Golden System Reference Design and Demo User Guide v1.0 for Lattice Avant-E Devices

Lattice Propel 2024.1

Lattice Radiant 2024.1.1

Reference Design

FPGA-RD-02296-1.1

December 2025

Disclaimers

Inclusive Language

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

Contents

Contents 3
Abbreviations in This Document 9
1. Introduction 10
1.1. Overview of the System 10
1.2. Quick Facts 10
1.3. Features 11
1.4. Naming Conventions 11
1.4.1. Nomenclature 11
1.4.2. Signal Names 11
2. Functional Description 12
2.1. System Architecture Overview 12
2.1.1. Avant-AT-E 12
2.2. Clocking 13
2.2.1. Clocking Overview for Avant-AT-E 13
2.3. Reset Scheme 14
2.4. IP Configurations 14
2.4.1. RISC-V RX CPU Core 14
2.4.2. LPDDR4 Memory Controller 17
2.4.3. QSPI Flash Controller 18
2.4.4. Tri-Speed Ethernet MAC 18
2.4.5. Scatter-Gather DMA 19
2.4.6. UART 20
2.4.7. GPIO 20
2.4.8. Multi-Boot Configuration Module 21
2.4.9. System Memory 21
2.5. System Level Interfaces 22
2.6. SoC Memory/Address Map 22
2.7. Functional Operation 23
3. Signal Description 24
4. Software Components 25
4.1. Primary and Golden Bootloader 25
4.2. Primary and Golden Application 25
5. Design Constraints 26
5.1. I/O Constraints 26
6. Resource Utilization 31
7. Demo User Guide 32
7.1. Boot-Up Sequence 32
7.2. Prerequisites 34
7.2.1. Software Requirements 34
7.2.2. Avant-AT-E Requirements 34
7.3. Implementing the GHRD/GSRD Demo 36
7.3.1. Setting up the UART Terminal 36
7.3.2. Setting up the Non-Volatile Memory Register 37
7.3.3. Programming the Standalone Golden or Primary GSRD Bitstream and Application Software 37
7.3.4. Programming the Golden, Primary Software and MCS file 49
8. Compiling the Reference Design 61
8.1. Building the Bootloader and FreeRTOS Binary Files Using Lattice Propel SDK 61
8.2. Validating and Generating the GSRD Design using Propel Builder 76
8.3. Synthesizing the RTL Files and Generating the Bitstream using Lattice Radiant 80
8.4. Generating the Multi-Boot MCS File 85
9. Customizing the IP in the GSRD Design 90
9.1. Changing QSPI Flash Controller User Interface Parameters 90

References 94
Technical Support Assistance 95
Revision History 96

Figures

Figure 2.1. GHRD Architecture on Avant-AT-E....12
Figure 2.2. Clocking Structure for Avant-AT-E GSRD 13
Figure 2.3. Reset Structure for Avant-AT-E GSRD....14
Figure 2.4. CPU Configuration – General....15
Figure 2.5. CPU Configuration – Debug 15
Figure 2.6. CPU Configuration – Buses 16
Figure 2.7. CPU Configuration – Interrupt....16
Figure 2.8. CPU Configuration – UART....17
Figure 2.9. LPDDR4 MC Configuration....17
Figure 2.10. QSPI Flash Controller Configuration 18
Figure 2.11. TSE MAC Configuration....19
Figure 2.12. SGDMA Configuration....19
Figure 2.13. UART Configuration 20
Figure 2.14. GPIO Configuration....20
Figure 2.15. Multi-Boot Configuration....21
Figure 2.16. System Memory Configuration....21
Figure 6.1. GSRD Resource Utilization on Avant-AT-E70 31
Figure 7.1. GSRD Boot-Up Sequence 33
Figure 7.2 .Avant-AT-E70 Evaluation Board....34
Figure 7.3.Ethernet PHY FMC Card....35
Figure 7.4. Connections and Buttons needed for Demonstration....35
Figure 7.5. UART Terminal Icon on Propel SDK Window 36
Figure 7.6. UART Launch Terminal Window 37
Figure 7.7. Device Manager Window on PC 37
Figure 7.8. Launch Radiant Programmer from Windows Start....38
Figure 7.9. Radiant Programmer Start Window 38
Figure 7.10. Radiant Programmer .xcf Window 38
Figure 7.11. Scan Device Icon on Radiant Programmer....39
Figure 7.12. Select Device for Programming 39
Figure 7.13. Device Selected for Programmer....39
Figure 7.14. Select the Target Memory for Programming....39
Figure 7.15. Device Properties to Erase the SPI Flash 40
Figure 7.16. Program Button to Program the SPI Flash....40
Figure 7.17. Erase Only Operation for CRAM Programming....41
Figure 7.18. One-Time Programmable Control NV Register1....41
Figure 7.19. Settings to Select Chip Value 42
Figure 7.20. Device Properties to Program the SPI Flash 42
Figure 7.21. Cable Settings for device programming....43
Figure 7.22. Radiant Programmer Console Output after Programming the SPI Flash....43
Figure 7.23. Device Properties to Program the FPGA Bitstream in CRAM 44
Figure 7.24. Radiant Programmer Console Output after Bitstream is Programmed....44
Figure 7.25. SW1 Reset Button....45
Figure 7.26. Golden GSRD - Output on UART Terminal for Bootloader and FreeRTOS Start 46
Figure 7.27. Golden GSRD - Output on UART Terminal for FreeRTOS Running....47
Figure 7.28. Primary GSRD Bootloader– Output on UART Terminal 48
Figure 7.29. Primary GSRD FreeRTOS– Output on UART Terminal....49
Figure 7.30.Device Properties window to setup MCS programming file....50
Figure 7.31. UART Terminal Output after Power-Cycling Board with MCS and Binaries Programmed 51
Figure 7.32. Switches to Golden GSRD upon SW2 PROGRAMN Button 52
Figure 7.33. Device Properties settings to program the Golden Application 53
Figure 7.34. Radiant Programmer Console Output After Programming 53
Figure 7.35. Primary GSRD – UART Output after SW1 Reset where CRC Matched 54

Figure 7.36. UART Output Switching to Golden GSRD where CRC Mis-matched Intentionally......55
Figure 7.37. Device Properties for Programming Golden App Binary with CRC......56
Figure 7.38. Radiant Programmer Console Output After Successful Programming......56
Figure 7.39. Device Properties Window for Primary App Binary without CRC......57
Figure 7.40. Radiant Programmer Console Output After Successful Programming......57
Figure 7.41. Primary GSRD without CRC Fails – UART Terminal......58
Figure 7.42. Automatically Jumps to Boot-up Golden GSRD......59
Figure 7.43. UART Terminal Output - After Pressing SW2 PROGRAMN Button......60
Figure 8.1. Propel Builder Launcher......61
Figure 8.2. Provide Name for Workspace Directory......61
Figure 8.3. Creating Lattice SoC Design Project......62
Figure 8.4. SoC Project Window......62
Figure 8.5. Launched SoC in Propel Builder......63
Figure 8.6. Validate Design in Propel Builder......63
Figure 8.7. TCL Console Output after Validating Design......63
Figure 8.8. Generate in Propel Builder......64
Figure 8.9. TCL Console Output after Generating Design.......64
Figure 8.10. sys_env.xml File Created......64
Figure 8.11. Creating Lattice C/C++ Project for Bootloader......64
Figure 8.12. Bootloader C/C++ Selection......65
Figure 8.13. C/C++ Lattice Toolchain Setting......66
Figure 8.14. Bootloader C Project Created......66
Figure 8.15. Creating C/C++ Project for FreeRTOS......67
Figure 8.16. FreeRTOS C/C++ Selection......67
Figure 8.17. FreeRTOS Project Created......68
Figure 8.18. Copy the CRC and NoCRC Add files......69
Figure 8.19. Paste in FreeRTOS C project......69
Figure 8.20. Copied Text Files......70
Figure 8.21. Updated Primary Address......70
Figure 8.22. Enable fw_softreset() Function......70
Figure 8.23. Update crc_add_debug.txt......71
Figure 8.24. Update noncrc_add.txt......71
Figure 8.25. Open Linker.Ld File......71
Figure 8.26. Update Linker.ld File......72
Figure 8.27. Workspace......72
Figure 8.28. Build Bootloader Project......72
Figure 8.29. Bootloader Build Project Console Output......73
Figure 8.30. Bootloader Binary Created......73
Figure 8.31. Properties......74
Figure 8.32. Select Release as Configuration......74
Figure 8.33. Set Release as Active Configuration......74
Figure 8.34. Adding Post Build Step for FreeRTOS Application CRC Binary Append......75
Figure 8.35. Build c_primary_app C/C++ Project......75
Figure 8.36. FreeRTOS App Build Project Console Output......76
Figure 8.37. FreeRTOS App Binaries Created with CRC and without CRC......76
Figure 8.38. Open Propel Builder......76
Figure 8.39. Propel Builder SoC Project......77
Figure 8.40. System Memory Highlighted......77
Figure 8.41. System Memory IP Configuration......78
Figure 8.42. Initialize System Memory with c_primary_bootloader.mem......78
Figure 8.43. Click Generate......79
Figure 8.44.Bootloader file updated in System Memory......79
Figure 8.45. Click Finish......80
Figure 8.46. Run Radiant Icon......80

Figure 8.47. Lattice Radiant 2024.1.1 Window 81
Figure 8.48. Constraint Files Folder 81
Figure 8.49. Add Existing Pre-Synthesis Constraint 81
Figure 8.50. Select clock_constraint.sdc File 82
Figure 8.51. Select soc_board_constraint.pdc File 82
Figure 8.52. Pre and Post-Synthesis Constraint Files Added 83
Figure 8.53. Update the Strategy....................83
Figure 8.54. Strategy used for Avant GSRD Testing 84
Figure 8.55. Generating the Bit File 84
Figure 8.56. Successful Radiant Flow and Bitstream Generation 84
Figure 8.57. Launch Radiant Programmer from Windows Start....................85
Figure 8.58. Radiant Programmer Getting Started Window 85
Figure 8.59. Error if No HW Board is Connected 85
Figure 8.60. Open Deployment Tool from Radiant Programmer 86
Figure 8.61. Deployment Tool Start Window 86
Figure 8.62. Options for Creating New Deployment 86
Figure 8.63. External Memory Step 1 of 4: Select Input Files....................87
Figure 8.64. External Memory Step 2 of 4: Select Options....................87
Figure 8.65. External Memory Step 2 of 4: Multi-Boot....................88
Figure 8.66. External Memory Step 3 of 4: Select Output File(s) 88
Figure 8.67. External Memory Step 4 of 4: General Development....................89
Figure 8.68. MCS File Generated Successfully 89
Figure 9.1. IP Block 90
Figure 9.2. Customize IP 90
Figure 9.3. TCL Console Output After Desired Customization....................91
Figure 9.4. Validate Design again is any IP is updated....................91
Figure 9.5. Generate again is any IP is updated....................91
Figure 9.6. New sys_env.xml File Generated....................91
Figure 9.7. Update Lattice C/C++ Project 91
Figure 9.8. Update System Dialog Window 92
Figure 9.9. New IP version Details 92
Figure 9.10. Generate BSP and Toolchain Parameters....................93
Figure 9.11. Confirm the Changes 93
Figure 9.12. Confirm the Configuration Changes in sys_platform.h....................93

Tables

Table 1.1. Summary of the System 10
Table 2.1. IP Versions 12
Table 2.2. Reset Scheme 14
Table 2.3. System Level Interfaces 22
Table 2.4. Address Map of GHRD 22
Table 3.1. Top-level I/O 24
Table 6.1. GSRD Total Resource Utilization 31
Table 7.1. Executable Files for Winbond Flash 36

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHBL	Advanced High-performance Bus-Lite
APB	Advanced Peripheral Bus
API	Application Programming Interface
AXI	Advanced eXtensible Interface
AXI4-Lite	Advanced eXtensible Interface-Lite
CPU	Central Processing Unit
DDR	Double Data Rate
FIFO	First-In-First-Out
FMC	FPGA Mezzanine Card
GPIO	General Purpose Input/Output
ISR	Interrupt Service Routines
LMMI	Lattice Memory Mapped Interface
LPDDR4	Low Power Double Data Rate Generation 4
MPMC	Multi-Port Memory Controller
QSPI	Quad Serial Peripheral Interface
RISC-V	Reduced Instruction Set Computer-V
RTL	Register Transfer Level
SFP	Small Form-Factor Pluggable
SGDMA	Scatter-Gather Direct Memory Access
SGMII PCS and GbE	Serial Gigabit Media Independent Interface
SoC	System on Chip
TSE MAC	Tri - Speed Ethernet Media Access Controller
UART	Universal Asynchronous Receiver-Transmitter

1. Introduction

1.1. Overview of the System

The Lattice FPGA-based GSRD SoC design presented herein is aimed at providing a versatile and efficient platform for embedded applications requiring high-performance computing, memory access, data transfer capabilities, and network communication. By integrating various components onto a single FPGA chip, this design offers flexibility, scalability, and cost-effectiveness for a wide range of applications.

The Lattice Golden Hardware Reference Design (GHRD) is a System-on-Chip (SoC) that can be used as a baseline design to create FPGA applications as per the user requirements. It is a RISC-V based design that interacts with various Lattice Soft-IPs and peripherals such as GPIO, UART, I2C, Timer, Tri-Speed Ethernet MAC (TSE MAC), QSPI Flash Controller, Scatter-Gather DMA (SGDMA) and LPDDR4 Memory Controller. All these building blocks are connected via industry standard protocols such as AXI4-Full, AXI-Steam for data transfers and AXI4-Lite, APB for control.

The Lattice Golden Software Reference Design (GSRD) is a comprehensive embedded system which incorporates drivers and relevant firmware needed to operate various design components. Free-RTOS and First Stage Bootloader (FSBL) is built on RISC-V RX CPU Core. The primary function of FSBL is to initialize the hardware blocks in the design using the respective IP drivers and ensure the integrity of the firmware by performing CRC check.

GHRD and GSRD are integrated together to establish a complete system for which relevant binaries and executables are generated by Lattice SW tools such as Propel SDK, Propel Builder and Radiant to program the FPGA Hardware.

As a part of multi-boot demo, the executables folder comprises of compatible binaries and executable images, i.e., FPGA Bitstream (.bit) and Firmware Binary (.bin) for both Primary and Golden GSRD projects. The only difference is that the Primary bitstream contains the code for multi-boot enablement and Golden bitstream does not enable multi-boot.

The GSRD for the Avant-AT-E device is developed and tested with the following Lattice Propel™ and Lattice Radiant™ software versions.

Note:

1. The Golden System Reference Design and Demonstration for Avant-E version 2.0 is available in the GHRD/GSRD Reference Design and GHRD/GSRD Demonstration web pages.

1.2. Quick Facts

Table 1.1. Summary of the System

SoC Requirements	Supported FPGA Family	Lattice Avant™-AT-E
	SoC Version	1.0
FPGA Device(s)	Targeted Devices	LAV-AT-E70
	Supported User Interface	AXI4-Full, AXI4-Lite, AXI-Stream, APB
Design Tool Support	Lattice Implementation	Lattice Propel Software 2024.1 Lattice Radiant Software 2024.1.1
	Synthesis	Synopsys® Synplify Pro®

1.3. Features

The key features of the system include:

- FPGA device supported in this document is Lattice Avant-AT-E
- RISC-V RX CPU Core, SGDMA, TSE MAC, LPDDR4, and QSPI Flash Controller over AXI4 Interface
- Low-speed peripherals like GPIO and UART
- Bootloader, Primary and Golden FreeRTOS Application
- Application Firmware’s CRC check by function implemented in RISC-V RX bootloader code
- FPGA bitstream CRC check done by FPGA Configuration Engine
- Manual and Automatic Multi-Boot capability
- FreeRTOS Application Software is run on LPDDR4 MC
- 256-bit AXI4 LPDDR4 data width to support 32-bit data to DDR at 800 MHz operating frequency
- AXI4/AXI4-Lite Peripherals at 140 MHz and APB Peripherals at 100 MHz
- 1 Gbps Ethernet throughput through RGMII support at 125 MHz

1.4. Naming Conventions

1.4.1. Nomenclature

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

1.4.2. Signal Names

Signal names that end with:

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

2. Functional Description

The GSRD/GHRD SoC architecture comprises of a RISC-V CPU core, LPDDR4 Memory Controller, SGDMA controller, QSPI Flash Controller, and a 1 Gbps TSE MAC, interconnected through a combination of high-speed and low-speed bus fabrics such as AXI4 Interconnect and APB Interconnect. This architecture enables seamless communication and data exchange between the components, facilitating efficient operation and system performance.

2.1. System Architecture Overview

2.1.1. Avant-AT-E

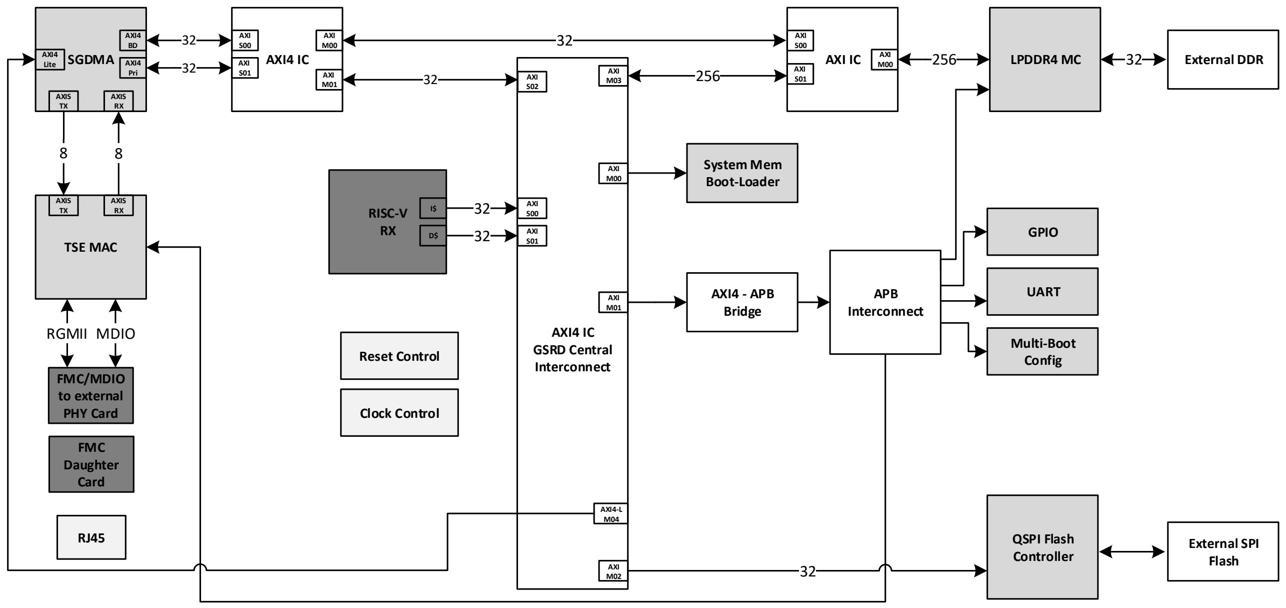


Figure 2.1. GHRD Architecture on Avant-AT-E

The design includes the following components:

Table 2.1. IP Versions

Soft-IP	IP Version
RISC-V RX Processor	v2.4.0
LPDDR4 Memory Controller	v2.2.1
System Memory	v.2.2.0
QSPI Flash Controller	v1.2.0
General Purpose I/O GPIO	v1.6.2
UART	v1.3.0
AXI4 Interconnect	v2.0.1
APB Interconnect	v1.2.1
AXI to APB Bridge	v1.2.0
Tri-Speed Ethernet MAC	v1.6.0
SGDMA	v2.2.0
AXI4 Register Slice	v1.0.0
Multi-Boot Configuration	V1.0.0
PLL	v1.9.0
OSC	V2.1.0

Each component in the block diagram is instantiated using the IP in Propel Builder. The IP features and parameters are described in the IP Configurations section.

The signals in each interface are described in the Signal Description section.

2.2. Clocking

This section describes the reference design clocking scheme. There are some minor differences between the clocking scheme and the Lattice Avant-AT-E device. Refer to the Clocking Overview for Avant-AT-E section.

2.2.1. Clocking Overview for Avant-AT-E

There are five clocks in GSRD Avant-AT-E Architecture listed as follows:

- External PLL Reference Clock for LPDDR4 is 100 MHz
- External Reference Clock for TSE MAC is 125 MHz
- On-Chip Oscillator Clock is 10 MHz for TSE MAC MDIO Interface
- External Oscillator Clock is 125 MHz as Input Reference Clock to the PLL
- RISC-V Low Frequency Realtime Clock derived from FPGA PLL is 16.384 MHz
- Internal System Clock derived from FPGA PLL is 125 MHz
- All AXI Peripherals are operating at 140 MHz
- All APB Peripherals are operating at 100 MHz

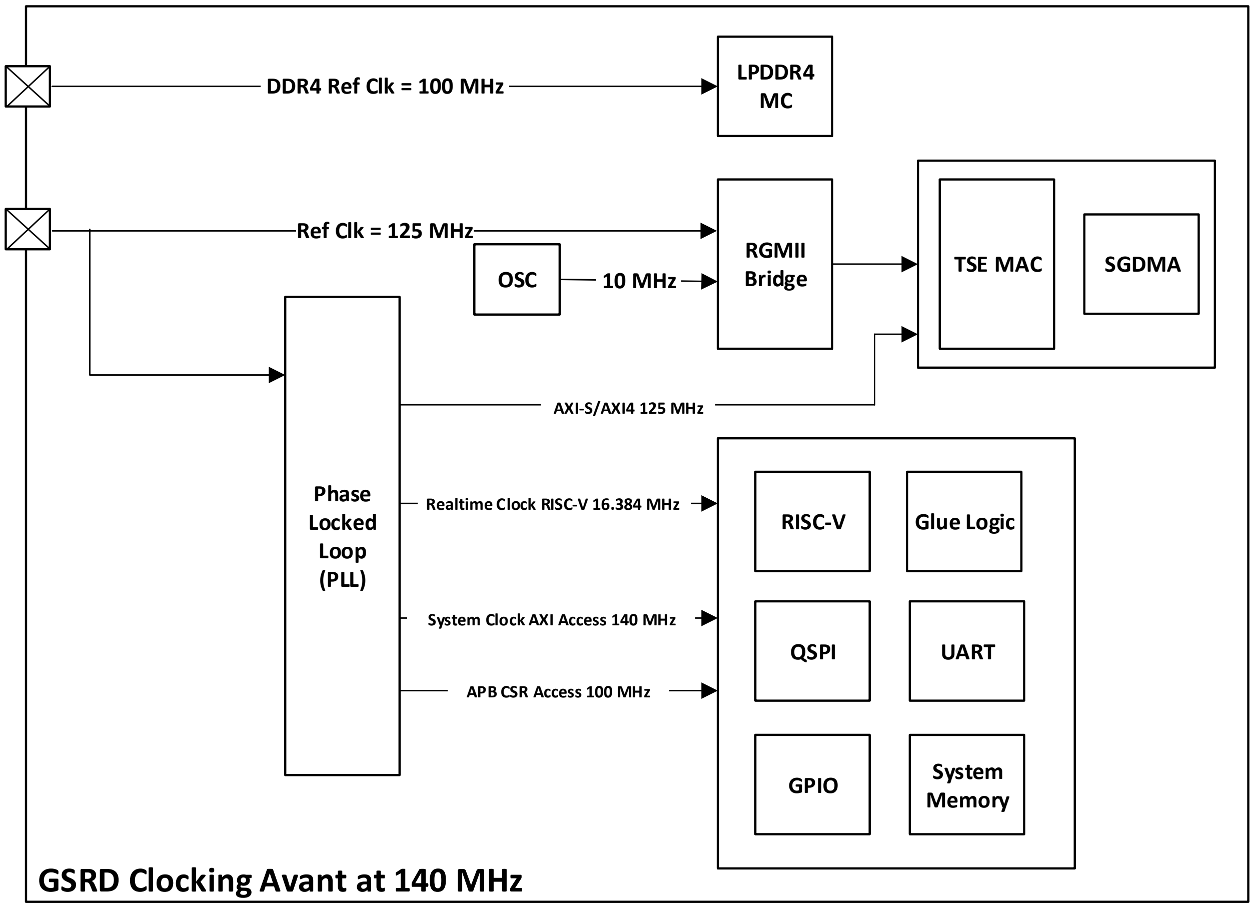


Figure 2.2. Clocking Structure for Avant-AT-E GSRD

2.3. Reset Scheme

There are two resets in the entire design:

- External Asynchronous Reset which is controlled by a push button
- Synchronous Reset for entire system is generated from RISC-V

Table 2.2. Reset Scheme

Reset Signal	Source	Destination	Description
rstn_i	Board Pushbutton	PLL reset input and Synchronizer	Reset the PLL
cpu_rstn_i	PLL Lock and synced rstn_i	RISC-V RX reset input	Release RISC-V RX from reset pin, synchronizer and PLL lock
system_rstn_o	RISC-V RX output	All components in system	RISC-V RX output reset provides reset to all components in the design. This also triggers the reset during CPU OCD debugging mode.

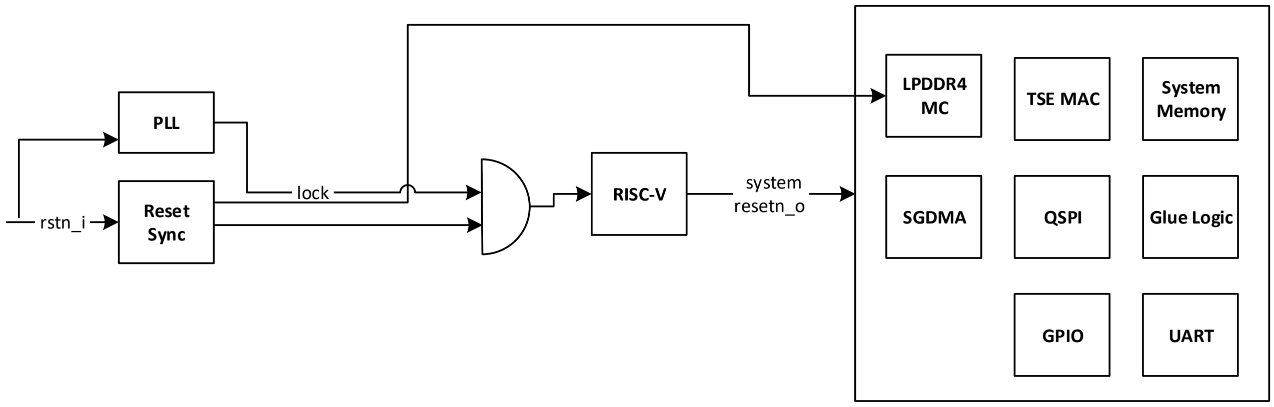


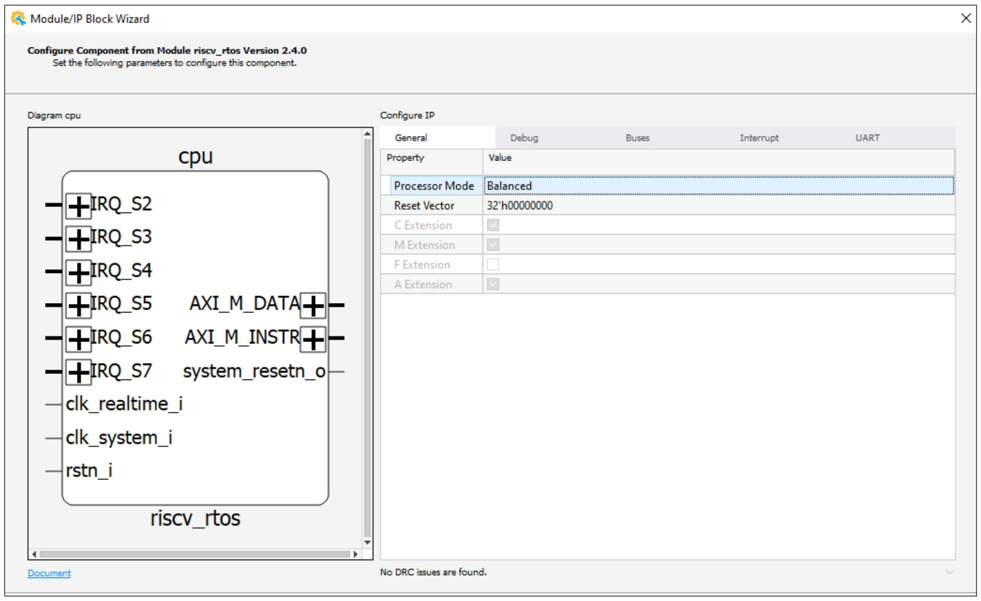
Figure 2.3. Reset Structure for Avant-AT-E GSRD

2.4. IP Configurations

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

2.4.1. RISC-V RX CPU Core

For more information about the IP core including register map information, refer to RISC-V RX CPU IP Core User Guide (FPGA-IPUG-02241). The RISC-V RX CPU IP has AXI-based instruction and data ports. The instruction ports are connected to the memory that contains the bootloader software or the FreeRTOS application software for CPU execution. The data port is connected to the memory and peripherals for control.

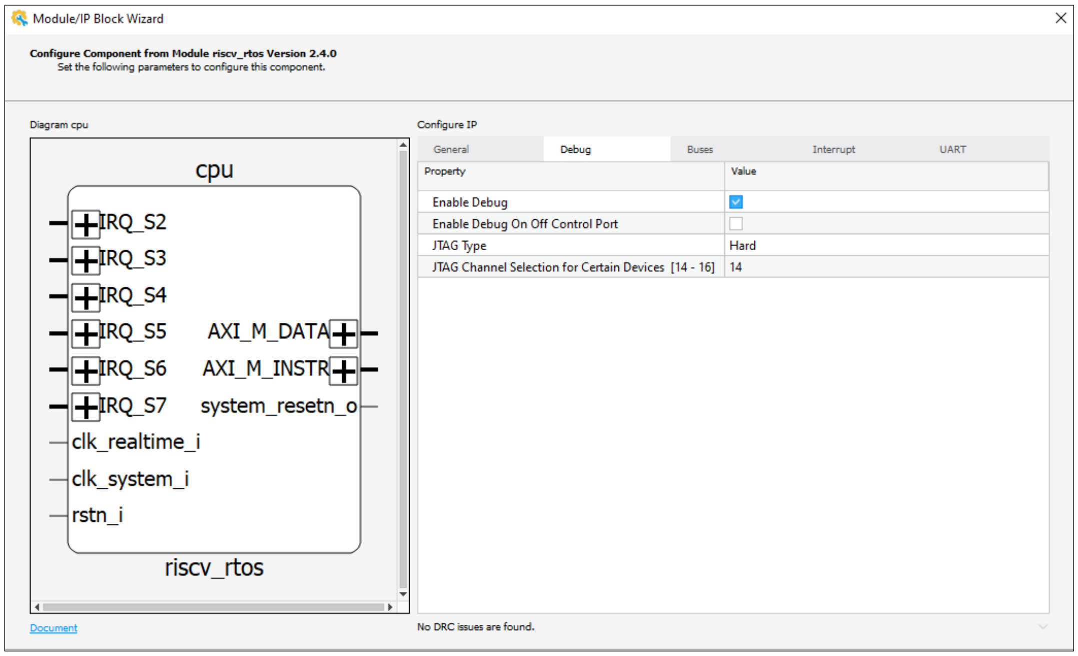


Configure Component from Module riscv_rtos Version 2.4.0
Set the following parameters to configure this component.

Property	Value
Processor Mode	Balanced
Reset Vector	32'h00000000
C Extension	✓
M Extension	✓
F Extension	
A Extension	✓

No DRC issues are found.

Figure 2.4. CPU Configuration – General

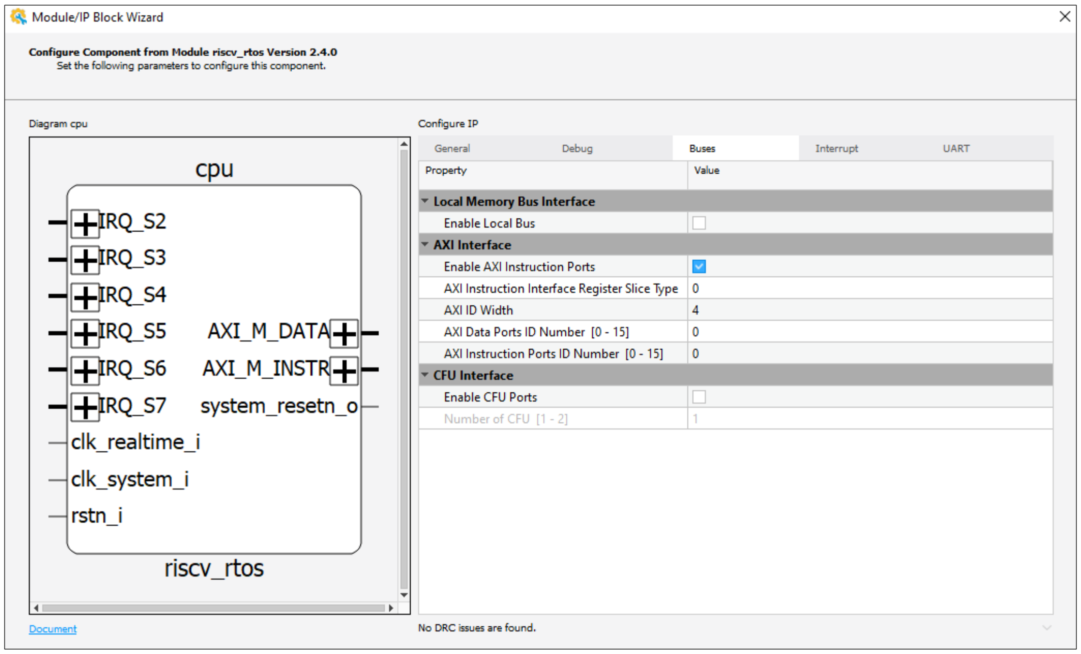


Configure Component from Module riscv_rtos Version 2.4.0
Set the following parameters to configure this component.

Property	Value
Enable Debug	✓
Enable Debug On Off Control Port	
JTAG Type	Hard
JTAG Channel Selection for Certain Devices [14 - 16]	14

No DRC issues are found.

Figure 2.5. CPU Configuration – Debug

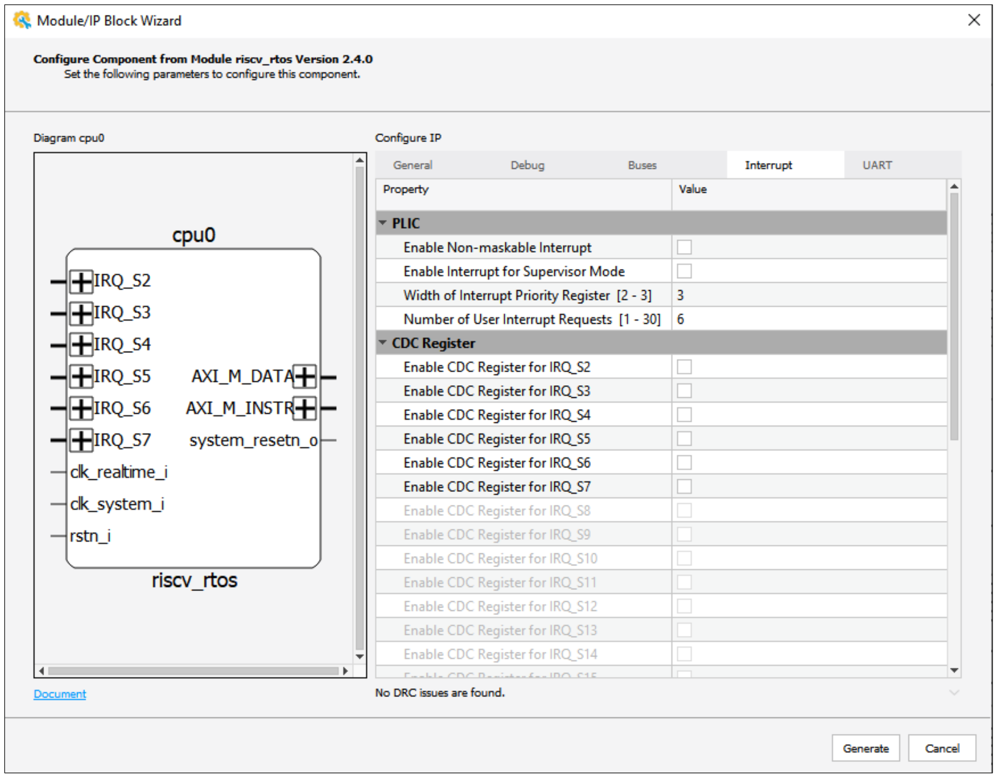


Configure Component from Module riscv_rtos Version 2.4.0
Set the following parameters to configure this component.

Property	Value
Local Memory Bus Interface	
Enable Local Bus	☐
AXI Interface	
Enable AXI Instruction Ports	✓
AXI Instruction Interface Register Slice Type	0
AXI ID Width	4
AXI Data Ports ID Number [0 - 15]	0
AXI Instruction Ports ID Number [0 - 15]	0
CFU Interface	
Enable CFU Ports	☐
Number of CFU [1 - 2]	1

No DRC issues are found.

Figure 2.6. CPU Configuration – Buses

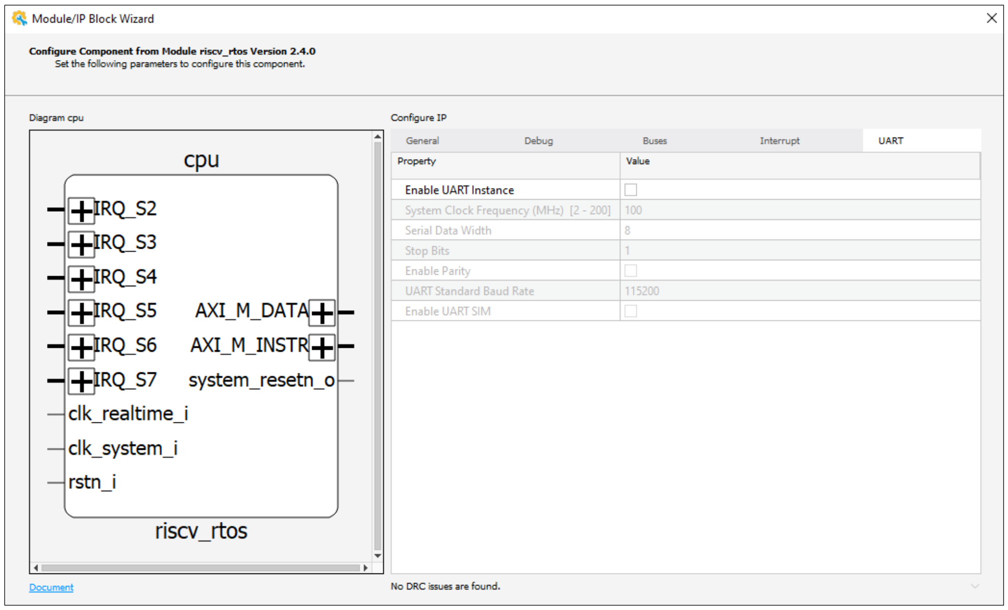


Configure Component from Module riscv_rtos Version 2.4.0
Set the following parameters to configure this component.

Property	Value
PLIC	
Enable Non-maskable Interrupt	☐
Enable Interrupt for Supervisor Mode	☐
Width of Interrupt Priority Register [2 - 3]	3
Number of User Interrupt Requests [1 - 30]	6
CDC Register	
Enable CDC Register for IRQ_S2	☐
Enable CDC Register for IRQ_S3	☐
Enable CDC Register for IRQ_S4	☐
Enable CDC Register for IRQ_S5	☐
Enable CDC Register for IRQ_S6	☐
Enable CDC Register for IRQ_S7	☐
Enable CDC Register for IRQ_S8	☐
Enable CDC Register for IRQ_S9	☐
Enable CDC Register for IRQ_S10	☐
Enable CDC Register for IRQ_S11	☐
Enable CDC Register for IRQ_S12	☐
Enable CDC Register for IRQ_S13	☐
Enable CDC Register for IRQ_S14	☐

No DRC issues are found.

Figure 2.7. CPU Configuration – Interrupt



Configure Component from Module riscv_rtos Version 2.4.0
Set the following parameters to configure this component.

Configure IP

Property	Value
Enable UART Instance	☐
System Clock Frequency (MHz) [2 - 200]	100
Serial Data Width	8
Stop Bits	1
Enable Parity	☐
UART Standard Baud Rate	115200
Enable UART SIM	☐

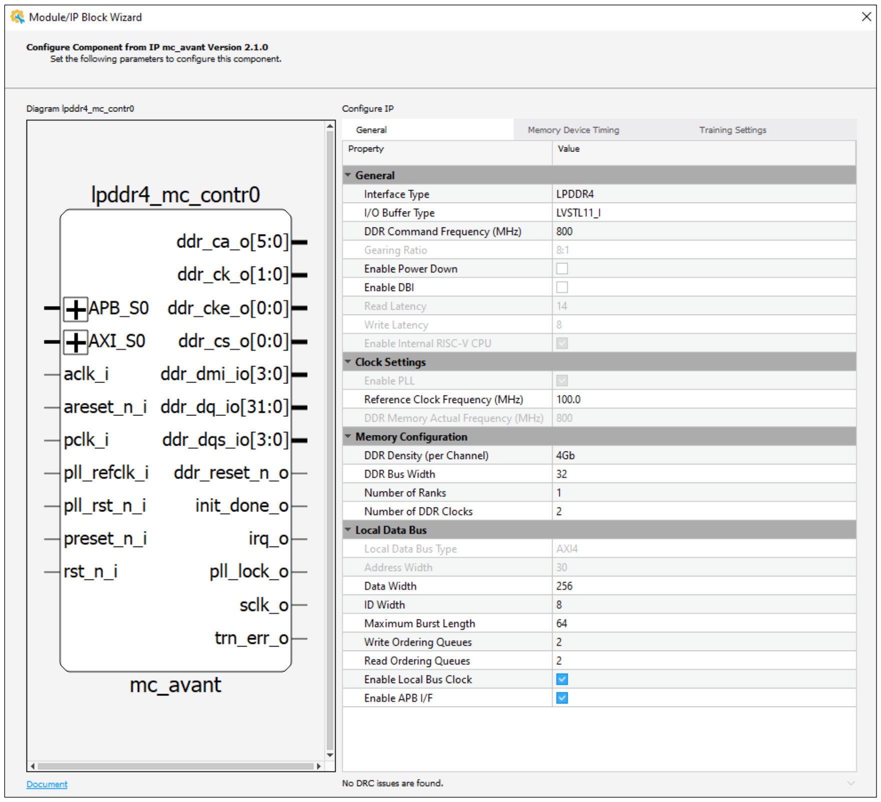
No DRC issues are found.

Figure 2.8. CPU Configuration – UART

2.4.2. LPDDR4 Memory Controller

For more information about the IP core including register map information, refer to Memory Controller IP Core for Avant Devices (FPGA-IPUG-02208).

The LPDDR4 Memory Controller IP enables access to the external LPDDR4 memory modules. The memory can be used to store CPU software code and data.



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

Configure IP

Property	Value
General	
Interface Type	LPDDR4
I/O Buffer Type	LVSTL11_I
DDR Command Frequency (MHz)	800
Gearing Ratio	8:1
Enable Power Down	☐
Enable DBI	☐
Read Latency	14
Write Latency	8
Enable Internal RISC-V CPU	✓
Clock Settings	
Enable PLL	✓
Reference Clock Frequency (MHz)	100.0
DDR Memory Actual Frequency (MHz)	800
Memory Configuration	
DDR Density (per Channel)	4Gb
DDR Bus Width	32
Number of Ranks	1
Number of DDR Clocks	2
Local Data Bus	
Local Data Bus Type	AXI4
Address Width	30
Data Width	256
ID Width	8
Maximum Burst Length	64
Write Ordering Queues	2
Read Ordering Queues	2
Enable Local Bus Clock	✓
Enable APB I/F	✓

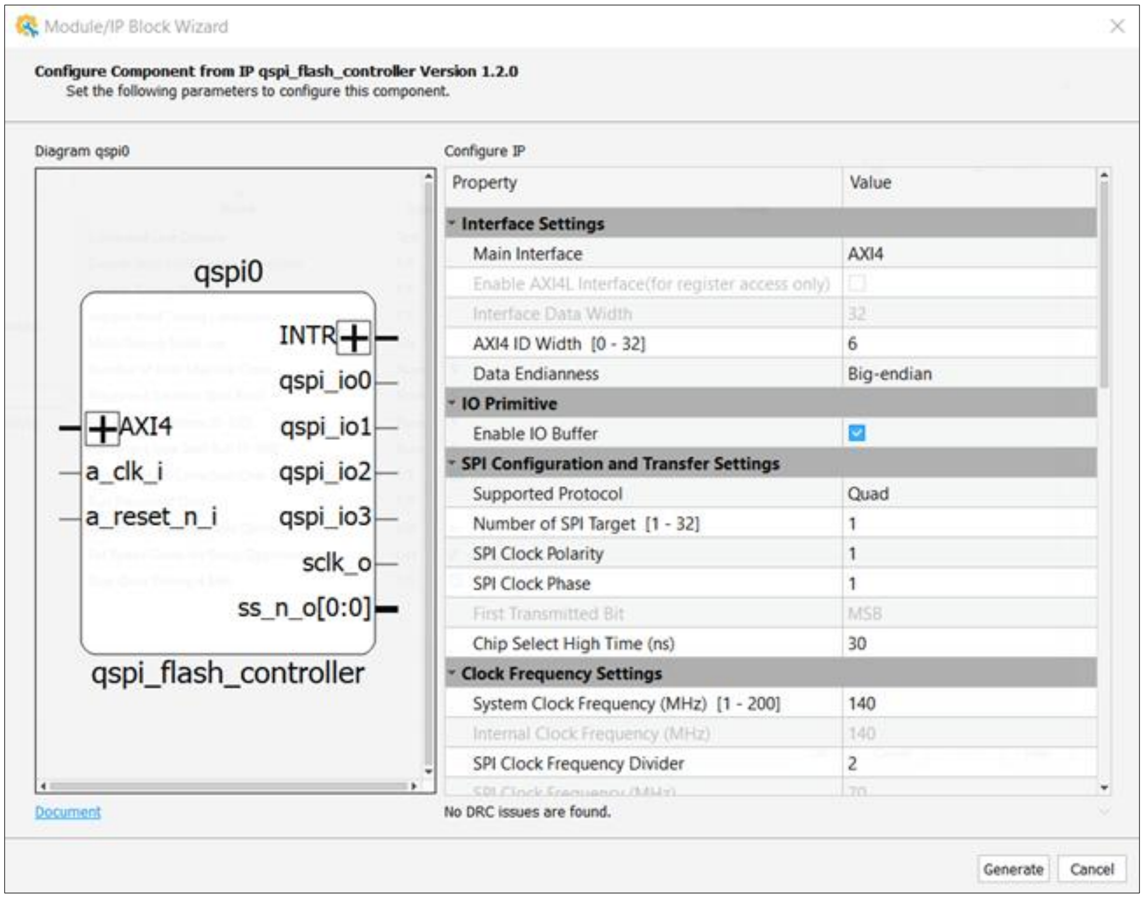
No DRC issues are found.

Figure 2.9. LPDDR4 MC Configuration

2.4.3. QSPI Flash Controller

For more information about the IP core including register map information, refer to QSPI Flash Controller IP User Guide (FPGA-IPUG-02248).

The QSPI Flash Controller is a four tri-state data line serial interface that is commonly used to store, program, erase and read SPI Flash memories. QSPI enhances the throughput of a standard SPI by four times since four bits are transferred every cycle. In GSRD, the QSPI is used to store the application software and bitstreams for both Primary and Golden systems.



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

Property	Value
Interface Settings	
Main Interface	AXI4
Enable AXI4L Interface(for register access only)	☐
Interface Data Width	32
AXI4 ID Width [0 - 32]	6
Data Endianness	Big-endian
IO Primitive	
Enable IO Buffer	✓
SPI Configuration and Transfer Settings	
Supported Protocol	Quad
Number of SPI Target [1 - 32]	1
SPI Clock Polarity	1
SPI Clock Phase	1
First Transmitted Bit	MSB
Chip Select High Time (ns)	30
Clock Frequency Settings	
System Clock Frequency (MHz) [1 - 200]	140
Internal Clock Frequency (MHz)	140
SPI Clock Frequency Divider	2
SPI Clock Frequency (MHz)	70

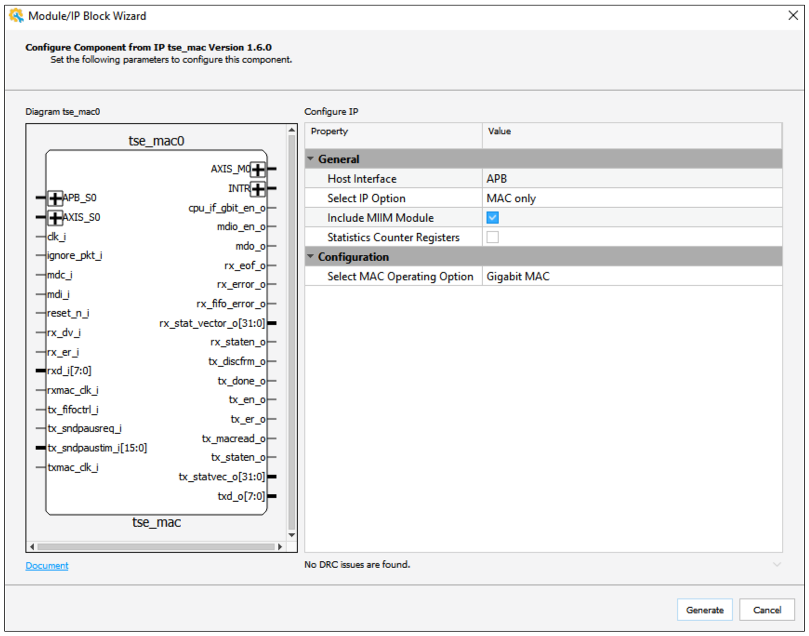
No DRC issues are found.

Figure 2.10. QSPI Flash Controller Configuration

2.4.4. Tri-Speed Ethernet MAC

For more information about the IP core including register map information, refer to Tri-Speed Ethernet MAC IP User Guide (FPGA-IPUG-02084).

The TSEMAC IP core is a 10/100/1000 Mbps network interface as per the IEEE 802.3 standard. It is complex core containing all the necessary logic, interfacing and clocking infrastructure to allow integrating an external industry-standard Ethernet PHY with an internal processor, with minimal overhead. The GSRD requires the Gigabit MAC to support 1 Gbps data-rates. The MIIM interface is used to control the external PHY control and status registers.



Configure Component from IP tse_mac Version 1.6.0
Set the following parameters to configure this component.

Configure IP

Property	Value
General	
Host Interface	APB
Select IP Option	MAC only
Include MIIM Module	✓
Statistics Counter Registers	☐
Configuration	
Select MAC Operating Option	Gigabit MAC

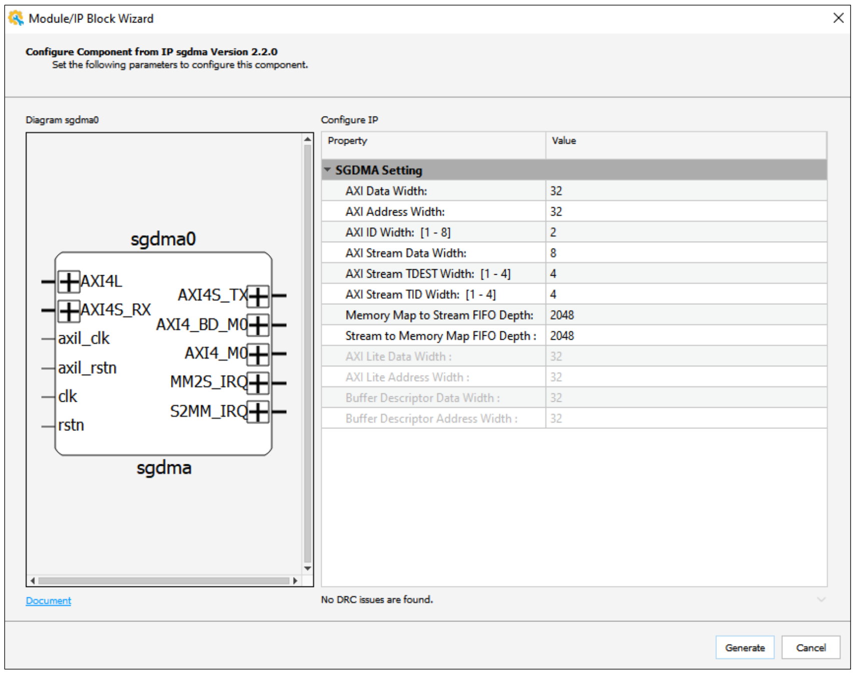
No DRC issues are found.

Figure 2.11. TSE MAC Configuration

2.4.5. Scatter-Gather DMA

For more information about the IP core including register map information, refer to SGDMA Controller IP Core (FPGA-IPUG-02131).

The SGDMA Controller IP core is to access the main memory independent of the CPU processor. It offloads processor intervention. The processor initiates transfer to SGDMA Controller and receives interrupts on completion of the transfer by the DMA engine. The core implements a configurable, AXI4-compliant DMA controller with scatter-gather capability. It also implements the AXI4-Stream interface to support stream data from TSE MAC module. The AXI4-Lite CSR interface is used to configure the control and status registers by the RISC-V CPU.



Configure Component from IP sgdma Version 2.2.0
Set the following parameters to configure this component.

Configure IP

Property	Value
SGDMA Setting	
AXI Data Width:	32
AXI Address Width:	32
AXI ID Width: [1 - 8]	2
AXI Stream Data Width:	8
AXI Stream TDEST Width: [1 - 4]	4
AXI Stream TID Width: [1 - 4]	4
Memory Map to Stream FIFO Depth:	2048
Stream to Memory Map FIFO Depth :	2048
AXI Lite Data Width :	32
AXI Lite Address Width :	32
Buffer Descriptor Data Width :	32
Buffer Descriptor Address Width :	32

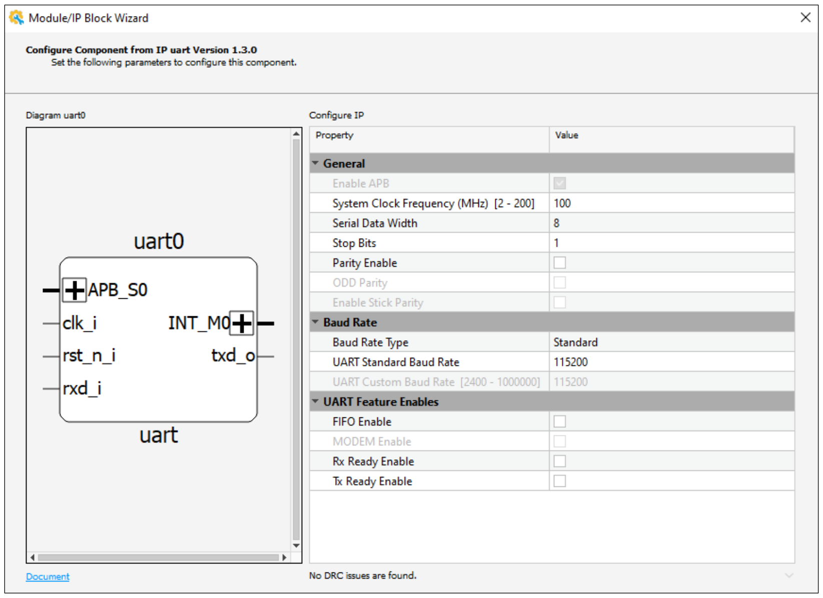
No DRC issues are found.

Figure 2.12. SGDMA Configuration

2.4.6. UART

For more information about the IP core including register map information, refer to UART IP User Guide (FPGA-IPUG-02105).

The Universal Asynchronous Receiver/Transmitted (UART) Transceiver IP core performs serial-to-parallel conversion of data characters received from a peripheral UART device and parallel-to-serial conversion of data characters received from the host locater insider the FPGA through an APB interface.



Configure Component from IP uart Version 1.3.0
Set the following parameters to configure this component.

Configure IP

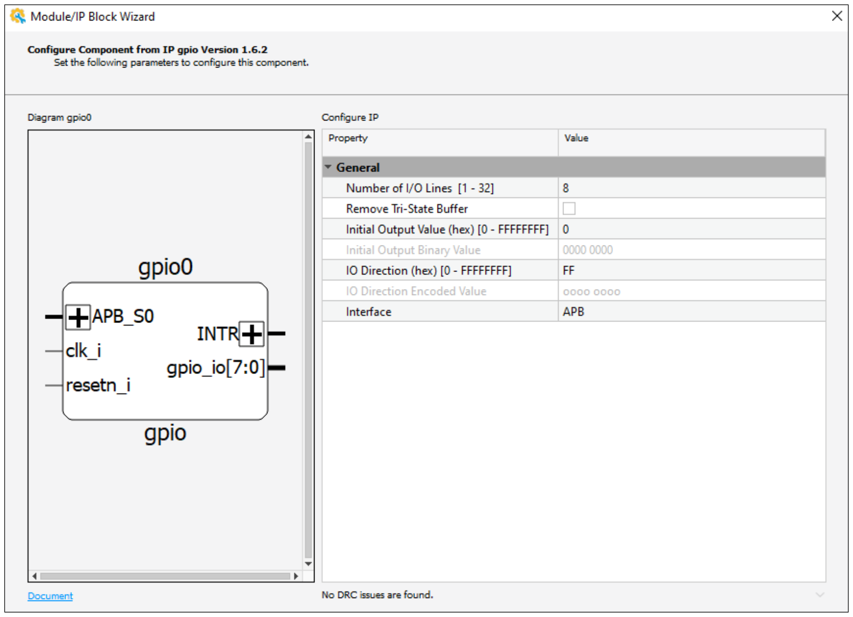
Property	Value
General	
Enable APB	✓
System Clock Frequency (MHz) [2 - 200]	100
Serial Data Width	8
Stop Bits	1
Parity Enable	☐
ODD Parity	☐
Enable Stick Parity	☐
Baud Rate	
Baud Rate Type	Standard
UART Standard Baud Rate	115200
UART Custom Baud Rate [2400 - 1000000]	115200
UART Feature Enables	
FIFO Enable	☐
MODEM Enable	☐
Rx Ready Enable	☐
Tx Ready Enable	☐

No DRC issues are found.

Figure 2.13. UART Configuration

2.4.7. GPIO

For more information about the IP core including register map information, refer to GPIO IP Core (FPGA-IPUG-02076). The General Purpose Input/Output (GPIO) peripheral IP provides dedicated memory-mapped interface to configure the GPIO ports as well as the number of input and output ports.



Configure Component from IP gpio Version 1.6.2
Set the following parameters to configure this component.

Configure IP

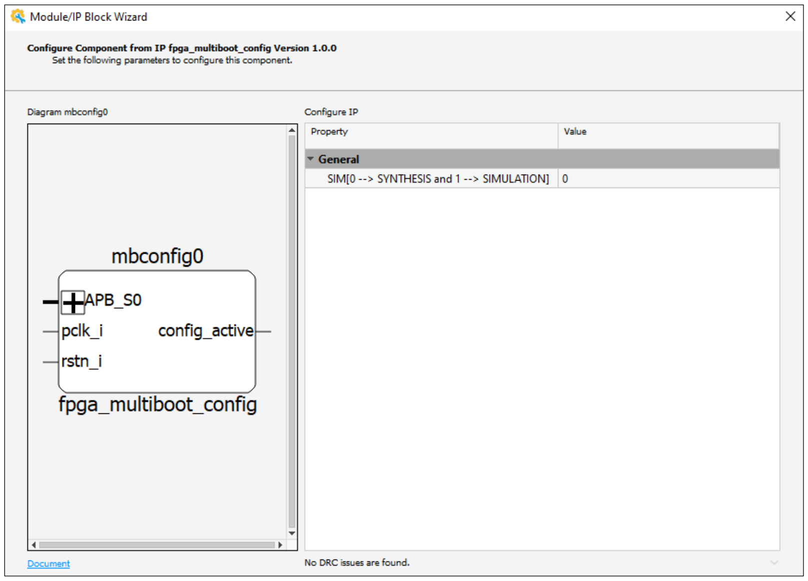
Property	Value
General	
Number of I/O Lines [1 - 32]	8
Remove Tri-State Buffer	☐
Initial Output Value (hex) [0 - FFFFFFFF]	0
Initial Output Binary Value	0000 0000
IO Direction (hex) [0 - FFFFFFFF]	FF
IO Direction Encoded Value	oooo oooo
Interface	APB

No DRC issues are found.

Figure 2.14. GPIO Configuration

2.4.8. Multi-Boot Configuration Module

The Multi-Boot Configuration is used to trigger an internal FPGA REFRESH/PROGRAMN command to LMMI logic. This core IP implements an APB endpoint which decodes the RISC-V CPU command data. The LMMI host FSM inside is used to execute the soft reset to load the next or alternate bitstream and application software data onto the FPGA.



Configure Component from IP fpga_multiboot_config Version 1.0.0
Set the following parameters to configure this component.

Property	Value
General	
SIM[0 --> SYNTHESIS and 1 --> SIMULATION]	0

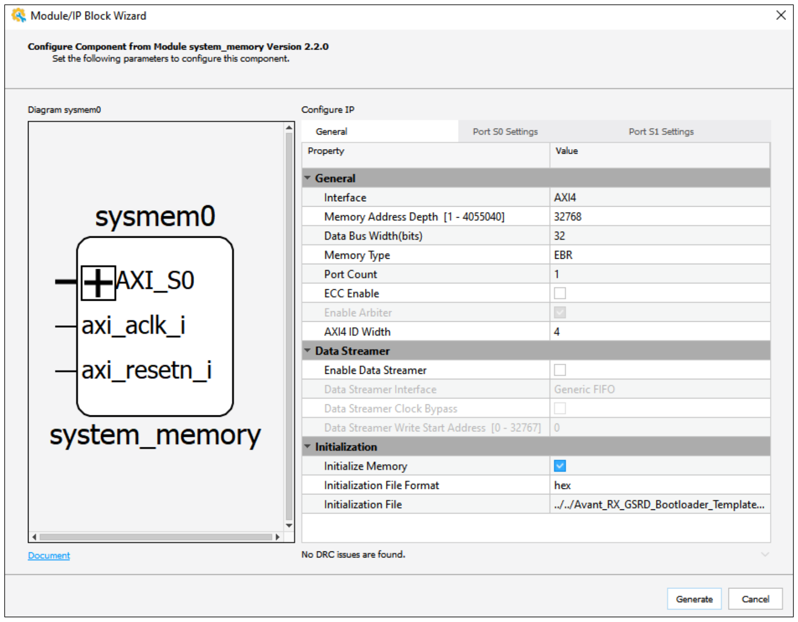
No DRC issues are found.

Figure 2.15. Multi-Boot Configuration

2.4.9. System Memory

For more information about the IP core including register map information, refer to System Memory Module (FPGA-IPUG-02073).

The System Memory implements EBR, LRAM, or Distributed Memory in either single port or dual port AHBL or AXI4 subordinate. In GSRD, an EBR and single AXI4 port is used. The system memory in this design is used to store the bootloader.



Configure Component from Module system_memory Version 2.2.0
Set the following parameters to configure this component.

Property	Value
General	
Interface	AXI4
Memory Address Depth [1 - 4055040]	32768
Data Bus Width(bits)	32
Memory Type	EBR
Port Count	1
ECC Enable	☐
Enable Arbiter	✓
AXI4 ID Width	4
Data Streamer	
Enable Data Streamer	☐
Data Streamer Interface	Generic FIFO
Data Streamer Clock Bypass	☐
Data Streamer Write Start Address [0 - 32767]	0
Initialization	
Initialize Memory	✓
Initialization File Format	hex
Initialization File	../../Avant_RX_GSRD_Bootloader_Template...

No DRC issues are found.

Figure 2.16. System Memory Configuration

2.5. System Level Interfaces

Table 2.3. System Level Interfaces

Top-Level Interface Name	Supported Protocol	Description
Advanced eXtensible Interface 4	AXI4	Used for Data/Control Interfaces on all IPs
Advanced eXtensible Interface 4 - Lite	AXI4-Lite	Used as Control Interface for SGDMA
Advanced Peripheral Interface	APB	Used as Control Interface for low-speed and LPDDR4 MC
Serial Peripheral Interface	SPI/QSPI	Used for communication with external SPI Flash
Dual Data Rate	LPDDR4	Used for communication with external LPDDR4
1G Ethernet	RGMII	Used for communication with external FMC

2.6. SoC Memory/Address Map

Table 2.4. Address Map of GHRD

Base Address	End Address	Size (kB/MB/1 GB)	Block
0x00000000	0x0003FFFF	256 kB	CPU Instruction/Data RAM
0xF2000000	0xF20FFFFF	1 MB	CLINT (CPU)
0xFC000000	0xFC3FFFFF	4 MB	PLIC (CPU)
0xF0000000	0xF00003FF	—	Reserved_Space1 (CPU)
0xF0000400	0xF1FFFFFF	—	Reserved_Space2 (CPU)
0xFC400000	0xFFFFFFFF	—	Reserved_Space3 (CPU)
0x40000000	0x40000FFF	4 kB	GPIO
0x40092000	0x40092FFF	4 kB	LPDDR4 APB
0x80000000	0xBFFFFFFF	1 GB	LPDDR4 AXI
0x40097000	0x40097FFF	4 kB	Multi-Boot Config APB
0x40300000	0x4030FFFF	64 kB	QSPI Flash Controller
0x40098000	0x40099FFF	4 kB	SGDMA
0x00000000	0x0001FFFF	128 kB	System Memory
0x40001000	0x40004FFF	16 kB	TSE MAC
0x40090000	0x40090FFF	4 kB	UART
0x40099000	0x40099FFF	4 kB	Watch Dog Timer

2.7. Functional Operation

The GSRD system comprises of two SoCs (System on Chip) such as Primary and Golden. Each SoC is linked with its respective First-Stage Bootloader (FSBL) and a FreeRTOS Application Software. These SoCs are built using Lattice Propel SDK, Lattice Propel Builder, and Lattice Radiant software tools. The FPGA image comprises of the entire system and is generated by the Radiant tool in the (.bit) bitstream format. The bootloader code is stored inside the system memory and is part of the bitstream. The bitstream and the FreeRTOS application software are stored into the external SPI Flash (Winbond).

During power-on, the Primary FPGA image is loaded into the device CRAM by the Config Engine. Before the device is completely programmed with the bitstream, the config engine checks the CRC (Cyclic Redundancy Check) for the bitstream to be loaded onto the FPGA. Once the FPGA is configured, DONE LED on the board glows green. Immediately, the RISC-V starts executing the bootloader software stored inside system memory and initialize all the soft-IP modules and peripherals to establish a base for communication and data transfers. This process includes configuring the IPs for the desired system operation. After all the IP configuration is complete, RISC-V initiates trigger the QSPI Flash controller to copy the FreeRTOS application software from external SPI Flash into the external LPDDR4 for software execution. Once the application software is copied, the RISC-V checks the CRC on entire copied application software. If the CRC check fails, RISC-V initiates a soft reset/refresh by instructing the multi-boot configuration module to start the PROGRAMN sequence. The PROGRAMN sequence loads the next bitstream into the FPGA which in this case would be the Golden FPGA and FW images, and the same process follows. Once CRC check passes for either Primary or Golden, the RISC-V jumps to the application instruction and starts the FreeRTOS FW execution.

3. Signal Description

Table 3.1 shows the input/output interface signals for the top-level module.

Table 3.1. Top-level I/O

Signal Name	I/O Type	I/O Width	Description
clk_125MHz	Input	1	Reference clock input for internal PLL and TSE MAC/SGMII
pll_refclk_i	Input	1	Reference clock input for LPDDR4 MC internal PLL
rstn_i	Input	1	Active low reset input for reference design. Activate by pressing SW1 push button on the board.
GPIO			
s0_gpio	Input/Output	8	General Purpose I/O signals connect to LED on board
UART			
s1_uart_txd_o	Output	1	UART transmit output. Connects to the board TXD signal.
s1_uart_rxd_i	Input	1	UART receive input. Connects to the board RXD signal.
QSPI Flash			
SPI_CLK	Output	1	Serial clock to SPI Flash
SPI_CSS	Output	1	SPI Flash chip select
SPI_MOSI	Input/Output	1	Serial data between FPGA and external SPI Flash
SPI_MISO	Input/Output	1	Serial data between FPGA and external SPI Flash
SPI_D2	Input/Output	1	Serial data between FPGA and external SPI Flash
SPI_D3	Input/Output	1	Serial data between FPGA and external SPI Flash
LPDDR4 MC			
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	External Memory chip reset signal
init_done_o	Output	1	Connects to LED for status check
irq_o	Output	1	Connects to LED for status check
pll_lock_o	Output	1	Connects to LED for status check
sclk_o	Output	1	Connects to LED for status check
trn_err_o	Output	1	Connects to LED for status check
RGMII Interface			
rgmii_rxc_i	Input	1	RGMII Receive Clock from external PHY
rgmii_rxctl_i	Input	1	RGMII Receive Control signal from external PHY
rgmii_rxd_i	Input	4	4-bit RGMII Receive Data from external PHY
rgmii_txc_o	Output	1	RGMII Transmit Clock to external PHY
rgmii_txctl_o	Output	1	RGMII Transmit Control signal to external PHY
rgmii_txd_o	Output	4	4-bit RGMII Transmit Data to external PHY
rgmii_mdio_o	Output	1	RGMII Transmit Control signal to external PHY
rgmii_mdc_o	Output	1	RGMII Transmit Control signal to external PHY
phy_resetn_o	Output	1	Hard reset generated from FPGA to reset the external PHY
Multi-Boot			
config_active_o	Output	1	Connects to LED to check Multi-Boot Configuration Block

4. Software Components

The GSRD (Golden System Reference Design) is enabled by RISC-V core based FreeRTOS and Lattice FPGA IP modules. Lattice developed BSP (Board Support Package) drivers in C language act as intermediaries, facilitating communication between the hardware elements on the FPGA and FreeRTOS software. During boot up, these drivers initialize and configure FPGA peripherals to establish effective coordination with the RISC-V processor.

4.1. Primary and Golden Bootloader

Bootloader is a bare-metal program that does the following IP configurations:

- Configures the GPIO and UART IP
- For LPDDR4 MC configuration, it first reads if the PLL Lock status is set and then initiates Memory Training and waits until training is complete.
- Configures the QSPI flash controller to read the application software stored into external flash while FIFO is disabled using API and copies it into LPDDR4.
- It then calculates the CRC value on the entire FW copied into DDR using crc16_ccit() API and compares the value with the original CRC value.
- Failure of the CRC check on the Primary application software triggers reconfiguration of the FPGA with the Golden FPGA bitstream image. This is done by triggering Multi-Boot Configuration module. This step is only used in Primary GSRD system.

4.2. Primary and Golden Application

- Executed by RISC-V RX CPU core from LPDDR4 where it initializes the BSP and Operating System
- Configures the TSE MAC handler with 1G Ethernet address, speed mode, MAC upper and lower. Passes TSE Core object to initialize the Ethernet IP and sets the MAC address.
- Creates an Ethernet frame using the API provided in ethernet_frame.h.
- Configure the SGDMA IP for MM2S to transfer the Ethernet frame to the TSE MAC and set up S2MM to receive packets from the TSE MAC at 2-second intervals. Runs FreeRTOS scheduler and Task Handler.

For the PHY initialization, it configures the PHY using mdio_phy_init() API by performing a PHY reset before starting the TSE packet traffic. The PHY configuration checks if the Marvell 88E151X PHY is attached before performing the configuration.

This also captures the packets sent from the Avant board at two second intervals.

5. Design Constraints

The design constraints are divided into two parts, Pre-Synthesis (SDC) and Post-Synthesis Physical (PDC) constraints. They are used to ensure that the design meets the required performance, timing closure, functionality and physical placement requirements as per the FPGA device.

5.1. I/O Constraints

```
ldc_set_sysconfig {MULTI_BOOT_MODE=ENABLE MULTI_BOOT_SEL=STATIC BOOT_SEL=DUAL
MSPI_ADDRESS_32BIT=ENABLE MSPI_COMMAND_32BIT=ENABLE PROGRAMN_RECOVERY=ENABLE}

create_clock -name {pll_refclk_i} -period 10 -waveform {0.000 5.000} [get_ports
pll_refclk_i]
create_clock -name {clk_125_in} -period 8 -waveform {0.000 4.000} [get_ports clk_125_in]
create_clock -name {rgmii_mdc_o} -period 100 [get_nets rgmii_mdc_o_c]
#100MHz LPDDR4 PLL REFERENCE CLOCK
ldc_set_location -site {AD32} [get_ports pll_refclk_i]
ldc_set_location -site {AL10} [get_ports clk_125_in]
#RESET
ldc_set_location -site {AC17} [get_ports rstn_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18 PULLMODE=UP} [get_ports rstn_i]
#UART
ldc_set_location -site {R3} [get_ports s1_uart_txd_o]
ldc_set_location -site {T3} [get_ports s1_uart_rxd_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports s1_uart_rxd_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports s1_uart_txd_o]

#GPIO
ldc_set_location -site {N7} [get_ports {s0_gpio[0]}]
ldc_set_location -site {L7} [get_ports {s0_gpio[1]}]
#ldc_set_location -site {L8} [get_ports {s0_gpio[2]}]
#ldc_set_location -site {P8} [get_ports {s0_gpio[3]}]
#ldc_set_location -site {M8} [get_ports {s0_gpio[4]}]
#ldc_set_location -site {M9} [get_ports {s0_gpio[5]}]
#ldc_set_location -site {P10} [get_ports {s0_gpio[6]}]
#ldc_set_location -site {N10} [get_ports {s0_gpio[7]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports {s0_gpio[*]}]


#SPI FLASH
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 SLEWRATE=FAST} [get_ports SPI_CLK]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 SLEWRATE=FAST} [get_ports SPI_CSS]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 SLEWRATE=FAST} [get_ports SPI_MOSI]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 SLEWRATE=FAST} [get_ports SPI_MISO]
ldc_set_port -iobuf {PULLMODE=UP} [get_ports SPI_D2]
ldc_set_port -iobuf {PULLMODE=UP} [get_ports SPI_D3]
ldc_set_location -site {L12} [get_ports SPI_MOSI]
ldc_set_location -site {L10} [get_ports SPI_D2]
ldc_set_location -site {L11} [get_ports SPI_MISO]
ldc_set_location -site {L9} [get_ports SPI_D3]
ldc_set_location -site {P9} [get_ports SPI_CLK]
ldc_set_location -site {P7} [get_ports SPI_CSS]
```

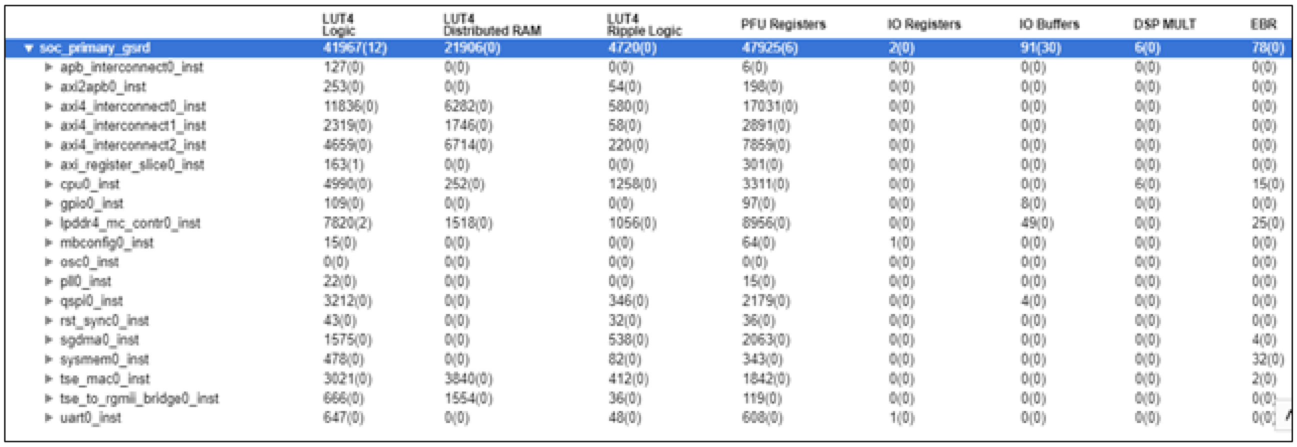
```
#FPGA CONFIG LED
ldc_set_location -site {M8} [get_ports config_active_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports config_active_o]

#LPDDR4 MC LEDs
ldc_set_location -site {N10} [get_ports pll_lock_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports pll_lock_o]
ldc_set_location -site {P10} [get_ports init_done_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports init_done_o]
ldc_set_location -site {M9} [get_ports trn_err_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports trn_err_o]

#LPDDR4 MC
ldc_set_location -site {AC26} [get_ports {ddr_ca_o[0]}]
ldc_set_location -site {AB27} [get_ports {ddr_ca_o[1]}]
ldc_set_location -site {AB28} [get_ports {ddr_ca_o[2]}]
ldc_set_location -site {AC29} [get_ports {ddr_ca_o[3]}]
ldc_set_location -site {AB29} [get_ports {ddr_ca_o[4]}]
ldc_set_location -site {AD27} [get_ports {ddr_ca_o[5]}]
ldc_set_location -site {AB25} [get_ports {ddr_ck_o[0]}]
ldc_set_location -site {AB26} [get_ports {ddr_cke_o[0]}]
ldc_set_location -site {AA31} [get_ports {ddr_cs_o[0]}]
ldc_set_location -site {AF34} [get_ports ddr_reset_n_o]
ldc_set_location -site {AC23} [get_ports {ddr_dqs_io[0]}]
ldc_set_location -site {AC34} [get_ports {ddr_dqs_io[1]}]
ldc_set_location -site {AD22} [get_ports {ddr_dq_io[0]}]
ldc_set_location -site {AD21} [get_ports {ddr_dq_io[1]}]
ldc_set_location -site {AE25} [get_ports {ddr_dq_io[2]}]
ldc_set_location -site {AE24} [get_ports {ddr_dq_io[3]}]
ldc_set_location -site {AA21} [get_ports {ddr_dq_io[4]}]
ldc_set_location -site {Y21} [get_ports {ddr_dq_io[5]}]
ldc_set_location -site {AB24} [get_ports {ddr_dq_io[6]}]
ldc_set_location -site {AC24} [get_ports {ddr_dq_io[7]}]
ldc_set_location -site {AD34} [get_ports {ddr_dq_io[8]}]
ldc_set_location -site {AD33} [get_ports {ddr_dq_io[9]}]
ldc_set_location -site {AE32} [get_ports {ddr_dq_io[10]}]
ldc_set_location -site {AE31} [get_ports {ddr_dq_io[11]}]
ldc_set_location -site {AE33} [get_ports {ddr_dq_io[12]}]
ldc_set_location -site {AE34} [get_ports {ddr_dq_io[13]}]
ldc_set_location -site {AB34} [get_ports {ddr_dq_io[14]}]
ldc_set_location -site {AB33} [get_ports {ddr_dq_io[15]}]
ldc_set_location -site {AC21} [get_ports {ddr_dmi_io[0]}]
ldc_set_location -site {AF33} [get_ports {ddr_dmi_io[1]}]
ldc_set_location -site {AB30} [get_ports {ddr_ck_o[1]}]
ldc_set_location -site {Y29} [get_ports {ddr_dqs_io[2]}]
ldc_set_location -site {W23} [get_ports {ddr_dqs_io[3]}]
ldc_set_location -site {U29} [get_ports {ddr_dq_io[16]}]
ldc_set_location -site {V29} [get_ports {ddr_dq_io[17]}]
ldc_set_location -site {U30} [get_ports {ddr_dq_io[18]}]
ldc_set_location -site {U31} [get_ports {ddr_dq_io[19]}]
ldc_set_location -site {Y28} [get_ports {ddr_dq_io[20]}]
ldc_set_location -site {AA28} [get_ports {ddr_dq_io[21]}]
ldc_set_location -site {V30} [get_ports {ddr_dq_io[22]}]
ldc_set_location -site {W30} [get_ports {ddr_dq_io[23]}]
ldc_set_location -site {AA24} [get_ports {ddr_dq_io[24]}]
ldc_set_location -site {Y24} [get_ports {ddr_dq_io[25]}]
ldc_set_location -site {U23} [get_ports {ddr_dq_io[26]}]
ldc_set_location -site {U22} [get_ports {ddr_dq_io[27]}]
ldc_set_location -site {U25} [get_ports {ddr_dq_io[28]}]
ldc_set_location -site {T25} [get_ports {ddr_dq_io[29]}]
ldc_set_location -site {U24} [get_ports {ddr_dq_io[30]}]
ldc_set_location -site {T24} [get_ports {ddr_dq_io[31]}]
ldc_set_location -site {V32} [get_ports {ddr_dmi_io[2]}]
ldc_set_location -site {V22} [get_ports {ddr_dmi_io[3]}]

# The ECLKDIV primitive already has constraint but the inferred name is not good.
# The constraint below only aims to assign a name to the generated clock.
create_generated_clock -name {sclk_o} -source [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/pll_inst/gen_ext_outclkdiv*.u_pll.PLLC_MODE_inst/CLKOPHY}]
-divide_by 8 [get_pins -hierarchical {lscc_mc_avant_inst/u_ddrphy/u_eclkdiv/ECLKDIVOUT
}]
set_clock_uncertainty -setup 0.075 [get_clocks sclk_o]

# The PLL lock is false as it toggles only once for a long time
# It is also sampled by double FF
set_false_path -to [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/i_csr/p_pll_lock_r1*/D}]

# This signal is asserted and then de-asserted while the clocks are stopped, thus, they
are false path.
set_false_path -to [get_pins -hierarchical {lscc_mc_avant_inst/u_ddrphy/u_eclkdiv/RST}]

# Constraints for the CDC logic on the DDRPHY's CSR
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/i_csr/wr_data_buff_r*/Q}] -to [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/i_csr/s_apb_data_r*/D}] -datapath_only 3
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/i_csr/addr_buff_r*/Q}] -to [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/i_csr/s_apb_addr_r*/D}] -datapath_only 3

#Constraints for MC's FIFO DC instances
# The distributed RAM data output port to FIFO_DC data output register is already
guaranteed by the CDC logic and above constraints.
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/AXI_BI.u_axi_if/u_rd/ASYNC.u_ctrl_fifo/u_fifo_dc/*.dpram_inst/DO*}]
-datapath_only 3
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/AXI_BI.u_axi_if/u_rd/u_rd_rsp/u_data_fifo/u_fifo_dc/*.dpram_inst/DO*
}] -datapath_only 3
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/AXI_BI.u_axi_if/u_wr/ASYNC.u_ctrl_fifo/u_fifo_dc/*.dpram_inst/DO*}]
-datapath_only 3
```

```
set_max_delay -from [get_pins -hierarchical
{lscc_mc_avant_inst/AXI_BI.u_axi_if/u_wr/ASYNC.u_data_fifo/u_fifo_dc/*.dpram_inst/DO*}]
-datapath_only 3


# The reset pin of double FF that synchronizes the reset de-assertion is false path
set_false_path -to [get_pins -hierarchical {lscc_mc_avant_inst/rst_n*/CD}]
set_false_path -to [get_pins -hierarchical {lscc_mc_avant_inst/reset_n*/CD}]
set_false_path -to [get_pins -hierarchical {lscc_mc_avant_inst/phy_srst_n*/CD}]
set_false_path -to [get_pins -hierarchical
{lscc_mc_avant_inst/u_ddrphy/start_rst_r*/PD}]
set_false_path -to [get_pins -hierarchical {lscc_mc_avant_inst/u_ddrphy/s_rst_n_r*/CD}]

# TSE MAC RGMII Constraints
ldc_set_location -site {Y10} [get_ports rgmii_txc_o]
ldc_set_location -site {AB12} [get_ports rgmii_txctl_o]
ldc_set_location -site {Y11} [get_ports {rgmii_txd_o[0]}]
ldc_set_location -site {AA2} [get_ports {rgmii_txd_o[1]}]
ldc_set_location -site {Y2} [get_ports {rgmii_txd_o[2]}]
ldc_set_location -site {AB13} [get_ports {rgmii_txd_o[3]}]
ldc_set_location -site {Y7} [get_ports rgmii_rxc_i]
ldc_set_location -site {Y6} [get_ports rgmii_rxctl_i]
ldc_set_location -site {AC7} [get_ports {rgmii_rxd_i[0]}]
ldc_set_location -site {AC6} [get_ports {rgmii_rxd_i[1]}]
ldc_set_location -site {AD6} [get_ports {rgmii_rxd_i[2]}]
ldc_set_location -site {AD5} [get_ports {rgmii_rxd_i[3]}]
ldc_set_location -site {AA8} [get_ports rgmii_mdc_o]
ldc_set_location -site {Y9}  [get_ports rgmii_mdio_o]

ldc_set_port -iobuf {IO_TYPE=LVCMOS18} [get_ports rgmii_rxc_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18} [get_ports rgmii_rxctl_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18} [get_ports {rgmii_rxd_i[*]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18 SLEWRATE=FAST} [get_ports rgmii_txc_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18 SLEWRATE=FAST} [get_ports rgmii_txctl_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18 SLEWRATE=FAST} [get_ports {rgmii_txd_o[*]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18} [get_ports rgmii_mdc_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18 PULLMODE=UP} [get_ports rgmii_mdio_o]
ldc_set_location -site {AA9} [get_ports phy_resetn_o]
ldc_set_port -iobuf {IO_TYPE=LVCMOS18} [get_ports phy_resetn_o]

#create_clock -name {rgmii_rxc_i} -period 8 -waveform {2 6} [get_ports rgmii_rxc_i]
create_clock -name {rgmii_rxc_i} -period 8 [get_ports rgmii_rxc_i]
create_clock -name {rgmii_rxc_i_pad_0} -period 8
#create_clock -name {rgmii_txc_o} -period 8 [get_nets rgmii_txc_o_c]
# - phase-shift 90 Degree
set_input_delay -clock [get_clocks rgmii_rxc_i] -max 0.5 [get_ports {rgmii_rxctl_i
rgmii_rxd_i[*]}]
set_input_delay -clock [get_clocks rgmii_rxc_i] -clock_fall -min -add_delay -0.5
[get_ports {rgmii_rxctl_i rgmii_rxd_i[*]}]
set_output_delay -clock [get_clocks rgmii_txc_o_c] -max 1 [get_ports {rgmii_txctl_o
rgmii_txd_o[*]}]
set_output_delay -clock [get_clocks rgmii_txc_o_c] -clock_fall -min -add_delay -0.8
[get_ports {rgmii_txctl_o rgmii_txd_o[*]}]
```

```
set_clock_groups -group [get_clocks clk_125_in] -group [get_clocks rgmii_txc_o_c] -group
[get_clocks rgmii_rxc_i] -group [get_clocks rgmii_mdc_o] -asynchronous

set_clock_groups -group [get_clocks clk_125_in] -group [get_clocks rgmii_txc_o_c] -group
[get_clocks pll0_inst_clkos2_o_net] -group [get_clocks
{pll0_inst/lscc_pll_inst/clkout_testclk_o}] -group [get_clocks pll0_inst_clkos_o_net] -
group [get_clocks sclk_o] -group [get_clocks {pll0_inst_clkos3_o_net[0]}] -asynchronous

ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports TCK]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports TDI]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports TMS]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports TDO]

set_false_path -from [get_ports rstn_i]
set_false_path -from [get_pins {cpu0_inst/system_resetn_o}]
```

6. Resource Utilization

Figure 6.1 shows the GSRD resource utilization and Table 6.1 shows the total LUT4, PFU register, I/O buffer, and EBR resource utilization for Avant-AT-E70.



	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	IO Registers	IO Buffers	DSP MULT	EBR
▼ soc_primary_gsrd	41967(12)	21906(0)	4720(0)	47925(6)	2(0)	91(30)	6(0)	78(0)
▶ apb_interconnect0_inst	127(0)	0(0)	0(0)	6(0)	0(0)	0(0)	0(0)	0(0)
▶ axi2apb0_inst	253(0)	0(0)	54(0)	198(0)	0(0)	0(0)	0(0)	0(0)
▶ axi4_interconnect0_inst	11836(0)	6282(0)	580(0)	17031(0)	0(0)	0(0)	0(0)	0(0)
▶ axi4_interconnect1_inst	2319(0)	1746(0)	58(0)	2891(0)	0(0)	0(0)	0(0)	0(0)
▶ axi4_interconnect2_inst	4659(0)	6714(0)	220(0)	7859(0)	0(0)	0(0)	0(0)	0(0)
▶ axi_register_slice0_inst	163(1)	0(0)	0(0)	301(0)	0(0)	0(0)	0(0)	0(0)
▶ cpu0_inst	4990(0)	252(0)	1258(0)	3311(0)	0(0)	0(0)	6(0)	15(0)
▶ gpio0_inst	109(0)	0(0)	0(0)	97(0)	0(0)	8(0)	0(0)	0(0)
▶ lpddr4_mc_contr0_inst	7820(2)	1518(0)	1056(0)	8956(0)	0(0)	49(0)	0(0)	25(0)
▶ mbconfig0_inst	15(0)	0(0)	0(0)	64(0)	1(0)	0(0)	0(0)	0(0)
▶ osc0_inst	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
▶ pll0_inst	22(0)	0(0)	0(0)	15(0)	0(0)	0(0)	0(0)	0(0)
▶ qspi0_inst	3212(0)	0(0)	346(0)	2179(0)	0(0)	4(0)	0(0)	0(0)
▶ rst_sync0_inst	43(0)	0(0)	32(0)	36(0)	0(0)	0(0)	0(0)	0(0)
▶ sgdma0_inst	1575(0)	0(0)	538(0)	2063(0)	0(0)	0(0)	0(0)	4(0)
▶ sysmem0_inst	478(0)	0(0)	82(0)	343(0)	0(0)	0(0)	0(0)	32(0)
▶ tse_mac0_inst	3021(0)	3840(0)	412(0)	1842(0)	0(0)	0(0)	0(0)	2(0)
▶ tse_to_rgmii_bridge0_inst	666(0)	1554(0)	36(0)	119(0)	0(0)	0(0)	0(0)	0(0)
▶ uart0_inst	647(0)	0(0)	48(0)	608(0)	1(0)	0(0)	0(0)	0(0)

Figure 6.1. GSRD Resource Utilization on Avant-AT-E70

Table 6.1. GSRD Total Resource Utilization

Resource	Usage	Percentage Utilization
LUT4 (Logic + Distributed RAM + Ripple Logic)	68593	17.25%
PFU Register	47925	12.05%
I/O Buffers	91	—
EBR	78	7.87%

7. Demo User Guide

7.1. Boot-Up Sequence

This section describes the RISC-V RX CPU boot up sequence that configures IP drivers, operating modes, bootloader and FreeRTOS application software execution.

Here is the description of the terms used throughout the document:

- Boot Up – Process of starting the RISC-V RX CPU, loading the FreeRTOS application software from external SPI Flash into the LPDDR4 memory and executing the application.
- Bootloader – Code that initializes and configures various peripherals and loads the FreeRTOS application software into LPDDR4 memory. It also checks for the CRC of the copied application and decides whether to execute the application software or load the next best bitstream hardware and corresponding software.
- FreeRTOS – Application software that is loaded into LPDDR4 Memory and executed by RISC-V CPU at the end of boot up process.
- SPI Flash – Non-volatile external memory that stores the FreeRTOS application software and multi-boot MCS bitstream.

The following is the boot up sequence shown in Figure 7.1:

- The system provides similar functionality both in Bare Metal and FreeRTOS mode. For the demo, the Golden and Primary application software binaries and their FPGA bitstream images are stored in external SPI Flash before the boot up sequence is initiated.
- The initial bootloader is a part of the internal system memory ROM embedded into the FPGA bitstream stored inside the external SPI Flash. Upon power-on boot up, the bootloader configures the peripherals and GSRD building blocks such as UART, GPIO, I2C, SGDMA, 1G TSE MAC, SGMII PCS, LPDDR4 and QSPI Controller.
- The bootloader loads the bitstream from the SPI Flash to program the SRAM of the FPGA and fetches the respective application software (Primary) via the QSPI flash controller.
- As the application software needs to be executed from the external memory, the RISC-V module loads it into LPDDR4 Memory controller.
- This application software is stored at the beginning of the LPDDR4 Memory. After the application software is loaded, the RISC-V CPU calculates the CRC of the application code in LPDDR4.
- The calculated LPDDR4 CRC is compared with the original CRC, that is a part of SPI Flash.
  - If the condition matches, the RISC-V CPU jumps to FreeRTOS execution from LPDDR4 memory
  - If the condition mismatches, the RISC-V CPU issues a FPGA REFRESH command to load the Golden bitstream and application software from SPI Flash.
- Upon the execution of the correct Primary or Golden application software from LPDDR4, the building blocks mentioned earlier are up and running with their associated drivers. For example, if any Ethernet data is expected to arrive, the RISC-V CPU sets up the SGDMA IP accordingly with address, data length and other configuration modes to successfully route the incoming Ethernet packets.
- When the Ethernet frame is received by the TSE MAC IP, it forwards it to SGDMA to transfer the data to its destination based on parameters set. The endpoint in this case is the main memory LPDDR4.
- You can also choose to store the data into another system memory based on SGDMA configuration.
- For outgoing data, the data is fetched from a location inside LPDDR4, and SGDMA transfers the data to TSE MAC for transmission outside the FPGA.
- When no data activity occurs over Ethernet, the RISC-V CPU continues running its tasks in the usual manner based on the loaded software execution.

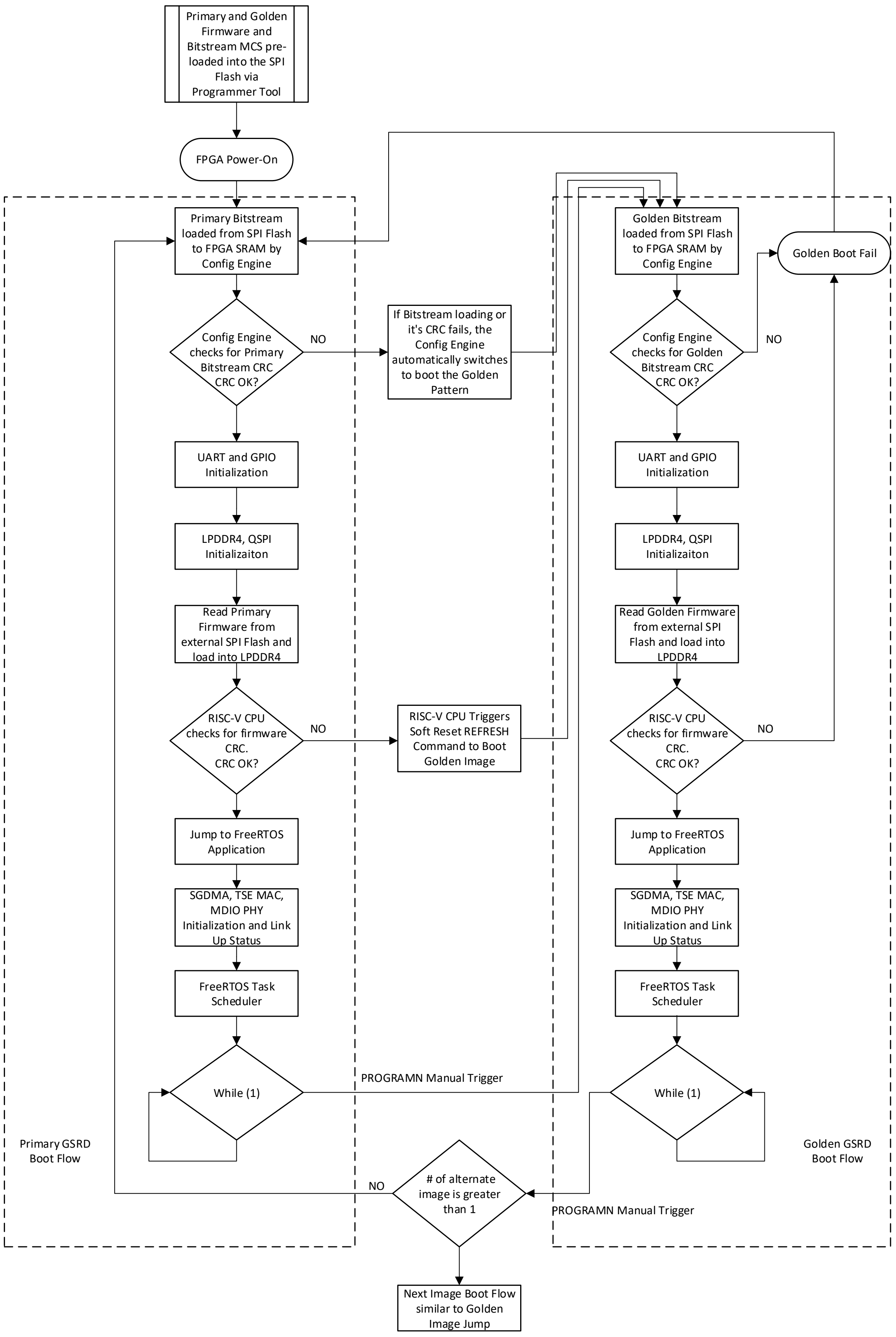

Primary and Golden Firmware and Bitstream MCS pre-loaded into the SPI Flash via Programmer Tool
FPGA Power-On
Primary Bitstream loaded from SPI Flash to FPGA SRAM by Config Engine
Config Engine checks for Primary Bitstream CRC CRC OK?
NO
If Bitstream loading or it's CRC fails, the Config Engine automatically switches to boot the Golden Pattern
UART and GPIO Initialization
LPDDR4, QSPI Initializaiton
Read Primary Firmware from external SPI Flash and load into LPDDR4
RISC-V CPU checks for firmware CRC. CRC OK?
NO
RISC-V CPU Triggers Soft Reset REFRESH Command to Boot Golden Image
Jump to FreeRTOS Application
SGDMA, TSE MAC, MDIO PHY Initialization and Link Up Status
FreeRTOS Task Scheduler
While (1)
PROGRAMN Manual Trigger
Primary GSRD Boot Flow
NO
# of alternate image is greater than 1
Next Image Boot Flow similar to Golden Image Jump
Golden Bitstream loaded from SPI Flash to FPGA SRAM by Config Engine
Golden Boot Fail
Config Engine checks for Golden Bitstream CRC CRC OK?
NO
UART and GPIO Initialization
LPDDR4, QSPI Initializaiton
Read Golden Firmware from external SPI Flash and load into LPDDR4
RISC-V CPU checks for firmware CRC. CRC OK?
NO
Jump to FreeRTOS Application
SGDMA, TSE MAC, MDIO PHY Initialization and Link Up Status
FreeRTOS Task Scheduler
While (1)
Golden GSRD Boot Flow
PROGRAMN Manual Trigger

Figure 7.1. GSRD Boot-Up Sequence

7.2. Prerequisites

The following sections show the hardware and software requirements to execute the GSRD demonstration.

7.2.1. Software Requirements

- Lattice Propel 2024.1 Package – contains both Lattice Propel SDK and Lattice Propel Builder
    - Download here: Lattice Propel 2024.1
- Lattice Radiant 2024.1.1 Package – contains IP Packager, Radiant Software, QuestaSim, and Programmer
    - Download here: Lattice Radiant 2024.1.1
- Lattice Propel 2024.1 Patch for Avant-AT-E Golden System Reference Design
    - Download here: Downloadable Software tab in the GHRD/GSRD Reference Design

7.2.2. Avant-AT-E Requirements

7.2.2.1. Hardware Needed

The section describes the hardware needed to run the GSRD demonstration.

- Lattice Avant-E70 ES1 Evaluation Board
- USB Type-A UART cable for programming the bitstream, application software and proper terminal prints
- Electrical Ethernet FMC daughter card for Ethernet connection over RGMII
    - To purchase the daughter card, go to https://ethernetfmc.com/ and select the part number **OP031-1V8**.
- Ethernet cable to connect one of the Ethernet ports on FMC board to the Host PC
- 12 V power adapter for board power

7.2.2.2. Hardware Setup

This section provides the procedure for setting up the Avant-E70 ES1 board, shown in Figure 7.2.

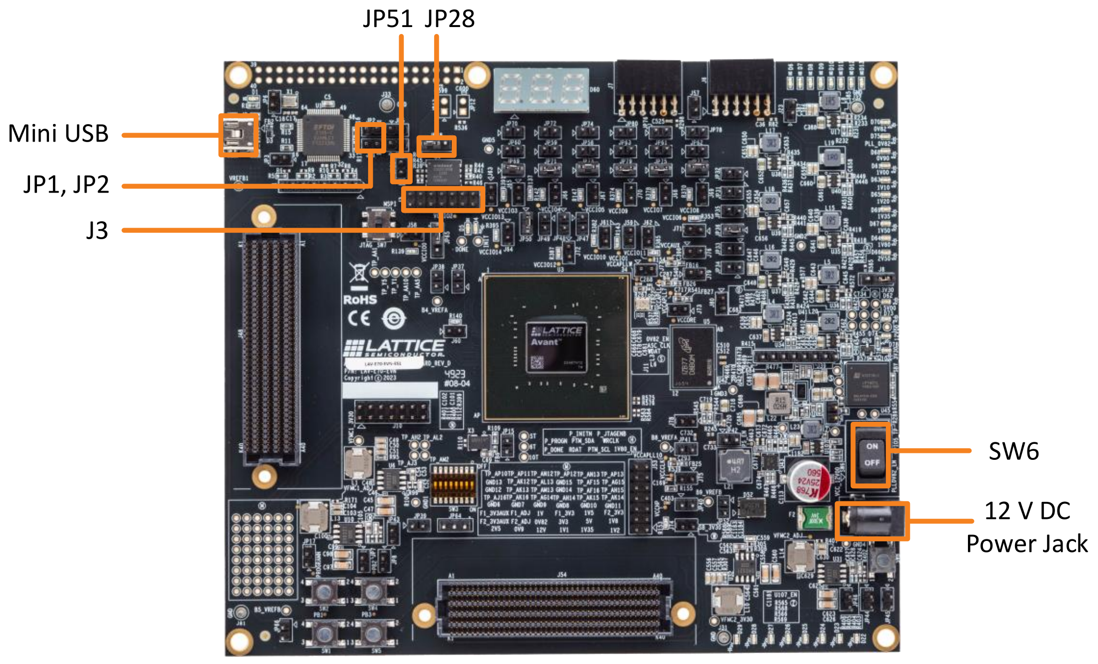


Figure 7.2 .Avant-AT-E70 Evaluation Board

Figure 7.3.Ethernet PHY FMC Card

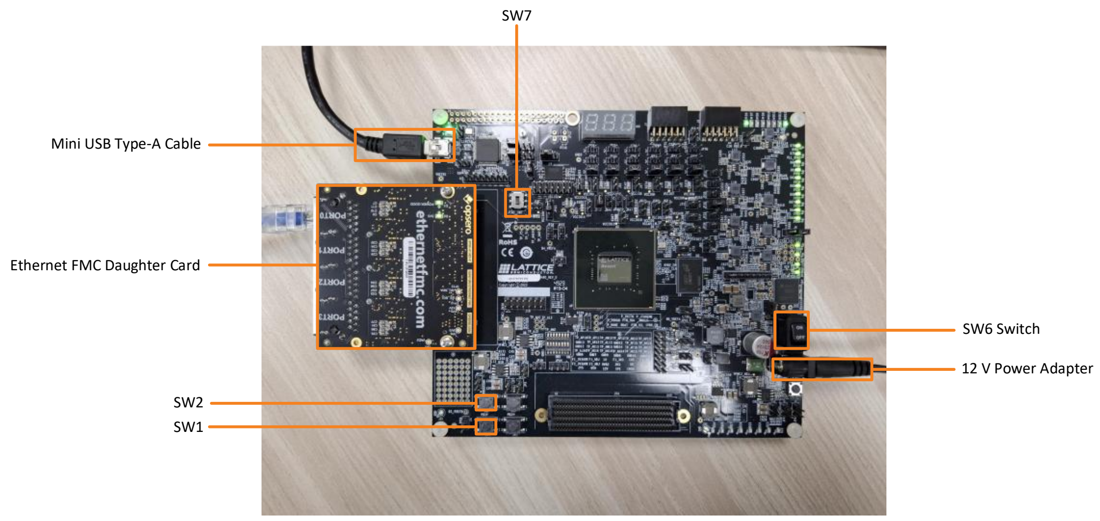


Figure 7.4. Connections and Buttons needed for Demonstration

To setup the Lattice Avant-AT-E board for GSRD demonstration:

1. Connect the 12 V power adapter to J50.
2. Connect the Mini-USB Type-A cable from PC to J2.
3. Connect jumpers on Pin 1 and Pin 2 on both JP1 and JP2 switches to enable UART.
4. Align and carefully connect the Ethernet FMC daughter card onto J48 connector.
5. Connect the Ethernet RJ45 cable from the host PC cable to the Port0 of FMC card.
6. For FPGA executables programming, use SW7 in JTAG mode. Once programmed, switch to MSPI.
7. Turn on the SW6 switch.
8. SW1 is the FPGA Reset pushbutton to reset the GSRD design.
9. SW2 is the PROGRAMN pushbutton to switch between Primary and Golden GSRD manually.

7.2.2.3. Executables

This section provides the directory structure, file names and locations of the executables (SPI Flash) required for running the GSRD demonstration.

- Download the design package from the Lattice Semiconductor website.
    - Go to *Design File* in the GHRD/GSRD Demonstration, download the Avant-AT-E Golden System Reference Design and Demo V1.0 – Bitstream file.

- Unzip the .zip file to your local directory, for example to *<C:\user_workspace>.*
- The extracted directory has the following executables listed in Table 7.1.

Table 7.1. Executable Files for Winbond Flash

File Description	File Name	Starting Address in SPI Flash
Primary Software with CRC	c_primary_appcrc.bin	0x028A 0000
Primary Software without CRC	c_primary_app.bin	0x028A 0000
Primary FPGA Bitstream	soc_primary_system.bit	0x0000 0000
Golden Software with CRC	c_golden_appcrc.bin	0x0280 0000
Golden Software without CRC	c_golden_app.bin	0x0280 0000
Golden FPGA Bitstream	soc_golden_system.bit	0x0000 0000
Multi-Boot MCS File (Golden + Primary Bitstream)	multiboot_system.mcs	0x0000 0000

7.3. Implementing the GHRD/GSRD Demo

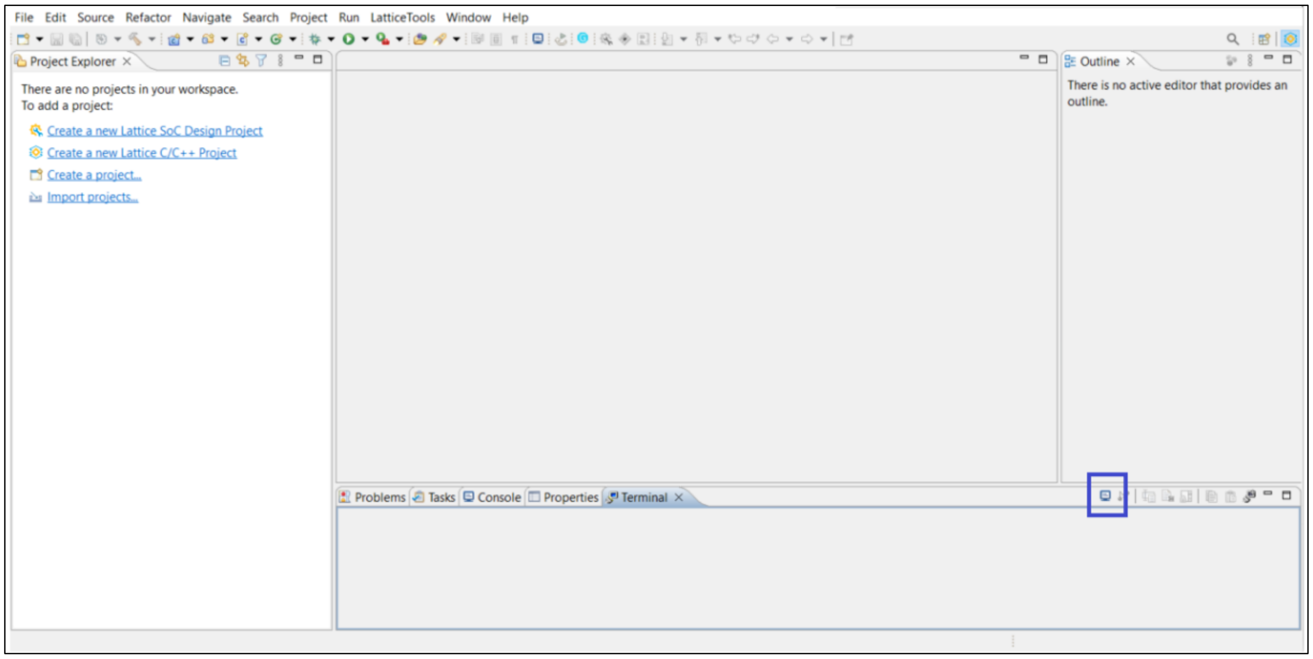
This section describes the procedure for running the GSRD/GHRD demo using the pre-built executables and binary files in the design package.

7.3.1. Setting up the UART Terminal

The software code during the GSRD demonstration displays messages on the terminal through the UART interface.

To setup the UART terminal:

1. Connect the Lattice Avant-AT-E Evaluation board to the PC/Laptop using USB Type-A UART cable.
2. Open Propel SDK 2024.1 tool.
3. Double-click on the terminal button shown in Figure 7.5.

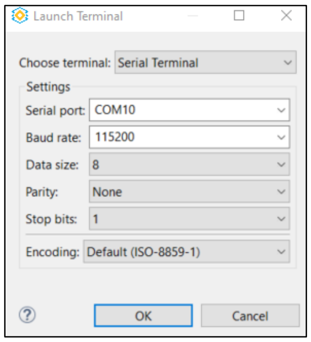

File Edit Source Refactor Navigate Search Project Run LatticeTools Window Help
Project Explorer
Create a new Lattice SoC Design Project
Create a new Lattice C/C++ Project
Create a project...
Import projects...
Outline
Problems
Tasks
Console
Properties
Terminal

There are no projects in your workspace.
To add a project:

There is no active editor that provides an outline.

Figure 7.5. UART Terminal Icon on Propel SDK Window

4. Choose **Serial Terminal** as shown in Figure 7.6. In Serial port dropdown list, select the last COM in the list as shown in Figure 7.7.
   Note: This detail can also be found under the Ports (COM and LPT) section in your local PC, under Device Manager. Your COM port number can be different. If a USB port does not work, try a different USB port.



Choose terminal: Serial Terminal
Settings
Serial port: COM10
Baud rate: 115200
Data size: 8
Parity: None
Stop bits: 1
Encoding: Default (ISO-8859-1)

Figure 7.6. UART Launch Terminal Window

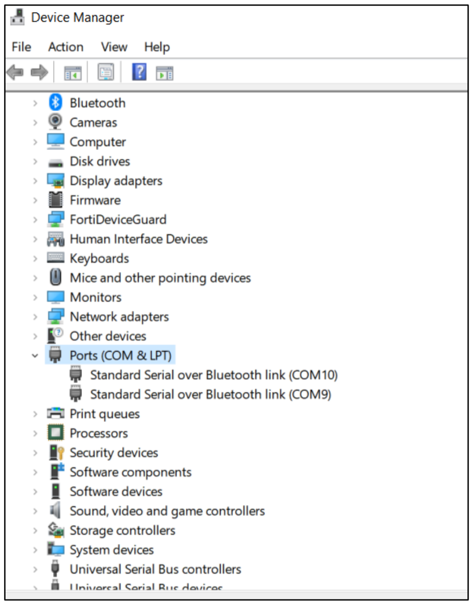


Figure 7.7. Device Manager Window on PC

5. Set Baud rate: **115200**.
6. Click **OK**.

7.3.2. Setting up the Non-Volatile Memory Register

For the GSRD design on Lattice Avant-AT-E design, you need to ensure the settings of the One-Time-Programmable Non-Volatile Configuration Memory. You may skip this section if you have already done this step before. Otherwise, perform a one-time step by JTAG to modify the default MSPI addressing mode from 24-bit to 32-bit. This is necessary for the multi-boot feature to function properly.

7.3.3. Programming the Standalone Golden or Primary GSRD Bitstream and Application Software

1. Connect the Avant-AT-E70 Evaluation Board to a PC/laptop using USB cable as per the hardware setup mentioned in Hardware Setup section. Make sure the JP1 and JP2 are connected properly.
2. Keep SW7 in JTAG mode.
3. Power-on the board.

4. Launch Lattice Programmer tool. In the Getting Started dialog box, select **Create a new blank project**. Browse to the Project Location on your local machine. In this case, it can be the same folder as downloaded executables folder.

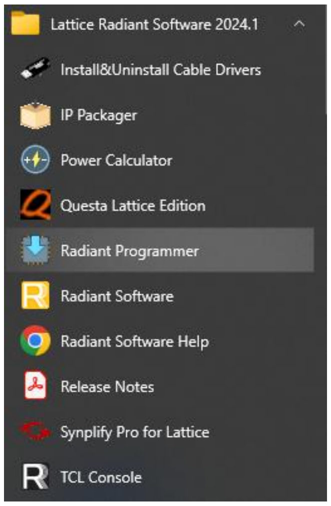
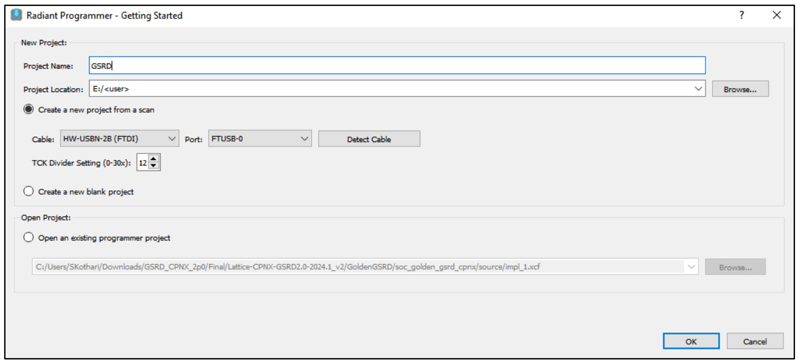


Figure 7.8. Launch Radiant Programmer from Windows Start


Radiant Programmer - Getting Started

New Project:

Project Name: GSRD

Project Location: E:/<user>

[✓ Create a new project from a scan]

Cable: HW-USBN-2B (FTDI) Port: FTUSB-0

TCK Divider Setting (0-30x): 12

Create a new blank project

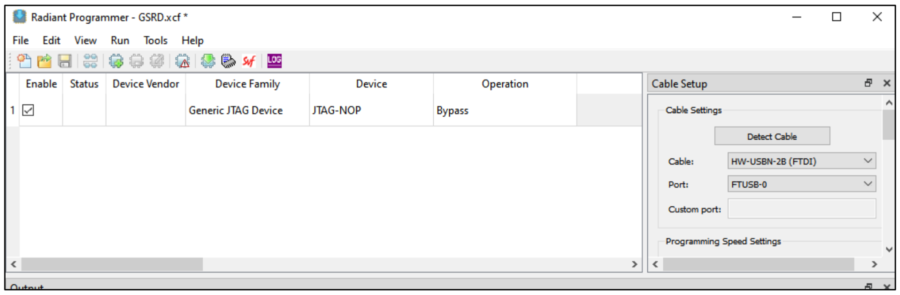
Open Project:

Open an existing programmer project

C:/Users/SKothari/Downloads/GSRD_CPNX_2p0/Final/Lattice-CPNX-GSRD2.0-2024.1_v2/GoldenGSRD/soc_golden_gsrd_cpnx/source/impl_1.xcf

Figure 7.9. Radiant Programmer Start Window

5. Click **OK**.


Radiant Programmer - GSRD.xcf *

	Enable	Status	Device Vendor	Device Family	Device	Operation
1	✓			Generic JTAG Device	JTAG-NOP	Bypass

Cable Setup

Cable Settings

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

Figure 7.10. Radiant Programmer .xcf Window

6. If the Device Family shows as Generic JTAG Device, click **Scan** Device as shown in Figure 7.11 to update the Device Family information automatically.

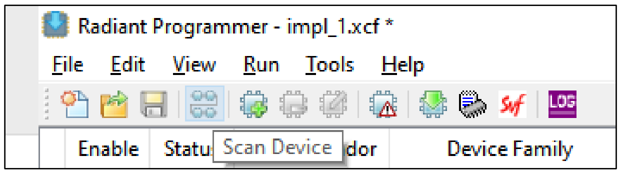
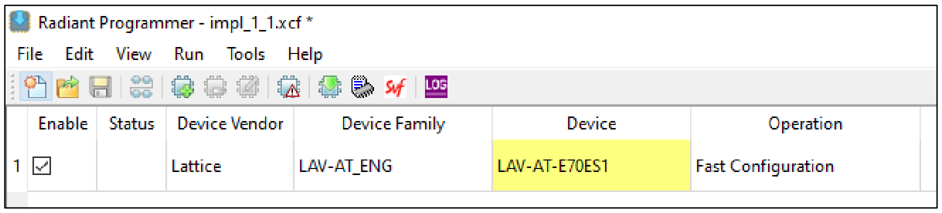

Radiant Programmer - impl_1.xcf *
File Edit View Run Tools Help
Enable Statu Scan Device dor Device Family

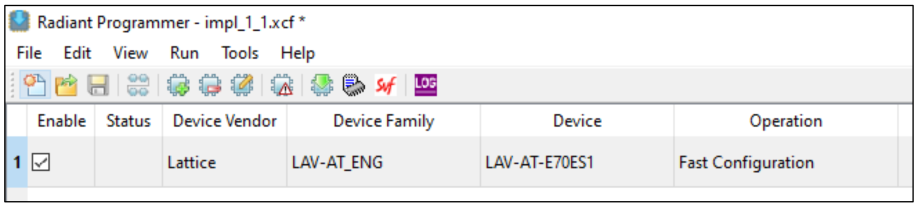
Figure 7.11. Scan Device Icon on Radiant Programmer


Radiant Programmer - impl_1_1.xcf *
File Edit View Run Tools Help

	Enable	Status	Device Vendor	Device Family	Device	Operation
1	☑		Lattice	LAV-AT_ENG	LAV-AT-E70ES1	Fast Configuration

Figure 7.12. Select Device for Programming

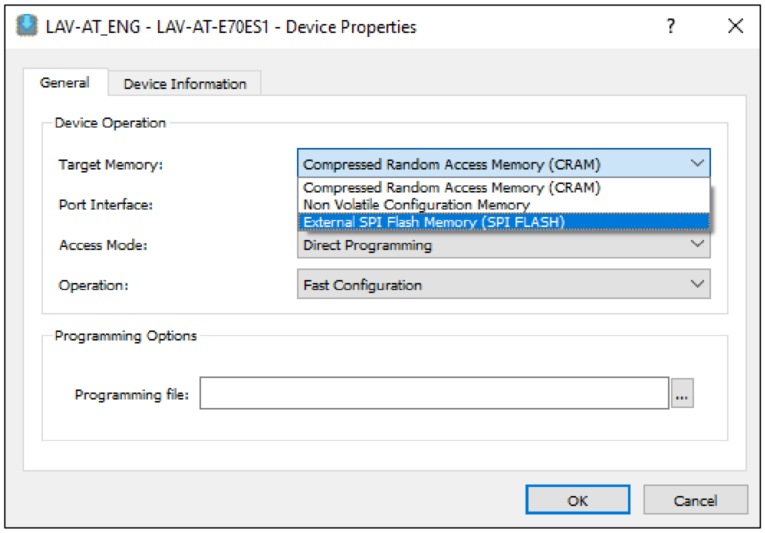
7. Click on the highlighted item under the **Device** field to un-highlight it as shown in Figure 7.13.


Radiant Programmer - impl_1_1.xcf *
File Edit View Run Tools Help

	Enable	Status	Device Vendor	Device Family	Device	Operation
1	☑		Lattice	LAV-AT_ENG	LAV-AT-E70ES1	Fast Configuration

Figure 7.13. Device Selected for Programmer

8. Double-click on the **Operation** tab or right-click and select **Device Properties**.



Device Operation

Target Memory: Compressed Random Access Memory (CRAM)
- Compressed Random Access Memory (CRAM)
- Non Volatile Configuration Memory
- External SPI Flash Memory (SPI FLASH)

Port Interface:

Access Mode: Direct Programming

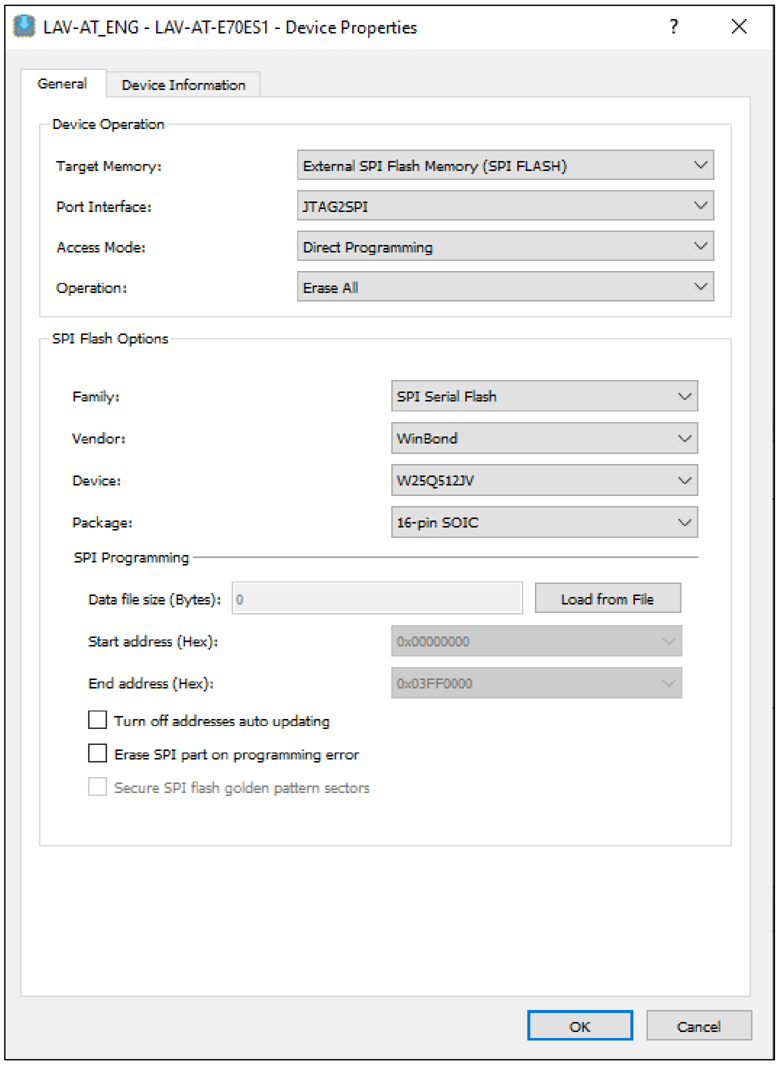
Operation: Fast Configuration

Programming Options

Programming file:

Figure 7.14. Select the Target Memory for Programming

9. Before programming, you need to erase the entire SPI Flash Memory by applying the settings shown in Figure 7.15.


LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

General | Device Information

Device Operation

Target Memory:	External SPI Flash Memory (SPI FLASH)
Port Interface:	JTAG2SPI
Access Mode:	Direct Programming
Operation:	Erase All

SPI Flash Options

Family:	SPI Serial Flash
Vendor:	WinBond
Device:	W25Q512JV
Package:	16-pin SOIC

SPI Programming

Data file size (Bytes):	0
Start address (Hex):	0x00000000
End address (Hex):	0x03FF0000

- [ ] Turn off addresses auto updating
- [ ] Erase SPI part on programming error
- [ ] Secure SPI flash golden pattern sectors

Figure 7.15. Device Properties to Erase the SPI Flash

10. Click **OK** and click on the Program Device Icon or the menu item, **Run > Program Device**. This erases the entire Flash Memory. Wait for the process to complete.

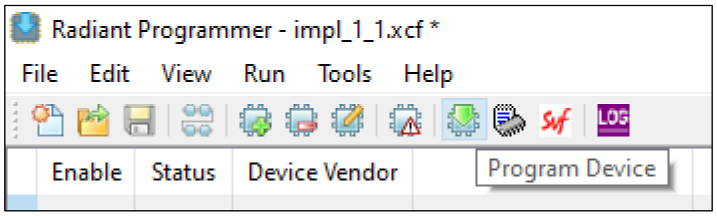
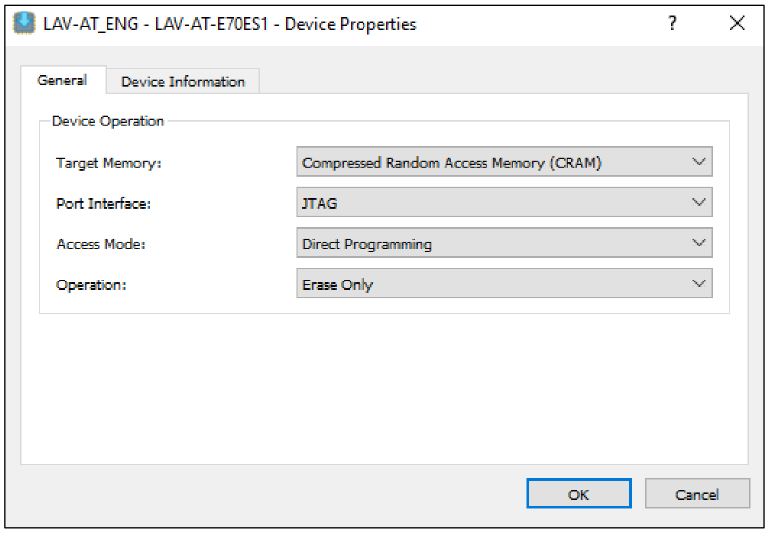


Figure 7.16. Program Button to Program the SPI Flash

11. Power cycle the Lattice Avant board.
12. Erase the FPGA CRAM. Click **OK**.


LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

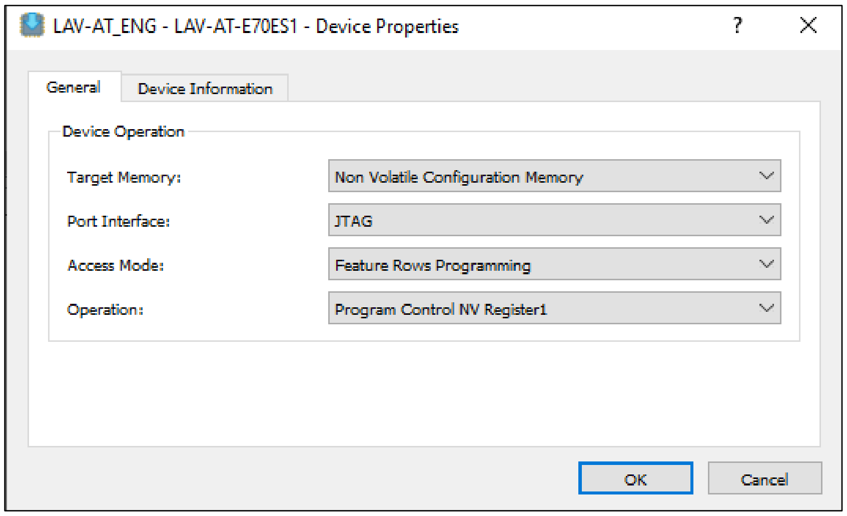
General | Device Information

Device Operation

Target Memory: Compressed Random Access Memory (CRAM)
Port Interface: JTAG
Access Mode: Direct Programming
Operation: Erase Only

Figure 7.17. Erase Only Operation for CRAM Programming

13. Skip steps 14 to 18 if you have performed the steps before. Otherwise, perform a one-time step required by JTAG to Program NV Register 1 to modify the default SPI Addressing and Command mode from 24 bits to 32 bits. You need to this only once for your board. Subsequent programming does not need to program NV Register 1 again.


LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

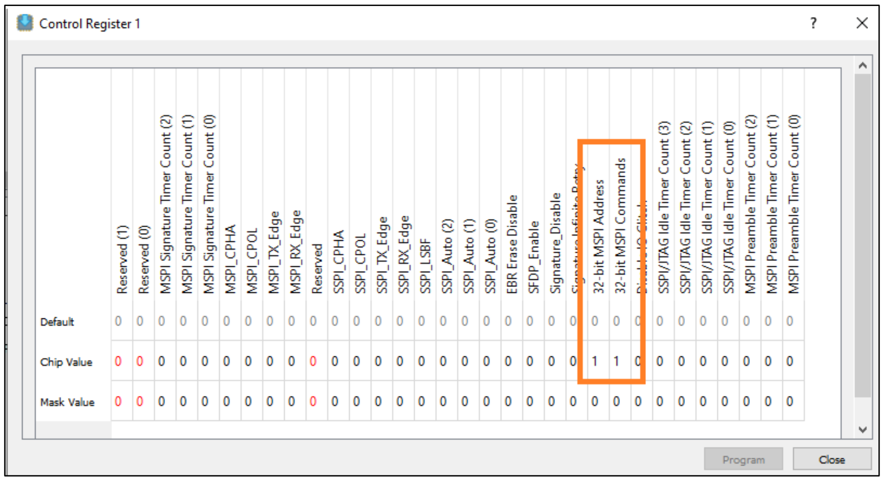
General | Device Information

Device Operation

Target Memory: Non Volatile Configuration Memory
Port Interface: JTAG
Access Mode: Feature Rows Programming
Operation: Program Control NV Register1

Figure 7.18. One-Time Programmable Control NV Register1

14. Click **OK** and click the **Program Device** Icon or go to the menu item, **Run > Program Device**.
15. Change bit 0 to 1 for 32-bit MSPI Address and 32-bit MSPI Commands as shown in the Figure 7.19. For changing these bits, just click once on those 0 in the Chip Value row.



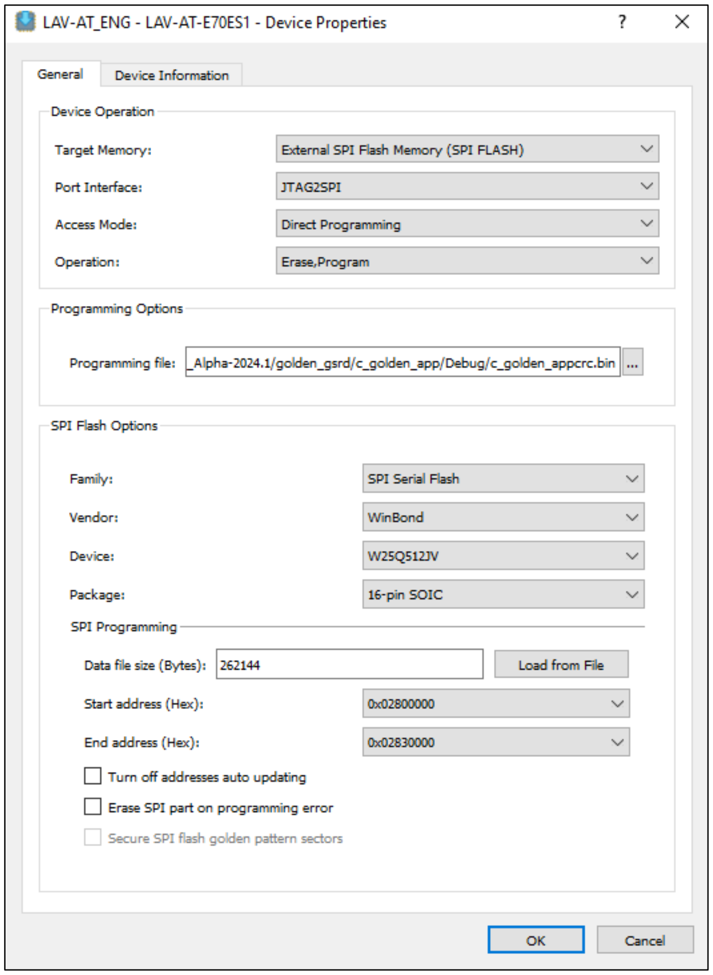
Control Register 1

	Reserved (1)	Reserved (0)	MSPI Signature Timer Count (2)	MSPI Signature Timer Count (1)	MSPI Signature Timer Count (0)	MSPI_CPHA	MSPI_CPOL	MSPI_TX_Edge	MSPI_RX_Edge	Reserved	SSPI_CPHA	SSPI_CPOL	SSPI_TX_Edge	SSPI_RX_Edge	SSPI_LSBF	SSPI_Auto (2)	SSPI_Auto (1)	SSPI_Auto (0)	EBR Erase Disable	SFDP_Enable	Signature_Disable	Signature Infinite Retry	32-bit MSPI Address	32-bit MSPI Commands	Disable IO Glitch	SSPI/JTAG Idle Timer Count (3)	SSPI/JTAG Idle Timer Count (2)	SSPI/JTAG Idle Timer Count (1)	SSPI/JTAG Idle Timer Count (0)	MSPI Preamble Timer Count (2)	MSPI Preamble Timer Count (1)	MSPI Preamble Timer Count (0)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Mask Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.19. Settings to Select Chip Value

Note: Update the value of the two highlighted fields. NV Register 1 is an OTP (One-Time-Programmable) Register.

16. Click **Program**.
17. Power cycle the Avant-AT-E Evaluation Board.
18. To program the **c_golden_appcrc.bin** file into the SPI Flash, follow the settings as shown in Figure 7.20.



LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

Device Operation

Target Memory: External SPI Flash Memory (SPI FLASH)
Port Interface: JTAG2SPI
Access Mode: Direct Programming
Operation: Erase,Program

Programming Options

Programming file: _Alpha-2024.1/golden_gsrd/c_golden_app/Debug/c_golden_appcrc.bin

SPI Flash Options

Family: SPI Serial Flash
Vendor: WinBond
Device: W25Q512JV
Package: 16-pin SOIC

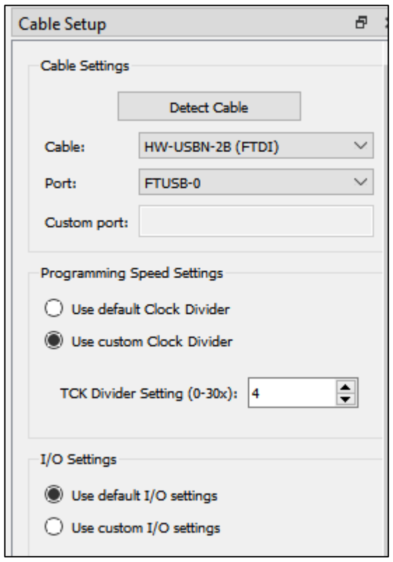
SPI Programming

Data file size (Bytes): 262144
Start address (Hex): 0x02800000
End address (Hex): 0x02830000

☐ Turn off addresses auto updating
☐ Erase SPI part on programming error
☐ Secure SPI flash golden pattern sectors

Figure 7.20. Device Properties to Program the SPI Flash

19. Click **OK**.
20. Use TCK Divider Setting (0-30x) to **4**.


Cable Setup

Cable Settings

Detect Cable

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

○ Use default Clock Divider

● Use custom Clock Divider

TCK Divider Setting (0-30x): 4

I/O Settings

● Use default I/O settings

○ Use custom I/O settings

Figure 7.21. Cable Settings for device programming

21. Click the **Program Device** Icon or go to the menu item, **Run > Program Device**. The output console displays the Operation Successful message. Note that the *Verifying…* display does not show.

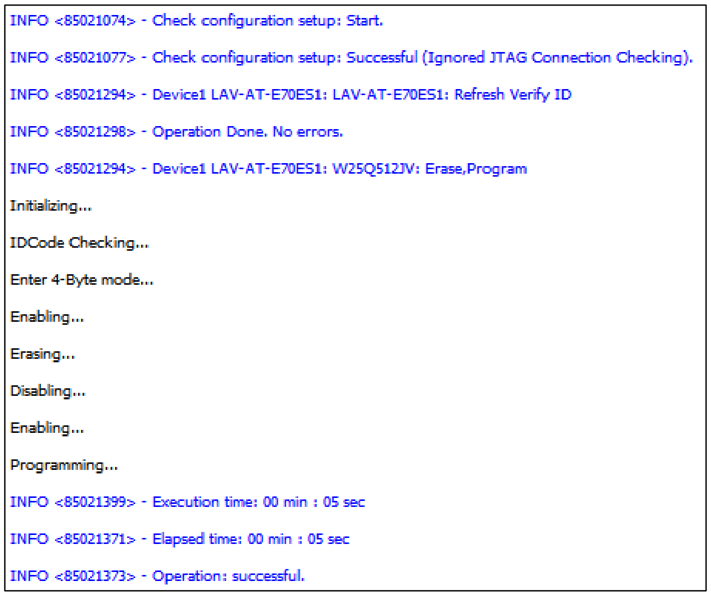
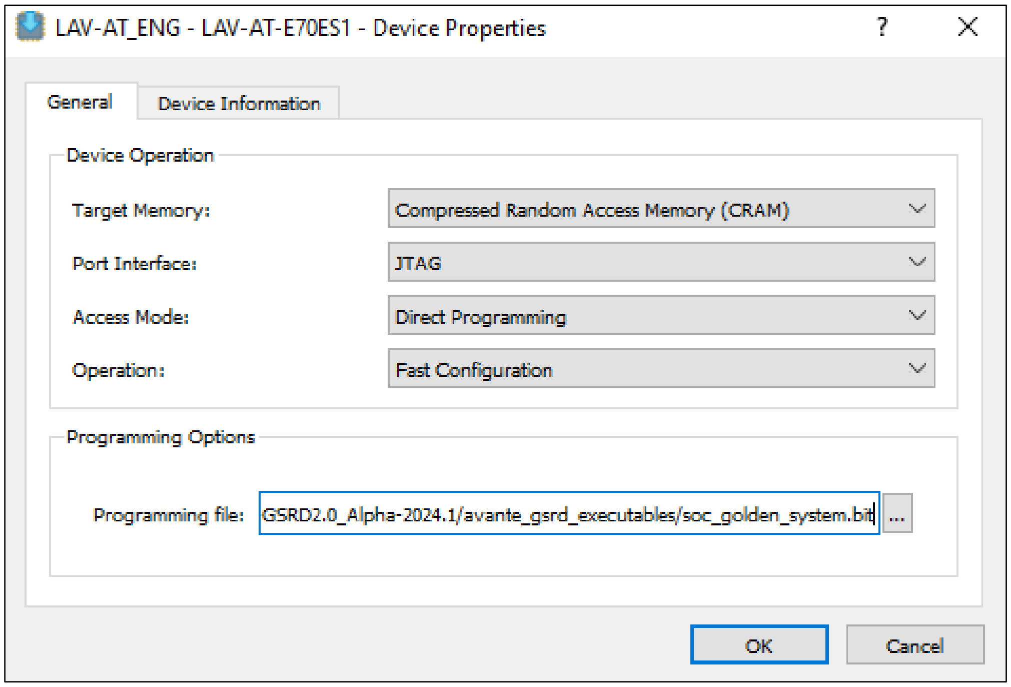

```
INFO <85021074> - Check configuration setup: Start.
INFO <85021077> - Check configuration setup: Successful (Ignored JTAG Connection Checking).
INFO <85021294> - Device1 LAV-AT-E70ES1: LAV-AT-E70ES1: Refresh Verify ID
INFO <85021298> - Operation Done. No errors.
INFO <85021294> - Device1 LAV-AT-E70ES1: W25Q512JV: Erase,Program
Initializing...
IDCode Checking...
Enter 4-Byte mode...
Enabling...
Erasing...
Disabling...
Enabling...
Programming...
INFO <85021399> - Execution time: 00 min : 05 sec
INFO <85021371> - Elapsed time: 00 min : 05 sec
INFO <85021373> - Operation: successful.
```

Figure 7.22. Radiant Programmer Console Output after Programming the SPI Flash

22. Power cycle the Avant-AT-E70 Evaluation board using **SW6.**
23. To program the FPGA Golden GSRD bitstream, double-click on the **Operation** tab to update the selections as shown in Figure 7.23. Make sure to provide the path to the .bit file location on your local machine where you have unzipped the executables.



Device Operation

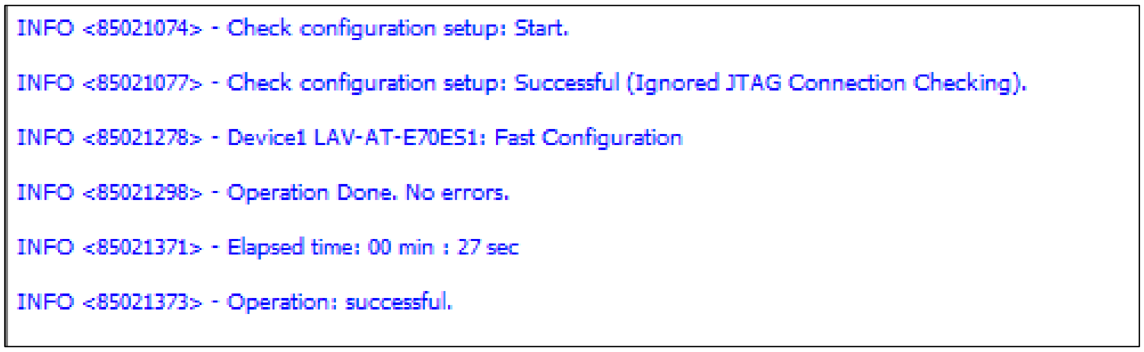
Target Memory: Compressed Random Access Memory (CRAM)
Port Interface: JTAG
Access Mode: Direct Programming
Operation: Fast Configuration

Programming Options

Programming file: GSRD2.0_Alpha-2024.1/avante_gsrd_executables/soc_golden_system.bit

Figure 7.23. Device Properties to Program the FPGA Bitstream in CRAM

24. Click **OK** and Click the **Program Device** Icon or go to the menu item**, Run > Program Device**. Wait until the operation is successful as shown in Figure 7.24.



```
INFO <85021074> - Check configuration setup: Start.
INFO <85021077> - Check configuration setup: Successful (Ignored JTAG Connection Checking).
INFO <85021278> - Device1 LAV-AT-E70ES1: Fast Configuration
INFO <85021298> - Operation Done. No errors.
INFO <85021371> - Elapsed time: 00 min : 27 sec
INFO <85021373> - Operation: successful.
```

Figure 7.24. Radiant Programmer Console Output after Bitstream is Programmed

25. Setup the UART terminal as mentioned in the Setting up the UART Terminal section.
26. Press **SW1 Reset** button on the board as highlighted in Figure 7.25.

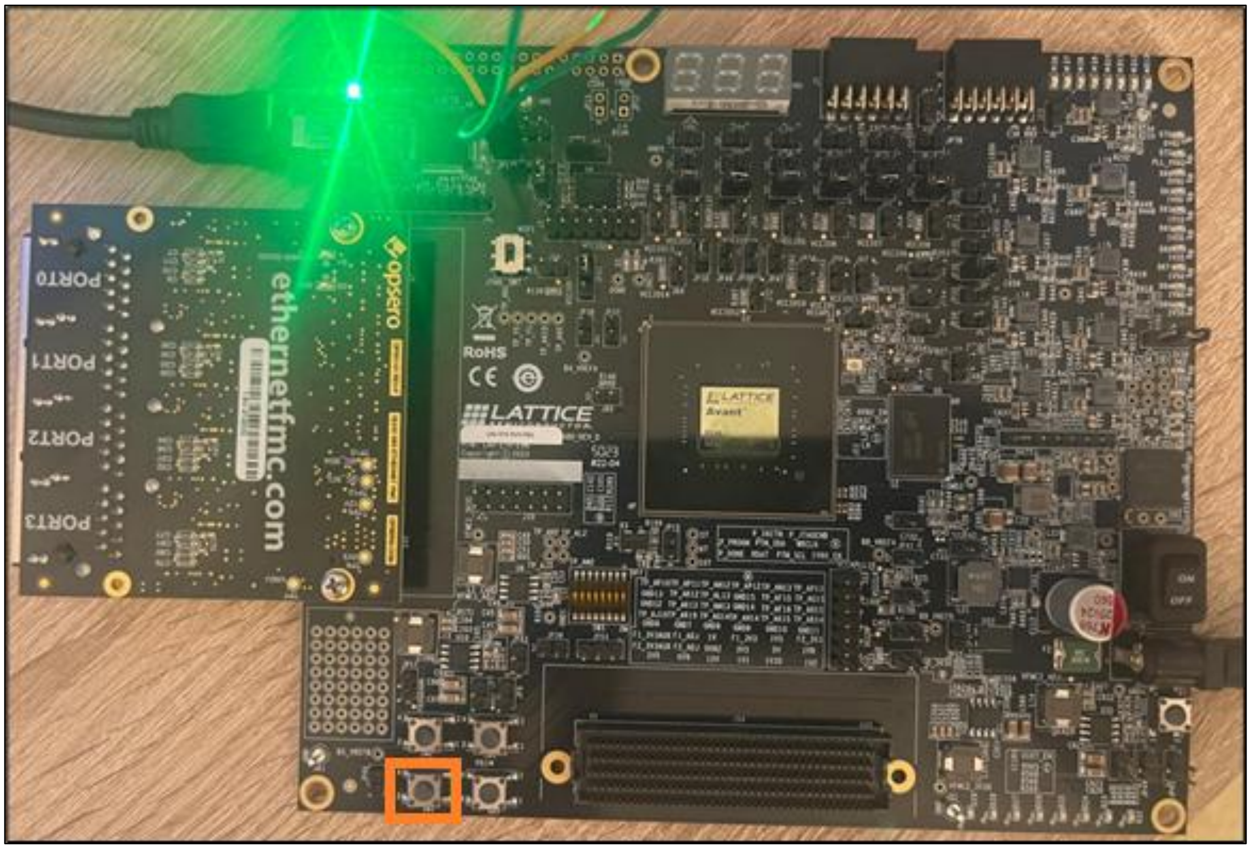
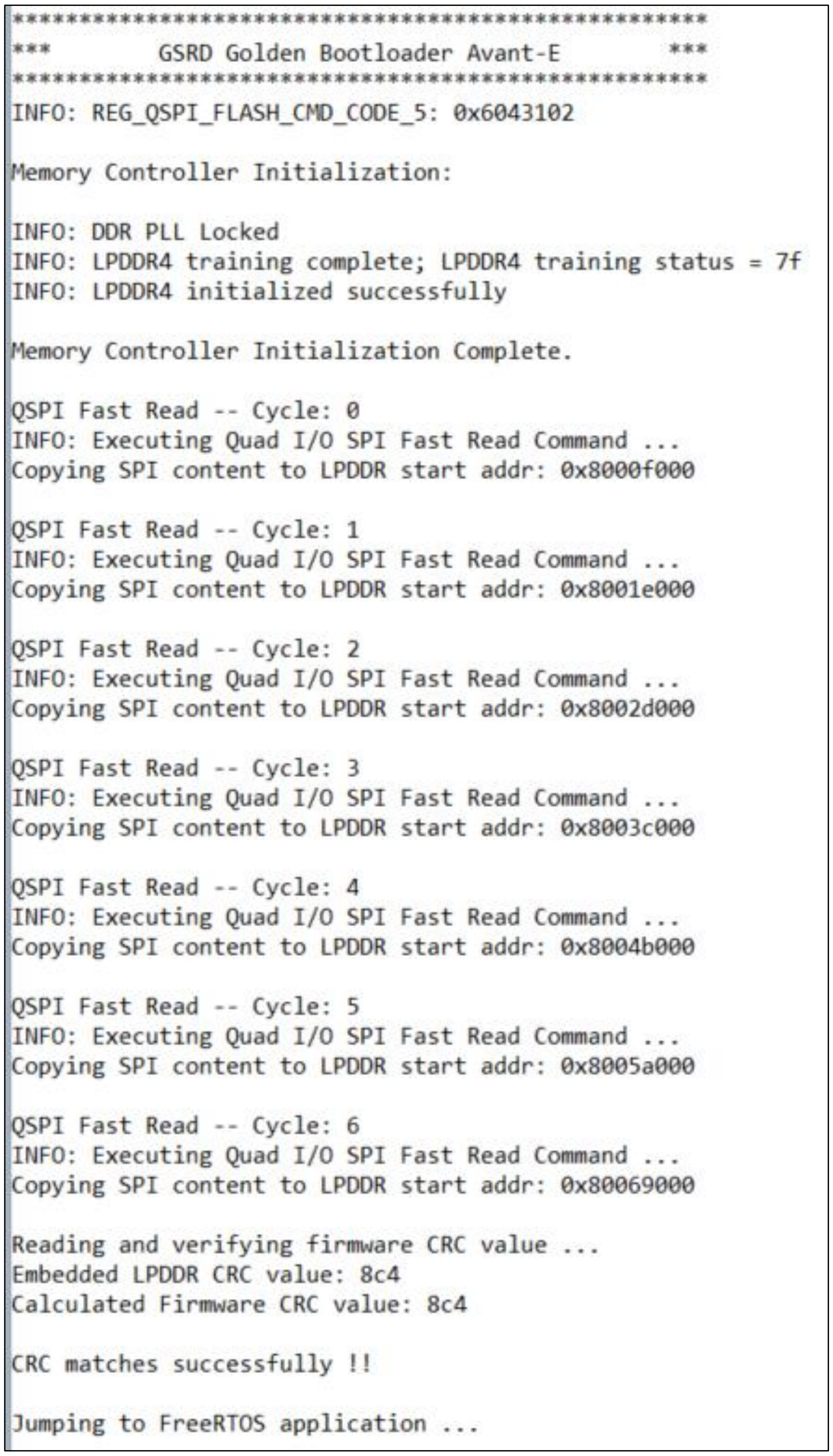


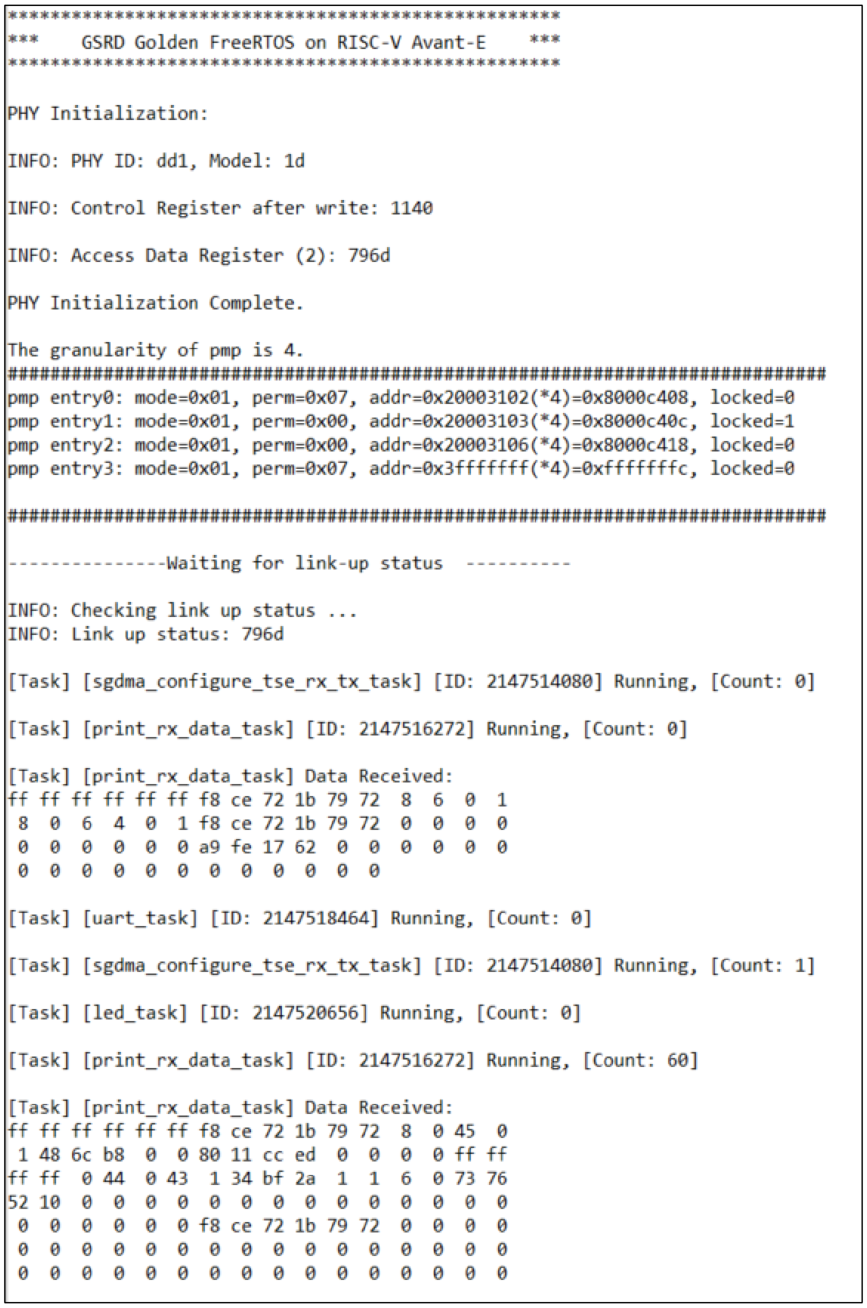
Figure 7.25. SW1 Reset Button

27. The Golden GSRD output results are as follows:
    a. Golden GSRD Bootloader
    b. LPDDR4 Configuration and Memory Training Status
    c. Ethernet PHY Configuration and Status
    d. LPDDR4 and QSPI CRC matches
    e. Application jumps to Golden FreeRTOS RISC-V.
    f. Waits for Ethernet Link Up. As soon as you plug the Ethernet Cable into Port0 of FMC, you can see the traffic.
    g. SGDMA Packet Received
    h. FreeRTOS Task Scheduler Running
    i. Hardware LED Status:
        - D12: PLL Lock
        - D13: LPDDR4 Init Done



```
****************************************************
***        GSRD Golden Bootloader Avant-E        ***
****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 8c4
Calculated Firmware CRC value: 8c4

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

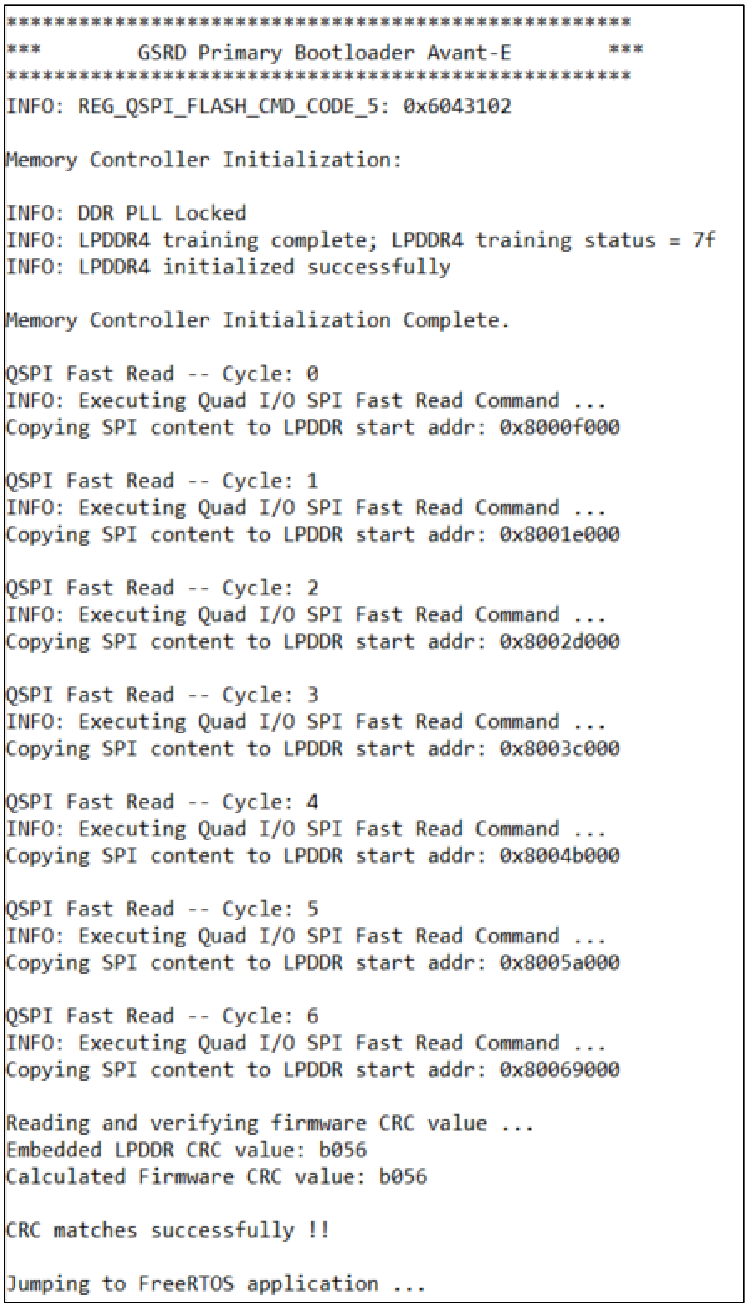
Figure 7.26. Golden GSRD - Output on UART Terminal for Bootloader and FreeRTOS Start



```
*****************************************************
***    GSRD Golden FreeRTOS on RISC-V Avant-E    ***
*****************************************************

PHY Initialization:

INFO: PHY ID: dd1, Model: 1d

INFO: Control Register after write: 1140

INFO: Access Data Register (2): 796d

PHY Initialization Complete.

The granularity of pmp is 4.
#############################################################################
pmp entry0: mode=0x01, perm=0x07, addr=0x20003102(*4)=0x8000c408, locked=0
pmp entry1: mode=0x01, perm=0x00, addr=0x20003103(*4)=0x8000c40c, locked=1
pmp entry2: mode=0x01, perm=0x00, addr=0x20003106(*4)=0x8000c418, locked=0
pmp entry3: mode=0x01, perm=0x07, addr=0x3fffffff(*4)=0xfffffffc, locked=0

#############################################################################

---------------Waiting for link-up status  ----------

INFO: Checking link up status ...
INFO: Link up status: 796d

[Task] [sgdma_configure_tse_rx_tx_task] [ID: 2147514080] Running, [Count: 0]

[Task] [print_rx_data_task] [ID: 2147516272] Running, [Count: 0]

[Task] [print_rx_data_task] Data Received:
ff ff ff ff ff ff f8 ce 72 1b 79 72  8  6  0  1
 8  0  6  4  0  1 f8 ce 72 1b 79 72  0  0  0  0
 0  0  0  0  0  0 a9 fe 17 62  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0

[Task] [uart_task] [ID: 2147518464] Running, [Count: 0]

[Task] [sgdma_configure_tse_rx_tx_task] [ID: 2147514080] Running, [Count: 1]

[Task] [led_task] [ID: 2147520656] Running, [Count: 0]

[Task] [print_rx_data_task] [ID: 2147516272] Running, [Count: 60]

[Task] [print_rx_data_task] Data Received:
ff ff ff ff ff ff f8 ce 72 1b 79 72  8  0 45  0
 1 48 6c b8  0  0 80 11 cc ed  0  0  0  0 ff ff
ff ff  0 44  0 43  1 34 bf 2a  1  1  6  0 73 76
52 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0 f8 ce 72 1b 79 72  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

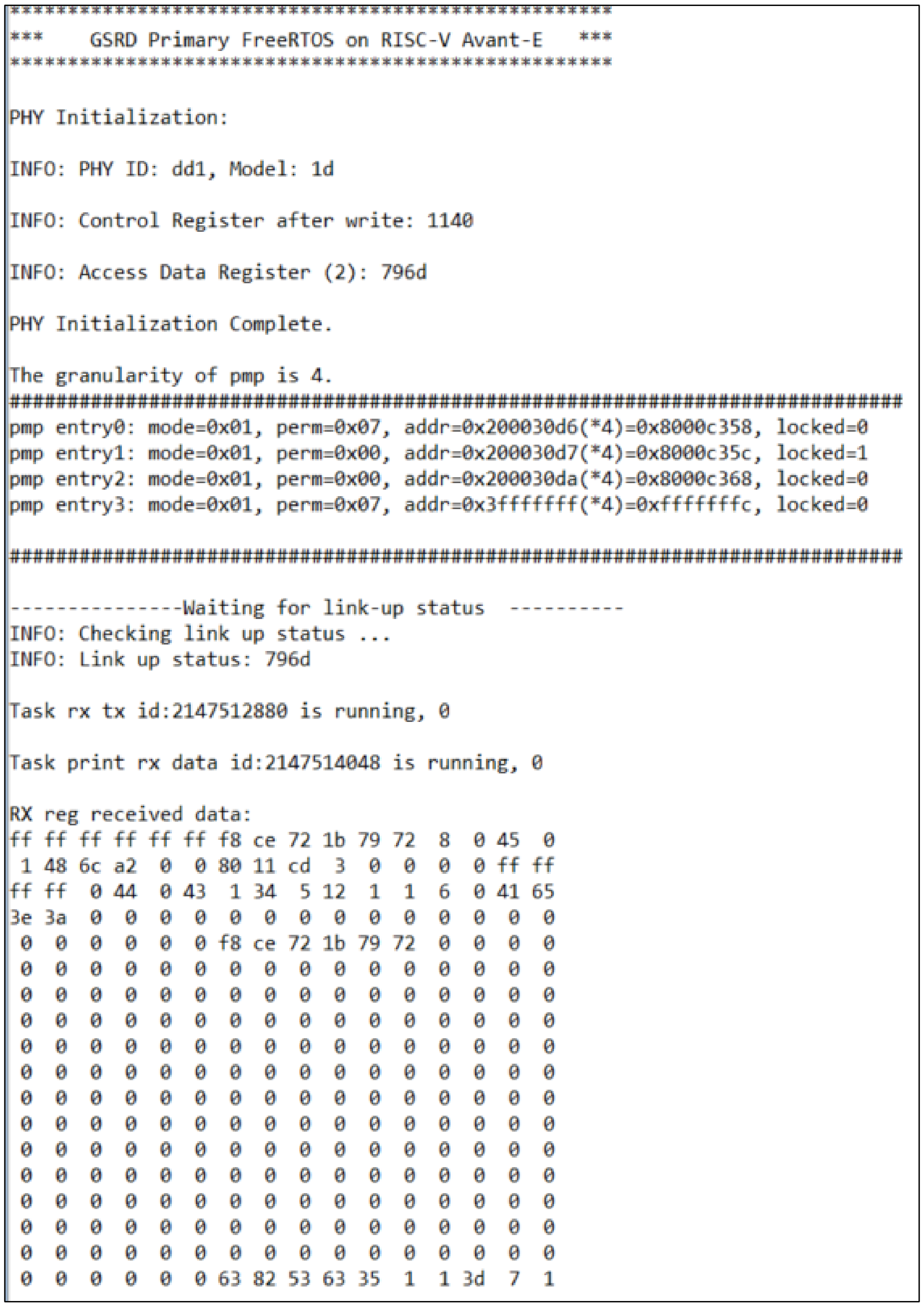
Figure 7.27. Golden GSRD - Output on UART Terminal for FreeRTOS Running

28. Follow the same steps 5 to Step 25 to load the Primary GSRD Software and Bitstream. For Primary App CRC software, refer to the Executables section for the folder and file names.
29. The Primary GSRD output results are as follows:



```
*****************************************************
***        GSRD Primary Bootloader Avant-E        ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: b056
Calculated Firmware CRC value: b056

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

Figure 7.28. Primary GSRD Bootloader– Output on UART Terminal



```
*****************************************************
***    GSRD Primary FreeRTOS on RISC-V Avant-E    ***
*****************************************************

PHY Initialization:

INFO: PHY ID: dd1, Model: 1d

INFO: Control Register after write: 1140

INFO: Access Data Register (2): 796d

PHY Initialization Complete.

The granularity of pmp is 4.
###############################################################################
pmp entry0: mode=0x01, perm=0x07, addr=0x200030d6(*4)=0x8000c358, locked=0
pmp entry1: mode=0x01, perm=0x00, addr=0x200030d7(*4)=0x8000c35c, locked=1
pmp entry2: mode=0x01, perm=0x00, addr=0x200030da(*4)=0x8000c368, locked=0
pmp entry3: mode=0x01, perm=0x07, addr=0x3fffffff(*4)=0xfffffffc, locked=0

###############################################################################

---------------Waiting for link-up status  ----------
INFO: Checking link up status ...
INFO: Link up status: 796d

Task rx tx id:2147512880 is running, 0

Task print rx data id:2147514048 is running, 0

RX reg received data:
ff ff ff ff ff ff f8 ce 72 1b 79 72  8  0 45  0
 1 48 6c a2  0  0 80 11 cd  3  0  0  0  0 ff ff
ff ff  0 44  0 43  1 34  5 12  1  1  6  0 41 65
3e 3a  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0 f8 ce 72 1b 79 72  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0 63 82 53 63 35  1  1 3d  7  1
```

Figure 7.29. Primary GSRD FreeRTOS– Output on UART Terminal

7.3.4. Programming the Golden, Primary Software and MCS file

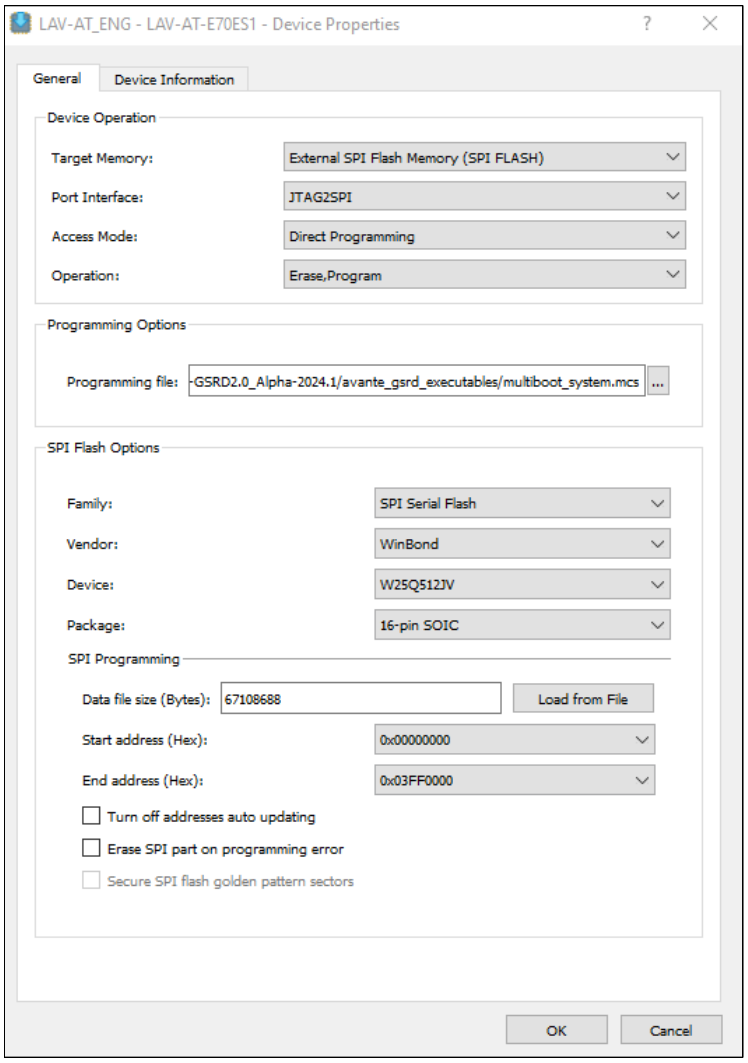
This section describes the three scenarios of GSRD boot-up.

Note: The intent of the three scenarios is to showcase how manual and automatic multi-boot results would look like. Scenario 1 should be enough for you to test out the multi-boot. Scenarios 2 and 3 are merely to showcase the different multi-boot scenarios in GSRD.

7.3.4.1. Scenario 1: Manual Booting when both Primary and Golden FW are with CRC with MCS File Programming

1. Follow Steps 1 to 9 from the Programming the Golden, Primary Software and MCS file section. Do not power-cycle the board.
2. Keep the folder*/file of executables* handy.

3. To program the **multiboot_system.mcs** file, locate the file in the executables folder that you downloaded and add the file in the Programming file area. Start Address and End Address is allocated automatically. Confirm the settings as shown in Figure 7.30.



Device Operation

Target Memory: External SPI Flash Memory (SPI FLASH)
Port Interface: JTAG2SPI
Access Mode: Direct Programming
Operation: Erase,Program

Programming Options

Programming file: -GSRD2.0_Alpha-2024.1/avante_gsrd_executables/multiboot_system.mcs

SPI Flash Options

Family: SPI Serial Flash
Vendor: WinBond
Device: W25Q512JV
Package: 16-pin SOIC

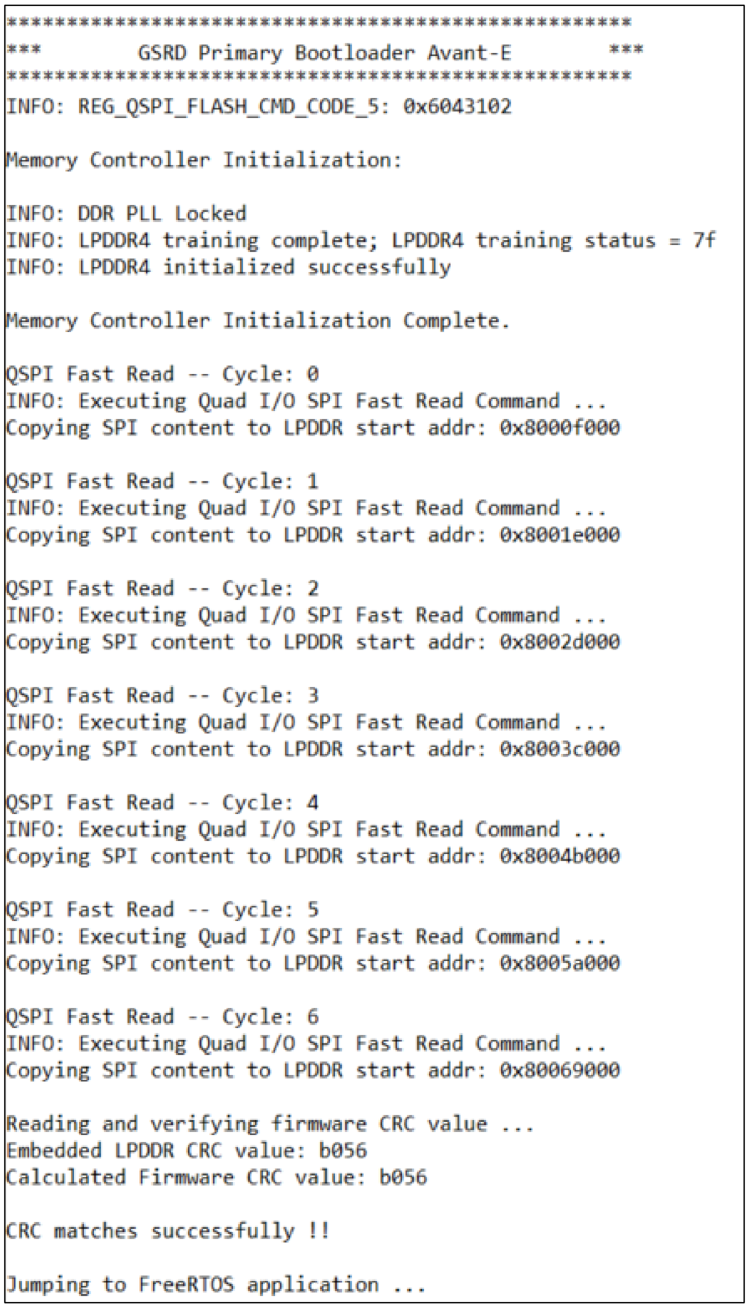
SPI Programming

Data file size (Bytes): 67108688 Load from File
Start address (Hex): 0x00000000
End address (Hex): 0x03FF0000
☐ Turn off addresses auto updating
☐ Erase SPI part on programming error
☐ Secure SPI flash golden pattern sectors

Figure 7.30.Device Properties window to setup MCS programming file

4. Click **OK** and the **Program Device** icon or go to the menu item **Run > Program Device**. Wait until the operation is successful, which takes about 30-40 minutes.
5. Do not power-cycle the board yet.
6. Program both the **primary_appcrc.bin** and **golden_appcrc.bin** file as per Programming the Golden, Primary Software and MCS file section, step 17. Make sure to confirm that Start Address (Hex) for both the binaries as per Executables section.
7. Once both binaries are programmed, switch off the board.
8. Move **SW7 to MSPI** mode.
9. Switch on the board.
10. Setup the UART terminal as mentioned in the Setting up the UART Terminal section.
11. Wait for about 20 to 30 seconds for the FPGA to load the bitstream from the flash. Press **SW1** Reset button.

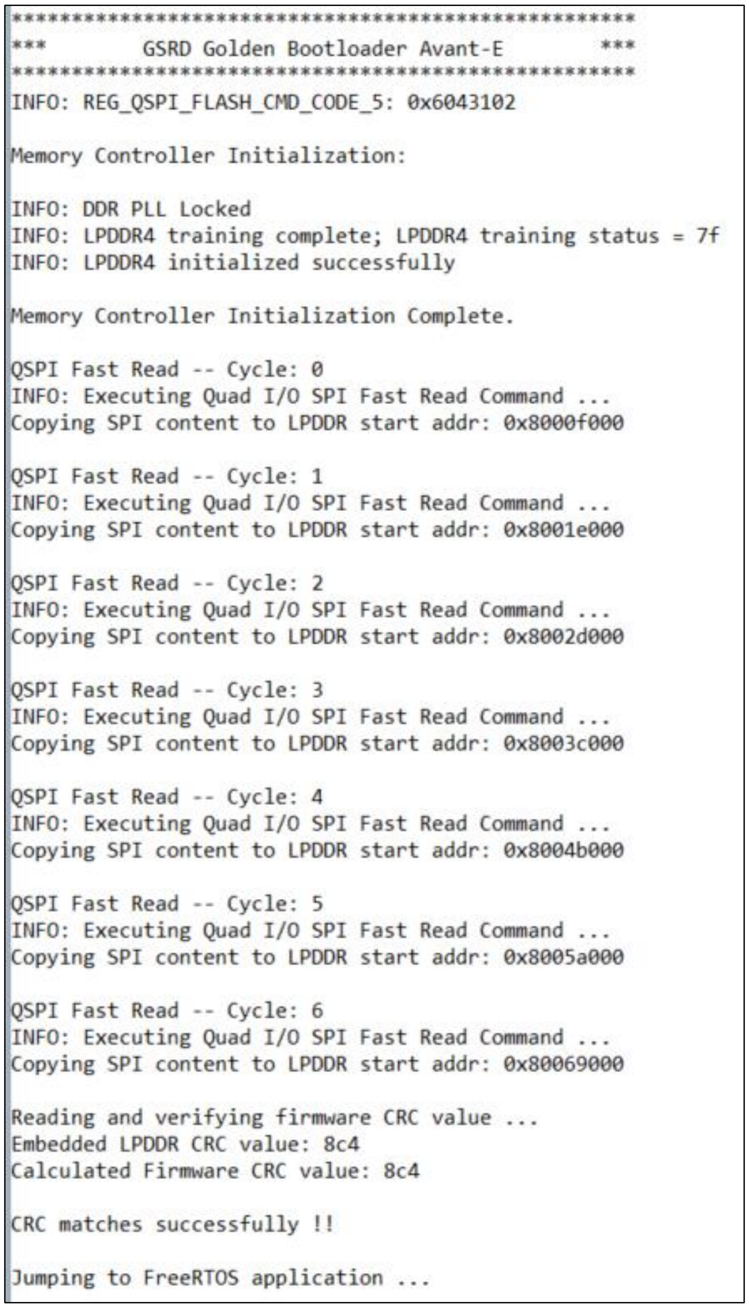
12. Results on UART Terminal is displayed as shown in Figure 7.31.
    - It loads the Primary GSRD project.
    - If you press the SW1 button again, it loads the same Primary GSRD project.



```
****************************************************
***         GSRD Primary Bootloader Avant-E        ***
****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: b056
Calculated Firmware CRC value: b056

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

Figure 7.31. UART Terminal Output after Power-Cycling Board with MCS and Binaries Programmed

13. For the manual multi-boot, press **SW2 (PROGRAMN)** yellow button from the Hardware Setup section on the board just above the **SW1 Reset** button.
14. The results on UART terminal are displayed as shown in Figure 7.32. It switches/jumps to Golden GSRD project and stays there unless the **SW2 PROGRAMN** button is pressed.

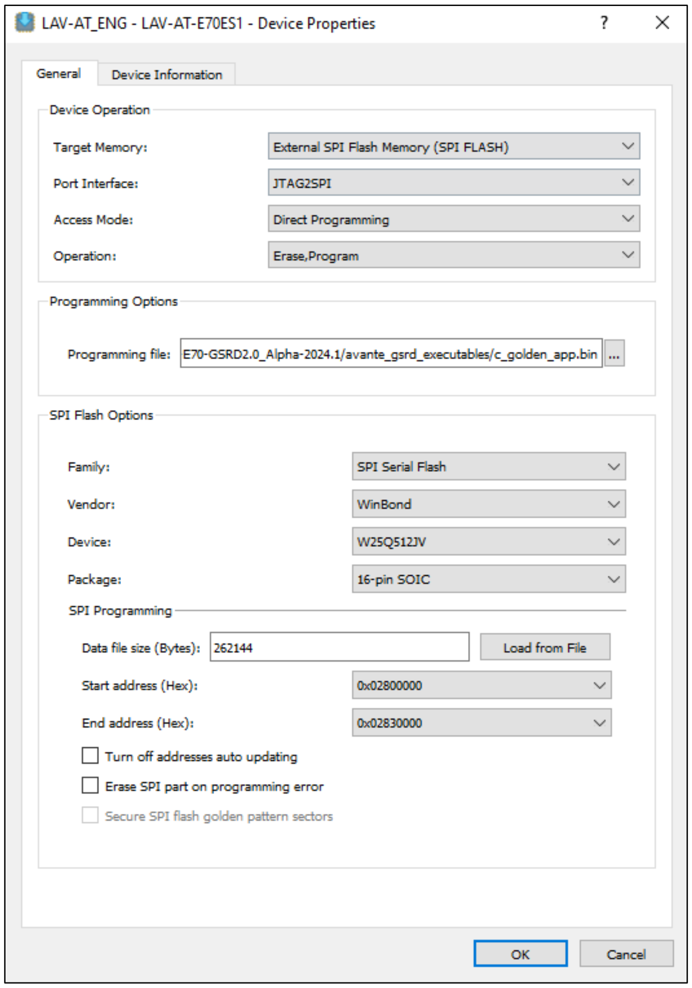


```
*****************************************************
***         GSRD Golden Bootloader Avant-E        ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 8c4
Calculated Firmware CRC value: 8c4

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

Figure 7.32. Switches to Golden GSRD upon SW2 PROGRAMN Button

7.3.4.2. Scenario 2: Manual Booting when Primary FW is with CRC, and Golden FW is without CRC

1. Power-off the board.
2. Keep SW7 on JTAG mode.
3. Power-on the board.
4. Program the MCS file as shown in the Scenario 1: Manual Booting when both Primary and Golden FW are with CRC with MCS File Programming section, step 2 to 5. If you have already run the scenario 1 entirely, skip this step. Otherwise, follow steps 2 to 5.
5. To program the **c_golden_app.bin** software file into the SPI Flash, follow the settings as shown in Figure 7.33.


LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

Device Operation

- Target Memory: External SPI Flash Memory (SPI FLASH)
- Port Interface: JTAG2SPI
- Access Mode: Direct Programming
- Operation: Erase,Program

Programming Options

- Programming file: E70-GSRD2.0_Alpha-2024.1/avante_gsrd_executables/c_golden_app.bin

SPI Flash Options

- Family: SPI Serial Flash
- Vendor: WinBond
- Device: W25Q512JV
- Package: 16-pin SOIC

SPI Programming

- Data file size (Bytes): 262144
- Start address (Hex): 0x02800000
- End address (Hex): 0x02830000
- ☐ Turn off addresses auto updating
- ☐ Erase SPI part on programming error
- ☐ Secure SPI flash golden pattern sectors

Figure 7.33. Device Properties settings to program the Golden Application

6. Click **OK** and the **Program Device** Icon or go to **Run > Program Device**. Wait until the operation is successful.

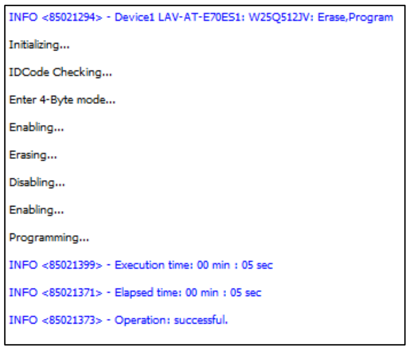
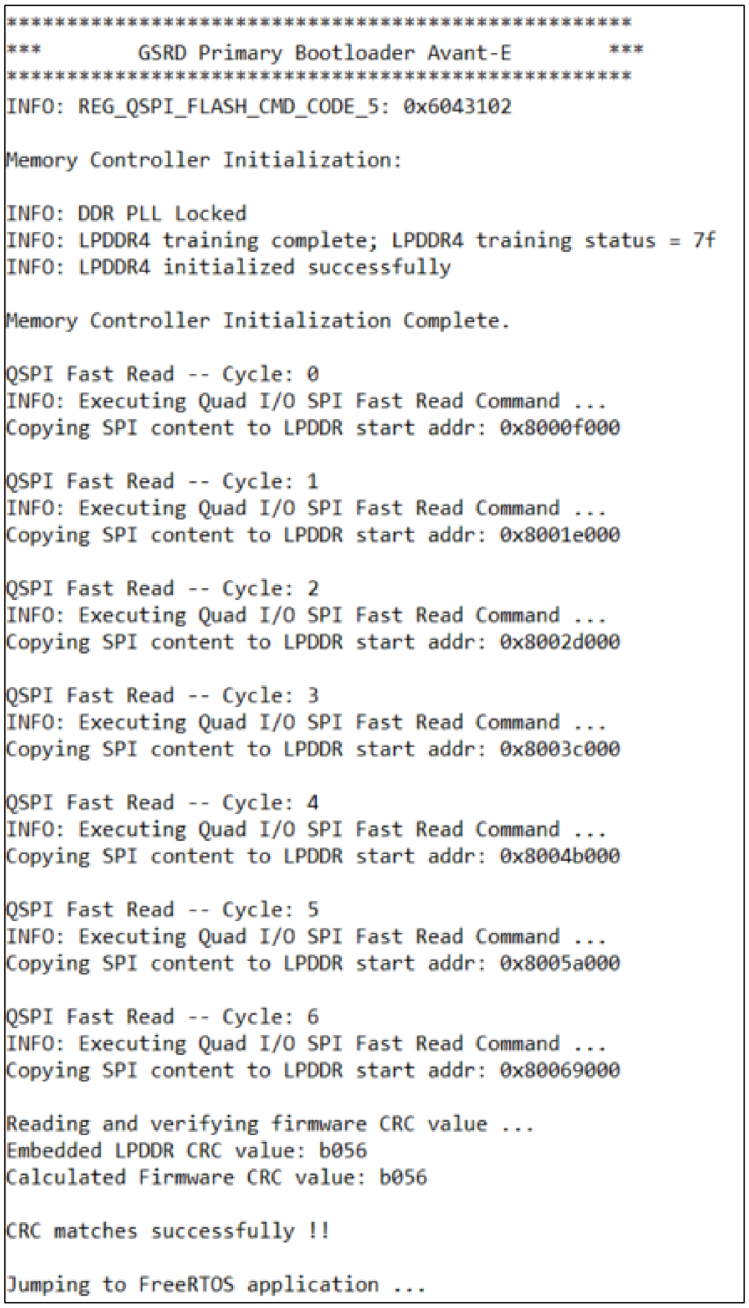

```
INFO <85021294> - Device1 LAV-AT-E70ES1: W25Q512JV: Erase,Program
Initializing...
IDCode Checking...
Enter 4-Byte mode...
Enabling...
Erasing...
Disabling...
Enabling...
Programming...
INFO <85021399> - Execution time: 00 min : 05 sec
INFO <85021371> - Elapsed time: 00 min : 05 sec
INFO <85021373> - Operation: successful.
```

Figure 7.34. Radiant Programmer Console Output After Programming

7. Switch off the board.
8. Move SW7 to MSPI Mode.
9. Switch on the board.
10. Setup UART terminal as mentioned in the Setting up the UART Terminal section.
11. Wait for about 20-30 seconds for the FPGA to load the bitstream from the flash press the **SW1 Reset** button.

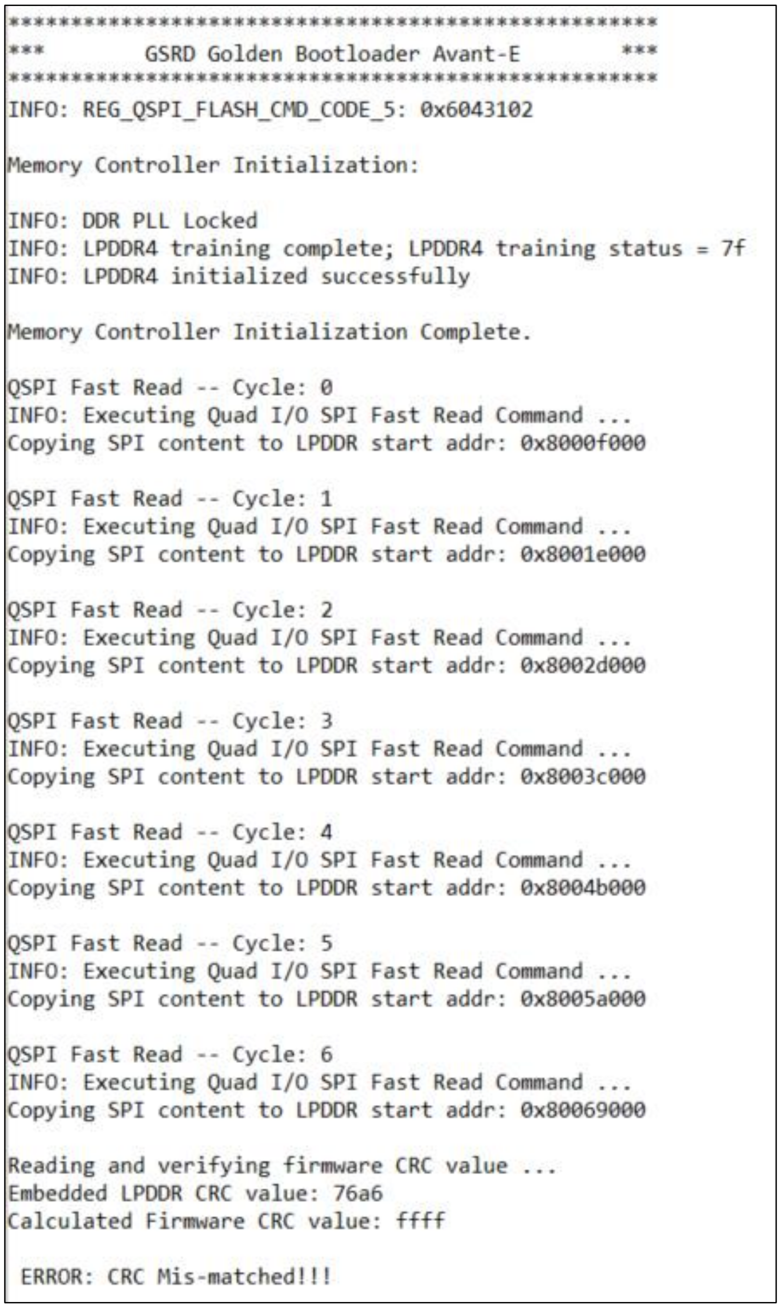
12. The results on the UART terminal are displayed as shown in Figure 7.35 and loads the Primary GSRD project.



```
*****************************************************
***          GSRD Primary Bootloader Avant-E         ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: b056
Calculated Firmware CRC value: b056

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

Figure 7.35. Primary GSRD – UART Output after SW1 Reset where CRC Matched

13. Manually press the **SW2 PROGRAMN** button.
14. Results on the UART terminal are displayed as shown in Figure 7.36. It jumps to Golden GSRD project and shows as CRC mismatch and stays there.



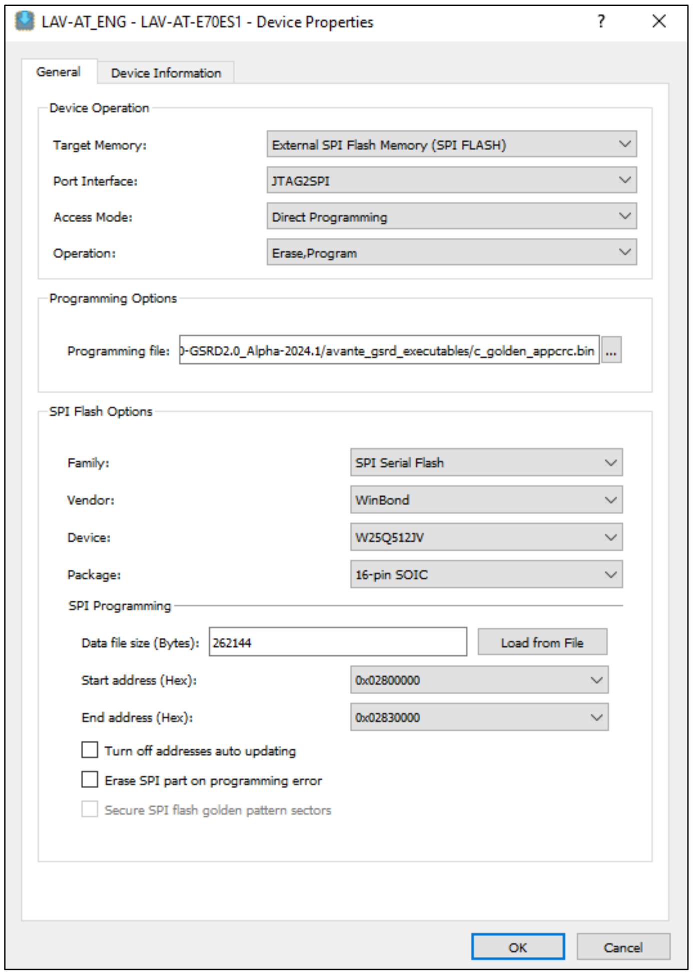
```
****************************************************
***        GSRD Golden Bootloader Avant-E        ***
****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 76a6
Calculated Firmware CRC value: ffff

 ERROR: CRC Mis-matched!!!
```

Figure 7.36. UART Output Switching to Golden GSRD where CRC Mis-matched Intentionally

15. Press **SW2 PROGRAMN** button, and it loads back the Primary GSRD project.

7.3.4.3. Scenario 3: Automatic Booting when Primary FW is without CRC and Golden FW has CRC

1. Power-off the board.
2. Keep SW7 on JTAG mode.
3. Power-on the board.
4. Program the MCS file as shown in the Scenario 1: Manual Booting when both Primary and Golden FW are with CRC with MCS File Programming section, Step 2 to 5. If you have already run the Scenario 1 entirely, skip this step. Otherwise, follow steps 2 to 5.
5. To program the **c_golden_appcrc.bin** software file into the SPI Flash, Follow the settings as shown in Figure 7.37.



LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

General | Device Information

Device Operation

Target Memory: External SPI Flash Memory (SPI FLASH)
Port Interface: JTAG2SPI
Access Mode: Direct Programming
Operation: Erase,Program

Programming Options

Programming file:)-GSRD2.0_Alpha-2024.1/avante_gsrd_executables/c_golden_appcrc.bin

SPI Flash Options

Family: SPI Serial Flash
Vendor: WinBond
Device: W25Q512JV
Package: 16-pin SOIC

SPI Programming

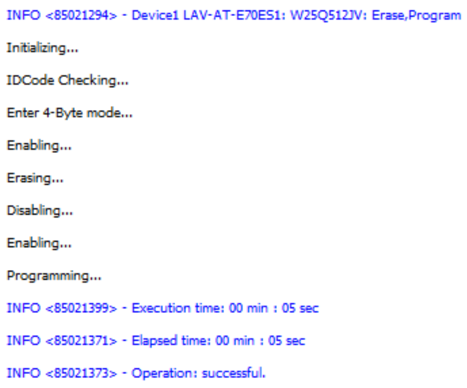
Data file size (Bytes): 262144 Load from File
Start address (Hex): 0x02800000
End address (Hex): 0x02830000

☐ Turn off addresses auto updating
☐ Erase SPI part on programming error
☐ Secure SPI flash golden pattern sectors

OK Cancel

Figure 7.37. Device Properties for Programming Golden App Binary with CRC

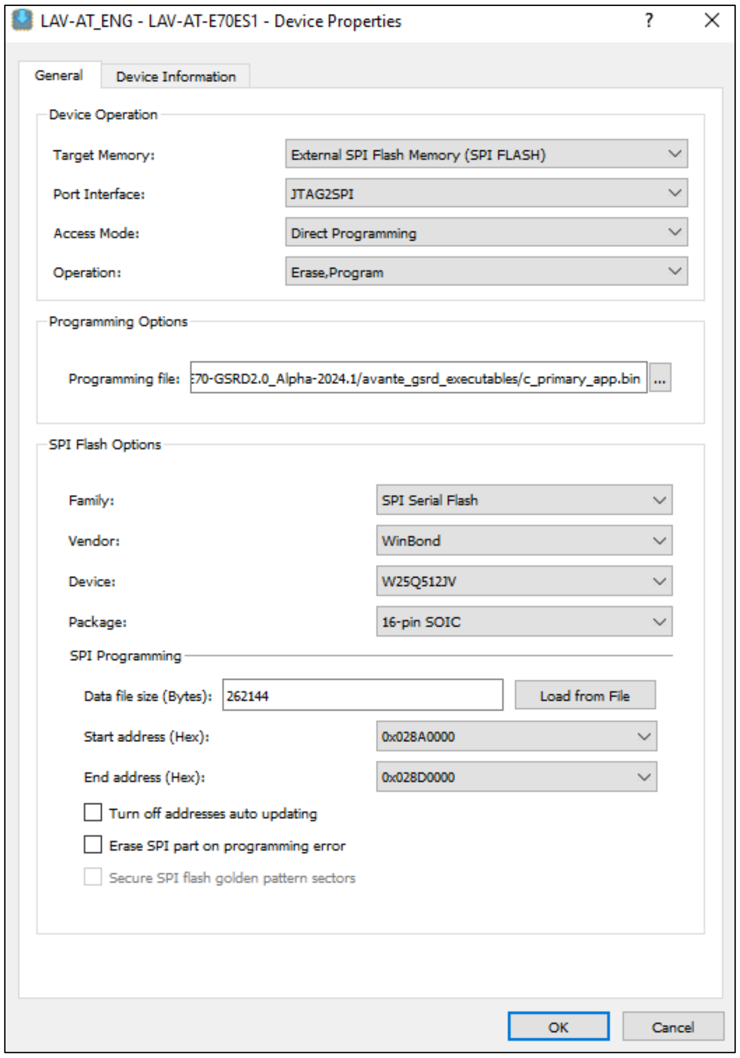
6. Click **OK** and the **Program Device** icon or go to **Run > Program Device**. Wait until the operation is successful.



```
INFO <85021294> - Device1 LAV-AT-E70ES1: W25Q512JV: Erase,Program
Initializing...
IDCode Checking...
Enter 4-Byte mode...
Enabling...
Erasing...
Disabling...
Enabling...
Programming...
INFO <85021399> - Execution time: 00 min : 05 sec
INFO <85021371> - Elapsed time: 00 min : 05 sec
INFO <85021373> - Operation: successful.
```

Figure 7.38. Radiant Programmer Console Output After Successful Programming

7. To program the **c_primary_app.bin** software file into the SPI Flash, Follow the settings as shown in Figure 7.39.


LAV-AT_ENG - LAV-AT-E70ES1 - Device Properties

Device Operation

Target Memory: External SPI Flash Memory (SPI FLASH)
Port Interface: JTAG2SPI
Access Mode: Direct Programming
Operation: Erase,Program

Programming Options

Programming file: E70-GSRD2.0_Alpha-2024.1/avante_gsrd_executables/c_primary_app.bin

SPI Flash Options

Family: SPI Serial Flash
Vendor: WinBond
Device: W25Q512JV
Package: 16-pin SOIC

SPI Programming

Data file size (Bytes): 262144
Start address (Hex): 0x028A0000
End address (Hex): 0x028D0000

[ ] Turn off addresses auto updating
[ ] Erase SPI part on programming error
[ ] Secure SPI flash golden pattern sectors

Figure 7.39. Device Properties Window for Primary App Binary without CRC

8. Click **OK** and the **Program Device** icon or go to **Run > Program Device**. Wait until the operation is successful.

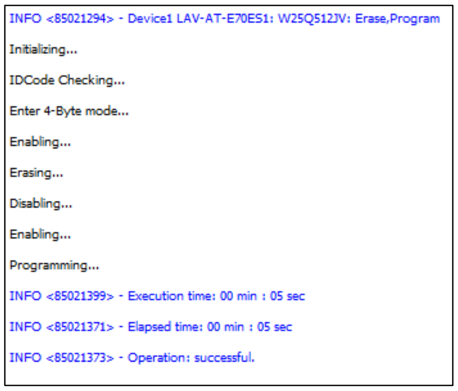
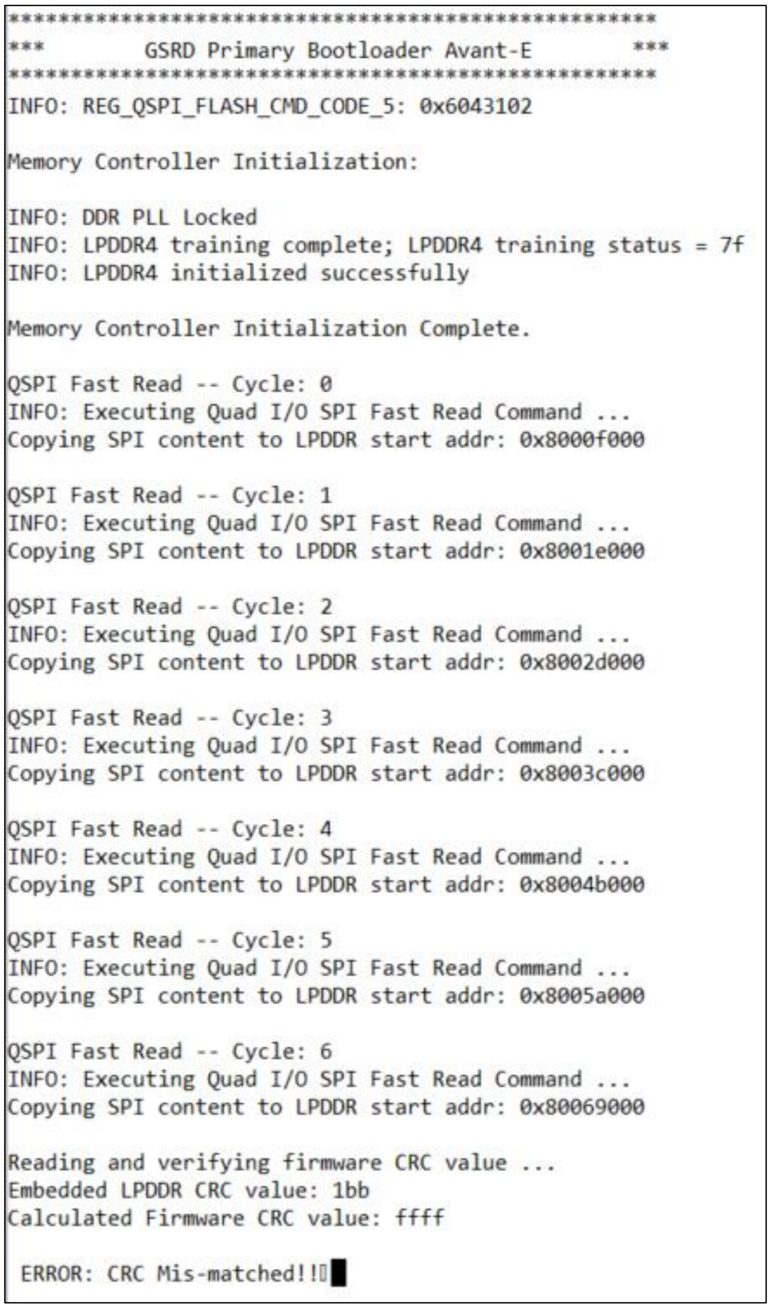

```
INFO <85021294> - Device1 LAV-AT-E70ES1: W25Q512JV: Erase,Program
Initializing...
IDCode Checking...
Enter 4-Byte mode...
Enabling...
Erasing...
Disabling...
Enabling...
Programming...
INFO <85021399> - Execution time: 00 min : 05 sec
INFO <85021371> - Elapsed time: 00 min : 05 sec
INFO <85021373> - Operation: successful.
```

Figure 7.40. Radiant Programmer Console Output After Successful Programming

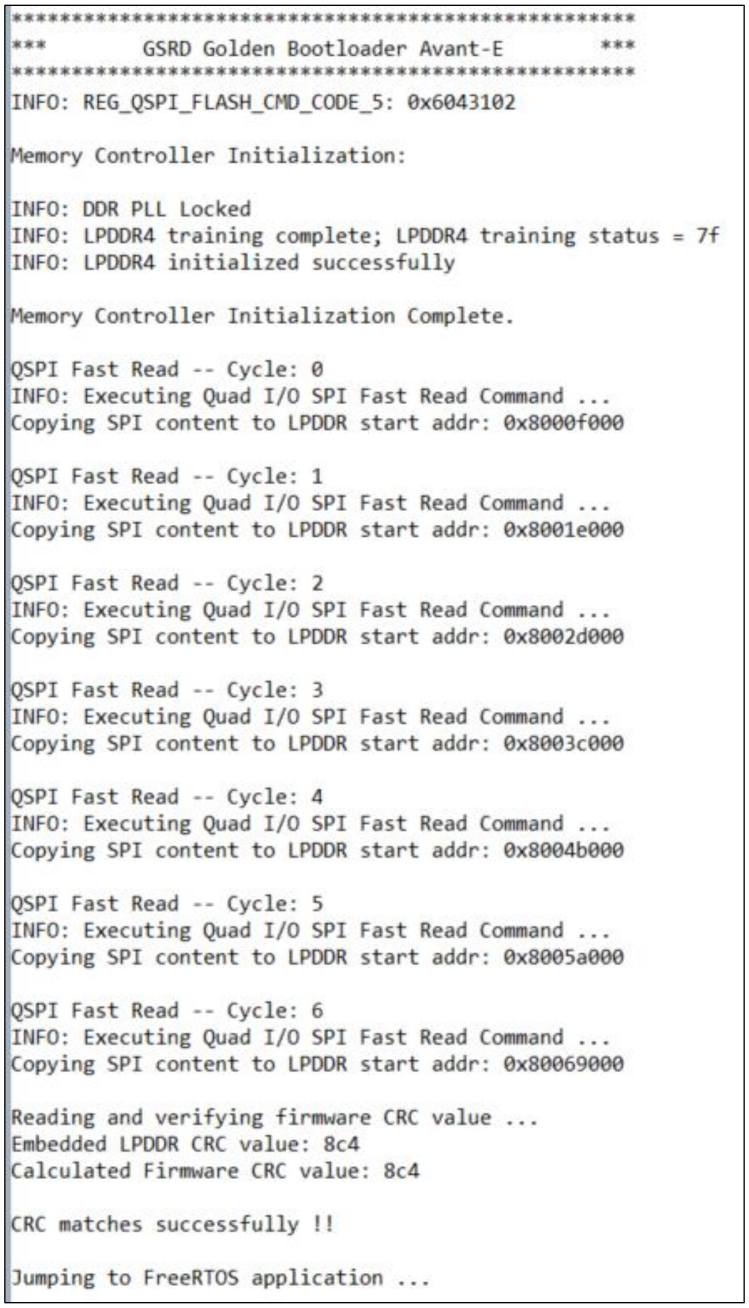
9. Switch off the board.
10. Move **SW7 to MSPI Mode.**

11. Switch on the board.
12. Setup UART terminal as mentioned in the Setting up the UART Terminal section.
13. Wait about 20 to 30 seconds for the FPGA to load the bitstream from the flash. Press **SW1 Reset** button.
14. The results on the UART terminal are displayed as shown in Figure 7.41.
    - It loads the Primary GSRD project first.
    - As soon as the FW CRC for Primary GSRD mismatches, it triggers a soft reset or automatic PROGRAMN.
    - It automatically loads the Golden GSRD bitstream and software.



```
*****************************************************
***        GSRD Primary Bootloader Avant-E        ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 1bb
Calculated Firmware CRC value: ffff

 ERROR: CRC Mis-matched!!
```

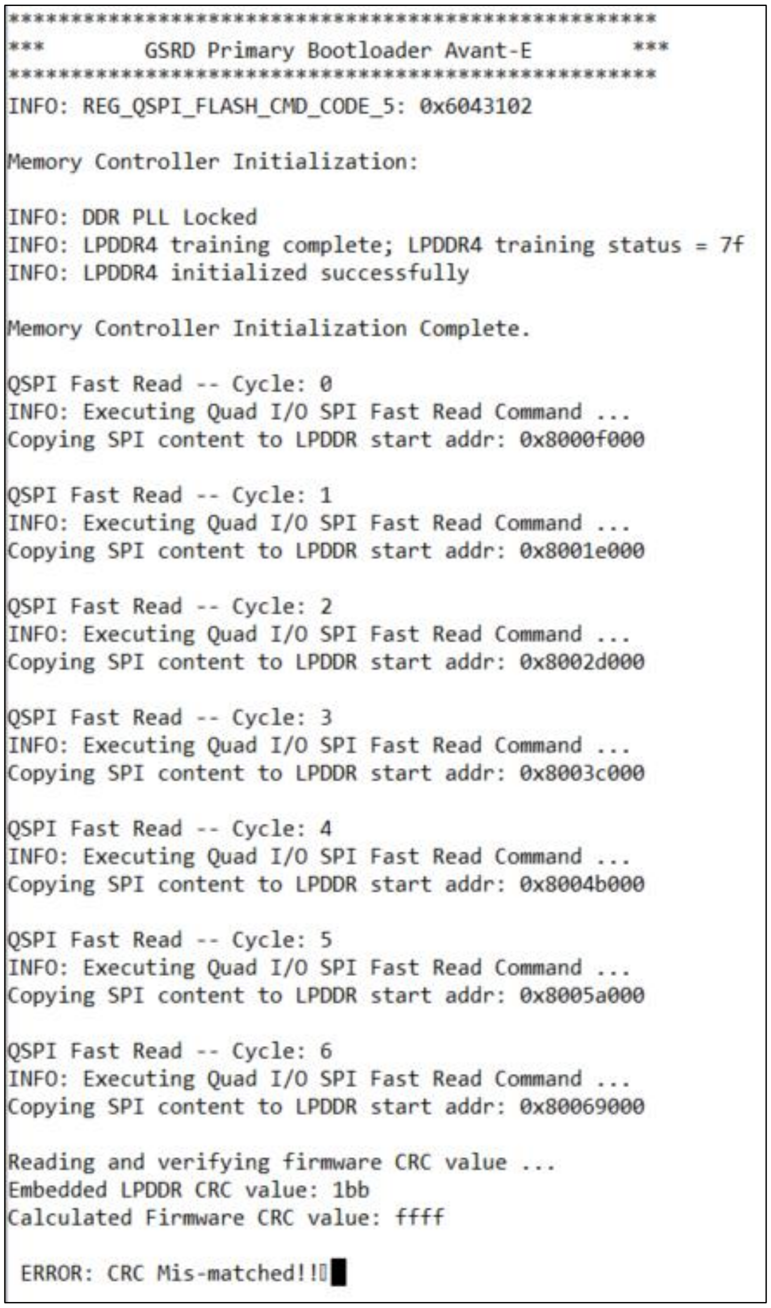
Figure 7.41. Primary GSRD without CRC Fails – UART Terminal



```
*****************************************************
***        GSRD Golden Bootloader Avant-E        ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 8c4
Calculated Firmware CRC value: 8c4

CRC matches successfully !!

Jumping to FreeRTOS application ...
```

Figure 7.42. Automatically Jumps to Boot-up Golden GSRD

15. Manually press **SW2 PROGRAMN** button.
16. The results on the UART terminal are displayed as shown in Figure 7.43. It jumps back to Primary GSRD where CRC fails again and switches to working the Golden GSRD project and stays there.



```
*****************************************************
***        GSRD Primary Bootloader Avant-E        ***
*****************************************************
INFO: REG_QSPI_FLASH_CMD_CODE_5: 0x6043102

Memory Controller Initialization:

INFO: DDR PLL Locked
INFO: LPDDR4 training complete; LPDDR4 training status = 7f
INFO: LPDDR4 initialized successfully

Memory Controller Initialization Complete.

QSPI Fast Read -- Cycle: 0
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8000f000

QSPI Fast Read -- Cycle: 1
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8001e000

QSPI Fast Read -- Cycle: 2
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8002d000

QSPI Fast Read -- Cycle: 3
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8003c000

QSPI Fast Read -- Cycle: 4
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8004b000

QSPI Fast Read -- Cycle: 5
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x8005a000

QSPI Fast Read -- Cycle: 6
INFO: Executing Quad I/O SPI Fast Read Command ...
Copying SPI content to LPDDR start addr: 0x80069000

Reading and verifying firmware CRC value ...
Embedded LPDDR CRC value: 1bb
Calculated Firmware CRC value: ffff

 ERROR: CRC Mis-matched!!
```

Figure 7.43. UART Terminal Output - After Pressing SW2 PROGRAMN Button

17. Connect the Ethernet cable to see the traffic.
18. Re-plug the Ethernet cable in the case of FreeRTOS stop at Ethernet link-up message.

8. Compiling the Reference Design

This section describes the process of compiling the GSRD/GHRD Reference Design. You must always start with the Golden GHRD/GSRD project which is a part of the Propel Template. Compilation is required to generate the necessary binary files and bitstreams from the source files. The compilation process involves the following software tools for generating the FPGA bitstream and software executable files.

- Lattice Propel Builder 2024.1 Patch
- Lattice Propel Software Development Kit (SDK) 2024.1
- Lattice Radiant Software 2024.1.1

These sections show the typical design and compilation follow for GSRD design.

- Building Bootloader and FreeRTOS binary files using Lattice Propel SDK
- Validating and generating the GSRD design using Lattice Propel Builder
- Synthesizing the RTL files and generating the bitstream using Lattice Radiant Software
- Generating the Multi-Boot MCS File

8.1. Building the Bootloader and FreeRTOS Binary Files Using Lattice Propel SDK

To build the Bootloader and FreeRTOS binary files:

1. Create a folder for your project on your PC.
2. Launch Propel SDK 2024.1 application.

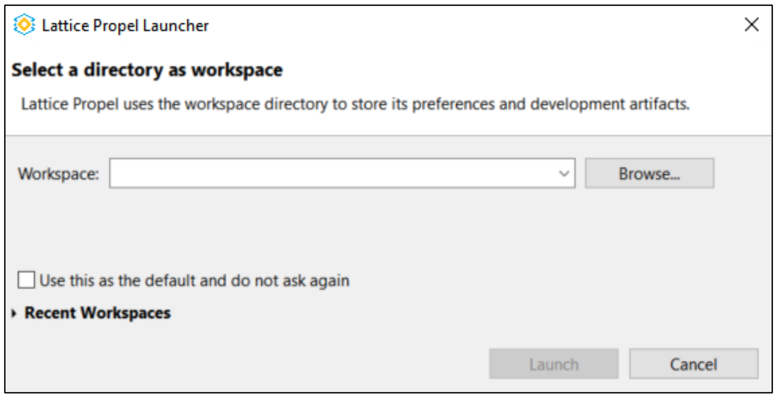
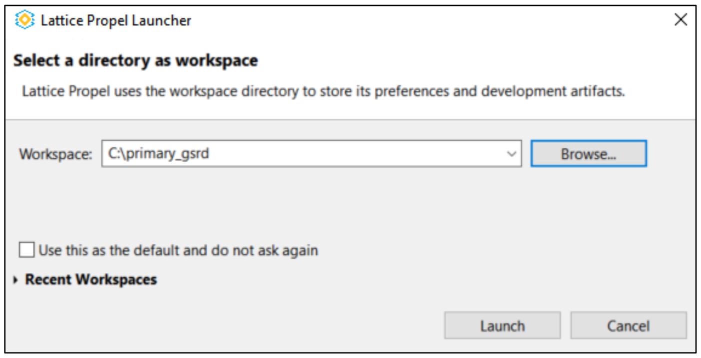

Lattice Propel Launcher
Select a directory as workspace
Lattice Propel uses the workspace directory to store its preferences and development artifacts.
Workspace:
Browse...
Use this as the default and do not ask again
Recent Workspaces
Launch
Cancel

Figure 8.1. Propel Builder Launcher

3. To select the workspace, browse to the created folder by clicking on the **Browse** button as shown in Figure 8.2 and click on **Launch** to launch the workspace.

   Note: Name given below is Primary GSRD because you may be using this template to create your own project titles and make modifications on top of Golden SoC/C Templates.


Lattice Propel Launcher
Select a directory as workspace
Lattice Propel uses the workspace directory to store its preferences and development artifacts.
Browse...
Use this as the default and do not ask again
Recent Workspaces
Launch
Cancel

Workspace: C:\primary_gsrd

Figure 8.2. Provide Name for Workspace Directory

4. Go to **File > New > Lattice SoC Design Project**.

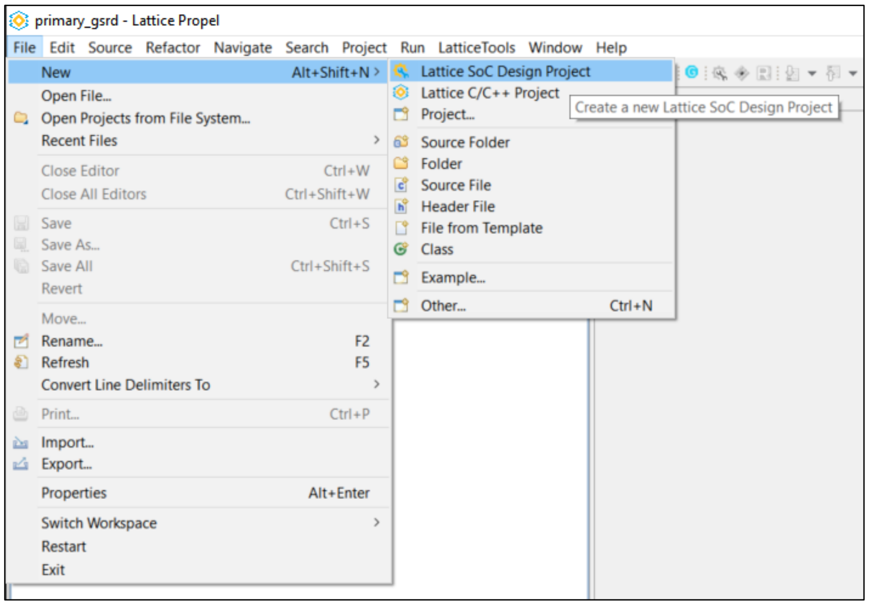
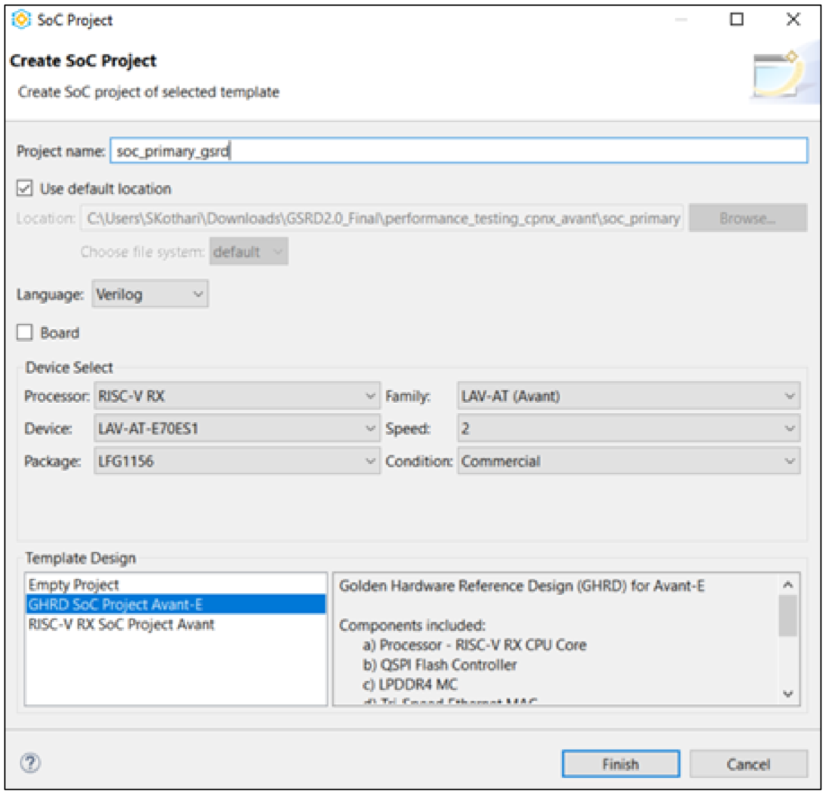


Figure 8.3. Creating Lattice SoC Design Project

5. On the SoC Project window, provide name for your SoC project. Choose Processor as **RISC-V RX**, Family as **LAV-AT**, Device as **LAV-AT-E70ES1**, Package as **LFG1156**, and choose the GHRD SoC Project Avant-AT-E from Template Design as shown in Figure 8.4. Since you can make your own desired modifications to this project, the project name is called as **soc_primary_gsrd**.


SoC Project

Create SoC Project

Create SoC project of selected template

Project name: soc_primary_gsrd

☑ Use default location

Location: C:\Users\SKothari\Downloads\GSRD2.0_Final\performance_testing_cpnx_avant\soc_primary

Choose file system: default

Language: Verilog

☐ Board

Device Select

Processor:	RISC-V RX	Family:	LAV-AT (Avant)
Device:	LAV-AT-E70ES1	Speed:	2
Package:	LFG1156	Condition:	Commercial

Template Design

Empty Project
GHRD SoC Project Avant-E
RISC-V RX SoC Project Avant

Golden Hardware Reference Design (GHRD) for Avant-E

Components included:
a) Processor - RISC-V RX CPU Core
b) QSPI Flash Controller
c) LPDDR4 MC

Figure 8.4. SoC Project Window

6. Click **Finish** and it launches the SoC in Propel Builder. This generates the HDL files for IPs instantiated in the project.

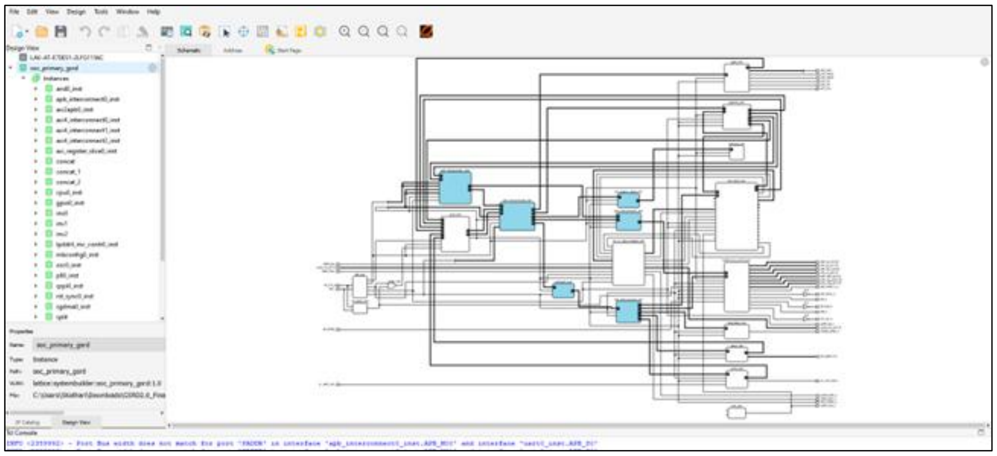


Figure 8.5. Launched SoC in Propel Builder

7. Click on **Design > Validate Design**.

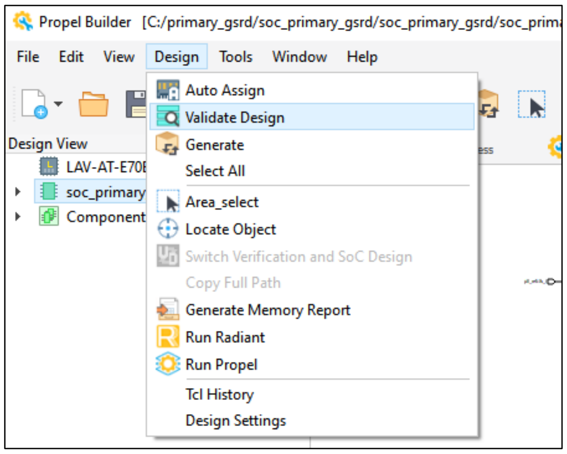
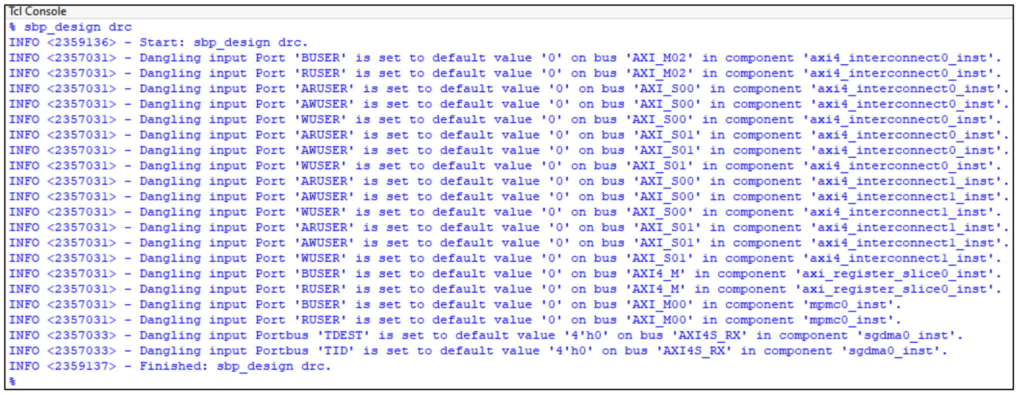


Figure 8.6. Validate Design in Propel Builder

8. The TCL Console should be as follows.



```
Tcl Console
% sbp_design drc
INFO <2359136> - Start: sbp_design drc.
INFO <2357031> - Dangling input Port 'BUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'RUSER' is set to default value '0' on bus 'AXI_M02' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect0_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S00' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'ARUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'AWUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'WUSER' is set to default value '0' on bus 'AXI_S01' in component 'axi4_interconnect1_inst'.
INFO <2357031> - Dangling input Port 'BUSER' is set to default value '0' on bus 'AXI4_M' in component 'axi_register_slice0_inst'.
INFO <2357031> - Dangling input Port 'RUSER' is set to default value '0' on bus 'AXI4_M' in component 'axi_register_slice0_inst'.
INFO <2357031> - Dangling input Port 'BUSER' is set to default value '0' on bus 'AXI_M00' in component 'mpmc0_inst'.
INFO <2357031> - Dangling input Port 'RUSER' is set to default value '0' on bus 'AXI_M00' in component 'mpmc0_inst'.
INFO <2357033> - Dangling input Portbus 'TDEST' is set to default value '4'h0' on bus 'AXI4S_RX' in component 'sgdma0_inst'.
INFO <2357033> - Dangling input Portbus 'TID' is set to default value '4'h0' on bus 'AXI4S_RX' in component 'sgdma0_inst'.
INFO <2359137> - Finished: sbp_design drc.
%
```

Figure 8.7. TCL Console Output after Validating Design

9. Click on **Design > Generate.**

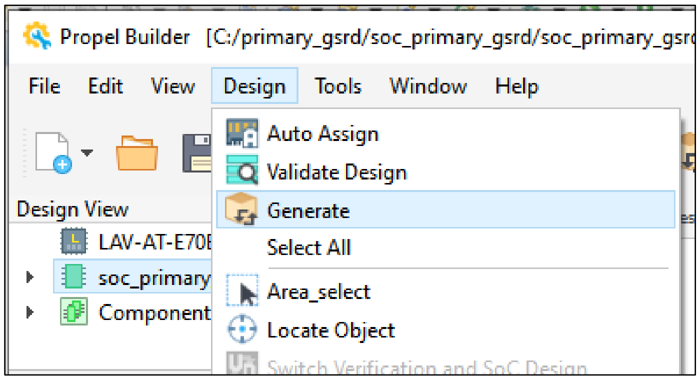

Propel Builder [C:/primary_gsrd/soc_primary_gsrd/soc_primary_gsrd
File Edit View Design Tools Window Help
Auto Assign
Validate Design
Generate
Select All
Area_select
Locate Object
Switch Verification and SoC Design
Design View
LAV-AT-E70E
soc_primary
Component

Figure 8.8. Generate in Propel Builder

10. The TCL Console should be as follows. Note that the WARNING and INFO messages are expected.

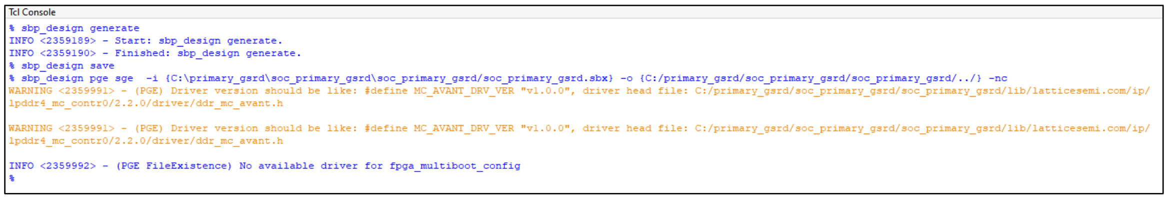

```
Tcl Console
% sbp_design generate
INFO <2359189> - Start: sbp_design generate.
INFO <2359190> - Finished: sbp_design generate.
% sbp_design save
% sbp_design pge sge  -i {C:\primary_gsrd\soc_primary_gsrd\soc_primary_gsrd/soc_primary_gsrd.sbx} -o {C:/primary_gsrd/soc_primary_gsrd/soc_primary_gsrd/../} -nc
WARNING <2359991> - (PGE) Driver version should be like: #define MC_AVANT_DRV_VER "v1.0.0", driver head file: C:/primary_gsrd/soc_primary_gsrd/soc_primary_gsrd/lib/latticesemi.com/ip/
lpddr4_mc_contr0/2.2.0/driver/ddr_mc_avant.h

WARNING <2359991> - (PGE) Driver version should be like: #define MC_AVANT_DRV_VER "v1.0.0", driver head file: C:/primary_gsrd/soc_primary_gsrd/soc_primary_gsrd/lib/latticesemi.com/ip/
lpddr4_mc_contr0/2.2.0/driver/ddr_mc_avant.h

INFO <2359992> - (PGE FileExistence) No available driver for fpga_multiboot_config
%
```

Figure 8.9. TCL Console Output after Generating Design.

11. Notice that after Generating the Design, it creates the sys_env.xml file in your Project Directory as shown in Figure 8.10.

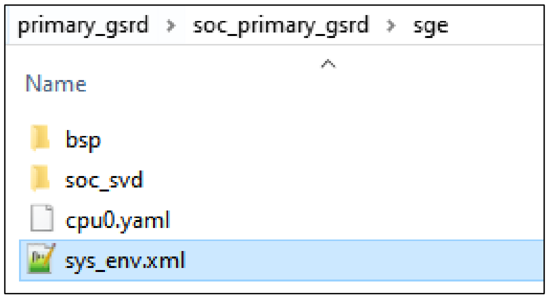


Figure 8.10. sys_env.xml File Created

12. Proceed to the Propel SDK window. Click on **File > New > Lattice C/C++ Project**.

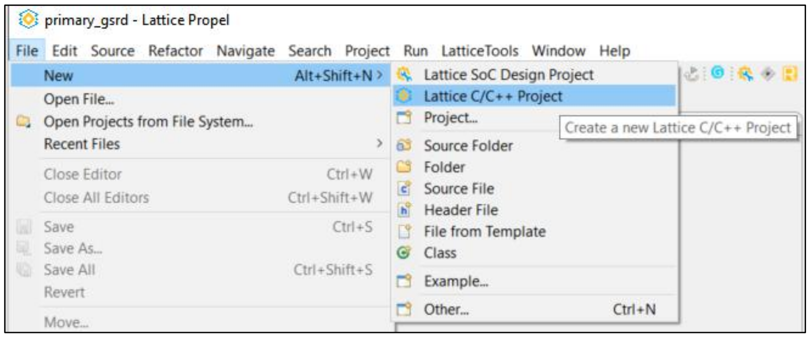
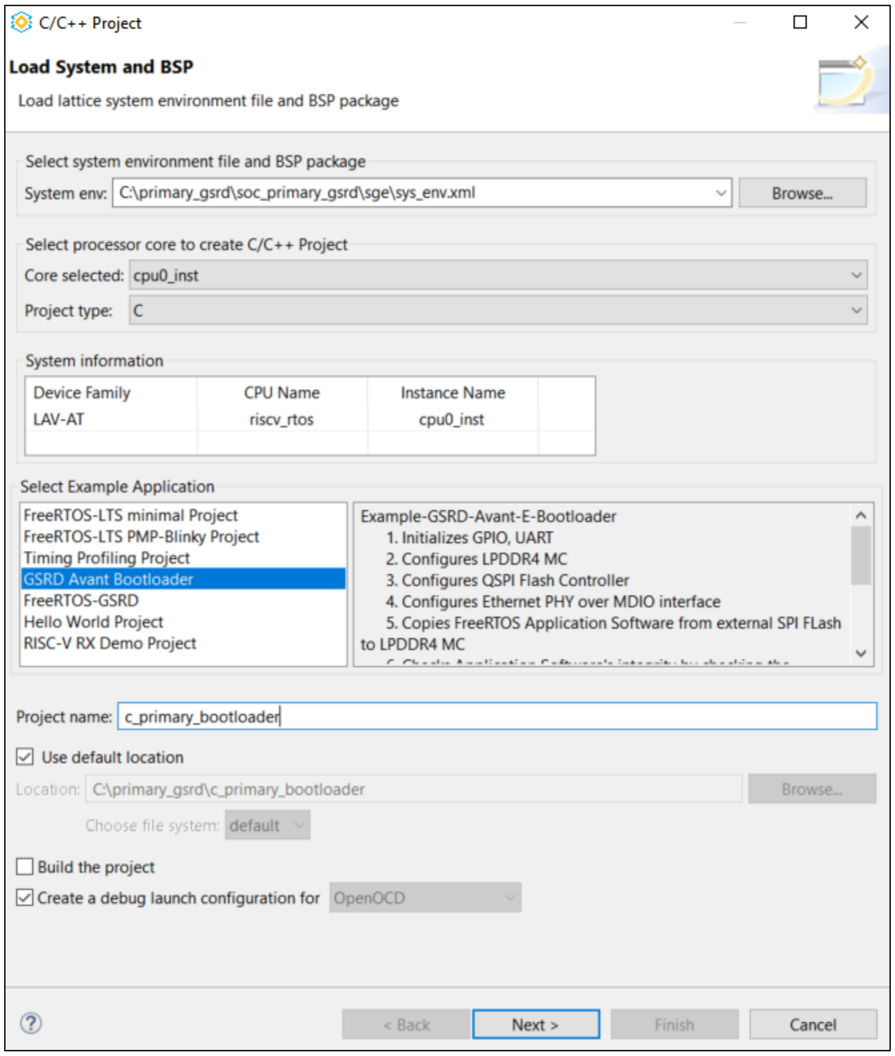


Figure 8.11. Creating Lattice C/C++ Project for Bootloader

13. In C/C++ Project window, it automatically selects the generated sys_env.xml file. From Select Example Application section, Select the GSRD Avant Bootloader, and provide a name for your bootloader as in Figure 8.12.



C/C++ Project

Load System and BSP

Load lattice system environment file and BSP package

Select system environment file and BSP package

System env: C:\primary_gsrd\soc_primary_gsrd\sge\sys_env.xml

Select processor core to create C/C++ Project

Core selected: cpu0_inst

Project type: C

System information

Device Family	CPU Name	Instance Name
LAV-AT	riscv_rtos	cpu0_inst

Select Example Application

- FreeRTOS-LTS minimal Project
- FreeRTOS-LTS PMP-Blinky Project
- Timing Profiling Project
- GSRD Avant Bootloader
- FreeRTOS-GSRD
- Hello World Project
- RISC-V RX Demo Project

Example-GSRD-Avant-E-Bootloader
1. Initializes GPIO, UART
2. Configures LPDDR4 MC
3. Configures QSPI Flash Controller
4. Configures Ethernet PHY over MDIO interface
5. Copies FreeRTOS Application Software from external SPI FLash to LPDDR4 MC

Project name: c_primary_bootloader

☑ Use default location

Location: C:\primary_gsrd\c_primary_bootloader

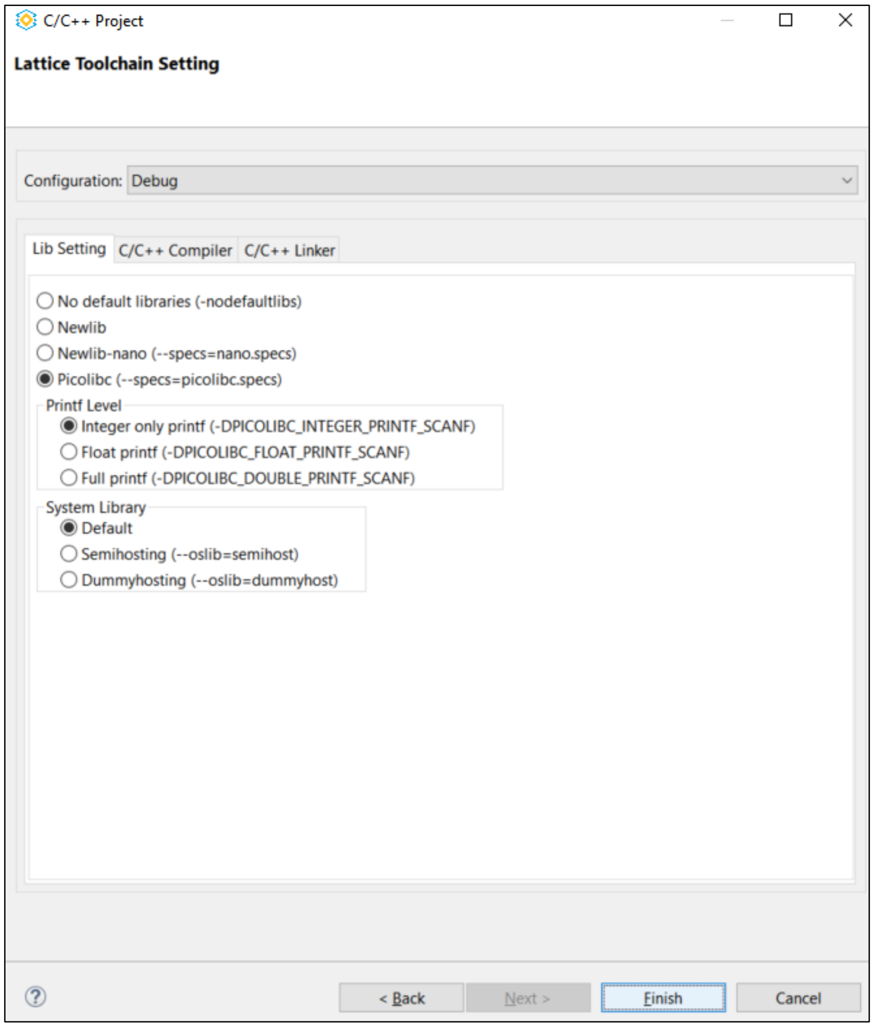
Choose file system: default

☐ Build the project

☑ Create a debug launch configuration for OpenOCD

Figure 8.12. Bootloader C/C++ Selection

14. Click **Next** and **Finish**.


C/C++ Project

Lattice Toolchain Setting

Configuration: Debug

Lib Setting | C/C++ Compiler | C/C++ Linker

( ) No default libraries (-nodefaultlibs)
( ) Newlib
( ) Newlib-nano (--specs=nano.specs)
(●) Picolibc (--specs=picolibc.specs)

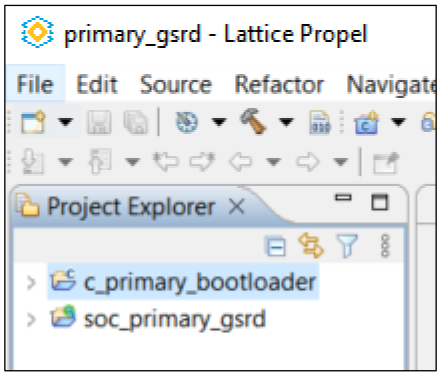
Printf Level
(●) Integer only printf (-DPICOLIBC_INTEGER_PRINTF_SCANF)
( ) Float printf (-DPICOLIBC_FLOAT_PRINTF_SCANF)
( ) Full printf (-DPICOLIBC_DOUBLE_PRINTF_SCANF)

System Library
(●) Default
( ) Semihosting (--oslib=semihost)
( ) Dummyhosting (--oslib=dummyhost)

< Back | Next > | Finish | Cancel

Figure 8.13. C/C++ Lattice Toolchain Setting

15. This loads the Bootloader project in your workspace shown in Figure 8.14.


primary_gsrd - Lattice Propel

File Edit Source Refactor Navigate

Project Explorer

c_primary_bootloader
soc_primary_gsrd

Figure 8.14. Bootloader C Project Created

16. Similarly, create the FreeRTOS C Project. Go to **File > New > Lattice C/C++ Project**.

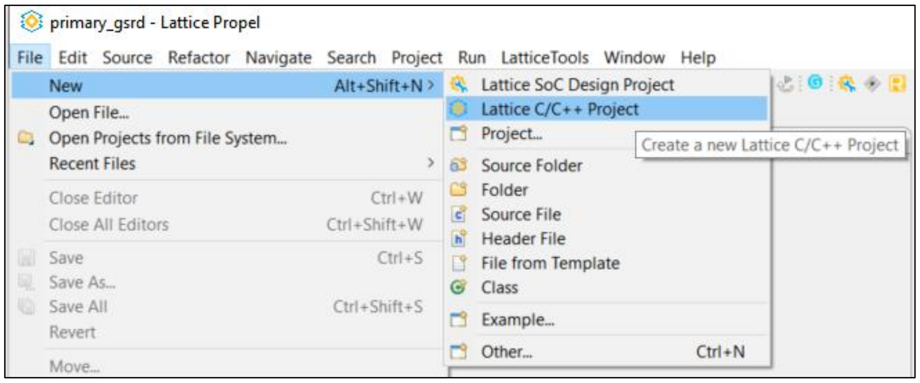
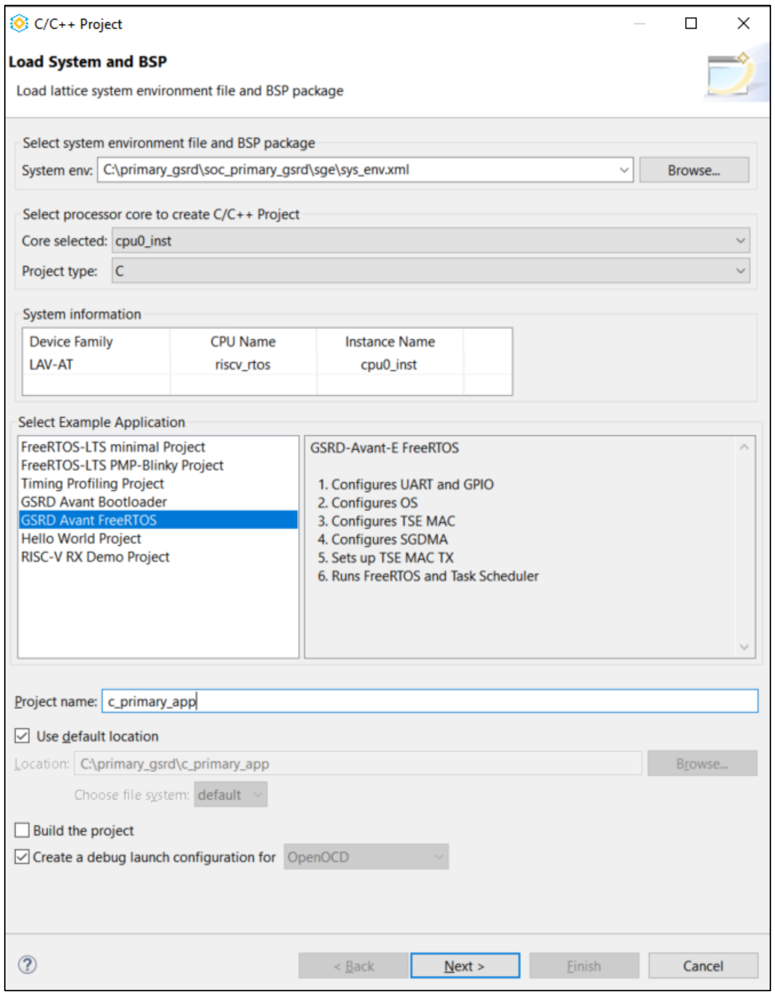


Figure 8.15. Creating C/C++ Project for FreeRTOS

17. From Select Example Application, Select the GSRD Avant FreeRTOS project. Provide the project name as shown in Figure 8.16.


C/C++ Project

Load System and BSP

Load lattice system environment file and BSP package

Select system environment file and BSP package
System env: C:\primary_gsrd\soc_primary_gsrd\sge\sys_env.xml

Select processor core to create C/C++ Project
Core selected: cpu0_inst
Project type: C

System information

Device Family	CPU Name	Instance Name
LAV-AT	riscv_rtos	cpu0_inst

Select Example Application
FreeRTOS-LTS minimal Project
FreeRTOS-LTS PMP-Blinky Project
Timing Profiling Project
GSRD Avant Bootloader
GSRD Avant FreeRTOS
Hello World Project
RISC-V RX Demo Project

GSRD-Avant-E FreeRTOS

1. Configures UART and GPIO
2. Configures OS
3. Configures TSE MAC
4. Configures SGDMA
5. Sets up TSE MAC TX
6. Runs FreeRTOS and Task Scheduler

Project name: c_primary_app
[✓] Use default location
Location: C:\primary_gsrd\c_primary_app
Choose file system: default
[ ] Build the project
[✓] Create a debug launch configuration for OpenOCD

Figure 8.16. FreeRTOS C/C++ Selection

18. Click **Next** and **Finish**.

    Note: This is a manual step you need to perform. After the application project is created, it includes the crc_add_debug.txt,nocrc_add_debug.txt, crc_add_release.txt,nocrc_add_release.txt as shown in Figure 8.17.

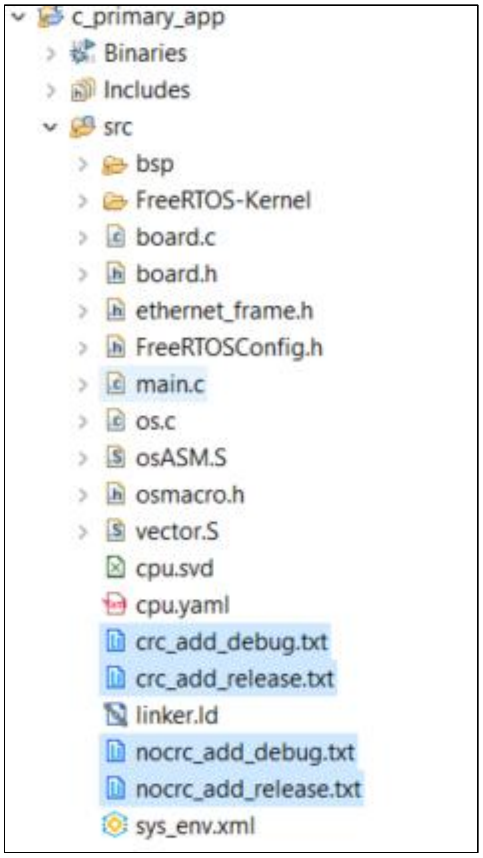


Figure 8.17. FreeRTOS Project Created

19. Copy these four files.

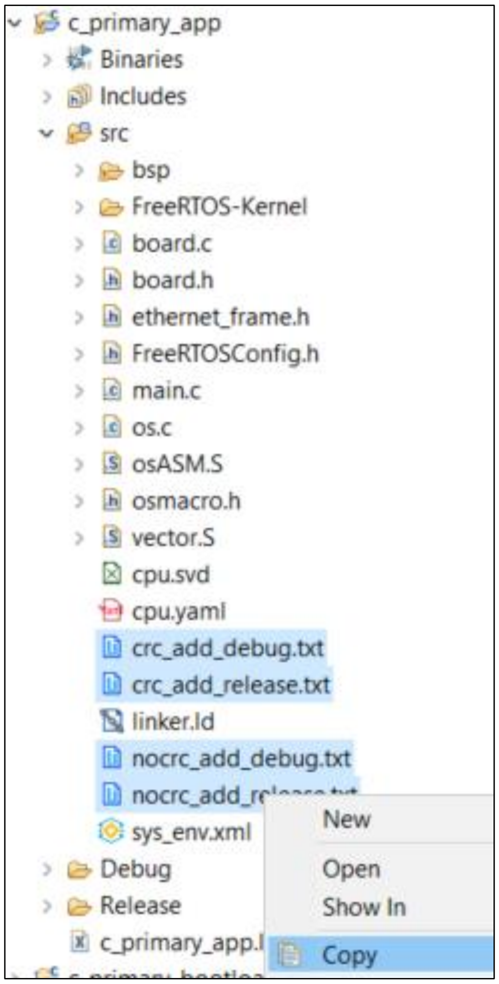


Figure 8.18. Copy the CRC and NoCRC Add files

20. Paste it in your main c_primary_app project as shown in Figure 8.19.

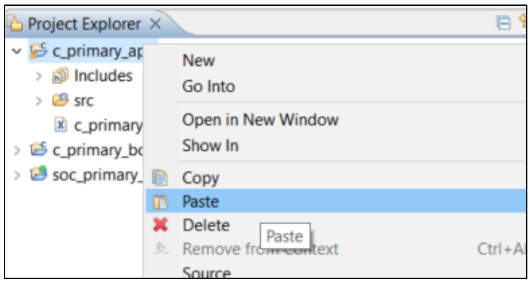
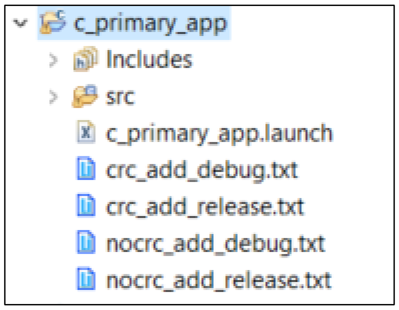


Figure 8.19. Paste in FreeRTOS C project

21. It now adds the txt files under the FreeRTOS App C project as shown in Figure 8.20.

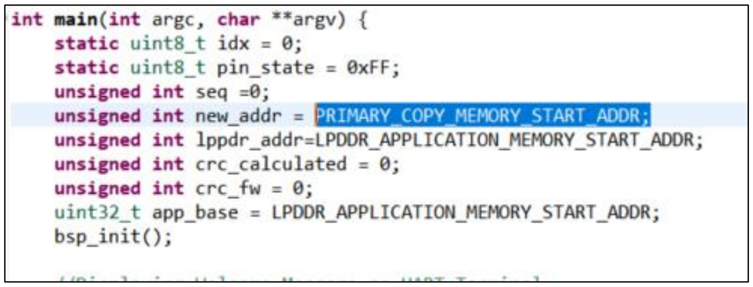


```
c_primary_app
  Includes
  src
  c_primary_app.launch
  crc_add_debug.txt
  crc_add_release.txt
  nocrc_add_debug.txt
  nocrc_add_release.txt
```

Figure 8.20. Copied Text Files

22. In the *c_primary_bootloader main.c* file, edit the *new_addr* from *GOLDEN_COPY_MEMORY_START_ADDR to PRIMARY_COPY_MEMORY_START_ADDR*.

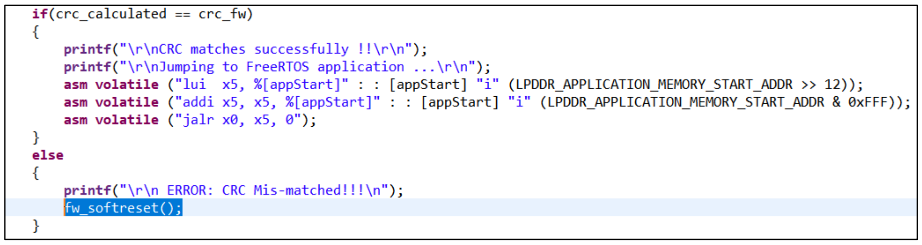
    Note: You must update this new_addr as per your design requirements. If you are just testing the Golden Template, skip this update.



```
int main(int argc, char **argv) {
    static uint8_t idx = 0;
    static uint8_t pin_state = 0xFF;
    unsigned int seq =0;
    unsigned int new_addr = PRIMARY_COPY_MEMORY_START_ADDR;
    unsigned int lppdr_addr=LPDDR_APPLICATION_MEMORY_START_ADDR;
    unsigned int crc_calculated = 0;
    unsigned int crc_fw = 0;
    uint32_t app_base = LPDDR_APPLICATION_MEMORY_START_ADDR;
    bsp_init();
```

Figure 8.21. Updated Primary Address

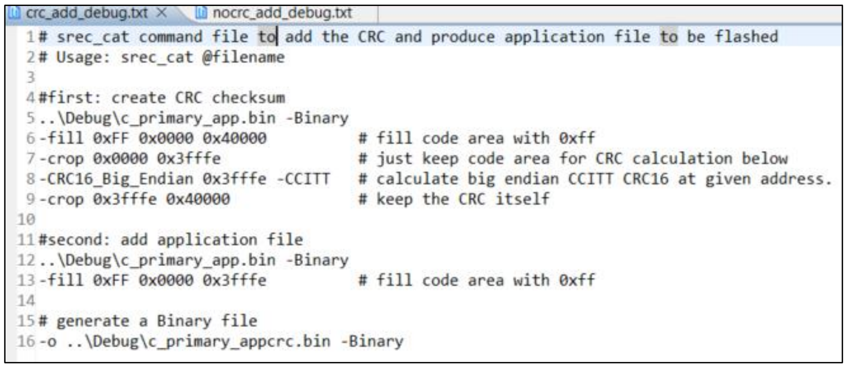
23. For RISC-V to initiate the Soft-Reset for multi-boot, you need to enable this function fw_softreset() as follows. This function is already declared in the bootloader main.c file. This function sends a trigger signal to the Multi-Boot Config Module IP which in-turn switches the Primary GSRD to Golden GSRD to showcase Multi-Boot functionality if you programmed the MCS file as per the demo.



```
if(crc_calculated == crc_fw)
{
    printf("\r\nCRC matches successfully !!\r\n");
    printf("\r\nJumping to FreeRTOS application ...\r\n");
    asm volatile ("lui  x5, %[appStart]" : : [appStart] "i" (LPDDR_APPLICATION_MEMORY_START_ADDR >> 12));
    asm volatile ("addi x5, x5, %[appStart]" : : [appStart] "i" (LPDDR_APPLICATION_MEMORY_START_ADDR & 0xFFF));
    asm volatile ("jalr x0, x5, 0");
}
else
{
    printf("\r\n ERROR: CRC Mis-matched!!!\n");
    fw_softreset();
}
```

Figure 8.22. Enable fw_softreset() Function

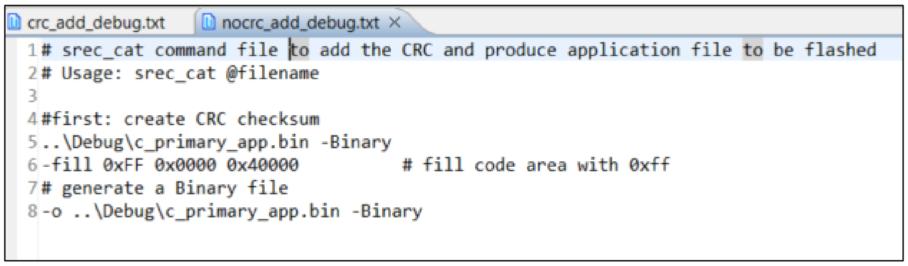
24. In the c_primary_app folder, update the crc_add_debug.txt or crc_add_release.txt file as shown in Figure 8.23. Lines 5 and 12 must be replaced with the app project name that you provided; in this case the name is c_primary_app. Hence, the name of the file is replaced with c_primary_app.bin. Line 16 must contain the name as c_primary_appcrc.bin. This output file has the CRC for application software appended at the end of binary file.



```
 1# srec_cat command file to add the CRC and produce application file to be flashed
 2# Usage: srec_cat @filename
 3
 4#first: create CRC checksum
 5..\Debug\c_primary_app.bin -Binary
 6-fill 0xFF 0x0000 0x40000           # fill code area with 0xff
 7-crop 0x0000 0x3fffe                # just keep code area for CRC calculation below
 8-CRC16_Big_Endian 0x3fffe -CCITT    # calculate big endian CCITT CRC16 at given address.
 9-crop 0x3fffe 0x40000               # keep the CRC itself
10
11#second: add application file
12..\Debug\c_primary_app.bin -Binary
13-fill 0xFF 0x0000 0x3fffe           # fill code area with 0xff
14
15# generate a Binary file
16-o ..\Debug\c_primary_appcrc.bin -Binary
```

Figure 8.23. Update crc_add_debug.txt

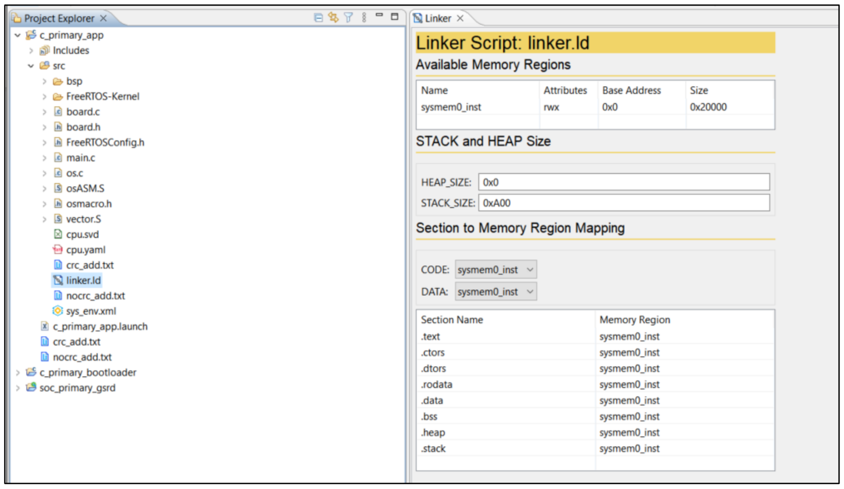
25. Similarly, in the noncrc_add_debug.txt or nocrc_add_release.txt, update the Lines 5 and 8 with the file name without CRC as shown in Figure 8.24.



```
1# srec_cat command file to add the CRC and produce application file to be flashed
2# Usage: srec_cat @filename
3
4#first: create CRC checksum
5..\Debug\c_primary_app.bin -Binary
6-fill 0xFF 0x0000 0x40000           # fill code area with 0xff
7# generate a Binary file
8-o ..\Debug\c_primary_app.bin -Binary
```

Figure 8.24. Update noncrc_add.txt

26. Open the Linker.ld File from c_primary_app.



Linker Script: linker.ld

Available Memory Regions

Name	Attributes	Base Address	Size
sysmem0_inst	rwx	0x0	0x20000

STACK and HEAP Size

HEAP_SIZE: 0x0
STACK_SIZE: 0xA00

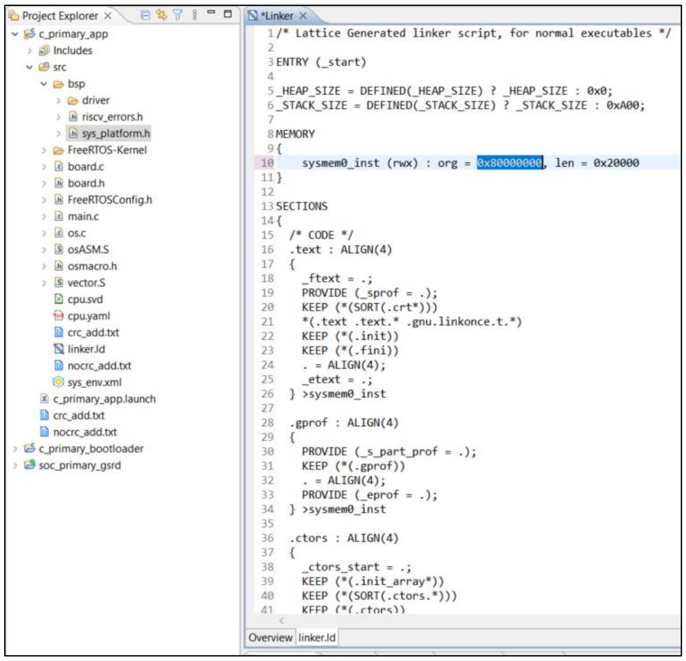
Section to Memory Region Mapping

CODE: sysmem0_inst
DATA: sysmem0_inst

Section Name	Memory Region
.text	sysmem0_inst
.ctors	sysmem0_inst
.dtors	sysmem0_inst
.rodata	sysmem0_inst
.data	sysmem0_inst
.bss	sysmem0_inst
.heap	sysmem0_inst
.stack	sysmem0_inst

Figure 8.25. Open Linker.Ld File

27. Update the MEMORY org address to 0x80000000 for FreeRTOS to run from LPDDR4.



```
/* Lattice Generated linker script, for normal executables */

ENTRY (_start)

_HEAP_SIZE = DEFINED(_HEAP_SIZE) ? _HEAP_SIZE : 0x0;
_STACK_SIZE = DEFINED(_STACK_SIZE) ? _STACK_SIZE : 0xA00;

MEMORY
{
    sysmem0_inst (rwx) : org = 0x80000000, len = 0x20000
}

SECTIONS
{
  /* CODE */
  .text : ALIGN(4)
  {
    _ftext = .;
    PROVIDE (_sprof = .);
    KEEP (*(SORT(.crt*)))
    *(.text .text.* .gnu.linkonce.t.*)
    KEEP (*(.init))
    KEEP (*(.fini))
    . = ALIGN(4);
    _etext = .;
  } >sysmem0_inst

  .gprof : ALIGN(4)
  {
    PROVIDE (_s_part_prof = .);
    KEEP (*(.gprof))
    . = ALIGN(4);
    PROVIDE (_eprof = .);
  } >sysmem0_inst

  .ctors : ALIGN(4)
  {
    _ctors_start = .;
    KEEP (*(.init_array*))
    KEEP (*(SORT(.ctors.*)))
    KEEP (*(.ctors))
```

Figure 8.26. Update Linker.ld File

28. Press **Ctrl + s** to save the change as shown in Figure 8.29.

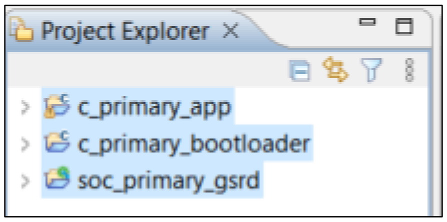


Figure 8.27. Workspace

29. To create the bootloader binary file, right-click on c_primary_bootloader and select **Build Project**.

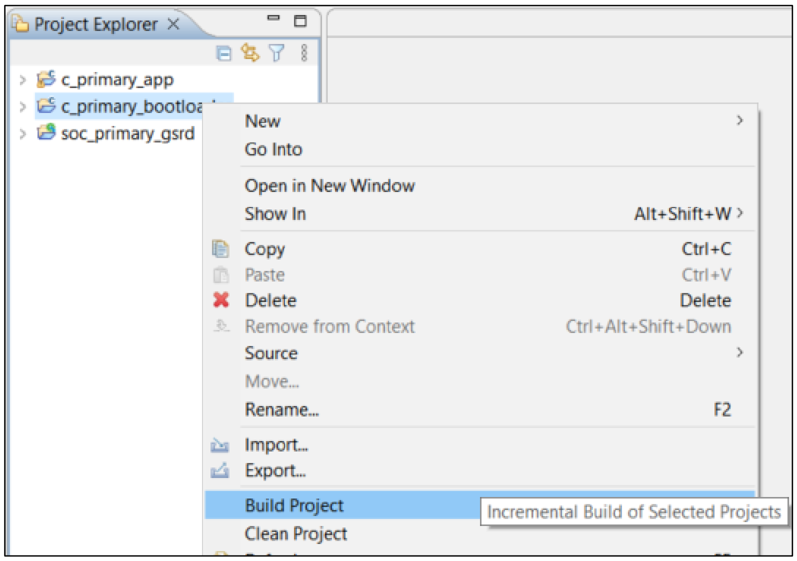
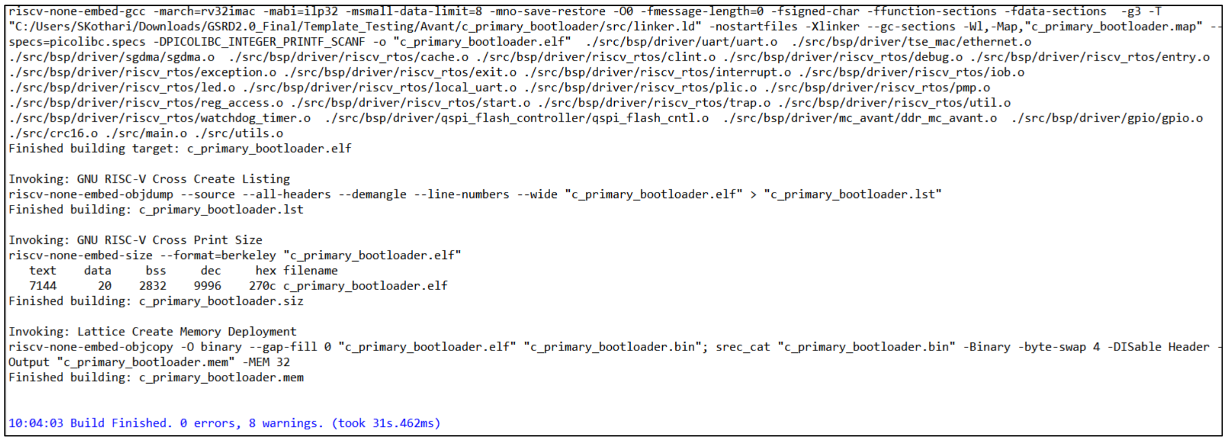


Figure 8.28. Build Bootloader Project

30. The console output is displayed as shown in Figure 8.31.



```
riscv-none-embed-gcc -march=rv32imac -mabi=ilp32 -msmall-data-limit=8 -mno-save-restore -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -T
"C:/Users/SKothari/Downloads/GSRD2.0_Final/Template_Testing/Avant/c_primary_bootloader/src/linker.ld" -nostartfiles -Xlinker --gc-sections -Wl,-Map,"c_primary_bootloader.map" --
specs=picolibc.specs -DPICOLIBC_INTEGER_PRINTF_SCANF -o "c_primary_bootloader.elf"  ./src/bsp/driver/uart/uart.o  ./src/bsp/driver/tse_mac/ethernet.o
./src/bsp/driver/sgdma/sgdma.o  ./src/bsp/driver/riscv_rtos/cache.o ./src/bsp/driver/riscv_rtos/clint.o ./src/bsp/driver/riscv_rtos/debug.o ./src/bsp/driver/riscv_rtos/entry.o
./src/bsp/driver/riscv_rtos/exception.o ./src/bsp/driver/riscv_rtos/exit.o ./src/bsp/driver/riscv_rtos/interrupt.o ./src/bsp/driver/riscv_rtos/iob.o
./src/bsp/driver/riscv_rtos/led.o ./src/bsp/driver/riscv_rtos/local_uart.o ./src/bsp/driver/riscv_rtos/plic.o ./src/bsp/driver/riscv_rtos/pmp.o
./src/bsp/driver/riscv_rtos/reg_access.o ./src/bsp/driver/riscv_rtos/start.o ./src/bsp/driver/riscv_rtos/trap.o ./src/bsp/driver/riscv_rtos/util.o
./src/bsp/driver/riscv_rtos/watchdog_timer.o  ./src/bsp/driver/qspi_flash_controller/qspi_flash_cntl.o  ./src/bsp/driver/mc_avant/ddr_mc_avant.o  ./src/bsp/driver/gpio/gpio.o
./src/crc16.o ./src/main.o ./src/utils.o
Finished building target: c_primary_bootloader.elf

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

Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "c_primary_bootloader.elf"
   text    data     bss     dec     hex filename
   7144      20    2832    9996    270c c_primary_bootloader.elf
Finished building: c_primary_bootloader.siz

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


10:04:03 Build Finished. 0 errors, 8 warnings. (took 31s.462ms)
```

Figure 8.29. Bootloader Build Project Console Output

31. This creates a debug folder as shown in Figure 8.30. The binary created is named as c_primary_bootloader.mem. This goes as a part of system memory in Propel Builder design.

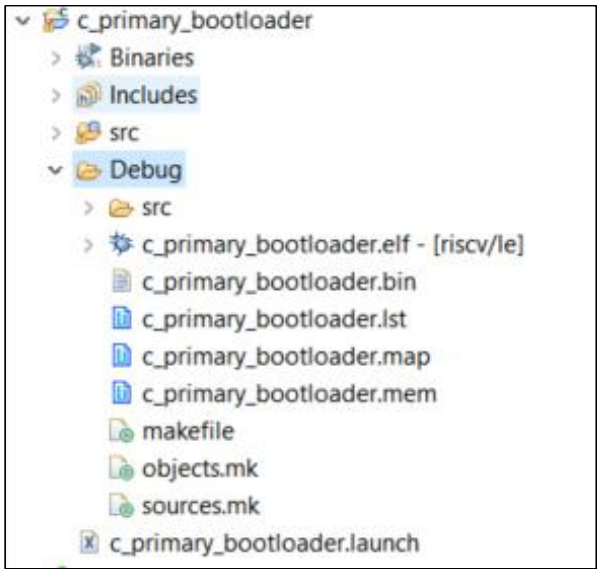


Figure 8.30. Bootloader Binary Created

32. Now, before creating the Binary for FreeRTOS project, Right-Click on c_primary_app project and click on **Properties**.

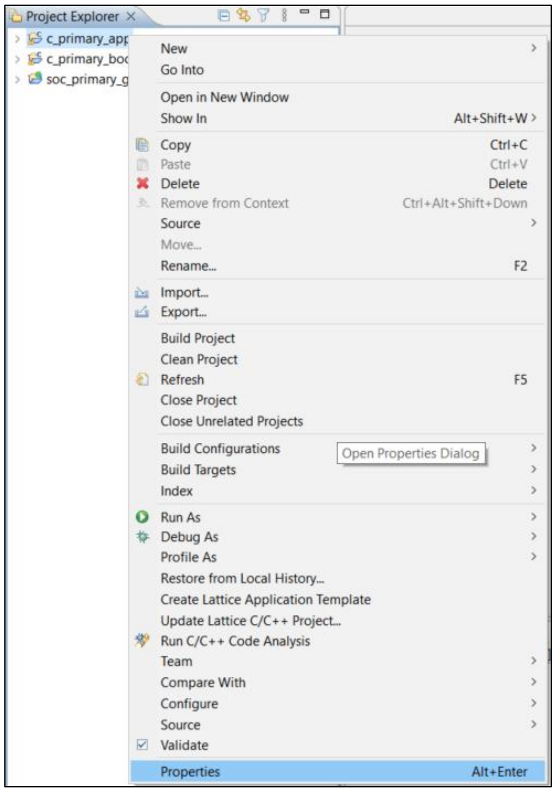


Figure 8.31. Properties

33. In case you wish to create a Release folder on your C project build, go to **C/C++ Build > Settings and Under Configuration tab**, select Release. Otherwise, skip to Step 47.

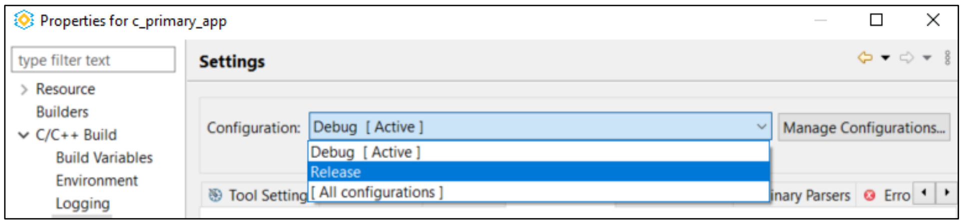


Figure 8.32. Select Release as Configuration

34. Click on **Manage Configurations**, select the **Release**, and click on **Set Active**.

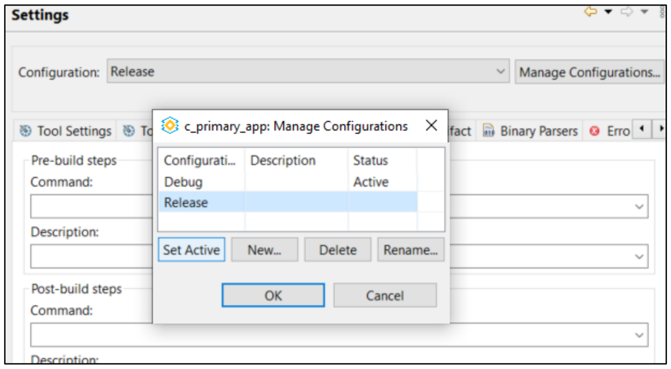
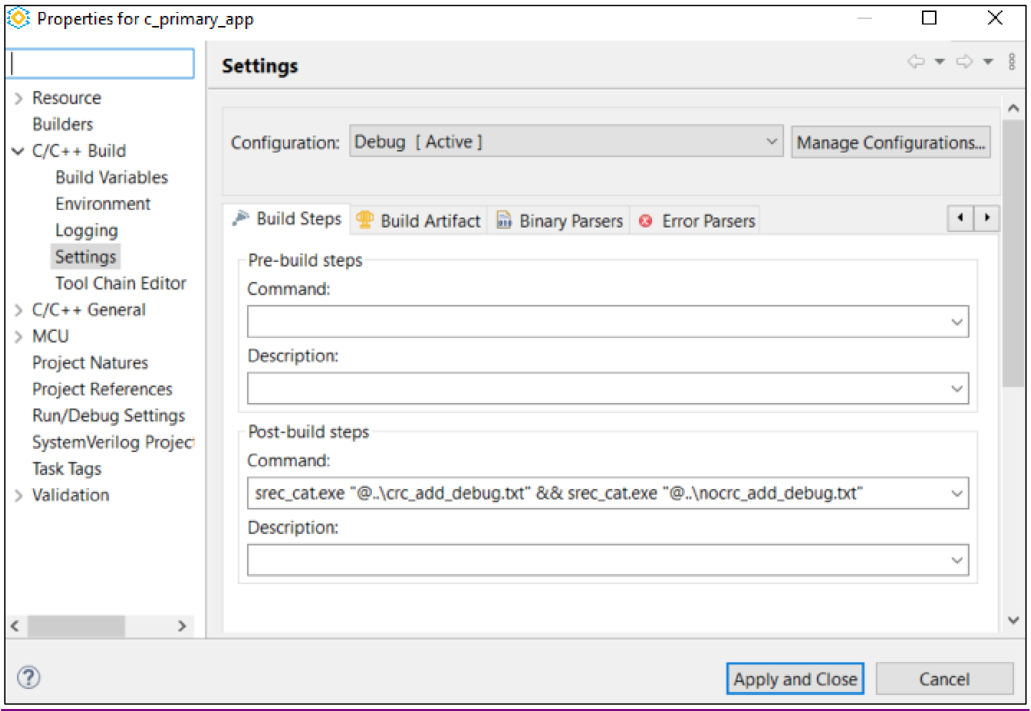


Figure 8.33. Set Release as Active Configuration

35. Go to **C/C++ Build > Settings > Build Steps** and under **Post-Build Steps > Command**, add these commands as shown below for your respective builds i.e. for Debug or Release.

```
srec_cat.exe "@..\crc_add_debug.txt" && srec_cat.exe "@..\nocrc_add_debug.txt"

srec_cat.exe "@..\crc_add_release.txt" && srec_cat.exe "@..\nocrc_add_release.txt"
```



Configuration: Debug [Active]

Pre-build steps
Command:
Description:

Post-build steps
Command: srec_cat.exe "@..\crc_add_debug.txt" && srec_cat.exe "@..\nocrc_add_debug.txt"
Description:

Figure 8.34. Adding Post Build Step for FreeRTOS Application CRC Binary Append

36. Click **Apply** and **Close**.
37. Right-click on **c_primary_app** and click on **Build Project**.

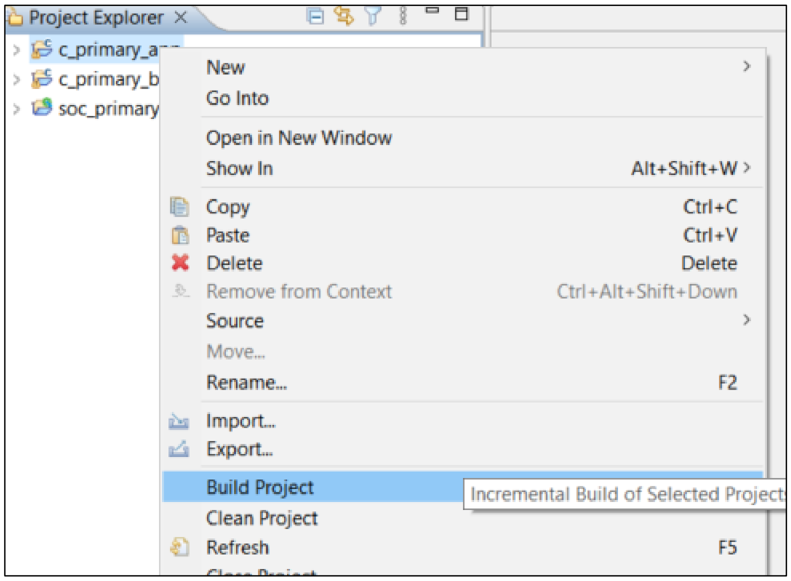
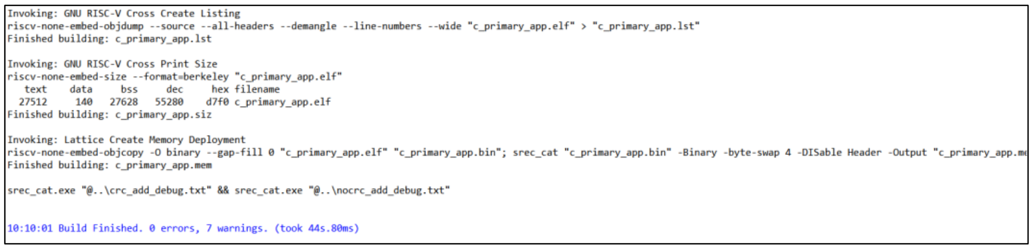


Figure 8.35. Build c_primary_app C/C++ Project

38. The console output is displayed as shown in Figure 8.36.



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

Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "c_primary_app.elf"
   text    data     bss     dec     hex filename
  27512     140   27628   55280    d7f0 c_primary_app.elf
Finished building: c_primary_app.siz

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

srec_cat.exe "@..\crc_add_debug.txt" && srec_cat.exe "@..\nocrc_add_debug.txt"


10:10:01 Build Finished. 0 errors, 7 warnings. (took 44s.80ms)
```

Figure 8.36. FreeRTOS App Build Project Console Output

39. This creates the debug folder and the following files with **c_primary_app.bin** and **c_primary_appcrc.bin** binaries with CRC and without CRC as shown in Figure 8.37.

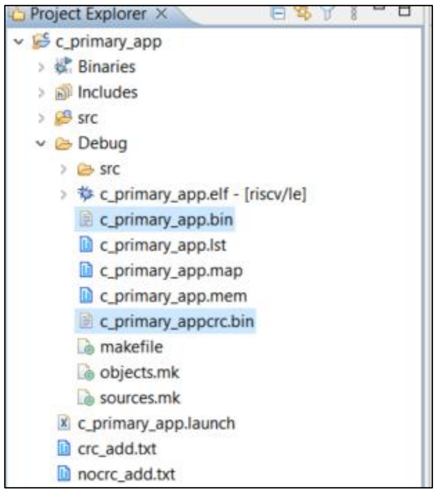


Figure 8.37. FreeRTOS App Binaries Created with CRC and without CRC

8.2. Validating and Generating the GSRD Design using Propel Builder

To validate and generate the GSRD design, perform the following steps:

1. Select the soc project and Launch Propel Builder from Propel SDK as shown in Figure 8.38, if it is not already opened.

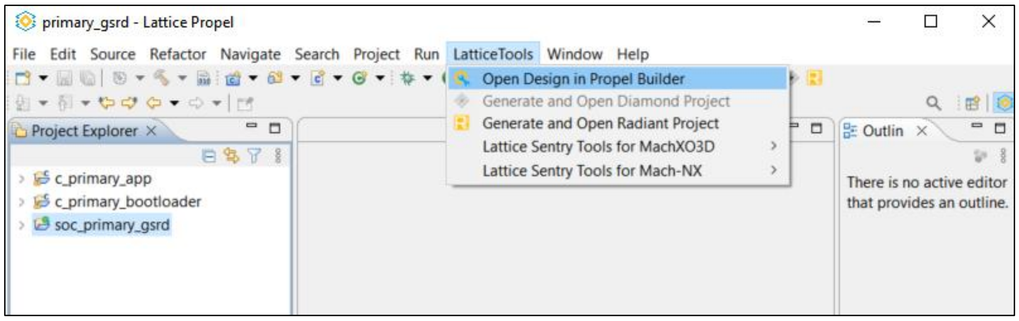

primary_gsrd - Lattice Propel
File Edit Source Refactor Navigate Search Project Run LatticeTools Window Help
Open Design in Propel Builder
Generate and Open Diamond Project
Generate and Open Radiant Project
Lattice Sentry Tools for MachXO3D
Lattice Sentry Tools for Mach-NX
Project Explorer
c_primary_app
c_primary_bootloader
soc_primary_gsrd
Outlin
There is no active editor that provides an outline.

Figure 8.38. Open Propel Builder

2. The Primary GHRD SoC opens as shown in Figure 8.39.

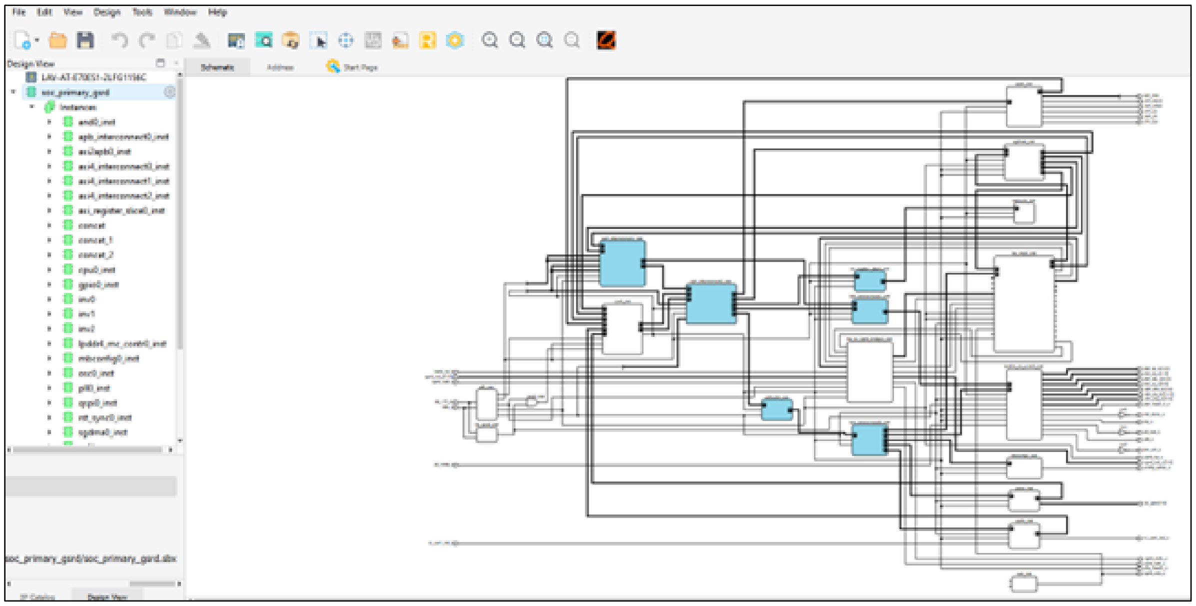


Figure 8.39. Propel Builder SoC Project

3. Before validating and generating the system, you need to pre-initialize the system memory.
4. Select the **System Memory** Instance from the Design View in the left column and it highlights the *sysmem0_inst* as shown in Figure 8.40.

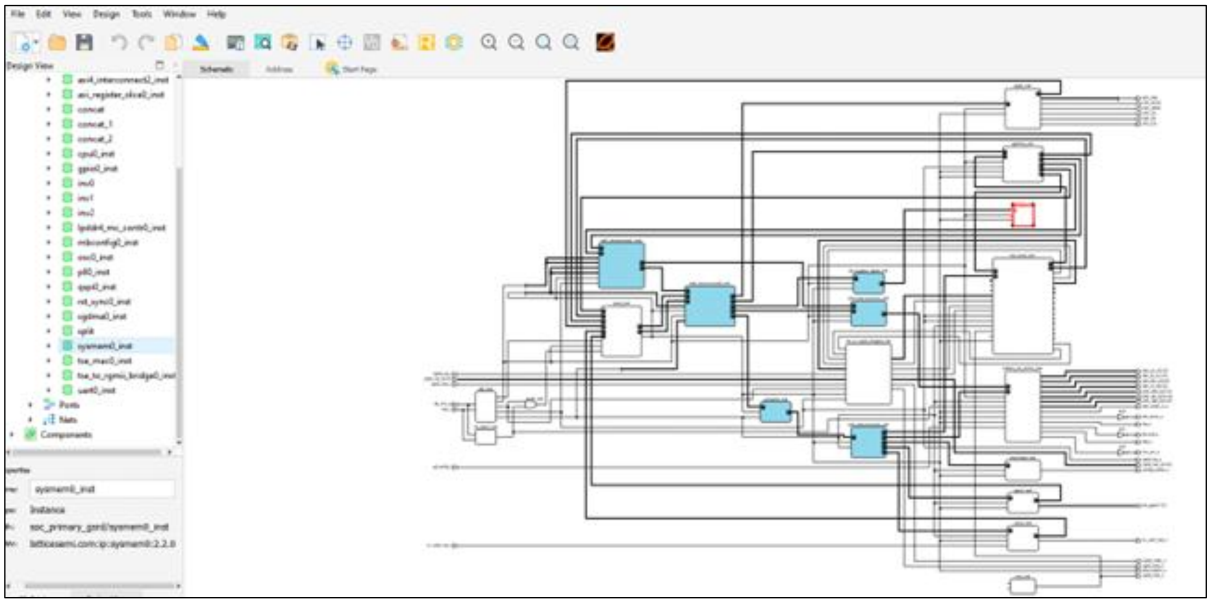
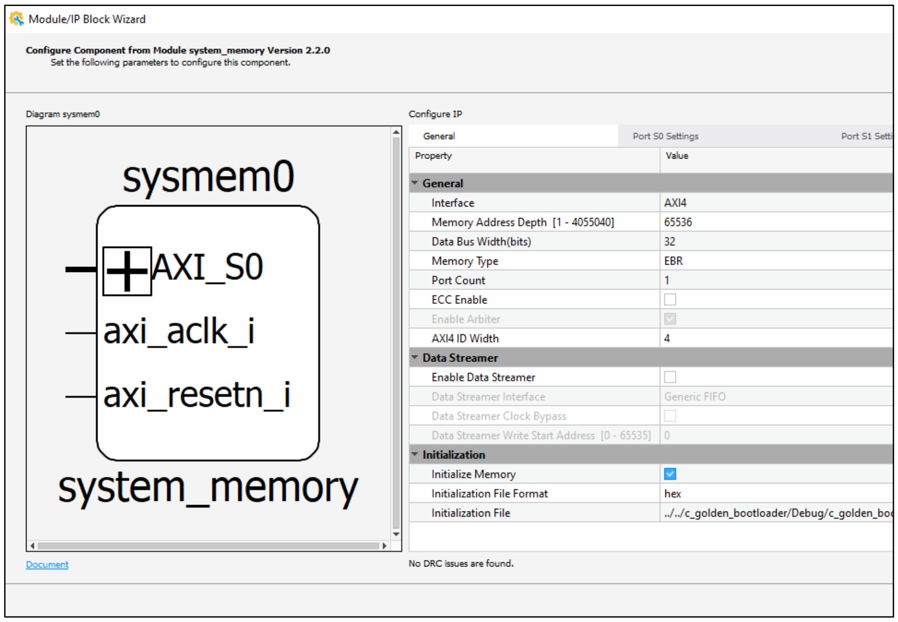


Figure 8.40. System Memory Highlighted

5. Double-click on the highlighted component shown in the schematic and it opens the Module/IP Block Wizard as shown in Figure 8.41.



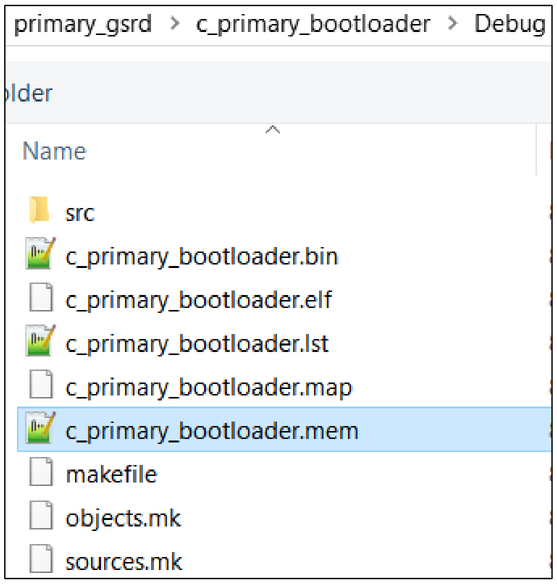
Configure Component from Module system_memory Version 2.2.0
Set the following parameters to configure this component.

Property	Value
General	
Interface	AXI4
Memory Address Depth [1 - 4055040]	65536
Data Bus Width(bits)	32
Memory Type	EBR
Port Count	1
ECC Enable	☐
Enable Arbiter	✓
AXI4 ID Width	4
Data Streamer	
Enable Data Streamer	☐
Data Streamer Interface	Generic FIFO
Data Streamer Clock Bypass	☐
Data Streamer Write Start Address [0 - 65535]	0
Initialization	
Initialize Memory	✓
Initialization File Format	hex
Initialization File	../../c_golden_bootloader/Debug/c_golden_bo

No DRC issues are found.

Figure 8.41. System Memory IP Configuration

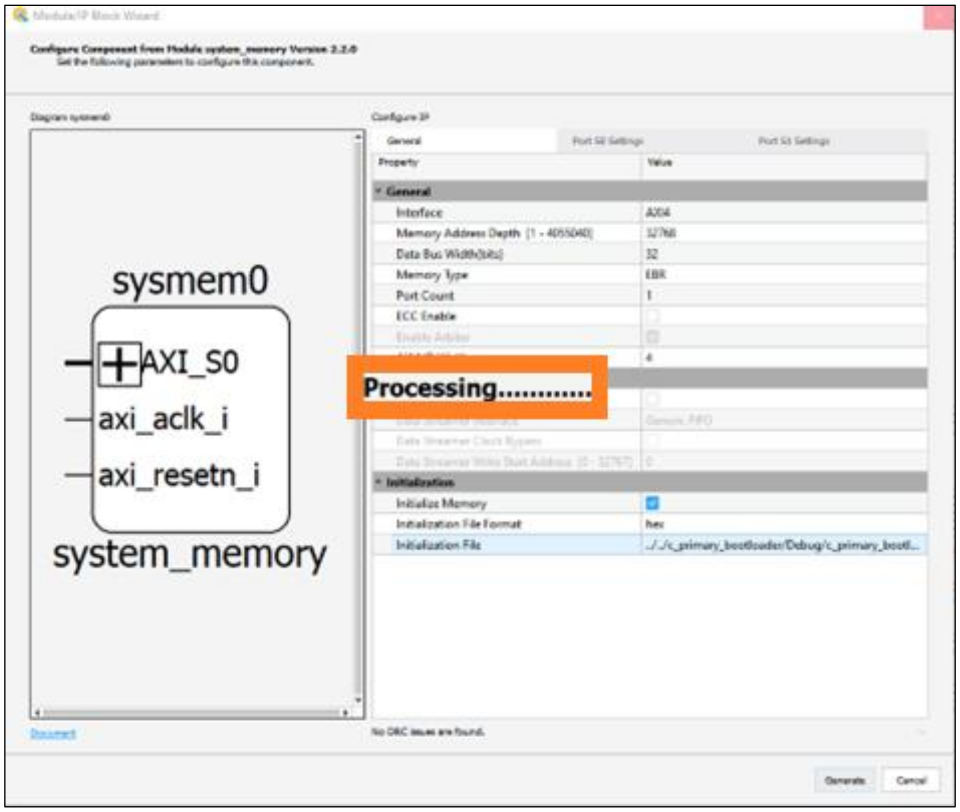
6. Click on **Initialization File's Value** area with three dots to locate the generated **c_primary_bootloader.mem** Bootloader file as shown in Figure 8.42.



src
c_primary_bootloader.bin
c_primary_bootloader.elf
c_primary_bootloader.lst
c_primary_bootloader.map
c_primary_bootloader.mem
makefile
objects.mk
sources.mk

Figure 8.42. Initialize System Memory with c_primary_bootloader.mem

7. Select the **c_primary_bootloader.mem** file created above.
8. Click **Generate** on the bottom-right corner of the IP Block Wizard as shown in Figure 8.43.

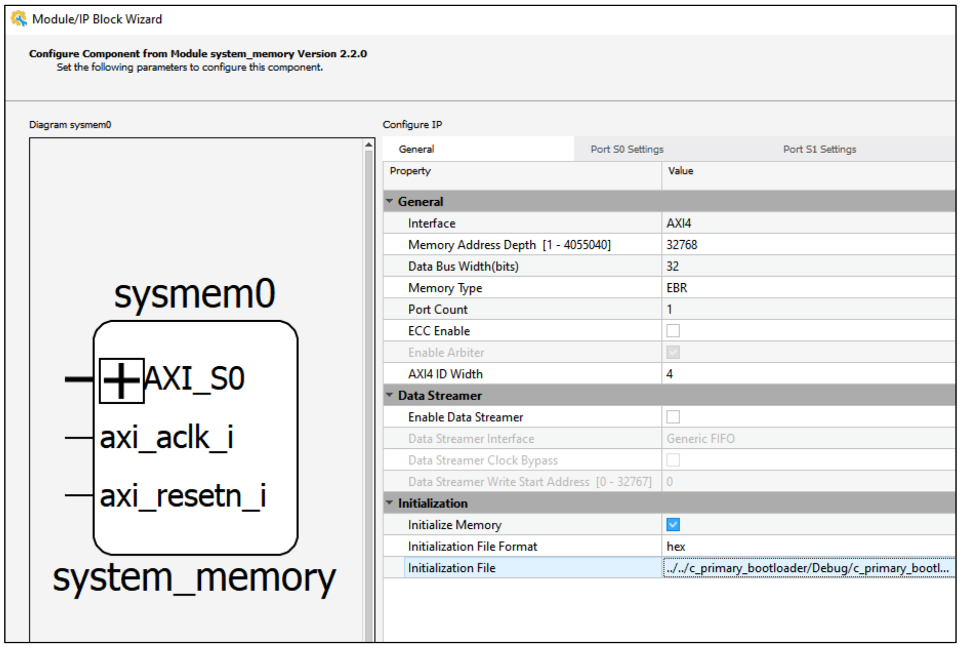


Configure Component from Module system_memory Version 2.2.0
Set the following parameters to configure this component.

Property	Value
General	
Interface	AXI4
Memory Address Depth [1 - 4055040]	32768
Data Bus Width(bits)	32
Memory Type	EBR
Port Count	1
ECC Enable	☐
Enable Arbiter	☐
AXI4 ID Width	4
Enable Data Streamer	☐
Data Streamer Interface	Generic FIFO
Data Streamer Clock Bypass	☐
Data Streamer Write Start Address [0 - 32767]	0
Initialization	
Initialize Memory	✓
Initialization File Format	hex
Initialization File	../../c_primary_bootloader/Debug/c_primary_bootl...

No DRC issues are found.

Figure 8.43. Click Generate

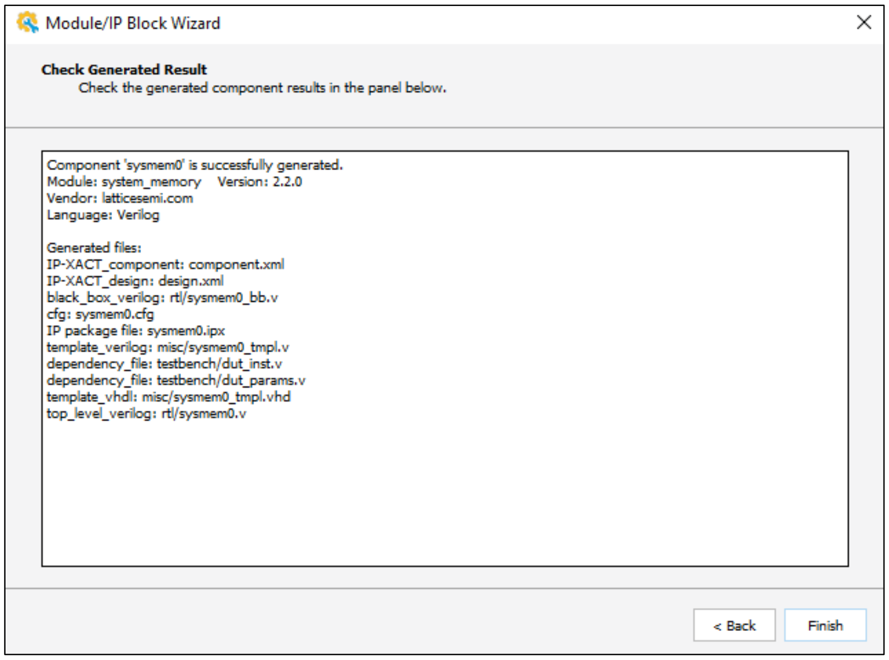


Configure Component from Module system_memory Version 2.2.0
Set the following parameters to configure this component.

Property	Value
General	
Interface	AXI4
Memory Address Depth [1 - 4055040]	32768
Data Bus Width(bits)	32
Memory Type	EBR
Port Count	1
ECC Enable	☐
Enable Arbiter	☐
AXI4 ID Width	4
Data Streamer	
Enable Data Streamer	☐
Data Streamer Interface	Generic FIFO
Data Streamer Clock Bypass	☐
Data Streamer Write Start Address [0 - 32767]	0
Initialization	
Initialize Memory	✓
Initialization File Format	hex
Initialization File	../../c_primary_bootloader/Debug/c_primary_bootl...

Figure 8.44.Bootloader file updated in System Memory

9. Wait until the processing is complete and click **Finish** as shown in Figure 8.45.



```
Component 'sysmem0' is successfully generated.
Module: system_memory    Version: 2.2.0
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/sysmem0_bb.v
cfg: sysmem0.cfg
IP package file: sysmem0.ipx
template_verilog: misc/sysmem0_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
template_vhdl: misc/sysmem0_tmpl.vhd
top_level_verilog: rtl/sysmem0.v
```

Figure 8.45. Click Finish

10. Go to **Design > Validate** and **Design and Design > Generate Icon**. . Output shown is similar to Figure 8.7 and Figure 8.9 Make sure there are no DRC errors.
11. Make sure there are no errors. Note that the *INFO <235999*>* messages on the TCL console are expected since these modules or IPs do not contain any software IP drivers.

8.3. Synthesizing the RTL Files and Generating the Bitstream using Lattice Radiant

To synthesize the RTL files and generate the bitstream, perform the following steps:

1. Click the Radiant icon from Propel Builder to launch the Radiant software as shown in Figure 8.46.

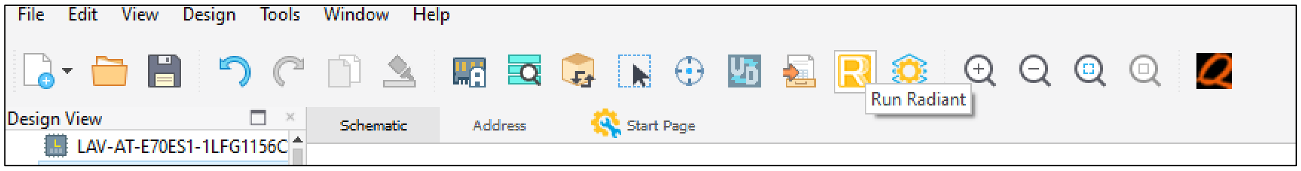
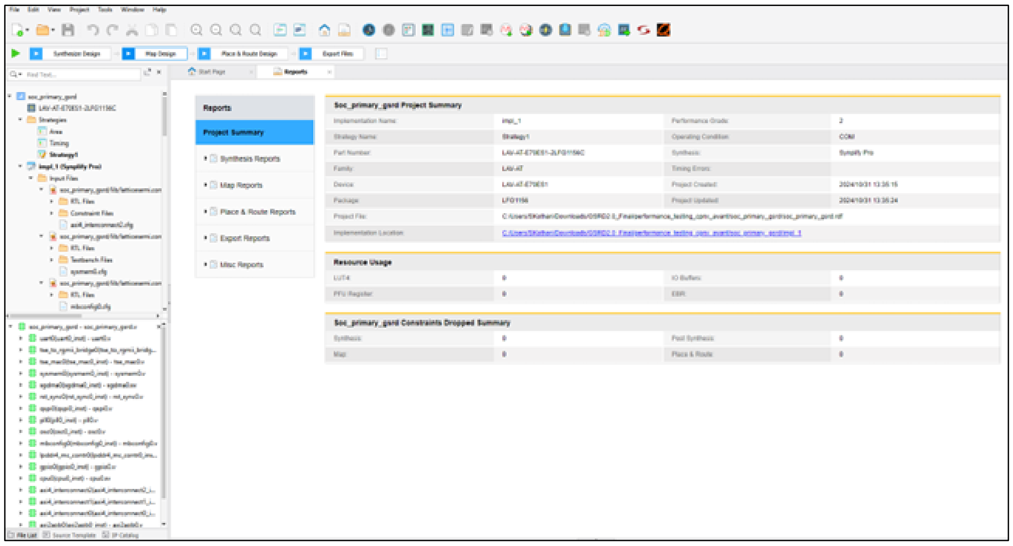


Figure 8.46. Run Radiant Icon

2. The Radiant window of your project is launched as shown in Figure 8.47.



Soc_primary_gsrd Project Summary

Implementation Name:	impl_1	Performance Grade:	2
Strategy Name:	Strategy1	Operating Condition:	COM
Part Number:	LAV-AT-E70ES1-2LFG1156C	Synthesis:	Synplify Pro
Family:	LAV-AT	Timing Errors:	
Device:	LAV-AT-E70ES1	Project Created:	2024/10/31 13:35:15
Package:	LFG1156	Project Updated:	2024/10/31 13:35:24
Project File:	C:/Users/SKathari/Downloads/GSRD2.0_Final/performance_testing_cpm_avant/soc_primary_gsrd/soc_primary_gsrd.rdf		
Implementation Location:	C:/Users/SKathari/Downloads/GSRD2.0_Final/performance_testing_cpm_avant/soc_primary_gsrd/impl_1		

Resource Usage

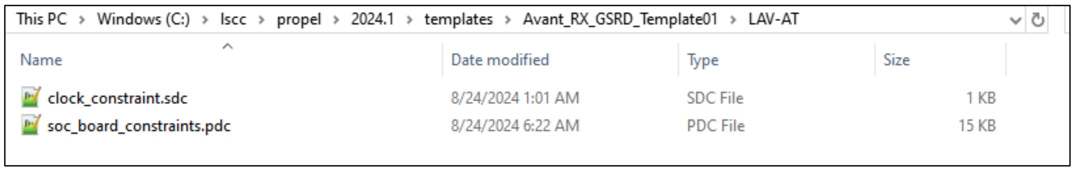
LUT4:	0	IO Buffers:	0
PFU Register:	0	EBR:	0

Soc_primary_gsrd Constraints Dropped Summary

Synthesis:	0	Post Synthesis:	0
Map:	0	Place & Route:	0

Figure 8.47. Lattice Radiant 2024.1.1 Window

3. Click on **Close** on the pop-up about Updates.
4. Locate the following clock_constraint.sdc and soc_board_constraint.pdc files in your Propel Installation area.



This PC > Windows (C:) > lscc > propel > 2024.1 > templates > Avant_RX_GSRD_Template01 > LAV-AT

Name	Date modified	Type	Size
clock_constraint.sdc	8/24/2024 1:01 AM	SDC File	1 KB
soc_board_constraints.pdc	8/24/2024 6:22 AM	PDC File	15 KB

Figure 8.48. Constraint Files Folder

5. Right-click on **Pre-Synthesis Constraint Files > Add > Existing File** as shown in Figure 8.49.

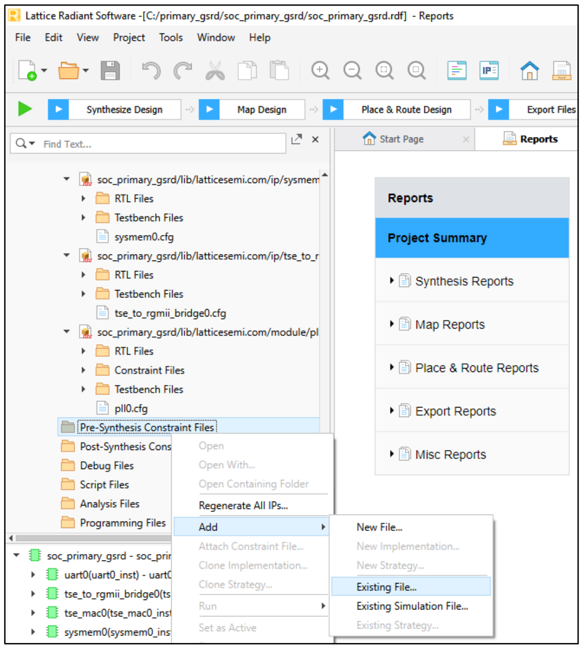
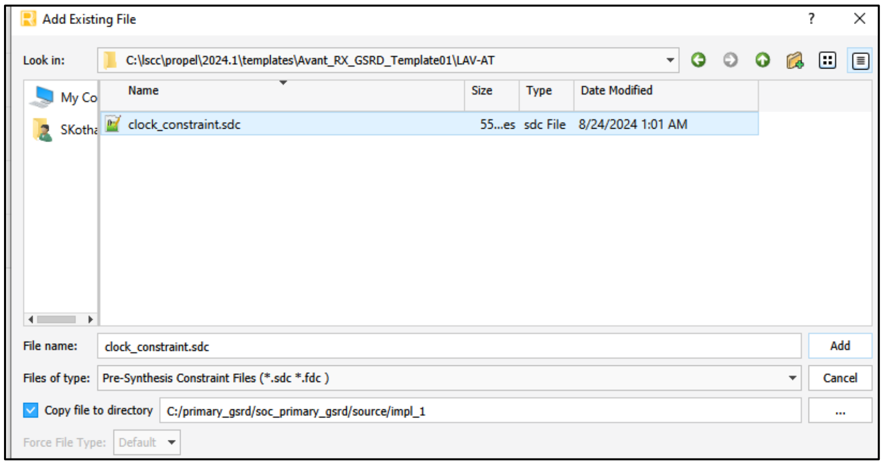


Figure 8.49. Add Existing Pre-Synthesis Constraint

6. Select the clock_constraint.sdc and ensure that the *Copy File to Directory* is enabled as shown in Figure 8.50. Click **Add**.


Add Existing File

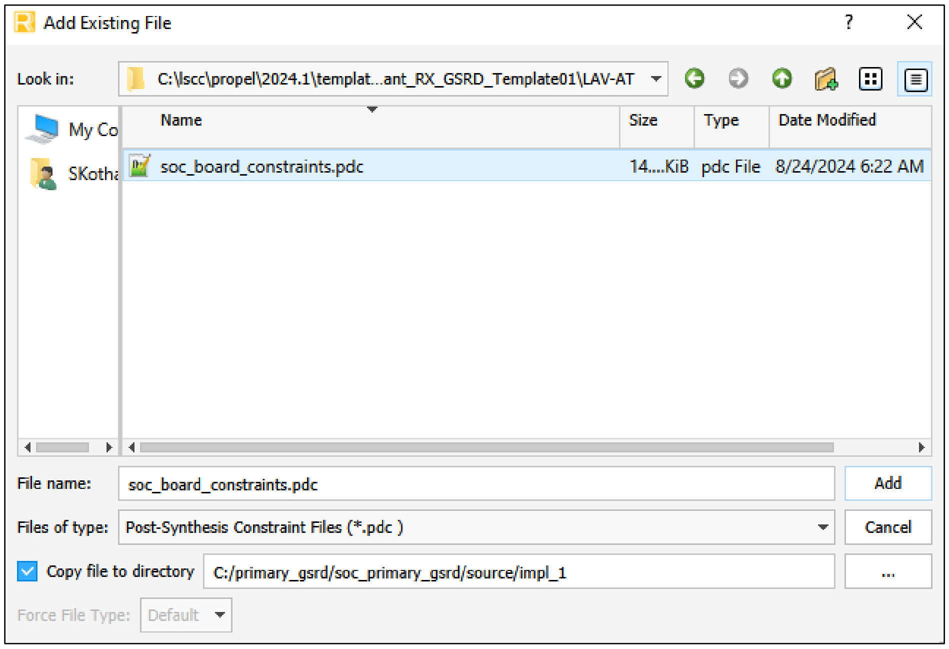
Look in: C:\lscc\propel\2024.1\templates\Avant_RX_GSRD_Template01\LAV-AT

Name	Size	Type	Date Modified
clock_constraint.sdc	55...es	sdc File	8/24/2024 1:01 AM

File name: clock_constraint.sdc
Files of type: Pre-Synthesis Constraint Files (*.sdc *.fdc)
Copy file to directory: C:/primary_gsrd/soc_primary_gsrd/source/impl_1
Force File Type: Default

Figure 8.50. Select clock_constraint.sdc File

7. Similarly, add the soc_board_constraints.pdc file to the Post-Synthesis Constraint Files as shown in Figure 8.51. Click **Add**.


Add Existing File

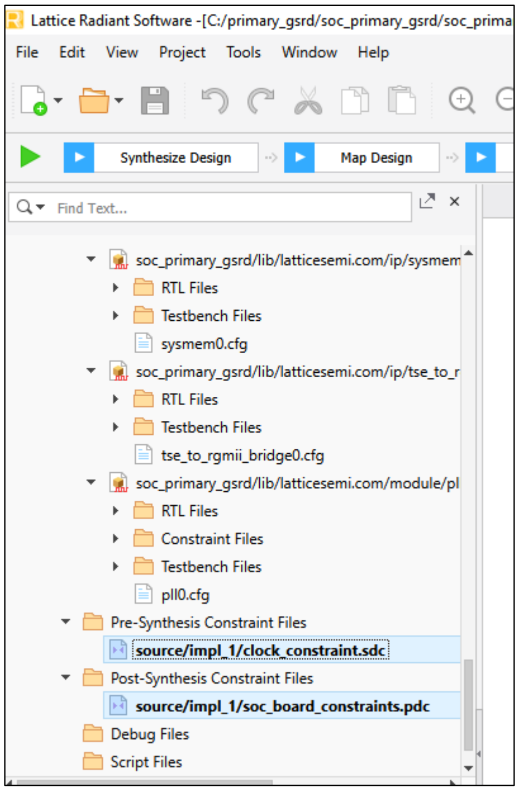
Look in: C:\lscc\propel\2024.1\templat...ant_RX_GSRD_Template01\LAV-AT

Name	Size	Type	Date Modified
soc_board_constraints.pdc	14....KiB	pdc File	8/24/2024 6:22 AM

File name: soc_board_constraints.pdc
Files of type: Post-Synthesis Constraint Files (*.pdc)
Copy file to directory: C:/primary_gsrd/soc_primary_gsrd/source/impl_1
Force File Type: Default

Figure 8.51. Select soc_board_constraint.pdc File

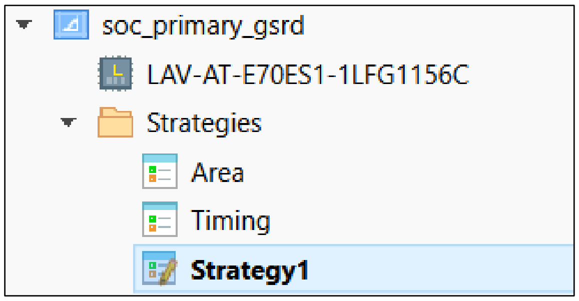
8. After you add these constraint files, the Radiant file list is updated as shown in Figure 8.52.


Lattice Radiant Software -[C:/primary_gsrd/soc_primary_gsrd/soc_prima
File Edit View Project Tools Window Help
Synthesize Design
Map Design
Find Text...

soc_primary_gsrd/lib/latticesemi.com/ip/sysmem
RTL Files
Testbench Files
sysmem0.cfg
soc_primary_gsrd/lib/latticesemi.com/ip/tse_to_r
RTL Files
Testbench Files
tse_to_rgmii_bridge0.cfg
soc_primary_gsrd/lib/latticesemi.com/module/pl
RTL Files
Constraint Files
Testbench Files
pll0.cfg
Pre-Synthesis Constraint Files
source/impl_1/clock_constraint.sdc
Post-Synthesis Constraint Files
source/impl_1/soc_board_constraints.pdc
Debug Files
Script Files

Figure 8.52. Pre and Post-Synthesis Constraint Files Added

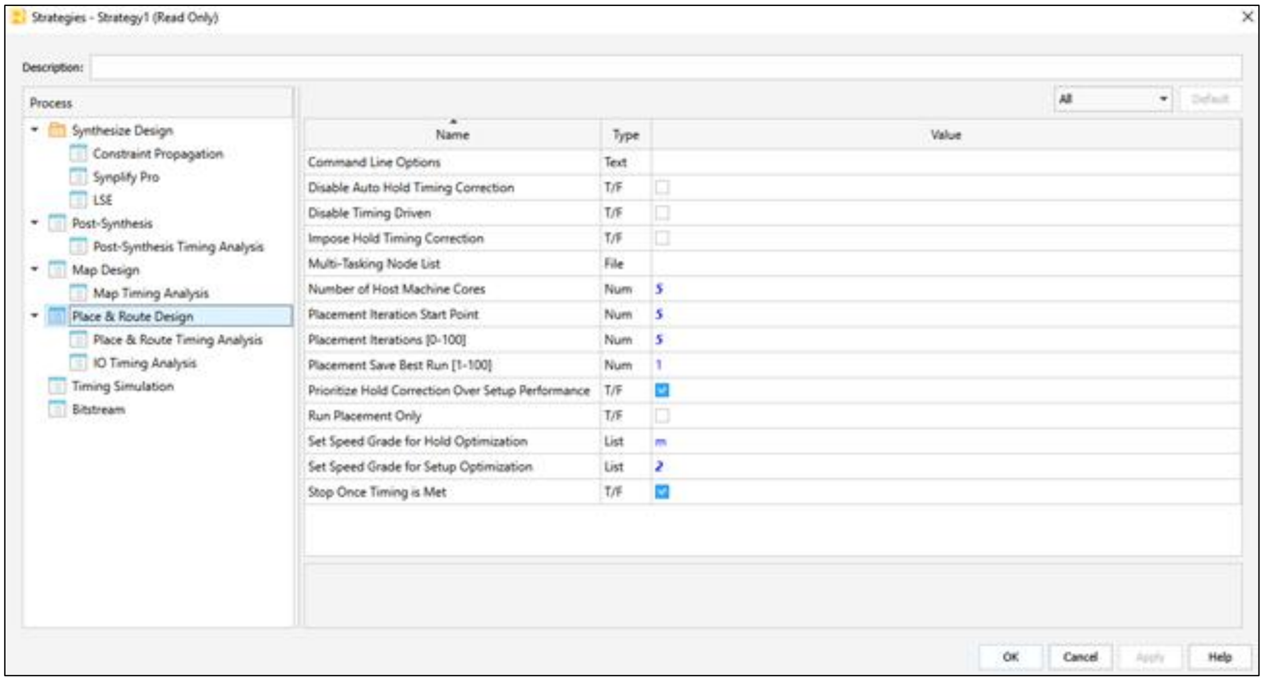
9. Double-click on Strategy1.



soc_primary_gsrd
LAV-AT-E70ES1-1LFG1156C
Strategies
Area
Timing
Strategy1

Figure 8.53. Update the Strategy

10. Place and Route Strategy used for GSRD testing.
Note: You can update these fields as per their machine and run-time requirements. However, due to this change, the Radiant tool might update the warnings and place & route details.



Name	Type	Value
Command Line Options	Text	
Disable Auto Hold Timing Correction	T/F	☐
Disable Timing Driven	T/F	☐
Impose Hold Timing Correction	T/F	☐
Multi-Tasking Node List	File	
Number of Host Machine Cores	Num	5
Placement Iteration Start Point	Num	5
Placement Iterations [0-100]	Num	5
Placement Save Best Run [1-100]	Num	1
Prioritize Hold Correction Over Setup Performance	T/F	✓
Run Placement Only	T/F	☐
Set Speed Grade for Hold Optimization	List	m
Set Speed Grade for Setup Optimization	List	2
Stop Once Timing is Met	T/F	✓

Figure 8.54. Strategy used for Avant GSRD Testing

11. Click on the **Green Run All** icon to generate a bit file. Wait for the bitstream generation and check the logs.

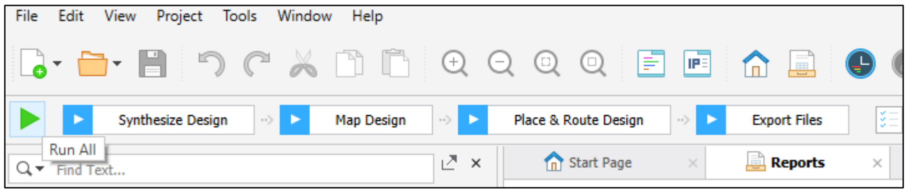


Figure 8.55. Generating the Bit File

12. The compilation flow takes some time to complete. The .bit file is generated/updated in the project path (<root_directory>/soc_primary_gsrd/ impl_1) after compilation is completed successfully as shown in Figure 8.56.

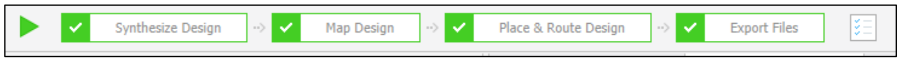


Figure 8.56. Successful Radiant Flow and Bitstream Generation

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

a. In Lattice Radiant software, go to **Project > Active Strategy > Place & Route Design Settings**.
b. Increase Placement Iterations.
c. Change Placement Save Best Run from 1 to 5.
d. Click **OK**.
e. Click on **Export Files**.

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

8.4. Generating the Multi-Boot MCS File

To generate the multi-boot MCS file, perform the following steps:

Note: Follow these steps only when you have or re-created both Golden and Primary bitstreams.

1. Launch the Lattice Programmer from Lattice Radiant as shown in Figure 8.57.

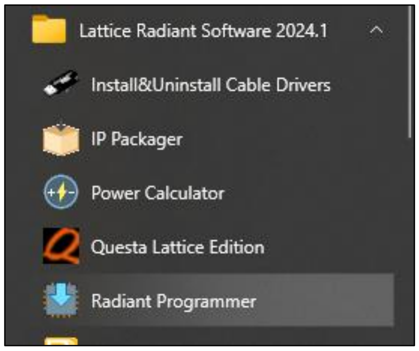


Figure 8.57. Launch Radiant Programmer from Windows Start

2. Provide the location of where you would like to store the programmer .xcf file. Click **OK**.

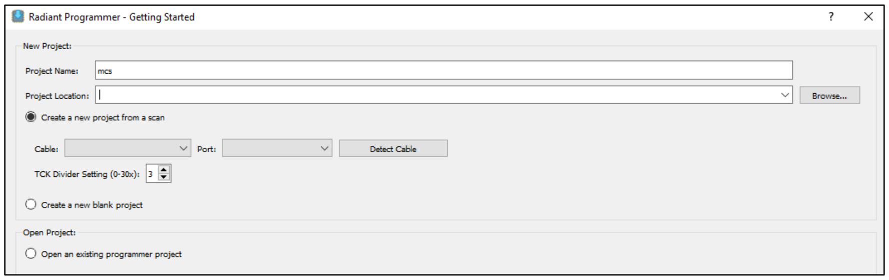
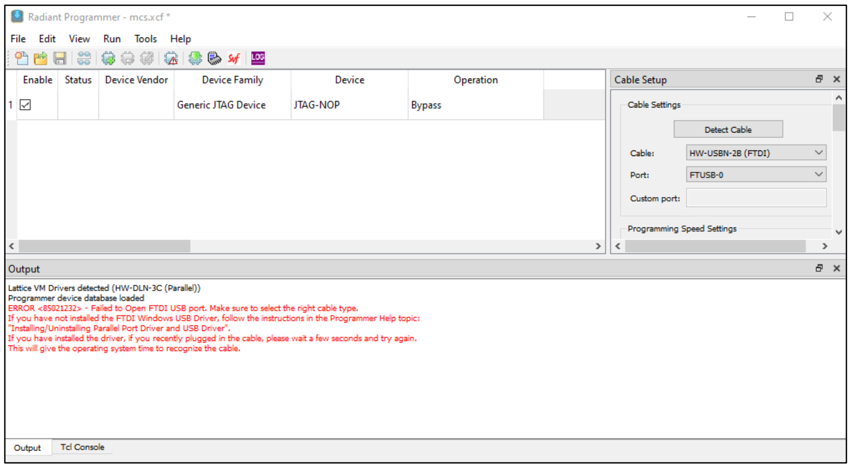

Radiant Programmer - Getting Started
New Project:
Project Name: mcs
Project Location:
Browse...
Create a new project from a scan
Cable:
Port:
Detect Cable
TCK Divider Setting (0-30x): 3
Create a new blank project
Open Project:
Open an existing programmer project

Figure 8.58. Radiant Programmer Getting Started Window

Note: This throws the error message shown in Figure 8.59 since there is no HW board connected.



	Enable	Status	Device Vendor	Device Family	Device	Operation
1	✓			Generic JTAG Device	JTAG-NOP	Bypass

```
Lattice VM Drivers detected (HW-DLN-3C (Parallel))
Programmer device database loaded
ERROR <85021232> - Failed to Open FTDI USB port. Make sure to select the right cable type.
If you have not installed the FTDI Windows USB Driver, follow the instructions in the Programmer Help topic:
"Installing/Uninstalling Parallel Port Driver and USB Driver".
If you have installed the driver, if you recently plugged in the cable, please wait a few seconds and try again.
This will give the operating system time to recognize the cable.
```

Figure 8.59. Error if No HW Board is Connected

3. Click **Tools > Deployment Tool**.

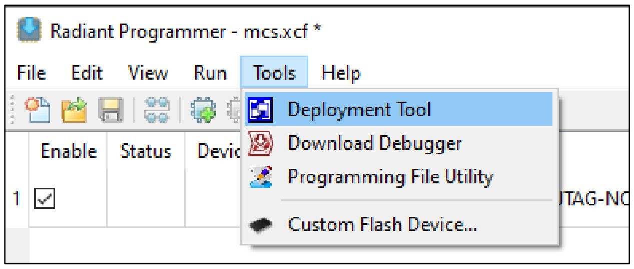
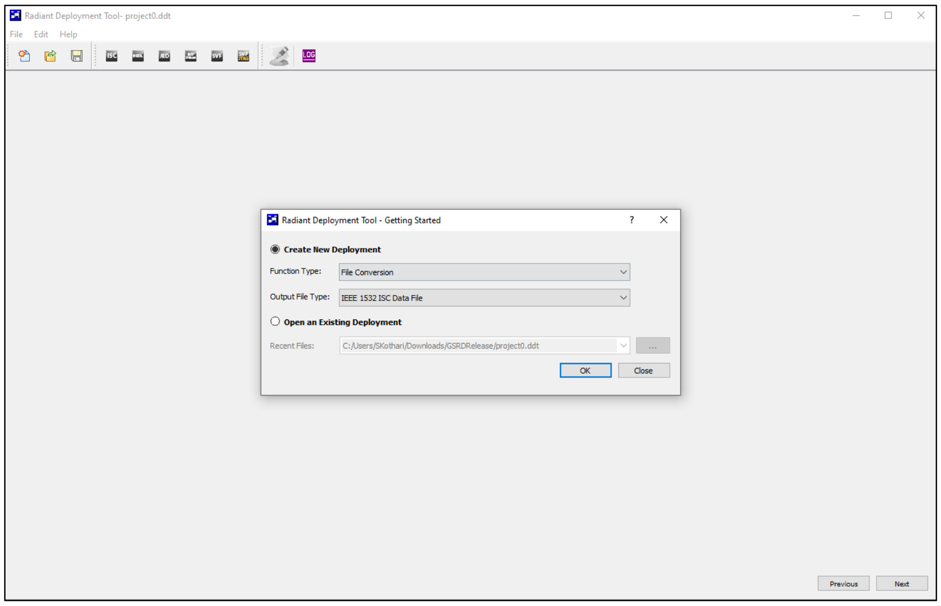


Figure 8.60. Open Deployment Tool from Radiant Programmer

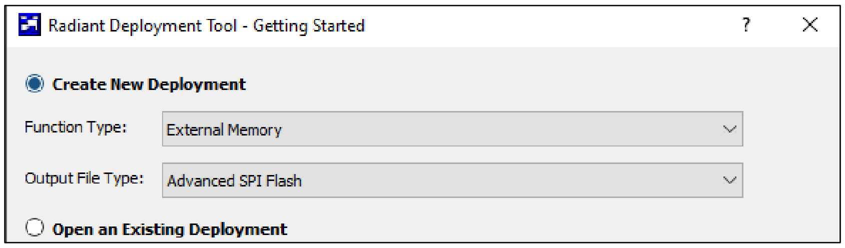
4. This opens the Deployment Tool window as shown in Figure 8.61.


Radiant Deployment Tool- project0.ddt
File
Edit
Help
Radiant Deployment Tool - Getting Started
OK
Close
Previous
Next

Create New Deployment [✓]
Function Type: File Conversion
Output File Type: IEEE 1532 ISC Data File
Open an Existing Deployment
Recent Files: C:/Users/SKothari/Downloads/GSRDRelease/project0.ddt

Figure 8.61. Deployment Tool Start Window

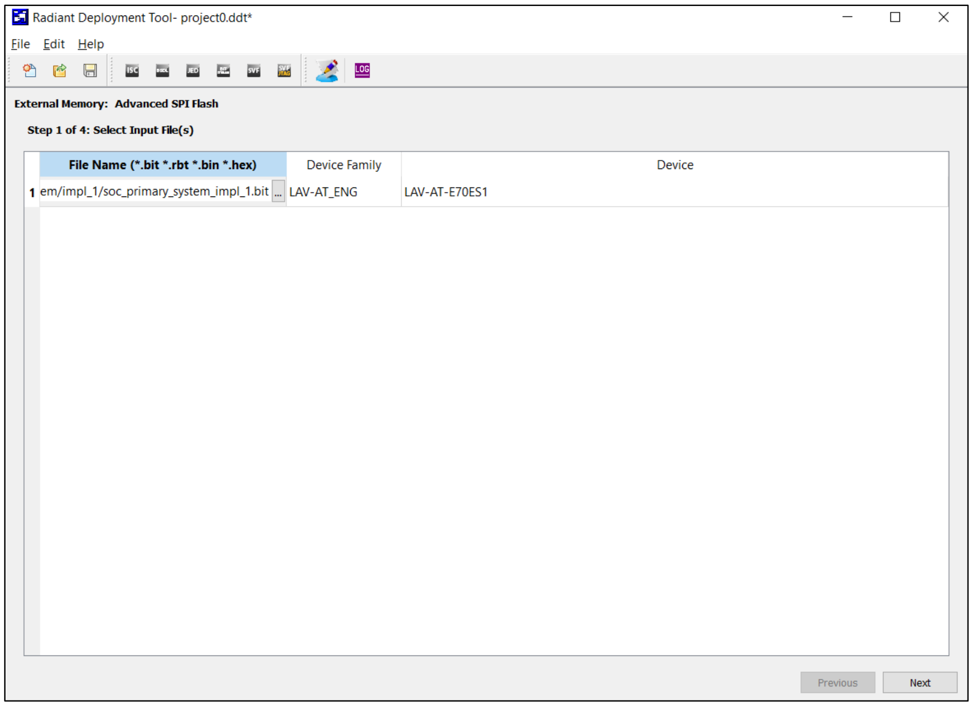
5. Apply the settings as shown in Figure 8.62 and click **OK**.



Create New Deployment [✓]
Function Type: External Memory
Output File Type: Advanced SPI Flash
Open an Existing Deployment

Figure 8.62. Options for Creating New Deployment

6. Step 1 of 4: Select the Input File(s) window, as shown in Figure 8.63:
    a. Click the **File Name field** to browse and select the **primary gsrd's .bit** from your primary soc project.
    b. The Device Family and Device fields auto populates based on the bitstream .bit file selected.
    c. Click **Next**.


Radiant Deployment Tool- project0.ddt*

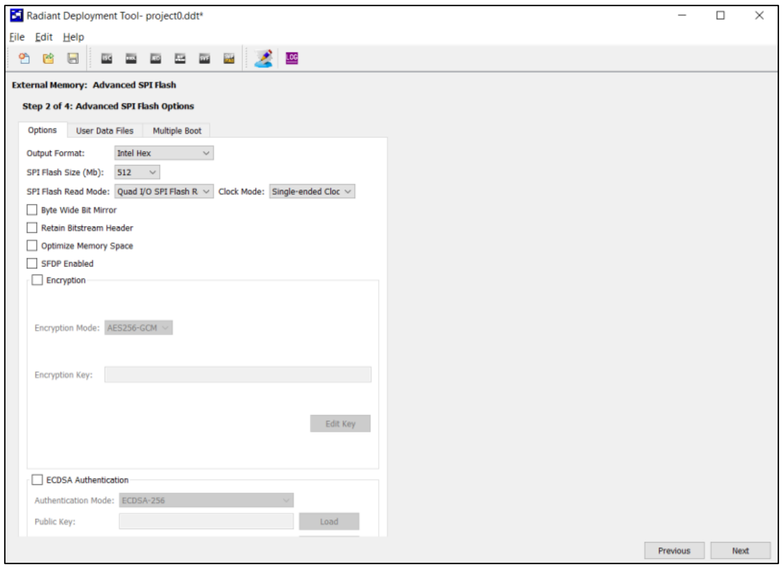
External Memory: Advanced SPI Flash

Step 1 of 4: Select Input File(s)

	File Name (*.bit *.rbt *.bin *.hex)	Device Family	Device
1	em/impl_1/soc_primary_system_impl_1.bit	LAV-AT_ENG	LAV-AT-E70ES1

Figure 8.63. External Memory Step 1 of 4: Select Input Files

7. In Step 2 of 4: Select the fields as shown in Figure 8.64.
   a. Under Options tab: Choose Output Format as **Intel Hex** and **SPI Flash Size(Mb)** as **512**.


Radiant Deployment Tool- project0.ddt*

External Memory: Advanced SPI Flash

Step 2 of 4: Advanced SPI Flash Options

Output Format: Intel Hex

SPI Flash Size (Mb): 512

SPI Flash Read Mode: Quad I/O SPI Flash R Clock Mode: Single-ended Cloc

☐ Byte Wide Bit Mirror

☐ Retain Bitstream Header

☐ Optimize Memory Space

☐ SFDP Enabled

☐ Encryption

Encryption Mode: AES256-GCM

Encryption Key:

☐ ECDSA Authentication

Authentication Mode: ECDSA-256

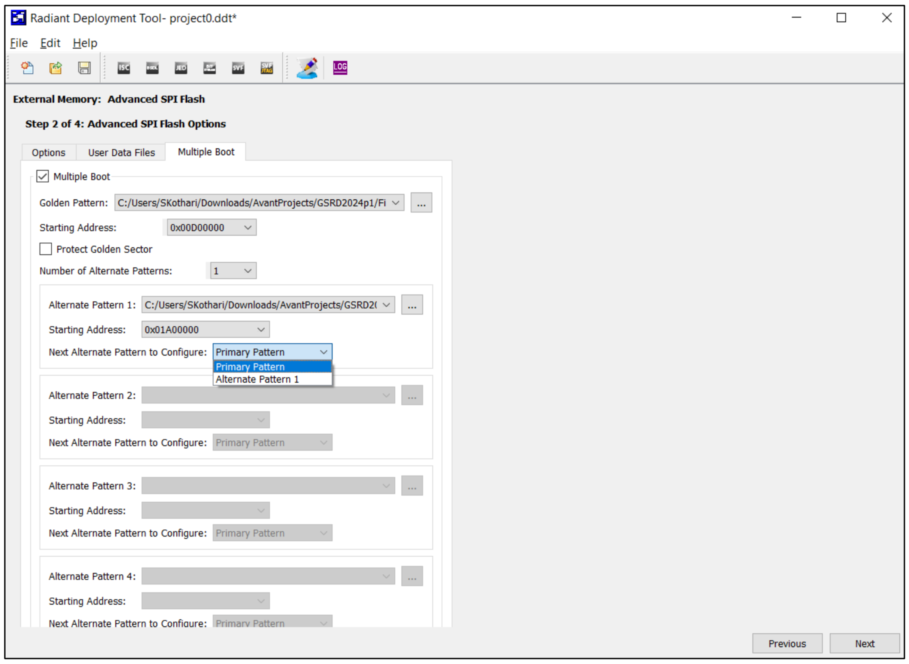
Public Key:

Figure 8.64. External Memory Step 2 of 4: Select Options

   b. No changes needed in **User Data Files** field.
   c. Under Multiple Boot tab.
      i. Select the Multiple Boot Option.
      ii. Select Number of Alternate Patterns as 1.
      iii. Under the Golden Pattern, browse and select the **Golden pattern bitstream** file. The Starting Address of Golden Pattern is automatically assigned. You can change it by clicking on the drop-down menu.

iv. Under Alternate Pattern 1 field, click on the browse button and select the **golden soc gsrd's bitstream**. The Starting Address of this pattern is automatically assigned. You can change it by clicking on the drop-down menu. This is the pattern loaded during an event of next PROGRAMN/REFRESH or soft reset.

v. Click Next.


Radiant Deployment Tool- project0.ddt*

External Memory: Advanced SPI Flash

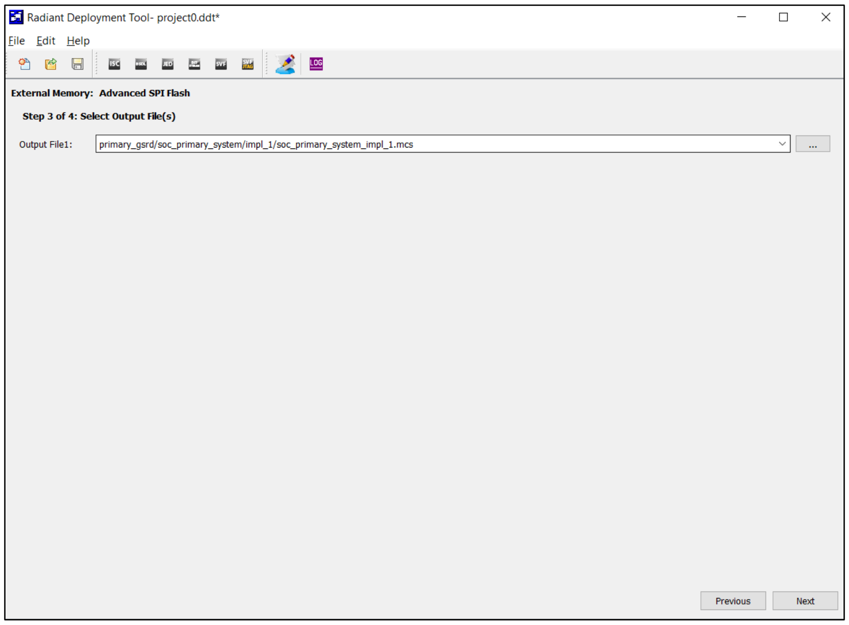
Step 2 of 4: Advanced SPI Flash Options

Multiple Boot: ✓
Golden Pattern: C:/Users/SKothari/Downloads/AvantProjects/GSRD2024p1/Fi
Starting Address: 0x00D00000
Protect Golden Sector: ☐
Number of Alternate Patterns: 1

Alternate Pattern 1: C:/Users/SKothari/Downloads/AvantProjects/GSRD2(
Starting Address: 0x01A00000
Next Alternate Pattern to Configure: Primary Pattern

Figure 8.65. External Memory Step 2 of 4: Multi-Boot

8. In Step 3 of 4: Select the Output File(s) as shown in Figure 8.66. Choose the location on your machine to generate an .mcs file.


Radiant Deployment Tool- project0.ddt*

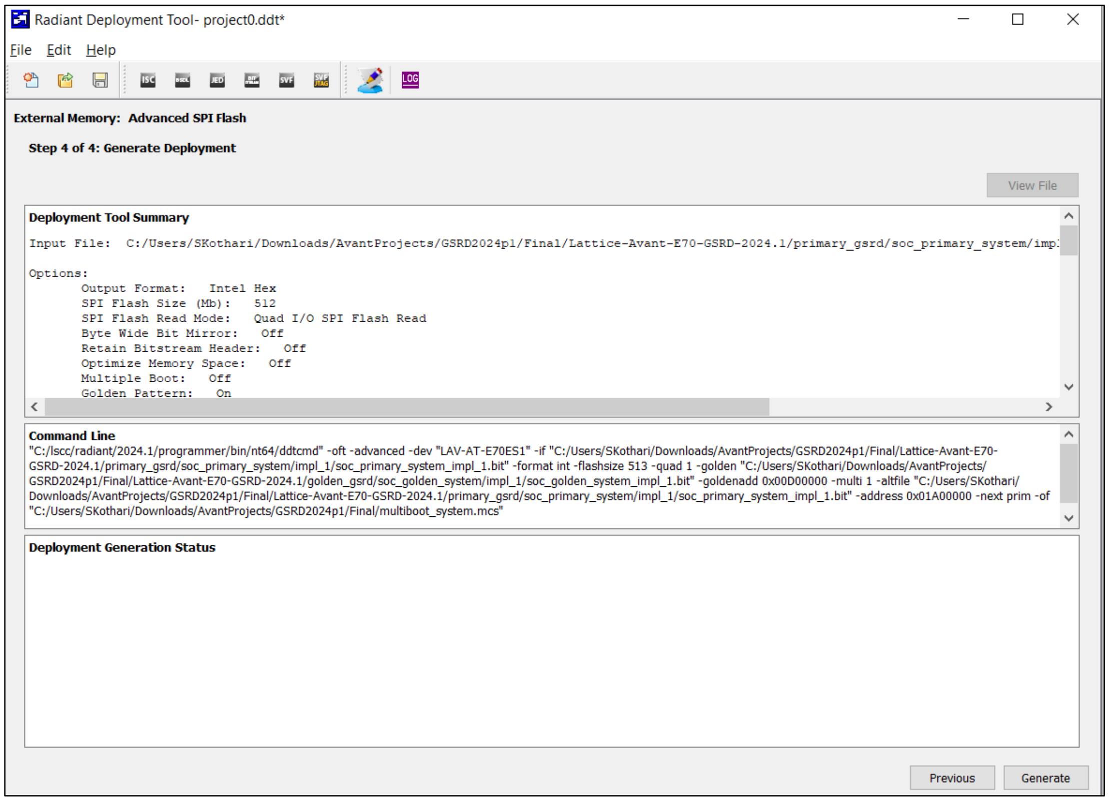
External Memory: Advanced SPI Flash

Step 3 of 4: Select Output File(s)

Output File1: primary_gsrd/soc_primary_system/impl_1/soc_primary_system_impl_1.mcs

Figure 8.66. External Memory Step 3 of 4: Select Output File(s)

9. Click **Next**.
10. In Step 4 of 4: Generate Deployment, click **Generate** at the bottom right corner as shown in Figure 8.67.



External Memory: Advanced SPI Flash

Step 4 of 4: Generate Deployment

Deployment Tool Summary

```
Input File:  C:/Users/SKothari/Downloads/AvantProjects/GSRD2024p1/Final/Lattice-Avant-E70-GSRD-2024.1/primary_gsrd/soc_primary_system/imp

Options:
       Output Format:   Intel Hex
       SPI Flash Size (Mb):   512
       SPI Flash Read Mode:   Quad I/O SPI Flash Read
       Byte Wide Bit Mirror:   Off
       Retain Bitstream Header:   Off
       Optimize Memory Space:   Off
       Multiple Boot:   Off
       Golden Pattern:   On
```

Command Line

```
"C:/lscc/radiant/2024.1/programmer/bin/nt64/ddtcmd" -oft -advanced -dev "LAV-AT-E70ES1" -if "C:/Users/SKothari/Downloads/AvantProjects/GSRD2024p1/Final/Lattice-Avant-E70-
GSRD-2024.1/primary_gsrd/soc_primary_system/impl_1/soc_primary_system_impl_1.bit" -format int -flashsize 513 -quad 1 -golden "C:/Users/SKothari/Downloads/AvantProjects/
GSRD2024p1/Final/Lattice-Avant-E70-GSRD-2024.1/golden_gsrd/soc_golden_system/impl_1/soc_golden_system_impl_1.bit" -goldenadd 0x00D00000 -multi 1 -altfile "C:/Users/SKothari/
Downloads/AvantProjects/GSRD2024p1/Final/Lattice-Avant-E70-GSRD-2024.1/primary_gsrd/soc_primary_system/impl_1/soc_primary_system_impl_1.bit" -address 0x01A00000 -next prim -of
"C:/Users/SKothari/Downloads/AvantProjects/GSRD2024p1/Final/multiboot_system.mcs"
```

Deployment Generation Status

Figure 8.67. External Memory Step 4 of 4: General Development

11. Check for the following output as shown in Figure 8.68.

```
Lattice Radiant Deployment Tool has exited successfully.
```

Figure 8.68. MCS File Generated Successfully

12. The generated final .mcs file is now ready to be programmed into the external flash using the Radiant Programmer.
13. Close the Deployment Tool window.

9. Customizing the IP in the GSRD Design

This section describes the customization that can be applied to IP in the reference design. The hardware related modifications are made using Propel Builder, since it is the main design entry tool. The software related modifications are made using Propel SDK.

The following demonstrates this using an example of QSPI Flash Controller IP.

9.1. Changing QSPI Flash Controller User Interface Parameters

The QSPI Flash Controller IP parameters are set according to the SPI Flash chips available on Avant-AT-E Board. You can change the IP parameters to suit the board and Flash chip in your project.

To modify the IP parameters:

1. In Propel Builder, double-click on the **qspi0_inst** block.

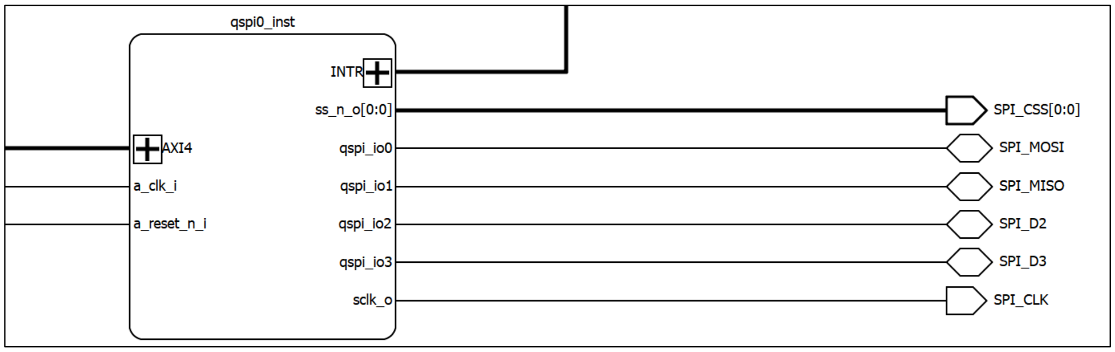
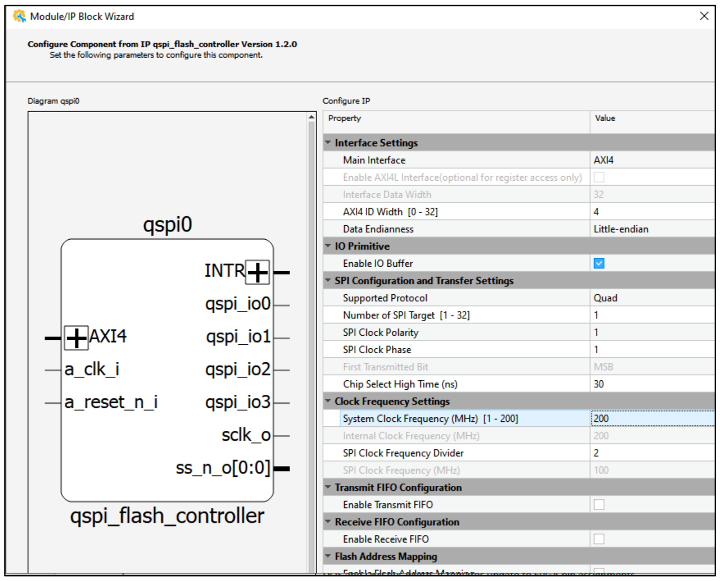


Figure 9.1. IP Block

2. In the Module/IP Block Wizard, make the desired changes to the parameters that suits your board and Flash chip. For example, you'd like to change any or all.
   - Data Endianness from Big Endian to Little Endian and
   - System Clock Frequency from 100 to 200



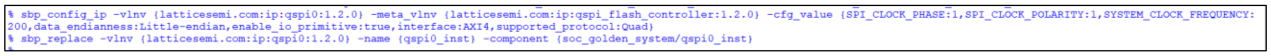
Configure Component from IP qspi_flash_controller Version 1.2.0
Set the following parameters to configure this component.

Property	Value
Interface Settings	
Main Interface	AXI4
Enable AXI4L Interface(optional for register access only)	☐
Interface Data Width	32
AXI4 ID Width [0 - 32]	4
Data Endianness	Little-endian
IO Primitive	
Enable IO Buffer	✓
SPI Configuration and Transfer Settings	
Supported Protocol	Quad
Number of SPI Target [1 - 32]	1
SPI Clock Polarity	1
SPI Clock Phase	1
First Transmitted Bit	MSB
Chip Select High Time (ns)	30
Clock Frequency Settings	
System Clock Frequency (MHz) [1 - 200]	200
Internal Clock Frequency (MHz)	200
SPI Clock Frequency Divider	2
SPI Clock Frequency (MHz)	100
Transmit FIFO Configuration	
Enable Transmit FIFO	☐
Receive FIFO Configuration	
Enable Receive FIFO	☐
Flash Address Mapping	

Figure 9.2. Customize IP

3. Click **Generate** to make sure RTL is updated as per the customization and must generate without an error.

4. You can see the message in the TCL Console as shown in Figure 9.3. Ensure that are no errors.



```
% sbp_config_ip -vlnv {latticesemi.com:ip:qspi0:1.2.0} -meta_vlnv {latticesemi.com:ip:qspi_flash_controller:1.2.0} -cfg_value {SPI_CLOCK_PHASE:1,SPI_CLOCK_POLARITY:1,SYSTEM_CLOCK_FREQUENCY:
200,data_endianness:Little-endian,enable_io_primitive:true,interface:AXI4,supported_protocol:Quad}
% sbp_replace -vlnv {latticesemi.com:ip:qspi0:1.2.0} -name {qspi0_inst} -component {soc_golden_system/qspi0_inst}
```

Figure 9.3. TCL Console Output After Desired Customization

5. Click the **Validate** button as shown in Figure 9.4. Ensure that there are no errors.

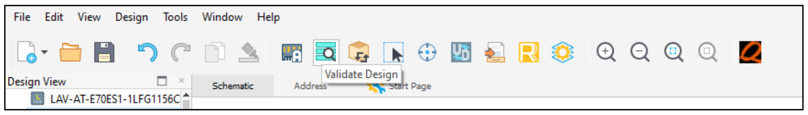


Figure 9.4. Validate Design again is any IP is updated

6. Click the **Generate** button as shown in Figure 9.5. Ensure that there are no errors.

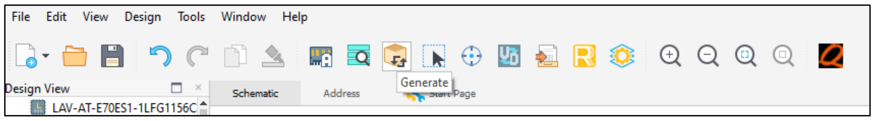


Figure 9.5. Generate again is any IP is updated

7. Once validated and generated, it updates the **sys_env.xml** file in the **sge** folder under your project directory as shown in Figure 9.6.

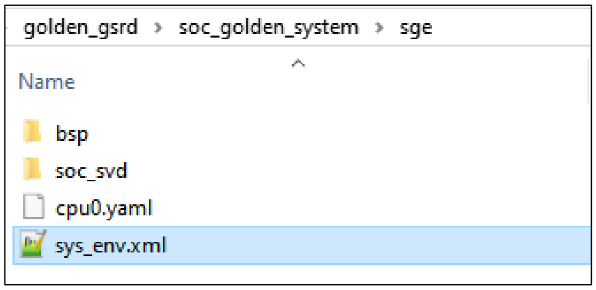


Figure 9.6. New sys_env.xml File Generated

8. If your IP is software driver dependent, the C projects must also be updated.
9. In your project under Propel SDK window, go to **Project > Update Lattice C/C++ Project**.

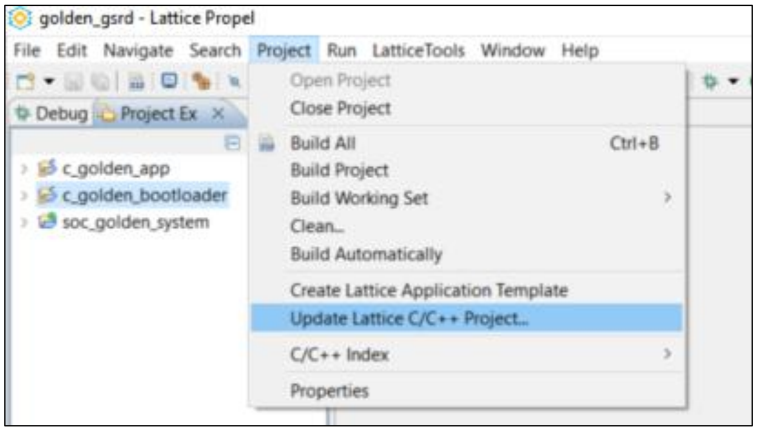
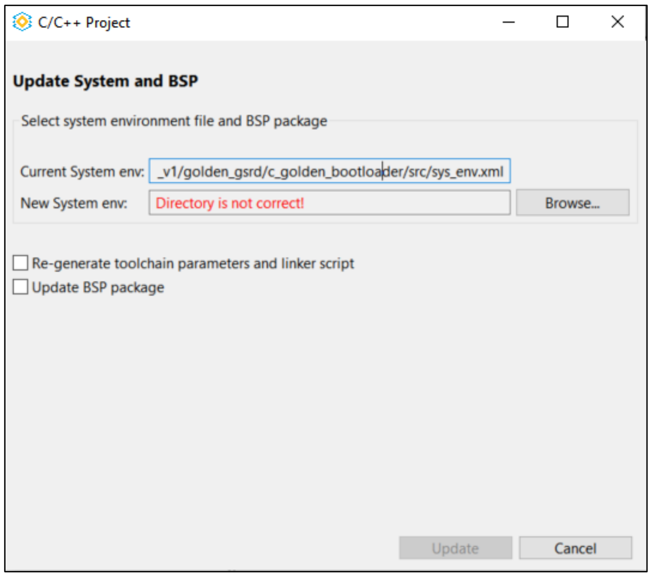


Figure 9.7. Update Lattice C/C++ Project

10. It opens the dialog box as shown in Figure 9.8.



Update System and BSP

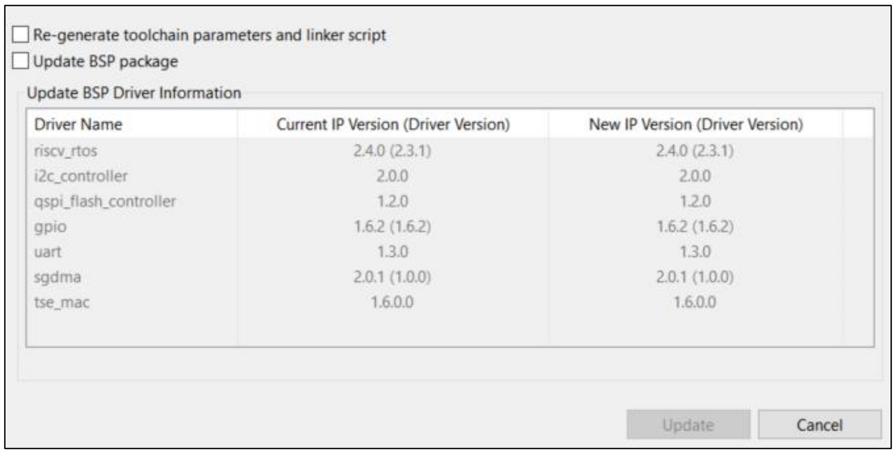
Select system environment file and BSP package

Current System env: _v1/golden_gsrd/c_golden_bootloader/src/sys_env.xml
New System env: Directory is not correct!

☐ Re-generate toolchain parameters and linker script
☐ Update BSP package

Figure 9.8. Update System Dialog Window

11. Click **Browse** and select the **sys_env.xml** file generated above.



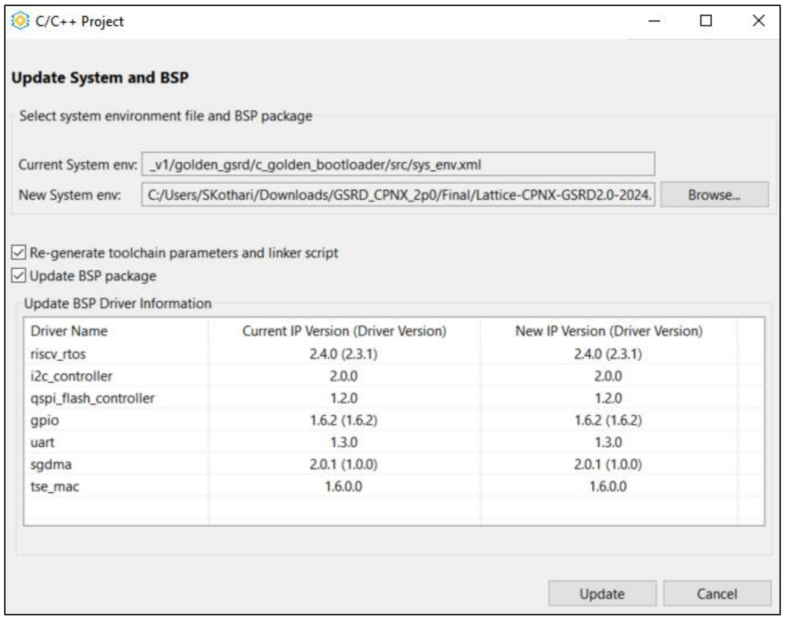
☐ Re-generate toolchain parameters and linker script
☐ Update BSP package

Update BSP Driver Information

Driver Name	Current IP Version (Driver Version)	New IP Version (Driver Version)
riscv_rtos	2.4.0 (2.3.1)	2.4.0 (2.3.1)
i2c_controller	2.0.0	2.0.0
qspi_flash_controller	1.2.0	1.2.0
gpio	1.6.2 (1.6.2)	1.6.2 (1.6.2)
uart	1.3.0	1.3.0
sgdma	2.0.1 (1.0.0)	2.0.1 (1.0.0)
tse_mac	1.6.0.0	1.6.0.0

Figure 9.9. New IP version Details

12. Select *Re-generate toolchain parameters and linker script* and *Update BSP Package* and click **Update** to reflect the latest configurations.


C/C++ Project

Update System and BSP

Select system environment file and BSP package

Current System env: _v1/golden_gsrd/c_golden_bootloader/src/sys_env.xml

New System env: C:/Users/SKothari/Downloads/GSRD_CPNX_2p0/Final/Lattice-CPNX-GSRD2.0-2024. Browse...

☑ Re-generate toolchain parameters and linker script

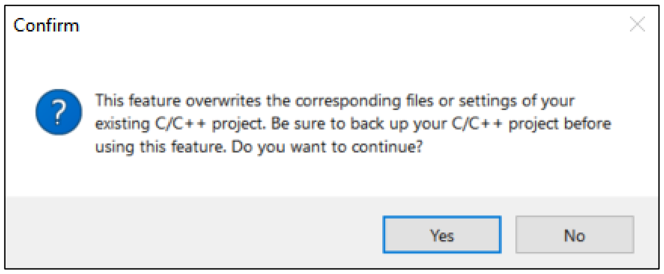
☑ Update BSP package

Update BSP Driver Information

Driver Name	Current IP Version (Driver Version)	New IP Version (Driver Version)
riscv_rtos	2.4.0 (2.3.1)	2.4.0 (2.3.1)
i2c_controller	2.0.0	2.0.0
qspi_flash_controller	1.2.0	1.2.0
gpio	1.6.2 (1.6.2)	1.6.2 (1.6.2)
uart	1.3.0	1.3.0
sgdma	2.0.1 (1.0.0)	2.0.1 (1.0.0)
tse_mac	1.6.0.0	1.6.0.0

Figure 9.10. Generate BSP and Toolchain Parameters

13. Click **Yes** as shown in Figure 9.11 and the update window closes.


Confirm

This feature overwrites the corresponding files or settings of your existing C/C++ project. Be sure to back up your C/C++ project before using this feature. Do you want to continue?

Figure 9.11. Confirm the Changes

14. Confirm that the BSP package was updated successfully. You can check it by reviewing the following:
    - **sys_platform.h** file to review the updated parameter change.

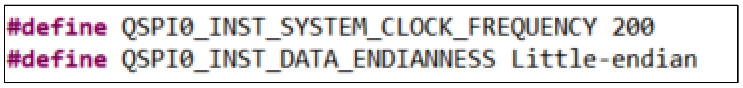

```
#define QSPI0_INST_SYSTEM_CLOCK_FREQUENCY 200
#define QSPI0_INST_DATA_ENDIANNESS Little-endian
```

Figure 9.12. Confirm the Configuration Changes in sys_platform.h

15. Clean and Build the Project.
16. If you hit any errors, it could be one of the following reasons:
    - Driver files has been updated.
    - Addresses in your C project might have changed.
17. Do the same XML updates for **c_golden_app** FreeRTOS project too.
18. Clean and Re-Build the Project.
19. Once new .mem file and binaries are created, follow the steps in the Validating and Generating the GSRD Design using Propel Builder section. Validate and generate the GSRD design using the Propel Builder.

References

- RISC-V RX CPU IP Core (FPGA-IPUG-02254)
- System Memory IP (FPGA-IPUG-02073)
- Tri-Speed Ethernet MAC IP Core (FPGA-IPUG-02084)
- LPDDR4 Memory Controller for Avant Devices (FPGA-IPUG-02208)
- QSPI Flash Controller IP Core (FPGA-IPUG-02248)
- AXI Multi Port Bridge for Memory Controller Module (FPGA-IPUG-02246)
- SGDMA Controller IP Core (FPGA-IPUG-02131)
- UART IP Core (FPGA-IPUG-02105)
- GPIO IP Core (FPGA-IPUG-02076)
- AXI4 Interconnect IP (FPGA-IPUG-02196)
- AXI to APB Bridge IP (FPGA-IPUG-02198)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Radiant FPGA design software
- Lattice Solutions IP Cores web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software User Guide
- 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, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.1, December 2025

Section	Change Summary
All	Changed document title from Golden System Reference Design and Demo User Guide for Lattice Avant-E Devices to *Golden System Reference Design and Demo User Guide v1.0 for Lattice Avant-E Devices*.
Abbreviations in This Document	Made editorial fix in the table.
Introduction	• Added the following note in the Overview of the System section. • *The Golden System Reference Design and Demonstration for Avant-E version 2.0 is available in the GHRD/GSRD Reference Design and GHRD/GSRD Demonstration web pages.* • Removed Licensing and Ordering Information section.

Revision 1.0, October 2024

Section	Change Summary
All	Initial preliminary release.