MIPI CSI-2 Virtual Channel Aggregation with CrossLink-NX

Reference Design

FPGA-RD-02148-1.1

December 2023

Disclaimers

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

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
LUT	Look Up Table
MIPI	Mobile Industry Processor Interface
PLL	Phase Locked Loop
GPLL	General Purpose PLL
RX	Receiver
TDM	Time Domain Multiplexing
TX	Transmitter
VC	Virtual Channel

Supported Device and IP

This reference design supports the following devices with IP versions.

Device Family	Part Number	Compatible IP
CrossLink-NX	LIFCL-40 LIFCL-17	D-PHY Receiver IP version 1.5.0 D-PHY Transmitter IP version 1.8.1

The IPs above are supported by Lattice Radiant® Software version 2022.1 or later.

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 MIPI CSI-2 Virtual Channel Aggregation reference design with CrossLink-NX for CrossLink-NX™ devices offers up to eight-channel aggregation. A different virtual channel identification (ID) is assigned to each receiver (RX) channel. CrossLink-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. Features List

- Two to eight independent RX channels can be aggregated.
- You can assign a unique virtual channel ID (0 to 15) to each RX channel.
- Each RX channel can have one, two, or four lanes.
- Number of TX lanes can be one, two, or four.
- Maximum RX bandwidth is 1.5 Gbps per lane.
- Maximum TX bandwidth is 2.5 Gbps per lane by using Hard D-PHY.
- 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. Each RX clock can be independent and does not have to come from the same clock source.

1.2. Block Diagram

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

Since TX D-PHY PLL has an input clock frequency requirement of between 24 MHz and 200 MHz, another on-chip GPLL may have to be used to create an appropriate clock.

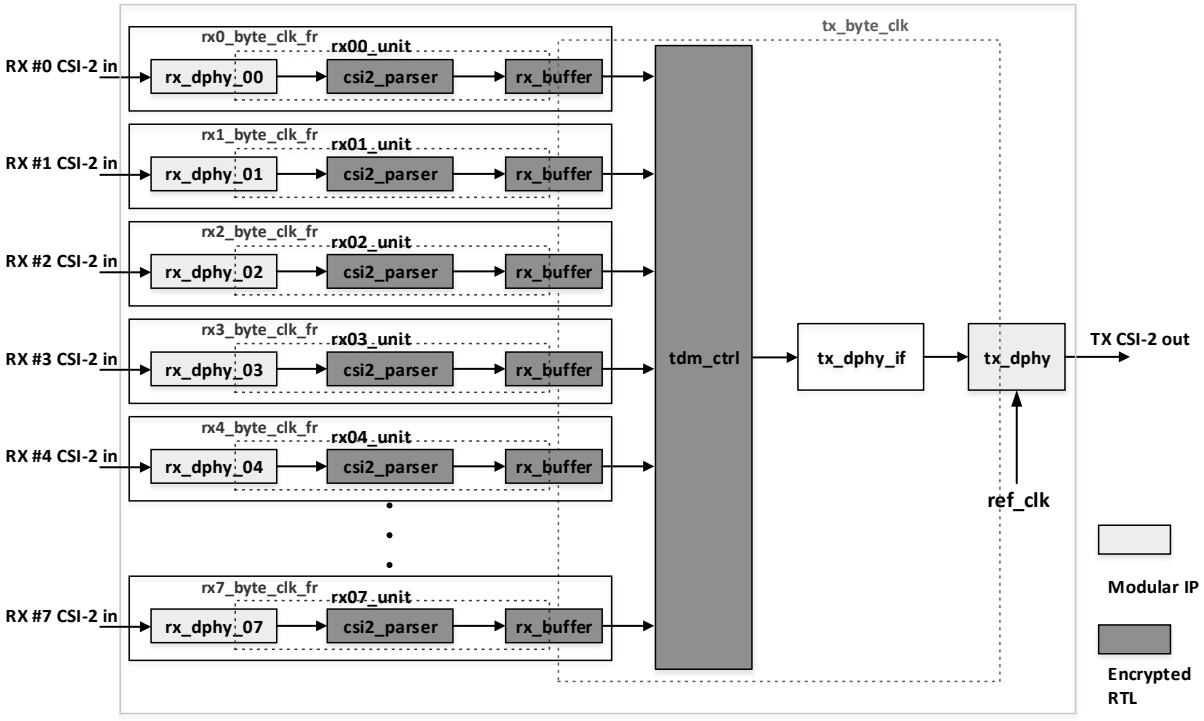

Figure 1.1. CSI-2 Virtual Channel Aggregation Block Diagram

Figure 1.2 shows a clocking scheme example. In this example, RX channel #0 is in continuous clock mode and all other RX channels are in non-continuous clock mode. Sync clock (> 60 MHz) is required for RX Soft D-PHY IP and a clock to drive LP (Low Power) HS (High Speed) mode detection logic is required in non-continuous clock mode. The internal oscillator is used to generate ~75 MHz clock for these purposes. GPLL generates five continuous byte clocks for seven RX channels utilizing the continuous byte clock comes from rx_dphy_00 as a reference clock. By utilizing RX FIFO in RX D-PHY IP, continuous clock does not have to be exactly the same frequency as non-continuous byte clock.

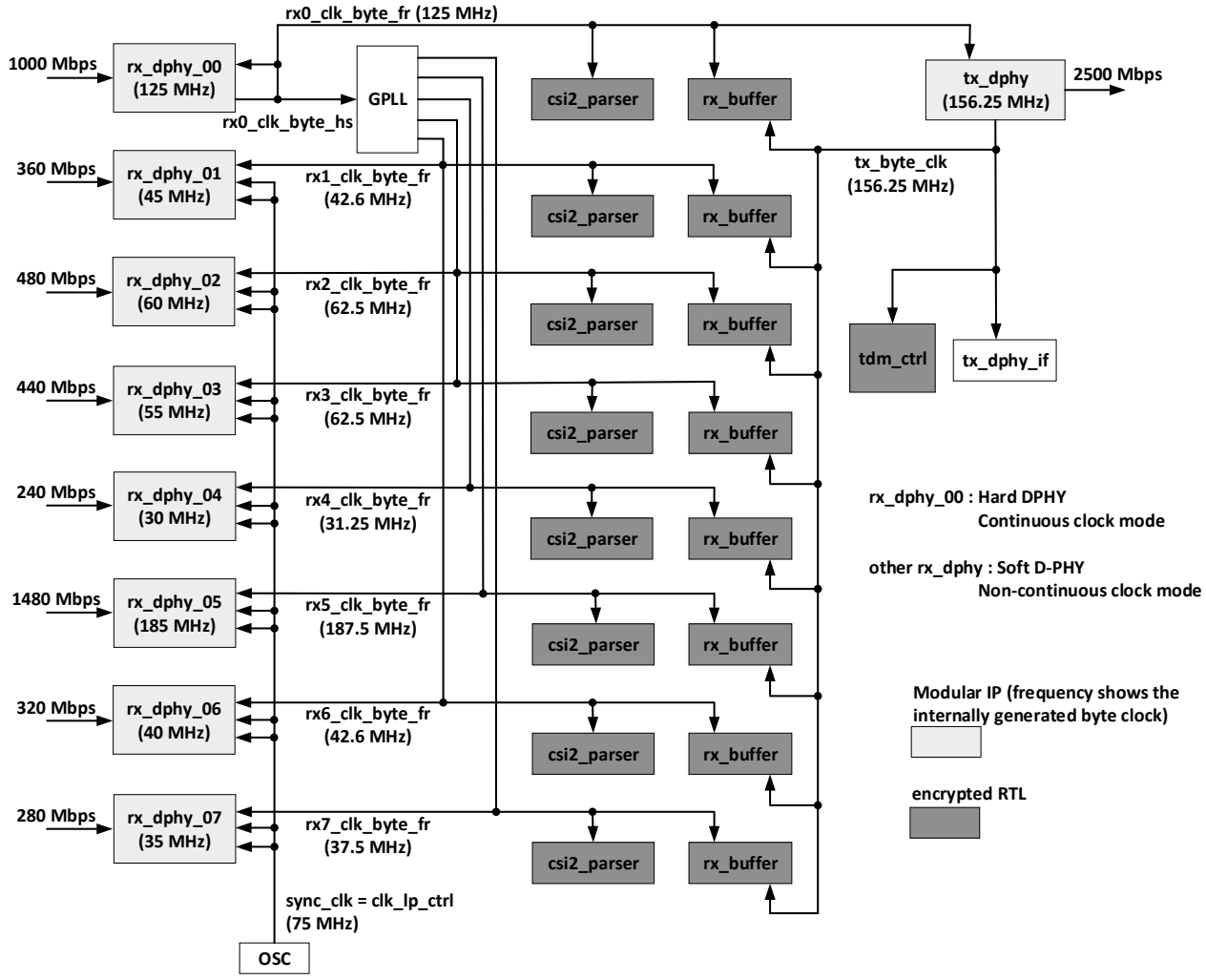

Figure 1.2. Clocking Scheme Example

1.3. RX and TX Permutations

This design allows all RX channels to be independent on number of lanes, bandwidth, and clock mode as long as the total RX bandwidth (plus some overhead) does not exceed the TX capability of 2.5 Gbps per lane. The amount of overhead varies by the number of RX channels, the horizontal blanking period of each RX channels, the active line period, etc., but less than 5% is enough for ordinary cases.

The Microsoft Excel sheet (vc_aggr_NX_clock.xlsx) included in the reference design package is for you to get the byte clock and number of required EBR (Embedded Block RAM) for the design. This sheet is useful to estimate the required TX bandwidth and configure D-PHY RX and TX IP. A sample entry is shown in Figure 1.3. The setting in the sheet is matched the configuration shown in Figure 1.2. By entering the values and modes indicated by the boxes, you can get the byte clock frequency and number of required EBR. In case of non-continuous clock mode, you can set the slightly

higher/lower frequency for the continuous byte clock utilizing RX FIFO when the exact same frequency is not available. Refer to RX FIFO section for details. Also, refer to the rx_buffer section to select the value of “actual RX Buffer Depth”.

Note that some permutations may not be feasible by LIFCL-17 due to limited I/O and resources.

Virtual Channel Aggregation with CrossLink-NX RD Frequency Calculator

		CH #0	CH #1	CH #2	CH #3	CH#4	CH #5	CH #6	CH #7	
Number of RX Channels	8									
RX Rate per lane (Mbps)		1000	360	480	440	240	1200	320	280	
RX DPHY Clock Frequency (Mhz)		500	180	240	220	120	600	160	140	
RX DPHY Clock Period (us)		2000	5555.555556	4166.666667	4545.454545	8333.333333	1666.666667	6250	7142.857143	
Number of RX Lanes		4	4	2	1	1	1	2	2	
RX Gear (fixed to 8)		8	8	8	8	8	8	8	8	
RX Data Type		RAW10	RAW12	RAW14	RAW8	RGB888	YUV420_10	YUV420_8	YUV422_10	
Horizontal Resolution		1920	4096	960	1440	1280	2048	640	800	
Word Count		2400	6144	1680	1440	3840	5120	1280	2000	
min. RX Buffer Depth		300	768	210	180	480	640	160	250	
actual RX Buffer Depth		1024	1024	512	512	1024	1024	512	512	
actual EBR used for RX Buffer		4	4	2	2	4	4	2	2	24 total EBR for RX Buffer
original RX Byte Clock Frequency		125	45	60	55	30	150	40	35	
RX DPHY Clock Mode		continuous	non-continuous	non-continuous	non-continuous	non-continuous	non-continuous	non-continuous	non-continuous	
continous byte clock frequency		**125**	**42.6**	**62.5**	**62.5**	**31.25**	**187.5**	**42.6**	**37.5**	
source of continuous byte clock		self	GPLL_os4	GPLL_os	GPLL_os	GPLL_os2	GPLL_op	GPLL_os4	GPLL_os3	
RX FIFO type		SINGLE	SINGLE	QUEUE	QUEUE	QUEUE	SINGLE	SINGLE	SINGLE	
RX FIFO depth (min. 512 for non-continuous)		16	1024	2048	2048	4096	1024	512	512	
RX FIFO delay		1	1	NA	NA	NA	646	50	100	
# of EBR used for RX FIFO		0	2	2	1	2	1	1	1	10 total EBR for RX FIFO
Recommended value for RX Data Settle Cycle		9 - 15	2 - 3	4 - 6	4 - 6	1 - 1	14 - 24	2 - 3	1 - 2	
Number of TX Lanes	4									34 Total EBR used (max. 84)
TX Gear	16									
min. TX Rate per lane (Mbps)	2370									
min. TX DPHY Clock Frequency (MHz)	1185									
min. TX Byte Clock Frequency (MHz)	148.125									
actual TX Rate per lane (Mbps)	2300									
actual TX DPHY Clock Frequency (MHz)	1150									
actual TX Byte Clock Frequency (MHz)	143.75									
TX DPHY CLK PERIOD	869.5652									

Note: "source of continuous byte clock" and "RX FIFO delay" are informative and do not affect clock frequency or EBR count.

Figure 1.3. Clocking Scheme

2. Parameters and Port List

There are two directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Radiant Software 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 and TX D-PHY IP settings created by Lattice Radiant.

2.1. Synthesis Directives

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

Table 2.1. Synthesis Directives

Category	Directive	Remarks
External reference clock	EXT_REF_CLK	Enable this when the reference clock is fed from a pin.
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	
	NUM_RX_CH_6	
	NUM_RX_CH_7	
	NUM_RX_CH_8	
RX channel lane count	NUM_RX0_LANE_1	Number of lanes in RX channel 0. Only one of these three directives must be defined.
	NUM_RX0_LANE_2	
	NUM_RX0_LANE_4	
	NUM_RX1_LANE_1	Number of lanes in RX channel 1. Only one of these three directives must be defined.
	NUM_RX1_LANE_2	
	NUM_RX1_LANE_4	
	NUM_RX2_LANE_1	Number of lanes in RX channel 2. Only one of these three directives must be defined. Effective when RX channel count is 3 or more.
	NUM_RX2_LANE_2	
	NUM_RX2_LANE_4	
	NUM_RX3_LANE_1	Number of lanes in RX channel 3. Only one of these three directives must be defined. Effective when RX channel count is 4 or more.
	NUM_RX3_LANE_2	
	NUM_RX3_LANE_4	
	NUM_RX4_LANE_1	Number of lanes in RX channel 4. Only one of these three directives must be defined. Effective when RX channel count is 5 or more.
	NUM_RX4_LANE_2	
	NUM_RX4_LANE_4	
	NUM_RX5_LANE_1	Number of lanes in RX channel 4. Only one of these three directives must be defined. Effective when RX channel count is 6 or more.
	NUM_RX5_LANE_2	
	NUM_RX5_LANE_4	
	NUM_RX6_LANE_1	Number of lanes in RX channel 4. Only one of these three directives must be defined. Effective when RX channel count is 7 or more.
	NUM_RX6_LANE_2	
	NUM_RX6_LANE_4	
	NUM_RX7_LANE_1	Number of lanes in RX channel 4. Only one of these three directives must be defined. Effective when RX channel count is 8.
	NUM_RX7_LANE_2	
	NUM_RX7_LANE_4	
RX D-PHY Clock Gear	RX0_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations of Hard D-PHY.
	RX0_GEAR_16	
	RX1_GEAR_8	Only one of these directives must be selected. Gear 16 can be used

Category	Directive	Remarks
	RX1_GEAR_16	for only 1- and 2-lane configurations for Hard D-PHY.
	RX2_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX2_GEAR_16	
	RX3_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX3_GEAR_16	
	RX4_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX4_GEAR_16	
	RX5_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX5_GEAR_16	
	RX6_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX6_GEAR_16	
	RX7_GEAR_8	Only one of these directives must be selected. Gear 16 can be used for only 1- and 2-lane configurations for Hard D-PHY.
	RX7_GEAR_16	
RX Hard D-PHY channel	RX0_DPHY_HARD	Specify RX channel that uses Hard D-PHY. Only one of these eight directives must be defined. If none of these is defined, all RX channels use Soft D-PHY.
	RX1_DPHY_HARD	
	RX2_DPHY_HARD	
	RX3_DPHY_HARD	
	RX4_DPHY_HARD	
	RX5_DPHY_HARD	
	RX6_DPHY_HARD	
	RX7_DPHY_HARD	
RX D-PHY Clock Mode[1]	RX0_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 0. Only one of these two directives must be defined.
	RX0_CLK_MODE_HS_LP	
	RX1_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 1. Only one of these two directives must be defined.
	RX1_CLK_MODE_HS_LP	
	RX2_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 2. Only one of these two directives must be defined.
	RX2_CLK_MODE_HS_LP	
	RX3_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 3. Only one of these two directives must be defined.
	RX3_CLK_MODE_HS_LP	
	RX4_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 4. Only one of these two directives must be defined.
	RX4_CLK_MODE_HS_LP	
	RX5_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 5. Only one of these two directives must be defined.
	RX5_CLK_MODE_HS_LP	
	RX6_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 6. Only one of these two directives must be defined.
	RX6_CLK_MODE_HS_LP	
	RX7_CLK_MODE_HS_ONLY	RX D-PHY Clock mode on channel 7. Only one of these two directives must be defined.
	RX7_CLK_MODE_HS_LP	
VC ID pass through[2]	RX0_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX1_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX2_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX3_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX4_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX5_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX6_VC_PASS_THROUGH	Enable when VC ID is passed through without change.
	RX7_VC_PASS_THROUGH	Enable when VC ID is passed through without change.

Category	Directive	Remarks
RX channel VC value[3]	RX0_NEW_VC {value}	Channel 0 Virtual Channel ID; 0 – 15
	RX1_NEW_VC {value}	Channel 1 Virtual Channel ID; 0 – 15
	RX2_NEW_VC {value}	Channel 2 Virtual Channel ID; 0 – 15
	RX3_NEW_VC {value}	Channel 3 Virtual Channel ID; 0 – 15
	RX4_NEW_VC {value}	Channel 4 Virtual Channel ID; 0 – 15
	RX5_NEW_VC {value}	Channel 5 Virtual Channel ID; 0 – 15
	RX6_NEW_VC {value}	Channel 6 Virtual Channel ID; 0 – 15
	RX7_NEW_VC {value}	Channel 7 Virtual Channel ID; 0 – 15
RX Channel Frame Counter[4]	RX0_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX1_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX2_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX3_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX4_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX5_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX6_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
	RX7_FRAME_COUNT	Enable when Data Field of Frame Start/End Short Packet is replaced with the Frame Counter value.
Maximum value of the frame counter	RX0_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX0_FRAME_COUNT is defined. Must be 2 – 65535.
	RX1_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX1_FRAME_COUNT is defined. Must be 2 – 65535.
	RX2_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX2_FRAME_COUNT is defined. Must be 2 – 65535.
	RX3_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX3_FRAME_COUNT is defined. Must be 2 – 65535.
	RX4_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX4_FRAME_COUNT is defined. Must be 2 – 65535.
	RX5_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX4_FRAME_COUNT is defined. Must be 2 – 65535.
	RX6_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX4_FRAME_COUNT is defined. Must be 2 – 65535.
	RX7_FRAME_COUNT_MAX {value}	Frame counter value goes back to 1 after it reaches this value. Effective when RX4_FRAME_COUNT is defined. Must be 2 – 65535.
RX Buffer Depth[5]	RX0_BUFFER_DEPTH_*	RX channel 0 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX1_BUFFER_DEPTH_*	RX channel 1 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX2_BUFFER_DEPTH_*	RX channel 2 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX3_BUFFER_DEPTH_*	RX channel 3 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX4_BUFFER_DEPTH_*	RX channel 4 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX5_BUFFER_DEPTH_*	RX channel 5 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX6_BUFFER_DEPTH_*	RX channel 6 FIFO Depth. * must be 512, 1024, 2048, or 4096.
	RX7_BUFFER_DEPTH_*	RX channel 7 FIFO Depth. * must be 512, 1024, 2048, or 4096.

Category	Directive	Remarks
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	
TX D-PHY Clock Mode	TX_CLK_MODE_HS_ONLY	TX D-PHY Clock mode. Only one of these two directives must be defined.
	TX_CLK_MODE_HS_LP	

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 for corresponding RX channel can be generated internally or directly fed from I/O pin.
2. You cannot assign the same VC value on different RX channels when this is defined.
3. Incoming VC values on RX CSI-2 data are overwritten by these VC values when RX*_VC_PASS_THROUGH is not defined. Values 4 and above are only supported by CSI-2 version 2.0 and above. If the opponent device supports only CSI-2 version 1.1, VC values of 4-15 should not be used.
4. When this is defined, the Data Field is replaced with the 16-bit counter value which begins with 1 after power on/reset and goes back to 1 after it reaches RX*_FRAME_COUNT_MAX. The Data Field is passed through when this is not defined.
5. This value affects necessary EBR used in the device. Number of necessary EBR per RX channel is
   (BUFFER_DEPTH/512) × 2 (in case of NUM_TX_LANE_4 and TX_GEAR_16), or
   (BUFFER_DEPTH/512) × 1 (others).
   Total number of EBR used in this design must not exceed 84 for LIFCL-40 and 24 for LIFCL-17.

2.2. Simulation Directives

Table 2.2 shows the simulation directives for this reference design.

Table 2.2. Simulation Directives

Category	Directive	Remarks
Reference clock period	REF_CLK_PERIOD {value}	Reference clock period in ps
RX D-PHY clock period	RX0_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 0 in ps
	RX1_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 1 in ps
	RX2_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 2 in ps
	RX3_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 3 in ps
	RX4_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 4 in ps
	RX5_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 5 in ps
	RX6_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 6 in ps
	RX7_DPHY_CLK_PERIOD {value}	RX DPHY clock period on Channel 7 in ps
RX Byte clock frequency	RX0_FREQ_TGT {value}	RX byte clock frequency on Channel 0 in MHz
	RX1_FREQ_TGT {value}	RX byte clock frequency on Channel 1 in MHz
	RX2_FREQ_TGT {value}	RX byte clock frequency on Channel 2 in MHz
	RX3_FREQ_TGT {value}	RX byte clock frequency on Channel 3 in MHz
	RX4_FREQ_TGT {value}	RX byte clock frequency on Channel 4 in MHz
	RX5_FREQ_TGT {value}	RX byte clock frequency on Channel 5 in MHz
	RX6_FREQ_TGT {value}	RX byte clock frequency on Channel 6 in MHz
	RX7_FREQ_TGT {value}	RX byte clock frequency on Channel 7 in MHz
TX byte clock frequency	TX_FREQ_TGT {value}	TX byte clock frequency in MHz
TX D-PHY clock period	TX_DPHY_CLK_PERIOD {value}	TX DPHY clock period in ps
Frame Start Detection	TX_WAIT_LESS_15MS	Always enable this directive.
VC value on RX channel	VC_CH0 {value}	VC value on incoming RX Channel 0; 0 – 15
	VC_CH1 {value}	VC value on incoming RX Channel 1; 0 – 15
	VC_CH2 {value}	VC value on incoming RX Channel 2; 0 – 15
	VC_CH3 {value}	VC value on incoming RX Channel 3; 0 – 15
	VC_CH4 {value}	VC value on incoming RX Channel 4; 0 – 15
	VC_CH5 {value}	VC value on incoming RX Channel 5; 0 – 15
	VC_CH6 {value}	VC value on incoming RX Channel 6; 0 – 15
	VC_CH7 {value}	VC value on incoming RX Channel 7; 0 – 15
Frame Number in Frame Start/End Short Packets	CH0_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH1_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH2_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH3_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH4_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH5_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH6_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
	CH7_FNUM_EMBED	Frame number is embedded in Data Field of Frame Start/End Short packet. The value “0” is filled when undefined.
Maximum value of Frame Number	CH0_FNUM_MAX {value}	The maximum value of the frame number; 2 - 65535
	CH1_FNUM_MAX {value}	The maximum value of the frame number; 2 – 65535

Category	Directive	Remarks
	CH2_FNUM_MAX {value}	The maximum value of the frame number; 2 – 65535
	CH3_FNUM_MAX {value}	The maximum value of the frame number; 2 – 65535
	CH4_FNUM_MAX {value}	The maximum value of the frame number; 2 - 65535
	CH5_FNUM_MAX {value}	The maximum value of the frame number; 2 - 65535
	CH6_FNUM_MAX {value}	The maximum value of the frame number; 2 - 65535
	CH7_FNUM_MAX {value}	The maximum value of the frame number; 2 - 65535
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	CH0_DPHY_LPS_GAP {value}	Gap time on RX Channel 0 in ps
	CH1_DPHY_LPS_GAP {value}	Gap time on RX Channel 1 in ps
	CH2_DPHY_LPS_GAP {value}	Gap time on RX Channel 2 in ps
	CH3_DPHY_LPS_GAP {value}	Gap time on RX Channel 3 in ps
	CH4_DPHY_LPS_GAP {value}	Gap time on RX Channel 4 in ps
	CH5_DPHY_LPS_GAP {value}	Gap time on RX Channel 5 in ps
	CH6_DPHY_LPS_GAP {value}	Gap time on RX Channel 6 in ps
	CH7_DPHY_LPS_GAP {value}	Gap time on RX Channel 7 in ps
Gap (LP) time between Frame End and Frame Start on RX Channel	CH0_DPHY_FRAME_GAP {value}	Gap time on RX Channel 0 in ps
	CH1_DPHY_FRAME_GAP {value}	Gap time on RX Channel 1 in ps
	CH2_DPHY_FRAME_GAP {value}	Gap time on RX Channel 2 in ps
	CH3_DPHY_FRAME_GAP {value}	Gap time on RX Channel 3 in ps
	CH4_DPHY_FRAME_GAP {value}	Gap time on RX Channel 4 in ps
	CH5_DPHY_FRAME_GAP {value}	Gap time on RX Channel 4 in ps
	CH6_DPHY_FRAME_GAP {value}	Gap time on RX Channel 4 in ps
	CH7_DPHY_FRAME_GAP {value}	Gap time on RX Channel 4 in ps
Video data configuration on RX Channel 0	CH0_NUM_FRAMES {value}	Number of frames to feed
	CH0_NUM_LINES {value}	Number of active lines per frame
	CH0_NUM_PIXELS {value}	Number of pixels per line
	CH0_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 1	CH1_NUM_PIXELS {value}	Number of pixels per line
	CH1_NUM_LINES {value}	Number of active lines per frame
	CH1_NUM_PIXELS {value}	Number of pixels per line
	CH1_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 2	CH2_NUM_FRAMES {value}	Number of frames to feed
	CH2_NUM_LINES {value}	Number of active lines per frame
	CH2_NUM_PIXELS {value}	Number of pixels per line
	CH2_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 3	CH3_NUM_FRAMES {value}	Number of frames to feed
	CH3_NUM_LINES {value}	Number of active lines per frame
	CH3_NUM_PIXELS {value}	Number of pixels per line

Category	Directive	Remarks
	CH3_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 4	CH4_NUM_FRAMES {value}	Number of frames to feed
	CH4_NUM_LINES {value}	Number of active lines per frame
	CH4_NUM_PIXELS {value}	Number of pixels per line
	CH4_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 5	CH5_NUM_FRAMES {value}	Number of frames to feed
	CH5_NUM_LINES {value}	Number of active lines per frame
	CH5_NUM_PIXELS {value}	Number of pixels per line
	CH5_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 6	CH6_NUM_FRAMES {value}	Number of frames to feed
	CH6_NUM_LINES {value}	Number of active lines per frame
	CH6_NUM_PIXELS {value}	Number of pixels per line
	CH6_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Video data configuration on RX Channel 7	CH7_NUM_FRAMES {value}	Number of frames to feed
	CH7_NUM_LINES {value}	Number of active lines per frame
	CH7_NUM_PIXELS {value}	Number of pixels per line
	CH7_DT_*	Data Type; * must be RAW8, RAW10, RAW12, RAW14, RGB888, YUV420_10, YUV420_8, YUV422_10, or YUV422_8.
Internal signal monitoring	MISC_ON	Enables internal signal 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 VC Aggregation Top Level I/O

Port Name	Direction	Description
Clocks and Resets		
ref_clk_i (optional)	I	Input reference clock. This port is declared only when EXT_REF_CLK is defined in synthesis_directives.v.
reset_n_i	I	Asynchronous active low system reset
CSI-2 RX Interface		
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)
rx3_d2_n_i	I	Negative differential RX Ch3 D-PHY input data 2 (in case of 4 lane configuration)

Port Name	Direction	Description
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)
CSI72 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

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

- rx0*_unit (* is 0 - 7)
  - rx_dphy_0* (* is 0 - 7)
  - csi2_parser
  - rx_buffer
- tdm_ctrl
- tx_dphy_if
- tx_dphy
- int_osc
- int_gpll

The top-level design has a reset synchronization logic. In addition, GPLL may be added if necessary according to RX and TX configurations.

3.1. rx0*_unit

This module is instantiated for each RX channel as a wrapper module to include rx_dphy, csi2_parser, and rx_buffer modules.

3.1.1. rx_dphy_0*

This module must be created for each RX channel according to channel conditions, such as the number of lanes, bandwidth, and others. Figure 3.1 shows an example of IP interface settings in Lattice Radiant for the CSI-2/DSI D-PHY Receiver Submodule IP. Create a separate rx_dphy IP for each RX channel even though their configurations are same. Refer to CSI/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081) for details.

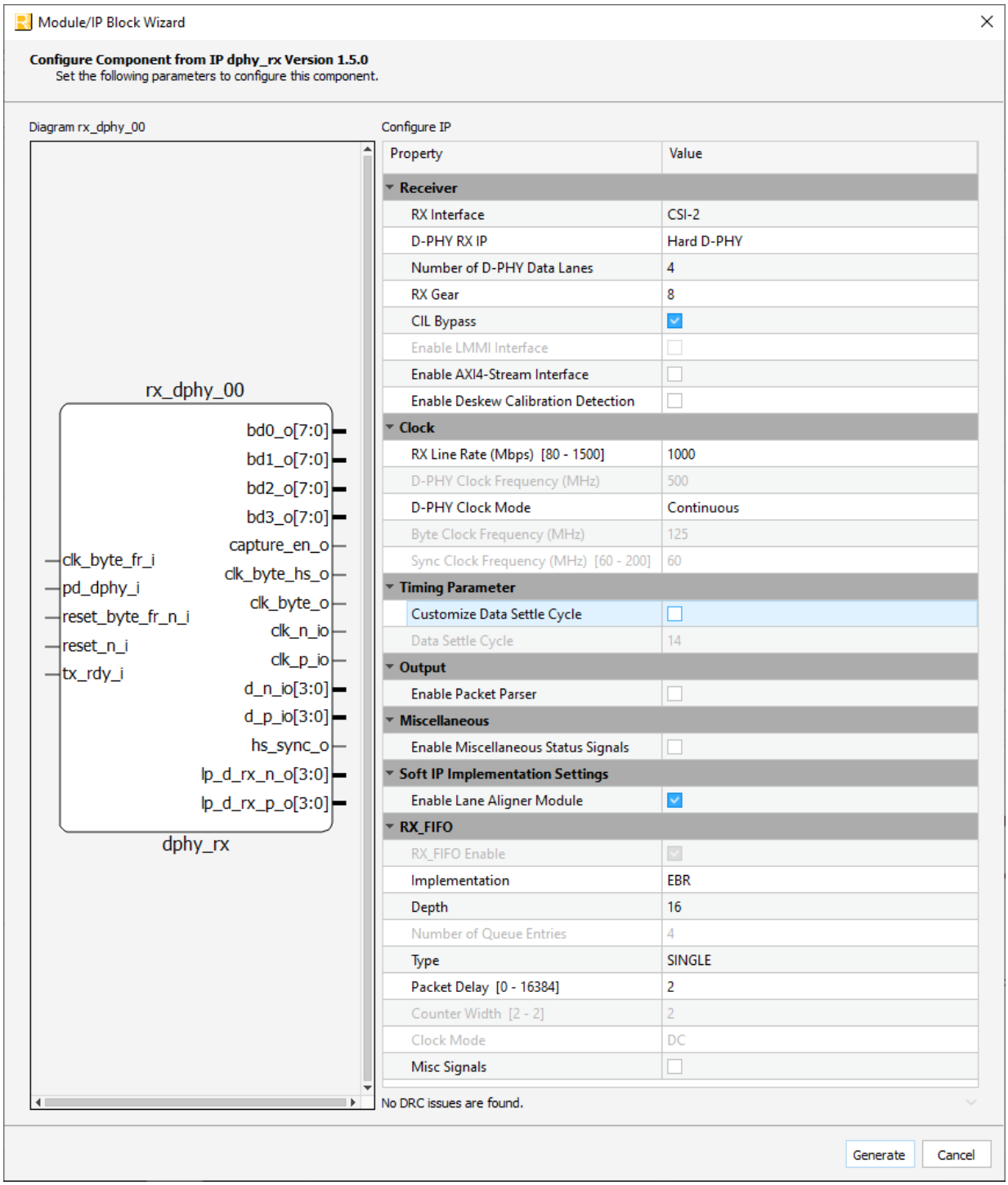

Figure 3.1. rx_dphy_0* IP Creation in Lattice Radiant

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

- RX Interface – Select CSI-2.
- D-PHY RX 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.
- 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 with Gear 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. 1500 is the maximum for Soft D-PHY and Hard D-PHY with 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).
- Enable Packet Parser – Select Disabled.
- Enable Miscellaneous Status Signals – Select Disabled (unchecked).
- Enable Lane Aligner Module – Select Enabled for Soft D-PHY with 2-lane and 4-lane configurations.
- Lane Aligner FIFO Depth – Select 8.
- FIFO_TYPE – Select LUT.
- RX_FIFO Enable – Select Disabled (unchecked) for continuous clock mode with Soft D-PHY, select Enabled for all other cases.
- Implementation – Select LUT for continuous clock mode with Hard D-PHY, refer to for continuous clock mode with Hard D-PHY, refer to RX FIFO section for non-continuous clock mode.
- Depth – Select 16 for continuous clock mode with Hard D-PHY, refer to RX FIFO section for non-continuous clock mode.
- Type – Select SINGLE for continuous clock mode with Hard D-PHY, refer to RX FIFO section for non-continuous clock mode.
- Packet Delay – Set 1 for continuous clock mode with Hard D-PHY, refer to RX FIFO section for non-continuous clock mode.
- Clock Mode – Select DC (dual clock).
- Misc Signals – Select Disabled (unchecked).

This module takes serial CSI-2 data and outputs byte data after de-serialization in CSI-2 High Speed mode. It is recommended to set the design name to *rx_dphy_00, rx_dphy_01, … , rx_dphy_07* so that you do not need to modify the instance names of these IPs in rx_0*_unit.v as well as the simulation setup file. Otherwise, you have to modify the names accordingly.

RX Gear 16 is supported for Hard D-PHY with 1-lane and 2-lane configurations.

3.1.1.1. RX FIFO

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

Hard D-PHY in Continuous Clock Mode

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

Soft D-PHY in Continuous Clock Mode

In this case, RX FIFO is not necessary and RX_FIFO Enable should be disabled.

Non-Continuous Clock Mode

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

- Continuous byte clock = non-continuous byte clock

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

- Continuous byte clock < non-continuous byte clock

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

- Continuous byte clock > non-continuous byte clock

  There are two options in this case:

  - Use type = SINGLE with large Packet Delay

    Set the Depth large enough to contain the necessary data to avoid RX FIFO underflow after FIFO read begins after the time specified by Packet Delay. In general, Packet Delay must be set close the depth of the RX FIFO. This configuration can be used when we have enough time interval between the last active line and the frame end short packet so that the frame end short packet is not written to RX FIFO while it still contains the last active line of video data.
  - Use type = QUEUE with Number of Queue Entries = 2

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

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

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

3.1.2. csi2_parser

This module is instantiated for each RX channel to handle CSI-2 protocol decoding and VC overwrite. Upon receiving byte data from rx_dphy_0*, csi2_parser detects short and long packet headers. When RX*_VC_PASS_THROUGH is not defined, it replaces VC values with the value specified by RX*_NEW_VC and calculates ECC value based on this new VC value and other packet data. As a result, VC value and ECC are replaced with new values and sent to the next module (rx_buffer). Since VC values of 4 and above are only supported by CSI-2 version 2.x, using 0 to 3 would be safe in case of 2- to 4-channel aggregation. The downstream device must support CSI-2 version 2.x in case that VC = 4 or above is used. On the other hand, packet header data including VC values are not changed and passed through when RX*_VC_PASS_THROUGH is defined.

Figure 3.2 shows short packet capture and a new VC assignment when RX*VC_PASS_THROUGH is not defined. The original VC=0 is replaced with VC=1 along with new ECC value calculated applying the new VC. 16-bit Data Field ({bd2_i, bd1_i}) is replaced with the frame counter value if RX*_FRAME_COUNT is defined. If not, the Data Field is passed through.

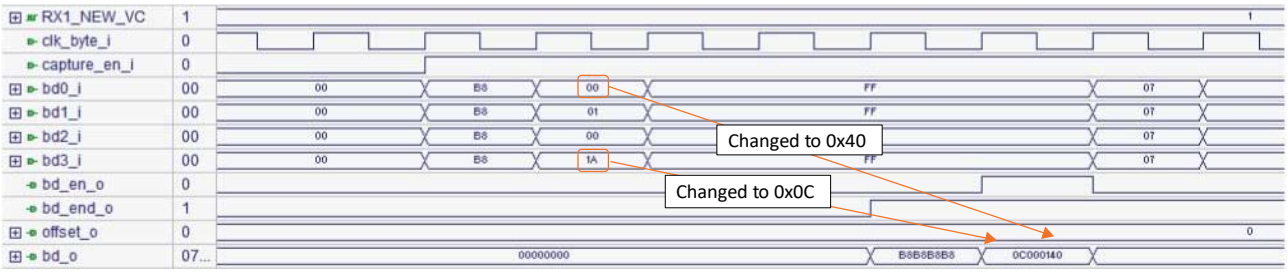

Figure 3.2. Short Packet Detection and VC Replacement

It is possible to do 16:1 aggregation using three CrossLink-NX devices with two CrossLink-NX devices in the first stage, both taking 8 RX channels and the third CrossLink-NX device takes the output of two CrossLink-NX to do 2:1 aggregation as the second stage. In this case RX*_VC_PASS_THROUGH should be defined in the second stage CrossLink-NX since each first stage TX output contains 8 channel data aggregated.
This module also detects the end of the current High Speed transmission by detecting the trailer byte and asserts the end flag sent to rx_buffer along with the offset info. Offset is a 3-bit data that indicates the number of remaining bytes in the final data transmission. This is modulo of 8 considering 64-bit transmission in case of four lanes with Gear 16 on TX side. Figure 3.3 and Figure 3.4 show the beginning and the end of long packet capture and transfer.

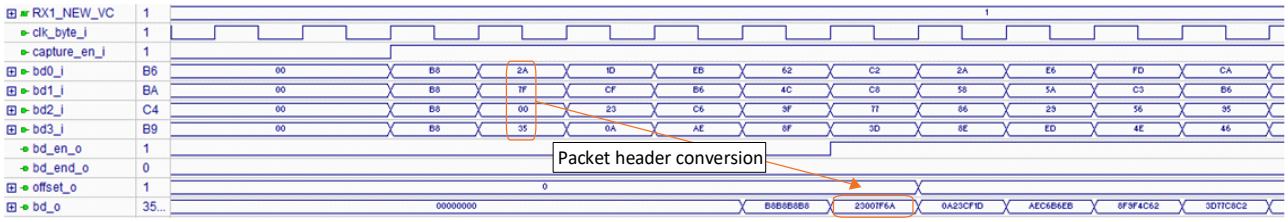

Figure 3.3. Long Packet Detection and VC Replacement

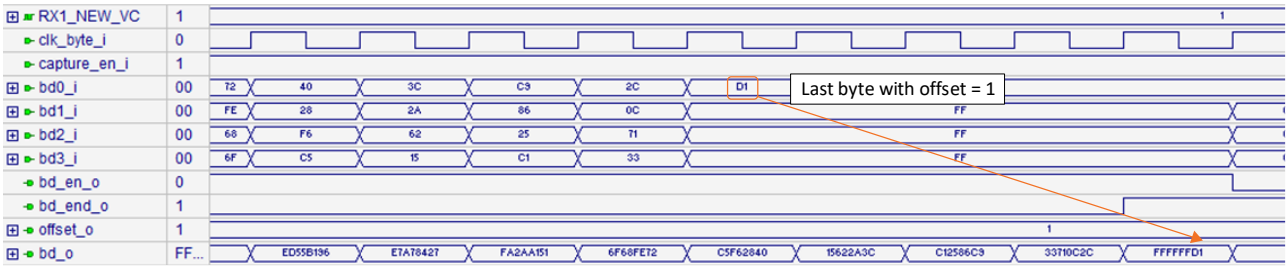

Figure 3.4. End of Long Packet with Trailer Bytes

3.1.3. rx_buffer

This module is instantiated for each RX channel to store HS transmission data to be read out by tdm_ctrl. Default buffer depth is 512, which means the buffer can store 512 × 64 bytes (NUM_TX_LANE_4 and TX_GEAR_16) or 512 × 32 bytes (others). 512 × 64 bytes FIFO requires two EBRs, therefore at least 16 EBRs are required for 8:1 aggregation with 4-lane TX and TX Gear 16 configuration. Buffer depth of 1024 or deeper is possible as long as the total number of EBR does not exceed 84 for LIFCL-40 and 24 for LIFCL-17. Data size of one HS transmission must not exceed FIFO size (for example, 1920 pixels of RAW10 makes 1920 × 5/4 = 2400 bytes, which is less than 1024 × 32, but more than 512 × 32). In case the data transfer to tdm_ctrl of the RX channel in question is in a cue and needs to wait for the completion of other channel data transmission, the FIFO has to begin storing the next line data. Therefore, it is safer to make the FIFO depth cover close to two HS transmission data. The first FIFO data is read out by this module itself with ready flag assertion. The FIFO has extra data width to contain data end flag and offset data. This FIFO is a boundary between RX byte clock (write side) and TX byte clock (read side) domain.

Figure 3.5 shows an example of short packet write transaction. An extra data write happens after the completion of every HS data transmission. This dummy write is necessary to ease the FIFO read timing to quit HS data read when it

gets the end data flag. tdm_ctrl can stop fifo_re_i assertion in the next cycles when it sees bd_end_o =1. This module automatically reads the first data of each HS transmission. Along with ready flag (bd_rdy_o) assertion, it notifies tdm_ctrl that HS data is ready to be acquired from this channel. In this example, the offset is 4 since TX Gear is 16, which means 8 bytes are passed to tdm_ctrl in a single read cycle.

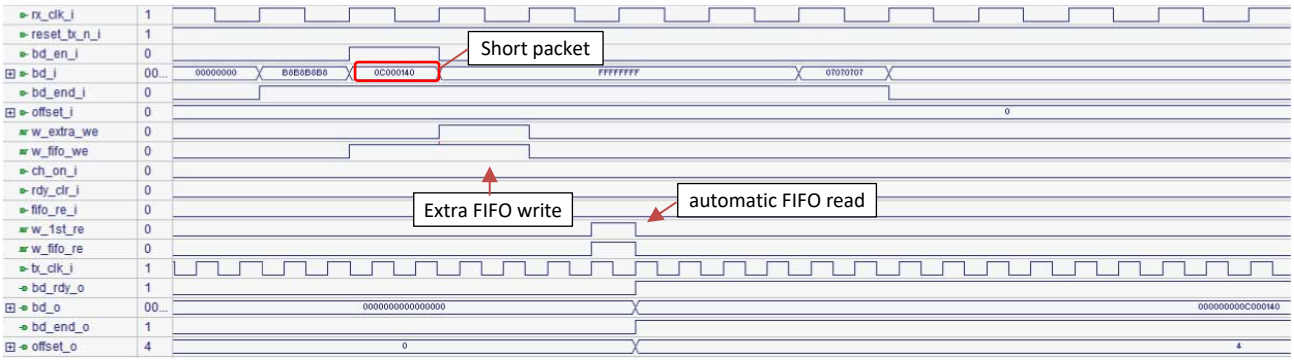

Figure 3.5. Short Packet Write

Figure 3.6 and Figure 3.7 show the beginning and end of long packet write transaction. In case of TX Gear 16, write happens every two cycles to form 64-bit byte data. Upon the detection of bd_end_i, an extra write enable is generated to write dummy data followed by the automatic first data read with bd_rdy_o assertion.

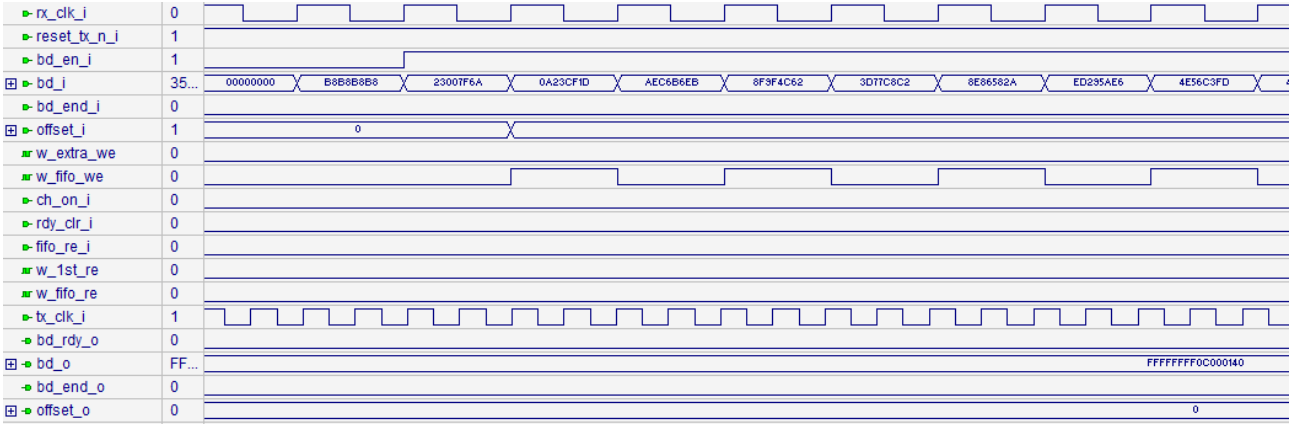

Figure 3.6. Beginning of Long Packet Write

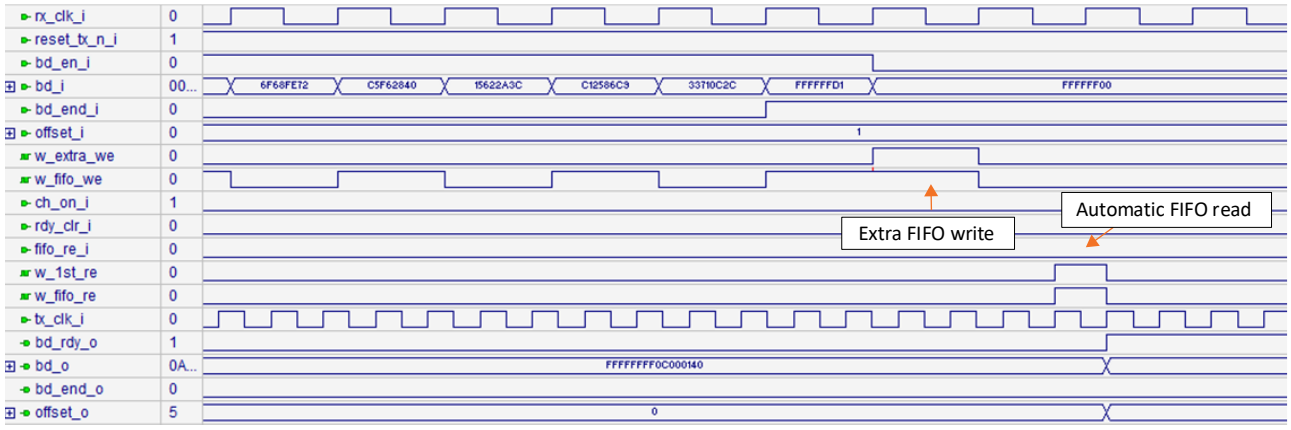

Figure 3.7. End of Long Packet Write

Figure 3.8 shows short packet read transaction. rdy_clr_i comes from tdm_ctrl to clear bd_rdy_o. The valid data (0x0c000140) is already on bd_o when fifo_re_i is asserted due to the automatic first data read. Therefore, this fifo_re_i reads an extra dummy data and discarded by tdm_ctrl. bd_end_o is used to terminate fifo_re_i by tdm_ctrl.

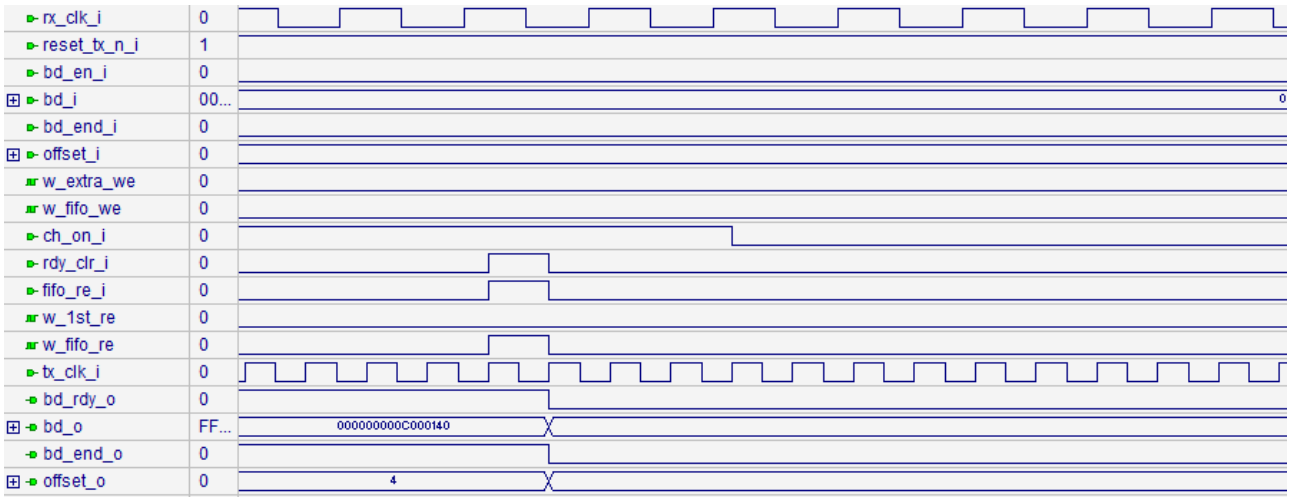

Figure 3.8. Short Packet Read

Figure 3.9 shows end of long packet read. Upon the detection of bd_end_o, fifo_re_i is de-asserted. In this example, the last data is “0xD133710C2C” since offset_o is 5.

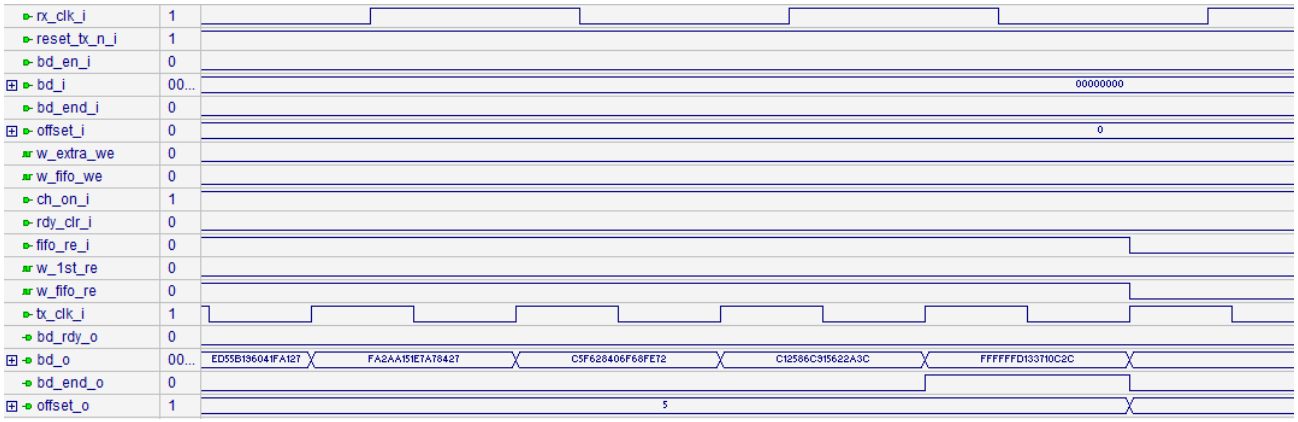

Figure 3.9. End of Long Packet Read

3.2. tdm_ctrl

This module monitors the ready flags of the RX channels and takes the HS transmission data of the selected channel. TX D-PHY data lane goes into LP mode after every HS data transfer of a single RX channel and come back to HS mode when the next HS transmission begins with the new RX channel data. The channel selection is based on a round robin fashion when multiple RX channel data are available signified by the assertion of the ready flags. Incoming data width is either 32 or 64, but the outgoing data width is adjusted to TX Lane count * TX_GEAR, which means 8, 16, 32, or 64. In case of 8 or 16, FIFO read enable is asserted once every 4 or 2 TX byte clock cycles. When the data end flag is detected, the trailer bytes are appended. Beginning of trailer byte within 16/32/64 bytes is judged by offset data obtained from rx_buffer when the end flag is detected.

Figure 3.10 shows an example of a 4-channel aggregation global sequence of tdm_ctrl. tdm_ctrl monitors all ready signals (rx0_rdy_i, … rx3_rdy_i) and decides which RX channel data is aggregated for the next transmission. In this example, channel 0 is selected since ready flag is asserted only on channel 0. tdm_bgn_i is a trigger signal to begin reading FIFO data from rx_buffer. rx0_rdy_clr_o is asserted to clear rx0_rdy_i and rx0_fifo_re_o is asserted to begin

reading FIFO data from rx0_buffer. Read data are sent to the next module (tx_dphy_if) for transmission. Other channel data are also transferred in the same manner.

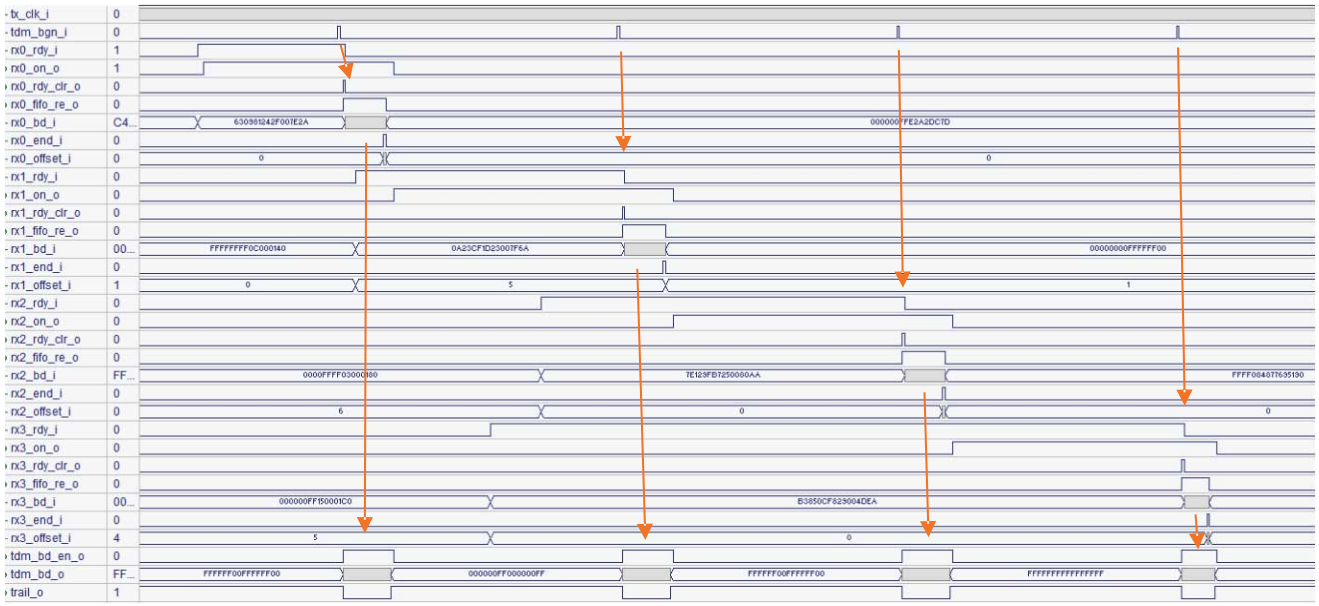

Figure 3.10. Global Sequence of tdm_ctrl

Trailer byte appending is another important feature of tdm_ctrl in addition to TDM of RX channel data. Figure 3.11 shows an example. The last valid data is 0x0C07D0 (rx3_offset_i = 3 when rx3_end_i = 1). Trailer bytes against these are 0xFFFF00. MSByte of the previous data is 0x59 and trailer byte against this is 0xFF. Therefore, 0xFFFFFF00FF is appended as trailer bytes, which appears as upper 5 bytes with the last valid data (0x0C07D0) followed by two sets of 0xFFFFFF00 along with trail_o assertion.

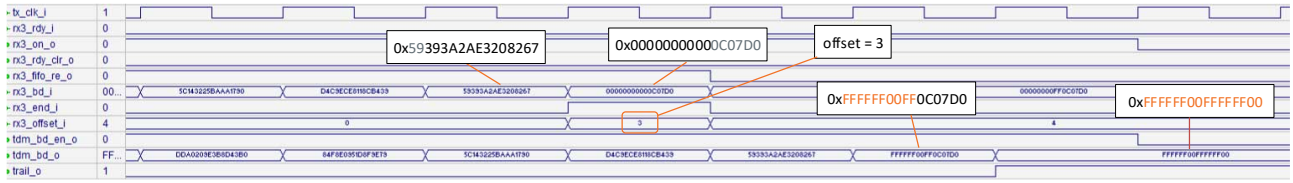

Figure 3.11. Trailer Byte Appending

3.3. tx_dphy_if

This module reallocates byte data coming from tdm_ctrl and sends to tx_dphy according to TX lane and TX Gear value. In addition, it monitors ready flag from rx_buffer modules as well as ready flag from tx_dphy to control tdm_bgn_o that triggers TDM operation in tdm_ctrl. Figure 3.12 shows global operation of tx_dphy_if including tx_dphy signals. tx_dphy_if asserts clk_hs_en_o and d_hs_en_o to make both clock and data lane from LP mode to HS mode. After that d_hs_rdy_i is asserted by tx_dphy. Confirming at least one of ready flags from rx_buffer is asserted, tdm_bgn_o is asserted to let tdm_ctrl begins a new HS data transmission.

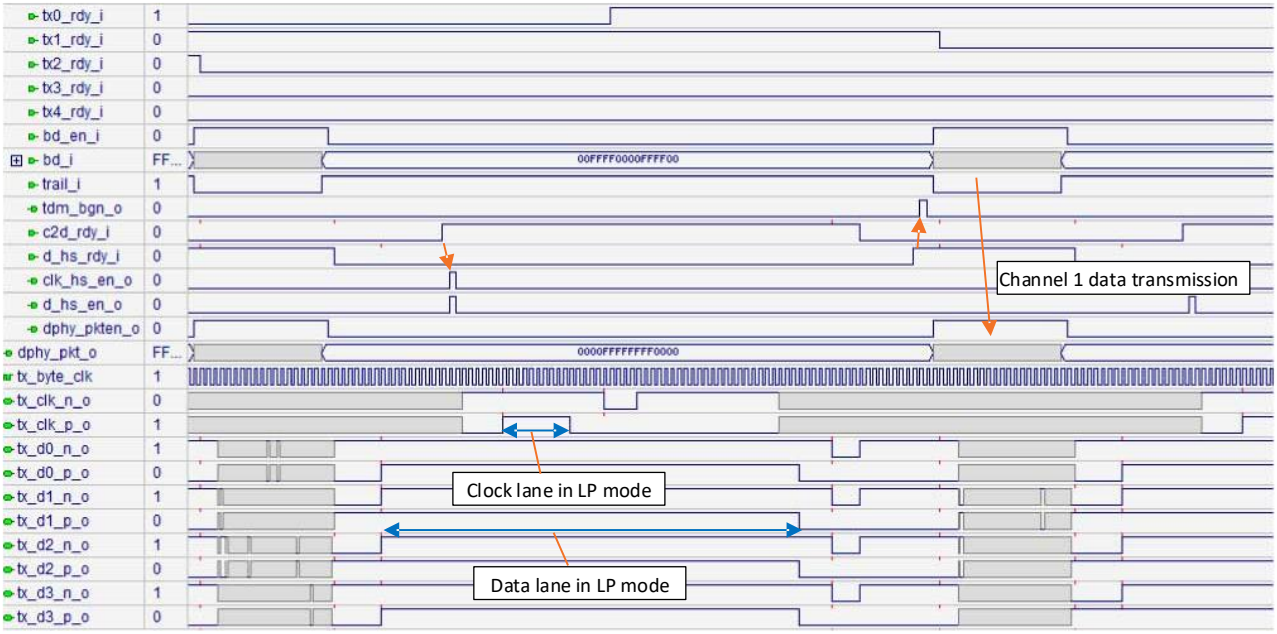

Figure 3.12. Global Operation of tx_dphy_if

3.4. tx_dphy

You must create this module according to channel conditions, such as number of lanes, bandwidth, and others. Figure 3.13 shows an example IP interface setting in Lattice Radiant for the CSI-2/DSI D-PHY Transmitter Submodule IP. Refer to CSI/DSI DPHY Tx IP Core User Guide (FPGA-IPUG-02080) for details.

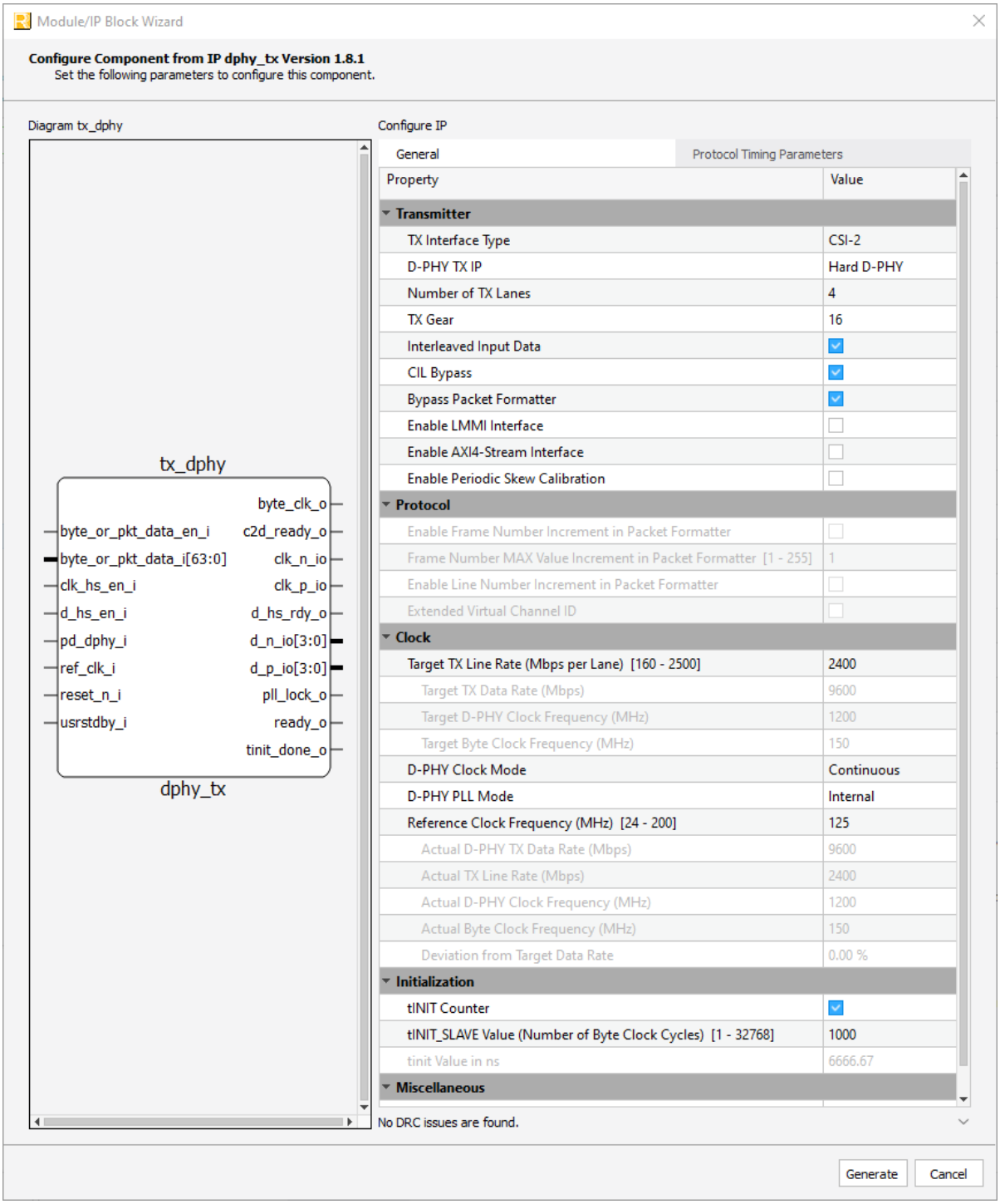

Figure 3.13. tx_dphy IP Creation in Lattice Radiant

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

- TX Interface – Select CSI-2.
- DPHY TX IP – Select Hard DPHY.
- Number of TX Lanes – Set according to channel configuration. Must match NUM_TX_LANE_* setting.
- TX Gear – Automatically set according to TX Line Rate.
- CIL Bypass – Select checkbox to enable.
- Bypass packet formatter – Select checkbox to enable.

- Enable LMMI Interface – De-select checkbox to disable.
- Enable AXI4-Stream Interface – De-select checkbox to disable.
- TX Line Rate per Lane – Set according to channel configuration. Total bandwidth must be fast enough to aggregate all RX channel data with extra overhead for the transitions between HS and LP modes.
- DPHY Clock Mode – Set according to channel configuration. Select Continuous when the TX bandwidth margin is small. Must match TX*_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- DPHY PLL Mode – Select Internal.
- Reference Clock Frequency – Set the appropriate value which can be obtained from ref_clk_i pin, one of continuous rx*_byte_clk, or on-chip GPLL. This clock frequency must in the range of 20 MHz – 200 MHz.
- tINIT counter – Enabled (checked) is recommended.
- tINIT_SLAVE Value – 1000 (default) 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_dphy* so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

TX Gear 8 is automatically selected by Lattice Radiant when the lane bandwidth is less than 1500 Mbps, which means TX byte clock could be ~187.5 MHz.

TX Line Rate setting is one of the key factors in this RD. Calculating the required TX lane bandwidth is derived from the following equation:

$$TX\_lane\_bandwidth = \frac{1}{m}\sum_{k=0}^{n-1} number\_of\_lanes\_in\_RX[k] \ * lane\_bandwidth\_in\_RX[k]$$

where m is number of TX lanes and n is number of RX channels.

Example: 8 to 1 aggregation with the following configuration

RX0: 4 lanes, 1000 Mbps / lane,

RX1: 4 lanes, 360 Mbps / lane,

RX2: 2 lanes, 480 Mbps / lane,

RX3: 1 lanes, 440 Mbps / lane,

RX4: 1 lane, 240 Mbps / lane,

RX5: 1 lane, 1480 Mbps / lane,

RX6: 2 lanes, 320 Mbps / lane,

RX7: 2 lanes, 280 Mbps / lane,

TX: 4 lanes

Then TX lane bandwidth = (4 x 1000 + 4 x 360 + 2 x 480 + 1 x 440 + 1 x 240 + 1 x 1480 + 2 x 320 + 2 x 280) / 4 = 2.44 Gbps.

The above configuration will not work if RX channels do not have enough length of blanking (LP) periods. The TX bandwidth is fast enough to cover all of RX channel HS data transmissions, but transition time going back and forth between HS and LP mode cannot be proportionally shrunk by increasing TX bandwidth. The guideline of minimum blanking period per active line is (500 ns x number of RX channels) if the above equation is used to get TX lane bandwidth, assuming continuous clock (HS_ONLY mode) is set in TX D-PHY. In case of non-continuous clock (HS_LP mode), the required blanking period is doubled. TX lane bandwidth has to be raised from the above value in case that RX channel cannot have enough blanking periods. Increasing RX FIFO size could be an alternative option in some cases, but most likely this is not as effective.

You should monitor FIFO overflow flags of all rx_buffer modules (rx*_fifo_full) instantiated and increase the TX bandwidth if any of rx_buffer FIFO overflow happens.

3.5. int_osc

This module generates ~75 MHz clock used for sync clock and rx*_clk_lp_ctrl. sync clock is used by Soft D-PHY RX IP and rx*_clk_lp_ctrl is used by all RX D-PHY IP, which are in non-continuous clock mode. If continuous clock above 60 MHz is available, you will be able use that without instantiating this module.

3.6. int_gpll

This module generates continuous byte clock(s) used for RX channels in non-continuous clock mode. If all RX channels are in continuous clock mode or continuous byte clock(s) for RX channels in non-continuous clock mode are already available, this module is not necessary.

In case of non-continuous clock (HS_LP) mode in RX, each RX channel requires two clock trees (non-continuous RX byte clock and continuous RX byte clock) in CrossLink-NX device. Since CrossLink has 32 primary clock trees, it is possible to use HS_LP mode in all RX channels when appropriate continuous byte clocks are obtained for all RX channels. Therefore, you have to make sure that the continuous clock can be obtained to set the non-continuous clock mode in RX D-PHY on RX channels. The following are possible candidates of the continuous clock:

- GPLL outputs driven by the external reference clock or a continuous byte clock from other channels
- Continuous byte clock from other channels (either directly or after divider)
- Continuous clock fed from I/O pin (either directly or after divider)

The sample design (csi2_aggr_vc_NX.v) assumes that RX channel #0 is in HS_ONLY mode and other RX channels are in HS_LP mode. In this example, the continuous byte clock for RX channel #1-7 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 rx*_clk_byte_fr (continuous byte clock for RX channel *), 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.

```
////////////////////////////////////////////////////////////////////////////////
///// 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 range of 20-200 MHz.                         /////
////////////////////////////////////////////////////////////////////////////////

wire gpll_refclk;

`ifdef EXT_REF_CLK
       assign tx_refclk = ref_clk_i;
       assign gpll_refclk = ref_clk_i;
`else
       assign tx_refclk = rx0_clk_byte_fr; // could be from other channels
       assign gpll_refclk = rx0_clk_byte_fr; // could be from other channels
`endif

////////////////////////////////////////////////////////////////////////////////
///// below is just a reference only and to be modified according to the   /////
///// customer's application                                             //////
////////////////////////////////////////////////////////////////////////////////

//assign gpll_lock = 1;       // when GPLL is not in use


//// CrossLink-NX GPLL
int_gpll pll_inst (
       .rstn_i        (reset_n_i),
       .clki_i        (gpll_refclk),
```

```
        .clkop_o        (gpll_clk_op),
        .clkos_o        (gpll_clk_os),
        .clkos2_o       (gpll_clk_os2),
        .clkos3_o       (gpll_clk_os3),
        .clkos4_o       (gpll_clk_os4),
        .lock_o         (gpll_lock)
);
```

On 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 continuous RX byte clocks are not appropriate to generate the desired clock for TX D-PHY. The clock to TX D-PHY must be continuous and in the range of 24 – 200 MHz.

4. Design and File Modifications by User

This RD is based on version 1.0.1 of the RX D-PHY IP and version 1.0.1 of the TX D-PHY IP. 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 GPLL setting in the current top-level file (csi2_aggr_vc_NX.v) is based on the configuration shown in Figure 1.2 and Figure 1.3. According to the configuration you use, you have to modify the GPLL instantiation as described in int_gpll section. Also, you can remove int_osc module if the appropriate sync clock and r*_clk_lp_ctrl are already available.

In addition, an instance name of TX D-PHY (tx_dphy) has to be modified if you create this IP with a different name.

4.2. rx0*_unit

An instance name of RX D-PHY (rx_dphy_0*) has to be modified if you create this IP with a different name.

5. Design Simulation

The script file (csi2_aggr_vc_NX_fsim.do) and testbench files are provided to run the functional simulation by Active HDL. If you follow the naming recommendations regarding design name and 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:

- Lattice Radiant installation directory path
- User project directory
- Remove line(s) if int_osc and/or int_gpll are not in use

```
### Set Radiant installation directory ###
set radiant_dir C:/lscc/radiant/2022.1

### Set Customer's simulation directory ###
set sim_dir C:/Users/destose/Pictures/1_Projects/MIPI_Regression_Final/csi2_aggr_vc_NX_RD1_0_pkg_working/simulation/lifcl

cd $sim_dir
```

Lattice Radiant directory

Project directory

Figure 5.1. Script Modification #1

```
vmap work work

vmap ovi_lifcl $radiant_dir/modeltech/lib/ovi_lifcl
vmap pmi_work $radiant_dir/modeltech/lib/pmi_work
#./../../int_osc/rtl/int_osc.v \

vlog -mfcu \
+incdir+"./../../source/verilog/lifcl"+"./"+"./../../testbench/verilog" ./../../testbench/verilog/csi2_aggr_vc_NX_tb.v \
-y ./../../testbench/verilog +libext+.v \
./../../source/verilog/lifcl/synthesis_directives.v \
./../../source/verilog/lifcl/csi2_aggr_vc_NX.v \
./../../source/verilog/lifcl/rx00_unit.v \
./../../source/verilog/lifcl/rx01_unit.v \
./../../source/verilog/lifcl/rx02_unit.v \
./../../source/verilog/lifcl/rx03_unit.v \
./../../source/verilog/lifcl/rx04_unit.v \
./../../source/verilog/lifcl/rx05_unit.v \
./../../source/verilog/lifcl/rx06_unit.v \
./../../source/verilog/lifcl/rx07_unit.v \
./../../source/verilog/lifcl/csi2_parser_encrypt.v \
./../../source/verilog/lifcl/rx_buffer_encrypt.v \
./../../source/verilog/lifcl/tdm_ctrl_encrypt.v \
./../../source/verilog/lifcl/tx_dphy_if.v \
./../../int_osc/rtl/int_osc.v \
./../../int_gpll/rtl/int_gpll.v \
./../../rx_dphy_00/rtl/rx_dphy_00.v \
./../../rx_dphy_01/rtl/rx_dphy_01.v \
./../../rx_dphy_02/rtl/rx_dphy_02.v \
./../../rx_dphy_03/rtl/rx_dphy_03.v \
./../../rx_dphy_04/rtl/rx_dphy_04.v \
./../../rx_dphy_05/rtl/rx_dphy_05.v \
./../../rx_dphy_06/rtl/rx_dphy_06.v \
./../../rx_dphy_07/rtl/rx_dphy_07.v \
./../../tx_dphy/rtl/tx_dphy.v \

vsim -t 10fs -voptargs="+acc=rn" -L work top -L pmi_work -L ovi_lifcl

#vsim +access +r top -L work top -L pmi_work -L ovi_lifcl
#add wave /top/dut/*
do wave.do
run -all
```

Comment out if not used

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 VC and ECC data modifications. It shows following statements while running and doing data comparison:

KERNEL: [31697536470][DPHY_CHK] payload data = f4 f4 db b8 --- Data matches ch1 : f4 db b8

KERNEL: [31697856470][DPHY_CHK] payload data = 70 38 05 70 --- Data matches ch1 : 70 38 05 70

KERNEL: 31698150000 DPHY CH 0 Driving Data

KERNEL: 31698150000 DPHY CH 0 Lane 0 : Driving with data = ac

KERNEL: 31698150000 DPHY CH 0 Lane 1 : Driving with data = ef

KERNEL: 31698150000 DPHY CH 0 Lane 2 : Driving with data = 6b

KERNEL: 31698150000 DPHY CH 0 Lane 3 : Driving with data = e3

KERNEL: 31698160600 DPHY CH 4 Driving Data

KERNEL: 31698160600 DPHY CH 4 Lane 0 : Driving with data = f4

KERNEL: [31698176470][DPHY_CHK] payload data = 28 3f 6c 04 --- Data matches ch1 : 28 3f 6c 04

KERNEL: [31698496470][DPHY_CHK] payload data = 17 2a c9 bd --- Data matches ch1 : 17 2a c9 bd

KERNEL: 31698504400 DPHY CH 3 Driving Data

KERNEL: 31698504400 DPHY CH 3 Lane 0 : Driving with data = ce

KERNEL: [31698816470][DPHY_CHK] payload data = 2d 57 60 7e --- Data matches ch1 : 2d 57 60 7e

KERNEL: 31698950000 DPHY CH 0 Driving Data

KERNEL: 31698950000 DPHY CH 0 Lane 0 : Driving with data = db

KERNEL: 31698950000 DPHY CH 0 Lane 1 : Driving with data = d9

KERNEL: 31698950000 DPHY CH 0 Lane 2 : Driving with data = c2

KERNEL: 31698950000 DPHY CH 0 Lane 3 : Driving with data = b0

KERNEL: [31699136470][DPHY_CHK] payload data = 8c b3 e6 26 --- Data matches ch1 : 8c b3 e6 26

…

Below is an example of the statements at the end of simulation:

…

KERNEL: 2225141273100 DPHY 7 Lane 0 : Driving stop

KERNEL: 2225141273100 DPHY 7 Lane 1 : Driving stop

KERNEL: 2225166519400 DPHY CH 7 CLK : Driving CLK-Trail

KERNEL: 2225172519400 DPHY CH 7 CLK : Driving CLK-Stop

KERNEL: [2225448056470][DPHY_TX_HS_LP_CHK] LP11 to LP01 Transition on D0 lane

KERNEL: [2225454456470][DPHY_TX_HS_LP_CHK] LP01 to LP00 Transition on D0 lane with LP01 period = 64 ns

KERNEL: [2225460216470][DPHY_TX_HS_LP_CHK] LP00 to HS00 Transition on D0 lane with LP00 period = 57 ns

KERNEL: [2225479396470][DPHY_TX_HS_LP_CHK] HS00 to HS Transition on D0 lane with LP00+HS00 period = 249 ns

KERNEL: [2225479936470][DPHY_CHK] Short Packet detected : Data type = 01

KERNEL: Check the Packet Header to identify incoming RX channel --- c1 06 00 f4 matches CH #7 Data

```
# KERNEL: [2225489016470][DPHY_TX_HS_LP_CHK] HS to LP11 Transition on D0 lane with HS-TRAIL period = 91 ns
# KERNEL:       2227172707200 DPHY CH 7 CLK CONT : Driving CLK-Trail
# KERNEL:       2227178519400 TEST END
# KERNEL:
# KERNEL: CH #0 (VC = 0) : 2700 / 2700 HS transmission completed successfully
# KERNEL: CH #1 (VC = 1) : 608 / 608 HS transmission completed successfully
# KERNEL: CH #2 (VC = 2) : 1160 / 1160 HS transmission completed successfully
# KERNEL: CH #3 (VC = 3) : 630 / 630 HS transmission completed successfully
# KERNEL: CH #4 (VC = 5) : 156 / 156 HS transmission completed successfully
# KERNEL: CH #5 (VC = 9) : 1054 / 1054 HS transmission completed successfully
# KERNEL: CH #6 (VC = 11) : 1554 / 1554 HS transmission completed successfully
# KERNEL: CH #7 (VC = 15) : 660 / 660 HS transmission completed successfully
# KERNEL:
# KERNEL: ### Simulation Completed ###
```

One HS transmission is either Frame Start/End short packet or long packet of one active video line. The result would be fine if the numerator is equal to denominator in the statements.

You should set small values in CH*_NUM_LINES and CH*_NUM_PIXELS directives in simulation_directives.v file, especially in the first simulation trial to minimize simulation time. On the other hand, it makes sense to set the actual value to CH*_NUM_PIXELS directives and CH*_DPHY_LPS_GAP directives when TX bandwidth cannot have extra margin against total RX bandwidth. By setting realistic values, FIFO overflow could be detected when the margin is not large enough.

Figure 5.3 shows an example of 4-channel aggregation. tx*_bd_rdy = 1 indicates the waiting period from the completion of one HS transmission data write (to rx_buffer) to the start of FIFO read by tdm_ctrl. This waiting period gets longer when the channel in question is in a cue to be read by tdm_ctrl.

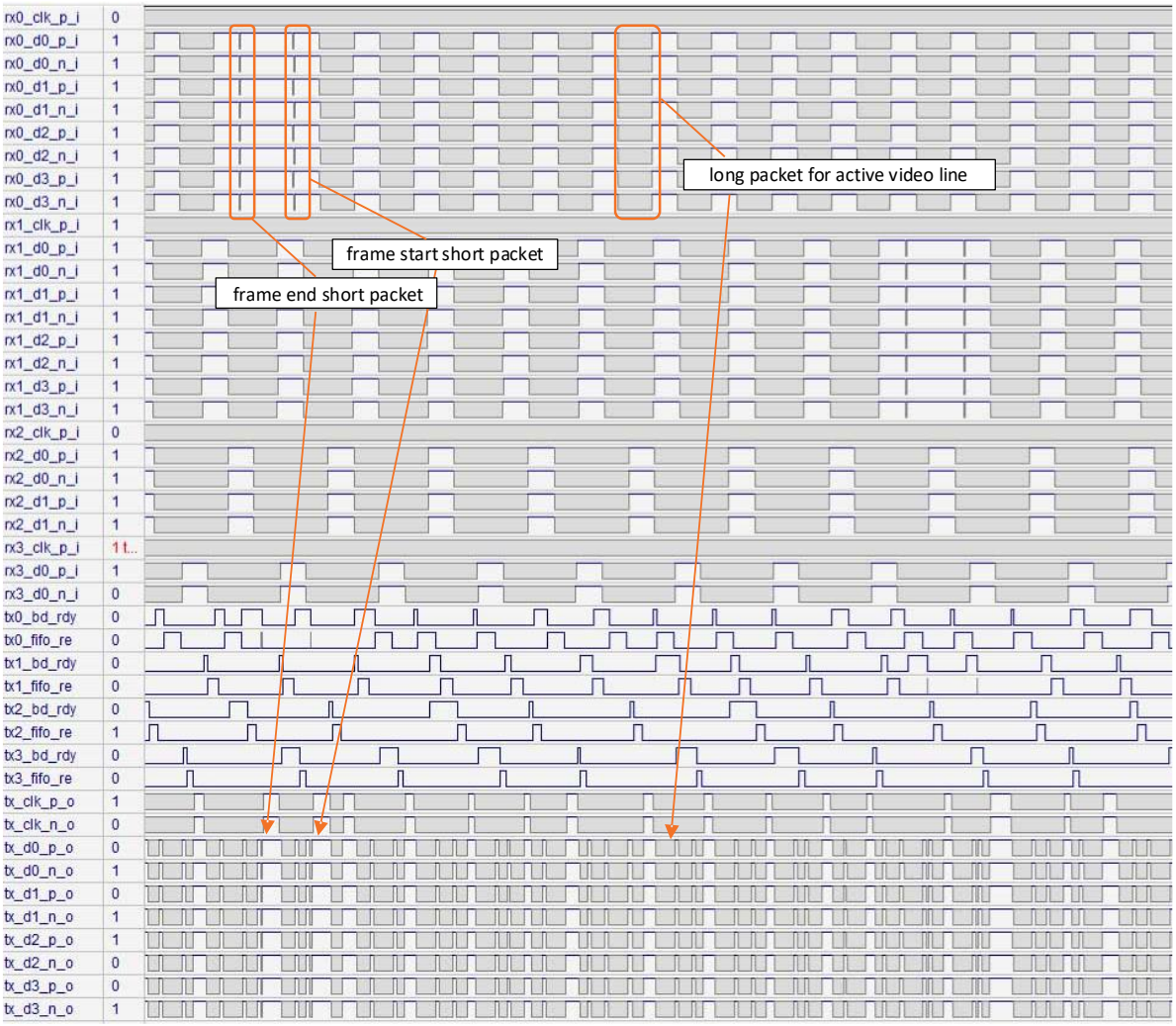

Figure 5.3. Functional Simulation Example

Figure 5.4 shows an example of FIFO overflow in rx_buffer. tx0_bd_rdy = 1 period get long and that leads to FIFO overflow on channel 0. Simulation automatically stops when FIFO overflow happens.

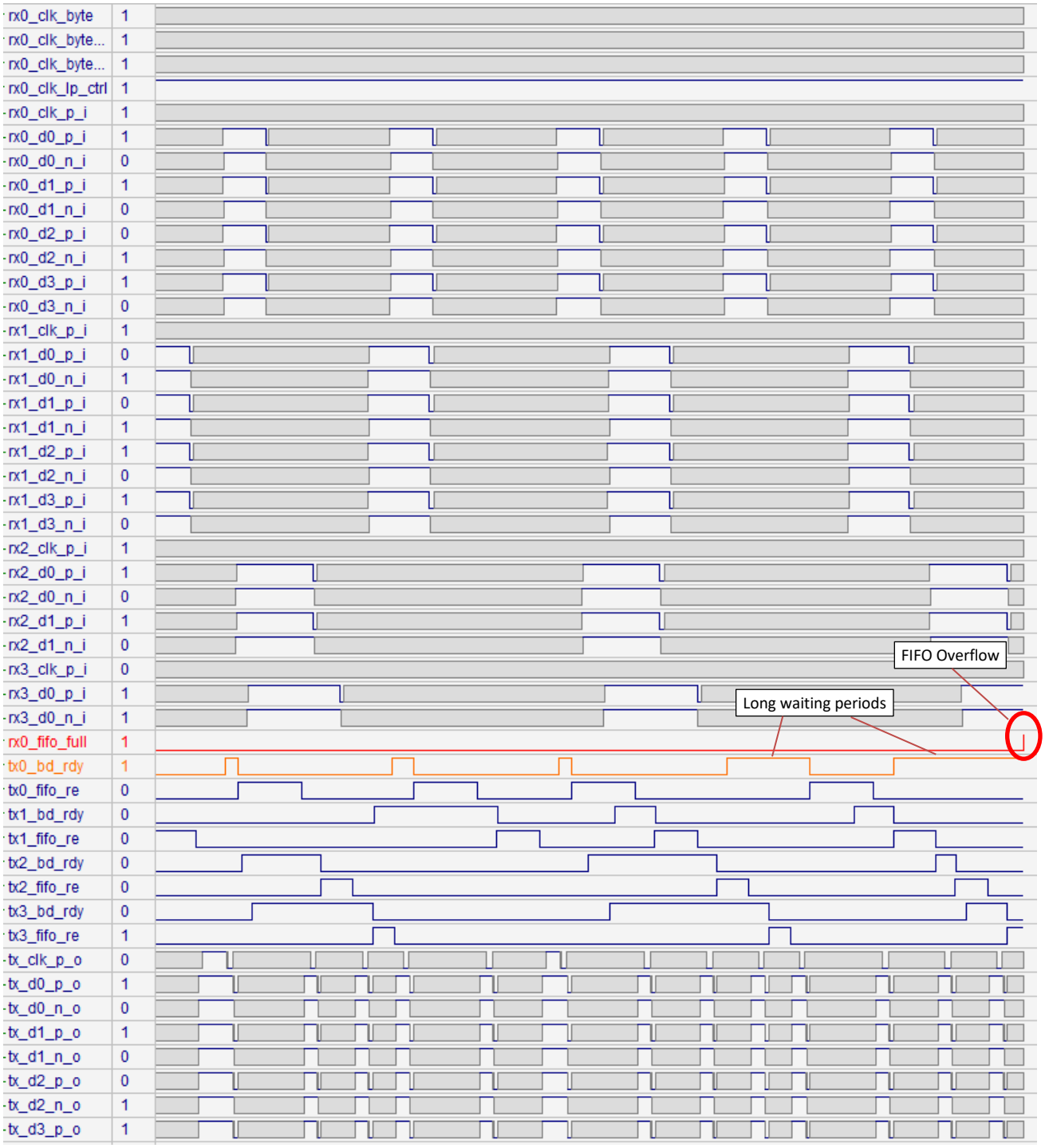

Figure 5.4. FIFO Overflow

6. Known Limitations

Following are the limitations of this reference design:

- RX Gear 16 with 4-lane configuration is not supported.
- TX Gear cannot be set manually (refer to the tx_dphy section).

7. Design Package and Project Setup

MIPI CSI-2 Virtual Channel Aggregation with CrossLink-NX Reference Design is available on www.latticesemi.com. Figure 7.1 shows the directory structure. The design is targeted for LIFCL-40CABGA400. synthesis_directives.v and simulation_directives.v are set to configure eight RX channels as an example shown below:

- RX CH #0 : 4 lanes with Hard D-PHY in continuous clock mode
- RX CH #1 : 4 lanes with Soft D-PHY in non-continuous clock mode
- RX CH #2 : 2 lanes with Soft D-PHY in non-continuous clock mode
- RX CH #3 : 1 lane with Soft D-PHY in non-continuous clock mode
- RX CH #4 : 1 lane with Soft D-PHY in non-continuous clock mode
- RX CH #5 : 1 lane with Soft D-PHY in non-continuous clock mode
- RX CH #6 : 2 lanes with Soft D-PHY in non-continuous clock mode
- RX CH #7 : 2 lanes with Soft D-PHY in non-continuous clock mode
- TX : 4 lanes with Gear 16

You can modify the directives for own configuration.

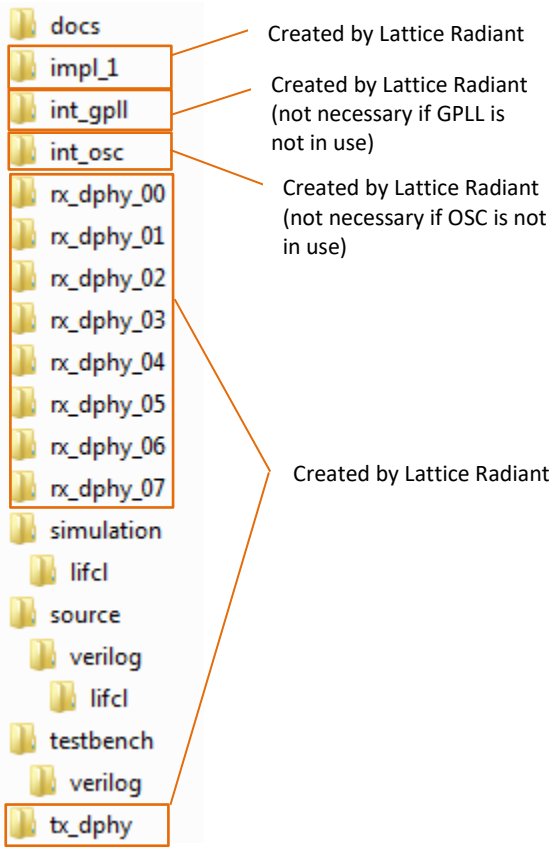

Figure 7.1. Directory Structure

Figure 7.2 shows design files used in the Lattice Radiant project. Including PLL and oscillator modules, Lattice Radiant creates 11 .ipx files. By specifying csi2_aggr_vc_NX as a top-level design, all unnecessary files are ignored.

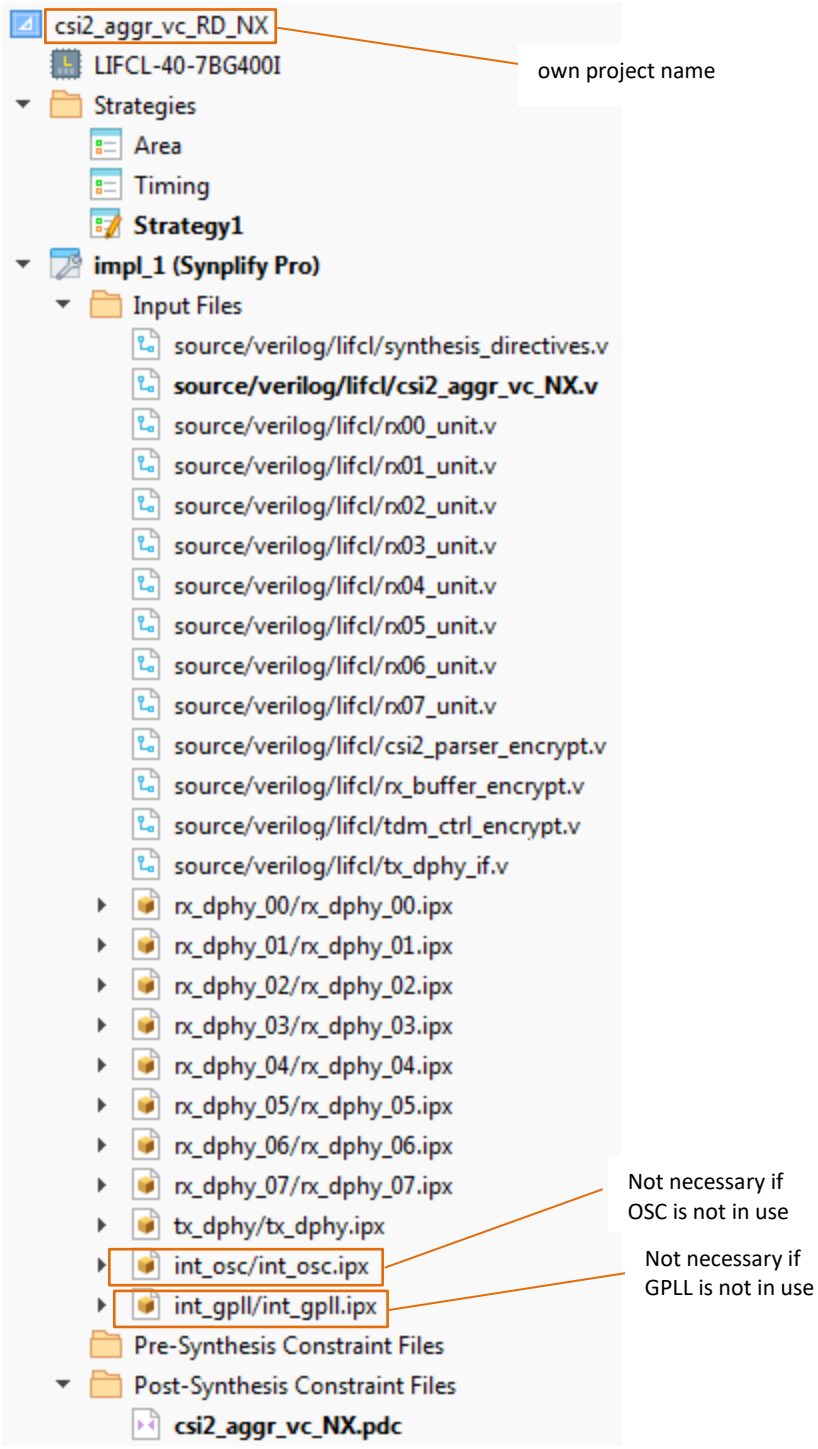

Figure 7.2. Project Files

8. Resource Utilization

Resource utilization depends on the configurations. Table 8.1 shows resource utilization examples under certain configurations targeting LIFCL-40. 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 8.1. Resource Utilization Examples

Configuration	LUT %	FF %	EBR	I/O
8 RX channels with 4/4/2/1/1/1/2/2 lanes, TX 4 lanes with Gear 16	9396	6684	49	61
6 RX channels with 4/4/2/1/1/1 lanes, TX 4 lanes with Gear 16	7157	5147	39	49
4 RX channels with 4/4/2/1 lanes, TX 4 lanes with Gear 16	5224	3770	28	41
2 RX channels with 4/4 lanes, TX 4 lanes with Gear 8	2900	2176	15	31

References

For more information refer to:

- CrossLink-NX Web page
- Lattice Radiant FPGA design software
- Lattice Radiant User Guide
- CSI/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081)
- CSI/DSI DPHY Tx IP Core User Guide (FPGA-IPUG-02080)
- MIPI® Alliance Specification for D-PHY Version 1.1
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.1
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 2.0
- Lattice Insights 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/en/Support/AnswerDatabase.

Revision History

Revision 1.1, December 2023

Section	Change Summary
Disclaimer	Updated this section.
Supported Device and IP	• Updated D-PHY Receiver IP version to 1.5.0. • Updated D-PHY Transmitter IP version to 1.8.1. • Updated Lattice Radiant Software version to 2022.1.
Introduction	Updated Figure 1.3. Clocking Scheme.
Design and Module Description	• Updated image of Figure 3.1. • Updated image of Figure 3.13.
Design Simulation	• Updated image of Figure 5.1. • Updated image of Figure 5.2.
References	Updated section contents.

Revision 1.0, February 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com