4 to 1 Image Aggregation with CrossLink-NX

Reference Design

FPGA-RD-02128-1.3

October 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AMBA	Advanced Microcontroller Bus Architecture
AP	Application Processor
APB	Advanced Peripheral Bus
CSI-2	Camera Serial Interface 2
DDR	Double Data Rate
DE	Data Enable
FSM	Finite state machine
FV	Frame Valid
GPLL	General Purpose PLL
GUI	Graphic User Interface
HBP	Horizontal Back Porch
HFP	Horizontal Front Porch
HS	High Speed
LMMI	Lattice Memory Mapped Interface
LP	Low Power
LV	Line Valid
MIPI	Mobile Industry Processor Interface
PLL	Phase Locked Loop
RD	Reference Design
RTL	Register transfer level
RX	Receiver
TX	Transmitter
VBP	Vertical Back Porch
VFP	Vertical Front Porch

Supported Device and IP

This reference design supports following devices with IP versions shown below.

Device Family	Part Number	Compatible IP
CrossLink™-NX	LIFCL-40 LIFCL-17	D-PHY Receiver IP version 1.5.0 Byte-to-Pixel Converter IP version 1.6.0 PLL IP version 1.8.0 I2C Controller IP version 2.0.0

The IPs above are supported by the Lattice Radiant™ software version 2023.1.1 or later.

1. Introduction

Mobile Industry Processor Interface (MIPI®) Camera Serial Interface 2 (CSI-2) is one of the most popular image sensor interface in the consumer market today. On the other hand, many applications require to handling multiple video images.

The Lattice Semiconductor 4 to 1 Image Aggregation with CrossLink-NX device reference design takes four channels of CSI-2 MIPI data and aggregates them after debayering and outputs parallel RGB data. The MIPI RX module can be realized by a MIPI hard macro IP or soft macro utilizing general DDR modules (D-PHY Soft IP).

1.1. Features

- Four CSI-2 RAW inputs to a single channel RAW or RGB888 parallel data output.
- Image sensors can be configured by CrossLink-NX through I2C I/F.
- RX channel can have one, two, or four lanes with the bandwidth up to 1.5 Gbps per lane as long as the pixel clock frequency is below the device maximum logic speed, 150 MHz to 200 MHz, depending on the design complexity.
- RAW data can be converted to RGB data on all RX channels before the aggregation.
- Image cropping option is available.
- 1/4 of the each RX channel data are aggregated to make the full size of the single image.
- Noise reduction filter is available for RGB data output.

1.2. Block Diagram

Figure 1.1 shows the block level diagram of the 4 to 1 Image Aggregation with CrossLink-NX device reference design of RGB output version. This is a conceptual diagram and not all modules are shown. rx_dphy module is either rx_dphy_h (Hard D-PHY RX IP) or rx_dphy_s (Soft D-PHY RX IP).

There exist two main clock domains for main video data path: byte clock and pixel clock. In most cases, GPLL is required to generate the pixel clock. Also, it might be used to generate the continuous byte clock when RX D-PHY is in non-continuous clock mode. Refer to int_gpll section for GPLL use. I^2C Controller modules configure image sensors after the reset release.

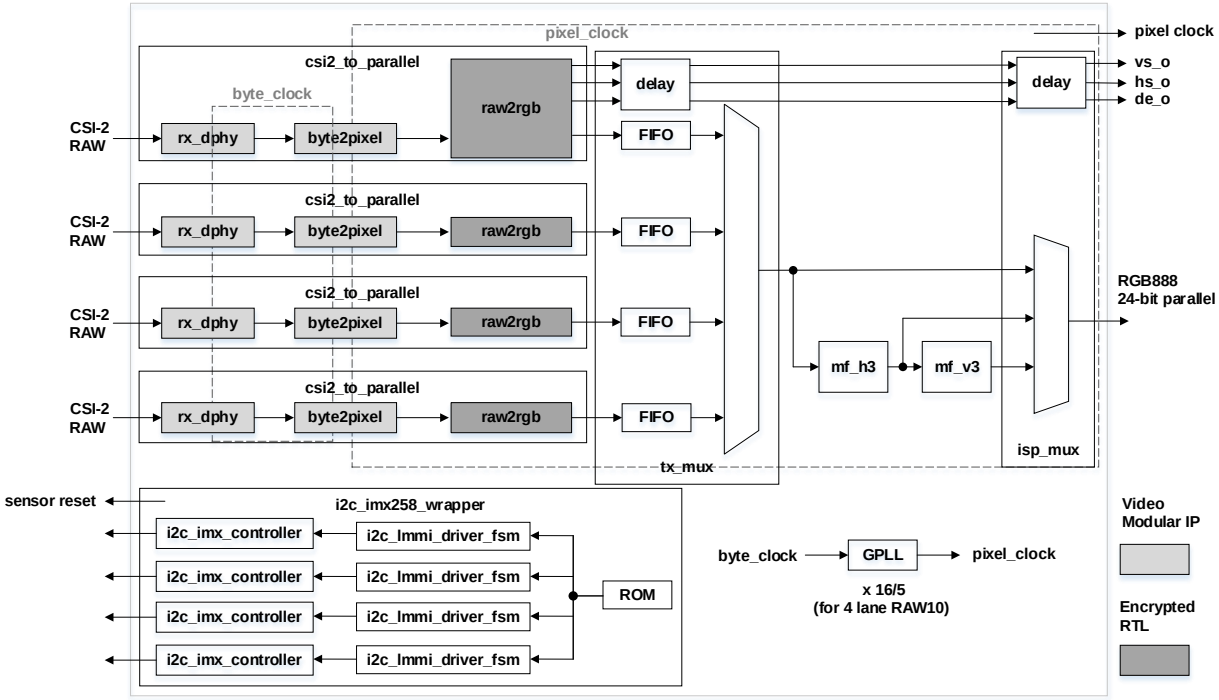

Figure 1.1. Block Diagram of 4 to 1 Image Aggregation with CrossLink-NX Devices (RGB Output)

Figure 1.2 shows the block level diagram of the 4 to 1 Image Aggregation with CrossLink-NX devices reference design of RAW output version.

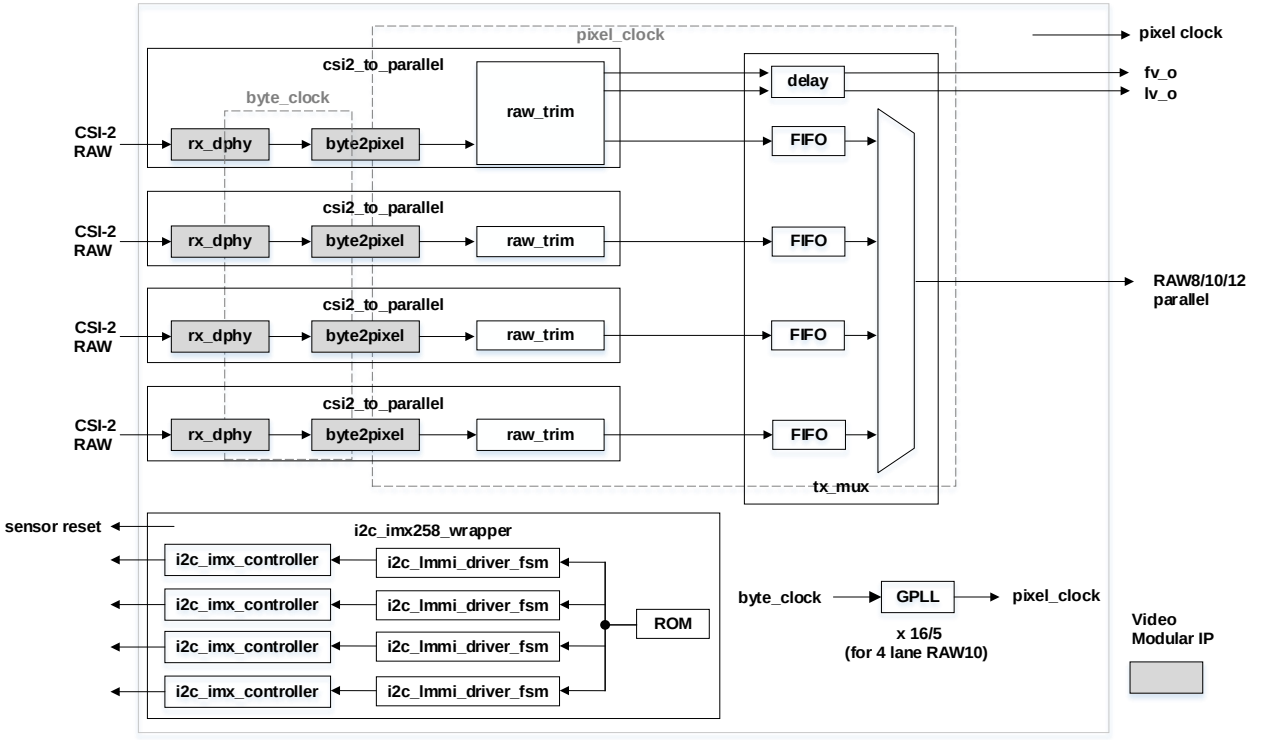

Figure 1.2. Block Diagram of 4 to 1 Image Aggregation with CrossLink-NX Devices (RAW Output)

Figure 1.3 shows a clocking scheme in continuous clock mode. In this example, Hard D-PHY IP is used on channel #0 and channel #1, and Soft D-PHY IP is used on channel #2 and channel #3. Byte clock (clk_byte_fr) is taken from one of the RX channels and used in all byte2pixel modules of all channels. Pixel clock is generated by GPLL. In this case, sync_clk is generated by another GPLL from the external reference clock and fed to Soft D-PHY RX IP. The external reference clock is also sent to all image sensors through CrossLink-NX for synchronous operation.

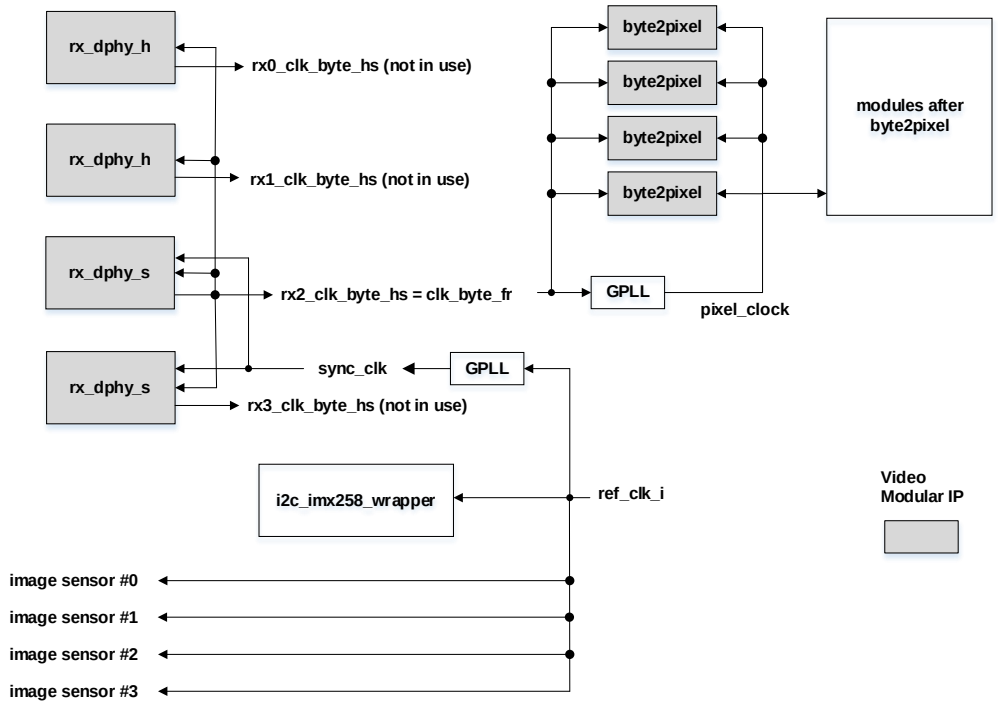

Figure 1.3. Sample Clocking Scheme in Continuous Clock Mode

Figure 1.4 shows a clocking scheme in non-continuous clock mode. Continuous byte clock is not available and using GPLL to create it along with pixel clock in this example. Also, a clock to drive the logic to judge High Speed (HS) and Low Power (LP) mode on CSI-2 clock lane (clk_lp_ctrl) is necessary in this mode. sync_clk and clk_lp_ctrl can be the same clock and it is generated by another GPLL from the external reference clock and fed to D-PHY RX IP. The external reference clock is also sent to all image sensors through CrossLink-NX devices for synchronous operation. Two GPLL are used in this example, but using only one GPLL could be possible if all necessary clocks can be generated by a single GPLL.

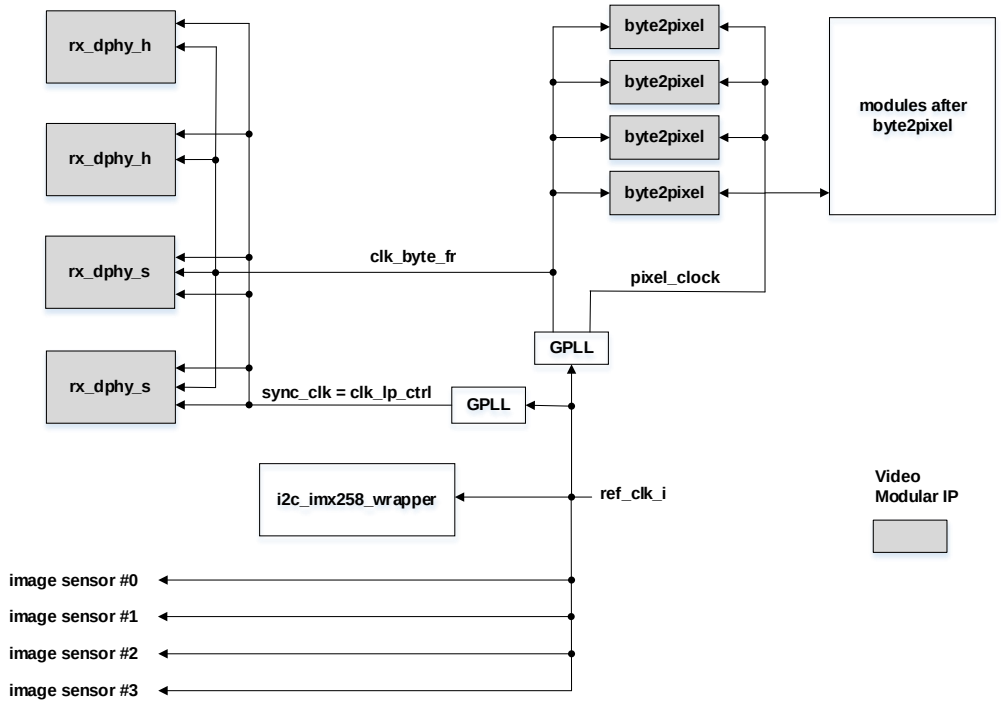

Figure 1.4. Sample Clocking Scheme in Non-Continuous Clock Mode

1.3. RX Permutations

Table 1.1 shows the available permutations of RX configurations. Since D-PHY Soft IP support only Gear 8 and at least two out of four RX D-PHY must be Soft D-PHY, we cannot use Gear 16 in this design. In addition, two pixel output option is available by Byte-to-Pixel IP, but not used in this design.

Table 1.1. RX Permutation

Data Type	Number of RX Lanes	RX Gear	Number of Pixels/ Pixel Clock
RAW8, RAW10, or RAW12	1	8	1
	2	8	1
	4	8	1

The Microsoft Excel sheet (aggr_4to1_clock.xlsx) included in the reference design package is for you to get the byte clock, the pixel clock from RX bandwidth, and other information. This sheet is useful in configuring IPs. A sample entry is shown in Figure 1.5. By entering MIPI bandwidth, Data Type, and number of RX lanes, Byte clock and Pixel clock are automatically calculated. The results can be used to configure D-PHY RX IP, Byte-to-Pixel IP, and GPLL. RX Gear (= 8), Number of TX channels (= 1), and TX Gear (= 7) are configurable only for reference purposes and should not be changed for this reference design.

	A	B	C	D
1	4 to 1 Image Aggregation with Crosslink-NX RD Frequency Calculator			
2	Fill in the cells with borders around them			
3				
4		RX Data Type	RAW10	
5		Byte2Pixel Output Width	10	
6		RX Line Rate	371.25	Mbps
7		RX DPHY Clock Frequency	185.625	MHz
8		Number of RX Lanes	4	
9		RX Gear	8	
10		Byte Clock Frequency	46.40625	MHz
11		Pixel Clock Frequency	148.5	MHz
12		RX Hard DPHY Data Settle Cycle	Default (unchecked)	
13		RX Soft DPHY Data Settle Cycle	2 - 4	Byte Clock Cycle
14		TX Data Type	TX_RGB888	
15		Number of TX Channels	1	
16		TX Gear	7	

Figure 1.5. Clock Frequency Calculator

2. Parameters and Port List

There are three directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Radiant software.
- synthesis_directives_sim.v – used for 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 and Byte-to-Pixel IP settings created by 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
RX Data Type	RX_RAW8	Define the data type on the RX channel. Only one of these three directives must be defined.
	RX_RAW10	
	RX_RAW12	
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 type[1]	RX0_DPHY_HARD	RX D-PHY type on channel #0. Only one of these two directives must be defined.
	RX0_DPHY_SOFT	
	RX1_DPHY_HARD	RX D-PHY type on channel #1. Only one of these two directives must be defined.
	RX1_DPHY_SOFT	
	RX2_DPHY_HARD	RX D-PHY type on channel #2. Only one of these two directives must be defined.
	RX2_DPHY_SOFT	
	RX3_DPHY_HARD	RX D-PHY type on channel #3. Only one of these two directives must be defined.
	RX3_DPHY_SOFT	
RX D-PHY Clock Mode[2]	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. 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	
TX Data Type	TX_RGB888	Define the data type on TX channel. Only one of these two must be defined.
	TX_RAW	
Bayer Pattern[3]	BAYER_PATTERN {value}	Define the Bayer pattern. Value must be 2'b00 – 2'b11.
Horizontal Front Porch[4]	HFP {value}	Define the horizontal front porch. Value must be 8'd1 – 8'd255.
HSYNC Pulse Length[4]	HS_LENGTH {value}	Define HSYNC pulse length. Value must be 8'd1 – 8'd255.
Vertical Front Porch[5]	VFP {value}	Define the vertical front porch. Value must be 6'd1 – 6'd63.
VSYNC Pulse Length[5]	VS_LENGTH {value}	Define VSYNC pulse length. Value must be 6'd1 – 6'd63. Applicable only to CSI-2 input.
Left Pixel Trimming[6]	LEFT_TRIM {value}	Define the number of pixels to be trimmed before TX. Value must be 6'd0 – 6'd63.
Horizontal Active Pixels on TX[6]	H_TX_PEL {value}	Define the number of active pixels to send on TX. Value must be 12'd10 – 12'd4095. The value must be even in case of TX_GEAR_14 or NUM_TX_CH_2.
Top Line Trimming[7]	TOP_TRIM {value}	Define the number of lines to be trimmed before TX. Value must be 6'd0 – 6'd63.

Category	Directive	Remarks
Vertical Active Lines on TX[7]	V_TX_LINE {value}	Define the number of active lines to be sent on TX. Value must be 12'd1 – 12'd4095.
Total Active Lines on RX	V_TOTAL_LINE {value}	Define the number of active lines exists on RX. Value must be 12'd1 – 12'd4095.

Notes:

1. Total number of Hard D-PHY IP cannot exceed two.
2. 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.
3. Refer to the RGB Data Creation section for pattern details.
4. (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.
5. (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.
6. (LEFT_TRIM + H_TX_PEL) cannot exceed the horizontal active pixel count of the incoming RX data. It is your responsibility to manage this.
7. (TOP_TRIM + V_TX_LINE) cannot exceed the total active line count of the incoming RX data (V_TOTAL_LINE). It is your 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 small counter values to avoid the long simulation time (such as counters for initial reset).
Reference clock period	REF_CLK {value}	External 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 pixels	NUM_PIXELS {value}	Number of active video pixels per line.
Line Gap period	DPHY_LPS_GAP {value}	Line Gap time in ps. Applicable to CSI-2 RX. This is LP period between two active lines.
Frame Gap period	FRAME_LPM_DELAY {value}	Frame Gap time in ps. This is LP period between Frame End and Frame Start.
Wait for I2C data transfer completion	WAIT_I2C_COMPLETE	Define this directive to wait for I2C data transfer from ROM to the targets to be completed. Testbench compares the data transfer duration with a static value based on the ROM IMX258 content.

2.3. Top-level I/O

Table 2.3 shows the top-level I/O of this reference design. Actual I/O depends on your lane configurations. All necessary I/O ports are automatically declared by referring synthesis_directives.v.

Table 2.3. 4 to 1 Image Aggregation with CrossLink-NX Device Top-level I/O

Port Name	Direction	Description
Clocks and Resets		
ref_clk_i	I	Input reference clock. Used to drive I^2C controller modules. Also this clock is provided to image sensors. Typical frequency is 27 MHz.
reset_n_i	I	Asynchronous active low system reset #1
gsr_n_i	I	Asynchronous active low system reset #2
sensor_reset_n_o	O	Asynchronous active low reset to image sensors
Control Interface		
ctrl_i	I	Output image control #1 to change aggregated image patterns
isp_ctrl_i	I	Output image control #2 to select the original or filtered images. Used in case of RGB output.
CSI-2 Sensor Interface		
rx0_mclk_o	O	Controller clock to image sensor on channel #0
rx0_scl	I/O	I^2C clock to image sensor on channel #0
rx0_sda	I/O	I^2C data to/from image sensor on channel #0
rx0_clk_p_i	I	Positive differential RX D-PHY input clock on channel #0
rx0_clk_n_i	I	Negative differential RX D-PHY input clock on channel #0
rx0_d0_p_i	I	Positive differential RX D-PHY input data 0 on channel #0
rx0_d0_n_i	I	Negative differential RX D-PHY input data 0 on channel #0
rx0_d1_p_i	I	Positive differential RX D-PHY input data 1 on channel #0 (in case of 2-lane or 4-lane configuration)
rx0_d1_n_i	I	Negative differential RX D-PHY input data 1 on channel #0 (in case of 2-lane or 4-lane configuration)
rx0_d2_p_i	I	Positive differential RX D-PHY input data 2 on channel #0 (in case of 4-lane configuration)
rx0_d2_n_i	I	Negative differential RX D-PHY input data 2 on channel #0 (in case of 4-lane configuration)
rx0_d3_p_i	I	Positive differential RX D-PHY input data 3 on channel #0 (in case of 4-lane configuration)
rx0_d3_n_i	I	Negative differential RX D-PHY input data 3 on channel #0 (in case of 4-lane configuration)
rx1_mclk_o	O	Controller clock to image sensor on channel #1
rx1_scl	I/O	I^2C clock to image sensor on channel #1
rx1_sda	I/O	I^2C data to/from image sensor on channel #1
rx1_clk_p_i	I	Positive differential RX D-PHY input clock on channel #1
rx1_clk_n_i	I	Negative differential RX D-PHY input clock on channel #1
rx1_d0_p_i	I	Positive differential RX D-PHY input data 0 on channel #1
rx1_d0_n_i	I	Negative differential RX D-PHY input data 0 on channel #1
rx1_d1_p_i	I	Positive differential RX D-PHY input data 1 on channel #1 (in case of 2-lane or 4-lane configuration)
rx1_d1_n_i	I	Negative differential RX D-PHY input data 1 on channel #1 (in case of 2-lane or 4-lane configuration)
rx1_d2_p_i	I	Positive differential RX D-PHY input data 2 on channel #1 (in case of 4-lane configuration)
rx1_d2_n_i	I	Negative differential RX D-PHY input data 2 on channel #1 (in case of 4-lane configuration)
rx1_d3_p_i	I	Positive differential RX D-PHY input data 3 on channel #1 (in case of 4-lane configuration)
rx1_d3_n_i	I	Negative differential RX D-PHY input data 3 on channel #1 (in case of 4-lane configuration)
rx2_mclk_o	O	Controller clock to image sensor on channel #2
rx2_scl	I/O	I^2C clock to image sensor on channel #2
rx2_sda	I/O	I^2C data to/from image sensor on channel #2

Port Name	Direction	Description
rx2_clk_p_i	I	Positive differential RX D-PHY input clock on channel #2
rx2_clk_n_i	I	Negative differential RX D-PHY input clock on channel #2
rx2_d0_p_i	I	Positive differential RX D-PHY input data 0 on channel #2
rx2_d0_n_i	I	Negative differential RX D-PHY input data 0 on channel #2
rx2_d1_p_i	I	Positive differential RX D-PHY input data 1 on channel #2 (in case of 2-lane or 4-lane configuration)
rx2_d1_n_i	I	Negative differential RX D-PHY input data 1 on channel #2 (in case of 2-lane or 4-lane configuration)
rx2_d2_p_i	I	Positive differential RX D-PHY input data 2 on channel #2 (in case of 4-lane configuration)
rx2_d2_n_i	I	Negative differential RX D-PHY input data 2 on channel #2 (in case of 4-lane configuration)
rx2_d3_p_i	I	Positive differential RX D-PHY input data 3 on channel #2 (in case of 4-lane configuration)
rx2_d3_n_i	I	Negative differential RX D-PHY input data 3 on channel #2 (in case of 4-lane configuration)
rx3_mclk_o	O	Controller clock to image sensor on channel #3
rx3_scl	I/O	I^2C clock to image sensor on channel #3
rx3_sda	I/O	I^2C data to/from image sensor on channel #3
rx3_clk_p_i	I	Positive differential RX D-PHY input clock on channel #3
rx3_clk_n_i	I	Negative differential RX D-PHY input clock on channel #3
rx3_d0_p_i	I	Positive differential RX D-PHY input data 0 on channel #3
rx3_d0_n_i	I	Negative differential RX D-PHY input data 0 on channel #3
rx3_d1_p_i	I	Positive differential RX D-PHY input data 1 on channel #3 (in case of 2-lane or 4-lane configuration)
rx3_d1_n_i	I	Negative differential RX D-PHY input data 1 on channel #3 (in case of 2-lane or 4-lane configuration)
rx3_d2_p_i	I	Positive differential RX D-PHY input data 2 on channel #3 (in case of 4-lane configuration)
rx3_d2_n_i	I	Negative differential RX D-PHY input data 2 on channel #3 (in case of 4-lane configuration)
rx3_d3_p_i	I	Positive differential RX D-PHY input data 3 on channel #3 (in case of 4-lane configuration)
rx3_d3_n_i	I	Negative differential RX D-PHY input data 3 on channel #3 (in case of 4-lane configuration)
Parallel TX Interface		
pix_clk_o	O	TX pixel clock
vs_o	O	Vertical Sync (active high), used in case of RGB output
hs_o	O	Horizontal Sync (active high), used in case of RGB output
de_o	O	Data Enable (active high), used in case of RGB output
rd_o[7:0]	O	Red Data, used in case of RGB output
gd_o[7:0]	O	Green Data, used in case of RGB output
bd_o[7:0]	O	Blue Data, used in case of RGB output
fv_o	O	Frame Valid (active high), used in case of RAW output
lv_o	O	Line Valid (active high), used in case of RAW output
rawd_o[n:0]	O	RAW Data, used in case of RAW output. n = 7 (RX_RAW8), 9 (RX_RAW10), or 11 (RX_RAW12).

3. Design and Module Description

The top-level design (csi2_4to1_NX.v) consists of the following modules:

- rst_ctrl
- i2c_imx258_wrapper
  - i2c_imx258_controller
  - i2c_lmmi_driver_fsm
  - imx258_rom_lmmi_data
- csi2_to_parallel
  - rx_dphy_h
  - rx_dphy_s
  - byte2pixel
  - data_mask
  - raw_trim
  - raw2rgb
- tx_mux
- ch_ctrl
- mf_h3
- mf_v3
- isp_mux
- sync_clk_gpll
- int_gpll

The top-level design has a reset synchronization logic.

3.1. rst_ctrl

This module takes external reset (reset_n_i & gsr_n_i) and generate three reset signals using the counter:

- sensor_rst_n_o : reset to four image sensor devices
- i2c_m_rst_n_o : reset to I2C controller wrapper modules
- nx_rst_n_o : reset to other modules

3.2. i2c_imx258_wrapper

This module contains four sets of I2C controllers (i2c_imx258_controller) and finite state machines (FSMs) to drive the Lattice Memory Mapped Interface (LMMI) buses for the controller (i2c_lmmi_driver_fsm) modules to configure four image sensors simultaneously.

3.2.1. i2c_imx258_controller

This module is an IP of i2c_controller. Figure 3.1 shows an example of IP interface and parameter settings in the Lattice Radiant software for the I2C Controller IP. You can use the .ipx file (i2c_imx258_controller/ i2c_imx258_controller.ipx) in the sample project to reconfigure your design. Refer to the I^{2}C Controller IP Core – Lattice Radiant Software User Guide (FPGA-IPUG-02071) for details.

Guidelines and parameter settings required for this reference design are provided below:

- APB Mode Enable – Uncheck (use LMMI bus instead).
- Remove Tristate Buffers – Uncheck.
- FIFO Depth – Select the size accordingly depending on the size of one I2C transaction. For IMX258 in this example, 16 is sufficient.
- Implementation of FIFO – Set to LUT.
- TX FIFO Almost Empty Flag [1 – 256] – Set to 1.

- RX FIFO Almost Full Flag [1 – 256] – Set to 15.
- System Clock Frequency (MHz) [10 – 200] – Set to the frequency of clk_i. In the design example, set to 27.
- Desired SCL Frequency (kHz) [100 – 1000] – Select the frequency supported by the targeted I2C devices. In this example, set to 400.

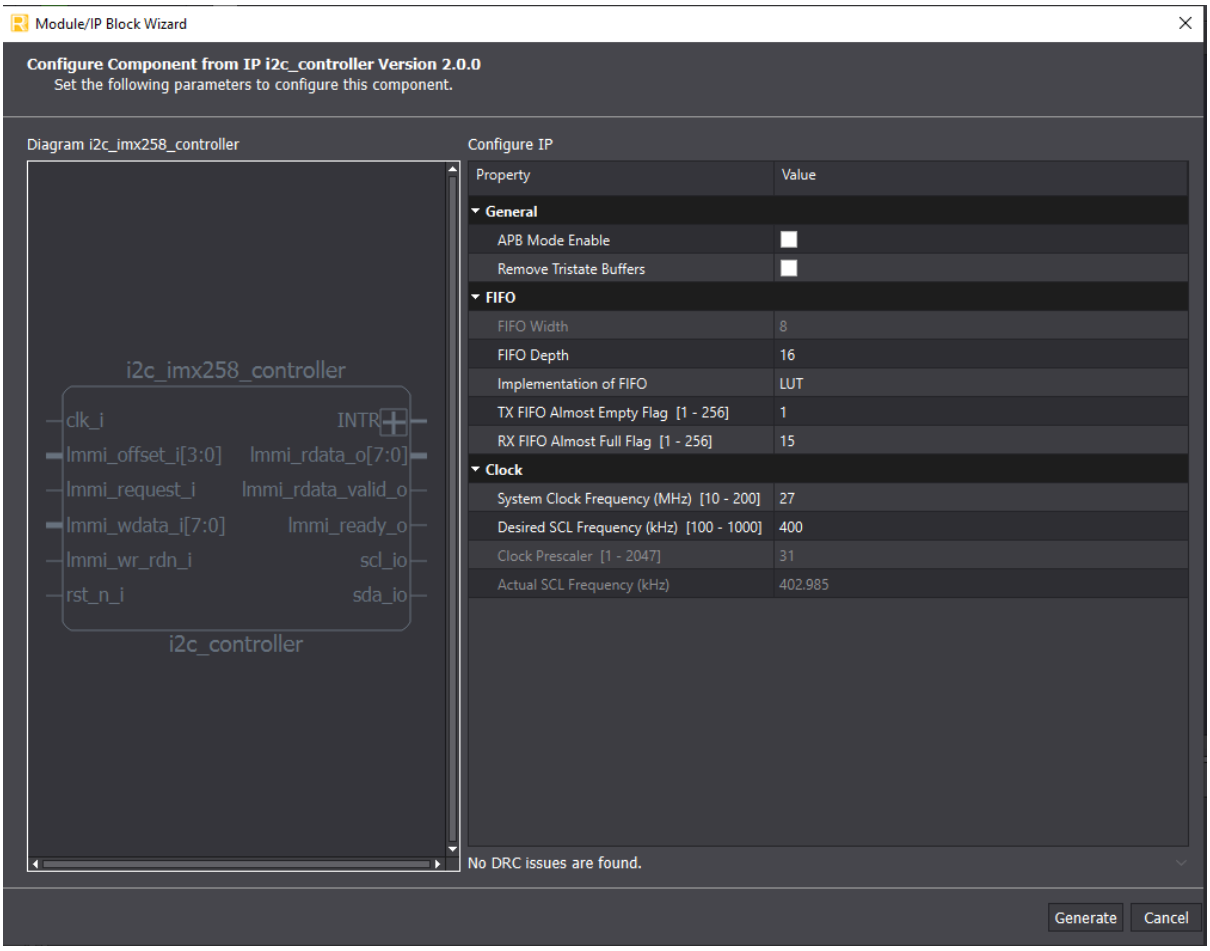

Figure 3.1. Example of i2c_imx258_controller IP Configuration

3.2.2. i2c_lmmi_driver_fsm

Drive the inputs of the I2C Controller IP either via the Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) or LMMI bus. This module implements an LMMI driver in the form of an FSM to drive the bus to the I2C Controller module. The data, offset, and commands are compiled into an ROM image in the imx258_rom_lmmi_data ROM module. The FSM starts around 1 ms after the reset is being asserted on the wrapper module.

Note: The wrapper module reset wait time is shortened significantly in the simulation.

Note: The current FSM implementation only supports write operations and interrupt-based read operations. Read operations are only supported for reading interrupt registers when the int_o pin of the i2c_imx258_controller module is triggered. This read operation is mainly used to wait for the tr_cmp_int bit to be asserted before issuing the next transaction, and to check if any other errors are triggered, such as nack_error_int.

3.2.3. imx258_rom_lmmi_data

This is the ROM module containing the data to be programmed into LMMI bus for the I2C Controller IP. The ROM image must be 16-bit wide with the following settings:

- Bit[7:0] – 8 bits data to be programmed on the lmmi_wdata_i pin of the I2C Controller IP.

- Bit[11:8] – 4 bits LMMI offset register address to be programmed on the lmmi_offset_i pin of the I2C Controller IP.
- Bit[12] – 1 bit to specify LMMI Read (1'b0) or Write (1'b1) mode.
- Bit[14:13] – Read Mode. Only WAIT_INT mode (2'b10) is supported.
- Bit[15] – Reserved.

Refer to the "Programming Flow" section of the I²C Controller IP Core – Lattice Radiant Software User Guide (FPGA-IPUG-02071) to arrange the sequence of the ROM content as recommended. You can locate the sample ROM image file for IMX258 in *source/imx258_rom_lmmi_data.mem*. The sample .mem file is annotated with comments (line begins with "//") for readability, and the data is in hexadecimal format.

3.3. csi2_to_parallel

This module contains main functional modules for a single channel operation.

3.3.1. rx_dphy_h

This module is an IP of Hard D-PHY RX. Figure 3.2 shows an example of IP interface and parameter settings in Lattice Radiant software for the CSI-2/DSI D-PHY Receiver Submodule IP. You can use the .ipx file (*rx_dphy_h/rx_dphy_h.ipx*) included in the sample project and reconfigure according to your needs. Refer to CSI/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081) for details.

Guidelines and parameter settings required for this reference design are provided below:

- RX Interface – Select CSI-2.
- D-PHY RX IP – Select Hard DPHY.
- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Select 8.
- CIL Bypass – Select Enabled.
- Enable LMMI Interface – Select Disabled (unchecked).
- Enable AXI4-Stream Interface – Select Disabled (unchecked).
- RX Line Rate – Set according to channel configuration. The value above 1500 cannot be used with Gear 8.
- D-PHY Clock Mode – Select Continuous or Non-continuous. Must match RX*_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- Customize Data Settle Cycle – Set to disable (unchecked).
- Enable Packet Parser – Select Enabled.
- Enable Miscellaneous Status Signals – Select Disabled (unchecked).
- RX FIFO Implementation – Select EBR or LUT.
- RX FIFO Depth – Select 16.
- RX FIFO Type – Select SINGLE.
- RX FIFO Packet Delay – Set to minimum value (2).
- RX FIFO Clock Mode – Select DC (dual clock).
- Misc Signals – Select Disabled (unchecked).

This module takes serial CSI-2 data and outputs byte data after de-serialization in CSI-2 High Speed mode and protocol decoding. If you create this IP from scratch, it is recommended to set the design name to *rx_dphy_h*. In this way, 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 is not necessary if you use only Soft D-PHY IP.

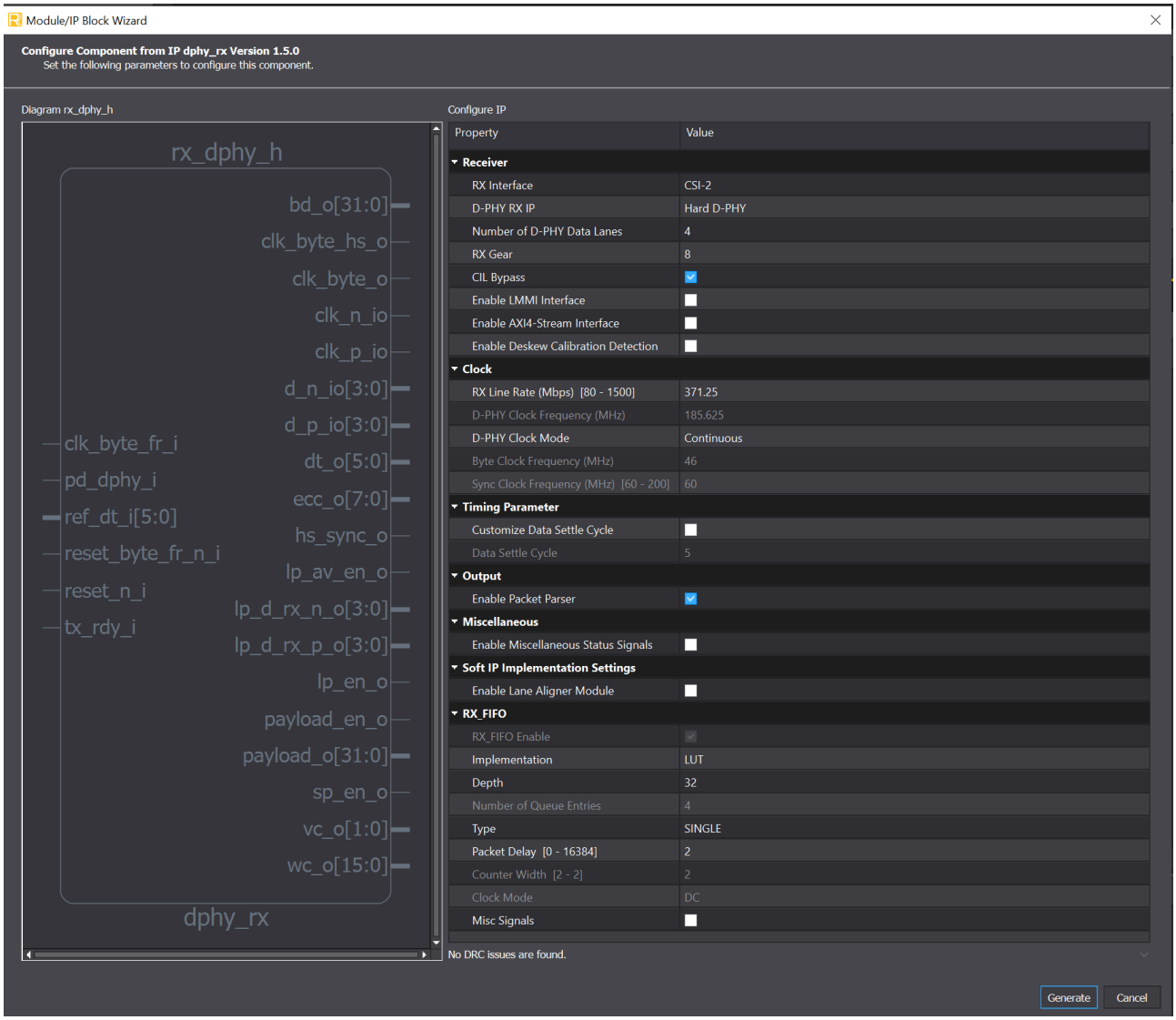

Figure 3.2. rx_dphy_h IP Creation by the Lattice Radiant Software

3.3.1.1. RX FIFO in Hard D-PHY RX IP

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

Continuous Clock Mode

In this case, the minimum configuration of RX FIFO is recommended as mentioned above (LUT based, Depth = 16, type = SINGLE, Packet Delay = Minimum (2), Clock Mode = DC).

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 = Minimum (2), 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 of all, it's 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% comparing 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 no 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.

3.3.2. rx_dphy_s

This module is an IP of Soft D-PHY RX. Figure 3.3 shows an example of IP interface and parameter settings in Lattice Radiant software for the CSI-2/DSI D-PHY Receiver Submodule IP. You can use the .ipx file (*rx_dphy_s/rx_dphy_s.ipx*) included in the sample project and reconfigure according to your needs. Refer to CSI/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081) for details.

Guidelines and parameter settings required for this reference design are provided below:

- RX Interface – Select CSI-2.
- D-PHY RX IP – Select Soft DPHY.
- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Fixed to 8.
- Enable AXI4-Stream Interface – Select Disabled (unchecked).
- RX Line Rate – Set according to channel configuration.
- 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 – Enter the appropriate Value.
- Customize Data Settle Cycle – Set to Enable (checked).
- Data Settle Cycle – Use values within the range from the Frequency Calculator spreadsheet.
- Enable Packet Parser – Select Enabled.
- Enable Miscellaneous Status Signals – Select Disabled (unchecked).
- Enable Lane Aligner Module – Select Enabled or Disabled according to the preference.
- Lane Aligner FIFO Depth – Select 8 when enabled.
- FIFO_TYPE – Select LUT when enabled.
- RX_FIFO Enable – Select Enabled.
- Implementation – Select EBR or LUT.
- Depth – Select 16.
- Type – Select SINGLE.
- Packet Delay – Set to minimum (2).
- Clock Mode – Select DC (dual clock).
- Misc Signals – Select Disabled (unchecked).

This module takes serial CSI-2 data and outputs byte data after de-serialization in CSI-2 High Speed mode and protocol decoding. If you create this IP from scratch, it is recommended to set the design name to *rx_dphy_s*. In this way, 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.

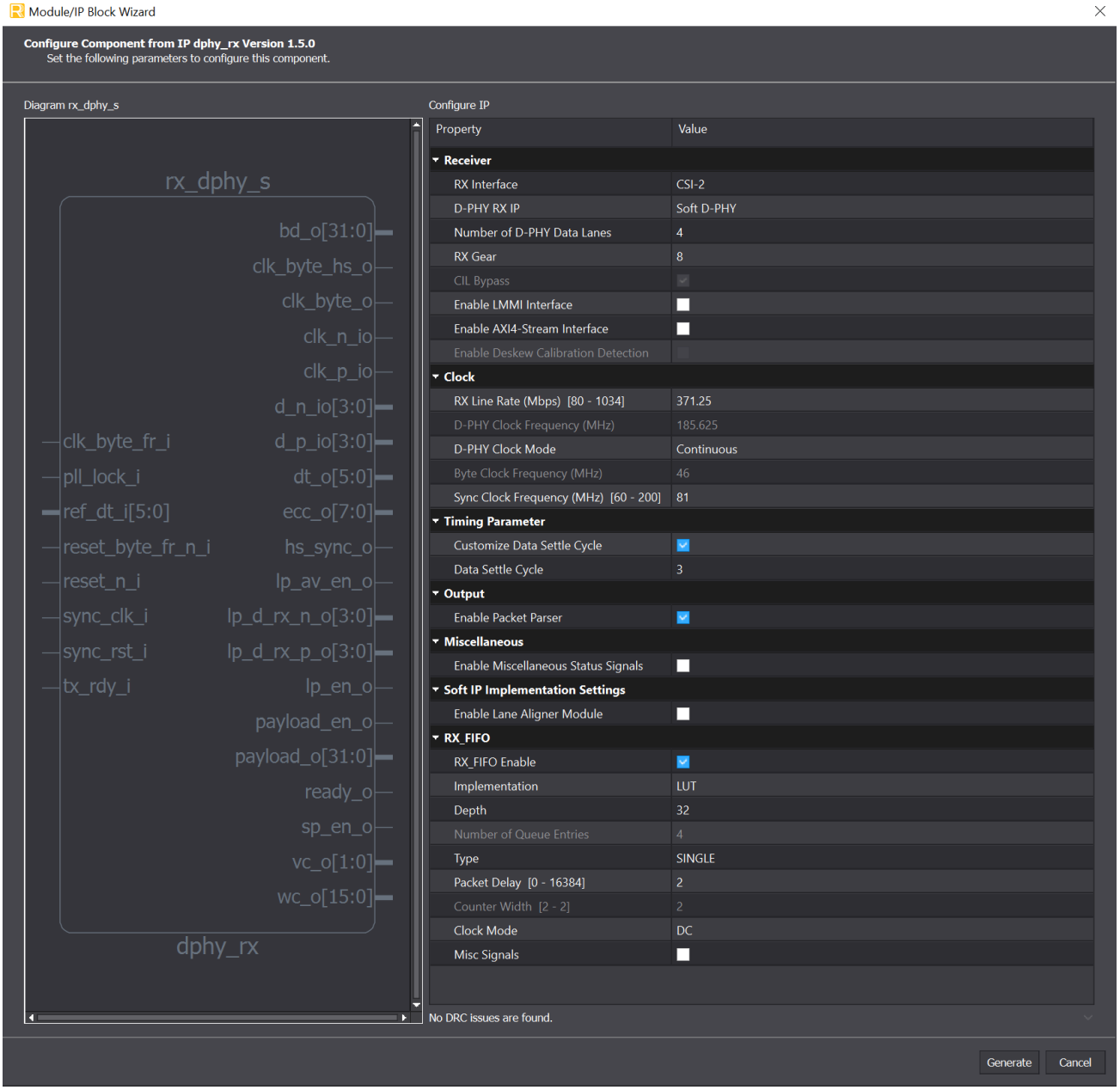

Figure 3.3. rx_dphy_s IP Creation by the Lattice Radiant Software

3.3.2.1. RX FIFO in Soft D-PHY RX IP

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

Continuous Clock Mode

In this case, RX FIFO is not necessary and RX_FIFO Enable should be disabled. The sample design still enables RX FIFO to have the similar timing with the RX channels which uses rx_dphy_h (Hard D-PHY RX IP).

Non-Continuous Clock Mode

Refer to the Non-Continuous Clock Mode section in RX FIFO in Hard D-PHY RX IP.

3.3.3. byte2pixel

This module must be created for the RX channel according to data type, the number of lanes, RX Gear, number of lanes, byte clock frequency, pixel clock frequency, data type, and word count. Figure 3.4 shows an example of IP interface settings in Lattice Radiant software for the byte2pixel IP. You can use the .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 Radiant Software User Guide (FPGA-IPUG-02079) for details.

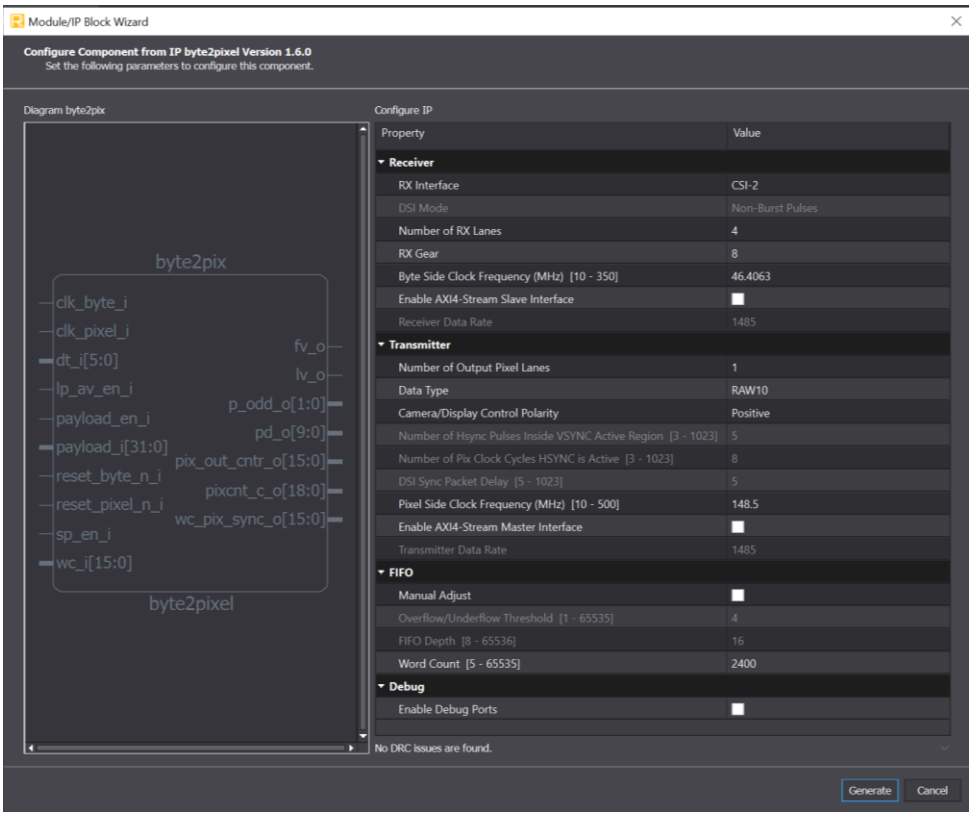

Figure 3.4. byte2pixel IP Creation in the Lattice Radiant Software

Guidelines and parameter settings required for this reference design are provided below:

- RX Interface – Select CSI-2.
- Number of RX Lanes – Set the same value as RX D-PHY IP.
- RX Gear – Select 8.
- Number of Output Pixels – Select 1.
- Byte Side Clock Frequency – Set the same value you get in RX D-PHY IP.
- Data Type – Select RAW8, RAW10, or RAW12 according to the data type on CSI-2.
- Camera/Display Control Polarity – Select Positive.
- Pixel Side Clock Frequency – Enter the appropriate value.
- Enable AXI4-Stream Interface – Select Disabled (unchecked).
- Manual Adjust – Select Disabled (unchecked).
- Word Count – Enter the appropriate value.
  This value is obtained from a following equation:
  $Word\ Count = \frac{horizontal\ resolution * raw\ data\ width}{8}$.
  Ex. in case of 1080P with RAW10, the value will be 1920 x 10/8 = 2400.
- Enable Debug Ports – Select Disabled (unchecked).

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 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. In general, pixel clock is provided by GPLL.

Figure 3.5 shows the interface timing diagram. Frame Valid (FV) and Line Valid (LV) are generated based on short packet enable and payload enable signals. Figure 3.6 shows global output timing. In case of RGB output, sync signals (VSYNC, HSYNC) have to be created from FV and LV to match the interface format of parallel RGB. In addition, RGB pixel data have to be created from RAW data. For these purposes, an additional module (raw2rgb) is required after byte2pixel for RGB output.

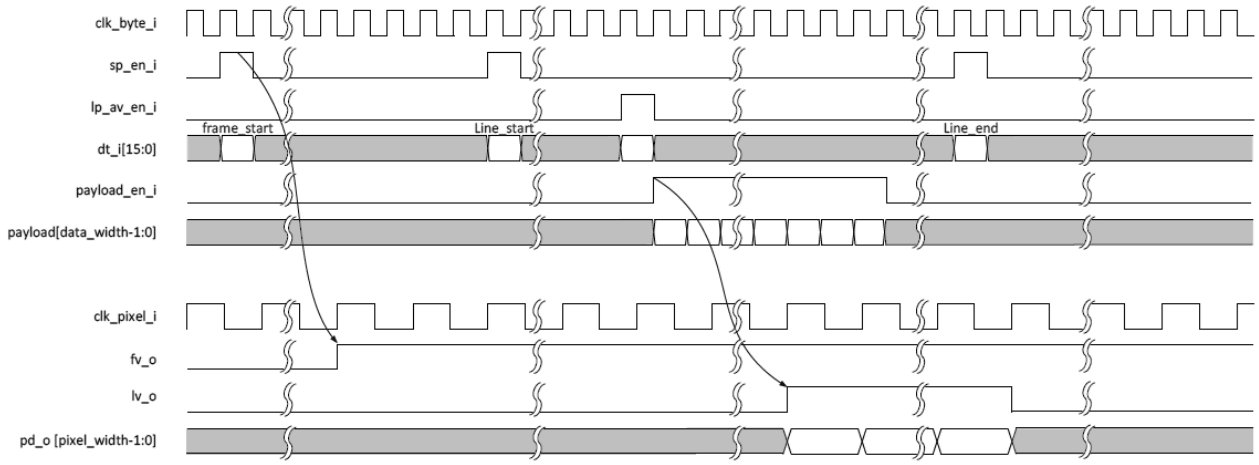

Figure 3.5. Interface Timing Diagram

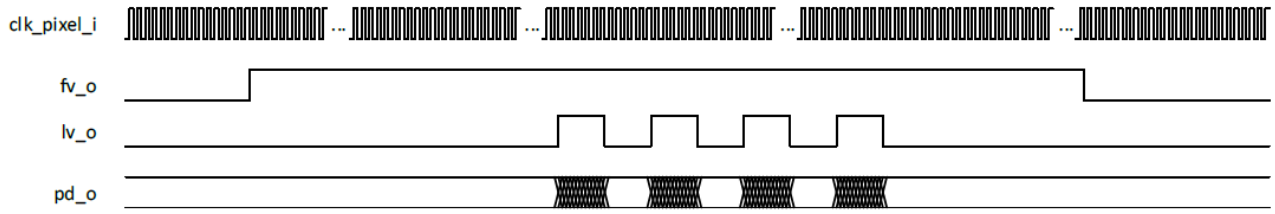

Figure 3.6. Global Output Timing Diagram

3.3.4. data_mask

This module masks enable signals come out from byte2pixel to make sure the next module (raw2rgb or raw_trim) receives the data from the start of the frame. After the pixel clock is available, it waits for FV = 1 to begin passing through the enable and pixel data to raw2rgb/raw_trim. This module can be bypassed in case that it is known that the pixel clock (output of GPLL) becomes available before the image sensor sends out the first packet data after reset release.

3.3.5. raw_trim

In some cases, video data from sensor devices have an area that is larger than what is required by the display. This module takes care of horizontal and vertical trimming of RAW data comes from byte2pixel. This module is used only in case of RAW data output. Following parameters are used for edge trimming:

- 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 are defined in synthesis_directives.v. Note that (LEFT_TRIM + H_TX_PEL) must not exceed the original horizontal active pixel count, otherwise, output image is corrupted. Similarly, (TOP_TRIM + V_TX_LINE) must not exceed the original active line count for the same reason.

3.3.6. raw2rgb

This module has three major functions: RGB data creation, sync signal generation and video data trimming. This module is used only in case of RGB data output.

3.3.6.1. 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 data. Figure 3.7 shows four scenarios of Bayer patterns of the color components at the top-left pixels come from the image sensor.

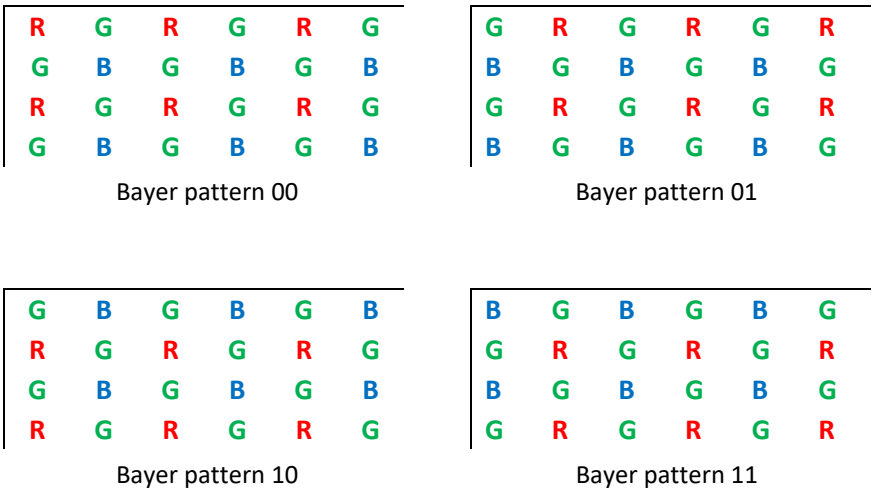

Figure 3.7. Bayer Pattern of RAW Data

Missing color components are created by interpolation using the neighborhood pixels. Figure 3.8 shows the 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.

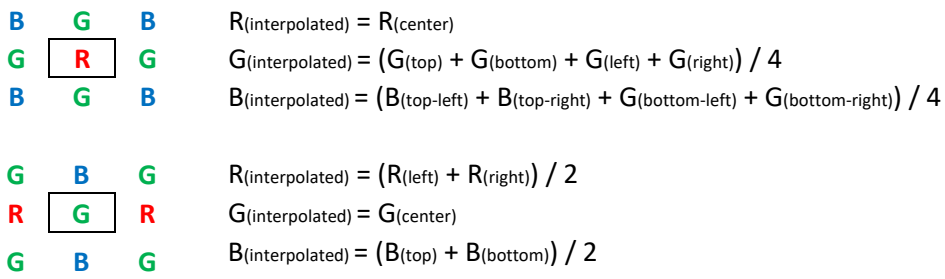

Figure 3.8. RGB Data Creation from RAW Data

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 is 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.

3.3.6.2. Sync Signal Generation

Since sync signal format is different between CSI-2 and parallel RGB, VSYNC and HSYNC signals have to be created from FV and LV comes from byte2pixel. 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 are defined in synthesis_directives.v. Note that (HFP + HS_LENGTH) must be shorter than the horizontal blanking period. Otherwise, HSYNC active period and Design Enable (DE) active period overlap and we cannot expect the anticipated image on the display. If you have no idea about the length of the horizontal blanking period, small values should be set. For example, HFP = 2, HS_LENGTH = 2, and others, as long as the display accepts. 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.

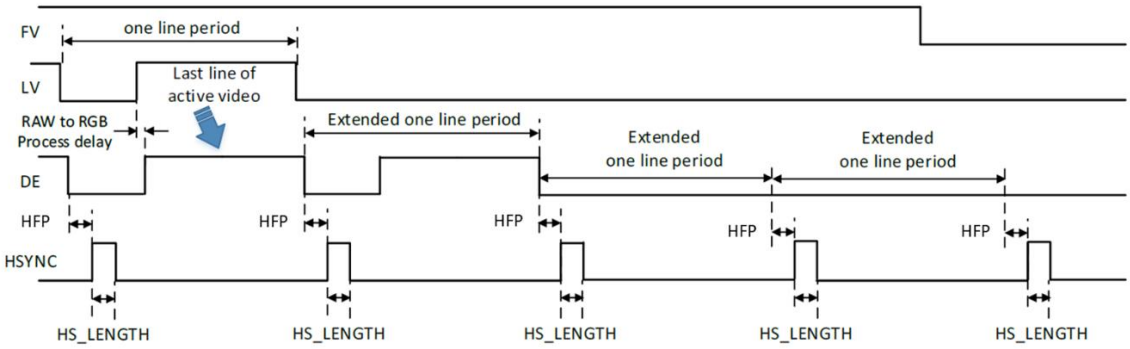

Figure 3.9. HSYNC Generation

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 3.9. 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, HSYNC generated utilizing the extended one line period is masked by the first LV in a new frame and HSYNC assertion timing is re-aligned based on LV timing of the new frame. Figure 3.10 shows the masking of HSYNC which overlaps LV period.

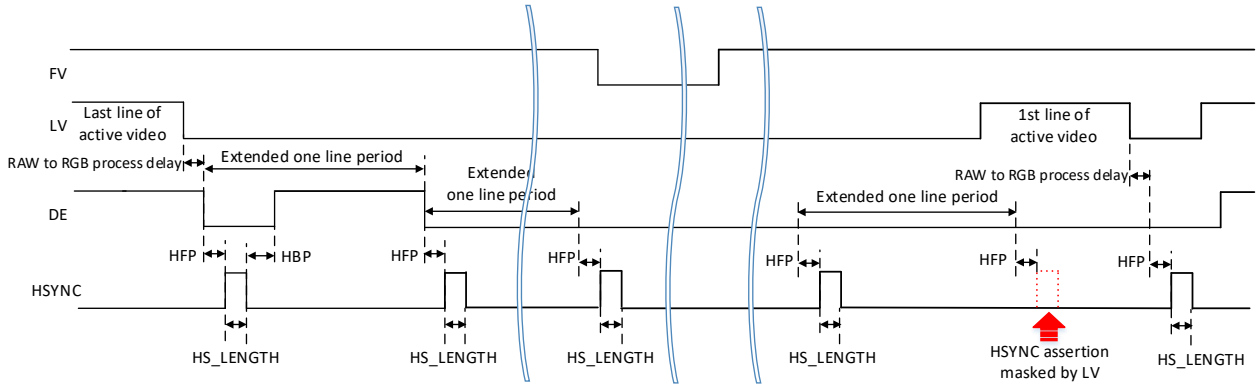

Figure 3.10. HSYNC Masking

3.3.6.3. Active Data Trimming

In case of RGB output, trimming is done after RGB data creation 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 are defined in synthesis_directives.v. 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. tx_mux

This module takes the parallel RGB888 data from four channels and select the specified channel data according to the info from ch_ctrl. Since there is no guarantee about the data timing among four channels, each channel data are stored into the FIFO and read back once all channel data are received. One frame image is divided into eight areas shown in Figure 3.11 and the selected channel is determined by the control data come from ch_ctrl.

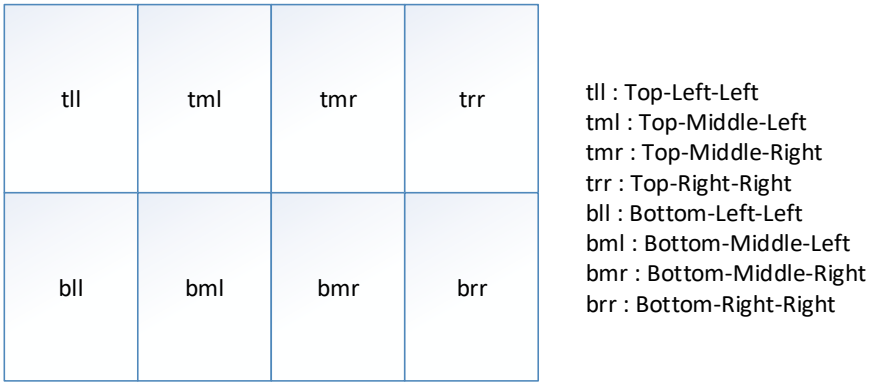

Figure 3.11. Image Segmentation

Note that the incoming image size from each channel and outgoing parallel data are same and no down scaling is done before the aggregation. This module picks up the selected channel data for each segment according to the control data from ch_ctrl. Two-bit control data is provided for each segment (tll_ch_i[1:0], tml_ch_i[1:0], …, brr_ch_i[1:0]; 00 : ch #0, 01 : ch #1, 10 : ch #2, 11 : ch #3). All four channel data are read out from the FIFOs at the same timing, then cropped and stitched to create a single image sent out from this module along with sync and enable signals.

3.5. ch_ctrl

This module controls the channel selection for each image segment. The selection pattern changes by every ctrl_i assertion (low active) assuming the push button operation. Currently, the pattern is coded as follows:

```
always @(*) begin
    casex (r_mux_cnt)
        3'b000 : begin w_tll_ch = 2'd0; w_tml_ch = 2'd0; w_tmr_ch = 2'd1; w_trr_ch = 2'd1;
w_bll_ch = 2'd2; w_bml_ch = 2'd2; w_bmr_ch = 2'd3; w_brr_ch = 2'd3; end
        3'b001 : begin w_tll_ch = 2'd0; w_tml_ch = 2'd0; w_tmr_ch = 2'd0; w_trr_ch = 2'd0;
w_bll_ch = 2'd0; w_bml_ch = 2'd0; w_bmr_ch = 2'd0; w_brr_ch = 2'd0; end
        3'b010 : begin w_tll_ch = 2'd1; w_tml_ch = 2'd1; w_tmr_ch = 2'd1; w_trr_ch = 2'd1;
w_bll_ch = 2'd1; w_bml_ch = 2'd1; w_bmr_ch = 2'd1; w_brr_ch = 2'd1; end
        3'b011 : begin w_tll_ch = 2'd2; w_tml_ch = 2'd2; w_tmr_ch = 2'd2; w_trr_ch = 2'd2;
w_bll_ch = 2'd2; w_bml_ch = 2'd2; w_bmr_ch = 2'd2; w_brr_ch = 2'd2; end
        3'b100 : begin w_tll_ch = 2'd3; w_tml_ch = 2'd3; w_tmr_ch = 2'd3; w_trr_ch = 2'd3;
w_bll_ch = 2'd3; w_bml_ch = 2'd3; w_bmr_ch = 2'd3; w_brr_ch = 2'd3; end
        3'b101 : begin w_tll_ch = 2'd0; w_tml_ch = 2'd1; w_tmr_ch = 2'd2; w_trr_ch = 2'd3;
w_bll_ch = 2'd0; w_bml_ch = 2'd1; w_bmr_ch = 2'd2; w_brr_ch = 2'd3; end
        default : begin w_tll_ch = 2'd0; w_tml_ch = 2'd0; w_tmr_ch = 2'd1; w_trr_ch = 2'd1;
w_bll_ch = 2'd2; w_bml_ch = 2'd2; w_bmr_ch = 2'd3; w_brr_ch = 2'd3; end
    endcase
end
```

The above code makes following pattern transition by every ctrl_i assertion:

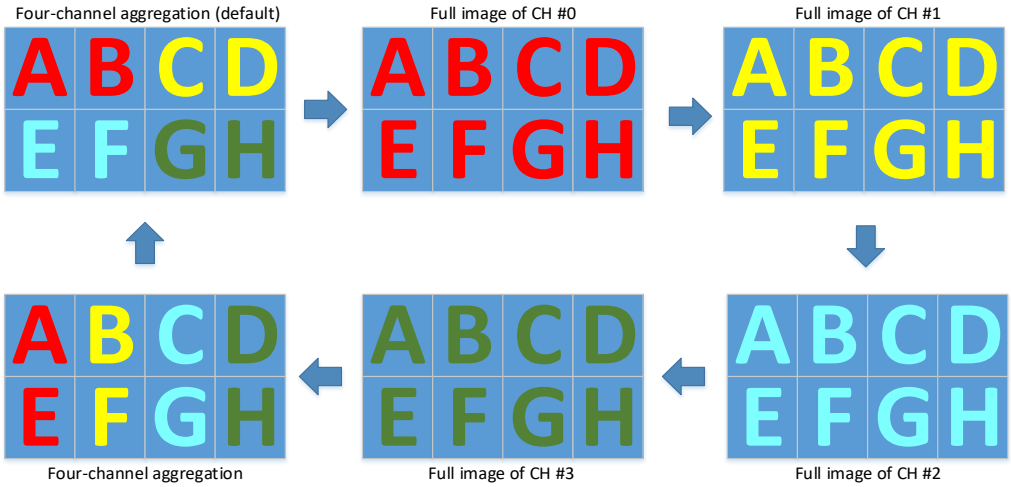

Figure 3.12. Display Image Patterns by Sample Design

You can make your own aggregation patterns by modifying this module.

3.6. mf_h3

This module applies horizontal 3-tap median filter for noise reduction. In case of left and right edge pixels, take the average between the own pixel and adjacent pixel since 3-tap processing is impossible. Use of this module is optional. This module only works for RGB data output.

3.7. mf_v3

This module applies vertical 3-tap median filter for noise reduction. In case of top and bottom edge lines, take the average between the own line and adjacent line since 3-tap processing is impossible. It requires FIFO that can contain two lines of data. Use of this module is optional. This module only works for RGB data output.

3.8. isp_mux

This module controls the final output image among following three:

- Original image from tx_mux
- Image after the filter by mf_h3
- Image after the filter by mf_h3 + mf_v3

By isp_ctrl_i (active low) assertion, the output image changes cyclically in the above order. The default is the original image. This module only works for RGB data output.

3.9. sync_clk_gpll

Soft D-PHY RX IP requires 60 MHz to 200 MHz clock (sync_clk) for its operation. This clock can be independent from other clocks. The sample design generates 81 MHz from 27 MHz reference clock for this purpose. You can use the ipx file (sync_clk_gpll/sync_clk_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 *sync_clk_gpll* 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.

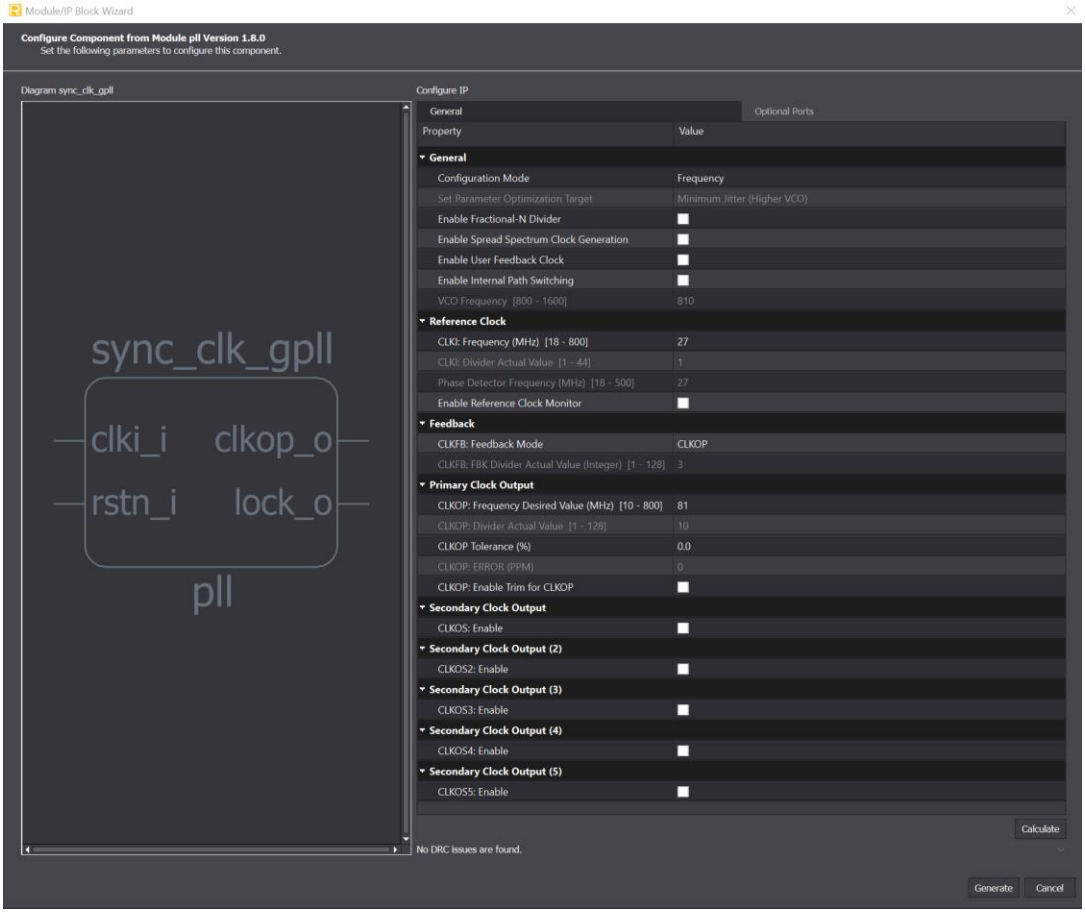

Figure 3.13. GPLL IP Creation for sync_clk

3.10. int_gpll

The frequency relationship between byte clock and pixel clock is as follows:

$$pixel\ clock = \frac{byte\ clock\ *\ number\ of\ RX\ lanes * 8}{raw\ data\ width}$$

In most cases, you have to create GPLL module to generate pixel clock from the continuous byte clock except for RAW8 with 1-lane configuration. You can use the ipx file (int_gpll/int_gpll.sbx) 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 in the simulation setup file. Otherwise, you need to modify the names accordingly. In the example shown in Figure 3.14, the desired pixel clock (148.5 MHz) cannot be created on CLKOP. Therefore, a dummy clock output is assigned to CLKOP and pixel clock is assigned to CLKOS (CLKI Frequency is 46.40625 MHz. The last digit *5* is not shown in the GUI).

Figure 3.14. GPLL IP Creation for Pixel Clock

Following is a code snippet from csi2_4to1_NX.v. You have to modify and make a proper PLL configuration and instantiation. If condition is met, it is possible to use only one GPLL to create necessary clocks (sync_clk, pixel clock, and continuous byte clock if necessary).

```
////////////////////////////////////////////////////////////////////////////
///// GPLL instantiation                                                /////
///// User has to modify this if necessary                              /////
////////////////////////////////////////////////////////////////////////////
`ifdef RX_CLK_MODE_HS_ONLY

//     assign clk_pixel = clk_byte_fr;   // 1 lane, RAW8

       int_gpll pll_inst (
              .rstn_i             (1'b1),
              .clki_i             (clk_byte_fr),
              .clkop_o      (clk_pixel),
              .lock_o             (pll_lock_p)
       );

/*
       ///// try this when pixel clock cannot get from clkop /////
       int_gpll pll_inst (
              .rstn_i             (1'b1),
              .clki_i             (clk_byte_fr),
              .clkop_o      (clk_byte_fr_pll),  // same as clk_byte_fr : dummy, not in use
              .clkos_o      (clk_pixel),
              .lock_o             (pll_lock_p)
       );
*/
`elsif RX_CLK_MODE_HS_LP
       int_gpll_nc pll_inst (
              .rstn_i             (1'b1),
              .clki_i             (ref_clk_i),  // 27 MHz
              .clkop_o      (clk_pixel),  // 148.5 MHz
              .clkos_o      (clk_byte_pll),      // 46.40625 MHz
              .lock_o             (pll_lock_p)
       );
`endif
////////////////////////////////////////////////////////////////////////////
```

4. Design and File Modification

This Reference Design is based on version 1.0.1 of the RX D-PHY IP and version 1.0.2 of the Byte-to-Pixel Converter IP. Some modifications are required depending on your configuration in addition to three directive files (synthesis_directives.v, synthesis_directives_sim.v, and simulation_directives.v).

4.1. Top Level RTL

According to the configuration you use, you might have to modify the GPLL instantiation as described in sync_clk_gpll and/or int_gpll sections.

In addition, instance names of RX D-PHY, Byte-to-Pixel, and PLL IP have to be modified if you created these IP with different names.

4.2. Image Sensor Configuration

The current design is made for Sony IMX258 image sensors. You need to make necessary modifications if you want to configure image sensors for your application.

5. Design Simulation

The script file (*csi2_4to1_NX_fsim.do*) and testbench files are provided to run the functional simulation by the ModelSim software. For the script file to run, keep the current design names and instance names when RX D-PHY, Byte-to-Pixel, GPLL, and ROM IPs are created by the Lattice Radiant software. You must launch the ModelSim Lattice Edition software from the Lattice Radiant software to import the necessary libraries.

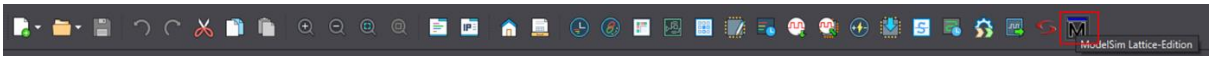

Figure 5.1. Launching the ModelSim Lattice Edition Software from the Lattice Radiant Software

You need to modify both *synthesis_directives_sim.v* and *simulation_directives.v* according to your configuration (refer to the Simulation Directives section for details). By executing the script in the ModelSim software, compilation and simulation are executed automatically. The testbench takes all data comparison between the expected data and output data from the RD. Note that the testbench is designed to make the data comparison based on the condition of default selection of tx_mux and isp_mux, which means quadrant-based 4 to 1 aggregation with no median filter processing. In case of RGB output, data comparison begins from the second input frame due to VSYNC signal creation delay. It shows the following statements while running and doing data comparison:

```
# KERNEL:  527660787 DPHY Lane 2 : Driving with data = 0d
# KERNEL:  527660787 DPHY Lane 3 : Driving with data = 85
# KERNEL: Frame 1, Line 2, Pixel 977 Active Data : R = 59, G = 43, B = 3b matched from CH #1
# KERNEL: Frame 1, Line 2, Pixel 978 Active Data : R = 44, G = 5f, B = 83 matched from CH #1
# KERNEL: Frame 1, Line 2, Pixel 979 Active Data : R = 8a, G = 0b, B = cb matched from CH #1
# KERNEL:  527682347 DPHY CH 0 Driving Data
# KERNEL:  527682347 DPHY Lane 0 : Driving with data = 92
# KERNEL:  527682347 DPHY Lane 1 : Driving with data = 0d
# KERNEL:  527682347 DPHY Lane 2 : Driving with data = 8b
# KERNEL:  527682347 DPHY Lane 3 : Driving with data = 62
# KERNEL:  527682347 DPHY CH 1 Driving Data
# KERNEL:  527682347 DPHY Lane 0 : Driving with data = b2
# KERNEL:  527682347 DPHY Lane 1 : Driving with data = 48
# KERNEL:  527682347 DPHY Lane 2 : Driving with data = d6
# KERNEL:  527682347 DPHY Lane 3 : Driving with data = da
# KERNEL:  527682347 DPHY CH 2 Driving Data
# KERNEL:  527682347 DPHY Lane 0 : Driving with data = 99
# KERNEL:  527682347 DPHY Lane 1 : Driving with data = c2
# KERNEL:  527682347 DPHY Lane 2 : Driving with data = 57
# KERNEL:  527682347 DPHY Lane 3 : Driving with data = bc
# KERNEL:  527682347 DPHY CH 3 Driving Data
# KERNEL:  527682347 DPHY Lane 0 : Driving with data = 12
# KERNEL:  527682347 DPHY Lane 1 : Driving with data = 96
# KERNEL:  527682347 DPHY Lane 2 : Driving with data = d4
# KERNEL:  527682347 DPHY Lane 3 : Driving with data = bd
# KERNEL: Frame 1, Line 2, Pixel 980 Active Data : R = d0, G = 6d, B = 9c matched from CH #1
# KERNEL: Frame 1, Line 2, Pixel 981 Active Data : R = db, G = 5f, B = 6c matched from CH #1
# KERNEL: Frame 1, Line 2, Pixel 982 Active Data : R = e5, G = 57, B = 7f matched from CH #1
# KERNEL:  527703907 DPHY CH 0 Driving Data
# KERNEL:  527703907 DPHY Lane 0 : Driving with data = 0b
# KERNEL:  527703907 DPHY Lane 1 : Driving with data = 5b
```

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

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

```
# KERNEL:  596784205 DPHY CH 3 CLK : Driving CLK-Stop
# KERNEL:  596784205 DPHY CSI-2 before frame gap
# KERNEL:
# KERNEL:  596784205 DPHY CH 2 CLK : Driving CLK-Stop
# KERNEL:  596784205 DPHY CSI-2 before frame gap
# KERNEL:
# KERNEL:  596784205 DPHY CH 1 CLK : Driving CLK-Stop
# KERNEL:  596784205 DPHY CSI-2 before frame gap
# KERNEL:
# KERNEL:  596784205 DPHY CH 0 CLK : Driving CLK-Stop
# KERNEL:  596784205 DPHY CSI-2 before frame gap
# KERNEL:
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### VS assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### VS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL:  746784205 DPHY CSI-2 after frame gap
# KERNEL:
# KERNEL:  746784205 DPHY CSI-2 FS begins...
# KERNEL:
# KERNEL:  746784205 DPHY CSI-2 after frame gap
```

When the simulation is finished, the following statements are displayed:

```
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### VS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
```

```
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: ##### HS assertion #####
# KERNEL: ##### HS de-assertion #####
# KERNEL: 1055900705 DPHY CSI-2 after frame gap
# KERNEL:
# KERNEL: 1055900705 DPHY CSI-2 after frame gap
# KERNEL:
# KERNEL: 1055900705 DPHY CSI-2 after frame gap
# KERNEL:
# KERNEL: 1055900705 DPHY CSI-2 after frame gap
# KERNEL:
# KERNEL: Total pixels checked were 22800 = 2 Frames x 1900 pels x 6 lines : all matched !!!!!
# KERNEL: Simulation Succeeded !!!!!
# KERNEL: 1061000705 TEST END
```

The above is the result after running three frames. 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 in case of RGB output with the first and the last line trimming. D-PHY clock and data lanes are shown only a part of channel #3 due to the display line limitation. Due to the Raw to RGB conversion, there exists one line delay for RGB data. Also the vertical filter causes another one-line delay for noise filter outputs (mfv_*). As described in Sync Signal Generation section, uneven interval of HSYNC happens at frame boundaries.

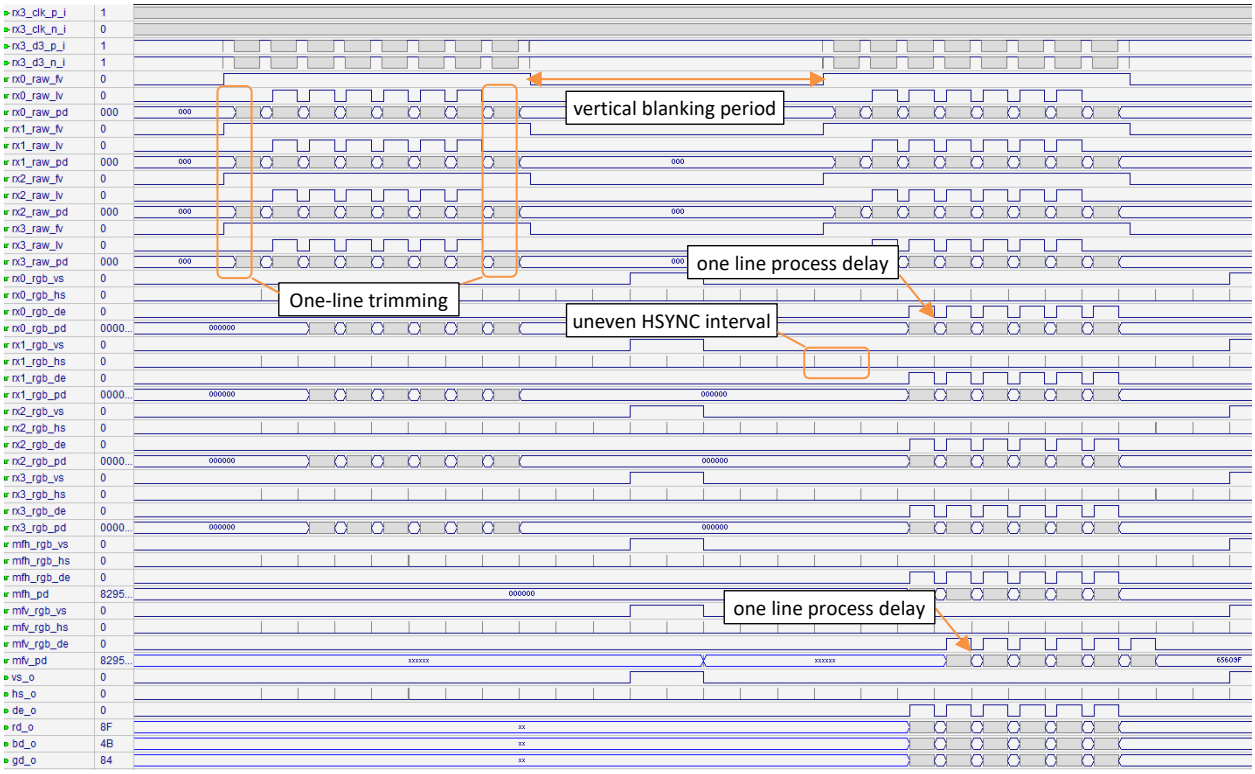

Figure 5.2. Global Timing of RGB output with Vertical Trimming

6. Known Limitations

The following is a limitation of this reference design:

- The I2C controller implementation in this reference design only focuses on writing to the I2C target devices with fixed configurations. This reference design does not support reading registers from I2C target devices.

7. Design Package and Project Setup

4 to 1 Image Aggregation with CrossLink-NX device reference design is available on www.latticesemi.com. Figure 7.1 shows the directory structure of the design package. The design is targeted for LIFCL-40-7BG400I. *synthesis_directives.v, sysnthesis_directives_sim.v* and *simulation_directives.v* are set to configure an example shown below:

- RX : RAW10 four lanes with two Hard D-PHY, two Soft D-PHY in continuous clock mode
- TX : parallel RGB888

You can modify the directives for own configuration.

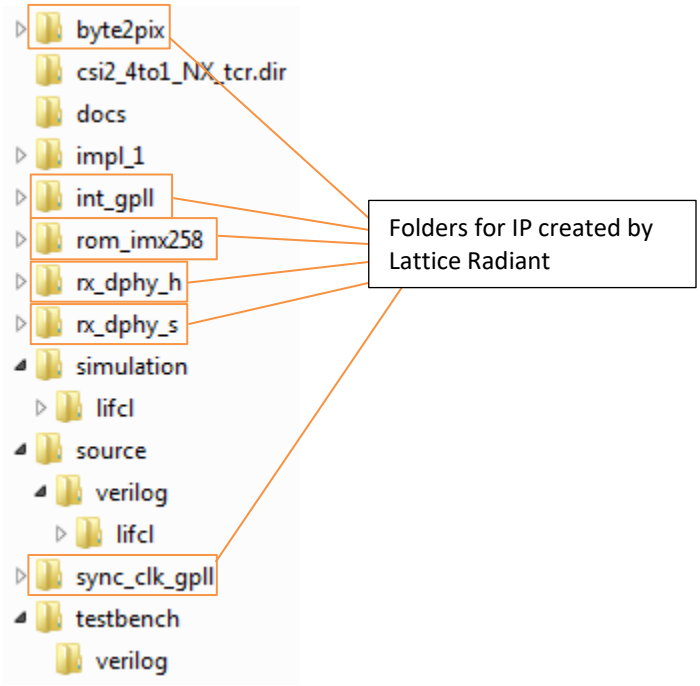

Figure 7.1. Directory Structure

Folders byte2pix, int_gll, rom_imx258_lut, rx_dphy_h, rx_dphy_s are created by Lattice Radiant software for corresponding IPs.

Figure 7.2 shows design files used in the Lattice Radiant project. Lattice Radiant creates five .ipx files. In this example, raw2rgb is encrypted. csi2_4to1_NX must be specified as a top-level design.

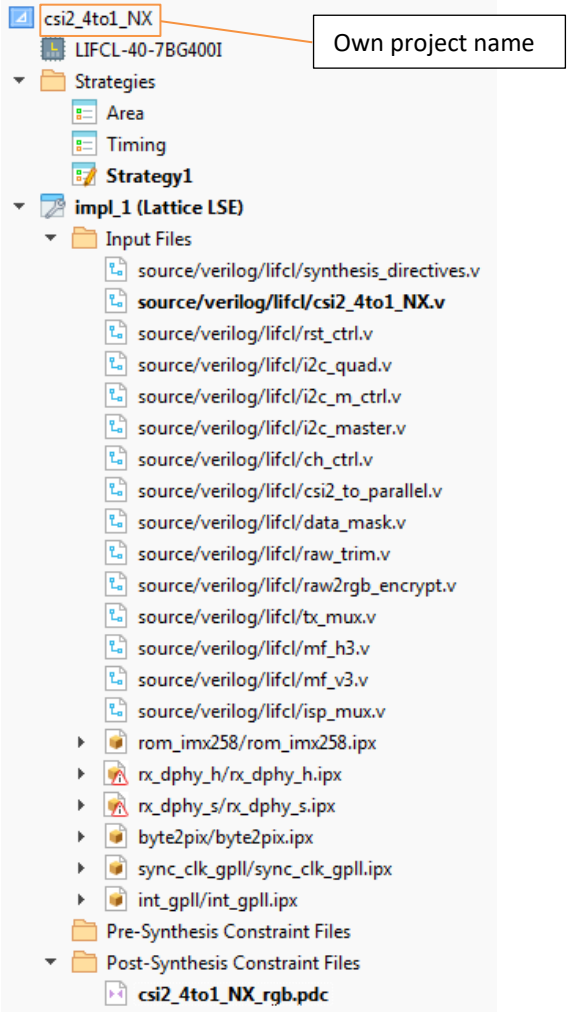

Figure 7.2. Project Files

Constraint file (csi2_4to1_NX_rgb.pdc) is also included in the project for reference. You can modify it according to your own configuration.

This sample project is targeted for CrossLink-NX VIP Sensor Input Board and functionality is confirmed. Refer to 4 to 1 Image Aggregation with CrossLink-NX VIP Sensor Input Board User Guide (FPGA-UG-02095) for details.

8. Resource Utilization

Resource utilization depends on the configurations. Table 8.1 shows resource utilization examples under certain configurations targeting LIF-40. LB_depth is 2048. Actual usage may vary.

Table 8.1. Resource Utilization Examples

Configuration	LUT	FF	EBR	I/O
Two Hard D-PHY RX + two Soft D-PHY RX with 4 lanes, RAW10 RX to RGB888 output	12699	9442	39	70
Two Hard D-PHY RX + two Soft D-PHY RX with 4 lanes, RAW10 RX to RAW10 output	9301	6823	13	54
Two Hard D-PHY RX + two Soft D-PHY RX with 2 lanes, RAW10 RX to RGB888 output	11085	8472	35	62
Two Hard D-PHY RX + two Soft D-PHY RX with 2 lanes, RAW10 RX to RAW10 output	7671	5835	9	46
Two Hard D-PHY RX + two Soft D-PHY RX with 1 lane, RAW12 RX to RGB888 output	9743	7742	31	58
Two Hard D-PHY RX + two Soft D-PHY RX with 1 lane, RAW12 RX to RAW12 output	6137	4964	5	44

9. References

- MIPI® Alliance Specification for D-PHY Version 1.1
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.1
- CSI-2/DSI D-PHY Rx IP Core User Guide (FPGA-IPUG-02081)
- Byte-to-Pixel IP Core User Guide (FPGA-IPUG-02079)
- CrossLink-NX web page
- Lattice Radiant Software for Lattice Radiant Project-based Environment, Design Flow, Implementation Flow, Tasks, Simulation Flow, and Lattice Radiant User Guide
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

For assistance, submit a technical support case at www.latticesemi.com/techsupport.

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

Revision History

Revision 1.3, October 2023

Section	Change Summary
Disclaimers	Updated disclaimers.
Inclusive Language	Added inclusive language boilerplate.
Acronyms in This Document	Added acronyms definition.
Supported Device and IP	• Updated D-PHY Receiver IP version from 1.1.1 to 1.5.0. • Updated Byte-To-Pixel Convertor IP version from 1.1.1 to 1.6.0. • Updated PLL IP version from 1.3.0 to 1.8.0. • Added I2C Controller IP version 2.0.0. • Changed the Lattice Radiant software version 3.0 to version 2023.1.1.

Introduction	• Updated the following diagrams per the the updated I2C implementation:
	  • Figure 1.1. Block Diagram of 4 to 1 Image Aggregation with CrossLink-NX Devices (RGB Output)
	  • Figure 1.2. Block Diagram of 4 to 1 Image Aggregation with CrossLink-NX Devices (RAW Output)
	  • Figure 1.3. Sample Clocking Scheme in Continuous Clock Mode
	  • Figure 1.4. Sample Clocking Scheme in Non-Continuous Clock Mode
	• Updated Figure 1.5. Clock Frequency Calculator to include data settle cycle calculation for Soft D-PHY in the RX Permutations section.

Parameters and Port List	• Removed Sensor I^2C Address, High Time of I^2C Clock, Low Time of I^2C Clock, and Gap time between I^2C transactions categories in Table 2.1. Synthesis Directives. • Added Wait for I2C data transfer completion category in Table 2.2. Simulation Directives.
Design and Module Description	• Implemented I2C programming to IMX258 sensor using the packaged IP in the Lattice Radiant software instead of custom register transfer level (RTL) code. • Updated IP configurations and implementation to the latest version.
Design Simulation	• Migrated simulation setup from Active HDL to the ModelSim Lattice Edition software. • Updated Figure 5.2. Global Timing of RGB output with Vertical Trimming.
Known Limitations	Updated the limitation of the reference design.
Resource Utilization	Updated Table 8.1. Resource Utilization Examples.
Technical Support Assistance	Added link to the Lattice Answer Database.
All	Updated Master and Slave terminologies to Controller and Target.

Revision 1.2, June 2021

Section	Change Summary
Supported Device and IP	• Changed D-PHY Receiver IP version from 1.0.1 to 1.1.1. • Changed Byte-To-Pixel Convertor IP version from 1.0.2 to 1.1.1. • Added PLL IP version 1.3.0. • Changed Lattice Radiant software version 2.0 service pack 1 to version 3.0.

Revision 1.1, Feburary 2020

Section	Change Summary
Supported Device and IP	• Added LIFCL-17 as a supported device. • Updated IP version number. • Added Lattice Radiant version support information.
Design and Module Description	• Added RX FIFO in Hard D-PHY RX IP in rx_dphy_h section. • Added RX FIFO in Soft D-PHY RX IP in rx_dphy_s section. • Changed RGB CMOS to RGB.
Design Package and Project Setup	Added reference to FPGA-UG-02095.

Revision 1.0, December 2019

Section	Change Summary
All	Initial release

www.latticesemi.com