

DG0762
Demo Guide
PolarFire FPGA DSP FIR Filter



a  **MICROCHIP** company



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 10.0	1
1.2	Revision 9.0	1
1.3	Revision 8.0	1
1.4	Revision 7.0	1
1.5	Revision 6.0	1
1.6	Revision 5.0	1
1.7	Revision 4.0	1
1.8	Revision 3.0	1
1.9	Revision 2.0	1
1.10	Revision 1.0	1
2	PolarFire FPGA - DSP FIR Filter Demo	2
2.1	Design Requirements	2
2.2	Prerequisites	3
2.3	DSP FIR Design	3
2.3.1	Design Implementation	4
2.3.2	Design Blocks and IP Configuration	5
2.4	Clocking Structure	9
2.5	Reset Structure	9
2.6	Simulating the Design	10
2.6.1	Simulation Flow	12
3	Libero Design Flow	14
3.1	Synthesize	14
3.2	Place and Route	14
3.2.1	Resource Utilization	15
3.3	Verify Timing	16
3.4	Generate FPGA Array Data	16
3.5	Generate Bitstream	16
3.6	Run PROGRAM Action	17
4	DSP FIR Demo GUI	18
5	Running the Demo	19
5.1	Installing and Starting the GUI	19
5.2	Generating the Filter Coefficients and Input Signals	19
5.3	Generating the Filter Output	21
6	Appendix 1: Programming the Device Using FlashPro Express	24
7	Appendix 2: Running the TCL Script	27
8	Appendix 3: Coefficient Test File Format	28
9	Appendix 4: References	29

Figures

Figure 1	DSP FIR Filter Demo Design Block Diagram	3
Figure 2	Top-level SmartDesign	4
Figure 3	DSP FIR Filter SmartDesign	5
Figure 4	Configuring Two-Port LSRAM	6
Figure 5	Configuring PF_COREFIR_0 IP	7
Figure 6	Configuring PF_COREFFT_0 IP	7
Figure 7	Configuring PF_ccc_0_0 - Clock Options PLL	8
Figure 8	Configuring PF_ccc_0_0 - Output Clocks	8
Figure 9	Clocking Structure	9
Figure 10	Reset Structure	9
Figure 11	Testbench and DSP Demo Design Interaction	10
Figure 12	Pre-Synthesized Simulation	11
Figure 13	Simulating the Pre-Synthesis Design	12
Figure 14	CoreFIR Input and Output Signals	12
Figure 15	Low-Pass Filter Output	13
Figure 16	Libero Design Flow Options	14
Figure 17	Board Setup	17
Figure 18	FIR Filter GUI	18
Figure 19	Filter Generation	19
Figure 20	The Filter Response and Filter Coefficient Plot	20
Figure 21	Input Signal Generation	20
Figure 22	Input Signal and Input Signal FFT Plot	21
Figure 23	UART Connection	21
Figure 24	Filter Output	22
Figure 25	Zooming the Filter Output	22
Figure 26	The Text Viewer Tab	23
Figure 27	FlashPro Express Job Project	24
Figure 28	New Job Project from FlashPro Express Job	25
Figure 29	Programming the Device	25
Figure 30	FlashPro Express—RUN PASSED	26
Figure 31	Coefficient File Example - 9 Taps, Decimal Values	28

Tables

Table 1	Design Requirements	2
Table 2	I/O Signals	4
Table 3	PF_TPSRAM_0 Configuration for Data Buffers	6
Table 4	Clocks	9
Table 5	Simulation Signals	11
Table 6	DSP FIR Filter Demo Resource Usage Summary for the design with Core FIR	15
Table 7	DSP FIR Filter Demo Resource Usage Summary for the design with RTL Inference	15
Table 8	MathBlocks Usage Summary for the design with Core FIR	15
Table 9	MathBlocks Usage Summary for the design with RTL Inference	15
Table 10	RAM Blocks Usage Summary for the Design with Core FIR	16
Table 11	RAM Blocks Usage Summary for the Design with RTL Inference	16
Table 12	Jumper Settings	17

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 10.0

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

1.2 Revision 9.0

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 8.0

Updated the document for Libero[®] SoC v12.0.

1.4 Revision 7.0

Updated the document for Libero[®] SoC PolarFire v2.3.

1.5 Revision 6.0

Updated the document for Libero[®] SoC PolarFire v2.2.

1.6 Revision 5.0

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

- The document was updated for Libero SoC PolarFire v2.1.
- Updated [Running the Demo](#), page 19.

1.7 Revision 4.0

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

- The document was updated for Libero SoC PolarFire v2.0.
- Updated the GUI screens in [Running the Demo](#), page 19.

1.8 Revision 3.0

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

- The GUI images were updated.

1.9 Revision 2.0

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

- The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v1.1 SP1 release.

1.10 Revision 1.0

Revision 1.0 was the first publication of this document.

2 PolarFire FPGA - DSP FIR Filter Demo

PolarFire® FPGA devices integrate a fifth-generation flash-based FPGA fabric architecture that includes embedded MathBlocks optimized specifically for Digital Signal Processing (DSP) applications such as Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) filters, and Fast Fourier Transform (FFT) functions.

This document describes how to run the DSP FIR filter demo on a PolarFire Evaluation board. The DSP FIR filter demo is implemented using two Libero SoC designs. The design files include the following Libero project folders:

- **CoreFIR:** This folder contains the DSP filter design implemented using Microsemi's COREFIR_PF and CoreFFT IP cores.
- **FIR_RTL:** This folder contains the DSP filter design implemented using the FIR RTL inference and Microsemi's CoreFFT IP core.

The demo design consists of:

- FIR filter of tap 127 with re-loadable coefficients.
- A 256 point FFT on filter output to view spectrum.
- A GUI -UART interface from host PC to generate filter coefficients, input signals (Pass-band frequency and Stop-band frequency). Also plots the input/output waveforms, and the required spectrum.

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 24.
- **Using Libero SoC:** To program the device using Libero SoC, see [Libero Design Flow](#), page 14. Use this option when the demo design is modified.

2.1 Design Requirements

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

Table 1 • Design Requirements

Requirements	Version
Operating System	Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT)	Rev D or later
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	

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 start, download the demo design files from the following location:

1. For demo design files download link:
http://soc.microsemi.com/download/rsc/?f=mpf_dg0762_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>
 The latest versions of ModelSim and Synplify Pro are included in the Libero SoC installation package.

Note: A Libero Gold license is required to evaluate the designs using the PolarFire Evaluation Kit.

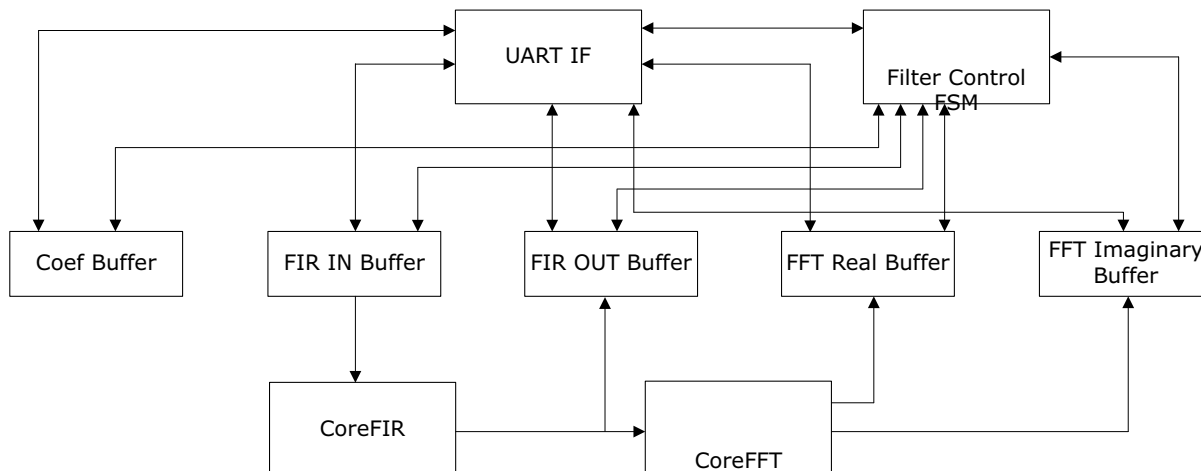
2.3 DSP FIR Design

The PolarFire DSP FIR demo design is developed for demonstrating filtering applications using DSP blocks such as FIR and FFT IPs. The following steps describe the data flow in the design:

1. Upon UART handshaking (sending and receiving the known patterns over UART bus to pre-verify the serial channel before actual usage), GUI sends the filter coefficients followed by filter input data.
2. UART IF block creates 16-bit packets and stores the data in the corresponding input data buffers and coefficient buffers.
3. Filter control FSM controls the following operations:
 - Reading the data from buffers
 - Writing the data into CoreFIR IP
4. Once the CoreFIR generates the output response, the data is stored into FIR OUT buffer.
5. CoreFFT real and imaginary outputs are stored into corresponding buffers.
6. UART IF block reads the data from FIR and FFT output buffers and sends the data to GUI through UART.

The top-level block diagram of the DSP FIR filter demo design is illustrated in the following figure.

Figure 1 • DSP FIR Filter Demo Design Block Diagram



2.3.1 Design Implementation

This section shows the DSP Filter design implemented using the CoreFIR and CoreFFT IP cores in Libero SoC.

In the DSP FIR filter demo design, the host interface and the FIR filter are implemented in the fabric for lowpass, bandpass, and bandstop filtering operations. The testbench provided for this demo uses pre-generated filter coefficients, input signals (Passband frequency and Stopband frequency), and passes the values to the demo design. The CoreFIR_PF IP is used to suppress unwanted frequency components, and the CoreFFT IP is used to generate the output spectrum to verify the filtering operation.

The following figure shows the top-level SmartDesign.

Figure 2 • Top-level SmartDesign

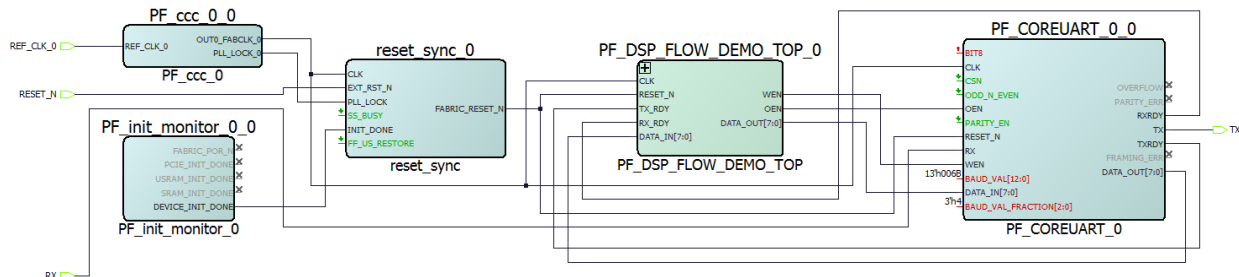
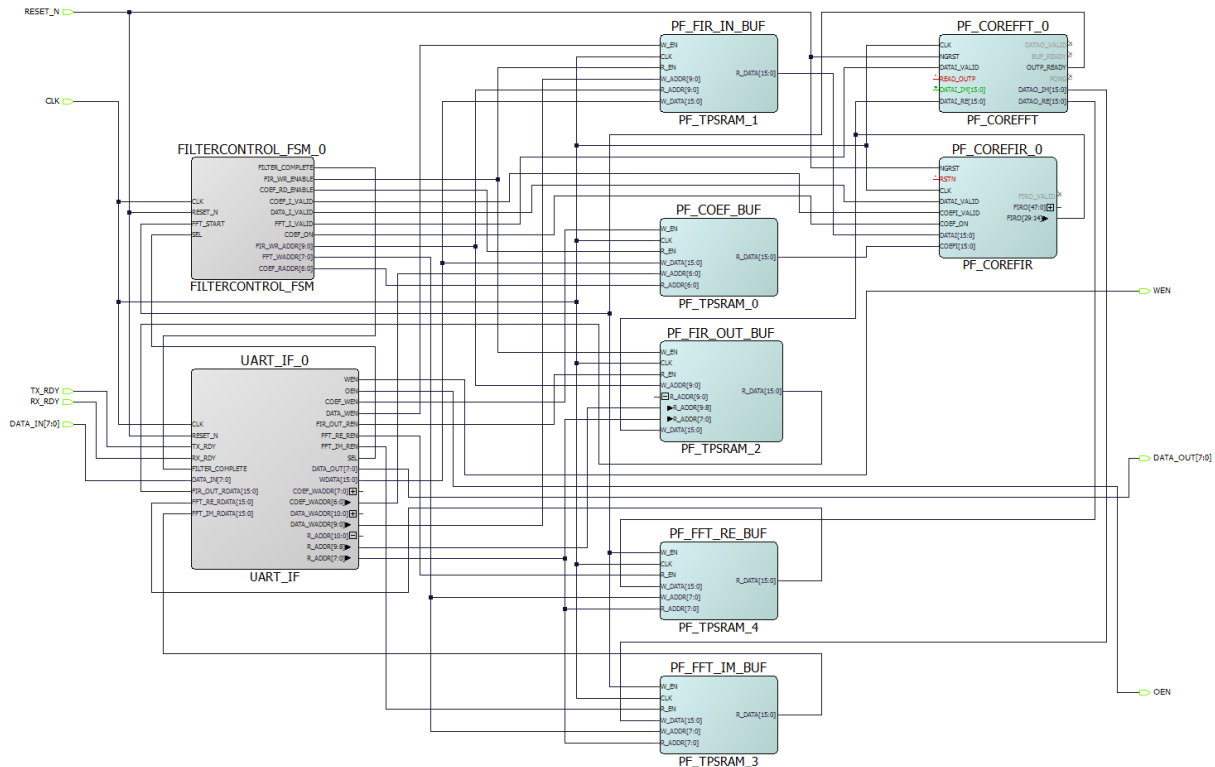


Table 2 • I/O Signals

Signal	Description
Input Signals	
REF_CLK_0	Reference clock obtained from on-board 50 MHz oscillator.
RESET_N	This is the reset signal obtained from the SW push-button switch on the board.
RX	This is the UART receive data input.
Output Signals	
TX	This is the UART transmit data output.

The following figure shows the PF_DSP_FLOW_DEMO_TOP_0 SmartDesign.

Figure 3 • DSP FIR Filter SmartDesign



Note: This design is similar to the design implemented using the FIR RTL inference and the CoreFFT IP core. The only difference is the use of the FIR RTL inference in the place of CoreFIR PF.

2.3.2 Design Blocks and IP Configuration

The following IPs need to be configured before simulating and implementing the demo design.

- PF_TPSRAM IP, page 5
- UART_IF_0, page 6
- FILTERCONTROL_FSM_0, page 6
- PF_COREFIR_0, page 7
- PF_COREFFT_0, page 7
- PF_ccc 0 0, page 8

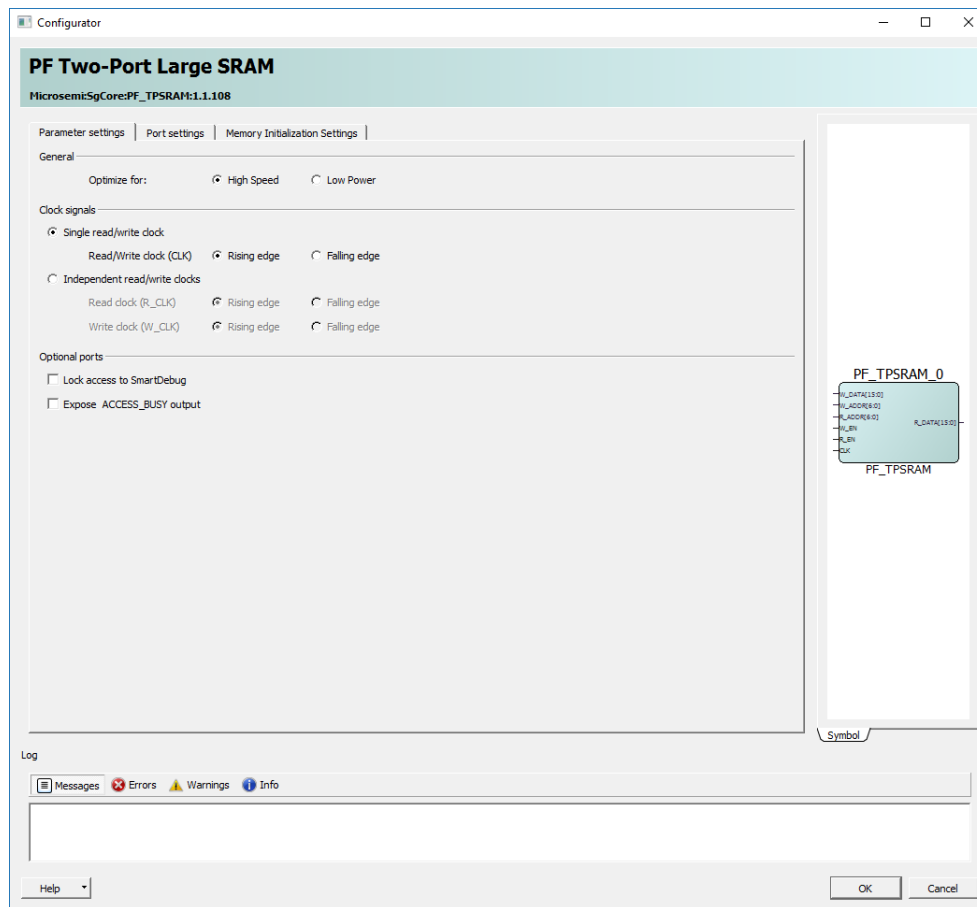
Note: For more information about IP blocks, see [Figure 2](#), page 4 and [Figure 3](#), page 5.

2.3.2.1 PF_TPSRAM IP

Five instances of PF_TPSRAM blocks in the design are described as follows:

- **Filter coefficient buffer (PF_COEF_BUF):** Stores the coefficients received from GUI before sending it to the FIR
- **Input signal data buffer (PF_FIR_IN_BUF):** Stores the input data received from GUI before sending it to the FIR
- **Output signal buffer (PF_FIR_OUT_BUF):** Stores the FIR output data received from FIR IP before sending it to GUI
- **Output signal FFT real data buffer (PF_FFT_RE_BUF):** Stores the output data (real part) received from FFT IP before sending it to GUI
- **Output signal FFT imaginary data buffer (PF_FFT_IM_BUF):** Stores the output data (imaginary part) received from FFT IP before sending it to GUI

The PF_COEF_BUF, PF_FIR_IN_BUF, PF_FIR_OUT_BUF, PF_FFT_RE_BUF, and PF_FFT_IM_BUF blocks are configured for RAM size depth and width, as shown in the following figure and table.

Figure 4 • Configuring Two-Port LSRAM**Table 3 • PF_TPSRAM_0 Configuration for Data Buffers**

Buffer	Write Port		Read Port	
	Depth	Width	Depth	Width
PF_COEF_BUF	128	16	128	16
PF_FIR_IN_BUF	1024	16	1024	16
PF_FIR_OUT_BUF	1024	16	1024	16
PF_FFT_RE_BUF	256	16	256	16
PF_FFT_IM_BUF	256	16	256	16

2.3.2.2 UART_IF_0

The UART_IF block (`UART_IF.v`) consists of a finite state machine handling control operations between the UART and fabric logic. Control operations include the loading of filter coefficients, filtering input data to the corresponding input data buffers and coefficient buffers, and sending and receiving data from the UART.

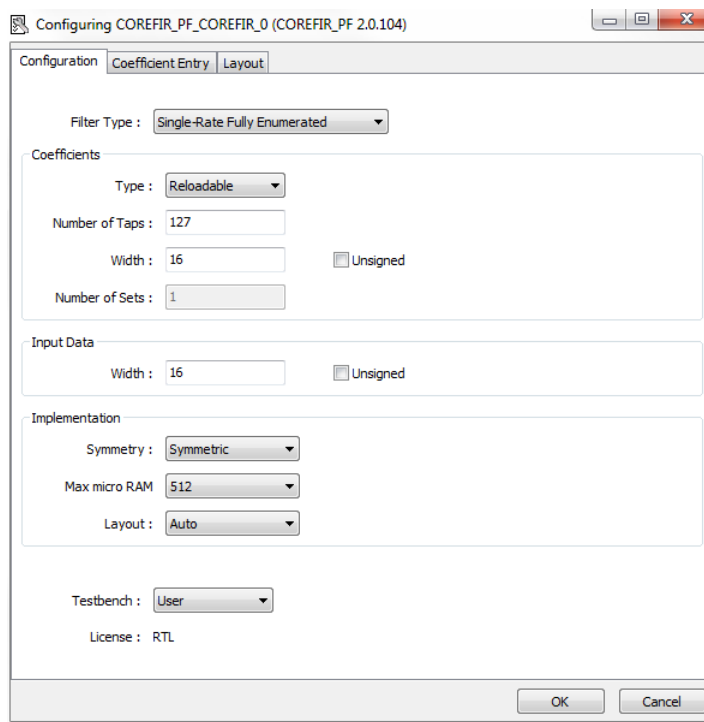
2.3.2.3 FILTERCONTROL_FSM_0

The filter control FSM block (`FILTER_CONTROL_FSM.v`) handles the data flow and controls signals of FIR filter and FFT. It loads the filtered data to the corresponding output buffer and moves the FFT output data to the corresponding FFT real and imaginary buffers.

2.3.2.4 PF_COREFIR_0

The PF_COREFIR_0 IP is used in reloadable coefficient mode to support lowpass, bandpass, and bandstop filters.

Figure 5 • Configuring PF_COREFIR_0 IP



Configuring COREFIR_PF_COREFIR_0 (COREFIR_PF 2.0.104)

Configuration | Coefficient Entry | Layout

Filter Type : Single-Rate Fully Enumerated

Coefficients

Type : Reloadable

Number of Taps : 127

Width : 16 ☐ Unsigned

Number of Sets : 1

Input Data

Width : 16 ☐ Unsigned

Implementation

Symmetry : Symmetric

Max micro RAM : 512

Layout : Auto

Testbench : User

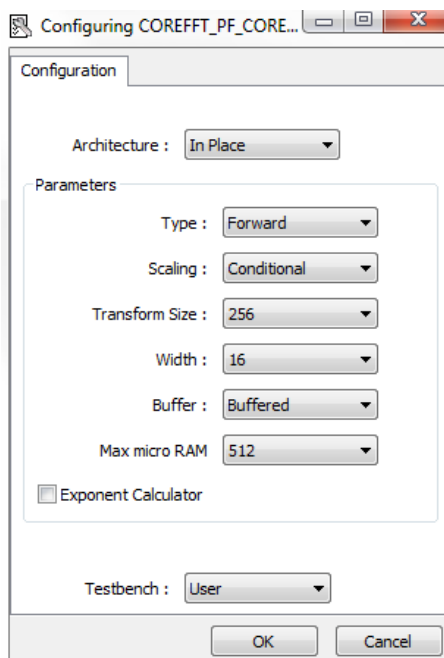
License : RTL

OK Cancel

2.3.2.5 PF_COREFFT_0

The PF_COREFFT_0 IP generates the frequency spectrum of the filtered data as shown in the following figure.

Figure 6 • Configuring PF_COREFFT_0 IP



Configuring COREFFT_PF_CORE...

Configuration

Architecture : In Place

Parameters

Type : Forward

Scaling : Conditional

Transform Size : 256

Width : 16

Buffer : Buffered

Max micro RAM : 512

☐ Exponent Calculator

Testbench : User

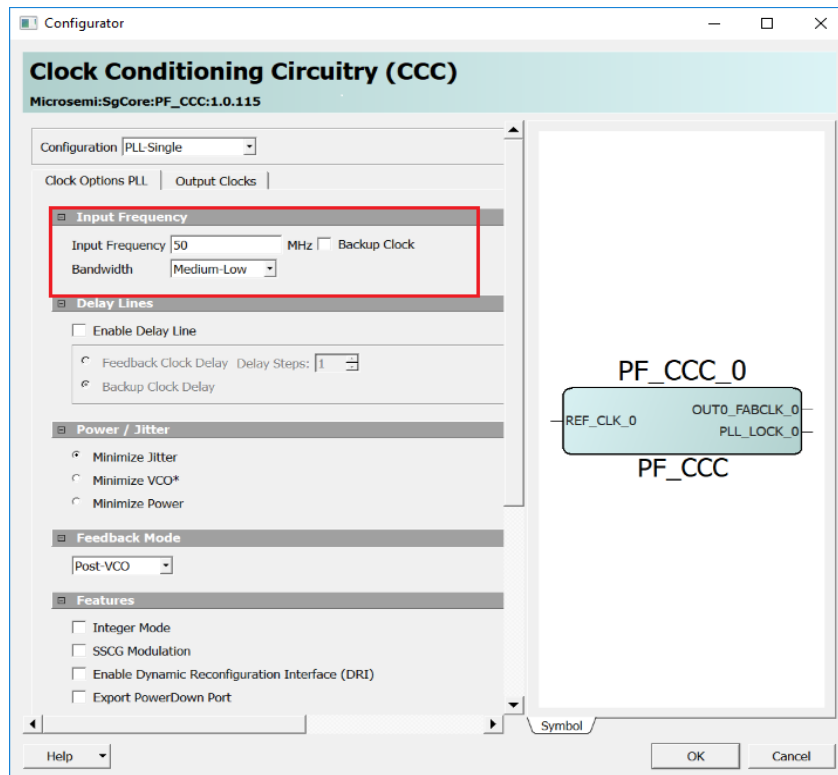
OK Cancel

2.3.2.6 PF_ccc_0_0

The PF_ccc_0_0 IP is configured to take 50 MHz reference clock as input and generate 200 MHz output clock as shown in the following figures.

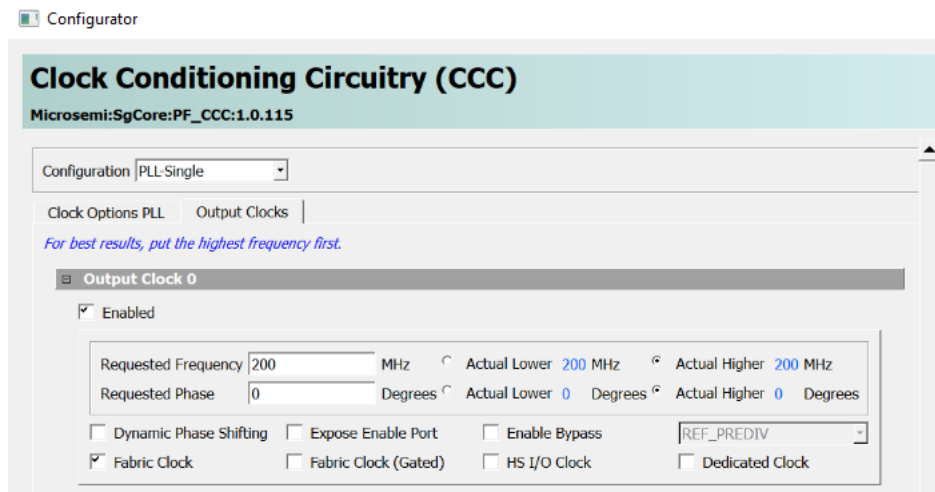
The following figure shows input clock configuration.

Figure 7 • Configuring PF_ccc_0_0 - Clock Options PLL



The following figure shows output clock configuration.

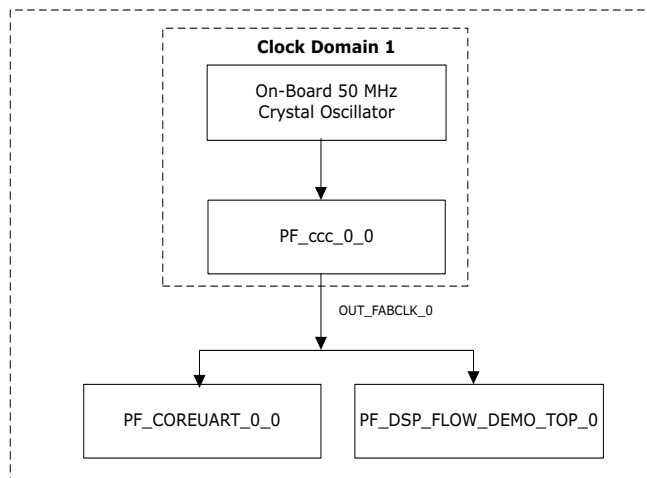
Figure 8 • Configuring PF_ccc_0_0 - Output Clocks



2.4 Clocking Structure

The demo design has only one clock domain. From the on-board 50 MHz crystal oscillator connected to the PF_ccc_0_0 block generates 200 MHz clock which drives PF_COREUART_0_0 and PF_DSP_FLOW_DEMO_TOP_0 blocks. The following figure shows the clocking structure in the demo design.

Figure 9 • Clocking Structure



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

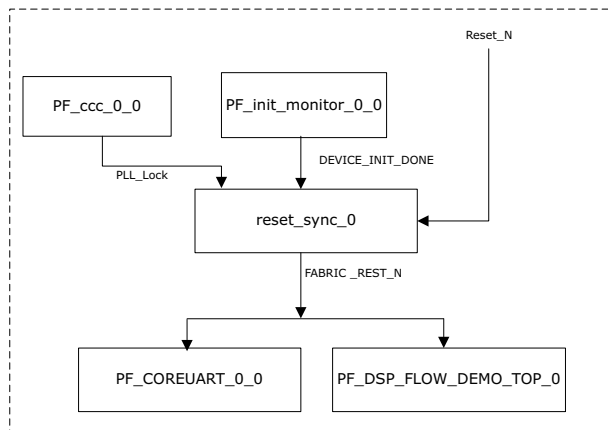
Table 4 • Clocks

Clock Name	Source	Frequency
OUT_FABCLK_0	PF_CCC_0_0	200 MHz

2.5 Reset Structure

The DEVICE_INIT_DONE, PLL Lock and the Reset_N signal mapped to K22 (evaluation board) initiates the reset signal (FABRIC_RESET_N) from the res_syn_0 block, which synchronizes with the Fabric Clock. The following figure shows the reset structure in the demo design.

Figure 10 • Reset Structure



2.6 Simulating the Design

Before you begin:

1. Start Libero SoC, and from the menu bar, select **Project -> Tool Profiles...**
2. Under the **Tool Profiles** window, select **Synthesis** and **Simulation** on the **Tools** panes and select the latest active installation directory paths for these two tools
3. In the **Project** menu, click **Open Project**. The Open Project dialog box opens
4. For the design with CoreFIR, use the location: `mpf_dg0762_df\Libero_Project\CoreFIR\Libero_Project`.
5. For the design with the FIR RTL inference, use the location: `mpf_dg0762_df\Libero_Project\FIR_RTL\Libero_Project`
6. Go to design files folder at `Libero_Project\CoreFIR` and select the `Libero_Project.PRJX` file.
7. Click **Open**. The PolarFire DSP FLOW project opens in Libero SoC.
8. Open the **Design Hierarchy** tab and double-click the **TOP** component. The SmartDesign page opens in the right-side pane and displays the high-level design.
9. Double-click the **PF_DSP_FLOW_DEMO_TOP_0** component. Top-level Simulation design internal modules are displayed.
10. Download the following IP cores from **Libero SoC -> Catalog**:
 - PF_COREFIR_0
 - PF_COREFFT_0
 - PF_TPSRAM
 - PF_ccc_0_0
 - PF_COREUART_0_0

A testbench is provided to simulate the design. The testbench simulates the filter pattern and waveform selection. It contains the test selection for the coefficient inputs (lowpass, bandpass, and bandstop) and data input. It also monitors the UART_IF module status signals, output signals (DATAOUT), and FFT output status signals (DATA Valid, output ready) for the verification of filter output.

Figure 11 • Testbench and DSP Demo Design Interaction

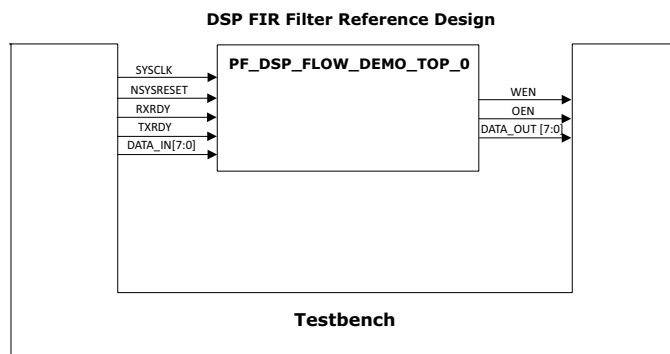


Table 5 • Simulation Signals

Signals	Description
Input Signals	
SYSCLK	200 MHz generated clock
NSYSRESET	Active low reset signal
RXRDY	Receive ready
TXRDY	Transmit ready
DATA_IN[7:0]	8-bit input data (Handshake/Coefficient/Data)
Output Signals	
WEN	Write enable
OEN	Output enable
DATA_OUT[7:0]	8-bit output data (Handshake/FIR Output/FFT Output Data)

11. To simulate the low-pass, high-pass, band-pass, or the band-reject functionality, include the corresponding DAT file in the testbench.v file as shown in (Figure 12, page 11).

The following DAT files are available for the respective filter functionality.

- For lowpass, coe_file_Low_Pass.dat
- For highpass, coe_file_High_pass.dat
- For bandpass, coe_file_Band_pass.dat
- For bandstop, coe_file_Band_reject.dat

Figure 12 • Pre-Synthesized Simulation

```

85 //Reading input data file
86 initial begin
87   data_file = $fopen("data_file.dat", "r");
88   if (data_file == `NULL) begin
89     $display("data_file handle was NULL");
90     $finish;
91   end
92 end
93 //COE File reading
94 initial begin
95   coe_data_file = $fopen("coe_file_Low_pass.dat", "r");
96   if (coe_data_file == `NULL) begin
97     $display("coe_file handle was NULL");
98     $finish;
99   end
100 end

```

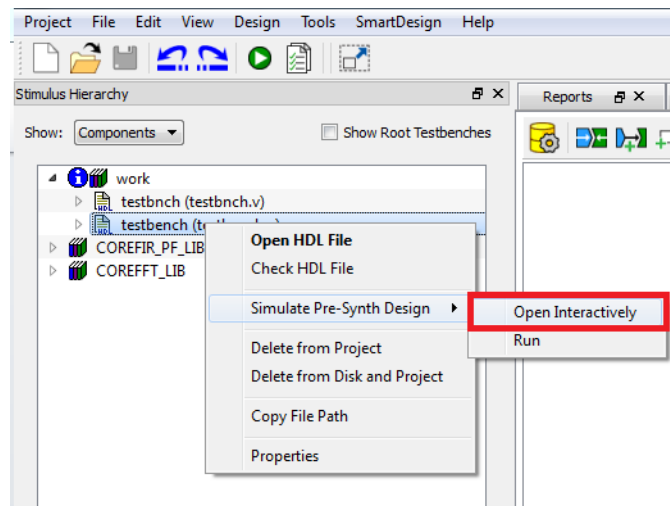
Note: For more information about parameters used to generate coefficient files for all the filters, see `Readme.txt`.

12. In the **Stimulus Hierarchy** tab, right-click **testbench**, and select **Open Interactively** from **Simulate Pre-Synth Design**, as shown in the following figure.

The ModelSim tool completes the simulation in 3 minutes.

When the simulation is initiated, the ModelSim tool compiles all of the design source files, runs the simulation, and configures the waveform viewer to show the simulation signals.

When the simulation is successful, the success status is updated in the transcript tab in modelsim.

Figure 13 • Simulating the Pre-Synthesis Design

2.6.1 Simulation Flow

The following steps describe the testbench simulation flow:

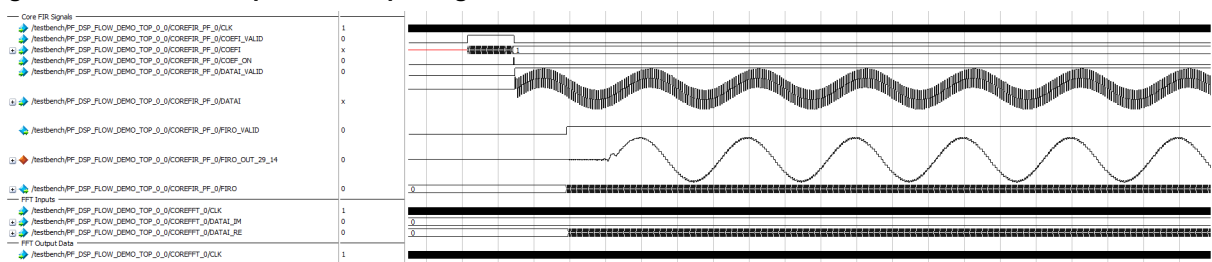
At the start, the `NSYSREST` signal resets all of the components.

1. On initializing the `UART_IF` block, coefficient values are sent to the block when the receiver is high.
Note: UART communication channel is initialized by sending data '9' from test bench and checking whether the expected data 'F' is sent from UART IF block, followed by 'h' and 'a' patterns.
2. Filter input data values are sent to the design on handshaking pattern 'r' received from UART IF block, when the `RX_READY` signal is high.
3. When FFT output is ready, the mean value is calculated in the testbench with imaginary and real outputs.
4. When the `FILTER_COMPLETE` signal is received, the testbench issues the "Test Completed successfully" message indicating that the simulation was successful.

Filter coefficients are generated with the following parameters:

- Filter Type: Low Pass (Low pass/Band pass/Band stop filter)
- Filter Window: Blackman
- Low Cut-off Frequency: Disabled for Low pass filter required
- High Cut-off Frequency: 20 MHz (Band Pass/Reject Low Cut off frequency: 10 MHz, High Cut-off frequency: 20 MHz)
- Signal generation Input Frequency 1: 1 MHz
- Signal generation Input Frequency 1: 50 MHz
- Filter Taps: 31

Note: For more information about parameters used to generate coefficient files, see `Readme.txt`.

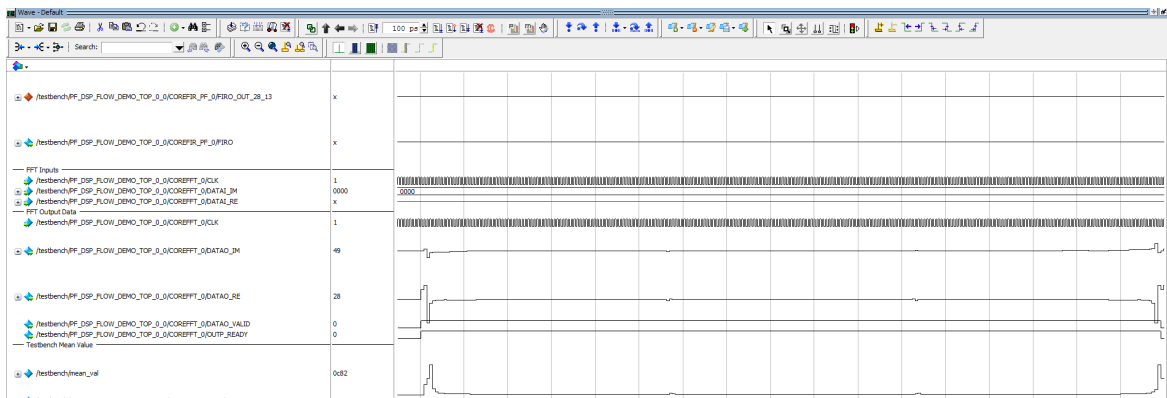
Figure 14 • CoreFIR Input and Output Signals

In the preceding figure:

- COEFFI represents Input coefficients
- DATAI represents FIR input data
- FIRO represents FIR output data

Note: Output data is valid when FIRO_VALID is high.

Figure 15 • Low-Pass Filter Output



In the preceding figure:

- DATAO_IM represents imaginary data output
- DATAO_RE represents real part of FFT output

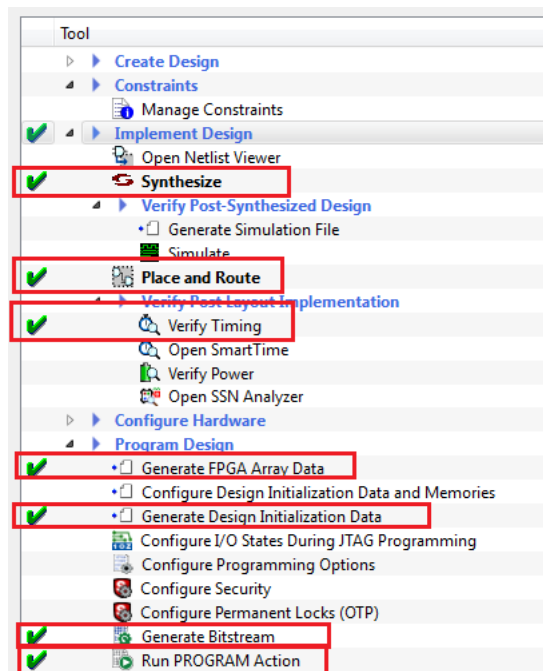
3 Libero Design Flow

The Libero design flow involves the following steps:

- Synthesize, page 14
- Place and Route, page 14
- Verify Timing, page 16
- Generate FPGA Array Data, page 16
- Generate Bitstream, page 16
- Run PROGRAM Action, page 17

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

Figure 16 • Libero Design Flow Options



3.1 Synthesize

To synthesize the design:

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

3.2 Place and Route

To place and route the design:

From the **Design Flow** tab, double-click **Place and Route**. When place and route is successful, a green tick mark appears as shown in Figure 16, page 14.

3.2.1 Resource Utilization

The DSP interface design is implemented on the PolarFire FPGA device (MPF300T_ES-FCG1152 package). The following table shows the resource utilization report after Place and Route process.

Note: The resource utilization may vary slightly based on different runs.

Table 6 • DSP FIR Filter Demo Resource Usage Summary for the design with Core FIR

Type	Used	Total	Percentage
4LUT	5012	299544	1.67
DFF	6068	299544	2.03
I/O Register	0	1536	0.00
Logic Element	6789	299544	2.27
μSRAM	50	2772	1.66
LSRAM	5	952	0.53
MATH	68	924	7.36

Table 7 • DSP FIR Filter Demo Resource Usage Summary for the design with RTL Inference

Type	Used	Total	Percentage
4LUT	4942	299544	1.65
DFF	7697	299544	2.57
I/O Register	0	1536	0.00
Logic Element	8404	299544	2.81
μSRAM	42	2772	1.52
LSRAM	5	952	0.53
MATH	68	924	7.36

The following tables show MathBlock usage summary.

Table 8 • MathBlocks Usage Summary for the design with Core FIR

CoreFIR	CoreFFT	Total
64	4	68

Table 9 • MathBlocks Usage Summary for the design with RTL Inference

FIR RTL	CoreFFT	Total
64	4	68

The following tables show RAM block usage summary.

Table 10 • RAM Blocks Usage Summary for the Design with Core FIR

RAM Type	CoreFIR	CoreFFT	Fabric Buffers	Total
μSRAM	4	42	0	46
LSRAM	0	0	5	5
Total	4	42	5	51

Table 11 • RAM Blocks Usage Summary for the Design with RTL Inference

RAM Type	FIR RTL	CoreFFT	Fabric Buffers	Total
μSRAM	0	42	0	42
LSRAM	0	0	5	5
Total	0	42	5	47

3.3 Verify Timing

To verify timing:

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.

3.4 Generate FPGA Array Data

To generate FPGA array data:

1. Double-click **Generate FPGA Array Data** from the **Design Flow** tab.
2. A green tick mark is displayed after the successful generation of the FPGA array data as shown in [Figure 16](#), page 14.

3.5 Generate Bitstream

To generate the bitstream:

1. 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.
2. Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.

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

3.6 Run PROGRAM Action

After generating bitstream, the PolarFire device must be programmed. Follow these steps to program the PolarFire device:

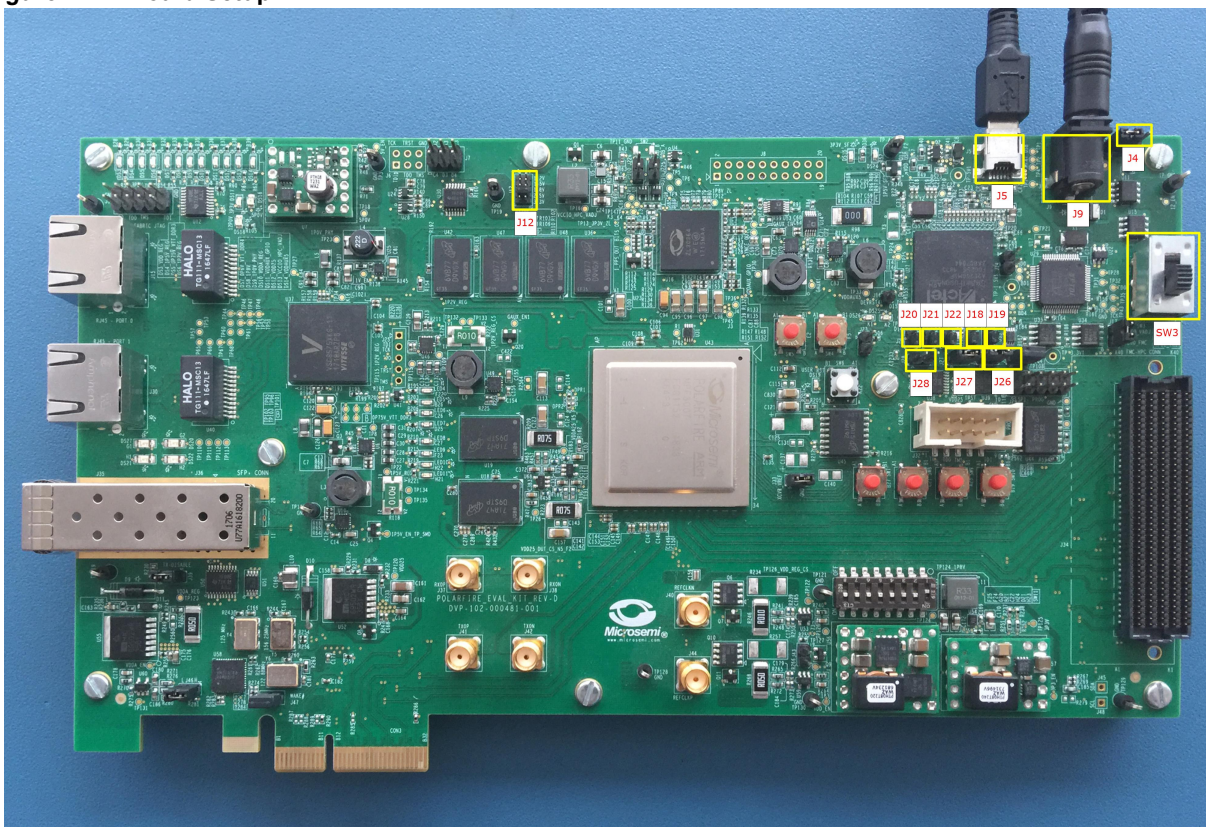
1. Ensure that the jumper settings on board are the same as those listed in the following table.

Table 12 • Jumper Settings

Jumper	Description
J18, J19, J20, J21, and J22	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Close pin 2 and 3 for programming through the on-board FlashPro5
J26	Close pin 1 and 2 for programming through the FTDI SPI
J27	Close pin 1 and 2 for programming through the FTDI SPI
J4	Close pin 1 and 2 for manual power switching using SW3
J12	Close pin 3 and 4 for 2.5 V

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.

Figure 17 • Board Setup



5. Double-click **Run PROGRAM Action** from the **Libero > Design Flow** tab.
6. Right-click **Run Program Action** and select **View Report** to view the corresponding log file in the **Reports** tab.

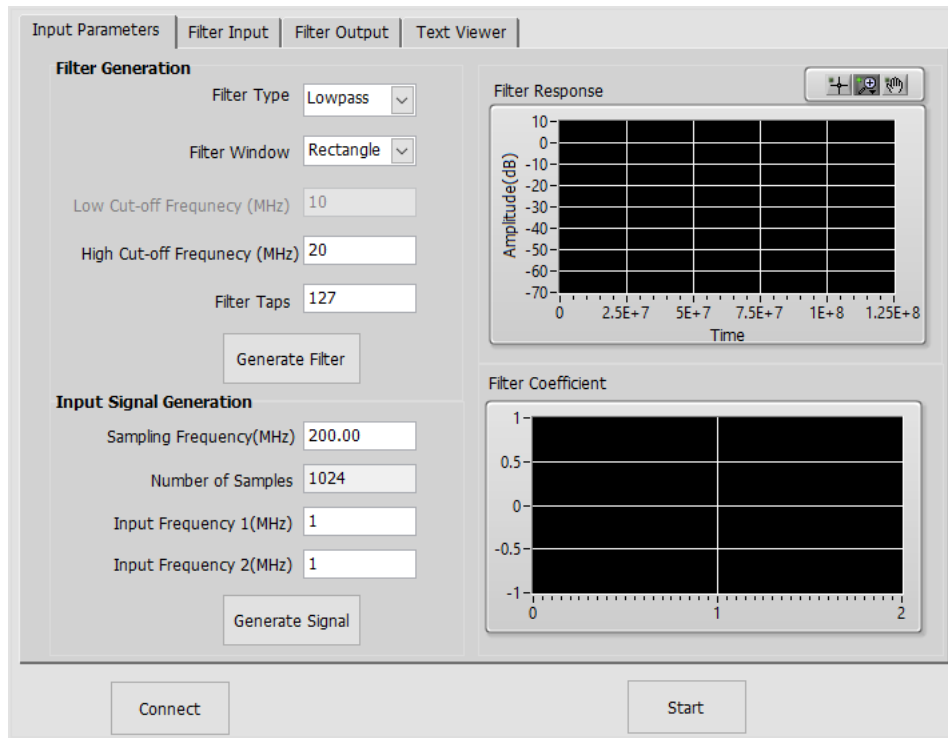
When the device is successfully programmed, a green tick mark appears as shown in Figure 16, page 14. See, [Running the Demo](#), page 19 to run the DSP FIR filter demo.

4 DSP FIR Demo GUI

The FIR Filter GUI application runs on the host PC connected to the PolarFire Evaluation Kit. UART is used as the communication protocol between the host PC and the PolarFire Evaluation Kit.

The following figure shows the DSP FIR Filter GUI.

Figure 18 • FIR Filter GUI



The DSP FIR demo window consists of the following tabs:

- **Input Parameters:** Configures the filter generation and signal generation
- **Filter Input:** Plots the input signal and its frequency spectrum
- **Filter Output:** Plots the output signal and its frequency spectrum
- **Text Viewer:** Shows the coefficients, input signal, output signal, and FFT data values

5 Running the Demo

Running the demo involves the following steps:

1. Installing and Starting the GUI, page 19
2. Generating the Filter Coefficients and Input Signals, page 19
3. Generating the Filter Output, page 21

Note: The steps to run the demo are the same for the CoreFIR IP design and the FIR RTL inference design. The following sections take the CoreFIR IP design as an example.

5.1 Installing and Starting the GUI

To install and start the GUI, perform the following steps:

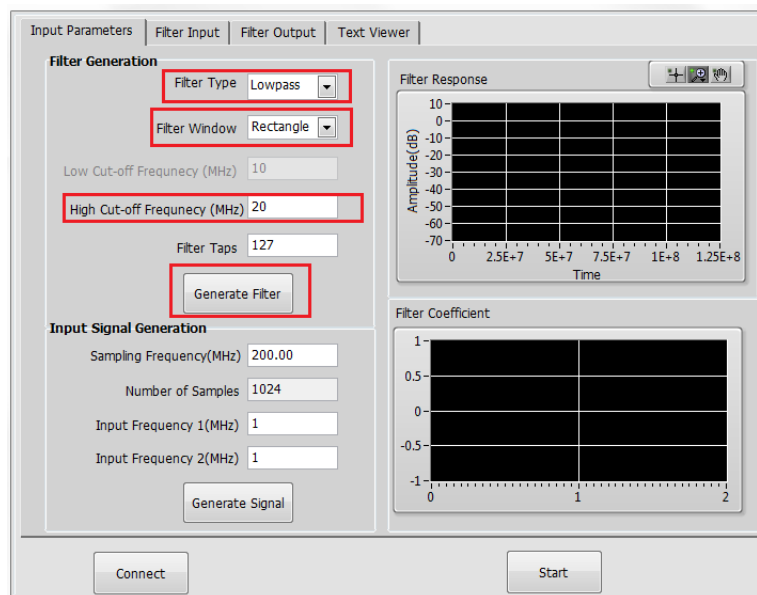
1. Double-click the **DSP FIR Demo GUI** application (`setup.exe`) from the following design files folder:
`mpf_dg0762_df\GUI\setup.exe`
2. Follow the installation wizard to install the GUI application.
3. Double-click the **FIR_Filter_GUI.exe** application from the installation directory to start the GUI application. The FIR Filter GUI window is displayed as shown in Figure 18, page 18.

5.2 Generating the Filter Coefficients and Input Signals

Follow these steps:

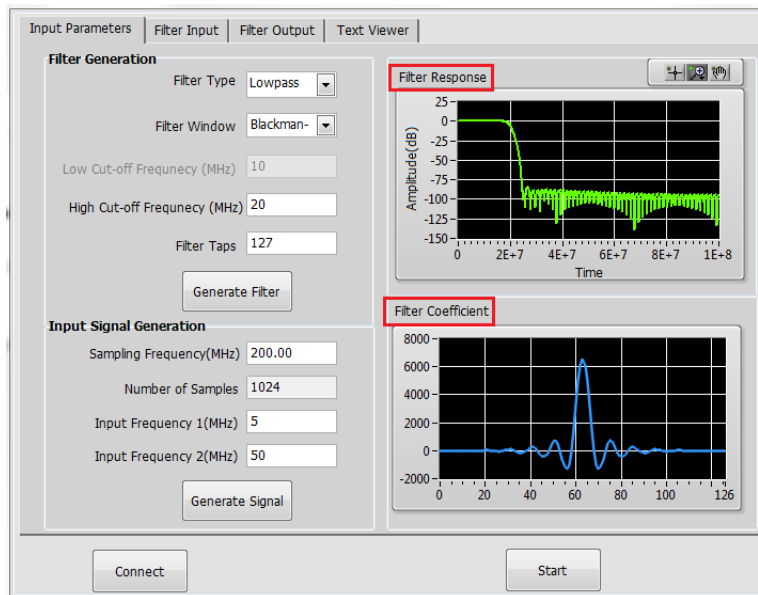
1. To generate the filter coefficients, set the following parameters in the GUI and click **Generate Filter**, as shown in Figure 19, page 19:
 - **Filter Type:** Low pass (Lowpass/Highpass/Bandpass/Bandstop filter)
 - **Filter Window:** Blackman-Harris (Blackman-Harris/Blackman/Hamming/Hanning/Rectangle/Flat Space/Kaiser window)
 - **Low Cut-off Frequency (MHz):** 10 (disabled for Low pass filter required)
 - **High Cut-off Frequency (MHz):** 20
 - **Filter Taps:** 127 (Fixed)

Figure 19 • Filter Generation



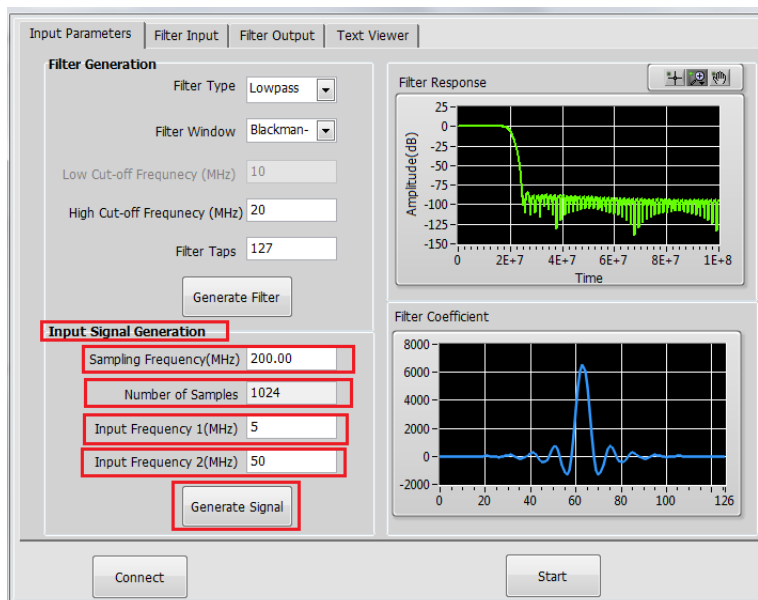
- After generating the filter coefficients, the Filter Response and the Filter Coefficient plots are displayed as shown in the following figure.

Figure 20 • The Filter Response and Filter Coefficient Plot



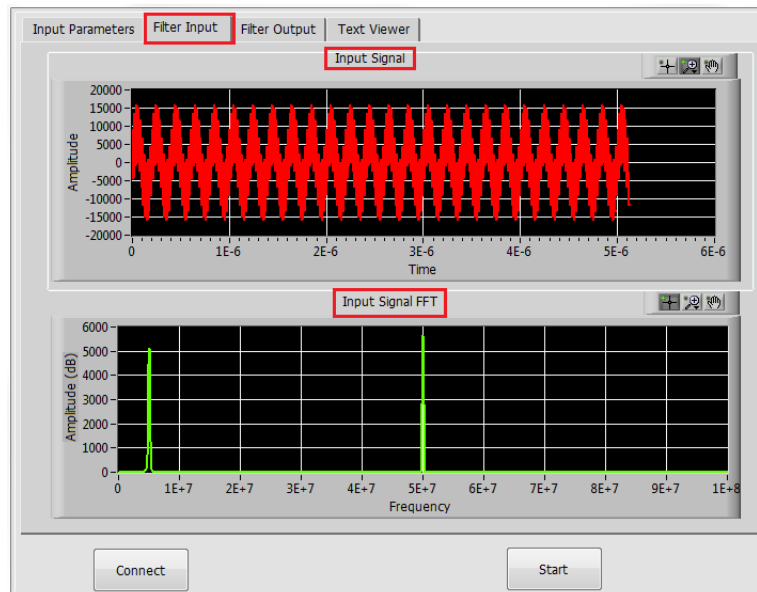
- To generate the inputs signals, set the following parameters in the GUI and click **Generate Signal** as shown in Figure 21, page 20:
 - Sampling Frequency (MHz):** 200 (Fixed)
 - Number of Samples:** 1024 (Fixed)
 - Input Frequency 1 (MHz):** Enter the signal frequency in the passband region. For example, 5 MHz for high cut-off frequency
 - Input Frequency 2 (MHz):** Enter the signal frequency in the stopband region. The stopband frequency is generally set to a value less than the half of the sampling frequency. For example, 50 MHz.

Figure 21 • Input Signal Generation



- The input signals and the frequency spectrum of the specified signals are displayed in the Filter inputs tab, as shown in the following figure.

Figure 22 • Input Signal and Input Signal FFT Plot

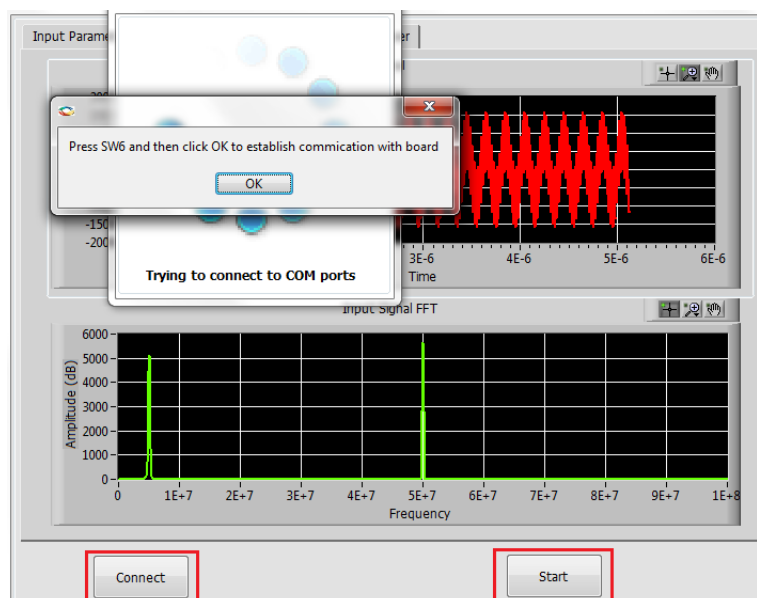


5.3 Generating the Filter Output

To generate the filter output, perform the following steps:

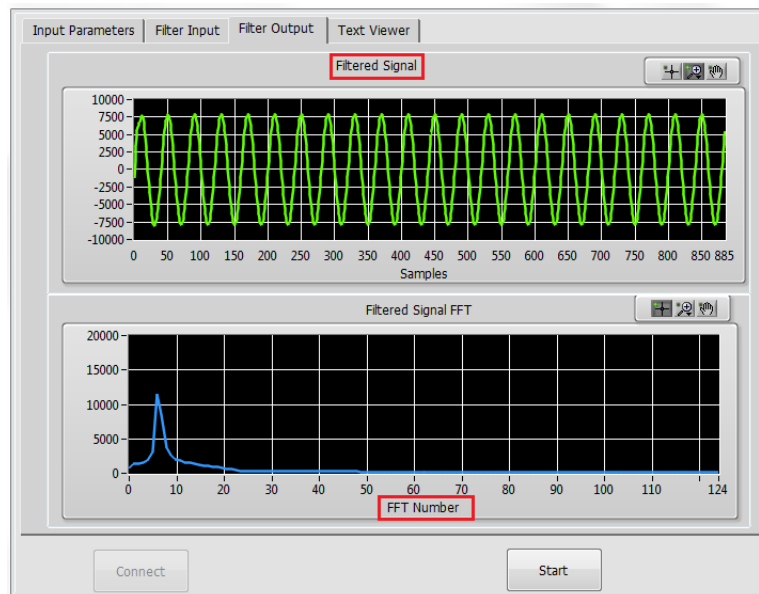
- To configure the input frequencies and coefficients, click **Connect** as shown in following figure. Once the COM port connection is established, click **Start**. The GUI prompts to press the onboard SW6 and click **OK** in the pop-up window, as shown in Figure 23, page 21.
- The GUI application sends the input data (1K samples) and the filter coefficients to the PolarFire device to perform the filtering operation.

Figure 23 • UART Connection



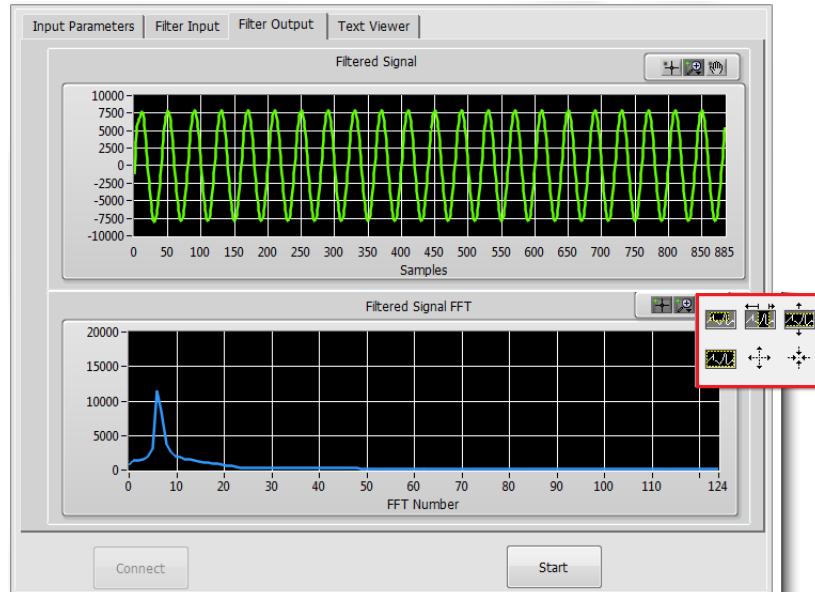
After completing the filter operation, the GUI displays the “Operation Completed” message and plots the filtered data and the FFT data on the Filter Output tab as shown in Figure 24, page 22. Since the Low pass filter option was selected, the high-frequency component is suppressed while the low-frequency signal is preserved. This can be observed in the frequency spectrum of the output signal.

Figure 24 • Filter Output



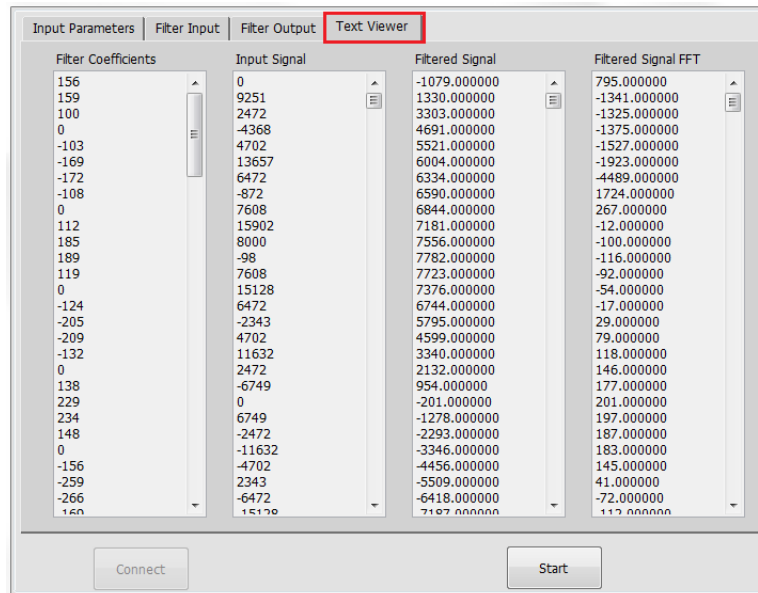
Waveform can be zoomed in or out by using the options provided as shown in the following image.

Figure 25 • Zooming the Filter Output



- The filter coefficients, input signal, output signal, and FFT output data values can be viewed in the **Text viewer** as shown in the following figure.

Figure 26 • The Text Viewer Tab



- To save the coefficients, select the text in **Filter Coefficients** pane, copy and paste in the required location.
- Close the GUI.

This concludes the DSP FIR Filter demo.

6 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_dg0762_df\Programming_Job\CoreFIR

and

mpf_dg0762_df\Programming_Job\FIR_RTL

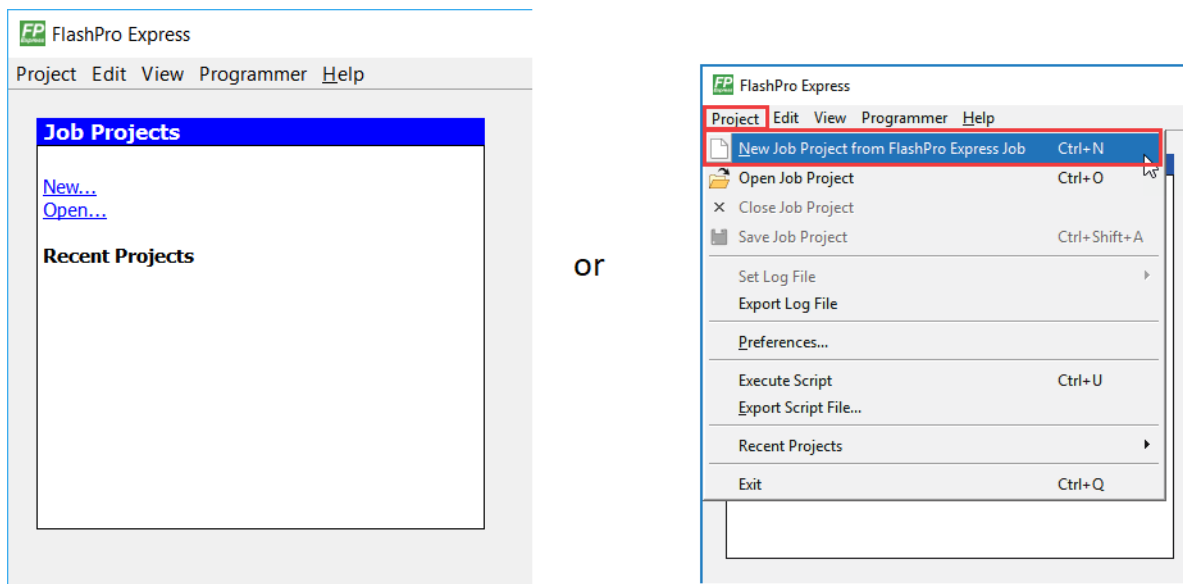
To program the PolarFire device using FlashPro Express, complete the following steps:

1. Ensure that the jumper settings on the board are the same as listed in Table 12, page 17.

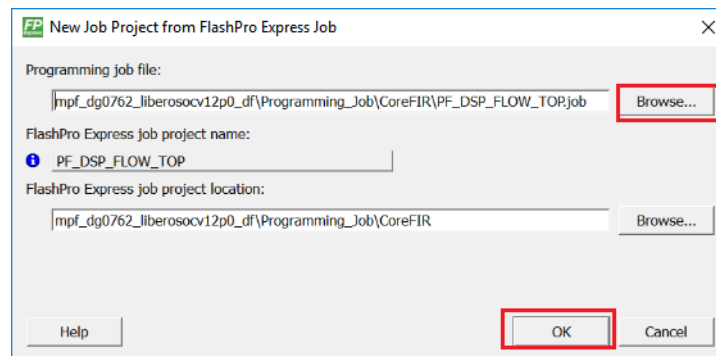
Note: The power supply switch must be switched off while making the jumper connections.

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. 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**, navigate to the location where the .job file is located, and select the file. The default location is: <download_folder>\mpf_dg0762_df\Programming_Job.
 - FlashPro Express job project location: Click **Browse** and navigate to the location where you want to save the project.

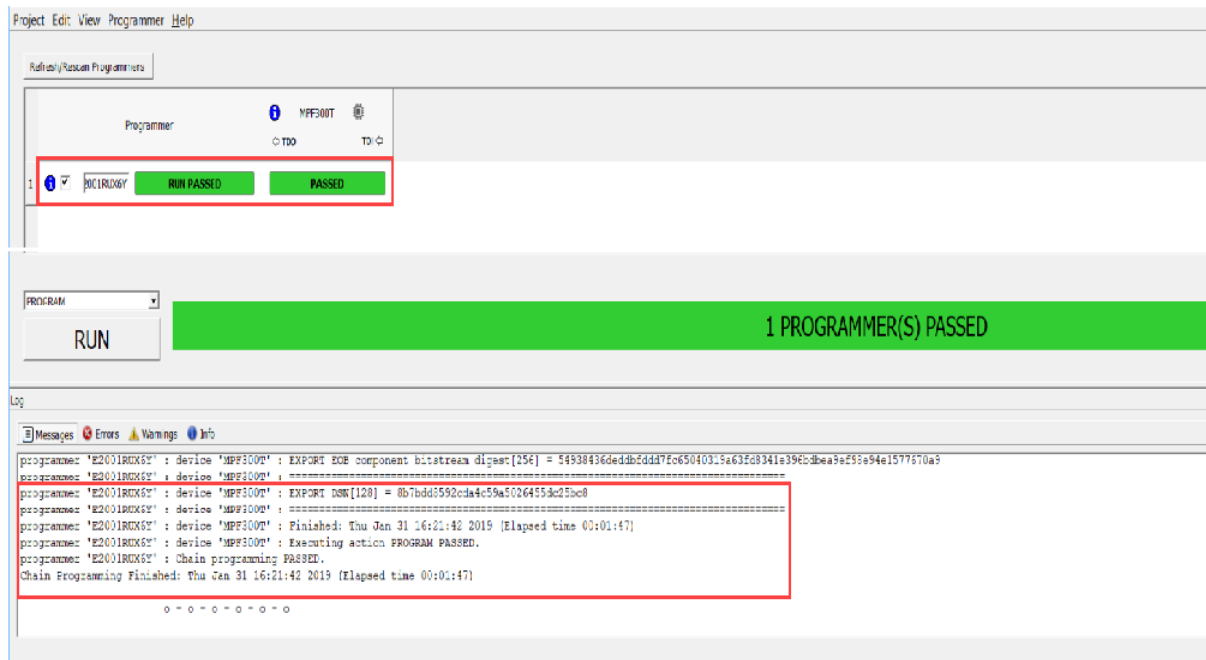
Figure 28 • New Job Project from FlashPro Express Job

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

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

Figure 30 • FlashPro Express—RUN PASSED



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

7 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:

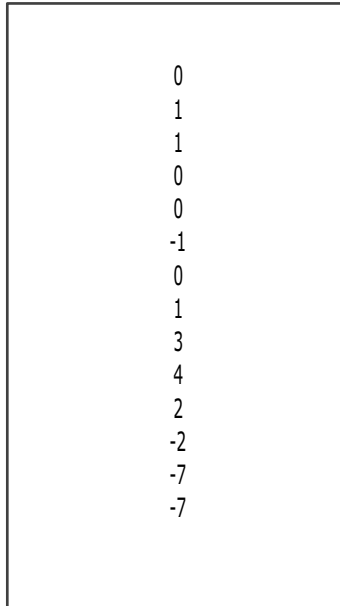
- `mpf_dg0762_df/TCL_Scripts/core_FIR/readme.txt`
- `mpf_dg0762_df/TCL_Scripts/FIR_RTL/readme.txt`

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.

8 Appendix 3: Coefficient Test File Format

The FIR filter coefficients can be loaded from an ASCII text file (*.txt). Create the coefficient file using a text editor. The format of the text file must be as shown in the following figure. The coefficient values are entered as integers. For asymmetric or anti-symmetric filter, only half of the coefficients must be listed in the file (this applies to the fully enumerated type only). Only one coefficient value per line is permitted. An empty line must be placed after the last coefficient of the last set.

Figure 31 • Coefficient File Example - 9 Taps, Decimal Values



```
0
1
1
0
0
-1
0
1
3
4
2
-2
-7
-7
```

9 Appendix 4: References

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

- For more information about PF_TPSRAM, see *UG0680: PolarFire FPGA Fabric User Guide*.
- For more information about PF_CCC, see *UG0684: PolarFire FPGA Clocking Resources User Guide*.
- For more information about CoreFIR, see *CoreFIR Handbook*.
- For more information about Core FFT, see *CoreFFT Handbook*.
- For more information about CoreUART, see *CoreUART Handbook*.
- For more information about Libero, ModelSim, and Synplify Pro, see the *Microsemi Libero SoC PolarFire webpage*.