



Pixel-to-Byte Converter IP Core - Lattice Radiant Software

User Guide

FPGA-IPUG-02094-1.2

August 2020

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

Acronyms in This Document	5
1. Introduction	6
1.1. Quick Facts	6
1.2. Features	7
1.2.1. Supported	7
1.2.2. Not Supported	7
1.3. Conventions	7
1.3.1. Nomenclature	7
1.3.2. Data Ordering and Data Types	7
1.3.3. Signal Names	7
2. Functional Description	8
2.1. Overview	8
2.2. Signal Description	9
2.3. Attributes Summary	12
2.4. Modules Description	14
2.4.1. Clock, Reset and Initialization	14
2.4.2. Pixel-to-Byte Wrapper Module	15
2.5. Timing Specifications	15
3. IP Generation and Evaluation	20
3.1. Licensing the IP	20
3.2. Generation and Synthesis	20
3.3. Functional Simulation	21
3.3.1. Required Post-Synthesis Constraints	22
4. Ordering Part Number	24
Appendix A. Resource Utilization	25
Appendix B. Limitations	26
References	27
Technical Support Assistance	28
Revision History	29

Figures

Figure 1.1. Sample Parallel Interface to MIPI DSI System Diagram	6
Figure 2.1. Pixel-to-Byte IP Functional Diagram	8
Figure 2.2. Clock Domain Crossing Block Diagram.....	14
Figure 2.3. Display Parallel Input Interface Timing Diagram (DSI)	15
Figure 2.4. Camera Sensor Parallel Input Interface Timing Diagram (CSI-2)	15
Figure 2.5. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 1 Pixel per Pixel Clock)	16
Figure 2.6. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 1 Pixel per Pixel Clock)	16
Figure 2.7. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 1 Pixel per Pixel Clock).....	16
Figure 2.8. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 1 Pixel per Pixel Clock).....	17
Figure 2.9. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 2 Pixels per Pixel Clock).....	17
Figure 2.10. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 2 Pixels per Pixel Clock).....	17
Figure 2.11. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 2 Pixels per Pixel Clock)	18
Figure 2.12. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 2 Pixels per Pixel Clock)	18
Figure 2.13. MIPI D-PHY High Speed Transmission Timing Diagram	18
Figure 2.14. Handshake Signals Timing Diagram	19
Figure 3.1. Configure Block of Pixel-to-Byte Converter	20
Figure 3.2. Synthesizing the Design	21
Figure 3.3. Simulation Wizard.....	21
Figure 3.4. Adding and Reordering Source	22
Figure 3.5. Adding Constraint	23

Tables

Table 1.1. Quick Facts	6
Table 2.1. Pixel-to-Byte Converter IP Ports.....	9
Table 2.2. Supported Configurations	12
Table 2.3. Attributes Table	13
Table 2.4. Clock Domain Crossing.....	14
Table 2.5. Output Byte Data Bus Width Allocation.....	15
Table A.1. Device and Tool Tested.....	25
Table A.2. Resource Utilization	25

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CSI	Camera Serial Interface
DSI	Display Serial Interface
FPGA	Field-Programmable Gate Array
LSE	Lattice Synthesis Engine
LVDS	Low-voltage Differential Signaling

1. Introduction

The Lattice Semiconductor Pixel-to-Byte Converter IP converts a standard parallel video interface to DSI or CSI-2 data for Lattice Semiconductor CrossLink™-NX and Certus™-NX FPGA family devices.

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.

This document is for Pixel-to-Byte Converter IP design version 1.1.



Figure 1.1. Sample Parallel Interface to MIPI DSI System Diagram

1.1. Quick Facts

Table 1.1 presents a summary of the IP Core.

Table 1.1. Quick Facts

IP Requirements	Supported FPGA Family	CrossLink-NX, Certus™-NX
Resource Utilization	Targeted Device	LIFCL-40, LIFCL-17, LFD2NX-40
	Resources	See Table A.2 .
Design Tool Support	Lattice Implementation	IP Core v1.0.x – Lattice Radiant® software 2.0 IP Core v1.1.x – Lattice Radiant software 2.1
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys® Synplify Pro® for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software 2.1 User Guide .

1.2. Features

1.2.1. Supported

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

- Support for RGB888, RGB666, RAW8, RAW10, RAW12, YUV420/YUV422 8/10-bit video formats
- Conversion of 1, 2, 4, 6, 8, or 10 pixels per pixel clock into MIPI D-DPHY byte packet compatible format
- Support for byte arrangement for 1, 2, or 4 MIPI D-PHY data lanes

1.2.2. Not Supported

- The IP does not support configuration through registers.
- The IP does not crop pixel data. It assumes all the data in the pixel lanes are valid while the input signal `de_i` is asserted. Hence, all are converted to byte data.

1.3. Conventions

1.3.1. Nomenclature

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

1.3.2. Data Ordering and Data Types

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

1.3.3. Signal Names

Signal Names that end with:

- `_n` are active low
- `_i` are input signals
- `_o` are output signals
- `_io` are bi-directional input/output signals

2. Functional Description

2.1. 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, it also consists of vertical and horizontal sync flags, and a data enable. For CSI-2, it also consists of a frame and line valid flags.

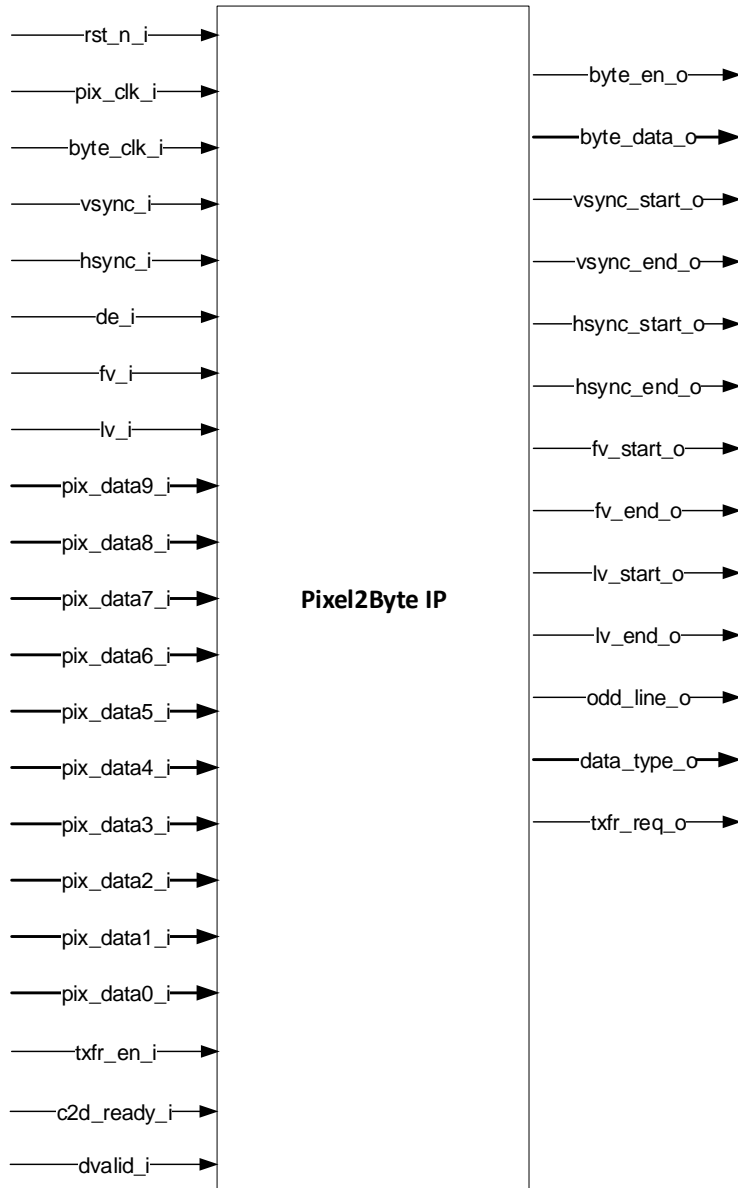


Figure 2.1. Pixel-to-Byte IP Functional Diagram

2.2. Signal Description

Table 2.1 lists the top Input and Output signals and their descriptions for the Pixel-to-Pixel IP.

Table 2.1. Pixel-to-Byte Converter IP Ports

Port Name	Direction	Description
Clocks and Reset		
rst_n_i	I	Asynchronous active low system reset 0 – System on reset
pix_clk_i	I	Input pixel clock.
byte_clk_i	I	Input byte clock.
Pixel Domain Inputs		
pix_data0_i	I	Input pixel data 0 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data1_i ¹	I	Input pixel data 1 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data2_i ¹	I	Input pixel data 2 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data3_i ¹	I	Input pixel data 3 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data4_i ¹	I	Input pixel data 4 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit

Port Name	Direction	Description
pix_data5_i ¹	I	Input pixel data 5 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data6_i ¹	I	Input pixel data 6 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data7_i ¹	I	Input pixel data 7 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data8_i ¹	I	Input pixel data 8 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
pix_data9_i ¹	I	Input pixel data 9 Bus width depends on the data type selected 24-bit bus width - RGB888 18-bit bus width - RGB666 12-bit bus width - Raw12 10-bit bus width - Raw10, YUV420/422 10-bit 8-bit bus width - Raw8, YUV420/422 8-bit
de_i ²	I	Input data enable for parallel interface
hsync_i ²	I	Input horizontal sync for parallel interface
vsync_i ²	I	Input vertical sync for parallel interface
fv_i ³	I	Input frame valid for parallel interface
lv_i ³	I	Input line valid sync for parallel interface
dvalid_i	I	Input data enable for parallel interface
Byte Domain Outputs		
vsync_start_o ²	O	Pulse signal used to indicate Vsync start
vsync_end_o ²	O	Pulse signal used to indicate Vsync end
hsync_start_o ²	O	Pulse signal used to indicate Hsync start
hsync_end_o ²	O	Pulse signal used to indicate Hsync end
fv_start_o ³	O	Pulse signal used to indicate frame start
fv_end_o ³	O	Pulse signal used to indicate frame end
lv_start_o ³	O	Pulse signal used to indicate line start
lv_end_o ³	O	Pulse signal used to indicate line end
byte_en_o	O	Indicates valid output byte data

Port Name	Direction	Description
byte_data_o	O	Output data width is (number of Tx Lanes * Tx gear)
Miscellaneous		
c2d_ready_i	I	Ready signal for transmit request
odd_line_o ^{4,5}	O	Indicates if current line is odd or even; used for YUV420 data type. <ul style="list-style-type: none"> • Output value for even line • Output value for odd line
data_type_o ⁵	O	6-bit output that indicates the data type based from MIPI DSI and CSI2 Specifications
txfr_en_i ^{6,7}	I	Enable flag from outside the IP to indicate that byte data can already be sent out from pixel2byte. 0 - do not send output byte data yet
txfr_req_o ^{6,7}	O	When asserted, it indicates that valid data (HSYNC, VSYNC, DE) is received and IP is ready to process the data.

Notes:

1. Available only when the number of input pixels data is more than 1.
2. Available only if data interface is DSI.
3. Available only if data interface is CSI-2.
4. Available only for YUV420 data type.
5. Can be turned-on if *Enable miscellaneous status signals* attribute is selected.
6. Turned on if *Enable handshake signals* attribute is selected. These signals are mandatory and are always available.
7. See the [Timing Specifications](#) section for details on their functionality.

2.3. Attributes Summary

Table 2.2 lists the supported configurations.

Table 2.2. Supported Configurations

Number of Input Pixel Lanes	Interface	Data Type	Tx Lane	Tx Gear
1 Lane	DSI	RGB666	1	8, 16
			2	8
			4	8
		RGB888	1	8, 16
			2	8, 16
			4	8
	CSI-2	RAW10 YUV420 10-bit YUV422 10-bit	1	8, 16
			2	8
			4	8
		RAW12	1	8, 16
			2	8
			4	8
		RAW 8 YUV420 8-bit YUV422 8-bit	1	8
			2	8
			4	8
RGB888	1	8, 16		
	2	8, 16		
	4	8		
2 Lanes	DSI	RGB666	2	16
			4	8, 16
		RGB888	4	8, 16
	CSI-2	RAW10 YUV420 10-bit YUV422 10-bit	2	16
			4	8
		RAW12	2	16
			4	8
		RAW 8 YUV420 8-bit YUV422 8-bit	1	16
			2	16
			4	8
RGB888	4	1		
4 Lanes	DSI	RGB666	4	16
		RGB888	4	16
	CSI-2	RAW10	4	8
		RAW12	4	8
	6 Lanes	CSI-2	RAW10	4
RAW12			4	8
8 Lanes	CSI-2	RAW10	4	16
		RAW12	4	16
10 Lanes	CSI-2	RAW10	4	16
		RAW12	4	16

Table 2.3 provides the list of user-selectable and compile time configurable attributes for the Pixel-to-Byte Converter IP. The attributes are specified using Pixel-to-Byte Converter IP Configuration user interface in Lattice Radiant.

Table 2.3. Attributes Table

User Interface Config. Category	Attribute	Options	Default	Dependency On Other Attributes	Description
Transmitter	Tx Interface	DSI, CSI2	DSI	None	Set the Tx interface.
	Data Type	RGB888, RGB666, RAW8, RAW10, RAW12, YUV420_8 YUV420_10 YUV422_8 YUV422_10	RGB888	None	Specify the data type depending on the Data Format selected
	Number of Tx Lanes	1, 2, 4	1	None	Set the target number of MIPI D-PHY Tx lanes.
	Number of Input Pixel Lanes	1,2,4,6,8,10	1	Tx Interface, Data Type, Number of Tx Lanes	Specify the number of input pixel lanes.
	Tx Gear	8, 16	8	Tx Interface, Data Type, Number of Tx Lanes, Number of Tx Lanes, number of Input Pixel Per Clock	Specify the target Tx gearing.
Miscellaneous	Enable miscellaneous status signals	Checked, unchecked	Unchecked	—	Adds data_type_o[5:0] output signal.
	Enable handshake signals	Checked, unchecked	Checked	Greyed out, always checked	—

2.4. Modules Description

2.4.1. Clock, Reset and Initialization

Active low reset is used in the design with synchronous release. This is the system reset input connected to the Pixel-to-Byte module.

Follow this initialization and reset sequence:

1. Assert active low system reset for at least three clock cycles of the slower clock (pixel clock or byte clock).
2. IP is ready to process data after reset.

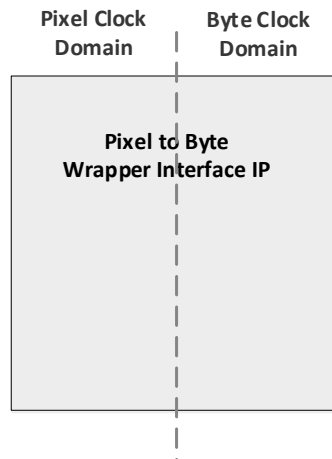


Figure 2.2. Clock Domain Crossing Block Diagram

Table 2.4. Clock Domain Crossing

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

The general formula for computing the required clocks of the IP (clocks used in the computation are in frequency domain):

$$\frac{\text{Byte Clock}}{\text{Pixel Clock}} = \frac{\text{bits per pixel} \times \text{pixel per pixclk}}{\text{number of TX lanes} \times \text{TX gear}}$$

Examples:

1. IP configuration: RGB888 data type; 1 Pixel per Pixel Clock, Tx Gear 8, 4 Tx Lanes:

$$\frac{\text{Byte Clock}}{\text{Pixel Clock}} = \frac{24 \times 1}{4 \times 8}$$

$$\frac{\text{Byte Clock}}{\text{Pixel Clock}} = \frac{3}{4}$$

2. IP configuration: RGB666 data type; 1 Pixel per Pixel Clock, Tx Gear 8, 1 Tx Lane:

$$\frac{\text{Byte Clock}}{\text{Pixel Clock}} = \frac{18 \times 1}{1 \times 8}$$

$$\frac{\text{Byte Clock}}{\text{Pixel Clock}} = \frac{9}{4}$$

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 byteclock, to maintain the required ratio and avoid clock drift.

2.4.2. Pixel-to-Byte Wrapper Module

Pixel-to-Byte Wrapper Module instantiates Pixel2Byte and Synchronizer modules.

Pixel2Byte module is the core module, which does pixel-to-byte packet conversion. It instantiates different wrapper modules for pixel-to-byte converter logic for each data type.

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

2.5. Timing Specifications

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

Figure 2.3 and Figure 2.4 show the timing diagram of display and camera parallel input interface, respectively. It follows the standard DSI/CSI-2 interface protocol with VSYNC, HSYNC, Data Enable for DSI and Frame Valid, Line Valid for CSI-2, pixel data, all 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 fixed to 64-bit. LSB of input pixel data is transmitted first. Byte arrangement depends on the gearing and targeted number of MIPI D-PHY lanes. Each 16-bit of the total bus width corresponds to lane number of the target Tx lanes.

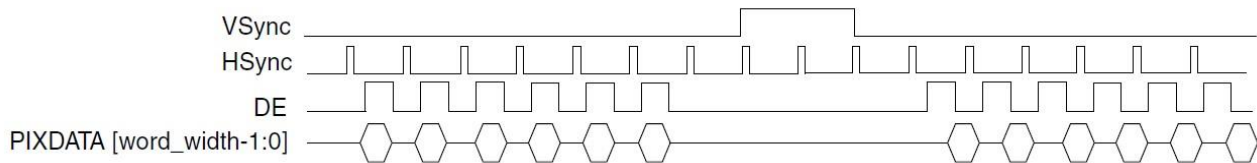


Figure 2.3. Display Parallel Input Interface Timing Diagram (DSI)

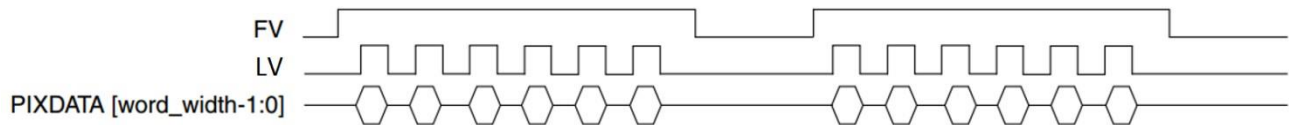


Figure 2.4. Camera Sensor Parallel Input Interface Timing Diagram (CSI-2)

Table 2.5. 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					

LSB is the first valid output byte. Depending on the gearing, each byte is placed accordingly. Unused bytes are padded with 0s.

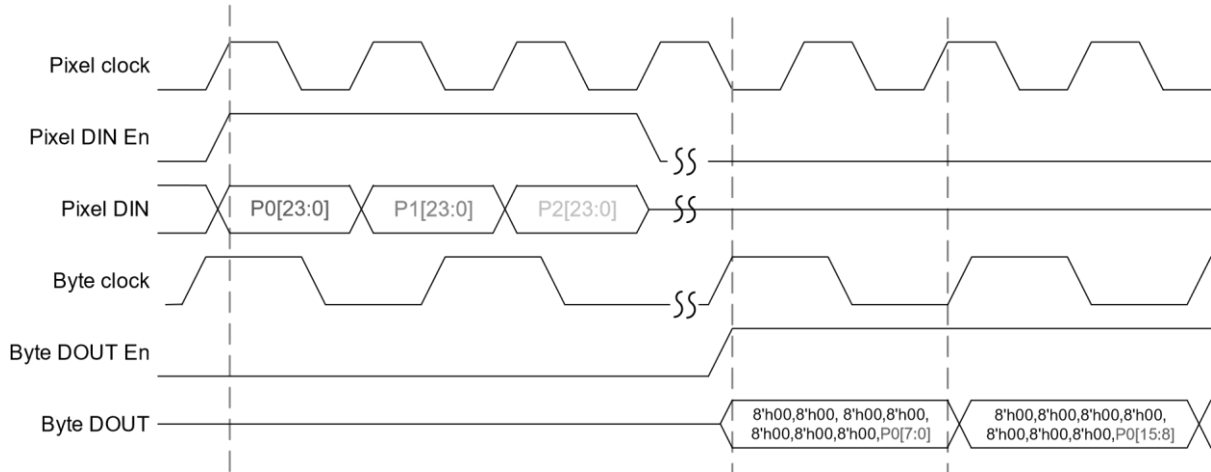


Figure 2.5. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 1 Pixel per Pixel Clock)

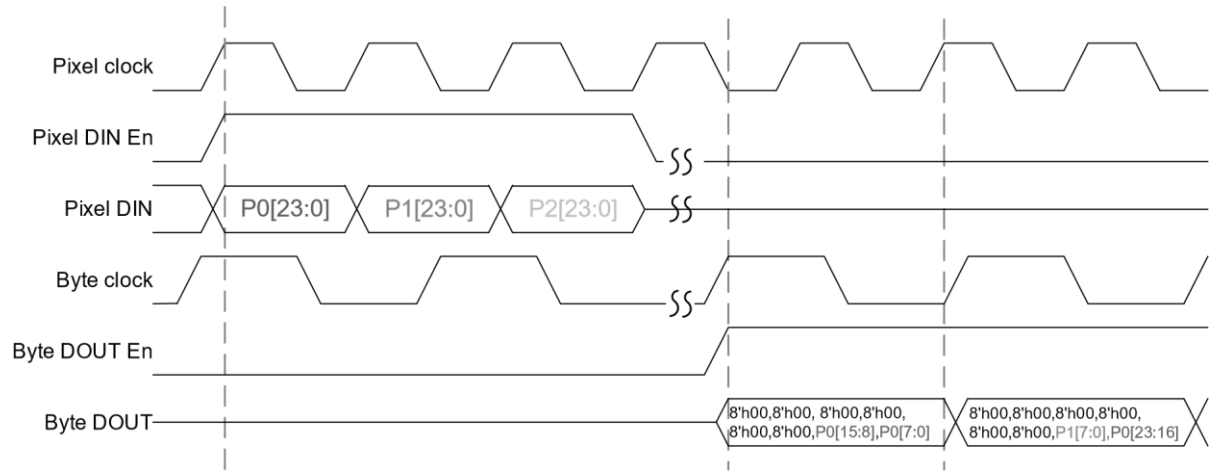


Figure 2.6. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 1 Pixel per Pixel Clock)

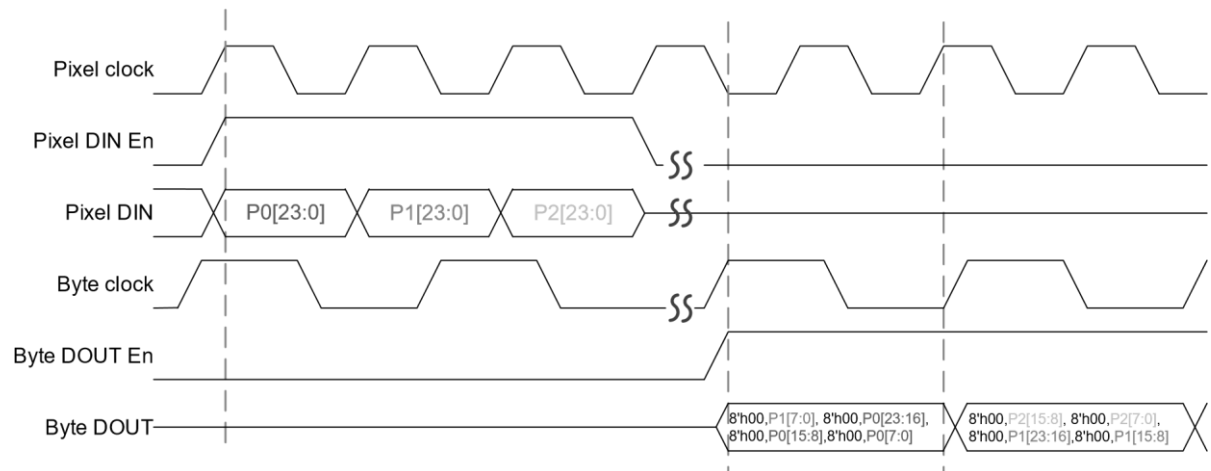


Figure 2.7. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 1 Pixel per Pixel Clock)

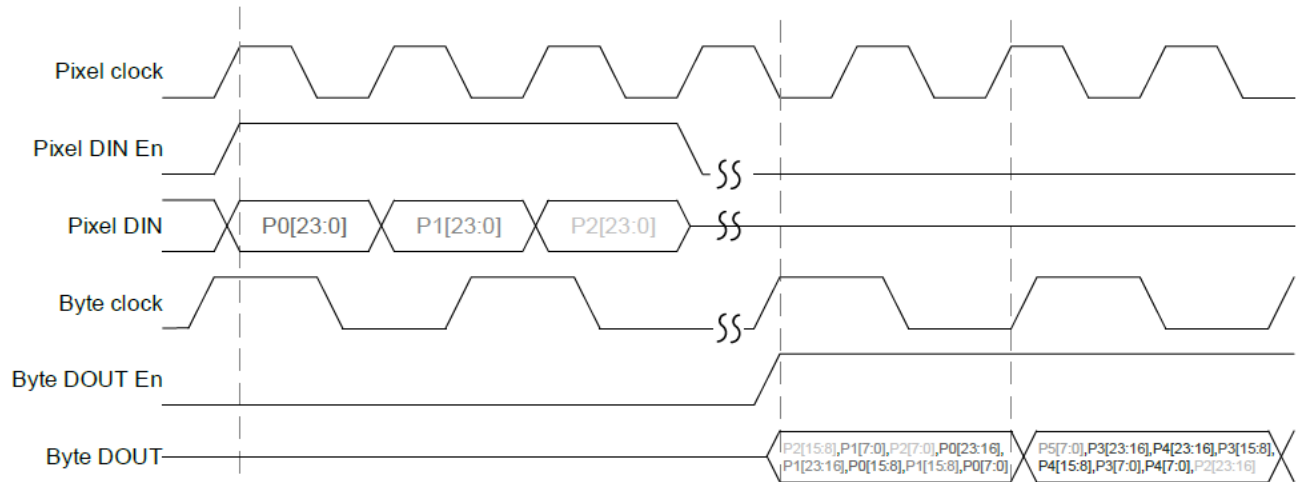


Figure 2.8. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 1 Pixel per Pixel Clock)

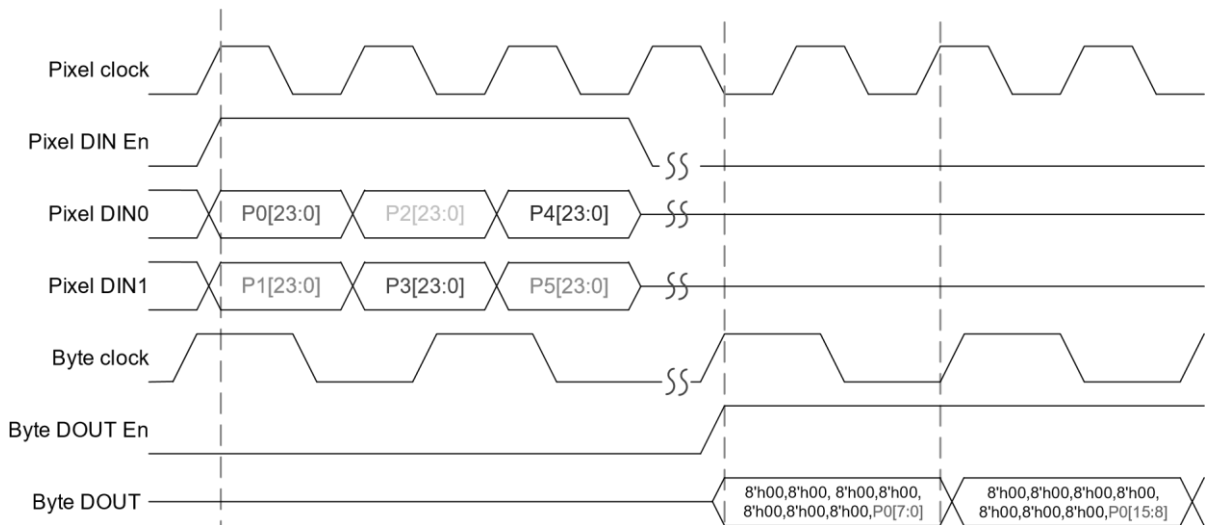


Figure 2.9. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 2 Pixels per Pixel Clock)

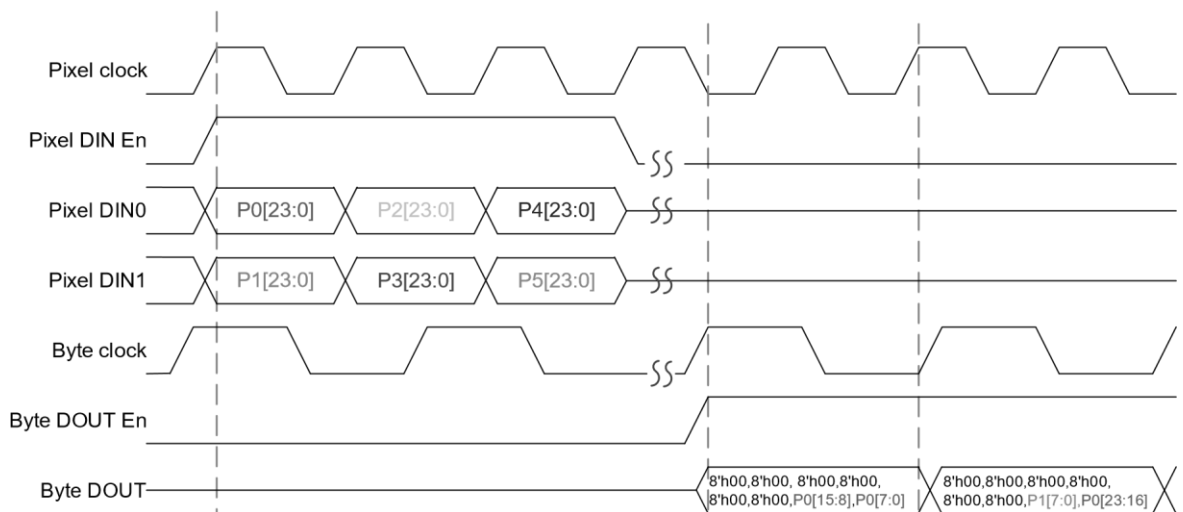


Figure 2.10. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 2 Pixels per Pixel Clock)

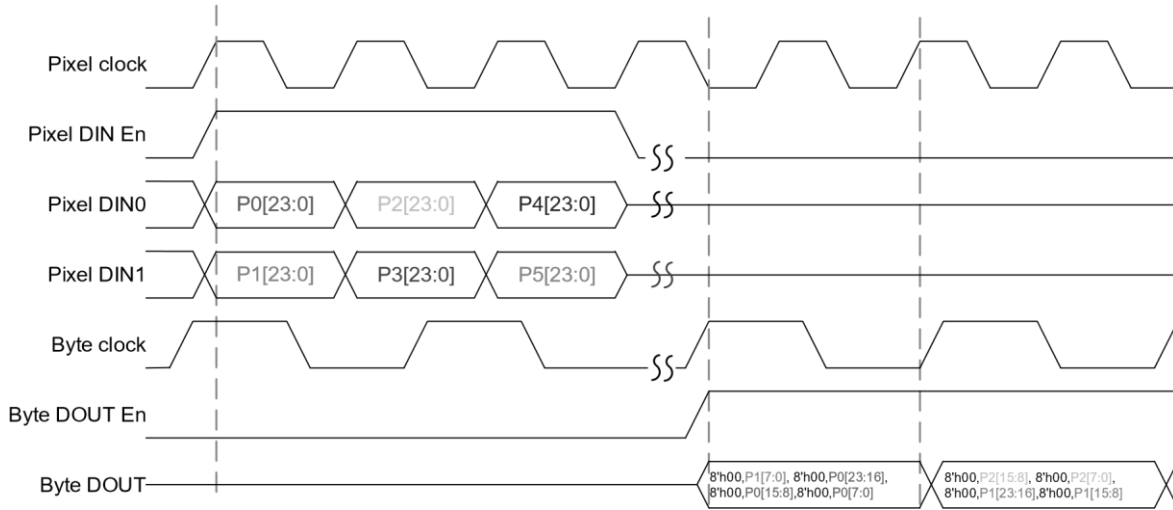


Figure 2.11. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 2 Pixels per Pixel Clock)

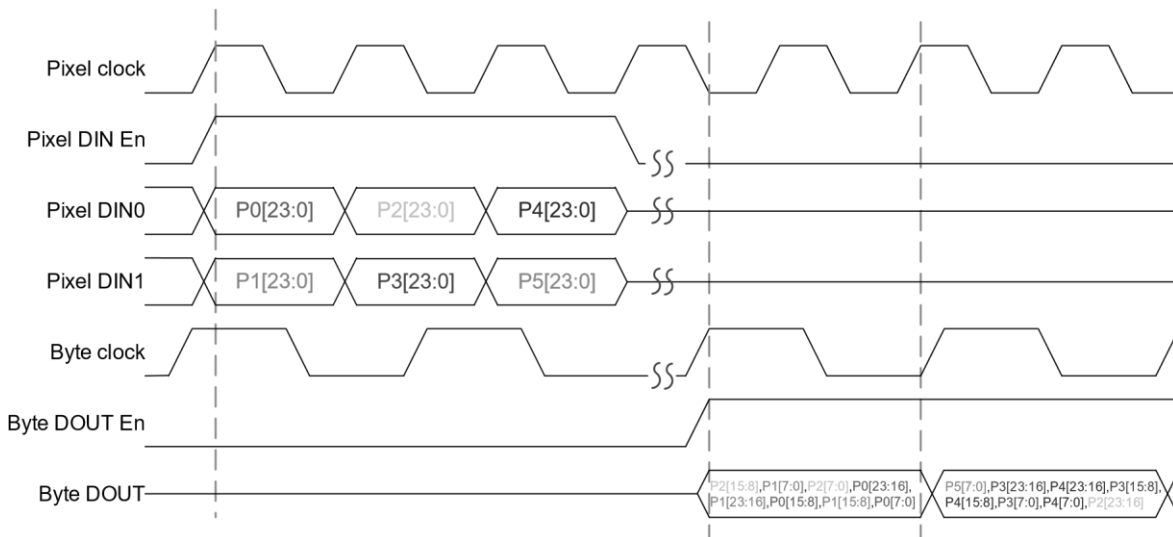


Figure 2.12. Sample Input to Output Timing Diagram (RGB888, Gear 16, 4 Tx lanes, 2 Pixels per Pixel Clock)

As Pixel-to-Byte Converter IP is interfaced to other MIPI D-PHY related IPs, input signal txfr_en_i is expected to be asserted when MIPI D-PHY lanes are already in High Speed (HS) mode. HS mode refers to the state after THS-ZERO until just before THS-TRAIL (see Figure 2.13.)

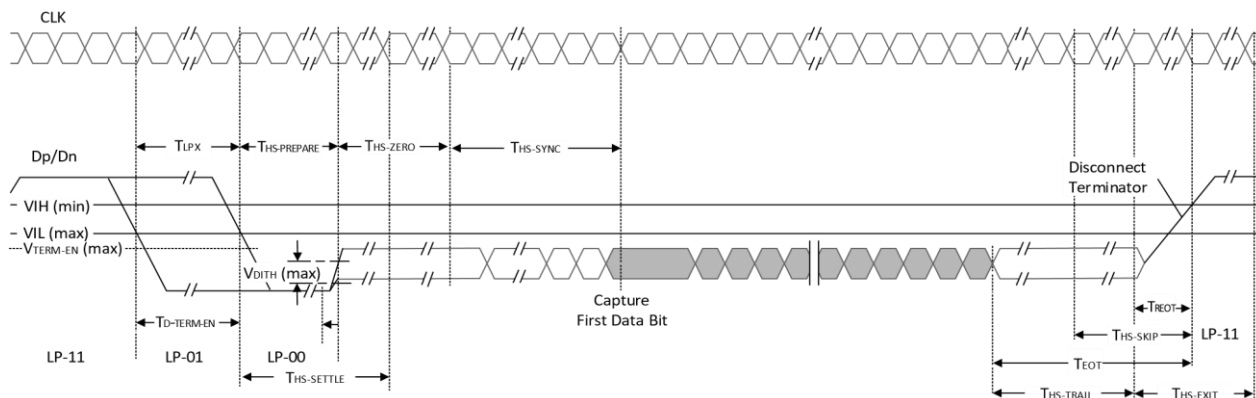


Figure 2.13. MIPI D-PHY High Speed Transmission Timing Diagram

Output signal txfr_req_o of the IP is asserted just before TLPX when the c2d_ready_i signal is Hi. The input signal txfr_en_i should be asserted just right after THS-ZERO until D-PHY lanes go to THS-TRAIL. 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.

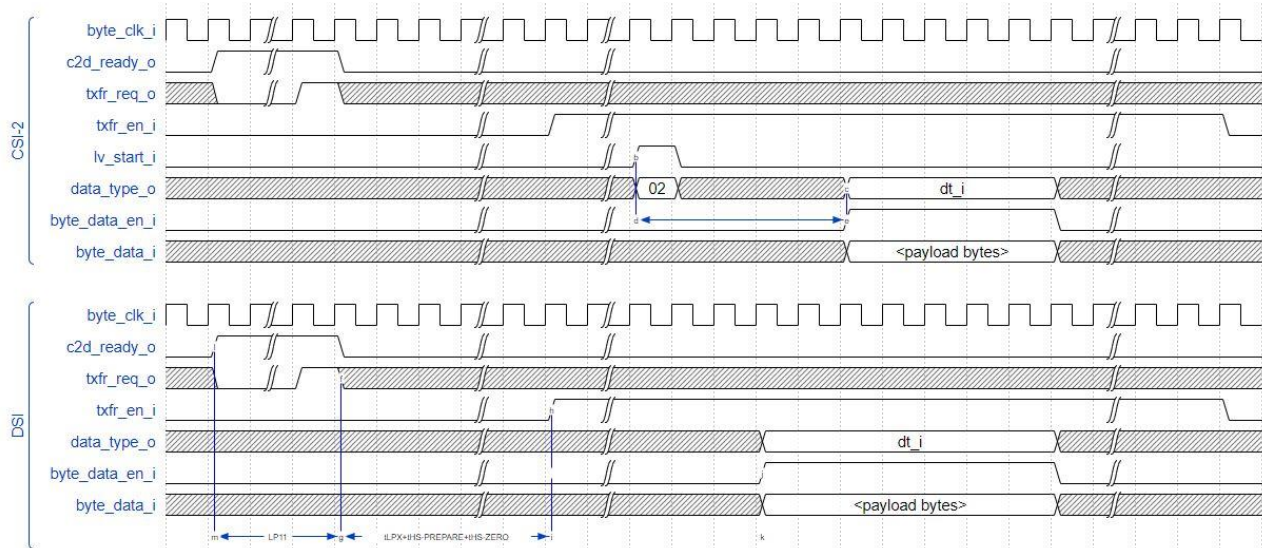


Figure 2.14. Handshake Signals Timing Diagram

3. IP Generation and Evaluation

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 on Lattice Radiant software, refer to the [Lattice Radiant Software 2.1 User Guide](#) and relevant Lattice tutorials.

3.1. Licensing the IP

An IP core-specific license string is required enable full use of this IP core in a complete, top-level design. You can fully evaluate the IP core through functional simulation and implementation (synthesis, map, place, and route) without an IP license string. This IP core supports Lattice’s IP hardware evaluation capability, which makes it possible to create versions of the IP core, which operate in hardware for a limited time (approximately four hours) without requiring an IP license string. See Hardware Evaluation section for further details. However, a license string is required to enable timing simulation and to generate bitstream file that does not include the hardware evaluation timeout limitation.

3.2. Generation and Synthesis

Lattice Radiant software allows you to generate and customize modules and IPs and integrate them into the device architecture. The procedure for generating Pixel-to-Byte Converter IP in Lattice Radiant software is described below.

To generate the Pixel-to-Byte Converter IP:

1. In the **Module/IP Block Wizard**, create a new Lattice Radiant software project for Pixel-to-Byte Converter module.
2. In the dialog box of the **Module/IP Block Wizard** window, configure the Pixel-to-Byte Converter module according to custom specifications using drop-down menus and check boxes. As a sample configuration, see [Figure 3.1](#). For configuration options, see [Table 2.2](#).

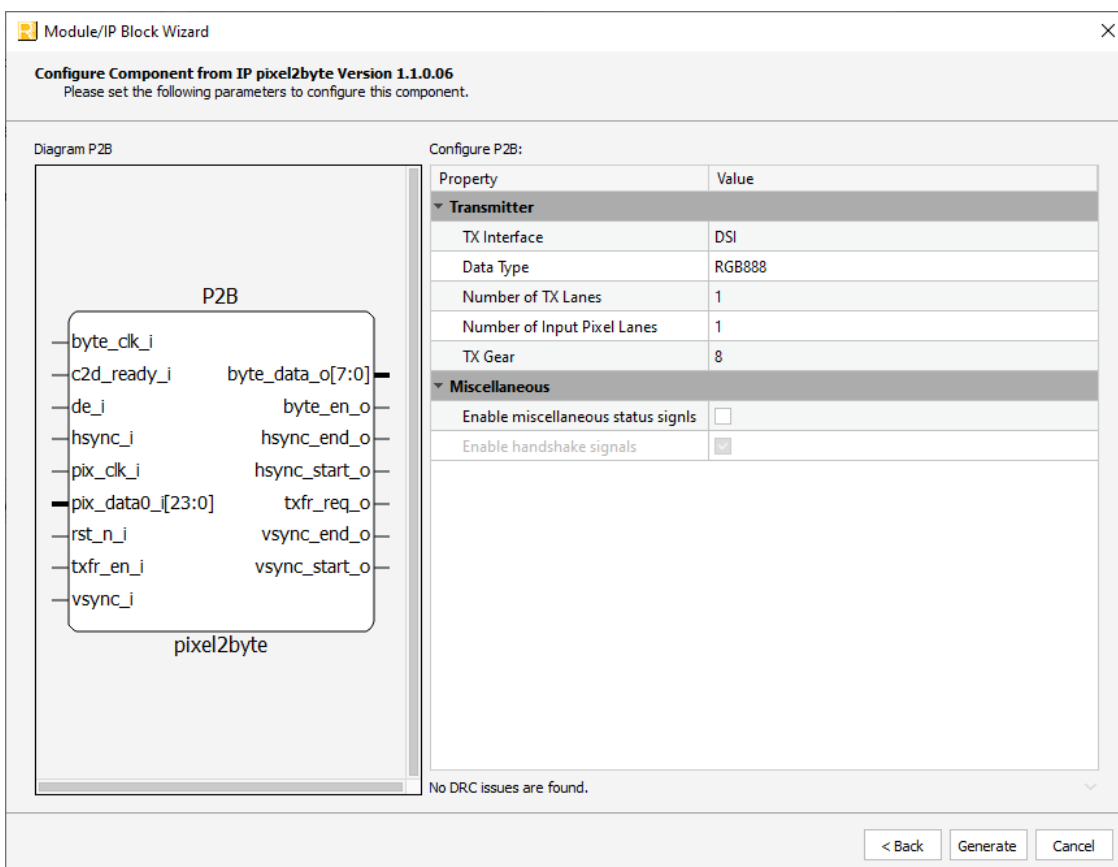


Figure 3.1. Configure Block of Pixel-to-Byte Converter

3. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results.
4. Click the **Finish** button to generate the Verilog file.
5. After generating the design, you can synthesize it by clicking **Synthesize Design** located on the top left corner of the screen, as shown in [Figure 3.2](#).

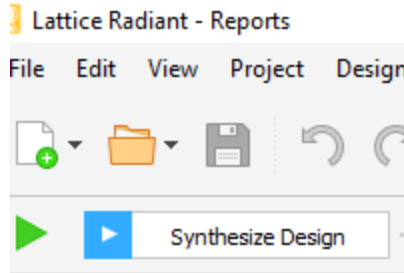



Figure 3.2. Synthesizing the Design

3.3. Functional Simulation

To run Verilog simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in [Figure 3.3](#).

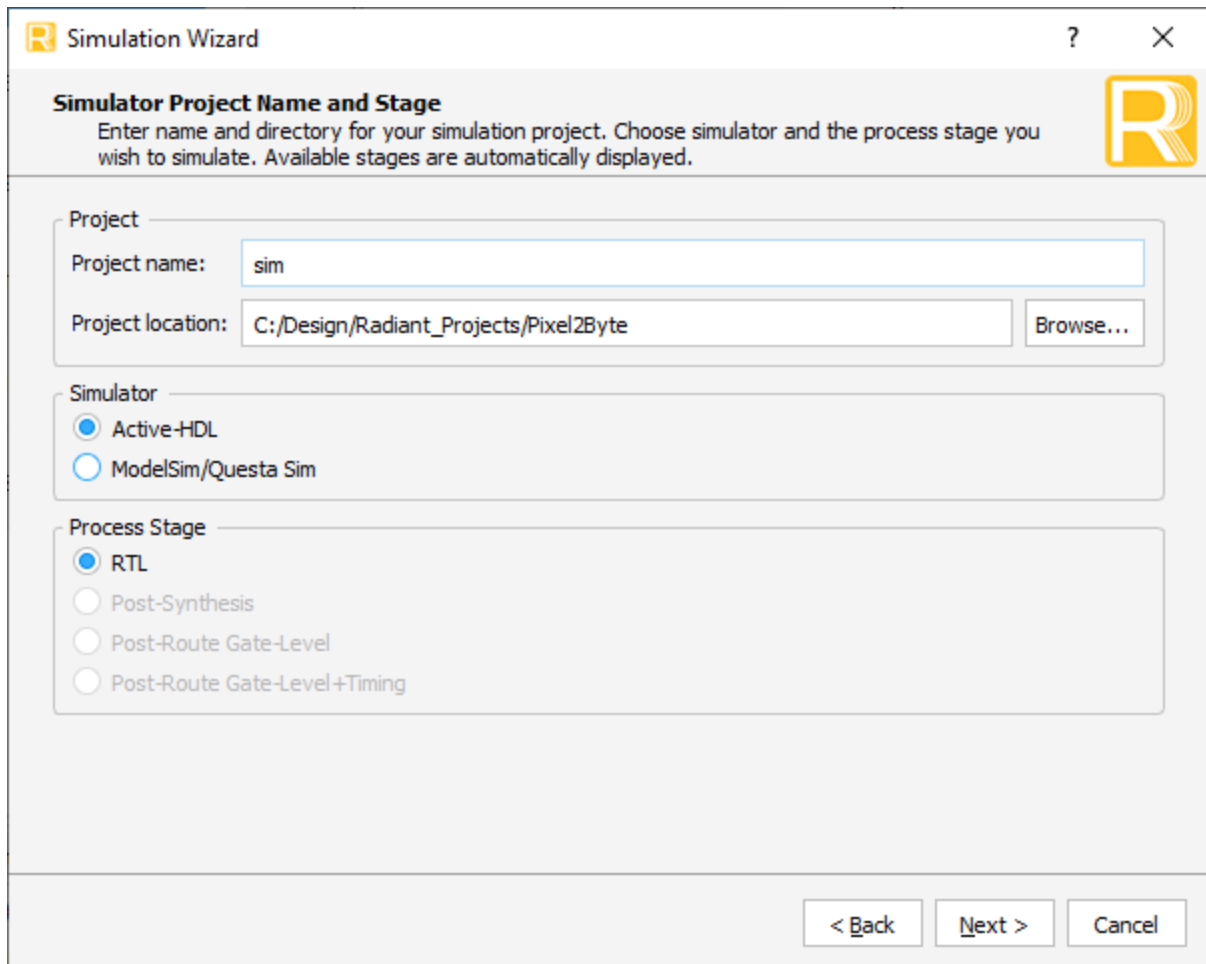


Figure 3.3. Simulation Wizard

2. Double-click **Next** to open the **Add and Reorder Source** window as shown in [Figure 3.4](#).

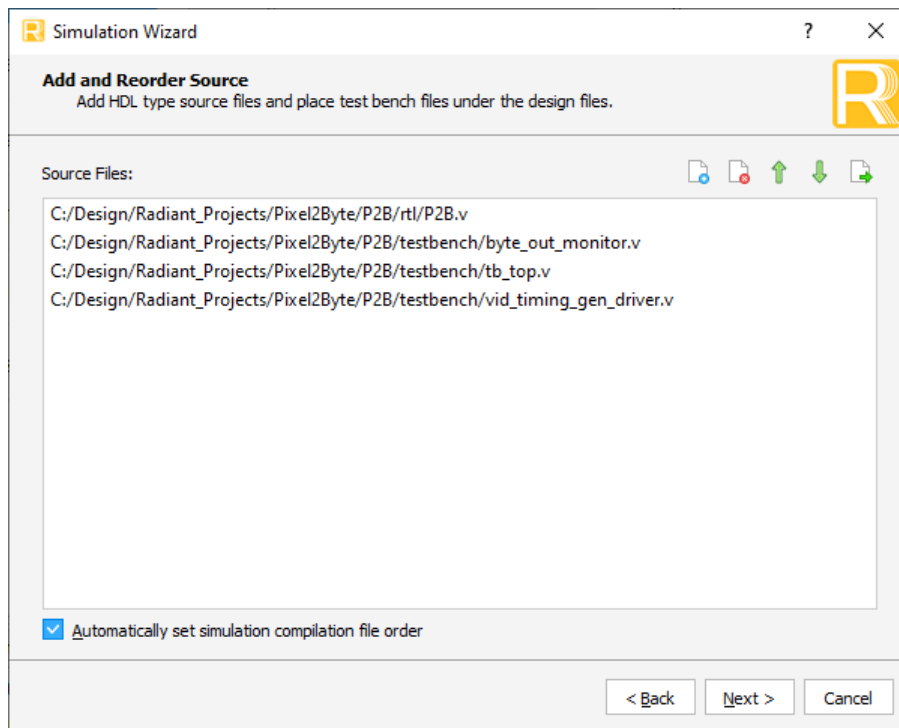


Figure 3.4. Adding and Reordering Source

3. Add the *tb_top.v* file from the *testbench* directory if not included.
4. Click **Next** to run the simulation.

3.3.1. Required Post-Synthesis Constraints

The Pixel to Byte Converter IP for Radiant has a default pre-synthesis constraints file *.ldc located in the constraints folder. For Map, Place and Route, a Post-Synthesis Constraints (*.pdc) file is needed as shown in [Figure 3.5](#).

To properly constrain the byte to pixel design post-synthesis:

1. By default, the periods of the clocks are for 200 MHz designs. Adjust those based on the actual design to relax the timing.
2. Modify the paths based on the generated netlist.

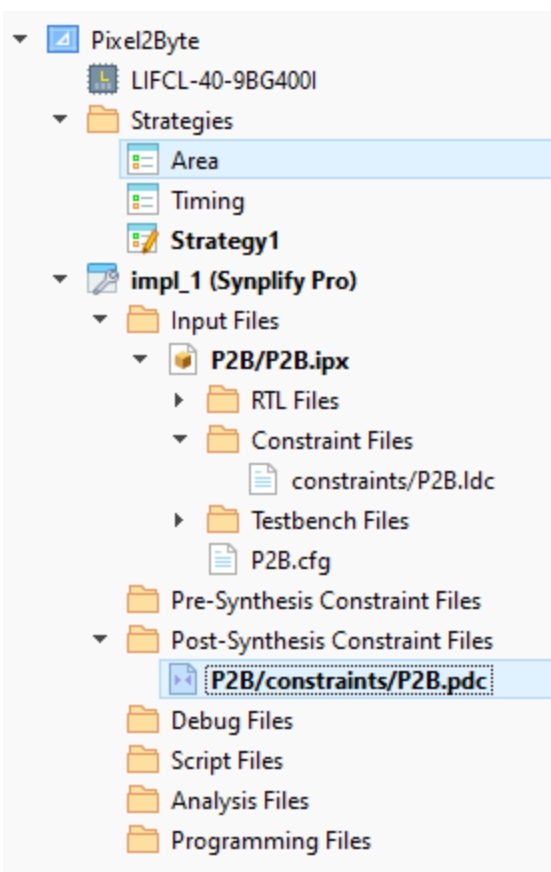


Figure 3.5. Adding Constraint

4. Ordering Part Number

The Ordering Part Number (OPN) for this IP Core are the following:

- PIXEL-BYTE-CNX-U – Pixel to Byte Converter for CrossLink-NX – Single Design License
- PIXEL-BYTE-CNX-UT – Pixel to Byte Converter for CrossLink-NX – Site License
- PIXEL-BYTE-CTNX-U – Pixel to Byte Converter for Certus-NX – Single Design License
- PIXEL-BYTE-CTNX-UT – Pixel to Byte Converter for Certus-NX – Site License

Appendix A. Resource Utilization

Table A.1 and Table A.2 show the details about the device, tools used and resource utilization for a certain IP configuration.

For more information on Lattice Radiant software, visit the Lattice website at www.latticesemi.com/Products/DesignSoftwareAndIP.

Table A.1. Device and Tool Tested

	Value
Diamond Software Version	Radiant 2.1 production build
Device Used	LIFCL-40-BCG400
Performance Grade	9_High-Performance_1.0V
Synthesis Tool	Synplify Pro (R) Q-2020.03LR, Build 134R, May 8 2020

Table A.2. Resource Utilization

Device	LUTs	Register	sysMEM EBRs	Programmable I/O
Default	200	150	1	45
DSI, RGB666, Number of TX Lanes 2	302	216	1	47
CSI-2, RGB888, Number of TX Lanes 4	255	249	2	69
CSI-2, RAW8, Number of TX Lanes 4	191	252	2	53
DSI, RGB888, Number of TX Lanes 4, Number of Input Pixel Per Clock 4, TX Gear 16	477	376	6	173
CSI-2, RGB888, Number of TX Lanes 2, TX Gear 16	336	191	1	69

Note: 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*.

Appendix B. Limitations

Post-Route Gate-Level simulation may fail on some configuration when using LSE but passing using Synplify Pro.

References

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, see the [Lattice Radiant Software 2.1 User Guide](#).

Technical Support Assistance

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

Revision History

Document Revision 1.2, August 2020

Section	Change Summary
Introduction	Updated Table 1.1 .
Functional Description	<ul style="list-style-type: none"> Updated Table 2.1 and Table 2.3 in Attributes Summary. Updated Figure 2.14.
IP Generation and Evaluation	<ul style="list-style-type: none"> Updated figures in this section. Added Required Post-Synthesis Constraints section.
Appendix A. Resource Utilization	<ul style="list-style-type: none"> 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