Byte-to-Pixel Converter IP

User Guide

FPGA-IPUG-02027-1.4

September 2021

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

1. Introduction

The Lattice Semiconductor Byte-to-Pixel Converter IP converts parallel data, such as output of D-PHY receiver module, to pixel format. Other than video packet conversion, this module also generates camera/video control signals in the pixel domain, based on CSI-2 or DSI synchronization packets.

Bridging applications are very popular in the market today due to the increasing demand for better displays. One common application interface is the Mobile Industry Processor Interface (MIPI®) D-PHY. It was developed primarily to support camera and display interconnections in mobile devices, and has today become the industry primary high-speed PHY solution for these applications in smartphones. It is typically used in conjunction with MIPI Camera Serial Interface–2 (CSI-2) and MIPI Display Interface (DSI) protocol Specifications. It meets the requirements of low-power, low noise generation, and high noise immunity that mobile phone designs demand.

This document is for Byte-to-Pixel Converter IP design version 1.3.

1.1. Quick Facts

Table 1.1 provides quick facts about the Byte-to-Pixel Converter IP for CrossLink™ and CrossLinkPlus™ devices.

Table 1.1. Byte-to-Pixel Converter IP Quick Facts

		Byte-to-Pixel Converter IP Configuration			
		Gear 16, 4-Lane, RGB888, 2-Pixel Output	**Gear 8, 4-Lane, RGB888, 2-Pixel Output**	**Gear 16, 2-Lane, RGB888, 1-Pixel Output**	**Gear 8, 2-Lane, RGB888, 1-Pixel Output**
IP Requirements	FPGA Families Supported	CrossLink/CrossLinkPlus			
Resource Utilization	Targeted Device	LIF-MD6000-6MG81I			
	LUTs	511	289	259	238
	sysMEM™ EBRs	6	3	3	2
	Registers	548	333	305	264
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.11 SP1			
	Synthesis	Lattice Synthesis Engine			
		Synplify Pro® N-2018.03L-SP1-1			
	Simulation	Aldec® Active HDL™ 10.5 Lattice Edition			

1.2. Features

1.2.1. Supported

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

- Supports the ff. MIPI DSI compatible video formats
  - RGB888
  - RGB666 packed
  - RGB666 loosely packed
- Supports the ff. MIPI CSI-2 compatible video formats
  - RGB888
  - RAW10
  - RAW12
  - RAW8
  - YUV420 8-bit
  - YUV422 8-bit
  - Legacy YUV420 8-bit
  - YUV420 8-bit CSPS
  - YUV420 10-bit
  - YUV420 10-bit CSPS
  - YUV422 10-bit
- Supports 1-, 2-, or 4-lane inputs
- Supports 8-bit (gear 8) or 16-bit (gear 16) inputs per lane
- Supports one, two or four output pixels per pixel clock cycle
- Supports DSI Non-Burst Mode with Sync Events, Non-Burst Mode with Sync Pulses, and Burst Mode

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 bidirectional input/output signals

2. Functional Description

The Byte-to-Pixel Converter IP converts DSI or CSI-2 video payload packets to standard pixel data format.

The input interface contains the control information extracted from the D-PHY packet header fields. For 4-lane, gear 16 configuration, it is possible that two header packets are received in a single byte clock cycle. Thus, a second set of input control signals are provided.

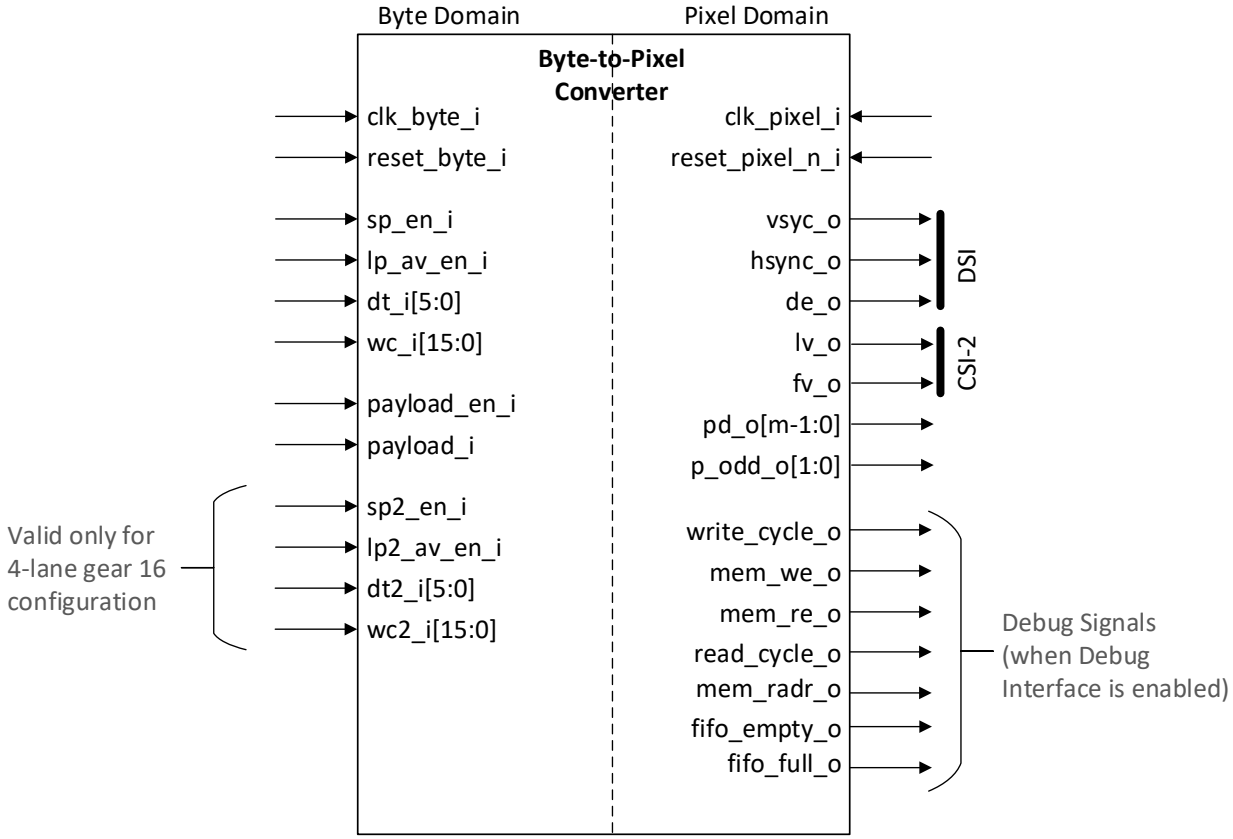

Figure 2.1. Top-level Block Diagram

Table 2.1. Byte-to-Pixel Converter IP Pin Function Description

Port Name	Direction	Function Description
Clocks and Reset		
clk_byte_i	I	Clock for the Rx byte clock domain logic (input).
reset_byte_n_i	I	Active low signal to reset the logic in the clk_byte_i domain.
clk_pixel_i	I	Clock for the pixel domain logic (output).
reset_pixel_n_i		Active low signal to reset the logic in the clk_pixel_i domain.
Byte Domain Inputs		
sp_en_i	I	Active high pulse to indicate a valid short packet in the Rx side.
lp_av_en_i	I	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]	I	Data type field of the D-PHY Rx header packet.
wc_i[15:0]	I	Word Count field of the D-PHY Rx header packet.
payload_i	I	This is the active video data stream. The width of the data bus depends on the gearing and number of D-PHY Rx lanes.
payload_en_i	I	Active high payload valid indicator.
sp2_en_i	I	This is valid only for gear 16, 4-lane configuration. Active high pulse to indicate a reception of a

Port Name	Direction	Function Description
		second valid short packet in the same byte clock cycle.
lp2_av_en_i	I	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]	I	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]	I	This is valid only for gear 16, 4-lane configuration. Word Count field of the second D-PHY RX header packet.
Pixel Domain Outputs		
vsync_o	O	VSYNC signal for DSI. Active high if CTRL_POL parameter is POSITIVE. Otherwise, this is active low.
hsync_o	O	HSYNC signal for DSI. Active high if CTRL_POL parameter is POSITIVE. Otherwise, this is active low.
fv_o	O	Frame Valid signal for CSI-2. Active high if CTRL_POL parameter is POSITIVE. Otherwise, this is active low.
lv_o	O	Line Valid signal for CSI-2. Active high if CTRL_POL parameter is POSITIVE. Otherwise, this is active low.
de_o	O	Data Enable signal for DSI. Active high if CTRL_POL parameter is POSITIVE. Otherwise, this is active low.
pd_o [pixel_width-1:0]	O	Pixel data output. The pixel_width may be 8, 10, 12, 18, 24, 36, 48, 72, or 96 bits. 8 (8-bit pixel) 10 (10-bit pixel) 12 (12-bit pixel) 16 (8-bit pixel x2) 18 (18-bit pixel) 20 (10-bit pixel x2) 24 (24-bit pixel or 12-bit pixel x2) 36 (18-bit pixel x2) 48 (24-bit pixel x2) 72 (18-bit pixel x4) 96 (24-bit pixel x4) In case of multiple output pixels per pixel clock cycle, the first received pixel is placed in the lower bits (LSB) of the data bus.
p_odd_o[1:0]	O	Module 4 of the current pixel count. This may be used to indicate the valid pixels for the last valid pixel data cycle in case of multiple pixel outputs per pixel clock cycle. 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.
Miscellaneous		
write_cycle_o[3:0]	O	Payload data write cycle (debug only).
mem_we_o	O	Payload data Write Enable, active high (debug only).
mem_re_o	O	Payload data Read Enable, active high (debug only).
read_cycle_o[1:0]	O	Pixel data read cycle (debug only).
mem_radr_o[2:0]	O	Pixel data read address (debug only).

2.1. Interface and Timing Diagram

Figure 2.2 shows the timing between input and output for DSI.

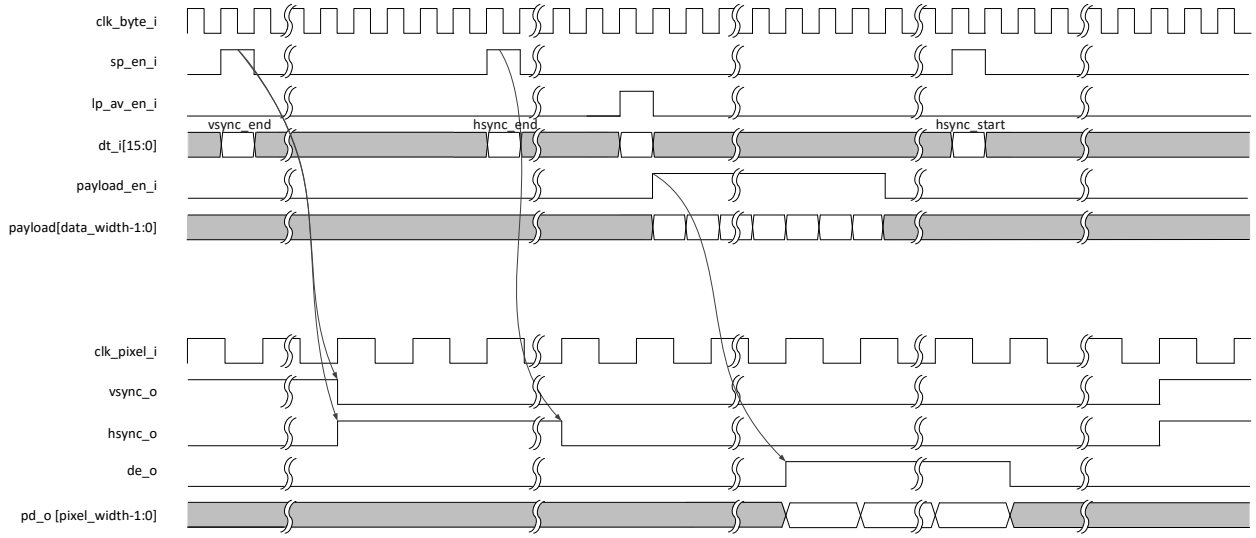

Figure 2.2. Interface Timing Diagram between Inputs and Outputs for DSI

Reception of a VSYNC start packet triggers the assertion of both hsync_o and vsync_o signals. On the other hand, VSYNC end packets trigger the deassertion of the vsync_o signal and the assertion of hsync_o signal. In both cases, an HSYNC end packets is expected next. This is in exception with Non-Burst Mode with Sync Events because VSYNC/HSYNC end packet doesn't exist in this condition.

Figure 2.3 shows the timing between input and output for CSI-2.

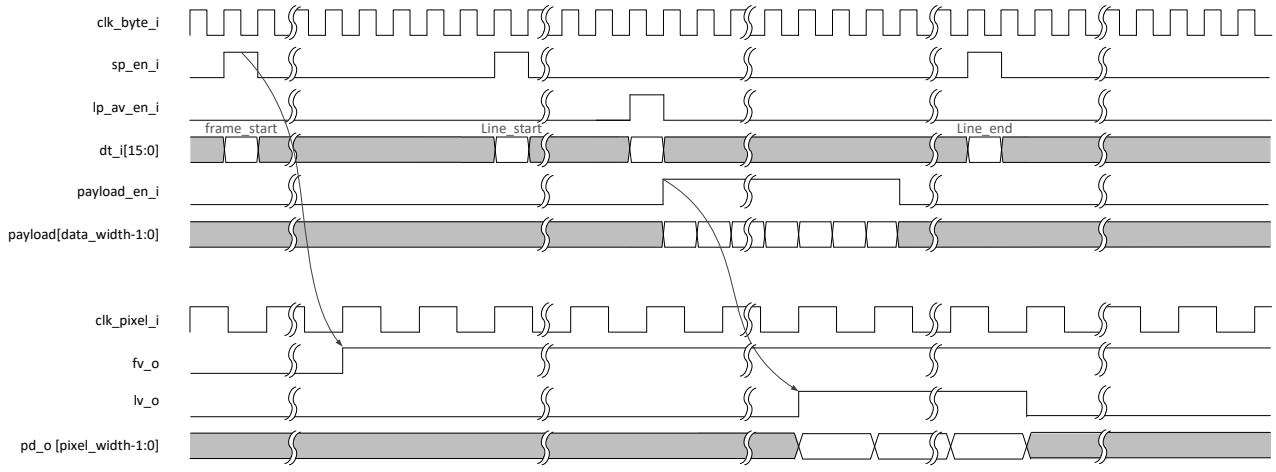

Figure 2.3. Interface Timing Diagram between Inputs and Outputs for CSI-2

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.

2.1.1. Input Timing

Figure 2.4 shows the timing diagram of the interface at the receiver side. The assertion of the payload_en_i 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.

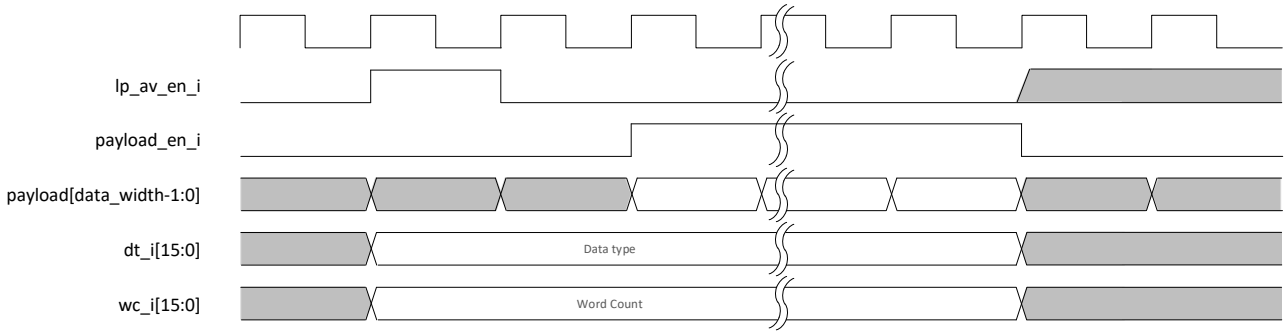

Figure 2.4. Input Interface Timing Diagram

2.1.2. Output Timing

The output timing from a DSI input is shown in Figure 2.5, while the timing from a CSI-2 input is shown in Figure 2.6.

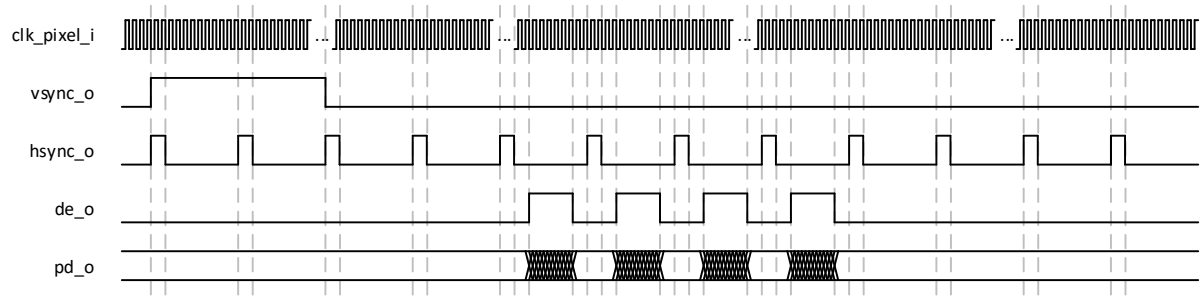

Figure 2.5. Output Timing Diagram from a DSI Input

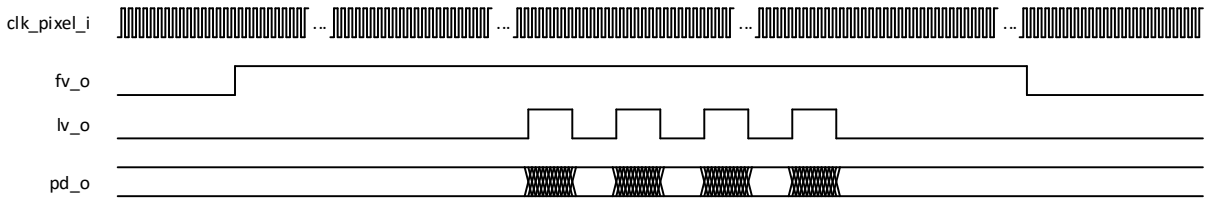

Figure 2.6. Output Timing Diagram from a CSI-2 Input

2.2. Pixel and Byte Count Restriction

The D-PHY Specification handles its data in byte as a minimum unit. Some data types therefore have restrictions on the number of pixels per line.

Table 2.2. Pixel and Byte Count Restriction

Data Type	Pixel Count Restriction	Byte Count Restriction
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 (odd line)	multiple of 2	multiple of 2
YUV420 8-bit (even line)	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 (odd line)	multiple of 4	multiple of 5
YUV420 10-bit (even line)	multiple of 4	multiple of 10
YUV422 10-bit	multiple of 2	multiple of 5
RAW12	multiple of 2	multiple of 3

In most cases, the received data are stuffed from LSB to MSB according to the output data width.

In the case of RAW10 and RAW12, MSB 8 bits and remaining LSB 2 or 4 bits are packed separately so that special data handlings are required. This also applies to YUV420 10 bit and YUV422 10 bit.

Note that RGB data order is different between DSI and CSI-2 (RGB-RGB in DSI, while it is BGR-BGR in CSI-2).

2.3. Supported Configurations

The following tables list the supported configurations of the Byte-to-Pixel Converter.

2.3.1. Supported Configurations for DSI

Table 2.3. Supported Configurations for DSI

Number of Output Pixels	D-PHY Lanes	RX Gearing	Data Type
1 output pixel	1 lane	8	RGB666, RGB666 loosely packed, RGB888
		16	RGB666, RGB666 loosely packed, RGB888
	2 lanes	8	RGB666, RGB666 loosely packed, RGB888
		16	RGB666, RGB666 loosely packed, RGB888
	4 lanes	8	RGB666, RGB666 loosely packed, RGB888
2 output pixels	2 lanes	16	RGB666, RGB666 loosely packed, RGB888
	4 lanes	8	RGB666, RGB666 loosely packed, RGB888
		16	RGB666, RGB666 loosely packed, RGB888
4 output pixels	4 lanes	16	RGB666, RGB666 loosely packed, RGB888

2.3.2. Supported Configurations for CSI-2

Table 2.4. Supported Configurations for CSI-2

Number of Output Pixels	D-PHY Lanes	RX Gearing	Data Type
1 output pixel	1 lane	8	8-bit*, 10-bit*, RAW12, RGB888
		16	10-bit, RAW12, RGB888
	2 lanes	8	8-bit, 10-bit, RAW12, RGB888
		16	RGB888
	4 lanes	8	8-bit, 10-bit, RAW12, RGB888
2 output pixels	1 lane	16	8-bit
	2 lanes	8	8-bit, 10-bit
		16	8-bit, 10-bit, RAW12, RGB888
	4 lanes	8	8-bit, 10-bit, RAW12, RGB888
		16	RGB888

*__Note__: Supported 8-bit CSI-2 data types are RAW8, YUV420 8-bit, Legacy YUV420 8-bit, YUV420 8-bit CSPS and YUV422 8-bit. Supported 10-bit CSI-2 data types are RAW10, YUV420 10-bit, YUV420 10-bit CSPS and YUV422 10-bit.

2.4. FIFO Implementation

A FIFO is used to synchronize the incoming D-PHY data bytes to the pixel clock domain. The FIFO depth is already computed based on the design configuration. 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 how many pixels are grouped together and written to the same FIFO address. For burst mode, the FIFO depth is fixed to 1024 to accommodate the entire video line.

Figure 2.7 through Figure 2.23 show FIFO write cycle and write data arrangement in each write cycle for various cases.

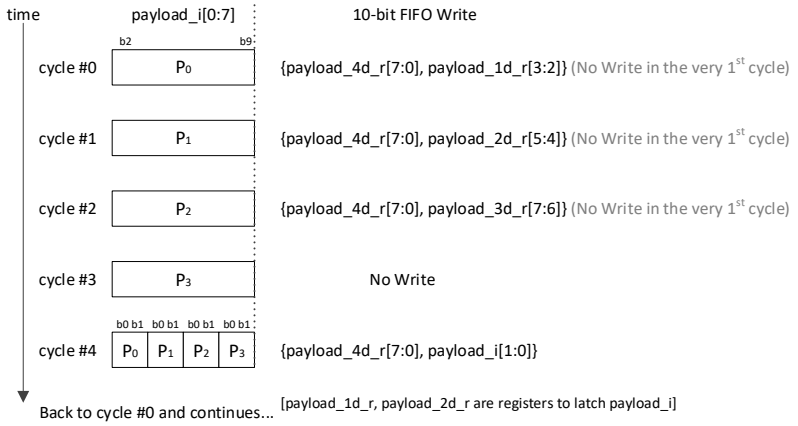

Figure 2.7. FIFO Write Data and Cycle Pattern for 1 Lane CSI-2 10-bit Data Types

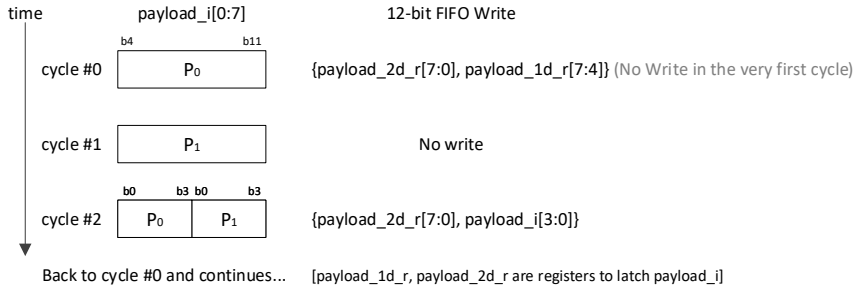

Figure 2.8. FIFO Write Data and Cycle Pattern for 1 Lane CSI-2 RAW12

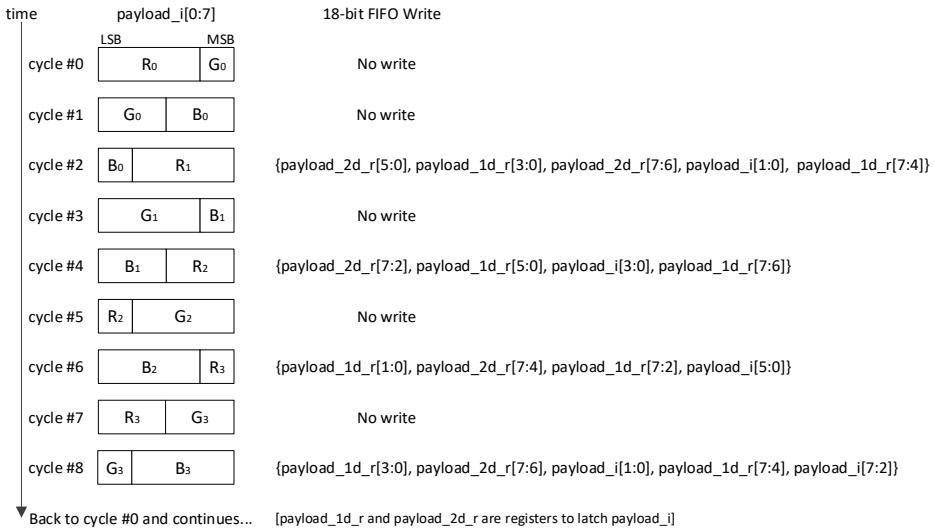

Figure 2.9. FIFO Write Data and Cycle Pattern for 1 Lane DSI RGB666

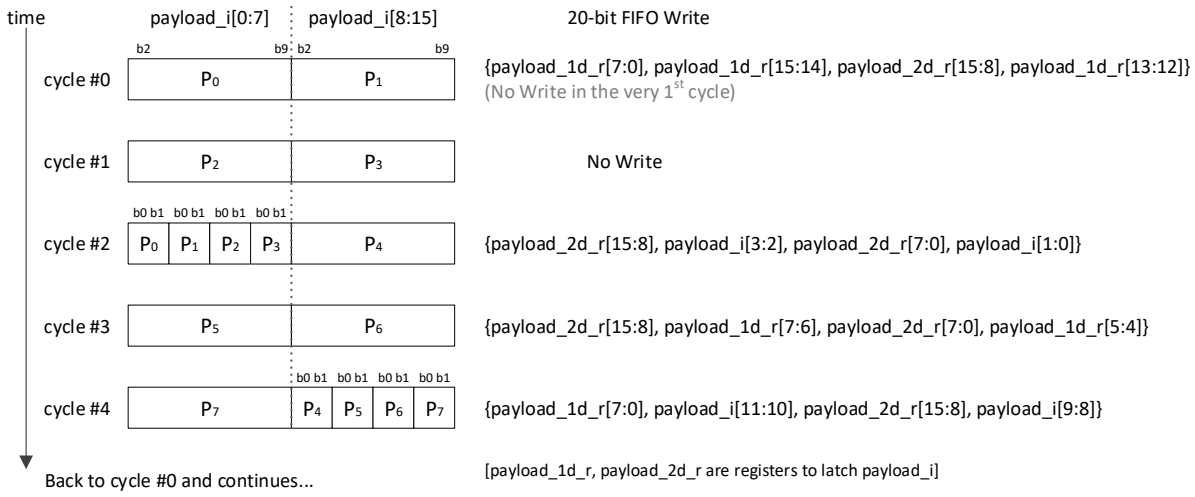

Figure 2.10. FIFO Write Data and Cycle Pattern for 2 Lane CSI-2 10-bit Data Types

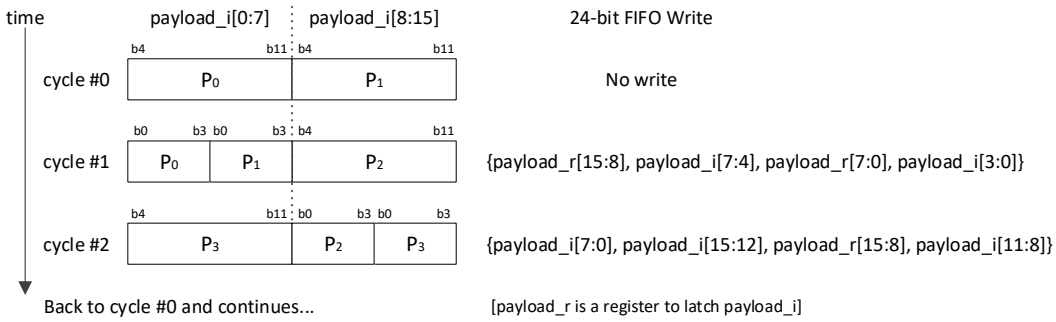

Figure 2.11. FIFO Write Data and Cycle Pattern for 2 Lane CSI-2 RAW12

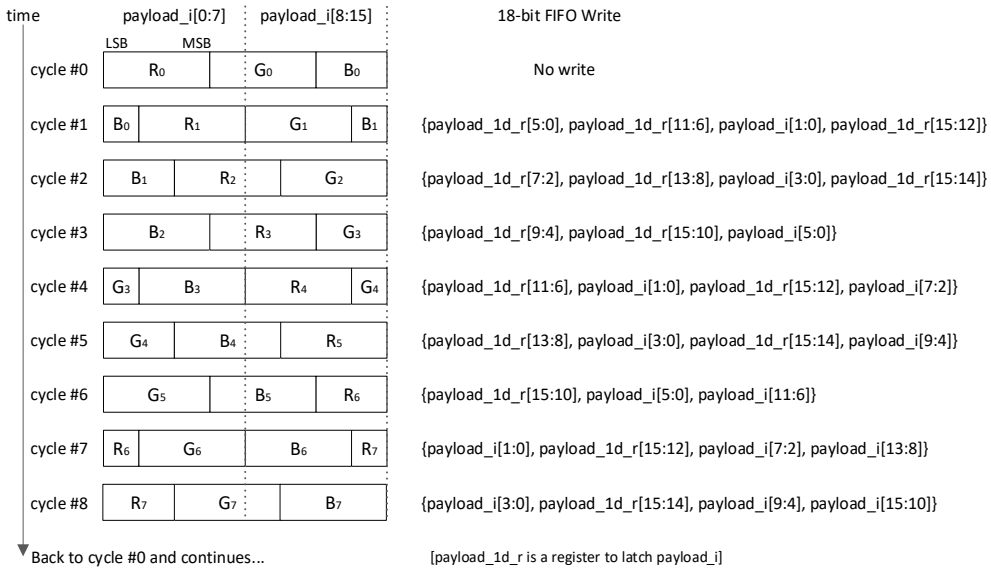

Figure 2.12. FIFO Write Data and Cycle Pattern for 2 Lane DSI RGB666

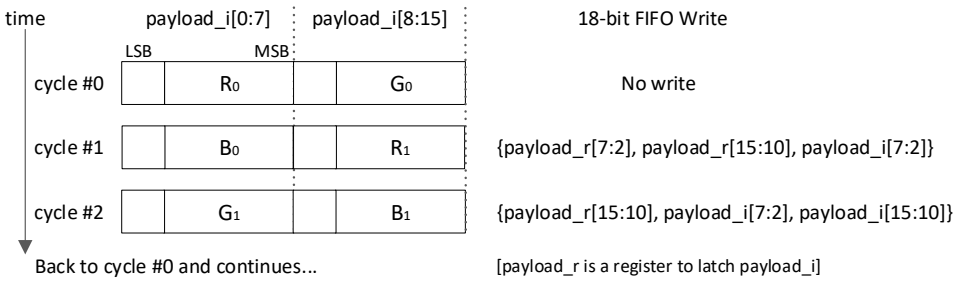

Figure 2.13. FIFO Write Data and Cycle Pattern for 2 Lane DSI RGB666 Loosely packed

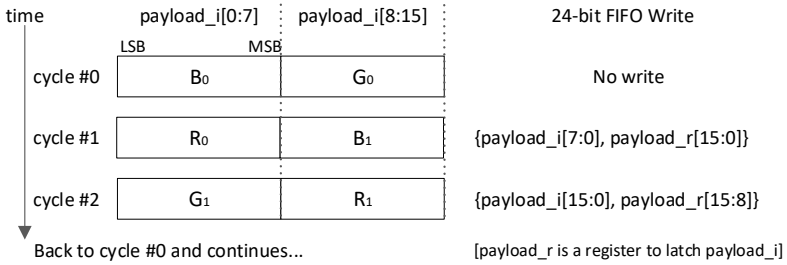

Figure 2.14. FIFO Write Data and Cycle Pattern for 2 Lane CSI-2 RGB888

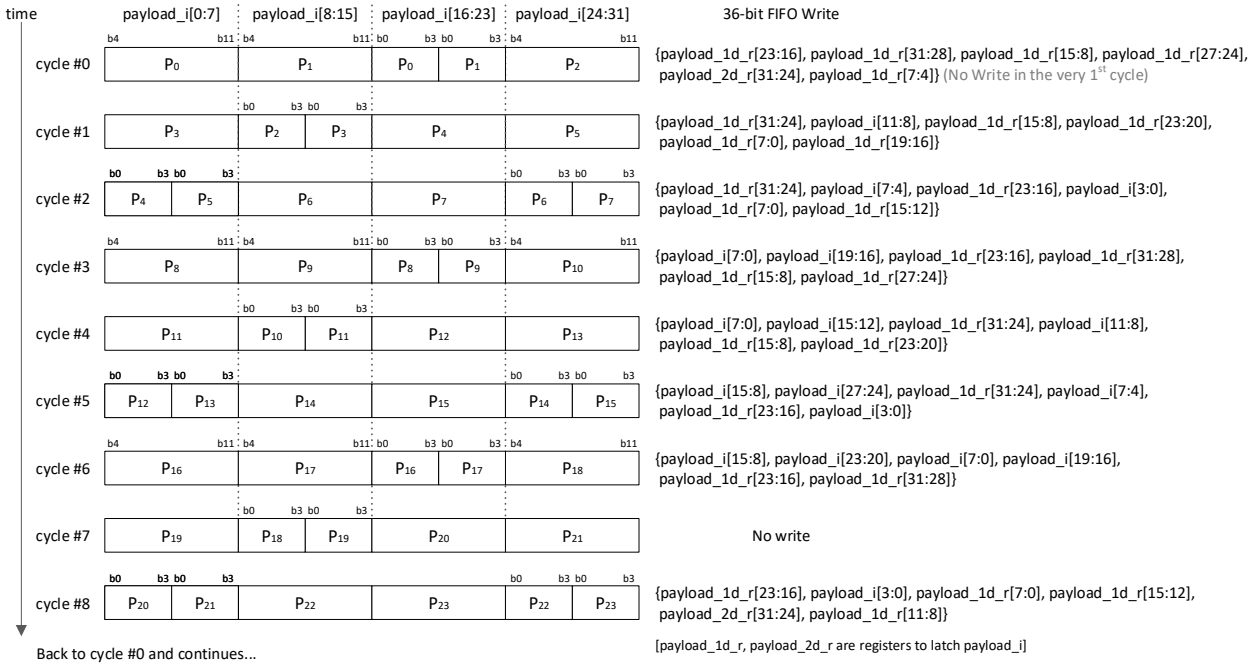

Figure 2.15. FIFO Write Data and Cycle Pattern for 4 Lane CSI2 RAW12

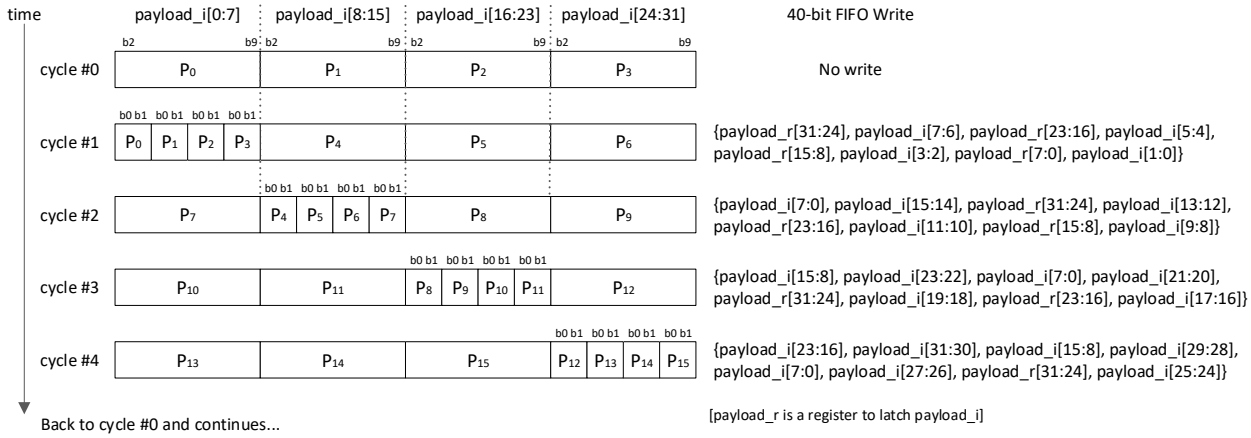

Figure 2.16. FIFO Write Data and Cycle Pattern for 4 Lane CSI-2 RAW10

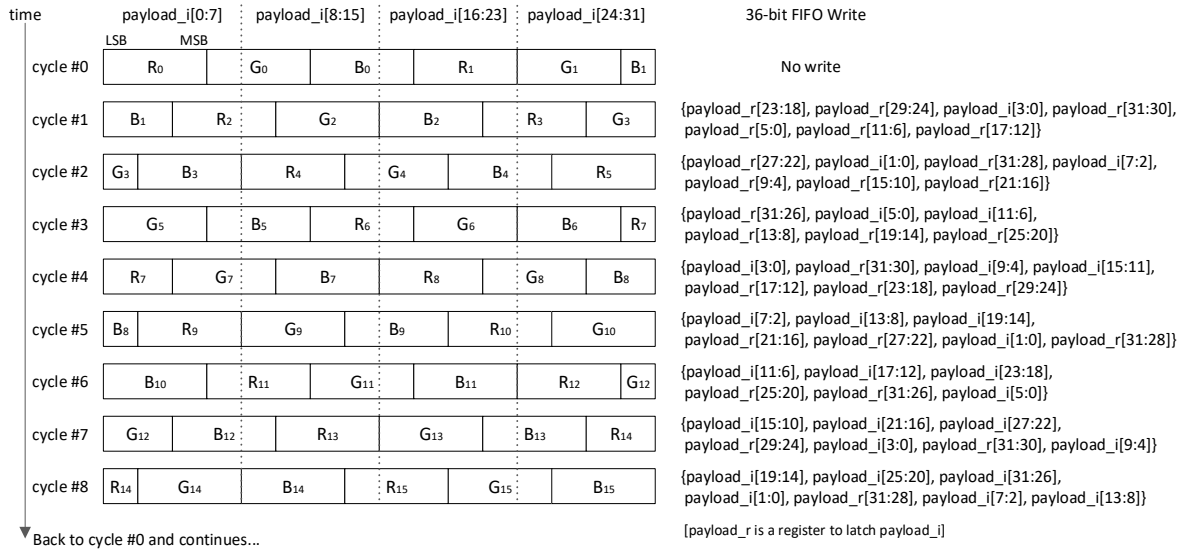

Figure 2.17. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB666

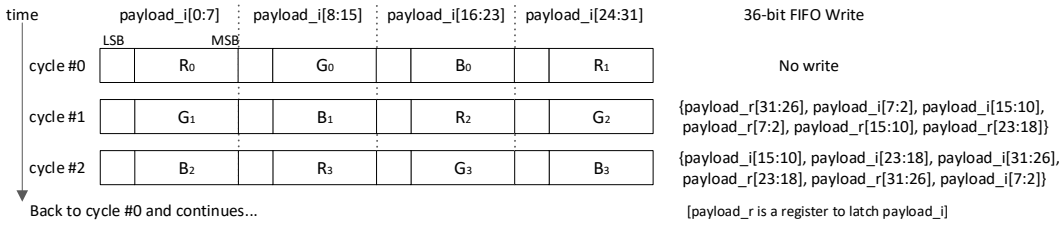

Figure 2.18. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB666 Loosely packed

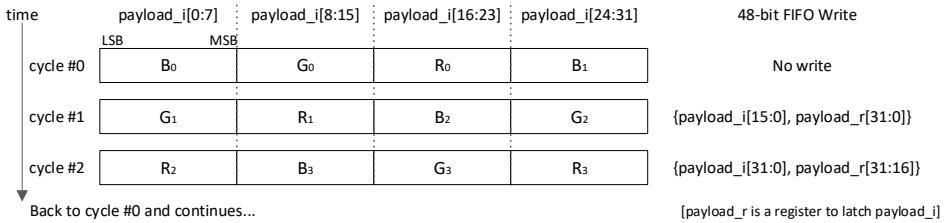

Figure 2.19. FIFO Write Data and Cycle Pattern for 4 Lane CSI-2 RGB888

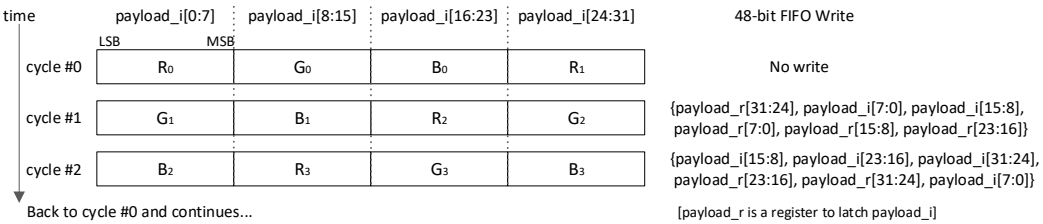

Figure 2.20. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB888

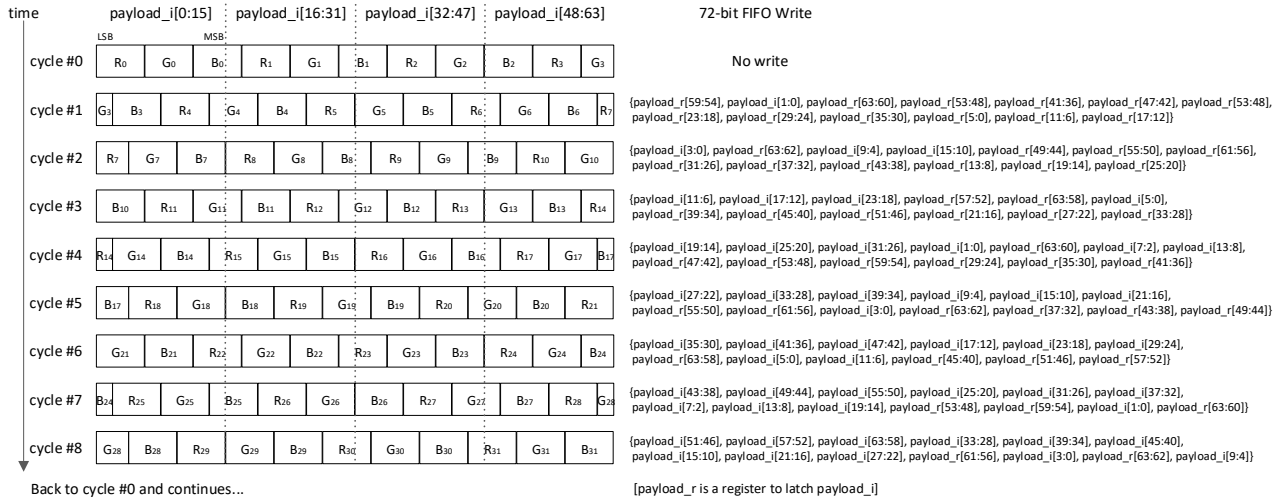

Figure 2.21. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB666 w/ Gear16

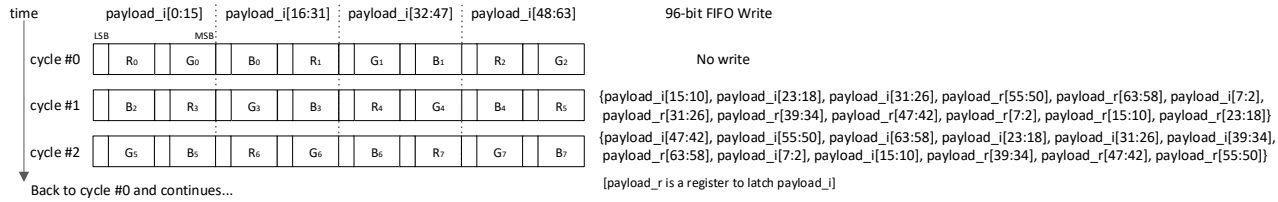

Figure 2.22. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB666 Loosely packed w/ Gear16

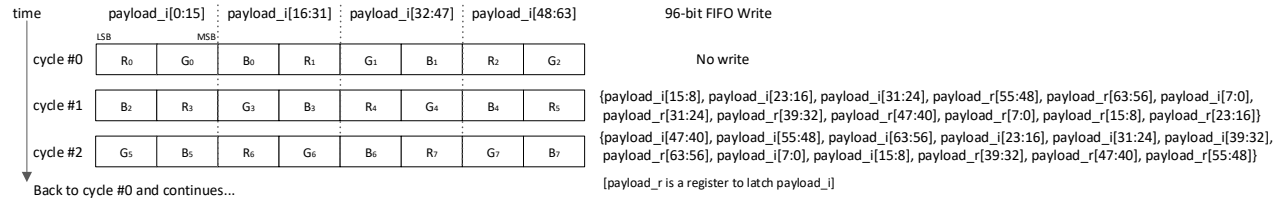

Figure 2.23. FIFO Write Data and Cycle Pattern for 4 Lane DSI RGB888 w/ Gear16

2.5. Clock, Reset and Initialization

2.5.1. Reset and Initialization

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

2.5.2. Clock Domains

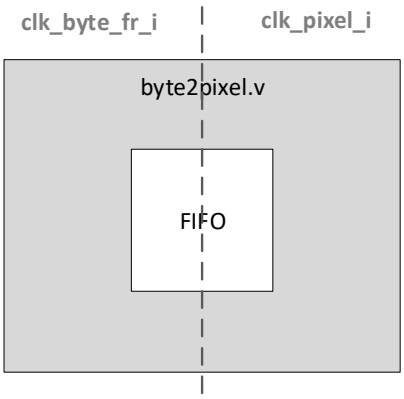

Figure 2.24. Clock Domain Crossing Block Diagram

Table 2.5. Clock Domain Crossing

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

For *CSI-2* and *DSI Non-Burst Mode*, the relationship between the two clocks is given by the equation below:

$$\frac{byte\ clock\ frequency}{pixel\ clock\ frequency} = \frac{(bits\ per\ pixel * number\ of\ output\ pixels)}{(number\ of\ D-PHY\ lanes * RX\ gearing)} = \frac{(pd\ bus\ width * number\ of\ tx\ channels)}{(number\ of\ D-PHY\ lanes * RX\ gearing)}$$

where *RX gearing* is 8 or 16, and the *number of output pixels* can be 1, 2 or 4.

For RGB666 Loosely Packed, 24 bits per pixel should be used when computing for the required clock frequencies.

Due to the fixed FIFO depth and width of the design as described in FIFO Implementation section, the relationship between the pixel and the byte clock must meet the requirement described by the equation above.

For *DSI Burst Mode*, the pixel clock is expected to be slower than the byte clock. A FIFO with a fixed depth of 1024 is used to be able to accommodate the entire video line.

3. Configuration Settings

Table 3.1 lists the parameters used to generate the Byte-to-Pixel Converter IP. All parameters are either set automatically or input in the user interface during the Byte-to-Pixel Converter IP generation.

Table 3.1. Byte-to-Pixel Converter IP User Interface Parameter Settings

Parameters	Attribute	Options	Description
RX Interface	User-Input	DSI or CSI-2	Selects between MIPI display or camera interface.
DSI Mode	User-Input	Non-Burst Mode with Sync Events Non-Burst Mode with Sync Pulses Burst Mode	This option is valid only when the RX Interface is DSI. This specifies which Video Mode will be used.
Number of Rx Lanes	User-Input	1, 2 or 4	Sets the number of MIPI D-PHY Rx Lanes.
Rx Gear	User-Input	8 or 16	Sets number of bits per input lane. This parameter, together with the number of Rx lanes, determines the data width of the input data in the byte clock domain.
Number of Pixels Per Clock	User-Input	1, 2 or 4	Specifies the number of output pixels per pixel clock.
Data type	User-Input	DSI data types: RGB666 RGB666_LOOSE RGB888 CSI-2 data types: RGB888 RAW10 RAW12 RAW8 YUV420_8 YUV422_8 LEGACY_YUV420_8 YUV420_8_CSPS YUV420_10 YUV420_10_CSPS YUV422_10	Specifies the data type to be converted.
Number of Hsync Pulses	User-Input	(3-1023)	This option is valid only for DSI Non-Burst Mode with Sync Events. This specifies the number of HSYNC pulses within the active VSYNC region.
Number of Pix Clock Cycles HSYNC is Active	User-Input	(3-1023)	This option is valid only for DSI Non-Burst Mode with Sync Events. This determines the duration of each hsync_o pulses.
Camera/Display Control Polarity	User-Input	Positive – active high Negative – active low	Sets the polarity of the camera or display sync and control signals (frame valid, line valid, vsync, hsync, data enable).
Enable Debug Signals	User-Input	Check or Unchecked	If checked, this shows some of the internal signals used to control the FIFO.

4. IP Generation and Evaluation

This section provides information on how to generate Lattice Byte-to-Pixel Converter IP using the Diamond Clarity Designer, and how to run simulation, synthesis, and hardware evaluation.

4.1. Licensing the IP

The Byte-to-Pixel Converter IP is available free of charge but an IP-specific license is required to enable full, unrestricted use of the Byte-to-Pixel Converter IP in a complete, top-level design.

Request your license by going to the link http://www.latticesemi.com/en/Support/Licensing and request the free Lattice Diamond license. In this form, select the desired CrossLink/CrossLinkPlus IP for your design.

You may download or generate the Byte-to-Pixel Converter IP and fully evaluate it through functional simulation and implementation (synthesis, map) without the IP license. The Byte-to-Pixel Converter IP also supports Lattice IP hardware evaluation capability, see the Hardware Evaluation section for further details.

HOWEVER, THE IP LICENSE IS REQUIRED TO ENABLE TIMING SIMULATION, TO OPEN THE DESIGN IN DIAMOND EPIC TOOL, OR TO GENERATE BITSTREAMS THAT DO NOT INCLUDE THE HARDWARE EVALUATION TIMEOUT LIMITATION.

4.2. Getting Started

The Byte-to-Pixel Converter IP is available for download from the Lattice IP Server using the Clarity Designer tool. The IP is available in the Clarity Designer user interface, as shown in Figure 4.1.

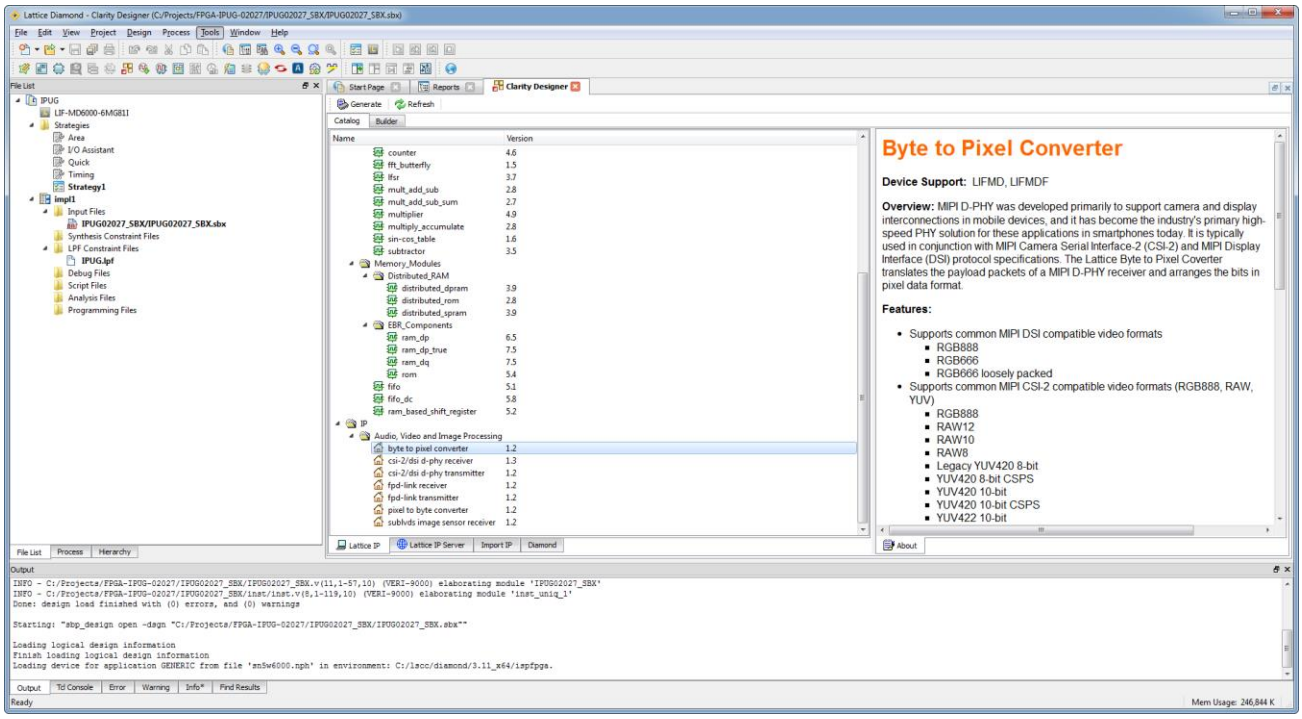

Figure 4.1. Clarity Designer View

4.3. Generating IP in Clarity Designer

Clarity Designer is a tool used to customize modules and IPs and place them into the device architecture. The procedure for generating Byte-to-Pixel Converter IP in Clarity Designer is described below. Clarity Designer is started from Lattice Diamond design environment.

To start Clarity Designer:

1. Create a new Diamond project for CrossLink or CrossLinkPlus family devices.
2. From the Diamond main window, choose **Tools** > **Clarity Designer**, or click  in Diamond toolbox. The Clarity Designer project dialog box is displayed.
3. Select and fill out the following items as shown in Figure 4.2:
   - **Create new Clarity design** – Click this to create a new Clarity Design project directory in which the Byte-to-Pixel Converter IP is generated.
   - **Design Location** – Clarity Design project directory path.
   - **Design Name** – Clarity Design project name.
   - **HDL Output** – Hardware Description Language Output Format (Verilog).

   The Clarity Designer project dialog box also allows you to open an existing Clarity Designer project by selecting the following:
   - **Open Clarity design** – Open an existing Clarity Design project.
   - **Design File** – Name of existing Clarity Design project file with .sbx extension.
4. Click the **Create** button. A new Clarity Designer project is created.

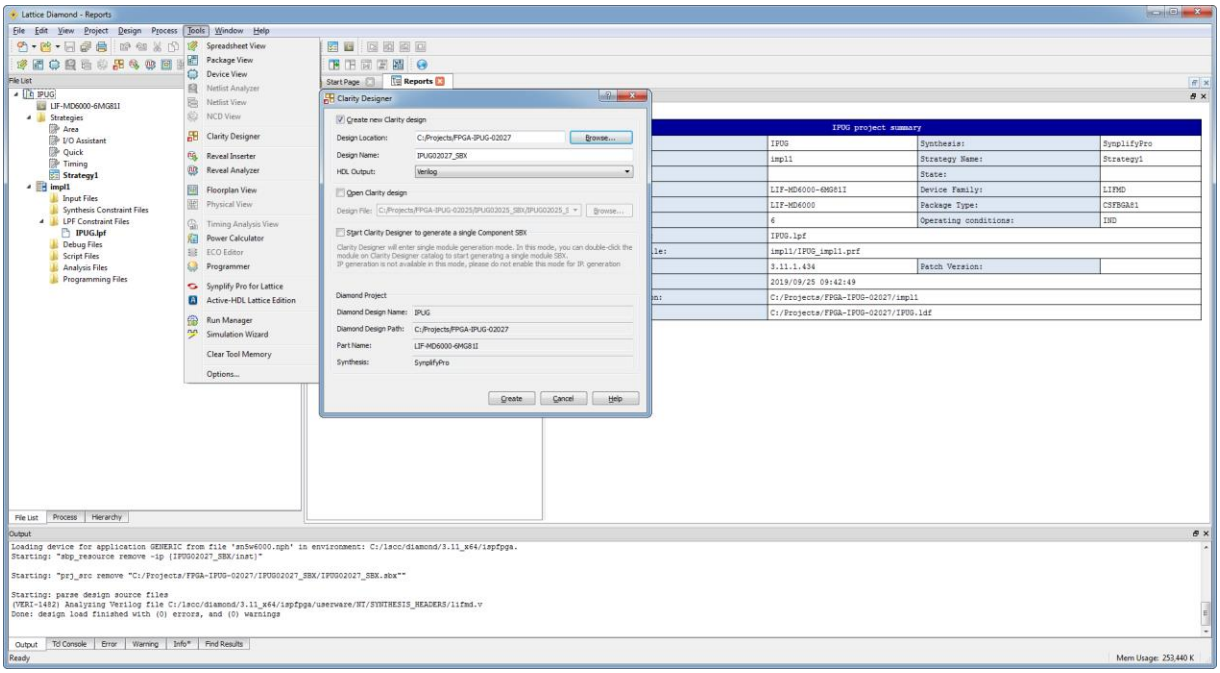

Figure 4.2. Starting Clarity Designer from Diamond Design Environment

To configure the Byte-to-Pixel Converter IP in Clarity Designer:

1. Double-click **byte to pixel converter** in the IP list of the Catalog view. The **byte to pixel converter** dialog box is displayed as shown in Figure 4.3.

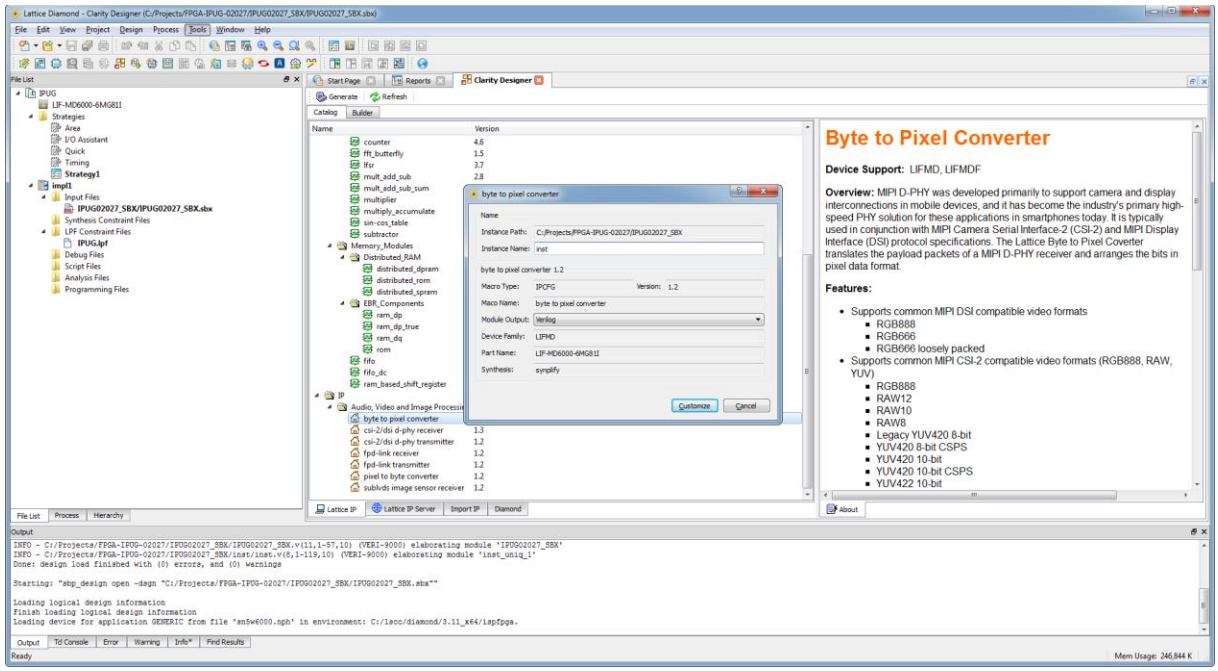

Figure 4.3. Configuring Byte-to-Pixel Converter IP in Clarity Designer

2. Enter the Instance Name.
3. Click the **Customize** button. An IP configuration user interface is displayed as shown in Figure 4.4. From this dialog box, you can select the IP parameter options specific to your application.

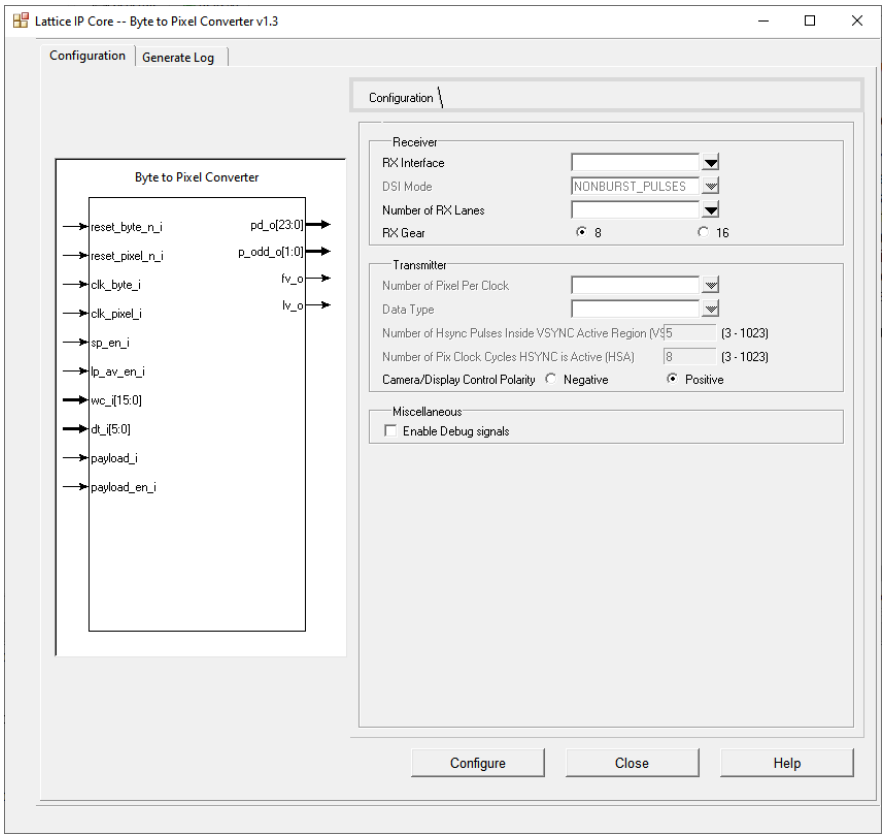

Figure 4.4. Configuration Tab in IP User Interface

4. Click the **Configure** button after the required parameters are selected.
5. Click **Close**.
6. Click Generate in the toolbox. Clarity Designer generates all the IPs and modules, and creates a top module to wrap them.

For detailed instructions on how to use the Clarity Designer, refer to the Lattice Diamond software user guide.

4.4. Generated IP Directory Structure and Files

Figure 4.5 shows the directory structure of the generated IP files.

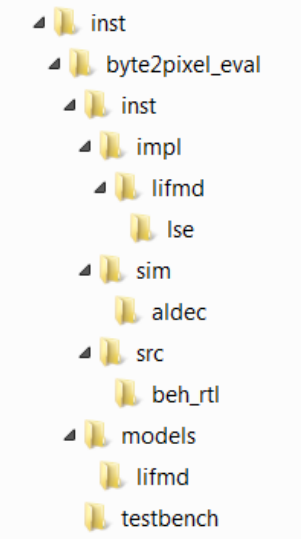

Figure 4.5. IP Directory Structure

The design flow for the IP created with Clarity Designer uses post-synthesized modules (NGO) of the IP core modules for synthesis. Protected models are used for simulation. The post-synthesized modules are customized when you configure the IP and are created automatically when the IP is generated. The protected models are common to all configurations.

Table 4.1 provides a list of key files and directories created by Clarity Designer with details on how they are used.

Table 4.1. Files Generated by Clarity Designer

File	Description
<instance_name>.v	Verilog top-level module of Byte-to-Pixel Converter IP used for both synthesis and simulation.
<instance_name>_*.v	Verilog submodules for simulation. Files that do not have equivalent black box modules are also used for synthesis.
<instance_name>_*_beh.v	Protected Verilog models for simulation.
<instance_name>_*_bb.v	Verilog black box modules for synthesis.
<instance_name>_*.ngo	User interface configured and synthesized modules for synthesis.
<instance_name>_params.v	Verilog parameters file which contains required compiler directives to successfully configure IP during synthesis and simulation.
<instance_name>.lpc	Lattice Parameters Configuration file. This file records all the IP configuration options set through Clarity Designer. It is used by the IP generation script to generate configuration-specific IP. It is also used to reload parameter settings in the IP user interface in Clarity Designer when it is being reconfigured.
<instance_name>_inst.v/vhd	Instance template if you want to instantiate the generated soft IP top-level in their own top-level module.

All IP files are generated inside \<*project_dir*> directory (inst folder in Figure 4.5). The \<*project_dir*> *is* <*design_location*>*design_name*>\<*instance_name*>, see the Generating IP in Clarity Designer section. A separate \<*project_dir*> is created each time Byte-to-Pixel Converter IP is created with a different IP instance name.

The *byte2pixel_eval* and subdirectories provide files supporting push-button IP evaluation through functional simulations, design implementation (synthesis, map, place and route) and hardware evaluation. Inside the *byte2pixel_eval* is the \<*instance_name*> folder (inst folder in Figure 4.5) which contains protected behavioral files in \<*instance_name*>*src**beh_rtl* and a pre-built Diamond project in \<*instance_name*>*impl*\<*device_family*>\<*synthesis_tool*>\, where <*device_family*> can either be lifmd for CrossLink or lifmdf for CrossLinkPlus devices.

The <*instance_name*> is the IP instance name that you specified in Clarity Designer. The simulation part of user evaluation provides testbench and test cases supporting RTL simulation for Active-HDL simulator under *testbench* folder. Separate directories located at \<*project_dir*>*byte2pixel_eval*\<*instance_name*>*sim**aldec* are provided and contain specific pre-built simulation script files. See the Running Functional Simulation section for more details.

The pll_wrapper model in \<*project_dir*>*models*\ is used for both simulation and implementation.

4.5. Running Functional Simulation

The generated IP package includes the behavioral models (<*instance_name*>_*_*beh.v*) provided in \<*project_dir*>*byte2pixel_eval*\<*instance_name*>*src**beh_rtl* for functional simulation. The testbench files can be found in \<*project_dir*>*byte2pixel_eval**testbench*.

To run the evaluation simulation on Active-HDL (Windows only):

1. Create new project using Lattice Diamond for Windows.
2. Open **Active-HDL Lattice Edition** user interface tool.
3. To customize the testbench parameters, edit the file tb_params.v inside the <*project_dir*>\<*instance_name*>\byte2pixel_eval\testbench folder. See Table 4.2 for the list of valid testbench compiler directives.
4. Click **Tools**, then click **Execute macro**.
5. Select the <*instance_name*>_run.do file inside the <*project_dir*>\<*instance_name*>\byte2pixel_eval\<*instance_name*>\sim\aldec folder.
6. Wait for the simulation to finish.
7. Input and output log files are saved in the **sim** directory.
8. Testbench parameters and directives can be modified by setting the define in the vlog command in the *.do file. Table 4.2 lists testbench directives common for DSI and CSI-2 Rx type.

Table 4.2. Testbench Directives Common for DSI and CSI-2

File	Description
SIP_BCLK	Used to set the period of the input byte clock (in ps).
SIP_PCLK	Used to override the pixel clock (in ps). By default, the testbench automatically calculates the pixel clock based from the byte clock and other design settings.
NUM_FRAMES	Used to set the number of video frames in a test.
NUM_LINES	Used to set the number of lines per frame.
NUM_OF_BYTES	Number of bytes in each line.

Table 4.3 is a list of additional testbench directives for DSI Rx type.

Table 4.3. Testbench Directives for DSI Rx Type

File	Description
HFP_PAYLOAD	Used to set horizontal front porch (number of byte clock cycles from payload_en_i negation to HSYNC start).
HSA_PAYLOAD	Used to set HSYNC width (number of byte clock cycles from HSYNC Start to HSYNC End).
HBP_PAYLOAD	Used to set Horizontal back porch (number of byte clock cycles from HSYNC end to payload_en_i rise).
VFP_LINES	Used to set vertical front porch (number of HSYNC pulses before VSYNC for next frame).
VSA_LINES	Used to set VSYNC width (number of HSYNC pulses within VSYNC).
VBP_LINES	Used to set vertical back porch (number of HSYNC pulses after VSYNC).

4.6. Simulation Strategies

This section describes the simulation environment which demonstrates basic Byte-to-Pixel Converter IP functionality. Figure 4.6 shows the block diagram of simulation environment.

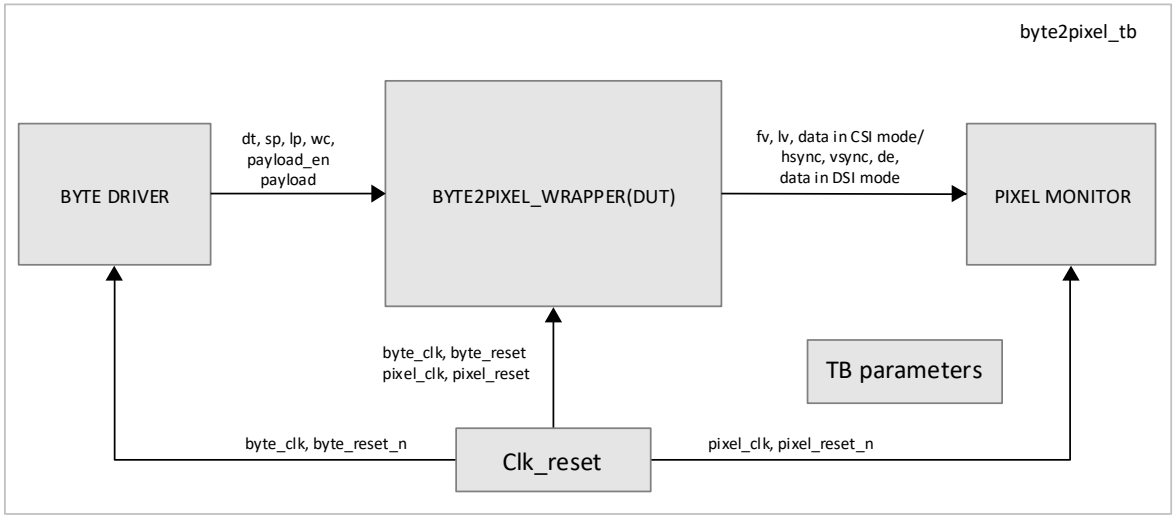

Figure 4.6. Simulation Environment Block Diagram

4.7. Simulation Environment

The simulation environment is made up of a byte clock domain input driver instance connected to the input of BYTE2PIXEL IP instance in the testbench. The byte clock domain input driver is configured based on BYTE2PIXEL configurations and testbench parameters. After reset, the testbench drives data packets and byte data in either DSI or CSI-2 mode based on configuration. For details on how to set the testbench parameters, refer to the testbench tb_params.v file available in the *<project_dir>\<instance_name>\byte2pixel_eval\testbench*. Input byte data and output pixel data are logged into input_data.log and output_data.log files, respectively. Compare these files to check if data is being transmitted properly.

Figure 4.7 shows an example simulation of CSI-2 configuration.

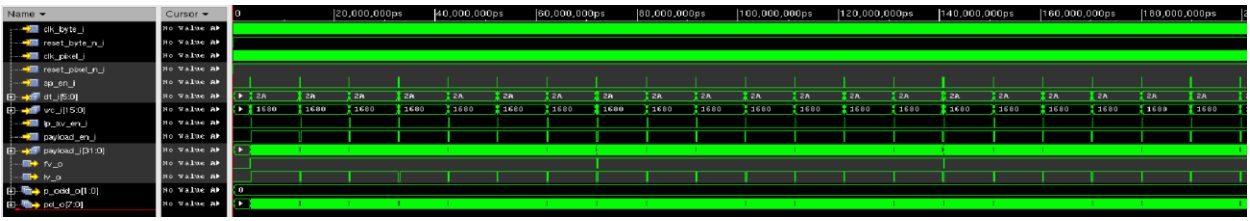

Figure 4.7. CSI-2 Simulation Waveform

Figure 4.8 shows an example simulation of DSI configuration.

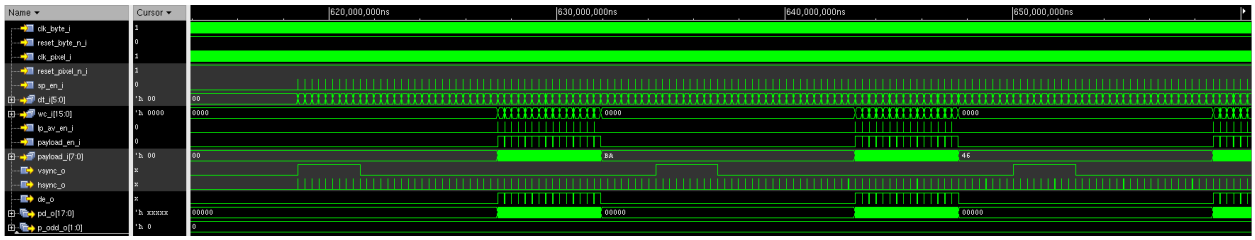

Figure 4.8. DSI Simulation Waveform

4.8. Instantiating the IP

The core modules of Byte-to-Pixel Converter IP are synthesized and provided in NGO format with black box Verilog source files for synthesis. A Verilog-HDL source file named *<instance_name>_byte2pixel.v* instantiates the black box of core modules. The top-level file *<instance_name>.v* instantiates *<instance_name>_byte2pixel.v*.

You do not need to instantiate the IP instances one by one manually. The top-level file along with other Verilog source files are provided in *\<project_dir>*. These files are refreshed each time the IP is regenerated.

A Verilog instance template *<instance_name>_inst.v* or VHDL instance template *<instance_name>_inst.vhd* is also provided as a guide on how to instantiate the generated soft IP in their own top-level module.

4.9. Synthesizing and Implementing the IP

In Clarity Designer, the Clarity Designer project file (.sbx) is added to Lattice Diamond as a source file after IP is generated. All required Verilog source files for implementation are invoked automatically. The IP can be directly synthesized, mapped and placed/routed in the Diamond design environment after the IP is generated. Note that default Diamond strategy (.sty) and default Diamond preference file (.lpf) are used. When using the .sbx approach, import the recommended strategy and preferences from *\<project_dir>\byte2pixel_eval\<instance_name>\impl\<device_family>\lse* or *\<project_dir>\byte2pixel_eval\<instance_name>\impl\<device_family>\synplify* directories and set them as active strategy and active preference file.

Push-button implementation of this IP with either Lattice Synthesis Engine (LSE) or Synopsys Synplify Pro RTL synthesis is supported via the pre-built Diamond project file *<instance_name>_top.ldf* located in *\<project_dir>\byte2pixel_eval\<instance_name>\impl\<device_family>\lse* or *\<project_dir>\byte2pixel_eval\<instance_name>\impl\<device_family>\synplify* directories.

To use the pre-built Diamond project file:

1. Choose **File > Open > Project**.
2. In the Open Project dialog box, browse to *\<project_dir>*\byte2pixel_eval*<instance_name>*\impl*<device_family>**<synthesis_tool>*.
3. Select and open *<instance_name>*_top.ldf. At this point, all of the files needed to support top-level synthesis and implementation are imported to the project.
4. Select the **Process** tab in the left-hand user interface window.
5. Implement the complete design via the standard Diamond user interface flow.

4.10. Hardware Evaluation

The Byte-to-Pixel Converter IP supports Lattice IP hardware evaluation capability. You can create versions of the IP that operate in hardware for a limited period of time without requiring the request of an IP license. It may also be used to evaluate the IP in hardware in user-defined designs.

4.10.1. Enabling Hardware Evaluation in Diamond

Choose Project > Active Strategy > Translate Design Settings. The hardware evaluation capability may be enabled or disabled in the Strategy dialog box. It is enabled by default.

4.11. Updating/Regenerating the IP

Clarity Designer allows you to update the local IPs from the Lattice IP server. The updated IP can be used to regenerate the IP instance in the design. To change the parameters of the IP used in the design, the IP must also be regenerated.

4.11.1. Regenerating an IP in Clarity Designer

To regenerate IP in Clarity Designer:

1. In the Builder tab, right-click the IP instance to be regenerated and select Config from the menu as shown in Figure 4.9.

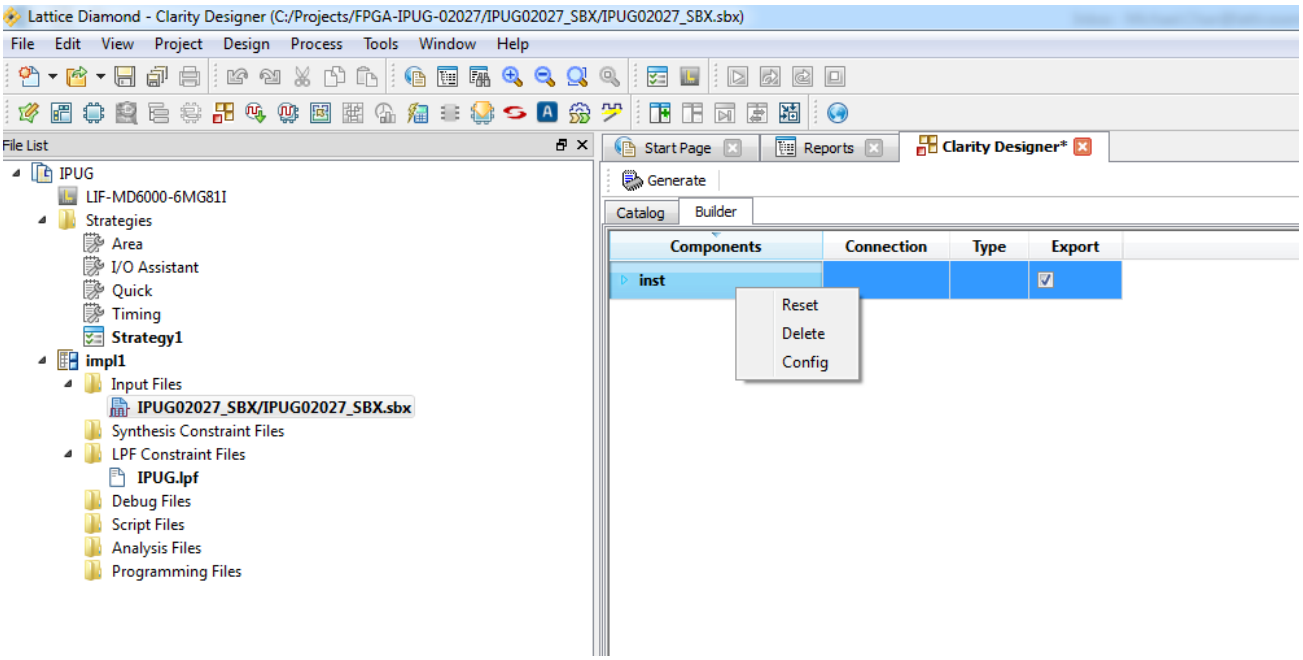

Figure 4.9. Regenerating IP in Clarity Designer

2. The IP Configuration user interface is displayed. Change the parameters as required and click the Configure button.
3. Click Generate in the toolbox. Clarity Designer regenerates all the instances which are reconfigured.

References

For more information about CrossLink and CrossLinkPlus devices, refer to CrossLink Family Data Sheet (FPGA-DS-02007) and CrossLinkPlus Family Data Sheet (FPGA-DS-02054).

Software documentation:

- Clarity Designer User Manual
- Lattice Diamond User Guide

For further information on interface standards, refer to:

- MIPI Alliance Specification for D-PHY, version 1.1, November 7, 2011, www.mipi.org
- MIPI Alliance Specification for Display Serial Interface, version 1.1, November 22, 2011, www.mipi.org
- MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2), version 1.1, July 18, 2012, www.mipi.org

Technical Support Assistance

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

Appendix A. Resource Utilization

Table A.1 lists resource utilization for Lattice CrossLink FPGAs using the Byte-to-Pixel Converter. The values shown below are based on map reports. The performance and utilization data target an LIF-MD6000-6MG81I device with –6 speed grade using Lattice Diamond 3.9 and Lattice Synthesis Engine. Performance may vary when using a different software version or targeting a different device density or speed grade within the CrossLink/CrossLinkPlus family.

The Target f_{MAX} is 150 MHz, which is the maximum supported frequency of the FPGA fabric in CrossLink devices. Actual f_{MAX} varies depending on the complete top level design.

Table A.1. Resource Utilization

IP User-Configurable Parameters	Slices	LUTs	Registers	sysMEM EBRs	Programmable I/O
RGB888, Gear 16, 4-lane, 2 pixel output	463	511	548	6	0
RGB888, Gear 8, 4-lane, 2 pixel output	267	289	333	3	0
RGB888, Gear 16, 2-lane, 1 pixel output	249	259	305	3	0
RGB888, Gear 8, 2-lane, 1 pixel output	215	238	264	2	0

Appendix B. What is Not Supported

- The signal p_odd_o, used to indicate valid pixels in multiple simultaneous pixel outputs, is not verified. The pixel count in all scenarios where the design is used is always in multiple of the number of pixel outputs.
- See Table 2.2 for the list of supported number of pixels and bytes.
- See Table 2.3 and Table 2.4 for the list of valid configurations.

Revision History

Revision 1.4, IP Version 1.3, September 2021

Section	Change Summary
Introduction	In the Supported section, modified item to *Supports DSI Non-Burst Mode with Sync Events, Non-Burst Mode with Sync Pulses and Burst Mode*.
Functional Description	• Added information regarding FIFO depth for burst mode In the FIFO Implementation section. • Indicated clock domain behavior in CSI-2 and DSI Non-Burst Mode as compared to DSI Burst Mode in the Clock Domains section.
Configuration Settings	General update to Table 3.1. Byte-to-Pixel Converter IP User Interface Parameter Settings.
IP Generation and Evaluation	Updated Figure 4.4.
Appendix B. What is Not Supported	Removed item on non-support of Burst Mode and Non-Burst Mode with Sync Events DSI Video Modes.

Revision 1.3, IP Version 1.3, March 2020

Section	Change Summary
Introduction	• Moved contents of Features section to a new subsection Supported. • Updated Signal Names section.
Functional Description	• Updated Figure 2.1 and Figure 2.4. • Updated Table 2.3 and Table 2.4.
Configuration Settings	• Changed section name from Compiler Directives and Parameter Settings to *Configuration Settings*. • Removed RTL Compiler Directives and Parameter Settings section.
IP Generation and Evaluation	Updated Figure 4.4.

Revision 1.2, IP Version 1.3, October 2019

Section	Change Summary
Disclaimer	Newly added section.
All	Added CrossLinkPlus devices support. Minor adjustments in style and formatting.
References	Updated.

Revision 1.1, IP Version 1.0, April 2019

Section	Change Summary
Introduction	Specified that this user guide can be used for IP design versions 1.x.
IP Generation and Evaluation	In Licensing the IP, modified the instructions for requesting free license.
Appendix B. What is Not Supported	Removed lead-in sentence.
Revision History	Updated revision history table to new template.
All	Minor adjustments in style and formatting.

Revision 1.0, IP Version 1.0, July 2017

Section	Change Summary
All	Initial release.

www.latticesemi.com