



CrossLinkU-NX USB Video Class

Reference Design

FPGA-RD-02306-1.1

September 2025

Disclaimers

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

Inclusive Language

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

Contents

Contents	3
Abbreviations in This Document.....	8
1. Introduction	9
1.1. Quick Facts	9
1.2. Features	9
1.3. Naming Conventions	10
1.3.1. Nomenclature.....	10
1.3.2. Signal Names	10
2. Directory Structure and Files	11
3. Functional Description.....	12
3.1. Design Components	12
3.1.1. PLL	12
3.1.2. USB23 Controller	14
3.1.3. AHB-Lite to LMMI Converter	14
3.1.4. RISC-V Microcontroller	14
3.1.5. System Memory	15
3.1.6. UART.....	16
3.1.7. GPIO	17
3.1.8. Lattice I2C Controller.....	18
3.1.9. AHB-Lite Interconnect	19
3.1.10. APB Interconnect.....	23
3.1.11. APB-Lite to APB Converter	24
3.1.12. RX DPHY.....	25
3.1.13. MIPI Packet Decoder	27
3.1.14. Byte to Pixel.....	27
3.1.15. B2P to AXI Stream Conversion	27
3.1.16. Debayer	27
3.1.17. Color Correction Matrix.....	29
3.1.18. AXI Stream to Parallel Conversion.....	29
3.1.19. Color Space Converter.....	29
3.1.20. YUV422 to UVC Bridge	31
3.1.21. IEBM In FIFO Interface.....	31
3.1.22. In Endpoint Buffer Manager	31
3.1.23. AHB-Lite to Memory Bridge	32
3.1.24. USB23 to AXI Bridge	32
3.1.25. RAW10 Test Pattern Generator	32
3.1.26. YUV422 Test Pattern Generator.....	32
3.1.27. YUV422 Test Pattern In FIFO Bridge	32
3.2. Clocking Scheme	32
3.2.1. Clocking Overview	32
3.3. Reset Scheme	33
3.3.1. Reset Overview	33
4. IN Endpoint Buffer Manager Architecture.....	34
4.1. IP Ports and Parameters.....	34
4.1.1. AHB-Lite Interface	34
4.1.2. AXI Interface	34
4.1.3. FIFO Interface	35
4.1.4. Miscellaneous Ports	36
4.1.5. User Configurable Parameters	36
4.2. Register Offset Map	36
4.3. Register Details	37
4.3.1. IP_VERSION	37

4.3.2.	SCRATCH.....	37
4.3.3.	INT_EN.....	37
4.3.4.	INT_SRC.....	38
4.3.5.	BUFR_CNFG.....	38
4.3.6.	HW_PARAMS_INFO.....	39
4.3.7.	CTRL.....	39
4.3.8.	BUFR_TRACKER_INFO.....	40
4.3.9.	BUFR_AVAILABILITY_INFO.....	40
4.3.10.	BUFR_XCHNG_CTRL.....	40
4.3.11.	SEL_BUFR_INFO_FOR_FW.....	41
4.3.12.	TIMESTAMP.....	41
4.3.13.	VALID_LINES_IN_FRAME.....	42
4.4.	IN Endpoint Buffer Management Architecture.....	42
4.5.	Buffer Read and Write Management Flow.....	42
4.5.1.	Buffer Write Flow.....	42
4.5.2.	Buffer Read Flow.....	43
4.6.	Core Operation.....	45
4.6.1.	IP Core Configuration or Reconfiguration.....	45
4.6.2.	FIFO Write Operation.....	45
4.6.3.	FIFO Write Examples.....	47
5.	USB23 AXI Manager to Memory Interface Bridge.....	50
5.1.	USB23 AXI Manager to Memory Bridge Architecture.....	50
5.2.	Block Description.....	50
5.2.1.	AXI Subordinate Interface.....	50
5.2.2.	Address/Data Mux.....	50
5.2.3.	FIFO.....	50
5.2.4.	Read Data Mux.....	50
5.2.5.	Memory 0.....	51
5.2.6.	Memory 1.....	51
5.2.7.	Memory 2.....	51
5.3.	IP Core Port and Parameters.....	51
5.3.1.	Clock and Global Reset.....	51
5.3.2.	Native Memory Interface.....	51
5.3.3.	AXI4 Interface.....	51
5.3.4.	Configurable Parameters.....	53
6.	UVC Reference Design Signal Description.....	54
7.	Building the Reference Design.....	55
7.1.	Running Propel SDK Project.....	55
7.1.1.	Opening Propel SDK Project.....	55
7.1.2.	Navigating Propel SDK Project.....	56
7.1.3.	Generating Output MEM file.....	57
7.2.	Running Propel Builder Design.....	58
7.2.1.	Install IPs at Local from File.....	58
7.2.2.	Opening Propel Builder Design.....	60
7.2.3.	Mapping Design with Firmware.....	61
7.2.4.	Exporting Design to Lattice Radiant Software.....	63
7.3.	Running Radiant Project.....	63
7.3.1.	Opening Radiant Project.....	63
7.3.2.	Generating Bitstream File.....	64
8.	LIFCL-33U Evaluation Board Programming.....	66
8.1.	LIFCL-33U Evaluation Board Connection for Programming.....	66
8.2.	Radiant Programmer GUI Setup.....	66
8.3.	Programming the Evaluation Board.....	67
9.	Running the Reference Design on Evaluation Board.....	68

9.1.	LIFCL-33U Evaluation Board Connection	68
9.2.	UVC Demonstration	69
10.	Customizing the Reference Design	72
10.1.	Change the Camera Resolution	72
10.1.1.	Update the user_cnfg_param.v File	72
10.1.2.	Update RISC-V Firmware	72
10.1.3.	Update Lattice Radiant IP Parameters Configuration	74
10.2.	Steps to Enable YUV422 Test Pattern	74
10.3.	Steps to Select Memory Type for the In Endpoint Buffer Manager	74
10.3.1.	Steps for IEBM Memory Type Selection	74
10.3.2.	Firmware Modification	74
10.4.	Steps to Migrate from the FCCSP104 Package to the WLCSP84 Package	75
10.4.1.	Reference Clock Selection for USB Controller	75
10.4.2.	Firmware Configuration	75
	Appendix A. Resource Utilization	77
	References	78
	Technical Support Assistance	79
	Revision History	80

Figures

Figure 2.1.	Directory Structure	11
Figure 3.1.	Reference Design Block Diagram	12
Figure 3.2.	PLL General Configuration	13
Figure 3.3.	PLL Optional Ports Configuration	14
Figure 3.4.	RISC-V MC IP Configuration	15
Figure 3.5.	System Memory General Configuration	15
Figure 3.6.	System Memory Port S0 Configuration	16
Figure 3.7.	System Memory Port S1 Configuration	16
Figure 3.8.	UART Configuration	17
Figure 3.9.	Hardware Version GPIO Configuration	17
Figure 3.10.	Test Pattern Enable GPIO Configuration	18
Figure 3.11.	Lattice I2C Controller Configuration	19
Figure 3.12.	AHB-Lite Interconnect General Configuration	19
Figure 3.13.	AHB-Lite Interconnect Main Configuration	20
Figure 3.14.	AHB-Lite Interconnect Manager Priority Setting	21
Figure 3.15.	AHB-Lite Interconnect Max Burst Setting	22
Figure 3.16.	APB Interconnect General Configuration	23
Figure 3.17.	APB Interconnect Main Settings Configuration	24
Figure 3.18.	APB-Lite to APB Converter Configuration	25
Figure 3.19.	RX DPHY General Configuration	25
Figure 3.20.	RX DPHY RX FIFO Configuration	26
Figure 3.21.	RX DPHY Soft PHY Configuration	26
Figure 3.22.	Byte to Pixel IP Configuration	27
Figure 3.23.	Debayer Configuration	28
Figure 3.24.	Debayer Test Parameters Configuration	28
Figure 3.25.	Color Correction Matrix Configuration	29
Figure 3.26.	Color Space Converter Input/Coefficient Configuration	30
Figure 3.27.	Color Space Converter Output Configuration	31
Figure 3.28.	IEBM IP Configuration	32
Figure 3.29.	Reference Design Clock Domain Block Diagram	33
Figure 3.30.	Reference Design Reset Scheme Diagram	33
Figure 4.1.	IN Endpoint Buffer Management Block Diagram	42

Figure 4.2. Buffer Write Operation Flow	43
Figure 4.3. Buffer Read Operation Flow	44
Figure 4.4. FIFO Operation Example – Writing 512 Bytes	47
Figure 4.5. FIFO Operation Example – Writing 1024 Bytes	47
Figure 4.6. FIFO Operation Example – Writing 513 Bytes	48
Figure 4.7. FIFO Operation Example – Writing 10 Bytes	48
Figure 4.8. FIFO Operation Example – Zero Length Packet Request	48
Figure 4.9. FIFO Operation Example – Invalid IN FIFO Access	49
Figure 5.1. USB23 AXI Bridge to Memory Bridge Architecture	50
Figure 7.1. Launch Lattice Propel SDK	55
Figure 7.2. Propel SDK Project Opened in Lattice Propel SDK	56
Figure 7.3. Navigating the Propel SDK Project	56
Figure 7.4. Set Propel Build Configuration Mode	57
Figure 7.5. Launch the Project Building Process	58
Figure 7.6. Install IP from File	59
Figure 7.7. Accept IP License Agreement	59
Figure 7.8. IPs Installed from Files	60
Figure 7.9. Open the Propel Builder Design	61
Figure 7.10. Opening System Memory Module Block Wizard	61
Figure 7.11. Mapping the System Memory Initialization File	62
Figure 7.12. Base Address Assignment	62
Figure 7.13. Export the Propel Builder Design to Radiant Software	63
Figure 7.14. Lattice Radiant Software	63
Figure 7.15. Open the Radiant Design	64
Figure 7.16. Generated Bitstream Log	65
Figure 7.17. Place & Route Design Strategy	65
Figure 8.1. Radiant Programmer GUI	66
Figure 8.2. Program the Device	67
Figure 9.1. CrossLinkU-NX Evaluation Board Setup for Demonstration	68
Figure 9.2. USB Enumeration	68
Figure 9.3. Open Webcam/Other Device in PotPlayer	69
Figure 9.4. Video Output on PotPlayer	69
Figure 9.5. Video Renderer Option	70
Figure 9.6. Playback/System Information	70
Figure 9.7. Example of Video Information	71
Figure 10.1. User Configuration Parameter File	72
Figure 10.2. Camera PLL Configurations	73
Figure 10.3. Camera PLL Configurations	73
Figure 10.4. Firmware Code to Enable the UYV422 Test Pattern	74
Figure 10.5. Firmware Configurable Registers	75
Figure 10.6. USB Reference Clock Selection	76

Tables

Table 1.1. Summary of the Reference Design	9
Table 2.1. File List	11
Table 4.1. AHB-Lite Subordinate Interface	34
Table 4.2. AXI Subordinate Interface	34
Table 4.3. FIFO Interface	35
Table 4.4. Miscellaneous Ports	36
Table 4.5. User Configurable Parameters	36

Table 4.6. IEBM Register Offset Map.....	36
Table 4.7. IP_VERSION, Offset = 0x00.....	37
Table 4.8. SCRATCH, Offset = 0x04	37
Table 4.9. INT_EN, Offset = 0x08	37
Table 4.10. INT_SRC, Offset = 0x0C	38
Table 4.11. BUFR_CNFG, Offset = 0x10	38
Table 4.12. HW_PARAMS_INFO, Offset = 0x14	39
Table 4.13. CTRL, Offset = 0x18	39
Table 4.14. BUFR_TRACKER_INFO, Offset = 0x1C.....	40
Table 4.15. BUFR_AVAILABILITY_INFO, Offset = 0x20.....	40
Table 4.16. BUFR_XCHNG_CTRL, Offset = 0x24.....	40
Table 4.17. SEL_BUFR_INFO_FOR_FW, Offset = 0x28	41
Table 4.18. TIMESTAMP_W0, Offset = 0x30.....	41
Table 4.19. TIMESTAMP_W0, Offset = 0x34.....	42
Table 4.20. BUFR_XCHNG_CTRL (Offset = 0x24)	42
Table 4.21. Input Ports Related to FIFO Write Operation	45
Table 5.1. Clock and Reset Ports.....	51
Table 5.2. Native Memory Interface Ports	51
Table 5.3. AXI4 Interface Ports	51
Table 5.4. User Configurable Parameters.....	53
Table 6.1. Primary I/O.....	54
Table A.1. Resource Utilization for LIFCL-33U-9CTG104C	77

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviations	Definition
AXI4	Advanced Extensible Interface 4
AHB	Advanced High-Performance Bus
AHB-Lite	Advanced High-Performance Bus – Lite
APB	Advanced Peripheral Bus
BTP	Byte to Pixel
CCM	Color Correction Matrix
CPU	Central Processing Unit
CSI-2	Camera Serial Interface 2
DMA	Direct Memory Access
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPS	Frame Per Second
GPIO	General Purpose Input/Output
HAL	Hardware Abstraction Layer
IEBM	In Endpoint Buffer Manager
IEDS	In Endpoint Data Source
I ² C	Inter-Integrated Circuit
INT	Interrupt
IP	Intellectual Property
IRQ	Interrupt Request
ISP	Image Signal Processing
LED	Light Emitting Diode
LMMI	Lattice Memory Mapped Interface
MC	Microcontroller
MIPI	Mobile Industry Processor Interface
PC	Personal Computer
PLL	Phase-Locked Loop
RAM	Random Access Memory
RGB	Red Green Blue
RISC-V	Reduced Instruction Set Computer – Five
RTL	Register Transfer Level
SCL	Serial Clock Line
SDA	Serial Data Line
SPI	Serial Peripheral Interface
TDP	True Dual Port
TRB	Transfer Request Block
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UVC	USB Video Class

1. Introduction

The Lattice Semiconductor CrosslinkU™-NX USB Video Class (UVC) reference design provides you with a template for video streaming from a camera sensor, utilizing the USB hard IP in a CrosslinkU-NX device.

1.1. Quick Facts

Download the reference design files from the [UVC reference design](#) web page.

Table 1.1. Summary of the Reference Design

General	Target Devices	LIFCL-33U
	Source Code Format	C code, RTL
Simulation	Functional Simulation	Not supported.
	Timing Simulation	Not supported.
	Hardware Validation	Fully validated.
Software Requirements	Software Tool and Version	<ul style="list-style-type: none"> Lattice Propel™ SDK 2024.2 Lattice Propel Builder 2024.2 Lattice Radiant™ Software Version 2024.2 Lattice Radiant Programmer Version 2024.2
	Propel Soft IP Version	<ul style="list-style-type: none"> RISC-V MC Version 2.7.0 System Memory Version 2.3.0 AHB-Lite Interconnect Version 1.3.2 AHB-Lite to APB Bridge Version 1.1.2 AHB-Lite Feedthrough Version 1.0.0 APB Interconnect Version 1.2.1 GPIO Version 1.6.2 I2C Controller Version 2.0.1 UART Version 1.3.0
	Radiant Soft IP Version	<ul style="list-style-type: none"> Byte to Pixel Converter Version 1.7.0 Color Space Converter Version 2.2.0 Debayer Version 1.2.2 CSI-2/DSI D-PHY Receiver Version 1.7.0 FIFO_DC Version 2.3.0
Hardware Requirements	Board	LIFCL-33U-EVN Evaluation Board REV-B
	Camera Sensor	Raspberry PI Camera Module V2
	Cable	<ul style="list-style-type: none"> USB C to USB C cable, or USB C to USB A 9-pin cable USB A to Micro USB cable

1.2. Features

Key features of the CrosslinkU-NX USB Video Class reference design include:

- Soft Mobile Industry Processor Interface (MIPI) D-PHY and Camera Serial Interface (CSI-2) for image sensor aggregation. The Lattice Semiconductor D-PHY Receiver IP converts CSI-2 data to 8-bit data.
- The Lattice Semiconductor Byte-to-Pixel Converter IP converts CSI-2 standard based video payload packets from the D-PHY Receiver module output to pixel format.
- The Lattice Semiconductor Debayer IP converts raw image data into an RGB image.
- Hardware USB IP for USB video class to stream the video from the image sensor to a PC through USB Type C connector.
- This reference design primarily targets the Raspberry PI Camera Module 2.
Refer to <https://www.raspberrypi.com/products/camera-module-v2/> for more information.

1.3. Naming Conventions

1.3.1. Nomenclature

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

1.3.2. Signal Names

- `_n` are active low, asserted when value is logic 0.
- `_i` are input signals.
- `_o` are output signals.

2. Directory Structure and Files

Figure 2.1 shows the directory structure.

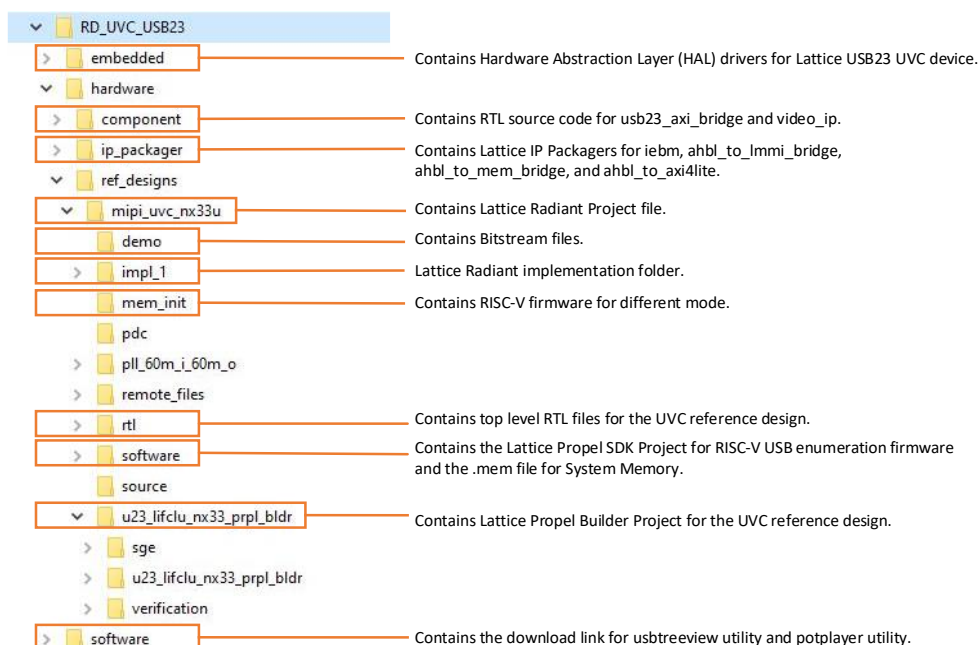


Figure 2.1. Directory Structure

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

Table 2.1. File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis closed box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.

The top-level block diagram of the CrosslinkU-NX USB Video Class reference design is shown in [Figure 3.1](#) below.



The CrosslinkU-NX USB Video Class reference design includes blocks described in the following sections, from [PLL](#) to [YUV422 Test Pattern In FIFO Bridge](#).

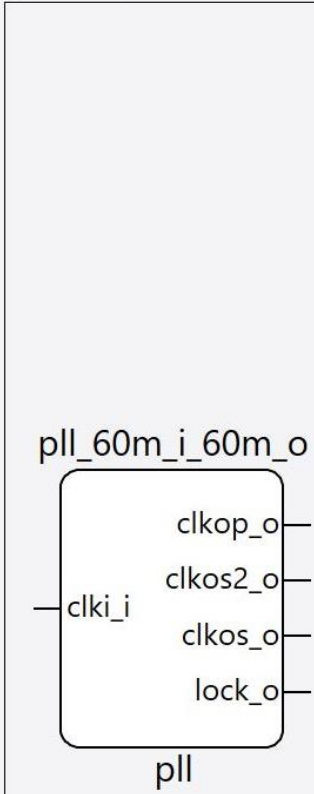
To operate some modules, a 60 MHz clock and a 72 MHz clock are needed. These clock signals can be generated from a 60 MHz input clock produced by the on-board crystal oscillator, and a Phase-Locked Loop (PLL) is utilized to do this. The PLL generates the 60 MHz clock and a 72 MHz clock signals, which are then routed to the submodules for further processing. MIPI RX-DPHY and the whole system operate at 72 MHz. USB23 PHY operates at 60 MHz PLL clock and all the Image Signal Processing (ISP) blocks operate on the MIPI byte clock.

Figure 3.2 and Figure 3.3 show the general and optional ports configuration of the PLL IP.

Module/IP Block Wizard

Configure Component from Module pll Version 1.9.0
Set the following parameters to configure this component.

Diagram pll_60m_i_60m_o



pll

Configure IP

Property	Value
General	
Configuration Mode	Frequency
Set Parameter Optimization Target	Minimum Jitter (Higher VCO)
Enable Fractional-N Divider	<input checked="" type="checkbox"/>
Enable Spread Spectrum Clock Generation	<input type="checkbox"/>
Enable Internal Path Switching	<input type="checkbox"/>
VCO Frequency [800 - 1600]	1080
Reference Clock	
CLKI: Frequency (MHz) [18 - 800]	60
CLKI: Divider Actual Value [1 - 44]	1
Phase Detector Frequency (MHz) [18 - 100]	60
Enable Reference Clock Monitor	<input type="checkbox"/>
Feedback	
CLKFB: Feedback Mode	INTCLKOP
CLKFB: FBK Divider Actual Value (Integer) [1 - 128]	18
CLKFB: FBK Divider Actual Value (Fractional) [0 - 4095]	0
CLKFB: FBK Divider Actual Value (Float)	18
Primary Clock Output	
CLKOP: Frequency Desired Value (MHz) [6.25 - 800]	60
CLKOP: Divider Actual Value [1 - 128]	18
CLKOP Tolerance (%)	0.0
CLKOP: ERROR (PPM)	0
CLKOP: Enable Trim for CLKOP	<input type="checkbox"/>
Secondary Clock Output	
CLKOS: Enable	<input checked="" type="checkbox"/>
CLKOS: Frequency Desired Value (MHz) [6.25 - 800]	72
CLKOS: Divider Actual Value [1 - 128]	15
CLKOS Tolerance (%)	0.0
CLKOS: ERROR (PPM)	0
CLKOS: Static Phase Shift (Degrees)	0
CLKOS: Enable Trim for CLKOS	<input type="checkbox"/>
Secondary Clock Output (2)	
CLKOS2: Enable	<input checked="" type="checkbox"/>
CLKOS2: Frequency Desired Value (MHz) [6.25 - 800]	72
CLKOS2: Divider Actual Value [1 - 128]	15
CLKOS2 Tolerance (%)	0.0
CLKOS2: ERROR (PPM)	0
CLKOS2: Static Phase Shift (Degrees)	0
Secondary Clock Output (3)	
CLKOS3: Enable	<input type="checkbox"/>

[User Guide](#)

No DRC issues are found.

Calculate

Generate Cancel

Figure 3.2. PLL General Configuration

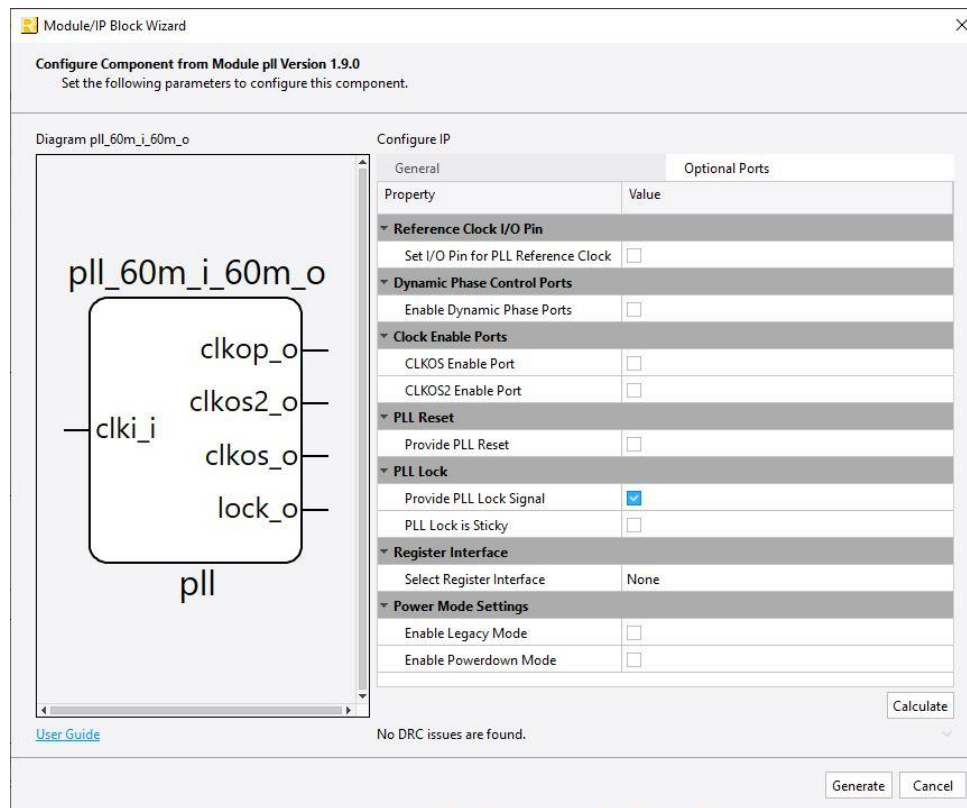


Figure 3.3. PLL Optional Ports Configuration

3.1.2. USB23 Controller

This is the USB23 Device Controller hardened IP. This IP communicates with the system memory through the Advanced eXtensible Interface 4 (AXI4) to Advanced High-Performance Bus – Lite (AHB-Lite) convertor module.

The AXI Direct Memory Access (DMA) Controller of the USB23 Controller is connected to two different memories through the AXI Interconnect Bridge.

- For Bulk transfer, the USB controller fetches data from the In Endpoint Buffer Manager (IEBM) module's data Random Access Memory (RAM). This RAM is not shared with any other controller and/or RISC-V processor.
- For all other operations, the USB controller talks to the system memory, which is shared with the RISC-V processor.

This kind of architecture helps to achieve higher Bulk throughput.

3.1.3. AHB-Lite to LMMI Converter

The USB23 controller register interface can be configured through its Lattice Memory Mapped Interface (LMMI). Since RISC-V MC supports only the AHB-Lite interface, we need an AHB-Lite to LMMI converter.

3.1.4. RISC-V Microcontroller

All the serial peripheral blocks are connected to the RISC-V MC soft processor through the AHB-Lite interface. The RISC-V MC CPU IP processes data and instructions while monitoring external interrupts coming from the USB23 controller and IEBM core. The RISC-V MC processor configuration is shown in Figure 3.4 below.

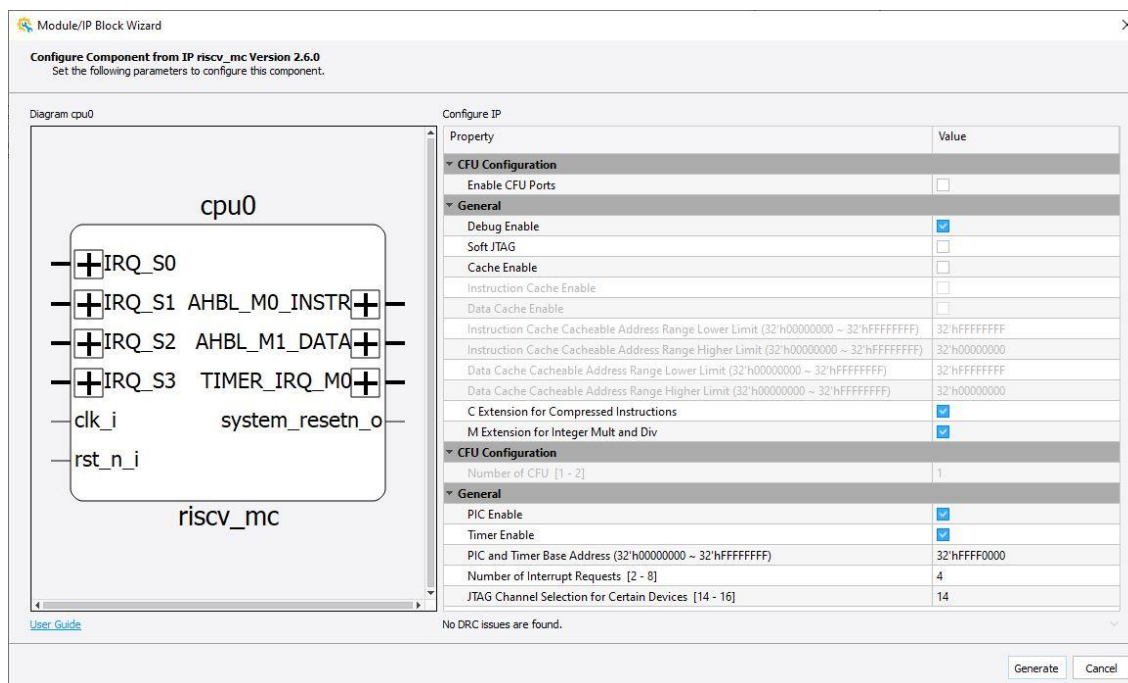


Figure 3.4. RISC-V MC IP Configuration

3.1.5. System Memory

This IP is a soft IP for memory which is used by the RISC-V MC processor. The processor reads the instructions stored in this memory through the AHB-Lite interface and takes actions accordingly. This IP has two interfaces, AHB-Lite and AXI4, through which the manager can communicate with it. The AHB-Lite interface is used to communicate with the RISC-V processor. The System Memory IP configuration is shown in Figure 3.5, Figure 3.6, and Figure 3.7.

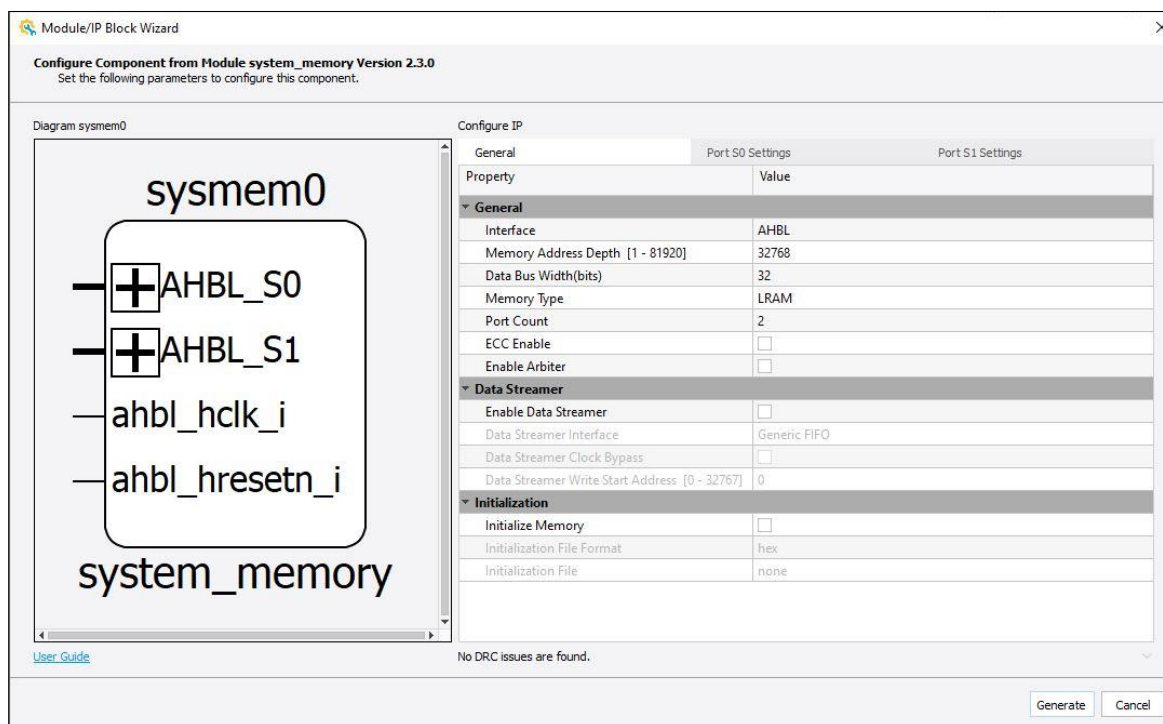


Figure 3.5. System Memory General Configuration

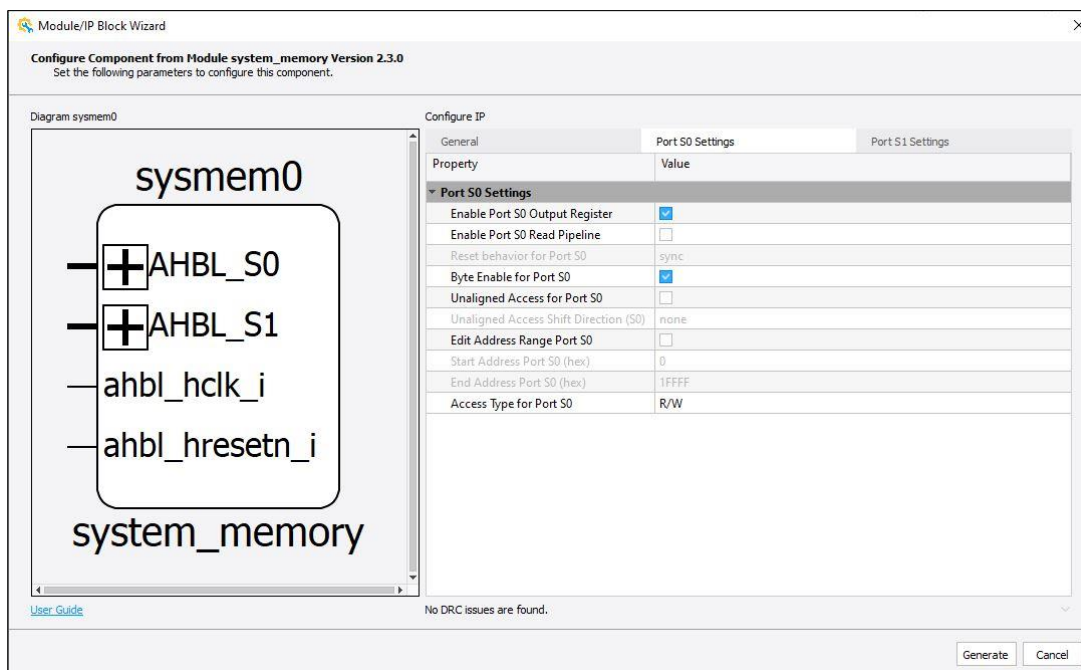


Figure 3.6. System Memory Port S0 Configuration

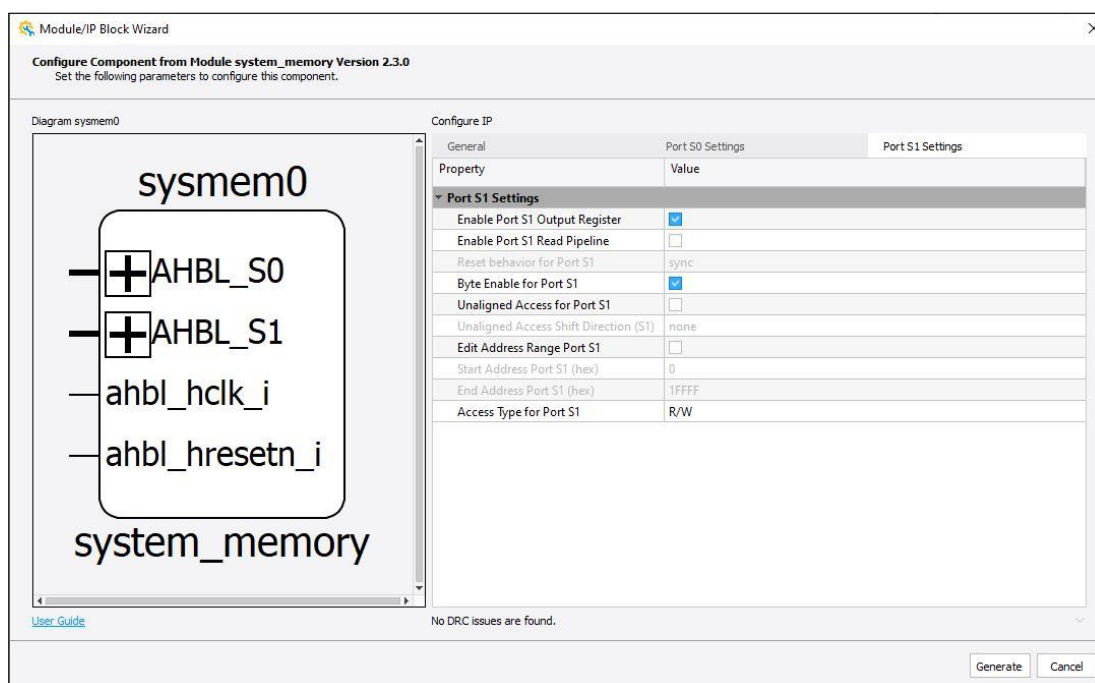


Figure 3.7. System Memory Port S1 Configuration

3.1.6. UART

The RISC-V MC processor can communicate with the UART and GPIO cores through its Advanced Peripheral Bus (APB) interface. Hence, the UART and GPIO cores are connected to the soft processor through the AHB-Lite to APB Bridge. Generally, the UART IP performs serial-to-parallel conversion on data characters received from a peripheral UART device. The IP performs parallel-to-serial conversion on data characters received from the host through an APB interface. In this reference design, the host is the RISC-V processor inside the FPGA. The UART IP configuration is shown in Figure 3.8.

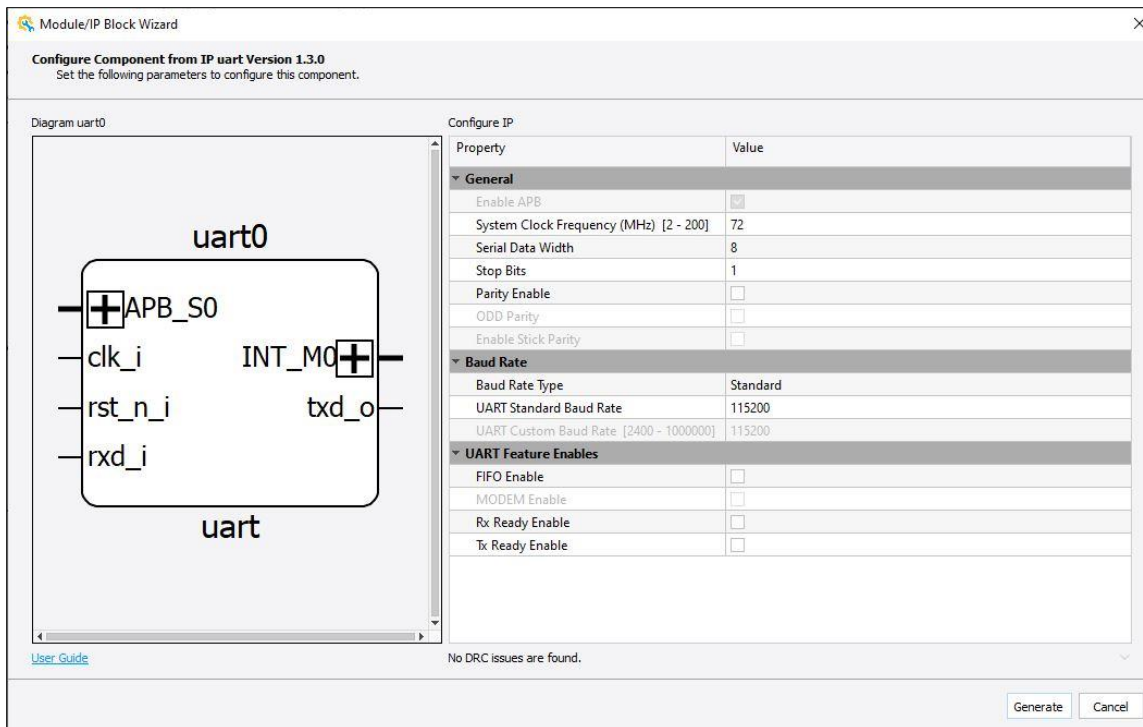


Figure 3.8. UART Configuration

3.1.7. GPIO

There is a 32-bit input GPIO, the hardware version GPIO, to monitor the FPGA design version for a particular release. The hardware version GPIO IP configuration is shown in Figure 3.9.

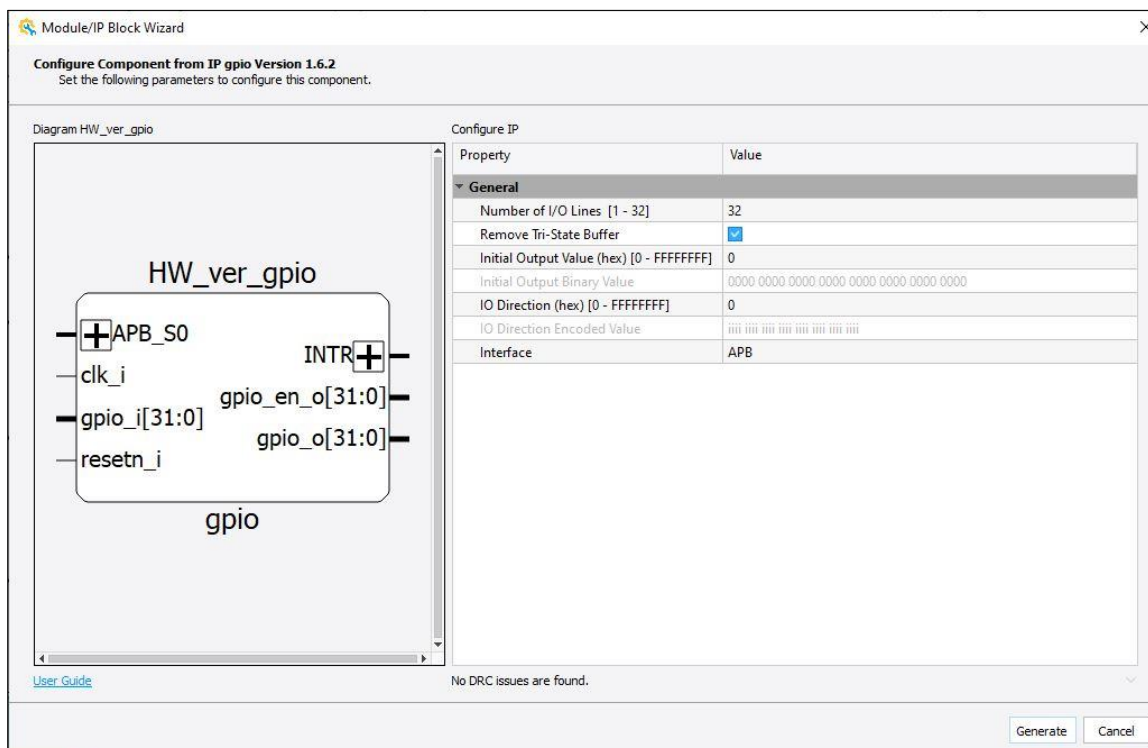


Figure 3.9. Hardware Version GPIO Configuration

Also, there is a 2-bit output GPIO, that is, the test pattern enable GPIO. This GPIO is used to select the data either from the camera sensor RX DPHY or MIPI RAW10 Test Pattern Generator, and Image Signal Processing stream or YUV422 Test Pattern Generator. The test pattern enable GPIO IP configuration shown in [Figure 3.10](#).

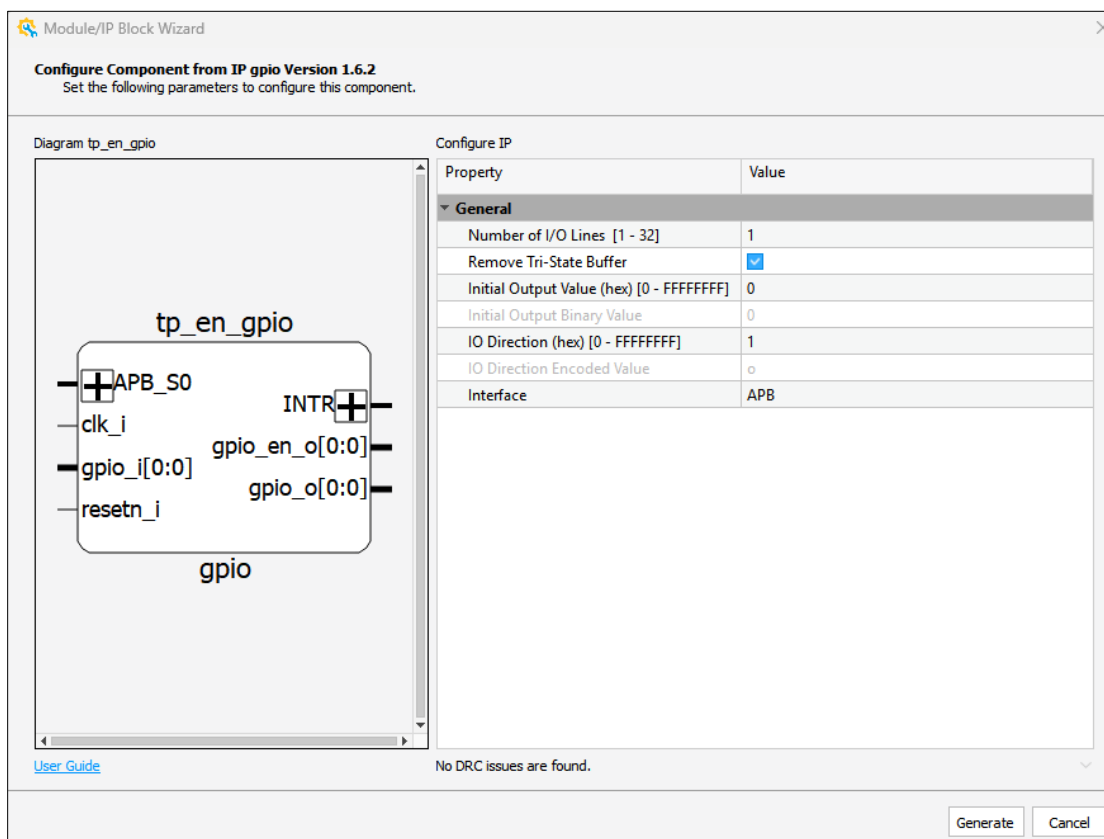


Figure 3.10. Test Pattern Enable GPIO Configuration

3.1.8. Lattice I2C Controller

This is a Lattice I2C Controller IP used to communicate with any I2C target device. With this IP, you can perform I2C write, I2C read, and I2C write followed by read operation on the I2C target device connected to it. In this design, the RISC-V MC processor configures the MIPI Camera Sensor register through this I2C Controller IP. The I2C Controller IP configuration is shown in [Figure 3.11](#).

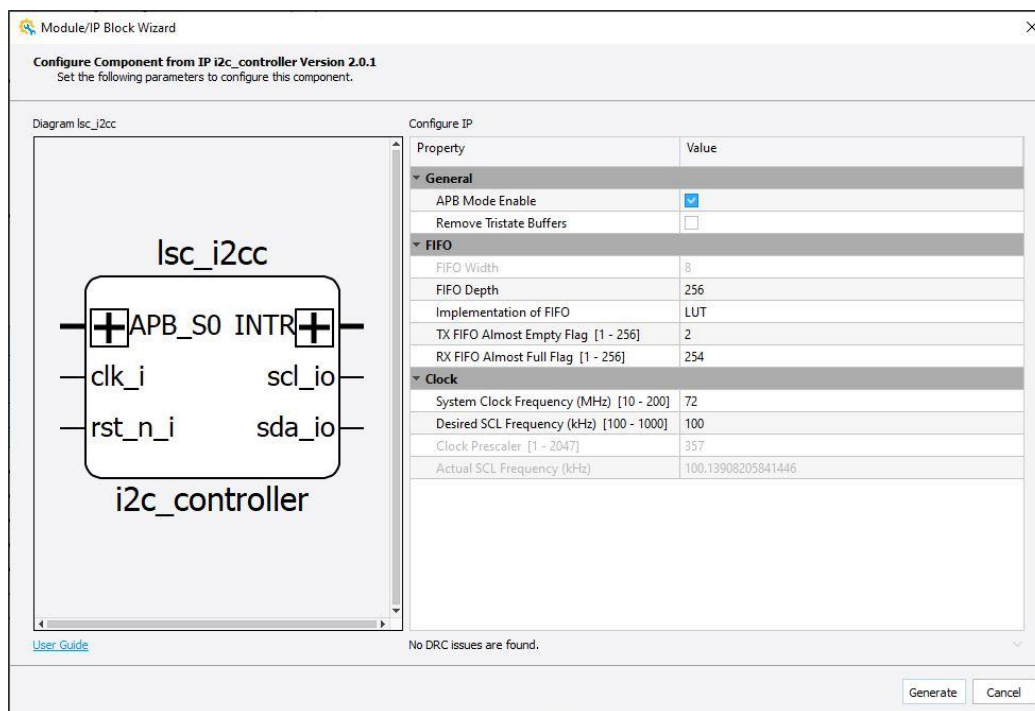


Figure 3.11. Lattice I2C Controller Configuration

3.1.9. AHB-Lite Interconnect

This IP provides the communicating interface between RISC-V soft processor and peripherals that support AHB-Lite interface, such as System Memory and AHB-Lite to LMMI Bridge. The AHB-Lite Interconnect module configuration is shown in the following figures, from Figure 3.12 to Figure 3.15.

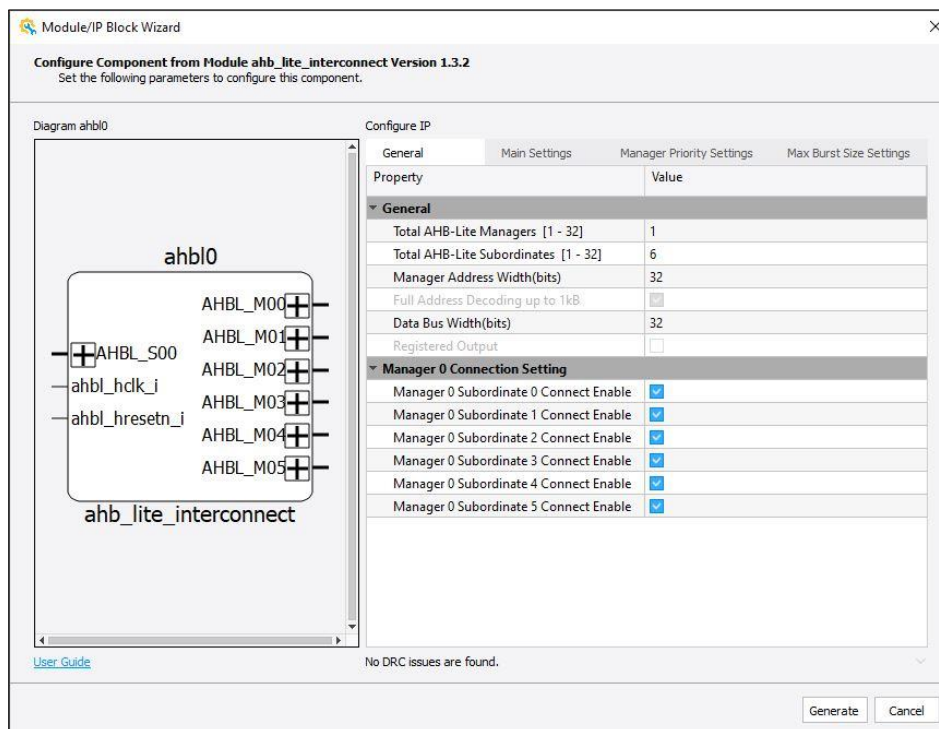


Figure 3.12. AHB-Lite Interconnect General Configuration

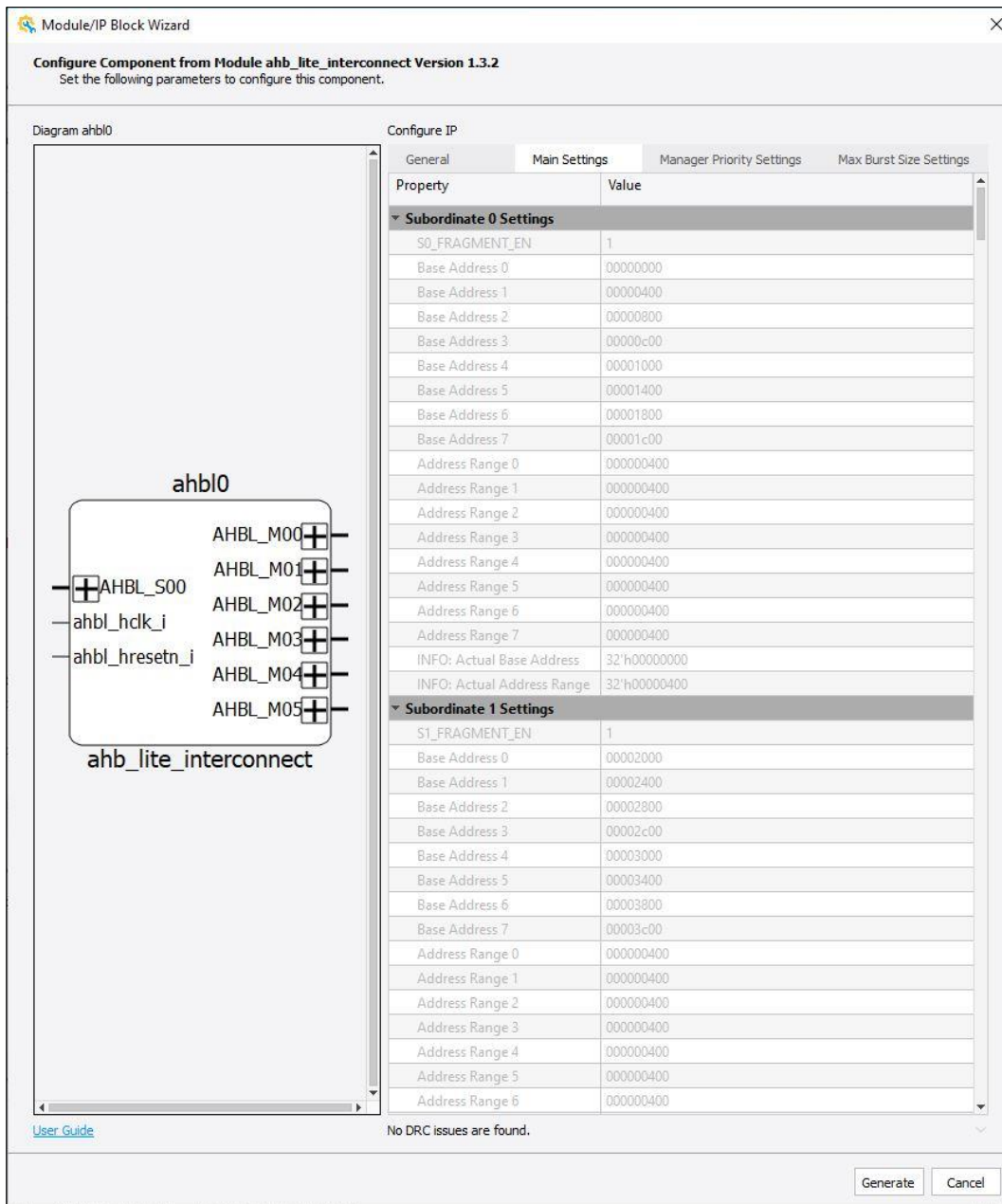


Figure 3.13. AHB-Lite Interconnect Main Configuration

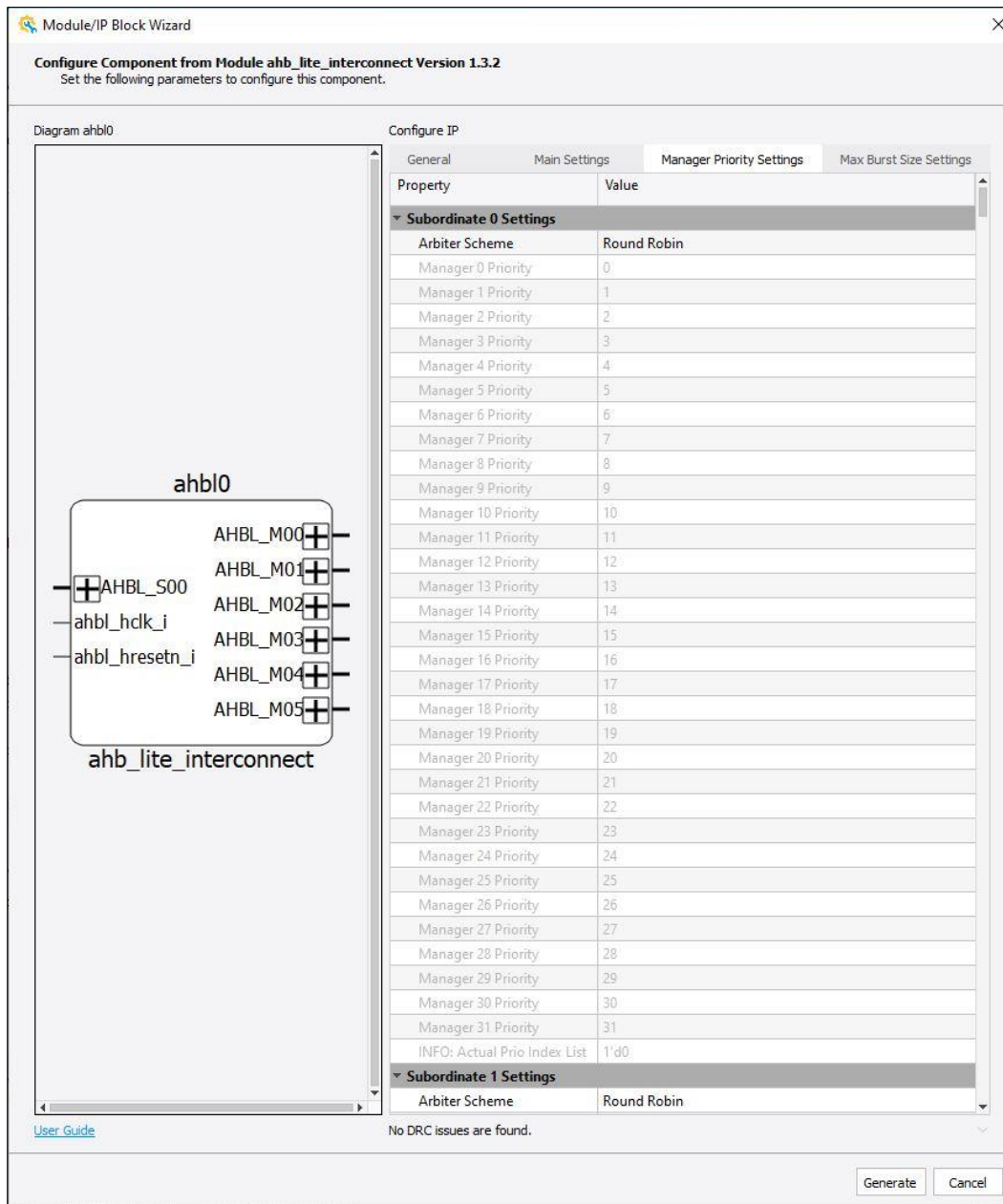


Figure 3.14. AHB-Lite Interconnect Manager Priority Setting

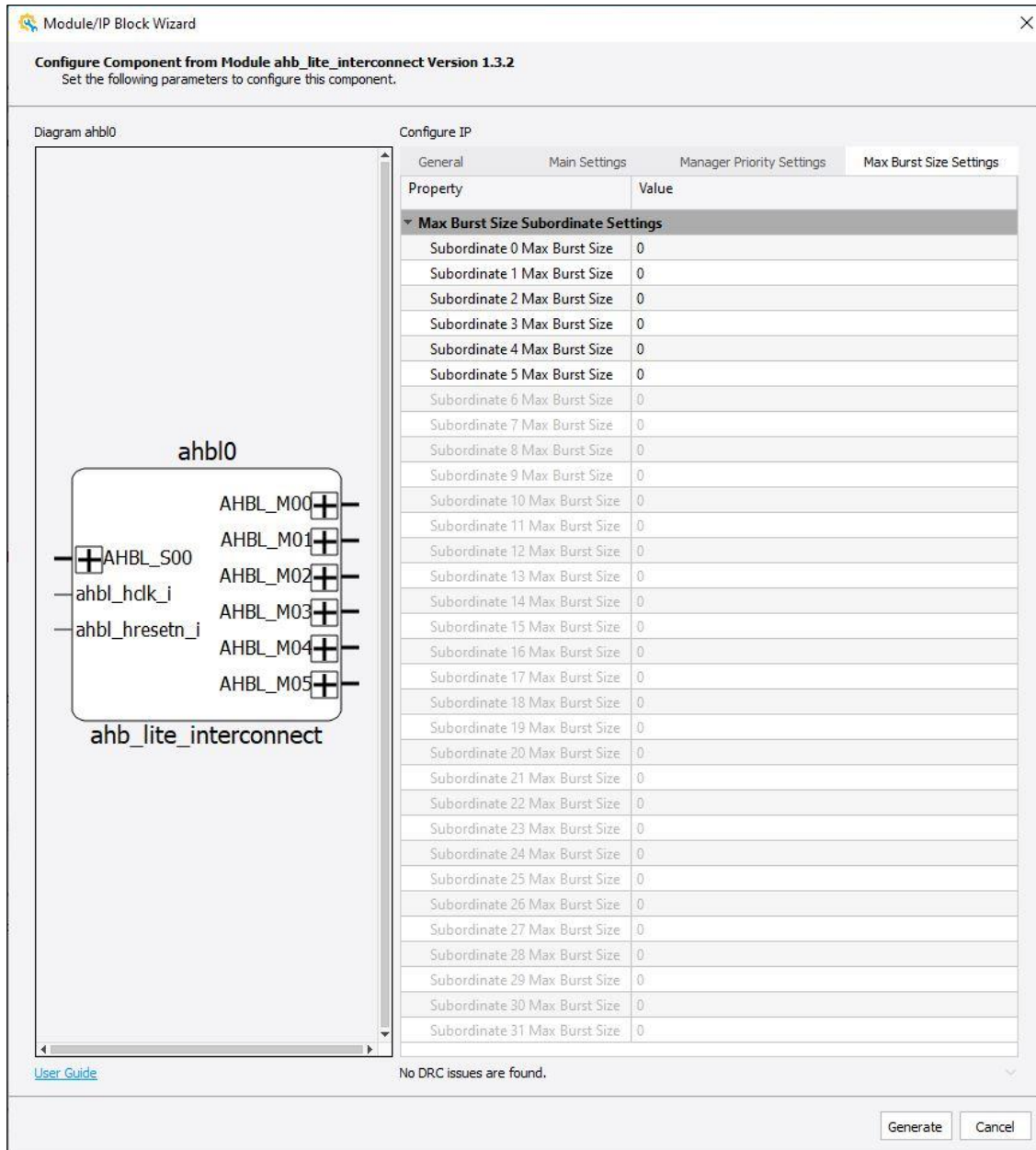


Figure 3.15. AHB-Lite Interconnect Max Burst Setting

3.1.10. APB Interconnect

This IP provides the communicating interface between the RISC-V MC soft processor and other slow peripherals that support the APB interface, such as UART and GPIO. APB interconnect configuration is shown in [Figure 3.16](#) and [Figure 3.17](#).

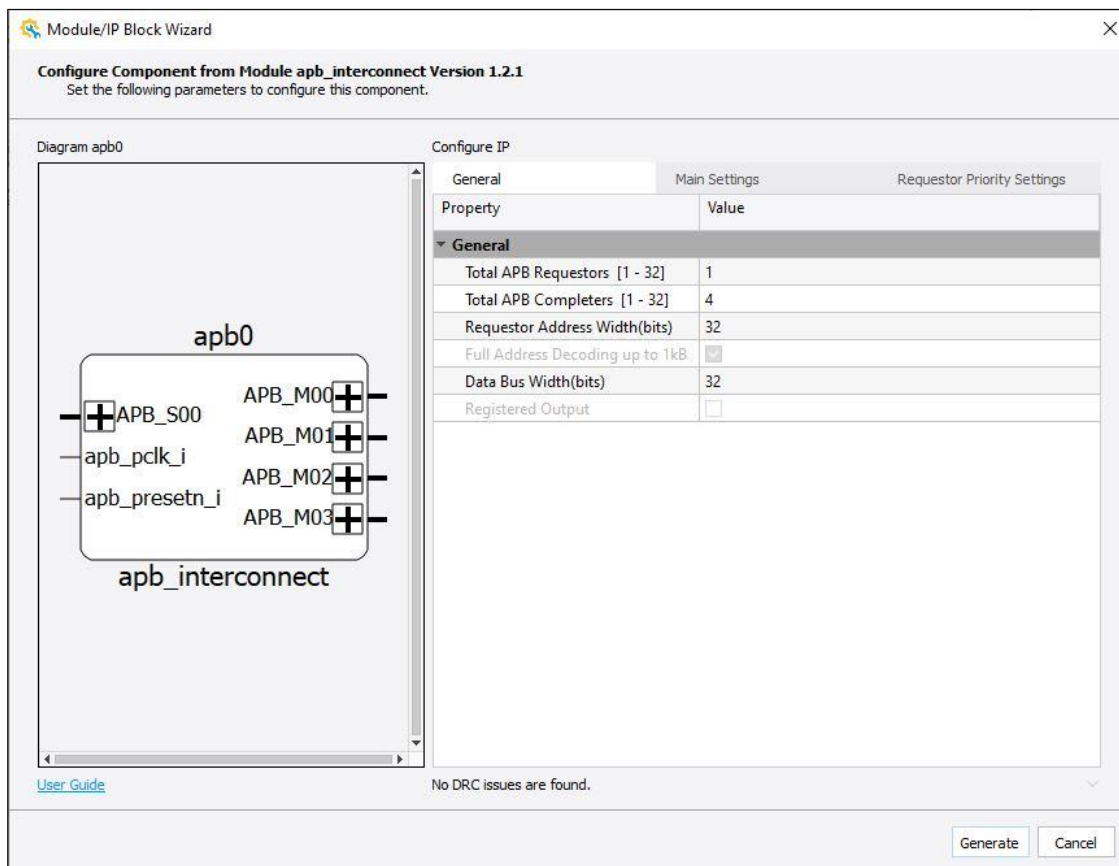


Figure 3.16. APB Interconnect General Configuration

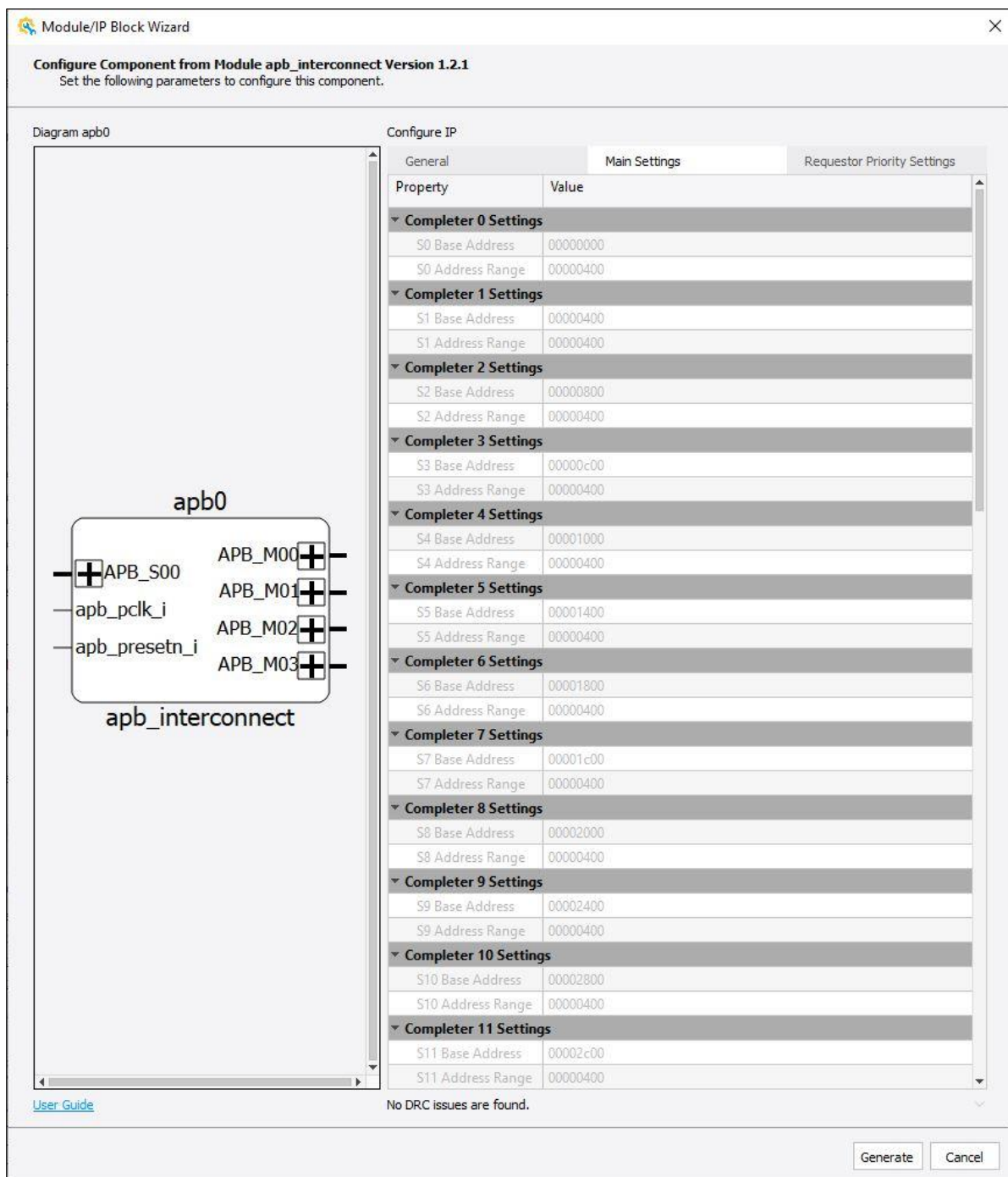


Figure 3.17. APB Interconnect Main Settings Configuration

3.1.11. APB-Lite to APB Converter

The AHB-Lite to APB Bridge provides an interface between the AHB-Lite manager and APB completer. Read and write transfers on the AHB are converted into equivalent transfers on the APB. The AHB-Lite to APB Converter IP configuration is shown in [Figure 3.18](#).

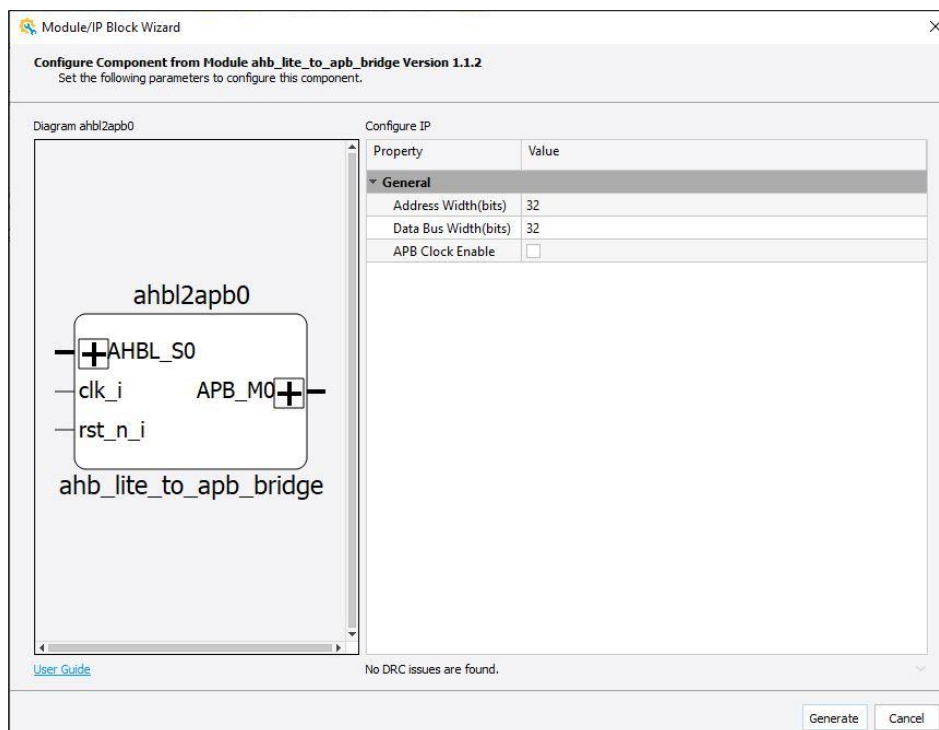


Figure 3.18. APB-Lite to APB Converter Configuration

3.1.12. RX DPHY

The MIPI camera sensor's data that you receive is in serial form. Hence, this IP converts those serial data into parallel by following the MIPI CSI-2 protocol. The RX DPHY IP configuration is shown in Figure 3.19, Figure 3.20, and Figure 3.21.

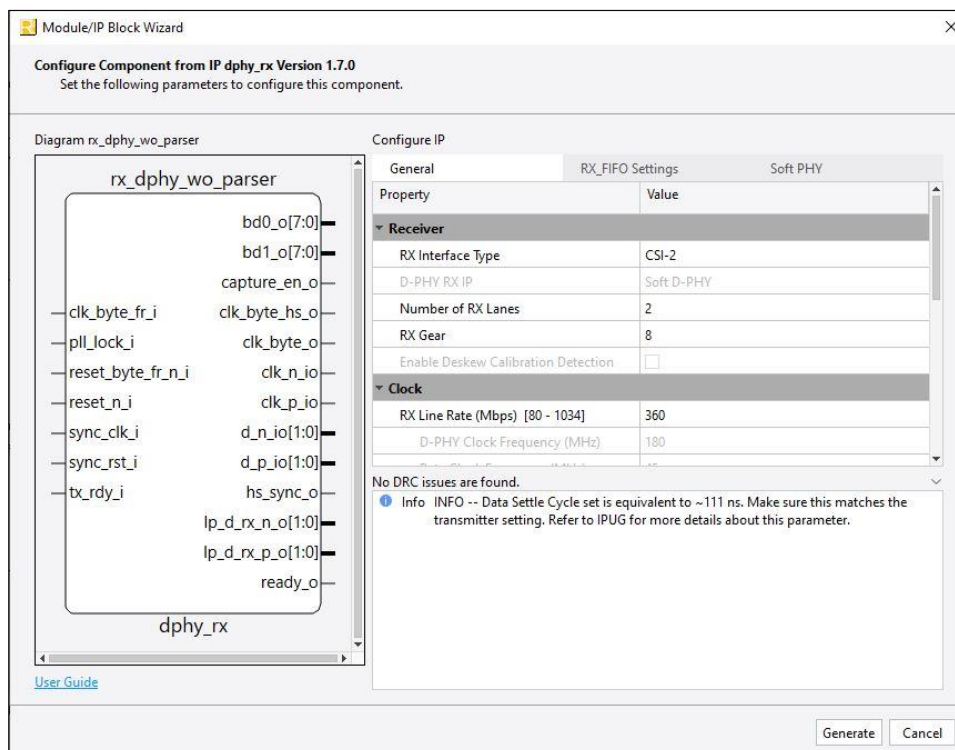


Figure 3.19. RX DPHY General Configuration

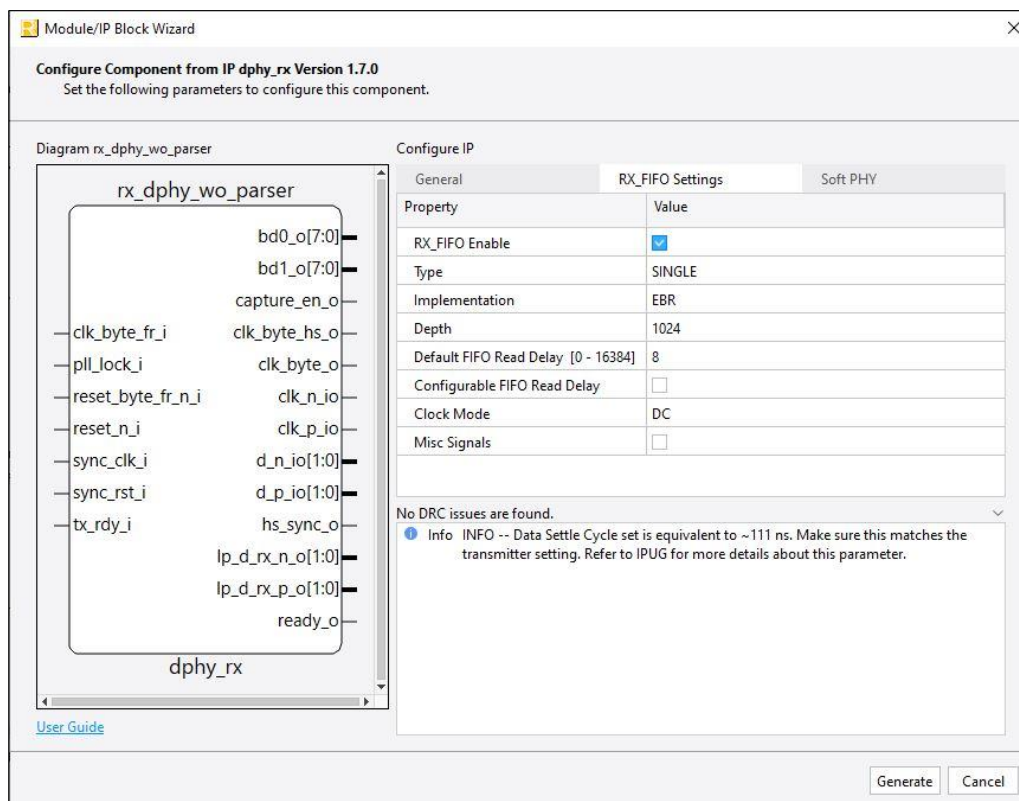


Figure 3.20. RX DPHY RX FIFO Configuration

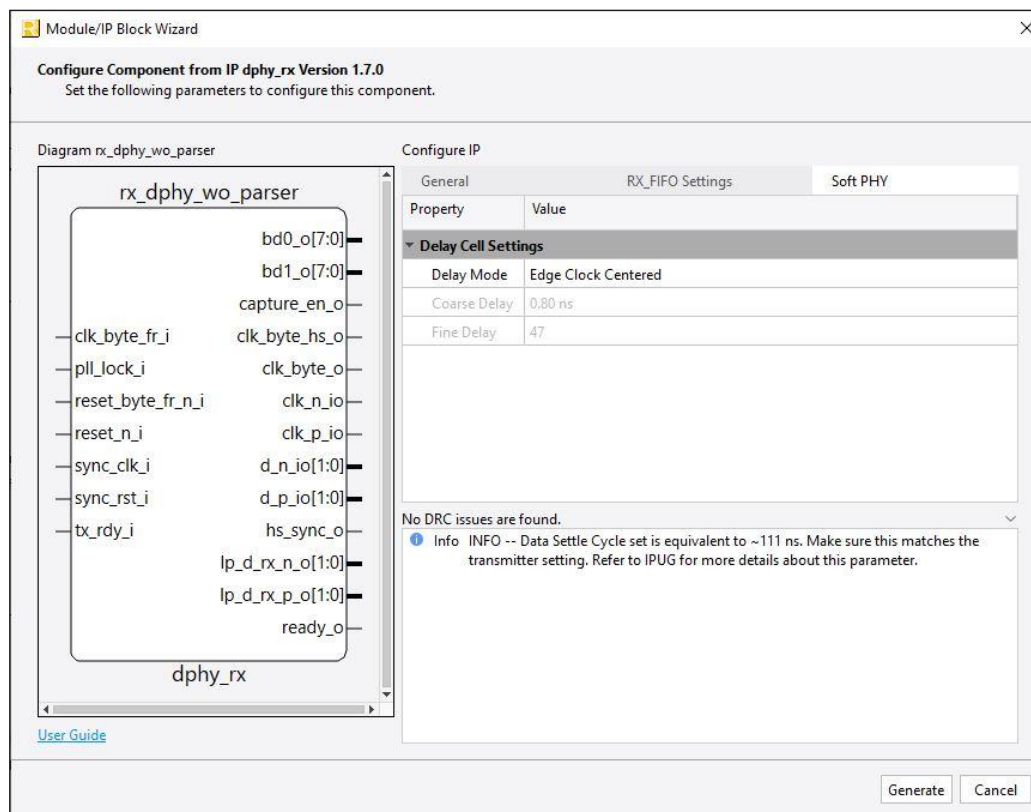


Figure 3.21. RX DPHY Soft PHY Configuration

3.1.13. MIPI Packet Decoder

This module decodes the data packet coming from the camera sensor. These packets may include data, control, synchronization, or configuration information. After decoding the packets, the RAW10 data is passed to the Byte to Pixel IP.

3.1.14. Byte to Pixel

This module converts the incoming RAW10 data into the pixel format. It receives raw bytes from a parser and rearranges them to output pixels. This module outputs 2-pixels in one clock cycle with valid signal. Each pixel is of 10-bit. The configuration of the Byte to Pixel IP is shown in Figure 3.22.

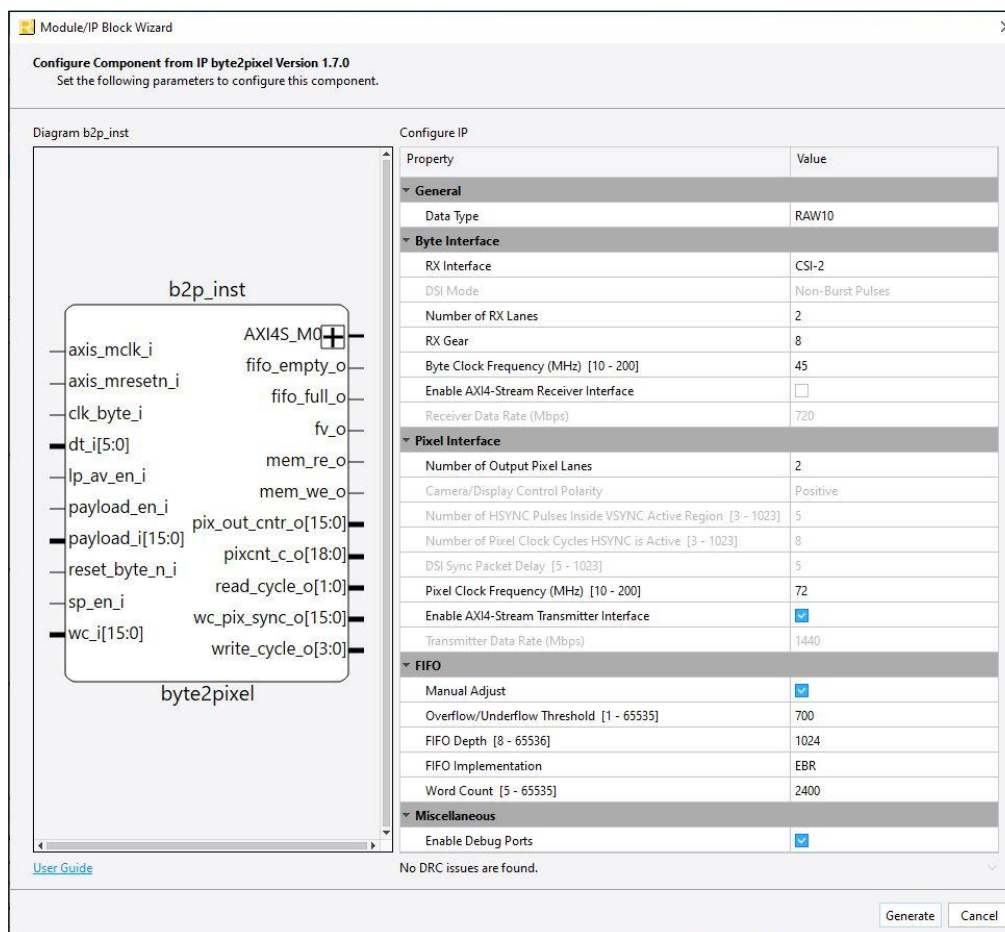


Figure 3.22. Byte to Pixel IP Configuration

3.1.15. B2P to AXI Stream Conversion

This module converts the pixel data according to the AXI Stream protocol to meet Lattice Debayer AXI Stream input requirement. Therefore, this module acts as a bridge between Lattice Byte to Pixel and Lattice Debayer IPs.

3.1.16. Debayer

The image sensor outputs pixel data in the Bayer format. In order to extract the R, G, and B components from Bayer data, we need the Debayer function. The Debayer converts raw10 image data into an RGB component. The Debayer IP configuration is shown in Figure 3.23 and Figure 3.24.

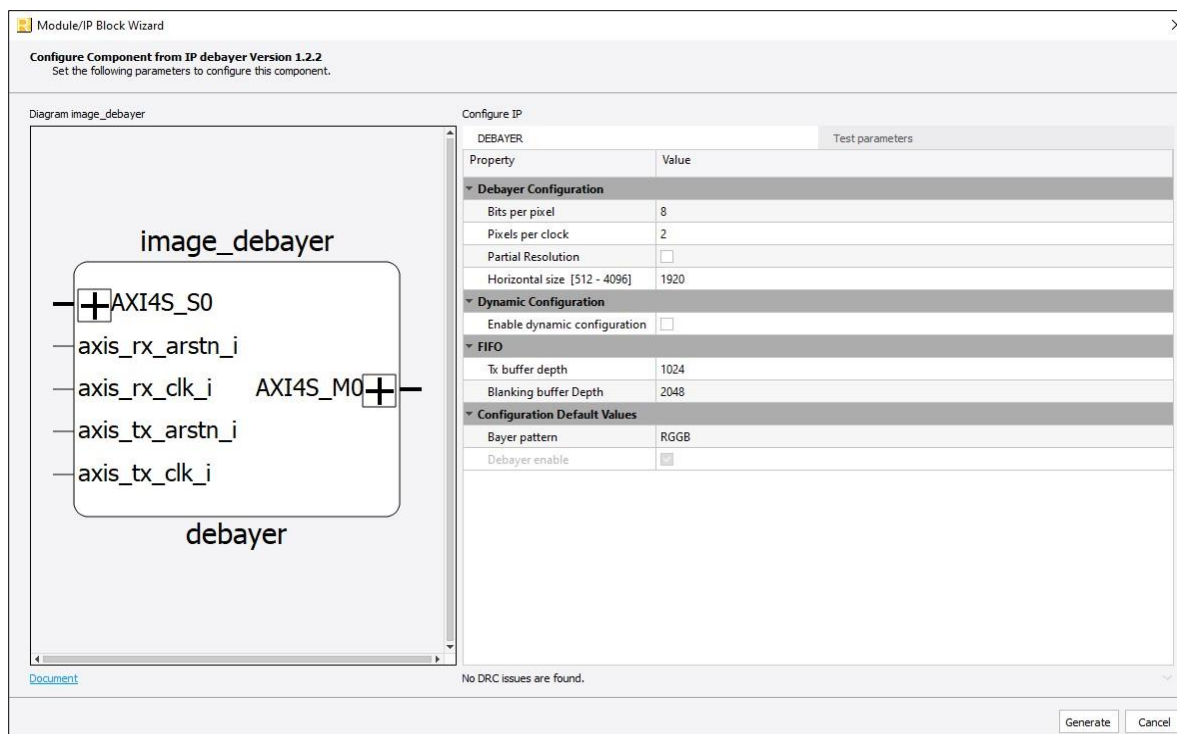


Figure 3.23. Debayer Configuration

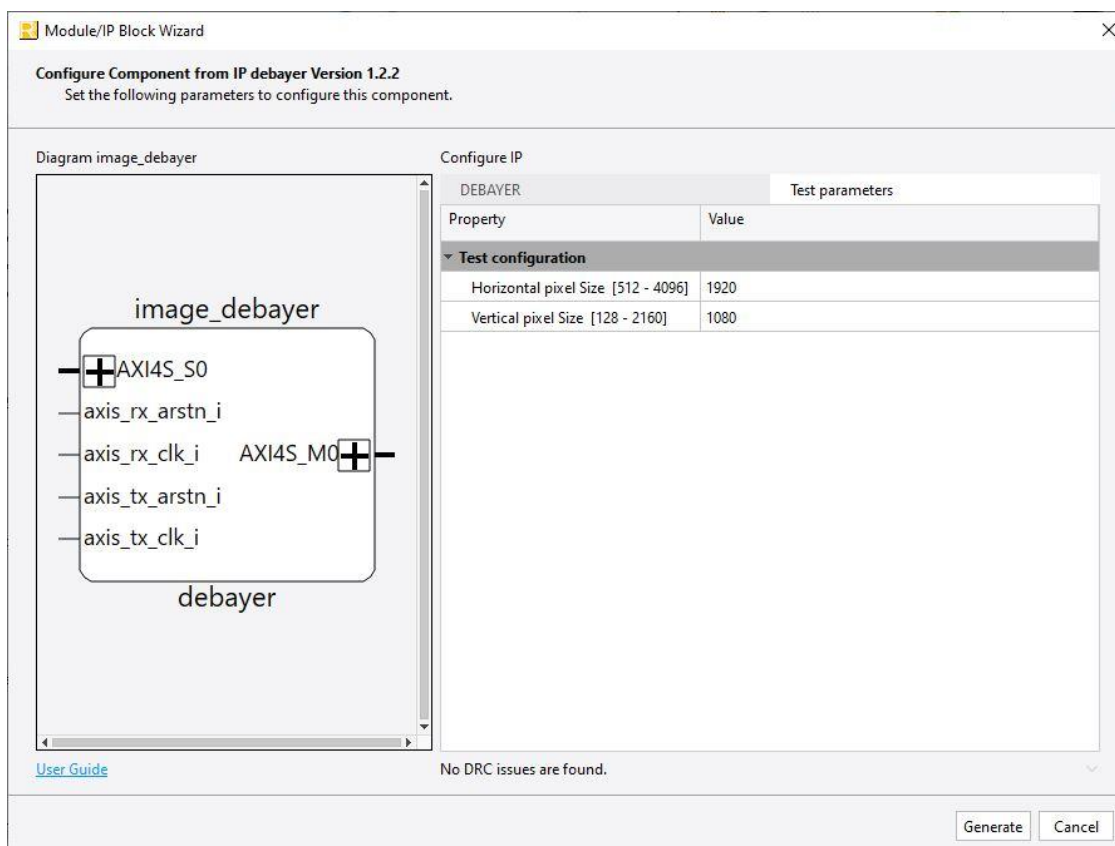


Figure 3.24. Debayer Test Parameters Configuration

3.1.17. Color Correction Matrix

To obtain the correct colors, the pixels need to be mapped from sensor RGB color space to standard RGB color space. This linear mapping of the color components is achieved by using a 3x3 matrix, called color correction matrix (CCM). The configuration of the CCM IP is shown in [Figure 3.25](#).

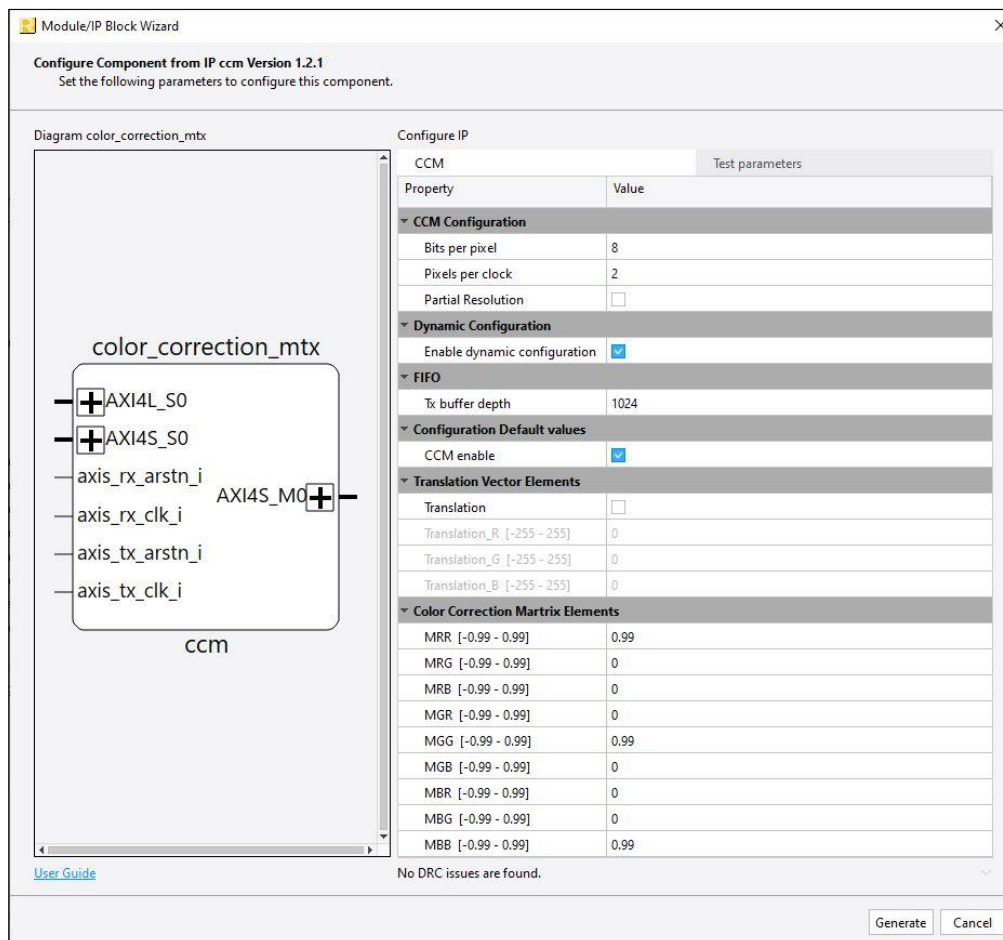


Figure 3.25. Color Correction Matrix Configuration

3.1.18. AXI Stream to Parallel Conversion

This module converts the AXI Stream protocol signal to parallel data. In this design, this module acts as a bridge between Lattice Debayer IP and Lattice CSC IP.

3.1.19. Color Space Converter

The Color Space Converter IP block converts RGB data into the YUV format and its configuration is shown in [Figure 3.26](#) and [Figure 3.27](#).

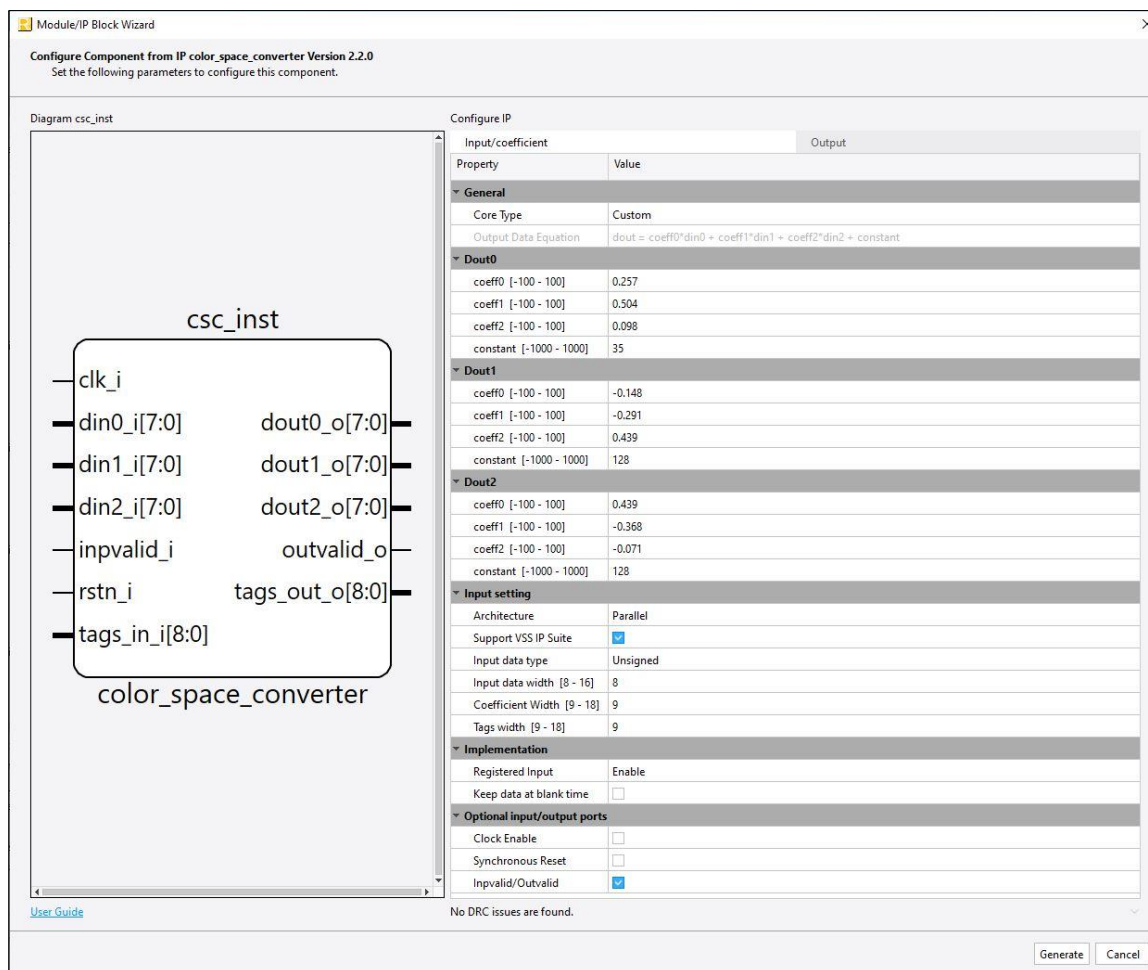


Figure 3.26. Color Space Converter Input/Coefficient Configuration

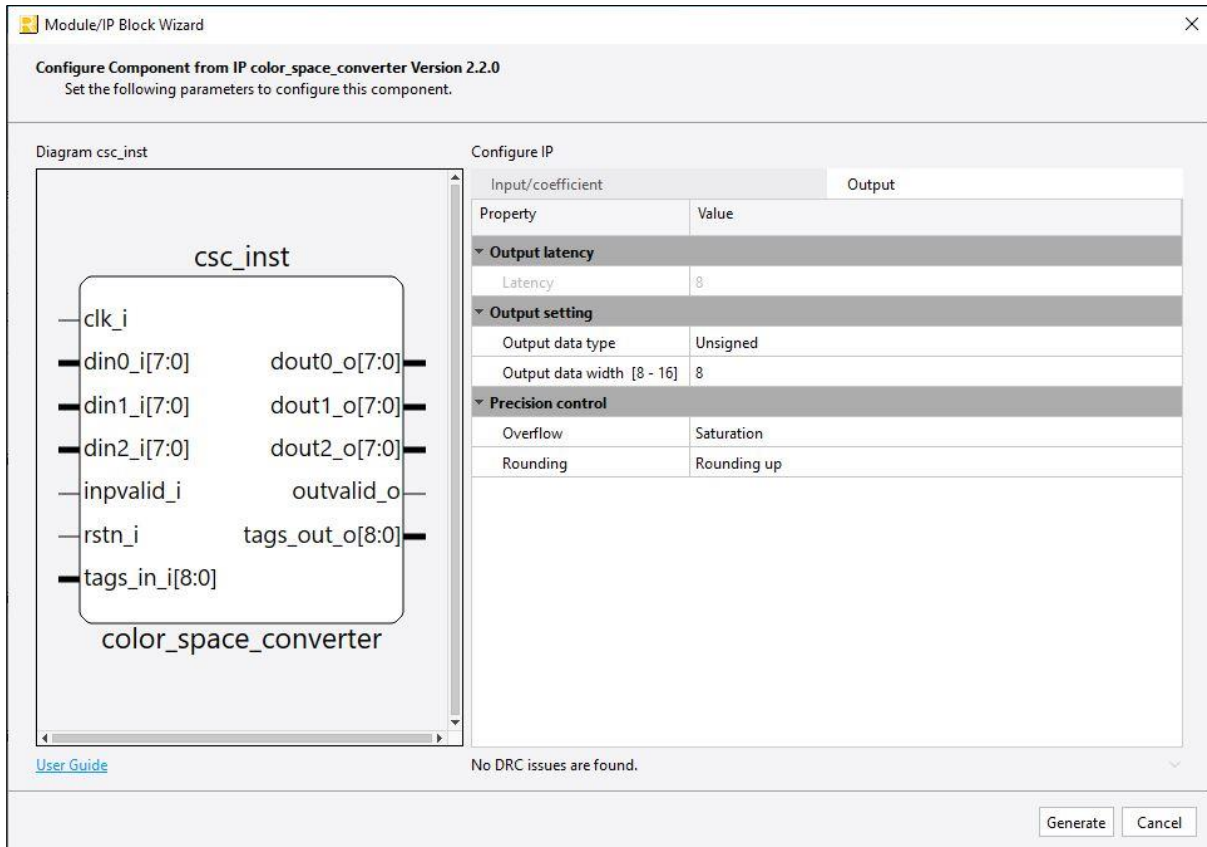


Figure 3.27. Color Space Converter Output Configuration

3.1.20. YUV422 to UVC Bridge

This module features an internal DC-FIFO called YUV422 FIFO, designed to store incoming YUV422 data along with associated data indicator codes. This write interface of this FIFO works on the pixel clock and the read interface works on the same clock as the IEBM. The YUV422 FIFO plays a critical role in managing and buffering video data, ensuring smooth data flow and processing.

3.1.21. IEBM In FIFO Interface

When the IEBM block grants permission to write the data, the IN FIFO controller module initiates the process. It reads the data from the YUV FIFO, where the video data is temporarily stored. Before writing this data into the IEBM buffers, the IN FIFO controller adds a two-byte UVC header at the start of each frame. Once the header is appended, the combined data, the header plus the YUV data, is written into the IEBM buffers.

3.1.22. In Endpoint Buffer Manager

The IEBM is responsible for managing the IN endpoint buffers for the USB23 controller. It handles the efficient and orderly flow of video data from the camera sensor or a test pattern generator. The data is written into different buffers sequentially through the FIFO interface. Once the data is available in these buffers, the USB23 controller reads it through the AXI interface. The entire process, from managing the FIFO writes to facilitating the AXI reads, is managed by this IEBM module. For more details about IEBM IP, refer to the [IN Endpoint Buffer Manager Architecture](#) section.

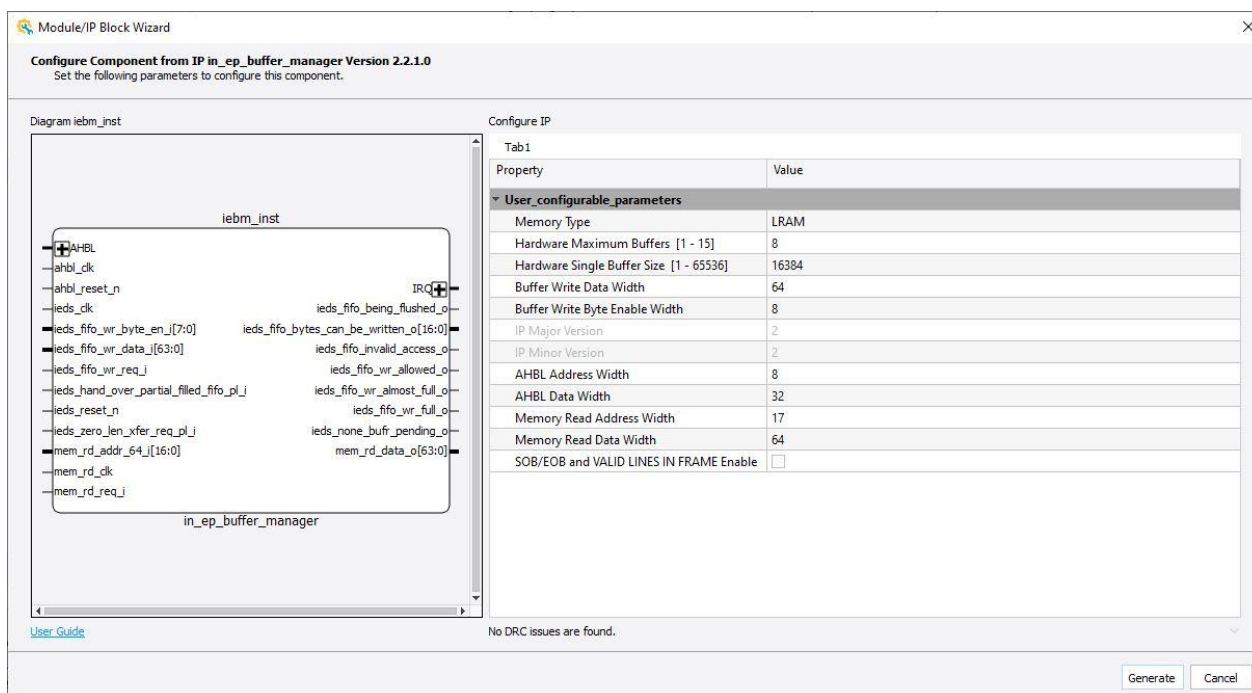


Figure 3.28. IEBM IP Configuration

3.1.23. AHB-Lite to Memory Bridge

This block helps RISC-V data manager to access the True Dual-Port (TDP) memory, which is dedicated for USB operations. This block converts the AHB-Lite subordinate signal into the native memory controller interface signals.

3.1.24. USB23 to AXI Bridge

The USB23 to AXI Bridge block helps the USB23 controller to access various memories. USB23 controller has an AXI manager interface while this block has an AXI subordinate interface. This converts AXI subordinate signals into native memory controller interface signals. This block also acts as a decoder to select appropriate memory-based input address.

3.1.25. RAW10 Test Pattern Generator

This block generates the RAW10 format color bar test pattern.

3.1.26. YUV422 Test Pattern Generator

This block generates the YUV422 solid color test pattern.

3.1.27. YUV422 Test Pattern In FIFO Bridge

This bridge receives the YUV422 solid color pattern from the Video Pattern Generator module, adds UVC header at the start of frame, and writes it into the IEBM buffer when the IEBM module allows for write.

3.2. Clocking Scheme

The CrosslinkU-NX USB Video Class Reference Design uses a PLL to generate the 72 MHz system clock, 72 MHz pixel clock for image signal processing, and 60 MHz USBPHY clock. All these clocks are generated from the 60 MHz crystal oscillator on LIFCL-33U Evaluation Board.

3.2.1. Clocking Overview

An overview of the CrosslinkU-NX USB Video Class Reference Design clocking scheme is shown in [Figure 3.29](#).

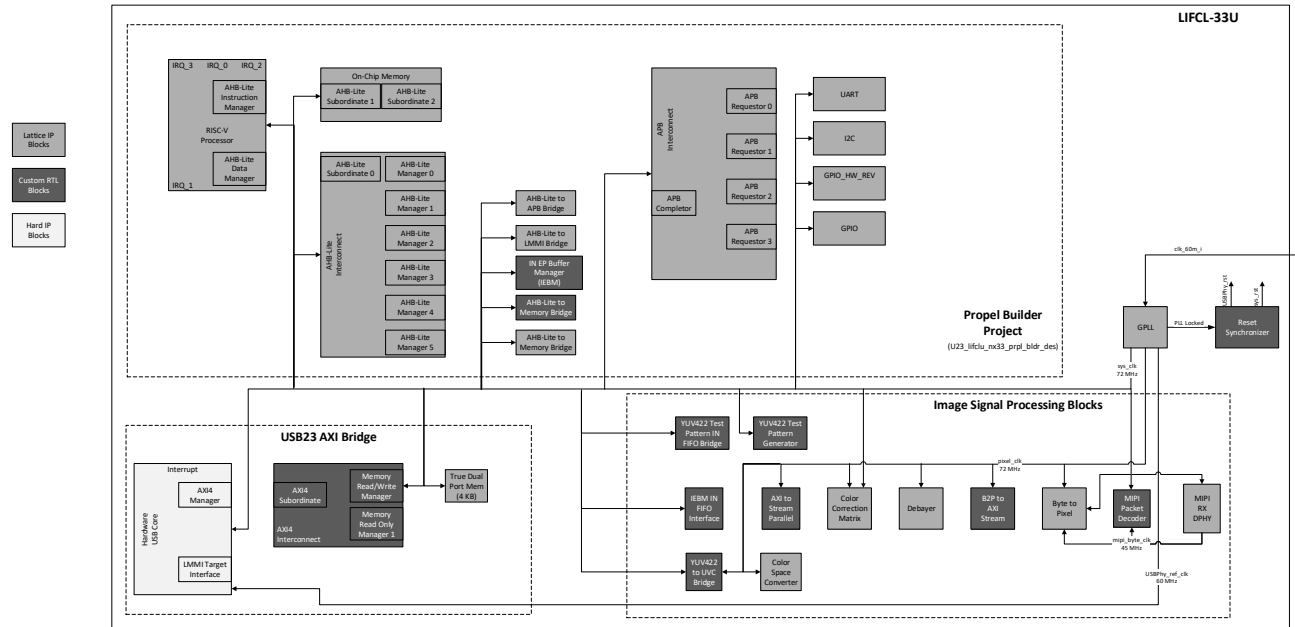


Figure 3.29. Reference Design Clock Domain Block Diagram

3.3. Reset Scheme

Synchronize resets are generated for each clock domain from the PLL lock signal and corresponding PLL clock outputs.

3.3.1. Reset Overview

An overview of the CrosslinkU-NX USB Video Class Reference Design clocking scheme is shown in Figure 3.30.

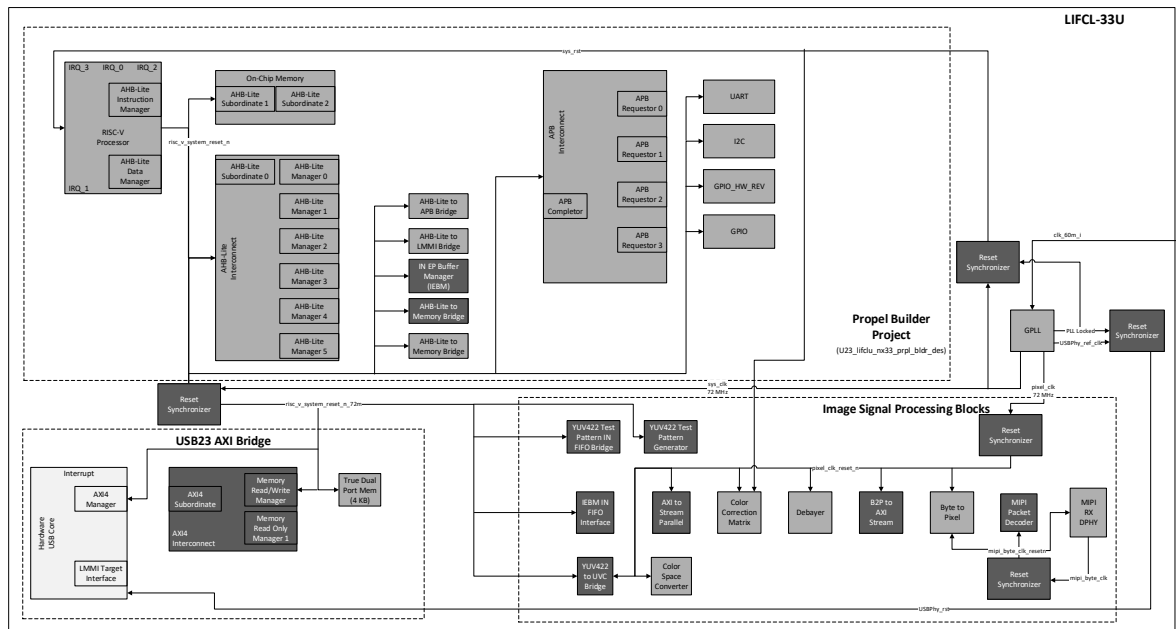


Figure 3.30. Reference Design Reset Scheme Diagram

4. IN Endpoint Buffer Manager Architecture

This section provides technical information about the IEBM IP.

4.1. IP Ports and Parameters

This section provides details of the IEBM IP ports and interface.

4.1.1. AHB-Lite Interface

This interface is used to access the registers of the IP. This AHB-Lite subordinate must be connected to the processor. This interface is synchronous to the AHB-Lite clock, ahbl_clk. Table 4.1 describes ports of the AHB-Lite subordinate interface.

Table 4.1. AHB-Lite Subordinate Interface

Signal Name	Width	Direction	Description
ahbl_clk	1	Input	AHB-Lite clock
ahbl_reset_n	1	Input	AHB-Lite reset, active low
ahbl_haddr_i	8	Input	AHB-Lite address
ahbl_hburst_i	3	Input	AHB-Lite burst type
ahbl_hmastlock_i	1	Input	AHB-Lite manager lock
ahbl_hprot_i	4	Input	AHB-Lite protection control
ahbl_hready_i	1	Input	AHB-Lite ready
ahbl_hsel_i	1	Input	AHB-Lite subordinate select
ahbl_hsize_i	3	Input	AHB-Lite size
ahbl_htrans_i	2	Input	AHB-Lite transfer type
ahbl_hwdata_i	32	Input	AHB-Lite write data
ahbl_hwrite_i	1	Input	AHB-Lite transfer direction. When HIGH, this signal indicates a write transfer. When LOW, it indicates a read transfer.
ahbl_hrdata_o	32	Output	AHB-Lite read data
ahbl_hreadyout_o	1	Output	AHB-Lite ready out, indicates the transfer has finished on the bus.
ahbl_hresp_o	1	Output	Valid lines in the One Frame register

4.1.2. AXI Interface

This subordinate interface is used to read the data from IP's data memory. This interface is synchronous to the AXI clock, axi_clk. Table 4.2 describes ports of the AXI subordinate interface.

Table 4.2. AXI Subordinate Interface

Signal Name	Width	Direction	Description
axi_clk	1	Input	AXI clock
axi_reset_n	1	Input	AXI reset, active low
axi_to_mem_araddr_i	8 ¹	Input	AXI Read address
axi_to_mem_arburst_i	2	Input	AXI Read burst type
axi_to_mem_arid_i	8	Input	AXI Read address ID
axi_to_mem_arlen_i	8	Input	AXI Read burst length
axi_to_mem_arsize_i	3	Input	AXI Read burst size
axi_to_mem_arvalid_i	1	Input	AXI Read address valid
axi_to_mem_arready_o	3	Output	AXI Read address ready
axi_to_mem_awaddr_i	8 ¹	Input	AXI Write address
axi_to_mem_awburst_i	2	Input	AXI Write burst type
axi_to_mem_awid_i	8	Input	AXI Write address ID

Signal Name	Width	Direction	Description
axi_to_mem_awlen_i	8	Input	AXI Write burst length
axi_to_mem_awsz_i	3	Input	AXI Write burst size
axi_to_mem_awvalid_i	1	Input	AXI Write address valid
axi_to_mem_awready_o	1	Output	AXI Write address ready
axi_to_mem_wdata_i	64	Input	AXI Write data
axi_to_mem_wlast_i	1	Input	AXI Write last
axi_to_mem_wstrb_i	8	Input	AXI Write strobe
axi_to_mem_wvalid_i	1	Input	AXI Write valid
axi_to_mem_wready_o	1	Output	AXI Write ready
axi_to_mem_bready_i	1	Input	AXI Write response ready
axi_to_mem_bid_o	8	Output	AXI Write response ID
axi_to_mem_bresp_o	2	Output	AXI Write response
axi_to_mem_bvalid_o	1	Output	AXI Write response valid
axi_to_mem_rready_i	1	Input	AXI Read ready
axi_to_mem_rdata_o	64	Output	AXI Read data
axi_to_mem_rid_o	8	Output	AXI Read ID
axi_to_mem_rlast_o	1	Output	AXI Read last
axi_to_mem_rresp_o	2	Output	AXI Read response
axi_to_mem_rvalid_o	1	Output	AXI Read valid

Note:

1. AXI Address width and data width are configurable. Based on the corresponding selection, this value changes. Depending on the FIFO data width selection, this value also changes. For example, if the selected FIFO data width is 16, this value is 2. If the selected FIFO data width is 32, this value is 4. If the selected FIFO data width is 64, this value is 8.

4.1.3. FIFO Interface

This interface is used to communicate with the external FIFO controller, efm. The external FIFO controller can read or write into the endpoint's buffer using this FIFO interface. This interface is synchronous to the external FIFO controller, clock, efm_clk. [Table 4.3](#) describes ports of the FIFO interface. Refer to the [FIFO Write Operation](#) section for more detail.

Table 4.3. FIFO Interface

Signal Name	Width	Direction	Description
ieds_clk	1	Input	External FIFO controller clock
ieds_fifo_wr_req_i	1	Input	Write request
ieds_fifo_wr_data_i	32 ¹	Input	Write data
ieds_fifo_wr_byte_en_i	4 ²	Input	Byte enable
ieds_hand_over_partial_filled_fifo_pl_i	1	Input	Partial transfer request pulse
ieds_zero_len_xfer_req_pl_i	1	Input	Zero length transfer request pulse
ieds_fifo_invalid_access_o	1	Output	Invalid FIFO access
ieds_fifo_wr_full_o	1	Output	FIFO is full.
ieds_fifo_wr_almost_full_o	1	Output	FIFO is almost full.
ieds_fifo_wr_allowed_o	1	Output	Write is allowed.
ieds_none_buf_pending_o	1	Output	No buffer is pending.
ieds_fifo_being_flushed_o	1	Output	FIFO is being flushed.
ieds_fifo_bytes_can_be_written_o	10	Output	Number of bytes that can be written

Notes:

1. FIFO data width is configurable. Based on the corresponding selection, this value changes.
2. Depending upon FIFO data width selection, this value changes. For example, if the selected FIFO data width is 16, this value is 2. If the selected FIFO data width is 32, this value is 4. If the selected FIFO data width is 64, this value is 8.

4.1.4. Miscellaneous Ports

Table 4.4 describes the miscellaneous ports of the IP.

Table 4.4. Miscellaneous Ports

Signal Name	Width	Direction	Description
timestamp_i	64	Input	Timestamp value
valid_lines_per_frame_i	13	Input	This indicates the valid lines in one frame.
first_buf_in_xfer_i	1	Input	First buffer in transfer indication
last_buf_in_xfer_i	3	Input	Last buffer in transfer indication

4.1.5. User Configurable Parameters

Table 4.5 provides the list of the user configurable parameters for the IP.

Table 4.5. User Configurable Parameters

Parameter	Description
IEBM_AXI4_ADDR_WIDTH_I	This parameter defines the address width of the AXI interface.
IEBM_AXI4_DATA_WIDTH_I	This parameter defines the read and write data width of AXI4 interface. Valid values: 32, 64, and 128
IEBM_AXI4_BYTE_EN_WIDTH_I	This parameter defines the byte enable width of the AXI interface. Calculated based on the IEBM_AXI4_DATA_WIDTH_I/8
IEBM_AHB_ADDR_WIDTH_I	This parameter defines the address width of the AHB-Lite interface.
IEBM_AHB_DATA_WIDTH_I	This parameter defines the read and write data width of AHB-Lite interface. Valid values: 32
IEBM_HW_MAX_BUFRS_I	This parameter defines the number of total buffers allocated for hardware. Valid values: 1 to 15
IEBM_HW_SINGLE_BUFR_SIZE_I	This parameter defines the In Endpoint Buffer Manager's single buffer depth in terms of bytes. Valid values: 1 to 4096
IEBM_BUFR_WR_DATA_WIDTH_I	This parameter defines the In Endpoint Buffer Manager's write data width in terms of bits. Valid values: 32, 64, 128
IEBM_BUFR_WR_BYTE_EN_WIDTH_I	This parameter defines the In Endpoint Buffer Manager's write data width. Valid values: 4, 8, 16
IEBM_IP_MAJOR_VER_I	This parameter defines the major version for the IP.
IEBM_IP_MINOR_VER_I	This parameter defines the minor version for the IP.

4.2. Register Offset Map

This section provides a register offset map for the IN Endpoint Buffer Manager IP.

Table 4.6. IEBM Register Offset Map

Register	Offset	Description
IP_VERSION	0x00	IP version register
SCRATCH	0x04	Scratch register
INT_EN	0x08	Interrupt enable register
INT_SRC	0x0C	Interrupt source register
BUFR_CNFG	0x10	Buffer configuration register
HW_PARAMS_INFO	0x14	Hardware parameters information register
CTRL	0x18	Control register
BUFR_TRACKER_INFO	0x1C	Buffer tracker information register

Register	Offset	Description
BUFR_AVAILABILITY_INFO	0x20	Buffer availability information register
BUFR_XCHNG_CTRL	0x24	Buffer exchange control register
SEL_BUFR_INFO_FOR_FW	0x28	Selected buffer information for firmware register
Reserved	0x2C	Reserved
TIMESTAMP_W0	0x30	Timestamp lower bytes register
TIMESTAMP_W1	0x34	Timestamp upper bytes register
VALID_LINES_IN_FRAME	0x38	Valid Lines in one frame register

4.3. Register Details

4.3.1. IP_VERSION

This register contains information about the IP version.

Table 4.7. IP_VERSION, Offset = 0x00

Bit	Access	Default Value	Description
31:24	RO	Varies upon IP release	IP major version
23:16	RO	Varies upon IP release	IP minor version
15:0	RO	0x0	Reserved

4.3.2. SCRATCH

This register can be used for testing and debugging purposes. When being read, the register returns the same value that is written to it.

Table 4.8. SCRATCH, Offset = 0x04

Bit	Access	Default Value	Description
31:0	RW	0x0	This field can be used for testing and debugging purposes. When being read, the register returns the same value that is written to it. Typically, this is used to check whether firmware is able to communicate with the IP properly or not during the initial design development phase.

4.3.3. INT_EN

This register indicates the interrupt enable which controls the reporting of interrupt to the software. It contains direct mapping to the INT_SRC register. When a bit is set and the corresponding interrupt is active.

Table 4.9. INT_EN, Offset = 0x08

Bit	Access	Default Value	Description
31:1	RO	0x0	Reserved
0	RW	0x0	Enable buffer available to firmware for processing interrupt. 1: Enable buffer available to firmware for processing interrupt. 0: Disable buffer available to firmware for processing interrupt.

4.3.4. INT_SRC

This register indicates the interrupt source, defining the events that determine the interrupt generation.

Table 4.10. INT_SRC, Offset = 0x0C

Bit	Access	Default Value	Description
31:1	RO	0x0	Reserved
0	RO	0x0	Buffer available to firmware for processing interrupts. This field indicates that there is at least one buffer available to firmware for further processing. This field remains high as long as no_of_bufers_pending_to_be_processed_by_fw is not zero.

4.3.5. BUFR_CNFG

This register allows firmware to configure buffer parameters according to different requirements. Firmware shall update this register upon steps such as initialization, configuration, or reconfiguration.

Warning: The firmware shall modify this register only when Configured Flag is zero. Failing to meet this requirement may cause unexpected behavior.

Table 4.11. BUFR_CNFG, Offset = 0x10

Bit	Access	Default Value	Description
31:22	RO	0x0	Reserved
21:17	RW	HW_MAX_BUFRS	FW_ALLOCATED_TOTAL_BUFRS This field defines the total number of buffers allocated by the firmware for this endpoint. This field must be less than or equal to HW_MAX_BUFRS, the hardware parameter which describes maximum buffers that can be used for this endpoint. This field shall not be zero. Typically, the firmware should use maximum buffers allowed by hardware design, that is, upon HW_MAX_BUFRS. The firmware may use a smaller number for debugging purposes.
16:0	RW	HW_SINGLE_BUFR_MAX_DEPTH	FW_ALLOCATED_SINGLE_BUFR_DEPTH This field defines single buffer depth in terms of bytes This field must be less than or equal to HW_SINGLE_BUFR_MAX_DEPTH, the hardware parameter that defines the maximum depth of a single buffer in terms of bytes. This field should not be zero. This field must be in multiples of the endpoint's maximum packet size. This ensures data packet to be sent on the USB bus is not split across multiple buffers. Examples: (1) Suppose HW_SINGLE_BUFR_MAX_DEPTH is 2048 and endpoint's maximum packet size is 512 bytes. In this case, valid values for this field are: 512, 1024, 1536, and 2048. (2) Suppose HW_SINGLE_BUFR_MAX_DEPTH is 512 and endpoint's maximum packet size is 64 bytes. In this case, valid values for this field are: 64, 128, 192, 256, 320, 384, 448, and 512. (3) Suppose HW_SINGLE_BUFR_MAX_DEPTH is 4096 and endpoint's maximum packet size is 700 bytes. In this case, valid values for this field are: 700, 1400, 2100, 2800, and 3500

4.3.6. HW_PARAMS_INFO

This register contains information about parameters set in the hardware design.

Table 4.12. HW_PARAMS_INFO, Offset = 0x14

Bit	Access	Default Value	Description
31:28	RO	Based on the hardware parameters	HW_MAX_BUFRS This field indicates the hardware parameter value that describes the maximum buffers that can be used for this endpoint.
27	RO	0x0	Reserved
26:10	RO	Based on the hardware parameters	HW_SINGLE_BUFR_MAX_DEPTH This field indicates the hardware parameter value that defines the maximum depth of a single buffer in terms of bytes. This field shall be in multiples of 8.
9:8	RO	0x0	Reserved
7:0	RO	Based on the hardware parameters	HW_FIFO_DATA_WIDTH This field indicates the hardware parameter value for the data width of FIFO interface in terms of bits.

4.3.7. CTRL

This register is used by the firmware to control the bridge IP's functionality. The firmware shall access this register for IP initialization, configuration, or reconfiguration steps.

Table 4.13. CTRL, Offset = 0x18

Bit	Access	Default Value	Description
31:2	RO	0x0	Reserved
1	RW	0x0	Flush Buffers flush_bufrs The firmware sets this field to request the bridge IP to flush corresponding buffers. Upon detecting this field going high, the IP takes the following actions: <ol style="list-style-type: none"> 1. Resets the IN Endpoint Buffer Manager Buffer Tracker and Firmware Buffer Tracker to 0. 2. Loads the number of buffers available to the IN Endpoint Buffer Manager with a value equal to the FW_ALLOCATED_TOTAL_BUFRS field. 3. Resets the number of buffers pending to be processed by the firmware to 0. 4. Resets other appropriate local variables. 5. Asserts one signal for 10 clock cycles of the FIFO clock to indicate to the IN Endpoint Buffer Manager about the re-initialization process. That block should reinitialize its local variables for a fresh start. 6. After this, the IP resets this field. The firmware can only write 1 to this field. Writing 0 has no effect. This field is automatically cleared by the IP.
0	RW	0x0	Configured This field indicates the firmware has configured or reconfigured the bridge IP and the IN endpoint data supplier block can now access the FIFO interface. The firmware shall configure or reconfigure the bridge IP every time the corresponding endpoint is configured or re-initialized. Typically, the endpoint is configured or re-initialized upon the following requests: SET configuration, SET interface, and Clear feature of Endpoint Halt.

4.3.8. BUFR_TRACKER_INFO

This register is used to track buffer pointers.

Table 4.14. BUFR_TRACKER_INFO, Offset = 0x1C

Bit	Access	Default Value	Description
31:4	RO	0x0	Reserved
3:0	RO	0x0	<p>Firmware Buffer Tracker fw_buftr_tracker</p> <p>This field indicates the buffer which is either being accessed or is to be accessed by firmware.</p> <p>This field is incremented by one once one buffer is processed by the firmware. This rolls over to 0 once it reaches the value that equals FW_ALLOCATED_TOTAL_BUFRS. In other words, the value of this field is in the range 0 to (FW_ALLOCATED_TOTAL_BUFRS-1).</p> <p>This is reset to 0 when the flush buffers bit of the control register is set.</p>

4.3.9. BUFR_AVAILABILITY_INFO

This register is used to get information about buffer availability.

Table 4.15. BUFR_AVAILABILITY_INFO, Offset = 0x20

Bit	Access	Default Value	Description
31:5	RO	0x0	Reserved
4:0	RO	0x0	<p>Number of buffers pending to be processed by firmware</p> <p>no_of_buftrs_pending_to_be_processed_by_fw</p> <p>This field indicates the number of buffers that are yet to be processed by the firmware. Upon initialization or re-initialization, this field is reset to 0.</p> <p>This is incremented by one once one buffer is handed over by the IN Endpoint Buffer Manager to the firmware.</p> <p>This is decremented by one once one buffer is processed by the firmware. For example, the corresponding transfer request block (TRB) is submitted to the USB23 IP.</p>

4.3.10. BUFR_XCHNG_CTRL

This register is used to exchange buffers between the IP and the firmware.

Table 4.16. BUFR_XCHNG_CTRL, Offset = 0x24

Bit	Access	Default Value	Description
31:2	RO	0x0	Reserved
1	RW	0x0	<p>Handover Buffer to IN Endpoint Buffer Manager handover_buftr_to_iebm</p> <p>The firmware sets this bit when it receives information from the USB23 IP that this buffer has been consumed. Upon detecting this, the number of buffers available to IN Endpoint Buffer Manager increments by 1.</p> <p>This bit is auto-cleared by the IP.</p>
1	RW	0x0	<p>Firmware processed one buffer fw_processed_one_buftr</p> <p>The firmware sets this bit when it has processed one buffer. In other words, when it has submitted corresponding TRB to the USB IP. Upon detecting this, the number of buffers pending to be processed by firmware field decrements by 1 and the Firmware Buffer Tracker field increments by 1.</p> <p>This bit is auto-cleared by the IP.</p>

4.3.11. SEL_BUFR_INFO_FOR_FW

This register indicates the selected buffer information to the firmware.

Table 4.17. SEL_BUFR_INFO_FOR_FW, Offset = 0x28

Bit	Access	Default Value	Description
31:23	RO	0x0	Reserved
22	RO	0x0	First buffer in transfer indicator The firmware knows whether the handed over buffer from the external controller is the first buffer in the transfer or not. 0: The selected buffer is not the first buffer in transfer. 1: This is the first buffer in the transfer.
21	RO	0x0	Last buffer in transfer indicator Firmware knows whether the handed over buffer from the external controller is the last buffer in the transfer or not. 0: The selected buffer is not the last buffer in transfer. 1: This is the last buffer in the transfer.
20:4	RO	0x0	Buffer size
3:0	RO	0x0	Selected buffer number, same value as Firmware Buffer Tracker The firmware can use this to find offset. $\text{BufFr_Offset} = \text{MEMORY_BASE_ADDRESS} + (\text{Selected buffer number} * \text{HW_ALLOCATED_SINGLE_BUFR_DEPTH})$ For Example: The selected buffer number or Firmware Buffer Tracker is 2. HW_ALLOCATED_SINGLE_BUFR_DEPTH is 4096. MEMORY_BASE_ADDRESS is 0. BufFr_Offset = 8192.

4.3.12. TIMESTAMP

The use case of this register is to know the current time of the ongoing transfer operation. The timestamp register is a 64-bit register. Hence, two separate 32-bit registers are used for the timestamp register. This is in terms of clock cycles. The firmware reads this register and calculates the current time based on the operating clock frequency.

- **TIMESTAMP_W0** – Timestamp lower word register
- **TIMESTAMP_W1** – Timestamp upper word register

See the example below.

If this register shows the value of 1055275, which is 0x00101A2B in hexadecimal, with **TIMESTAMP_W0** = 0x1A2B and **TIMESTAMP_W1** = 0x0010, and the operating clock frequency is 60 MHz, giving a clock period of 11.11 ns, then the current time can be calculated as follows:

$$\text{TIMESTAMP} * (\text{clock period}) = 1055275 * 11.11 \text{ ns} = 11.724 \text{ ms.}$$

4.3.12.1. TIMESTAMP_W0

Timestamp lower word register, 32 bits.

Table 4.18. TIMESTAMP_W0, Offset = 0x30

Bit	Access	Default Value	Description
31:0	RO	0x0	Timestamp_w0 This indicates the lower 32 bits of information for the current timestamp value.

4.3.12.2. TIMESTAMP_W1

Timestamp upper word register, 32 bits.

Table 4.19. TIMESTAMP_W0, Offset = 0x34

Bit	Access	Default Value	Description
31:0	RO	0x0	Timestamp_w1 This indicates the upper 32 bits of information for the current timestamp value.

4.3.13. VALID_LINES_IN_FRAME

This register provides information regarding the total number of lines that have been received and written into the buffers.

Table 4.20. BUFR_XCHNG_CTRL (Offset = 0x24)

Bit	Access	Default Value	Description
31:13	RO	0x0	Reserved
12:0	RO	0x0	active_lines_in_frame This indicates the total number of lines that have been received and written into the buffers.

4.4. IN Endpoint Buffer Management Architecture

The buffer management architecture is shown in Figure 4.1 below.

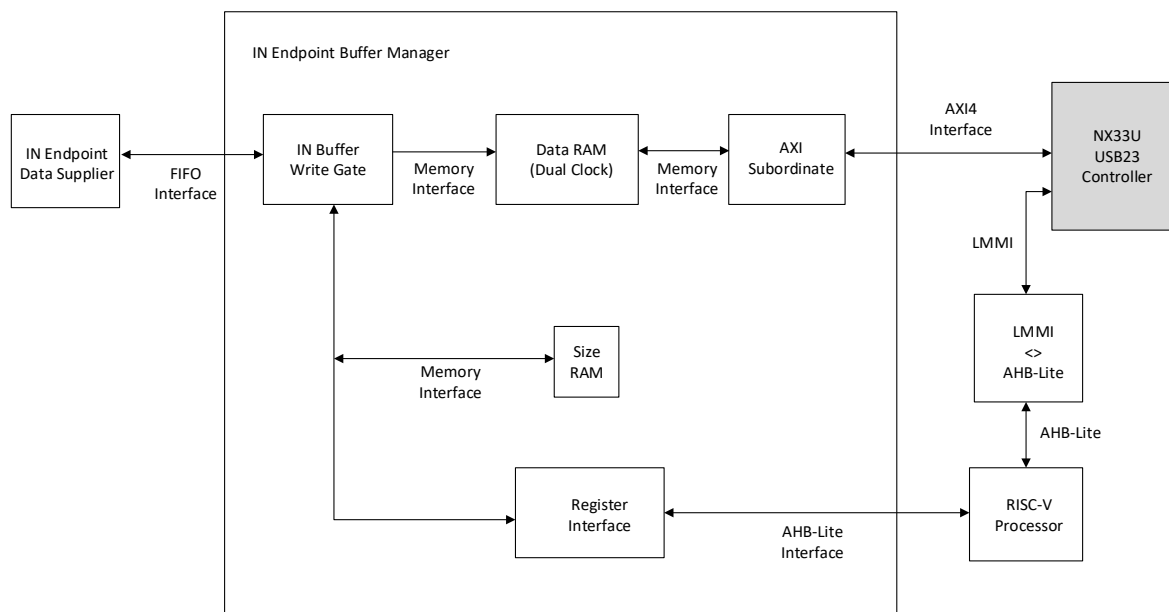


Figure 4.1. IN Endpoint Buffer Management Block Diagram

4.5. Buffer Read and Write Management Flow

This section describes how one buffer is written into the data RAM and read from data RAM. Buffer data is written by the IN Endpoint Buffer Manager and it is read back from data RAM by the CrosslinkU-NX USB23 controller.

4.5.1. Buffer Write Flow

The buffer write operation contains the following steps:

1. The IN FIFO write gate module checks whether the buffer is available or not. When you examine the write allowed signal, `ieds_fifo_wr_allowed_o`, if it is high, it indicates that the access is granted to the IN Endpoint Buffer Manager.
2. Once access is granted by the IN FIFO write access gate module, IN Endpoint Buffer Manager stores the data into the IN Data RAM until one buffer becomes full.
3. When one buffer becomes full, corresponding buffer length information is stored in the Size RAM. Also, the corresponding buffer-related information is updated in the registers.
4. Upon completion, the interrupt, Buffer Available to firmware for processing, is generated.
5. Once the interrupt is generated, jump to step 1.

The flow chart below shows how one buffer is written into the data RAM (Figure 4.2).

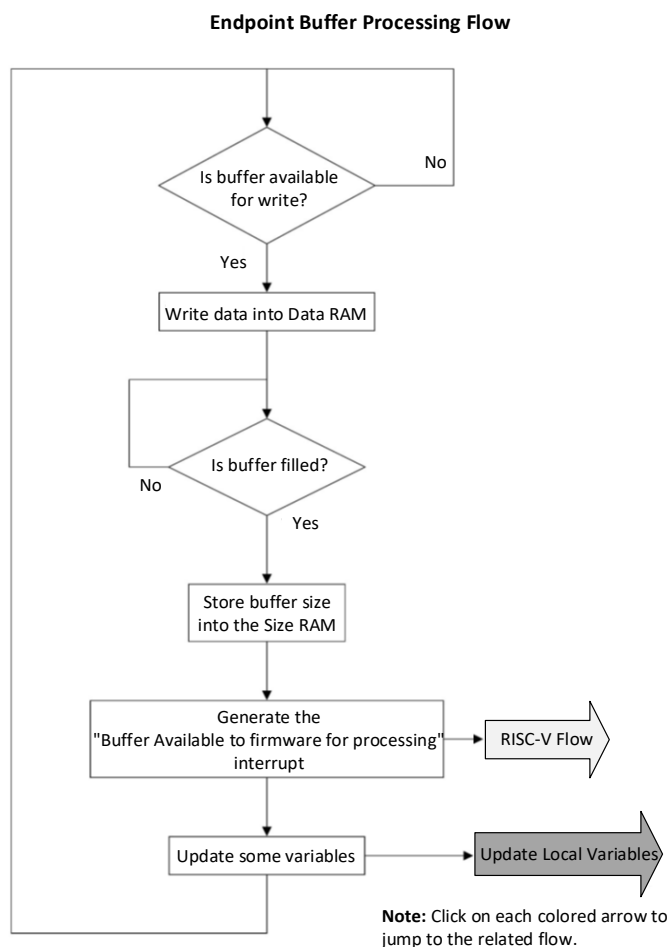


Figure 4.2. Buffer Write Operation Flow

4.5.2. Buffer Read Flow

Once one buffer is written into the Data RAM, the corresponding interrupt and buffer information is updated in the register through the register interface module. Below are the steps for the RISC-V firmware to process one buffer.

1. The Buffer Available to firmware for processing interrupt is generated. This interrupt remains high until `no_of_bufs_pending_to_be_processed_by_fw` is not zero.
2. The firmware reads the `no_of_bufs_pending_to_be_processed_by_fw` field from the Buffer Availability Information register and stores it into the local variable called `bufr_avail_for_fw_processing`.
3. The firmware checks whether the `bufr_avail_for_fw_processing` is zero or not. If it is zero, it means the whole TRB of the buffer has been initiated by the firmware. Go to step 1 and wait for an interrupt.

4. If the bufr_avail_for_fw_processing variable is not zero, that means the firmware needs to prepare the TRB.
5. For TRB, the firmware needs two fields. The first is the TRB starting address that is decided upon the buffer number. The second is the TRB size, which is the buffer size field of the SEL_BUFR_INFO_FOR_FW register. Hence, Read the SEL_BUFR_INFO_FOR_FW register to know the buffer size and buffer number. With this information, the firmware needs to prepare the TRB for the selected buffer.
6. Set the Firmware Processed one buffer bit in the Buffer Exchange Control register.
7. Decrement the local variable bufr_avail_for_fw_processing by 1 and jump back to step 3.
8. In parallel to this, Once the data packet is sent from the USB23 controller to USB HOST, the host sends the ACK in response to the data packet being sent. After that, the USB23 controller generates an xfer complete event. Based on that event, the firmware should set the handover_bufr_to_iebm bit.

The flow chart below shows how one buffer is read from the data RAM (Figure 4.3).

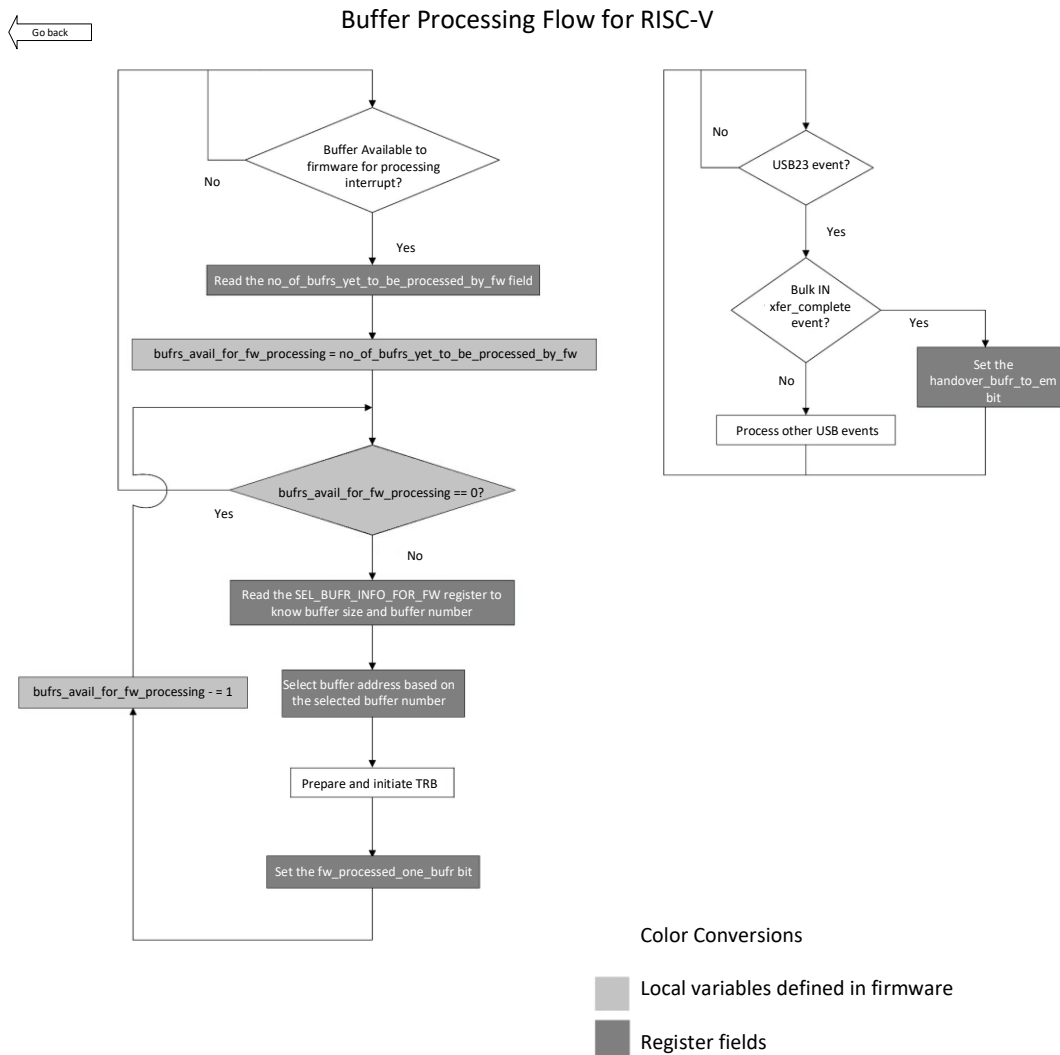


Figure 4.3. Buffer Read Operation Flow

4.6. Core Operation

This section describes how firmware can configure and reconfigure buffers.

4.6.1. IP Core Configuration or Reconfiguration

To configure or reconfigure the IP:

1. Disable interrupt generation by writing 0 to the INT_EN register.
2. Reset the Configured bit in the CTRL register.
3. Load the Firmware Buffer Configuration register with appropriate values.
4. Set the Flush buffers bit in the CTRL register.
5. Poll until the Flush buffers bit in the CTRL register is reset by the IP.
6. If the firmware plans to use the interrupt mode, enable interrupt generation by writing appropriate value to the INT_EN register.
7. Set the Configured bit in the CTRL register.

4.6.2. FIFO Write Operation

This section describes how In Endpoint Data Source (IEDS) performs FIFO write operation. [Table 4.21](#) describes ports that are used during FIFO write operation.

Table 4.21. Input Ports Related to FIFO Write Operation

Port	Direction	Description
ieds_fifo_wr_req_i	Input	<p>Write Request Assert this signal to request write operation. Do not assert this signal if any of following conditions is true:</p> <ul style="list-style-type: none"> • FIFO is being flushed – ieds_fifo_being_flushed_o is high. • FIFO is full – ieds_fifo_wr_full_o is high. • FIFO write operation is not allowed – ieds_fifo_wr_allowed_o is low.
ieds_fifo_wr_data_i	Input	<p>Write Data It holds the data to be written in the FIFO when the write request is asserted.</p>
ieds_fifo_wr_byte_en_i	Input	<p>Write Byte Enable It holds the byte enable bits when the write request is asserted. Make sure the following conditions are met:</p> <ul style="list-style-type: none"> • This is intended to be used only during partial transfer. • In all other cases, all bits in this signal must be kept asserted when the write request is high. <p>Note: During partial transfer, if the number of bytes to be written is less than the data width of the bus, all bytes must be adjacent and start from the least significant position only. For example, you want to write 0xABCD as the last transfer on the 32-bit bus. ieds_fifo_wr_data_i is 0x0000ABCD and ieds_fifo_wr_byte_en_i is 0x3. ieds_fifo_wr_byte_en_i must not be other than 0x3.</p>
ieds_hand_over_partial_filled_fifo_pl_i	Input	<p>Hand Over Partially Filled FIFO Pulse Assert the signal for one clock cycle to hand over the partially filled buffer. It means, whenever the last data has been written into FIFO and the ieds_fifo_wr_full_o signal is low on the next clock cycle of it, this signal must be asserted for one clock cycle. Make sure the following conditions are met:</p> <ul style="list-style-type: none"> • This must be an active high pulse of a single clock cycle. • Once you assert this signal, the buffer is handed over to the firmware for processing and ieds_fifo_wr_allowed_o goes low after one clock cycle. Wait for the ieds_fifo_wr_allowed_o signal to go high again before initiating the next write operation.

Port	Direction	Description
ieds_zero_len_xfer_req_pl_i	Input	<p>Zero Length Packet Transfer Pulse</p> <p>Assert the signal for one clock cycle to transfer zero length packet. In a special case, when the last data has been written into the FIFO and ieds_fifo_wr_full_o signal becomes high at the next clock cycle, it is mandatory to assert this signal for one clock cycle when ieds_fifo_wr_allowed_o becomes high again.</p>
ieds_fifo_wr_full_o	Output	<p>Full</p> <p>When asserted, the IN FIFO is considered full.</p> <p>Note: Do not perform write operation when the FIFO is full.</p>
ieds_fifo_wr_almost_full_o	Output	<p>Almost Full</p> <p>Asserted when there is a space for only single write operation. It is used as an early indication of the full signal.</p>
ieds_fifo_wr_allowed_o	Output	<p>Write Allowed</p> <p>Asserted when FIFO write operation is allowed. This signal would go low for one clock cycle after either ieds_fifo_wr_full_o, or ieds_hand_over_partial_filled_fifo_pl_i, or ieds_zero_len_xfer_req_pl_i goes high. This signal also goes low when ieds_zero_len_xfer_req_pl_i goes high.</p> <p>Make sure the following conditions are simultaneously met:</p> <ul style="list-style-type: none"> The Configured bit of the CTRL register of this IP is set by the firmware. At least one buffer is available with the FIFO controller, In Endpoint Data Source, to write data. <p>Example:</p> <p>Assume four buffers are available to n FIFO controller, the In Endpoint Data Source. After the first buffer is written by IEDS and the FIFO full signal goes high, this signal goes low. It takes around three to four efm_clk clock cycles to go high again. During this period, the first buffer is handed over to the firmware and the buffer pointer switches to the next buffer. Same as the first buffer, once you write the second buffer and FIFO full signal goes high, this signal goes low again for three to four efm_clk clock cycles. Thus, if all buffers are filled in this manner, this signal goes low and would go high again only when at-least one buffer is sent by the USB device against the IN request from the USB host.</p> <p>Notes:</p> <ul style="list-style-type: none"> Once FIFO is full and this signal is low, it takes around three to four clock cycles of the FIFO clock to go HIGH again if at-least one buffer is available to the IEDS. Do not perform write operation when this signal is low.
ieds_fifo_invalid_access_o	Output	<p>Invalid Access Pulse</p> <p>Asserted when IN FIFO is being accessed improperly.</p> <p>The following are the conditions considered as invalid access to the FIFO.</p> <ul style="list-style-type: none"> Data is being written into the FIFO while write operation is not allowed. That is, ieds_fifo_wr_req_i is high while ieds_fifo_wr_allowed_o is low. Data is being written into FIFO when FIFO is full. That is, ieds_fifo_wr_req_i is high while ieds_fifo_wr_full_o is high.
ieds_none_buf_pending_o	Output	<p>None Buffer Pending</p> <p>Asserted when there is no buffer pending for processing. This can be used to know whether there is any buffer pending to be transmitted or not.</p>
ieds_fifo_being_flushed_o	Output	<p>FIFO Being Flushed</p> <p>This signal is asserted during the IP initialization process. This indicates the FIFO is being flushed. All buffers are available to the external FIFO controller once FIFO is flushed.</p> <p>Note: Do not perform write operation when this signal is high.</p>

Port	Direction	Description
ieds_fifo_bytes_can_be_written_o	Output	<p>Number of Bytes That Can Be Written</p> <p>This indicates the number of bytes that can be written in the FIFO. An IEDS can use this information to know how many bytes can be written until the FIFO becomes full.</p> <p>Note: This is considered valid only when ieds_fifo_wr_allowed_o is high.</p>

4.6.3. FIFO Write Examples

This section contains examples showing different cases for FIFO write operations, such as full transfer and partial transfer. For these examples, the following configuration is used:

- USB 2.0 speed
- Endpoint's maximum packet size: 512 bytes
- Number of IEBM buffers: 4
- Write data width: 32 bits

Some typical FIFO write operation examples are shown in Figure 4.4 to Figure 4.9.

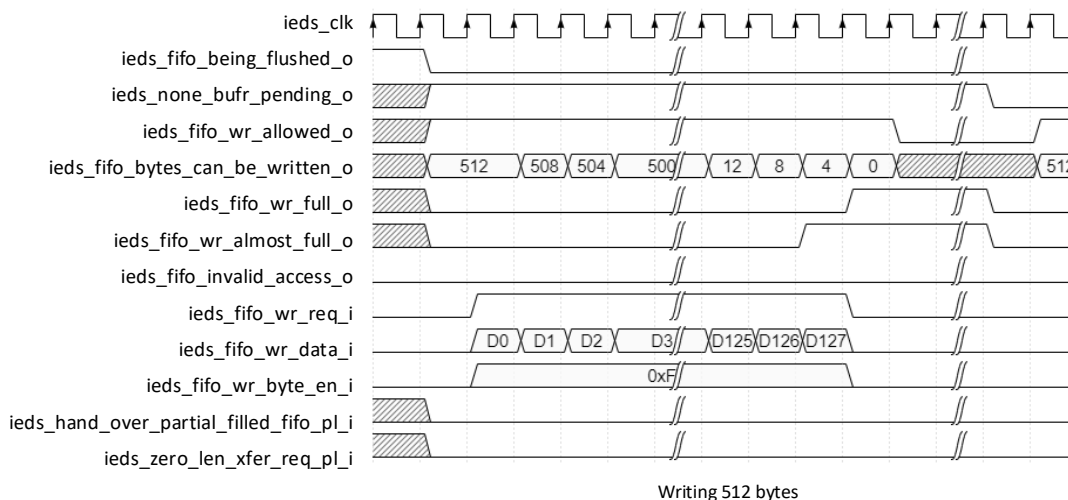


Figure 4.4. FIFO Operation Example – Writing 512 Bytes

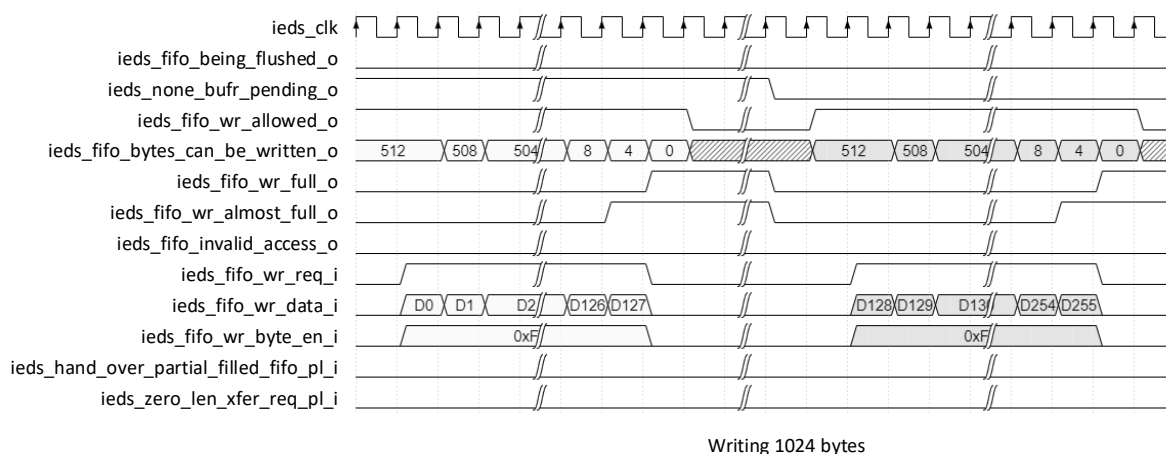


Figure 4.5. FIFO Operation Example – Writing 1024 Bytes

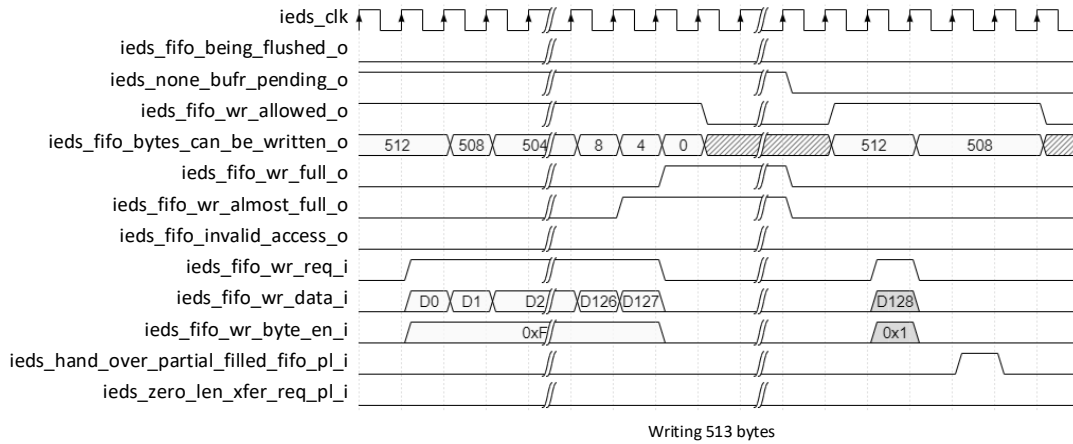


Figure 4.6. FIFO Operation Example – Writing 513 Bytes

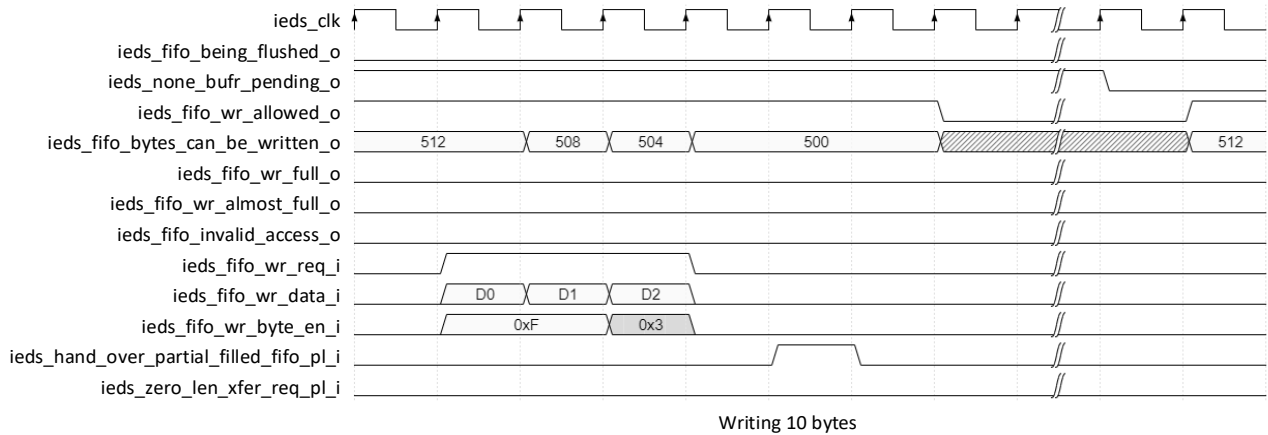


Figure 4.7. FIFO Operation Example – Writing 10 Bytes

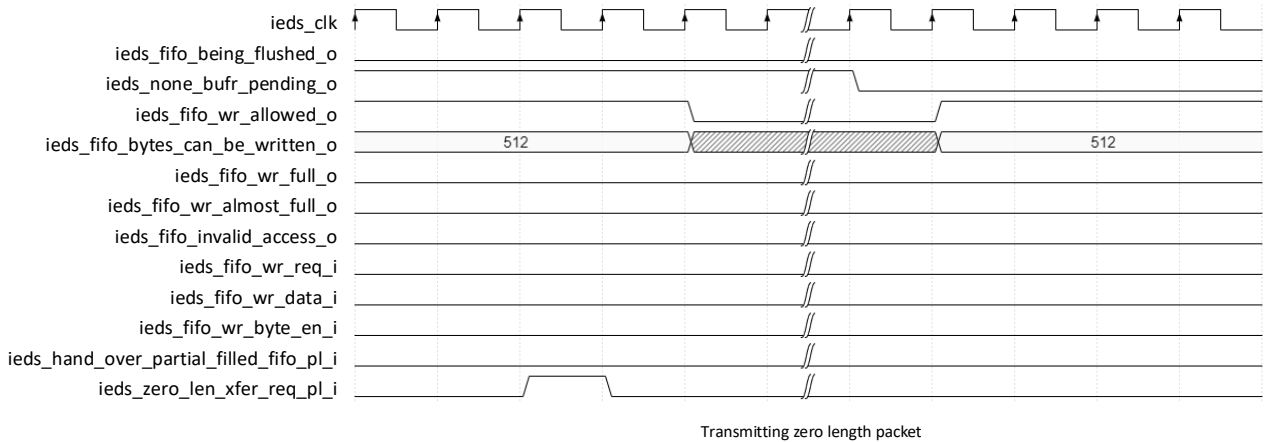


Figure 4.8. FIFO Operation Example – Zero Length Packet Request

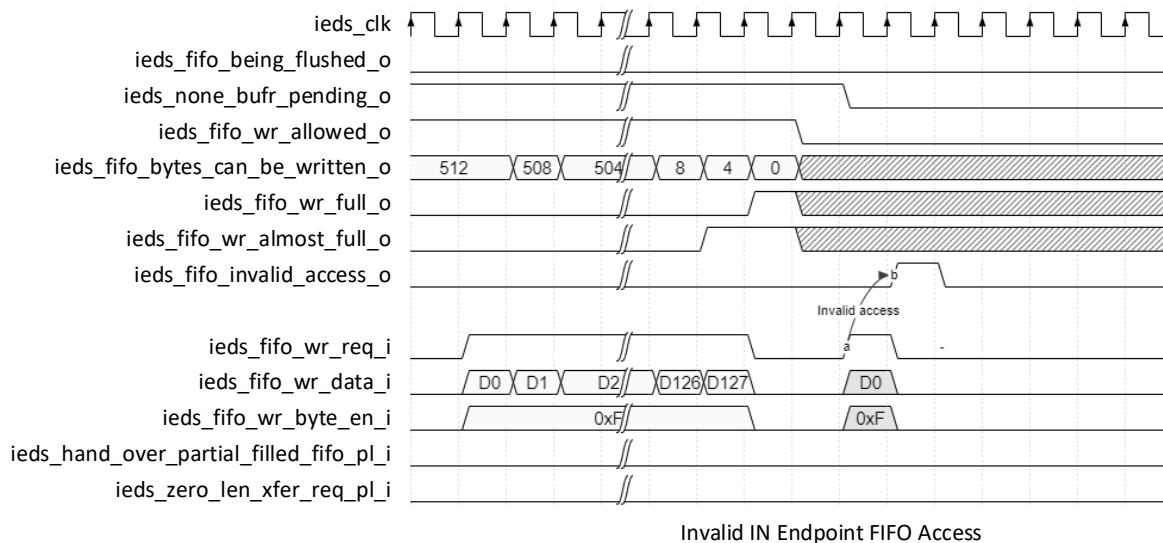


Figure 4.9. FIFO Operation Example – Invalid IN FIFO Access

5. USB23 AXI Manager to Memory Interface Bridge

This section describes the architecture, functionality, and other technical information of the USB23 AXI manager to memory interface bridge. This bridge enables the conversion of AXI4 transfer signals into the native memory interface, which is crucial for performing read and write operations with the True Dual Port RAM. The USB23 CPU utilizes this path to access data provided by the RISC-V processor. Furthermore, the bridge offers a secondary memory interface, allowing the USB23 controller to retrieve data from the IEBM data RAM.

5.1. USB23 AXI Manager to Memory Bridge Architecture

The architecture of the USB23 AXI manager to memory bridge is shown in [Figure 5.1](#).

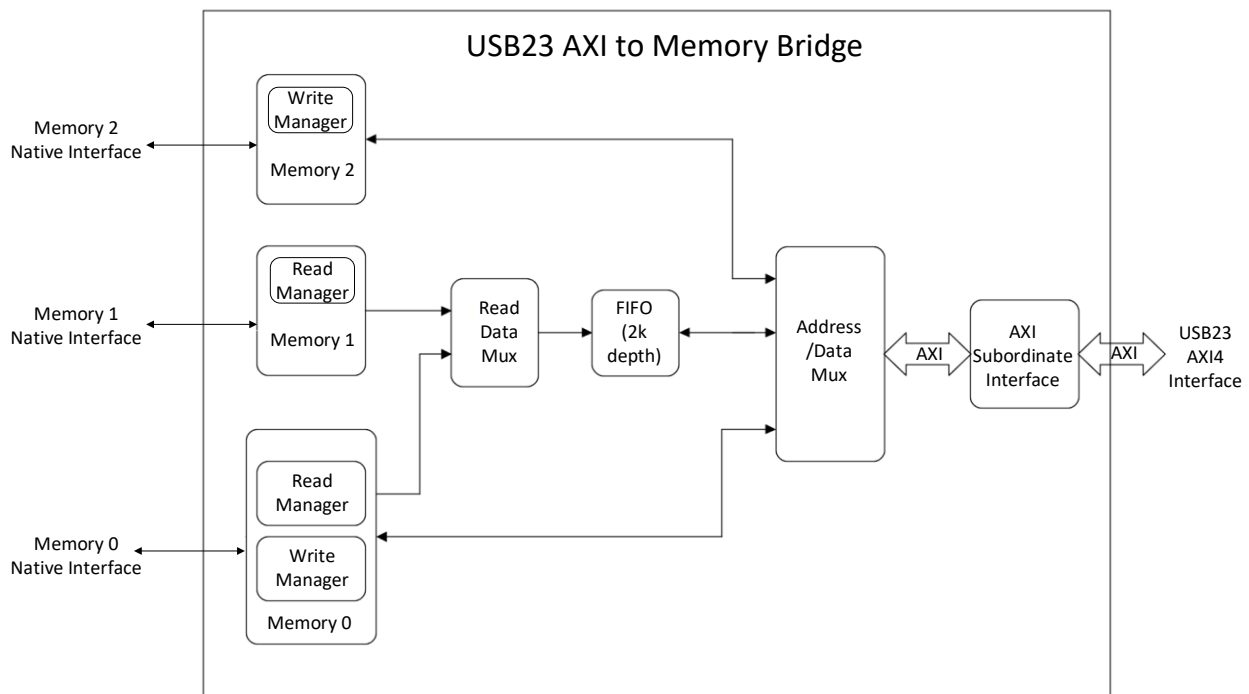


Figure 5.1. USB23 AXI Bridge to Memory Bridge Architecture

5.2. Block Description

The functional blocks are described in the following sections.

5.2.1. AXI Subordinate Interface

The AXI subordinate interface is connected to the AXI4 bus of the USB23 device controller.

5.2.2. Address/Data Mux

This block is responsible for decoding the write address or read address of the specific memory block.

5.2.3. FIFO

This FIFO acts as the intermediate storage when the AXI manager is not ready to accept data. The FIFO size is 2048 KB, 256 in Depth and with 64 bits Data Width.

5.2.4. Read Data Mux

This block is responsible for decoding the memory block that requires a read operation based on the read address.

5.2.5. Memory 0

This block contains both a write manager and a read manager, enabling it to perform both write and read operations.

5.2.6. Memory 1

This block contains a read manager, which is designed to handle read operations.

5.2.7. Memory 2

This block contains a read manager, which is designed to handle read operations.

5.3. IP Core Port and Parameters

This section provides details of the USB23 AXI manager to Memory Bridge IP ports and interface.

5.3.1. Clock and Global Reset

Table 5.1 describes ports for clock and global reset.

Table 5.1. Clock and Reset Ports

Signal Name	Width	Direction	Description
clk	1	Input	Global clock.
reset_n	1	Input	Global reset, active low.

5.3.2. Native Memory Interface

Table 5.2 describes ports for the native memory interface.

Table 5.2. Native Memory Interface Ports

Signal Name	Width	Direction	Description
wr_req_o	1	Input	Write request
wr_addr_8_o	18	Output	Write address
wr_data_o	64	Output	Write data
wr_byte_en_o	8	Output	Write byte enable
rd_data_i	64	Input	Read data
rd_req_o	1	Output	Read request
rd_addr_8_o	18	Output	Read address

5.3.3. AXI4 Interface

Table 5.3 describes ports of the native AXI4 interface.

Table 5.3. AXI4 Interface Ports

Signal Name	Width	Direction	Description
axis_mi_awid_i	8	Input	Write Transaction ID: A unique identifier for the write transaction.
axis_mi_awaddr_i	32	Input	Write Address: The address of the write operation. It is used by the subordinate to determine where to write data.
axis_mi_awlen_i	8	Input	Write Burst Length: The number of transfers in a burst, measured in bytes.
axis_mi_awsz_i	3	Input	Write Burst Size: The size of each transfer in the burst, such as 8-bit, 16-bit, and 32-bit.
axis_mi_awburst_i	2	Input	Write Burst Type: The type of burst transfer, such as fixed, incrementing, or wrapping.
axis_mi_awvalid_i	1	Input	Write Address Valid: Indicates that a valid write address is available from the manager.

Signal Name	Width	Direction	Description
axis_mi_awready_o	1	Output	Write Address Ready: Indicates that the subordinate is ready to accept the write address.
axis_mi_wdata_i	64	Input	Write Data: The data being written to the subordinate. The subordinate stores this data at the address specified in AWADDR.
axis_mi_wstrb_i	8	Input	Write Strobes: A set of byte-enable signals that indicate which bytes of the data are valid for writing.
axis_mi_wlast_i	1	Input	Write last: This signal indicates the last transfer in a write burst.
axis_mi_wvalid_i	1	Input	Write Data Valid: Indicates that valid data is available on the WDATA bus for the subordinate.
axis_mi_wready_o	1	Output	Write Data Ready: Indicates that the subordinate is ready to accept the write data.
axis_mi_bid_o	8	Output	Write Transaction ID: A unique identifier for the write transaction, used to match the response with the request.
axis_mi_bresp_o	2	Output	Write Response: The status of the write operation. It can indicate a successful operation, represented by OKAY, or an error, such as SLVERR.
axis_mi_bvalid_o	1	Output	Write Response Valid: Indicates that the subordinate has a valid response for the write transaction.
axis_mi_bready_i	1	Input	Write Response Ready: Indicates that the manager is ready to receive the write response from the subordinate.
axis_mi_arid_i	8	Input	Read address ID: This signal is the identification tag for the read address group of signals.
axis_mi_araddr_i	32	Input	Read address: The read address gives the address of the first transfer in a read burst transaction.
axis_mi_arlen_i	8	Input	Burst length: This signal indicates the exact number of transfers in a burst.
axis_mi_arsize_i	3	Input	Burst size: This signal indicates the size of each transfer in the burst.
axis_mi_arburst_i	2	Input	Burst type: The burst type and the size information determine how the address for each transfer within the burst is calculated.
axis_mi_arvalid_i	1	Input	Read address valid: This signal indicates that the channel is signaling valid read address and control information.
axis_mi_arready_o	1	Output	Read address ready: This signal indicates that the subordinate is ready to accept an address and associated control signals.
axis_mi_rready_i	1	Input	Read Data Ready: Indicates that the manager is ready to accept read data from the subordinate.
axis_mi_rid_o	8	Output	Read Transaction ID: A unique identifier for the read transaction, used to match the response with the request.
axis_mi_rdata_o	64	Output	Read Data: The data being read from the subordinate. The manager reads this data at the address specified in ARADDR.
axis_mi_rresp_o	2	Output	Read Response: The status of the read operation. It can indicate a successful operation, represented by OKAY, or an error, such as SLVERR.
axis_mi_rlast_o	1	Output	Read last: This signal indicates the last transfer in a read burst.
axis_mi_rvalid_o	1	Output	Read Data Valid: Indicates that valid read data is available for the manager to read.

5.3.4. Configurable Parameters

Table 5.4 describes the configurable parameters of the USB23 AXI Manager to Memory Bridge IP.

Table 5.4. User Configurable Parameters

Parameter	Description
AXI_ADDR_WIDTH_8_I	AXI Subordinate Address Width, in terms of byte offset.
MEM0_ADDR_WIDTH_8_I	Memory 0 Address Width, in terms of byte offset. Valid Value: 18
MEM1_ADDR_WIDTH_8_I	Memory 1 Address Width, in terms of byte offset. Valid Value: 18
MEM2_ADDR_WIDTH_8_I	Memory 2 address width, in terms of byte offset. Valid Value: 18
MEM0_READ_LATENCY_I	Read Latency of External Memory connected with Memory 0 Valid Value: 1
MEM1_READ_LATENCY_I	Read Latency of External Memory connected with Memory 1 Valid Value: 2
MEM2_READ_LATENCY_I	Read Latency of External Memory connected with Memory 2 Valid Value: 1
AXI_DATA_WIDTH_I	AXI Subordinate Data Width, in terms of bits. Valid Value: 64
AXI_BYTE_EN_WIDTH_I	AXI Byte Enable Width Valid Value: 8
BASE_ADDRESS_S0	Base Address of Memory 0 Note: A memory span of 256 KB is needed.
END_ADDRESS_S0	End Address of Memory 0 Note: A memory span of 256 KB is needed.
BASE_ADDRESS_S1	Base Address of Memory 1 Note: A memory span of 256 KB is needed.
END_ADDRESS_S1	End Address of Memory 1 Note: A memory span of 256 KB is needed.
BASE_ADDRESS_S2	Base Address of Memory 2 Note: A memory span of 256 KB is needed.
END_ADDRESS_S2	End Address of Memory 2 Note: A memory span of 256 KB is needed.

6. UVC Reference Design Signal Description

The input and output interface signals for the CrosslinkU-NX USB Video Class Reference Design are shown in [Table 6.1](#).

Table 6.1. Primary I/O

Port Name	Direction	LIFCL-33U Ball	Description
clk_60m_i	Input	H8	60 MHz input clock form the on-board crystal oscillator.
dp_z	Input/Output	D7	USB2.0 positive differential data line
dm_z	Input/Output	E7	USB2.0 negative differential data line
u3_rxm_i	Input	A8	USB3.0 positive differential data line for receiver
u3_rxp_i	Input	B8	USB3.0 negative differential data line for receiver
vbus_z	Input/Output	E5	Power signal
u2_reset_ext_z	Input/Output	—	Synchronous reset for USB 2.0
REFINCLKEXTP_i	Input	F8	USB3.0 PHY external positive differential clock
REFINCLKEXTM_i	Input	E8	USB3.0 PHY external negative differential clock
u3_txm_o	Output	A7	USB3.0 positive differential data line for the transmitter
u3_txp_o	Output	A6	USB3.0 negative differential data line for the transmitter
cam_scl_z	Input/Output	J6	I2C controller serial clock line (SCL) for the camera sensor
cam_sda_z	Input/Output	H6	I2C controller serial data line (SDA) for the camera sensor
cam_en_o	Output	L7	Camera sensors enable
rx_clk_p_i	Input/Output	E7	Camera positive differential clock line
rx_clk_n_i	Input/Output	E6	Camera negative differential clock line
rx_d0_p_i	Input/Output	M7	Camera positive differential data0 line
rx_d0_n_i	Input/Output	M6	Camera negative differential data0 line
rx_d1_p_i	Input/Output	N7	Camera positive differential data1 line
rx_d1_n_i	Input/Output	N6	Camera negative differential data1 line
uart_rxd_i	Input	F1	UART input for RISC-V microcontroller debugging
uart_txd_o	Output	G1	UART output for RISC-V microcontroller debugging

7. Building the Reference Design

This section describes how to run the UVC reference design using the Lattice Propel design environment and Lattice Radiant software. For more details on these tools used, refer to the [Lattice Propel SDK and Builder User Guide](#), and [Lattice Radiant Software User Guide](#).

7.1. Running Propel SDK Project

The UVC reference design utilizes the hardened USB block inside the LIFCL-33U device, which needs to be configured and perform USB enumeration through the RISC-V firmware. This section describes how to build and run a C project in the Lattice Propel SDK software.

7.1.1. Opening Propel SDK Project

To open the Propel SDK project in Lattice Propel SDK, launch Lattice Propel SDK.

Click the **Browse** button to set the workspace folder to `\hardware\ref_designs\mipi_uvc_nx33u\software`. Then, click the **Launch** button, as shown in [Figure 7.1](#).

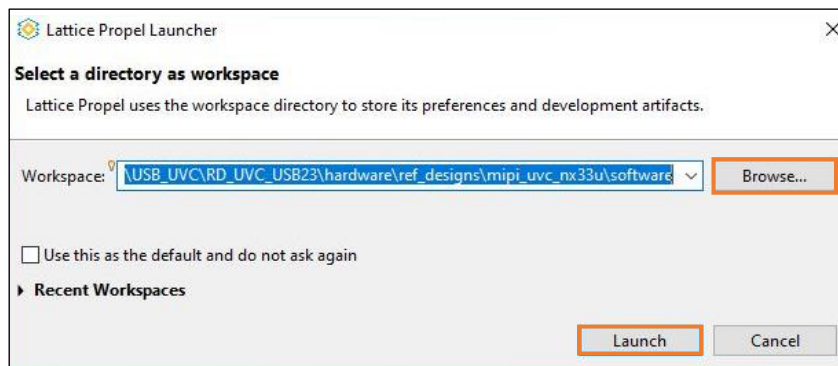


Figure 7.1. Launch Lattice Propel SDK

The Propel SDK project is opened and shown as the RDFW_UVC project ([Figure 7.2](#)).

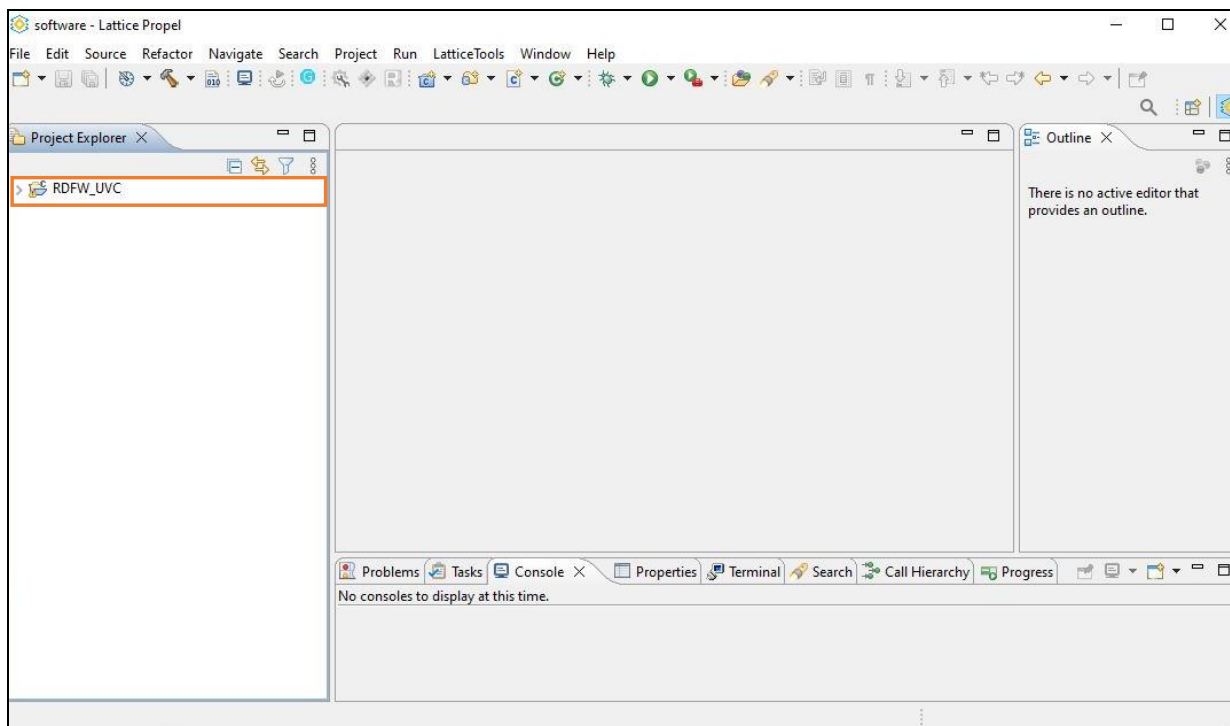


Figure 7.2. Propel SDK Project Opened in Lattice Propel SDK

7.1.2. Navigating Propel SDK Project

In Project Explorer, navigate to RDFS_UVC > src to expand the project file list, as shown in Figure 7.3.

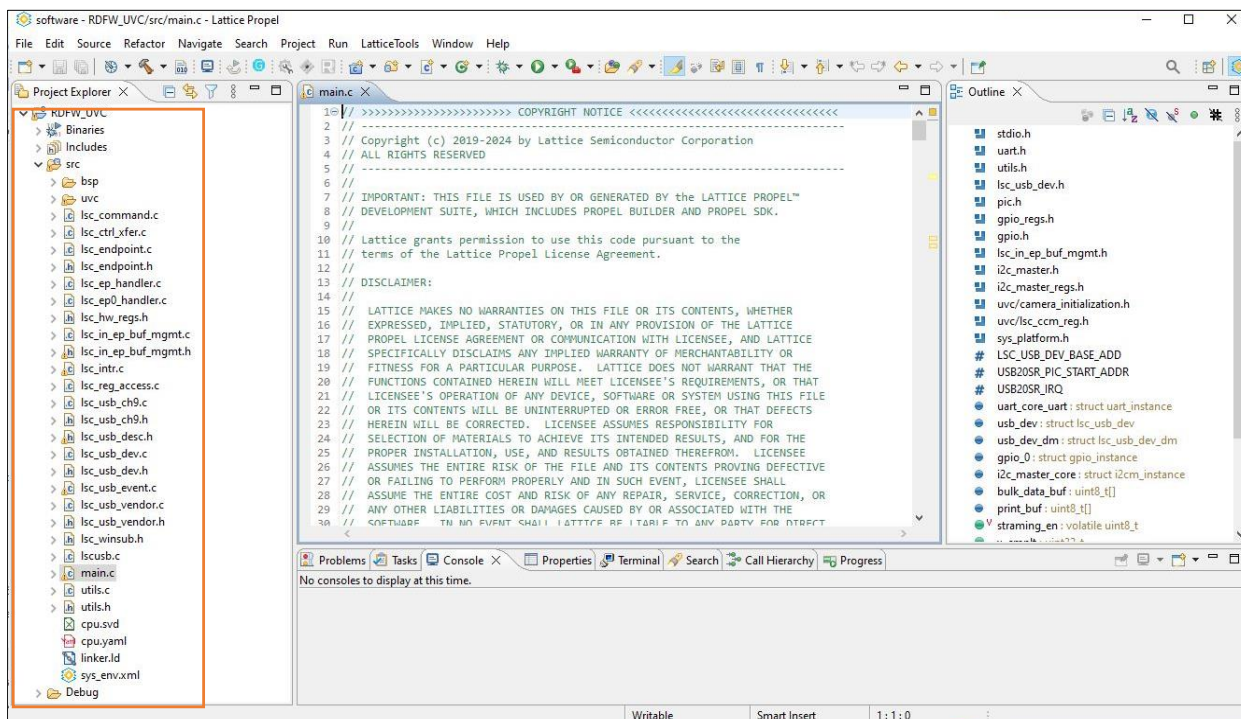


Figure 7.3. Navigating the Propel SDK Project

7.1.3. Generating Output MEM file

To compile the Propel SDK project to generate the memory image for the RISC-V firmware, select **Project > Build Configurations > Set Active > 1 Debug** (Figure 7.4).

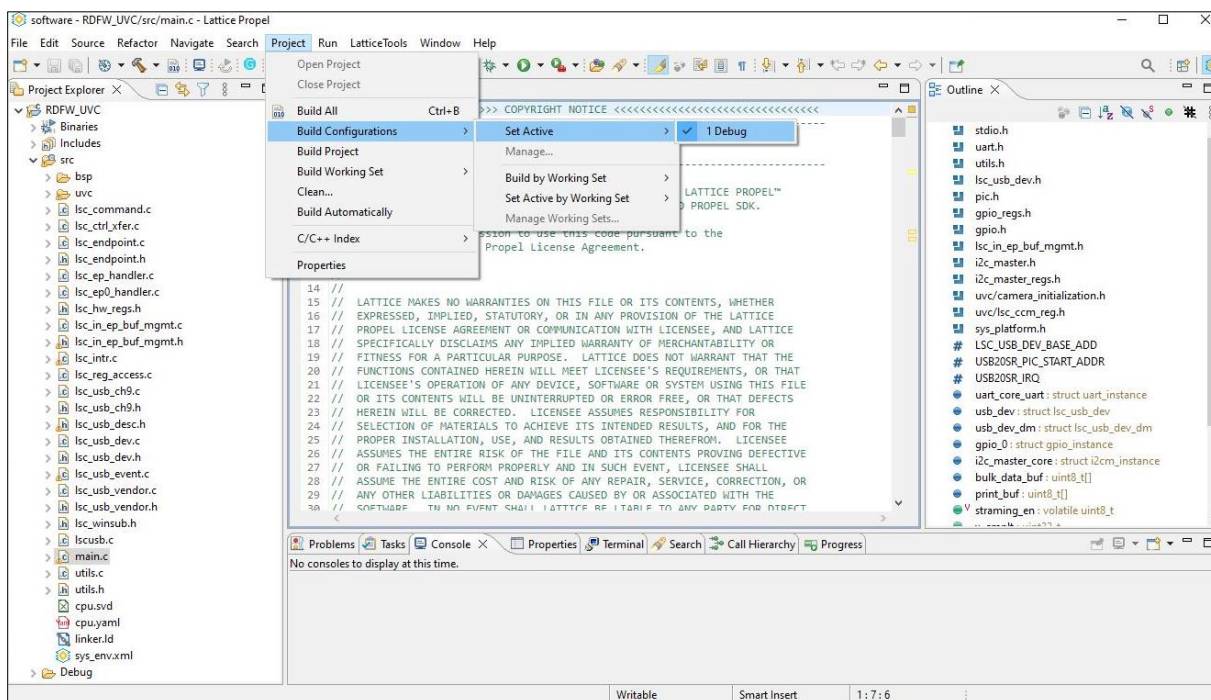


Figure 7.4. Set Propel Build Configuration Mode

To launch the project building process, right-click on the process name **RDFW_UVC**, select **Clean Project** from the menu to clean up the output folders. Then, select **Build Project** from the menu, as shown in Figure 7.5.

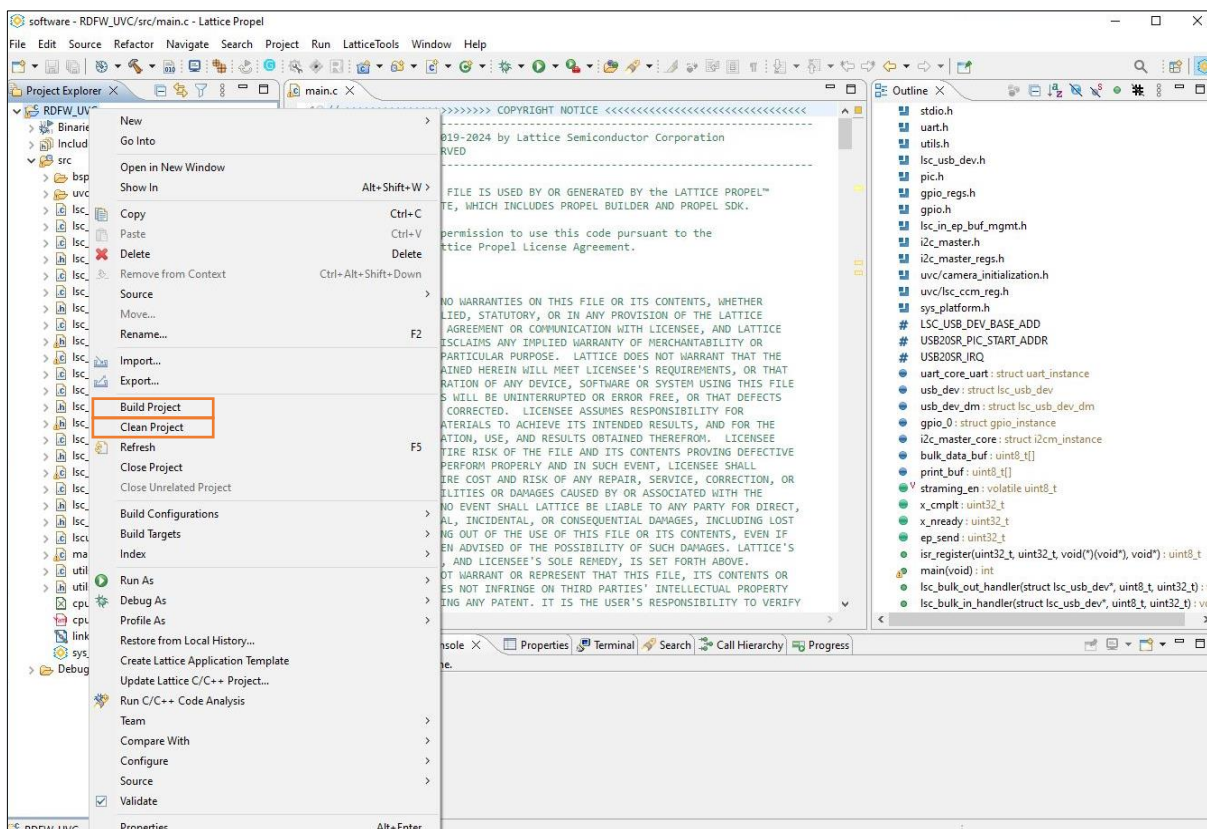


Figure 7.5. Launch the Project Building Process

The output `RDFW_UVC.mem` file is located at:
`\hardware\ref_designs\mipi_uvc_nx33u\software\RDFW_UVC\Debug\RDFW_UVC.mem`.

7.2. Running Propel Builder Design

The UVC reference design utilizes the hardened USB block inside the LIFCL-33U device, which requires a RISC-V microcontroller to handle the USB hard IP enumeration and USB traffic control. This section describes how to evaluate the RISC-V design, including the peripheral soft IPs, in the Lattice Propel Builder software tool.

7.2.1. Install IPs at Local from File

Before opening the UVC Propel Builder project, there are four soft IPs that need to be installed into the IP on Local section in the Lattice Propel Builder software. To perform the soft IP installation process, launch Lattice Propel Builder and follow steps below.

Note: for the latest version of the local IP files, contact Lattice Sales or [Tech Support](#).

1. Right click **IP on Local** and select **Install IP from file...**, as shown in [Figure 7.6](#).

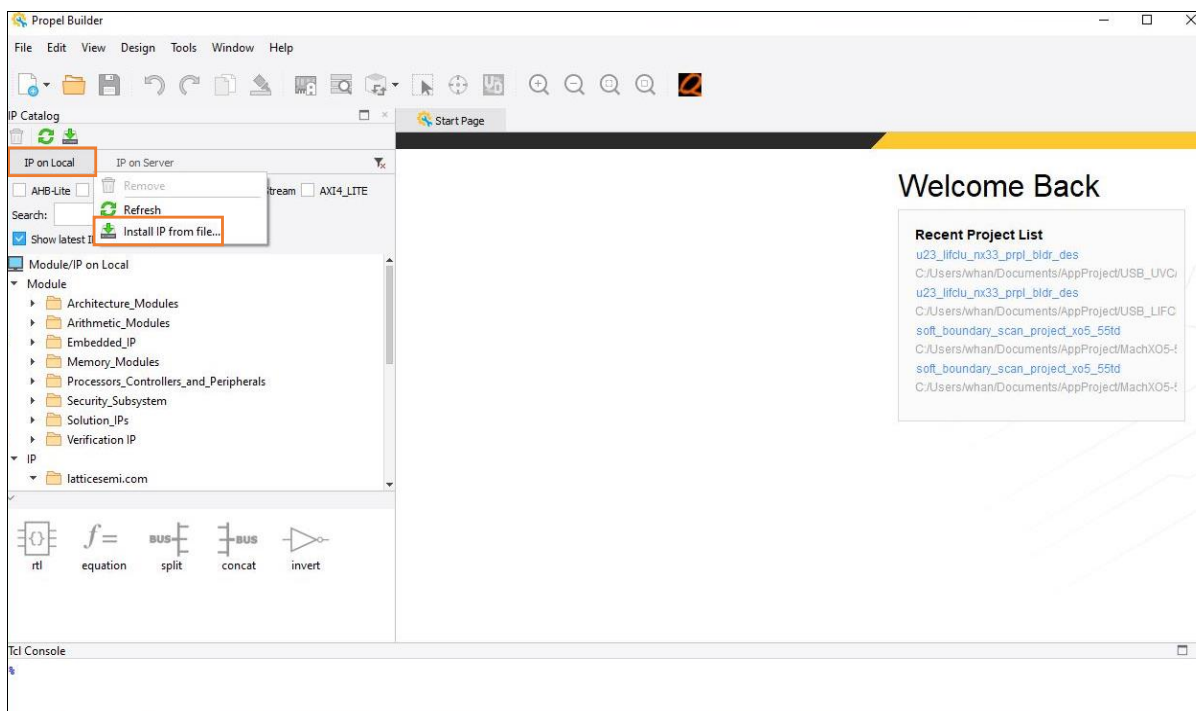


Figure 7.6. Install IP from File

2. In the popped up file browser, Select the following:
\\hardware\ip_packager\ahbl_to_axi4lite\ipk\latticesemi.com_ahbl_to_axi4lite_bridge_1.0.0.0.ipk.
3. Click **OK**.
4. In the **IP License Agreement** window, click **Accept** to finish the IP installation, as shown in Figure 7.7.

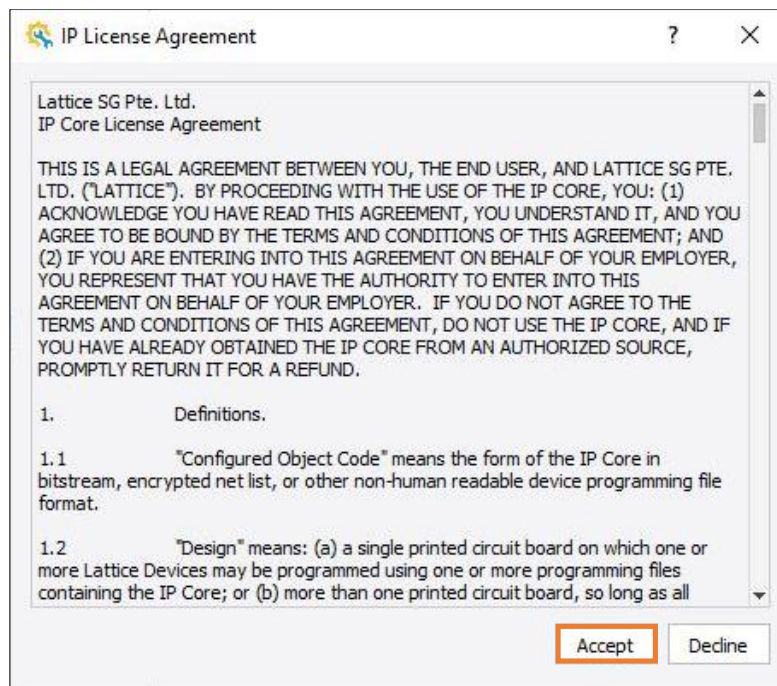


Figure 7.7. Accept IP License Agreement

Repeat Step 1 to Step 4 above to install the rest of the IPs listed below:

- .\hardware\ip_packager\ahbl_to_lmmb_bridge\ipk\latticesemi.com_ahbl_to_lmmb_bridge_1.1.0.0.ipk.
- .\hardware\ip_packager\ahbl_to_mem_bridge\ipk\latticesemi.com_ahbl_to_mem_bridge_1.0.0.0.ipk.
- .\hardware\ip_packager\iebm\ipk\latticesemi.com_in_ep_buffer_manager_2.2.1.0.ipk.

After completing the installation process, these IPs mentioned above should be available in the IP on Local section, as shown in [Figure 7.8](#).

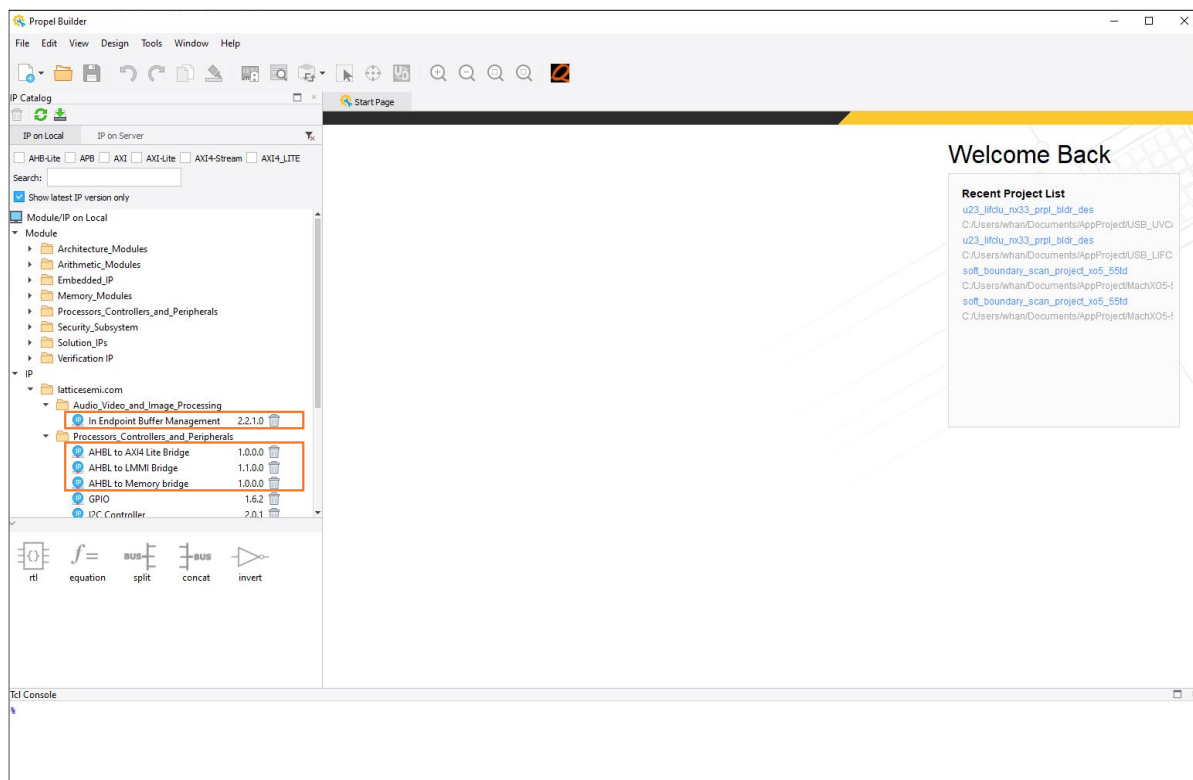


Figure 7.8. IPs Installed from Files

7.2.2. Opening Propel Builder Design

To open the Lattice Propel Builder project in Lattice Propel Builder, launch the software tool. Click **File > Open Design** and navigate through the Open sbx window to open the following ([Figure 7.9](#)):

\\hardware\ref_designs\mipi_uvc_nx33u\u23_lifclu_nx33_prpl_bldr\u23_lifclu_nx33_prpl_bldr\u23_lifclu_nx33_prpl_bldr_des.sbx

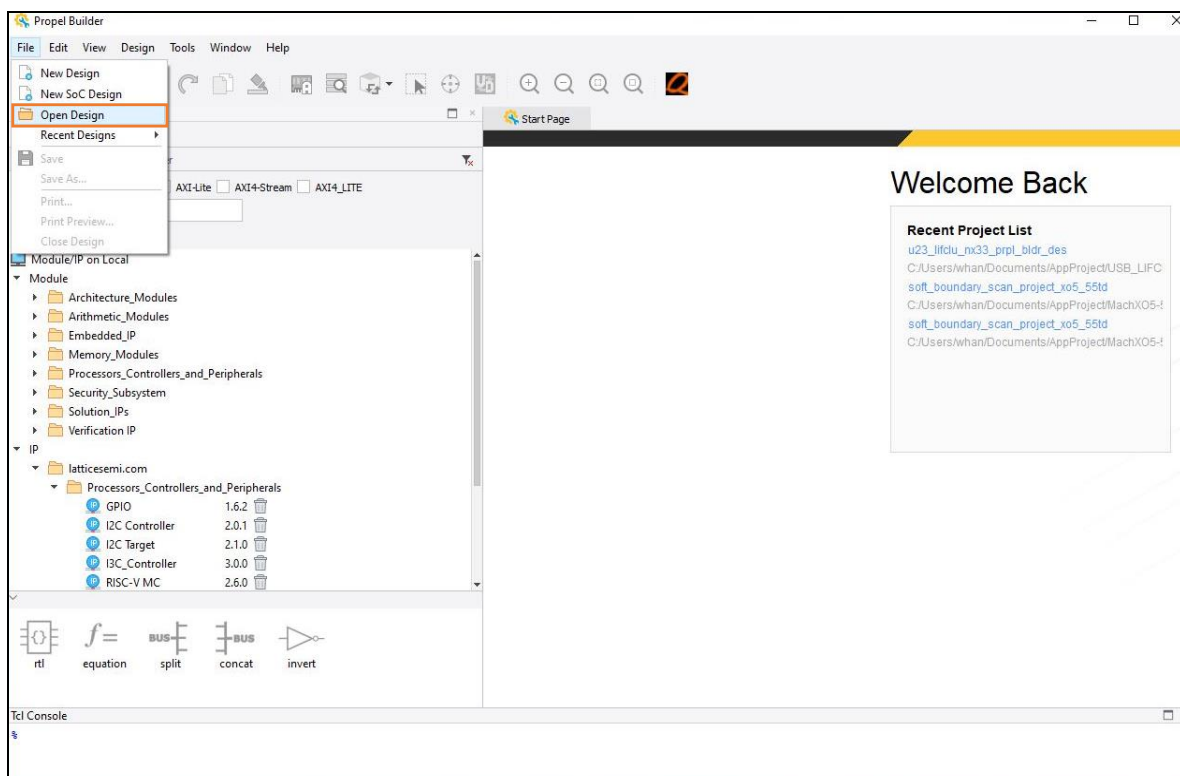


Figure 7.9. Open the Propel Builder Design

7.2.3. Mapping Design with Firmware

In the Propel Builder design schematic, double-click the **systemmem0_inst** to open the system_memory Module/IP Block Wizard, as shown in Figure 7.10.

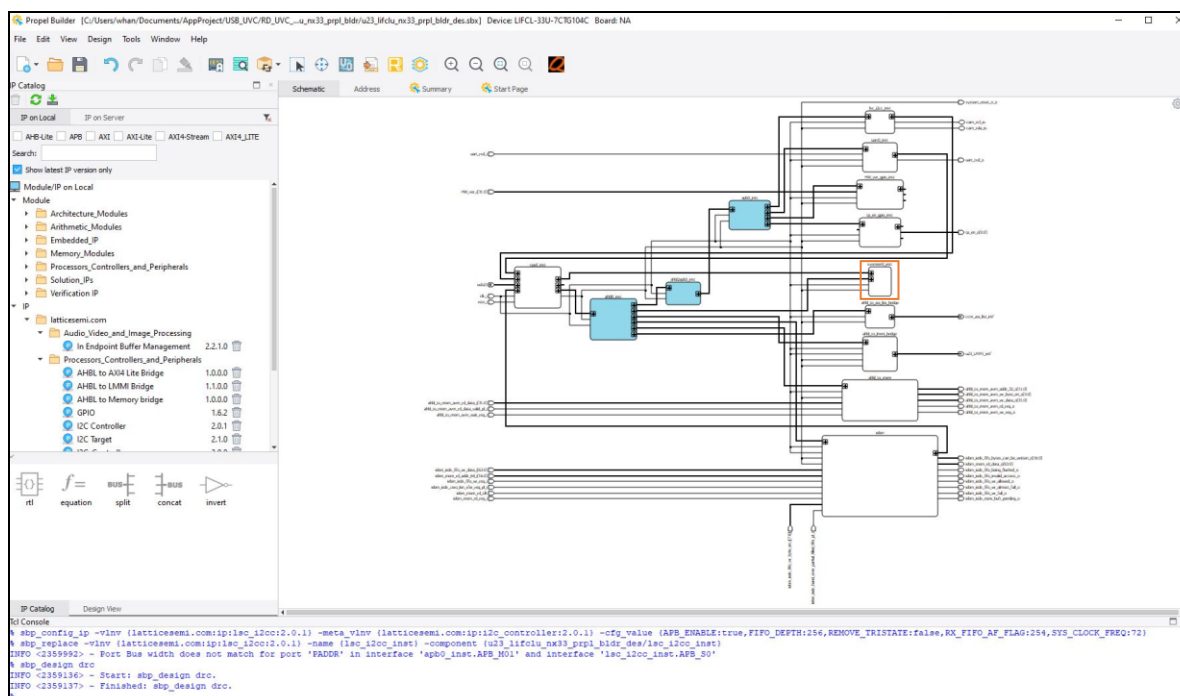


Figure 7.10. Opening System Memory Module Block Wizard

Module/IP Block Wizard

Configure Component from Module system_memory Version 2.3.0
Set the following parameters to configure this component.

Diagram system0

systemmem0

— AHBL_S0

— AHBL_S1

— ahbl_hclk_i

— ahbl_hresetn_i

system_memory

Configure IP

General		Port S0 Settings	Port S1 Settings
Property	Value		
General			
Interface	AHBL		
Memory Address Depth [1 - 81920]	32768		
Data Bus Width(bits)	32		
Memory Type	LRAM		
Port Count	2		
ECC Enable	<input type="checkbox"/>		
Enable Arbiter	<input type="checkbox"/>		
Data Streamer			
Enable Data Streamer	<input type="checkbox"/>		
Data Streamer Interface	Generic FIFO		
Data Streamer Clock Bypass	<input type="checkbox"/>		
Data Streamer Write Start Address [0 - 32767]	0		
Initialization			
Initialize Memory	<input checked="" type="checkbox"/>		
Initialization File Format	hex		
Initialization File	.../software/RDFW_UVC/Debug/RDFW_UVC.mem		

No DRC issues are found.

[User Guide](#)

Generate Cancel

Click the **Address** tab to review the base address assignment for the system and all peripherals within the Propel Builder Project, as shown in [Figure 7.12](#).



7.2.4. Exporting Design to Lattice Radiant Software

To export the Lattice Propel Builder design to the Lattice Radiant software, click on the **Generate** icon as shown in Figure 7.13.

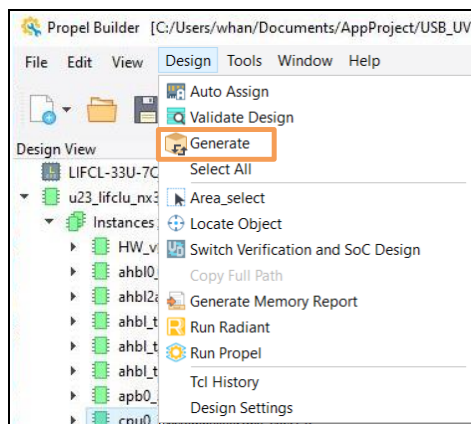


Figure 7.13. Export the Propel Builder Design to Radiant Software

The exported files are located in the ./Design/u23_lifclu_nx33_prpl_bldr/u23_lifclu_nx33_prpl_bldr folder.

7.3. Running Radiant Project

7.3.1. Opening Radiant Project

This section provides the procedure of creating your FPGA bitstream file using the Lattice Radiant software.

To create the FPGA bitstream file, launch the Lattice Radiant software from Lattice Propel Builder. Then, click on the **Open Project** icon, as shown in Figure 7.14.

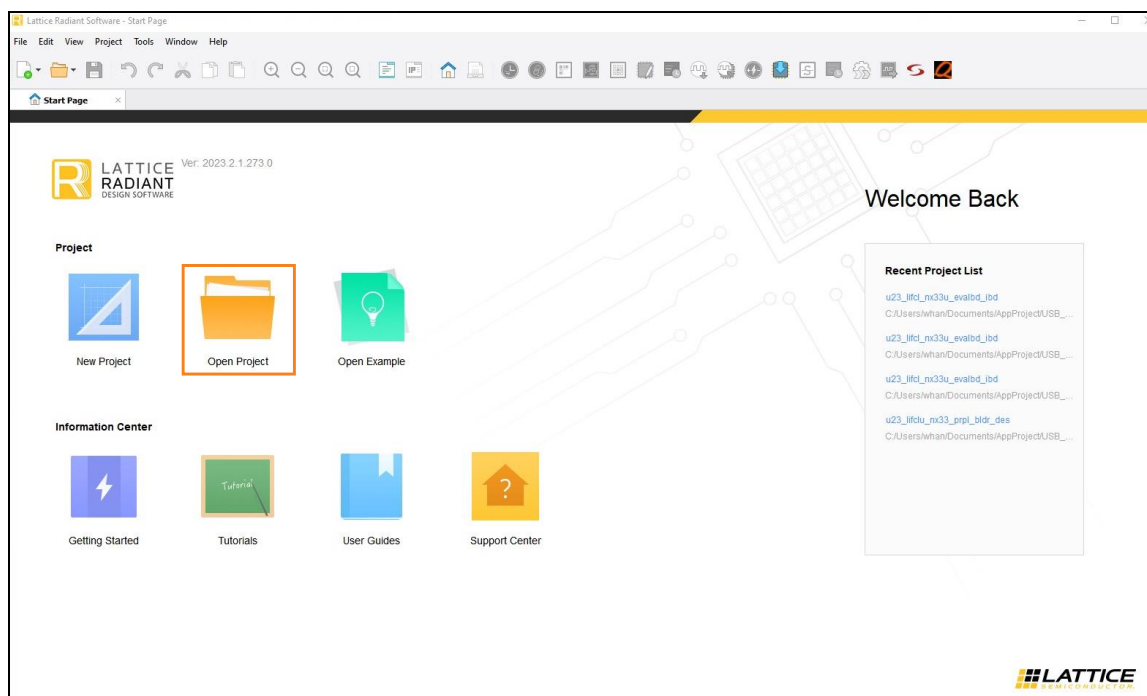


Figure 7.14. Lattice Radiant Software

Open the Radiant project file `mipi_uvc_nx33u_revB_bd_u23axibridge.rdf` from the `\hardware\ref_designs\mipi_uvc_nx33u\` folder, as shown in Figure 7.15. The design should have no errors but there are 1074 warnings due to some unused signals from some instance. The warnings do not affect the design compilation and bitstream generation.

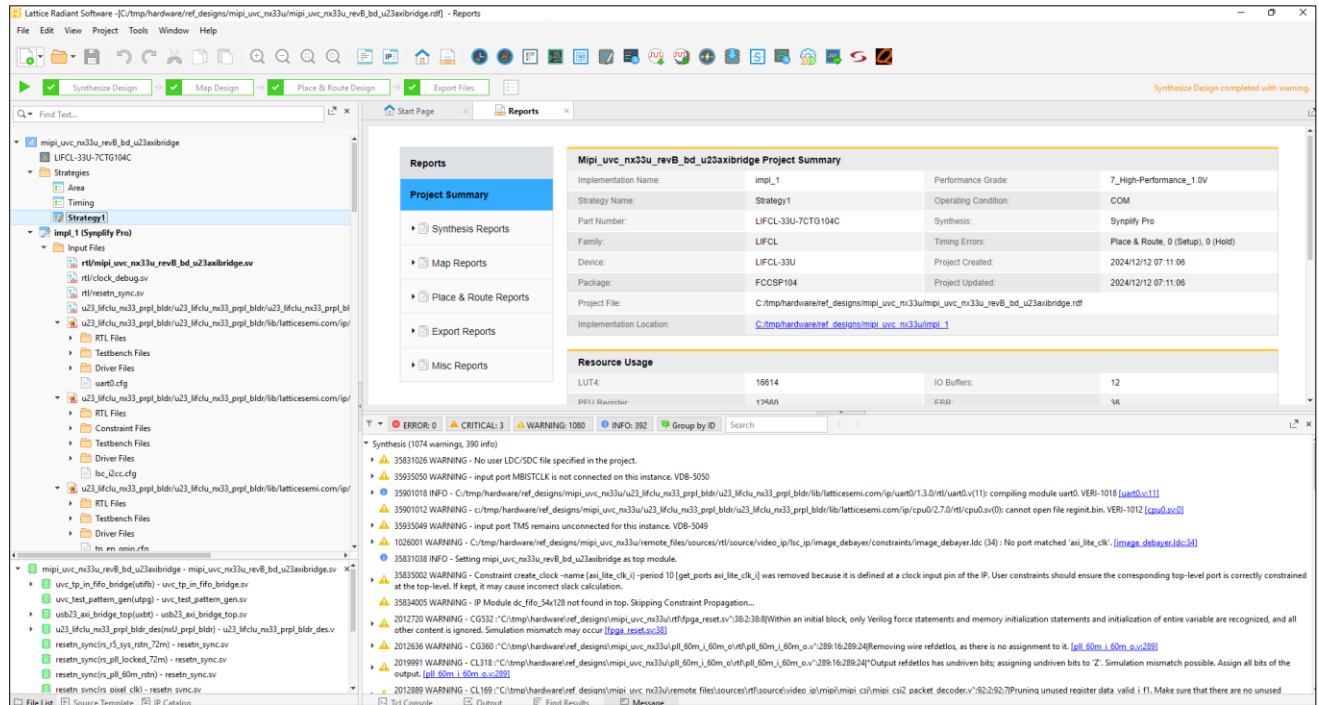


Figure 7.15. Open the Radiant Design

7.3.2. Generating Bitstream File

Click **Export Files** to generate the bitstream file, as shown in Figure 7.16. View the log message from the `\hardware\ref_designs\mipi_uvc_nx33u\impl_1` folder.

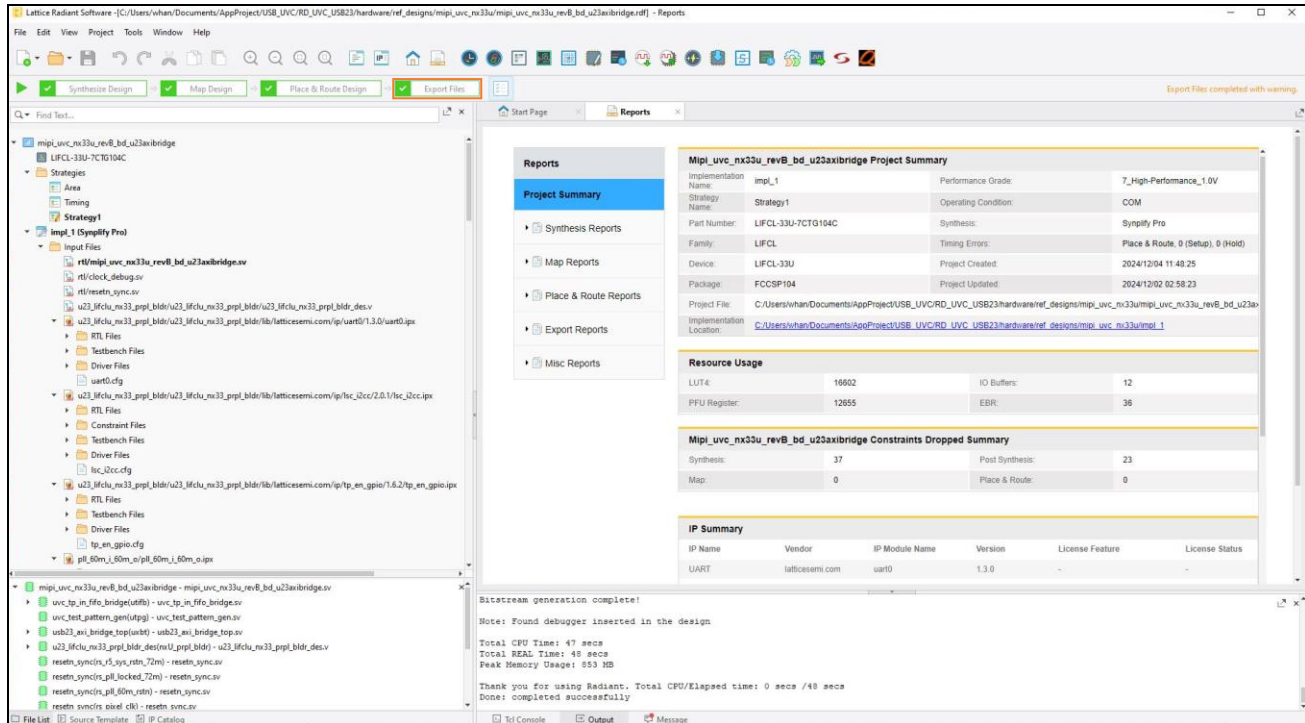


Figure 7.16. Generated Bitstream Log

The generated bitstream is located at:

\\hardware\\ref_designs\\mipi_uvc_nx33u\\impl_1\\mipi_uvc_nx33u_revB_bd_u23axibridge_impl_1.bit.

If the Place & Route Design process encounters an out-of-memory error, reduce the number of host machine cores defined in Strategy1 > Place & Route Design from 7 to a smaller number (Figure 7.17).

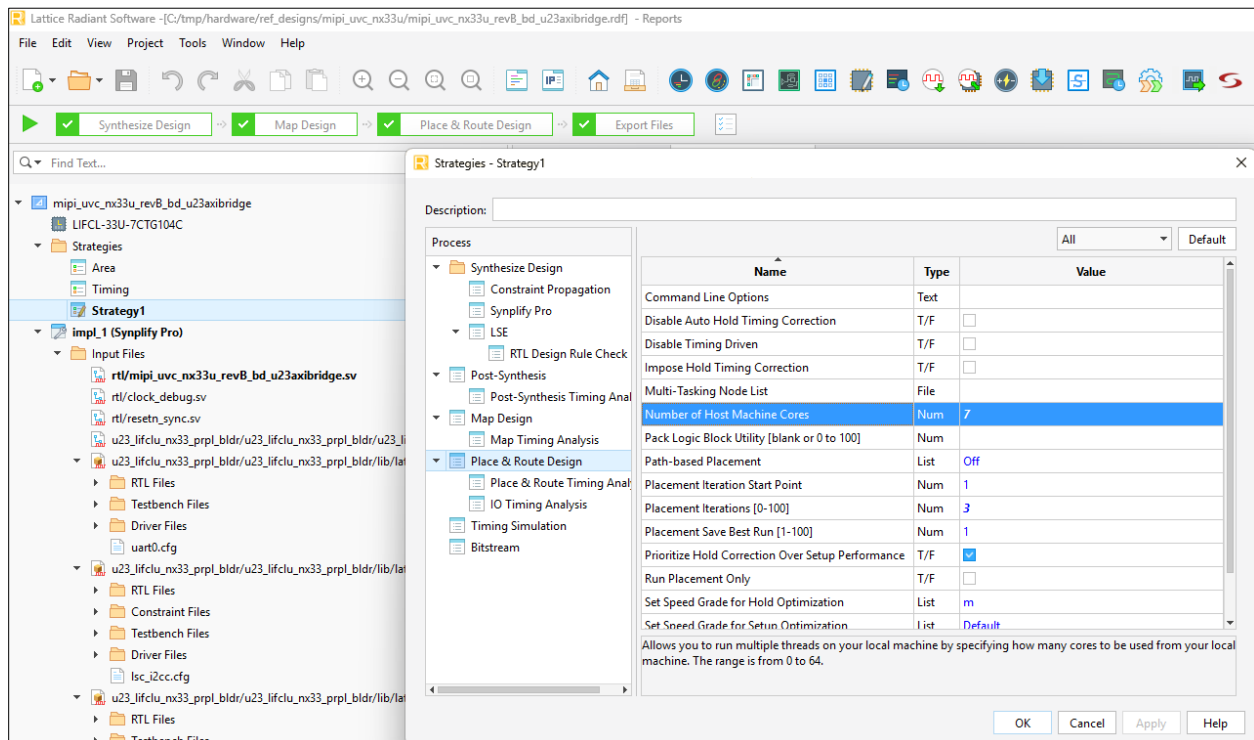


Figure 7.17. Place & Route Design Strategy

8. LIFCL-33U Evaluation Board Programming

In case the LIFCL-33U Evaluation Board needs to be programmed or re-programmed, refer to steps in the following subsections.

8.1. LIFCL-33U Evaluation Board Connection for Programming

Connect the Micro USB port (J2) of the LIFCL-33U Evaluation Board to your PC by using a Micro USB cable.

8.2. Radiant Programmer GUI Setup

1. Launch the Radiant Programmer. Navigate to the **Edit > Device Properties...** menu item or click the **Operation** area to bring up the **Device Properties** window, as shown in [Figure 8.1](#).

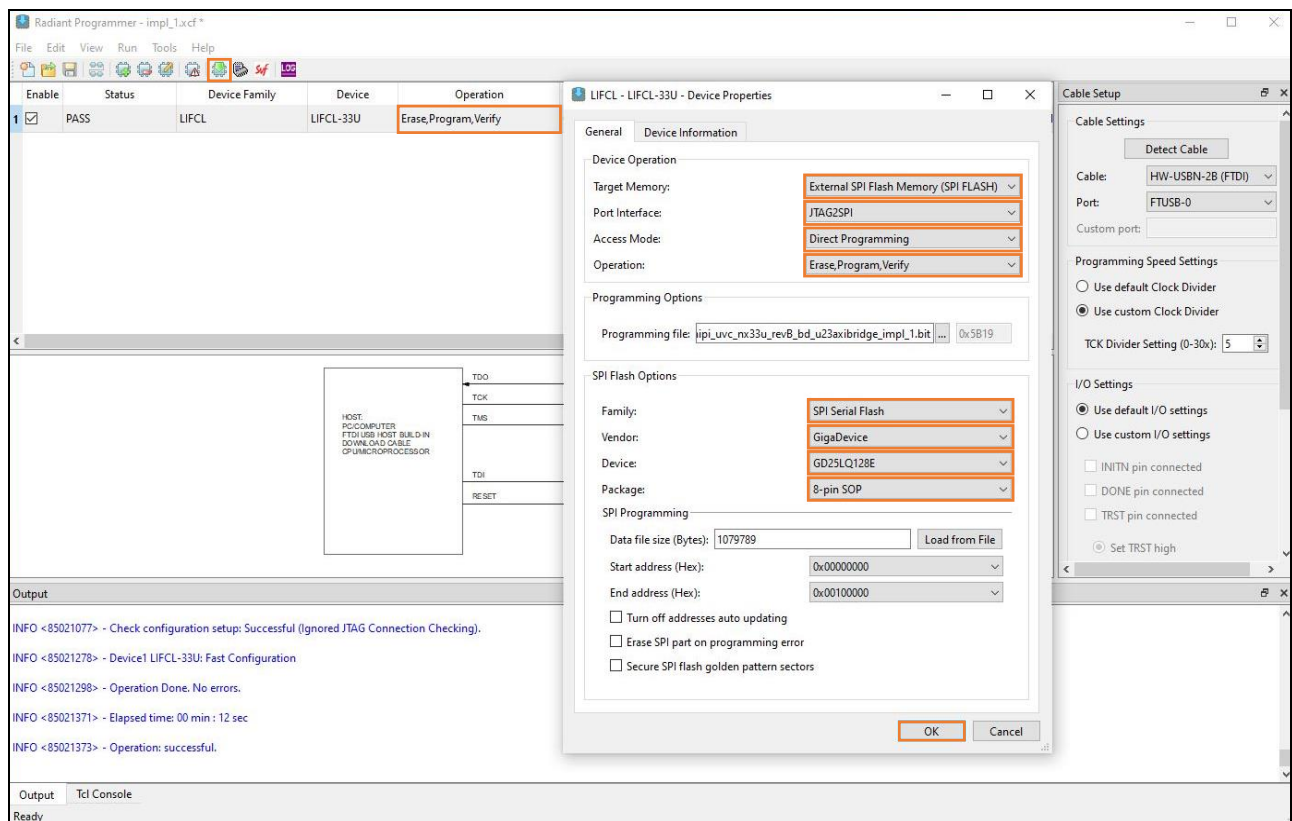


Figure 8.1. Radiant Programmer GUI

If the flash device is a GigaDevice flash, then the SPI Flash options are GD25LQ128E and 8-pin SOP.

If the flash device is a Winbond flash, then the SPI Flash options are W25Q512JV and 8-pad WSON.

2. Select your desired **Target Memory**, **Port Interface**, **Access Mode**, and **Operation**.
3. Select the bitstream file from the bit files folder and choose the **SPI Flash Options** shown in [Figure 8.1](#).
4. Click **OK** after configuration.

8.3. Programming the Evaluation Board

Click the **Program Device** icon from the toolbar to perform the device programming, as shown in [Figure 8.2](#).

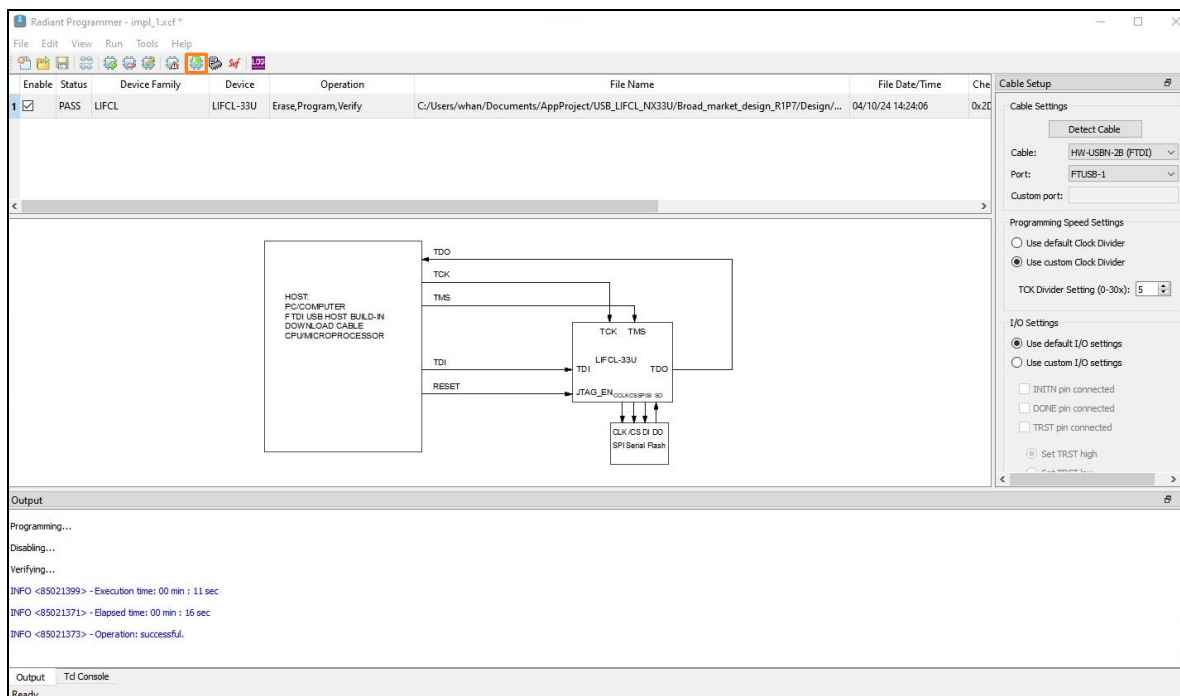


Figure 8.2. Program the Device

9. Running the Reference Design on Evaluation Board

This section describes how to run the UVC Reference Design demo on the LIFCL-33U Evaluation Board. For more details on the LIFCL-33U Evaluation Board, contact [Lattice Sales](#) for the board documentation.

9.1. LIFCL-33U Evaluation Board Connection

Connect the Raspberry Pi Camera Module 2 to the CN1 connector, which is the camera connector. Connect the USB Type C connector CN2 on the pre-programmed LIFCL-33U Evaluation Board to your PC using the USB type C to USB type C cable or the USB type C to USB type A 9 pins cable, as shown in [Figure 9.1](#).

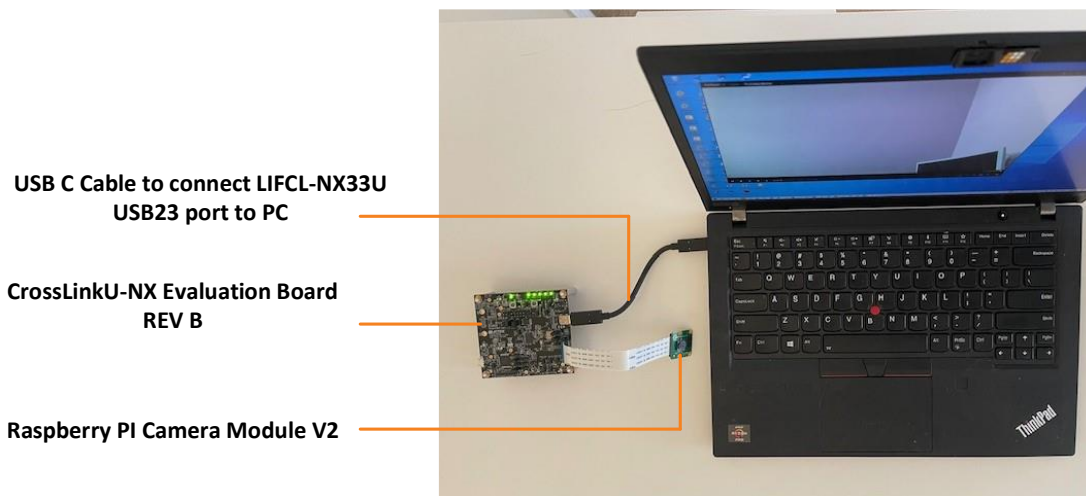


Figure 9.1. CrossLinkU-NX Evaluation Board Setup for Demonstration

Now the device is enumerated as **Lattice USB23**, which can be found in Windows Device Manager ([Figure 9.2](#)).

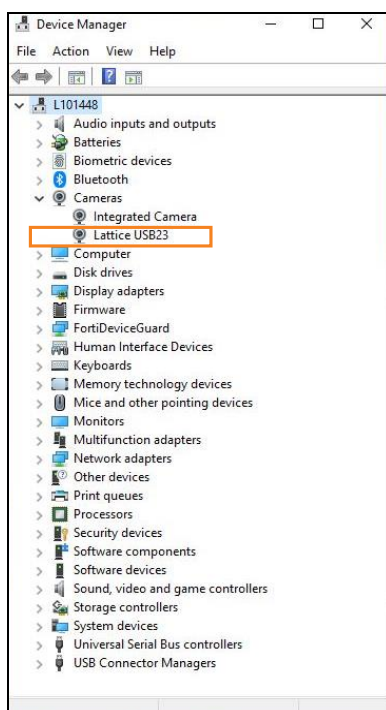


Figure 9.2. USB Enumeration

9.2. UVC Demonstration

To demonstrate the UVC reference design:

1. Open the pot player application.
2. To open video, either hit shortcut Ctrl+J or Click **PotPlayer > Open > Open Webcam/Other Device**, as shown in Figure 9.3.

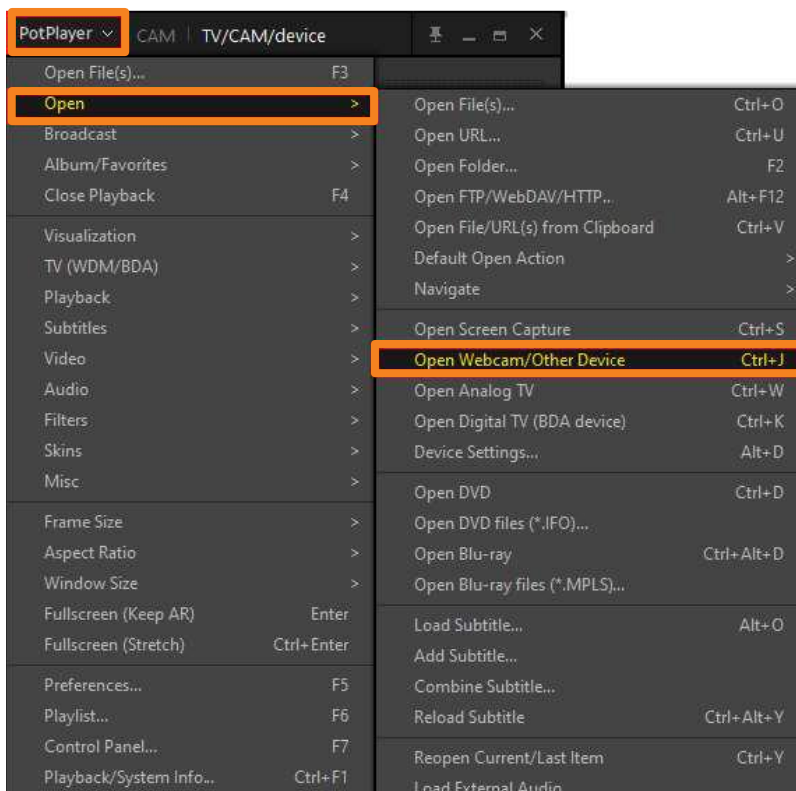


Figure 9.3. Open Webcam/Other Device in PotPlayer

3. The video of the camera sensor can be seen on the PotPlayer screen, as shown in Figure 9.4.



Figure 9.4. Video Output on PotPlayer

- To change the Video Renderer option in PotPlayer, right click on the video and select **Video > Video Renderer > Built in Direct3D 11 Video Renderer**, as shown in Figure 9.5.

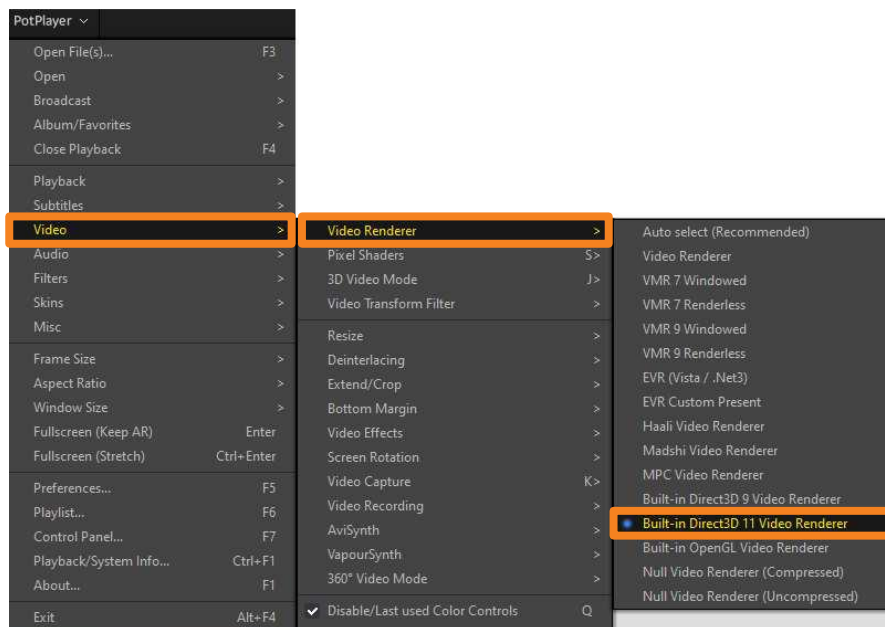


Figure 9.5. Video Renderer Option

- To view the frame per second (FPS), either hit shortcut Ctrl + F1 or click **PotPlayer > Playback/System Info ...**, as shown in Figure 9.6. An example of the video information is shown in Figure 9.7.

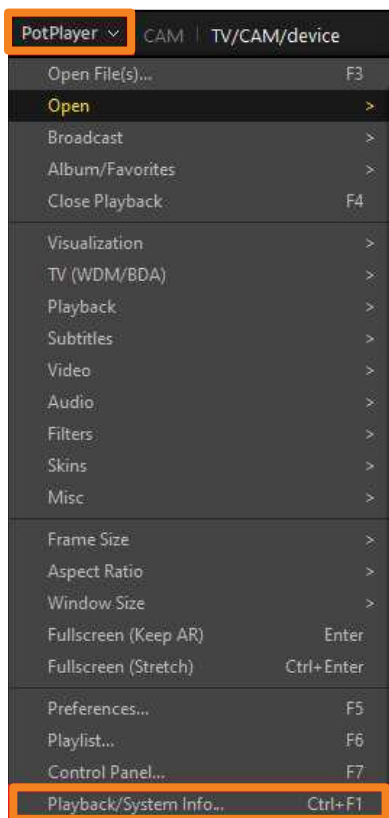


Figure 9.6. Playback/System Information

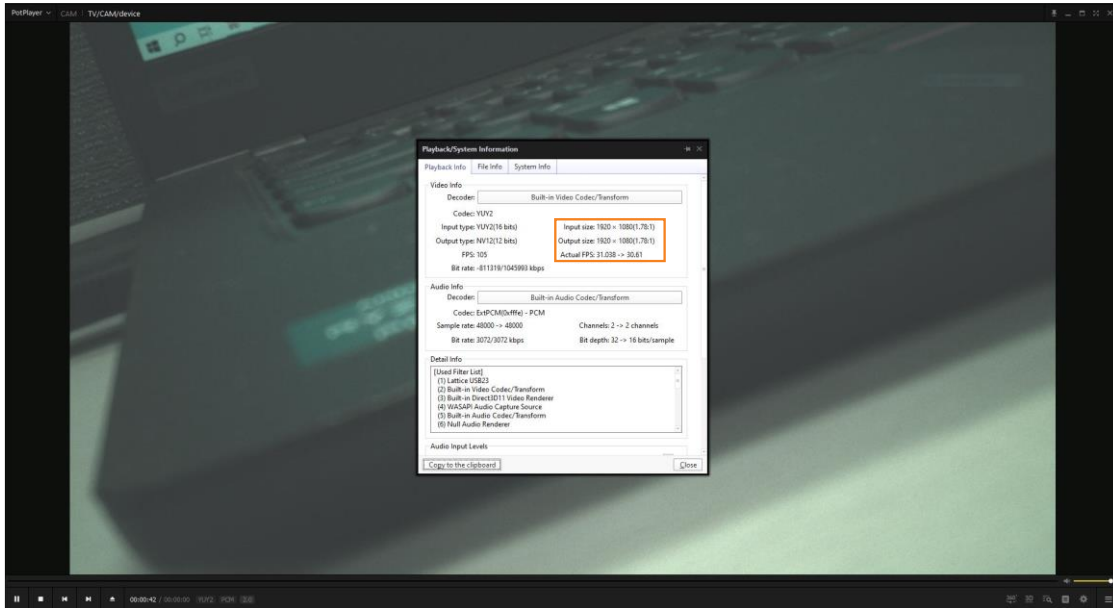


Figure 9.7. Example of Video Information

10. Customizing the Reference Design

This section provides a description for camera support and resolution adjustment. It also provides a way to enable or disable the YUV422 test pattern. For more customization or modification of the UVC reference design, contact [Lattice Sales](#) for information.

10.1. Change the Camera Resolution

This section explains how to change the camera video resolution, which requires changing various files.

10.1.1. Update the user_cnfg_param.v File

1. Open the <MIPI UVC Design Name>/rtl/include/user_cnfg_params.v file.
2. Set the **IMAGE_WIDTH** at line 12 and **IMAGE_HEIGHT** at line 16 according to the desired resolution, as shown in [Figure 10.1](#).

```
6 // !----- User configurable parameters starts -----!  
7  
8 // UCPl.0  
9 // Following parameter is being used in Test pattern generator module.  
10 // Test pattern resolution width parameter.  
11 // Image width in terms of pixels. This number in terms pixels/line.  
12 localparam IMAGE_WIDTH = 1280;  
13  
14 // UCPl.1  
15 // Image depth - This number is terms no of lines per frame.  
16 localparam IMAGE_HEIGHT = 720;  
17
```

Figure 10.1. User Configuration Parameter File

10.1.2. Update RISC-V Firmware

10.1.2.1. Update the lsc_uvc.h File

1. Open the lsc_uvc.h file.
2. Go to line 316 and set the **IMAGE_WIDTH** according to the desired resolution.
3. Go to line 317 and set the **IMAGE_HEIGHT** according to the desired resolution.

10.1.2.2. Update the camera_initialization.h File

1. Open the camera_initialization.h file.
2. To configure the IMX219 camera PLL settings, search for the functions below, as shown in [Figure 10.2](#).
 - cam_pll_setting_u3
 - cam_pll_setting_u2


```

401
402 };
403
404 static unsigned char cam_pll_setting_u2[] = {
405     // PLL_VT_MPY = 20 (Multiplier)
406     // PLL1
407     0x03, 0x06, 0x00,
408     0x03, 0x07, 0x0D, // 1080p
409     // 0x03, 0x07, 0x12, // 720p
410
411     // PLL_OP_MPY = 40 (Multiplier)
412     // PLL2
413     0x03, 0x0C, 0x00,
414     0x03, 0x0D, 0x1A, // 1080p
415     // 0x03, 0x0D, 0x24, // 720p
416 };
417
418 static unsigned char cam_pll_setting_u3[] = {
419     // PLL_VT_MPY = 43 (Multiplier)
420     // PLL1
421     0x03, 0x06, 0x00,
422     0x03, 0x07, 0x25, // 1080p
423     // 0x03, 0x07, 0x24, // 720p
424
425     // PLL_OP_MPY = 86 (Multiplier)
426     // PLL2
427     0x03, 0x0C, 0x00,
428     0x03, 0x0D, 0x4A, // 1080p
429     // 0x03, 0x0D, 0x48, // 720p
430 };
431
432
433
434

```

Figure 10.2. Camera PLL Configurations

10.1.2.3. Update the camera_initialization.c File

1. Open the camera_initialization.c file.
2. Search for the **set_resolution** function. Enable the for loop for required resolutions (Figure 10.3).

```

67 }
68
69 void set_resolution()
70 {
71     for ( int i = 0; i < sizeof(mode_1280x720); i = i + 3)
72     {
73         tx_config_buf[0]=mode_1280x720[i];
74         tx_config_buf[1]=mode_1280x720[i+1];
75         tx_config_buf[2]=mode_1280x720[i+2];
76
77         i2c_master_write (&i2c_master_core,
78             TRGT_SLV_ADDR,
79             0x3, // Every time fix 2 byte of offset address
80             tx_config_buf // Tx length
81         );
82
83         delayMS(5);
84     }
85
86     for ( int i = 0; i < sizeof(mode_1920x1080); i = i + 3)
87     {
88         tx_config_buf[0]=mode_1920x1080[i];
89         tx_config_buf[1]=mode_1920x1080[i+1];
90         tx_config_buf[2]=mode_1920x1080[i+2];
91
92         i2c_master_write (&i2c_master_core,
93             TRGT_SLV_ADDR,
94             0x3, // Every time fix 2 byte of offset address
95             tx_config_buf // Tx length
96         );
97
98         delayMS(5);
99     }
100 }
101
102
103

```

Figure 10.3. Camera PLL Configurations

10.1.3. Update Lattice Radiant IP Parameters Configuration

10.1.3.1. Update Lattice B2P IP

Set the **Word Count** value according to the requirement. Refer to [Figure 3.22](#) to set the **Word Count** field.

The word count for 1920(W) x 1080(H) resolutions can be calculated based on below equation:

$$\text{Word Count} = \frac{(1920 * 10)}{8} = 2400$$

Here 10 is bits per pixel. For example, RAW10 has 10 bits per pixel.

10.1.3.2. Update Lattice Debayer IP

1. Refer to [Figure 3.23](#), Under the **DEBAYER** setting, set the **Horizontal size** value, that is, **IMAGE_WIDTH** value, according to the requirement.
2. Refer to [Figure 3.24](#), Under **Test parameters** setting, set the **Horizontal pixel Size** and **Vertical pixel Size** according to the desired resolution.

10.2. Steps to Enable YUV422 Test Pattern

1. Open the main.c file from the RISC-V Firmware.
2. Uncomment line 268, as shown in [Figure 10.4](#).
3. Write data value 0x1 to enable the YUV422 test pattern.
4. Rebuild the Propel SDK project.

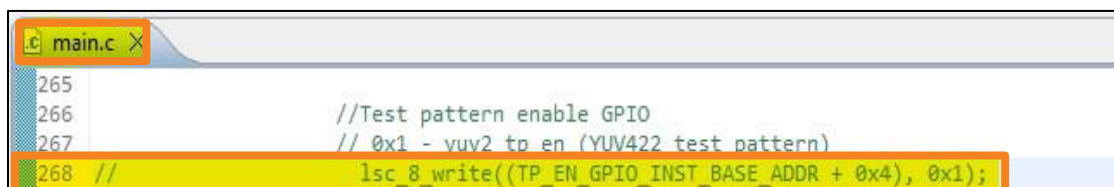


Figure 10.4. Firmware Code to Enable the UYV422 Test Pattern

10.3. Steps to Select Memory Type for the In Endpoint Buffer Manager

This section explains the steps to select memory configuration for the IEBM.

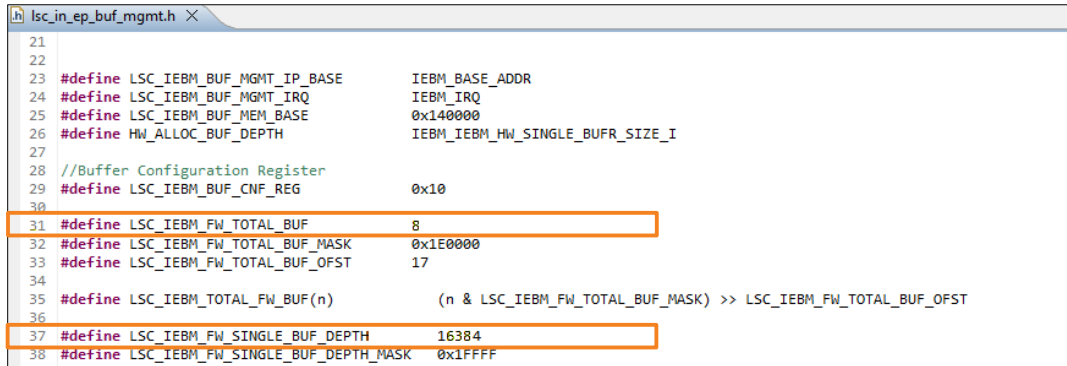
10.3.1. Steps for IEBM Memory Type Selection

1. Open the Propel Builder project from the following path:
mipi_uvc_nx33u_revB_bd_u23axibridge/u23_lifclu_nx33_prpl_bldr/u23_lifclu_nx33_prpl_bldr/u23_lifclu_nx33_prpl_bldr_des.sbx
2. Find the IEBM IP and open the configuration window, as shown in [Figure 3.28](#).
3. Select the **Memory Type** parameter according to your requirement. Choose either **EBR** or **LRAM**.
4. Enter the **Hardware Maximum Buffers** and **Hardware Single Buffer Size** parameter according to your requirement.
5. Click **Generate**, as shown in [Figure 3.28](#).
6. Validate and generate the Propel builder design, as shown in [Figure 7.13](#).

10.3.2. Firmware Modification

1. Open the lsc_in_ep_buf_mgmt.h file.
2. Enter the value for **LSC_IEBM_FW_SINGLE_BUF_DEPTH** based on your requirement. The value should be less than or equal to that of the **Hardware Single Buffer Size** parameter of the IEBM IP.

3. Enter the value for **LSC_IEBM_FW_TOTAL_BUF** based on your requirement. It should be less than or equal to that of the Hardware Maximum Buffers parameter of the IEBM IP (Figure 10.5).



```

21
22
23 #define LSC_IEBM_BUF_MGMT_IP_BASE      IEBM_BASE_ADDR
24 #define LSC_IEBM_BUF_MGMT_IRQ         IEBM_IRQ
25 #define LSC_IEBM_BUF_MEM_BASE         0x140000
26 #define HW_ALLOC_BUF_DEPTH           IEBM_IEBM_HW_SINGLE_BUFR_SIZE_I
27
28 //Buffer Configuration Register
29 #define LSC_IEBM_BUF_CNF_REG          0x10
30
31 #define LSC_IEBM_FW_TOTAL_BUF          8
32 #define LSC_IEBM_FW_TOTAL_BUF_MASK    0x1E0000
33 #define LSC_IEBM_FW_TOTAL_BUF_OFST    17
34
35 #define LSC_IEBM_TOTAL_FW_BUF(n)      ((n & LSC_IEBM_FW_TOTAL_BUF_MASK) >> LSC_IEBM_FW_TOTAL_BUF_OFST)
36
37 #define LSC_IEBM_FW_SINGLE_BUF_DEPTH  16384
38 #define LSC_IEBM_FW_SINGLE_BUF_DEPTH_MASK 0x1FFFF
  
```

Figure 10.5. Firmware Configurable Registers

10.4. Steps to Migrate from the FCCSP104 Package to the WLCSP84 Package

This section explains the steps to migrate the reference design in the FCCSP104 package to the WLCSP84 package.

10.4.1. Reference Clock Selection for USB Controller

For the FCCSP104 package, three reference clock sources could be chosen for the USB controller:

- External differential clock source (REFIN_CLK_EXT_P/M)
- External single-end clock source (PB6A/B)
- Internal clock source (PLL instantiation)

For the WLCSP84 package, the external differential clock source is not available.

The pin mapping of the external clocks is based on the location names below:

- Package FCCSP104
 - REFIN_CLK_EXT_P/M
 - ldc_set_location -site {E8} [get_ports REFINCLKEXTM_i]
 - ldc_set_location -site {F8} [get_ports REFINCLKEXTP_i]
 - PB6A/B
 - ldc_set_location -site {H8} [get_ports clk_60m_i]
- Package WLCSP84
 - PB6A/B
 - ldc_set_location -site {M6} [get_ports clk_60m_i]

10.4.2. Firmware Configuration

The default source clock selection is the external single end-clock source. No modification is required when migrating the reference design to the WLCSP84 package.

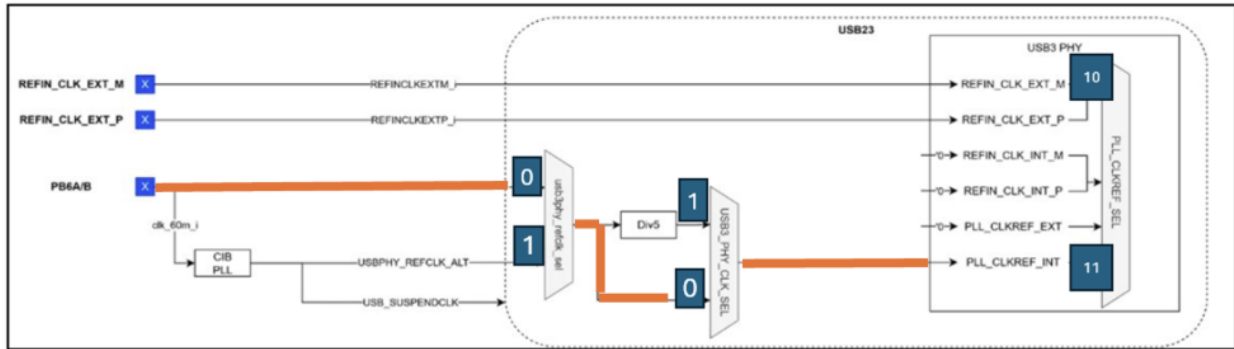


Figure 10.6. USB Reference Clock Selection

Appendix A. Resource Utilization

Table A.1 shows the resource utilization of the UVC reference design for LIFCL-33U-9CTG104C using Synplify Pro® of Lattice Radiant software 2024.2.

Table A.1. Resource Utilization for LIFCL-33U-9CTG104C

Module/Resource Utilization		LUTs	Registers	EBRs	Large RAMs	DSP MULT
Microcontroller System	RISC-V Micro Controller	3791	1918	2	0	6
	System Memory	126	35	0	2	0
	Bus Controller & Converters	509	338	0	0	0
	IEBM	406	448	0	2	0
	GPIO	293	285	0	0	0
	I2C Controller	1378	625	0	0	0
	UART	246	146	0	0	0
Image Signal Processing	MIPI DPHY/CSI	530	329	4	0	0
	Byte to Pixel Converter	590	281	3	0	0
	B2P to AXI Stream	115	72	0	0	0
	Image Debayer	5809	5226	10	0	6
	Color Correction Matrix	1758	1863	6	0	18
	AXI Stream to Parallel	6	53	0	0	0
	Collor Space Converter	288	388	0	0	12
	YUV422 to UVC Bridge	204	167	3	0	0
	IEBM IN FIFO Interface	57	19	0	0	0
	UVC Test Pattern Generation & Bridge	248	138	2	0	0
	USB23 AXI to Memory Bridge	239	173	6	0	0
Misc	Reset Generation & Clock Domain Crossing	7	55	0	0	0
Total		16,586	12,559	36	4	42
Percentage Over the Device		60%	45%	56%	80%	66%

References

- [Lattice Radiant Software User Guide](#)
- [Lattice Propel SDK User Guide \(FPGA-UG-02218\)](#)
- [Lattice Propel Builder User Guide \(FPGA-UG-02219\)](#)
- [RISC-V MC CPU IP User Guide \(FPGA-IPUG-02267\)](#)
- [AHB-Lite Interconnect Module User Guide \(FPGA-IPUG-02051\)](#)
- [AHB-Lite to APB Bridge Module User Guide \(FPGA-IPUG-02053\)](#)
- [System Memory Module User Guide \(FPGA-IPUG-02073\)](#)
- [GPIO IP User Guide \(FPGA-IPUG-02076\)](#)
- [SPI Controller IP User Guide \(FPGA-IPUG-02069\)](#)
- [SPI Target IP User Guide \(FPGA-IPUG-02070\)](#)
- [I2C Controller IP User Guide \(FPGA-IPUG-02071\)](#)
- [I2C Target IP User Guide \(FPGA-IPUG-02072\)](#)
- [UART 16550 IP User Guide \(FPGA-IPUG-02100\)](#)
- [USB 2.0/3.2 IP \(FPGA-IPUG-02237\)](#)
- [CrossLinkU-NX web page](#)
- [Lattice Radiant Software web page](#)
- [Lattice Propel Design Environment web page](#)
- [Lattice Insights web page 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, September 2025

Section	Change Summary
Functional Description	Updated Figure 3.1. Reference Design Block Diagram , Figure 3.10. Test Pattern Enable GPIO Configuration , Figure 3.29. Reference Design Clock Domain Block Diagram , and Figure 3.30. Reference Design Reset Scheme Diagram .
Building the Reference Design	<ul style="list-style-type: none"> Updated Figure 7.12. Base Address Assignment. In the Install IPs at Local from File section, added the following note: <i>Note: for the latest version of the local IP files, contact Lattice Sales or Tech Support.</i> In the Opening Radiant Project section: <ul style="list-style-type: none"> changed <i>To create the FPGA bitstream file, open the Lattice Radiant software to To create the FPGA bitstream file, launch the Lattice Radiant software from Lattice Propel Builder;</i> changed <i>The design should have no errors but there are 30 warnings due to some unused signals from some instance to The design should have no errors but there are 1074 warnings due to some unused signals from some instance;</i> updated Figure 7.15. Open the Radiant Design. In the Generating Bitstream File section: <ul style="list-style-type: none"> added <i>If the Place & Route Design process encounters an out-of-memory error, reduce the number of host machine cores defined in Strategy1 > Place & Route Design from 7 to a smaller number;</i> added Figure 7.17. Place & Route Design Strategy.
LIFCL-33U Evaluation Board Programming	In the Radiant Programmer GUI Setup section, added <i>If the flash device is a GigaDevice flash, then the SPI Flash options are GD25LQ128E and 8-pin SOP. If the flash device is a Winbond flash, then the SPI Flash options are W25Q512JV and 8-pad WSON.</i>
Customizing the Reference Design	Added the Steps to Migrate from the FCCSP104 Package to the WLCSP84 Package section.

Revision 1.0, December 2024

Section	Change Summary
All	Production release.



www.latticesemi.com