



RISC-V RX and LPDDR4 Memory Controller

Reference Design

FPGA-RD-02278-1.0

November 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

| | |
|---|----|
| Contents..... | 3 |
| Acronyms in This Document | 7 |
| 1. Introduction | 8 |
| 1.1. Quick Facts | 8 |
| 1.2. Features..... | 9 |
| 1.3. Naming Conventions | 9 |
| 1.3.1. Nomenclature..... | 9 |
| 1.3.2. Signal Names | 9 |
| 2. Directory Structure and Files | 10 |
| 3. Functional Description..... | 11 |
| 3.1. Design Block Diagram..... | 11 |
| 3.2. Clocking Scheme..... | 12 |
| 3.2.1. Clocking Scheme Overview - CertusPro-NX Devices..... | 12 |
| 3.2.2. Clocking Scheme Overview - Lattice Avant Devices | 13 |
| 3.3. Reset Scheme | 14 |
| 4. IP Configuration and Parameter Description | 15 |
| 4.1. RISC-V RX CPU | 15 |
| 4.2. LPDDR4 Memory Controller | 17 |
| 4.2.1. LPDDR4 Memory Controller for CertusPro-NX Devices..... | 17 |
| 4.2.2. LPDDR4 Memory Controller for Lattice Avant Devices | 20 |
| 4.3. AXI4 Interconnect IP..... | 22 |
| 4.4. System Memory | 25 |
| 5. Signal Description | 27 |
| 6. RISC-V RX Software Flow | 28 |
| 6.1. RISC-V RX Boot up Sequence..... | 28 |
| 6.2. Bootloader Software Flow Chart | 29 |
| 6.3. Application Image Format | 30 |
| 6.4. Application Image Generation | 31 |
| 7. Implementing the Reference Design on the Board | 32 |
| 7.1. Extracting the Reference Design Files | 32 |
| 7.2. Setting up the CertusPro-NX Versa Board..... | 32 |
| 7.3. Setting up the Lattice Avant-AT-E Evaluation Board | 33 |
| 7.4. Setting up the UART Terminal | 34 |
| 7.5. Programming the Application Image to SPI Flash | 35 |
| 7.6. Programming the FPGA Bitstream | 37 |
| 7.7. Verifying the Results | 37 |
| 8. Compiling the Reference Design..... | 39 |
| 8.1. Building and Generating the RISC-V Processor using the Lattice Propel Builder | 39 |
| 8.2. Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software | 41 |
| 8.2.1. Floor-planning Constraints | 42 |
| 8.3. Building the Software Project using Propel SDK..... | 42 |
| 8.3.1. Setting Up the Project Workspace | 42 |
| 8.3.2. Building Bootloader and Application Software | 44 |
| 8.4. Pre-initializing System Memory | 46 |
| 8.5. Generating the Flash Image | 48 |
| 9. Customizing the Reference Design | 49 |
| 9.1. Changing LPDDR4 Memory Controller Parameters..... | 49 |
| 9.2. Adding New Component to the Propel Builder System | 50 |
| 9.2.1. Hardware Flow | 50 |
| 9.2.2. Software Flow..... | 51 |
| 9.3. Changing the Application Image Load Address | 52 |
| 9.4. Updating Linker Script to LPDDR4 Memory Address | 52 |

| | |
|---|----|
| 10. Debugging the Design | 53 |
| 10.1. SPI Flash Programming Fail | 53 |
| 10.1.1. Check Flash Device for CertusPro-NX Devices..... | 53 |
| 10.1.2. Check Correct Flash Device for Lattice Avant Devices..... | 55 |
| 10.1.3. Check TCK Divider Setting..... | 55 |
| 10.1.4. Check Cable Settings | 55 |
| 10.2. UART Serial Terminal Prints Incorrect Characters | 56 |
| 11. Resource Utilization | 57 |
| References | 58 |
| Technical Support Assistance | 59 |
| Revision History | 60 |

Figures

| | |
|--|----|
| Figure 2.1. Directory Structure | 10 |
| Figure 3.1. Reference Design Top Level Block Diagram | 11 |
| Figure 3.2. Clocking Block Diagram – Lattice CertusPro-NX Device | 12 |
| Figure 3.3. Clocking Block Diagram - Lattice Avant Device | 13 |
| Figure 3.4. Reset Scheme Block Diagram | 14 |
| Figure 4.1. RISC-V RX CPU IP Configuration | 15 |
| Figure 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices | 18 |
| Figure 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices | 20 |
| Figure 4.4. AXI4 Interconnect IP Configuration | 22 |
| Figure 4.5. System Memory IP Configuration | 25 |
| Figure 6.1. Boot up Sequence | 28 |
| Figure 6.2. Bootloader Software Flow | 29 |
| Figure 6.3. Application Image Format | 30 |
| Figure 6.4. Application Image Generation | 31 |
| Figure 7.1. CertusPro-NX Versa Board | 32 |
| Figure 7.2. Lattice Avant-AT-E Evaluation Board | 33 |
| Figure 7.3. Tera Term New Connection | 34 |
| Figure 7.4. Tera Term Serial Port Setup and Connection | 35 |
| Figure 7.5. Windows Start Manu > Radiant Programmer | 35 |
| Figure 7.6. Radiant Programmer - Getting Started Dialog Box | 35 |
| Figure 7.7. Output Console | 36 |
| Figure 7.8. Output Console | 37 |
| Figure 7.9. RISC-V RX and LPDDR4 Reference Design Results | 38 |
| Figure 8.1. Lattice Propel Builder Schematic View | 39 |
| Figure 8.2. Windows Start Manu > Lattice Propel Builder | 40 |
| Figure 8.3. TCL Console – No Error for Validate | 40 |
| Figure 8.4. TCL Console – No Error for Generate | 40 |
| Figure 8.5. Windows Start Menu > Radiant Software | 41 |
| Figure 8.6. Export Files | 41 |
| Figure 8.7. Windows Start Manu > Lattice Propel 2023.1 | 42 |
| Figure 8.8. Propel SDK Workspace Setup | 43 |
| Figure 8.9. Select Existing Projects | 44 |
| Figure 8.10. Import Projects | 44 |
| Figure 8.11. Project Explorer | 45 |
| Figure 8.12. Console | 45 |
| Figure 8.13. system0_inst Block | 46 |
| Figure 8.14. System Memory IP Setting to Pre-initialize with Bootloader Software | 47 |
| Figure 8.15. Python Verification | 48 |
| Figure 9.1. lpddr4_inst Block | 49 |
| Figure 9.2. New System Error | 51 |
| Figure 10.1. Flash Device Properties for Macronix | 53 |
| Figure 10.2. Flash Device Properties for Winbond | 54 |
| Figure 10.3. Lattice Avant Device Flash Programmer Project Settings | 55 |
| Figure 10.4. TCK Divider Error | 55 |
| Figure 10.5. Programming Speed Settings | 55 |
| Figure 10.6. UART Settings | 56 |

Tables

| | |
|--|----|
| Table 1.1. Reference Design Summary | 8 |
| Table 2.1. Directory List | 10 |
| Table 3.1. Clock Description – Lattice CertusPro-NX Target Device | 12 |
| Table 3.2. Clock Signal Description – Lattice Avant Target Device | 13 |
| Table 3.3. Reset Signal Description..... | 14 |
| Table 4.1. RISC-V RX IP Configuration | 16 |
| Table 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices | 19 |
| Table 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices | 21 |
| Table 4.4. AXI4 Interconnect IP Configuration..... | 23 |
| Table 4.5. System Memory IP Configuration | 26 |
| Table 5.1. Primary I/O | 27 |
| Table 6.1. Application Image Header..... | 30 |
| Table 7.1. Programming Files Selection | 36 |
| Table 9.1. LPDDR4 Memory Controller IP Parameters Customization | 49 |
| Table 11.1. Resource Utilization for CertusPro-NX Devices..... | 57 |
| Table 11.2. Resource Utilization for Lattice Avant Devices | 57 |

Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---------|-------------------------------------|
| AHB | Advanced High-Performance Bus |
| APB | Advanced Peripheral Bus |
| API | Application Programming Interface |
| AXI4 | Advanced eXtensible Interface 4 |
| BSP | Board Support Package |
| CLINT | Core Local Interrupt controller |
| CPU | Central Processing Unit |
| PLIC | Platform Level Interrupt Controller |
| RTOS | Real Time Operating System |
| SoC | System on Chip |

1. Introduction

The RISC-V RX CPU and LPDDR4 Memory Controller Reference Design provides an example usage of the RISC-V RX soft IP and LPDDR4 memory controller in Lattice Avant™ and CertusPro™-NX FPGA devices.

The CPU is connected to the LPDDR4 memory controller IP for interfacing to external LPDDR4 SDRAM. This design demonstrates how a software program is stored on an external SDRAM and the CPU is able to execute the software from the memory. It addresses applications that require a large memory but the FPGA on-chip memory is not sufficient. This design contains an example bootloader program to load software from the SPI Flash device to the LPDDR4 SDRAM. The CPU connects to all peripherals in the system through the AXI Interconnect and bridges.

The Lattice Semiconductor RISC-V RX CPU contains a 32-bit RISC-V processor core and several submodules – Platform Level Interrupt Controller (PLIC), Core Local Interrupter (CLINT), and Watchdog. The CPU core supports the RV32IMC instruction set and debug feature, which is JTAG – IEEE 1149.1 compliant. The modules outside are accessed by the processor core using AXI or Local Bus Interface.

The Lattice Semiconductor LPDDR4 Memory Controller IP Core provides a solution to interface with LPDDR4 SDRAM. Lattice provides a turnkey solution consisting of a controller, DDR PHY, and associated clocking and training logic. The LPDDR4 Memory Controller IP Core reduces the effort required to integrate a memory controller with user application designs. It minimizes the need to directly handle the LPDDR4 SDRAM signals by providing AXI4 interface support.

1.1. Quick Facts

Download the reference design files from the [RISC-V RX and LPDDR4 Memory Controller](#) web page.

Table 1.1. Reference Design Summary

| | | |
|------------------------------|---------------------------|---|
| General | Target device | CertusPro-NX devices Lattice Avant-AT-E devices |
| | Source code format | Verilog C |
| Simulation | Functional simulation | Not performed |
| | Timing simulation | Not performed |
| | Test bench | Not available |
| Software Requirements | Software tool and version | Lattice Radiant™ software version 2023.1 Lattice Propel™ Builder version 2023.1 Python 3 |
| | IP version | RISC-V RX 2.2.0 LPDDR4 Memory Controller for Nexus Devices 2.1.0 Memory Controller for Avant 1.2.0 SPI Flash Controller System Memory 2.0.0 AXI4 Interconnect 1.2.0 AXI4 to AHB-Lite Bridge 1.1.0 AXI4 to APB Bridge 1.1.0 APB Interconnect 1.2.0 GPIO 1.6.1 |
| Hardware Requirements | Board | CertusPro-NX Versa Board Lattice Avant-AT-E Evaluation Board |
| | Cable | USB-A to Mini-B cable Lattice HW-USBN-2B cable (for Lattice Avant device only) |

1.2. Features

The key features of the reference design include:

- RISC-V RX soft IP features described in [RISC-V RX CPU IP - Lattice Propel Builder 2023.1 \(FPGA-IPUG-02230\)](#)
- LPDDR4 memory controller interface IP core for CertusPro-NX device features described in [LPDDR4 Memory Controller for Nexus Devices \(FPGA-IPUG-02127\)](#)
- LPDDR4 memory controller interface IP core for Lattice Avant device features described in [Avant Memory Controller IP Core - Lattice Radiant Software \(FPGA-IPUG-02208\)](#)
- AXI4 Interconnect features described in [AXI4 Interconnect Module - Lattice Propel Builder \(FPGA-IPUG-02196\)](#)
- SPI Flash memory controller features described in [SPI Flash Memory Controller IP Core - Lattice Radiant Software \(FPGA-IPUG-02134\)](#)
- Bootloader software example to copy (with CRC check) application software from flash to SDRAM
- FreeRTOS demo as application software that runs on SDRAM

1.3. Naming Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

- `_n` are active low (asserted when value is logic 0)
- `_i` are input signals
- `_o` are output signals
- `_io` are bi-directional signals

2. Directory Structure and Files

Figure 2.1 shows the directory structure.

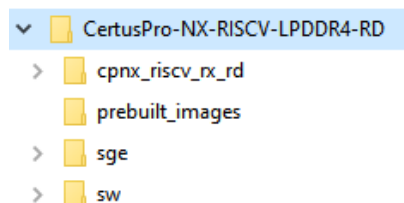


Figure 2.1. Directory Structure

Table 2.1 shows the list of directories included in the reference design package.

Note: In the reference design targeted to the Lattice Avant device, *Avant* replaces *cpnx* in folder/file names.

Table 2.1. Directory List

| Directory/File | Description |
|------------------|---|
| cpnx_riscv_rx_rd | Contains the Propel Builder project and generated HDL files for IP. |
| prebuilt_images | Contains the bitstream file, software image file, and Lattice Radiant Programmer scripts. |
| sge | Contains the Propel Builder project software handoff files. |
| sw | Contains the software project and source code used in this reference design. |

3. Functional Description

3.1. Design Block Diagram

The reference design demonstrates a complete and functional RISC-V processor system. This processor system is created using Propel Builder. All components in the system are available in Propel Builder. The components are instantiated and connected in Propel Builder, which also generates the corresponding RTL design files.

Figure 3.1 shows the top-level block diagram and connections of the reference design.

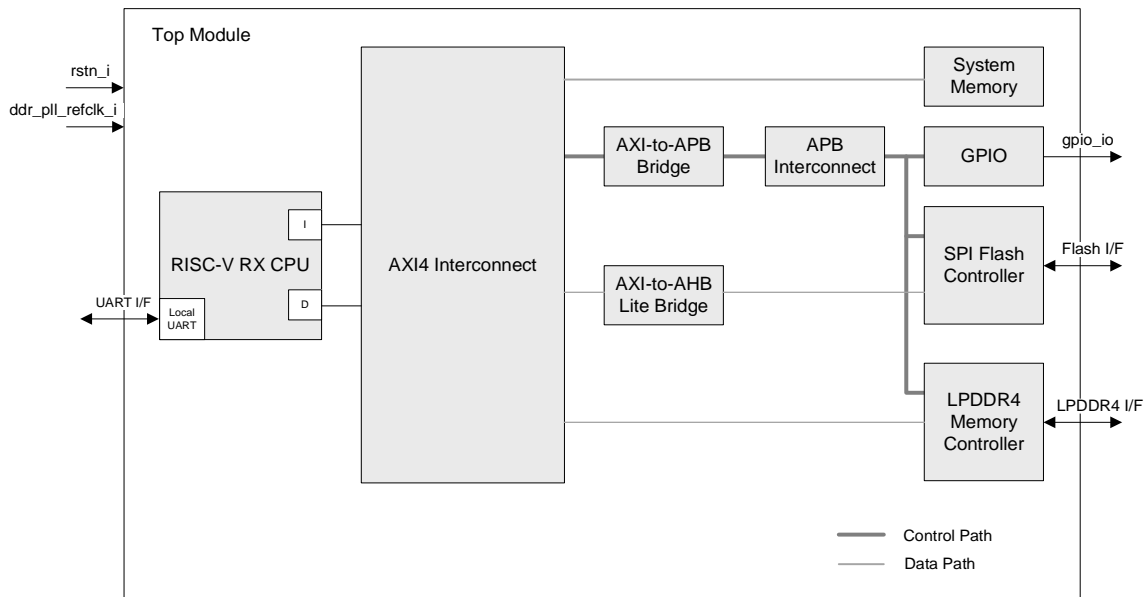


Figure 3.1. Reference Design Top Level Block Diagram

The design includes the following components:

- RISC-V RX processor with local UART
- LPDDR4 memory controller
- System Memory
- SPI flash controller
- General Purpose I/O (GPIO)
- AXI4 interconnect
- APB interconnect
- AXI to APB bridge
- AXI to AHB-Lite bridge

Each component in the block diagram is instantiated using the IP in Propel Builder. The IP features and parameters are described in the [IP Configuration and Parameter Description](#) section.

Figure 3.1 shows the multiple interfaces at the design top level. The signals in each interface are described in the [Signal Description](#) section.

3.2. Clocking Scheme

This section describes the reference design clocking scheme. There are some minor differences between the clocking scheme for Certus Pro-NX and Lattice Avant devices. Refer to the [Clocking Scheme Overview - CertusPro-NX Devices](#) section or to the [Clocking Scheme Overview - Lattice Avant Devices](#) section.

3.2.1. Clocking Scheme Overview - CertusPro-NX Devices

Figure 3.2 shows the clocking scheme of the reference design when targeted to the CertusPro-NX device. Table 3.1 provides the clock signal details.

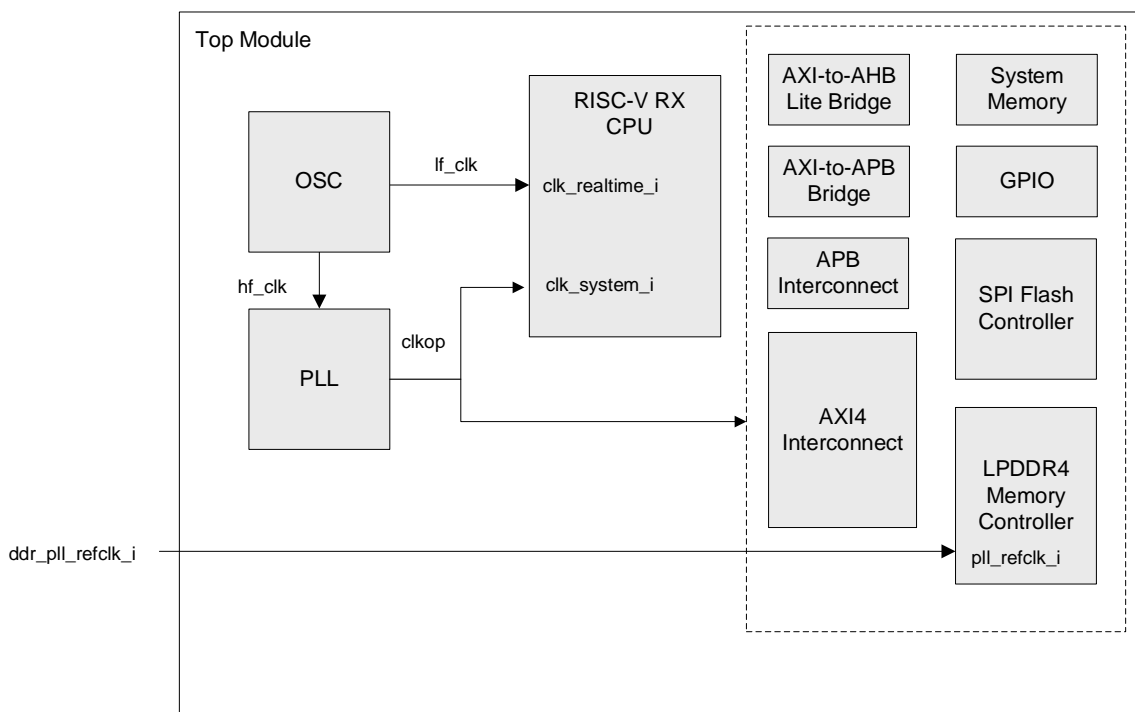


Figure 3.2. Clocking Block Diagram – Lattice CertusPro-NX Device

Table 3.1. Clock Description – Lattice CertusPro-NX Target Device

| Clock Signal | Source | Destination | Clock Frequency | Description |
|------------------|-------------------------------|---|---------------------|--|
| lf_clk | FPGA Oscillator | RISC-V RX CPU | 32 kHz | Low speed real time clock to RISC-V RX |
| hf_clk | FPGA Oscillator | PLL | 50 MHz | Reference clock input to the PLL |
| clkop | PLL | RISC-V RX CPU, all IPs in the system | 50 MHz ¹ | PLL primary clock output to clocks the RISC-V RX high speed input and all components in the system |
| ddr_pll_refclk_i | Oscillator on development kit | LPDDR4 memory controller | 100 MHz | Reference clock input to the LPDDR4 memory controller's PLL |

Note:

- The PLL clkop output frequency is adjustable to a value other than 50 MHz. This increases the RISC-V RX and system clock frequency. However, the current reference design is closed timing at 50 MHz. Hence, the PLL clkop is configured to 50 MHz.

3.2.2. Clocking Scheme Overview - Lattice Avant Devices

Figure 3.2 shows the clocking scheme of the reference design when targeted to the Lattice Avant device. Table 3.1 provides the clock signal details.

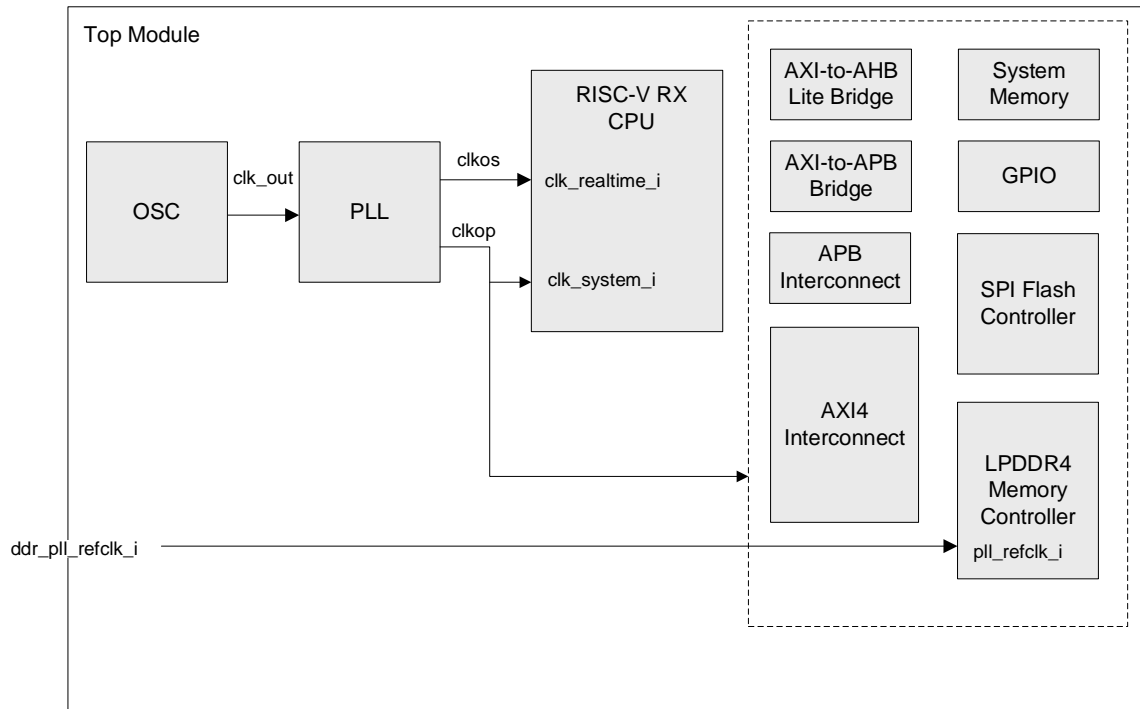


Figure 3.3. Clocking Block Diagram - Lattice Avant Device

Table 3.2. Clock Signal Description – Lattice Avant Target Device

| Clock Signal | Source | Destination | Clock Frequency | Description |
|------------------|-------------------------------|---|---------------------|--|
| clk_out | FPGA Oscillator | PLL | 50 MHz | Reference clock input to the PLL |
| clkop | PLL | RISC-V RX CPU, all IPs in the system | 50 MHz ² | PLL primary clock output for RISC-V RX high speed input and all components in the system |
| clkos | PLL | RISC-V RX CPU | 20 MHz ¹ | PLL secondary clock output for RISC-V RX low speed real time clock input |
| ddr_pll_refclk_i | Oscillator on development kit | LPDDR4 memory controller | 100 MHz | Reference clock input to the LPDDR4 memory controller PLL |

Note:

1. Clock divider is used to divide the PLL clkos from 20 MHz to 2.5 MHz for clocking the clk_realtime_i of RISC-V RX. This is because the PLL output frequency is not as low.
2. The PLL clkop output frequency is adjustable to a value other than 50 MHz. This increases the RISC-V RX and system clock frequency. However, the current reference design is closed timing at 50 MHz. Hence, the PLL clkop is configured to 50 MHz.

3.3. Reset Scheme

This section describes the reference design reset scheme for both CertusPro-NX and Lattice Avant target devices. Figure 3.4 shows the reset block diagram. Table 3.3 provides the reset signal details.

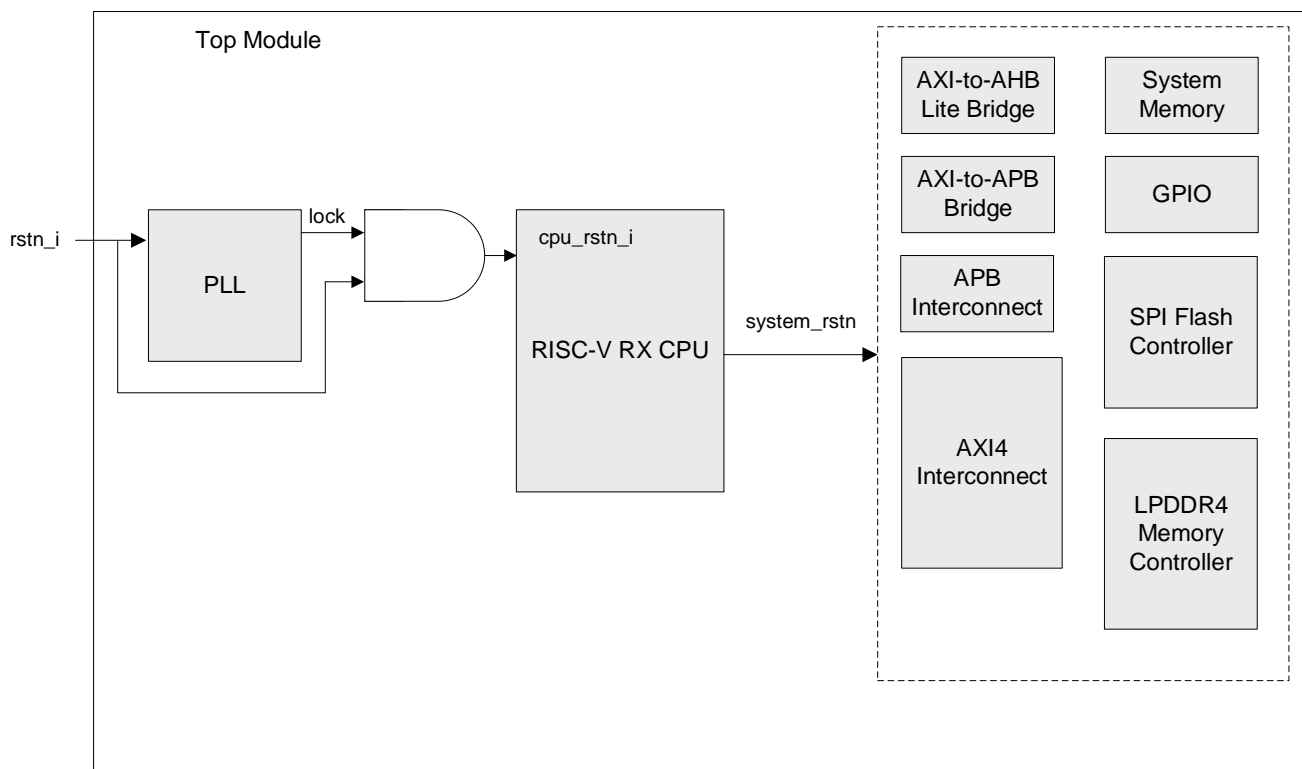


Figure 3.4. Reset Scheme Block Diagram

Table 3.3. Reset Signal Description

| Reset Signal | Source | Destination | Description |
|--------------|------------------------|--------------------------|--|
| rstn_i | Push button on dev kit | PLL reset input | Reset the PLL |
| cpu_rstn_i | PLL lock and rstn_i | RISC-V RX reset input | Release RISC-V RX from reset once rstn_i is not asserted AND PLL is locked |
| system_rstn | RISC-V RX | All components in system | RISC-V RX output reset is used to reset all components in the system. This triggers reset during CPU debugging mode. |

4. IP Configuration and Parameter Description

The reference design is created using Lattice Propel Builder. The top-level HDL file is generated by Propel Builder and is used as the top module for the design. The design parameterization is performed by configuring the IP in Propel Builder. This section describes the following IPs and their parameters.

- RISC-V RX CPU
- LPDDR4 Memory Controller
- AXI4 Interconnect
- System Memory

4.1. RISC-V RX CPU

The RISC-V RX CPU IP has AXI-based instruction and data ports. The instruction port is connected to the memory that contains the software code for CPU execution. The data port is connected to the memory and peripherals for control. The local UART feature is enabled to allow communication through serial protocol.

Figure 4.1 shows the RISC-V RX CPU IP configuration.

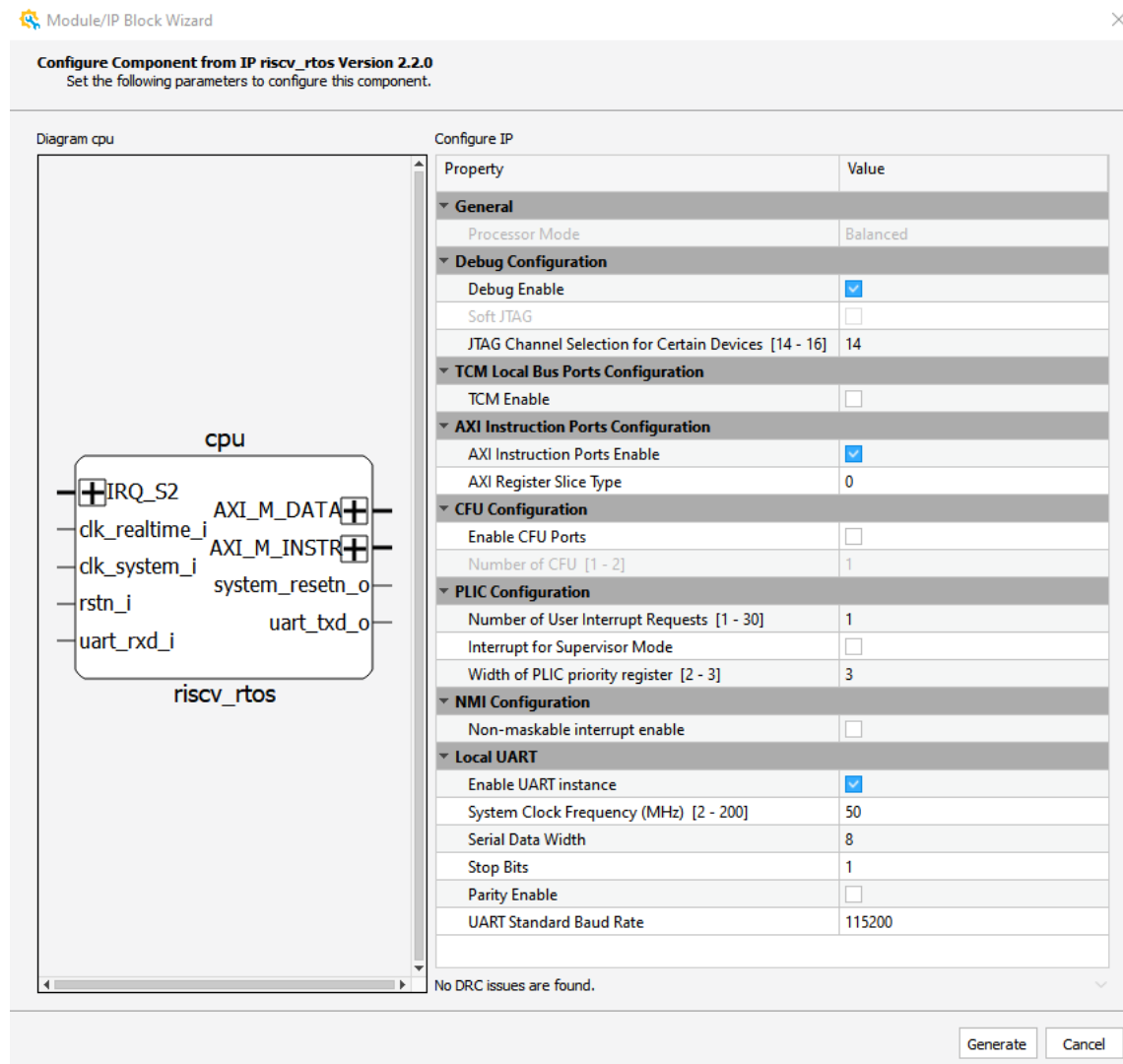


Figure 4.1. RISC-V RX CPU IP Configuration

Table 4.1 provides the configuration details of the RISC-V RX IP.

Table 4.1. RISC-V RX IP Configuration

| Property | Value | Description |
|-----------------------------|--|--|
| Debug Enable | True | Enable the JTAG interface for CPU debug capability. |
| Soft JTAG | False (CertusPro-NX devices) True (Lattice Avant devices) | Allow debugger JTAG interface routed to FPGA soft logic. Enable only when Lattice Avant device is the target device. |
| JTAG Channel | 14 | Channel number used by the debugger. |
| TCM Enable | False | Tightly coupled memory interface is not used in the design. LPDDR4 memory is used. ¹ |
| AXI Instruction Port Enable | True | The AXI instruction port is used to connect to the LPDDR4 memory controller, which has AXI-based interface. |
| AXI Register Slice Type | 0 | Bypass register slice on the AXI instruction port. |
| Enable CFU Ports | False | CFU (custom function unit) is not used in the design. |
| Enable UART instance | True | Enable local UART within the RISC-V RX CPU. There is no need to add extra UART IP in the system. |
| System Clock Frequency | 50 MHz | Specify the clock frequency for RISC-V RX clk_system_i to match with the actual clock frequency. |
| Serial Data Width | 8 | 8-bit of serial data |
| Stop Bits | 1 | 1 bit of stop |
| Parity Enable | False | Parity is not enabled |
| UART Standard Baud Rate | 115200 | Baud rate of 115200. Need matching with terminal setting. |

Note:

1. Using tightly coupled memory with external memory is not supported.

4.2. LPDDR4 Memory Controller

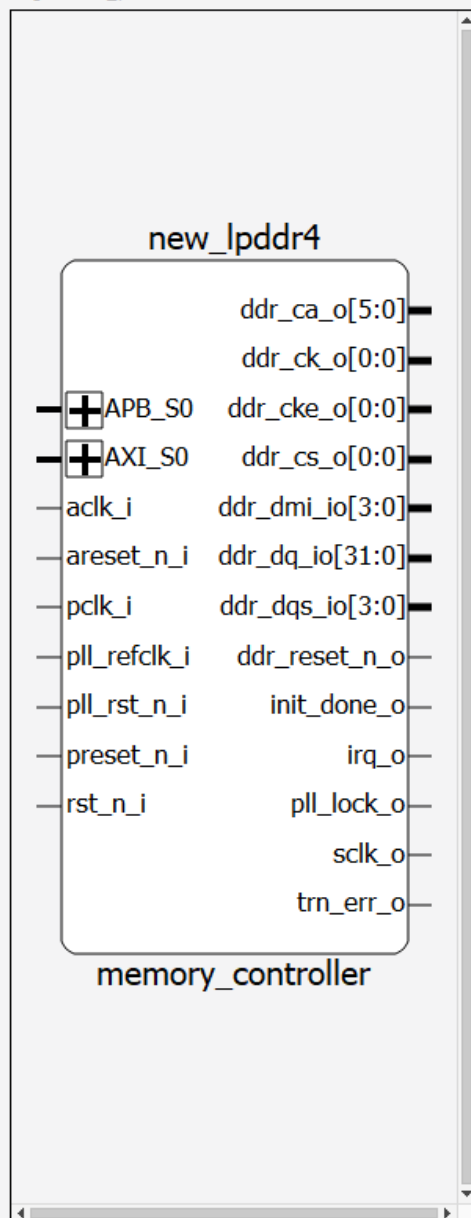
The LPDDR4 Memory Controller IP enables access to external LPDDR4 memory modules. The memory can be used to store CPU software code and data. The memory controller IP are different on CertusPro-NX and Lattice Avant devices. The following subsections describe the IP configuration for each device family.

4.2.1. LPDDR4 Memory Controller for CertusPro-NX Devices

Figure 4.2 shows the LPDDR4 Memory Controller IP configuration for the CertusPro-NX device.

Configure Component from IP memory_controller Version 2.1.0
Set the following parameters to configure this component.

Diagram new_lpddr4



Configure IP

| General | | Memory Device Timing | Training Settings |
|---|--|-------------------------------------|-------------------|
| Property | | Value | |
| ▼ General | | | |
| DDR Interface Type | | LPDDR4 | |
| I/O Buffer Type | | LVSTL_I | |
| DDR Command Frequency (MHz) | | 533 | |
| Gearing Ratio | | 8:1 | |
| Enable ECC | | <input type="checkbox"/> | |
| Enable Power Down | | <input type="checkbox"/> | |
| Enable DBI | | <input type="checkbox"/> | |
| Read Latency | | 10 | |
| Write Latency | | 6 | |
| Enable Internal RISC-V CPU | | <input checked="" type="checkbox"/> | |
| ▼ Clock Settings | | | |
| Enable PLL | | <input checked="" type="checkbox"/> | |
| PLL Reference Clock from Pin | | <input checked="" type="checkbox"/> | |
| I/O Standard for Reference Clock | | LVSTLD_I | |
| RefClock (MHz) | | 100 | |
| DDR Command Actual Frequency (MHz) [6.25 - 800] | | 533.333 | |
| ▼ Memory Configuration | | | |
| DDR Density (per Channel) | | 8Gb | |
| DDR Bus Width | | 32 | |
| Number of Ranks | | 1 | |
| Number of DDR Clocks | | 1 | |
| Number of Chip Selects | | 1 | |
| ▼ Local Data Bus | | | |
| Local Data Bus Type | | AXI4 | |
| Address Width | | 31 | |
| Data Width | | 256 | |
| ID Width | | 2 | |
| Write Ordering Queues | | 4 | |
| Read Ordering Queues | | 4 | |
| Enable Local Bus Clock | | <input checked="" type="checkbox"/> | |
| Enable APB I/F | | <input checked="" type="checkbox"/> | |

Calculate

No DRC issues are found.

Generate

Cancel

Figure 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices

Table 4.2 provides the configuration details of the LPDDR4 Memory Controller IP for the CertusPro-NX device.

Table 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices

| Property | Value | Description |
|------------------------------|---------|---|
| I/O Buffer Type | LVSTL_I | Set the I/O buffer type standard to LVSTL_I for better performance. |
| DDR Command Frequency | 533 MHz | Maximum supported frequency on CertusPro-NX devices. |
| Enable Power Down | False | Not used in this reference design |
| Enable DBI | False | Not used in this reference design |
| PLL Reference Clock from Pin | True | Export the PLL reference clock input to be connected from the top module. Refer to the Clocking Scheme Overview - CertusPro-NX Devices section for details. |
| DDR Density (per Channel) | 8 Gb | Configure the DDR DRAM density based on the memory module on the development kit. |
| DDR Bus Width | 32 | Configure the DDR DRAM bus width based on the memory module on development kit. |
| Data Width | 256 | Data width on the AXI interface of the memory controller. |
| ID Width | 2 | ID width on the AXI interface of the memory controller. This value must match with the AXI Manager in the Interconnect IP. Refer to the AXI4 Interconnect IP section for details. |
| Write/Read Ordering Queues | 4 | Set the ordering queues that process the write and read data requests. More queues means better ordering of access to the DDR bus. This minimizes the gaps between accesses but consumes more resources. The default value is used to balance performance and resource utilization. |
| Enable Local Bus Clock | True | Enable the clock domain crossing logic for the local data bus. |
| Enable APB I/F | True | Enable APB interface, which allows the CPU to access the memory controller registers. |

Note:

1. Default settings are used for properties in the *Memory Device Timing* and the *Training Settings* tabs.

4.2.2. LPDDR4 Memory Controller for Lattice Avant Devices

Figure 4.3 shows the LPDDR4 Memory Controller IP configuration for Lattice Avant devices.

Module/IP Block Wizard

Configure Component from IP mc_avant Version 1.2.0
Set the following parameters to configure this component.

Diagram lpddr4

Configure IP

| General | | Memory Device Timing |
|-----------------------------------|-------------------------------------|----------------------|
| Property | Value | |
| General | | |
| Interface Type | LPDDR4 | |
| I/O Buffer Type | LVSTL11_II | |
| DDR Command Frequency (MHz) | 800 | |
| Gearing Ratio | 8:1 | |
| Enable Power Down | <input type="checkbox"/> | |
| Enable DBI | <input type="checkbox"/> | |
| Read Latency | 14 | |
| Write Latency | 8 | |
| Enable Internal RISC-V CPU | <input checked="" type="checkbox"/> | |
| Clock Settings | | |
| Enable PLL | <input checked="" type="checkbox"/> | |
| Reference Clock: Frequency (MHz) | 100.0 | |
| DDR Memory Actual Frequency (MHz) | 800 | |
| Memory Configuration | | |
| DDR Density (per Channel) | 8Gb | |
| DDR Bus Width | 16 | |
| Number of Ranks | 1 | |
| Number of DDR Clocks | 2 | |
| Local Data Bus | | |
| Local Data Bus Type | AXI4 | |
| Address Width | 30 | |
| Data Width | 128 | |
| ID Width | 2 | |
| Write Ordering Queues | 4 | |
| Read Ordering Queues | 4 | |
| Enable Local Bus Clock | <input checked="" type="checkbox"/> | |
| Enable APB I/F | <input checked="" type="checkbox"/> | |

No DRC issues are found.

Generate Cancel

Figure 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices

Table 4.3 provides the configuration details of the LPDDR4 Memory Controller IP for the Lattice Avant device.

Table 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices

| Property | Value | Description |
|----------------------------|----------|---|
| I/O Buffer Type | LVSTL_II | Set the I/O buffer type standard to LVSTL_II |
| DDR Command Frequency | 800 MHz | Maximum supported frequency on Lattice Avant devices |
| Enable DBI | False | Not used in this reference design |
| Reference Clock: Frequency | 100 MHz | PLL reference clock input to be connected from the top module. Refer to the Clocking Scheme Overview - Lattice Avant Devices section for details. |
| DDR Density (per Channel) | 8 Gb | Configure the DDR DRAM density based on the memory module on the development kit. |
| DDR Bus Width | 16 | Configure the DDR DRAM bus width based on the memory module on the development kit. Note: The Lattice Avant-AT-E Evaluation Board limits only single DDR channel usage, hence the Bus Width is set to 16 only. |
| Number of DDR Clocks | 2 | Total of two DDR clocks per dev kit |
| Data Width | 128 | Data width on the AXI interface of the memory controller |
| ID Width | 2 | ID width on the AXI interface of the memory controller. This value must match with the AXI Manager in the Interconnect IP/ Refer to the AXI4 Interconnect IP section for details. |
| Write/Read Ordering Queues | 4 | Set the ordering queues that process the write and read data requests. More queues means better ordering of access to the DDR bus. This minimizes the gaps between accesses but consumes more resources. The default value is used to balance performance and resource utilization. |
| Enable Local Bus Clock | True | Enable the clock domain crossing logic for the local data bus. |
| Enable APB I/F | True | Enable APB interface, which allows CPU to access the memory controller registers. |

Note:

1. Default settings are used for properties in the *Memory Device Timing* tab.

4.3. AXI4 Interconnect IP

The AXI4 Interconnect IP is used to connect the RISC-V RX instruction and data AXI ports (managers) to various subordinates within the system. This includes the system memory, LPDDR4 memory controller, SPI flash controller, and peripherals. The interconnect output interfaces are AXI-based. As such, conversion bridges are needed when connecting the AXI4 Interconnect to AHB-Lite or APB-based subordinates.

Figure 4.4 shows the AXI4 Interconnect IP configuration in the *General* tab.

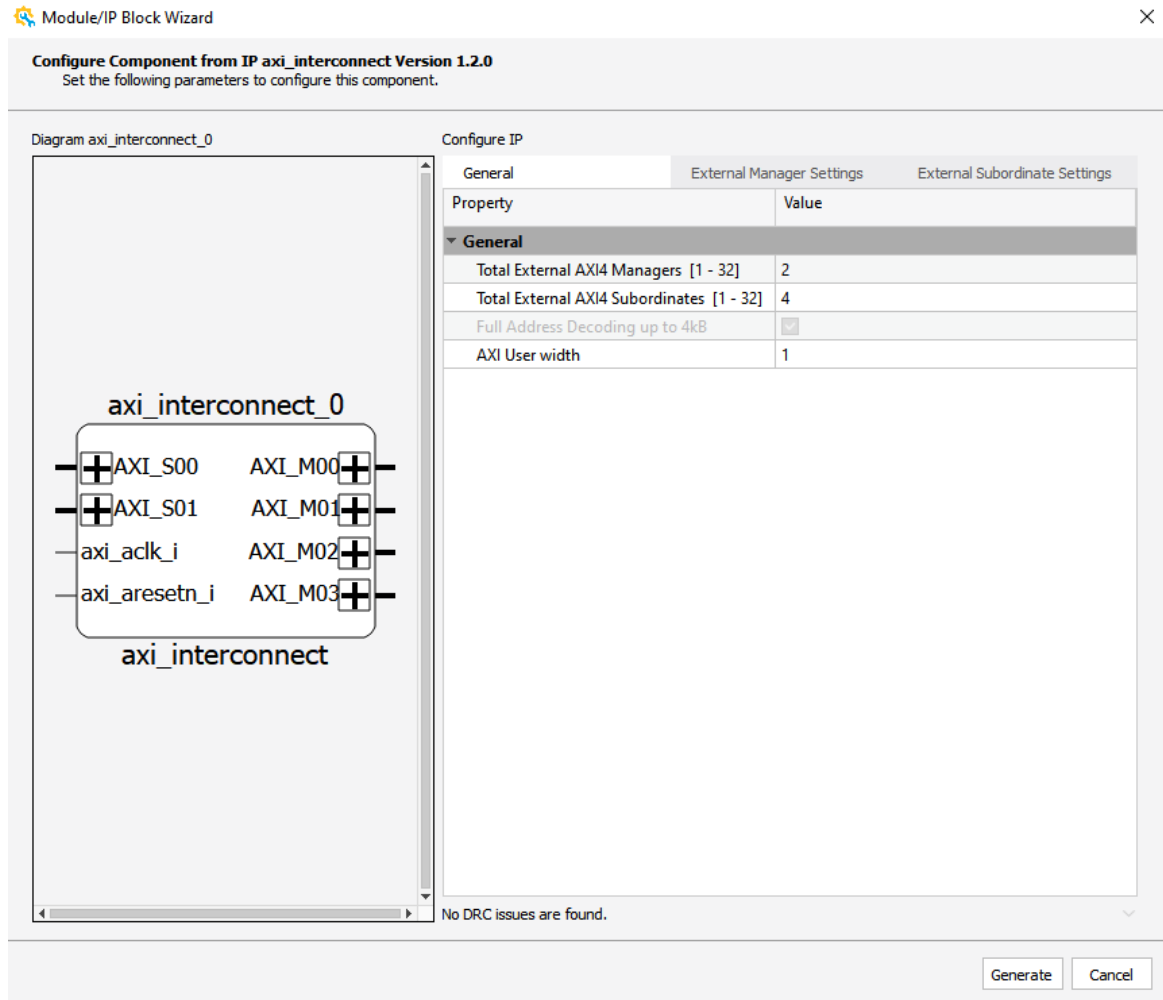


Figure 4.4. AXI4 Interconnect IP Configuration

Table 4.4 provides the configuration details of the AXI4 Interconnect IP.

Table 4.4. AXI4 Interconnect IP Configuration

| Property | Value | Description |
|--------------------------------------|-------------|---|
| General Tab | | |
| Total External AXI4 Managers | 2 | For interface to RISC-V RX instruction and data ports respectively |
| Total External AXI4 Subordinates | 4 | Four subordinates on which to interface: <ul style="list-style-type: none"> LPDDR4 AXI data path System memory data path SPI flash data path All APB based registers space (via APB Interconnect) |
| AXI User Width | 1 | AXI User is not used and set to minimum value |
| External Manager Settings Tab | | |
| External Manager AXI ID width | 1 | Set to 1 to match the RISC-V RX AXI interface width |
| AXI Manager Max Address Width | 32 | Maximum address width that can be set for AXI manager |
| AXI Manager Max Data Width | 32 | Maximum data width that can be set for AXI manager |
| AXI Manager Max no of ID supports | 1 | Maximum number of IDs that can be set for AXI manager |
| External Manager AXI Access Type | WR | Write and read access for both RISC-V RX AXI managers |
| External Manager AXI protocol | AXI4 | Full AXI4 for both RISC-V RX AXI managers |
| External Manager Address Width | 32 | Actual address width for each RISC-V RX AXI manager interface |
| External Manager Data Width | 32 | Actual data width for each RISC-V RX AXI manager interface |
| External Manager No of ID | 1 | Actual number of IDs (RISC-V RX has only one ID) |
| External Manager Write/Read accept | 8 | Number of outstanding write and read transactions that each manager interfaces can accept. Use default value in this design. |
| External Manager Priority | Round Robin | Use <i>Round Robin</i> scheme for each manager interface. Choose responses from different subordinates (at write response channel and read response channel). |

| Property | Value | Description |
|--|--|---|
| External Subordinate Settings Tab | | |
| External Subordinate AXI ID Width | 2 | Subordinate ID width = Manager ID width + log2 (number of managers) = 1 + log2(2) = 2 |
| External Subordinate Max Address Width | 32 | Address width maximum at 32 bits for all subordinates |
| External Subordinate Max Data width | 256 (CertusPro-NX devices) 128 (Lattice Avant devices) | Data width maximum at 256/128 bits to match with LPDDR4 memory controller AXI interface |
| External Subordinate AXI Access Type | WR | Write and read access for all subordinates |
| External Subordinate Protocol Type | AXI4 | Full AXI4 for all subordinates. Subsequent conversion from AXI4 to AHB-Lite or APB involves bridges. |
| External Subordinate Address width | 32 | Actual address width for each subordinate |
| External Subordinate Data width | 256 (subordinate 0 on CertusPro-NX devices) 128 (subordinate 0 on Lattice Avant devices) 32 (other subordinates) | Data width is set to 256/128 on Subordinate 0, which is connected to LPDDR4 memory controller AXI interface. The remaining are set to 32. |
| External Subordinate Write/Read Issue | 8 | Number of write and read transactions that each subordinate can issue. Use default value in this design. |
| External Subordinate Fragment Cnt | 8 | Number of address fragments that each subordinate has. Propel Builder will populate it according to connections and address map assigned. |
| External Subordinate Priority | Round Robin | Use <i>Round Robin</i> scheme for each subordinate. Choose requests from different Managers (at write and read address channel) |

4.4. System Memory

The System Memory IP is used to store the initial software code that RISC-V RX executes out from reset. The LPDDR4 external memory is not ready until the memory training is completed. The initial software is required to ensure the LPDDR4 memory is ready before RISC-V RX accesses it. Using the System Memory IP allows the initial software to be pre-compiled into the FPGA bitstream file. Once the device configuration process is completed, the System Memory will contain the valid initial software. Refer to the [RISC-V RX Software Flow](#) section for details.

Figure 4.5 shows the System Memory IP configuration in the *General* tab.

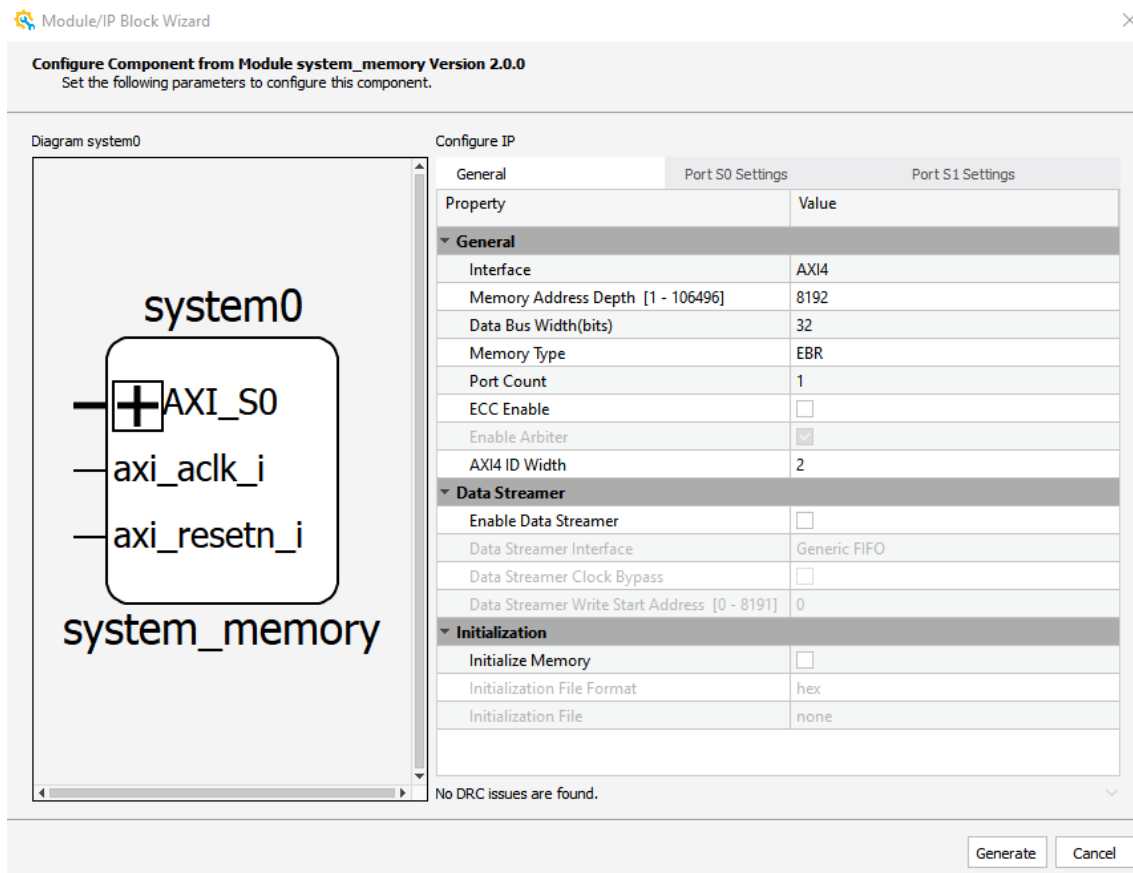


Figure 4.5. System Memory IP Configuration

Table 4.5 provides the configuration details of the System Memory IP.

Table 4.5. System Memory IP Configuration

| Property | Value | Description |
|----------------------------|----------------------|---|
| Interface | AXI4 | Select AXI4 as memory interface that match with RISC-V RX AXI managers |
| Memory Address Depth | 8192 | Total memory size = $8192 * 32 / 8 = 32$ KB |
| Data Bus Width | 32 | Total memory size = $8192 * 32 / 8 = 32$ KB |
| Memory Type | EBR | Use embedded ram to construct the system memory |
| Port Count | 1 | Enable single AXI subordinate port as the arbitration is handled by AXI4 interconnect |
| AXI ID Width | 2 | ID width is set accordingly to match AXI4 Interconnect property |
| Initialize Memory | True | Enable this option to point to the software memory initialization file (.mem) which will compile the software into bitstream after Lattice Radiant compilation. |
| Initialization File Format | Hex | |
| Initialization File | Path to the mem file | |

5. Signal Description

Table 5.1 shows the input/output interface signals for the top-level module.

Table 5.1. Primary I/O

| Signal Name | I/O Type | I/O Width | Description |
|--------------------------------|--------------|-----------|--|
| ddr_pll_refclk_i | Input | 1 | Reference clock input for LPDDR4 memory controller's PLL. |
| rstn_i | Input | 1 | Active low reset input for the reference design. Activate by pressing push button on the board. CertusPro NX devices: SW3 Avant devices: SW1 |
| gpio_io | Input/Output | 8 | General purpose I/O signals connect to LED on board. |
| UART Interface | | | |
| uart_rxd_i | Input | 1 | UART receive input. Connects to the board FTDI2232 TXD signal. |
| uart_txd_o | Output | 1 | UART transmit output. Connects to the board FTDI2232 RXD signal. |
| SPI Flash Interface | | | |
| sclk_o | Output | 1 | Serial clock to SPI flash |
| ss_n_o | Output | 1 | SPI flash chip select |
| mosi_o | Output | 1 | Serial data from SPI flash controller (FPGA) to SPI flash |
| miso_i | Input | 1 | Serial data to SPI flash controller (FPGA) from SPI flash |
| wpj_o | Output | 1 | SPI flash write protect. Not used. |
| flash_rst_n | Output | 1 | External SPI flash reset signal. Assign to 1'b1 to release flash from reset. Note: This applies only if Lattice Avant device is the target device. |
| LPDDR4 Memory Interface | | | |
| ddr_ca_o | Output | 6 | LPDDR4 command/address |
| ddr_ck_o | Output | 1 | LPDDR4 clock |
| ddr_cke_o | Output | 1 | LPDDR4 clock enable |
| ddr_cs_o | Output | 1 | LPDDR4 chip select |
| ddr_dmi_io | Input/Output | 4 | LPDDR4 data mask |
| ddr_dq_io | Input/Output | 32 | LPDDR4 data |
| ddr_dqs_io | Input/Output | 4 | LPDDR4 data strobe |
| ddr_reset_n_o | Output | 1 | Memory reset signal |

6. RISC-V RX Software Flow

This section describes the software flow of the reference design.

6.1. RISC-V RX Boot up Sequence

This section describes the RISC-V RX CPU boot up sequence that enables the CPU to execute software code from the LPDDR4 memory.

Here are the terms used and their descriptions:

- **Boot up** – Process of starting the RISC-V RX CPU, loading software into the LPDDR4 memory, and executing the software
- **Bootloader** – Software that initializes, sets up, and loads the application image into the LPDDR4 memory
- **Application** – Software that is loaded into the LPDDR4 memory and executed by the CPU at the end of boot up process
- **SPI Flash** – Non-volatile memory that can store application software persistently
- **Application image** – Converted application software with extra information and is used to program into the flash

Figure 6.1 illustrates the boot up sequence with reference to the terminologies above.

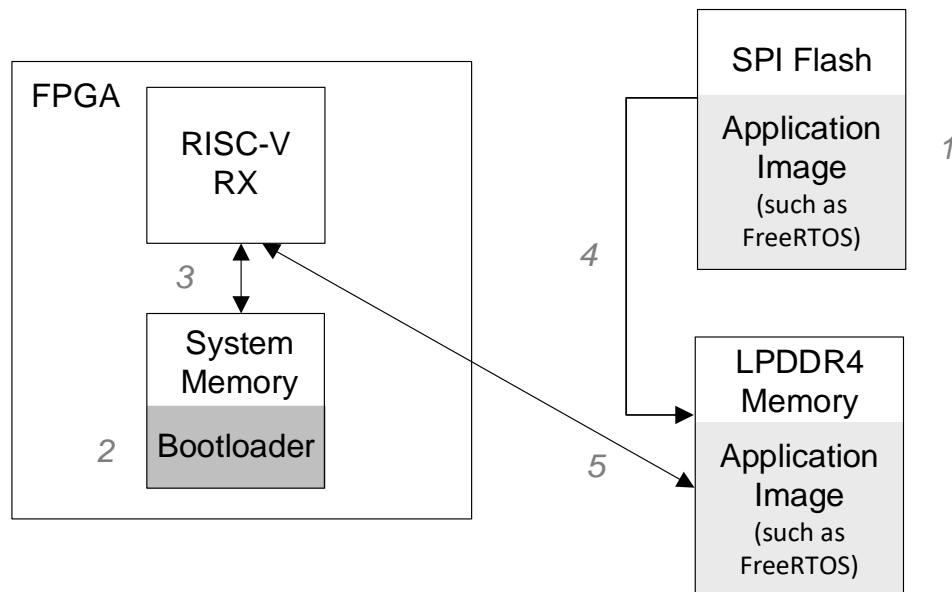


Figure 6.1. Boot up Sequence

The following is the boot up sequence shown in Figure 6.1:

1. The application image (such as FreeRTOS) is pre-programmed into flash before booting up.
2. The system memory is pre-initialized with the bootloader (in bitstream file) during the FPGA configuration process.
3. The RISC-V RX CPU executes the bootloader when reset is de-asserted.
4. The bootloader initializes the LPDDR4 memory controller and copies the application image from flash to memory.
5. The RISC-V RX CPU executes the application in the LPDDR4 memory when it is ready and this completes the boot up process.

6.2. Bootloader Software Flow Chart

Figure 6.2 illustrates the bootloader software flow.

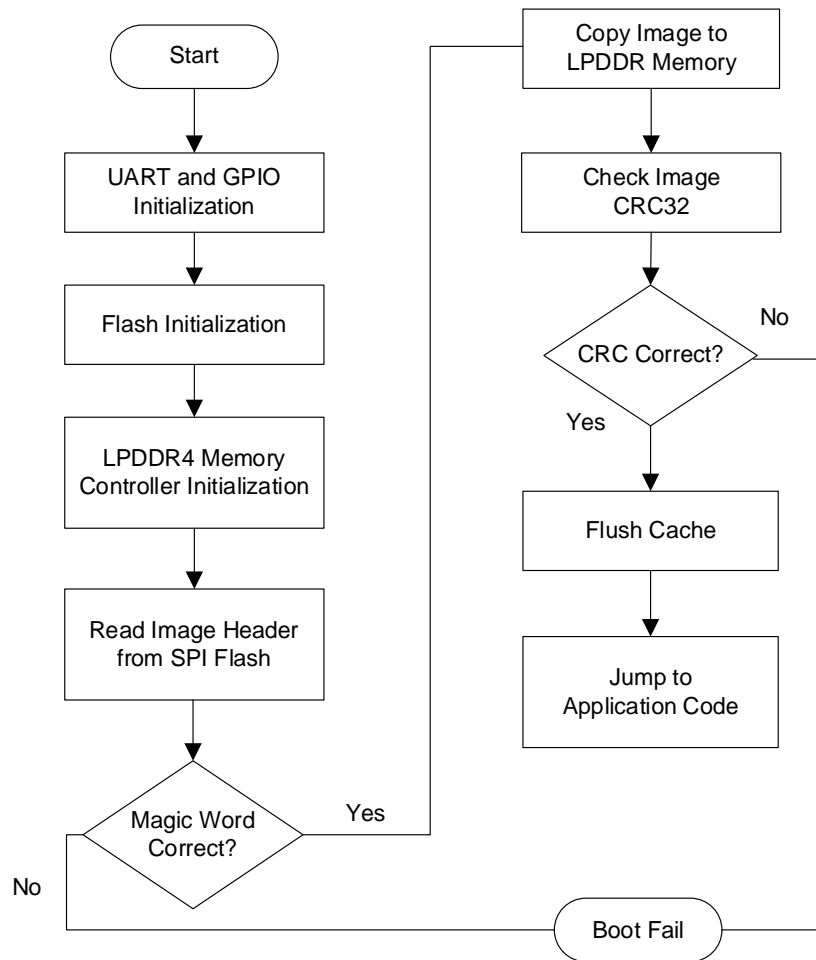


Figure 6.2. Bootloader Software Flow

The following is the bootloader sequence shown in Figure 6.2:

1. Initialize the UART interface and GPIO (for driving LED)
2. Initialize the SPI flash controller. Read the flash ID and display the value.
3. Initialize the LPDDR4 memory controller. Trigger the memory training process and monitor the completion. Ensure that all trainings pass before proceeding.
4. Read the application image header from SPI flash and check for the magic word in the header.
Refer to the [Application Image Format](#) section for details.
5. Proceed if the magic word matches. Otherwise, go to boot fail and stop booting.
6. Copy the application image from flash to the LPDDR4 memory.
7. Generate CRC32 checksum for image in the LPDDR4 memory and compare it with the value from the image header.
8. Proceed if checksum matches. Otherwise, go to boot fail and stop booting.
9. Flush the CPU cache.
10. Jump to the application code address in the LPDDR4 memory.

6.3. Application Image Format

The application image contains the application software binary and extra information appended as the header. Figure 6.3 illustrates the application image format. The header information is required for bootloader to perform image validation and integrity check during boot up.

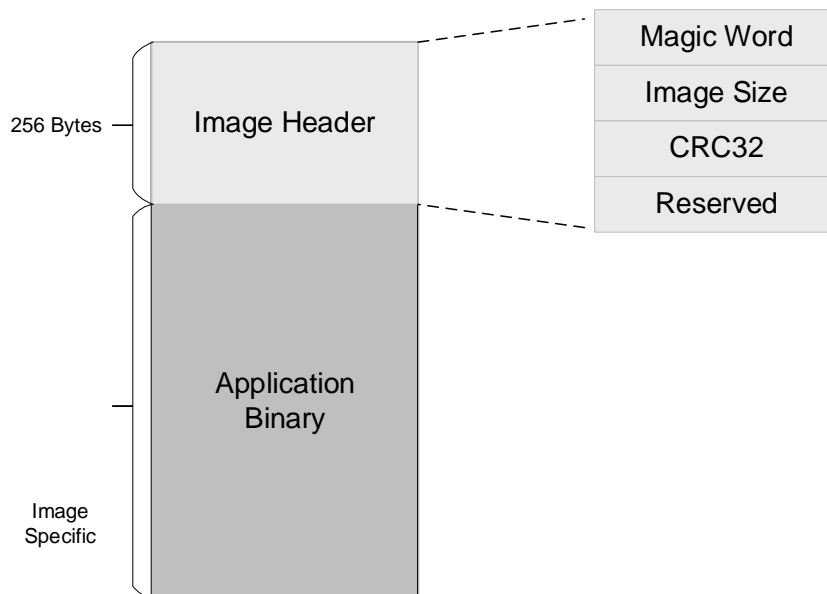


Figure 6.3. Application Image Format

Table 6.1 provides the image header details.

Table 6.1. Application Image Header

| Header Property | Value | Description |
|-----------------|------------|--|
| Magic Word | 0x4C534343 | Represents LSCC in ASCII notation. Magic Word is used to check if a valid image header exists in flash. |
| Image Size | Variable | Application binary size in number of bytes. This provides the total size to be copied by the bootloader. |
| CRC32 Checksum | Variable | CRC32 checksum calculated for the application binary. |
| Reserved | N/A | Padding to align the image header to flash page size (256 bytes). |

6.4. Application Image Generation

This section describes the application image generation process to produce the image described in the [Application Image Format](#) section.

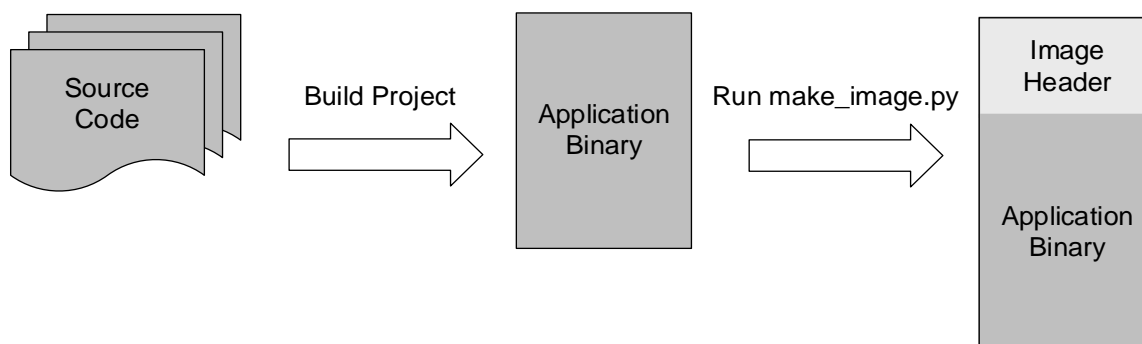


Figure 6.4. Application Image Generation

The following is the application image generation process shown in Figure 6.4.

1. Build the application software project from source code using Propel SDK. This creates the application binary file (.bin).
2. Run the *make_image* python script (provided by this reference design) to append the image header on top of application binary.

Refer to the [Generating the Flash Image](#) section for details.

7. Implementing the Reference Design on the Board

This section describes the procedure for running the RISC-V LPDDR4 reference design using the prebuilt binary files in the design package.

The following Lattice development boards are used to run the design:

- [CertusPro-NX Versa Board](#)
- [Avant-E Evaluation Board](#)

Follow the procedure in the [Setting up the CertusPro-NX Versa Board](#) section or in the [Setting up the Lattice Avant-AT-E Evaluation Board](#) section, depending on the board that you are using.

7.1. Extracting the Reference Design Files

Before running the reference design, download the design package .zip file from the Lattice Semiconductor website.

- CertusPro-NX-RISC-V-LPDDR4-RD.zip
- AVANT-RISC-V-LPDDR4-RD.zip

Unzip the .zip file to your local directory, for example to C:\workspace\.

The extracted directory is named *CertusPro-NX-RISC-V-LPDDR4-RD* or *AVANT-RISC-V-LPDDR4-RD*, depending on which design you downloaded into your workspace. This path is referred to hereinafter as *<my_work_dir>*. For example, *<my_work_dir>* = C:\workspace\CertusPro-NX-RISC-V-LPDDR4-RD.

7.2. Setting up the CertusPro-NX Versa Board

This section provides the procedure for setting up the CertusPro-NX Versa Board, shown in [Figure 7.1](#)

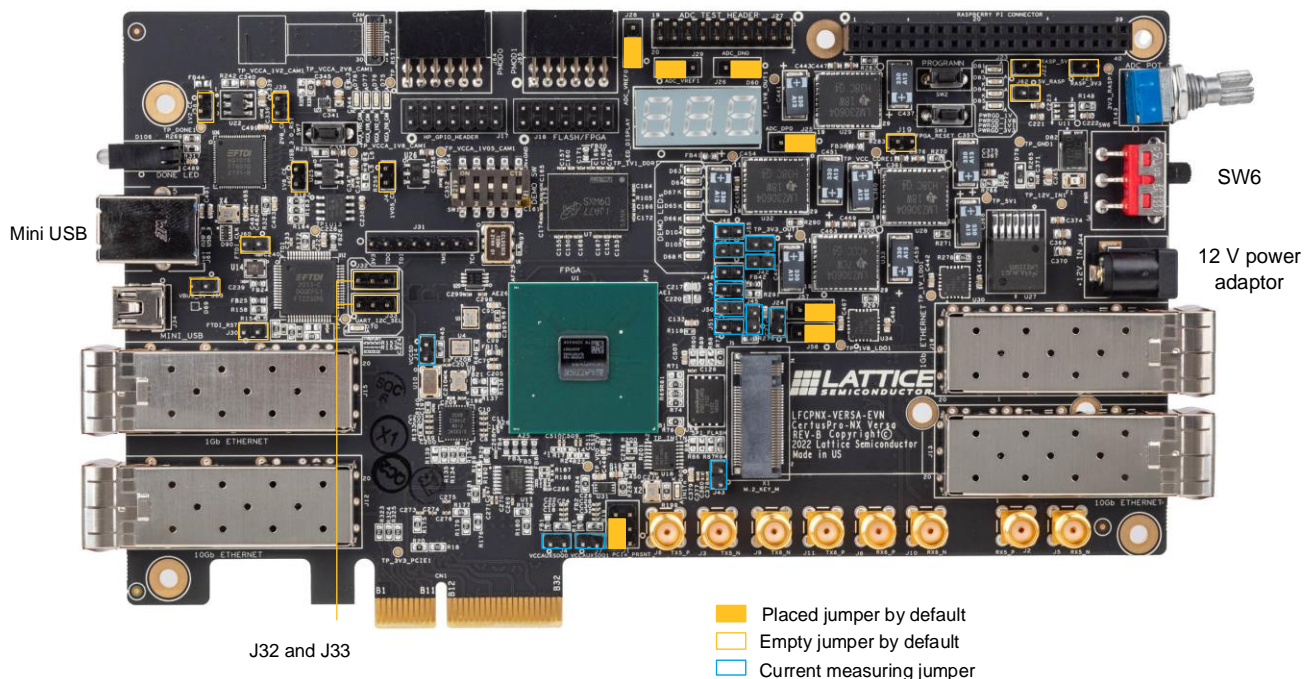


Figure 7.1. CertusPro-NX Versa Board

To set up the CertusPro-NX board:

1. Connect the 12 V power adaptor to J44.
2. Connect the Mini USB Type A cable from the PC to J34.
3. Connect (using jumpers) Pin 1 and Pin 2 on both J32 and J33 to enable UART.
4. Turn on switch SW6.

7.3. Setting up the Lattice Avant-AT-E Evaluation Board

This section provides the procedure for setting up the Lattice Avant-AT-E Evaluation Board, shown in Figure 7.2.

Note: The Lattice USB Programming Cable [HW-USBN-2B](#) is required to program the SPI flash on the board.

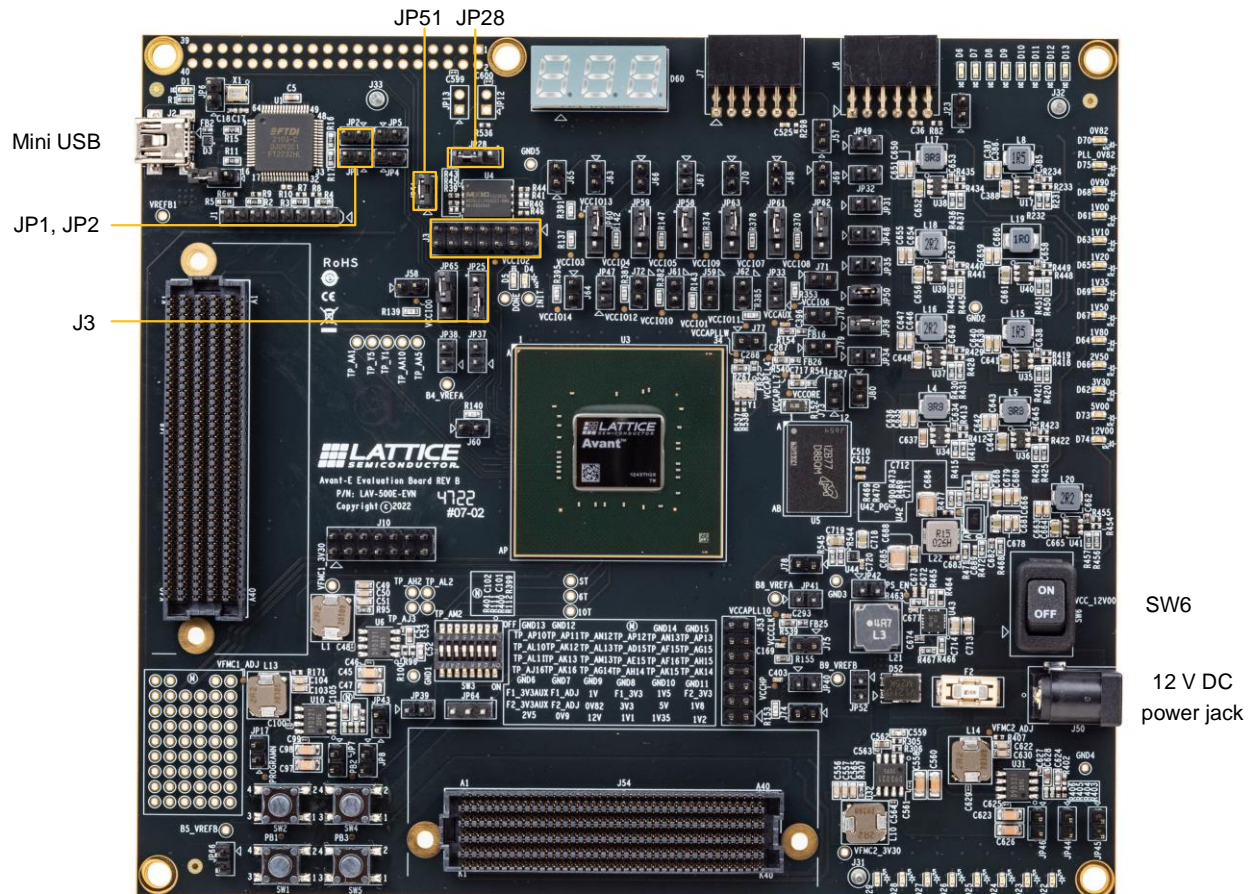


Figure 7.2. Lattice Avant-AT-E Evaluation Board

To set up the Lattice Avant-AT-E board:

1. Connect the 12 V power adaptor to J50.
2. Connect the Mini USB Type A cable from the PC to J2.
3. Connect (using jumpers) Pin 1 and Pin 2 on both JP1 and JP2 to enable UART.

4. Connect the HW-USB-N-2B cable to J3 in the following manner (this is used for SPI flash programming):
 - a. Connect ISPEN/PROG (yellow wire) from cable to J3 pin 2.
 - b. Connect TDI/SI (orange wire) from cable to J3 pin 5.
 - c. Connect TDO/SO (brown wire) from cable to J3 pin 7.
 - d. Connect VCC (red wire) from cable to J3 pin 10.
 - e. Connect TCK/SCLK (white wire) from cable to J3 pin 12.
 - f. Connect GND (black wire) from cable to J3 pin 14.
5. Connect (using jumpers) the following pins:
 - a. Pins 1 and 2 on JP51
 - b. Pins 2 and 3 on JP28
 - c. Pins 1 and 2 on JP25
 - d. Pins 1 and 2 on JP65
6. Turn on switch SW6.

7.4. Setting up the UART Terminal

The software code in this reference design displays messages on the terminal through the UART interface.

To set up the UART terminal:

1. Launch Tera Term.
Note: Other terminal emulator software, such as PuTTY, may be used.
2. In the **New connection** dialog box, select **Serial**.
3. Select the correct COM port from the **Port** drop-down menu. It is usually the larger number. For example, if you see COM1 and COM2, choose COM2.
4. Click **OK**

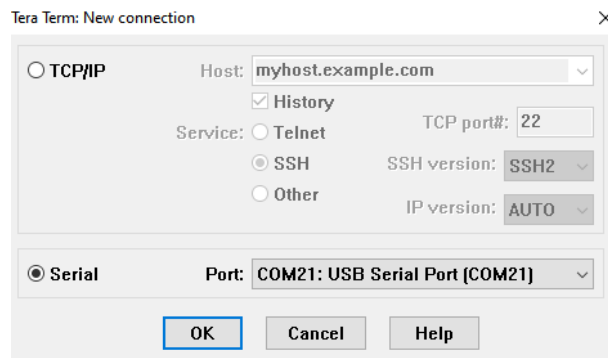


Figure 7.3. Tera Term New Connection

5. Click **Setup > Serial Port**.
6. In the **Serial port setup and connection** interface, select **115200** in **Speed**.
7. Select the rest of the options based on [Figure 7.4](#).
8. Click **New setting** to complete the process.

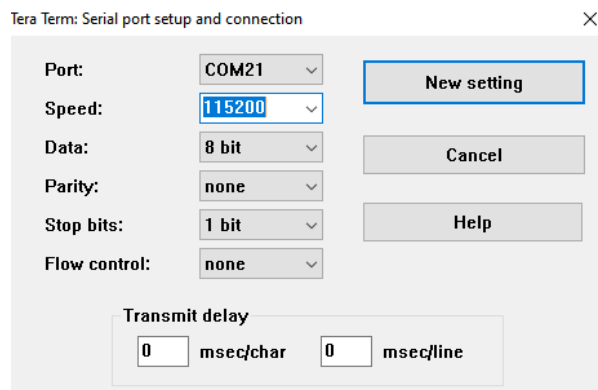


Figure 7.4. Tera Term Serial Port Setup and Connection

7.5. Programming the Application Image to SPI Flash

The application image is programmed into SPI Flash before starting the bootloader software. This ensures there is a valid image for bootloader to load into LPDDR4 memory.

To program the application image to SPI Flash:

1. Launch Radiant Programmer 2023.1 from the Windows Start menu.

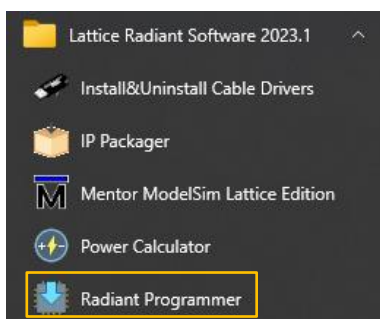


Figure 7.5. Windows Start Menu > Radiant Programmer

2. In the In the **Getting Started** dialog box, select **Open an existing programmer project**.

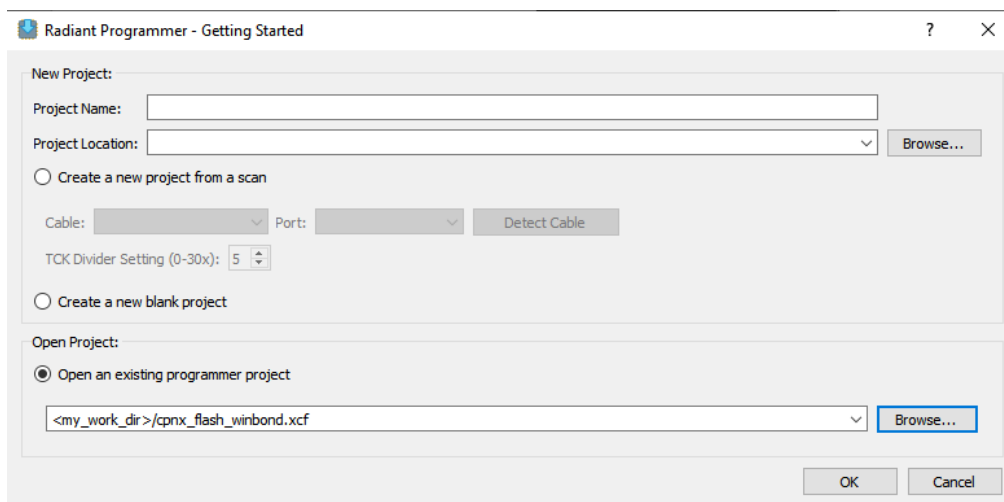


Figure 7.6. Radiant Programmer - Getting Started Dialog Box

3. Click the **Browse** button and navigate to `<my_work_dir>\prebuilt_images` directory.
4. Select the corresponding .xcf file for your board as indicated in [Table 7.1](#).

Table 7.1. Programming Files Selection

| Board | Flash Device | XCF File |
|-------------------------------------|----------------------|--------------------------|
| CertusPro-NX Versa Board | Winbond W25Q512JV | cpnx_flash_winbond.xcf |
| CertusPro-NX Versa Board | Macronix MX25L51245G | cpnx_flash_macronix.xcf |
| Lattice Avant-AT-E Evaluation Board | Macronix MX25L51245G | avant_flash_macronix.xcf |

5. Click **OK**.
6. In the main interface, click **Run > Program Device**.
7. The **Output** console displays the **Operation successful** message.

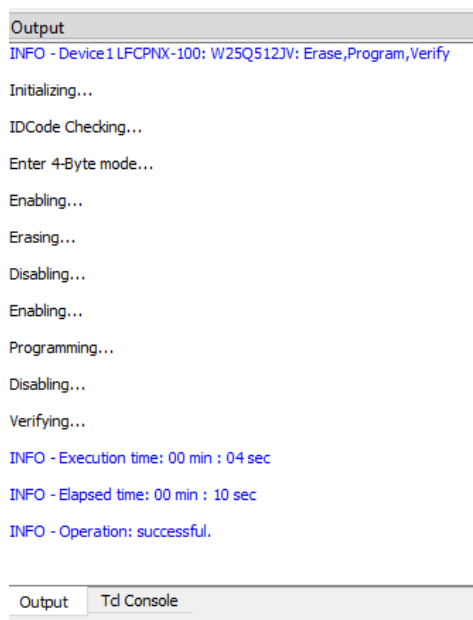


Figure 7.7. Output Console

Note: If the flash programming fails, refer to the [Debugging the Design](#) section.

7.6. Programming the FPGA Bitstream

This bitstream (.bit) file is programmed into the FPGA device. This bitstream contains the RISC-V LPDDR4 design with the bootloader code pre-initialized in system_memory.

To program the bitstream into the device:

1. In the Radiant Programmer main interface, click **File > Open Project**.
2. To open the Programmer project file, browse to `<my_work_dir>\prebuilt_images` and select `cpnx_bitstream_program.xcf` or `avant_bitstream_program.xcf`.
3. Click **Open**.
4. In the main interface, click **Run > Program Device**.
5. The **Output** console displays the **Operation successful** message.

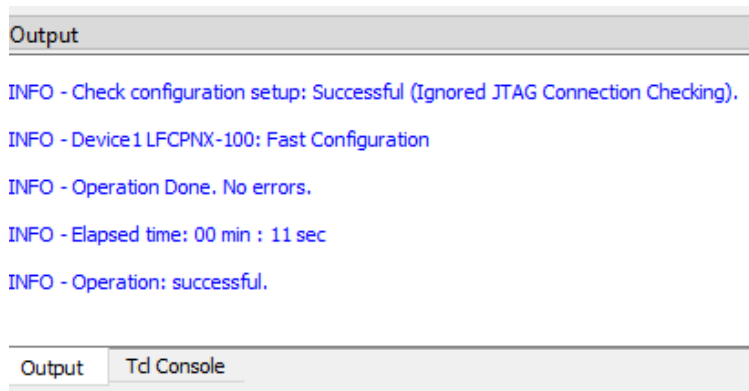


Figure 7.8. Output Console

Note: If you encounter error during bitstream programming for the Avant device, follow these steps:

1. Disconnect the HW-USB-2B cable from PC.
2. Power cycle the board (switch off and on).
3. Repeat the Program Device steps.

7.7. Verifying the Results

After FPGA configuration is completed, observe the messages in the Tera Term terminal as shown in Figure 7.9.

```

COM21 - Tera Term VT
File Edit Setup Control Window Help
*****
* RISC-U RX Reference Design Bootloader *
*****

SPI Flash Controller Initialization: Passed
Flash manufacturer ID: ef

Memory Controller Initialization : Passed

Load and CRC check image
SRC : Flash addr 0
DEST: DDR addr 40000000
Passed

Jumping to the application
.....

FreeRTOS v10.0.1 on RISC-U.
the granularity of pmp is 4.
#####
pmp entry0: mode=0x01, perm=0x07, addr=0x10001f6a(*4)=0x40007da8, locked=0
pmp entry1: mode=0x01, perm=0x00, addr=0x10001f6b(*4)=0x40007dac, locked=1
pmp entry2: mode=0x01, perm=0x00, addr=0x10001f6e(*4)=0x40007db8, locked=0
pmp entry3: mode=0x01, perm=0x07, addr=0x3fffffff(*4)=0xffffffffc, locked=0
#####
pmp test begin:
p sstatus=0x00000100:
p mstatus=0x00000188:

p write/read secured region in task:

p expectation<locked>: write/read 0x40007da8 will fail
p write/read 0x40007da8 begin, in=0x00001234
p write/read 0x40007da8 done, out=0x00001234

p expectation<m-mode>: write/read 0x40007dac will succeed
p write/read 0x40007dac begin, in=0x00005678
p write/read 0x40007dac done, out=0x00005678

p write/read secured region with sys-call:

p expectation: write/read 0x40007dac will succeed
p write/read 0x40007dac begin, in=0x0000def0
p write/read 0x40007dac done, out=0x0000def0

p pmp test end
#####
task 1073763688 is running, 0
task 1073761544 is running, 0
task 1073763688 is running, 1
task 1073761544 is running, 1
task 1073763688 is running, 2
task 1073761544 is running, 2
task 1073763688 is running, 3
suspend task 1073763688
task 1073761544 is running, 3
task 1073761544 is running, 4

```

Figure 7.9. RISC-V RX and LPDDR4 Reference Design Results

The following process is observed:

1. The bootloader software executes first. It initializes the SPI flash controller and the LPDDR4 memory controller.
2. The bootloader copies the image from flash to the memory. It checks the CRC to be correct before proceeding.
3. If everything is good, the bootloader instructs the CPU to jump to the LPDDR4 memory address and start executing the application.
4. The FreeRTOS application is executed from the LPDDR4 memory.

8. Compiling the Reference Design

This section describes the process of compiling the RISC-V and LPDDR4 Reference Design. Compilation is required to generate the binary files from the source files. The compilation process involves the following software tools for generating the FPGA bitstream and the RISC-V processor software executable files:

- Lattice Propel Builder 2023.1
- Lattice Radiant Software 2023.1
- Propel Software Development Kit (SDK) 2023.1

These sections show the typical design flow for Lattice RISC-V based design:

- [Building and Generating the RISC-V Processor using the Lattice Propel Builder](#)
- [Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software](#)
- [Building the Software Project using Propel SDK](#)

These sections show the specific flow that is needed for this reference design to create the bootloader software and application image:

- [Pre-initializing System Memory](#)
- [Generating the Flash Image](#)

8.1. Building and Generating the RISC-V Processor using the Lattice Propel Builder

The RISC-V processor system in this reference design is built and generated using Lattice Propel Builder. The components (IP) are instantiated and connected using the schematic graphical user interface of Propel builder, as shown in [Figure 8.1](#).

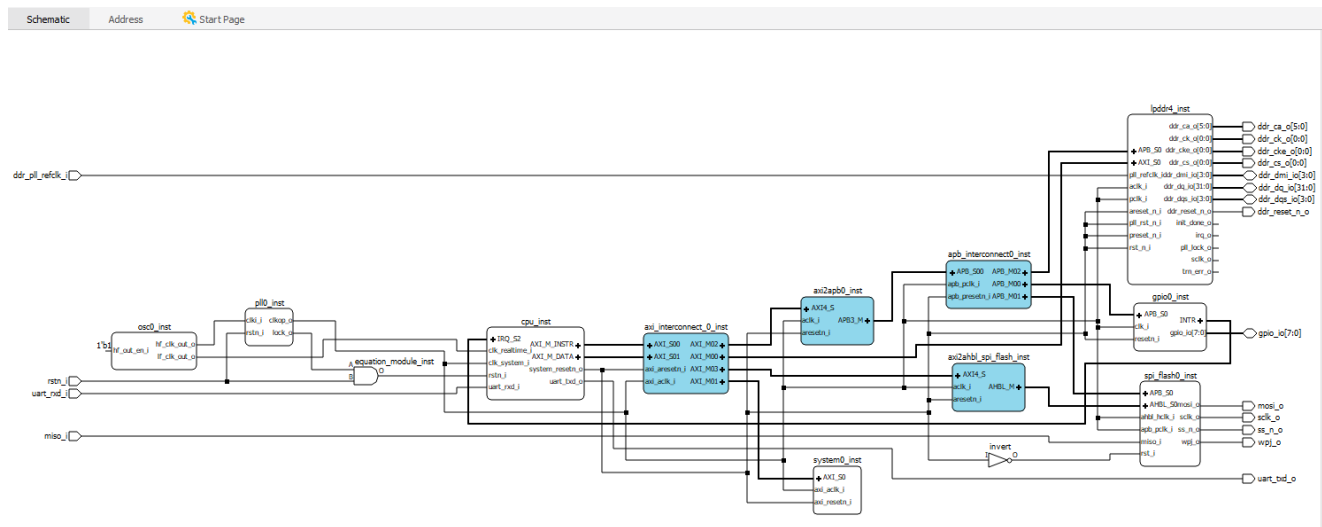


Figure 8.1. Lattice Propel Builder Schematic View

The following procedure is required after making modifications to the design. Refer to the [Customizing the Reference Design](#) section for details.

To generate the RISC-V processor system:

1. Launch Lattice Propel Builder 2023.1 from Windows Start menu.

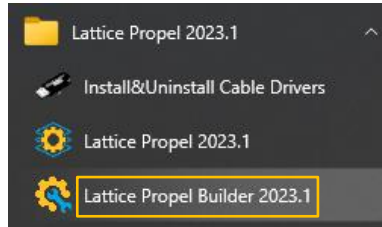


Figure 8.2. Windows Start Menu > Lattice Propel Builder

2. In the Propel Builder main interface, click **File > Open Design**. This opens the **Open sbx** dialog box.
3. Navigate to `<my_work_dir>\cpnx_riscv_rx_rd` or `<my_work_dir>\avant_riscv_rx_rd`.
Note: Before performing this step, make sure you have already completed the procedure in the [Extracting the Reference Design Files](#) section.
4. Select `cpnx_riscv_rx_rd.sbx` or `avant_riscv_rx_rd.sbx` and click **Open**.
5. In Propel Builder, click **Design > Validate Design**.
You should see no error in the **Tcl Console** window.

```
Tcl Console
% sbp_design drc
INFO <2359136> - Start: sbp_design drc.
INFO <2359137> - Finished: sbp_design drc.
```

Figure 8.3. TCL Console – No Error for Validate

6. Click **Design > Generate**. You should see no error in the Tcl Console window:

```
Tcl Console
% sbp_design generate
INFO <2359189> - Start: sbp_design generate.
INFO <2359190> - Finished: sbp_design generate.
% sbp_design save
% sbp_design pge sge -i {C:/CertusPro-NX-RISC-V-LPDDR4-RD/cpnx_riscv_rx_rd/
cpnx_riscv_rx_rd.sbx} -o {C:/CertusPro-NX-RISC-V-LPDDR4-RD/cpnx_riscv_rx_rd/..} -nc
Warning[FileExistence]:No available driver for memory_controller
```

Figure 8.4. TCL Console – No Error for Generate

Note: The memory controller driver warning can be safely ignored.

8.2. Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software

Lattice Radiant software is used to synthesize the RTL design files (from Propel Builder) and generate the FPGA bitstream file.

To compile the project.

1. Open Lattice Radiant Software 2023.1 from the Windows **Start** menu.

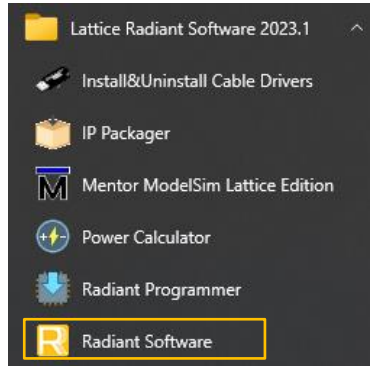


Figure 8.5. Windows Start Menu > Radiant Software

Note: Do not launch the Lattice Radiant software from Lattice Propel Builder. This causes the Radiant top-level setup to be overwritten. This is a known issue to be fixed in a future Lattice Propel Builder version.

2. In the Lattice Radiant software main interface, click **File > Open > Project**.
3. Navigate to `<my_work_dir>\cpnx_riscv_rx_rd.rdf` or `<my_work_dir>\avant_riscv_rx_rd.rdf` and click **Open**.
4. Click **Export Files** to start bitstream compilation.

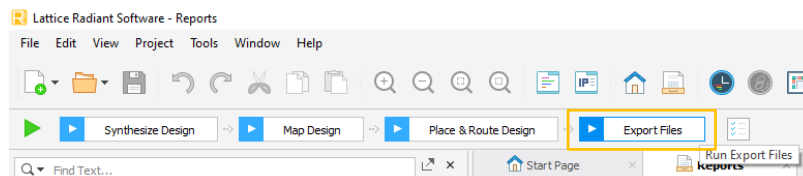


Figure 8.6. Export Files

5. The compilation may take some time to complete. The .bit file is generated/updated in the project path (`<my_work_dir>\impl_1`) after compilation is completed successfully.

Note: If you observe timing violations after Lattice Radiant compilation, this may be due to the Placement Iteration Start Point number for Place & Route not working optimally on your machine. In this case, perform the following steps:

1. In Lattice Radiant software, click **Project > Active Strategy > Place & Route Design Settings**.
2. Change **Placement Iterations** from 1 to 5.
3. Change **Placement Save Best Run** from 1 to 5.
4. Click **OK**.
5. Click **Export Files**.

This reruns the Place & Route Design with five different incremental start point values. The Place & Route Reports show all five placement results and selects the best timing result.

8.2.1. Floor-planning Constraints

The RISC-V LPDDR4 reference design for Avant devices uses physical constraints (floor-planning) to achieve better hold time slacks for the SPI Flash Controller IP used in this design. Groups are created using the `ldc_create_group` constraint for the sub-modules in the SPI Flash Controller as follows:

- `ahb_intf`
- `apb_intf`
- `spi_flash_intf`

Each group are assigned to a specific site in FPGA using the `ldc_set_location` constraint. The `ahb_intf` and `apb_intf` groups are placed on sites with specific routing distance from the `spi_flash_intf` group. The routing distances compensate for the large clock skews between the sub-modules.

The actual physical constraints, which are in the `avant_riscv_rx_rd.pdc` file, are listed as follows:

```
# Floorplanning constrains for SPI flash hold time
ldc_create_group -name ahb_intf -bbox {24 25} [get_cells
{avant_riscv_rx_rd_inst/spi_flash_inst/lscs_spi_flash_inst/ahb_intf_inst }]
ldc_set_location -site {R66C37F} [ldc_get_groups ahb_intf]
ldc_create_group -name apb -bbox {18 26} [get_cells
{avant_riscv_rx_rd_inst/spi_flash_inst/lscs_spi_flash_inst/apb_intf_inst }]
ldc_set_location -site {R47C37F} [ldc_get_groups apb]
ldc_create_group -name spi_flash_intf -bbox {41 20} [get_cells
{avant_riscv_rx_rd_inst/spi_flash_inst/lscs_spi_flash_inst/spi_flash_intf_inst }]
ldc_set_location -site {R47C340F} [ldc_get_groups spi_flash_intf]
```

8.3. Building the Software Project using Propel SDK

Lattice Propel SDK is used to build the RISC-V processor software project. This section provides the procedure building the software project in Propel SDK.

8.3.1. Setting Up the Project Workspace

Launching the Propel SDK tool and setting up an Eclipse workspace for software projects are discussed in this section.

Note: These steps are required only when you create a Propel SDK workspace for the first time. On subsequent Propel SDK launches, you can reuse the previously created workspace.

To create a new Propel SDK workspace:

1. Launch Lattice Propel 2023.1 from the Windows Start Menu.

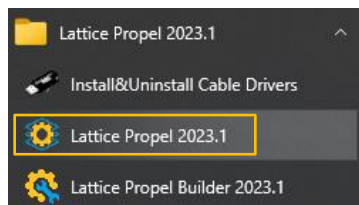


Figure 8.7. Windows Start Menu > Lattice Propel 2023.1

2. In the Lattice Propel Launcher dialog box, click **Browse**.
3. Navigate to `<my_work_dir>`.
4. Create a new folder named `propel_workspace` and click **Select Folder**
5. Figure 8.8 shows the reference design workspace setup. When ready, click **Launch**.

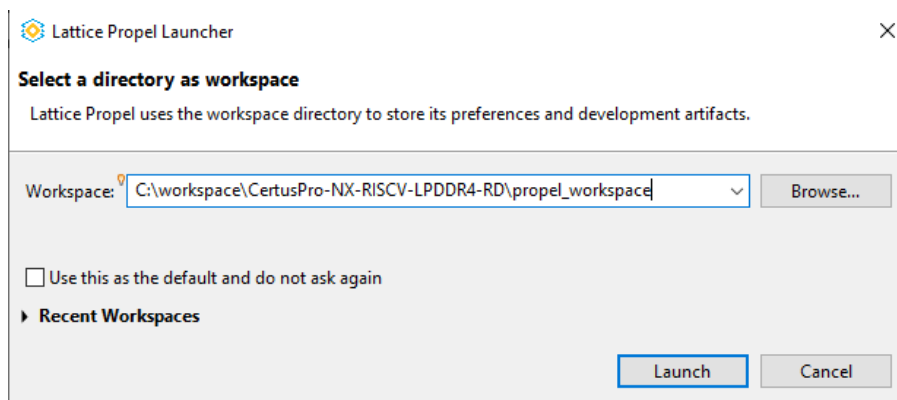


Figure 8.8. Propel SDK Workspace Setup

8.3.2. Building Bootloader and Application Software

The source code for both the bootloader and the application software are provided in the reference design. These are located in `<my_work_dir>\sw`.

- *bootloader_sw.zip* contains the project and code for bootloader software
- *freertos_app_sw.zip* contains the project and code for FreeRTOS application software

To build the software project in Propel SDK.

1. In Propel main interface, click **File > Import**.
2. In the Import wizard Select page, choose Existing Projects into Workspace under General.

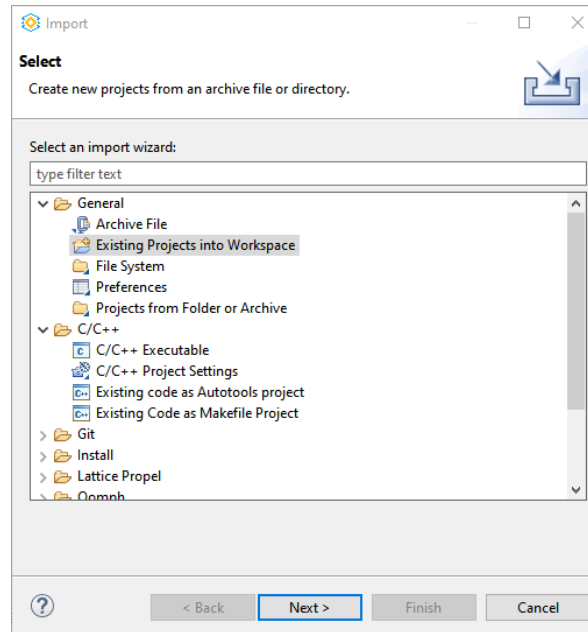


Figure 8.9. Select Existing Projects

3. Click **Next**.
4. In the **Import Projects** page, choose **Select archive file**.
5. Click the **Browse** button and navigate to `<my_work_dir>\sw \bootloader_sw.zip`.
6. Click **Open**.

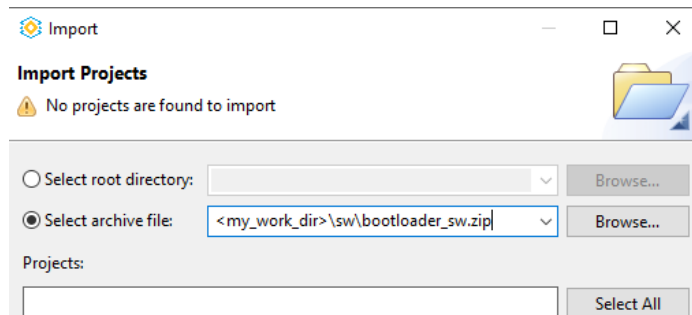


Figure 8.10. Import Projects

7. Click **Finish** to complete the process.
8. The `bootloader_sw` project is listed in **Project Explorer**.



Figure 8.11. Project Explorer

9. In **Project Explorer**, right-click `bootloader_sw` and select **Build Project**.
10. Verify that no error message is shown in the **Console** window.
11. Make sure that the `bootloader_sw.mem` file is generated in the `<my_work_dir>\propel_workspace\bootloader_sw\Debug` directory.

```

CDT Build Console [bootloader_sw]

Building target: bootloader_sw.elf
Invoking: GNU RISC-V Cross C Linker
riscv-none-embed-gcc -march=rv32imc -mabi=ilp32 -msmall-data-limit=8 -mno-save-restore -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -g3 -T "C:/lsc/f2f_emb_traini
Finished building target: bootloader_sw.elf

Invoking: GNU RISC-V Cross Create Listing
riscv-none-embed-objdump --source --all-headers --demangle --line-numbers --wide "bootloader_sw.elf" > "bootloader_sw.lst"
Finished building: bootloader_sw.lst

Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "bootloader_sw.elf"
   text    data     bss     dec     hex filename
   5224      20    2672    7916    1eec bootloader_sw.elf
Finished building: bootloader_sw.siz

Invoking: Lattice Create Memory Deployment
riscv-none-embed-objcopy -O binary --gap-fill 0 "bootloader_sw.elf" "bootloader_sw.bin"; srec_cat "bootloader_sw.bin" -Binary -byte-swap 4 -DISable Header -Output "bootloader_sw.mem" -MEM 32
Finished building: bootloader_sw.mem

19:04:00 Build Finished. 0 errors, 2 warnings. (took 10s.522ms)
  
```

Figure 8.12. Console

12. Repeat steps 1 to 11 to build the FreeRTOS application using `freertos_app_sw.zip`.
13. After successfully building the FreeRTOS application, make sure that the `freertos_app_sw.bin` file is generated in the `<my_work_dir>\propel_workspace\freertos_app_sw\Debug` directory.

8.4. Pre-initializing System Memory

As described in the [RISC-V RX Boot up Sequence](#) section, the RISC-V RX processor executes the bootloader software from the system memory when out from reset. The system memory content is pre-initialized with the bootloader software through FPGA configuration.

Note: These steps are required when changing the bootloader software or initializing the system memory with your own software program.

To initialize system memory:

1. In the Propel Builder tool, double click on the **system0_inst** block. This opens the **Module/IP Block Wizard**.

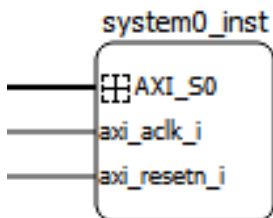


Figure 8.13. system0_inst Block

2. In the **Module/IP Block Wizard**, select the **Initialize Memory** box.
3. Set **Initialization File Format** to **hex**.
4. In the **Initialization File** field, click the “...” browse button.
5. In the **Select File** dialog box, browse to `<my_work_dir>\propel_workspace\bootloader_sw\Debug`.
6. Select `bootloader_sw.mem` and click **Open**.
The final settings are shown in [Figure 8.14](#).
7. In the **Module/IP Block Wizard**, click **Generate** and then **Finish**.

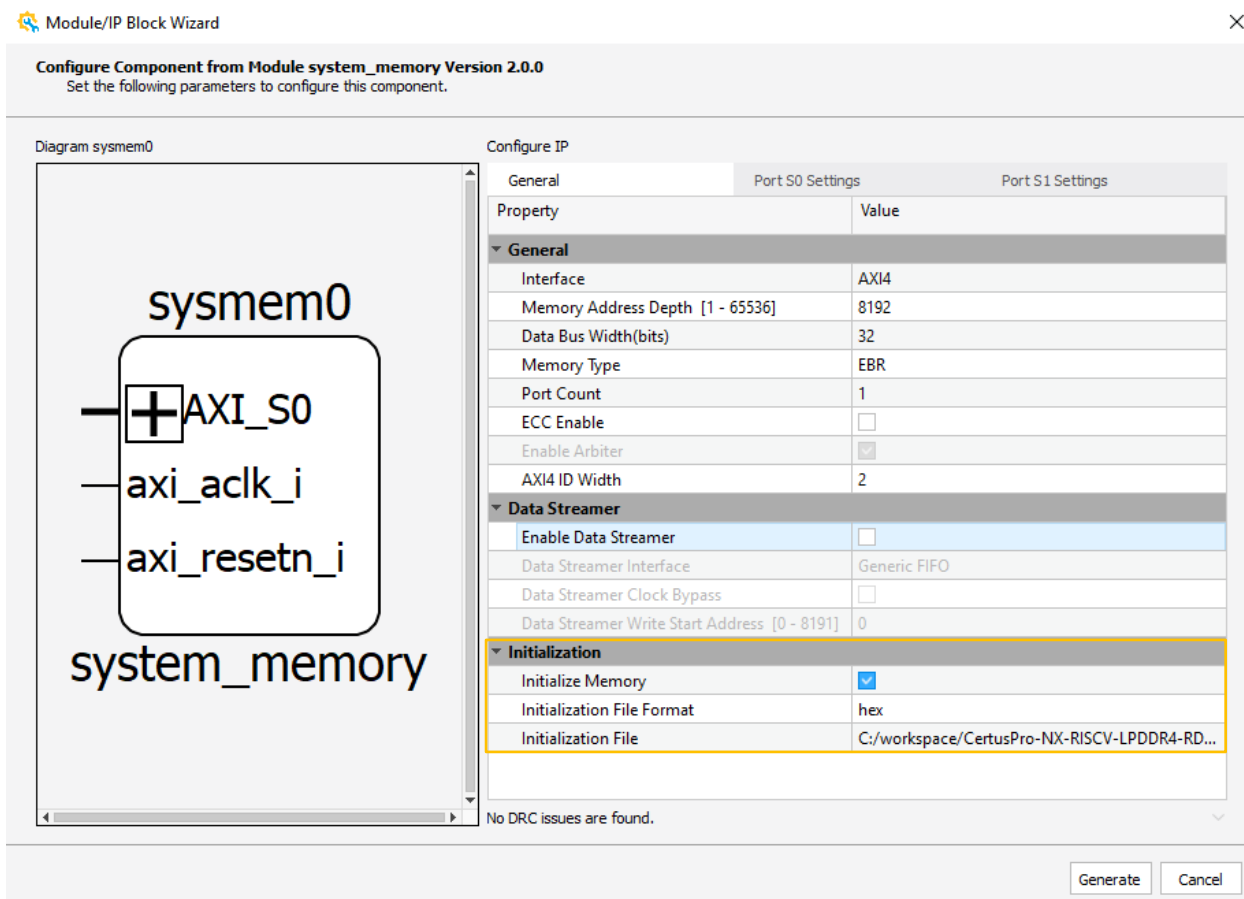


Figure 8.14. System Memory IP Setting to Pre-initialize with Bootloader Software

- In Lattice Propel Builder, click **Design > Generate** to update the output.
- In Lattice Radiant, click **Export Files** to generate the updated bitstream file.

Note: Refer to the following sections for the Propel Builder and Radiant compilation steps:

- [Building and Generating the RISC-V Processor using the Lattice Propel Builder](#)
- [Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software](#)

8.5. Generating the Flash Image

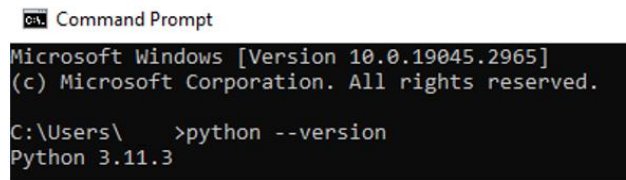
The application image requires header appended using a provided Python script, as described in the [Application Image Generation](#) section. The *make_image.py* script is provided in the reference design package at the `<my_work_dir>\sw` directory.

Note: This is required to make modifications to the application software or to replace the FreeRTOS application with others. The pre-requisite .bin file is available, as described in the [Building Bootloader and Application Software](#) section.

To generate the flash image:

1. Launch Command Prompt from the Windows Start menu.
2. Make sure you have Python 3 installed. To check, enter `python --version`. The output is shown in [Figure 8.15](#).

Note: If you do not have Python 3 installed, go to Microsoft Store to install it.



```
Command Prompt
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ >python --version
Python 3.11.3
```

Figure 8.15. Python Verification

3. In Command Prompt, enter `cd <my_work_dir>\sw` to change directory.
4. Enter the following to run the *make_image* Python script:
`python.exe .\make_image.py ..\propel_workspace\freertos_app_sw\Debug\freertos_app_sw.bin
..\prebuilt_images\freertos_app_boot.bin`
5. The *freertos_app_boot.bin* file generated in the *prebuilt_images* directory.
6. Repeat the steps in the [Programming the Application Image to SPI Flash](#) section to program the image into SPI flash.

9. Customizing the Reference Design

This section describes the customization that can be applied to the reference design. Hardware related modifications are made using the Propel Builder, since it is the primary design entry tool. Software related modifications are made using Propel SDK.

9.1. Changing LPDDR4 Memory Controller Parameters

The LPDDR4 memory controller IP parameters are set according to the DRAM chips available on the CertusPro-NX Versa Board and in the Lattice Avant-AT-E Evaluation Board. You can change the IP parameters to suit the board and DRAM chip in your project.

To modify the IP parameters:

1. In Propel Builder, double-click on the **lpddr4_inst** block.

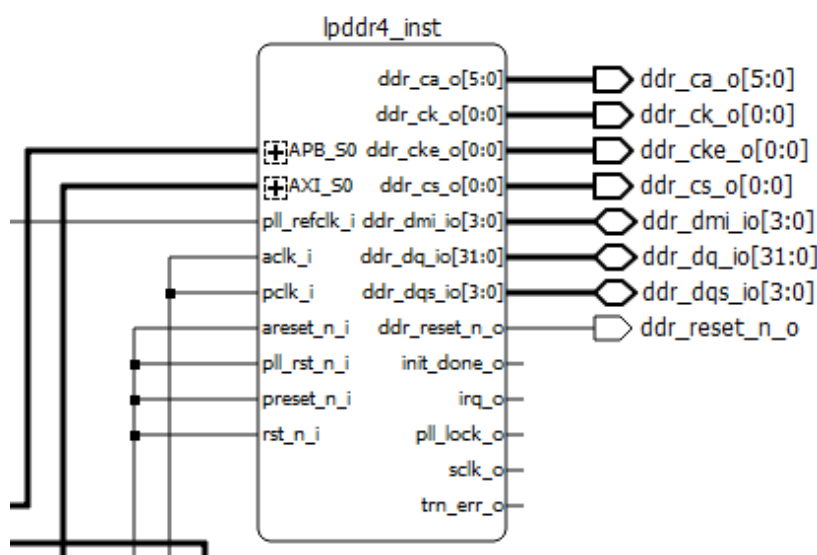


Figure 9.1. lpddr4_inst Block

2. In the **Module/IP Block Wizard**, make changes to the parameters that suit your board and DRAM chip. Table 9.1 shows the possible customization.

Table 9.1. LPDDR4 Memory Controller IP Parameters Customization

| Parameter | Description |
|-----------------------------|--|
| DDR Command Frequency (MHz) | Change the frequency to the desired value that match your design. For Lattice Avant devices, the maximum is 800 MHz while for CertusPro-NX devices, it is 533 MHz. |
| DDR Density | Change the density (in terms of Gb) per channel to match with the DRAM chip. This affects the AXI4 address bus width of the IP interface. |
| DDR Bus Width | Change the bus width for DDR data bus to match with the DRAM chip. This affects the DQ, DQS and DMI bus width. It requires update to FPGA pin assignments. |

3. In the **Module/IP Block Wizard**, click the **Generate** button to make sure RTL is updated per customization.

Note: Regenerate the Propel Builder system and in Lattice Radiant click **Export Files** to update the bitstream.

9.2. Adding New Component to the Propel Builder System

This reference design can be used as base design to add a new component or IP that your project requires. You can perform this in Propel Builder using the Schematic view.

The following procedure shows the design flow in general:

9.2.1. Hardware Flow

To add a new IP:

1. In Propel Builder, select and add new IP from the **IP Catalog** window. Complete the IP configuration using the wizard and generate.
Note: If you need to create a custom IP, refer to [Lattice IP Packager 2023.1 \(FPGA-UG-02187\)](#).
2. Connect the new IP to the RISC-V CPU or other IPs in the system. There are three primary interface types:
 - AXI4 or AXI-lite – Add new Manager/Subordinate interface in `axi_interconnect_0_inst` (depending on the interface type on the new IP). Connect the new IP to the newly added interface on the interconnect.
 - APB – Add new Requestor/Completer interface in `apb_interconnect0_inst`. Connect the new IP to the newly added interface on the interconnect.
 - AHB-Lite – Add new Manager/Subordinate interface in `axi_interconnect_0_inst`. Since the new IP has AHB-lite interface, you need to add AXI4 to AHB-Lite Bridge IP in between.
3. Connect the clock and reset signals for the new IP.
4. Export the I/O from new IP to top level module (if applicable).
5. In Propel Builder, go to **Address** view to assign the base address for the new IP.
6. Upon completion, click **Generate** in Propel Builder.
7. In Lattice Radiant, assign pins for the new IP (if applicable).
8. Click **Export Files** to generate the updated bitstream.

9.2.2. Software Flow

If the new IP contains software driver, update the Board Support Package (BSP) in the software project.

To update the BSP:

1. In Propel SDK, right-click on the software project that you are working on and select **Update Lattice C/C++ Project**.
In the **Update System and BSP** dialog box, you may observe a **Directory is not correct** error as shown in Figure 9.2. This is because the software project is created in another PC with a different system environment path when the project is imported to SDK.

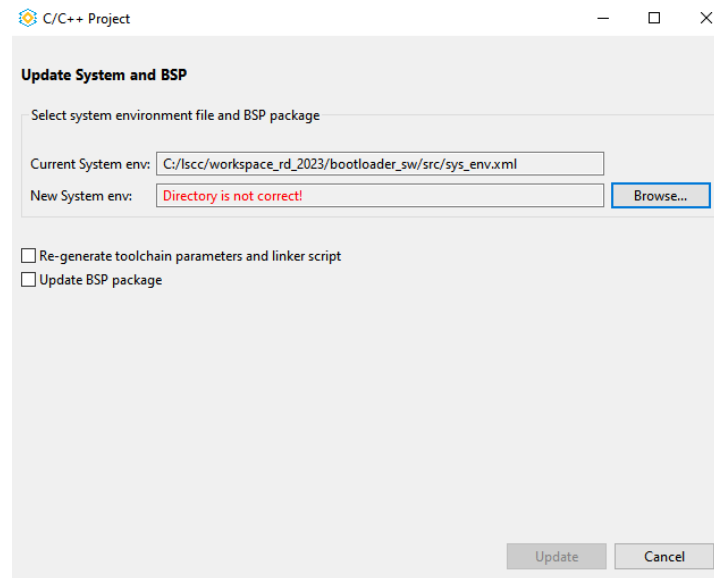


Figure 9.2. New System Error

2. Click the **Browse** button to navigate to the new system env path in your PC.
For example: `<my_work_dir>/sge/sys_env.xml`
Note: This step updates the software project to point to the Propel Builder system env file located in your PC. The correct path is displayed instead of the previous error.
3. Select the **Update BSP package** box. This shows the new IP driver and version available for update.
Note: DO NOT select **Re-generate toolchain parameters and linker script** box. The software projects provided in this reference design contains modified linker script. Selecting this option overwrites the modifications.
4. Click **Update** to complete the process.
5. Build the software project to obtain updated executable files (.elf, .mem and .bin).

9.3. Changing the Application Image Load Address

The bootloader copies the application image to LPDDR4 memory at a pre-defined address. The default address is set to 0x4000_0000, which is the beginning of the memory. If you need to change the load address, modify the bootloader source code.

To change the application image load address:

1. In Propel SDK Project Explorer, open **bootloader_sw > src > main.c**.

2. Edit the following line in *main.c* to the new load address.

```
#define APPLICATION_MEMORY_START_ADDR 0x40000000
```

Note: The load address must be 16-byte aligned. Examples of valid addresses are: 0x40000010, 0x40000020, and others.

3. Rebuild the **bootloader_sw** project.

4. The application software must be updated to match the new load address.

- a. Open **freertos_app_sw > src > linker.ld**.

- b. Edit the following line in *linker.ld* to the new load address. For example, if the new load address is 0x40000010, the **OFFSET_SIZE** should be 0x10.

```
OFFSET_SIZE = 0x10;
```

5. Right-click **freertos_app_sw** and select **Clean Project**.

6. Right-click and select **Build Project**.

7. Repeat the procedure in the [Generating the Flash Image](#) section to generate the new flash image.

8. Repeat the procedure in the [Programming the Application Image to SPI Flash](#) section to program image to flash.

9.4. Updating Linker Script to LPDDR4 Memory Address

By default, the software application created is not set to execute from the LPDDR4 memory. Modify the linker script so that you can execute your software project from the LPDDR4 external memory.

To modify the linker script:

1. In the Propel SDK Project Explorer, open the linker script (*linker.ld*) for your software project.

2. In *linker.ld*, locate the **MEMORY** region. The following example shows the **MEMORY** region set to the System Memory (*system0_inst*) starts from address 0x0 and has the length of 32 Kbytes.

```
MEMORY
{
    system0_inst (rwx) : org = 0x0, len = 0x8000
}
```

3. Modify the **MEMORY** region to set to LPDDR4 memory (*ddr*) as follows, where *ddr* starts from address 0x40000000 and has the length of 1 GBytes.

```
MEMORY
{
    ddr (rwx) : org = 0x40000000, len = 0x40000000
}
```

4. Replace *system0_inst* with *ddr* for each sections, including *.text*, *.ctors*, *.dtors*, *.rodata*, *.data*, *.bss*, *.heap*, and *.stack*.

5. Save the *linker.ld* file.

6. Right-click the project and select **Clean Project**.

7. Right-click and select **Build Project**.

10. Debugging the Design

This section lists possible issues and provides troubleshooting procedures.

10.1. SPI Flash Programming Fail

Check the following if the SPI flash programming fails when performing the procedure in the [Programming the Application Image to SPI Flash](#) section.

10.1.1. Check Flash Device for CertusPro-NX Devices

To check flash for CertusPro-NX devices:

1. In Radiant Programmer, double-click on **Erase, Program, Verify** under the **Operation** column. This opens the **Device Properties** dialog box.
2. In the **General** tab, select the options as shown in [Figure 10.1](#) (if using Macronix flash device) or [Figure 10.2](#) (if using Winbond flash device).
 - Target Memory – SPI Flash
 - Operation –Erase, Program, Verify
 - **Programming file** – <my_work_dir>\prebuilt_images\freertos_app_boot.bin
 - Vendor – Macronix or Winbond
 - **Device** – **MX25L51245G** for Macronix flash device or **W25Q512JV** for Winbond flash device
3. Click **Load from File** to make sure the data file size is correct.

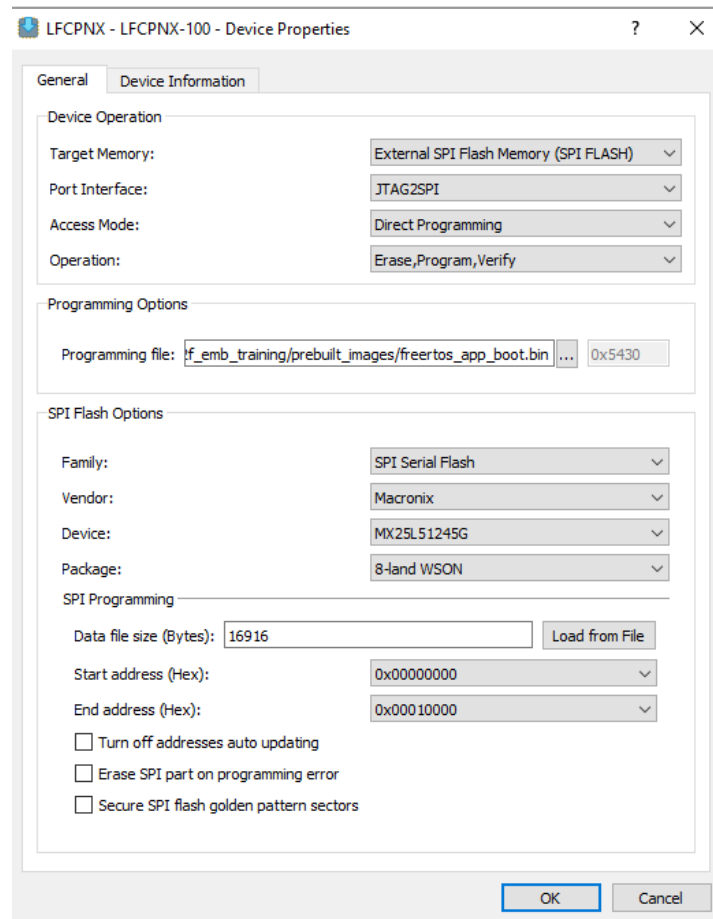


Figure 10.1. Flash Device Properties for Macronix

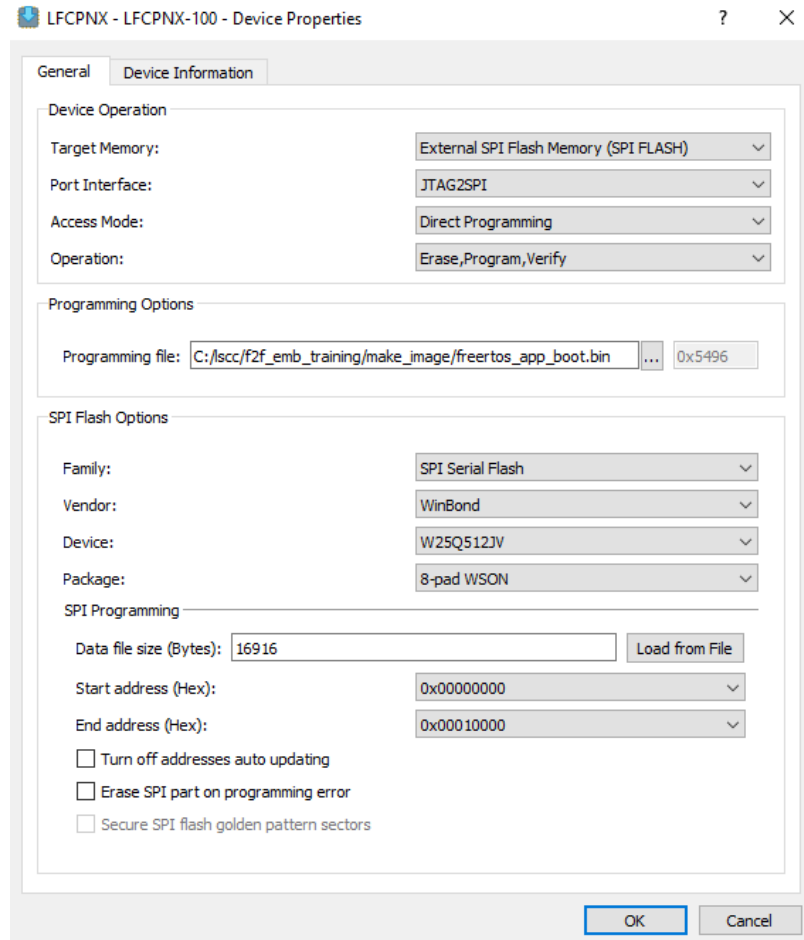


Figure 10.2. Flash Device Properties for Winbond

10.1.2. Check Correct Flash Device for Lattice Avant Devices

To check flash for Lattice Avant devices:

1. In Radiant Programmer, make sure the settings match [Figure 10.3](#) settings.

| Enable | Status | Device Vendor | Device Family | Device |
|-------------------------------------|--------|---------------|------------------|-------------|
| <input checked="" type="checkbox"/> | PASS | Macronix | SPI Serial Flash | MX25L51245G |

Figure 10.3. Lattice Avant Device Flash Programmer Project Settings

2. Double-click on **Erase, Program, Verify** under the **Operation** column. This opens the **Device Properties** dialog box.
3. In the **General** tab, make sure **Programming file** points to `<my_work_dir>\prebuilt_images\freertos_app_boot.bin`.
4. Click **Load from File** to make sure the data file size is correct.

10.1.3. Check TCK Divider Setting

To check TCK Divider setting:

1. If you see the following error, select **Use Custom Clock Divider** under **Programming Speed Settings**.
2. Increase the **TCK Divider Setting** value to **3** or higher.

```

Output
INFO <85021294> - Device1 LFCPNX-100: W25Q512JV: Erase,Program,Verify
Initializing...
IDCode Checking...
ERROR <85021012> - Verification Error...when Processing function: 'CHECK_ID'
Failed to read the device's ID.
Please set the custom TCK Divider Setting value to 3 or more and try again...
ERROR <85021372> - Operation: unsuccessful.
  
```

Figure 10.4. TCK Divider Error

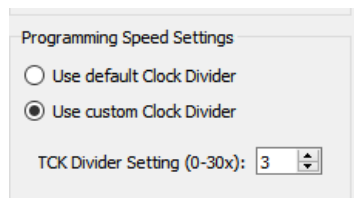


Figure 10.5. Programming Speed Settings

10.1.4. Check Cable Settings

Flash programming failure may occur if the incorrect programming cable is selected.

To check cable setting:

1. Click **Detect Cable** in the **Cable Settings** panel located at top right of the Radiant Programmer main interface.
Note: If you connect multiple cables to your PC, **Detect Cable** lists all the connected cables. Select the correct cable that connects to the board that you are working on. Alternatively, connect only one cable from the PC to the board to avoid detection of multiple cables.
2. For the Lattice Avant board, make sure to connect the HW-USBN-2B cable properly to the board. Refer to the [Setting up the Lattice Avant-AT-E Evaluation Board](#) section.

10.2. UART Serial Terminal Prints Incorrect Characters

Incorrect (junk) characters may be displayed on the serial terminal.

To resolve the display of incorrect characters:

1. Make sure the **Baud Rate** or **Speed** in the terminal software is set to **115200**.
2. If you modify the PLL output clock, update the system clock frequency value in the Propel Builder **RISC-V RX** wizard.
Double-click **RISC-V RX IP** and set the correct **System Clock Frequency** value under **Local UART**.

| Local UART | |
|--|-------------------------------------|
| Enable UART instance | <input checked="" type="checkbox"/> |
| System Clock Frequency (MHz) [2 - 200] | 50 |
| Serial Data Width | 8 |
| Stop Bits | 1 |
| Parity Enable | <input type="checkbox"/> |
| UART Standard Baud Rate | 115200 |

Figure 10.6. UART Settings

11. Resource Utilization

Table 11.1 shows the resource utilization for CertusPro-NX devices and Table 11.2 for Lattice Avant devices.

Note: The values are only for reference and actual usage may vary.

Table 11.1. Resource Utilization for CertusPro-NX Devices

| Configuration | LUTs (Utilization/Total) | Registers (Utilization/Total) | EBRs (Utilization/Total) | I/O (Utilization/Total) |
|--|-----------------------------|----------------------------------|-----------------------------|----------------------------|
| RISC-V LPDDR4 Reference Design targeting CertusPro-NX devices | 36582/79872 | 20132/80769 | 68/208 | 80/299 |

Table 11.2. Resource Utilization for Lattice Avant Devices

| Configuration | LUTs (Utilization/Total) | Registers (Utilization/Total) | EBRs (Utilization/Total) | I/O (Utilization/Total) |
|---|-----------------------------|----------------------------------|-----------------------------|----------------------------|
| RISC-V LPDDR4 Reference Design targeting Lattice Avant devices | 28049/397440 | 16167/399141 | 45/990 | 84/567 |

References

- [RISC-V RX CPU IP - Lattice Propel Builder 2023.1 \(FPGA-IPUG-02230\)](#)
- [LPDDR4 Memory Controller for Nexus Devices \(FPGA-IPUG-02127\)](#)
- [Avant Memory Controller IP Core - Lattice Radiant Software \(FPGA-IPUG-02208\)](#)
- [AXI4 Interconnect Module - Lattice Propel Builder \(FPGA-IPUG-02196\)](#)
- [SPI Flash Memory Controller IP Core - Lattice Radiant Software \(FPGA-IPUG-02134\)](#)
- [Lattice Propel 2023.1 Builder User Guide \(FPGA-UG-02185\)](#)
- [Lattice Propel 2023.1 SDK User Guide \(FPGA-UG-02186\)](#)
- [CertusPro-NX Family Data Sheet \(FPGA-DS-02086\)](#)
- [Lattice Avant Platform - Overview Data Sheet \(FPGA-DS-02107\)](#)
- [Lattice Avant Platform - Specifications Data Sheet \(FPGA-DS-02112\)](#)
- [CertusPro-NX Evaluation Board User Guide \(FPGA-EB-02046\)](#)
- [Avant Evaluation Board - User Guide \(FPGA-EB-02057\)](#)
- [CertusPro-NX web page](#)
- [Avant-E web page](#)
- [Lattice Radiant design software web page](#)
- [Lattice Propel design software web page](#)
- [RISC-V RX and LPDDR4 Memory Controller web page](#)
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, November 2023

| Section | Change Summary |
|---------|------------------|
| All | Initial release. |



www.latticesemi.com