Pixel-to-Byte Converter IP

IP Version: v1.9.1

User Guide

FPGA-IPUG-02094-2.0

June 2025

Disclaimers

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

Inclusive Language

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

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
APB	Advanced Peripheral Bus
AXI	Advanced eXtensible Interface
CSI	Camera Serial Interface
DSI	Display Serial Interface
EBR	Embedded Block RAM
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
I/O	Input/Output
IP	Intellectual Property
LSB	Least Significant Bit
LSE	Lattice Synthesis Engine
LUT	Look-Up Table
LVDS	Low-voltage Differential Signaling
MIPI	Mobile Industry Processor Interface
RAM	Random Access Memory
RGB	Red Green Blue
RO	Read-Only
RTL	Register Transfer Level
RW	Read-Write
Rx	Receiver
Tx	Transmitter
W1C	Write 1 to Clear

1. Introduction

1.1. Overview of the IP

The Lattice Semiconductor Pixel-to-Byte Converter IP converts a standard parallel video interface to DSI or CSI-2 data. This IP is used together with the Lattice CrossLink™-NX, Certus™-NX, CertusPro™-NX, MachXO5™-NX, Lattice Avant™, and Certus-N2 FPGA device families.

The increasing demand for better displays makes bridging applications very popular. Mobile Industry Processor Interface (MIPI®) D-PHY has become the industry's primary high-speed PHY solution for camera and display interconnection in mobile devices. It is typically used in conjunction with MIPI Camera Serial Interface-2 (CSI-2) and MIPI Display Serial Interface (DSI) protocol specifications. It meets the requirements of low-power, low noise generation, and high noise immunity that mobile phone designs demand.

MIPI D-PHY is designed to replace traditional parallel bus based on LVCMOS or LVDS. However, many processors and displays/cameras still use an RGB or CMOS as interface. So, to connect to a MIPI D-PHY IP, a converter logic is required to convert the parallel interface into MIPI D-PHY byte packet compatible format.

This document describes the use of Lattice FPGA technology for applications requiring conversion of parallel interface to MIPI D-PHY byte packet compatible format. This design can be used in multiple configurations.

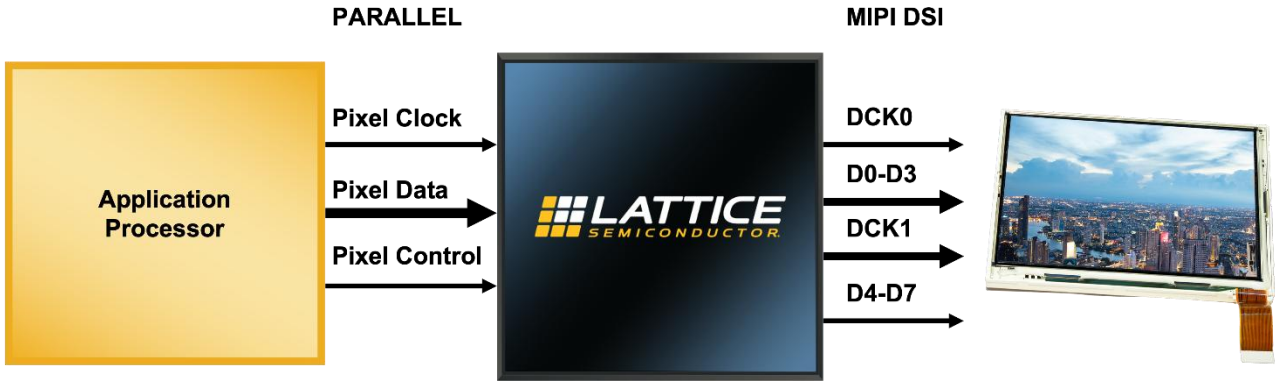

Figure 1.1. Sample Parallel Interface to MIPI DSI System Diagram

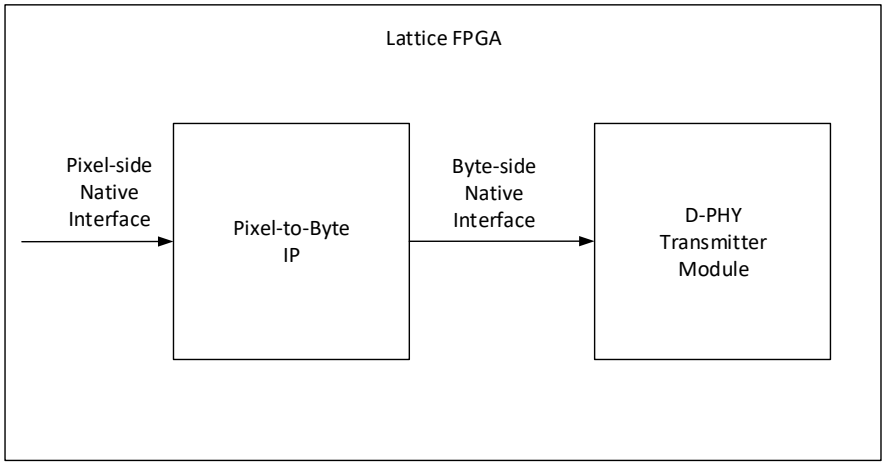

Figure 1.2. Pixel-to-Byte Converter IP General Diagram

1.2. Quick Facts

Table 1.1 presents a summary of the IP Core.

Table 1.1. Quick Facts

IP Requirements	Supported Devices	Lattice Avant, CrossLink-NX, Certus-NX, CertusPro-NX, MachXO5-NX, and Certus-N2
	IP Changes	For a list of changes to the IP, refer to the Pixel-to-Byte Converter IP Release Notes (FPGA-RN-02020).
Resource Utilization	Supported User Interface	Native Interface, AXI-4 Stream Interface
	Resources	Refer to Appendix A. Resource Utilization
Design Tool Support	Lattice Implementation	IP Core v1.9.1 – Lattice Radiant™ Software 2025.1
	Synthesis	Lattice Synthesis Engine (LSE), Synopsys Synplify Pro® for Lattice
	Simulation	Refer to the Lattice Radiant user guide for the list of supported simulators.

1.3. Features

1.3.1. Supported Features

The key features of the Pixel-to-Byte Converter IP include:

- Support for RGB888, RGB666, RGB444, RGB555, RGB565, RAW8, RAW10, RAW12, RAW14, RAW16, YUV420/YUV422 8/10-bit video formats
- Conversion of 1, 2, 4, 6, 8, or 10 pixels per pixel clock into MIPI D-PHY byte packet compatible format
- Support for byte arrangement for 1, 2, or 4 MIPI D-PHY data lanes
- Optional AXI4 Streaming interface for pixel and byte data
- APB Interface for configuration and status

1.3.2. Unsupported Features

The IP does not support configuration through registers.

1.4. Licensing and Ordering Information

An IP specific license string is required to enable full use of the Pixel-to-Byte Converter IP in a complete, top-level design.

The IP can be fully evaluated through functional simulation and implementation (synthesis, map, place and route) without an IP license string. This IP supports Lattice's IP hardware evaluation capabilities. You can create versions of the IP to operate in hardware for a limited time (approximately four hours) without requiring an IP license string. A license string is required to enable timing simulation and to generate a bitstream file that does not include the hardware evaluation timeout limitation.

For more information about pricing and availability of the Pixel-to-Byte Converter IP, contact your local Lattice Sales Office.

1.4.1. Ordering Part Number

Table 1.2. Ordering Part Number

Device Family	Part Number	
	Single Seat Annual	Single Seat Perpetual
CrossLink-NX	PIXEL-BYTE-CNX-US	PIXEL-BYTE-CNX-UT
Certus-NX	PIXEL-BYTE-CTNX-US	PIXEL-BYTE-CTNX-UT
CertusPro-NX	PIXEL-BYTE-CPNX-US	PIXEL-BYTE-CPNX-UT
MachXO5-NX	PIXEL-BYTE-XO5-US	PIXEL-BYTE-XO5-UT
Avant-E	PIXEL-BYTE-AVE-US	PIXEL-BYTE-AVE-UT
Avant-G	PIXEL-BYTE-AVG-US	PIXEL-BYTE-AVG-UT
Avant-X	PIXEL-BYTE-AVX-US	PIXEL-BYTE-AVX-UT
Certus-N2	PIXEL-BYTE-CN2-US	PIXEL-BYTE-CN2-UT
Bundled	MIPI-BNDL-US	MIPI-BNDL-UT

1.5. IP Validation Summary

The IP is validated with the following synthesis tool:

Table 1.3 shows the validation status for the Pixel-to-Byte Converter IP. The ✓ mark indicates whether the IP has been validated for Simulation, Timing, or with Hardware.

Table 1.3. IP Validation Level

Device Family Used	Performance Grade	IP Version	Software Version	Synthesis Tool	Validation Level		
					Simulation	Timing	Hardware
Lattice Avant	3	v1.9.1	Radiant 2025.1	Synplify Pro, May 2025	✓	✓	—
Certus Pro-NX	9	V1.9.1	Radiant 2025.1	Synplify Pro, May 2025	✓	✓	—

1.6. Minimum Device Requirements

Refer to Appendix A. Resource Utilization for the minimum required resources to instantiate this IP.

1.7. Naming Conventions

1.7.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL. This includes radix indications and logical operators.

1.7.2. Data Ordering and Data Types

The most significant bit within the pixel data is the highest index.

1.7.3. Signal Names

Signal Names that end with:

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

2. Functional Description

2.1. IP architecture Overview

The Pixel-to-Byte Converter IP converts a standard parallel video interface to either DSI or CSI-2 byte packets. The input interface for the design consists of a pixel clock and pixel bus. For DSI, the packets also include vertical and horizontal sync flags, and a data enable. For CSI-2, they consist of frame, line, and data valid flags.

Figure 2.1 shows the Pixel-to-Byte IP Functional Diagram which consists of configurable interfaces – Pixel-side Native, Byte-side Native, AXI4-Stream Receiver/Transmitter, APB Receiver, synchronizers, and Pixel-to-Byte Core Module. The pixel domain is responsible for the reception and processing of pixel data, while the byte domain processes and transmits byte data in the CSI-2/DSI standard. This byte data stream is compatible with different MIPI D-PHY applications.

Note: The dashed lines in the figure are optional components/signals and may not be available in the IP depending on the attribute selected.

In summary, the Pixel-to-Byte IP consists of the following blocks:

- Pixel-side Native Interface*
- Byte-side Native Interface*
- AXI4-Stream Receiver Interface*
- AXI4-Stream Transmitter Interface*
- APB Receiver Interface
- Pixel-to-Byte Core Module

***Note:** The Native Interface is disabled when its corresponding AXI4 Stream interface is enabled, and vice versa.

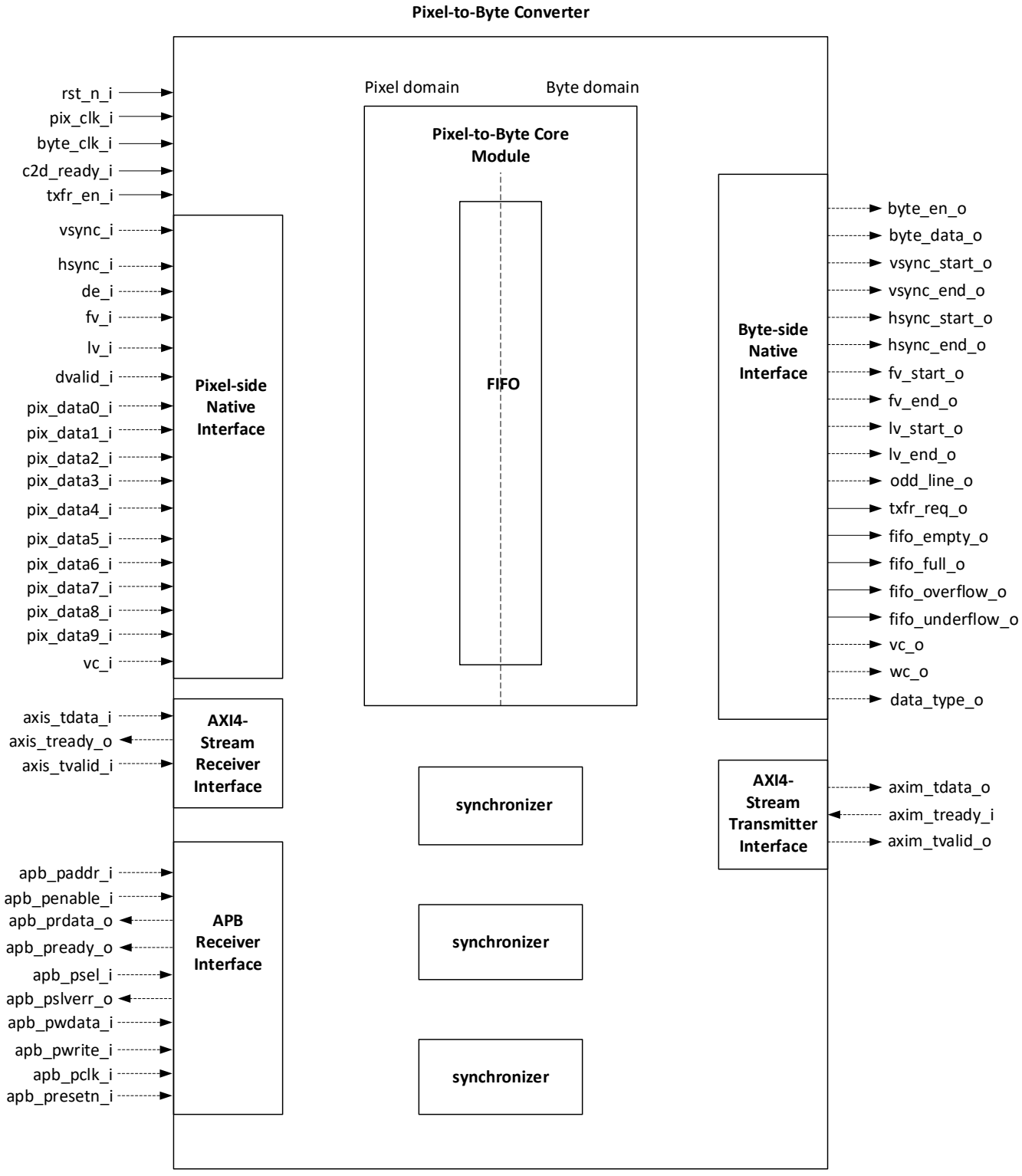

Figure 2.1. Pixel-to-Byte Converter IP Functional Diagram

2.1.1. Pixel-side Native Interface

The Pixel-side Native Interface receives pixel data stream from a parallel interface to be converted to CSI2/DSI format byte data.

2.1.2. Byte-side Native Interface

Byte-side Native Interface outputs the converted byte data stream in CSI2/DSI format. Figure 2.2 shows the ports of the Pixel-to-Byte IP if both Pixel-side and Byte-side Native interfaces are enabled. Ports are replaced accordingly depending on the chosen configuration.

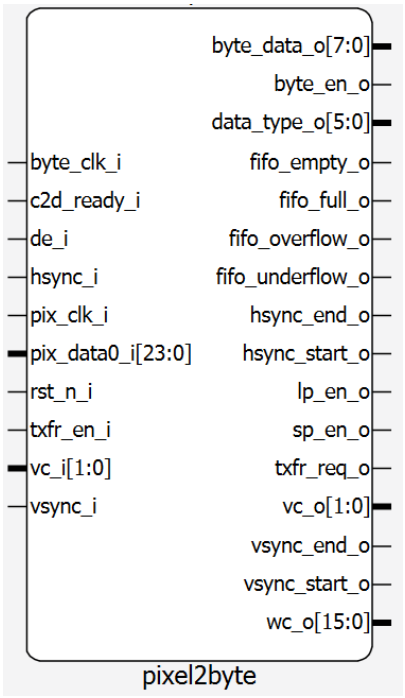

Figure 2.2. Ports of Pixel-to-Byte for Pixel/Byte Native Interface

Byte data output is dependent on the provided word count. The word count may be set to the minimum based on the data type chosen, or be manually provided. When inputting word count values not aligned with the number of byte clock cycles required to complete the whole byte conversion, you must check which byte data are considered valid. Refer to Figure 2.3 for checking of valid byte data in scenario with misaligned word count.

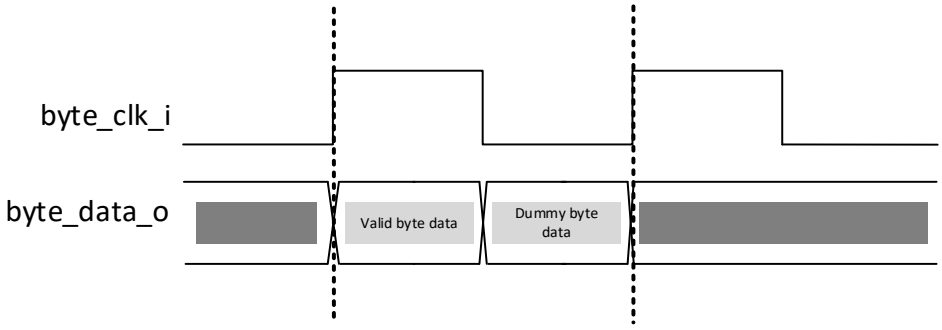

Figure 2.3. Checking of Valid Byte Data with Misaligned Word Count

2.1.3. AXI4 – Stream Receiver Interface

When the AXI4 - Stream Receiver Interface is enabled, the Pixel-side Native Interface's pixel data ports are disabled and are instead received through the AXI4 - Stream Receiver Interface. This interface implementation does not support backpressure. Also, axis_tready_o is always asserted, signifying that all incoming pixel data are considered valid and are processed by the IP. Only axis_tvalid_i assertion from you is required to signify valid pixel data to the IP.

Figure 2.4 shows the parsing of input pixels from AXI4-Stream Receiver Interface. Bus width of axis_tdata_i is set to AXI_PIX_WIDTH and its value is always a byte-multiple as defined by the AXI4-Stream protocol. Padding of zeroes is done if the product of *Number of Input Pixel Lanes* and pixel data bus width (PD_BUS_WIDTH) is less than the ceiled value of AXI_PIX_WIDTH. Pixels are mapped accordingly based on the configured *Number of Input Pixel Lanes* where each lane has a pixel data bus width value that depends on *Data Type* attribute. This information can be seen in the *Notes* of Table 4.1.

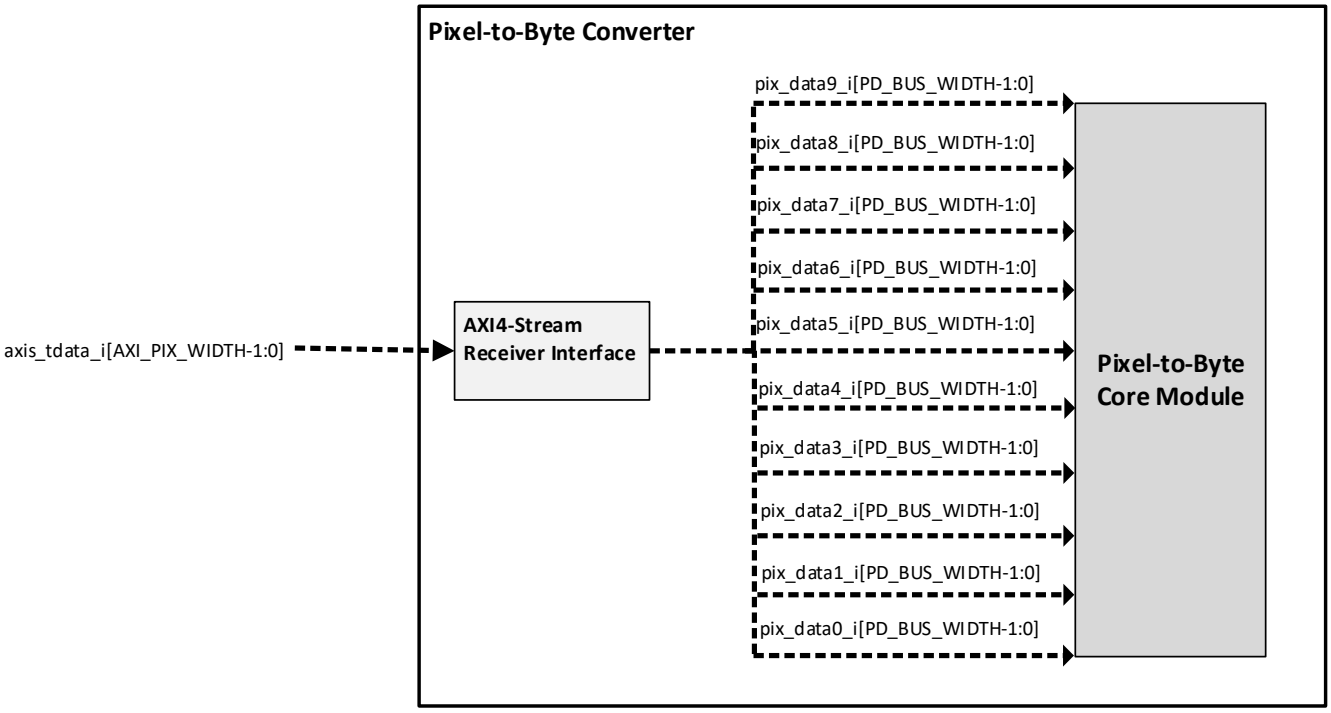

Figure 2.4. Pixel Data Mapping with AXI4-Stream Receiver Interface

2.1.4. AXI4 – Stream Transmitter Interface

When the AXI4 - Stream Transmitter Interface is enabled, converted byte data is transmitted through this interface instead of the Byte-side Native Interface. The axim_tvalid_o signal acts as a byte valid/enable and byte data is driven on axim_tdata_o. Additionally, axim_tvalid_o may be asserted to indicate changes in data type for sync signals without any corresponding valid byte data.

The axim_tdata_o consists of parsed data from byte_data_o, wc_o, and data_type_o. The size of the axim_tdata_o varies depending on the *Number of Tx Lanes* (NUM_TX_LANES) and *Tx Gear* (TX_GEAR) configured. Table 2.1 shows the details of mapping of axim_tdata_o. This interface does not support backpressure. It is expected to always have axim_tready_i asserted, meaning that byte data is always considered to be read once made available by the IP through axim_tvalid_o assertion.

Figure 2.5 shows the ports of Pixel-to-Byte IP when both AXI4-Stream Receiver and Transmitter Interface are enabled.

Table 2.1. Mapping of axim_tdata_o Port

AXI4-Stream Transmitter	Byte side Native Interface
axim_tdata_o [5: 0]	data_type_o
axim_tdata_o [21: 6]	wc_o
axim_tdata_o [DATA_WIDTH +22-1: 22]	byte_data_o
axim_tdata_o[DATA_WIDTH+24-1 :DATA_WIDTH+22]	2'b00

Note: DATA_WIDTH = NUM_TX_LANES × TX_GEAR

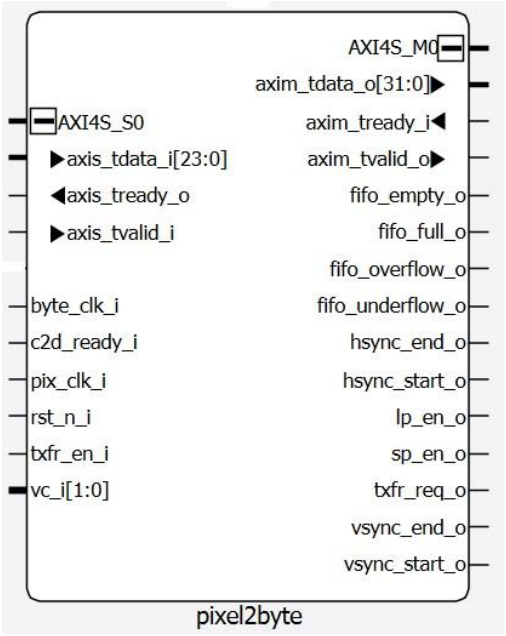

Figure 2.5. Ports of Pixel-to-Byte with AXI4-Stream Receiver and Transmitter Interfaces Enabled

2.1.5. APB Receiver Interface

When APB Receiver Interface is enabled, the IP implements an APB 3.0 receiver interface for register programming and status reading. Figure 2.6 shows the ports of Pixel-to-Byte IP when APB Receiver Interface enabled. It maintains REG0 and REG1 registers for configuration/status. Refer to *APB 3.0 specification* for protocol details. For more information regarding the registers, see the Register Description section.

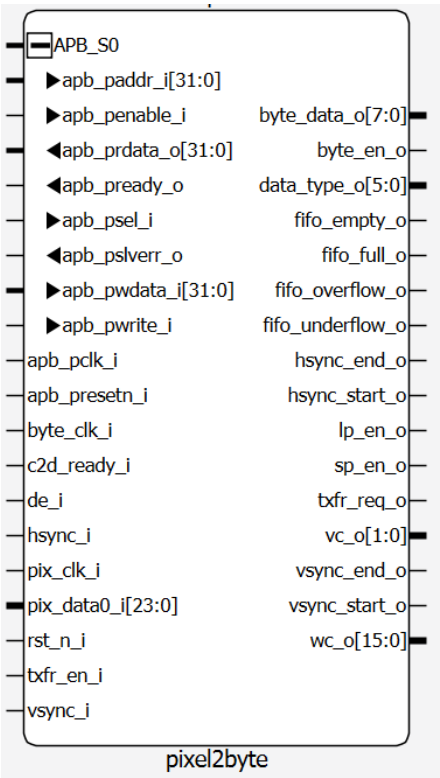

Figure 2.6. Ports of Pixel-to-Byte with APB Interface Enabled

2.1.6. Pixel-to-Byte Core Module

The top wrapper instantiates the different modules used in the IP. It instantiates the Pixel-to-Byte Core Module which does pixel-to-byte packet conversion. It also instantiates different wrapper modules for Pixel-to-Byte converter logic for each data type, including the FIFO and synchronizer module.

2.1.6.1. FIFO

FIFO is being implemented in this module to store pixel data in pixel clock and turn it to byte format in byte clock. Figure 2.7 shows the Pixel-to-Byte FIFO Diagram.

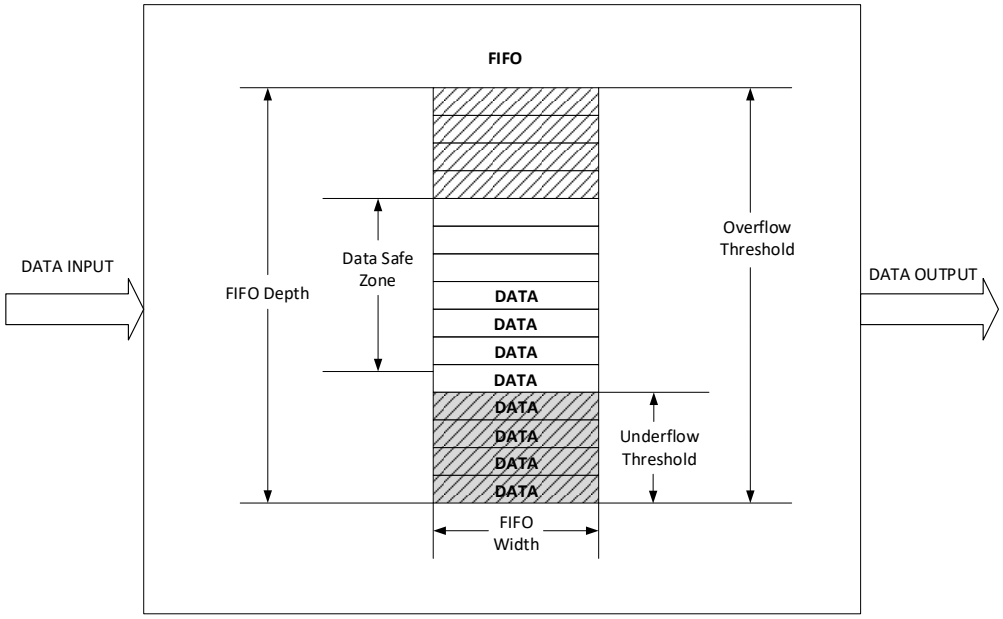

Figure 2.7. Pixel-to-Byte FIFO Diagram

The FIFO width is equal to the lowest multiple of the input pixel data width. The input pixel data width is greater than or equal to the input bus width. The FIFO width also determines the number of pixels that are grouped together and written to the same FIFO address.

FIFO depth is the maximum number of data that can be stored. To avoid FIFO full/overflow and empty/underflow states, define the Data Safe Zone by setting the Read Delay attribute. The Read Delay attribute is the minimum value that triggers the start of reading data from the FIFO. Read Delay is the same as FIFO Threshold.

2.1.6.2. FIFO Computations and Considerations

The three attributes in the FIFO are as follows:

- FIFO_width – indicates the size per write in the FIFO
- FIFO_depth – indicates the maximum capacity of the FIFO
- FIFO_threshold – triggers the start of data transfer in the byte side

The three attributes are automatically calculated in the FIFO unless Manual Adjust attribute box is checked.

The FIFO_width, FIFO_depth, and FIFO_threshold values have dependencies on the attributes listed below. For more information, refer to Table 3.1.

- *Data Type*
- *Number of Input Pixel Lanes*
- *Pixel Clock Frequency*
- *PD_BUS_WIDTH*
- *Number of Tx Lanes*
- *TX Gear*
- *Byte Clock Frequency*
- *Word Count*

Hence, FIFO_width, FIFO_depth, and FIFO_threshold values may vary depending on the other attributes listed above. Additional definitions that can help in FIFO computations is listed in Table 2.2.

Table 2.2. Additional FIFO Terminologies

Expression	Value
Rx_width	*PD_BUS_WIDTH* x *Number of Input Pixel Lanes*
Tx_width	*Number of Tx Lanes* x *TX Gear*
Receiver Data Rate	Rx_width x *Pixel Clock Frequency*
Transmitter Data Rate	Tx_width x *Byte Clock Frequency*
N_ratio	*Transmitter Data Rate / Receiver Data Rate*

Rx_width is the word width on the Pixel side and Tx_width is the word width on the byte side.

The FIFO width depends on the following values:

- *Data Type*
- Rx_width
- Tx_width

For RAW12, RAW14, and RAW16 data types, FIFO width also depends on the Byte Count Restriction. This is shown in Table 3.2.

Then, the FIFO_width can be calculated as follows:

```
    If DATA_TYPE = RAW12
       FIFO_Width = 3.0 * Rx_width
    Else if DATA_TYPE = RAW14
       FIFO_Width = 7.0 * Rx_width
    Else if DATA_TYPE = RAW 16,
       FIFO_Width = 2.0 * Rx_width
    Else
       FIFO_Width = Tx_width * ceil (Rx_width / Tx_width)

Where ceil function returns an integer value that is equal or greater than the calculated
value of Rx_width/Tx_width.
```

In addition, the data flow speed on the pixel side is the *Receiver Data Rate* and the data flow speed on the byte side is the *Transmitter Data Rate*. The FIFO Depth and FIFO Threshold depend on the ratio of data flow speed on the pixel side and on the byte side. This ratio is the N_ratio. *Receiver Data Rate* should be less than or equal to *the Transmitter Data Rate* to ensure no overflow/underflow states would occur.

To get FIFO_depth and FIFO_threshold, *Word Count* must be converted into a number of valid pixels per line. This value is called horizontal resolution (hres). Table 2.3 shows the different hres computation according to the *Data Type* selected.

Table 2.3. hres Computation Based on Data Type

Data Type	hres Equivalent
RAW8, YUV420-8bit, YUV422-8bit	*Word Count/Number of Input Pixel Lanes*
RAW10, YUV420-10bit, YUV422-10bit	(4/5)*(*Word Count/Number of Input Pixel Lanes)*
RAW12	(2/3)*(*Word Count/Number of Input Pixel Lanes)*
RAW14	(4/7)*(*Word Count/Number of Input Pixel Lanes)*
RAW16	(1/2)*(*Word Count/Number of Input Pixel Lanes)*
RGB444	(1/2)*(*Word Count/Number of Input Pixel Lanes)*
RGB555	(1/2)*(*Word Count/Number of Input Pixel Lanes)*
RGB565	(1/2)*(*Word Count/Number of Input Pixel Lanes)*
RGB888	(1/3)*(*Word Count/Number of Input Pixel Lanes)*
RGB666	(4/9)*(*Word Count/Number of Input Pixel Lanes)*

FIFO_threshold can be calculated as follows:

```
        FIFO_threshold = int(ceil(hres)-floor(hres/n_ratio))

Where floor function returns an integer value that is equal or lesser than the calculated
value of hres/n_ratio.
```

FIFO_depth can be calculated as follows:

```
    For YUV420-8bit and YUV420-10bit:
        FIFO_depth=max(2^clog2(2*FIFO_threshold+6), 512)
    For other data types:
        FIFO_depth=max(2^clog2(FIFO_threshold+6), 512)

Where clog2 function returns a ceil-ed integer value after logarithmic base 2 operation is
applied to the expression.
```

For more information regarding FIFO_depth, refer to the Design Considerations section.

2.1.6.3. Synchronizer

Synchronizer module is a two-level synchronizer used to synchronize the input data into a different clock domain. In the design, the synchronizer module is used to synchronize the system reset into pixel, byte, and APB clock domains before it is used in the system.

2.2. Clocking

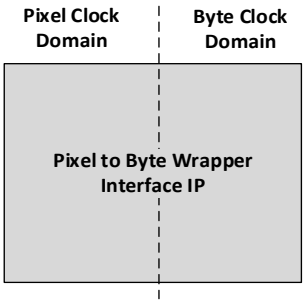

Figure 2.8. Clock Domain Crossing Block Diagram

Table 2.4. Clock Domain Crossing

Clock Domain Crossing	Handling Approach
Pixel Clock to Byte Clock	Parameterized Module Interfacing FIFO IP

The general guideline maintained in Pixel-to-Byte IP is for the N_ratio to be greater than or equal to 1. For more details regarding N_ratio, refer to FIFO Computations and Considerations.

The formula for N_ratio and example cases are shown below:

$$N\_ratio = \frac{Transmitter\ Data\ Rate}{Receiver\ Data\ Rate} = \frac{Number\ of\ Tx\ lanes\ \times\ Tx\ Gear\ \times\ Byte\ Clock\ Frequency \times Useful\ Rate}{Pixel\ Data\ Bus\ Width\ \times\ Number\ of\ Input\ Pixel\ Lanes\ \times\ Pixel\ Clock\ Frequency}$$

Where, *Useful Rate* depends on *Data Type* and is used for easier byte alignment.

```
If DATA_TYPE = RGB444
                  Useful_Rate = 12/16
Else if DATA_TYPE = RGB555
    Useful_Rate = 15/16
Else
    Useful_Rate = 1.0
```

Where, *Transmitter Data Rate* ≥ *Receiver Data Rate*.

Examples:

- IP configuration: *Data Type == RGB888; Number of Input Pixel Lanes == 1, Tx Gear == 8, Number of Tx Lanes == 4, Pixel Clock Frequency = 60 MHz, Byte Clock Frequency = 80 MHz*:

$$N\_ratio = \frac{Transmitter\ Data\ Rate}{Receiver\ Data\ Rate} = \frac{4 \times 8 \times 80}{24 \times 1 \times 60}$$

$$N\_ratio = 1.778$$

- IP configuration: *Data Type == RGB666; Number of Input Pixel Lanes == 1, Tx Gear == 8, Number of Tx Lanes == 2, Pixel Clock Frequency = 60 MHz, Byte Clock Frequency = 80 MHz*:

$$N\_ratio = \frac{Byte\ Clock}{Pixel\ Clock} = \frac{8 \times 2 \times 80}{18 \times 1 \times 60}$$

$$N\_ratio = 1.185$$

To maintain the N_ratio requirement, it is recommended to derive both the pixel and the byte clocks from one common source or use the pixel clock as the reference to the PLL generating the byte clock. Refer to Table 4.1 for more information on clock signal interface and requirement.

2.3. Reset

Table 2.5. Reset Signal

Reset Signal	Direction	Description
rst_n_i	Input	• Asynchronous active low system reset. • 0 - System on reset

rst_n_i connected to the Pixel-to-Byte Converter is an active low reset with a synchronous release is used in the design. Follow the initialization and reset sequence below:

1. Assert active low system reset for at least three cycles of the slower clock*.
2. Pixel-to-Byte Converter is ready to process data after reset.

***Note:** The slower clock can be either pix_clk_i or byte_clk_i.

2.4. User Interfaces

Table 2.6. User Interfaces and Supported Protocol

User Interface	Supported Protocols	Description
Pixel-side Native Interface	Camera Parallel Interface	CPI is one of the image sensor interfaces specified by the MIPI alliance. It connects multiple signals from parallel interface to video data bus from an image sensor.
Byte-side Native Interface	DSI (Display Serial Interface)	DSI protocol is commonly targeted at LCD and similar display technologies. It defines a serial bus communication protocol between the host (source of the image data), and the device which is the destination.
	CSI-2 (Camera Serial Interface 2)	MIPI CSI-2 is a widely adopted, high speed protocol for transmission of still and video images from image sensors to application processors. Communication protocol between the host (source of the image data), and the device which is the destination.

2.5. Timing Specifications

This section contains operational timing diagrams applicable to the Pixel-to-Byte IP.

Figure 2.9 and Figure 2.10 show the timing diagram of display and camera parallel input interface respectively. It follows the standard DSI/CSI-2 interface protocol with the following signals:

- VSYNC, HSYNC, Data Enable for DSI
- Frame Valid, Line Valid, and Data Valid for CSI-2
- Pixel data

All signals listed above are clocked by pixel clock. The number of pixels per pixel clock depends on the number of input pixel clocks selected during design configuration. Input pixel data is converted to byte data compatible with MIPI D-PHY packet with a bus width that is dependent on gearing and configured number of transmitter lanes. LSB of input pixel data is transmitted first. Byte arrangement is shown in Table 2.7.

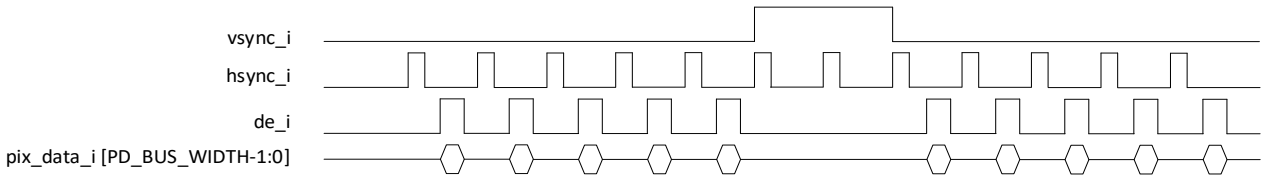

Figure 2.9. Display Parallel Input Interface Timing Diagram (DSI)

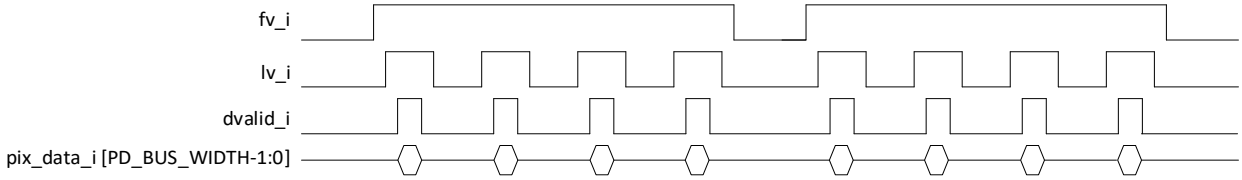

Figure 2.10. Camera Sensor Parallel Input Interface Timing Diagram (CSI-2)

Table 2.7. Output Byte Data Bus Width Allocation

byte_data_o	4-Lane		2-Lane		1-Lane	
	Gear 16	Gear 8	Gear 16	Gear 8	Gear 16	Gear 8
[7:0]	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0
[15:8]	Byte 1	Byte 1	Byte 1	Byte 1	Byte 1	
[23:16]	Byte 2	Byte 2	Byte 2			
[31:24]	Byte 3	Byte 3	Byte 3			
[39:32]	Byte 4					
[47:40]	Byte 5					
[55:48]	Byte 6					
[63:56]	Byte 7					

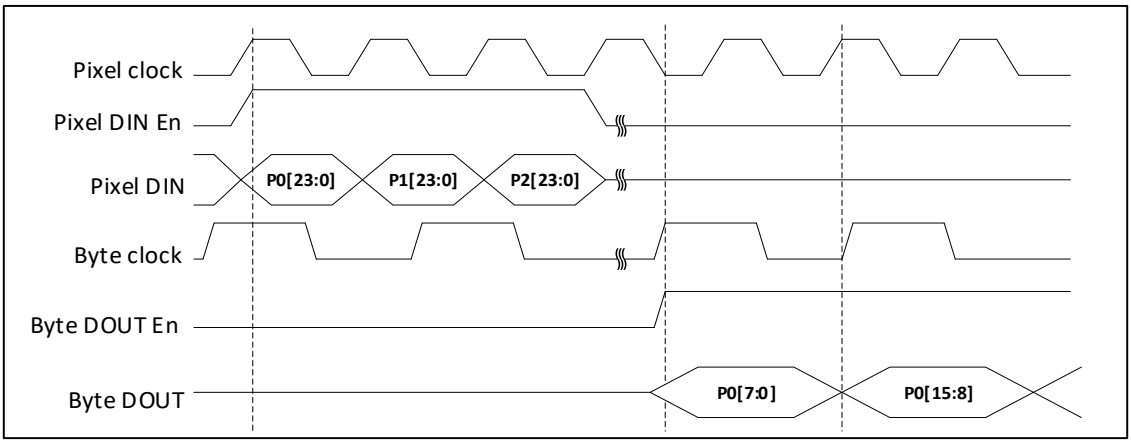

Figure 2.11. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 1 Pixel per Pixel Clock)

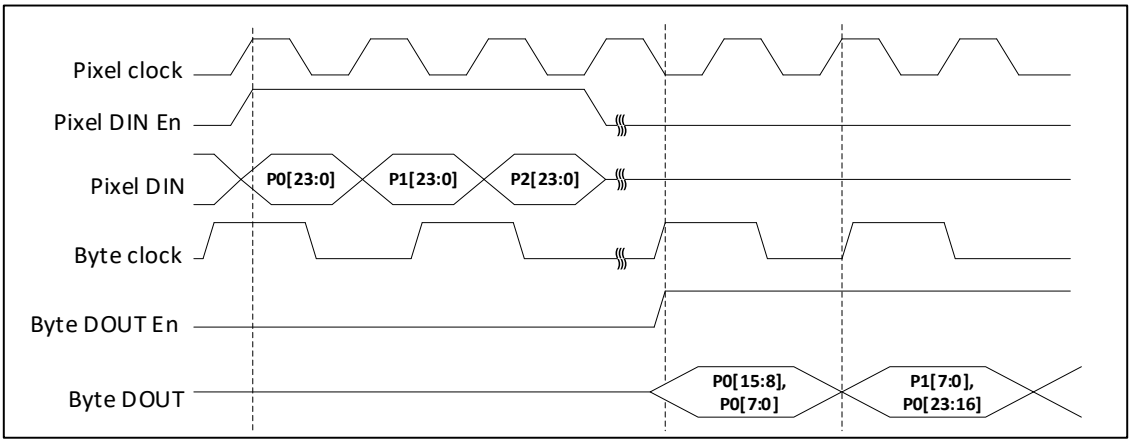

Figure 2.12. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 1 Pixel per Pixel Clock)

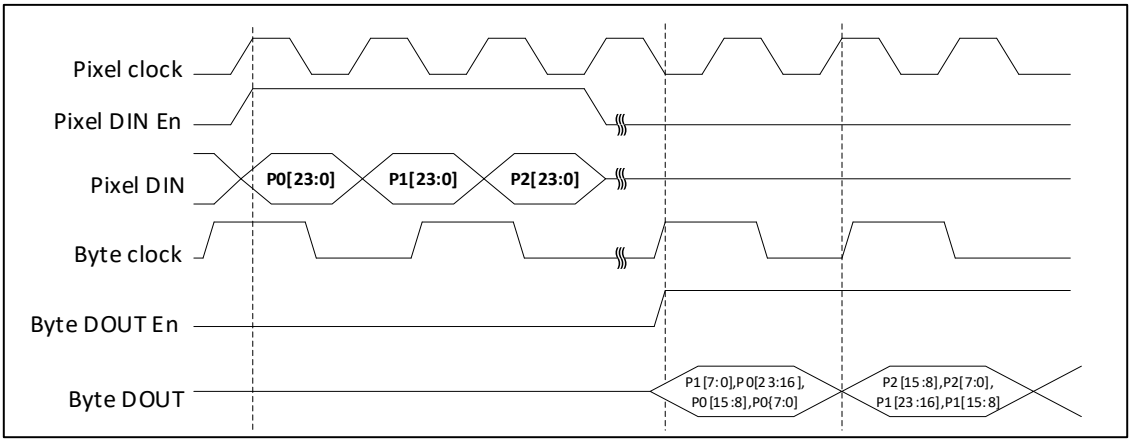

Figure 2.13. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 1 Pixel per Pixel Clock)

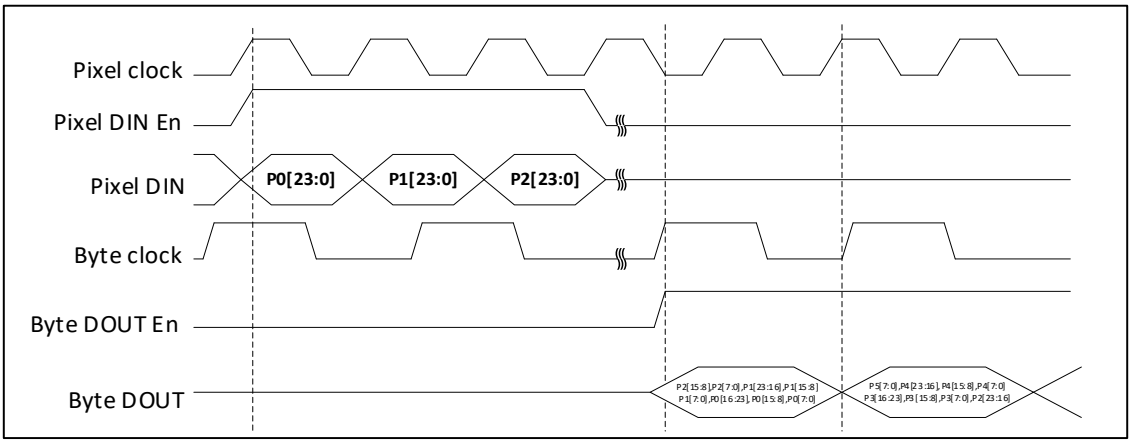

Figure 2.14. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 1 Pixel per Pixel Clock)

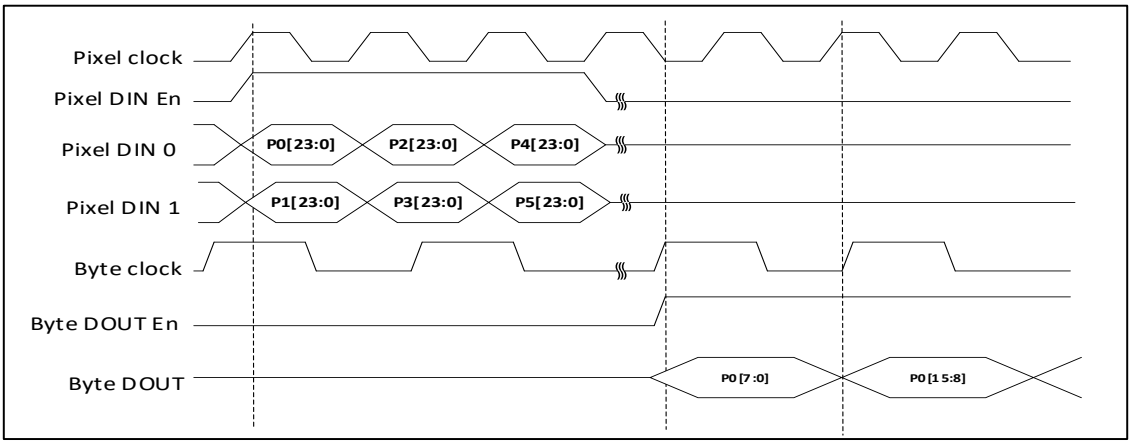

Figure 2.15. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 2 Pixels per Pixel Clock)

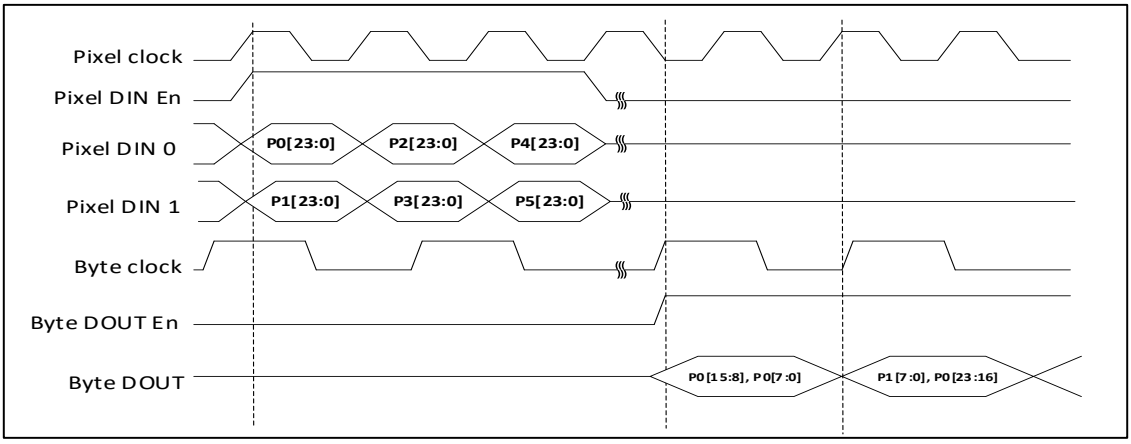

Figure 2.16. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 2 Pixels per Pixel Clock)

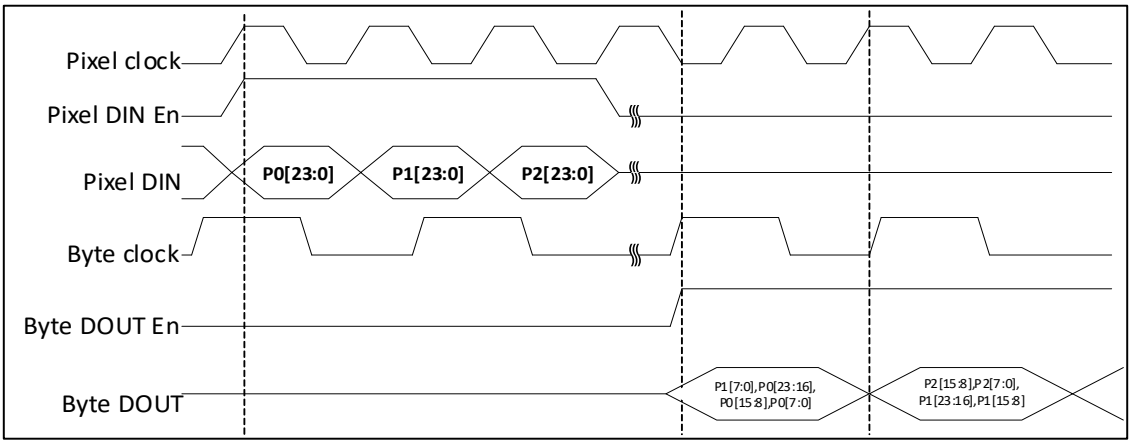

Figure 2.17. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 2 Pixels per Pixel Clock)

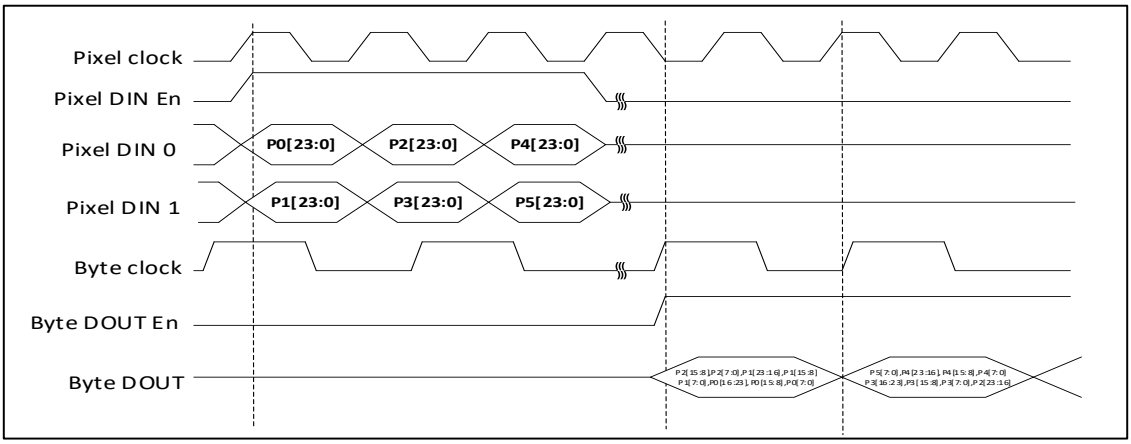

Figure 2.18. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 2 Pixels per Pixel Clock)

As Pixel-to-Byte Converter IP is interfaced to other MIPI D-PHY related IPs, handshaking is needed. Signals available in the Pixel-to-Byte Converter IP for handshaking are as follows:

- c2d_ready_i
- txfr_req_o
- txfr_en_i

Input signal txfr_en_i is expected to be asserted when MIPI D-PHY lanes are already in High Speed (HS) mode. The HS mode refers to the state after THS-ZERO until just before THS-TRAIL. Output signal txfr_req_o of the IP is asserted just before TLPX and when the c2d_ready_i signal is High. The input signal txfr_en_i should be asserted just right after THS-ZERO and until D-PHY lanes go to THS-TRAIL. For more details, refer to *Table 14 Global Operation Timing Parameters* of *MIPI Alliance Specification for D-PHY (v1.1)*.

Figure 2.19 shows handshake signals timing diagram. The handshaking sequence is as follows:

1. T0: c2d_ready_i is required to be asserted for handshaking to happen between Pixel-to-Byte and MIPI D-PHY.
2. T1: Once a DSI or CSI-2 sync signal or a valid pixel input is detected, txfr_req_o is asserted to request for HS mode to be enabled.
3. T2: Once clock cycle after request for HS mode is issued, c2d_ready_i is deasserted. This means that handshaking is acknowledged.
4. T3: After one clock cycle of handshaking is acknowledged, request for HS mode is cleared in which txfr_req_o is deasserted.
5. T4: The distance between assertion of txfr_req_o and txfr_en_i should be approximately the minimum requirement for TLPX + THS-PREPARE + THS-ZERO. Refer to *Table 14 Global Operation Timing Parameters* of *MIPI Alliance Specification for D-PHY, version 1.1*, for their corresponding values. For the Pixel-to-Byte IP, this is set to at least 195 ns.
6. T5: One clock cycle after assertion of txfr_en_i, sp_en_o or lp_en_o is asserted depending on the event detected by the handshaking. The sp_en_o is asserted for sync data handshake, while lp_en_o is asserted for payload data handshake.
7. T6-T7: After sp_en_o or lp_en_o is detected, txfr_en_i is deasserted. Its duration depends on whether short packet or long packet is identified. This is represented by wc_delay.
   - For sp_en_o, wc_delay is equal to 5 byte clock cycles
   - For lp_en_o, wc_delay is the time it takes for the whole byte conversion to be finished
8. T8: 160 ns + 10 unit intervals after the deassertion of txfr_en_i, c2d_ready_i is asserted to start the handshaking process again for both sync signals or payload data.

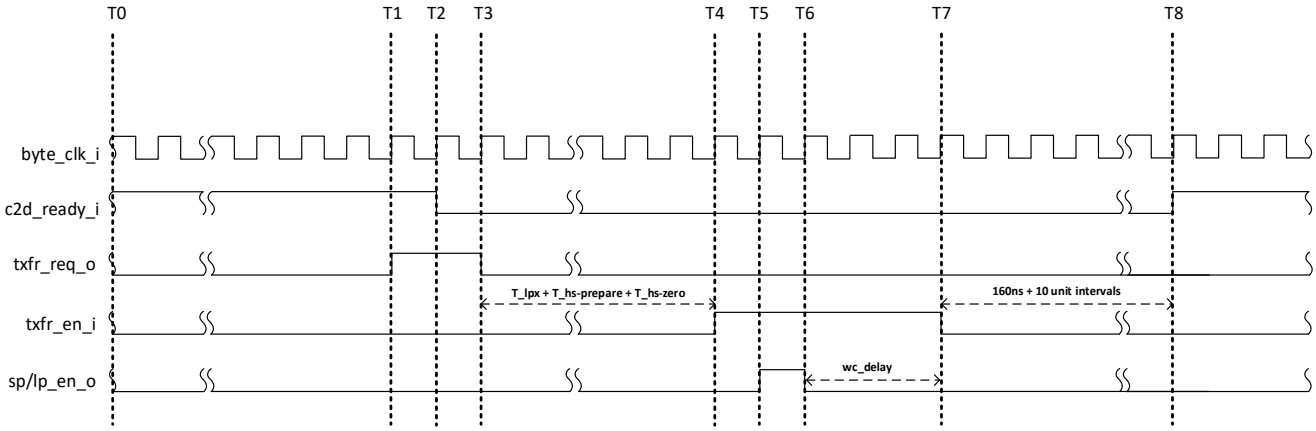

Figure 2.19. Handshake Signals Timing Diagram

Figure 2.20 and Figure 2.21 show the handshaking process for DSI and CSI-2 sync signals respectively. For concurrent events like positive and negative edges of hsync_i and vsync_i, this is considered as one sync event. Handshaking is enabled between T0 and T1 while c2d_ready_i is asserted. Once handshaking is finished, the process will start again for the next sync event at T8.

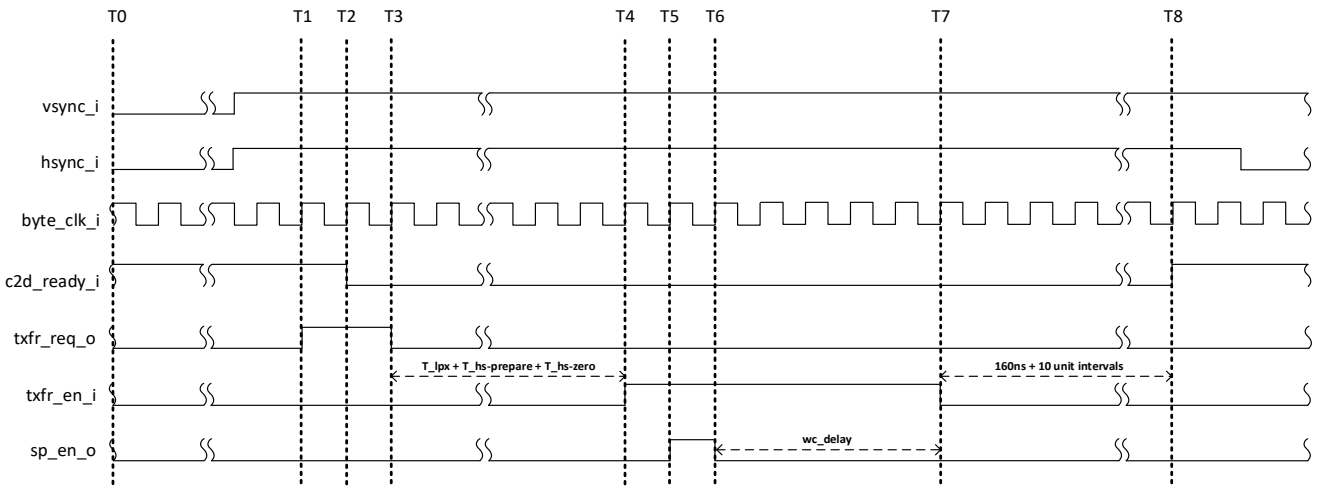

Figure 2.20. Handshake Timing Diagram for DSI Short Packet

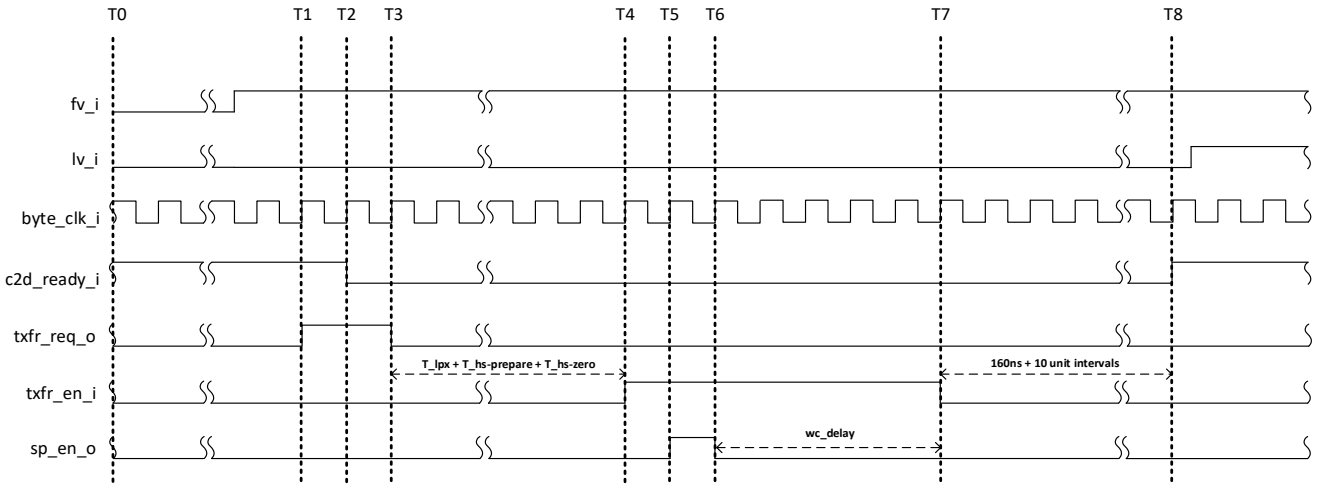

Figure 2.21. Handshake Timing Diagram for CSI-2 Short Packet

Figure 2.22 shows the handshaking process with respect to pixel payload. This is similar to the process in Figure 2.20 and Figure 2.21, but the difference is txfr_en_i is asserted for both lp_en_o assertion and enabled byte conversion from T4 to T7. New pixel transmission and handshaking for the next queued event will only take place at T8 to ensure that overlapping with byte conversion at T6 to T7 would not occur.

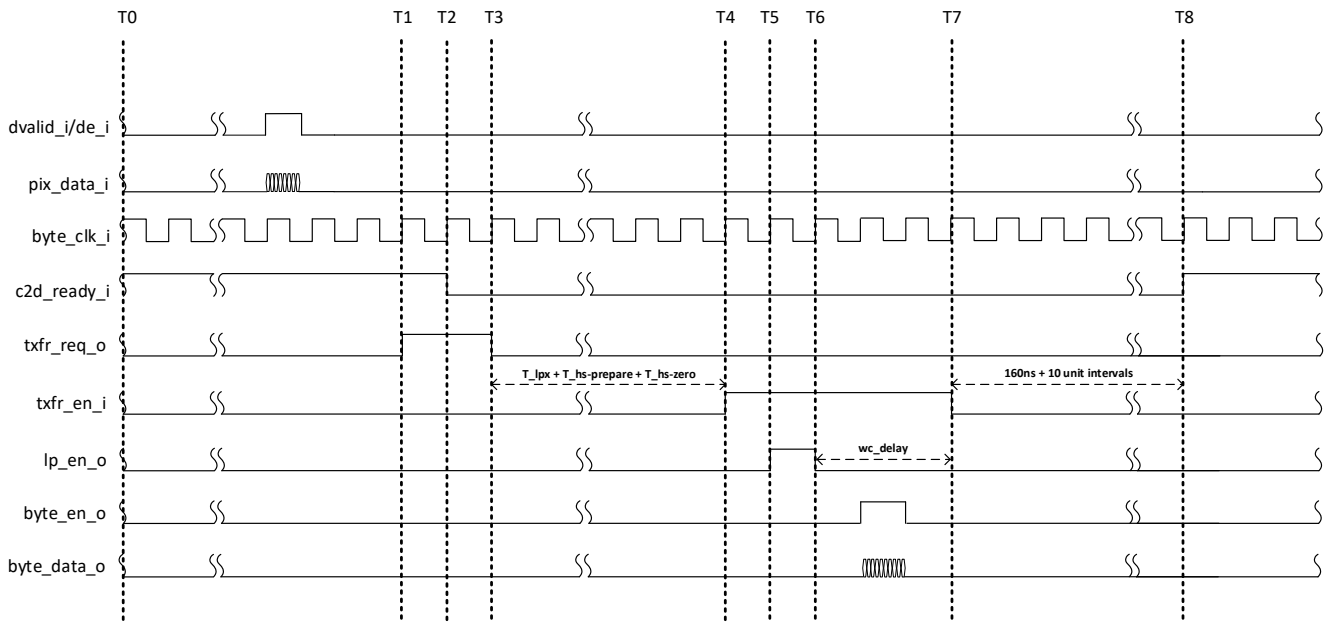

Figure 2.22. Handshake Timing Diagram for Long Packet

3. IP Parameter Description

This chapter provides information on how to generate and synthesize Pixel-to-Byte Converter IP Core using Lattice Radiant software and how to run simulation. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide and relevant Lattice tutorials.

The configurable attributes of the Pixel-to-Byte IP are shown in the following tables. You can configure the IP by setting the attributes accordingly in the IP Catalog's Module/IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in **bold**.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Dependency on Other Attributes	Description
General			
Data Type[1, 3]	RGB888, RGB666, RGB444, RGB555, RGB565, RAW8, **RAW10**, RAW12, RAW14, RAW16, YUV420_8, YUV420_10, YUV422_8, YUV422_10	Tx Interface	Specify the data type depending on the Data Format selected
Pixel Interface			
Number of Input Pixel Lanes[1, 3, 4]	**1**, 2, 4, 6, 8, 10	Tx Interface, Data Type, Number of Tx Lanes	Specify the number of input pixel lanes
Pixel Clock Frequency (MHz)[1, 3]	10-200, **60** (non-Avant) 10-250, **60** (Avant)	None	Specify the Pixel Clock Frequency to be used
Enable AXI4-Stream Receiver Interface	**OFF**, ON	None	Enable AXI4-Stream Receiver Interface
Receiver Data Rate (Mb/s)	Non-editable The default value is **600**	Data Type, Number of Input Pixel Lanes, Pixel Clock Frequency	Specify the data rate on the pixel domain
Byte Interface			
Tx Interface[5]	**CSI2**, DSI	None	Set the Tx interface
DSI Mode	**Non-Burst Pulses**, Non-Burst Events	Tx Interface	Specify the mode for DSI
Number of Tx Lanes[2, 3]	**1**, 2, 4	None	Set the target number of MIPI D-PHY Tx lanes
Tx Gear[2, 3]	**8**, 16	Tx Interface, Data Type, Number of Tx Lanes, Number of Tx Lanes, number of Input Pixel Per Clock	Specify the target Tx gearing
Byte Clock Frequency (MHz)[2, 3]	10-200, **80** (non-Avant) 10-250, **80** (Avant)	None	Specify the Byte Clock Frequency
Enable AXI4-Stream Transmitter Interface	**OFF**, ON	None	Enable AXI4-Stream Transmitter Interface
Transmitter Data Rate (Mb/s)	Non-editable The default value is **640**	Number of Tx Lanes, Tx Gear, Byte Clock Frequency	Specify the data rate on the pixel domain
Miscellaneous			
Enable APB Interface	**OFF**, ON	None	Enable APB Receiver Interface
Enable Handshake Signals	**Checked**, unchecked	Greyed out, always checked	—
Enable Line Valid Mask Signals	**OFF**, ON	Tx Interface	Enable Line Valid Mask Signals

Attribute	Selectable Values	Dependency on Other Attributes	Description
FIFO			
Word Count	3 – 65535 The default value is equal to the **MIN_WC value**	Data Type	MIN_WC is a computed value, which depends on the Data Type. Refer to Table 3.2. Byte Count Restriction for the actual value being used.
Manual Adjust	**Unchecked**, Checked	None	
Read Delay [1 – 65535]	1 – 65535 The default value is computed	Manual Adjust	Grayed out unless Manual Adjust box is checked
FIFO Depth [8-65536]	8 – 65536, **512**	Manual Adjust	Grayed out unless Manual Adjust box is checked

Notes:

1. Attribute is included in computation of *Transmitter Data Rate*. See the FIFO section for the computation.
2. Attribute is included in computation of *Receiver Data Rate*. See the FIFO section for the computation.
3. *Transmitter Data Rate* must be greater than or equal to *Receiver Data Rate* to avoid FIFO overflow/underflow.
4. Pixel data bus width of YUV data types refers to bits per YUV component instead of bits per YUV pixel.
   - For example, in YUV420-8bit, selecting *Number of Input Pixel Lanes == 2* means two parallel 8-bit component is received per pix_clk_i cycle.
5. For RGB data types (For CSI-2 and DSI formats), LSB is always transmitted first and considered as byte 0 in the byte conversion:
   - For CSI-2 format, the color arrangement must be in RGB order with B as the least significant byte.
   - For DSI format, the color arrangement must be in BGR order with R as the least significant byte.

Table 3.2. Byte Count Restriction

Data Type	Byte Count Restriction
RGB888	multiple of 3
RGB666	multiple of 9
RGB444	multiple of 2
RGB555	multiple of 2
RGB565	multiple of 2
RAW8	multiple of 1
RAW10	multiple of 5
RAW12	multiple of 3
RAW14	multiple of 7
RAW16	multiple of 2
YUV420_8	multiple of 4
YUV420_10	multiple of 10
YUV422_8	multiple of 4
YUV422_10	multiple of 5

3.2. Supported Configurations

Table 3.3. Supported Configurations for DSI

Number of Input Pixel Lanes	D-PHY Tx Lanes	Tx Gearing	Data Type
1	1	8	RGB666, RGB888
		16	RGB666, RGB888
	2	8	RGB666, RGB888
		16	RGB666, RGB888
	4	8	RGB666, RGB888
		16	RGB666, RGB888

Number of Input Pixel Lanes	D-PHY Tx Lanes	Tx Gearing	Data Type
2	1	8	RGB666, RGB888
		16	RGB666, RGB888
	2	8	RGB666, RGB888
		16	RGB666, RGB888
	4	8	RGB666, RGB888
		16	RGB666, RGB888
4	1	8	RGB666, RGB888
		16	RGB666, RGB888
	2	8	RGB666, RGB888
		16	RGB666, RGB888
	4	8	RGB666, RGB888
		16	RGB666, RGB888

Table 3.4. Supported Configurations for CSI-2

Number of Input Pixel Lanes	D-PHY Tx Lanes	Tx Gearing	Data Type
1	1	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
	2	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
	4	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
2	1	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565

Number of Input Pixel Lanes	D-PHY Tx Lanes	Tx Gearing	Data Type
	2	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
	4	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
4	1	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
	2	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
	4	8	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
		16	RAW10, YUV420 10-bit, YUV422 10-bit, RAW12, RAW14, RAW16, RAW8, YUV420 8-bit, YUV422 8-bit, RGB888, RGB444, RGB555, RGB565
6	4	8	RAW10, RAW12
8	4	16	RAW10, RAW12
10	4	16	RAW10, RAW12

4. Signal Description

Table 4.1. Pixel-to-Byte IP Signal Description

Port name	Type	Width	Clock Domain	Default Values	Description
Clocks and Resets					
rst_n_i	Input	1	—	—	Asynchronous active low system reset
byte_clk_i[1]	Input	1	byte_clk_i	80 MHz	Source clock for Byte Domain Interface
pix_clk_i[1]	Input	1	pix_clk_i	60 MHz	Source clock for Pixel Domain Interface
apb_presetn_i[10]	Input	1	—	—	Asynchronous active low system reset for APB Interface
apb_pclk_i[1, 10]	Input	1	apb_pclk_i	—	Source clock for APB Interface
Pixel Domain Interface					
pix_data0_i	Input	PD_BUS_WIDTH[2]	pix_clk_i	—	Input pixel data 0
pix_data1_i					Input pixel data 1
pix_data2_i					Input pixel data 2
pix_data3_i					Input pixel data 3
pix_data4_i[3]					Input pixel data 4–9
pix_data5_i[3]					
pix_data6_i[3]					
pix_data7_i[3]					
pix_data8_i[3]					
pix_data9_i[3]					
de_i[4]	Input	1	pix_clk_i	—	Input data enable for parallel interface
hsync_i[4]	Input	1	pix_clk_i	—	Input horizontal sync for parallel interface
vsync_i[4]	Input	1	pix_clk_i	—	Input vertical sync for parallel interface
dvalid_i[5]	Input	1	pix_clk_i	—	Input data enable for parallel interface
fv_i[5]	Input	1	pix_clk_i	—	Input frame valid for parallel interface
lv_i[5]	Input	1	pix_clk_i	—	Input line valid sync for parallel interface
Byte Domain Interface					
vsync_start_o[4]	Output	1	byte_clk_i	0	Pulse signal used to indicate Vsync start
vsync_end_o[4, 6]	Output	1	byte_clk_i	0	Pulse signal used to indicate Vsync end
hsync_start_o[4]	Output	1	byte_clk_i	0	Pulse signal used to indicate Hsync start
hsync_end_o[4, 6]	Output	1	byte_clk_i	0	Pulse signal used to indicate Hsync end
fv_start_o[5]	Output	1	byte_clk_i	0	Pulse signal used to indicate frame start
fv_end_o[5]	Output	1	byte_clk_i	0	Pulse signal used to indicate frame end
lv_start_o[5, 6]	Output	1	byte_clk_i	0	Pulse signal used to indicate line start
lv_end_o[5, 6]	Output	1	byte_clk_i	0	Pulse signal used to indicate line end
byte_en_o[14]	Output	1	byte_clk_i	0	Indicates valid output byte data
byte_data_o[14]	Output	NUM_TX_LANE × TX_GEAR[7]	byte_clk_i	0	Valid output byte data
sp_en_o	Output	1	byte_clk_i	0	Pulse signal used to indicate the start of short packet signal transmission
lp_en_o	Output	1	byte_clk_i	0	Pulse signal used to indicate the start of pixel data long packet transmission
AXI4-Stream Receiver Interface					
axis_tvalid_i[8]	Input	1	pix_clk_i	—	AXI4-Stream Receiver data valid
axis_tdata_i[8]	Input	AXI_PIX_WIDTH[15]	pix_clk_i	—	AXI4-Stream Receiver data mapped from pixel data input from the Native Pixel Interface

Port name	Type	Width	Clock Domain	Default Values	Description
axis_tready_o[8]	Output	1	pix_clk_i	—	AXI4-Stream Receiver ready. This is always asserted for the pixel data coming in. No backpressure is supported.
AXI4-Stream Transmitter Interface					
axim_tdata_o[9, 11]	Output	NUM_TX_LANE × TX_GEAR[7]	axim_aclk_i	0	AXI4-Stream transmitter data. It is mapped from Native Byte Interface.
axim_tready_i[9]	Input	1	axim_aclk_i	1	AXI4-Stream Transmitter ready signal. It is always set to 1 to read byte data once available. No backpressure is supported.
axim_tvalid_o[9]	Output	1	axim_aclk_i	0	AXI4-Stream Transmitter data valid. A data is considered valid when the FIFO is not empty.
APB Interface					
apb_prdata_o[10]	Output	32	apb_pclk_i	0	APB Receiver read data
apb_pready_o[10]	Output	1	apb_pclk_i	0	APB Receiver ready signal
apb_pslverr_o[10]	Output	1	apb_pclk_i	0	APB Receiver completer error
apb_paddr_i[10]	Input	32	apb_pclk_i	—	APB Receiver address
apb_pwdata_i[10]	Input	32	apb_pclk_i	—	APB Receiver write data
apb_penable_i[10]	Input	1	apb_pclk_i	—	APB Receiver enable
apb_psel_i[10]	Input	1	apb_pclk_i	—	APB Receiver completer select
apb_pwrite_i[10]	Input	1	apb_pclk_i	—	APB Receiver completer write
Miscellaneous Interface					
c2d_ready_i	Input	1	byte_clk_i	—	Ready signal for transmit request
odd_line_o[12]	Output	1	byte_clk_i	0	Indicates if current line is odd or even
data_type_o[14, 16]	Output	6	byte_clk_i	0	Indicates the data type based on MIPI DSI and CSI-2 Specifications
txfr_en_i	Input	1	byte_clk_i	—	Enable flag from outside the IP to indicate that byte data is ready to be sent out from Pixel-to-Byte IP
txfr_req_o	Output	1	byte_clk_i	0	When asserted, it indicates that valid data is received and IP is ready to process the data
vc_i[13]	Input	2	pix_clk_i	—	Virtual Channel input
fifo_empty_o	Output	1	byte_clk_i	0	Indicates that the FIFO is empty
fifo_full_o	Output	1	pix_clk_i	0	Indicates that the FIFO is full
fifo_overflow_o	Output	1	pix_clk_i	0	Indicates that the FIFO overflows
fifo_underflow_o	Output	1	byte_clk_i	0	Indicates that the FIFO underflows
vc_o	Output	2	byte_clk_i	0	Virtual Channel output
wc_o[14]	Output	16	byte_clk_i	WORD COUNT	Word count is a user-input value from GUI

Notes:

1. The recommended clock frequency is:
   - For Avant Devices: 10 MHz – 250 MHz
   - For non-Avant devices: 10 MHz – 200 MHz
2. PD_BUS_WIDTH refers to the Pixel Data Bus Width and the value depends on selected *Data Type*:
   - 24-bit bus width – RGB888
   - 18-bit bus width – RGB666
   - 16-bit bus width – RAW16, RGB565
   - 15-bit bus width – RGB555
   - 14-bit bus width – RAW14
   - 12-bit bus width – RAW12, RGB444
   - 10-bit bus width – RAW10, YUV420/422 10-bit

- 8-bit bus width – RAW8, YUV420/422 8-bit

3. Available only if the *Data Type == RAW10,RAW12* and *Number of Tx Lanes == 4*.
4. Available only if *Tx Interface == DSI*.
5. Available only if *Tx Interface == CSI-2*.
6. This signal is set to low when *Enable Line Valid Mask Signals == ON*.
7. NUM_TX_LANE refers to the *Number of Tx Lanes* in Byte Domain, and TX_GEAR refers to the *Tx Gear* in Byte Domain.
8. Available only if AXI4-Stream Receiver Interface is enabled.
9. Available only if AXI4-Stream Transmitter Interface is enabled.
10. Available only if APB Receiver Interface is enabled.
11. axim_tdata_o mapping:
    - axim_tdata_o [5: 0] – data_type_o
    - axim_tdata_o [21: 6] – wc_o
    - axim_tdata_o [NUM_TX_LANE*TX_GEAR+22 -1 : 22] – byte_data_o
12. Available for YUV420 data types:
    - 0 – Output value for even line
    - 1 – Output value for odd line
13. Available only if Native Pixel Interface is enabled.
14. Available for Native Byte and APB Interfaces.
15. AXI_PIX_WIDTH refers to the bus width of axis_tdata_i and its value is the product of *Number of Input Pixel Lanes* and *Pixel Data Bus Width* that is ceiled up to the next multiple of 8 value.
16. For this IP, the data_type_o has a reset/error value of 6'h3F. The rest of data_type_o values follow the MIPI DSI/CSI-2 Specifications.

5. Register Description

Registers listed are accessible through the APB Completer Interface.

The following access type nomenclatures are used throughout the document:

- RW: Read-write
- RO: Read-only
- W1C: Write 1 to clear

5.1. Register Address Map

Table 5.1. Register Address Map

Address Offset	Name	Description	Access	Default
0x0000	REG0	FIFO Status Register	W1C	32'h0
0x0004	REG1	Virtual Channel/Word Count Status Register	RW/RO	32'h0/*WORD COUNT*

5.2. REG0 Register

Table 5.2. REG0 Register

Field	Name	Description	Access	Default
[31:2]	rsvd	Reserved	RO	'h0
[1]	TX-FIFO Underflow	FIFO underflow indicator, from fifo_underflow_o port. Flag from this register indicates that read operation is done even when FIFO is empty.	W1C	1'b0
[0]	TX-FIFO Overflow	FIFO overflow indicator, from fifo_overflow_o port. Flag from this register indicates that write operation is done even when FIFO is full.	W1C	1'b0

5.3. REG1 Register

Table 5.3. REG1 Register

Field	Name	Description	Access	Default
[31:18]	rsvd	Reserved	RO	'h0
[17:2]	WC	Word Count. Stored value is equal to user input from GUI.	RO	*WORD COUNT*
[1:0]	VC	Virtual Channel	RW	1'b0

6. Designing with the IP

This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

6.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device's architecture. The steps below describe how to generate the Pixel-to-Byte Converter IP in the Lattice Radiant software.

To generate the Pixel-to-Byte Converter IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **Pixel-to-Byte Converter** under **IP, Audio_Video_and_Image_Processing** category. The **Module/IP Block Wizard** opens as shown in Figure 6.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

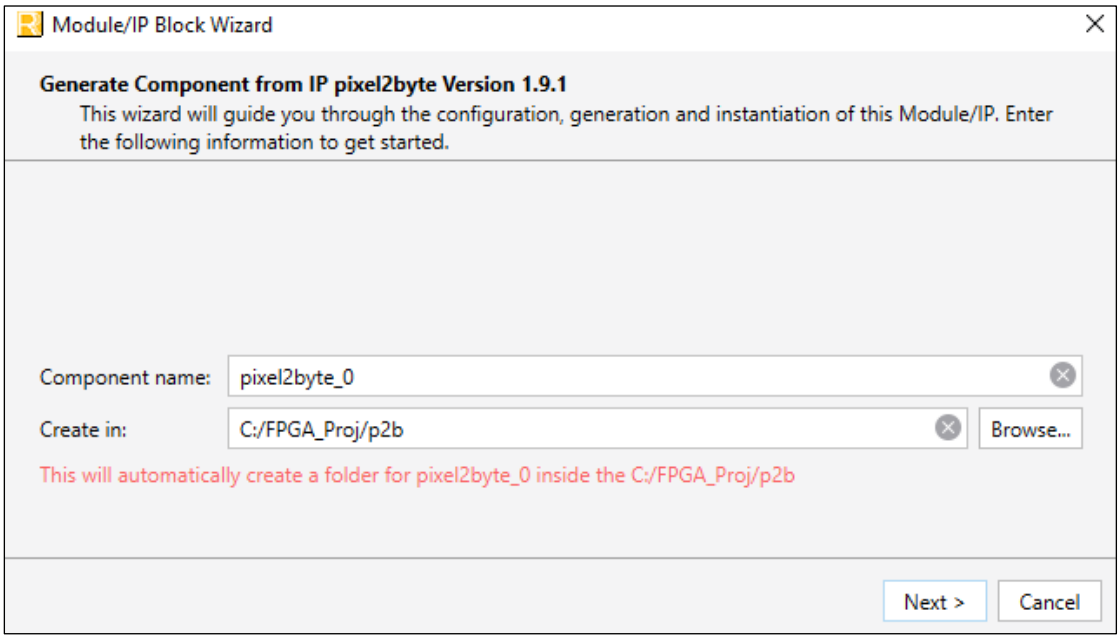

Figure 6.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected Pixel-to-Byte Converter IP using drop-down lists and check boxes. Figure 6.2 shows an example configuration of the Pixel-to-Byte Converter IP. For details on the configuration options, refer to the IP Parameter Description section.

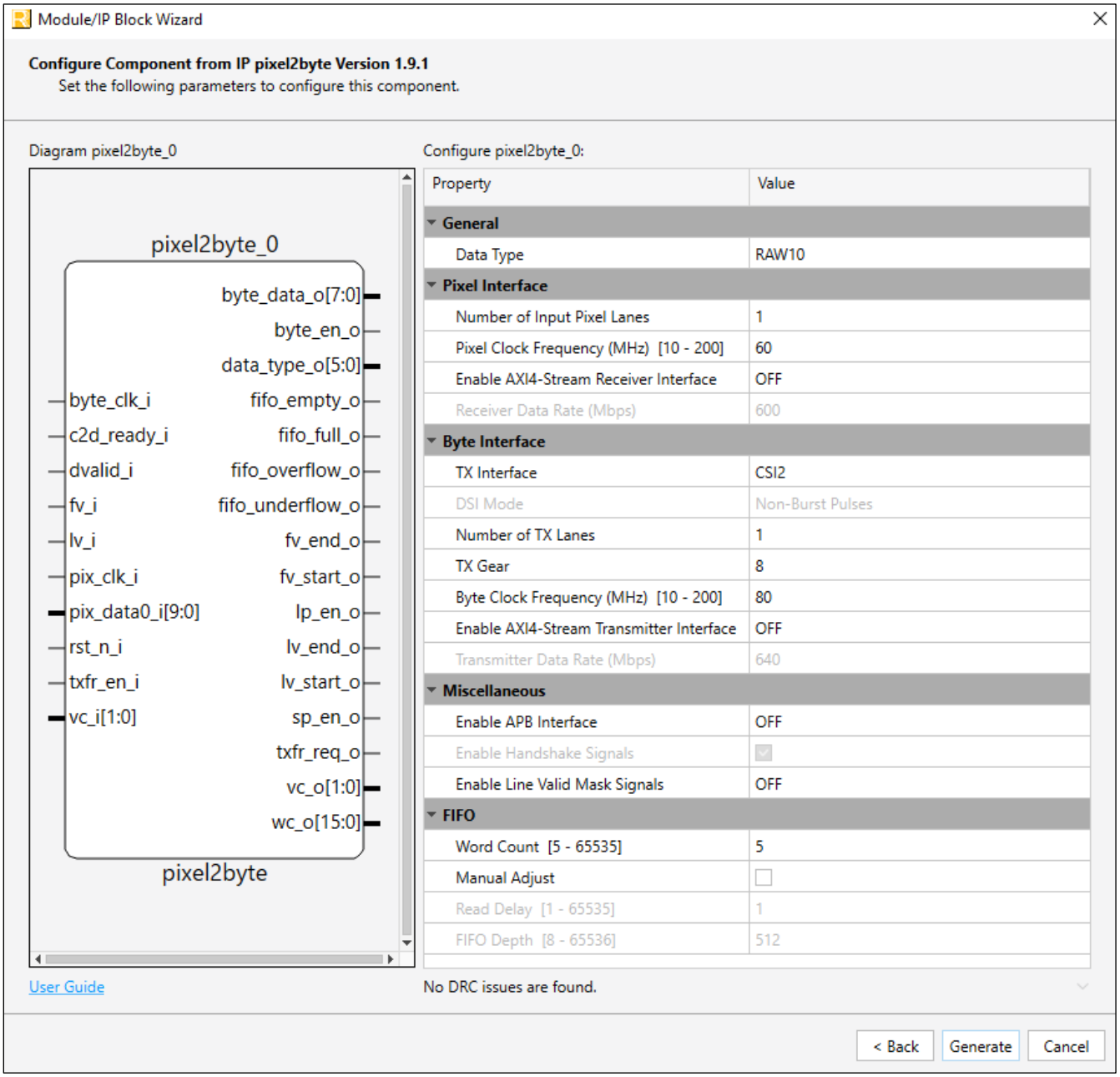

Figure 6.2. IP Configuration

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 6.3.

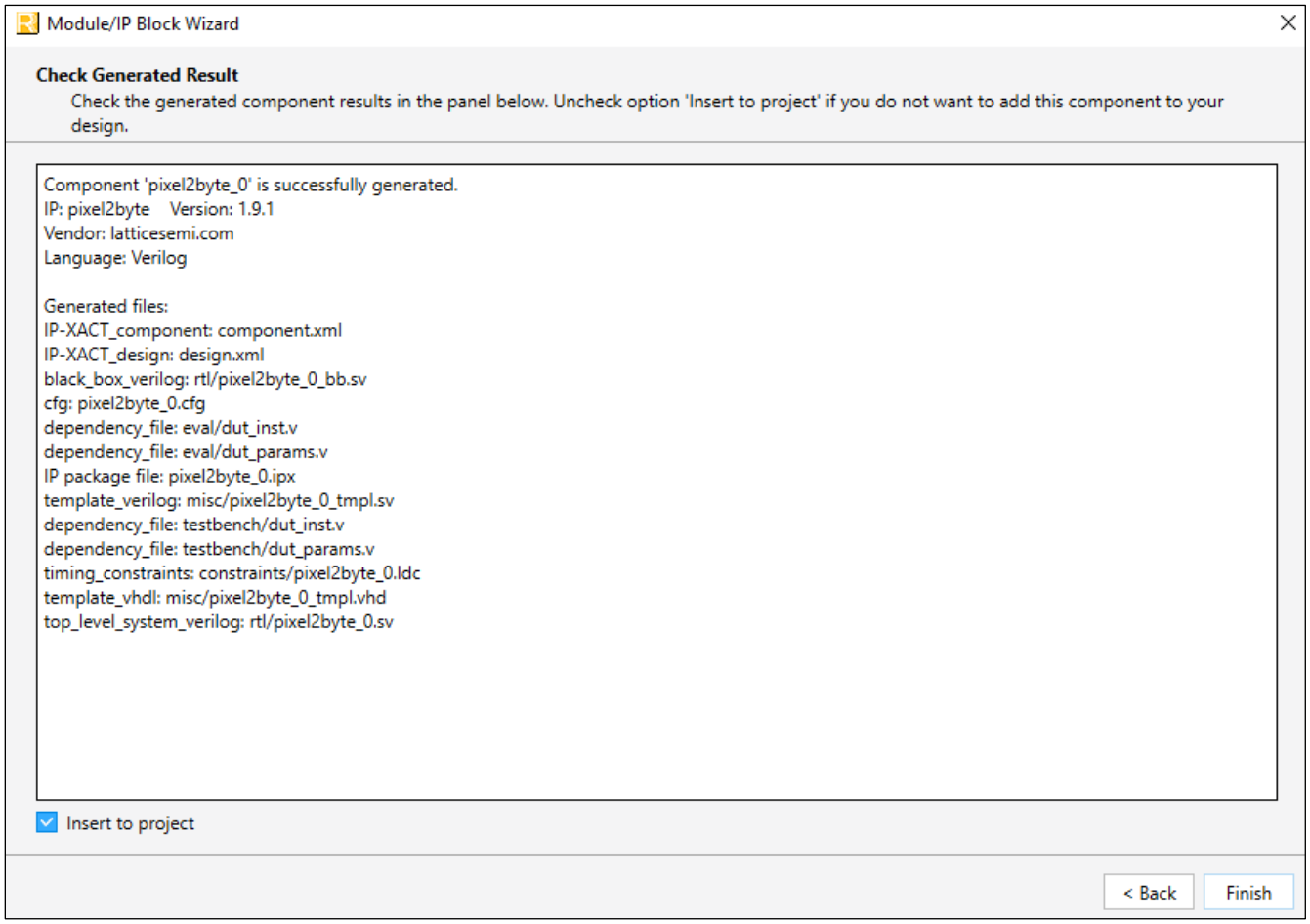

Figure 6.3. Generated Result

5. **Click Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 6.1.

6.1.1. Generated Files and File Structure

The generated Pixel-to-Byte Converter IP module package includes the closed-box (P2B_bb.v) and instance templates (P2B_tmpl.v/vhd) that can be used to instantiate the core in a top-level design.

An example RTL top-level reference source file (P2B.v) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design.

The generated files are listed in Table 6.1.
Note: The component name used in this example is *P2B* which is customizable based on your preference <Instance Name>.

Table 6.1. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format
rtl/<Instance Name>.v	This file provides an example RTL top file that instantiates the IP core.

Attribute	Description
rtl/<Instance Name>_bb.v	This file provides the synthesis closed-box.
misc/<Instance Name>_tmpl.v misc/<Instance Name>_tmpl.vhd	These files provide instance templates for the IP core.
eval/constraint.pdc	This file provides information on how to constrain the IP.

6.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC File.

Post-Synthesis constraint files (.pdc) contain both timing and non-timing constraint.pdc source files for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

6.3. Constraining the IP

To ensure that the design meets its desired performance goals on the FPGA, you need to provide proper timing and physical design constraints. The content of the following IP constraint file can be added to the user design constraints:

<IP Instance_Path>/<IP Instance_Name>/eval/constraint.pdc

The above constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. It can be modified but modifications should be made with thorough understanding of the effect of each constraint. Below are the steps on how to properly constrain the IP:

1. Copy the contents of constraint.pdc to the top-level design constraint for post-synthesis.
2. Modify the period for the *create_clock* constraints according to the desired value to be used in the IP.
3. Uncomment the *set_clock_groups* constraint only when the two clocks are asynchronous of each other.

Refer to the Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) for details on how to constraint your design.

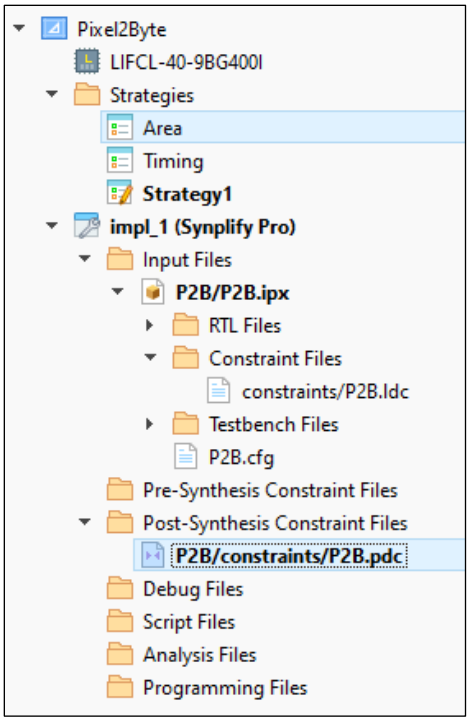

Figure 6.4. Adding Constraint

The example below shows the IP timing constraints generated for the Pixel-to-Byte Converter Module IP:

```
# ------------------------------ GENERAL NOTES ---------------------------------------#
## This post-synthesis constraint file is generated in LDC format and is compatible with
## Radiant SW. Note that “IP_INST” is a reserved keyword. This should be used as prefix of
## the variables used in the file.

# -------------------------------- SETTINGS ---------------------------------------#
Pixeset IP_INST_BYTECLK_PERIOD [expr{double(round(1000000/$IP_INST_BYTE_CLK_FREQ))/1000}]
set IP_INST_PIXELCLK_PERIOD [expr{double(round(1000000/$IP_INST_PIX_CLK_FREQ))/1000}]
set IP_INST_APB_PERIOD 4

## -------------------------------- CLOCKS  ---------------------------------------- ##
# -- PIXEL CLOCK -- #
create_clock -name {pix_clk_i} -period $IP_INST_PIXELCLK_PERIOD [get_ports pix_clk_i]
# -- BYTE CLOCK -- #
create_clock -name {byte_clk_i} -period $IP_INST_BYTECLK_PERIOD [get_ports byte_clk_i]
# -- APB CLOCK --#
create_clock -name {apb_pclk_i} -period $IP_INST_APB_PERIOD [get_ports apb_pclk_i]
```

Figure 6.5. Timing Constraints

6.4. Running Functional Simulation

6.4.1. Setting Up Test Bench

To set up testbench parameters:

The directory structure of the testbench is shown in Figure 6.6.

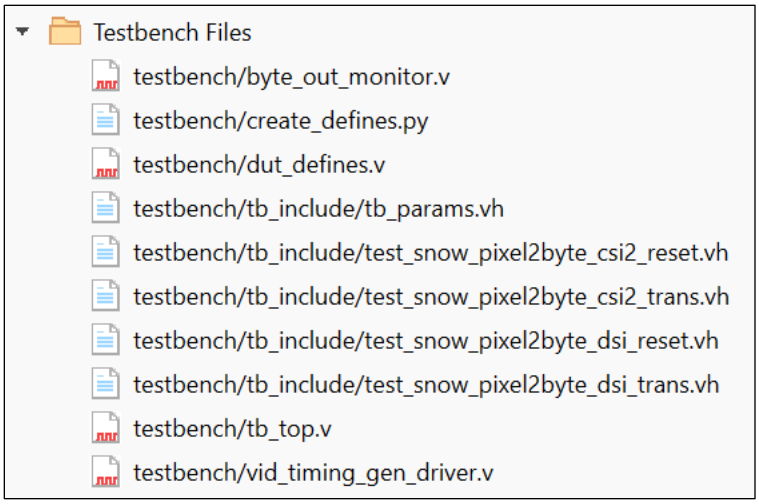

Figure 6.6. Testbench Directory Structure

The Pixel to Byte Converter IP for Radiant has a default testbench parameter file located in <ip_instance_path>/testbench/tb_include/tb_params.vh where the testcases and input frame parameter values to be used by the testbench can be set.

6.4.2. Running Simulation

You can run functional simulation after the IP is generated. To run functional simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 6.7.

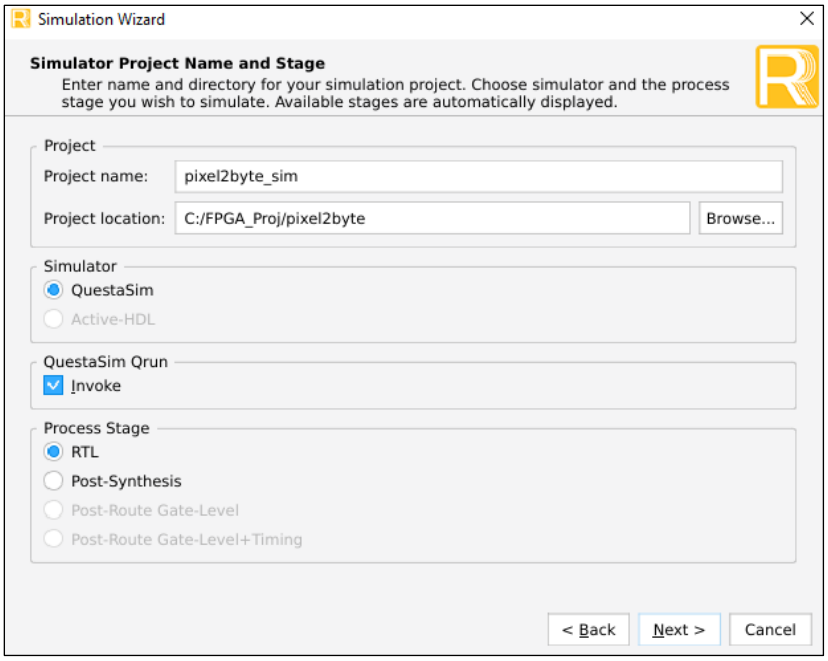

Figure 6.7. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 6.8.

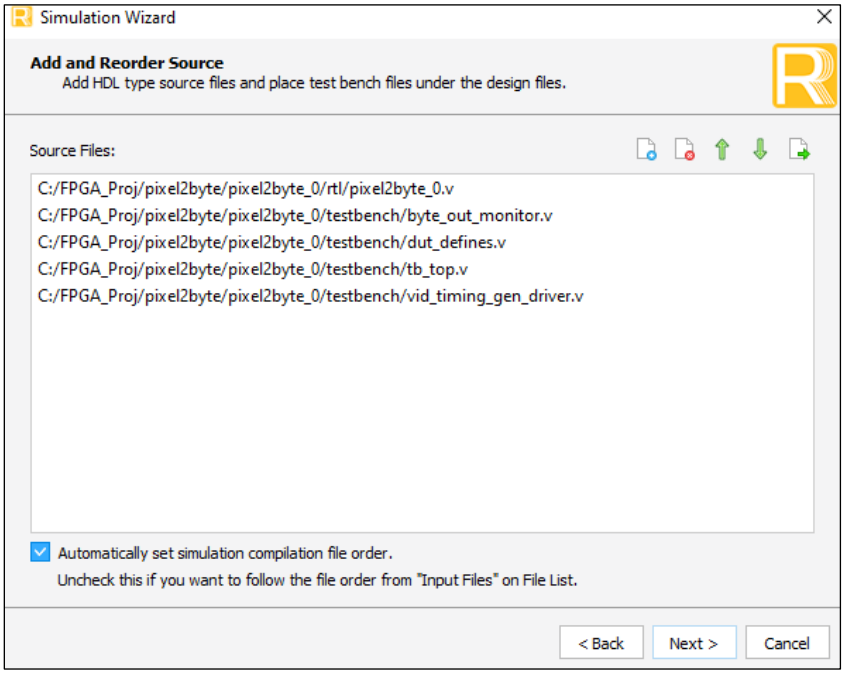

Figure 6.8. Add and Reorder Source

3. Add the *tb_top.v* file from the testbench directory. Refer to the Setting Up Test Bench section for the testbench setup details.
4. Click **Next**. The **Summary** window is shown.
5. Click **Finish** to run the simulation. The waveform in Figure 6.9 shows an example of the simulation result.

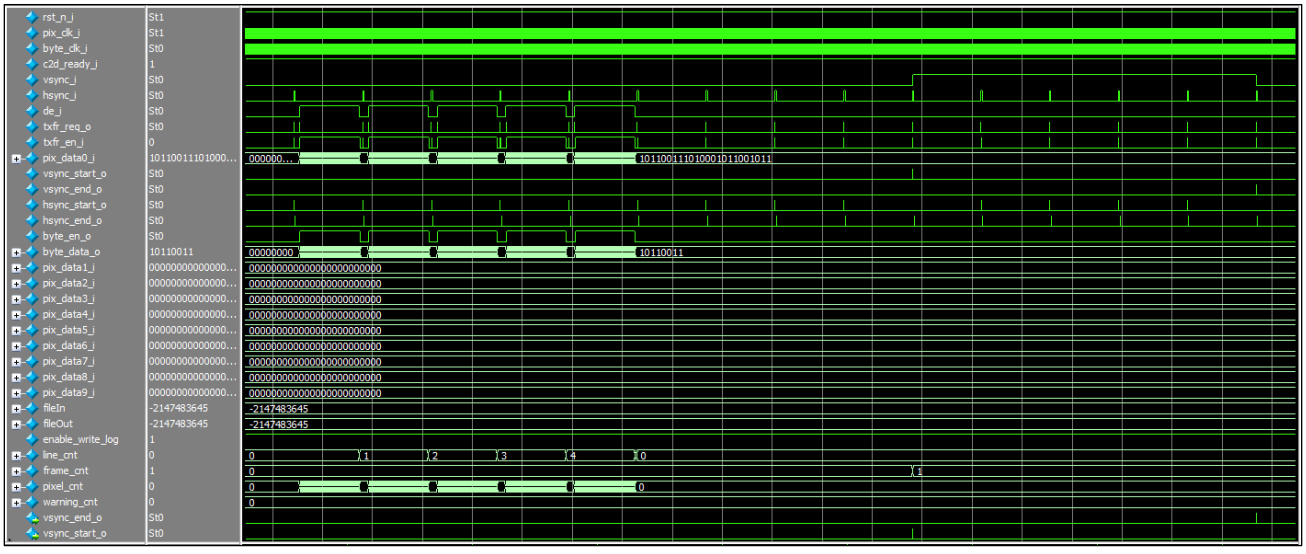

Figure 6.9. Simulation Wizard

Notes:

- It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant Software Suite.
- When simulation finishes successfully, “Simulation Passed/Failed” is displayed in the terminal.

6.4.3. Simulation Results

From the simulation waveforms, all signals for both input and output should be shown. In the transcript terminal, it shows information of how many pixels and blanking are being sent. Below is the sample display of the terminal.

```
# 0 TEST START
#
# No. of lines :            5
#
# Pixels per line :         1920
#
# HSYNC frontporch :          88
#
# HSYNC pulse width :          44
#
# HSYNC backporch  :         148
#
# VSYNC frontporch :           4
#
# VSYNC pulse width :           5
#
# VSYNC backporch  :          36
#
# Starting DSI drive task
# Starting drive_vsync_pulse_hsync...
# Done with drive_vsync_pulse_hsync...
# Starting drive_vbackporch_hsync...
# Done with drive_vbackporch_hsync...
# Starting drive_pixel_data...
# Sending line #:           1
# Sending line #:           2
# Sending line #:           3
# Sending line #:           4
# Sending line #:           5
# Done drive_pixel_data...
# Starting drive_vfrontporch_hsync...
# Done with drive_vfrontporch_hsync...
# Starting DSI drive task
# Starting drive_vsync_pulse_hsync...
# Done with drive_vsync_pulse_hsync...
# Starting drive_vbackporch_hsync...
# Done with drive_vbackporch_hsync...
# Starting drive_pixel_data...
# Sending line #:           1
# Sending line #:           2
# Sending line #:           3
# Sending line #:           4
# Sending line #:           5
# Done drive_pixel_data...
# Starting drive_vfrontporch_hsync...
# Done with drive_vfrontporch_hsync...
# Starting DSI drive task
# Starting drive_vsync_pulse_hsync...
# Done with drive_vsync_pulse_hsync...
# Starting drive_vbackporch_hsync...
# Done with drive_vbackporch_hsync...
# Starting drive_pixel_data...
# Sending line #:           1
# Sending line #:           2
# Sending line #:           3
# Sending line #:           4
# Sending line #:           5
# Done drive_pixel_data...
# Starting drive_vfrontporch_hsync...
# Starting drive_vfrontporch_hsync...
# Done with drive_vfrontporch_hsync...
# Starting DSI drive task
# Starting drive_vsync_pulse_hsync...
# Done with drive_vsync_pulse_hsync...
# Starting drive_vbackporch_hsync...
# Done with drive_vbackporch_hsync...
# Starting drive_pixel_data...
# Sending line #:           1
# Sending line #:           2
# Sending line #:           3
# Sending line #:           4
# Sending line #:           5
# Done drive_pixel_data...
# Starting drive_vfrontporch_hsync...
# Done with drive_vfrontporch_hsync...
# ------------------------------------------
# ------------------------------------------
##### DATA COMPARISON IS STARTED #####
# ------------------------------------------
# ------------------------------------------
# NO WARNINGS
# Test fail count : 0
#
# ------------------------------------------------------
# ----------------- SIMULATION PASSED ------------------
# ------------------------------------------------------
# 4175900000000 TEST END
```

Figure 6.10. Transcript Terminal

7. Design Considerations

- To ensure accurate pixel-to-byte conversion for RAW10 and RAW12 data types with input pixel lanes of 6, 8, and 10, adhere to the following guidelines:
  - *Transmitter Data Rate* must equal *Receiver Data Rate*.
  - The equivalent number of active pixels of the word count must be divisible by the *Number of Input Pixel Lanes*.
    Example: If *Number of Input Pixel Lanes == 6*, active pixels must be divisible by 6.
  - The equivalent total number of expected output bytes of the word count must be divisible by the (*Number of Tx Lanes* × *Tx Gear*) ÷ 8.
    Example: If *Number of Tx Lanes == 4*, *Tx Gear == 8*, the output byte must be divisible by 4.
    Example Configuration:
    - *Data Type == RAW10*
    - *Number of Input Pixel Lanes == 6*
    - *Number of Tx Lanes == 4*
    - *Tx Gear == 8*

    Valid word count: 240 bytes (satisfies divisibility by 6 and 4)
    - Total active bits: 240 × 8 = 1920 bits
    - Active pixels = 1920 ÷ 10 = 192 pixels (divisible by 6)
    - Expected output bytes = 240 bytes (divisible by 4)
- FIFO Depth of 512 cannot support a large (T_lpx + T_hs-prepare + T_hs-zero) value.

  If you need to set a larger *(T_lpx + T_hs-prepare + T_hs-zero)* value, then you have to check *Manual Adjust* in the GUI and increase the *FIFO Depth* according to the following formula:

  `((T_lpx + T_hs-prepare + T_hs-zero)/(1/Byte Clock Frequency))<(adjusted FIFO Depth - 5)`
- Make sure the new pixel data transmissions do not overlap with ongoing byte transmissions. This may be done by increasing the blanking period in between lines and frames or increasing the *Word Count* value. For more details see the Timing Specifications section for more information.
- When the Word Count value is very low, the pixel data written into the FIFO may be insufficient for correct byte data transmission. To avoid this scenario, increasing the blanking period can help ensure that long packets are not dropped.
- Pixel-to-Byte IP version 1.9.1 only supports handshaking with the Lattice CSI-2/DSI D-PHY Transmitter IP when using either the Soft or Hard D-PHY configuration with CIL Bypass option enabled.

Appendix A. Resource Utilization

The following tables show the details about the device, tools used, and resource utilization for a certain IP configuration.

Table A.1. CertusPro-NX Device and Tool Tested

Test Parameter	Value
Software Version	Lattice Radiant software 2025.1
Device Used	LFCPNX-100-7ASG256C
Performance Grade	7_High-Performance_1.0V
Synthesis Tool	Synplify Pro, May 2025

Table A.2. Resource Utilization on CertusPro-NX Device

Configuration	Pixel/Byte Interface	Register Interface	pix_clk Fmax (MHz)	byte_clk Fmax (MHz)	LUTs	Slice Registers	sysMEM EBRs	Programmable I/O
Default	Native	Off	150.8	161.3	170	182	1	64
DSI, RGB666, Number of TX Lanes=2, Others=Default	AXI4-Stream	APB	162.6	134.4	339	316	1	131
CSI-2, RGB888, Number of TX Lanes 4, Others=Default	Native	Off	135.2	161.8	226	312	3	102
CSI-2, RAW8, Number of TX Lanes=4, Others=Default	AXI4-Stream	APB	151.3	142.6	246	335	2	137
DSI, RGB888, Number of TX Lanes=4, TX Gear=16, Others=Default	Native	Off	112.3	144.4	300	483	6	153
CSI-2, RGB888, Number of TX Lanes=2, TX Gear=16, Others=Default	AXI4-Stream	APB	200	192.2	483	421	3	153
CSI-2, RAW10, Number of TX Lanes=2, Number of Input Pixel Lanes=2, Others=Default	Native	Off	155.9	150.9	211	225	2	82
CSI-2, RAW14, Number of TX Lanes=4, Number of Input Pixel Lanes=2, TX Gear=16, Others=Default	AXI4-Stream	APB	200	162.1	936	777	7	189

Notes:

1. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among *logic*, *distributed RAM*, and *ripple logic*.
2. For more information regarding a specific configuration, generate the IP, run synthesis, and run Map. Check the Map reports for resource utilization. Number may vary when using a different software version or targeting a different device density, synthesis tool, or speed grade. For better Static Timing Analysis performance, you are recommended to run multiple iterations of Place and Route and/or set Optimization Goal to Timing in the *Strategy* section of the software tool.

References

- Pixel-to-Byte Converter IP Release Notes (FPGA-RN-02020)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-N2 web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- Lattice Radiant web page
- Pixel to Byte Converter IP Core web page
- Lattice Solutions IP Cores web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Document Revision 2.0, IP v1.9.1, June 2025

Section	Change Summary
All	• Updated the IP version information on the cover page. • Made editorial fixes.
Abbreviations in This Document	Added *EBR*, *I/O*, *RAM*, *RGB*, *RTL*, *Rx*, and *Tx*.
Introduction	• Updated the description in the Overview of the IP section. • Updated Table 1.1. Quick Facts, Table 1.2. Ordering Part Number, and Table 1.3. IP Validation Level.
Functional Description	• Updated the Clocking section. • Added a caption for Table 2.5. Reset Signal.
Designing with the IP	Updated Figure 6.1. Module/IP Block Wizard, Figure 6.2. IP Configuration, and Figure 6.3. Generated Result.
Design Considerations	Updated this section.
Appendix A. Resource Utilization	Updated the resource utilization for the Lattice Radiant software version 2025.1.

Document Revision 1.9, IP v1.9.0, December 2024

Section	Change Summary
All	• Added the IP version information on the cover page. • Made editorial fixes.
Introduction	• Added the *Certus-N2* family device support to the Overview of the IP section. • Updated Table 1.1. Quick Facts and Table 1.3. IP Validation Level. • Added the *Certus-N2* OPNs to Table 1.2. Ordering Part Number.
Designing with the IP	Updated Figure 6.1. Module/IP Block Wizard, Figure 6.2. IP Configuration, and Figure 6.3. Generated Result in the Generating and Instantiating the IP section.
References	Added the *Pixel-to-Byte Converter IP Release Notes (FPGA-RN-02020)* and *Certus-N2* web page.

Document Revision 1.8, June 2024

Section	Change Summary
All	• Removed *Core* from the document title. • Made editorial fixes.

Abbreviations in This Document	• Replaced *acronyms* with *abbreviations* in this section.
	• Added the following abbreviations:
	  • *Advance Peripheral Bus (APB)*
	  • *Advance eXtensible Interface (AXI)*
	  • *First In, First Out (FIFO)*
	  • *Graphical User Interface (GUI)*
	  • *Intellectual Property (IP)*
	  • *Least Significant Bit (LSB)*
	  • *Look-Up Table (LUT)*
	  • *Mobile Industry Processor Interface (MIPI)*
	  • *Read Only (RO)*
	  • *Read and Write (RW)*
	  • *Write 1 to Clear (W1C)*

Introduction	• Added *IP Core v1.7.x* to Table 1.1. Quick Facts.
	• Updated the Supported Features section:
	  • Added *Features* to the section title.
	  • Added *RGB444*, *RGB555*, and *RGB565* data types.

Section	Change Summary
	• Added *AXI4 Streaming interface* and *APB interface* features. • Removed information on *crop pixel data* in the Unsupported Features section. • Added *Bundled* Ordering Part Number in Table 1.2. Ordering Part Number. • Updated the *Device Used*, *IP Version*, and *Software Version* in Table 1.3. IP Validation Level.

Functional Description	• Updated Figure 2.1. Pixel-to-Byte Converter IP Functional Diagram and its descriptions.
	• Renamed instances of *Pixel-to-Byte Wrapper Module* to *Pixel2Byte Core Module*.
	• Added Figure 2.2. Ports of Pixel-to-Byte for Pixel/Byte Native Interface and its descriptions.
	• Updated the descriptions in the Pixel-side Native Interface and Byte-side Native Interface sections.
	• Added the following sections and updated the header numbers of remaining sections accordingly:
	  • AXI4 – Stream Receiver Interface
	  • AXI4 – Stream Transmitter Interface
	  • APB Receiver Interface
	• Renamed the previous *Pixel-to-Byte Wrapper Module* section to *Pixel-to-Byte Core Module*.
	• Replaced the previous *Pixel2Byte Module* subsection with a new FIFO and FIFO Computations and Considerations subsections.
	• Updated the contents of the following sections:
	  • Synchronizer
	  • Clocking
	  • Timing Specifications
	• Updated Figure 2.9. Display Parallel Input Interface Timing Diagram (DSI) and Figure 2.10. Camera Sensor Parallel Input Interface Timing Diagram (CSI-2).
	• Removed the previous *Figure 2.13. Handshake Signals Timing Diagram* and its descriptions.
	• Added Figure 2.19. Handshake Signals Timing Diagram – Figure 2.22. Handshake Timing Diagram for Long Packet and their descriptions.

IP Parameter Description	• Updated Table 3.1. General Attributes:
	  • Added table notes.
	  • Updated the *Selectable Values*, *default value*, and *Dependency on Other Attributes* of the *Data Type* attribute.
	  • Added the following attributes:
	  • *Pixel Clock Frequency (MHz)*
	  • *Enable AXI4-Stream Receiver Interface*
	  • *Receiver Data Rate (Mb/s)*
	  • *Byte Clock Frequency (MHz)*
	  • *Enable AXI4-Stream Transmitter Interface*
	  • *Transmitter Data Rate (Mb/s)*
	  • *Enable APB Interface*
	  • *Enable Line Valid Mask Signals*
	  • *Word Count*
	  • *Manual Adjust*
	  • *Read Delay [1 – 65535]*
	  • *FIFO Depth [8-65536]*
	• Added Table 3.2. Byte Count Restriction and updated the header numbers of remaining tables accordingly.
	• Updated *Supported Configurations for DSI* for *4 Input Pixel Lanes* in Table 3.3. Supported Configurations for DSI.
	• Updated Table 3.4. Supported Configurations for CSI-2:
	  • Added *RGB444*, *RGB555*, and *RGB565* data types for the following data:
	  • *1 Input Pixel Lane*
	  • *2 Input Pixel Lane – 4 D-PHY Tx Lanes – 16 Tx Gearing*
	  • Updated *D-PHY Tx Lanes*, *Tx Gearing*, and *Data Type* for *4 Input Pixel Lanes*.

Section	Change Summary
Signal Description	Replaced all previous tables in this section with Table 4.1. Pixel-to-Byte IP Signal Description.
Register Description	Added this section and updated the header numbers of remaining sections accordingly.

Designing with the IP	• Updated the following figures:
	  • Figure 6.1. Module/IP Block Wizard
	  • Figure 6.2. IP Configuration
	  • Figure 6.3. Generated Result
	  • Figure 6.7. Simulation Wizard
	• Changed the term *black box* to *closed-box* in the Generated Files and File Structure section.
	• Removed previous *Table 5.2. Testbench Parameters* and its related information in the Setting Up Test Bench section.

Design Considerations	Updated this section.
References	Added references to *MachXO5-NX* and *Lattice Solutions IP Cores* web pages.

Document Revision 1.7, December 2023

Section	Change Summary
All	• Updated the document title from Pixel-to-Byte Converter IP Core – Lattice Radiant Software to Pixel-to-Byte-Converter IP Core. • Reworked the document structure for clarity by re-arranging section and subsections.
Introduction	• Reworked section contents. • Reworked old Section 4 – Ordering Part Number and converted to Subsection 1.4 Licensing and Ordering Information. • Added IP Validation Summary subsection. • Added Minimum Device Requirements subsection. • Reworked old Subsection 1.3 Conventions and renamed to Subsection 1.7 Naming Conventions.
Functional Description	• Added the User Interface subsection. • Reworked section contents.
IP Parameter Description	Reworked old Subsection 2.3 – Attributes Summary, and moved under this main section.
Signal Description	Reworked old Subsection 2.2 – Signal Description, and converted it to this main section
Designing with the IP	• Reworked old Section 3 – IP Generation, Simulation and Validation, and converted it to this main section. • Reworked old Subsection 3.2 – Running Functional Simulation and moved to this main section. • Reworked old Subsection 3.3 - Constraining the IP and moved to this main section
Design Considerations	Added this section.

Document Revision 1.6, December 2023

Section	Change Summary
Disclaimers	Updated the Disclaimers language.
Inclusive Language	Added this section.
Introduction	Updated Table 1.1. Quick Facts. • Added Avant-AT-G and Avant-AT-X devices in Supported FPGA Families.

Functional Description	• Updated Figure 2.1 contents
	• Signal Description subsection
	  • Updated Table 2.1. Pixel-to-Byte Converter IP Ports.
	  • Removed AXI Slave Interface.
	  • Removed AXI Master Interface.
	  • Removed APB Interface.
	• Attributes Summary subsection

Section	Change Summary

	• Updated Table 2.2. Supported Configurations
	• Updated Table 2.3, changed the default value of the Byte Interface to *DSI*
	• Removed Table 2.4. Word Count Restriction
	Removed data type : RGB444, RGB555 and RGB565 from Table 2.2.
	• Modules Description subsection
	  • Removed sections 2.4.3. AXI4 Stream Slave, 2.4.4. AXI4 Stream Master, and 2.4.5. APB Slave
	• Timing Specifications subsection
	  • Updated the following signals:
	  • fv to fv_i
	  • lv to lv_i
	  • dvalid to dvalid_i
	  • hsync to hsync_i
	  • vsync to vsync_i
	  • de to de_i

IP Generation, Simulation, and Validation	• Updated Figure 3.1. Configure Block of Pixel-to-Byte Converter.. • Updated Section 3.3. Constraining the IP. • Updated Figure 3.4. Simulation Wizard.. • Updated Figure 3.5. Adding and Reordering Source.. • Updated the steps to run Verilog simulation. Added step 5 and Figure 3.6 and Figure 3.7. • Changed Lattice Avant to *Lattice devices* under subsection 3.4 IP Evaluation.
Ordering Part Number	• Applied the latest OPN. • Added Ordering Part Numbers (OPN) for Avant-AT-G and Avant-AT-X devices.
Appendix A. Resource Utilization	• Updated Table A.1. Device and Tool Tested (LAV-AT E70) • Added Table A.2. Resource Utilization using LAV-AT-E70 • Added Table A.3. Device and Tool Tested (LAV-AT G70) • Added Table A.4. Resource Utilization using LAV-AT-G70..
References	Updated section contents.

Document Revision 1.5, December 2022

Section	Change Summary

Introduction	• Updated Table 1.1. Quick Facts.
	  • Revised Supported FPGA Families
	  • Revised Targeted Devices

Functional Description	• Updated Table 2.2. Supported Configurations and Table 2.3. Attributes Table. Added Table 2.4 Word Counr Restriction. • Updated Section 2.4.1. Clock, Rest and Initialization.
IP Generation, Simulation, and Validation	Updated the title of the Section 3. • Deleted Licensing the IP Section. • Updated Figure 3.1. Configure Block of Pixel-to-Byte Converter. • Updated the title of Section 3.1 from Generation and Synthesis to Generating the IP. • Updated the title of Section 3.2 Functional Simulation to Running Functional Simulation. • Updated Section 3.3. Constraining the IP. • Added Section 3.4. IP Evaluation.
Ordering Part Number	Added Avant-E part numbers.
Appendix A. Resource Utilization	Updated Table A.1. Device and Tool and Table A.2. Resource Utilization.
References	Added Avant webpage link in References section.

Document Revision 1.4, June 2021

Section	Change Summary

Introduction	• Removed *This document is for Pixel-to-Byte Converter IP design version 1.1*.
	Added CertusPro-NX in first paragraph.
	• Updated Table 1.1. Quick Facts.
	  • Revised Supported FPGA Families
	  • Revised Targeted Devices
	  • Revised Lattice Implementation to IP Core v1.3.x - Lattice Radiant software 3.0.

Functional Description	• Updated pix_data1_i1 description in Table 2.1. Pixel-to-Byte Converter IP Ports. • Updates values in Table 2.2. Supported Configurations.
IP Generation and Evaluation	Updated Figure 3.1. Configure Block of Pixel-to-Byte Converter.
Ordering Part Number	Added part numbers.
Appendix A. Resource Utilization	Updated Table A.2. Resource Utilization.
Appendix B. Limitation	Removed this section.
References	Added reference to the CertusPro-NX web page.

Document Revision 1.3, December 2020

Section	Change Summary
Introduction	• Updated Table 1.1. Modified Lattice Implementation details. • Added RAW14 and RAW16 to supported video formats.
Functional Description	Updated Table 2.2 and Table 2.3 in Attributes Summary.
IP Generation and Evaluation	Updated Figure 3.1.
Appendix A. Resource Utilization	Removed reference to Lattice Radiant software web page.
References	Updated this section. Added references to product web pages.
All	Updated Lattice Radiant Software User Guide references.

Document Revision 1.2, August 2020

Section	Change Summary
Introduction	Updated Table 1.1.
Functional Description	• Updated Table 2.1 and Table 2.3 in Attributes Summary • Updated Figure 2.14.
IP Generation and Evaluation	• Updated figures in this section. • Added Required Post-Synthesis Constraints section.
Appendix A. Resource Utilization	• Updated section content. • Added Table A.2.
Appendix B. Limitations	Added this section.

Document Revision 1.1, February 2020

Section	Change Summary
Introduction	Updated Table 1.1 and added Data Ordering and Data Types section.
Functional Description	Updated Table 2.3 in Attributes Summary.
IP Generation and Evaluation	Updated Figure 3.4 in Functional Simulation.
Appendix A. Resource Utilization	Updated Table A.1.

Document Revision 1.0, December 2019

Section	Change Summary
All	Initial release

www.latticesemi.com