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 1.0 ...................................................................... 1

2 Building a Mi-V Processor Subsystem ........................................ 2
  2.1 Design Requirements ....................................................... 2
  2.2 Design Files .................................................................... 2
  2.3 Design Description .......................................................... 3
  2.4 Clocking Structure ............................................................ 4
  2.5 Reset Structure ................................................................. 5
  2.6 Hardware Implementation ............................................... 5
    2.6.1 IP Blocks .................................................................. 6
    2.6.2 Memory Map ............................................................. 7
  2.7 Software Implementation ............................................... 7
    2.7.1 Bootloader ................................................................ 7
    2.7.2 Target Application ................................................... 8

3 Setting Up the Hardware ......................................................... 9

4 Running the Demo ................................................................. 10
  4.1 Programming the RTG4 Device ......................................... 10
  4.2 Running the Mi-V Bootloader ............................................ 10

5 Appendix: Programming the Device Using FlashPro Express ........ 13

6 Appendix: Programming the Device Using Libero SoC ............... 16
Figures

Figure 1  Top Level Block Diagram ................................................................. 3
Figure 2  Clocking Structure ................................................................. 4
Figure 3  Reset Structure ................................................................. 5
Figure 4  SmartDesign Module ................................................................. 5
Figure 5  Bootloader Linker Script ................................................................. 7
Figure 6  Target Application Linker Script ......................................................... 8
Figure 7  Board Image ................................................................. 9
Figure 8  Mi-V Bootloader GUI ................................................................. 10
Figure 9  Connect COM Port ................................................................. 11
Figure 10  Import the Target Application File ...................................................... 11
Figure 11  SPI Flash Programmed ................................................................. 12
Figure 12  Execute Application From DDR ....................................................... 12
Figure 13  FlashPro Express Job Project ......................................................... 13
Figure 14  New Job Project from FlashPro Express Job ........................................ 14
Figure 15  Programming the Device ................................................................. 14
Figure 16  FlashPro Express—RUN PASSED ..................................................... 15
Figure 17  Libero Design Flow ................................................................. 16
Figure 18  Edit Data Storage Client ................................................................. 17
Figure 19  Import Memory File ................................................................. 17
Figure 20  Generate Bitstream ................................................................. 17
# Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Design Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>IP Blocks</td>
<td>6</td>
</tr>
<tr>
<td>Table 3</td>
<td>Memory Map</td>
<td>7</td>
</tr>
<tr>
<td>Table 4</td>
<td>Jumpers</td>
<td>9</td>
</tr>
</tbody>
</table>
Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

This is the first publication of the document.
Building a Mi-V Processor Subsystem

Microchip offers the Mi-V processor IP, a 32-bit RISC-V processor and software toolchain 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. RTG4® FPGAs support Mi-V soft processor to run user applications. This application note describes how to build a Mi-V processor subsystem to execute a user application from the designated fabric RAMs or DDR memory.

2.1 Design Requirements

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

<table>
<thead>
<tr>
<th>Design Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Requirements</strong></td>
<td></td>
</tr>
<tr>
<td>RTG4 Development Kit</td>
<td>Rev B</td>
</tr>
<tr>
<td>– RTG4 Development Board with RT4G150 -1CG 1657PROTO FPGA</td>
<td></td>
</tr>
<tr>
<td>– 12V, 5A AC power adapter</td>
<td></td>
</tr>
<tr>
<td>– USB A to mini-B cable</td>
<td></td>
</tr>
<tr>
<td>Host PC or Laptop</td>
<td>Any 64-bit Windows operating system</td>
</tr>
<tr>
<td><strong>Software Requirements</strong></td>
<td></td>
</tr>
<tr>
<td>-Libero® SoC</td>
<td>See the readme.txt file provided in the design files for all software versions needed to create the Mi-V processor subsystem.</td>
</tr>
<tr>
<td>-FlashPro Express</td>
<td></td>
</tr>
<tr>
<td>-SoftConsole</td>
<td></td>
</tr>
</tbody>
</table>

2.2 Design Files

1. Download the reference design files from the following link:
   http://soc.microsemi.com/download/rsc/?f=rtg4_ac490_df

2. Download and install Libero SoC from the following link:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads
2.3 Design Description

The size of RTG4 µPROM is 57 KB. User applications that do not exceed the µPROM size can be stored in µPROM and executed from internal Large SRAM memories (LSRAM). User applications that exceed the µPROM size must be stored in an external non-volatile memory. In this case, a bootloader executing from µPROM is required to initialize internal or external SRAM memories with the target application from the non-volatile memory.

The reference design demonstrates the bootloader capability to copy the target application (of size 7 KB) from SPI flash to DDR memory, and execute from the DDR memory. The bootloader is executed from internal memories. The code section is located in µPROM and the data section is located in internal Large SRAM (LSRAM).

Note: For more information about how to build the Mi-V bootloader Libero project and how to the build SoftConsole project, refer TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial.

Figure 1 shows the top-level block diagram of the design.
As shown in Figure 1, the following points describe the data flow of the design:

- The Mi-V processor executes the bootloader from the µPROM and designated LSRAMs. The bootloader interfaces with the GUI through the CoreUARTapb block and waits for the commands.
- When the SPI flash program command is received from the GUI, the bootloader programs the SPI flash with the target application received from the GUI.
- When the boot command is received from the GUI, the bootloader copies the application code from the SPI flash to DDR and then executes it from DDR.

### 2.4 Clocking Structure

There are two clock domains (55.5MHz and 27.7MHz) in the design. The on-board 50 MHz crystal oscillator is connected to the PF_CCC block which generates 55.5 MHz and 27.7 MHz clocks. The 55.5 MHz system clock drives the complete Mi-V processor subsystem except µPROM. The 27.7 MHz clock drives the RTG4 µPROM and the RTG4 µPROM APB interface. RTG4 µPROM supports a clock frequency of up to 30 MHz. DDR_FIC is configured for AHB bus interface, which operates at 55.5 MHz. The DDR memory operates at 333 MHz.

Figure 2 shows the clocking structure.

*Figure 2 • Clocking Structure*
2.5 Reset Structure

The POWER_ON_RESET_N and the LOCK signals are ANDed and the output signal (INIT_RESET_N) is used to reset the RTG4FDDRC_INIT block. After releasing the FDDR reset, the FDDR controller gets initialized and then the INIT_DONE signal is asserted. The INIT_DONE signal is used to reset the Mi-V processor, peripherals, and other blocks in the design.

![Figure 3 • Reset Structure](image)

2.6 Hardware Implementation

Figure 4 shows the Libero design of the Mi-V reference design.

![Figure 4 • SmartDesign Module](image)

**Note:** Libero SmartDesign screen shot shown in this application note is for illustration purpose only. Open the Libero project to see the latest updates and IP versions.
2.6.1 IP Blocks

Figure 2 lists the IP blocks used in the Mi-V processor subsystem reference design and their function.

<table>
<thead>
<tr>
<th>IP Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIV_RV32IMAF_L1_AHB</td>
<td>Mi-V soft processor.</td>
</tr>
<tr>
<td>CoreJTAGDEBUG</td>
<td>Facilitates the connection of Joint Test Action Group (JTAG) compatible soft core processors to the JTAG header for debugging. It provides fabric access to the JTAG interface using UJTAG macro.</td>
</tr>
<tr>
<td>CoreAHBLite</td>
<td>Multi-master AHB-Lite bus.</td>
</tr>
<tr>
<td>RTG4_SRAM_AHBL_AXI</td>
<td>RTG4 LSRAM. Used as system memory for Mi-V processor.</td>
</tr>
<tr>
<td>CoreAHBLtoAPB3</td>
<td>Bridge between AHB master and APB slave.</td>
</tr>
<tr>
<td>CoreUARTapb, CoreSPI, and CoreGPIO</td>
<td>UART, SPI, and GPIO controllers with APB interface.</td>
</tr>
<tr>
<td>RTG4FCCC</td>
<td>Macro to access RTG4 CCC block. It is used to synthesize 55.5 MHz and 27.7 MHz clock frequencies from the CCC with an on-board 50 MHz reference clock.</td>
</tr>
<tr>
<td>RTG4UPROM</td>
<td>Used for storing the bootloader program.</td>
</tr>
<tr>
<td>RTG4UPROMIF_APB</td>
<td>CoreUPROMIF_APB is an APB wrapper core that provides read-only access to the µPROM memory block within the RTG4 fabric via the APB interface. This facilitates easy access to the µPROM for APB masters.</td>
</tr>
<tr>
<td>RTG4FDDRC_INIT</td>
<td>The Fabric External Memory DDR (FDDR) Configurator is used to configure the external DDR memory parameters.</td>
</tr>
<tr>
<td>CoreAPB3</td>
<td>CoreAPB3 is an advanced microcontroller bus architecture (AMBA) 3 advanced peripheral bus (APB) fabric for interconnecting between an APB master and up to 16 APB slaves.</td>
</tr>
</tbody>
</table>

1. All the IP user guides and handbooks are available from Libero SoC -> Catalog.

RTG4 µPROM stores up to 10,400 36-bit words (374,400 bits of data). It supports only read operations during normal device operation after the device is programmed. The MIV_RV32IMAF_L1_AHB processor core comprises of an instruction fetch unit, an execution pipeline, and a data memory system. The MIV_RV32IMAF_L1_AHB processor memory system includes instruction cache and data cache. The MIV_RV32IMAF_L1_AHB core includes two external AHB interfaces-the AHB memory (MEM) bus master interface and the AHB Memory Mapped I/O (MMIO) bus master interface. The cache controller uses the AHB MEM interface to refill the instructions and the data caches. The AHB MMIO interface is used for an un-cached accesses to I/O peripherals.

The memory maps of the AHB MMIO interface and the MEM interface are 0x60000000 to 0X6FFFFFFF and 0x80000000 to 0x8FFFFFFF, respectively. The processor's reset vector address is configurable. The MI_UV32IMAF_L1_AHB's reset is an active-low signal, which must be de-asserted in sync with the system clock through a reset synchronizer.

The MIV_RV32IMAF_L1_AHB processor accesses the application execution memory using the AHB MEM interface. The CoreAHBLite_C0_0 bus instance is configured to provide 16 slave slots, each of
size 1 MB. The RTG4 LSRAM, RTG μPROM memory and RTG4FDDRC blocks are connected to this bus. μPROM is used for storing the bootloader application.

The MIV_RV32IMAF_L1_AHB processor directs the data transactions between addresses 0x60000000 and 0x6FFFFFFF to the MMIO interface. The MMIO interface is connected to the CoreAHBLite_C1_0 bus to communicate with peripherals connected to its slave slots. The CoreAHBLite_C1_0 bus instance is configured to provide 16 slave slots, each of size 256 MB. The UART, CoreSPI and CoreGPIO peripherals are connected to the CoreAHBLite_C1_0 bus via the CoreAHBTOAPB3 bridge and the CoreAPB3 bus.

### 2.6.2 Memory Map

Table 3 lists the memory map of the memories and peripherals.

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Start Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSRAM</td>
<td>0x80000000</td>
</tr>
<tr>
<td>μPROM</td>
<td>0x80100000</td>
</tr>
<tr>
<td>DDR3</td>
<td>0x80200000</td>
</tr>
<tr>
<td>UART</td>
<td>0x60000000</td>
</tr>
<tr>
<td>GPIO</td>
<td>0x60001000</td>
</tr>
<tr>
<td>SPI</td>
<td>0x60002000</td>
</tr>
</tbody>
</table>

### 2.7 Software Implementation

The reference design files include the SoftConsole workspace that contains the following software projects:

- **Bootloader**
- **Target Application**

#### 2.7.1 Bootloader

The bootloader application is programmed on the μPROM during device programming. The bootloader implements the following functions:

- Programming the SPI Flash with the target application.
- Copying the target application from SPI Flash to DDR3 memory.
- Switching the program execution to the target application available in DDR3 memory.

The bootloader application must be executed from μPROM with LSRAM as stack. Hence, the addresses of ROM and RAM in the linker script are set to the starting address of μPROM and designated LSRAMs, respectively. The code section is executed from ROM and data section is executed from RAM as shown in Figure 5.

#### Figure 5 • Bootloader Linker Script

The linker script (microsemi-riscv-ram_rom.ld) is available at the SoftConsole_Project\mivr32im-bootloader folder of the design files.
2.7.2 Target Application

The target application blinks the on board LEDs 1, 2, 3, and 4 and prints UART messages. The target application must be executed from DDR3 memory. Hence, the code and stack sections in the linker script are set to the starting address of DDR3 memory as shown in Figure 6.

Figure 6 • Target Application Linker Script

The linker script (microsemi-riscv-ram.ld) is available at the SoftConsole_Project\miv-rv32i-ddr-application folder of the design files.
3 Setting Up the Hardware

The following steps describe how to setup the hardware:

1. Ensure that the board is powered OFF using SW6 switch.
2. Connect the jumpers on the RTG4 development kit as shown in the following table:

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Pin From</th>
<th>Pin To</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>J11, J17, J19, J23, J26, J21, J32, and J27</td>
<td>1</td>
<td>2</td>
<td>Default</td>
</tr>
<tr>
<td>J16</td>
<td>2</td>
<td>3</td>
<td>Default</td>
</tr>
<tr>
<td>J33</td>
<td>1</td>
<td>2</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

3. Connect the host PC to the J47 connector using the USB cable.
4. Connect the power supply to J9 connector and switch ON the power supply switch, SW6.

Figure 7 • Board Image
Running the Demo

This chapter describes steps to program the RTG4 device with the reference design, programming the SPI Flash with the target application, and booting the target application from DDR memory using the Mi-V Bootloader GUI.

Running the demo involves the following steps:

1. Programming the RTG4 Device
2. Running the Mi-V Bootloader

4.1 Programming the RTG4 Device

The RTG4 device can be programmed either using FlashPro Express or Libero SOC.

- To program the device using FlashPro Express, see Appendix: Programming the Device Using FlashPro Express.
- To program the device using Libero SoC, see Appendix: Programming the Device Using Libero SoC.

4.2 Running the Mi-V Bootloader

On successful completion of programming, follow these steps:

1. Run the setup.exe file available at the following design files location.
   
   `<$Download_Directory>\rtg4_ac490_df\GUI_Installer\Mi-V Bootloader_Installer_V1.4`

2. Follow the installation wizard to install the Bootloader GUI application.

   Figure 8 shows the RTG4 Mi-V Bootloader GUI.

   ![Mi-V Bootloader GUI](image-url)
3. Select the highest COM port and click the connect button. After successful connection the Red indicator turns Green as shown in Figure 9.

**Figure 9** • Connect COM Port

4. Click the Import button and select the target application file (.bin). After importing, the path of the file is displayed on the GUI as shown in Figure 10.

<${Download_Directory}\rtg4_ac490_df\Source_files>

**Figure 10** • Import the Target Application File
5. As shown in Figure 10, click the **Program SPI Flash** option to program the target application on the SPI Flash. A pop-up is displayed after the SPI Flash is programmed as shown in Figure 11. Click **OK**.

*Figure 11 • SPI Flash Programmed*

6. Select the **Start Boot** option to copy the application from SPI Flash to DDR3 memory and start executing the application from DDR3 memory. After successful booting of the target application from DDR3 memory, the application prints UART messages and blinks on-board user LED1,2,3, and 4 as shown in Figure 12.

*Figure 12 • Execute Application From DDR*

7. The application is running from the DDR3 memory and this concludes the demo. Close the Mi-V Bootloader GUI.
Appendix: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the .job programming file using FlashPro Express. The .job file is available at the following design files folder location:

```
rtg4_ac490_df\Programming_Job
```

To program the device, complete the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 4, page 9.

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

2. Connect the power supply cable to the J9 connector.
3. Connect the USB cable from the host PC to the J47 (FTDI port).
4. Power ON the board using the SW6 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 13 • FlashPro Express Job Project**
7. Enter the following in the New Job Project from FlashPro Express Job dialog box:
   • **Programming job file**: Click **Browse** and navigate to the location where the .job file is located and select the file. The default location is:
     <download_folder>\rtg4_ac490_df\Programming_Job
   • **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

   ![Figure 14 • New Job Project from FlashPro Express Job](image)

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 15 • Programming the Device](image)
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. To run the demo, see Running the Demo, page 10.

*Figure 16 • FlashPro Express—RUN PASSED*

11. Close **FlashPro Express** (Project > Exit).
Appendix: Programming the Device Using Libero SoC

The reference design files include the Mi-V processor subsystem project created using Libero SoC. The RTG4 device can be programmed using Libero SoC. The Libero SoC project is completely built and run from Synthesis, Place and Route, Timing Verification, FPGA Array Data Generation, Update uPROM Memory Content, Bitstream Generation, FPGA Programming.

The Libero design flow is shown in the following figure.

![Libero Design Flow](image)

To program the RTG4 device, the Mi-V processor subsystem project must be opened in Libero SoC and the following steps must be re-run:

1. **Update uPROM Memory Content**: In this step, µPROM is programmed with the bootloader application.
2. **Bitstream Generation**: In this step, the STAPL file is generated for the RTG4 device.
3. **FPGA Programming**: In this step, the RTG4 device is programmed using the STAPL file.

Follow these steps:

1. From Libero Design Flow, select **Update uPROM Memory Content**.
2. Create a client using the Add option.
3. Select the client and then choose the Edit option.
4. Select Content from file and then select the Browse option as shown in Figure 18.
5. Navigate to the following design files location and select the miv-rv32im-bootloader.hex file as shown in Figure 19.
<$Download_Directory>\rtg4_ac490_df
  • Set the File Type as Intel-Hex (*.hex).
  • Select Use relative path from project directory.
  • Click OK.

6. Click OK.
The µPROM content is updated.

7. Double-click Generate Bitstream as shown in Figure 20.
8. Double-click **Run PROGRAM Action** to program the device as shown in Figure 20. The RTG4 device is programmed. See *Running the Demo*, page 10.