N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation

Reference Design

FPGA-RD-02192-1.0

April 2020

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AP	Application Processor
CSI-2	Camera Serial Interface 2
DDR	Double Data Rate
EBR	Embedded Block RAM
ECC	Error Correction Code
HS	High Speed
ID	Identification Data
LP	Low Power
MIPI	Mobile Industry Processor Interface
PLL	Phase Locked Loop
GPLL	General Purpose PLL
RX	Receiver
SoT	Start of Transmission
TX	Transmitter
VC	Virtual Channel
VCO	Voltage Controlled Oscillator
WC	Word Count

1. Introduction

The majority of image sensors and application processors (AP) in the consumer market use the Mobile Industry Processor Interface (MIPI®) Camera Serial Interface 2 (CSI-2) as a video signal interface. In some cases, the AP has to take multiple image data for various applications without increasing the physical interface signals.

The Lattice Semiconductor N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation reference design for CrossLink™ devices offers up to five-channel aggregation. Multiple channel image data are concatenated horizontally line by line. CrossLink has two MIPI hard macro IPs, which can be used as MIPI TX or RX module (D-PHY Hard IP). The RX module can also be realized by a soft macro utilizing general DDR modules (D-PHY Soft IP).

1.1. Supported Device and IP

This reference design supports the following devices with IP versions:

Table 1.1. Supported Device and IP

Device Family	Part Number	Compatible IP
CrossLink	LIF-MD6000 LIA-MD6000	D-PHY Receiver IP version 1.4 D-PHY Transmitter IP version 1.3
CrossLinkPlus™	LIF-MDF6000	

Note: *CrossLink* refers to both CrossLink and CrossLinkPlus in this document unless noted.

1.2. Features

- Two to five RX channels can be aggregated.
- One D-PHY Hard IP is used on TX channel.
- One Hard D-PHY IP is recommended to be used on RX Channel 0 to save FPGA resources.
- All RX channels must be in the same configuration.
- RX channels do not necessarily have to share the same clock source as long as their clock tolerance is within 500 ppm.
- All RX channels are expected to have almost same frame timing (20-pixel timing or less).
- RX channel can have one, two, or four lanes as long as the total number of RX clock and data lanes by RX D-PHY Soft IP does not exceed 15.
- Maximum RX bandwidth is 1.2 Gbps per lane.
- Number of TX lanes can be one, two, or four.
- Maximum TX bandwidth is 1.5 Gbps per lane.
- Non-continuous clock mode on RX channels is possible as long as the continuous clock can be obtained internally or fed directly from the pin.
- External reference clock is required in case RX channels are in non-continuous clock mode.

1.3. Block Diagram

Figure 1.1 shows the block level diagram of the MIPI CSI-2 Side-by-Side Aggregation reference design with five RX channels.

Since TX D-PHY PLL has an input clock frequency requirement of between 24 MHz and 30 MHz (or a multiple of between 24 and 30 MHz), an on-chip GPLL may have to be used to create an appropriate clock.

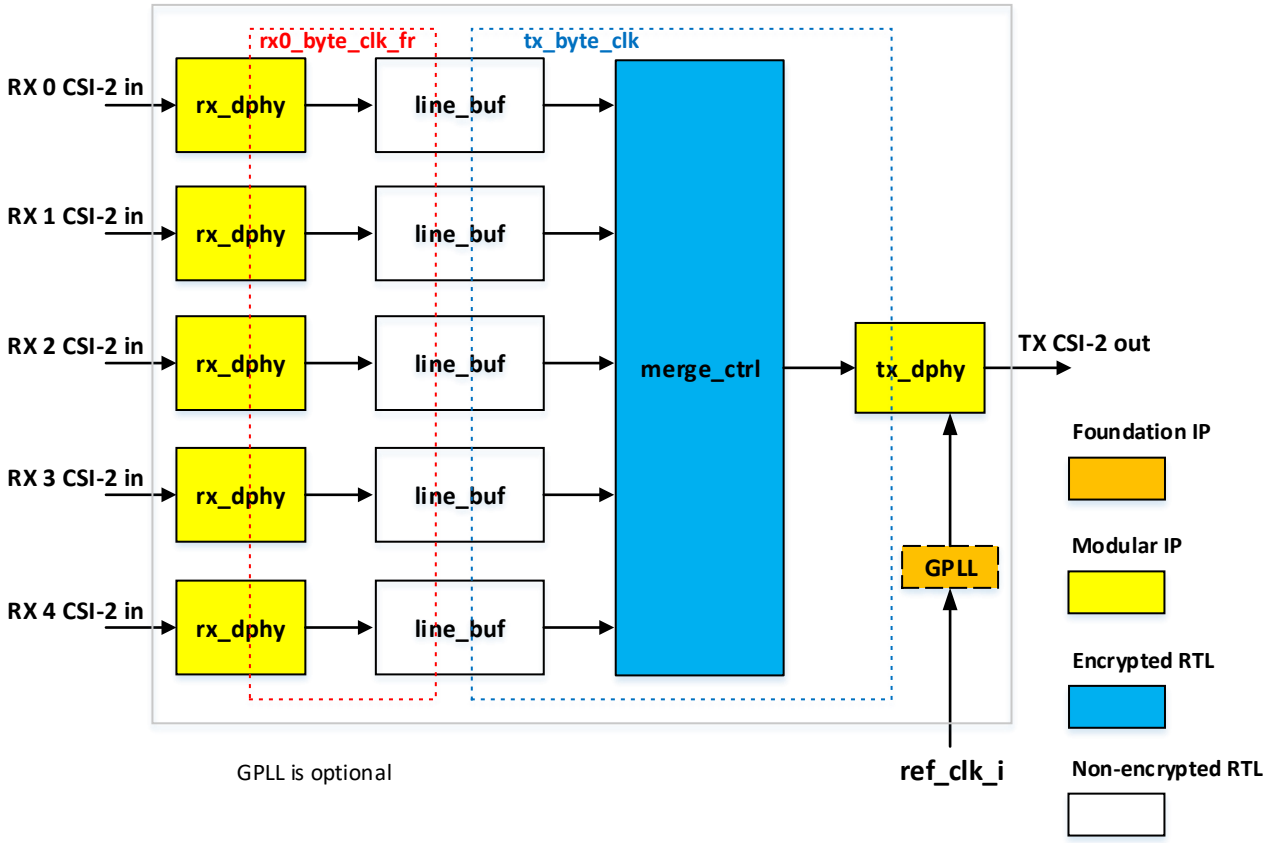

Figure 1.1. MIPI CSI-2 Side-by-Side Aggregation Block Diagram

Figure 1.2 shows an example of clocking scheme in continuous clock mode. If continuous byte clock cannot be directly fed to tx_dphy, using GPLL could be a solution.

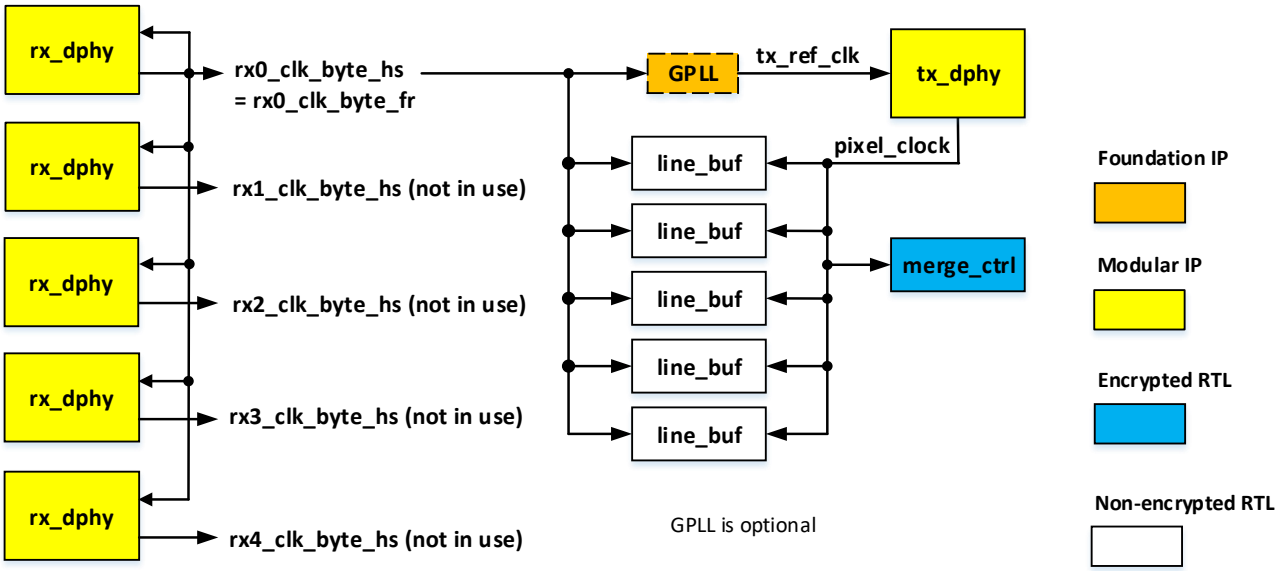

Figure 1.2. Clocking Scheme Example in Continuous Clock Mode

Figure 1.3 shows an example of clocking scheme in non-continuous clock mode. In this case, the external clock (ref_clk_i) is mandatory to generate both continuous byte clock and tx_ref_clk. GPLL may be bypassed if ref_clk_i is equal to rx0_byte_clk_fr and TX D-PHY can create tx_byte_clk from this external reference clock.

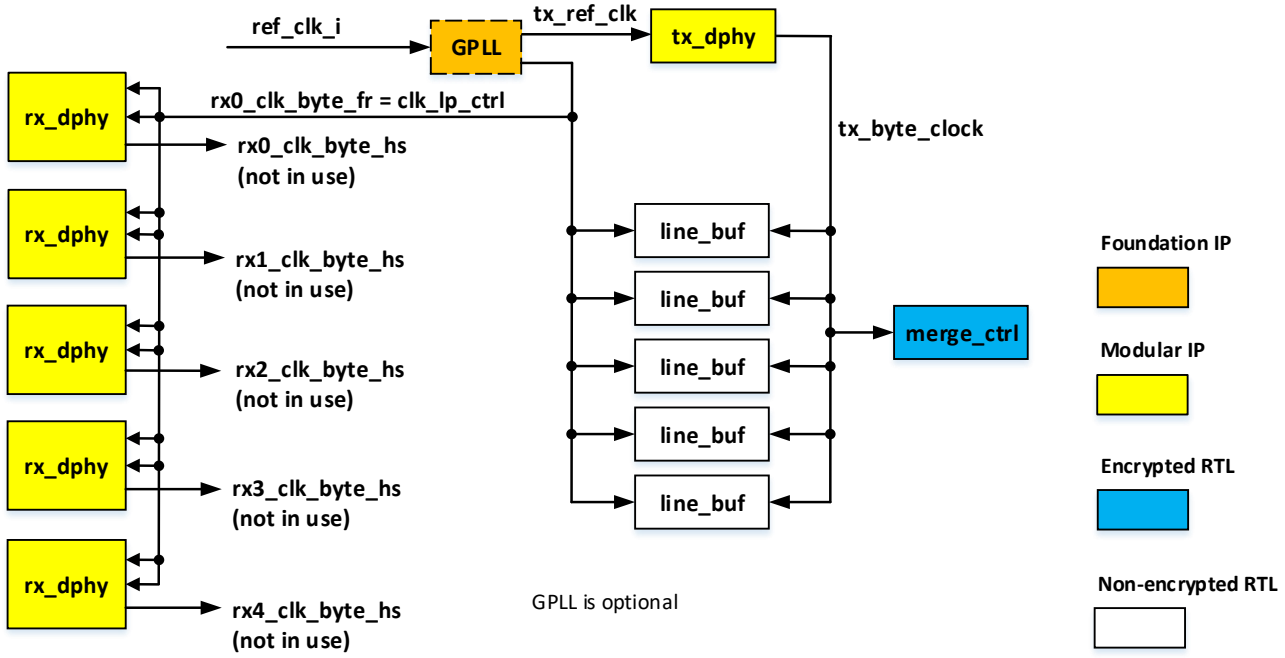

Figure 1.3. Clocking Scheme Example in Non-Continuous Clock Mode

1.4. RX/TX Permutations and Parameter Calculation

Table 1.2 shows available permutations of RX and TX configurations. Due to the minimum lane bandwidth supported by D-PHY RX IP (160 Mbps) and number of differential I/O pairs (up to 15 differential I/O for Soft D-PHY RX), some permutations are not possible and grayed out in the table.

Table 1.2. RX and TX Permutation

Number of RX Channels	Number of RX Lanes	RX Lane Bandwidth (Mbps)	Number of TX Lanes	TX Lane Bandwidth/RX Lane Bandwidth
2	1	160 - 750	1	2
		160 -1200	2	1
		160 - 1200	4	0.5
	2	160 - 375	1	4
		160 - 750	2	2
		160 - 1200	4	1
	4	160 - 187.5	1	8
		160 - 375	2	4
		160 - 750	4	2
3	1	160 - 500	1	3
		160 - 1000	2	1.5
		160 - 1200	4	0.75
	2	160 - 250	1	6
		160 - 500	2	3
		160 - 1000	4	1.5

Number of RX Channels	Number of RX Lanes	RX Lane Bandwidth (Mbps)	Number of TX Lanes	TX Lane Bandwidth/RX Lane Bandwidth
4	4	≤ 125*	1*	12*
		160 - 250	2	6
		160 - 500	4	3
	1	160 - 375	1	4
		160 - 750	2	2
		160 - 1200	4	1
	2	160 - 187.5	1	8
		160 - 375	2	4
		160 - 750	4	2
	4	≤ 93.75*	1*	16*
		160 - 187.5	2	8
		160 - 375	4	4
5	1	160 - 300	1	5
		160 - 600	2	2.5
		160 - 1200	4	1.25
	2	≤ 150*	1*	10*
		160 - 300	2	5
		160 - 600	4	2.5
	4*	≤ 75*	1*	20*
		≤ 150*	2*	10*
		160 – 300*	4*	5*

*__Note__: These permutations are not supported.

The Excel sheet (csi2_aggregation_ss_param.xls) is provided to calculate the byte clock, number of channels, and other information. Also, some parameter values can be obtained from RX Data Type and horizontal pixel count. Figure 1.4 shows a sample entry of this sheet. In CSI-2, the amount of active video data per line must be a multiple of 8 bits and that sets some restrictions on horizontal pixel count according to RX Data Type. This amount shows up as Word Count in the sheet and as a byte count of active video data per line when you enter the Horizontal Pixel Count value. You must enter the correct Horizontal Pixel Count value so that Word Count has an integer value. Less common data types (for example, RGB555) are not covered by this sheet, but it is possible to support by manually entering the Word Count value in the sheet as long as the value is integer.

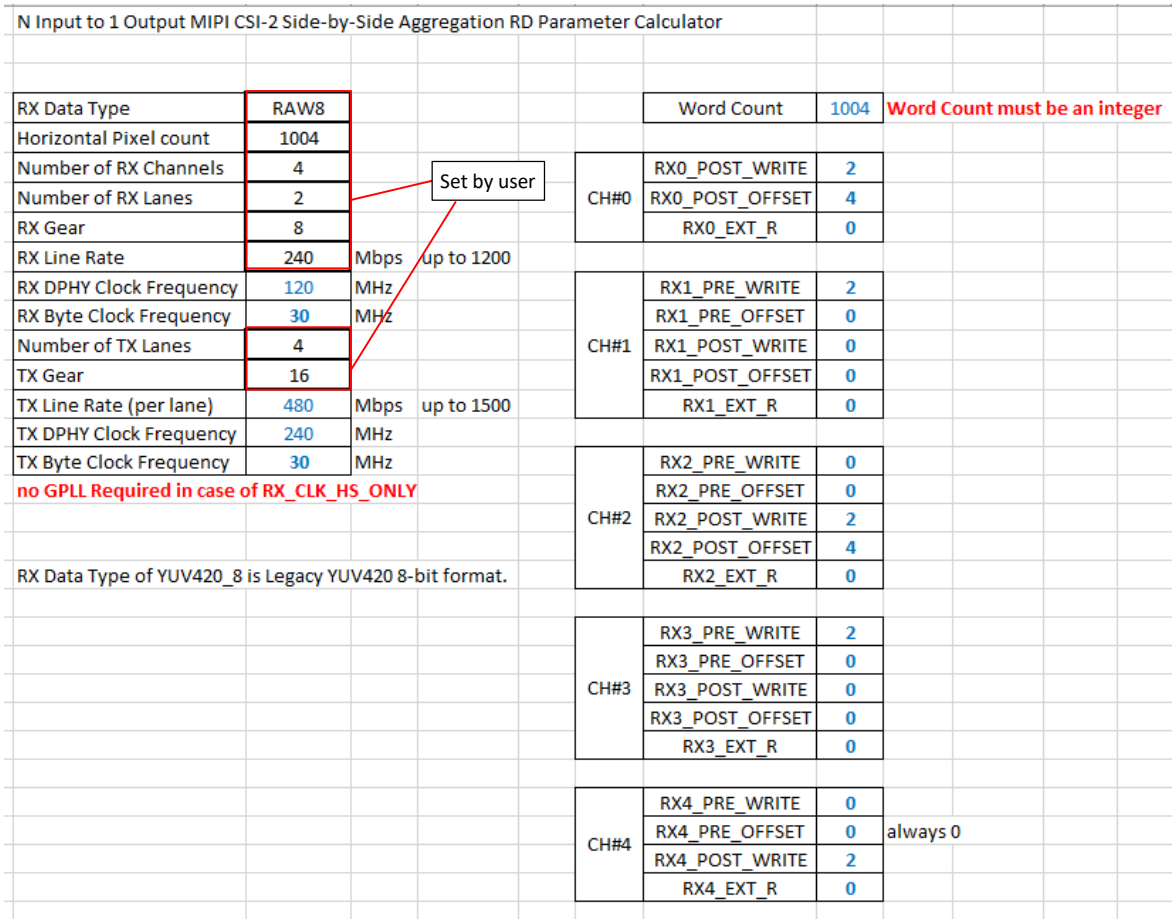

Figure 1.4. Bandwidth and Parameter Calculator

2. Parameters and Port List

There are two directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Diamond® and 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, TX D-PHY IP, and other module settings described in the Design and Module Description section.

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
Frame Sync Output	FS_OUT	Enable to output Frame Sync Pulse when defined.
Frame Sync Interval	FS_INTERVAL {value}	Interval period of Frame Sync in ref_clk_i cycles. Effective when FS_OUT is defined. Value must be equal or less than 20'd1048575.
Frame Sync Pulse Length	FS_LENGTH {value}	Pulse width of Frame Sync in ref_clk_i cycles. Effective when FS_OUT is defined. Value must be 8'd1 – 8'd255.
Frame Sync Polarity	FS_POL_POS	Polarity of Frame Sync Pulse. Effective when FS_OUT is defined. Only one of these two directives must be defined.
	FS_POL_NEG	
Data Type	RAW8	Data Type of the payload video data. Only one of these nine directives must be defined.
	RAW10	
	RAW12	
	RAW14	
	RGB888	
	RGB666	
	RGB565	
	YUV422_8	
	YUV422_10	
RX Channel Count	NUM_RX_CH_2	Number of RX channels aggregated. Only one of these four directives must be defined.
	NUM_RX_CH_3	
	NUM_RX_CH_4	
	NUM_RX_CH_5	
RX Channel Lane Count	NUM_RX_LANE_1	Number of lanes in each 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	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations.
	RX_GEAR_16	
RX Hard D-PHY Channel	RX0_DPHY_HARD	Specify RX channel that uses Hard D-PHY. Only one of these five directives can be defined. If none of these is defined, all RX channels use Soft D-PHY (up to four channels).
	RX1_DPHY_HARD	
	RX2_DPHY_HARD	
	RX3_DPHY_HARD	
	RX4_DPHY_HARD	

Category	Directive	Remarks
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	
Use GPLL	USE_GPLL	Use GPLL to create a reference clock to be fed to PLL of TX D-PHY IP.
Hard D-PHY Word Alignment	WORD_ALIGN	Enable word aligner in RX Hard D-PHY IP. Ignored when Hard D-PHY is not used.
RX Buffer Size[2]	RX0_BUF_SIZE_*	RX channel 0 FIFO buffer size in byte. * must be 1024, 2048, 4096, or 8192.
	RX1_BUF_SIZE_*	RX channel 1 FIFO buffer size in byte. * must be 1024, 2048, 4096, or 8192.
	RX2_BUF_SIZE_*	RX channel 2 FIFO buffer size in byte. * must be 1024, 2048, or 4096.
	RX3_BUF_SIZE_*	RX channel 3 FIFO buffer size in byte. * must be 1024, 2048, or 4096.
	RX4_BUF_SIZE_*	RX channel 4 FIFO buffer size in byte. * must be 1024, 2048, or 4096.
RX Buffer Pre-Write[3]	RX1_PRE_WRITE {value}	Pre-write cycle count to RX channel 1 FIFO buffer. Value must be 0 – 7.
	RX2_PRE_WRITE {value}	Pre-write cycle count to RX channel 2 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX3_PRE_WRITE {value}	Pre-write cycle count to RX channel 3 FIFO buffer. Value must be 0 – 7.
	RX4_PRE_WRITE {value}	Pre-write cycle count to RX channel 4 FIFO buffer. Value must be 0 or 4.
RX Buffer Pre-Offset[3]	RX1_PRE_OFFSET {value}	Pre-offset byte count to RX channel 1 FIFO buffer. Value must be 0 – 3.
	RX2_PRE_OFFSET {value}	Pre-offset byte count to RX channel 2 FIFO buffer. Value must be 0 or 2.
	RX3_PRE_OFFSET {value}	Pre-offset byte count to RX channel 3 FIFO buffer. Value must be 0 – 3.
RX Buffer Post-Write[3]	RX0_POST_WRITE {value}	Post-write cycle count to RX channel 0 FIFO buffer. Value must be 0 – 7.
	RX1_POST_WRITE {value}	Post-write cycle count to RX channel 1 FIFO buffer. Value must be 0 – 7.
	RX2_POST_WRITE {value}	Post-write cycle count to RX channel 2 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX3_POST_WRITE {value}	Post-write cycle count to RX channel 3 FIFO buffer. Value must be 0 or 4.
	RX4_POST_WRITE {value}	Post-write cycle count to RX channel 4 FIFO buffer. Value must be 0 - 7.
RX Buffer Post-Offset[3]	RX0_POST_OFFSET {value}	Post-offset byte count to RX channel 0 FIFO buffer. Value must be 0 – 7.
	RX1_POST_OFFSET {value}	Post-offset byte count to RX channel 1 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX2_POST_OFFSET {value}	Post-offset byte count to RX channel 2 FIFO buffer. Value must be 0 - 7.
	RX3_POST_OFFSET {value}	Post-offset byte count to RX channel 3 FIFO buffer. Value must be 0 or 4.
RX Buffer Extra Read[3]	RX0_EXT_R {value}	Extra Read Flag for RX channel 0 FIFO Buffer. Value must be 0 or 1.
	RX1_EXT_R {value}	Extra Read Flag for RX channel 1 FIFO Buffer. Value must be 0 or 1.
	RX2_EXT_R {value}	Extra Read Flag for RX channel 2 FIFO Buffer. Value must be 0 or 1.
	RX3_EXT_R {value}	Extra Read Flag for RX channel 3 FIFO Buffer. Value must be 0 or 1.
	RX4_EXT_R {value}	Extra Read Flag for RX channel 4 FIFO Buffer. Value must be 0 or 1.
Long Packet Transfer Request Delay	LP_TX_REQ_DLY {value}	Long Packet Transfer Request Delay Time in tx_byte_clk cycles. Value must be 1 – 4095. Automatically calculated value is used if not defined. Refer to Line Buffer Read and Data Merge Operation section for details.
Virtual Channel ID	VC {value}	Virtual Channel ID. Value must be 2'd0 – 2'd3. The value on RX channel 0 is used if not defined.
TX channel lane count	NUM_TX_LANE_1	Number of lanes in TX channel. Only one of these three directives must be defined.
	NUM_TX_LANE_2	
	NUM_TX_LANE_4	
TX D-PHY Clock Gear	TX_GEAR_8	TX D-PHY Clock Gear. Only one of these two directives must be defined.
	TX_GEAR_16	

Category	Directive	Remarks
TX D-PHY Clock Mode[4]	TX_CLK_MODE_HS_ONLY	TX D-PHY Clock mode. Only one of these two directives must be defined.
	TX_CLK_MODE_HS_LP	
TX Lane Bandwidth	TX_LANE_BW {value}	TX D-PHY Lane Bandwidth in Mbps. Effective when TX_CLK_MODE_HS_LP is defined.
Keep HS mode	KEEP_HS	Keep the clock lane in HS mode during the horizontal blanking periods of active video lines when defined. Effective when TX_CLK_MODE_HS_LP is defined.

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 obtained directly or indirectly from the external clock.
2. This value affects necessary EBR used in the device. Total number of EBR must not exceed 20. Refer to line_buf section for details.
3. These parameter values are obtained by the provided Excel sheet as shown in Figure 1.4.
4. HS_LP mode means *non-continuous clock mode* and HS_ONLY means *continuous clock mode*. This mode does not have to be same as RX D-PHY Clock Mode.

2.2. Simulation Directives

Table 2.2 shows the simulation directives for this reference design.

Table 2.2. Simulation Directives

Category	Directive	Remarks
Simulation	SIM	Select behavioral models for simulation.
Reference clock period	REF_CLK_PERIOD {value}	Reference clock period in ps
RX D-PHY clock period	RX0_DPHY_CLK_PERIOD {value}	RX Channel 0 DPHY clock period in ps
	RX1_DPHY_CLK_PERIOD {value}	RX Channel 1 DPHY clock period in ps
	RX2_DPHY_CLK_PERIOD {value}	RX Channel 2 DPHY clock period in ps
	RX3_DPHY_CLK_PERIOD {value}	RX Channel 3 DPHY clock period in ps
	RX4_DPHY_CLK_PERIOD {value}	RX Channel 4 DPHY clock period in ps
Initial delay on RX channel	CH0_DELAY {value}	Initial delay to activate RX Channel 0 in ps
	CH1_DELAY {value}	Initial delay to activate RX Channel 1 in ps
	CH2_DELAY {value}	Initial delay to activate RX Channel 2 in ps
	CH3_DELAY {value}	Initial delay to activate RX Channel 3 in ps
	CH4_DELAY {value}	Initial delay to activate RX Channel 4 in ps
Gap (LP) time between active lines on RX Channel	RX_DPHY_LPS_GAP {value}	Horizontal Blanking Gap time on RX Channels in ps
Gap (LP) time between Frame End and Frame Start on RX Channel	RX_DPHY_FRAME_GAP {value}	Vertical Blanking Gap time on RX Channels in ps
RX Channel 0 VC	RX0_VC {value}	Virtual Channel ID on RX Channel 0. The value must be 4'd0 – 4'd3. The same values are used on all RX channels.
Video data configuration on RX Channels	NUM_FRAMES {value}	Number of frames to feed
	NUM_LINES {value}	Number of active lines per frame
Internal signal monitoring	MISC_ON	Enables internal signals to be monitored by the testbench. Always enable this directive.

2.3. Top-Level I/O

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

Table 2.3. CSI-2 Side-by-Side Aggregation Top-Level I/O

Port Name	Direction	Description
Clocks and Resets		
ref_clk_i (optional)	I	Input reference clock. Used to feed a clock to TX D-PHY PLL directly or indirectly. This port is declared only when RX_CLK_MODE_HS_LP or FS_OUT is defined in synthesis_directives.v.
reset_n_i	I	Asynchronous active low system reset
CSI-2 RX Interface		
fs_o (optional)	O	Frame Sync. Only declared when FS_OUT is defined in synthesis_directives.v.
rx0_clk_p_i	I	Positive differential RX Ch0 D-PHY input clock
rx0_clk_n_i	I	Negative differential RX Ch0 D-PHY input clock
rx0_d0_p_i	I	Positive differential RX Ch0 D-PHY input data 0
rx0_d0_n_i	I	Negative differential RX Ch0 D-PHY input data 0
rx0_d1_p_i	I	Positive differential RX Ch0 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx0_d1_n_i	I	Negative differential RX Ch0 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx0_d2_p_i	I	Positive differential RX Ch0 D-PHY input data 2 (in case of 4-lane configuration)
rx0_d2_n_i	I	Negative differential RX Ch0 D-PHY input data 2 (in case of 4-lane configuration)
rx0_d3_p_i	I	Positive differential RX Ch0 D-PHY input data 3 (in case of 4-lane configuration)
rx0_d3_n_i	I	Negative differential RX Ch0 D-PHY input data 3 (in case of 4-lane configuration)
rx1_clk_p_i	I	Positive differential RX Ch1 D-PHY input clock
rx1_clk_n_i	I	Negative differential RX Ch1 D-PHY input clock
rx1_d0_p_i	I	Positive differential RX Ch1 D-PHY input data 0
rx1_d0_n_i	I	Negative differential RX Ch1 D-PHY input data 0
rx1_d1_p_i	I	Positive differential RX Ch1 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx1_d1_n_i	I	Negative differential RX Ch1 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx1_d2_p_i	I	Positive differential RX Ch1 D-PHY input data 2 (in case of 4-lane configuration)
rx1_d2_n_i	I	Negative differential RX Ch1 D-PHY input data 2 (in case of 4-lane configuration)
rx1_d3_p_i	I	Positive differential RX Ch1 D-PHY input data 3 (in case of 4-lane configuration)
rx1_d3_n_i	I	Negative differential RX Ch1 D-PHY input data 3 (in case of 4-lane configuration)
rx2_clk_p_i	I	Positive differential RX Ch2 D-PHY input clock
rx2_clk_n_i	I	Negative differential RX Ch2 D-PHY input clock
rx2_d0_p_i	I	Positive differential RX Ch2 D-PHY input data 0
rx2_d0_n_i	I	Negative differential RX Ch2 D-PHY input data 0
rx2_d1_p_i	I	Positive differential RX Ch2 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx2_d1_n_i	I	Negative differential RX Ch2 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx2_d2_p_i	I	Positive differential RX Ch2 D-PHY input data 2 (in case of 4-lane configuration)
rx2_d2_n_i	I	Negative differential RX Ch2 D-PHY input data 2 (in case of 4-lane configuration)
rx2_d3_p_i	I	Positive differential RX Ch2 D-PHY input data 3 (in case of 4-lane configuration)
rx2_d3_n_i	I	Negative differential RX Ch2 D-PHY input data 3 (in case of 4-lane configuration)
rx3_clk_p_i	I	Positive differential RX Ch3 D-PHY input clock
rx3_clk_n_i	I	Negative differential RX Ch3 D-PHY input clock
rx3_d0_p_i	I	Positive differential RX Ch3 D-PHY input data 0
rx3_d0_n_i	I	Negative differential RX Ch3 D-PHY input data 0
rx3_d1_p_i	I	Positive differential RX Ch3 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx3_d1_n_i	I	Negative differential RX Ch3 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx3_d2_p_i	I	Positive differential RX Ch3 D-PHY input data 2 (in case of 4-lane configuration)

Port Name	Direction	Description
rx3_d2_n_i	I	Negative differential RX Ch3 D-PHY input data 2 (in case of 4 lane configuration)
rx3_d3_p_i	I	Positive differential RX Ch3 D-PHY input data 3 (in case of 4 lane configuration)
rx3_d3_n_i	I	Negative differential RX Ch3 D-PHY input data 3 (in case of 4 lane configuration)
rx4_clk_p_i	I	Positive differential RX Ch4 D-PHY input clock
rx4_clk_n_i	I	Negative differential RX Ch4 D-PHY input clock
rx4_d0_p_i	I	Positive differential RX Ch4 D-PHY input data 0
rx4_d0_n_i	I	Negative differential RX Ch4 D-PHY input data 0
rx4_d1_p_i	I	Positive differential RX Ch4 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx4_d1_n_i	I	Negative differential RX Ch4 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx4_d2_p_i	I	Positive differential RX Ch4 D-PHY input data 2 (in case of 4-lane configuration)
rx4_d2_n_i	I	Negative differential RX Ch4 D-PHY input data 2 (in case of 4-lane configuration)
rx4_d3_p_i	I	Positive differential RX Ch4 D-PHY input data 3 (in case of 4-lane configuration)
rx4_d3_n_i	I	Negative differential RX Ch4 D-PHY input data 3 (in case of 4-lane configuration)
CSI-2 TX Interface		
tx_clk_p_o	O	Positive differential TX D-PHY output clock
tx_clk_n_o	O	Negative differential TX D-PHY output clock
tx_d0_p_o	O	Positive differential TX D-PHY output data 0
tx_d0_n_o	O	Negative differential TX D-PHY output data 0
tx_d1_p_o	O	Positive differential TX D-PHY output data 1 (in case of 2-lane or 4-lane configuration)
tx_d1_n_o	O	Negative differential TX D-PHY output data 1 (in case of 2-lane or 4-lane configuration)
tx_d2_p_o	O	Positive differential TX D-PHY output data 2 (in case of 4-lane configuration)
tx_d2_n_o	O	Negative differential TX D-PHY output data 2 (in case of 4-lane configuration)
tx_d3_p_o	O	Positive differential TX D-PHY output data 3 (in case of 4-lane configuration)
tx_d3_n_o	O	Negative differential TX D-PHY output data 3 (in case of 4-lane configuration)

3. Design and Module Description

The top-level design (csi2_aggregation_ss.v) consists of the following modules:

- rx_dphy (rx_dphy_h / rx_dphy_s)
- line_buf
- merge_ctrl
- tx_dphy

The top-level design has a reset synchronization logic. In addition, GPLL may be added if necessary according to RX and TX configurations. When FS_OUT is defined, Frame Sync pulse (fs_o) is generated. This could be useful to give the frame timing synchronization to multiple camera sensors.

3.1. rx_dphy

Since all RX channels must be in the same configuration, a single module can cover all RX channels when Hard D-PHY is not in use. Assuming one of RX channels uses a Hard D-PHY IP, two modules are considered; rx_dphy_h for Hard D-PHY RX IP and rx_dphy_s for Soft D-PHY RX IP. Figure 3.1 shows an example of IP interface settings in Clarity for the CSI-2/DSI D-PHY Receiver Submodule IP. Refer to CSI-2/DSI D-PHY Receiver Submodule IP User Guide (FPGA-IPUG-02025) for details.

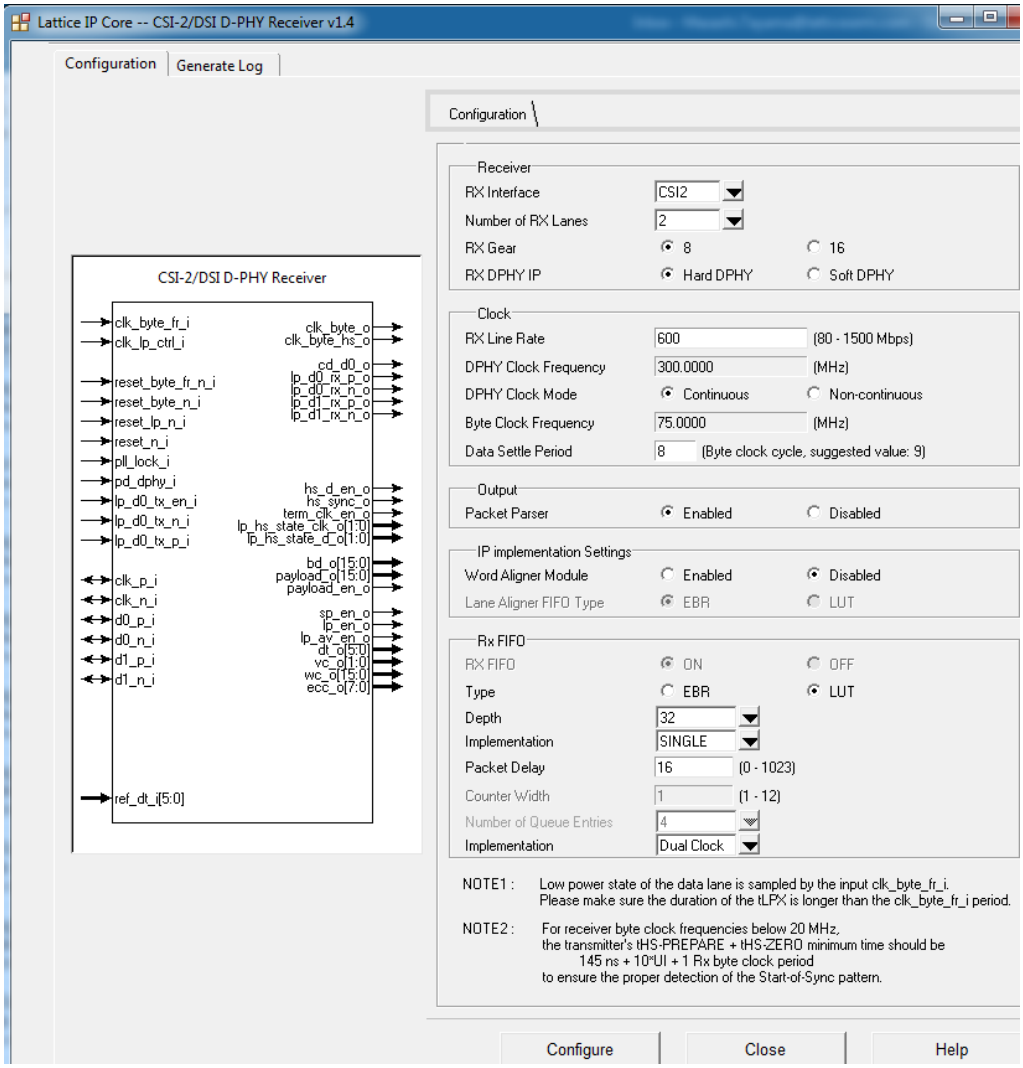

Figure 3.1. rx_dphy IP Creation in Clarity Designer

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

- RX Interface – Select CSI-2.
- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Select 8 or 16; 16 is supported for 1-lane and 2-lane configurations only. 16 is recommended when the RX byte clock speed exceeds 100 MHz or STA fails with Gear 8.
- RX D-PHY IP – Hard D-PHY can be selected only for one RX channel, others must be Soft D-PHY. The setting must match RX*_D-PHY_HARD setting.
- RX Line Rate – Set according to channel configuration. Must be 750 or below for 4 lane configuration.
- D-PHY Clock Mode – Select Continuous or Non-continuous. Must match RX_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- Data Settle Period – Enter the suggested value. You can try decreasing this value one by one when RX D-PHY is not operating properly.
- Packet Parser – Select Enabled.
- Word Aligner Module – Select Enabled in case of Hard D-PHY in high-temperature condition (ambient temperature > 125 °C). Automatically enabled for Soft D-PHY.
- Lane Aligner Module – LUT or EBR for 2 and 4 lane configuration. LUT is highly recommended.

Refer to RX FIFO section for RX FIFO related settings.

This module takes serial CSI-2 data and outputs byte data after de-serialization in CSI-2 High Speed mode. It is recommended to set the design name to *rx* and module names to *rx_dphy_h* and *rx_dphy_s* so that you do not need to modify the instance names of these IPs in the top-level design as well as the simulation setup file. Otherwise, you have to modify the names accordingly.

RX Gear 16 is supported for only 1-lane and 2-lane configurations.

Data Settle Period determines when this module begin hunting SoT (start of Transmission) code (0xB8) on each lane after LP to HS transition is detected. The suggested value is around the middle of the allowable timing range defined by MIPI D-PHY specification so that this module does not work properly when SoT begins earlier than the timing obtained by this parameter. Therefore, reducing this value is one of the option you can try when this module is not working.

3.1.1. RX FIFO

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 in case of Hard D-PHY RX IP and resides before the word aligner in case of Soft D-PHY RX IP.

3.1.1.1. Hard D-PHY in Continuous Clock Mode

In this case, the minimum configuration of RX FIFO is recommended (LUT based, Depth = 16, *Type* Implementation = SINGLE, Packet Delay = 1, *Clock* Implementation = Dual Clock).

3.1.1.2. Soft D-PHY in Continuous Clock Mode

In this case, RX FIFO is not necessary and can be set to OFF when the read clock is the same as write clock. In this application, read clocks on RX channels are often unified and the read clock might come from different RX channels even though the frequency is same. In such cases, LUT based RX FIFO must be used with Depth = 16, *Type* Implementation = SINGLE, Packet Delay = 1, Clock Implementation = Dual Clock.

3.1.1.3. 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* Implementation = SINGLE, Packet Delay = 1, *Clock* Implementation = Dual Clock).

- Continuous byte clock < non-continuous byte clock

In this case, *Type* Implementation = 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 µs and the horizontal blanking is 4 µs, 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* Implementation = 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* Implementation = 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 this 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* Implementation = SINGLE with setting Packet Delay to the midpoint of FIFO depth when the tolerance is in ppm level. For example, assuming the clock tolerance is within +/- 500ppm for 2 lane Gear 8 RAW10 with 1920 horizontal pixels. The payload byte count is 1920 x 5/4 = 2400 so that each lane takes 1200 bytes. 1200 x 500 ppm = 0.6, which means small FIFO with middle point read delay (for example, FIFO depth = 16 with read delay of 8-byte clock cycles) works fine. Using LUT is preferable as long as FIFO depth is small since using EBR here might cause the shortage of EBR in line_buf modules. QUEUE can also be used as described above even though it requires more EBR.

In case you do not have detailed information regarding RX data (whether containing line 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. You may use *Type* Implementation = QUEUE with Number of Queue Entries = 4, even though it may require more EBR resources comparing to *Type* Implementation = SINGLE. You have to make sure that total number of EBR used in the device does not exceed 20.

3.2. line_buf

This module is instantiated for each RX channel. It contains a dual clock FIFO to store payload data of active video lines. Data write is based on RX byte clock and data read is based on TX byte clock for merging. Data widths of write and read sides depend on bus width of RX and TX side respectively, which means bus widths could be different. Several things have to be considered regarding FIFO operation.

3.2.1. Buffer Size

The required buffer size is obtained as shown below:

$$Buffer\ Size[CH\ \#n] \cong \frac{(N+n-1)}{N} * (Payload\ byte\ count\ per\ line)\ bytes,$$

where n is a channel number (0 to 4) and N is a number of RX channels (2 to 5).

The above equation shows that the required buffer size for the first channel is less than 1-line data, the one for the second channel equals to 1-line data and rests are between 1 line and 2 line data. Required sizes for third and later channels are substantially smaller than the above calculated values due to the existence of horizontal blanking period. Figure 3.2 shows an example of brief buffer write and read timings in case of 3-channel aggregation.

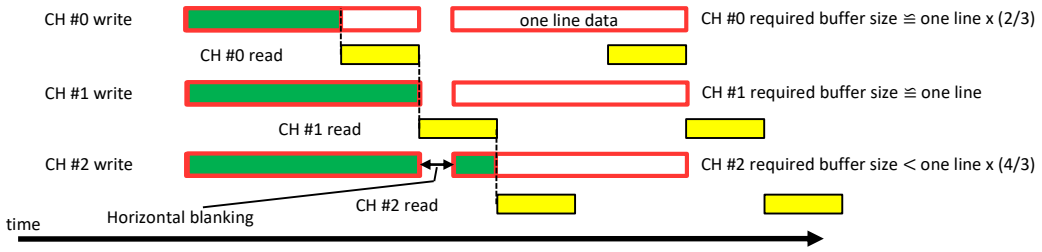

Figure 3.2. Buffer Size and Brief Write/Read Timings

On the other hand, the minimum count of EBR (Embedded Block RAM) to form the FIFO depends on the wider bus width between RX and TX, where bus width is derived by (lane count) x Gear. Table 3.1 shows the relationship between RX/TX bus width and unit count of EBR and payload byte count. The number of EBR per RX channel is a multiple of the unit count shown in Table 3.1.

Table 3.1. Unit EBR and Payload Byte Count of RX Buffer

RX Bus Width	TX Bus Width	Unit EBR Count	Unit Payload Byte Count
8	8	1	1024
	16	1	1024
	32	2	2048
	64	4	4096
16	8	1	1024
	16	1	1024
	32	2	2048
	64	4	4096
32	8*	2*	2048*
	16	2	2048
	32	2	2048
	64	4	4096

*__Note:__ This permutation does not exist.

Example: In case of 4 RX Channels with 2 lane, Gear 8, RAW10, 1920 pixels per line to TX with 4 lane, Gear 8:

Required RX Buffer Size [CH #0] = (4+0-1)/4 x (1920 x 10/8) = 1800 bytes,

Required RX Buffer Size [CH #1] = (4+1-1)/4 x (1920 x 10/8) = 2400 bytes,

Required RX Buffer Size [CH #2] = (4+2-1)/4 x (1920 x 10/8) = 3000 bytes,

Required RX Buffer Size [CH #3] = (4+3-1)/4 x (1920 x 10/8) = 3600 bytes.

Unit EBR count = 2, unit payload byte count = 2048.

Therefore, CH#0 requires 2 EBR and CH#1 – CH#3 require 4 EBR, which means 14 EBR are required in total.

Note that the total number of EBR cannot exceed 20 (assuming no EBR is used in rx_dphy modules). Since the buffer read timing might vary depending on TX side status, it might be safe to raise the buffer size in case that the required buffer size is close to the byte count you get from the above table. For example, if the pixel count is changed from 1920 to 2180 in the above example, the required RX Buffer size of CH#3 changes to 4087.5 bytes, which is closer to 4096 bytes using 4 EBR and it might be better to raise the buffer size. On the other hand, if the horizontal blanking is long enough, it is possible to handle larger payload byte count by the EBR, which can contain less number of byte than the value obtained by the equation shown above. For these reasons, the values specified by RX*BUF_SIZE* you can get from the above equation are reference only and you need to determine the actual values.

3.2.2. RX Buffer Offset Control

In case that the payload byte count does not match a multiple of RX bus width and/or TX bus width in the unit of byte, some considerations are necessary regarding RX buffer write and read. Figure 3.3 shows an example of (RX bus width) < (TX bus width). If WC (payload byte count) is not a multiple of TX bus width, RX side has to write some dummy data to the FIFO, otherwise the final data cannot read out from TX side. In this example, in the end of CH #0 data write transaction, bd #n is written to the FIFO with dummy data. Also, one extra dummy write cycle is necessary (RX0_POST_WRITE = 1) to make this data readable from TX side. The amount of the residual byte info (RX0_POST_OFFSET= 1) is used by merge_ctrl module to make a proper concatenation of the previous channel data and the current channel data. In the beginning of the write transaction of CH #1, the write data has to be shifted according to the offset info (RX1_PRE_OFFSET = 1). This makes the concatenation process easier on TX side by merge_ctrl. In the beginning of the write transaction of CH #2, one dummy write cycle is necessary (RX2_PRE_WRITE = 1) to make a necessary data shift since the offset in the end of previous channel write (RX1_POST_OFFSET = 2) is equal to the RX bus width. Figure 3.4 shows another example. In this case, RX*_PRE_OFFSET is always 0 and RX*_POST_OFFSET of the current channel always equals to RX*_PRE_WRITE of the next channel since RX bus width = 8.

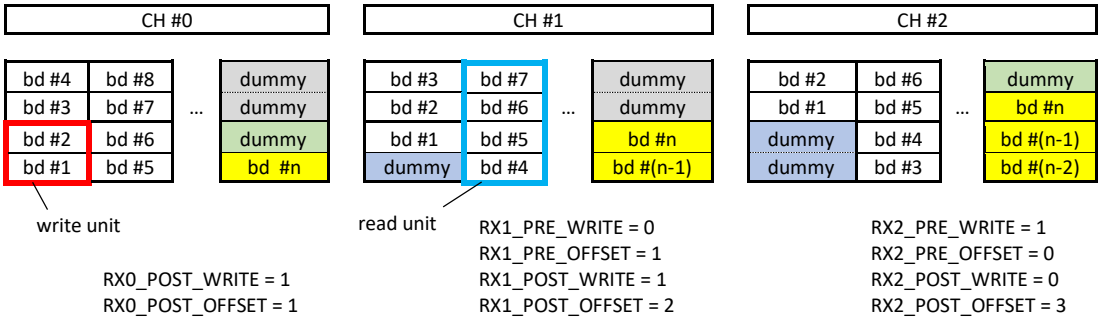

Figure 3.3. RX Buffer Offset Example #1 (RX bus width = 16, TX bus_width = 32, WC residual = 1)

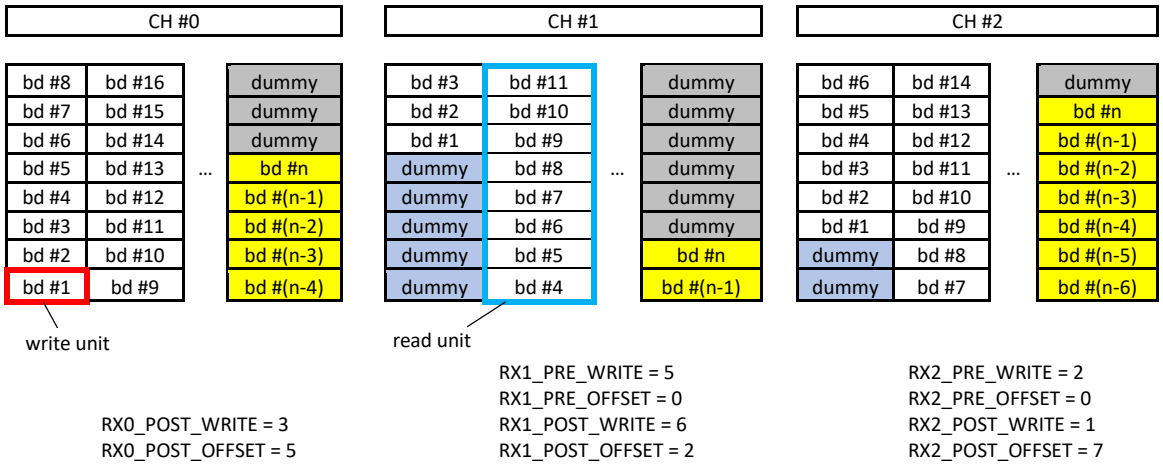

Figure 3.4. RX Buffer Offset Example #2 (RX bus width = 8, TX bus_width = 64, WC residual = 5)

Figure 3.5 shows an example of (RX bus width) > (TX bus width). In this case, TX side has to have an extra dummy reads cycle to match the amount of write data and read data by the parameter RX*_EXT_R = 1.

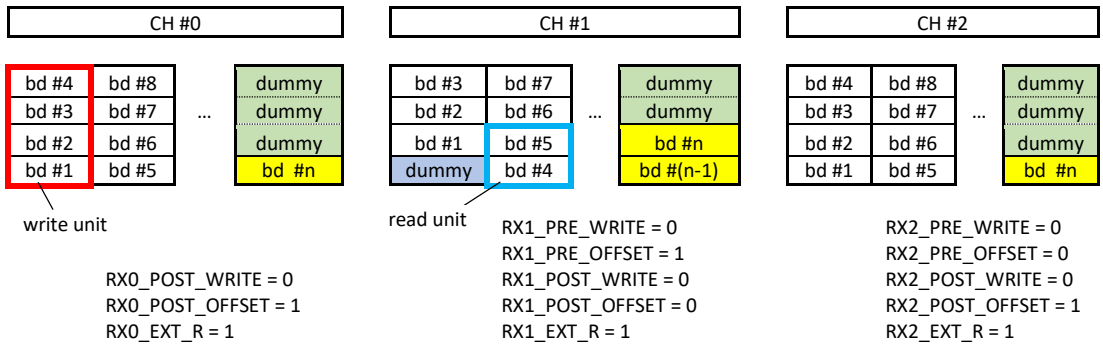

Figure 3.5. RX Buffer Offset Example #2 (RX bus width = 32, TX bus_width = 16, WC residual = 1)

These FIFO related parameter values are automatically obtained using the provided Excel sheet shown in Figure 1.4.

3.3. merge_ctrl

This module monitors the read ready flag of line_buf of RX channel 0, then reads RX Buffer data from channel 0 followed by channel 1, 2, … The read data are concatenated and sent to tx_dphy as single line data along with a new WC value. This module uses two FIFO related parameters; RX*_POST_OFFSET are used to make a necessary data shift when the new channel data are concatenated after finishing the previous channel data read. RX*_EXT_R is used to make an extra read in some cases when (RX bus width) > (TX bus width).

3.3.1. LP-HS Control in Non-Continuous Clock Mode

This module controls LP-HS-LP transition of tx_dphy module as shown in Figure 3.6 and Figure 3.7 with a following sequence in case of non-continuous clock mode:

1. Check tx_c2d_rdy = 1, then assert clk_hs_en and txfr_req (at least one tx_byte_clk cycle).
2. Clock lane goes into HS mode.
3. Data lane goes into HS mode.
4. Wait for txfr_en = 1, then assert tx_sp_en or tx_lp_en for one tx_byte_clk cycle.
5. In case of Long Packet data, assert tx_bd_en along with tx_bd two cycles after ld_pyld = 1.
6. HS data transmission.
7. After HS data transmission is done, txfr_en goes 0 and data lane goes into LP mode.
8. Clock lane goes into LP mode.
9. After all HS transaction ends, tx_c2d_rdy becomes 1, which means tx_dphy is ready to handle the next HS transaction.

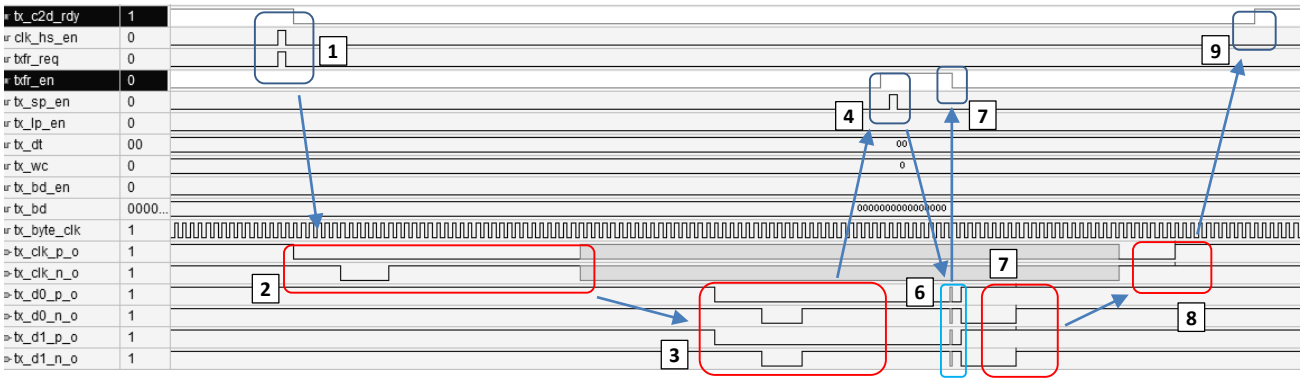

Figure 3.6. LP-HS-LP Transition in Non-Continuous Clock Mode (Short Packet)

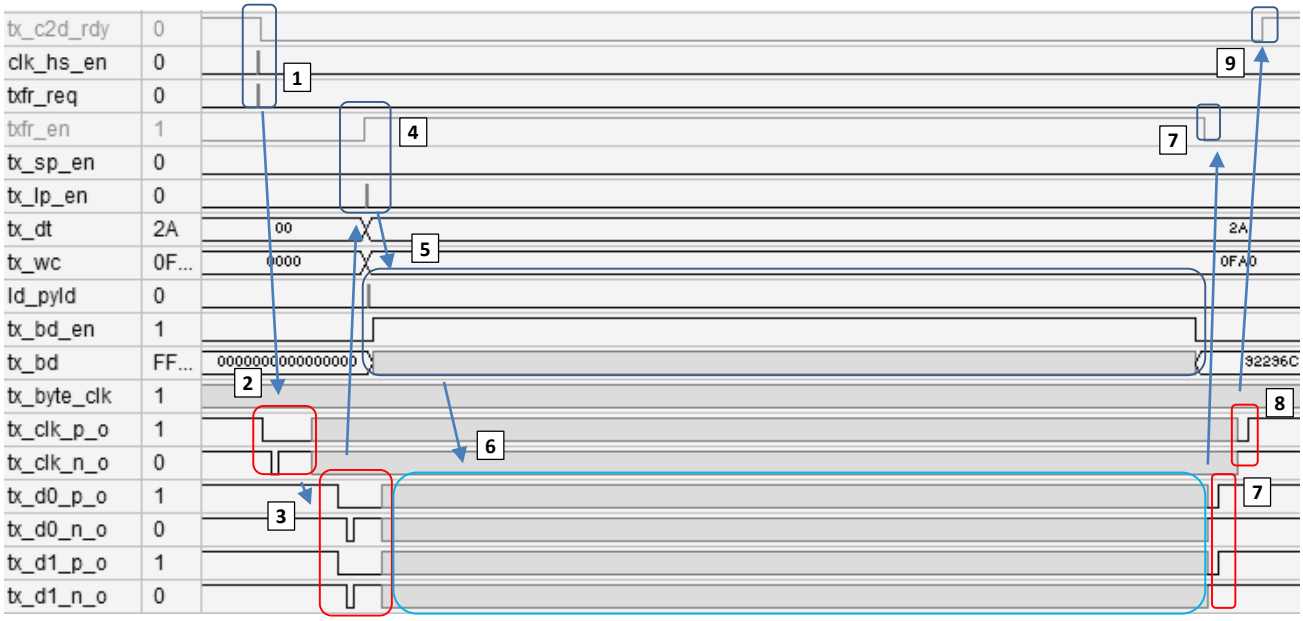

Figure 3.7. LP-HS-LP Transition in Non-Continuous Clock Mode (Long Packet)

In case of non-continuous clock mode, it is possible to keep the clock lane in HS mode during active video period. By enabling `define KEEP_HS in synthesis_directives.v, the clock lane goes into LP mode only between frame end short packet and frame start short packet as shown in Figure 3.8. clk_hs_en stays 1 from Frame Start to Frame End that makes clock lane in HS mode without going in to LP mode during horizontal blanking periods. This is useful when the horizontal blanking period is too short to make both clock lane and data lane go into LP mode, which requires more overhead time.

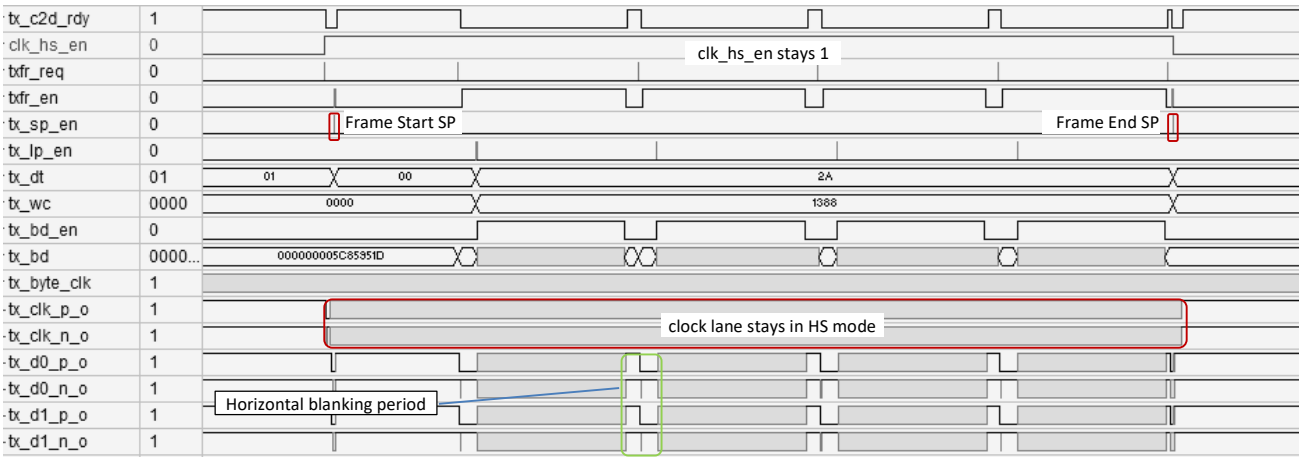

Figure 3.8. LP-HS-LP Transition in Non-Continuous Clock Mode with KEEP_HS

3.3.2. LP-HS Control in Continuous Clock Mode

Figure 3.9 shows LP-HS-LP transition in continuous clock mode for Short Packet transaction. The control scheme by this module is the same as non-continuous clock mode. The only difference from Figure 3.6 is clock lane stays in HS mode all the time and does not go into LP mode.

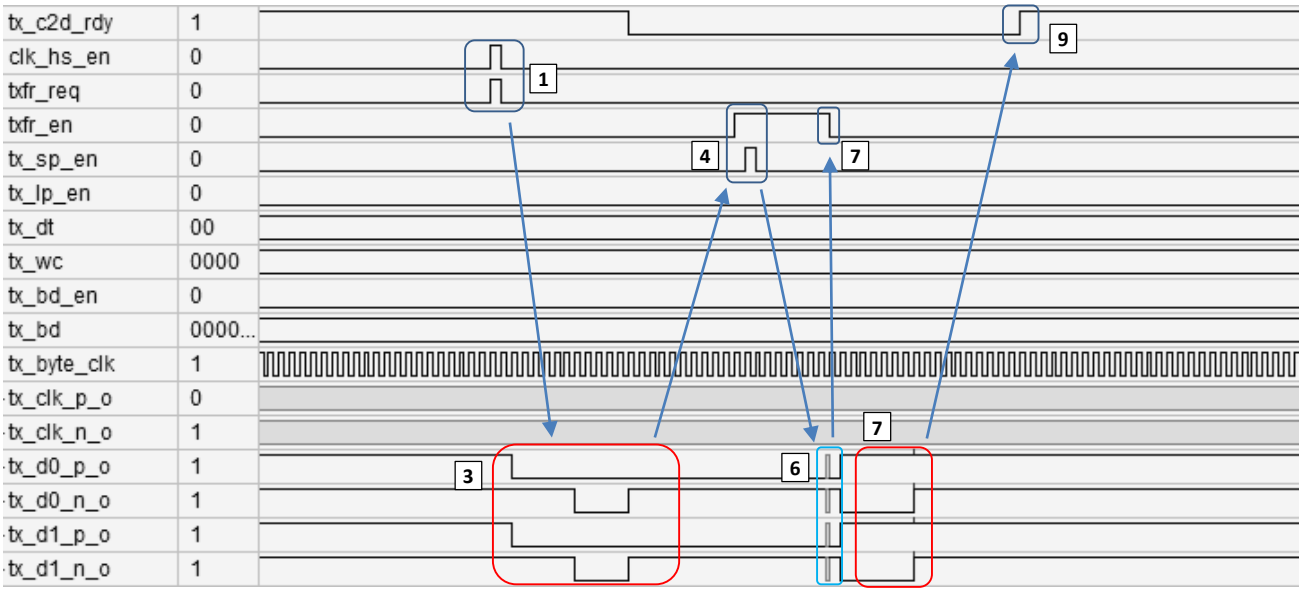

Figure 3.9. LP-HS-LP Transition in Continuous Clock Mode (Short Packet)

3.3.3. Line Buffer Read and Data Merge Operation

Figure 3.10 shows the global timing of the 5-channel merge operation. Only RX channel 0 signals are shown before Line Buffer module since timing are almost same in all RX channels. This module begins to read RX0 data when rx0_bufd_rdy is asserted followed by RX1, RX2… RX4 data reads for concatenation. The concatenated data are sent to tx_dphy as tx_bd. As described in line_buf section, rx0_bufd_rdy is asserted when ~4/5 of one line data are written to the line buffer FIFO in case of 5-channel aggregation. On the other hand, tx_dphy must be in HS mode before this module begins sending concatenated video data. To make this happen, the parameter *LP_TX_REQ_DLY* is provided. This value determines the delay time from rx0_lp_av_en assertion to txfr_req assertion in tx_byte_clk cycles. The value is automatically calculated by the design. This calculation considers the transition time from LP to HS mode for data lane in case of continuous clock mode and both clock lane and data lane in case of non-continuous clock mode. When KEEP_HS is defined in non-continuous clock mode, only data lane transition is considered.

You can set your own value by enabling *`define LP_TX_REQ_DLY xxx* in synthesis_directives.v to overwrite the automatically calculated value. The smaller value makes data lane (and clock lane in case of non-continuous clock mode) goes into HS mode earlier, but that leads to higher power consumption. The larger value makes power consumption lower, but video data is corrupted if the value is too big and timing overlap happens between LP to HS transition and RX0 buffer read.

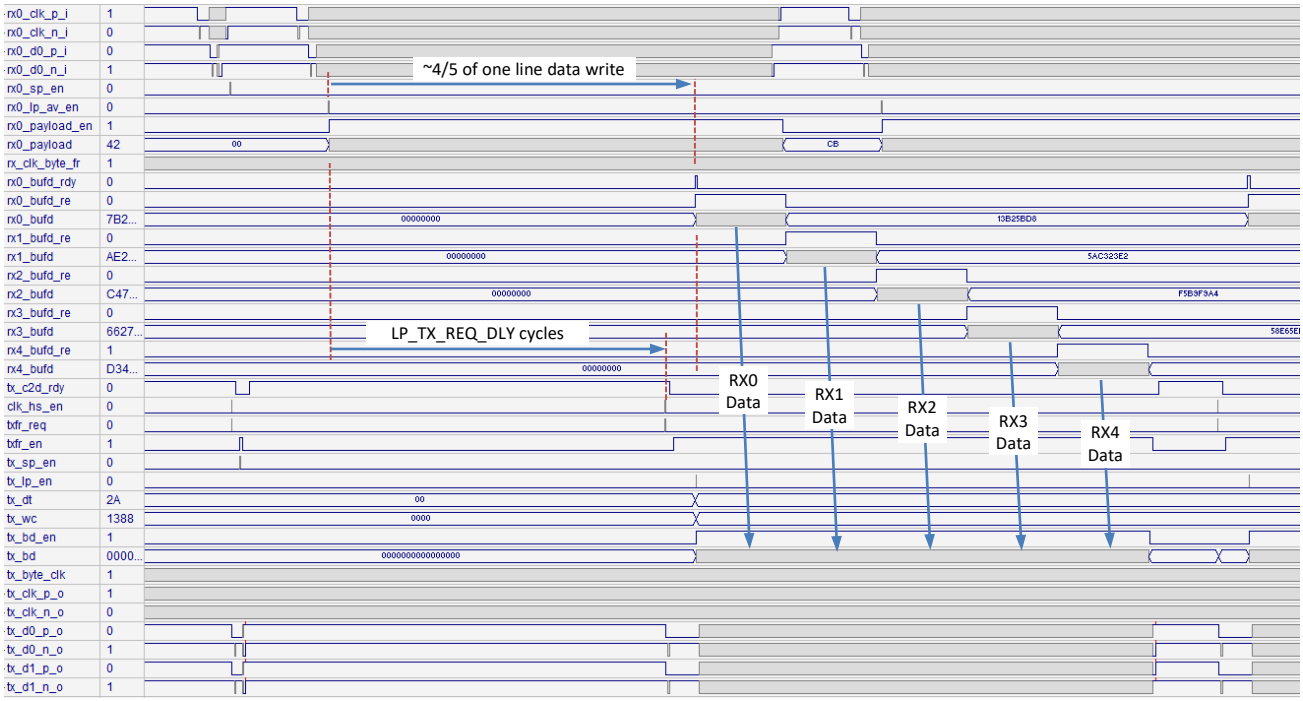

Figure 3.10. Line Buffer Read and Data Merge for 5-Channel Aggregation

Regarding Word Count (payload video data byte count per active line), this module takes the original WC value on RX channel 0 and multiplies it by the number of RX channels. The result is sent to tx_dphy as the final WC value of the aggregated video data.

Regarding VC (Virtual Channel ID), the value specified by *`define VC x* is used if defined, otherwise VC of RX channel 0 is sent to tx_dphy.

3.4. tx_dphy

You must create this module according to channel conditions, such as number of lanes, bandwidth, and others. Figure 3.11 shows an example IP interface setting in Clarity Designer for the CSI-2/DSI D-PHY Transmitter Submodule IP. Refer to CSI-2/DSI D-PHY Transmitter Submodule IP User Guide (FPGA-IPUG-02024) for details.

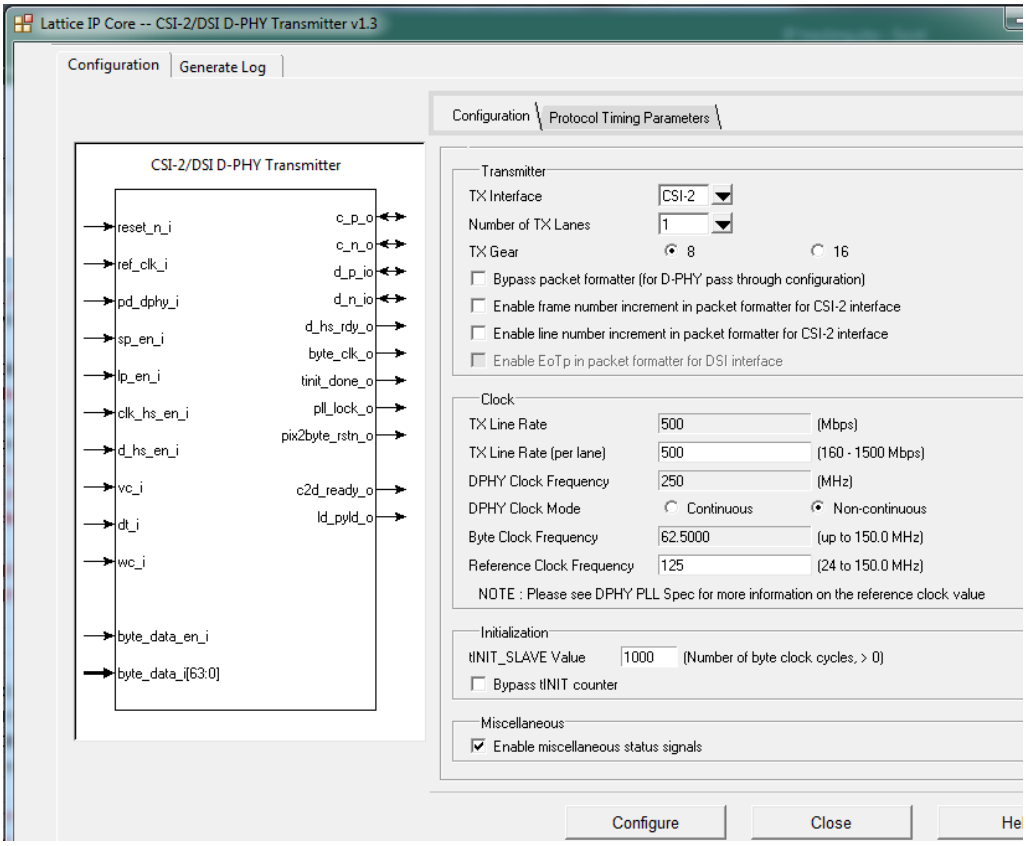

Figure 3.11. tx_dphy IP Creation in Clarity Designer

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

- TX Interface – Select CSI-2.
- Number of TX Lanes – Set according to channel configuration. Must match NUM_TX_LANE_* setting.
- TX Gear – Set according to TX Line Rate. In general, 16 must be selected when TX Line Rate is 900 or higher.
- Bypass packet formatter – Must be disabled (unchecked).
- Enable frame number – Set according to the preference.
- Enable line number – Must be disabled (unchecked).
- TX Line Rate – Set according to channel configuration. This value must be equal to (number of RX channel) x (RX channel bandwidth) / (number of TX lanes).
- D-PHY Clock Mode – Set according to channel configuration. Must select Continuous when the horizontal blanking period is short.
- Reference Clock Frequency – Set the appropriate value, which can be obtained from ref_clk_i pin, continuous rx0_byte_clk, or on-chip GPLL. This clock frequency must be in one of the following ranges:
  - 24-30 MHz
  - 48-60 MHz
  - 72-90 MHz
  - 96-150 MHz
- tINIT_SLAVE Value – 1000 (default) is recommended.

- Bypass tINIT counter – Disabled (unchecked) is recommended.
- Enable miscellaneous status signals – must be set to enabled (checked).
- Protocol Timing Parameters tab – Default values are recommended.

This module takes the byte data and outputs CSI-2 data after serialization in CSI-2 High Speed mode. It is recommended to set the design name to *tx* and module name to *tx_dphy* so that you do not need to modify the instance name of this IP in the top-level design as well as the simulation setup file. Otherwise, you need to modify the names accordingly.

General guideline of TX Gear setting is to set 8 when the lane bandwidth is less than 900 Mbps, which means TX byte clock could be ~112 MHz. If this causes timing violations in Static Timing Analysis (STA), TX Gear should be changed to 16.

You also should be aware of the relationship between the reference clock and DPHY clock. DPHY clock is generated by the internal PLL of TX D-PHY IP. Following is the equation to generate DPHY clock:

$$TX\_Line\_Rate\_per\_lane = \frac{1}{NI} * ref\_clk\_frequency * \frac{M}{NO}$$

where NI = 1, 2, 3, 4, or 5; M = 16, 17, …, or 255; NO = 1, 2, 4, or 8. The following restrictions also exist:

$$24MHz \leq \frac{1}{NI} * ref\_clk\_frequency \ \leq 30\ MHz,$$

$$640MHz \leq \frac{1}{NI} * ref\_clk\_frequency * M \leq 1500MHz$$

You must set the appropriate TX Line Rate (per lane) which can be obtained by the above equations applying the given reference clock frequency.

3.5. Clock Distribution

In this design, a single continuous byte clock is used in all RX channels to obtain the byte data from rx_dphy modules. If the original RX byte clocks among RX channels have slight frequency differences, RX FIFO inside RX D-PHY IP must be used to absorb the tolerance as described in RX FIFO section.

In case that non-continuous clock is used in RX channels, the continuous RX byte clock has to be obtained from the external clock source, either directly or indirectly. The following are possible candidates of the continuous clock:

- PLL outputs driven by the external reference clock
- Clock divider driven by the external reference clock
- The external clock itself when its frequency matches RX byte clock

The sample design (csi2_aggregation_ss.v) assumes that RX channels are in HS_LP mode. In that case, the continuous byte clock for RX channels and TX byte clock are generated by the on-chip GPLL taking the external clock as a reference clock. The code snippets are shown below. rx_clk_lp_ctrl (clock signal for LP and HS mode control module for clock lane) could be different from rx0_clk_byte_fr (continuous byte clock for RX channels), but recommended to be the same to save the primary clock tree resources since CrossLink can have only up to 8 primary clock trees. *int_gpll* is the name used in this top-level design. This name has to be changed if the different name is used.

```
////////////////////////////////////////////////////////////////////////////////
///// Reference Clock generation to TX D-PHY
///// Customer has to modify here according to the available clock for TX-DPHY
///// tx_refclk must be in the ranges of 24-30, 48-60, 72-90, or 96-150 MHz
///// Crosslink GPLL must be used to avoid frequency holes of Mixel PLL if
///// ref_clk_i nor rx0_clk_byte_fr is in these ranges.
////////////////////////////////////////////////////////////////////////////////
`ifdef RX_CLK_MODE_HS_ONLY
	`ifdef USE_GPLL
		int_gpll int_gpll (
			.CLKI   (rx0_clk_byte_fr),
			.CLKOP  (pll_clkop),
			.LOCK   (gpll_lock)
		);
		assign tx_ref_clk = pll_clkop;
	`else
		assign tx_ref_clk = rx0_clk_byte_fr;
		assign gpll_lock = 1'b1;
	`endif
`elsif RX_CLK_MODE_HS_LP
	`ifdef USE_GPLL
		int_gpll int_gpll (
			.CLKI   (ref_clk_i),
			.CLKOP  (pll_clkop),
			.CLKOS  (rx_clk_byte_fr),
			.LOCK   (gpll_lock)
		);
		assign tx_ref_clk = pll_clkop;
	`else  // very rare case!!!
		assign tx_ref_clk = ref_clk_i;
		assign rx_clk_byte_fr = ref_clk_i;
		assign gpll_lock = 1'b1;
	`endif
`endif
```

On the TX side, using continuous or non-continuous clock mode does not affect the number of necessary clock trees (always uses one clock tree). To feed a clock to TX D-PHY IP, the external clock is necessary if the continuous RX byte clock is not appropriate to generate the desired clock for TX D-PHY. The clock to TX D-PHY must be continuous and within one of the following frequency ranges:

- 24-30 MHz
- 48-60 MHz
- 72-90 MHz
- 96-150 MHz

4. Design and File Modifications

This RD is based on version 1.4 of the RX D-PHY IP and version 1.3 of the TX D-PHY IP. Due to the limitation of these 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

The current top-level file (csi2_aggreagation_ss.v) takes the primary GPLL clock to feed a clock to TX D-PHY when USE_GPLL is defined in synthesis_directives.v and takes the byte clock of RX channel 0 or external clock when USE_GPLL is not defined as shown in Clock Distribution section. This part must be modified, if the different clocking scheme is necessary.

In addition, instance names of RX/TX D-PHY (rx_dphy_h, rx_dphy_s, tx_dphy) have to be modified if you created these IP with different names.

5. Design Simulation

The script file (csi2_aggregation_ss_fsim.do) and testbench files are provided to run the functional simulation by Active HDL. You have to launch Active HDL from Diamond. If you follow the naming recommendations regarding design name and instance name when RX and TX D-PHY IPs are created by Clarity Designer, the following are the only changes required in the script file:

- Diamond installation directory path
- User project directory
- Comment out if GPLL is not in use
- Comment out or remove lines for Hard D-PHY IP if not in use

```
### Set Diamond installation directory ###
set diamond_dir C:/lscc/diamond/3.11_x64

### Set Customer's simulation directory ###
set sim_dir C:/Users/[redacted]/csi2_aggr_ss_RD/simulation/lifmd

cd $sim_dir
```

Diamond directory

own project directory

Figure 5.1. Script Modification #1

```
vlog -v2k5 -dbg ¥
+incdir+"./../../source/verilog/lifmd"+"./"+"./../../testbench/verilog" ./../../testbench/verilog/csi2_aggregation_ss_tb.v ¥
./../../source/verilog/lifmd/csi2_aggregation_ss.v ¥
./../../source/verilog/lifmd/line_buf.v ¥
./../../source/verilog/lifmd/beh/merge_ctrl_beh.v ¥
./../../rx/rx_dphy_s/dphy_rx_eval/rx_dphy_s/src/beh_rtl/capture_ctrl_beh.v ¥
./../../rx/rx_dphy_s/dphy_rx_eval/rx_dphy_s/src/beh_rtl/rx_global_ctrl_beh.v ¥
./../../rx/rx_dphy_s/dphy_rx_eval/rx_dphy_s/src/beh_rtl/dphy_rx_wrap_beh.v ¥
./../../int_gpll/int_gpll.v ¥

##### RX D-PHY IP (Hard) #####
vlog -v2k5 -dbg ¥
./../../rx/rx_dphy_h/rx_dphy_h.v ¥
./../../rx/rx_dphy_h/rx_dphy_h_dphy_rx.v ¥
./../../rx/rx_dphy_h/rx_dphy_h_rx_global_ctrl.v ¥
./../../rx/rx_dphy_h/rx_dphy_h_dphy_rx_wrap.v ¥
./../../rx/rx_dphy_h/rx_dphy_h_dphy_wrapper.v ¥
./../../rx/rx_dphy_h/rx_dphy_h_capture_ctrl.v ¥

##### RX D-PHY IP  (Soft) #####
vlog -v2k5 -dbg ¥
./../../rx/rx_dphy_s/rx_dphy_s.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_dphy_rx.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_rx_global_ctrl.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_dphy_rx_wrap.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_dphy_wrapper.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_capture_ctrl.v ¥
./../../rx/rx_dphy_s/rx_dphy_s_soft_dphy_rx.v ¥
./../../rx/rx_dphy_s/dphy_rx_eval/rx_dphy_s/src/beh_rtl/soft_dphy_rx_beh.v ¥

##### TX D-PHY IP #####
vlog -v2k5 -dbg ¥
./../../tx/tx_dphy/tx_dphy.v ¥
./../../tx/tx_dphy/tx_dphy_dphy_tx.v ¥
./../../tx/tx_dphy/tx_dphy_pkt_formatter.v ¥
./../../tx/tx_dphy/tx_dphy_synchronizer.v ¥
./../../tx/tx_dphy/tx_dphy_tx_global_operation.v ¥
./../../tx/tx_dphy/tx_dphy_tinit_count.v ¥
./../../tx/tx_dphy/tx_dphy_dci_wrapper.v ¥
./../../tx/tx_dphy/dphytx_eval/tx_dphy/src/beh_rtl/tinit_count_beh.v ¥
./../../tx/tx_dphy/dphytx_eval/tx_dphy/src/beh_rtl/pkt_formatter_beh.v ¥
./../../tx/tx_dphy/dphytx_eval/tx_dphy/src/beh_rtl/tx_global_operation_beh.v ¥

vsim +access +r top -L pmi_work -L ovi_lifmd
wave /top/dut/*
run -all
```

Comment out if GPLL is not in use

Comment out if Hard D-PHY RX is not in use

Figure 5.2. Script Modification #2

You need to modify simulation_directives.v according to your configuration (refer to Simulation Directives for details). By executing the script in Active HDL, compilation and simulation are executed automatically. The testbench takes all data comparison between the expected data and output data from the RD, including CRC data. In the beginning, the following statements should appear before the testbench starts feeding the CSI-2 data:

```
# KERNEL: 2459291800 tinit_done detected
# KERNEL:
# KERNEL:           2464291800 Activating dphy models
# KERNEL:
# KERNEL:           2464291900 DPHY CH 1 model activated
# KERNEL:
# KERNEL:           2464292200 DPHY CH 4 model activated
# KERNEL:
# KERNEL:           2464292700 DPHY CH 2 model activated
# KERNEL:
# KERNEL:           2464292700 DPHY CH 3 model activated
# KERNEL:          10464391900 DPHY CH 1 CLK : Driving HS-CLK-RQST
# KERNEL:          10464392200 DPHY CH 4 CLK : Driving HS-CLK-RQST
# KERNEL:          10464392700 DPHY CH 2 CLK : Driving HS-CLK-RQST
# KERNEL:          10464392700 DPHY CH 3 CLK : Driving HS-CLK-RQST
# KERNEL:          10464392900 DPHY CH 0 CLK : Driving HS-CLK-RQST
# KERNEL:          10469391900 DPHY CH 1 CLK : Driving HS-Prpr
# KERNEL:          10469392200 DPHY CH 4 CLK : Driving HS-Prpr
# KERNEL:          10469392700 DPHY CH 2 CLK : Driving HS-Prpr
# KERNEL:          10469392700 DPHY CH 3 CLK : Driving HS-Prpr
# KERNEL:          10469392900 DPHY CH 0 CLK : Driving HS-Prpr
# KERNEL:          10473191900 DPHY CH 1 CLK : Driving HS-Go
# KERNEL:          10473192200 DPHY CH 4 CLK : Driving HS-Go
# KERNEL:          10473192700 DPHY CH 2 CLK : Driving HS-Go
# KERNEL:          10473192700 DPHY CH 3 CLK : Driving HS-Go
# KERNEL:          10473192900 DPHY CH 0 CLK : Driving HS-Go
# KERNEL:          10499391900 DPHY CH 1 CLK : Driving HS-0/HS-1
# KERNEL:          10499392200 DPHY CH 4 CLK : Driving HS-0/HS-1
# KERNEL:          10499392700 DPHY CH 2 CLK : Driving HS-0/HS-1
# KERNEL:          10499392700 DPHY CH 3 CLK : Driving HS-0/HS-1
# KERNEL:          10499392900 DPHY CH 0 CLK : Driving HS-0/HS-1
# KERNEL:          10500723100 DPHY CH 1 DATA : Driving HS-RQST
# KERNEL:          10500724700 DPHY CH 2 DATA : Driving HS-RQST
# KERNEL:          10500726500 DPHY CH 0 DATA : Driving HS-RQST
# KERNEL:          10500727100 DPHY CH 3 DATA : Driving HS-RQST
# KERNEL:          10500727800 DPHY CH 4 DATA : Driving HS-RQST
# KERNEL:          10505723100 DPHY CH 1 DATA : Driving HS-Prpr
# KERNEL:          10505724700 DPHY CH 2 DATA : Driving HS-Prpr
# KERNEL:          10505726500 DPHY CH 0 DATA : Driving HS-Prpr
# KERNEL:          10505727100 DPHY CH 3 DATA : Driving HS-Prpr
# KERNEL:          10505727800 DPHY CH 4 DATA : Driving HS-Prpr
# KERNEL:          10510388700 DPHY CH 1 CLK : Driving HS-Go
# KERNEL:          10510390700 DPHY CH 2 CLK : Driving HS-Go
# KERNEL:          10510393300 DPHY CH 0 CLK : Driving HS-Go
# KERNEL:          10510394300 DPHY CH 3 CLK : Driving HS-Go
# KERNEL:          10510395600 DPHY CH 4 CLK : Driving HS-Go
# KERNEL:          10521910900 DPHY CH 3 CLK : Driving SYNC Data
# KERNEL:          10521910900 DPHY CH 3 Lane 0 : Driving with data = b8
# KERNEL:          10521937400 DPHY CH 0 CLK : Driving SYNC Data
# KERNEL:          10521937400 DPHY CH 0 Lane 0 : Driving with data = b8
…
```

The following messages show the payload data checking of concatenated data between RX channel 2 and 3:

```
# KERNEL: [28639633314][DPHY_CHK] Frame 2, Line 4, Byte Count 3003 - 3004, Payload Data
matches 72, 11 from CH#2
# KERNEL: [28640166594][DPHY_CHK] Frame 2, Line 4, Byte Count 3005 - 3006, Payload Data
matches f3, b1 from CH#2
# KERNEL: [28640699874][DPHY_CHK] Frame 2, Line 4, Byte Count 3007 - 3008, Payload Data
matches 58, 2c from CH#2
# KERNEL: [28641233154][DPHY_CHK] Frame 2, Line 4, Byte Count 3009 - 3010, Payload Data
matches 55, ed from CH#2/CH#3
# KERNEL: [28641766434][DPHY_CHK] Frame 2, Line 4, Byte Count 3011 - 3012, Payload Data
matches 92, 6a from CH#3
```

The following messages show an example of the end of the successful simulation:

```
# KERNEL: [29174566354][DPHY_CHK] Frame 2, Line 4, Byte Count 5009 - 5010, Payload Data
matches 52, ac from CH#4
# KERNEL: [29175099634][DPHY_CHK] Frame 2, Line 4, Byte Count 5011 - 5012, Payload Data
matches e5, f4 from CH#4
# KERNEL: [29175633314][DPHY_CHK] Frame 2, Line 4, Byte Count 5013 - 5014, Payload Data
matches 3c, 97 from CH#4
# KERNEL: [29176166594][DPHY_CHK] Frame 2, Line 4, Byte Count 5015, Payload Data matches e3
from CH#4
# KERNEL: [29176699874][DPHY_CHK] CRC = b601, matches
# KERNEL: [29184633214][DPHY_HS_LP_CHK] HS to LP11 Transition on D0 lane with HS-TRAIL period
= 79 ns
# KERNEL: [29198499694][DPHY_HS_LP_CHK] LP11 to LP01 Transition on D0 lane
# KERNEL: [29204899854][DPHY_HS_LP_CHK] LP01 to LP00 Transition on D0 lane with LP01 period =
64 ns
# KERNEL: [29210233054][DPHY_HS_LP_CHK] LP00 to HS00 Transition on D0 lane with LP00 period =
53 ns
# KERNEL: [29230199724][DPHY_HS_LP_CHK] HS00 to HS Transition on D0 lane with LP00+HS00
period = 252 ns
# KERNEL: [29231099634][DPHY_CHK] Short Packet detected : Data type = 01
# KERNEL: [29231633314][DPHY_CHK] Header1 = 00
# KERNEL: Check the Packet Header --- 01 00 00 07 matches Expected Short Packet Data
# KERNEL: [29239032974][DPHY_HS_LP_CHK] HS to LP11 Transition on D0 lane with HS-TRAIL period
= 74 ns
# KERNEL:          32663488000 DPHY CH 1 CLK CONT : Driving CLK-Trail
# KERNEL:          32669488000 DPHY CH 1 CLK CONT : Driving CLK-Stop
# KERNEL:          32671129500 DPHY CH 0 CLK CONT : Driving CLK-Trail
# KERNEL:          32677129500 DPHY CH 0 CLK CONT : Driving CLK-Stop
# KERNEL:          32684535600 DPHY CH 2 CLK CONT : Driving CLK-Trail
# KERNEL:          32689964400 DPHY CH 3 CLK CONT : Driving CLK-Trail
# KERNEL:          32690535600 DPHY CH 2 CLK CONT : Driving CLK-Stop
# KERNEL:          32695964400 DPHY CH 3 CLK CONT : Driving CLK-Stop
# KERNEL:          32697045200 DPHY CH 4 CLK CONT : Driving CLK-Trail
# KERNEL:          32703025200 TEST END
# KERNEL:
# KERNEL: ##### 2 Frames x 4 Lines x 5015 Payload Bytes comparison succeeded!!! #####
# KERNEL:
# KERNEL: ### Simulation Completed ###
```

You should set small values in NUM_LINES and NUM_FRAMES directives in simulation_directives.v file, especially in the first simulation trial to minimize the simulation time. On the other hand, it is very important to set the actual value to RX_WC directives in synthesis_directives.v since this directly affects the design parameters. Also, you need to set the actual (or close to the actual) value of RX_DPHY_LPS_GAP in simulation_directives.v, especially RX buffer FIFO size is close to the value obtained by the equation in line_buf section, so that FIFO overflow could be detected when the margin is not long enough.

When the simulation does not advance as expected, the most common issue is related to PLL setting.

```
[case #1] Reference clock to D-PHY TX IP is not in the required range
```

As described in tx_dphy and Clock Distribution sections, the reference clock to tx_dphy must be in the range of 24 - 30 MHz or its multiple. If the clock is out of the range, the following messages appear:

```
# KERNEL:  ERROR : Fref: Reference Clock is out of range
# KERNEL:  ERROR : Fff: Clock after input divider is out of range
# KERNEL:  ERROR : Fvcout: Clock before output divider is out of range
# KERNEL:  ERROR : PLL won't work
```

Assuming the external clock drives directly this reference clock and the frequency is 30 MHz, the clock setting of this external clock can be *`define REF_CLK_PERIOD 33333* in simulation_directives.v, but this fails since the clock period of 33.333 ns leads the clock frequency slightly higher than 30.000 MHz. PLL model of D-PHY TX IP cannot accept this and PLL does not work. Therefore, the value must be 33334 or larger in this case.

```
[case #2] PLL's VCO frequency is out of range
```

As described in tx_dphy section, VCO output frequency of TX D-PHY PLL must be 640 – 1500 MHz. If this violation happens, the following messages appear and PLL does not work:

```
# KERNEL:  ERROR : Fvcout: Clock before output divider is out of range
# KERNEL:  ERROR : PLL won't work
```

Assuming the RX lane bandwidth is 600 Mbps with Gear 8 in continuous clock mode and TX lane bandwidth is 1500 Mbps. The continuous byte clock is 75 MHz and this can drive the reference clock of tx_dphy. RX0 D-PHY clock frequency of 300 MHz (= 600/2) can be set by *`define RX0_DPHY_CLK_PERIOD 3333* in simulation_directives.v. But, this fails since the clock period of 3.333 ns leads the clock frequency slightly higher than 75.000 MHz of RX byte clock and PLL's VCO frequency above 1500 MHz. PLL model of D-PHY TX IP cannot accept this and PLL does not work. Therefore, the value must be 3334 or larger in this case.

Figure 5.3 shows an example of 5-channel aggregation. Signals timings before line_buf are almost same among all RX channels and only RX channel 0 signals are shown. Both RX and TX channels uses non-continuous clock mode and `define KEEP_HS is enabled. TX clock lane goes into LP mode only during the vertical blanking period.

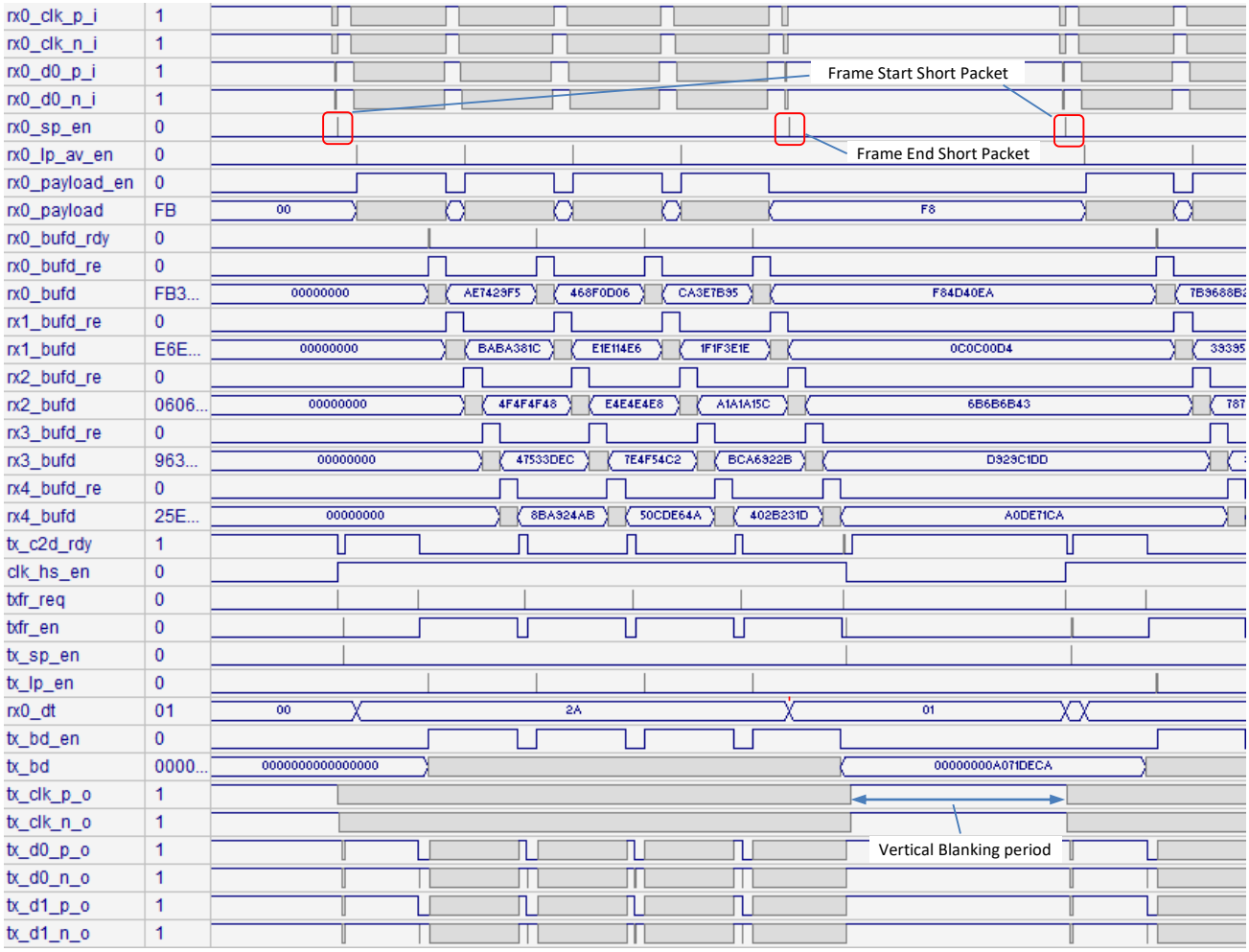

Figure 5.3. Functional Simulation Example

6. Known Limitations

The following are the limitations of this reference design:

- RX Gear 16 with 4-lane configuration is not supported.
- Total number of RX clock and data lanes by RX D-PHY Soft IP cannot exceed 15.
- This design does not support the data type, which changes the amount of video data line by line, like YUV420 8-bit, YUV420 10-bit.

7. Design Package and Project Setup

N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation Reference Design for CrossLink is available on www.latticesemi.com. Figure 7.1 shows the directory structure. The design is targeted for LIF_MD6000-6KMG80I. synthesis_directives.v and simulation_directives.v are set to configure five RX channels as an example shown below:

- RX CH #0 : 1 lanes with Hard D-PHY with Gear 8 in non-continuous clock mode
- RX CH #1 : 1 lanes with Soft D-PHY with Gear 8 in non-continuous clock mode
- RX CH #2 - #4 : same as RX CH #1
- TX : 2 lanes with Gear 16

You can modify the directives for own configuration.

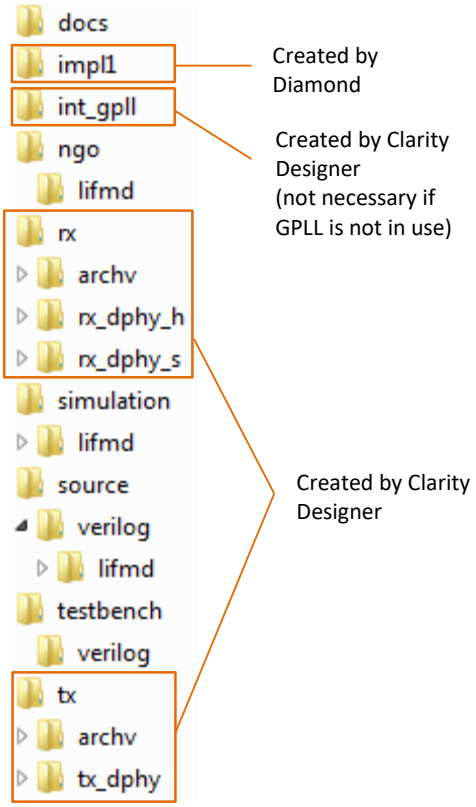

Figure 7.1. Directory Structure

Folders *rx* and *tx* are expected to be added under your project directory when Clarity Designer creates RX and TX D-PHY IPs. The *ngo\lifmd* folder contains mapped netlists of *merge_ctrl* with all possible configurations designated by suffixes. Following shows explanation of the suffixes:

- _C* : Number of RX channels; 2 – 5
- _TL*G** : * Number of TX lanes; 1, 2, or 4, ** TX Gear; 8 or 16

The blackbox files (*_bb.v) are also contained in the *ngo\lifmd* folder. You can include corresponding _bb.v files as design files in Diamond according to own configuration. Alternatively, it is possible to include all _bb.v files as long as *csi2_aggregation_ss* is specified as the top-level design. Figure 7.2 shows design files used in the Diamond project. Clarity Designer creates three .sbx files. In this example, all 24 _bb.v files are included in the project. By specifying csi2_aggregation_ss as a top-level design, all unnecessary files are ignored. GPLL design file must be added if necessary.

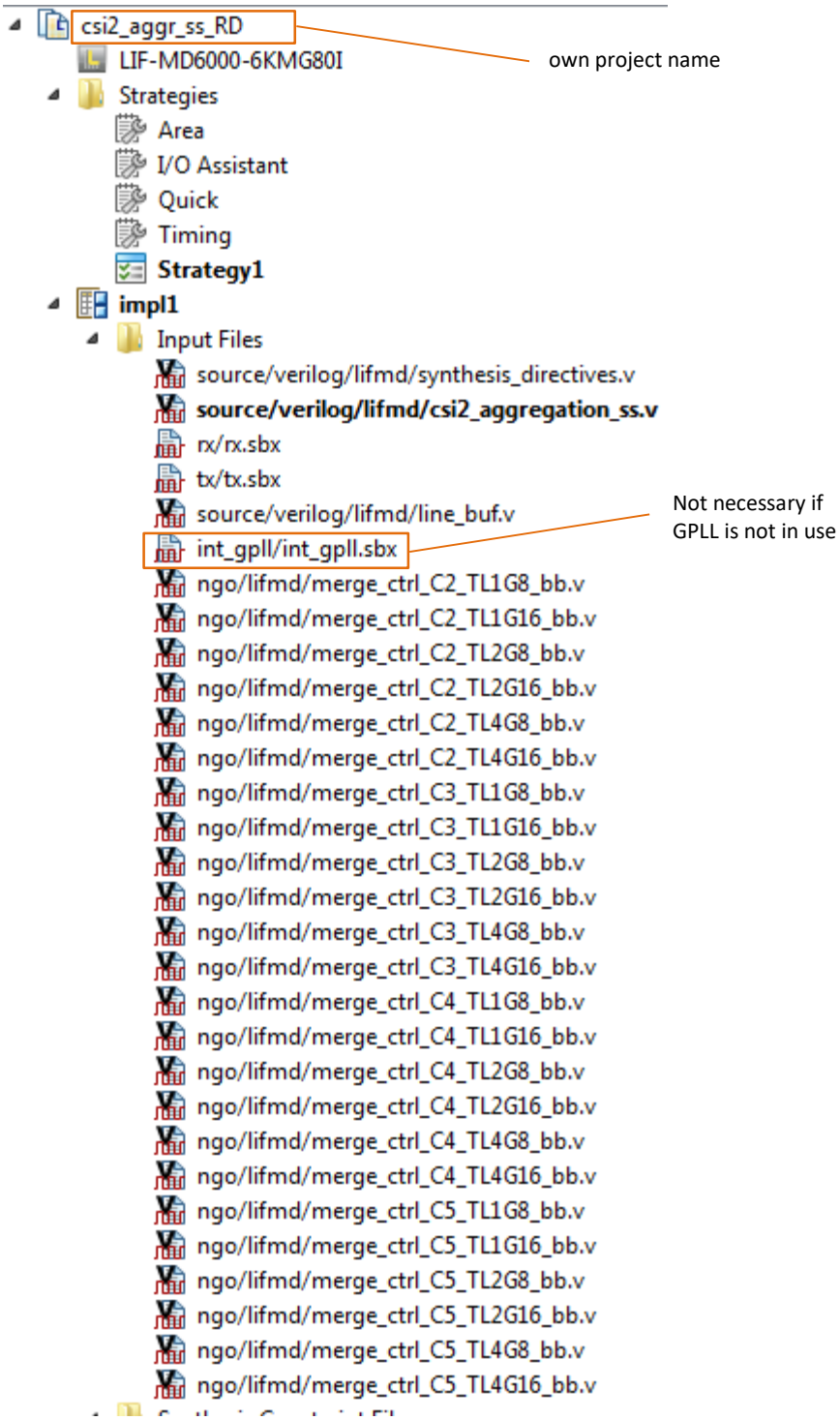

Figure 7.2. Project Files

You need to set the path for .ngo files in the translation stage. This can be done by editing strategy file (*Strategy1*) shown in Figure 7.2. Select *Translate Design* and set Macro Search Path as shown in Figure 7.3.

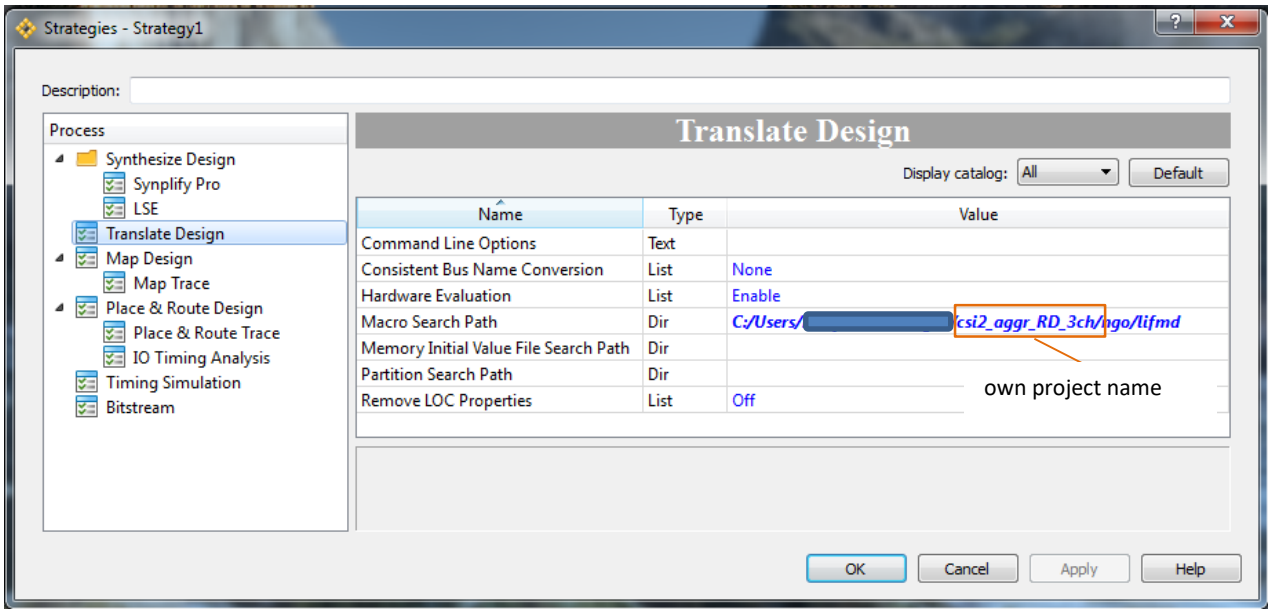

Figure 7.3. Path Setting for .ngo Files

8. Resource Utilization

Resource utilization depends on the configurations. Table 8.1 shows resource utilization examples under certain configurations. Actual usage may vary.

Table 8.1. Resource Utilization Examples

Configuration	LUT %	FF %	EBR	I/O
4 RX Channels with 2 lanes Gear 8 in continuous clock mode, TX 4 lanes Gear 16, RAW8 x 1000	90	52	16	37
2 RX Channels with 2 lanes Gear 8 in continuous clock mode, TX 1 lane Gear 8, RAW8 x 1000	41	25	4	17
5 RX Channels with 2 lane Gear 8 in continuous clock mode, TX 4 lanes Gear 16, RAW10 x 1920, Word Aligner off on ch #0	94	58	20	41
4 RX Channels with 4 lane Gear 8 in continuous clock mode, TX 4 lanes Gear 16, RAW12 x 1920, Word Aligner off on ch #0	**112**	65	20	52

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 Receiver Submodule IP User Guide (FPGA-IPUG-02025)
- CSI-2/DSI D-PHY Transmitter Submodule IP User Guide (FPGA-IPUG-02024)

For more information on the CrossLink FPGA device, visit http://www.latticesemi.com/Products/FPGAandCPLD/CrossLink.

For more information on the CrossLinkPlus FPGA device, visit http://www.latticesemi.com/Products/FPGAandCPLD/CrossLinkPlusPlus.

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

Technical Support Assistance

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

Revision History

Revision 1.0, April 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com