SubLVDS to MIPI CSI-2 Image Sensor Bridge with CertusPro-NX

Reference Design

FPGA-RD-02250-1.0

March 2022

Disclaimers

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

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AP	Application Processor
CMOS	Complementary Metal Oxide Semiconductor
CSI-2	Camera Serial Interface 2
DDR	Double Data Rate
EAV	End of Active Video
FV	Frame Valid
GPLL	General Purpose PLL
HS	High Speed
I2C	Inter-Integrated Circuit
IP	Intellectual Property
ISP	Image Signal Processor
LP	Low Power
LV	Line Valid
LVDS	Low Voltage Differential Signal
MIPI	Mobile Industry Processor Interface
OSCI	Internal Oscillator
PLL	Phase Locked Loop
RD	Reference Design
RX	Receiver
SAV	Start of Active Video
TX	Transmitter
WC	Word Count
XHS	Horizontal Sync Pulse
XVS	Vertical Sync Pulse

1. Introduction

Many Image Signal Processors (ISP) or Application Processors (AP) use the Mobile Industry Processor Interface (MIPI®) Camera Serial Interface 2 (CSI-2) standard for image sensor inputs. However, some high-resolution CMOS image sensors use a proprietary SubLVDS output format.

The Lattice Semiconductor SubLVDS to MIPI CSI-2 Image Sensor Bridge reference design for CertusPro-NX™ devices solves the mismatch between SubLVDS output image sensor and an ISP/AP using CSI-2 interface.

1.1. Supported Device, IP and Software

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

Device Family	Part Number	Compatible IP	Lattice Radiant version
CertusPro-NX	LFCPNX-100	SubLVDS Receiver IP version 1.1.1 Pixel-to-Byte Converter IP version 1.3.0 D-PHY Transmitter IP version 1.4.0	Lattice Radiant software version 3.0 and above

1.2. Features

- Supports 4-, 6-, 8-, or 10-lane SubLVDS input to 1-, 2-, or 4-lane MIPI CSI-2 output
- Supports input lane bandwidth of up to 1.25 Gbps and output lane bandwidth of up to 1.5 Gbps
- Image cropping option
- VSYNC and HSYNC can be generated to control sensor timing
- Dynamic parameter setting through I2C

1.3. Block Diagram and Clock Distribution

Figure 1.1 shows the block level diagram of the SubLVDS to MIPI CSI-2 Image Sensor Bridge reference design. It contains three major IPs and interfacing modules between them. Image data from the sensor come in along with the SubLVDS clock in double data rate (DDR) fashion. This clock is divided by 4 or 8 to generate pixel clock according to RX Gear. Pixel clock is fed to trim_ctrl and Pixel-to-Byte IP modules. On the other hand, pixel clock is fed to TX D-PHY IP as a reference clock and TX D-PHY IP creates MIPI clock using its internal PLL. MIPI clock is divided by 8 to generate byte clock and byte clock is fed to Pixel-to-Byte IP and lane_ctrl module. In some configurations, the data bus going to TX D-PHY is half of the data bus coming out from Pixel-to-Byte IP. In that case, the byte clock generated by TX D-PHY must be 2x of the byte clock used in Pixel-to-Byte IP and the original byte clock (hs_byte_clk) is divided by two and fed to Pixel-to-Byte IP and lane_ctrl. The lane_ctrl module takes care of bus width difference. When the image sensor is in slave mode, FPGA has to feed the sync signals to the sensor. In that case, the Rx SubLVDS module takes the clock from the sensor to generate sync signals. I2C slave module is optional to change the configurations on the fly.

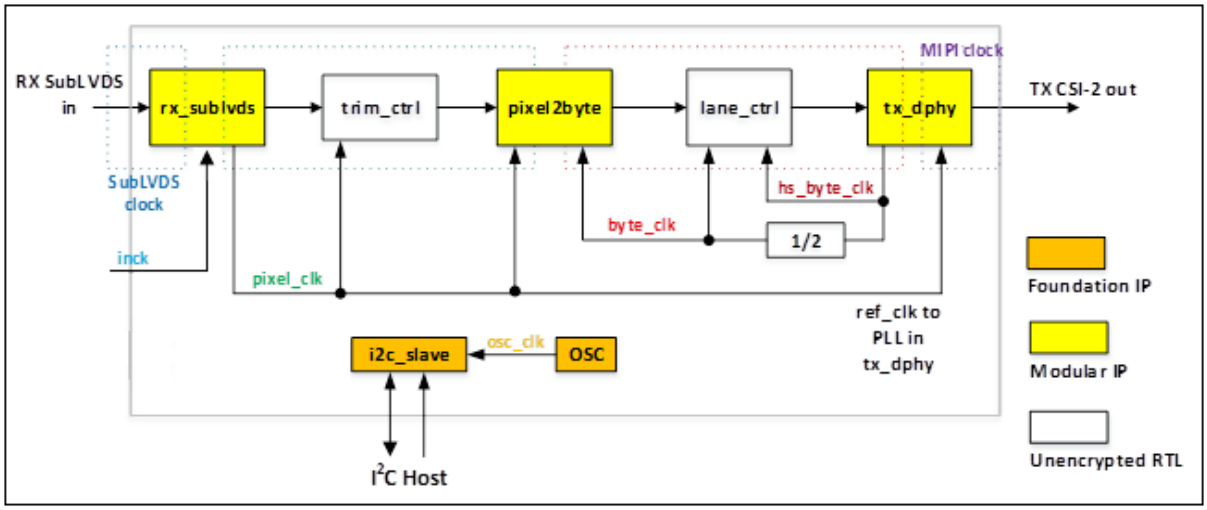


Figure 1.1. SubLVDS to MIPI CSI-2 Image Sensor Bridge Block Diagram

1.4. RX and TX Permutations

Table 1.1 shows the available permutations of RX and TX configurations. Same permutations apply to both RAW10 and RAW12. In addition, Pixel-to-Byte IP supports only 4 lanes on TX. To overcome these limitations, the different TX Gear setting is applied in D-PHY TX IP to handle 1 lane or 2 lane outputs, which require faster byte clock (hs_byte_clk = 2x).

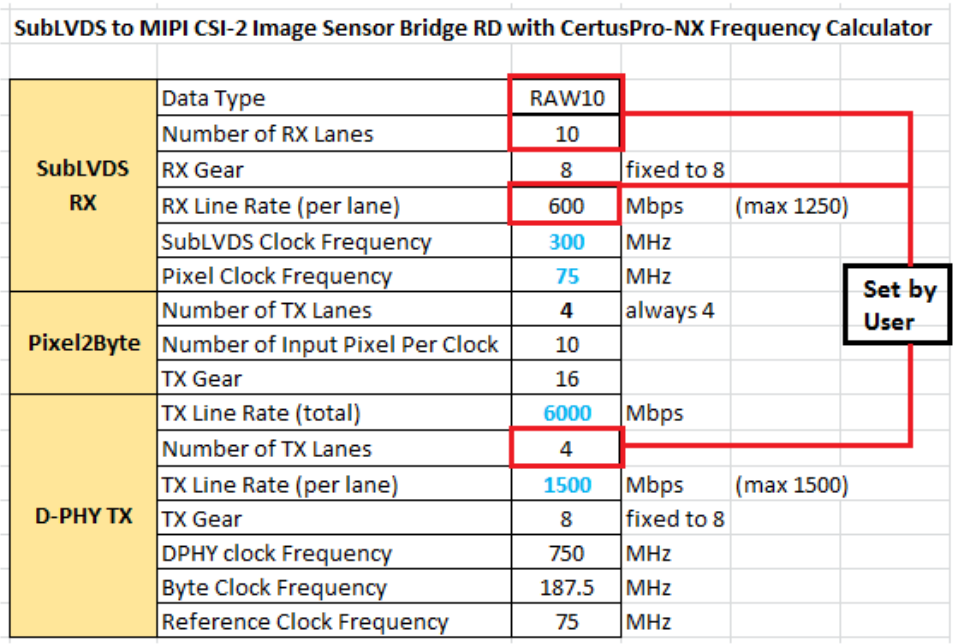
Table 1.1. RX and TX Permutations

Data Type	RX Lane Count	RX Gear	TX Gear Setting in Pixel-to-Byte IP	TX Lane Count	TX Gear	hs_byte_clk Ratio Against byte_clk
RAW10 RAW12	4	8	8	4	8	1
				2		2
	6			4		1
				2		2
	8		16	4		2
	10			4		2

* assumes Maximum TX lane bandwidth by Gear 8 is ~1500 Mbps.

The Excel sheet (*SubLVDS2CSI2_clock_LFCPNX.xlsx*) is provided to calculate the pixel clock, byte clock, and others, from RX bandwidth and other information. This sheet can be useful to configure IPs. A sample entry is shown in Figure 1.2. By setting four rows shown in the table, pixel clock, byte clock, and TX bandwidth are automatically calculated. Those can be used to configure D-PHY TX IP. In the following situations, the byte clock that comes out from D-PHY TX IP is hs_byte_clk and half of this clock is byte_clk fed to Pixel-to-Byte IP (and lane_ctrl):

2 x (*TX Gear setting in Pixel-to-Byte IP*) = (*Number of TX Lanes in D-PHY TX) x (TX Gear in D-PHY TX*).



SubLVDS to MIPI CSI-2 Image Sensor Bridge RD with CertusPro-NX Frequency Calculator

SubLVDS RX	Data Type	RAW10		
	Number of RX Lanes	10		
	RX Gear	8	fixed to 8	
	RX Line Rate (per lane)	600	Mbps	(max 1250)
	SubLVDS Clock Frequency	300	MHz	
	Pixel Clock Frequency	75	MHz	
Pixel2Byte	Number of TX Lanes	4	always 4	
	Number of Input Pixel Per Clock	10		
	TX Gear	16		
D-PHY TX	TX Line Rate (total)	6000	Mbps	
	Number of TX Lanes	4		
	TX Line Rate (per lane)	1500	Mbps	(max 1500)
	TX Gear	8	fixed to 8	
	DPHY clock Frequency	750	MHz	
	Byte Clock Frequency	187.5	MHz	
	Reference Clock Frequency	75	MHz	

Set by User

Figure 1.2. Bandwidth and Clock Frequency Calculator

2. Parameters and Port List

There are two directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Radiant® and 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 SubLVDS RX IP, Byte-to-Pixel IP, and TX D-PHY IP settings created by 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. Some parameter selections are restricted by other parameter settings.

Table 2.1. Synthesis Directives

Category	Directive	Remarks
Image Sensor control1	SENSOR_MODE_MASTER	Use SLAVE when sync signals (xvs_o/xhs_o) must be sent from FPGA to the image sensor. Only one of these two directives must be defined.
	SENSOR_MODE_SLAVE	
Image Sensor Sync Polarity1	SENSOR_SYNC_NEG	Polarity setting for sync signals to the image sensor. Only effective when SENSOR_MODE_SLAVE is defined. Only one of these two directives must be defined.
	SENSOR_SYNC_POS	
XVS (Vertical Sync) assertion period1	XVS_LENGTH_XHS	Select the active pulse length of xvs_o between xhs_o and one horizontal line. Only applicable in case of SENSOR_MODE_MASTER. Only one of these two directives must be defined.
	XVS_LENGTH_LINE	
Total line count1	V_TOTAL {value}	Total line count for one frame including blanking. Only effective when SENSOR_MODE_SLAVE is defined. Value must be 12'd10 – 12'd4095.
Total horizontal cycle1	H_TOTAL {value}	Total cycle count for one line including blanking in the unit of inck_i. Only effective when SENSOR_MODE_SLAVE is defined. Value must be 12'd10 – 12'd4095.
XHS (Horizontal Sync) pulse cycle1	XHS_LENGTH {value}	Active pulse width of xhs_o. Only effective when SENSOR_MODE_SLAVE is defined. Value must be 8'd1 – 8'd255.
RX Data Type	RAW10	Define the data type on RX channel. Only one of these two directives must be defined.
	RAW12	
RX channel lane count	NUM_RX_LANE_4	Number of lanes in RX channel. Only one of these four directives must be defined.
	NUM_RX_LANE_6	
	NUM_RX_LANE_8	
	NUM_RX_LANE_10	
RX SubLVDS Clock Gear	RX_GEAR_8	RX SubLVDS Clock Gear. Always use only Gear 8.
RX SubLVDS Dropped Line Mode	RX_DATA_DROPPED_LINE_MODE	Define only if the RX Dropped Line Mode is Dynamic.
RX SubLVDS Dropped Pixel Mode	RX_DATA_DROPPED_PIXEL_MODE	Define only if the RX Dropped Pixel Mode is Dynamic.
RX SubLVDS Word Count Mode	RX_DATA_WORD_COUNT_MODE	Define only if the RX Word Count Mode is Dynamic.
TX D-PHY Clock mode2	TX_CLK_MODE_HS_LP	TX D-PHY Clock mode. Only one of these two directives must be defined.
	TX_CLK_MODE_HS_ONLY	
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	Number of TX Clock Gear on RX channel. Always use only Tx Gear 8.
Parameter set by I2C	USE_I2C	Define this to use I2C I/F to configure the image sensor from FPGA.

Category	Directive	Remarks
Software Reset Register3	SW_RST_N {value}	Default value of the software reset register of I2C Slave module. Value must be 1'b0 or 1'b1. Applicable only when USE_I2C is defined. Active low.
Top Line Trimming4, 5	TOP_TRIM {value}	Define the number of lines to be trimmed before TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 6'd0 – 6'd63.
Vertical Active Lines on TX5	V_ACTIVE {value}	Define the number of active lines to be sent on TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 12'd1 – 12'd4095.
Left Pixel Unit Trimming6, 8	LEFT_TRIM_UNIT {value}	Define the number of pixel units to be trimmed before TX. 1 pixel unit = number of RX lanes. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 6'd0 – 6'd63.
Left Pixel Trimming6, 8	LEFT_TRIM_LANE {value}	Define the number of pixels to be trimmed before TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 4'd0 – 4'd9 and less than the RX lane count.
Horizontal Active Pixel units on TX7, 8	H_ACTIVE_UNIT {value}	Define the number of active pixels to be sent on TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 10'd1 – 10'd1023. The value must be even in case of TX_GEAR_16 or NUM_TX_CH_2.
Active Word Count9	WC {value}	Define the number of byte count of active pixels to be sent to TX. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 16'd5 – 16'd65535.
Virtual Channel ID	VC {value}	Define the virtual Channel ID. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. Value must be 2'd0 – 2'd3.
KEEP clock lane in HS mode	KEEP_HS	When defined, clock lane is kept in HS mode during active line periods even if TX clock lane is in non-continuous clock mode. This value is used as a fixed value when USE_I2C is not defined and used as the default value of the I2C register when USE_I2C is defined. The value is 1'b1 when defined and 1'b0 when not defined.

Notes:

1. Refer to the image sensor data sheet for proper settings.
2. HS_LP mode means non-continuous clock mode and HS_ONLY means continuous clock mode.
3. Logical OR between this register and system reset (reset_n_i) is used to reset modules other than I2C slave module.
4. Value = 0 is not allowed when no line is trimmed by SubLVDS RX IP.
5. (TOP_TRIM + V_ACTIVE) cannot exceed the vertical active line count of the incoming RX data. It is your responsibility to manage this.
6. Number of pixels trimmed from the left edge is ((LEFT_TRIM_UNIT x number of RX lanes x (RX_GEAR / 8)) + LEFT_TRIM_LANE).
7. Active pixel count sent to TX is (H_ACTIVE_UNIT x number of RX lanes x (RX_GEAR / 8)).
8. ((LEFT_TRIM_UNIT + H_ACTIVE_UNIT) x number of RX lanes x (RX_GEAR / 8) + LEFT_TRIM_LANE) cannot exceed the horizontal active pixel count of the incoming RX data. It is your responsibility to manage this.
9. WC ≤ (H_ACTIVE_UNIT x number of RX lanes x (RX_GEAR / 8)) x (RAW number (10 or 12)) / 8. Refer to the trim_ctrl section for details.

2.2. Simulation Directives

Table 2.2 shows the simulation directives for this reference design. Some parameter selections are restricted by other parameter settings including Table 2.1.

Table 2.2. Simulation Directives

Category	Directive	Remarks
RX SubLVDS clock period	PIX_CLK {value}	RX SubLVDS clock period in ps.
INCK clock period	INCK_PERIOD {value}	INCK clock period in ps. Applicable only when SENSOR_MODE_SLAVE is defined.
Number of frames to run	NUM_FRAMES {value}	Number of video frames fed by testbench.
Number of active lines1	NUM_LINES {value}	Number of RX active video lines per frame.
Number of active pixels2	NUM_PIXELS {value}	Number of RX active video pixels per line.
Vertical Blanking from XVS to active line1	VFRONT_BLNK {value}	Number of blanking lines before the active video line.
Vertical Blanking after active line1	VREAR_BLNK {value}	Number of blanking lines after the active video line.
Horizontal Blanking period2	HB_PERIOD {value}	Horizontal Blanking period in SubLVDS clock cycles.
I2C Slave Address	I2C_SLAVE_ADR {value}	Define 7-bit of I2C Slave Address. Value must match the one set for i2c_s module in Radiant. Applicable only when USE_I2C is defined.
Software Reset Register3	I2C_SW_RST_N {value}	Write value to the software reset register of I2C Slave module. Value must be 1'b0 or 1'b1. Applicable only when USE_I2C is defined. Active low.
Top Line Trimming4, 5	I2C_TOP_TRIM {value}	Write value to the top trim register of I2C Slave module. Value must be 6'd0 – 6'd63. Applicable only when USE_I2C is defined.
Vertical Active Lines on TX5	I2C_V_ACTIVE {value}	Write value to the vertical active line register of I2C Slave module. Value must be 12'd1 – 12'd4095. Applicable only when USE_I2C is defined.
Left Pixel Unit Trimming6, 8	I2C_LEFT_TRIM_UNIT {value}	Write value to the pixel unit trim register of I2C Slave module. Value must be 6'd0 – 6'd63. Applicable only when USE_I2C is defined.
Left Pixel Trimming6, 8	I2C_LEFT_TRIM_LANE {value}	Write value to the pixel trim register of I2C Slave module. Value must be 4'd0 – 4'd9 and less than the RX lane count. Applicable only when USE_I2C is defined.
Horizontal Active Pixel units on TX7, 8	I2C_H_ACTIVE_UNIT {value}	Write value to the horizontal active pixel unit register of I2C Slave module. Value must be 10'd1 – 10'd1023. Applicable only when USE_I2C is defined.
Active Word Count9	I2C_WC {value}	Write value to the word count register of I2C Slave module. Value must be 16'd5 – 16'd65535. Applicable only when USE_I2C is defined.
Virtual Channel ID	I2C_VC {value}	Write value to the virtual channel ID register of I2C Slave module. Value must be 2'd0 – 2'd3. Applicable only when USE_I2C is defined.
KEEP HS mode	I2C_KEEP_HS {value}	Write value to the keep HS mode register of I2C Slave module. Value must be 1'b0 or 1'b1. Applicable only when USE_I2C is defined.
Total line count10	I2C_V_TOTAL {value}	Write value to the total line count register of I2C Slave module. Value must be 12'd10 – 12'd4095. Applicable only when USE_I2C and SENSOR_MODE_SLAVE are defined.
Total horizontal cycle10	I2C_H_TOTAL {value}	Write value to the total horizontal cycle count register of I2C Slave module. Value must be 12'd10 – 12'd4095. Applicable only when USE_I2C and SENSOR_MODE_SLAVE are defined.
XHS (Horizontal Sync) pulse cycle10	I2C_XHS_LENGTH {value}	Write value to the XHS pulse length register of I2C Slave module. Value must be 8'd1 – 8'd255. Applicable only when USE_I2C and

Category	Directive	Remarks
		SENSOR_MODE_SLAVE are defined.

Notes:

1. Total number of lines per frame is (NUM_LINES + VFRONT_BLNK + VREAR_BLNK).
2. In the case of SENSOR_MODE_MASTER, total number of SubLVDS clock cycles per line is (((NUM_PIXELS/NUM_RX_LANE) + 8) x (RAW number (10 or 12)) / 2) + HB_PERIOD).
3. Logical OR between this register and system reset (reset_n_i) is used to reset modules other than I2C slave module.
4. Value = 0 is not allowed when no line is trimmed by SubLVDS RX IP.
5. (I2C_TOP_TRIM + I2C_V_ACTIVE) cannot exceed the vertical active line count of the incoming RX data. It is your responsibility to manage this.
6. Number of pixels trimmed from the left edge is ((I2C_LEFT_TRIM_UNIT x number of RX lanes x (RX_GEAR / 8)) + I2C_LEFT_TRIM_LANE).
7. Active pixel count sent to TX is (I2C_H_ACTIVE_UNIT x number of RX lanes x (RX_GEAR / 8)).
8. ((I2C_LEFT_TRIM_UNIT + I2C_H_ACTIVE_UNIT) x number of RX lanes x (RX_GEAR / 8) + I2C_LEFT_TRIM_LANE) cannot exceed the horizontal active pixel count of the incoming RX data. It is your responsibility to manage this.
9. I2C_WC ≤ (I2C_H_ACTIVE_UNIT x number of RX lanes x (RX_GEAR / 8)) x (RAW number (10 or 12)) / 8. Refer to the trim_ctrl section for details.
10. Refer to the image sensor data sheet for proper settings.

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. SubLVDS to MIPI CSI2 Top-Level I/O

Port Name	Direction	Description
Reset		
reset_n_i	I	Asynchronous active low system reset
Control Interface (conditional)		
inck_i	I	Clock to control XVS and XHS. Only used in case of SENSOR_MODE_SLAVE.
xvs_o	O	Vertical Sync signal to the image sensor. Only used in case of SENSOR_MODE_SLAVE.
xhs_o	O	Horizontal Sync signal to the image sensor. Only used in case of SENSOR_MODE_SLAVE.
Control Interface (optional)		
scl	I/O	I2C clock. Only used in case of USE_I2C.
sda	I/O	I2C data Only used in case of USE_I2C.
SubLVDS RX Interface		
clk_p_i	I	Positive differential RX SubLVDS input clock
clk_n_i	I	Negative differential RX SubLVDS input clock
d0_p_i	I	Positive differential RX SubLVDS input data 0
d0_n_i	I	Negative differential RX SubLVDS input data 0
d1_p_i	I	Positive differential RX SubLVDS input data 1
d1_n_i	I	Negative differential RX SubLVDS input data 1
d2_p_i	I	Positive differential RX SubLVDS input data 2
d2_n_i	I	Negative differential RX SubLVDS input data 2
d3_p_i	I	Positive differential RX SubLVDS input data 3
d3_n_i	I	Negative differential RX SubLVDS input data 3
d4_p_i	I	Positive differential RX SubLVDS input data 4 (in case of 6/8/10-lane configuration)
d4_n_i	I	Negative differential RX SubLVDS input data 4 (in case of 6/8/10-lane configuration)
d5_p_i	I	Positive differential RX SubLVDS input data 5 (in case of 6/8/10-lane configuration)
d5_n_i	I	Negative differential RX SubLVDS input data 5 (in case of 6/8/10-lane configuration)
d6_p_i	I	Positive differential RX SubLVDS input data 6 (in case of 8/10-lane configuration)

Port Name	Direction	Description
d6_n_i	I	Negative differential RX SubLVDS input data 6 (in case of 8/10-lane configuration)
d7_p_i	I	Positive differential RX SubLVDS input data 7 (in case of 8/10-lane configuration)
d7_n_i	I	Negative differential RX SubLVDS input data 7 (in case of 8/10-lane configuration)
d8_p_i	I	Positive differential RX SubLVDS input data 8 (in case of 10-lane configuration)
d8_n_i	I	Negative differential RX SubLVDS input data 8 (in case of 10-lane configuration)
d9_p_i	I	Positive differential RX SubLVDS input data 9 (in case of 10-lane configuration)
d9_n_i	I	Negative differential RX SubLVDS input data 9 (in case of 10-lane configuration)
CSI-2 TX Interface		
clk_p_o	O	Positive differential TX CSI-2 output clock
clk_n_o	O	Negative differential TX CSI-2 output clock
d0_p_o	O	Positive differential TX CSI-2 output data 0 (in case of 1-lane configuration)
d0_n_o	O	Negative differential TX CSI-2 output data 0 (in case of 1-lane configuration)
d1_p_o	O	Positive differential TX CSI-2 output data 1 (in case of 2/4-lane configuration)
d1_n_o	O	Negative differential TX CSI-2 output data 1 (in case of 2/4-lane configuration)
d2_p_o	O	Positive differential TX CSI-2 output data 2 (in case of 4-lane configuration)
d2_n_o	O	Negative differential TX CSI-2 output data 2 (in case of 4-lane configuration)
d3_p_o	O	Positive differential TX CSI-2 output data 3 (in case of 4-lane configuration)
d3_n_o	O	Negative differential TX CSI-2 output data 3 (in case of 4-lane configuration)

3. Design and Module Description

The top-level design (sublvds2csi2_LFCPNX.v) consists of the following modules:

- rx_sublvds
- trim_ctrl
- pix2byte
- lane_ctrl
- tx_dphy
- int_gpll
- i2c_slave (optional)
- int_osc (optional)

The top-level design has a reset synchronization logic.

3.1. rx_sublvds

This module must be created for RX channel according to channel conditions, such as the number of lanes, bandwidth, and others. Figure 3.1 and Figure 3.2 show an example of IP interface settings by Radiant for the SubLVDS Image Sensor Receiver IP Core. You can use the ipx file (rx_sublvds/rx_sublvds.ipx) included in the sample project and re-configure according to your needs. Refer to SubLVDS Image Sensor Receiver IP Core User Guide (FPGA-IPUG-02093) for details.

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

- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting (4, 6, 8, or 10).
- RX Gear – Always 8. Not editable.
- RX Line Rate – Set according to channel configuration. The following are the maximum values for different lane configurations:
  - 4-Lane, Gear 8 – 1250 Mbps
  - 6-Lane, Gear 8 – 1250 Mbps
  - 8-Lane, Gear 8 – 1250 Mbps
  - 10-Lane, Gear 8 – 1000 Mbps
- SubLVDS Clock frequency – Automatially set (in MHz). Not editable. This value is equal to (Rx Line rate / 2)
- Pixel clock Frequency – Automatically set (in MHz). Not editable. This value is equal to (Rx Line rate / Rx Gear)
- Dropped Line Mode – Can be set as Dynamic or Static according to the channel configuration.
- Dropped Line Count – Set 0. This value must not be 0 when TOP_TRIM / I2C_TOP_TRIM are set to 0.
- Dropped Pixel Mode – Can be set as Dynamic or Static according to the channel configuration.
- Dropped Pixel Count – Set 0.
- Word Count Mode – Can be set as Dynamic or Static according to the channel configuration.
- Word Count – Set 0.
- Video Packet Data Type – Select RAW10 or RAW12.
- Image Sensor Mode – Can be set as Master or Slave.
- V_TOTAL – Set 30. Applicable only if Image sensor mode is Slave.
- H_TOTAL – Set 640. Applicable only if Image sensor mode is Slave.
- V_H_BLANK – Set 4. Applicable only if Image sensor mode is Slave.

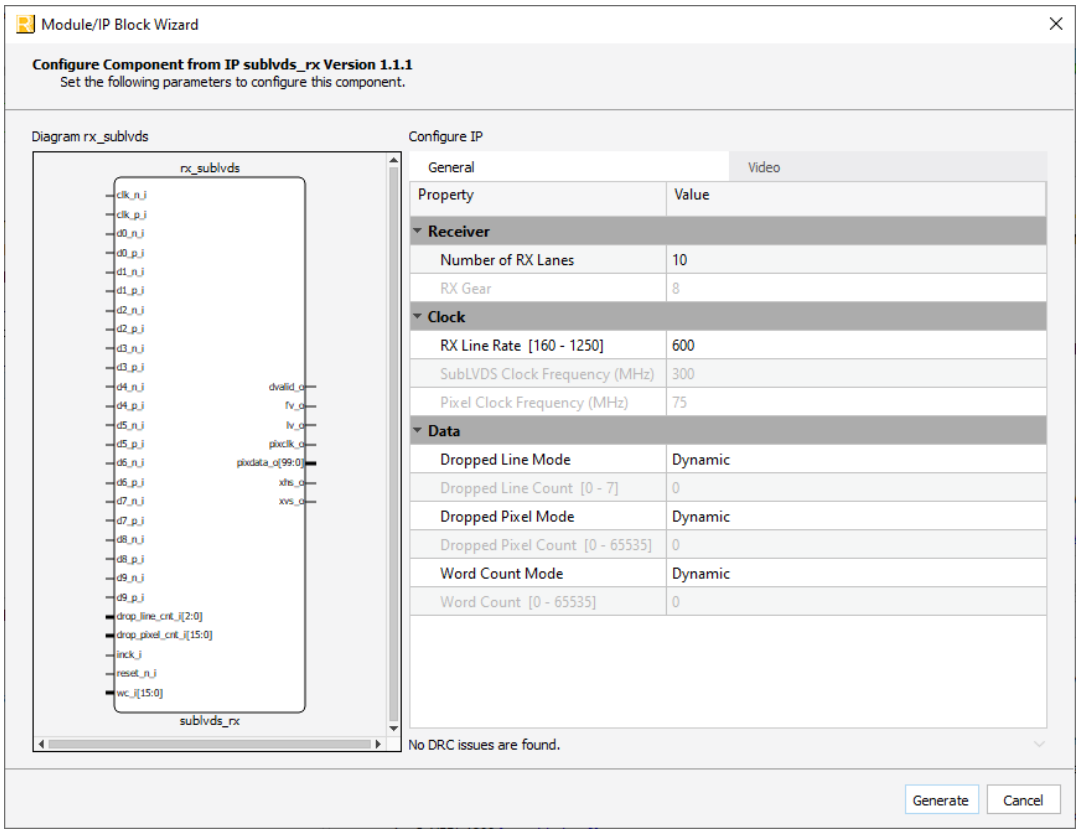


Figure 3.1. rx_sublvds IP Creation #1

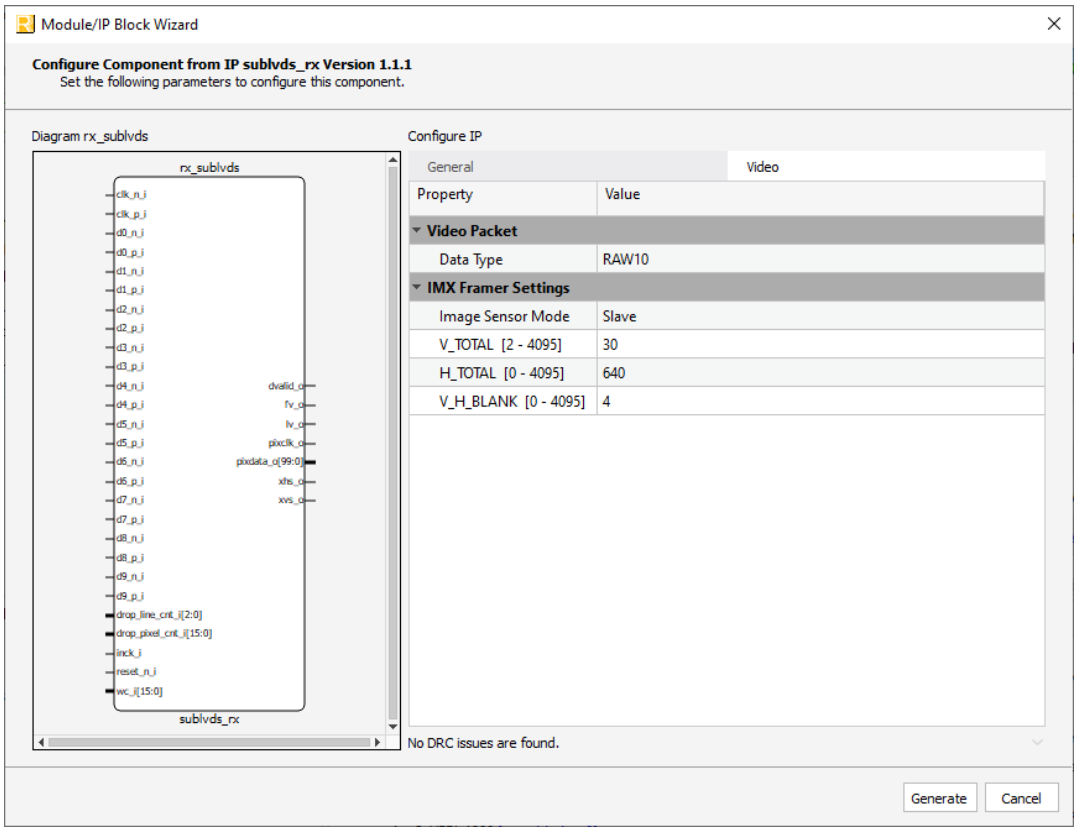


Figure 3.2. rx_sublvds IP Creation #2

This module takes serial SubLVDS data from the image sensor and outputs pixel data after de-serialization. In case that you generate this IP from scratch, it is recommended to set the module name to rx_sublvds 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 have to modify the names accordingly.

Figure 3.3 shows a global timing example of RAW10 in 10-lane configuration. This IP takes data between SAV (Start of Active Video) and EAV (End of Active Video) as active video data and outputs along with the assertion of dvalid. In case of Gear 8, the pixel clock frequency is ¼ of the incoming SubLVDS clock. Incoming data is DDR, which means 8 bits of data are de-serialized every pixel clock cycle. In both RAW10 and RAW12 cases, a single pixel clock cycle is not enough to retrieve one pixel data on each lane. Therefore, dvalid has on and off cycles. In case of RAW10, dvalid is asserted 4 out of every 5-pixel clock cycles (8 bits x 5 = 40 bits: 4 pixel data). In case of RAW12, dvalid is asserted 2 out of every 3-pixel clock cycles (8 bits x 3 = 24 bits: 2 pixel data).

Table 3.1 shows the sync code details. Both SAV and EAV comes from four words of data. The first word is always all 1 and second and third words are always all 0. The fourth word determines the type of the sync codes.

As you can see in Table 3.1, there are only blanking lines and non-blanking lines and no concept of front porch or back porch against vertical sync. As shown in Figure 3.3, XVS (Vertical sync) belongs to the first non-blaking lines in a frame. In case of Figure 3.3, more than 18 lines exist from the first non-blanking line to the active video line. Therefore, top line trimming is effective to process all necessary lines. On the other hand, it will require a special attention if we need to process the very first line since we don't have enough time to process frame start packet transaction in CSI-2 which requires low power to high-speed mode transition while acquiring the payload data of the first line. To avoid that condition, this design does not process the first line payload data. In other words, this design mandate the first line is always skipped. This is guaranteed by "Dropped line in subLVDS RX IP GUI" + "TOP_TRIM/I2C_TOP_TRIM set in synthesis_directives/simulation_directives" > 0.

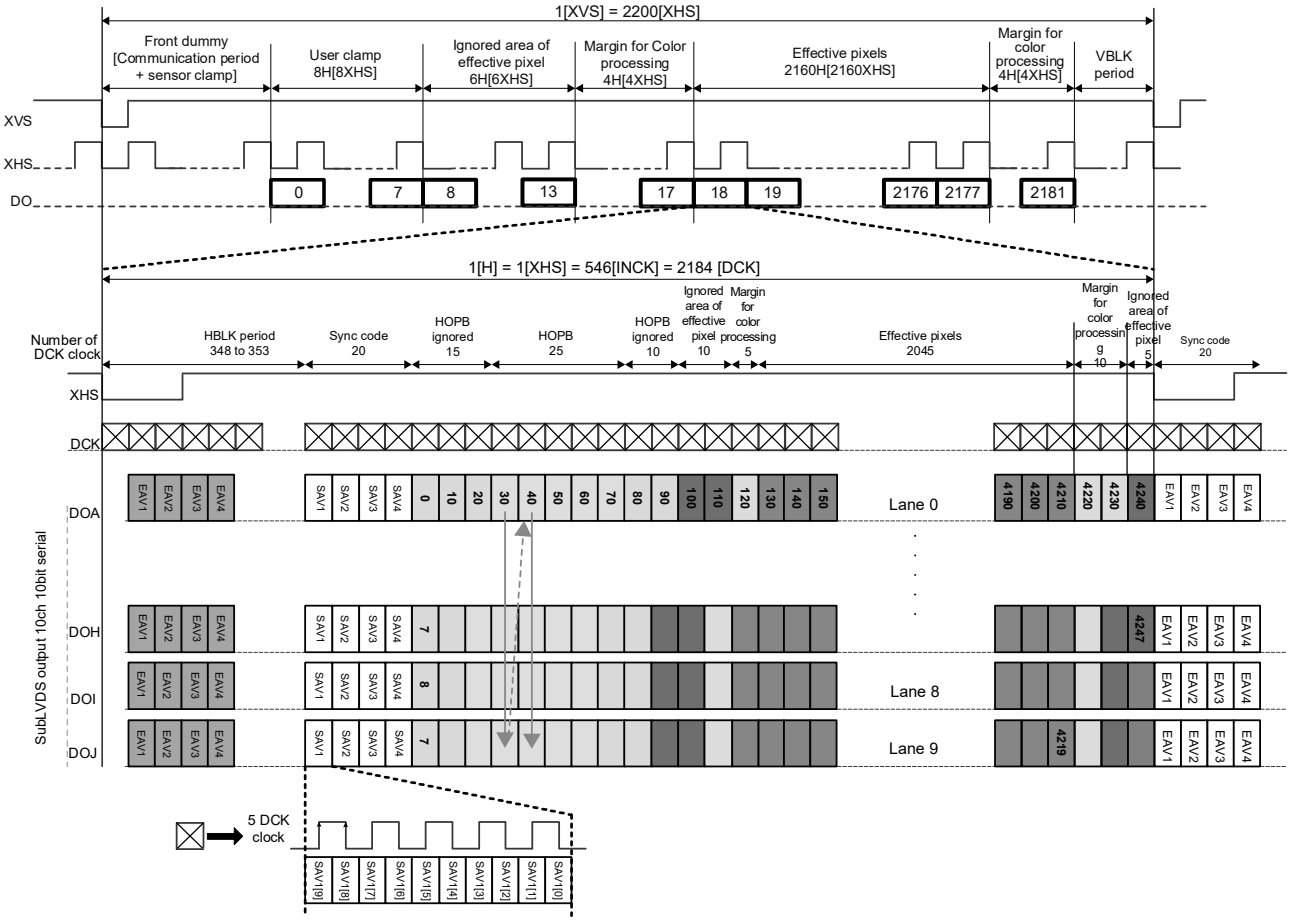


Figure 3.3. SubLVDS Input Global Timing (RAW10, 10 Lanes)

Table 3.1. Sync Code Details

LVDS Output Bit No.		Sync code				
12-Bit Output	10-Bit Output	First Word	Second Word	Third Word	Fourth Word	
11	9	1	0	0	1	
10	8	1	0	0	0	
9	7	1	0	0	V	1: Blanking line 0: Except blanking line
8	6	1	0	0	H	1: End sync code 0: Start sync code
7	5	1	0	0	P3	Protection bits
6	4	1	0	0	P2	
5	3	1	0	0	P1	
4	2	1	0	0	P0	
3	1	1	0	0	0	
2	0	1	0	0	0	
1	—	1	0	0	0	
0	—	1	0	0	0	

		Protection Bits			
V	H	P3	P2	P1	P0
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

Moreover, Rx SubLVDS IP also manages to feed the horizontal and vertical sync signals (xhs_o, xvs_o) to the image sensor using the external clock (inck_i). Sync pulse polarity and interval can be changed by directives described in the Synthesis Directives and Simulation Directives sections. Figure 5.4 shows xvs_o having one line length of active pulse by XVS_LENGTH_LINE directive.

3.2. trim_ctrl

In many cases, the data that comes out from rx_sublvds includes unnecessary data and discarded by the downstream devices. This module enables to trim the edge data based on the given parameters. The following parameters are taken as input data.

- top_trim_i[5:0] – Number of top lines to be trimmed. This value doesn't include the dropped lines by SubLVDS RX IP. When the number of dropped line by SubLVDS RX IP is 0, top_trim_i cannot be 0.
- v_active_i[11:0] – Number of active lines to be sent to TX module.
- left_trim_unit_i[5:0] – Number of unit pixels to be trimmed. One unit is pixels equal to RX lane count in case of RX Gear 8.
- left_trim_lane_i[3:0] – Number of pixels to be trimmed after unit trimming. Total number of pixels to be trimmed is (left_trim_unit_i) x (RX lane count) x (RX Gear / 8) + (left_trim_lane_i).
- h_active_unit_i[9:0] – Number of unit pixels to be sent to TX module. One unit is equal to RX lane count in case of RX Gear 8.

Figure 3.4 shows a global timing example of trim_ctrl. In this case, the first and last lines are trimmed by trim_ctrl (note that the original first line is not trimmed by rx_sublvds).

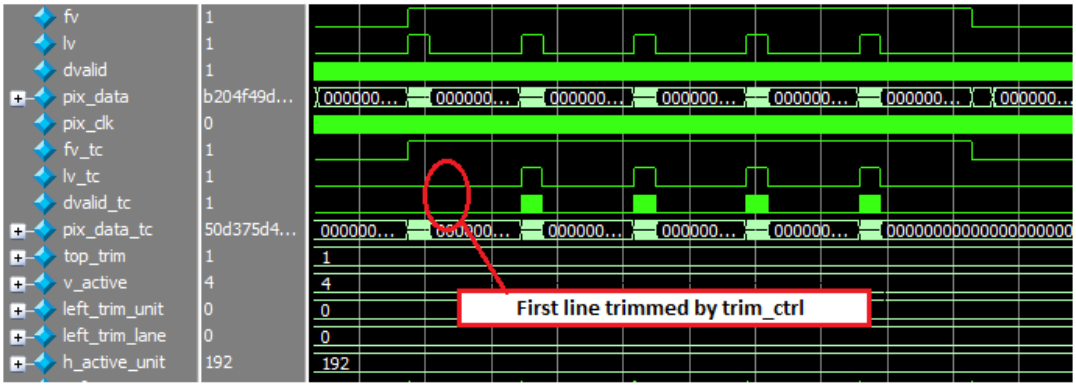


Figure 3.4. trim_ctrl Global Timing (RAW10, 10 Lanes)

Figure 3.5 shows the close-up of the above focusing the beginning of the line. In this case, the pixel counts to be trimmed is 4 (left_trim_unit) x 10 (RX lane count) + 3 (left_trim_lane) = 43. That means 4-unit data are trimmed from the beginning and 3 pixel data (LSB 30 bits) are trimmed from the fifth unit data (pix_data_A). The rest of the fifth unit data are shifted down towards LSB and LSB 30 bits of the next data (pix_data_B) are placed at the MSB 30 bits. The result is the first output data from lane_ctrl (pix_data_tc_A). This means:

```
pix_data_tc_A[99:0] = {pix_data_B[29:0], pix_data_A[99:30]}.
```

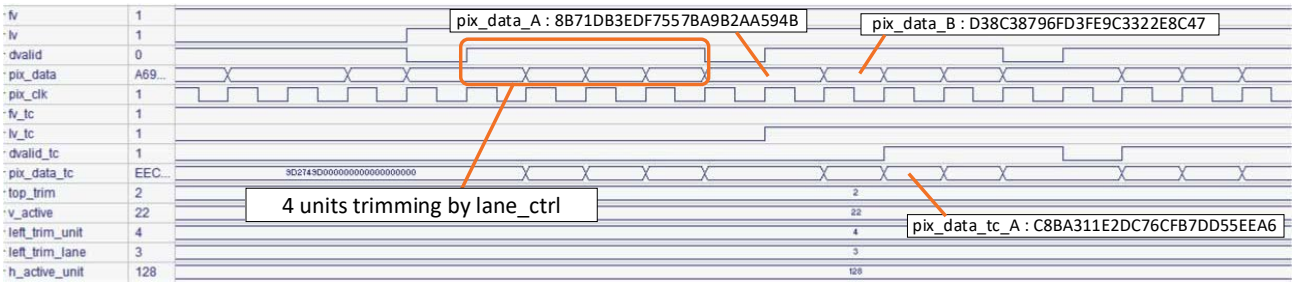


Figure 3.5 Trimming in the Beginning of the Line (RAW10, 10 Lanes)

The number of lines and pixels to be trimmed can be set in the unit of 1 line or 1 pixel as well as number of active lines to be cropped. However, there exist some limitations on the number of pixels to be cropped. Pixel cropping has to be in the unit of RX lane count as a parameter h_active_unit. In addition to this, the number of active pixels must be a multiple of 4 in case of RAW10 and a multiple of 2 in case of RAW12 according to CSI-2 spec. Moreover, due to the FIFO data width in rx_sublvds, another restriction has to be applied. Table 3.2 shows the unit value (granularity) of h_active_unit for all cases considering all of limitations mentioned above. In most cases, cropping with finer granularity is possible through WC setting. This WC is fed to lane_ctrl as the second stage cropping.

Example:

```
RAW10, 10 RX lane, 4 TX lane with cropping 1920 pixels from 2400 active pixel
input
The unit value of h_active_unit for RAW10, 10 RX lane is 16.
Since 1920 / (10 lanes x 16 units) = 12.
set h_active_unit = 16 x 12 = 192 to cover 1920 pixels.
WC = 1920 x 5/4 = 2400.
```

Table 3.2. Granularity of h_active_unit and WC

Data Type	RX Lane Count	RX Gear	TX Lane Count	TX Gear	Unit Value of h_active_unit	Granularity of WC
RAW10	4	8	1	16	4 (= 16 pixels)	10 (= 8 pixels)
			2	16	4 (= 16 pixels)	20 (= 16 pixels)
			4	8	4 (= 16 pixels)	20 (= 16 pixels)
	6	8	1	16	8 (= 48 pixels)	10 (= 8 pixels)
			2	16	8 (= 48 pixels)	20 (= 16 pixels)
			4	8	8 (= 48 pixels)	20 (= 16 pixels)
	8	8	2	16	4 (= 32 pixels)	20 (= 16 pixels)
			4	16	4 (= 32 pixels)	40 (= 32 pixels)
	10	8	2	16	16 (= 160 pixels)	20 (= 16 pixels)
			4	16	16 (= 160 pixels)	40 (= 32 pixels)
RAW12	4	8	1	16	2 (= 8 pixels)	6 (= 4 pixels)
			2	16	2 (= 8 pixels)	12 (= 8 pixels)
			4	8	2 (= 8 pixels)	12 (= 8 pixels)
	6	8	1	16	4 (= 24 pixels)	6 (= 4 pixels)
			2	16	4 (= 24 pixels)	12 (= 8 pixels)
			4	8	4 (= 24 pixels)	12 (= 8 pixels)
	8	8	2	16	2 (= 16 pixels)	12 (= 8 pixels)
			4	16	2 (= 16 pixels)	24 (= 16 pixels)
	10	8	2	16	8 (= 80 pixels)	12 (= 8 pixels)
			4	16	8 (= 80 pixels)	24 (= 16 pixels)

3.3. pixel2byte

This module must be created for RX channel according to data type, the number of RX lanes, RX Gear, and others. Figure 3.6 shows an example of IP interface settings in Radiant for the pixel2byte IP. You can use the ipx file (pix2byte/pix2byte.ipx) included in the sample project and re-configure according to your needs. Refer to Pixel-to-Byte Converter IP Core User Guide (FPGA-IPUG-02094) for details.

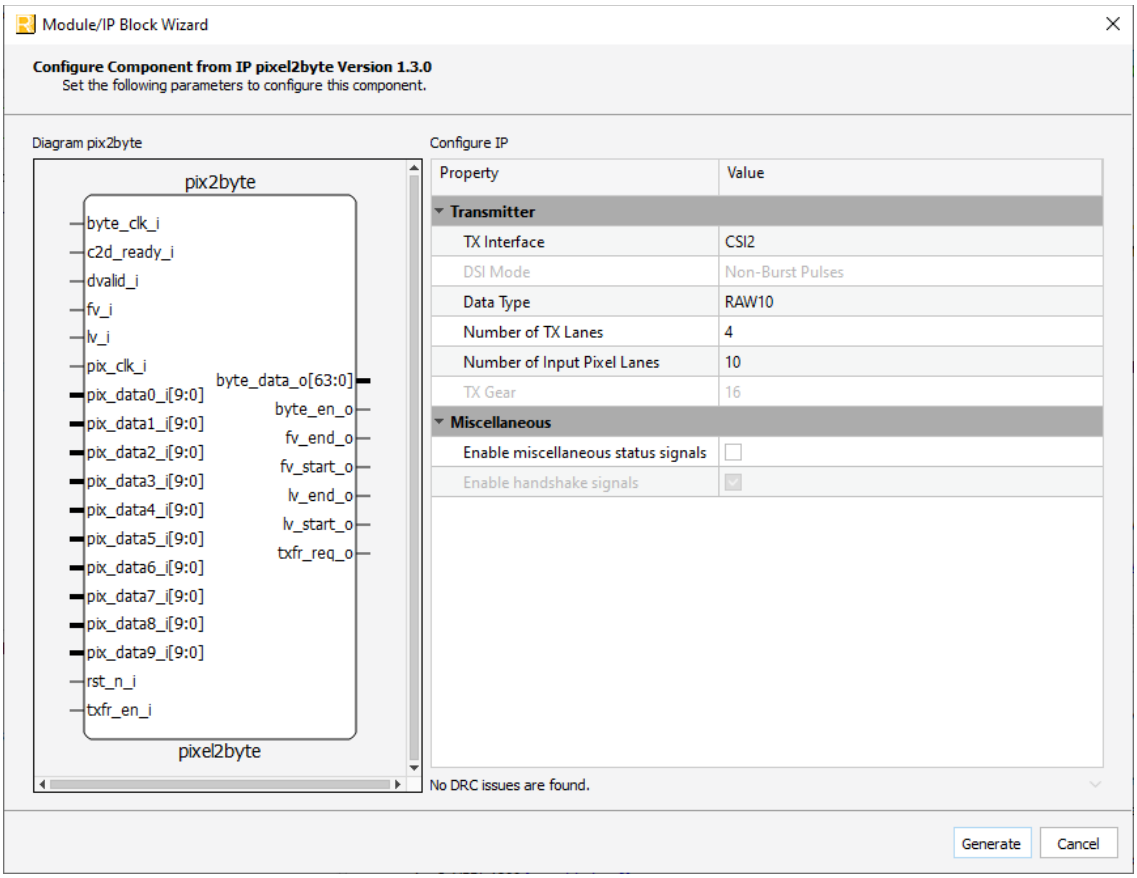


Figure 3.6. pixel2byte IP Creation

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

- TX Interface – Select CSI-2.
- Data Type – Select RAW10 or RAW12. Others are not supported by RX SubLVDS IP.
- Number of TX Lanes – Always set 4.
- Number of Input Pixel Per Clock – Set the value equal to (RX lane count) x (RX Gear / 8).
- TX Gear – automatically set in case of Number of Pixel Per Clock = 6, 8, or 10. 8 or 16 can be selected in case of Number of Input Pixel Per Clock = 4 (refer to Table 1.1).
- Enable miscellaneous status signals – unchecked.

To generate this IP from scratch, it is recommended to set the module name to pix2byte 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 have to modify the names accordingly.

This module receives pixel data (pix_data_tc) from trim_ctrl along with fv_tc, lv_tc, and dvalid_tc (frame valid, line valid and data valid) and re-organizes the data to form the byte data via FIFO according to the data type specified. FIFO is used as a data buffer as well as a clock domain bridge. Pixel data are written to FIFO in pixel clock domain and read in byte clock domain. Byte clock is provided from TX D-PHY module. Figure 3.7 shows the global timing of pixel2byte in the case of RX 10-lane with RX Gear 8.

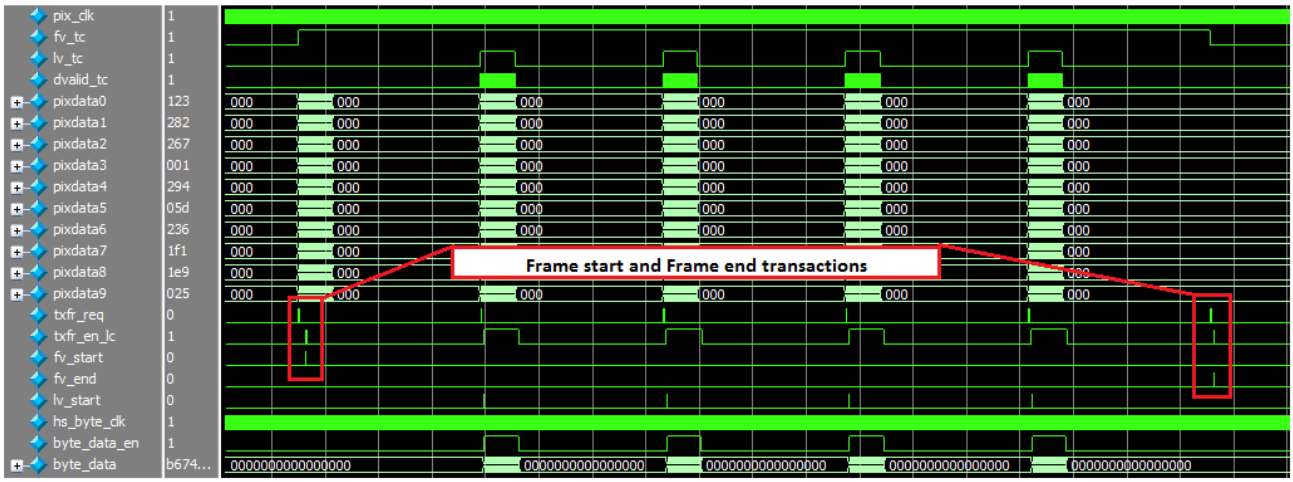


Figure 3.7. Global Timing of pixel2byte

Figure 3.8 shows line transactions. pixel2byte asserts txfr_req after receiving the valid line data and start sending byte based data following lv_start assertion. At the end of one-line valid data, txfr_req is asserted again. This assertion is redundant and masked by lane_ctrl and does not affect the system operation.

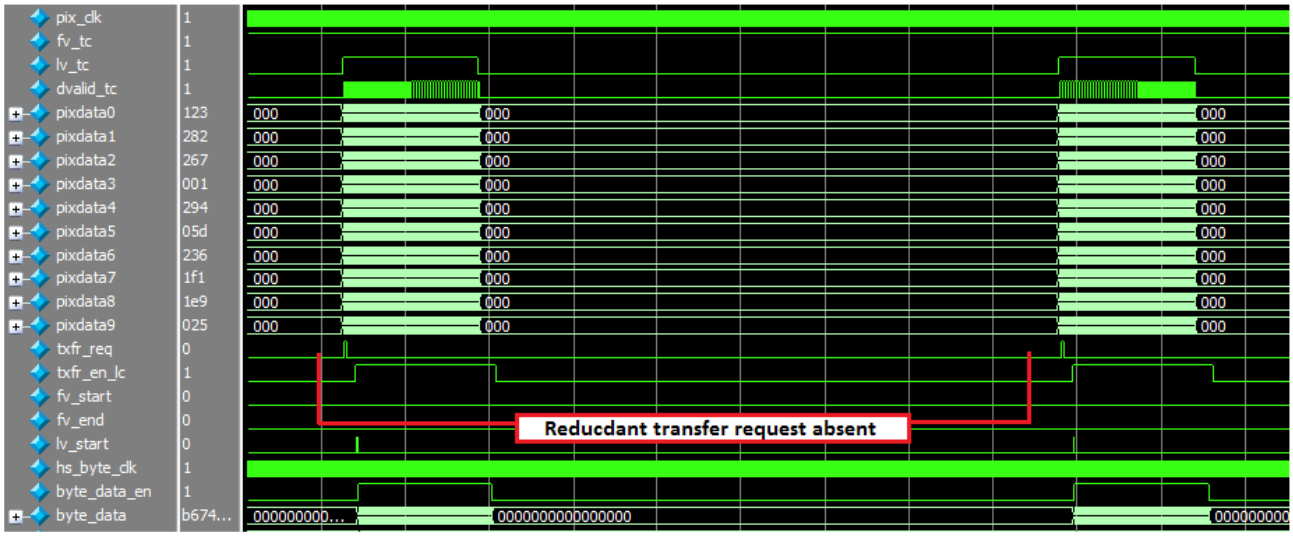


Figure 3.8. Line Transactions of pixel2byte

3.4. lane_ctrl

This module resides between pixel2byte and tx_phy. It has two major functions:

- Communication control between pixel2byte and tx_dphy
- Bus width and Lane assignment conversion between pixel2byte and tx_dphy

3.4.1. Communication Control

pixel2byte and tx_dphy have handshake signals to request and grant data transfer from pixel2byte to tx_dphy, but current version of IP cannot handle a certain condition (Frame End event happens just after the valid data transmission of the last line) when those signals are directly connected. lane_ctrl holds the assertion of sp_en when txfr_en is high and waits for txfr_req_lc assertion until txfr_en goes low. Figure 3.9 shows the global timing of lane_ctrl in case of RAW10, RX lane count = 10 with RX Gear 8, TX lane count = 4 with TX Gear 8. In this case, byte data that comes from pixel2byte is 64 bits and active data input of TX D-PHY is 32 bits. Therefore hs_byte_clk (= 2x byte_clk) is used to match the input and output bandwidths. As described in the pixel2byte section, txfr_req assertion around the end of valid data transmission is masked by this module and is not transmitted to tx_dphy. This module also handles the clock domain crossing in case that hs_byte_clk is used. A compiler directive of HS_BYTE_CLK is automatically defined in the top-level RTL when hs_byte_clk is required.

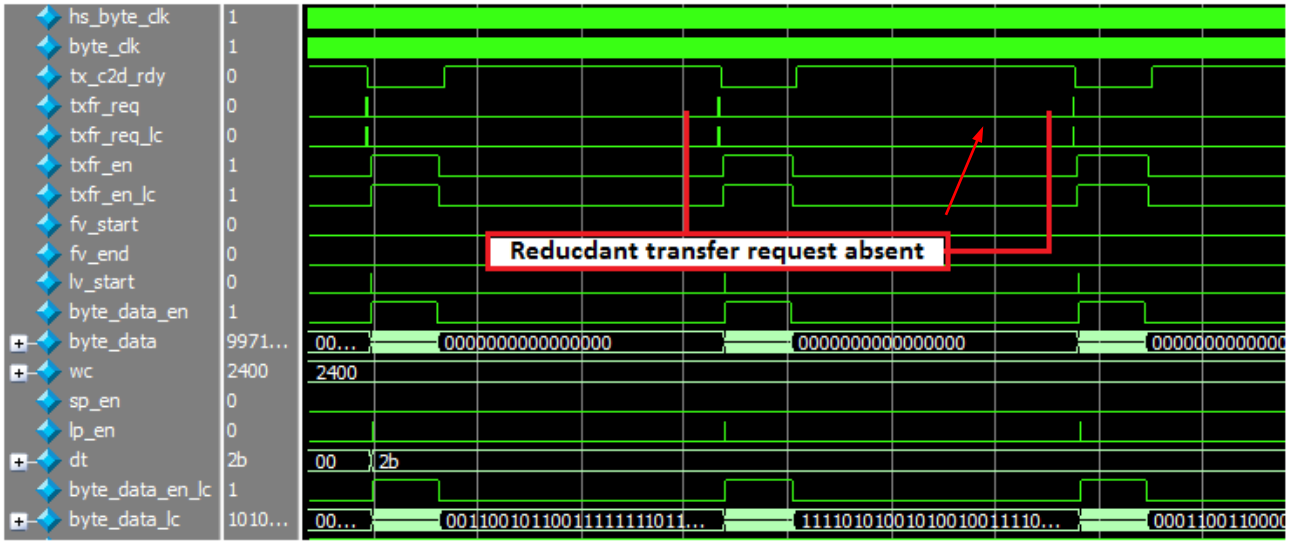


Figure 3.9. Global Timing of lane_ctrl

3.4.2. LP-HS Control in Continuous Clock Mode

Figure 3.10 shows the handshake flow from trim_ctrl to tx_dphy to send Frame Start short packet. fv_tc from trim_ctrl lets pixel2byte asserts txfr_req. Then txfr_req is forwarded as txfr_req_lc when tx_c2d_rdy = 1, which means tx_dphy is ready to begin a new HS transmission. After tx_dphy goes in to HS mode, txfr_en goes 1 and that was transferred as txfr_en_lc to pixel2byte. That lets pixel2byte assert fv_start and lane_ctrl asserts sp_en, which results in short packet transmission by tx_dphy.

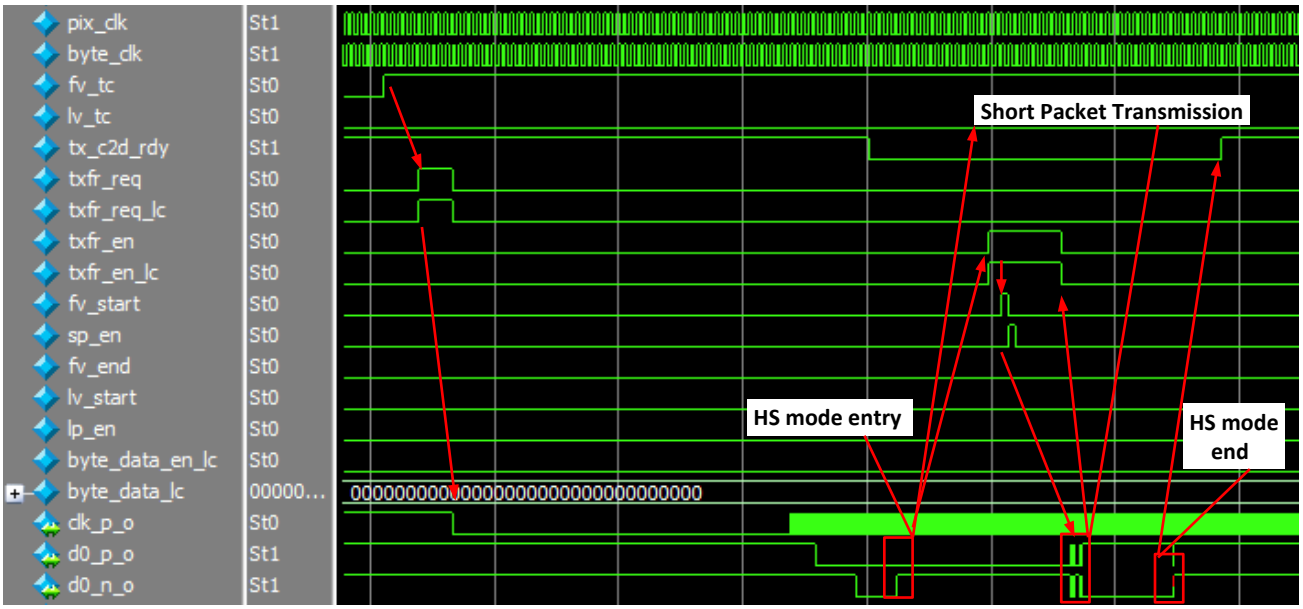


Figure 3.10. LP-HS Transition in Continuous clock mode (Short Packet)

Figure 3.11 shows an example of the handshake for Long Packet transmission. As mentioned earlier, txfr_req while txfr_en = 1 is masked by lane_ctrl and is not transferred to tx_dphy.

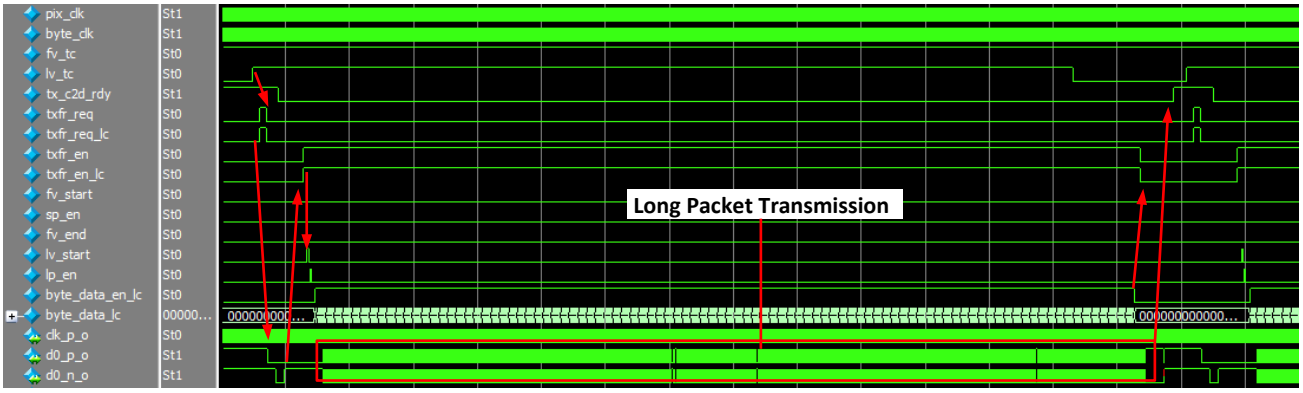


Figure 3.11. LP-HS Transition in Continuous clock mode (Long Packet)

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

This module controls LP-HS transition of tx_dphy module as shown in Figure 3.12 and Figure 3.13 with a following sequence in case of non-continuous clock mode.

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

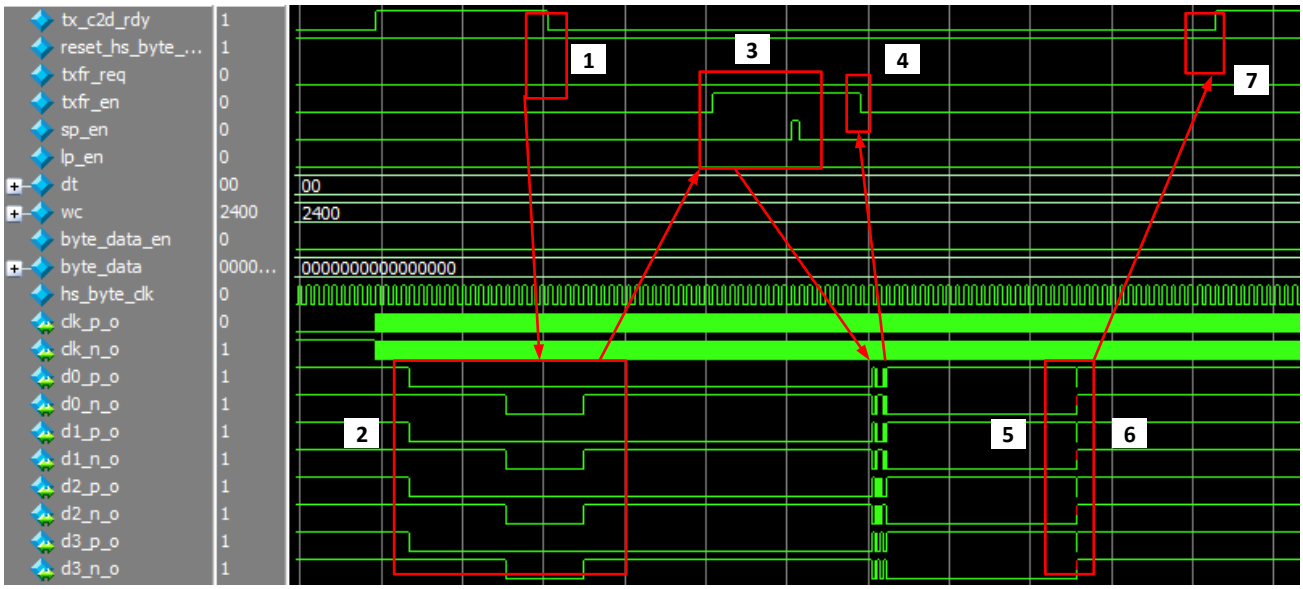


Figure 3.12. LP-HS Transition in Non-Continuous clock mode (Short Packet)

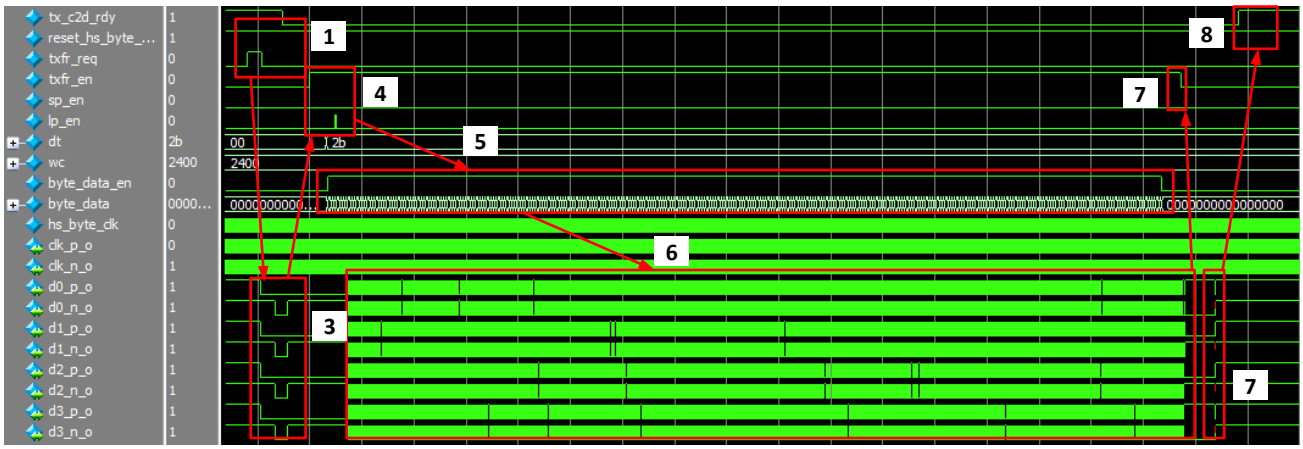


Figure 3.13. LP-HS Transition in Non-Continuous clock mode (Long Packet)

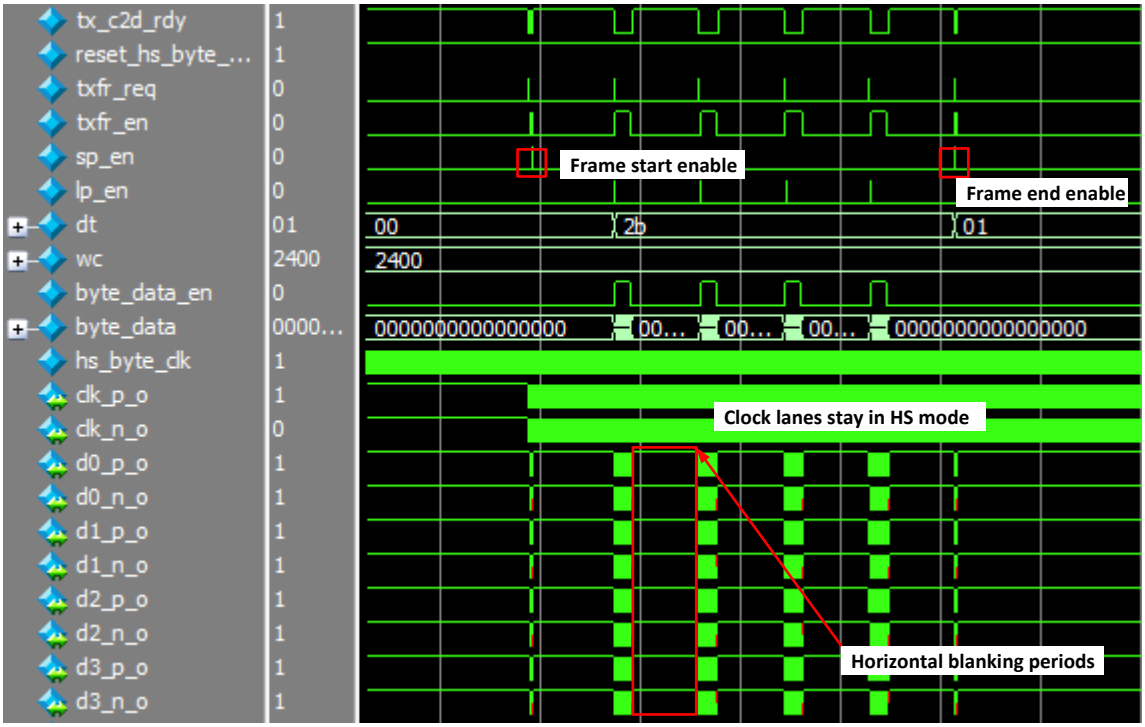


Figure 3.14. LP-HS Transition in Non-Continuous clock mode with KEEP_HS

3.4.4. Bus Width Conversion

Since pixel2byte configuration is always with 4 TX lanes, bus width conversion is necessary when TX lane count is 1 or 2. Figure 3.15 shows the example in case of RAW10, RX lane count = 10 with RX Gear 8, TX lane count = 4 with TX Gear 8. 64-bit data are stored into the internal FIFO using byte_clk and read back using hs_byte_clk in every other cycles and sent to tx_dphy.

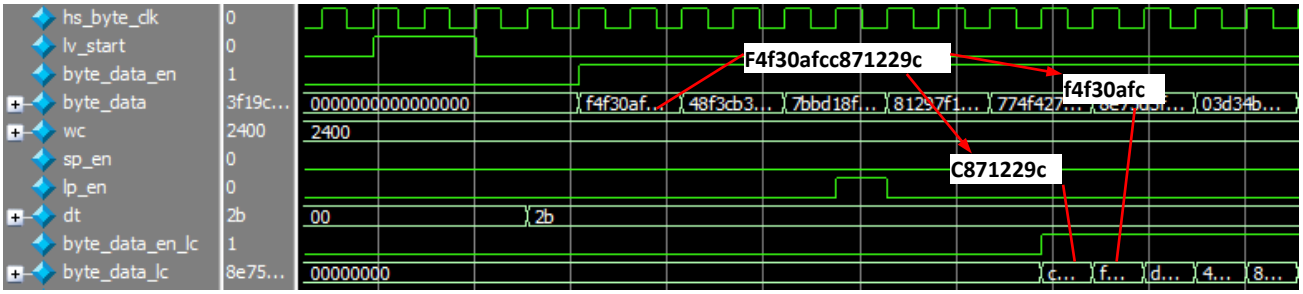


Figure 3.15. Bus Width Conversion by lane_ctrl

3.5. tx_dphy

This module must be created for TX channel according to the number of TX lanes, TX Line Rate, and others. Figure 3.16 shows an example of IP interface settings in Radiant for the TX D-PHY IP. You can use the ipx file (tx_dphy/tx_dphy.ipx) included in the sample project and re-configure according to your needs. Refer to CSI-2/DSI D-PHY Transmitter IP Core User Guide (FPGA-IPUG-02080) for details.

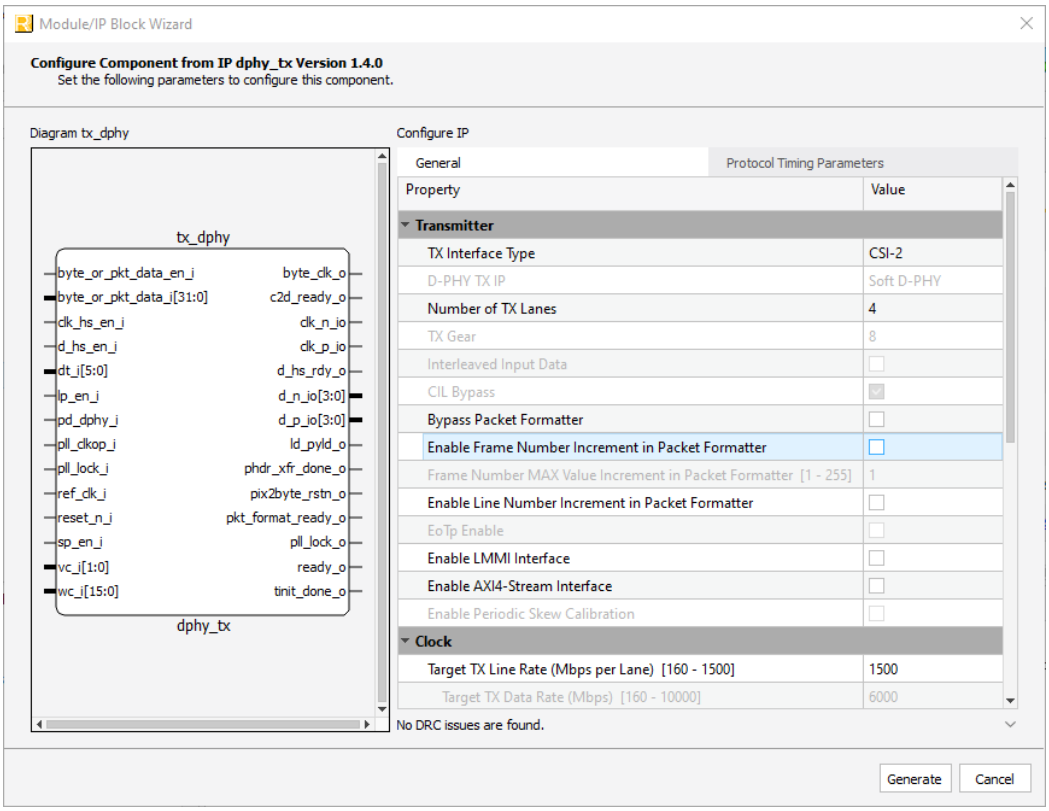


Figure 3.16. tx_dphy IP Creation #1

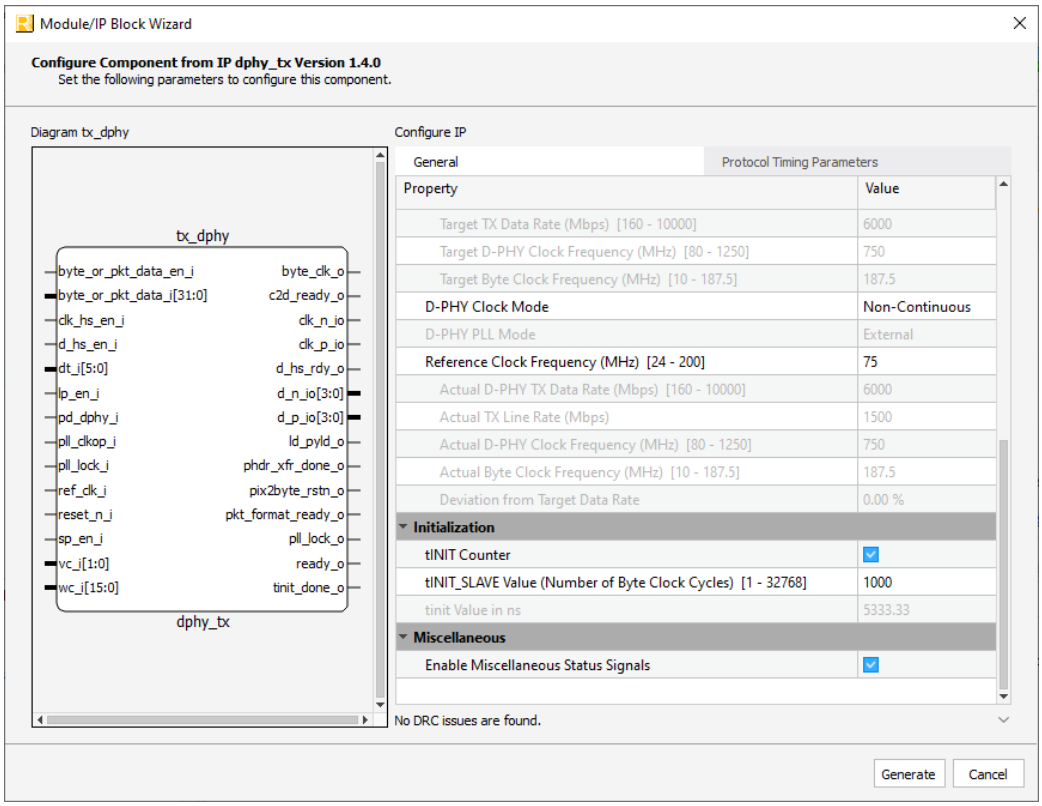


Figure 3.17. tx_dphy IP Creation #2

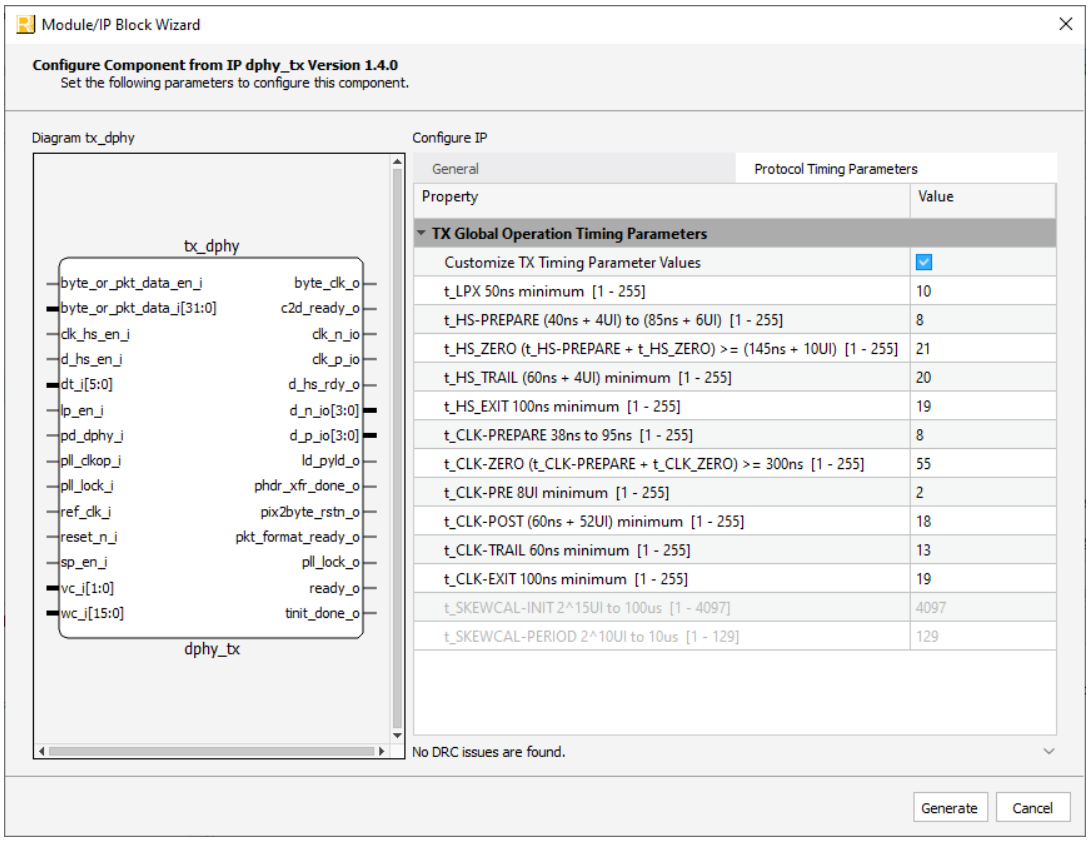


Figure 3.18. tx_dphy IP Creation #3

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

- TX Interface type – Select CSI-2.
- DPHY TX IP – Always Soft DPHY.
- Number of TX Lanes – Set according to TX lane configuration. Must match NUM_TX_LANE_* setting.
- TX Gear – Automatically set according to TX Line Rate.
- Interleaved Input Data – Must be disabled (unchecked).
- CIL Bypass – Must be enabled (checked).
- Bypass packet formatter – Must be disabled (unchecked).
- Enable frame number increment in Packet Formatter – Must be disabled (unchecked).
- Enable line number increment in Packet Formatter – Must be disabled (unchecked).
- TX Line Rate per Lane – Use the value specified in the Excel sheet.
- DPHY Clock Mode – Set according to TX channel configuration. Continuous is recommended when the length of the horizontal blanking period is unknown.
- DPHY PLL Mode – Always set Internal.
- Reference Clock Frequency – Set the appropriate value which can be obtained by pix_clk or GPLL output. This clock frequency must be 24 – 200 MHz.
- tINIT counter – Enabled (checked) is recommended.
- tINIT_SLAVE Value – value to make tinit Value more than 100 µs 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. In case that you generate this IP from scratch, it is recommended to set the design name to tx and module name to tx_dphy so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.

TX Line Rate is derived from the following equation:

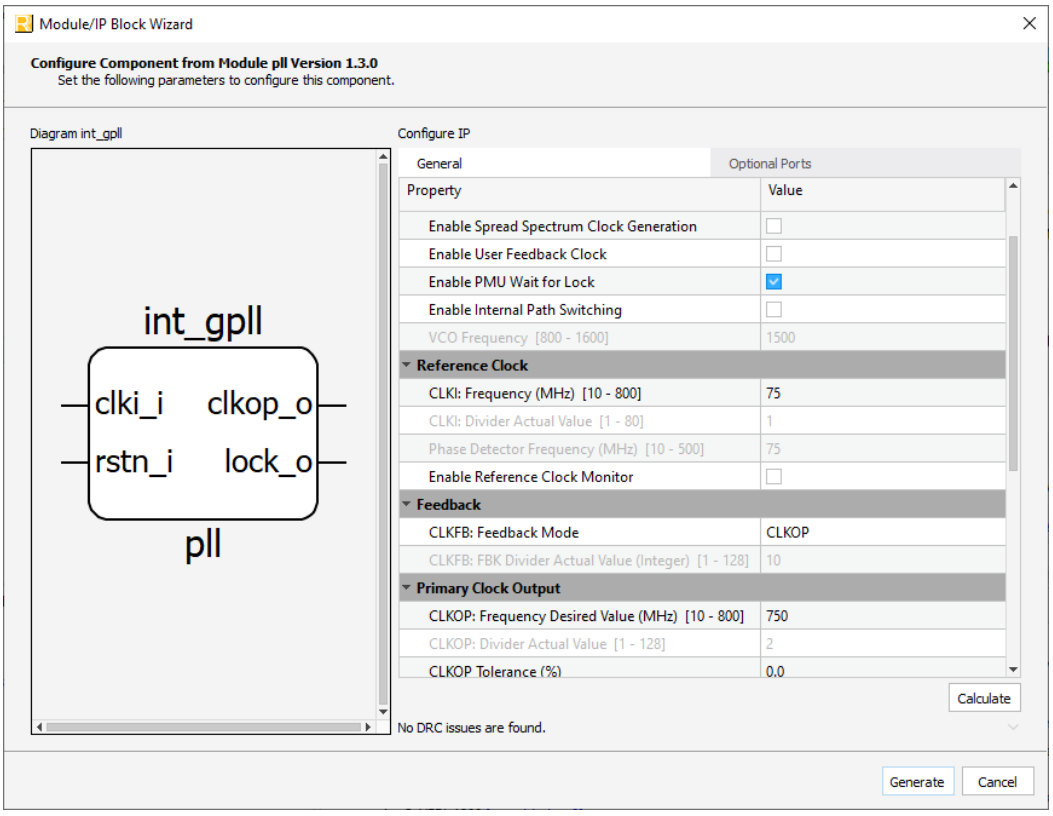
$$TX\_lane\_bandwidth = \frac{RX\_lane\_bandwidth * number\_of\_RX\_lane}{number\_of\_TX\_lane}$$

Example #1: RAW10 RX with 10-lane at RX lane bandwidth = 600 Mbps with 4 TX lanes
TX lane bandwidth = (600 x 10) / 4 = 1500 Mbps.

Example #2: RAW12 RX with 4-lane at RX lane bandwidth = 1200 Mbps with 4 TX lanes
TX lane bandwidth = (1200 x 4) / 4 = 1200 Mbps.

3.6. int_gpll

GPLL module is always used to generate the DPHY clock frequency for TX DPHY IP using pix_clk frequency from RxSubLVDS IP. You can use the ipx file (int_gpll/int_gpll.ipx) included in the sample project and re-configure according to your needs. In case if you generate this IP from scratch, it is recommended to set the design name to int_gpll so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly.



3.7. i2c_slave

This module is instantiated when USE_I2C is defined and enables you to change parameters on the fly through I2C connections. I2C Hard IP is instantiated and used as an I2C slave device. You can use the ipx file (i2c_s/i2c_s.ipx) included in the sample project and re-configure. In case that you generate this IP from scratch, it is recommended to set the module name to i2c_s so that you do not need to modify the instance name of this IP in i2c_slave as well as

simulation setup file. Otherwise, you need to modify the names accordingly. Refer to I2C Slave IP Core User Guide (FPGA-IPUG-02072) for details. Figure 3.20 shows basic settings of I2C Slave IP. The following shows guidelines and parameter settings required for this reference design.

- APB Mode Enable – Recommended to be disabled (unchecked).
- Addressing Mode – 7-bit is recommended.
- I2C Slave Address – Set the value of your selection.
- System Clock Frequency – Set the appropriate frequency.
- FIFO Depth – 16 is recommended.
- Implementation of FIFO – LUT is Recommended.
- TX FIFO Almost Empty Flag – Set the value of your selection (unused).
- TX FIFO Almost Full Flag – Set the value of your selection (unused).

In the current design, I2C data written from the external host device is stored in the FIFO and those data are read when I2C write transaction is completed. Since the first write data is for I2C sub address, total 17 data must be stored in case of consecutive full 16 data write. The current FIFO depth setting is 16 and FIFO overflow happens. To avoid that, consecutive data must be up to 15 data. Alternately, you can increase the FIFO depth to 32 if you want to write all 16 data in a single write transaction. In the sample design, 45 MHz clock generated by the on-chip oscillator is fed as a system clock.

One typical usage of this module is changing parameters during the system development and/or debugging stages so that you can try many cases without updating FPGA bitfiles as long as I2C registers cover what you would like to change. After that you can recreate a bitfile using the finalized parameters without USE_I2C directive for the product.

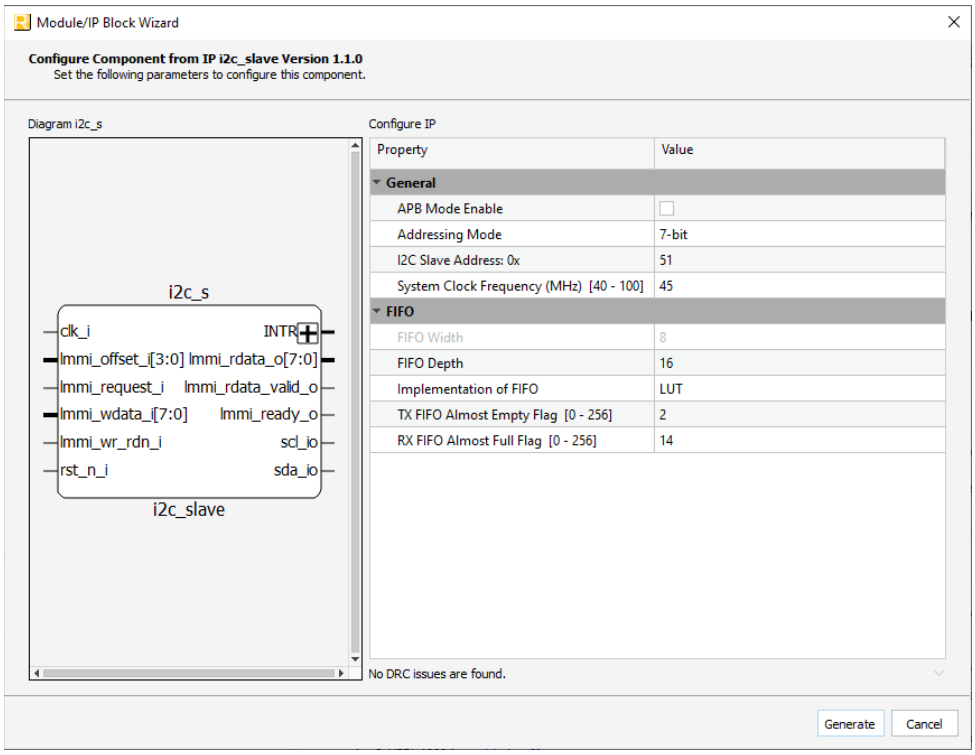


Figure 3.20. I2C Slave IP Creation

This module is equipped with parameter registers of 4-bit address area of I2C sub-address shown in Table 3.3.

Table 3.3. I2C Slave Register Map

Sub Address	Name	Bit assignment	Description
0	SW Reset	[0]	Software reset register. When this is active, all modules except for i2c_slave are in reset condition. Active low.

Sub Address	Name	Bit assignment	Description
1	TOP_TRIM	[5:0]	Top edge trim register. The value must be 6'd1 – 6'd63.
2	V_ACTIVE[7:0]	[7:0]	Vertical active line register. The value must be 12'd1 – 12'd4095.
3	V_ACTIVE[11:8]	[3:0]	
4	LEFT_TRIM_UNIT	[5:0]	Left edge trim register by the unit. The value must be 6'd0 – 6'd63.
5	LEFT_TRIM_LANE	[3:0]	Left edge trim register within the lanes. The value must be 4'd0 – 4'd9.
6	H_ACTIVE_UNIT[7:0]	[7:0]	Horizontal Active Pixel register by the unit. The value must be 10'd1 – 10'd1023.
7	H_ACTIVE_UNIT[9:8]	[1:0]	
8	WC[7:0]	[7:0]	Word count register. The value must be 16'd1 – 16'd65535.
9	WC[15:8]	[15:8]	
10	VC	[1:0]	Virtual channel ID register. The value must be 2'd0 – 2'd3.
	KEEP_HS	[7]	Force clock lane in HS mode during active lines regardless of TX clock mode.
11	V_TOTAL[7:0]	[7:0]	Total line count register used in Sensor Slave Mode. The value must be 12'd10 – 12'd4095.
12	V_TOTAL[11:8]	[3:0]	
13	H_TOTAL[7:0]	[7:0]	Total horizontal cycle count register used in Sensor Slave Mode. The value must be 12'd10 – 12'd4095.
14	H_TOTAL[11:8]	[3:0]	
15	XHS_LENGTH	[7:0]	XHS pulse length register. Used in Sensor Slave Mode. The value must be 8'd1 - 8'd255.

All registers are set to the default values specified by corresponding directives defined in synthesis_directives.v.

Software reset works as the system reset (reset_n_i) for all modules other than i2c_slave, therefore you can assert this while updating other I2C registers, then release Software reset upon completing the register update to avoid an unexpected operation during register update. Refer to the Simulation Directives and trim_ctrl sections for register details.

3.8. int_osc

In the sample design, a 45 MHz clock is generated by the on-chip oscillator and fed to i2c_slave module. You can use the ipx file (int_osc/int_osc.ipx) included in the sample project and re-configure according to your needs. In case you generate this IP from scratch, it is recommended to set the module name to int_osc so that you do not need to modify the instance name of this IP in the top level design as well as simulation setup file. Otherwise, you need to modify the names accordingly. This module is not necessary when i2c_slave is not used.

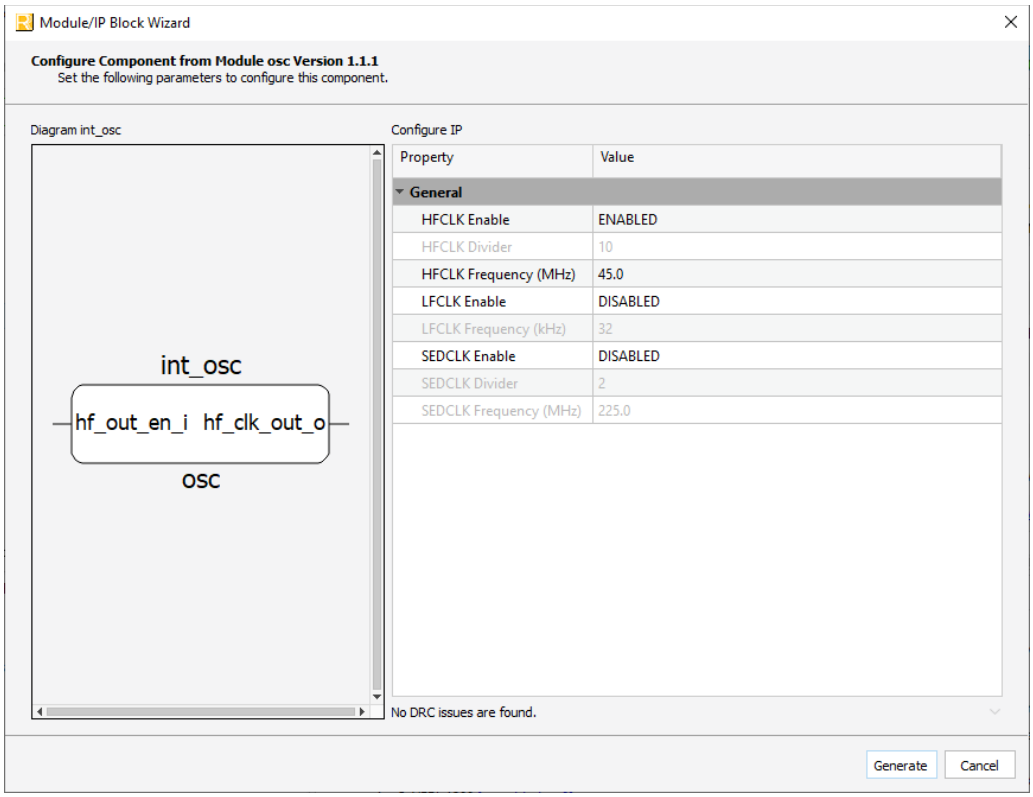


Figure 3.21. OSC IP Creation

4. Design and File Modification by User

This Reference Design is based on version 1.1.1 of the SubLVDS Image Sensor Receiver IP, version 1.3.0 of the Pixel-to-Byte Converter IP, and version 1.4.0 of the CSI-2/DSI D-PHY Transmitter IP.

4.1. Top-level RTL

You have to change IP instance names if you generate IPs created in Radiant with different instance names in the sample project.

If you use I2C Slave module with a clock other than the internal oscillator clock, you will have to disable the int_osc (internal oscillator) instantiation and connect an appropriate clock to clk_i port of i2c_slave.

5. Design Simulation

The script file (SubLVDS_to_csi2_LFCPNX_msim.do) and testbench files are provided to run the functional simulation by ModelSim Lattice FPGA Edition 2020.3. If you follow the naming recommendations regarding design names and instance names when RX SubLVDS, Pixel-to-Byte, TX D-PHY, GPLL, and I²C IPs are created by Radiant, the following will be the only change required in the script file.

User Project directory shown by red box.

```
set project_dir ~/Desktop/SubLVDS_to_CSI2_LFCPNX
set sim_dir     $project_dir/simulation/lfcpnx
```

Figure 5.1. Script File Modification

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. It shows the following statements while running and doing data comparison:

```
# [227263190860][CSI-2_CHK] Frame 3, Line 4, Byte Count 1937 - 1940, payload data = 57 bb d3 b7 --- Data matches : 57 bb d3 b7
# [227263723980][CSI-2_CHK] Frame 3, Line 4, Byte Count 1941 - 1944, payload data = 58 88 76 06 --- Data matches : 58 88 76 06
# [227264257100][CSI-2_CHK] Frame 3, Line 4, Byte Count 1945 - 1948, payload data = 7f 64 50 a9 --- Data matches : 7f 64 50 a9
# h_lane_on = 1, pixel_cnt = 1900
# unit_cnt = 190, lane_no = 0, payload_cnt = 2375, payload[9:2] = e2 by rx_dat[1900] = 388
# h_lane_on = 1, pixel_cnt = 1901
# unit_cnt = 190, lane_no = 1, payload_cnt = 2376, payload[9:2] = 4a by rx_dat[1901] = 129
# h_lane_on = 1, pixel_cnt = 1902
# unit_cnt = 190, lane_no = 2, payload_cnt = 2377, payload[9:2] = 19 by rx_dat[1902] = 067
# h_lane_on = 1, pixel_cnt = 1903
# unit_cnt = 190, lane_no = 3, payload_cnt = 2378, payload[9:2] = 4b by rx_dat[1903] = 12d
### unit_cnt = 190, after lane_no = 3, payload_cnt = 2379, payload_LSBs = 74
# h_lane_on = 1, pixel_cnt = 1904
# unit_cnt = 190, lane_no = 4, payload_cnt = 2380, payload[9:2] = f5 by rx_dat[1904] = 3d7
# h_lane_on = 1, pixel_cnt = 1905
# unit_cnt = 190, lane_no = 5, payload_cnt = 2381, payload[9:2] = d9 by rx_dat[1905] = 367
# h_lane_on = 1, pixel_cnt = 1906
# unit_cnt = 190, lane_no = 6, payload_cnt = 2382, payload[9:2] = 74 by rx_dat[1906] = 1d3
# h_lane_on = 1, pixel_cnt = 1907
# unit_cnt = 190, lane_no = 7, payload_cnt = 2383, payload[9:2] = 55 by rx_dat[1907] = 157
### unit_cnt = 190, after lane_no = 7, payload_cnt = 2384, payload_LSBs = ff
# h_lane_on = 1, pixel_cnt = 1908
# unit_cnt = 190, lane_no = 8, payload_cnt = 2385, payload[9:2] = e6 by rx_dat[1908] = 399
# h_lane_on = 1, pixel_cnt = 1909
# unit_cnt = 190, lane_no = 9, payload_cnt = 2386, payload[9:2] = 26 by rx_dat[1909] = 099
# [227264790220][CSI-2_CHK] Frame 3, Line 4, Byte Count 1949 - 1952, payload data = f0 6f a8 6c --- Data matches : f0 6f a8 6c
# [227265323340][CSI-2_CHK] Frame 3, Line 4, Byte Count 1953 - 1956, payload data = 1d cb 76 95 --- Data matches : 1d cb 76 95
# [227265856460][CSI-2_CHK] Frame 3, Line 4, Byte Count 1957 - 1960, payload data = cf 7f b0 d3 --- Data matches : cf 7f b0 d3
```

Data Comparison

Simulation halts when data comparison fails. Following statements are shown when simulation is completed.

```
#          228972874200 : ### EAV Insertion ###
#
#          230352655400 : ### SAV Insertion ###
#
#          230679191400 : ### EAV Insertion ###
#
#          232059305800 : ### SAV Insertion ###
#
#          232385841800 : ### EAV Insertion ###
#
#          232392505800 : End of frame :           2
#
# Total payload bytes checked were 28800 = 3 Frames x 2400 bytes (1920 pixels) x 4 lines : all matched !!!!!
# Simulation Succeeded !!!!!
# TEST END!!!
```

The above shows results after running three frames.

Figure 5.2 shows the global timing of 10-lane RX and 4-lane TX. In this case, the first line is trimmed.

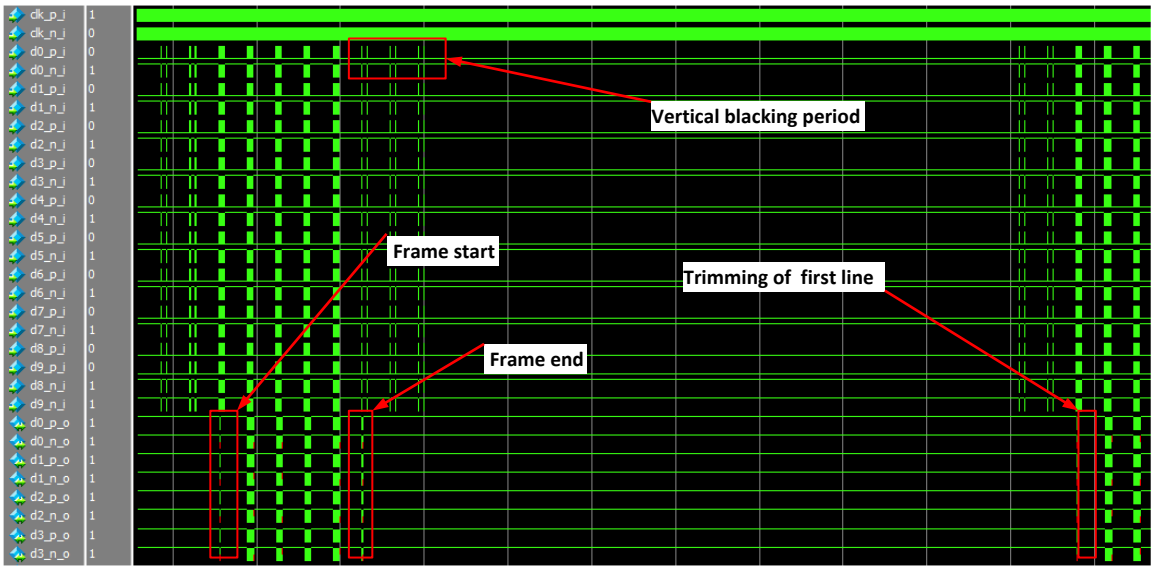


Figure 5.2. Global Timing of 10-Lane RX and 4-Lane TX

Figure 5.3 shows the close-up of the above waveform of one line period. Non-continuous clock mode is selected in TX D-PHY in this case so that CSI-2 TX clock lane goes into LP (Low Power) mode after the data lanes go into LP mode, then comes back to HS (High Speed) mode followed by the data lane transition to HS mode.

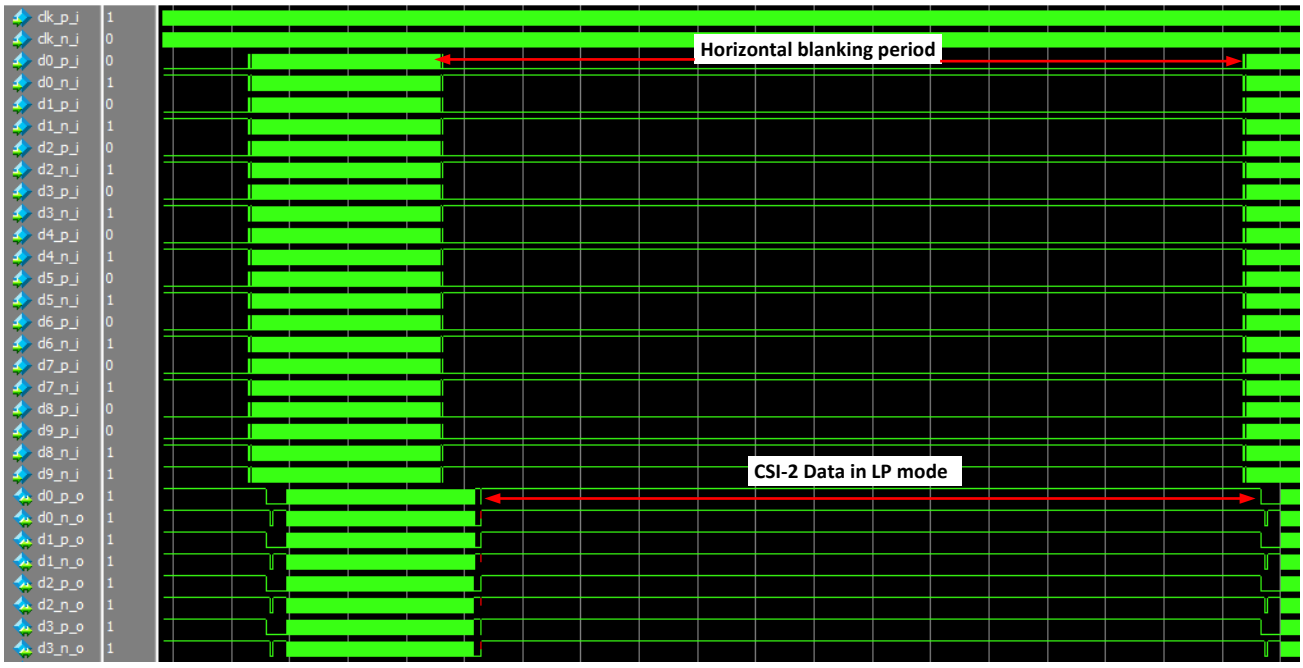


Figure 5.3. CSI-2 LP/HS Mode Transitions

Figure 5.4 shows the global timing of RAW12 , 6-lane RX to 1-lane TX with Sensor Master Mode. FPGA generates xvs_o and xhs_o as vertical and horizontal sync signals.

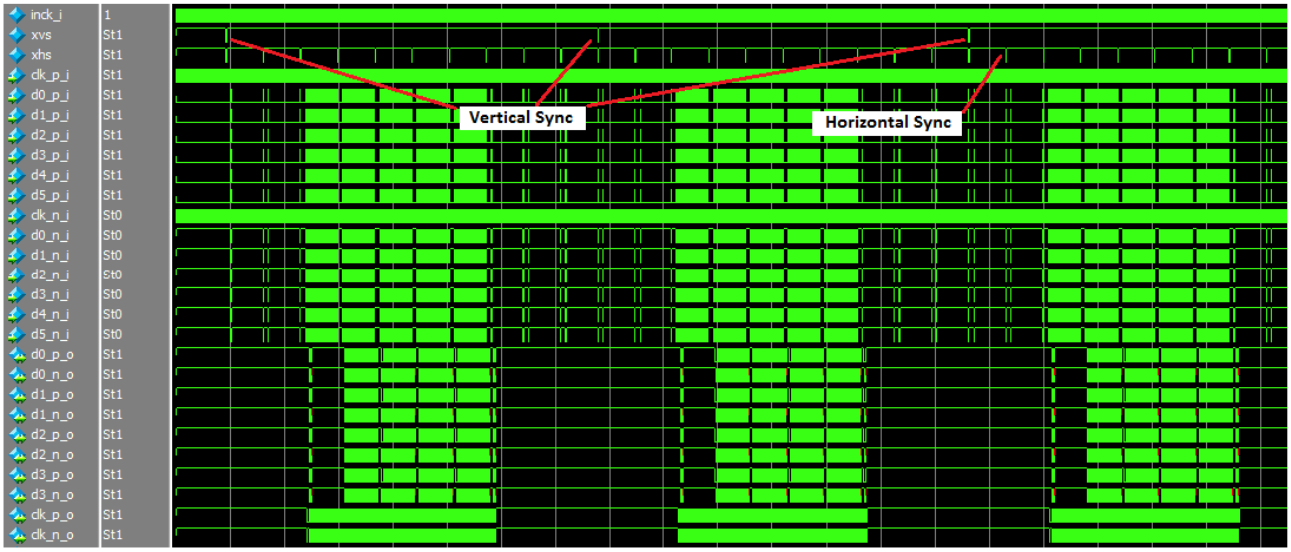


Figure 5.4. Global Timing with Sensor Slave Mode

6. Design Debug on Hardware

Hardware debug on MIPI related design is a challenge when the system is not working and no idea where the issue comes from, including it is by FPGA or not. This section shows some guidelines to find where the problem is happening.

Note that what described here gives some ideas how to debug the system including FPGA functionality, but that does not mean this can cover all possible scenarios.

6.1. Top-Level

In any design cases, the following are essential check points 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. 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. Along with Non-Continuous clock mode you can enable `define KEEP_HS in synthesis_directives.v as described in lane_ctrl section in case that you cannot let clock lane go into LP mode during the horizontal blanking period.

[Timing Parameter Modification]

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 3.18. These values are based on the byte clock cycles.

7. Known Limitations

The following are the limitations of this reference design.

- The first line must be always trimmed by either rx_sublvds or trim_ctrl.
- Granularity of CSI-2 output video data is coarser than CSI-2 specification for both RAW10 and RAW12 as described in trim_ctrl section.
- The design does not support RX Gear 16. It also does not support hs_byte_clk ratio of 0.5x and above 4x.

8. Design Package and Project Setup

SubLVDS to MIPI CSI-2 Image Sensor Bridge Reference Design with CertusPro-NX is available on www.latticesemi.com. Figure 8.1 shows the directory structure. The design is targeted for LIFCL-40. synthesis_directives.v and simulation_directives.v are set to configure an example shown below:

- RX – RAW10, 10 lanes, Gear 8, I^2C enabled
- TX – 4 lanes with Gear 8

You can modify the directives for your own configuration.

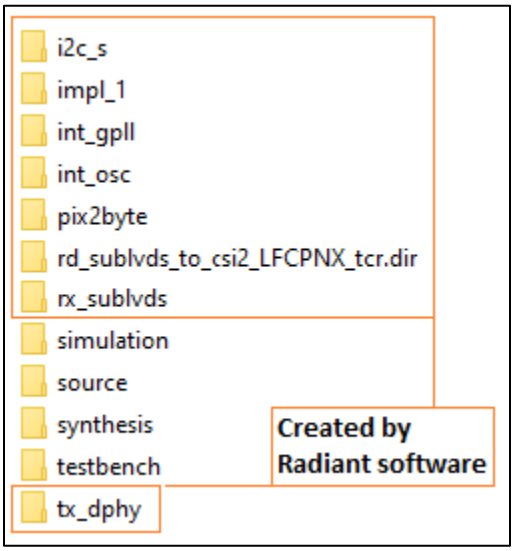


Figure 8.1. Directory Structure

Folders i2c_s, int_gpll, int_osc, pix2byte, rx_sublvds, and tx_dphy are created by Radiant for corresponding IPs. Figure 8.2 shows design files used in the Radiant project. Radiant creates six .ipx files.

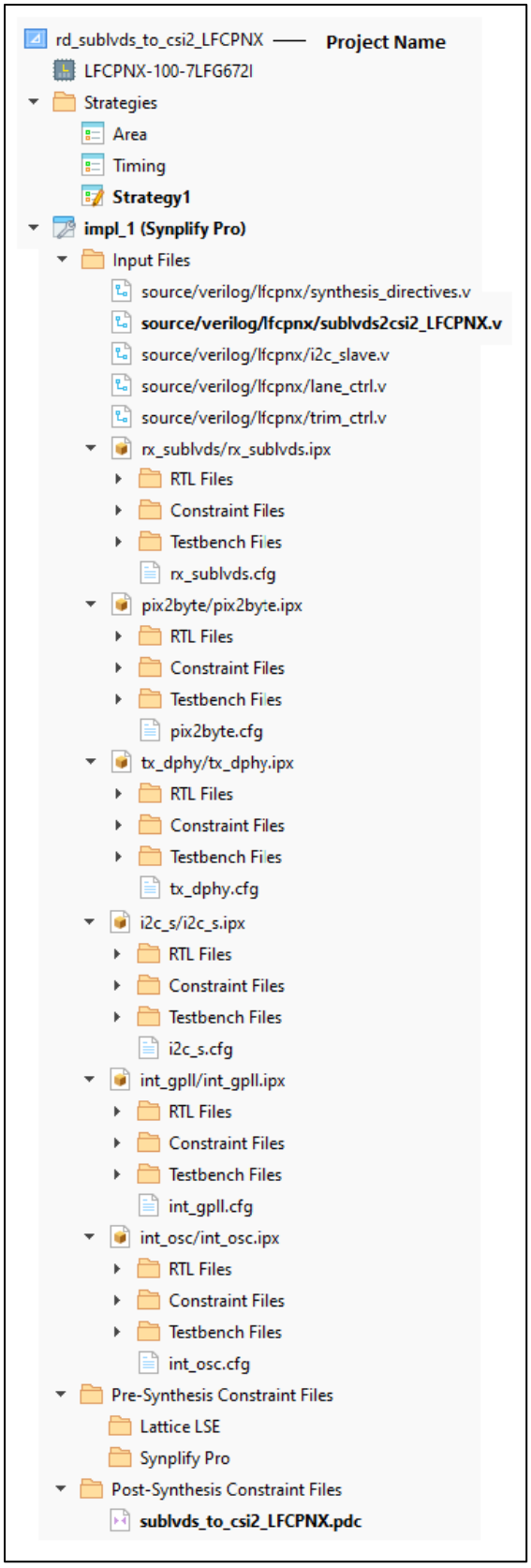


Figure 8.2. Project Files

9. Resource Utilization

Resource utilization depends on the configurations. Table 9.1 shows resource utilization examples under certain configurations. Actual usage may vary.

Table 9.1. Resource Utilization Examples

Configuration	LUT (Utilization/Total)	FF (Utilization/Total)	EBR (Utilization/Total)	I/O (Utilization/Total)
RAW10 RX 4 lanes, Gear 8 to TX 2 lanes with I2C, Sensor Slave Mode	2143/79872	1402/80769	5/208	17/299
RAW10 RX 8 lanes, Gear 8 to TX 4 lanes with I2C, Sensor Master Mode	2900/79872	1892/80769	8/208	22/299
RAW10 RX 10 lanes, Gear 8 to TX 4 lanes with I2C, Sensor Slave Mode	4146/79872	2144/80769	8/208	27/299
RAW12 RX 8 lanes, Gear 8 to TX 4 lanes with I2C, Sensor Master Mode	3118/79872	2013/80769	8/208	22/299
RAW12 RX 10 lanes, Gear 8 to TX 4 lanes with I2C, Sensor Master Mode	4014/79872	2238/80769	10/208	24/299

References

- MIPI® Alliance Specification for D-PHY Version 1.2
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.2
- SubLVDS Image Sensor Receiver IP Core User Guide (FPGA-IPUG-02093)
- Pixel-to-Byte Converter IP Core User Guide (FPGA-IPUG-02094)
- CSI-2/DSI D-PHY Transmitter IP Core User Guide (FPGA-IPUG-02080)
- I^2C Slave IP Core User Guide (FPGA-IPUG-02072)

For more information on the CertusPro FPGA device, visit

https://www.latticesemi.com/Products/FPGAandCPLD/CertusPro-NX

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

Technical Support Assistance

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

Revision History

Revision 1.0, March 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com