MIPI to Parallel with CertusPro-NX

Reference Design

FPGA-RD-02238-1.2

February 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 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
CSI-2	Camera Serial Interface 2
DE	Data Enable
DSI	Display Serial Interface
EBR	Embedded Block RAM
FIFO	First In, First Out
GPLL	General Purpose PLL
HS	High Speed
HSYNC	Horizontal Sync
LP	Low Power
LUT	Look Up Table
MIPI	Mobile Industry Processor Interface
PHY	Physical Layer
PLL	Phase Locked Loop
RX	Receiver
STA	Static Timing Analysis
TX	Transmitter
VSYNC	Vertical Sync

1. Introduction

The Mobile Industry Processor Interface (MIPI®) D-PHY is developed primarily to support camera and display interconnections in mobile devices, and it has become the industry's primary high-speed PHY solution for these applications in smartphones. It is typically used in conjunction with MIPI Camera Serial Interface-2 (CSI-2) and MIPI Display Serial Interface (DSI) protocol specifications. It meets the demanding requirements of low power, low noise generation, and high noise immunity that mobile phone designs demand.

MIPI D-PHY is a practical PHY for typical camera and display applications. It is designed to replace traditional parallel bus based on LVCMOS or LVDS. However, many processors and displays/cameras still use RGB, CMOS, or MIPI Display Pixel Interface (DPI) as interface.

The Lattice Semiconductor MIPI to Parallel with CertusPro™-NX reference design allows quick interface between a processor with a MIPI DSI and a display using RGB; or between a camera with a MIPI CSI-2 and a processor with a Parallel interface. The reference design provides this conversion for Lattice Semiconductor CertusPro-NX devices. This is useful for wearables, tablets, human machine interfacing, medical equipment, and many other applications.

1.1. Supported Device and IP

This reference design supports the following device with IP versions.

Device Family	Part Number	Compatible IP
CertusPro-NX	LFCPNX-100	D-PHY Receiver IP version 1.4.3 Byte-to-Pixel Converter IP version 1.4.0

The IPs above are supported by Lattice Radiant™ software version 2022.1 or later.

1.2. Features List

The key features of the MIPI to Parallel with CertusPro-NX reference design are:

- Compliance with MIPI D-PHY v1.2, MIPI DSI v1.2, and MIPI CSI-2 v1.2 specifications
- MIPI D-PHY interfacing from 80 Mb/s up to 1.5 Gb/s
- One, two, or four data lanes and one clock lane
- Continuous and non-continuous MIPI D-PHY clock
- Support for common MIPI DSI compatible video formats (RGB888, RGB666)
- Support for common MIPI CSI-2 compatible video formats (RGB888, RAW8, RAW10, RAW12, RAW14)
- MIPI DSI Video Mode operation of Non-Burst Mode with Sync Pulses and Non-Burst Mode with Sync Events
- Dedicated End of Transmission short packet (EoTp)

1.3. Block Diagram

Figure 1.1 shows the block level diagram of the MIPI to Parallel Reference Design.

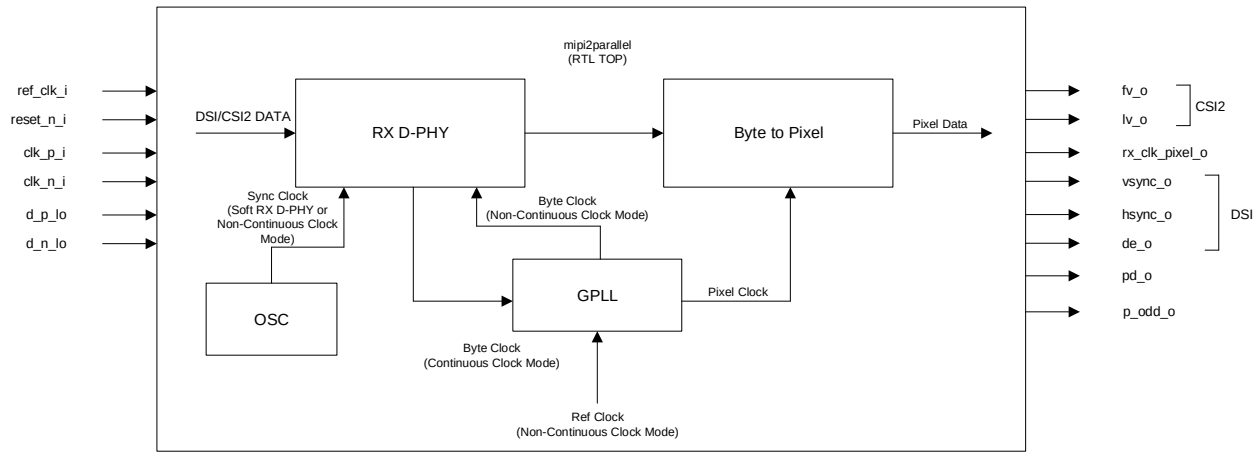

Figure 1.1. MIPI to Parallel Reference Design Block Diagram

As shown in the Figure 1.1, the block level diagram of the MIPI to Parallel reference design mainly consists of the RX D-PHY and Byte to Pixel IPs. There are two primary clocks for the main video data path: byte clock and pixel clock. Generally, a GPLL is used to generate the pixel clock. The same GPLL can also be used to generate the continuous byte clock when the RX D-PHY is in non-continuous clock mode. The input to the GPLL is byte clock in continuous clock mode or ref clock 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.

1.4. Functional Description

The MIPI D-PHY receive interface has one clock lane and a configurable number of data lanes. The clock lane is center aligned to the data lanes. The MIPI D-PHY clock can either be continuous (high speed only) or non-continuous.

When the MIPI D-PHY clock is non-continuous, proper transition from low power (LP) to high-speed (HS) mode of clock lane is required. The data lanes also require proper transition from LP to HS modes. In HS mode, data stream from each data lane is deserialized to byte data. The deserialization is done with 1:8 gearing. The byte data is word-aligned based on the SoT Sync sequence defined in the MIPI D-PHY Specification version 1.2.

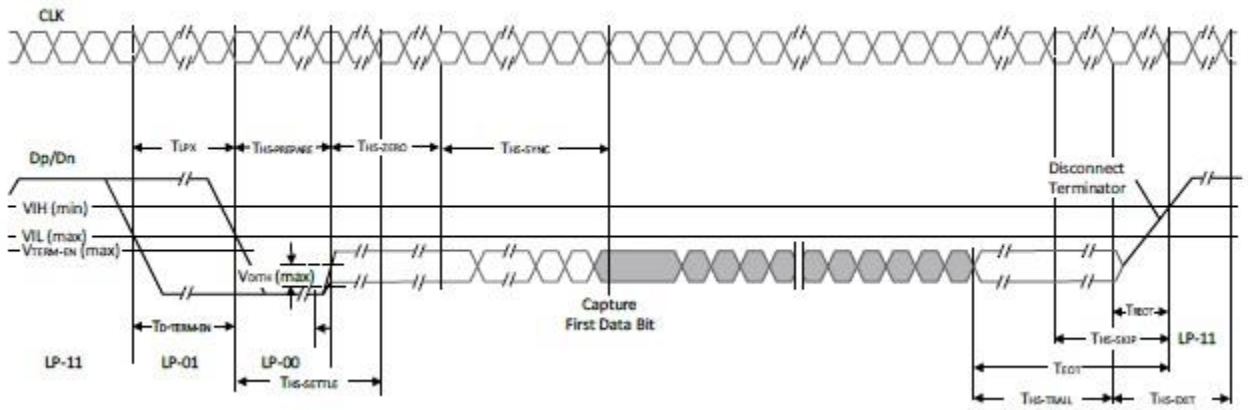

Figure 1.2. High Speed Data Transmission

The parallel transmit interface consists of clock, pixel data, and control signals. The pixel data width is configurable depending on the data type. The control signals are either data enable (DE), vertical and horizontal sync flags (VSYNC and HSYNC) for MIPI DSI applications or frame valid and line valid for MIPI CSI-2 applications.

The clock is edge-aligned against data and control signals. All signal transitions happen in sync with the rising edge of pixel clock as shown in Figure 1.3 and Figure 1.4.

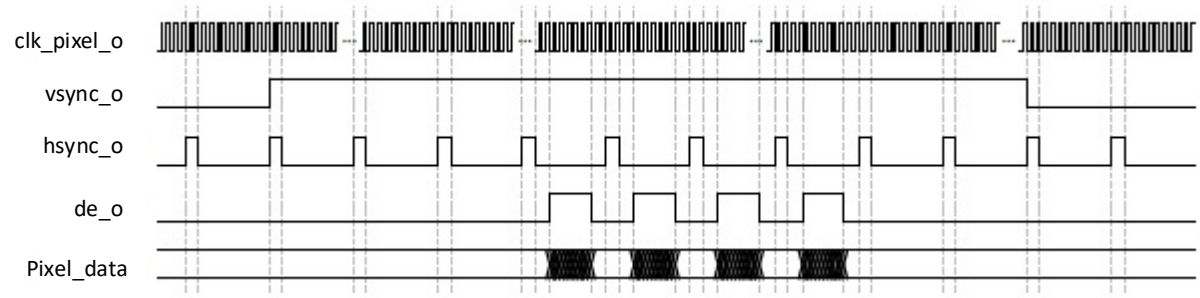

Figure 1.3. Parallel Transmit Interface Timing Diagram (DSI)

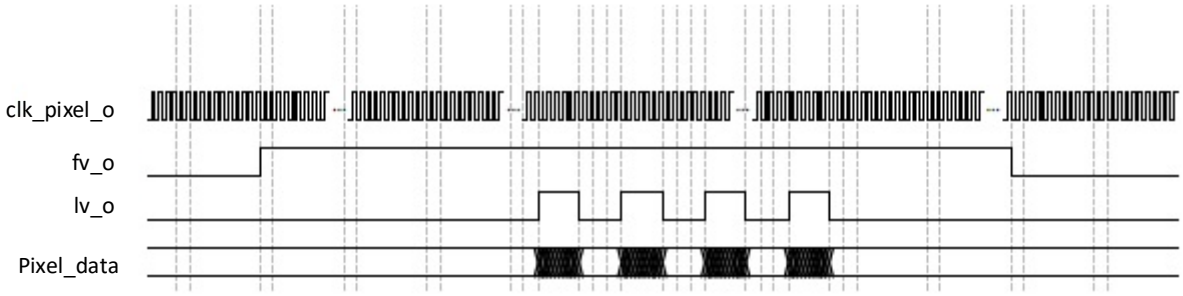

Figure 1.4. Parallel Transmit Interface Timing Diagram (CSI-2)

1.5. Conventions

1.5.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL. This includes radix indications and logical operators.

1.5.2. Data Ordering and Data Types

The highest bit within a data bus is the most significant bit.

1-bit data stream from each MIPI D-PHY data lane is deserialized into 8-bit parallel data where bit 0 is the first received bit.

Table 1.1 lists the pixel data order coming from the core module.

Table 1.1. Pixel Data Order

Data Type	Format
RGB	{Red[MSB:0], Green[MSB:0], Blue[MSB:0]}
RAW	RAW[MSB:0]

1.5.3. Signal Names

Signal names that end with:

- *_n* are active low
- *_i* are input signals
- Some signals are declared as bidirectional (I/O) but are only used as input. Hence, *_i* identifier is used.
- *_o* are output signals
- Some signals are declared as bidirectional (I/O) but are only used as output. Hence, *_o* identifier is used.
- *_io* are bidirectional signals

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 D-PHY, Byte to Pixel, GPLL and OSC 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
RX Interface	RX_TYPE_DSI	D-PHY Receive Interface. Only one of these directives must be defined.
	RX_TYPE_CSI2	
Number of RX Lanes	NUM_RX_LANE_1	Number of lanes in RX interface. Only one of these three directives must be defined.
	NUM_RX_LANE_2	
	NUM_RX_LANE_4	
RX D-PHY Clock Gear	RX_GEAR_8	Rx Gear 8
Sync Signal Polarity	SYNC_POLARITY_POS	Sync signal (VSYNC, HSYNC) polarity. Only one of these two directives must be defined. Applicable only to DSI.
	SYNC_POLARITY_NEG	
RX D-PHY Clock Mode[1]	RX_CLK_MODE_HS_ONLY	RX D-PHY Clock mode. Only one of these two directives must be defined.
	RX_CLK_MODE_HS_LP	
RX Data Type	RX_RGB888	Data type on the D-PHY RX interface. Only one of these directives must be defined. Supported MIPI DSI data types: RGB888, RGB666 Supported MIPI CSI-2 data types: RGB888, RAW8, RAW10, RAW12, RAW14
	RX_RGB666	
	RX_RAW8	
	RX_RAW10	
	RX_RAW12	
	RX_RAW14	
Number of Pixels Per Clock	RX_PEL_PER_CLK x	Number of pixels per clock at output where x = 1, or 2. Only one of these values must be defined

Note: 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.

2.2. Simulation Directives

Table 2.2 shows the simulation directives for this reference design.

Table 2.2. Simulation Directives

Category	Directive	Remarks
Video data configuration on RX Interface	NUM_FRAMES {value}	Number of video frames fed by the testbench
	NUM_LINES {value}	Number of active lines per frame
	NUM_PIXELS {value}	Number of active video pixels per line
Blanking Mode	LP_BLANKING	Enable the Low Power Blanking during Vertical Blanking Period. If not defined, then High-Speed Blanking is used by default.[1]
Sync Type of Video Data	NON_BURST_SYNC_EVENTS	Enable Non-Burst Sync Event mode. If not defined, then Non-Burst Sync Pulse mode is used by default.[1]
RX D-PHY clock period	DPHY_CLK {value}	RX D-PHY clock period in ps
Vertical Sync Width	VSYNC_WIDTH {value}	Vertical SYNC width. This value must match the *number of HSYNC pulses inside VSYNC active region* setting of Byte to Pixel IP. [1]
Horizontal Sync Width	HSYNC_WIDTH {value}	Horizontal SYNC width. This value must match the *number of pix clock cycles HSYNC remains active* setting of the Byte to Pixel IP. [1]
Internal signal monitoring	DPHY_DEBUG_ON	Enables D-PHY internal signal monitoring by the testbench.

Note:

1. Applicable to DSI Mode only.

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 configurations. All necessary I/O ports are automatically declared by compiler directives.

Table 2.3. MIPI to Parallel 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 RX_CLK_MODE_HS_LP is defined in simulation_directives.v.
reset_n_i	I	Asynchronous active low system reset
MIPI D-PHY RX Interface		
clk_p_i	I/O	Positive differential RX D-PHY input clock
clk_n_i	I/O	Negative differential RX D-PHY input clock
d_p_io[BUS_WIDTH – 1:0][1]	I/O	Positive differential RX D-PHY input data
d_n_io[BUS_WIDTH – 1:0][1]	I/O	Negative differential RX D-PHY input data
Parallel Interface		
clk_pixel_o	O	Pixel clock generated from internal GPLL
vsync_o	O	Vertical Sync Indicator (active high/low). Goes high/low when VSYNC start short packet is received. Goes low/high when VSYNC end short packet is received. Available only for MIPI DSI mode[2]
hsync_o	O	Horizontal Sync Indicator (active high/low). Goes high when either VSYNC/HSYNC start or VSYNC end short packet is received. Goes low when HSYNC end short packet is received. Available only for MIPI DSI mode[3]
de_o	O	Data Enable Indicator (active high). Goes high at the start of valid pixel data and goes low at the end of valid pixel data. Available only for MIPI DSI mode.
fv_o	O	Frame Valid Indicator (active high). Goes high when frame start short packet is received and goes low when frame end short packet is received. Available only for MIPI CSI-2 mode.
lv_o	O	Line Valid Indicator (active high). Goes high at the start of valid pixel data and goes low at the end of valid pixel data. Available only for MIPI CSI-2 mode.
pd_o[PD_BUS_WIDTH*RX_PEL_PER_CLK - 1:0][4,5]	O	Pixel data output. Data width depends on selected data type and Pixel per clock configurations.
p_odd_o[1:0]	O	This signal is used to indicate the valid pixels for the last valid pixel data cycle in case of multiple pixel outputs per pixel clock cycle. 00 – All pixels are valid 01 – Only the first pixel (LSB) is valid 10 – Only the lower two pixels in the lower bits are valid 11 – The last pixel (MSB) is not valid

Notes:

1. BUS_WIDTH – Number of D-PHY Lanes 1, 2, or 4 (available on the user interface).
2. In case of Non-Burst Sync Events, VSYNC end short packet is not received, and in that case vsync_o goes high/low as specified in VSYNC_WIDTH in simulation_directives.v
3. In case of Non-Burst Sync Events, HSYNC end short packet is not received, and in that case hsync_o goes high/low as specified in HSYNC_WIDTH in simulation_directives.v
4. PD_BUS_WIDTH - It can be 8, 10, 12, 14, 18 or 24 as per the selected data type RAW8, RAW10, RAW12, RAW14, RGB666, and RGB888 respectively. It can be 18 or 24 for RGB666 or RGB888 respectively for DSI. It can be 8, 10, 12, 14 or 24 for RAW8, RAW10, RAW12, RAW14 or RGB888 respectively for CSI-2.
5. RX_PEL_PER_CLK – Number of Pixels per Clock as defined in the synthesis_directives.v.

3. Design and Module Description

The top-level design (mipi2parallel_LFCPNX.v) consists of the following modules:

- rx_dphy
- b2p
- int_osc
- int_gpll

3.1. rx_dphy

This module must be created for the RX interface according to the required configuration, such as the number of lanes, bandwidth, and others. Figure 3.1 and Figure 3.2 show an example of IP interface settings in Lattice Radiant for the CSI-2/DSI D-PHY Receiver Submodule IP. Refer to CSI-2/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081) for details.

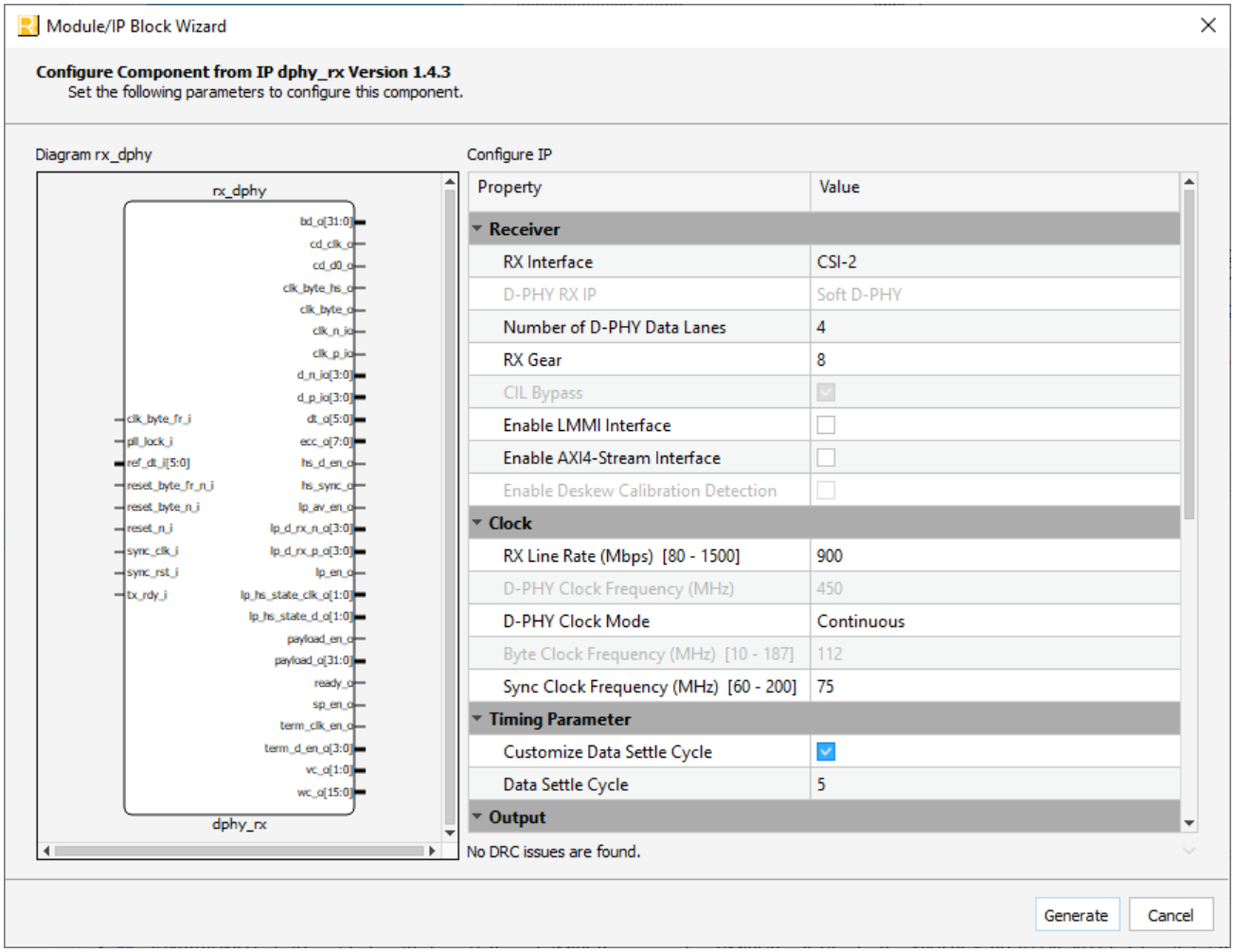

Figure 3.1. rx_dphy IP Creation in Lattice Radiant (1/2)

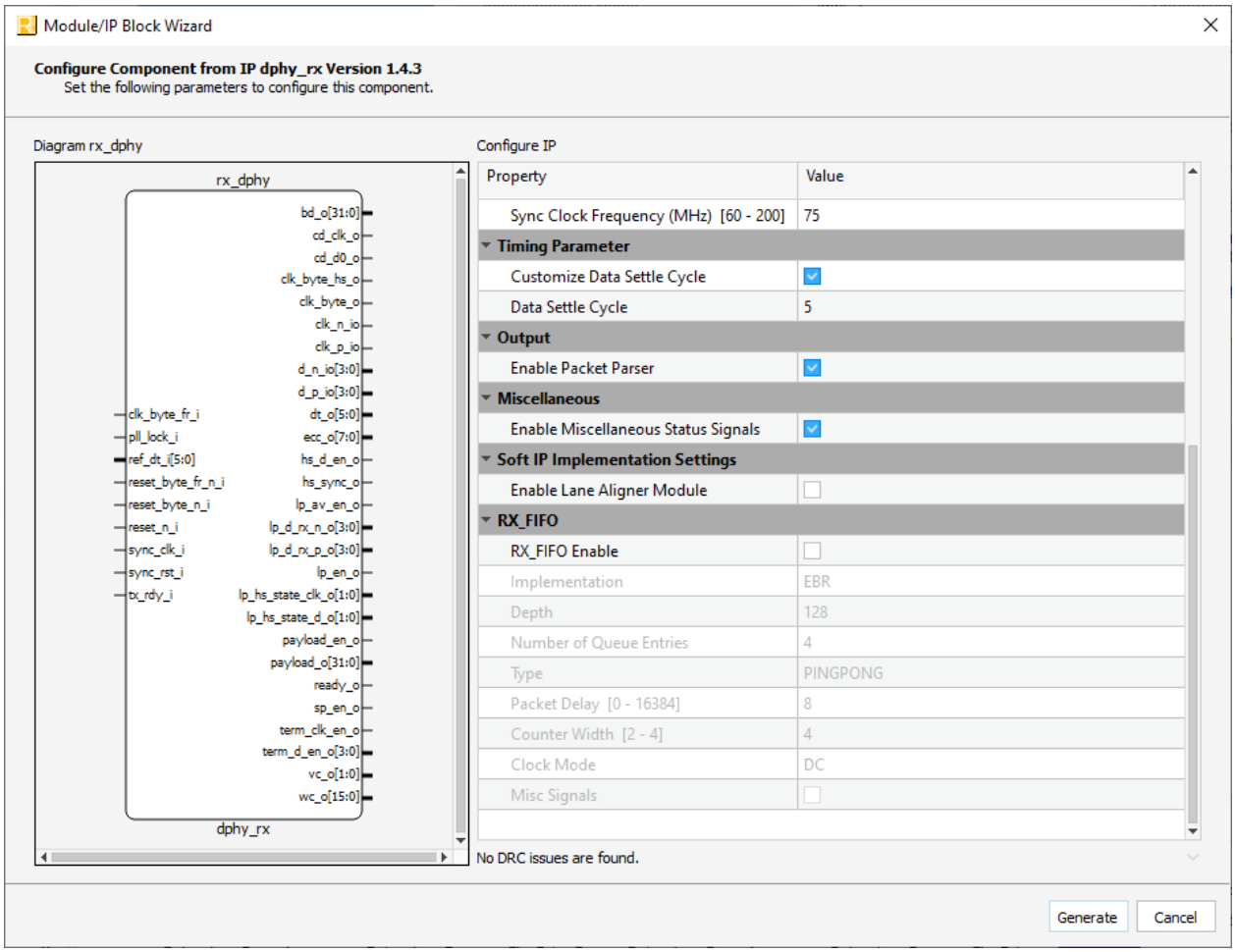

Figure 3.2. rx_dphy IP Creation in Lattice Radiant (2/2)

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

- RX Interface – Select CSI-2 or DSI.
- D-PHY RX IP – CertusPro-NX supports only SOFT DPHY type.
- Number of RX Lanes – Set according to RX interface configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Select 8;
- CIL Bypass – Select checkbox to enable (checked).
- Enable LMMI Interface – Select disabled (unchecked).
- Enable AXI4-Stream Interface – Select disabled (unchecked).
- RX Line Rate – Set according to the RX interface configuration. 1500 is the maximum for Soft D-PHY.
- 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 checkbox to enable (checked).
- Enable Miscellaneous Status Signals – Select enable (checked).
- Enable Lane Aligner Module – Select disabled (unchecked).
- RX_FIFO Enable – Set according to channel configuration. For RX SOFT DPHY it can be enabled or disabled.
- RX FIFO Memory implementation – Set according to channel configuration.
- RX FIFO Depth – Set according to channel configuration. Minimum value 16 is required.
- RX FIFO Type – Set according to channel configuration.
- RX FIFO Packet delay – Set according to channel configuration. Minimum value 1 is required.

- RX Clock mode – Always set to DC (Dual clock) for RX SOFT DPHY.
- RX FIFO Misc Signals – Always enable (checked) FIFO Miscellaneous signals for SOFT DPHY channels.

This module takes serial CSI-2/DSI data and outputs byte data after de-serialization in MIPI High Speed mode. The .ipx file included in the project (rx_dphy/rx_dphy.ipx) can be used to reconfigure the IP as per the user configuration requirements. If you are creating this IP from scratch, it is recommended to set the design name to *rx_dphy* so that you do not need to modify the instance names of these IPs in mipi2parallel_LFCPNX.v as well as the simulation setup file. Otherwise, you need to modify the names accordingly.

3.1.1. 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 before the word aligner in case of Soft D-PHY RX IP.

3.1.1.1. Soft D-PHY in Continuous Clock Mode

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

3.1.1.2. Non-Continuous Clock Mode

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

- Continuous Byte Clock = 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 µs, then we have 10 % of extra time to process the active data. This means the continuous byte clock can be as slow as ~-10% comparing to the non-continuous byte clock to avoid RX FIFO overflow.
- Continuous Byte Clock > Non-Continuous Byte Clock

  There are two options in this case:
  - Use type = 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 overhead 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 a 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.

Sometimes 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). In this case, 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.

3.2. b2p

This module must be created for the RX interface according to the required configuration, such as data type, the number of lanes, RX Gear, and others. Figure 3.3 shows an example of IP interface settings in Lattice Radiant software for the byte2pixel IP. Refer to Byte-to-Pixel Converter IP Core User Guide (FPGA-IPUG-02079) for details.

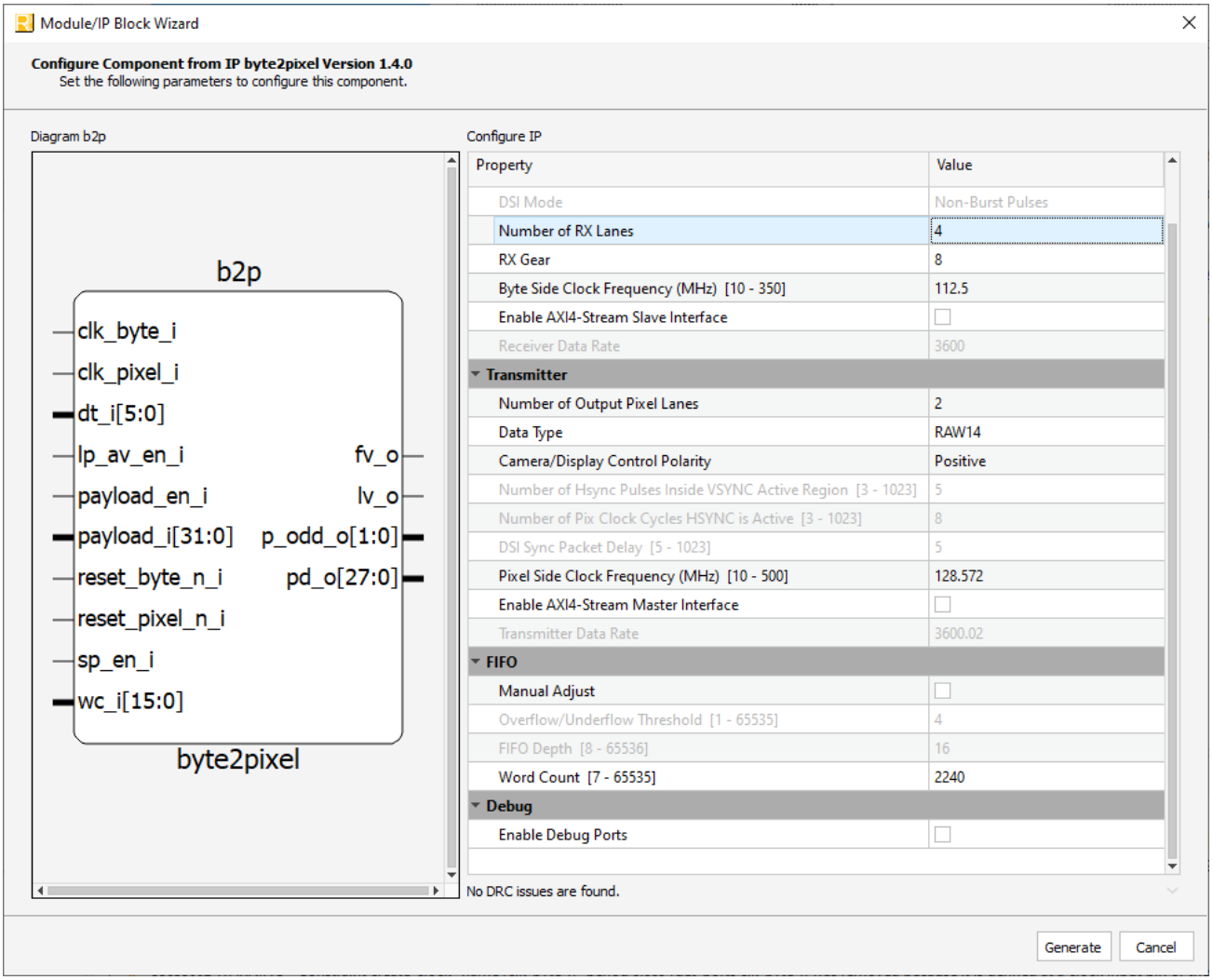

Figure 3.3. b2p IP Creation in Lattice Radiant

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

- RX Interface – Select DSI or CSI-2. Set the same type as RX D-PHY IP.
- DSI Mode – Select Non-Burst Pulses (applicable to DSI Interface only)
- Number of RX Lanes – Select 1, 2 or 4. Set the same value as RX D-PHY IP.
- RX Gear – Select 8. Set the same value as RX D-PHY IP.
- Byte Side Clock Frequency – Set the same value as is obtained in the RX D-PHY IP.
- Enable AXI4-Stream Slave Interface – Select disabled (unchecked).
- Number of Output Pixels – Select 1, or 2.
- Data Type – Select RGB888 or RGB666 for DSI and RGB888, RAW8, RAW10, RAW12, or RAW14 for CSI-2. Others are not supported in this reference design.
- Camera/Display Control Polarity – Always Select Positive. if user wants to use Negative polarity, still select this options as Positive. And use Negative polarity define from Synthesis directive.
- Pixel Side Clock Frequency – Enter the appropriate value.
- Enable AXI4-Stream Master Interface – Select disabled (unchecked).
- Manual Adjust – Select disabled (unchecked).
- Word Count – Enter the appropriate value as per the following equation:
  Word Count = (NUM_PIXELS * PD_BUS_WIDTH)/8
  For example: In case of 1080P with RAW10, the value is 1920 x 10/8 = 2400.
- Enable Debug Ports – Select disabled (unchecked).

The Byte-to-Pixel Converter IP converts the D-PHY CSI-2/DSI standard based byte data stream to standard pixel data format. The .ipx file included in the project (b2p/b2p.ipx) can be used to reconfigure the IP as per the user configuration requirements. If you are creating this IP from scratch, it is recommended to set the design name to *b2p* so that you do not need to modify the instance name of this IP in the top-level design as well as in the simulation setup file. Otherwise, you need to modify the names accordingly.

3.3. int_osc

This module generates ~75 MHz clock used for sync clock and rx_clk_lp_ctrl. The 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 another continuous clock greater than 60 MHz is avaiable, you are able to use that without instantiating this module. Figure 3.4 shows an example of IP interface settings in Lattice Radiant software for the byte2pixel IP.

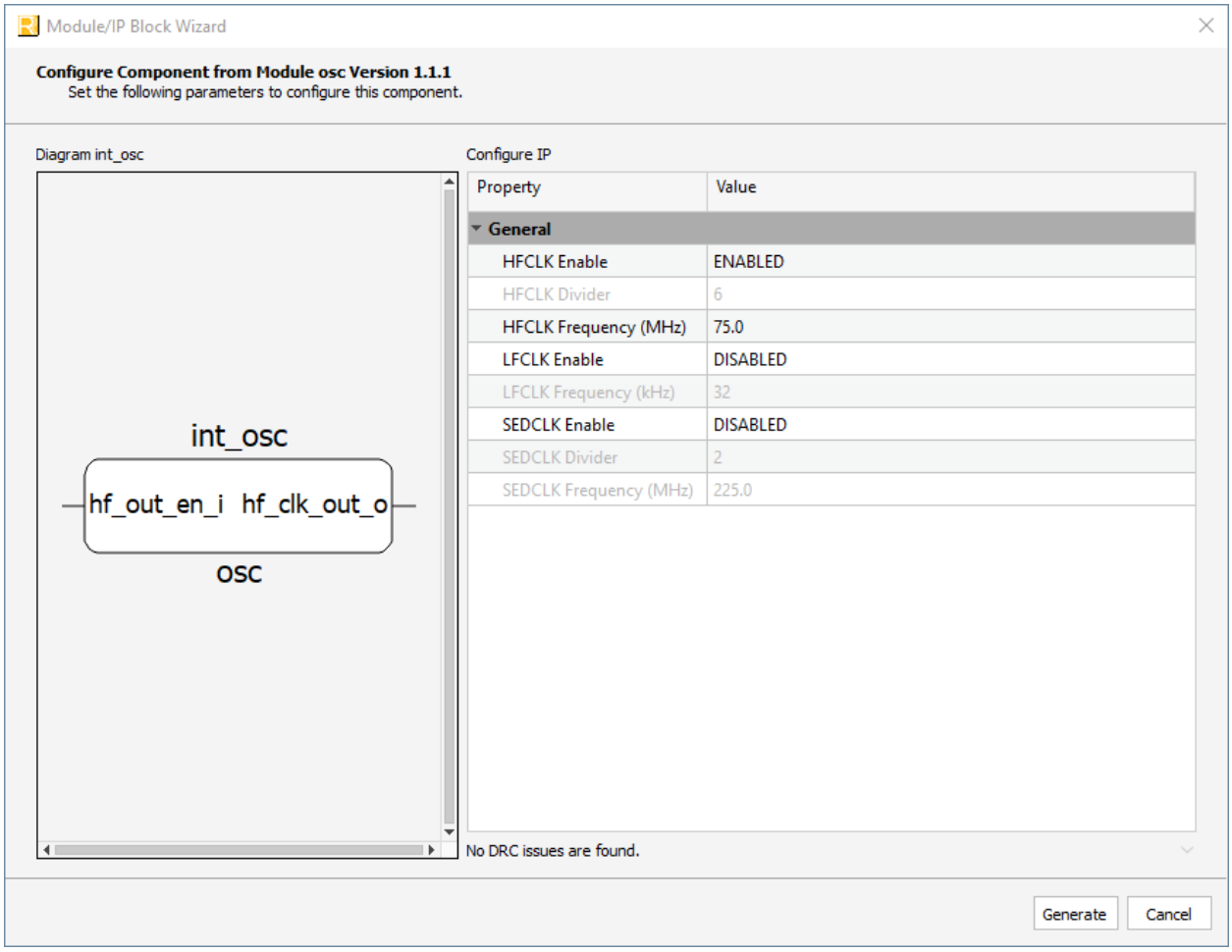

Figure 3.4. int_osc IP Creation in Lattice Radiant

3.4. int_gpll

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

*Pixel Clock = (byte clock * number of RX lanes * 8) / data width*

In most cases, you need to create GPLL module to generate pixel clock from the continuous byte clock except for RAW8 with 1-lane configuration. In non-continuous clock mode, this module generates continuous byte clock used for RX interface and the pixel clock for the pixel data. If the RX D-PHY is in continuous clock mode or continuous byte clock for RX D-PHY in non-continuous clock mode is already available, and the pixel clock is also available, then this module is not necessary.

Figure 3.5 shows an example of IP interface settings in Lattice Radiant for the GPLL IP. Refer to PLL Module User Guide (FPGA-IPUG-02063) for details.

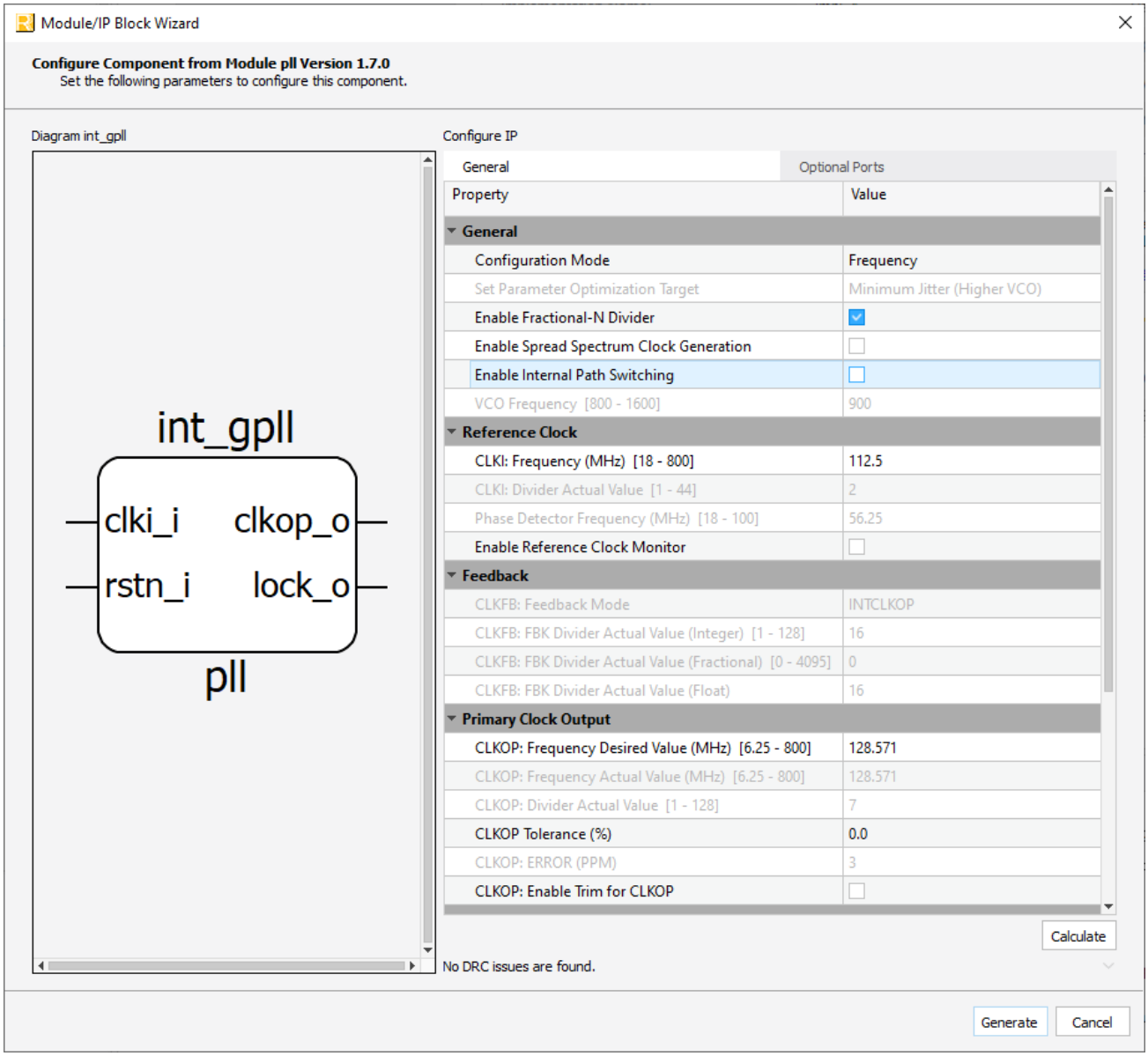

Figure 3.5. int_gpll IP Creation in Lattice Radiant

The .ipx file included in the project (int_gpll/int_gpll.ipx) can be used to reconfigure the IP as per the user configuration requirements. If you are creating 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 names of these IPs in mipi2parallel_LFCPNX.v file. Otherwise, you need to modify the name accordingly.

The following code snippet shows the condition in which pixel clock is generated directly from clkop_o of int_gpll in case of continuous clock mode.

```
int_gpll pll_inst (
  .rstn_i     (1'b1),
  .clki_i     (rx_clk_byte_fr),
  .clkop_o    (clk_pixel),
  .lock_o     (gpll_lock)
);
```

The following code snippet shows the condition in which byte clock and pixel clock is generated from int_gpll in case of non-continuous clock mode.

```
 int_gpll pll_inst (
  .rstn_i         (1'b1),
  .clki_i         (ref_clk_i),
  .clkop_o        (rx_clk_byte_fr_pll),
  .clkos_o        (clk_pixel),
  .lock_o         (gpll_lock)
);
```

More than one GPLL can be used if the required clock frequencies for the rx_clk_byte_fr_pll and clk_pixel cannot be generated using a single GPLL.

4. Design and File Modifications

This Reference Design is based on version 1.3.0 of the RX D-PHY IP and version 1.4.0 of the Byte2Pixel IP. Some modifications are required depending on user configuration in addition to the two directive files (synthesis_directives.v and simulation_directives.v).

4.1. Top Level RTL

The GPLL setting in the current top-level file (mipi2parallel_LFCPNX.v) is based on the configuration selected for this reference design. According to the configuration you use, you need to modify the GPLL instantiation as described in int_gpll section. Also, you can remove the int_osc module if the appropriate sync clock and rx_clk_lp_ctrl are already available. In addition, an instance names of RX D-PHY (rx_dphy), Byte to Pixel (b2p), internal OSC (int_osc) and internal GPLL (int_gpll) need to be modified if you create this IP(s) with different name(s).

5. Design Simulation

The script file (mipi_to_parallel_LFCPNX_msim.do) and testbench files are provided to run the functional simulation by Modelsim. If you follow the naming recommendations regarding design name and instance name when the RX D-PHY, Byte2Pixel, Internal OSC, Internal GPLL IPs are created by Lattice Radiant, the following are the only changes required in the script file:

- User project directory

```
### Set Customer's project/simulation directory ###
set project_dir C:/Users/destose/Pictures/1_Projects/MIPI_Regression_Final/MIPI2Parallel/CertusPro-NX/FPGA-RD-02238_2022.1
set sim_dir     $project_dir/simulation/lfcpnx
```

Own Project Directory

Figure 5.1. Script Modification #1

```
### Compiling modules ###
vlog \
+incdir+"$project_dir/int_gpll/rtl/" \
+incdir+"$project_dir/int_osc/rtl/" \
+incdir+"$project_dir/b2p/rtl/" \
+incdir+"$project_dir/rx_dphy/rtl/" \
+incdir+"$project_dir/testbench/verilog/" \
+incdir+"$project_dir/testbench/verilog/tb_include/" \
$project_dir/int_gpll/rtl/int_gpll.v \
$project_dir/int_osc/rtl/int_osc.v \
$project_dir/rx_dphy/rtl/rx_dphy.v \
$project_dir/b2p/rtl/b2p.v \
$project_dir/testbench/verilog/simulation_directives.v \
$project_dir/source/verilog/lfcpnx/synthesis_directives.v \
$project_dir/source/verilog/lfcpnx/mipi2parallel_LFCPNX.v \
$project_dir/testbench/verilog/mipi2parallel_LFCPNX_tb.v \
+define+SIM_STOP_AT_HSYNC_VSYNC_WIDTH_FAIL=1 \
+define+NUM_FRAMES=$num_frames+NUM_LINES=$num_lines \

vsim -voptargs=+acc=ap work.mipi2parallel_LFCPNX_tb  -L pmi_work -L ovi_lfcpnx
-c -do "add wave -r mipi2parallel_LFCPNX_tb/* ;run -all; quit" -t fs -suppress 3085
```

Figure 5.2. Script Modification #2

You need to modify *simulation_directives.v* according to your configuration (refer to Simulation Directives for details). By executing the script in Modelsim, compilation and simulation are executed automatically. The testbench takes all data comparison between the expected data and output data from the RD.

```
#            75000000 DPHY CLK CONT : Driving HS-Prpr
#            126000000 DPHY CLK CONT : Driving HS-Go
#            388000000 DPHY CLK CONT : Driving HS-0/HS-1
#         150245272000 Waiting for PLL lock...
#
#         150245272000 PLL lock DONE
#
#         150245272000 MIPI D-PHY Clock begins...
#
```

```
#          150345272000 Activating DSI/CSI-2 model
#
#          150345272000 DPHY CSI-2 model activated
#
#          230345272000 DPHY CSI-2 FS begins...
#
#          230346272000 DPHY CLK : Driving HS-CLK-RQST
#          230420272000 DPHY CLK : Driving HS-Prpr
#          230471272000 DPHY CLK : Driving HS-Go
#          230733272000 DPHY CLK : Driving HS-0/HS-1
#          230738600000 DPHY DATA : Driving HS-RQST
#          230812600000 DPHY DATA : Driving HS-Prpr
#          230863600000 DPHY CLK : Driving HS-Go
#          230964805000 DPHY CLK : Driving SYNC Data
#          230970465000 DPHY DATA : Driving FS
#          230970465000 DPHY DATA : Driving data = 00 00 00 00
#          230975793000 DPHY DATA : Driving HS-Trail
#          230975793000 DPHY 0 Lane 0 : Driving trail bytes
#          230975793000 DPHY 0 Lane 1 : Driving trail bytes
#          230975793000 DPHY 0 Lane 2 : Driving trail bytes
#          230975793000 DPHY 0 Lane 3 : Driving trail bytes
#          231064291000 DPHY DATA : Driving HS-Stop
#          231064291000 DPHY 0 Lane 0 : Driving stop
#          231064291000 DPHY 0 Lane 1 : Driving stop
#          231064291000 DPHY 0 Lane 2 : Driving stop
#          231064291000 DPHY 0 Lane 3 : Driving stop
##### FV assertion #####
#          231196572000 DPHY CLK : Driving CLK-Trail
#          231256572000 DPHY CLK : Driving CLK-Stop
#          236257572000 DPHY CLK : Driving HS-CLK-RQST
#          236331572000 DPHY CLK : Driving HS-Prpr
#          236382572000 DPHY CLK : Driving HS-Go
#          236644572000 DPHY CLK : Driving HS-0/HS-1
#          236649900000 DPHY DATA : Driving HS-RQST
#          236723900000 DPHY DATA : Driving HS-Prpr
#          236774900000 DPHY CLK : Driving HS-Go
#          236876221000 DPHY CLK : Driving SYNC Data
#          236881881000 DPHY Driving Data header
#          236881881000 DPHY Driving Data header for data = 2d 40 1a 2e
##### LV assertion #####
#          245838249000 DPHY Driving CRC[15:8] = c5; CRC[7:0] = 36
#          245838249000 DPHY 0 Lane 2 : Driving trail bytes
#          245838249000 DPHY 0 Lane 3 : Driving trail bytes
#          245843577000 DPHY 0 Lane 0 : Driving trail bytes
#          245843577000 DPHY 0 Lane 1 : Driving trail bytes
#          245932075000 DPHY 0 Lane 0 : Driving stop
#          245932075000 DPHY 0 Lane 1 : Driving stop
#          245932075000 DPHY 0 Lane 2 : Driving stop
#          245932075000 DPHY 0 Lane 3 : Driving stop
#          246027726000 DPHY CLK : Driving CLK-Trail
##### LV de-assertion #####
#          246087726000 DPHY CLK : Driving CLK-Stop
#          251088726000 DPHY CLK : Driving HS-CLK-RQST
#          251162726000 DPHY CLK : Driving HS-Prpr
#          251213726000 DPHY CLK : Driving HS-Go
#          251475726000 DPHY CLK : Driving HS-0/HS-1
#          251481054000 DPHY DATA : Driving HS-RQST
#          251555054000 DPHY DATA : Driving HS-Prpr
#          251606054000 DPHY CLK : Driving HS-Go
```

```
#          251708041000 DPHY CLK : Driving SYNC Data
#          251713701000 DPHY Driving Data header
#          251713701000 DPHY Driving Data header for data = 2d 40 1a 2e

When the simulation is finished, the following statements are displayed

# ------------------------------------------------
# **** I N F O : Byte Count is 60480
# **** I N F O : NUM_FRAMES=3, NUM_LINES=3, TOTAL BYTES =60480
# ------------------------------------------------
# ------------------------------------------------------
# ----------------- SIMULATION PASSED ------------------
# ------------------------------------------------------
# 404400032000 TEST END
#
```

Figure 5.3 shows the simulation waveform of the full view of four lines and two frames for the DSI interface. Figure 5.4 shows the zoom view of the simulation waveform shown in Figure 5.3 for the DSI interface. The waveform shows all the top level I/O and few other signals.

Figure 5.3. Simulation Waveform for DSI (1/2)

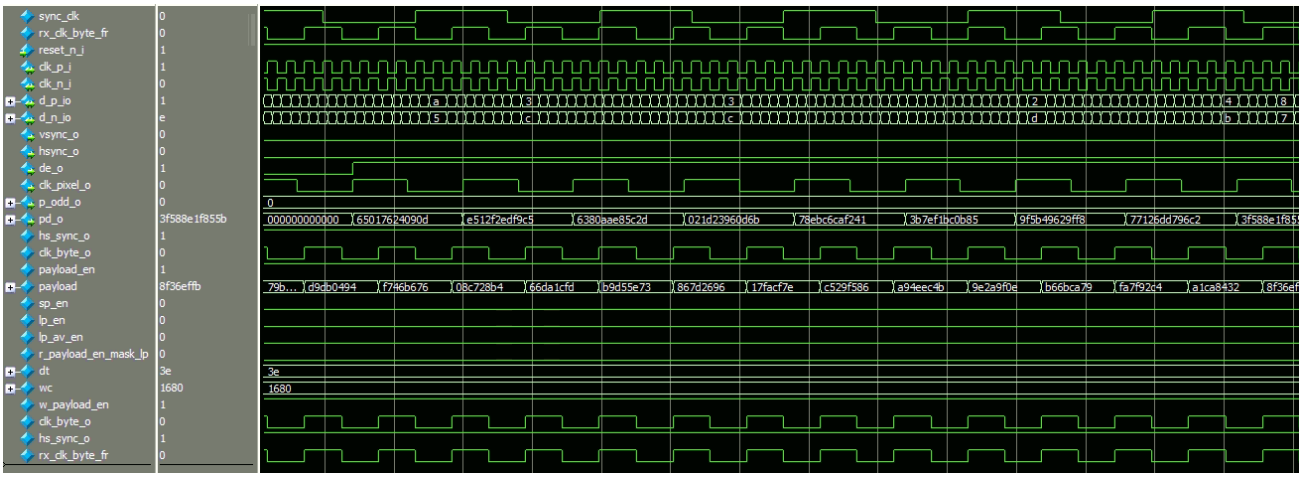

Figure 5.4. Simulation Waveform for DSI (2/2)

Figure 5.5 shows the simulation waveform of the full view of three lines and three frames for CSI-2 interface. Figure 5.6 shows the zoom view of the simulation waveform shown in Figure 5.5 for the CSI-2 interface. The waveform shows all the top level I/O and few other signals.

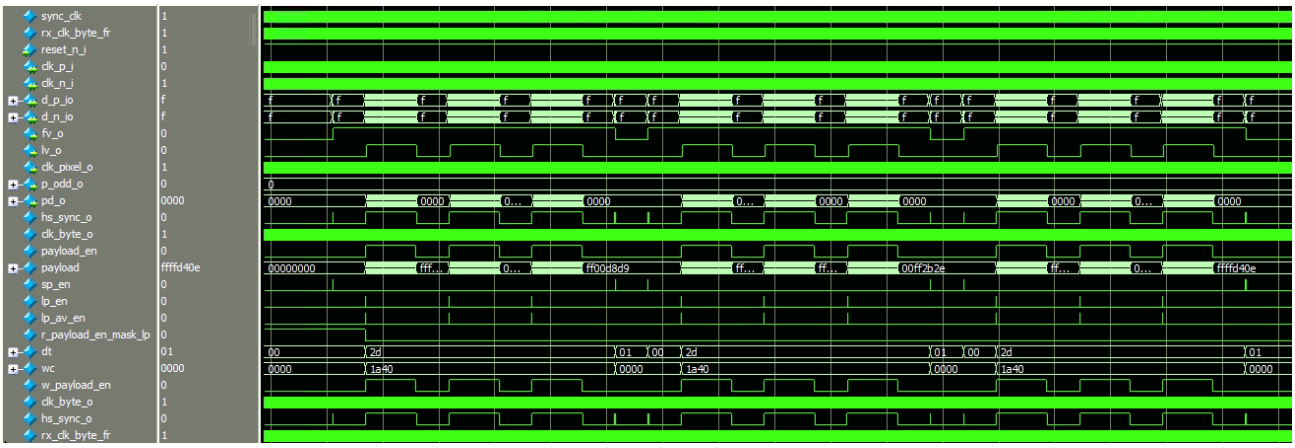

Figure 5.5. Simulation Waveform for CSI-2 (1/2)

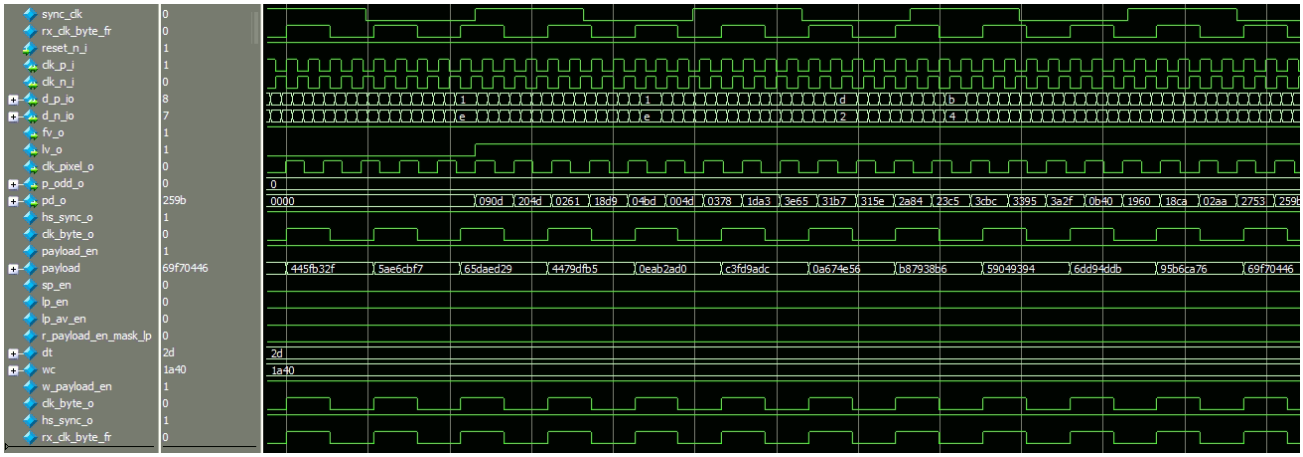

Figure 5.6. Simulation Waveform for CSI-2 (2/2)

The simulation waveform can be accessed by opening the *vsim.wlf* file in the Modelsim from the simulation directory. More signals of a module can be added to the waveform as required.

6. Known Limitations

The following are the limitations of this reference design:

- Only the following data types are supported for MIPI DSI interface: RGB888, RGB666
- Only the following data types are supported for MIPI CSI-2 interface: RGB888, RAW8, RAW10, RAW12, RAW14

7. Design Package and Project Setup

The MIPI to Parallel with CertusPro-NX Reference Design is available on www.latticesemi.com. Figure 7.1 shows the directory structure. The design is targeted for LFCPNX-100-7LFG672I. *synthesis_directives.v* and *simulation_directives.v* are set to configure the design with following configuration:

- RX – 4-lanes, Gear 8 with Soft D-PHY in continuous clock mode
- TX – CSI-2, RAW14 parallel data with 1 pixels/clock

You can modify the directives for your own configuration.

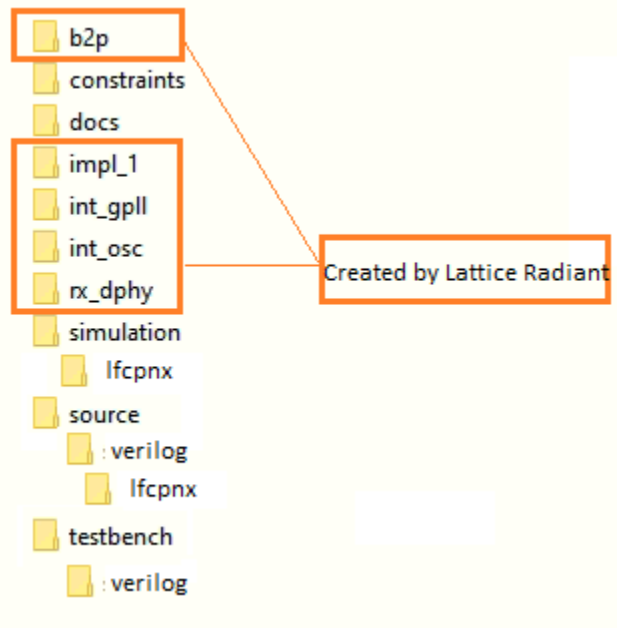

Figure 7.1. Directory Structure

Figure 7.2 shows the design files used in the Lattice Radiant project. Including PLL and oscillator modules, Lattice Radiant creates four .ipx files. By specifying mipi2parallel_LFCPNX as a top-level design, all unnecessary files are ignored. Constraint file (mipi_to_parallel_LFCPNX.pdc) is also included in the project for reference. You can modify it according to your own configuration.

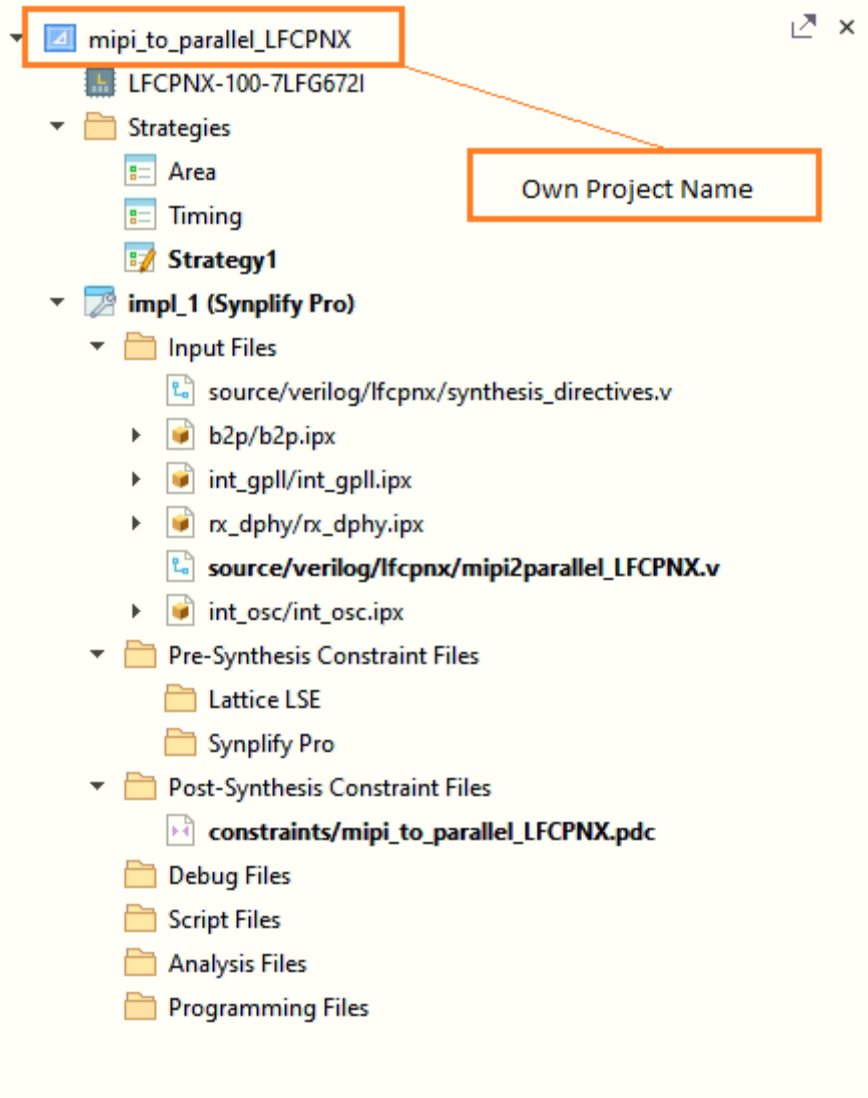

Figure 7.2. Project Files

8. Resource Utilization

Resource utilization depends on the configuration used. Table 8.1 shows the resource utilization examples under certain configurations targeting LFCPNX-100. This is just a reference and actual usage varies, especially when 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 (Utilization/Total)	FF (Utilization/Total)	EBR (Utilization/Total)	I/O (Utilization/Total)
4-lane, Gear 8, Soft D-PHY, CSI-2, RAW14, 1 Pixels/clock	1637/79872	1113/80769	0/208	30/299
2-lane, Gear 8, Soft D-PHY, CSI-2, RAW8, 2 Pixels/clock	879/79872	651/80769	1/208	28/299
4-lane, Gear 8, Soft D-PHY, DSI, RGB888, 2 Pixels/clock	1781/79872	1083/80769	2/208	65/299
1-lane, Gear 8, Soft D-PHY, DSI, RGB666, 1 Pixels/clock	990/79872	662/80769	2/208	30/299

References

- MIPI Alliance Specification for D-PHY Version 1.2
- MIPI Alliance Specification for Display Serial Interface 2 (DSI) Version 1.2
- MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.2
- CSI-2/DSI DPHY Rx IP Core User Guide (FPGA-IPUG-02081)
- Byte-to-Pixel Converter IP Core User Guide (FPGA-IPUG-02079)
- PLL Module User Guide (FPGA-IPUG-02063)

For more information on the CertusPro-NX 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 software user guide.

Technical Support Assistance

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

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

Revision History

Revision 1.2, February 2023

Section	Change Summary
Introduction	Updated Supported Device and IP section for below: • Changed D-PHY Receiver IP version from 1.3.0 to 1.4.3. • Changed the Radiant software version from 3.0.0.24.1 to 2022.1.
Design and Module Description	Updated below Figures: • Figure 3.1. rx_dphy IP Creation in Lattice Radiant (1/2) • Figure 3.2. rx_dphy IP Creation in Lattice Radiant (2/2) • Figure 3.3. b2p IP Creation in Lattice Radiant • Figure 3.5. int_gpll IP Creation in Lattice Radiant
Design Simulation	Updated below Figures: • Figure 5.1. Script Modification #1 • Figure 5.2. Script Modification #2
Technical Support Assistance	Added FAQ website link.

Revision 1.1, November 2021

Section	Change Summary
Introduction	Updated IP versions in the Supported Device and IP section.
Design and Module Description	Updated screen captures to reflect the associated IP versions.
Design and File Modifications	Updated IP versions in the section introduction.

Revision 1.0, September 2021

Section	Change Summary
All	Initial release.

www.latticesemi.com