OpenLDI/FPD-LINK/LVDS Transmitter Interface IP

User Guide

FPGA-IPUG-02022-1.2

October 2019

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

1. Introduction

The Lattice Semiconductor OpenLDI/FPD-LINK/LVDS Transmitter Interface IP translates parallel video streams to LVDS interface for an FDP-Link connection to displays.

The increasing demand for better displays makes bridging applications very popular. The Flat Panel Display Link (FPD-Link) is a common application interface. Applications such as Channel Link, Flat Panel Display Link (FPD-Link), and Camera Link use LVDS interface for physical layer.

The Low Voltage Differential Signaling (LVDS) standard is commonly used for high-speed differential interface in consumer devices, industrial control, medical, and automotive. The LVDS interface offers low voltage, low power and improved signal integrity advantages over single-ended technology.

The 7:1 LVDS interface is a popular standard for source synchronous interfaces which consist of multiple data bits and clocks. Typically, one channel of 7:1 LVDS interface consists of five LVDS pairs (one clock and four data) depending on the data type it supports.

This document describes the use of Lattice FPGA technology for applications requiring LVDS interface and how to use the IP. This design can be used in multiple configurations.

This document is for OpenLDI/FPD-LINK/LVDS Transmitter Interface IP design version 1.x.

Figure 1.1. Sample MIPI DSI interfaced to OpenLDI/FPD-LINK/LVDS Transmitter System Diagram

1.1. Quick Facts

Table 1.1 provides quick facts about the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP for CrossLink™ and CrossLinkPlus™ devices.

Table 1.1. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Quick Facts

		OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Configuration			
		1x7 Tx, RGB888 Configuration	**2x7 Tx, RGB888 Configuration**	**1x14 Tx, RGB888 Configuration**	**2x14 Tx, RGB888 Configuration**
IP Requirements	FPGA Families Supported	CrossLink/CrossLinkPlus			
Resource Utilization	Targeted Device	LIF-MD6000-6MG81I			
	Data Path Width	Parallel input, 28-bit output	Parallel input , 56-bit output (28-bit per Tx channel)	Parallel input, 28-bit output	Parallel input , 56-bit output (28-bit per Tx channel)
	LUTs	39	40	40	38
	sysMEM™ EBRs	0	0	0	0
	Registers	18	18	18	18
	HW MIPI Block	0	0	0	0
	Programmable I/O	41	75	65	123
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.11 SP1			
	Synthesis	Lattice Synthesis Engine			
		Synplify Pro® N-2018.03L-SP1-1			
	Simulation	Aldec® Active HDL™ 10.5 Lattice Edition			

1.2. Features

The key features of the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP include:

- Compliant with Open LVDS Display Interface (OpenLDI) v0.95 specifications
- Transmits in OpenLDI unbalanced operating mode format
- Supports RGB888 and RGB666 video formats
- Supports transmitting in Dual Channel Flat Panel Display Link Protocol (7:1 LVDS)
- Supports 3-4 LVDS data lanes per channel
- Supports 1, 2, or 4 input pixel data per pixel clock
- Supports interfacing up to 9.6 Gb/s

Table 1.2. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Features Summary

IP Configuration	Options
Number of Channels	1, 2
Data type	RGB666, RGB888
Number of Tx Lanes per Channel[1]	3, 4
DDR Gear[2]	7, 14

Notes:

1. The Number of Tx Lanes per Channel is automatically set depending on the Data Type.
2. For Interface bandwidth greater than 900 Mb/s, it is recommended to use Gear 14. Otherwise, use Gear 7.

1.3. Conventions

1.3.1. Nomenclature

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

1.3.2. Data Ordering and Data Types

- The most significant bit within the pixel data is the highest index.
- Pixel data order before distribution to LVDS lanes is {Red[MSB:0], Green[MSB:0], Blue[MSB:0]}. One, two or four pixels may be sent for distribution to LVDS lanes in one pixel clock cycle, depending on the number of Tx channels and Tx gear setting. If there are multiple pixels per clock cycle, the pixel in the lower bits is the first received pixel. For instance, the pixel order for four pixels per clock is {pixel3, pixel2, pixel1, pixel0}, where pixel0 is received first and pixel3 is received last.
- Pixel data is transmitted over LVDS lanes according to OpenLDI 18-bit and 24-bit unbalanced operating mode format.

1.3.3. Signal Names

Signal names that end with:

- *_n* are active low
- *_i* are input signals
- *_o* are output signals
- *_io* are bidirectional signals

2. Functional Descriptions

The OpenLDI/FPD-LINK/LVDS Transmitter Interface IP converts pixel data into a standard OpenLDI serial video interface domain. The input interface for the design consists of the RGB control signals, pixel clock, and up to four pixel data per pixel clock. Output interface consists of a data bus, vertical and horizontal sync flags, a data enable, and a clock in OpenLDI (LVDS7:1) interface format and optional debug signal/s.

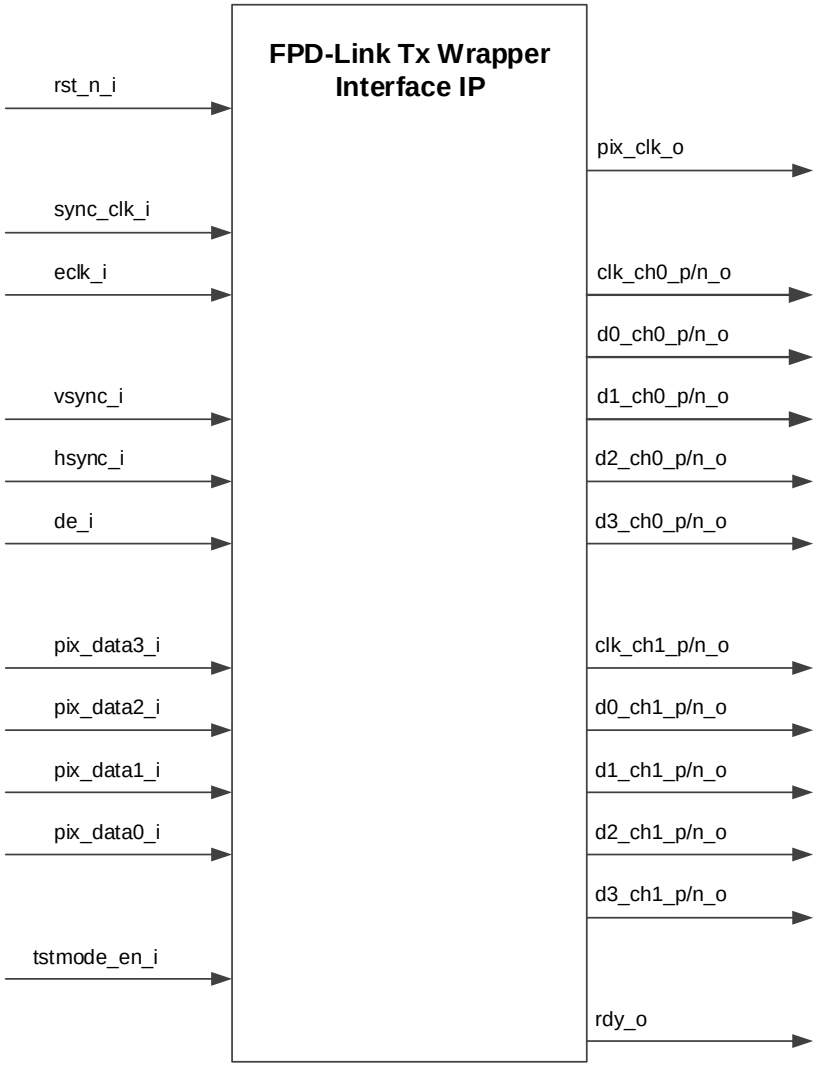

Figure 2.1. Top-level Block Diagram

Table 2.1. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Pin Function Description

Pin Name	Direction	Function Description
Clocks and Reset		
rst_n_i	I	Asynchronous active low system reset. 0 – System on reset
sync_clk_i	I	Clock for GDDR_SYNC; must be slower than all the clocks in the system.
eclk_i	I	Edge clock for LVDS side.
Pixel Domain Interface		
pix_clk_o	O	Output pixel clock.

Pin Name	Direction	Function Description
pix_data0_i	I	Input pixel data 0. Bus width depends on the data type selected. 24 – RGB888 18 – RGB666
pix_data1_i[1,2]	I	Input pixel data 1. Bus width depends on the data type selected. 24 – RGB888 18 – RGB666
pix_data2_i[1]	I	Input pixel data 2. Bus width depends on the data type selected. 24 – RGB888 18 – RGB666
pix_data3_i[1,2]	I	Input pixel data 3. Bus width depends on the data type selected. 24 – RGB888 18 – RGB666
de_i	I	Input data enable for parallel interface.
hsync_i	I	Input horizontal sync for parallel interface.
vsync_i	I	Input vertical sync for parallel interface.
OpenLDI/FPD-LINK Interface		
clk_ch0_p_o	O	Positive LVDS output clock to LVDS7:1 Tx Wrapper channel 0.
clk_ch0_n_o	O	Negative LVDS output clock to LVDS7:1 Tx Wrapper channel 0, complement of clk_ch0_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
clk_ch1_p_o	O	Positive LVDS output clock to LVDS7:1 Tx Wrapper channel 1.
clk_ch1_n_o	O	Negative LVDS output clock to LVDS7:1 Tx Wrapper channel 1, complement of clk_ch1_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d0_ch0_p_o	O	Positive LVDS output data lane 0 to LVDS7:1 Tx Wrapper channel 0.
d0_ch0_n_o	O	Negative LVDS output data lane 0 to LVDS7:1 Tx Wrapper channel 0, complement of d0_ch0_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d1_ch0_p_o	O	Positive LVDS output data lane 1 to LVDS7:1 Tx Wrapper channel 0.
d1_ch0_n_o	O	Negative LVDS output data lane 1 to LVDS7:1 Tx Wrapper channel 0, complement of d1_ch0_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d2_ch0_p_o	O	Positive LVDS output data lane 2 to LVDS7:1 Tx Wrapper channel 0.
d2_ch0_n_o	O	Negative LVDS output data lane 2 to LVDS7:1 Tx Wrapper channel 0, complement of d2_ch0_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d3_ch0_p_o[3]	O	Positive LVDS output data lane 3 to LVDS7:1 Tx Wrapper channel 0
d3_ch0_n_o[3]	O	Negative LVDS output data lane 3 to LVDS7:1 Tx Wrapper channel 0, complement of d3_ch0_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d0_ch1_p_o[4]	O	Positive LVDS output data lane 0 to LVDS7:1 Tx Wrapper channel 1.
d0_ch1_n_o[4]	O	Negative LVDS output data lane 0 to LVDS7:1 Tx Wrapper channel 1, complement of d0_ch1_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d1_ch1_p_o[4]	O	Positive LVDS output data lane 1 to LVDS7:1 Tx Wrapper channel 1.
d1_ch1_n_o[4]	O	Negative LVDS output data lane 1 to LVDS7:1 Tx Wrapper channel 1, complement of d1_ch1_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d2_ch1_p_o[4]	O	Positive LVDS output data lane 2 to LVDS7:1 Tx Wrapper channel 1.
d2_ch1_n_o[4]	O	Negative LVDS output data lane 2 to LVDS7:1 Tx Wrapper channel 1, complement of d2_ch1_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d3_ch1_p_o[3,4]	O	Positive LVDS output data lane 3 to LVDS7:1 Tx Wrapper channel 1.

Pin Name	Direction	Function Description
d3_ch1_n_o[3,4]	O	Negative LVDS output data lane 3 to LVDS7:1 Tx Wrapper channel 1, complement of d3_ch1_p_o. Not available in netlist, because this is automatically set to complement by the synthesis tool.
Miscellaneous		
tstmode_en_i[5]	tstmode_en_i[5]	tstmode_en_i[5]
rdy_o[6]	rdy_o[6]	rdy_o[6]

Notes:

1. Available only when the number of input pixels data is more than 1. See Table 2.2 for more details.
2. These pixel data are transmitted by channel 1 when dual channel in selected.
3. Available only when RGB888 data type is selected.
4. LVDS channel 1 output ports are not available when single LVDS channel is selected.
5. This is in beta mode, and was not tested in the initial release. See the Debug Features section for more details on enabling test mode.
6. Can be turned-on if MISC_ON is selected in the user interface upon IP generation, or MISC_ON is defined in the defines file.

2.1. Interface and Timing Diagrams

Figure 2.2 shows the timing diagram of parallel video input interface. It follows the standard parallel video interface protocol with VSYNC, HSYNC, Data Enable, and pixel data, all clocked by pixel clock. The number of pixels per pixel clock depends on the number of Tx Channels and Tx Gear selected.

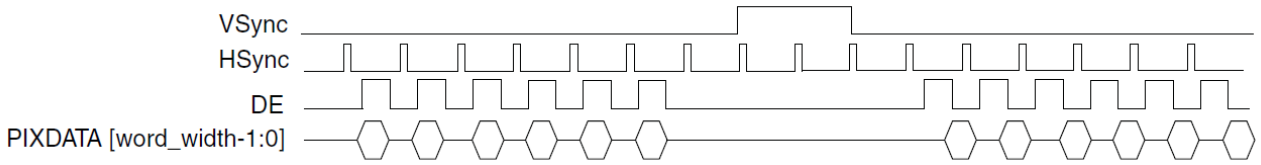

Figure 2.2. Parallel Video Input Interface Timing Diagram

Figure 2.3 shows the pixel data RGB mapping.

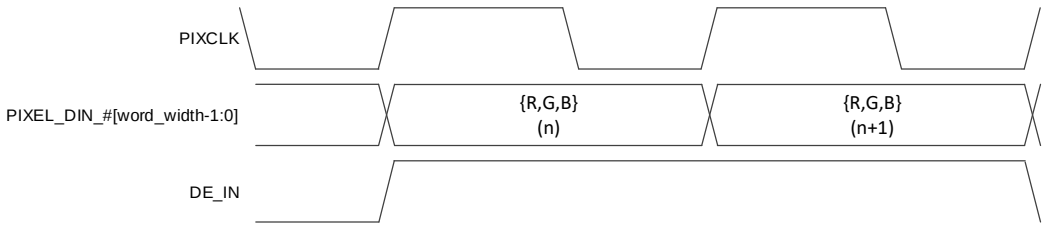

Figure 2.3. Input Pixel Data RGB Arrangement

Figure 2.4 shows the timing of LVDS7:1 output interface. There is a 2-bit offset between the rising edge of LVDS clock and the word boundary. Each word is 7-bit long.

DATAOUT0, DATAOUT1, DATAOUT2, and DATAOUT3 are the data lanes. CLOCKOUT is the LVDS clock lane. For every 7-bit data packet, LSB is the first output serial data to the receiver.

A processor sends parallel video packet data to the FPGA chip. Each data lane is serialized using DDR primitive, depending on Tx Gear setting (TX_GEAR). One channel of LVDS7:1 transmitter has a maximum of five lanes. Each channel consists of one LVDS clock pair and four LVDS data pairs (RGB888) or three LVDS data pairs (RGB666).

The clock lane is generated by feeding constant *1100011* or *11000111100011* depending on the Tx Gear. The clock is edge-aligned against data. The clock runs at 1/7th of the data rate, as per the standard for LVDS7:1 interface. Seven bits of data are transmitted in one LVDS clock cycle. The default mode for the LVDS operating system is Unbalanced, as this is commonly used. The maximum supported data rate per lane for LVDS is 1.2 Gb/s.

When Tx Gear is 14, the first pixel received is transmitted first, and the second pixel received is transmitted in the next clock cycle. Maximum two LVDS7:1 channels can be used. When dual channel is selected, additional data lanes are activated.

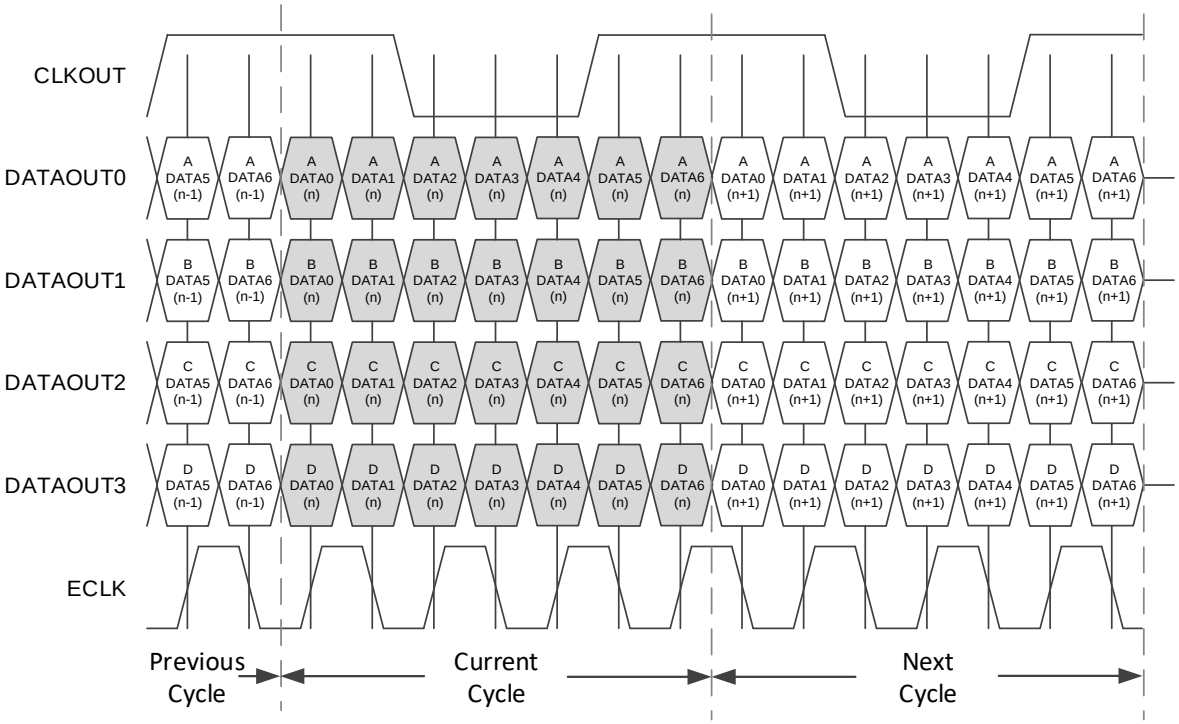

Figure 2.4. OpenLDI/FPD-LINK/LVDS Output Bus Waveform

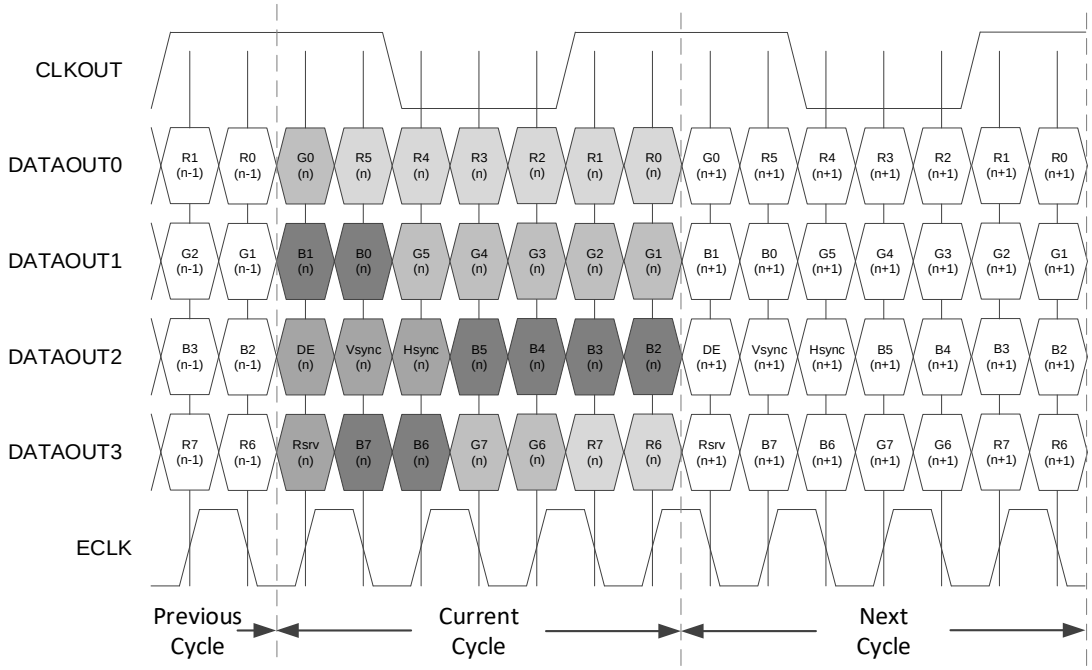

Figure 2.5. Single Channel OpenLDI/FPD-LINK/LVDS Output Bus Waveform for RGB888 Format

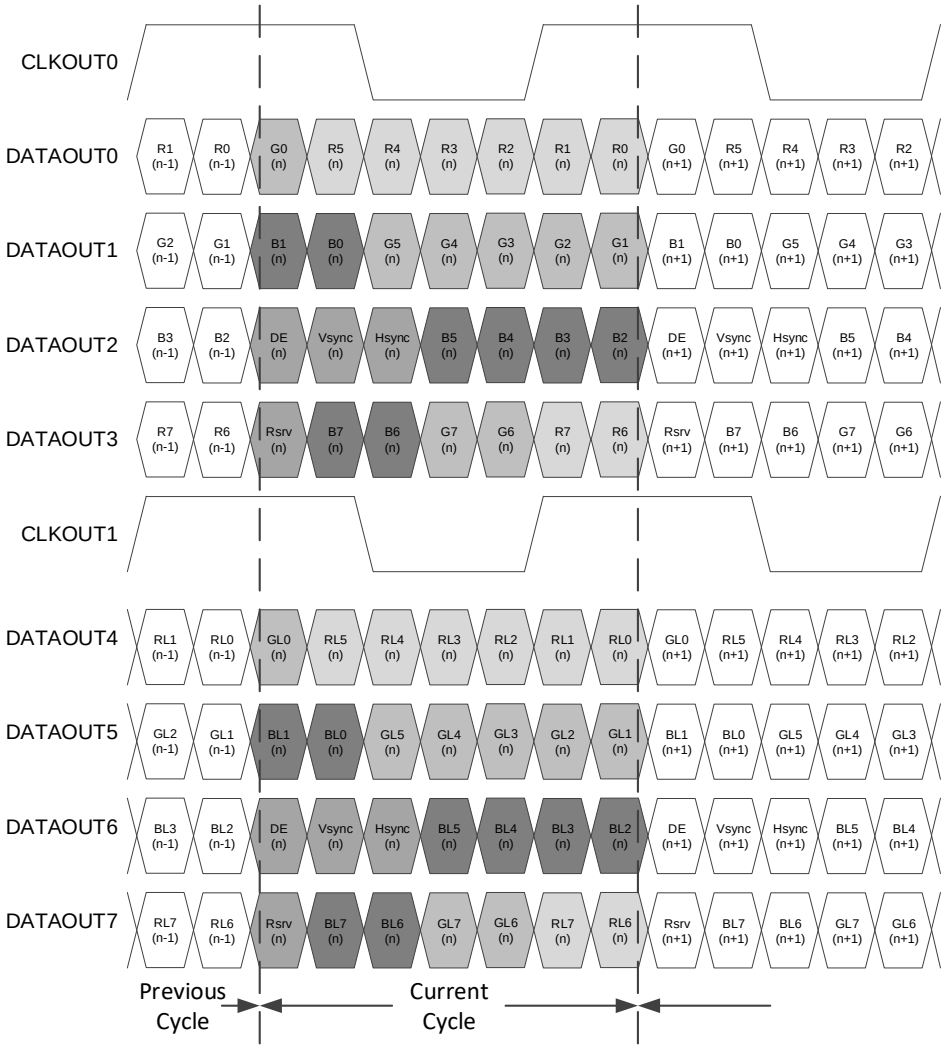

Figure 2.6. Dual Channel OpenLDI/FPD-LINK/LVDS Output Bus Waveform for RGB888 Format

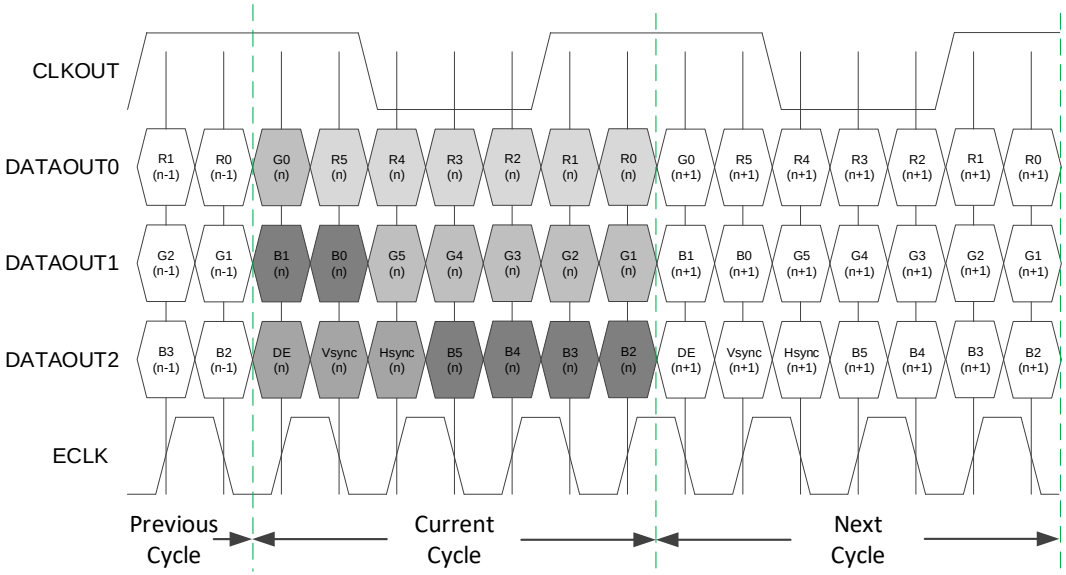

Figure 2.7. Single Channel OpenLDI/FPD-LINK/LVDS Output Bus Waveform for RGB666 Format

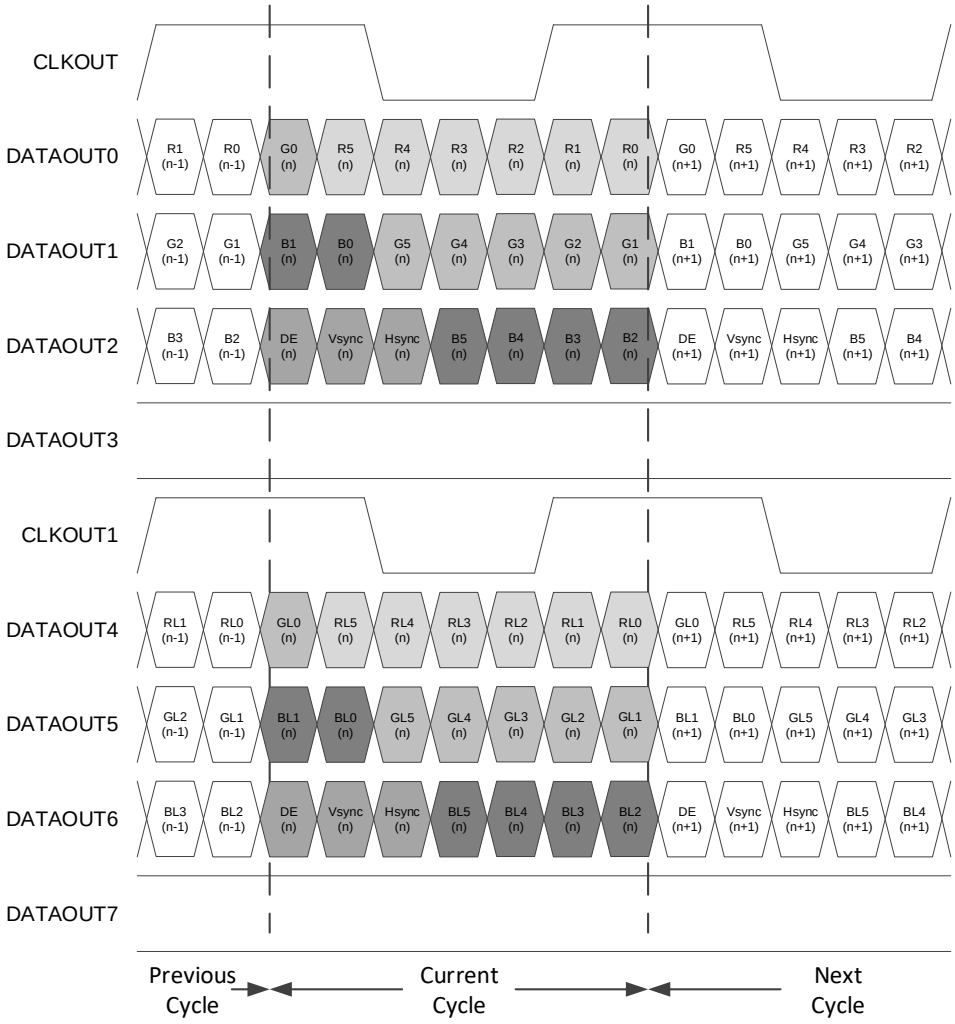

Figure 2.8. Dual Channel OpenLDI/FPD-LINK/LVDS Output Bus Waveform for RGB666 Format

Table 2.2. Input Pixel Data Summary

Number of FPD-Link Channels	Gear	Number of Input Pixel Data	Pixel Clock
1	7	1	LVDS Clock
	14	2	LVDS Clock/2
2	7	2	LVDS Clock
	14	4	LVDS Clock/2

General arrangement on how pixel data are mapped based on configuration are shown in Figure 2.9 and Figure 2.10.

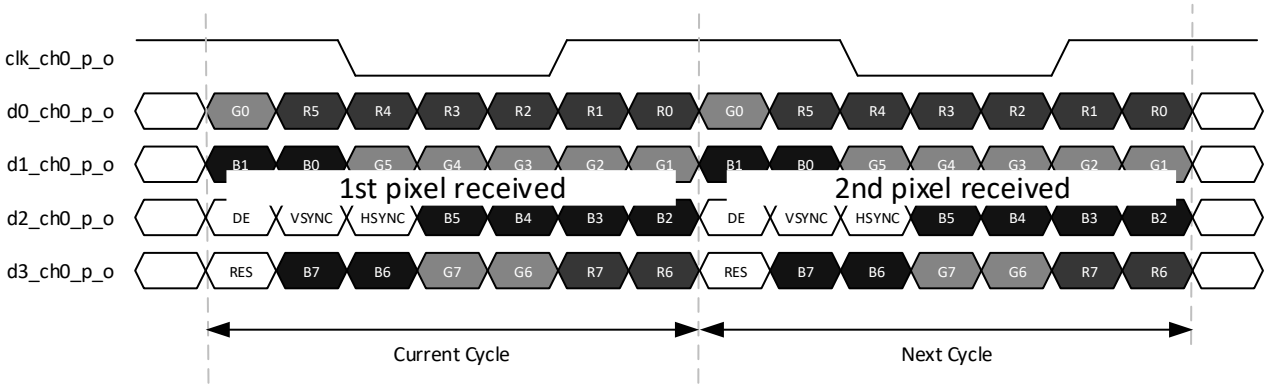

Figure 2.9. Output Pixel Data Arrangement for Single Channel OpenLDI/FPD-LINK/LVDS

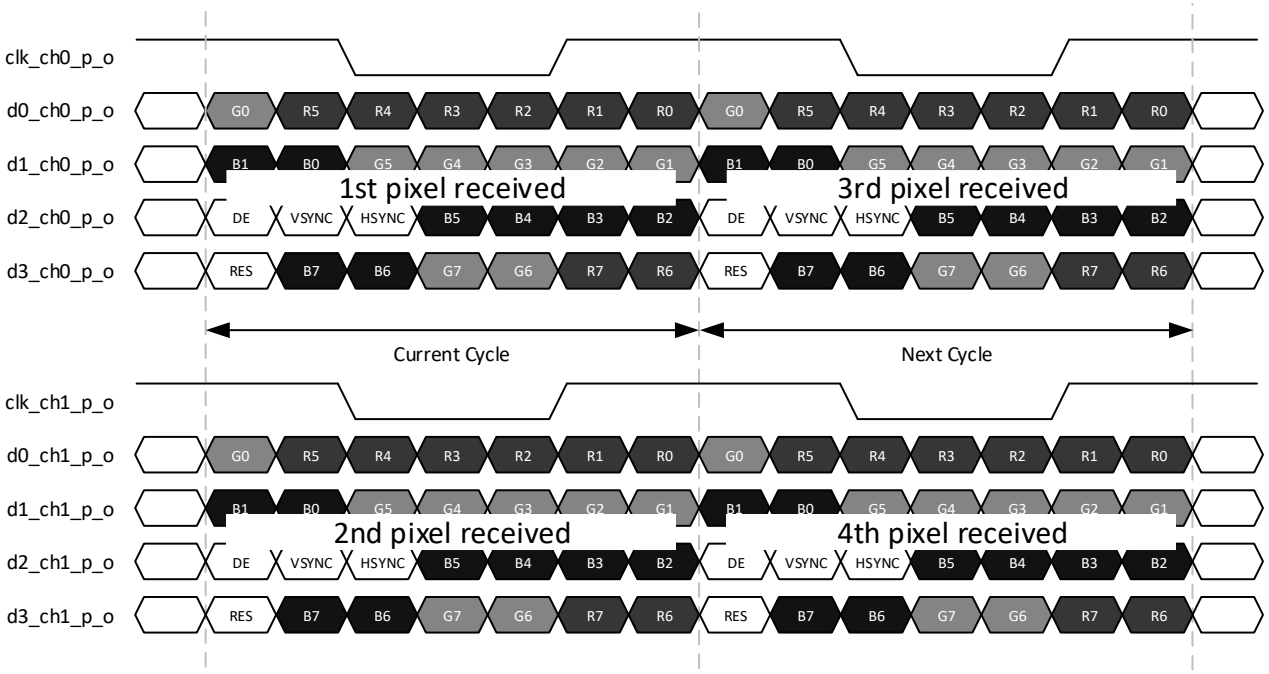

Figure 2.10. Output Pixel Data Arrangement for Dual Channel OpenLDI/FPD-LINK/LVDS

2.2. Clock, Reset and Initialization

Active low reset is used in the design with synchronous release. This is the system reset input connected to LVDS7:1 Tx module.

Follow this initialization and reset sequence:

1. Assert active low system reset for at least three clock cycles of the slowest clock (sync clock). It is expected that input clock is already stable after reset. Clock synchronization is started immediately after release of system reset.
2. Wait for *rdy_o* to be asserted. The *rdy_o* is used to indicate that LVDS7:1 Tx clock synchronization is done. Only when *rdy_o* is asserted, valid data can be sampled and correctly transmitted by FPD-Link IP.

The LVDS7:1 Tx Wrapper is now ready to receive valid data.

Note: Step 2 is used to make sure that the system is ready to receive valid data.

2.2.1. Clock Domains and Clock Domain Crossing

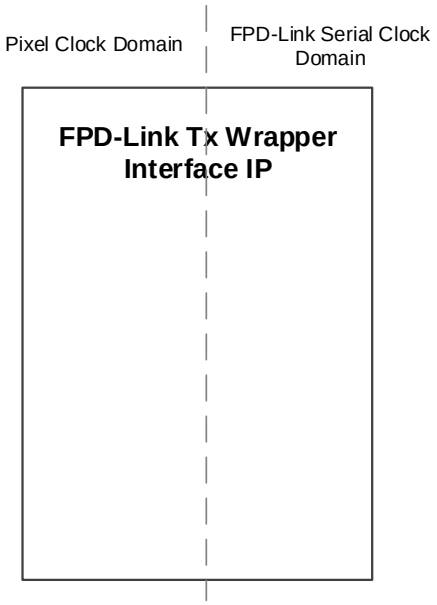

Figure 2.11. Clock Domain Crossing Block Diagram

Table 2.3. Clock Domain Crossing

Clock Domain Crossing	Handling Approach
Pixel Clock to FPD-Link Serial Clock	7:1/14:1 gearbox DDR Hard IP

The general formula for computing the required clocks of the IP:

Tx line rate (total) $= TX\ line\ rate\ (per\ lane) *\ no.of\ TX\ lane * no.of\ Tx\ Channel$

Pixclk (Pixel Clock) $= \frac{TX\ line\ rate\ (per\ lane)}{TX\ gear}$

Tx LVDS Output Clock $= pixclk * \frac{TX\ gear}{7}$

Tx LVDS ECLK $= pixclk * \frac{TX\ gear}{2}$

Number of Pixels per Pixel Clock $= \frac{TX\ gear}{7} * no.of\ Tx\ Channel$

Example:

IP configuration: Single channel FPD-Link Tx, RGB888 data type, 1039.5 Mb/s Tx Line Rate, Tx Gear 7

Tx line rate (total) $= 1039.5\ Mb/s *\ 4$ $= 4.158\ Gb/s$

Pixclk (Pixel Clock) $= \frac{1039.5\ Mb/s}{7}$ $= 148.5\ MHz$

Tx LVDS Output Clock $= 148.5\ MHz * \frac{7}{7}$ $= 148.5\ MHz$

Tx LVDS ECLK $= 148.5\ MHz * \frac{7}{2}$ $= 519.75\ MHz$

Number of Pixels per Pixel Clock $= \frac{7}{7} * 1$ $= 1\ Pixel\ per\ Pixel\ Clock$

2.3. Design and Module Description

2.3.1. FPD-Link Tx Wrapper Module

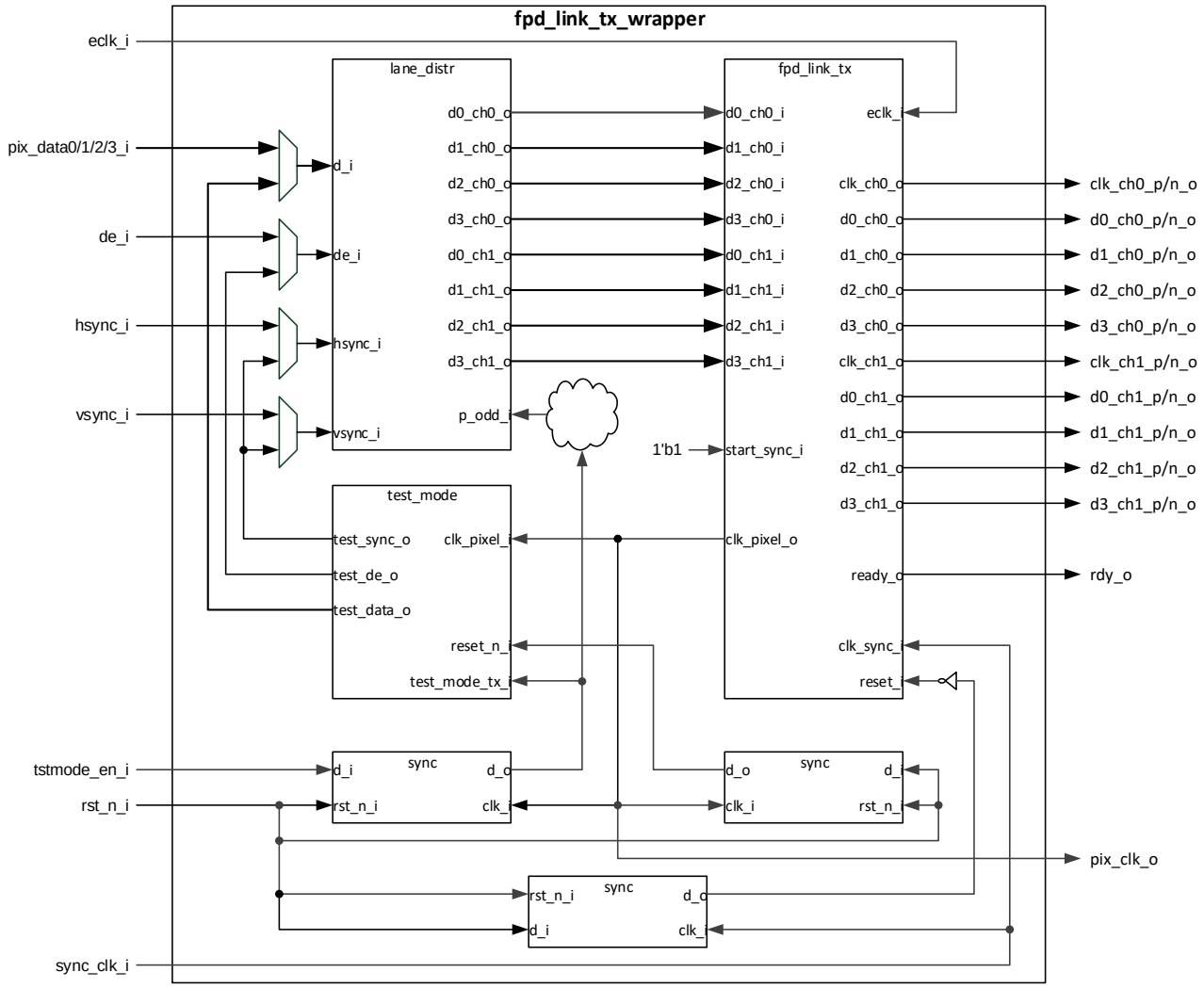

Figure 2.12. FPD-Link Tx Wrapper Block Diagram

The *fpd_link_tx_wrapper.v* module instantiates *fpd_link_tx*, *test_mode*, *lane_distr*, and *synchronizer* modules. The *fpd_link_tx* module is the core module that performs parallel to serial conversion. The *test_mode* is used for automatic generation of pixel data inside the system during debug mode. *lane_distr* is used for distributing pixel data to different LVDS lanes for OpenLDI unbalanced format. Synchronizers are two-level synchronizers used to sync the system reset and test mode enable signal into different clock domains before it is used in the system.

2.3.2. FPD-Link Tx Module

The *fpd_link_tx* module instantiates *GDDR_SYNC, lvds_oddr*, and *lvds_clk_tree* modules. The *GDDR_SYNC* module is required to initialize and synchronize DDR clock, and tolerate the large skew between stop and reset of the DDR components. The sub-block connection is based on CrossLink/CrossLinkPlus GDDR7:1 Tx Usage Model guide.

Parallel data are fed to the I/O logic DDR71 register in the *lvds_oddr* module. This module is used to convert the incoming parallel data into serial format. *lvds_clk_tree* is used to generate the clocks needed by the system. The *SCLK* is used as the output pixel clock of DDR71 IP.

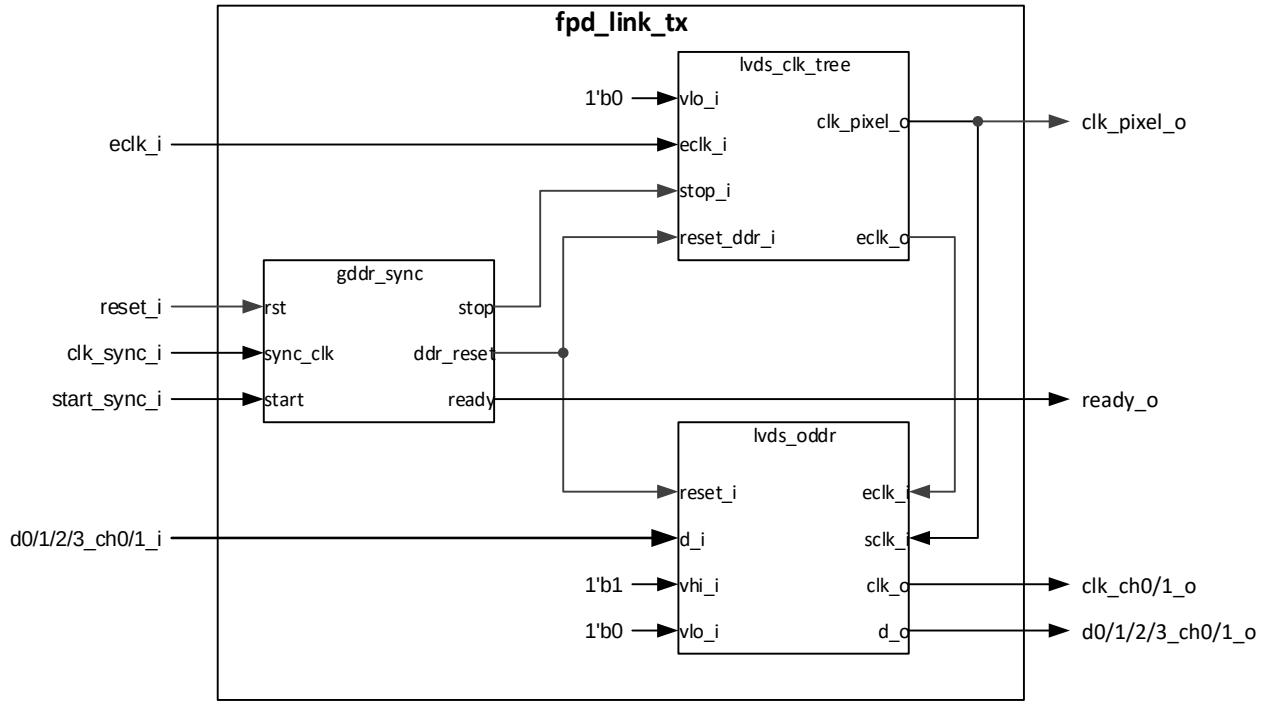

Figure 2.13. FPD-Link Tx Block Diagram

Table 2.4. FPD-Link Tx Pin List Summary

Port Name	Width	Dir	Type	Description
reset_i	1	I	LVCMOS	Asynchronous active high reset for GDDR_SYNC. 1 – System on reset
start_sync_i	1	I	LVCMOS	Active high start signal for GDDR_SYNC. 1 – Start clock synchronization.
clk_sync_i	1	I	LVCMOS	Clock for GDDR_SYNC. Must be slower than all the clocks in the system.
eclk_i	1	I	LVCMOS	Edge clock for LVDS side.
d0_ch0_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 0 channel 0
d1_ch0_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 1 channel 0
d2_ch0_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 2 channel 0
d3_ch0_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 3 channel 0
d0_ch1_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 0 channel 1
d1_ch1_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 1 channel 1
d2_ch1_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 2 channel 1
d3_ch1_i	TX_GEAR	I	LVCMOS	LVDS parallel data for lane 3 channel 1
clkwd_ch0_o	1	O	LVDS	LVDS output clock for channel 0
d3_ ch0_o	1	O	LVDS	LVDS output data for lane 3 channel 0
d2_ ch0_o	1	O	LVDS	LVDS output data for lane 2 channel 0

Port Name	Width	Dir	Type	Description
d1_ ch0_o	1	O	LVDS	LVDS output data for lane 1 channel 0
d0_ ch0_o	1	O	LVDS	LVDS output data for lane 0 channel 0
clkwd_ch1_o	1	O	LVDS	LVDS output clock for channel 1
d3_ch1_o	1	O	LVDS	LVDS output data for lane 3 channel 1
d2_ch1_o	1	O	LVDS	LVDS output data for lane 2 channel 1
d1_ch1_o	1	O	LVDS	LVDS output data for lane 1 channel 1
d0_ch1_o	1	O	LVDS	LVDS output data for lane 0 channel 1
clk_pixel_o	1	O	LVCMOS	Output pixel clock; Equivalent to SCLK
ready_o	1	O	LVCMOS	1 – Indicates that DDR synchronization (via GDDR_SYNC) is done 0 – Indicates that DDR synchronization (via GDDR_SYNC) is still not started or in progress

Table 2.5. FPD-Link Tx Parameter List

Parameters	Value	Description	Operation
NUM_TX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_TX_CH = <val>
TX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 7:1 Gearing 14 – 14:1 Gearing	parameter TX_GEAR = <val>
NUM_TX_LANE	3, 4	Specify number of data lanes per Tx link. 3 – RGB666 4 – RGB888	parameter NUM_TX_LANE = <val>

2.3.3. LVDS ODDR Group Module

The *LVDS DDR* group module is used to convert the incoming parallel data into serial format. Output bus width depends on the Number of Tx Channels and Lanes set through parameter.

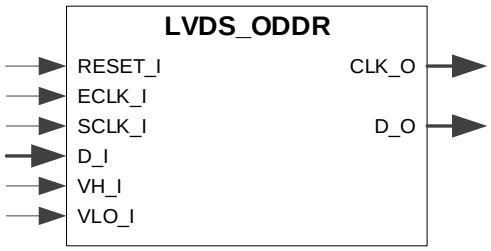

Figure 2.14. LVDS ODDR Group Block Diagram

Table 2.6. LVDS ODDR Group Module Pin List

Port Name	Width	Dir	Type	Description
reset_i	1	I	LVCMOS	Active high reset.
eclk_i	1	I	LVCMOS	Edge clock from ECLKSYNC.
sclk_i	1	I	LVCMOS	Primary clock from CLKDIV.
d_i	NUM_TX_CH*NUM_TX_LANE*TX_GEAR	I	LVCMOS	Parallel data input, LSB goes to lane 0 ch 0.
vhi_i	1	I	LVCMOS	VDD
vlo_i	1	I	LVCMOS	GND
clk_o	NUM_TX_CH	O	LVDS	LVDS output clock.
d_o	NUM_TX_CH*NUM_TX_LANE	O	LVDS	LVDS output data.

Table 2.7. LVDS ODDR Group Parameter List

Parameters	Value	Description	Operation
NUM_TX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_TX_CH = <val>
TX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 7:1 Gearing 14 – 14:1 Gearing	parameter TX_GEAR = <val>
NUM_TX_LANE	3, 4	Specify number of data lanes per Tx link. 3 – RGB666 4 – RGB888	parameter NUM_TX_LANE = <val>

2.3.4. LVDS Clock Tree Module

The *LVDS_CLK_TREE* is used to generate SCLK and ECLK for the LVDS interface.

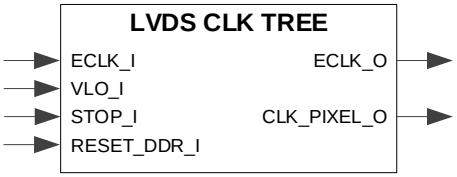

Figure 2.15. LVDS_CLK_TREE Block Diagram

Table 2.8. LVDS Clock Tree Pin List Summary

Port Name	Width	Dir	Type	Description
eclk_i	1	I	LVCMOS	Edge clock source.
stop_i	1	I	LVCMOS	Stop input for ECLKSYNC.
reset_ddr_i	1	I	LVCMOS	Asynchronous reset input for CLKDIV.
vlo_i	1	I	LVCMOS	Ground.
eclk_o	1	O	LVCMOS	Edge clock for LVDS DDR IPs.
clk_pixel_o	1	O	LVCMOS	Pixel clock (sclk for LVDS DDR IPs).

Table 2.9. LVDS Clock Tree Parameter List

Parameters	Value	Description	Operation
TX_GEAR	7, 14	Specify what DDR71 gearing are used 7 – 7:1 Gearing 14 – 14:1 Gearing	parameter TX_GEAR = <val>

2.3.5. GDDR SYNC Module

GDDR_SYNC is used for DDR clock initialization and synchronization. Start of DDR synchronization depends on either the assertion of *GPLL* lock if *GPLL* is used or release of reset if *GPLL* is not used. When started, *GDDR_SYNC* stops the *ECLKSYNC* block and resets the DDR and *CLKDIV* before starting *ECLKSYNC* again. This is to ensure that *ECLK* and *SCLK* are synchronized.

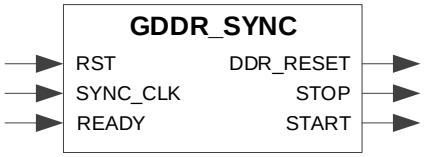

Figure 2.16. GDDR_SYNC Block Diagram

2.3.6. Lane Distribution Module

lane_distr module is used for distributing pixel data to different LVDS lanes for Openldi unbalanced format.

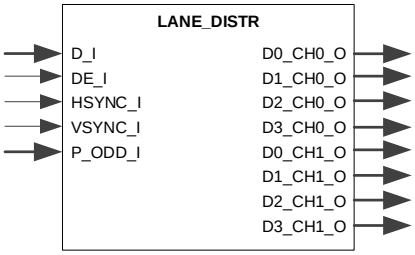

Figure 2.17. LANE_DISTR Block Diagram

Table 2.10. Lane Distribution Pin List Summary

Port Name	Width	Dir	Type	Description
d_i	NUM_PIX_TOTAL*PIX_WIDTH	I	LVCMOS	Input pixel data to be converted to Openldi format
de_i	1	I	LVCMOS	Data enable
hsync_i	1	I	LVCMOS	HSYNC
vsync_i	1	I	LVCMOS	VSYNC
p_odd_i	2	I	LVCMOS	Indicates which pixels are valid
d0_ch0_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 0 ch 0
d1_ch0_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 1 ch 0
d2_ch0_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 2 ch 0
d3_ch0_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 3 ch 0
d0_ch1_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 0 ch 1
d1_ch1_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 1 ch 1
d2_ch1_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 2 ch 1
d3_ch1_o	TX_GEAR	O	LVCMOS	Data input for DDR of lane 3 ch 1

Table 2.11. Lane Distribution Parameter List

Parameters	Value	Description	Operation
TX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 7:1 Gearing 14 – 14:1 Gearing	parameter TX_GEAR = <val>
NUM_PIX_TOTAL	1, 2, 4	Specify the total number of pixels data per pixel clock.	parameter NUM_PIX_TOTAL = <val>
PIX_WIDTH	18, 24	Specify the bus width of each pixel data. 18 – RGB666 24 – RGB888	parameter PIX_WIDTH = <val>
NUM_TX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_TX_CH = <val>
NUM_TX_LANE	3, 4	Specify number of data lanes per TX link. 3 – RGB666 4 – RGB888	parameter NUM_TX_LANE = <val>

2.3.7. CLKDIV

CLKDIV is a Lattice primitive. It is used as a clock divider. In FPD-Link Tx, CLKDIV is always set to either 3.5 (GEAR 7) or 7.0 (GEAR 14) ratio depending on the DDR GEAR selected.

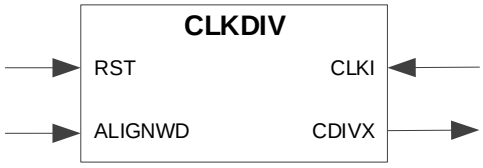

Figure 2.18. CLKDIV Block Diagram

2.3.8. ECLKSYNC

ECLKSYNC is a Lattice primitive. It is used to sync ECLK to the fabric clock. DDR only accepts ECLK from ECLKSYNC.

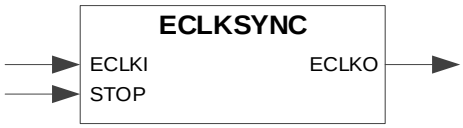

Figure 2.19. ECLKSYNC Block Diagram

2.3.9. Test Mode Tx Module

The *test_mode_tx* is used to automatically generate test data inside the system through *TEST_DATA* parameter and transmit them. Data comparison should be done outside the design. See Debug Features section for details on how to enable test mode.

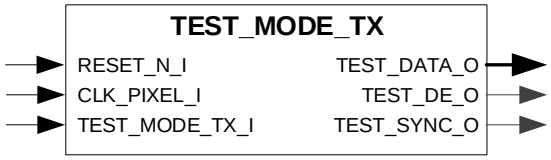

Figure 2.20. Test Mode Tx Block Diagram

Table 2.12. Test Mode Tx Pin List Summary

Port Name	Width	Dir	Type	Description
reset_n_i	1	I	LVCMOS	Asynchronous active low reset.
clk_pixel_i	1	I	LVCMOS	Input pixel clock.
test_mode_tx_i	1	I	LVCMOS	Enable or disable test mode. 1 - Enable test mode. 0 - Disable test mode.
test_data_o	NUM_PIX_TOTAL*PIX_WIDTH	O	LVCMOS	Output pixel data in open LDI format.
test_de_o	1	O	LVCMOS	Output data enable.
test_sync_o	1	O	LVCMOS	Output sync flags (hsync, vsync).

Table 2.13. Test Mode Tx Parameter List

Parameters	Value	Description	Operation
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For multiple pixels per pixel clock configuration, the same TEST DATA is used for all pixels. 24 – data width for RGB888 18 – data width for RGB666	parameter TEST_DATA = <val>
NUM_PIX_TOTAL	1,2,4	Specify the total number of pixel data per pixel clock.	parameter NUM_PIX_TOTAL = <val>
PIX_WIDTH	18, 24	Specify the bus width of each pixel data. 18 – RGB666 24 – RGB888	parameter PIX_WIDTH = <val>

2.3.10. Synchronizer Module

Synchronizer is a two-level synchronizer used to sync the input data into a different clock domain. In the design, this is used to synchronize the system reset into different clock domains before it is used in the system.

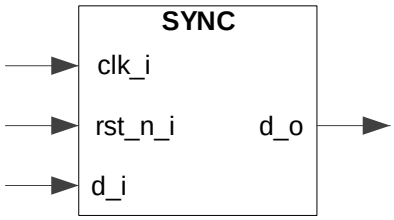

Figure 2.21. Synchronizer Block Diagram

Table 2.14. Synchronizer Pin List Summary

Port Name	Width	Dir	Type	Description
clk_i	1	I	LVCMOS	Input clock. Input data is synchronized with this clock.
rst_n_i	1	I	LVCMOS	Asynchronous active low reset.
d_i	BUS_WIDTH	I	LVCMOS	Input data.
d_o	BUS_WIDTH	O	LVCMOS	Output data.

Table 2.15. Synchronizer Parameter List

Parameters	Value	Description	Operation
BUS_WIDTH	<value>	Specify the bus width for both input and output data.	parameter BUS_WIDTH = <val>
RST_VAL	0,1	Specify the default value for the output data when in reset.	parameter RST_VAL = <val>

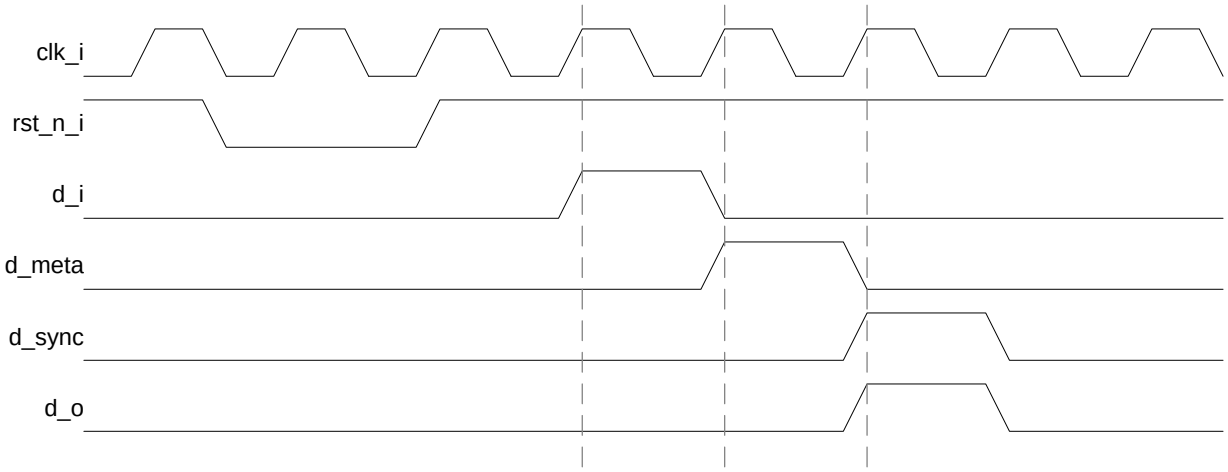

Figure 2.22. Synchronizer Timing Diagram

3. Compiler Directives and Parameter Settings

This section lists the compiler directives and parameters used to configure the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP.

3.1. Parameters Settings

Table 3.1 lists the parameters that can be set to configure the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP when the IP is not packaged.

Table 3.1. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Non-packaged Parameter Settings

Parameters	Value	Description	Operation
NUM_TX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_TX_CH = <val>
TX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 7:1 Gearing 14 – 14:1 Gearing	parameter TX_GEAR = <val>
DATA_TYPE	RGB888, RGB666	Specify the data type. RGB666 – automatically set NUM_TX_LANE to 3 RGB888 – automatically set NUM_TX_LANE to 4	parameter DATA_TYPE = “<val>”
NUM_TX_LANE	3, 4	Specify number of data lanes per TX link. 3 – RGB666 4 – RGB888	parameter NUM_TX_LANE = <val>
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For multiple pixels per pixel clock configuration, the same TEST DATA is used for all pixels. 24 – data width for RGB888 18 – data width for RGB666	parameter TEST_DATA = <val>

Table 3.2 lists the parameters used to generate the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP. All parameters are either set automatically or input in the user interface during the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP generation.

Table 3.2. OpenLDI/FPD-LINK/LVDS Transmitter User Interface IP Parameter Settings

Parameter	Attribute	Options	Description
Number of Tx Channels	User-Input	1, 2	Number of LVDS Tx channels.
Tx Interface	Read-Only	—	Tx is set to LVDS interface.
Number of Tx Lanes	Read-Only	3, 4	Generates LVDS I/O to either three or four depending on the data type per channel.
Tx Gear	User-Input	7, 14	Specify the Tx gearing. Gear 14 is recommended for Tx Line rate that is more than 900 Mb/s.
Tx Total Aggregate Bandwidth	Read-Only	<Value>	Tx total line rate. Automatically computed based on target Tx line rate.
Tx Line Rate (per lane)	User-Input	<Value>	Target line rate per lane. Maximum of 1200 Mb/s.
Pixel Clock Frequency	Read-Only	<Value>	Pixel clock. Automatically computed based on target Tx line rate.
LVDS Output Clock Frequency	Read-Only	<Value>	LVDS clock; Automatically computed based on target Tx line rate.
LVDS Edge Clock Frequency	Read-Only	<Value>	LVDS Edge clock; Automatically computed based on target Tx line rate.
Data Type	User-Input	RGB888, RGB666	Specify the data type.

Parameter	Attribute	Options	Description
Number of Input Pixels	Read-Only	1,2,4	Specify the number of input pixels per pixel clock. Computed based on selected Number of Tx Channels and Tx Gear.
Test Mode Data	User-Input	<Value>	Available only when Test Mode is enabled. 24 – Data width for RGB888 18 – Data width for RGB666

3.2. Compiler Directives

Aside from parameters, non-packaged OpenLDI/FPD-LINK/LVDS Transmitter Interface IP can also be configured through compiler directives.

Table 3.3. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Non-packaged Compiler Directives

Parameters	Value	Description	Operation
NUM_TX_CH_<#>	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	`define NUM_TX_CH_<val>
TX_GEAR_<#>	7, 14	Specify what DDR71 gearing is used. 7 – 7:1 Gearing 14 – 14:1 Gearing	`define TX_GEAR_ <val>
RGB<#>	888, 666	Specify the data type. RGB666 – Automatically set NUM_TX_LANE to 3 RGB888 – Automatically set NUM_TX_LANE to 4	`define RGB<val>
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For multiple pixels per pixel clock configuration, the same TEST DATA is used for all pixels. 24 – Data width for RGB888 18 – Data width for RGB666	`define TEST_DATA <val>
PIX<#>_PER_CLK	1,2,4	Specify the number of input pixels per pixel clock. 4 – Tx CH=2, Gear 14 2 – Tx CH=2, Gear 7 or Tx CH=1, Gear 14 1 – Tx CH=1, Gear 7	`define PIX<val>_PER_CLK
DEBUG_ON	—	Test mode can only be enabled when this is defined.	`define DEBUG_ON
MISC_ON	—	Define to access debug ports.	`define MISC_ON

4. Debug Features

4.1. Test Mode

This debug feature is used for automatic transmission of test data that is configured using *TEST_DATA* compiler directive. The test mode module drives channel 0 and channel 1 (if enabled) with predefined test data. Data comparison should be done outside the design. See Test Mode Tx Module section for the module description.

To enable test mode:

1. Configure *TEST_DATA*.
2. Define *DEBUG_ON* in the defines file.
3. Drive *tstmode_en_i* to 1'b1.
4. Perform reset sequence specified in Clock, Reset and Initialization section.
5. Monitor data transmission over LVDS lanes (test data is sent continuously with DE = high, VSYNC = HSYNC = RSVD = 0)

   Note that TEST_DATA is equivalent to {R[7:0],G[7:0],B[7:0]}.

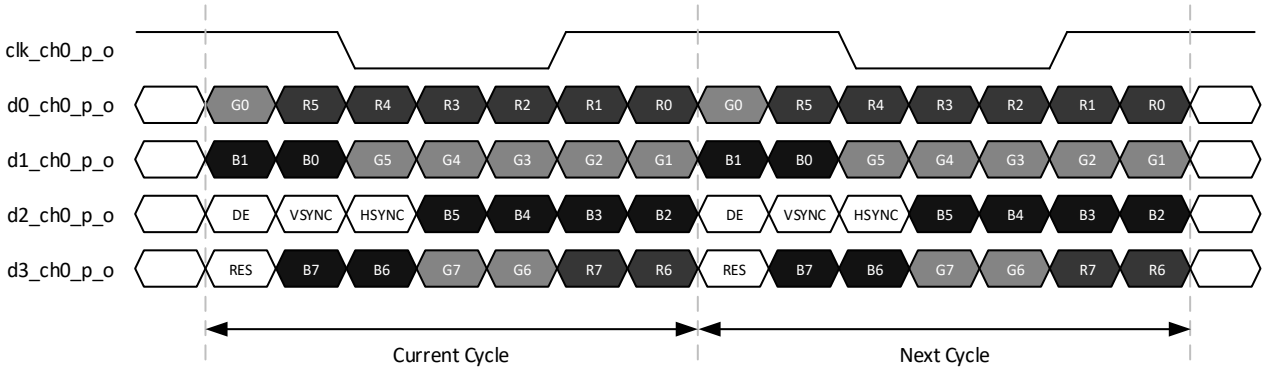

Figure 4.1. Sample Single LVDS Output (RGB888)

5. IP Generation and Evaluation

This section provides information on how to generate the Lattice OpenLDI/FPD-LINK/LVDS Transmitter Interface IP code using Lattice Diamond Clarity Designer and how to run simulation, synthesis and hardware evaluation.

5.1. Licensing the IP

The OpenLDI/FPD-LINK/LVDS Transmitter Interface IP is available free of charge, but an IP-specific license is required to enable full, unrestricted use of the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP in a complete, top-level design.

Request your license by going to the link http://www.latticesemi.com/en/Support/Licensing and request the free Lattice Diamond license. In this form, select the desired CrossLink and/or CrossLinkPlus IP for your design.

You may download and generate the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP and fully evaluate the IP through functional simulation and implementation (synthesis, map, place and route) without an IP license. The OpenLDI/FPD-LINK/LVDS Transmitter Interface IP also supports Lattice IP hardware evaluation capability. See the Hardware Evaluation section for further details.

HOWEVER, THE IP LICENSE IS REQUIRED TO ENABLE TIMING SIMULATION, TO OPEN THE DESIGN IN DIAMOND EPIC TOOL, OR TO GENERATE BITSTREAMS THAT DO NOT INCLUDE THE HARDWARE EVALUATION TIMEOUT LIMITATION.

5.2. Getting Started

The OpenLDI/FPD-LINK/LVDS Transmitter Interface IP is available for download from the Lattice IP Server using the Clarity Designer tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP is installed, the IP is available in the Clarity Designer user interface as shown in Figure 5.1.

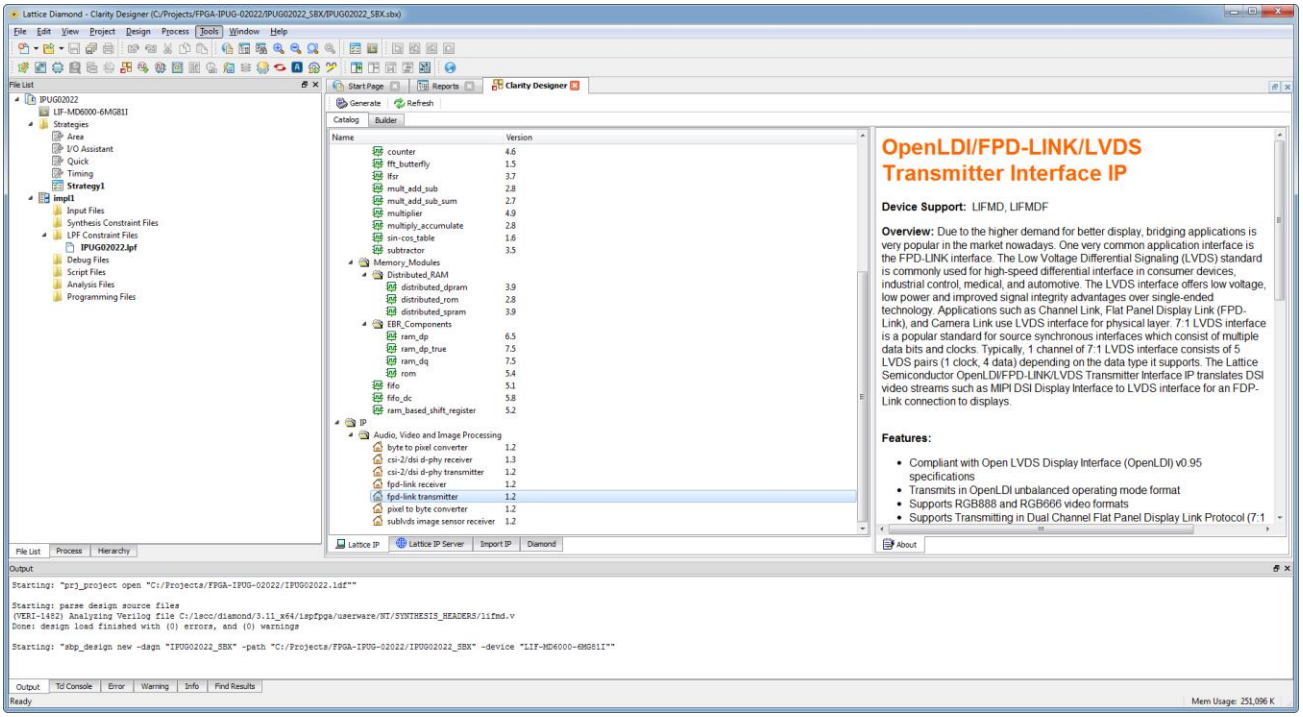

Figure 5.1. Clarity Designer Window

5.3. Generating IP in Clarity Designer

The Clarity Designer tool is used to customize modules and IPs and place them into the device architecture. Besides configuration and generation of modules and IPs, Clarity Designer can also create a top module template in which all generated modules and IPs are instantiated.

The procedure for generating OpenLDI/FPD-LINK/LVDS Transmitter Interface IP in Clarity Designer is described below. Clarity Designer can be started from the Diamond design environment.

To start Clarity Designer:

1. Create a new Diamond project for CrossLink or CrossLinkPlus family devices.
2. From the Lattice Diamond main window, choose **Tools > Clarity Designer**, or click  in Diamond toolbox. The Clarity Designer project dialog box is displayed.
3. Select and/or fill out the following items as shown in Figure 5.2.
   - **Create new Clarity design** – Select this to create a new Clarity Design project directory in which the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP is generated.
   - **Design Location –** Clarity Design project directory path.
   - **Design Name –** Clarity Design project name.
   - **HDL Output –** Hardware Description Language Output Format (Verilog HDL).

   The Clarity Designer project dialog box also allows you to open an existing Clarity Designer project by selecting the following:
   - **Open Clarity design** – Open an existing Clarity Design project.
   - **Design File** – Name of existing Clarity Design project file with .sbx extension.
4. Click the **Create** button. A new Clarity Designer project is created.

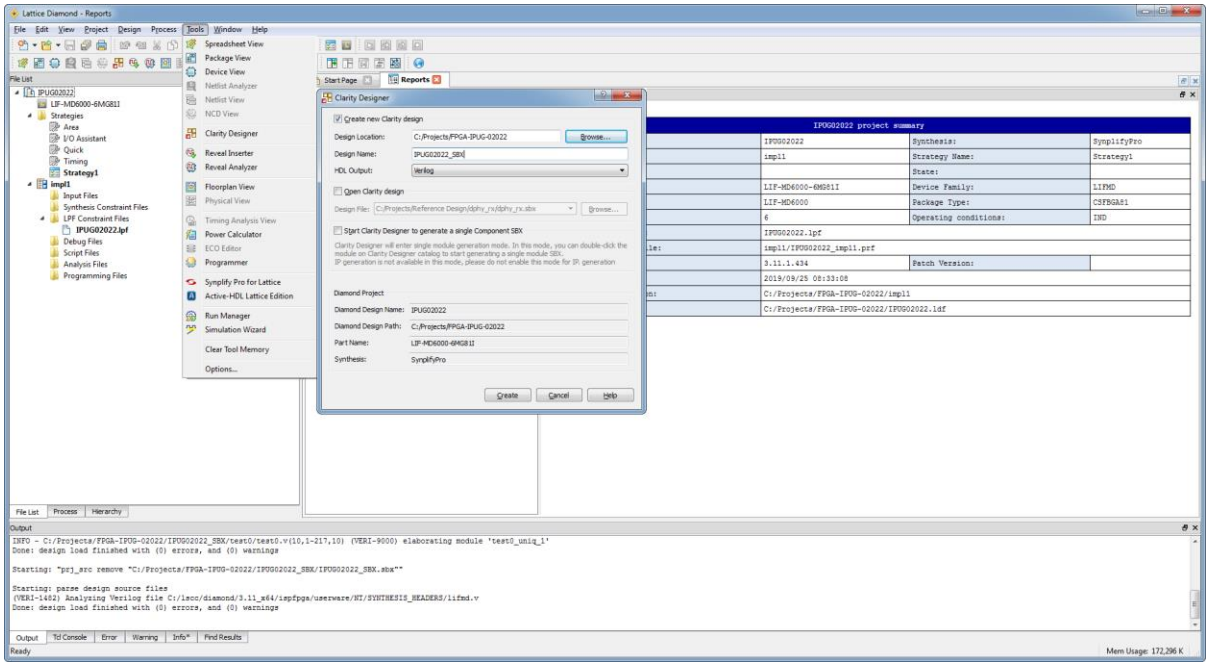

Figure 5.2. Starting Clarity Designer from Lattice Diamond Design Environment

To configure OpenLDI/FPD-LINK/LVDS Transmitter Interface IP in Clarity Designer:

1. Double-click **FPD-LINK TRANSMITTER** in the IP list of the System Catalog view. The **fpd-link transmitter** dialog box is displayed as shown in Figure 5.3.

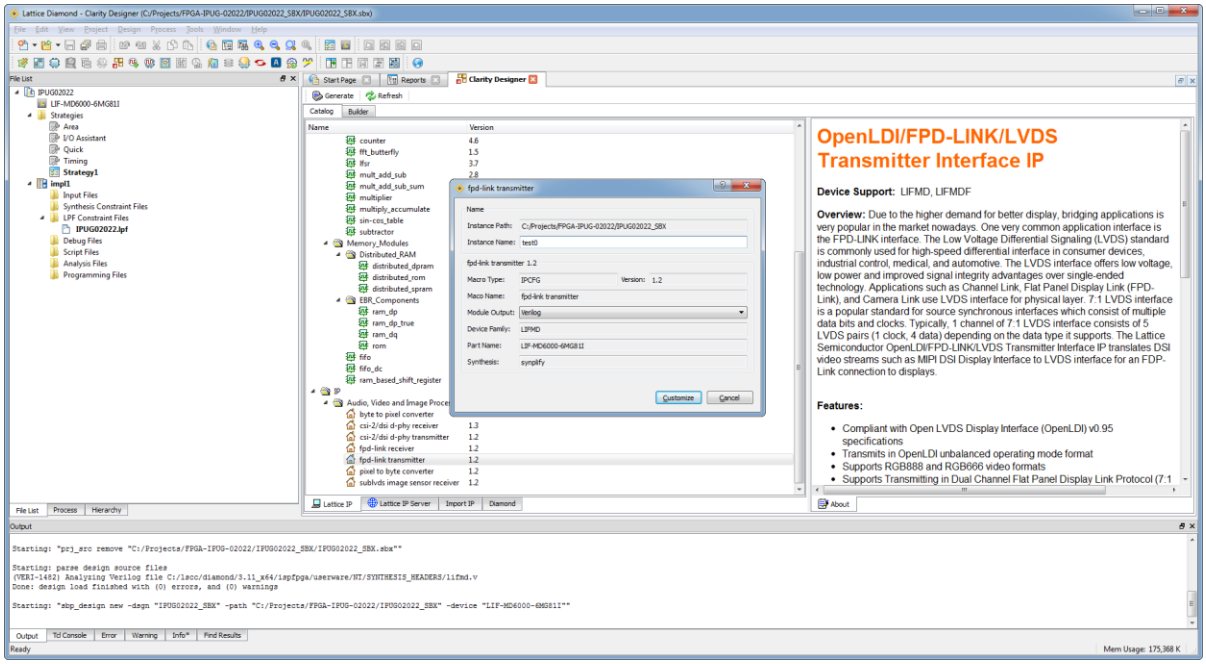

Figure 5.3. Configuring OpenLDI/FPD-LINK/LVDS Transmitter Interface IP in Clarity Designer

2. Enter the **Instance Name**.
3. Click the **Customize** button. An IP configuration interface is displayed as shown in Figure 5.4. From this dialog box, you can select the IP configuration specific to your application.
4. Input valid values in the required fields in the **Configuration** tab.
5. After selecting the required parameters, click the **Configure** button.
6. Click **Close**.
7. Click 

in the toolbox. Clarity Designer generates all the IPs and modules, and creates a top module to wrap them.

For detailed instructions on how to use the Clarity Designer, refer to the Lattice Diamond software user guide.

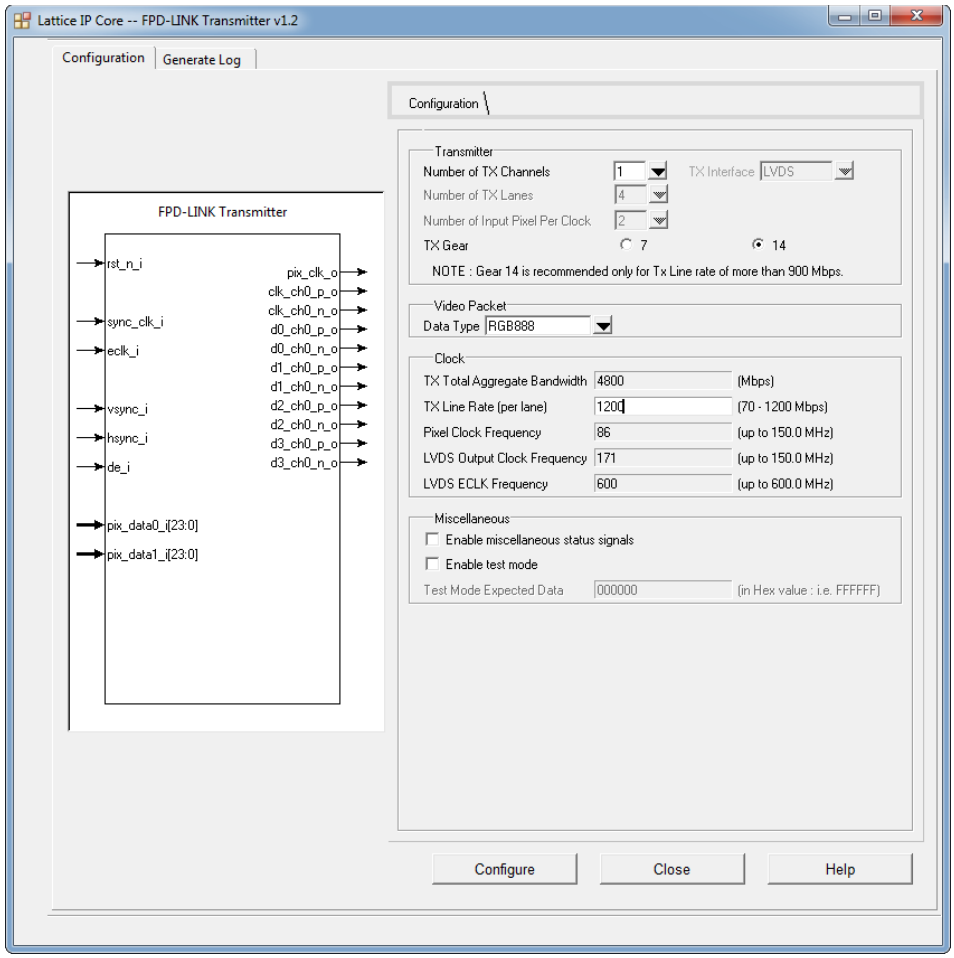

Figure 5.4. Configuration Tab in IP User Interface

5.4. Generated IP Directory Structure and Files

Figure 5.5 shows the directory structure of generated IP and supporting files.

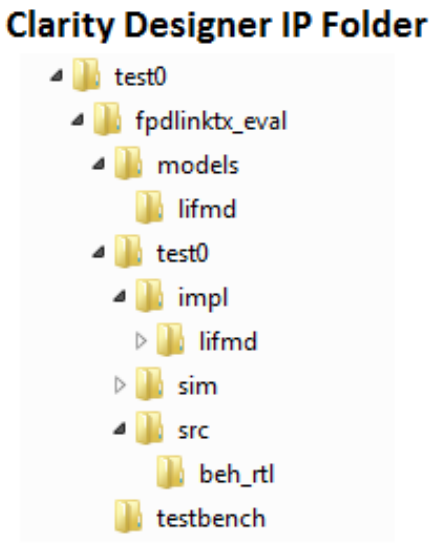

Figure 5.5. OpenLDI/FPD-LINK/LVDS Transmitter Interface IP Directory Structure

The design flow for the IP created with Clarity Designer uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module and protected model are customized when you configure the IP and are created automatically when the IP is generated.

Table 5.1 provides a list of key files and directories created by Clarity Designer and how they are used. The post-synthesized module (NGO), the protected simulation model, and all other files are also generated based on your configuration and provided as examples to use or evaluate the IP.

Table 5.1. Files Generated in Clarity Designer

File	Description
<instance_name>.v	Verilog top-level module of OpenLDI/FPD-LINK/LVDS Transmitter Interface IP used for both synthesis and simulation.
<instance_name>_.v	Verilog submodules for simulation. Files that do not have equivalent black-box modules are also used for synthesis.
*<instance_name>_*_beh.v	Protected Verilog models for simulation.
<instance_name>_ _bb.v	Verilog black-box modules for synthesis.
<instance_name>_.ngo	User interface configured and synthesized modules for synthesis.
*<instance_name>*_params.*v*	Verilog parameters file, which contains required compiler directives to successfully configure IP during synthesis and simulation.
<instance_name>.lpc	Lattice Parameters Configuration file. This file records all the IP configuration options set through Clarity Designer. It is used by IP generation script to generate configuration-specific IP. It is also used to reload parameter settings in the IP user interface in Clarity Designer when it is being reconfigured.
*<instance_name>*_inst.v/vhd	Template for instantiating the generated soft IP top-level in another user-created top module.

Aside from the files listed in the tables, most of the files required to evaluate the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP are available under the directory *<fpdlinktx_eval>*, including the simulation model. Lattice Diamond project files are also included under the folder at *<fpdlinktx_eval>**<instance_name>**impl**<device_family>* \ *<synthesis_tool>*\, where *<device_family>* can either be **lifmd** for CrossLink or **lifmdf** for CrossLinkPlus devices.

The *<instance_name>* folder (test0 in Figure 5.5) contains files/folders with content specific to the *<instance_name>* configuration. This directory is created by Clarity Designer each time the IP is generated and regenerated with the same file name. A separate *<instance_name>* directory is generated for IPs with different names, such as *<my_IP_0>*,*<my_IP_1>*, and others.

The folder*<instance_name>*, the *fpdlinktx_eval*, and subdirectories provide files supporting OpenLDI/FPD-LINK/LVDS Transmitter Interface IP evaluation that includes files/folders with content that is constant for all configurations of the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP. The *fpdlinktx_eval* directory is created by Clarity Designer the first time the IP is generated, when multiple OpenLDI/FPD-LINK/LVDS Transmitter Interface IPs are generated in the same root directory and updated each time the IP is regenerated.

You can use the prebuilt Diamond projects provided at *<project_root>**fpdlinktx_eval**<instance_name>**impl**<device_family>**<synthesis_tool>*\ to evaluate the implementation (synthesis, map, place and route) of the IP in Lattice Diamond tool. The *src* directory contains the behavioral models of the black-boxed modules and the *models* directory provides library elements.

5.5. Running Functional Simulation

To run simulations using Active-HDL:

1. Under the **Tools** menu, select **Active-HDL**.
2. Modify the **tb_params.v** file located in **\<project_dir>\fpdlinktx_eval\testbench**.
   Update testbench parameters to customize data size and/or other settings. For example,
   Modify WC value as needed in your simulation.
   // WC - Number of bytes sent per line
   `define WC 4320
   See additional information about testbench parameters in Table 5.2.
3. In **Active-HDL** window, under the **Tools** tab, select **Execute Macro**.
4. Select the **.do** file **\<project_dir>\fpdlinktx_eval\<instance_name>\sim\aldec*run.do**.
5. Click **OK**.
6. Wait for the simulation to finish.

Table 5.2 lists the testbench directives which can be modified by setting the define in the *tb_params.v* file.

Table 5.2. Testbench Compiler Directives

Compiler Directive	Description
NUM_FRAMES	Sets the number of video frames.
TOTAL_LINES	Sets the number of lines per frame.
HFRONT	Number of cycles before HSYNC signal asserts (Horizontal Front Blanking).
HPULSE	Number of cycles HSYNC signal asserts.
HBACK	Number of cycles after HSYNC signal asserts (Horizontal Rear Blanking).
VFRONT	Number of cycles before VSYNC signal asserts (Vertical Front Blanking).
VPULSE	Number of cycles VSYNC signal asserts.
VBACK	Number of cycles after VSYNC signal asserts (Vertical Rear Blanking).
INIT_DRIVE_DELAY	If miscellaneous signals are disabled, driving delay should be set before test bench starts sending data to IP.
WC	Number of bytes sent per line.

5.6. Simulation Strategies

This section describes the simulation environment which demonstrates basic FPD-Link Tx functionality. Figure 5.6 shows the block diagram of simulation environment.

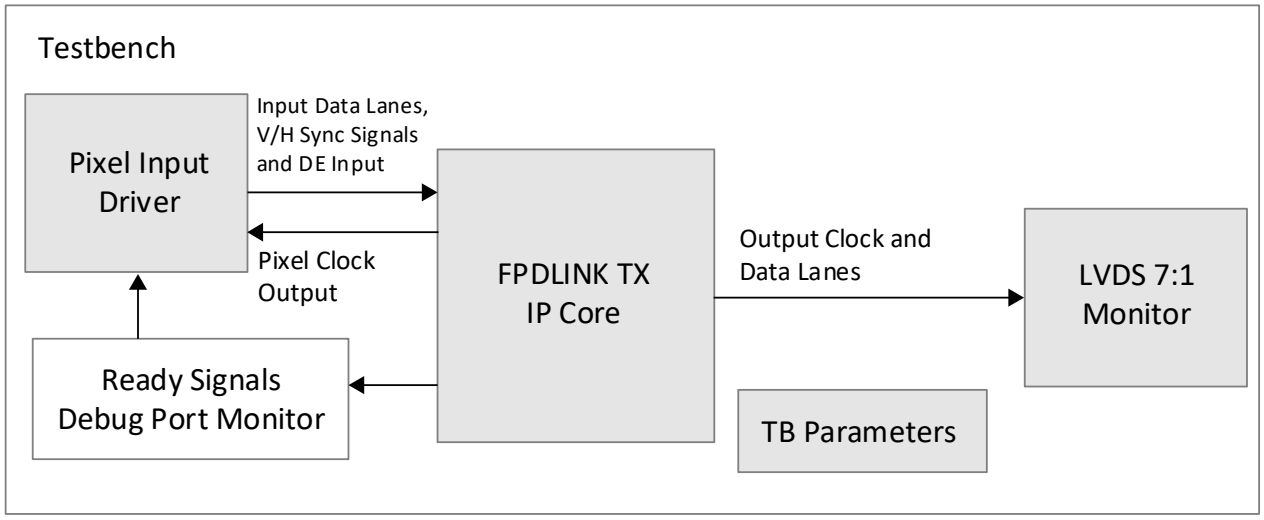

Figure 5.6. Simulation Environment Block Diagram

5.7. Simulation Environment

The simulation environment is made up of a Pixel domain input driver instance connected to the input of FPD-Link Tx IP core instance in the testbench. The Pixel domain input driver is configured based on the FPD-Link Tx IP configurations and testbench parameters. The testbench can be configured as one or two Tx channels. If the miscellaneous signal *rdy_o* debug port is included in the IP core, the testbench would monitor assertion of this signal before sending the DSI video data. If miscellaneous signal is disabled, you should set initial driving delay before the testbench starts sending out data to the IP core.

Input pixel data and output LVDS 7:1 data are logged into *input_data*.log* and *pixel_out*.log* files, respectively. Tester can compare these files to check if data is being transmitted properly. See the Running Functional Simulation section for details on how to set testbench parameters.

Figure 5.7 shows an example simulation of two Tx channels configuration.

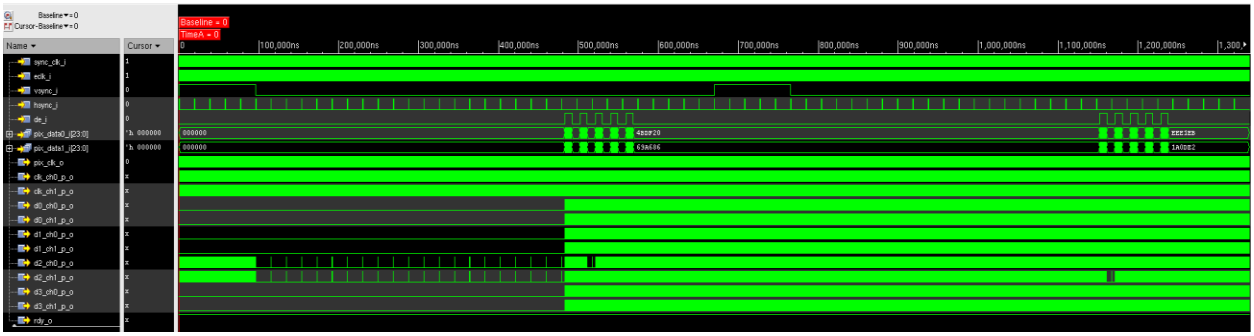

Figure 5.7. Two Tx Channels Configuration

Figure 5.8 shows an example simulation of one Tx channel configuration.

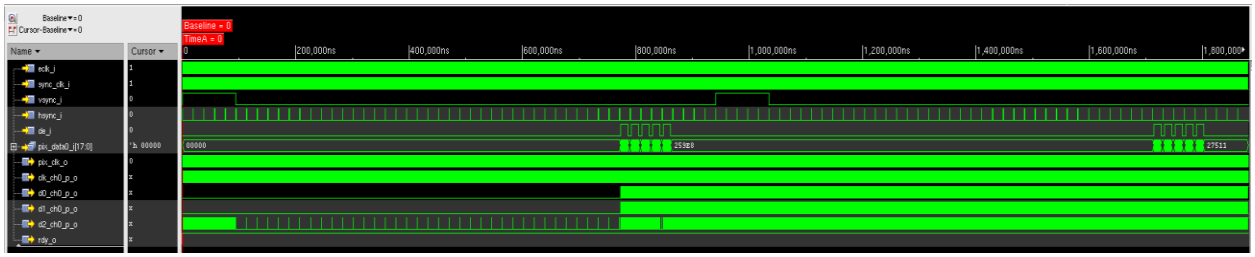

Figure 5.8. One Tx Channel Configuration

5.8. Instantiating the IP

The core modules of the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP are synthesized and provided in NGO format with black box Verilog source files for synthesis. A Verilog source file named *<instance_name>_fpd_link_tx.v* instantiates the black box of core modules. The top-level file *<instance_name>.v* instantiates *<instance_name>_fpd_link_tx.v and <instance_name>_lane_distr.v*.

A Verilog instance template *<instance_name>_inst.v* or VHDL instance template *<instance_name>_inst.vhd* is also provided as a guide, if the design is to be included in another top-level module.

You do not need to instantiate the IP instances one by one manually. The top-level file and other Verilog source files are provided in *\<project_dir>*. These files are refreshed each time the IP is regenerated.

5.9. Synthesizing and Implementing the IP

In Clarity Designer, the Clarity Designer project file (.sbx) is added to Lattice Diamond as a source file after all IPs are generated. Note that default Diamond strategy (.sty) and default Diamond preference file (.lpf) are used. When using the .sbx approach, import the recommended strategy from *\<project_dir>\fpdlinktx_eval\<instance_name>\impl\<device_family>\lse* or *\<project_dir>\fpdlinktx_eval\<instance_name>\impl\<device_family>\synplify* directories and set them as active strategy. A sample preference file (.lpf) is also included in the directory. All required files are invoked automatically. You can directly synthesize, map and place/par the design in the Diamond design environment after the cores are generated.

Push-button implementation of this top-level design with either Synplify or Lattice Synthesis Engine is supported via the Diamond project files *<instance_name>_top.ldf* which is located in *\<project_dir>\fpdlinktx_eval\<instance_name>\impl\<device_family>\<synthesis_tool>* directory.

To use the pre-built Diamond project files:

1. Choose **File > Open > Project**.
2. In the **Open Project** dialog box browse to **\<project_dir>\fpdlinktx_eval\<instance_name>\impl\<device_family>\<synthesis_tool>**.
3. Select and open **<instance_name>_top.ldf**. At this point, all of the files needed to support top-level synthesis and implementation are imported to the project.
4. Select the **Process** tab in the left-hand user interface window.
5. Implement the complete design via the standard Diamond user interface flow.

5.10. Hardware Evaluation

The OpenLDI/FPD-LINK/LVDS Transmitter Interface IP supports Lattice IP hardware evaluation capability. You can create versions of the IP that operate in hardware for a limited period of time without requiring the request of an IP license. It may also be used to evaluate the IP in hardware in user-defined designs.

5.10.1. Enabling Hardware Evaluation in Diamond

Choose **Project** > **Active Strategy** > **Translate Design Settings**. The hardware evaluation capability may be enabled or disabled in the **Strategy** dialog box. It is enabled by default.

5.11. Updating/Regenerating the IP

The Clarity Designer interface allows you to update the local IPs from the Lattice IP server. The updated IP can be used to regenerate the IP in the design. To change the parameters of the IP used in the design, the IP must also be regenerated.

5.11.1. Regenerating an IP in Clarity Designer

To regenerate IP in Clarity Designer:

1. In the **Builder** tab, right-click the IP instance to be regenerated and select **Config** from the menu as shown in Figure 5.9.

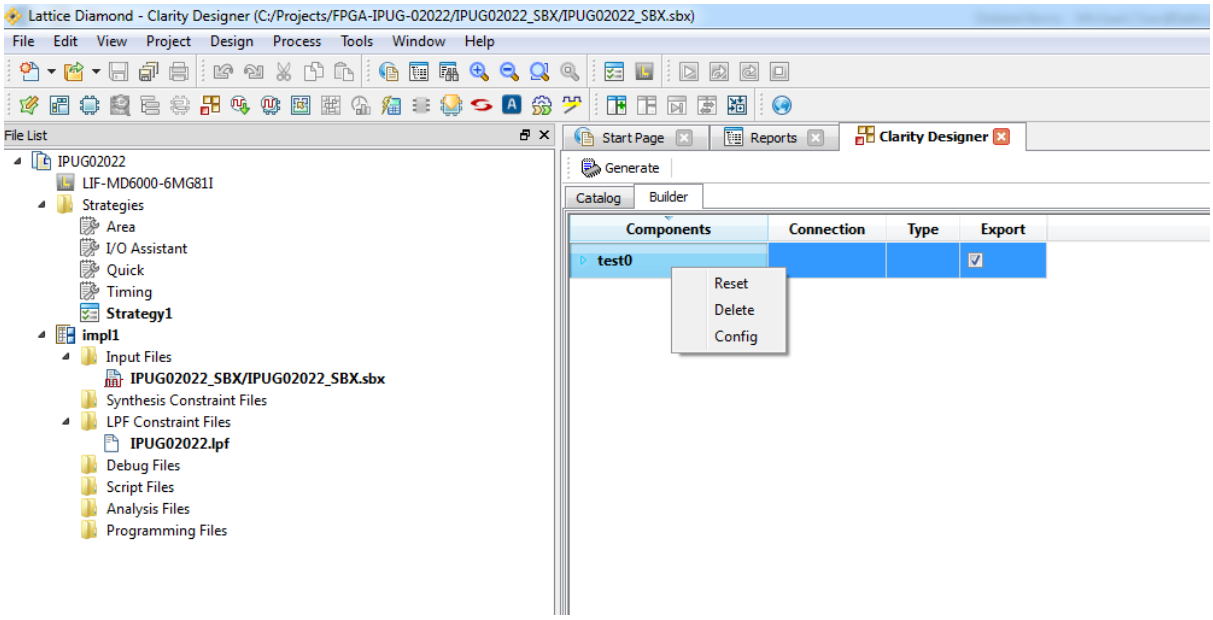

Figure 5.9. IP Regeneration in Clarity Designer

2. The IP Configuration user interface is displayed. Change the parameters as required and click the **Configure** button.
3. Click 

in the toolbox. Clarity Designer regenerates all the instances which are reconfigured.

References

For more information about CrossLink and CrossLinkPlus devices, refer to CrossLink Family Data Sheet (FPGA-DS-02007) and CrossLinkPlus Family Data Sheet (FPGA-DS-02054).

Software documentation:

- Clarity Designer User Manual
- Lattice Diamond User Guide

Technical Support Assistance

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

Appendix A. Resource Utilization

Table A.1 lists resource utilization information for Lattice CrossLink FPGA using the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP.

Clarity Designer is the Lattice IP configuration utility, and is included as a standard feature of Lattice Diamond tool. For details about the usage of Clarity Designer, refer to the Clarity Designer and Diamond help system.

For more information on the Diamond design tools, visit the Lattice web site at www.latticesemi.com/Products/DesignSoftwareAndIP.

Table A.1. Resource Utilization[1]

IP User-Configurable Parameters	Slices	LUTs	Registers	sysMEM EBRs	Target f_{MAX} (MHz)[2]
1x7 Tx, RGB888	24	39	18	0	150
2x7 Tx, RGB888	26	40	18	0	150
1x14 Tx, RGB888	27	40	18	0	150
2x14 Tx, RGB888	24	38	18	0	150

Notes:

1. Performance and utilization data target an LIF-MD6000-6MG81I device using Lattice Diamond 3.9 and Lattice Synthesis Engine software. Performance may vary when using a different software version or targeting a different device density or speed grade within the CrossLink family. This does not show all possible configurations of the OpenLDI/FPD-LINK/LVDS Transmitter Interface IP.
2. The f_{MAX} values are based on internal pixel clock.

Appendix B. What is Not Supported

The IP does not support configuration through registers.

The IP has the following limitation:

Odd multiple number of pixels is not supported when Tx Gear 14 is used.

Revision History

Revision 1.2, IP Version 1.3, October 2019

Section	Change Summary
Disclaimer	Newly added section.
All	• Added CrossLinkPlus device support. • Minor adjustments in style and formatting.

Revision 1.1, IP Version 1.0, April 2019

Section	Change Summary
Introduction	Specified that this user guide can be used for IP design versions 1.x.
IP Generation and Evaluation	In Licensing the IP, modified the instructions for requesting free license.
Revision History	Updated revision history table to new template.
All	Minor adjustments in style and formatting.

Revision 1.0, IP Version 1.0, July 2017

Section	Change Summary
All	Initial release.

www.latticesemi.com