MIPI CSI-2 to Ethernet

Reference Design

FPGA-RD-02311-1.1

August 2025

Disclaimers

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

Inclusive Language

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

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations and abbreviations used in this document.

Abbreviations	Definition
AVTP	Audio Video Transport Protocol
FIFO	First In, First Out
HDMI	High-Definition Multimedia Interface
IP	Intellectual Property
MAC	Medium Access Control
MIPI	Mobile Industry Processor Interface
MTU	Maximum Transmission Unit
PHY	Physical Layer
RD	Reference Design

1. Introduction

The MIPI CSI-2 to Ethernet reference design is an advanced system that integrates two camera sensors with each Lattice Avant™ Versa board. It leverages the Audio Video Transport Protocol (AVTP) layer 2 for efficient video transmission, ensuring high-quality video streaming and seamless integration across various components. This design is ideal for applications requiring high-quality, real-time video streaming due to its modular design and support for high-definition video output.

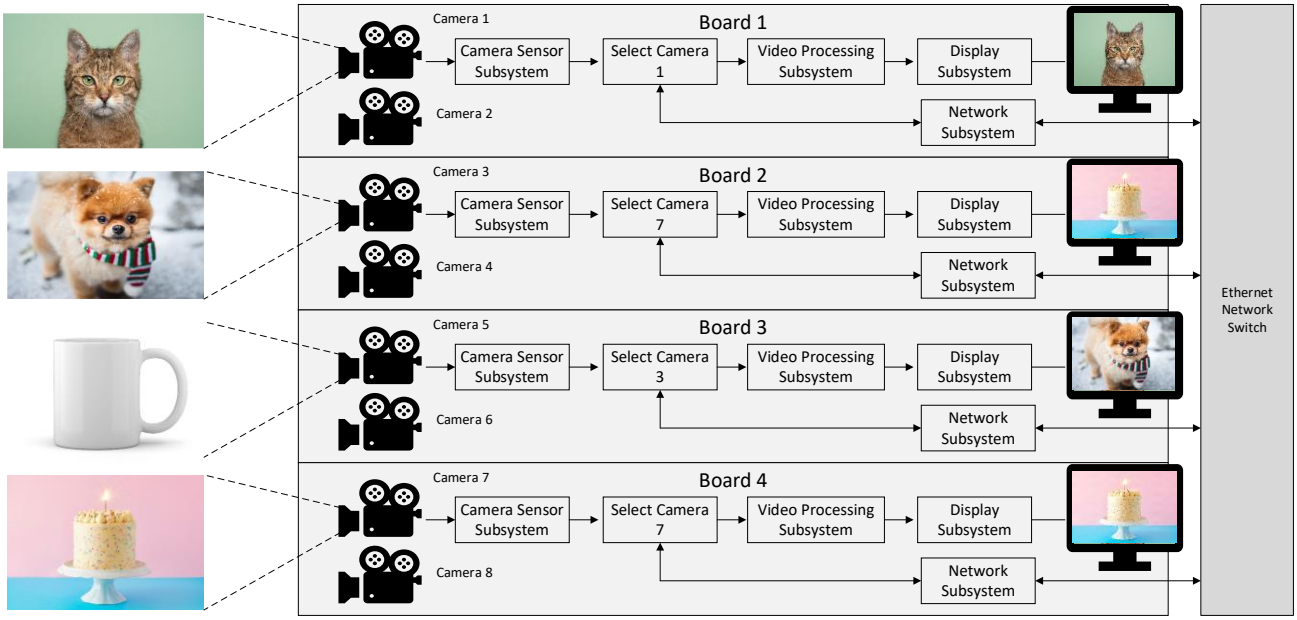


Figure 1.1. Overview of MIPI CSI-2 to Ethernet Reference Design

1.1. Quick Facts

Download the reference design files from the Lattice reference design web page.

Table 1.1. Summary of the Reference Design

General	Target Devices	Avant-AT-X
	Source code format	Verilog and System Verilog
Simulation	Functional simulation	Performed for reference design only (refer to the Simulating Reference Design section).
	Timing simulation	Not performed
	Testbench	Available
	Testbench format	System Verilog
Software Requirements	Software tool and version	• Lattice Radiant™ software version 2025.1 • QuestaSim™ software 2024.1
	IP version (if applicable)	• MIPI CSI-2 D-PHY Receiver IP v2.0.0 • Byte-to-Pixel Converter IP v1.9.1 • Debayer IP v1.2.2 • Automatic White Balance (AWB) IP v1.3.0 • Color Correction Matrix (CCM) IP v1.2.1 • PLL IP v2.6.1 • ROM IP v2.5.0 • Video frame buffer IP v1.3.0 • Memory Controller IP v2.6.0

		• DP RAM IP v2.4.0 • DC FIFO IP v2.3.0 • 10G Ethernet v3.3.0 • 25G Ethernet v2.2.0
Hardware Requirements	Board	• Avant-X Versa Board • HDMI-FMC • Sony IMX258 camera module
	Cable	• 10 Gb SFP+ optical transceiver module • 25 Gb SFP28 optical transceiver module • Optical fiber cable • HDMI cable

1.2. Features

Key features of the MIPI CSI-2 to Ethernet reference design include:

- Supports up to two camera sensors per Lattice Avant Versa board
- Supports 10 Gbps and 25 Gbps Ethernet operating speed
- HDMI display output interface
- Video format:
  - 4K video resolution at 30 frames per second
  - 1080p video resolution at 60 frames per second
- Layer 2 AVTP

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 signals (asserted when value is logic 0)
- _*i* are input signals
- _*o* are output signals

2. Directory Structure and Files

The following figure shows the directory structure of the reference design.

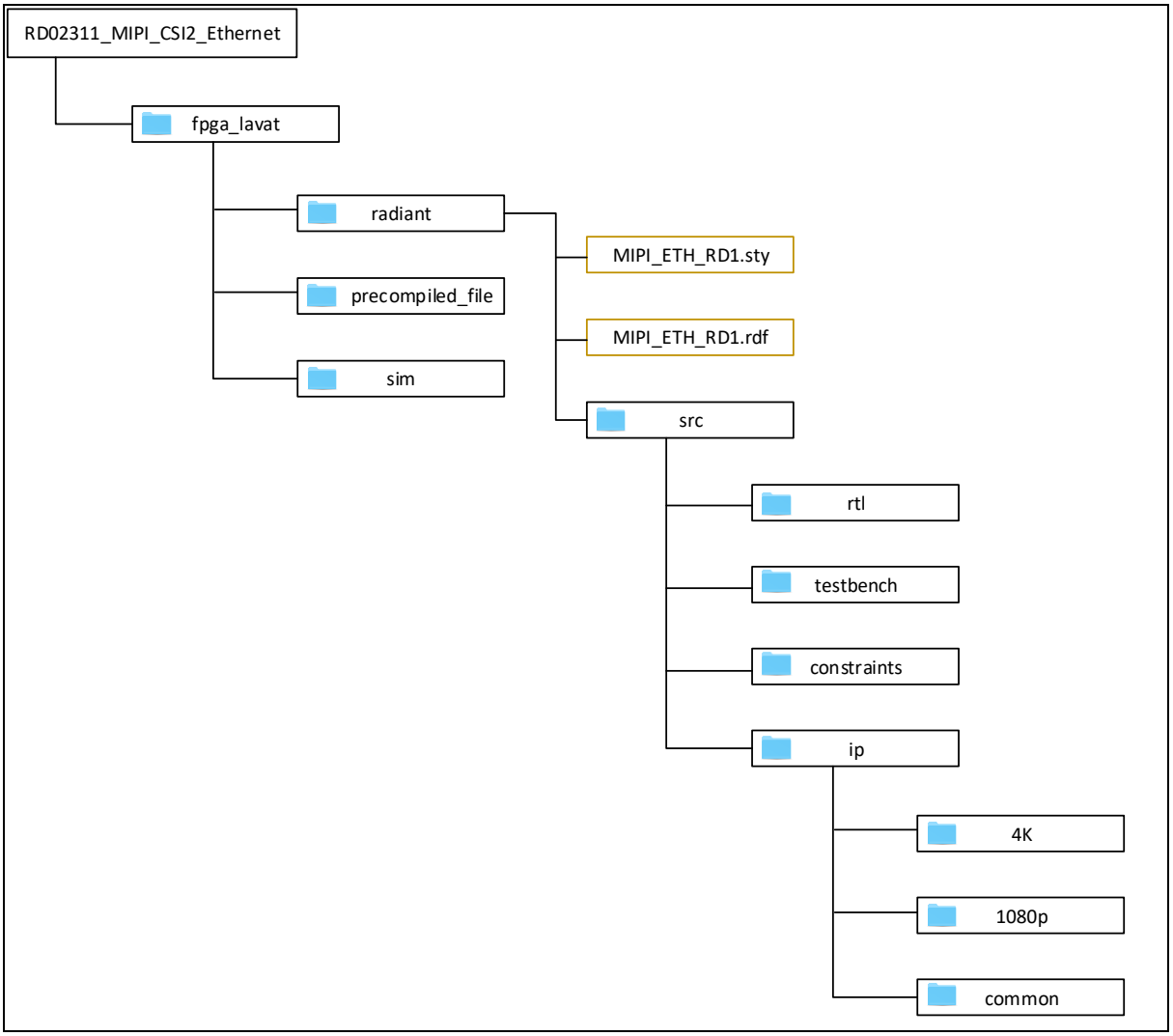


Figure 2.1. Directory Structure

The MIPI CSI-2 to Ethernet reference design is a comprehensive solution that integrates the functionalities of MIPI CSI-2 to HDMI with Ethernet connectivity. This combined approach leverages the strengths of each individual design to create a robust and versatile system.

The following table lists the files included in the reference design package.

Table 2.1. File List

Folders	Description
fpga_lavat	This folder contains the complete project design files, simulation waveform script (*wave.do*), and precompiled bitstream files.
radiant	This folder includes the *project.rdf* file and a src directory where all design-related collateral is organized.
precompiled_file	This folder stores precompiled bitstreams for both 4K and 1080p resolutions supporting 10 Gb and 25 Gb configurations.
sim	This folder contains the wave.do simulation script intended to be launched using the QuestaSim software.
src	This folder contains RTL modules, testbench files, constraints files, and IP directories.
ip	This folder comprises three subdirectories with IP cores categorized by resolution and shared/common IP cores used across both resolution implementations.

3. Functional Description

This section describes the architecture of the MIPI CSI-2 to Ethernet reference design. The system architecture of the MIPI CSI-2 to Ethernet reference design involves several key subsystems working together seamlessly.

Each Lattice Avant Versa board is equipped with two camera sensors that capture video data, which is initially processed through the MIPI CSI-2 D-PHY Rx intellectual property (IP). This IP receives the raw video input from the camera sensors and converts it to pixel format for processing by other subsystems.

The video data in pixel format is then packetized as AVTP packets by the Video Transport Subsystem and transmitted over the Ethernet network. The AVTP that is designed for real-time audio and video streaming, ensures low latency and synchronized delivery of media streams. This protocol is crucial for maintaining the integrity and timing of the video data as it travels through the network.

The Video Transport Subsystem plays a pivotal role in reference design. It not only facilitates the transmission of video data but also allows for the selection of which camera feed to display, which is controlled via a push button on the Lattice Avant Versa board.

The Network Subsystem manages network-related aspects to ensure seamless data transmission supporting high-speed connectivity with 10 Gbps and 25 Gbps Ethernet operating speeds, while the Video Processing Subsystem performs various tasks to enhance the video quality and prepare it for display. Finally, the video stream is routed to the Display Subsystem, which converts the video data into a format suitable for display on HDMI-compatible devices. Supporting high-definition resolutions, including 1080p and 4K, as illustrated in Figure 3.1. The Display Subsystem handles tasks such as frame rate conversion and other necessary adjustments to match the device display capabilities.

This comprehensive flow ensures that the video captured by the camera sensors is accurately processed and displayed with high quality.

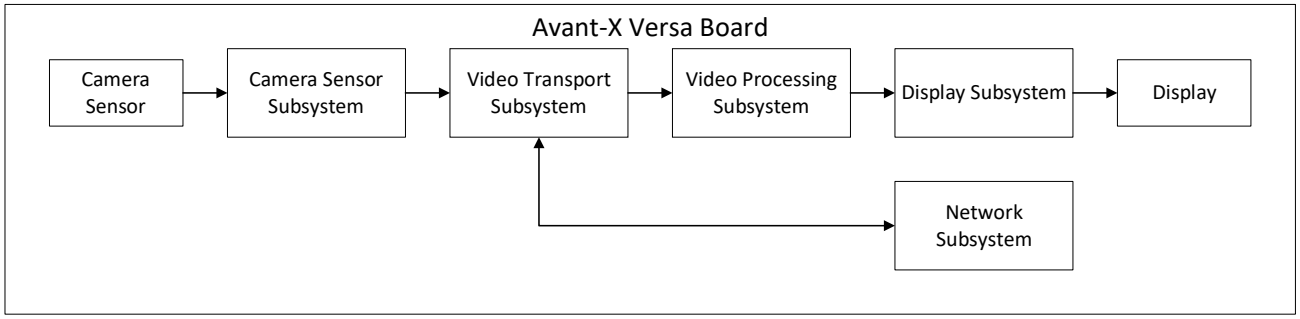


Figure 3.1. Reference Design Block Diagram

3.1. Transport Protocols

The AVTP is utilized to carry video data and transmit it over Ethernet. Defined in IEEE 1722-2016, the AVTP serves as the transport protocol within the Audio Video Bridging (AVB) profile of the Time Sensitive Network (TSN).

The primary objective of this reference design is to demonstrate the transport of video data over Ethernet, with AVTP selected as the transport protocol for this purpose. Therefore, full compliance with AVTP and AVB standards is beyond the scope of this reference design. For example, time synchronization with gPTP is not supported. You may leverage this reference design to expand its functionality for full compliance with AVB or modify the transport protocol for other video-to-Ethernet applications.

3.1.1. Frame Size and Bandwidth Calculations

IEEE 1722 Annex C specify that AVTP frames are encapsulated with Ethernet frame with VLAN tag. The calculation for required frame size and bandwidth of transporting 4K video over Ethernet are described in this section.

General frame format of AVTP frame and the size of each field are shown in the following table.

Table 3.1. General AVTP Frame Format

Field	Size (Bytes)
Ethernet Header	14
VLAN Tag	4
AVTP Header	32
Video Payload	Variable
FCS	4

Let:

MTU = 1,504 Bytes (Q-tagged frames – from IEEE 802.3 Clause 3.2.7)
Pixel Size = 1 Byte (for RAW8)
4K Resolution : 3,840 x 2,160
Frame Rate = 30 Frame per Second

Calculation:

Headers Size = (VLAN Tag + AVTP Header)
= (4 + 32)
= 36 Bytes

Note: Ethernet header and FCS are not part of MTU

Available size for video data per Ethernet frame = MTU – Headers size
= 1,504 – 36
= 1,468 Bytes

Maximum number of pixels per Ethernet frame = Round down (Max available video data size / pixel size)
= Round down (1,468 / 1)
= 1,468 pixels

Number of Ethernet frames per video line = Round Up (Pixels per video line / Max pixel per Ethernet frame)
= Round up (3,840 / 1468)
= Round up (2.6158…)
= 3 Ethernet frames

Number of pixels per Ethernet frame = 3,840 / 3
= 1,280 pixels

Ethernet frame length = Ethernet header + Headers size + (Pixels per Ethernet frame * pixel size) + FCS
= 14 + 36 + (1,280 * 1) + 4
= 1,334 Bytes

Number of Ethernet frame per video frame = (Lines per video frame) * (Ethernet frame per line)

= 2,160 * 3
= 6480 Ethernet frames

Ethernet bandwidth consumed = (Preamble + Ethernet packet length + IPG) * (Packets per frame) * (Frame rate) * (Byte-to-bit conversion)
= (8 + 1334 + 12) * 6480 * 30 * 8
= **2.1057408 Gbps**

3.1.2. Frame Format

Figure 3.2 and Figure 3.3 show the format of AVTP frame in AXI-S data bus for 25G Ethernet, transmitting RAW8 video data in RGB Bayer format. Figure 3.4 and Figure 3.5 presents the corresponding AVTP frame format for the 10G Ethernet AXI-S data bus.

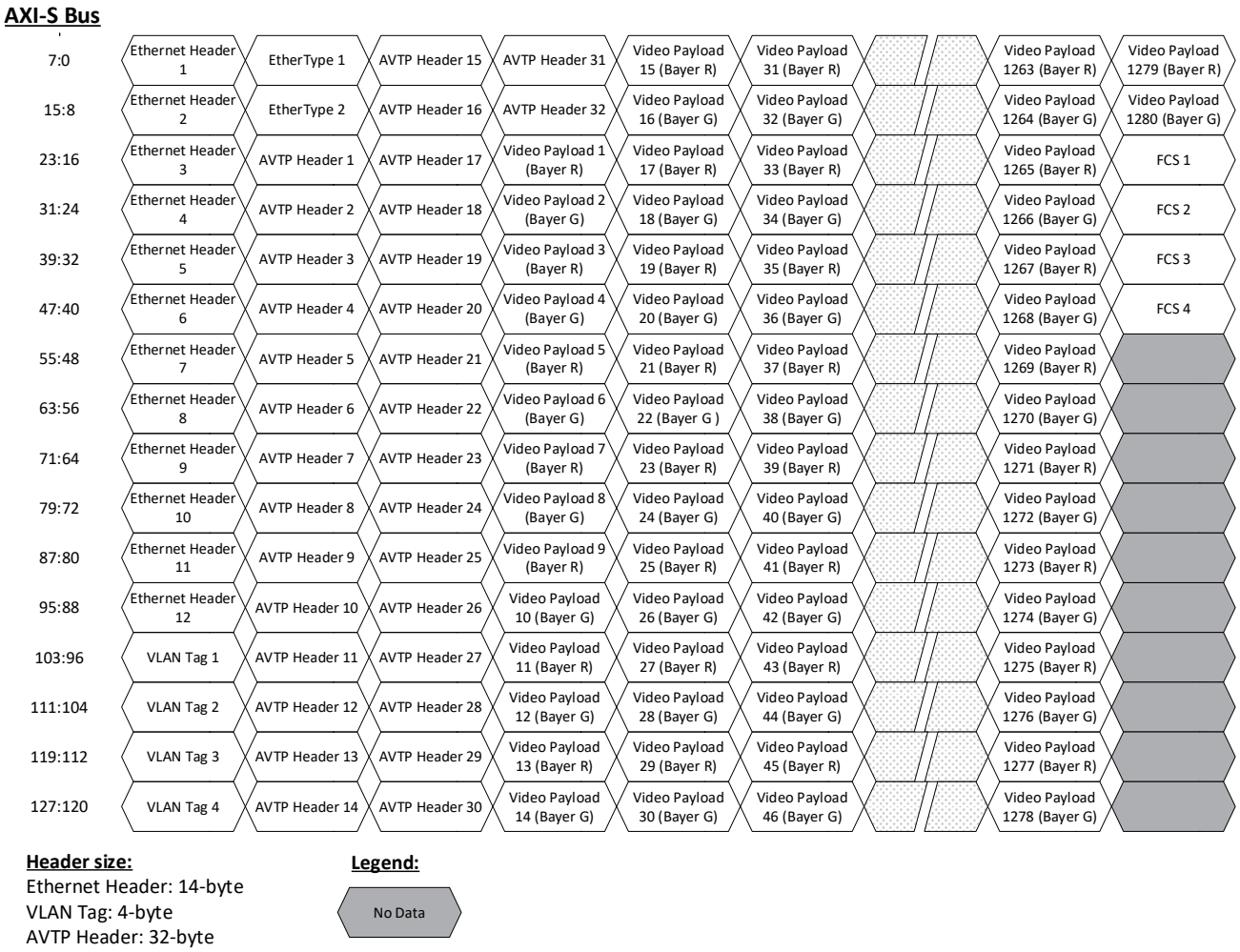


Figure 3.2. AVTP Frame in AXI-S Data Bus for 25G Ethernet (Odd Line Number)

AXI-S Bus

7:0	Ethernet Header 1	EtherType 1	AVTP Header 15	AVTP Header 31	Video Payload 15 (Bayer G)	Video Payload 31 (Bayer G)		Video Payload 1263 (Bayer G)	Video Payload 1279 (Bayer G)
15:8	Ethernet Header 2	EtherType 2	AVTP Header 16	AVTP Header 32	Video Payload 16 (Bayer B)	Video Payload 32 (Bayer B)		Video Payload 1264 (Bayer B)	Video Payload 1280 (Bayer B)
23:16	Ethernet Header 3	AVTP Header 1	AVTP Header 17	Video Payload 1 (Bayer G)	Video Payload 17 (Bayer G)	Video Payload 33 (Bayer G)		Video Payload 1265 (Bayer G)	FCS 1
31:24	Ethernet Header 4	AVTP Header 2	AVTP Header 18	Video Payload 2 (Bayer B)	Video Payload 18 (Bayer B)	Video Payload 34 (Bayer B)		Video Payload 1266 (Bayer B)	FCS 2
39:32	Ethernet Header 5	AVTP Header 3	AVTP Header 19	Video Payload 3 (Bayer G)	Video Payload 19 (Bayer G)	Video Payload 35 (Bayer G)		Video Payload 1267 (Bayer G)	FCS 3
47:40	Ethernet Header 6	AVTP Header 4	AVTP Header 20	Video Payload 4 (Bayer B)	Video Payload 20 (Bayer B)	Video Payload 36 (Bayer B)		Video Payload 1268 (Bayer B)	FCS 4
55:48	Ethernet Header 7	AVTP Header 5	AVTP Header 21	Video Payload 5 (Bayer G)	Video Payload 21 (Bayer G)	Video Payload 37 (Bayer G)		Video Payload 1269 (Bayer G)	
63:56	Ethernet Header 8	AVTP Header 6	AVTP Header 22	Video Payload 6 (Bayer B)	Video Payload 22 (Bayer B)	Video Payload 38 (Bayer B)		Video Payload 1270 (Bayer B)	
71:64	Ethernet Header 9	AVTP Header 7	AVTP Header 23	Video Payload 7 (Bayer G)	Video Payload 23 (Bayer G)	Video Payload 39 (Bayer G)		Video Payload 1271 (Bayer G)	
79:72	Ethernet Header 10	AVTP Header 8	AVTP Header 24	Video Payload 8 (Bayer B)	Video Payload 24 (Bayer B)	Video Payload 40 (Bayer B)		Video Payload 1272 (Bayer B)	
87:80	Ethernet Header 11	AVTP Header 9	AVTP Header 25	Video Payload 9 (Bayer G)	Video Payload 25 (Bayer G)	Video Payload 41 (Bayer G)		Video Payload 1273 (Bayer G)	
95:88	Ethernet Header 12	AVTP Header 10	AVTP Header 26	Video Payload 10 (Bayer B)	Video Payload 26 (Bayer B)	Video Payload 42 (Bayer B)		Video Payload 1274 (Bayer B)	
103:96	VLAN Tag 1	AVTP Header 11	AVTP Header 27	Video Payload 11 (Bayer G)	Video Payload 27 (Bayer G)	Video Payload 43 (Bayer G)		Video Payload 1275 (Bayer G)	
111:104	VLAN Tag 2	AVTP Header 12	AVTP Header 28	Video Payload 12 (Bayer B)	Video Payload 28 (Bayer B)	Video Payload 44 (Bayer B)		Video Payload 1276 (Bayer B)	
119:112	VLAN Tag 3	AVTP Header 13	AVTP Header 29	Video Payload 13 (Bayer G)	Video Payload 29 (Bayer G)	Video Payload 45 (Bayer G)		Video Payload 1277 (Bayer G)	
127:120	VLAN Tag 4	AVTP Header 14	AVTP Header 30	Video Payload 14 (Bayer B)	Video Payload 30 (Bayer B)	Video Payload 46 (Bayer B)		Video Payload 1278 (Bayer B)	

Header size:
Ethernet Header: 14-byte
VLAN Tag: 4-byte
AVTP Header: 32-byte

Legend:

No Data

Figure 3.3. AVTP Frame in AXI-S Data Bus for 25G Ethernet (Even Line Number)

AXI-S Bus

7:0	Ethernet Header 1	Ethernet Header 9	EtherType 1	AVTP Header 7	AVTP Header 15	AVTP Header 23	AVTP Header 31	Video Payload 1279 (Bayer R)
15:8	Ethernet Header 2	Ethernet Header 10	EtherType 2	AVTP Header 8	AVTP Header 16	AVTP Header 24	AVTP Header 32	Video Payload 1280 (Bayer G)
23:16	Ethernet Header 3	Ethernet Header 11	AVTP Header 1	AVTP Header 9	AVTP Header 17	AVTP Header 25	Video Payload 1 (Bayer R)	FCS 1
31:24	Ethernet Header 4	Ethernet Header 12	AVTP Header 2	AVTP Header 10	AVTP Header 18	AVTP Header 26	Video Payload 2 (Bayer G)	FCS 2
39:32	Ethernet Header 5	VLAN Tag 1	AVTP Header 3	AVTP Header 11	AVTP Header 19	AVTP Header 27	Video Payload 3 (Bayer R)	FCS 3
47:40	Ethernet Header 6	VLAN Tag 2	AVTP Header 4	AVTP Header 12	AVTP Header 20	AVTP Header 28	Video Payload 4 (Bayer G)	FCS 4
55:48	Ethernet Header 7	VLAN Tag 3	AVTP Header 5	AVTP Header 13	AVTP Header 21	AVTP Header 29	Video Payload 5 (Bayer R)	
63:56	Ethernet Header 8	VLAN Tag 4	AVTP Header 6	AVTP Header 14	AVTP Header 22	AVTP Header 30	Video Payload 6 (Bayer G)	

Header:
Ethernet Header: 14-byte
VLAN Tag: 4-byte
AVTP Header: 32-byte

Legend:

No Data

Figure 3.4. AVTP Frame in AXI-S Data Bus for 10G Ethernet (Odd Line Number)

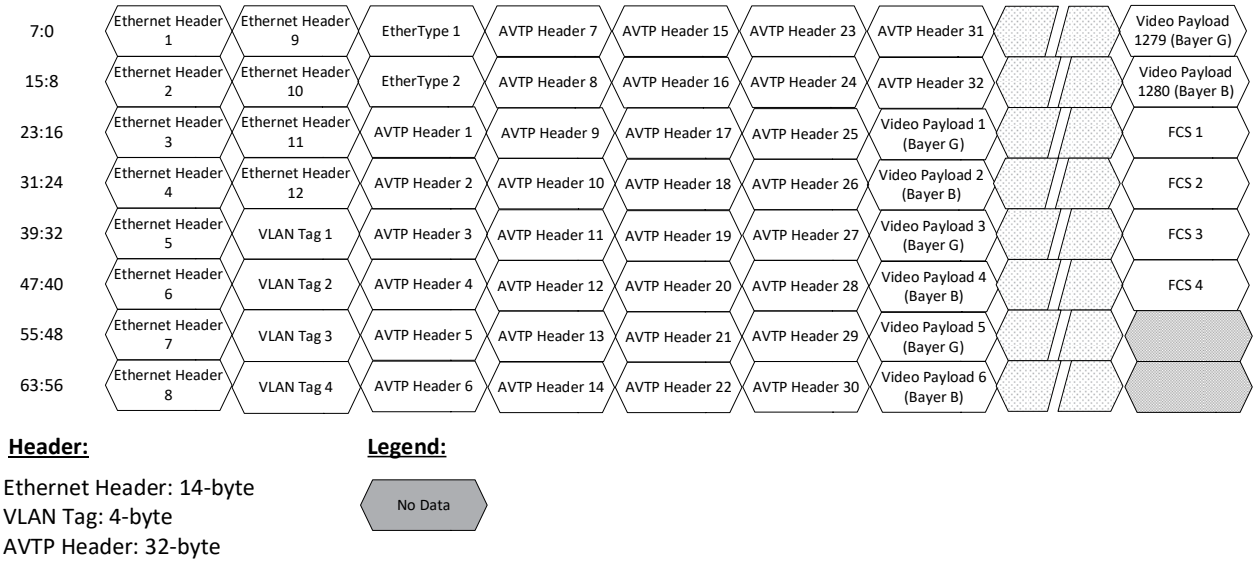


Figure 3.5. AVTP Frame in AXI-S Data Bus for 10G Ethernet (Even Line Number)

Figure 3.6 shows the detailed format of AVTP header for Raw Video Format. Refer to Table 3.3. AVTP Header Format Values in the Reference Design for description and value of each field. Refer to IEEE 1722 and IEEE 802.1Q specification for details of each field.

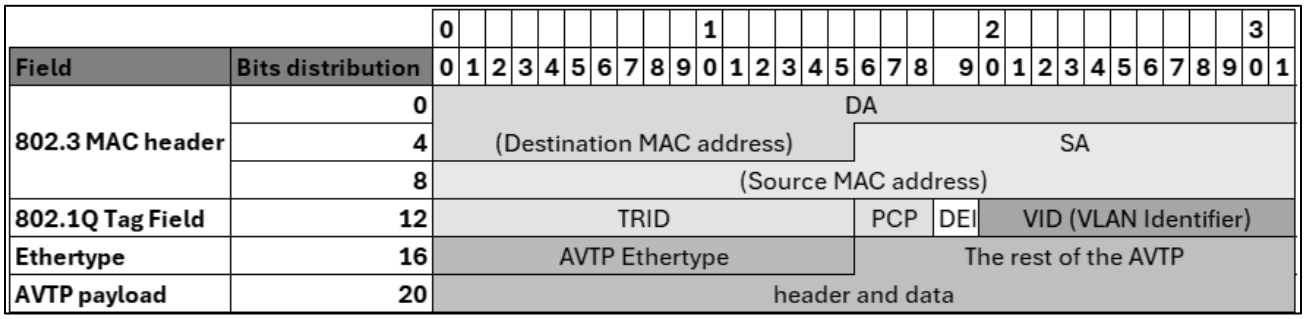


Figure 3.6. AVTP Frame Format – Ethernet Header

Table 3.2. Ethernet Header Format Values in Reference Design

Field	Width (Bit)	Specification	Value	Description
DA (destination MAC address)	48	IEEE 1722 Annex B.4, Annex C.2.1.2 and Annex D	91:E0:F0:00:FE:0n	n is the stream ID. Multicast.
SA (source MAC address)	48	IEEE 1722 Annex C.2.1.3	00:D0:BD:17:22:0m	m is the board ID. Unicast.
TPID (tagged protocol identifier)	16	IEEE 1722 Annex C.2.1.4	0x8100	VLAN tag EthernetType value.
PCP (priority code point)	3	IEEE 1722 Annex C.2.1.5 IEEE 802.1Q Clause 35.2.2.9.3	3	SR class A default priority.

Field	Width (Bit)	Specification	Value	Description
DEI (drop_eligible)	1	IEEE 1722 Annex C.2.1.6 IEEE 802.1Q Clause 9.6	0	—
VID (VLAN ID)	12	IEEE 1722 Annex C.2.1.7 IEEE 802.1Q Clause 35.2.1.4	2	Stream related traffic.
AVTP EtherType	16	IEEE 1722 Annex C.2.1.8, Clause 4.4.2	0x22F0	AVTP EtherType value.

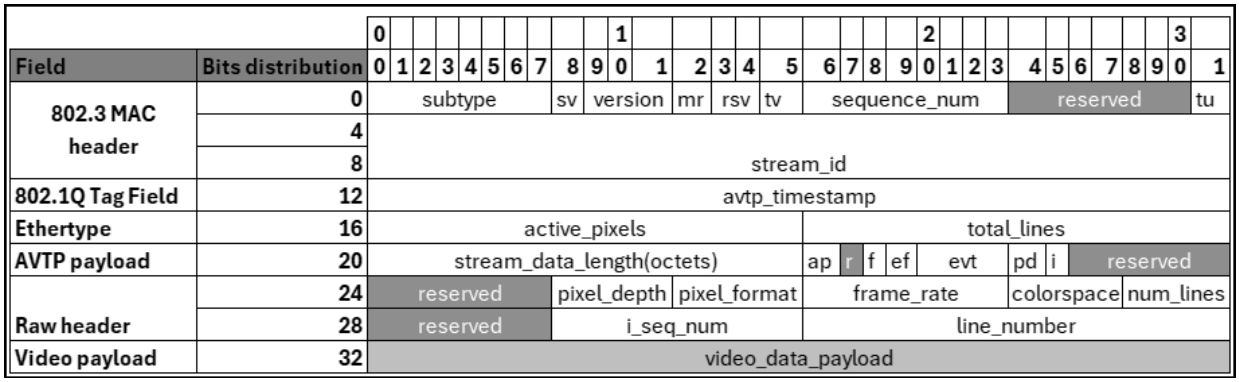


Figure 3.7. AVTP Frame Format – AVTP Header

Table 3.3. AVTP Header Format Values in the Reference Design

Field	Width (Bit)	Specification	Value	Description
subtype	8	IEEE 1722 Clause 4.4.3.2	0x07	Raw Video Format (RVF).
sv (stream_id valid)	1	IEEE 1722 Clause 4.4.4.2.	1	Stream ID field is valid.
version	3	IEEE 1722 Clause 4.4.3.4.	0	—
mr (media clock restart)	1	IEEE 1722 Clause 4.4.4.3.	0	No change in the source of media clock in this reference design.
tv (avtp_timestamp_valid)	1	IEEE 1722 Clause 4.4.4.5 and Clause 12.2.2.	1 = first frame of the video line. 0 = remaining frames.	gPTP clock is not supported.
sequence_num (sequence number)	8	IEEE 1722 Clause 4.4.4.6.	Value incremented for every AVTP frame sent.	Value set but not used in this reference design.
tu (timestamp uncertain)	1	IEEE 1722 Clause 4.4.4.7.	1	gPTP clock is not supported.
stream_id	64	IEEE 1722 Clause 4.4.4.8 IEEE 802.1Q Clause 35.2.2.8.2	0x00D0BD17220m_000n	*m* is the board ID. *n* is the stream ID.
avtp_timestamp	32	in IEEE 1722 Clause 4.4.4.9 and Clause 12.2.2.	0	Time synchronization is not supported.
active_pixels	16	IEEE 1722 Clause 12.2.3.	4K: 3840 1080p: 1920	Number of horizontal pixels per video line.

Field	Width (Bit)	Specification	Value	Description
total_lines	16	IEEE 1722 Clause 12.2.4.	4K: 2160 1080p: 1280	Number of lines per video frame.
stream_data_length	16	IEEE 1722 Clause 4.4.4.10.	4K: 1280 1080p: 960	Video data payload size in octets.
ap (active pixels)	1	IEEE 1722 Clause 12.2.5.	1	No blanking in the video payload.
r (reserved)	1		0	—
f (field)	1	IEEE 1722 Clause 12.2.6.	0	Always 0 for progressive video.
ef (end frame)	1	IEEE 1722 Clause 12.2.7.	1 = last Ethernet frame of the video frame. 0 = other Ethernet frames.	End of video frame.
evt	4	IEEE 1722 Clause 12.2.8.	0	Reserved for upper-level protocols.
pd (pull-down)	1	IEEE 1722 Clause 12.2.9.	0	30/60 Hz video frame rate is used in this design.
i (interlaced)	1	IEEE 1722 Clause 12.2.10.	0	0 for progressive video.
pixel_depth	4	IEEE 1722 Clause 12.2.11.	0x1	8 bits per pixel.
pixel_format	4	IEEE 1722 Clause 12.2.12.	0x9	RGGB Bayer pattern.
frame_rate	8	IEEE 1722 Clause 12.2.13.	30 fps: 0x15 60 fps: 0x18	Video frame rate.
colorspace	4	IEEE 1722 Clause 12.2.14.	0xF	User defined. Bayer pattern is not part of all listed colorspace.
num_lines	4	IEEE 1722 Clause 12.2.15 and 12.2.17.	0	Number of video lines within single AVTP packet. 0 for 4K and 1080p, which need three and two packets for single video line.
i_seq_num	8	IEEE 1722 Clause 12.2.16.	0 = first Ethernet frame of the video line Increment for subsequent Ethernet frame of same video line.	Number reset to 0 for first Ethernet frame of the next video line.
line_number	16	IEEE 1722 Clause 12.2.17.	1 = first line of video frame Increment for Ethernet frames belongs to subsequent video lines.	All Ethernet frames that belong to the same video line have identical value.
video_data_payload	0 to *n* octets	IEEE 1722 Clause 12.2.18, 12.2.19 and 12.2.20.	Contain pixel data from camera sensor in Bayer format.	—

3.2. Design Components

The MIPI CSI-2 to Ethernet reference design includes the following subsystems:

- Camera Sensor Subsystem
- Video Transport Subsystem
- Video Processing Subsystem
- Display Subsystem
- Network Subsystem

The following figure shows the block diagram and submodules of each subsystem. The description of each subsystem and submodules is described in subsequent sections.

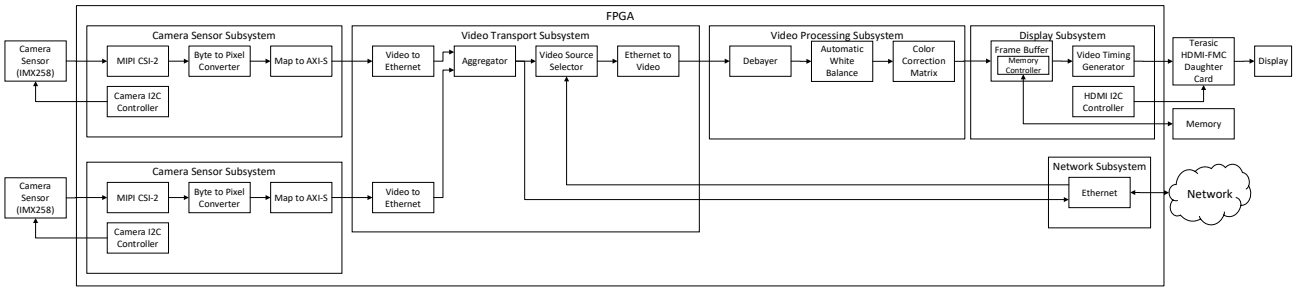


Figure 3.8. MIPI CSI-2 to Ethernet Reference Design Components Block Diagram

3.2.1. Camera Sensor Subsystem

The Camera Sensor Subsystem captures video images using a camera sensor. It converts the captured video data and sends it to other subsystems. This subsystem uses designs from the Lattice MIPI CSI-2 to HDMI reference design. For more details, refer to the MIPI CSI-2 to HDMI Reference Design User Guide.

The following lists brief explanations of each part within this subsystem in Figure 3.9:

- **Camera Sensor**: The subsystem captures video images using the IMX258 camera sensor, which sends the video data to an FPGA using the MIPI CSI-2 interface.
- **MIPI CSI-2 IP**: This component receives video data and sends it to the Byte-to-Pixel Converter IP, which converts the data into pixel format.
- **Map to AXI-S Block**: The pixel data is then converted to the AXI-S interface format by this block. All subsequent data transactions within this subsystem use the AXI-S interface.
- **I2C Controller**: This controller is used to configure the camera sensor.

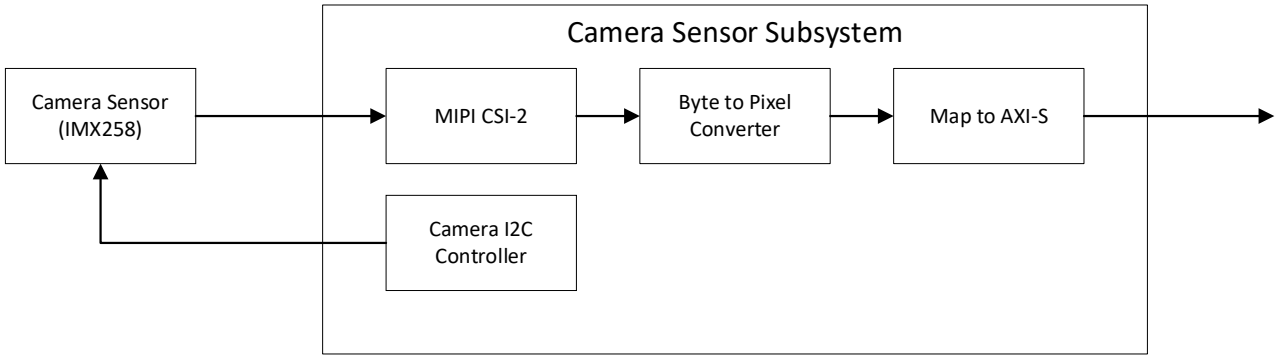


Figure 3.9. Camera Sensor Subsystem Block Diagram

3.2.2. Video Processing Subsystem

The Video Processing Subsystem uses designs from the Lattice MIPI CSI-2 to HDMI reference design. For more details, refer to the MIPI CSI-2 to HDMI Reference Design User Guide.

Within the subsystem shown in Figure 3.10, the Debayer IP converts RAW8 data (raw image data) into RGB888 format (standard color image data). Following this, the Automatic White Balance IP and the Color Correction Matrix IP work together to enhance the quality of the video images. These enhancements ensure that the colors are accurate, and the overall image quality is improved before the video data is sent to other subsystems.

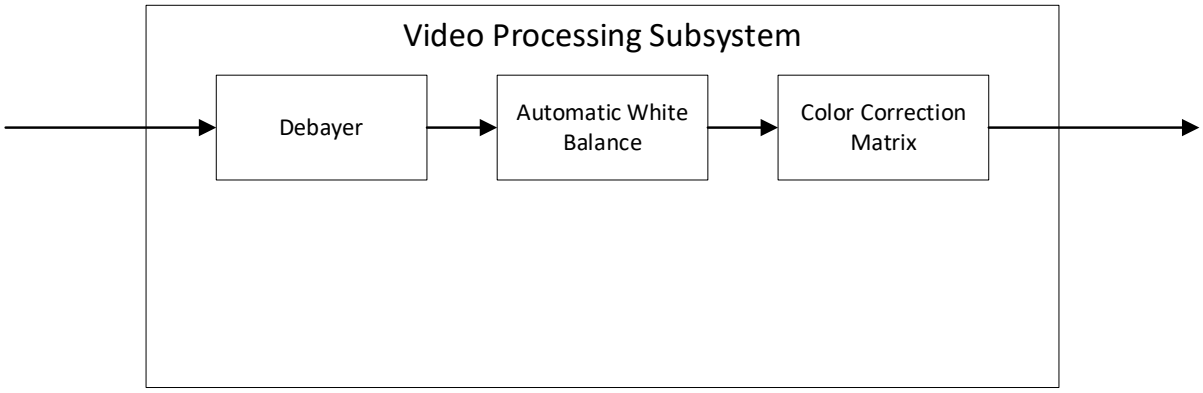


Figure 3.10. Video Processing Subsystem Block Diagram

3.2.3. Display Subsystem

The Display Subsystem is part of the MIPI CSI-2 to HDMI reference design. It receives video images in RGB888 format from the Video Processing Subsystem and sends these images to the display via the Terasic HDMI-FMC daughter card.

The following lists brief explanations of each part within the subsystem in Figure 3.11:

- **Video Frame Buffer**: By storing video frames temporarily, the Frame Buffer ensures that the video output is continuous and smooth. This prevents any interruptions or glitches that could degrade the viewing experience.
- **Video Timing Generator**: This component generates precise timing signals for the HDMI output, which is essential for synchronizing the video display. Proper timing ensures that the video is displayed correctly without any distortions or misalignments.
- **I2C Controller**: This controller configures the HDMI-FMC daughter card to ensure that the hardware settings are optimized for the best possible video output.

Overall, these components work together to ensure that the video images are displayed smoothly and accurately, which preserves the quality of the video processed by the previous subsystems.

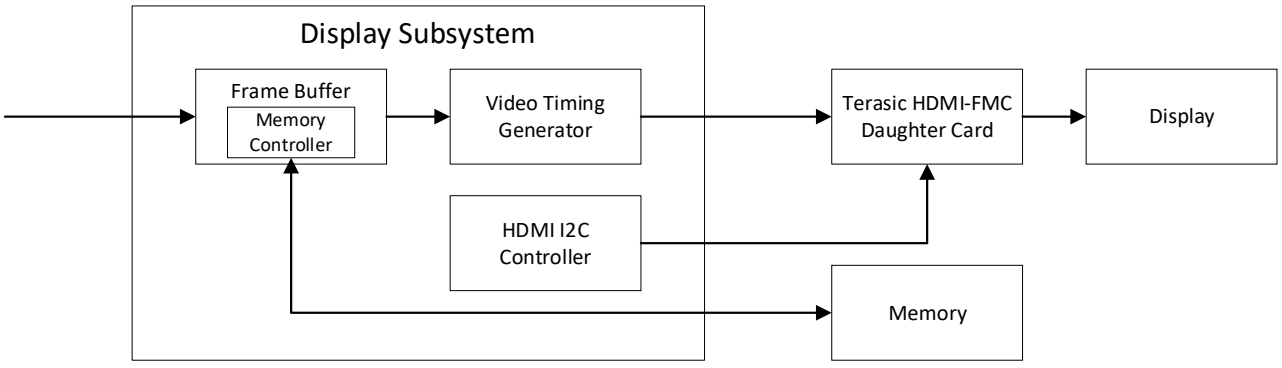


Figure 3.11. Display Subsystem Block Diagram

3.2.4. Network Subsystem

The Network Subsystem is responsible for sending captured video data from the Video Transport Subsystem to the network and receiving video data from the network. It includes Ethernet Media Access Control (MAC) and Physical Layer (PHY) IP components, which handle the basic functions of network communication. These components support the first two layers of the OSI model:

- **Layer 1 (Physical Layer)**: This layer deals with the physical connection between devices, including the transmission and reception of raw data bits over a physical medium.
- **Layer 2 (Data Link Layer)**: This layer is responsible for node-to-node data transfer, error detection, and managing data frames between devices on the same network.

By supporting these functionalities, the Network Subsystem ensures that video data can be efficiently transmitted and received over the network, which maintains the integrity and quality of the video stream.

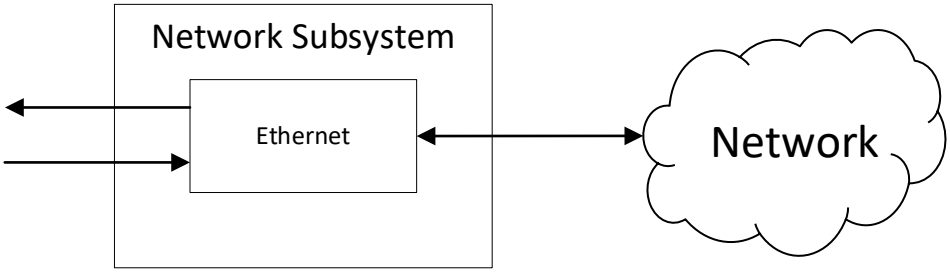


Figure 3.12. Network Subsystem Block Diagram

3.2.5. Video Transport Subsystem

The Video Transport Subsystem as shown in Figure 3.13 receives video frame data in RAW8 format from the Camera Sensor Subsystems. It then encapsulates this data into AVTP payloads. These AVTP frames are sent to both the Video Source Selector and the Network Subsystem.

An Aggregator with round-robin scheduling is used to multiplex the AVTP frames from multiple Camera Sensor Subsystems before forwarding them to the Video Source Selector or the Network Subsystem. This ensures that video data from multiple cameras is efficiently managed and transmitted.

The Video Source Selector receives video data from the Aggregator and from the network. It dropped all packets except AVTP frames with matching selected video stream ID.

Ethernet to Video decapsulates the AVTP frames from the user-selected source and forwards the video data to the Display Subsystem.

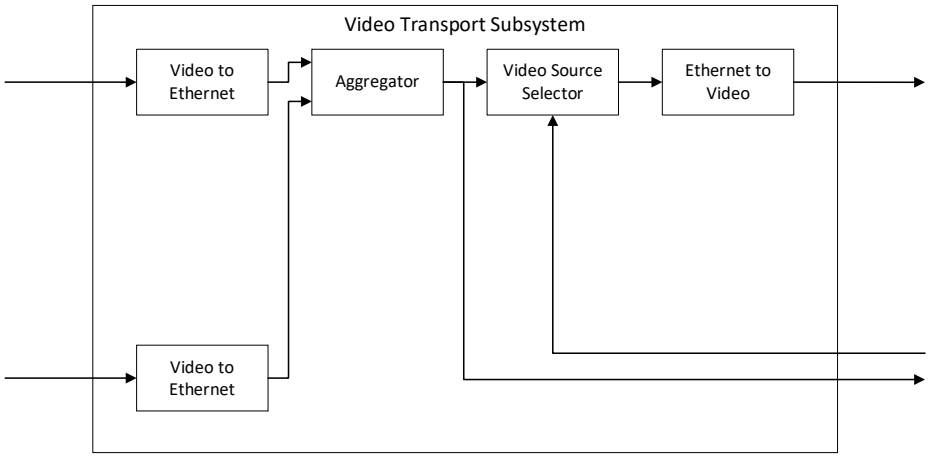


Figure 3.13. Video Transport Subsystem Block Diagram

Detailed architecture of the Video Switching Network Subsystem is shown in Figure 3.14 and explained further in the following sub-chapters of Video Transport Subsystem.

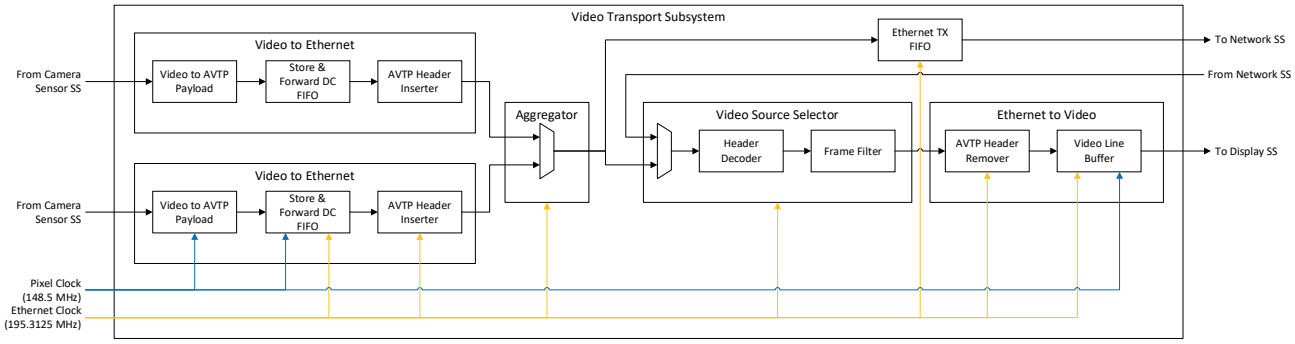


Figure 3.14. Video Switching Network Subsystem Detail Architecture

3.2.5.1. Video to AVTP Payload

Within the Video Transport Subsystem, the process begins with the Video to AVTP Payload module.

This module receives video pixel data in RAW8 format from the Camera Sensor Subsystem and forms the payload portion of the AVTP frame. Figure 3.15 and Figure 3.16 illustrate the input AXI-S data bus from the Camera Sensor Subsystem, which processes two pixels per cycle with the RGGB Bayer pattern.

Figure 3.17 shows the output AVTP frame for 25G Ethernet, where the video data is encapsulated in a 128-bit AXI-S data bus. The position of the first video data in the 128-bit data bus is adjusted based on the lengths of the Ethernet and AVTP headers. TUSER[2] signal indicates end of current video frame.

In contrast, for 10G data rate, video data is similarly encapsulated using a 64-bit AXI-Stream data bus, following the format shown in Figure 3.18.

As calculated in the Frame Size and Bandwidth Calculations section, each line of 4K video is encapsulated across three AVTP frames, with each AVTP frame carrying 1,280 pixels.

In the internal architecture of video to Ethernet, Figure 3.19 shows the input pixel data is concatenated into a wider data bus according to the AVTP_WIDTH parameter. After a cycle's worth of data is formed, it is forwarded downstream. When the expected number of pixel data has been received, an end-of-packet boundary indicated by TLAST is sent out.

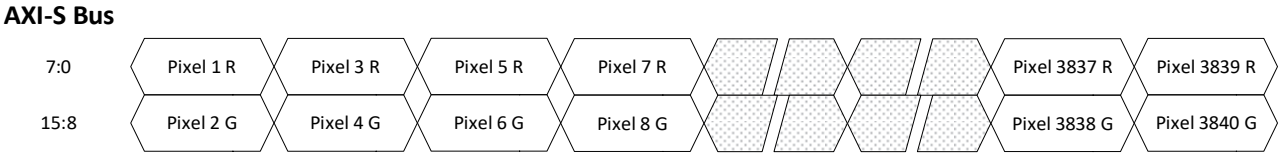


Figure 3.15. Video to AVTP Payload AXI-S Input Data (Odd Line Number)

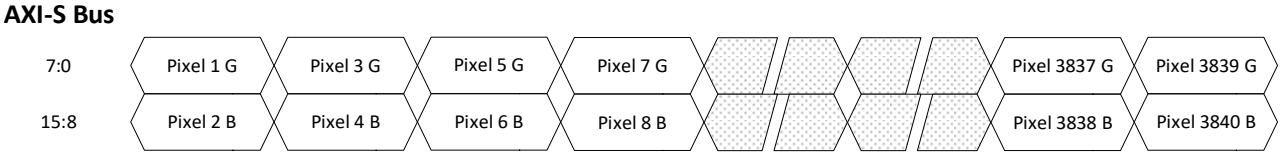


Figure 3.16. Video to AVTP Payload AXI-S Input Data (Even Line Number)

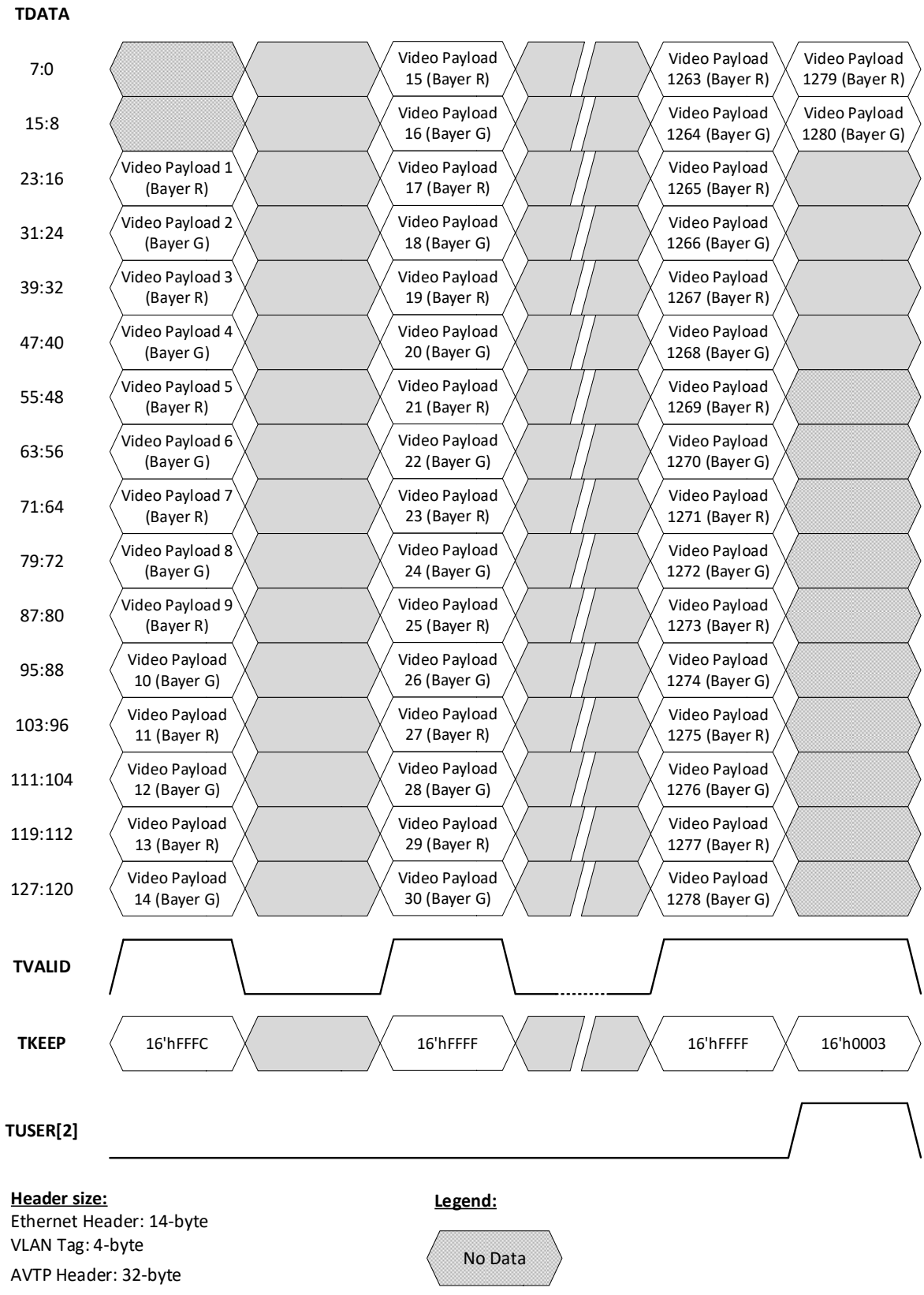


Figure 3.17. Video to AVTP Header AXI-S Output Data for 25G Ethernet (Odd Line Number)

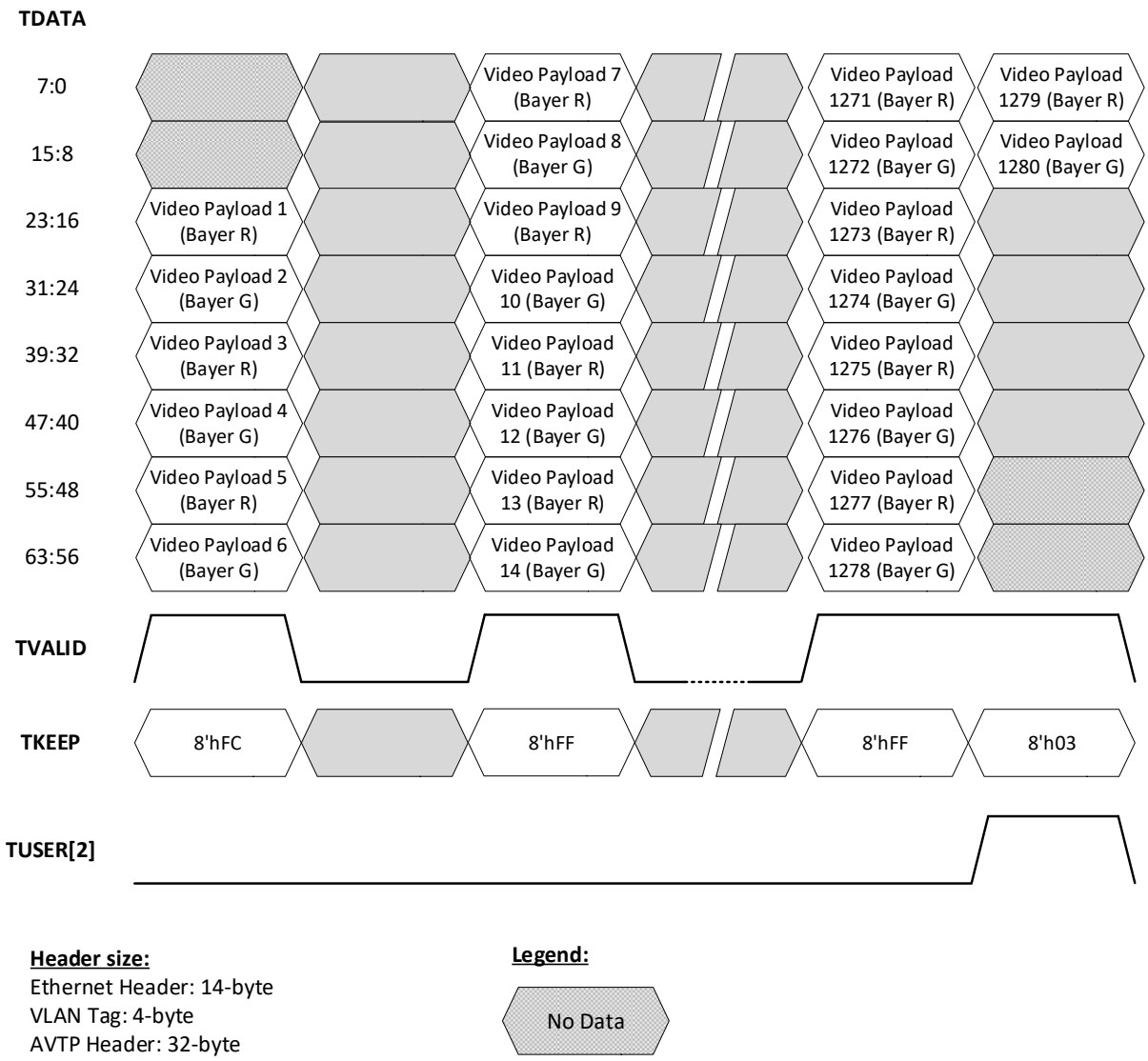


Figure 3.18. Video to AVTP Header AXI-S Output Data for 10G Ethernet (Odd Line Number)

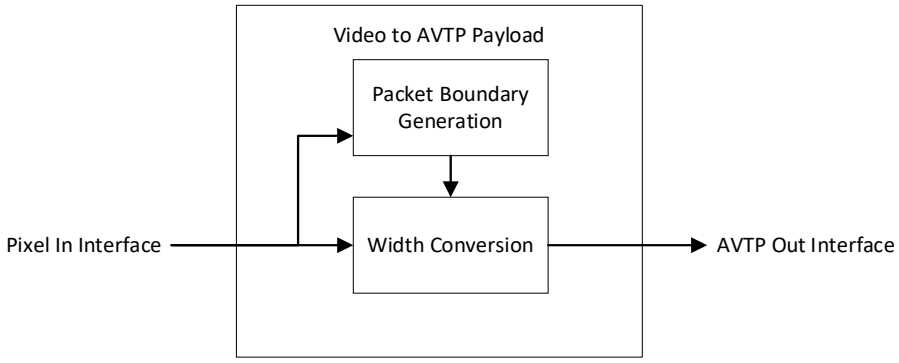


Figure 3.19. Video to Ethernet Encapsulation Internal Architecture

3.2.5.2. Store & Forward FIFO

The AVTP frame output from the Video-to-AVTP Payload module becomes valid once every 8 clock cycles in the Ethernet clock domain. This is due to the pixel data width (16-bit) is combined to match the Ethernet data width (25G: 128-bit, 10G: 64-bit). Additionally, due to the clock domain crossing from the slower pixel clock (4K: 148.5 MHz, 1080p: 74.25 MHz) to the faster Ethernet clock (25G: 195.3125 MHz, 10G: 156.25 MHz), the data valid gap increased further.

The Ethernet MAC requires continuous data in the transmit data path to prevent packet underflow. Therefore, the Ethernet frame data must be continuous before being forwarded to Ethernet MAC. The Store & Forward FIFO buffer stores the entire AVTP frame data from the start-of-packet to the end-of-packet and only provides valid output when the end-of-packet is received. This module also handles the clock domain crossing from the pixel clock domain (4K: 148.5 MHz, 1080p: 74.25 MHz) to the Ethernet clock domain (25G: 195.3125 MHz, 10G: 156.25MHz). Figure 3.20 and Figure 3.21 show the AXI-S output interface waveforms for 10G and 25G Ethernet, respectively.

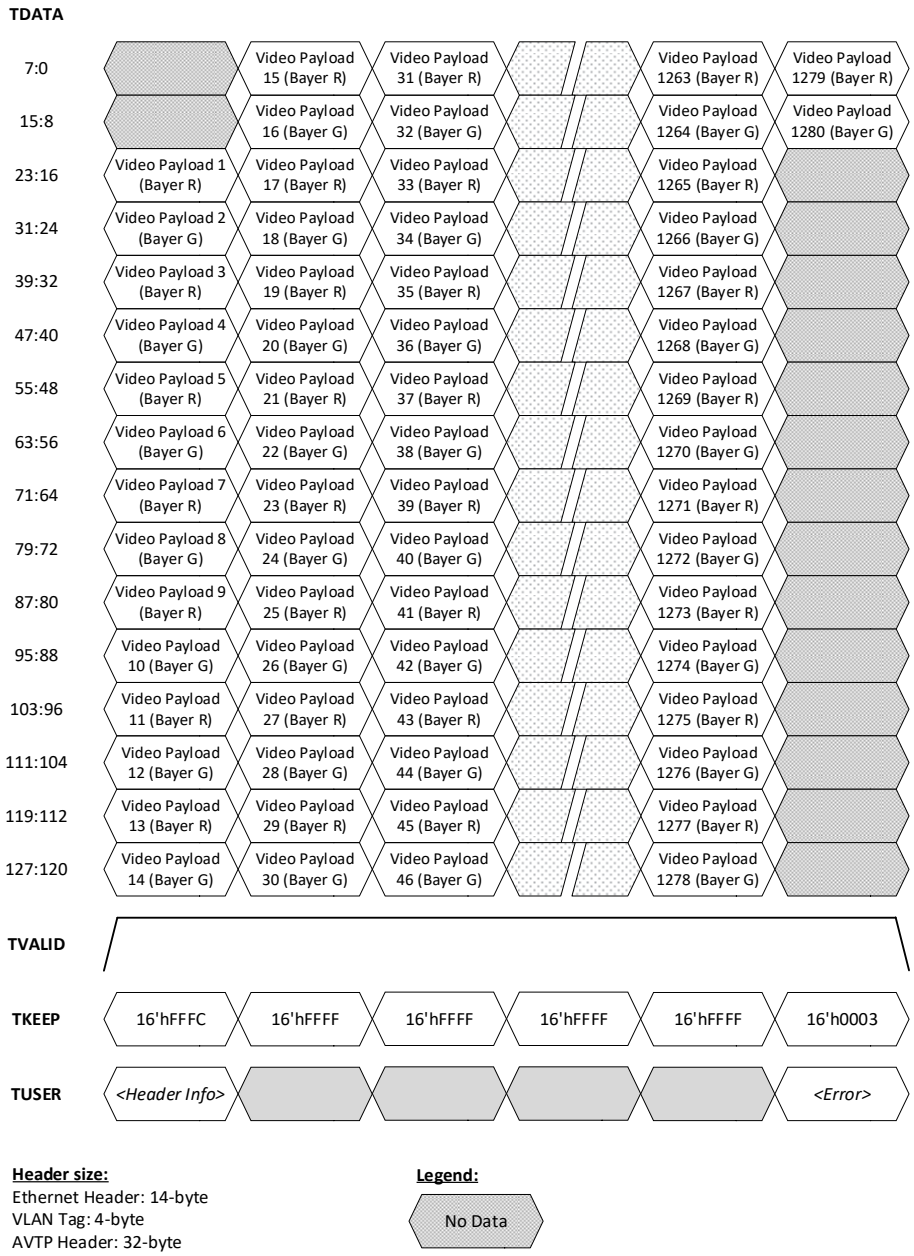


Figure 3.20. Store & Forward FIFO AXI-S Output Data for 25G Ethernet (Odd Line Number)

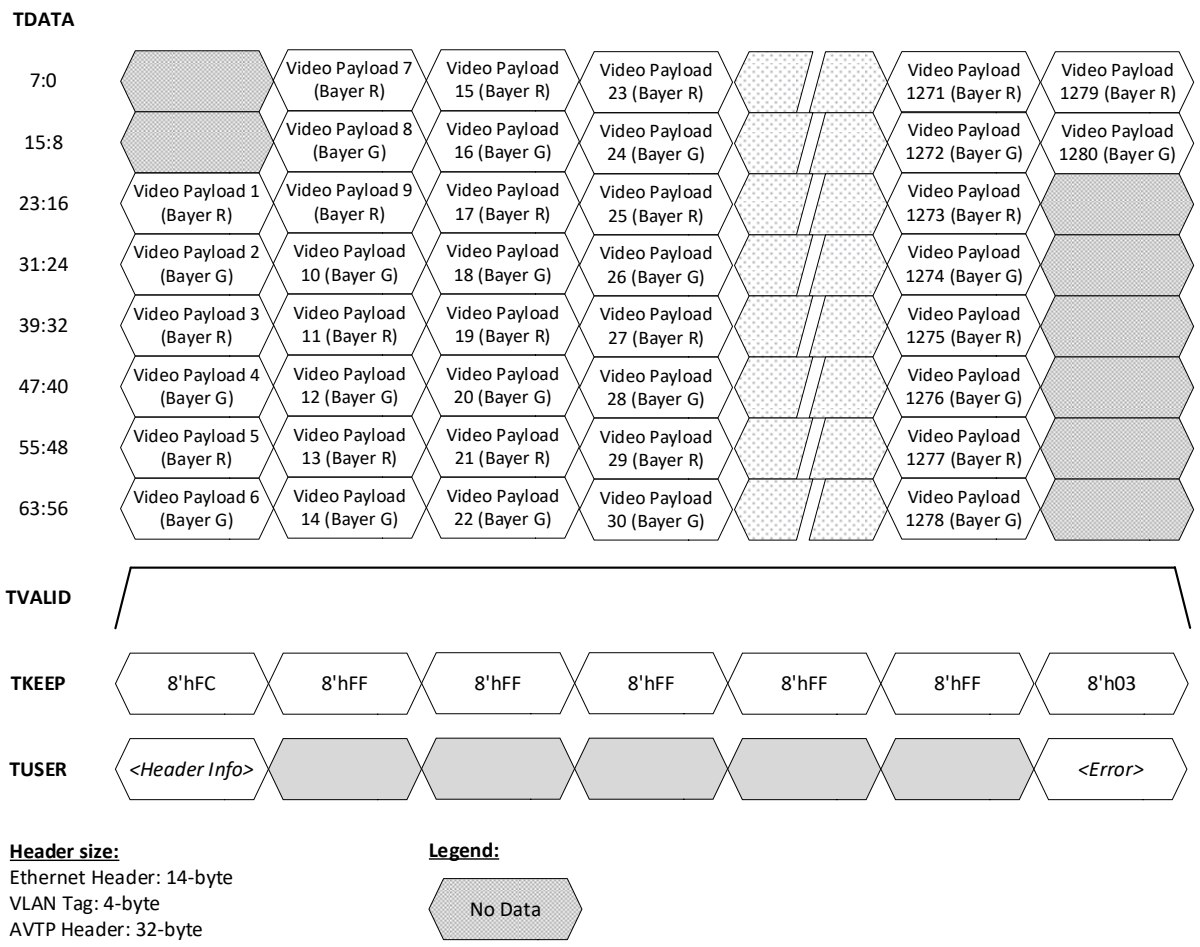


Figure 3.21. Store & Forward FIFO AXI-S Output Data for 10G Ethernet (Odd Line Number)

The AVTP header contains information indicating the end of the video frame, which is available only at the end of the packet from upstream, as shown by the TUSER[2] signal in Figure 3.17.

Figure 3.22 shows the module internal architecture that includes two FIFOs with different widths and depths to manage data flow:

- A 1-bit width FIFO stores the information indicated by TUSER[2] (end of current video frame) when TLAST (indicating the end of a packet) is asserted and presents it to downstream at the start of the packet.
- 148-bit width DC FIFO is used to store remaining packet data for 25G and a 76-bit DC FIFO is used for 10G.

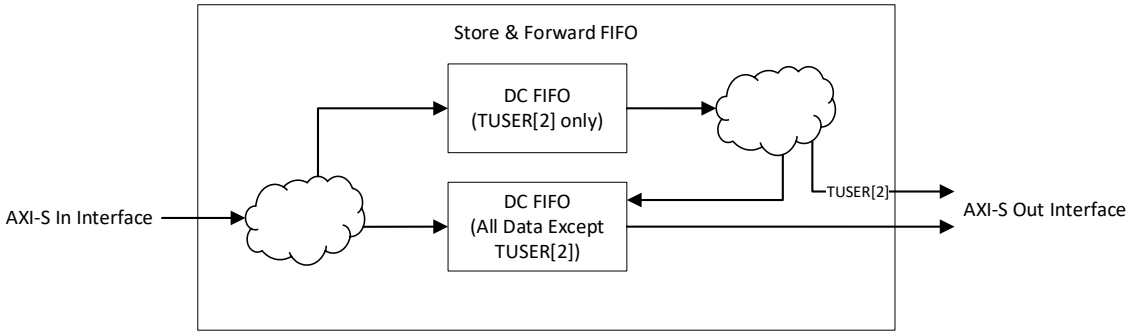


Figure 3.22. Store & Forward FIFO Internal Architecture

The DC FIFO width for 25G is calculated as follows:

- **FIFO Width** = TDATA + TLAST + TKEEP + TUSER = 128 + 1 + 16 + (4 - 1) = 148 bits
- **Minimum FIFO Depth** = [AVTP frame size without FCS] / Data width = 1330 / 16 = 83.125 entries, rounded up to 128 entries.
- **Depth of FIFO for TUSER[2]** = Two entries

Whereas the DC FIFO width for 10G is calculated as follows:

- **FIFO Width** = TDATA + TLAST + TKEEP + TUSER = 64 + 1 + 8 + (4 - 1) = 76 bits
- **Minimum FIFO Depth** = [AVTP frame size without FCS] / Data width = 1330 / 8 = 166.25 entries, rounded up to 256 entries.
- **Depth of FIFO for TUSER[2]** = Two entries

When the packet input data is valid, all signals of the AXI-S In Interface are written to the main FIFO, except TUSER[2]. The FIFO for TUSER[2] is written only when a valid TLAST is asserted. Data stored in the main FIFO is read out and presented downstream only when valid data is available in the FIFO for TUSER[2] and downstream is ready for data. The data read from the main FIFO continues for the same Ethernet frame until TLAST is read out. The FIFO for TUSER[2] is read only when the first data of the packet from the main FIFO is read out.

3.2.5.3. AVTP Header Inserter

After the data exits the Store & Forward FIFO module, it enters the AVTP Header Inserter module. This module is responsible for adding the necessary headers to the video data to prepare it for network transmission. The following figure shows the internal components of this module.

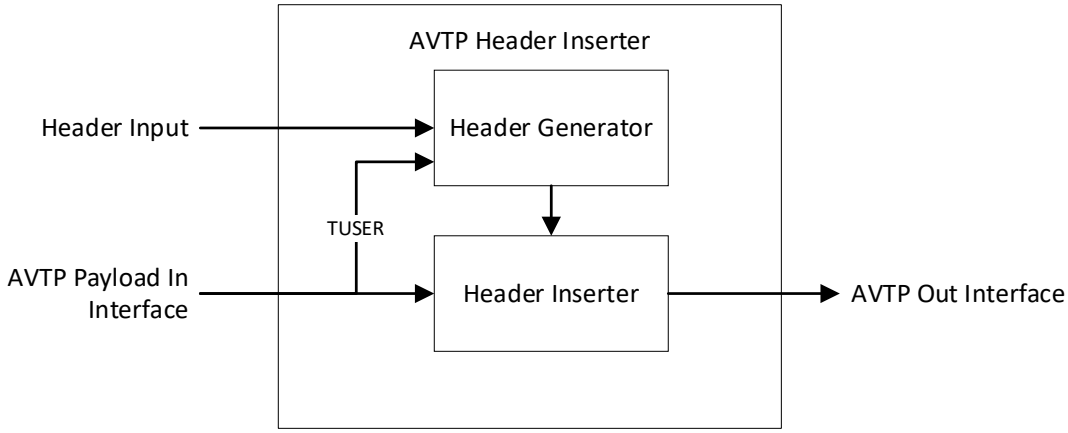


Figure 3.23. AVTP Payload Inserter Internal Architecture

The following describes the AVTP Header Inserter module:

- **Header Generation and Insertion**: The AVTP Header Inserter adds both Ethernet and AVTP headers to the video data (payload) it receives from the upstream module. These headers contain important information needed for the data to be correctly transmitted and interpreted by the receiving devices.
- **Data Flow**:
  - **Input**: Figure 3.20 shows the input AXI-S data bus from the Store & Forward FIFO, which carries the video data without headers.
  - **Output**: Figure 3.24 shows the output AVTP frame, which now includes the complete headers and the encapsulated video data in a 128-bit AXI-S data bus for 25G Ethernet.
    The generated AVTP header is inserted right before the AVTP payload (the video data). This combined data (header + payload) is then sent to the downstream output interface, ready for network transmission.

By adding these headers, the AVTP Header Inserter ensures that the video data is properly formatted for transmission over the network, which allows it to be correctly received and processed by other devices.

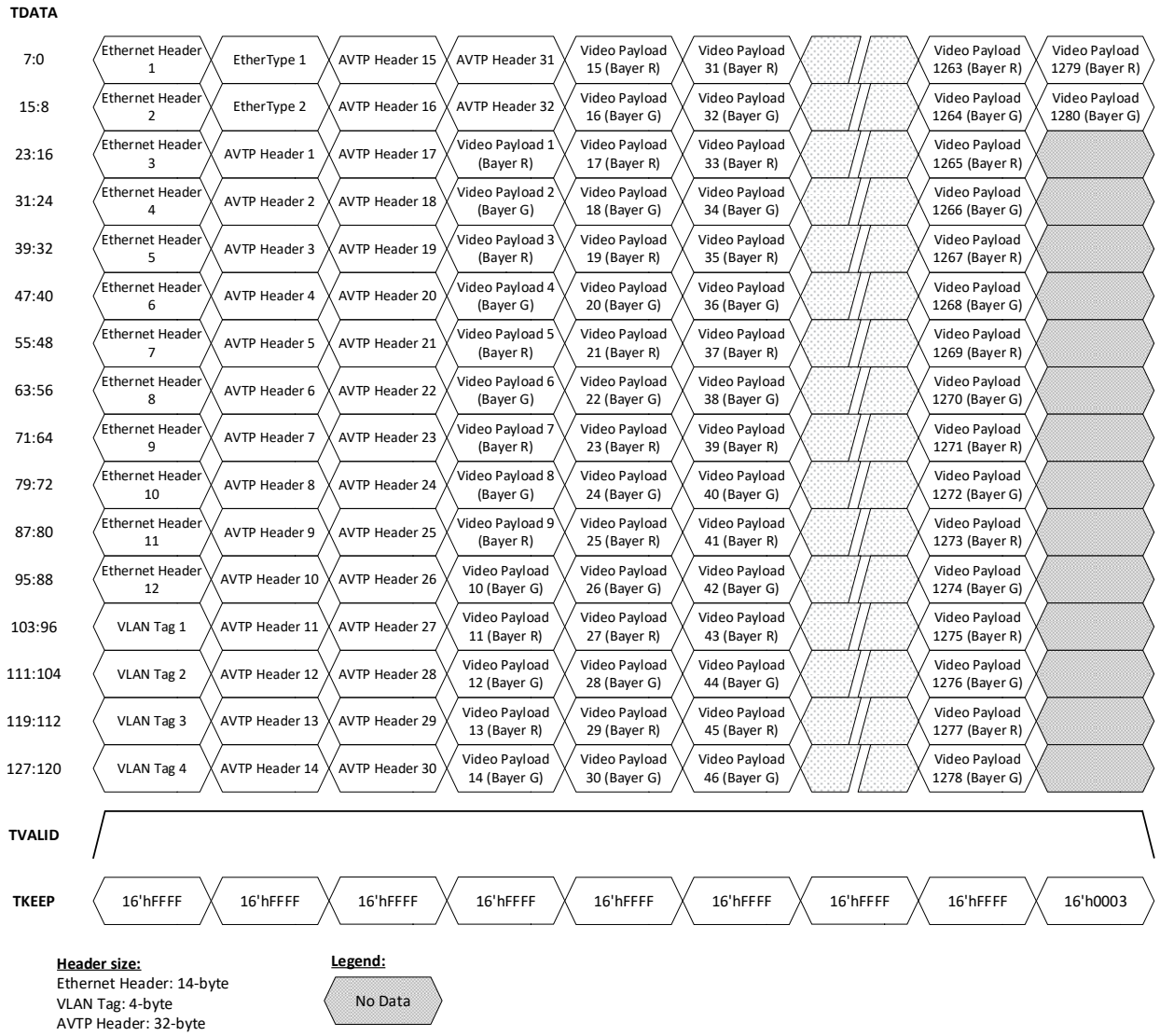


Figure 3.24. AVTP Header Inserter AXI-S Output Data for 25G Ethernet (Odd Line Number)

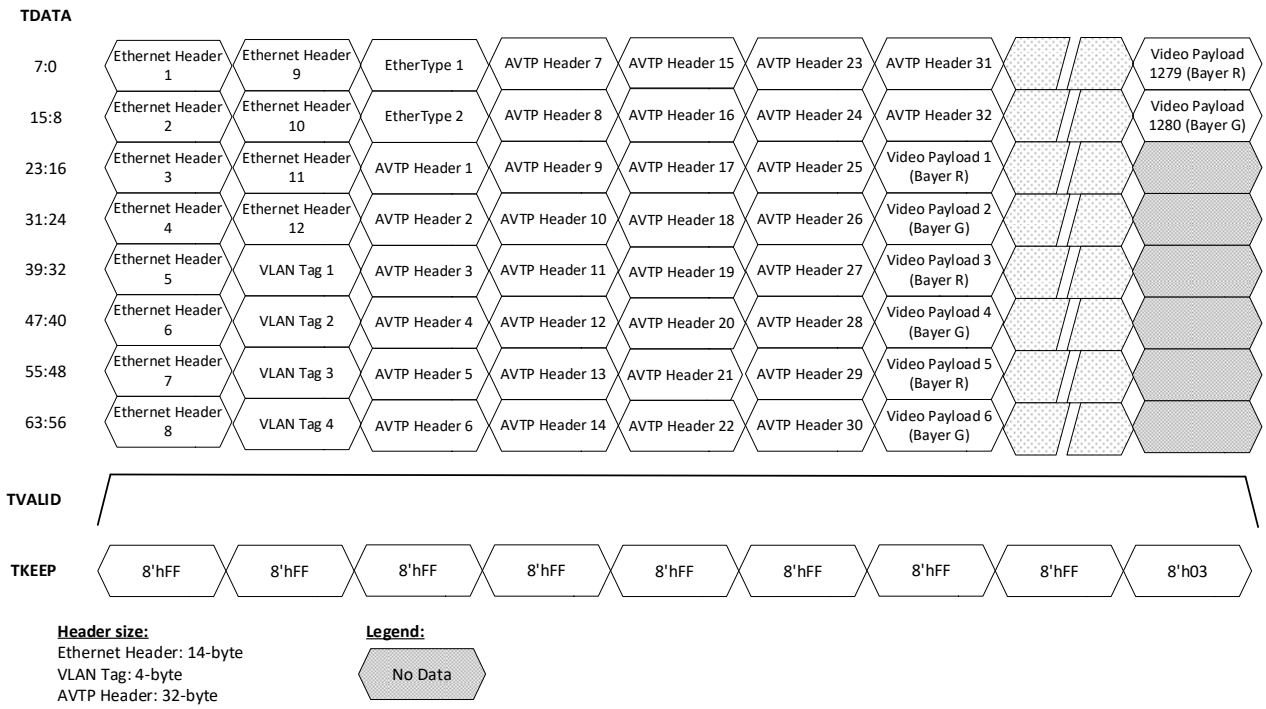


Figure 3.25. AVTP Header Inserter AXI-S Output Data for 10G Ethernet (Odd Line Number)

3.2.5.4. Aggregator

The Aggregator multiplexes AVTP frames from two AXI-S video streams using a round-robin scheduling method as shown in Figure 3.26 into single AXI-S interface. It then forwards these frames to the Ethernet TX FIFO and the Video Source Selector.

The Aggregator consists of the following criteria:

- **Multiplexing**: The Aggregator takes video data from multiple cameras and combines it into a single stream. This is done in a round-robin fashion, meaning it takes turns processing data from each camera.
- **Source Selection**: The Aggregator only selects a new video source when it is not currently processing an AVTP frame or has finished processing the current frame. This ensures smooth transitions between video sources.
- **No Backpressure at Output**: The system is designed to avoid backpressure, which means the downstream components must always be ready to receive data without causing delays.

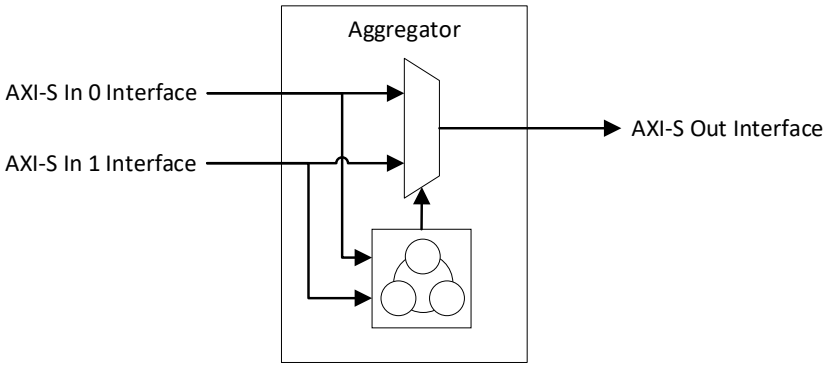


Figure 3.26. Aggregator Internal Architecture

3.2.5.5. Video Source Selector

This module is responsible for selecting video sources in two stages, ensuring that only the desired video data is processed and transmitted as shown in Figure 3.27.

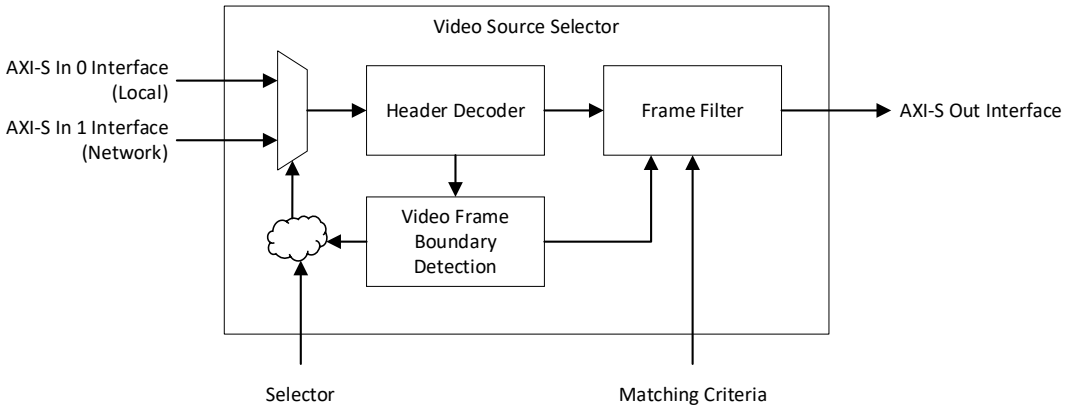


Figure 3.27. Video Source Selector Internal Architecture

Stage 1: Initial Source Selection (Local Camera Sensors or Network)

- **Data Reception**: The module receives Ethernet frames from both local Camera Sensor Subsystems and the Network Subsystem.
- **Source Filtering**: It only forwards the Ethernet frames from the selected source to the next stage. Frames from other sources are ignored and dropped internally.

Stage 2: Header Decoding and Frame Filtering

- **Header Decoding**: The module decodes the headers of the Ethernet frames.
- **Frame Filtering**: It forwards only the AVTP frames that match specific criteria (as described in Table 3.4) to the downstream blocks. Frames that do not match are ignored and dropped internally.

Table 3.4. Video Source Selector Ethernet Frame Matching Criteria

Fields	Byte Offsets	Matching Criteria
DA (destination MAC address)	0 – 5	91:E0:F0:00:FE:0*n*, where *n* is the selected stream ID.
TPID (tagged protocol identifier)	12 – 13	0x8100
AVTP EtherType	16 – 17	0x22F0
subtype	18	0x17
sv (stream_id valid)	19 (bit-7)	1
stream_id	22 – 29	0x00_D0_BD_17_22_0*m*_00_0*n*, where *n* is the selected stream ID, and *m* = *n*/2.

Filtering Based on Video Frame Boundaries

Filtering is essential to ensure that only complete and valid video frames are processed and transmitted. The system detects the boundaries of video frames, identifying the start and end of each frame. This detection is crucial for ensuring that only complete frames are processed. After the frame boundaries are detected, the system filters the video data to ensure that only frames meeting specific criteria are forwarded. This process may include verifying frame stream ID and ensuring that the frames originate from the selected video source.

Handling Different Scenarios

The module ensures that only complete video frames are output by handling various scenarios:

- **Normal Operation**: Outputs video packets from the start to the end of the video frame for the selected source.
- **Source Switching Outside Frame Boundaries**: If the video source switches outside the start and end of a video frame boundary, the switch happens immediately.

- **Source Switching Within Frame Boundaries**: If the switch occurs between the start and end of a video frame, the module waits until the current frame is fully processed before switching.
- **Receiving Mid-Frame Packets from New Source**: If a switch happens outside frame boundaries but the new source packets start mid-frame, the module waits until the current frame is fully output before switching. It then waits for the start of the new frame from the new source.
- **Receiving Mid-Frame Packets from Current Source**: If packets from the current source start mid-frame, the module waits for the start of the next frame before outputting data. This can happen if the display board connects to the network after the video source has already started transmitting.
- **Incomplete Frame Reception**: If only part of a frame is received (For example, due to a reset or disconnection), the module requires system-level detection to handle the scenario. If no packets are received after a timeout, a system-level reset may be needed, which is beyond the scope of this design.

3.2.5.6. AVTP Header Remover

This module is responsible for processing AVTP frames by removing their headers and forwarding the pixel data to the Video Line Buffer.

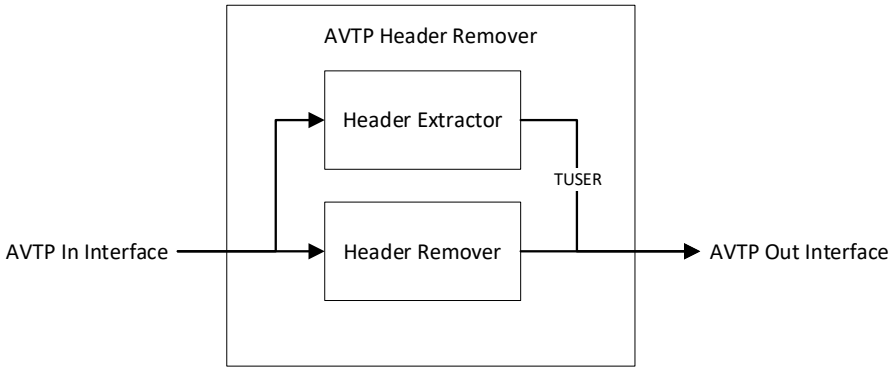


Figure 3.28. AVTP Header Remover Internal Architecture

The following lists the operations included for the AVTP Header Remover module:

- **Receiving AVTP Frames**: The module receives AVTP frames, which include video data along with Ethernet and AVTP headers.
- **Header Extraction**: The Header Extractor component extracts important header information from the incoming AVTP frames. This information is presented as TUSER (sideband signal) on the output interface when the first pixel of the AVTP frame is sent out. This ensures that the necessary control information accompanies the video data.
- **Removing Headers**: In the main data path, the Ethernet and AVTP headers are removed. This is achieved by de-asserting the TVALID and TKEEP signal for the corresponding bytes, effectively stripping the headers from the data. Figure 3.28 illustrates this process.
  - **AVTP Input**: Figure 3.24 shows the input AXI-S data bus from the Video Source Selector, which includes the headers.
  - **AVTP Output**: Figure 3.29 shows the output AXI-S data bus with the headers removed, resulting in a 128-bit data stream that contains only the video pixel data for 25G Ethernet.

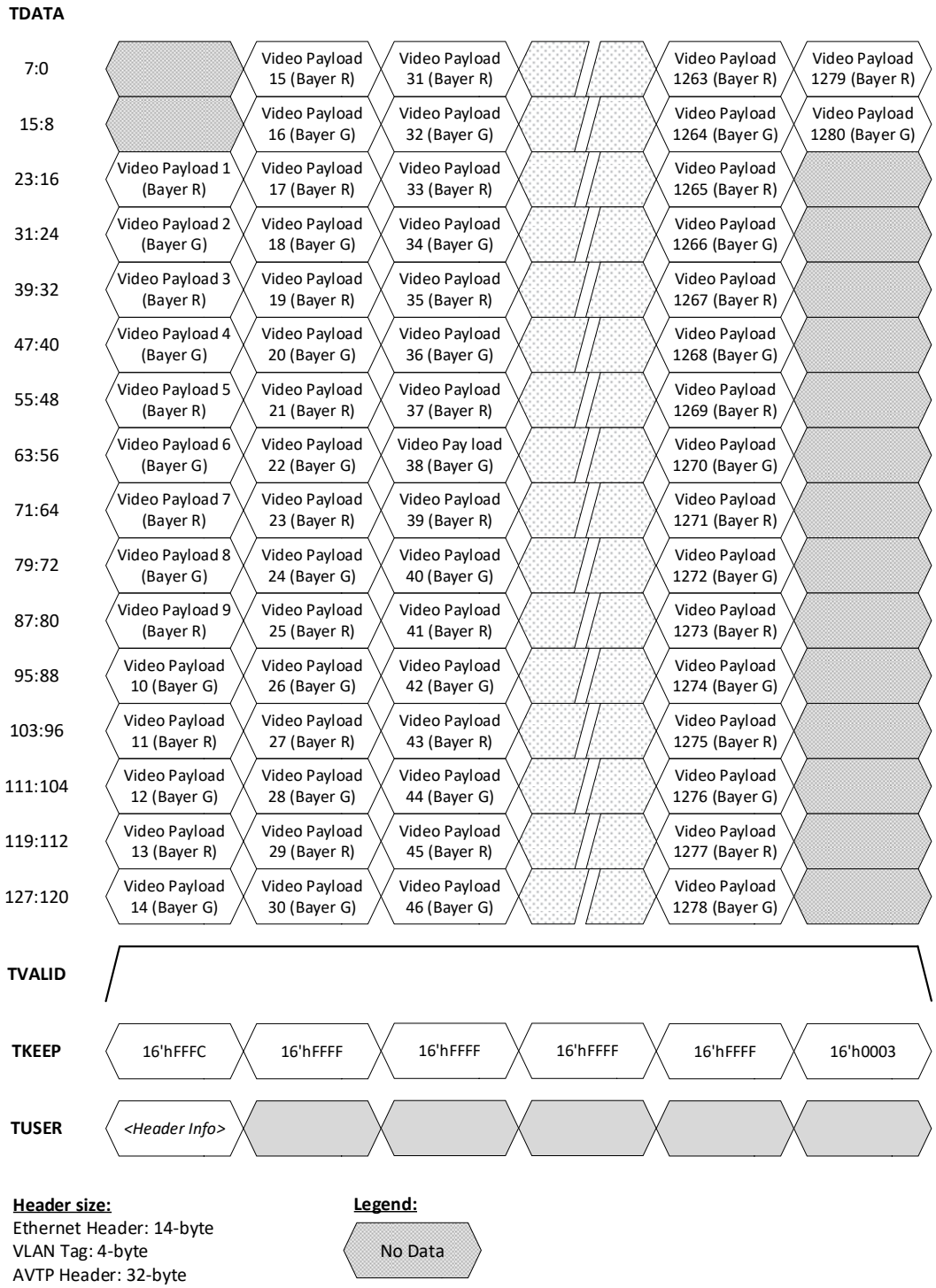


Figure 3.29. AVTP Header Remover AXI-S Output Data for 25G Ethernet (Odd Line Number)

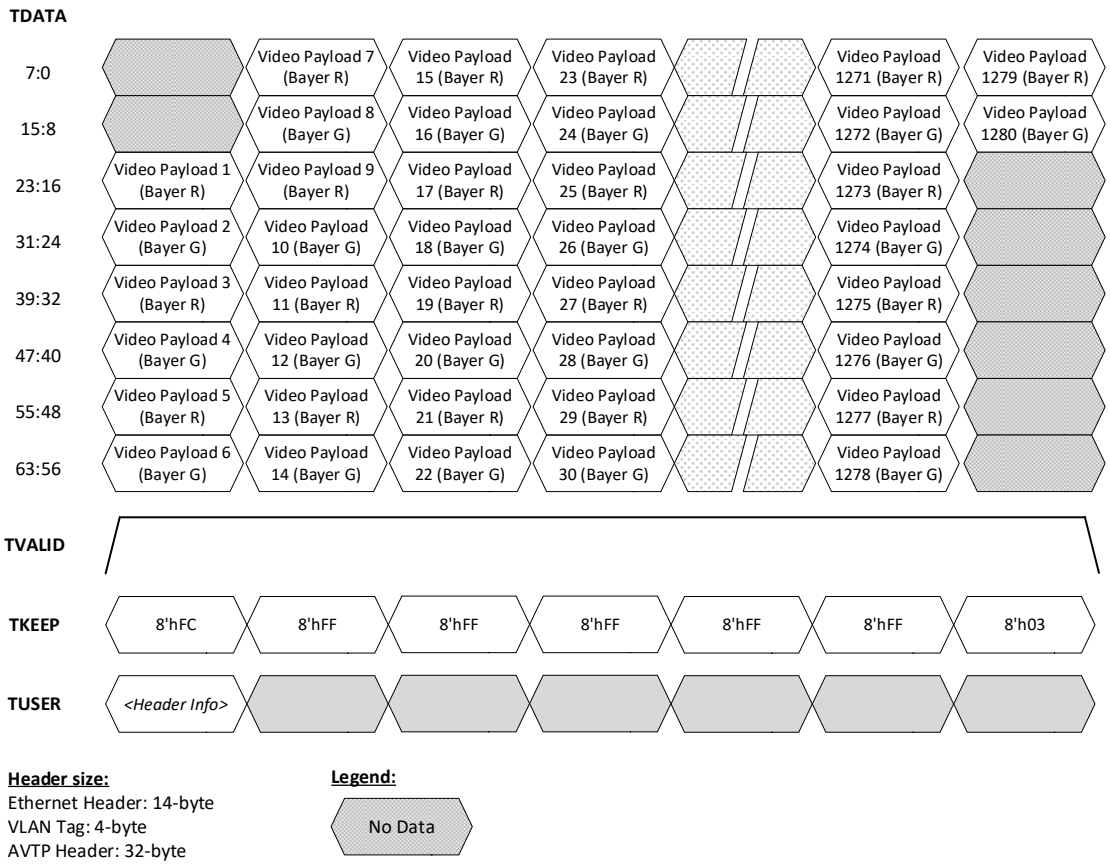


Figure 3.30. AVTP Header Remover AXI-S Output Data for 10G Ethernet (Odd Line Number)

- **Forwarding Header Information**: Certain header information is forwarded to the Video Line Buffer as part of the AXI-S TUSER signal along with the first cycle of every AVTP packet. This includes:
  - active_pixels: The number of active pixels in the frame.
  - total_lines: The total number of lines in the frame.
  - stream_data_length: The length of the video data stream.
  - line_number: The current line number being processed.
  - i_seq_num: The input sequence number.
  - ef (end frame): Indicates the end of the video frame.

  For detailed video format value, refer to Table 3.3.

3.2.5.7. Video Line Buffer

When video data is transferred over an Ethernet network, the received Ethernet frames containing the video data may encounter several issues:

- The order of the packets received may differ from the original sequence.
- Variable latency of packets may be introduced by the network.
- Some packets may be dropped in the network due to various reasons.

To address these issues, data buffering and reordering are necessary to ensure that continuous video data in the correct sequence is transferred to the Video Processing Subsystem.

This module includes a pseudo dual-port memory that can store up to two lines of video data as shown in Figure 3.31. When video data from the AVTP Header Remover is received, the data is stored in the memory dedicated to the first

line, based on the pixel offset derived from the i_seq_num field of the AVTP header. Once all video data for the scan line is received, the data is sent downstream.

In cases where packets are received out of sequence (For example, data for the next line is received before data for the current line), the data for the next line is stored in the memory dedicated to the next line, while the data for the current line, received later, is stored in the memory for the current line.

If packets are lost or experience significant delays, the design cannot wait indefinitely for these packets. Therefore, a timeout mechanism is implemented to start sending out video data for the specific line after a timeout period. For segments with lost or delayed data, the previous content from the memory will be sent as part of the current line.

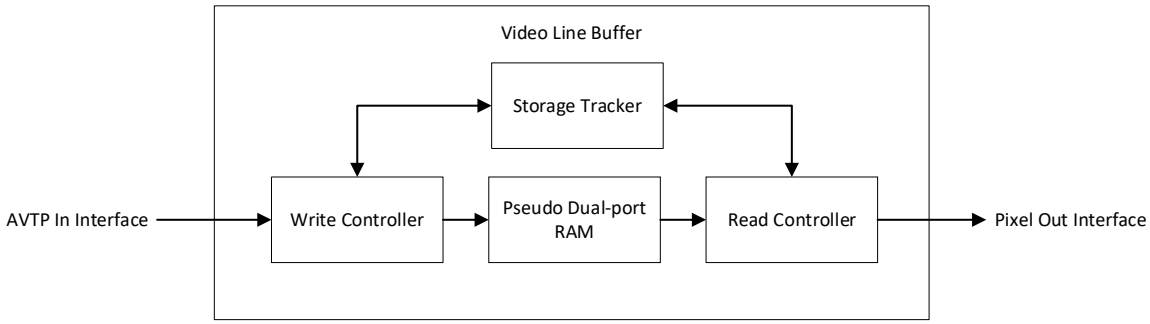


Figure 3.31. Video Line Buffer Internal Architecture

This module also functions as a clock-crossing data buffer, transitioning data from the Ethernet clock domain (25G: 195.3125 MHz, 10G: 156.25 MHz) to the pixel clock domain (4K: 148.5 MHz, 1080p: 74.25 MHz). Figure 3.15 shows the input AXI-S data bus from the AVTP Header Remover, while Figure 3.29 shows the output pixel data.

To comply with the input requirements of downstream video IP, horizontal blanking is inserted after the end of each video line, and vertical blanking is inserted after the end of each video frame.

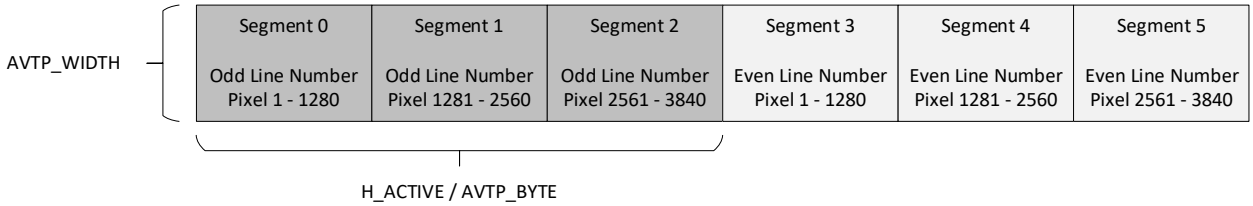


Figure 3.32. Video Line Buffer RAM Logical Structure

The video line buffer uses a Pseudo Dual-port RAM to store up to two lines of video data. Each video line is divided into three segments to manage the data efficiently.

The following lists the descriptions for Figure 3.31 and Figure 3.32:

- Storing Video Data:
  - The buffer can store up to two lines of video data at a time.
  - Each video line is divided into three segments (4K) or two segments (1080p) to organize the data.
  - The pixel data is written to a specific segment based on the line_number and i_seq_num (sequence number) from the TUSER signals.
  - For odd-numbered lines, the data is stored in segments 0, 1, or 2.
  - For even-numbered lines, the data is stored in segments 3, 4, or 5.
- Handling Out-of-Sequence Data:
  The system can handle Ethernet frames that arrive out of order. It places the pixel data in the correct segment according to the control signals, ensuring the data is in the right sequence.
- Write Controller:
  When video data is written to a segment, the Write Controller updates the status of that segment to 1 in the Storage Tracker. This indicates that the segment is filled with data.

- Read Controller:
  - The Read Controller checks the Storage Tracker. If all segments for a line are filled, it starts reading the video data and sends it downstream for further processing.
  - After reading the data, the Read Controller resets the status of the segment to 0 in the Storage Tracker, indicating that the segment is ready for new data.

Timeout Calculation

- 4K:
  Time required to transfer 3,840 pixels = (3,840 pixels / 2 pixels per clock) / (148.5 MHz) = 12.93 µs. Therefore, the current video lines must be transmitted within expected 12.93 µs after reception of first pixel of the video lines.
  Convert to number of clock cycle of 148.5 MHz clock = RoundDown [(12.93 µs) / (148.5 MHz)] = 1,920 clock cycles
- 1080p:
  Time required to transfer 1,920 pixels = (1,920 pixels / 2 pixels per clock) / (74.25 MHz) = 12.93 µs.
  Therefore, the current video lines must be transmitted within expected 12.93 µs after reception of first pixel of the video lines.
  Convert to number of clock cycle of 74.25 MHz clock = RoundDown [(12.93 µs) / (74.25 MHz)] = 960 clock cycles

Status Generation

Out of sequence status is generated by detecting if the current i_seq_num is not increment of previous AVTP frames. Exception is when i_seq_num is 0. Segment lost error status will be generated if it is time to read/transmit the data for that segment, but it is not yet received.

3.2.5.8. Ethernet TX FIFO

The Ethernet TX FIFO buffer is an important component in managing data flow in Ethernet communication. The Ethernet TX FIFO helps manage data flow between different parts of the Ethernet system. It temporarily stores data packets to handle situations where the Ethernet MAC interface is busy or needs to pause.

By default, the FIFO can store up to three AVTP frames. However, you can adjust this capacity based on specific needs. Figure 3.33 shows the flow of internal architecture.

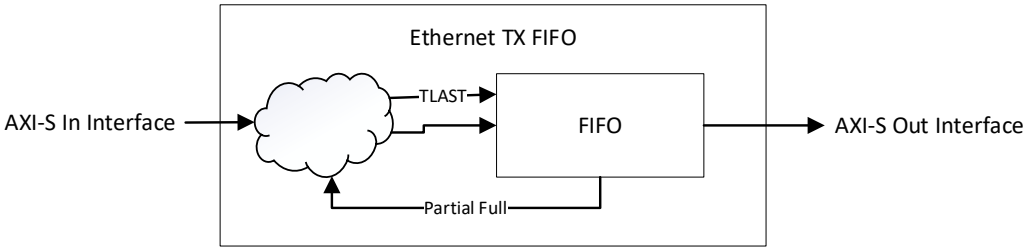


Figure 3.33. Ethernet TX FIFO Internal Architecture

The following describes the Ethernet TX FIFO buffer:

- Handling Data Flow:
  - The FIFO stores data packets when the Ethernet MAC interface cannot immediately process them. This helps prevent data loss and ensures smooth communication.
  - If the Ethernet MAC receives a pause frame (a signal to temporarily stop sending data), the FIFO will store incoming packets until the MAC is ready to continue.
- No Backpressure Support:
  - The FIFO input interface does not support backpressure. This means it cannot signal the sender to stop sending data if the FIFO is full.
  - If the FIFO becomes full, it will mark the last available slot with a special signal (TLAST=high) to indicate the end of the packet. It will also set an error signal (TUSER=high) to show that there was an issue.
  - Any additional data received while the FIFO is full will be discarded until the current packet is fully processed.

The input and output interfaces of the FIFO follow the same structure as shown in Figure 3.24.

When input interface is valid, all signals of AXI-S In Interface are written to the FIFO.

The FIFO is read out whenever there is valid data, as indicated by empty flag, and when the AXI-S output TREADY is asserted.

FIFO Depth Calculation

AVTP frame length (without FCS)	= 1,330 bytes
FIFO Width	= 128 bits
	= 16 bytes
Depth for each AVTP frame	= Round Up (1,330 bytes / 16 bytes)
	= 84
Depth for three AVTP frames	= 84 x 3
	= 252
Therefore, the depth of FIFO	= 256

3.3. Clocking Scheme

3.3.1. Clocking Overview

The following figure shows the clock network architecture for the MIPI CSI-2 to Ethernet Reference Design in Lattice Avant devices.

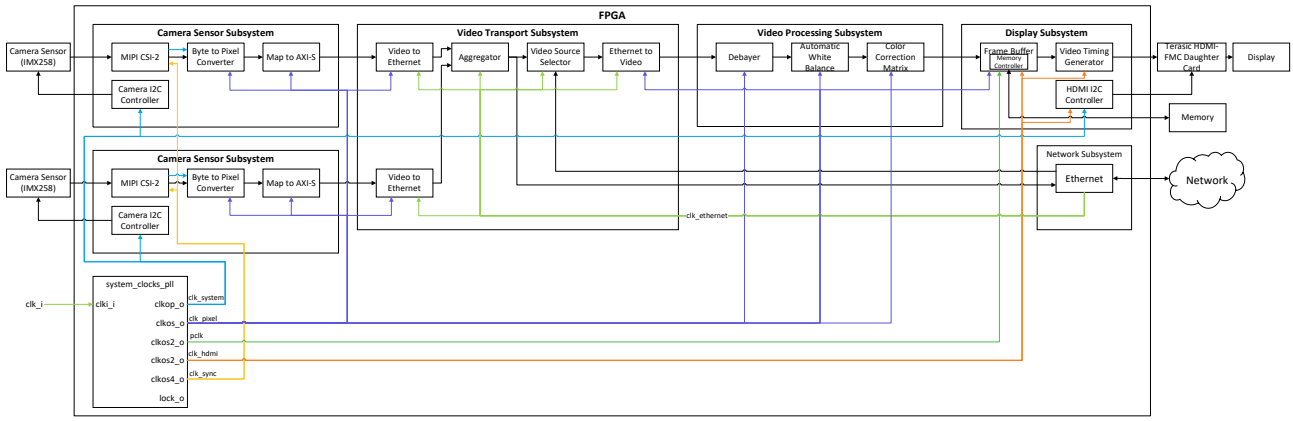


Figure 3.34. Reference Design Clock Domain Block Diagram

The following table lists the details of the clock frequency and distribution.

Table 3.5. Clock Domain Distribution

Clocks	Frequency (MHz)	Description
clk_i	27	Reference clock for system PLL.
clk_system	27	I2C clock for camera sensors and HDMI display.
clk_pixel	4K: 148.5 1080p: 74.25	Clock for all AXI-S pixel interfaces of • Camera sensor subsystems • Video to Ethernet and Ethernet to Video of Video Transport Subsystem • Video Processing Subsystem • Display Subsystem
pclk	90	Clock for Video Frame Buffer.

Clocks	Frequency (MHz)	Description
clk_hdmi	4K: 297 1080p: 148.5	Clock for HDMI display.
clk_sync	66	Clock for MIPI CSI-2 D-PHY Rx IP.
clk_eth	25 Gb: 195.3125 10 Gb: 156.25	Clock for data path of Network Subsystem and AVTP interface of Video Transport Subsystem.

3.4. Reset Scheme

3.4.1. Reset Overview

The following figure shows an overview of the reference design reset domain. In this design, the asynchronous reset pin is used as a system reset. This means you can reset the entire device by pressing a push button connected to this pin. The reset input from the push button is debounced before its first use to prevent issues caused by unstable reset. System PLL locked status is used as input to reset controller to ensure all other logic come out from reset only when the clocks driving them are stable.

The reset signals are synchronized to respective clock domain to ensure de-assertion of the reset are synchronized to the clock driving the logic.

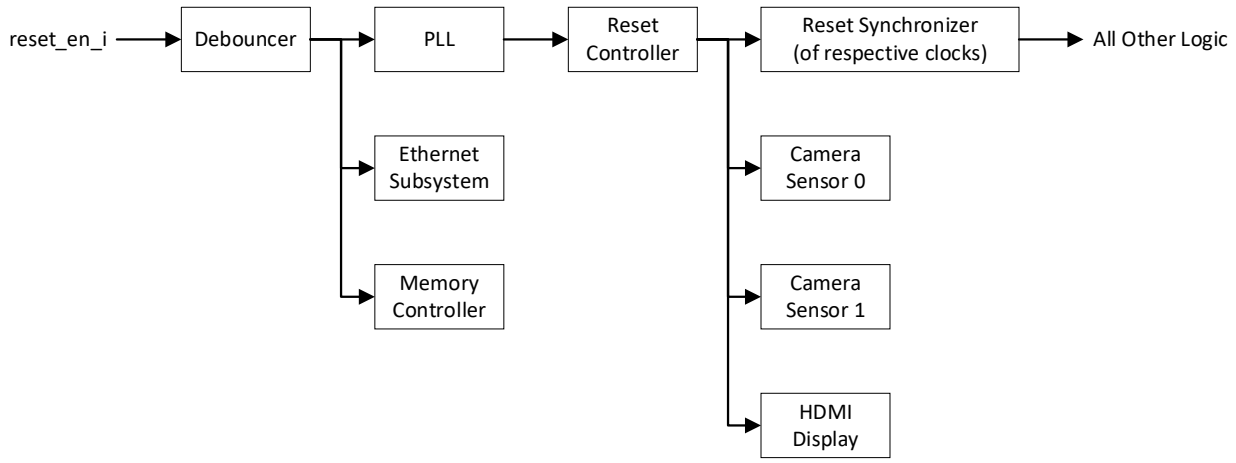


Figure 3.35. Reference Design Reset Block Diagram

4. Reference Design IP and Parameter Description

4.1. IP Description

This reference design is an advancement of the MIPI CSI-2 to HDMI reference design and the 25G and 10G Ethernet IP. Therefore, the IP and parameters are expected to be largely the same. This chapter provides a brief explanation of the IP used and focuses on the additional changes from the individual reference designs.

In this reference design, the project has organized the IP into three folders: one for 4K, one for 1080p, and a common IP directory used for both resolutions. This structure eliminates the need for you to regenerate IP cores when switching resolution between 4K and 1080p. Subchapters 4.1.1 to 4.1.10 in the IP Description section detail the IP cores used in 4K and 1080p with different configurations, while subchapters 4.1.11 to 4.1.13 cover the common IP used in both implementations with the same configuration.

Note 1: Regenerate the Automatic White Balance (AWB) IP only when the project location changes.

Note 2: To prevent overwriting internal IP constraints, do not regenerate the Memory Controller IP unless necessary.

If regeneration is required, follow these steps:

1. Manually copy the constraints from:
   RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\constraints\mc_constraint.sdc.
2. Replace the constraint **starting from line 252 to the end of the file** based on your resolution:
   - For 4K:
     RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\ip\4K\mc\constraints\constraints.sdc.
   - For 1080p:
     RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\ip\1080p\mc\constraints\constraints.sdc.

For more details regarding the individual design, refer to the MIPI CSI-2 to HDMI Reference Design User Guide (FPGA-UG-02206), 25G Ethernet MAC+PHY IP Core User Guide (FPGA-IPUG-02249), and 10G Ethernet MAC+PHY IP Core User Guide (FPGA-IPUG-02245).

4.1.1. MIPI CSI-2 D-PHY Receiver IP

The MIPI CSI-2 D-PHY Receiver IP for the Lattice Avant Versa board converts CSI-2 serial data on the D-PHY bus to a 32-bit parallel byte data. It utilizes four serial data lanes geared down to 8. This IP is implemented as a soft IP on the FPGA. Additionally, it generates an output byte clock based on the data type, which is then passed to the top-level logic of the DUT. Table 4.1 lists the parameters that must be changed when switching between different resolutions. Note for both configurations, the sensor is outputting a continuous D-PHY clock. In this mode, the recommendation is to disable RX_FIFO in the RX_FIFO settings and drive the clk_byte_fr_i input clock with the clk_byte_hs_o output clock directly.

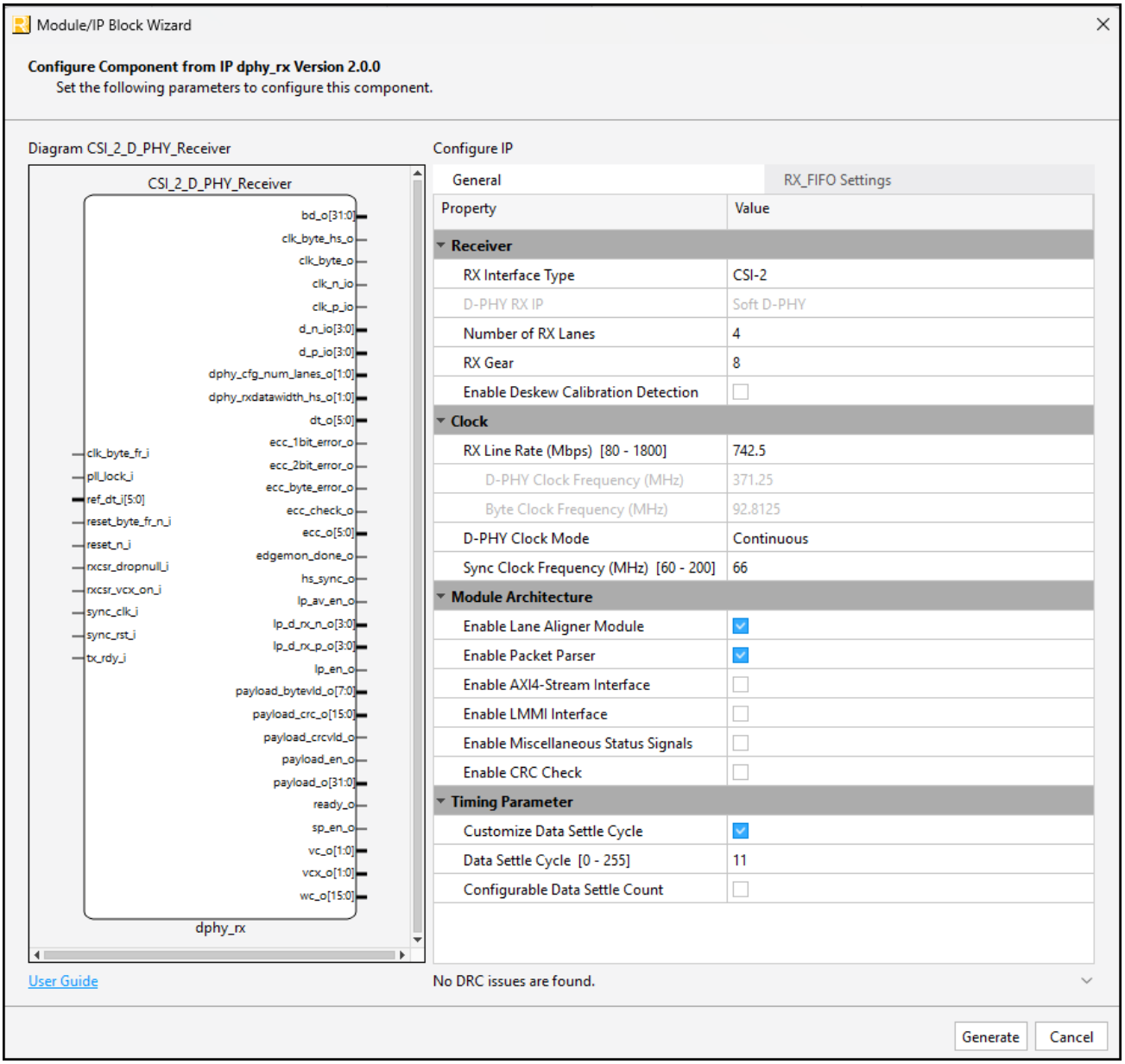


Figure 4.1. CSI-2 to HDMI IP GUI

Table 4.1. Parameters to Change in MIPI CSI-2 D-PHY IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Clock		
Rx Line Rate (Mbps)	371.25	742.25

4.1.2. Byte-to-Pixel Converter IP

The Byte-to-Pixel Converter IP converts CSI-2 standard-based video payload packets from the D-PHY Receiver Module output to pixel format. Additionally, the Byte-to-Pixel Converter IP generates image sensor control signals such as line valid and frame valid in the pixel domain based on the CSI-2 short packets. Table 4.2 lists the parameters that must be changed when switching between different resolutions. Note that with the IMX258 configurations for both resolutions, the Data Type from the sensor is configured to be RAW10, which means 10-bit per pixel. However, the downstream

components such as Debayer IP only requires 8-bit per pixel. Thus, in this design, each of the 10-bit pixel from this module is truncated at the low significant bits, turning them into an 8-bit pixel format.

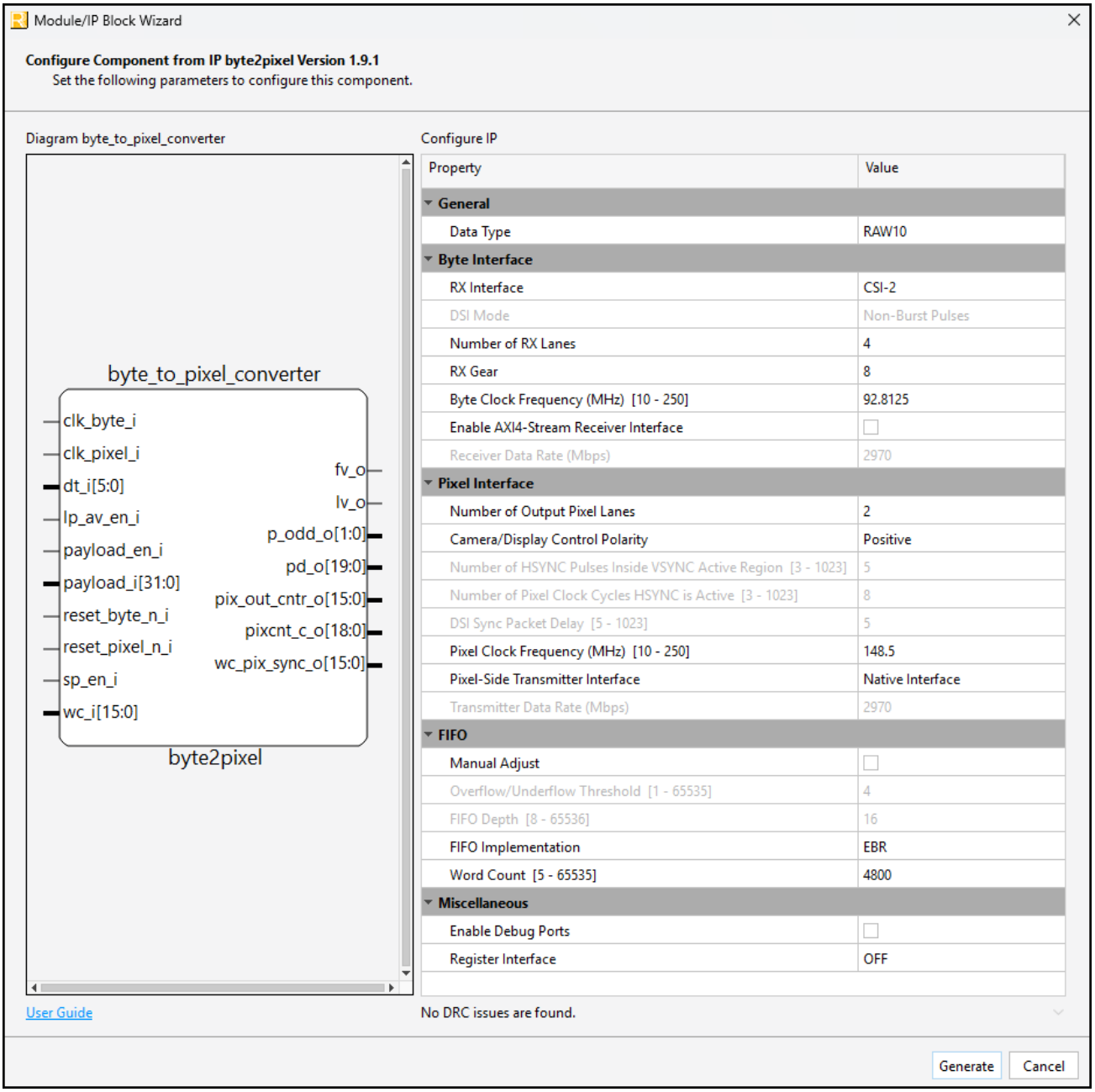


Figure 4.2. Byte-to-Pixel IP GUI

Table 4.2. Parameters to Change in Byte-to-Pixel IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Byte Interface		
Byte Clock Frequency (MHz)	46.40625	92.81250
Clock		
Pixel Clock Frequency (MHz)	74.25	148.50
FIFO		
Word Count	2,400	4,800

4.1.3. Debayer IP

CMOS color image sensors do not capture all three-color components for each pixel. They capture only one of the three-color components per pixel. Because green is the dominant component that closely captures luminance compared to red or blue, half of the pixels in a sensor capture the green component, while a quarter of the pixels capture the blue component, and the other quarter capture the red component. The process of interpolating and recreating the missing color components not captured by the sensor is called Debayering, De-Mosaicing, or Color Filter Array (CFA) Interpolation. After Debayering, each pixel is represented by all three-color components. Table 4.3 lists the parameters that must be changed when switching between different resolutions. Note that with the IMX258, the Bayer pattern is set to RGGB pattern.

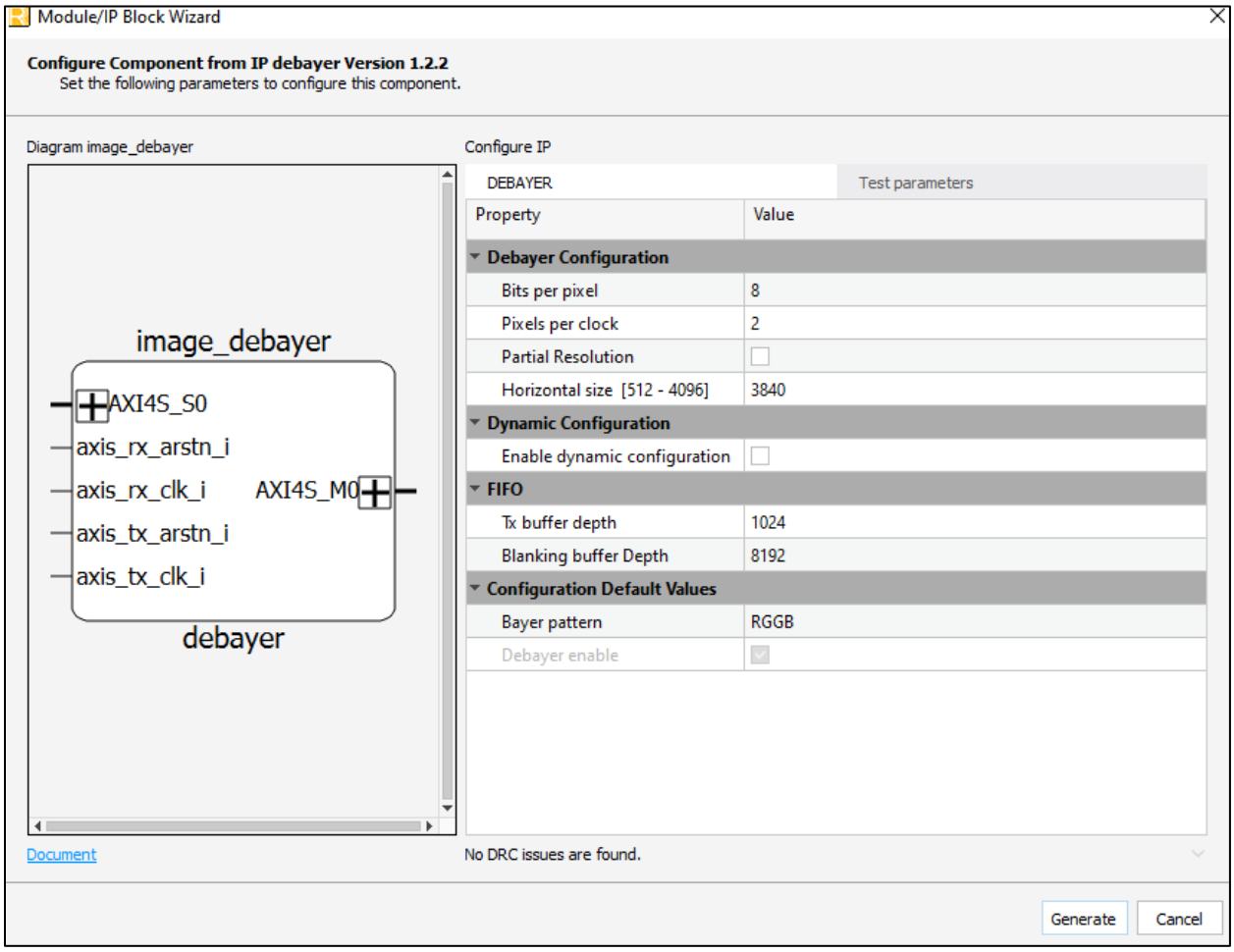


Figure 4.3. Debayer IP GUI

Table 4.3. Parameters to Change in Debayer IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Debayer → Debayer Configuration		
Horizontal Size	1,920	3,840
Test Parameter → Test Configuration		
Horizontal pixel size	1,920	3,840
Vertical pixel size	1,080	2,160

4.1.4. Automatic White Balance IP

The Automatic White Balance (AWB) IP automatically compensates for illumination temperature-based color differences, ensuring that white appears white. This operation is primarily performed in the Bayer domain but can also be done in the RGB domain. AWB is a pixel-based operation that uses image statistics. Because pixel correction is scene-based, it does not require calibration. White balancing is a two-step process: determining the nature of the illuminant and correcting the image based on the illuminant. Table 4.4 lists the parameters that must be changed when switching between different resolutions.

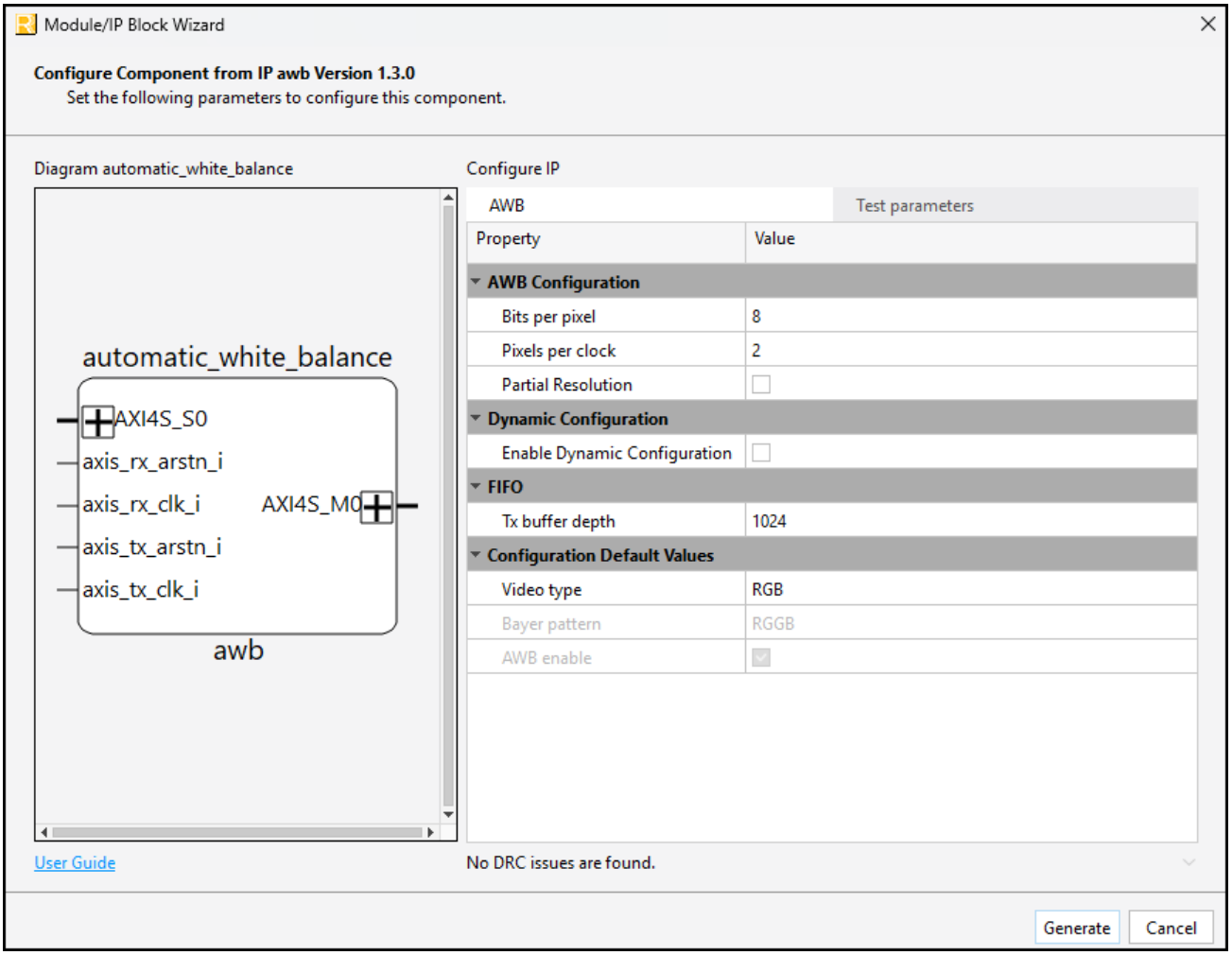


Figure 4.4. Automatic White Balance IP GUI

Table 4.4. Parameters to Change in Automatic White Balance IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Test parameters → Test configuration		
Horizontal pixel size	1,920	3,840
Vertical pixel size	1,080	2,160

4.1.5. Color Correction Matrix IP

The measured RGB values from the image sensors differ from the true RGB values of the image, primarily due to the characteristics of the optical filter overlay in the sensor. To obtain accurate colors, the pixels need to be mapped from the sensor RGB color space to the standard RGB color space. This linear mapping of the color components is achieved using a 3x3 matrix, known as the Color Correction Matrix (CCM). CCM is a pixel-level operation that does not require any line buffers and is performed in the RGB domain. Table 4.5 lists the parameters that must be changed when switching between different resolutions.

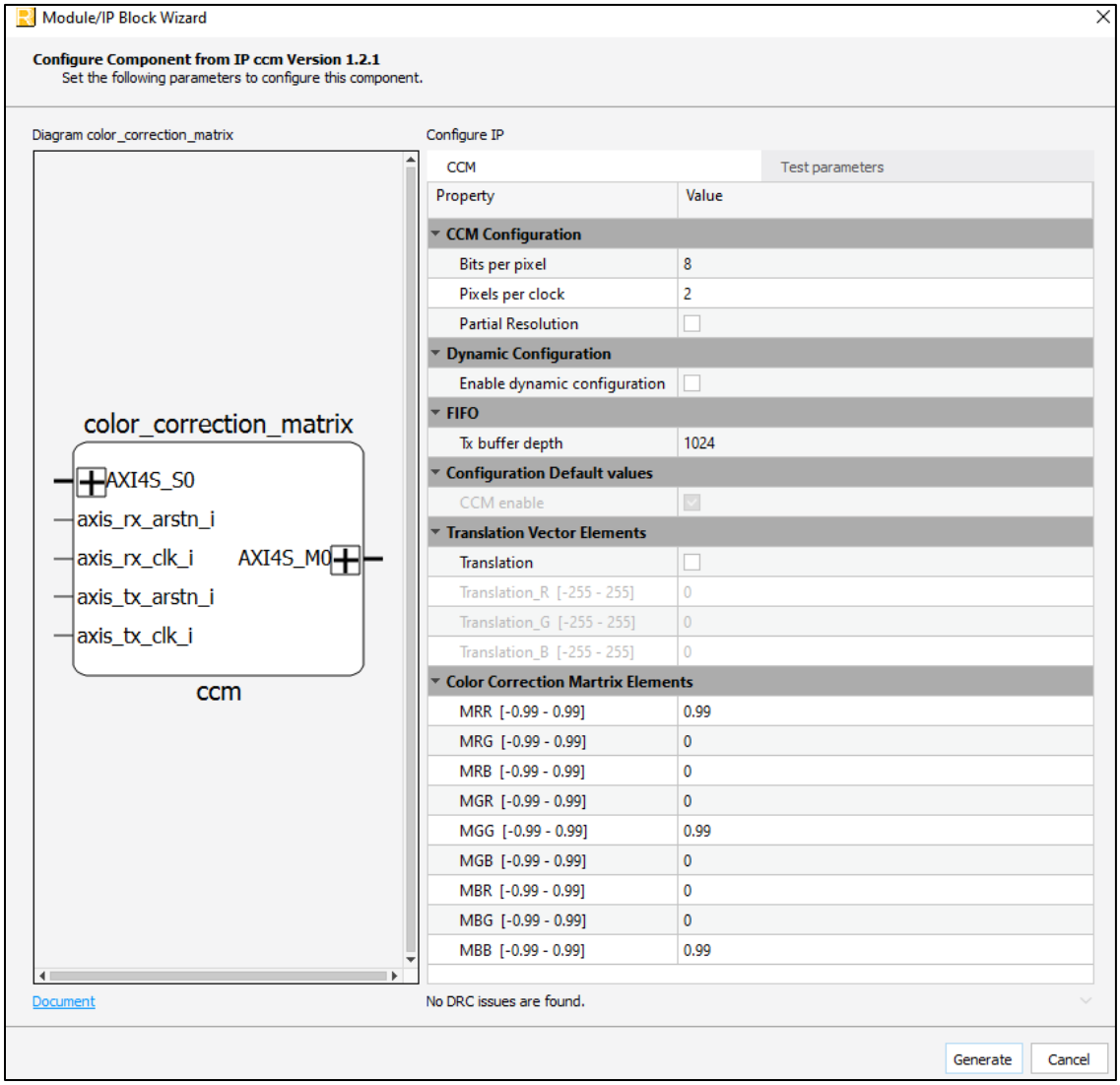


Figure 4.5. Color Correction Matrix IP GUI

Table 4.5. Parameters to Change in Color Correction Matrix IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Test parameters → Test configuration		
Horizontal pixel size	1,920	3,840
Vertical pixel size	1,080	2,160

4.1.6. System Clocks PLL IP

The PLL IP is used to generate clocks for the Byte-to-Pixel Converter (B2P), succeeding IP cores, HDMI, image sensor, I2C modules, and the sync clock input for the D-PHY Receiver. It takes a 27 MHz clock input from the board and the D-PHY Receiver output byte clock as an input. The PLL must be configured according to the operating resolution and frequency of the design. Table 4.6 shows the respective clock output frequencies for the design.

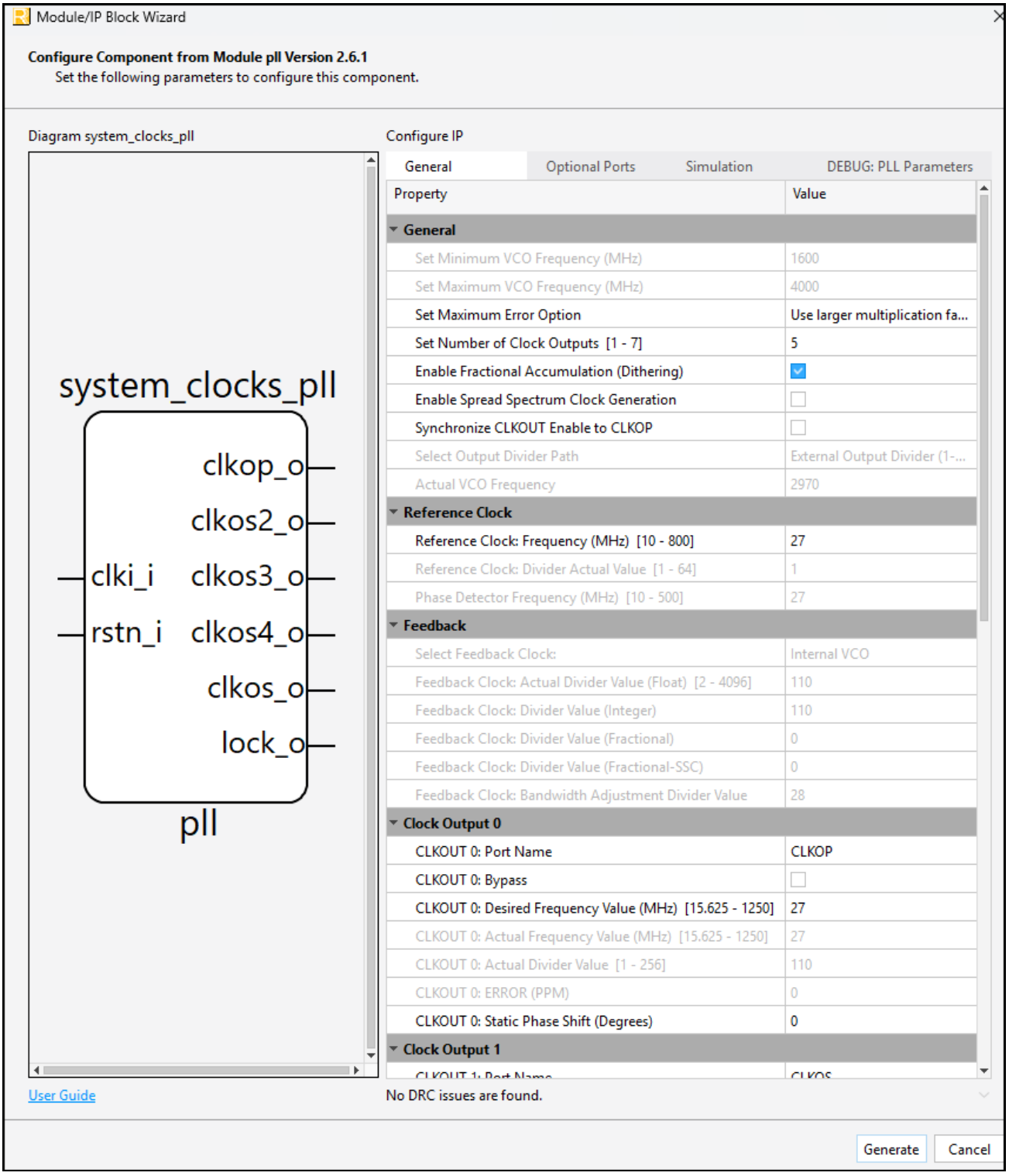


Figure 4.6. System Clock PLL IP GUI

Table 4.6. System Clocks PLL IP Output Frequency

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Clock output		
CLKOUT 0: Desired Frequency Value (MHz)	27	27
CLKOUT 1: Desired Frequency Value (MHz)	74.25	148.5
CLKOUT 2: Desired Frequency Value (MHz)	90	90
CLKOUT 3: Desired Frequency Value (MHz)	148.5	297
CLKOUT 4: Desired Frequency Value (MHz)	66	66

4.1.7. ROM IP for Image Sensor I2C

This ROM IP stores the I2C commands used to configure the image sensor for 1080p and 4K resolutions in the required sequence. You must regenerate this IP with the correct memory file stored. Load it from *RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\rtl\cam_i2c*.

Figure 4.8 shows the file location.

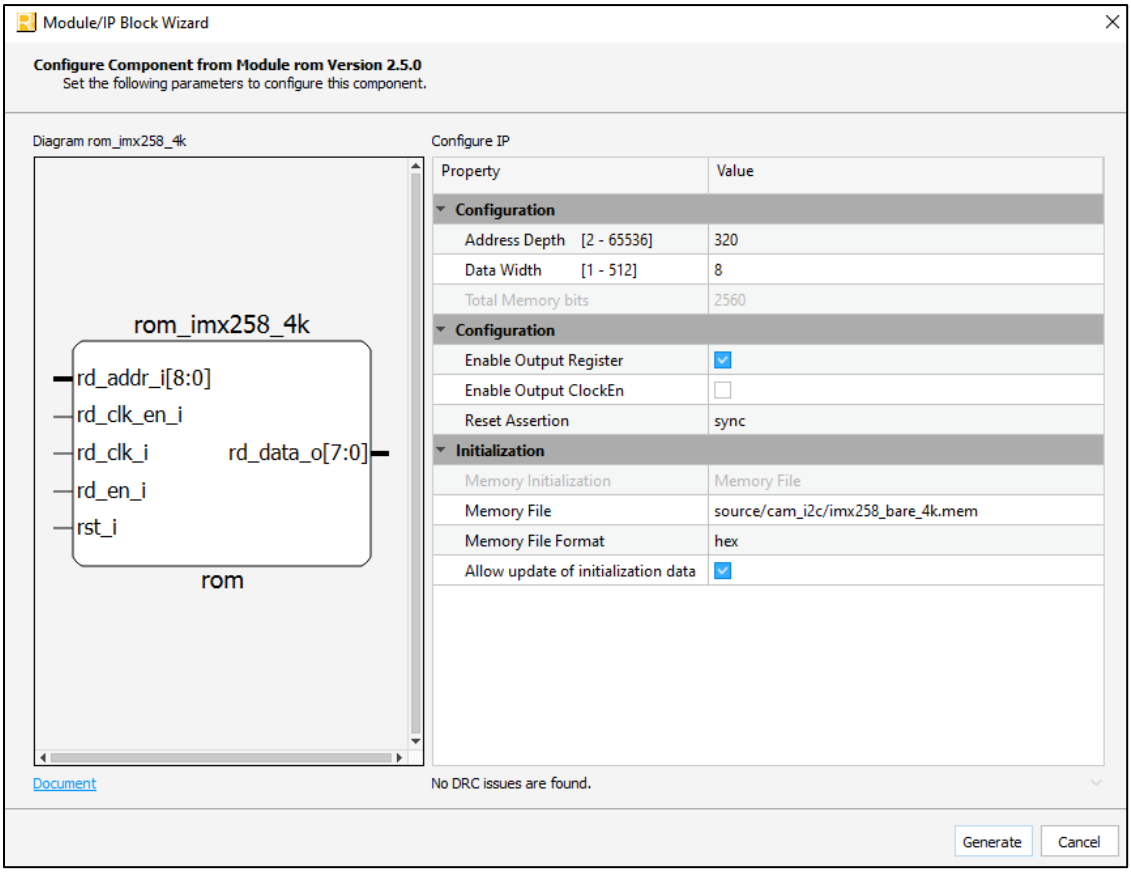


Figure 4.7. ROM IP GUI for Image Sensor I2C

Figure 4.8. Image Sensor Memory File Location

4.1.8. ROM IP for HDMI I2C

This ROM IP stores the I2C commands used to configure the HDMI for 1080p and 4K resolutions in the required sequence. You must regenerate this IP with the correct memory file stored. Load it from *RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\rtl\hdmi_i2c*. Figure 4.10 shows the file location.

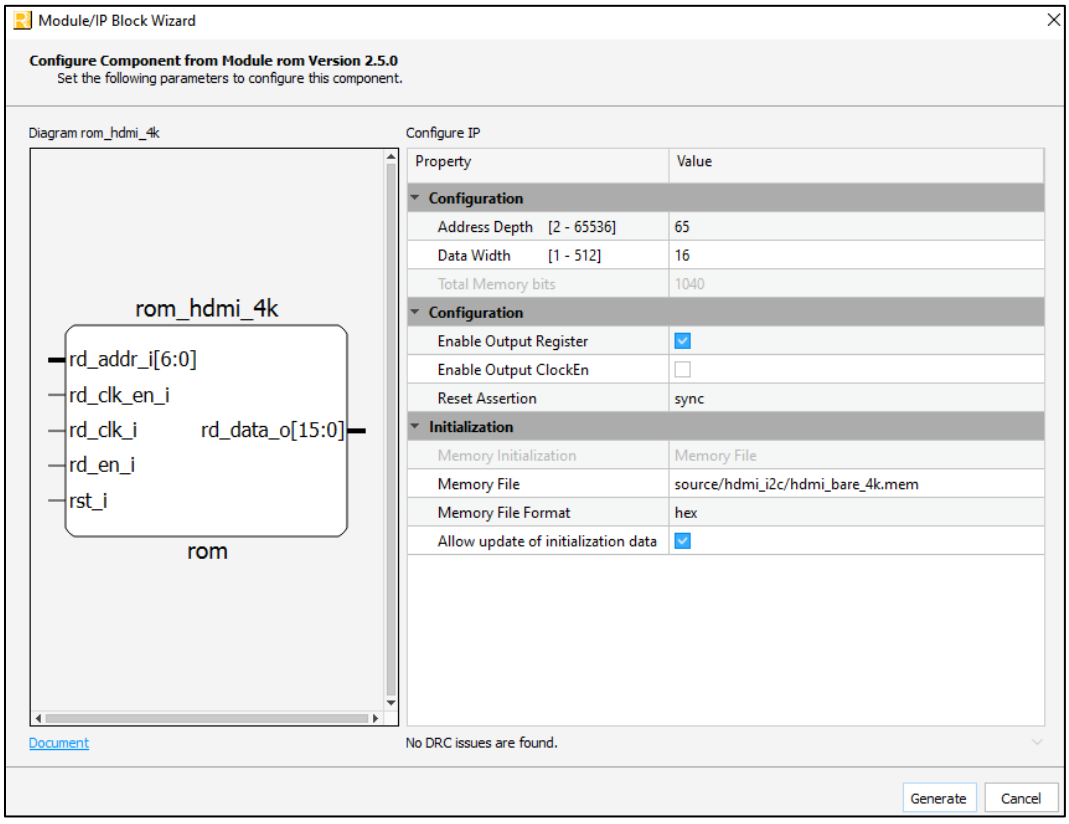


Figure 4.9. ROM IP GUI for HDMI I2C

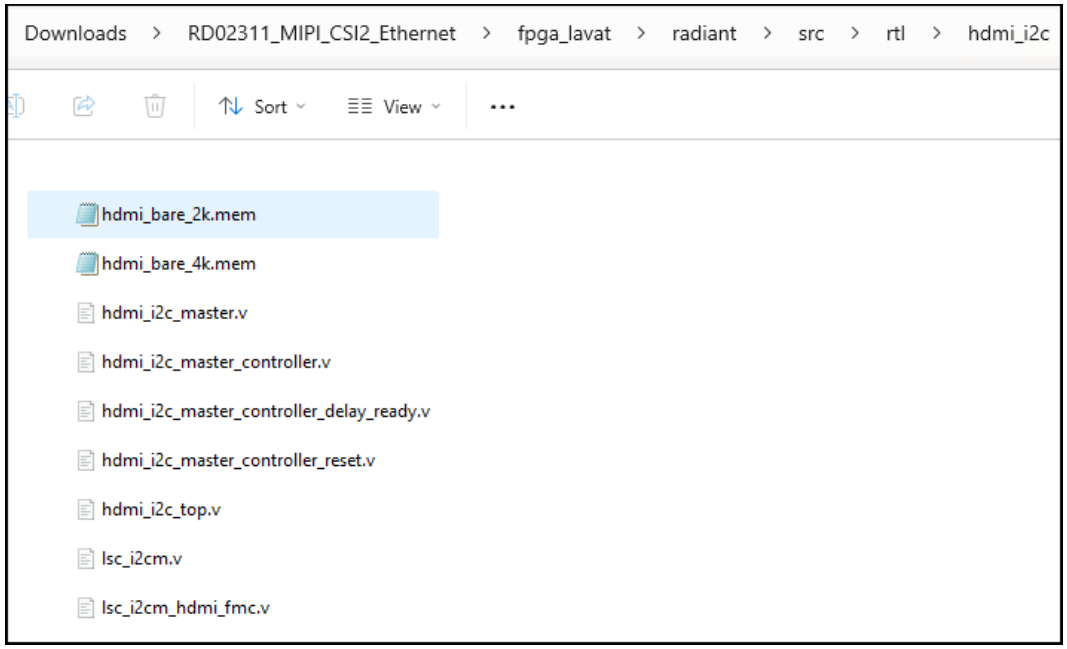


Figure 4.10. HDMI Memory File Location

4.1.9. Video Frame Buffer IP

A frame buffer is used between the CCM output and HDMI input, inside the CSI-2 to HDMI core design. A frame buffer is required for the demonstration design to handle timing synchronization between the sensor stream and HDMI transmitter input. Frame buffer IP is used to store the video frames onto the on-board memory by interfacing with the Memory Controller IP to make sure outgoing video are continuous with the compliant blanking periods

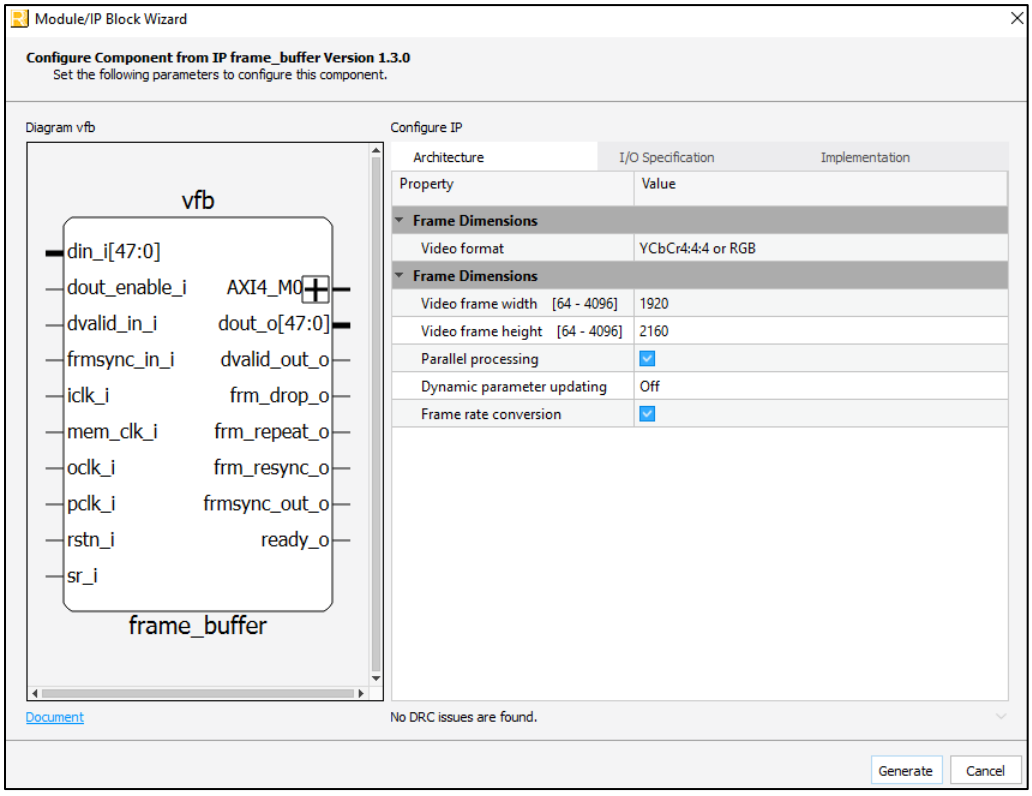


Figure 4.11. Video Frame Buffer IP GUI

Table 4.7. Parameters to Change in Video Frame Buffer IP

Attribute	Value for 1080p Configuration	Value for 4K Configuration
Architecture → Frame Dimension		
Video Frame Width	960	1,920
Video Frame Height	1,080	2,160
I/O Specification → Memory Interface		
Memory Base Address	25'b0000000000000000000000000	27'b000000000000000000000000000

4.1.10. Memory Controller for Avant IP

The Memory Controller IP core for Lattice Avant devices consists of three main blocks:

- Memory controller
- DDRPHY
- Training engine

The data interface allows you to initiate command/address/control and read/write operations to external DDR4/LPDDR4 SDRAM. The configuration interface provides access to the training engine and the Configuration Set Registers (CSRs), which configure the memory controller and perform the DDR4/LPDDR4 training sequences. The memory interface allows the selected Lattice FPGA to communicate with the external DDR4/LPDDR4 memory. For more information on the data and configuration interfaces, refer to the DDR Memory Controller IP User Guide (FPGA-IPUG-02208).

Note: The IP constraint file is already pre-generated. To prevent overwriting these constraints, you **must not regenerate** the IP.

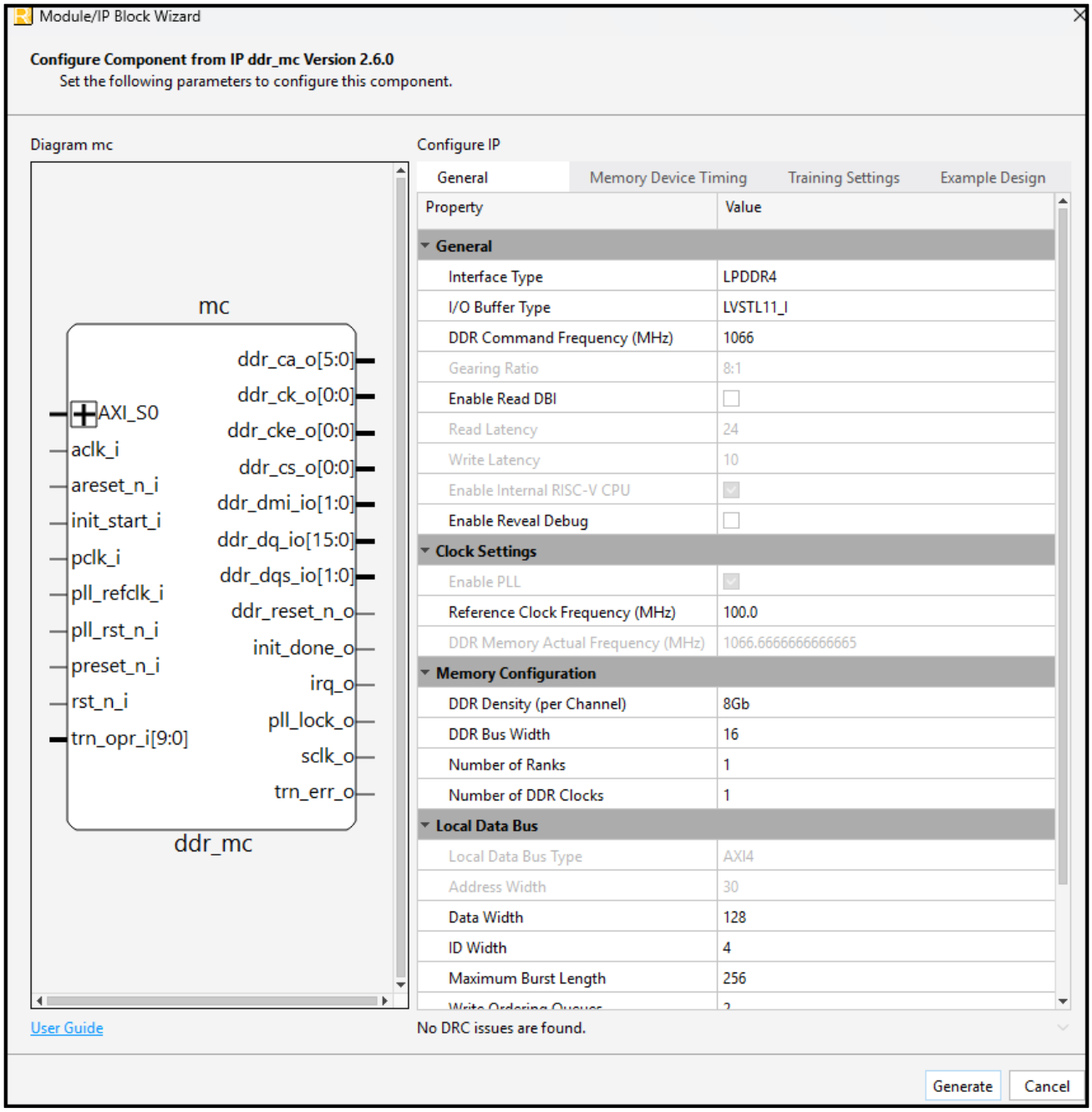


Figure 4.12. Memory Controller Avant IP GUI

4.1.11. DP RAM IP for Video Line Buffer

Video Line Buffer module within the Video Transport Subsystem described in the Video Line Buffer section requires RAM to store the video data. The DP RAM IP is instantiated as memory to store the video data.

For more information, refer to the module in *RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\rtl\video_transport\lscc_video_line_buffer.sv*.

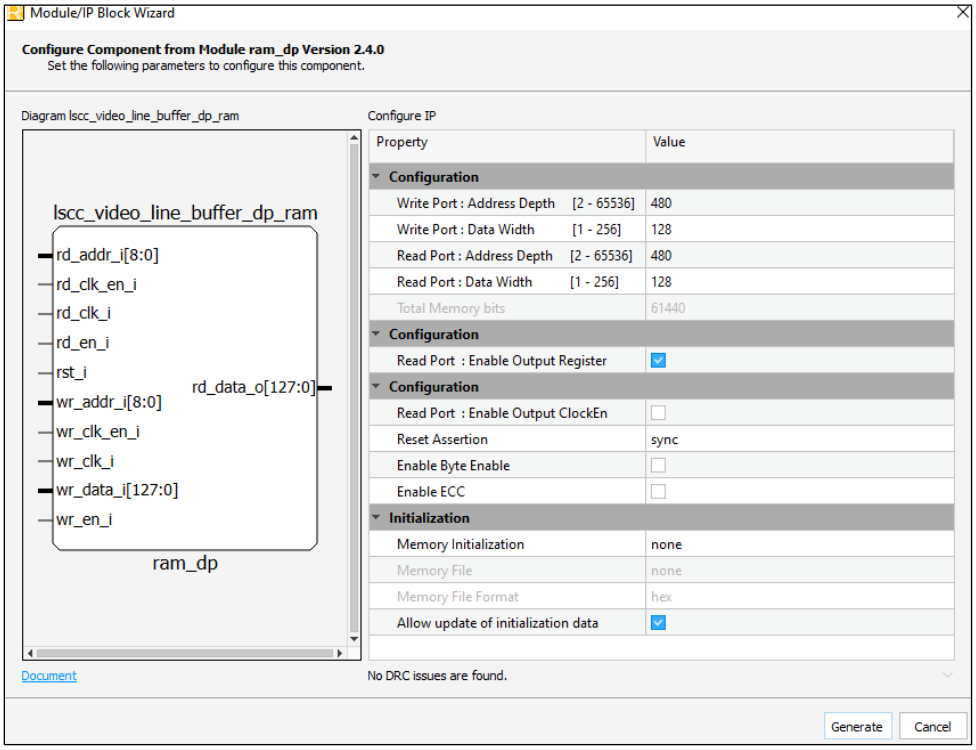


Figure 4.13. Video Line Buffer DP RAM IP GUI for 25G Ethernet Data Rate

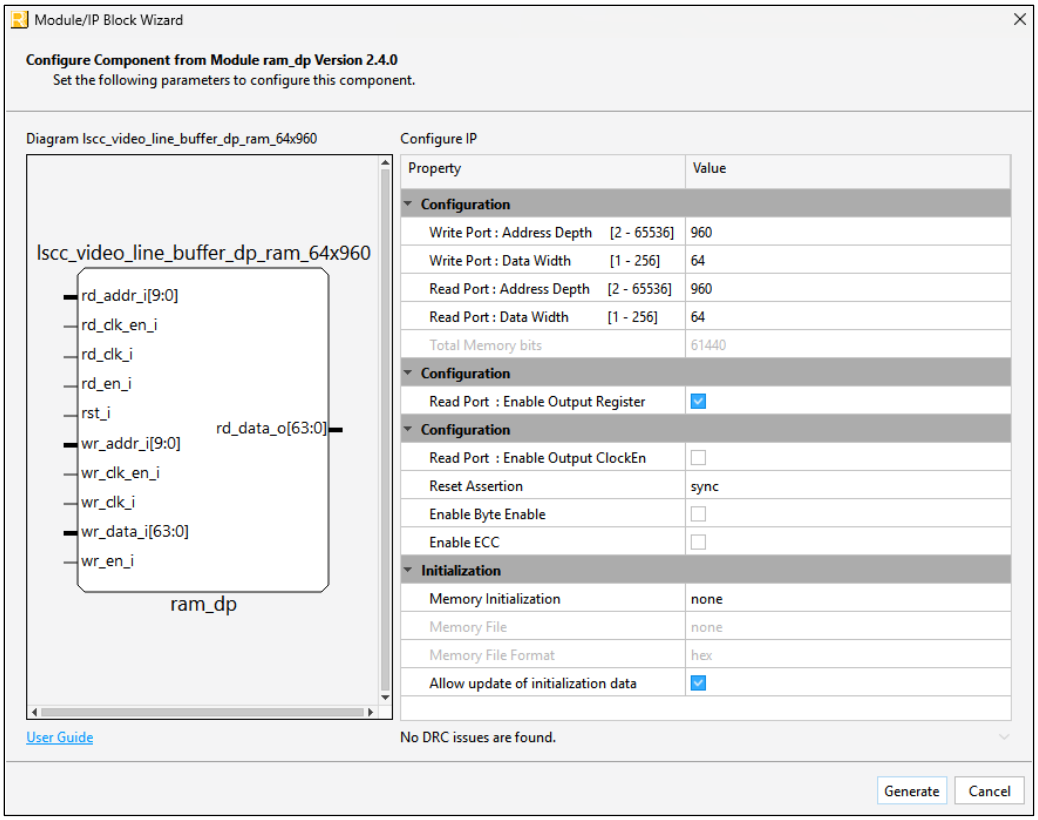


Figure 4.14. Video Line Buffer DP RAM IP GUI for 10G Ethernet Data Rate

4.1.12. FIFO IP

Store & Forward FIFO module and Ethernet TX FIFO module within Video Transport Subsystem described in the Store & Forward FIFO section and Ethernet TX FIFO requires FIFO memory to store the AVTP packet data. FIFO IP cores are instantiated as memory to store the packet data.

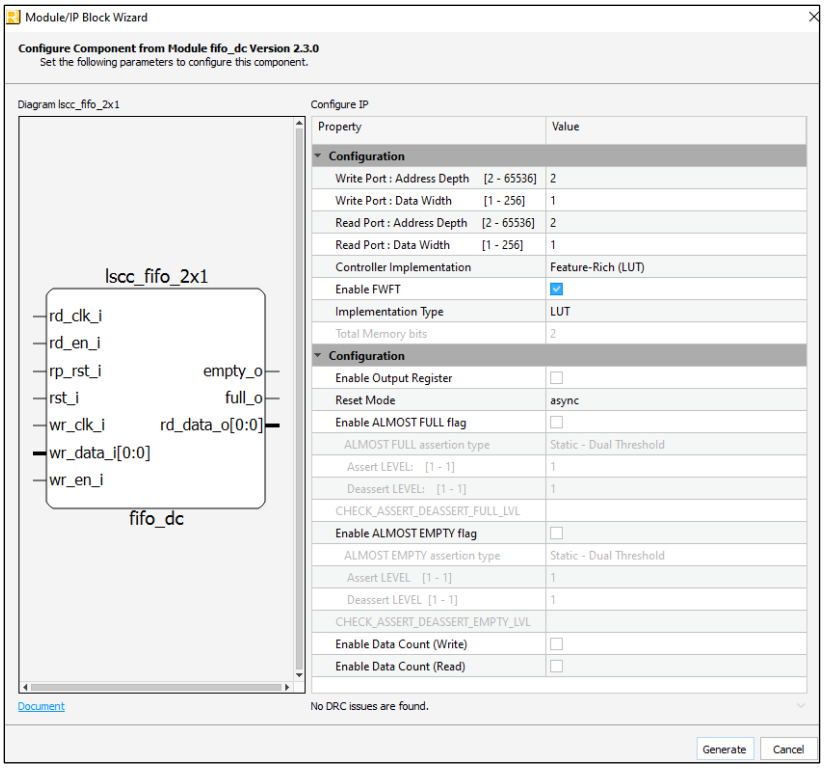


Figure 4.15. FIFO 2x1 IP GUI

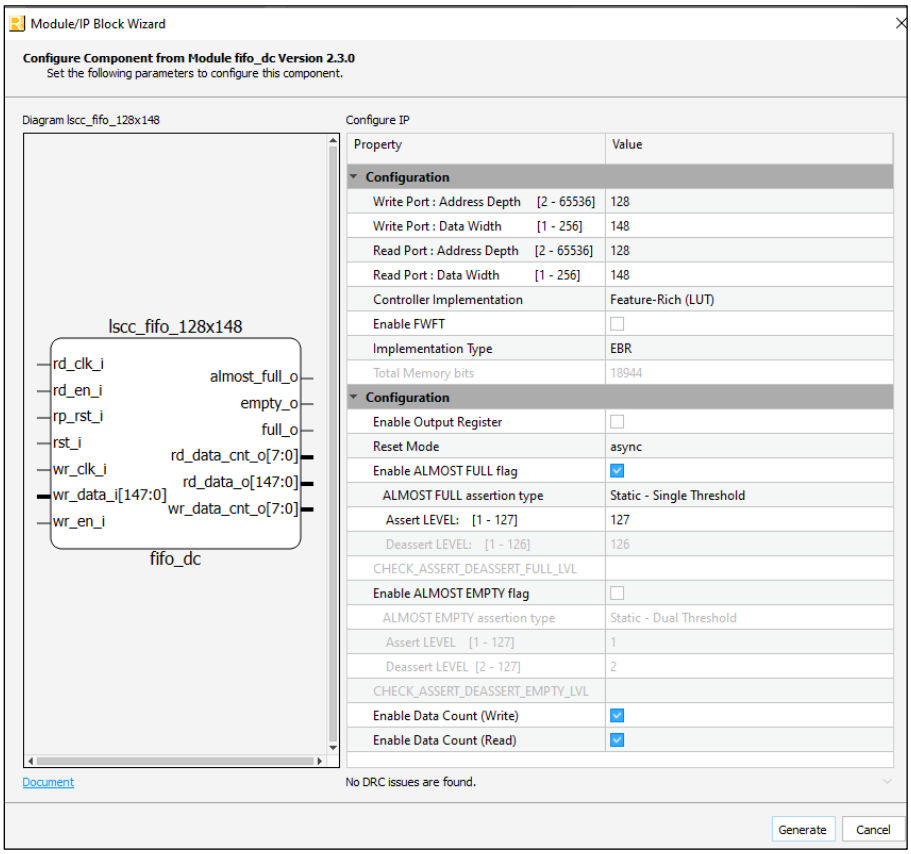


Figure 4.16. FIFO 128x148 IP GUI for 25G Ethernet Data Rate

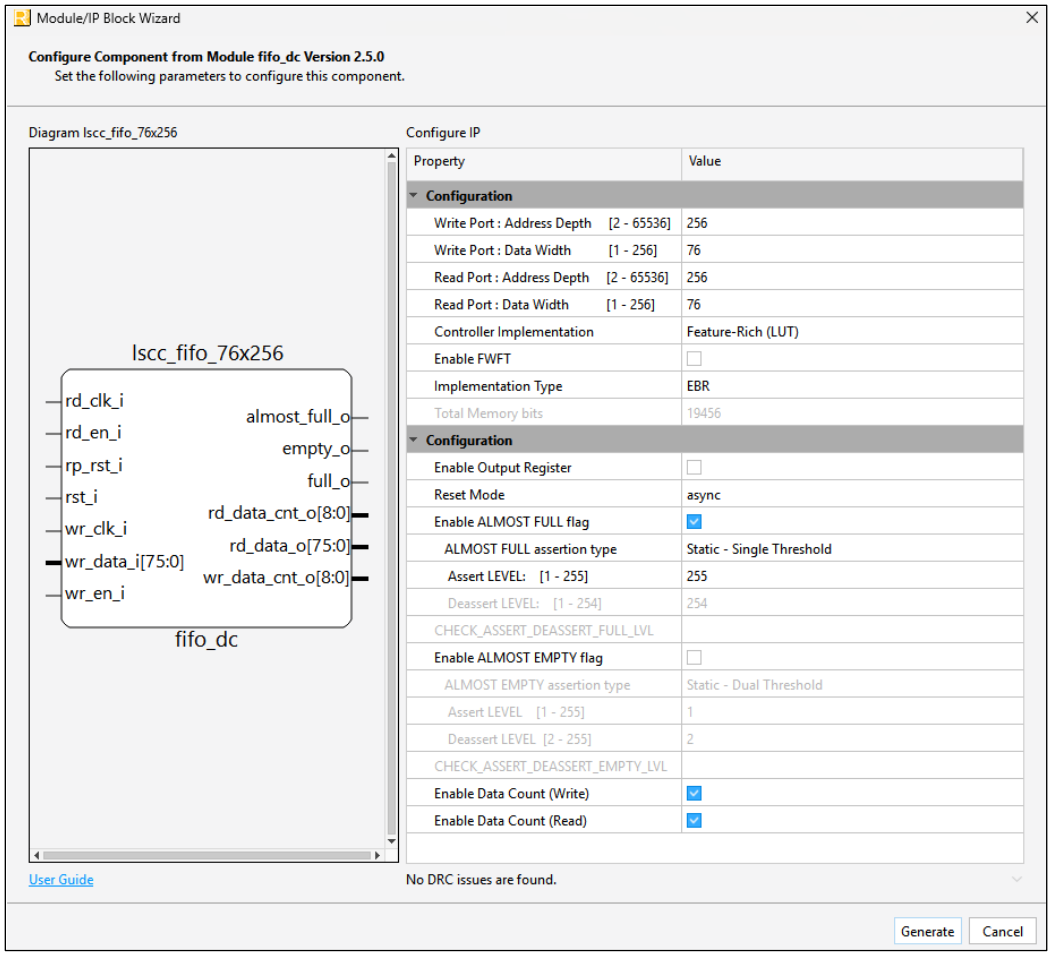


Figure 4.17. FIFO 256x76 IP GUI for 10G Ethernet Data Rate

4.1.13. 10G and 25G Ethernet IP

The reference design supports flexible integration of either the 25G or 10G Ethernet IP core, each comprising a Media Access Control (MAC) and Physical Layer (PHY) components. You have the flexibility to select the desired Ethernet data rate 25G or 10G based on your specific application requirements.

The 25G Ethernet IP core uses the 25-Gigabit Media Independent Interface (25GMII) to connect the MAC and PHY, while the 10G Ethernet IP core employs the 10-Gigabit Media Independent Interface (XGMII) for the same purpose. In both configurations, the MAC ensures compliance with IEEE 802.3 Ethernet standards during frame transmission and reception. On the transmit path, the MAC receives data frames via the AXI4-Stream interface and forwards them to PHY. On the receive path, the MAC extracts frame components from the PHY and delivers them to higher-level applications through the AXI4-Stream interface.

The PHY implements the Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA) functions, conforming to the IEEE 802.3 25GBASE-R or 10GBASE-R specifications, depending on the selected configuration. The MAC and PHY are interconnected through their respective media-independent interfaces (25GMII or XGMII), and the MAC receives its clock input from the PHY via the xg_tx_gclk_o signal.

The configuration parameters for the 25G and 10G Ethernet IP cores are detailed in and Table 4.8 and Table 4.9, respectively.

Table 4.8. 25G Ethernet IP Configuration Supported by the Reference Design

Parameter Settings	Value
IP Option	MAC + PHY
Multicast Address Filtering	Disabled
Statistic Counter Registers	Disabled
FEC Mode	Disabled

Table 4.9. 10G Ethernet IP Configuration Supported by the Reference Design

Parameter Settings	Value
IP Option	MAC + PHY
Host Interface	AXI4-lite
Loopback Mode	Disabled
Multicast Address Filtering	Disabled
TX Pause Frame Generation via Ports	Disabled
RX Coupling Mode	AC Coupling
RX Loss of Sig port Enabled	Enabled
DFE Enable	Enabled
DFE Tap Configuration	All Taps

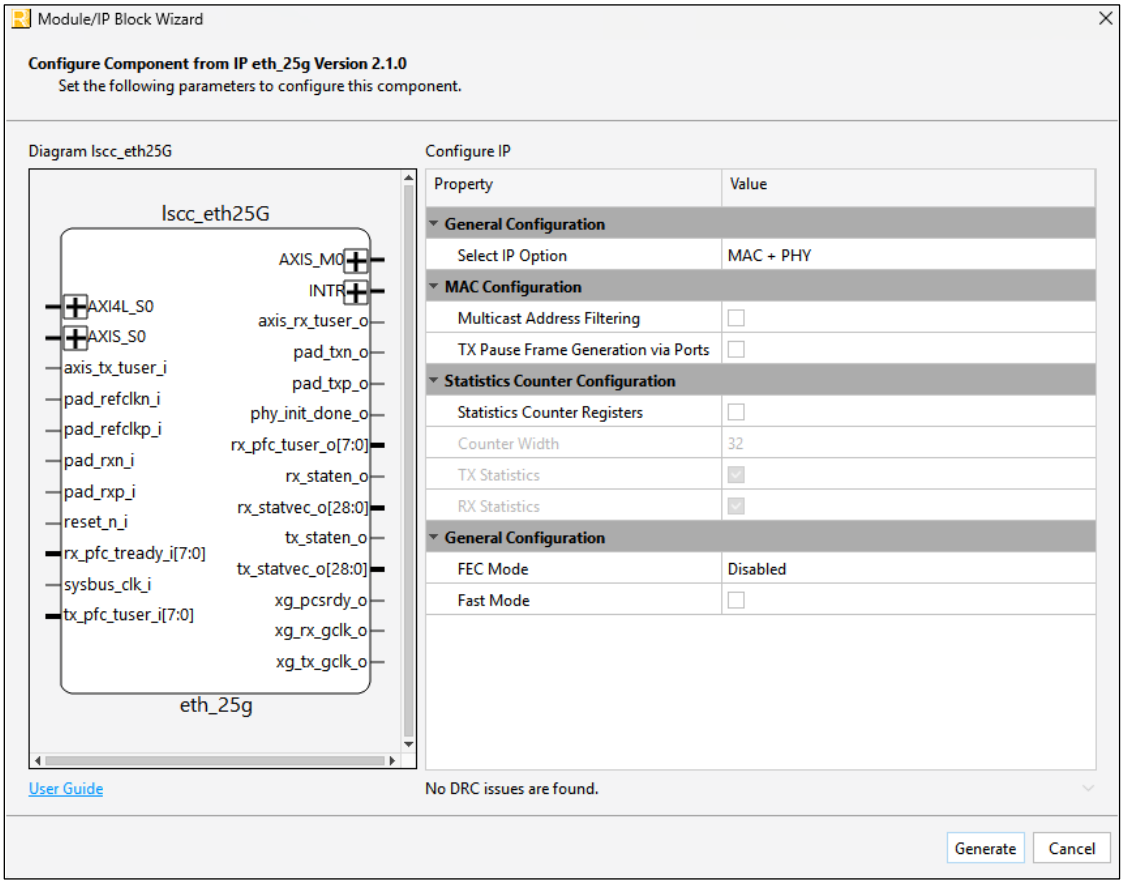


Figure 4.18. 25G Ethernet IP GUI

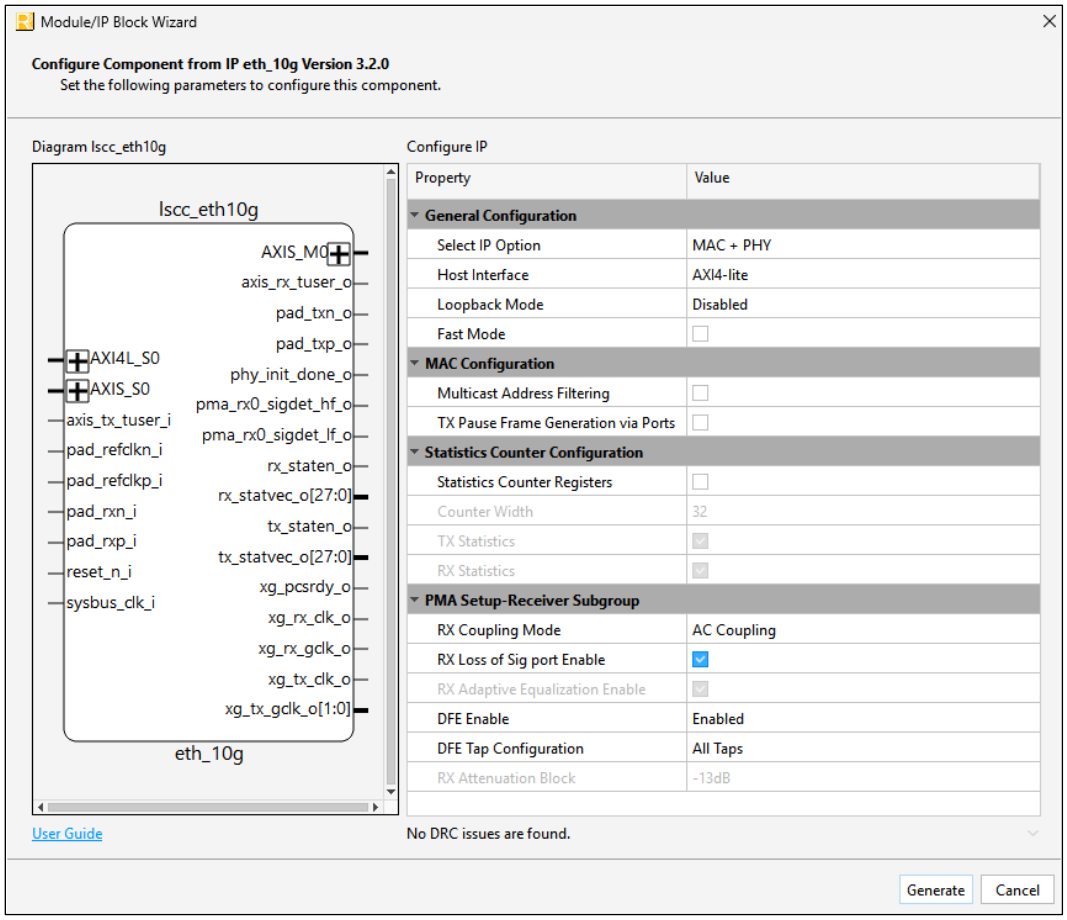


Figure 4.19. 10G Ethernet IP GUI

4.2. Reference Design Top File

The MIPI CSI-2 to Ethernet reference design includes the parameters shown in Table 4.10. You can modify the parameters by editing the *fpga_top.sv* file.

Table 4.10. Parameters in fpga_top.sv

Parameter	Default Value	Description
ETH_EN	1	This parameter is used to speed up simulation time by bypassing the Ethernet: • 0: Disables the Ethernet. The AXI-Stream TX interface loops back directly to the AXI-Stream RX interface, bypassing the full Ethernet path. • 1: Enables the full Ethernet MAC and PHY for 25G or 10G operation. Its value is overridden by simulation testbench tb_top to 0 by default.
SIM	0	No changes are needed. For simulation, make sure simulation top module is selecting tb_top as shown in Figure 7.2, which will overwrite this parameter.
`include "rd_params.svh"	4K	To change the resolution to 4K or 1080p, go to the file location *RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\rtl\rd_params.svh*, define only IP_DIR_1080P to select 1080p video resolution, or define only IP_DIR_4K to select 4K video resolution. Example: 4K: `//`define IP_DIR_1080P` `define IP_DIR_4K`

Parameter	Default Value	Description
		1080p: `define IP_DIR_1080P //`define IP_DIR_4K
ETH_RATE	1	0: To select 10G Ethernet data rate for reference design. 1: To select 25G Ethernet data rate for reference design. Detailed steps are shown in Figure 6.5.

5. Signal Description

The input or output interface signals for the reference design top level module (*fpga_top.sv*) are shown in Table 5.1.

Table 5.1. fpga_top.sv I/O

Port Name	I/O	Width	Description
reset_n_i	In	1	Active low asynchronous system reset.
clk_i	In	1	27 MHz.
mcddr_refclk_i	In	1	100 MHz - go straight to ddr_controller pll.
cam_0_scl_io	Inout	1	Image sensor I2C configuration serial clock lane for camera 0.
cam_0_sda_io	Inout	1	Image sensor I2C configuration serial data lane for camera 0.
clk_p_0_io	Inout	1	MIPI D-PHY positive clock lane for camera 0.
clk_n_0_io	Inout	1	MIPI D-PHY negative clock lane for camera 0.
d_p_0_io	Inout	4	MIPI D-PHY positive data lane for camera 0.
d_n_0_io	Inout	4	MIPI D-PHY negative data lane for camera 0.
cam_0_rst_n_o	Out	1	Output reset Camera 0 sent to Modular FMC Adapter.
cam_0_clk_o	Out	1	Output clock Camera 0 sent to Modular FMC Adapter.
cam_1_scl_io	inout	1	Image sensor I2C configuration serial clock lane for camera 1.
cam_1_sda_io	Inout	1	Image sensor I2C configuration serial data lane for camera 1.
clk_p_1_io	Inout	1	MIPI D-PHY positive clock lane for camera 1.
clk_n_1_io	Inout	1	MIPI D-PHY negative clock lane for camera 1.
d_p_1_io	Inout	4	MIPI D-PHY positive data lane for camera 1.
d_n_1_io	Inout	4	MIPI D-PHY negative data lane for camera 1.
cam_1_rst_n_o	Out	1	Output reset Camera 1 sent to Modular FMC Adapter.
cam_1_clk_o	Out	1	Output clock Camera 1 sent to Modular FMC Adapter.
ddr_dq_io	Inout	16	LPDDR4 DQ signal.
ddr_dqs_io	Inout	2	LPDDR4 DQS signal.
ddr_dmi_io	Inout	2	LPDDR4 DMI signal.
ddr_ck_o	Out	1	LPDDR4 CK signal.

Port Name	I/O	Width	Description
ddr_cke_o	Out	1	LPDDR4 CKE signal.
ddr_cs_o	Out	1	LPDDR4 CS signal.
ddr_ca_o	Out	6	LPDDR4 CA signal.
ddr_odt_ca_o	Out	1	LPDDR4 CA signal.
ddr_reset_n_o	Out	1	LPDDR4 Memory reset signal.
hdmi_scl_io	Inout	1	HDMI I2C configuration serial clock lane.
hdmi_sda_io	Inout	1	HDMI I2C configuration serial data lane.
hdmi_clk_o	Out	1	Output clock sent to HDMI FMC board.
hdmi_rst_n_o	Out	1	Output reset sent to HDMI FMC board.
data_o	Out	24	HDMI output data payload.
zeros_o	Out	12	Stream of zeros sent to HDMI along with data.
vsync_o	Out	1	HDMI output vertical sync signal.
hsync_o	Out	1	HDMI output horizontal sync signal.
de_o	Out	1	HDMI output data enable.
dbg_led_o	Out	3	Debug LED.
sfp_refclkp_i	In	1	156.25 MHz SERDES PLL reference clock signal (+).
sfp_refclkn_i	In	1	156.25 MHz SERDES PLL reference clock signal (-).
sfp1_txp_o	Out	1	Ethernet SFP TX+ differential signal.
sfp1_txn_o	Out	1	Ethernet SFP TX- differential signal.
sfp1_rxp_i	In	1	Ethernet SFP RX+ differential signal.
sfp1_rxn_i	In	1	Ethernet SFP RX- differential signal.

5.1. Push Button

Table 5.2. Push Button

Port Name	Board Reference	Description
reset_n_i	SW12	0 (Push): Assert global reset. 1 (Release): Deassert global reset.
board_id_tg_i	SW13	Press to change the ID of the board. The selected board ID is displayed in Digit 1 of 7-segment LED. The board ID selection will toggle in following sequence when this push button is pressed. 0 → 1 → 2 → 3 → 0 The board ID is used as: • Unicast MAC address of Ethernet port. • Multicast MAC address of AVTP frames. • Stream ID of camera sensors. • Camera 1: Board ID x 2. • Camera 2: Board ID x 2 + 1. Example: If Board ID is 3 Camera 1 ID = 6 Camera 2 ID = 7
video_src_tg_i	SW14	Press to change the selection of video source ID to be displayed. The selected video source ID is displayed in Digit 2 of 7-segment LED. The video source ID selection will toggle in the following sequence when this push button is pressed. 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → L0 → L1 → 0 0 to 7: Video source from network. L0, L1: Video source from local camera sensors. The selected video source ID from network is displayed in Digit 2 of 7-segment LED. L0 is displayed with alphabet 'L' in Digit 2 with DP LED off. L1 is displayed with alphabet 'L' in Digit 2 with DP LED on.

5.2. 7-Segment LED

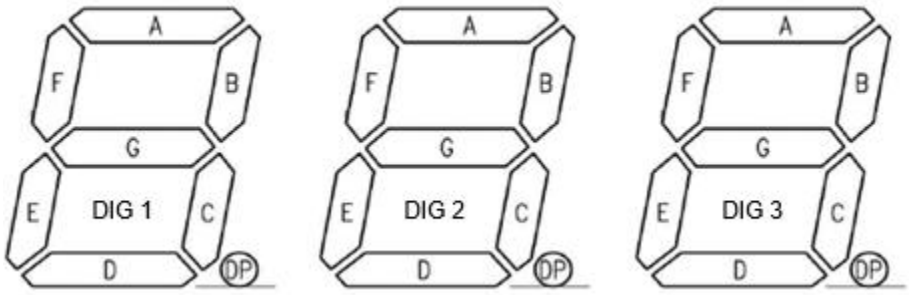


Figure 5.1. 7-Segment LED

Table 5.3. 7-Segment LED

Digit	Segment	Description
1	A	Display the ID of the board selected by board_id_tg_i push button.
	B	
	C	
	D	
	E	
	F	
	G	
	DP	Unused.
2	A	Display the ID of the video stream source selected by video_src_tg_i push button.
	B	
	C	
	D	
	E	
	F	
	G	
	DP	N/A
3	A	SFP1 – FCS error • Off: No error. • Blink: Error occurs currently. • On: Error received previously (latched status).
	B	SFP1 – Frames are received. • Off: No frames are received. • Blink: Frame are received.
	C	Segment lost error – AVTP frame carrying segment of video data missing or experience unexpected large delay • Off: No error. • Blink: Error occurs currently. • On: Error received previously (latched status). Latched status clear if video source has changed.
	D	Store & Forward FIFOs error – FIFO full • Off: No error • Blink: Error occurs currently • On: Error received previously (latched status)
	E	Ethernet SC FIFOs error – FIFO full • Off: No error. • Blink: Error occurs currently. • On: Error received previously (latched status).
	F	SFP1 – Frames transmitted • Off: No frames are transmitting. • Blink: Frames are transmitting.
	G	SFP1 – PCS is ready • No glow: Link is down. • Blink: Link is not stable. • On: Link is up.
	DP	Out of sequence error – Sequence ID of received AVTP frame missing or out of order. • No glow: No error. • Blink: Error occurs currently. • On: Error received previously (latched status). Latched status clear if video source has changed.

6. Running Reference Design

This section describes how to run the MIP CSI-2I to Ethernet reference design using the Lattice Radiant software. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

6.1. Compiling the Reference Design

The following steps guide you through the implementation of the design. This reference design includes two implementations (1080p and 4K), and two Ethernet data rate selection. The 4K design will be used as an example to describe the compilation steps. The steps are the same for compiling the 1080p design.

1. Select 4K resolution:
   a. To select the 4K resolution as the reference design, modify the parameter in the project folder *RD02311_MIPI_CSI2_Ethernet\fpga_lavat\radiant\src\rtl\rd_params.svh*.
   b. Comment out line 5 and uncomment line 4.

```
3   // Use either IP_DIR_4K for 4K video resolution, or use IP_DIR_1080P for 1080p video resolution
4   `define IP_DIR_4K
5   //`define IP_DIR_1080P
```

Figure 6.1. Enable 4K IP Collateral Files in rd_params.svh Folder

2. Launch the Lattice Radiant software, as shown in Figure 6.2.

   Note that the project is compiled using the Radiant software version 2025.1.0.39.0.

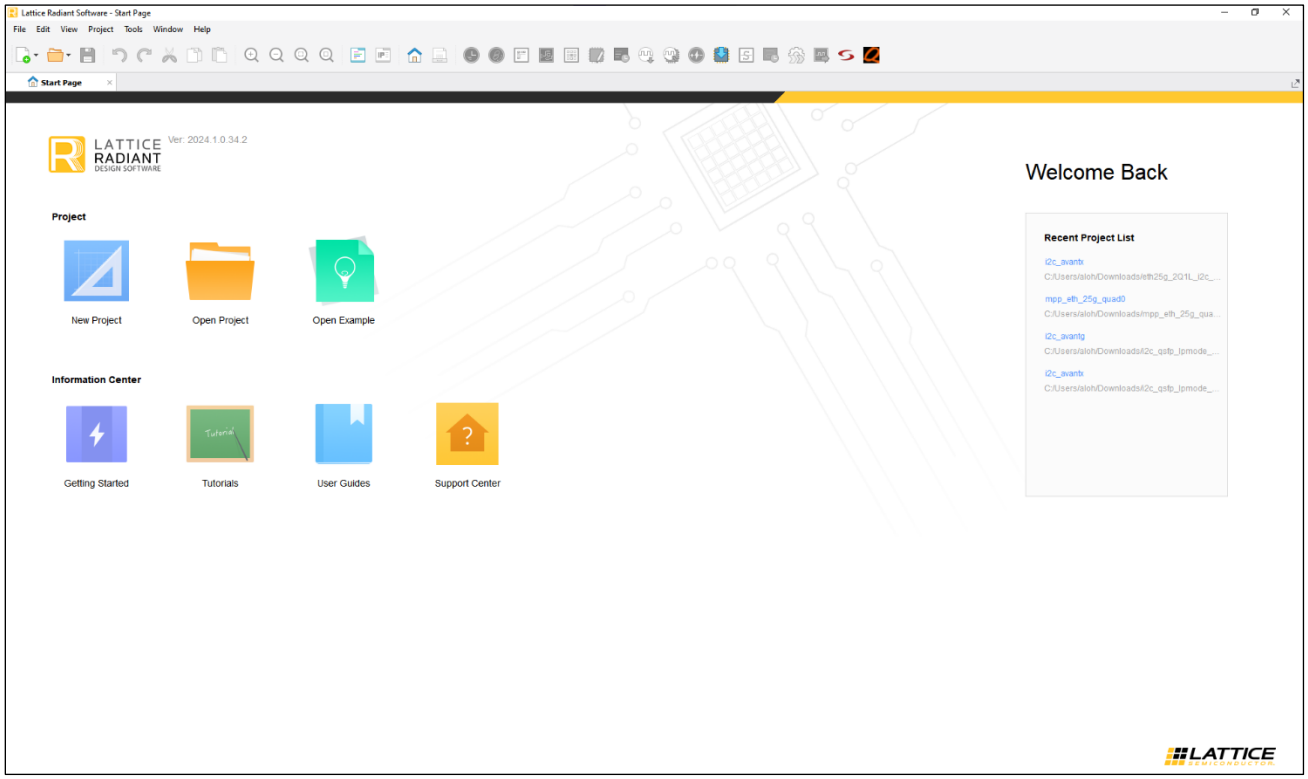


Figure 6.2. Lattice Radiant Software

3. To open the project, follow these steps:
   a. Click **File > Open Project**.
   b. From the project database, open the Radiant software project file (*MIPI_ETH_RD.rdf*), as shown in Figure 6.3.

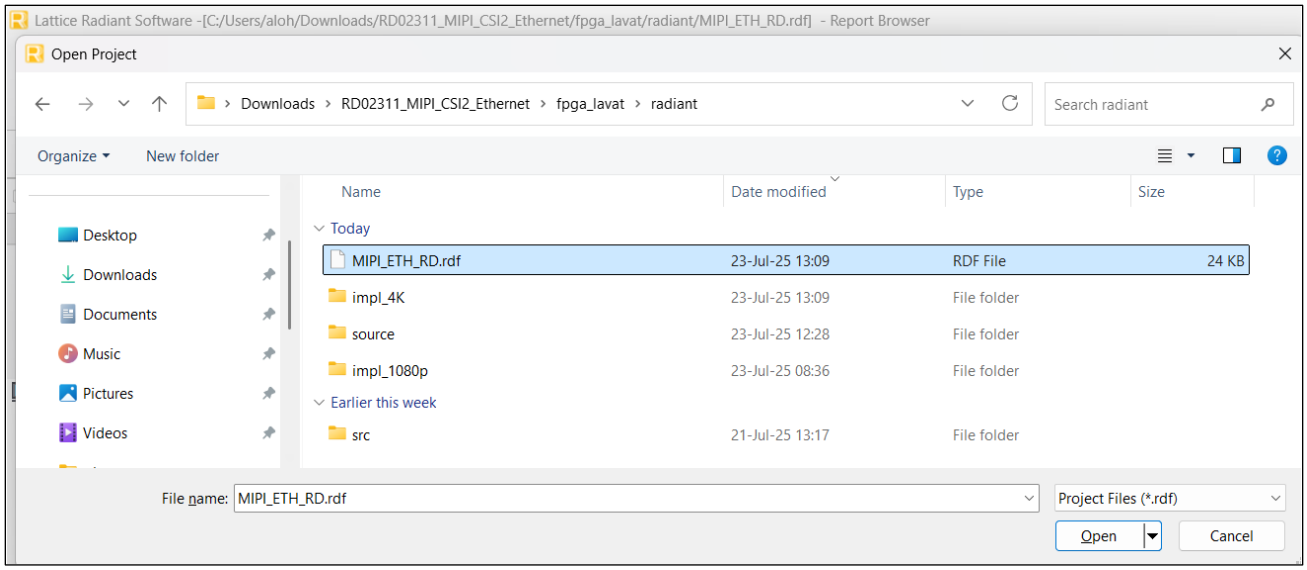


Figure 6.3. Open Project File

4. To set the Active Implementation, follow these steps:
   a. On the left side of the project file list interface, you will see two implementations listed.
   b. Right-click on **impl_4k** and select **Set as Active Implementation**.
   c. The active implementation design appears in bold.

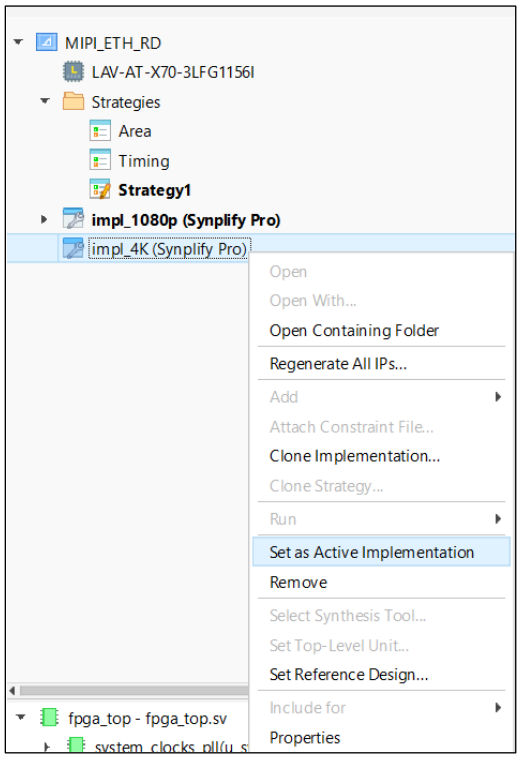


Figure 6.4. Select Active Implementation

5. To configure the Ethernet data rate used in this reference design, follow these steps:
   a. In the active implementation project, navigate to the list of design files containing the modules and instantiated IP cores.
   b. Open the *fpga_top.sv* file.
   c. Within the top-level module, locate the parameter named **ETH_RATE**.
   d. Set **ETH_RATE** to '0' to enable the **10G Ethernet** data rate or leave it at the default value of '1' to operate with the **25G Ethernet** data rate.

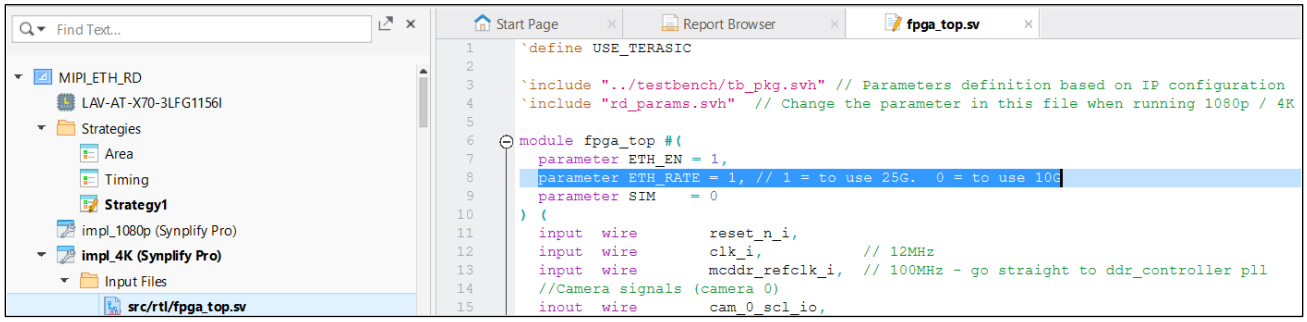


Figure 6.5. Select Ethernet Data Rate in the fpga_top.sv File

Notes:
1. Select the appropriate implementation and data rate based on your desired resolution, and configure the parameters accordingly in *rd_params.svh* and *fpga_top.sv*.
2. Before generating the bitstream, ensure that the timing constraint file *(constraints_versa_XX.pdc)* is updated to match the selected data rate as shown in Figure 6.6.

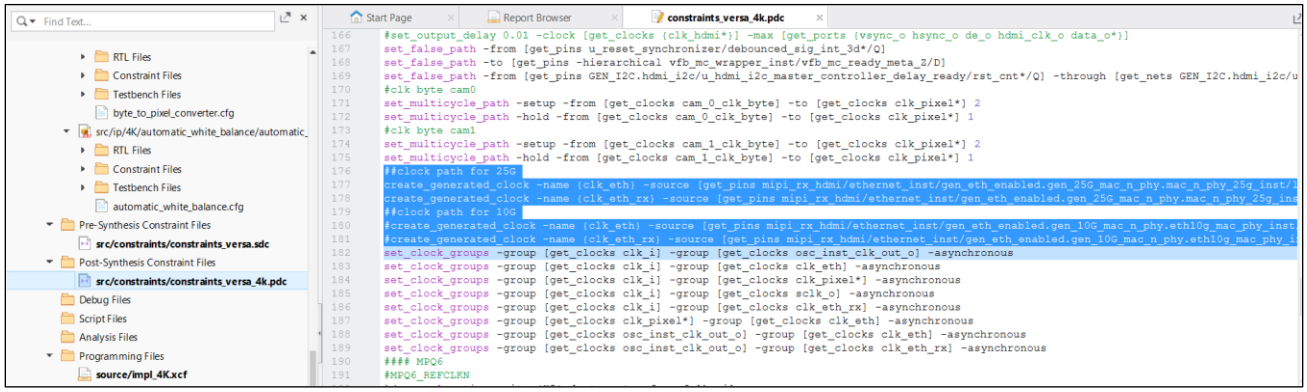


Figure 6.6. Clock Path Adjustment Based on Selected Data Rate in constraints_versa_XX.pdc

6.2. Generating the Bitstream File

This section provides the procedure to create your FPGA bitstream file using the Lattice Radiant software.

To create the FPGA bitstream file, follow these steps:

1. Click **Export Files** to generate the bit file. View the log message in the Export Reports folder for the generated bitstream.

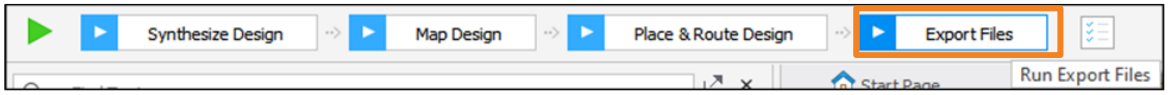


Figure 6.7. Generate and Export Bitstream File

Note: Four pre-generated bitstreams are available for immediate programming onto the board. These can be found in the following directories:

RD02311_MIPI_CSI2_Ethernet\fpga_lavat\precompiled_file

Table 6.1. Pre-generated Bitstream Directories

Video Resolution	Ethernet Rate	Bitstream
4K	10G	MIPI_ETH_RD_impl_4K_10G.bit
	25G	MIPI_ETH_RD_impl_4K_25G.bit
1080p	10G	MIPI_ETH_RD_impl_1080p_10G.bit
	25G	MIPI_ETH_RD_impl_1080p_25G.bit

7. Simulating Reference Design

To simulate the design, follow these steps:

1. Open the reference design project file (*MIPI_ETH_RD.rdf*) using the Lattice Radiant software.
2. To launch the Simulation Wizard, click on **Tools > Simulation Wizard**.
3. To configure the simulation project, enter project name and click **Next**.

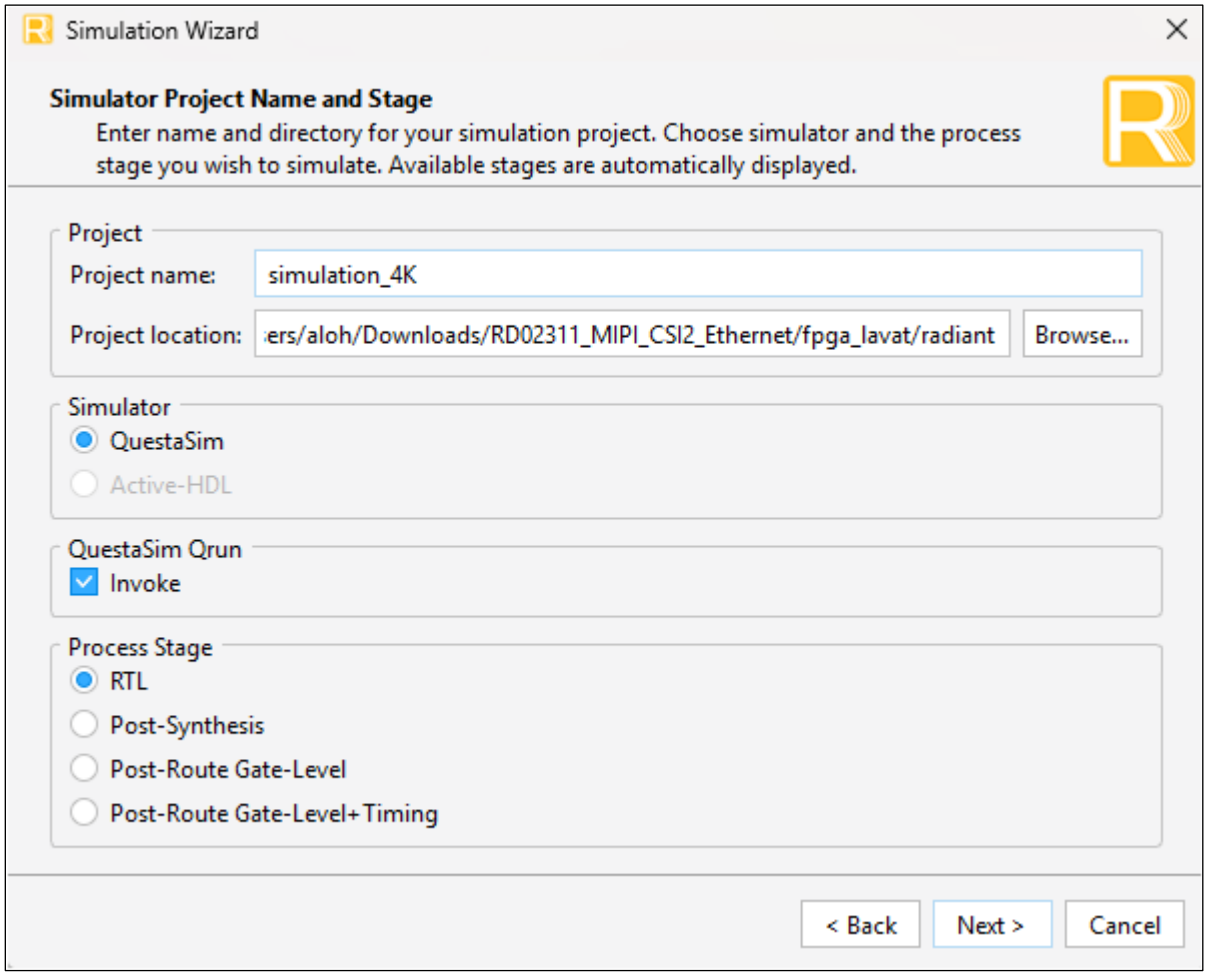


Figure 7.1. Simulation Wizard: Create Simulation Project

Note: Ensure the simulation project is created in the same location as the reference design project location (*MIPI_ETH_RD.rdf*) to prevent data fetching issues due to the csi2_model location not being found.

4. Set the simulation Top Module to *tb_top* as testbench and click **Next**.

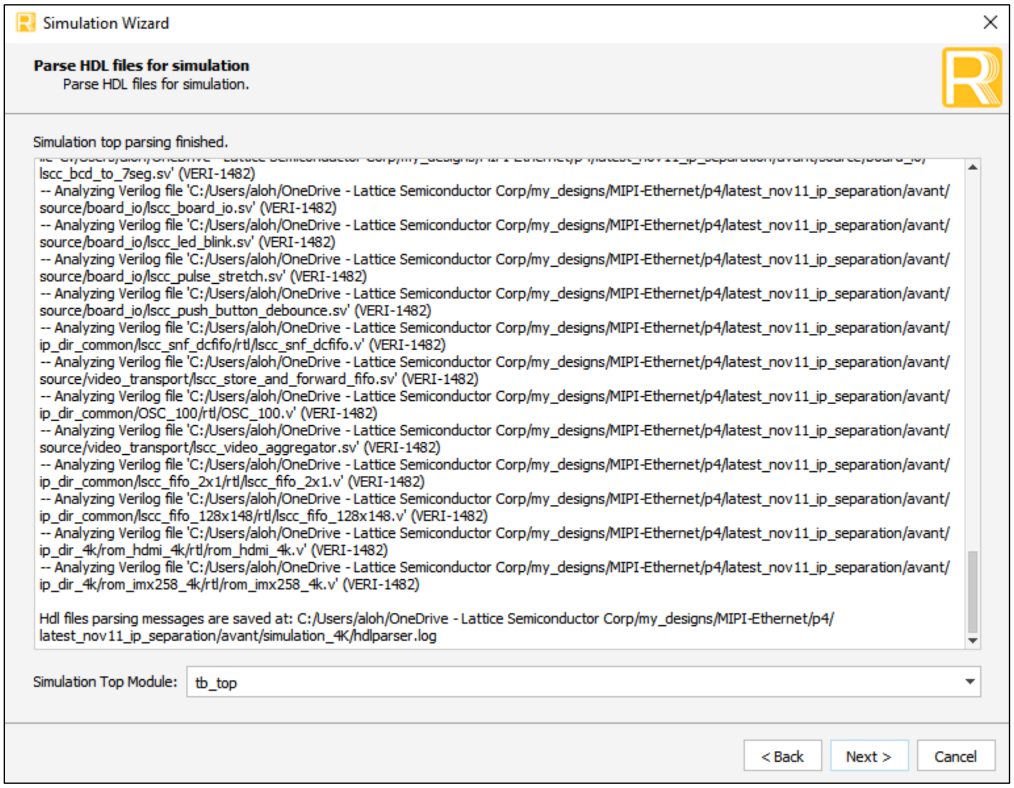


Figure 7.2. Simulation Wizard: Select Simulation Top Module

5. Select the configuration as shown in Figure 7.3 and click **Finish**.

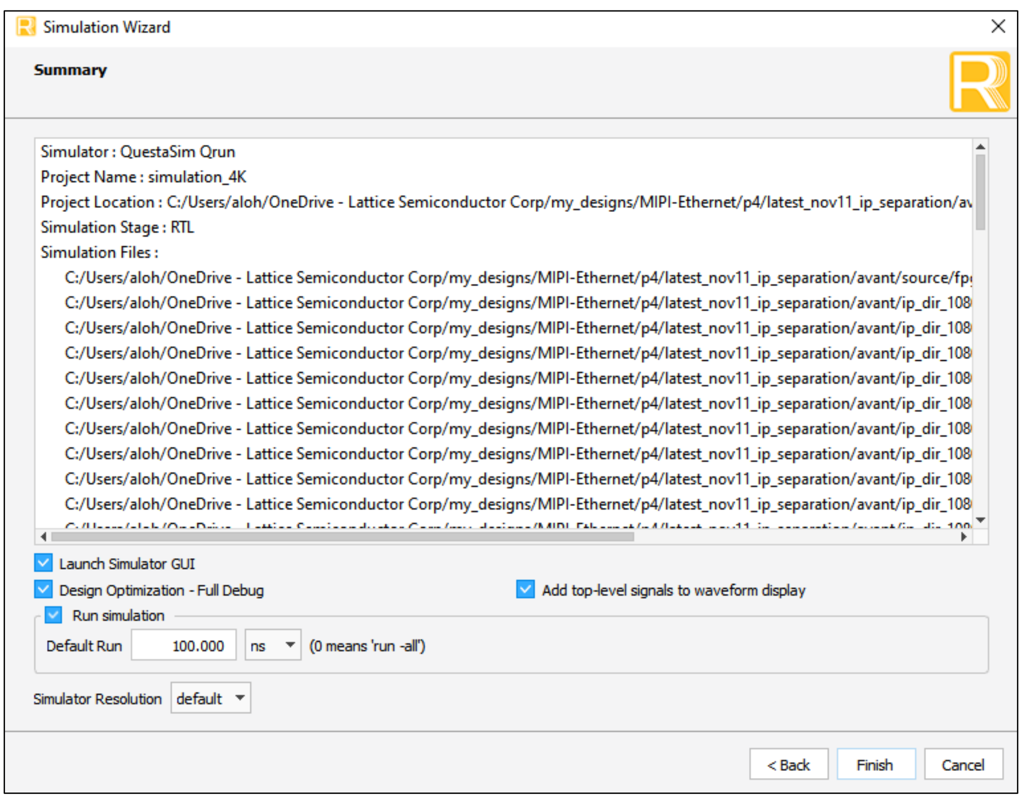


Figure 7.3. Simulation Wizard: Summary Page

6. Wait until the QuestaSim Lattice FPGA Edition software loads.
   Note: If you encounter an error while executing vsim, as shown in Figure 7.4., go to the simulation folder, open the transcript file, copy the *qrun* command, and paste it into the QuestaSim prompt.

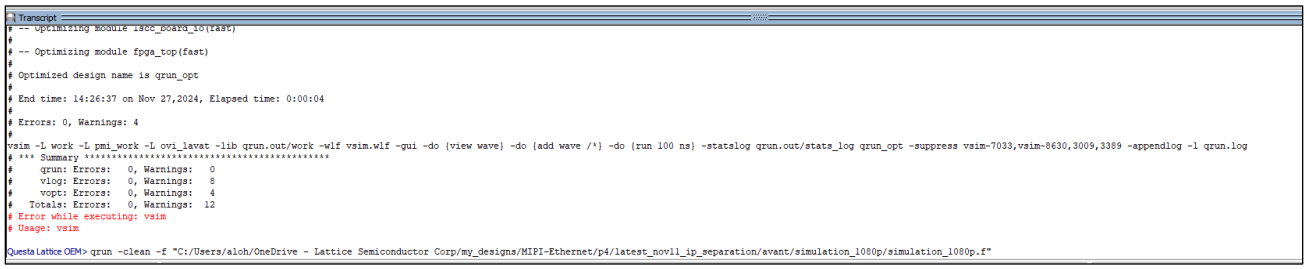


Figure 7.4. Error while Executing vsim

7. Go to the **Wave** window from the **View** menu.

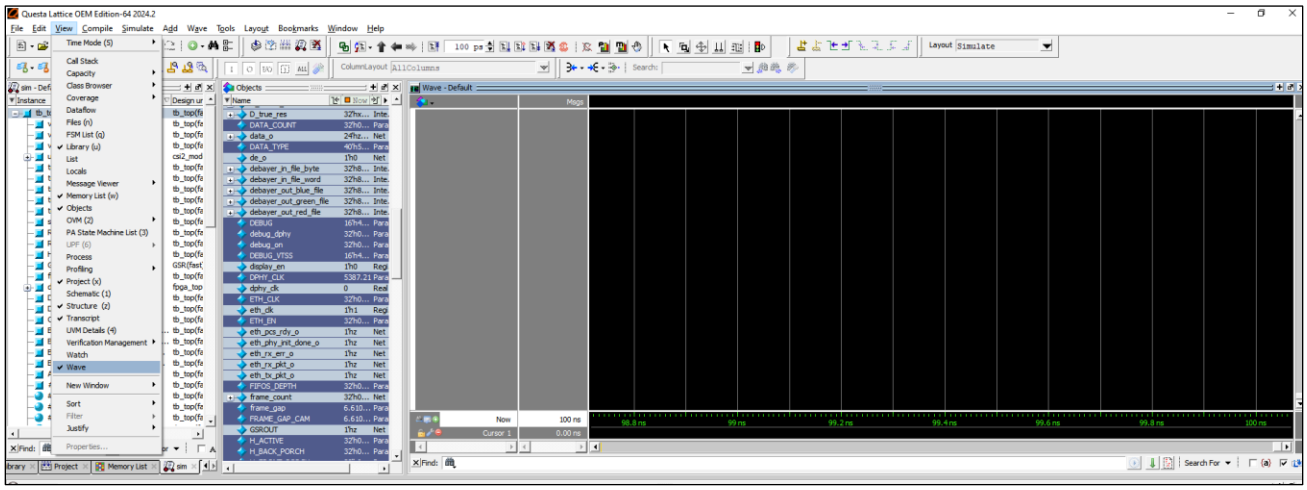


Figure 7.5. QuestaSim Interface

8. To load the *.do* file, click on **Tools** > **TCL** > **Execute Macro**, and select the *wave.do* file from simulation_4K folder.

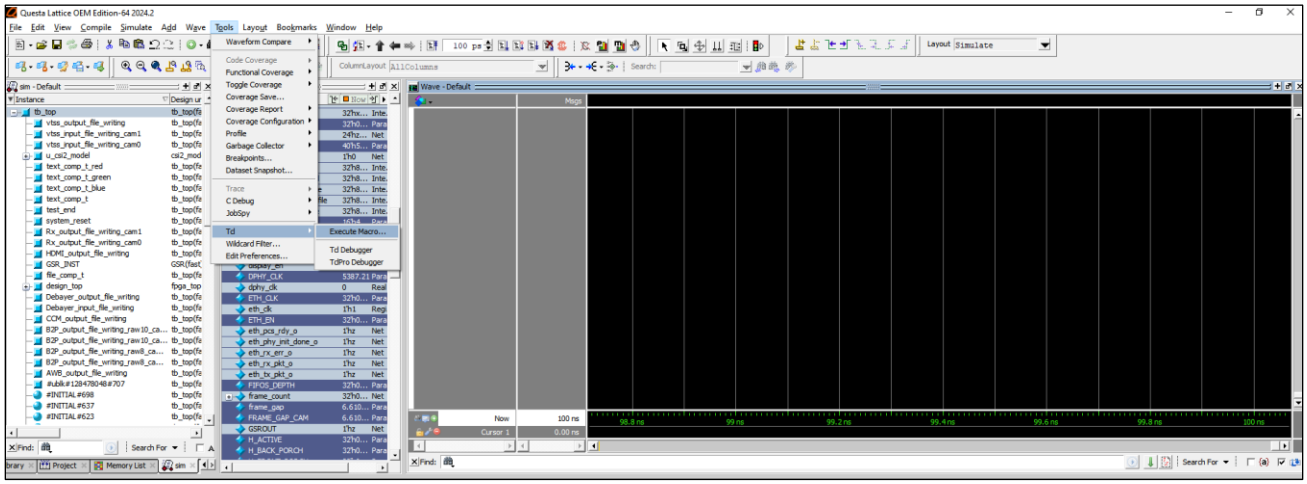


Figure 7.6. QuestaSim Interface: Execute wave.do File

9. The added signals are displayed in the waveform window.

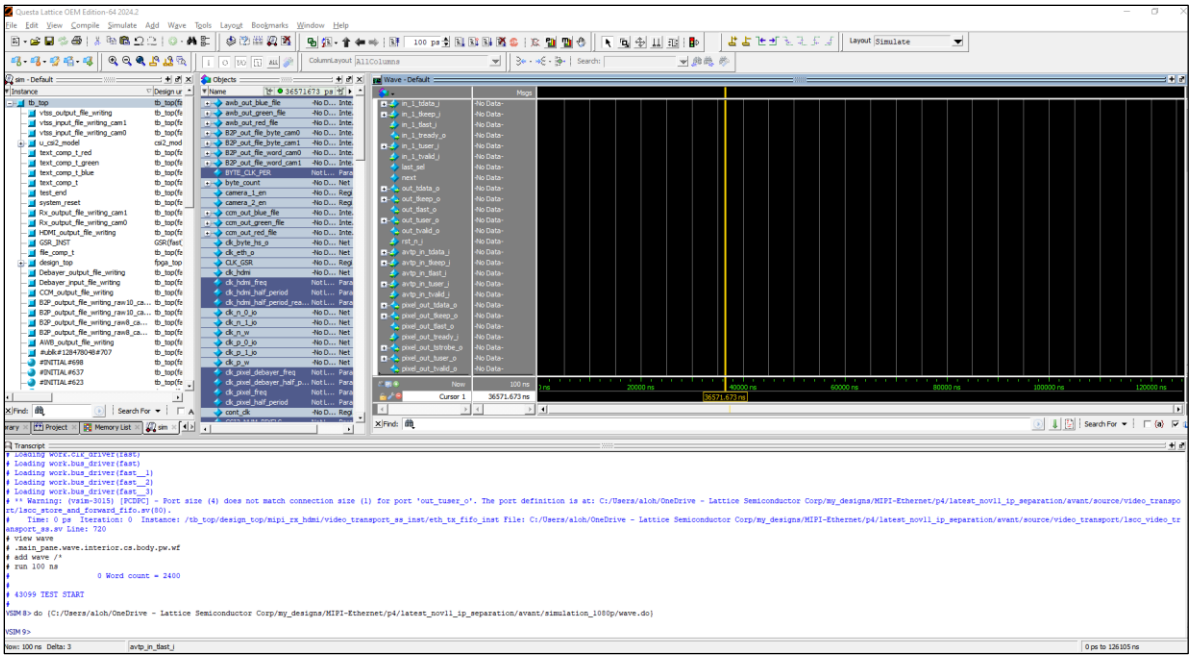


Figure 7.7. QuestaSim Interface: Signals Added

10. To run the simulation, type restart, *log -r /**, and *run 150us* in the Transcript window.

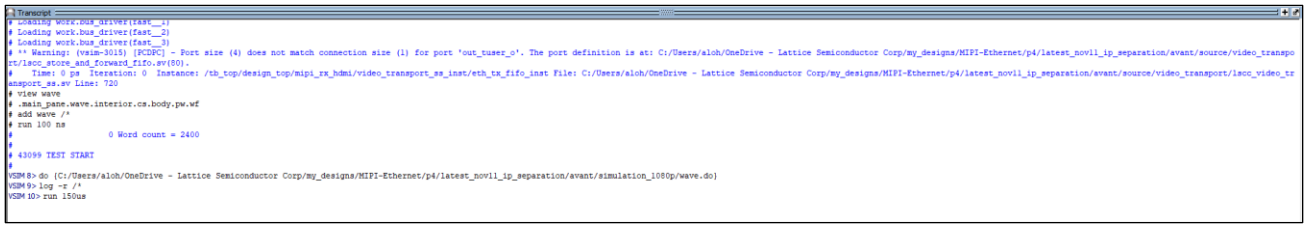


Figure 7.8. QuestaSim Interface: Restart Simulation

11. Wait for the simulation to complete to see the waveforms.

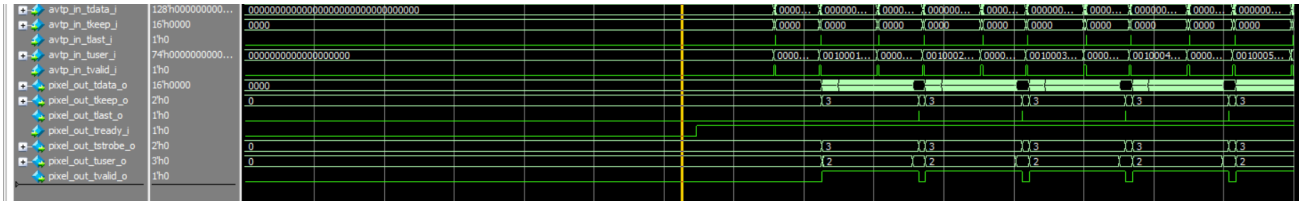


Figure 7.9. QuestaSim Interface: Passing Condition

12. Type *quit -sim* in the ModelSim console to terminate the simulation.

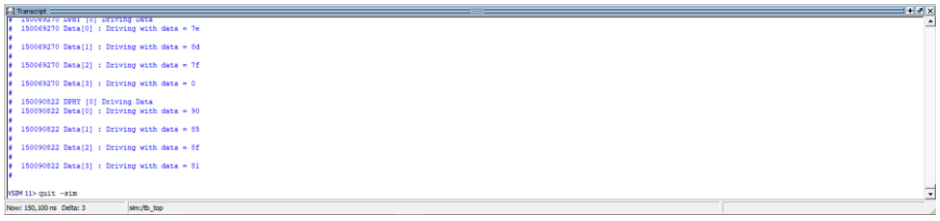
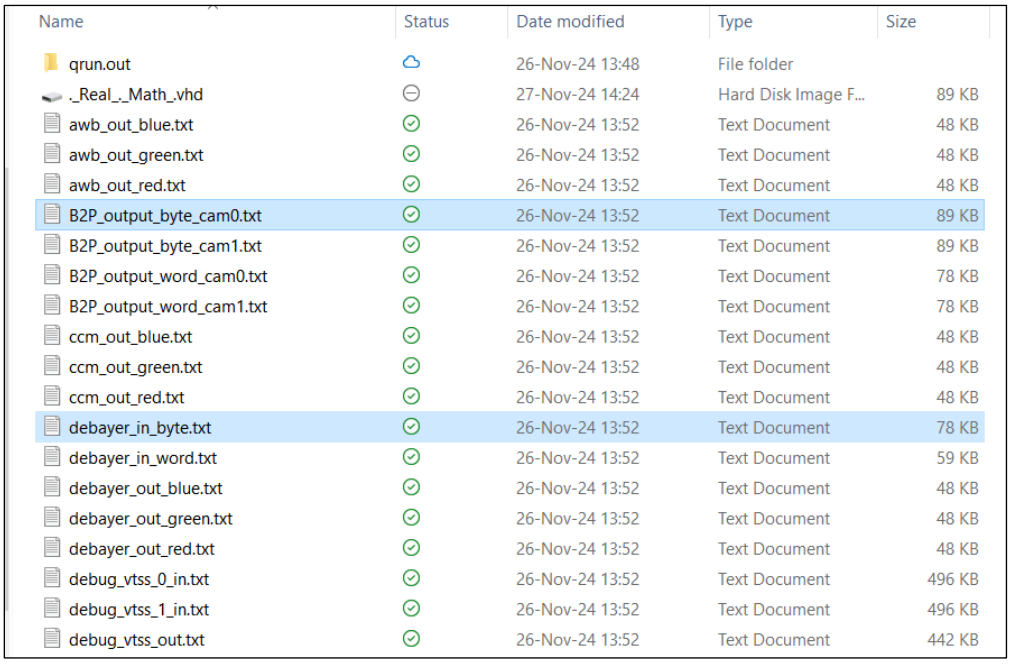


Figure 7.10. QuestaSim Interface: Quit Simulation

7.1. Simulation Results

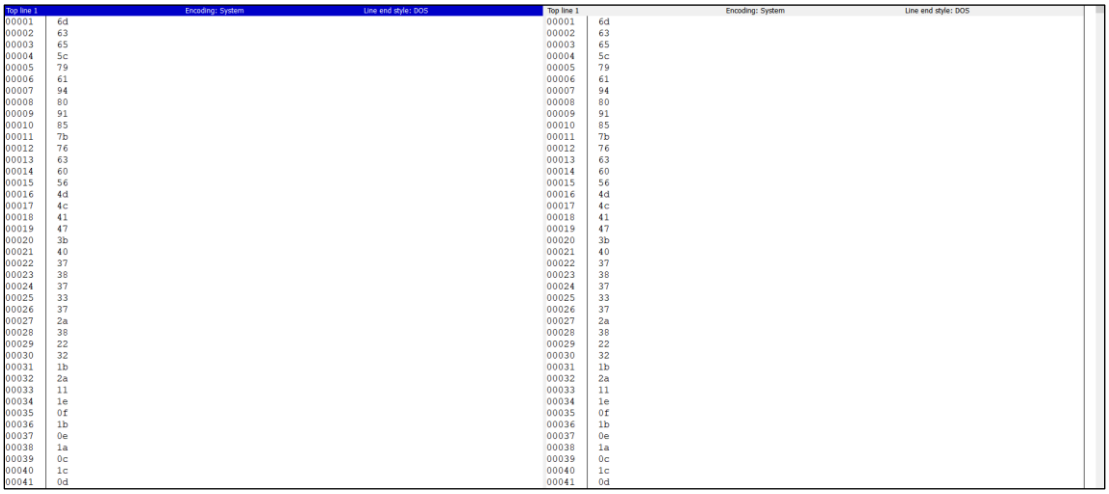
1. After the simulation is completed, the simulation folder contains the results generated in *.txt*.



Name	Status	Date modified	Type	Size
qrun.out		26-Nov-24 13:48	File folder	
._Real_._Math_.vhd		27-Nov-24 14:24	Hard Disk Image F...	89 KB
awb_out_blue.txt		26-Nov-24 13:52	Text Document	48 KB
awb_out_green.txt		26-Nov-24 13:52	Text Document	48 KB
awb_out_red.txt		26-Nov-24 13:52	Text Document	48 KB
B2P_output_byte_cam0.txt		26-Nov-24 13:52	Text Document	89 KB
B2P_output_byte_cam1.txt		26-Nov-24 13:52	Text Document	89 KB
B2P_output_word_cam0.txt		26-Nov-24 13:52	Text Document	78 KB
B2P_output_word_cam1.txt		26-Nov-24 13:52	Text Document	78 KB
ccm_out_blue.txt		26-Nov-24 13:52	Text Document	48 KB
ccm_out_green.txt		26-Nov-24 13:52	Text Document	48 KB
ccm_out_red.txt		26-Nov-24 13:52	Text Document	48 KB
debayer_in_byte.txt		26-Nov-24 13:52	Text Document	78 KB
debayer_in_word.txt		26-Nov-24 13:52	Text Document	59 KB
debayer_out_blue.txt		26-Nov-24 13:52	Text Document	48 KB
debayer_out_green.txt		26-Nov-24 13:52	Text Document	48 KB
debayer_out_red.txt		26-Nov-24 13:52	Text Document	48 KB
debug_vtss_0_in.txt		26-Nov-24 13:52	Text Document	496 KB
debug_vtss_1_in.txt		26-Nov-24 13:52	Text Document	496 KB
debug_vtss_out.txt		26-Nov-24 13:52	Text Document	442 KB

Figure 7.11. Simulation Results

2. Compared the input file (*debayer_in_byte.txt*) and output file of camera 0 (*B2P_output_byte_cam0.txt*) or camera 1 (*B2P_output_byte_cam1.txt*). The results must be identical for both files.



Top line 1	Encoding: System	Line end style: DOS	Top line 1	Encoding: System	Line end style: DOS
00001	6d		00001	6d	
00002	63		00002	63	
00003	65		00003	65	
00004	5c		00004	5c	
00005	79		00005	79	
00006	61		00006	61	
00007	94		00007	94	
00008	80		00008	80	
00009	91		00009	91	
00010	85		00010	85	
00011	7b		00011	7b	
00012	76		00012	76	
00013	63		00013	63	
00014	60		00014	60	
00015	56		00015	56	
00016	4d		00016	4d	
00017	4c		00017	4c	
00018	41		00018	41	
00019	47		00019	47	
00020	3b		00020	3b	
00021	40		00021	40	
00022	37		00022	37	
00023	38		00023	38	
00024	37		00024	37	
00025	33		00025	33	
00026	37		00026	37	
00027	2a		00027	2a	
00028	38		00028	38	
00029	22		00029	22	
00030	32		00030	32	
00031	1b		00031	1b	
00032	2a		00032	2a	
00033	11		00033	11	
00034	1e		00034	1e	
00035	0f		00035	0f	
00036	1b		00036	1b	
00037	0e		00037	0e	
00038	1a		00038	1a	
00039	0c		00039	0c	
00040	1c		00040	1c	
00041	0d		00041	0d	

Figure 7.12. Passing Results for Camera 0

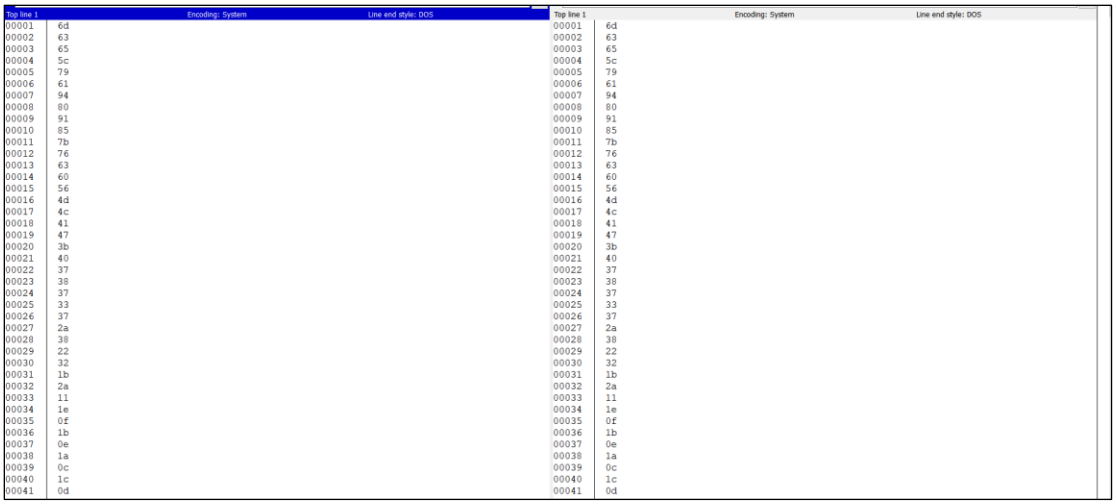


Figure 7.13. Passing Results for Camera 1

8. Implementing the Reference Design on Board

8.1. Requirements

The following lists the requirements for the reference design implementation on board:

- Lattice Avant G/X ES
- USB cable for programming
- Power supply
- Optical fiber cables
- 25G Ethernet SFP28 module
- 10G Ethernet SFP+ module
- HDMI cable
- 1080p 60fps or 4K 30fps capable monitor
- 2 IMX258 camera sensor
- Third-party HDMI card
- Optical loopback module

8.2. Device Hardware

This section describes the device hardware and setup on the board for the reference design implementation—Lattice Avant Versa Board X.

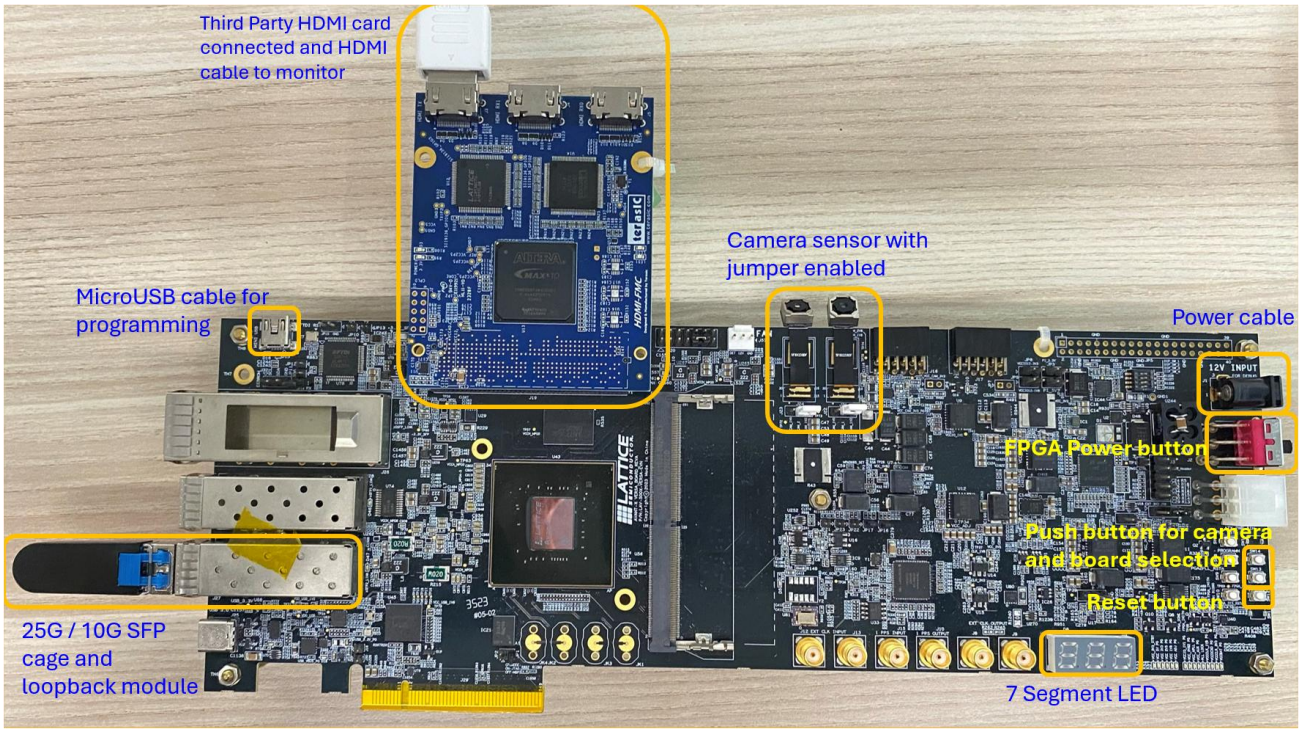


Figure 8.1. Overview Hardware Setup

8.2.1. Camera Sensor

To enable the sensor, you must connect the jumper and short pin 2-3 respectively on J24 and J21.

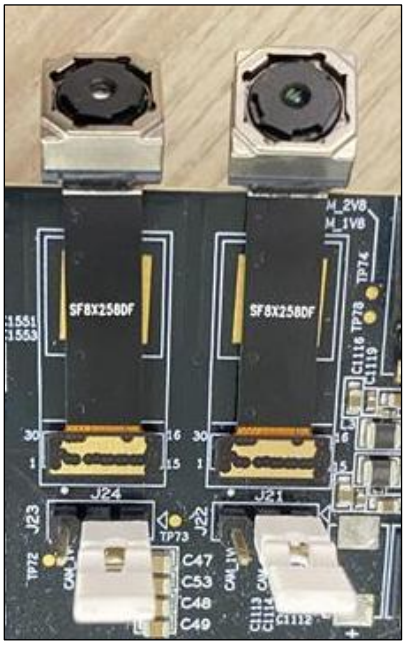


Figure 8.2. Camera Sensor Setup

8.2.2. Third-Party HDMI Card

Attach the HDMI card onto J19 pin, FPGA Mezzanine Card Plus (FMC+) connector on the Lattice Avant Versa board, connect the HDMI cable to J3, and connect the HDMI TX connector on the HDMI card.

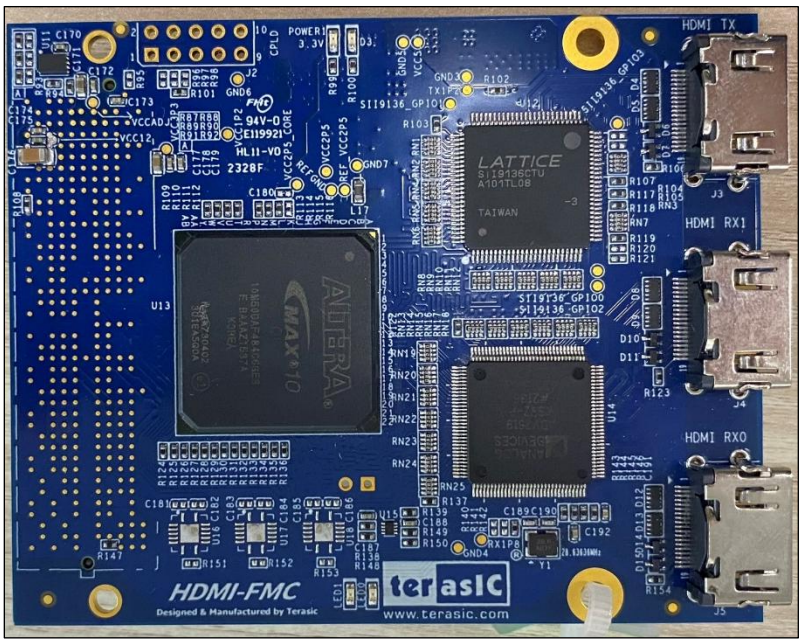


Figure 8.3. Third-Party HDMI Setup

8.3. Board Testing

This section provides the details of the reference design hardware testing on the board.

8.3.1. Bit File Programming

This section guides you through the process of uploading the *.bit* file.

Make sure the board is connected to the PC with the mini–USB Type-A cable, as shown in Figure 8.4.

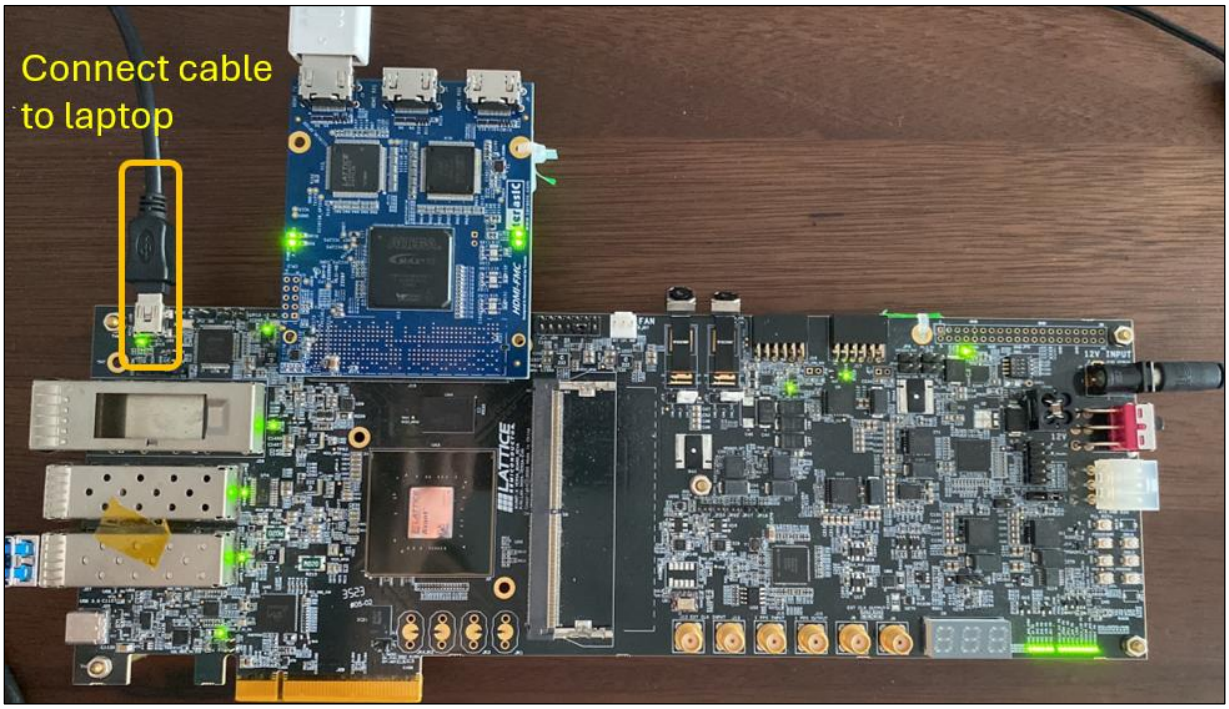


Figure 8.4. Avant Versa Board

1. Click on the Radiant Programmer icon ( ) if the project is already opened inside the Radiant software. Otherwise, launch the standalone Radiant Programmer. Enter the **Project Name**, **Project Location**, and click **OK**. The Radiant Programmer window is shown in Figure 8.5.

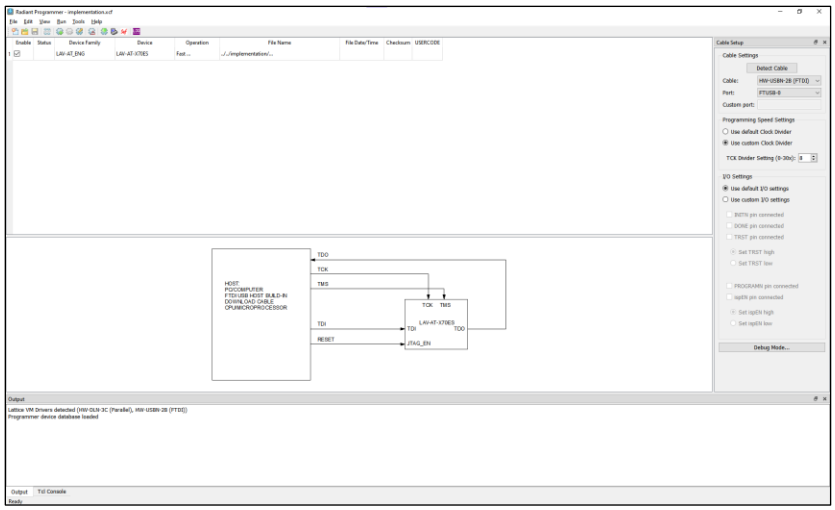


Figure 8.5. Radiant Programmer Window

2. On the cable setup branch, click on **Detect Cable** and select **FTUSB-1** and click **OK** to proceed.

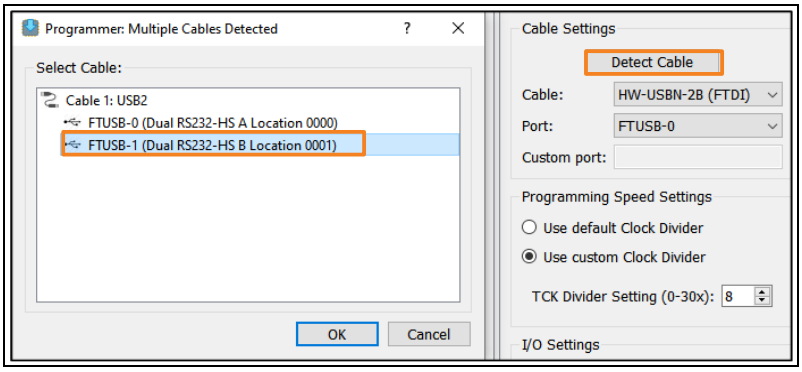


Figure 8.6. Select Cable Settings

3. Click on the **…** button under the **File Name** bar and select the bitstream file to program. Refer to Table 6.1. to choose the appropriate pre-generated bitstream for demonstrating the design.

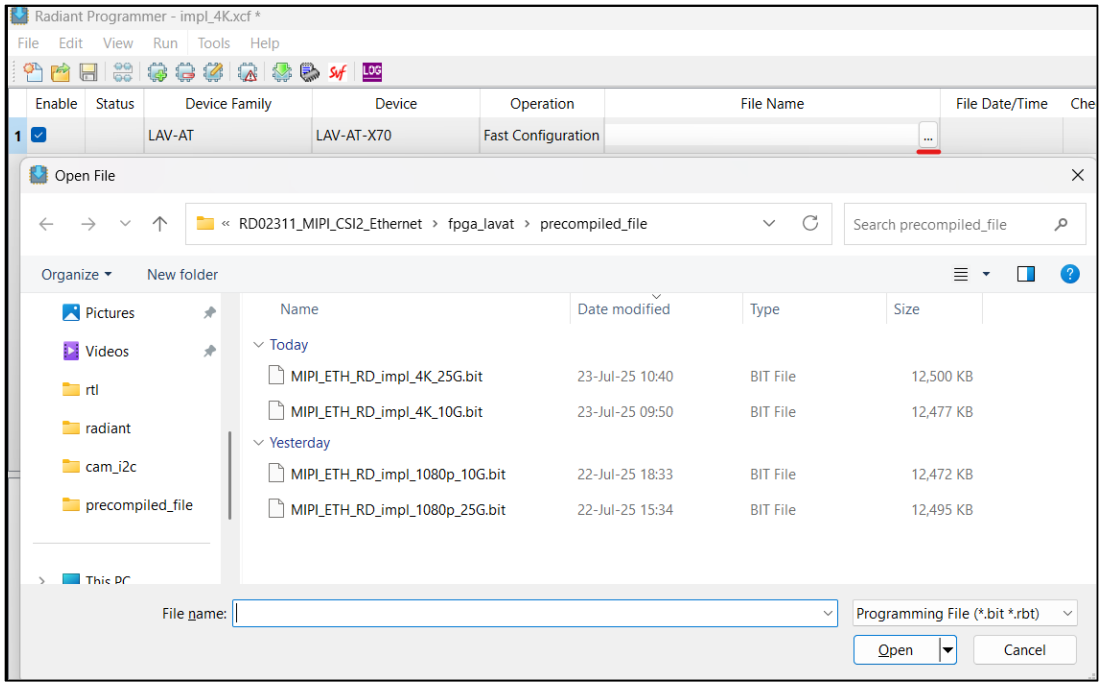


Figure 8.7. Load Bitstream File

4. Click on the program device toolbar icon to load the bitstream onto the board as shown in Figure 8.8.

Figure 8.8. Program Device Toolbar Icon

5. On the output window, verify that the status of the program device is "*Operation: successful*", as shown in Figure 8.9.

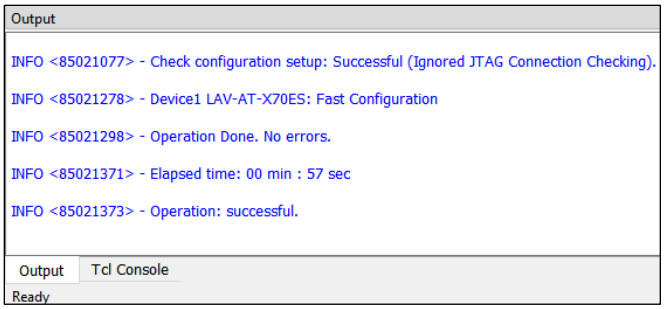


Figure 8.9. Message on Successful Programming

8.3.2. External Loopback Setup

1. Connect the optical cable back-to-back (TX-RX), as shown in Figure 8.10.

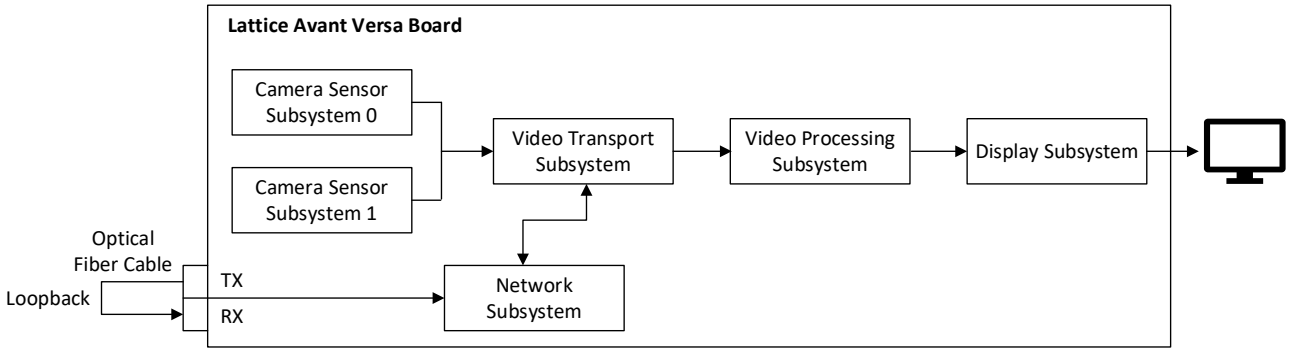


Figure 8.10. External Loopback Connection

2. After successfully programming the bitstream onto the board, you will see the third digit of the seven-segment display (D) blinking. Press the reset button (SW12) to reset the reference design.

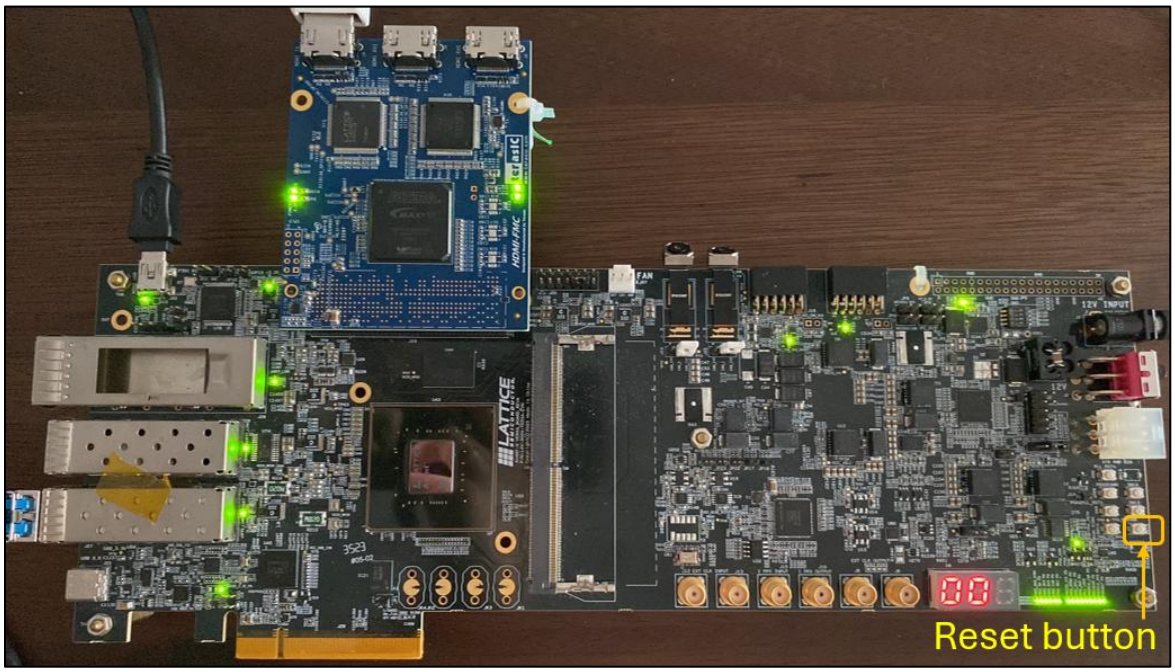


Figure 8.11. Press Reset Button to Initialize the Reference Design

3. After the reset, the design selects camera 0 by default. As shown in Figure 8.12, camera 1 is arranged vertically for a better view. On the three-digit seven-segment LED, you will see TX and RX transactions, indicating that the video is successfully captured and displayed on the monitor. The results are as shown in Figure 8.13.
4. To switch to camera 1, press SW14 and observe the difference. Figure 8.14 shows the monitor displaying video from camera 1.

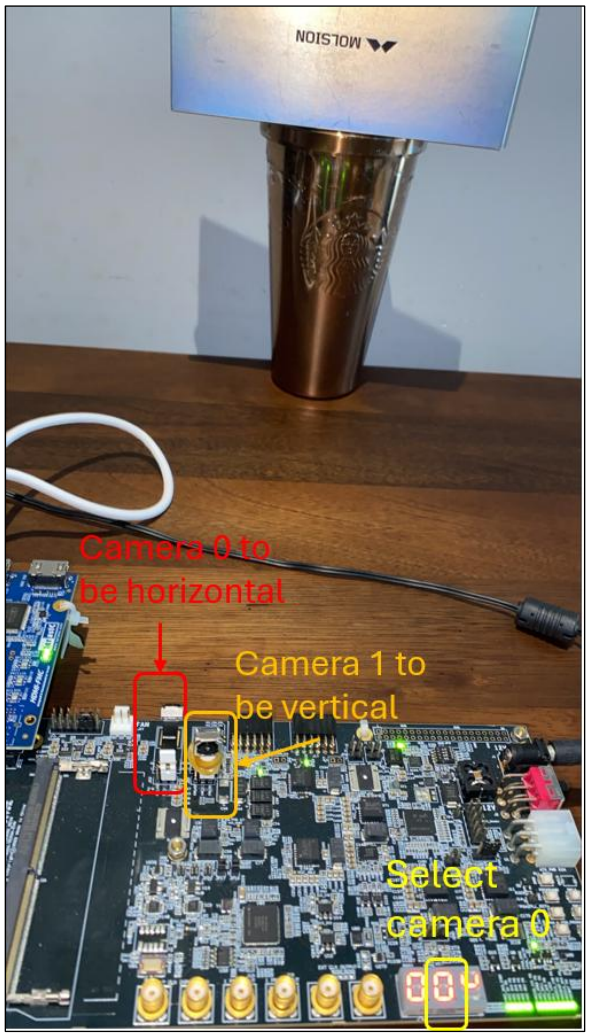


Figure 8.12. Camera 0 Successfully Captures Video

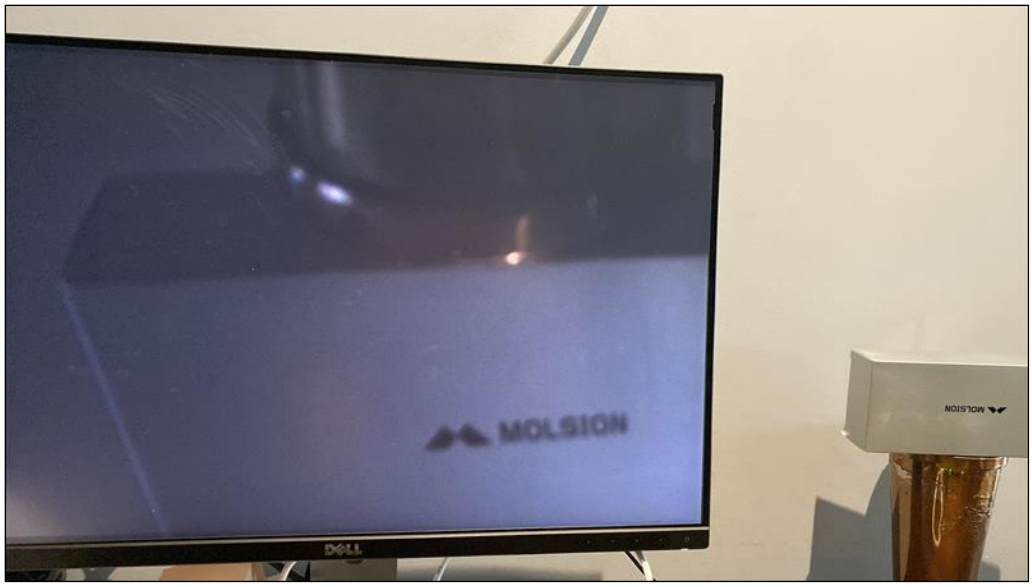


Figure 8.13. Video is Successfully Displayed on the Monitor

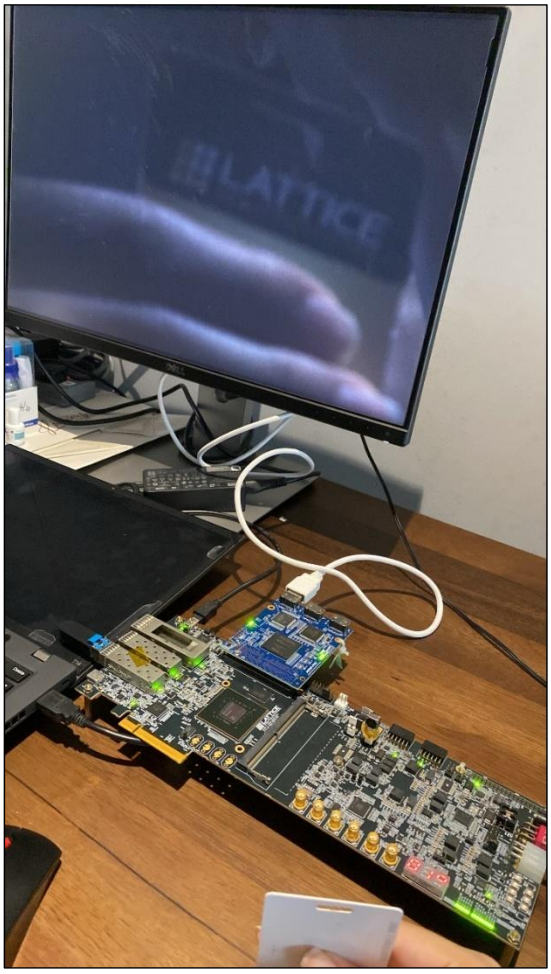


Figure 8.14. Monitor Shows Camera 1 Video

9. Resource Utilization

The following tables show the sample resource utilization of the MIPI CSI-2 to Ethernet reference design for the Lattice Avant Versa X70-3LFG1156C device using the Synplify Pro tool of the Lattice Radiant software 2025.1.0.39.0. The default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table 9.1. Reference Design Map Resource Utilization for 4K Resolution and 25G Data Rate

	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	DSP MULT	EBR
Camera Sensor 0 Subsystem	1,703	48	126	1,196	0	5
Camera Sensor 1 Subsystem	1,668	48	126	1,195	0	5
Video Transport Subsystem	2,898	894	290	3,318	0	11
Video Processing Subsystem	4,954	432	3,114	7,680	24	19
Display Subsystem	8,230	1,254	1,338	8,674	0	34
25G Ethernet Network Subsystem	9,289	0	264	6,462	0	4
Miscellaneous	501	0	300	802	0	1.5
Total	29,243	2,676	5,558	29,327	24	79.5

Note: Other submodules are categorized as miscellaneous in this table.

Table 9.2. Reference Design Map Resource Utilization for 4K Resolution and 10G Data Rate

	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	DSP MULT	EBR
Camera Sensor 0 Subsystem	1,688	48	126	1,195	0	5
Camera Sensor 1 Subsystem	1,693	48	126	1,196	0	5
Video Transport Subsystem	2,153	462	378	2,179	0	8
Video Processing Subsystem	4,954	432	3,114	7,680	24	19
Display Subsystem	8,231	1,254	1,338	8,671	0	34
10G Ethernet Network Subsystem	3,870	0	150	3,127	0	2
Miscellaneous	492	0	300	634	0	1.5
Total	23,081	2,244	5,532	24,682	24	74.5

Note: Other submodules are categorized as miscellaneous in this table.

Table 9.3. Reference Design Map Resource Utilization for 1080p Resolution and 25G Data Rate

	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	DSP MULT	EBR
Camera Sensor 0 Subsystem	1,701	48	124	1,194	0	5
Camera Sensor 1 Subsystem	1,708	48	124	1,197	0	5
Video Transport Subsystem	2,912	894	280	3,300	0	11
Video Processing Subsystem	4,960	432	3,114	7,672	24	19
Display Subsystem	8,206	1,242	1,322	8,624	0	33
25G Ethernet Network Subsystem	9,289	0	264	6,462	0	4

	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	DSP MULT	EBR
Miscellaneous	492	0	300	802	0	1.5
Total	29,268	2,664	5,528	2,9251	24	78.5

Note: Other submodules are categorized as miscellaneous in this table.

Table 9.4. Reference Design Map Resource Utilization for 1080p Resolution and 10G Data Rate

	LUT4 Logic	LUT4 Distributed RAM	LUT4 Ripple Logic	PFU Registers	DSP MULT	EBR
Camera Sensor 0 Subsystem	1,669	48	124	1,191	0	5
Camera Sensor 1 Subsystem	1,671	48	124	1,190	0	5
Video Transport Subsystem	2,148	462	370	2,161	0	8
Video Processing Subsystem	4,960	432	3,114	7,672	24	19
Display Subsystem	8,205	1,242	1,322	8,621	0	33
10G Ethernet Network Subsystem	3,870	0	150	3,127	0	2
Miscellaneous	490	0	300	634	0	1.5
Total	23,013	2,232	5,504	24,596	24	73.5

Note: Other submodules are categorized as miscellaneous in this table.

References

- MIPI CSI-2 to HDMI Reference Design (FPGA-UG-02206)
- 25G Ethernet MAC+PHY IP Core (FPGA-IPUG-02249)
- 10G Ethernet MAC+PHY IP Core (FPGA-IPUG-02245)
- DDR Memory Controller IP Core (FPGA-IPUG-02208)
- CSI-2/DSI D-PHY Receiver IP Core (FPGA-IPUG-02081)
- Byte-to-Pixel Converter IP Core (FPGA-IPUG-02079)
- Debayer IP core (FPGA-IPUG-02203)
- Automatic White Balance IP Core (FPGA-IPUG-02204)
- Color Correction Matrix IP Core (FPGA-IPUG-02214)
- Video Frame Buffer IP Core (FPGA-IPUG-02137)
- Avant-G/X Versa Board User Guide (FPGA-EB-02063)
- Avant-X web page
- Lattice Radiant FPGA design software
- Lattice Solutions Reference Designs web page
- Lattice Solutions IP Cores web page
- Lattice Propel Design Environment web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 1.1, August 2025

Section	Change Summary
Introduction	Updated Table 1.1. Summary of the Reference Design.
Directory Structure and Files	Updated Figure 2.1. Directory Structure.
Functional Description	• Updated the introductory section. • Updated the Frame Format section. • Added Figure 3.4. AVTP Frame in AXI-S Data Bus for 10G Ethernet (Odd Line Number). • Added Figure 3.5. AVTP Frame in AXI-S Data Bus for 10G Ethernet (Even Line Number). • Updated the Video to AVTP Payload section. • Updated the Store & Forward FIFO section. • Updated the AVTP Header Inserter section. • Updated the AVTP Header Remover section. • Updated the Video Line Buffer section. • Updated Table 3.5. Clock Domain Distribution.

Reference Design IP and Parameter Description	• Updated the IP Description section.
	• Updated the following figures:
	  • Figure 4.1. CSI-2 to HDMI IP GUI.
	  • Figure 4.2. Byte-to-Pixel IP GUI.
	  • Figure 4.4. Automatic White Balance IP GUI.
	  • Figure 4.6. System Clock PLL IP GUI.
	  • Figure 4.8. Image Sensor Memory File Location.
	  • Figure 4.10. HDMI Memory File Location.
	  • Figure 4.12. Memory Controller Avant IP GUI.
	• Added the following figures:
	  • Figure 4.14. Video Line Buffer DP RAM IP GUI for 10G Ethernet Data Rate.
	  • Figure 4.17. FIFO 256x76 IP GUI for 10G Ethernet Data Rate.
	• Updated the 10G and 25G Ethernet IP section.
	• Updated Table 4.10. Parameters in fpga_top.sv.

Running Reference Design	Updated the following sections: • Compiling the Reference Design section. • Generating the Bitstream File section.
Simulating Reference Design	Updated the simulation steps and Figure 7.1. Simulation Wizard: Create Simulation Project.
Implementing the Reference Design on Board	• Updated the Requirements section. • Updated Figure 8.1. Overview Hardware Setup. • Updated step 3 and Figure 8.7. Load Bitstream File in the Bit File Programming section.

Resource Utilization	• Updated the following tables:
	  • Table 9.1. Reference Design Map Resource Utilization for 4K Resolution and 25G Data Rate.
	  • Table 9.3. Reference Design Map Resource Utilization for 1080p Resolution and 25G Data Rate.
	• Added the following tables:
	  • Table 9.2. Reference Design Map Resource Utilization for 4K Resolution and 10G Data Rate.
	  • Table 9.4. Reference Design Map Resource Utilization for 1080p Resolution and 10G Data Rate.

Revision 1.0, March 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com