MIPI DSI_CSI2-OpenLDI-LVDS-Int-Bridge-LFCPNX

Reference Design

FPGA-RD-02252-1.0

March 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, with all faults and associated risk the responsibility entirely of the 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
AP	Application Processor
DSI	Display Serial Interface
CSI-2	Camera Serial Interface 2
DDR	Double Data Rate
FV	Frame Valid
GPLL	General Purpose PLL
HBP	Horizontal Back Porch
HFP	Horizontal Front Porch
HS	High Speed
IP	Intellectual Property
LP	Low Power
LV	Line Valid
LVDS	Low Voltage Differential Signal
MIPI	Mobile Industry Processor Interface
OpenLDI	Open LVDS Display Interface
PLL	Phase Locked Loop
RD	Reference Design
RX	Receiver
TX	Transmitter
VBP	Vertical Back Porch
VFP	Vertical Front Porch
VCO	Voltage Controlled Oscillator

Supported Device and IP

This reference design supports the following devices with IP versions.

Device Family	Part Number	Compatible IP
CertusPro-NX	LFCPNX-100-7LFG672I	Byte-to-Pixel Converter IP version 1.4.0 RX-DPHY IP version 1.3.0 OpenLDI/LVDS Transmitter Interface IP version 1.1.0

The IPs above are supported by Lattice Radiant™ software version 3.0.0.24.1 or later.

1. Introduction

Mobile Industry Processor Interface (MIPI®) Display Serial Interface (DSI) is one of the most popular display interfaces in the consumer market today. On the other hand, OpenLDI LVDS is still popular in some areas as the main predecessor of the display interface. The majority of image sensors and application processors (AP) in the consumer market use MIPI Camera Serial Interface 2 (CSI-2) as a video signal interface. In some cases, the interface and/or format conversion is useful to connect devices that cannot connect directly.

The Lattice Semiconductor MIPI DSI/CSI-2 to OpenLDI LVDS Interface Bridge reference design for CertusPro-NX™ devices takes DSI or CSI-2 MIPI data and converts them to OpenLDI format on LVDS. The MIPI RX module can also be realized by a soft macro utilizing general DDR modules (D-PHY Soft IP) while the LVDS TX module is realized by soft macro utilizing general DDR modules.

1.1. Features List

- Single DSI input (RGB888) to single or dual-channel LVDS outputs (RGB888)
- Single CSI-2 input (RGB888, RAW8, RAW10, RAW12 or RAW14) to single or dual-channel RGB888 LVDS outputs (RGB888)
- RX channel can have one, two, or four lanes with the bandwidth up to 1.5 Gbps per lane using RX D-PHY Soft IP.
- Number of TX data lanes is four (RGB888) per TX channel
- Maximum TX bandwidth is 945 Mbps per lane
- Image cropping option is available in case of CSI-2 input
- Dynamic parameter setting is possible via I^{2}C in the case of CSI-2 input

1.2. Block Diagram

Figure 1.1 shows the block level diagram of the MIPI DSI/CSI-2 to OpenLDI LVDS Interface Bridge reference design with single TX channel.

There exist two main clock domains for the main video data path: byte clock and pixel clock. GPLL is required to generate the edge clock of LVDS TX module. Also, it might be used to generate the continuous byte clock when RX D-PHY is in HS_LP mode. I2C Slave modules enable parameter change on the fly if necessary for CSI-2 RX.

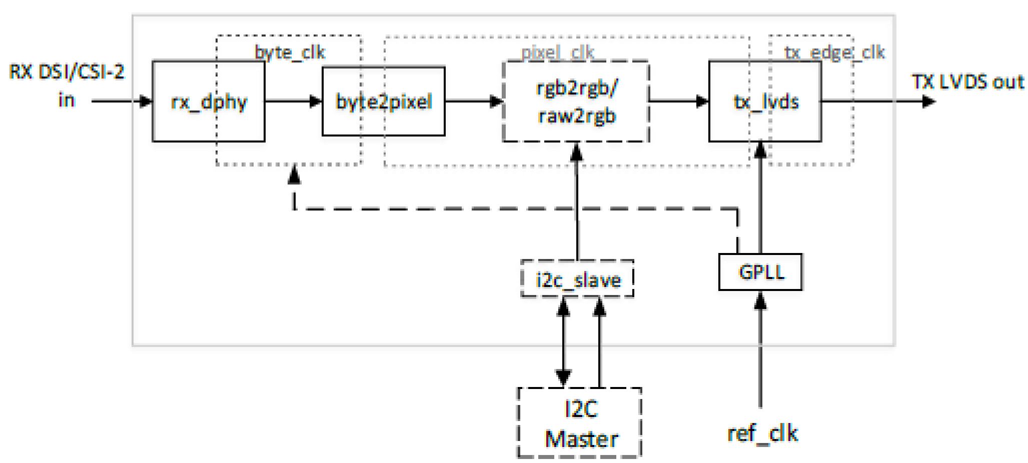

Figure 1.1. DSI/CSI-2 to OpenLDI LVDS Interface Bridge Block Diagram

1.3. RX and TX Permutations

Table 1.1 shows the available permutations of RX and TX configurations. Some permutations exist only for DSI and some only for CSI-2 due to the limitations of the Byte-to-Pixel IP used in this reference design. Note that TX Data Type is the same as the RX Data Type in the case of DSI. The TX Data Type is always RGB888 in the case of CSI-2.

Table 1.1. RX and TX Permutation

D-PHY Type	Data Type	Number of RX Lanes	RX Gear	Number of Pixels/ Pixel Clock	TX Gear	Number of TX Channel
DSI	RGB888	1	8	1	7	1
		2	8	1	7	1
		4	8	1	7	1
				2	7	2
CSI-2	RGB888, RAW8, RAW10, RAW12	1	8	1	7	1
		2	8	2	7	2
		4	8	1	7	1
	RAW14	1	8	1	7	1
		2	8	2	7	2
		4	8	2	7	1
						2

The Microsoft Excel sheet (mipi2lvds_clock_LFCPNX.xlsx) is provided to get the byte clock, TX edge clock, and others from RX bandwidth, in addition to other information. This sheet is useful in configuring IPs. A sample entry is shown in Figure 1.2. By entering MIPI bandwidth and other information, Byte clock, LVDS bandwidth, and TX ECLK are automatically calculated. The results can be used to configure TX LVDS IP and GPLL.

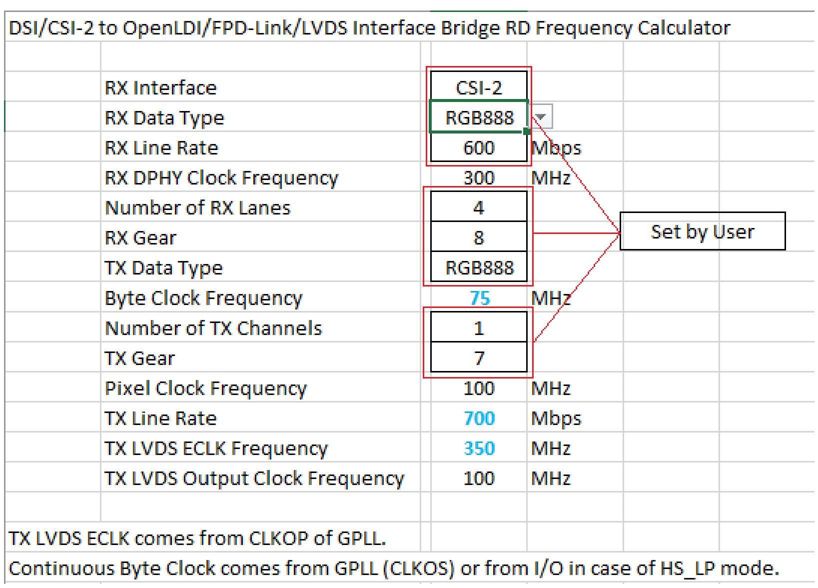

Figure 1.2. Bandwidth and Clock Frequency Calculator

2. Parameters and Port List

There are two directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Radiant® and simulation.
- simulation_directives.v – used for simulation.

You can modify these directives according to your own configuration. The settings in these files must match RX D-PHY IP, Byte-to-Pixel IP, and TX LVDS IP settings created in Lattice Radiant Software.

2.1. Synthesis Directives

Table 2.1 shows the synthesis directives that affect this reference design. These are used for both synthesis and simulation. As shown in Table 2.1 and Table 2.2, some parameter selections are restricted by other parameter settings.

Table 2.1. Synthesis Directives

Category	Directive	Remarks
External reference clock	EXT_REF_CLK	Always enabled to externally provide a synchronous clock to TxLVDS IP. This clock value must be lower than all clocks in the system.
RX D-PHY Type	RX_TYPE_DSI	Define the D-PHY type on the RX channel. Only one of these directives must be defined.
	RX_TYPE_CSI2	
RX Data Type	RX_RGB888	Define the data type on the RX channel. Only one of these directives must be defined. In the case of DSI, only RGB888 is allowed. In the case of CSI-2, only RGB888, RAW8, RAW10, RAW12, and RAW14 are allowed.
	RX_RAW8	
	RX_RAW10	
	RX_RAW12	
	RX_RAW14	
Number of RX Channel Lanes	NUM_RX_LANE_1	Number of lanes in the RX channel. Only one of these three directives must be defined.
	NUM_RX_LANE_2	
	NUM_RX_LANE_4	
RX D-PHY Clock Gear	RX_GEAR_8	Number of RX clock gears on the RX channel. Always will be Rx Gear 8 since CertusPro-NX supports only RX SOFT DPHY.
RX D-PHY Clock Mode[1]	RX_CLK_MODE_HS_ONLY	RX D-PHY clock mode. Only one of these two directives must be defined.
	RX_CLK_MODE_HS_LP	
Line Buffer Depth	LB_DEPTH_512	Define the depth of the line buffer in case the RX Data Type is RAW8, RAW10, RAW12, or RAW14. This depth must be equal or greater than the active pixel count per line, irrespective of the data type.
	LB_DEPTH_1024	
	LB_DEPTH_2048	
	LB_DEPTH_4096	
Sync Signal Polarity	SYNC_POLARITY_POS	Define sync signal (VSYNC, HSYNC) polarity. Only one of these two directives must be defined.
	SYNC_POLARITY_NEG	
Data Enable Polarity	DE_POLARITY_POS	Define data enable (DE) polarity. Only one of these two directives must be defined.
	DE_POLARITY_NEG	
TX Data Type[2]	TX_RGB888	Define the data type on the TX channel. Only RGB888 must be defined in all cases. (User must not modify this)
Number of TX Channels	NUM_TX_CH_1	Number of channels on TX. Only one of these two directives must be defined.
	NUM_TX_CH_2	

TX LVDS Gear	TX_GEAR_7	Number of TX clock gears on RX channel. Only TX_GEAR_7 is used. (User must not modify this)
Parameter set by I²C	USE_I2C	Define this to use I²C I/F to set the parameters on the fly. This is applicable to CSI-2 input only.
Software Reset Register[4]	SW_RST_N {value}	Default value of the software reset register of I²C Slave module. Value must be 1'b0 or 1'b1 Applicable only when USE_I2C is defined. Active low.
Shift Register Delay[5]	SR_DELAY {value}	Define shift register delay. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 8'd2 – 8'd255. Applicable only to CSI-2 input with RGB888.
Bayer Pattern[6]	BAYER_PATTERN {value}	Define the Bayer pattern. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 2'b00 – 2'b11. Applicable only to CSI-2 input with RAW8/RAW10/RAW12/RAW14.
Horizontal Front Porch[7]	HFP {value}	Define the horizontal front porch. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 8'd1 – 8'd255. Applicable only to CSI-2 input.
HSYNC Pulse Length[8]	HS_LENGTH {value}	Define HSYNC pulse length. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 8'd1 – 8'd255. Applicable only to CSI-2 input.
Vertical Front Porch[8]	VFP {value}	Define the vertical front porch. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 6'd1 – 6'd63. Applicable only to CSI-2 input.
VSYNC Pulse Length[8]	VS_LENGTH {value}	Define VSYNC pulse length. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 6'd1 – 6'd63. Applicable only to CSI-2 input.
Left Pixel Trimming[9]	LEFT_TRIM {value}	Define the number of pixels to be trimmed before TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 6'd0 – 6'd63. Applicable only to CSI-2 input.
Horizontal Active Pixels on TX[9]	H_TX_PEL {value}	Define the number of active pixels to send on TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 12'd1 – 12'd4095. The value must be even in the case of NUM_TX_CH_2. Applicable only to CSI-2 input.
Top Line Trimming[10]	TOP_TRIM {value}	Define the number of lines to be trimmed before TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 6'd0 – 6'd63. Applicable only to CSI-2 input.
Vertical Active Lines on TX[10]	V_TX_LINE {value}	Define the number of active lines to be sent on TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I²C register when USE_I2C is defined. Value must be 12'd1 – 12'd4095. Applicable only to CSI-2 input.

Notes:

1. HS_LP mode means non-continuous clock mode and HS_ONLY means continuous clock mode. HS_LP mode works only if RX byte clock can be generated internally or directly fed from I/O pin.
2. TX data type must match RX data type in case of RGB input. RGB888 output is used for all RAW input.

3. LSB two bits of I2C slave address are automatically set when I2C IP is created by Radiant Software.
4. Logical OR between this register and system reset (reset_n_i) is used to reset modules other than I2C slave module.
5. Refer to the 3.3.1 Sync Signal Generation section for details.
6. Refer to the 3.4.1 RGB Data Creation section for pattern details.
7. (HFP + HS_LENGTH) must be less than the horizontal blanking period after the byte-to-pixel conversion. It is your responsibility to manage this. Small values are recommended if you have no idea about the length of the blanking period.
8. (VFP + VS_LENGTH) must be less than the vertical blanking period after the byte-to-pixel conversion. It is your responsibility to manage this. Small values are recommended if you have no idea about the length of the blanking period.
9. (LEFT_TRIM + H_TX_PEL) cannot exceed the horizontal active pixel count of the incoming RX data. It is user's responsibility to manage this.
10. (TOP_TRIM + V_TX_LINE) cannot exceed the vertical active line count of the incoming RX data. It is user's responsibility to manage this.

2.2. Simulation Directives

Table 2.2 shows the simulation directives for this reference design.

Table 2.2. Simulation Directives

Category	Directive	Remarks
Simulation setup	SIM	Define to select the right behavioral model in simulation.
Reference clock period	REF_CLK {value}	Reference clock period in ps
RX D-PHY clock period	DPHY_CLK {value}	RX DPHY clock period in ps
Number of frames to run	NUM_FRAMES {value}	Number of video frames fed by testbench
Number of active lines	NUM_LINES {value}	Number of active video lines per frame
Number of active pixels	NUM_PIXELS {value}	Number of active video pixels per line
HSYNC period in DSI	DSI_HSA_PAYLOAD {value}	HSYNC active count in the unit of payload byte. Applicable to DSI RX.
Blanking period in DSI	DSI_BLLP_PAYLOAD {value}	Blanking period in the unit of payload byte. Used in HS_ONLY mode. Applicable to DSI RX.
Horizontal Back Porch in DSI[1]	DSI_HBP_PAYLOAD {value}	Horizontal Back Porch or Low Power period in the unit of payload byte count. Used in HS_LP mode or non-burst sync pulse in HS_ONLY mode. Applicable to DSI RX.
Horizontal Front Porch in DSI[1]	DSI_HFP_PAYLOAD {value}	Horizontal Front Porch or Low Power period in the unit of payload byte count. Used in HS_LP mode or non-burst sync pulse in HS_ONLY mode. Applicable to DSI RX.
VSYNC period in DSI	DSI_VSA_LINES {value}	Number of VSYNC active lines. Applicable to DSI RX.
Vertical Back Porch in DSI	DSI_VBP_LINES {value}	Number of Vertical Back Porch lines. Applicable to DSI RX.
Vertical Front Porch in DSI	DSI_VFP_LINES {value}	Number of Vertical Front Porch lines. Applicable to DSI RX.
EOTP packet on/off in DSI	DSI_ETOP_ENABLE {value}	Enable/Disable EOTP (End of Transmission Packet). Applicable to DSI RX. Value must be 1 (enable) or 0 (disable).
Blanking period in DSI	DSI_LPS_BLLP_DURATION {value}	Blanking (LP) period in ps. Used in HS_LP mode. Applicable to DSI RX.
Horizontal Back Porch period in DSI[1]	DSI_LPS_HBP_DURATION {value}	Horizontal Back Porch period (using LP mode) in ps. Used for non-burst sync events and burst mode in HS_LP mode. Applicable to DSI RX.
Horizontal Front Porch period in DSI[1]	DSI_LPS_HFP_DURATION {value}	Horizontal Front Porch period (using LP mode) in ps. Used for non-burst sync events and burst mode in HS_LP mode. Applicable to DSI RX.
Initial delay on RX channel	READY_DURATION {value}	Initial delay to activate RX channel in ps.
Line Gap period in CSI-2	DPHY_LPS_GAP {value}	Line Gap time in ps. Applicable to CSI-2 RX.
Frame Gap period	FRAME_LPM_DELAY {value}	Frame Gap time in ps. This is LP period between Frame End and Frame Start in case of CSI-2 and LP period after VFP period end and before VSYNC start in case of DSI.

Notes:

1. Refer to MIPI® Alliance Specification for Display Serial Interface (DSI) Version 1.2 for details.
2. Logical OR between this register and system reset (reset_n_i) is used to reset modules other than I^2C slave module.
3. Refer to the 3.3.1 Sync Signal Generation section for details.
4. Refer to the Refer to the 3.4.1 RGB Data Creation section for pattern details.
5. (I2C_HFP + I2C_HS_LENGTH) must be less than the horizontal blanking period after the byte-to-pixel conversion. It is your responsibility to manage this. Small values are recommended if you have no idea about the length of the blanking period.
6. (I2C_VFP + I2C_VS_LENGTH) must be less than the vertical blanking period after the byte-to-pixel conversion. It is your responsibility to manage this. Small values are recommended if you have no idea about the length of the blanking period.
7. (I2C_LEFT_TRIM + I2C_H_TX_PEL) cannot exceed the horizontal active pixel count of the incoming RX data. It is your responsibility to manage this.
8. (I2C_TOP_TRIM + I2C_V_TX_LINE) cannot exceed the vertical active line count of the incoming RX data. It is your responsibility to manage this.

2.3. Top-Level I/O

Table 2.3 shows the top-level I/O of this reference design. Actual I/O depends on the customer's channel and lane configurations. All necessary I/O ports are automatically declared by compiler directives.

Table 2.3. DSI/CSI-2 to OpenLDI LVDS Interface Bridge Top Level I/O

Port Name	Direction	Description
Clocks and Resets		
ref_clk_i	I	Input reference clock. Used to feed a clock to TX D-PHY PLL directly or indirectly. This port is declared only when EXT_REF_CLK is defined in synthesis_directives.v.
reset_n_i	I	Asynchronous active low system reset
Control Interface (optional for CSI-2 RX)		
scl	I/O	I²C clock
sda	I/O	I²C data
DSI/CSI-2 RX Interface		
rx_clk_p_i	I	Positive differential RX D-PHY input clock
rx_clk_n_i	I	Negative differential RX D-PHY input clock
rx_d0_p_i	I	Positive differential RX D-PHY input data 0
rx_d0_n_i	I	Negative differential RX D-PHY input data 0
rx_d1_p_i	I	Positive differential RX D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx_d1_n_i	I	Negative differential RX D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx_d2_p_i	I	Positive differential RX D-PHY input data 2 (in case of 4-lane configuration)
rx_d2_n_i	I	Negative differential RX D-PHY input data 2 (in case of 4-lane configuration)
rx_d3_p_i	I	Positive differential RX D-PHY input data 3 (in case of 4-lane configuration)
rx_d3_n_i	I	Negative differential RX D-PHY input data 3 (in case of 4-lane configuration)
LVDS TX Interface		
tx0_clk_p_o	O	Positive differential TX LVDS output clock
tx0_d0_p_o	O	Positive differential TX LVDS output data 0
tx0_d1_p_o	O	Positive differential TX LVDS output data 1
tx0_d2_p_o	O	Positive differential TX LVDS output data 2
tx0_d3_p_o	O	Positive differential TX LVDS output data 3 (in case of TX_RGB888)
tx1_clk_p_o	O	Positive differential TX LVDS output clock on 2nd TX channel (in case of dual TX channel configuration)
tx1_d0_p_o	O	Positive differential TX LVDS output data 0 on 2nd TX channel (in case of dual TX channel configuration)
tx1_d1_p_o	O	Positive differential TX LVDS output data 1 on 2nd TX channel (in case of dual TX channel configuration)

tx1_d2_p_o	O	Positive differential TX LVDS output data 2 on 2nd TX channel (in case of dual TX channel configuration)
tx1_d3_p_o	O	Positive differential TX LVDS output data 3 on 2nd TX channel (in case of dual TX channel configuration with TX_RGB888)

3. Design and Module Description

The top-level design (mipi2lvds_LFCPNX.v) consists of the following modules:

- rx_dphy
- byte2pix
- rgb2rgb (used only for CSI-2 RGB888)
- raw2rgb (used only for CSI-2 RAW8/RAW10/RAW12/RAW14)
- tx_lvds
- int_gpll
- i2c_slave (optional for CSI-2)
- int_osc

The top-level design has reset synchronization logic.

3.1. rx_dphy

This module must be created for the RX channel according to channel conditions, such as the number of lanes, bandwidth, and others. Figure 3.1 shows an example of IP interface settings in Radiant Software for the CSI-2/DSI D-PHY Receiver Submodule IP. You can use the .ipx file (*rx_dphy/rx_dphy.ipx*) included in the sample project and reconfigure according to your needs. Refer to CSI-2/DSI D-PHY Rx IP Core - Lattice Radiant (FPGA-IPUG-02081-1.5) for details.

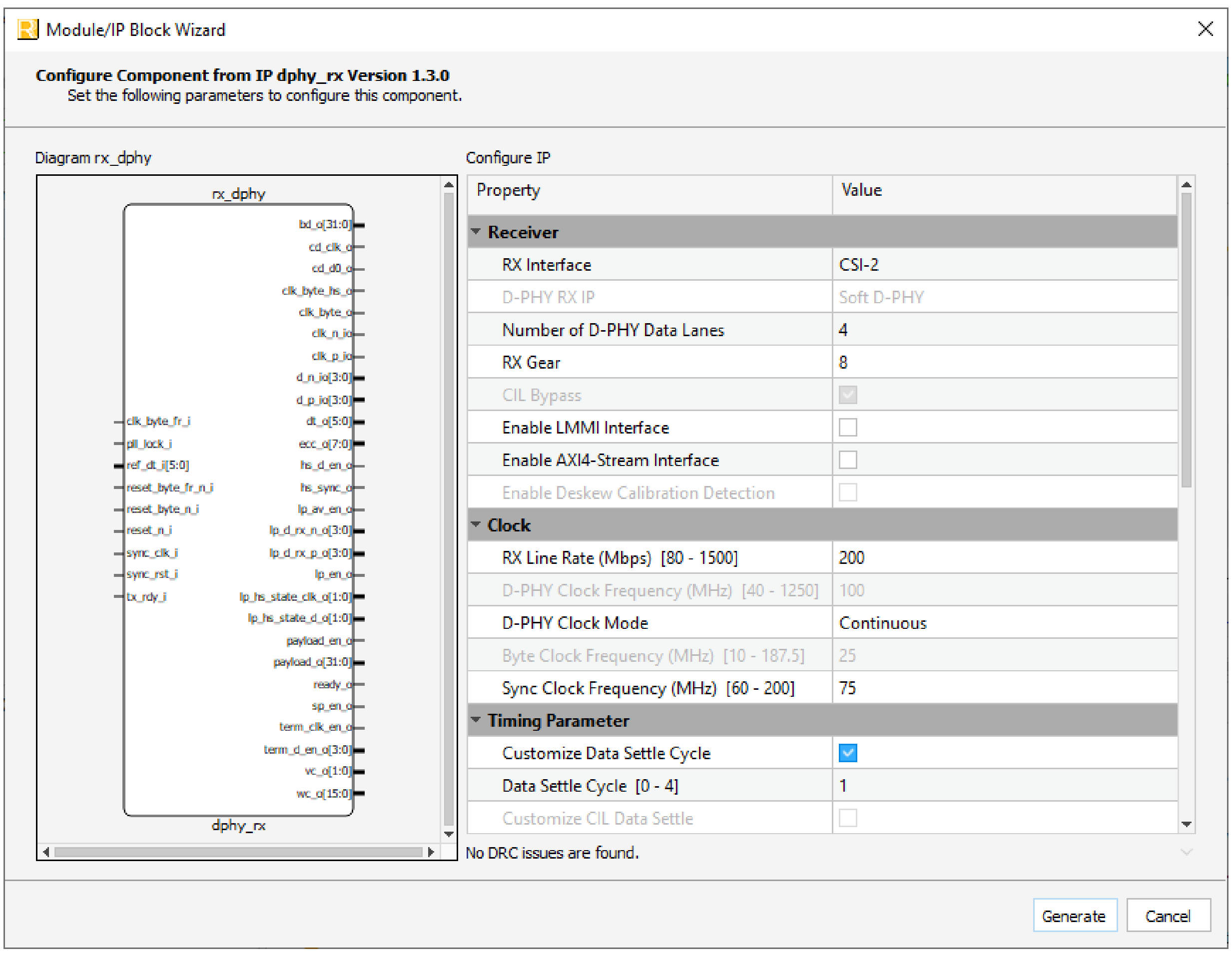

Figure 3.1. rx_dphy IP Creation#1 in Radiant Software

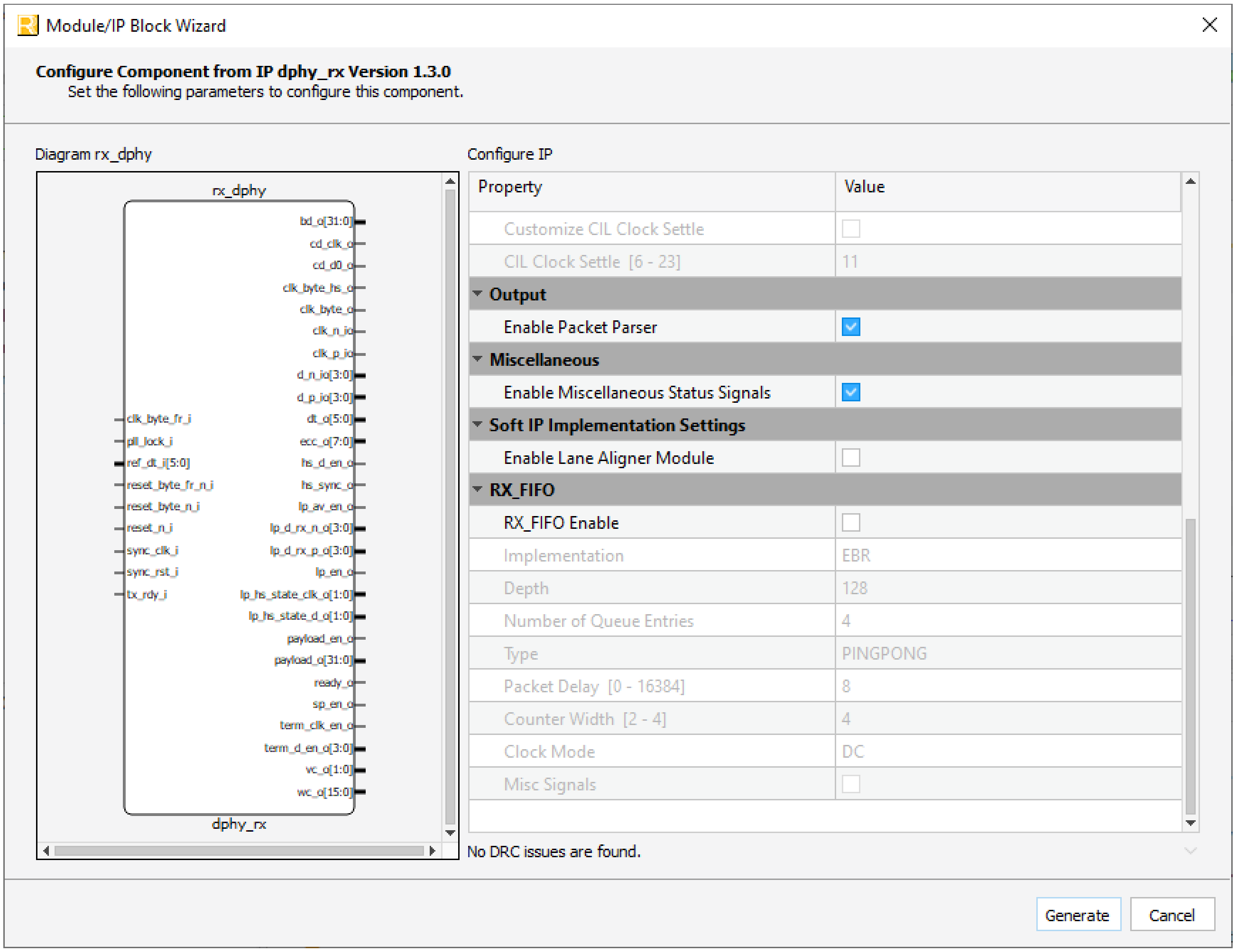

Figure 3.2. rx_dphy IP Creation#2 in Radiant Software

The following provides the guidelines and parameter settings required for this reference design.

- RX Interface – Select DSI or CSI-2.
- D-PHY RX IP- CertusPro-NX supports only SOFT RX DPHY.
- Number of D-PHY Data Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Always will be 8.
- CIL Bypass – Always enabled (checked). Not editable.
- Enable LMMI Interface – Select disabled (unchecked).
- Enable AXI4-Stream Interface – Select disabled (unchecked).
- Enable Deskew Calibration Detection – Always disabled (unchecked). Not editable.
- RX Line Rate – Set according to channel configuration. 800 or below is recommended for 4 lane configurations.
- D-PHY Clock Mode – Select Continuous or Non-continuous. Must match RX*_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- Sync Clock Frequency – Always use 75 MHz.
- Customize Data Settle Cycle – Select checkbox to enable (checked). By default, it will be disabled.
- Data Settle Cycle – Available only if Customize data settle cycle is enabled. By default, it shows the value recommended by the IP as per Clock configurations. In case RxDPHY IP does not behave as expected, try to check by reducing the data settle values.
- Enable Packet Parser – Select checkbox to enable (checked).

- Enable Miscellaneous Status Signals – Select the checkbox to enable (checked).
- Enable Lane Aligner Module – Select checkbox to disable (unchecked) for 2-lane and 4-lane configurations.
- RX_FIFO Enable – Select disabled (unchecked) for continuous clock mode with Soft D-PHY, select checkbox to enable (checked) for all other cases.
- Implementation – Select EBR or LUT. Refer to RX FIFO section for non-continuous clock mode.
- Depth – Select 16, 32, 64, 128, 256, 512, 1024, 2048 or 4096. Refer to RX FIFO section for non-continuous clock mode.
- Type – Select SINGLE for continuous clock mode. Refer to RX FIFO section for non-continuous clock mode.
- Packet Delay – Set 1 for continuous clock mode. Refer to RX FIFO section for non-continuous clock mode.
- Clock Mode – Select DC (dual clock).
- Misc Signals – Select disabled (unchecked).

This module takes serial CSI-2/DSI data and outputs byte data after de-serialization in MIPI High Speed mode. The .ipx file included in the project (rx_dphy/rx_dphy.ipx) can be used to reconfigure the IP as per the user configuration requirements. If you are creating this IP from scratch, it is recommended to set the design name to *rx_dphy* so that you do not need to modify the instance names of these IPs in the top file as well as the simulation setup file. Otherwise, you need to modify the names accordingly.

3.1.1. RX FIFO

RX FIFO is useful especially in non-continuous clock mode and the continuous byte clock cannot have the exact same frequency as the non-continuous byte clock used in D-PHY RX IP. It resides before the word aligner.

3.1.1.1. Non-Continuous Clock Mode

In this case, RX FIFO configuration depends on the relationship between the non-continuous byte clock in D-PHY RX IP and the continuous byte clock, which is most likely generated by GPLL. The non-continuous byte clock is used to write the data to RX FIFO and the continuous byte clock is used to read the data from RX FIFO.

Continuous byte clock = non-continuous byte clock

In this case, the minimum configuration of RX FIFO is recommended (LUT based, Depth = 16, type = SINGLE, Packet Delay = 1, Clock Mode = DC).

Continuous byte clock < non-continuous byte clock

In this case, type = SINGLE and Packet Delay = 1 is recommended and others depend on the frequency ratio between these two clocks. When the clock speed difference gets larger, the required depth of RX FIFO gets larger. First, it is important to know the horizontal blanking period of the incoming RX channel. For example, in case that one-line active video period is 40 us and the horizontal blanking is 4 us, then we have 10 % of extra time to process the active data. This means the continuous byte clock can be as slow as ~-10% compared to the non-continuous byte clock to avoid RX FIFO overflow.

Continuous byte clock > non-continuous byte clock

There are two options in this case:

- Use type = SINGLE with large Packet Delay

Set the Depth large enough to contain the necessary data to avoid RX FIFO underflow after FIFO read begins after the time specified by Packet Delay. In general, Packet Delay must be set close the depth of the RX FIFO. This configuration can be used when we have enough time interval between the last active line and the frame end short packet so that the frame end short packet is not written to RX FIFO while it still contains the last active line of video data.

- Use type = QUEUE with Number of Queue Entries = 2

This is useful when the time interval between the last active line and frame end short packet is short or unknown. Depth must be set large enough to contain one active line data (plus some more for short packet data). This mode is also useful when line start, and the line end short packets exist in the incoming RX stream. In that case, Number of Queue Entries = 4 and extra depth is required (one line plus two short packet data). FIFO read begins after each HS data transaction is completed. EBR must be used. Counter Width is determined by the amount of the one-line video data plus extra overheads by preceding HS zero data and trail byte in the end of HS transmission.

- Frequency relationship is unknown

When the continuous byte clock is within the certain range against the non-continuous byte clock (for example, two clocks come from different clock sources which have ppm tolerance), we have no idea which clock is faster. The simplest way is to use type = SINGLE with setting Packet Delay to the midpoint of FIFO depth when the tolerance is in ppm level. QUEUE can also be used as described above.

In case you do not have detailed information regarding RX data (whether containing lane start/end short packet, interval of the horizontal blanking period against active line period), the safest way is to set the continuous byte clock faster than the non-continuous byte clock and use type = QUEUE with Number of entries = 4 even though it might require more EBR resources comparing to type = SINGLE.

3.2. byte2pix

This module must be created for the RX channel according to D-PHY type, data type, the number of lanes, RX Gear, and others. Figure 3.3 shows an example of IP interface settings in Radiant Software for the byte2pixel IP. You can use the byte2pix.ipx file (*byte2pix/byte2pix.ipx*) included in the sample project and reconfigure according to your needs. Refer to Byte-to-Pixel Converter IP Core - Lattice (FPGA-IPUG-02079-1.4) for details.

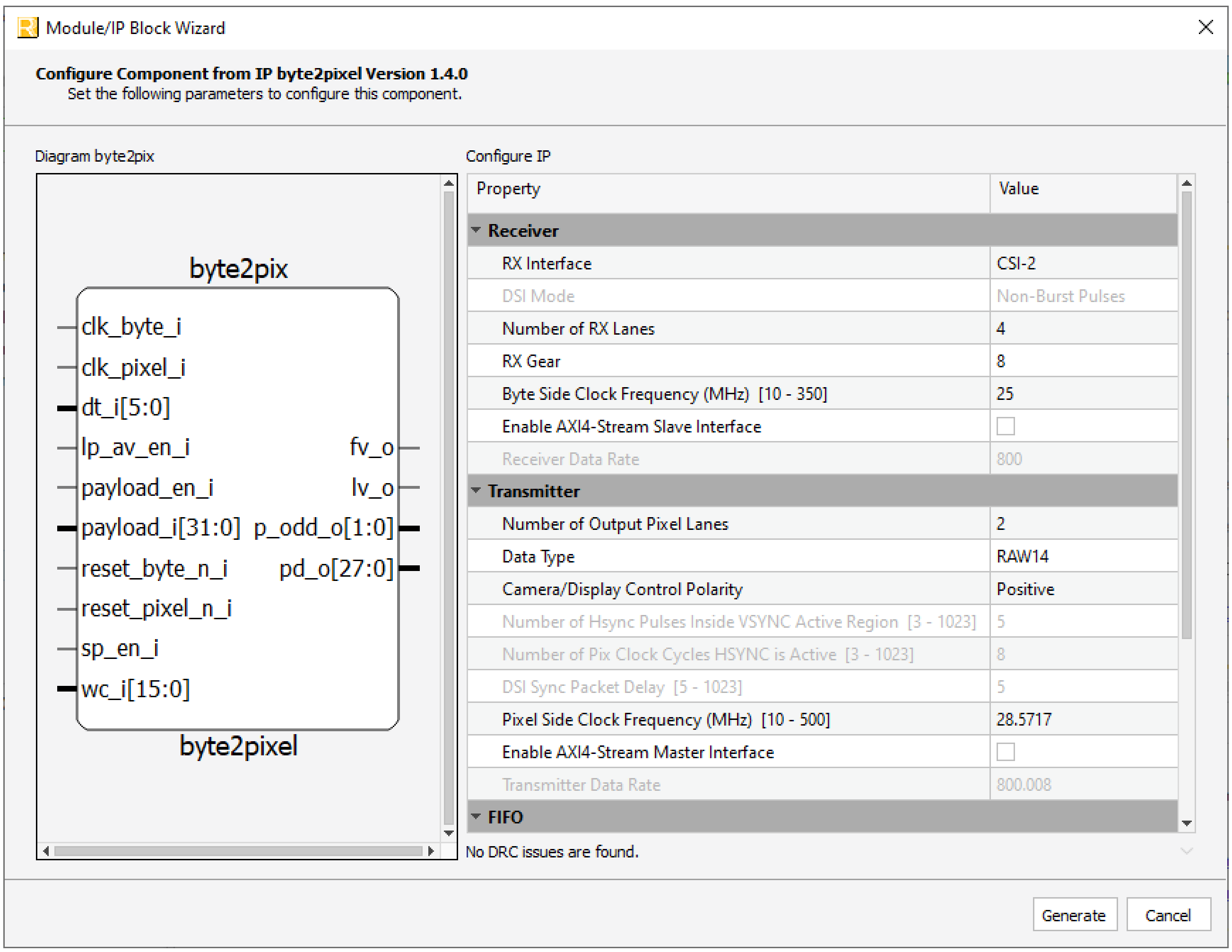

Figure 3.3. byte2pixel IP Creation in Radiant Software

The following provides the guidelines and parameter settings required for this reference design.

- RX Interface – Select DSI or CSI-2. Set the same type as RX D-PHY IP.
- DSI Mode – Always Non-Burst Pulses for CSI-2. Non-Burst Events or Burst applicable to DSI Interface only.
- Number of RX Lanes – Select 1, 2, or 4. Set the same value as RX D-PHY IP.

- RX Gear – Select 8 or 16. Set the same value as RX D-PHY IP.
- Byte Side Clock Frequency – Set the same value as is obtained in the RX D-PHY IP.
- Enable AXI4-Stream Slave Interface – Select disabled (unchecked).
- Number of Output Pixels Lanes – Select 1, 2, or 4.
- Data Type – Select RGB888 for DSI and RGB888, RAW8, RAW10, RAW12, or RAW14 for CSI-2. Others are not supported in this reference design.
- Camera/Display Control Polarity – Select Positive. Polarity setting of sync and data enable is done outside of this IP.
- Pixel Side Clock Frequency – Enter the appropriate value (can be calculated from the mipi2lvds_clock_LFCPNX.xlsx).
- Enable AXI4-Stream Master Interface – Select disabled (unchecked).
- Manual Adjust – Select disabled (unchecked).
- Word Count – Enter the appropriate value as per the following equation:
    - Word Count = (NUM_PIXELS * PD_BUS_WIDTH)/8
    - e.g., in case of 1080P with RAW10, the value will be 1920 x 10/8 = 2400.
- Enable Debug Ports – Select disabled (unchecked).

The Byte-to-Pixel Converter IP converts the D-PHY CSI-2/DSI standard-based byte data stream to standard pixel data format. The .ipx file included in the project (byte2pix/byte2pix.ipx) can be used to reconfigure the IP as per the user configuration requirements. If you are creating this IP from scratch, it is recommended to set the design name to *byte2pix* so that you do not need to modify the instance name of this IP in the top-level design as well as in the simulation setup file. Otherwise, you need to modify the names accordingly.

This module receives byte data from RX D-PHY along with packet enable signal, among others, and reorganizes the data to form the desired pixel data via FIFO according to the data type specified. FIFO is used as a data buffer as well as clock domain conversion. Byte data are written to FIFO in byte clock domain and read in pixel clock domain. Pixel clock is provided from TX LVDS module.

Depending on the D-PHY type (DSI or CSI-2), different types of sync and data enable signals are generated. Figure 3.4 shows the interface timing diagram for DSI. VSYNC, HSYNC, and DE are generated based on short packet enable and payloads enable signals. In case of DSI, these signals are sent to LVDS TX module as-is. Depending on the configuration, output data (pd_o) could have one, two, or four pixels per pixel clock. Figure 3.5 shows global output timing for DSI. Due to the clock domain crossing, the pulse length and/or sync signal intervals may vary slightly.

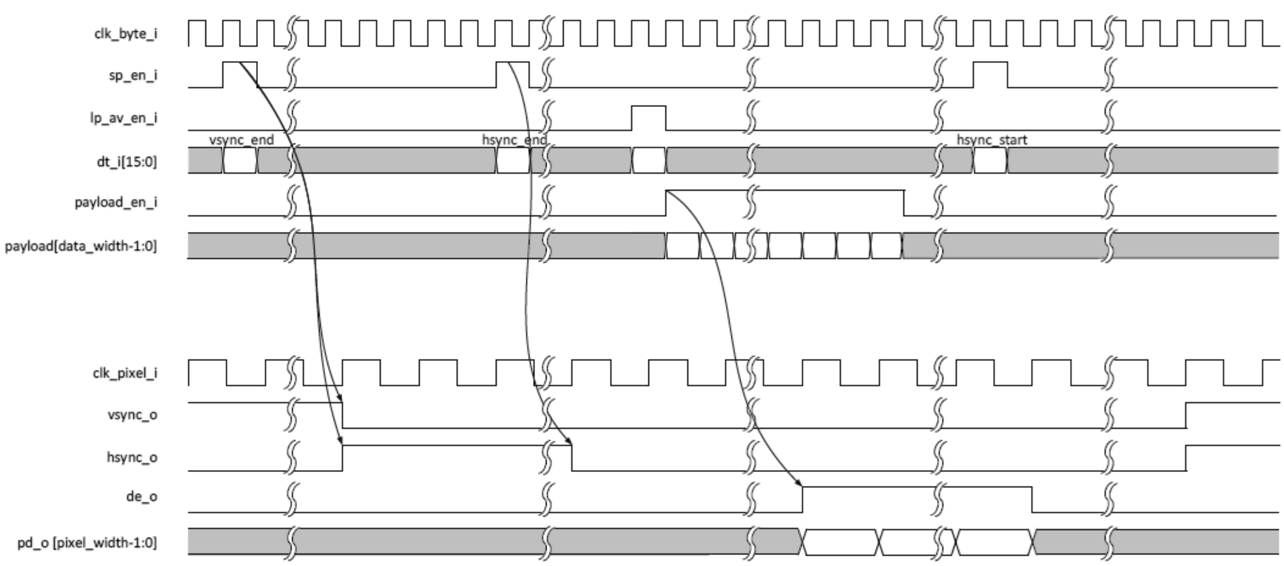

Figure 3.4. Interface Timing Diagram for DSI

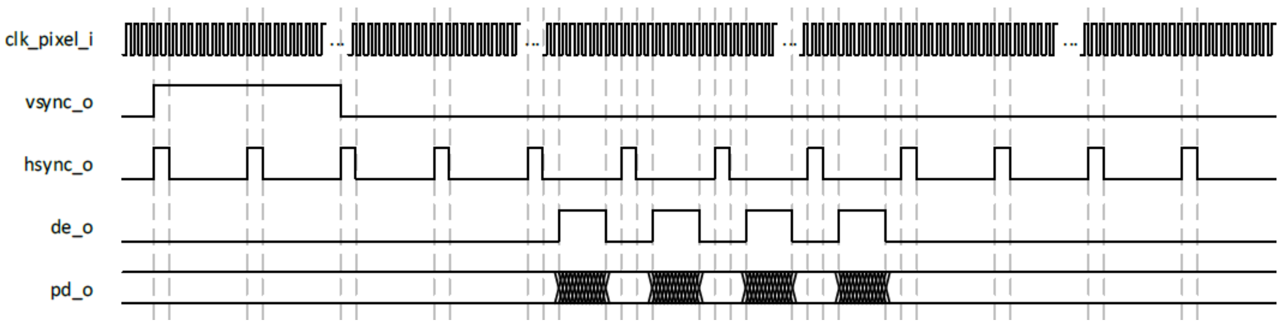

Figure 3.5. Global Output Timing Diagram for DSI

Figure 3.6 shows the interface timing diagram for CSI-2. FV and LV are generated based on short packet enable and payload enable signals. Figure 3.7 shows global output timing for CSI-2. In case of CSI-2, sync signals (VSYNC, HSYNC) have to be created from FV and LV to match the interface format of LVDS. Also, in case of RAW8/RAW10/RAW12, RGB pixel data have to be created from RAW data. For these purposes, an additional module (rgb2rgb or raw2rgb) is required for CSI-2 between byte2pixel and tx_lvds. In case of two TX channel outputs, output data (pd_o) has two pixels per pixel clock.

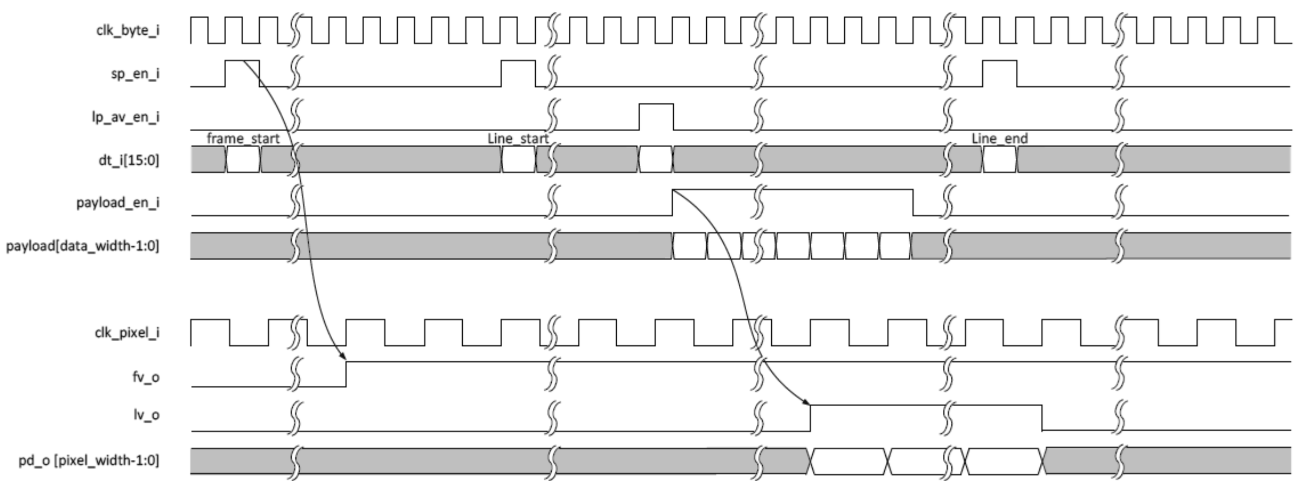

Figure 3.6. Interface Timing Diagram for CSI-2

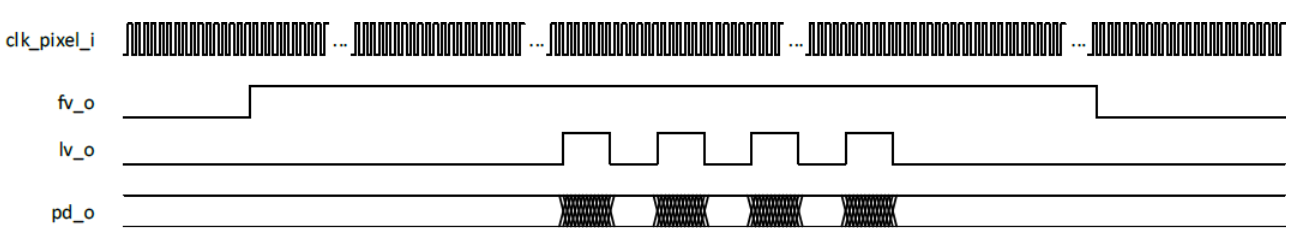

Figure 3.7. Global Output Timing Diagram for CSI-2

3.3. rgb2rgb

This module is instantiated for CSI-2 RX with RGB888 and has two major functions: sync signal generation and video data trimming.

3.3.1. Sync Signal Generation

Since sync signal format is different between CSI-2 and LVDS, VSYNC and HSYNC signals have to be created from FV and LV comes from byte2pix. For that purpose, the following parameters are prepared:

- HFP (Horizontal Front Porch)
- HS_LENGTH (Horizontal Sync Length)
- VFP (Vertical Front Porch)
- VS_LENGTH (Vertical Sync Length)

The above parameter values can be changed through register writes when USE_I2C is defined. Otherwise, the values specified in synthesis_directives.v are applied. Note that (HFP + HS_LENGTH) must be shorter than the horizontal blanking period. Otherwise, HSYNC active period and DE active period overlap, and we cannot expect the anticipated image on the display.

If you don't know the length of the horizontal blanking period, a small value should be set. For example, HFP = 2, HS_LENGTH = 2, and others, as long as the display accepts it. In addition, (VFP + VS_LENGTH) must be smaller than the vertical blanking period for the same reason. HFP and VFP counts begin at the end of non- trimmed active pixel and active line. That means the actual blanking periods are larger than HPF or VFP if trimming happens on the right edge or bottom edge.

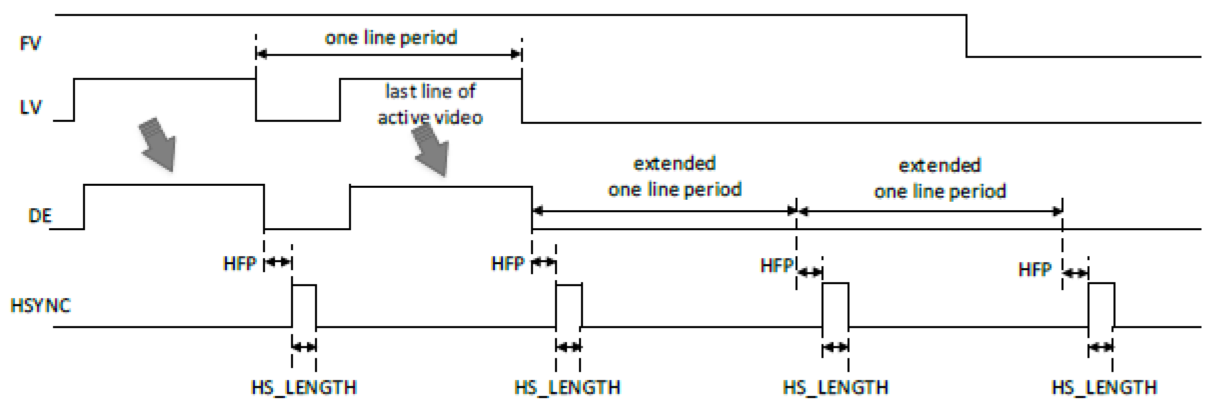

Figure 3.8. HSYNC Generation

During the vertical blanking period, HSYNC is generated by extending the line interval obtained by LV falling edges during the active line period as shown in Figure 3.8. One potential problem, however, is that HSYNC assertion could happen when the first DE of the next frame comes since there is no guarantee that the length of the vertical blanking period is multiple of one horizontal line period. To avoid this situation, a shift register is provided. FV/LV and payload data are all written to the shift register FIFO and comes out after the specified pixel clock cycle delay set by the parameter SR_DELAY. Figure 3.9 shows the masking of HSYNC which overlaps DE period. If the display has the minimum requirement of HBP (Horizontal Blanking Period: period from the end of HSYNC to the beginning of DE), SR_DELAY value should satisfy the following equation:

SR_DELAY > HS_LENGTH + minimum HBP

VSYNC/HSYNC assertions are controlled based on these delayed signals, but HSYNC assertion is disabled when the original LV is active. By changing this parameter value, you can change the HSYNC mask period. This parameter can also be changed through register write when USE_I2C is defined.

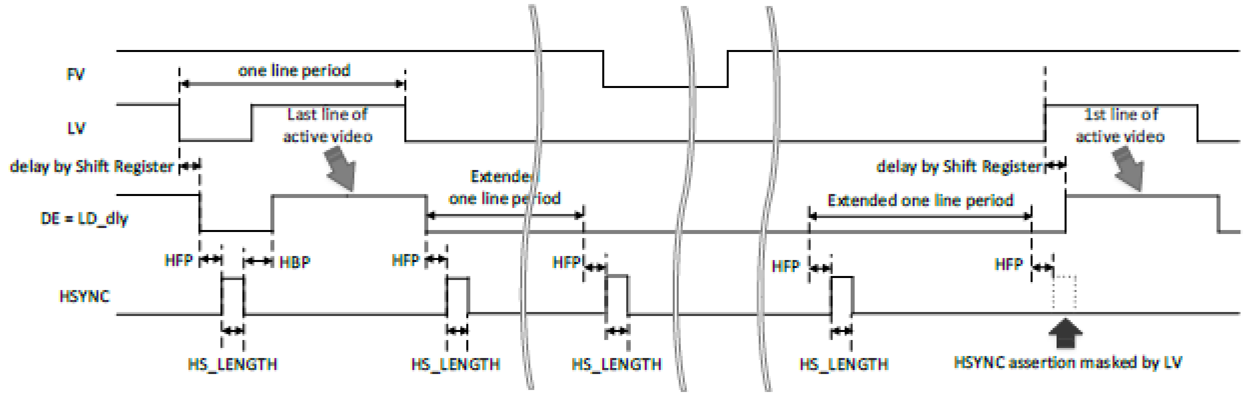

Figure 3.9. HSYNC masked by LVs

Since VSYNC generation is based on the assertion timing of HSYNC, and HSYNC assertion is based on the falling edge of LV, VSYNC of the current outgoing frame requires the previous incoming video timings. Therefore, no VSYNC assertion happens for the first incoming video frame and DE is masked for that frame. Valid VSYNC and video data outputs begin from the second incoming frame.

3.3.2. Active Data Trimming

In some cases, video data from sensor devices have an area that is larger than what is required by the display. For this reason, edge trimming capability is provided using following parameters:

- LEFT_TRIM (trimming pixels from the left edge)
- H_TX_PEL (active pixel count)
- TOP_TRIM (trimming lines from the top edge)
- V_TX_LINE (active line count)

The above parameter values can be changed through register write when USE_I2C is defined. Otherwise, the values specified in synthesis_directives.v are applied. Note that (LEFT_TRIM + H_TX_PEL) must not exceed the original horizontal active pixel count, otherwise output image is corrupted. In addition, (TOP_TRIM + V_TX_LINE) must not exceed the original active line count for the same reason.

3.4. raw2rgb

This module is instantiated for CSI-2 RX with RAW8/RAW10/RAW12/RAW14 and has three major functions: RGB data creation, sync signal generation and video data trimming.

3.4.1. RGB Data Creation

In the case of RAW data format, one pixel contains only one-color component of RGB. Therefore, missing color components must be created by interpolation to form RGB888. Figure 3.10 shows four scenarios of Bayer patterns of the color components at the top-left pixels come from the image sensor.

R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B

Bayer pattern 00

G	R	G	R	G	R
B	G	B	G	B	G
G	R	G	R	G	R
B	G	B	G	B	G

Bayer pattern 01

G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G

Bayer pattern 10

B	G	B	G	B	G
G	R	G	R	G	R
B	G	B	G	B	G
G	R	G	R	G	R

Bayer pattern 11

Figure 3.10. Bayer Pattern of RAW Data

Missing color components are created by interpolation using the neighborhood pixels. Figure 3.11 shows examples of interpolations to create all three-color components for the center pixel. Either 2-pixel averaging, or 4-pixel averaging is applied, depending on the location of the pixel, to create the missing color components.

B	G	B
G	$\boxed{R}$	G
B	G	B

$R_{(interpolated)} = R_{(center)}$

$G_{(interpolated)} = (G_{(top)} + G_{(bottom)} + G_{(left)} + G_{(right)}) / 4$

$B_{(interpolated)} = (B_{(top-left)} + B_{(top-right)} + G_{(bottom-left)} + G_{(bottom-right)}) / 4$

G	B	G
R	$\boxed{G}$	R
G	B	G

$R_{(interpolated)} = (R_{(left)} + R_{(right)}) / 2$

$G_{(interpolated)} = G_{(center)}$

$B_{(interpolated)} = (B_{(top)} + B_{(bottom)}) / 2$

Figure 3.11. RGB Data Creation from RAW Data

Due to the necessity of vertical interpolation, this module has two line buffers with the depth specified by the directive LB_DEPTH_*. In case of edge and corner pixel interpolations (top line, bottom line, left edge, right edge), *mirroring* is applied to cover the non-existing pixels. For example, there are no top pixels for the first line pixel interpolation and the bottom (second line) pixels are used instead of the non-existing top pixels. Same method applies to left edge, right edge, and bottom line pixel interpolations. Note that the first line output is roughly aligned with the second incoming line due to the line buffering delay.

3.4.2. Sync Signal Generation

Scheme is same as CSI-2 RGB888 input except for the non-existence of the shift register. In case of RAW data processing, a single line process delay happens due to the line buffer mentioned in the Refer to the 3.4.1 RGB Data Creation section for pattern details.

section, the HSYNC masking feature is not required using the shift register. Refer to the 3.4.2 Sync Signal Generation section for details.

3.4.3. Active Data Trimming

Scheme is same as CSI-2 RGB888 input. Refer to the 3.4.3 Active Data Trimming section for details.

3.5. tx_lvds

You must create this module according to channel conditions, such as number of lanes, bandwidth, and others. Figure 3.12 shows an example IP interface setting in Radiant Software for the OpenLDI/LVDS Transmitter Interface IP. You can use the ipx file (tx_lvds/tx_lvds.ipx) included in the sample project and re-configure according to your needs. Refer to OpenLDI/FPD-LINK/LVDS Transmitter IP Core (FPGA-IPUG-02117-1.0) for details.

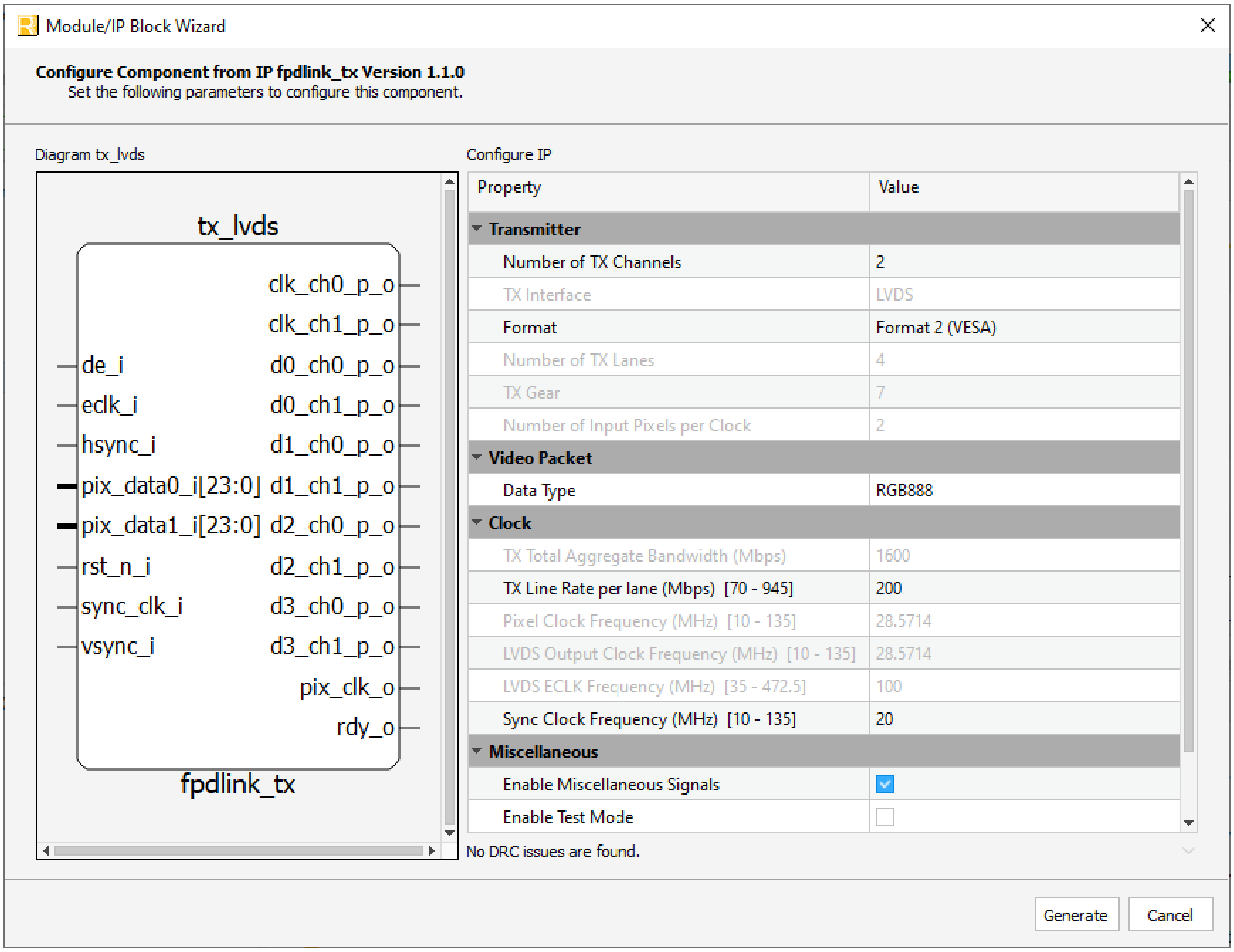

Figure 3.12. tx_lvds IP Creation in Radiant Software

The following provides the guidelines and parameter settings required for this reference design.

- Number of TX Channels – Set according to channel configuration. Must match NUM_TX_CH_* setting. Refer to Table 1.1 for possible TX channel setting.
- Format – Select Format 2 (VESA).
- Data Type – Always select RGB888.
- TX Line Rate – Set the value derived from the TX Lane Bandwidth equation below.
- Sync Clock Frequency – Select the value, which is equal to or lower than the slowest clock in the system
- Enable miscellaneous status signals – Must be enabled (checked).
- Enable test mode – Must be disabled (unchecked).

This module takes the pixel data and outputs LVDS data after 7:1 serialization. If you are creating this IP from scratch, it is recommended to set the design name to tx_lvds and module name to tx_lvds so that you do not need to modify the instance name of this IP in the top-level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

TX Line Rate is derived from the following equation:

$$TX\_lane\_bandwidth = \frac{RX\_lane\_bandwidth * number\_of\_RX\_lane * 7}{RX\_pixel\_bus\_width * number\_of\_TX\_channel}$$

Example 1 : CSI-2 RAW10 RX with 4-lane at RX lane bandwidth = 200

Mbps TX lane bandwidth = (200 x 4 x 7) / (10 x 1) = 560 Mbps.

Example 2: DSI RGB888 RX with 4-lane at RX lane bandwidth = 1200 Mbps with dual channel TX outputs

TX lane bandwidth = (1200 x 4 x 7) / (24 x 2) = 700 Mbps.

Figure 3.13 and Figure 3.14 show the output data timings of single and dual channel configuration with RGB888. RGB888 uses four data lanes per channel.

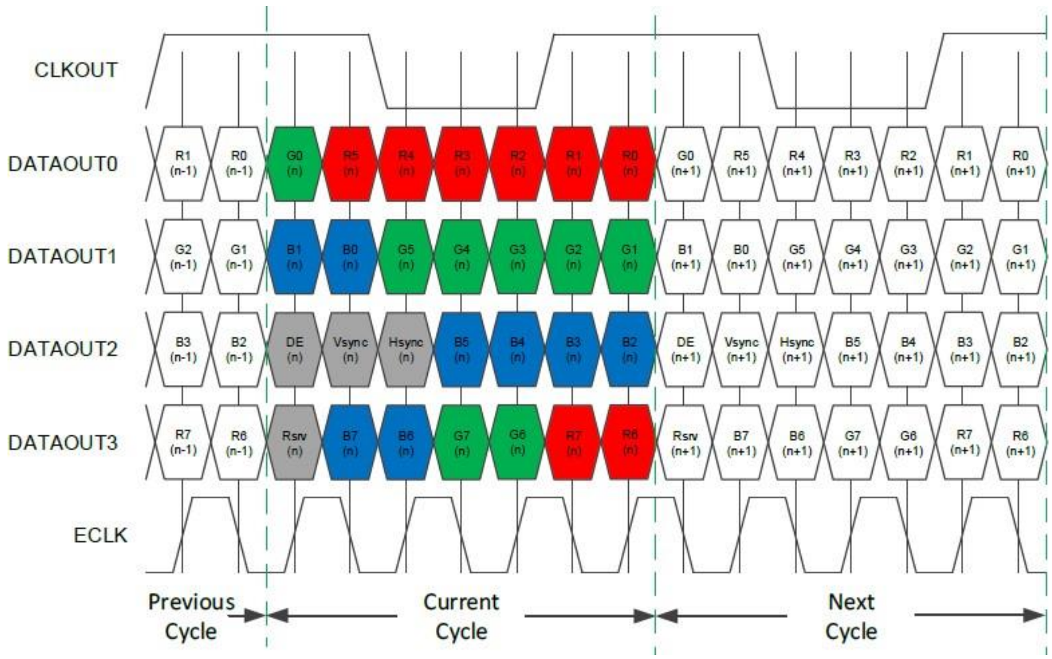

Figure 3.13. Single Channel LVDS Output of RGB888

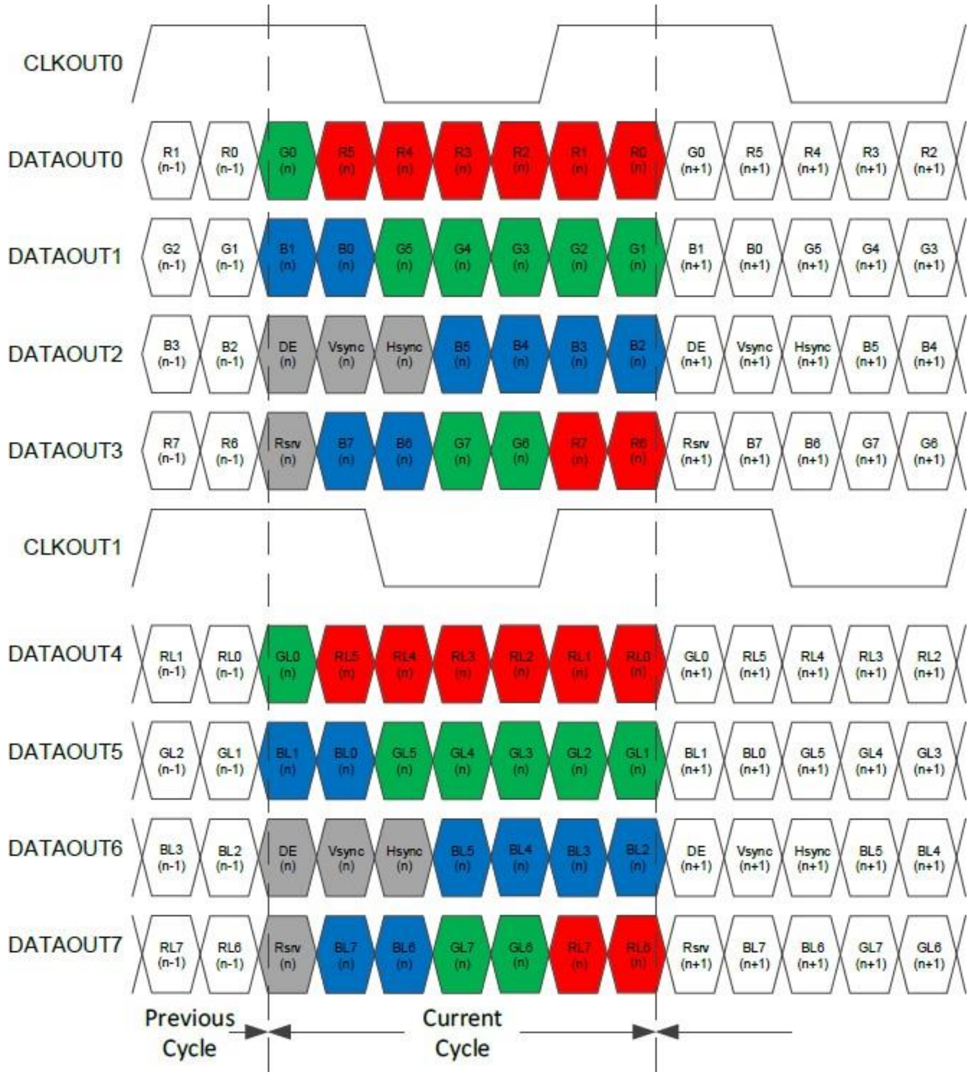

Figure 3.14. Dual Channel LVDS Output of RGB888

3.6. int_gpll

You must create GPLL module to feed the edge clock to tx_lvds. Additionally, the continuous byte clock could be generated when RX D-PHY is in HS_LP mode. You can use the ipx file (int_gpll/int_gpll.ipx) included in the sample project and re-configure according to your needs. In case that you make this IP from scratch, it is recommended to set the design name to *int_gpll* so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

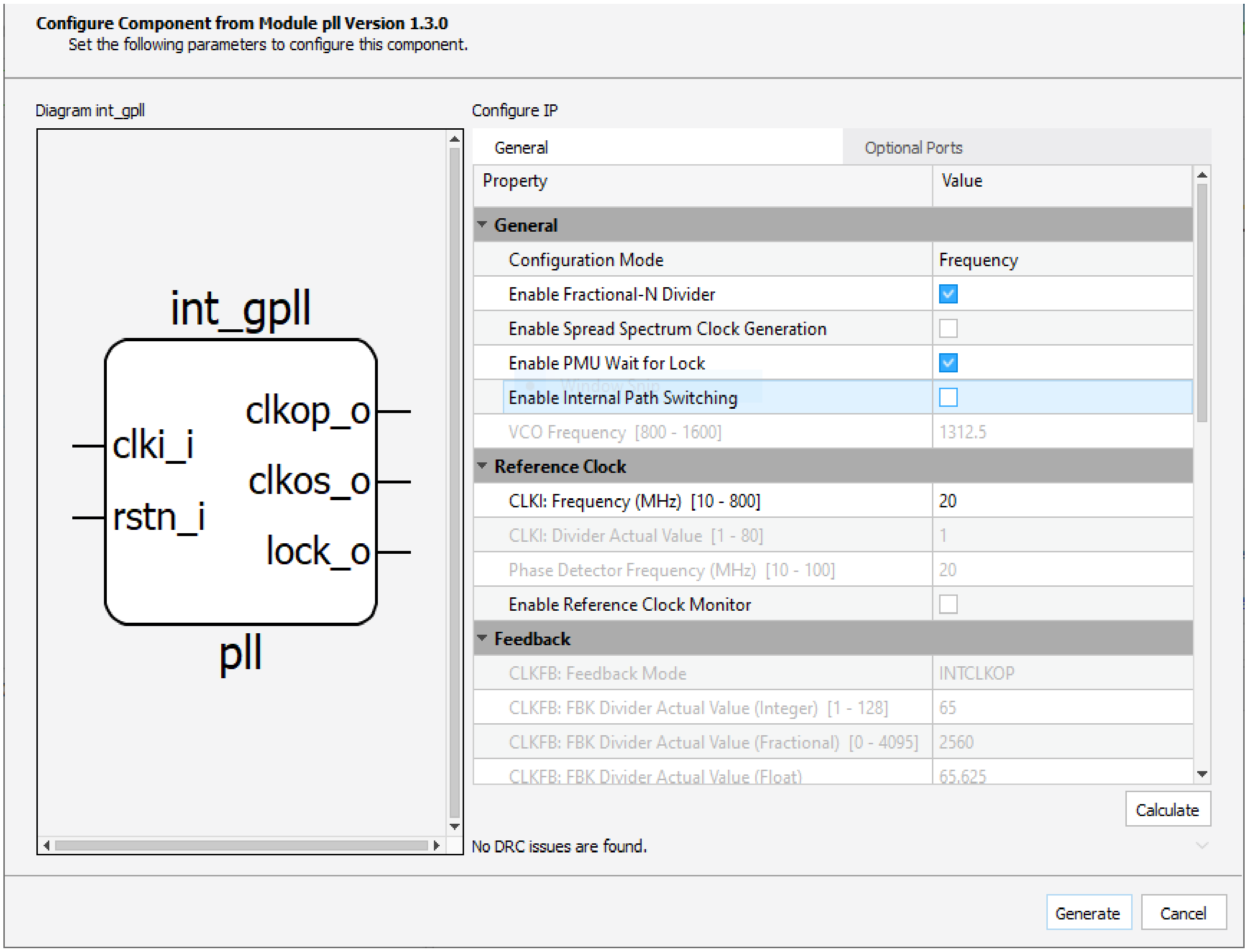

Figure 3.15. GPLL IP Creation#1 in Radiant Software

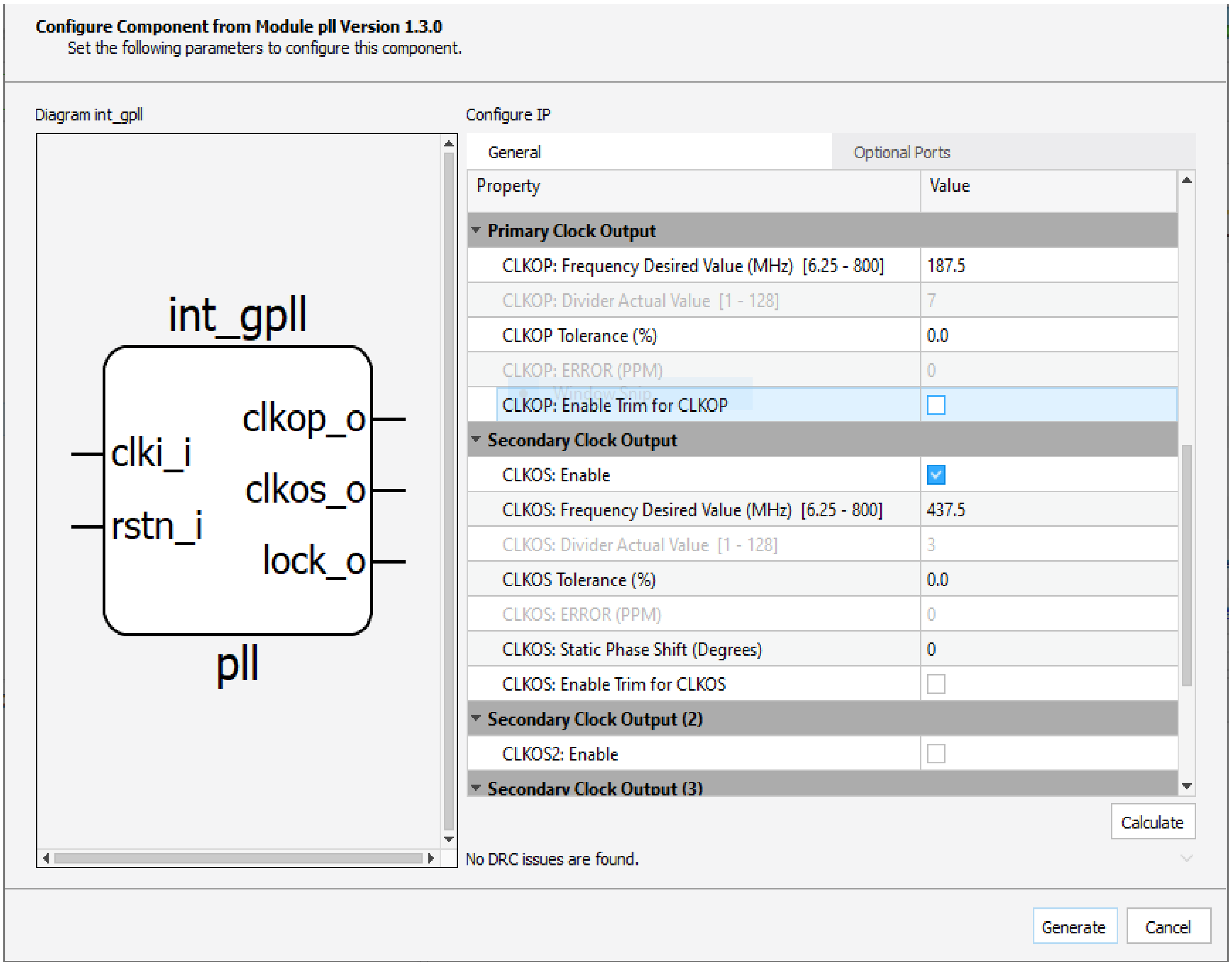

Figure 3.16. GPLL IP Creation#2 in Radiant Software

The TX edge clock frequency is half of the TX lane bandwidth set for tx_lvds. This clock must be assigned to CLKOP. If the byte clock has to be generated, assign it to CLKOS. Note that the VCO clock of GPLL has to be a common multiple of these two clocks. There is a case wherein TX edge clock and byte clock cannot be generated together. In this instance, give up generating the byte clock by GPLL and perform either of the following options:

- Change RX D-PHY to HS_ONLY mode; or
- Feed the continuous byte clock directly from one of the available I/O pins.

If the RX D-PHY is in HS_ONLY mode and GPLL can generate the TX edge clock using the byte clock as the input clock of GPLL (clki_i), the external reference clock is not necessary. int_gpll instantiation based on the configuration is shown below as a code snippet of mipi2lvds_LFCPNX.v

```
//////////////////////////////////////////////////////////////////////////////
///// GPLL instantiation                                                /////
///// User has to modify this if necessary                             /////
//////////////////////////////////////////////////////////////////////////////
`ifdef EXT_REF_CLK
        `ifdef RX_CLK_MODE_HS_LP
        int_gpll pll_inst (
                    .clki_i     (ref_clk_i),        // ref clock
                    .clkop_o (pll_eclk),            // tx edge clock
                    .clkos_o (pll_byte_clk),   // byte clock
                    .lock_o   (pll_lock)
                    );
    `elsif RX_CLK_HS_ONLY
        int_gpll pll_inst (
                    .clki_i (ref_clk_i),
                    .clkop_o (pll_eclk),
                    .lock_o (pll_lock)
                   );
    `endif
`elsif RX_CLK_HS_ONLY
        int_gpll pll_inst (
                    .clki_i (ref_clk_byte_fr),
                    .clkop_o (pll_eclk),
                    .lock_o (pll_lock)
                   );
`endif
//////////////////////////////////////////////////////////////////////////////
```

If you can use ref_clk_i (or simply divided clock of this) as byte_clk in HS_LP mode, you should disable CLKOS and add the following in mipi2lvds_LFCPNX.v:

```
assign pll_byte_clk = ref_clk_i;   // ref_clk_i could be simply divided clock of this
```

More than one GPLL can be used if the required clock frequencies for the rx_clk_byte_fr and pll_eclk cannot be generated using a single GPLL.

3.7. i2c_slave

This module is instantiated when USE_I2C is defined and enables you to change CSI-2 related parameters on the fly through I^{2}C connections. I^{2}C Hardened IP is instantiated and used as an I^{2}C slave device. You can use the .ipx file (*i2c_s/i2c_s.ipx*) included in the sample project and re-configure. In case that you create this IP from scratch, it is recommended to set the design name to *i2c_s* so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

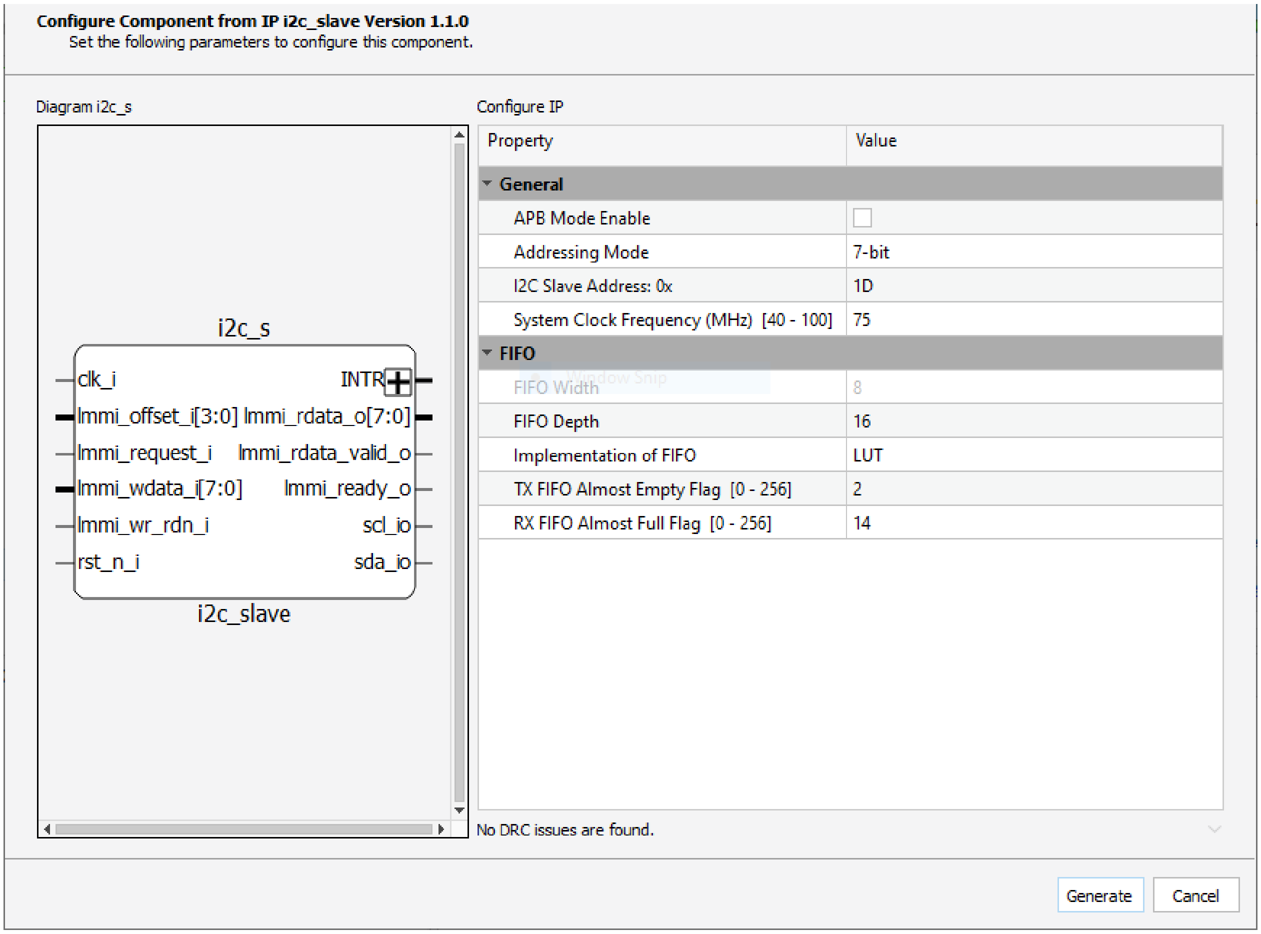

Figure 3.17. I2C IP Creation in Radiant Software

Figure 3.17 shows basic settings of I^{2}C IP. You can change the settings according to your design's needs.

This module is equipped with parameter registers of 4-bit address area of I^{2}C sub address shown in Table 3.1. I2C Slave Register Map.

Table 3.1. I2C Slave Register Map

Sub address	Name	Bits	Description
0	SW Reset	[0]	Software reset register. When this is active, all modules except for i2c_slave are in reset condition. Active low.
1	SR Delay	[7:0]	Shift Register delay register. Sync and data from byte2pixel are delayed before processed in rgb2rgb. The value must be 8'd2 – 8'd255.
2	Bayer Pattern	[1:0]	Bayer Pattern register. The value must be 2'00, 2'b01, 2'b10, or 2'b11.
3	HFP	[7:0]	Horizontal Front Porch register. The value must be 8'd1 – 8'd255.
4	HS_LENGTH	[7:0]	HSYNC Length register. The value must be 8'd1 – 8'd255.
5	VFP	[5:0]	Vertical Front Porch register. The value must be 6'd1 – 6'd63.
6	VS_LENGTH	[5:0]	VSYNC Length register. The value must be 6'd1 – 6'd63.
7	LEFT_TRIM	[5:0]	Left Edge Trim register. The value must be 6'd0 – 6'd63.
8	H_TX_PEL_LSB	[7:0]	Horizontal Active Pixel register. The value must be 12'd1 – 12'd4095.
9	H_TX_PEL_MSB	[11:8]	
10	TOP_TRIM	[5:0]	Top Edge Trim register. The value must be 6'd0 – 6'd63.
11	V_TX_LINE_LSB	[7:0]	Vertical Active Line register. The value must be 12'd1 – 12'd4095.
12	V_TX_LINE_MSB	[11:8]	

13	—	N/A	Reserved
14	—	N/A	Reserved
15	—	N/A	Reserved

Notes:

1. All registers less than 8-bit data width are aligned to LSB of 8-bit data area.
2. All registers are set to the default values specified by corresponding directives defined in synthesis_directives.v
3. Software reset works as the system reset (reset_n_i) for all modules other than i2c_slave, therefore you can assert this while updating other I²C registers, then release Software reset upon completing the register update to avoid an unexpected operation during register update. Refer to 3.3 rgb2rgb and 3.4 raw2rgb sections for register details.

3.8. int_osc

In the sample design, a 75 MHz clock is generated by the on-chip oscillator and fed to i2c_slave module or/and RX D-PHY IP. You can use the ipx file (int_osc/int_osc.ipx) included in the sample project and re-configure according to your needs. In case you generate this IP from scratch, it is recommended to set the module name to int_osc so that you do not need to modify the instance name of this IP in the top-level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

Figure 3.18 shows an example of IP interface settings in Lattice Radiant software for the int_osc IP.

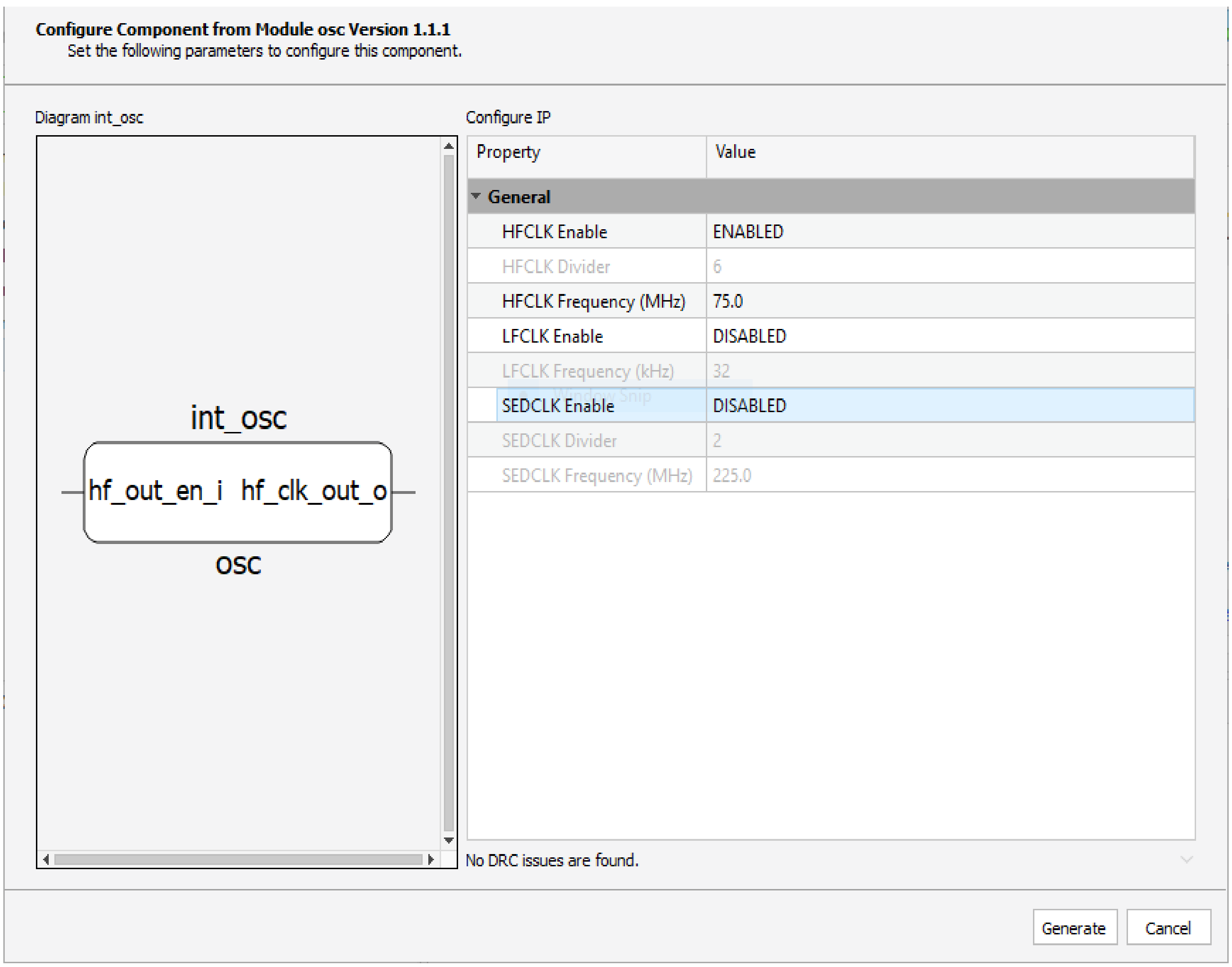

Figure 3.18. int_osc IP Creation in Lattice Radiant

4. Design and File Modification

This RD is based on version 1.3.0 of the RX D-PHY IP, version 1.4.0 of the Byte-to-Pixel Converter IP, and version 1.1.0 of the OpenLDI/LVDS Transmitter Interface IP. Due to the limitation of this IPs, some modifications are required depending on user configuration in addition to two directive files (synthesis_directives.v, simulation_directives.v).

4.1. Top Level RTL

If you use I^{2}C Slave module with a clock other than the internal oscillator clock, you will have to disable the int_osc (internal oscillator) instantiation and connect an appropriate clock to clk_i port of i2c_slave.

In case of HS_LP mode to use the external reference clock can be used as byte_clk without using GPLL, you have to modify the related code as described in the 3.6 int_gpll section. In addition, instance names of RX D-PHY, Byte-to-Pixel, TX LVDS IP have to be modified if you created these IP with different names.

5. Design Simulation

The script file (*mipi_to_lvds_LFCPNX_msim.do*) and testbench files are provided to run the functional simulation by ModelSim. If you keep the current design names and instance names when RX D-PHY, Byte-to-Pixel, TX LVDS, GPLL, and I^{2}C IPs are created in radiant software, the following are the only changes required in the script file:

- Radiant installation directory path
- User project director

```
### Set Customer's simulation directory ###
set project_dir ~/Desktop/rd_mipi_to_lvds_LFCPNX/
set sim_dir     $project_dir/simulation/lfcpnx/
```

Own Project Directory

Figure 5.1. Script File Modification #1

```
vlog -mfcu \
+incdir+$project_dir/testbench/verilog \
+incdir+$project_dir/source/verilog/lfcpnx \
+incdir+$project_dir/int_gpll/rtl \
+incdir+$project_dir/int_osc/rtl \
+incdir+$project_dir/byte2pix/rtl \
+incdir+$project_dir/tx_lvds/rtl \
+incdir+$project_dir/rx_dphy/rtl \
+incdir+$project_dir/i2c_s/rtl \
$project_dir/source/verilog/lfcpnx/synthesis_directives.v \
$project_dir/testbench/verilog/simulation_directives.v \
$project_dir/source/verilog/lfcpnx/raw2rgb.v \
$project_dir/source/verilog/lfcpnx/rgb2rgb.v \
$project_dir/int_gpll/rtl/int_gpll.v \
$project_dir/int_osc/rtl/int_osc.v \
$project_dir/i2c_s/rtl/i2c_s.v \
$project_dir/rx_dphy/rtl/rx_dphy.v \
$project_dir/byte2pix/rtl/byte2pix.v \
$project_dir/tx_lvds/rtl/tx_lvds.v \
$project_dir/testbench/verilog/mipi2lvds_LFCPNX_tb.v \
$project_dir/source/verilog/lfcpnx/mipi2lvds_LFCPNX.v \
```

Remove if USE_I2C is not defined from synthesis directives

Figure 5.2. Script File Modification #2

You need to modify simulation_directives.v according to your configuration (refer to the 2.2 Simulation Directives section for details). By executing the script in ModelSim, compilation and simulation are executed automatically. The testbench takes all data comparison between the expected data and output data from the RD. Note that data comparison begins from the second input frame in case of CSI-2 input due to VSYNC signal creation delay. It shows following statements while running and doing data comparison:

```
# Frame 1, Line    0, Pixel    0 CH0 Active Data : R = 83, G = 7a, B = 19 matched
#  297690312 DPHY Driving Data
#  297690312 DPHY Lane 0 : Driving with data = 2e
#  297695312 DPHY Driving Data
#  297695312 DPHY Lane 0 : Driving with data = 26
# Frame 1, Line    0, Pixel    1 CH0 Active Data : R = 4a, G = ef, B = 19 matched
#  297700312 DPHY Driving Data
#  297700312 DPHY Lane 0 : Driving with data = 8c
# Frame 1, Line    0, Pixel    2 CH0 Active Data : R = 10, G = d3, B = 22 matched
```

```
#  297705312 DPHY Driving Data
#  297705312 DPHY Lane 0 : Driving with data = 0d
#  297710312 DPHY Driving Data
#  297710312 DPHY Lane 0 : Driving with data = 77
# Frame 1, Line    0, Pixel    3 CH0 Active Data : R = 37, G = c9, B = 2b matched
#  297715312 DPHY Driving Data
#  297715312 DPHY Lane 0 : Driving with data = a8
# Frame 1, Line    0, Pixel    4 CH0 Active Data : R = 5e, G = b8, B = 5d matched
#  297720312 DPHY Driving Data
#  297720312 DPHY Lane 0 : Driving with data = a7
#  297725312 DPHY Driving Data
```

Data comparison is based on the detection of the DE assertion after the deserialization by the testbench. Simulation stops when the data comparison failure happens.

Also, you should find the following statements during the vertical blanking periods since there is no DE assertion:

```
##### tx0 DE de-assertion #####
##### tx0 HSYNC assertion #####
##### tx0 HSYNC de-assertion #####
##### tx0 HSYNC assertion #####
##### tx0 HSYNC de-assertion #####
##### tx0 VSYNC assertion #####
##### tx0 HSYNC assertion #####
##### tx0 HSYNC de-assertion #####
##### tx0 HSYNC assertion #####
##### tx0 HSYNC de-assertion #####
##### tx0 VSYNC de-assertion #####
```

When the simulation is finished, the following statements are displayed (example of CSI-2):

```
##### tx0 HSYNC assertion #####
##### tx0 HSYNC de-assertion #####
#  806980625 DPHY CSI-2 after frame gap
#
# Total pixels checked were 1920 = 2 Frames x 1920 x pels x 5 lines : all matched !!!!!
# Simulation Succeeded !!!!!
# 812080625 TEST END
```

The above is the result after running three frames. In case of CSI-2, DE is not asserted in the first frame and data comparison occurs in the second frame and after.

Figure 5.2 shows the global timing of DSI RGB888 with single channel LVDS output. In case of DSI, all signals are passed through and de-serialized LVDS signals (tx0_des_*) by the de-serializer model in the testbench shows the same behavior as rx_dphy outputs (rx_vsync/hsync/de) and tx_lvds inputs (tx_vsync/hsync/de). In this case, LP period exists before VSYNC.

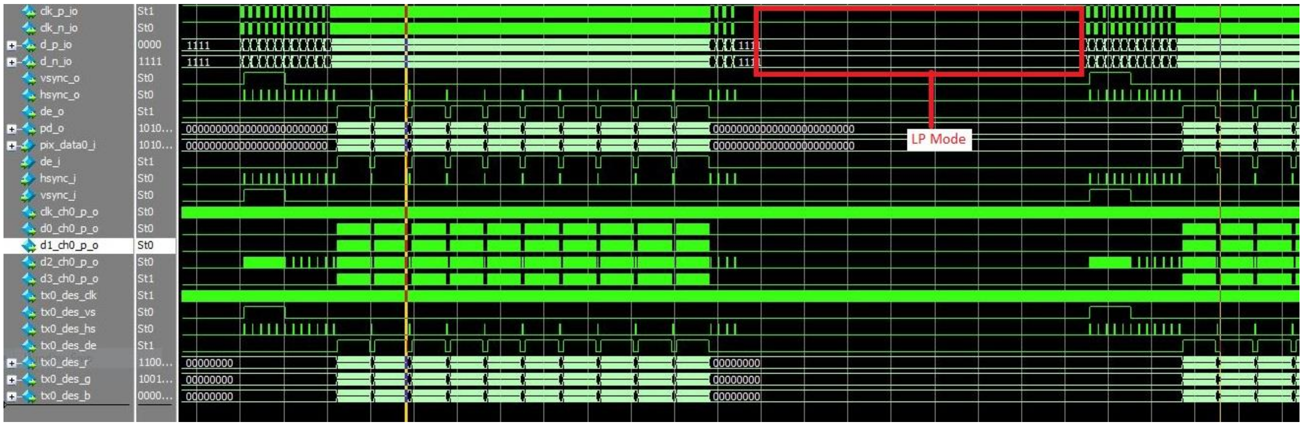

Figure 5.3. Global Timing of DSI RGB888 without Vertical Trimming

Figure 5.3 shows the global timing of CSI-2 RGB888 without vertical trimming. HSYNC is created during the LP mode period between two frames. In this simulation, VFP = 2 and VS_LENGTH = 2 are set.

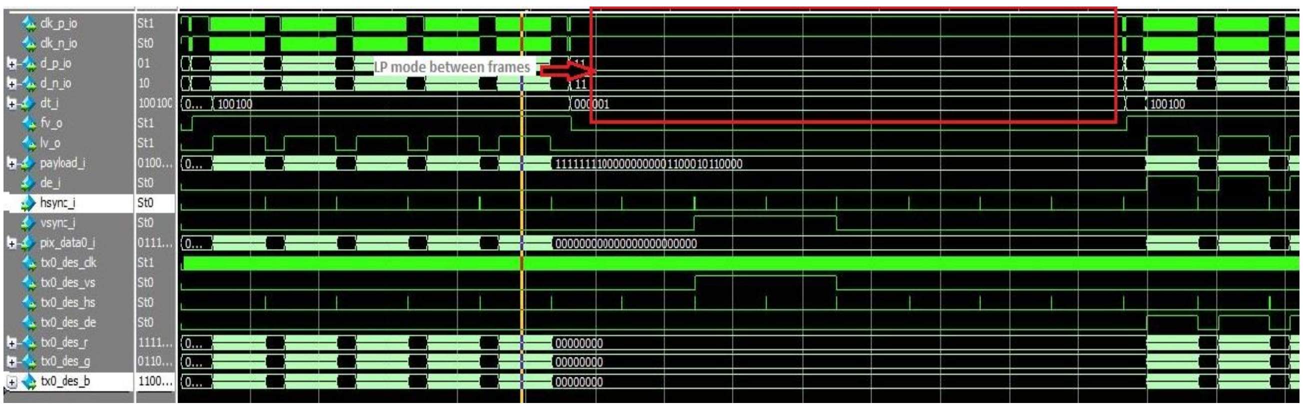

Figure 5.4. Global Timing of CSI-2 RGB888

Figure 5.4 shows the global timing of CSI-2 RAW10 without vertical trimming. DE assertion begins from the second frame after VSYNC creation with VFP = 2 and VS_LENGTH = 2. Due to RAW to RGB conversion, there exists one-line delay between RX data and TX data.

Figure 5.5. Global Timing of CSI-2 RAW10 without Vertical Trimming

Figure 5.5 shows the global timing of CSI-2 RAW10 with vertical trimming with VFP = 2 and VS_LENGTH = 2. The first line and last line are trimmed. The actual VFP is 3 due to the trimming. Before the first frame data are fed, parameter update happens through I²C while software reset (int_rst_n) is asserted. Upon the completion of parameter update, the reset is released and the first frame data input begins.

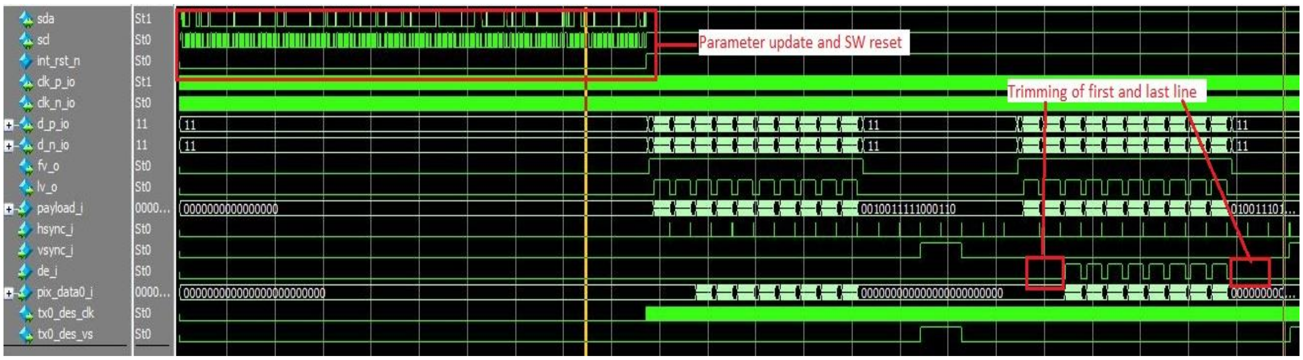

Figure 5.6. Global Timing of CSI-2 RAW10 with Vertical Trimming

6. Known Limitations

The following are the limitations of this reference design:

- Only following data types are supported for MIPI DSI interface: RGB888
- Only following data types are supported for MIPI CSI-2 interface: RGB888, RAW8, RAW10, RAW12, and RAW14

7. Design Package and Project Setup

MIPI DSI/CSI-2 to OpenLDI/LVDS Interface Bridge Reference Design for CertusPro-NX is available on www.latticesemi.com. Figure 7.1 shows the directory structure. synthesis_directives.v and simulation_directives.v are set to configure an example shown below:

- RX : RGB888 four lanes, Gear 8 in continuous clock mode
- TX : RGB888 four lanes with Gear 7

You can modify the directives for own configuration.

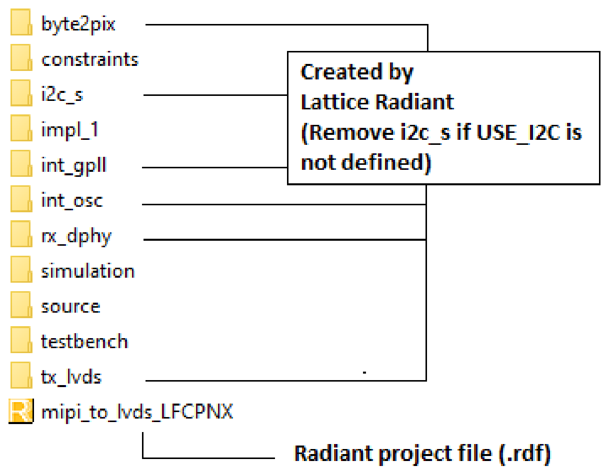

Figure 7.1. Directory Structure

Folders byte2pix, i2c_s, int_gpll, rx_dphy, and tx_lvds are created by Radiant Software for corresponding IPs. The following provides the descriptions of the suffixes:

- _R* : * RAW data width; 8, 10, 12 or 14
- _LB* : * Line Buffer depth; 512, 1024, 2048, or 4096
- _TXC*G** : Number of TX channels; 1 or 2, ** TX Gear; 7

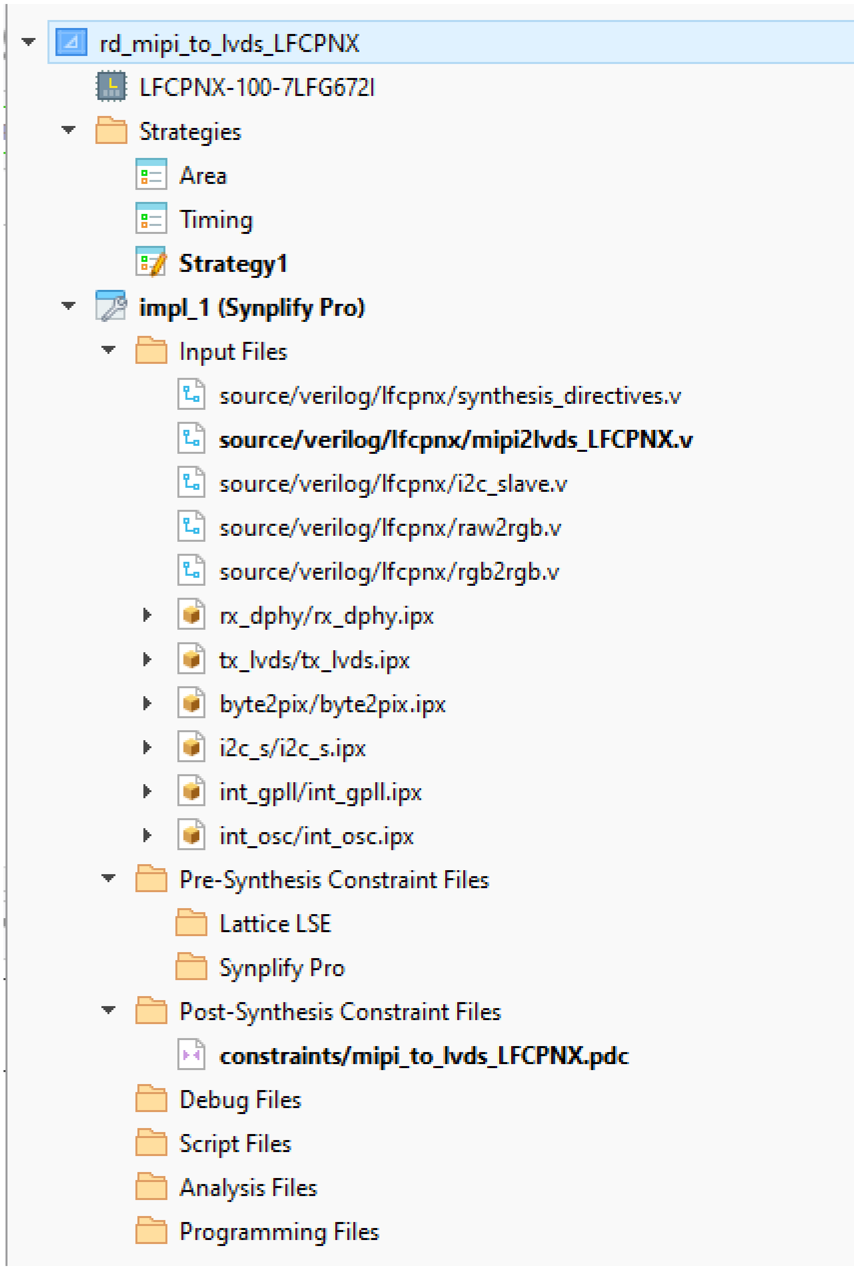

Figure 7.2. Project Files

Figure 7.2 shows design files used in the Radiant project. Radiant Software creates five .ipx files. By specifying mipi2lvds_LFCPNX.v as a top-level design, all unnecessary files are ignored.

8. Resource Utilization

Resource utilization depends on the configurations. Table 8.1 shows resource utilization examples under certain configurations targeting LFCPNX-100. This is just a reference and actual usage may vary.

Table 8.1. Resource Utilization Examples

Configuration	LUT (Utilization/Total)	FF (Utilization/Total)	EBR (Utilization/Total)	I/O (Utilization/Total)
CSI2 RGB888 RX channel with four lanes, Gear 8 to two CSI2 TX channel with Gear 7	3185/79872	1832/80769	8/208	22/299
CSI2 RGB888 RX channel with four lanes, Gear 8 to single CSI2 TX channel with Gear 7	1756/79872	1164/80769	3/208	17/299
DSI RGB888 RX channel with four lanes, Gear 8 to two DSI TX channel with Gear 7	2386/79872	1163/80769	3/208	22/299
DSI RGB888 RX channel with single lane, Gear 8 to single DSI TX channel with Gear 7	854/79872	678/80769	1/208	11/299

References

- MIPI® Alliance Specification for D-PHY Version 1.2
- MIPI® Alliance Specification for Display Serial Interface (DSI) Version 1.2
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.2
- CSI-2/DSI D-PHY Rx IP Core - Lattice Radiant (FPGA-IPUG-02081-1.5)
- Byte-to-Pixel Converter IP Core - Lattice (FPGA-IPUG-02079-1.4)
- OpenLDI/FPD-LINK/LVDS Transmitter IP Core (FPGA-IPUG-02117-1.0)
- I2C Slave IP Core (FPGA-IPUG-02072-1.2)

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow, Tasks, and Simulation Flow, see the Lattice Radiant User Guide.

Technical Support Assistance

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

Revision History

Revision 1.0, March 2022.

Section	Change Summary
All	Initial release.

www.latticesemi.com