RISC-V RX and LPDDR4 Memory Controller

Reference Design

FPGA-RD-02278-1.1

August 2025

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 REDUDANCIES, 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

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	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
IP	Intellectual Property (core)
HDL	Hardware Description Language
JTAG	Joint Test Action Group (IEEE 1149.1)
PLL	Phase-Locked Loop
SDRAM	Synchronous Dynamic Randon-Access Memory
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
GPIO	General Purpose Input/Output
TCK	Test Clock (JTAG)
CRC	Cyclic Redundancy Check
SDK	Software Development Kit
TCL	Tool Command Language
EBR	Embedded Block RAM
DDR	Double Data Rate (memory)
LVSTL_I / LVSTL11_I	Low Voltage Sub Series Terminated Logic (I/O standard)
RX	Receive (or RISC-V RX core)
TXD / RXD	Transmit Data / Receive Data
MSB	Most Significant Bit
TDM	Time Division Multiplexing (implied in memory access context)
R/W	Read/Write
TCM	Tightly Coupled Memory

1. Introduction

The RISC-V RX CPU and LPDDR4 Memory Controller Reference Design demonstrates how to use the RISC-V RX soft IP and LPDDR4 memory controller in Lattice Avant™ and CertusPro™-NX FPGA devices.

In this design, the CPU connects to the LPDDR4 memory controller IP to interface with external LPDDR4 SDRAM. The design shows how the CPU executes software stored in external SDRAM, which is useful for applications that require more memory than the FPGA's on-chip memory can provide. It includes a sample bootloader program that loads software from an SPI Flash device into the LPDDR4 SDRAM. The RISC-V RX CPU accesses all system peripherals through the AXI Interconnect and bridges.

The Lattice Semiconductor RISC-V RX CPU integrates a 32-bit RISC-V CPU core and several submodules, including the Platform-Level Interrupt Controller (PLIC), Core Local Interrupter (CLINT), and Watchdog. The latest RISC-V RX supports RV32IMACF instruction set and includes a JTAG-compliant debug feature (IEEE 1149.1). The CPU core accesses external modules through either AXI or Local Bus Interface.

The Lattice Semiconductor LPDDR4 Memory Controller IP Core offers a complete solution for interfacing with LPDDR4 SDRAM. Lattice provides a turnkey package that includes the controller, DDR PHY, and the necessary clocking and training logic. The LPDDR4 Memory Controller IP Core simplifies integration with user application designs by supporting the AXI4 interface which reduces the need to manage LPDDR4 SDRAM signals directly.

1.1. Quick Facts

You can download the reference design files for the RISC-V RX and LPDDR4 Memory Controller from the Lattice Semiconductor website.

Table 1.1. Reference Design Summary

General	Target device	CertusPro-NX devices Lattice Avant-E devices
	Source code format	Verilog, C
Simulation	Functional simulation	Not supported
	Timing simulation	Not performed
	Test bench	Not available
Software Requirements	Software tool and version	Lattice Propel™ Design Environment 2025.1 Lattice Radiant™ software version 2025.1 Python 3.11.x or later
	IP version	RISC-V RX v2.6.0 LPDDR4 Memory Controller for Nexus Devices v2.6.0 DDR Memory Controller v2.6.0 (For Avant devices) SPI Flash Memory Controller v2.0.0 Tightly Coupled Memory v1.5.2 AXI4 Interconnect v2.2.0 AXI4 to APB Bridge v1.4.0 AXI4 to AHB-Lite Bridge v1.4.0 APB Interconnect v1.3.0 GPIO v1.8.0
Hardware Requirements	Board	CertusPro-NX Versa Board Lattice Avant-E Evaluation Board
	Cable	USB-A to Mini-B cable

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 2025.1 (FPGA-IPUG-02280)
- LPDDR4 Memory Controller Interface IP Core for CertusPro-NX device features described in LPDDR4 Memory Controller IP Core for Nexus Devices (FPGA-IPUG-02127)
- LPDDR4 Memory Controller Interface IP Core for Lattice Avant device features described in DDR Memory Controller IP Core (FPGA-IPUG-02208)
- AXI4 Interconnect features described in AXI4 Interconnect IP Core User Guide (FPGA-IPUG-02196)
- SPI Flash memory controller features described in SPI Flash Memory Controller IP Core User Guide (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.

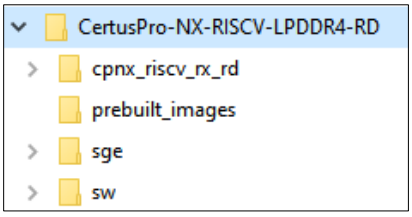

Figure 2.1. Directory Structure

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

Note: In the reference design targeting the Lattice Avant device, folder and file name uses *avant* in place of *cpnx*.

Table 2.1. Directory List

Directory/File	Description
cpnx_riscv_rx_rd	Contains the Lattice Propel Builder project and generated HDL files for IP.
prebuilt_images	Contains the bitstream file, software image file, Lattice Radiant Programmer scripts, and miscellaneous files
sge	Contains the software handoff file from the Lattice Propel Builder project.
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 CPU system built using Lattice Propel Builder. All system components are available within Lattice Propel Builder, where they are instantiated, connected and used to generate the corresponding RTL design files.

Figure 3.1 illustrates the top-level block diagram and interconnections of the reference design.

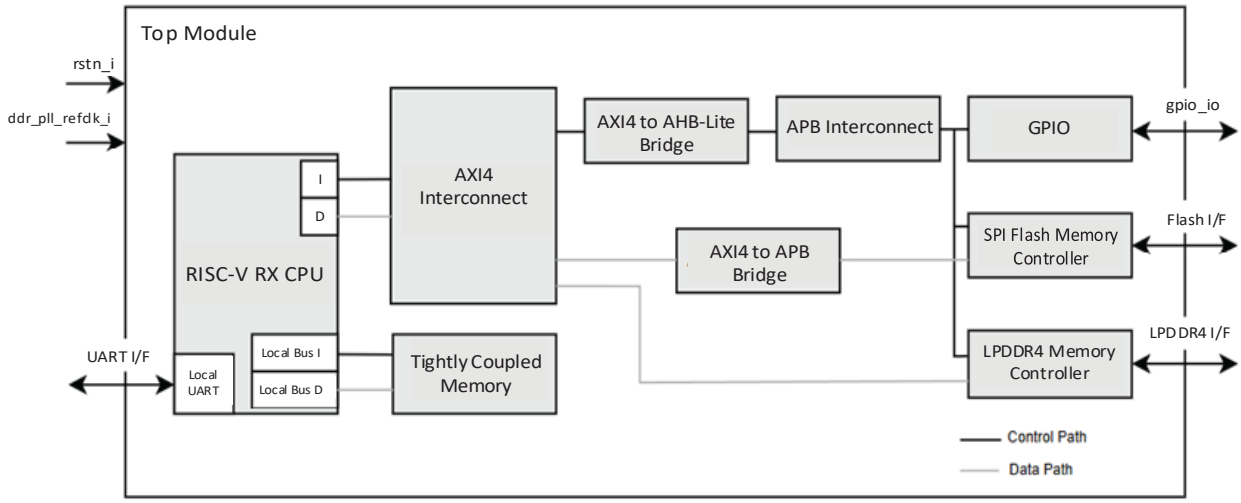

Figure 3.1. Reference Design Top Level Block Diagram

The design includes the following components:

- RISC-V RX CPU with local UART
- LPDDR4 Memory Controller
- SPI Flash Controller
- Tightly Coupled Memory
- AXI4 Interconnect
- AXI4 to APB ridge
- AXI4 to AHB-Lite Bridge
- APB Interconnect
- General Purpose I/O (GPIO)

Each component in the block diagram is instantiated using IP available in Lattice Propel Builder. The features and parameters of each IP are detailed in the IP Configuration and Parameter Description section.

Figure 3.1 shows the multiple interfaces present at the top level of the design. The Signal Description section provides details on the signals associated with each interface.

3.2. Clocking Scheme

This section describes the clocking scheme used in the reference design. While the overall structure remains similar, there are minor differences between the clocking scheme for Certus Pro-NX and Lattice Avant devices. For more details 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 illustrates the clocking scheme of the reference design when targeting the CertusPro-NX device. Table 3.1 lists the details of each clock signal used in the design.

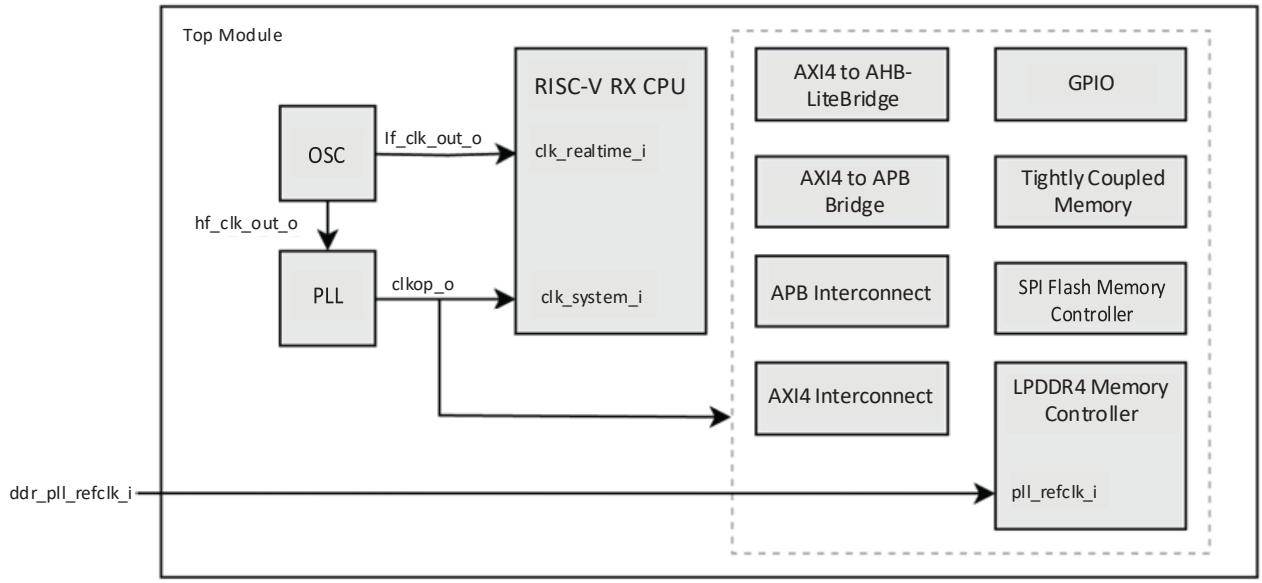

Figure 3.2. Clocking Block Diagram – CertusPro-NX Devices

Table 3.1. Clock Description – 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 input to RISC-V RX CPU
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 drives the high-speed clock input of the RISC-V RX CPU and all other 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:

1. The PLL clkop output frequency is adjustable and can be set higher than 50 MHz to increase the clock frequency of the RISC-V RX CPU and the overall system. However, the current reference design achieves timing closure at 50 MHz, so the PLL clkop is configured to operate at that frequency.

3.2.2. Clocking Scheme Overview - Lattice Avant Devices

Figure 3.2 illustrates the clocking scheme of the reference design when targeting the Lattice Avant device. Table 3.1 provides detailed information about each clock signal used in the design.

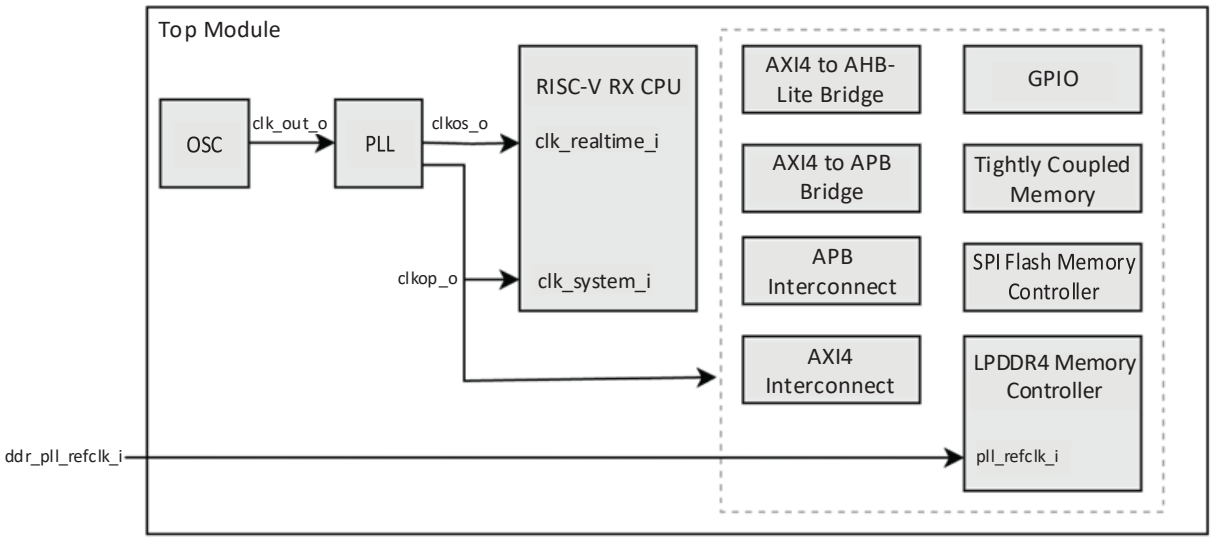

Figure 3.3. Clocking Block Diagram - Lattice Avant Devices

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	100 MHz[1]	PLL's primary clock output for RISC-V RX high-speed input and all system components
clkos	PLL	RISC-V RX CPU	16.384 MHz[2]	PLL's 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

Notes:

1. The PLL clkop output frequency is adjustable and can be set higher than 100 MHz to increase the clock frequency of the RISC-V RX and the overall system. However, the current reference design achieves timing closure at 100 MHz, so the PLL clkop is configured to operate at that frequency.
2. For Lattice Avant family devices, the 32 kHz output cannot be directly obtained from the OSC and PLL. Therefore, a 512-clock divider is implemented for the real-time clock port of the RX core, clk_realtime_i. It is recommended to use a 16.384 MHz clock signal as the input to this port, which will then generate a 32 kHz clock signal for the mtimecmp and mtime registers. For more detailed information, refer to RISC-V RX CPU IP-Lattice Propel Builder 2025.1 (FPGA-IPUG-02280).

3.3. Reset Scheme

This section describes the reset scheme used in the reference design for both CertusPro-NX and Lattice Avant target devices. Figure 3.4 illustrates the reset block diagram, while Table 3.3 provides detailed information about each reset signal.

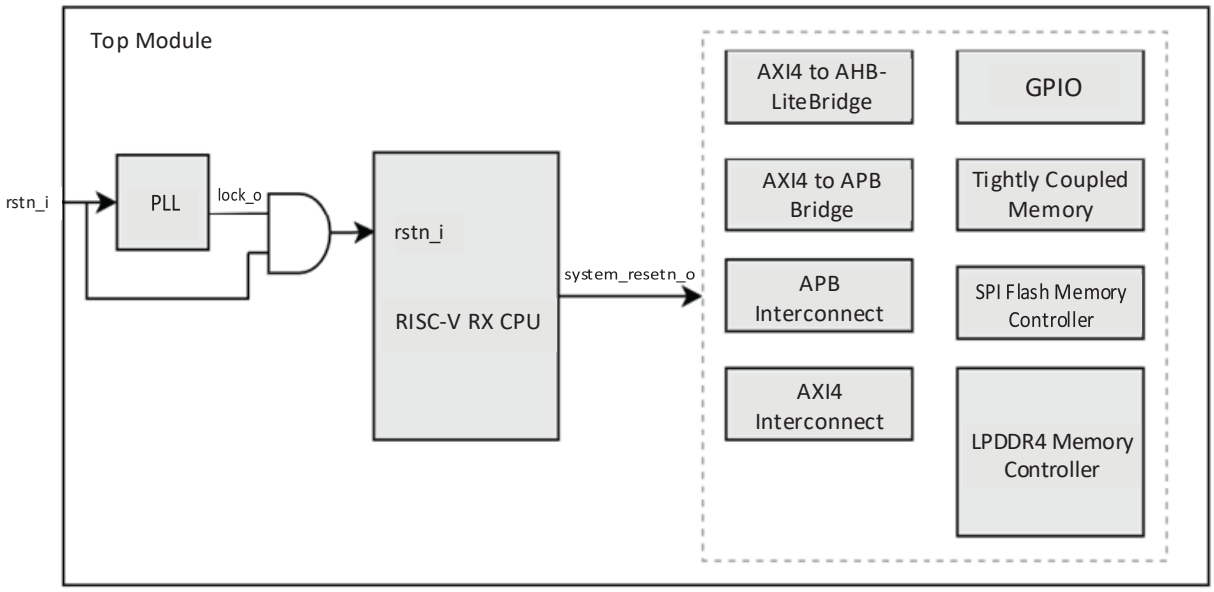

Figure 3.4. Reset Scheme Block Diagram

Table 3.3. Reset Signal Description

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

4. IP Configuration and Parameter Description

The reference design is developed using Lattice Propel Builder. Lattice Propel Builder generates the top-level HDL file, which serves as the top module of the design. Design parameterization is performed by configuring the IP blocks within Lattice Propel Builder. This section describes the following IPs and their configurable parameters. The screenshots provided are for reference only. The details may vary depending on the version of the IP or software being used.

- RISC-V RX CPU
- LPDDR4 Memory Controller for Nexus Devices
- DDR Memory Controller (for Avant Devices)
- SPI Flash Memory Controller
- Tightly Coupled Memory
- AXI4 Interconnect
- AXI4 to APB Bridge
- AXI4 to AHB-Lite Bridge
- APB Interconnect
- GPIO

4.1. RISC-V RX CPU

The RISC-V RX CPU IP features AXI-based instruction and data ports. The instruction port connects to memory that stores the software code executed by the CPU, while the data port connects to memory and peripherals for control operations. The design enables the local UART feature to support communication over a serial protocol. Figure 4.1 shows the configuration of the RISC-V RX CPU IP.

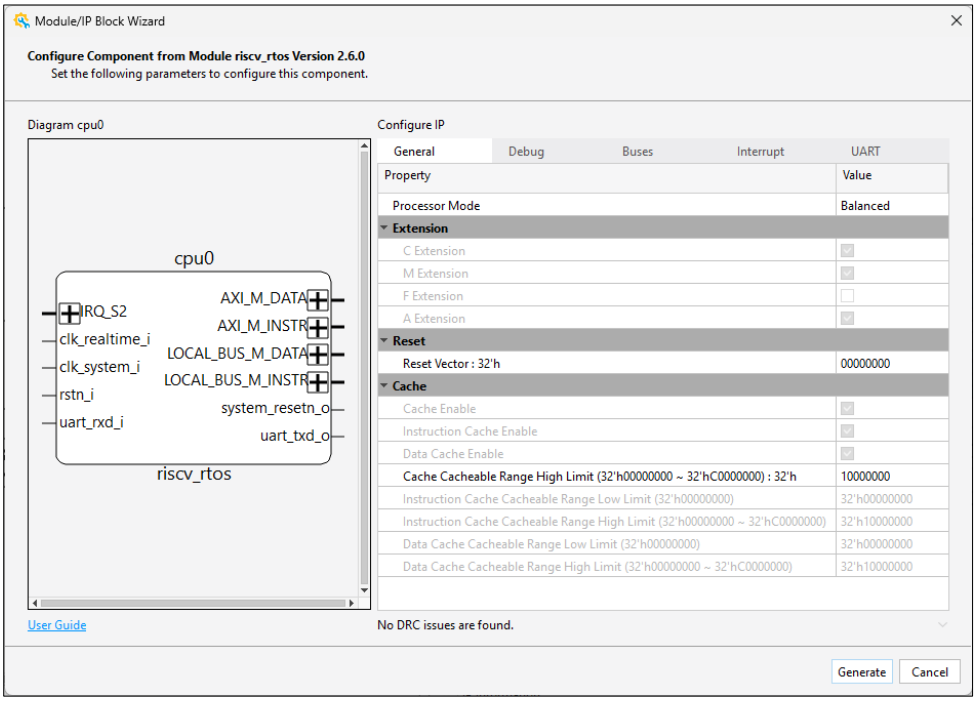

Figure 4.1. RISC-V RX CPU IP Configuration

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

Table 4.1. RISC-V RX IP Configuration

Property	Value	Description
General Tab		
CPU Mode	Balanced	Specifies the CPU mode. The default value is used to balance performance and resource utilization.
Reset Vector	00000000	Reset vector initial value.
Cache Cacheable Range High Limit	10000000	Specifies the higher limit of the cache's cacheable range.
Debug Tab		
Enable Debug	True	Enable the JTAG interface for CPU debug capability.
Enable Debug On Off Control Port	False	Enables the presence of the Debug On Off Control port on the generated IP. The port is disabled in this design.
JTAG Type	Hard	Specifies the JTAG Type.
Extend JTAG Channel	False	Enables the JTAG channel's range extension.
JTAG Channel	14	Specifies the channel of RX JTAG block which is used by the debugger.
Bus Tab		
Enable Local Bus	True	Enables the presence of the local bus on the generated IP. In this design, tightly coupled memory is connected to the local bus port.
Enable AXI Instruction Ports	True	Enables the presence of AXI instructions ports on the generated IP.
AXI Instruction Interface Register Slice Type	0	Type of AXI instruction ports channel register slice. The default is used in this design, which is bypass register slice.
AXI Data Interface Register Slice Type	0	Type of AXI data ports channel register slice. The default is used in this design, which bypasses register slice.
AXI ID Width	4	Specifies the AXI ID signal width.
AXI Data Ports ID Number	0	Specifies the value of RX AXI data ports ID signals.
AXI Instruction Ports ID Number	0	Specifies the value of RX AXI instructions ports ID signals.
Enable CXU Ports	False	CXU ports are disabled in this design.
Enable RVFI	False	RVFI interface is disabled in this design.
Interrupt Tab		
Enable CLINT	True	Enable CLINT is selectable when RX is not in Fmax mode. Enable CLINT is fixed as disabled when RX is in Fmax mode.
Enable PLIC	True	Enable PLIC is selectable when RX is not in Fmax mode. Enable PLIC is fixed as disabled when RX is in Fmax mode.
Enable Non-maskable Interrupt	False	Non-maskable interrupt is disabled in this design.
Enable Interrupt for Supervisor Mode	False	Interrupt for supervisor mode is disabled; all external interrupts go to the Machine mode only.
Width of Interrupt Priority Register	3	Specifies the data width of PLIC priority register. The default is 3-bit.
Number of User Interrupt Request	1	Specifies the supported number of interrupts for peripherals. There is only interrupt from GPIO for this design.
Enable CDC Register for IRQ_S2	False	The ports for 2-stage synchronizer on enabled IRQ_S interface are disabled in this design.
UART Tab		
Enable UART Instance	True	Enable local UART within the RISC-V RX CPU. There is no need to add extra UART IP to the system.
System Clock Frequency [2 – 200 MHz]	100 MHz	Specify the clock frequency for RISC-V RX CPU 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

Property	Value	Description
Enable Parity	False	Parity is not enabled
UART Standard Baud Rate	115200	Baud rate of 115,200. Needs matching with terminal setting.
Enable UART SIM	False	UART SIM is disabled in this design. The RX core does not drive uart_txd_o to output the character signal. Strings are not printed inside simulation tool transcript.

4.2. LPDDR4 Memory Controller

The LPDDR4 Memory Controller IP enables access to external LPDDR4 memory modules, which can store both CPU software code and data. The memory controller IP differs between 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 devices.

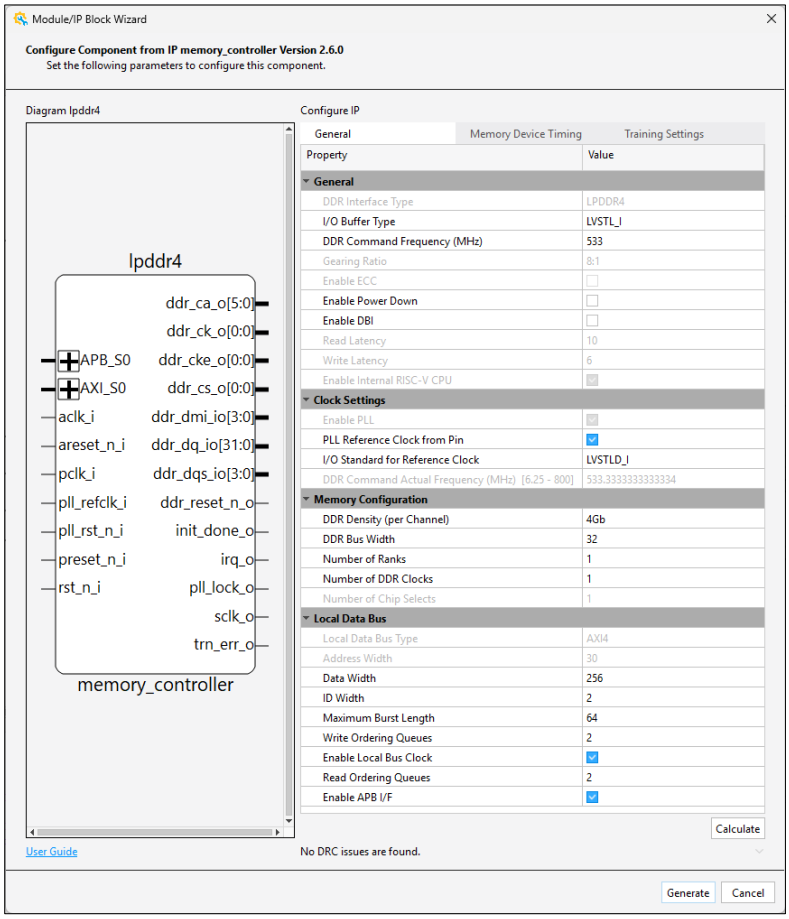

Figure 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices

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
General Tab		
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)	4 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 the development kit.
Number of Ranks	1	Configure the number of ranks based on the memory module on the development kit.
Number of DDR Clocks	1	Configure the number of CK/CK# clock pairs to be driven to DDR DRAM based on the memory module on development kit. There is only 1 DDR clock channel available for CertusPro-NX Versa Board.
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 section for details.
Write Ordering Queues	2	Set the ordering queues that process the write and read data requests. More queues mean 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.
Read Ordering Queues	2	Set the ordering queues that process the write and read data requests. More queues mean 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 APB I/F	True	Enable APB interface, which allows the CPU to access the memory controller registers.

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

4.2.2. DDR Memory Controller

Figure 4.3 and Figure 4.4 show the DDR4 Memory Controller IP configuration for Lattice Avant devices.

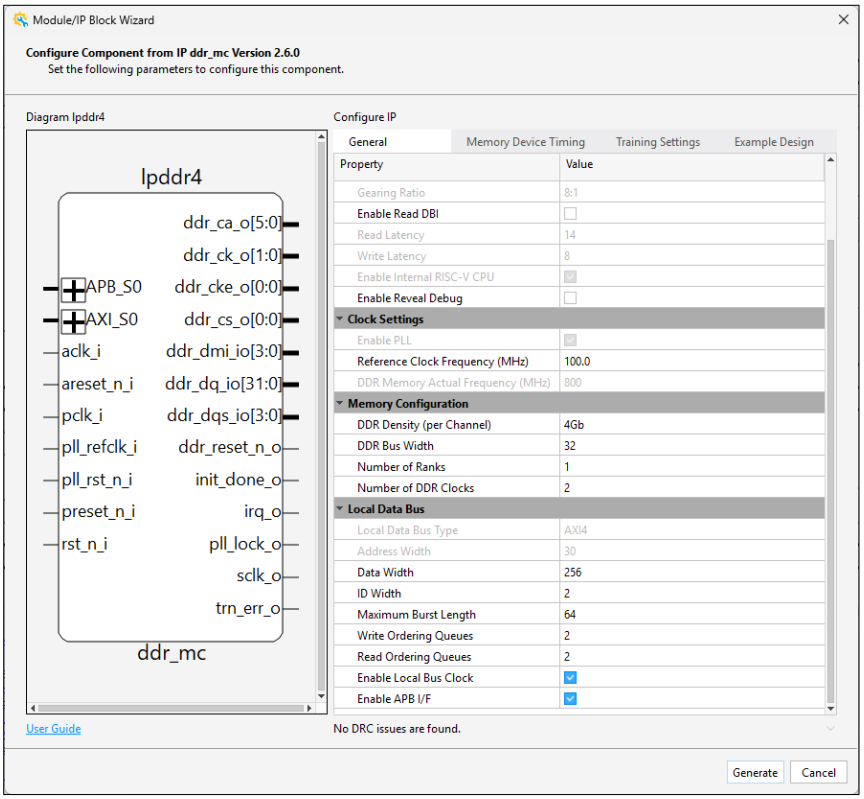

Figure 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices – General Tab

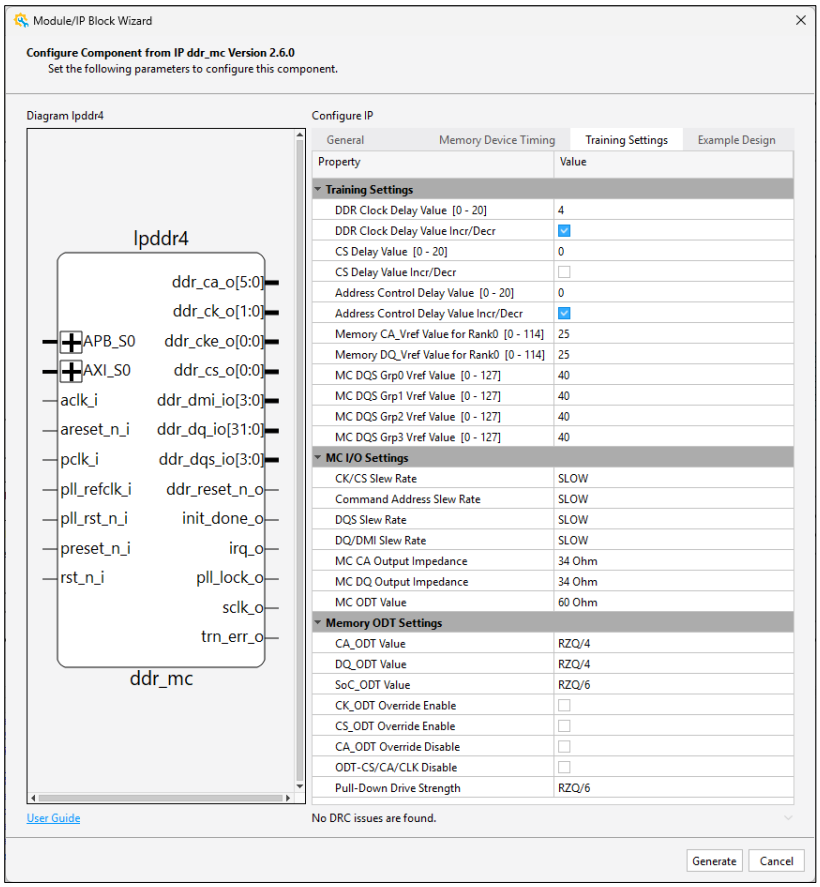

Figure 4.4. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices – Training Settings

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
General Tab		
Interface Type	LPDDR4	Configure the interface type based on the memory module used in the development kit.
I/O Buffer Type	LVSTL11_I	Set the I/O buffer to LVSTL_II standard.
DDR Command Frequency (MHz)	800	Maximum supported frequency for Lattice Avant devices.
Enable DBI	False	Not used in this Reference design.
Reference Clock Frequency (MHz)	100	PLL reference clock input connected from the top module. Refer to the Clocking Scheme Overview - Lattice Avant Devices section for details.
DDR Density (per Channel)	4 Gb	Configure based on the DDR DRAM density of the memory module on the development kit.
DDR Bus Width	32	Set according to the DDR DRAM bus width of the memory module on the development kit.
Number of Ranks	1	Configure based on the number of ranks in the memory module on the development kit.
Number of DDR Clocks	2	Set the number of CK/CK# clock pairs to be driven to DDR DRAM based on the memory module on the development kit. Lattice Avant-E boards support two DDR clock channels.

Property	Value	Description
Data Width	256	Specifies the data width on the AXI interface of the memory controller.
ID Width	2	Defines the ID width on the AXI interface of the memory controller. Must match the AXI Manager in the Interconnect IP. Refer to the AXI4 Interconnect section for details.
Maximum Burst Length	64	Sets the maximum AXI4 burst length. Recommended to adjust based on the system requirements to optimize EBR resource usage. The default value is used to balance performance and resource utilization.
Write Ordering Queues	2	Set the ordering queues that process the write and read data requests. More queues mean 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.
Read Ordering Queues	2	Set the ordering queues that process the write and read data requests. More queues mean 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	Enables the clock domain crossing logic for the local data bus.
Enable APB I/F	True	Enables the APB interface, allowing CPU to access memory controller registers.
Training Settings Tab		
DDR Clock Delay Value	4	Specifies the DDR clock delay so that the first DQS toggle is at CK = 0 during write leveling. It is recommended to increase this value if write leveling fails
DDR Clock Delay Value Incr/Decr	True	Specifies the adjustment direction Checked: DDR Clock Delay Value is added to the ddr_ck_o signal Unchecked: DDR Clock Delay Value is subtracted from the ddr_ck_o signal
CS Delay Value	0	Specifies the DDR CS delay value
CS Delay Value Incr/Decr	False	Specifies the adjustment direction Checked: CS Delay Value is added to the ddr_cs_o signal Unchecked: CS Delay Value is subtracted from the ddr_cs_o signal
Address Control Delay Value	0	Specifies the DDR address/control signals delay. It is recommended to increase this value if command bus training fails
Address Control Delay Value Incr/Decr	True	Specifies the adjustment direction Checked: Address Control Delay Value is added to the DDR address/control signals Unchecked: Address Control Delay Value is subtracted from the DDR address/control signals
Memory CA_Vref Value for Rank 0	25	Specifies the DRAM CA_Vref Value to be written to LPDDR4 DRAM's MR12 VR-CA and VREF(CA) for Rank0/Rank1. This should be set based on SI/PI simulation for the target board design. Please refer to LPDDR4 JEDEC Standard for allowed values in MR12
Memory DQ_Vref Value for Rank 0	25	Specifies the DRAM DQ_Vref Value to be written to LPDDR4 DRAM's MR14 VR(DQ) and VREF(DQ) for Rank0/Rank1. When the register field TRN_OP_REG.mem_vref_training_en=0, this will be used as the final DRAM DQ Vref. Please refer to LPDDR4 JEDEC Standard for allowed values in MR14.

Property	Value	Description
MC DQS Grp0 Vref Value	40	Specifies the MC DQ_Vref Value for each DQS group when DQ_Vref training is disabled by setting TRN_OP_REG.mc_vref_training_en=0. The theoretical mV value is: Vref decimal value × 5mV
MC DQS Grp1 Vref Value	40	Specifies the MC DQ_Vref Value for each DQS group when DQ_Vref training is disabled by setting TRN_OP_REG.mc_vref_training_en=0. The theoretical mV value is: Vref decimal value × 5mV
MC DQS Grp2 Vref Value	40	Specifies the MC DQ_Vref Value for each DQS group when DQ_Vref training is disabled by setting TRN_OP_REG.mc_vref_training_en=0. The theoretical mV value is: Vref decimal value × 5mV
MC DQS Grp3 Vref Value	40	Specifies the MC DQ_Vref Value for each DQS group when DQ_Vref training is disabled by setting TRN_OP_REG.mc_vref_training_en=0. The theoretical mV value is: Vref decimal value × 5mV
CK/CS Slew Rate	SLOW	Specifies the CK/CS driver strength
Command Address Slew Rate	SLOW	Specifies the CA driver strength
DQS Slew Rate	SLOW	Specifies the DQS driver strength
DQ/DMI Slew Rate	SLOW	Specifies the DQ/DMI driver strength
MC CA Output Impedance	34 Ω	Memory controller CA I/O impedance to reach VDDQ threshold
MC DQ Output Impedance	34 Ω	Memory controller DQ I/O impedance to reach VDDQ threshold
MC ODT Value	60 Ω	Memory controller termination resistance value at FPGA
CA_ODT Value	RZQ/4	Memory side CA ODT resistance value
DQ_ODT Value	RZQ/4	Memory side DQ ODT resistance value
SoC_ODT Value	RZQ/6	Memory controller ODT resistance value correlating to ZQ calibration
CK_ODT Override Enable	False	Overrides the default CK ODT value for the non-terminating rank when option is checked. Refer to MR22 in LPDDR4 device datasheet for more details
CS_ODT Override Enable	False	Overrides the default CS ODT value for the non-terminating rank when option is checked. Refer to MR22 in LPDDR4 device datasheet for more details
CA_ODT Override Enable	False	Disable termination for CA when option is checked. Refer to MR22 in LPDDR4 device datasheet for more details
ODT-CS/CA/CLK Disable	False	Disable ODT for CS/CA/CLK when option is checked
Pull-Down Drive Strength	RZQ/6	Memory side pull-down drive strength value

Notes:

- Default settings are used for all properties in Memory Device Timing.
- Refer to the Known Issues section for details about issues affecting Avant DDR Memory Controller IP version 2.6.0.

4.3. SPI Flash Memory Controller

Figure 4.5 shows the SPI Flash Memory Controller IP configuration

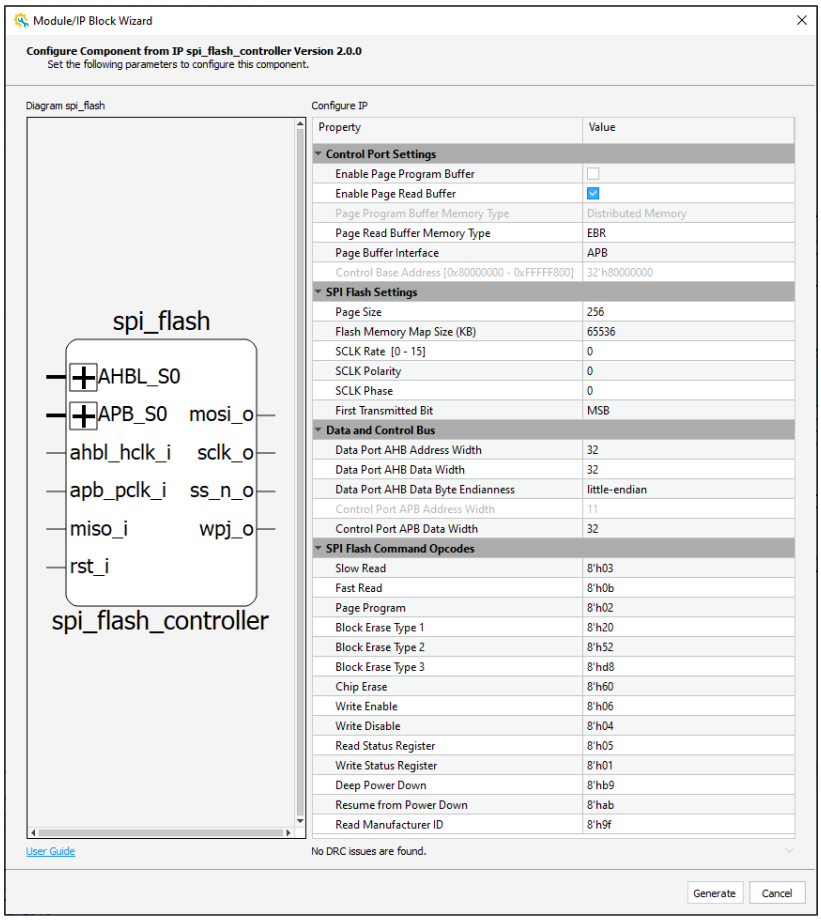

Figure 4.5. SPI Flash Memory Controller IP Configuration

Table 4.4 provides the configuration details of the SPI Flash Memory Controller IP.

Table 4.4. SPI Flash Memory Controller IP Configuration

Property	Value	Description
Enable Page Program Buffer	False	When enabled, page write data is locally stored before writing the entire page to SPI Flash. Disabled in this design.
Enable Page Read Buffer	True	Speeds up page reads by reading an entire page from SPI Flash and locally storing it. Buffer size equals the page size.
Page Read Buffer Memory Type	EBR	Specifies whether EBR or Distributed RAM is used for the Page Read Buffer.
Page Buffer Interface	APB	Indicates the interface used for data buffer configuration.
Page Size	256	Defines the size of each page in SPI Flash. Refer to the SPI Flash data sheet to obtain this value.
Flash Memory Map Size (KB)	65,536	The size of the memory mapped in KB. Available options must be a power of 2. Example: 1, 2, 4, 8 up to 2,097,152.

Property	Value	Description
SCLK Rate [0 – 15]	0	Specifies the factor for deriving sclk_o from the 0 – 15 component input clock (apb_pclk_i). sclk_o is derived from the following equation: sclk_o = apb_pclk_i/(2 x SCLK_Rate) when SCLK_RATE is greater than 0. If SCLK_RATE is equal to 0 sclk_o = apb_pclk_i. For example:For SCLK Rate = 1, sclk_o = apb_pclk_i/2 Note: This is configured in the bootloader software code.
SCLK Polarity	0	Sets the idle state polarity of the sclk_o. Default value is used in this design.
SCLK Phase	0	Determines whether data is sampled and shifted on the rising or falling edge of the clock. Default value is used in this design.
First Transmitted Bit	MSB	Specifies bit-shifting direction in the SPI Interface. Default value is used in this design.
Data Port AHB Address Width	32	Width of the AHB-Lite address bus.
Data Port AHB Data Width	32	Width of the AHB-Lite data bus.
Data Port AHB Data Byte Endianness	Little-endian	Endianness of AHB-Lite data bus. Default value is used in this design.
Control Port APB Data Width	32	Width of the APB data bus.
Slow Read	8'h03	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Fast Read	8'h0b	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Page Program	8'h02	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Block Erase Type 1	8'h20	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Block Erase Type 2	8'h52	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Block Erase Type 3	8'hd8	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Chip Erase	8'h60	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Write Enable	8'h06	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Write Disable	8'h04	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Read Status Register	8'h05	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Write Status Register	8'h01	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.

Property	Value	Description
Deep Power Down	8'hb9	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Resume from Power Down	8'hab	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.
Read Manufacturer ID	8'h9f	Specifies the equivalent opcode of a flash command of the off-chip SPI flash memory device. Default value is used in this design.

4.4. Tightly Coupled Memory

The Tightly Coupled Memory IP is used to store the initial software code executed by the RISC-V RX CPU immediately after resetting. Since the LPDDR4 external memory is not available until memory training is completed, this initial software ensures that the LPDDR4 memory is ready before any access by the RISC-V RX CPU.

By using the Tightly Coupled Memory IP, the initial software can be pre-compiled directly into the FPGA bitstream. Once device configuration is complete, the tightly coupled memory contains the valid initial software. Refer to the RISC-V RX Software Flow section for more details.

Figure 4.6 shows the configuration of the Tightly Coupled Memory IP in the General tab.

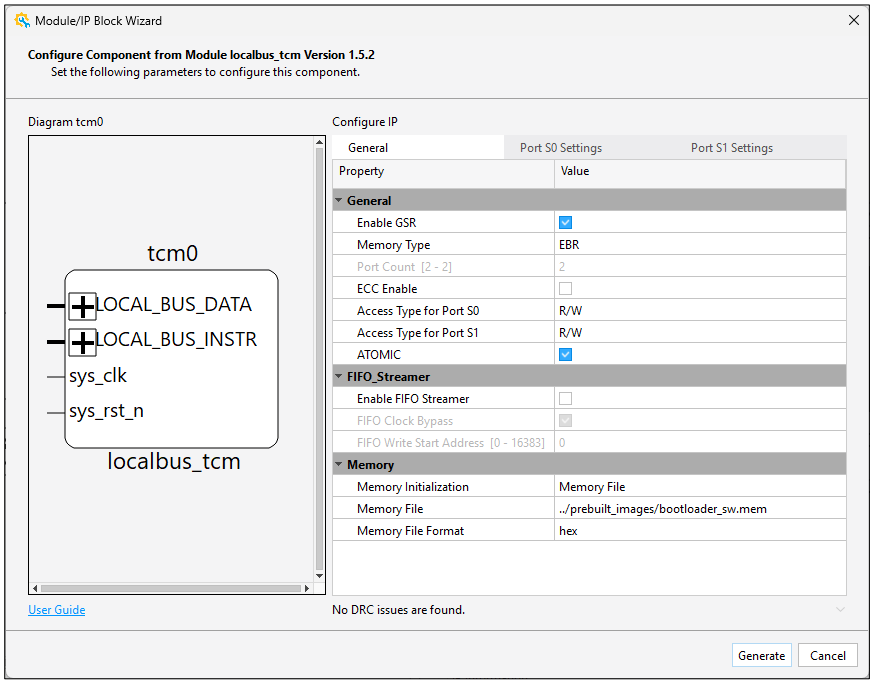

Figure 4.6. Tightly Coupled Memory IP Configuration

Table 4.5 provides the configuration details of the Tightly Coupled Memory IP.

Table 4.5.Tightly Coupled Memory IP Configuration

Property	Value	Description
General Tab		
Enable GSR	True	Enables global set/reset functionality.
Memory Type	EBR	Specifies the types of memory used for this TCM instance.
ECC Enable	False	Determines whether the Error Correction Code (ECC) is enabled. Default value is used in this design.
Access Type for Port S0	R/W	Sets read/write access for the INSTR port. Default value is used in this design.
Access Type for S1	R/W	Sets read/write access for the DATA port. Default value is used in this design.
ATOMIC	True	Indicate that TCM supports atomic instructions.
Enable FIFO Streamer	False	Enables the FIFO Streamer interface. If port count = 2, the streamer shares the DATA port; otherwise, it uses a dedicated port.
Memory Initialization	Memory File	Enable this option to specify the software memory initialization file **.mem.** This file allows the software to be compiled into the FPGA bitstream during the Lattice Radiant compilation process.
Memory File	Path to the mem file	
Memory File Format	hex	
Port S0 Settings Tab		
Port S0 : Data Width	32	Specifies the data width for the INSTR port. Default value is used in this design.
Port S0 : Address Depth	16,384	Defines the address depth for the INSTR port.
Read Port : Enable Output Register (S0)	False	Determines whether a registered output is applied to the INSTR port. Default value is used in this design.
Byte Enable for Port S0	True	Enables byte-level access for the INSTR port. Default value is used in this design.
Reset behavior for Port S0	sync	Specifies the reset mode for the INSTR port. Default value is used in this design.
Edit Address Range Port S0	False	Determines whether a memory address offset is enabled for the INSTR port. Default value is used in this design.
Port S1 Settings Tab		
Port S1 : Data Width	32	Specifies the date width for the DATA port.
Port S1 : Address Depth	16384	Defines the address depth for the DATA port. Maximum address depth depends on data width, memory type, and device.
Read Port : Enable Output Register (S1)	False	Determines whether a registered output is applied to the DATA port. Default value is used in this design.
Byte Enable for Port S1	True	Enables byte-level access for the DATA port. Default value is used in this design.
Reset behavior for Port S1	sync	Specifies the reset mode for the DATA port. Default value is used in this design.
Edit Address Range Port S1	False	Determines whether a memory address offset is enabled for the DATA port. Default value is used in this design.

4.5. AXI4 Interconnect

The AXI4 Interconnect IP connects the RISC-V RX CPU's instruction and data AXI ports (managers) to various subordinates within the system. These include the LPDDR4 memory controller, the SPI flash controller, and other peripherals.

Since the interconnect output is AXI-based, protocol conversion bridges are required when interfacing with subordinates that use AHB-Lite or APB protocols.

Figure 4.7 shows the configuration of the AXI4 Interconnect IP in the *General* tab.

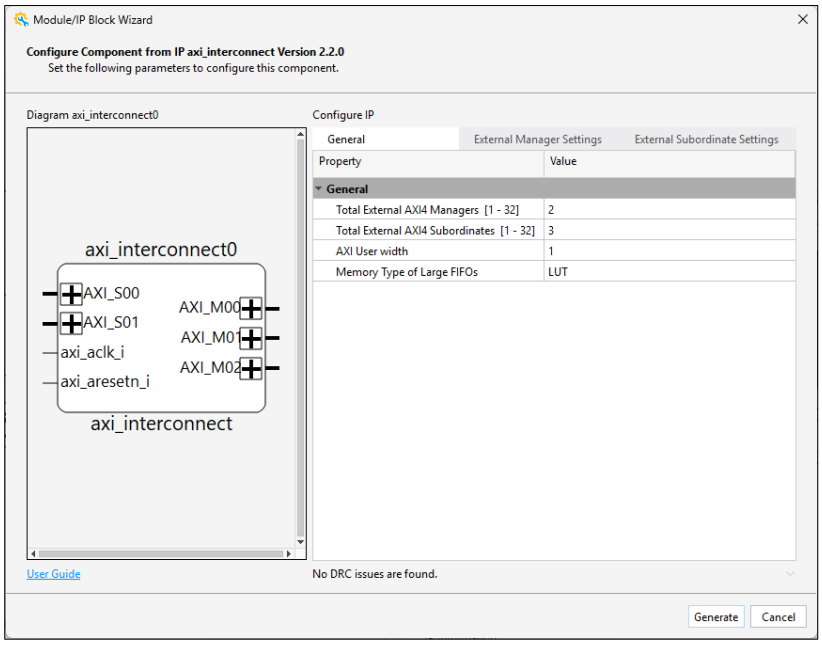

Figure 4.7. AXI4 Interconnect IP Configuration

Table 4.6 provides the configuration details of the AXI4 Interconnect IP.

Table 4.6. AXI4 Interconnect IP Configuration

Property	Value	Description
General Tab		
Total External AXI4 Managers	2	Interfaces with the RISC-V RX instruction and data AXI ports.
Total External AXI4 Subordinates	3	Interfaces with four subordinate components: • LPDDR4 AXI data path • SPI flash data path • APB-based register space (via APB Interconnect)
AXI User Width	1	AXI User signal is not used in this design and is set to the minimum value.
Memory Type of Large FIFOs	LUT	Allows you to select the memory type for larger FIFOs
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 supported by the AXI manager.
AXI Manager Max Data Width	32	Maximum data width supported by the AXI manager.
AXI Manager Max number of ID supports	1	Maximum number of IDs supported by the AXI manager.
External Manager AXI Access Type	WR	Enables both write and read access for the RISC-V RX AXI managers.

Property	Value	Description
External Manager AXI protocol	AXI4	Full AXI4 protocol is used for both RISC-V RX AXI managers.
External Manager CDC Enable	False	External Manager port interface <N> clock domain crossing enable/disable. This is enabled when external Manager port interface <N> is asynchronous to the AXI interconnect clock axi_aclk_i.
External Manager Address width	32	The 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 number of ID	1	The actual number of IDs used by the RISC-V RX (only one ID).
External Manager ID order Enable	False	ID order enable/disable for the external Manager <N>. Default value is used in this design.
External Manager Write/Read Accept	8	The number of outstanding write and read transactions each manager can accept. Use default value in this design.
External Manager Write Response Buffer Depth	8	Internal FIFO depth for the Write Response channel. Default value is used in this design.
External Manager Write/Read Data Buffer Depth	16	Internal FIFO depth for the Write/Read Data channel.
External Manager Priority	Round Robin	Use *Round Robin* scheme for each manager interface. Choose responses from different subordinates (at write response channel and read response channel).
External Subordinate Settings Tab		
External Subordinate AXI ID Width	2	Subordinate ID width = Manager ID width + log2 (number of managers) = 1 + log2(2) = 2
AXI Subordinate Max Address Width	32	Set the maximum address width (32 bits) for all AXI subordinates.
AXI Subordinate Max Data Width	256	Sets the maximum data width (256 bits) for AXI subordinates. This matches the data width of the LPDDR4 memory controller interface.
AXI Subordinate Max Fragment Count	8	Holds the maximum fragment count of the available external Subordinate
External Subordinate AXI Access Type	WR	Enables both write and read access for all subordinates.
External Subordinate Protocol Type	AXI4	Full AXI4 protocol is used for all subordinates. Conversion to AHB-Lite or APB is handled via protocol bridges.
External Subordinate CDC Enable	False	External Subordinate port interface <M> clock domain crossing enable/disable. Enabled only if the external subordinate port <M> is asynchronous to the AXI interconnect clock axi_aclk_i.
External Subordinate Address Width	32	Actual address width for each subordinate
External Subordinate Data Width	32	Data width is set to 256 on Subordinate 2, which is connected to LPDDR4 memory controller AXI interface. The remaining are set to 32.
External Subordinate ID returned out-of-order	True	Allows subordinates to return IDs out of order. Default value is used in this design.
External Subordinate Write/Read Issue	8	Number of outstanding write and read transactions each subordinate can issue. Default value is used in this design.

Property	Value	Description
External Subordinate Write Response Buffer Depth	8	Internal FIFO depth for the Write Response channel. Default value is used in this design.
External Subordinate Write/Read Data Buffer Depth	64	Internal FIFO depth for the Write/Read Data channel. Default value is used in this design.
External Subordinate Fragment Count	8	Number of address fragments per subordinate. Automatically populated by Lattice Propel Builder based on address maps and connections.
External Subordinate Priority	Round Robin	Use *Round Robin* scheme for each subordinate. Choose requests from different Managers (at write and read address channel)

5. Signal Description

Table 5.1 lists 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. This signal is triggered by pressing the push button on the board. CertusPro-NX devices: SW3 Lattice Avant devices: SW1
gpio_io	Input/Output	8	General-purpose Input/Output (GPIO) signals connected to the on-board LEDs.
UART Interface			
uart_rxd_i	Input	1	UART receives an input signal. This connects to the TXD output of the FTDI2232 chip on the board.
uart_txd_o	Output	1	UART transmits output signal. This connects to the RXD input of the FTDI2232 chip on the board.
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 signal. This signal is not used in the current Reference design.
flash_rst_n	Output	1	External SPI flash reset signal. This should be assigned to 1'b1 to release the flash from reset. **Note:** This signal is applicable only when targeting Lattice Avant devices.
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 outlines the software flow of the reference design.

6.1. RISC-V RX Boot up Sequence

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

Key terms and Definitions:

- **Boot up** – the process of starting the RISC-V RX CPU, loading software into the LPDDR4 memory, and executing the software.
- **Bootloader** – a software component that initializes the system, sets up the environment, and loads the application image into LPDDR4 memory.
- **Application** – the software loaded into LPDDR4 memory and executed by the CPU after the boot-up process.
- **SPI Flash** – a non-volatile memory device used to persistently store the application software.
- **Application image** – a converted version of the application software that includes additional metadata and is used for SPI flash programming.

Figure 6.1 illustrates the boot-up sequence in relation to the terms defined above.

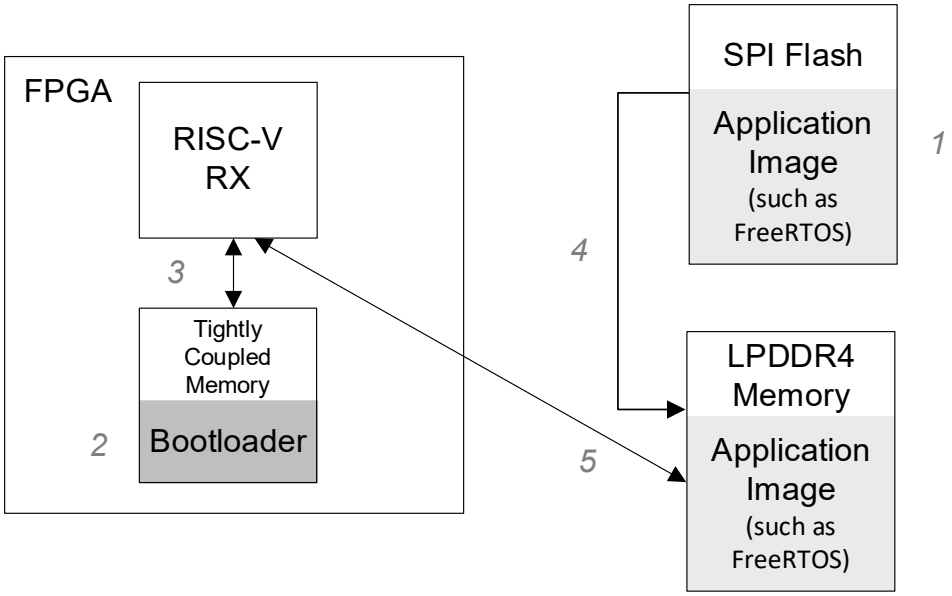

Figure 6.1. Boot up Sequence

The following steps describe the boot-up sequence of the RISC-V RX CPU, as shown in Figure 6.1:

1. The application image (such as FreeRTOS) is pre-programmed into SPI flash before the system boots.
2. The tightly coupled memory is pre-initialized with the bootloader during the FPGA configuration process via the bitstream file.
3. When the reset is de-asserted, the RISC-V RX CPU begins executing the bootloader.
4. The bootloader initializes the LPDDR4 memory controller and copies the application image from SPI flash into LPDDR4 memory.
5. Once the memory is ready and the image is verified, the RISC-V RX CPU jumps to the LPDDR4 memory address and begins executing the application, completing the boot-up process.

6.2. Bootloader Software Flow Chart

Figure 6.2 illustrates the bootloader software flow.

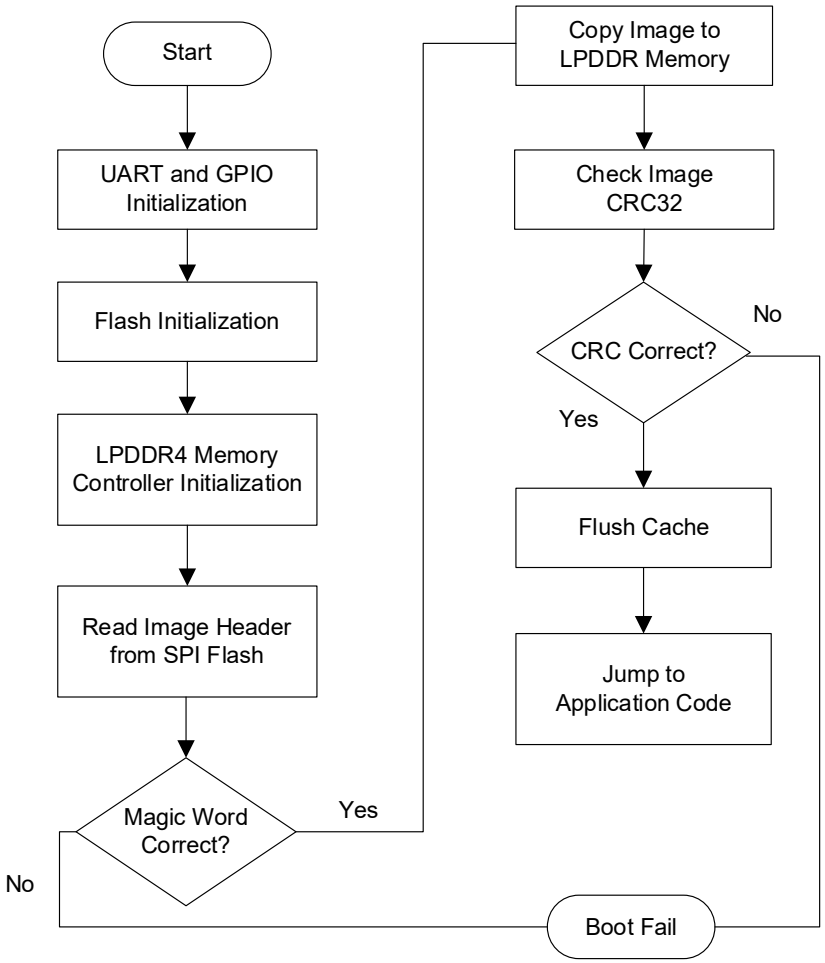

Figure 6.2. Bootloader Software Flow

The following steps describe the bootloader sequence as shown in Figure 6.2:

1. Initialize the UART interface and GPIO (used for driving the onboard LED).
2. Initialize the SPI flash controller. Read the flash ID and display its value.
3. Initialize the LPDDR4 memory controller. Trigger the memory training process and monitor its completion. Proceed only if all training steps pass successfully.
4. Read the application image header from SPI flash and check for the magic word. Refer to the Application Image Format section for details.
5. If the magic word matches, continue to the next step. If not, enter the boot fail state and halt the boot process.
6. Copy the application image from SPI flash to LPDDR4 memory.
7. Generate a CRC32 checksum for the image in LPDDR4 memory and compare it with the value from the image header.
8. If checksum matches proceed. If not, enter the boot fail state and halt the boot process.
9. Flush the CPU cache.
10. Jump to the application code address in LPDDR4 memory to begin execution.

6.3. Application Image Format

The application image consists of the application software binary along with a header containing supplementary information. As illustrated in Figure 6.3 this header is appended to the binary and is essential for the bootloader to perform image validation and integrity checks during startup. Table 6.1 provides the image header details.

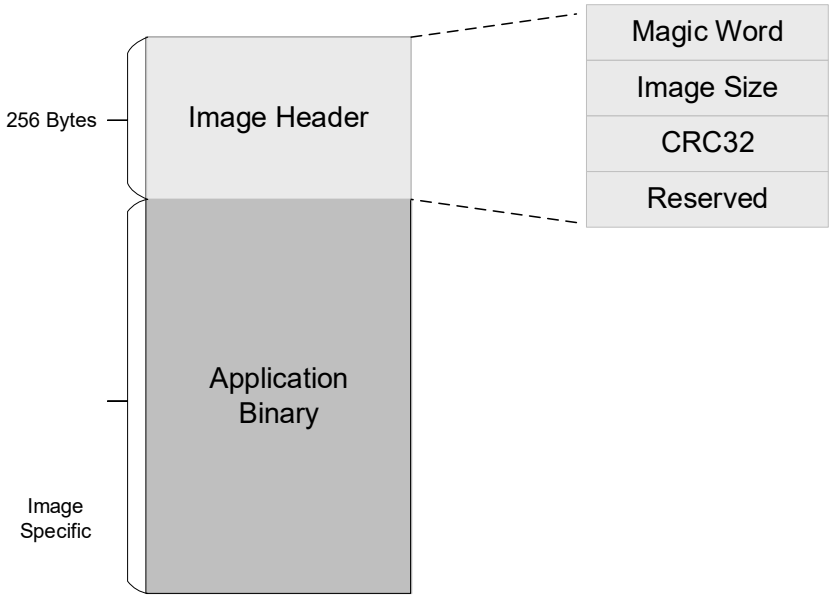

Figure 6.3. Application Image Format

Table 6.1. Application Image Header

Header Property	Value	Description
Magic Word	0x4C534343	Represents LSCC in ASCII notation. It is used to verify the presence of a valid image header in flash memory.
Image Size	Variable	Specifies the size of the application binary in bytes. This value indicates the total amount of data that the bootloader must copy during the boot process.
CRC32 Checksum	Variable	A CRC32 checksum computed for the application binary. It is used to verify data integrity during the boot process.
Reserved	N/A	Padding use to align the image header to the flash page size (256 bytes).

6.4. Application Image Generation

This section outlines the process for generating the application image described in the Application Image Format section.

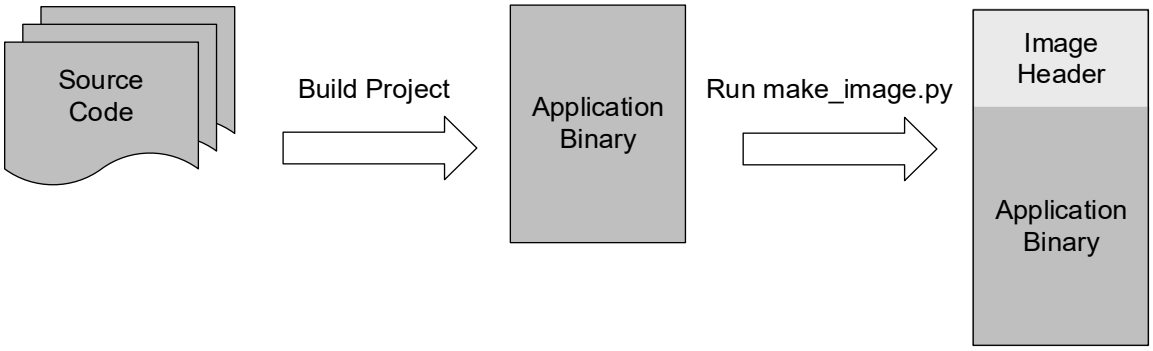

Figure 6.4. Application Image Generation

The image generation workflow, shown in Figure 6.4 consists of the following steps:

1. Build the application software project from source code using the Lattice Propel SDK. This step produces the application binary file **.bin**.
2. Run the *make_image* python script, provided with this reference design, to append the image header on the top of application binary.

For detailed instructions, refer to the Generating the Flash Image section.

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 included in the design package.

The following Lattice development boards are supported:

- CertusPro-NX Versa Board
- Lattice Avant-E Evaluation Board

To proceed, follow the instructions in the appropriate setup section based on the board you are using:

- Setting up the CertusPro-NX Versa Board
- Setting up the Lattice Avant-E Evaluation Board

7.1. Extracting the Reference Design Files

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

- CertusPro-NX-RISCV-LPDDR4-RD.zip
- Avant-RISCV-LPDDR4-RD.zip

Once downloaded, extract the contents of the **.zip** file to your local directory—for example C:\workspace\.

The extracted directory will be named based on the design package you downloaded:

- CertusPro-NX-RISCV-LPDDR4-RD
- Avant-RISCV-LPDDR4-RD

This path will be referred to as *<my_work_dir>*. For example, ***<my_work_dir> = C:\workspace\CertusPro-NX-RISCV-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

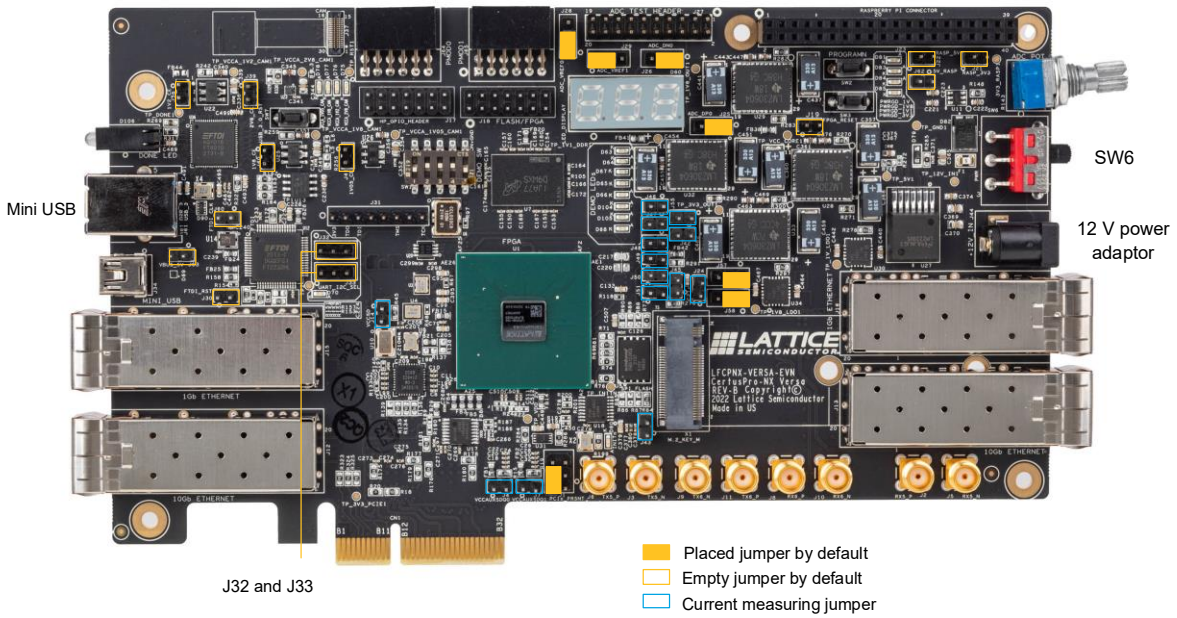

Figure 7.1. CertusPro-NX Versa Board

To set up the board:

1. Connect the 12 V power adaptor to J44.
2. Connect a Mini USB Type-A cable from your PC to J34.
3. Enable UART by placing jumpers between Pin 1 and Pin 2 on both J32 and J33.
4. Turn on SW6 to ON.

7.3. Setting up the Lattice Avant-E Evaluation Board

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

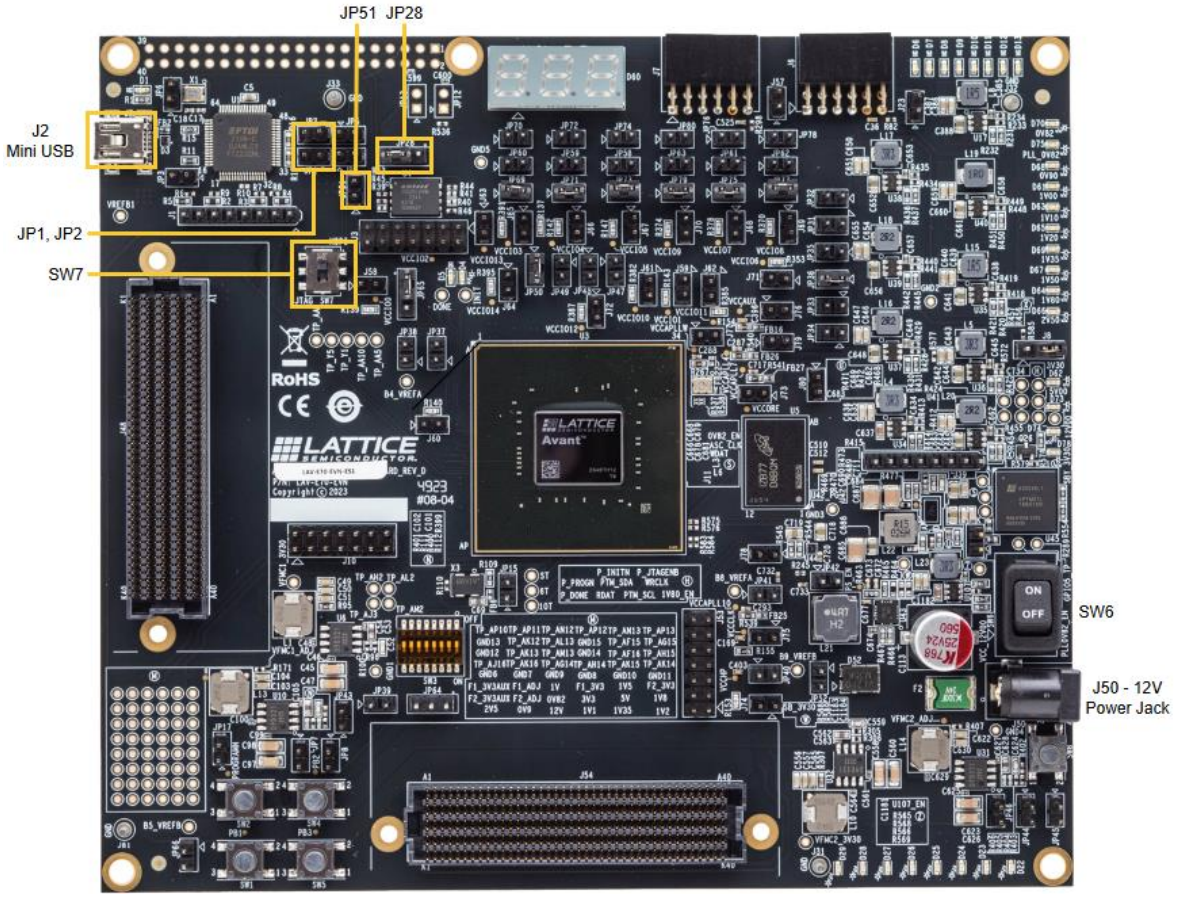

Figure 7.2. Lattice Avant-E Evaluation Board

To set up board:

1. Connect the 12V power adaptor to J50 (Power Jack).
2. Connect a Mini USB Type-A cable from your PC to J2 (Mini USB).
3. Enable UART by placing jumpers between Pin 1 and Pin 2 on both JP1 and JP2.
4. Toggle switch SW7 to JTAG.
5. Connect the following jumper pins:
   a. Pins 1 and 2 on JP51
   b. Pins 2 and 3 on JP28
6. Turn on SW6 to ON.

7.4. Setting up the UART Terminal

The software code in this reference design displays output messages to a terminal via 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. From the **Port** drop-down menu, select the correct COM port. It is typically the one with the higher number.
   Example: If both **COM1** and **COM2** are listed, choose **COM2**.
4. Click **OK**

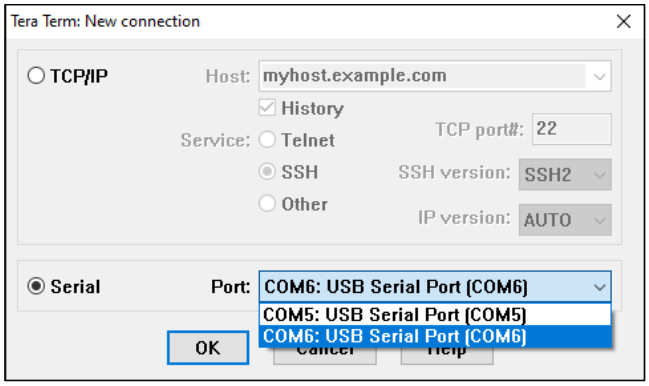

Figure 7.3. Tera Term New Connection

5. Go to **Setup > Serial Port**.
6. In the **Serial port setup and connection** dialog box, set **Speed** to **115200**.
7. Configure the remaining settings as shown in Figure 7.4.
8. Click **New setting** to apply the configuration.

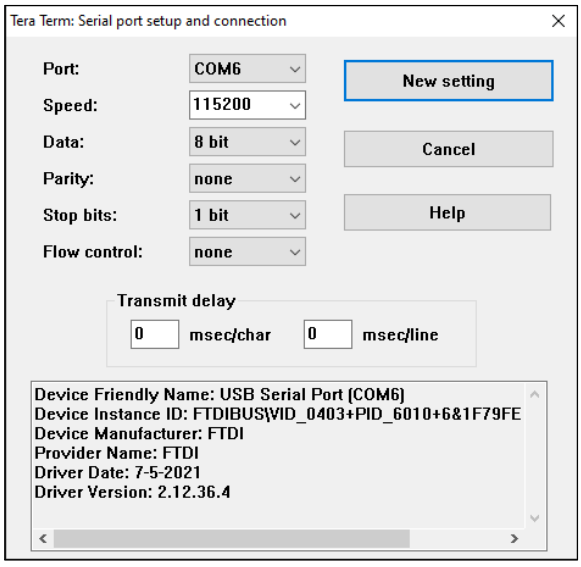

Figure 7.4. Tera Term Serial Port Setup and Connection

7.5. Programming the Application Image to SPI Flash

Before starting the bootloader software, the application image must be programmed into SPI Flash. This ensures that a valid image is available for the bootloader to load into LPDDR4 memory.

To program the application image:

1. Launch **Radiant Programmer 2025.1** from the Windows **Start** menu.

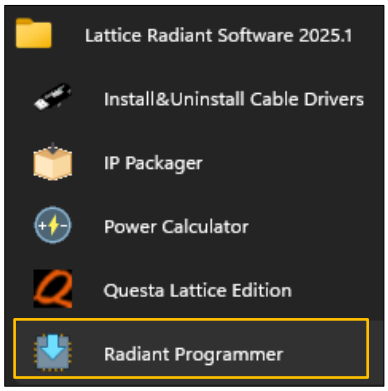

Figure 7.5. Windows Start Menu > Radiant Programmer

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

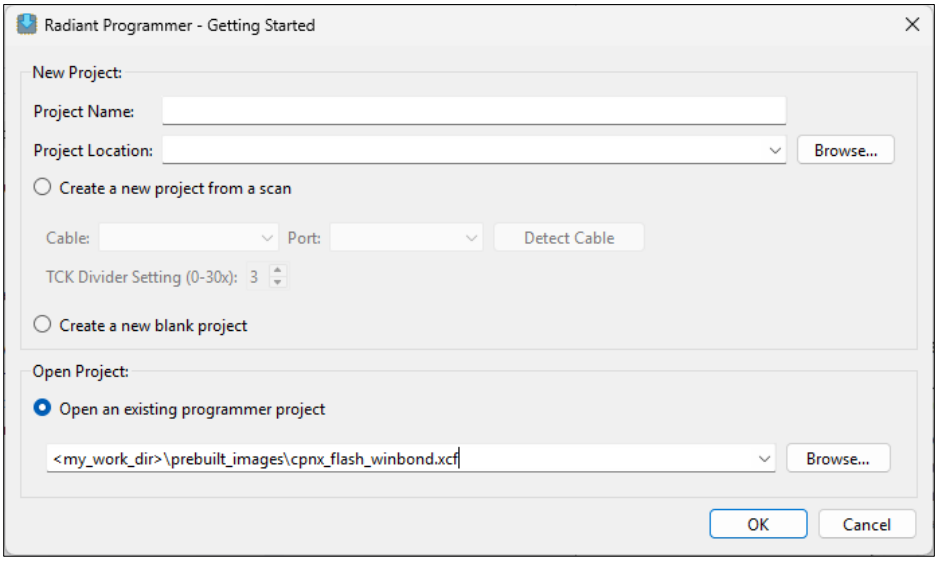

Figure 7.6. Radiant Programmer - Getting Started Dialog Box

3. Click the **Browse** button and navigate to this directory: *<my_work_dir>\prebuilt_images*.
4. Select the appropriare **.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-E Evaluation Board	Winbond W25Q512JV	avant_flash_winbond.xcf

5. Click **OK**.

6. In the main interface, go to **Run > Program Device**.
7. Wait for the process to be completed. The **Output** console should display the message **Operation successful**.

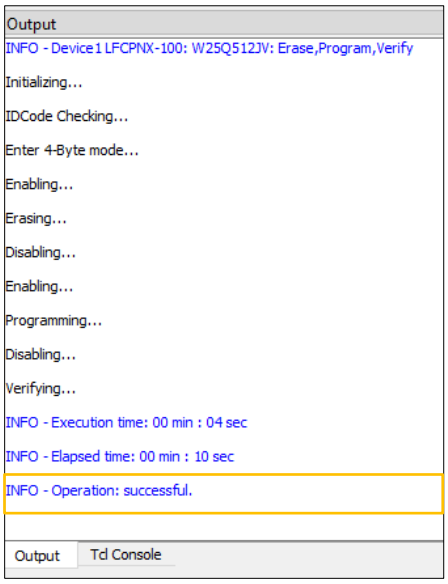

Figure 7.7. Output Console

Note: If the flash programming fails, refer to the Debugging the Design section.

7.6. Programming the FPGA Bitstream

The bitstream **.bit** file is programmed into the FPGA device. This bitstream includes the RISC-V LPDDR4 design with the bootloader code pre-initialized in tightly coupled memory.

To program the bitstream:

1. In the Radiant Programmer main interface, go to **File** > **Open Project**.
2. Browse to the following directory: *<my_work_dir>\prebuilt_images.*
3. Select the appropriate **.xcf** file for your board:
   - *cpnx_bitstream_program.xcf*
   - *avant_bitstream_program.xcf*
4. Click **Open**.
5. In the main interface, go to **Run > Program Device**.
6. Wait for the process to be completed. The **Output** console should display the message "**Operation: successful".**

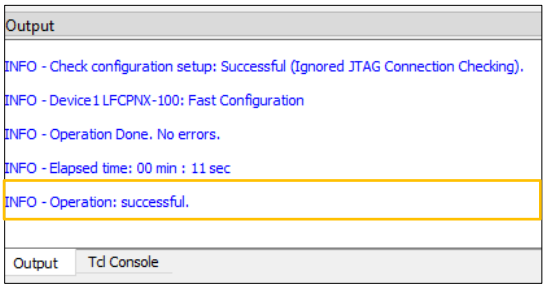

Figure 7.8. Output Console

Note: If you encounter an error while programming the bitstream on the Avant device, follow these steps:

1. Power cycle the board (turn it off and then on).
2. Repeat the Program Device steps.

7.7. Verifying the Results

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

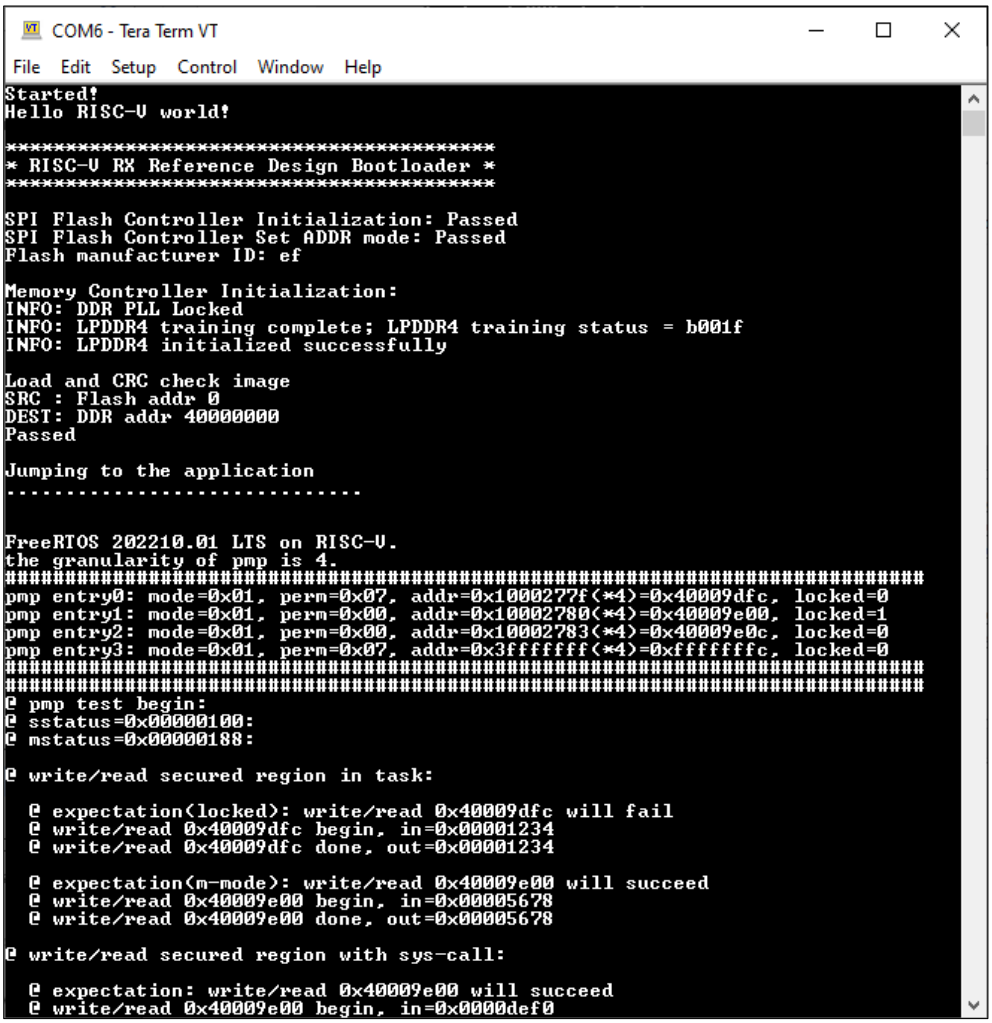

Figure 7.9. RISC-V RX and LPDDR4 Reference Design Results

The following process is observed:

1. The bootloader software is executed first. It initializes the SPI flash controller and the LPDDR4 memory controller.
2. The bootloader copies the application image from SPI flash to LPDDR4 memory. It performs a CRC check to verify the integrity of the image before proceeding.
3. If the CRC check passes, the bootloader instructs the CPU to jump to the LPDDR4 memory address and begin executing the application.
4. The FreeRTOS application is then executed from the LPDDR4 memory.

8. Compiling the Reference Design

This section outlines the process of compiling the RISC-V and LPDDR4 reference design. Compilation is necessary to generate binary files from the provided source files. The process uses a set of software tools to produce both the FPGA bitstream and the RISC-V CPU software executables.

Required Tools

The following tools are used in the compilation process:

- Lattice Propel Builder 2025.1
- Lattice Radiant Software 2025.1
- Lattice Propel SDK 2025.1

Design Flow Overview

The typical design flow for a Lattice RISC-V based project includes:

- Building and Generating the RISC-V CPU using the Lattice Propel Builder
- Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software
- Building the Software Project using Lattice Propel SDK

Reference Design-Specific Flow

For this reference design, additional steps are required to create the bootloader software and application image:

- Pre-initializing Tightly Coupled Memory
- Generating the Flash Image

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

The RISC-V CPU system in this reference design is built and generated using Lattice Propel Builder. Components (IP blocks) are instantiated and interconnected using Lattice Propel Builder's schematic graphical user interface, as shown in Figure 8.1.

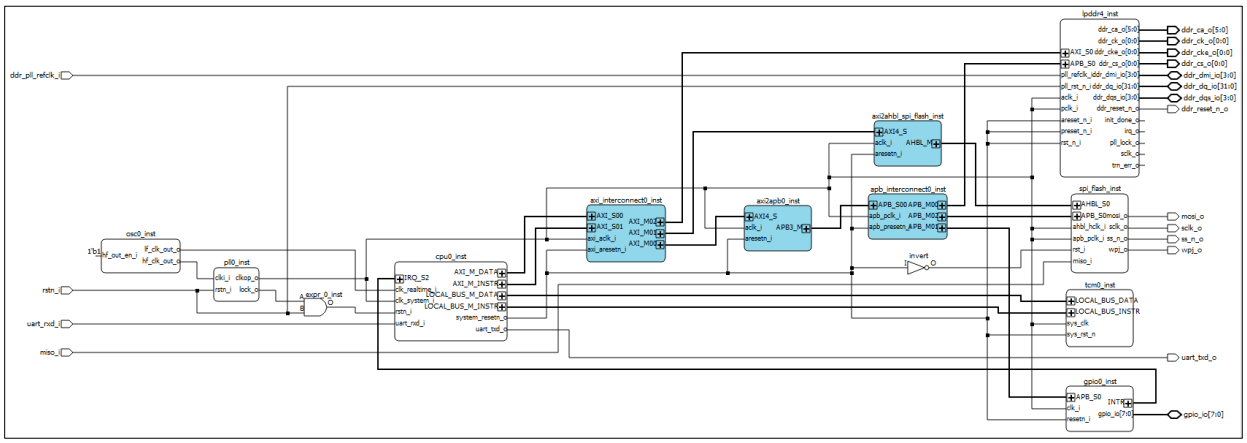

Figure 8.1. Lattice Propel Builder Schematic view

Note: If you have any modification to the reference design, refer to the Customizing the Reference Design section before proceeding.

To generate the RISC-V CPU system:

1. Launch **Lattice Propel Builder 2025.1** from the Windows start menu.

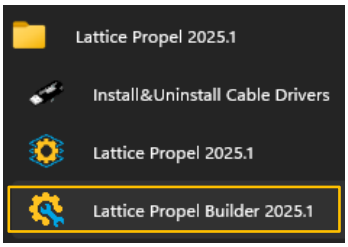

Figure 8.2. Windows Start Menu > Lattice Propel Builder

2. In the Lattice Propel Builder main interface, go to **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 doing this step, ensure you have completed steps in the Extracting the Reference Design Files section.
4. Select either *cpnx_riscv_rx_rd.sbx* or *avant_riscv_rx_rd.sbx*, then click **Open**.
   Note: The warning *Port Bus width does not match for port INFO* can be safely ignored.
5. In Lattice Propel Builder, go to **Design > Validate Design**.

   Note: You should see no error in the **Tcl Console** window. The following warning *Dangling input INFO* can be safely ignored.

```
Tcl Console
INFO <2359136> - Start: sbp_design drc.
INFO <2357031> - Dangling input Port 'BUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'RUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'HMASTLOCK' is set to default value '0' on bus 'AHBL_S0' in component 'spi_flash_inst'.
INFO <2359137> - Finished: sbp_design drc.
```

Figure 8.3. TCL Console – No Error for Validate

6. Click **Design > Generate**.

   Note: You should see no error in the **Tcl Console** window. The following warnings can be safely ignored:
   - *Dangling input INFO*
   - *Tightly coupled memory driver* warning

```
Tcl Console
% sbp_design generate
INFO <2359189> - Start: sbp_design generate.
INFO <2357031> - Dangling input Port 'BUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'RUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'HMASTLOCK' is set to default value '0' on bus 'AHBL_S0' in component 'spi_flash_inst'.
% sbp_design save
% sbp_design pge sge  -i {C:/workspace/CertusPro-NX-RISCV-LPDDR4-RD/cpnx_riscv_lpddr4_rd/cpnx_riscv_lpddr4_rd.sbx} -o {C:/workspace/CertusPro-NX-
RISCV-LPDDR4-RD/cpnx_riscv_lpddr4_rd/../} -nc
```

Figure 8.4. TCL Console – No Error for Generate

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 (generated from Lattice Propel Builder) and produce the FPGA bitstream file.

To compile the project.

1. Launch **Lattice Radiant Software 2025.1** from the Windows **Start** menu. Alternatively, you can launch Lattice Radiant software directly from Lattice Propel Builder.

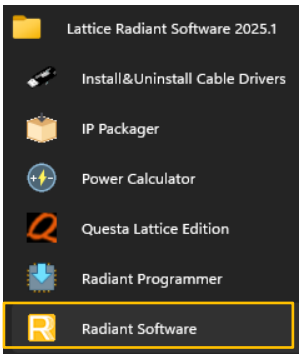

Figure 8.5. Windows Start Menu > Radiant Software

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

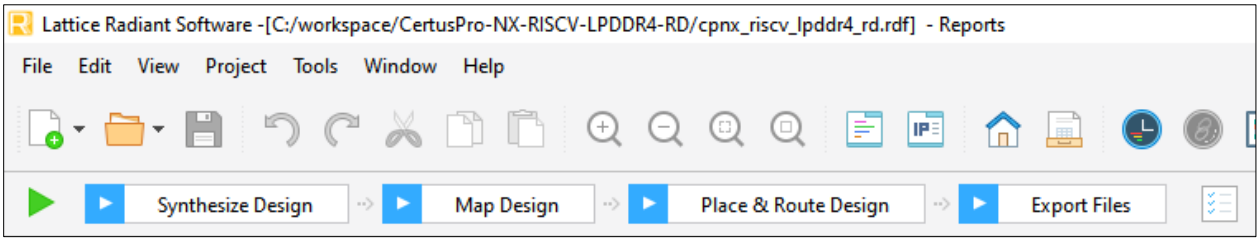

Figure 8.6. Export Files

5. Wait for the compilation to be completed. The ***.bit*** file will be generated or updated in the project path: *<my_work_dir>\impl_1*

Note: If you encounter timing violations after compilation, this may be due to suboptimal Placement Iteration Start Point settings for Place and Route on your machine. In this case, follow these steps to adjust Place and Route settings:

1. In Lattice Radiant software, go to **Project > Active Strategy > Place & Route Design Settings**.
2. Change the following parameters:
   - Placement Iteration Start Point from ***1*** to ***2***.
   - Placement Iterations from ***1*** to ***5***.
   - Placement Save Best Run from ***1*** to **5**.
3. Click **OK** to apply changes.
4. Click **Export Files** to rerun the Place and Route process.

This reruns the design with five different incremental start points. The Place and Route Reports will display all five results and automatically select the one with the best timing performance.

8.3. Building the Software Project using Lattice Propel SDK

Lattice Propel SDK is used to build the software project for RISC-V CPU. This section outlines the procedure for compiling the software components required by the reference design.

8.3.1. Setting Up a New Lattice Propel SDK Workspace

Before building software projects in Lattice Propel SDK, you need to set up an Eclipse-based workspace. This step is only required the first time you launch the Lattice Propel SDK. For subsequent Lattice Propel SDK launches, you can reuse the previously created workspace.

To create a new Lattice Propel SDK workspace:

1. Launch **Lattice Propel 2025.1** from the Windows **Start** Menu.

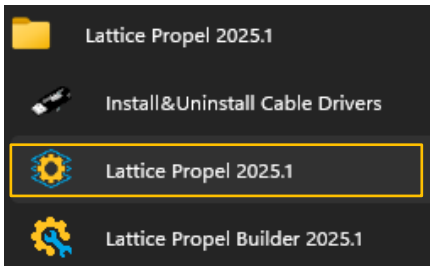

Figure 8.7. Windows Start Menu > Lattice Propel 2025.1

2. In the Lattice Propel Launcher dialog box, click **Browse**.
3. Navigate to your working directoy: *<my_work_dir>\sw,* then click **Select Folder.**
4. When the workspace path is set, click **Launch** to open the SDK. Figure 8.8 illustrates the reference design workspace setup.

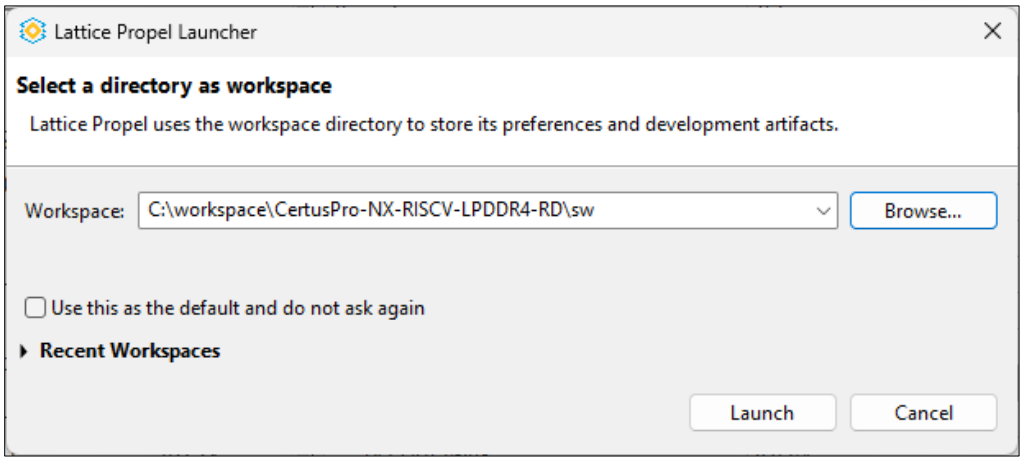

Figure 8.8. Propel SDK Workspace Setup

8.3.2. Building Bootloader and Application Software

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

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

To build the software project in Lattice Propel SDK:

1. In the Lattice Propel SDK main interface, go to **File > Import**.
2. In the **Import Wizard Select** dialog box, expand **General** and select **Existing Projects into Workspace**.

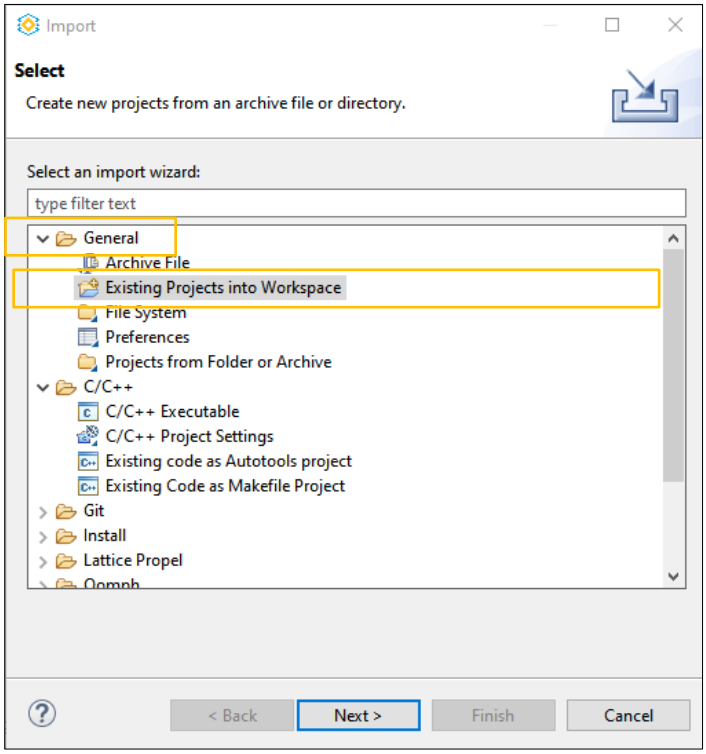

Figure 8.9. Select Existing Projects

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

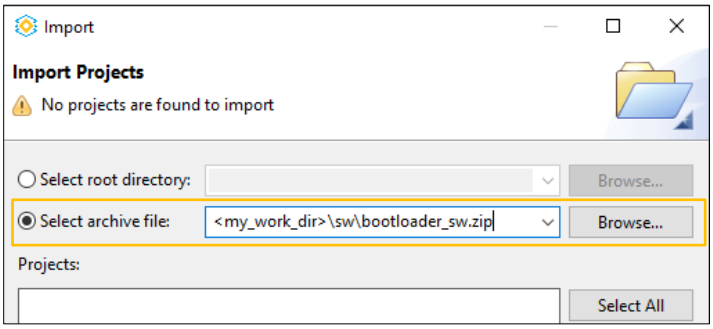

Figure 8.10. Import Projects

7. Click **Finish** to import the project.
8. The *bootloader_sw* project will now appear in the **Project Explorer**.

Figure 8.11. Project Explorer

9. In **Project Explorer**, right-click *bootloader_sw* and select **Build Project**.
10. Confirm that no errors appear in the console window.
11. Verify that the *bootloader_sw.mem* file is generated in this directory: *<my_work_dir>\sw\bootloader_sw\Debug*

Figure 8.12. Console

12. Repeat steps 1 to 11 to import and build the FreeRTOS application project using *freertos_app_sw.zip*.
13. After successfully building the FreeRTOS application, verify that *freertos_app_sw.bin* file is generated in this directory: *<my_work_dir>\sw\freertos_app_sw\Debug*

8.4. Pre-initializing Tightly Coupled Memory

As described in the RISC-V RX Boot up Sequence section, the RISC-V RX CPU executes the bootloader software from the tightly coupled memory after reset. The memory content is pre-initialized with the bootloader software during FPGA configuration.

Note: Perform these steps only if you are updating the bootloader software or initializing the tightly coupled memory with your own software program.

To initialize tightly coupled memory:

1. In Lattice Propel Builder, double click the **tcm0_inst** block. This opens the **Module/IP Block Wizard**.

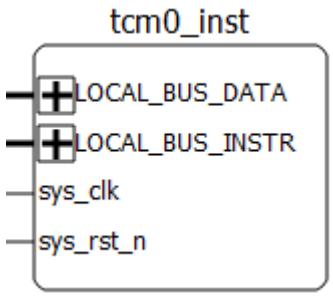

Figure 8.13. system0_inst Block

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, navigate to: *<my_work_dir>\sw\bootloader_sw\Debug*.
6. Select *bootloader_sw.mem* and click Open. Figure 8.14 shows the final configuration.
7. In the **Module/IP Block Wizard**, click **Generate,** then **Finish**.

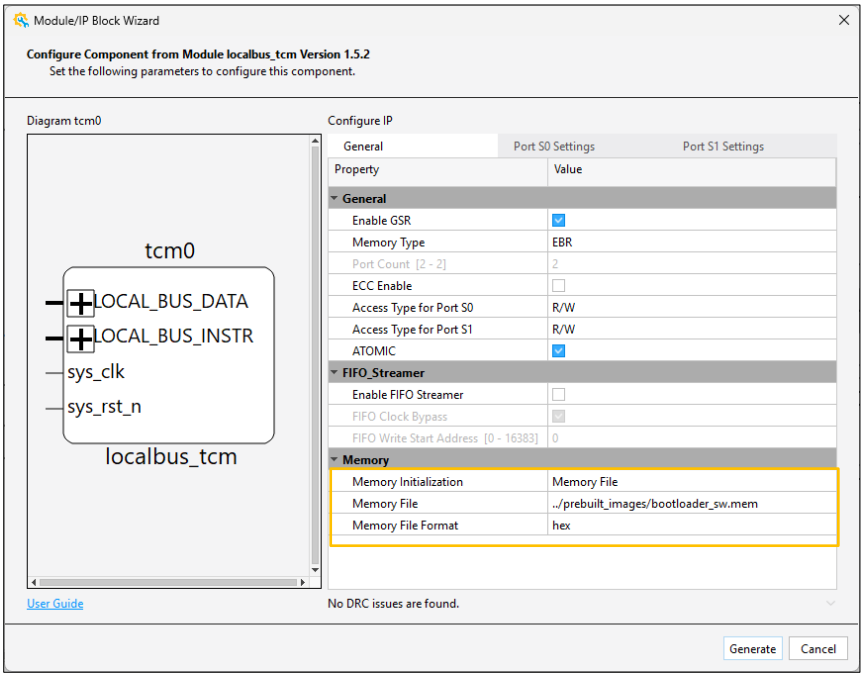

Figure 8.14. Tightly Coupled Memory IP Setting to Pre-initialize with Bootloader Software

8. Back in Lattice Propel Builder, go to **Design > Generate** to update the output files.
9. Open Lattice Radiant and click **Export Files** to regenerate the updated bitstream file.

Note: For detailed steps on using Lattice Propel Builder and Lattice Radiant software, refer to:

- Building and Generating the RISC-V CPU using the Lattice Propel Builder
- Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software

8.5. Generating the Flash Image

To prepare the application image for SPI flash programming, a header must be appended to the compiled binary using the provided Python script, as described in the Application Image Generation section. This process is required when modifying the application software or replacing the FreeRTOS application with a custom program.

The script *make_image.py* is included in the reference design package in this directory: *<my_work_dir>\sw*.d

Pre-requisite: Ensure the **.bin** file has been generated 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 Python 3 is not installed, download it from the Microsoft Store.

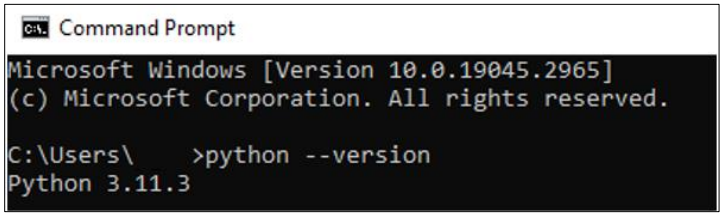

Figure 8.15. Python Verification

3. In command prompt, enter cd <my_work_dir>\sw to change directory.
4. Run the Python script to generate the flash image by entering the following:

   python.exe make_image.py freertos_app_sw\Debug\freertos_app_sw.bin
   ..\prebuilt_images\freertos_app_boot.bin
5. After successful execution, the file *freertos_app_boot.bin* will be generated in this directory: *prebuilt_images*.
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 Lattice Propel Builder, which serves as the primary design entry tool.
- Software-related modifications are performed using Lattice Propel SDK.

9.1. Changing LPDDR4 Memory Controller Parameters

The LPDDR4 memory controller IP parameters are configured based on the DRAM chips on the CertusPro-NX Versa Board and the Lattice Avant-E Evaluation Board. You can modify these IP parameters to match the specific board and DRAM chip used in your project.

To modify the IP parameters:

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

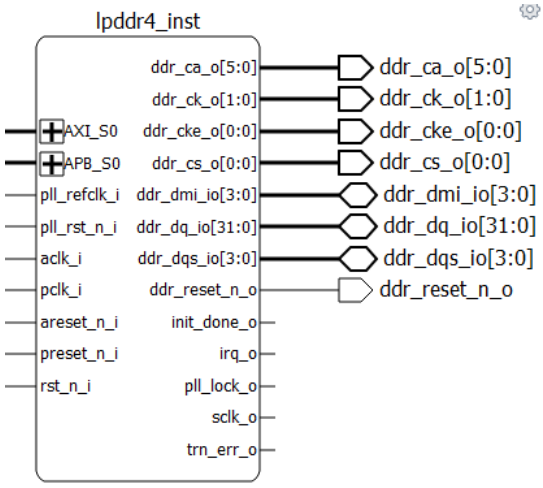

Figure 9.1. lpddr4_inst Block

2. In the **Module/IP Block Wizard**, adjust the parameters to match your board and DRAM chip. Refer to Table 9.1 for possible customization options.

Table 9.1. LPDDR4 Memory Controller IP Parameters Customization

Parameter	Description
DDR Command Frequency (MHz)	Set the frequency to a value that matches your design requirements. For Lattice Avant devices, the maximum supported frequency is 800 MHz. For CertusPro-NX devices, the maximum supported frequency is 533 MHz.
DDR Density	Set the density per channel (GB) to match the specifications of the DRAM chip used in your design. This setting directly affects the AXI4 address bus width of the IP interface.
DDR Bus Width	Set the DDR data bus to match the specifications of the DRAM chip used in your design. This configuration affects the width of the DQ, DQS and DMI buses, and requires updates to the FPGA pin assignments.

3. In the **Module/IP Block Wizard**, click **Generate** to ensure the RTL is updated according to your changes.

Note: After customizing the IP, regenerate the Lattice Propel Builder system and in Lattice Radiant click **Export Files** to update the bitstream.

9.2. Adding New Component to the Lattice Propel Builder System

This reference design can serve as a base design for adding new components or IPs required by your project. You can perform these modifications using the Schematic view in Lattice Propel Builder.

The following procedure outlines the general design flow:

9.2.1. Hardware Flow

To add a new IP:

1. In Lattice Propel Builder, select and add a new IP from the **IP Catalog** window. Complete the IP configuration using the wizard and generate.

   Note: To create a custom IP, refer to Lattice IP Packager 2025.1 (FPGA-UG-02236).

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 a new Manager/Subordinate interface in axi_interconnect_0_inst (depending on the interface type of the new IP). Connect the new IP to the newly added interface on the interconnect.
   - APB – Add a new Requestor/Completer interface in apb_interconnect0_inst. Connect the new IP to the newly added interface on the interconnect.
   - AHB-Lite – Add a new Manager/Subordinate interface in axi_interconnect_0_inst. Since the new IP uses an AHB-Lite interface insert an AXI4 to AHB-Lite Bridge IP between the interconnect and the new IP.
3. Connect the clock and reset signals to the new IP.
4. Export the I/O of the new IP to the top-level module (if applicable).
5. In Lattice Propel Builder, go to the **Address** view and assign a base address to the new IP.
6. Once all connections are complete, 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 Lattice Propel SDK, right-click the software project you are working on and select **Update Lattice C/C++ Project**.

   In the **Update System and BSP** dialog box, you may encounter a **Directory is not correct** error (see Figure 9.2). This occurs when the software project was created on a different PC with a different system environment path and then imported into your SDK.

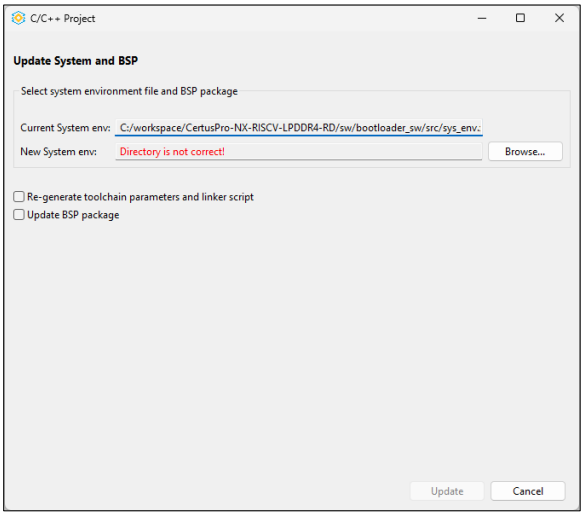

Figure 9.2. New System Error

2. Click the **Browse** button to navigate to the new system environment path on your PC.

   Example: *<my_work_dir>/sge/sys_env.xml*

   Note: This step updates the software project to point to the Lattice Propel Builder system environment file on your PC. The correct path will now be displayed instead of the previous error.

3. Select the **Update BSP package** checkbox. This will show the new IP driver and version available for update.

   Note: DO NOT select **Re-generate toolchain parameters and linker script** checkbox. The software projects provided in this reference design contain a modified linker script. Selecting this option will overwrite those modifications.

4. Click **Update** to complete the process.
5. Build the software project to generate the 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 predefined address. By default, this address is set to 0x40000000, which corresponds to the beginning of the memory. If you need to change the load address, you must modify the bootloader source code.

To change the application image load address:

1. In Lattice Propel SDK Project Explorer, open ***bootloader_sw*** > ***src*** > ***main.c***.
2. Edit the following line in ***main.c*** to reflect the new load address.

   #define APPLICATION_MEMORY_START_ADDR 0x40000000

   Note: The load address must be 16-byte aligned. Valid examples include: 0x40000010, 0x40000020, and so on.

3. Rebuild the *bootloader_sw* project.

9.4. Updating Linker Script to LPDDR4 Memory Address

By default, the software application is not configured to execute from LPDDR4 memory. To enable execution of your software project from the LPDDR4 external memory, you must modify the linker script.

To modify the linker script:

1. In Lattice 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 tightly coupled memory (tcm0_inst), starting at address 0x0 with a length of 32 KB.

```
MEMORY
{
    tcm0_inst (rwx) : org = 0x0, len = 0x10000
}
```

3. Modify the MEMORY region to set the LPDDR4 memory (DDR) as below, starting at address 0x40000000 with a length of 1 GB.

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

4. Replace instances of tcm0_inst with DDR in each section, including
   - .text
   - .ctors
   - .dtors
   - .rodata
   - .data

- .bss
- .heap
- .stack

5. Save the *linker.ld* file.
6. Right-click the project and select **Clean Project**.
7. Right-click again and select **Build Project**.
8. Repeat the steps in the Generating the Flash Image section to generate the new flash image.
9. Repeat the steps in the Programming the Application Image to SPI Flash section to program image to flash.

10. Debugging the Design

This section lists potential issues and their troubleshooting steps.

10.1. SPI Flash Programming Fail

Check the following items if SPI flash programming fails during the procedure described 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. Open Radiant Programmer and double-click **Erase, Program, Verify** under the **Operation** column. This opens the **Device Properties** dialog box.
2. In the **General** tab, configure the following options based on the flash device used. Refer to 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
     W25Q512JV for Winbond
3. Click **Load from File** to verify that the data file size is correct.

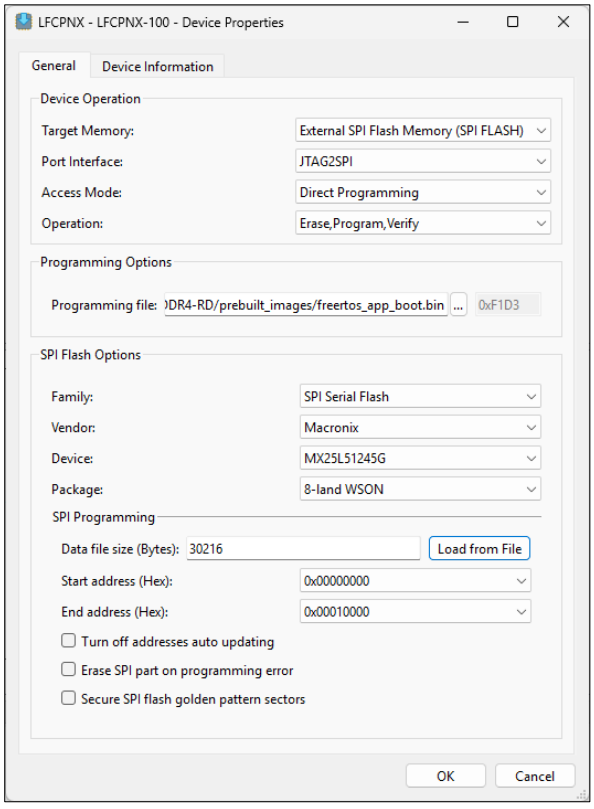

Figure 10.1. Flash Device Properties for Macronix

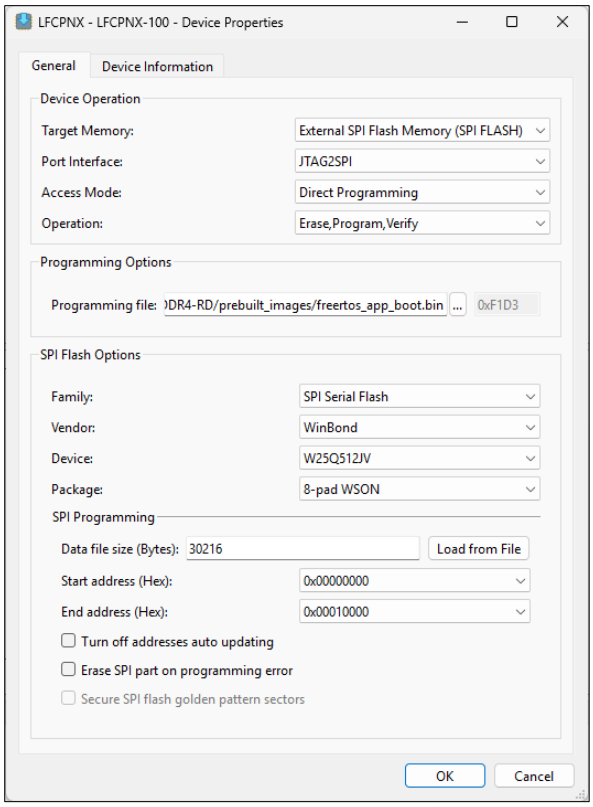

Figure 10.2. Flash Device Properties for Winbond

10.1.2. Check TCK Divider Setting

To check TCK Divider setting:

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

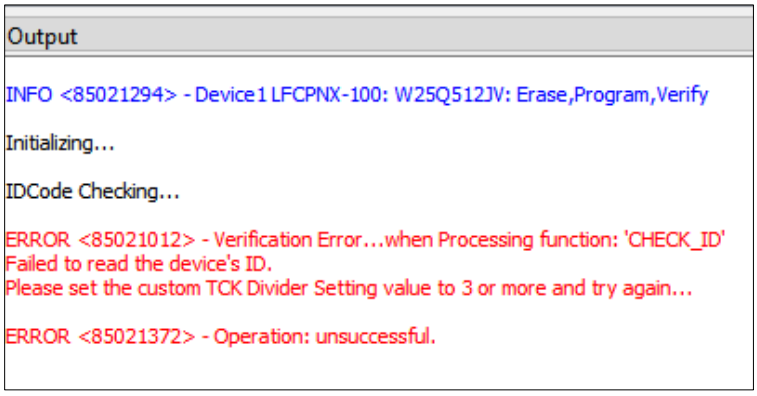

Figure 10.3. TCK Divider Error

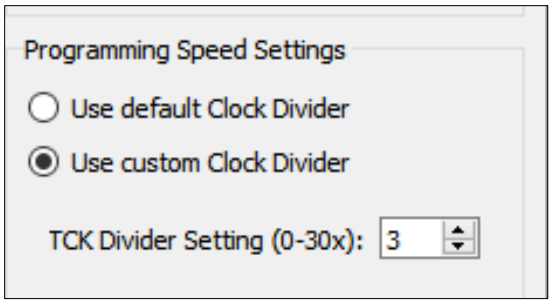

Figure 10.4. Programming Speed Settings

10.1.3. 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 multiple cables are connected to your PC, the **Detect Cable** function will list all of them. Select the correct cable that connects to the board you are working on. To avoid confusion, it is recommended to connect only one cable between the PC and the board.
2. For the Lattice Avant board, ensure that the *HW-USBN-2B* cable is properly connected to the board. Refer to the Setting up the Lattice Avant-E Evaluation Board section for detailed instructions.

10.2. UART Serial Terminal Prints Incorrect Characters

If Incorrect (junk) characters appear on the serial terminal, follow these steps:

1. Verify the **Baud Rate** or **Speed** settings in your terminal software. Ensure it is set to **115200**.
2. Update the clock frequency if you modify the PLL output clock.
   - In the Lattice Propel Builder, open RISC-V RX wizard
   - Double-click RISC-V RX CPU IP block
   - Under local UART, set the correct System Clock Frequency value

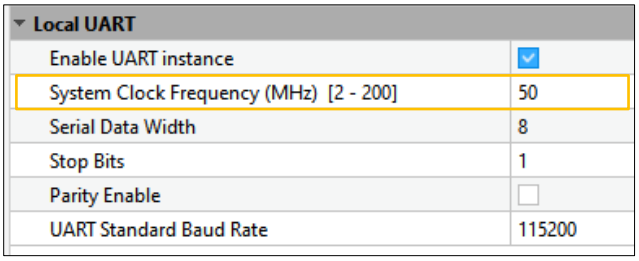

Figure 10.5. UART Settings

11. Known Issues

11.1. Avant LPDDR4 Link Training Failure

Link training may fail if constraints or driver settings are not updated correctly. Refer to sections 11.1.1 and 11.1.2 for required updates.

Note: This issue applies only to Avant DDR Memory Controller IP version 2.6.0 and will be resolved in a future release.

11.1.1. Avant DDR Memory Controller IP Constraints Update

If regeneration of the Avant DDR Memory Controller IP is required, the constraints file must be manually overridden to ensure proper configuration.

To update the constraints for the Avant DDR Memory Controller IP, follow these steps:

1. Locate the source constraints file.
   - Path: <my_work_dir>\prebuilt_images\mc_constraint.sdc
2. Modify the Target Constraints File
   - Open the following file in your work directory:
     <my_work_dir>\avant_riscv_rx_rd\lib\latticesemi.com\ip\lpddr4\2.6.0\constraints\constraint.sdc
   - Scroll to line 252
   - Delete all lines from line 252 to the end of the file
   - Paste the contents of mc_constraint.sdc into the file starting at line 252
   - Save the changes

11.1.2. Avant DDR Memory Controller IP Driver Header File Update

If a BSP update is required, follow these steps after completing section 9.2.2. software flow:

1. Locate the driver header file
   - Identify the file named *ddr_mc_avant.h* in the following project paths:
     bootloader_sw\src\bsp\driver\ddr_mc
     freertos_app_sw\src\bsp\driver\ddr_mc
2. Modify the Macro Definition
   - In both copies of *ddr_mc_avant.h*, locate the following line 83:
     #define DDR_MC_DONE_BITS (0x0000007D)
   - Replace it with:
     #define DDR_MC_DONE_BITS (0x0000007F)
3. Save and Rebuild
   - Save the changes in both projects
   - Rebuild the software projects to apply the updated driver configuration

12. Resource Utilization

Table 12.1 shows the resource utilization for CertusPro-NX devices and Table 12.2 shows the data for Lattice Avant devices.

Note: These values are for reference only; actual usage may vary.

Table 12.1. 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	34,588/79,872	21,739/80,769	84/208	71/299

Table 12.2. 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	31,581/397,440	22,320/399,141	59/990	75/567

References

- RISC-V RX CPU IP - Lattice Propel Builder 2025.1 (FPGA-IPUG-02280)
- LPDDR4 Memory Controller IP Core for Nexus Devices User Guide (FPGA-IPUG-02127)
- DDR Memory Controller IP Core User Guide (FPGA-IPUG-02208)
- AXI4 Interconnect IP Core User Guide (FPGA-IPUG-02196)
- SPI Flash Memory Controller IP Core User Guide (FPGA-IPUG-02134)
- Lattice Propel 2025.1 Builder User Guide (FPGA-UG-02235)
- Lattice Propel 2025.1 SDK User Guide (FPGA-UG-02234)
- 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 and Schematics (FPGA-EB-02046)
- Avant-E Evaluation Board - User Guide and Schematics (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 1.1, August 2025

Section	Change Summary
All	• Minor editorial fixes. • Removed all instances of *System Memory* and replaced with *Tightly Coupled Memory*.
Abbreviations in This Document	• Replaced *Acronyms* with *Abbreviations* and updated section contents.

Introduction	• Updated Table 1.1. Reference Design Summary.
	  • Updated Lattice Radiant Software version to *2025.1 SP1*
	  • Updated Lattice Propel Builder version to *2025.1 SP1*
	  • Updated RISC-V RX version to *2.5.0*
	  • Updated LPDDR4 Memory Controller for Nexus Devices version to *2.3.1*
	  • Updated DDR Memory Controller version to *2.4.0* (for Avant devices)
	  • Added SPI Flash Memory Controller version *2.0.0*
	  • Added *Tightly Coupled Memory 1.5.0*
	  • Updated AXI4 Interconnect version to *2.1.0*
	  • Added *AXI4 to APB Bridge 1.3.0*
	  • Updated AXI4 to AHB-Lite Bridge version to *1.3.0*
	  • Added *APB Interconnect 1.2.1*
	  • Updated GPIO version to *1.7.0*
	  • Removed *System Memory 2.0.0*
	• Removed Lattice HW USBN 2B cable (for Lattice Avant device)

Functional Description	• Replaced *System Memory* block diagram with *Tightly Coupled Memory* block diagram for Figure 3.1. Reference Design Top Level Block Diagram.
	• Removed from the following from the design components bullet:
	  • *System Memory* — replaced by *Tightly Coupled Memory*
	  • *AXI to APB Bridge* — replaced by *AXI4 to APB Bridge*
	  • *AXI to AHB-Lite Bridge* — replaced by *AXI4 to AHB-Lite Bridge*
	• Replaced *System Memory* block diagram with *Tightly Coupled Memory* block diagram for Figure 3.2. Clocking Block Diagram – CertusPro-NX Devices.
	• Replaced *System Memory* block diagram with *Tightly Coupled Memory* block diagram for Figure 3.3. Clocking Block Diagram - Lattice Avant Devices.
	• Updated block diagram of Figure 3.4. Reset Scheme Block Diagram.

IP Configuration and Parameter Description	• Removed *System Memory* from the bullet lists of IPs.
	• Added the following in the bullet lists of IPs.
	  • *DDR Memory Controller (for Avant devices)*
	  • *SPI Flash Memory Controller*
	  • *Tightly Coupled Memory*
	  • *AXI4 to APB Bridge*
	  • *AXI4 to AHB-Lite Bridge*
	  • *APB Interconnect*
	  • *GPIO*
	• Updated screenshot of Figure 4.1. RISC-V RX CPU IP Configuration to reflect the RISC-V RX version *2.5.0*.
	• Updated Table 4.1. RISC-V RX IP Configuration to reflect Tightly Coupled Memory IP configuration values.
	• Updated screenshot of Figure 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices to reflect the RISC-V RX version *2.5.0*.
	• Updated Table 4.2. LPDDR4 Memory Controller IP Configuration for CertusPro-NX Devices.
	  • Added a *General Tab*
	  • Added *Number of Ranks*
	  • Added *Number of DDR Clocks*

Section	Change Summary
	• Added *Read Ordering Queues* • Removed *Write/Read Ordering Queues* — replaced by *Write Ordering Queues* • Updated DDR Density (per Channel) value from *8 Gb* to *4 Gb* • Updated screenshot of Figure 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices to reflect the DDR Memory Controller version *2.4.0*. • Updated Table 4.3. LPDDR4 Memory Controller IP Configuration for Lattice Avant Devices. • Added *General Tab* • Added *Interface Type* • Added *Number of Ranks* • Added *Maximum Burst Length* • Added *Write Ordering Queues* • Removed *Write/Read Ordering Queues* — replaced by *Read Ordering Queues* • Updated I/O Buffer Type value from *LVSTL_I* to *LVSTL_II* • Updated DDR Density (per Channel) value from *8 Gb* to *4 Gb* • Updated DDR Bus Width value from *16 Gb* to *32 Gb* and updated description • Updated Data Width value from *128* to *256* • Updated *description* of Number of DDR Clocks • Added SPI Flash Memory Controller section. • Added Tightly Coupled Memory section. • Updated screenshot of Figure 4.5. SPI Flash Memory Controller IP Configuration to reflect the AXI4 Interconnect version *2.1.0.* • Removed *System Memory* section.
RISCV-RX Software Flow	Replaced *System Memory* block diagram with *Tightly Coupled Memory* block diagram for Figure 6.1. Boot up Sequence.
Implementing the Reference Design on the Board	• Removed from the Setting up the Lattice Avant-E Evaluation Board section the note, *The Lattice USB Programming Cable HW USBN-2B is required to program the SPI flash on the board*. • Updated Figure 7.2. Lattice Avant-E Evaluation Board screenshot to *Rev D*. • Updated the *Set-up procedure of the Lattice-E board* in the Setting up the Lattice Avant-E Evaluation Board section. • Updated the screenshot of Figure 7.3. Tera Term New Connection, Figure 7.4. Tera Term Serial Port Setup and Connection, Figure 7.5. Windows Start Menu > Radiant Programmer, and Figure 7.9. RISC-V RX and LPDDR4 Reference Design Results. • Updated *Flash Device* of the Lattice Avant-E Evaluation Board from *Macronix* to *Winbond* in Table 7.1. Programming Files Selection. • Removed *HW-USB-2B* from the procedure *To program the bitstream into the device* in the Programming the FPGA Bitstream.
Compiling the Reference Design	• Updated the Lattice Propel Builder version to *2025.1*. • Updated the Radiant Software version to *2025.1*. • Updated the Lattice Propel Software Development Kit (SDK) version to *2025.1*. • Updated the screenshot of Figure 8.1. Lattice Propel Builder Schematic view, Figure 8.2. Windows Start Menu > Lattice Propel Builder, Figure 8.3. TCL Console – No Error for Validate, Figure 8.4. TCL Console – No Error for Generate, Figure 8.5. Windows Start Menu > Radiant Software, Figure 8.7. Windows Start Menu > Lattice Propel , and Figure 8.12. Console. • Added *warning messages* in the procedure, *To generate the RISC-V CPU system*. • Removed from the Synthesizing RTL Files and Generating the Bitstream using the Lattice Radiant Software section the 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*. • Removed *Floor Planning Constraints* section. • Removed *Pre-initializing System Memory* section and replaced with Pre-initializing Tightly Coupled Memory section.

Section	Change Summary
Customizing the Reference Design	• Updated the screenshot of Figure 9.1. lpddr4_inst Block. • Updated the procedure, *To change the application image load address* in Changing the Application Image Load Address section. • Updated the procedure, *To modify linker script* in Updating Linker Script to LPDDR4 Memory Address section. • Removed *Check Correct Flash Device for Lattice Avant Devices* section.
Known Issues	Added this section.

Revision 1.0, November 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com