# 1 to N MIPI CSI-2/DSI Duplicator with Crosslink

## Reference Design

FPGA-RD-02194-1.1

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

# Contents

## Figures

## Tables

# Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| AP | Application Processor |
| CRC | Cyclic Redundancy Check |
| CSI-2 | Camera Serial Interface 2 |
| DDR | Double Data Rate |
| DSI | Display Serial Interface |
| EBR | Embedded Block RAM |
| ECC | Error Correction Code |
| FS | Frame Start |
| HS | High Speed |
| ID | Identification Data |
| LP | Low Power |
| MIPI | Mobile Industry Processor Interface |
| PLL | Phase Locked Loop |
| GPLL | General Purpose PLL |
| RX | Receiver |
| STA | Static Timing Analysis |
| TX | Transmitter |
| VC | Virtual Channel |
| VSS | Vsync Start |
| WC | Word Count |

# 1. Introduction

Many new applications such as virtual reality, augmented reality, and digital cameras require expansion of the number of camera or display interface on application processors (AP). This often occurs when there is either not enough ports or some ports are used for other purposes on the AP.

The Lattice Semiconductor 1 to N MIPI CSI-2/DSI Duplicator reference design for CrossLink™ devices has one or two-channel outputs for duplicator. CrossLink has two MIPI hard macro IPs, which can be used as MIPI TX or RX module (D-PHY Hard IP). The RX module can also be realized by a soft macro utilizing general DDR modules (D-PHY Soft IP).

## 1.1. Supported Device and IP

This reference design supports the following devices with IP versions:

**Table 1.1. Supported Device and IP**

| Device Family | Part Number | Compatible IP |
|---|---|---|
| CrossLink | LIF-MD6000<br>LIA-MD6000 | D-PHY Receiver IP version 1.6<br>D-PHY Transmitter IP version 1.4 |
| CrossLinkPlus | LIF-MDF6000 | |

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

## 1.2. Features

- One RX channel is duplicated to one or two TX channels.
- RX channel can have one, two, or four lanes.
- Using Hard D-PHY IP is recommended on RX channel in case of single TX channel to save FPGA resources.
- Maximum RX bandwidth is 1.5 Gbps per lane using D-PHY Hard IP and 1.2 Gbps per lane using D-PHY Soft IP.
- MIPI D-PHY TX is not supported by Soft D-PHY IP and Hard D-PHY IP must be used on TX channel(s).
- Number of TX lanes can be one, two, or four. This is independent from the number of RX lanes.
- Maximum TX bandwidth is 1.5 Gbps per lane.
- Non-continuous clock mode on RX channels is possible as long as the continuous clock can be obtained internally or fed directly from the pin.
- External reference clock may be required in case a proper clock is not derived from RX side to drive TX D-PHY PLL.

## 1.3. Block Diagram

Figure 1.1 shows the block level diagram of the 1 to N MIPI CSI-2/DSI Duplicator reference design with two TX channels.

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

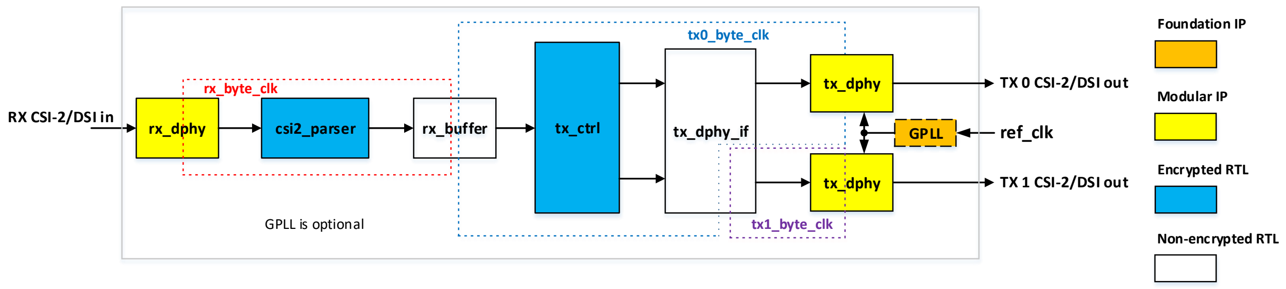


**Figure 1.1. 1 to N MIPI CSI-2/DSI Duplicator Block Diagram**

## 1.4. RX/TX Permutations

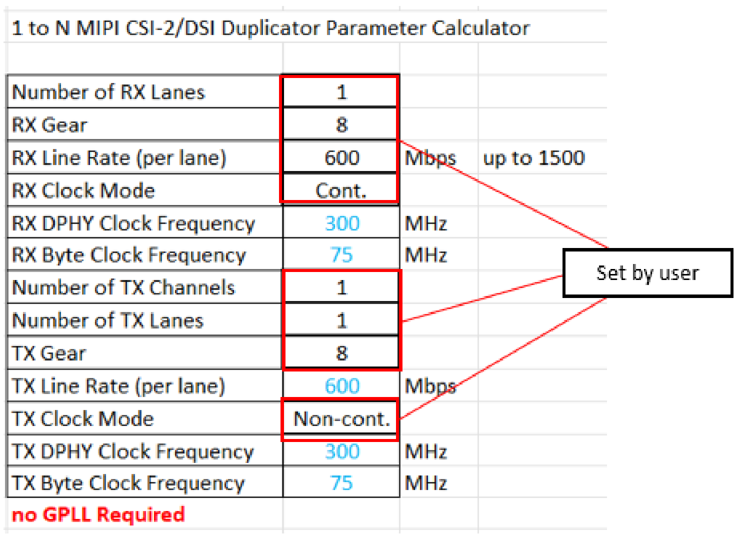
Table 1.2 shows the possible RX/TX permutations and Figure 1.2 shows the calculated bandwidth in the Excel sheet provided with this reference design. Some unrealistic permutations are excluded due to bandwidth limitations. When the lane bandwidth is around 900 Mbps, both Gear 8 and Gear 16 may be used. If Gear 8 cannot meet timing (see static timing analysis (STA) report), Gear 16 should be considered. The maximum RX lane bandwidth depends on the type of D-PHY IP, Hard D-PHY or Soft D-PHY. In case of two-channel outputs, no Hard D-PHY IP is available on RX and maximum RX lane bandwidth is always 1200 Mbps. The permutations shown in Table 1.2 is based on the assumption that total bandwidth (lane bandwidth x number of lane) is equal between RX and TX. In some case, RX and TX total bandwidth could be different and the values shown in the table and figure might not directly apply. These cases are described in rx_buffer section.

**Table 1.2. RX/TX Permutations**

| Number of RX Lane | RX Gear[2] | RX Lane Bandwidth[1] x (Mbps) | Number of TX Lane | TX Gear[2] | TX Lane Bandwidth y (Mbps) | TX Byte Clock/ RX Byte Clock |
|---|---|---|---|---|---|---|
| 1 | 16 | ~900 ≤ x ≤ 1200/1500 | 1 | 16 | ~900 ≤ y ≤ 1200/1500 | 1 |
| | | | 2 | 8 | ~450 ≤ y ≤ 600/750 | 1 |
| | | | 4 | 8 | ~225 ≤ y ≤ 300/375 | 0.5 |
| | 8 | 160 ≤ x ≤ ~900 | 1 | 8 | 160 ≤ y ≤ ~900 | 1 |
| | | 320 ≤ x ≤ ~900 | 2 | | 160 ≤ y ≤ ~450 | 0.5 |
| | | 640 ≤ x ≤ ~900 | 4 | | 160 ≤ y ≤ ~225 | 0.25 |
| 2 | 16 | ~900 ≤ x ≤ 1500/1200 | 1 | 16 | y ≥ ~1800 | NA |
| | | ~900 ≤ x ≤ 1500/1200 | 2 | 8 | ~450 ≤ y ≤ 600/750 | 1 |
| | | ~900 ≤ x ≤ 1500/1200 | 4 | 8 | ~225 ≤ y ≤ 300/375 | 0.5 |
| | 8 | ~450 ≤ x ≤ 750 | 1 | 16 | ~900 ≤ y ≤ 1500 | 1 |
| | | 160 ≤ x ≤ ~450 | | 8 | 320 ≤ y ≤ ~900 | 2 |
| | | 320 ≤ x ≤ ~900 | 2 | 8 | 160 ≤ y ≤ ~450 | 0.5 |
| | | 640 ≤ x ≤ ~900 | 4 | 8 | 160 ≤ y ≤ ~225 | 0.25 |
| 4 | 16 | ~900 ≤ x ≤ 1200/1500 | 1 | 16 | y ≥ ~3600 | NA |
| | | ~900 ≤ x ≤ 1200/1500 | 2 | 16 | y ≥ ~1800 | NA |
| | | ~900 ≤ x ≤ 1200/1500 | 4 | 16 | ~900 ≤ y ≤ 1200/1500 | 1 |
| | 8 | ~225 ≤ x ≤ 375 | 1 | 16 | ~900 ≤ y ≤ 1500 | 2 |
| | | 160 ≤ x ≤ ~225 | | 8 | 640 ≤ y ≤ ~900 | 4 |
| | | ~450 ≤ x ≤ 750 | 2 | 16 | ~900 ≤ y ≤ 1500 | 1 |
| | | 160 ≤ x ≤ ~450 | | 8 | 320 ≤ y ≤ ~900 | 2 |
| | | 160 ≤ x ≤ ~900 | 4 | 8 | 160 ≤ y ≤ ~900 | 1 |

**Notes:**

1. The maximum RX lane bandwidth depends on the type of D-PHY RX IP (1500 Mbps: Hard D-PHY, 1200 Mbps: Soft D-PHY).
2. The bandwidth border guideline of Gear selection between 8 and 16 is 900 Mbps. When the lane bandwidth is around 900 Mbps, user can select either one based on the result of the actual timing report (STA).



1 to N MIPI CSI-2/DSI Duplicator Parameter Calculator

| | | | |
|---|---|---|---|
| Number of RX Lanes | 1 | | |
| RX Gear | 8 | | |
| RX Line Rate (per lane) | 600 | Mbps | up to 1500 |
| RX Clock Mode | Cont. | | |
| RX DPHY Clock Frequency | 300 | MHz | |
| RX Byte Clock Frequency | 75 | MHz | |
| Number of TX Channels | 1 | | |
| Number of TX Lanes | 1 | | |
| TX Gear | 8 | | |
| TX Line Rate (per lane) | 600 | Mbps | |
| TX Clock Mode | Non-cont. | | |
| TX DPHY Clock Frequency | 300 | MHz | |
| TX Byte Clock Frequency | 75 | MHz | |
| no GPLL Required | | | |

**Figure 1.2. Bandwidth and Parameter Calculator**

The excel file for the parameter calculator is located in the design package source code. Open the source code folder, locate the **docs** folder, and locate the **mipi_dup_RD.xlsx**.

# 2. Parameters and Port List

There are two directive files for this reference design:

- synthesis_directives.v – used for design compilation by Lattice Diamond® and for simulation.
- simulation_directives.v – used for simulation.

The user can modify these directives according to the own configuration. The settings in these files must match RX D-PHY IP, TX D-PHY IP, and other module settings described in Design and Module Description section.

## 2.1. Synthesis Directives

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

**Table 2.1. Synthesis Directives**

| Category | Directive | Remarks |
|---|---|---|
| External reference clock | EXT_REF_CLK | Enable this when the reference clock is fed from a pin. Minimum 10 MHz. |
| RX D-PHY Type | CSI2 | Only one of these two directives must be defined. |
| | DSI | |
| RX channel lane count | NUM_RX_LANE_1 | Number of lanes in each RX channel. Only one of these three directives must be defined. |
| | NUM_RX_LANE_2 | |
| | NUM_RX_LANE_4 | |
| RX D-PHY Clock Gear | RX_GEAR_8 | Only one of these directives must be selected. Gear 16 can be used for only 1 and 2 lane configurations. |
| | RX_GEAR_16 | |
| RX Hard D-PHY | RX_DPHY_HARD | Enable Hard D-PHY on RX channel. When not defined, RX channel uses Soft D-PHY. This can be defined only for single TX channel output. |
| RX D-PHY Clock Mode[1, 2] | RX_CLK_MODE_HS_ONLY | RX D-PHY Clock mode. Only one of these two directives must be defined. |
| | RX_CLK_MODE_HS_LP | |
| Wait for FS/VSS | WAIT_FS | Wait for FS (Frame Start) or VSS (Vsync Start) to begin capturing RX data. Capturing begins with any data if not defined. |
| RX Buffer Depth | RX_BUFFER_DEPTH_512 | Depth of RX Buffer FIFO. Only one of these four directives must be defined. 4096 cannot be defined in case of (NUM_RX_LANE_4 and RX_GEAR_16) due to the limitation of available EBR. |
| | RX_BUFFER_DEPTH_1024 | |
| | RX_BUFFER_DEPTH_2048 | |
| | RX_BUFFER_DEPTH_4096 | |
| Byte Data Read Delay | BD_RD_DLY {value} | Byte Data Read Delay from RX Buffer in rx_byte_clk cycles. The value must be 1 – 8191. |
| Use GPLL | USE_GPLL | Use GPLL to create a reference clock to be fed to PLL of TX D-PHY IP. |
| TX Channel count | NUM_TX_CH_1 | Number of TX channel. Only one of these two directives must be defined. |
| | NUM_TX_CH_2 | |
| TX channel lane count | NUM_TX_LANE_1 | Number of lanes in TX channel. Only one of these three directives must be defined. |
| | NUM_TX_LANE_2 | |
| | NUM_TX_LANE_4 | |
| TX D-PHY Clock Gear | TX_GEAR_8 | TX D-PHY Clock Gear. Only one of these two directives must be defined. |
| | TX_GEAR_16 | |
| TX D-PHY Clock Mode[1] | TX_CLK_MODE_HS_ONLY | TX D-PHY Clock mode. Only one of these two directives must be defined. |
| | TX_CLK_MODE_HS_LP | |
| Keep HS mode | KEEP_HS | Keep the clock lane in HS mode during the horizontal blanking periods of active video lines when defined. Effective when CSI2 and TX_CLK_MODE_HS_LP are defined. |

**Notes**:

1. HS_LP mode means *non-continuous clock mode* and HS_ONLY means *continuous clock mode*. HS_LP mode works only if RX byte clock for corresponding RX channel can be generated internally or directly fed from I/O pin.
2. 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 |
|---|---|---|
| simulation | SIM | Select behavioral models for simulation. |
| Reference clock period | REF_CLK_PERIOD {value} | Reference clock period in ps |
| RX D-PHY clock period | RX_DPHY_CLK_PERIOD {value} | RX Channel DPHY clock period in ps |
| Initial delay on RX channel | RX_DELAY {value} | Initial delay to activate RX Channel in ps |
| Gap (LP) time between active lines on RX Channel | RX_DPHY_LPS_GAP {value} | Gap time on RX Channels in ps |
| Gap (LP) time between Frame End and Frame Start on RX Channel | RX_DPHY_FRAME_GAP {value} | Gap time on RX Channels in ps |
| Video data configuration on RX Channel | NUM_FRAMES {value} | Number of frames to feed |
| | NUM_LINES {value} | Number of active lines per frame |
| | NUM_PIXELS {value} | Number of pixels per active line |
| | RAW8 | Data Type of the payload video data. Only one of these nine directives must be defined. |
| | RAW10 | |
| | RAW12 | |
| | RAW14 | |
| | RGB888 | |
| | RGB666 | |
| | RGB565 | |
| | YUV422_8 | |
| | YUV422_10 | |
| DSI Sync Mode | DSI_SYNC_MODE_EVENT | Select non-burst with sync event or no-burst with sync pulse mode. Applicable when DSI is defined in synthesis_directives.v. |
| | DSI_SYNC_MODE_PULSE | |
| EoTp insertion | EOTP | Insert EoTp (End of Transmission Packet) when defined. Applicable when DSI is defined in synthesis_directives.v. |
| VSA length | VSA_LENGTH {value} | Number of Vsync active lines. Applicable when DSI is defined in synthesis_directives.v. |
| VBP length | VBP_LENGTH {value} | Number of Vertical Back Porch lines. Applicable when DSI is defined in synthesis_directives.v. |
| VFP length | VFP_LENGTH {value} | Number of Vertical Front Porch lines. Applicable when DSI is defined in synthesis_directives.v. |
| HSA length | HSA_LENGTH {value} | Number of WC in Null Packet of Hsync active period. Applicable when DSI is defined in synthesis_directives.v. |
| HBP length | HBP_LENGTH {value} | Number of WC in Blanking Packet of Horizontal Back Porch. Applicable when DSI is defined in synthesis_directives.v. |
| VSA length | VSA_LENGTH {value} | Number of Vsync active lines. Applicable when DSI is defined in synthesis_directives.v. |
| Initialization Time monitor | MISC_TINITDONE | Enables internal signal tx0_tinit_done monitored by the testbench. Enable this directive when "Bypass tINIT counter" is unchecked in TX D-PHY settings in Clarity for tx_dphy. |

## 2.3. Top-Level I/O

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

**Table 2.3. CSI-2/DSI Duplicator Top-Level I/O**

| Port Name | Direction | Description |
|---|---|---|
| **Clocks and Resets** | | |
| ref_clk_i (optional) | I | Input reference clock. Used to feed a clock to TX D-PHY PLL directly or indirectly. This port is declared only when EXT_REF_CLK is defined in synthesis_directives.v. |
| reset_n_i | I | Asynchronous active low system reset |
| **CSI-2/DSI RX Interface** | | |
| rx_clk_p_i | I | Positive differential RX D-PHY input clock |
| rx_clk_n_i | I | Negative differential RX D-PHY input clock |
| rx_d0_p_i | I | Positive differential RX D-PHY input data 0 |
| rx_d0_n_i | I | Negative differential RX D-PHY input data 0 |
| rx_d1_p_i | I | Positive differential RX D-PHY input data 1 (in case of 2-lane or 4-lane configuration) |
| rx_d1_n_i | I | Negative differential RX D-PHY input data 1 (in case of 2-lane or 4-lane configuration) |
| rx_d2_p_i | I | Positive differential RX D-PHY input data 2 (in case of 4-lane configuration) |
| rx_d2_n_i | I | Negative differential RX D-PHY input data 2 (in case of 4-lane configuration) |
| rx_d3_p_i | I | Positive differential RX D-PHY input data 3 (in case of 4-lane configuration) |
| rx_d3_n_i | I | Negative differential RX D-PHY input data 3 (in case of 4-lane configuration) |
| **CSI-2/DSI TX Interface** | | |
| tx0_clk_p_o | O | Positive differential TX D-PHY output clock on TX channel 0 |
| tx0_clk_n_o | O | Negative differential TX D-PHY output clock on TX channel 0 |
| tx0_d0_p_o | O | Positive differential TX D-PHY output data 0 on TX channel 0 |
| tx0_d0_n_o | O | Negative differential TX D-PHY output data 0 on TX channel 0 |
| tx0_d1_p_o | O | Positive differential TX D-PHY output data 1 on TX channel 0 (in case of 2/4-lane configuration) |
| tx0_d1_n_o | O | Negative differential TX D-PHY output data 1 on TX channel 0 (in case of 2/4-lane configuration) |
| tx0_d2_p_o | O | Positive differential TX D-PHY output data 2 on TX channel 0 (in case of 4-lane configuration) |
| tx0_d2_n_o | O | Negative differential TX D-PHY output data 2 on TX channel 0 (in case of 4-lane configuration) |
| tx0_d3_p_o | O | Positive differential TX D-PHY output data 3 on TX channel 0 (in case of 4-lane configuration) |
| tx0_d3_n_o | O | Negative differential TX D-PHY output data 3 on TX channel 0 (in case of 4-lane configuration) |
| tx1_clk_p_o | O | Positive differential TX D-PHY output clock on TX channel 1 (in case of 2 TX channel configuration) |
| tx1_clk_n_o | O | Negative differential TX D-PHY output clock on TX channel 1 (in case of 2 TX channel configuration) |
| tx1_d0_p_o | O | Positive differential TX D-PHY output data 0 on TX channel 1 (in case of 2 TX channel configuration) |
| tx1_d0_n_o | O | Negative differential TX D-PHY output data 0 on TX channel 1 (in case of 2 TX channel configuration) |
| tx1_d1_p_o | O | Positive differential TX D-PHY output data 1 on TX channel 1 (in case of 2 TX channel with 2/4-lane configuration) |
| tx1_d1_n_o | O | Negative differential TX D-PHY output data 1 on TX channel 1 (in case of 2 TX channel with 2/4-lane configuration) |
| tx1_d2_p_o | O | Positive differential TX D-PHY output data 2 on TX channel 1 (in case of 2 TX channel with 4-lane configuration) |
| tx1_d2_n_o | O | Negative differential TX D-PHY output data 2 on TX channel 1 (in case of 2 TX channel with 4-lane configuration) |
| tx1_d3_p_o | O | Positive differential TX D-PHY output data 3 on TX channel 1 (in case of 2 TX channel with 4-lane configuration) |
| tx1_d3_n_o | O | Negative differential TX D-PHY output data 3 on TX channel 1 (in case of 2 TX channel with 4-lane configuration) |

# 3. Design and Module Description

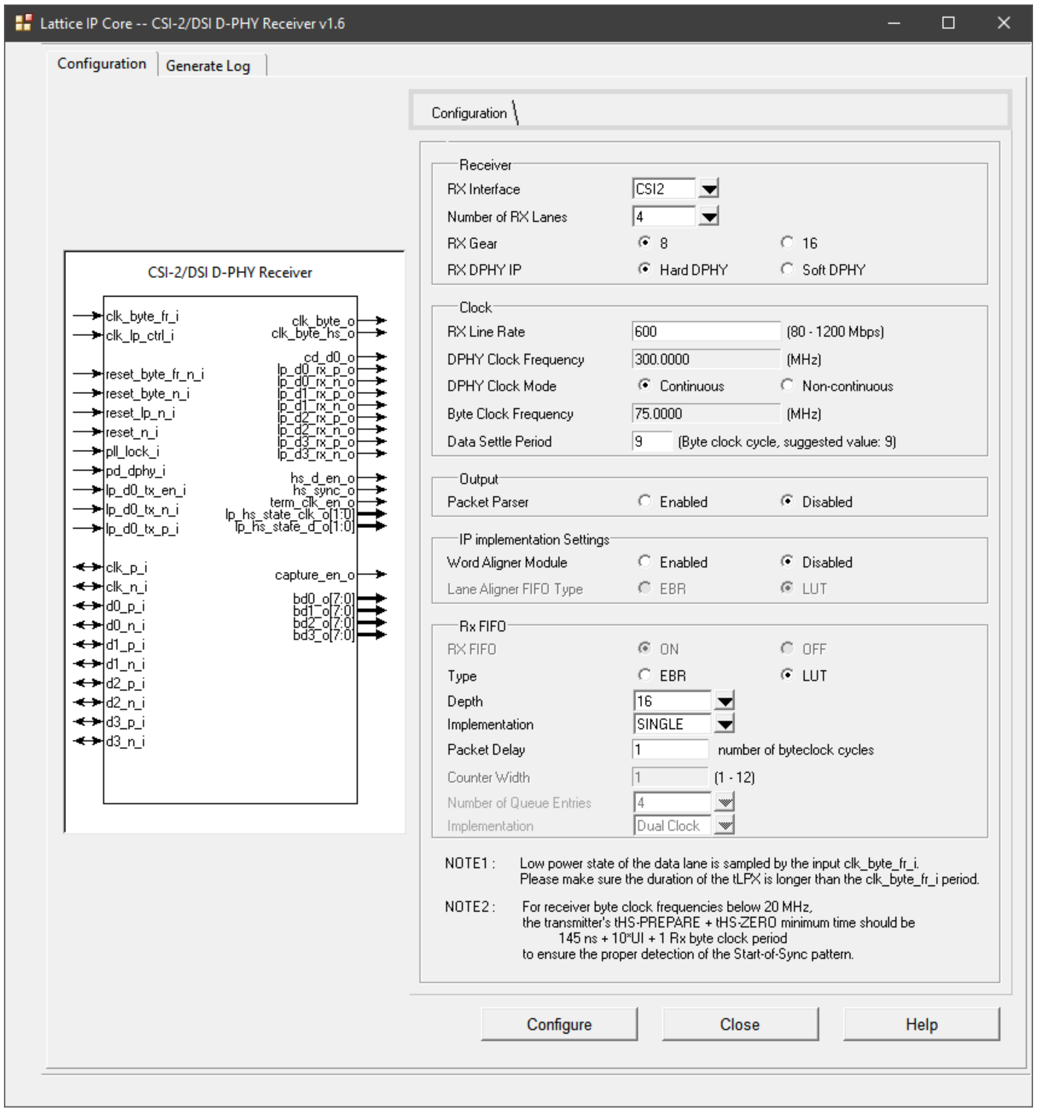
The top-level design (mipi_duplicator.v) consists of the following modules:

- rx_dphy
- mipi_parser
- rx_buffer
- tx_ctrl
- tx_dphy_if
- tx_dphy

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

## 3.1. rx_dphy

This module must be created for RX channel according to channel conditions, such as the number of lanes, bandwidth, and others. Figure 3.1 shows an example of IP interface settings in Clarity for the CSI-2/DSI D-PHY Receiver Submodule IP. Refer to CSI-2/DSI D-PHY Receiver Submodule IP User Guide (FPGA-IPUG-02025) for details.



Receiver

| | | |
|---|---|---|
| RX Interface | CSI2 | |
| Number of RX Lanes | 4 | |
| RX Gear | [✓ 8] | 16 |
| RX DPHY IP | [✓ Hard DPHY] | Soft DPHY |

Clock

| | | |
|---|---|---|
| RX Line Rate | 600 | (80 - 1200 Mbps) |
| DPHY Clock Frequency | 300.0000 | (MHz) |
| DPHY Clock Mode | [✓ Continuous] | Non-continuous |
| Byte Clock Frequency | 75.0000 | (MHz) |
| Data Settle Period | 9 | (Byte clock cycle, suggested value: 9) |

Output

| | | |
|---|---|---|
| Packet Parser | Enabled | [✓ Disabled] |

IP implementation Settings

| | | |
|---|---|---|
| Word Aligner Module | Enabled | [✓ Disabled] |
| Lane Aligner FIFO Type | EBR | [✓ LUT] |

Rx FIFO

| | | |
|---|---|---|
| RX FIFO | [✓ ON] | OFF |
| Type | EBR | [✓ LUT] |
| Depth | 16 | |
| Implementation | SINGLE | |
| Packet Delay | 1 | number of byteclock cycles |
| Counter Width | 1 | (1 - 12) |
| Number of Queue Entries | 4 | |
| Implementation | Dual Clock | |

NOTE1 : Low power state of the data lane is sampled by the input clk_byte_fr_i.
Please make sure the duration of the tLPX is longer than the clk_byte_fr_i period.

NOTE2 : For receiver byte clock frequencies below 20 MHz,
the transmitter's tHS-PREPARE + tHS-ZERO minimum time should be
145 ns + 10*UI + 1 Rx byte clock period
to ensure the proper detection of the Start-of-Sync pattern.

**Figure 3.1. rx_dphy IP Creation in Clarity Designer**

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

- RX Interface – Select CSI-2 or DSI.
- Number of RX Lanes – Set according to channel configuration. The value must match NUM_RX_LANE_* setting.
- RX Gear – Select 8 or 16; 16 is supported for 1-lane and 2-lane configurations only. 16 is recommended when the RX byte clock speed exceeds 100 MHz with Gear 8 and STA fails.
- RX DPHY IP – Hard D-PHY can be selected only for one TX channel configuration. The setting must match RX_D-PHY_HARD setting.
- RX Line Rate – Set according to channel configuration.
- DPHY Clock Mode – Select Continuous or Non-continuous. Must match RX_CLK_MODE_* setting (Continuous = HS_ONLY, Non-continuous = HS_LP).
- Data Settle Period – Recommended to use the suggested value. The user can try reducing this value one by one when RX D-PHY is not working as expected.
- Packet Parser – Select Disabled.
- Word Aligner Module – Select Enabled in case of Hard D-PHY in high-temperature condition (ambient temperature above 125 °C). Automatically enabled for Soft D-PHY.
- Lane Aligner Module – LUT or EBR for 2 and 4 lane configuration. LUT is recommended.
- This module takes serial CSI-2/DSI data and outputs byte data after de-serialization in CSI-2/DSI High Speed mode. It is recommended to set the design name to *rx* and module names to *rx_dphy* so that the user do not need to modify the instance name of this IP in the top-level design as well as the simulation setup file. Otherwise, user have to modify the name 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 after the word aligner in case of Hard D-PHY RX IP and resides before the word aligner in case of Soft D-PHY RX IP.

#### 3.1.1.1. Hard D-PHY in Continuous Clock Mode

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

#### 3.1.1.2. Soft D-PHY in Continuous Clock Mode

In this case, RX FIFO is not necessary and RX_FIFO should be set to OFF.

#### 3.1.1.3. Non-Continuous Clock Mode

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

- Continuous byte clock = non-continuous byte clock

  In this case, the minimum configuration of RX FIFO is recommended (LUT based, Depth = 16, *Type* Implementation = SINGLE, Packet Delay = 1, *Clock* Implementation = DC).
- Continuous byte clock < non-continuous byte clock

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

- Continuous byte clock > non-continuous byte clock

  There are two options in this case:

  - Use *Type* Implementation = SINGLE with large Packet Delay

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

  - Use *Type* Implementation = QUEUE with Number of Queue Entries = 2

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

- Frequency relationship is unknown

  When the continuous byte clock is within the certain range against the non-continuous byte clock (for example, two clocks come from different clock sources which have ppm tolerance), we have no idea which clock is faster. The simplest way is to use *Type* Implementation = SINGLE with setting Packet Delay to the midpoint of FIFO depth when the tolerance is in ppm level. For example, assuming the clock tolerance is within +/- 500ppm for 2 lane Gear 8 RAW10 with 1920 horizontal pixels. The payload byte count is 1920 x 5/4 = 2400 so that each lane takes 1200 bytes. 1200 x 500 ppm = 0.6, which means small FIFO with middle point read delay (ex. FIFO depth = 16 with read delay of 8-byte clock cycles) works fine. Using LUT is preferable as long as FIFO depth is small since using EBR here might cause the shortage of EBR in line_buf modules. QUEUE can also be used as described above even though it requires more EBRs.

  In case user do not have detailed information regarding RX data (whether containing line start/end short packet, interval of the horizontal blanking period against active line period), the safest way is to set the continuous byte clock faster than the non-continuous byte clock. Use *Type* Implementation = QUEUE with Number of entries = 4 even though it might require more EBR resources comparing to *Type* Implementation = SINGLE, but ensure that total number of EBR used in the device do not exceed 20, the maximum number of EBRs available in CrossLink.

## 3.2. mipi_parser

This module handles CSI-2/DSI protocol decoding. It detects the sync word (0xB8) and trailer bytes (0x00 or 0xFF) to properly assert rx_bd_en, which indicates active packet data. Figure 3.2 shows an example of handling a CSI-2 Frame Start short packet. *B8* and *FF* are trimmed from the incoming data by the rx_bd_en assertion (Num_RX_LANE_4, RX_GEAR_8). rx_bd_end indicates the end of the valid packet data.

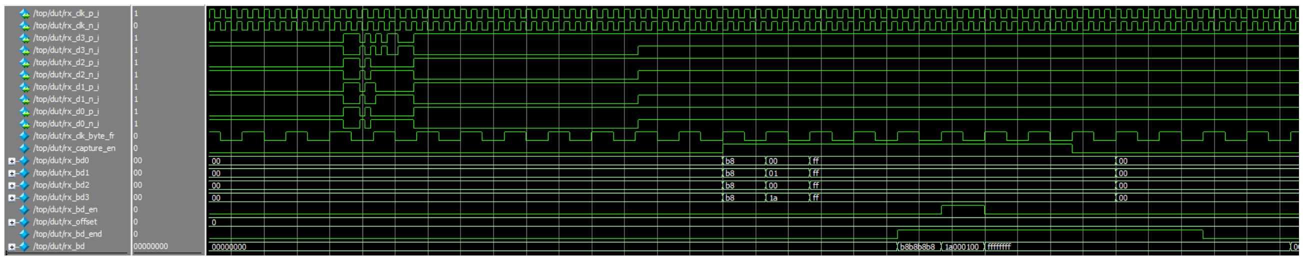


**Figure 3.2. Active Data Detection (CSI-2 Frame Start)**

Figure 3.3 shows the end of CSI-2 long packet detection. rx_bd_end is asserted in the last cycle of rx_bd_en. In this case, rx_offset = 0 which means only LSB data (0x1a76) is active in the last rx_bd data indicated by rx_bd_en = 1 and rx_bd_end = 1. MSB 1 byte (0xff) is a trailer byte.

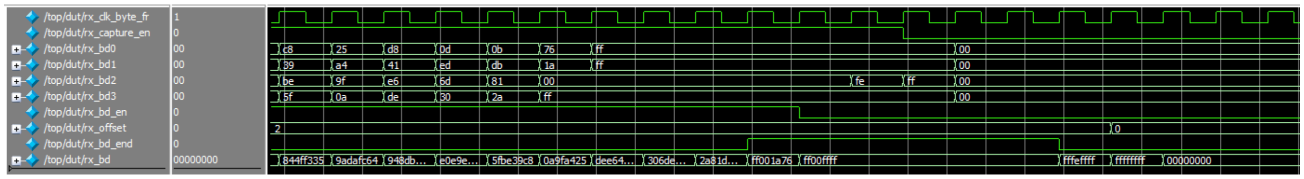


**Figure 3.3. End of Active Data Detection (CSI-2 Long Packet)**

## 3.3. rx_buffer

This module contains a dual clock FIFO using EBR to store valid packet data. Buffer depth is 512, 1024, 2048, or 4096 and data width is either 64 (when NUM_RX_LANE = 4 and RX Gear = 16) or 32 (others). The first FIFO data is read out by this module itself with ready flag assertion. The FIFO has extra data width to contain data end flag and offset data. Figure 3.4 shows the example of short packet write and read (NUM_RX_LANE = 4, FIFO data width is 32). rx_bd_en makes the FIFO write after the data width conversion from 16 bits to 32 bits and it also triggers the internal counter driven by rx_byte_clk. When this counter reaches the value specified by BD_RD_DLY, the internal flag is asserted and that status is transferred to tx0_byte_clk domain, which makes buf_rdy = 1 to notify tx_ctrl module that the data are ready to be read. On the other hand, the 1st FIFO data is automatically read out by this module itself so that the 1st data is already available on buf_d when buf_rdy is asserted. In case of a short packet, the amount of active data is only 4 bytes and buf_d_end is also asserted along with buf_rdy and short packet data of 0x1A000100 on buf_d. In this example BD_RD_DLY = 10 and buf_rdy is asserted soon after rx_bd_en = 1, but the data read (buf_re = 1) from the next module (tx_ctrl) does not begin due to LP to HS transition time on TX channel. This delay is expected to ~1 µs – 1.5 µs in case that both clock and data lanes requires LP to HS transition.

In general, the small value of BD_RD_DLY is preferable to make a process delay shorter, but it makes sense to set a larger value when there exists some frequency difference or tolerance between rx_byte_clk and tx0_byte_clk. If the clock ratio between these two is not exactly same as shown in Table 1.2, this FIFO could be used to absorb the frequency difference. Concept-wise, it is similar to SINGLE mode of RX FIFO in rx_dphy. When the tx0_byte_clk is faster than rx_byte_clk, setting BD_RD_DLY to a higher value (close to the rx_byte_clk cycle count to complete the payload data write for long packet in active video lines) helps to avoid FIFO underflow as long as the clock tolerance is absorbed during the LP period. On the other hand, setting a small value helps to avoid FIFO overflow when tx0_byte_clk is slower than rx_byte_clk. If no idea which is faster but tolerance is expected, then setting close to the middle value makes sense. The minimum buffer depth is 512 and that requires 2 or 4 EBRs depends on data width. This depth can be expanded to 1024, 2048, or 4096 as long as the total number of EBR does not exceed 20.

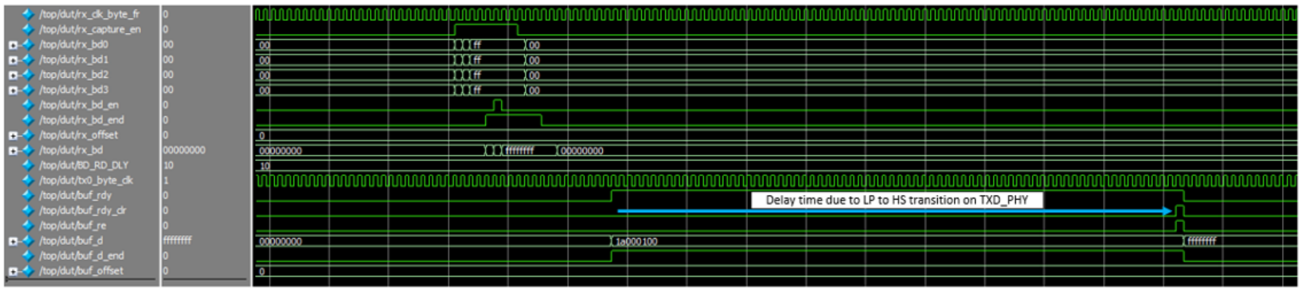


**Figure 3.4. Short Packet Write and Read**

Figure 3.5 and Figure 3.6 show the difference by BD_RD_DLY in the same design. In this case, WC (payload byte count) is 2400, which means it takes 1200 rx_byte_clk cycles to write all long packet payload data to FIFO. tx0_byte_clk exactly matches the derived clock ratio by Table 1.2. In case of BD_RD_DLY = 400, FIFO read (buf_re = 1) begins while FIFO has some room to be filled up. In case of BD_RD_DLY = 1000, data is corrupted due to FIFO overflow.

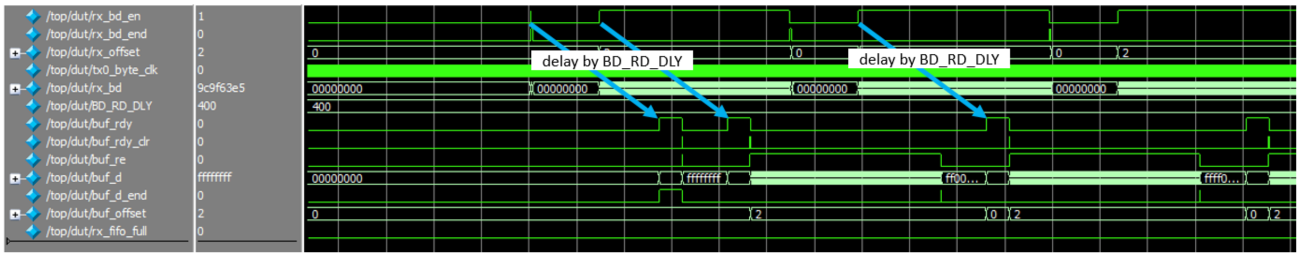


**Figure 3.5. rx_buffer Write/Read Example with Depth = 512, BD_RD_DLY = 400**

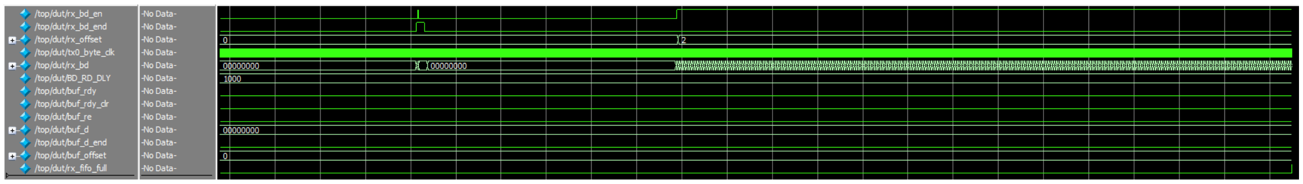


**Figure 3.6. rx_buffer FIFO Overflow with Depth = 512, BD_RD_DLY = 1000**

The safest way to avoid FIFO overflow in case that total TX bandwidth > RX total bandwidth (byte clock ratio value is larger than the ratio shown in Table 1.2) is to set the FIFO depth larger than the size of long packet data. Figure 3.7 shows the successful transaction setting the FIFO depth to 1024 with BD_RD_DLY = 1200. buf_rdy is asserted almost at the same timing of the completion of the one-line video data write to the FIFO. The FIFO depth of 1024 is deep enough to store the whole one-line data (1024 x 4 (bytes) > (4 (Header) + 2400 (payload) + 2 (CRC))) and FIFO overflow does not occur.

Figure 3.8 shows the example of successful transactions of (TX total bandwidth > RX total bandwidth). In this case, TX total bandwidth is 2x of RX. The FIFO read period (buf_re = 1) is half of FIFO write period (rx_bd_en = 1), which leads more LP mode time on TX channel. This method (setting BD_RD_DLY close to the end of payload data write, setting FIFO depth to cover the long packet data size) works for any TX bandwidth as long as it is faster than RX bandwidth and does not exceed the maximum lane bandwidth of 1.5 Gbps in case of CSI-2, which mandates the data lane goes into LP mode after every packet transaction.

In case of DSI which keeps the data lane in HS mode during the horizontal blanking periods using null or blanking packets, this method cannot apply and TX bandwidth has to be exactly same (or close enough) to avoid FIFO overflow/underflow.

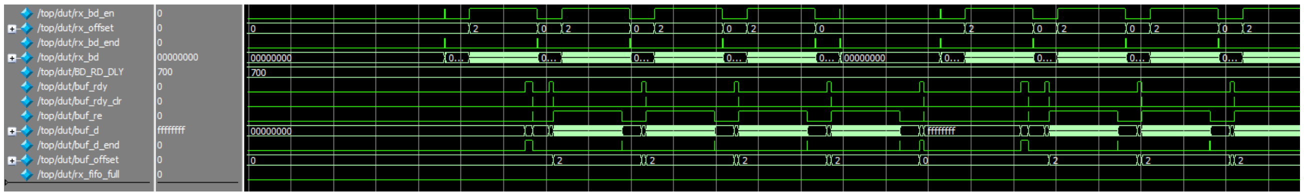


**Figure 3.7. rx_buffer Successful Transactions with Depth = 1024, BD_RD_DLY = 700**

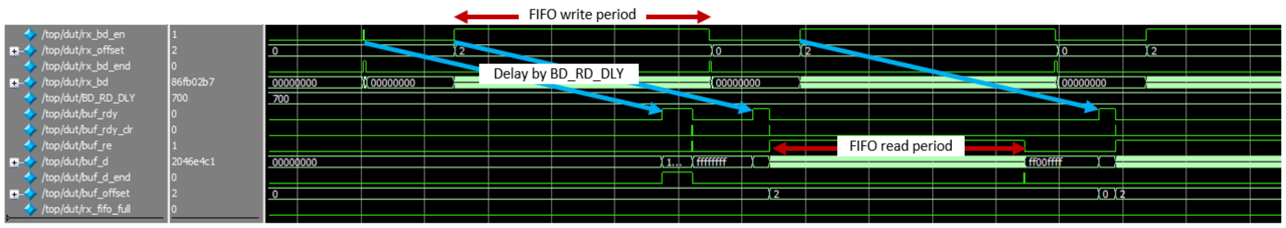


**Figure 3.8. TX bandwidth = 2 x RX Bandwidth with Depth = 1024, BD_RD_DLY = 700**

## 3.4. tx_ctrl

This module monitors the read ready flag of rx_buffer and then reads RX Buffer data. The read data are sent to tx_dphy through tx_dphy_if. Figure 3.9 shows an example of a short packet transaction. After receiving tx_bgn from tx_dphy_if, this module begins capturing buf_d and sends these on tx_bd with proceeding *B8* (sync word) and appending *00* or *FF* (trailer byte).

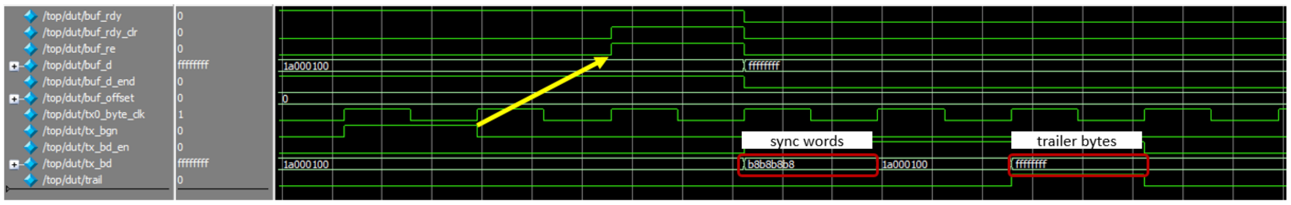


**Figure 3.9. Short Packet Transaction (NUM_TX_LANE_4, TX_GEAR_8)**

sync words trailer bytes

Figure 3.10 shows an example of the end of long packet transaction. In this case, the lane configuration between RX and TX is different so that trailer bytes are recreated and appended by this module.

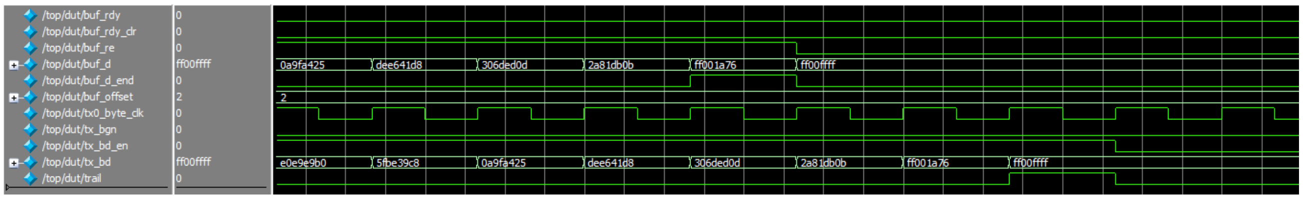


**Figure 3.10. End of Long Packet Transaction (NUM_RX_LANE_4, RX_GEAR_8 to NUM_TX_LANE_4, TX_GEAR_8)**

When TX D-PHY is set to non-continuous clock mode, it is possible to keep the clock lane in HS mode during the horizontal blanking periods. This is enabled when KEEP_HS is defined in synthesis_directives.v. This feature is useful when the horizontal blanking period is not long enough to have both clock and data lanes go into LP mode, which requires more overhead time. In that case, this module keeps clk_hs_en = 1 during the active video period including the horizontal blanking periods and makes clk_hs_en = 0 only during the vertical blanking period so that the clock lane goes into LP mode only during the vertical blanking period. This feature is available only for CSI-2.

## 3.5. tx_dphy_if

This module resides between tx_ctrl and tx_dphy to transfer control signals and data including data bus allocation and clock domain conversion in case of two channel TX outputs. Figure 3.11 shows an example of short packet transaction in non-continuous clock mode (only data lane 0 is shown in the figure).

1. Wait for buf_rdy = 1
2. Check tx0_c2d_rdy = 1, then assert tx0_clk_hs_en and tx0_d_hs_en (at least one tx0_byte_clk cycle)
3. Clock lane goes into HS mode
4. Data lane goes into HS mode
5. Wait for tx0_d_hs_rdy = 1
6. Assert tx_bgn
7. FIFO read happens by tx_ctrl
8. Receive HS data by tx_bd_en
9. Send HS data to tx_dphy along with tx0_dphy_pkten = 1
10. HS data transmission by tx_dphy
11. After HS transmission is done, tx0_d_hs_rdy goes 0 and data lane goes into LP mode
12. Clock lane goes into LP mode
13. After all HS transaction ends, tx0_c2d_rdy becomes 1, which means tx_dphy is ready to handle the next HS transaction.

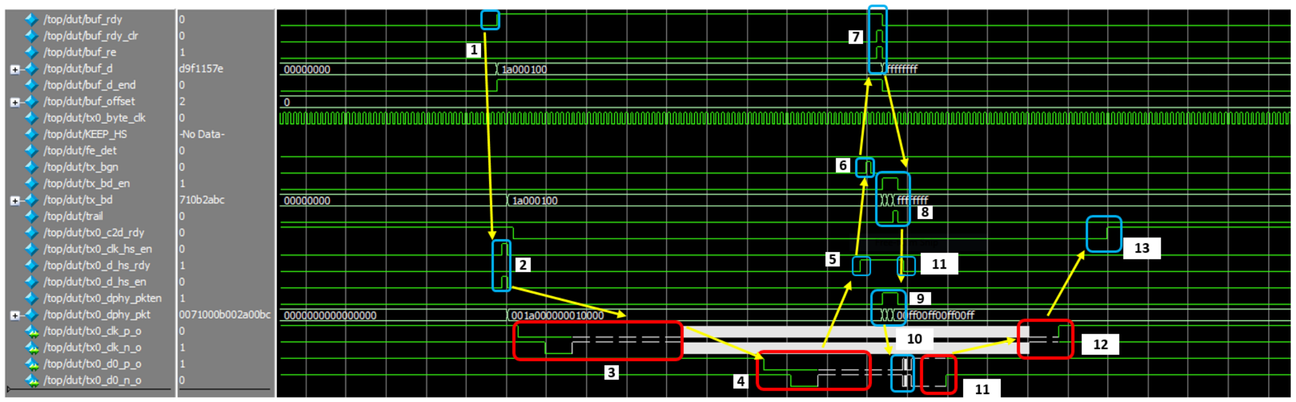


Figure 3.11. Short Packet Transaction in Non-Continuous Clock Mode

In case of CSI-2 in non-continuous clock mode, it is possible to keep clock lane in HS mode during the horizontal blanking periods by defining KEEP_HS in synthesis_directives.v. Figure 3.12 shows the global sequence without defining KEEP_HS and Figure 3.13 shows the case with KEEP_HS. When KEEP_HS is defined, the control logic for tx0_clk_hs_en takes fe_det (Frame End detect) and keeps tx0_clk_hs_en = 1 during the horizontal blanking periods.

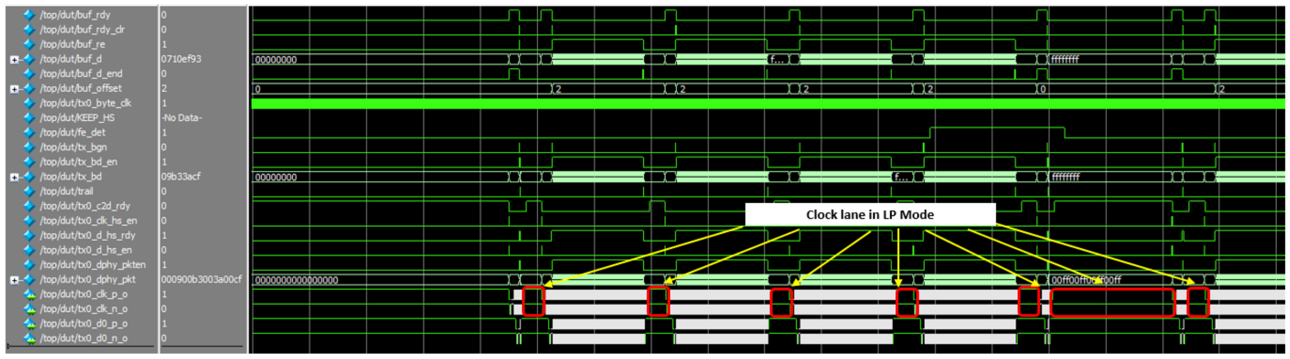


Figure 3.12. Global Sequence in Non-Continuous Clock Mode without KEEP_HS

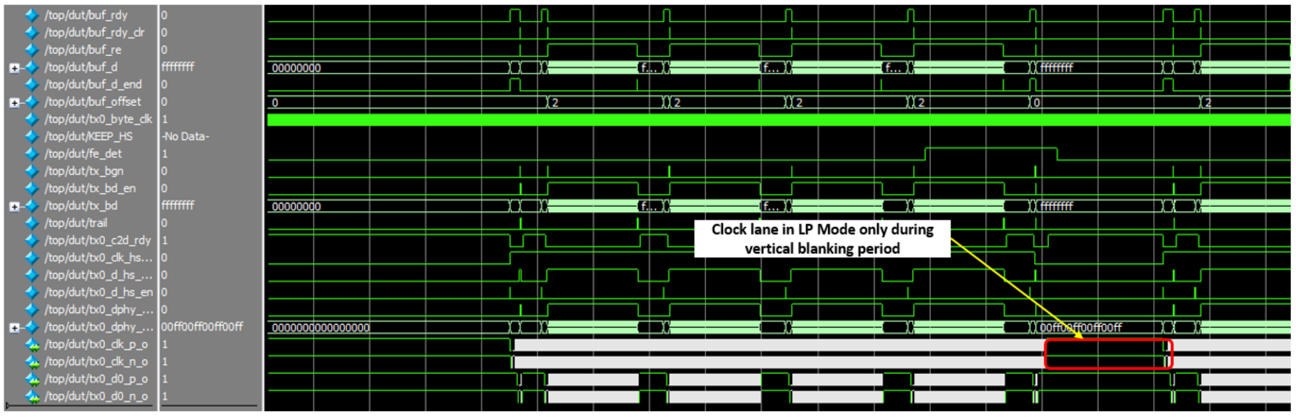
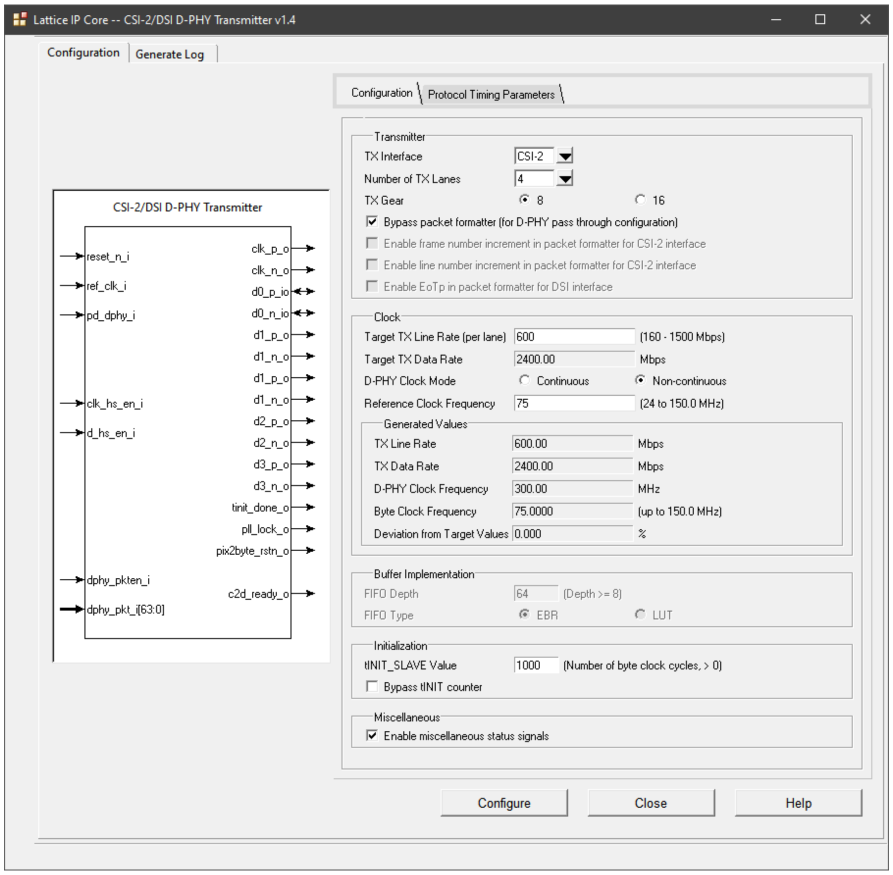


Figure 3.13. Global Sequence in Non-Continuous Clock Mode with KEEP_HS

This module takes care of two TX channels by communicating with two tx_dphy modules. Operations are almost similar to one TX channel output except for clock domain conversions, which lead to a few cycles delay in transactions.

## 3.6. tx_dphy

The user must create this module according to channel conditions, such as number of lanes, bandwidth, and others. Figure 3.14 shows an example of IP interface setting in Clarity Designer for the CSI-2/DSI D-PHY Transmitter Submodule IP. Refer to CSI-2/DSI D-PHY Transmitter Submodule IP User Guide (FPGA-IPUG-02024) for details.



Transmitter
TX Interface: CSI-2
Number of TX Lanes: 4
TX Gear: [✓ 8] 16
[✓] Bypass packet formatter (for D-PHY pass through configuration)
[ ] Enable frame number increment in packet formatter for CSI-2 interface
[ ] Enable line number increment in packet formatter for CSI-2 interface
[ ] Enable EoTp in packet formatter for DSI interface

Clock
Target TX Line Rate (per lane): 600 (160 - 1500 Mbps)
Target TX Data Rate: 2400.00 Mbps
D-PHY Clock Mode: Continuous [✓ Non-continuous]
Reference Clock Frequency: 75 (24 to 150.0 MHz)

Generated Values
TX Line Rate: 600.00 Mbps
TX Data Rate: 2400.00 Mbps
D-PHY Clock Frequency: 300.00 MHz
Byte Clock Frequency: 75.0000 (up to 150.0 MHz)
Deviation from Target Values: 0.000 %

Buffer Implementation
FIFO Depth: 64 (Depth >= 8)
FIFO Type: [✓ EBR] LUT

Initialization
tINIT_SLAVE Value: 1000 (Number of byte clock cycles, > 0)
[ ] Bypass tINIT counter

Miscellaneous
[✓] Enable miscellaneous status signals

**Figure 3.14. tx_dphy IP Creation in Clarity Designer**

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

- TX Interface – Select CSI-2 or DSI.
- Number of TX Lanes – Set according to channel configuration. Must match NUM_TX_LANE_* setting.
- TX Gear – Set according to TX Line Rate. In general, 16 must be selected when TX Line Rate is 900 or higher.
- Bypass packet formatter – Must be enabled (checked).
- TX Line Rate – Set according to channel configuration. This value must be equal to or larger than (number of RX channel) x (RX channel bandwidth) / (number of TX lanes).
- D-PHY Clock Mode – Set according to channel configuration. Select Continuous when the horizontal blanking period is short in DSI. In case of CSI-2, Non-continuous and defining KEEP_HS could be an option for short horizontal blanking.
- Reference Clock Frequency – Set the appropriate value, which can be obtained from ref_clk_i pin, continuous rx_byte_clk, or on-chip GPLL. This clock frequency must be in one of the following ranges:
  - 24-30 MHz
  - 48-60 MHz
  - 72-90 MHz
  - 96-150 MHz
- tINIT_SLAVE Value – 1000 (default) is recommended.
- Bypass tINIT counter – Disabled (unchecked) is recommended.

- Enable miscellaneous status signals – must be set to enabled (checked).
- Protocol Timing Parameters tab – Default values are recommended.

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

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

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

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

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

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

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

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

## 3.7. Clock Distribution

In case of non-continuous clock (HS_LP) mode, two RX byte clock trees (non-continuous RX byte clock and continuous RX byte clock) exist in CrossLink device. The following are possible candidates of the continuous clock in case of non-continuous clock mode:

- PLL outputs driven by the external reference clock
- Continuous clock by the external reference clock (either direct or after divider)

The sample design (mipi_duplicator.v) has typical clock assignments to rx_clk_byte_fr and rx_clk_lp_ctrl according to the clock mode on RX channel. The code snippet is shown below. rx_clk_lp_ctrl (clock signal for LP and HS mode control module for clock lane) could be different from rx_clk_byte_fr (continuous byte clock for RX channel), but recommended to be the same to save the primary clock tree resources. *int_gpll* is the name used in this top-level design. This name has to be changed if the different name is used.

```
`ifdef RX_CLK_MODE_HS_ONLY
       assign rx_clk_byte_fr = rx_clk_byte_hs;
       assign rx_clk_lp_ctrl = 1'b1;
`else
// User must provide the continuous RX byte clock from ref_clk_i or somewhere
       assign rx_clk_byte_fr = pll_clkop;
//     assign rx_clk_byte_fr = pll_clkos;
       assign rx_clk_lp_ctrl = pll_clkop;
`endif

/////////////////////////////////////////////////////////////////////////////////
///// Reference Clock generation to TX D-PHY
///// tx_refclk must be in the ranges of 24-30, 48-60, 72-90, or 96-150 MHz
///// added option to use Crosslink GPLL to avoid frequency holes of Mixed PLL
///// One of rx_clk_byte_fr could be used as tx_refclk, then no need of ref_clk_i
/////////////////////////////////////////////////////////////////////////////////
`ifdef USE_GPLL
       int_gpll int_gpll (
       `ifdef EXT_REF_CLK
              .CLKI  (ref_clk_i),
       `else
              .CLKI  (rx_clk_byte_fr),
       `endif
              .CLKOP (pll_clkop),
       `ifdef RX_CLK_MODE_HS_LP
              .CLKOS (pll_clkos),
       `endif
              .LOCK  (gpll_lock)
       );
       assign tx_refclk = pll_clkop;
`else
       assign tx_refclk = rx_clk_byte_fr;
`endif

/////////////////////////////////////////////////////////////////////////////////
```

On TX side, using continuous or non-continuous clock mode does not affect the number of necessary clock trees (always uses one clock tree per TX channel). In general, using the same clock mode as RX side is recommended and user have to pay extra attention if user want to use non-continuous mode on TX side while RX side uses continuous mode since LP-HS transition time on RX side might not be long enough to allow TX side to makes LP-HS transitions, including the clock lane, which leads to FIFO overflow. To feed a clock to TX D-PHY IP, the external clock is necessary if continuous RX byte clock is not appropriate to generate the desired clock for TX D-PHY. The clock to TX D-PHY must be continuous and within one of the following ranges:

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

# 4. Design and File Modifications

Depends on the clocking scheme, some modifications are required depending on user configuration in addition to two directive files (synthesis_directives.v, simulation_directives.v).

## 4.1. Top-Level RTL

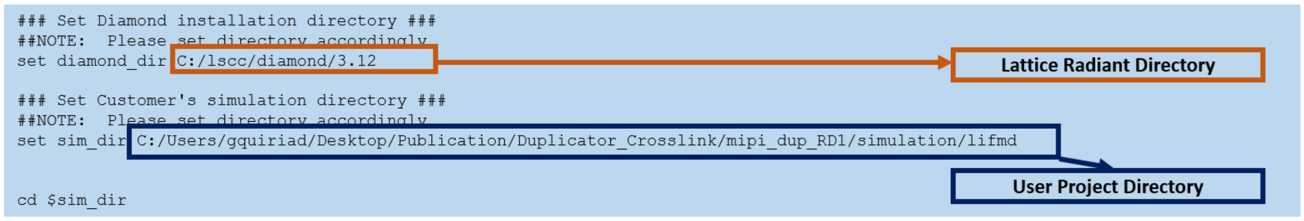
The current top-level file (mipi_duplicator.v) includes sample assignments and GPLL instantiation according to RX clock mode and external clock availability defined in synthesis_directives.v as described in Clock Distribution section. This part may need to be modified depending on the clocking scheme.

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

# 5. Design Simulation

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

- Diamond installation directory path
- User project directory
- Comment out if GPLL is not in use



```
### Set Diamond installation directory ###
##NOTE:  Please set directory accordingly
set diamond_dir C:/lscc/diamond/3.12

### Set Customer's simulation directory ###
##NOTE:  Please set directory accordingly
set sim_dir C:/Users/gquiriad/Desktop/Publication/Duplicator_Crosslink/mipi_dup_RD1/simulation/lifmd

cd $sim_dir
```

Figure 5.1. Script Modification #1

```
##-- Modelsim work library creation
vlib  work
vdel -lib work -all
vlib work

vlog +incdir+./../../source/verilog/lifmd+./+./../../testbench/verilog \
./../../testbench/verilog/mipi_duplicator_tb.v \
./../../source/verilog/lifmd/mipi_duplicator.v \
./../../source/verilog/lifmd/beh/mipi_parser_beh.v \
./../../source/verilog/lifmd/rx_buffer.v \
./../../source/verilog/lifmd/beh/tx_ctrl_beh.v \
./../../source/verilog/lifmd/tx_dphy_if.v \
./../../source/verilog/lifmd/clk_xfer.v \
./../../int_gpll/int_gpll.v \

##### RX D-PHY IP #####
vlog ./../../rx/rx_dphy/rx_dphy.v \
./../../rx/rx_dphy/rx_dphy_dphy_rx.v \
./../../rx/rx_dphy/rx_dphy_rx_global_ctrl.v \
./../../rx/rx_dphy/rx_dphy_dphy_rx_wrap.v \
./../../rx/rx_dphy/rx_dphy_dphy_wrapper.v \
./../../rx/rx_dphy/rx_dphy_soft_dphy_rx.v \
./../../rx/rx_dphy/dphy_rx_eval/rx_dphy/src/beh_rtl/soft_dphy_rx_beh.v \
./../../rx/rx_dphy/dphy_rx_eval/rx_dphy/src/beh_rtl/rx_global_ctrl_beh.v \
./../../rx/rx_dphy/dphy_rx_eval/rx_dphy/src/beh_rtl/dphy_rx_wrap_beh.v \

##### TX D-PHY IP #####
vlog ./../../tx/tx_dphy/tx_dphy.v \
./../../tx/tx_dphy/tx_dphy_dphy_tx.v \
./../../tx/tx_dphy/tx_dphy_synchronizer.v \
./../../tx/tx_dphy/tx_dphy_tx_global_operation.v \
./../../tx/tx_dphy/tx_dphy_tinit_count.v \
./../../tx/tx_dphy/tx_dphy_dci_wrapper.v \
./../../tx/tx_dphy/dphytx_eval/tx_dphy/src/beh_rtl/tinit_count_beh.v \
./../../tx/tx_dphy/dphytx_eval/tx_dphy/src/beh_rtl/tx_global_operation_beh.v \

vsim -L work -L pmi_work -L ovi_lifmd top
view wave
add wave /*
do wave.do
run -all
```

**Figure 5.2. Script Modification #2**

User need to modify simulation_directives.v according to the configuration (refer to Simulation Directives for details). By executing the script in Active HDL, compilation and simulation are executed automatically. The testbench takes all data comparison between the expected data and output data from the RD, including VC, ECC and CRC data. It shows following statements while running and doing data comparison:

```
25565695400 DPHY RX CH Lane 3 : Driving with data = 2c
# [25565862220][DPHY_TX0_CHK] payload data = 19 3b 90 0f --- [25565862220][DPHY_TX0_CHK]Data matches ch0 : 19 3b 90 0f
#          25567029000 DPHY RX CH Driving Data
#          25567029000 DPHY RX CH Lane 0 : Driving with data = b3
#          25567029000 DPHY RX CH Lane 1 : Driving with data = 4b
#          25567029000 DPHY RX CH Lane 2 : Driving with data = 99
#          25567029000 DPHY RX CH Lane 3 : Driving with data = 07
# [25567195980][DPHY_TX0_CHK] payload data = 7f 27 dd 56 --- [25567195980][DPHY_TX0_CHK]Data matches ch0 : 7f 27 dd 56
#          25568362600 DPHY RX CH Driving Data
#          25568362600 DPHY RX CH Lane 0 : Driving with data = 99
#          25568362600 DPHY RX CH Lane 1 : Driving with data = e2
#          25568362600 DPHY RX CH Lane 2 : Driving with data = d0
#          25568362600 DPHY RX CH Lane 3 : Driving with data = 49
# [25568529740][DPHY_TX0_CHK] payload data = 77 e2 8c bf --- [25568529740][DPHY_TX0_CHK]Data matches ch0 : 77 e2 8c bf
#          25569696200 DPHY RX CH Driving Data
#          25569696200 DPHY RX CH Lane 0 : Driving with data = 31
#          25569696200 DPHY RX CH Lane 1 : Driving with data = 39
#          25569696200 DPHY RX CH Lane 2 : Driving with data = d4
#          25569696200 DPHY RX CH Lane 3 : Driving with data = ac
```

When the simulation is finished, the following statements are shown:

```
# [27834315820][DPHY_TX0_CHK]CRC = 27a7
# [27834315820][DPHY_TX0_CHK] Trail Bytes in lane #2/#3 = 00 00 --- Check OK
# [27835649580][DPHY_TX0_CHK] Trail Bytes = 00 ff 00 00 --- Check OK
# [27841484300][DPHY_TX0_HS_LP_CHK] HS to LP11 Transition on D0 lane with HS-TRAIL period = 85 ns
# [28084199340][DPHY_TX0_HS_LP_CHK] LP11 to LP01 Transition on D0 lane
# [28090867180][DPHY_TX0_HS_LP_CHK] LP01 to LP00 Transition on D0 lane with LP01 period = 66 ns
# [28097535500][DPHY_TX0_HS_LP_CHK] LP00 to HS00 Transition on D0 lane with LP00 period = 66 ns
# [28118789420][DPHY_TX0_HS_LP_CHK] HS00 to HS Transition on D0 lane with LP00+HS00 period = 279 ns
# [28121040140][DPHY_TX0_CHK] Short Packet detected : Data type = 01
# Check the Packet Header to identify incoming RX channel --- 01 01 00 1d matches CH #0 Data
# [28122373420][DPHY_TX0_CHK] Trail Bytes = ff ff ff ff --- Check OK
# [28129541900][DPHY_TX0_HS_LP_CHK] HS to LP11 Transition on D0 lane with HS-TRAIL period = 85 ns
# [28208224140][DPHY_TX0_HS_LP_CHK] HS to LP11 Transition on clock lane with TRAIL period = 73 ns
#          29330364900 DPHY RX CH CLK CONT : Driving CLK-Trail
#          29336257100 TEST END
#
# TX CH #0 : 18 / 18 HS transmission completed successfully
#
### Simulation Completed ###
```

One HS transmission is most likely either Frame Start/End short packet or long packet of one active video line in case of CSI-2. In case of DSI, one HS transmission might include multiple short and long packets. The result is if the numerator is equal to denominator in the statements.

User should set small values in NUM_LINES and NUM_FRAMES directives in simulation_directives.v file, especially in the first simulation trial to minimize simulation time. On the other hand, it makes sense to set the actual value to NUM_PIXELS directives and RX_DPHY_LPS_GAP directives when TX bandwidth cannot have extra margin against total RX bandwidth. By setting realistic values, FIFO overflow is detected when the margin is not large enough.

Figure 5.3 shows an example of DSI duplication of single TX channel in continuous clock mode on RX and TX channel. RX is 2-lane and TX is 1-lane. Both lane bandwidth is the same and delay time between RX and TX is very small (BD_RD_DLY = 10).

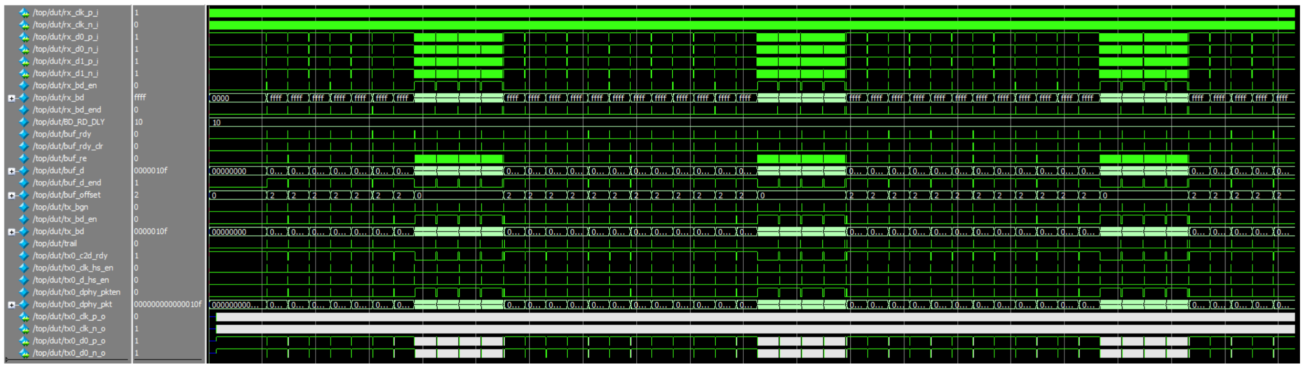


**Figure 5.3. Functional Simulation Example of DSI with Single TX Channel**

Figure 5.4 shows an example of DSI duplication of two TX channels with the RX channel in continuous clock mode and TX channels in non-continuous clock mode. RX is 4-lane and TX is 2-lane. RX lane bandwidth is 600 Mbps and TX lane bandwidth is 1200 Mbps. In this case, BD_RD_DLY is set to 100 to have enough read delay to avoid rx_buffer underflow.

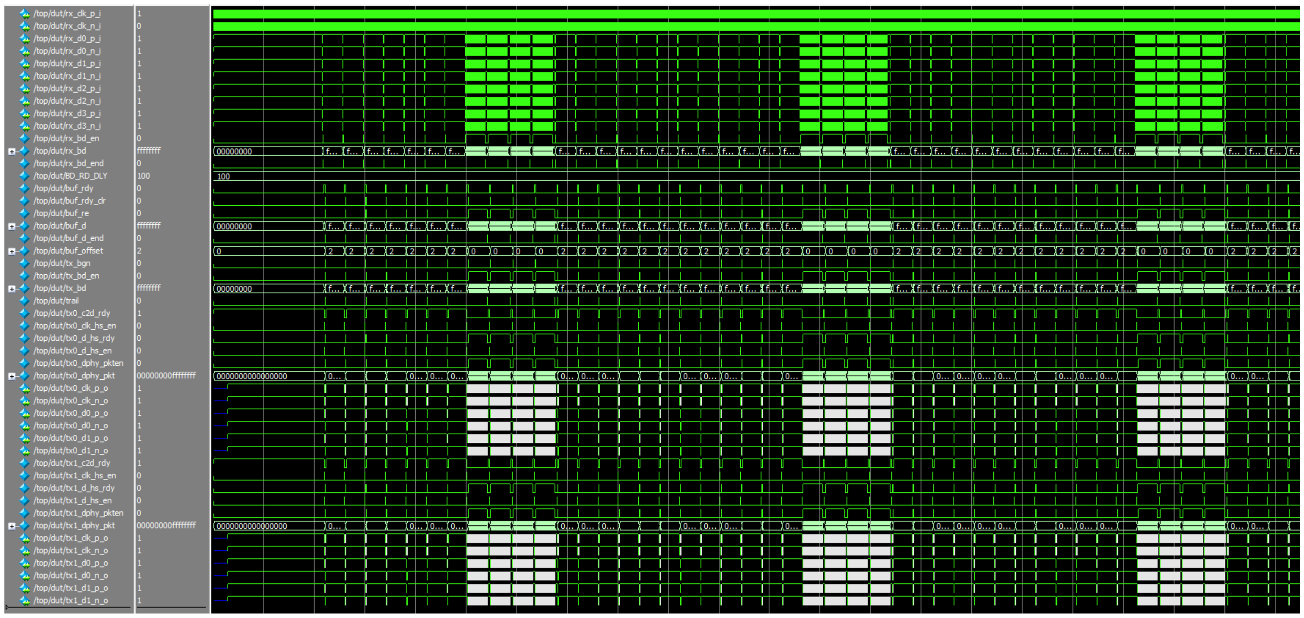


**Figure 5.4. Functional Simulation Example of DSI with Two TX Channels**

# 6. Known Limitations

The following are the limitations of this reference design:

- Data lane(s) must go into LP (Low Power) mode during horizontal blanking periods.
- In case of 2-channel output, both TX channels must have the same configuration.

# 7. Design Package and Project Setup

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

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

User can modify the directives for own configuration.

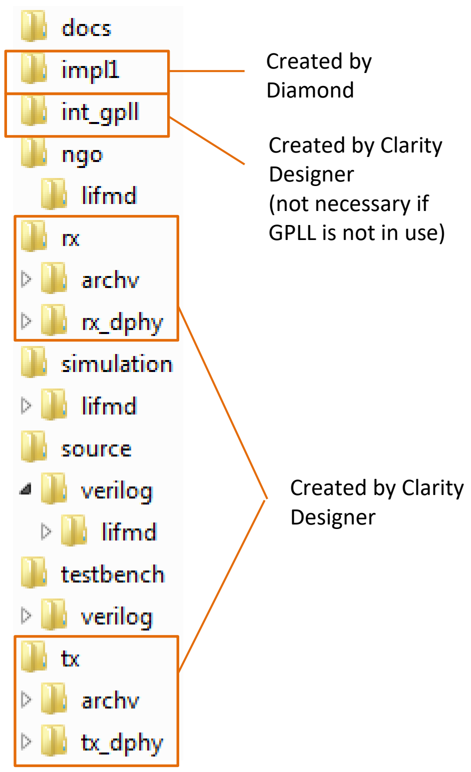


**Figure 7.1. Directory Structure**

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

- [mipi_parser]
  - L*G** – * Number of RX lanes; 1, 2, or 4, ** RX Gear; 8 or 16
  - DSI or CSI2 – D-PHY Type; DSI or CSI-2
  - WFS or NWFS – wait for Frame Start or Vsync to begin capturing or immediate capture upon reset release
- [tx_ctrl]
  - L*G** – * Number of TX lanes; 1, 2, or 4, ** TX Gear; 8 or 16

The blackbox files (*_bb.v) are also contained in the *ngo\lifmd* folder. User can include corresponding _bb.v files as design files in Diamond according to own configuration. Alternatively, it is possible to include all _bb.v files as long as *mipi_duplicator* is specified as the top-level design. Figure 7.2 shows design files used in the Diamond project. Clarity Designer creates three .sbx files. In this example, all 30 _bb.v files are included in the project. By specifying mipi_duplicator as a top-level design, all unnecessary files are ignored. GPLL design file must be added if necessary.

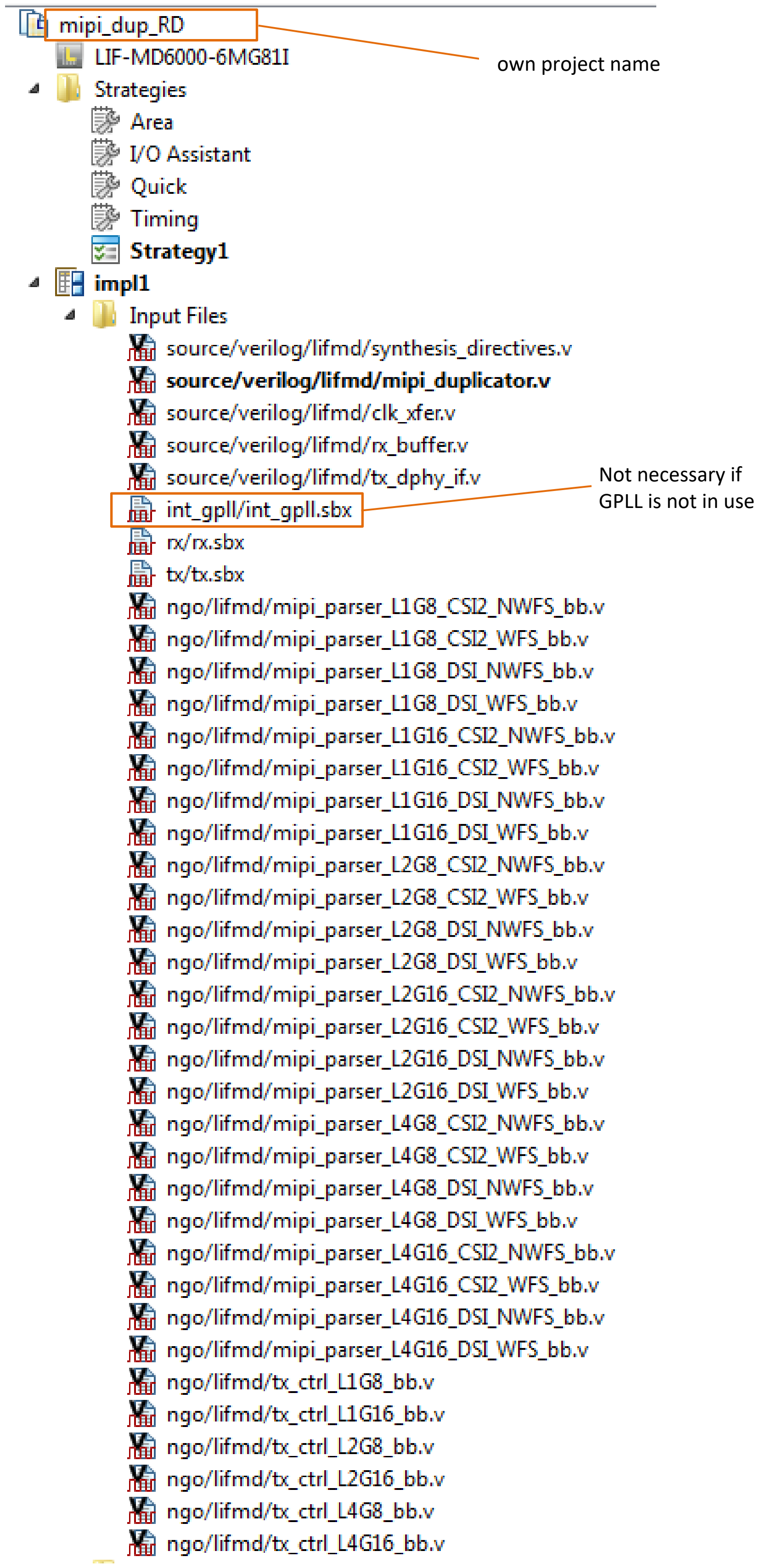
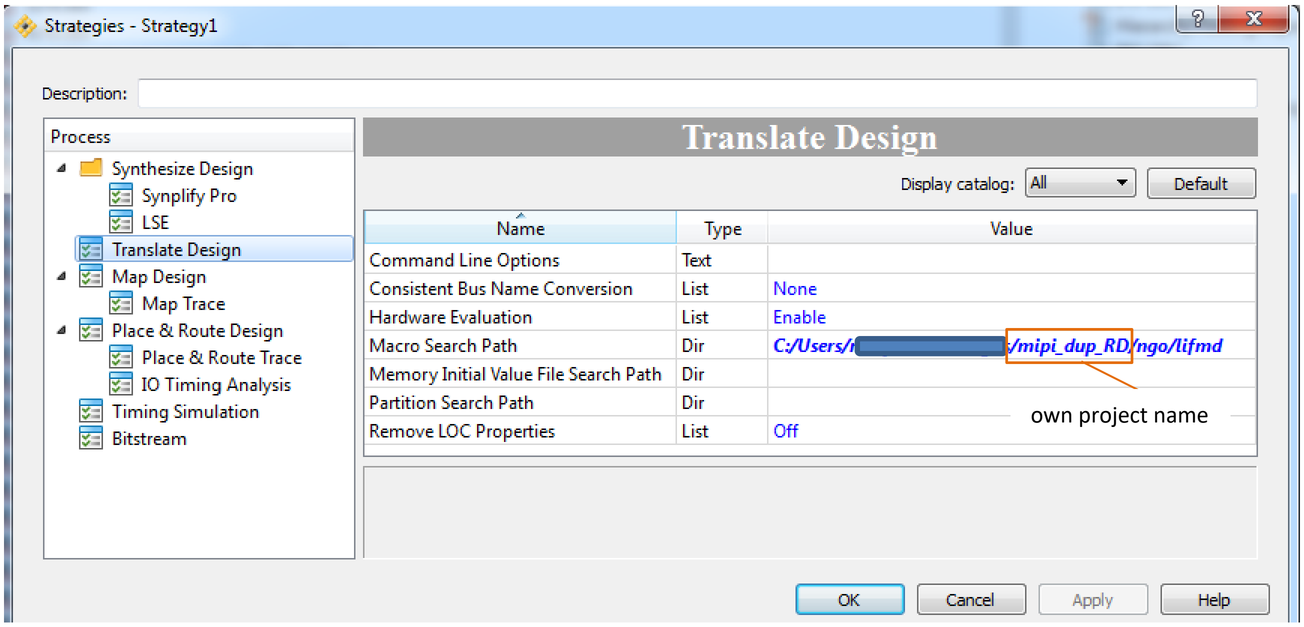

mipi_dup_RD
LIF-MD6000-6MG81I
Strategies
Area
I/O Assistant
Quick
Timing
Strategy1
impl1
Input Files
source/verilog/lifmd/synthesis_directives.v
source/verilog/lifmd/mipi_duplicator.v
source/verilog/lifmd/clk_xfer.v
source/verilog/lifmd/rx_buffer.v
source/verilog/lifmd/tx_dphy_if.v
int_gpll/int_gpll.sbx
rx/rx.sbx
tx/tx.sbx
ngo/lifmd/mipi_parser_L1G8_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L1G8_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L1G8_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L1G8_DSI_WFS_bb.v
ngo/lifmd/mipi_parser_L1G16_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L1G16_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L1G16_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L1G16_DSI_WFS_bb.v
ngo/lifmd/mipi_parser_L2G8_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L2G8_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L2G8_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L2G8_DSI_WFS_bb.v
ngo/lifmd/mipi_parser_L2G16_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L2G16_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L2G16_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L2G16_DSI_WFS_bb.v
ngo/lifmd/mipi_parser_L4G8_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L4G8_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L4G8_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L4G8_DSI_WFS_bb.v
ngo/lifmd/mipi_parser_L4G16_CSI2_NWFS_bb.v
ngo/lifmd/mipi_parser_L4G16_CSI2_WFS_bb.v
ngo/lifmd/mipi_parser_L4G16_DSI_NWFS_bb.v
ngo/lifmd/mipi_parser_L4G16_DSI_WFS_bb.v
ngo/lifmd/tx_ctrl_L1G8_bb.v
ngo/lifmd/tx_ctrl_L1G16_bb.v
ngo/lifmd/tx_ctrl_L2G8_bb.v
ngo/lifmd/tx_ctrl_L2G16_bb.v
ngo/lifmd/tx_ctrl_L4G8_bb.v
ngo/lifmd/tx_ctrl_L4G16_bb.v

**Figure 7.2. Project Files**

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



| Name | Type | Value |
|---|---|---|
| Command Line Options | Text | |
| Consistent Bus Name Conversion | List | None |
| Hardware Evaluation | List | Enable |
| Macro Search Path | Dir | C:/Users/[redacted]/mipi_dup_RD/ngo/lifmd |
| Memory Initial Value File Search Path | Dir | |
| Partition Search Path | Dir | |
| Remove LOC Properties | List | Off |

**Figure 7.3. Path Setting for .ngo Files**

# 8. Resource Utilization

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

**Table 8.1. Resource Utilization Examples**

| Configuration | LUT % | FF % | EBR | I/O |
|---|---|---|---|---|
| RX channel with 4-lane, Gear 8 to two TX channels with 2-lane, Gear 16 | 32 | 23 | 8 | 23 |
| RX channel with 4-lane, Gear 8 to one TX channel with 4-lane, Gear 8 | 16 | 14 | 2 | 21 |
| RX channel with 1-lane, TX Gear 8 to one TX channel with 1-lane, Gear 8 | 14 | 11 | 2 | 9 |

# 9. References

- MIPI® Alliance Specification for D-PHY Version 1.1
- MIPI® Alliance Specification for Camera Serial Interface 2 (CSI-2) Version 1.1
- MIPI® Alliance Specification for Display Serial Interface (DSI) Version 1.1
- CSI-2/DSI D-PHY Receiver Submodule IP User Guide (FPGA-IPUG-02025)
- CSI-2/DSI D-PHY Transmitter Submodule IP User Guide (FPGA-IPUG-02024)
- For more information on the CrossLink FPGA device, visit http://www.latticesemi.com/Products/FPGAandCPLD/CrossLink.
- For more information on the CrossLinkPlus FPGA device, visit http://www.latticesemi.com/Products/FPGAandCPLD/CrossLinkPlusPlus.
- For complete information on Lattice Diamond Project-Based Environment, Design Flow, Implementation Flow, Tasks, and Simulation Flow, see the Lattice Diamond User Guide.

## Technical Support Assistance

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

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

# Revision History

### Revision 1.1, January 2023

| Section | Change Summary |
|---|---|
| All | Updated the title of the document from *1 to N MIPI CSI-2/DSI Duplicator* to *1 to N MIPI CSI-2/DSI Duplicator with Crosslink*. |
| Introduction | • Updated the Table 1.1. Supported Device and IP:<br>  • Updated D-PHY Receiver IP version from *1.4* to *1.6*.<br>  • Updated D-PHY Transmitter IP version from *1.3* to *1.4.*<br>• Updated Figure 1.2. Bandwidth and Parameter Calculator.<br>• Added the sentence *The excel file for the parameter calculator is located in the design package source code. Open the source code folder, locate the docs folder, and locate the mipi_dup_RD.xlsx*. |
| Design and Module Description | • Updated below figures:<br>  • Figure 3.1. rx_dphy IP Creation in Clarity Designer<br>  • Figure 3.2. Active Data Detection (CSI-2 Frame Start)<br>  • Figure 3.3. End of Active Data Detection (CSI-2 Long Packet)<br>  • Figure 3.4. Short Packet Write and Read<br>  • Figure 3.5. rx_buffer Write/Read Example with Depth = 512, BD_RD_DLY = 400<br>  • Figure 3.6. rx_buffer FIFO Overflow with Depth = 512, BD_RD_DLY = 1000<br>  • Figure 3.7. rx_buffer Successful Transactions with Depth = 1024, BD_RD_DLY = 700<br>  • Figure 3.8. TX bandwidth = 2 x RX Bandwidth with Depth = 1024, BD_RD_DLY = 700<br>  • Figure 3.9. Short Packet Transaction (NUM_TX_LANE_4, TX_GEAR_8)<br>  • Figure 3.10. End of Long Packet Transaction (NUM_RX_LANE_4, RX_GEAR_8 to NUM_TX_LANE_4, TX_GEAR_8)<br>  • Figure 3.11. Short Packet Transaction in Non-Continuous Clock Mode<br>  • Figure 3.12. Global Sequence in Non-Continuous Clock Mode without KEEP_HS<br>  • Figure 3.13. Global Sequence in Non-Continuous Clock Mode with KEEP_HS<br>  • Figure 3.14. tx_dphy IP Creation in Clarity Designer<br>• Updated the below texts:<br>  • From Num_RX_LANE_2 to Num_RX_LANE_4,<br>  • From *rx_offset = 1* to *rx_offset = 0*<br>  • From *LSB 1 byte data (0x96)* to *LSB data (0x1a76)*<br>  • From *MSB 1 byte (0x00)* to *MSB 1 byte (0xff)*<br>  • From *BD_RD_DLY = 800* to *BD_RD_DLY = 400*<br>  • From *tx_c2d_rdy = 1* to *tx0_c2d_rdy = 1* |
| Design Simulation | • Updated below figures:<br>  • Figure 5.1. Script Modification #1<br>  • Figure 5.2. Script Modification #2<br>  • Figure 5.3. Functional Simulation Example of DSI with Single TX Channel<br>  • Figure 5.4. Functional Simulation Example of DSI with Two TX Channels<br>• Replaced HDL with ModelSim.<br>• Deleted the sentence *which means total bandwidth of TX (1500 x 2 = 3000 Mbps) is faster than RX (600 x 4 = 2400 Mbps).* |

### Revision 1.0, April 2020

| Section | Change Summary |
|---|---|
| All | Initial release. |

www.latticesemi.com