CrossLink-NX-33 and CrossLinkU-NX High-Speed I/O Interface

Preliminary Technical Note

FPGA-TN-02280-0.84

March 2024

Disclaimers

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CA	Command and Address
CLKDIV	Edge Clock Dividers
DDR	Double Data Rate
DLL	Delay-Locked Loops
DM	Data Mask
DMI	Data Mask Inversion
DSP	Digital Signal Processing
IDDR	Input DDR
ECLK	Edge Clock
HP	High Performance
LLC	Lower Left Corner
LRC	Lower Right Corner
ODDR	Output DDR
ODT	On-Die Termination
PCB	Printed Circuit Board
PCLK	Primary Clock
PLC	Programmable Logic Cell
PIC	Programmable I/O Cell
PIO	Programmable I/O
RX	Receiver
SCLK	System Clock
SDR	Single Data Rate
SSN	Simultaneous Switching Noise
TX	Transmitter
WR	Wide Range

1. Introduction

Lattice Semiconductor CrossLink™-NX-33 and CrossLinkU™-NX devices support high-speed I/O interfaces, including Double Data Rate (DDR) and Single Data Rate (SDR) interfaces, using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance.

This document discusses how to utilize the capabilities of the LIFCL-33 devices to implement generic high-speed interfaces.

2. External Interface Description

This technical note uses two types of external interface definitions, centered and aligned.

- A centered external interface means that, at the device pins, the clock is centered in the data opening.
- An aligned external interface means that, at the device pins, the clock and data transition are aligned. This is sometimes called edge-on-edge.

Figure 2.1 shows the external interface waveform for SDR and DDR.

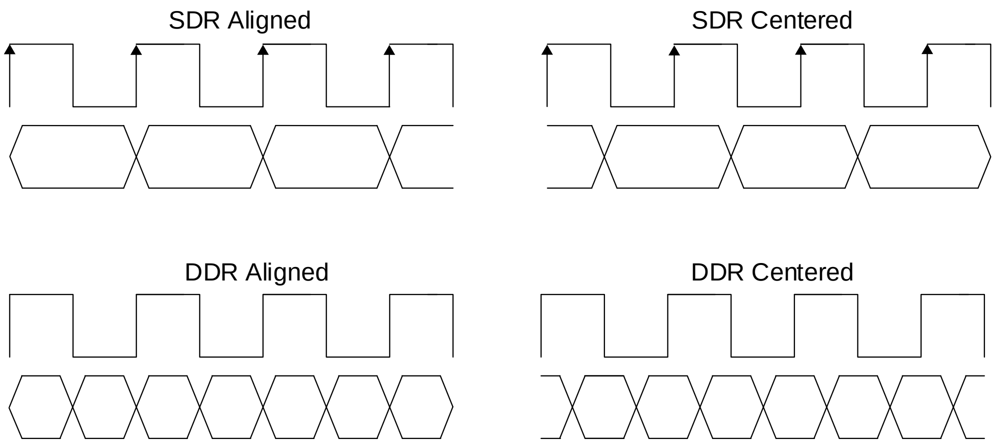


Figure 2.1. External Interface Definitions

The interfaces described are referenced as centered or aligned interfaces.

- An aligned interface needs to adjust the clock location to satisfy the capture flip-flop setup and hold times.
- A centered interface needs to balance the clock and data delay to the first flip-flop to maintain the setup and hold already provided.

3. High-Speed I/O Interface Building Blocks

The LIFCL-33/33U devices contain dedicated functions for building high-speed interfaces. This section describes when and how to use these functions. A complete description of the library elements, including descriptions and attributes, is provided at the end of this document.

Figure 3.1 shows a high-level diagram of the clocking resources available in the LIFCL-33/33U devices for building high-speed I/O interfaces.

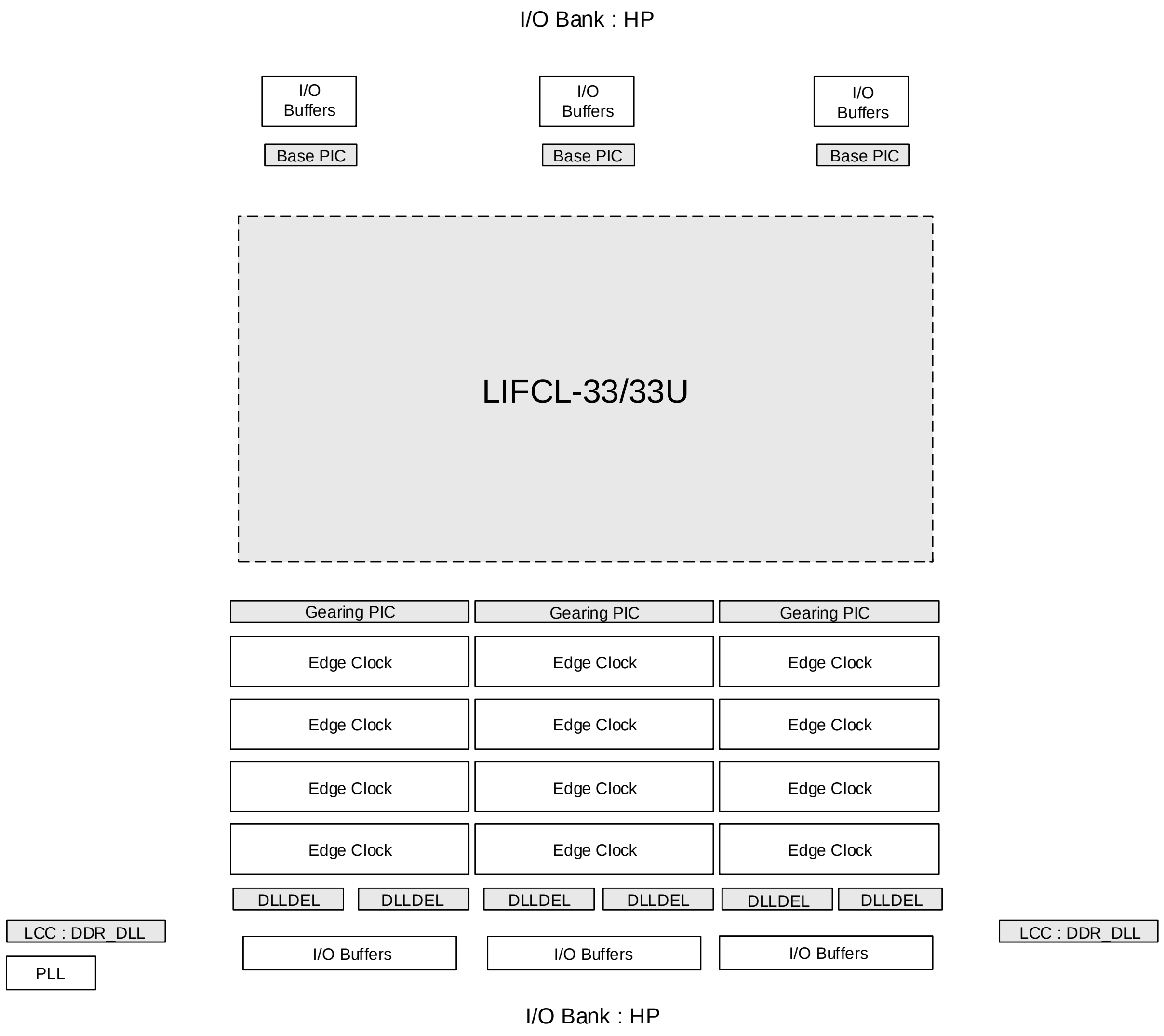


Figure 3.1. LIFCL-33/33U Device Clocking Diagram

In Figure 3.1, the locations of the Base PIC, Gearing PIC, DDRDLL, and DLLDEL are shown. One DLLDEL is provided for each of the four Edge Clocks. Two DDRDLLs are provided, one at the lower left corner and the other at the lower right corner. WR I/O bank uses Base PICs and HP I/O bank uses Gearing PICs.

3.1. I/O Banks

There are six I/O banks in all LIFCL-33 devices. Bank 0, Bank 1, and Bank 5 are located on the top side of the device while Bank 2, Bank 3, and Bank 4 are located on the bottom side of the device.

Table 3.1. LIFCL-33 I/O Banks

Bank ID	I/O Reference	Comments
BANK 0	3.3 V/1.8 V/1.2 V	WR[1]
BANK 1	3.3 V/1.8 V/1.2 V	WR[1]
BANK 2	3.3 V/1.8 V/1.2 V	WR[1]
BANK 3	1.8 V/1.2 V/1.0 V	HP
BANK 4	1.8 V/1.2 V/1.0 V	HP
BANK 5	1.8 V/1.2 V/1.0 V	WR[1]

Note:

1. WR can be used as emulated differential output port.

There are five I/O banks in all LIFCL-33U devices. Bank 0 and Bank 1 are located on the top side of the device while Bank 2, Bank 3, and Bank 4 are located on the bottom side of the device.

Table 3.2. LIFCL-33U I/O Banks

Bank ID	I/O Reference	Comments
BANK 0	3.3 V/1.8 V/1.2 V	WR[1]
BANK 1	3.3 V/1.8 V/1.2 V	WR[1]
BANK 2	1.8 V/1.2 V/1.0 V	HP
BANK 3	1.8 V/1.2 V/1.0 V	HP
BANK 4	1.8 V/1.2 V/1.0 V	HP

Note:

1. WR can be used as emulated differential output port.

3.2. Clock Scheme

A complete description of the LIFCL-33/33U device families clocking resources and clock routing restrictions are available in sysCLOCK PLL Design and Usage Guide for Nexus Platform (FPGA-TN-02095).

Below is a brief description of each of the major elements used for building various high-speed interfaces. The I/O Logic (DDR) User Primitives and Attributes section of this document describes the library elements for these components.

3.2.1. Primary Clocks

Primary Clocks (PCLK) refer to the system clocks of the design. The System Clock (SCLK) ports of the DDR primitives are connected to the system clocks of the design.

3.2.2. Edge Clocks

Edge Clocks (ECLK) are high-speed, low-skew I/O dedicated clocks. They are arranged in groups of four per I/O bank on the bottom side of the device. Each of these edge clocks can be used to implement a high-speed interface.

3.3. PLL

The PLL provides frequency synthesis, with additional static and dynamic phase adjustment, as well. Six output ports are provided, CLKOP, CLKOS, CLKOS2, CLKOS3, CLKOS4, and CLKOS5. All six outputs have the same set of dividers. For LIFCL-33/33U devices, PLL is located in the lower left corner.

3.4. DDRDLL

The DDRDLL is a dedicated DLL for creating the 90-degree clock delay. The DDRDLL outputs delay codes are used in the DLLDEL module to delay the input clock. There are two DDRDLL at bottom corners of the LIFCL-33 and the LIFCL-33U devices. The DDRDLL on the bottom corners of the device can drive delay codes to two adjacent edges of the device, providing possible two DDRDLL codes for an edge.

3.5. DLLDEL

DLLDEL provides phase shift on the receive side clocks to each SCLK or ECLK (Figure 3.2), shifting the clock input by the delay set by the DDRDLL delay code, before the clock drives the clock tree. The DLLDEL element has the ability to further adjust the delay from the delay set by the DDRDLL code for margin test purpose. Control signal *LOADN* (Figure 3.3) asynchronous resets the delay setting to original (DLL delay control code + fuse adjustment). Control pulse *MOVE* changes the delay setting by +1/-1 tap each time according to the *DIRECTION* value. *0* means increasing the setting by 1 tap, *1* means decreasing the setting by 1 tap. When the delay setting reaches the minimum values *0* or maximum value *255* (8 bits control), the cell generates COUT flag to indicate the under/over flow situation and the delay setting does not roll-over even if the *MOVE* pulse is still coming in.

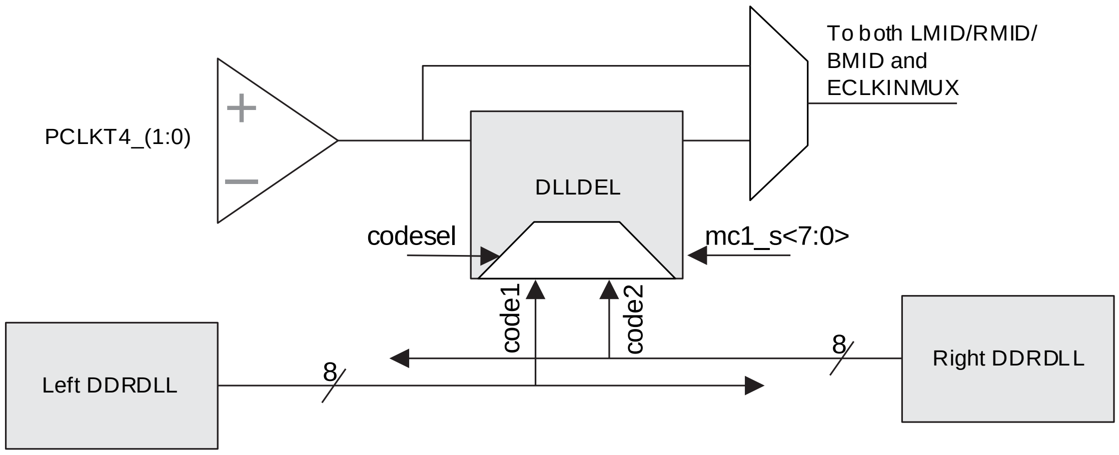


Figure 3.2. DLLDEL and Code Control

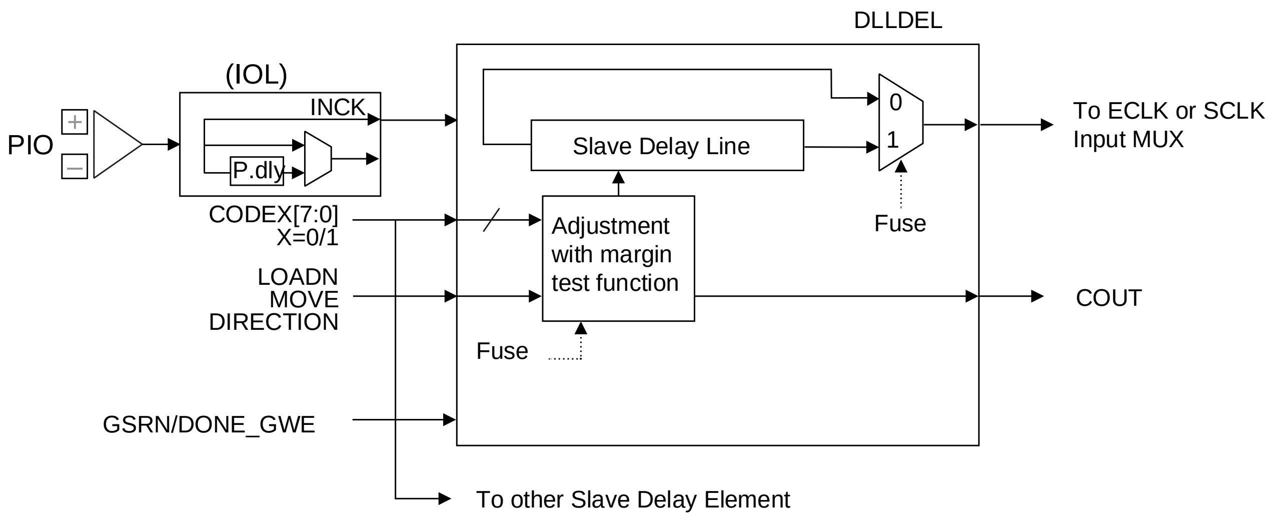


Figure 3.3. DDR Delay Block Diagram

3.6. Input DDR (IDDR)

The input DDR function can be used in the 1× (2:1), 2× (4:1), 4× (8:1), 7:1, and 5× (10:1) gearing modes. In the 1× gearing mode, 1x receives 1-bit DDR data and outputs 2-bit wide parallel data synchronized to the SCLK. There is no clock domain transfer involved in the 1x gearing.

The 2× receives 1-bit DDR data synchronized to the ECLK and outputs four bits of parallel data synchronized to the SCLK. The same function applies to the 4×, which receives a single bit of DDR data synchronized to the ECLK and outputs eight bits of parallel data synchronized to the SCLK. The 7:1 gearing shares the same structure as the 4× gearing. The 7:1 gearing outputs seven bits of parallel data instead of eight.

The 5× receives 1-bit DDR data synchronized to the ECLK and outputs ten bits of parallel data synchronized to the SCLK

3.7. Output DDR (ODDR)

The output DDR function can also be supported in 1× (2:1), 2× (4:1), 4× (8:1), 7:1, and 5× (10:1) gearing modes. In the 1× gearing mode, the ODDR element receives 2-bit wide data from the FPGA fabric and generates a single DDR data output or clock output.

Similar to input interfaces, the 2×/4×/5× gearing mode is used for data rate higher than 500/1000 Mbps, which requires higher than 250/500 MHz system clock. Here, the ODDR element receives 4-bit/8-bit/10-bit wide data from the FPGA fabric and generates a single DDR data output or clock output. The 2×/4×/5× element uses high-speed edge clock (ECLK) to clock the data out for generic high-speed interfaces.

In 7:1 gearing mode, the ODDR element receives 7-bit wide data from the FPGA fabric and generates a single DDR data output or clock output. The 7:1 element sends out data using high-speed edge clock.

3.8. Edge Clock Dividers (CLKDIV)

Clock dividers are provided to create the divided down clocks used with the I/O Mux/Demux gearing logic (SCLK inputs to the DDR) and drives to the Primary Clock routing to the fabric. There are twelve CLKDIV per device, locating near the bottom high-speed I/O banks.

3.9. Input/Output DELAY

Similar to using the DLLDEL to tune the clock delay path, there are delay cells (DELAYB and DELAYA) in the PIC that can be utilized to tune the data path (Figure 3.4). DELAYB is static and DELAYA is dynamic, providing a data delay to compensate for clock injection delay. The margin test capability is similar to the one used in DLLDEL. See the DLLDEL section for more details. The DELAYA element also allows you to set the delay value using 128 steps of delay. Each delay step generates ~12.5 ps of delay. You can overwrite the delay setting dynamically using the MOVE and DIRECTION control inputs. The LOADN resets the delay back to the default value.

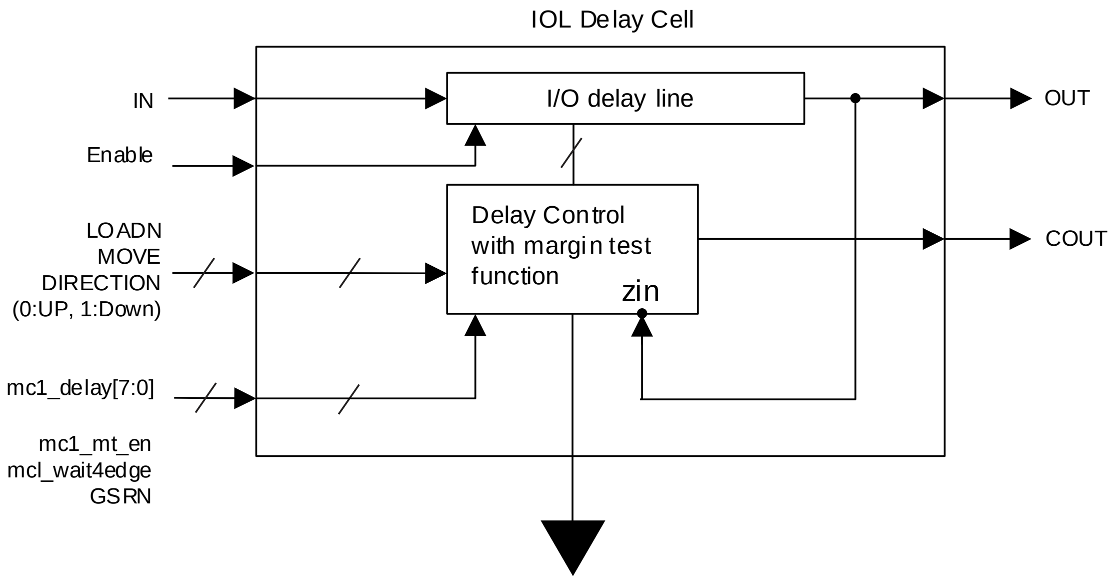


Figure 3.4. PIC Delay Cell Diagram

4. Building Generic High Speed Interfaces

This section describes in detail the high-speed interfaces that can be built using the building blocks described in the section above. The IP Catalog tool in Lattice Radiant™ design software builds these interfaces based on external interface requirements.

4.1. Types of High-Speed I/O Interfaces

This section describes the types of high-speed I/O interfaces available in the LIFCL-33/33U devices.

Table 4.1 lists these interfaces. The naming conventions used for each interface are provided below the table.

Table 4.1. Generic High-Speed I/O Interfaces

Mode	Interface Name	Description
Receive SDR	GIREG_RX.SCLK	SDR Input register using SCLK.
RX DDRX1 Aligned	GDDRX1_RX.SCLK.Aligned	DDR 1x Input using SCLK. Data is edge-to-edge with incoming clock. DLLDEL is used to shift the incoming clock.
RX DDRX1 Centered	GDDRX1_RX.SCLK.Centered	DDR x1 Input using SCLK. Clock is already centered in data window.
RX DDRX2 Aligned	GDDRX2_RX.ECLK.Aligned	DDR x2 Input using ECLK. Data is edge-to-edge with incoming clock. Generic DDR X2 using Edge Clock. DLLDEL is used to shift the incoming clock.
RX DDRX2 Centered	GDDRX2_RX.ECLK.Centered	DDR x2 Input using ECLK. Clock is already centered in data window.
RX DDRX4 Aligned	GDDRX4_RX.ECLK.Aligned	DDR x4 Input using ECLK. Data is edge-to-edge with incoming clock. Generic DDR X4 using Edge Clock. DLLDEL is used to shift the incoming clock.
RX DDRX4 Centered	GDDRX4_RX.ECLK.Centered	DDR x4 Input using ECLK. Clock is already centered in data window.
RX DDRX5 Aligned	GDDRX5_RX.ECLK.Aligned	DDR x5 Input using ECLK. Data is edge-to-edge with incoming clock. Generic DDR X5 using Edge Clock. DLLDEL is used to shift the incoming clock.
RX DDRX5 Centered	GDDRX5_RX.ECLK.Centered	DDR x5 Input using ECLK. Clock is already centered in data window.
RX DDRX71	GDDRX71_RX.ECLK	DDR 7:1 input using ECLK.
RX DDRX4 MIPI	GDDRX4_RX.MIPI	DDRx4 Input using ECLK to MIPI interface.
Transmit SDR	GOREG_TX.SCLK	SDR Output using SCLK. Clock is forwarded through ODDR.
TX DDRX1 Aligned	GDDRX1_TX.SCLK.Aligned	DDR x1 Output using SCLK. Data is edge-on-edge using same clock through ODDR.
TX DDRX1 Centered	GDDRX1_TX.SCLK.Centered	DDR x1 Output using SCLK. Clock is centered using PLL with different SCLK.
TX DDRX2 Aligned	GDDRX2_TX.ECLK.Aligned	DDR x2 Output that is edge-on-edge using ECLK.
TX DDRX2 Centered	GDDRX2_TX.ECLK.Centered	DDR x2 Output that is pre-centered PLL generated 90-degree phase, and output on ECLKs.
TX DDRX4 Aligned	GDDRX4_TX.ECLK.Aligned	DDR x4 Output that is edge-on-edge using ECLK.
TX DDRX4 Centered	GDDRX4_TX.ECLK.Centered	DDR x4 Output that is pre-centered PLL generated 90-degree phase, and output on ECLKs.
TX DDRX5 Aligned	GDDRX5_TX.ECLK.Aligned	DDR x5 Output that is edge-on-edge using ECLK.
TX DDRX5 Centered	GDDRX5_TX.ECLK.Centered	DDR x5 Output that is pre-centered PLL generated 90-degree phase, and output on ECLKs.
TX DDRX71	GDDRX71_TX.ECLK	DDR 7:1 output using ECLK.
TX DDRX4 MIPI	GDDRX4_TX.MIPI	DDRx4 Output using ECLK to MIPI interface.

The following describes the naming conventions used for each of the interfaces listed in Table 4.1:

- G – Generic
- IREG – SDR Input I/O Register
- OREG – SDR Output I/O Register
- DDRX1 – DDR 1x gearing I/O Register
- DDRX2 – DDR 2x gearing I/O Registers
- _RX – Receive Interface
- _TX – Transmit Interface
- .ECLK – Uses ECLK (Edge Clock) clocking resource
- .SCLK – Uses SCLK (Primary Clock) clocking resource
- .Centered – Clock is centered to the data when coming into the device

5. High-Speed DDR Interface Details

This section describes each of the generic high-speed interfaces in detail, including the clocking to be used for each interface. For detailed information about the LIFCL-33/33U devices clocking structure, refer to sysCLOCK PLL Design and Usage Guide for Nexus Platform (FPGA-TN-02095). The various interface rules listed under each interface should be followed to build these interfaces successfully.

Some of these interfaces may require a soft IP in order utilize all the features available in the hardware. These soft IP cores are available in IP Catalog and are described in this section. Some of these are mandatory for the module to function as expected and are automatically generated when building the interface through IP Catalog.

5.1. GDDRX1_RX.SCLK.Centered

This section describes the Generic Receive interface with the X1 gearing using SCLK. The clock is coming in centered to the data. This interface must be used for speeds up to 250 MHz.

This DDR interface uses the following modules:

- IDDRX1 element is used to capture the data.
- The incoming clock is routed through the Primary (SCLK) clock tree.
- Static data delay element DELAYB is used to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.

The following figures show the static delay and dynamic delay options for this interface.

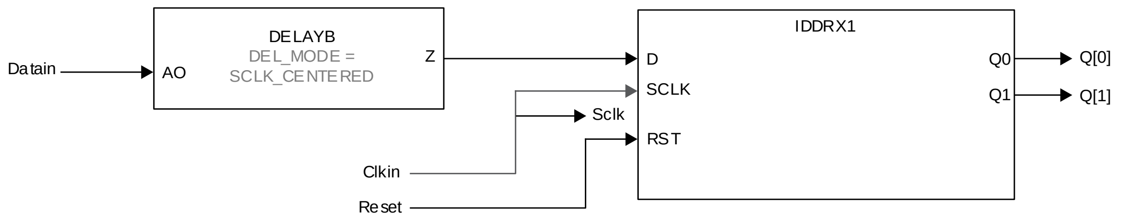


Figure 5.1. GDDRX1_RX.SCLK.Centered Interface (Static Delay)

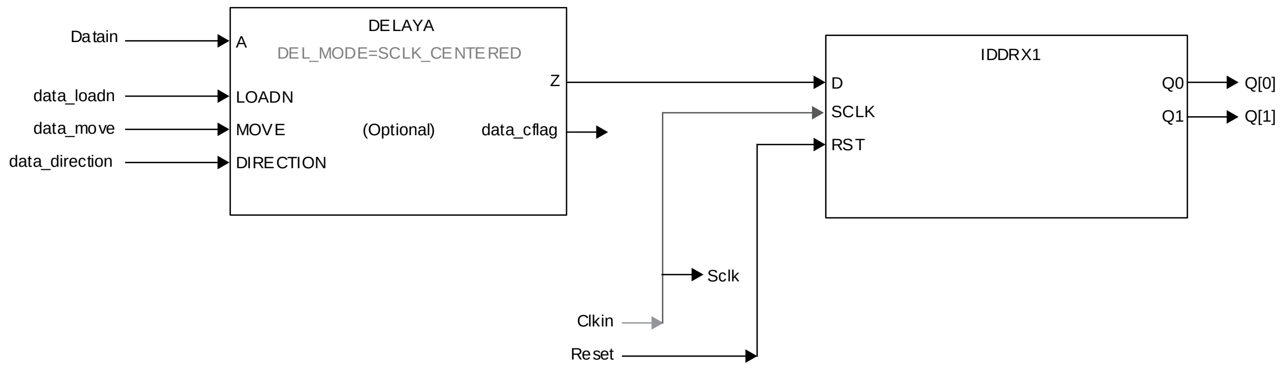


Figure 5.2. GDDRX1_RX.SCLK.Centered Interface (Dynamic Data Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the primary clock tree.

5.2. GDDRX1_RX.SCLK.Aligned

This section describes the Generic Receive interface with the X1 gearing using SCLK. The clock is coming in edge aligned to the Data. This interface must be used for speeds up to 250 MHz.

This DDR interface uses the following modules:

- IDDRX1 element to capture the data
- DDRDLL/DLLDEL blocks are used to phase shift the incoming clock going to primary clock tree (SCLK).
- Static data delay element DELAYB is used to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- Dynamic Margin adjustment in the DDRDLL module can be optionally used to adjust the DDRDLL delay dynamically.

The output of the DLLDEL module is also used as the clock input to the DDRDLL, which sends the delay values to the DLLDEL module. The Receiver Synchronization (RX_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP prevents any updates to the DLLDEL at start until the DDRDLL is locked. Once locked the DLLDEL is updated and FREEZE on the DDRDLL is removed. This soft IP is automatically generated by IP Catalog.

The following figures show the static delay and dynamic delay options for this interface.

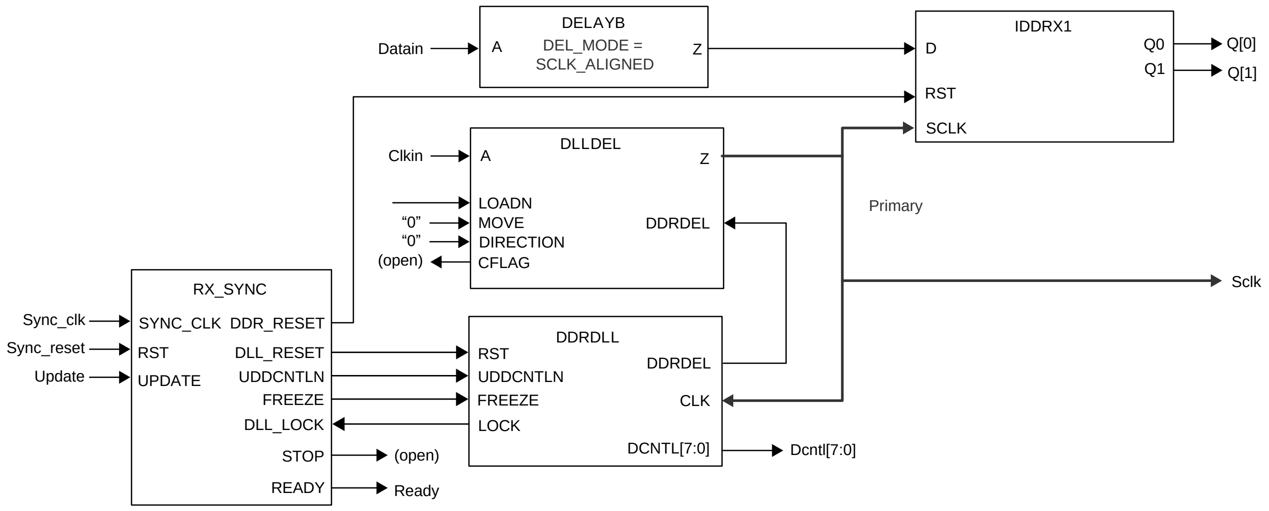


Figure 5.3. GDDRX1_RX.SCLK.Aligned Interface (Static Delay)

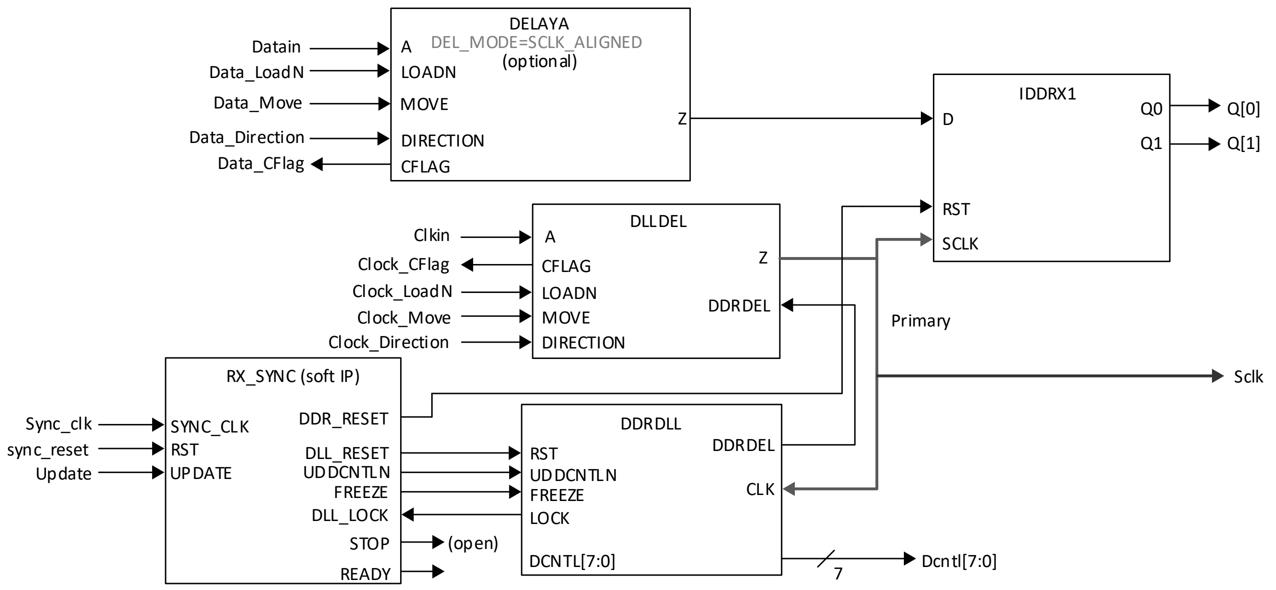


Figure 5.4. GDDRX1_RX.SCLK.Aligned Interface (Dynamic Data/Clock Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the DLLDELD input.

5.3. GDDRX2_RX.ECLK.Centered

This section describes the Generic Receive DDR with the X2 gearing using Edge Clock Tree (ECLK). Input clock is centered to the input Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX2 element for X2 mode to capture the data
- The incoming clock is routed to the ECLK tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 2 to generate the SCLK.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- ECLKSYNCB element. This element can be enabled through IP Catalog.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.

The following figures show the static delay and dynamic delay options for this interface.

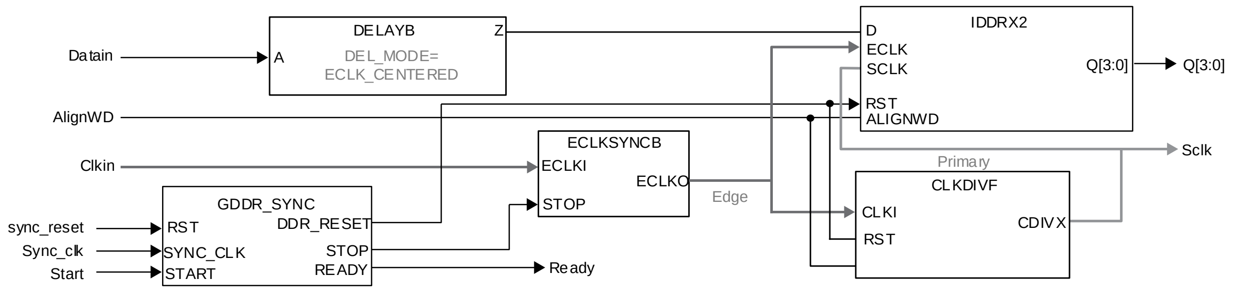


Figure 5.5. GDDRX2_RX.ECLK.Centered Interface (Static Delay)

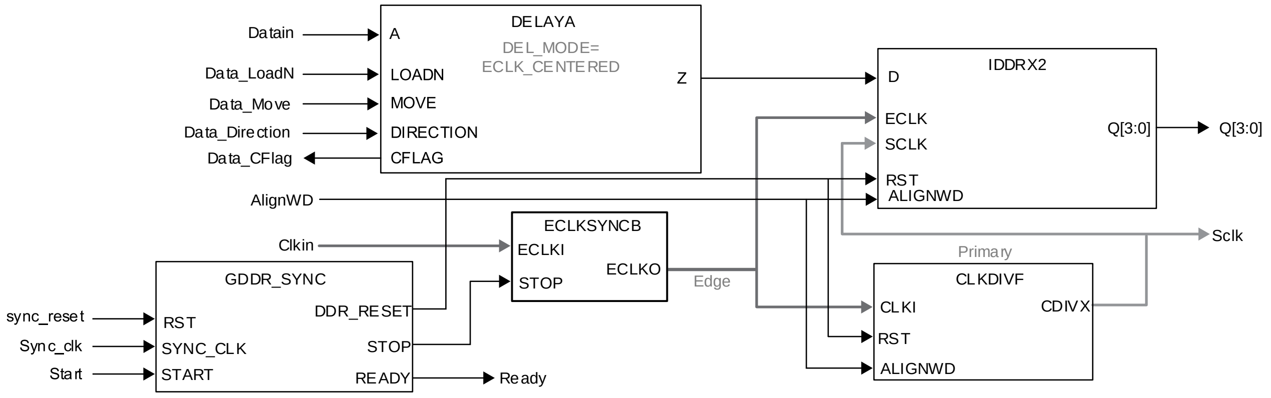


Figure 5.6. GDDRX2_RX.ECLK.Centered Interface (Dynamic Data Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the Edge Clock tree.
- ECLK must use the Edge Clock tree and the SCLK out of the CLKDIVF must use the Primary Clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.4. GDDRX2_RX.ECLK.Aligned

This section describes the Generic Receive DDR with the X2 gearing using ECLK. Input Clock is coming in edge aligned to the Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX2 element for X2 mode to capture the data
- DDRDLL/DLLDEL blocks are used to phase shift the incoming clock routed to the ECLK tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 2.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- Dynamic Margin adjustment in the DDRDLLA module can be optionally used to adjust the DDRDLLA delay dynamically.

The output of the DLLDEL module is also used as the clock input to the DDRDLL, which sends the delay values to the DLLDEL module. The Receiver Synchronization (RX_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP prevents any updates to the DLLDEL at start until the DDRDLL is locked. Once locked, the DLLDEL is updated and FREEZE on the DDRDLL is removed. This soft IP is automatically generated by IP Catalog.

The following figures show the static delay and dynamic delay options for this interface.

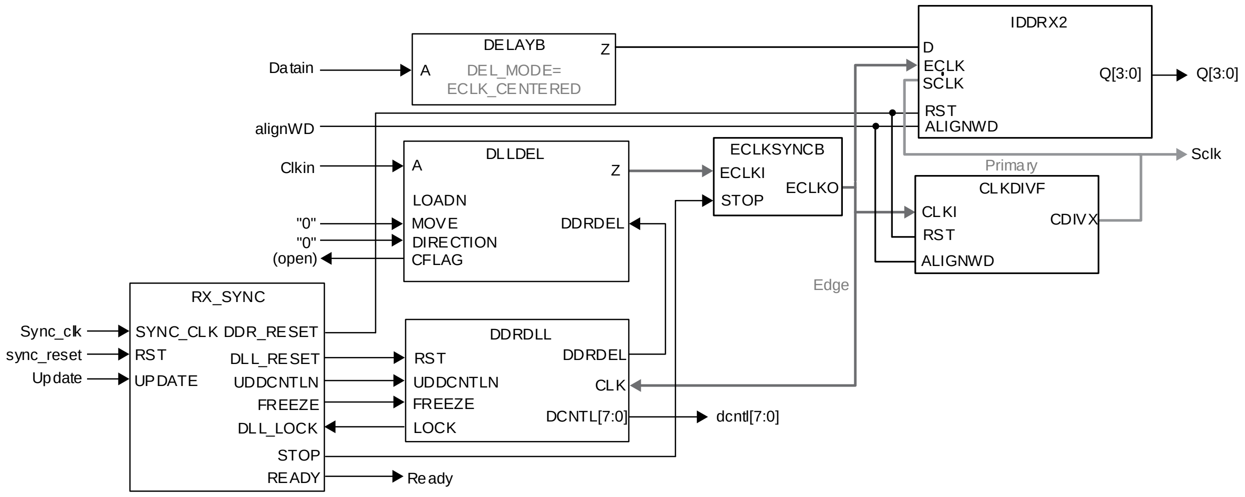


Figure 5.7. GDDRX2_RX.ECLK.Aligned Interface (Static Delay)

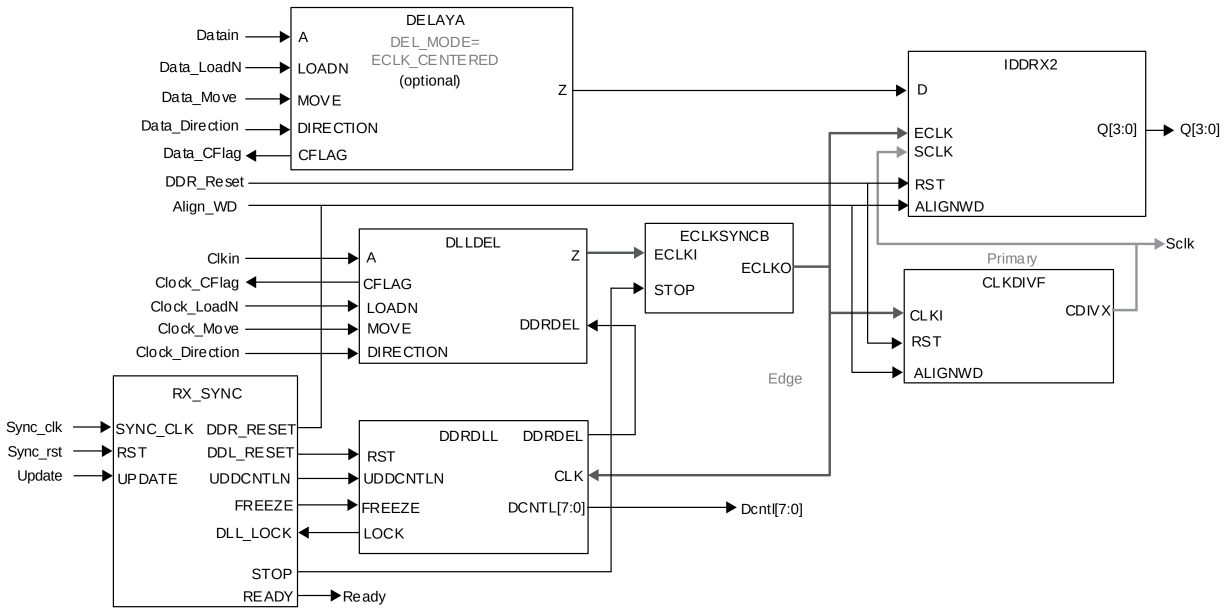


Figure 5.8. GDDRX2_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

Interface Requirements

- The clock input must use a dedicated PCLK input so that it can be routed directly to the DLLDEL module.
- ECLK must use the Edge Cock tree and the SCLK out of the CLKDIVD must use the Primary Clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.5. GDDRX4_RX.ECLK.Centered

This section describes the Generic Receive DDR with the X4 gearing using Edge Clock Tree (ECLK). Input clock is centered to the input Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX4 element for X4 mode to capture the data.
- The incoming clock is routed to the ECLK tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 4 to generate the SCLK.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.

The following figures show the static delay and dynamic delay options for this interface.

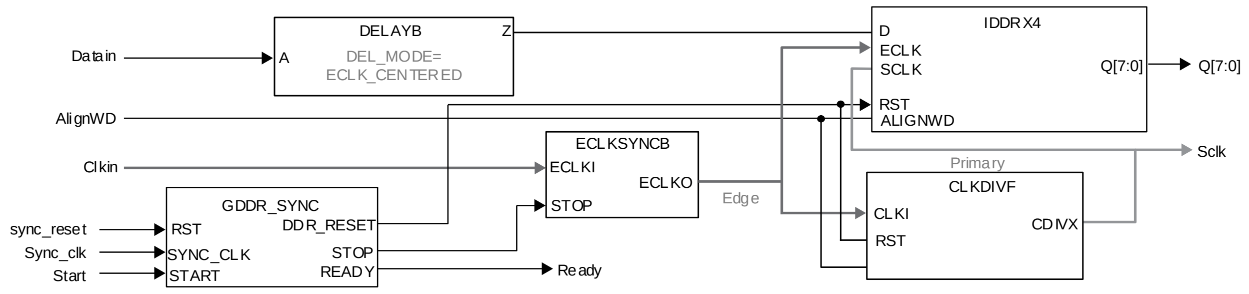


Figure 5.9. GDDRX4_RX.ECLK.Centered Interface (Static Delay)

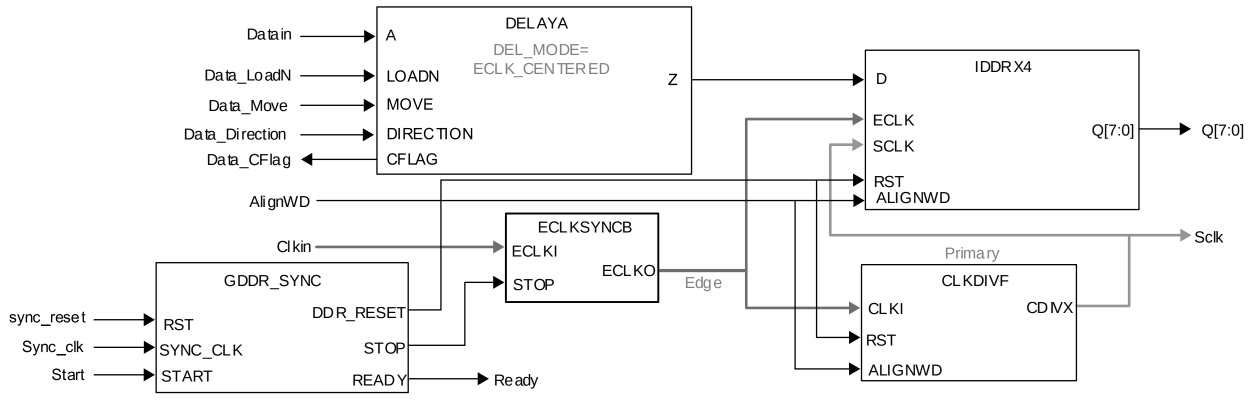


Figure 5.10. GDDRX4_RX.ECLK.Centered Interface (Dynamic Data Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the edge clock tree.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVF must use the Primary clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.6. GDDRX4_RX.ECLK.Aligned

This section describes the Generic Receive DDR with the X4 gearing using ECLK. Input Clock is coming in edge aligned to the Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX4 element for X4 mode to capture the data.
- DDRDLL/DLLDEL blocks are used to phase shift the incoming clock routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 4.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- Dynamic Margin adjustment in the DDRDLL module can be optionally used to adjust the DDRDLL delay dynamically.

The output of the DLLDEL module is also used as the clock input to the DDRDLL, which sends the delay values to the DLLDEL module. The Receiver Synchronization (RX_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP prevents any updates to the DLLDEL at start until the DDRDLL is locked. Once locked, the DLLDEL is updated and FREEZE on the DDRDLL is removed. This soft IP is automatically generated by IP Catalog.

The following figures show the static delay and dynamic delay options for this interface.

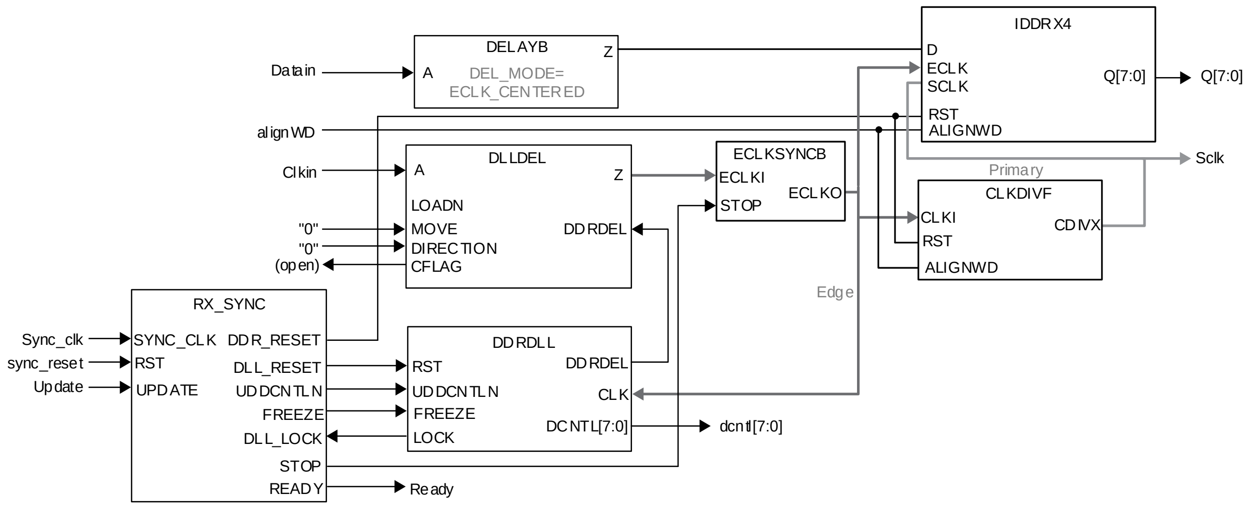


Figure 5.11. GDDRX4_RX.ECLK.Aligned Interface (Static Delay)

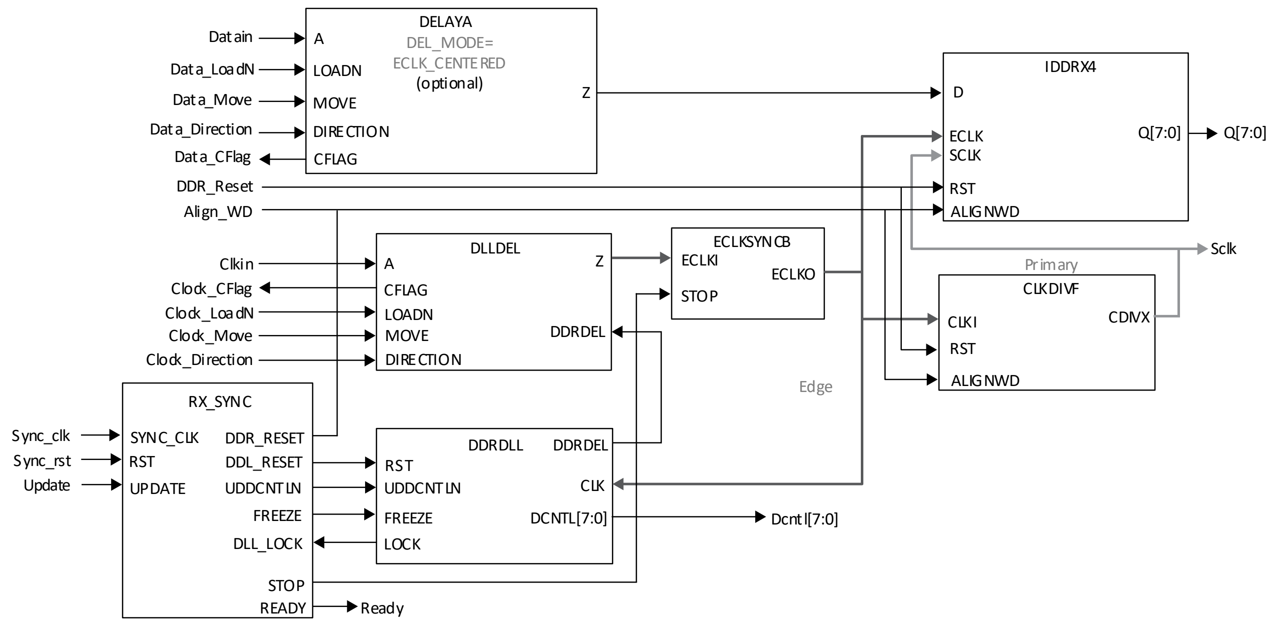


Figure 5.12. GDDRX4_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

Interface Requirements

- The clock input must use a dedicated PCLK input so that it can be routed directly to the DLLDEL module.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVD must use the Primary clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.7. GDDRX5_RX.ECLK.Centered

This section describes the Generic Receive DDR with the X5 gearing using Edge Clock Tree (ECLK). Input clock is centered to the input Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX5 element for X5 mode to capture the data.
- The incoming clock is routed to the ECLK tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 5 to generate the SCLK.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time.
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.

The following figures show the static delay and dynamic delay options for this interface.

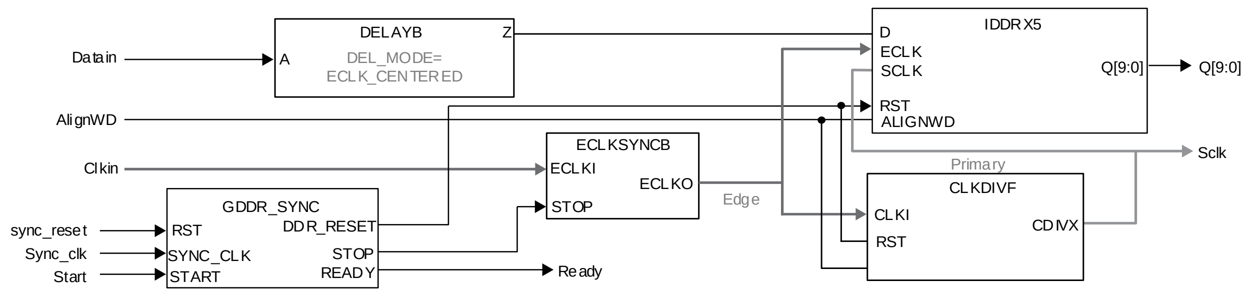


Figure 5.13. GDDRX5_RX.ECLK.Centered Interface (Static Delay)

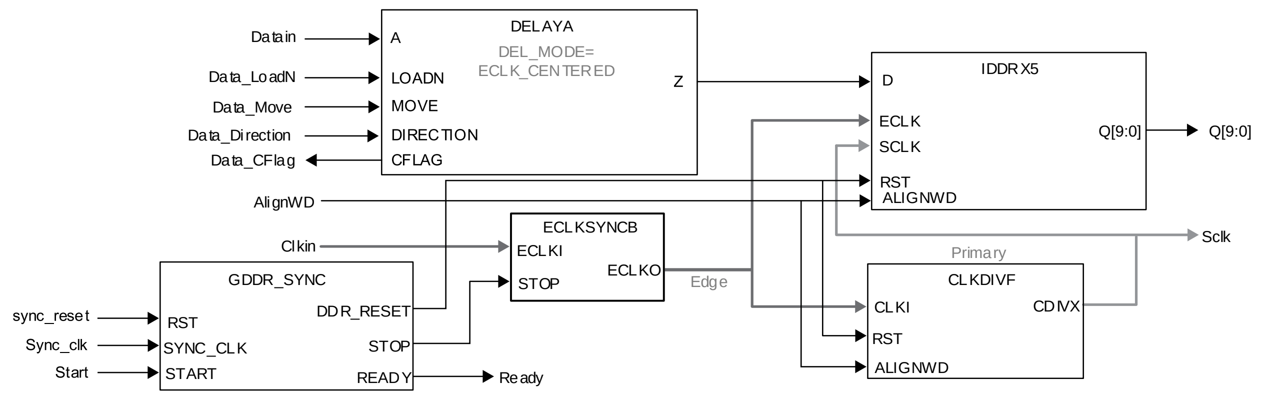


Figure 5.14. GDDRX5_RX.ECLK.Centered Interface (Dynamic Data Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the edge clock tree.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVF must use the Primary clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.8. GDDRX5_RX.ECLK.Aligned

This section describes the Generic Receive DDR with the X5 gearing using ECLK. Input Clock is coming in edge aligned to the Data. This interface must be used for speeds above 250 MHz.

This DDR interface uses the following modules:

- IDDRX5 element for X5 mode to capture the data.
- DDRDLL/DLLDEL blocks are used to phase shift the incoming clock routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the incoming clock by 5.
- Static data delay element DELAYB to delay the incoming data enough to remove the clock injection time
- Optionally, you can choose to use Dynamic Data delay adjustment using DELAYA element to control the delay on the DATA dynamically. DELAYA also allows you to override the input delay set. The type of delay required can be selected through IP Catalog.
- DEL_MODE attribute is used with DELAYB and DELAYA element to indicate the interface type so that the correct delay value can be set in the delay element.
- Dynamic Margin adjustment in the DDRDLL module can be optionally used to adjust the DDRDLL delay dynamically.

The output of the DLLDEL module is also used as the clock input to the DDRDLL, which sends the delay values to the DLLDEL module. The Receiver Synchronization (RX_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP prevents any updates to the DLLDEL at start until the DDRDLL is locked. Once locked, the DLLDEL is updated and FREEZE on the DDRDLL is removed. This soft IP is automatically generated by IP Catalog.

The following figures show the static delay and dynamic delay options for this interface.

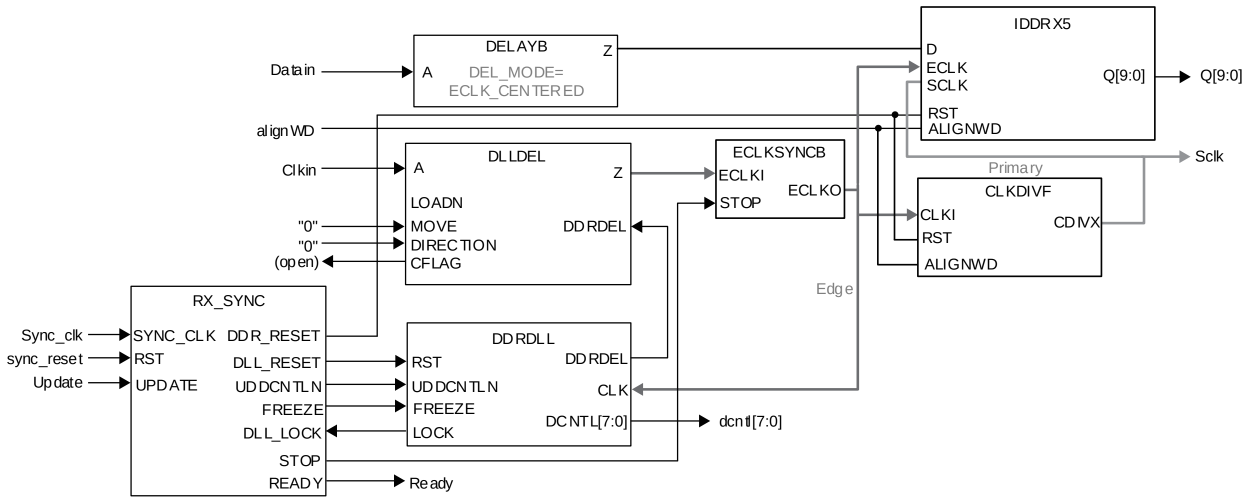


Figure 5.15. GDDRX5_RX.ECLK.Aligned Interface (Static Delay)

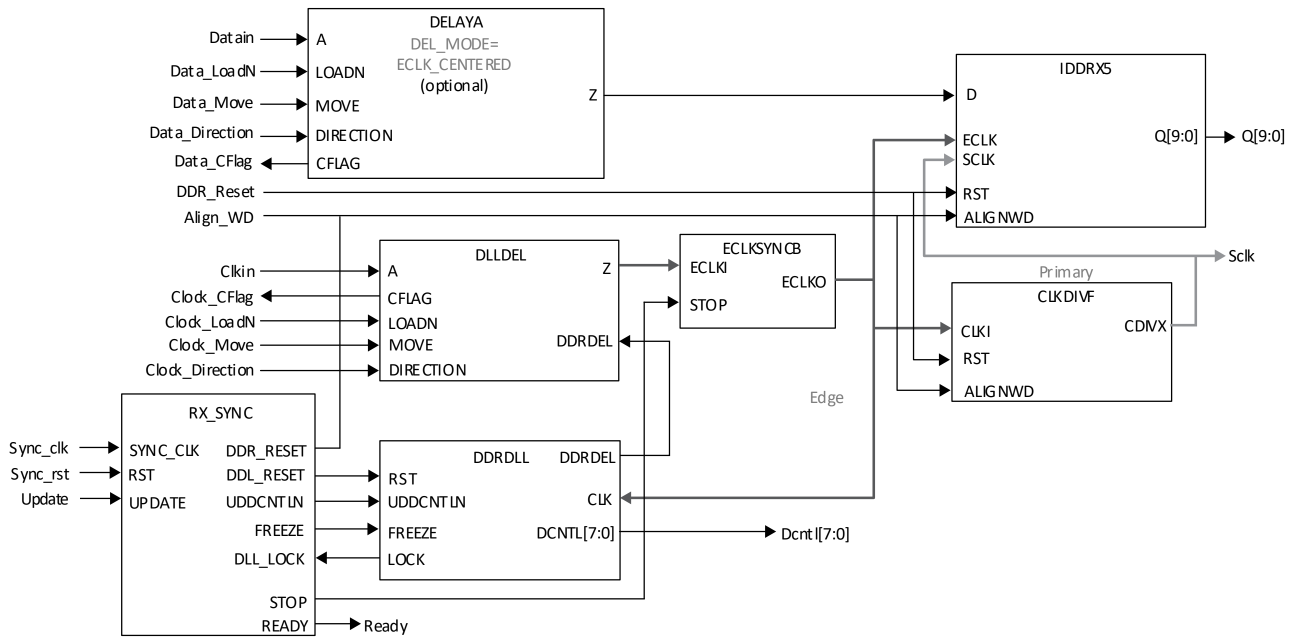


Figure 5.16. GDDRX5_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

Interface Requirements

- The clock input must use a dedicated PCLK input so that it can be routed directly to the DLLDEL module.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVD must use the Primary clock tree, software errors out if these dedicated clock routes are not used.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.9. GDDRX71_RX.ECLK

This interface is used to implement 7:1 LVDS Receiver interface using the 1 to 7 gearing with ECLK. Slow speed clock coming in is multiplied 3.5X using a PLL. This clock is used to capture the data at the receiver IDDR71 module.

This DDR interface uses the following modules:

- IDDR71 element is used to capture the data.
- EHXPLLK multiplies the input clock by 3.5 and phase shift the incoming clock based on the dynamic phase shift input.
- This clock is routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module.
- CLKDIVF module is used to divide the ECLK by 3.5 and is routed to the primary clock tree used as the SCLK input.
- A second IDDR71 element is used with data connected to clock input to generate 7-bit clock phase that can be used for word alignment.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- An optional Bit and Word alignment soft IP (BW_ALIGN) can be enabled in IP Catalog. The Bit alignment module rotates PLL 16 phases to center Edge clock to middle of data eye and the word alignment module uses ALIGNWD function of CLKDIVD and IDDRX71 to achieve 7-bit word alignment.

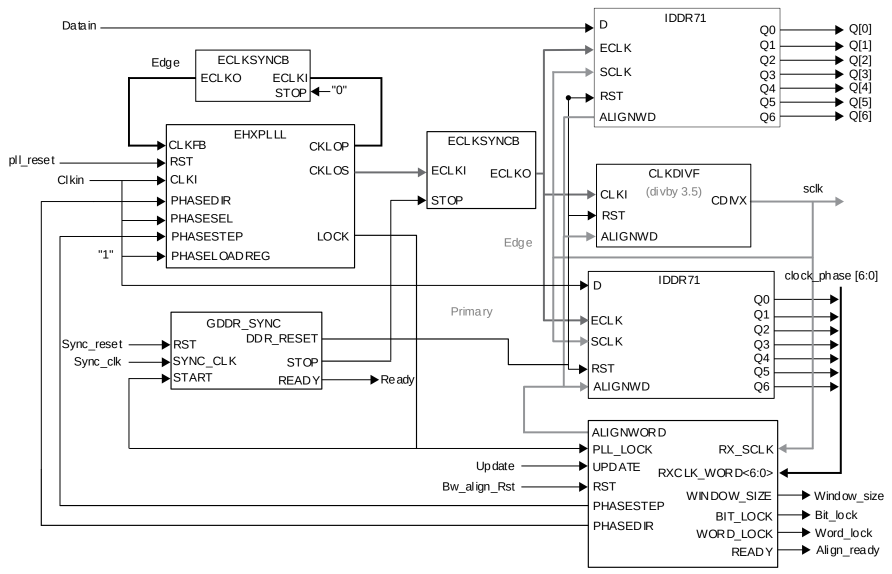


Figure 5.17. GDDRX71_RX.ECLK Interface

Interface Requirements

The clock input must use a dedicated PLL input pin so it is routed directly to the PLL.

- CLKOP output of the PLL must be used as feedback using another Edge Clock tree to compensate for ECLK tree delay used by CLKOS. Hence, this interface uses two ECLK trees.
- ECLK must use the Edge Clock tree and the SCLK out of the CLKDIVF must use the Primary Clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK out of the CLKDIVF module.

5.10. Soft MIPI D-PHY Receive Interfaces

Soft MIPI D-PHY receive interface is supported for LIFCL-33/33U. The Input DDR elements can be configured as MIPI D-PHY inputs to receive MIPI CSI-2/DSI data using the X4 gearing with ECLK sync and clock divider elements. MIPI D-PHY uses a center-aligned clock. The interface uses the programmable LVDS I/O in bottom banks as MIPI input buffers.

The input and output signals are listed and described in Table 5.1.

Table 5.1. MIPI D-PHY Module Signal Descriptions

Port Name	I/O	Width	Description
Clock and Reset			
sync_clk_i	In	1	GDDR SYNC low speed continuously running input clock
sync_rst_i	In	1	GDDR SYNC active high input reset
clk_byte_o	Out	1	Byte clock
clk_p_io, clk_n_io	In/Out	1	MIPI D-PHY differential clock lanes
lp_tx_clk_p_i[1]	In	1	Low power transmit positive clock
lp_tx_clk_n_i[1]	In	1	Low power transmit negative clock
lp_rx_clk_p_o[2]	Out	1	Low power receive positive clock
lp_rx_clk_n_o[2]	Out	1	Low power receive negative clock
MIPI D-PHY High-Speed Tx			
hs_tx_en_i	In	1	High-speed transmit mode enable
hs_tx_clk_en_i	In	1	High-speed transmit mode clock enable
hs_tx_data_i	In	*Bus Width * Gearing Ratio*	High-speed transmit mode data
MIPI D-PHY High-Speed Rx			
hs_rx_en_i	In	1	High-speed receive mode enable
hs_rx_data_o	Out	*Bus Width * Gearing Ratio*	High-speed receive mode data. The data is gated by clk_byte_o clock.
MIPI D-PHY Low Power Signal			
lp_tx_en_i[1]	In	*1*	Low power transmit enable
lp_tx_data_p_i[1]	In	*Bus width* if *Interface Type == Transmit* else 1	Low power transmit positive data lane
lp_tx_data_n_i[1]	In	*Bus width* if *Interface Type == Transmit* else 1	Low power transmit negative data lane
lp_rx_en_i[3]	In	1	Low power receive enable
lp_rx_data_p_o[3]	Out	*Bus width* if *Interface Type == Receive* else 1	Low power receive positive data
lp_rx_data_n_o[3]	Out	*Bus width* if *Interface Type == Receive* else 1	Low power receive negative data
MIPI D-PHY			
data_p_io, data_n_io	In/Out	*Bus Width*	MIPI D-PHY differential data lanes
Misc			
pll_clkop_i[4]	In	1	Input clock from external PLL
pll_clkos_i[4]	In	1	90-degree shifted input clock from external PLL
pll_lock_i[5]	In	1	Lock signal from external PLL
pd_dphy_i[6]	In	1	Power down input
ready_o	Out	1	Ready output signal from D-PHY or from PLL

Notes:

1. These ports are available only when Interface Type == Receive and MIPI Interface Application == DSI, or Interface Type == Transmit.
2. These ports are available only when *Interface Type* == *Receive*.
3. These ports are available only when Interface Type == Transmit and MIPI Interface Application == DSI, or Interface Type == Receive.
4. These ports are available only when *Interface Type==Transmit* and *D-PHY PLL Mode==External*.
5. This port is available only when Interface Type==Transmit and D-PHY PLL Mode==External, or Interface Type==Receive.
6. These ports are available only when *Interface Type* == *Transmit*.

5.10.1. Implementation Details

- GDDR SYNC soft IP module is used to start up the RX interface with X4 gearing. This synchronizes the ECLKDIV and DDR elements and signals that RX is ready for operation.
- ECLKSYNC module provides the clock alignment function when ECLKSYNC.STOP is asserted. This alignment function is enabled when ECLKSYNC STOP_EN parameter is set to ENABLED.
- ECLKDIV provides the fix divided down frequency clock to drive the IDDRX4 SCLK input signals to support the required RX gearing data ratio.
- DELAYB is used to delay the input data. This is used with the IDDR module.
- IDDRX4 module is used to implement the X4 gearing of the RX interface.
- MIPI primitive is used to receive MIPI data and clock.
- The clock input must use a PCLK input so that it can be routed directly to the edge clock tree.

Figure 5.18 shows the available input and output ports, while Figure 5.19 shows the interconnection of the modules and primitive on MIPI D-PHY RX interface.

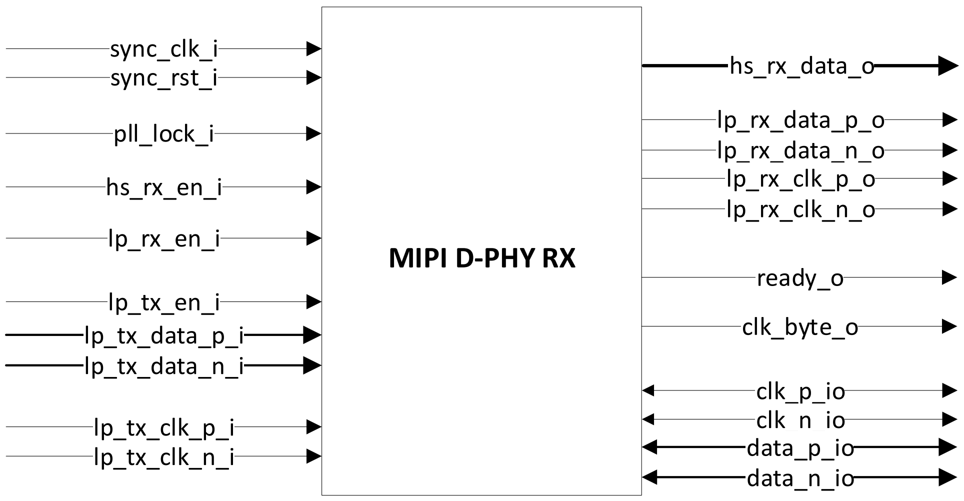
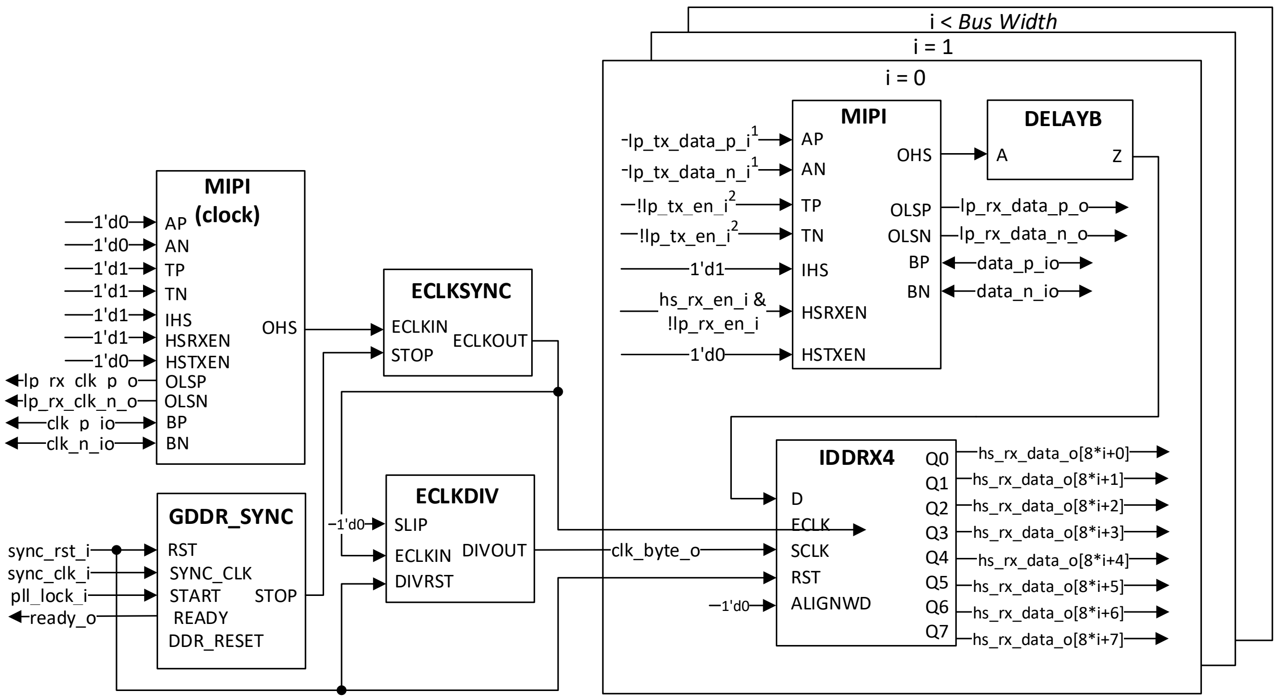


Figure 5.18. MIPI D-PHY RX Block Diagram



Notes:

1. When i !=0 this is equal to 1'd0
2. When i !=0 this is equal to 1'd1

Figure 5.19. MIPI D-PHY RX Interface Diagram

5.11. GDDRX1_TX.SCLK.Aligned

This interface is used to implement Generic Transmit DDR with X1 gearing using primary clock (SCLK). The Clock output is aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX1 element is used to generate the data output.
- The primary clock (SCLK) is used as the clock for both data and clock generation.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the output data.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

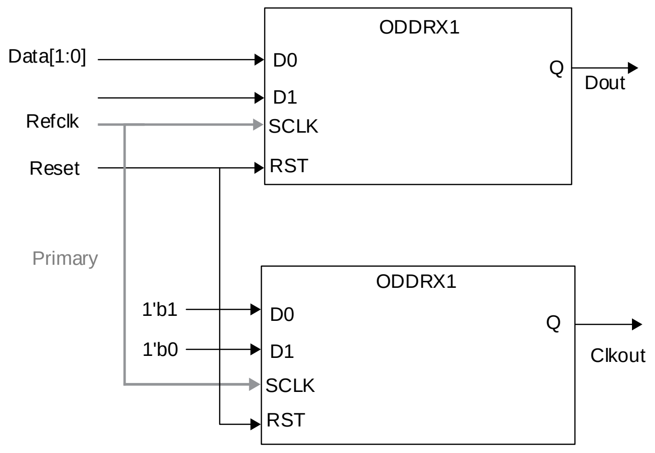


Figure 5.20. GDDRX1_TX.SCLK.Aligned Interface

Interface Requirement

- The clock to the output DDR modules must be routed on the primary clock tree.

5.12. GDDRX1_TX.SCLK.Centered

This section describes the Generic Transmit DDR using X1 gearing with SCLK. Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX1 element is used to generate the data output.
- The EHXPLLL element is used to generate the clocks for the data and clock ODDRX1F modules. The clock used to generate the clock output is delayed 90 degrees to center to data at the output.
- Both these clocks are routed on primary clock tree.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the output data.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

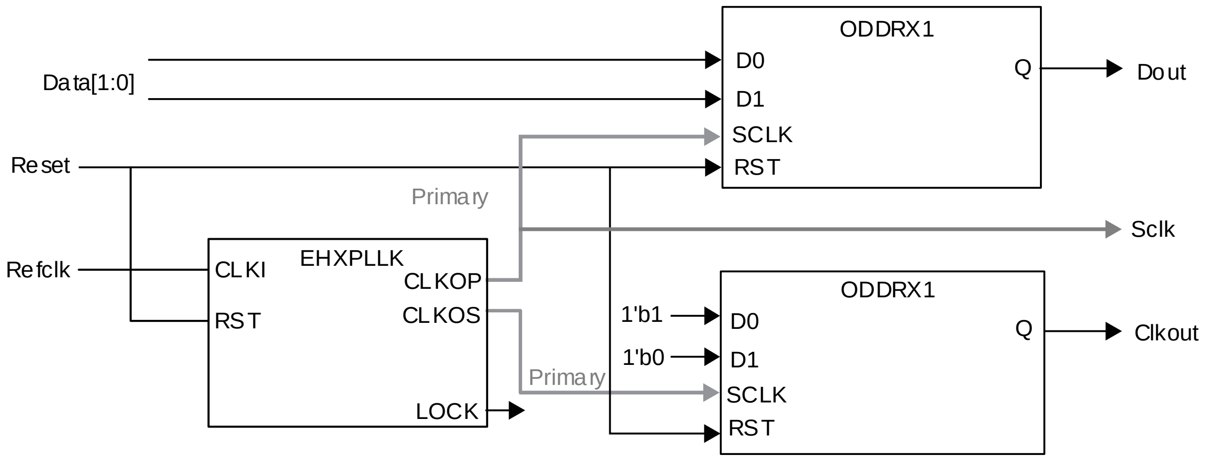


Figure 5.21. GDDRX1_TX.SCLK.Centered Interface

Interface Requirement

- The clock to the output DDR modules must be routed on the primary clock tree.

5.13. GDDRX2_TX.ECLK.Aligned

This interface is used to generate Generic Transmit DDR with X2 gearing using high-speed edge clock (ECLK). The Clock output is edge aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX2 for X2 gearing is used to generate the output data.
- The high-speed ECLK is routed to the edge clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

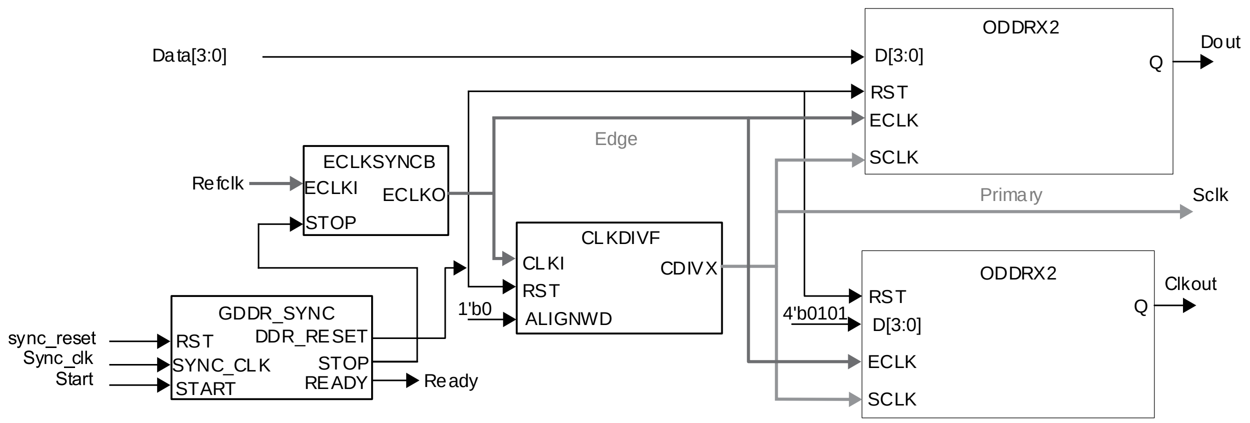


Figure 5.22. GDDRX2_TX.ECLK.Aligned Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the Edge Clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.14. GDDRX2_TX.ECLK.Centered

This interface is used to implement Generic Transmit DDR with X2 gearing using Edge Clock (ECLK). The Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX2 for X2 gearing is used to generate the data output.
- The high-speed ECLK is routed to the Edge Clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The EHXPLLL element is used to generate the clocks for the data and clock ODDR modules. The clock used to generate the clock output is delayed 90 degrees to center to data at the output.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

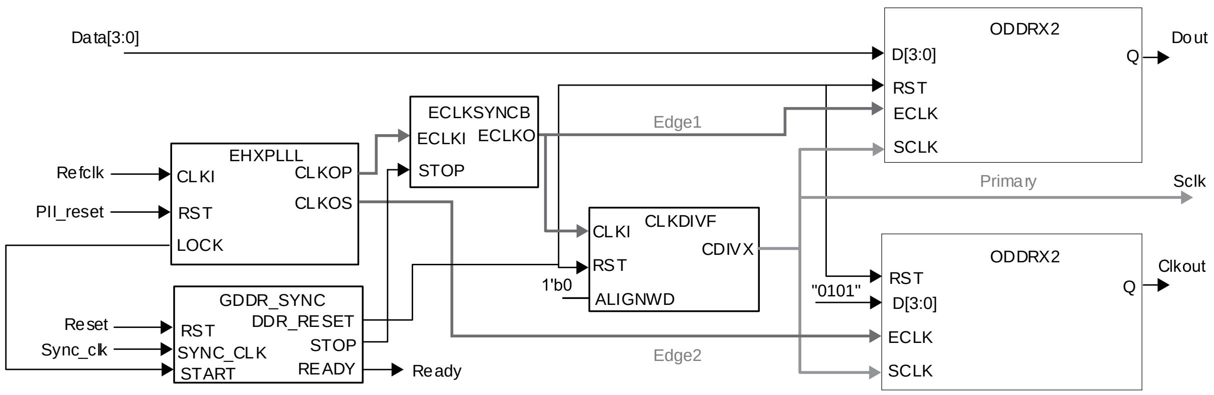


Figure 5.23. GDDRX2_TX.ECLK.Centered Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the Edge Clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.15. GDDRX4_TX.ECLK.Aligned

This interface is used to generate Generic Transmit DDR with X4 gearing using high-speed edge clock (ECLK). The Clock output is edge aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX4 for X4 gearing is used to generate the output data.
- The high-speed ECLK is routed to the edge clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

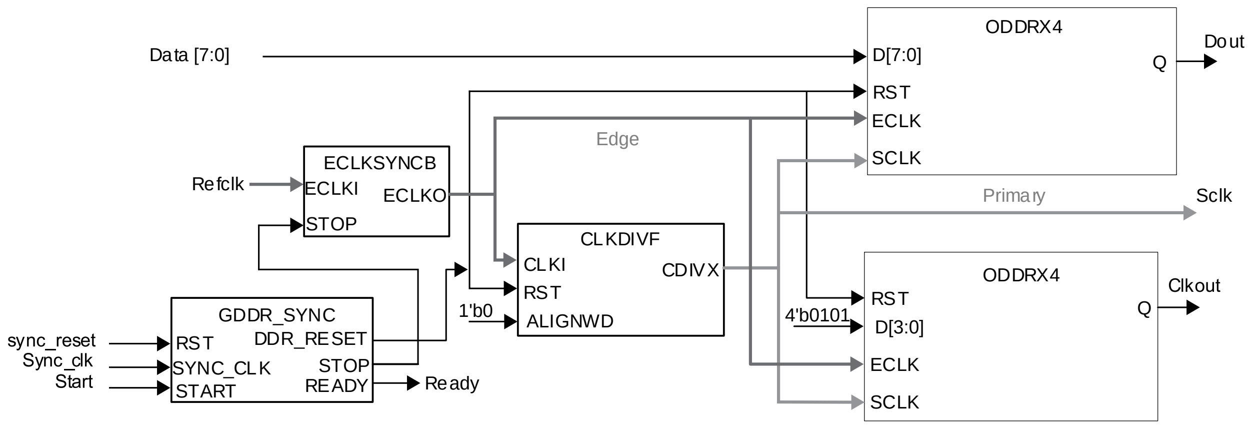


Figure 5.24. GDDRX4_TX.ECLK.Aligned Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.16. GDDRX4_TX.ECLK.Centered

This interface is used to implement Generic Transmit DDR with X4 gearing using edge clock (ECLK). The Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX4 for X4 gearing is used to generate the data output.
- The high-speed ECLK is routed to the edge clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The EHXPLLL element is used to generate the clocks for the data and clock ODDR modules. The clock used to generate the clock output is delayed 90 to center to data at the output.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

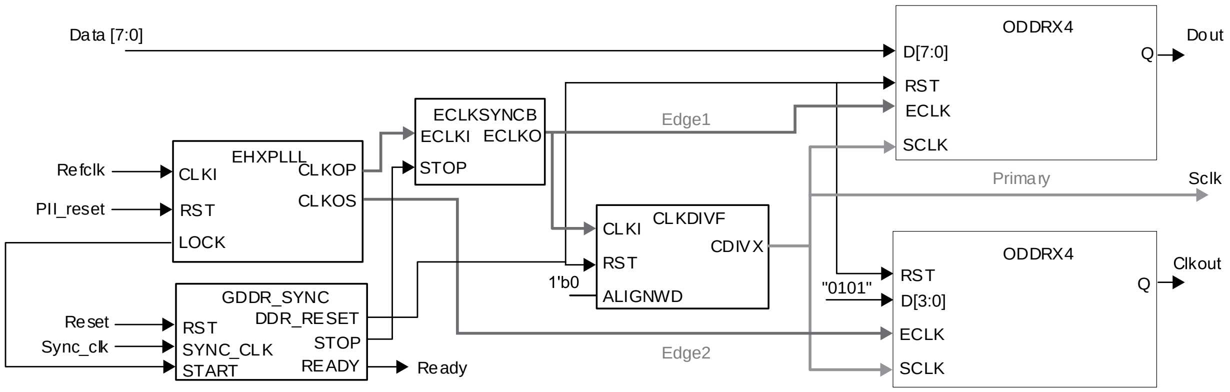


Figure 5.25. GDDRX4_TX.ECLK.Aligned Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.17. GDDRX5_TX.ECLK.Aligned

This interface is used to generate Generic Transmit DDR with X5 gearing using high-speed edge clock (ECLK). The Clock output is edge aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX5 for X5 gearing is used to generate the output data.
- The high-speed ECLK is routed to the edge clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

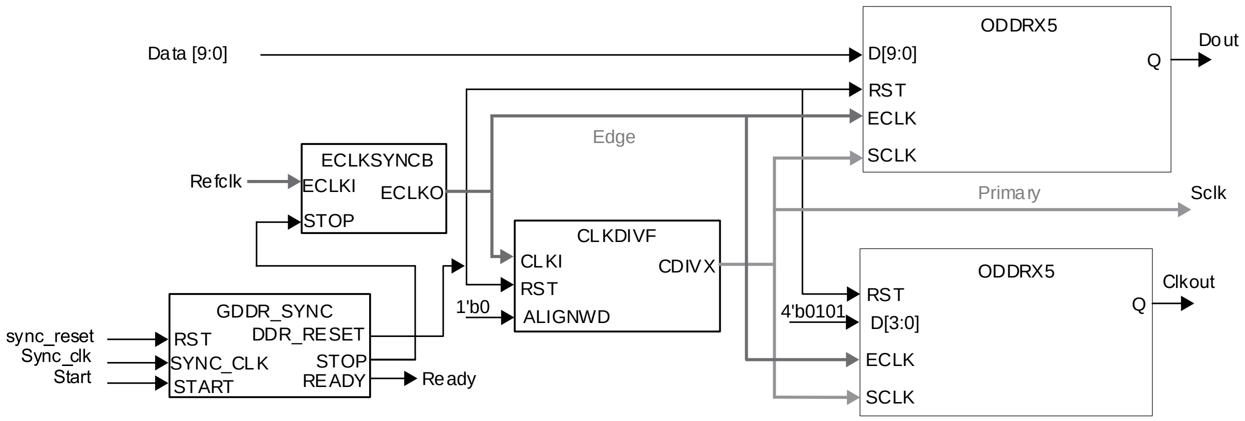


Figure 5.26. GDDRX5_TX.ECLK.Aligned Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.18. GDDRX5_TX.ECLK.Centered

This interface is used to implement Generic Transmit DDR with X5 gearing using edge clock (ECLK). The Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX5 for X5 gearing is used to generate the data output.
- The high-speed ECLK is routed to the edge clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The EHXPLLL element is used to generate the clocks for the data and clock ODDR modules. The clock used to generate the clock output is delayed 90 to center to data at the output.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tri-stated using either a tri-state input going through an I/O register.

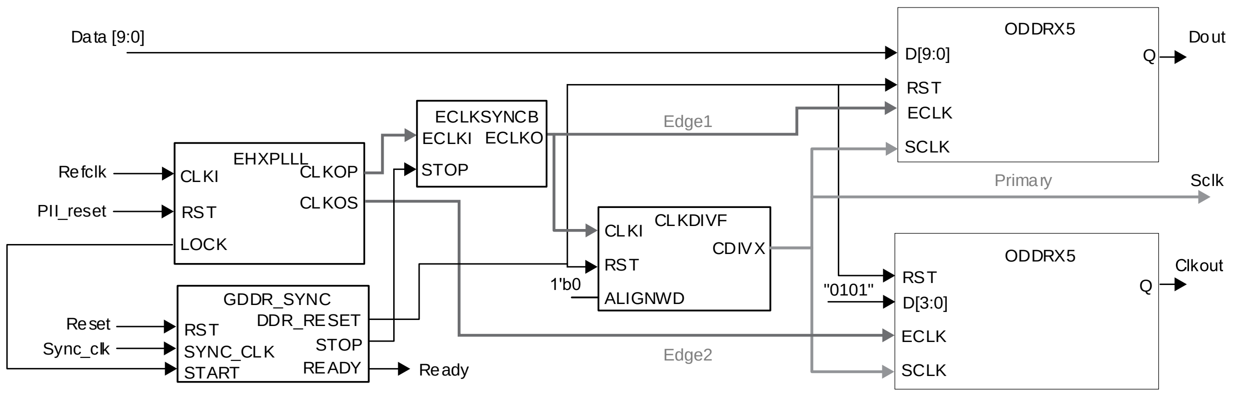


Figure 5.27. GDDRX5_TX.ECLK.Aligned Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.19. GDDRX71_TX.ECLK

This interface is used to implement transmit side of the 7:1 LVDS interface DDR using the 7 to 1 gearing with ECLK. The clock output is aligned to the data output.

This DDR interface uses the following modules:

- ODDR71 is used to generate the data output.
- The high-speed ECLK is routed to the Edge Clock tree through the ECLKSYNCB module.
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVD module.
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.

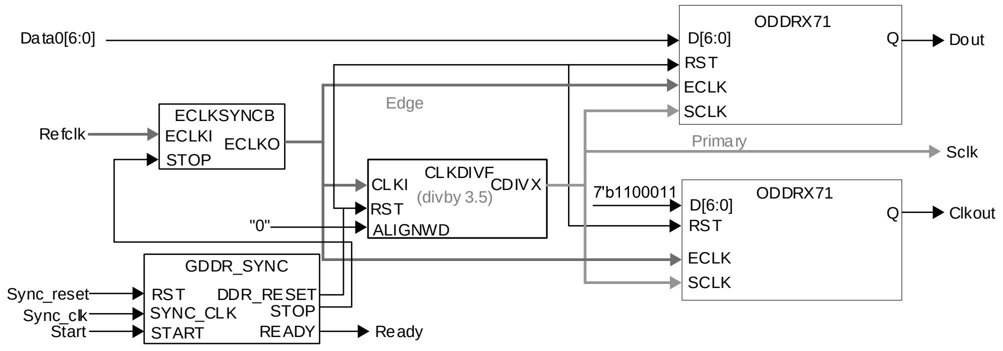


Figure 5.28. GDDRX71_TX.ECLK Interface

Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the Edge Clock tree.
- *USE PRIMARY* preference may be assigned to the SCLK net.

5.20. Soft MIPI D-PHY Transmit Interfaces

Soft MIPI D-PHY transmit interface is supported for LIFCL-33/33U devices. The output DDR elements can be configured as MIPI D-PHY outputs to transmit MIPI CSI-2/DSI data using the X4 gearing with ECLK sync and clock divider elements. MIPI D-PHY uses a center-aligned clock. The interface uses the programmable LVDS I/O in bottom banks as MIPI output buffers.

Detailed descriptions of MIPI D-PHY Tx ports are available in Table 5.1.

5.20.1. Implementation Details

- GDDR SYNC soft IP module is used to start up the TX interface with X4 gearing. This synchronizes the ECLKDIV and DDR elements and signals that TX is ready for operation.
- ECLKSYNC module provides the clock alignment function when ECLKSYNC.STOP is asserted. This alignment function is enabled when ECLKSYNC STOP_EN parameter is set to ENABLED.
- ECLKDIV provides the fix divided down frequency clock to drive the ODDRX4 SCLK input signals to support the required TX gearing data ratio.
- ODDRX4 module is used to implement the X4 gearing of the TX interface.
- MIPI primitive is used to transmit MIPI data and clock.

Figure 5.29 shows the available input and output ports, while Figure 5.30 shows the interconnection of the modules and primitive on MIPI D-PHY TX interface.

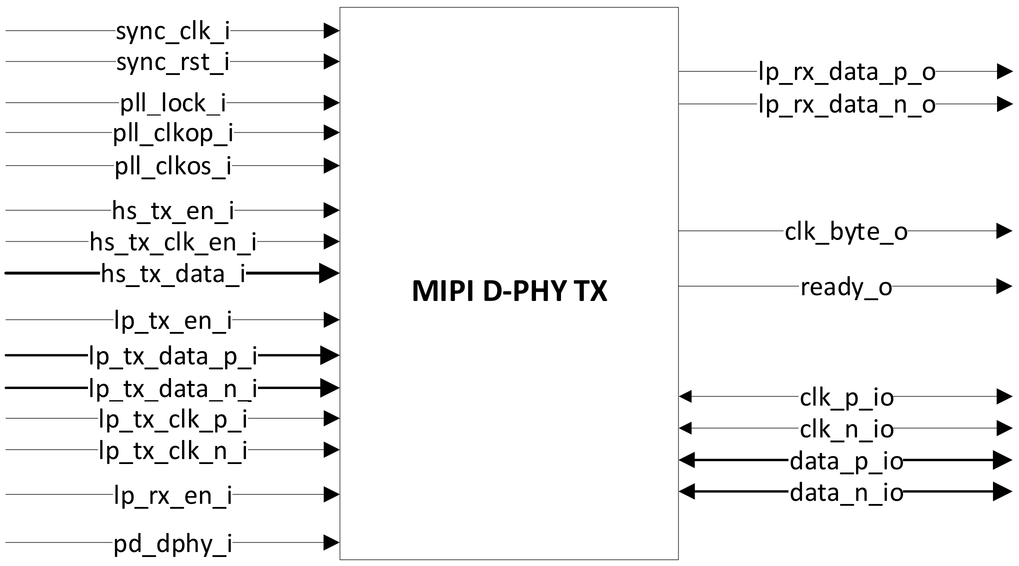
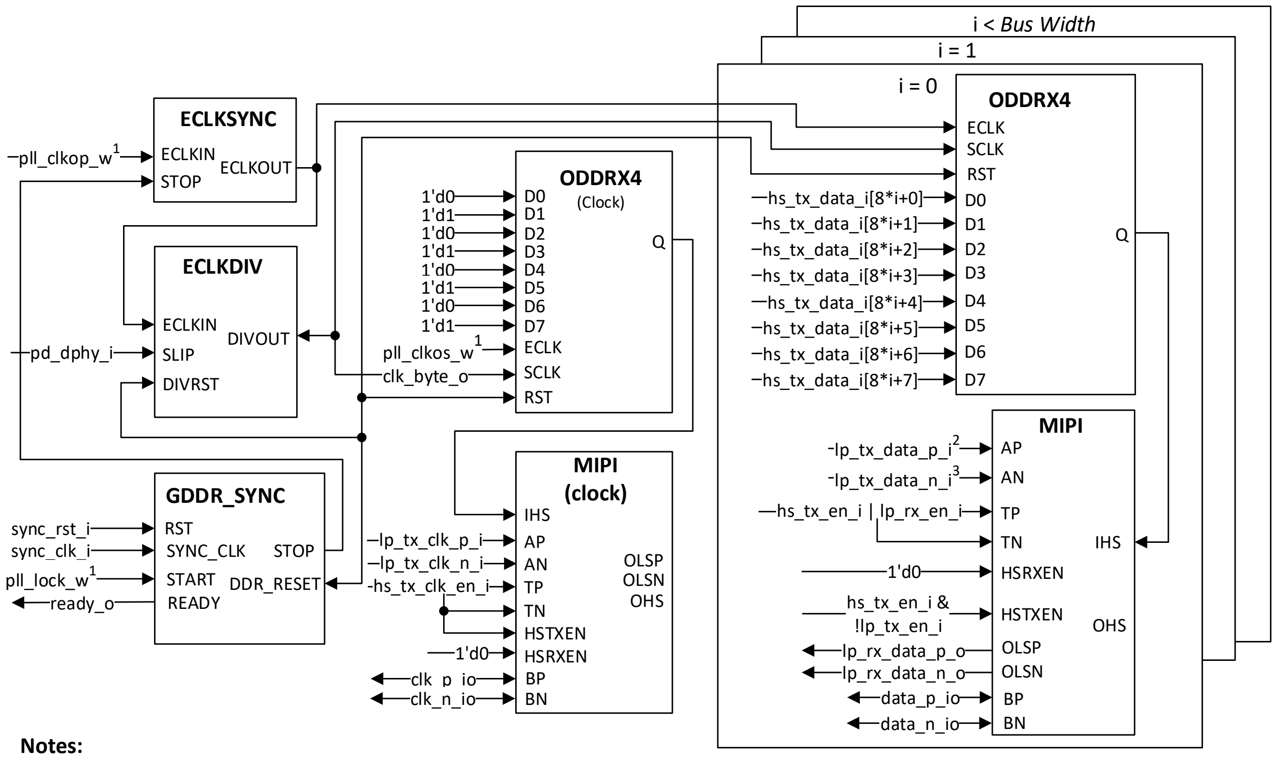


Figure 5.29. MIPI D-PHY TX Block Diagram



Notes:

1. These ports connect to internal or external PLL (pll_lock_i, pll_clkop_w and pll_clkos_w
2. When i ==0 this is equal to lp_tx_data_p_i[0] & !lp_rx_en_i
3. When i ==0 this is equal to lp_tx_data_n_i[0] & !lp_rx_en_i

Figure 5.30. MIPI D-PHY TX Interface Diagram

5.21. Generic DDR Design Guidelines

This section describes the various design guidelines used for building generic high-speed DDR interfaces in LIFCL-33/33U devices. In addition to these guidelines, it is also required to follow the Interface Rules described for each type of interface. Refer to the High-Speed DDR Interface Details section to find the interface you are building.

5.21.1. Receive Interface Guidelines

- Differential DDR interface can be implemented on the bottom side of the device.
- There are four different Edge Clocks available on the bottom side of the device can be used to generate either a center or aligned interface.
- Each Bank on the bottom side of the device has four CLKDIV modules, which means that you can implement four different GDDRX2 RX interface per Bank since each 2×, 4×, or 5× gearing would require CLKDIV module to generate a slower SCLK.
- There are two DDRDLLs located in the lower left and lower right corners of the device.
- Each DLLDEL has access to two DDRDLLs. Hence, two different RX rates are available on the bottom side.
- The Receive clock input should be placed on a dedicated PCLK input pin. The PCLK pin has direct access to the Edge Clock tree for centered interface and it also has direct connection to the DLLDEL when implementing an aligned interface.
- When implementing IDDRX71 interface, the complementary PAD is not available for other functions since the IDDRX71 used the I/O registers of the complementary PAD as well.
- It is recommend that clock input be located on the same side as data pins.
- The top, left, and right side of the device do not have Edge Clocks. Hence, these can only be used to receive lower speed interfaces (<250 MHz) that use 1× gearing. These can be used for single ended interfaces only.
- Interfaces using the ×1 gearing uses the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.
- In addition to the dedicated PCLK pins, LIFCL-33/33U devices have GR_PCLK pins. These pins use the shortest general route path to get to the primary clock tree. These pins are not recommended for use with DDR interfaces. They can be used for SDR or other generic FPGA designs.

5.21.2. Transmit interface Guidelines

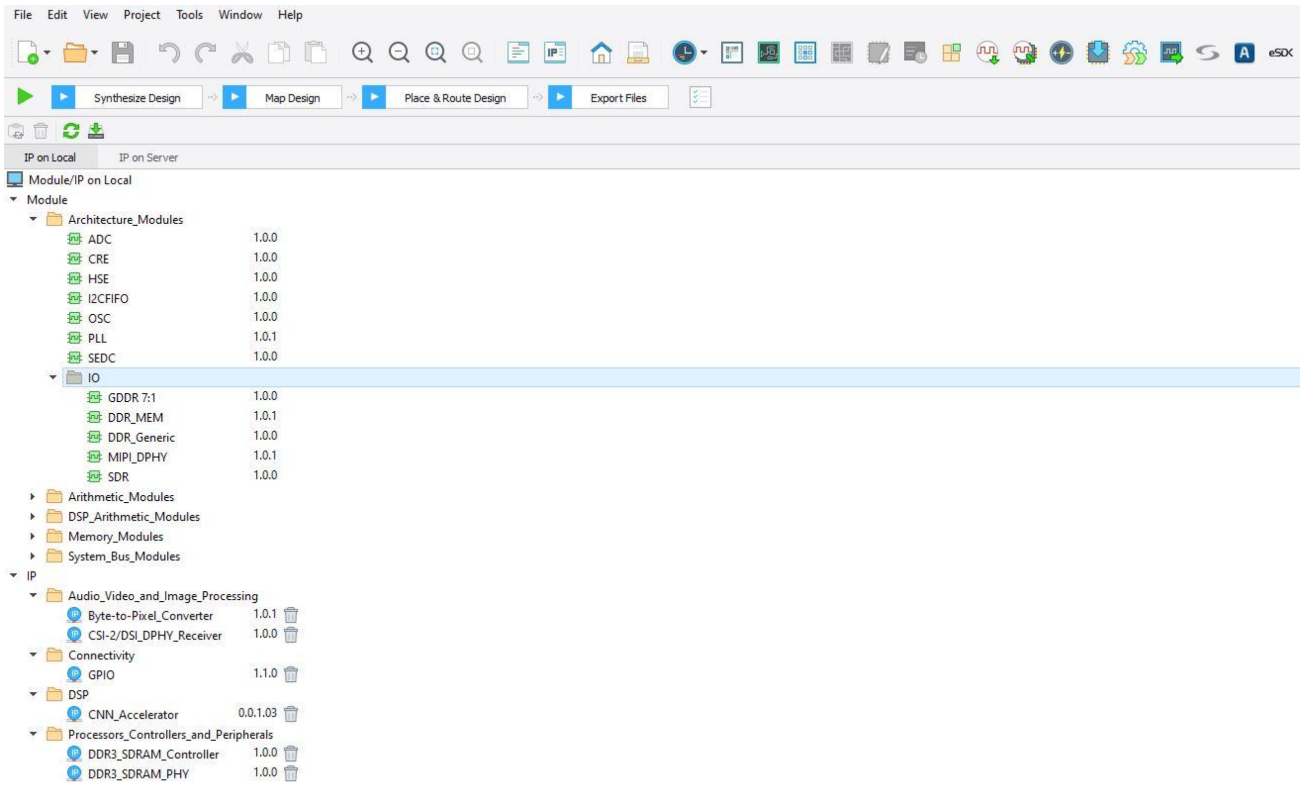
- Use PADA and PADB for all TX using true LVDS interfaces.
- When implementing Transmit Centered interface, two ECLKs are required. Once to generate the Data Output and the other to generate the CLK Output.
- When implementing Transmit Aligned interface, only one ECLK is required for both Data output and Clock output.
- Each Bank on the bottom side of the device has two CLKDIV modules, which means that you can implement two different GDDRX2 TX interface per Bank since each 2×, 4×, or 5× gearing would require CLKDIV module to generate a slower SCLK.
- The top, left and right side of the device does not have Edge Clocks hence can only be used to receive lower speed interfaces (<250 MHz) that use 1X gearing. It can be used for single ended interfaces only.
- Interfaces using the ×1 gearing uses the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.

5.21.3. Clocking Guidelines for Generic DDR Interface

- The Edge Clock and Primary Clock resources are used when implementing a 2×, 4×, or 5× receive or transmit interface.
- Only the Primary Clock (PCLK) resources are used when implementing x1 receive or transmit interfaces.
- Each Edge Clock can only span up to one side of the device, hence all the data bits of the in the ×2 interface must be locked to one side of the device.
- The ECLK to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DDRDLL outputs. See sysCLOCK PLL Design and Usage Guide for Nexus Platform (FPGA-TN-02095) for details.
- Primary Clock to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, and CLKDIV outputs. See sysCLOCK PLL Design and Usage Guide for Nexus Platform (FPGA-TN-02095) for details.
- None of the clocks going to the DDR registers can come from internal general routing.
- For LIFCL-33, DQS clocking is used for DDR memory interface implementation. DQS clock spans every 12 I/O including the DQS pins. Refer to the DQS Grouping section for pinout assignment rules when using DQS clocking.

6. Using IP Catalog to Build and Plan High Speed DDR Interfaces

The IP Catalog tool is used to configure, build, and plan placement all DDR interfaces.



```
Module/IP on Local
Module
  Architecture_Modules
    ADC                              1.0.0
    CRE                              1.0.0
    HSE                              1.0.0
    I2CFIFO                          1.0.0
    OSC                              1.0.0
    PLL                              1.0.1
    SEDC                             1.0.0
    IO
      GDDR 7:1                       1.0.0
      DDR_MEM                        1.0.1
      DDR_Generic                    1.0.0
      MIPI_DPHY                      1.0.1
      SDR                            1.0.0
  Arithmetic_Modules
  DSP_Arithmetic_Modules
  Memory_Modules
  System_Bus_Modules
IP
  Audio_Video_and_Image_Processing
    Byte-to-Pixel_Converter          1.0.1
    CSI-2/DSI_DPHY_Receiver          1.0.0
  Connectivity
    GPIO                             1.1.0
  DSP
    CNN_Accelerator                  0.0.1.03
  Processors_Controllers_and_Peripherals
    DDR3_SDRAM_Controller            1.0.0
    DDR3_SDRAM_PHY                   1.0.0
```

Figure 6.1. IP Catalog Main Window

Note: It is recommend that all the DDR modules required for the current design be generated in the same project. This allows the Design Rule Check and the Resource Conflict Check for all the modules at the same time.

6.1. Configuring DDR Modules in IP Catalog

IP Catalog lists all the DDR architecture modules available on LIFCL-33/33U devices.

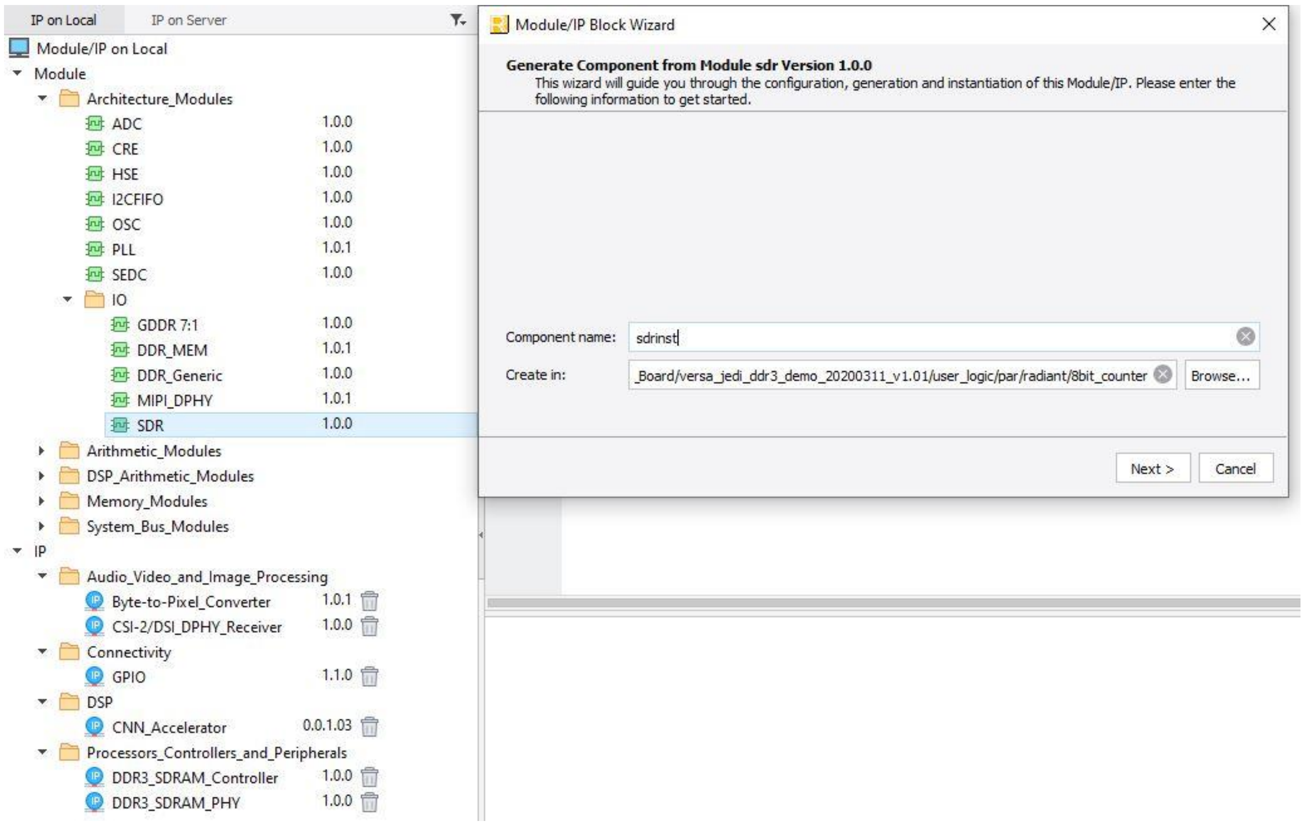
All the DDR modules are located under Architecture Modules – I/O. This includes:

- SDR – Select to build SDR Modules.
- DDR_GENERIC – Select to build any DDR Generic Receive and Transmit Interfaces.
- GDDR_7:1 – Select to build 7:1 LVDS Receiver and Transmit Interface.
- MIPI_DPHY – Select to build MIPI D-PHY Receiver and Transmit Interface
- DDR_MEM – Select to build DDR Memory Interfaces. (*For LIFCL-33 only.*)

To see the detailed block diagram for each interface generated by IP Catalog, see the High-Speed DDR Interface Details section.

6.2. Configuring SDR Modules

To build an SDR interface, select SDR option under Architecture Modules – I/O in the IP Catalog tab. Enter the name of the module. Figure 6.2 shows the type of interface selected as SDR and the module name entered. This module can then be configured by clicking the Next button.

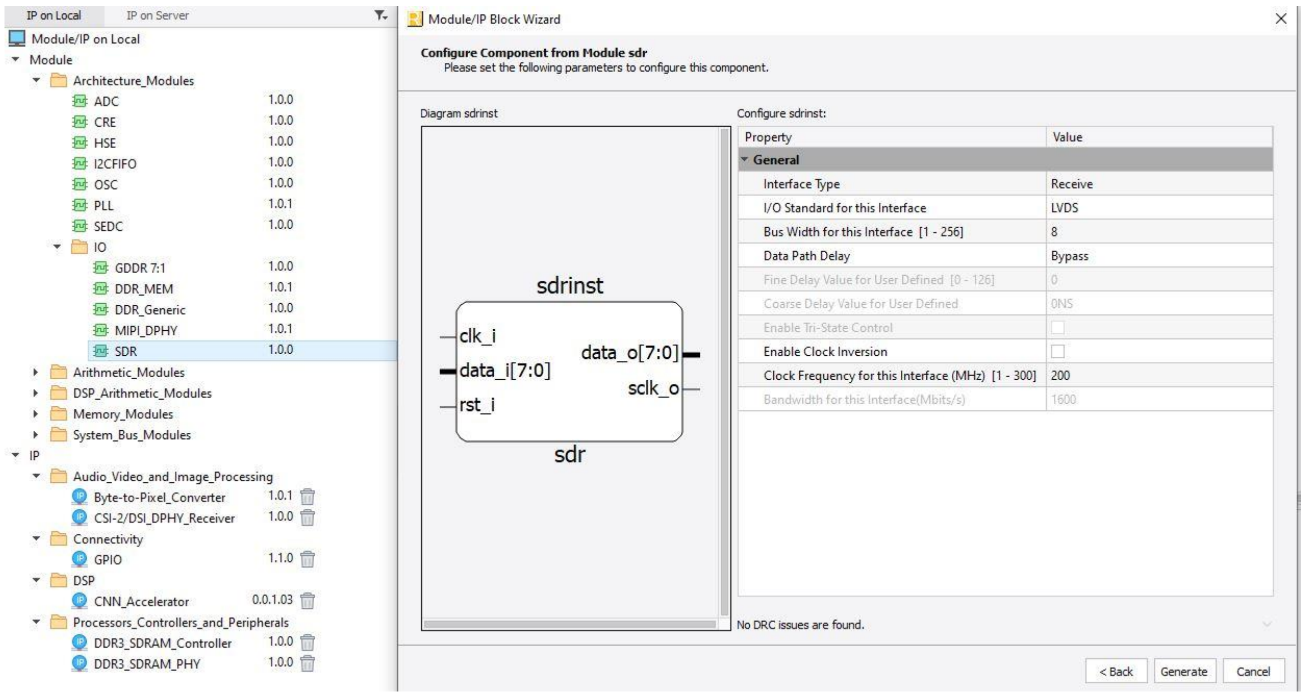


Generate Component from Module sdr Version 1.0.0
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Please enter the following information to get started.

Component name: sdrinst
Create in: _Board/versa_jedi_ddr3_demo_20200311_v1.01/user_logic/par/radiant/8bit_counter

Figure 6.2. SDR Option Selected in the IP Catalog Tab

Figure 6.3 shows the Configuration tab for SDR module. You can make selections and then click Generate.



Configure Component from Module sdr
Please set the following parameters to configure this component.

Configure sdrinst:

Property	Value
General	
Interface Type	Receive
I/O Standard for this Interface	LVDS
Bus Width for this Interface [1 - 256]	8
Data Path Delay	Bypass
Fine Delay Value for User Defined [0 - 126]	0
Coarse Delay Value for User Defined	0NS
Enable Tri-State Control	☐
Enable Clock Inversion	☐
Clock Frequency for this Interface (MHz) [1 - 300]	200
Bandwidth for this Interface(Mbits/s)	1600

No DRC issues are found.

Figure 6.3. SDR Configuration Tab

Table 6.1 explains the various configuration options available for SDR modules.

Table 6.1. EBR-Based Single-Port Memory Port Definitions

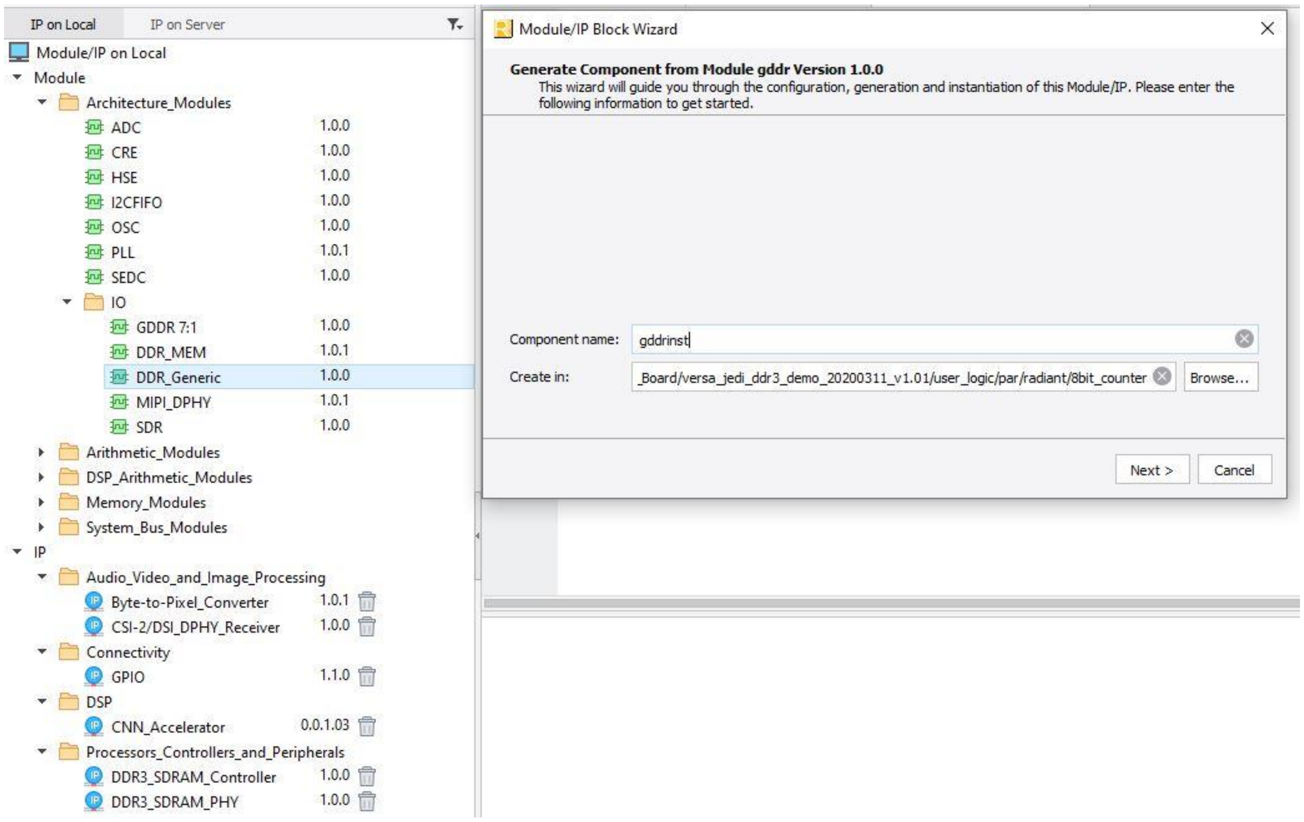
User Interface Option	Description	Values	Default
Interface Type	Type of Interface (Transmit or Receive)	Transmit, Receive	Receive
I/O Standard for this Interface	I/O Standard to be used for the interface	All Valid IO_TYPES	LVDS
Bus Width for this Interface	Bus size for the interface	1–256	8
Clock Frequency for this Interface (MHz)	Interface Speed	1–300	200
Bandwidth (Calculated) (Mbps)	This is the calculated from the Clock frequency entered.	(Calculated)	(Calculated)
Interface	Interface selected based on previous entries	Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default)	GIREG_RX.SCLK
Data Path Delay[1]	Data input can be optionally delayed using the DELAY block.	If Interface Type = Receive then: Bypass, Static Default Dynamic Default Static User Defined Dynamic User Defined If Interface Type = Transmit then: Bypass, Static User Defined Dynamic User Defined	Bypass
Fine Delay Value for User Defined	If Delay type selected above is *user defined*, delay values can be entered with this parameter.	0 to 126	0
Coarse Delay Value for User Defined	If Delay type selected above is *user defined*, delay values can be entered with this parameter.	0ns, 0.8ns,1.6ns	0ns
Enable Tri-State Control	If Delay type selected above is *user defined*.	Disable, Enable	Disable
Enable Clock Inversion	Option to invert the clock input to I/O Register.	Disable, Enable	Disable

Note:

1. When Data Path Delay value is:
    - Bypass: No delay cell is used.
    - Static Default: Static delay element DELAYB is used with attribute DEL_MODE set to SCLK_ZEROHOLD.
    - Static User Defined: Static delay element DELAYB is used with attribute DEL_MODE set to USER_DEFINED.
    - Dynamic Default: Dynamic delay element DELAYA is used with attribute DEL_MODE set to SCLK_ZEROHOLD.
    - Dynamic User Defined: Dynamic delay element DELAYA is used with attribute DEL_MODE = USER_DEFINED.

6.3. Configuring DDR Generic Modules

To build a DDR Generic interface, select the DDR_Generic option under Architecture Modules – I/O in the IP Catalog tab. Enter the name of the module. Figure 6.4 shows the type of interface selected as DDR_Generic and the module name entered. This module can then be configured by clicking the Next button.



Generate Component from Module gddr Version 1.0.0

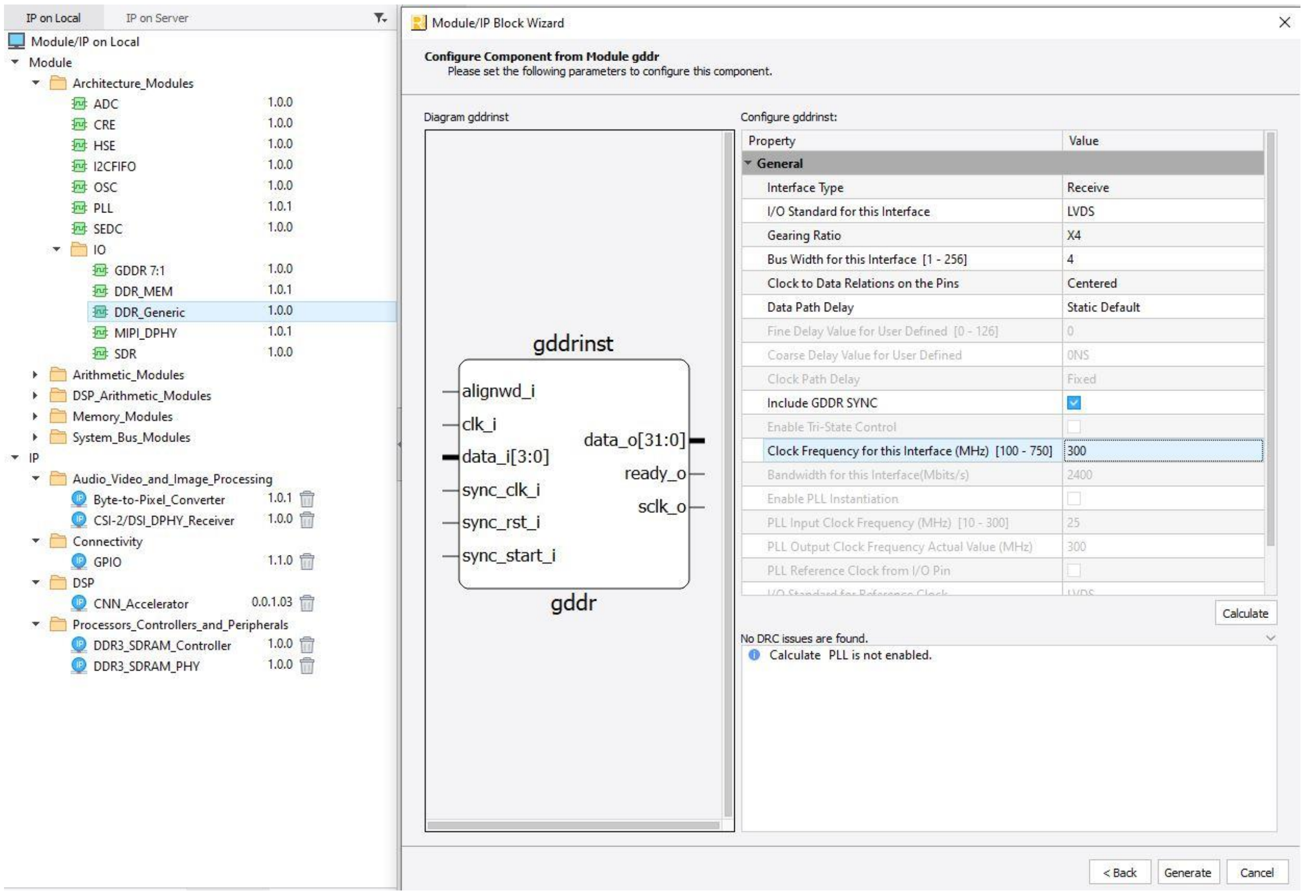
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Please enter the following information to get started.

Component name: gddrinst

Create in: _Board/versa_jedi_ddr3_demo_20200311_v1.01/user_logic/par/radiant/8bit_counter

Figure 6.4. DDR_Generic Option Selected in the IP Catalog Tab

Figure 6.5 shows the Configuration tab.



Configure Component from Module gddr
Please set the following parameters to configure this component.

Configure gddrinst:

Property	Value
General	
Interface Type	Receive
I/O Standard for this Interface	LVDS
Gearing Ratio	X4
Bus Width for this Interface [1 - 256]	4
Clock to Data Relations on the Pins	Centered
Data Path Delay	Static Default
Fine Delay Value for User Defined [0 - 126]	0
Coarse Delay Value for User Defined	0NS
Clock Path Delay	Fixed
Include GDDR SYNC	✓
Enable Tri-State Control	☐
Clock Frequency for this Interface (MHz) [100 - 750]	300
Bandwidth for this Interface(Mbits/s)	2400
Enable PLL Instantiation	☐
PLL Input Clock Frequency (MHz) [10 - 300]	25
PLL Output Clock Frequency Actual Value (MHz)	300
PLL Reference Clock from I/O Pin	☐
I/O Standard for Reference Clock	LVDS

No DRC issues are found.

Calculate PLL is not enabled.

Figure 6.5. DDR_Generic Configuration Tab

Table 6.2 explains the various parameters in the Configuration tab.

Table 6.2. DDR_Generic Configuration Tab Parameters

User Interface Option	Description	Values	Default
Interface Type	Type of Interface (Transmit or Receive)	Transmit, Receive	Receive
I/O Standard	I/O Standard used for the interface	All Legal Input & Output standards	LVDS
Gearing Ratio	DDR register gearing ratio	×1,×2,×4,×5	×1
Bus Width	Bus width for each interface	1 – 256	8
Clock To Data Relations on the pins	Clock to Data alignment	Edge-to-Edge or Centered	Centered
Data Path Delay	Data input can be optionally delayed using the DELAY block. Default Value is selected based on Interface Type.	If Interface Type = Receive: Bypass Static Default Dynamic Default Static User Defined Dynamic User Defined If Interface Type = Transmit: Bypass Static User Defined Dynamic User Defined	Bypass
Fine Delay Value for User Defined	When Data Path Delay of user defined is selected, you also need to set the number of delay steps to be used.	0 – 126	0
Coarse Delay Value for the User Defined	If Delay type selected above is *user defined*, delay values can be entered with this parameter	0 ns, 0.8 ns, 1.6 ns	0 ns
Clock Path Delay	If Interface Type is receiver	Fixed, Dynamic	Fixed
Include GDDR Sync	—	Disable, Enable	Disable
Enable Tristate Control	Generate Tristate control for Transmit Interfaces	Disable, Enable	Disable
Clock Frequency (MHz)	Speed of the Interface	100 MHz – 750 MHz	150 MHz
Interface Bandwidth (Calculated) (Mbps)	—	Clock Frequency for *2* Bus Width	—
Enable PLL Instantiation	When this option is enabled for Transmit interfaces, the PLL used to generate the clocks is included in the generated module.	Disable, Enable	Disable
PLL Input Clock Frequency	Frequency of the clock used as PLL Input	10 MHz – 150 MHz	25
PLL Output Clock Frequency Actual Value	Displays the achieved PLL output clock frequency	Actual PLL output clock frequency achieved based on interface requirement	150
PLL Reference Clock from I/O Pin	—	Disable, Enable	Disabled
I/O Standard for Reference Clock	—	All I/O Standard	LVDS
PLL Out Clock Tolerance (%)	—	Disable, Enable	0.0

Table 6.3 shows how the interfaces are selected.

Table 6.3. IP Catalog DDR_Generic Interface Selection

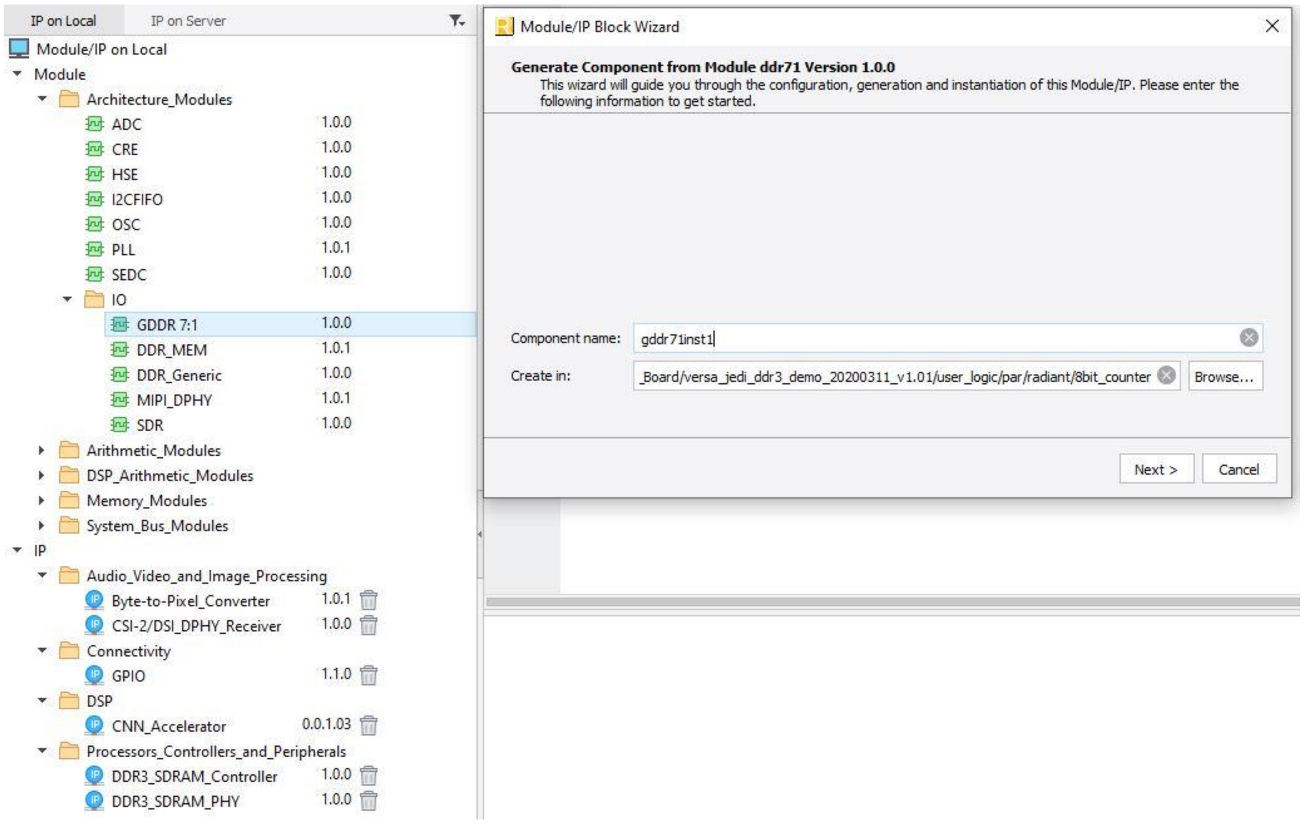
Interface Type	Gearing Ratio	Alignment	Default Interface
Receive	2:1 (×1)	Edge-to-Edge	GDDRX1_RX.SCLK.Aligned
Receive	2:1 (×1)	Centered	GDDRX1_RX.SCLK.Centered
Receive	4:1 (×2)	Edge-to-Edge	GDDRX2_RX.ECLK.Aligned
Receive	4:1 (×2)	Centered	GDDRX2_RX.ECLK.Centered
Receive	8:1 (×4)	Edge-to-Edge	GDDRX4_RX.ECLK.Aligned
Receive	8:1 (×4)	Centered	GDDRX4_RX.ECLK.Centered
Receive	10:1 (×5)	Edge-to-Edge	GDDRX5_RX.ECLK.Aligned
Receive	10:1 (×5)	Centered	GDDRX5_RX.ECLK.Centered
Transmit	2:1 (×1)	Edge-to-Edge	GDDRX1_TX.SCLK.Aligned
Transmit	2:1 (×1)	Centered	GDDRX1_TX.SCLK.Centered
Transmit	4:1 (×2)	Edge-to-Edge	GDDRX2_TX.ECLK.Aligned
Transmit	4:1 (×2)	Centered	GDDRX2_TX.ECLK.Centered
Transmit	8:1 (×4)	Edge-to-Edge	GDDRX4_TX.ECLK.Aligned
Transmit	8:1 (×4)	Centered	GDDRX4_TX.ECLK.Centered
Transmit	10:1 (×5)	Edge-to-Edge	GDDRX5_TX.ECLK.Aligned
Transmit	10:1 (×5)	Centered	GDDRX5_TX.ECLK.Centered

Refer to the High-Speed DDR Interface Details section to see implementation details for each of these interfaces.

6.4. Configuring 7:1 LVDS Interface Modules

To build a 7:1 LVDS DDR interface, select GDDR_7:1 option under Architecture Modules – I/O in the IP Catalog. Enter the name of the module.

Figure 6.6 shows the type of interface selected as GDDR_7:1 and module name entered. This module can then be configured by clicking the Next button.

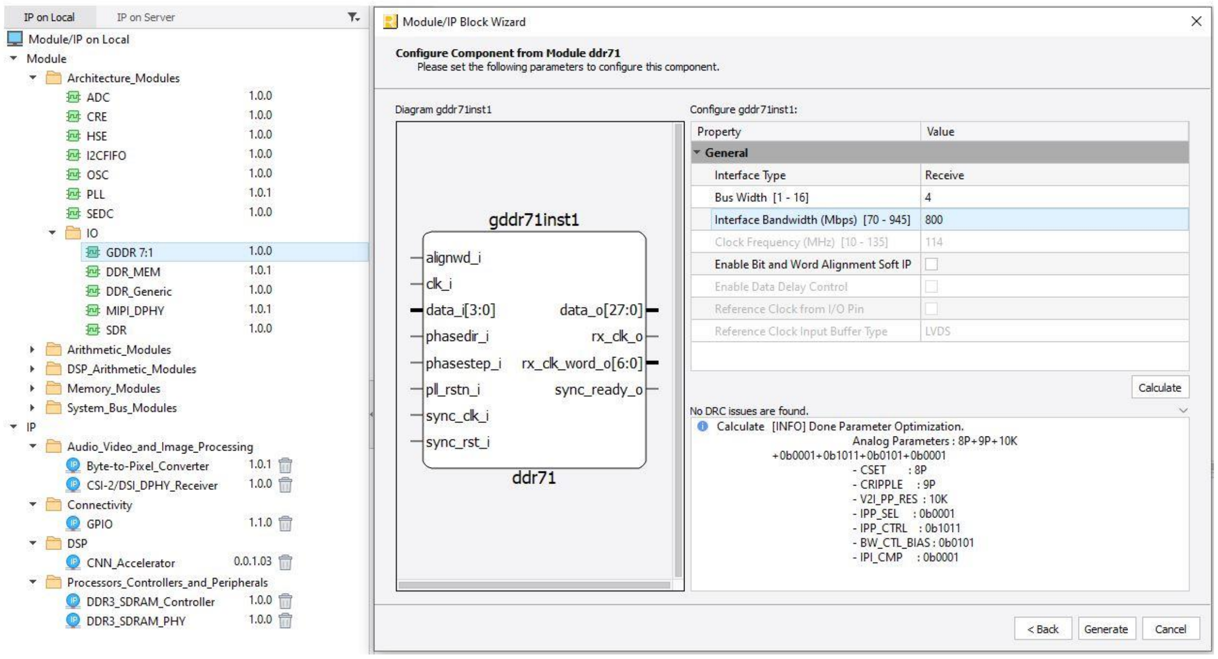


Generate Component from Module ddr71 Version 1.0.0
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Please enter the following information to get started.

Component name: gddr71inst1
Create in: _Board/versa_jedi_ddr3_demo_20200311_v1.01/user_logic/par/radiant/8bit_counter

Figure 6.6. GDDR_7:1 Option Selected in the IP Catalog Tab

Clicking Customize displays the Configuration tab where the 7:1 LVDS interface can be configured. Figure 6.7 shows the Configuration tab for 7:1 LVDS interfaces.



Configure Component from Module ddr71
Please set the following parameters to configure this component.

Configure gddr71inst1:

Property	Value
General	
Interface Type	Receive
Bus Width [1 - 16]	4
Interface Bandwidth (Mbps) [70 - 945]	800
Clock Frequency (MHz) [10 - 135]	114
Enable Bit and Word Alignment Soft IP	☐
Enable Data Delay Control	☐
Reference Clock from I/O Pin	☐
Reference Clock Input Buffer Type	LVDS

No DRC issues are found.

```
Calculate  [INFO] Done Parameter Optimization.
                         Analog Parameters : 8P+9P+10K
          +0b0001+0b1011+0b0101+0b0001
                         - CSET        : 8P
                         - CRIPPLE     : 9P
                         - V2I_PP_RES  : 10K
                         - IPP_SEL     : 0b0001
                         - IPP_CTRL    : 0b1011
                         - BW_CTL_BIAS : 0b0101
                         - IPI_CMP     : 0b0001
```

Figure 6.7. GDDR_7:1 LVDS Configuration Tab

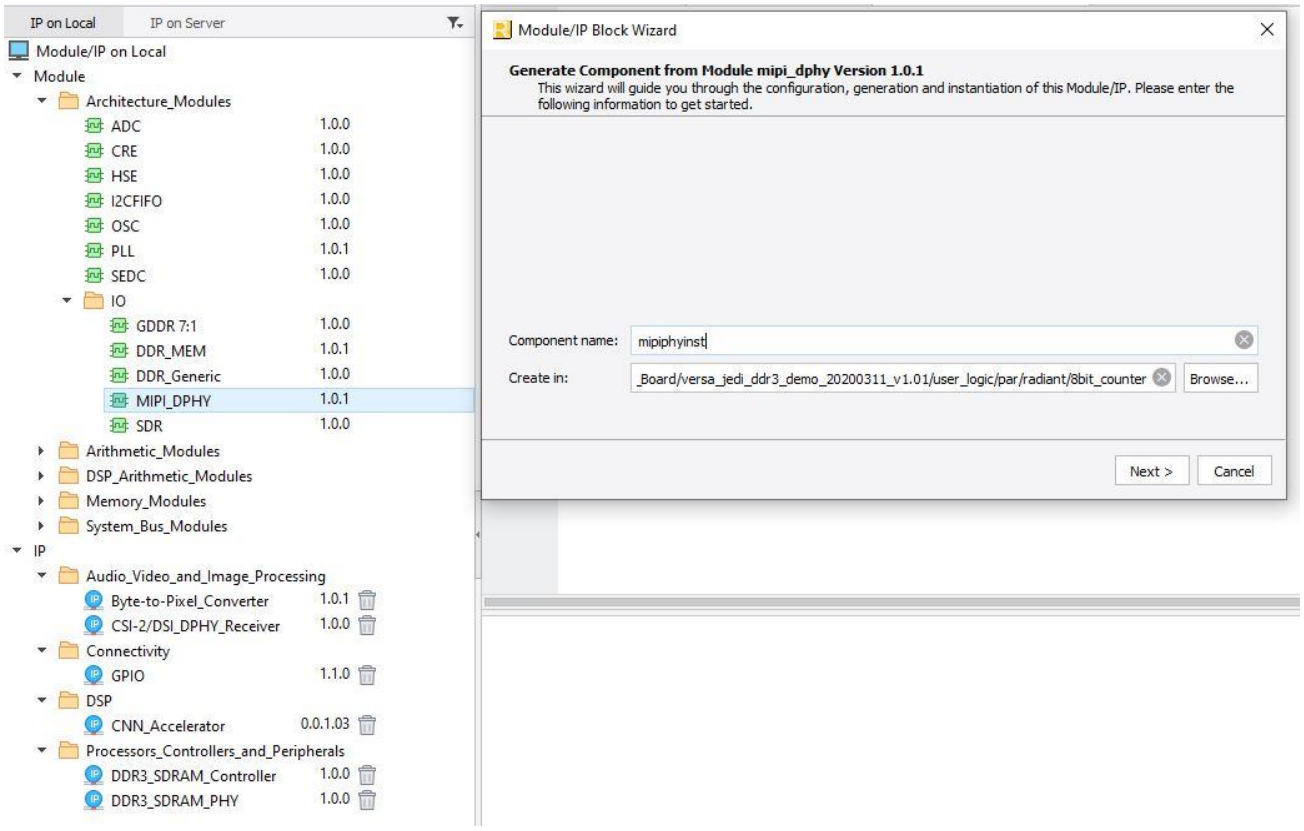
Table 6.4 explains the various parameters for GDDR_7:1 configuration.

Table 6.4. GDDR_7:1 LVDS Configuration Parameters

User Interface Option	Description	Values
Interface Type	Type of interface (Receive or Transmit)	Transmit, Receive
Bus Width	Bus width for one channel of 7:1 LVDS interface	1 – 16
Clock Frequency (MHz)	Pixel clock speed	10 MHz – 135 MHz
Interface Bandwidth (Mbps)	Interface bandwidth	70 Mbps – 945 Mbps
Enable Bit and Word Alignment Soft IP	Soft IP included with the module to implement Bit and Word alignment for receive interface	Enable, Disable
Enable Data Delay Control	Only As Enable Bit and Word Alignment soft IP is selected	Enable, Disable
Reference Clock from I/O pin	For transmit interface	Enable, Disable
Reference Clock Input Buffer Type	Only as Reference Clock from I/O Pins is selected	All I/O type (LVDS as default)

6.5. Configuring MIPI D-PHY Modules

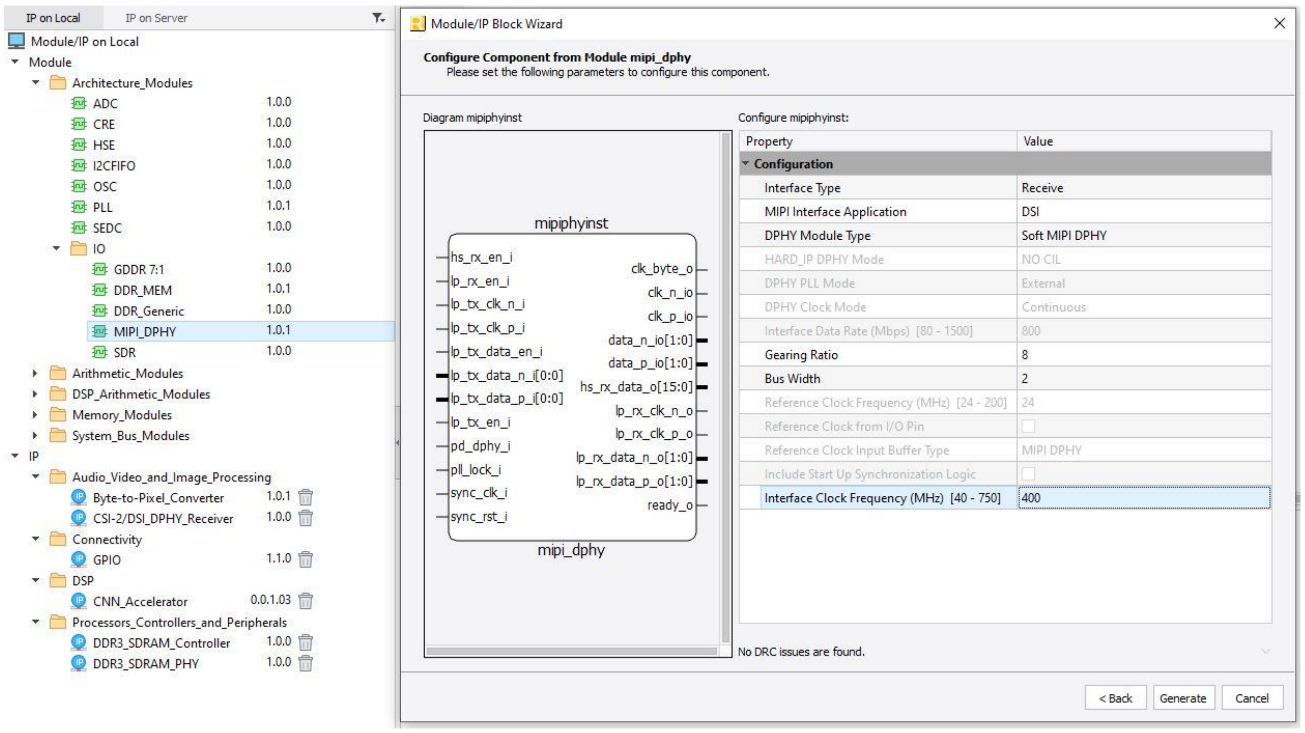
To build a MIPI D-PHY interface, select the MIPI_DPHY option under Architecture Modules – I/O in the IP Catalog tab. Enter the name of the module. Figure 6.8 shows the type of interface selected as MIPI_DPHY and the module name entered. This module can then be configured by clicking the Next button.



Generate Component from Module mipi_dphy Version 1.0.1
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Please enter the following information to get started.

Component name: mipiphyinst
Create in: _Board/versa_jedi_ddr3_demo_20200311_v1.01/user_logic/par/radiant/8bit_counter

Figure 6.8. MIPI_DPHY Option Selected in the IP Catalog Tab



Configure Component from Module mipi_dphy
Please set the following parameters to configure this component.

Configure mipiphyinst:

Property	Value
Configuration	
Interface Type	Receive
MIPI Interface Application	DSI
DPHY Module Type	Soft MIPI DPHY
HARD_IP DPHY Mode	NO CIL
DPHY PLL Mode	External
DPHY Clock Mode	Continuous
Interface Data Rate (Mbps) [80 - 1500]	800
Gearing Ratio	8
Bus Width	2
Reference Clock Frequency (MHz) [24 - 200]	24
Reference Clock from I/O Pin	☐
Reference Clock Input Buffer Type	MIPI DPHY
Include Start Up Synchronization Logic	☐
Interface Clock Frequency (MHz) [40 - 750]	400

No DRC issues are found.

Figure 6.9. MIPI_DPHY Configuration Settings Tab

Table 6.5 explains the various configurations options available for MIPI D-PHY Interface.

Table 6.5. Configuration Options for MIPI D-PHY Interface

User Interface Option	Range	Default Value
Interface Type	Receiver, Transmit	Receiver
MIPI Interface Application	DSI, CSI-2	DSI
DPHY Module Type	Soft MIPI DPHY	Soft
DPHY PLL Mode	External, Internal for Transmit Interface	External
DPHY Clock Mode	Continuous, Non-Continuous	Continuous
Interface Data Rate (Mbps)	Calculated display value: Interface clk frequency x 2, 80 -2500 Mbps	1500
Gearing Ratio	Soft MIPI DPHY : 8	8
Bus Width	1, 2, 4	1
Reference Clock Frequency (MHz)	24 MHz – 200 MHz for Transmit Interface only	100
Reference Clock from I/O Pin	Enable, Disable	Disable
Reference Clock Input Buffer Type	MIPI DPHY	MIPI DPHY
Include Start Up Sync Logic	Enable, Disable	Disable
Interface Clock Frequency (MHz)	Soft MIPI DPHY: 40 – 750	750

7. I/O Logic (DDR) User Primitives and Attributes

This section describes the user primitives for I/O logic that can be used to implement all the DDR interfaces. These primitives are divided into ones that are used to implement the DDR data and ones for DDR Strobe signal or the Source Synchronous clock.

Table 7.1. Software Primitives

Type	Primitive	Description
Input/Output Delay	DELAYA	Dynamic Input/Output Delay Element
	DELAYB	Static Input/Output Delay Element
DDR Data Input	IDDRX1	Generic ×1 IDDR
	IDDRX2	Generic ×2 IDDR
	IDDRX4	Generic ×4 IDDR
	IDDRX5	Generic ×5 IDDR
	IDDR71	7:1 LVDS IDDR
DDR Data Output	ODDRX1	Generic ×1 ODDR
	ODDRX2	Generic ×2 ODDR
	ODDRX4	Generic ×4 ODDR
	ODDRX5	Generic X5 ODDR
	ODDR71	7:1 LVDS ODDR
DDR Mem Addr/Cmd	OSHX2	CS_N for DDR3 interface
	OSHX4	CS_N for DDR3 interface
DDR Mem Data Input	IDDRX2DQ	DQ Input for DDR3 memory
	IDDRX4DQ	DQ Input for DDR3 memory
DDR Mem Data Output	ODDRX2DQ	DQ output for DDR3 memory
	ODDRX4DQ	DQ output for DDR3 memory
DDR Mem Data Tristate	TSHX2DQ	DQ tri-state control for DDR3 memory
	TSHX4DQ	DQ tri-state control for DDR3 memory
DDR Mem DQS	ODDRX2DQS	DQS Output for DDR3 memory
	ODDRX4DQS	DQS Output for DDR3 memory
DDR Mem DQS Tristate	TSHX2DQS	DQS tri-state control for DDR3 memory
	TSHX4DQS	DQS tri-state control for DDR3 memory
LPDDR4 Output Delay	OUTDELAYA	Dynamic delay module

7.1. Input/Output DELAY

The DELAY block can be used to delay the input data from the input pin to the IDDR or IREG or FPGA fabric or to delay the output data from the ODDR, OREG or FPGA fabric to the output pin. It is useful to adjust for any skews amongst the input or output data bus. It can also be used to generate skew between the bits of output bus to reduce SSO noise.

The DELAY block can be used with IDDR or ODDR modules, SDR module, as well as on the direct input to the FPGA. The DELAY is shared by the input and output paths and hence can only be used either to delay the input data or the output data on a bi-direction pin.

The data input to this block can be delayed using:

- Pre-determined delay value (for Zero Hold time, delay based on Interface Type)
- Fixed Delay values that you enter
- Counter up and down controls, which can be dynamically updated

You can optionally bypass the DELAY block completely as well.

7.2. DELAYA

By default, the DELAYA is configured to factory delay settings based on the clocking structure. You can overwrite the DELAY setting using the MOVE and DIRECTION control inputs. The LOADN resets the delay back to the default value.

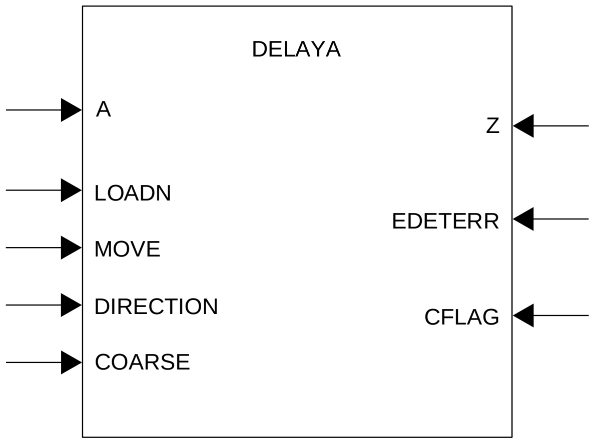


Figure 7.1. DELAYA Primitive

Table 7.2. DELAYA Port List

Port	I/O	Description
A	I	Data input from pin or output register block.
LOAD_N	I	*0* on LOADN resets to default delay setting.
MOVE	I	*Pulse* on MOVE changes delay setting. DIRECTION is sampled at the falling edge of MOVE.
DIRECTION	I	*1* to decrease delay and '0' to increase delay
COARSE[1:0]	I	Dynamic coarse delay control (2 bits) 00: no coarse delay 01: 800ps delay 10: 1600ps delay 11: invalid
Z	O	Delayed data to input register block or to pin.
EDETERR	O	Error detected when using the edge monitor logic.
CFLAG	O	Flag indicating the delay counter has reached the max (when moving up) or min (when moving down) value.

7.3. DELAYB

By default, the DELAYB is configured to factory delay settings based on the clocking structure. You cannot change the delay when using this module.

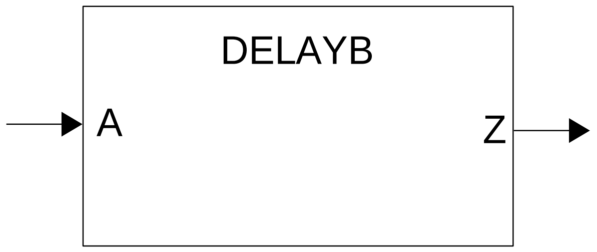


Figure 7.2. DELAYB Primitive

Table 7.3. DELAYB Port List

Port	I/O	Description
A	I	Data input from pin or output register block
Z	O	Delayed data to input register block or to pin

7.4. DELAY Attribute Description

Table 7.4 describes the attributes available for the DELAYA and DELAYB elements.

Table 7.4. DELAYA and DELAYB Attributes

Attribute	Description	Values[1, 2, 3]	Default	DELAY
DEL_MODE	Sets the delay mode to be used	USER_DEFINED SCLK_ZEROHOLD ECLK_ALIGNED ECLK_CENTERED SCLK_ALIGNED SCLK_CENTERED DQS_CMD_CLK DQS_ALIGNED_X2 DQS_ALIGNED_X4 DQS_CENTERED_X2 DQS_CENTERED_X4	USER_DEFINED	DELAYA DELAYB
DEL_VALUE	Sets delay value when DEL_MODE is set to USER_DEFINED	0..127	0	DELAYA DELAYB Step is 12.5ps
COARSE_DELAY_MODE	Select MC1 or CIB coarse delay code control; 0–Selects from MC1 (STATIC), 1–Selects from CIB (DYNAMIC)	STATIC DYNAMIC	STATIC	DELAYA
COARSE_DELAY	Coarse delay setting for lol delay cell	0NS 0P8NS 1P6NS	0NS	DELAYA
EDGE_MONITOR	To enable edge monitor when in an IDDR ×2, ×7to1, ×4, or ×5 mode	ENABLED; DISABLED	ENABLED	DELAYA
WAIT_FOR_EDGE	Used for SPI4.2 implementation	ENABLED; DISABLED	ENABLED	DELAYA

Notes:

1. DQS_CMD_CLK is only for the DDR Memory CMD and CLK outputs.
2. DQS_ALIGNED_x2/x4 is shared by DQS generic and the DDR memory inputs.
3. DQS_CENTERED_x2/x4 is used for DQS generic inputs.

7.5. DDRDLL (Master DLL)

The DDRDLL is used to generate a 90-degree delay for the DQS Strobe Input during a memory interface or for the clock input for a generic DDR interface.

There is one DDRDLL module on each corner of the device. The DDRDLL outputs delay codes that are used in the DLLDEL module to delay the input clock. DDRDLL, by default, generates 90-degree phase shift.

7.5.1. DDRDLLA

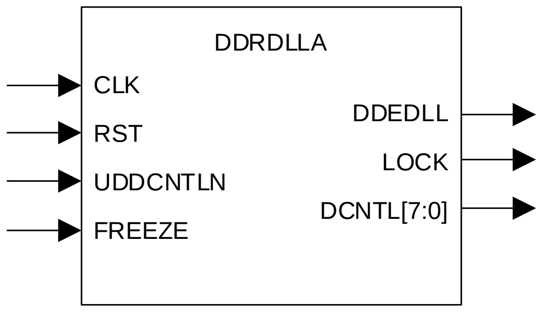


Figure 7.3. DDRDLLA Primitive

Table 7.5. DDRDLLA Port List

Port	I/O	Description
CLK	I	Reference clock input to the DDRDLL. Should run at the same frequency as the clock to be delayed.
RST	I	Reset input to the DDRDLL.
UDDCNTLN	I	Update control to update the delay code. When low, the delay code out of the DDRDLL is updated. Should not be active during a read or a write cycle.
FREEZE	I	Releases the DDRDLL input clock.
DDRDEL	O	The delay codes from the DDRDLL to be used in DLLDEL.
LOCK	O	Lock output to indicate the DDRDLL has valid delay output.
DCNTL [7:0]	O	The delay codes from the DDRDLL available for the user IP.

Table 7.6. DDRDLL Attributes

Attribute	Description	Values	Default
FORCE_MAX_DELAY[1]	Bypass DLL locking procedure at low frequency.	YES, NO	NO

Note:

1. When Fin is =<30 MHz. The software sets Force_max_delay to YES. DDRDLL does not go through the locking process, but is locked to maximum delay steps under such conditions.

7.6. DLL Delay (DLLDEL)

The DLLDEL receive delay codes from the DDRDLL and generates a delayed clock output.

The DLLDEL receives delay from the DDRDLL. The delayed clock output of the DLLDEL can be connected to the IDDR module.

The delay from the DLLDEL can be dynamically adjusted using counter margin control signals that can shift the delay up or down DLLDELD.

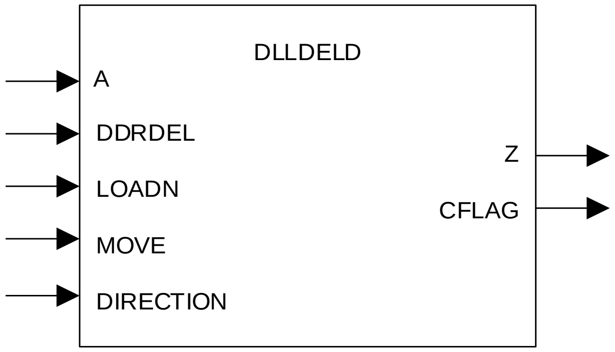


Figure 7.4. DLLDELD Primitive

Table 7.7. DLLDELD Port List

Port	I/O	Description
A	I	Clock input.
DDRDEL	I	Delay inputs from DDRDLL.
LOADN	I	Used to reset back to 90-degree delay.
MOVE	I	Pulse is required to change delay settings. The value on Direction is sampled at the falling edge of MOVE.
DIRECTION	I	Indicates delay direction. *1* to decrease delay and *0* to increase delay.
CFLAG	O	Indicates the delay counter has reached its maximum value when moving up or minimum value when moving down.
Z	O	Delayed clock output.

Table 7.8. DLLDELD Attributes[1]

Attribute	Description	Values	Default
DEL_ADJ[2]	Sign bit for READ delay adjustment, DDR input	PLUS, MINUS	PLUS
DEL_VAL[2]	Value of delay for input DDR.	0 to 255 (PLUS) 1 to 256 (MINUS)	Note[2]

Notes:

1. Attributes are only available through EPIC and ECL Editor. It is recommended that values of this attribute are not updated without consulting Lattice Semiconductor Technical Support.
2. Default value is set based on device characterization to achieve the 90-degree phase shift.

7.7. Input DDR Primitives

The following are the primitives used to implement various Generic DDR Input configurations.

7.7.1. IDDRX1

This primitive is used for the Generic x1 IDDR implementation.

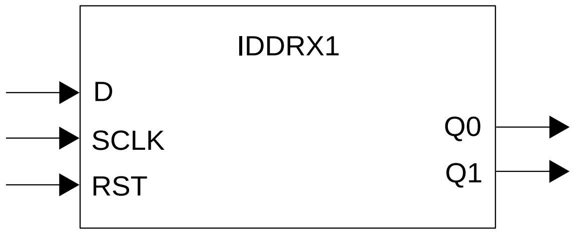


Figure 7.5. IDDRX1 Primitive

Table 7.9. IDDRX1 Port List

Port	I/O	Description
D	I	DDR data input
SCLK	I	Primary Clock input
RST	I	Reset to DDR registers
Q0	O	Data at the positive edge of the clock
Q1	O	Data at the negative edge of the clock

7.7.2. IDDRX2

This primitive is used for the Generic x2 IDDR implementation.

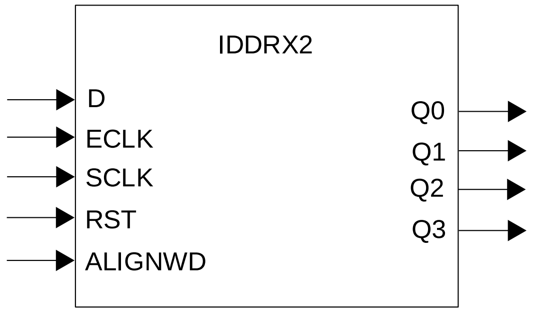


Figure 7.6. IDDRX2 Primitive

Table 7.10. IDDRX2 Port List

Port	I/O	Description
D	I	DDR data input
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RST	I	Reset to DDR registers
ECLK	I	Fast edge clock
ALIGNWD	I	This signal is used for word alignment. It shifts the word by one bit.
Q[3:0]	O	Parallel data output. Q0, Q2 are the data at the positive edges of the input ECLK. Q1 and Q3 are data at negative edge of input ECLK.

7.7.3. IDDRX4

This primitive is used for the Generic x4 IDDR implementation.

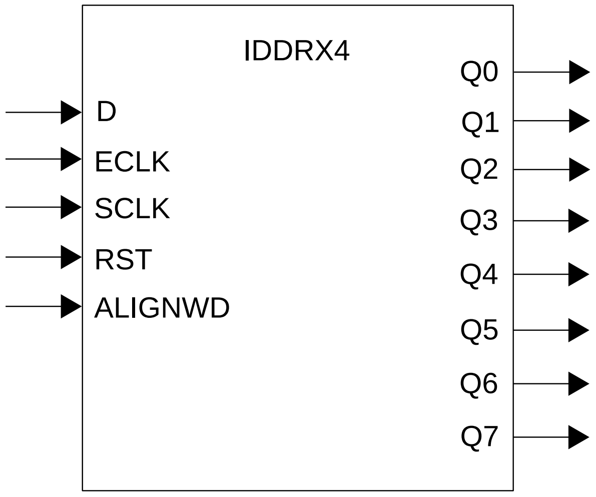


Figure 7.7. IDDRX4 Primitive

Table 7.11. IDDRX4 Port List

Port	I/O	Description
D	I	DDR data input
SCLK	I	Primary clock input (divide-by-4 of ECLK)
RST	I	Reset to DDR registers
ECLK	I	Fast edge clock
ALIGNWD	I	This signal is used for word alignment. It shifts the word by one bit.
Q[7:0]	O	Parallel data output. Q0, Q2, Q4, Q6 are the data at the positive edges of the input ECLK. Q1, Q3, Q5, Q7 are data at negative edge of input ECLK.

7.7.4. IDDRX5

This primitive is used for the Generic x5 IDDR implementation.

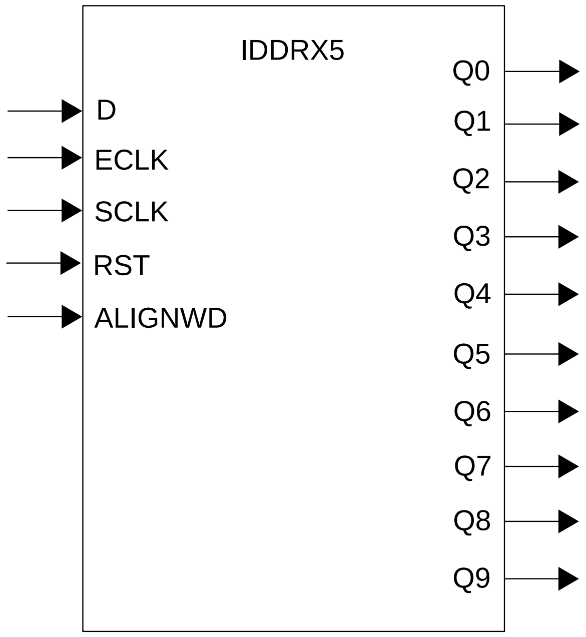


Figure 7.8. IDDRX5 Primitive

Table 7.12. IDDRX5 Port List

Port	I/O	Description
D	I	DDR data input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-5 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for word alignment. It shifts the word by one bit.
Q[9:0]	O	Parallel data output. Q0, Q2, Q4, Q6, Q8 are the data at the positive edges of the input ECLK. Q1, Q3, Q5, Q7, Q9 are data at negative edge of input ECLK.

7.7.5. IDDR71

This primitive is used for 7:1 LVDS input side implementation.

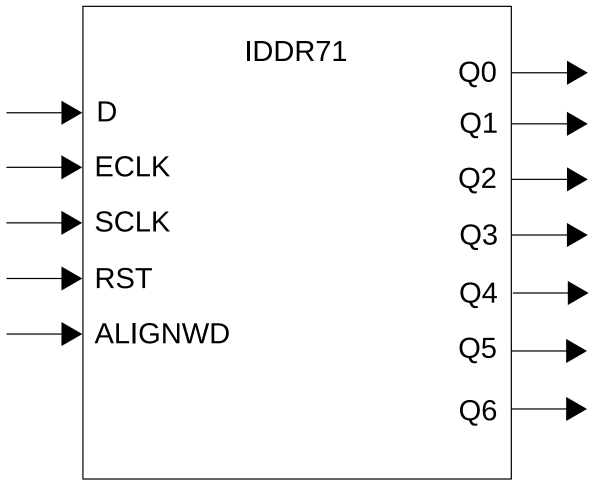


Figure 7.9. IDDR71

Table 7.13. IDDR71 Port List

Port	I/O	Description
D	I	DDR data input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-3.5 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for word alignment. It shifts the word by one bit.
Q[6:0]	O	7 bits of output data.

7.8. Output DDR Primitives

The following are the primitives used to implement various Generic ODDR output configurations.

7.8.1. ODDRX1

This primitive is used for Generic x1 ODDR implementation.

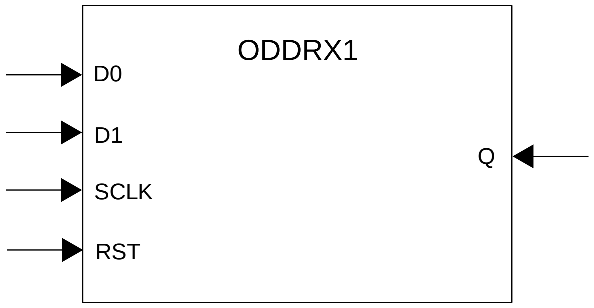


Figure 7.10. ODDRX1

Table 7.14. ODDRX1F Port List

Port	I/O	Description
D0, D1	I	Parallel data input to ODDR (D0 is sent out first then D1)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of SCLK

7.8.2. ODDRX2

This primitive is used to receive Generic x2 ODDR implementation.

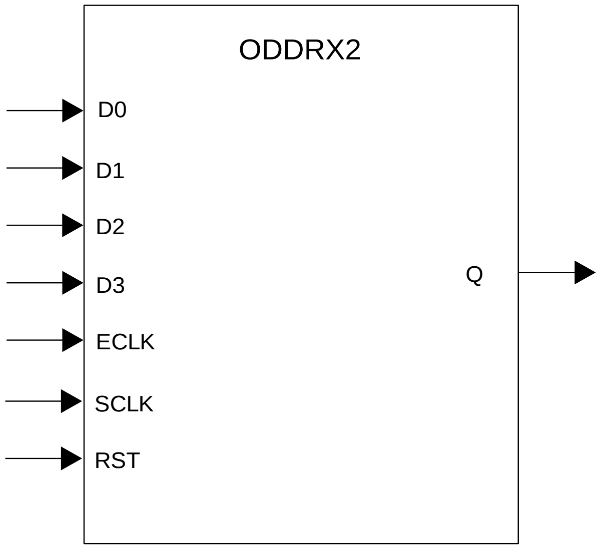


Figure 7.11. ODDRX2

Table 7.15. ODDRX2 Port List

Port	I/O	Description
D0, D1, D2, D3	I	Parallel Data input to the ODDR (D0 is sent out first and D3 last)
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RST	I	Reset to DDR registers
Q	O	DDR data output on both edges of ECLK

7.8.3. ODDRX4

This primitive is used for the Generic x4 ODDR implementation.

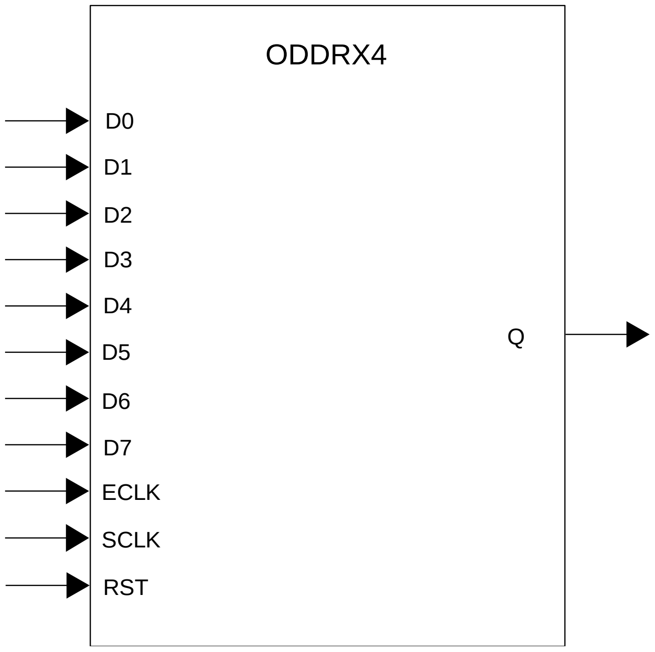


Figure 7.12. ODDRX4 Primitive

Table 7.16. ODDRX4 Port List

Port	I/O	Description
D0, D1, D2, D3, D4, D5, D6, D7	I	Parallel Data input to the ODDR (D0 is sent out first and D7 last)
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-4 of ECLK)
RST	I	Reset to DDR registers
Q	O	DDR data output on both edges of ECLK

7.8.4. ODDRX5

This primitive is used for the Generic x5 ODDR implementation.

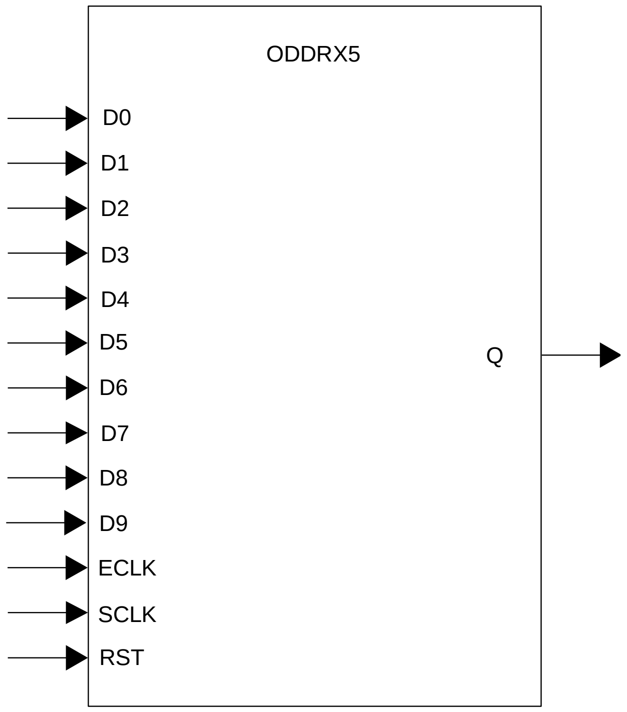


Figure 7.13. ODDRX5 Primitive

Table 7.17. ODDRX5 Port List

Port	I/O	Description
D0, D1, D2, D3, D4, D5,D6, D7, D8, D9	I	Parallel Data input to the ODDR (D0 is sent out first and D9 last)
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-5 of ECLK)
RST	I	Reset to DDR registers
Q	O	DDR data output on both edges of ECLK

7.8.5. ODDR71

This primitive is used for 7:1 LVDS ODDR implementation.

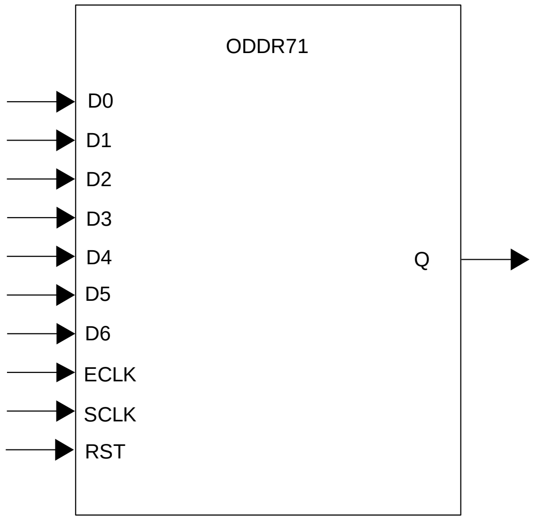


Figure 7.14. ODDR71 Primitive

Table 7.18. ODDR71 Port List

Port	I/O	Description
D0, D1, D2, D3, D4,D5,D6	I	Parallel Data input to the ODDR (D0 is sent out first and D6 last)
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-3.5 of ECLK)
RST	I	Reset to DDR registers
Q	O	DDR data output on both edges of ECLK

8. Soft IP Modules

The following soft IP Modules are available for use with the Generic DDR interfaces described above. All of the soft IP Modules can be generated using IP Catalog. Table 8.1 summarizes the list of soft IPs available and the ones that are optional versus the ones that are automatically generated with the interface in IP Catalog.

Table 8.1. List of Soft IPs Supported

Soft IP Name	Function	Required
RX_SYNC	Used to break up the DDRDLL to DLLDEL clock loop for Aligned Interfaces.	Yes
GDDR_SYNC	Needed to tolerate large skew between stop and reset input.	Yes
7:1 LVDS Bit and Word Alignment (BW_ALIGN)	The soft IP is used to perform bit and word alignment using PLL dynamic phase shift interface and aligned input of IDDR71C.	Optional
MIPI_FILTER	Implements low pass filter on low speed MIPI data	Optional

8.1. Detailed Description of Each Soft IP

8.1.1. GDDR_SYNC

This module is needed for startup all RX Centered and all TX interfaces with 2× gearing.

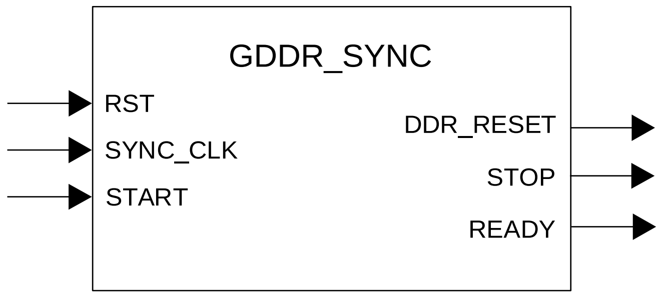


Figure 8.1. GDDR_SYNC Ports

Table 8.2. GDDR_SYNC Ports Description

Port	I/O	Description
SYNC_CLK	I	Startup clock. It can be a low speed continuously running clock, for example, an oscillator clock. It cannot be the RX_CLK or divided version.
RST	I	Active high reset to this sync circuit. When RST=1, STOP=0, DDR_RESET=1, READY=0
START	I	Start sync process. This is used to wait for PLL lock, then start sync process in 7:1 LVDS interface.
STOP	O	Connect to ECLKSYNC.STOP.
DDR_RESET	O	Reset to all IDDRX or ODDRX components and CLKDIV.
READY	O	Indicate that startup is finished and RX circuit is ready to operate.

8.1.2. RX_SYNC

This module is needed for startup RX-aligned interfaces with 2× gearing.

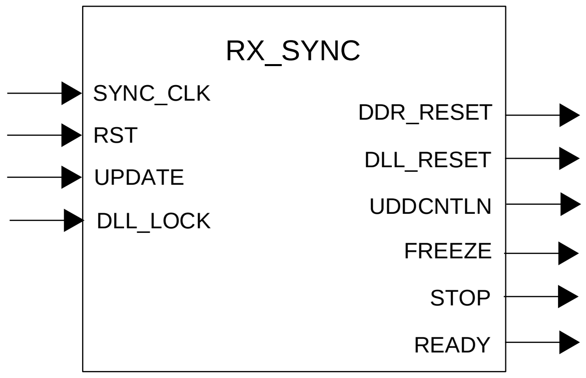


Figure 8.2. RX_SYNC Ports

Table 8.3. GDDR_SYNC Ports Description

Port	I/O	Description
SYNC_CLK	I	Startup clock. It can be a low speed continuously running clock, for example, an oscillator clock. It cannot be the RX_CLK or divided version
RST	I	Active high reset to this sync circuit. When RST=1, STOP=0, FREEZE=0, UDDCNTLN=1, DLL_RESET=1, DDR_RESET=1, READY=0.
DLL_LOCK	I	LOCK output from DDRDLL.
UPDATE	I	UPDATE can be used to re-start sync process. READY goes low and waits for the sync process to be completed before going high again. This can only be performed when no traffic is present.
STOP	O	Connect to ECLKSYNC.STOP.
FREEZE	O	Connect to DDRDLL.FREEZE.
UDDCNTLN	O	Connect to DDRDLL.UDDCNTLN.
DLL_RESET	O	Reset to DDRDLL.
DDR_RESET	O	Reset to all IDDRX components and CLKDIV.
READY	O	Indicate that startup is finished and RX circuit is ready to operate.

8.1.3. BW_ALIGN

This module is used to perform 7:1 video RX bit and word alignment. This module is optional and can be enabled in IP Catalog.

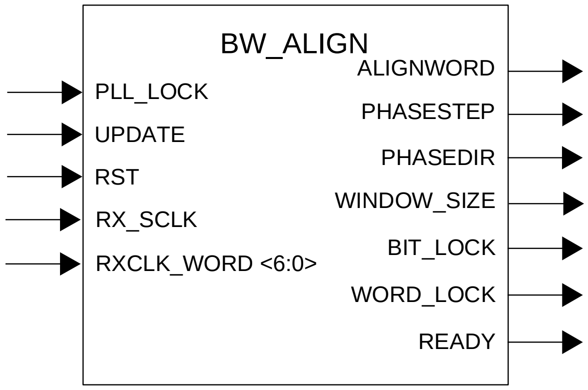


Figure 8.3. BW_ALIGN Ports

Table 8.4. BW_ALIGN Port Description

Port	I/O	Description
RX_SCLK	I	Divided RX clock from the 7:1 RX interface, produced by CLKDIV.
RST	I	Active high reset to this circuit. When RST=1, All outputs=0.
PLL_LOCK	I	Connect to PLL LOCK output. Start the alignment procedures after PLL lock goes high.
UPDATE	I	Start the procedure, or re-start if need to optimize again.
RXCLK_WORD<6:0>	I	Parallel data output from the 2nd IDDRX71 attached to RX CLK Input.
PHASESTEP	O	Rotate phase for PLL.
PHASEDIR	O	Phase rotation direction for PLL, fixed to forward (0) for this design.
ALIGNWORD	O	Connect to IDDRX71.ALIGNWORD, for word rotation.
WINDOW_SIZE	O	Final valid window size.
BIT_LOCK	O	Status output, bit lock is achieved.
WORD_LOCK	O	Status output, word lock is achieved.
READY	O	Indicate that alignment procedure is finished and RX circuit is ready to operate.

With Bit Alignment, the goal is to place Edge Clock (under PLL dynamic phase shift control) to the center of valid window for the clock word and data words. The PLL phase rotation goes through all 16 phases. The PLL high-speed output is used to sample RX input clock. Transitions are detected on the second IDDR71 output, which inputs the RX Clock and phases close to transition are identified. The IP chooses the phase most away from transition as the final phase to use.

The low speed clock has two transitions per 7-bit word. It is not the worst case in terms of inter-symbol interference. On the other hand, we do have 8 possible sample points per bit period. Minimum eye-opening of 3/8 UI is needed to achieve lock. Jitter tolerance is around 0.25 UI, about 300 ps at 756 Mb/sec.

After bit alignment is achieved, word alignment is needed, so video data (in 7-bit words) can be processed in core. The IP uses the ALIGNWD function of the IDDRX71 primitive for word alignment. Each pulse on ALIGNWD rotates the 7-bit bus by 2 bits. In maximum 7 ALIGNWD operations, the word loops through all seven possibilities. The goal is to get 7’b1100011 (7’h63) in the clock word. The clock word is the clock (4 bit 1 and 3’b 0) converted to parallel data, exactly as the video data traffic. For the 7:1 video, the RX input Clock serves as:

- Frequency reference to generate high-speed.
- Phase reference as source synchronized RX, since the clock is edge aligned with the data bits.
- Word alignment reference.

8.1.4. MIPI_FILTER

This module is needed to filter low speed signal for MIPI RX. It filters out narrow pulses. It allows pulse width above 40 ns to pass.

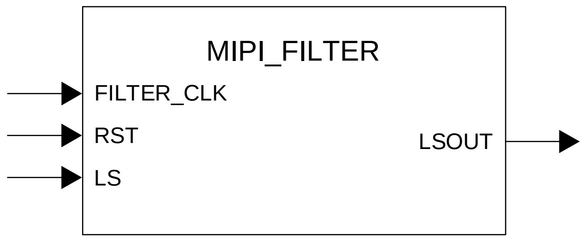


Figure 8.4. MIPI_FILTER Ports

Table 8.5. MIPI_FILTER Port Description

Port	I/O	Description
FILTER_CLK	I	Clock used to drive digital filter. Min freq=100 MHz. Recommendation is to use internal oscillator at 133 MHz.
LS	I	Low speed signal from MIPI PHY.
RST	I	Active high reset. When RST=1, LSOUT=0.
LS_OUT	O	Filtered output signal.

cutoff	Parameter	Parameterize the circuit, default=4, pass signal above 4 cycles.
		cutoff=round (20 ns/period (filter_clk)).
		Filter_clk=100 MHz, cutoff=4.
		Filter_clk=133 MHz, cutoff=6.
		Filter_clk=175 MHz, cutoff=7.
		Filter_clk=200 MHz. cutoff=8.

References

- CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104)
- CrossLink-NX website

For more information on CrossLink-NX-related IP, reference designs, and board documents, refer to the following web pages:

- IP and Reference Designs for CrossLink-NX
- Development Kits and Boards for CrossLink-NX

A variety of technical notes for the LIFCL-33, LIFCL-33U, and CrossLink-NX devices are available.

- CrossLink-NX-33 and CrossLinkU-NX Hardware Checklist (FPGA-TN-02308)
- sysI/O Usage Guide for Nexus Platform (FPGA-TN-02067)
- sysCLOCK PLL/DLL Design and Usage Guide for Nexus Platform (FPGA-TN-02095)
- sysCONFIG Usage Guide for Nexus Platform (FPGA-TN-02099)
- Memory Usage Guide for Nexus Platform (FPGA-TN-02094)
- CrossLink-NX High-Speed I/O Interface (FPGA-TN-02097)
- Power Management and Calculation for CrossLink-NX Devices (FPGA-TN-02075)
- sysDSP Usage Guide for Nexus Platform (FPGA-TN-02096)

Other references:

- Lattice Insights for Lattice Semiconductor training courses and learning plans
- Lattice Radiant FPGA design software

Technical Support Assistance

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

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

Revision History

Revision 0.84, March 2024

Section	Change Summary
High-Speed DDR Interface Details	• Removed reference to Timing Analysis for High Speed DDR Interfaces section. • Removed interface requirement referencing Timing Analysis for High Speed DDR Interfaces section in GDDRX1_RX.SCLK.Centered, GDDRX1_RX.SCLK.Aligned, GDDRX2_RX.ECLK.Centered, GDDRX2_RX.ECLK.Aligned, GDDRX4_RX.ECLK.Centered, GDDRX4_RX.ECLK.Aligned, GDDRX5_RX.ECLK.Centered, GDDRX5_RX.ECLK.Aligned, GDDRX2_TX.ECLK.Aligned, GDDRX2_TX.ECLK.Centered, GDDRX4_TX.ECLK.Aligned, GDDRX4_TX.ECLK.Centered, GDDRX5_TX.ECLK.Aligned, GDDRX5_TX.ECLK.Centered, and GDDRX71_TX.ECLK sections. • Removed Timing Analysis for High Speed DDR Interfaces section.

Revision 0.83, February 2024

Section	Change Summary
All	• Updated the document reference for the CrossLink-NX-33 and CrossLinkU-NX Data Sheet to reflect the correct title across the document. • Removed LIFCL-33 Memory Interfaces section.
Introduction	Removed DDR/LPDDR references in this section.
High-Speed I/O Interface Building Blocks	• Updated this section, including Figure 3.1. LIFCL-33/33U Device Clocking Diagram to remove DQS_BUF blocks/information. • Updated Input DDR (IDDR) section to remove 10:1 gearing mode and add 1-bit DDR information. • Updated Output DDR (ODDR) section to remove 10:1 gearing mode, add 5x and 10-bit information. • Removed DQS Lane and DQSBUF sections.
High-Speed DDR Interface Details	Removed DQSBUF in Receive Interface Guidelines section.
Using IP Catalog to Build and Plan High Speed DDR Interfaces	Removed Configuring DDR Memory Interfaces (LIFCL-33 Only) section.
I/O Logic (DDR) User Primitives and Attributes	• Removed DQSBUF information in this section, including Table 7.5. DDRDLLA Port List. • Removed DSQBUF (DQS Strobe Control Block), IDDR/ODDR Modules for Memory DDR Implementation (LIFCL-33 Only), Memory Output DDR Primitives for DQS Output (LIFCL-33 Only), Memory Output DDR Primitives for Tristate Output Control (LIFCL-33 Only), Memory Output DDR Primitives for Address and Command (LIFCL-33 Only), and LPDDR4 Output Delay Primitives (LIFCL-33 Only) sections.
Soft IP Modules	• Updated Table 8.1. List of Soft IPs Supported to remove MEM_SYNC. • Removed Soft IP Used in Each Interfaces table and MEM_SYNC section.
References	Updated the document reference for the CrossLink-NX-33 and CrossLinkU-NX Data Sheet and Hardware Checklist to reflect the correct titles.

Revision 0.82, October 2023

Section	Change Summary
All	• Updated document title to CrossLink-NX-33 and CrossLinkU-NX High-Speed I/O Interface. • Used LIFCL-33/33U to refer to the CrossLink-NX-33 and CrossLinkU-NX devices.

Revision 0.81, September 2023

Section	Change Summary
All	• Updated document title to *CrossLink-NX-33 and CrossLink-NX-33U High-Speed I/O Interface*.

Section	Change Summary
	• Added CrossLink-NX-33U information. • Indicated contents that apply only to CrossLink-NX-33 devices. • Updated reference to the combined CrossLink-NX-33 and CrossLink-NX-33U Data Sheet.
Introduction	Added CrossLink-NX-33U support.
High-Speed I/O Interface Building Blocks	• Added CrossLink-NX-33U support. • Added Table 3.2. CrossLink-NX-33U I/O Banks in the I/O Banks section. • Indicated contents that apply only to CrossLink-NX-33 devices.
Building Generic High Speed Interfaces	Added CrossLink-NX-33U support.
High-Speed DDR Interface Details	Added CrossLink-NX-33U support. Added references to the combined CrossLink-NX-33 and CrossLink-NX-33U Data Sheet.
Using IP Catalog to Build and Plan High Speed DDR Interfaces	Indicated contents that apply only to CrossLink-NX-33 devices.
I/O Logic (DDR) User Primitives and Attributes	Indicated contents that apply only to CrossLink-NX-33 devices.
References	Added this section.

Revision 0.80, June 2022

Section	Change Summary
All	Preliminary release.