Byte-to-Pixel Converter IP

User Guide

FPGA-IPUG-02079-1.9

June 2024

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
AXI	Advance eXtensible Interface
CSI-2	Camera Serial Interface-2
DSI	Display Serial Interface
EBR	Embedded Block RAM
FPGA	Field-Programmable Gate Array
I/O	Input/Output
LUT	Look-Up Table
RAM	Random Access Memory
RTL	Register Transfer Language

1. Introduction

1.1. Overview of the IP

Lattice Semiconductor Byte-to-Pixel Converter IP converts CSI-2/DSI standard-based video payload packets from D-PHY Receiver Module output to pixel format. In addition, Byte-to-Pixel Converter IP generates camera and video control signals in the pixel domain based on CSI-2 or DSI synchronization packets.

Figure 1.1 shows the Byte-to-Pixel Converter IP accepts CSI-2/DSI standard-based video payload packets and generates Pixel Format output.

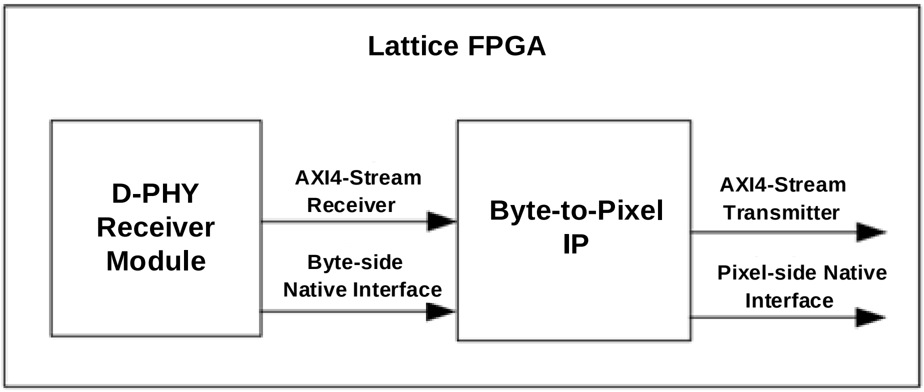


Figure 1.1. Byte-to-Pixel Convertor IP General Diagram

1.2. Quick Facts

Table 1.1. Summary of the Byte-to-Pixel Converter IP

IP Requirements	Supported FPGA Family	CrossLink™-NX, Certus™-NX, CertusPro™-NX, Lattice Avant™, MachXO5™-NX
	IP Version	IP Core v1.0.x – Lattice Radiant™ software 2.0 IP Core v1.1.x – Lattice Radiant software 2.1 IP Core v1.2.x - Lattice Radiant Software 2.2 IP Core v.1.3.x – Lattice Radiant Software 3.1 IP Core v.1.4.x – Lattice Radiant Software 3.1 IP Core v.1.5.x – Lattice Radiant Software 2022.1 IP Core v.1.6.x – Lattice Radiant Software 2023.1 IP Core v.1.7.x – Lattice Radiant Software 2024.1
Resource Utilization	Targeted Devices	LIFCL-40, LIFCL-17, LFD2NX-40, LFD2NX-17, LFCPNX-100, LAV-AT-E70, LAV-AT-G70, LAV-AT-X70, LFMXO5-25, LIFCL-33
	Supported User Interface	Native Interface AXI4-Stream Interface
	Resources	Refer to Appendix A. Resource Utilization
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys® Synplify Pro® for Lattice
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.

1.3. Features

The Byte-to-Pixel Converter IP supports:

- MIPI DSI compatible video formats
- MIPI CSI-2 compatible video formats
- 1-, 2-, or 4-lane inputs
- 8-bit (gear 8) or 16-bit (gear 16) inputs per lane
- 1, 2, or 4 output pixels per pixel clock cycle
- Burst mode, non-burst mode with sync events and non-burst mode with sync pulse
- AXI4-stream transmitter and receiver interface

1.4. Licensing and Ordering Information

An IP-specific license string is required to enable full use of the Byte-to-Pixel 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 Byte-to-Pixel 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 Machine Annual	**Multi-Site Perpetual**
CrossLink-NX	BYTE-PIXEL-CNX-US	BYTE-PIXEL-CNX-UT
Certus-NX	BYTE-PIXEL-CTNX-US	BYTE-PIXEL-CTNX-UT
CertusPro-NX	BYTE-PIXEL-CPNX-US	BYTE-PIXEL-CPNX-UT
MachXO5-NX	BYTE-PIXEL-XO5-US	BYTE-PIXEL-XO5-UT
Avant-AT-G	BYTE-PIXEL-AVG-US	BYTE-PIXEL-AVG-UT
Avant-AT-X	BYTE-PIXEL-AVX-US	BYTE-PIXEL-AVX-UT
Avant-AT-E	BYTE-PIXEL-AVE-US	BYTE-PIXEL-AVE-UT
Bundled	MIPI-BNDL-US	MIPI-BNDL-UT

1.5. IP Validation Summary

Table 1.3 shows the validation status for the Byte-to-Pixel Converter IP core. The ✓ mark indicates whether the IP has been validated for simulation, timing, or with hardware.

Table 1.3. IP Validation Level

Device Family	IP Version	Validation Level		
		Simulation	**Timing**	**Hardware**
Lattice Avant	v1.7.0	✓	✓	—

1.6. Minimum Device Requirements

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

1.7. Naming Conventions

1.7.1. Nomenclature

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

1.7.2. Signal Names

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

2. Functional Description

2.1. IP Architecture Overview

The Byte-to-Pixel Converter IP is used to convert a D-PHY CSI-2/DSI standard-based byte data stream to a standard-pixel data format. It acts as a conversion bridge between the CSI-2/DSI standard-based input byte data stream and the pixel-format output data stream. The optional AXI4-Stream of the Byte Domain Inputs/Outputs is used for receiving video payload packets.

Figure 2.1 shows the general block diagram of Byte-to-Pixel IP, FIFO, AXI4 Device Receiver, and AXI4 Device Transmitter to synchronize the incoming D-PHY data bytes to the pixel clock domain.

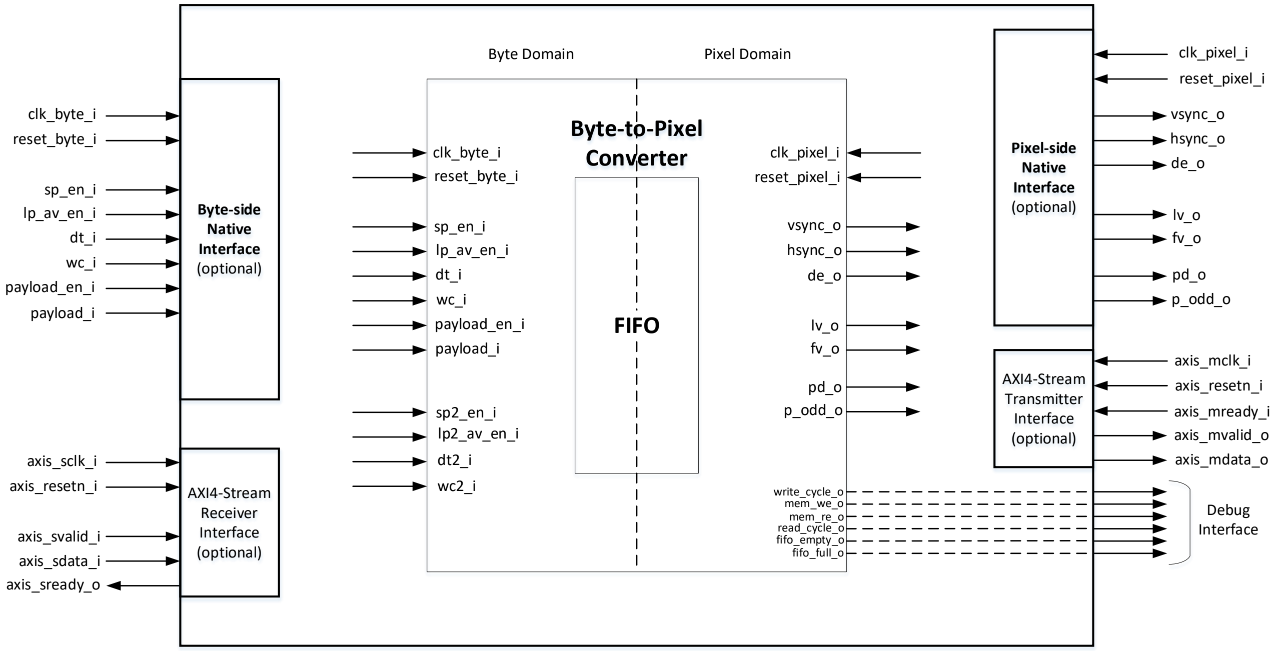


Figure 2.1. Byte-to-Pixel IP Functional Diagram

The Byte-to-Pixel Converter IP includes the following layers:

- Byte-to-Pixel Converter IP Core
- Byte-side Native Interface[1] (optional)
- AXI4-Stream Receiver Interface (optional)
- Pixel-side Native Interface[1] (optional)
- AXI4-Stream Transmitter Interface (optional)
- FIFO module
- Debug Interface (optional)

Note:

1. The Native Interface is disabled when its corresponding AXI stream interface is enabled, and vice versa.

Various configurations of the Byte-to-Pixel Converter IP are illustrated in the following Figure 2.2 – Figure 2.5.

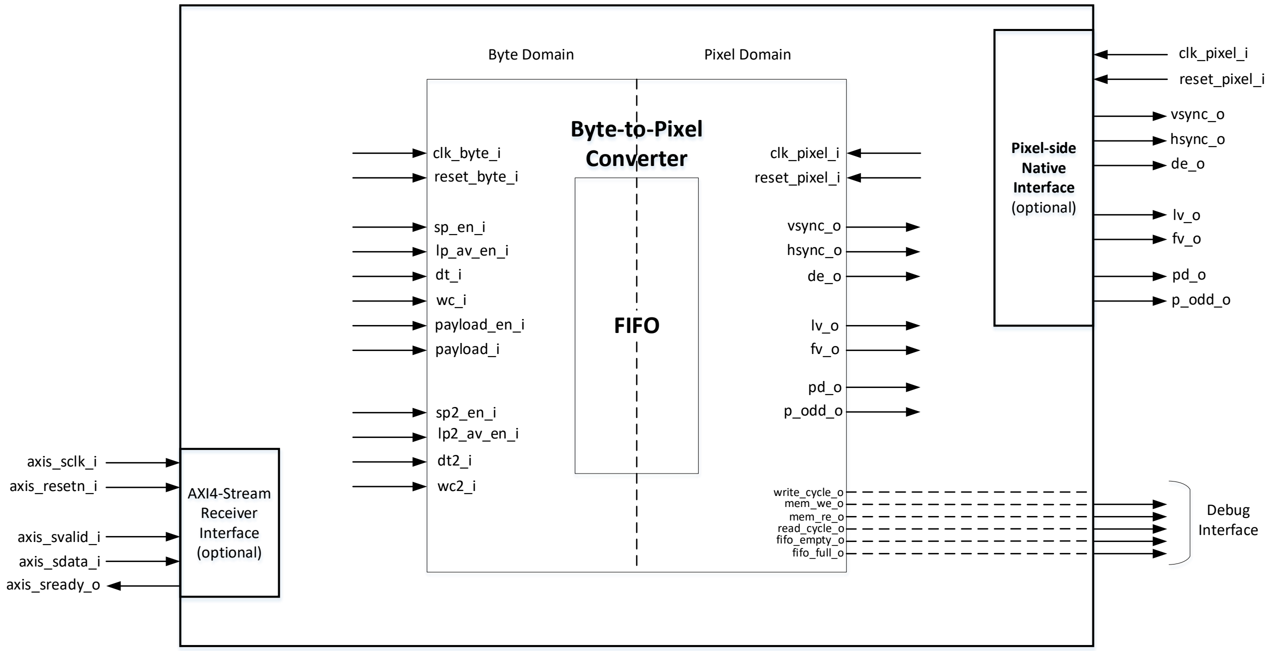


Figure 2.2. Byte-to-Pixel Converter IP Block Diagram with AXI4-Stream Receiver Interface Enabled

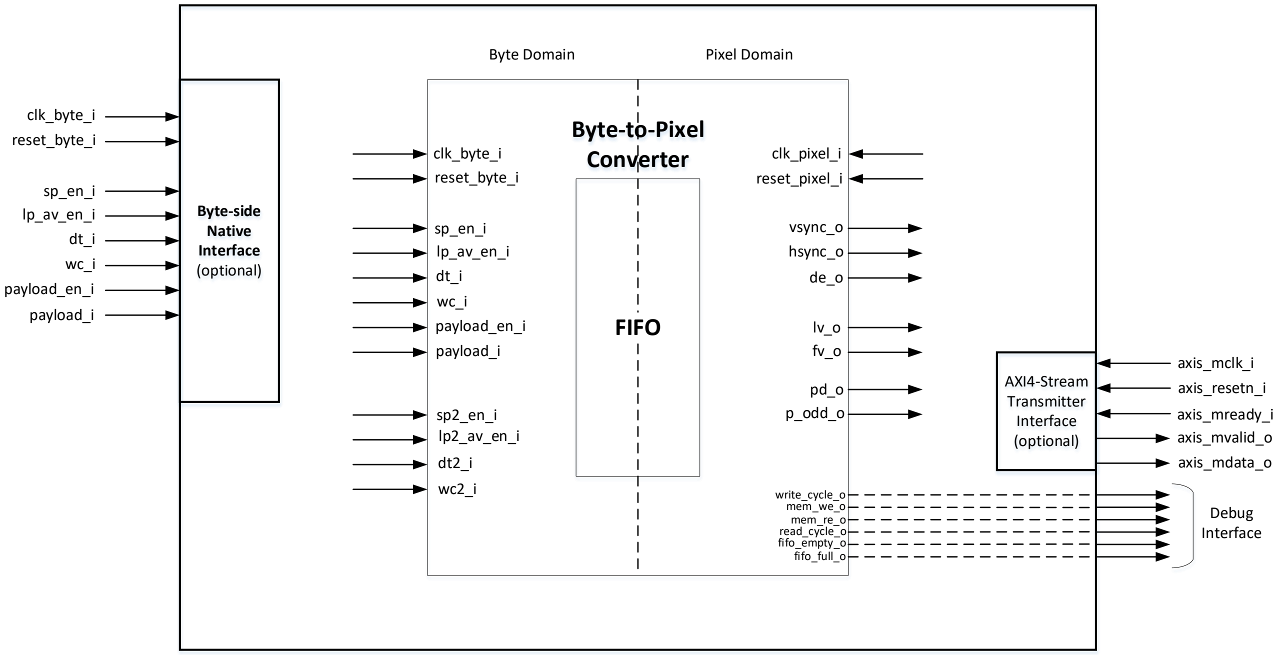


Figure 2.3. Byte-to-Pixel Converter IP Block Diagram with AXI4-Stream Transmitter Interface Enabled

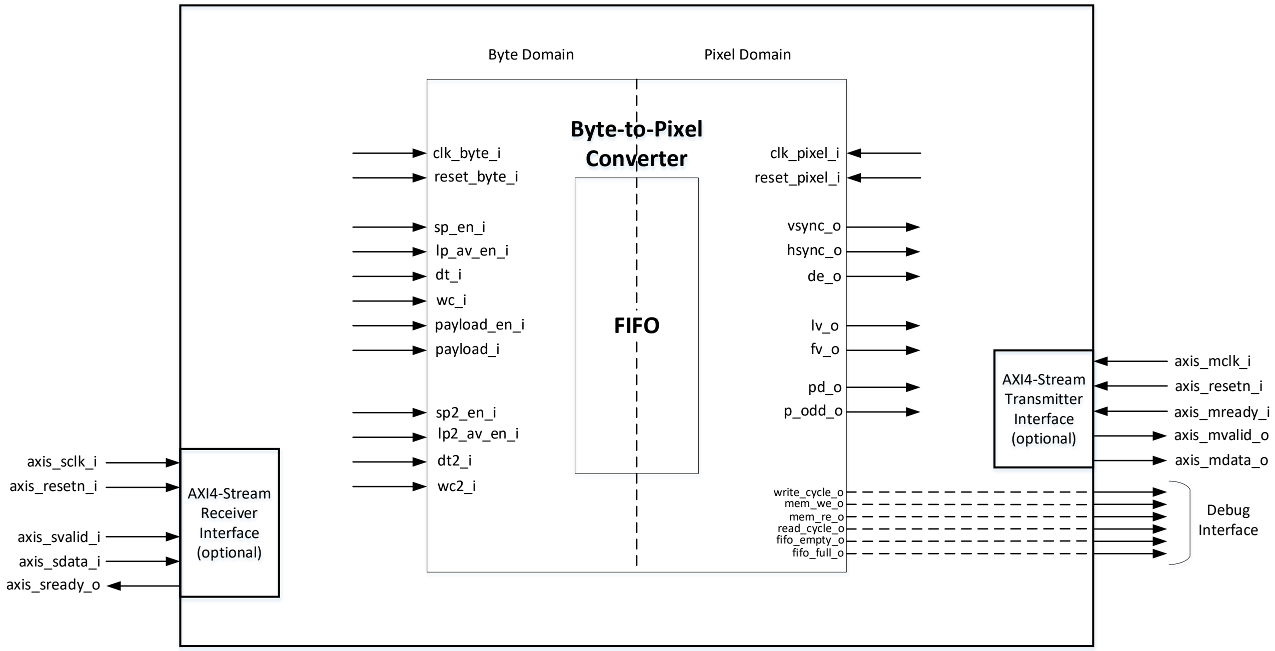


Figure 2.4. Byte-to-Pixel Converter IP Block Diagram with Both AXI4-Stream Interfaces Enabled

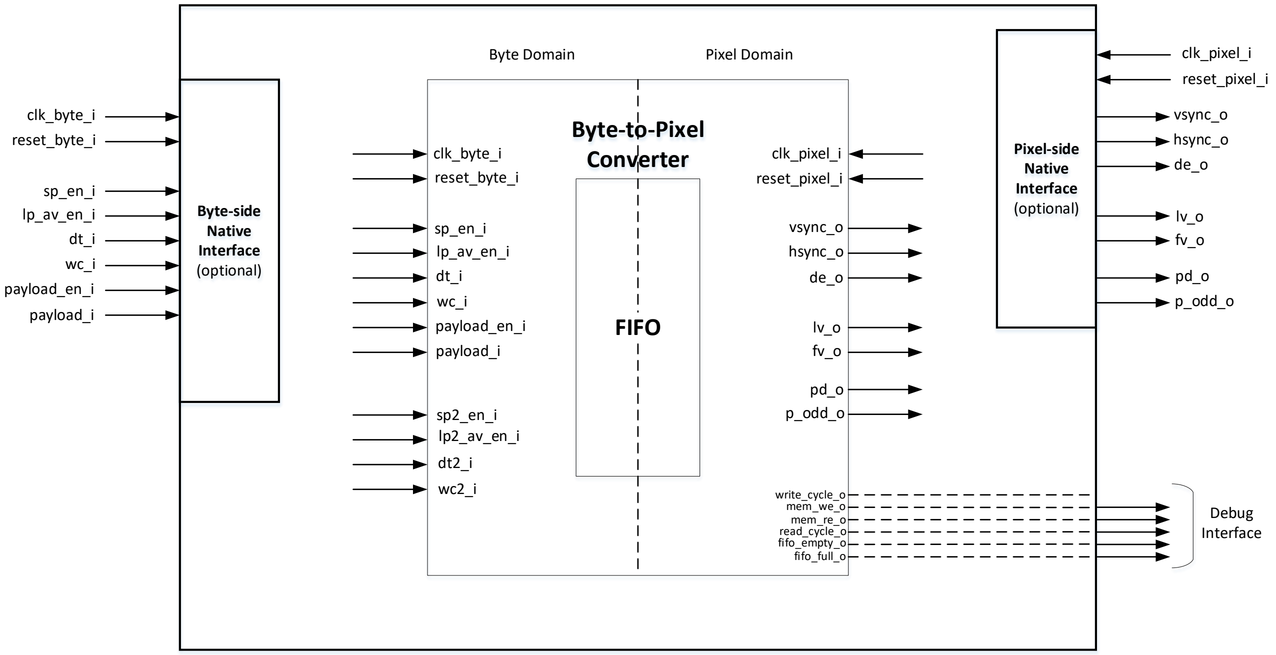


Figure 2.5. Byte-to-Pixel Converter IP Block Diagram with Both Native Interfaces Enabled

2.2. Clocking

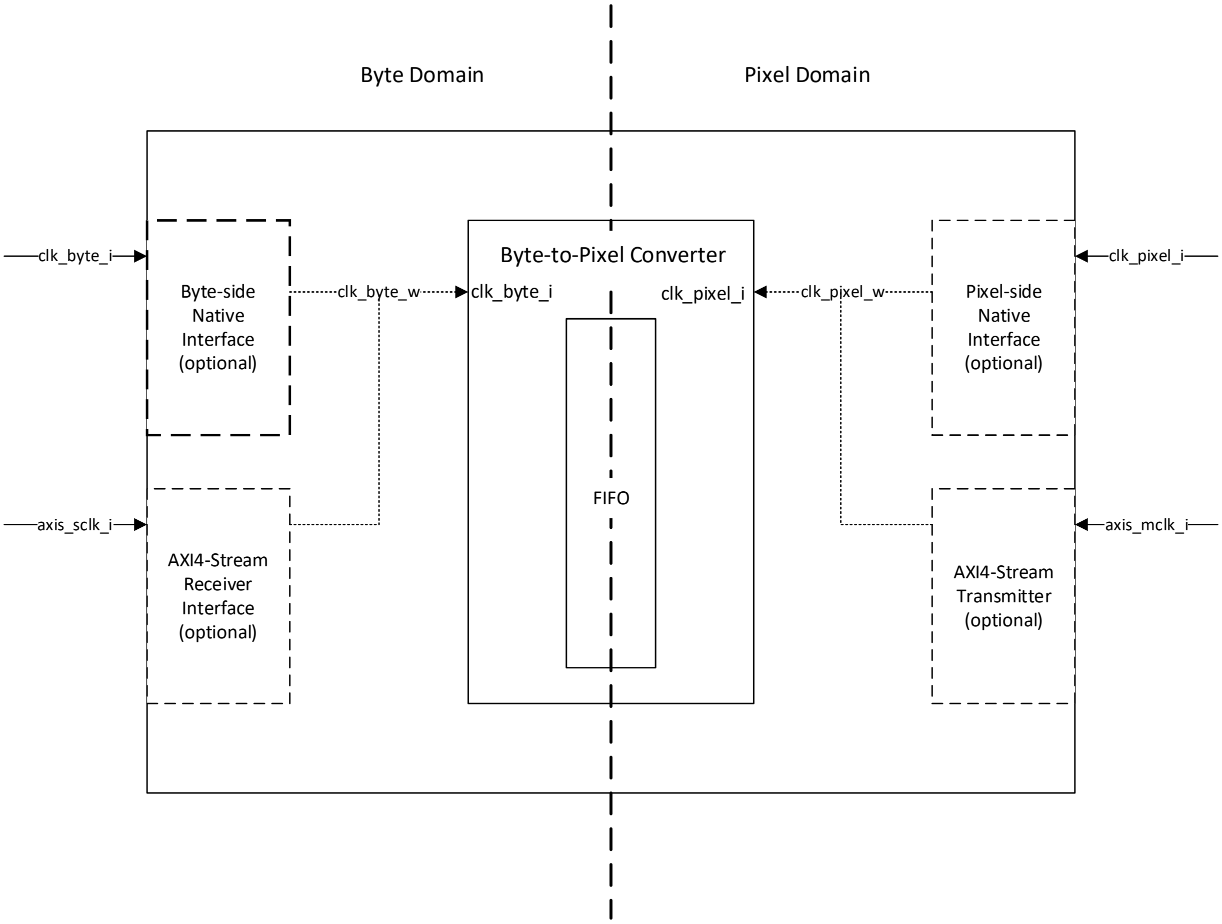

Byte Domain
Pixel Domain
clk_byte_i
Byte-side Native Interface (optional)
clk_byte_w
Byte-to-Pixel Converter
clk_byte_i
clk_pixel_i
FIFO
clk_pixel_w
Pixel-side Native Interface (optional)
clk_pixel_i
axis_sclk_i
AXI4-Stream Receiver Interface (optional)
AXI4-Stream Transmitter (optional)
axis_mclk_i

Figure 2.6. Clock Domain Crossing Block Diagram

Refer to the Clock Interface section for more information on Clock Signal Interface and requirement.

2.3. Reset

Table 2.1. Reset Signal

Reset Signal	Direction	Description
reset_byte_n_i	Input	• System reset • Active low signal to reset the logic in the Byte Domain Native Interface
axis_sresetn_i	Input	Active low signal to reset the logic in the AXI4-Stream Receiver Interface
reset_pixel_n_i	Input	Active low signal to reset the logic in the Pixel Domain Native Interface
axis_mresetn_i	Input	Active low signal to reset the logic in the AXI4-Stream Transmitter Interface

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

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

Note:

[1] The slower clock can be either Pixel Interface Clock (clk_pixel_i) or Byte Interface Clock (clk_byte_i).

2.4. User Interfaces

Table 2.2 lists the available user interface and protocols used on the Byte-to-Pixel Converter IP.

Table 2.2. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Byte Domain Native Interface	MIPI 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.
Pixel Domain 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.
AXI4-Stream Receiver Interface	AXI4-Stream Protocol	Receives the payload data (byte data with data type and word count).
AXI4-Stream Transmitter Interface	AXI4-Stream Protocol	Transmits the pixel data.

2.4.1. Native Interfaces

This section contains operational timing diagrams applicable to the Byte-to-Pixel IP Native interfaces. Figure 2.7 shows the timing between input and output for DSI.

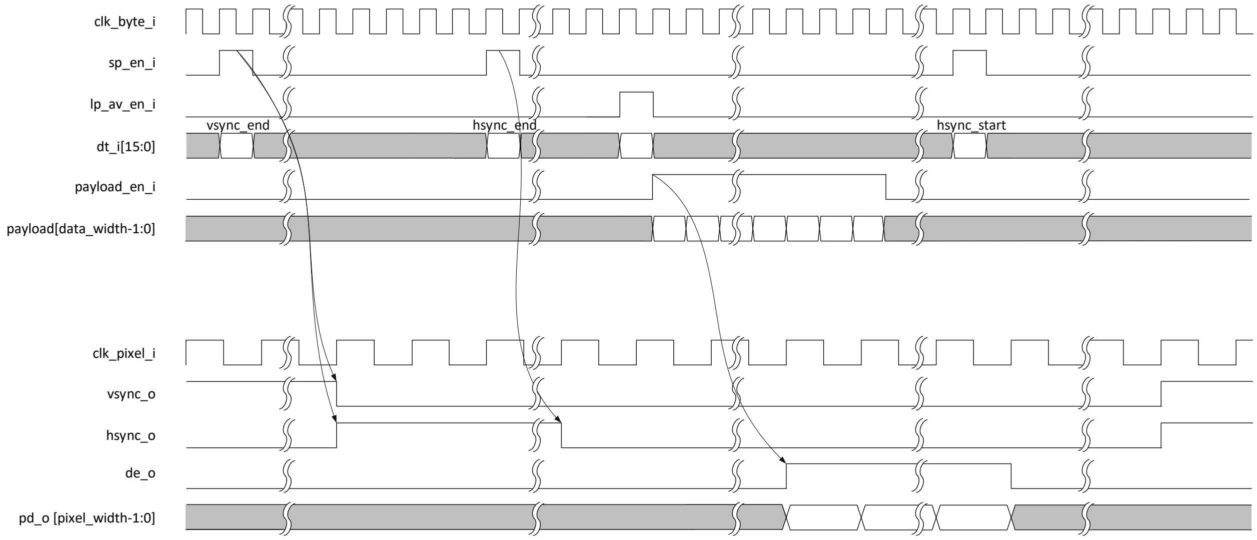


Figure 2.7. Timing Diagram between Inputs and Outputs for DSI

Reception of a VSYNC start packet triggers the assertion of both hsync_o and vsync_o signals. VSYNC end packets, on the other hand, trigger the deassertion of the vsync_o signal and the assertion of hsync_o signal.

Figure 2.8 shows the timing between input and output for CSI-2.

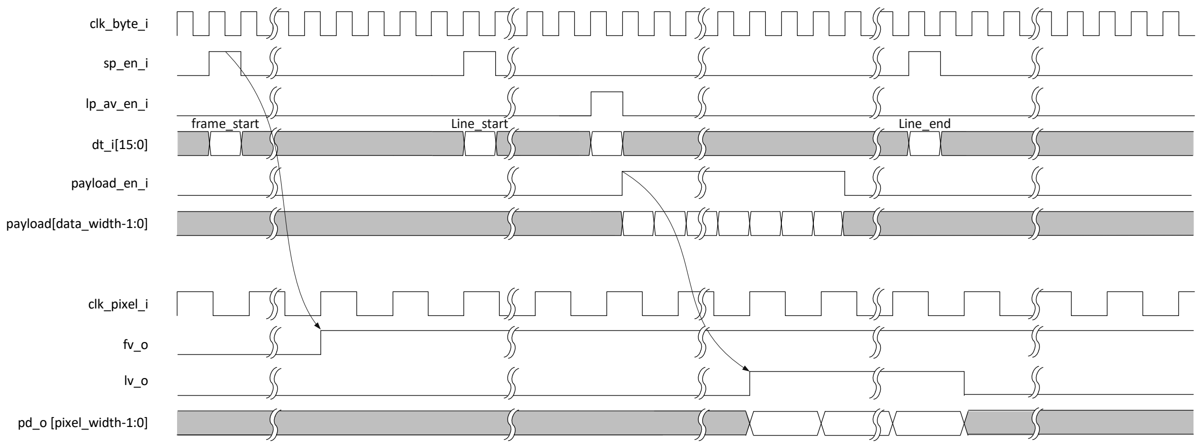


Figure 2.8. Timing Diagram between Inputs and Outputs for CSI-2

The behavior of the output synchronization signals (frame and line valid for CSI-2, and VSYNC and HSYNC for DSI) depend on the reception of the corresponding short packets. Due to the crossing of clock domains, pulse width and intervals between pulses may vary.

The pixel data is buffered and processed differently than the sync packets. Because of this, the pixel data might come out later that the sync signals. For the DSI protocol, the DSI Sync Packet Delay attribute may be increased to reduce the skew between the sync signals and the pixel data.

Figure 2.9 shows the timing diagram of the interface at the receiver side. The assertion of the payload_en_o with respect to the lp_av_en_i may vary depending on the gearing and number of lanes. The signals dt_i, vc_i and wc_i must be valid with the assertion of the lp_av_en_i.

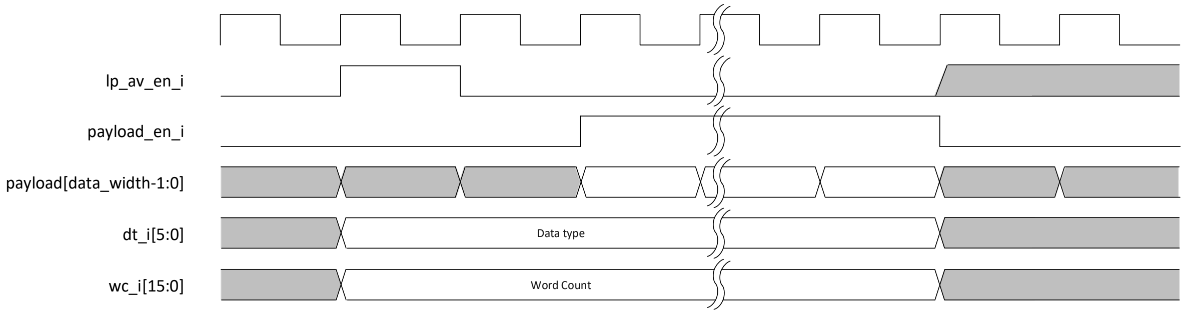


Figure 2.9. Input Timing Diagram for Byte Domain

The output timing from a DSI input is shown in Figure 2.10 while the timing from a CSI-2 input is shown in Figure 2.11.

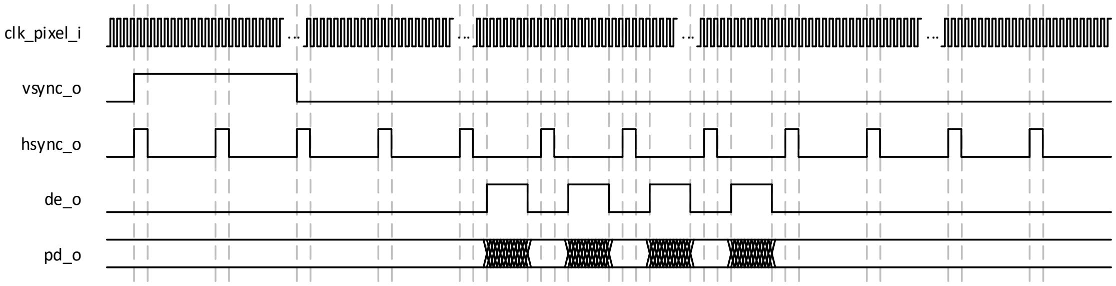


Figure 2.10. Output Timing Diagram from a DSI Input

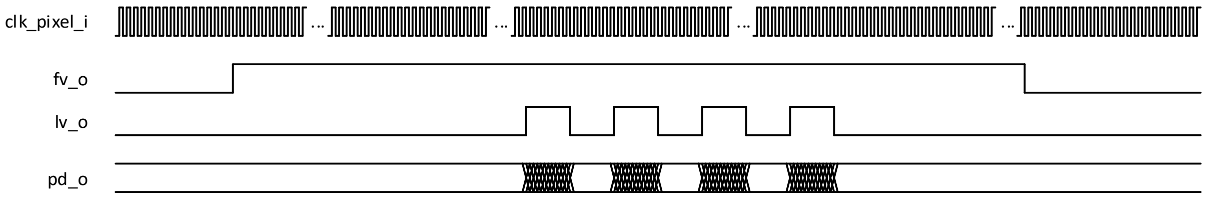


Figure 2.11. Output Timing Diagram from a CSI-2 Input

2.4.2. AXI4-Stream Receiver Interface

The AXI4-Stream Receiver provides transmission of payload packets to the Byte-to-Pixel Converter module. The axis_sready_o is always set to *1*. When axis_svalid_i is asserted to *1*, then axis_sdata_i receives the payload video stream. Figure 2.12 shows data format when AXI4-Stream Receiver interface is enabled.

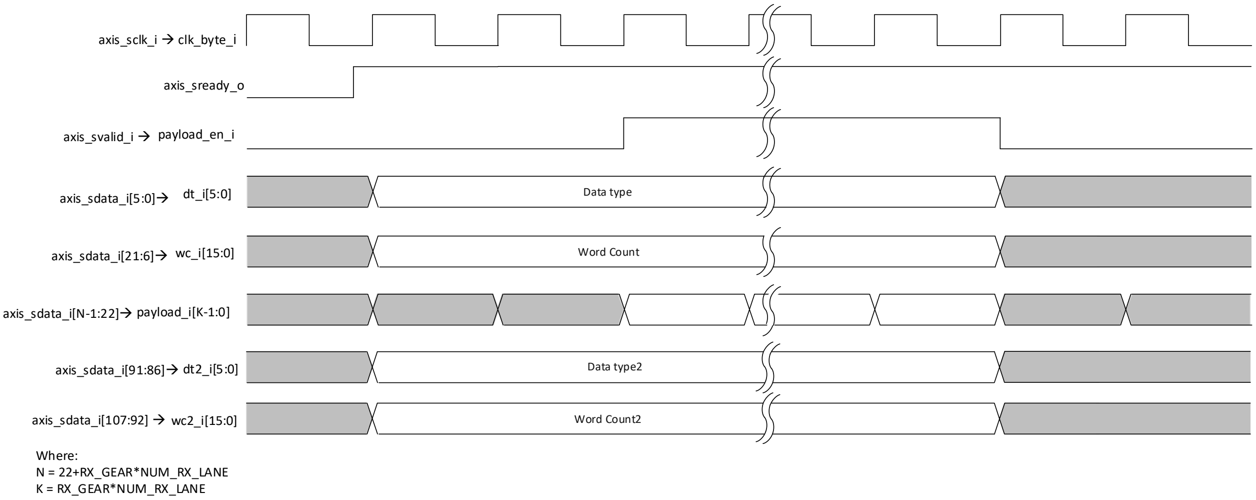

axis_sclk_i → clk_byte_i
axis_sready_o
axis_svalid_i → payload_en_i
axis_sdata_i[5:0]→ dt_i[5:0]
Data type
axis_sdata_i[21:6]→ wc_i[15:0]
Word Count
axis_sdata_i[N-1:22]→ payload_i[K-1:0]
axis_sdata_i[91:86]→ dt2_i[5:0]
Data type2
axis_sdata_i[107:92] → wc2_i[15:0]
Word Count2
Where:
N = 22+RX_GEAR*NUM_RX_LANE
K = RX_GEAR*NUM_RX_LANE

Figure 2.12. AXI4 Stream Receiver Interface Diagram

If the AXI4-Stream Receiver is not enabled, the following signals turn to top-level input signals:

- payload_en_i
- payload_i
- dt_i
- dt2_i
- wc_i
- wc2_i

2.4.3. AXI4-Stream Transmitter Interface

The AXI4-Stream Transmitter provides transmission of data converted to pixel format. Figure 2.13 shows data format when AXI4-Stream Transmitter interface is enabled.

For DSI and CSI-2 modes, the output data is the concatenation of two signals {p_odd_o, pd_o}, respectively, in that order. As the p_odd_o is a two-bit signal, the total width of AXI4 stream data (axis_mdata_o) is equal to 2 + pixel_width × number_of_pixels. Data valid (axis_mvalid_o) becomes active when de_o signals become *1*.

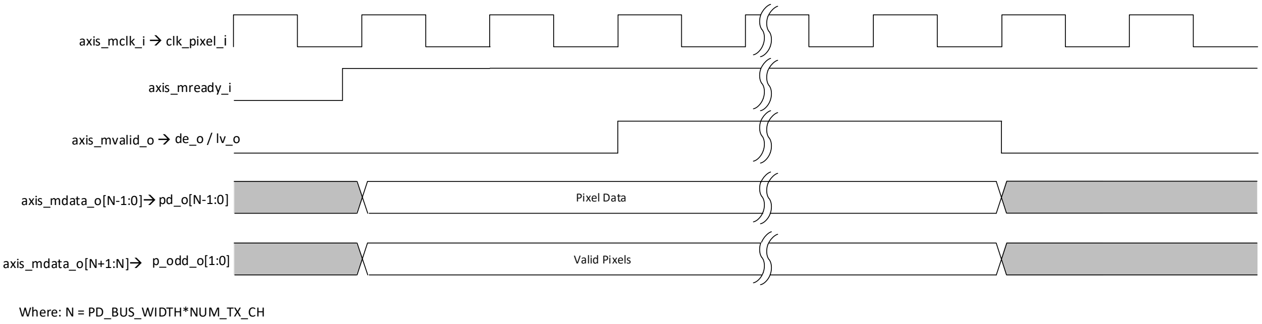

axis_mclk_i → clk_pixel_i
axis_mready_i
axis_mvalid_o → de_o / lv_o
axis_mdata_o[N-1:0]→ pd_o[N-1:0]
Pixel Data
axis_mdata_o[N+1:N]→ p_odd_o[1:0]
Valid Pixels
Where: N = PD_BUS_WIDTH*NUM_TX_CH

Figure 2.13. AXI4-Stream Transmitter Interface Diagram

If the AXI4-Stream Transmitter is not enabled, the following signals turn to top-level input signals:

- de_o[1]
- lv_o[2]
- pd_o
- p_odd_o

Notes:

1. Data Enable signal when RX Interface == DSI.
2. Data Enable signal when RX Interface == CSI-2.

2.5. Other IP Specific Blocks/Layers/Interfaces

2.5.1. Byte-to-Pixel Converter IP Core

Byte-to-Pixel Converter is a FIFO-based block that provides conversion of D-PHY CSI-2 or DSI video payload packets to standard pixel format through generic CSI-2/DSI standard-based input/output signaling.

2.5.2. FIFO Implementation

The FIFO depth is defined based on the design configuration, as shown in the IP Parameter Description section. The width of the FIFO is the lowest multiple of the output pixel data width that is greater than or equal to the input bus width. This determines the number of pixels that are grouped together and written to the same FIFO address. To avoid the FIFO full or empty states, the Data Safe Zone is defined by setting the data Overflow/Underflow Threshold attribute in Table 3.1. The threshold value is used to trigger the start of reading from the FIFO.

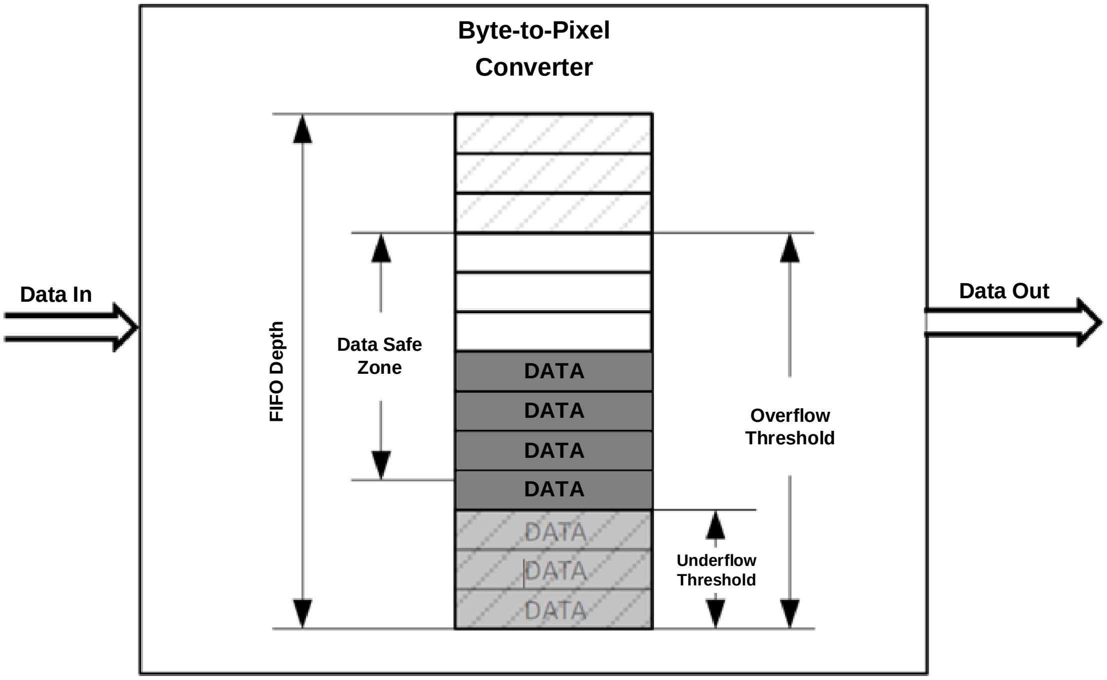


Figure 2.14. Byte-to-Pixel IP FIFO Diagram

The FIFO has the following three attributes:

- FIFO_Width
- FIFO_Depth
- FIFO_Threshold

The FIFO_Threshold is used to synchronize data flows between byte and pixel sides. Data transfer on the pixel side is started after the FIFO_Threshold is reached.

The FIFO_Width, FIFO_Depth, and FIFO_Threshold have dependencies on several attributes listed below:

- DATA_TYPE – Supported Data Type (available on the user interface)
- NUM_RX_LANE – Number of Rx Lanes (available on the user interface)
- RX_GEAR – Rx Gear (available on the user interface)
- Byte Clock Frequency (clk_byte_i) – Frequency on the Byte side (available on the user interface)
- NUM_TX_CH – Number of Output Pixel Lanes (available on the user interface)
- PD_BUS_WIDTH – Pixel Data Bus Width (not available on the user interface; however, it depends only on the data type that is available on the user interface; refer to the Pixel and Byte Count Restriction section.
- Pixel Clock Frequency (clk_pixel_i) – Frequency on the Pixel side (available on the user interface)
- Word Count (WC) - Number of bytes in a line transaction (available on the user interface)
- Useful Rate – Used for Data Types that don't have don't care bits (not available on the user interface)

The Useful_Rate depends on DATA_TYPE and is used for easier byte alignment. So,

```
    If DATA_TYPE = RGB666_LOOSE
         Useful_Rate = 6.0/8
    Else if DATA_TYPE = YCbCr422_20_LOOSE
          Useful_Rate = 40.0/48
    Else
          Useful Rate = 1.0
```

Denote Rx_width as the word width on the byte side and Tx_width as the word width on the pixel side. It can be calculated as:

- Rx_width = NUM_Rx_LANE × Rx_GEAR
- Tx_width = PD_BUS_WIDTH × NUM_Tx_CH

The FIFO width depends on DATA_TYPE, Rx_width values, and Tx_width values. It is the minimum multiple of Tx_width, which is not less than Rx_width. When using data types RAW12, RAW14, and RAW16, FIFO Width also depends on the Byte Count Restriction. So,

```
    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 = RAW16, then
          FIFO_Width = 2.0 * Rx_width
    Else,
          FIFO_Width = TX_width × ceil(RX_width / TX_width)
```

In addition, denote the data flow speed on the byte side as Rx_rate and the data flow speed on the pixel side as Tx_rate. It can be calculated as:

- RX_rate = RX_width × clk_byte_i x Useful_Rate
- TX_rate = TX_width × clk_pixel_i

The FIFO Depth (minimum value) and FIFO Threshold depend on the ratio (N_ratio) of the data flow speed on the byte side (RX_rate) and data flow speed on the pixel side (TX_rate). The RX_rate should not be less than TX_rate.

The value of FIFO Depth and FIFO Threshold is doubled when using data types YUV420_8 and YUV420_10; this is to make the even lines in the data transaction fit within the FIFO.

- N_ratio = TX_rate/Rx_rate

FIFO Depth and FIFO Threshold can be calculated by the following formulas:

```
    If (TX_rate == RX_rate)
        FIFO_Threshold = 4;
        FIFO_Depth = 16;
    Else
        FIFO_Threshold = int(math.ceil(8 × WC × (N_ratio-1)/(N_ratio × FIFO_Width) + 1));
        FIFO_Depth = 2 ** clog2(FIFO_Threshold + 6);
```

This computation will satisfy the needed FIFO_Threshold value to properly process the incoming data according to the configuration when the WC set by the user is not enough for the FIFO to write data.

```
    If (FIFO_Threshold > ((8 x WC) / FIFO_Width):
                  FIFO_Threshold = int(math.ceil((8 x WC)/FIFO_Width);
```

2.5.3. Debug Interface

See the Debugging section for more information about this interface.

3. IP Parameter Description

The configurable attributes of the Byte-to-Converter 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** font.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
General		
Data Type	RGB565, RGB666, RGB666_LOOSE, RGB888, RAW8, **RAW10**, RAW12, RAW14, RAW16, YUV420_8, YUV420_8_CSPS, LEGACY_YUV420_8, YUV420_10, YUV420_10_CSPS, YUV422_8, YUV422_10, YCbCr422_16, YCbCr422_20_LOOSE, YCbCr422_24	Byte-to-Pixel Converter IP supported data types. Refer to the Supported Configurations for DSI and Supported Configurations for CSI-2 sections for configuration options.
Byte Interface		
RX Interface	DSI, **CSI-2**	Byte-to-Pixel Converter IP RX interface.
DSI Mode	**Non-Burst Pulses**, Non-Burst Events, Burst	DSI Modes. Configurable when *RX Interface* == *DSI*.
Number of RX Lanes	**1**, 2, 4	Number of Byte-to-Pixel Converter RX high-speed ports. Refer to the Supported Configurations for DSI and Supported Configurations for CSI-2 sections for configuration options.
RX Gear	**8**, 16	Number of Byte-to-Pixel Converter RX gear. Refer to the Supported Configurations for DSI and Supported Configurations for CSI-2 sections for configuration options.
Byte Clock Frequency (MHz) [10–250]	**10**–250	Byte Clock Frequency. The upper frequency limit varies for below devices: • Avant devices: 250 • non-Avant devices: 200
Enable AXI4-Stream Receiver Interface	Checked, **Not Checked**	Enables AXI4-Stream Receiver interface.
Receiver Data Rate	Calculated	Non-configurable.
Pixel Interface		
Number of Output Pixel Lanes	**1**, 2, 4	Byte-to-Pixel Converter IP supported number of output pixel lanes. Refer to the Supported Configurations for DSI and Supported Configurations for CSI-2 sections for configuration options.
Camera/Display Control Polarity	**Positive**, Negative	—

Attribute	Selectable Values	Description
Number of HSYNC Pulses Inside VSYNC Active Region 3–1023]	3–1023, **5**	When the video mode is non-burst with sync events, this is used to determine the deassertion of the vsync_o signal. When non-burst with sync pulses, this is used only by the testbench for simulation; the actual vsync_o deassertion still depends on the reception of the vsync end packet. Configurable when *RX Interface = DSI*.
Number of Pixel Clock Cycles HSYNC is Active [3–1023]	3-1023, **8**	When the video mode is non-burst with sync events, this is used to determine the deassertion of the hsync_o signal. When non-burst with sync pulses, this is used only by the testbench for simulation; the actual hsync_o deassertion still depends on the reception of the hsync end packet. Configurable when *RX Interface = DSI*.
DSI Sync Packet Delay [5–1023]	**5**-1023	This is the number of Pixel Clock cycle to delay the assertion of the HSYNC and VSYNC signals. Configurable when *RX Interface = DSI*.
Pixel Clock Frequency (MHz) [10–250]	10–250, **50**	Pixel Clock Frequency. The upper frequency limit varies for below devices: • Avant devices: 250 • non-Avant devices: 200
Enable AXI4-Stream Transmitter Interface	Checked, **Not Checked**	Enables AXI4-Stream Transmitter interface.
Transmitter Data Rate	Calculated	Non-configurable.
FIFO		
Manual Adjust	Checked, **Not Checked**	—
Overflow/Underflow Threshold [1–65535]	1–65535, **4**	Configurable when Manual Adjust box is checked.
FIFO Depth [8–65536]	8–65536, **16**	Configurable when Manual Adjust box is checked.
FIFO Implementation	**EBR**, LUT	—
Word Count [MIN_WC - 65535]	1–65535, **5**	Data Type Word Count value depends on the Data Type. Refer to column Byte Count Restriction for the actual value being used in the Pixel and Byte Count Restriction section.
Debug		
Enable Debug Ports	Checked, **Not Checked**	—

3.2. Supported Configurations for DSI

Table 3.2. Supported Configurations for DSI

Number of Output Pixel Lanes	D-PHY Lanes	RX Gearing	Data Type
1 output pixel	1 lane	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
		16	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
	2 lanes	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
		16	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
	4 lanes	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
2 output pixel	1 lane	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565
		16	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565
	2 lanes	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565
		16	YCbCr422 – 16 bit, YCbCr422 – 24 bit,

Number of Output Pixel Lanes	D-PHY Lanes	RX Gearing	Data Type
			YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
	4 lanes	8	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
		16	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888
4 output pixels	4 lanes	16	YCbCr422 – 16 bit, YCbCr422 – 24 bit, YCbCr422 loosely packed – 20 bit, RGB565, RGB666, RGB666 loosely packed, RGB888

3.3. Supported Configurations for CSI-2

Table 3.3. Supported Configurations for CSI-2

Number of Output Pixel Lanes[3]	D-PHY Lanes	RX Gearing	Data Type
1 output pixel	1 lane	8	RGB565, 8-bit[1], 10-bit[2], RAW12, RAW14, RAW16, RGB888
		16	RGB565, 10-bit, RAW12, RAW14, RAW16, RGB888
	2 lanes	8	RGB565, 8-bit, 10-bit, RAW12, RAW14, RAW16, RGB888
		16	RGB565, RAW12, RAW14, RAW16, RGB888
	4 lanes	8	RGB565, 8-bit, 10-bit,

Number of Output Pixel Lanes[3]	D-PHY Lanes	RX Gearing	Data Type
			RAW12, RAW14, RAW16 RGB888
		16	RGB565, RAW12, RAW14, RAW16
2 output pixel	1 lane	8	RGB565, RAW12, RAW14, RAW16
		16	RGB565, 8-bit, RAW12, RAW14, RAW16
	2 lanes	8	RGB565, 8-bit, 10-bit, RAW12, RAW14, RAW16
		16	RGB565, 8-bit, 10-bit, RAW12, RAW14, RAW16 RGB888
	4 lanes	8	RGB565, 8-bit, 10-bit, RAW12, RAW14, RAW16 RGB888
		16	RGB565, RAW12, RAW14, RAW16 RGB888
4 output pixel	1 lane	16	RAW12
	2 lanes	8	RAW12
		16	RAW12
	4 lanes	8	RAW12
		16	RAW12 RGB888

Notes:

1. Supported 8-bit CSI-2 data types are RAW8, YUV420 8-bit, Legacy YUV420 8-bit, YUV420 8-bit CSPS, and YUV422 8-bit.
2. Supported 10-bit CSI-2 data types are RAW10, YUV420 10-bit, YUV420 10-bit CSPS, and YUV422 10-bit.
3. For YUV or YCbCr data type, the pixel data bus width refers to its actual bit per data type (YUV or YCbCr) component instead of bits per data type (YUV or YCbCr) pixel. For example, in YUV420 8-bit, selecting *Number of Output Pixel Lanes == 2* means two parallel 8-bit component is received per clk_pixel_i.

3.4. Pixel and Byte Count Restriction

Table 3.4. Pixel and Byte Count Restriction

Data Type	Pixel Count Restriction	Byte Count Restriction
RGB565	multiple of 1	multiple of 2
RGB666	multiple of 4	multiple of 9
RGB666, Loosely packed	multiple of 1	multiple of 3
RGB888	multiple of 1	multiple of 3
RAW8	multiple of 1	multiple of 1
Legacy YUV420 8-bit	multiple of 2	multiple of 3
YUV420 8-bit	multiple of 2	multiple of 4
YUV422 8-bit	multiple of 2	multiple of 4
RAW10	multiple of 4	multiple of 5
YUV420 10-bit	multiple of 4	multiple of 10
YUV422 10-bit	multiple of 2	multiple of 5
RAW12	multiple of 2	multiple of 3
YCbCr422 16-bit	multiple of 2	multiple of 4
YcbCr422 24-bit	multiple of 4	multiple of 6
YcbCr422 Loosely packed 20-bit	multiple of 4	multiple of 6
RAW14	multiple of 4	multiple of 7
RAW16	multiple of 1	multiple of 2

4. Signal Description

This section describes the Byte-to-Pixel Converter IP ports.

4.1. Clock Interface

Table 4.1. Clock Interface Signal Descriptions

Port	Type	Description
clk_byte_i	Input	• This signal is the source clock for the Byte Domain Native Interface • The recommended clock frequency for this signal is: • For Avant devices: 10 MHz – 250 MHz • For non-Avant devices: 10 MHz – 200 MHz
axis_sclk_i	Input	• This signal is the source clock for AXI4-Stream Receiver Interface • The recommended clock frequency for this signal is: • For Avant devices: 10 MHz – 250 MHz • For non-Avant devices: 10 MHz – 200 MHz
clk_pixel_i	Input	• This signal is the source clock for the Pixel Domain Interface • The recommended clock frequency for this signal is: • For Avant devices: 10 MHz – 250 MHz • For non-Avant devices: 10 MHz – 200 MHz
axis_mclk_i	Input	• This signal is the source clock for AXI4-Stream Transmitter Interface • The recommended clock frequency for this signal is: • For Avant devices: 10 MHz – 250 MHz • For non-Avant device: 10 MHz – 200 MHz

4.2. Reset Interface

Table 4.2. Reset Interface Signal Descriptions

Port	Type	Description
reset_byte_n_i	Input	• System Reset • Active low signal to reset the logic in the Byte Domain Native Interface
axis_sresetn_i	Input	• Active low signal to reset the logic in the AXI4-Stream Receiver Interface • Active when the AXI4-Stream Receiver Interface is enabled
axis_mresetn_i	Input	Active low signal to reset the logic in the Pixel Domain Interface
reset_pixel_n_i	Input	• Active low signal to reset the logic in the AXI4-Stream Transmitter Interface • Active when the AXI4-Stream Transmitter Interface is enabled

4.3. Byte Domain Interface

Table 4.3. Byte Domain Signal Descriptions

Port	Type	Description
sp_en_i	Input	Active high pulse to indicate a valid short packet in the Rx side
lp_av_en_i	Input	Active high pulse to indicate an active video long packet in the Rx side. The byte2pixel module prepares for the arrival of the video stream.
dt_i[5:0]	Input	Data type field of the D-PHY Rx header packet
wc_i[15:0]	Input	Word Count field of the D-PHY Rx header packet
payload_i[NUM_RX_LANE*RX_GEAR-1:0]	Input	This is the active video data stream. The width of the data bus depends on the gearing and the number of D-PHY Rx lanes. Refer to the IP Parameter Description section for possible values for NUM_RX_LANE (Number of RX Lanes) and RX_GEAR (RX Gear).
payload_en_i	Input	Active high payload valid indicator
sp2_en_i	Input	This is valid only for gear 16, 4-lane configuration. Active high pulse to indicate a reception of a second valid short packet in the same byte clock cycle.
lp2_av_en_i	Input	This is valid only for gear 16, 4-lane configuration. Active high pulse to indicate a second valid active video long packet in the same byte clock cycle.
dt2_i[5:0]	Input	This is valid only for gear 16, 4-lane configuration. Data type field of the second D-PHY RX header packet.
wc2_i[15:0]	Input	This is valid only for gear 16, 4-lane configuration. Word Count field of the second D-PHY RX header packet.
pixcnt_c_o[18:0]	Output	This is an internal net with critical path. It is ported out so that constraints can still be applied to this signal even on encrypted IP. This port may be left unconnected.
pix_out_cntr_o[15:0]	Output	
wc_pix_sync_o[15:0]	Output	

4.4. AXI4-Stream Receiver Interface

Table 4.4. AXI4-Stream Receiver Interface Signal Descriptions

Port	Type	Description
axis_svalid_i	Input	AXI4-Stream Receiver valid input signal
axis_sready_o	Output	AXI4-Stream Receiver ready output signal. Currently, the value of the signal is always set to 1.
axis_sdata_i[RECEIVER_DATA_W-1:0]	Input	AXI4-Stream Receiver data*

***Notes:**

- For the case when NUM_RX_LANE*RX_GEAR != 64:
  - RECEIVER_DATA_W = dt_i + wc_i + payload_i = 6 + 16 + (NUM_RX_LANE*RX_GEAR).
- For the case when NUM_RX_LANE*RX_GEAR = 64:
  - RECEIVER_DATA_W = dt_i + wc_i + dt2_i + wc2_i + payload_i = 6+16+6+16+64 = 108.

4.5. Pixel Domain Interface

Table 4.5. Pixel Domain Interface Signal Descriptions

Port Name	Direction	Description
vsync_o	Output	VSYNC signal for DSI. Active High if *Camera/Display Control Polarity* attribute is Positive. Otherwise, this is active Low.
hsync_o	Output	HSYNC signal for DSI. Active High if *Camera/Display Control Polarity* attribute is Positive. Otherwise, this is active Low.
fv_o	Output	Frame Valid signal for CSI-2. Active High if *Camera/Display Control Polarity* attribute is Positive. Otherwise, this is active low.
lv_o	Output	Line Valid signal for CSI-2. Active High if Camera/Display Control Polarity attribute is Positive. Otherwise, this is active Low.
de_o	Output	Data Enable signal for DSI. Active high if Camera/Display Control Polarity attribute is Positive E. Otherwise, this is active low.
pd_o[PD_BUS_WIDTH*NUM_TX_CH-1:0]	Output	Pixel data output. The pixel_width may be 8, 10, 12, 18, 24, 36, 48, 72, or 96 bits. Refer the IP Parameter Description section for possible values for PD_BUS_WIDTH (Data Type Width) and NUM_TX_CH (Number of Output Pixel Lanes).
p_odd_o[1:0]	Output	This signal is used to indicate the valid pixels for the last valid pixel data cycle in case of multiple pixel outputs per pixel clock cycle. This is a single bit signal for a 2-pixel output configuration, or a 2-bit bus for a 4-pixel output configuration. 00 – All pixels are valid 01 – Only the first pixel (LSB) is valid 10 – Only the lower two pixels in the lower bits are valid 11 – The last pixel (MSB) is not valid

4.6. AXI4-Stream Transmitter Interface

Table 4.6. AXI4-Stream Transmitter Interface Signal Descriptions

Port Name	Direction	Description
axis_mvalid_o	Output	AXI4-Stream Transmitter valid output signal
axis_mready_i	Input	AXI4-Stream Transmitter ready input signal
axis_mdata_o[TRANSMITTER_DATA_W-1:0]	Output	AXI4-Stream Transmitter data*

***Note:** TRANSMITTER_DATA_W = p_odd_o + pd_o = 2 + (PD_BUS_WIDTH*NUM_TX_CH)

4.7. Debug Interface

Optional interfaces serve for debug purpose only.

Table 4.7. Debug Interface

Port	Type	Description
write_cycle_o[3:0]	Output	Payload data write cycle
mem_we_o	Output	Payload data Write Enable, active high
mem_re_o	Output	Payload data Read Enable, active high
read_cycle_o[1:0]	Output	Pixel data read cycle
fifo_empty_o	Output	Indicates FIFO empty condition
fifo_full_o	Output	Indicates FIFO full condition

5. Designing with the IP

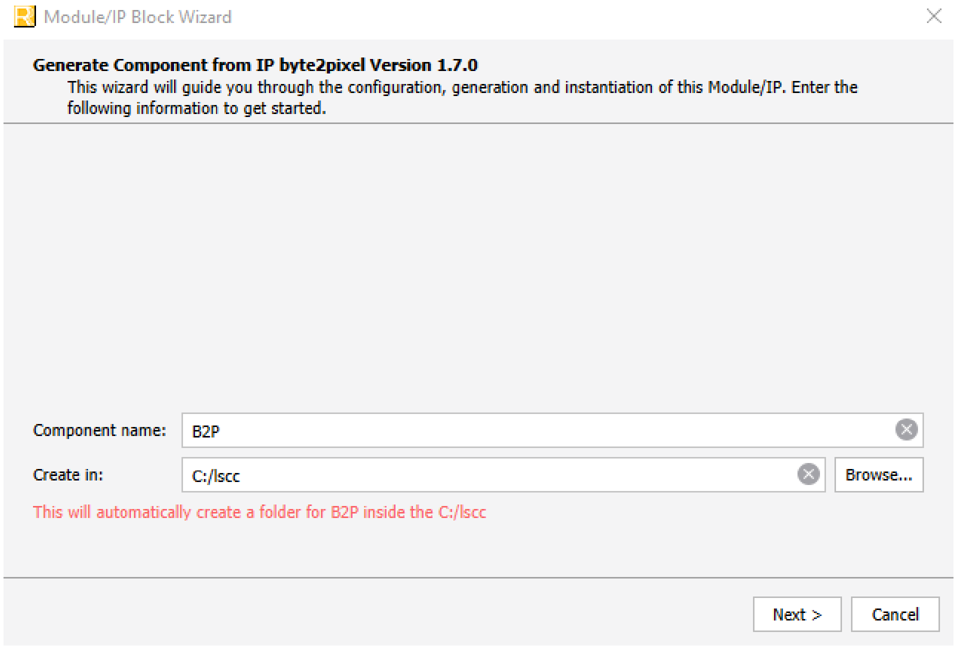
This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

5.1. 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 Byte-to-Pixel Converter IP in the Lattice Radiant software.

To generate the Byte-to-Pixel Converter IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **Byte-to-Pixel Converter** under **IP, Audio_Video_and_Image_Processing** category. The **Module/IP Block Wizard** opens as shown in Figure 5.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.



Generate Component from IP byte2pixel Version 1.7.0

This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

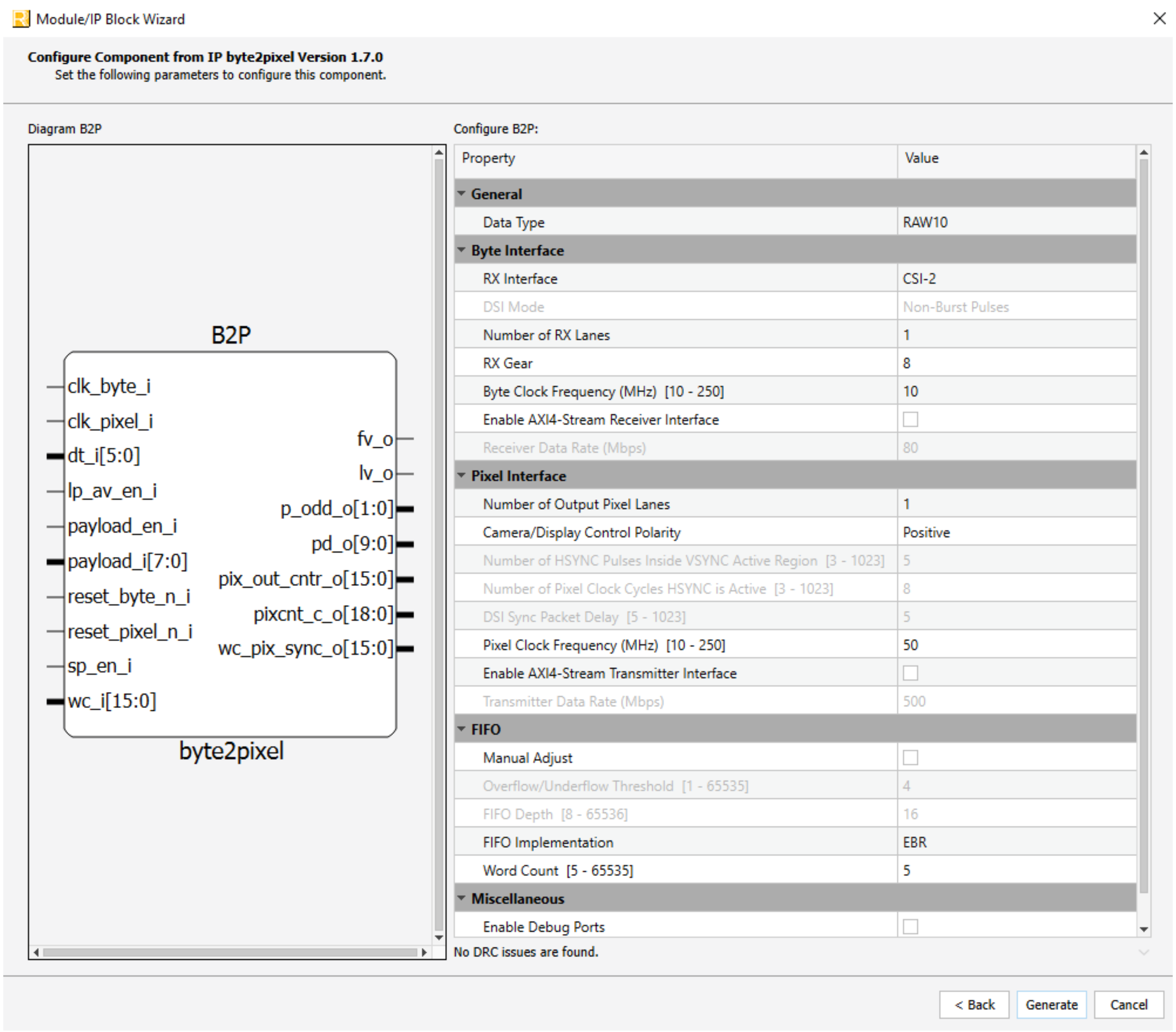
Component name: B2P

Create in: C:/lscc

This will automatically create a folder for B2P inside the C:/lscc

Figure 5.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected Byte-to-Pixel Converter IP using the drop-down lists and checkboxes. Figure 5.2 shows an example configuration of the Byte-to-Pixel Converter IP. For details on the configuration options, refer to the IP Parameter Description section.



Module/IP Block Wizard

Configure Component from IP byte2pixel Version 1.7.0
Set the following parameters to configure this component.

Diagram B2P

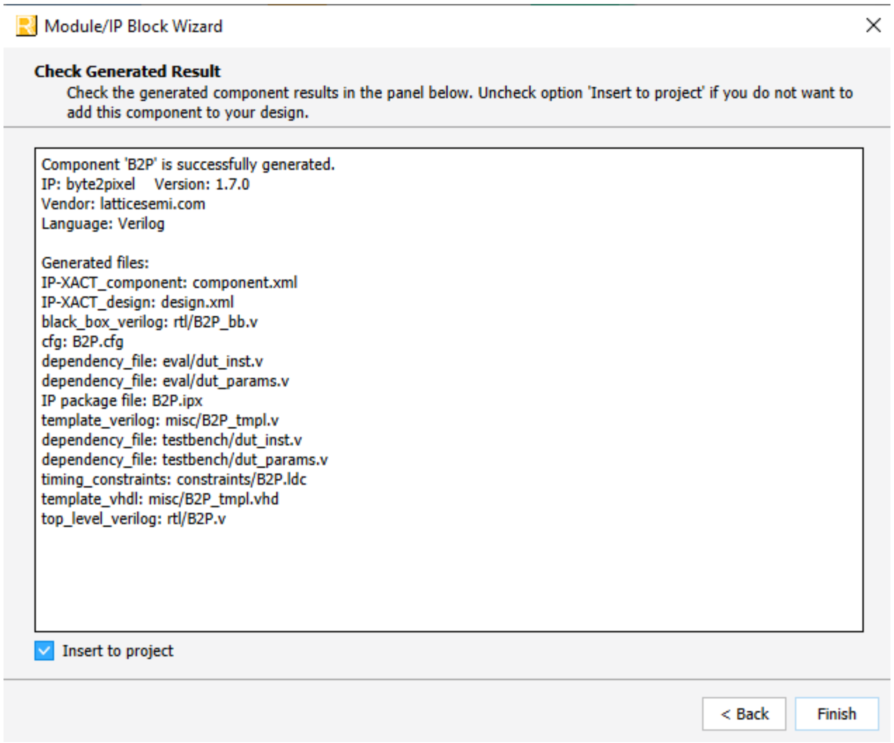
Configure B2P:

Property	Value
General	
Data Type	RAW10
Byte Interface	
RX Interface	CSI-2
DSI Mode	Non-Burst Pulses
Number of RX Lanes	1
RX Gear	8
Byte Clock Frequency (MHz) [10 - 250]	10
Enable AXI4-Stream Receiver Interface	□
Receiver Data Rate (Mbps)	80
Pixel Interface	
Number of Output Pixel Lanes	1
Camera/Display Control Polarity	Positive
Number of HSYNC Pulses Inside VSYNC Active Region [3 - 1023]	5
Number of Pixel Clock Cycles HSYNC is Active [3 - 1023]	8
DSI Sync Packet Delay [5 - 1023]	5
Pixel Clock Frequency (MHz) [10 - 250]	50
Enable AXI4-Stream Transmitter Interface	□
Transmitter Data Rate (Mbps)	500
FIFO	
Manual Adjust	□
Overflow/Underflow Threshold [1 - 65535]	4
FIFO Depth [8 - 65536]	16
FIFO Implementation	EBR
Word Count [5 - 65535]	5
Miscellaneous	
Enable Debug Ports	□

No DRC issues are found.

Figure 5.2. IP Configuration

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing the design block messages and results as shown in Figure 5.3.



Check Generated Result

Check the generated component results in the panel below. Uncheck option 'Insert to project' if you do not want to add this component to your design.

```
Component 'B2P' is successfully generated.
IP: byte2pixel    Version: 1.7.0
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/B2P_bb.v
cfg: B2P.cfg
dependency_file: eval/dut_inst.v
dependency_file: eval/dut_params.v
IP package file: B2P.ipx
template_verilog: misc/B2P_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
timing_constraints: constraints/B2P.ldc
template_vhdl: misc/B2P_tmpl.vhd
top_level_verilog: rtl/B2P.v
```

☑ Insert to project

Figure 5.3. Check 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 5.1.

5.1.1. Generated Files and File Structure

The generated Byte-to-Pixel Converter IP module package includes the black box (B2P_bb.v) and instance templates (B2P_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (B2P.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 their complete design. The generated files are listed in Table 5.1.

Note: The component name used in this example is *B2P* which is customizable based on your preference <Instance Name>.

Table 5.1. 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.
rtl/<Instance Name>_bb.v	This file provides the synthesis black box.
misc/<Instance Name>_tmpl.v misc/<Instance Name>_tmpl.vhd	These files provide instance templates for the IP core.
constraints/constraint.sdc	This file provides information on how to constrain the IP in your design.
eval/constraint.pdc	This file constrains the clock used in your design. Refer to the Timing Constraints section on how to use this file.

5.2. Design Implementation

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

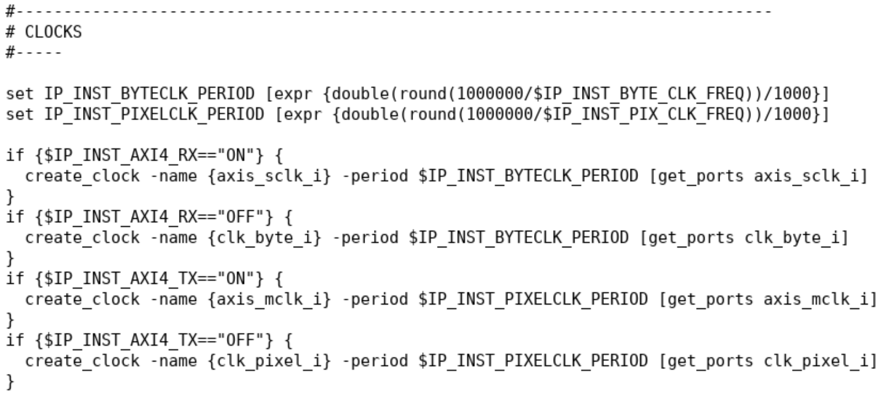
Post-Synthesis constraint files (.pdc) contain both timing and non-timing constraint.pdc source files for storing logical timing and physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file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

5.3. Timing Constraints

The Byte-to-Pixel Converter IP generates the following constraint files:

- A constraint file in SDC format (*<ip_instance_path>/constraints/constraint.sdc*) that contains post-synthesis IP constraints. These constraints are automatically used and propagated by the software tool for the Lattice Radiant software version 2024.1 and higher.
- A constraint file in PDC format (*<ip_instance_path>/eval/constraint.pdc*) that contains clock constraints. These constraints can be modified given the frequency of clocks you want to use in the design.



```
#------------------------------------------------------------------------------
# CLOCKS
#-----

set 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}]

if {$IP_INST_AXI4_RX=="ON"} {
  create_clock -name {axis_sclk_i} -period $IP_INST_BYTECLK_PERIOD [get_ports axis_sclk_i]
}
if {$IP_INST_AXI4_RX=="OFF"} {
  create_clock -name {clk_byte_i} -period $IP_INST_BYTECLK_PERIOD [get_ports clk_byte_i]
}
if {$IP_INST_AXI4_TX=="ON"} {
  create_clock -name {axis_mclk_i} -period $IP_INST_PIXELCLK_PERIOD [get_ports axis_mclk_i]
}
if {$IP_INST_AXI4_TX=="OFF"} {
  create_clock -name {clk_pixel_i} -period $IP_INST_PIXELCLK_PERIOD [get_ports clk_pixel_i]
}
```

Figure 5.4. Clock Constraining in Constraint PDC file

To run the software implementation flow using the provided constraint file after the IP is generated, follow these steps:

1. In the Post-Synthesis Constraint Files section, add *<ip_instance_name>/eval/constraint.pdc* as show in Figure 5.5.

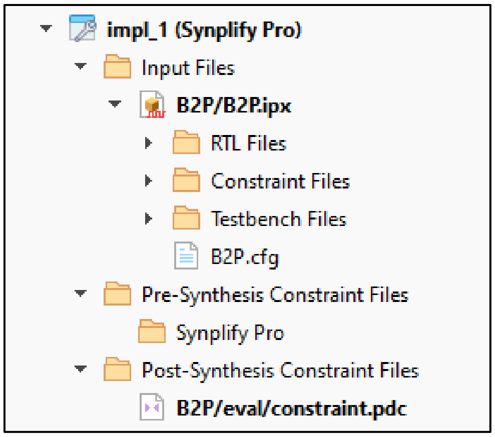


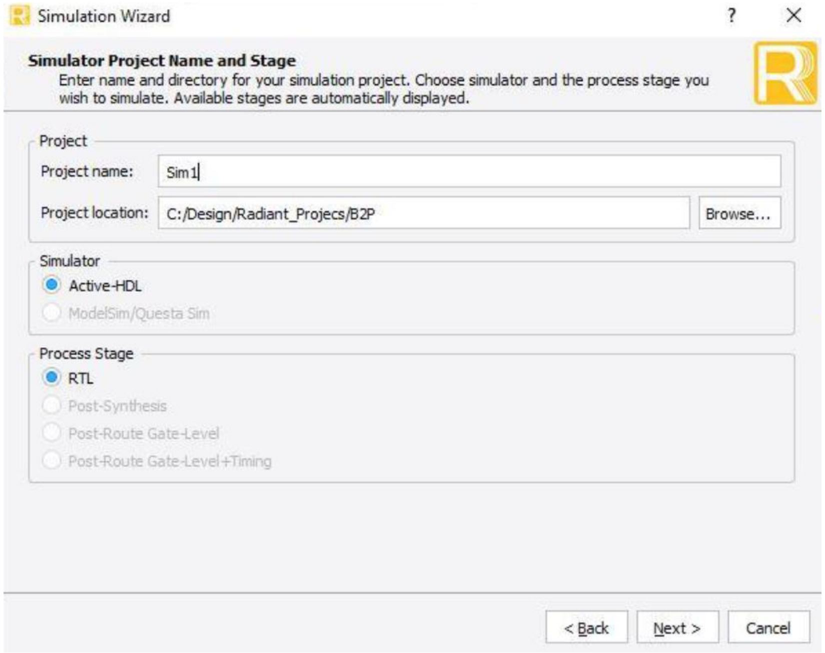
Figure 5.5. Adding Constraint in PDC file

2. Run the implementation flow.

5.4. Running Functional 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 5.6.



Simulation Wizard

Simulator Project Name and Stage
Enter name and directory for your simulation project. Choose simulator and the process stage you wish to simulate. Available stages are automatically displayed.

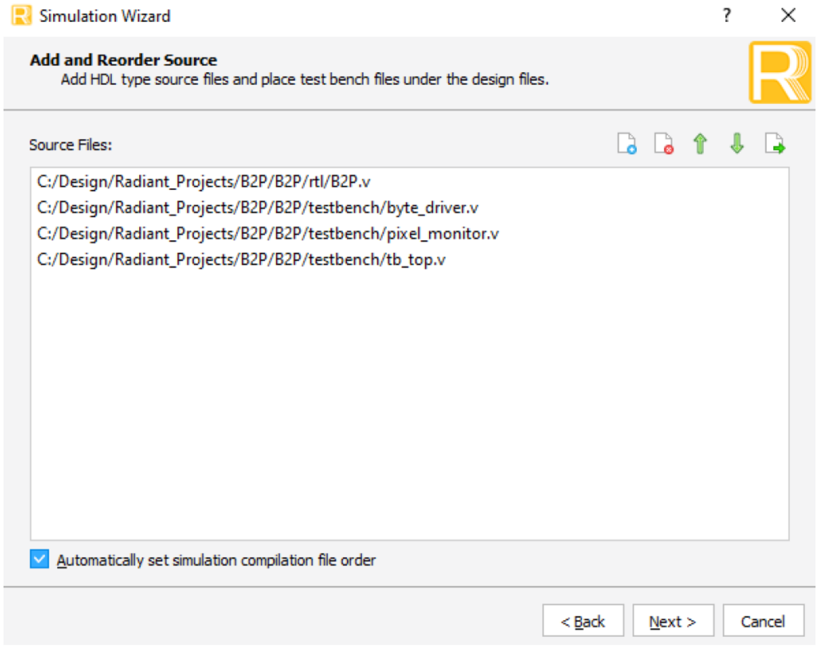
Project
Project name: Sim1
Project location: C:/Design/Radiant_Projecs/B2P

Simulator
● Active-HDL
○ ModelSim/Questa Sim

Process Stage
● RTL
○ Post-Synthesis
○ Post-Route Gate-Level
○ Post-Route Gate-Level+Timing

Figure 5.6. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 5.7.



Simulation Wizard

Add and Reorder Source
Add HDL type source files and place test bench files under the design files.

Source Files:

C:/Design/Radiant_Projects/B2P/B2P/rtl/B2P.v
C:/Design/Radiant_Projects/B2P/B2P/testbench/byte_driver.v
C:/Design/Radiant_Projects/B2P/B2P/testbench/pixel_monitor.v
C:/Design/Radiant_Projects/B2P/B2P/testbench/tb_top.v

☑ Automatically set simulation compilation file order

Figure 5.7. Add and Reorder Source

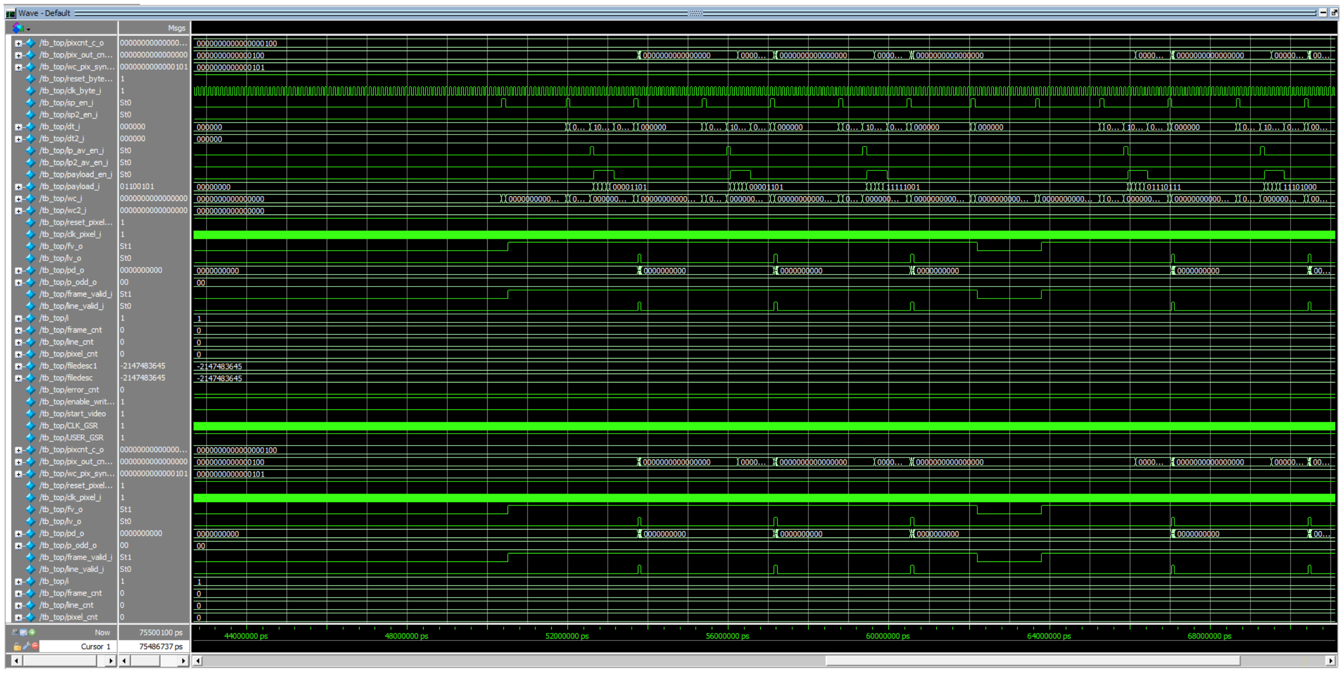
3. Add the tb_top.v file from the testbench directory.

Table 5.2. Testbench File List

Testbench Files	Description
testbench/tb_top.v	Top testbench to run loopback test of generated <Instance Name>.v
testbench/byte_driver.v	Testbench to create log files for monitoring byte data during transmission.
testbench/pixel_monitor.v	Testbench to create log files for monitoring pixel data during transmission.

4. Click **Next**. The **Summary** window is shown.
5. Click **Finish** to run the simulation.

The waveform in Figure 5.8 shows an example simulation result.

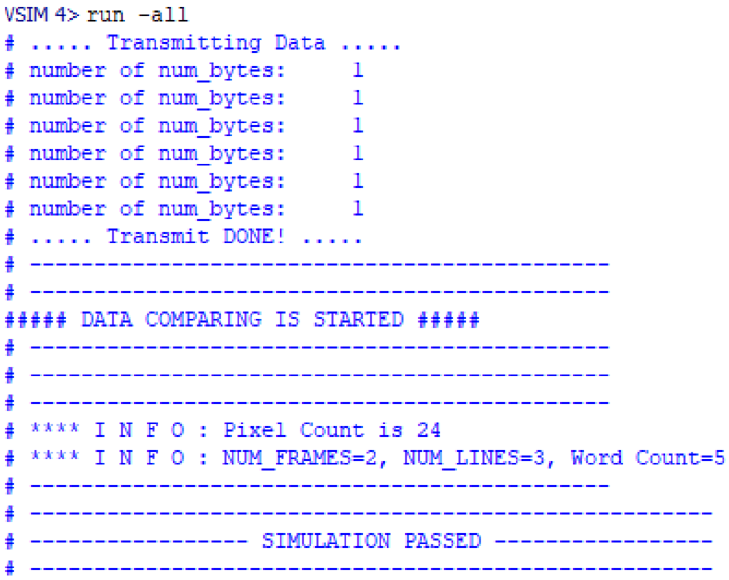


	Msgs
/tb_top/pixcnt_c_o	00000000000000...
/tb_top/pix_out_cn...	0000000000000000
/tb_top/wc_pix_syn...	0000000000000101
/tb_top/reset_byte...	1
/tb_top/clk_byte_i	1
/tb_top/sp_en_i	St0
/tb_top/sp2_en_i	St0
/tb_top/dt_i	000000
/tb_top/dt2_i	000000
/tb_top/lp_av_en_i	St0
/tb_top/lp2_av_en_i	St0
/tb_top/payload_en_i	St0
/tb_top/payload_i	01100101
/tb_top/wc_i	0000000000000000
/tb_top/wc2_i	0000000000000000
/tb_top/reset_pixel...	1
/tb_top/clk_pixel_i	1
/tb_top/fv_o	St1
/tb_top/lv_o	St0
/tb_top/pd_o	0000000000
/tb_top/p_odd_o	00
/tb_top/frame_valid_i	St1
/tb_top/line_valid_i	St0
/tb_top/i	1
/tb_top/frame_cnt	0
/tb_top/line_cnt	0
/tb_top/pixel_cnt	0
/tb_top/filedesc1	-2147483645
/tb_top/filedesc	-2147483645
/tb_top/error_cnt	0
/tb_top/enable_writ...	1
/tb_top/start_video	1
/tb_top/CLK_GSR	1
/tb_top/USER_GSR	1
/tb_top/pixcnt_c_o	00000000000000...
/tb_top/pix_out_cn...	0000000000000000
/tb_top/wc_pix_syn...	0000000000000101
/tb_top/reset_pixel...	1
/tb_top/clk_pixel_i	1
/tb_top/fv_o	St1
/tb_top/lv_o	St0
/tb_top/pd_o	0000000000
/tb_top/p_odd_o	00
/tb_top/frame_valid_i	St1
/tb_top/line_valid_i	St0
/tb_top/i	1
/tb_top/frame_cnt	0
/tb_top/line_cnt	0
/tb_top/pixel_cnt	0
Now	75500100 ps
Cursor 1	75486737 ps

Figure 5.8. Simulation Waveform

5.4.1. Simulation Results

Simulating the IP, the expected result is shown in Figure 5.9.



```
VSIM 4> run -all
# ..... Transmitting Data .....
# number of num_bytes:     1
# number of num_bytes:     1
# number of num_bytes:     1
# number of num_bytes:     1
# number of num_bytes:     1
# number of num_bytes:     1
# ..... Transmit DONE! .....
# ---------------------------------------------
# ---------------------------------------------
##### DATA COMPARING IS STARTED #####
# ---------------------------------------------
# ---------------------------------------------
# ---------------------------------------------
# **** I N F O : Pixel Count is 24
# **** I N F O : NUM_FRAMES=2, NUM_LINES=3, Word Count=5
# ---------------------------------------------
# -----------------------------------------------------
# ----------------- SIMULATION PASSED -----------------
# -----------------------------------------------------
```

Figure 5.9. Simulation Result

6. Debugging

6.1. Debug Interface Ports

Optional interface for debugging purposes only.

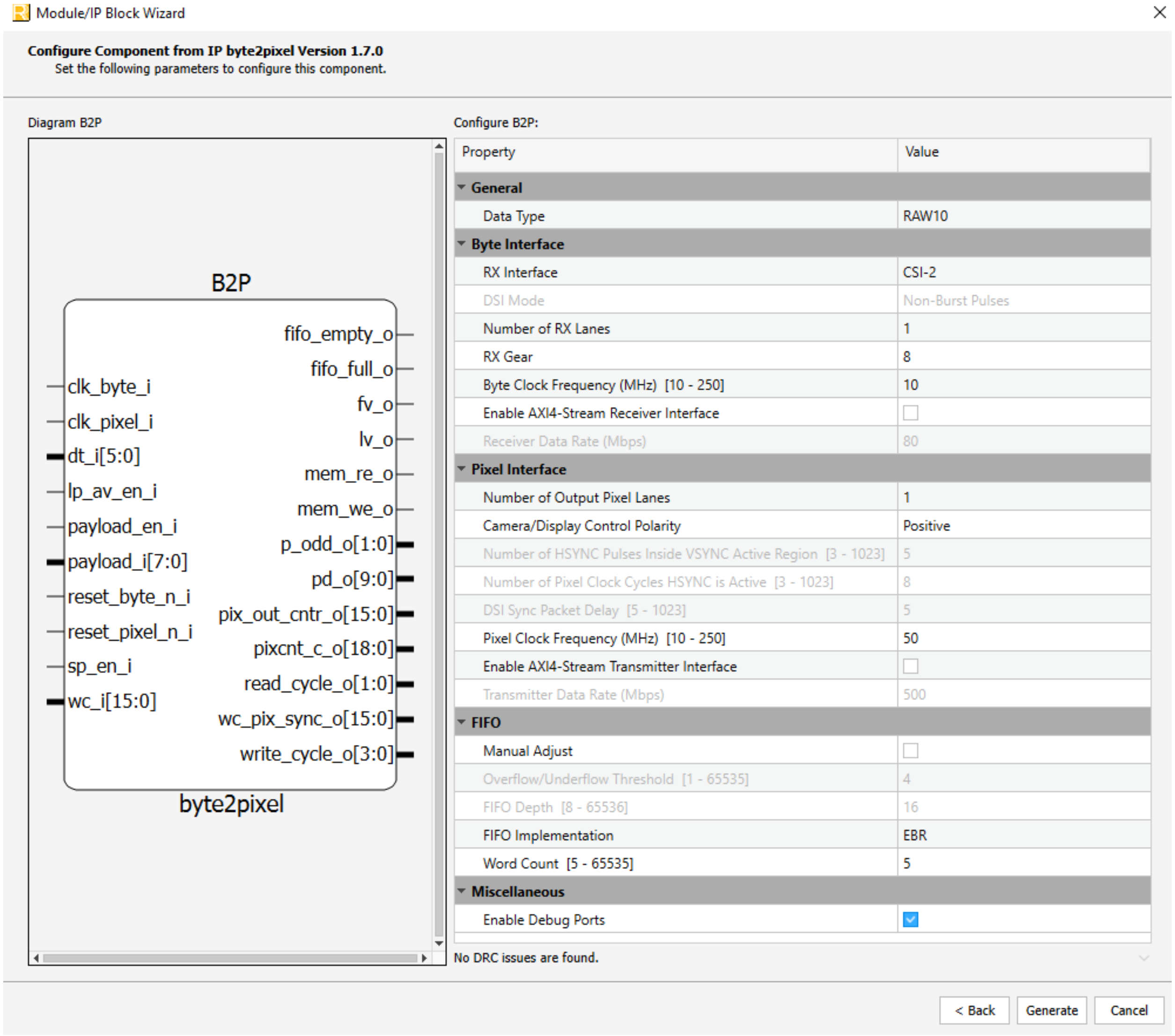
Table 6.1. Debug Interface Ports

Port	Type	Description
write_cycle_o[3:0][1][2]*	Output	Data write cycle to the FIFO
mem_we_o	Output	Active high data Write Enable to the FIFO
read_cycle_o[1:0][1][3]*	Output	Data read cycle to the FIFO
mem_re_o	Output	Active high data Read Enable to the FIFO
fifo_empty_o	Output	Indicates FIFO empty condition
fifo_full_o	Output	Indicates FIFO full condition

Notes:

1. These signals are different per data type.
2. *write_cycle_o* signal is not continuous because the data is accumulated before writing to the FIFO.
3. *read_cycle_o* signal is not continuous because the logic will remap the data first after reading from the FIFO.

6.2. Debugging Using Debug Ports (Sample Configuration)



Module/IP Block Wizard

Configure Component from IP byte2pixel Version 1.7.0
Set the following parameters to configure this component.

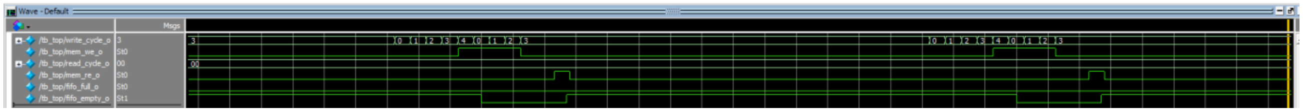
Diagram B2P

Configure B2P:

Property	Value
General	
Data Type	RAW10
Byte Interface	
RX Interface	CSI-2
DSI Mode	Non-Burst Pulses
Number of RX Lanes	1
RX Gear	8
Byte Clock Frequency (MHz) [10 - 250]	10
Enable AXI4-Stream Receiver Interface	□
Receiver Data Rate (Mbps)	80
Pixel Interface	
Number of Output Pixel Lanes	1
Camera/Display Control Polarity	Positive
Number of HSYNC Pulses Inside VSYNC Active Region [3 - 1023]	5
Number of Pixel Clock Cycles HSYNC is Active [3 - 1023]	8
DSI Sync Packet Delay [5 - 1023]	5
Pixel Clock Frequency (MHz) [10 - 250]	50
Enable AXI4-Stream Transmitter Interface	□
Transmitter Data Rate (Mbps)	500
FIFO	
Manual Adjust	□
Overflow/Underflow Threshold [1 - 65535]	4
FIFO Depth [8 - 65536]	16
FIFO Implementation	EBR
Word Count [5 - 65535]	5
Miscellaneous	
Enable Debug Ports	☑

No DRC issues are found.

Figure 6.1. Sample Configuration with Enabled Debug Ports



Wave - Default

	Msgs
/tb_top/write_cycle_o	3
/tb_top/mem_we_o	St0
/tb_top/read_cycle_o	00
/tb_top/mem_re_o	St0
/tb_top/fifo_full_o	St0
/tb_top/fifo_empty_o	St1

Figure 6.2. Debug Ports Simulation using the Sample Configuration

***Disclaimer:** The debug signal value is needed for customer support because debug ports are internal in the design.

7. Design Considerations

7.1. Design considerations of Byte-to-Pixel Converter as a standalone IP and/or connected to different Video IPs

- Verify the supported MIPI CSI-2-compatible video formats.
- Verify the supported MIPI DSI-compatible video formats.
- Verify the AXI4-Stream write and read transactions.

7.2. Limitations

- AXI4-Stream interface does not support back-pressure.
- The Byte-to-Pixel Converter IP does not support small horizontal blanking when *WORD_CNT* is maximum. (Note: Maximum *WORD_CNT* values differ per *DATA_TYPE.* Please refer to Table 3.4 for Pixel and Byte count restriction per Data Types.).

Appendix A. Resource Utilization

Table A.1 and Table A.2 show the maximum frequency and resource utilization for a certain IP configuration.

Table A.1. Device and Tool Tested

Test Parameter	Value
Software Version	Lattice Radiant software 2024.1 production build
Device Used	LFCPNX-100-7LFG672C
Performance Grade	7_High-Performance_1.0V
Synthesis Tool	Synplify Pro, June 2024

Table A.2. Resource Utilization using LFCPNX-100-7LFG672C Device

Configuration	Pixel/Byte Interface	FMax (MHz)	LUTs	Registers	sysMem EBRs	Programmable I/O
Default	Native	183.993	466	265	1	102
RGB888, Word Count=720, Others=Default	AXI4-Stream	186.951	418	288	1	117
RGB888, Number of RX Lanes=4, Word Count=2052, Others=Default	Native	193.498	428	367	1	140
RGB888, Number of RX Lanes=4, Word Count=3600, Others=Default	AXI4-Stream	166.639	467	366	1	141
DSI,RGB666, Word Count=2160, Others=Default	Native	184.196	421	313	1	111
DSI,RGB666, Number of RX Lanes=2, Word Count=2160, Others=Default	AXI4-Stream	200	521	331	1	120

Note: The *distributed RAM* utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distribution among *logic*, *distributed RAM*, and *ripple logic*.

References

For more information, refer to:

- Lattice Radiant Software 2023.1 User Guide
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Lattice Radiant Software web page
- Byte to Pixel 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

Revision 1.9, Lattice Radiant SW version 2024.1, 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: • *Embedded Block RAM (EBR)* • *Input/Output (I/O)* • *Look-Up Table (LUT)* • *Random Access Memory (RAM)*
Introduction	• In Table 1.1. Summary of the Byte-to-Pixel Converter IP: • Updated the description for *IP Core v.1.6.x*. • Added *IP Core v.1.7.x*. • Added *Native Interface*. • Added the *AXI4-stream transmitter and receiver interface* in the Features section. • Updated Table 1.2. Ordering Part Number and Table 1.3. IP Validation Level. • Removed *_io* from the list in the Signal Names section.
Functional Description	• In the IP Architecture Overview section: • Updated the layer descriptions of *Byte-to-Pixel IP* functional block. • Removed the previous section header *2.1.5. Byte-to-Pixel Converter IP Interface Configurations*. • Updated all figures and their captions. • Added *clk_pixel_i* and *clk_byte_i* to *Note* in the Reset section. • Moved the following previous sections into the User Interfaces and Other IP Specific Blocks/Layers/Interfaces sections: • *2.1.1. Byte-to-Pixel Converter* • *2.1.2. AXI4 Stream Receiver* • *2.1.3. AXI4 Stream Transmitter* • *2.1.4. FIFO Implementation* • In Table 2.2. User Interfaces and Supported Protocols: • Removed *AXI-4 Stream Receiver/Transmitter Interface*. • Added *AXI-4 Stream Receiver Interface*. • Added *AXI-4 Stream Transmitter Interface*. • Renamed and changed the header number of the previous *2.5. Timing Specifications* section to the *2.4.1. Native Interfaces* section. • Removed the following previous section headers: • *2.5.1. Input Timing* • *2.5.2. Output Timing* • Added the Debug Interface section.
IP Parameter Description	• Updated Table 3.1. General Attributes. • In Table 3.3. Supported Configurations for CSI-2: • Removed *Data Type RGB888* from *RX Gearing 16 – 2 lanes – 1 output pixel*. • Added *Note 3*.
Signal Description	• Updated the descriptions for all ports in Table 4.1. Clock Interface Signal Descriptions and Table 4.2. Reset Interface Signal Descriptions. • Added *Signal Descriptions* to all table captions in this section except for Table 4.7. Debug Interface. • Moved *AXI4-Stream Receiver Interface* information from the Byte Domain Interface section to the newly added AXI4-Stream Receiver Interface section. • Moved *AXI4-Stream Transmitter Interface* information from the Pixel Domain Interface section to the newly added AXI4-Stream Transmitter Interface section.

Section	Change Summary
Designing with the IP	• Updated the following figures: • Figure 5.1. Module/IP Block Wizard • Figure 5.2. IP Configuration • Figure 5.3. Check Generated Result • Figure 5.4. Clock Constraining in Constraint PDC file • Added the following attributes to Table 5.1. Generated File List: • *constraints/constraint.sdc* • *eval/constraint.pdc* • Updated the Design Implementation and Timing Constraints sections.
Design Considerations	• Updated Figure 6.1. Sample Configuration with Enabled Debug Ports. • Added the Limitations section.
Resource Utilization	Updated this section.
References	Added the following references: • *Certus-NX web page* • *CertusPro-NX web page* • *CrossLink-NX web page* • *MachXO5-NX web page* • *Avant-E web page* • *Avant-G web page* • *Avant-X web page* • *Lattice Solutions IP Cores web page*

Revision 1.8, Lattice Radiant SW version 2023.2, December 2023

Section	Change Summary
All	• Updated the document title from Byte-to-Pixel Converter IP Core – Lattice Radiant Software to Byte-to-Pixel 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 1.3 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 Clocking, Reset, and User Interface subsections. • Reworked old Subsection 2.4 – Modules Description and renamed to Subsection 2.1 IP Architecture.
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.
Debugging	Added this section.
Design Considerations	Added this section.
Appendix A. Resource Utilization	Reworked section contents.
Appendix B. Limitations	Removed this section.
References	Reworked section contents.

Revision 1.7, Lattice Radiant SW version 2023.1, April 2023

Section	Change Summary
All	Updated for inclusive language.
Inclusive Language	Added this section.
Introduction	• In Table 1.1. Quick Facts: • Added MachXO5-NX in the Supported FPGA Family field. • Added LFMXO5-25 and LIFCL-33 in the Targeted Devices field.
Functional Description	• Updated Table 2.1. Byte-to-Pixel IP Ports. • Updated Table 2.2. Attributes Table.
IP Generation, Simulation, and Validation:	Updated Figure 3.1. Configure Block of Byte-to-Pixel Converter.
Ordering Part Number	Added part numbers for MachXO5-NX.
Appendix A. Resource Utilization	• Added Table A.3. Lattice Avant Device and Tool Tested. • Added Table A.4. Resource Utilization using Lattice Avant.
Technical Support Assistance	Added reference link to the Lattice Answer Database.

Revision 1.6, Lattice Radiant SW version 2022.1, November 2022

Section	Change Summary
Introduction	• In Table 1.1. Quick Facts: • Added Lattice Avant in the Supported FPGA Family field. • Added LAV-AT-500E in the Targeted Devices field.
IP Generation, Simulation, and Validation:	• Revised the title from 'IP Generation and Evaluation' to 'IP Generation, Simulation, and Validation'. • Deleted the section 'Licensing the IP'. • Revised the title of section 3.1 from 'Generation and Synthesis' to 'Generating the IP'. • Added Constraining the IP section. • Added IP Evaluation section.
Ordering Part Number	Added part numbers for Lattice Avant-E

Revision 1.5, Lattice Radiant SW version 3.1, November 2021

Section	Change Summary
Functional Description	• Added DSI Sync Packet Delay to Table 2.2. Attributes Table. • Updated the descriptions of timing diagrams in the Timing Specifications section.

Revision 1.4, Lattice Radiant SW version 3.0, June 2021

Section	Change Summary
Introduction	Updated Table 1.1 to include CertusPro-NX support.
Ordering Part Number	Added part number for CertusPro-NX.

Revision 1.3, Lattice Radiant SW version 2.1, November 2020

Section	Change Summary
Introduction	Updated reference to the Lattice Radiant Software User Guide.
Functional Description	• Added ports to Table 2.1. Byte-to-Pixel IP Ports. • Added RAW14 and RAW16 selectable values to Table 2.2. Attributes Table. • Added RAW12, RAW14, and RAW16 data types to Table 2.4. Supported Configuration for CSI-2. • Added RAW14 and RAW16 data types to Table 2.5. Pixel and Byte Count Restriction. • Updated FIFO Implementation section.

Section	Change Summary
Core Generation, Simulation, and Validation	Updated reference to the Lattice Radiant Software User Guide
References	Updated reference to the Lattice Radiant Software User Guide

Revision 1.2, Lattice Radiant SW version 2.0, August 2020

Section	Change Summary
Acronyms in This Document	Updated content.
Introduction	• Updated Table 1.1. • Updated Features section.
Functional Description	• Updated Table 2.1, Table 2.2, Table 2.3, Table 2.4, and Table 2.5. • Updated FIFO Implementation section.
IP Generation and Evaluation	• Updated Figure 3.1 and Figure 3.4. • Added Required Post-Synthesis Constraints section.
Appendix A. Resource Utilization	Updated section content including Table A.1 and Table A.2.
Appendix B. Limitations	Added this section.

Revision 1.1, Lattice Radiant SW version 2.0, February 2020

Section	Change Summary
Introduction	Updated Table 1.1 to add LIFCL-17 as targeted device.
Functional Description	• Updated descriptions for p_odd_o[1:0], axis_mdata_o[MASTER_DATA_W-1:0], and for axis_sdata_i[SLAVE_DATA_W-1:0] in Table 2.1. Byte-to-Pixel IP Ports. • Updated Transmitter and FIFO attributes in Table 2.2. Attributes Table. • Changed caption to Figure 2.6. Byte-to-Pixel IP FIFO Diagram. • Updated formulas in FIFO Implementation section.

Revision 1.0, Lattice Radiant SW version 2.0, November 2019

Section	Change Summary
All	Changed document status from Preliminary to final.
Introduction	65536axUpdated Table 1.1. Quick Facts.
Functional Description	Updated Receiver and Transmitter attributes in Table 2.2. Attributes Table.
Ordering Part Number	Added this section.
Appendix A. Resource Utilization	Added this section.

Revision 0.80, Lattice Radiant SW version 2.0, October 2019

Section	Change Summary
All	Preliminary release

LATTICE SEMICONDUCTOR®

www.latticesemi.com