OpenLDI/FPD-LINK/LVDS Receiver Interface IP

User Guide

FPGA-IPUG-02021-1.4

April 2024

Disclaimers

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

Contents

Figures

Tables

1. Introduction

The Lattice Semiconductor OpenLDI/FPD-LINK/LVDS Receiver Interface IP translates video streams from a processor with an OpenLDI/FDP-Link/LVDS interface connection to pixel clock domain. This can be used to connect with other application interfaces, such as the Mobile Industry Processor Interface (MIPI®) Display Serial Interface (DSI) by integrating it with Pixel-to-Byte Converter and CSI-2/DSI D-PHY Transmitter Submodule IPs.

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, 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 Receiver Interface IP design version 1.3.

Figure 1.1. Sample OpenLDI/FPD-LINK/LVDS Receiver Interfaced to MIPI DSI System Diagram

1.1. Quick Facts

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

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

		OpenLDI/FPD-LINK/LVDS Receiver Interface IP Configuration			
		1x7 Rx, RGB888 Configuration	**2x7 Rx, RGB888 Configuration**	**1x14 Rx, RGB888 Configuration**	**2x14 Rx, RGB888 Configuration**
IP Requirements	FPGA Families Supported	CrossLink/CrossLinkPlus			
Resource Utilization	Targeted Device	LIF-MD6000-6MG81I			
	Data Path Width	28-bit input, parallel output	56-bit input (28-bit per Rx channel), parallel output	28-bit input, parallel output	56-bit input (28-bit per Rx channel), parallel output
	LUTs	233	237	318	390
	sysMEM™ EBRs	0	0	0	0
	Registers	119	143	176	248
	HW MIPI Block	0	0	0	0
	Programmable I/O	34	62	58	110
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 Receiver 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 receiving in Dual Channel Flat Panel Display Link Protocol (7:1 LVDS)
- Supports 3-4 LVDS data lanes per channel
- Supports 1, 2, or 4 output pixels per pixel clock
- Supports interfacing up to 7.2 Gb/s

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

IP Configuration	Options
Number of Channels	1, 2
Data Type	RGB666, RGB888
Number of Rx Lanes per Channel*	3, 4
DDR Gear	7, 14

***Note**: The Number of Rx Lanes per Channel is automatically set depending on the Data Type.

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.

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 Receiver Interface IP converts a standard OpenLDI serial video interface into pixel clock domain. The input interface for the design consists of a data bus, vertical and horizontal sync flags, a data enable, and a clock in OpenLDI (LVDS 7:1) interface format. Output interface consists of the RGB control signals, pixel clock, up to four pixel data per pixel clock, and debug signals.

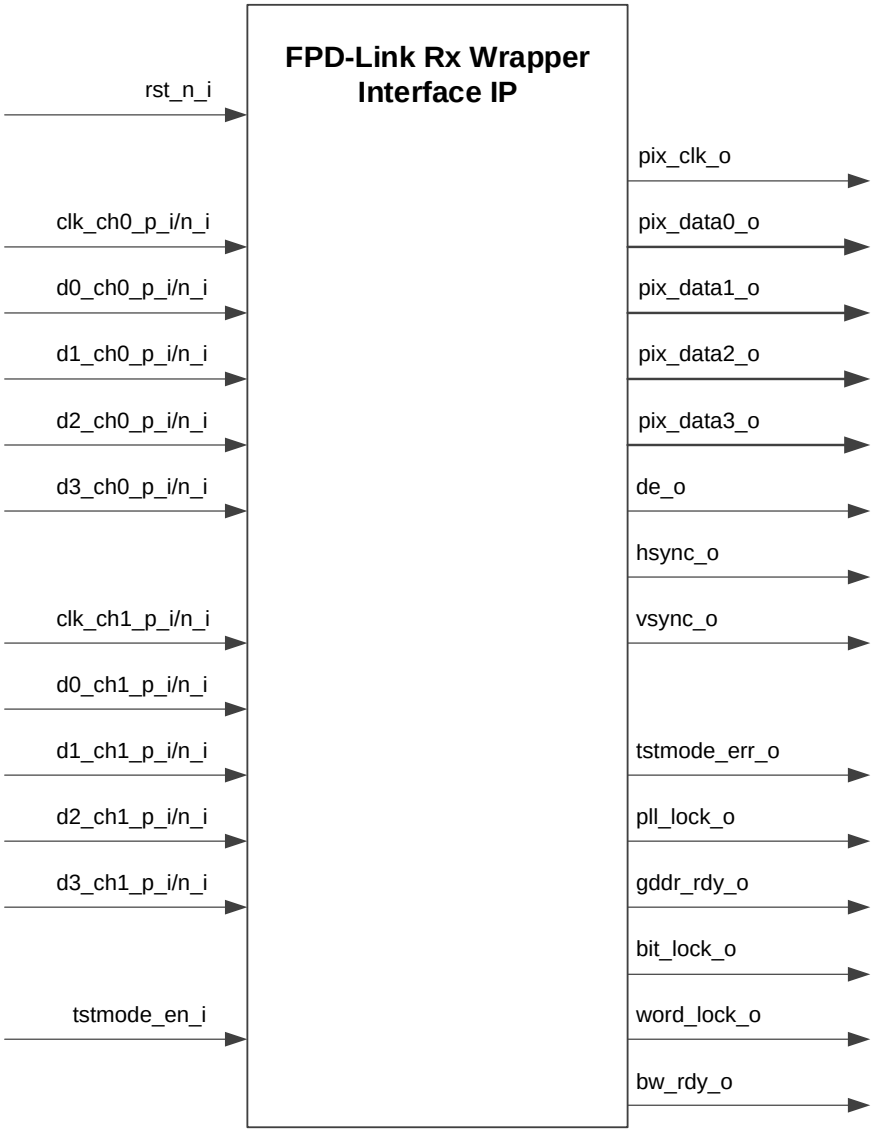

Figure 2.1. Top-level Block Diagram

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

Pin Name	Direction	Function Description
Reset		
rst_n_i	I	Asynchronous active low system reset. 0 – System on reset
OpenLDI/FPD-LINK Interface		
clk_ch0_p_i	I	Positive LVDS Input clock to LVDS 7:1 Rx Wrapper channel 0. For dual LVDS channel configuration, only channel 0 clock is used as reference clock for the system.

Pin Name	Direction	Function Description
clk_ch0_n_i	I	Negative LVDS Input clock to LVDS 7:1 Rx Wrapper channel 0, complement of clk_ch0_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool. For dual LVDS channel configuration, only channel 0 clock is used as reference clock for the system.
clk_ch1_p_i	I	Positive LVDS Input clock to LVDS 7:1 Rx Wrapper channel 1. For dual LVDS channel configuration, only channel 0 clock is used as reference clock for the system, channel 1 clock is unused.
clk_ch1_n_i	I	Negative LVDS Input clock to LVDS 7:1 Rx Wrapper channel 1, complement of clk_ch1_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool. For dual LVDS channel configuration, only channel 0 clock is used as reference clock for the system, channel 1 clock is unused.
d0_ch0_p_i	I	Positive LVDS Input data lane 0 to LVDS 7:1 Rx Wrapper channel 0.
d0_ch0_n_i	I	Negative LVDS Input data lane 0 to LVDS 7:1 Rx Wrapper channel 0, complement of d0_ch0_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d1_ch0_p_i	I	Positive LVDS Input data lane 1 to LVDS 7:1 Rx Wrapper channel 0.
d1_ch0_n_i	I	Negative LVDS Input data lane 1 to LVDS 7:1 Rx Wrapper channel 0, complement of d1_ch0_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d2_ch0_p_i	I	Positive LVDS Input data lane 2 to LVDS 7:1 Rx Wrapper channel 0.
d2_ch0_n_i	I	Negative LVDS Input data lane 2 to LVDS 7:1 Rx Wrapper channel 0, complement of d2_ch0_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d3_ch0_p_i[1]	I	Positive LVDS Input data lane 3 to LVDS 7:1 Rx Wrapper channel 0
d3_ch0_n_i[1]	I	Negative LVDS Input data lane 3 to LVDS 7:1 Rx Wrapper channel 0, complement of d3_ch0_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d0_ch1_p_i[2]	I	Positive LVDS Input data lane 0 to LVDS 7:1 Rx Wrapper channel 1.
d0_ch1_n_i[2]	I	Negative LVDS Input data lane 0 to LVDS 7:1 Rx Wrapper channel 1, complement of d0_ch1_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d1_ch1_p_i[2]	I	Positive LVDS Input data lane 1 to LVDS 7:1 Rx Wrapper channel 1.
d1_ch1_n_i[2]	I	Negative LVDS Input data lane 1 to LVDS 7:1 Rx Wrapper channel 1, complement of d1_ch1_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d2_ch1_p_i[2]	I	Positive LVDS Input data lane 2 to LVDS 7:1 Rx Wrapper channel 1.
d2_ch1_n_i[2]	I	Negative LVDS Input data lane 2 to LVDS 7:1 Rx Wrapper channel 1, complement of d2_ch1_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
d3_ch1_p_i[1,2]	I	Positive LVDS Input data lane 3 to LVDS 7:1 Rx Wrapper channel 1.
d3_ch1_n_i[1,2]	I	Negative LVDS Input data lane 3 to LVDS 7:1 Rx Wrapper channel 1, complement of d3_ch1_p_i. Not available in netlist, because this is automatically set to complement by the synthesis tool.
Pixel Domain Interface		
pix_clk_o	O	Output pixel clock.
pix_data0_o	O	Output pixel data 0. Bus width depends on the data type selected. 24 - RGB888 18 - RGB666
pix_data1_o[3,6]	O	Output pixel data 1. Bus width depends on the data type selected. 24 - RGB888 18 - RGB666
pix_data2_o[3]	O	Output pixel data 2. Bus width depends on the data type selected. 24 - RGB888 18 - RGB666
pix_data3_o[3,6]	O	Output pixel data 3. Bus width depends on the data type selected. 24 - RGB888 18 - RGB666
de_o	O	Output data enable for parallel interface.
hsync_o	O	Output horizontal sync for parallel interface.

Pin Name	Direction	Function Description
vsync_o	O	Output vertical sync for parallel interface.
Miscellaneous		
tstmode_en_i[4]	I	Enable or disable test mode. 1 - Enable test mode. 0 - Disable test mode.
tstmode_err_o[4]	O	1 - Indicates that compare mismatch is encountered when test mode is enabled. 0 - No error is encountered. Automatically set to 0 when test mode is disabled.
pll_lock_o[5]	O	Active high GPLL lock signal.
gddr_rdy_o[5]	O	1 - Indicates that DDR synchronization is already done. 0 - DDR synchronization is not yet started or still in progress.
bit_lock_o[5]	O	1 - Indicates that bit alignment is already done. 0 - Bit alignment is not yet started or still in progress.
word_lock_o[5]	O	1 - Indicates that word alignment is already done. 0 - Word alignment is not yet started or still in progress.
bw_rdy_o	O	1 - Indicates that data training is already done. 0 – Data training is not yet started or still in progress.

Notes:

1. Used only when RGB888 data type is used, otherwise, this port is unused and is not available.
2. LVDS channel 1 input ports are not available when single LVDS channel is selected.
3. Available only when number of output pixels data is more than one. See Table 2.2 for more details.
4. See the Debug Features section for more details on enabling test mode.
5. Can be turned-on if MISC_ON is selected in the interface upon IP generation, or MISC_ON is defined in the defines file.
6. These pixel data are generated from channel 1 when dual channel in selected.

2.1. Interface and Timing Diagrams

Figure 2.2 shows the timing of LVDS 7:1 input interface. There is a 2-bit offset between the rising edge of LVDS clock and the word boundary. Each word is 7-bit long.

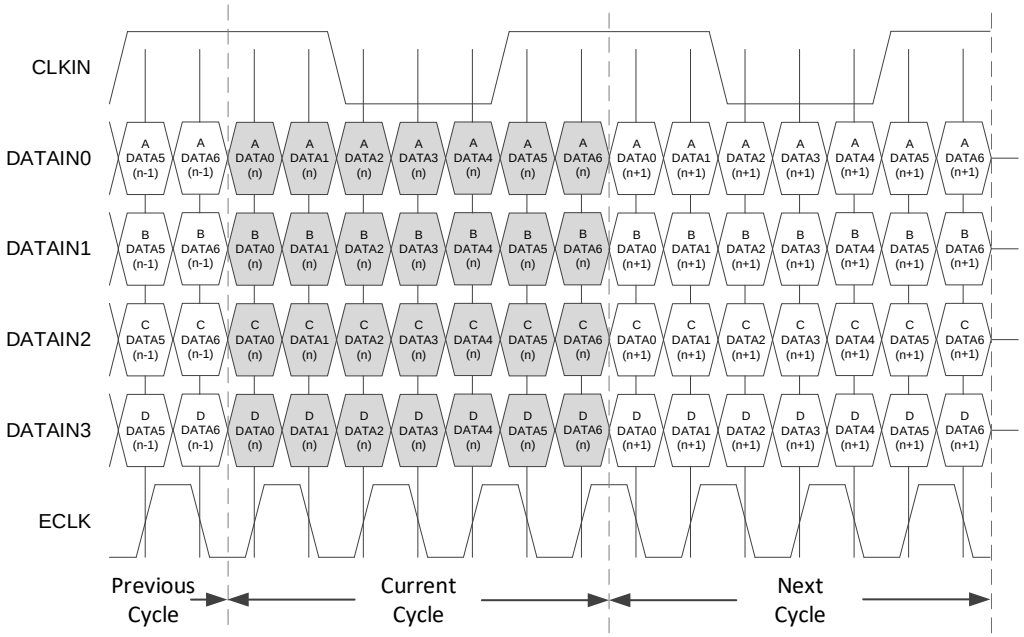

Figure 2.2. OpenLDI/FPD-LINK/LVDS Input Bus Waveform

DATAIN0, DATAIN1, DATAIN2, and DATAIN3 are the data lanes. CLKIN is the LVDS clock lane. For every 7-bit data packet, LSB is the first input serial data to the receiver.

A processor sends video packet data to the FPGA chip via OpenLDI/FPD-LINK (LVDS 7:1) interface. One channel of LVDS 7:1 receiver has a maximum of 5 lanes. Each channel consists of one LVDS clock pair and four LVDS data pairs (RGB888) or three LVDS data pairs (RGB666). Maximum two LVDS 7:1 channels can be used. When dual channel is selected, additional data lanes are activated. The clock runs at 1/7th of the data rate, as per the standard for LVDS 7:1 interface. 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 900 Mb/s.

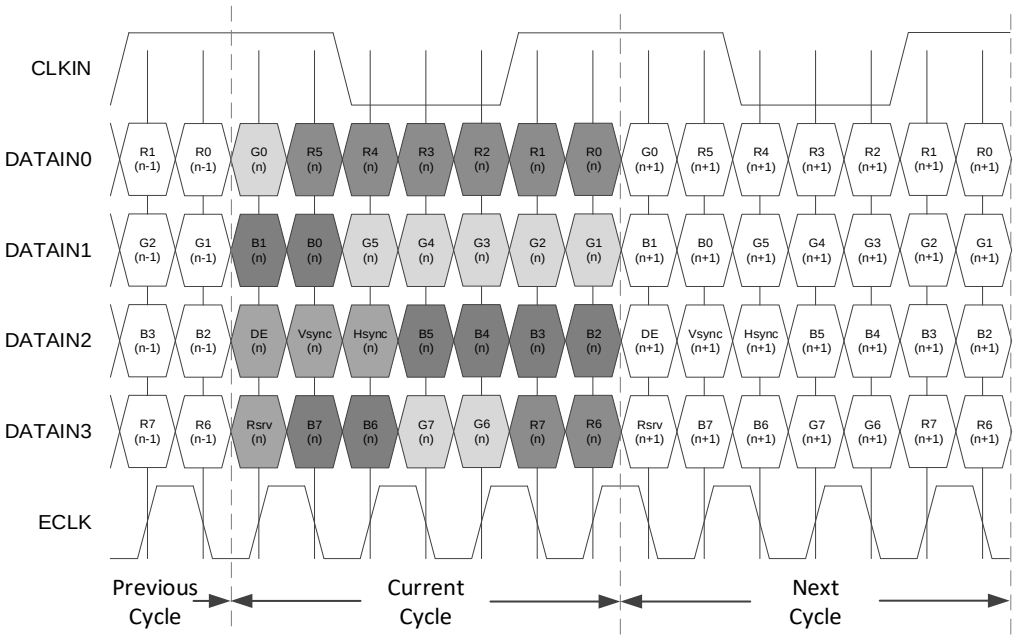

Figure 2.3. Single Channel OpenLDI/FPD-LINK/LVDS Input Bus Waveform for RGB888 Format

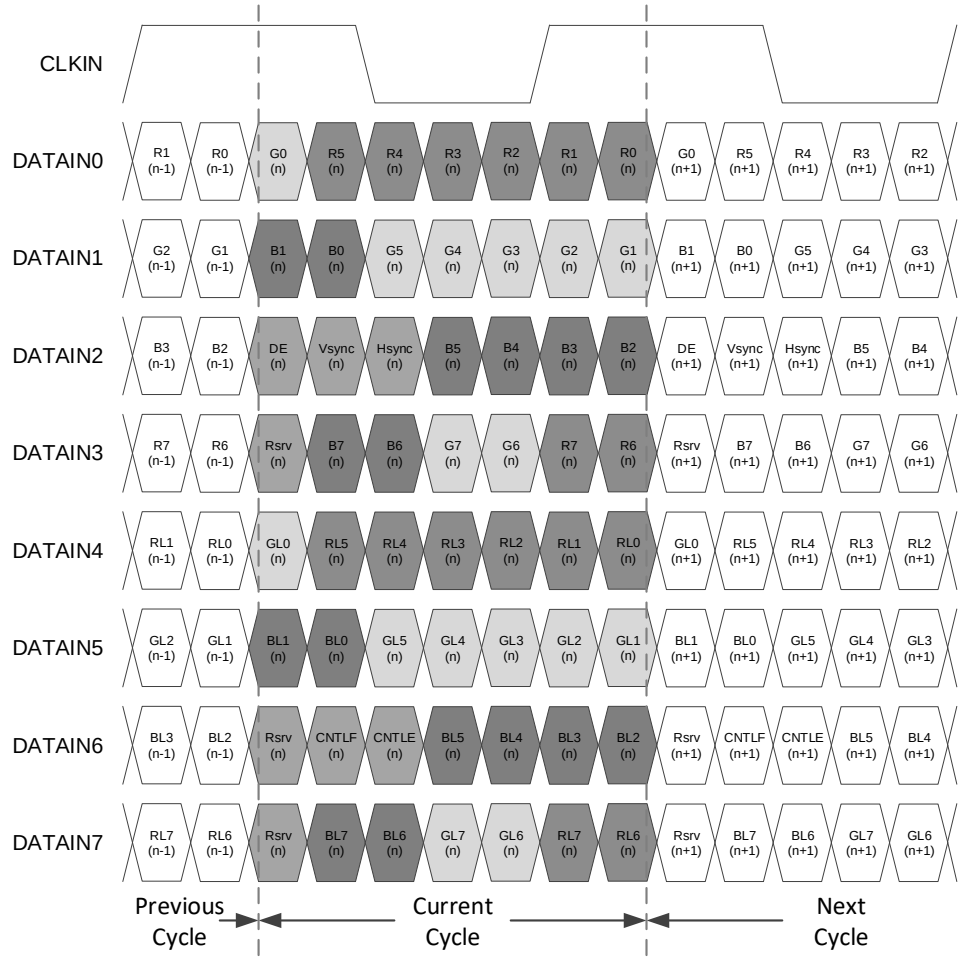

Figure 2.4. Dual Channel OpenLDI/FPD-LINK/LVDS Input Bus Waveform for RGB888 Format

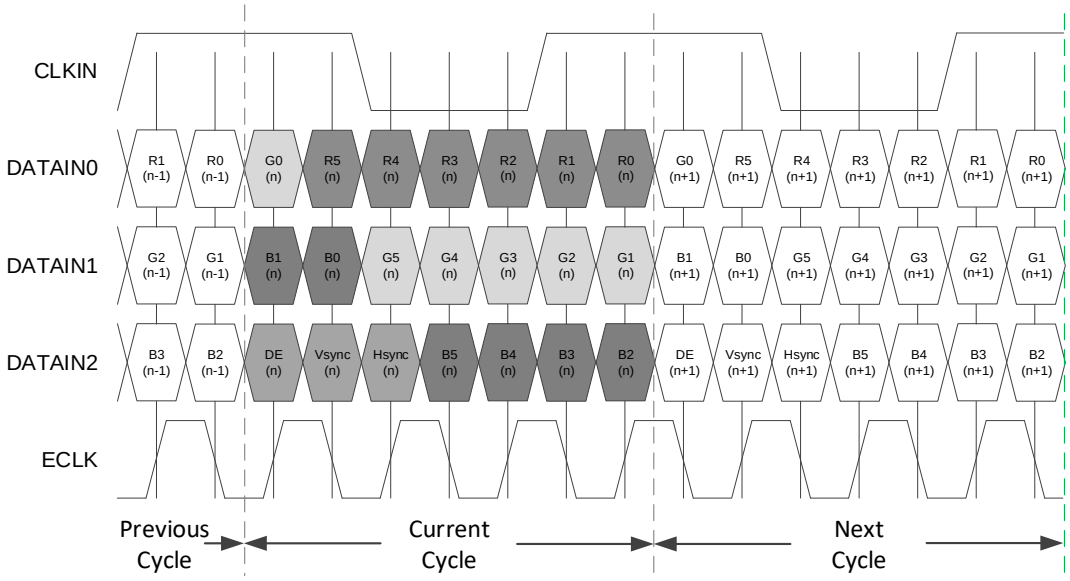

Figure 2.5. Single Channel OpenLDI/FPD-LINK/LVDS Input Bus Waveform for RGB666 Format

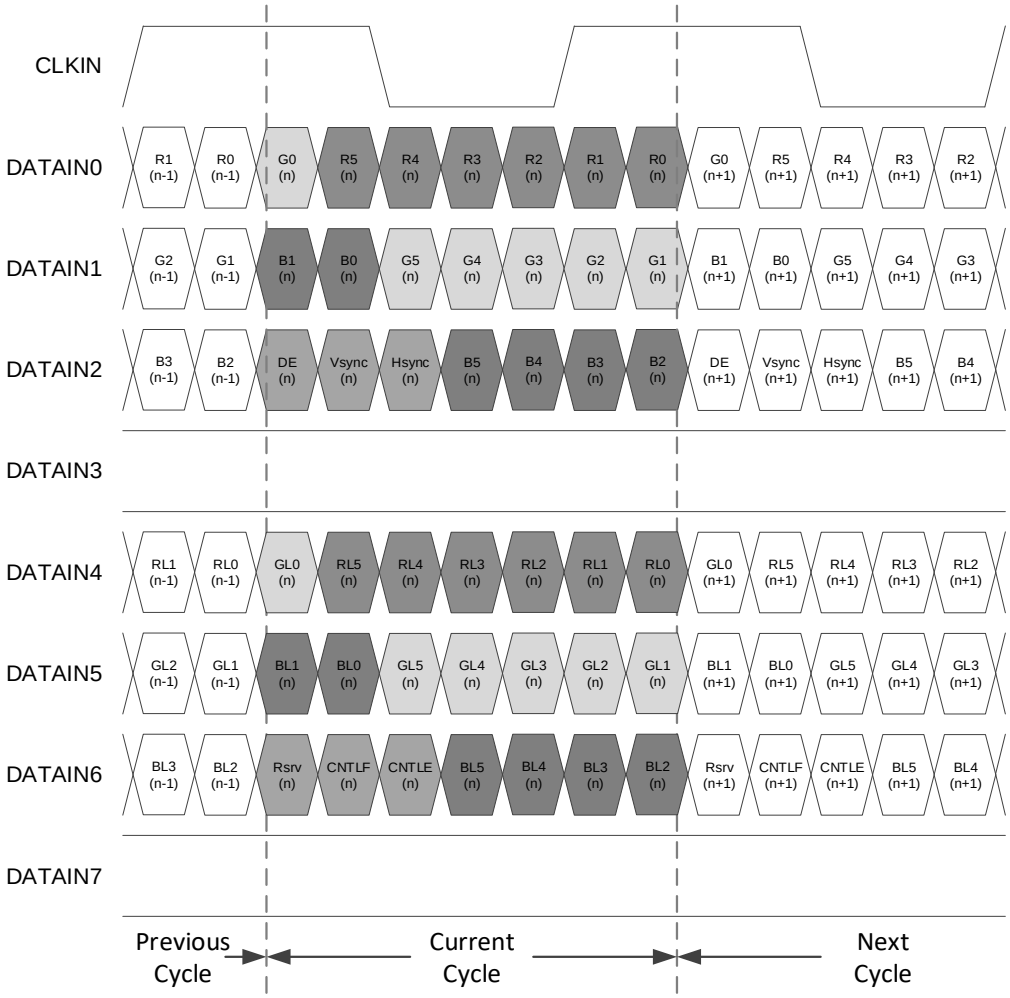

Figure 2.6. Dual Channel OpenLDI/FPD-LINK/LVDS Input Bus Waveform for RGB666 Format

From the processor, data and clock are transmitted serially to the LVDS 7:1 receiver. CrossLink/CrossLinkPlus device performs data processing, such as converting the received LVDS serial data to pixel format.

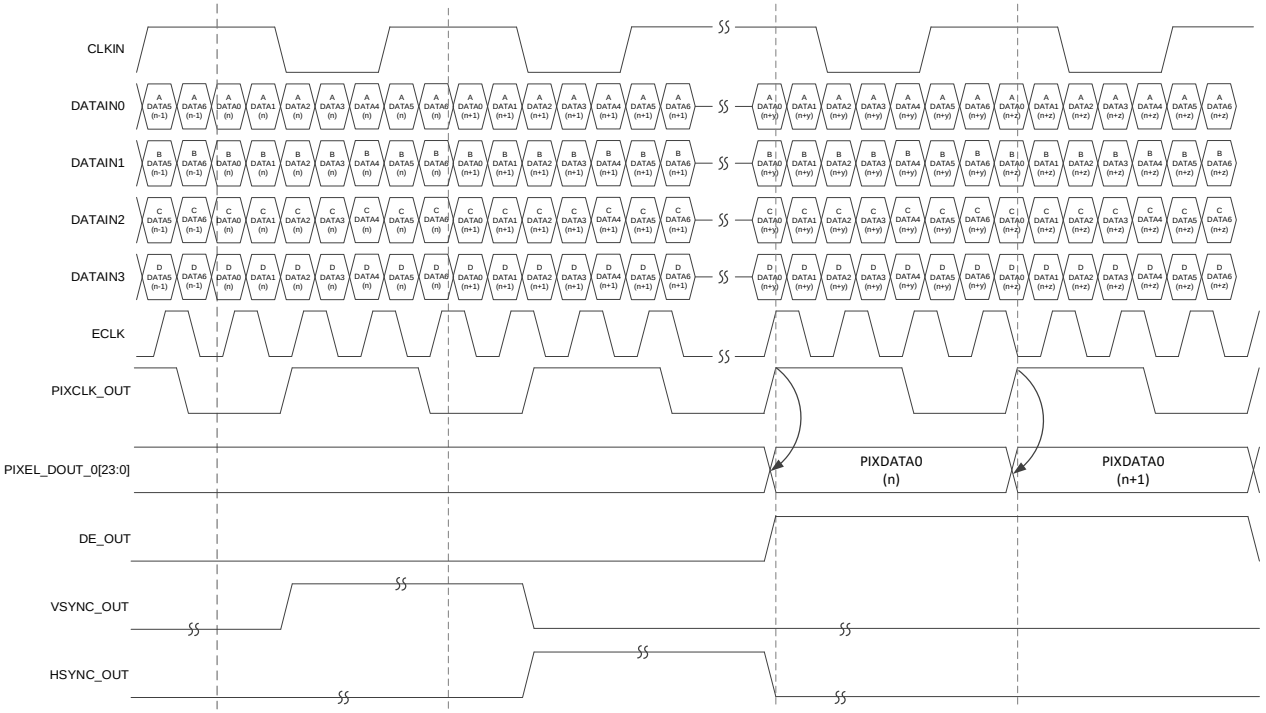

Figure 2.7. Input to Output Waveform for Single Channel OpenLDI/FPD-LINK/LVDS, Rx Gear 7

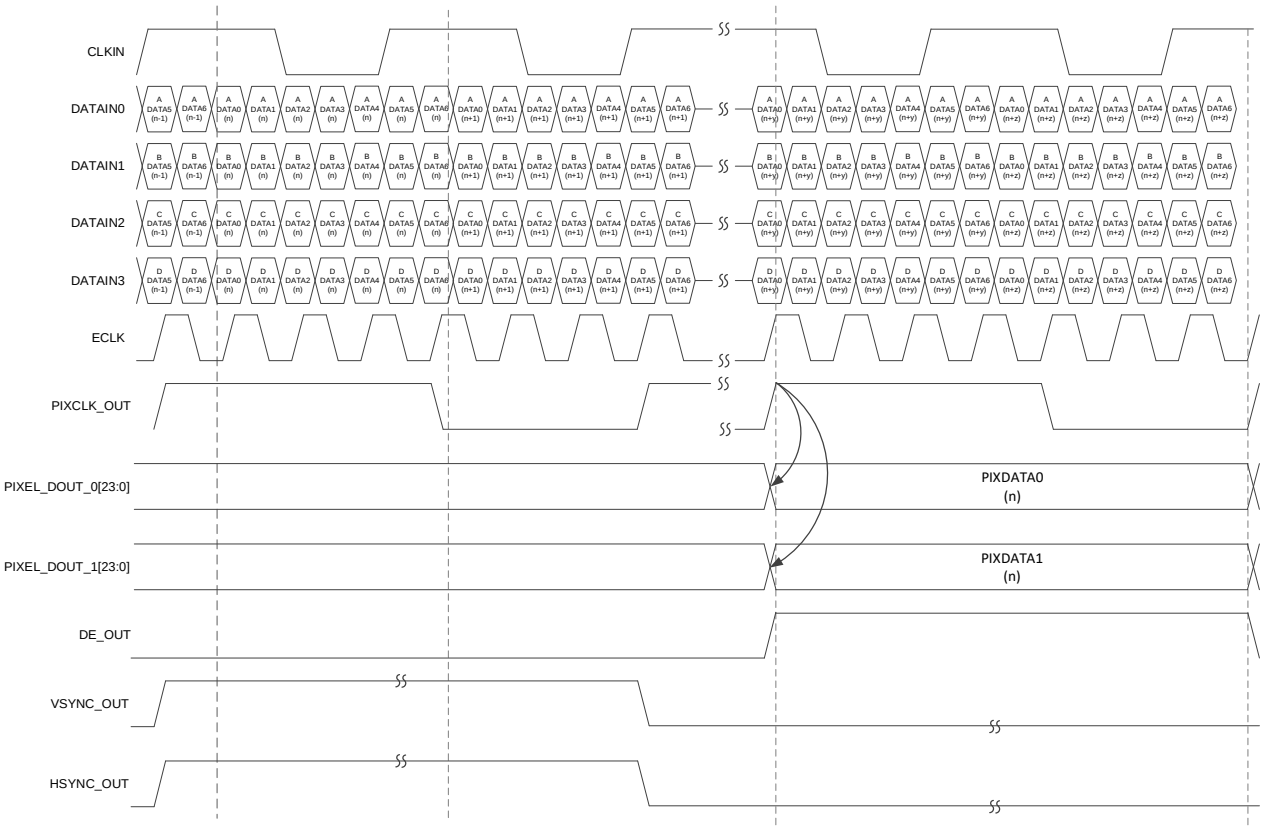

Figure 2.8. Input to Output Waveform for Single Channel OpenLDI/FPD-LINK/LVDS, Rx Gear 14

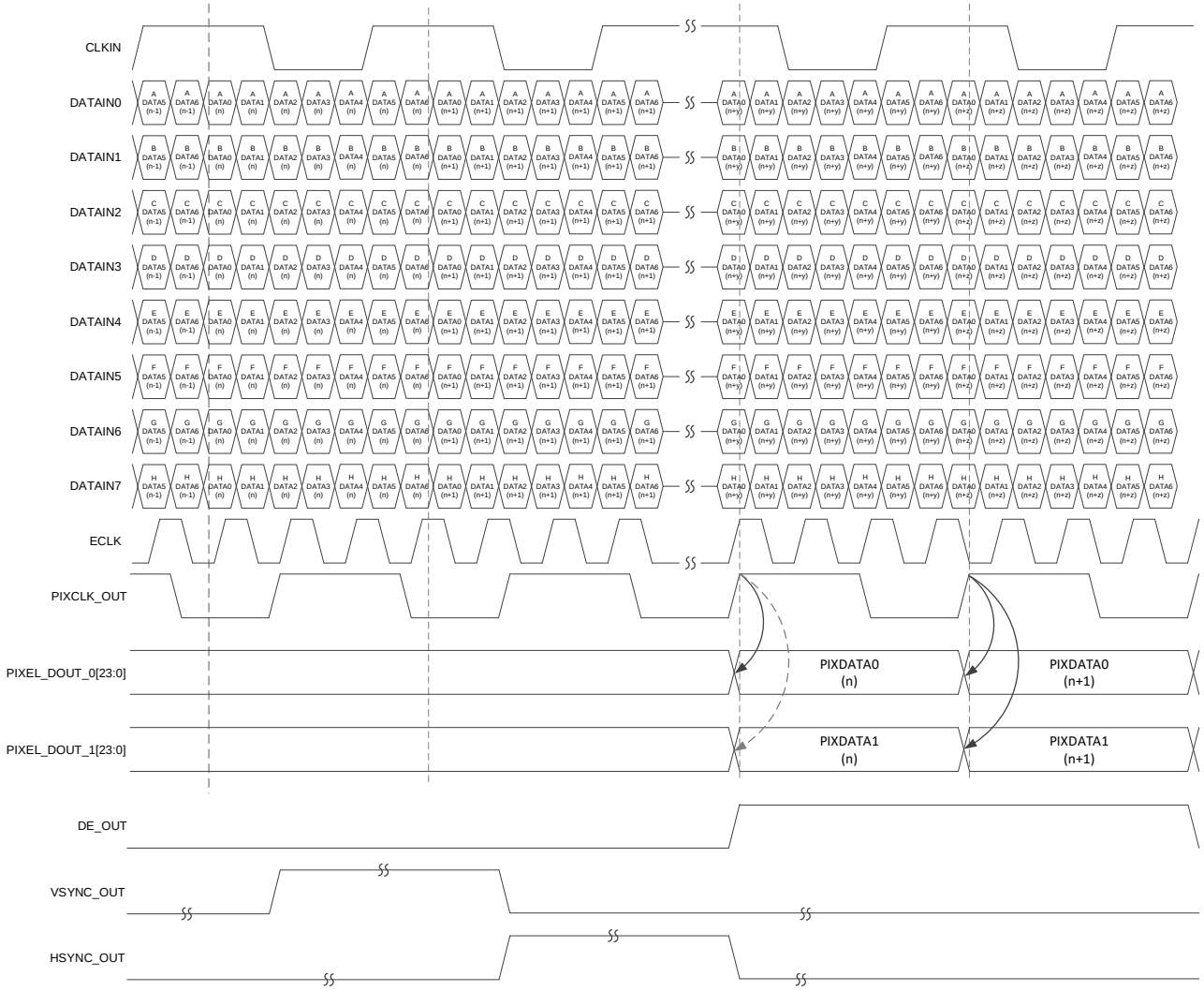

Figure 2.9. Input to Output Waveform for Dual Channel OpenLDI/FPD-LINK/LVDS, Rx Gear 7

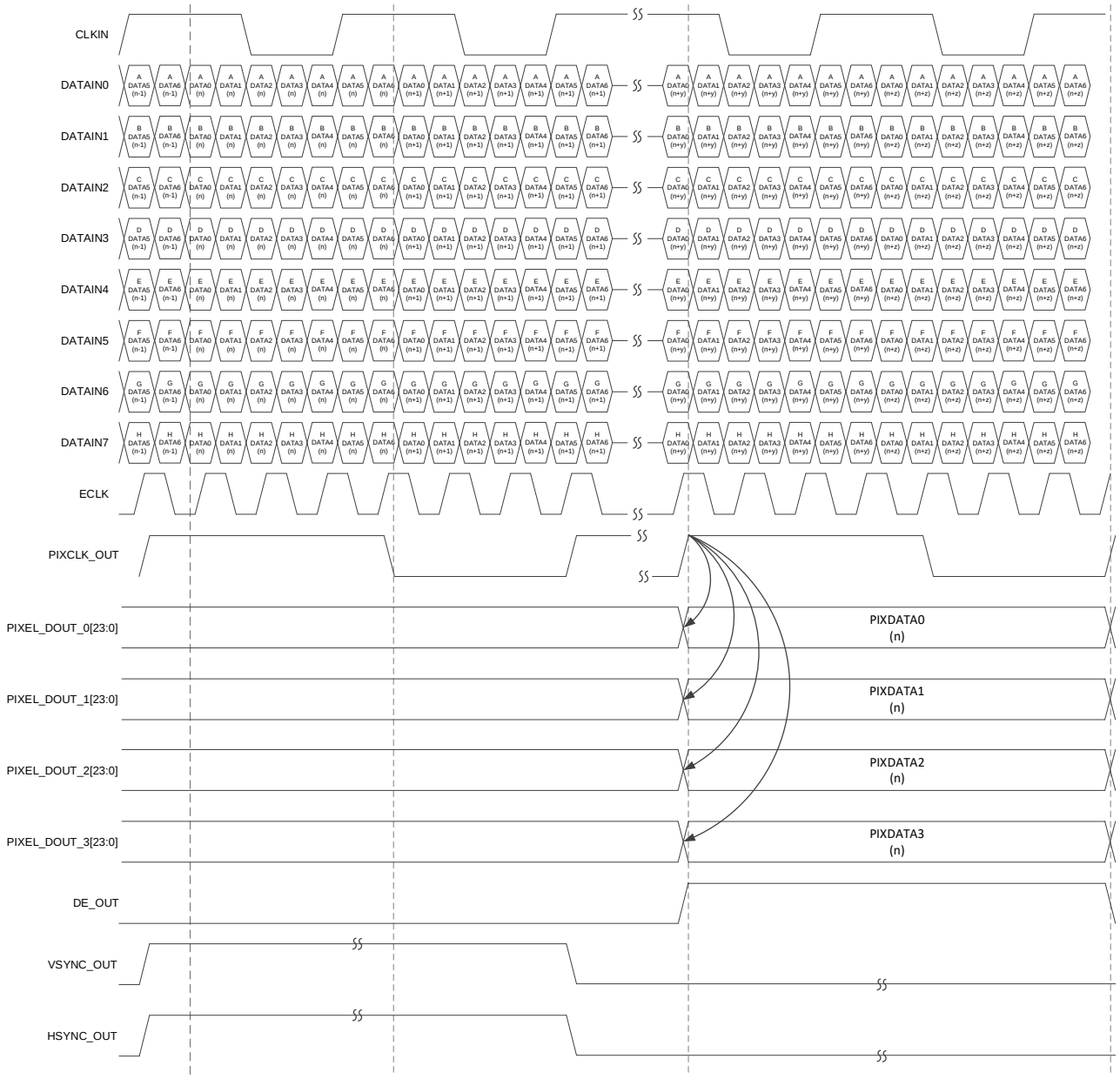

Figure 2.10. Input to Output Waveform for Dual Channel OpenLDI/FPD-LINK/LVDS, Rx Gear 14

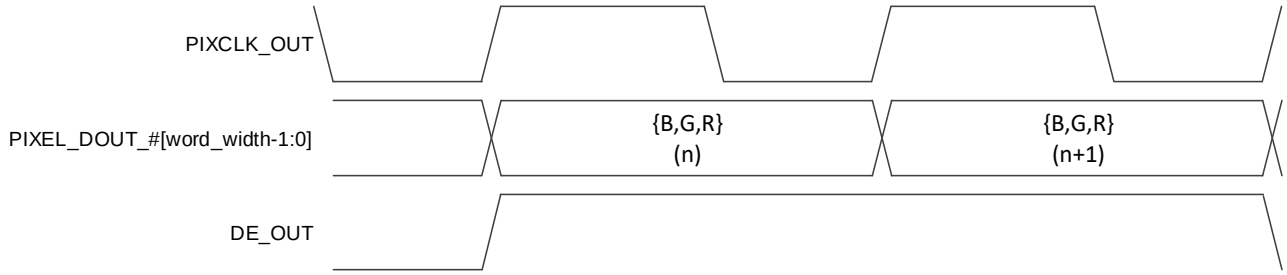

Figure 2.11. Output Pixel Data RGB Arrangement

Table 2.2. Output Pixel Data Summary

Number of FPD-Link Channels	Gear	No. of Output Pixel Data	Output Pixel Clock
1	7	1	LVDS input clock
	14	2	LVDS input clock/2
2	7	2	LVDS input clock
	14	4	LVDS input clock/2

General arrangement on how pixel data are mapped based on configuration are shown in Figure 2.12 and Figure 2.13.

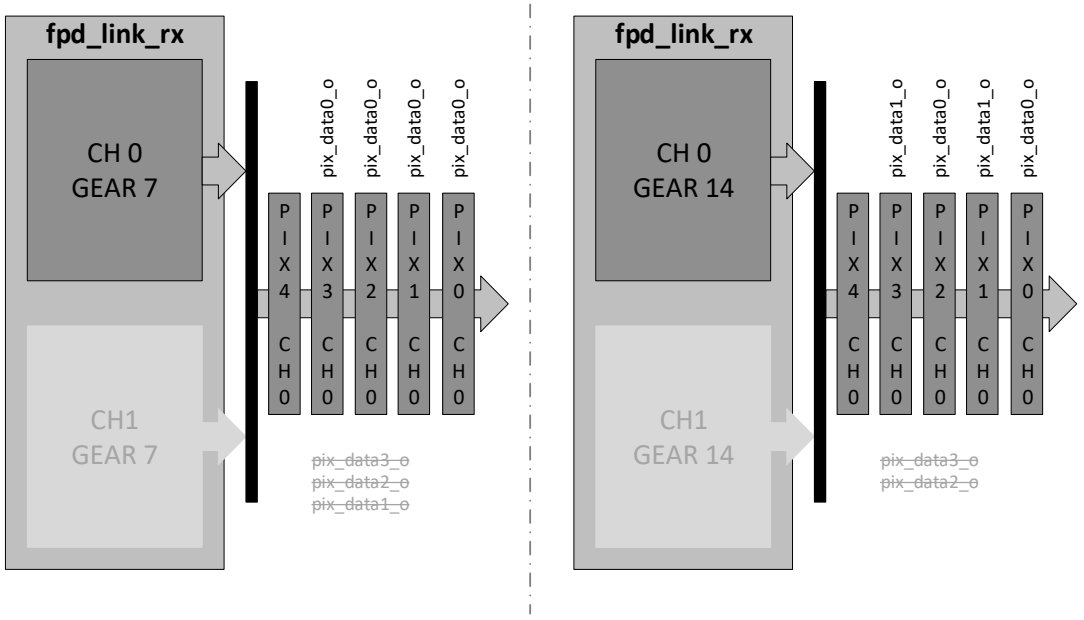

Figure 2.12. Output Pixel Data Arrangement for Single Channel OpenLDI/FPD-LINK/LVDS

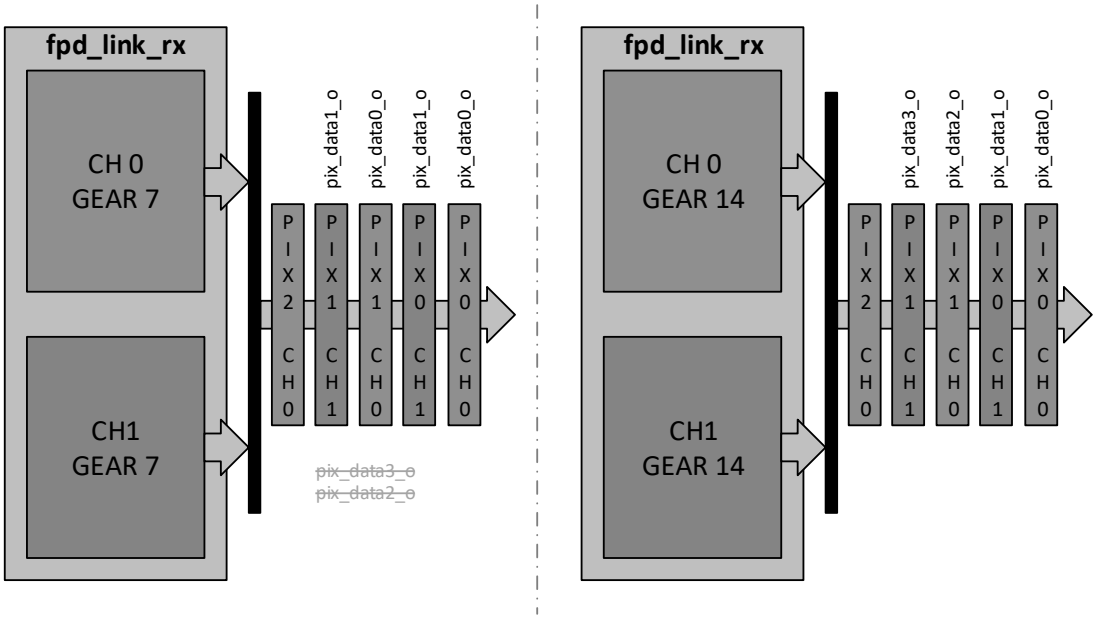

Figure 2.13. Output Pixel Data Arrangement for Dual Channel OpenLDI/FPD-LINK/LVDS

2.2. Clock, Reset and Initialization

2.2.1. Reset and Initialization

Active low reset is used in the design with synchronous release. This is the system reset input connected to LVDS 7:1 Rx module.

Follow this initialization and reset sequence:

1. Assert active low system reset for at least 500 ns.
2. Wait for GPLL to lock (*pll_lock_o*) to be asserted. The *pll_lock_o* is used to indicate that output clock/s of GPLL is/are already stable.
3. Wait for *bw_rdy_o* to be asserted. The *bw_rdy_o* is used to indicate LVDS 7:1 Rx data training is done. Only when *bw_rdy_o* is asserted, valid data can be sampled and correctly transmitted by FPD-Link IP.

The LVDS 7:1 Rx Wrapper is now ready to receive valid data.

Note: Steps 2 and 3 are used to make sure that the system is ready to receive valid data. Exact wait time cannot be specified as the design alignment is dynamic and these flag signals are used to observe when to send valid data.

2.2.2. Clock Domains and Clock Domain Crossing

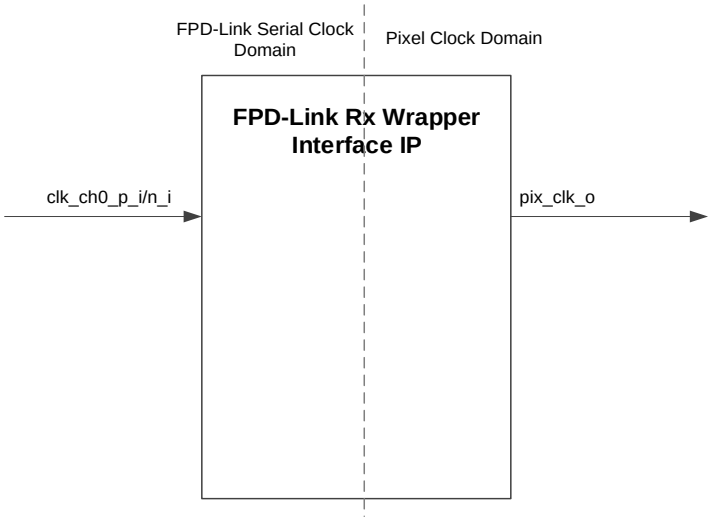

Figure 2.14. Clock Domain Crossing Block Diagram

Table 2.3. Clock Domain Crossing

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

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

Rx Line Rate (Total) $= RX\ Line\ Rate\ (per\ Lane) \times\ No.of\ Rx\ Lanes \times No.of\ Rx\ Channels$

Pixclk (Output Pixel Clock) $= \frac{Rx\ Line\ Rate\ (per\ Lane)}{Rx\ Gear}$

Rx LVDS Input Clock $= Pixclk \times \frac{Rx\ Gear}{7}$

Rx LVDS ECLK $= Pixclk \times \frac{Rx\ Gear}{2}$

No. of Pixels per Pixel Clock $= \frac{Rx\ Gear}{7} \times No.of\ Rx\ Channels$

Example:
IP configuration: Single channel FPD-Link Rx, RGB888 data type, 900 Mb/s Rx Line Rate, Rx Gear 7

Rx Line Rate (Total)	= $900\ Mb/s \times\ 4\ \times 1$	= $3.60\ Gb/s$
Pixclk (Output Pixel Clock)	= $\frac{900\ Mb/s}{7}$	= $128.57\ MHz$
Rx LVDS Input Clock	= $128.57\ MHz \times \frac{7}{7}$	= $128.57\ MHz$
Rx LVDS ECLK	= $128.57\ MHz \times \frac{7}{2}$	= $450\ MHz$
No. of Pixels per Pixel Clock	= $\frac{7}{7} \times 1$	= 1

The general formula to calculate the Rx line rate and clocks of the IP:

Pixels per Second	$= (Total\ Width\ \times Total\ Height) \times Frames\ per\ Second$
Total Bandwidth	$= Pixels\ per\ Second \times Pixel\ Value$
Rx Line Rate (per Lane)	$= \frac{Total\ Bandwidth}{Rx\ Lane}$
Rx Line Rate (Total)	$= Rx\ Line\ Rate\ (per\ Lane) \times No.of\ Rx\ Lanes \times No.of\ Rx\ Channels$
Pixclk (Output Pixel Clock)	$= \frac{Rx\ Line\ Rate\ (per\ Lane)}{Rx\ Gear}$
Rx LVDS Input Clock	$= Pixclk \times \frac{Rx\ Gear}{7}$
Rx LVDS ECLK	$= Pixclk \times \frac{Rx\ Gear}{2}$
No. of Pixels per Pixel Clock	$= \frac{Rx\ Gear}{7} \times No.of\ Rx\ Channels$

Example:
IP configuration: Single channel FPD-Link Rx, RGB888 data type, Rx Gear 7, Rx Lane = 4, Resolution (Total Width × Total Height) = 2200 × 1125, Frames per Second = 60

Pixels per Second	$= (2200\ \times 1125) \times 60$	$= 148500000$
Pixel Value	$= 24$ (derived from RGB888)	
Total Bandwidth	$= 148500000 \times 24$	$= 3564000000$
Rx Line Rate (per Lane)	$= \frac{3564000000}{4}$	$= 891\ Mb/s$
Rx Line Rate (Total)	$= 891\ Mb/s \times 4 \times 1$	$= 3.564\ Gb/s$
Pixclk (Output Pixel Clock)	$= \frac{891\ Mb/s}{7}$	$= 127.28\ MHz$
Rx LVDS Input Clock	$= 127.28\ MHz \times \frac{7}{7}$	$= 127.28\ MHz$
Rx LVDS ECLK	$= 127.28\ MHz \times \frac{7}{2}$	$= 445.48\ MHz$
No. of Pixels per Pixel Clock	$= \frac{7}{7} \times 1$	$= 1$

2.3. Design and Module Description

2.3.1. FPD-Link Rx Wrapper Module

The *fpd_link_rx_wrapper.v* module instantiates *fpd_link_rx*, general purpose PLL (*GPLL*), *test_mode*, *lvds71_pxlmap*, and synchronizer modules. The *fpd_link_rx* module is the core module which performs the data training and serial to parallel conversion. *GPLL* is used for fast clock generation. The *test_mode* is used for internal self-checking. *lvds71_pxlmap* is used to decode the output parallel data of *fpd_link_rx* and convert them into pixel format. Synchronizers are two-level synchronizers used to sync the system reset into different clock domains before it is used in the system.

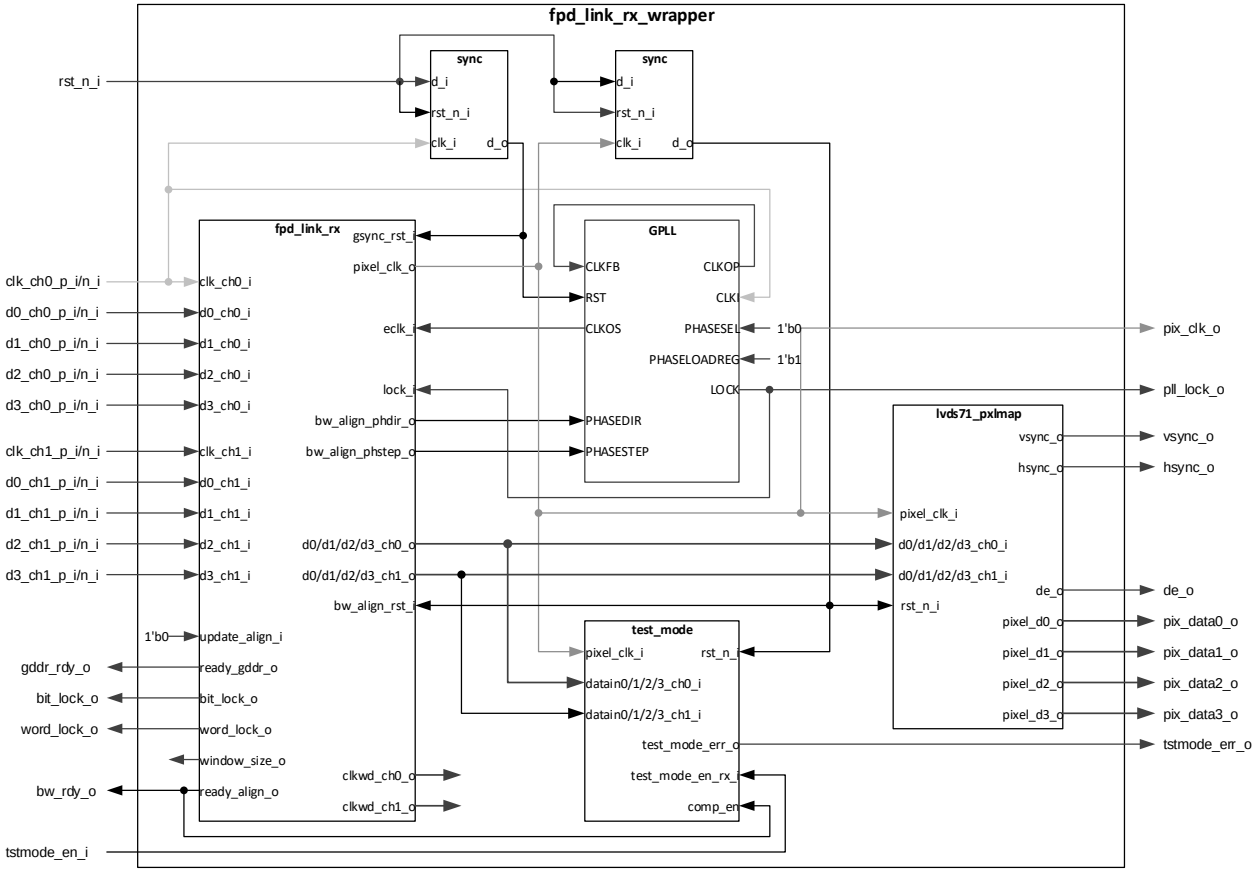

Figure 2.15. FPD-Link Rx Wrapper Block Diagram

2.3.2. FPD-Link Rx Module

The *fpd_link_rx* module instantiates *GDDR_SYNC, BW_ALIGN*, several Lattice primitives, and *lvds71_ddr_group.v* module. *GDDR_SYNC* module is required to initialize and synchronize DDR clock and tolerate the large skew between stop and reset of the DDR components. To properly sample the received data, *BW_ALIGN* is used to do data training that includes bit and word alignment with respect to LVDS input clock. Both 1:7 and 1:14 DDR gearing can be used. The sub-block connection is based on CrossLink/CrossLinkPlus GDDR7:1 Rx Usage Model guide.

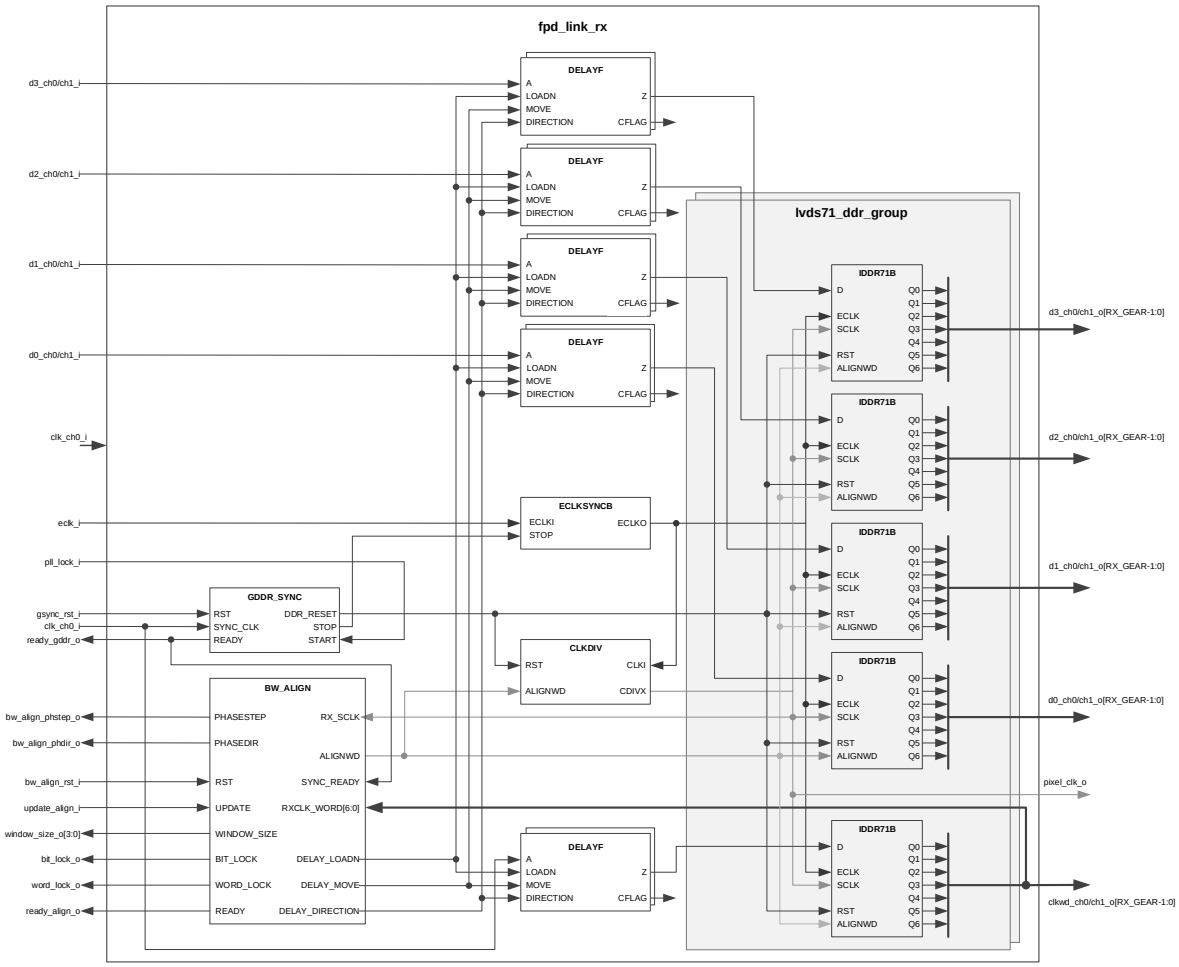

Figure 2.16. FPD-Link Rx Block Diagram

LVDS data are fed to the I/O logic IDDR71 register. LVDS clock is also fed to the I/O logic IDDR71 register, as well as to PLL. PLL is used to generate the sampling clock (*ECLK*) which is 3.5 times faster than the LVDS clock. To allow the placing of the edge of clock in the middle of the data valid window, alignment IP together with PLL is used to shift *ECLK* to normally 90 degrees. *CLKDIV* is used to generate a slower clock (*SCLK*), which is 3.5 times slower than *ECLK. SCLK* is used as the output clock of IDDR71 IP.

After *ECLK* and *SCLK* are generated, LVDS 7:1 soft IP performs bit and word alignment. Bit alignment is placing *ECLK* in the middle of the data training window, and word alignment is getting the correct training sequence after bit alignment. Training pattern is *7'b1100011* and alignment is done with respect to LVDS clock. It is expected for the LVDS clock and data to be edge-aligned with each other. So when alignment is done with respect to the LVDS clock, the same goes for the LVDS data lanes. Dual x4 LVDS 7:1 Rx can only be supported, if the two channels are sharing the same clock.

Table 2.4. FPD-Link Rx Pin List Summary

Port Name	Width	Dir	Type	Description
gsync_rst_i	1	I	LVCMOS	Asynchronous active high reset for GDDR_SYNC. 1 – System on reset
bw_align_rst_i	1	I	LVCMOS	Asynchronous active high reset for BW_ALIGN. 1 – System on reset
eclk_i	1	I	LVCMOS	Fast clock input for ECLKSYNC. Must come from CLKOS of GPLL.
clk_ch0_i	1	I	LVDS	LVDS Input clock to LVDS 7:1 RX Wrapper channel 0. For all dual channel configuration, only channel 0 clock is used as reference clock for the system.
clk_ch1_i	1	I	LVDS	LVDS Input clock to LVDS 7:1 RX Wrapper channel 1. For all 2:1 configuration, only channel 0 clock is used as reference clock for the system, channel 1 clock is unused.
d0_ch0_i	1	I	LVDS	LVDS Input data lane 0 to LVDS 7:1 RX Wrapper channel 0
d1_ch0_i	1	I	LVDS	LVDS Input data lane 1 to LVDS 7:1 RX Wrapper channel 0
d2_ch0_i	1	I	LVDS	LVDS Input data lane 2 to LVDS 7:1 RX Wrapper channel 0
d3_ch0_i	1	I	LVDS	LVDS Input data lane 3 to LVDS 7:1 RX Wrapper channel 0
d0_ch1_i	1	I	LVDS	LVDS Input data lane 0 to LVDS 7:1 RX Wrapper channel 1
d1_ch1_i	1	I	LVDS	LVDS Input data lane 1 to LVDS 7:1 RX Wrapper channel 1
d2_ch1_i	1	I	LVDS	LVDS Input data lane 2 to LVDS 7:1 RX Wrapper channel 1
d3_ch1_i	1	I	LVDS	LVDS Input data lane 3 to LVDS 7:1 RX Wrapper channel 1
clkwd_ch0_o	RX_GEAR	O	LVCMOS	Parallel output clkword for channel 0
d3_ ch0_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 3 channel 0
d2_ ch0_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 2 channel 0
d1_ ch0_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 1 channel 0
d0_ ch0_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 0 channel 0
clkwd_ch1_o	RX_GEAR	O	LVCMOS	Parallel output clkword for channel 1
d3_ch1_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 3 channel 1
d2_ch1_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 2 channel 1
d1_ch1_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 1 channel 1
d0_ch1_o	RX_GEAR	O	LVCMOS	Parallel output data for lane 0 channel 1
pixel_clk_o	1	O	LVCMOS	Output clock of LVDS data Equivalent to SCLK
update_align_i	1	I	LVCMOS	Active high input to BW_ALIGN module that is used to restart the bit and word alignment. (This feature is not used in FPD-Link, always tied to 0.) 1 – Restart alignment process
lock_i	1	I	LVCMOS	Starts the GDDR_SYNC process; connected to lock signal of GPLL.
bw_align_phdir_o	1	O	LVCMOS	Output of BW_ALIGN. Connected to GPLL that is used to indicate the phase direction for bit and word alignment.
bw_align_phstep_o	1	O	LVCMOS	Output of BW_ALIGN. Connected to GPLL that is used to indicate the phase step for bit and word alignment.
ready_gddr_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
ready_align_o	1	O	LVCMOS	1 – Indicates that bit and word alignment are done (via BW_ALIGN) 0 – Indicates that bit and word alignment is still not started or in progress (via BW_ALIGN)

Port Name	Width	Dir	Type	Description
bit_lock_o	1	O	LVCMOS	1 – Indicates that bit alignment is done (via BW_ALIGN) 0 – Indicates that bit alignment is not yet done (via BW_ALIGN)
word_lock_o	1	O	LVCMOS	1 – Indicates that word alignment is done (via BW_ALIGN) 0 – Indicates that word alignment is not yet done (via BW_ALIGN)
window_size_o	4	O	LVCMOS	Indicates the size of the data window in the LVDS 7:1 RX

Table 2.5. FPD-Link Rx Parameter List

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

2.3.3. LVDS71 DDR Group Module

LVDS71 DDR group module is used to convert the incoming serial data into parallel format. Output bus width depends on the DDR gearing set through parameter. For dual channel FPD-Link configuration, two instances of LVDS71 DDR group are instantiated.

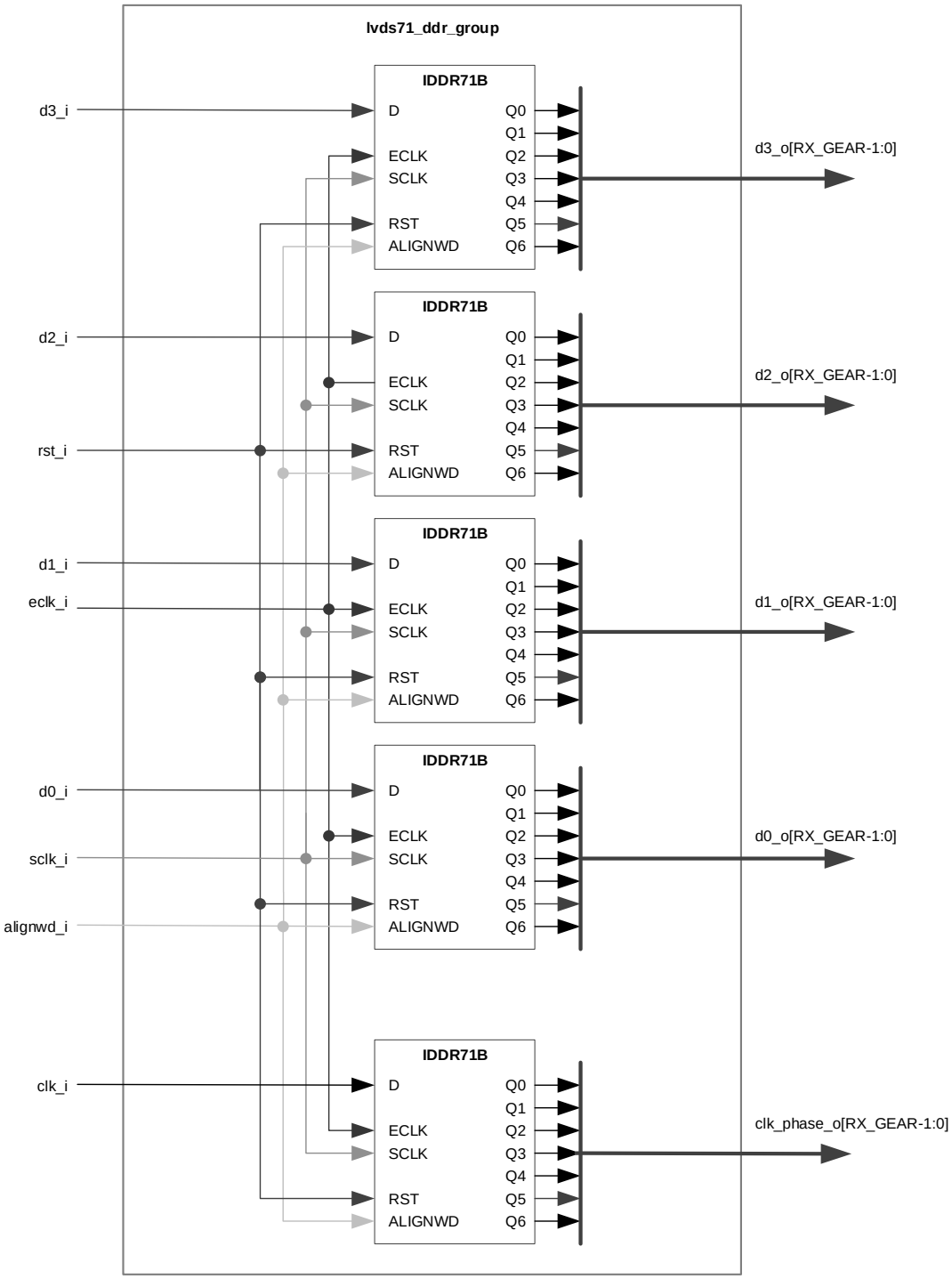

Figure 2.17. LVDS71 DDR Group Block Diagram

Table 2.6. LVDS71 DDR Group Module Pin List

Port Name	Width	Dir	Type	Description
d0_i	1	I	LVDS	LVDS Input data lane 0
d1_i	1	I	LVDS	LVDS Input data lane 1
d2_i	1	I	LVDS	LVDS Input data lane 2
d3_i	1	I	LVDS	LVDS Input data lane 3
clk_i	1	I	LVDS	LVDS Input clock. For all 2:1 configuration, only channel 0 clock is used as reference clock for the system.
eclk_i	1	I	LVCMOS	Input ECLK to the DDR. Must come from ECLKSYNC.
sclk_i	1	I	LVCMOS	Input SCLK to the DDR. Must come from CLKDIV.
rst_i	1	I	LVCMOS	Synchronous reset to DDR. Must come from GDDR_SYNC.
alignwd_i	1	I	LVCMOS	Alignwd input for DDR. Shifts data with respect to clock. Must come from BW_ALIGN.
clk_phase_o	RX_GEAR	O	LVCMOS	Parallel output data (clkword).
d3_o	RX_GEAR	O	LVCMOS	Parallel output data 3
d2_o	RX_GEAR	O	LVCMOS	Parallel output data 2
d1_o	RX_GEAR	O	LVCMOS	Parallel output data 1
d0_o	RX_GEAR	O	LVCMOS	Parallel output data 0

Table 2.7. LVDS71 DDR Group Parameter List

Parameters	Value	Description	Operation
RX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 1:7 Gearing 14 – 1:14 Gearing	parameter RX_GEAR = <val>
NUM_RX_LANE	3, 4	Specify number of data lanes per Rx link. 3 – RGB666 4 – RGB888	parameter NUM_RX_LANE = <val>

2.3.4. GDDR SYNC Module

GDDR_SYNC is used for DDR clock initialization and synchronization. This is a required module for FPD-Link Rx Interface. Start of DDR synchronization is dependent on the assertion of *GPLL* lock. When PLL lock is asserted, *GDDR_SYNC* stops *ECLKSYNC* block and resets the DDR and *CLKDIV* before starting *ECLKSYNC* again. This is to ensure that *ECLK* and *SCLK* are synchronized.

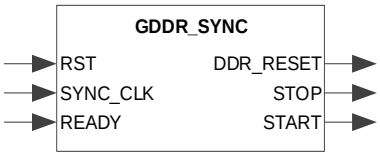

Figure 2.18. GDDR_SYNC Block Diagram

2.3.5. BW ALIGN Module

BW_ALIGN module is used for data training of FPD-Link Rx. Bit and word alignment is done by this module. *BW_ALIGN* IP is started when the ready signal of *GDDR_SYNC* is asserted. Bit alignment places *ECLK* in the middle of the data training window, and word alignment gets the correct training sequence after bit alignment. The training pattern is *7'b1100011*, and alignment is done with respect to LVDS clock. If correct training data is already sampled properly, ready signal of *BW_ALIGN* is asserted. Only then can the valid data be sampled correctly by LVDS 7:1 Rx. It is expected for the LVDS clock and LVDS data to be edge-aligned with each other so that when alignment is done with respect to the LVDS clock, LVDS data lanes are also aligned.

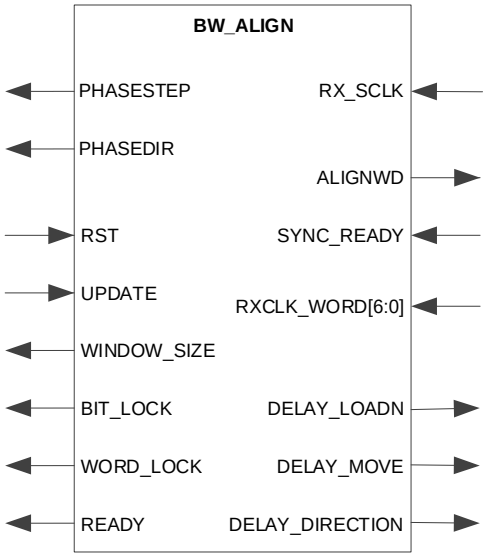

Figure 2.19. BW_ALIGN Block Diagram

2.3.6. CLKDIV

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

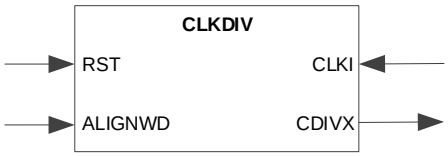

Figure 2.20. CLKDIV Block Diagram

2.3.7. ECLKSYNC

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

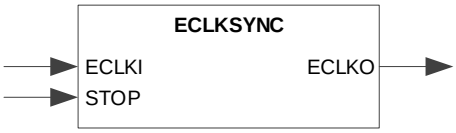

Figure 2.21. ECLKSYNC Block Diagram

2.3.8. GPLL

GPLL is a general purpose Lattice PLL. In FPD-Link Rx interface, use of PLL is required. This is used to generate the ECLK and start the initialization, synchronization, and data alignment processes.

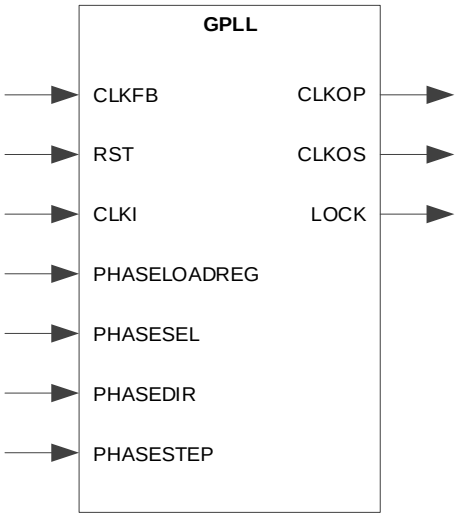

Figure 2.22. GPLL Block Diagram

2.3.9. LVDS71 Pixel Map Module

lvds71_pxlmap is used to decode the output parallel data of *fpd_link_rx* and convert them into pixel format. Up to four valid output pixel data is supported depending on the design configuration. The timing diagrams in Figure 2.12 and Figure 2.13 show how data is mapped.

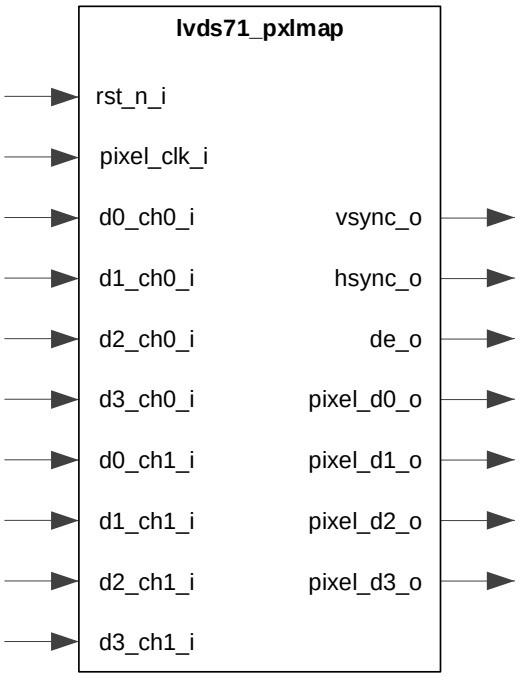

Figure 2.23. LVDS71 Pixel Map Block Diagram

Table 2.8. LVDS71 Pixel Map Pin List Summary

Port Name	Width	Dir	Type	Description
pixel_clk_i	1	I	LVCMOS	Input pixel clock
rst_n_i	1	I	LVCMOS	Asynchronous active low reset
d0_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 0 of channel 0
d1_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 1 of channel 0
d2_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 2 of channel 0
d3_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 3 of channel 0
d0_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 0 of channel 1
d1_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 1 of channel 1
d2_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 2 of channel 1
d3_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 3 of channel 1
pixel_d0_o	24 - RGB88 18 - RGB666	O	LVCMOS	Output pixel data 0. Bus width depends on the data type selected.
pixel_d1_o	24 - RGB88 18 - RGB666	O	LVCMOS	Output pixel data 1. Bus width depends on the data type selected.
pixel_d2_o	24 - RGB88 18 - RGB666	O	LVCMOS	Output pixel data 2. Bus width depends on the data type selected.
pixel_d3_o	24 - RGB88 18 - RGB666	O	LVCMOS	Output pixel data 3. Bus width depends on the data type selected.
de_o	1	O	LVCMOS	Output data enable for parallel interface.
hsync_o	1	O	LVCMOS	Output horizontal sync for parallel interface.
vsync_o	1	O	LVCMOS	Output vertical sync for parallel interface.

***Note:** Port is always available. Drive to 0s if port is not used.

Table 2.9. LVDS71 Pixel Map Parameter List

Parameters	Value	Description	Operation
NUM_RX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_RX_CH = <val>
NUM_RX_LANE	3, 4	Specify number of data lanes per Rx link. 3 – LVDS lane - set automatically to 3 4 – LVDS lane - set automatically to 4	parameter NUM_RX_LANE = <val>
RX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 1:7 Gearing 14 – 1:14 Gearing	parameter RX_GEAR = <val>

2.3.10. Test Mode Module

The *test_mode* module is used to check data from FPD-link Rx before it is decoded into control and pixel data. A predefined set of data (set in *TEST_DATA* and driven as LVDS input data) is compared internally to the actual output of *fpd_link_rx* module.

The comparison of data is enabled only after bit and word alignment is completed. If data mismatch is encountered, *test_mode_err_o* is set to high until reset is asserted or chip is powered down.

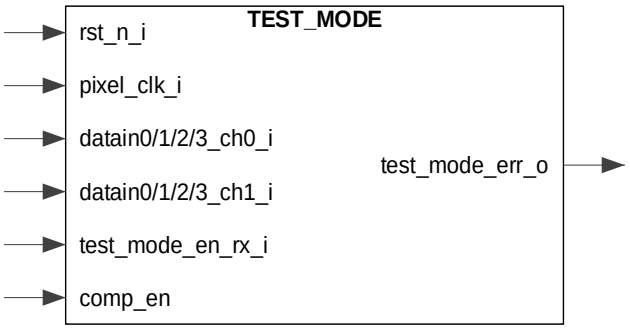

Figure 2.24. Test Mode Block Diagram

Table 2.10. Test Mode Pin List Summary

Port Name	Width	Dir	Type	Description
pixel_clk_i	1	I	LVCMOS	Input pixel clock.
rst_n_i	1	I	LVCMOS	Asynchronous active low reset
test_mode_en_rx_i	1	I	LVCMOS	Enable or disable test mode. 1 - Enable test mode. 0 - Disable test mode.
comp_en	1	I	LVCMOS	Indicates when to start comparing data. Connected to ready_align_o output of fpd_link_rx module.
test_mode_err_o	1	O	LVCMOS	1 - Indicates that compare mismatch is encountered when test mode is enabled. 0 - No error is encountered. Automatically set to 0 when test mode is disabled.
datain0_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 0 of channel 0
datain1_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 1 of channel 0
datain2_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 2 of channel 0
datain3_ch0_i*	RX_GEAR	I	LVCMOS	Input data from lane 3 of channel 0
datain0_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 0 of channel 1
datain1_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 1 of channel 1
datain2_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 2 of channel 1
datain3_ch1_i*	RX_GEAR	I	LVCMOS	Input data from lane 3 of channel 1

***Note:** Port is always available. Drive to 0s if port is not used.

Table 2.11. Test Mode Parameter List

Parameters	Value	Description	Operation
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For dual channel configuration, the same TEST DATA is used for both channels. 28 – Data width for RGB888 21 – Data width for RGB666	parameter TEST_DATA = <val>
NUM_RX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_RX_CH = <val>
NUM_RX_LANE	3, 4	Specify number of data lanes per Rx link. 3 – LVDS lane - Set automatically to 3 4 – LVDS lane - Set automatically to 4	parameter NUM_RX_LANE = <val>
RX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 1:7 Gearing 14 – 1:14 Gearing	parameter RX_GEAR = <val>

2.3.11. Synchronizer Module

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

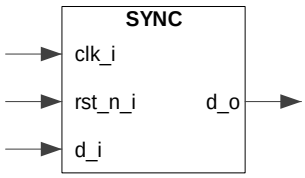

Figure 2.25. Synchronizer Block Diagram

Table 2.12. Synchronizer Pin List Summary

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

Table 2.13. 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>

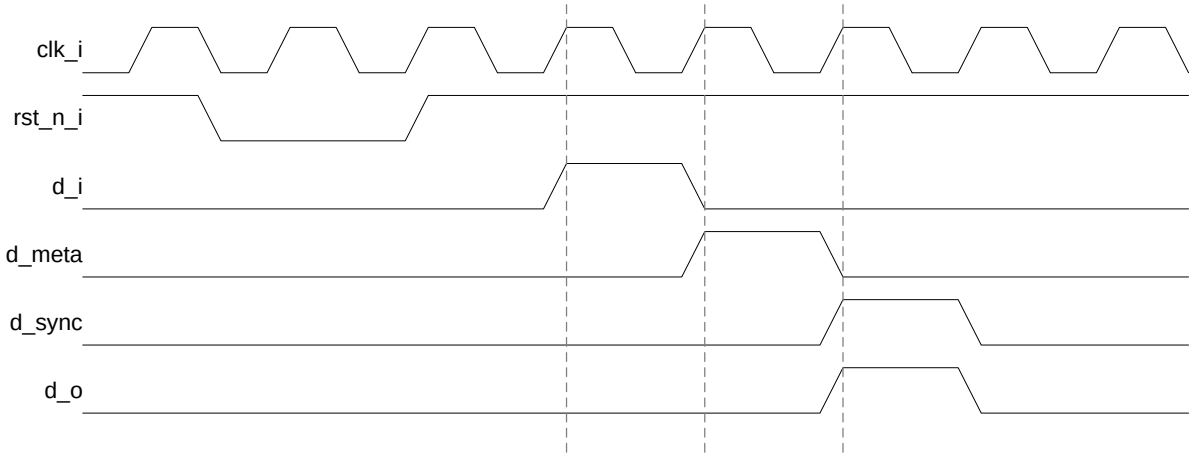

Figure 2.26. Synchronizer Timing Diagram

3. Compiler Directives and Parameter Settings

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

3.1. Parameters Settings

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

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

Parameters	Value	Description	Operation
NUM_RX_CH	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	parameter NUM_RX_CH = <val>
RX_GEAR	7, 14	Specify what DDR71 gearing is used. 7 – 1:7 Gearing 14 – 1:14 Gearing	parameter RX_GEAR = <val>
DATA_TYPE	RGB888, RGB666	Specify the data type. RGB666 - Automatically set NUM_RX_LANE to 3 RGB888 - Automatically set NUM_RX_LANE to 4	parameter DATA_TYPE = “<val>”
NUM_RX_LANE	3, 4	Specify number of data lanes per RX link 3 – RGB666 4 – RGB888	parameter NUM_RX_LANE = <val>
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For dual channel configuration, the same TEST DATA is used for both channels. 28 – Data width for RGB888 21 – Data width for RGB666	parameter TEST_DATA = <val>

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

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

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

Parameter	Attribute	Options	Description
Number of Output Pixels	Read-Only	1,2,4	Specify the number of output pixels per pixel clock. Computed based on selected Number of Rx Channels and Rx Gear.
Test Mode Data	User-Input	<Value>	Available only when Test Mode is enabled. 28 – Data width for RGB888 21 – Data width for RGB666

3.2. Compiler Directives

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

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

Parameters	Value	Description	Operation
NUM_RX_CH_<#>	1, 2	Specify how many LVDS links are used. 1 – Single Link 2 – Dual Link	`define NUM_RX_CH_<val>
RX_GEAR_<#>	7, 14	Specify what DDR71 gearing is used. 7 – 1:7 Gearing 14 – 1:14 Gearing	`define RX_GEAR_ <val>
RGB<#>	888, 666	Specify the data type. RGB666 – Automatically set NUM_RX_LANE to 3 RGB888 – Automatically set NUM_RX_LANE to 4	`define RGB<val>
TEST_DATA	<value>	Pre-defined data used when test mode is enabled. For dual channel configuration, the same TEST DATA is used for both channels. 28 – Data width for RGB888 21 – Data width for RGB666	`define TEST_DATA <val>
PIX<#>_PER_CLK	1, 2, 4	Specify the number of output pixels per pixel clock. 4 – Rx CH=2, Gear 14 2 – Rx CH=2, Gear 7 or Rx CH=1, Gear 14 1 – Rx 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 to check data from FPD-Link Rx before it is decoded to control and pixel data. See Test Mode Module section for the module description. A predefined set of data (set in *TEST_DATA* and driven as LVDS input data) is compared internally to the actual output of fpd_link_rx module. The comparison of data is enabled only after bit and word alignment is completed. If data mismatch is encountered, *tstmode_err_o* is set to high until reset is asserted or chip is powered down. For dual channel configuration, the same *TEST_DATA* is used.

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. Assert active low system reset for at least 500 ns. Release system reset.
5. Continuously drive the 28-bit/21-bit data (should be the same as configured in *TEST_DATA*) per input clock per channel.
   - 28-bit data is divided into 4 data lanes (see Figure 2.2).
   - MSB is *DATAIND[6]* and LSB is *DATAINA[0].*
   - {DATAIND[6:0], DATAINC[6:0], DATAINB[6:0], DATAINA[6:0]}.
   - LSB of each lane is the 1st datain.
   - *CLKIN* and *DATAIN* should maintain the same relationship throughout the test.
6. Wait for *bw_rdy_o* to be asserted. Comparison is enabled only after this is asserted.
7. *tstmode_err_o* is asserted if data mismatch is encountered.

5. IP Generation and Evaluation

This section provides information on how to generate the Lattice OpenLDI/FPD-LINK/LVDS Receiver 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 Receiver 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 Receiver 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/CrossLinkPlus IP for your design.

You may download and generate the OpenLDI/FPD-LINK/LVDS Receiver 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 Receiver 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 Receiver Interface IP is available for download from the Lattice IP Server using the Lattice 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.

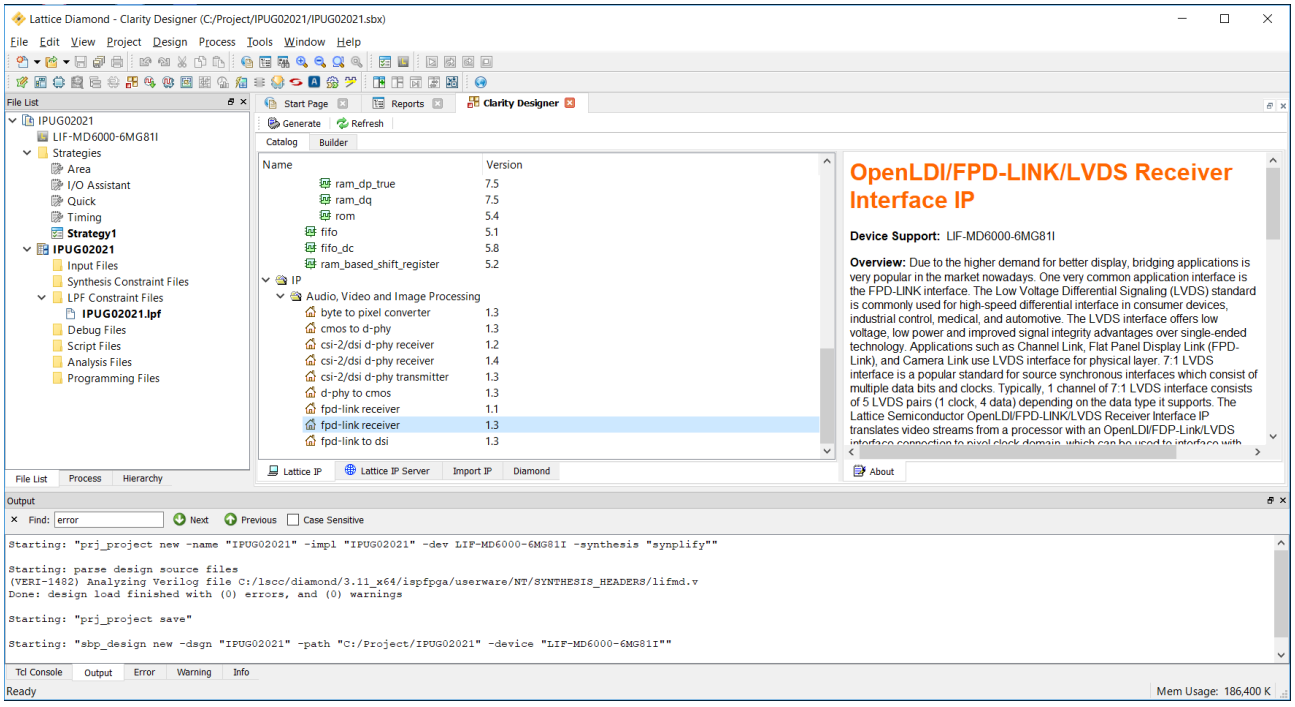

Figure 5.1. Clarity Designer Window

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 Receiver Interface IP in Clarity Designer is described below. Clarity Designer can be started from the Lattice Diamond design environment.

To start Clarity Designer:

1. Create a new Diamond project for CrossLink or CrosslinkPlus family devices.
2. From the 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 Receiver 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.

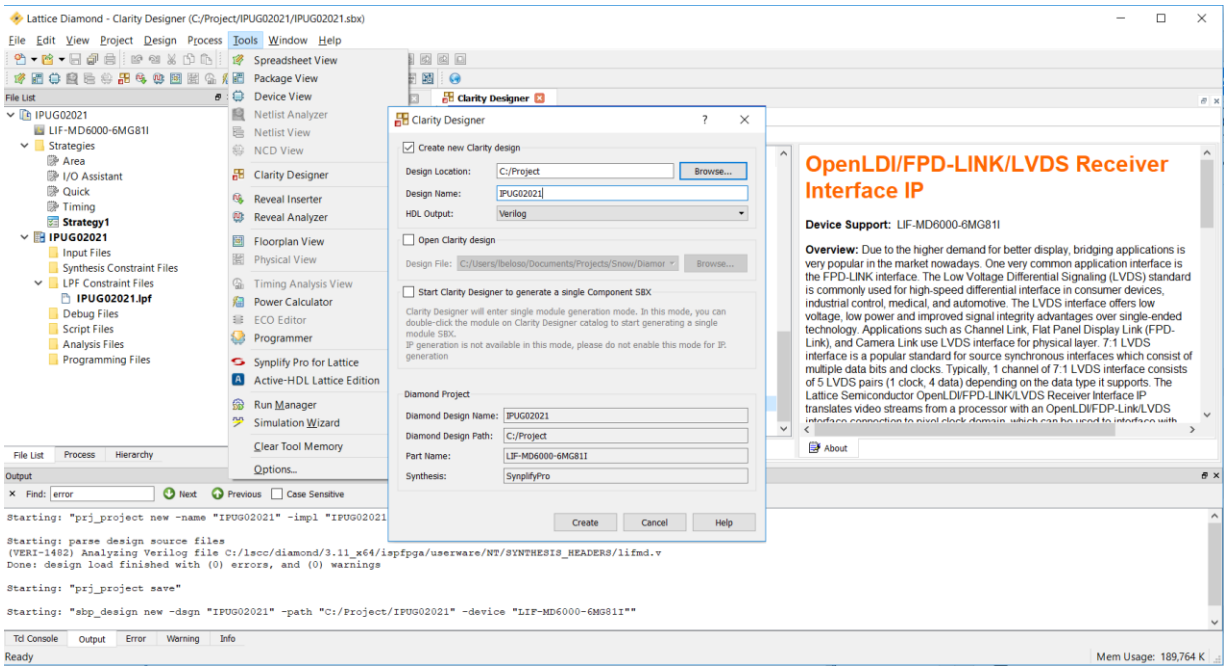

Figure 5.2. Starting Clarity Designer from Diamond Design Environment

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

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

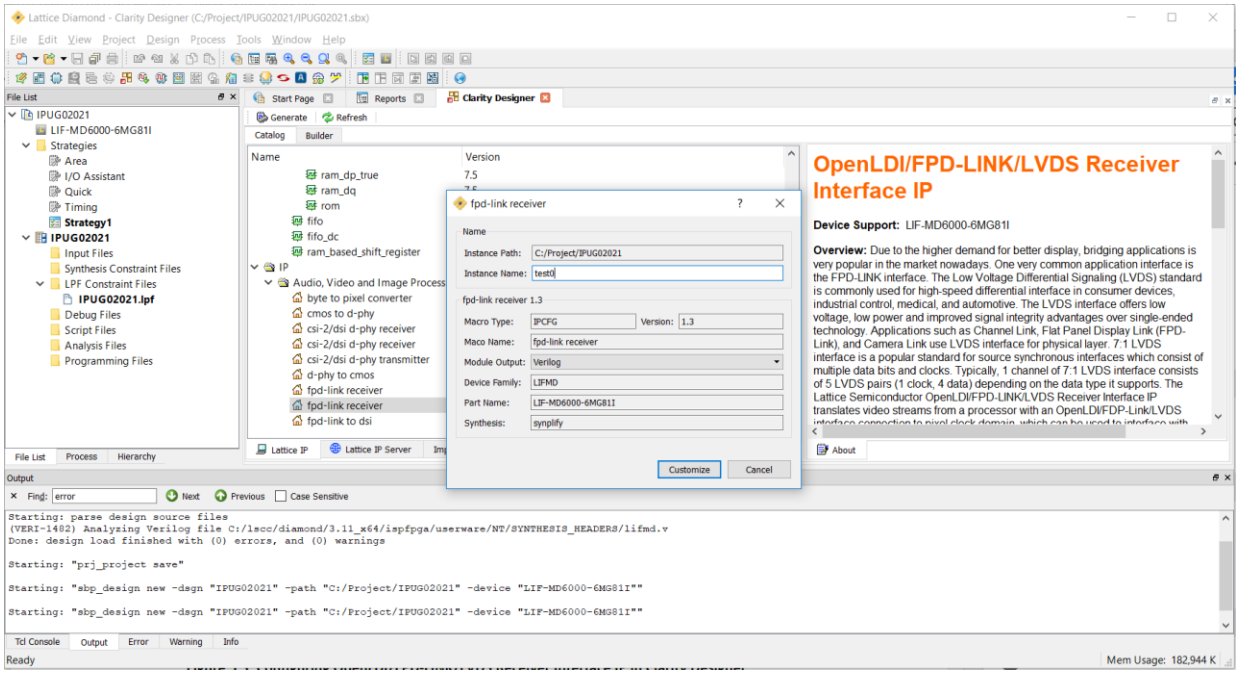

Figure 5.3. Configuring OpenLDI/FPD-LINK/LVDS Receiver Interface IP in Clarity Designer

2. Enter the Instance Name.
3. Click the **Customize** button. An IP configuration user 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.

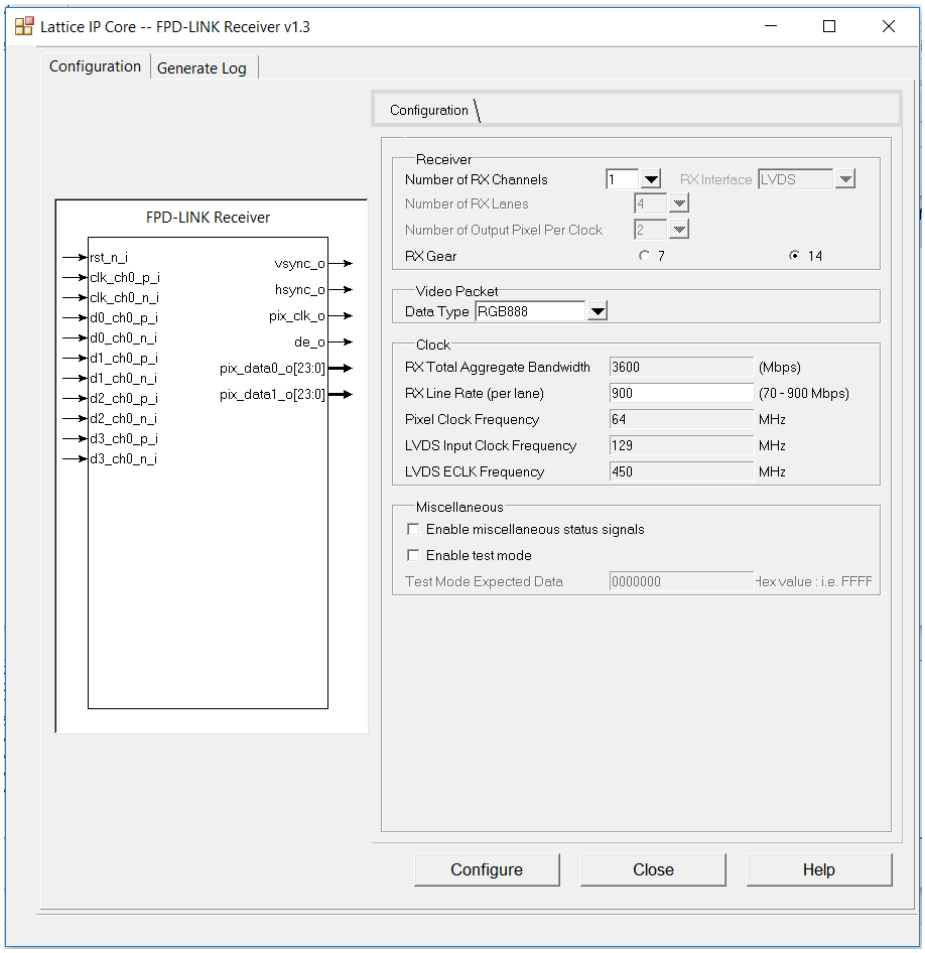

Figure 5.4. Configuration Tab in IP Interface

5.4. Generated IP Directory Structure and Files

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

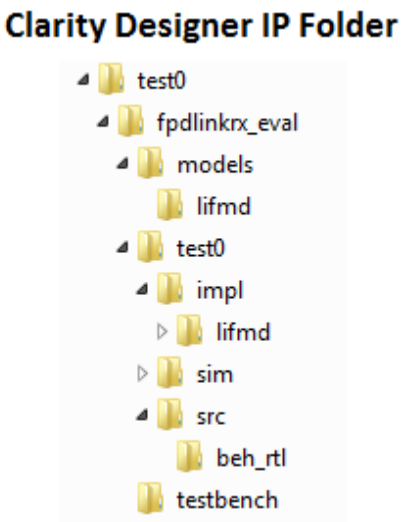

Figure 5.5. OpenLDI/FPD-LINK/LVDS Receiver Interface IP Directory Structure

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 Receiver 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 Receiver Interface IP are available under the directory *<fpdlinkrx_eval>*, including the simulation model. Lattice Diamond project files are also included under the folder at *<fpdlinkrx_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 *fpdlinkrx_eval*, and subdirectories provide files supporting OpenLDI/FPD-LINK/LVDS Receiver Interface IP evaluation that includes files/folders with content that is constant for all configurations of the OpenLDI/FPD-LINK/LVDS Receiver Interface IP. The *fpdlinkrx_eval* directory is created by Clarity Designer the first time the IP is generated, when multiple OpenLDI/FPD-LINK/LVDS Receiver 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>**fpdlinkrx_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>\fpdlinkrx_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>\fpdlinkrx_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 testbench 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 Rx functionality. Figure 5.6 shows the block diagram of simulation environment.

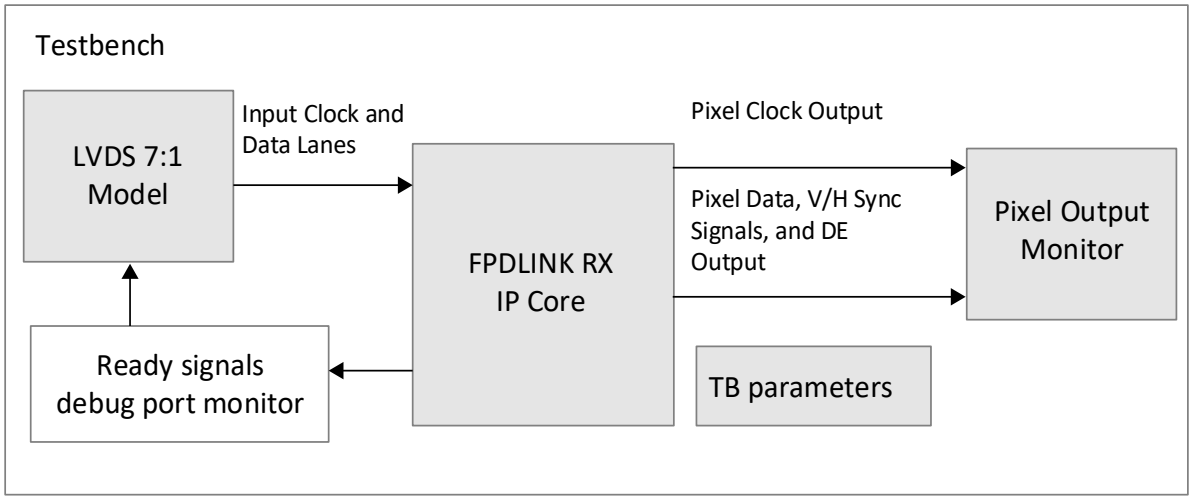

Figure 5.6. Simulation Environment Block Diagram

5.7. Simulation Environment

The simulation environment is made up of an LVDS 7:1 model instance connected to the input of FPD-Link Rx IP core instance in the testbench. The LVDS 7:1 model is configured based on the FPD-Link Rx IP configurations and testbench parameters. The testbench can be configured as one or two Rx channels.

If miscellaneous signals, *such as pll_lock_o*, *gddr_rdy_o*, *bit_lock_o*, and *word_lock_o*, *bw_rdy_o* debug ports are included in the IP core, the testbench monitors assertion of these signals before sending the DSI video data. If miscellaneous signals are disabled, you should set initial driving delay before the testbench starts sending out data to the IP core.

Input LVDS 7:1 data and output pixel 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 Rx channels configuration.

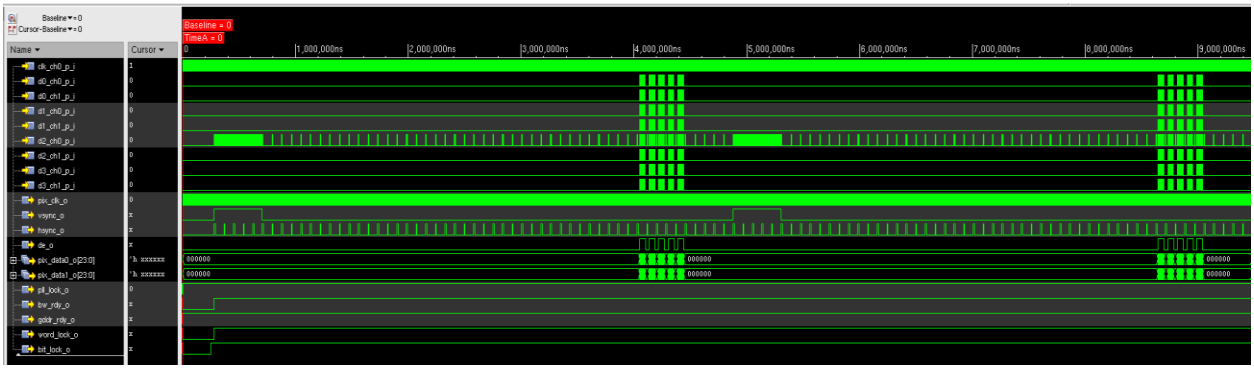

Figure 5.7. Two Rx Channels Configuration

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

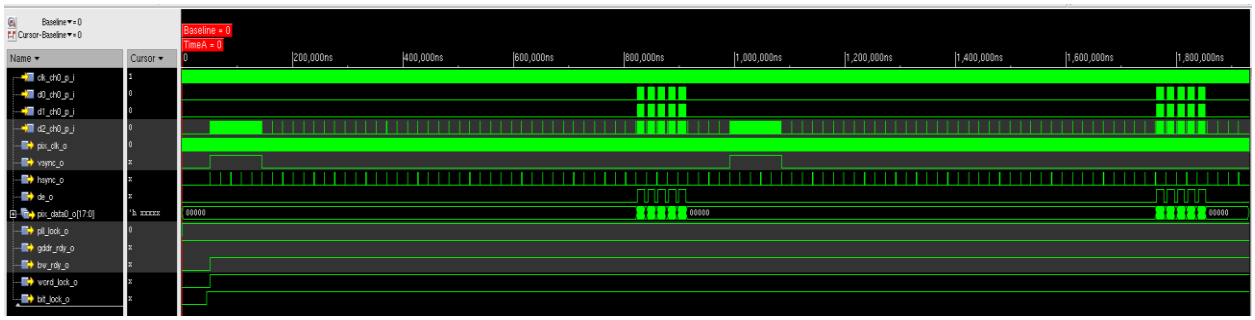

Figure 5.8. One Rx Channel Configuration

5.8. Instantiating the IP

The core modules of the OpenLDI/FPD-LINK/LVDS Receiver 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_rx.v* instantiates the black box of core modules. The top-level file *<instance_name>.v* instantiates *<instance_name>_fpd_link_rx.v* and PLL wrapper.

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*>*fpdlinkrx_eval*\<*instance_name*>*impl*\<*device_family*>*lse* or \<*project_dir*>*fpdlinkrx_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*>*fpdlinkrx_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*>\fpdlinkrx_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 Receiver 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 user 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.

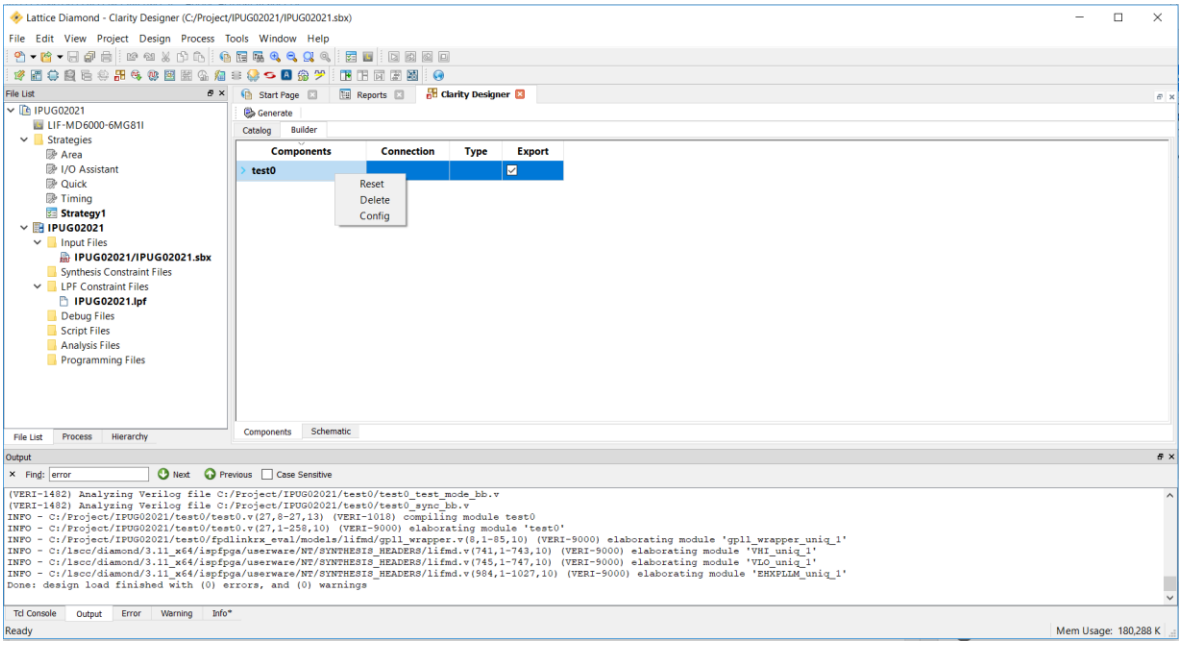

Figure 5.9. IP Regeneration in Clarity Designer

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

Clarity Designer is the Lattice IP configuration utility, and is included as a standard feature of the 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

IP User-Configurable Parameters	Slices	LUTs	Registers	sysMEM EBRs
1x7 Rx, RGB888	154	233	119	0
2x7 Rx, RGB888	168	237	143	0
1x14 Rx, RGB888	188	318	176	0
2x14 Rx, RGB888	221	390	248	0

Note: 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 Receiver Interface IP.

Appendix B. What is Not Supported

The IP does not support configuration through registers.

The IP has the following limitations:

- Odd multiple number of pixels is not supported when Rx Gear 14 is used.
- For Dual Channel configuration, only the clock from Channel 0 is used.

Revision History

Revision 1.4, IP Version 1.3, April 2024

Section	Change Summary
Disclaimers	Updated disclaimers.
Functional Descriptions	In Clock Domains and Clock Domain Crossing section: • Minor editorial changes. • Added general formula to calculate the Rx line rate and clocks of the IP and example calculations.

Revision 1.3, IP Version 1.3, March 2020

Section	Change Summary
Introduction	• Changed IP design version from 1.x to 1.3. • Updated feature to Supports interfacing up to 7.2 Gb/s • Removed footnote 2 from Table 1.2.
Functional Description	• Updated example for computing the required clocks of the IP in the Clock Domains and Clock Domain Crossing section. • Minor adjustments in style and formatting.
Compiler Directives and Parameter Settings	Updated the Rx Gear and the Rx Total Aggregate Bandwidth descriptions in Table 3.2.
IP Generation and Evaluation	Updated user interface screenshots: • Figure 5.1. Clarity Designer Window • Figure 5.2. Starting Clarity Designer from Diamond Design Environment • Figure 5.3. Configuring OpenLDI/FPD-LINK/LVDS Receiver Interface IP in Clarity Designer • Figure 5.4. Configuration Tab in IP Interface • Figure 5.9. IP Regeneration in Clarity Designer
Appendix A. Resource Utilization	Removed Target fMAX (MHz) values and footnote 2 from Table A.1. Resource Utilization.

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