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

User Guide

FPGA-IPUG-02094-1.5

December 2022

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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
APB	Advance Peripheral Bus
AXI	Advance eXtensible Interface
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, Certus™-NX, CertusPro™-NX, MachXO5-NX, and Avant 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-DPHY byte packet compatible format.

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

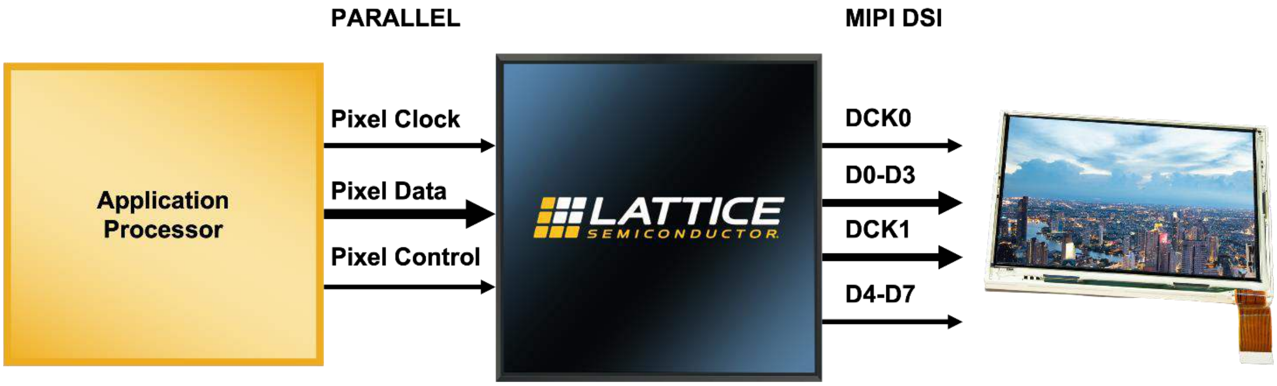


Figure 1.1. Sample Parallel Interface to MIPI DSI System Diagram

1.1. Quick Facts

Table 1.1 presents a summary of the IP Core.

Table 1.1. Quick Facts

IP Requirements	Supported FPGA Families	CrossLink-NX, Certus-NX, CertusPro-NX, MachXO5-NX, Avant-E
Resource Utilization	Targeted Devices	LIFCL-40, LIFCL-17, LFD2NX-40, LFD2NX-17, LFCPNX-100, LIFCL-33, LFMXO5-25, LAV-AT-500E
	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 IP Core v1.3.x - Lattice Radiant software 3.0 or later
	Synthesis	Lattice Synthesis Engine (LSE)
		Synopsys® Synplify Pro® for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software 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, RAW14, RAW16, YUV420/YUV422 8/10-bit video formats
- Conversion of 1, 2, 4, 6, 8, or 10 pixels per pixel clock into MIPI D-DPHY byte packet compatible format
- Support for byte arrangement for 1, 2, or 4 MIPI D-PHY data lanes
- Optional AXI4 Streaming interface for byte and pixel data
- APB interface for configuration and status

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.

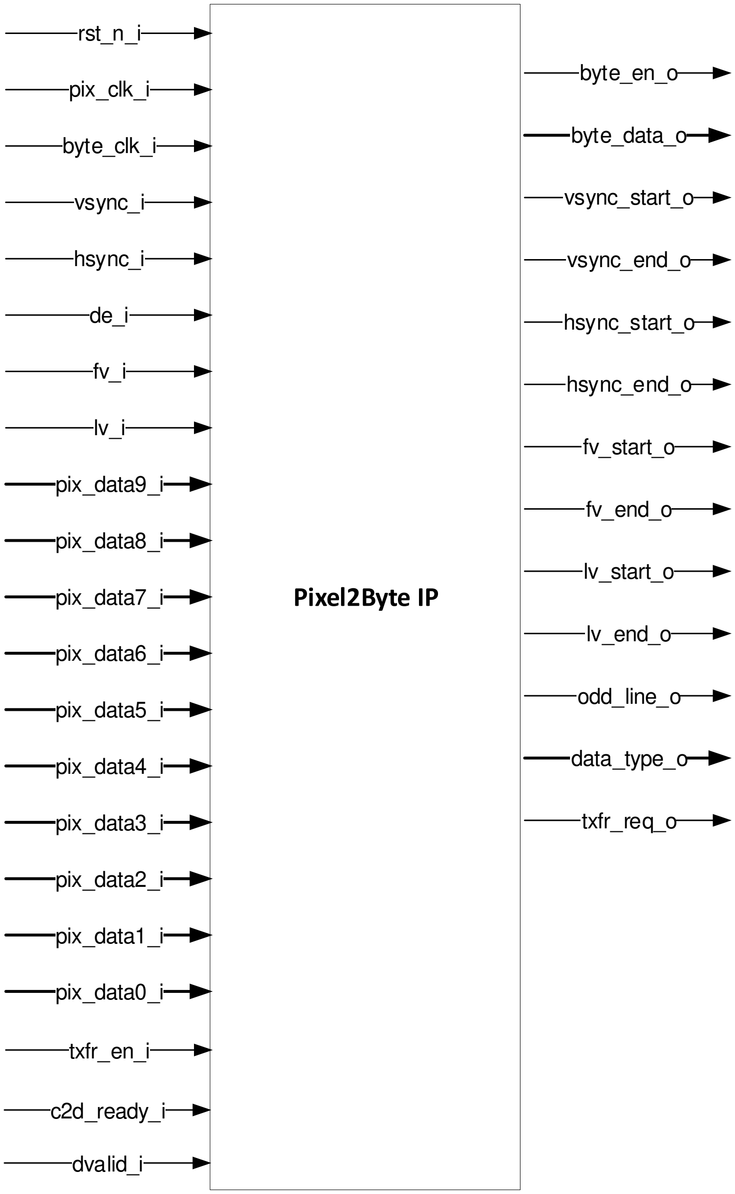


Figure 2.1. Pixel-to-Byte IP Functional Diagram

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 16-bit bus width - RAW16 14-bit bus width - RAW14 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 16-bit bus width - RAW16 14-bit bus width - RAW14 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
AXI Slave interface		
axis_tvalid_i	I	AXI4 data valid
axis_tdata_i	I	AXI4 data. Data width is (number of Pix Lanes * Pix data width)
axis_tuser_i	I	AXI4 user signals. Bit[1:0] : Reserved Bit[2] : hsync_i/fv_i Bit[3] : vsync_i/lv_i
axis_tready_o	O	AXI4 slave ready. This is to throttle the Pixel data coming in
Byte Domain Outputs		
vsync_start_o[2]	O	Pulse signal used to indicate Vsync start

Port Name	Direction	Description
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
byte_data_o	O	Output data width is (number of Tx Lanes * Tx gear)
AXI Master interface		
axim_tvalid_o	O	AXI4 data valid
axim_tdata_o	O	AXI4 data. Data width is (number of Tx Lanes * Tx gear)
axim_tready_i	O	AXI4 ready signal. Salve uses this to throttle to the byte data going out
APB interface		
apb_pclk_i	I	APB clock. The maximum frequency is 200MHz
apb_presetn_i	I	APB active low reset.
apb_prdata_o	O	APB read data
apb_pready_o	O	APB ready
apb_pslverr_o	O	APB slave error
apb_paddr_i	I	APB address
apb_pwdata_i	I	APB write data
apb_penable_i	I	APB enable
apb_psel_i	I	APB slave select
apb_pwrite_i	I	APB slave write
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. • 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 Timing Specifications section for details on their functionality.

2.3. Attributes Summary

Table 2.2 lists the supported configurations.

Table 2.2. Supported Configurations

Configuration Type	Number of input pixel lanes	Tx Lanes	Tx Gearing	Data Type
DSI	1	1	8	RGB666 RGB888
			16	RGB666 RGB888
		2	8	RGB666 RGB888
			16	RGB666 RGB888
		4	8	RGB666 RGB888
			16	RGB666 RGB888
	2	1	8	RGB666 RGB888
			16	RGB666 RGB888
		2	8	RGB666 RGB888
			16	RGB666 RGB888
		4	8	RGB666 RGB888
			16	RGB666 RGB888
	4	4	16	RGB666 RGB888
CSI-2	1	1	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565

Configuration Type	Number of input pixel lanes	Tx Lanes	Tx Gearing	Data Type
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565
		2	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565
		4	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565

Configuration Type	Number of input pixel lanes	Tx Lanes	Tx Gearing	Data Type
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888 RGB444 RGB555 RGB565
	2	1	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888
		2	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888

Configuration Type	Number of input pixel lanes	Tx Lanes	Tx Gearing	Data Type
		4	8	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888
			16	RAW10 YUV420 10-bit YUV422 10-bit RAW12 RAW14 RAW16 RAW8 YUV420 8-bit YUV422 8-bit RGB888
	4	4	8	RAW10 RAW12 RGB444 RGB555 RGB565
	6	4	8	RAW10 RAW12
	8	4	16	RAW10 RAW12
	10	4	16	RAW10 RAW12

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
General	Data Type	RGB888, RGB666, RGB444, RGB555, RGB565, RAW8, RAW10, RAW12, RAW14, RAW16, YUV420_8 YUV420_10 YUV422_8 YUV422_10	RGB888	None	Specify the data type depending on the Data Format selected.
Pixel Interface	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.
	Pixel Clock Frequency (MHz)	Range 10-350 MHz	80	None	Specify the Pixel clock frequency.
	Enable AXI4-Stream Receiver Interface	ON/OFF	OFF	None	If this is “ON”, Slave AXI4 Interface is used instead of native Pixel interface.
Byte Interface	Tx Interface	DSI, CSI2	CSI2	None	Set the Tx interface.
	DSI Mode	Non-Burst Pulses, Non-Burst Events	Non-Burst Pulses	Tx Interface	Specify the mode for DSI.
	Number of Tx Lanes	1, 2, 4	1	None	Set the target number of MIPI D-PHY Tx 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.
	Byte Clock Frequency (MHz)	Range 10-500 MHz	80	Based on GEAR, TX/RX width, pixel clock and the data type	Specify the Byte clock frequency.
	Enable AXI4-Stream Transmitter Interface	ON/OFF	OFF	None	If this is “ON”, Master AXI4 Interface is used instead of native Byte interface
Miscellaneous	Enable APB Interface	ON/OFF	ON	None	If this is “ON”, APB Slave interface is used instead of the native Pixel Interface.
	Enable Handshake Signals	Checked, unchecked	Checked	Greyed out, always checked	—
FIFO	Word Count	Range	24	Data Type	MIN_WC value depends on

		MIN_WC-65535			the data type. See Table 2.5 to see the count restriction.
	Manual Adjust	Checked, Unchecked	Unchecked	None	Check this to set the Read delay and FIFO depth manually.
	Read delay	Range 2 - 65535	2	Manual Adjust	If Manual Adjustment is checked, Read delay can be set manually.
	FIFO Depth	Range 8 – 65536	512	Manual Adjust	If Manual Adjustment is checked, FIFO depth can be set manually.

Table 2.4. Word Count Restriction

Data Type	Word Count Restriction
RGB888	multiple of 24
RGB666	multiple of 18
RGB444	multiple of 12
RGB555	multiple of 15
RGB565	multiple of 16
RAW8	multiple of 8
RAW10	multiple of 10
RAW12	multiple of 12
RAW14	multiple of 14
RAW16	multiple of 16
YUV420_8	multiple of 8
YUV420_10	multiple of 10
YUV422_8	multiple of 8
YUV422_10	multiple of 10

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.

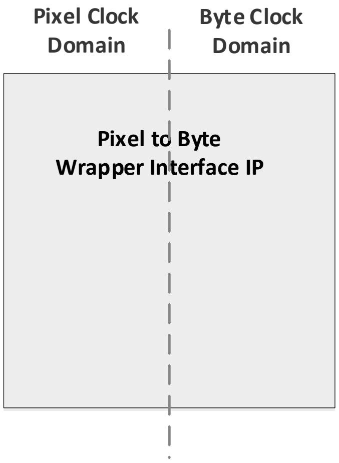


Figure 2.2. Clock Domain Crossing Block Diagram

Table 2.5. Clock Domain Crossing

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

The general formula for maintaining the same data bandwidth between pixel domain and byteclock domain is shown in the equation below:

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

Examples:

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

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

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

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

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

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

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. Otherwise, the FIFO depth and read delay can be adjusted to compensate for the difference in bandwidth

2.4.2. Pixel-to-Byte Wrapper Module

Pixel-to-Byte Wrapper Module instantiates the 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.

2.4.3. AXI4 Stream Slave

In Pixel domain, AXI4-Stream Slave provides transmission of payload packets to Pixel-to-Bye Converter module. The axis_sready_o is used to throttle the Pixel data if incase of TX-FIFO is becoming full. When axis_svalid_i asserted to 1, then axis_sdata_o receives the payload video stream. AXI data width is pixel data*num of pixel lanes.

2.4.4. AXI4 Stream Master

In byte domain, AXI4-Stream Master provides transmission of data converted to byte format.

AXI transmit data valid acts as a byte valid/enable and data is drive on the AXI transmit data bus.

2.4.5. APB Slave

Design implements an APB 3.0 slave interface for register programming and status reading. Design maintains below registers for configuration/status. Please refer APB 3.0 spec for protocol details.

REG0

Bits	Attribute	Reset value	Description
Bit-0 : TX-FIFO Full	*W1C	0	FIFO full indication. Sticky bit. Write 0 to clear
Bit-1 : TX-FIFO Empty	*RW	0	FIFO empty indication
Bit-31:2 : Reserved	NA	0	Reserved

REG1

Bits	Attribute	Reset value	Description
Bit-1:0 : VC	RW	0	Virtual channel
Bit-17:2 : WC	RW	0	Word count
Bit-31:18 : Reserved	NA	0	Reserved

*W1C: Write 1 to clear; RW: Read & Write; NA: Not Applicable

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.

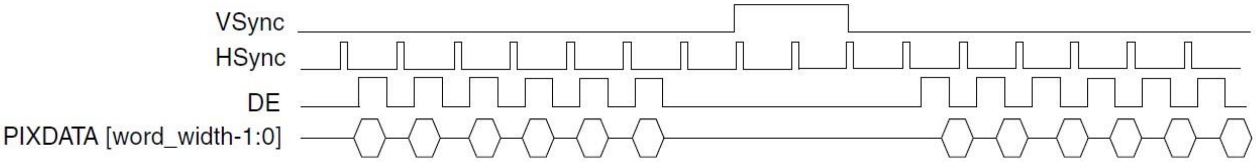


Figure 2.3. Display Parallel Input Interface Timing Diagram (DSI)

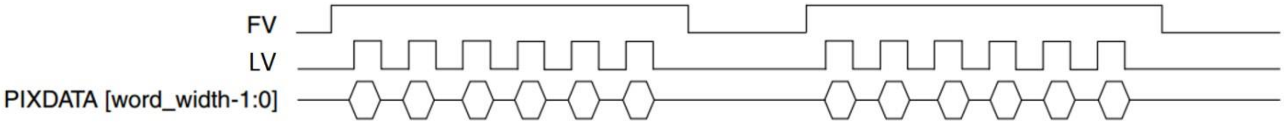


Figure 2.4. Camera Sensor Parallel Input Interface Timing Diagram (CSI-2)

Table 2.6. 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.

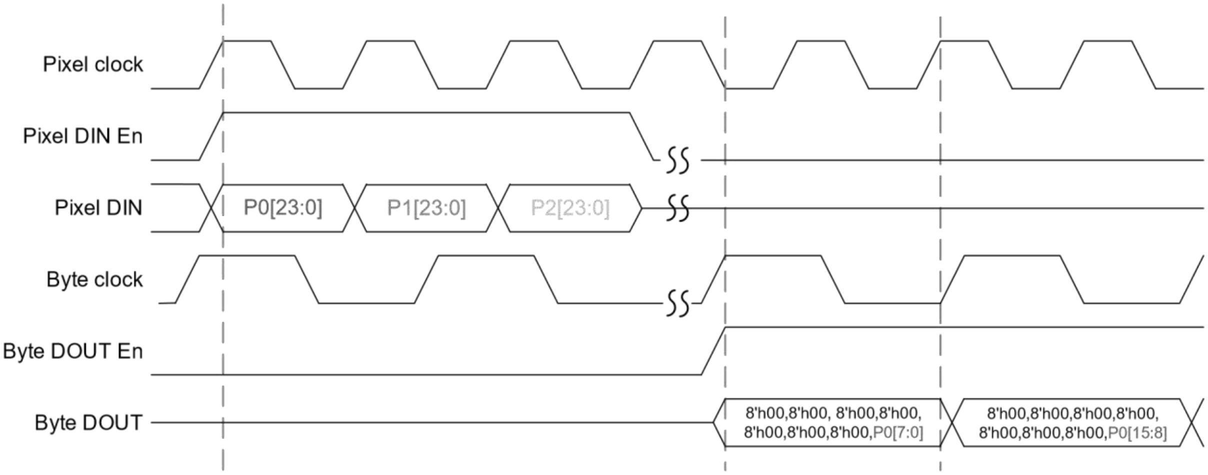


Figure 2.5. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 1 Pixel per Pixel Clock)

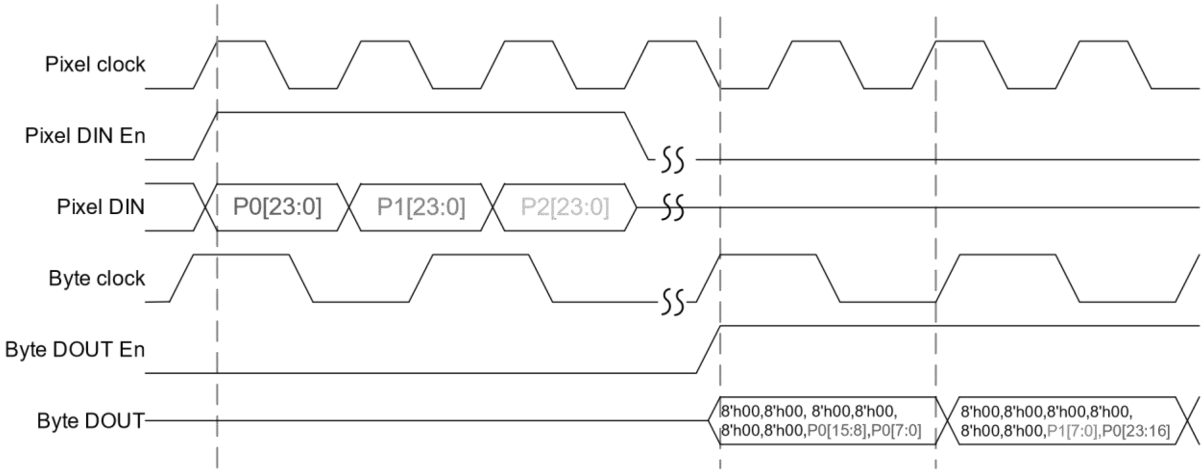


Figure 2.6. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 1 Pixel per Pixel Clock)

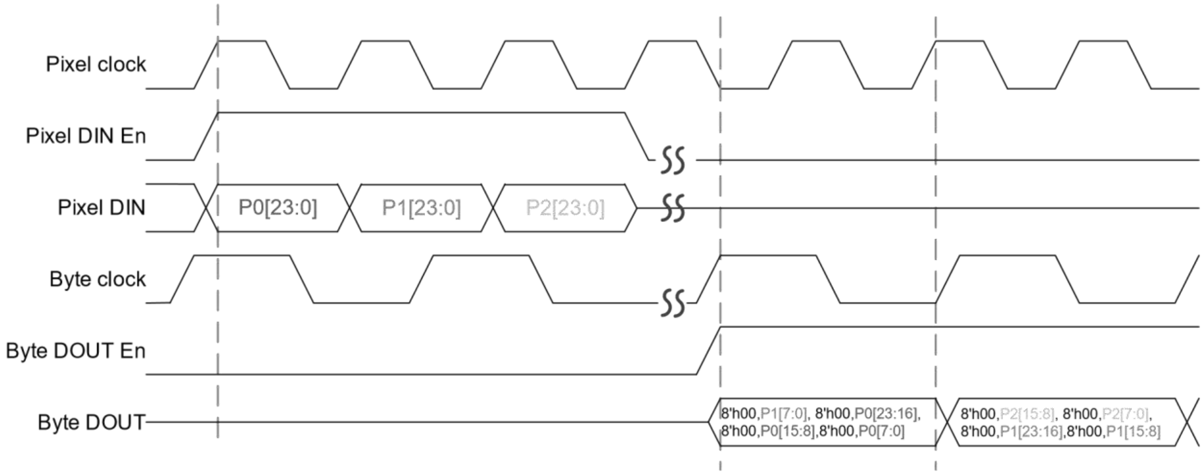

Pixel clock
Pixel DIN En
Pixel DIN
P0[23:0]
P1[23:0]
P2[23:0]
Byte clock
Byte DOUT En
Byte DOUT
8'h00,P1[7:0], 8'h00,P0[23:16],
8'h00,P0[15:8],8'h00,P0[7:0]
8'h00,P2[15:8], 8'h00,P2[7:0],
8'h00,P1[23:16],8'h00,P1[15:8]

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

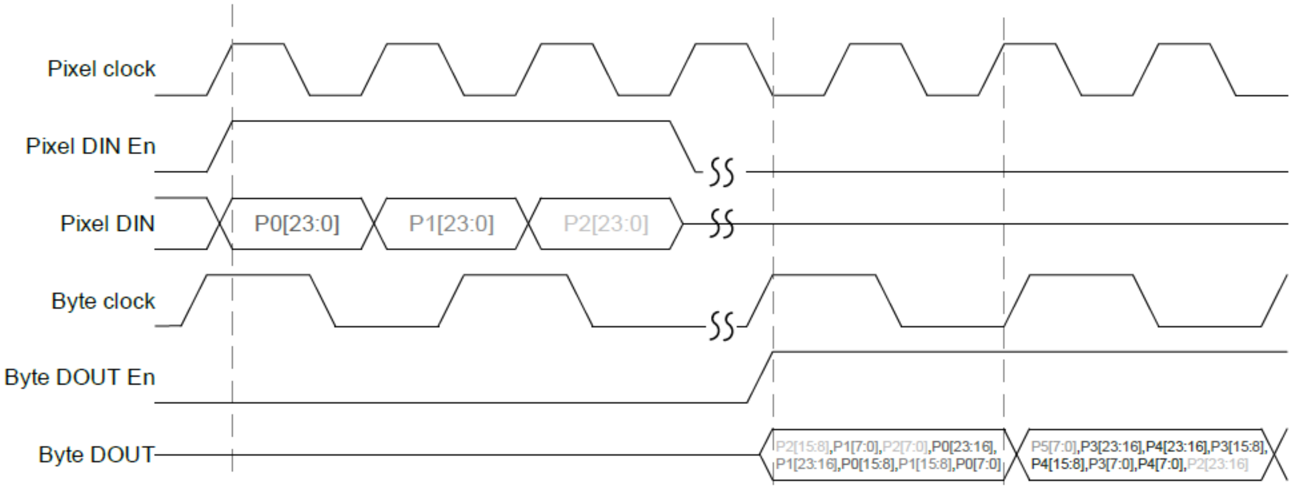

Pixel clock
Pixel DIN En
Pixel DIN
P0[23:0]
P1[23:0]
P2[23:0]
Byte clock
Byte DOUT En
Byte DOUT
P2[15:8],P1[7:0],P2[7:0],P0[23:16],
P1[23:16],P0[15:8],P1[15:8],P0[7:0]
P5[7:0],P3[23:16],P4[23:16],P3[15:8],
P4[15:8],P3[7:0],P4[7:0],P2[23:16]

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

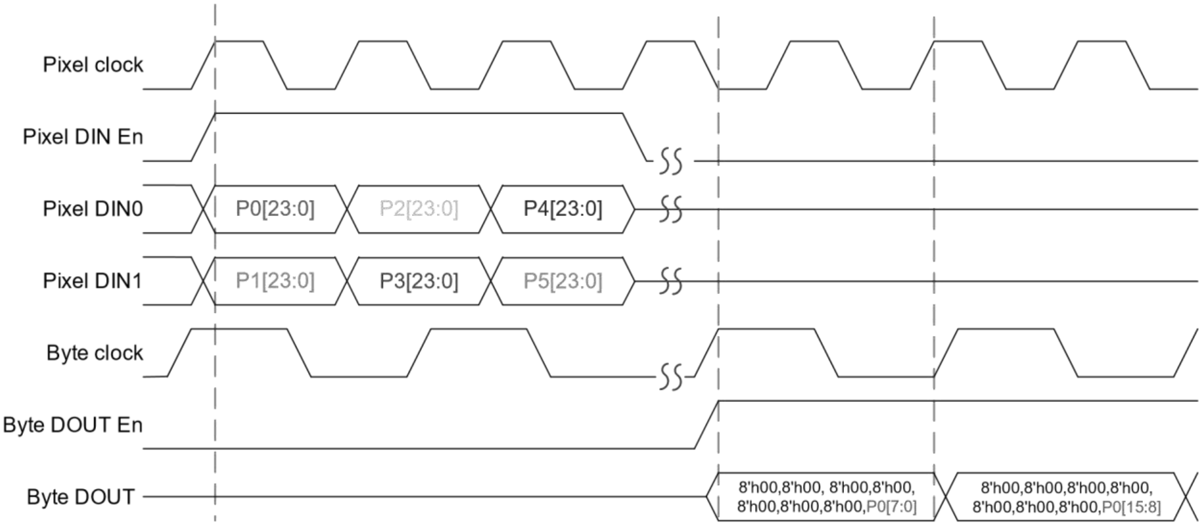


Figure 2.9. Sample Input to Output Timing Diagram (RGB888, Gear 8, 1 Tx lane, 2 Pixels per Pixel Clock)

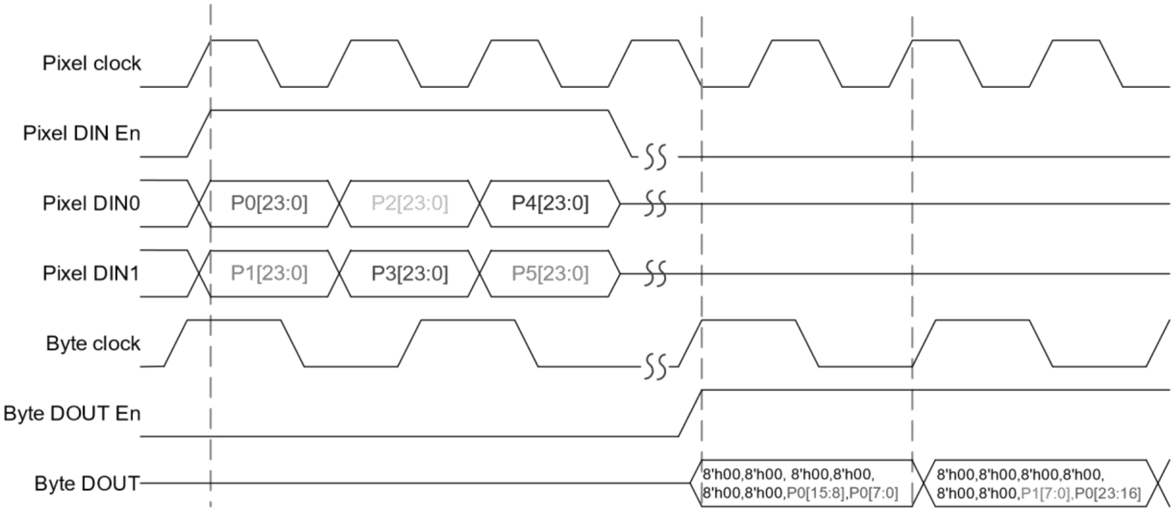


Figure 2.10. Sample Input to Output Timing Diagram (RGB888, Gear 16, 1 Tx lane, 2 Pixels per Pixel Clock)

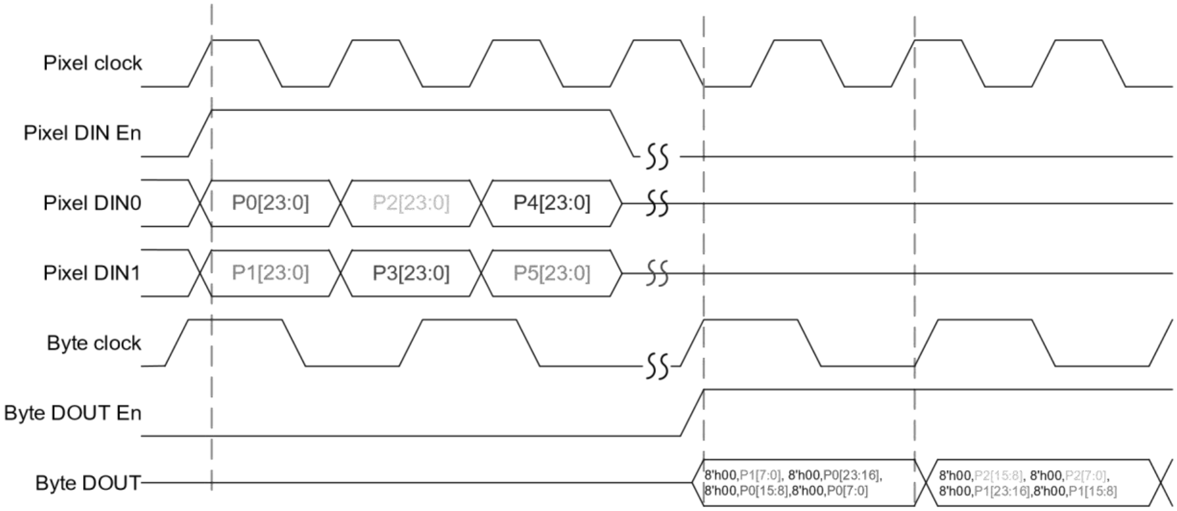


Figure 2.11. Sample Input to Output Timing Diagram (RGB888, Gear 8, 4 Tx lanes, 2 Pixels per Pixel Clock)

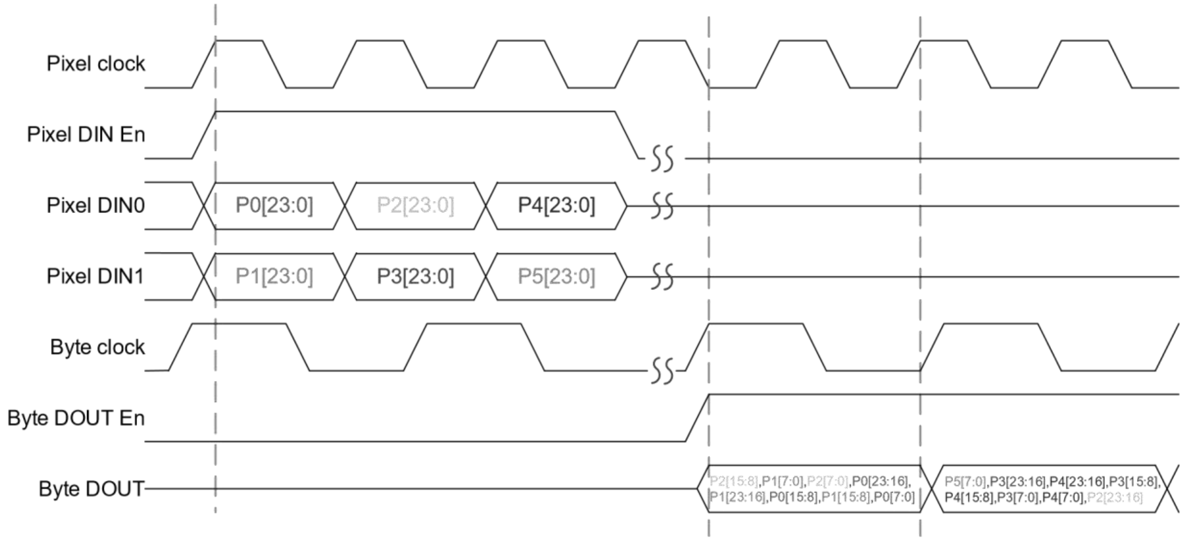

Pixel clock
Pixel DIN En
Pixel DIN0
P0[23:0]
P2[23:0]
P4[23:0]
Pixel DIN1
P1[23:0]
P3[23:0]
P5[23:0]
Byte clock
Byte DOUT En
Byte DOUT
P2[15:8],P1[7:0],P2[7:0],P0[23:16],
P1[23:16],P0[15:8],P1[15:8],P0[7:0]
P5[7:0],P3[23:16],P4[23:16],P3[15:8],
P4[15:8],P3[7:0],P4[7:0],P2[23:16]

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).

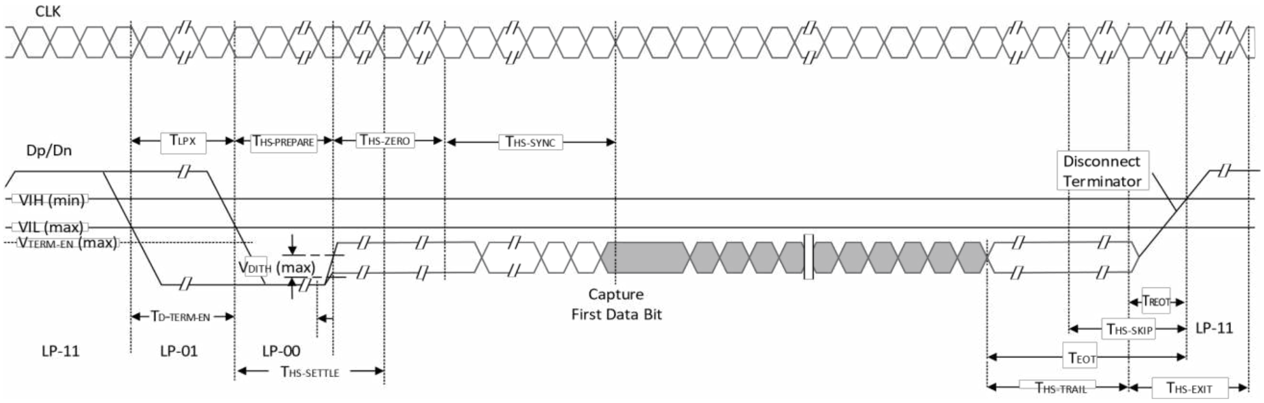


Figure 2.13. MIPI D-PHY High Speed Transmission Timing Diagram

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.

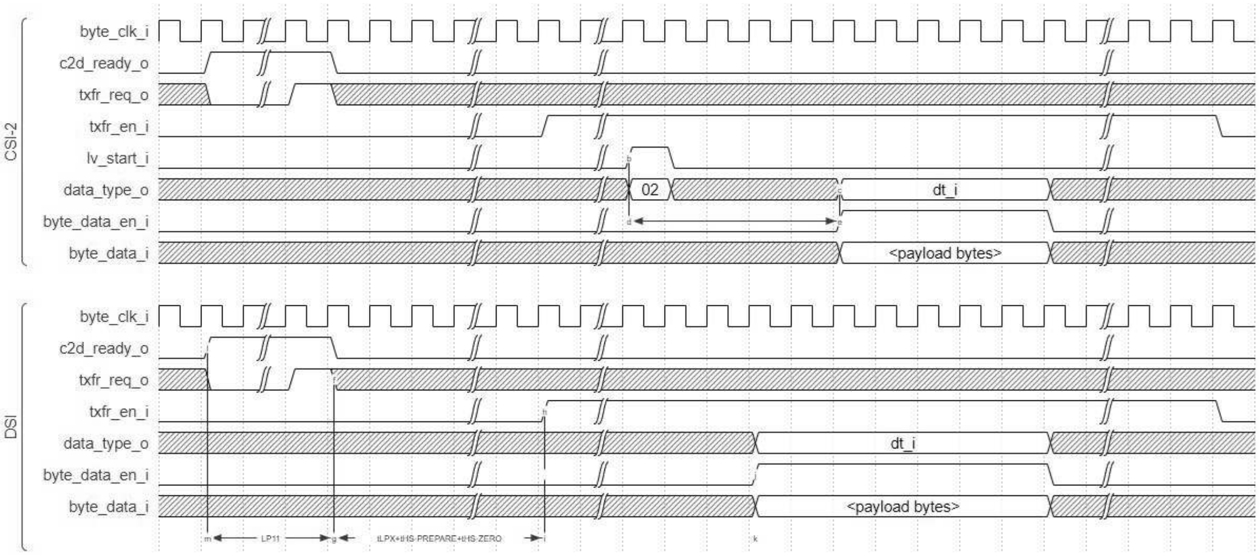


Figure 2.14. Handshake Signals Timing Diagram

3. IP Generation, Simulation, and Validation

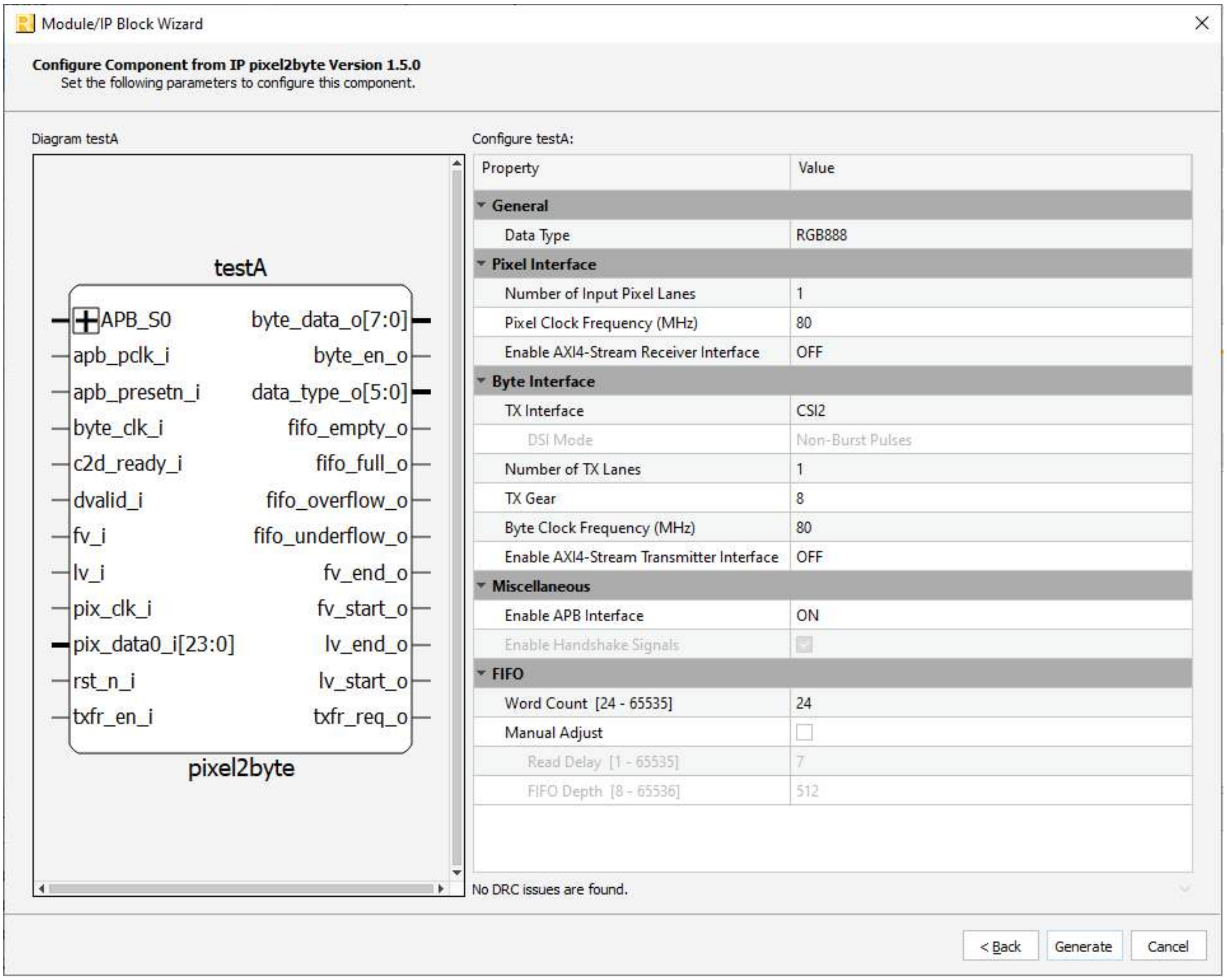
This chapter provides information on how to generate and synthesize Pixel-to-Byte Converter IP Core using Lattice Radiant software and how to run simulation. For more on Lattice Radiant software, refer to the Lattice Radiant Software User Guide and relevant Lattice tutorials.

3.1. Generating the IP

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.



Module/IP Block Wizard

Configure Component from IP pixel2byte Version 1.5.0
Set the following parameters to configure this component.

Diagram testA

Configure testA:

Property	Value
General	
Data Type	RGB888
Pixel Interface	
Number of Input Pixel Lanes	1
Pixel Clock Frequency (MHz)	80
Enable AXI4-Stream Receiver Interface	OFF
Byte Interface	
TX Interface	CSI2
DSI Mode	Non-Burst Pulses
Number of TX Lanes	1
TX Gear	8
Byte Clock Frequency (MHz)	80
Enable AXI4-Stream Transmitter Interface	OFF
Miscellaneous	
Enable APB Interface	ON
Enable Handshake Signals	☑
FIFO	
Word Count [24 - 65535]	24
Manual Adjust	□
Read Delay [1 - 65535]	7
FIFO Depth [8 - 65536]	512

No DRC issues are found.

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.

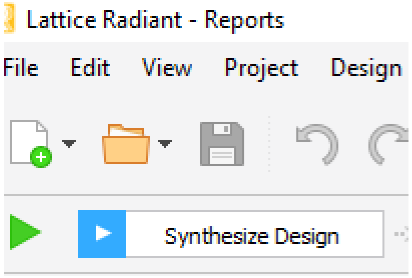
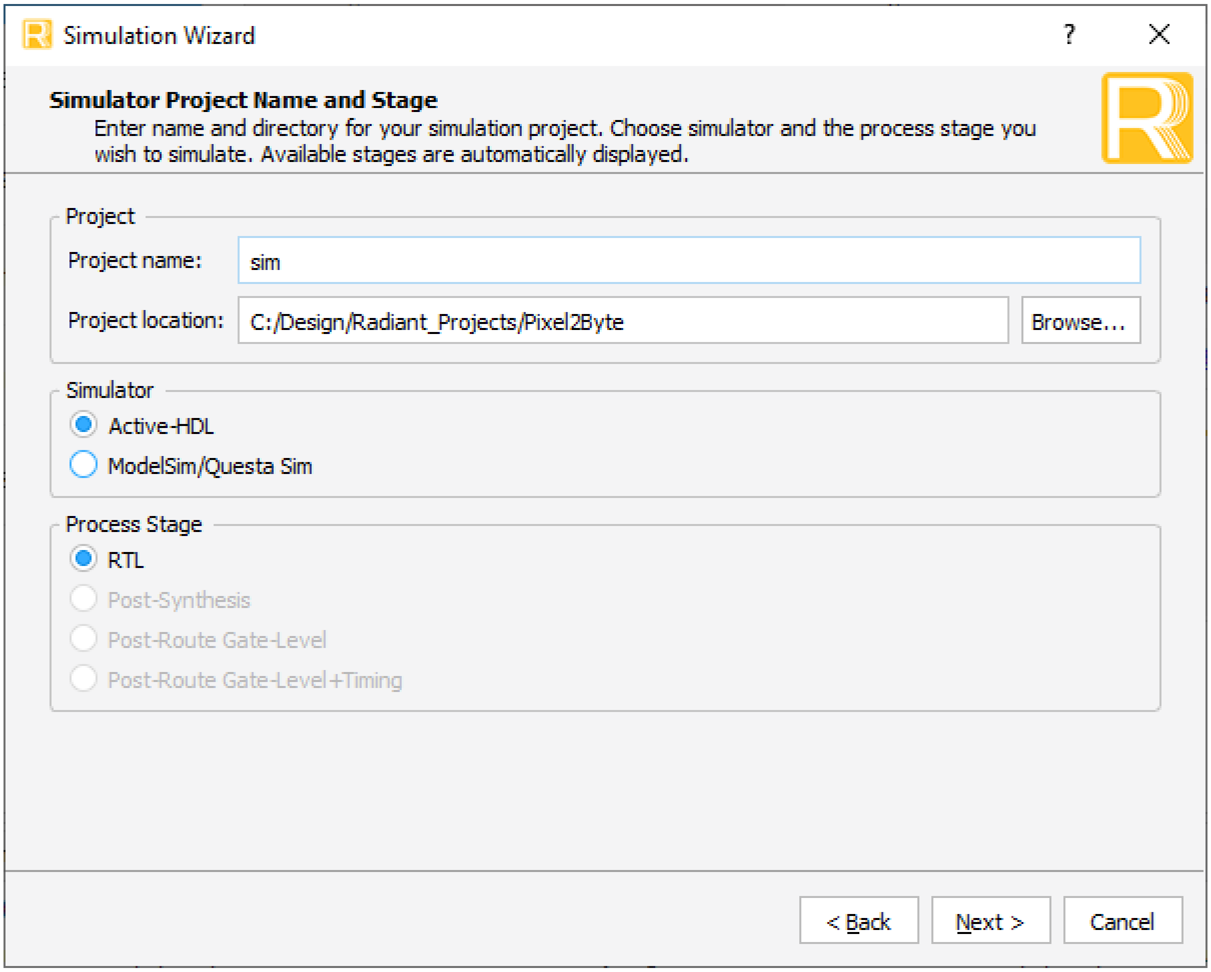


Figure 3.2. Synthesizing the Design

3.2. Running Functional Simulation

To run Verilog simulation:

1. Click the button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 3.3.



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: sim

Project location: C:/Design/Radiant_Projects/Pixel2Byte

Simulator

● Active-HDL

○ ModelSim/Questa Sim

Process Stage

● RTL

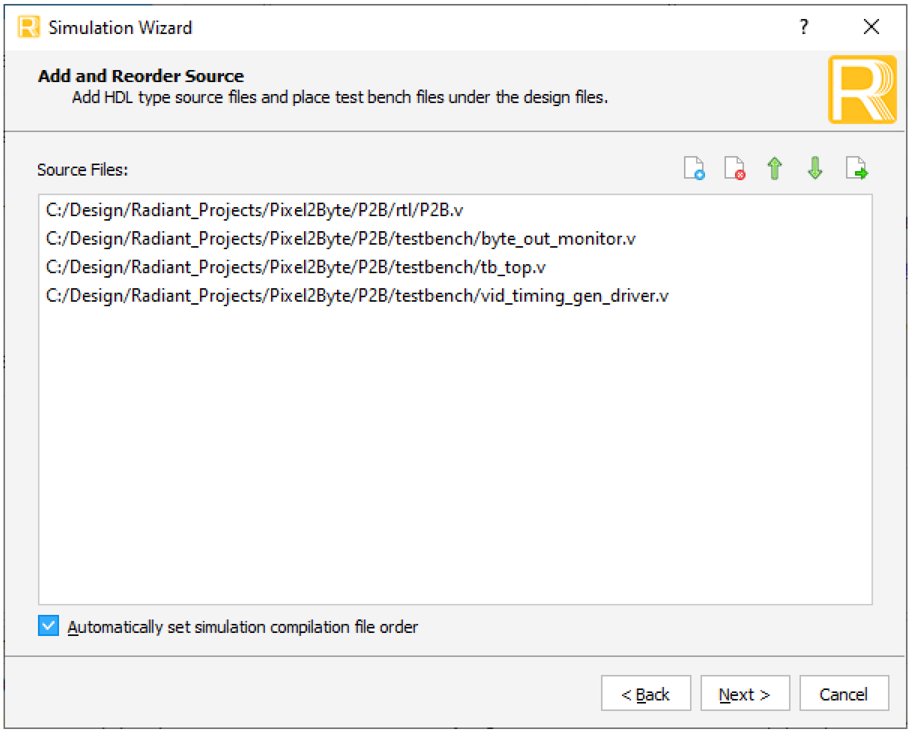
○ Post-Synthesis

○ Post-Route Gate-Level

○ Post-Route Gate-Level+Timing

Figure 3.3. Simulation Wizard

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


Simulation Wizard
?
Add and Reorder Source
Add HDL type source files and place test bench files under the design files.
Source Files:
< Back
Next >
Cancel

C:/Design/Radiant_Projects/Pixel2Byte/P2B/rtl/P2B.v
C:/Design/Radiant_Projects/Pixel2Byte/P2B/testbench/byte_out_monitor.v
C:/Design/Radiant_Projects/Pixel2Byte/P2B/testbench/tb_top.v
C:/Design/Radiant_Projects/Pixel2Byte/P2B/testbench/vid_timing_gen_driver.v

☑ Automatically set simulation compilation file order

Figure 3.4. Adding and Reordering Source

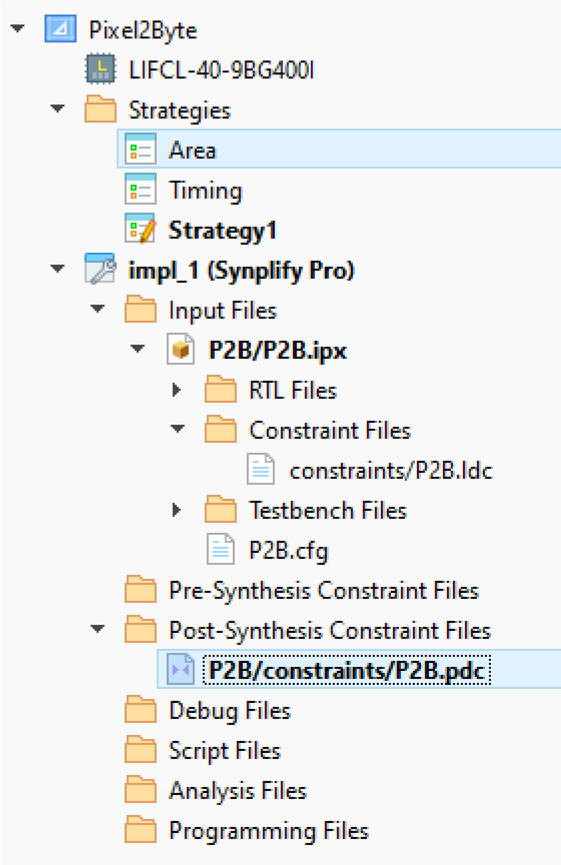
3. Add the *tb_top.v file* from the *testbench* directory if not included.
4. Click **Next** to run the simulation.

3.3. Constraining the IP

The Pixel to Byte Converter IP for Radiant has a default pre-synthesis constraints file *.ldc located in <ip_instance_path>/constraints/<instance_name>.ldc that shows more details on how to constrain the IP. There is also a Post-Synthesis Constraints (*.pdc) file provided as shown in Figure 3.5 located in the <ip_instance_path>/eval/<instance_name>pdc.

To properly constrain the pixel to byte IP 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.



- Pixel2Byte
  - LIFCL-40-9BG400I
  - Strategies
    - Area
    - Timing
    - **Strategy1**
  - **impl_1 (Synplify Pro)**
    - Input Files
      - **P2B/P2B.ipx**
        - RTL Files
        - Constraint Files
          - constraints/P2B.ldc
        - Testbench Files
        - P2B.cfg
    - Pre-Synthesis Constraint Files
    - Post-Synthesis Constraint Files
      - **P2B/constraints/P2B.pdc**
    - Debug Files
    - Script Files
    - Analysis Files
    - Programming Files

Figure 3.5. Adding Constraint

3.4. IP Evaluation

The IP Core supports Lattice's IP evaluation capability when used with Lattice Avant. This makes it possible to create versions of the IP core that operate in hardware for a limited period (approximately four hours) without requiring the purchase of an IP license. The IP evaluation capability may be enabled/disabled in the **Strategies Dialog Box** which is disabled by default. To change this setting, go to Strategies>Strategy1 (active strategy) > Bitstream.

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
- PIXEL-BYTE-CPNX-U – Pixel to Byte Converter for CertusPro-NX – Single Design License
- PIXEL-BYTE-CPNX-UT – Pixel to Byte Converter for CertusPro-NX – Site License
- PIXEL-BYTE-AVE-U – CSI-2/DSI DPHY Transmitter for Avant-E – Single Design License
- PIXEL-BYTE-AVE-UT – CSI-2/DSI DPHY Transmitter for Avant-E – Site License
- PIXEL-BYTE-AVE-US – CSI-2/DSI DPHY Transmitter for Avant-E – One Year Subscription 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.

Table A.1. Device and Tool Tested

	Value
Software Version	Radiant 2022.1 production build
Device Used	LAV-AT-500E-3LFG1156C
Performance Grade	3
Synthesis Tool	Synplify Pro (R) Q-2022.01LR, Build 45R, November 8, 2022

Table A.2. Resource Utilization

Device	LUTs	Register	sysMEM EBRs	Programmable I/O
Default	455	259	1	4
DSI, RGB666, Number of TX Lanes 2	553	296	1	12
CSI-2, RGB888, Number of TX Lanes 4	503	372	1	28
CSI-2, RAW8, Number of TX Lanes 4	394	319	1	28
DSI, RGB888, Number of TX Lanes 4, Number of Input Pixel Per Clock 4, TX Gear 16	689	562	2	60
CSI-2, RGB888, Number of TX Lanes 2, TX Gear 16	503	372	1	28
CSI-2, RAW10, Number of TX Lanes 2, Number of Input Pixel Per Clock 2	471	281	1	12
CSI-2, RAW14, Numbber of TX Lanes 4, Number of Input Pixel Per Clock 2, TX Gear 16	913	696	2	60

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*.

References

- CrossLink-NX FPGA web page at www.latticesemi.com
- Certus-NX FPGA web page at www.latticesemi.com
- CertusPro-NX FPGA web page at www.latticesemi.com
- Avant-E Web Page at www.latticesemi.com

Technical Support Assistance

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

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

Revision History

Document Revision 1.5, December 2022

Section	Change Summary
Introduction	• Updated Table 1.1. Quick Facts. • Revised Supported FPGA Families • Revised Targeted Devices
Functional Description	• Updated Table 2.2. Supported Configurations and Table 2.3. Attributes Table. • Added Table 2.4. Word Count Restriction. • Updated Section 2.4.1. Clock, Rest and Initialization.
IP Generation, Simulation, and Validation	• Updated the title of the Section 3. • Deleted Licensing the IP Section. • Updated Figure 3.1. Configure Block of Pixel-to-Byte Converter. • Updated the title of Section 3.1 from Generation and Synthesis to Generating the IP. • Updated the title of Section 3.2 Functional Simulation to Running Functional Simulation. • Updated Section 3.3. Constraining the IP. • Added Section 3.4. IP Evaluation.
Ordering Part Number	Added Avant-E part numbers.
Appendix A. Resource Utilization	Updated Table A.1. Device and Tool Tested and Table A.2. Resource Utilization.
References	Added Avant webpage link in References section.

Document Revision 1.4, June 2021

Section	Change Summary
Introduction	• Removed *This document is for Pixel-to-Byte Converter IP design version 1.1*. • Added CertusPro-NX in first paragraph. • Updated Table 1.1. Quick Facts. • Revised Supported FPGA Families • Revised Targeted Devices • Revised Lattice Implementation to IP Core v1.3.x - Lattice Radiant software 3.0.
Functional Description	• Updated pix_data1_i1 description in Table 2.1. Pixel-to-Byte Converter IP Ports. • Updates values in Table 2.2. Supported Configurations.
IP Generation and Evaluation	Updated Figure 3.1. Configure Block of Pixel-to-Byte Converter.
Ordering Part Number	Added part numbers.
Appendix A. Resource Utilization	Updated Table A.2. Resource Utilization.
Appendix B. Limitation	Removed this section.
References	Added reference to the CertusPro-NX web page.

Document Revision 1.3, December 2020

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

Document Revision 1.2, August 2020

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

Document Revision 1.1, February 2020

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

Document Revision 1.0, December 2019

Section	Change Summary
All	Initial release

LATTICE SEMICONDUCTOR®

www.latticesemi.com