43
LSRAM	97	952	10.19
Math	2	924	0.22
H-Chip Global	10	48	20.83
Local Global	4	1008	0.40
PLL	1	8	12.50
DLL	0	8	0.00
CRN_INT	1	24	4.17
UJTAG	1	1	100.00
INIT	1	1	100.00
OSC_RC1600 MHz	1	1	100.00
Transceiver Lanes	2	16	12.50
Transceiver PCIe	0	2	0.00
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09
ICB_CLKINT	4	72	5.56
ICB_CLKDIV	1	24	4.17
ICB_INT	1	12	8.33

3.2 Place and Route

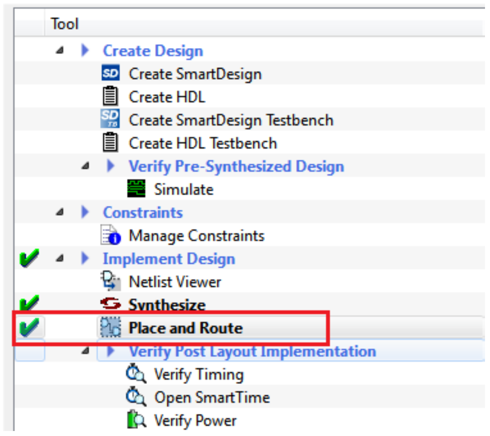
To place and route the design, TX_PLL, XCVR_REF_CLK, and PF_XCVR, must be configured using the I/O Editor. For On-board transceiver loop back, the Lane2 and Lane3 of Quard2 are used, as shown in the following figure.

Figure 15 • **I/O Editor Option—XCVR View**



1. On the **Design Flow** window, double-click **Place and Route**. When place and route is successful, a green tick mark appears as shown in the following figure.

Figure 16 • **Place and Route**



The following table lists the resource utilization after place and route.

Table 7 • **Resource Utilization—Place and Route**

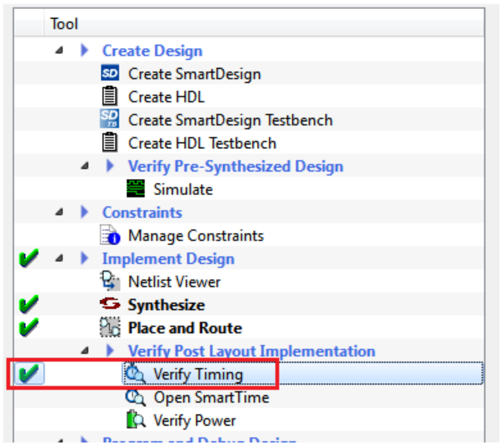
Type	Used	Total	Percentage
4LUT	32019	299544	10.69
DFF	20284	299544	6.77
I/O Register	0	510	0.00
Logic Elements	38062	299544	12.71

3.3 Verify Timing

To verify timing, perform the following steps:

1. On the **Design Flow** window, double-click **Verify Timing**.
   When the design meets the timing requirements, a green tick mark appears next to **Verify Timing**.
2. Right-click **Verify Timing** and select **View Report** to view the verify timing report and log files in the **Reports** tab.

Figure 17 • **Verify Timing**



3.4 Design and Memory Initialization

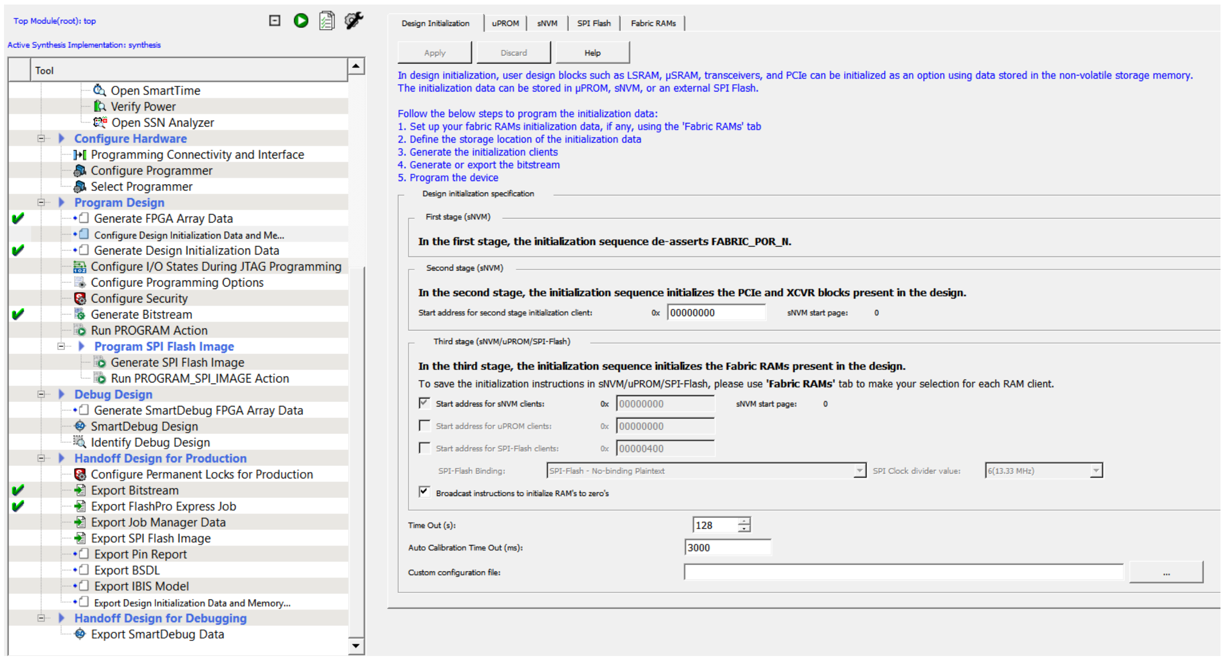
The **Configure Design Initialization Data and Memories** generates the LSRAM initialization client and adds it to sNVM, µPROM, or an external SPI flash, based on the type of non-volatile memory selected. In this demo, the LSRAM initialization client is stored in the sNVM.

This process requires the user application executable file (hex file) to initialize the LSRAM blocks on device power-up. The hex file (CPRI_config.hex) is available in the `DesignFiles_Directory\mpf_dg0843_df\Libero_Project` folder. When the hex file is imported, a memory initialization client is generated for LSRAM blocks.

To initialize the memory, perform the following steps:

1. On the Design Flow window, double-click **Configure Design Initialization Data and Memories.** The Design and Memory Initialization window opens as shown in the following figure.

Figure 18 • **Design and Memory Initialization Window**



In design initialization, user design blocks such as LSRAM, µSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in µPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:
1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC_POR_N.

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000 sNVM start page: 0

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use **'Fabric RAMs'** tab to make your selection for each RAM client.

[✓] Start address for sNVM clients: 0x 00000000 sNVM start page: 0
[ ] Start address for uPROM clients: 0x 00000000
[ ] Start address for SPI-Flash clients: 0x 00000400

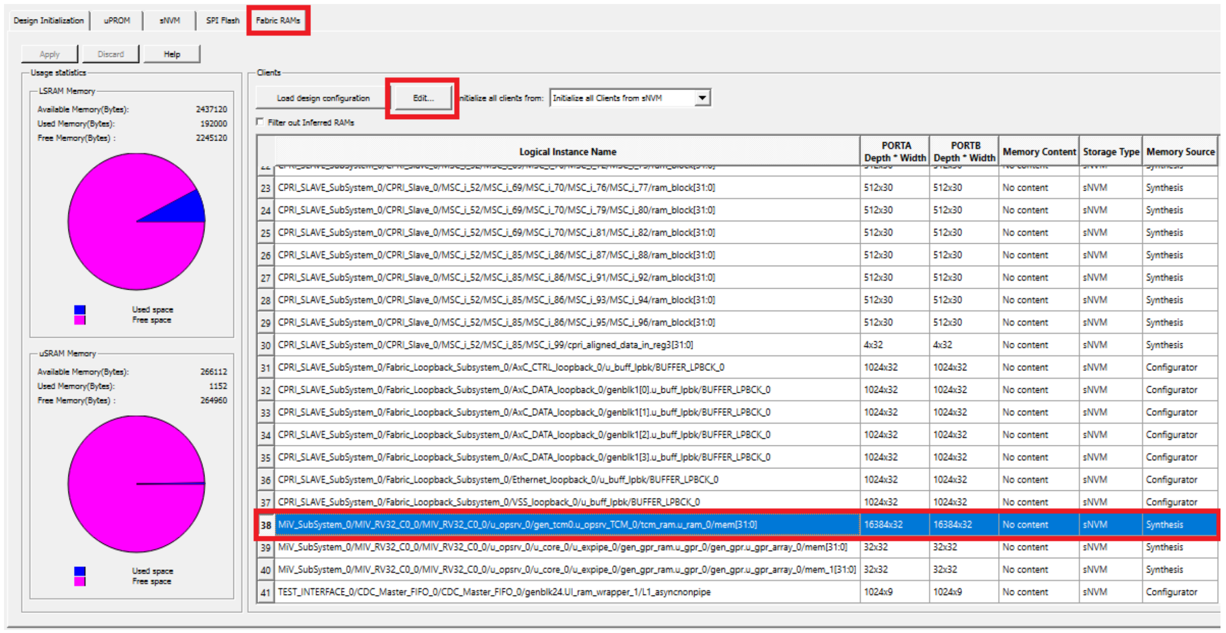
SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 6(13.33 MHz)

[✓] Broadcast instructions to initialize RAM's to zero's

Time Out (s): 128
Auto Calibration Time Out (ms): 3000
Custom configuration file:

2. In the **Fabric RAMs** tab, select the LSRAM client from the list, and then click **Edit,** as shown in the following figure.

Figure 19 • **Fabric RAM**



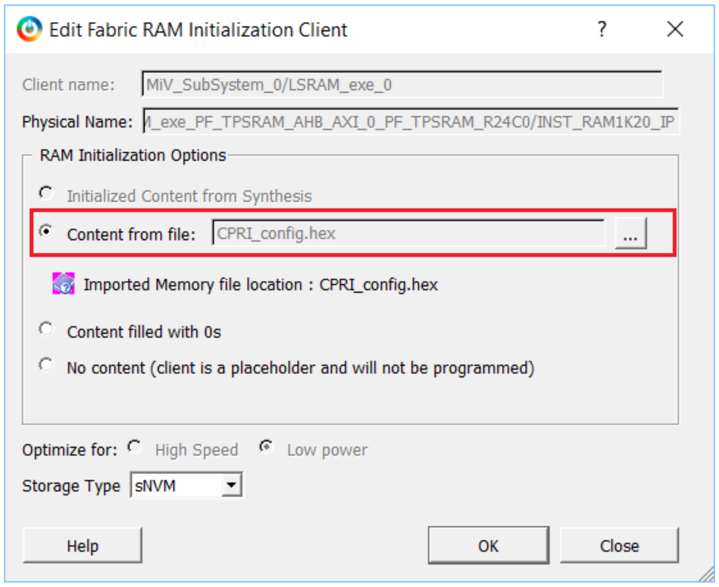
LSRAM Memory
Available Memory(Bytes): 2437120
Used Memory(Bytes): 192000
Free Memory(Bytes) : 2245120

uSRAM Memory
Available Memory(Bytes): 266112
Used Memory(Bytes): 1152
Free Memory(Bytes) : 264960

	Logical Instance Name	PORTA Depth * Width	PORTB Depth * Width	Memory Content	Storage Type	Memory Source
23	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_76/MSC_i_77/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
24	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_79/MSC_i_80/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
25	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_81/MSC_i_82/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
26	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_87/MSC_i_88/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
27	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_91/MSC_i_92/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
28	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_93/MSC_i_94/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
29	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_95/MSC_i_96/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
30	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_99/cpri_aligned_data_in_reg3[31:0]	4x32	4x32	No content	sNVM	Synthesis
31	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_CTRL_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
32	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[0].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
33	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[1].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
34	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[2].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
35	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[3].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
36	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/Ethernet_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
37	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/VSS_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
38	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/gen_tcm0.u_opsrv_TCM_0/tcm_ram.u_ram_0/mem[31:0]	16384x32	16384x32	No content	sNVM	Synthesis
39	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/u_core_0/u_expipe_0/gen_gpr_ram.u_gpr_0/gen_gpr.u_gpr_array_0/mem[31:0]	32x32	32x32	No content	sNVM	Synthesis
40	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/u_core_0/u_expipe_0/gen_gpr_ram.u_gpr_0/gen_gpr.u_gpr_array_0/mem_1[31:0]	32x32	32x32	No content	sNVM	Synthesis
41	TEST_INTERFACE_0/CDC_Master_FIFO_0/CDC_Master_FIFO_0/genblk24.UI_ram_wrapper_1/L1_asyncnonpipe	1024x9	1024x9	No content	sNVM	Configurator

3. In the **Edit Fabric RAM Initialization Client** dialog box, select the **Content from file** option, and locate the CPRI_config.hex file from `DesignFiles_directory\mpf_dg0843_df\Libero_Project` folder and then click **OK,** as shown in the following figure.

Figure 20 • Edit Fabric RAM Initialization Client



Edit Fabric RAM Initialization Client

Client name: MiV_SubSystem_0/LSRAM_exe_0

Physical Name: /I_exe_PF_TPSRAM_AHB_AXI_0_PF_TPSRAM_R24C0/INST_RAM1K20_IP

RAM Initialization Options

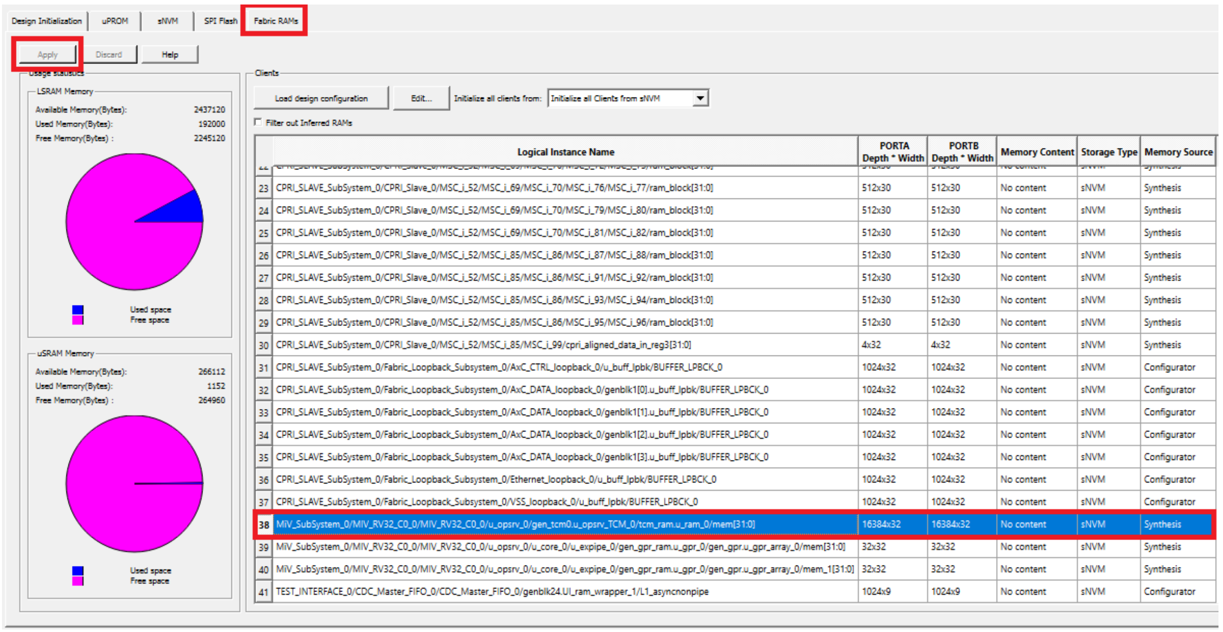
- Initialized Content from Synthesis
- [✓ Content from file:] CPRI_config.hex
- Imported Memory file location : CPRI_config.hex
- Content filled with 0s
- No content (client is a placeholder and will not be programmed)

Optimize for: High Speed [✓ Low power]

Storage Type: sNVM

4. Click **Apply** as shown in the following figure.

Figure 21 • Design Initialization



Usage statistics

LSRAM Memory

Available Memory(Bytes):	2437120
Used Memory(Bytes):	192000
Free Memory(Bytes) :	2245120

🟦 Used space
🟪 Free space

uSRAM Memory

Available Memory(Bytes):	266112
Used Memory(Bytes):	1152
Free Memory(Bytes) :	264960

🟦 Used space
🟪 Free space

Clients

Initialize all clients from: Initialize all Clients from sNVM

Filter out Inferred RAMs

	Logical Instance Name	PORTA Depth * Width	PORTB Depth * Width	Memory Content	Storage Type	Memory Source
23	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_76/MSC_i_77/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
24	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_79/MSC_i_80/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
25	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_69/MSC_i_70/MSC_i_81/MSC_i_82/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
26	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_87/MSC_i_88/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
27	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_91/MSC_i_92/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
28	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_93/MSC_i_94/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
29	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_86/MSC_i_95/MSC_i_96/ram_block[31:0]	512x30	512x30	No content	sNVM	Synthesis
30	CPRI_SLAVE_SubSystem_0/CPRI_Slave_0/MSC_i_52/MSC_i_85/MSC_i_99/cpri_aligned_data_in_reg3[31:0]	4x32	4x32	No content	sNVM	Synthesis
31	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_CTRL_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
32	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[0].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
33	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[1].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
34	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[2].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
35	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/AxC_DATA_loopback_0/genblk1[3].u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
36	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/Ethernet_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
37	CPRI_SLAVE_SubSystem_0/Fabric_Loopback_Subsystem_0/VSS_loopback_0/u_buff_lpbk/BUFFER_LPBCK_0	1024x32	1024x32	No content	sNVM	Configurator
38	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/gen_tcm0.u_opsrv_TCM_0/tcm_ram.u_ram_0/mem[31:0]	16384x32	16384x32	No content	sNVM	Synthesis
39	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/u_core_0/u_expipe_0/gen_gpr_ram.u_gpr_0/gen_gpr.u_gpr_array_0/mem[31:0]	32x32	32x32	No content	sNVM	Synthesis
40	MiV_SubSystem_0/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/u_core_0/u_expipe_0/gen_gpr_ram.u_gpr_0/gen_gpr.u_gpr_array_0/mem_1[31:0]	32x32	32x32	No content	sNVM	Synthesis
41	TEST_INTERFACE_0/CDC_Master_FIFO_0/CDC_Master_FIFO_0/genblk24.UI_ram_wrapper_1/L1_asyncnonpipe	1024x9	1024x9	No content	sNVM	Configurator

5. To generate design initialization data, click **Generate Initialization Data** on the Design Flow window. After successful generation of the Initialization data, a green tick mark appears next to Generate Initialization Data option as shown in the Figure 21, page 20.

3.5 Generate Bitstream

To generate bitstream, perform the following steps:

1. On the Design Flow window, double-click **Generate Bitstream**.

   When the bitstream is successfully generated, a green tick mark appears next to **Generate Bitstream**, as shown in Figure 14, page 15.

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

3.6 Run Program Action

To program the PolarFire device, perform the following steps.

1. Ensure that the jumper settings on the board are as listed in the following table.

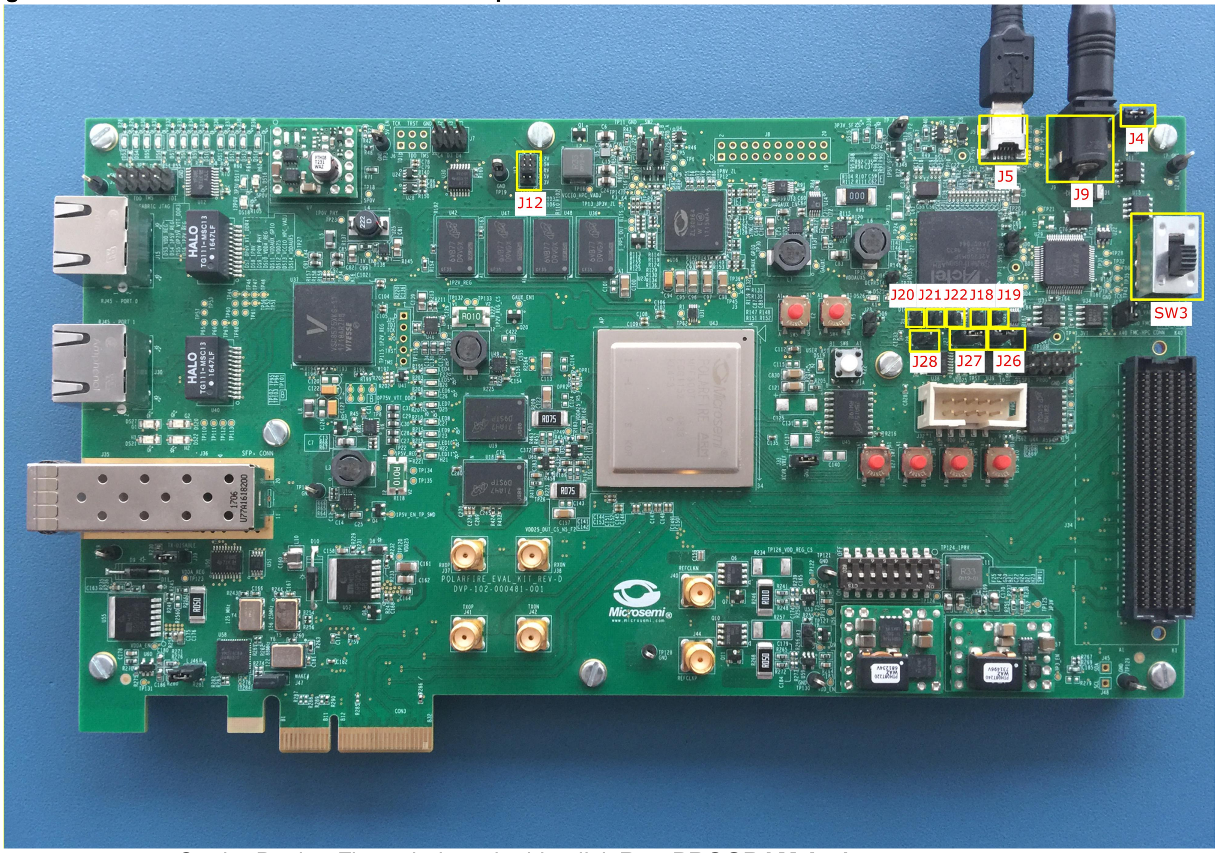
Table 8 • **PolarFire Evaluation Board Jumper Settings**

Jumper	Description
J18, J19, J20, J21, and J22	Short pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Short the pin 2 and 3 for programming through the on-board FlashPro5
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 side. Open pin 1 and 2 for routing 122.88 MHz differential clock oscillator to the line side.

2. Connect the power supply cable to the J9 connector.
3. Connect the USB cable from the host PC to the J5 (FTDI port).
4. Power on the board using the SW3 slide switch.

The following figure shows the board setup.

Figure 22 • **PolarFire Evaluation Board Setup**



5. On the Design Flow window, double-click **Run PROGRAM Action.**

The device is successfully programmed and a green tick mark appears next to **Run PROGRAM Action** as shown in Figure 14, page 15. LED 4 is asserted once the device is programmed.

4 Running the Demo

This section describes how to install and use the CPRI Demo application Graphic User Interface (GUI). The PolarFire CPRI demo application is a simple GUI that runs on the host PC to communicate with the PolarFire Device.

Installing CPRI Demo Application

To install CPRI demo application, perform the following steps:

1. Install the GUI (setup.exe) from the following design files folder `mpf_dg0843_df/GUI`

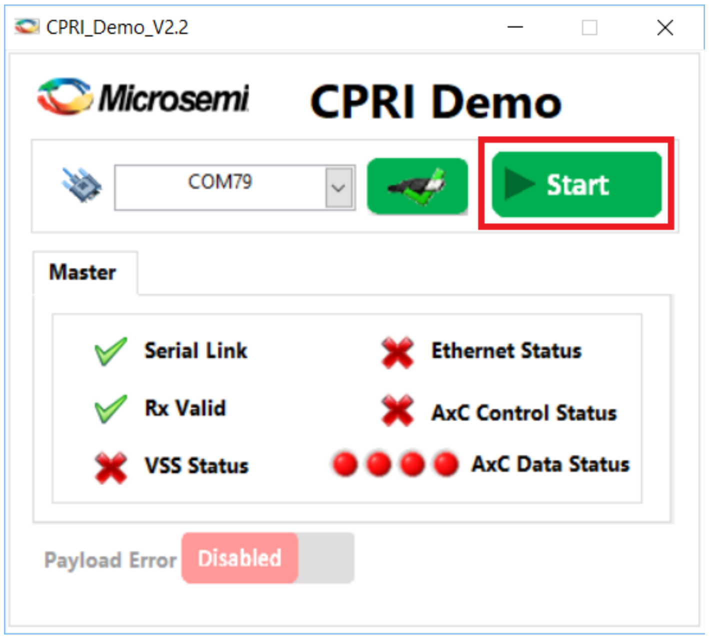
The following steps describes how to install and use the GUI to run the CPRI Loopback demo. The following procedure assumes that:

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

To run the CPRI Loopback demo, perform the following steps:

1. Extract the contents of the `mpf_dg0843_df.zip file`.
2. From the `mpf_dg0843_df\GUI_Installer` folder, double-click the setup.exe file.
3. Follow the instructions displayed on the installation wizard.
4. After successful installation, CPRI_GUI appears on the **Start** menu of the host PC desktop.
5. From the **Start** menu, click **CPRI_GUI** to launch the application.
6. The GUI detects the COM port number and automatically connects to the board, as shown in the following figure. COM Port numbers may vary.

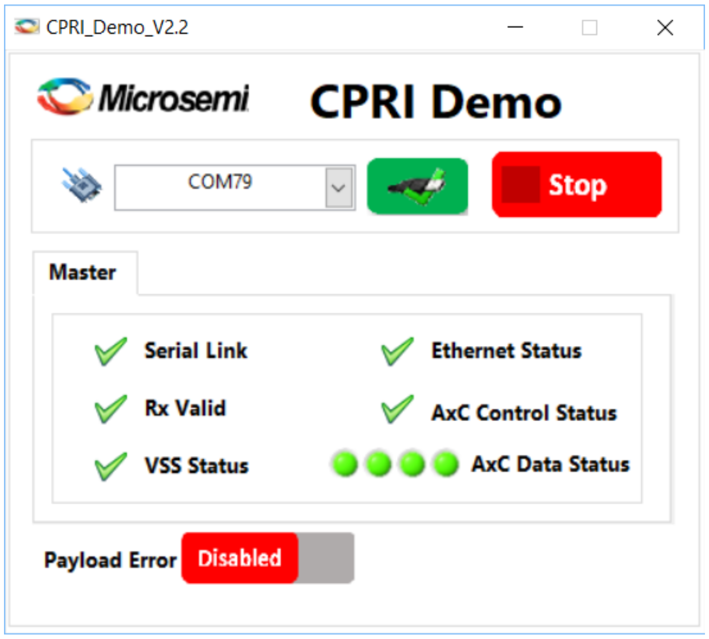
Figure 23 • **Selecting COM Port and Connecting**



7. Click **Start.** The data is generated and framed using the CPRI master module and sent over the serial transmit link. It is then received by the receiver, and CPRI slave IP unpacks the data and loopbacks the data to the CPRI master module. The checker in the Master system checks the incoming data for any errors. The status can be monitored using the status signals on the GUI at any time, as shown in the following figure. The following are the status signals:
   - Serial Link: Indicates transceiver link status
   - Rx Valid: Indicates if the transmitter and receiver data are locked
   - VSS status: Indicates the received VSS data is valid.
   - Ethernet status: Indicates the received Ethernet data is valid.
   - AxC Control status: Indicates the received Antenna carrier control data is valid.
   - AxC Data Status: Indicates the received Antenna carrier data is valid.

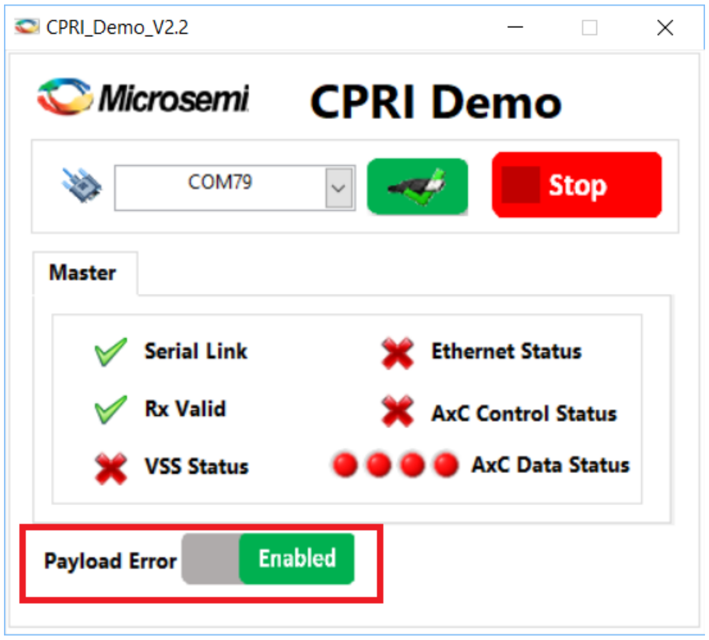
The following shows the status signals.The LED4, LED5, LED6, LED7, LED8, LED9, LED10, and LED11 is asserted at the same time.

Figure 24 • **Master Checker Status—Pass**



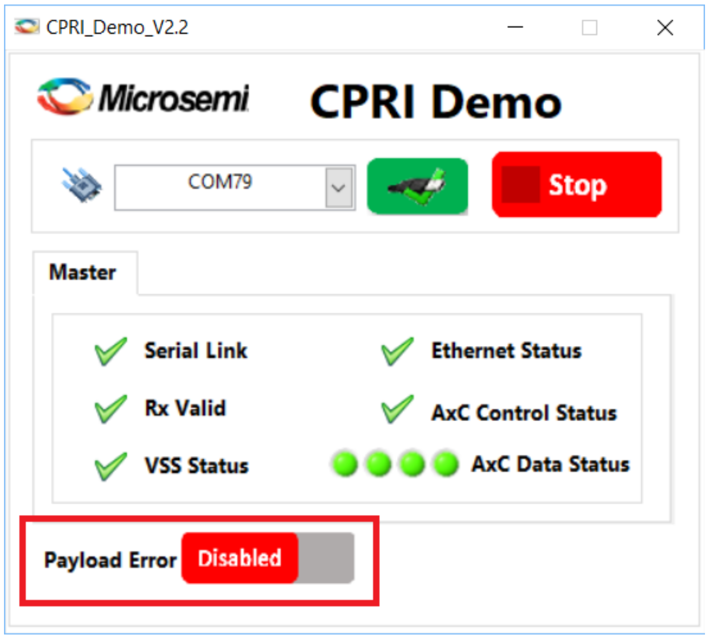
8. Click **Payload Error** to induce error in VSS data, Ethernet data, Antenna carrier control data and Antenna carrier Data. Observe the error status of master as shown following figure. The LED5, LED6, LED7, LED8, LED9, LED10, and LED11 is deasserted at the same time.

Figure 25 • **Payload Error Enabled—Master**



9. Disable Payload Error to stop generating an error and observe that the Serial Link, Rx Valid, and all the status signals of master turn green, as shown in the following figure. When the error is cleared, the LED5 to LED11 is asserted at the same time.

Figure 26 • **Payload Error Disabled—Master**



The CPRI demo is successfully run.

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 .stp file is: `mpf_dg0843_df\Programming_Job`

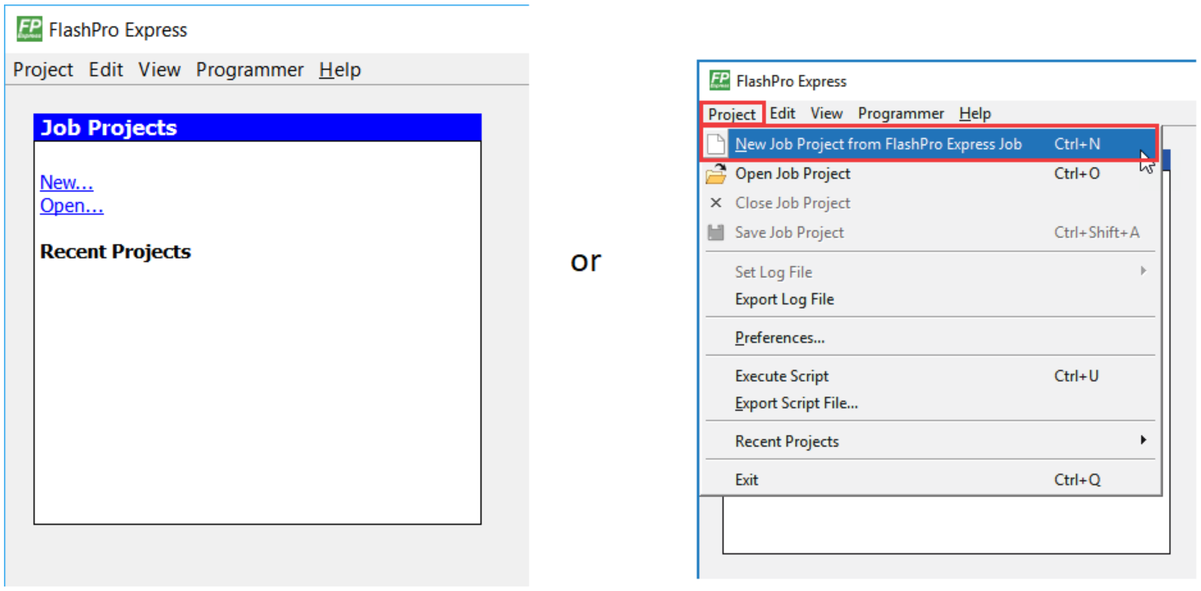
To program the PolarFire device using FlashPro Express, perform the following steps:

1. Ensure that the jumper settings on the board are the same as listed in Table 8, page 21.

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. 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.

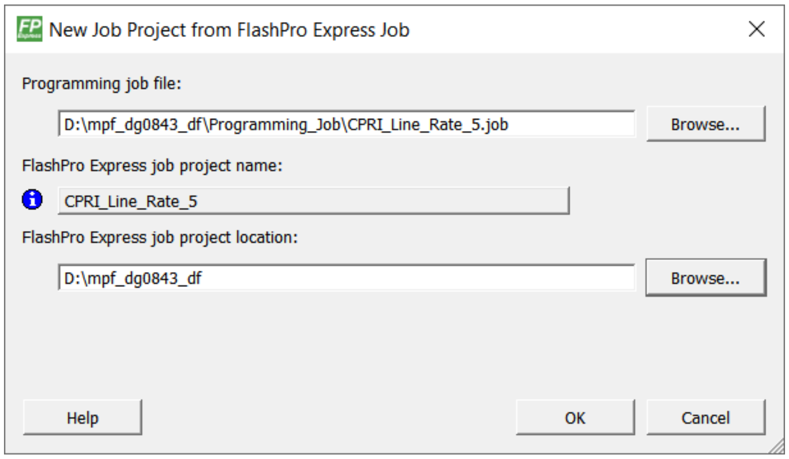
Figure 27 • **FlashPro Express Job Project**



7. Enter the following in the New Job Project from FlashPro Express Job dialog box:

- Programming job file: Click Browse, and navigate to the location where the .job file is located and select the file. The default location is: *<download_folder>\mpf_dg0843_df\Programming_Job*.
- FlashPro Express job project location: Click **Browse** and navigate to the location where you want to save the project.

Figure 28 • **New Job Project from FlashPro Express Job**



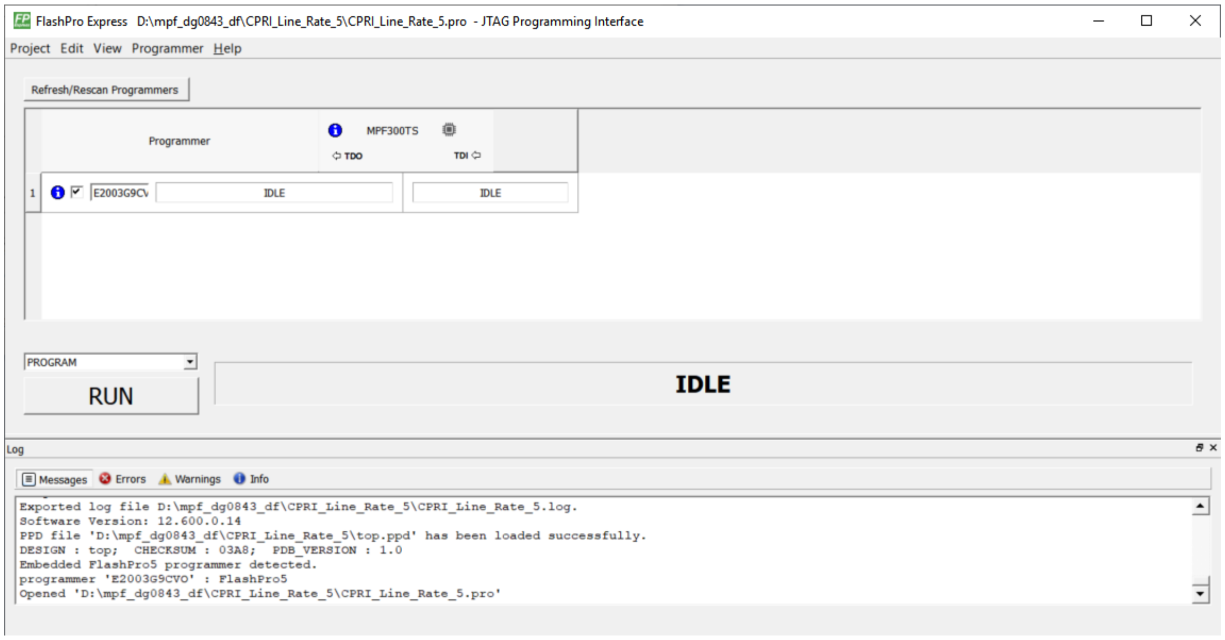
Programming job file: D:\mpf_dg0843_df\Programming_Job\CPRI_Line_Rate_5.job

FlashPro Express job project name: CPRI_Line_Rate_5

FlashPro Express job project location: D:\mpf_dg0843_df

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.

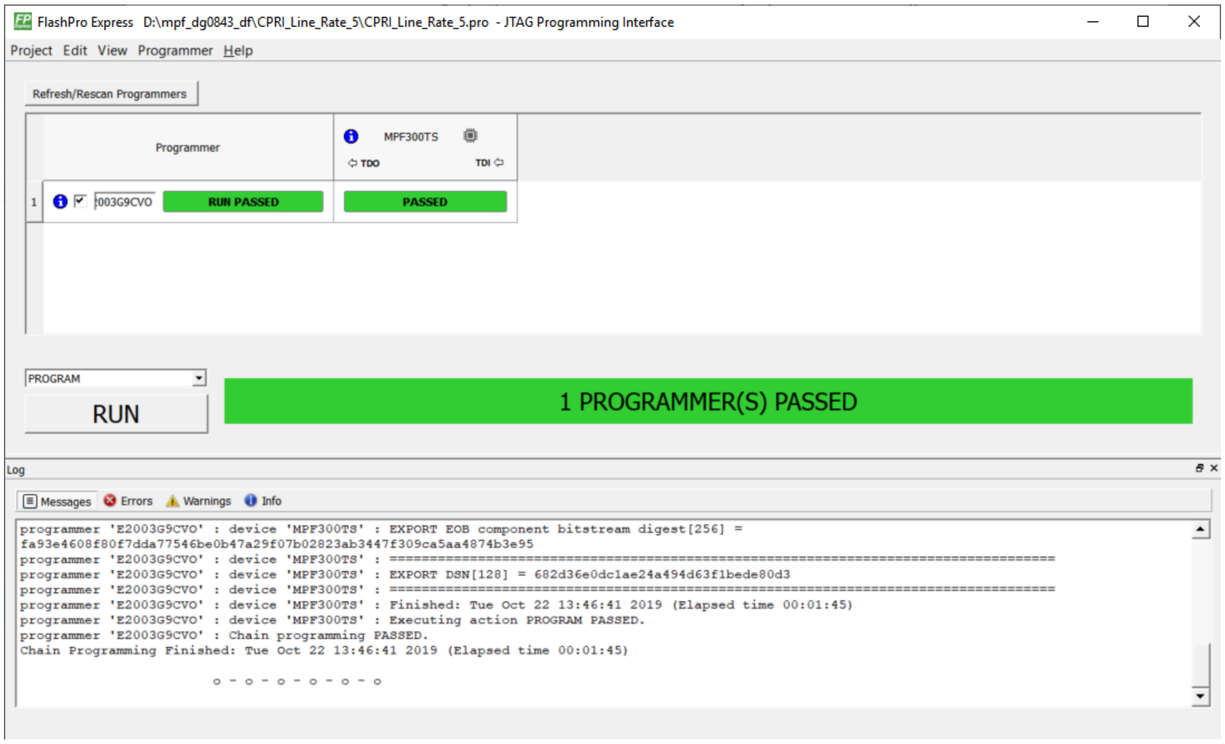
Figure 29 • **Programming the Device**


FlashPro Express D:\mpf_dg0843_df\CPRI_Line_Rate_5\CPRI_Line_Rate_5.pro - JTAG Programming Interface
Project
Edit
View
Programmer
Help
Refresh/Rescan Programmers
Programmer
MPF300TS
TDO
TDI
1
E2003G9CV
IDLE
IDLE
PROGRAM
RUN
IDLE
Log
Messages
Errors
Warnings
Info

```
Exported log file D:\mpf_dg0843_df\CPRI_Line_Rate_5\CPRI_Line_Rate_5.log.
Software Version: 12.600.0.14
PPD file 'D:\mpf_dg0843_df\CPRI_Line_Rate_5\top.ppd' has been loaded successfully.
DESIGN : top;  CHECKSUM : 03A8;  PDB_VERSION : 1.0
Embedded FlashPro5 programmer detected.
programmer 'E2003G9CVO' : FlashPro5
Opened 'D:\mpf_dg0843_df\CPRI_Line_Rate_5\CPRI_Line_Rate_5.pro'
```

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

Figure 30 • **FlashPro Express—RUN PASSED**


FlashPro Express D:\mpf_dg0843_df\CPRI_Line_Rate_5\CPRI_Line_Rate_5.pro - JTAG Programming Interface
Project Edit View Programmer Help
Refresh/Rescan Programmers
Programmer
MPF300TS
TDO
TDI
1
003G9CVO
RUN PASSED
PASSED
PROGRAM
RUN
1 PROGRAMMER(S) PASSED
Log
Messages
Errors
Warnings
Info

```
programmer 'E2003G9CVO' : device 'MPF300TS' : EXPORT EOB component bitstream digest[256] =
fa93e4608f80f7dda77546be0b47a29f07b02823ab3447f309ca5aa4874b3e95
programmer 'E2003G9CVO' : device 'MPF300TS' : ==================================================================================
programmer 'E2003G9CVO' : device 'MPF300TS' : EXPORT DSN[128] = 682d36e0dc1ae24a494d63f1bede80d3
programmer 'E2003G9CVO' : device 'MPF300TS' : ==================================================================================
programmer 'E2003G9CVO' : device 'MPF300TS' : Finished: Tue Oct 22 13:46:41 2019 (Elapsed time 00:01:45)
programmer 'E2003G9CVO' : device 'MPF300TS' : Executing action PROGRAM PASSED.
programmer 'E2003G9CVO' : Chain programming PASSED.
Chain Programming Finished: Tue Oct 22 13:46:41 2019 (Elapsed time 00:01:45)

                         o - o - o - o - o - o
```

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_dg0843_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.

7 Appendix 3: References

This section lists the documents that provide more information about the concepts and features covered in this demo guide.

- For more information about PolarFire transceiver blocks, see *UG0677: PolarFire FPGA Transceiver User Guide*.
- For more information about Libero, ModelSim, and Synplify, refer *Microsemi Libero SoC webpage*.
- For more information about CPRI IP, see **Libero Catalog** -> **Solution-Wireless**->**Microsemi CPRI User Guide**
- For more information about PolarFire Evaluation kit, see *UG0747: PolarFire FPGA Evaluation Kit User Guide*
- For more information about Power-Up and Reset, see *UG0725: PolarFire FPGA Device Power-Up and Resets User Guide*
- For more information about the CoreJTAGDEBUG IP core, see CoreJTAGDebug_HB.pdf from **Libero**->**Catalog.**
- For more information about the MIV_RV32 IP core, see *MIV_RV32 Handbook* from the Libero SoC Catalog.
- For more information about the CoreAHBtoAPB3 IP core, see *CoreAHBtoAPB3_HB.pdf*.
- For more information about the CoreUARTapb IP core, see *CoreUARTapb_HB.pdf*.
- For more information about the CoreAHBLite IP core, see *CoreAHBLite_HB.pdf*.
- For more information about the CoreAPB3 IP core, see *CoreAPB3_HB.pdf*.
- For more information about the CoreGPIO IP core, see *CoreGPIO_HB.pdf*