PolarFire FPGA: Building a Mi-V Processor Subsystem
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 5.0 ................................................................. 1
  1.2 Revision 4.0 ................................................................. 1
  1.3 Revision 3.0 ................................................................. 1
  1.4 Revision 2.0 ................................................................. 1
  1.5 Revision 1.0 ................................................................. 1

2 Building a Mi-V Processor Subsystem ....................................... 2
  2.1 Requirements .............................................................. 2
  2.2 Prerequisites .............................................................. 2
  2.3 Design Description ....................................................... 3
    2.3.1 Fabric RAMs Initialization ........................................ 3
    2.3.2 Creating a Mi-V Processor Subsystem ............................... 4
    2.4.1 Creating a Libero Project ......................................... 4
    2.4.2 Creating a New SmartDesign Component ........................... 5
    2.4.3 Instantiating IP Cores in SmartDesign ............................. 5
    2.4.4 Connecting IP Instances in SmartDesign ......................... 19
    2.4.5 Generating SmartDesign Component ............................... 24
    2.4.6 Managing Timing Constraints ..................................... 24
    2.4.7 Running the Libero Design Flow .................................. 25

3 Building the User Application Using SoftConsole ....................... 35
  3.1 Creating a Mi-V SoftConsole Project .................................. 35
  3.2 Downloading the Firmware Drivers .................................... 37
  3.3 Importing the Firmware Drivers ...................................... 39
  3.4 Creating the main.c File ............................................... 41
  3.5 Mapping Firmware Drivers and the Linker Script .................... 42
  3.6 Mapping Memory and Peripheral Addresses ........................... 48
  3.7 Setting the UART Baud Rate .......................................... 50
  3.8 Building the Mi-V Project ............................................. 50
  3.9 Debugging the User Application Using SoftConsole .................. 51
  3.10 Debugging the User Application from DDR3 Memory ................... 57

4 Appendix ............................................................................. 58
  4.1 References .................................................................... 58
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Download New Cores Option</td>
</tr>
<tr>
<td>2</td>
<td>Block Diagram</td>
</tr>
<tr>
<td>3</td>
<td>New Project Details</td>
</tr>
<tr>
<td>4</td>
<td>Device Selection</td>
</tr>
<tr>
<td>5</td>
<td>Create New SmartDesign</td>
</tr>
<tr>
<td>6</td>
<td>Mi-V Configuration</td>
</tr>
<tr>
<td>7</td>
<td>CoreAXI4Interconnect Configurator – Bus Configuration Section</td>
</tr>
<tr>
<td>8</td>
<td>CoreAXI4Interconnect - Master0 and Master1 Configurations</td>
</tr>
<tr>
<td>9</td>
<td>CoreAXI4Interconnect Configurator – Slave0 Configuration</td>
</tr>
<tr>
<td>10</td>
<td>CoreAXI4Interconnect Configurator – Slave1 Configuration</td>
</tr>
<tr>
<td>11</td>
<td>CoreAXI4Interconnect Configurator – Slave2 Configuration</td>
</tr>
<tr>
<td>12</td>
<td>Crossbar Configuration and Enabling Master Write Access Settings</td>
</tr>
<tr>
<td>13</td>
<td>Enable Master Read Access Settings</td>
</tr>
<tr>
<td>14</td>
<td>PF_SRAM_AHBL_AXI Configurator</td>
</tr>
<tr>
<td>15</td>
<td>Apply Option for MPF300T</td>
</tr>
<tr>
<td>16</td>
<td>DDR3 General Configuration</td>
</tr>
<tr>
<td>17</td>
<td>DDR3 Controller Configuration</td>
</tr>
<tr>
<td>18</td>
<td>CoreAHBLite Configuration</td>
</tr>
<tr>
<td>19</td>
<td>CoreAPB3 Configuration</td>
</tr>
<tr>
<td>20</td>
<td>CoreGPIO Configuration</td>
</tr>
<tr>
<td>21</td>
<td>CoreGPIO Configuration</td>
</tr>
<tr>
<td>22</td>
<td>CCC Configurator Clock Options PLL Tab</td>
</tr>
<tr>
<td>23</td>
<td>CCC Configurator Output Clocks Tab</td>
</tr>
<tr>
<td>24</td>
<td>INIT_MONITOR Configuration</td>
</tr>
<tr>
<td>25</td>
<td>PROC_SUBSYSTEM with All Components Instantiated</td>
</tr>
<tr>
<td>26</td>
<td>Connection Method</td>
</tr>
<tr>
<td>27</td>
<td>Mi-V Subsystem Connected</td>
</tr>
<tr>
<td>28</td>
<td>Edit Slices Window</td>
</tr>
<tr>
<td>29</td>
<td>AHBLite_0 Peripherals Address Map</td>
</tr>
<tr>
<td>30</td>
<td>APB3_0 Peripherals Address Map</td>
</tr>
<tr>
<td>31</td>
<td>Generate Component Icon</td>
</tr>
<tr>
<td>32</td>
<td>Derive Constraints Button</td>
</tr>
<tr>
<td>33</td>
<td>Derived Constraints</td>
</tr>
<tr>
<td>34</td>
<td>I/O Attributes</td>
</tr>
<tr>
<td>35</td>
<td>Edit with I/O Editor Option</td>
</tr>
<tr>
<td>36</td>
<td>Port View</td>
</tr>
<tr>
<td>37</td>
<td>I/O Editor Design View – DDR3 Selection</td>
</tr>
<tr>
<td>38</td>
<td>Memory View [active] Tab with DDR3 Subsystem Placement</td>
</tr>
<tr>
<td>39</td>
<td>DDR3_0 Placed</td>
</tr>
<tr>
<td>40</td>
<td>Fabric RAMs Tab</td>
</tr>
<tr>
<td>41</td>
<td>Edit Fabric RAM Initialization Client Dialog Box</td>
</tr>
<tr>
<td>42</td>
<td>Import Memory File Dialog Box</td>
</tr>
<tr>
<td>43</td>
<td>Fabric RAMs Tab - Apply Button</td>
</tr>
<tr>
<td>44</td>
<td>Design Initialization Data</td>
</tr>
<tr>
<td>45</td>
<td>SPI Flash Tab</td>
</tr>
<tr>
<td>46</td>
<td>COM Port Number</td>
</tr>
<tr>
<td>47</td>
<td>Connection Type Selection</td>
</tr>
<tr>
<td>48</td>
<td>PuTTY Configuration</td>
</tr>
<tr>
<td>49</td>
<td>Hello World String</td>
</tr>
<tr>
<td>50</td>
<td>Hello World String After the Board is Power Cycled</td>
</tr>
<tr>
<td>51</td>
<td>Workspace Launcher</td>
</tr>
<tr>
<td>52</td>
<td>New C Project Creation</td>
</tr>
<tr>
<td>53</td>
<td>C Project Dialog Box</td>
</tr>
<tr>
<td>54</td>
<td>Select Configurations Dialog Box</td>
</tr>
</tbody>
</table>
Figure 55  Empty Mi-V Project ................................................................. 37
Figure 56  Firmware Catalog Window .................................................. 38
Figure 57  RISCV HAL Files Report ...................................................... 38
Figure 58  CoreUARTapb Files Report .................................................... 38
Figure 59  CoreGPIO Files Report ......................................................... 39
Figure 60  CoreSPI Driver Files Report ................................................ 39
Figure 61  Import Option ................................................................. 39
Figure 62  Import Dialog Box ............................................................. 40
Figure 63  Import Dialog Box - Page 2 .................................................. 40
Figure 64  main.c File Creation ............................................................ 41
Figure 65  The main.c file ................................................................. 41
Figure 66  Unresolved Header Files ...................................................... 42
Figure 67  C/C++ Build Settings .......................................................... 42
Figure 68  Target Processor Tool Settings ............................................. 43
Figure 69  GNU RISC-V Cross C Compiler Tool Settings ....................... 44
Figure 70  Add Directory Path Dialog Box ............................................. 44
Figure 71  CoreGPIO Folder Selection .................................................. 45
Figure 72  Tool Settings Tab with CoreGPIO Path Added ......................... 45
Figure 73  Tool Settings Tab After Successful Mapping .......................... 46
Figure 74  Selecting the Linker Script ................................................... 46
Figure 75  Linker Script Default Mapping ............................................. 47
Figure 76  RISC-V Flash Image Settings ............................................... 48
Figure 77  Linker Script ................................................................. 49
Figure 78  Updated sample_hw_platform.h File ..................................... 49
Figure 79  System Clock Frequency Definition ..................................... 50
Figure 80  Hex File ................................................................. 50
Figure 81  Debug Icon ................................................................. 51
Figure 82  Create, manage, and run configurations Window – Main Tab .... 51
Figure 83  MiV_uart_blinky.elf Selection .............................................. 52
Figure 84  Create, manage, and run configurations Window – Debugger Tab 52
Figure 85  Debug Settings- Startup Tab ................................................. 53
Figure 86  Confirm Perspective Switch Dialog Box ................................ 54
Figure 87  First Instruction in the main.c File ....................................... 54
Figure 88  Resume Application Execution ............................................ 55
Figure 89  Hello World in Debug Mode ............................................... 55
Figure 90  Mi-V Register Values ......................................................... 56
Figure 91  Variable Values ............................................................. 56
Figure 92  RAM Start Address Parameters .......................................... 57
Figure 93  Debugging from DDR3 ....................................................... 57
Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Tutorial Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>CCC_0_0 Pin Connections</td>
<td>20</td>
</tr>
<tr>
<td>Table 3</td>
<td>DEBUG_TARGET Pin Connections</td>
<td>21</td>
</tr>
<tr>
<td>Table 4</td>
<td>AXI4_Interconnect_0 Pin Connections</td>
<td>22</td>
</tr>
<tr>
<td>Table 5</td>
<td>APB3_0 Pin Connections</td>
<td>22</td>
</tr>
<tr>
<td>Table 6</td>
<td>Jumper Settings</td>
<td>31</td>
</tr>
</tbody>
</table>
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 5.0
The following is a summary of the changes made in this revision.

- Updated for Libero SoC v12.2.
- Updated the design for AXI-based Mi-V Soft Processor for an enhanced performance with DDR memories.
- Removed Libero SoC and SoftConsole version numbers.

1.2 Revision 4.0
The following is a summary of the changes made in this revision.

- Added Fabric RAMs Initialization, page 3.
- The document was updated for Libero SoC v12.0.

1.3 Revision 3.0
The following is a summary of the changes made in this revision.

- Added Design Description, page 3.
- The document was updated for Libero SoC PolarFire v2.1.

1.4 Revision 2.0
The following is a summary of the changes made in this revision.

- The document was updated for the Mi-V processor upgrade.
- The document was updated for Libero SoC PolarFire v2.0 and SoftConsole v5.2. For more information, see Building the User Application Using SoftConsole, page 35.
- Information about LSRAM initialization from external SPI flash was added. For more information, see Configure Design Initialization Data and Memories, page 27.

1.5 Revision 1.0
The first publication of this document.
Building a Mi-V Processor Subsystem

Microchip offers the Mi-V processor IP and software toolchain free of cost to develop RISC-V processor-based designs. RISC-V, a standard open instruction set architecture (ISA) under the governance of the RISC-V foundation, offers numerous benefits, which include enabling the open source community to test and improve cores at a faster pace than closed ISAs.

PolarFire® FPGAs support Mi-V soft processors to run user applications. The objective of the tutorial is to build a Mi-V processor subsystem that can execute an application from the designated fabric RAMs initialized from the sNVM/SPI Flash. The tutorial also describes how to build a RISC-V application using SoftConsole and run it on a PolarFire Evaluation Board.

2.1 Requirements

The following table lists the tutorial requirements for building a Mi-V processor subsystem.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td></td>
</tr>
<tr>
<td>Host PC</td>
<td>Windows 7, 8.1, or 10</td>
</tr>
<tr>
<td>POLARFIRE-EVAL-KIT (MPF300TS-FCG1152I)</td>
<td>Rev D or later</td>
</tr>
<tr>
<td>- PolarFire Evaluation Board</td>
<td></td>
</tr>
<tr>
<td>- 12 V/5 A AC power adapter and cord</td>
<td></td>
</tr>
<tr>
<td>- USB 2.0 A to mini-B cable</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>Libero SoC Design Suite</td>
<td>See the readme.txt file provided in the design files for all software versions needed to create this reference design.</td>
</tr>
<tr>
<td>Firmware Catalog</td>
<td>See the readme.txt file provided in the design files for all software versions needed to create this reference design.</td>
</tr>
<tr>
<td>SoftConsole</td>
<td></td>
</tr>
</tbody>
</table>

PuTTY (serial terminal emulation program)

1. Firmware catalog is included in the installation package of Libero SoC.

2.2 Prerequisites

1. Download the design files from:
   http://soc.microsemi.com/download/rsc/?f=mpf_tu0775_df
   The design files folder contains the following folders:
   - Programming_Job: Contains the programming file (.job) for reference
   - Solution: Contains the final Libero and SoftConsole projects for reference
   - Source: Contains the source files required to complete this tutorial
2. Download and install Libero SoC from:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads
3. Download and install SoftConsole from:
   https://www.microsemi.com/products/fpga-soc/design-resources/design-software/softconsole#downloads
4. From the Libero Catalog, download the latest versions of the IP cores from the warning pop-up as shown in the following figure.

*Figure 1 • Download New Cores Option*

### 2.3 Design Description

The tutorial describes how to create a Mi-V subsystem for executing user applications. The user application can be stored in µPROM, sNVM, or an external SPI flash. At device power-up, the PolarFire System Controller initializes the designated LSRAMs with the user application and releases the system reset. If the user application is stored in SPI Flash, the System Controller uses the SC_SPI interface for reading the user application from SPI Flash. The given user application prints the UART message “Hello World!” and blinks user LEDs on the board.

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

*Figure 2 • Block Diagram*

#### 2.3.1 Fabric RAMs Initialization

Each logical RAM instance in the design can be initialized from a different source—sNVM, µPROM, or SPI-Flash. The initialization client storage location is configurable. Generate the initialization data to add the initialization clients to the chosen non-volatile memories and program the device. Program SPI-Flash, if chosen as storage location for initialization data. For more information, see Configure Design Initialization Data and Memories, page 27.

*Note:* Libero SmartDesign and configuration screen shots shown in this tutorial are for illustration purpose only. Open the Libero project to see the latest updates and IP versions.
2.4 Creating a Mi-V Processor Subsystem

Creating a Mi-V processor subsystem involves:

- Creating a Libero Project, page 4
- Creating a New SmartDesign Component, page 5
- Instantiating IP Cores in SmartDesign, page 5
- Connecting IP Instances in SmartDesign, page 19
- Generating SmartDesign Component, page 24
- Managing Timing Constraints, page 24
- Running the Libero Design Flow, page 25

This section describes all of the steps required to create a Mi-V processor subsystem on a new SmartDesign canvas.

2.4.1 Creating a Libero Project

Follow these steps to create a Libero project:

1. On the Libero Menu bar, click **Project > New Project**.
2. Enter the following details, and click **Next**.
   - Project name: PF_Mi_V_Tut
   - Project location: For example, F:/Libero_Projects
   - Preferred HDL type: Verilog

Figure 3 • New Project Details
3. To choose the PolarFire device present on the PolarFire Evaluation Board, select the following settings in the Device Selection window, and click Next.
   • Family: PolarFire
   • Die: MPF300TS
   • Package: FCG1152
   • Speed: -1
   • Range: IND
   • Part Number: MPF300TS-1FCG1152I

   ![Device Selection](image)

4. In the Device Settings window, click Next to retain the default core voltage and I/O settings.
5. In the Add HDL Sources window, click Next to retain the default settings.
6. In the Add constraints window, click Import file to import the I/O constraint file.
7. In the Import files window, locate the io_constraints.pdc file in the DesignFiles_directory\Source\io folder, and double-click it.
8. Click Finish.
   The Log pane displays a message indicating that the PF_Mi_V_Tut project was created.

2.4.2 Creating a New SmartDesign Component

To create a new SmartDesign component:

1. In Libero, select File > New > SmartDesign.
2. In the Create New SmartDesign dialog box, enter PROC_SUBSYSTEM as the name of the new SmartDesign project, as shown in the following figure.

   ![Create New SmartDesign](image)

3. Click OK.
   The PROC_SUBSYSTEM SmartDesign component is created.

2.4.3 Instantiating IP Cores in SmartDesign

When an IP core is dragged from the Catalog to SmartDesign, Libero prompts you to name the component, and if applicable, to configure the IP core. After the core is configured, Libero generates the component for that core and instantiates it in SmartDesign.
2.4.3.1 Instantiating Mi-V Processor IP
1. From the Catalog, drag the Mi-V RV32IMA_L1_AXI to SmartDesign. In this tutorial AXI version of the Mi-V core is used for optimum performance.
2. In the Create Component dialog box, enter MiV_AXI as the component name, and click OK.
3. In the Configurator, set the following configuration:
   • Set Reset Vector Address -> Upper 16 bits (Hex) to 0x8000 and retain the default setting for Lower 16 Bits (Hex). This is the address the processor will start executing from after a reset. The processor's main memory must be accessible to Mi-V AXI memory interface whose memory-mapped address ranges from 0x80000000 to 0x8FFFFFFF. The Mi-V memory interface supports cached transactions, whereas Mi-V MMIO interface does not support them.
   • Set the MASTER_TYPE to AXI4.
4. Retain the default settings for MEM_WID, and MMIO_WID options.

   Figure 6 • Mi-V Configuration

2.4.3.2 Instantiating AXI Interconnect Bus IP
The AXI interconnect bus must be configured to connect the Mi-V core with memory and peripherals.
1. From the Catalog, drag the CoreAXI4Interconnect IP core to SmartDesign.
2. In the Create Component dialog box, enter AXI4_Interconnect as the component name, and click OK.
   The Configurator opens.
3. In the Bus Configuration section, configure the AXI4_Interconnect IP to have three slaves with an ID width of 4, as shown in the following figure. Leave the rest as defaults.

   Figure 7 • CoreAXI4Interconnect Configurator – Bus Configuration Section
4. In the Master Configuration section, retain the following Master0 and Master1 default settings:
   • M0 Type: AXI4
   • M0 Data Width: 64 bits
   • M0 DWC Data FIFO Depth: 16
   • M0 Register Slice: Selected
Building a Mi-V Processor Subsystem

- **M1 Type:** AXI4
- **M1 Data Width:** 64 bits
- **M1 DWC Data FIFO Depth:** 16
- **M1 Register Slice:** Selected

The Master0 port must be connected to the Mi-V AXI MEM interface and Master1 port must be connected to the Mi-V AXI MMIO interface. The following figure shows the Master0 and Master1 configurations.

*Figure 8* • CoreAXI4Interconnect - Master0 and Master1 Configurations

5. In the *Slave Configuration* section, configure the Slave0 port as follows:
   - **S0 SLAVE Start Address (Lower 32 bits):** 0x80000000
   - **S0 SLAVE End Address (Lower 32 bits):** 0x8000FFFF
   - **S0 Clock Domain Crossing:** Disabled
   - Leave the rest as defaults

The S0 slot is used to connect 64KB of internal LSRAMs through AXI4 interface. The address range for the S0 slot is set to 0x80000000 - 0x8000FFFF which is accessible by Mi-V processor through MEM interface. The Mi-V MEM interface address ranges from 0x80000000-0x8FFFFFFF.

*Figure 9* • CoreAXI4Interconnect Configurator – Slave0 Configuration

6. In the *Slave Configuration* section, configure Slave1 port as follows:
   - **S1 SLAVE Start Address (Lower 32 bits):** 0x80010000
   - **S1 SLAVE End Address (Lower 32 bits):** 0x80FFFFFF
   - **S1 Clock Domain Crossing:** Enabled
   - Leave the rest as defaults

The S1 slot is used to connect an external DDR3 memory through a DDR3 controller IP. The address range for the S1 slot is set to 0x80010000 - 0x80FFFFFF which is accessible by Mi-V processor through MEM interface. The Mi-V processor MEM interface operates at 111.111 MHz whereas the DDR3 controller AXI4 interface operates at 166.666 MHz. Hence the **S1 Clock Domain Crossing** option is enabled to handle the data transfers between two clock domains.
7. In the **Slave Configuration** section, configure Slave2 port as follows:
   - S2 Type: AXI3
   - S2 SLAVE Start Address (Lower 32 bits): 0x60000000
   - S2 SLAVE End Address (Lower 32 bits): 0x60FFFFFF
   - S2 Clock Domain Crossing: Disabled
   - Leave the rest as defaults

   The S2 slot is used to connect the peripherals to the Mi-V processor. The address range for the S2 slot is set to 0x60000000 - 0x60FFFFFF, which is accessible by Mi-V processor through the MMIO interface. The Mi-V MMIO interface address ranges from 0x60000000-0x60FFFFFF.

8. In the **Crossbar Configuration** section, ensure that the following options are set:
   - Under **Enable Master Write Access**, enable M0 access S0, M0 access S1, and M0 access S2.
   - Under **Enable Master Write Access**, enable M1 access S0, M1 access S1, and M1 access S2.
   - Under **Enable Master Read Access**, enable M0 access S0, M0 access S1, and M0 access S2.
   - Under **Enable Master Read Access**, enable M1 access S0, M1 access S1, and M1 access S2.
   - Leave the rest as defaults.
2.4.3.3 Instantiating On-chip SRAM

1. From the Catalog, drag the PolarFire SRAM (AHBLite and AXI) IP core to SmartDesign.
2. In the Create Component dialog box, enter LSRAM as the component name, and click OK.
3. In the Port Settings tab of the PF_SRAM_AHBL_AXI Configurator, select the following settings:
   - Memory Depth (in words): 8192 (this creates 64 KB (8192 × 8 bytes) of memory)
   - Fabric Interface type: AXI
   - Data Width: 64
   - Width of ID: 5
   - Wrap Bust support: Enabled
   - Leave the rest as defaults
4. Click OK.

2.4.3.4 Instantiating DDR3 Memory Controller

This tutorial demonstrates how to build and debug an application from DDR3 memory. Executing an application from DDR3 memory in the release mode requires a bootloader. The bootloader use case is not in the scope of this tutorial.

1. From the Catalog, drag the PolarFire DDR3 IP core to SmartDesign.
2. In the Create Component dialog box, enter DDR3_0 as the component name, and click OK.
3. In the left pane of the Configurator, expand Microsemi PolarFire Evaluation Kits > PolarFire Evaluation Kit > MPF300T.
4. Left-click MT41K1G8SN-125, and click Apply, as shown in the following figure. This configures the DDR3 controller with the initialization and timing parameters of the DDR3 memory (MT41K1G8SN-125) present on the PolarFire Evaluation Kit.

5. On the General tab, set the CCC PLL Clock Multiplier to 6, and the DQ Width to 16, as shown in Figure 16, page 11. The clock multiplier value of 6 sets the CCC PLL reference clock frequency to 111.111 MHz. A reference clock of this frequency is required for the PLL present inside the DDR3 subsystem. The PLL generates a 666.666 MHz DDR3 memory clock frequency and a 166.666 MHz DDR3 AXI clock frequency.
The DQ width is set to 16 to match the width of the DDR3 memory present on the board.

*Figure 16 • DDR3 General Configuration*
6. On the **Controller** tab, ensure that the settings are as follows:
   - Instance Number: 0
   - Fabric Interface: AXI4
   - AXI ID Width: 4
   - Enable Rank0 - ODT0 check box: Selected

   *Figure 17 • DDR3 Controller Configuration*

7. Retain the default settings for others tabs and click **OK**.
2.4.3.5 Instantiating the AXI3 to AHB-Lite Bridge

The CoreAXItoAHBL IP connects an AXI bus to an AHB-Lite bus, enabling an AXI master to communicate with an AHBL slave/subsystem. To instantiate the CoreAXItoAHBL IP:

1. From the Catalog, drag the CoreAXItoAHBL IP core to SmartDesign.
2. In the Create Component dialog box, enter CORAXItoAHBL_0 as the component name, and click OK.
3. In the CoreAXItoAHBL Configurator, retain the default configuration, and click OK.

2.4.3.6 Instantiating the AHB-Lite Bus

1. From the Catalog, drag the CoreAHBLite IP core to SmartDesign.
2. In the Create Component dialog box, enter COREAHBLITE_0 as the component name, and click OK.
3. In the CoreAHBLite Configurator, do the following:
   - From the Memory space list, select 4GB addressable space apportioned into 16 slave slots, each of size 256MB.
   - Under Enable Master Access, select M0 can access slot 6.
   - This configuration sets the slave address map to 0x60000000-0x6FFFFFFF. The peripherals must be connected on this slave interface for the Mi-V processor to access them.

![CoreAHBLite Configuration](image)

4. Click OK.

2.4.3.7 Instantiating the AHB-Lite to APB3 Bridge

The AHB-Lite to APB3 Bridge connects APB peripherals such as UART, SPI and GPIO to AHB-Lite masters. To instantiate the AHB-Lite to APB3 Bridge:

1. From the Catalog, drag the CoreAHBLtoAPB3 IP core to SmartDesign.
2. In the Create Component dialog box, enter COREAHBLtoAPB3_0 as the component name, and click OK.
3. In the CoreAHBtoAPB3 Configurator, retain the default configuration, and click **OK**.

### 2.4.3.8 Instantiating APB3 Bus

1. From the Catalog, drag the **CoreAPB3** IP core to SmartDesign.
2. In the **Create Component** dialog box, enter **APB3** as the component name, and click **OK**.
3. In the CoreAPB3 Configurator, select the following data width and address configuration settings, as shown in the following figure:
   - **APB Master Data Bus Width**: 32-bit
   - **Number of address bits driven by master**: 16
   - **Position in slave address of upper 4 bits of master address**: [27:24] (Ignored if master address width >= 32 bits)
   - **Enabled APB Slave Slots**: Slot 0, Slot 1, and Slot 2.

This configuration sets the slave address map as follows:
- Slot0: 0x0000 - 0x0FFF
- Slot1: 0x1000 - 0x1FFF
- Slot2: 0x2000 - 0x2FFF

*Figure 19 • CoreAPB3 Configuration*

4. Click **OK**.
2.4.3.9  Instantiating UART Controller
1. From the Catalog, drag the CoreUARTapb IP core to SmartDesign.
2. In the Create Component dialog box, enter UART_apb as the component name, and click OK.
3. In the CoreUARTapb Configurator, retain the default configuration, and click OK.

2.4.3.10 Instantiating the GPIO Controller
1. From the Catalog, drag the CoreGPIO IP core to SmartDesign.
2. In the Create Component dialog box, enter CoreGPIO_0 as the component name, and click OK.
3. In the CoreGPIO Configurator, select the following Global Configuration settings, as shown in the following figure:
   • APB Data Width: 32
   • Number of I/Os: 4
   • Single-bit interrupt port: Disabled
   • Output enable: Internal
4. Under I/O bit 0, I/O bit 1, I/O bit 2, and I/O bit 3, do the following, as shown in the following figure:
   • Select Fixed Config.
   • Set the I/O type as Output.
   • Select the interrupt type as Disabled.
Four GPIO outputs are configured.

Figure 20 • CoreGPIO Configuration

5. Click OK to close the CoreGPIO Configurator.
2.4.3.11 Instantiating CoreSPI

The PolarFire Evaluation board contains two SPI Flash memories. One SPI Flash is connected to the System Controller SPI interface (SC_SPI) for design initialization. The CoreSPI IP is used to interface with the other SPI Flash, which is connected to the fabric I/Os. To instantiate CoreSPI:

1. From the Catalog, drag the CoreSPI IP core to SmartDesign.
2. In the Create Component dialog box, enter SPI_Controller as the component name, and click OK.
3. In the CoreSPI Configurator, do the following:
   - Set the APB Data Width to 32
   - In the SPI Configuration section, set the mode to Motorola, frame size to 8, FIFO depth to 32, and clock rate to 16.
   - In the Motorola Configuration section, set the mode to Mode 0, and select the Keep SSEL active check box.

4. Click OK.

Figure 21 • CoreSPI Configuration
2.4.3.12 Instantiating PolarFire Clock Conditioning Circuitry (CCC)

The PolarFire Clock Conditioning Circuitry (CCC) block generates a 111.111 MHz clock to the processor subsystem, which is used as a reference clock to the DDR3_0_0 PLL. To instantiate the CCC block:

1. From the Catalog, drag the Clock Conditioning Circuitry (CCC) core to SmartDesign.
2. In the Create Component dialog box, enter CCC_0 as the component name, and click OK.
3. In the Configurator, set the configuration to PLL-Single.
4. In the Clock Options PLL tab, do the following:
   • Set the input frequency to 50 MHz.
   • Under Power/Jitter, select Maximize VCO for Lowest Jitter.
   • Set the feedback mode to Post-VCO.
   • Set the Bandwidth to High.

*Figure 22 • CCC Configurator Clock Options PLL Tab*
5. In the **Output Clocks** tab, under the **Output Clock 0** section, do the following:
   - Select the **Enabled** check box to enable PLL output 0.
   - Set the requested frequency to 111.111 MHz.
   - Select the **Global Clock** check box.

*Figure 23 • CCC Configurator Output Clocks Tab*

6. Click OK and acknowledge the pop-up.

### 2.4.3.13 Instantiating PolarFire Initialization Monitor

The PolarFire Initialization Monitor is used to get the status of device initialization including the LSRAM initialization. To instantiate the PolarFire Initialization Monitor:

1. From the Catalog, drag the **PolarFire Initialization Monitor** core to SmartDesign.
2. In the **Create Component** dialog box, enter **INIT_Monitor** as the component name, and click OK.
3. In the **INIT_MONITOR Configurator > Bank Monitor** tab, clear all the check boxes under **Calibration Monitor** except for **BANK1_CALIB_STATUS**, and click OK.

*Figure 24 • INIT_MONITOR Configuration*
2.4.3.14 Instantiating CORERESET_PF

Two instances of the CORERESET_PF IP are required in this design.

1. From the Catalog, drag the CORERESET_PF IP.
2. In the Component Name dialog box, enter reset_syn_0 as the name of this component, and click OK.
3. Retain the default configuration for this IP and click OK.
4. Similarly, instantiate another instance with reset_syn_1 as its name.

2.4.3.15 Instantiating CoreJTAGDebug

The CoreJTAGDebug IP connects the Mi-V soft processor to the JTAG header for debugging. To instantiate CoreJTAGDebug:

1. From the Catalog, drag the CoreJTAGDebug IP core to SmartDesign.
2. In the Create Component window, enter COREJTAGDebug_0 as the component name, and click OK.
3. In the Configurator, retain the default configuration, and click OK.

The following figure shows the PROC_SUBSYSTEM in SmartDesign after all the components are instantiated.

Figure 25 • PROC_SUBSYSTEM with All Components Instantiated

2.4.4 Connecting IP Instances in SmartDesign

Connect the IP blocks in SmartDesign using any of the following methods:

- Using the Connection Mode icon: You can initiate the connection mode in SmartDesign by clicking the Connection Mode icon in the SmartDesign toolbar, as shown in the following figure. The cursor changes from a normal arrow to the shape of the connection mode icon. To make a connection in this mode, click the first pin and drag it to the second pin that you want to connect.

Figure 26 • Connection Method

- Using the Connect option in the Context menu: You can also connect pins by selecting the pins, and then selecting Connect from the context menu. To connect multiple pins, hold down the Ctrl key while selecting the pins. Right-click the input source signal, and select Connect. To disconnect signals, right-click the input source signal, and select Disconnect.
- Right-clicking on a pin provides a list of options like Mark Unused, Edit Slice, Tie Low, Promote to Top-Level, and Tie High. Use these options for individual pins settings.
Figure 27, page 20 shows the Mi-V subsystem in SmartDesign with all IP blocks connected and top-level I/Os.

**Figure 27 • Mi-V Subsystem Connected**

Note: Grayed out pins are marked unused, green pins are tied Low, and red pins are tied High. Ensure that unused, tied-low, and tied-high pins are strictly set as per Figure 27, page 20.

Follow these steps to connect the IP blocks as per Figure 27, page 20:

1. Set the pins as follows on INIT_MONITOR_0:
   - Select FABRIC_POR_N, PCIE_INIT_DONE, USRAM_INIT_DONE, SRAM_INIT_DONE, XCVR_INIT_DONE, USRAM_INIT_FROM_SNVM_DONE, USRAM_INIT_FROM_UPROM_DONE, USRAM_INIT_FROM_SPI_DONE, SRAM_INIT_FROM_SNVM_DONE, SRAM_INIT_FROM_UPROM_DONE, SRAM_INIT_FROM_SPI_DONE, and AUTOCALIB_DONE pins.
   - Right-click the pins, and select Mark Unused.
   - Connect the DEVICE_INIT_DONE pin to reset_syn_0_0:INIT_DONE and BANK_1_CALIB_STATUS pin to reset_syn_1_0:INIT_DONE.

2. Set the pins as follows on CCC_0_0:
   - Right-click the REF_CLK_0 pin, and select Promote to Top Level.
   - Connect the other pins as specified in the following table:

<table>
<thead>
<tr>
<th>Connect From</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL_LOCK_0</td>
<td>reset_syn_0_0:PLL_LOCK and reset_syn_1_0:PLL_LOCK</td>
</tr>
</tbody>
</table>
3. Set the pins of reset_syn_0_0 as follows:
   - Connect EXT_RST_N pin to DDR3_0_0:CTRLR_READY.
   - Right-click SS_BUSY and FF_US_RESTORE pins and tie them low select Tie Low.

4. Connect the reset_syn_0_0:FABRIC_RESET_N to the following pins:
   - MiV_AXI_0:RESETN
   - AXI4_Interconnect_0:ARESETN
   - LSRAM_0:ACLK
   - CORAXITOAHBL_0_0:ACLK
   - CORAXITOAHBL_0_0:HCLK
   - DDR3_0_0:PLL_REF_CLK
   - SPI_Controller_0:PCLK
   - COREAHBLITE_0_0:HCLK
   - COREAHBTOAPB3_0_0:HCLK
   - UART_apb_0:PCLK
   - COREGPIO_0:PCLK
   - AXI4_Interconnect_0:ACLK

   Note: As DDR3_0_0:CTRLReady pin is connected to reset_syn_0_0:EXT_RST_N, the Mi-V processor is held in reset until the DDR3 controller is ready. The rest of the system is out of reset as soon as device initialization is done.

5. Set the pins of reset_syn_1_0 as follows:
   - Right-click SS_BUSY and FF_US_RESTORE pins and tie them low using the Tie Low option.
   - Select the EXT_RST_N pin and promote it to top level and rename it to resetn.
   - Connect the FABRIC_RESET_N pin to DDR3_0_0:SYS_RESET_N.

6. Set the pins as follows on COREJTAGDebug_0_0:
   - Expand JTAG HEADER.
   - Right-click the TDI, TCK, TMS, and TRSTB pins, and select Promote to Top Level.
   - Expand JTAG HEADER.
   - Right-click the TDO pin, and select Promote to Top Level.
   - Connect the other pins as specified in the following table

Table 3 • DEBUG_TARGET Pin Connections

<table>
<thead>
<tr>
<th>Connect From</th>
<th>Connect to</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TCK</td>
<td>MiV_AXI_0:TCK</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TSRT</td>
<td>MiV_AXI_0:TRST</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TMS</td>
<td>MiV_AXI_0:TMS</td>
</tr>
</tbody>
</table>
7. Set the pins as follows on MiV_AXI_0:
   • Right-click the IRQ[30:0] pin, and select **Tie Low**.
   • Right-click the DRV_TDO pin, and select **Mark Unused**.
   • Connect MEM_MST_AXI4 to AXI4_Interconnect_0:AXI4mmaster0.
   • Connect MMIO_MST_AXI4 to AXI4_Interconnect_0:AXI4mmaster1.
8. Connect the AXI4_Interconnect_0 pins as specified in the following table.

<table>
<thead>
<tr>
<th>AXI4_Interconnect_0 Pin Name</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_CLK1</td>
<td>DDR3_0_0:SYS_CLK</td>
</tr>
<tr>
<td>AXI4mslave0</td>
<td>LSRAM_0:AXI4_Slave</td>
</tr>
<tr>
<td>AXI4mslave1</td>
<td>DDR3_0_0:AXI4slave0</td>
</tr>
<tr>
<td>AXI3mslave2</td>
<td>CORAXITOAHBL_0_0:AXISlaveIF</td>
</tr>
</tbody>
</table>

9. Connect CORAXITOAHBL_0_0:AHBMasterIF to COREAHBLITE_0_0:AHBmmaster0.
10. Set the pins as follows on COREAHBLITE_0_0:
    • Right-click the REMAP_M0 pin, and select **Tie Low**.
    • Connect AHBMslave6 to COREAHBTOAPB3_0_0:AHBslave.
11. Connect COREAHBTOAPB3_0_0:APBmaster to APB3_0:APB3mmaster.
12. Connect the APB3_0 pins as specified in the following table.

<table>
<thead>
<tr>
<th>APB3_0 Pin Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect From</td>
</tr>
<tr>
<td>APB3_0:APBmslave0</td>
</tr>
<tr>
<td>APB3_0:APBmslave1</td>
</tr>
<tr>
<td>APB3_0:APBmslave2</td>
</tr>
</tbody>
</table>

13. Set the pins as follows on DDR3_0_0:
    • Right-click the PLL_LOCK output pin, and select **Mark Unused**.
    • Right-click the CTRLR_READY pin, and select **Promote to Top Level** for debug purpose. The CTRLR_READY signal is used to monitor the status of the DDR controller.
14. Set the pins as follows on SPI_Controller_0:
    • Right-click the SPISSI pin, and select **Tie High**.
    • Right-click the SPICLKI pin, and select **Tie Low**.
    • Right-click the SPIINT, SPIRXAVAIL, SPITXRFM, SPIOEN, and SPIMODE pins, and select **Mark Unused**.
    • Right-click the SPISDI, SPISCCLKO and SPISDO pins, and select **Promote to Top Level**.
15. Right-click the SPISS[7:0] pin, select **Edit Slices**, and edit the slices shown in the following figure.

   **Note:** In this tutorial, a single SPI Flash is used. Hence, while settings the pins of the SPI_Controller_0 block, we need only 0th bit of the SPISS. Bits 1:7 need to be sliced and marked as unused.
16. Set the pins as follows on UARTapb_0:
   - Right-click the RX and TX pins, and select **Promote to Top Level**.
   - Right-click the TXRDY, RXRDY, PARITY_ERR, OVERFLOW, FRAMING_ERR pins, and select **Mark Unused**.

17. Set the pins as follows on GPIO_0:
   - Right-click the GPIO_IN[3:0] pin, and select **Tie Low**.
   - Right-click the INT[3:0] pin, and select **Mark Unused**.
   - Right-click the GPIO_OUT[3:0] pin, and select **Promote to Top Level**.

18. Right-click the PROC_SUBSYSTEM SmartDesign canvas, and select **Auto Arrange Layout**.

The IP blocks are successfully connected. Figure 27, page 20 shows all the IP blocks of the Mi-subsystem connected.

After the Mi-V processor subsystem is successfully designed in SmartDesign, you can view the system address map by right-clicking the SmartDesign canvas and selecting **Modify Memory Map**. The following figure shows the address map for the COREAHBLITE_0 peripherals.

The following figure shows the address map for the APB3_0 peripherals.
Figure 30 • APB3_0 Peripherals Address Map

Note: The AXI memory map is not supported. For more information about the AXI memory map, see Instantiating AXI Interconnect Bus IP, page 6.

2.4.5 Generating SmartDesign Component

To generate the SmartDesign component:

1. In Design Hierarchy, right-click PROC_SUBSYSTEM, and select Set As Root.
2. Save the project.
3. Click the Generate Component icon (shown in the following figure) on the SmartDesign toolbar.

Figure 31 • Generate Component Icon

When the Mi-V component is generated, the Message window displays the message, “The PROC_SUBSYSTEM was generated successfully.”

2.4.6 Managing Timing Constraints

Before running the Libero design flow, you must derive the timing constraints and import the JTAG and asynchronous clocking constraints.

2.4.6.1 Deriving Constraints

To derive constraints:

2. In the Manage Constraints window, select the Timing tab, and click Derive Constraints, as shown in the following figure.

Figure 32 • Derive Constraints Button

The design hierarchy is built, and the PROC_SUBSYSTEM_derived_constraints.sdc file is generated in the project folder.

In the dialog box that appears, click Yes to associate the SDC file to the Synthesis, Place and Route, and Timing Verification tools, as shown in the following figure.
2.4.6.2 Importing Other Constraint Files

The JTAG clock constraint and the asynchronous clocks constraint must be imported. These constraints (.sdc) files are available in the DesignFiles_directory\Source folder.

To import and map the constraint files:

1. On the Timing tab, click Import.
2. Navigate to the DesignFiles_directory\Source folder, and select the timing_user_constraints.sdc file.
3. Select the Synthesis, Place and Route, and Timing Verification check boxes next to the timing_user_constraints.sdc file.
   This constraint file defines that the CCC_0_0 output clock and DDR3_0_0 AXI clock are asynchronous clocks.
4. Save the project.

2.4.7 Running the Libero Design Flow

This section describes the Libero design flow, which involves the following steps:

- Synthesis, page 25
- Place and Route, page 25
- Verify Timing, page 27
- Generate FPGA Array Data, page 27
- Configure Design Initialization Data and Memories, page 27
- Generate Design Initialization Data, page 30
- Generate Bitstream, page 31
- Run PROGRAM Action, page 31
- Generate SPI Flash Image, page 33
- Run PROGRAM_SPI_IMAGE Action, page 34

After each step is completed, a green tick mark appears next to the step on the Design Flow tab.

2.4.7.1 Synthesis

To synthesize the design:

   When the synthesis is complete, a green tick mark appears next to Synthesize.
2. Right-click Synthesize and select View Report to view the synthesis report in the Reports tab.

2.4.7.2 Place and Route

The place and route process requires the following steps to be completed:

- Selecting the already imported io_constraints.pdc file
- Placing the DDR3_0_0 block using the I/O Editor
- Ensuring all the I/Os are locked

To complete these steps and to place and route the design:

2. On the I/O Attributes tab, select the check box next to the io_constraints.pdc file, as shown in the following figure. The io_constraints.pdc file contains the I/O assignment for reference clock, UART, GPIO, and SPI interfaces, and other top-level I/Os.
3. From the Edit drop-down list, select Edit with I/O Editor, as shown in the following figure.

4. In the I/O Editor, click the Port View [active] tab, and lock the CTRLR_READY port to pin C27, as shown in the following figure. This ensures that the CTRLR_READY port is assigned to pin C27, which is connected to an user LED for debug purposes.

5. To place the DDR3 I/O lanes, In the I/O Editor Design View, click the Port tab in the left pane, and select DDR3, as shown in the following figure.
6. Drag and place the DDR3 subsystem on the NORTH_NE side, as shown in the following figure. The DDR3 memory on the board is connected to DDR I/Os present on the north-east side.

**Figure 38** • Memory View [active] Tab with DDR3 Subsystem Placement

The DDR3 subsystem is placed on the NORTH_NE side, as shown in the following figure.

**Figure 39** • DDR3_0 Placed

7. From I/O Editor Port View tab, check if there are any unlocked I/Os, and lock them as mapped in the io_constraints.pdc file available in the Design_Files_Directory\Source\io folder.

8. Click Save.

9. Close the I/O Editor.

   A user.pdc file is created for DDR3_0_0 block in the Constraint Manager > I/O Attributes and Floor Planner tabs.

   **Note:** DDR3_0_0 can also be placed using the fp_constraints.pdc. Import the fp_constraints.pdc from Constraint Manager > Floor Planner tab and select the place and route option after synthesis. This constraint file is available in the Design_Files_Directory\Source\fp folder.

10. Double-click Place and Route from the Design Flow tab.

    When place and route is successful, a green tick mark appears next to Place and Route.

2.4.7.3 Verify Timing


   When the design successfully 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 in the Reports tab.

2.4.7.4 Generate FPGA Array Data

Double-click Generate FPGA Array Data on the Design Flow tab.

When the FPGA array data is generated, a green tick mark appears next to Generate FPGA Array Data.

2.4.7.5 Configure Design Initialization Data and Memories

The Configure Design Initialization Data and Memories step in the Libero design flow is used to configure the LSRAM initialization data and storage location. User can use µPROM, sNVM, or SPI Flash as storage location based on the size of the initialization data and design requirements. In this tutorial, the SPI Flash memory is used to store the LSRAM initialization data.
This process requires the user application executable file (HEX file) as input to initialize the LSRAM blocks after device power-up. The hex file is provided with the design files. For more information about building the user application, see Building the User Application Using SoftConsole, page 35.

**Note:** To make the HEX file generated by SoftConsole compatible with the process of configuring design initialization data and memories in Libero, delete the extended linear record present in the first line of the HEX file. The HEX file available in the DesignFiles_Directory\Source folder is already modified to be compatible.

To generate an LSRAM initialization client and add it to an external SPI flash device:

1. Double-click **Configure Design Initialization Data and Memories** on the **Design Flow** tab.
2. On the **Fabric RAMs** tab, select **PROC_SUBSYSTEM/LSRAM_0** from the list of logical instances, and click **Edit**, as shown in the following figure. The PROC_SUBSYSTEM/LSRAM_0 instance is the Mi-V processor’s main memory. The System Controller initializes this instance with the imported client at power-up.

   *Figure 40 • Fabric RAMs Tab*

3. In the **Edit Fabric RAM Initialization Client** dialog box, set **Storage type** to **SPI-Flash** and click the **Import** button next to **Content from file**, as shown in the following figure.

   *Figure 41 • Edit Fabric RAM Initialization Client Dialog Box*

4. In the **Import Memory File** dialog box, locate the **MiV_uart_blinky.hex** file from DesignFiles_directory\Source folder. Select the “Use relative path from project directory” option.
5. In the **Edit Fabric RAM Initialization Client** window, click **OK**.
6. On the **Fabric RAMs** tab, click **Apply**, as shown in the following figure.

**Figure 43 • Fabric RAMs Tab - Apply Button**
7. In the **Design Initialization** tab, under Third stage (uPROM/sNVM/SPI-Flash), select the **SPI-Flash - No-binding Plaintext** option is selected and ensure that the SPI Clock divider value is set to 6, as shown in the following figure. This means that the imported user application will be written to SPI-Flash without encryption and authentication.

**Note:** The SPI Clock divider value specifies the required SPI SCK frequency to read the initialization data from SPI Flash. The SPI Clock divider value must be selected based on the external SPI Flash operating frequency range.

8. Click **Apply**.

**Figure 44 • Design Initialization Data**

[Design Initialization Data diagram]

This concludes the configuring of the storage type and application file for the fabric RAMs initialization.

### 2.4.7.6 Generate Design Initialization Data

1. Double-click **Generate Design Initialization Data** on the **Design Flow** tab.

   When the design initialization data is generated successfully, a green tick mark appears next to **Generate Design Initialization Data** in the Libero Design flow, and the following messages appear in the **Log** window:

   Info: 'Generate design initialization data' has completed successfully.
   Info: Stage 1 initialization client has been added to sNVM.
   Info: Stage 3 initialization client has been added to SPI.

2. Click the **SPI Flash** tab to verify that the bin file has been added, as shown in the following figure. The PolarFire Evaluation Board uses a Micron SPI flash. Therefore, ensure that Micron is selected in the **Manufacturer** list.
2.4.7.7 Generate Bitstream

To generate the programming bitstream:

- Double-click Generate Bitstream on the Design Flow tab.
  When the bitstream is generated, a green tick mark appears next to Generate Bitstream.

2.4.7.8 Run PROGRAM Action

After generating the bitstream, the PolarFire Evaluation Board must be set up so the device is ready to be programmed. Also, the serial terminal emulation program (PuTTY) must be set up to view the output of the user application.

2.4.7.8.1 Board Setup

To set up the board:

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

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J18, J19, J20, J21, J22</td>
<td>Short pins 2 and 3 for programming the PolarFire FPGA through FTDI.</td>
</tr>
<tr>
<td>J28</td>
<td>Short pins 1 and 2 for programming through the on-board FlashPro5.</td>
</tr>
<tr>
<td>J26</td>
<td>Short pins 1 and 2 for programming through the FTDI SPI.</td>
</tr>
<tr>
<td>J27</td>
<td>Short pins 1 and 2 for programming through the FTDI SPI.</td>
</tr>
<tr>
<td>J23</td>
<td>Open pins 1 and 2 for programming SPI flash.</td>
</tr>
<tr>
<td>J4</td>
<td>Short pins 1 and 2 for manual power switching using SW3</td>
</tr>
<tr>
<td>J12</td>
<td>Short pins 3 and 4 for 2.5 V</td>
</tr>
</tbody>
</table>

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

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the host PC to the J5 (USB) port on the PolarFire Evaluation Board using the USB cable.
4. Power on the board using the SW3 slide switch.
2.4.7.8.2 Serial Terminal Emulation Program (PuTTY) Setup

The user application (MiV_uart_blinky.hex file) prints the string **Hello World!** on the serial terminal through the UART interface.

Follow these steps to set up the serial terminal:

1. Start the PuTTY program.
2. Start Device Manager, note the second-highest COM port number, and use that in the PuTTY configuration. For example, in the list of ports shown in the following figure, COM93 is the port with the second highest number assigned to it.

![COM Port Number](image)

3. Select **Serial** as the **Connection type**, as shown in the following figure.

![Connection Type Selection](image)
4. Set the serial line to connect to the COM port number noted in Step 3.
5. Set the Speed (baud) to 115200 and Flow Control to None, as shown in the following figure.

**Figure 48 • PuTTY Configuration**

![](image)

6. Click Open.

PuTTY opens successfully, and the serial terminal emulation program is set up.

### 2.4.7.8.3 Running the PROGRAM Action

To run the PROGRAM action:

  
  When the device is programmed, a green tick mark appears next to Run PROGRAM action.

### 2.4.7.9 Generate SPI Flash Image

To generate the SPI flash image:

- Double-click Generate SPI Flash Image on the Design Flow tab.
  
  When the SPI file image is successfully generated, a green tick mark appears next to Generate SPI Flash Image.
2.4.7.10 Run PROGRAM_SPI_IMAGE Action

To program the SPI image:

2. In the dialog box that appears, click Yes.
   When the SPI image is successfully programmed on to the device, a green tick mark appears next to Run PROGRAM_SPI_IMAGE.

After SPI flash programming is completed, the device needs to be reset to execute the application. The following sequence of operations occurs after device reset or power-cycling the board:

1. The PolarFire System Controller initializes the LSRAM with the user application code from the external SPI flash and releases the system reset.
2. The Mi-V processor exits reset after DDR3 controller is ready and executes the user application from the LSRAM. As a result, LEDs 4, 5, 6, and 7 blink, and the string **Hello World!** is printed on the serial terminal, as shown in the following figure.

   **Figure 49 • Hello World String**

   ```
   Hello World!
   ```

3. When the board is power cycled, the device performs the same sequence of operations. As a result, LEDs 4, 5, 6, and 7 blink, and **Hello World!** is printed again on the serial terminal, as shown in the following figure.

   **Figure 50 • Hello World String After the Board is Power Cycled**

   ```
   Hello World!
   ```
3 Building the User Application Using SoftConsole

This section describes how to build a RISC-V user application executable (.hex) file and debug it using SoftConsole v6.1 or higher.

Building the user application involves the following steps:

- Creating a Mi-V SoftConsole Project, page 35
- Downloading the Firmware Drivers, page 37
- Importing the Firmware Drivers, page 39
- Creating the main.c File, page 41
- Mapping Firmware Drivers and the Linker Script, page 42
- Mapping Memory and Peripheral Addresses, page 48
- Setting the UART Baud Rate, page 50
- Building the Mi-V Project, page 50

3.1 Creating a Mi-V SoftConsole Project

To create a Mi-V SoftConsole project:

1. Create a SoftConsole workspace folder on the host PC for storing SoftConsole projects. For example, D:\Tutorial\MiV_Workspace.
2. Start SoftConsole.
3. In the Workspace Launcher dialog box, paste D:\Tutorial\MiV_Workspace as the workspace location, and click Launch, as shown in the following figure.

![Workspace Launcher](image)

When the workspace is successfully created, the SoftConsole main window opens.

4. Select File > New > Project, as shown in the following figure.

![New C Project Creation](image)

5. Expand C/C++ and select C Project in the New Project dialog box, and click Next.
6. In the C Project dialog box, do the following:
   • Enter a name for the project in the **Project name** field. For example, MiV_uart_blinky.
   • In the **Project type** pane, expand **Executable**, and select **Empty Project** and RSIC-V Cross GCC, as shown in the following figure. Then, click **Next**.

   **Figure 53 • C Project Dialog Box**

   ![C Project Dialog Box](image)

7. After selecting the platforms and configurations you want to deploy, click **Next**.

   **Figure 54 • Select Configurations Dialog Box**

   ![Select Configurations Dialog Box](image)

8. Click **Finish** in the GNU RISC-V Cross Toolchain wizard.
An empty Mi-V project (MiV_uart_blinky) is created, as shown in the following figure.

**Figure 55 • Empty Mi-V Project**

3.2 Downloading the Firmware Drivers

The empty Mi-V project requires the RISC-V Hardware Abstraction Layer (HAL) files and the following peripheral drivers:

- CoreGPIO
- CoreUARTapb
- CoreSPI Driver

Download the RISC-V HAL files and drivers using the Firmware Catalog application.
To download the drivers:

1. Create a folder named `firmware` in the Mi-V project workspace.
2. Open Firmware Catalog. The following figure shows the Firmware Catalog window.

![Firmware Catalog Window](image)

3. If new cores are available, click **Download them now!**
4. Right-click **RISC-V Hardware Abstraction Layer (HAL)**, and select **Generate**.
5. In the **Generate Options** window, enter `D:\Tutorial\MiV_Workspace\firmware` as the project folder, and click **OK**.
When the files are generated, the Reports window lists the files generated, as shown in the following figure.

![RISCV HAL Files Report](image)

6. Right-click **CoreUARTapb Driver**, and select **Generate**.
7. In the **Generate Options** window, enter `D:\Tutorial\MiV_Workspace\firmware` as the project folder, and click **OK**.
When the files are generated, the **Reports** window lists the files, as shown in the following figure.

![CoreUARTapb Files Report](image)

8. Right-click **CoreGPIO Driver**, and select **Generate**.
9. In the **Generate Options** dialog box, enter `D:\Tutorial\MiV_Workspace\firmware` as the project folder, and click **OK**.
When the files are generated, the **Reports** window lists the files, as shown in the following figure.

**Figure 59 • CoreGPIO Files Report**

10. Right-click **CoreSPI Driver**, and select **Generate**.
11. In the **Generate Options** window, enter `D:\tutorial\MiV_Workspace\firmware` as the project folder, and click **OK**.

When the files are generated, the Reports window lists the files, as shown in the following figure.

**Figure 60 • CoreSPI Driver Files Report**

The RISC-V HAL and firmware drivers are generated.

### 3.3 Importing the Firmware Drivers

After the driver files are downloaded, they must be imported into the empty project.

To import the drivers:

1. In SoftConsole, right-click the **MiV_uart_blinky** project, and select **Import**, as shown in the following figure.

**Figure 61 • Import Option**
2. In the **Import** dialog box, expand the **General** folder, and double-click **File System**, as shown in the following figure.

   **Figure 62 • Import Dialog Box**

3. On the next page of the **Import** dialog box, do the following (see **Figure 63**, page 40):
   - Click **Browse**, and locate the `D:\Tutorial\MiV_Workspace\firmware` folder.
   - Select the **firmware** folder, and click **OK**.
   - Expand the **firmware** folder, and select the **drivers**, **hal**, and **riscv_hal** folders.
   - Click **Finish**.

   **Figure 63 • Import Dialog Box - Page 2**

The **riscv_hal**, **hal**, and driver files are imported into the **MiV_uart_blinky** project.
3.4 Creating the main.c File

To update the main.c file:

2. In the New Source File dialog box, enter main.c in the Source file field, and click Finish, as shown in the following figure.

*Figure 64 • main.c File Creation*

![Source File dialog box](image)

The main.c file is created inside the project, as shown in the following figure.

*Figure 65 • The main.c file*

3. Copy all of the content of the DesignFiles_directory\Source\main.c file, and paste it in the main.c file of the SoftConsole project.
3.5 Mapping Firmware Drivers and the Linker Script

At this stage, the drivers and the Mi-V HAL files are not mapped. Therefore, the corresponding header files in the main.c file are unresolved, as shown in the following figure.

*Figure 66 • Unresolved Header Files*

To map the drivers and HAL files:

1. Right-click the MiV_uart_blinky project, and select Properties.
2. Expand C/C++ Build, and select Settings.
3. Set the configuration to All Configurations, as shown in the following figure. This setting applies the upcoming tool settings to both release and debug modes.

*Figure 67 • C/C++ Build Settings*
4. In the **Tool Settings** tab, expand **Target Processor**, and select the following settings:
   - Architecture: RV32I(-march=rv32i*)
   - Integer ABI: ILP32(-mabi=ilp32*)
   - Multiply extension: Enabled

*Figure 68 • Target Processor Tool Settings*

5. Expand **GNU RISC-V Cross C Compiler**, and select **Includes**.
6. Click Add to add the driver and Mi-V HAL directories, as shown in the following figure.

**Figure 69 • GNU RISC-V Cross C Compiler Tool Settings**

![GNU RISC-V Cross C Compiler Tool Settings](image)

**Note:** This application does not require including system paths and other files.

7. In the **Add directory path** dialog box, click **Workspace**, as shown in the following figure.

**Figure 70 • Add Directory Path Dialog Box**

![Add Directory Path Dialog Box](image)

8. In the **Folder Selection** dialog box, expand **MIV_uart_blinky project > drivers**, select the CoreGPIO folder, and click **OK**, as shown in the following figure.
9. In the **Add directory path** dialog box, click **OK**. The CoreGPIO folder path is added, as shown in the following figure.

**Figure 71 • CoreGPIO Folder Selection**

![CoreGPIO Folder Selection](image)

10. Repeat the preceding steps to add the CoreUARTapb, CoreSPI, hal, and riscv_hal folder paths. The drivers and riscv-HAL files are successfully mapped, as shown in the following figure.

**Figure 72 • Tool Settings Tab with CoreGPIO Path Added**

![Tool Settings Tab](image)
11. Select the GNU RISC-V Cross C Linker > General to map the linker script.
12. Click Add as shown in Figure 69, page 44, and in the Add file path dialog, click Workspace as shown in Figure 70, page 44.
13. In the File Selection dialog box, expand MIV_uart_blinky and select the linker script as shown in the following figure.

14. The linker script is mapped as shown in Figure 75, page 47.
15. Select the Do not use standard start files (-nostartfiles) option as shown in Figure 75, page 47.
16. Select the GNU RISC-V Cross Create Flash Image > General and set Other Flags to "--change-section-lma *-0x80000000" as shown in Figure 76, page 48. This excludes the extended linear record in the first line of the hex file.
17. Click **Apply** and when prompted to rebuild, choose **Yes**.
18. Then click **Apply and Close**.

The firmware drivers and linker script are successfully mapped. Notice that the header files are now resolved in the `main.c` file.

### 3.6 Mapping Memory and Peripheral Addresses

In the Libero design flow, the Mi-V processor execution memory address is mapped to 0x80000000, and its size is set to 64 KB. This information must be checked in the linker script before building the application.

To map the memory address:

1. Open the linker script (microsemi-riscv-ram.ld) available in the riscv_hal folder.
2. Ensure that the ram ORIGIN address is mapped to 0x80000000.
3. Ensure that the LENGTH of the ram is 64 KB.
4. Ensure that the RAM_START_ADDRESS is mapped to 0x80000000.
5. Ensure that the RAM_SIZE is 64 KB.
6. Ensure that the STACK_SIZE is 2 KB.
7. Ensure that the HEAP_SIZE is 2 KB.
8. Save the file.
The following figure shows the linker script.

**Figure 77 • Linker Script**

```markdown
1  * (c) Copyright 2016-2018 Microsemi SoC Products Group. All rights reserved.
2  *
3  * file name: microsemi-riscv-rom.ld
4  * MI-V soft processor linker script for creating a SoftConsole downloadable
5  * debug image executing in SRAM.
6  *
7  * This linker script assumes that the SRAM is connected at on the MI-V soft
8  * processor memory space. The start address and size of the memory space must
9  * be correct as per the Libero design.
10  *
11  * SVN $Revision: 9661 $
13  */
14
15 OUTPUT_ARCH( "riscv" )
16 ENTRY(start)
17
20 MEMORY
21 {  
22  rom (rwx) : ORIGIN = 0x80000000, LENGTH = 64k
23  
24  RAM_START_ADDRESS = 0x80000000;
25  /* Must be the same value MEMORY region rom ORIGIN above. */
26  ROM_SIZE = 64k;
27  /* Must be the same value MEMORY region rom LENGTH above. */
28  STACK_SIZE = 2k;
29  /* needs to be calculated for your application */
30  HEAP_SIZE = 2k;
31  /* needs to be calculated for your application */
```

In the Libero design flow, the UART, GPIO, and SPI peripherals addresses are mapped to 0x60000000, 0x60001000, and 0x60002000 respectively. This information needs to be provided in the `sample_hw_platform.h` file provided in the riscv_hal folder.

To map the peripheral address:

1. Open the hardware platform header file (sample_hw_platform.h) available in the riscv_hal folder.
2. Locate the COREUARTAPB0_BASE_ADDR macro, and define it as 0x60000000UL.
3. Locate the COREGPIO_OUT_BASE_ADDR macro, and define it as 0x60001000UL.
4. Locate the FLASH_CORE_SPI_BASE macro, and define it as 0x60002000UL.
5. Save the file.

The following figure shows the `sample_hw_platform.h` file after these updates.

**Figure 78 • Updated sample_hw_platform.h File**

```
30 */ Non-memory Peripheral base addresses
31 /* Format of define is: */
32 */ (corename)_BASE_ADDR 0xnnnnnnnnUL
33
34 define COREUARTAPB0_BASE_ADDR 0x60000000UL
35 define COREGPIO_IN_BASE_ADDR 0x60000200UL
36 define CORETHIRD_BASE_ADDR 0x60000300UL
37 define CORETHIRD_BASE_ADDR 0x60000400UL
38 define COREGPIO_OUT_BASE_ADDR 0x60000500UL
39 define FLASH_CORE_SPI_BASE 0x60002000UL
40 define CORE16590_BASE_ADDR 0x60007000UL
```

The memory and peripheral addresses are successfully mapped.
3.7 Setting the UART Baud Rate

The value of the `BAUD_VALUE_115200` macro in the `sample_hw_platform.h` file must be defined according to the system clock frequency to achieve the UART baud rate of 115200. The baud value is calculated using the following formula.

\[
\text{BAUD_VALUE} = \frac{\text{CLOCK}}{(16 \times \text{BAUD_RATE})} - 1
\]

To define the system clock frequency:

1. Look for `#define SYS_CLK_FREQ` statement in the `sample_hw_platform.h` file.
2. Define it as:
   ```
   #define SYS_CLK_FREQ 111111000UL
   ```

   The `SYS_CLK_FREQ` value must be same as that of the clock generated in the design.

   The following figure shows the system clock frequency definition.

   ![System Clock Frequency Definition](image)

3.8 Building the Mi-V Project

To build the Mi-V project, right-click the `MiV_uart_blinky` project in SoftConsole, and select `Build Project`.

The project is built successfully, and the hex file is generated in the `Debug` folder, as shown in the following figure.

![Hex File](image)

The HEX file can be used for Design and Memory Initialization. For more information, see Configure Design Initialization Data and Memories, page 27.
3.9 Debugging the User Application Using SoftConsole

Before debugging, the board and the serial terminal must be set up. For more information about the board and serial terminal setup, see Board Setup, page 31 and Serial Terminal Emulation Program (PuTTY) Setup, page 32.

To debug the application:

1. From the Project Explorer, select the MiV_uart_blinky project, and then click the Debug icon from the SoftConsole toolbar, as shown in the following figure.

   Figure 81 • Debug Icon

2. In the Create, manage and run configurations window, double-click GDB OpenOCD Debugging to generate the debug configuration for the MiV_uart_blinky project.

3. Select the generated MiV_uart_blinky Debug configuration, and click Search Project, as shown in the following figure.

   Figure 82 • Create, manage, and run configurations Window – Main Tab
4. Select the *MiV_uart_blinky.elf* binary, and click **OK**, as shown in the following figure.

*Figure 83 • MiV_uart_blinky.elf Selection*

5. Go to the **Debugger** tab, and replace the **Config Options**, **Executable**, and **Commands** as follows:
   - **Config Options:** `--file board/microsemi-riscv.cfg`
   - **Executable:** `${cross_prefix}gdb${cross_suffix}`
   - **Commands:** `set mem inaccessible-by-default off, set $target_riscv = 1, set remotetimeout 7, and set arch riscv:rv32`
Figure 84 • Create, manage, and run configurations Window – Debugger Tab
6. In **Debug Configurations -> Startup** tab, clear the **Pre-run/Restart reset** check box to halt the program at the main () function and clear the **Enable ARM semihosting** check box.

*Figure 85 • Debug Settings - Startup Tab*

![Debug Settings - Startup Tab](image)

7. Click **Apply**, and then click **Debug**, as shown in the preceding figure. The **Confirm Perspective Switch** dialog opens, as shown in *Figure 86*, page 54.

*Figure 86 • Confirm Perspective Switch Dialog Box*

![Confirm Perspective Switch Dialog Box](image)

8. Click **Yes**. The debugger halts the execution at the first instruction in the **main.c** file, as shown in the following figure.
9. On the SoftConsole toolbar, click Resume to resume the application execution, as shown in the following figure.

**Figure 88 • Resume Application Execution**

10. The string *Hello World!* is printed on the serial terminal, as shown in the following figure. Also, LEDs 4, 5, 6, 7 on the PolarFire Evaluation Board blink.

**Figure 89 • Hello World in Debug Mode**

11. On the SoftConsole menu, click Run > Suspend to suspend the execution of the application.

12. Click the Registers tab to view the values of the Mi-V internal registers, as shown in the following figure.
13. Click the **Variables** tab to view the values of variables in the source code, as shown in the following figure.

**Figure 91 • Variable Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spmic_pattern</td>
<td>uint32_t</td>
<td>10</td>
</tr>
<tr>
<td>delay_count</td>
<td>volatile int32_t</td>
<td>413318</td>
</tr>
</tbody>
</table>

14. From the SoftConsole toolbar, use the **Step Over** option to view the application execution line by line, or use the **Step Into** option to execute the instructions inside a function. Use the **Step Return** option to come out the function. You can also add breakpoints in the application source code.

15. On the SoftConsole toolbar, click **Terminate** to terminate the debugging of the application.

3.10 Debugging the User Application from DDR3 Memory

The SoftConsole debugger loads the application to the memory-mapped RAM based on the RAM start address specified in the `microsemi-riscv-ram.ld` linker file. The following figure shows the RAM Start Address parameters in the linker file.

Figure 92 • RAM Start Address Parameters

![Figure 92](image)

The SoftConsole reference project specifies the LSRAM start address, which is 0x80000000 (highlighted in Figure 92, page 57). To perform application debugging from DDR3 memory, modify this value to the DDR3 memory starting address, 0x80010000. After modifying the value, clean and build the project.

When the application is debugged from DDR3, the stack pointer and locations in the disassembly must point to DDR3 address, as shown in the following figure.

Figure 93 • Debugging from DDR3

![Figure 93](image)
4 Appendix

4.1 References

This section lists documents that provide more information about RISC-V and other IP cores used to build the RISC-V subsystem.

- For more information about Mi-V, see Mi-V RV32IMA_L1_AXI_HB.pdf from the Libero SoC Catalog.
- For more information about CoreJTAGDebug, see CoreJTAGDebug_HB.pdf.
- For more information about CoreAHBtoAPB3, see CoreAHBtoAPB3_HB.pdf.
- For more information about CoreAXITOAHBL, see CoreAXITOAHBL_HB.pdf.
- For more information about CoreGPIO, see CoreGPIO_HB.pdf.
- For more information about CoreAHBLite, see CoreAHBLite_HB.pdf.
- For more information about CoreAPB3, see CoreAPB3_HB.pdf.
- For more information about PolarFire Initialization Monitor, see UG0725: PolarFire FPGA Device Power-Up and Resets User Guide.
- For more information about PolarFire Clock Conditioning Circuitry (CCC), see UG0684: PolarFire FPGA Clocking Resources User Guide.
- For more information about PolarFire SRAM, see UG0680: PolarFire FPGA Fabric User Guide.
- For more information about Libero, ModelSim, and Synplify, see the Libero SoC PolarFire webpage.
- For more information about SoftConsole, see the SoftConsole webpage.
- For more information about loading a Job file using FlashPro Express, see the User Guide from FlashPro Express - > Help - > User Guide.