



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

Reference Design

FPGA-RD-02253-1.3

September 2024

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 REDUNDANCIES, 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

Contents	3
Acronyms in This Document	5
1. Introduction	6
1.1. Supported Device, IP, and Software	6
1.2. Features	6
1.3. Block Diagram	6
1.4. RX/TX Permutations and Parameter Calculation	9
2. Parameters and Port List	12
2.1. Synthesis Directives	12
2.2. Simulation Directives	15
2.3. Top-Level I/O	16
3. Design and Module Description	19
3.1. rx0*_unit	19
3.1.1. rx_dphy	19
3.1.2. RX FIFO	21
3.2. line_buf	22
3.2.1. Buffer Size	22
3.2.2. RX Buffer Offset Control	23
3.3. merge_ctrl	25
3.3.1. LP-HS Control in Non-Continuous Clock Mode	25
3.3.2. LP-HS Control in Continuous Clock Mode	26
3.3.3. Line Buffer Read and Data Merge Operation	27
3.4. tx_dphy	28
3.5. Clock Distribution	31
4. Design and File Modifications	32
4.1. Top-Level RTL	32
5. Design Simulation	33
6. Design Debug on Hardware	38
6.1. Top-Level	38
6.2. MIPI Clock Lane Transactions in D-PHY RX IP	38
6.3. MIPI Data Lane Transactions in D-PHY RX IP	40
6.4. Inter-Channel Synchronization and Line Buffer Control	43
6.5. D-PHY TX Control	44
7. Known Limitations	45
8. Design Package and Project Setup	46
9. Resource Utilization	48
References	49
Technical Support Assistance	50
Revision History	51

Figures

Figure 1.1. N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation Block Diagram	7
Figure 1.2. Clocking Scheme Example in Continuous Clock Mode	8
Figure 1.3. Clocking Scheme Example in Non-Continuous Clock Mode	8
Figure 1.4. Bandwidth and Parameter Calculator	11
Figure 3.1. rx_dphy IP Creation in Lattice Radiant #1	19
Figure 3.2. rx_dphy IP Creation in Lattice Radiant #2	20
Figure 3.3. Buffer Size and Brief Write/Read Timings	22
Figure 3.4. RX Buffer Offset Example #1 (RX bus_width = 16, TX bus_width = 32, WC residual = 1)	24
Figure 3.5. RX Buffer Offset Example #2 (RX bus_width = 8, TX bus_width = 64, WC residual = 5)	24
Figure 3.6. RX Buffer Offset Example #2 (RX bus_width = 32, TX bus_width = 16, WC residual = 1)	24
Figure 3.7. LP-HS-LP Transition in Non-Continuous Clock Mode (Short Packet)	25
Figure 3.8. LP-HS-LP Transition in Non-Continuous Clock Mode (Long Packet)	25
Figure 3.9. LP-HS-LP Transition in Non-Continuous Clock Mode with KEEP_HS	26
Figure 3.10. LP-HS-LP Transition in Continuous Clock Mode (Short Packet)	26
Figure 3.11. Line Buffer Read and Data Merge for 5-Channel Aggregation	27
Figure 3.12. tx_dphy IP Creation in Lattice Radiant	29
Figure 5.1. Script Modification #1	33
Figure 5.2. Script Modification #2	33
Figure 5.3. Functional Simulation Example	37
Figure 6.1. MIPI Clock Lane Transition and Expected Debug Signal Behaviors	38
Figure 6.2. MIPI Clock Lane Transition	38
Figure 6.3. MIPI Clock Lane Transition from LP to HS Mode	40
Figure 6.4. Debug Signals of D-PHY RX IP	41
Figure 6.5. MIPI Data Lane Signal Transitions	41
Figure 6.6. Short Packet and Long Packet Transaction Example with Debug Signals	42
Figure 6.7. Line Buffer Control	43
Figure 6.8. D-PHY TX IP Timing Parameters	44
Figure 8.1. Directory Structure	46
Figure 8.2. Project Files	47

Tables

Table 1.1. Supported Device and IP	6
Table 1.2. RX and TX Permutation	9
Table 2.1. Synthesis Directives	12
Table 2.2. Simulation Directives	15
Table 2.3. CSI-2 Side-by-Side Aggregation Top-Level I/O	16
Table 3.1. Unit EBR and Payload Byte Count of RX Buffer	23
Table 6.1. Timing Parameters for Clock Lane Signal Transitions	39
Table 6.2. Timing Parameters for Data Lane Signal Transitions	42
Table 9.1. Resource Utilization Examples	48

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AP	Application Processor
CH	Channel
CSI-2	Camera Serial Interface 2
DDR	Double Data Rate
DSI	Display Serial Interface
EBR	Embedded Block RAM
ECC	Error Correction Code
FIFO	First In First Out
HS	High Speed
ID	Identification Data
IP	Intellectual Property
LP	Low Power
LUT	Lookup Table
MIPI	Mobile Industry Processor Interface
PLL	Phase Locked Loop
GPLL	General Purpose PLL
RAM	Random Access Memory
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 CertusPro™-NX devices offers up to eight-channel aggregation. Multiple channel image data are concatenated horizontally line by line. CertusPro-NX 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, IP, and Software

This reference design supports the following devices with IP and software versions shown below:

Table 1.1. Supported Device and IP

Device Family	Part Number	Compatible IP	Software Version
CertusPro-NX	LFPCNX-100	D-PHY Receiver IP version 1.3.0 D-PHY Transmitter IP version 1.5.0	Lattice Radiant Software version 3.0 and above

1.2. Features

- Two to eight Soft RX channels can be aggregated.
- One D-PHY Soft IP is used on TX channel.
- 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.
- Maximum RX bandwidth is 1.5 Gb/s per lane.
- Number of TX lanes can be one, two, or four.
- Maximum TX bandwidth is 1.5 Gb/s per lane.
- Non-continuous clock mode on RX channels is possible as long as the continuous clock is obtained directly or indirectly from the external reference clock.

1.3. Block Diagram

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

In case of non-continuous mode on RX channels, an external clock is necessary to obtain a continuous clock. An on-chip GPLL may have to be used to create an appropriate clock since TX D-PHY PLL has an input clock frequency requirement of between 24 MHz and 200 MHz.

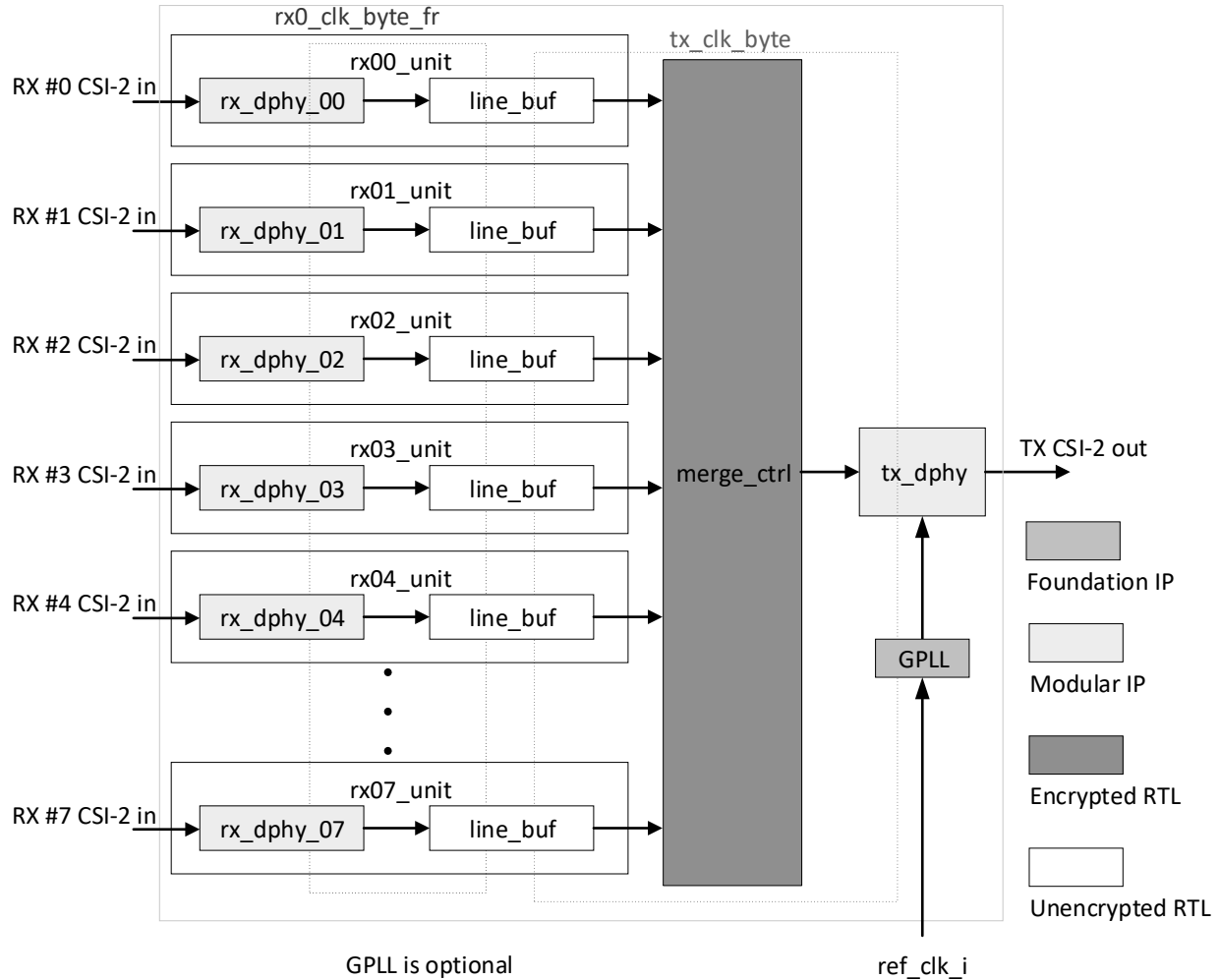


Figure 1.1. N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation Block Diagram

Figure 1.2 shows an example of clocking scheme in continuous clock mode. In this case, RX Soft D-PHY IP used on channel #0 and continuous byte clock generated by rx_dphy_00 (rx0_clk_byte_hs) is shared by all RX channels. Also requires sync clock and the internal oscillator is used in this example.

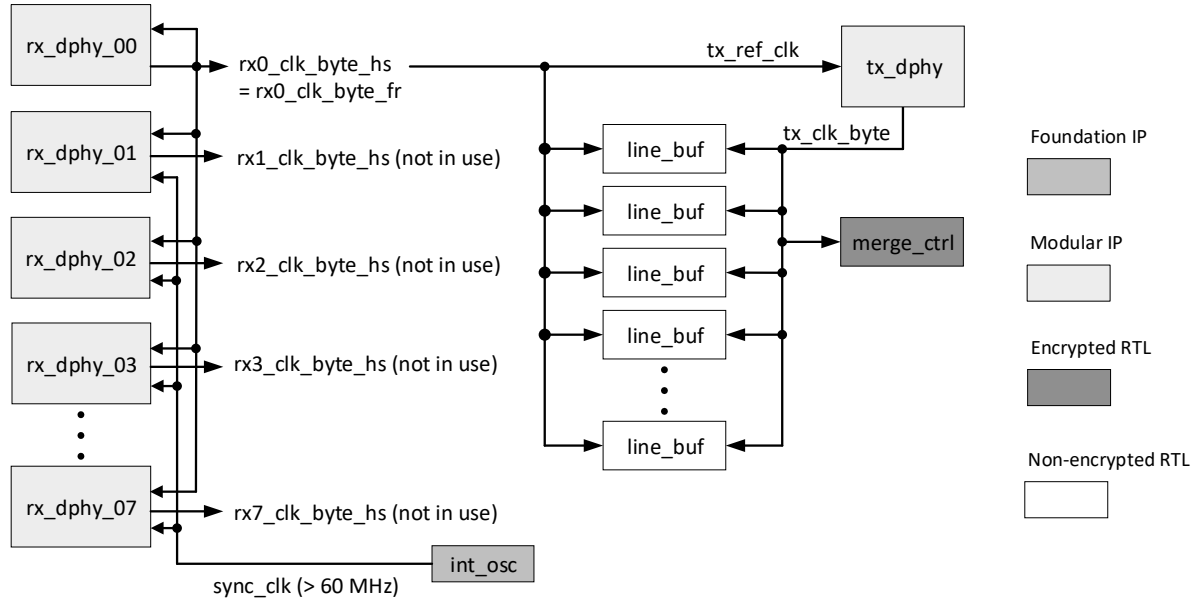


Figure 1.2. Clocking Scheme Example in Continuous Clock Mode

Figure 1.3 shows an example of clocking scheme in non-continuous clock mode. In this case, Rx Soft D-PHY IP used on channel #0. The external clock (ref_clk_i) is mandatory to feed both continuous byte clock and tx_ref_clk. GPLL can be bypassed if ref_clk_i is equal to rx0_byte_clk_fr. sync_clk can be obtained by GPLL instead of the oscillator if GPLL generates a clock above 60 MHz.

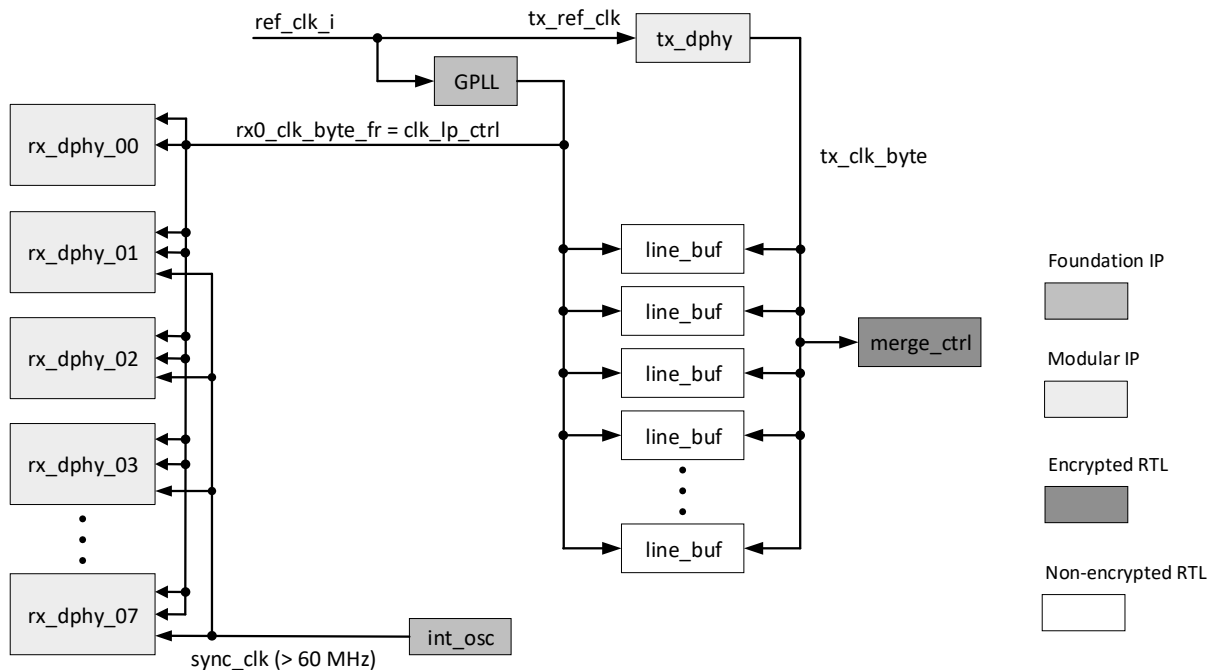


Figure 1.3. Clocking Scheme Example in Non-Continuous Clock Mode

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 (80 Mb/s), 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 (Mb/s)	Number of TX Lanes	TX Lane Bandwidth/RX Lane Bandwidth
2	1	80 – 1250	1	2
		80 – 1500	2	1
		80 – 1500	4	0.5
	2	80 – 625	1	4
		80 – 1250	2	2
		80 – 1500	4	1
	4	80 – 312.5	1	8
		80 – 625	2	4
		80 – 1250	4	2
3	1	80 – 833	1	3
		80 – 1500	2	1.5
		80 – 1500	4	0.75
	2	80 – 416	1	6
		80 – 833	2	3
		80 – 1500	4	1.5
	4	80 – 208*	1*	12*
		80 – 416	2	6
		80 – 833	4	3
4	1	80 – 625	1	4
		80 – 1250	2	2
		80 – 1500	4	1
	2	80 – 312.5	1	8
		80 – 625	2	4
		80 – 1250	4	2
	4	80 ≤ 156.25*	1*	16*
		80 – 312.5	2	8
		80 – 625	4	4
5	1	80 – 500	1	5
		80 – 1000	2	2.5
		80 – 1500	4	1.25
	2	80 – 250*	1*	10*
		80 – 500	2	5
		80 – 1000	4	2.5
	4	80 ≤ 125*	1*	20*
		80 – 250*	2*	10*
		80 – 500*	4	5
6	1	80 – 416	1	6
		80 – 833	2	3
		80 – 1500	4	1.5
	2	80 – 208*	1*	12*
		80 – 416	2	6
		80 – 833	4	3

Number of RX Channels	Number of RX Lanes	RX Lane Bandwidth (Mb/s)	Number of TX Lanes	TX Lane Bandwidth/RX Lane Bandwidth
	4	80 ≤ 104*	1*	24*
		80 – 208*	2*	12*
		80 – 416	4	6
7	1	80 – 357	1	7
		80 – 714	2	3.5
		80 – 1428	4	1.75
	2	80 – 178*	1*	14*
		80 – 357	2	7
		80 – 714	4	3.5
	4	80 ≤ 89*	1*	28*
		80 – 178*	2*	14*
		80 – 357	4	7
8	1	80 – 312.5	1	8
		80 – 625	2	4
		80 – 1250	4	2
	2	80 ≤ 156.25*	1*	16*
		80 – 312.5	2	8
		80 – 625	4	4
	4	80 ≤ 78.125*	1*	32*
		80 ≤ 156.25*	2*	16*
		80 – 312.5	4	8

***Note:** These permutations are not supported.

The Excel sheet (csi2_aggregation_ss_param_LFCPNX.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.

N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation RD with Certus Pro Parameter Calculator

RX Data Type	RAW10	
Horizontal Pixel Count	1920	
Number of RX Channels	8	
Number of RX Lanes	2	
RX Gear	8	
RX Line Rate (per lane)	200	Mbps up to 1500
RX DPHY Clock Frequency	100	MHz
RX Byte Clock Frequency	25	MHz
Number of TX Lanes	4	
TX Gear	8	
TX Line Rate (per lane)	800	Mbps up to 2500
TX DPHY Clock Frequency	400	MHz
TX Byte Clock Frequency	100	MHz

RX Data Type of YUV420_8 is Legacy YUV420 8-bit format.

Word Count		2400
Word Count must be an integer		

CH#0	RX0_POST_WRITE	0
	RX0_POST_OFFSET	0
	RX0_EXT_R	0

CH#1	RX1_PRE_WRITE	0
	RX1_PRE_OFFSET	0
	RX1_POST_WRITE	0
	RX1_POST_OFFSET	0
CH#1	RX1_EXT_R	0

CH#2	RX2_PRE_WRITE	0
	RX2_PRE_OFFSET	0
	RX2_POST_WRITE	0
	RX2_POST_OFFSET	0
CH#2	RX2_EXT_R	0

CH#3	RX3_PRE_WRITE	0
	RX3_PRE_OFFSET	0
	RX3_POST_WRITE	0
	RX3_POST_OFFSET	0
CH#3	RX3_EXT_R	0

CH#4	RX4_PRE_WRITE	0
	RX4_PRE_OFFSET	0
	RX4_POST_WRITE	0
	RX4_POST_OFFSET	0
CH#4	RX4_EXT_R	0

always 0

CH#5	RX5_PRE_WRITE	0
	RX5_PRE_OFFSET	0
	RX5_POST_WRITE	0
	RX5_POST_OFFSET	0
CH#5	RX5_EXT_R	0

CH#6	RX6_PRE_WRITE	0
	RX6_PRE_OFFSET	0
	RX6_POST_WRITE	0
	RX6_POST_OFFSET	0
CH#6	RX6_EXT_R	0

CH#7	RX7_PRE_WRITE	0
	RX7_PRE_OFFSET	0
	RX7_POST_WRITE	0
	RX7_POST_OFFSET	0
CH#7	RX7_EXT_R	0

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 Radiant® 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 six directives must be defined.
	RAW10	
	RAW12	
	RGB888	
	YUV422_8	
	YUV422_10	
RX Channel Count	NUM_RX_CH_2	Number of RX channels aggregated. Only one of these seven directives must be defined.
	NUM_RX_CH_3	
	NUM_RX_CH_4	
	NUM_RX_CH_5	
	NUM_RX_CH_6	
	NUM_RX_CH_7	
	NUM_RX_CH_8	
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	RX D-PHY Clock Gear always set as 8.
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	
RX Buffer Size ²	RX0_BUF_SIZE_*	RX channel 0 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX1_BUF_SIZE_*	RX channel 1 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX2_BUF_SIZE_*	RX channel 2 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX3_BUF_SIZE_*	RX channel 3 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX4_BUF_SIZE_*	RX channel 4 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.

Category	Directive	Remarks
	RX5_BUF_SIZE_*	RX channel 5 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX6_BUF_SIZE_*	RX channel 6 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
	RX7_BUF_SIZE_*	RX channel 7 FIFO buffer size in byte. * must be 2048, 4096, 8192, or 16384.
RX Buffer Pre-Write ³	RX1_PRE_WRITE 0	Pre-write cycle count to RX channel 1 FIFO buffer. Value must be 0 – 7.
	RX2_PRE_WRITE 0	Pre-write cycle count to RX channel 2 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX3_PRE_WRITE 0	Pre-write cycle count to RX channel 3 FIFO buffer. Value must be 0 – 7.
	RX4_PRE_WRITE 0	Pre-write cycle count to RX channel 4 FIFO buffer. Value must be 0 or 4.
	RX5_PRE_WRITE 0	Pre-write cycle count to RX channel 5 FIFO buffer. Value must be 0 – 7.
	RX6_PRE_WRITE 0	Pre-write cycle count to RX channel 6 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX7_PRE_WRITE 0	Pre-write cycle count to RX channel 7 FIFO buffer. Value must be 0 – 7.
RX Buffer Pre-Offset ³	RX1_PRE_OFFSET 0	Pre-offset byte count to RX channel 1 FIFO buffer. Value must be 0 – 3.
	RX2_PRE_OFFSET 0	Pre-offset byte count to RX channel 2 FIFO buffer. Value must be 0 or 2.
	RX3_PRE_OFFSET 0	Pre-offset byte count to RX channel 3 FIFO buffer. Value must be 0 – 3.
	RX5_PRE_OFFSET 0	Pre-offset byte count to RX channel 5 FIFO buffer. Value must be 0 – 3.
	RX6_PRE_OFFSET 0	Pre-offset byte count to RX channel 6 FIFO buffer. Value must be 0 or 2.
	RX7_PRE_OFFSET 0	Pre-offset byte count to RX channel 7 FIFO buffer. Value must be 0 – 3.
RX Buffer Post-Write ³	RX0_POST_WRITE 0	Post-write cycle count to RX channel 0 FIFO buffer. Value must be 0 – 7.
	RX1_POST_WRITE 0	Post-write cycle count to RX channel 1 FIFO buffer. Value must be 0 – 7.
	RX2_POST_WRITE 0	Post-write cycle count to RX channel 2 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX3_POST_WRITE 0	Post-write cycle count to RX channel 3 FIFO buffer. Value must be 0 or 4.
	RX4_POST_WRITE 0	Post-write cycle count to RX channel 4 FIFO buffer. Value must be 0 - 7.
	RX5_POST_WRITE 0	Post-write cycle count to RX channel 5 FIFO buffer. Value must be 0 or 4.
	RX6_POST_WRITE 0	Post-write cycle count to RX channel 6 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX7_POST_WRITE 0	Post-write cycle count to RX channel 7 FIFO buffer. Value must be 0 or 4.
RX Buffer Post-Offset ³	RX0_POST_OFFSET 0	Post-offset byte count to RX channel 0 FIFO buffer. Value must be 0 – 7.
	RX1_POST_OFFSET 0	Post-offset byte count to RX channel 1 FIFO buffer. Value must be 0, 2, 4, or 6.
	RX2_POST_OFFSET 0	Post-offset byte count to RX channel 2 FIFO buffer. Value must be 0 - 7.
	RX3_POST_OFFSET 0	Post-offset byte count to RX channel 3 FIFO buffer. Value must be 0 or 4.
	RX3_POST_OFFSET 0	Post-offset byte count to RX channel 4 FIFO buffer. Value must be 0 – 7.
	RX5_POST_OFFSET 0	Post-offset byte count to RX channel 5 FIFO buffer. Value must be 0 or 4.
	RX6_POST_OFFSET 0	Post-offset byte count to RX channel 6 FIFO buffer. Value must be 0 – 7.
RX Buffer Extra Read ³	RX0_EXT_R 0	Extra Read Flag for RX channel 0 FIFO Buffer. Value must be 0 or 1.
	RX1_EXT_R 0	Extra Read Flag for RX channel 1 FIFO Buffer. Value must be 0 or 1.
	RX2_EXT_R 0	Extra Read Flag for RX channel 2 FIFO Buffer. Value must be 0 or 1.
	RX3_EXT_R 0	Extra Read Flag for RX channel 3 FIFO Buffer. Value must be 0 or 1.
	RX4_EXT_R 0	Extra Read Flag for RX channel 4 FIFO Buffer. Value must be 0 or 1.
	RX5_EXT_R 0	Extra Read Flag for RX channel 5 FIFO Buffer. Value must be 0 or 1.
	RX6_EXT_R 0	Extra Read Flag for RX channel 6 FIFO Buffer. Value must be 0 or 1.

Category	Directive	Remarks
	RX7_EXT_R 0	Extra Read Flag for RX channel 7 FIFO Buffer. Value must be 0 or 1.
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 always set as 8.
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 Mb/s. 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. 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
	RX5_DPHY_CLK_PERIOD {value}	RX Channel 5 DPHY clock period in ps
	RX6_DPHY_CLK_PERIOD {value}	RX Channel 6 DPHY clock period in ps
	RX7_DPHY_CLK_PERIOD {value}	RX Channel 7 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
	CH5_DELAY {value}	Initial delay to activate RX Channel 5 in ps
	CH6_DELAY {value}	Initial delay to activate RX Channel 6 in ps
	CH7_DELAY {value}	Initial delay to activate RX Channel 7 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)

Port Name	Direction	Description
rx3_d2_p_i	I	Positive differential RX Ch3 D-PHY input data 2 (in case of 4-lane configuration)
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)
rx5_clk_p_i	I	Positive differential RX Ch5 D-PHY input clock
rx5_clk_n_i	I	Negative differential RX Ch5 D-PHY input clock
rx5_d0_p_i	I	Positive differential RX Ch5 D-PHY input data 0
rx5_d0_n_i	I	Negative differential RX Ch5 D-PHY input data 0
rx5_d1_p_i	I	Positive differential RX Ch5 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx5_d1_n_i	I	Negative differential RX Ch5 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx5_d2_p_i	I	Positive differential RX Ch5 D-PHY input data 2 (in case of 4-lane configuration)
rx5_d2_n_i	I	Negative differential RX Ch5 D-PHY input data 2 (in case of 4-lane configuration)
rx5_d3_p_i	I	Positive differential RX Ch5 D-PHY input data 3 (in case of 4-lane configuration)
rx5_d3_n_i	I	Negative differential RX Ch5 D-PHY input data 3 (in case of 4-lane configuration)
rx6_clk_p_i	I	Positive differential RX Ch6 D-PHY input clock
rx6_clk_n_i	I	Negative differential RX Ch6 D-PHY input clock
rx6_d0_p_i	I	Positive differential RX Ch6 D-PHY input data 0
rx6_d0_n_i	I	Negative differential RX Ch6 D-PHY input data 0
rx6_d1_p_i	I	Positive differential RX Ch6 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx6_d1_n_i	I	Negative differential RX Ch6 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx6_d2_p_i	I	Positive differential RX Ch6 D-PHY input data 2 (in case of 4-lane configuration)
rx6_d2_n_i	I	Negative differential RX Ch6 D-PHY input data 2 (in case of 4-lane configuration)
rx6_d3_p_i	I	Positive differential RX Ch6 D-PHY input data 3 (in case of 4-lane configuration)
rx6_d3_n_i	I	Negative differential RX Ch6 D-PHY input data 3 (in case of 4-lane configuration)
rx7_clk_p_i	I	Positive differential RX Ch7 D-PHY input clock
rx7_clk_n_i	I	Negative differential RX Ch7 D-PHY input clock
rx7_d0_p_i	I	Positive differential RX Ch7 D-PHY input data 0
rx7_d0_n_i	I	Negative differential RX Ch7 D-PHY input data 0
rx7_d1_p_i	I	Positive differential RX Ch7 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx7_d1_n_i	I	Negative differential RX Ch7 D-PHY input data 1 (in case of 2-lane or 4-lane configuration)
rx7_d2_p_i	I	Positive differential RX Ch7 D-PHY input data 2 (in case of 4-lane configuration)
rx7_d2_n_i	I	Negative differential RX Ch7 D-PHY input data 2 (in case of 4-lane configuration)
rx7_d3_p_i	I	Positive differential RX Ch7 D-PHY input data 3 (in case of 4-lane configuration)
rx7_d3_n_i	I	Negative differential RX Ch7 D-PHY input data 3 (in case of 4-lane configuration)

Port Name	Direction	Description
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_aggr_ss_LFCPNX.v) consists of the following modules:

- rx0*_unit (* is 0 – 7)
- rx_dphy (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. rx0*_unit

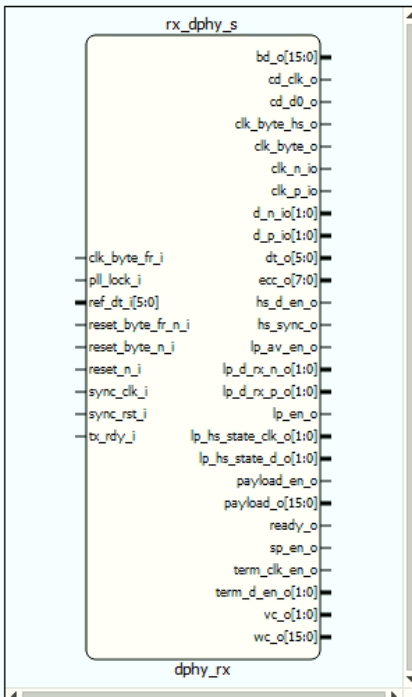
This module is instantiated for each RX channel as a wrapper module to include rx_dphy (rx_dphy_s) and line_buf modules.

3.1.1. rx_dphy

Since all RX channels must be in the same configuration, a single rx_dphy_s module for Soft D-PHY RX IP can cover all RX channels. Figure 3.1 and Figure 3.2 show an example of IP interface settings in Radiant for the CSI-2/DSI D-PHY Receiver Submodule IP. Refer to [CSI/DSI D-PHY Rx IP Core User Guide \(FPGA-IPUG-02081\)](#) for details.

Configure Component from IP dphy_rx Version 1.3.0
Set the following parameters to configure this component.

Diagram rx_dphy_s



Configure IP

Property	Value
Receiver	
RX Interface	CSI-2
D-PHY RX IP	Soft D-PHY
Number of D-PHY Data Lanes	2
RX Gear	8
CIL Bypass	☒
Enable LMMI Interface	☐
Enable AXI4-Stream Interface	☐
Enable Deskew Calibration Detection	☐
Clock	
RX Line Rate (Mbps) [80 - 1500]	200
D-PHY Clock Frequency (MHz) [40 - 1250]	100
D-PHY Clock Mode	Continuous
Byte Clock Frequency (MHz) [10 - 187.5]	25
Sync Clock Frequency (MHz) [60 - 200]	75
Timing Parameter	
Customize Data Settle Cycle	☐
Data Settle Cycle [0 - 4]	2
Customize CIL Data Settle	☐
CIL Data Settle [7 - 15]	10

No DRC issues are found.

Generate

Cancel

Figure 3.1. rx_dphy IP Creation in Lattice Radiant #1

Configure Component from IP dphy_rx Version 1.3.0
Set the following parameters to configure this component.

Diagram rx_dphy_s

Configure IP

Property	Value
CIL Data Settle [7 - 15]	10
Customize CIL Clock Settle	☐
CIL Clock Settle [6 - 23]	11
Output	
Enable Packet Parser	☒
Miscellaneous	
Enable Miscellaneous Status Signals	☒
Soft IP Implementation Settings	
Enable Lane Aligner Module	☒
RX_FIFO	
RX_FIFO Enable	☐
Implementation	EBR
Depth	128
Number of Queue Entries	4
Type	PINGPONG
Packet Delay [0 - 16384]	8
Counter Width [2 - 4]	4
Clock Mode	DC
Misc Signals	☐

No DRC issues are found.

Generate Cancel

Figure 3.2. rx_dphy IP Creation in Lattice Radiant #2

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

- RX Interface – Select CSI-2.
- DPHY RX IP – Always Soft D-PHY is selected for rx_dphy_s.
- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Always 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.
- DPHY Clock Mode – Select Continuous or Non-continuous. Must match RX_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- Customize Data Settle Cycle – Select Disabled (unchecked) for initial HW test and select Enabled (checked) for simulation.
- Data Settle Cycle – Select the smallest value for simulation.
- Sync Clock Frequency – Enter the value of sync_clk in case of Soft D-PHY.
- Enable Packet Parser – Select Enabled.
- Enable Miscellaneous Status Signals – Select Enabled.
- Enable Lane Aligner Module – Select Enabled in case of Soft D-PHY in 2-lane and 4-lane configurations.

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 module names to `rx_dphy_s` so that you do not need to modify the instance names of these IPs in `rx0*_unit` as well as the simulation setup file. Otherwise, you have to modify the names accordingly.

Data Settle Cycle determines when this module begins 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 options you can try when this module is not working.

3.1.2. 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. Also, it is useful in this design even in continuous clock mode since the continuous byte clock can be shared among all RX channels by utilizing this FIFO. It resides before the word aligner in case of Soft D-PHY RX IP.

RX_FIFO Enable and Misc Signals options must be enabled in all cases.

3.1.2.1. 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 = 4, Clock Implementation = Dual Clock.

3.1.2.2. 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 frequency = non-continuous byte clock frequency
In this case, the minimum configuration of RX FIFO is recommended (LUT based, Depth = 16, *Type Implementation* = SINGLE, Packet Delay = 4, *Clock Implementation* = Dual Clock).
- Continuous byte clock frequency < non-continuous byte clock frequency
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 μ s and the horizontal blanking is 4 μ s, 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 frequency > non-continuous byte clock frequency

There are two options in this case:

- Use *Type Implementation* = SINGLE with large Packet Delay
Set the Depth large enough to store the enough amount of 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 $1920 \times 5/4 = 2400$ so that each lane takes 1200 bytes. $1200 \times 500 \text{ ppm} = 0.6$, 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 84.

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:

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

where n is a channel number (0 to 7) and N is a number of RX channels (2 to 8).

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 3.3 shows an example of brief buffer write and read timings in case of 3-channel aggregation.

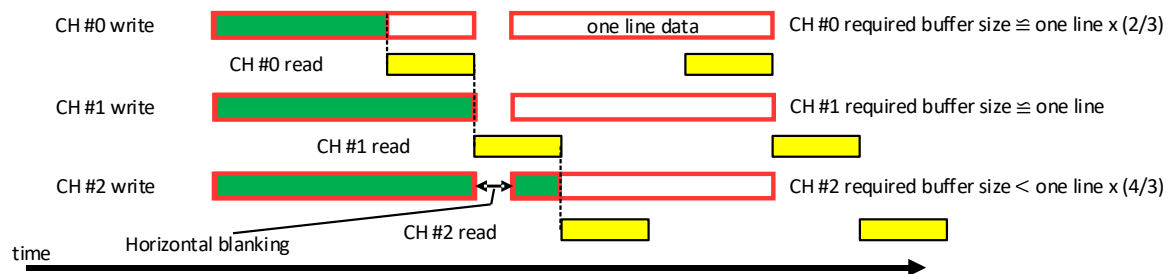


Figure 3.3. Buffer Size and Brief Write/Read Timings

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	2048
	16	1	2048
	32	1	2048
	64	2	4096
16	8	1	2048
	16	1	2048
	32	1	2048
	64	2	4096
32	8	1	2048
	16	1	2048
	32	1	2048
	64	2	4096

Example: In case of 4 RX Channels with 2 lanes, Gear 8, RAW10, 3840 pixels per line to TX with 4 lanes, Gear 8:

Required RX Buffer Size [CH #0] = $(4+0-1)/4 \times (3840 \times 10/8) = 3600$ bytes,

Required RX Buffer Size [CH #1] = $(4+1-1)/4 \times (3840 \times 10/8) = 4800$ bytes,

Required RX Buffer Size [CH #2] = $(4+2-1)/4 \times (3840 \times 10/8) = 6000$ bytes,

Required RX Buffer Size [CH #3] = $(4+3-1)/4 \times (3840 \times 10/8) = 7200$ bytes.

Unit EBR count = 1, 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 in the device cannot exceed 84. 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 Table 3.1. 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 2 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 a smaller 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 equations 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 3.4 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 3.5 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.

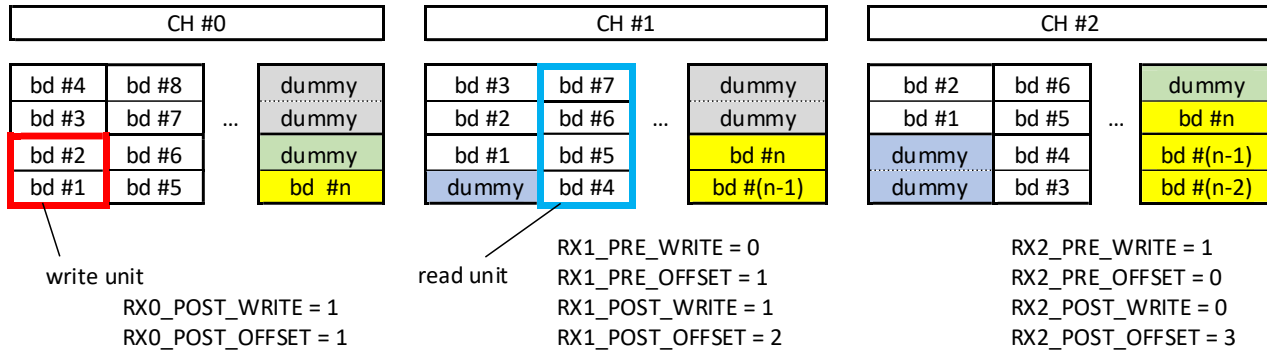


Figure 3.4. RX Buffer Offset Example #1 (RX bus_width = 16, TX bus_width = 32, WC residual = 1)

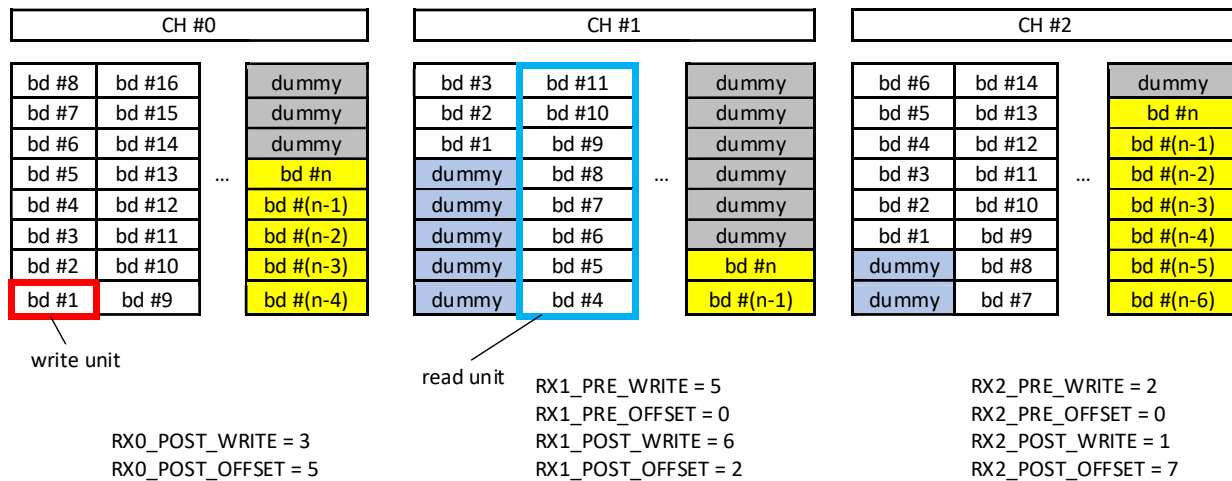


Figure 3.5. RX Buffer Offset Example #2 (RX bus width = 8, TX bus_width = 64, WC residual = 5)

Figure 3.6 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.

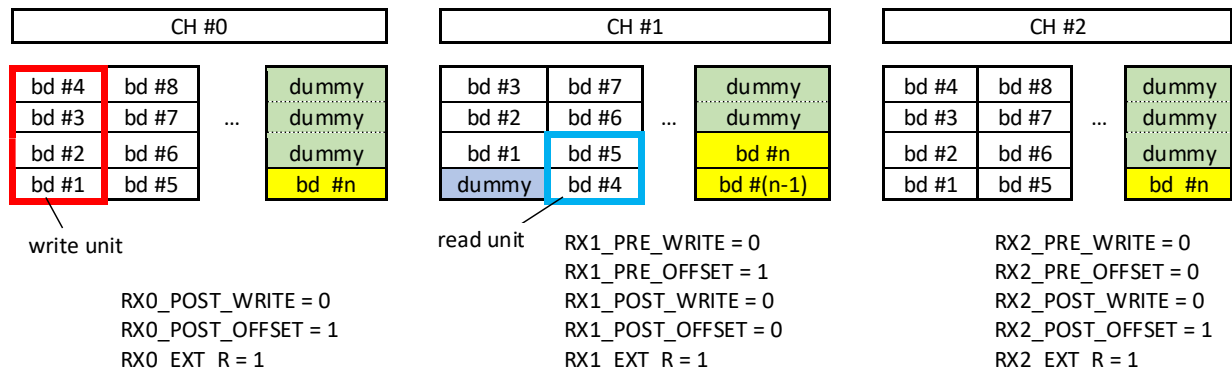


Figure 3.6. RX Buffer Offset Example #2 (RX bus_width = 32, TX bus_width = 16, WC residual = 1)

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 3.7 and Figure 3.8 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_clk_byte 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_clk_byte 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.

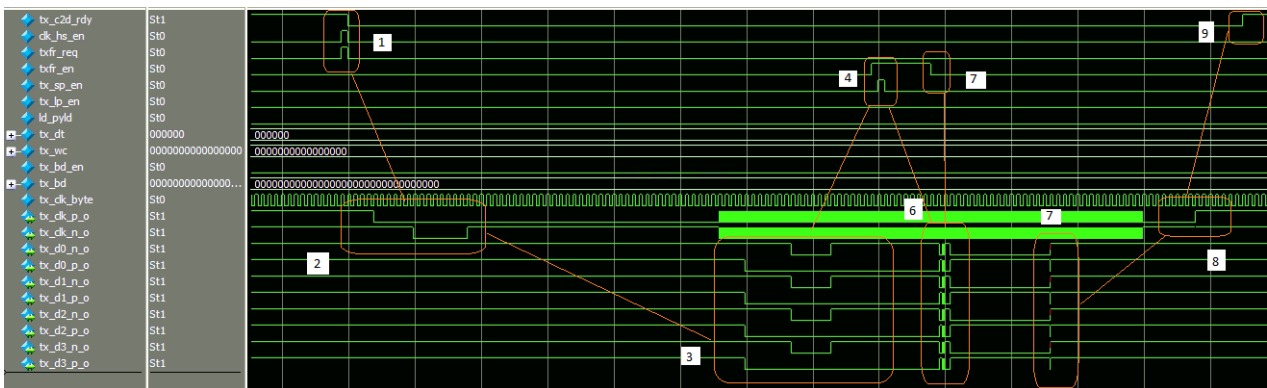


Figure 3.7. LP-HS-LP Transition in Non-Continuous Clock Mode (Short Packet)

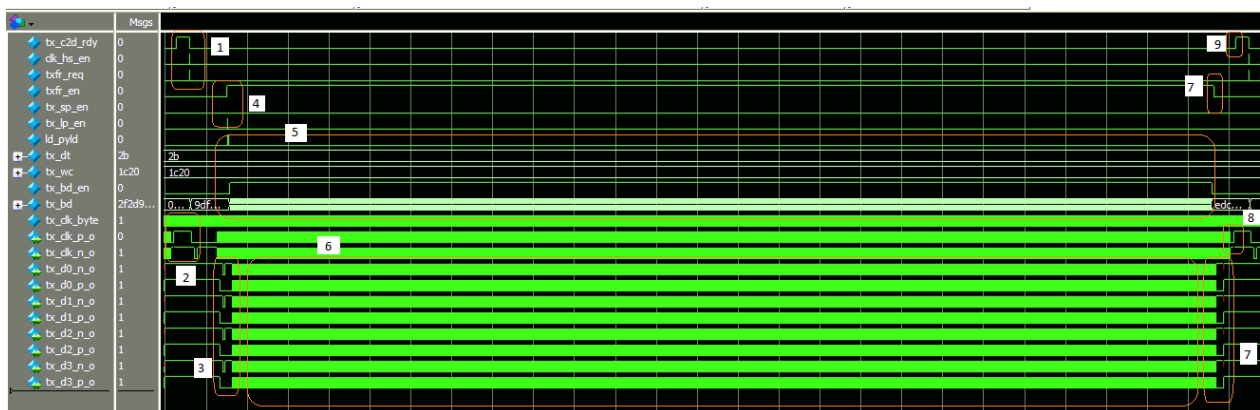


Figure 3.8. LP-HS-LP Transition in Non-Continuous Clock Mode (Long Packet)

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 3.9. clk_hs_en stays 1 from Frame Start to Frame End that makes clock lane in HS mode without going into 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.

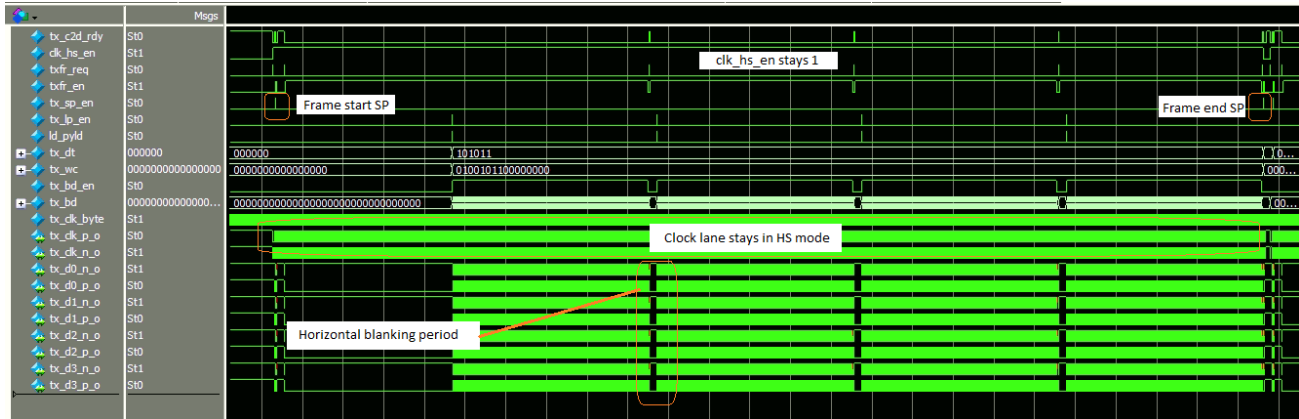


Figure 3.9. LP-HS-LP Transition in Non-Continuous Clock Mode with KEEP_HS

3.3.2. LP-HS Control in Continuous Clock Mode

Figure 3.10 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 3.7 is clock lane stays in HS mode all the time and does not go into LP mode.

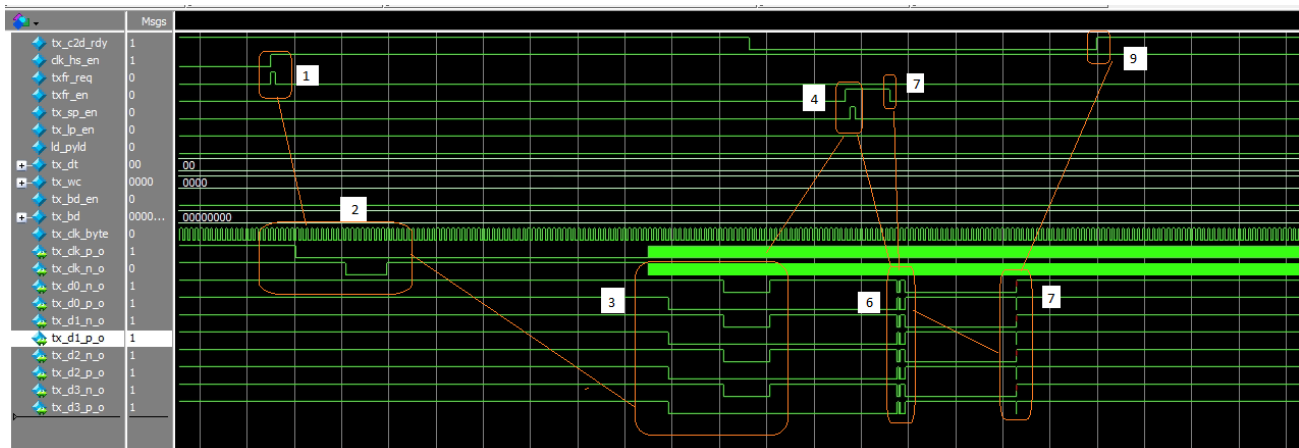


Figure 3.10. LP-HS-LP Transition in Continuous Clock Mode (Short Packet)

Figure 3.11 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_clk_byte 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.

[illegible]

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.

© 2024 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.4. tx_dphy

You must create this module according to channel conditions, such as number of lanes, bandwidth, and others.

Figure 3.12 shows an example IP interface setting in Clarity Designer for the CSI-2/DSI D-PHY Transmitter Submodule IP. Refer to [CSI-2/DSI DPHY Tx IP Core User Guide \(FPGA-IPUG-02080\)](#) for details.

Configure Component from IP dphy_tx Version 1.4.0
Set the following parameters to configure this component.

Diagram tx_dphy

Configure IP

General		Protocol Timing Parameters	
Property	Value		
Transmitter			
TX Interface Type	CSI-2		
D-PHY TX IP	Soft D-PHY		
Number of TX Lanes	4		
TX Gear	8		
Interleaved Input Data	☐		
CIL Bypass	☒		
Bypass Packet Formatter	☐		
Enable Frame Number Increment in Packet Formatter	☐		
Frame Number MAX Value Increment in Packet Formatter [1 - 255]	1		
Enable Line Number Increment in Packet Formatter	☐		
EoTp Enable	☐		
Enable LMMI Interface	☐		
Enable AXI4-Stream Interface	☐		
Enable Periodic Skew Calibration	☐		
Clock			
Target TX Line Rate (Mbps per Lane) [160 - 1500]	800		
Target TX Data Rate (Mbps) [160 - 10000]	3200		
Target D-PHY Clock Frequency (MHz) [80 - 1250]	400		
Target Byte Clock Frequency (MHz) [10 - 187.5]	100		
D-PHY Clock Mode	Non-Continuous		

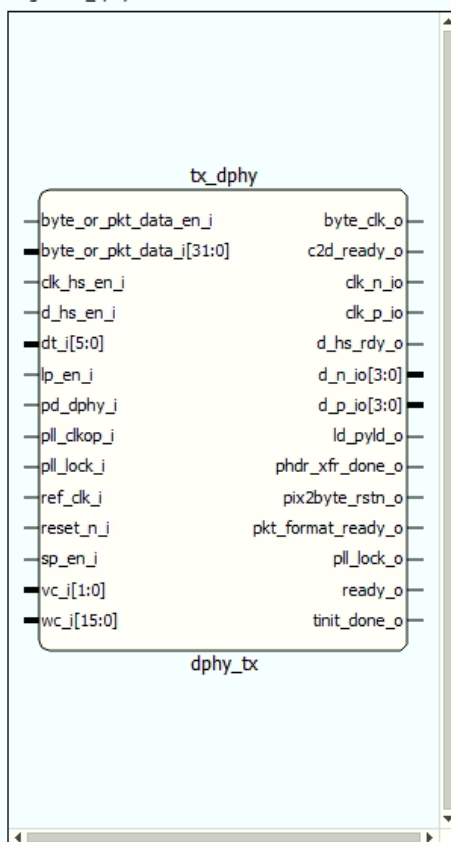
No DRC issues are found.

Generate

Cancel

Configure Component from IP dphy_tx Version 1.4.0
Set the following parameters to configure this component.

Diagram tx_dphy



Configure IP

General		Protocol Timing Parameters	
Property	Value		
Enable Periodic Skew Calibration	☐		
▼ Clock			
Target TX Line Rate (Mbps per Lane) [160 - 1500]	800		
Target TX Data Rate (Mbps) [160 - 10000]	3200		
Target D-PHY Clock Frequency (MHz) [80 - 1250]	400		
Target Byte Clock Frequency (MHz) [10 - 187.5]	100		
D-PHY Clock Mode	Non-Continuous		
D-PHY PLL Mode	External		
Reference Clock Frequency (MHz) [24 - 200]	25		
Actual D-PHY TX Data Rate (Mbps) [160 - 10000]	3200		
Actual TX Line Rate (Mbps)	800		
Actual D-PHY Clock Frequency (MHz) [80 - 1250]	400		
Actual Byte Clock Frequency (MHz) [10 - 187.5]	100		
Deviation from Target Data Rate	0.00 %		
▼ Initialization			
tINIT Counter	☒		
tINIT_SLAVE Value (Number of Byte Clock Cycles) [1 - 32768]	15000		
tinit Value in ns	150000		
▼ Miscellaneous			
Enable Miscellaneous Status Signals	☒		

No DRC issues are found.

Generate Cancel

Figure 3.12. tx_dphy IP Creation in Lattice Radiant

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

- TX Interface – Select CSI-2.
- DPHY TX IP – Always Soft DPHY.
- Number of TX Lanes – Set according to channel configuration. Must match NUM_TX_LANE_* setting.
- TX Gear – Always 8.
- CIL Bypass – Select Enabled.
- Bypass Packet Formatter – Must be disabled (unchecked).
- Enable Frame Number Increment – Set according to the preference (unchecked).
- Enable Line Number Increment – Must be disabled (unchecked).
- Enable LMMI Interface – Must be disabled (unchecked).
- Enable AXI4-Stream Interface – Must be disabled (unchecked).
- TX Line Rate per Lane – Set according to channel configuration. This value must be equal to (number of RX channel) x (RX channel bandwidth) / (number of TX lanes).
- DPHY Clock Mode – Set according to channel configuration. Must select Continuous when the horizontal blanking period is short.
- DPHY PLL Mode – Select External.
- 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 between 24 and 200 MHz.
- tINIT Counter – Enabled (checked) is recommended.
- tINIT_SLAVE Value – the value to make tinit Value over 100 μ s 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 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 1500 Mb/s, which means TX byte clock could be ~187.5 MHz.

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

$$24MHz \leq \frac{1}{NI} * ref_clk_frequency \leq 50 MHz,$$

$$1250MHz \leq \frac{1}{NI} * ref_clk_frequency * M \leq 2500MHz$$

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_aggr_ss_LFCPNX.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. int_gpll is the name used in this top-level design. This name has to be changed if the different name is used.

```

////////////////////////////////////
//  PLL CLOCKING STRUCTURE
//  Continuous clock mode : Use PLL to generate only TxDPHY clock (Tx linerate/ 2)
//  Non-Cont clock mode : Use PLL to generate continuous Rx byte clock and TxDPHY clock
////////////////////////////////////

`ifndef RX_CLK_MODE_HS_ONLY

    int_gpll (
        .rstn_i          (1'b1          ),
        .clki_i           (rx0_clk_byte_hs), // Continuous Rx byte clock
        .clkop_o          (pll_clkop     ),
        .lock_o           (gpll_lock     )
    );
    assign rx0_clk_byte_fr = rx0_clk_byte_hs;
    assign tx_ref_clk      = rx0_clk_byte_hs; // Soft Tx DPHY input reference
clock
    assign tx_pll_clkop    = pll_clkop;      // Soft Tx DPHY clock

`elsif RX_CLK_MODE_HS_LP

    int_gpll (
        .rstn_i          (1'b1          ),
        .clki_i           (ref_clk_i),      // external clock input
        .clkop_o          (pll_clkop),
        .clkos_o          (pll_clkos),
        .lock_o           (gpll_lock)
    );
    assign rx_clk_byte_fr = pll_clkop;
    assign tx_pll_clkop   = pll_clkos;
    assign tx_ref_clk     = ref_clk_i;

`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 24 to 200 MHz.

4. Design and File Modifications

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_aggr_ss_LFCPNX.v) takes the primary GPLL clock to feed a clock to TX D-PHY and takes the byte clock of RX channel 0 or external clock 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_s, tx_dphy) have to be modified if you created these IP with different names.

5. Design Simulation

The script file (csi2_aggr_ss_LFCPNX_msim.do) and testbench files are provided to run the functional simulation by ModelSim. You have to launch ModelSim from Lattice Radiant 3.0 or above. If you follow the naming recommendations regarding instance name when RX and TX D-PHY IPs are created by Lattice Radiant, the following are the only changes required in the script file:

- User project directory

```
### Set Customer's project/simulation directory ###
set project_dir ~/Desktop/rd_MIPI_Side_by_Side_LFCPNX/
set sim_dir $project_dir/simulation/lfcpx/

cd $sim_dir
```

Own Project Directory

Figure 5.1. Script Modification #1

```
### Compiling modules ###
vlog -mfcu \
+incdir+$project_dir/testbench/verilog \
+incdir+$project_dir/source/verilog/lfcpx \
+incdir+$project_dir/int_osc/rtl \
+incdir+$project_dir/int_gpll/rtl \
+incdir+$project_dir/tx_dphy/rtl \
+incdir+$project_dir/rx_dphy_s/rtl \
$project_dir/testbench/verilog/simulation_directives.v \
$project_dir/source/verilog/lfcpx/synthesis_directives.v \
$project_dir/int_osc/rtl/int_osc.v \
$project_dir/int_gpll/rtl/int_gpll.v \
$project_dir/tx_dphy/rtl/tx_dphy.v \
$project_dir/rx_dphy_s/rtl/rx_dphy_s.v \
$project_dir/testbench/verilog/csi2_aggr_ss_LFCPNX_tb.v \
$project_dir/source/verilog/lfcpx/csi2_aggr_ss_LFCPNX.v \
$project_dir/source/verilog/lfcpx/rx_unit.v \
$project_dir/source/verilog/lfcpx/line_buf.v \
$project_dir/source/verilog/lfcpx/merge_ctrl_encrypt.v \

vsim -voptargs=+acc=ap work.top -L pmi_work -L ovi_lfcpx -c -do "add wave -r top/* ;run -all; quit"
```

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 Modelsim, 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:

```
# 127257560 tinit_done detected
# 15119750000 tinit_done detected
#
#      15124750000 Activating dphy models
#
#      15124750100 DPHY CH 1 model activated
#
#      15124750400 DPHY CH 4 model activated
#
#      15124750500 DPHY CH 5 model activated
#
#      15124750700 DPHY CH 6 model activated
#
#      15124750800 DPHY CH 7 model activated
#
#      15124750900 DPHY CH 2 model activated
#
#      15124750900 DPHY CH 3 model activated
#
#      15124751100 DPHY CH 0 model activated
#
#      23124850100 DPHY CH 1 CLK : Driving HS-CLK-RQST
#      23124850400 DPHY CH 4 CLK : Driving HS-CLK-RQST
#      23124850500 DPHY CH 5 CLK : Driving HS-CLK-RQST
#      23124850700 DPHY CH 6 CLK : Driving HS-CLK-RQST
#      23124850800 DPHY CH 7 CLK : Driving HS-CLK-RQST
#      23124850900 DPHY CH 2 CLK : Driving HS-CLK-RQST
#      23124850900 DPHY CH 3 CLK : Driving HS-CLK-RQST
#      23124851100 DPHY CH 0 CLK : Driving HS-CLK-RQST
#      23129850100 DPHY CH 1 CLK : Driving HS-Prpr
#      23129850400 DPHY CH 4 CLK : Driving HS-Prpr
#      23129850500 DPHY CH 5 CLK : Driving HS-Prpr
#      23129850700 DPHY CH 6 CLK : Driving HS-Prpr
#      23129850800 DPHY CH 7 CLK : Driving HS-Prpr
#      23129850900 DPHY CH 2 CLK : Driving HS-Prpr
#      23129850900 DPHY CH 3 CLK : Driving HS-Prpr
#      23129851100 DPHY CH 0 CLK : Driving HS-Prpr
#      23133650100 DPHY CH 1 CLK : Driving HS-Go
#      23133650400 DPHY CH 4 CLK : Driving HS-Go
#      23133650500 DPHY CH 5 CLK : Driving HS-Go
#      23133650700 DPHY CH 6 CLK : Driving HS-Go
#      23133650800 DPHY CH 7 CLK : Driving HS-Go
#      23133650900 DPHY CH 2 CLK : Driving HS-Go
#      23133650900 DPHY CH 3 CLK : Driving HS-Go
#      23133651100 DPHY CH 0 CLK : Driving HS-Go
#      23159850100 DPHY CH 1 CLK : Driving HS-0/HS-1
#      23159850400 DPHY CH 4 CLK : Driving HS-0/HS-1
#      23159850500 DPHY CH 5 CLK : Driving HS-0/HS-1
#      23159850700 DPHY CH 6 CLK : Driving HS-0/HS-1
#      23159850800 DPHY CH 7 CLK : Driving HS-0/HS-1
#      23159850900 DPHY CH 2 CLK : Driving HS-0/HS-1
#      23159850900 DPHY CH 3 CLK : Driving HS-0/HS-1
#      23159851100 DPHY CH 0 CLK : Driving HS-0/HS-1
#      23163850100 DPHY CH 1 DATA : Driving HS-RQST
#      23163850400 DPHY CH 4 DATA : Driving HS-RQST
#      23163850500 DPHY CH 5 DATA : Driving HS-RQST
#      23163850700 DPHY CH 6 DATA : Driving HS-RQST
#      23163850800 DPHY CH 7 DATA : Driving HS-RQST
#      23163850900 DPHY CH 2 DATA : Driving HS-RQST
#      23163850900 DPHY CH 3 DATA : Driving HS-RQST
```

```
# 23163851100 DPHY CH 0 DATA : Driving HS-RQST
# 23168850100 DPHY CH 1 DATA : Driving HS-Prpr
# 23168850400 DPHY CH 4 DATA : Driving HS-Prpr
# 23168850500 DPHY CH 5 DATA : Driving HS-Prpr
# 23168850700 DPHY CH 6 DATA : Driving HS-Prpr
# 23168850800 DPHY CH 7 DATA : Driving HS-Prpr
# 23168850900 DPHY CH 2 DATA : Driving HS-Prpr
# 23168850900 DPHY CH 3 DATA : Driving HS-Prpr
# 23168851100 DPHY CH 0 DATA : Driving HS-Prpr
# 23174850100 DPHY CH 1 CLK : Driving HS-Go
# 23174850400 DPHY CH 4 CLK : Driving HS-Go
# 23174850500 DPHY CH 5 CLK : Driving HS-Go
# 23174850700 DPHY CH 6 CLK : Driving HS-Go
# 23174850800 DPHY CH 7 CLK : Driving HS-Go
# 23174850900 DPHY CH 2 CLK : Driving HS-Go
# 23174850900 DPHY CH 3 CLK : Driving HS-Go
# 23174851100 DPHY CH 0 CLK : Driving HS-Go
# 23188500100 DPHY CH 0 CLK : Driving SYNC Data
# 23188500100 DPHY CH 0 Lane 0 : Driving with data = b8
# 23188500100 DPHY CH 0 Lane 1 : Driving with data = b8
# 23188500100 DPHY CH 1 CLK : Driving SYNC Data
# [72087812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19141 - 19144, Payload Data matches 55,
19, c5, d1 from CH#7
# [72088812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19145 - 19148, Payload Data matches 1b,
f3, 74, 3b from CH#7
# [72089812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19149 - 19152, Payload Data matches 60,
95, 37, fc from CH#7
# [72090812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19153 - 19156, Payload Data matches 81,
5b, 53, 6c from CH#7
# [72091812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19157 - 19160, Payload Data matches b1,
99, eb, db from CH#7
# [72092812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19161 - 19164, Payload Data matches fd,
9b, f4, 91 from CH#7
# [72093812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19165 - 19168, Payload Data matches 48,
a6, 3f, cf from CH#7
# [72094812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19169 - 19172, Payload Data matches 4d,
f4, 73, cb from CH#7
# [72095812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19173 - 19176, Payload Data matches a0,
ba, 11, b5 from CH#7
# [72096812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19177 - 19180, Payload Data matches ae,
23, 6f, b7 from CH#7
# [72097812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19181 - 19184, Payload Data matches b9,
55, be, ef from CH#7
# [72098812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19185 - 19188, Payload Data matches dd,
6a, 05, 76 from CH#7
# [72099812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19189 - 19192, Payload Data matches 0f,
76, 23, 5b from CH#7
# [72100812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19193 - 19196, Payload Data matches 19,
86, d2, a7 from CH#7
# [72101812500][DPHY_CHK] Frame 2, Line 4, Byte Count 19197 - 19200, Payload Data matches 9f,
9e, 9f, 5a from CH#7
# [72102812500][DPHY_CHK] CRC = 68bd, matches
# [72117750000][DPHY_HS_LP_CHK] HS-TRAIL period is too short or too long on D0 lane or clock
lane is not in HS mode!!! HS_TRAIL period = 150 ns, must be between 65 and 120 ns!!!
# [72139750000][DPHY_HS_LP_CHK] Unexpected D0 Lane Transition from HS to LP00!!!
# [72163812500][DPHY_CHK] Short Packet detected : Data type = 01
# [72163812500][DPHY_CHK] 01 00 00 07 matches Expected Short Packet Data
# [72178750000][DPHY_HS_LP_CHK] HS-TRAIL period is too short or too long on D0 lane or clock
lane is not in HS mode!!! HS_TRAIL period = 0 ns, must be between 65 and 120 ns!!!
# [72200750000][DPHY_HS_LP_CHK] HS to LP11 Transition on clock lane with TRAIL period = 80 ns
# 72373000000 TEST END
#
```

```
##### 2 Frames x 4 Lines x 19200 Payload Bytes comparison succeeded!!! #####
#
### Simulation Success ###
#
```

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 - 200 MHz or its multiple. Following is an example of the error message when the PLL's input clock is out of the range:

```
# ERROR : PLL won't work!
# ERROR : error_clkref_N_M_range
# ERROR : clkref_N_M_period      868 - clkref_N_M_min_period =      400 -
clkref_N_M_max_period =      800
```

Assuming the external clock drives directly this reference clock and the frequency is 24 MHz, the clock setting of this external clock can be ``define REF_CLK_PERIOD 41667` in simulation_directives.v, but this fails since the clock period of 41.667 ns leads the clock frequency slightly lower than 24.000 MHz. PLL model of D-PHY TX IP cannot accept this and PLL does not work. Therefore, the value must be 41666 or smaller 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 1250 – 2500 MHz. Following is an example of the error message when PLL's VCO frequency is out of the range:

```
# ERROR : PLL won't work!
# ERROR : error_clkref_N_M_range
# ERROR : clkref_N_M_period      400 - clkref_N_M_min_period =      400 -
clkref_N_M_max_period =      800
```

Assuming the RX lane bandwidth is 500 Mb/s with Gear 8 in continuous clock mode and TX lane bandwidth is 1500 Mb/s. The continuous byte clock is 62.5 MHz and this can drive the reference clock of tx_dphy. RX0 D-PHY clock frequency of 250 MHz (= 500/2) can be set by ``define RX0_DPHY_CLK_PERIOD 4000` in simulation_directives.v, but it will fail if the clock period is set to 3999 since it will leads the byte clock frequency slightly higher than 62.500 MHz and PLL's VCO frequency will be above 2500 MHz. PLL model of D-PHY TX IP cannot accept this and PLL does not work. Therefore, the value must be 4000 or larger in this case.

The image displays a Wireshark packet capture of a corrupted Ethernet II frame. The packet list on the left shows a packet of length 1500. The packet details pane shows the frame structure: Ethernet II (Type II), Internet Protocol Version 4, and Hypertext Transfer Protocol. The packet bytes pane highlights the 'Frame Start Short Packet' and 'Frame End Short Packet' anomalies. The packet bytes pane shows the frame structure: Ethernet II (Type II), Internet Protocol Version 4, and Hypertext Transfer Protocol.

© 2024 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

6. Design Debug on Hardware

Hardware debug on MIPI related design is a challenge when the system is not working and it is not clear where the issue comes from, no matter it is related to FPGA or not. This section shows some guidelines to find where the problem is happening. Note that what described here provides some ideas about how to debug the system including FPGA functionality, but that does not mean this section covers all possible scenarios.

6.1. Top-Level

In any design cases, the following are essential checkpoints you should check first:

- Power Supply or any board related issues
- Pin assignments / Signal Connections
- Reset Signals
- Clock Signals including PLL outputs
- Parameter and/or Mode mismatch

6.2. MIPI Clock Lane Transactions in D-PHY RX IP

After the initial checking is done, the typical way to check the behavior is to start from the module closest to the incoming signals, which is D-PHY RX IP. In case of continuous clock mode, MIPI clock is always alive in HS mode and usually a problem doesn't occur. In case of non-continuous clock mode, MIPI clock changes its mode between LP (Low Power) and HS (High Speed). [Figure 6.2](#) shows MIPI Clock lane transition between LP and HS mode with debug signals. If you don't see rx*_clk_byte_hs toggles, you can check rx*_lp_hs_state_clk and rx*_term_clk_en to see those signals behaves as expected. If not, you might need to probe MIPI clock signals to check its timings. [Figure 6.2](#) and [Table 6.1](#) show MIPI clock lane transition and its timing requirements defined by MIPI. [Figure 6.3](#) shows the clock lane transition from LP to HS mode captured by a scope using single-end probes.

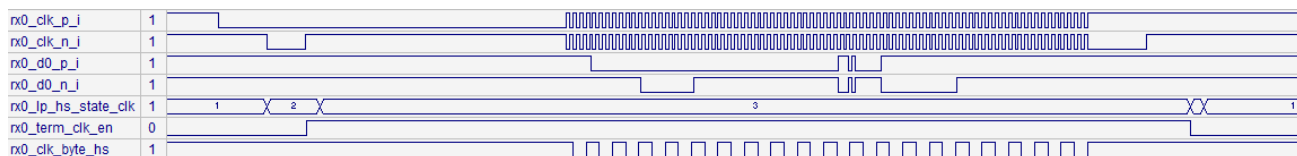


Figure 6.1. MIPI Clock Lane Transition and Expected Debug Signal Behaviors

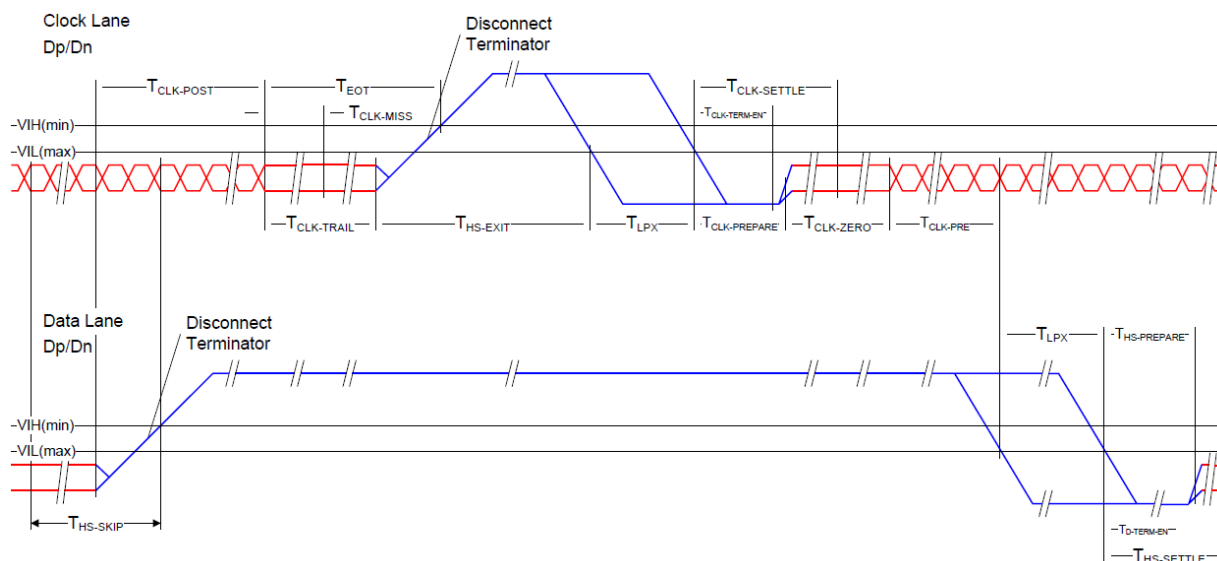


Figure 6.2. MIPI Clock Lane Transition

Table 6.1. Timing Parameters for Clock Lane Signal Transitions

Parameter	Description	Min	Typ	Max	Unit
T _{CLK-MISS}	Timeout for receiver to detect absence of Clock transitions and disable the Clock Lane HS-RX.			60	ns
T _{CLK-POST}	Time that the transmitter continues to send HS clock after the last associated Data lane has transitioned to LP Mode. Interval is defined as the period from the end of T _{HS-TRAIL} to the beginning of T _{CLK-TRAIL} .	60 ns + 52*UI			ns
T _{CLK-PRE}	Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode.	8			UI
T _{CLK-PREPARE}	Time that the transmitter drives the Clock Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	38		95	ns
T _{CLK-SETTLE}	Time interval during which HS receiver should ignore any Clock Lane HS transition, starting from the beginning of T _{CLK-PREPARE} .	95		300	ns
T _{CLK-TERM-EN}	Time for the Clock Lane receiver to enable the HS line termination, starting from the time point when Dn crosses V _{IL,MAX} .	Time for Vn to reach V _{TERM-EN}		38	ns
T _{CLK-TRAIL}	Time that the transmitter drives the HS-0 state after the last payload clock bit of a HS transmission burst.	60			ns
T _{CLK-PREPARE} + T _{CLK-ZERO}	T _{CLK-PREPARE} + time that the transmitter drives the HS-0 state prior to starting the Clock.	300			ns
T _{D-TERM-EN}	Time for the Data Lane receiver to enable the HS line termination, starting from the time point when Dn crosses V _{IL,MAX} .	Time for Vn to reach V _{TERM-EN}		35 ns + 4*UI	
T _{EOT}	Transmitted time interval from the start of T _{HS-TRAIL} or T _{CLK-TRAIL} , to the start of the LP-11 state following an HS burst.			105 ns + n*12*UI	
T _{HS-EXIT}	Time that the transmitter drives LP-11 following an HS burst.	100			ns

Note: UI is Unit Interval (= 1 HS Data Period = ½ of MIPI HS Clock Cycle)

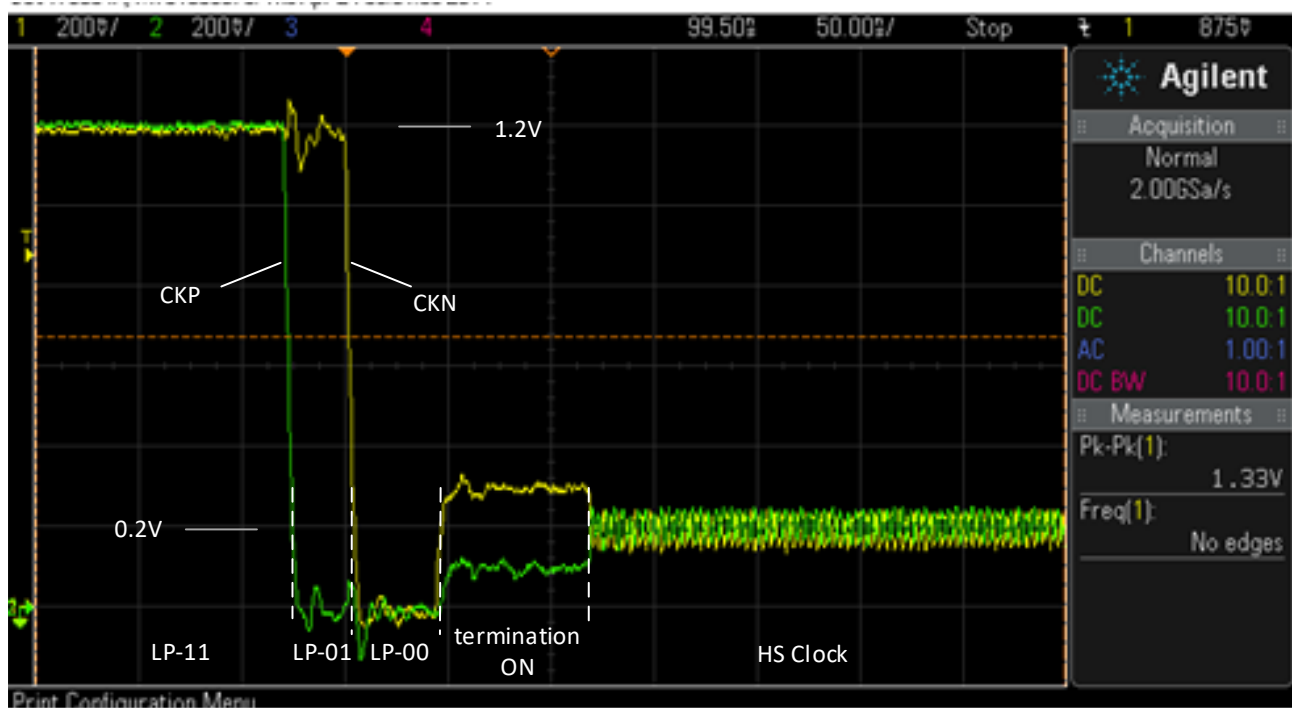


Figure 6.3. MIPI Clock Lane Transition from LP to HS Mode

6.3. MIPI Data Lane Transactions in D-PHY RX IP

In case of MIPI data lane checking, the global operation can be checked by monitoring `rx*_sp_en`, `rx*_lp_av_en`, and `rx*_payload_en` shown in Figure 5.3 since these signals indicate the detection of the expected HS mode transactions. `sp_en` must be asserted twice every frame and `lp_av_en` and `payload_en` must be asserted as many times as active line counts every frame with even intervals. In case that you see `sp_en` assertions, but no `lp_av_en` assertion, there is high possibility of Data Type (RAW10, RAW12, etc.) mismatch between the sensor and FPGA design (Data Type directives in `synthesis_driectives.v`).

When these signals are not asserted at all or asserted irregularly, then following are two most common scenarios:

[SoT Detection failure]

SoT (Start of Transmission) sync word (0xB8) exists in all lanes preceding Short Packet or Long Packet header. You can check the signal `rx*_hs_sync` comes out from RX D-PHY IP. Figure 6.4 shows a typical signal behavior of `rx*_hs_sync` along with other signals in Short Packet transaction. `rx*_hs_sync` is asserted as a result of B8 detection and kept high until the end of HS transmissions. If `hs_sync` assertion doesn't happen, no HS transaction including `sp_en`, `lp_av_en` will happen. When this issue occurs, most probable cause is missing the detection of B8 due to long waiting time, which is led by an improper value of Data Settle Cycle set in RX D-PHY IP (`rx_dphy_s/rx_dphy_s`) GUI on Lattice Radiant when this IP was created. In that case, you can expect to make it work by decreasing this value.

When Data Settle Cycle adjustment still does not solve the issue, you might have to check the MIPI signal themselves to confirm those are not violating MIPI spec. Figure 6.5 shows the data lane signal transitions from LP mode to HS mode and HS mode to LP mode. Also Table 6.2 shows timing parameters used in Figure 6.5. These timings are defined by data lane bandwidth (= 2x of MIPI clock frequency).

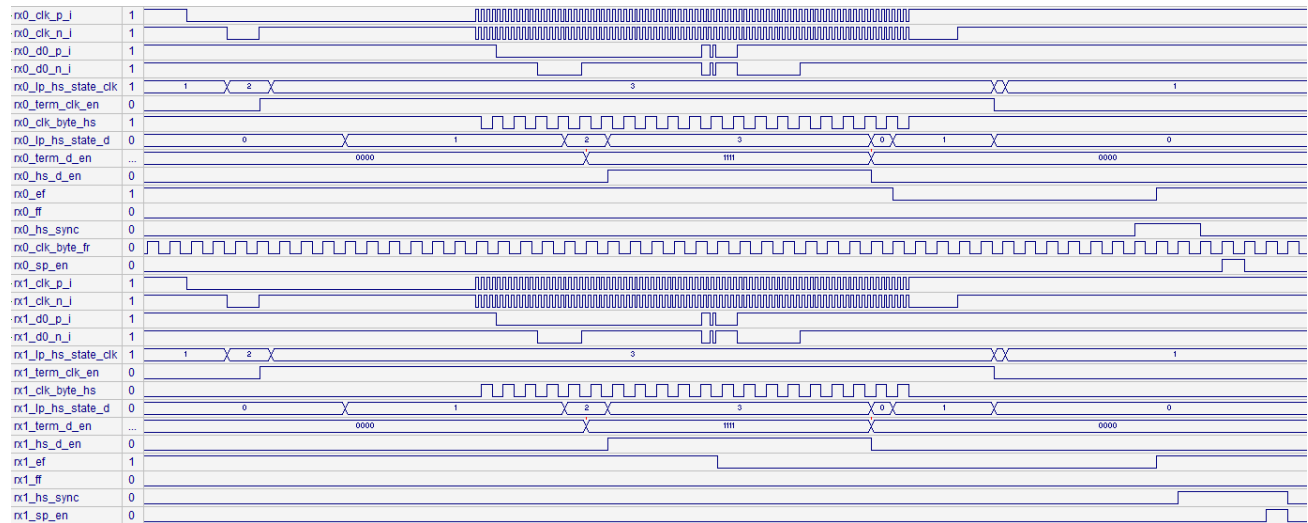


Figure 6.4. Debug Signals of D-PHY RX IP

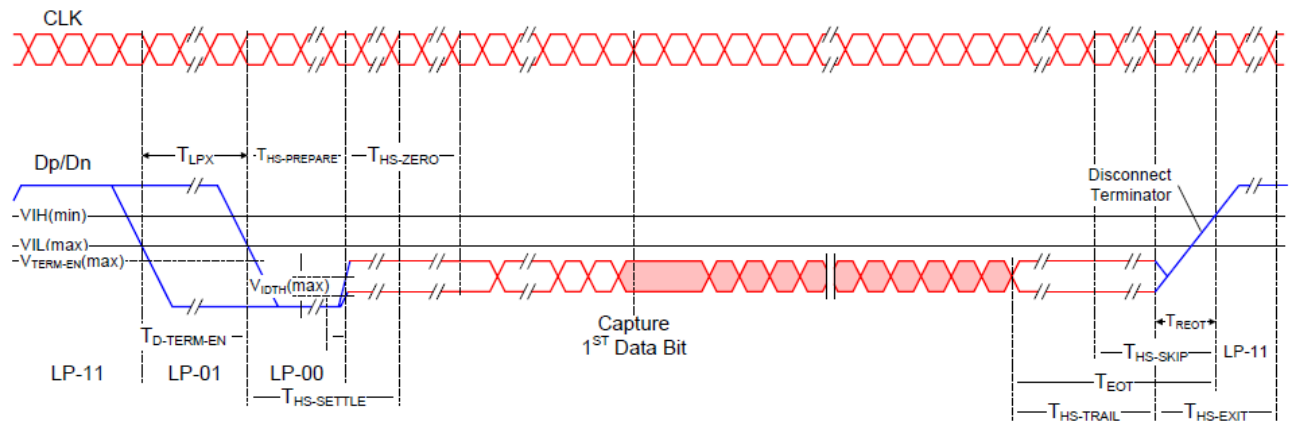


Figure 6.5. MIPI Data Lane Signal Transitions

Table 6.2. Timing Parameters for Data Lane Signal Transitions

Parameter	Description	Min	Typ	Max	Unit
$T_{HS-PREPARE}$	Time that the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	$40\text{ ns} + 4*UI$		$85\text{ ns} + 6*UI$	ns
$T_{HS-PREPARE} + T_{HS-ZERO}$	$T_{HS-PREPARE}$ + time that the transmitter drives the HS-0 state prior to transmitting the Sync sequence.	$145\text{ ns} + 10*UI$			ns
$T_{HS-SETTLE}$	Time interval during which HS receiver shall ignore any Data Lane HS transitions, starting from the beginning of $T_{HS-PREPARE}$. The HS receiver shall ignore any Data Lane transitions before the minimum value, and the HS receiver shall respond to any Data Lane transitions after the maximum value.	$85\text{ ns} + 6*UI$		$145\text{ ns} + 10*UI$	ns
$T_{HS-SKIP}$	Time interval during which the HS-RX should ignore any transitions on the Data lane, following an HS burst. The end point of the interval is defined as the beginning of the LP-11 state following the HS burst.	40		$55\text{ ns} + 4*UI$	ns
$T_{HS-TRAIL}$	Time that the transmitter drives the flipped differential state after last payload data bit of an HS transmission burst.	$\max(n*8*UI, 60\text{ ns} + n*4*UI)$			ns

Note: UI is Unit Interval (= 1 HS Data Period = ½ of MIPI HS Clock Cycle)

[RX FIFO Underflow / Overflow]

Another possible case related to RX D-PHY IP is underflow or overflow of RX FIFO. As described in [RX FIFO](#) section, proper parameter settings are required to avoid underflow/overflow of RX FIFO according to the frequency relationship between the original byte clock and the continuous byte clock in case of non-continuous clock mode. [Figure 6.6](#) shows an example of one Short Packet and one Long Packet transaction. $rx*_ef$ and $rx*_ff$ are RX FIFO Empty flag and Full flag respectively. As a global operation, $rx*_ef$ behaves oppositely against $rx*_hs_sync$ and $rx*_ff$ stays 0 all the time.

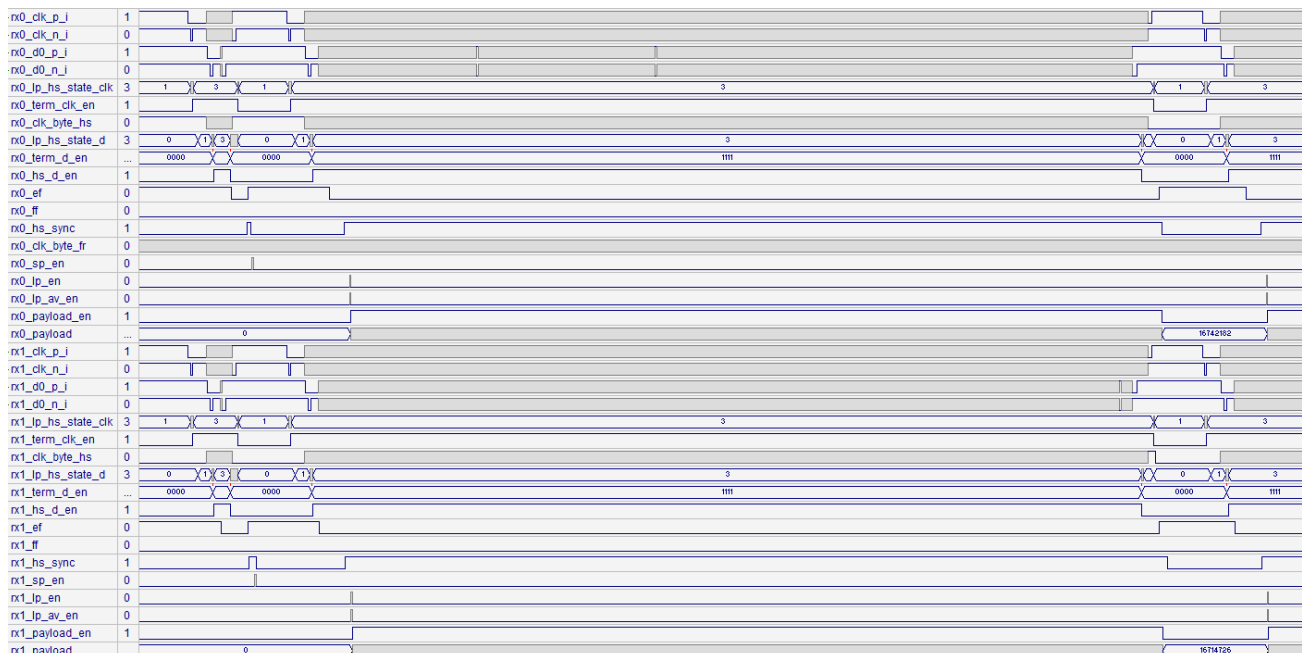


Figure 6.6. Short Packet and Long Packet Transaction Example with Debug Signals

6.4. Inter-Channel Synchronization and Line Buffer Control

Figure 6.7 shows the Line Buffer controls in case of 6 channel aggregation. lb*_uf and lb*_of are underflow and overflow flags of line buffers in respective channels. In this case, each read enable (lb*_lbd_re) must be asserted for 1/6 period against rx*_payload_en and asserted sequentially. All lb*_uf and lb*_of should not be asserted. In this example, all channel data come in almost at the same time, which results in semi-synchronous assertion of all rx*_payload_en. Since line buffers are read out from CH #0 to CH #5 in sequential, it is fine that payload_en assertion of CH #n happens before that of CH #(n+1), but the other way will cause the problem if the timing difference is too large and that makes line buffer underflow and/or overflow. Therefore, monitoring lb*_uf and lb*_of is one way to check whether line buffer control is properly working or not.

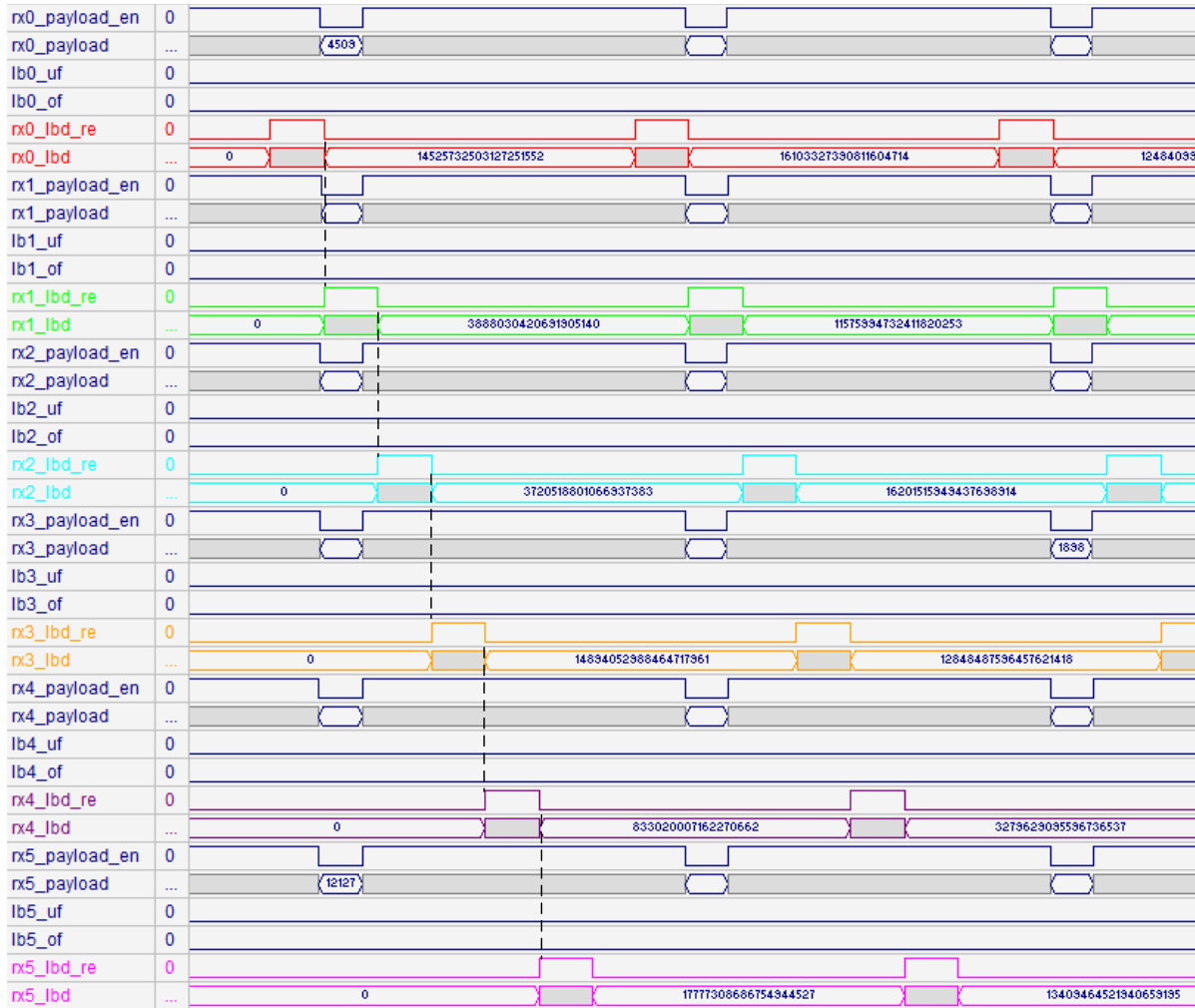


Figure 6.7. Line Buffer Control

6.5. D-PHY TX Control

At least two scenarios exist in case that the system doesn't work related to D-PHY TX IP.

[continuous clock mode operation]

Some downstream devices need LP to HS transition on clock lane even though stating to support continuous clock mode. Using non-continuous clock mode is a way to eliminate this potential issue. You can enable `define KEEP_HS in synthesis_directives.v as described in merge_ctrl section in case that you cannot let clock lane go into LP mode during the horizontal blanking period.

[timing parameter modification]

As described in MIPI Clock Lane Transactions in D-PHY RX IP and MIPI Data Lane Transactions in D-PHY RX IP sections, there exist timing requirements on MIPI signals. The simulation testbench includes the timing checker and issues an error when outgoing MIPI signals violates the MIPI timing specs, but you might want to give some more timing margins when the timing is close to the minimum or maximum values specified by the spec. In that case, you can modify the timing parameters shown in Figure 6.8 on Lattice Radiant GUI to recreate D-PHY TX IP. These values are based on the byte clock cycles.

Configure Component from IP dphy_tx Version 1.4.0
Set the following parameters to configure this component.

Diagram tx_dphy

Configure IP

General		Protocol Timing Parameters	
Property		Value	
TX Global Operation Timing Parameters			
Customize TX Timing Parameter Values		☒	
t_LPX 50ns minimum [1 - 255]		6	
t_HS-PREPARE (40ns + 4UI) to (85ns + 6UI) [1 - 255]		5	
t_HS_ZERO (t_HS-PREPARE + t_HS_ZERO) >= (145ns + 10UI) [1 - 255]		12	
t_HS_TRAIL (60ns + 4UI) minimum [1 - 255]		13	
t_HS_EXIT 100ns minimum [1 - 255]		11	
t_CLK-PREPARE 38ns to 95ns [1 - 255]		8	
t_CLK-ZERO (t_CLK-PREPARE + t_CLK_ZERO) >= 300ns [1 - 255]		41	
t_CLK-PRE 8UI minimum [1 - 255]		2	
t_CLK-POST (60ns + 52UI) minimum [1 - 255]		13	
t_CLK-TRAIL 60ns minimum [1 - 255]		8	
t_CLK-EXIT 100ns minimum [1 - 255]		11	
t_SKEWCAL-INIT 2^15UI to 100us [1 - 4097]		4097	
t_SKEWCAL-PERIOD 2^10UI to 10us [1 - 129]		129	

No DRC issues are found.

Generate

Cancel

Figure 6.8. D-PHT TX IP Timing Parameters

7. Known Limitations

The following are the limitations of this reference design:

- This design does not support RAW14, RGB666 and RGB565 data types.
- 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.

8. Design Package and Project Setup

N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation with CertusPro-NX Reference Design is available on www.latticesemi.com. Figure 8.1 shows the directory structure. The design is targeted for LFCPNX-100. `synthesis_directives.v` and `simulation_directives.v` are set to configure eight RX channels as an example shown below:

- RX CH #8 : 2 lanes with Soft D-PHY with Gear 8 in continuous clock mode
- TX : 4 lanes with Gear 8 in non-continuous clock mode

You can modify the directives for your own configuration.

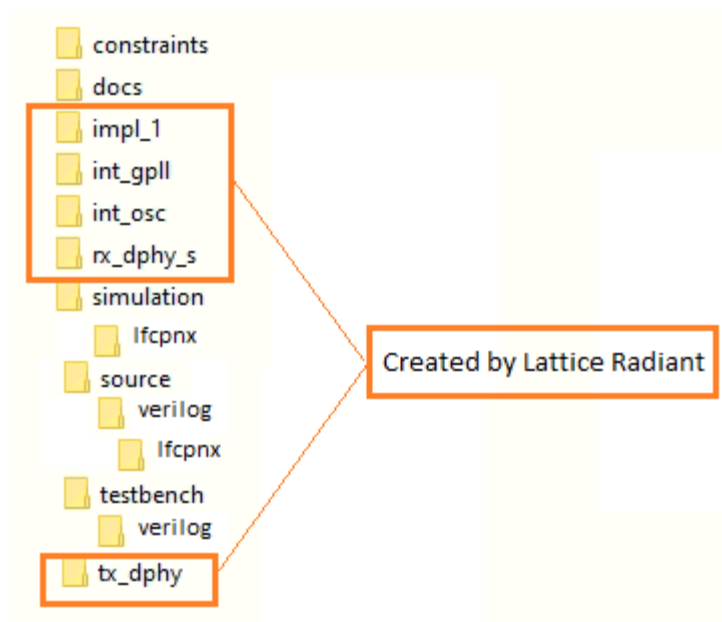


Figure 8.1. Directory Structure

Figure 8.2 shows design files used in the Lattice Radiant project. Including PLL and oscillator modules, Lattice Radiant creates five .ipx files. By specifying `csi2_aggr_ss_LFCPNX` as a top-level design, all unnecessary files will be ignored.

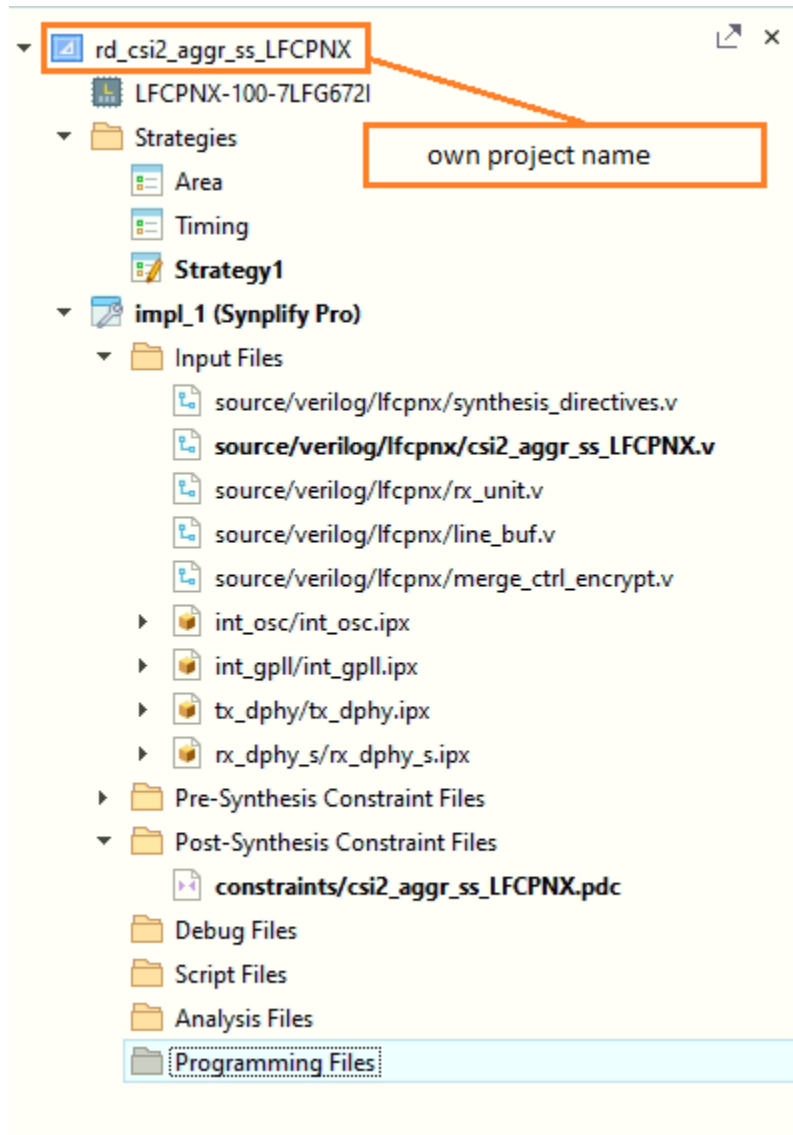


Figure 8.2. Project Files

9. Resource Utilization

Resource utilization depends on the configurations. Table 9.1 shows resource utilization examples under certain configurations targeting LFCPNX-100. This is just a reference and actual usage varies. Especially, EBR usage depends on the horizontal resolution of each RX channels and clock frequency relationship between non-continuous byte clock and continuous byte clock when non-continuous clock mode is used.

Table 9.1. Resource Utilization Examples

Configuration	LUT (Utilization/Total)	FF (Utilization/Total)	EBR (Utilization/Total)	I/O (Utilization/Total)
8 RX Channels with 2 lanes Gear 8 in continuous clock mode, TX 4 lanes Gear 8, RAW10 x 1920 x pixels per line	7714/79872	4670/80769	34/208	59/299
2 RX Channels with 4 lanes Gear 8 in continuous clock mode, TX 4 lane Gear 8, RAW10 x 1920 pixels per line	3410/79872	2111/80769	12/208	31/299
5 RX Channels with 1 lane Gear 8 in non-continuous clock mode, TX 4 lanes Gear 8, RAW10 x 1920 pixels per line	4705/79872	2723/80769	15/208	32/299
6 RX Channels with 2 lane Gear 8 in non-continuous clock mode, TX 2 lanes Gear 8, RAW10 x 1920 pixels per line	6205/79872	3505/80769	30/208	44/299

References

- MIPI® Alliance Specification for D-PHY Version 1.2
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.2
- [CSI/DSI DPHY Rx IP User Guide \(FPGA-IPUG-02081\)](#)
- [CSI/DSI DPHY Tx IP User Guide \(FPGA-IPUG-02080\)](#)
- [CertusPro-NX](#) web page
- [Lattice Radiant Software](#) web page
- [Lattice Insights](#) web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 1.3, September 2024

Section	Change Summary
All	- Renamed document from <i>N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation with CertusPro-NX</i> to <i>CertusPro-NX N Input to 1 Output MIPI CSI-2 Side-by-Side Aggregation</i>. - Changed the unit from <i>Mbps</i> to <i>Mb/s</i>.
Design Simulation	Updated the TX lane bandwidth from 2500 Mb/s to 1500 Mb/s.
References	Updated references.

Revision 1.2, April 2024

Section	Change Summary
Disclaimers	Updated boilerplate.
Inclusive Language	Added boilerplate.
Introduction	Made the following changes to the Features section: - Changed the unit from <i>Gbps</i> to <i>Gb/s</i>. - Updated the <i>maximum TX bandwidth</i> from <i>2.5 Gb/s per lane</i> to <i>1.5 Gb/s per lane</i>.
References	Added a reference to the Lattice Insights web page.
Technical Support Assistance	Added information on the <i>Lattice Answer Database</i> .

Revision 1.1, August 2022

Section	Change Summary
Introduction	Updated the D-PHY Transmitter IP version to 1.5.0 in Table 1.1. Supported Device and IP.
Design and Module Description	Updated Figure 3.11. Line Buffer Read and Data Merge for 5-Channel Aggregation.
Design Debug on Hardware	General update to the introductory paragraph.

Revision 1.0, March 2022

Section	Change Summary
All	Initial release.



www.latticesemi.com