CrossLink-NX High-Speed I/O Interface

Technical Note

FPGA-TN-02097-1.3

July 2020

Disclaimers

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

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CLKDIV	Edge Clock Dividers
DDR	Double Data Rate
DLL	Delay-Locked Loops
DM	Data Mask
DSP	Digital Signal Processing
IDDR	Input DDR
ECLK	Edge Clock
ODDR	Output DDR
PCB	Printed Circuit Board
PCLK	Primary Clock
PIO	Programmable I/O
SCLK	Serial Clock
SDR	Single Data Rate
SSN	Simultaneous Switching Noise

1. Introduction

Lattice Semiconductor CrossLink™-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. CrossLink-NX device I/O also have dedicated circuitry that is used along with the DDR I/O to support DDR3, DDR3L, LPDDR2, and LPDDR3 SDRAM memory interfaces.

This document discusses how to utilize the capabilities of the CrossLink-NX devices to implement high-speed Generic DDR interface and the DDR memory interfaces. Refer to the Implementing DDR Memory Interfaces section of this document for more information.

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 also sometimes called edge-on-edge.

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

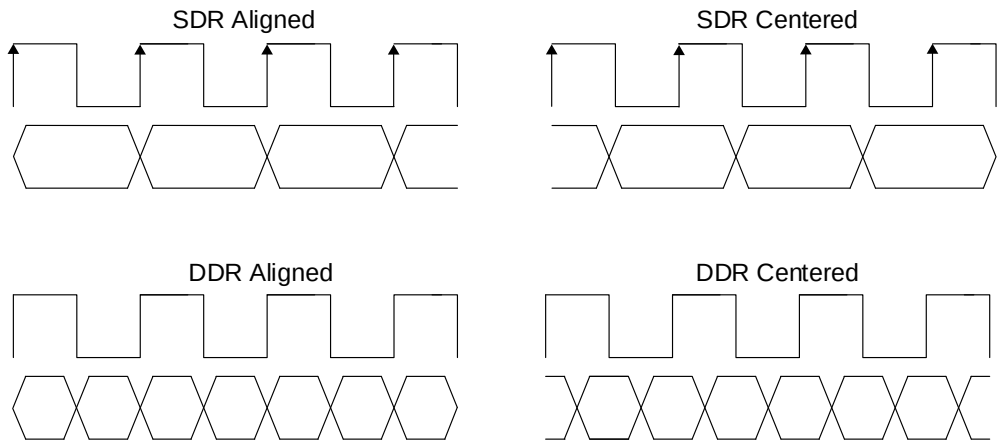


Figure 2.1. External Interface Definitions

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 CrossLink-NX device contains 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 CrossLink-NX device for building high-speed I/O interfaces.

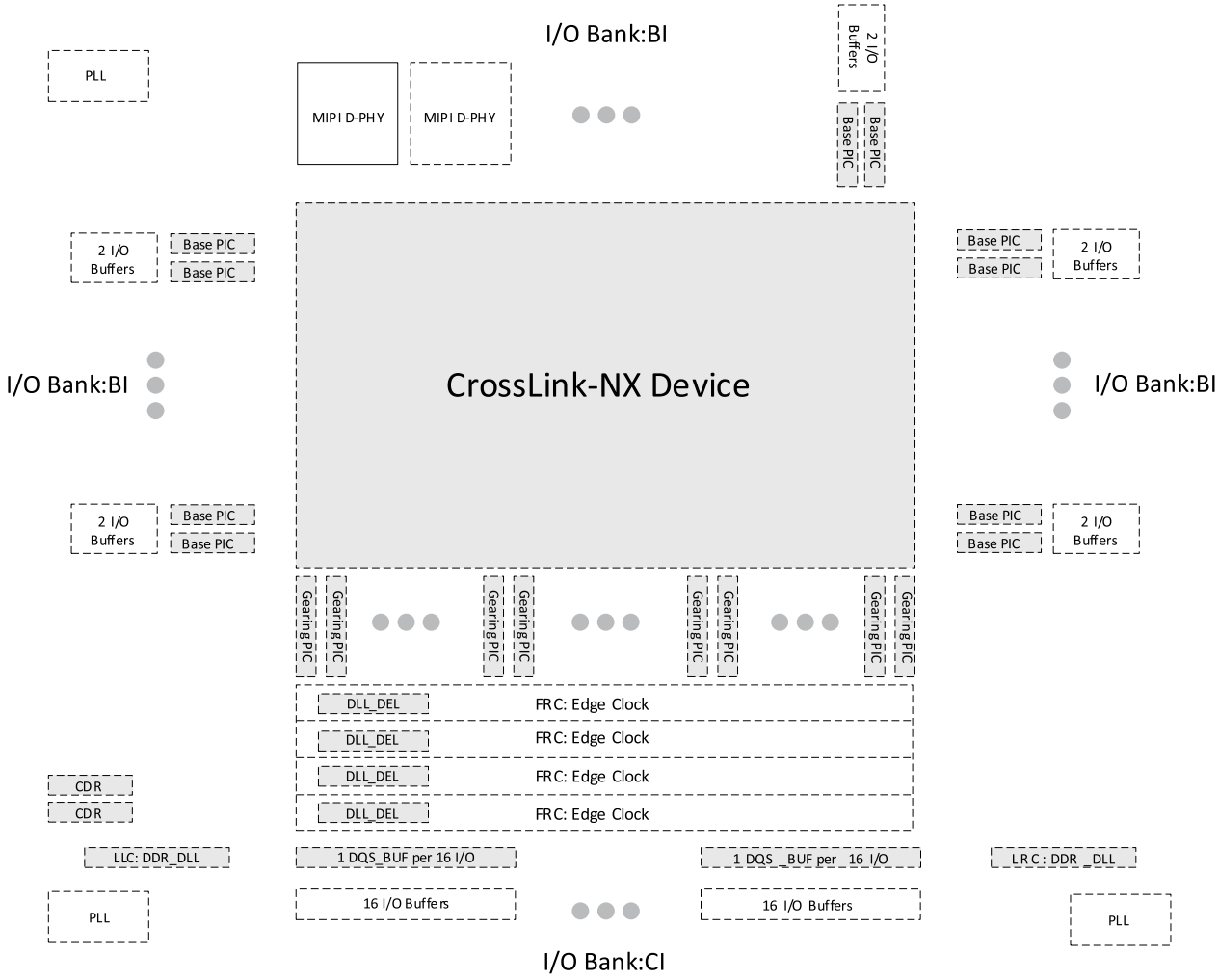


Figure 3.1. CrossLink-NX Device Clocking Diagram

In Figure 3.1, the locations of the Base PIC, Gearing PIC, DDR_DLL, DLL_DELAY, and DQS_BUF are provided. One DQS_BUF is provided per 16 I/O. One DLL_DELAY is provided for each of the four FRC: Edge Clocks. Two DDR_DLLs are provided, one at the lower left hand corner and one at the lower right hand corner. I/O bank B1 uses Base PICs and bank C1 uses Gearing PICs.

3.1. SGMII Clock Data Recovery (CDR)

The CrossLink-NX device includes two hardened CDR (Clock Data Recovery) components. The CDRs enable SGMII (Serial Gigabit Media Independent Interface) solutions. There are three main blocks in each CDR: the CDR, deserializer, and FIFO. Each CDR features two loops. The first loop is locked to the reference clock. Once locked, the loop switches to the data path loop where the CDR tracks the data signals to generate the correcting signals needed to achieve and maintain phase lock with the data. The data is then passed through a deserializer, which deserializes the data to 10-bit parallel data. The 10-bit parallel data is then sent to the FIFO bridge, which allows the CDR to interface with the rest of the FPGA. Figure 3.2 shows a block diagram of the SGMII CDR IP.

The two hardened blocks are located at the bottom left of the chip and use the high-speed I/O Bank 5 for the differential pair input. It is recommended that the reference clock be entered through a GPIO that is connected to the PLL on the lower left corner as well.

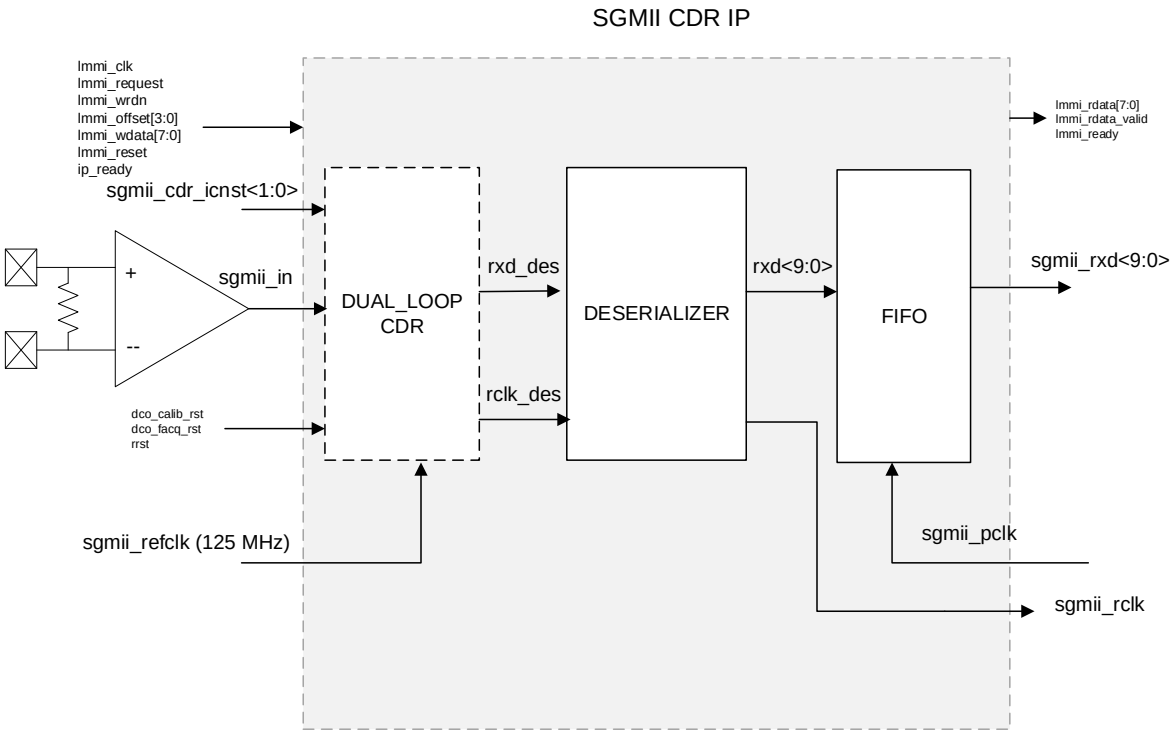


Figure 3.2. CrossLink-NX SGMII CDR IP

3.2. I/O Banks

There are eight banks in all CrossLink-NX devices. The banks are named in a clockwise direction starting from the top right.

Table 3.1. CrossLink-NX I/O Banks

Bank ID	I/O Reference	Comments
BANK 0	3.3 V/1.8 V/1.2 V	GPIO33
BANK 1	3.3 V/1.8 V/1.2 V	GPIO33
BANK 2	3.3 V/1.8 V/1.2 V	GPIO33
BANK 3	1.8 V/1.2 V/1.0 V	DIFFIO
BANK 4	1.8 V/1.2 V/1.0 V	DIFFIO
BANK 5	1.8 V/1.2 V/1.0 V	DIFFIO
BANK 6	3.3 V/1.8 V/1.2 V	GPIO33
BANK 7	3.3 V/1.8 V/1.2 V	GPIO33

3.3. Clock Scheme

A complete description of the CrossLink-NX device family clocking resources and clock routing restrictions are available in CrossLink-NX sysClock PLL/DLL Design and Usage Guide (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.

3.4. Primary Clocks

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

3.5. 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, and it can be cascaded together to form a wider high-speed interface. An Edge Clock Bridge (ECLKBRIDGECS) allows you to build large interfaces by bridging the edge clocks from one bank to the other at the bottom side of the chip.

3.6. DQS Lane

A DQS Lane uses the embedded circuit for memory interfaces. Each DQS Lane provides a clock pair (DQSP and DQSN) for the DQS strobe and up to 16 pins (8 DQ + DM+dedicated DQS/DQSb) for DQ data and DM data mask signals. The number of DQS Lanes on the device is different for each device size. CrossLink-NX device support DQS lanes on bottom side of the device.

3.7. 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. There is one PLL in upper left corner and bottom two corners on the CrossLink-NX 40K device, totaling to three PLLs for CrossLink-NX 40K device. For CrossLink-NX 17K device, there are only two PLLs located at the bottom corners.

3.7.1. PLL to PCLK

A PLL pattern can be configured by Software to utilize the Primary Clock Tree. A source of the clock signal first drives the PLL's reference mux. From there, the PLL outputs to the midmux, which feeds into the center mux. The clock signal is distributed along the center mux network, to the spine, tap drivers, then to the PLC. Afterwards, the signal is routed back to the PLL with a delay shift. The PLL takes this delay shift and adjusts the signal accordingly to stabilize the clocking signal.

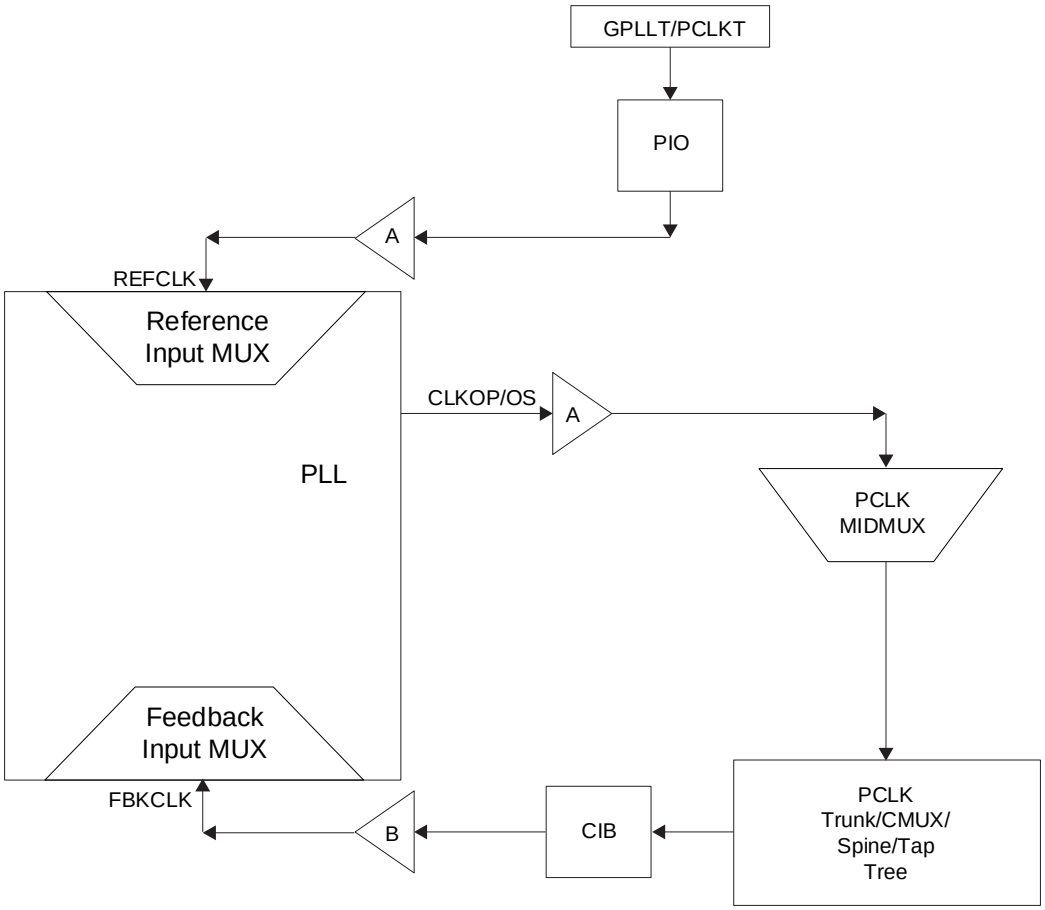


Figure 3.3. PLL to PCLK

3.7.2. PLL to ECLK

In the case of high-speed DDR applications, the PLL pattern can be configured by software to utilize the Edge Clock Tree. This time the PLL signal is sent directly to the ECLK, which is distributed to the bottom bank I/O. A dedicated feedback path is provided in the ECLK.

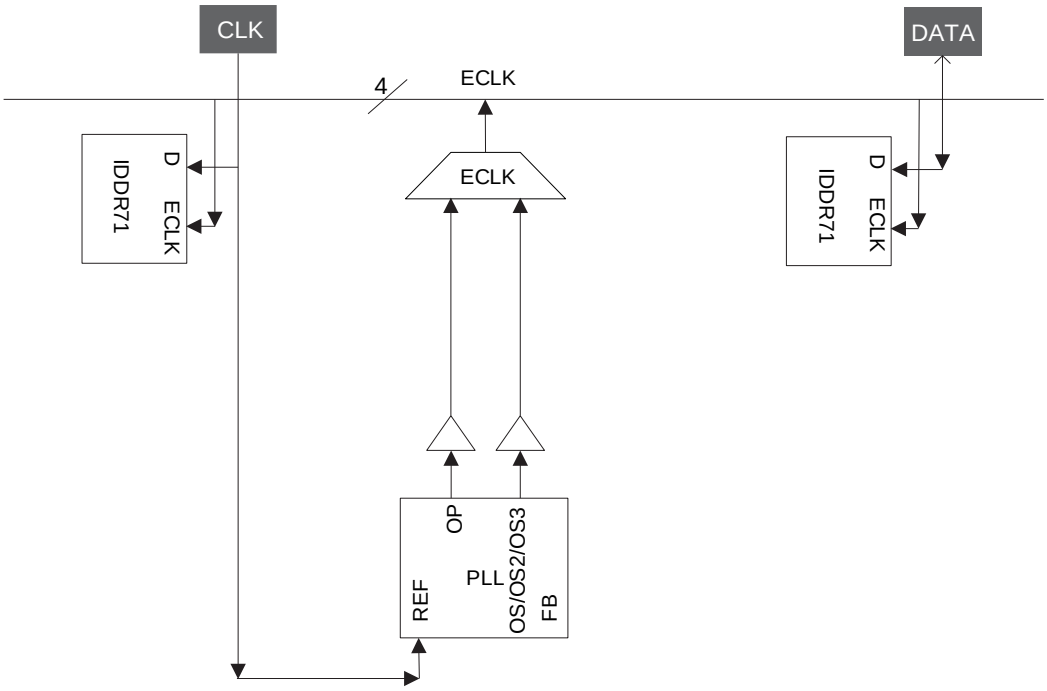


Figure 3.4. PLL to ECLK

3.8. DDRDLL

The DDRDLL is a dedicated DLL for creating the 90-degree clock delay. The DDRDLL outputs delay codes that are used in the DQSBUF elements to delay the DQS input or in the DLLDEL module to delay the input clock. There are two DDRDLL at bottom corners of the CrossLink-NX device. The DDRDLL on the bottom corners of the device can drive delay codes to two adjacent edges of the device, providing a possible two DDRDLL codes for an edge.

3.9. DLLDEL

DLLDEL provides phase shift on the receive side clocks to each ECLK, shifting the clock input by 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* 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.

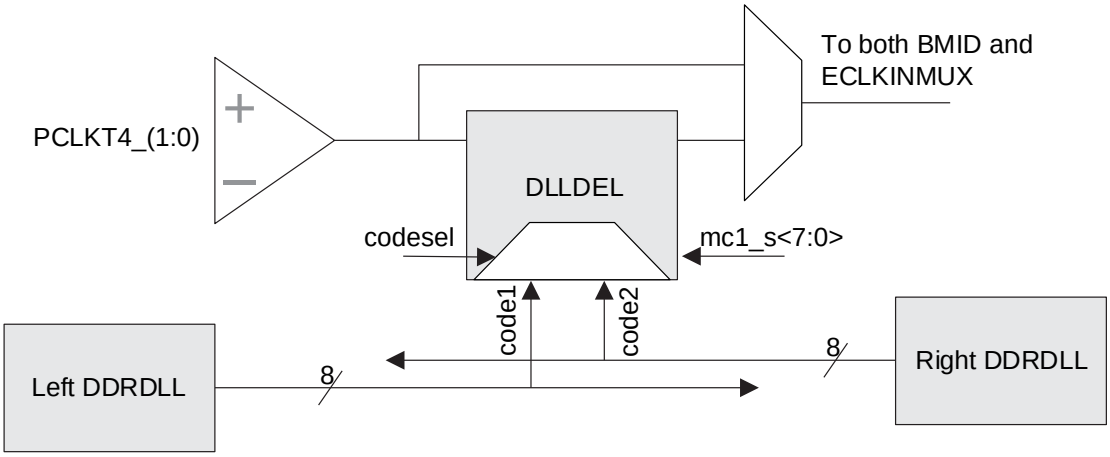


Figure 3.5. DLLDEL and Code Control

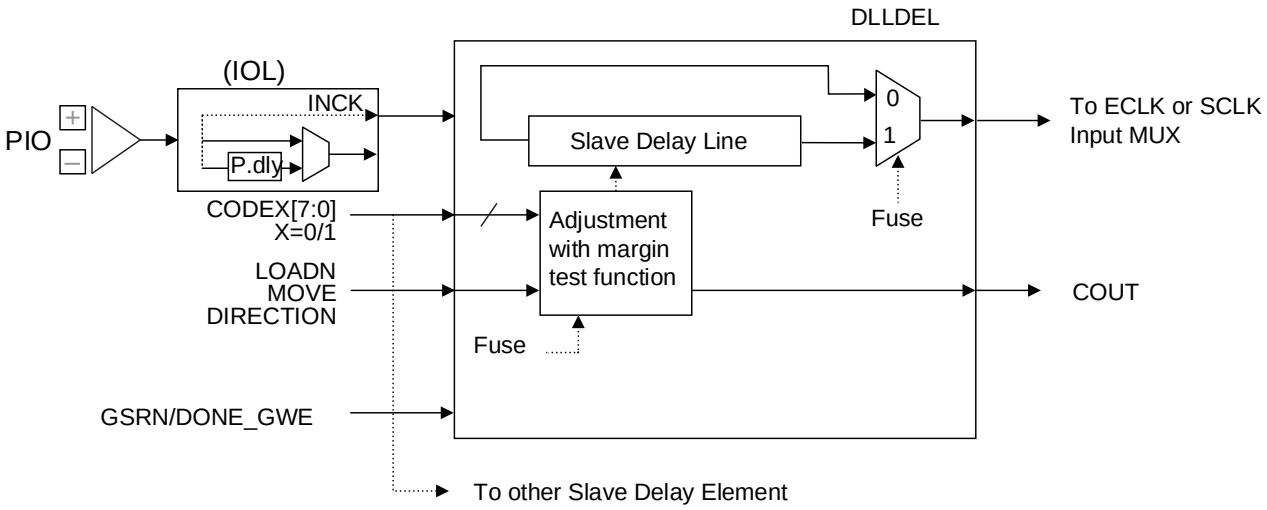


Figure 3.6. DDR Delay Block Diagram

3.10. DQSBUF

There is one DQSBUF for each DQS lane (every 16 I/O depending on the selected device). The DQS input is used when interfacing to DDR memories. It generates the delay on the DQS pin of the DQS lane, to provide a 90-degree phase shift on DQS to clock the DDR data at the center. The delay is set by a delay code generated in the DDRDLL component. Each DQSBUF can receive delay codes from two different DDRDLLs hence two different DDR memory interfaces can be built on the bottom side of the device.

Each of the DQSBUF modes has an additional feature that allows you to adjust the delay from the delay set by the DDRDLL code, similar to the one used in DLLDEL for margin test, by using the MOVE and DIRECTION inputs controlled by the user logic. The LOADN resets the delay back to the DDRDLL code.

3.11. Input DDR (IDDR)

The input DDR function can be used in the 1X (2:1), 2X (4:1), 4X (8:1), 7:1, and 10:1 gearing modes. In the 1X gearing mode, the IDDR module inputs a single DDR data input and SCLK (primary clock) and provides a 2-bit wide data synchronized with the SCLK (primary clock) to the FPGA fabric.

The 2X/4X gearing mode is used for interfaces with data rate higher than 400/800 Mbps, which require higher than 200/400 MHz system clock. Here, the IDDR element inputs a single DDR data input and DQS clock (for DDR memory interface) or ECLK (for all other high-speed interfaces) and provides a 4/8 bit wide parallel data synchronized with the SCLK (primary clock) to the FPGA fabric.

In the 7:1 gearing mode, mostly used in video applications requiring 7:1 interface, the IDDR element inputs a single DDR data input and ECLK and outputs a 7-bit wide parallel data synchronized with the SCLK (primary clock) to the FPGA fabric.

The 10:1 gearing mode is for SGMII/CDR application.

3.12. Output DDR (ODDR)

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

Similar to input interfaces, the 2X/4X gearing mode is used for data rate higher than 400/800 Mbps, which require higher than 200/400 MHz system clock. Here, the ODDR element receives 4/8 bit wide data from the FPGA fabric and generates a single DDR data output and clock output. The 2X/4X element uses high-speed edge clock (ECLK) to clock the data out for Generic high-speed interfaces and DQS clock for DDR memory 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 and clock output. The 7:1 element sends out data using high-speed edge clock.

The 10:1 gearing mode is for SGMII/CDR application.

3.13. 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 two clock dividers on each side of the device.

3.14. Input/Output DELAY

Similar to using the DLLDEL to tune the clock delay path, there are delay cells (DELAYB and DELAYA) in PIC can be utilized to tune the data path. 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). 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.

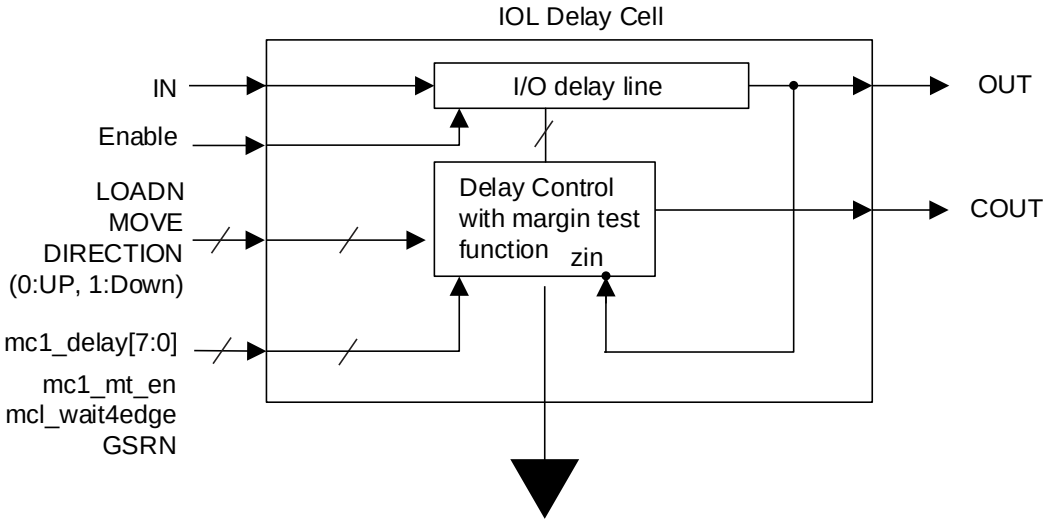


Figure 3.7. PIC Delay Cell Diagram

4. Building Generic High Speed Interfaces

This section describes in detail on how 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 different types of high-speed I/O interfaces available in the CrossLink-NX device.

Table 4.1 lists these interfaces. The naming conventions use 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 CrossLink-NX device clocking structure, refer to CrossLink-NX sysClock PLL/DLL Design and Usage Guide (FPGA-TN-02095). The various interface rules listed under each interface should be followed to build these interfaces successfully. Refer to the Timing Analysis for High Speed DDR Interfaces section of this document for more information about the timing analysis on these interfaces.

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.

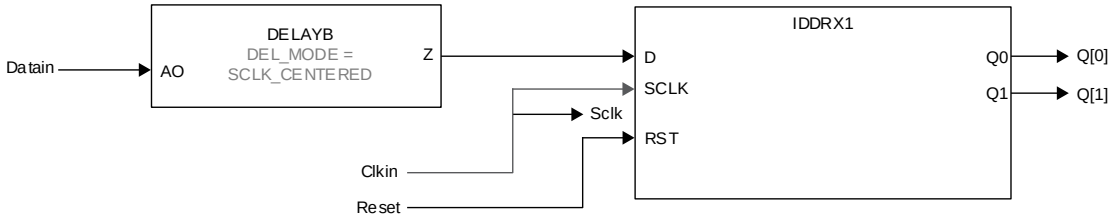


Figure 5.1. GDDRX1_RX.SCLK.Centered Interface (Static Delay)

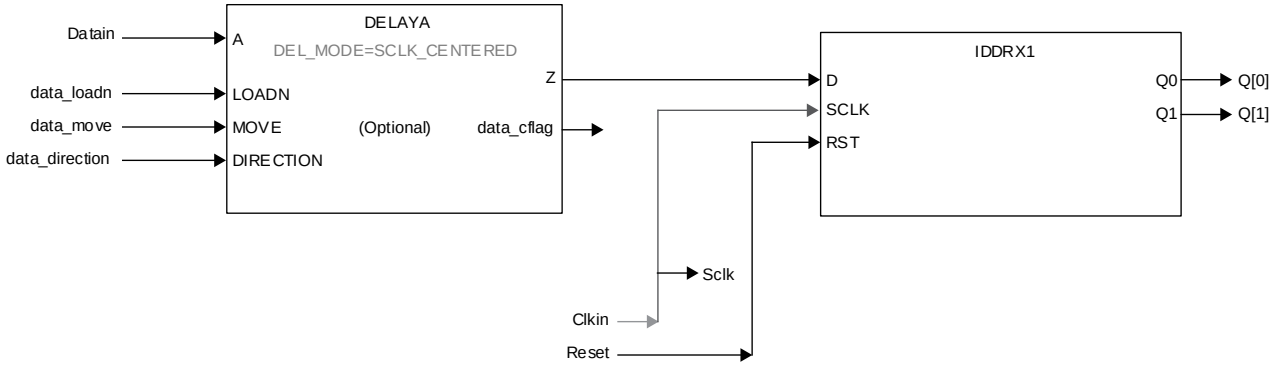


Figure 5.2. GDDRX1_RX.SCLK.Centered Interface (Dynamic Data delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the primary clock tree.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.

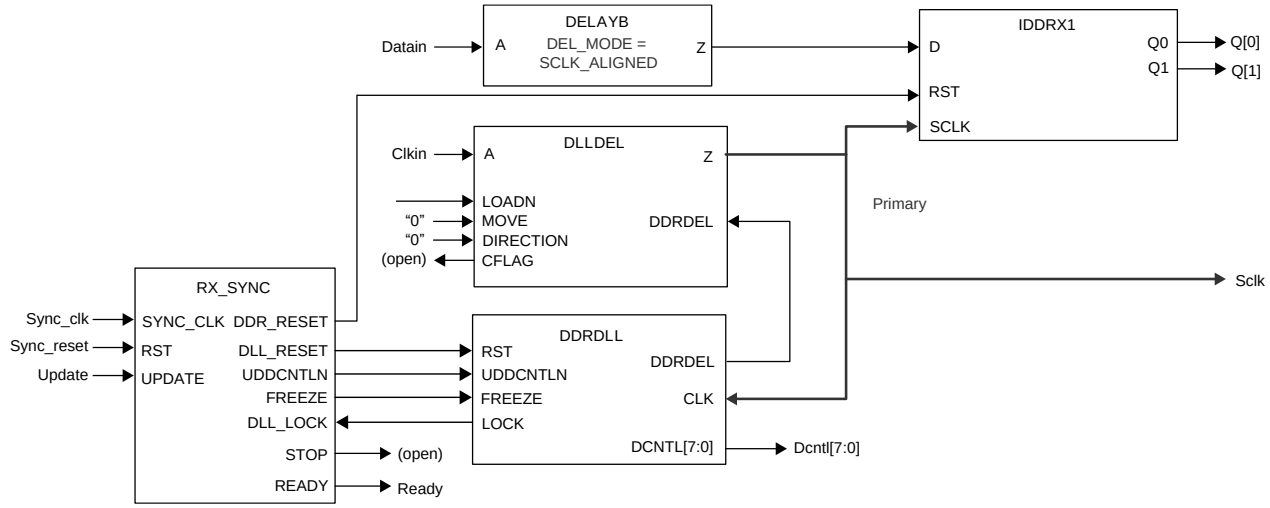


Figure 5.3. GDDRX1_RX.SCLK.Aligned Interface (Static Delay)

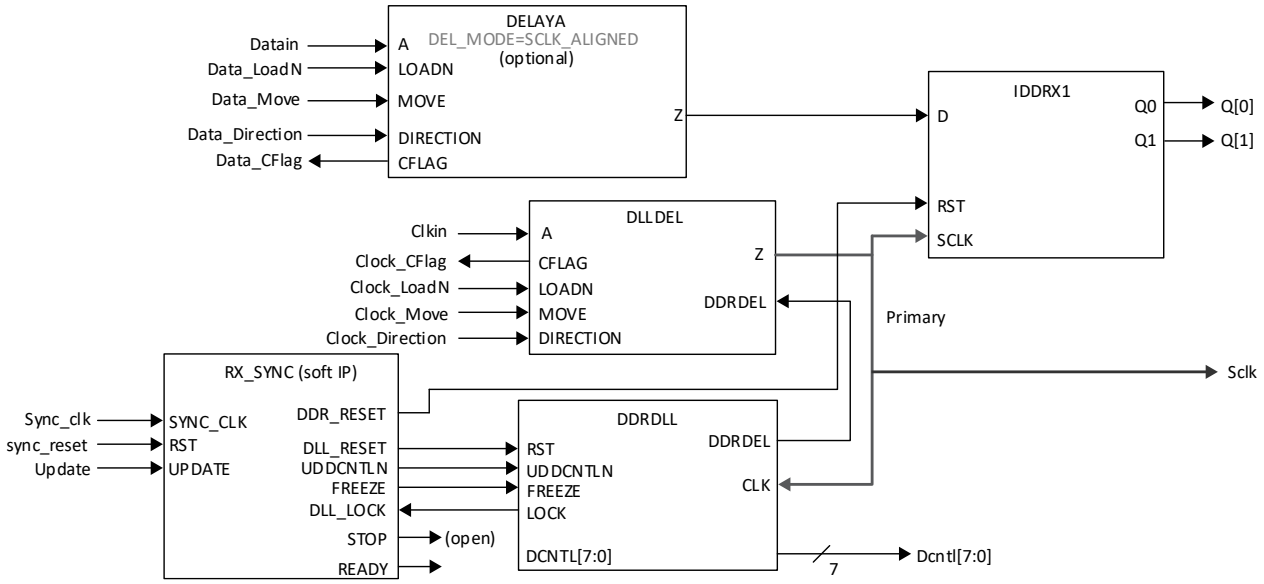


Figure 5.4. GDDRX1_RX.SCLK.Aligned Interface (Dynamic Data/Clock Delay)

Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the DLLDELD input.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the 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.

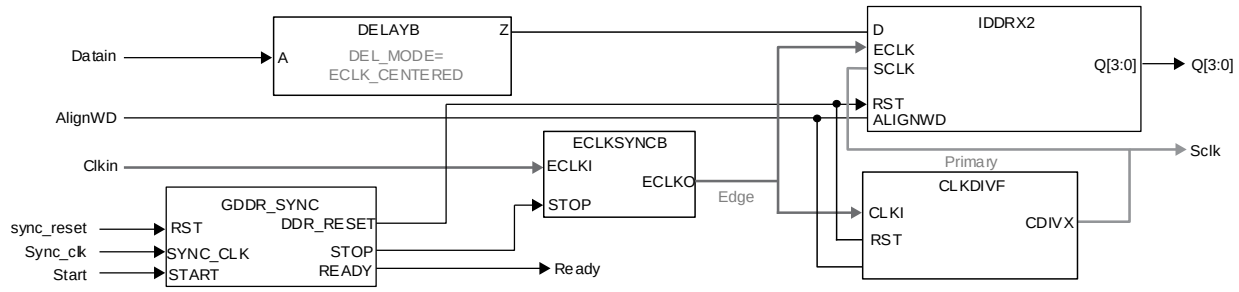


Figure 5.5. GDDRX2_RX.ECLK.Centered Interface (Static Delay)

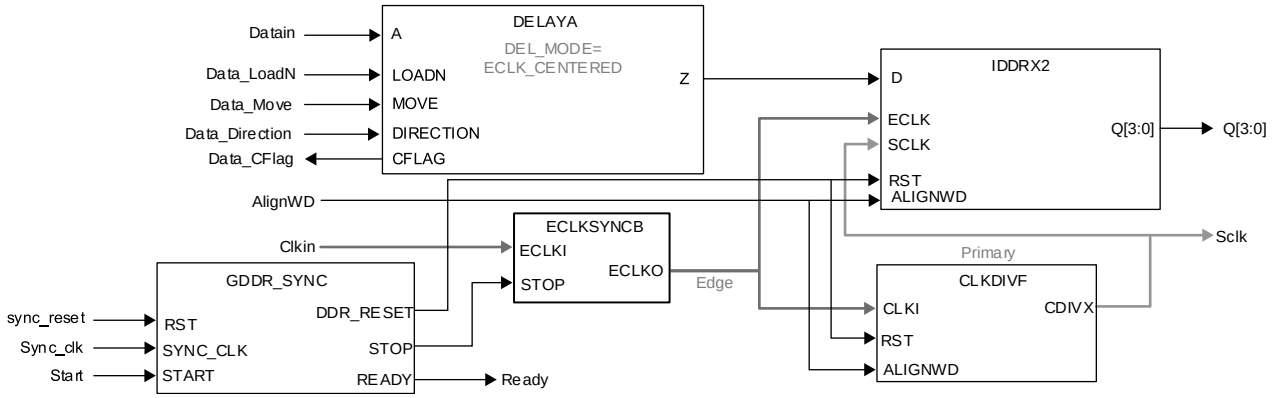


Figure 5.6. GDDRX2_RX.ECLK.Centered Interface (Dynamic Data Delay)

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.

- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- 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.

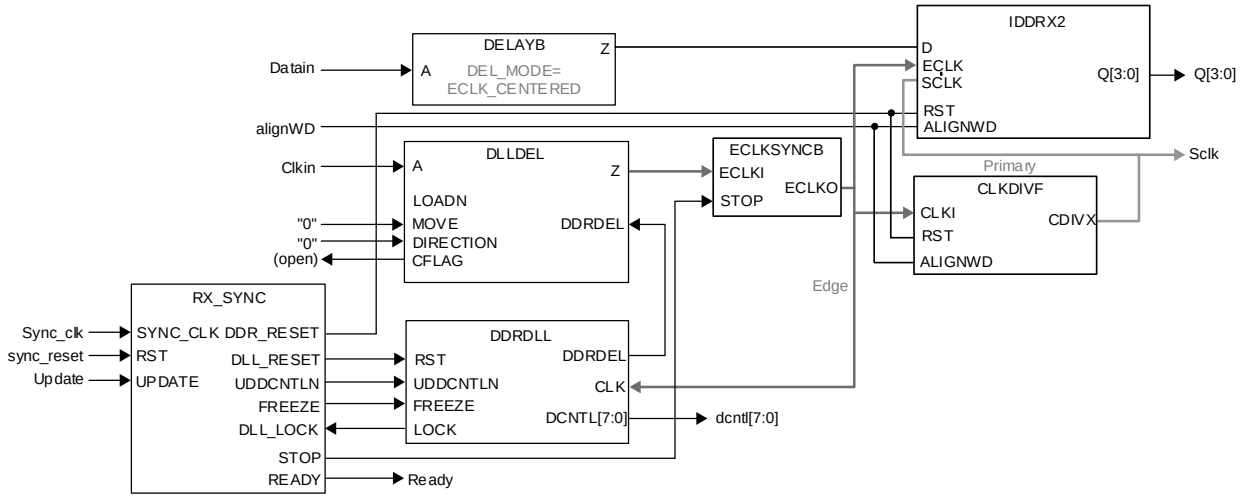


Figure 5.7. GDDRX2_RX.ECLK.Aligned Interface (Static Delay)

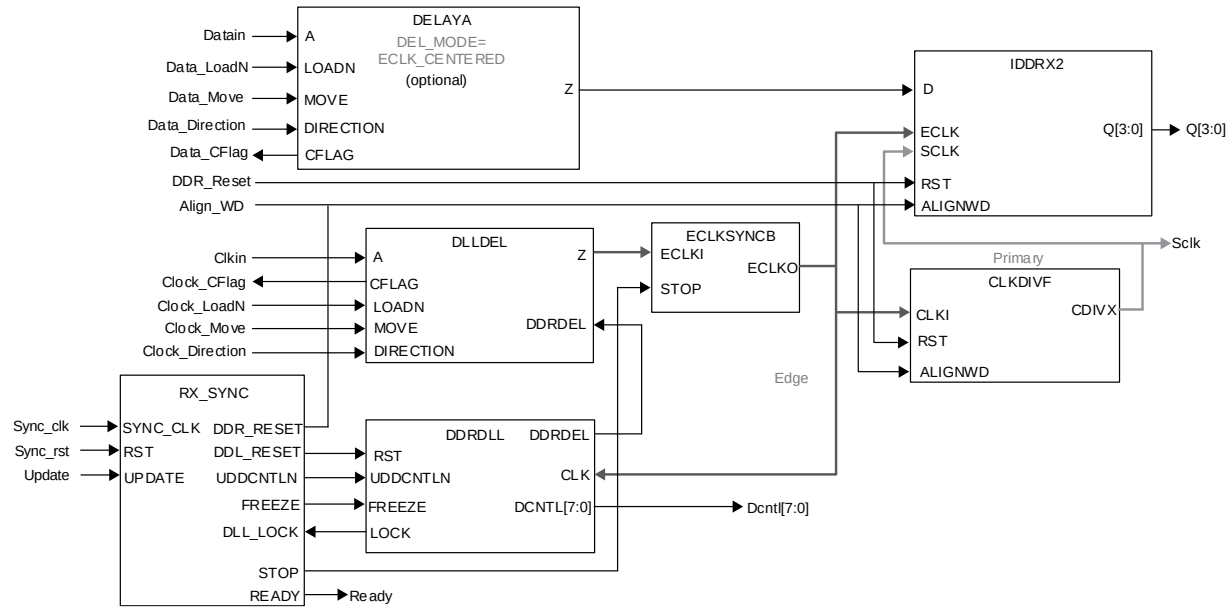


Figure 5.8. GDDRX2_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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 ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the 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.

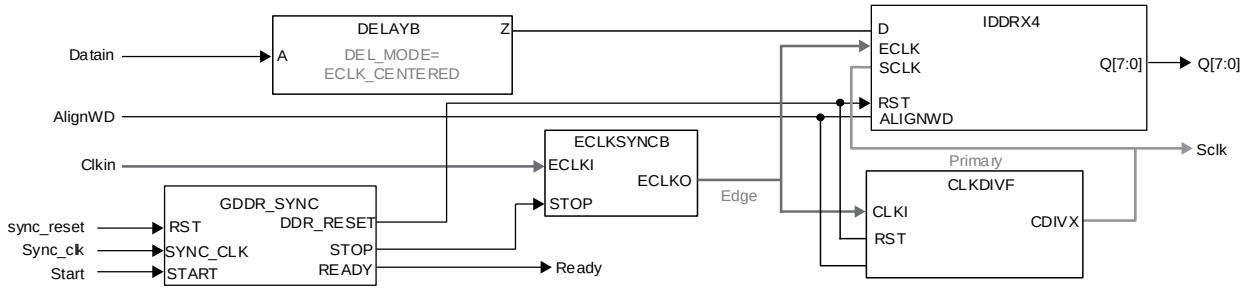


Figure 5.9. GDDRX4_RX.ECLK.Centered Interface (Static Delay)

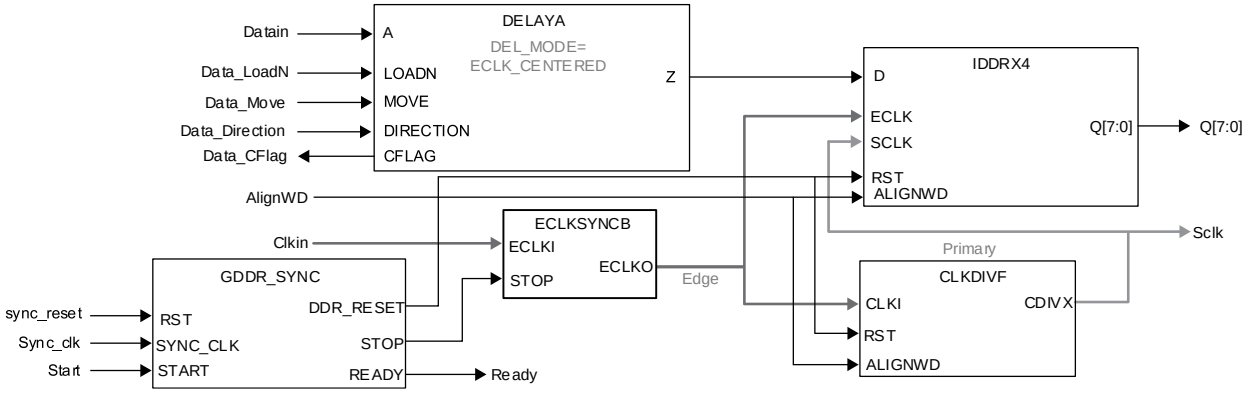


Figure 5.10. GDDRX4_RX.ECLK.Centered Interface (Dynamic Data Delay)

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- 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.

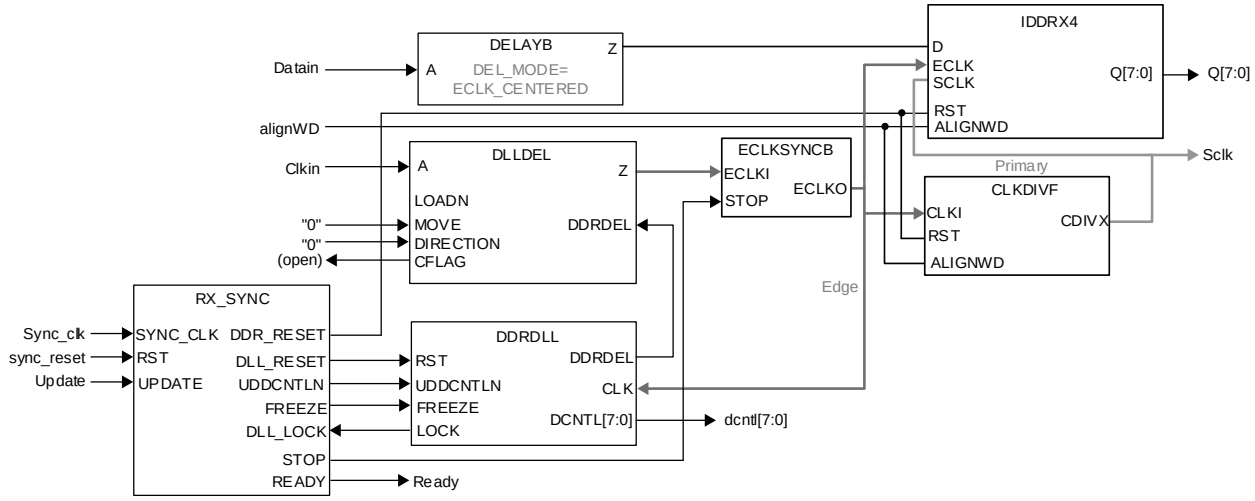


Figure 5.11. GDDRX4_RX.ECLK.Aligned Interface (Static Delay)

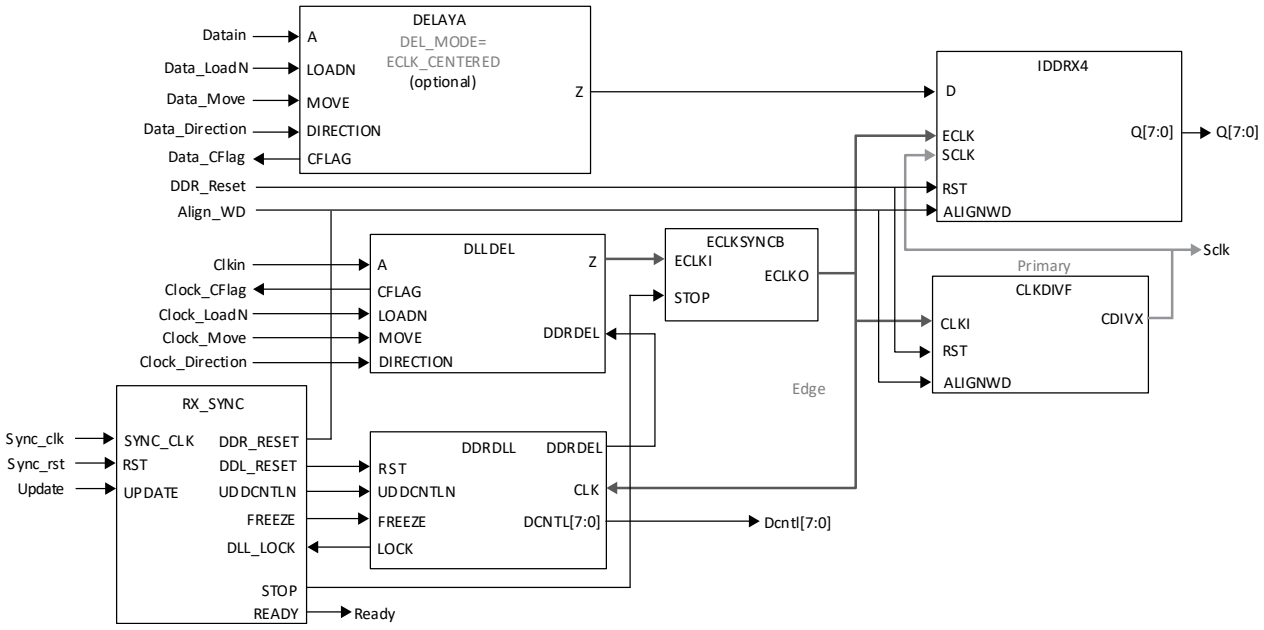


Figure 5.12. GDDRX4_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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 ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the 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.

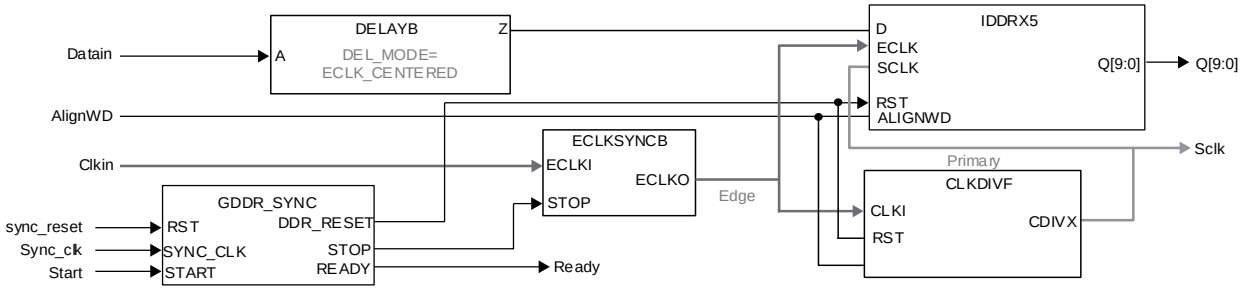


Figure 5.13. GDDRX5_RX.ECLK.Centered Interface (Static Delay)

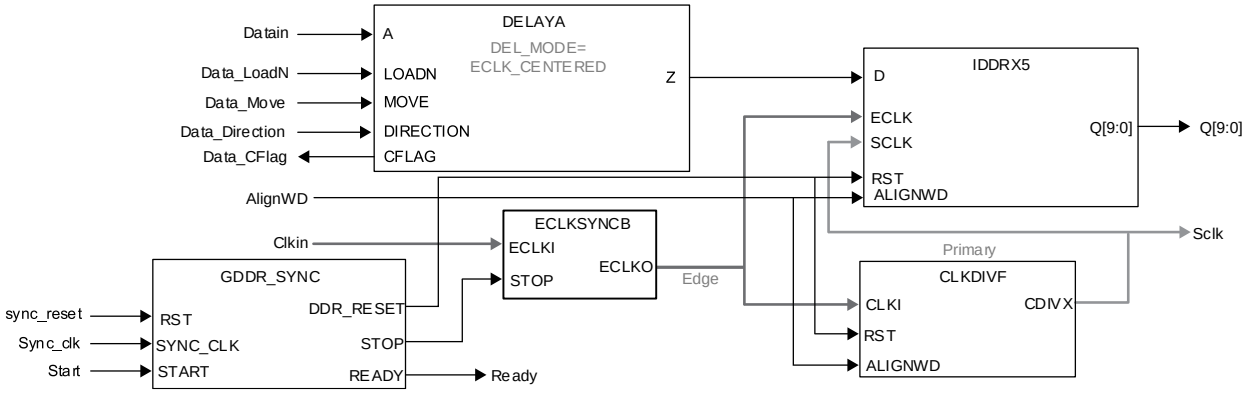


Figure 5.14. GDDRX5_RX.ECLK.Centered Interface (Dynamic Data Delay)

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- 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.

Figure 5.15 and Figure 5.16 show the static delay and dynamic delay options for this interface.

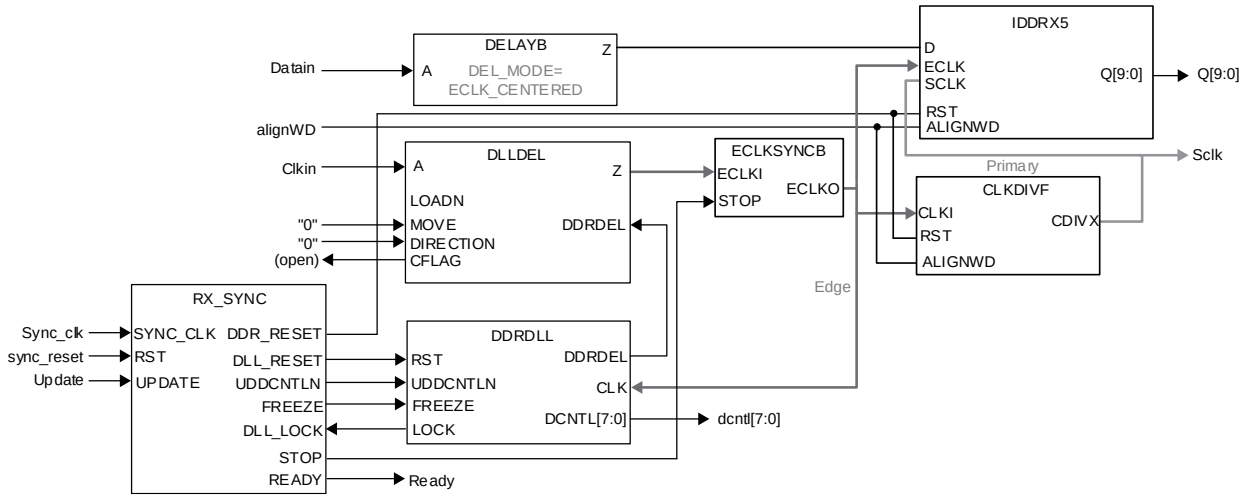


Figure 5.15. GDDRX5_RX.ECLK.Aligned Interface (Static Delay)

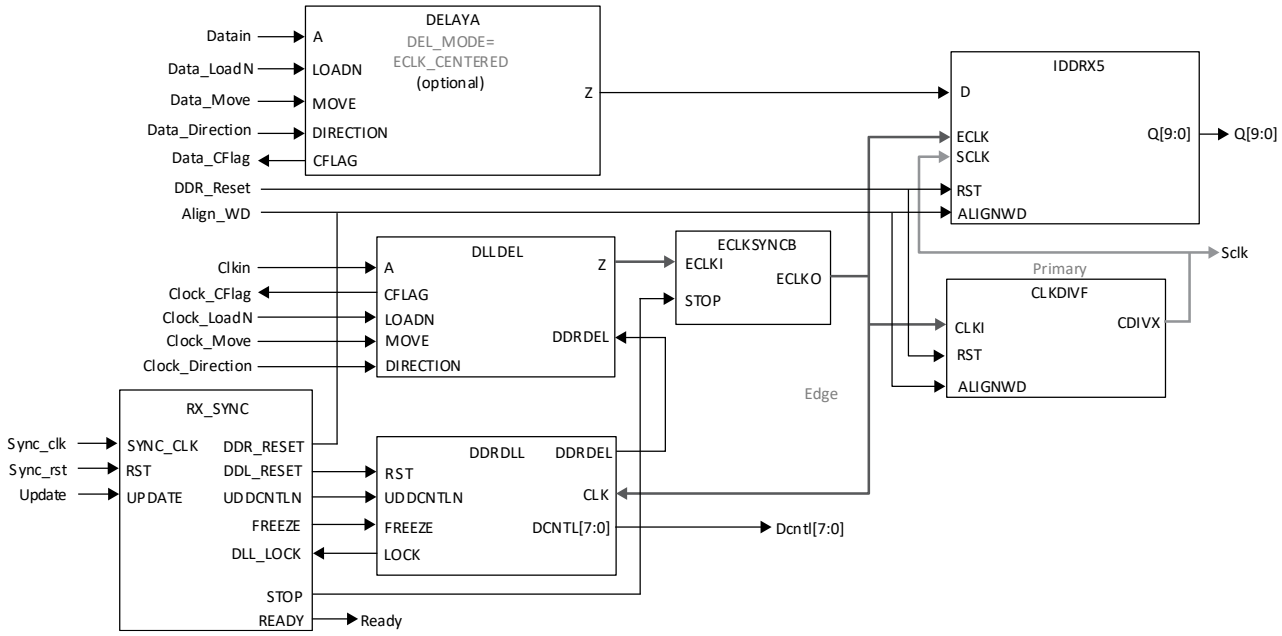


Figure 5.16. GDDRX5_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)

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.

You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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's 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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.

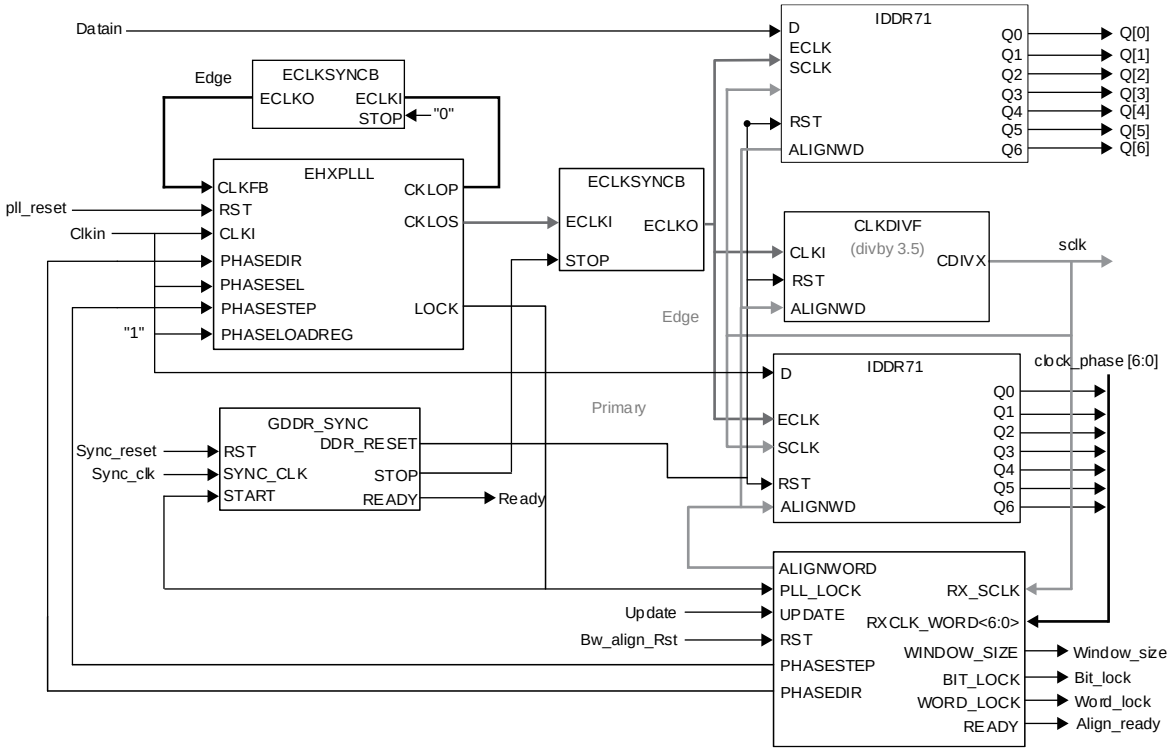


Figure 5.17. GDDRX71_RX.ECLK Interface

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

The CrossLink-NX device family provides two hardened MIPI D-PHY blocks supporting both RX and TX. Refer to the CrossLink-NX Hardened D-PHY Usage Guide (FPGA-TN-02081) for details.

Soft MIPI D-PHY receive interface is supported as well in CrossLink-NX. 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 Bank3, Bank4, and Bank5 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

Port Name	I/O	Width	Description
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.

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.

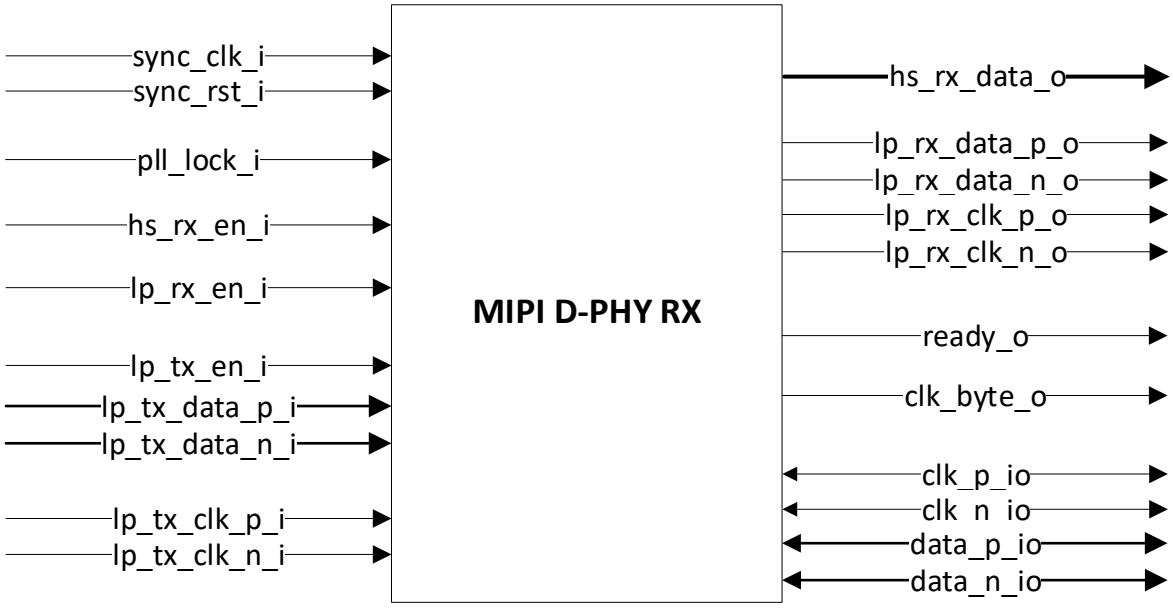
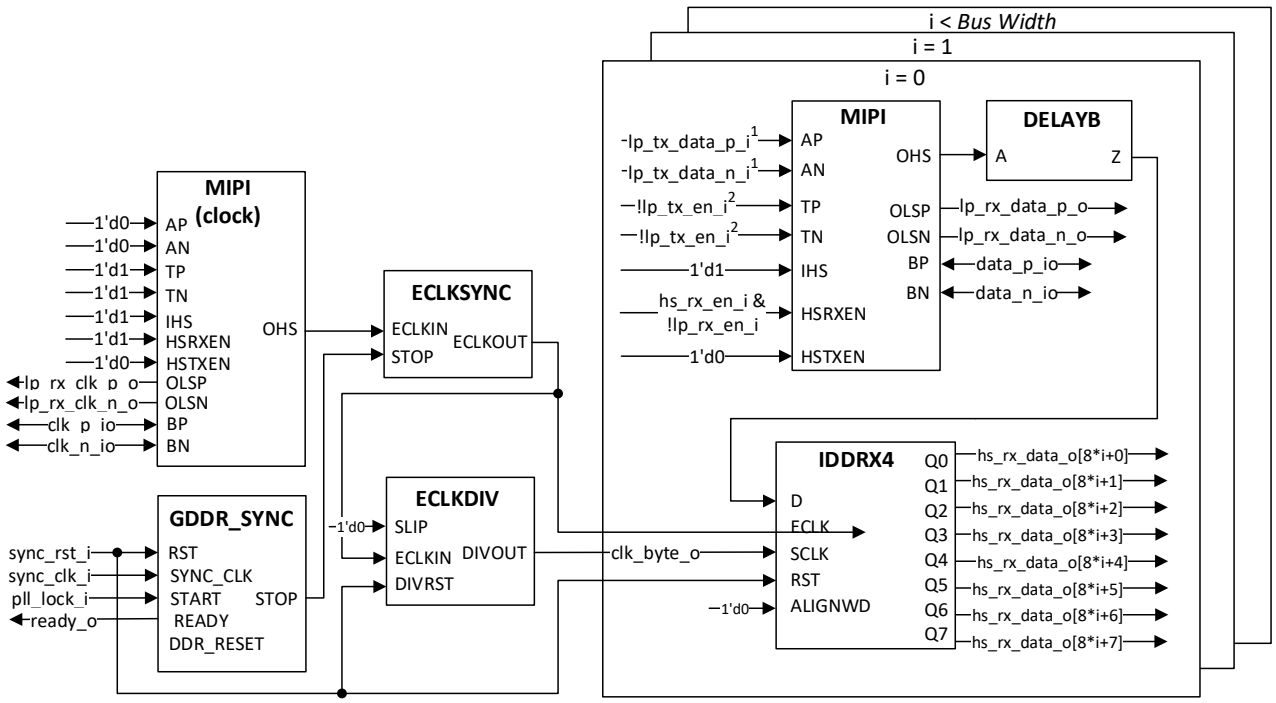


Figure 5.18. MIPI D-PHY RX Block Diagram



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 tristated using either a tristate input going through an I/O register.

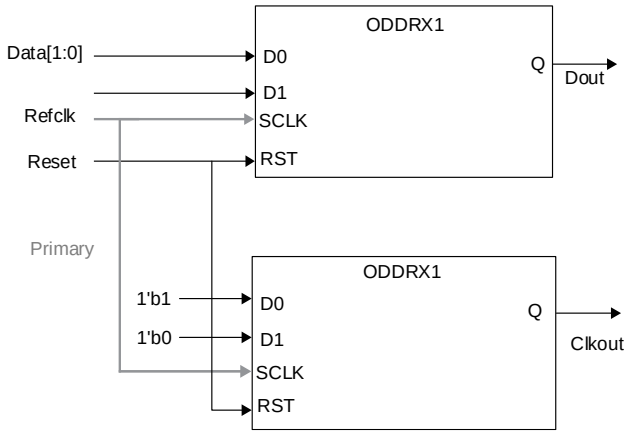


Figure 5.20. GDDRX1_TX.SCLK.Aligned Interface

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 tristated using either a tristate input going through an I/O register.

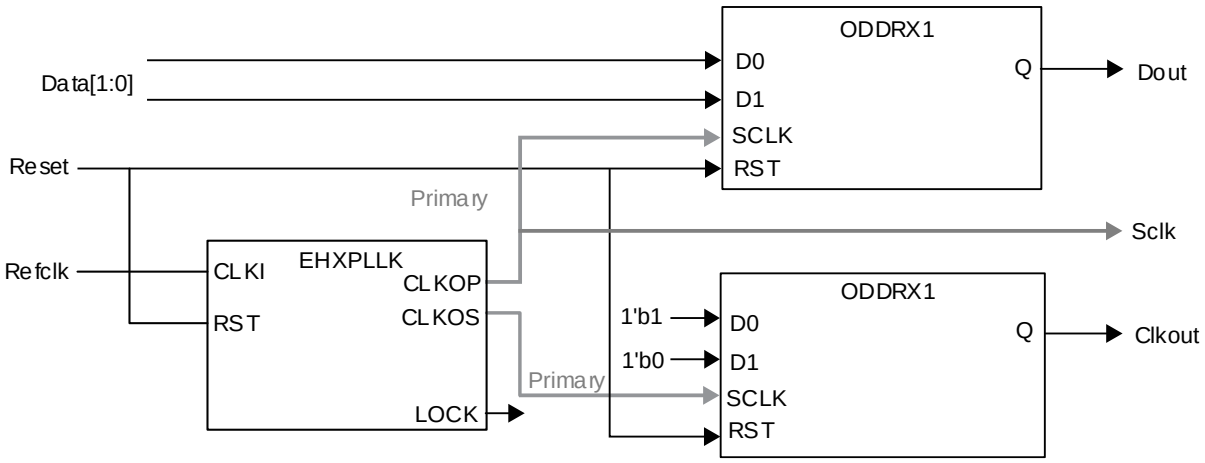


Figure 5.21. GDDRX1_TX.SCLK.Centered Interface

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tristated using either a tristate input going through an I/O register.

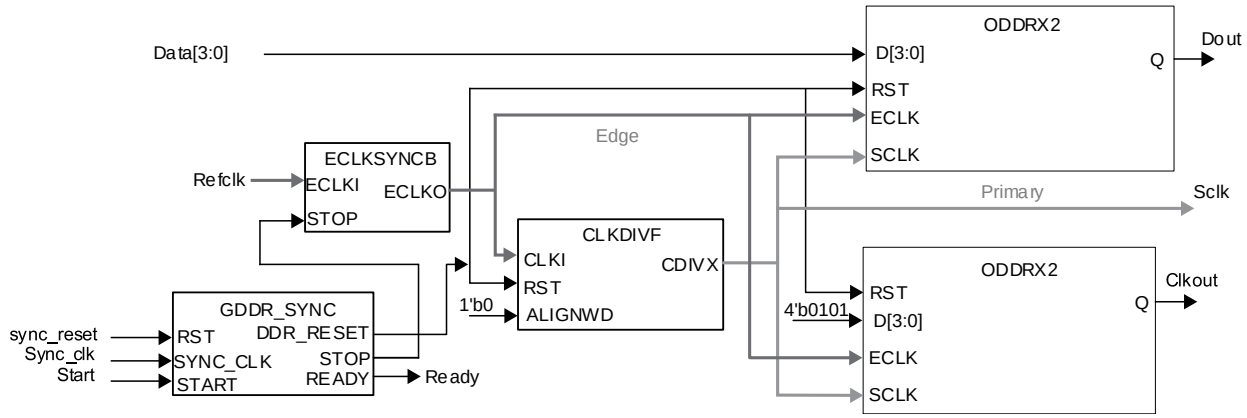


Figure 5.22. GDDRX2_TX.ECLK.Aligned Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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 tristated using either a tristate input going through an I/O register.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.

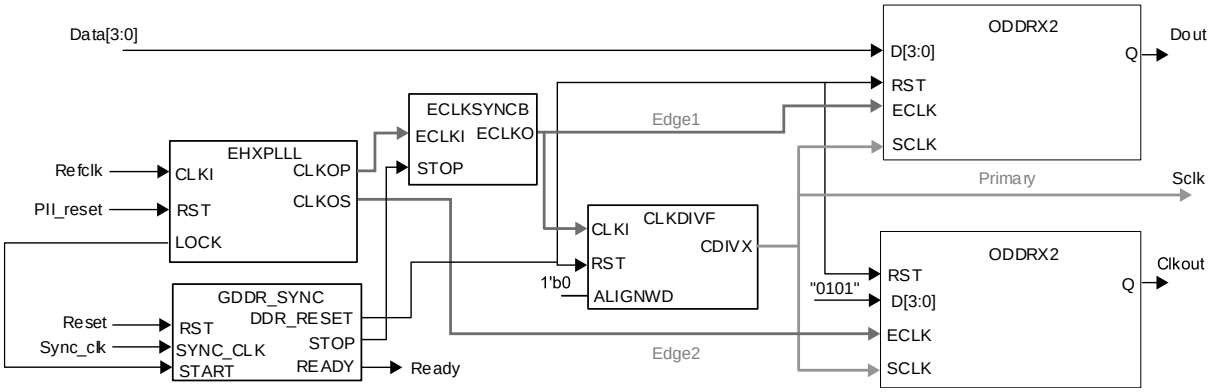


Figure 5.23. GDDRX2_TX.ECLK.Centered Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tristated using either a tristate input going through an I/O register.

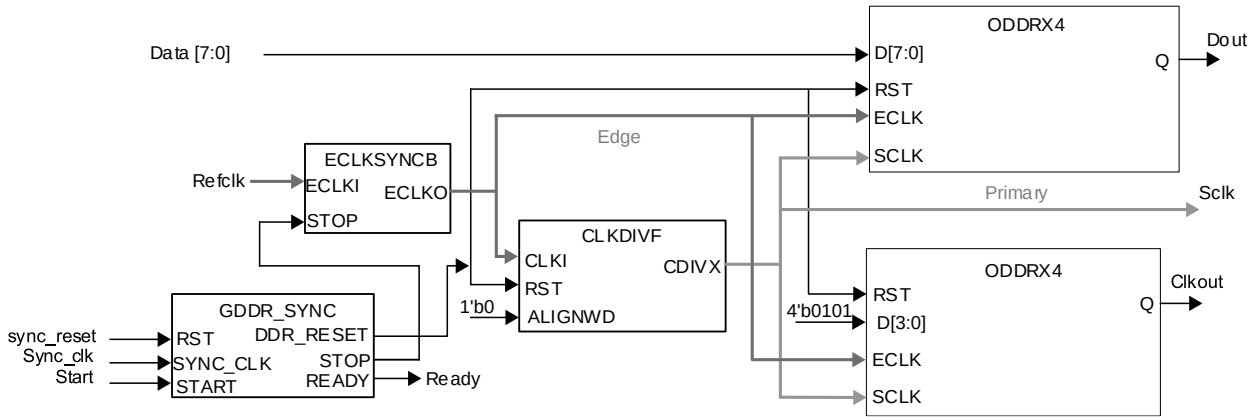


Figure 5.24. GDDRX4_TX.ECLK.Aligned Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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 tristated using either a tristate input going through an I/O register.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.

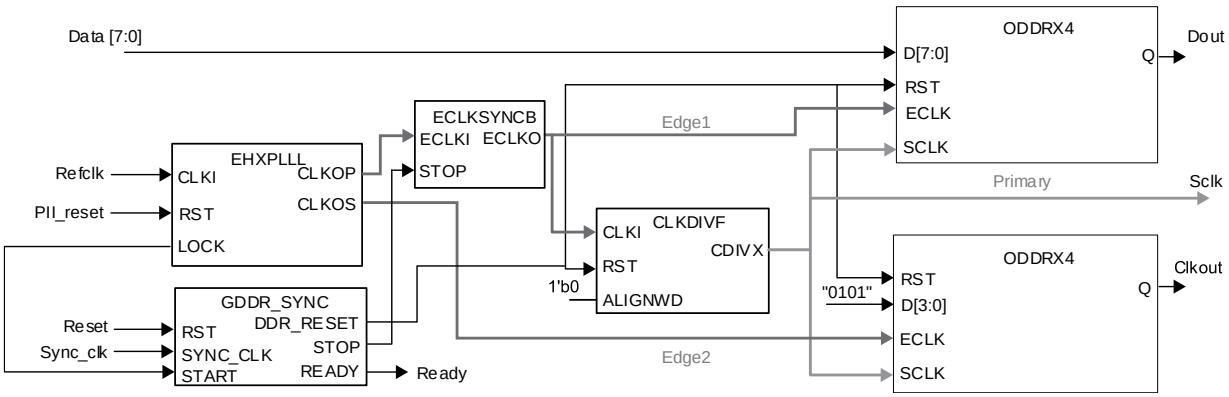


Figure 5.25. GDDRX4_TX.ECLK.Aligned Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.
- Optionally, you can choose to use the DELAYB or DELAYA element to delay the data output.
- The output data can be optionally tristated using either a tristate input going through an I/O register.

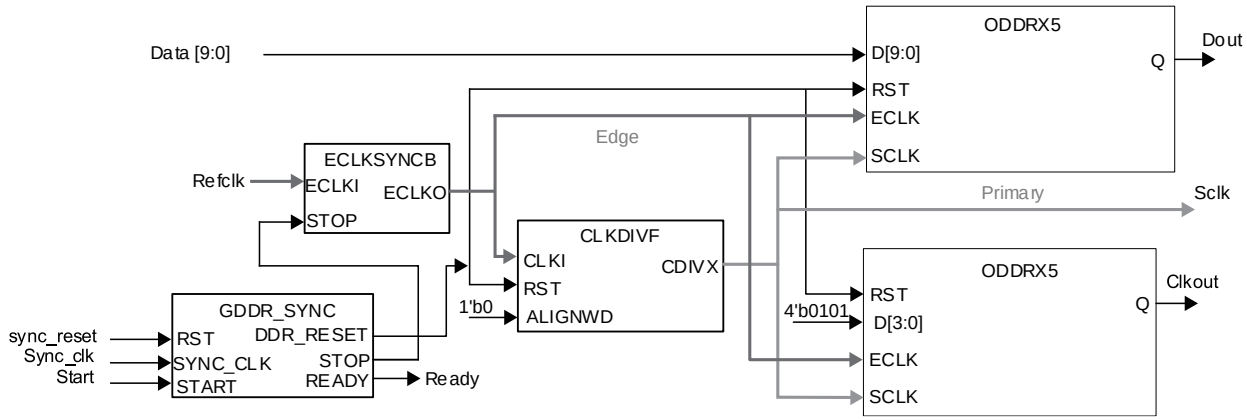


Figure 5.26. GDDRX5_TX.ECLK.Aligned Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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 tristated using either a tristate input going through an I/O register.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.

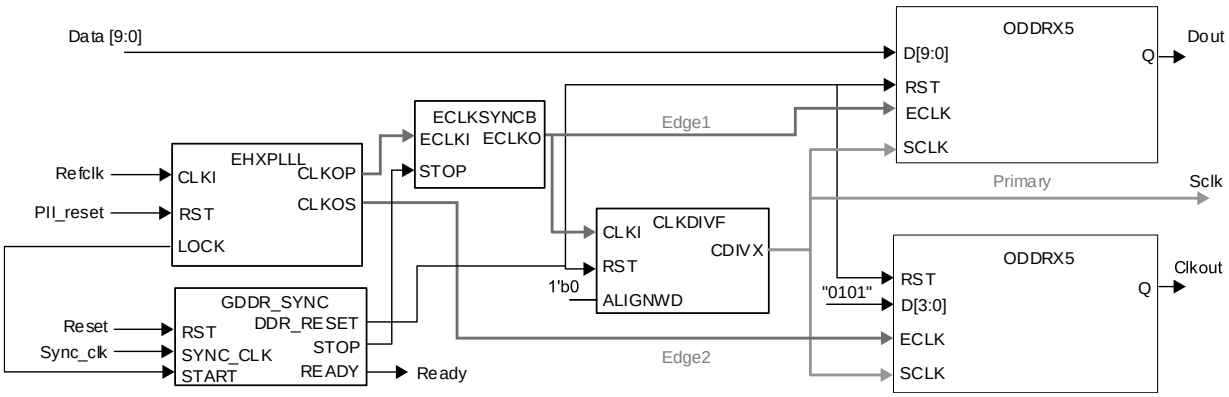


Figure 5.27. GDDRX5_TX.ECLK.Aligned Interface

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.

You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

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.
- The ECLKBRIDGE can be optionally enabled if the data bus is crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled, then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through IP Catalog.

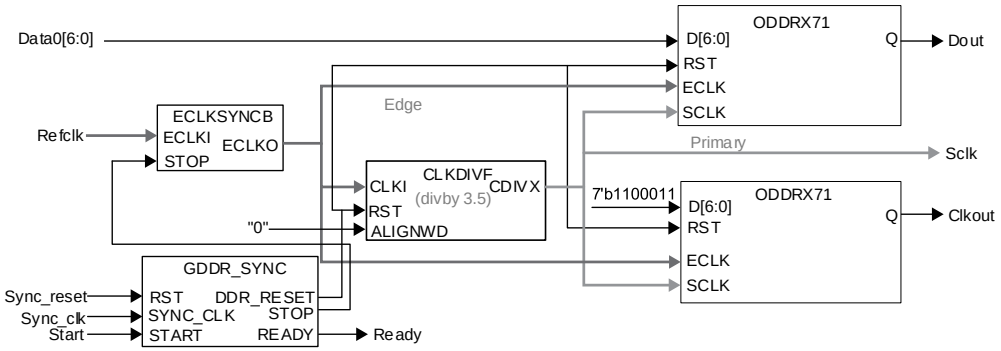


Figure 5.28. GDDRX71_TX.ECLK Interface

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.
- You must set the timing preferences as indicated in the Timing Analysis for High Speed DDR Interfaces section.

5.20. Soft MIPI D-PHY Transmit Interfaces

The CrossLink-NX device family provides two hardened MIPI D-PHY blocks supporting both RX and TX. Refer to the CrossLink-NX Hardened D-PHY Usage Guide (FPGA-TN-02081) for details.

Soft MIPI D-PHY transmit interface is supported as well in CrossLink-NX. 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 Bank3, Bank4, and Bank5 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.

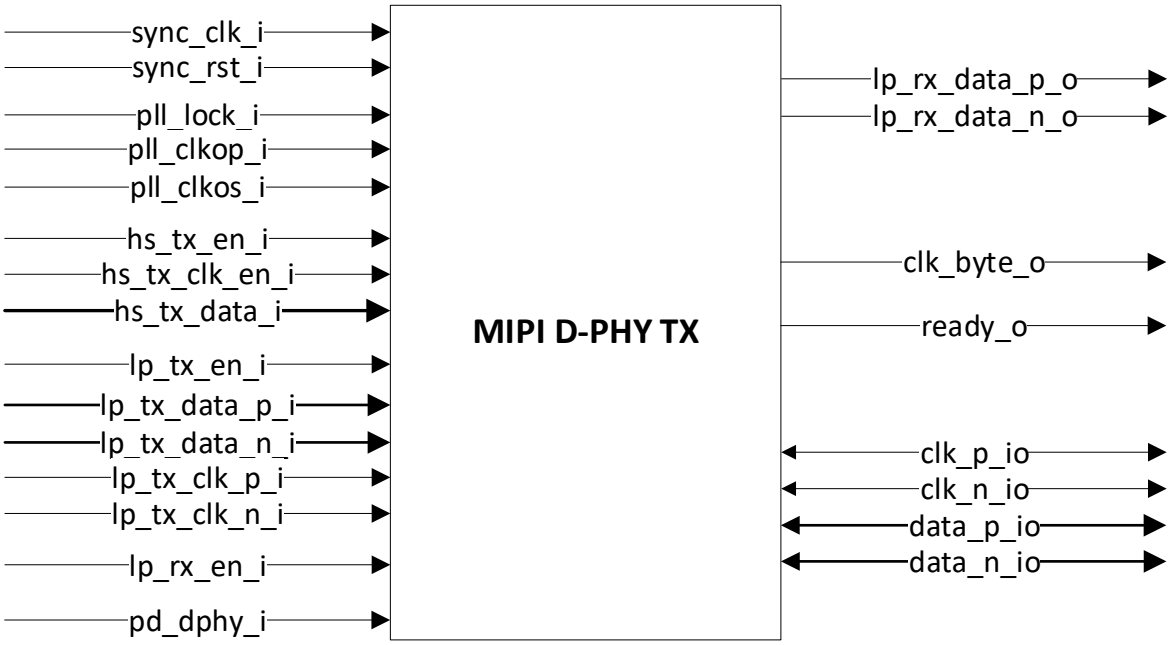
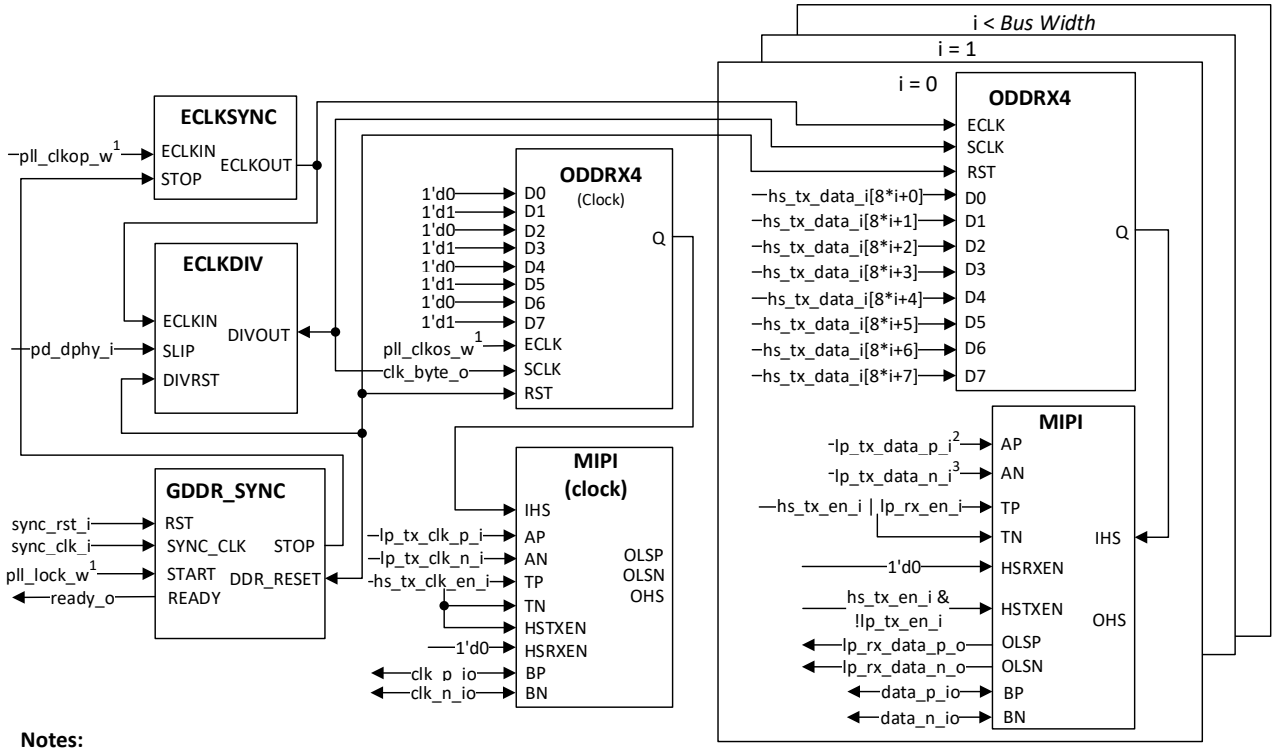


Figure 5.29. MIPI D-PHY TX Block Diagram



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 CrossLink-NX 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 two CLKDIV modules, which means that you can implement two different GDDRX2 RX interface per Bank since each 2X, 4X, or 5X 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 DQSBUF/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 1X gearing. These can be used for single ended interfaces only.
- Interfaces using the x1 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, CrossLink-NX 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 2X, 4X, or 5X 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 x1 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 2X, 4X, or 5X 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 x2 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 CrossLink-NX sysClock PLL/DLL Design and Usage Guide (FPGA-TN-02095) for details.

- Primary Clock to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, and CLKDIV outputs. See CrossLink-NX sysClock PLL/DLL Design and Usage Guide (FPGA-TN-02095) for details.
- None of the clocks going to the DDR registers can come from internal general routing.
- DQS clocking is used for DDR memory interface implementation. DQS clock spans every 16 I/O including the DQS pins. Refer to the section for pinout assignment rules when using DQS clocking.

5.22. Timing Analysis for High Speed DDR Interfaces

It is recommended that you run Static Timing Analysis in the software for each of the high-speed interfaces. This section describes the timing preferences to used for each type of interface and the expected trace results. The preferences can either be entered directly in the .lpf file or through the Design Planner graphical user interface.

The External Switching Characteristics section of CrossLink-NX Family Data Sheet (FPGA-DS-02049) should be used along with this section. The data sheet specifies the actual values for these constraints for each of the interfaces.

5.22.1. Frequency Constraints

It is required that you explicitly specify FREQUENCY (or PERIOD) PORT preferences to all input clocks in the design. This preference may not be required if the clock is generated out of a PLL or DLL or is input to a PLL or DLL.

5.22.2. DDR Input Setup and Hold Time Constraints

All of the Receive (RX) interfaces, both x1 and x2 can be constrained with setup and hold preference.

5.22.2.1. Receive Centered Interface

Figure 5.31 below shows the Data and Clock relationship for a Receive Centered Interface. The clock is centered to the data, so it goes into the devices with a setup and hold time.

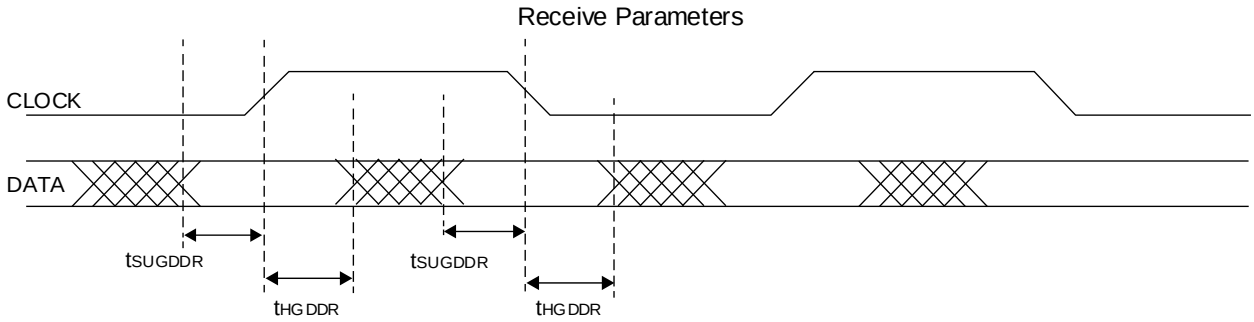


Figure 5.31. RX Centered Interface Timing

Note: tSUGDDR = Setup Time, tHOGDDR = Hold Time

In this case, you must specify in the software preference the amount of setup and hold time available. These parameters are listed in Figure 5.31 as tSU_GDDRX1/2 and tHO_GDDRX1/2. These can be directly provided using the INPUT_SETUP and HOLD preference as –

INPUT_SETUP PORT “DATA” <tSU_GDDRX1/2> ns HOLD <tHO_GDDRX1/2> ns CLKPORT “CLOCK”;

where:

Data = Input Data Port

Clock = Input Clock Port

The external Switching Characteristics section of CrossLink-NX Family Data Sheet (FPGA-DS-02049) specifies the MIN setup and hold time required for each of the high-speed interfaces running at MAX speed. These values can be picked up from the data sheet if the interface is running at MAX speed.

Example:

For GDDRX2_RX.ECLK.Centered Interface running at max speed of 400 MHz, the preference would be –

INPUT_SETUP PORT "datain" 0.320000 ns HOLD 0.320000 ns CLKPORT "clk”;

Note: Refer to CrossLink-NX Family Data Sheet (FPGA-DS-02049) for the latest tSUDDR and tHOGDDR numbers.

5.22.2.2. Receive Aligned Interface

Figure 5.32 below shows the Data and Clock relationship for a Receive Aligned Interface. The clock is aligned edge to edge the data.

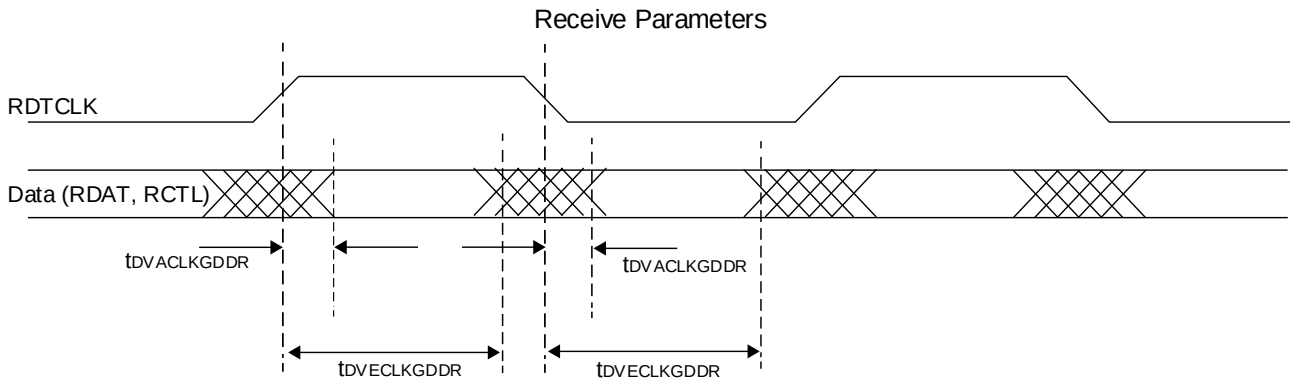


Figure 5.32. RX Aligned Interface Timing

Note: tDVA_GDDRX1/2 = Data Valid after CLK, tDVE_GDDRX1/2 = Data Hold After CLK

In this case, the worst case data may occur after the clock edge hence has a negative setup time when entering the device. In this case, the worst case setup is specified by the tDVACLKGDDR after the clock edge and the worst case hold time is specified as tDVECLKGDDR. For this case the setup and hold time can be specified as –

INPUT_SETUP PORT “din” <-tDVA_GDDRX1/2 > ns HOLD < tDVE_GDDRX1/2> ns CLKPORT “clk”;

Note: Negative number is used for SETUP time as the data occurs after the clock edge in this case.

The External Switching Characteristics section of CrossLink-NX Family Data Sheet (FPGA-DS-02049) specifies the MIN tDVA_GDDRX1/2 and tDVE_GDDRX1/2 values required for each of the high-speed interfaces running at MAX speed. These values can be picked up from the data sheet if the interface is running at MAX speed. The data sheet numbers for this preference is listed in ns + ½ UI (Unit Interface). 1 UI is equal to ½ the Clock Period. Hence, these numbers need to be calculated from the CLK Period used.

Preference Example:

For GDDRX2_RX.ECLK.Aligned interface running at max speed of 400 MHz (UI = 1.25 ns)

tDVA_GDDRX2 = - 0.344 ns + ½ UI = 0.281 ns, tDVE_GDDRX2 = 0.344 ns + ½ UI =0.969 ns

The preference for this case would be –

INPUT_SETUP PORT "datain" -0.2810000 ns HOLD 0.969 ns CLKPORT "clk";

Note: Please check CrossLink-NX Family Data Sheet (FPGA-DS-02049) for the latest tDVA_GDDRX1/X2 and tDVE_GDDRX1/X2 numbers.

5.22.2.3. Receive Dynamic Interfaces

Static Timing Analysis does not show timing for all the Dynamic interfaces cases as either the Clock or the Data delay is dynamically updated at run time.

5.22.3. DDR Clock to Out Constraints for Transmit Interfaces

All of the Transmit (TX) interfaces both x1 and x2 can be constrained with Clock to out constraint to detect the relationship between the Clock and Data when leaving the device.

Figure 5.33 shows how the clock to out is constrained in the software. Min tCO is the minimum time after the clock edge transition that the data does not transition. Max tCO is the maximum time after clock transition before which the data transitions. Therefore, any data transition must occur between the tCO Min and tCO Max values.

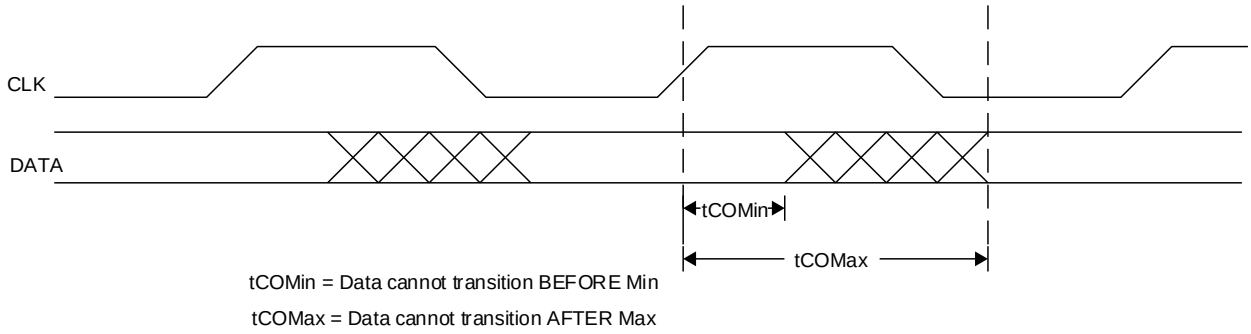


Figure 5.33. tCO Min and Max Timing Analysis

5.22.3.1. Transmit Centered Interfaces

In this case, the transmit clock is expected to be centered to the data when leaving the device. Figure 5.34 shows the timing for a centered transmit interface.

tDVBGDDR = Data valid before clock

tDVAGDDR = Data valid after clock

tU = Data transition

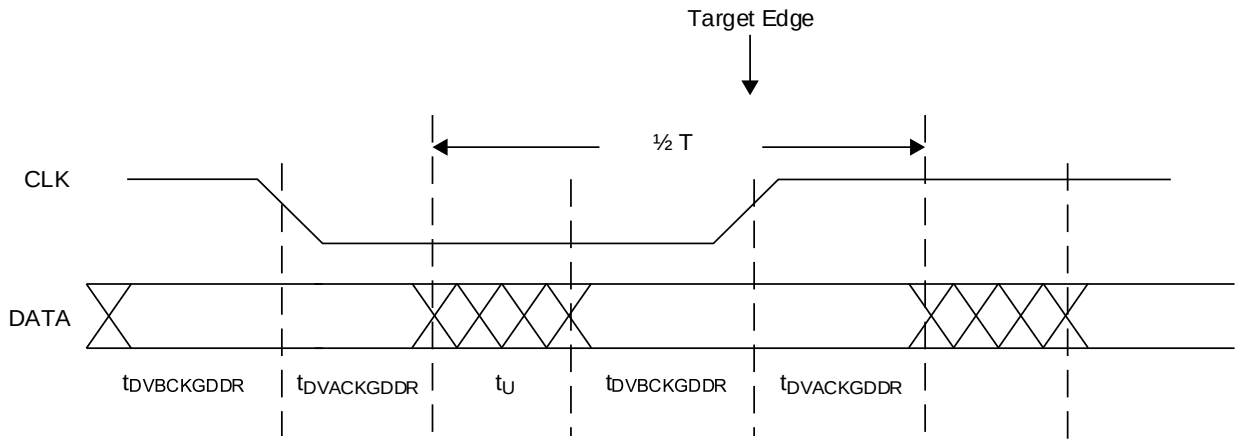


Figure 5.34. Transmit Centered Interface Timing

Figure 5.34 shows that max value after which the data cannot transition is – tVB_GDDR. The min value before which the data cannot transition is – (tU+tVB_GDDR). Negative sign is used because in this particular case where clock is forwarded centered aligned to the data these two conditions occurs before the clock edge.

The CrossLink-NX Family Data Sheet (FPGA-DS-02049) specifies the tDVB_GDDRX1/X2 and tDVA_GDDRX1/X2 values at maximum speed. However, we do not have the tU value. Hence, min tCO can be calculated using the following equation.

tCO Min = - (tVB_GDDRX1/X2 + tU)

½ T = tDVA_GDDRX1/X2 + tVB_GDDRX1/X2 + tU

- (tVB_GDDRX1/X2 + tU) = 1/2T - tDVA_GDDRX1/X2

tCO Min = 1/2T - tDVA_GDDRX1/X2

The clock to out time in the software can be specified as –

CLOCK_TO_OUT PORT “dataout” MAX <-tDVB_GDDRX1/X2> MIN <tDVA_GDDRX1/X2 -1/2 Clock Period> CLKPORT “clk” CLKOUT PORT “clkout”;

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The values for tDVBCKGDDR and tDVACKGDDR can be picked up from the External Switching Characteristics section of CrossLink-NX Family Data Sheet (FPGA-DS-02049) for the MAX speed.

Preference Example:

For GDDRX1_TX.SCLK.Centered interface running at 250 MHz, tDVB_GDDRX1 = tDVA_GDDRX1 = 0.67 ns, the preference would be –

CLOCK_TO_OUT PORT "dataout" MAX -0.670000 ns MIN -1.330000 ns CLKPORT "clk" CLKOUT PORT "clkout”;

Note: Refer to CrossLink-NX Family Data Sheet (FPGA-DS-02049) for the latest tDVAGDDR and Tdvbgddr numbers.

5.22.3.2. Transmit Aligned Interfaces

In this case, the clock and data are aligned when leaving the device. Figure 5.35 below shows the timing diagram for this interface.

tDIAGDDR = Data valid after clock.

tDIBGDDR = Data valid before clock.

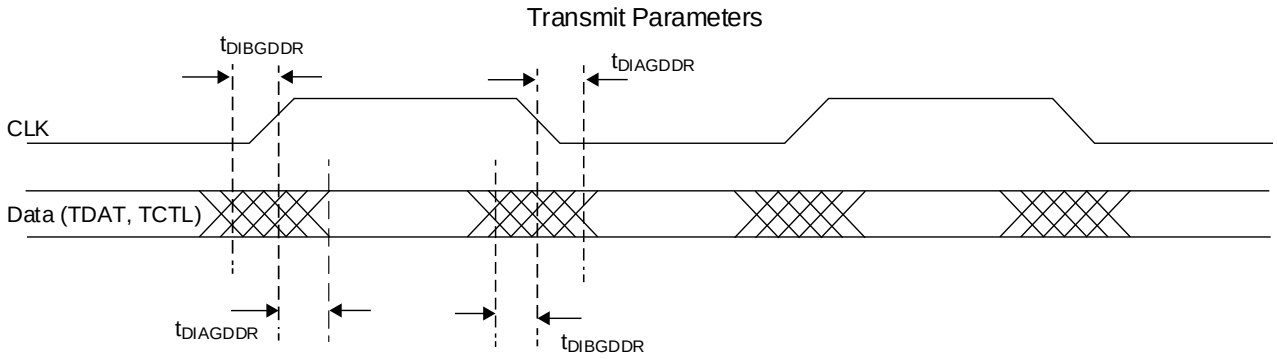


Figure 5.35. Transmit Aligned Interface Timing

Figure 5.35 shows that max value after which the data cannot transition is tDIA_GDDRX1/X2. The min value before which the data cannot transition is – tDIB_GDDRX1/X2. Negative sign is used for the minimum value is because in this particular case the minimum condition occurs before the clock edge.

The clock to out time in the software can be specified as –

CLOCK_TO_OUT PORT “dataout” MAX <tDIA_GDDRX1/X2> MIN <-tDIB_GDDRX1/X2> CLKPORT “clk” CLKOUT PORT “clk”;

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

Both tDIA_GDDRX1/X2 and tDIB_GDDRX1/X2 numbers are available in the External Switching Characteristics section of CrossLink-NX Family Data Sheet (FPGA-DS-02049) for maximum speed.

Preference Example:

For GDDRX2_TX.Aligned case running at 400 MHz, tDIA_GDDRX2= tDIB_GDDRX2=0.16 ns. The preference would be –

CLOCK_TO_OUT PORT "dataout" MAX 0.16 ns MIN -0.16 ns CLKPORT "clk" CLKOUT PORT "clkout”;

Note: Refer to CrossLink-NX Family Data Sheet (FPGA-DS-02049) for the latest tDIA_GDDX1/X2 and tDIB_GDDRX1/X2 numbers.

6. CrossLink-NX Memory Interfaces

All of the DDR SDRAM interface transfers data at both the rising and falling edges of the clock. The I/O DDR registers in the CrossLink-NX device can be used to support DDR3, DDR3L, LPDDR2, and LPDDR3 memory interfaces.

These memory interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DQS strobe is a differential signal.

Figure 6.1 shows typical DDR memory signals. DDR3, DDR3L, LPDDR2 and LPDDR3 memory interfaces are typically implemented with eight DQ data bits per DQS. Therefore, a 16-bit DDR memory interface uses two DQS signals, and each DQS is associated with eight DQ bits, respectively. Both the DQ and DQS are bi-directional ports and are used to read and write to the memory.

When reading data from the external memory device, data coming into the FPGA controller is edge-aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90° before the FPGA logic can sample the read data. When writing to a DDR memory, the memory controller (FPGA) must shift the DQS by 90° to center-align with the data signals (DQ). A clock signal is also provided to the memory. This clock is provided as differential clock (CK and CK#) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read through a DLL inside the memory. The figures below show DQ and DQS timing relationships for read and write cycles.

During read, the DQS signal is low for some duration after it comes out of tristate. This state is called preamble.

The state when the DQS is low before it goes into tristate is the postamble state. This is the state after the last valid data transition.

DDR memories also require a Data Mask (DM) signal to mask data bits during write cycles. Note that the ratio of DQS to data bits is independent of the overall width of the memory. Figure 6.1 shows a typical 8-bit interface that has eight associated DQ data bits per DQS strobe signal.

The DDR3 memory module uses fly-by routing topology for the address, command, control, and clock signals. This requires the memory controller to support read and write leveling to adjust for leveled delay on read and write data transfers. LPDDR3 does not use fly-by routing but write leveling may be supported if you emulate the fly-by routing using board traces. You can see more information in the DDR pin placement and layout guidelines section of this document.

One major difference between DDR3/DDR3L and LPDDR2/LPDDR3 is lack of DLL in LPDDR2/LPDDR3 memory device. This means that in LPDDR2 and LPDDR3, the clock to data output delay from memory device is not compensated by DLL as in traditional DDR3, thus the delay is much larger and has larger spread. Theoretically, there is no low frequency limitation on LPDDR2/LPDDR3, although most manufactures place a low limit of 10 MHz. Note that on the FPGA memory controller side, DLL is still needed to manage write and read phase shift, for all memory interfaces including LPDDR2 and LPDDR3.

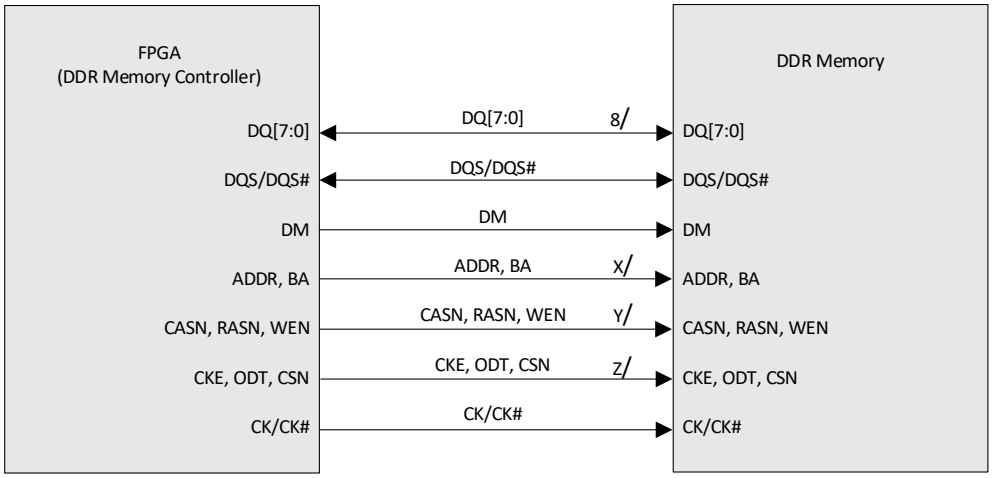


Figure 6.1. Typical DDR3/DDR3L Memory Interface

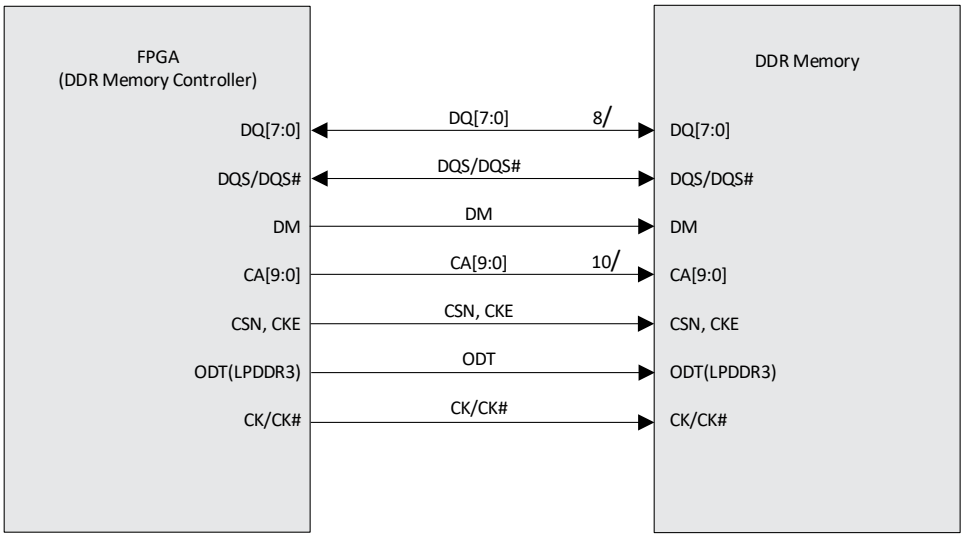


Figure 6.2. Typical LPDDR2/LPDDR3 Memory Interface

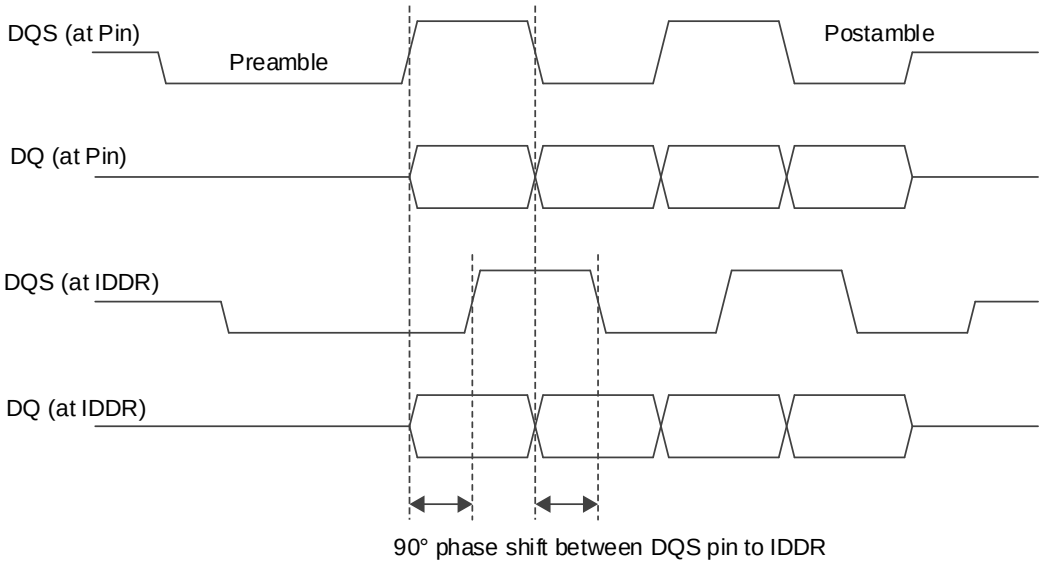


Figure 6.3. DQ-DQS During Read

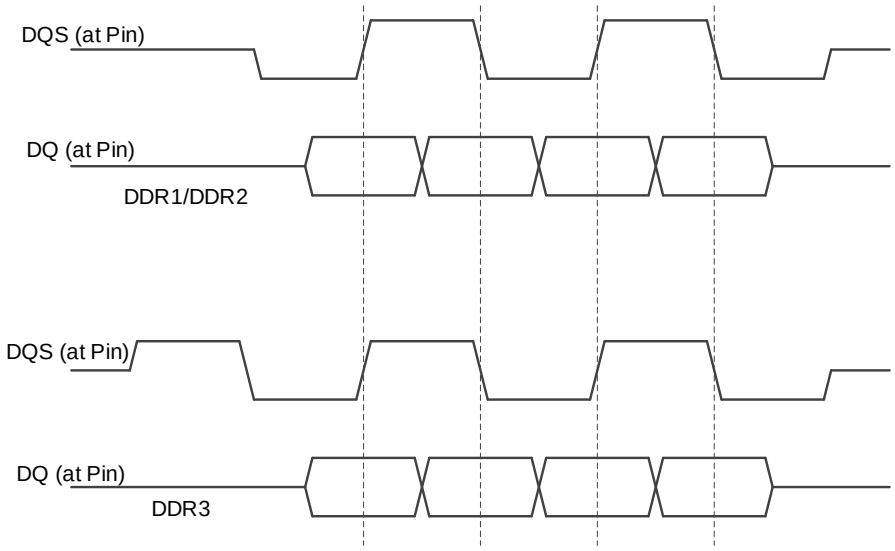


Figure 6.4. DQ-DQS During Write

Table 6.1 below shows the different DDR memory configurations and features supported by the CrossLink-NX device.

Table 6.1. DDR Memory Configurations Support

DDR Memory	Data Width	VCCIO	DQ	DQS	Modules Types	Rank	Chip Selects	Write Leveling	CMD/ADDR Timing	Fmax
DDR3	8, 16, 24, 32 bits	1.5 V	SSTL15_I	SSTL15D_I	UDIMM, SODIMM, RDIMM	Single, Dual	1, 2,4	Yes	2T[3]	533 MHz
	8, 16, 24, 32 bits	1.5 V	SSTL15_I	SSTL15D_I	Embedded	Single, Dual	1, 2,4	Yes[1]	2T[3]	533 MHz
DDR3L	8, 16, 24, 32 bits	1.35 V	SSTL135_II	SSTL135D_II	UDIMM, SODIMM, RDIMM	Single, Dual	1, 2,4	Yes	2T[3]	533 MHz
	8, 16, 24, 32 bits	1.35 V	SSTL135_II	SSTL135D_II	Embedded	Single, Dual	1, 2,4	Yes[1]	2T[3]	533 MHz
LPDDR2	16 and 32 bits	1.2 V	HSUL12	HSUL12D	Embedded (Single Channel)	Single	1	No	ODDRX2	533 MHz
LPDDR3	16 and 32 bits	1.2 V	HSUL12	HSUL12D	Embedded (Single Channel)	Single	1	Yes[2]	ODDRZ2	533 MHz

Notes:
1. If fly-by wiring is implemented.
2. Fly-by wiring is emulated using board traces (guidelines are in the section below) CSN uses 1T timing.

6.1. DDR Memory Interface Requirements

As described in the overview section, all the DDR memory interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the CrossLink-NX device is edge-aligned with respect to the DQS signal. Therefore, the CrossLink-NX device needs to shift the incoming DQS (90° phase shift) before using it to sample the read data.

To implement the write portion of a DDR memory interface, parallel single data rate data must be multiplexed depending on the IDDR and ODDR register gearing mode together with data transitioning on both edges of the clock. In addition, during a write cycle, the CrossLink-NX devices generate a DQS signal that is center aligned with the DQ, the data signal. This is accomplished by ensuring a DQS strobe is 90-degree shifted relative to the DQ data. The CrossLink-NX devices provide the solutions to achieve the following design challenges to implement DDR memory write functions:

- DQ/DM needs to be center-aligned to DQS.
- DQS needs to be edge-aligned to CK. In DDR3 interfaces where fly-by routing is used, write leveling should be used to compensate for skews between CK and DQS.
- The DDR output data must be multiplexed into a single outgoing DDR data stream.
- Generate ADDR/CMD signal edge-aligned to CK falling edge to maximize the tIS and tIH timing parameters.
- Differential CK signals (CK and CK#) need to be generated.
- The controller must meet the DDR interface specification for the tDSS and tDSH parameters, defined as DQS falling edge setup and hold time to/from CK rising edge, respectively. The skews, if caused by the fly-by topology, are compensated by write-leveling.
- In case of LPDDR2 and LPDDR3, the CA[9:0] bus needs to be 90 degrees from the CLKP/CLKN signal.
- For DDR3 memory, the memory controller also needs to handle the write leveling required by the interface when the fly-by topology is applied.

6.2. Features for Memory Interface Implementation

The CrossLink-NX devices contain a variety of features to simplify implementation of the read and write operations of a DDR interface:

- DQS Clock Tree spanning the DQS group
- DDRDLL used to generate the 90-degree delay codes
- DLL-compensated DQS delay elements
- Input FIFO for read data clock domain transfer
- Dedicated DDR Memory input and output registers
- Dynamic Margin Control Circuit to adjust Read and Write delays
- Input/Output Data Delay used to compensate for DQS clock tree delay

6.2.1. DQS Grouping

In DDR interfaces with eight DQ pads associated to one DQS pad, each DQS group generally consists of at least 11 I/O (two DQS, eight DQ, one DM) to implement a complete 8-bit DDR3/DDR3L/LPDDR2/LPDDR3 memory interface. In case of LPDDR2/LPDDR3, two additional DQS groups are required to generate the CA[9:0] (with 10 I/O) and Control/CLKP/CLKN (with five I/O for LPDDR3 and four I/O on LPDDR2) outputs.

In CrossLink-NX devices, a DQS group consists of 11 to 16 I/O depending on the device and package selected to accommodate these DDR interface needs. CrossLink-NX devices support DQS signals on the bottom side of the device.

Each DQS signal spans across 11 to 16 I/O. Any 11 (for DDR3/DDR3L/LPDDR2/LPDDR3) of these 16 I/O spanned by the DQS can be used to implement an 8-bit data side interface. For LPDDR2/LPDDR3, any group with ten I/O is required for CA[9:0] bus and another group with five I/O for LPDDR3 and four I/O for LPDDR2 is required to generate the Control and CLKP/CLKN outputs. In addition to the DQS grouping, you must also assign the reference voltage (VREF) input to an I/O in that bank required to implement the referenced I/O standard.

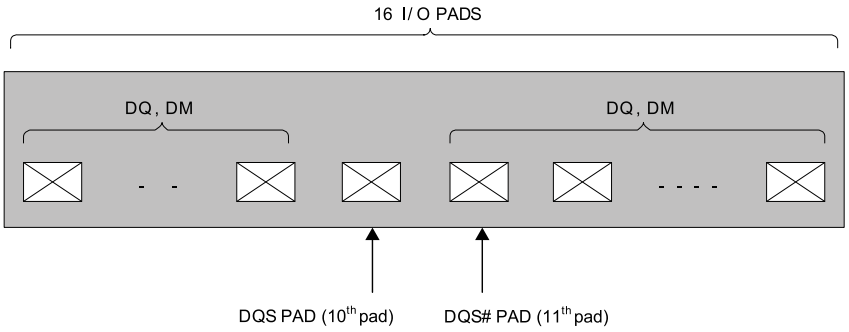


Figure 6.5. DQ-DQS Grouping

Figure 6.5 shows a typical DQ-DQS group for CrossLink-NX devices. The 10th I/O Pad of this 16 I/O group is the dedicated DQS pin. All the nine pads before the DQS and six pads after the DQS are covered by this DQS bus span. If a differential DQS pair is required then the 11th pad is used by the DQS# signal. You can assign any other I/O pads to be DQ data or DM pins. Therefore, for example, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups when eight-to-one DQ-DQS association is used.

In case of LPDDR2/LPDDR3, two additional DQS groups are required to assign the CA[9:0] and the control signals.

In case of DDR3/DDR3L, additional I/O pads are required to implement address, command, and control. They do not have to be I/O in DQS groups but need to use 1X gearing generic output DDR capable pads.

Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pin out sheets included as part of CrossLink-NX Family Data Sheet (FPGA-DS-02049) shows pin locations for each of the DQS groups.

6.2.2. DLL-Compensated DQS Delay Elements

The DQS to and from the memory is connected to the DQS delay element inside the CrossLink-NX device. The DQS delay block receives the delay control code, DDRDEL, from the on-chip DDRDLL. The code generated by DDRDLL is connected to the DQSBUF circuit to perform 90-degree read phase shift and 90-degree write phase shift. DDRDLL requires the frequency reference from PLL, normally going through the Edge Clock tree.

CrossLink-NX devices support one DDRDLL modules in each corner of the device. The DQSBUF modules that receive the DDRDEL code from either bottom left or bottom right DDRDLL. Hence, each side can support up to two different rate DDR memory interfaces.

Table 6.2. DDRDLL Connectivity

DDRDLL Location	Left DQSBUFs	Right DQSBUFs
DDRDLL_BR	—	X
DDRDLL_BL	X	—

The DQS received from the memory is delayed in the DQS delay element in the DQSBUF block, and this delayed DQS is used to clock the first set stage DDR input registers.

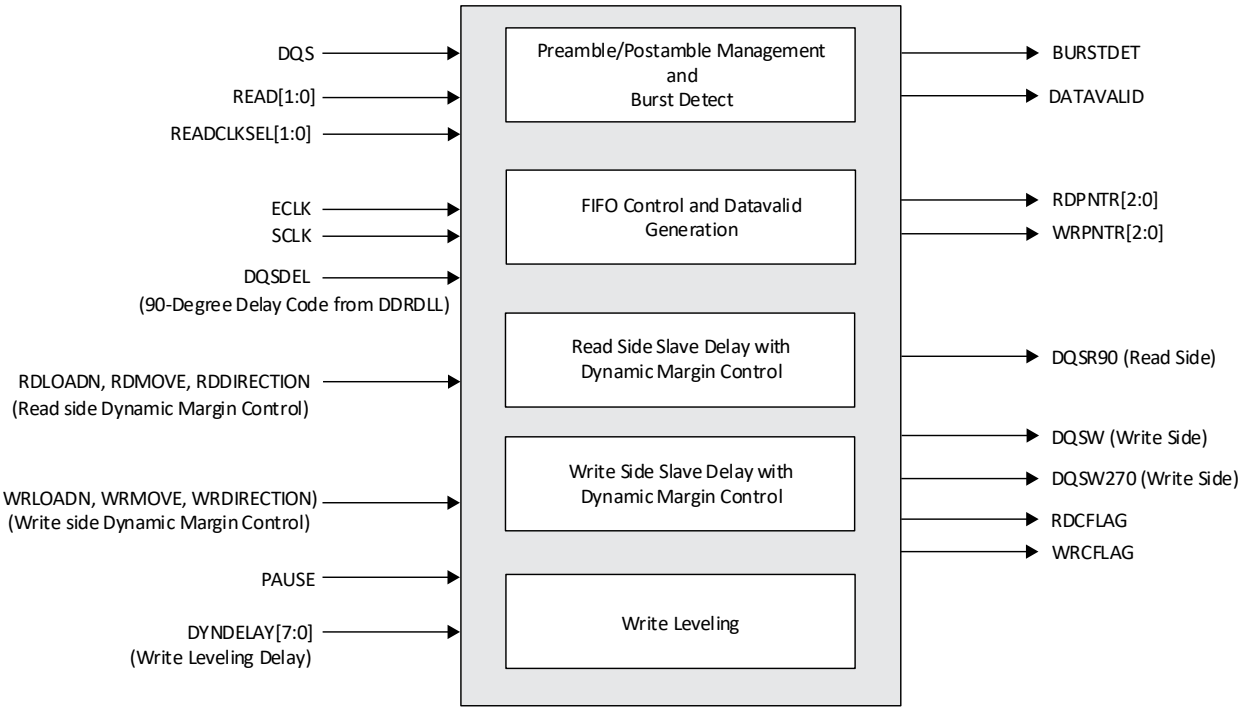


Figure 6.6. DQSBUF Block Functions

6.2.3. Data Valid Module

The DQSBUF block generates a DATAVALID signal. This signal indicates the timing that the IDDR module drives the valid read data transmitting to the FPGA fabric. DATAVALID is a level-sensitive active high signal and indicates that the read data output from the IDDR module is valid while it is asserted high. The DATAVALID signal can stay asserted during the IDDR module outputs back-to-back valid read data until all consecutive read operations are completed.

6.2.4. READ Pulse Positioning Optimization

The memory controller is required to provide the READ1/0 signal to the DQSBUF block to position the internal READ pulse and generate the DATAVALID output that indicates the proper timing window of the valid read data. The internal READ pulse is also used to get a clean internal DQS signal between the preamble and postamble periods. The clean DQS is 90-degree shifted internally to be used to capture the read data.

Due to the DQS round trip delay that includes PCB routing and I/O pad delays, proper positioning of the READ pulse is crucial for successful read operations. The CrossLink-NX device DQSBUF block provides the dynamic READ pulse positioning function, which allows the memory controller to locate the READ pulse to an appropriate timing window for the read operations by monitoring the positioning result.

The READCLKSEL2/1/0 and BURSTDET signals are used to accomplish the READ pulse positioning function for a corresponding DQSBUF block. The READ1/0, READCLKSEL2/1/0 signals are driven by the user logic and are part of the DDR memory controller.

The READ1/0 signal needs to be asserted high a certain amount of time before the read preamble starts. The suggested READ1/0 signal assertion timing and the required duration of assertion are listed in Table 6.3. When the internal READ pulse is properly positioned, BURSTDET is asserted high and guarantee that the generated DATAVALID signal properly indicates the valid read data time window. The READ1/0 signal must stay asserted as long as the number of SCLK cycles that is equal to one fourth of the total burst length as listed in Table 6.3.

Table 6.3. DDRDLL Connectivity

Gearing Mode	READ Control	DQSBUF Block	Initial READ Assertion Position*	READ Width in SCLK
X2 Gearing (All DDR Memory Interfaces)	READ1 READ0 READCLKSEL2 READCLKSEL1 READCLKSEL0	DQSBUFM	At least 5.5T before preamble	Total Burst Length/4

***Note**: Subject to change after validation tests. The number shown does not include DQS round trip delay.
1T = 1 tCK memory clock cycle

Once the controller initially positions the internal READ pulse using the READ1/0 and READCLKSEL2/1/0 signals, BURSTDET can be used to monitor the positioning result to optimize the READ pulse position. The BURSTDET signal provides a feedback mechanism to inform the memory controller whether the READ pulse has reached to the optimal position for the read operations or not with the current READ1/0 and READCLKSEL2/1/0 values. When it reaches to the optimal position, BURSTDET is asserted High after a read operation. Otherwise, BURSTDET remains Low. A minimum burst length of eight on the memory bus must be used in the training process. This can be done either with two consecutive BL4 (BL=4) read accesses or one BL8 read access. Any even number of BL4, or any multiplication of BL8 (BL=8) can also be used. This read pulse training process must be performed during the initial training and can also be periodically calibrated during the normal operations.

The BURSTDET signal is asserted after the last DQS transition is completed during a read operation and lasts until the next read cycle is started. Once a read operation is started, the memory controller should wait until the DATAVALID signal from DQSBUFM is asserted and then sample the BURSTDET signal at the next cycle to monitor the READ pulse positioning result. If there is no assertion on BURSTDET, it means that the READ pulse has not been located to the optimal position yet. Then, the memory controller needs to shift the READ1/0 signal and/or increase the READCLKSEL2/1/0 value until it detects a BURSTDET assertion. It is recommended that at least 128 read operations be performed repetitively at a READ pulse position during the initialization for getting jitter immunity. 16 read operations can be performed in a periodic calibration if used during the normal operation. The memory controller can determine the proper position alignment when there is no failure on BURSTDET assertions during these multiple trials.

To reposition the internal READ pulse:

1. The memory controller sets READ1/0 to an initial position before starting the read pulse training. READ1/0 must be asserted for the number of SCLK cycles that is equal to one-fourth of the current read burst length as listed in Table 6.3. Each READ bit (READ1 or READ0) in the system clock domain is translated to a 1T time slot of the memory clock domain as shown in Figure 6.7.
2. Once READ1/0 positions the READ pulse, READCLKSEL2/1/0 can be used to shift the READ pulse by 1/4T per step. With the total eight possible combinations from 000 to 111, READCLKSEL2/1/0 covers the READ pulse shift up to a whole 2T timing window. If BURSTDET is asserted with a certain READCLKSEL2/1/0 value, it indicates that the READ pulse has been located to the optimal position. If no BURSTDET is asserted during this step, the READ pulse needs to be moved to the next timing window.
3. To shift the READ pulse timing window, READ1/0 can be moved to the next cycle. If a READ bit is asserted in the next cycle while the other READ bit remains in the current cycle, only the corresponding time slot on the READ pulse moves to the next allotted slot. Therefore, only READ0 must be moved to the next cycle if the READ pulse needs to be shifted only by 1T. However, if only READ1 is moved to the next SCLK cycle, then the READ pulse gets two short pulses in wrong timing. Since READCLKSEL2/1/0 covers a whole 2T timing, it is recommended that both READ1 and READ0 be moved to the next cycle together to shift the READ pulse to the next 2T window as the example shown in Figure 6.7.

Repeat *step 2* and *step 3* until BURSTDET is asserted.

Figure 6.7 shows an example of a burst length 8 (BL8) read operation. The bottom side of the diagram indicates the case that the incoming DQS (DQSI) gets slightly more than 2T delay after a round trip. Due to this round trip delay, both READ1 and READ0 need to be shifted to the next SCLK cycle so that the internal READ pulse gets a 2T shift. Then, the READCLKSEL2/1/0 signals can be used to fine-tune the READ signal position throughout the training process. When any of READCLKSEL2/1/0 is changed at any time after a system reset, the PAUSE input to DQSBUFM must be asserted before 4T of the change and remain asserted for another 4T after the change to avoid glitches and malfunction.

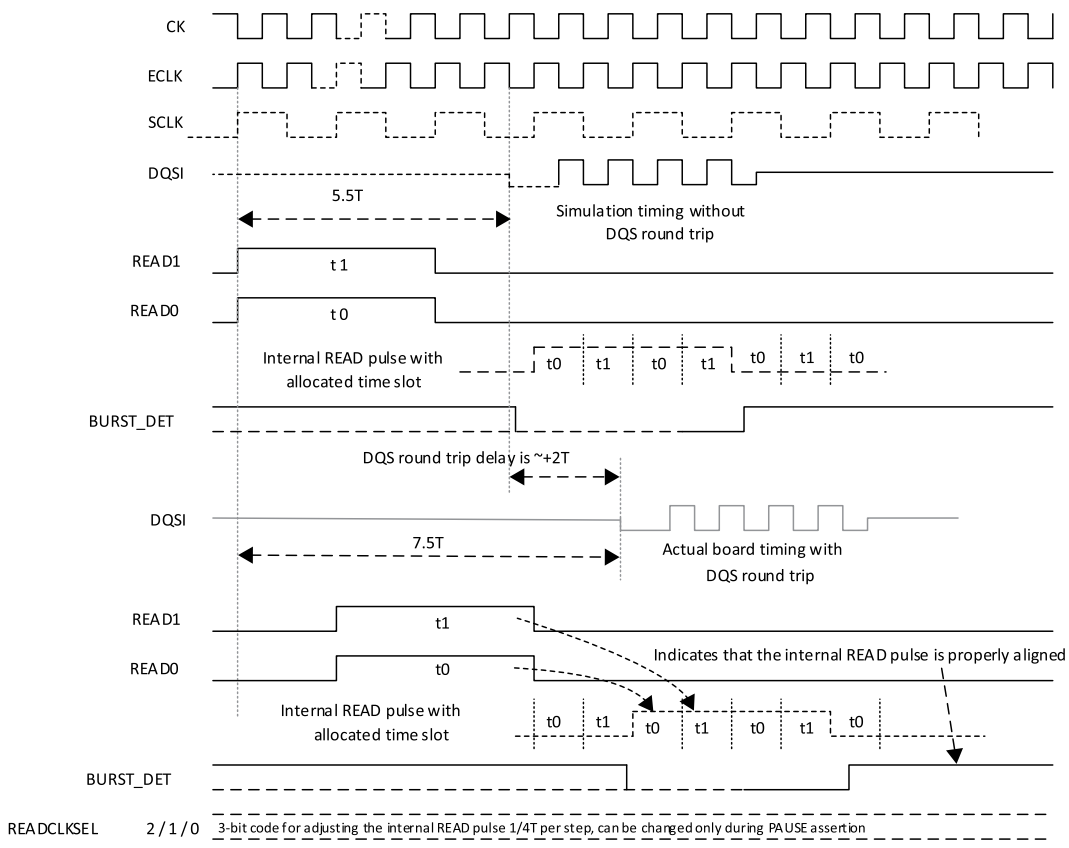


Figure 6.7. READ Signal Training Process

Note that the DYNDELAY[7:0] signal, margin control signals (WR/RDMOVE, WR/RDLOADN) and DDRDLL update signal (UDDCNTLN) also have the same PAUSE requirement.

6.2.5. Dynamic Margin Control on DQSBUF

The CrossLink-NX family includes dynamic margin control signals in the DQSBUF module allows you to dynamically adjust the read or write side DQS delays generated in the DDRDLL.

Once the margin control mode is enabled by de-asserting WRLOADN (=1) or RDLOADN (=1), the DQSBUF's phase shift control to make a center aligned interface is no longer controlled by the DDRDLL component. You must complete the margin control training to maximize the valid window and then continuously monitor the DDRDLL delay code (DCNTL7~DCNTL0) and controls the DQSBUF delays accordingly using the WR/RDMOVE and WR/RDDIRECTION signals to compensate the PVT variations.

6.2.6. Read Data Clock Domain Transfer Using Input FIFO

Each IDDR module in the CrossLink-NX device has a dedicated input FIFO to provide a safe clock domain transfer from the DQS domain to the ECLK or SCLK domain. The input FIFO is 8-level deep with 3-bit write and read pointers. It transfers the read data from the non-continuous DQS domain to the continuous ECLK. The FIFO is written by the DQS strobe and read back by ECLK, which has the identical frequency rate as DQS.

The input FIFO also performs the read leveling function. When each DQS strobe signal and its associated DQ data signals arrive at slightly different time with others to the FPGA, the input FIFO allows the skewed read data to be captured and transferred properly.

Each DQS group has one FIFO control in the DQSBUF block. It distributes the FIFO read/write pointers, WRPNTR [2:0] and RDPNTR [2:0], to each memory IDDR module in the same DQS group. Safe domain crossing between ECLK and SCLK is guaranteed by the CrossLink-NX device hardware design.

6.2.7. DDR Input and Output Registers (IDDR/ODDR)

CrossLink-NX devices provide dedicated input DDR (IDDR) and output DDR (ODDR) functions supporting 4:1(X2) gearing modes that are used to implement the DDR memory functions. These automatically handle the transfer of data from ECLK domain to the SCLK (FPGA clock) domain.

6.3. Memory Interface Implementation

The following sections explain the DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces implementation using the X2 gearing mode. CrossLink-NX devices support these memory interfaces generation through the IP Catalog tool. IP Catalog generates one module that includes the Read and Write side implementation shown below.

All of the memory interfaces use DQS clock, one ECLK and one SCLK to implement the read and write side operations. ECLK must always be routed on the Edge Clock tree and SCLK on the primary clock tree.

6.3.1. Read Implementation

The read side implementation is shown in Figure 6.8.

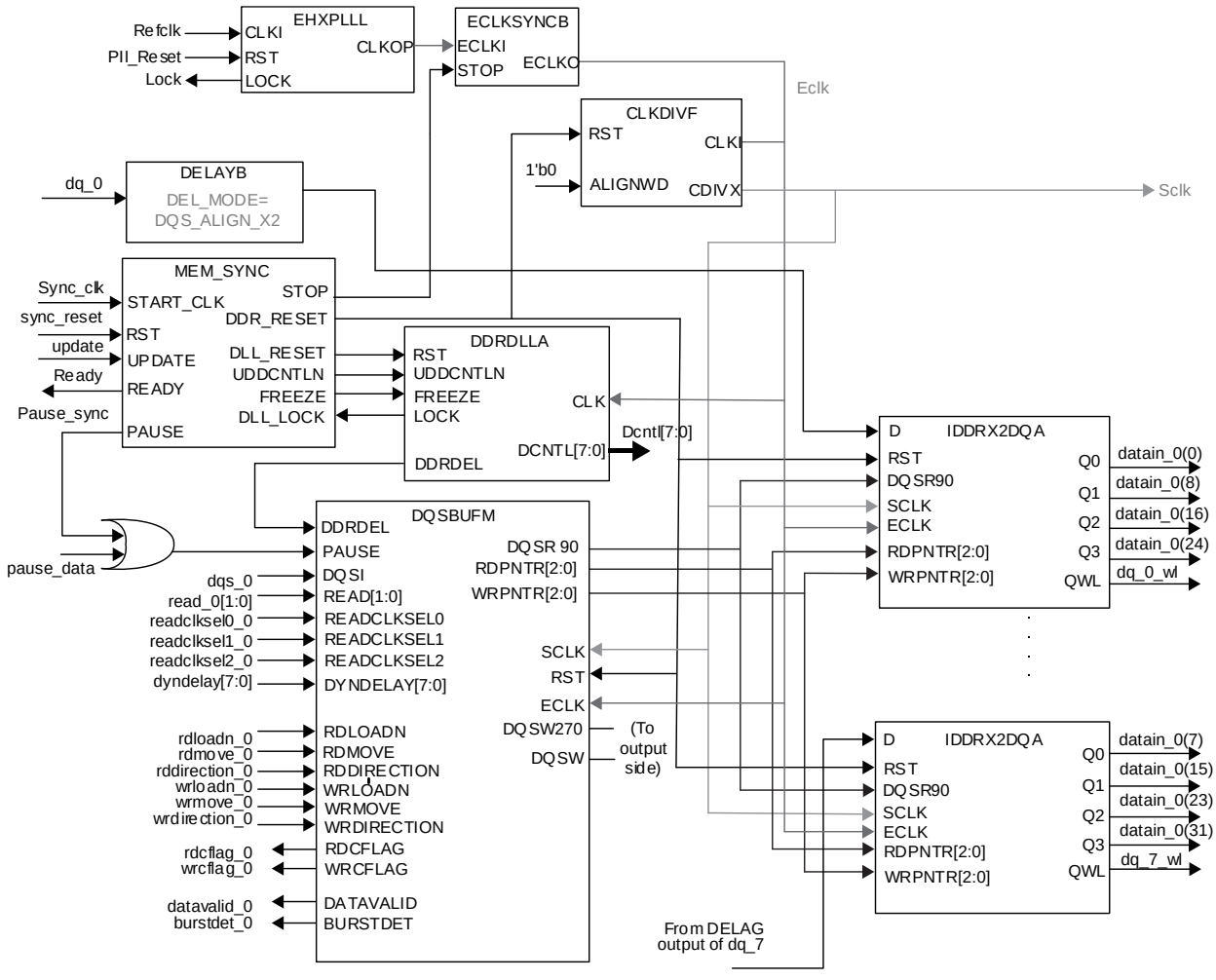


Figure 6.8. DDR3/DDR3L and LPDDR2/LPDDR3 Read side Implementation

The read side is implemented using the following software elements.

- DDRX2DQA element to capture the data.
- DDRDLLA is used to generate the delay code for DQSBUFM to get the 90-degree phase shift on the DQS input (DQSR90).
- The incoming DQS clock (DQSI) is routed through the DQSBUFM module to the DQS clock tree.
- The DQSBUFM receives the delay code from DDRDLLA and generates the delayed DQS signal to IDDRX2DQA.
- The DQSBUFM is used to generate the Read and Write pointers that is used to transfer data from the DQS to ECLK inside the IDDRX2DQA module.
- Read 1, 0, and Readclksel_2, 1, 0 signals of DQSBUFM are used by the user logic to obtain the optimal READ pulse position and driven by the user logic to generate a clean DQS output signal based on the trained READ pulse with respect to preamble and postamble.
- The dynamic delay control ports are available on the DQSBUFM module when you select the *enable dynamic margin control* option.
- DYNDELAY[7:0] of DQSBUFM is used to perform write leveling. If write leveling is not used, it is connected to 0.
- Port QWL of IDDRX2DQA is used for DDR3/DDR3L and LPDDR3 to support write leveling. It is used to deliver the write leveling monitor signals from the memory device to the FPGA user logic.
- MEM_SYNC soft IP must always be included in the interface. It is required to avoid issues on DDR memory bus and update code in operation without interrupting interface operation. When a DDR memory interface IP is generated from IP Catalog, the MEM_SYNC soft IP block is also generated and included.
- The Pause_sync output of the MEM_SYNC soft IP is used to request DQSBUFM pause for the DDRDLL update and goes to an output port of the IP Catalog module. The input port Pause_data goes to DQSBUFM. It is required by user logic, to OR the Pause_sync output of the MEM_SYNC module with the user pause to drive the Pause_data input of the DQSBUFM. This OR would need to be implemented outside of the IP Catalog module in your design.
- CLKDIVF set to divide by 2 function is used to generate the SCLK from the ECLK.

6.3.2. Write Implementation (DQ, DQS, and DM)

Figure 6.9 shows the DDR3/DDR3L and LPDDR2/LPDDR3 memory interface write side implementation to generate DQ, DQS, and DM outputs.

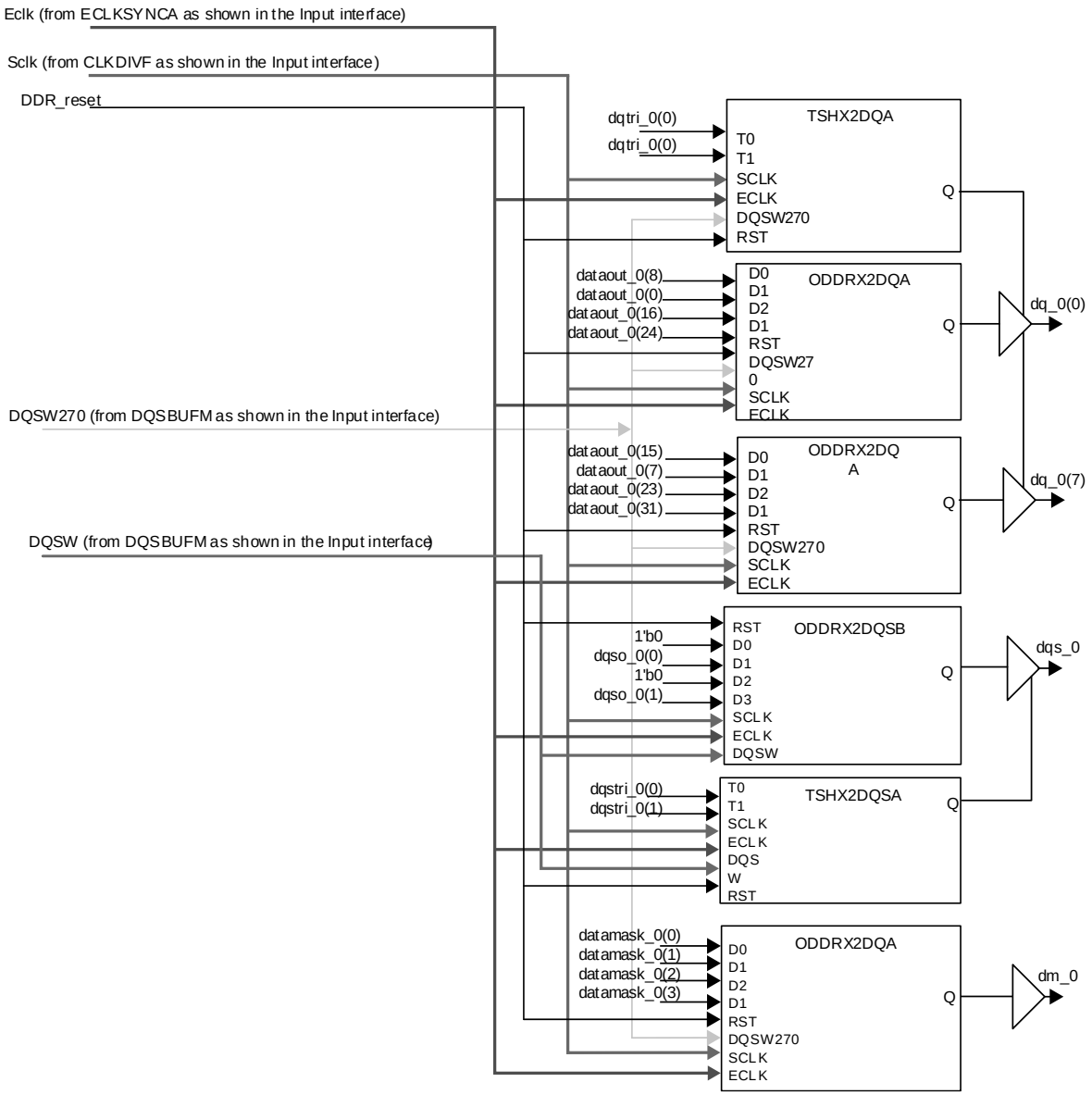


Figure 6.9. DDR3/DDR3L and LPDDR2/LPDDR3 Write Side (DQ, DQS, and DM)

This interface uses the following modules:

- ODDRX2DQA to generate the data DQ and DM signals. TSHX2DQA is used to generate the tristate control for the DQ output.
- ODDRX2DQSB to generate DQS output. TSHX2DQSA is used to generate the DQS tristate control.
- DQSW270, which is the 270 degree delayed DQS signal is used to generate the DQ and DM outputs.
- DQSW is 90 degrees shifted from the DQSW270 is used to generate DQS output.
- The DQSW270 and DQSW clocks are generated in the DQSBUFM module shown on the Read side Implementation figure.
- When write leveling is enabled, the dynamic delay for write leveling (DYNDELAY[7:0]) is applied to both DQSW and DQSW270 so that the DQ and DQS phase relationship is maintained.
- ECLK and SCLK are used inside the ODDRX2 module before data is transferred to the DQSW270 and DQSW clocks. The ECLK is generated by the EHXPLLL module and the SCLK is generated by the CLKDIVF module, both shown in the Read side implementation.

- Figure 6.9 shows one tristate. The software generates one tristate element for each DQ port.

6.3.3. Write Implementation (DDR3/DDR3L Address, Command, and Clock)

DDR3 and DDR3L write side interface side to generate the Clock, Address and Command uses the following modules:

- ODDRX2F with inputs tied to constants to generate the DDRCLK output.
- The ADDR, BA, CASN, RASN, WEN, CKE, and ODT command and address signals are generated using the ODDRX1F. CSN output is generated using OSHX2A.
- Both ECLK and SCLK is used in these elements. This is same ECLK and SLCK generated in the Read side.

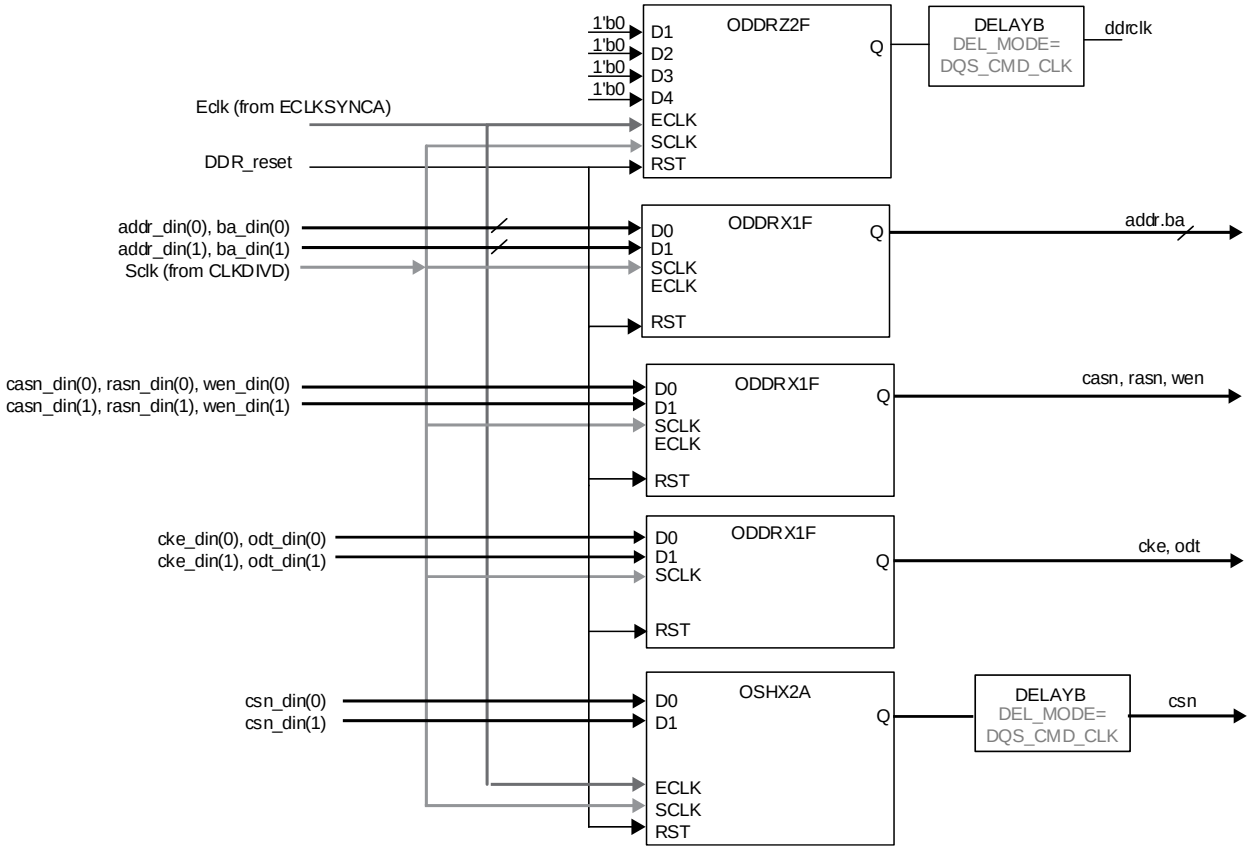


Figure 6.10. DDR3/DDR3L Address, Command, and Clock Generation

6.3.4. Write Implementation (LPDDR2 and LPDDR3 Address, Command, and Clock)

LPDDR2 and LPDDR3 output side interface side to generate the Clock, Address, and Command uses the following modules:

- Two DQSBUFM are required generate the LPDDR2 and LPDDR3 address/command and clock signals. One of the DQSBUFM is used for CA output and the other for CKE, CSN, ODT, and CLKP/CLKN (note that ODT is only for LPDDR3).
- ODDRX2DQA module to generate the CA[9:0] outputs
- ODDRX2DQSB with D0 and D1 tied together and D2 and D3 tied together to generate CSN, CKE signals
- ODDRX2DQSB with inputs tied to 0 and 1 is used to generate the CLKP/CLKN outputs.
- On LPDDR3, even the ODT has D0 and D1 tied together and D2 and D3 tied together and uses same DQSBUFM.
- The DQSBUFM used for CA requires a separate input. Ideally, you must take Pause_sync output of the MEM_SYNC module and make an OR gate with user CA pause to drive the PAUSE input port of DQSBUFM. The Pause_CA pause is connected to the user CA training logic PAUSE required. If CA training is not used, then user CA pause should be tied to GND.
- Both ECLK and SCLK is used in these elements. This is same ECLK and SLCK generated in the Input Read side module shown above.

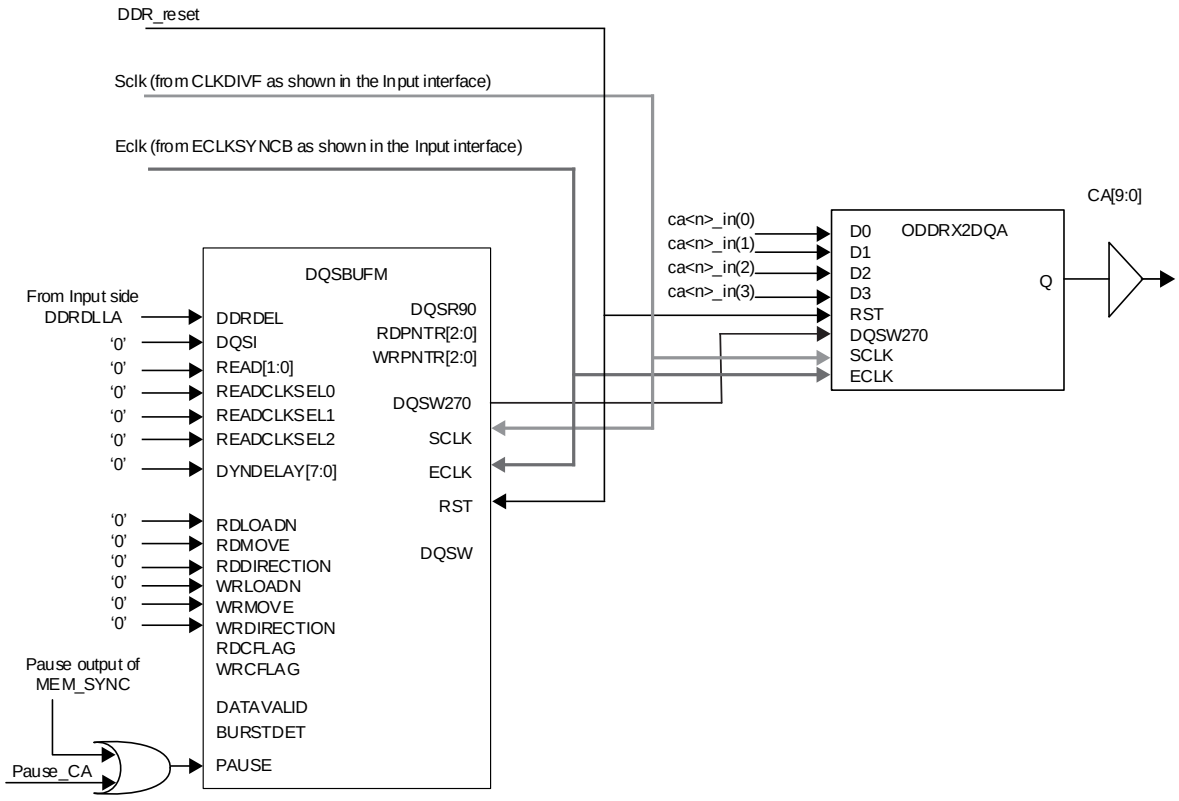


Figure 6.11. LPDDR2 Output for CA Generation

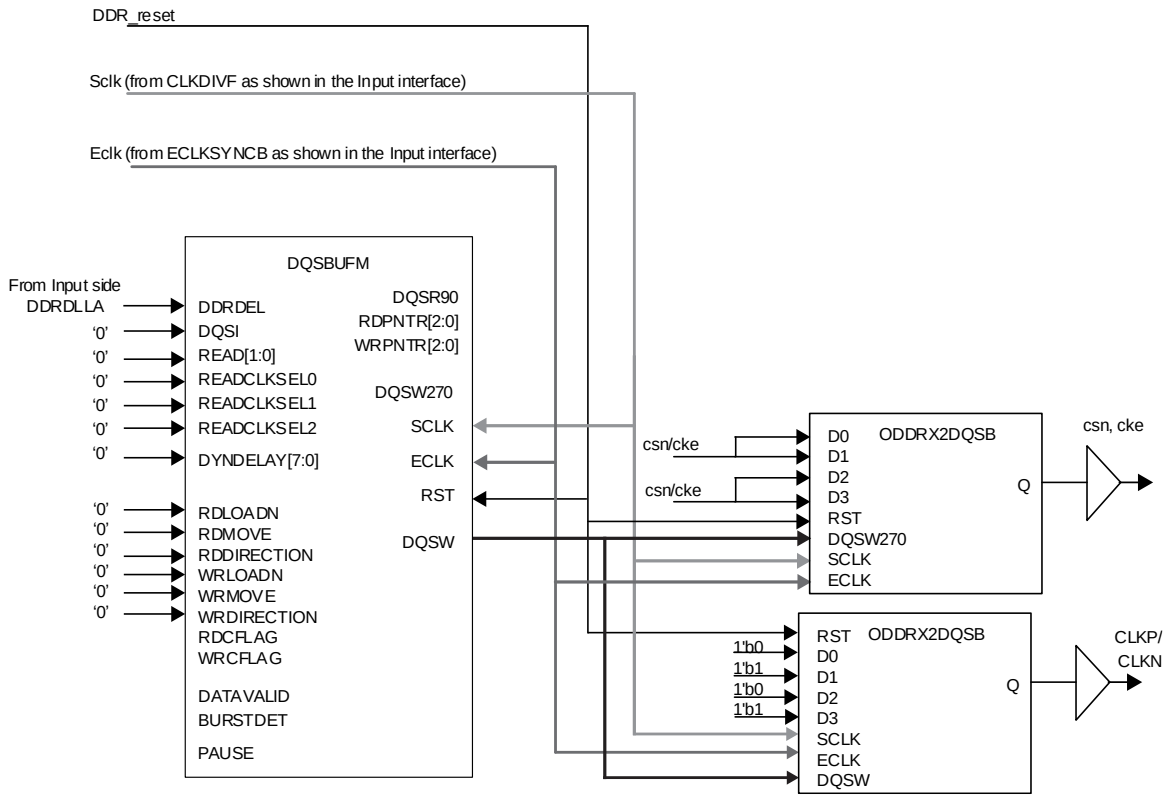


Figure 6.12. LPDDR2 Output for CSN, CKE, and CLOCK Generation

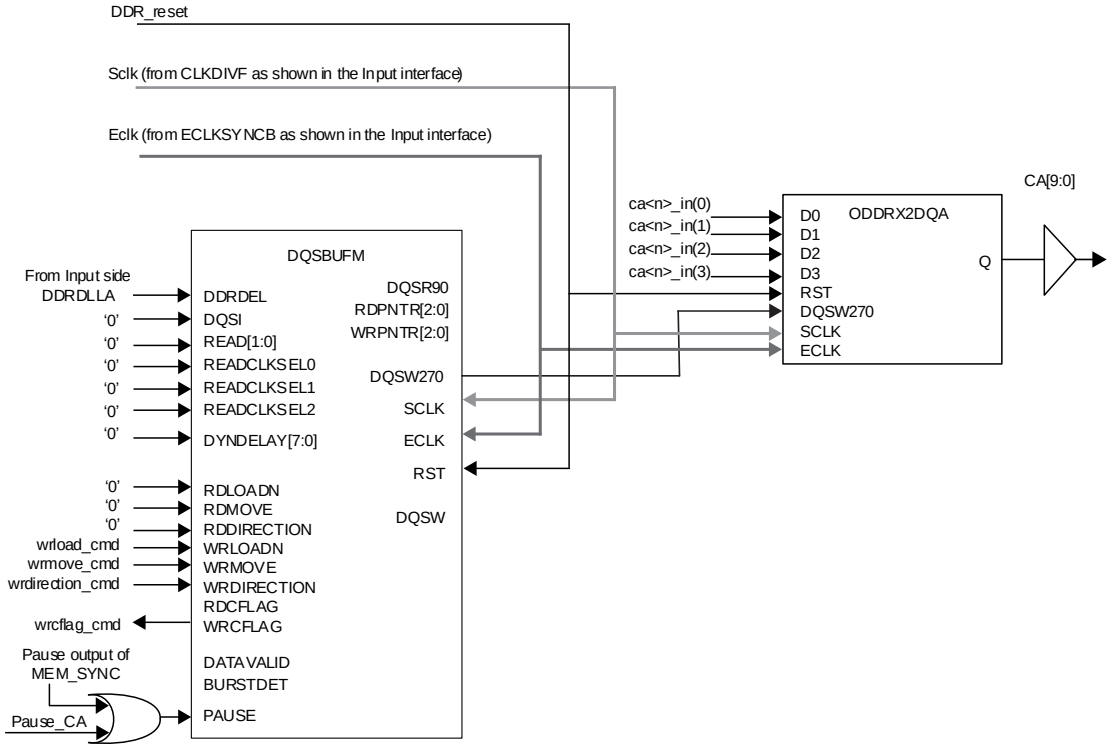


Figure 6.13. LPDDR3 Output Side for CA Generation

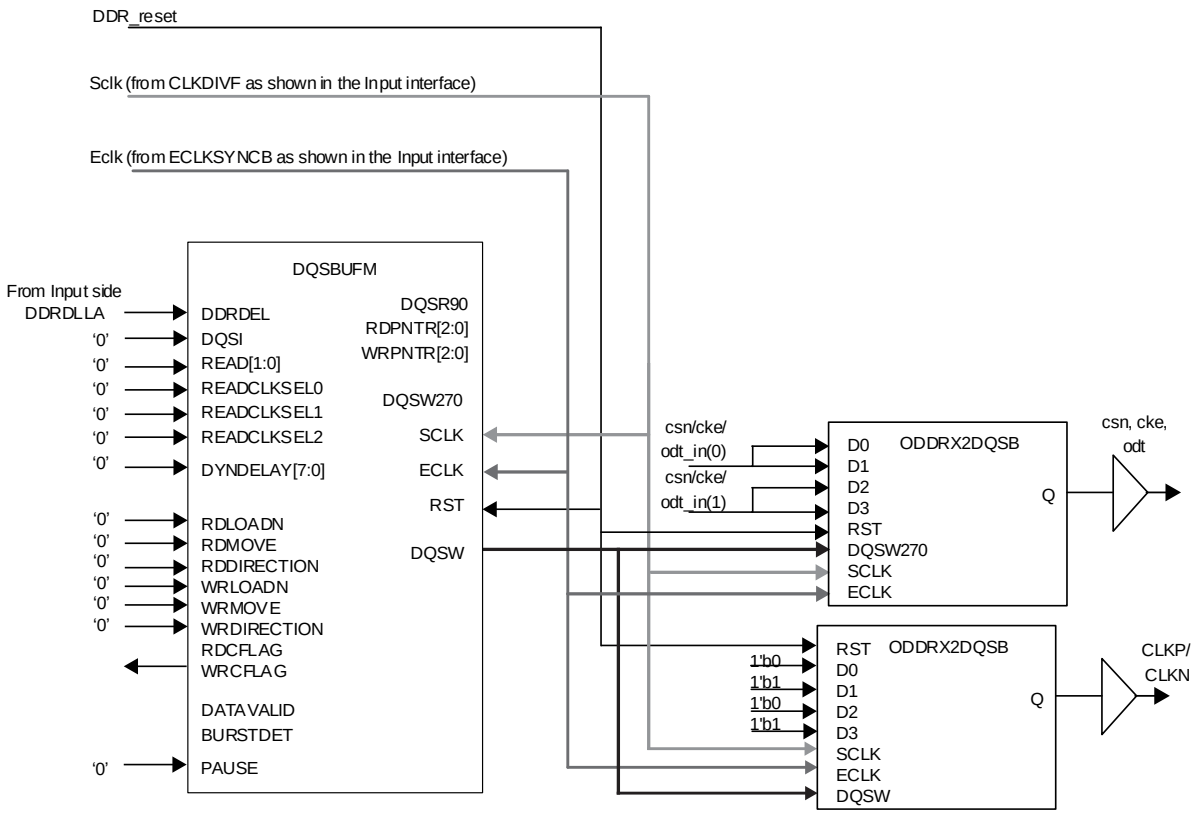


Figure 6.14. LPDDR3 Output Side for CSN, CKE, ODT, and CLOCK Generation

6.4. DDR Memory Interface Design Rules and Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in CrossLink-NX devices. CrossLink-NX devices have dedicated DQS banks with the associated DQ pads.

- DDRDLLA primitive should be instantiated for all DDR memory interfaces. Each DDRDLLA generates 90-degree digital delay code for all the connected DQS delay blocks based on the reference clock input to the DDRDLLA. Therefore, all the DDR memory interfaces under the same DDRDLLA coverage must run at the same frequency.
- There are two DDRDLLs on each device, one DDRDLL in each bottom corner of the device. Each DQSBUF module can receive delay codes from either of the DDRDLLs in bottom corner of the device.
- When a DDR memory interface is added to the side where another DDR memory interface is running at a different frequency, another available DDRDLLA for the side must be instantiated and used for the new interface.
- The reference clock input to the PLL used in the DDR memory interface implementation must be located to the dedicated PLL pin or a PCLK pin. The dedicated PLL input pin is preferred due to less skew.
- Each DDR memory interface must use its corresponding I/O standard.
  - For the DDR3 memory interface, these signals should be connected to the SSTL15 standards.
  - For the DDR3L memory interface, these signals should be connected to the SSTL135 standards.
  - For the LPDDR2/LPDDR3 memory interface, the interface signal should use the HSUL12 standard.
  - DDR3/DDR3L and LPDDR2/LPDDR3 memory interfaces also requires differential DQS signals.

Table 6.4 shows the IO_TYPE setup for each of the DDR memory interfaces.

Table 6.4. I/O Standards for DDR Memory

		DDR3	DDR3L	LPDDR2	LPDDR3
DQ		SSTL15_I, SSTL15_II	SSTL135_I, SSTL135_II	HSUL12	HSUL12
DQS		SSTL15D_I, SSTL15D_II	SSTL135D_I, SSTL135D_II	HSUL12D	HSUL12D
Cmd/Addr		SSTL15_I, SSTL15_II	SSTL135_I, SSTL135_II	HSUL12	HSUL12
CK		SSTL15D_I, SSTL15D_II	SSTL135D_I, SSTL135D_II	HSUL12D	HSUL12D

Note: When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

6.5. DDR3 Memory Interface Termination Guidelines

(These updates are still preliminary. Another update is needed once the whole validation processes are completed.)

Proper termination of a DDR memory interface is an important part of implementation that ensures reliable data transactions at high-speed. Below is the general termination guideline for the CrossLink-NX device DDR memory interface.

6.5.1. Termination for DQ, DQS, and DM

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.
- Do not locate any termination on the FPGA side. The CrossLink-NX device has internal termination on DQ and DQS, which is dynamically controlled. Use the TERMINATION preference for DQ and DQS pads to enable the internal parallel termination to VCCIO/2. The TERMINATION preference has the OFF, 50-, 60-, and 75-Ω options. (Recommended setting for each interface TBD.)

6.5.2. Termination for CK

DDR memory clocks require differential termination because they use a differential signaling. Use SSTL15D in DDR3 to drive the clock signals. You can locate an effective 100-Ω termination resistance on the memory side to achieve the differential termination using the following guideline:

- Locate a 100-Ω resistor between the positive and negative clock signal, OR
- Connect one end of an Rtt resistor to the positive pin and one end of another Rtt to the negative pin of a CK pair, then connect the other ends of two Rtt resistors together and return to VDD or GND through a Ctt capacitance. Note that the JEDEC CK termination scheme defined in the DIMM specifications uses 36-Ω for Rtt with 0.1 µF Ctt for DIMM for DDR3 DIMMs returning to VDD. 50-Ω Rtt can also be used for non-DIMM applications.
- Use of series termination resistors at the FPGA side is not recommended.

When fly-by wiring is used in DDR3, the CK termination resistor should be located after the last DDR3 SDRAM device.

6.5.3. Termination for Address, Commands, and Controls

- Parallel termination to VTT on address, command and control lines is typically required at the DDR3/DDR3L memory side.
- Locate a 50-Ω parallel-to-VTT resistor (or a best known resistance obtained from your SI simulation) to each address, command, and control line on the memory side.
- Series termination resistors can be optionally used on the address, command and control signals to suppress overshoot/undershoot and to help decrease overall SSO noise level. 22-Ω or 15-Ω series termination is recommended when used.
- When fly-by wiring is used in DDR3, the address, command, and control termination resistors should be located after the last DDR3 SDRAM device.

No termination is required on the LPDDR2/LPDDR3 CA bus and control lines. They use point-to-point connections.

6.5.4. Termination for DDR3/DDR3L DIMM

The DDR3 DIMMs incorporate internal termination following the requirements defined by the JEDEC DIMM specification. For this reason, the user termination requirement for the DDR3 DIMM is slightly different from that of DDR3 SDRAM devices:

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS, and DM is dynamically controlled by the DDR3 SDRAM’s ODT function.
- Do not locate differential termination on CK at the memory side because the DIMM already has termination on the module.
- Do not locate parallel termination to VTT on address, command, and control signals at the memory side because the DIMM already has termination on the module.
- Follow the termination for DQ, DQS, and DM guideline above for the FPGA side termination.

6.6. DDR Memory Interface Pinout Guidelines

The CrossLink-NX device contains dedicated I/O functions for supporting DDR memory interfaces. The following pinout rules must be followed to properly use the dedicated I/O functions.

- The DQS-DQ association rule must be followed.
- All associated DQs to a DQS must be in the same DQS group.
- A data mask (DM) must be part of the corresponding DQS group.
  Example: DM[0] must be in the DQS-16 group that has DQ[7:0], DQS[0].
- A DQS pad must be allocated to a dedicated DQS True (+) pad.
- A DQS# pad is auto-placed when a differential SSTL type (SSTL15D in DDR3, SSTL135D in DDR3L, and HSUL12D in LPDDR2/LPDDR3) is selected.
- Do not assign any signal to a DQS# pad if used as differential strobe. The software automatically places DQS# when a differential I/O type is applied.
- DQS signal must use the DQS/DQS# pads only.
- Data group signals (DQ, DQS, DM) can use only bottom side of the CrossLink-NX device as well as they keep the DQS-DQ association rule.
- It is recommended that the CK/CK# outputs be located on the same side where the DQ and DQS pads are located to minimize the skew.
- Place the address, command, and control signals either on the same side as where the DQ and DQS pads are located or on the top side for DDR3/DDR3L memory interfaces.
- In DDR3 interface, the RST# can be located anywhere an output is available as long as the same I/O Standard as the memory interface is applicable.
- The input reference clock to the PLL must be assigned to use dedicated clock routing. The dedicated PLL input pads are recommended while PCLK inputs can also be used.
- VREF1 of the bank where the DQ, DQS, and DM pads are located must be available to be used as a reference voltage input.
  - Do not assign an I/O signal to VREF1 in the preference file. VREF1 for the bank has to be available for the DDR memory interface.
  - Unused VREF1 can be taken as a general purpose I/O in the bank where no DQ/DQS pad is located.

6.7. Pin Placement Considerations for Improved Noise Immunity

In addition to the general pinout guidelines, you need to pay attention to additional pinout considerations to minimize simultaneous switching noise (SSN) impact. The following considerations are generally necessary to control SSN within the required level:

- Properly terminated interface
- SSN optimized PCB layout
- SSN considered I/O pad assignment
- Use of pseudo power pads

The guidelines listed below address the I/O pad assignment and pseudo power pad usage. Unlike the pinout guidelines, they are not absolute requirements. However, it is recommended that the pin placement follow the guidelines as much as possible to increase the SSO/SSI immunity.

- Place the DQS groups for data implementation starting from the middle of the (right or left) edge of the CrossLink-NX device. Allow a corner DQS group to be used as a data group only when necessary to implement the required width.
- Locate a spacer DQS group between the data DQS groups if possible. A DQS group becomes a spacer DQS group if the I/O pads inside the group are not used as data pads (DQ, DQS, DM).
  - In DDR3/DDR3L, the pads in a spacer group can be used for address, command, control, or CK pads as well as for user logic or the pseudo power pads.
  - It would provide better noise immunity if no more than two data DQS groups are consecutively placed. If more data DQS groups need to be placed consecutively, use the pseudo power pads as many as possible to isolate each DQS group more effectively from others.
- It is recommended that you locate a few pseudo VCCIO/ground (GND) pads inside a spacer DQS group and at least one pseudo VCCIO in the data DQS group. An I/O pad becomes a pseudo power pad when it is configured to OUTPUT with its maximum driving strength (that is, SSTL15, 10 mA for DDR3) and connected to the external VCCIO or ground power source on the PCB.
  - Your design needs to drive the pseudo power I/O pads according to the external connection. (that is, you assign them as OUTPUT and let your design drive *1* for pseudo VCCIO pads and *0* for pseudo GND pads in your RTL coding.)
  - Locating two to four pseudo power pads in a spacer DQS group should be sufficient to provide suppressing the SSN impact.
  - Locate a pseudo power pad in a location where it can provide the best balanced and isolated separation.
- You may have one or more remaining pads in a data DQS group, which are not assigned as a data pad in a DDR memory interface. Assign them to pseudo VCCIO or pseudo GND. Preferred location is in the middle of the group (right next to a DQS pad pair) if the DQS group is isolated by a spacer DQS group. If consecutively placed, locating the pseudo power pads to the edge of the group may be more effective. Note that you may not have this extra pad if the DQS group has 12 pins only and includes a VREF pad for the bank.

The additional guidelines below are not as effective as the ones listed above. However, following them is still recommended to improve the SSN immunity further:

- Assign the DM (data mask) pad in a data DQS group close to the other side of DQS pads where a pseudo power pad is located. If the data DQS group includes VREF1, locate DM to the other side of VREF with respect to DQS. It can be used as an isolator due to its almost static nature in most applications.
- Other DQS groups (neither data nor spacer group) can be used for accommodating DDR memory interface's address, command, control, and clock pads.
- You can assign more unused I/O pads to pseudo power if you want to increase the SSN immunity. Note that the SSN immunity does not get increased at the same rate as the increased number of pseudo power pads. The first few pseudo power pad placements described above are more crucial. Keep the total pseudo power pad ratio (VCCIO versus GND) between 2:1 to 3:1.
- It is a good idea to shield the VREF pad by locating pseudo power pads around it if extra pins are available in the bank where the VREF1 pad is not located.

7. 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.

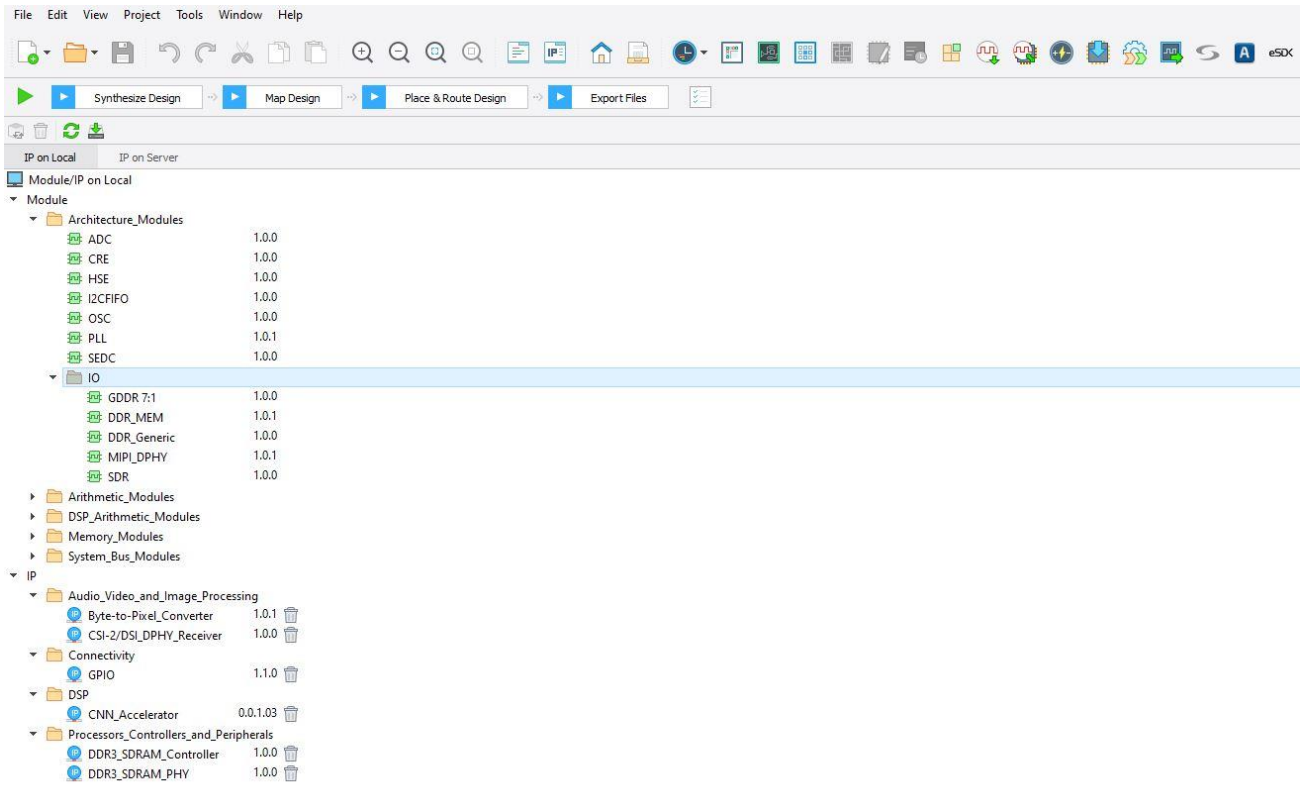


Figure 7.1. IP Catalog Main Window

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

7.1. Configuring DDR Modules in IP Catalog

IP Catalog lists all the DDR architecture modules available on CrossLink-NX.

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.
- DDR_MEM – Select to build DDR Memory Interfaces.
- MIPI_DPHY – Select to build MIPI D-PHY Receiver and Transmit Interface

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

7.2. Configuring SDR Modules

To build and SDR interface, select SDR option under Architecture Modules – I/O in the IP Catalog tab. Enter the name of the module. Figure 7.2 shows the type of interface selected as SDR and module name entered. This module can then be configured by clicking the Next button.

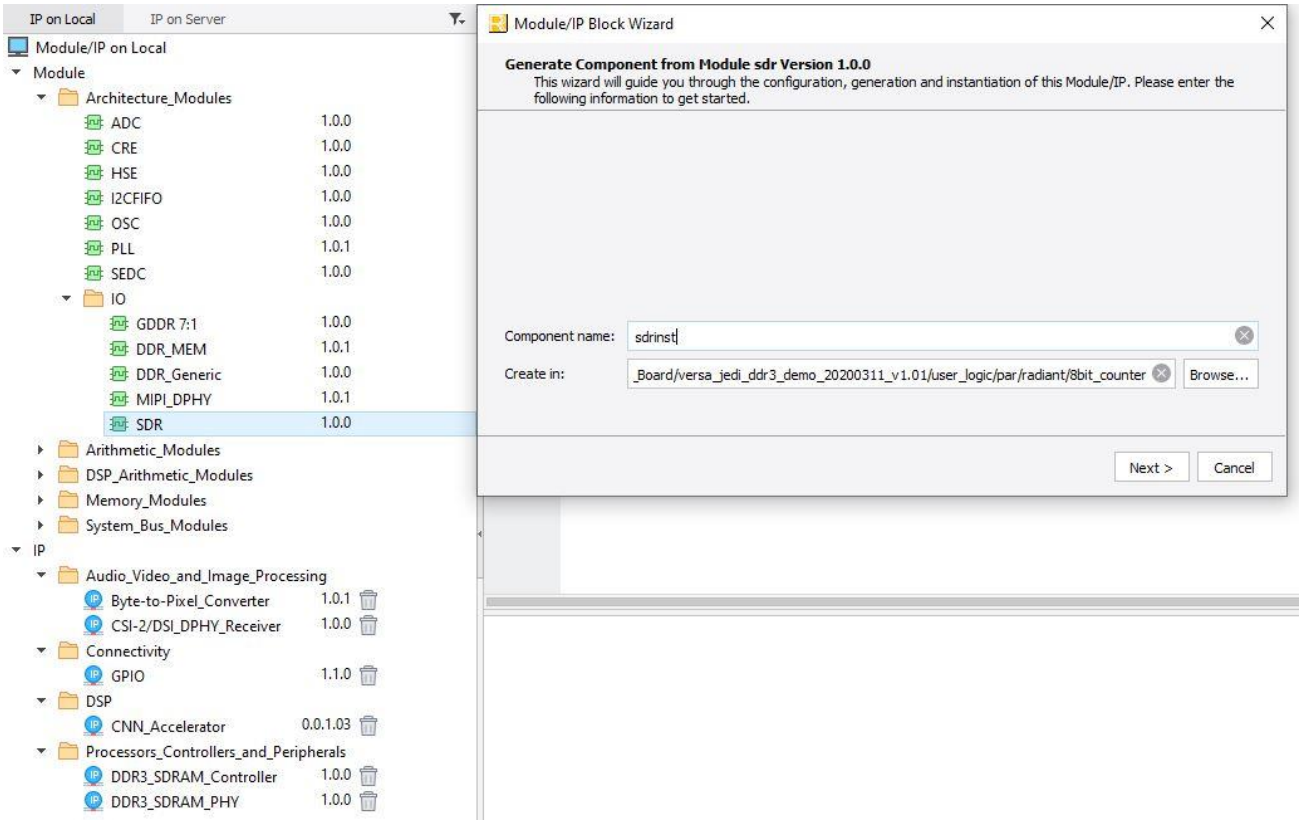


Figure 7.2. SDR Option Selected in the IP Catalog Tab

Figure 7.3 shows the Configuration tab for SDR module. You can make selections and then click Generate.

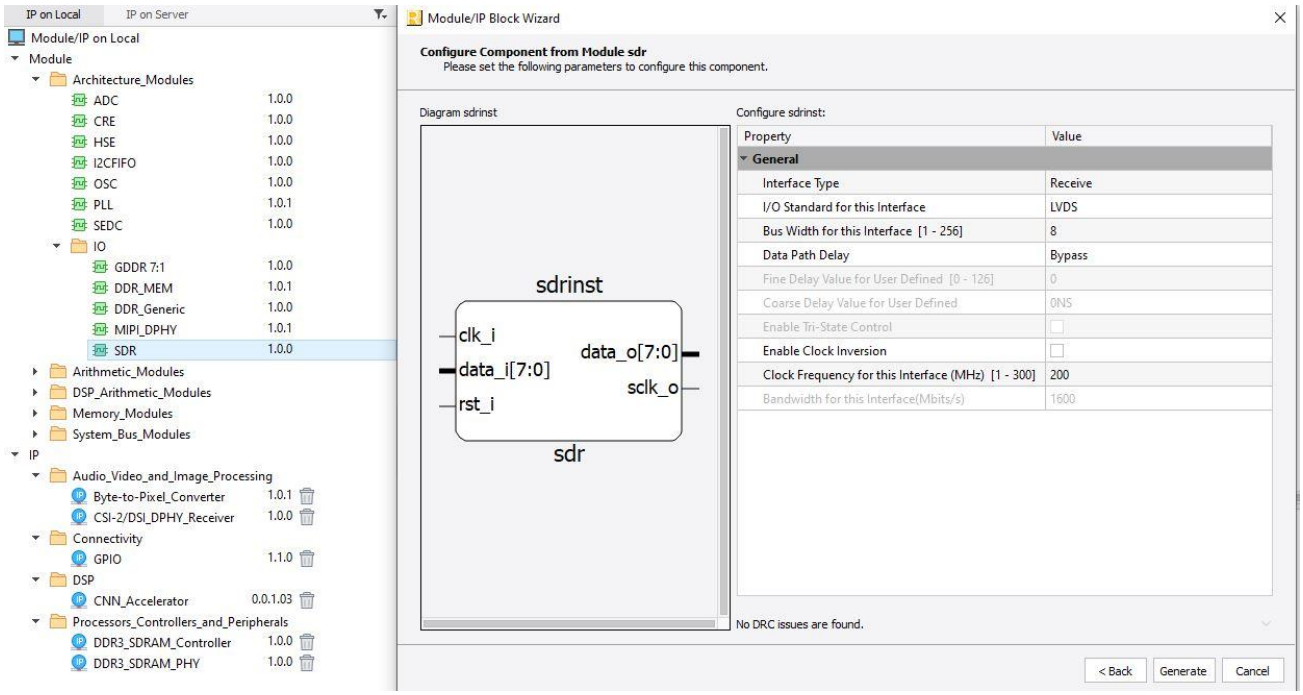


Figure 7.3. SDR Configuration Tab

Table 7.1 explains the various configurations options available for SDR modules.

Table 7.1. 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*	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**: 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.

7.3. 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 7.4 shows the type of interface selected as DDR_Generic and module name entered. This module can then be configured by clicking the Next button.

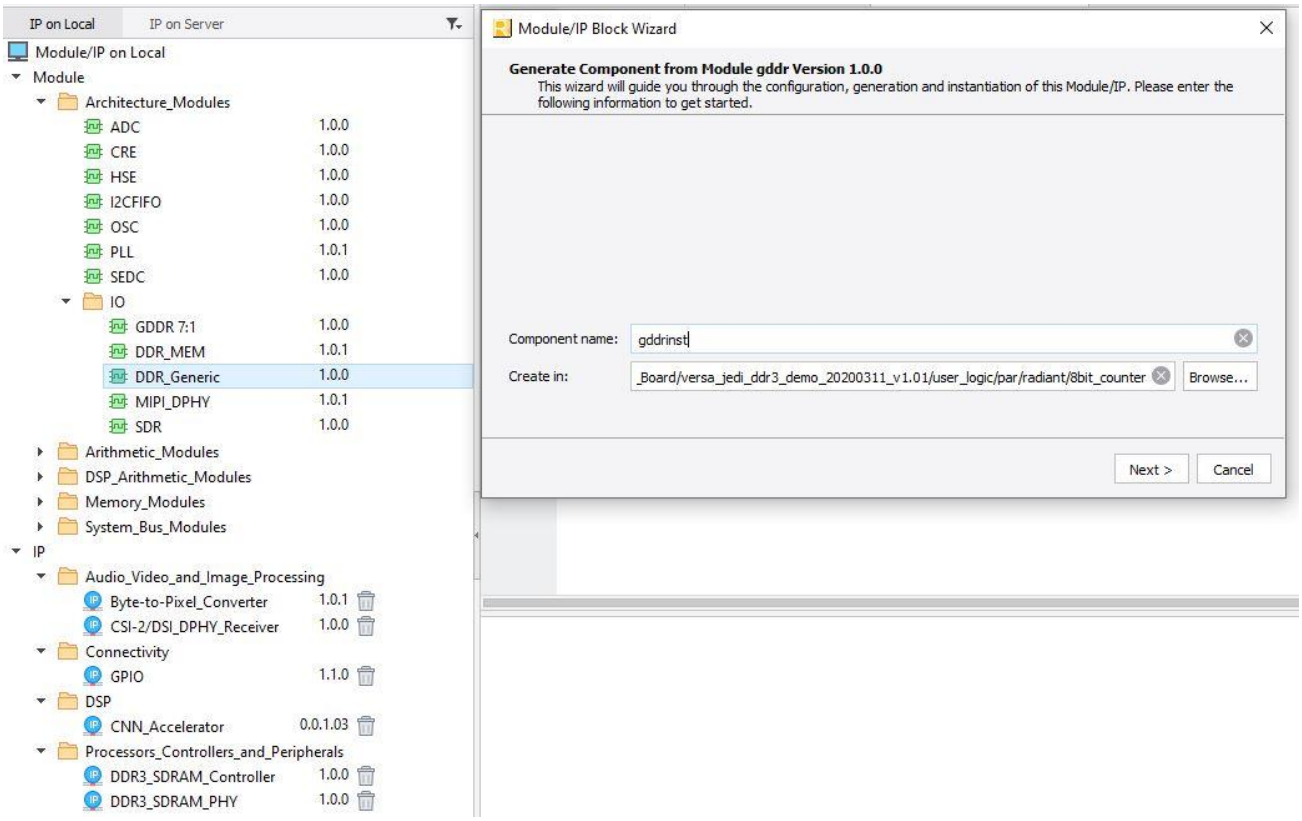


Figure 7.4. DDR_Generic Option Selected in the IP Catalog Tab

Figure 7.5 shows the Configuration tab.

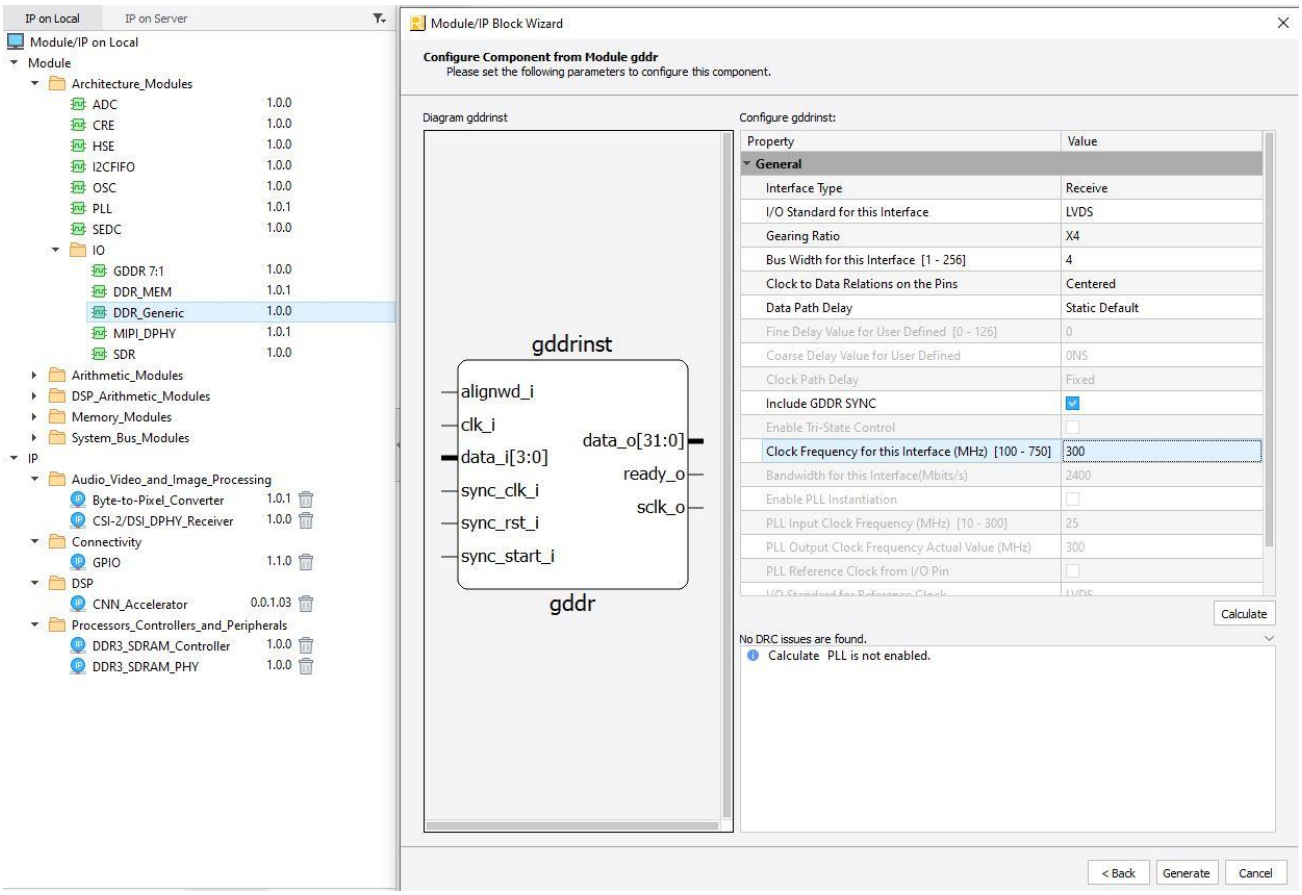


Figure 7.5. DDR_Generic Configuration Tab

Table 7.2 explains the various parameters in the Configuration tab.

Table 7.2. 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	X1,X2,X4,X5	X1
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 is 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 achived 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 7.3 shows how the interfaces are selected.

Table 7.3. IP Catalog DDR_Generic Interface Selection

Interface Type	Gearing Ratio	Alignment	Default Interface
Receive	2:1 (X1)	Edge-to-Edge	GDDRX1_RX.SCLK.Aligned
Receive	2:1 (X1)	Centered	GDDRX1_RX.SCLK.Centered
Receive	4:1 (X2)	Edge-to-Edge	GDDRX2_RX.ECLK.Aligned
Receive	4:1 (X2)	Centered	GDDRX2_RX.ECLK.Centered
Receive	8:1 (X4)	Edge-to-Edge	GDDRX4_RX.ECLK.Aligned
Receive	8:1 (X4)	Centered	GDDRX4_RX.ECLK.Centered
Receive	10:1 (X5)	Edge-to-Edge	GDDRX5_RX.ECLK.Aligned
Receive	10:1 (X5)	Centered	GDDRX5_RX.ECLK.Centered
Transmit	2:1 (X1)	Edge-to-Edge	GDDRX1_TX.SCLK.Aligned
Transmit	2:1 (X1)	Centered	GDDRX1_TX.SCLK.Centered
Transmit	4:1 (X2)	Edge-to-Edge	GDDRX2_TX.ECLK.Aligned
Transmit	4:1 (X2)	Centered	GDDRX2_TX.ECLK.Centered
Transmit	8:1 (X4)	Edge-to-Edge	GDDRX4_TX.ECLK.Aligned
Transmit	8:1 (X4)	Centered	GDDRX4_TX.ECLK.Centered
Transmit	10:1 (X5)	Edge-to-Edge	GDDRX5_TX.ECLK.Aligned
Transmit	10:1 (X5)	Centered	GDDRX5_TX.ECLK.Centered

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

7.4. 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 7.6 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.

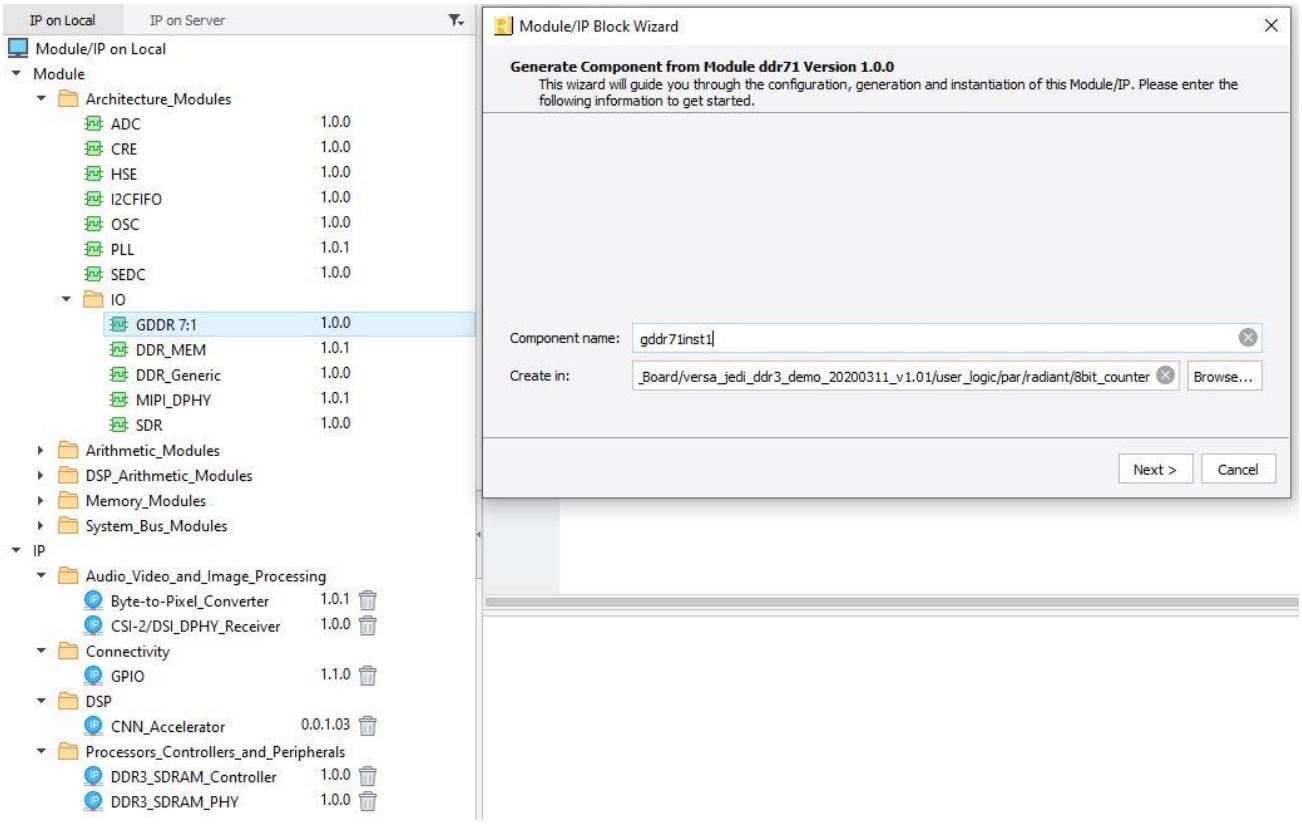


Figure 7.6. GDDR_7:1 Option Selected in the IP Catalog Tab

Clicking Customize displays the Configuration tab where the 7:1 LVDS interface can be configured. Figure 7.7 shows the Configuration tab for 7:1 LVDS interfaces.

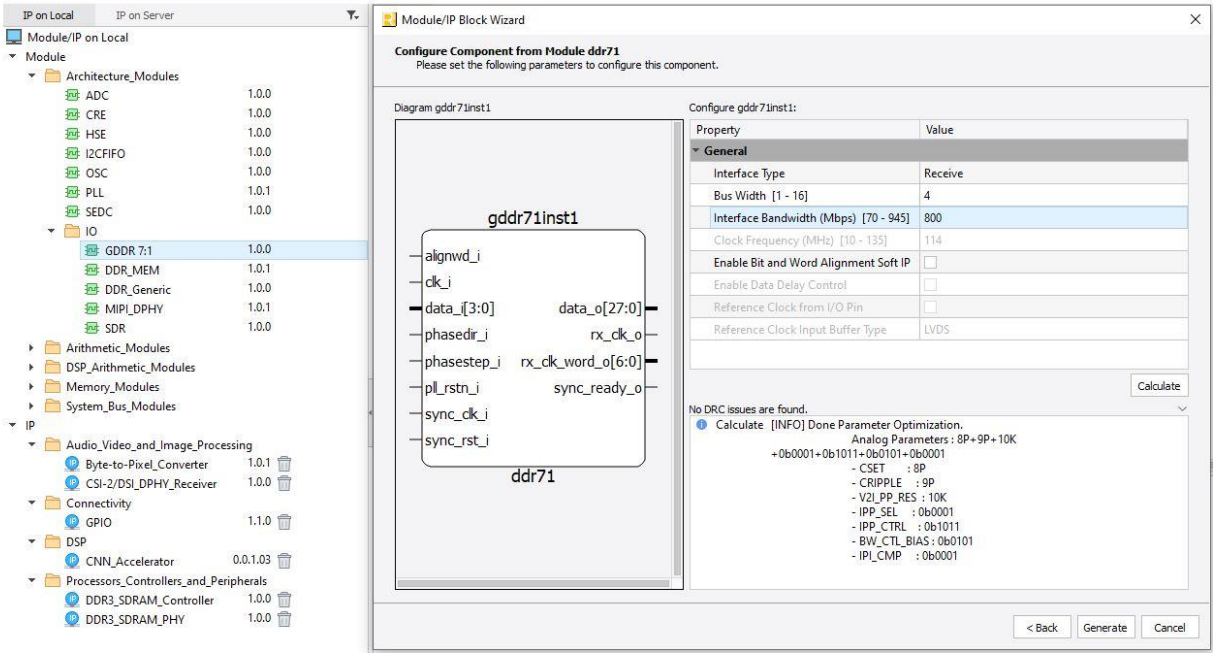


Figure 7.7. GDDR_7:1 LVDS Configuration Tab

Table 7.4 explains the various parameters for GDDR_7:1 configuration.

Table 7.4. 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)

7.5. Configuring DDR Memory Interfaces

IP Catalog is used to configure the PHY portion of the DDR3, DDR3L, LPDDR2, and LPDDR3 memory interfaces. For the detailed block diagram for each interface, see the Memory Interface Implementation section.

To build a DDR Memory interface, select DDR_MEM option under Architecture Modules – I/O in the IP Catalog tab. Enter the name of the module.

Figure 7.8 shows the type of interface selected as GDDR_MEM and module name entered. This module can then be configured by clicking the Next button.

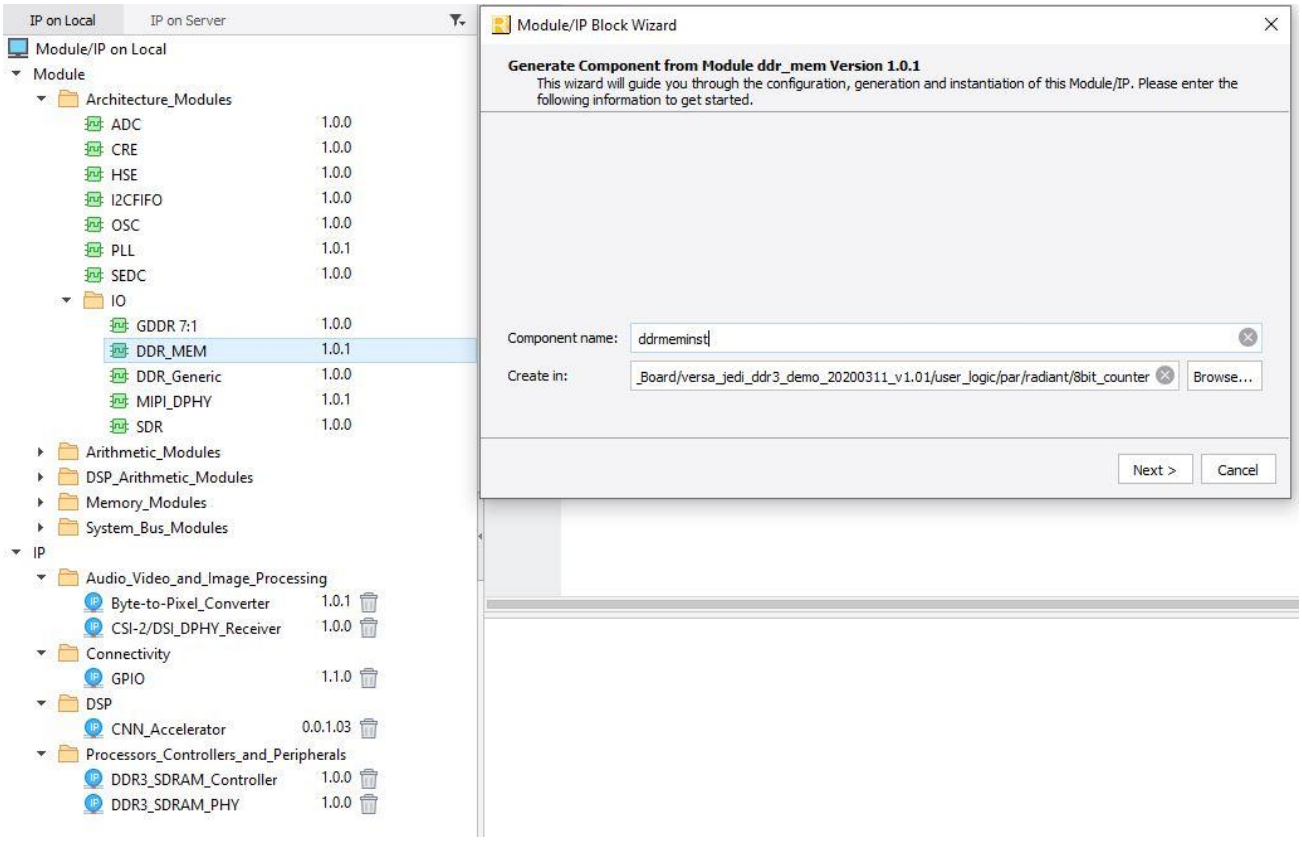


Figure 7.8. DDR_MEM Option Selected in the IP Catalog Tab

Figure 7.9 shows the General tab for the DDR_MEM interface.

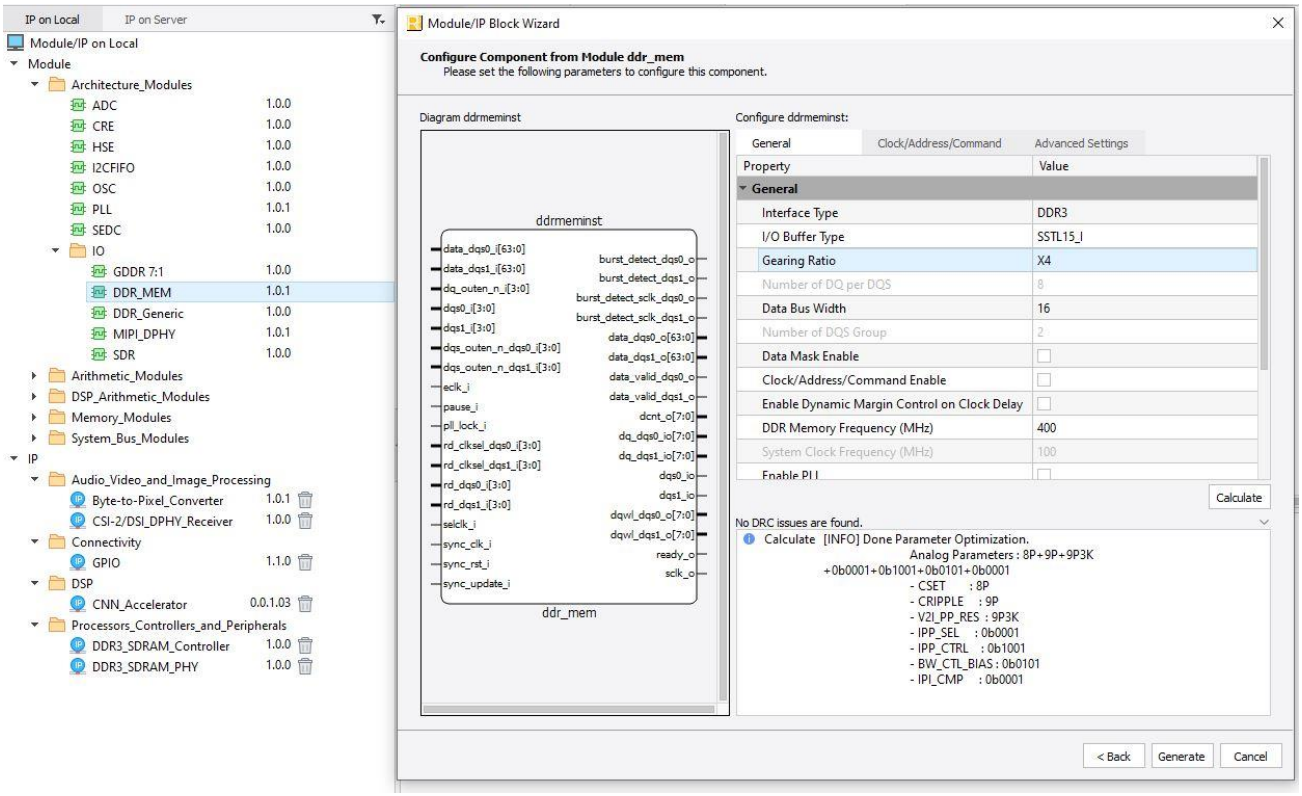


Figure 7.9. DDR_MEM Configuration

Table 7.5 below describes the various settings shown in the General tab.

Table 7.5. DDR_MEM General Tab Parameters

User Interface Option	Description	Values	Default
Interface	DDR memory interface type	DDR3, DDR3L, LPDDR2, LPDDR3	DDR3
I/O Buffer Configuration	I/O type configuration for DDR pins	DDR3: SSTL15_I, SSTL15_II DDR3L: SSTL135_I, SSTL135_II LPDDR2: HSUL12 LPDDR3: HSUL12	DDR3: SSTL15_I DDR3L: SSTL135_I LPDDR2: HSUL2 LPDDR3: HSUL12
Gearing Ratio	DDR Register gearing ratio	X2, X4	X2
Number of DQ per DQS	Number of associated DQ per DQS pin	8	8
Data Bus Width	DDR memory interface data bus width	DDR3/DDR3L: 8, 16, 24, 32 LPDDR2/LPDDR3: 16, 32	16
Number of DQS Group	Total number of DQS groups. Not user-selectable, display only.	1, 2, 3, 4	2
Data Mask Enable	Data mask pins added with this option checked	Enable, Disable	Disable
Clock/Address/Command Enable	Clock/address/command pins added with this option checked	Enable, Disable	Disable
Enable Dynamic Margin Control on the Clock Delay	Dynamic margin control ports added with this option checked	Enable, Disable	Disable
DDR Memory Frequency (MHz)	Target DDR memory interface frequency	DDR3: 400, 533 DDR3L: 400, 533 LPDDR2: 400, 533 LPDDR3: 400, 533	DDR3: 400 MHz DDR3L: 400 MHz LPDDR2: 400 MHz LPDDR3: 400 MHz
System Clock Frequency (Calculated) (MHz)	Calculated system clock frequency. Not user-selectable, display only.	SCLK Frequency Value, DDR Memory Frequency/2	200
Enable PLL	PLL included with this option checked	Enable, Disable	DIisabled
PLL Input Clock (CLK1) Frequency (MHz)	Input reference clock frequency	10 MHz – 800 MHz	100
PLL Reference Clock from I/O Pin	—	Enable, Disable	Disabled
I/O Standard for Reference Clock	—	All I/O standard	LVDS
PLL Out Clock (CLKOP) Tolerance (%)	—	—	0.0
Actual DDR Memory Frequency (MHz)	Calculated actual memory bus frequency. Not user-selectable, display only.	—	—

If you choose to generate the Clock/Address/Command signals, then the settings in the Clock/Address/Command tab are active and can be set up as required.

Figure 7.10 shows the Clock/Address/Command tab.

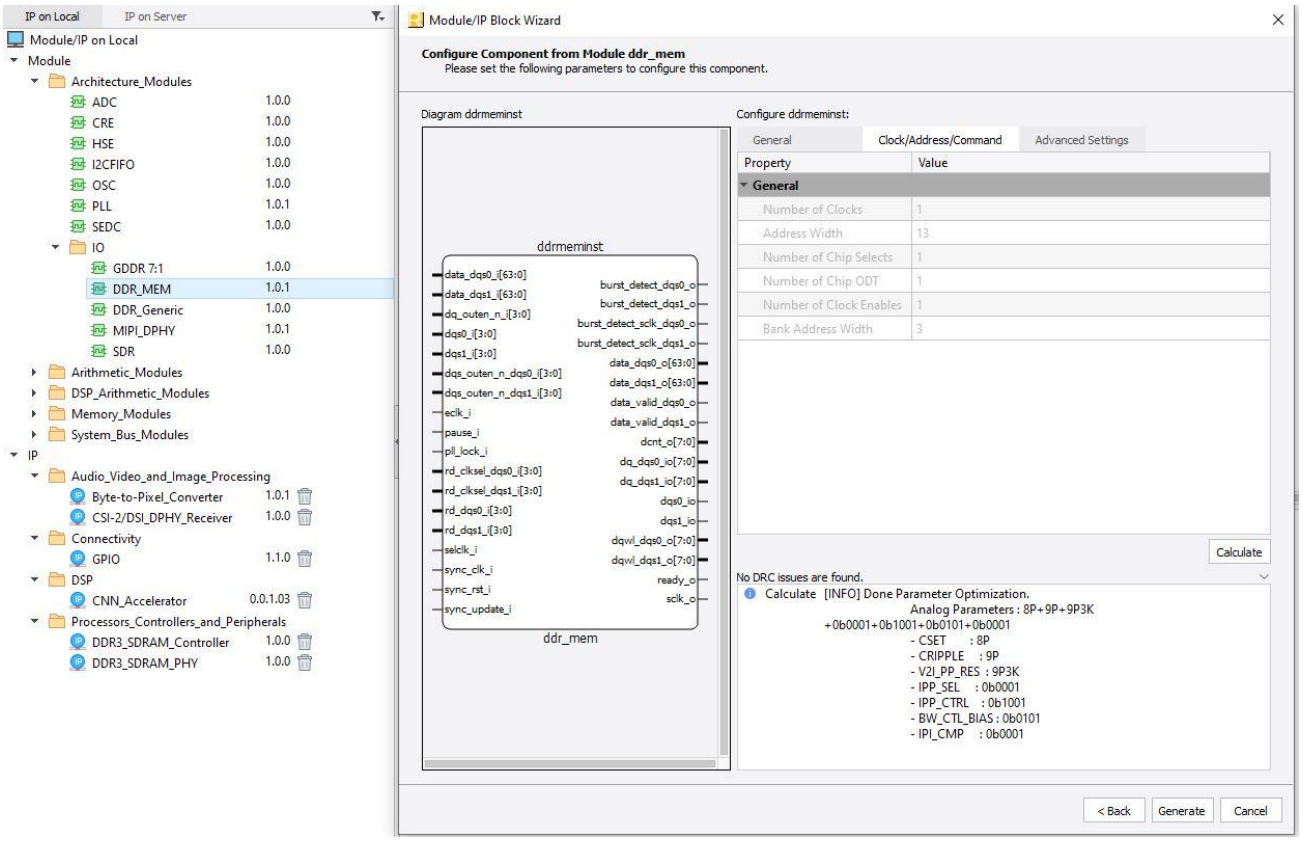


Figure 7.10. DDR_MEM Clock/Address/Command Tab

Table 7.6 lists the values that can be used for the Clock/Address/Command settings.

Table 7.6. DDR_MEM Clock/Address/Command Parameters

User Interface Option	Range	Default Value
Number of Clocks	DDR3/DDR3L: 1, 2, 4 LPDDR2/LPDDR3: 1	DDR3/DDR3L: 1 LPDDR2/LPDDR3: 1
Address Width	DDR3/DDR3L: 13 – 16 LPDDR2/LPDDR3: Blank	DDR3/DDR3L: 13 LPDDR2/LPDDR3: Blank
Bank Address Width	DDR3/DDR3L: 3 DDR3L: 3 LPDDR2/LPDDR3: Blank	DDR3/DDR3L: 3 LPDDR2/LPDDR3: Blank
Number of Chip Selects	DDR3/DDR3L: 1, 2, 4 LPDDR2/LPDDR3: 1	DDR3/DDR3L: 1 LPDDR2/LPDDR3: 1
Number of Clock Enables	= Number of Chip Selects	= Number of Chip Selects
Number of ODT	DDR3, DDR3L = Number of Chip Selects LPDDR2: Blank LPDDR3 = Number of Chip Selects	DDR3, DDR3L = Number of Chip Selects LPDDR2: Blank LPDDR3 = Number of chip Selects

There is an additional tab called Advanced Settings for the CrossLink-NX device that can be used to adjust the default DQS Read and Write Delay settings.

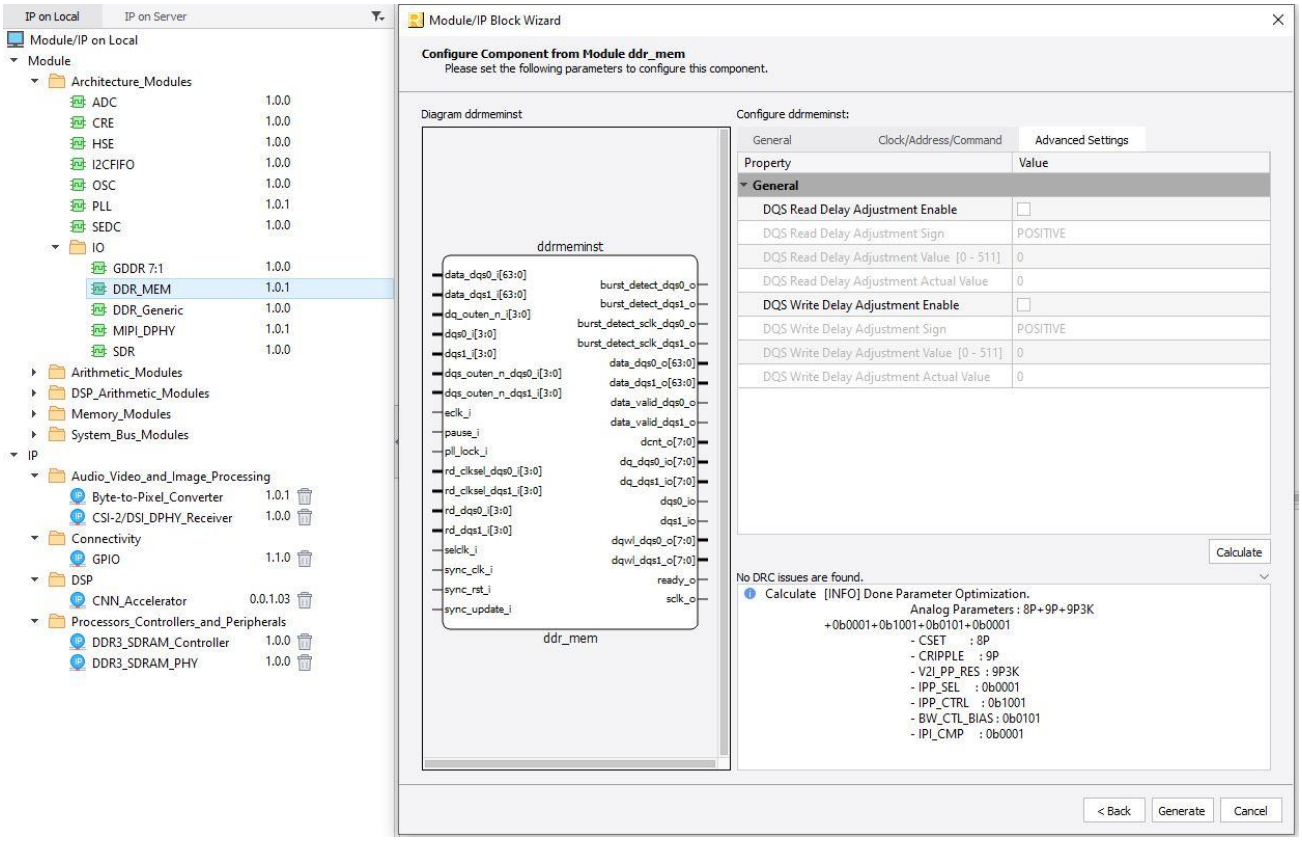


Figure 7.11. DDR_MEM Advanced Settings Tab

Figure 7.7 shows the available values in this tab.

Table 7.7. DDR_MEM Advanced Settings Tab Parameters

User Interface Option	Range	Default Value
DQS Read Delay Adjustment Enable	Enable, Disable	Disable
DQS Read Delay Adjustment Sign	Positive, Complement	Positive
DQS Read Delay Adjustment Value	0 – 511	0
DQS Write Delay Adjustment Enable	Enable, Disable	Disable
DQS Write Delay Adjustment Sign	Positive, Complement	Positive
DQS Write Delay Adjustment Value	0 – 511	0
DQS Write Delay Adjustment Actual Value	Display Value	0

7.6. 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 7.12 shows the type of interface selected as MIPI_DPHY and module name entered. This module can then be configured by clicking the Next button.

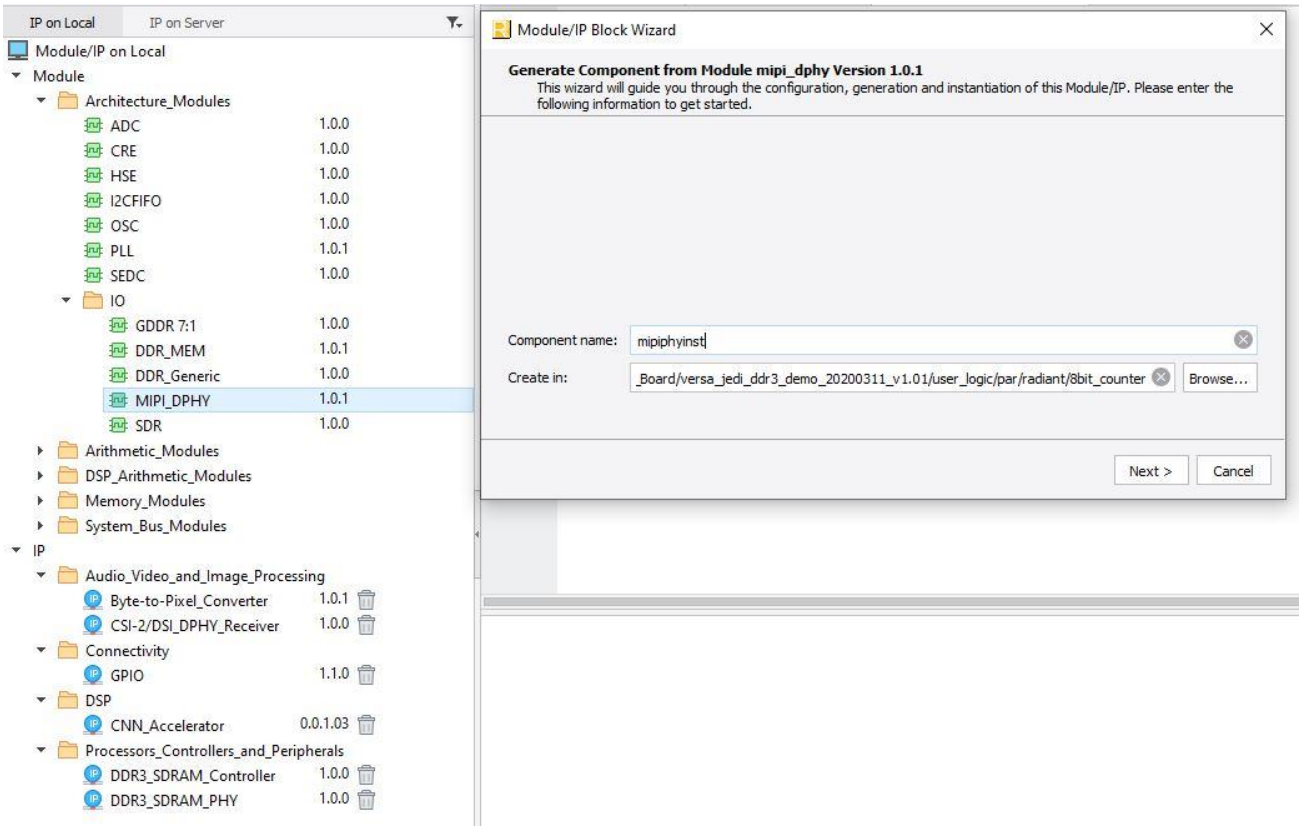


Figure 7.12. MIPI_DPHY Option Selected in the IP Catalog Tab

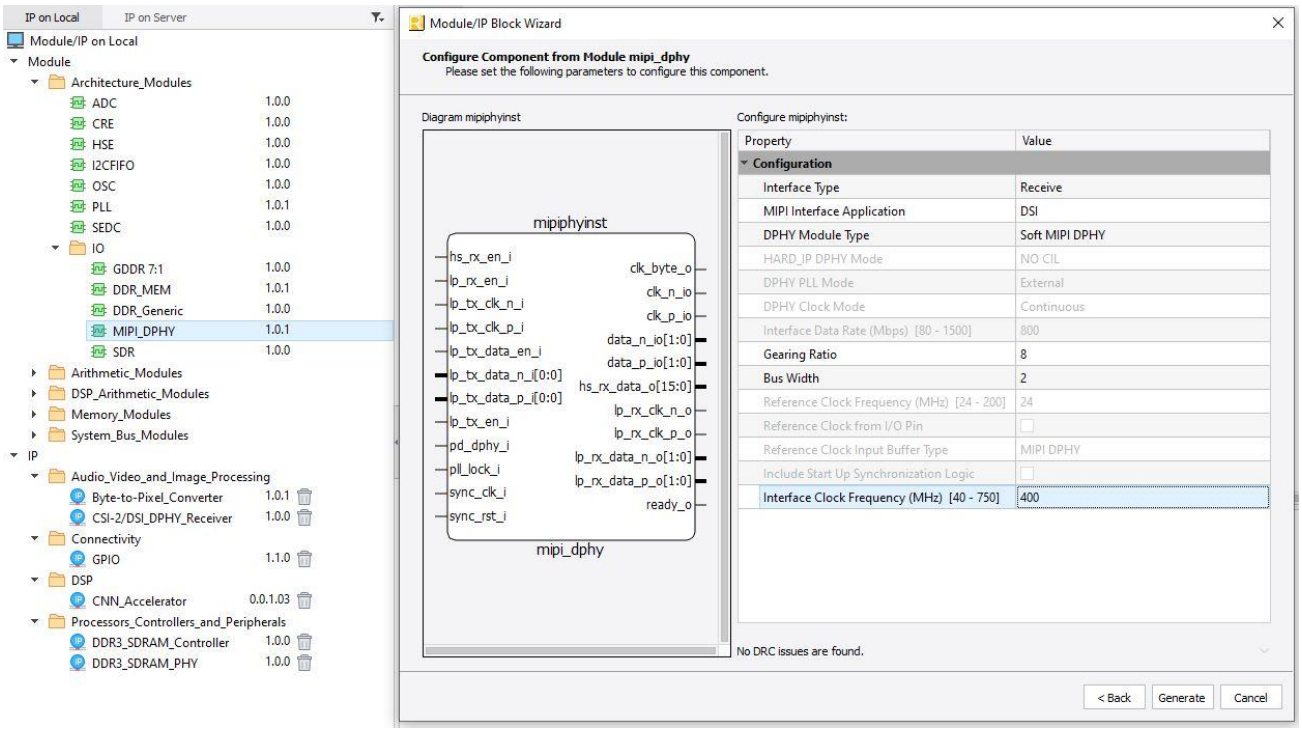


Figure 7.13. MIPI_DPHY Configuration Settings Tab

Table 7.8 explains the various configurations options available for MIPI D-PHY Interface.

Table 7.8. 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, Hard MIPI DPHY	Soft
Hard IP DPHY Mode	NO CIL, CIL	No CIL
DPHY PLL Mode	External, Internal for Transmit Interface	External
DPHY Clock Mode	Continuous, Non-Continuous for Hard MIPI DPY	Continuous
Interface Data Rate (Mbps)	Calculated display value: Interface clk frequency x 2, 80 -2500 Mbps	1500
Gearing Ratio	Soft MIPI DPHY : 8 Hard MIPI DPHY: 8, 16	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 Hard MIPI DPHY: 40 – 1250	750

8. 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. The DQSBUF primitives are used to generate the signals required to correctly capture the data from the DDR memory.

Table 8.1. 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 X1 IDDR
	IDDRX2	Generic X2 IDDR
	IDDRX4	Generic X4 IDDR
	IDDRX5	Generic X5 IDDR
	IDDR71	7:1 LVDS IDDR
DDR Data Output	ODDRX1	Generic X1 ODDR
	ODDRX2	Generic X2 ODDR
	ODDRX4	Generic X4 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 tristate control for DDR3 memory
	TSHX4DQ	DQ tristate 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 tristate control for DDR3 memory
	TSHX4DQS	DQS tristate control for DDR3 memory

8.1. 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 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 given 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
- Can be dynamically updated using counter up and down controls

You can optionally bypass the DELAY block completely as well.

8.2. 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.

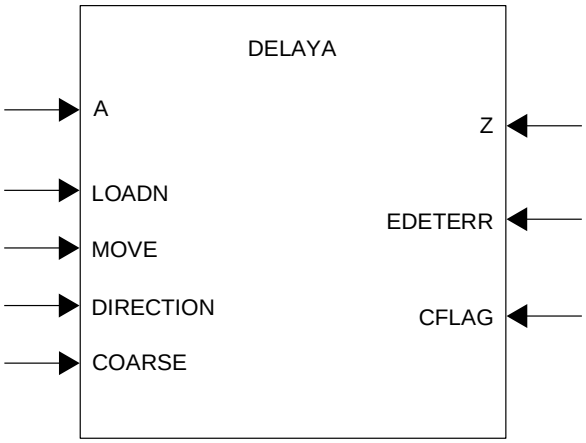


Figure 8.1. DELAYA Primitive

Table 8.2. 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

8.3. 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.

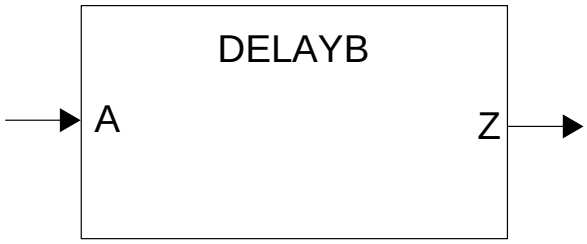


Figure 8.2. DELAYB Primitive

Table 8.3. 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

8.4. DELAY Attribute Description

Table 8.4 describes the attributes available for the DELAYA and DELAYB elements.

Table 8.4. 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 X2, X7to1, X4, or X5 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.

8.5. 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 DQSBUF elements to delay the DQS input, or in the DLLDEL module to delay the input clock. DDRDLL, by default, generates 90-degree phase shift.

8.5.1. DDRDLLA

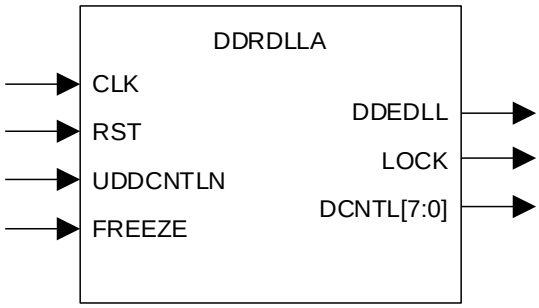


Figure 8.3. DDRDLLA Primitive

Table 8.5. 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 DQSBUF or 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 8.6. DDRDLL Attributes

Attribute	Description	Values	Default
FORCE_MAX_DELAY*	Bypass DLL locking procedure at low frequency	YES, NO	NO

***Note**: 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.

8.6. 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.

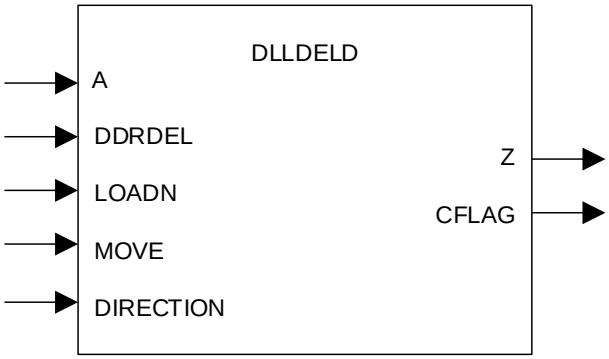


Figure 8.4. DLLDELD Primitive

Table 8.7. 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 8.8. DLLDELD Attributes

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.

8.7. Input DDR Primitives

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

8.7.1. IDDRX1

This primitive is used for the Generic x1 IDDR implementation.

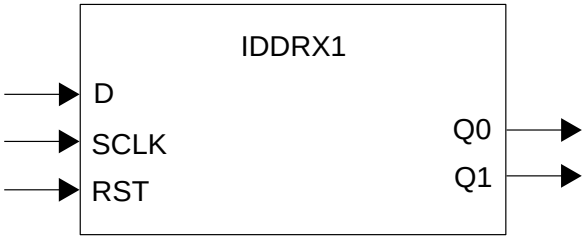


Figure 8.5. IDDRX1 Primitive

Table 8.9. 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

8.7.2. IDDRX2

This primitive is used for the Generic x2 IDDR implementation.

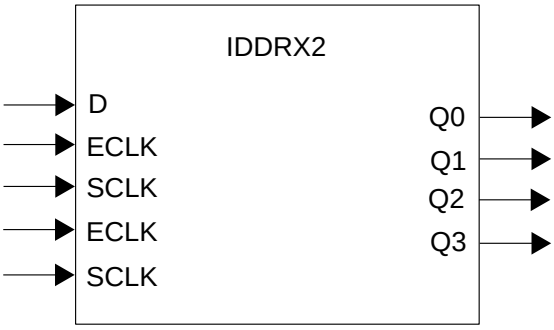


Figure 8.6. IDDRX2 Primitive

Table 8.10. 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.

8.7.3. IDDRX4

This primitive is used for the Generic x4 IDDR implementation.

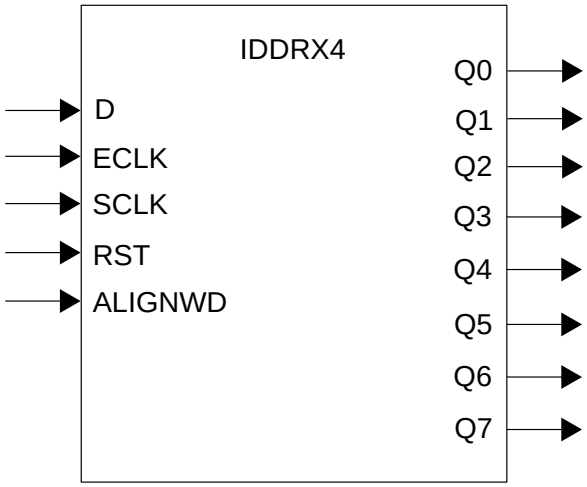


Figure 8.7. IDDRX4 Primitive

Table 8.11. 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.

8.7.4. IDDRX5

This primitive is used for the Generic x5 IDDR implementation.

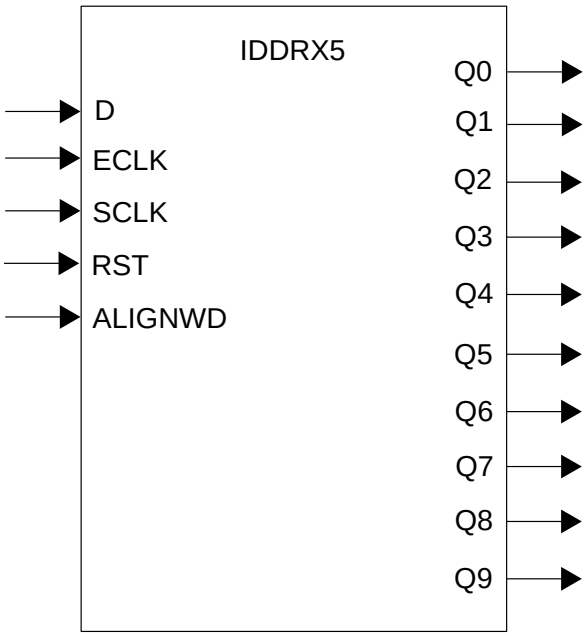


Figure 8.8. IDDRX5 Primitive

Table 8.12. 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.

8.7.5. IDDR71

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

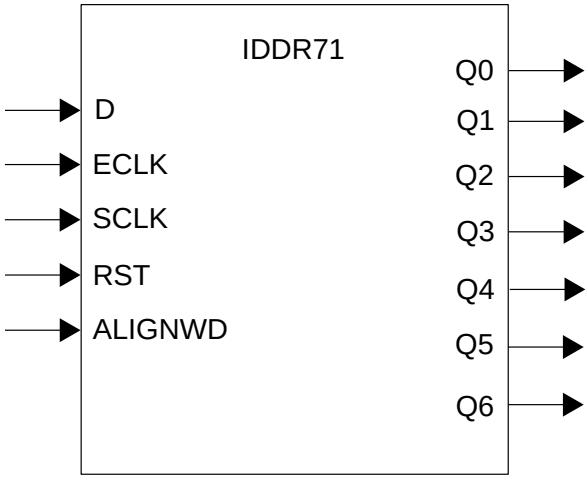


Figure 8.9. IDDR71

Table 8.13. 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.

8.8. Output DDR Primitives

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

8.8.1. ODDRX1

This primitive is used for Generic x1 ODDR implementation.

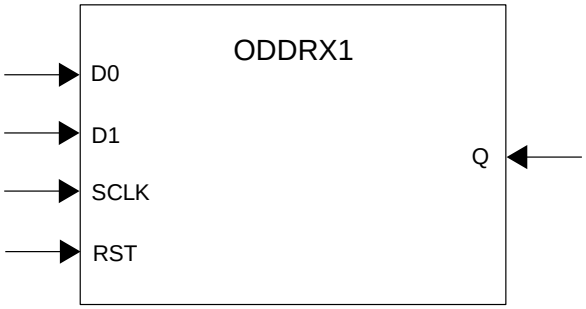


Figure 8.10. ODDRX1

Table 8.14. 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

8.8.2. ODDRX2

This primitive is used to receive Generic x2 ODDR implementation.

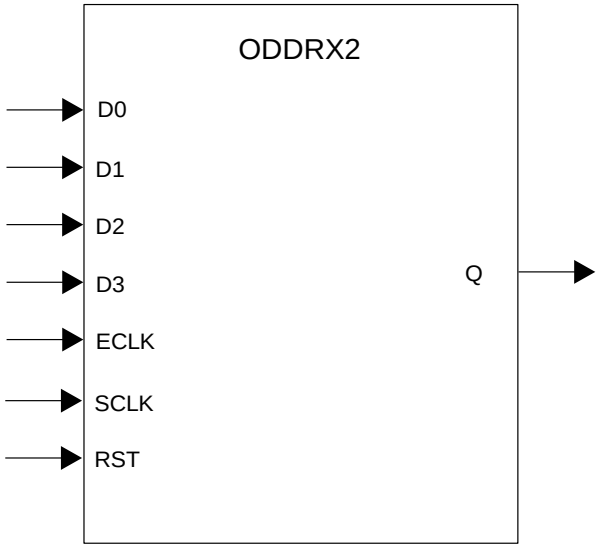


Figure 8.11. ODDRX2

Table 8.15. 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

8.8.3. ODDRX4

This primitive is used for the Generic x4 ODDR implementation.

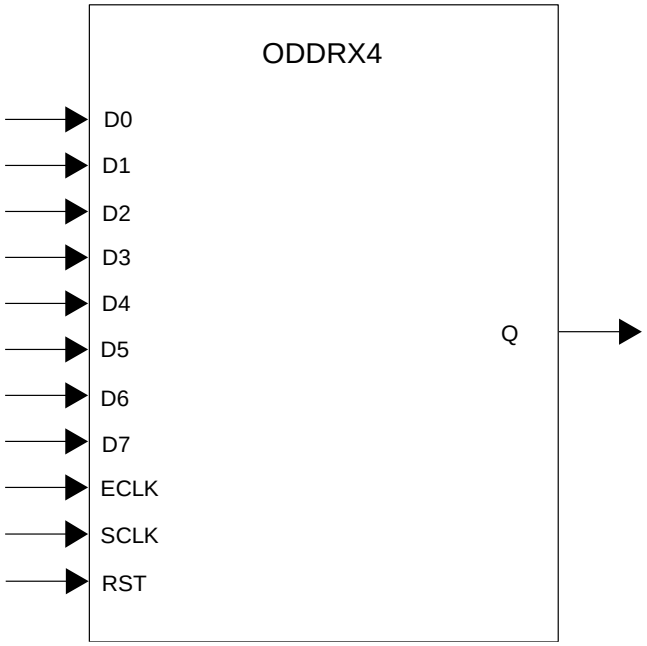


Figure 8.12. ODDRX4 Primitive

Table 8.16. 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

8.8.4. ODDRX5

This primitive is used for the Generic x5 ODDR implementation.

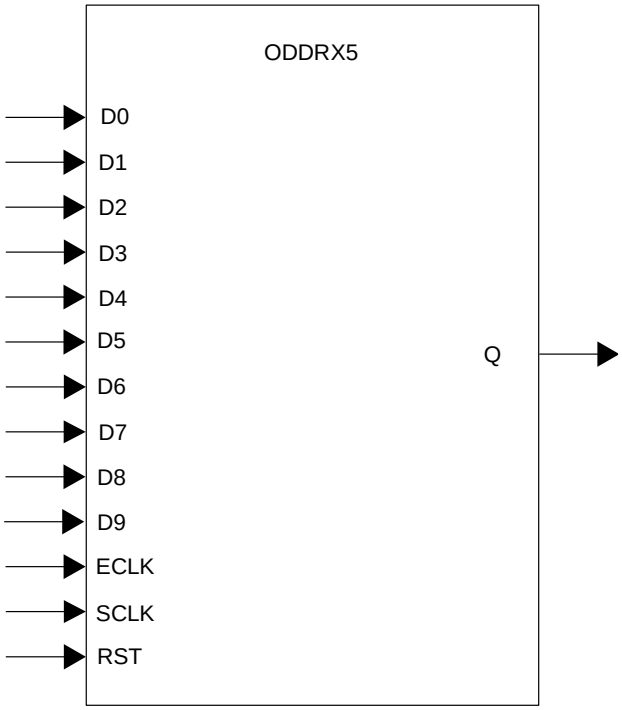


Figure 8.13. ODDRX5 Primitive

Table 8.17. 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

8.8.5. ODDR71

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

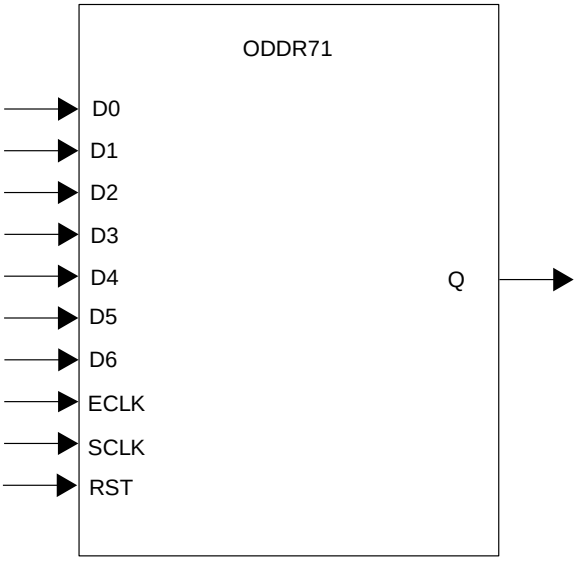


Figure 8.14. ODDR71 Primitive

Table 8.18. 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

8.8.6. DQSBUF (DQS Strobe Control Block)

DQSBUF block is used to delay the incoming DQS signal by 90 degrees. DQSBUF receives a delay code from DDRDLL and shifts the signal accordingly. There is one DQSBUF block for every 16 I/O. The DQSBUF should be used when the DQS clock tree is used for clocking the IDDR module.

The following describes the functions of the DQSBUF module:

- Receives the delay code from the DDRDLL and generates the 90-degree delayed DQS signal that is used as a Write clock in the IDDR module
- You can choose to move the delay up or down using the dynamic margin control signals (loadn, move, and direction).
- When this margin control circuit is used and LOADN goes high, any further delay code changes from the DDRDLL are not reflected in the delay DQS signal. A soft IP is required to detect the code changes from the DDRDLL and update the MOVE pulse input to the DQSBUF so that the DDRDLL code changes can be tracked.
- If margin control is not used, then LOADN should be low to continuously get code from DDRDLL.
- Pause should be asserted prior to changing readclksel, DYNDEL<>, or DLL code update.
- Receives READ clock select signals that are used to correctly position the READ signal with respect to the DQS preamble.
- Generates a BURSTDET output that can be used to validate the READ pulse positioning.
- Generates the Read and Write pointers required in the IDDR to correctly transfer data between the DQS and ECLK clock domains.

- Generates DQS write clocks to be used in ODDR modules to generate DQ and DQS.
- Generates the Write Leveling delay required for DDR3 or LPDDR3 interfaces.

8.9. IDDR/ODDR Modules for Memory DDR Implementation

This section describes the primitives used to build DDR, DDR2 and DDR3 memory interfaces. Some of these primitives are also used to generate generic DDR functions that use DQS clocking tree. CrossLink-NX IDDR/ODDR modules support 4:1(x2), 8:1(x4), gearing modes that are used to implement the memory functions. The DDR2 at higher speeds and DDR3 uses the x2 gearing primitives to implement the interfaces. The DDR3 interface at higher speeds requires the x4 gearing function. Each of these gearing functions is available on each pin when interfacing to the memory.

8.9.1. Primitive Symbols

Table 8.19 shows a summary of all the DDR memory primitives. See the sections below for detailed descriptions.

Table 8.19. Summary of all DDR Memory Primitives

DDR Memory	DQ Input	DQ Output	DQ Tristate	DQS Output	DQS Tristate	Addr/Cmd	Clock
DDR2/DDR3	IDDRX2DQ	ODDRX2DQ	TSHX2DQ	ODDRX2DQS	TSHX2DQS	OSHX2	ODDRX2
(x2 Gearing)	IDDRX4DQ	ODDRX4DQ	TSHX4DQ	ODDRX4DQS	TSHX4DQS	OSHX4	ODDRX4

8.9.2. IDDRX2DQ

This primitive is used to implement the DDR2 memory input interface at higher speeds and DDR3 memory interface.

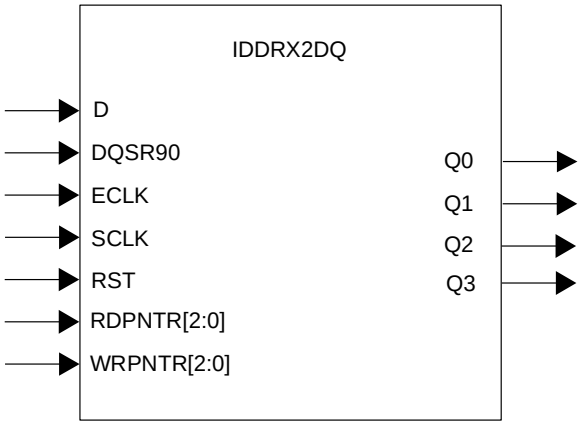


Figure 8.15. IDDRX2DQ Primitive

Table 8.20. IDDRX2DQ Port List

Port	I/O	Description
D	I	DDR data input
DQSR90	I	DQS clock input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RST	I	Reset to DDR registers
RDPNTR[2:0]	I	Read pointer from the DQSBUF module used to transfer data to ECLK
WRPNTR[2:0]	I	Write pointer from the DQSBUF module used to transfer data to ECLK
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.

8.9.3. IDDRX42DQ

This primitive is used to implement the DDR3 memory input interface at higher speeds.

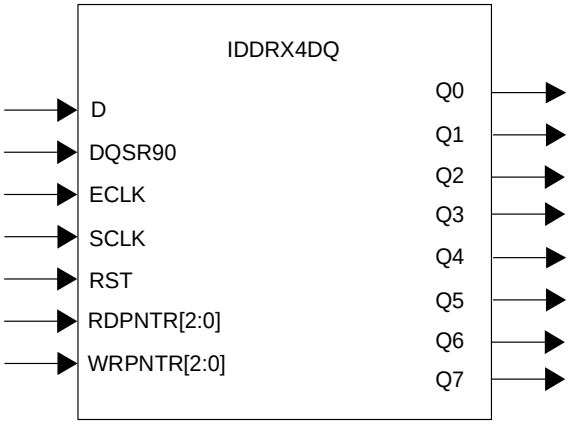


Figure 8.16. IDDRX4DQ Primitive

Table 8.21. IDDRX4DQ Port List

Port	I/O	Description
D	I	DDR data input
DQSR90	I	DQS clock input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-4 of ECLK)
RST	I	Reset to DDR registers
RDPNTR[2:0]	I	Read pointer from the DQSBUF module used to transfer data to ECLK
WRPNTR[2:0]	I	Write pointer from the DQSBUF module used to transfer data to ECLK
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.

8.9.4. ODDRX2DQ

This primitive is used to generate DQ data output for DDR2 with x2 gearing and for DDR3 memory interface.

Figure 8.17. ODDRX2DQ

Table 8.22. ODDRX2DQ Port List

Port	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR (D0 is output first, D3 last)
DQSW270	I	Clock that is 90 degrees ahead of clock used to generate the DQS output
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 DQSW270

8.9.5. ODDRX4DQ

This primitive is used to generate DQ data output for DDR3 memory interface using x4 gearing.

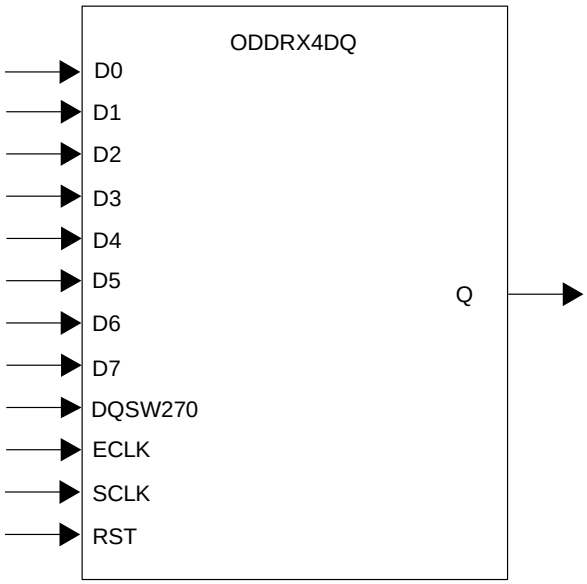


Figure 8.18. ODDRX4DQ Primitive

Table 8.23. ODDRX4DQ Port List

Port	I/O	Description
D0, D1, D2, D3,	I	Data input to the ODDR (D0 is output first, D7 last)
D4, D5, D6, D7	I	Clock that is 90 degrees ahead of DQSW used to generate DQ. DQSW includes write leveling phase shift from ECLK
DQSW270	I	Fast edge clock
ECLK	I	Primary clock input (divide-by-4 of ECLK)
SCLK	I	Reset to DDR registers
RST	O	DDR data output on both edges of DQSW270

8.10. Memory Output DDR Primitives for DQS Output

The following are the primitives used for DQS output.

8.10.1. ODDRX2DQS

This primitive is used to generate DQS clock output for DDR2 and DDR3 memory.

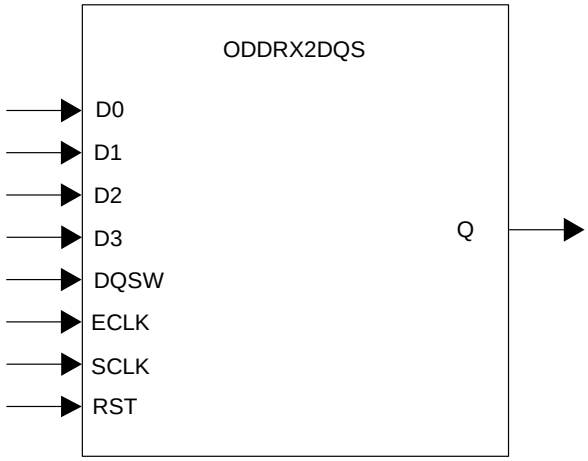


Figure 8.19. ODDRX2DQS Primitive

Table 8.24. ODDRX2DQS Port List

Port	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR (D0 is output first, D3 last)
DQSW	I	DQSW includes write leveling phase shift from ECLK
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 DQSW

8.10.2. ODDRX4DQS

This primitive is used to generate DQS clock output for DDR3 memory using x4 gearing.

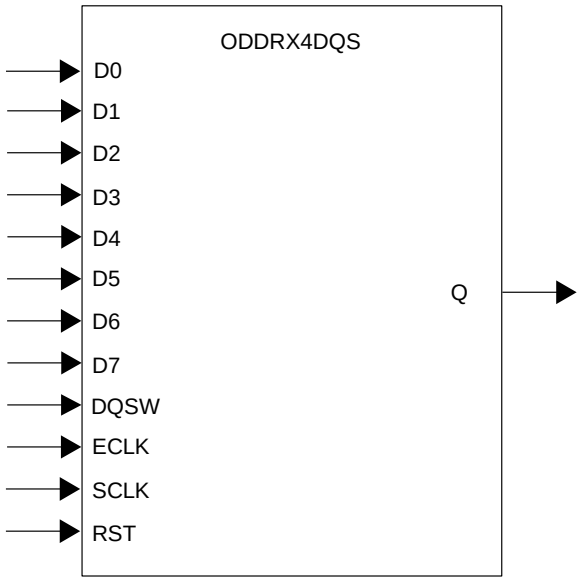


Figure 8.20. ODDRX4DQS Primitive

Table 8.25. ODDRX4DQS Port List

Port	I/O	Description
D0, D1, D2, D3, D4, D5, D6, D7	I	Data input to the ODDR (D0 is output first, D7 last)
DQSW	I	DQSW includes write leveling phase shift from ECLK
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 DQSW

8.11. Memory Output DDR Primitives for Tristate Output Control

The following are the primitives used to implement tristate control for the outputs to the DDR memory.

8.11.1. TSHX2DQ

This primitive is used to generate the tristate control for DQ data output for DDR2 memory with x2 gearing and DDR3 memory.

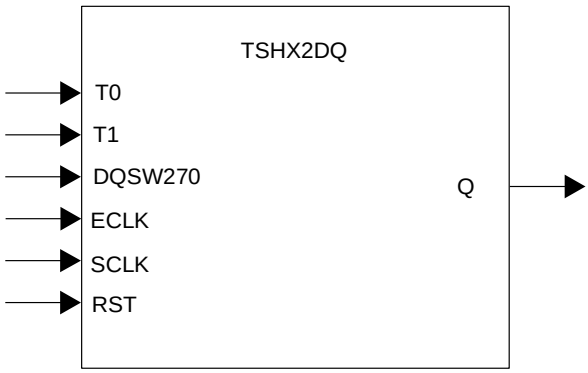


Figure 8.21. TSHX2DQ Primitive

Table 8.26. TSHX2DQ Port List

Port	I/O	Description
T0, T1	I	Tristate input (T0 is output first, followed by T1)
DQSW270	I	Clock that is 90 degrees ahead of the clock used to generate the DQS output
ECLK	I	ECLK input
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RST	I	Reset input
Q	O	Tristate output

8.11.2. TSHX4DQ

This primitive is used to generate the tristate control for DQ data output for DDR3 memory interface with x4 gearing.

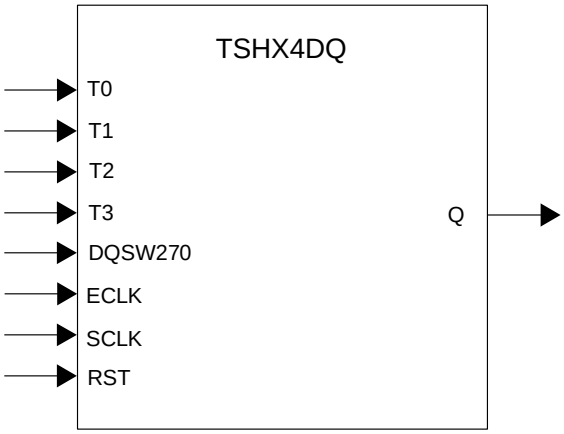


Figure 8.22. TSHX4DQ Primitive

Table 8.27. TSHX4DQ Port List

Port	I/O	Description
T0, T1, T2, T3	I	Tristate input (T0 is output first, followed by T1)
DQSW270	I	Clock that is 90 degrees ahead of the clock used to generate the DQS output
ECLK	I	ECLK input
SCLK	I	Primary clock input (divide-by-4 of ECLK)
RST	I	Reset input
Q	O	Tristate output

8.11.3. TSHX2DQS

This primitive is used to generate the tristate control for DQS output.

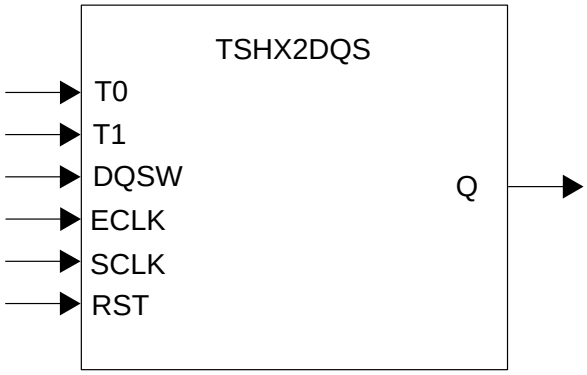


Figure 8.23. TSHX2DQS Primitive

Table 8.28. TSHX2DQS Port List

Port	I/O	Description
T0, T1	I	Tristate input (T0 is output first, followed by T1)
DQSW	I	DQSW includes write leveling phase shift from ECLK
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	Primary clock input
RST	I	Reset input
Q	O	Tristate output

8.11.4. TSHX4DQS

This primitive is used to generate the tristate control for DQS data output for DDR3 memory interface with x4 gearing.

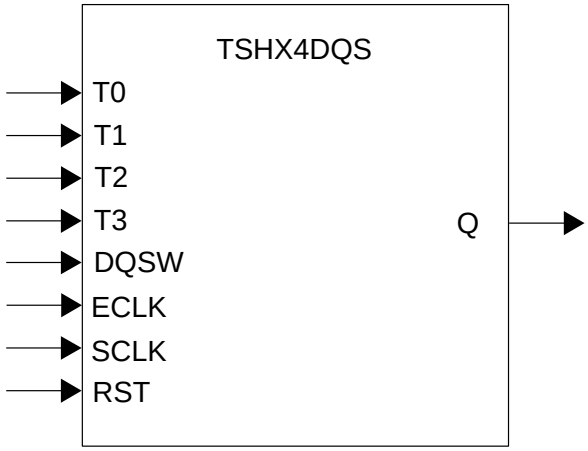


Figure 8.24. TSHX4DQS Primitive

Table 8.29. TSHX4DQS Port List

Port	I/O	Description
T0, T1, T2, T3	I	Tristate input (T0 is output first, followed by T3)
DQSW	I	DQSW includes write leveling phase shift from ECLK
ECLK	I	ECLK input (4x speed of SCLK)
SCLK	I	Primary clock input
RST	I	Reset input
Q	O	Tristate output

8.12. Memory Output DDR Primitives for Address and Command

The following are the primitives used to implement the address and command outputs to the DDR memory.

8.12.1. OSHX2

This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling.

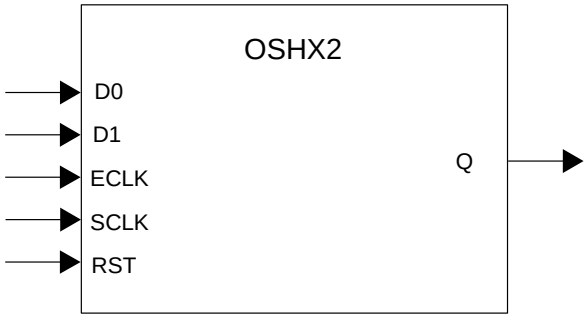


Figure 8.25. OSHX2 Primitive

Table 8.30. OSHX2 Port List

Port	I/O	Description
D0, D1	I	Data input (D0 is output first then D1)
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	Address and command output

8.12.2. OSHX4

This primitive is used to generate the address and command for DDR3 memory with x4 gearing and write leveling.

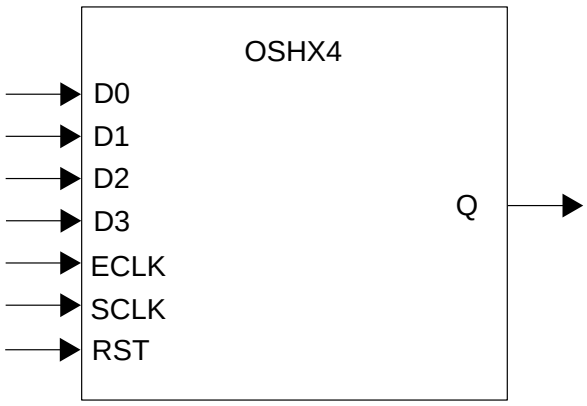


Figure 8.26. OSHX4 Primitive

Table 8.31. OSHX4 Port List

Port	I/O	Description
D0, D1, D2, D3	I	Data input (D0 is output first, D3 last)
ECLK	I	ECLK input (4x speed of SCLK)
SCLK	I	Primary clock input
RST	I	Reset input
Q	O	Address and command output

9. 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 9.1 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 9.1. 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
MEM_SYNC	Needed to avoid issues on DDR memory bus and update code in operation without interrupting interface operation.	Yes
7:1 LVDS Bit and Word Alignment (BW_ALIGN)	The soft IP is used to perform bit and word alignment using PLL's dynamic phase shift interface and aligned input of IDDR71C.	Optional
MIPI_FILTER	Implements low pass filter on low speed MIPI data	Optional

Table 9.2 below summarized the soft IPs used in each interface.

Table 9.2. Soft IP Used in Each Interface

Interface	Soft IP
GDDRX1_RX.SCLK.Centered	None
GDDRX1_RX.SCLK.Aligned	GDDRX71_TX.ECLK
GDDRX2_RX.ECLK.Centered	GDDR_SYNC
GDDRX2_RX.ECLK.Aligned	RX_SYNC
GDDRX2_RX.MIPI	GDDR_SYNC, MIPI_FILTER
GDDRX71_RX.ECLK	GDDR_SYNC, BW_ALIGN
GDDRX1_TX.SCLK.Centered	None
GDDRX1_TX.SCLK.Aligned	None
GDDRX2_TX.ECLK.Centered	GDDR_SYNC
GDDRX2_TX.ECLK.Aligned	GDDR_SYNC
GDDRX71_TX.ECLK	GDDR_S

9.1. Detailed Description of Each Soft IP

9.1.1. GDDR_SYNC

This module is needed to startup al RX Centered and all TX interfaces with 2x gearing.

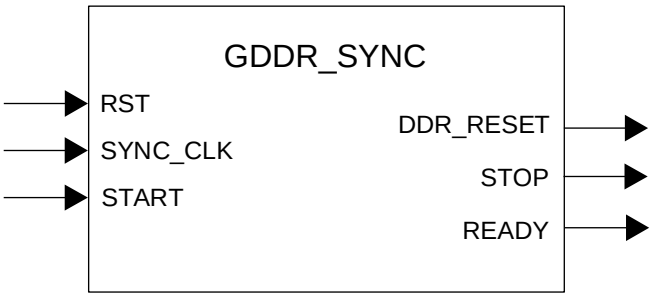


Figure 9.1. GDDR_SYNC Ports

Table 9.3. GDDR_SYNC Port List description

Port	I/O	Description
SYNC_CLK	I	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock
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	Connects 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

9.1.2. RX_SYNC

This module is needed to startup RX Aligned interfaces with 2X gearing.

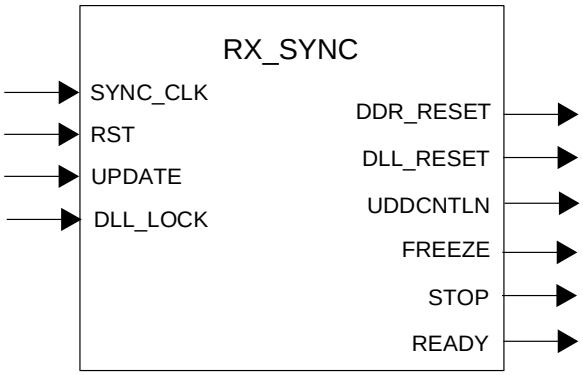


Figure 9.2. RX_SYNC Ports

Table 9.4. GDDR_SYNC Port List description

Port	I/O	Description
SYNC_CLK	I	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock.
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.

9.1.3. MEM_SYNC

This module is needed to startup external memory controller interfaces with 2X gearing.

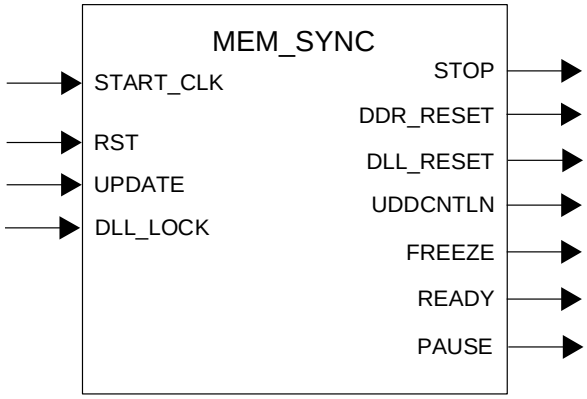


Figure 9.3. MEM_SYNC Ports

Table 9.5. MEM_SYNC Port Description

Port	I/O	Description
SYNC_CLK	I	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock.
RST	I	Active high reset to this sync circuit. When RST=1, STOP=0, FREEZE=0, UDDCNTLN=1, DLL_REEST=1, DDR_RESET=1, READY=0, PAUSE =0.
DLL_LOCK	I	LOCK output from DDRDLL
UPDATE	I	After ready goes high, you can use UPDATE to update code in DQSBUF, perform training (change read_clk_sel) or write leveling (change dyndelay<>).
PAUSE	O	Connect to DQSBUF.PAUSE
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, ODDRX, OSHX components, DQSBUF, and CLKDIV
READY	O	Indicate that startup is finished and RX circuit is ready to operate.

9.1.4. 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.

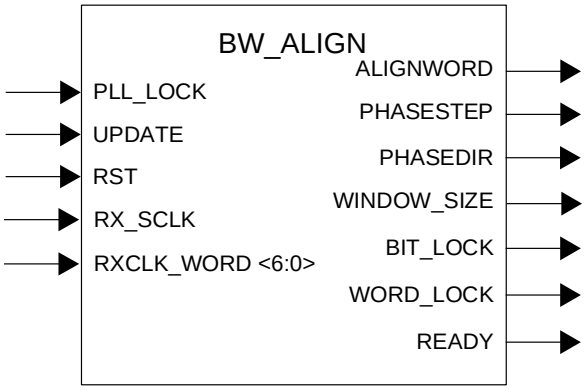


Figure 9.4. BW_ALIGN Ports

Table 9.6. 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's 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's 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.

9.1.5. 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.

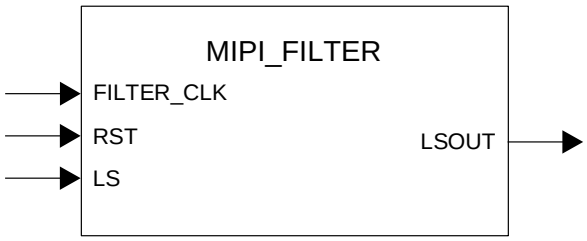


Figure 9.5. MIPI_FILTER Ports

Table 9.7. 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.

Technical Support Assistance

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

Revision History

Revision 1.3, July 2020

Section	Change Summary
High-Speed I/O Interface Building Blocks	• Remove *40K* since the 17K device is also supported. • Updated figure caption to Figure 3.2. CrossLink-NX SGMII CDR IP. • Added information regarding the 17K device PLLs in the PLL subsection.
Building Generic High Speed Interfaces	Updated subsection heading to Types of High-Speed I/O Interfaces.
All	Minor editorial changes.

Revision 1.2, April 2020

Section	Change Summary
High-Speed DDR Interface Details	• Updated information on the soft MIPI D-PHY RX implementation. • Added the Soft MIPI D-PHY Transmit Interfaces section. • Removed hard MIPI D-PHY sections and provided reference/link to related technical note.

Revision 1.1, March 2020

Section	Change Summary
High-Speed DDR Interface Details	• Updated GDDR_71 (both RX and TX) • Updated GDRRX5 (both RX and TX)
CrossLink-NX Memory Interfaces	Removed DDR2 memory.
Using IP Catalog to Build and Plan High Speed DDR Interfaces	Updated Lattice Radiant software user interface.
I/O Logic (DDR) User Primitives and Attributes	Removed DDR2 from Table 8.1. Software Primitives.
All	• Updated linked references to CrossLink-NX documents. • Minor adjusments on style/formatting.

Revision 1.0, November 2019

Section	Change Summary
All	Initial release

www.latticesemi.com