MachXO4 Implementing High-Speed I/O Interface

Technical Note

FPGA-TN-02410-1.0

December 2025

Disclaimers

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

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Abbreviations in This Document 7
1. Introduction 8
2. Architecture for High-Speed Interfaces 9
2.1. Gearing Logic Distribution 9
2.2. Different Types of I/O Logic Cells 9
2.3. Clock Domain Transfer at PIO Cells 12
3. External High-Speed Interface Description 17
4. High-Speed Interface Building Blocks 18
4.1. ECLK 18
4.2. ECLKSYNC 18
4.3. CLKDIV 18
4.4. SCLK 18
4.5. PLL 18
4.6. DQSDLL 18
4.7. Input DDR (IDDR) 18
4.8. Output DDR (ODDR) 19
4.9. Delays 19
5. Generic High-Speed DDR Interfaces 20
5.1. High-Speed GDDR Interface Types 20
5.2. High-Speed GDDR Interface Details 21
5.2.1. Receive Interfaces 21
5.2.2. Transmit Interfaces 28
6. Using IP Catalog to Build High-Speed DDR Interfaces 36
6.1. Configuring the SDR Modules 36
6.2. Configuring DDR Generic Modules 40
6.3. Configuring Generic DDR 7:1 Modules 44
7. Generic High-Speed DDR Design Guidelines 46
7.1. I/O Logic Cells and Gearing Logic 46
7.2. High-Speed ECLK Bridge 46
7.2.1. Reset Synchronization Requirement 46
8. DDR Software Primitives and Attributes 48
8.1. Input DDR Primitives 48
8.1.1. IDDRXE 48
8.1.2. IDDRX2E 49
8.1.3. IDDRX4B 49
8.1.4. IDDRX71A 50
8.2. Output DDR Primitives 51
8.2.1. ODDRXE 51
8.2.2. ODDRX2E 52
8.2.3. ODDRX4B 52
8.2.4. ODDRX71A 53
8.3. DDR Control Logic Primitives 54
8.3.1. DQSDLLC 54
8.3.2. DELAYH 55
8.3.3. DELAYG 56
8.3.4. DLLDELC 57
References 58

Technical Support Assistance 59
Revision History 60

Figures

Figure 2.1. Basic PIO Cell Support ×1 Gearing Ratio 10
Figure 2.2. Video PIO Cell for ×2/×4 and 7:1 Applications in MachXO4 12
Figure 2.3. 7:1 Deserializer Timing in MachXO4 Device 13
Figure 2.4. ×4 Deserializer Timing (Even Phases, SEL=0) 13
Figure 2.5. ×4 Deserializer Timing (Odd Phases, SEL=1) 14
Figure 2.6. ×2 Deserializer Timing (Even Phases, SEL=0) 14
Figure 2.7. ×2 Deserializer Timing (Odd Phases, SEL=1) 14
Figure 2.8. 7:1 Deserializer Timing in Response to ALIGNWD in MachXO4 Device 15
Figure 2.9. ×4 Deserializer Timing in Response to ALIGNWD (Even-to-Odd Phase) 15
Figure 2.10. ×4 Deserializer Timing in Response to ALIGNWD (Odd-to-Even Phase) 16
Figure 3.1. External Interface Definition 17
Figure 5.1. GIREG_RX Interface 21
Figure 5.2. GDDRX1_RX.SCLK.Aligned Interface Using DQSDLL 22
Figure 5.3. GDDRX1_RX.SCLK.Centered 22
Figure 5.4. GDDRX2_RX.ECLK.Aligned Interface 23
Figure 5.5. GDDRX2_RX.ECLK.Centered Interface 24
Figure 5.6. GDDRX4_RX.ECLK.Aligned Interface 25
Figure 5.7. GDDRX4_RX.ECLK.Centered Interface 26
Figure 5.8. GDDR71_RX.ECLK.7:1 Interface 27
Figure 5.9. GOREG_TX.SCLK Interface 28
Figure 5.10. GDDRX1_TX.SCLK.Aligned Interface 29
Figure 5.11. GDDRX1_TX.SCLK.Centered Interface 30
Figure 5.12. GDDRX2_TX.ECLK.Aligned Interface 31
Figure 5.13. GDDRX2_TX.ECLK.Centered Interface 32
Figure 5.14. GDDRX4_TX.ECLK.Aligned Interface 33
Figure 5.15. GDDRX4_TX.ECLK.Centered Interface 34
Figure 5.16. GDDR71_TX.ECLK.7:1 Interface 35
Figure 6.1. SDR Module Selection at the IP Catalog Main Window 36
Figure 6.2. SDR Module Customization in Module/IP Block Wizard 37
Figure 6.3. SDR Module Configuration in Module/IP Block Wizard 37
Figure 6.4. DDR_Generic Module Selection at the IP Catalog Main Window 40
Figure 6.5. DDR Generic Module Configuration in Module/IP Block Wizard 41
Figure 6.6. GDDR 7:1 Module Selection at the IP Catalog Main Window 44
Figure 6.7. Configuration Tab of GDDR 7:1 Module 45
Figure 7.1. Reset Synchronization for Receive Interfaces 47
Figure 7.2. Reset Synchronization for Transmit Interfaces 47
Figure 8.1. IDDRXE Symbol 48
Figure 8.2. IDDRX2E Symbol 49
Figure 8.3. IDDRX4B Symbol 49
Figure 8.4. IDDRX71A Symbol 50
Figure 8.5. ODDRXE Symbol 51
Figure 8.6. ODDRX2E Symbol 52
Figure 8.7. ODDRX4B Symbol 52
Figure 8.8. ODDRX71A Symbol 53
Figure 8.9. DQSDLLC Symbol 54
Figure 8.10. DELAYH Symbol 55
Figure 8.11. DELAYG Symbol 56
Figure 8.12. DLLDELC Symbol 57

Tables

Table 2.1. Gearing Logic Distribution for MachXO4 Devices 9
Table 5.1. Generic High-Speed I/O DDR Interfaces 20
Table 6.1. SDR Module General Attributes....38
Table 6.2. SDR Module Attributes Description 38
Table 6.3. DDR_Generic Module General Attributes....41
Table 6.4. GDDR 7:1 General Attributes 45
Table 7.1. Gearing Logic Supported by Mixed Mode of I/O Logic Cells for MachXO4 Devices 46
Table 8.1. MachXO4 DDR Software Primitives 48
Table 8.2. DQSDLLC Signals....54
Table 8.3. Attribute for DQSDLLC 55
Table 8.4. DELAYH Signals....55
Table 8.5. DELAYH Attributes 55
Table 8.6. DEL_MODE Values Corresponding to the GDDR Interface 56
Table 8.7. DELAYG Signals....56
Table 8.8. DLLDELC Signals 57

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
DDR	Double Data Rate
GDDR	Generic DDR
HP	High Performance
IDDR	Input DDR
IP	Intellectual Property
LP	Low Power
ODDR	Outdoor DDR
PIC	Programmable I/O Cell
PIO	Programmable I/O
PLD	Programmable Logic Device
RX	Receive
SDR	Single Data Rate
TX	Transmit

1. Introduction

The Lattice Semiconductor MachXO4™ devices support high-speed interfaces for both Double Data Rate (DDR) and Single Data Rate (SDR) applications through built-in Programmable I/O (PIO) logic. SDR captures data either the rising edge or the falling edge of a clock to transfer data, while DDR captures data on both the edges of the clock to transfer data, thus doubling the performance.

This document focuses on the implementation of high-speed generic DDR interfaces in the MachXO4 devices. It also provides guidelines for making use of the built-in capabilities of the MachXO4 devices to achieve the best performance for high-speed interfaces.

2. Architecture for High-Speed Interfaces

2.1. Gearing Logic Distribution

The high-speed generic DDR (GDDR) interfaces are supported through the built-in gearing logic in the Programmable I/O (PIO) cells. This gearing is necessary to support high-speed I/O while reducing the performance requirement on the FPGA fabric.

There are four gearing ratio settings available in the MachXO4 devices depending on the I/O bank locations and the logic density.

The ×1 gearing ratio is available in all banks for all the device densities. The ×2, ×4, and the 7:1 gearing ratio are available in the top and bottom banks of the MachXO4 devices. The 7:1 gearing ratio is mainly used for video display applications. The ×2/×4 gearing circuit is shared with the 7:1 circuit on both receive and the transmit sides. Table 2.1 gives a breakdown of gearing logic support in the different I/O banks. Details of PIO cells can be found in MachXO4 Family Data Sheet (FPGA-DS-02125).

Table 2.1. Gearing Logic Distribution for MachXO4 Devices

Gearing Logic	Definition	Gearing Ratio	Left	Right	Bottom	Top
DDR ×1[1]	GDDR	1:2 or 2:1	Yes	Yes	Yes	Yes
Input DDR ×2	GDDR	1:4	—	—	Yes	—
Input DDR ×4	GDDR	1:8	—	—	Yes	—
Input DDR 7:1	GDDR	1:7	—	—	Yes	—
Output DDR ×2	GDDR	4:1	—	—	—	Yes
Output DDR ×4	GDDR	8:1	—	—	—	Yes
Output DDR 7:1	GDDR	7:1	—	—	—	Yes

Note:

1. DDR ×1 is available for all MachXO4 device densities.

2.2. Different Types of I/O Logic Cells

To support various gearing ratios, the MachXO4 devices support two types of PIO logic cells. These include a basic PIO cell and a video PIO cell.

The basic PIO cell supports traditional SDR registers and DDR ×1 registers. It is available on all sides of all MachXO4 devices. The video PIO cell supports the ×2, ×4 (in MachXO4 devices) and 7:1 gearing applications. They are available on the bottom side for the receive interfaces, and on the top side for the transmit interfaces. The input and output structures of each type of PIO cell are discussed in detail in MachXO4 Family Data Sheet (FPGA-DS-02125). The block diagrams of the PIO cells are shown here again in this document for reference.

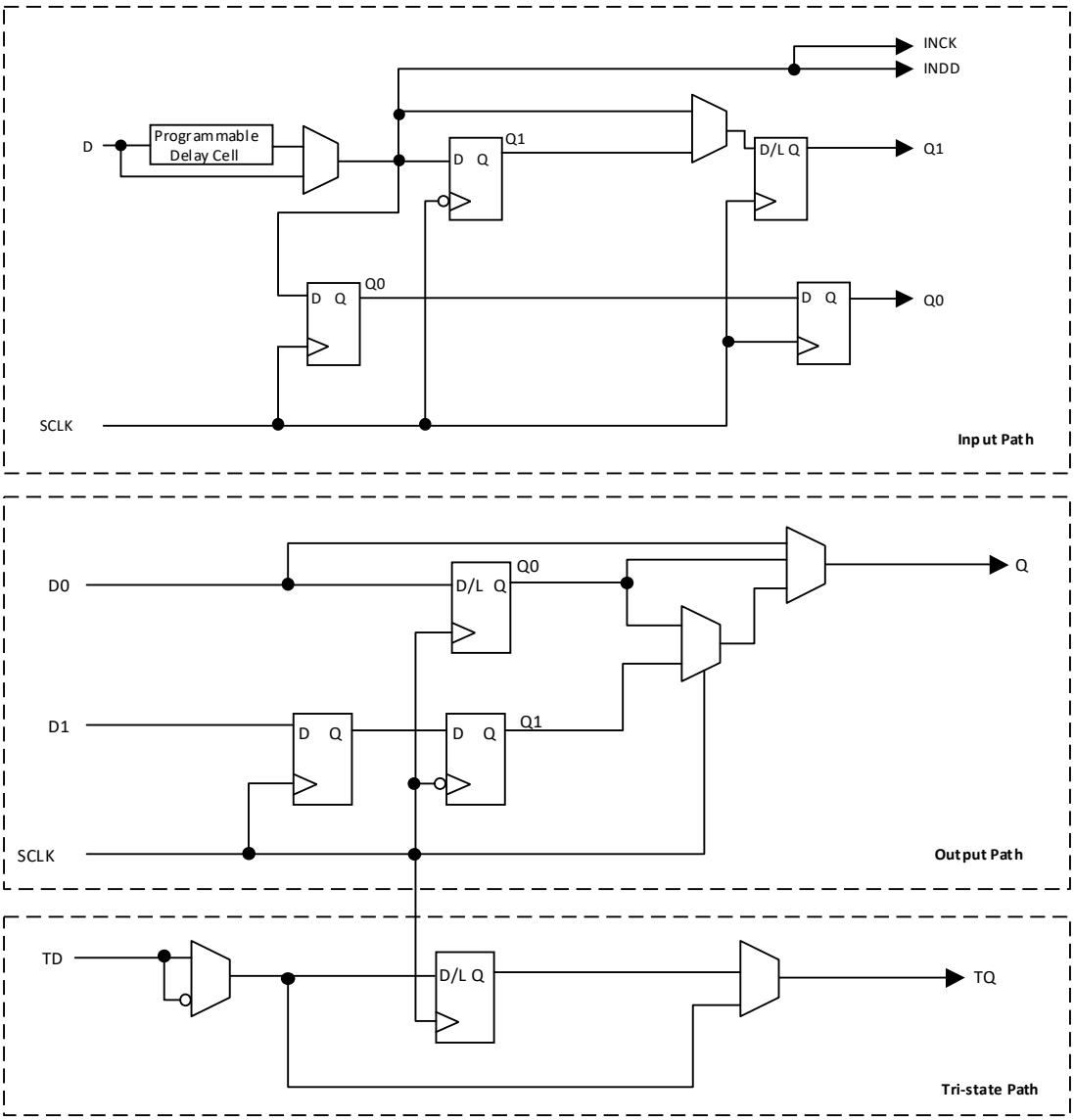
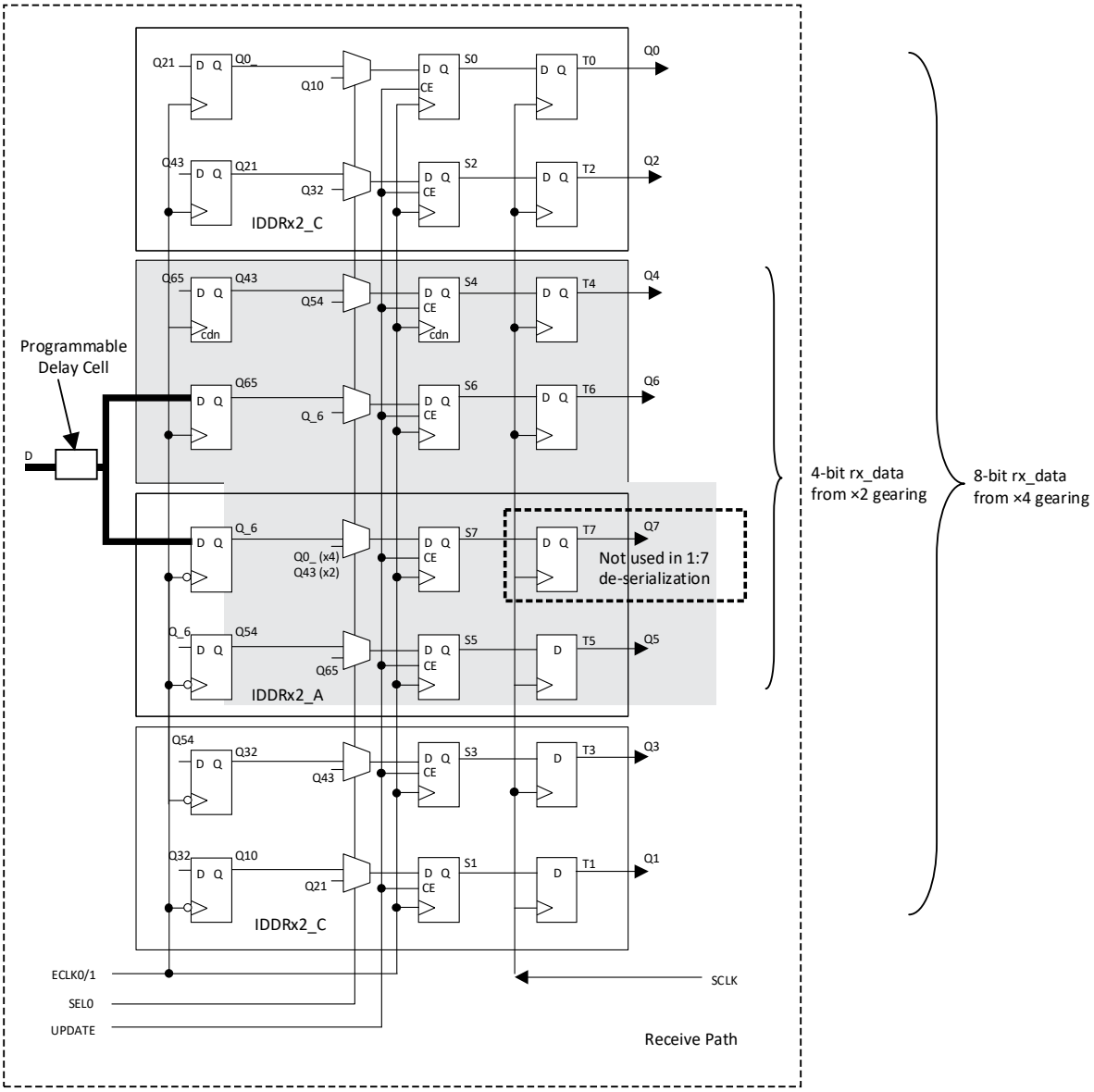


Figure 2.1. Basic PIO Cell Support ×1 Gearing Ratio



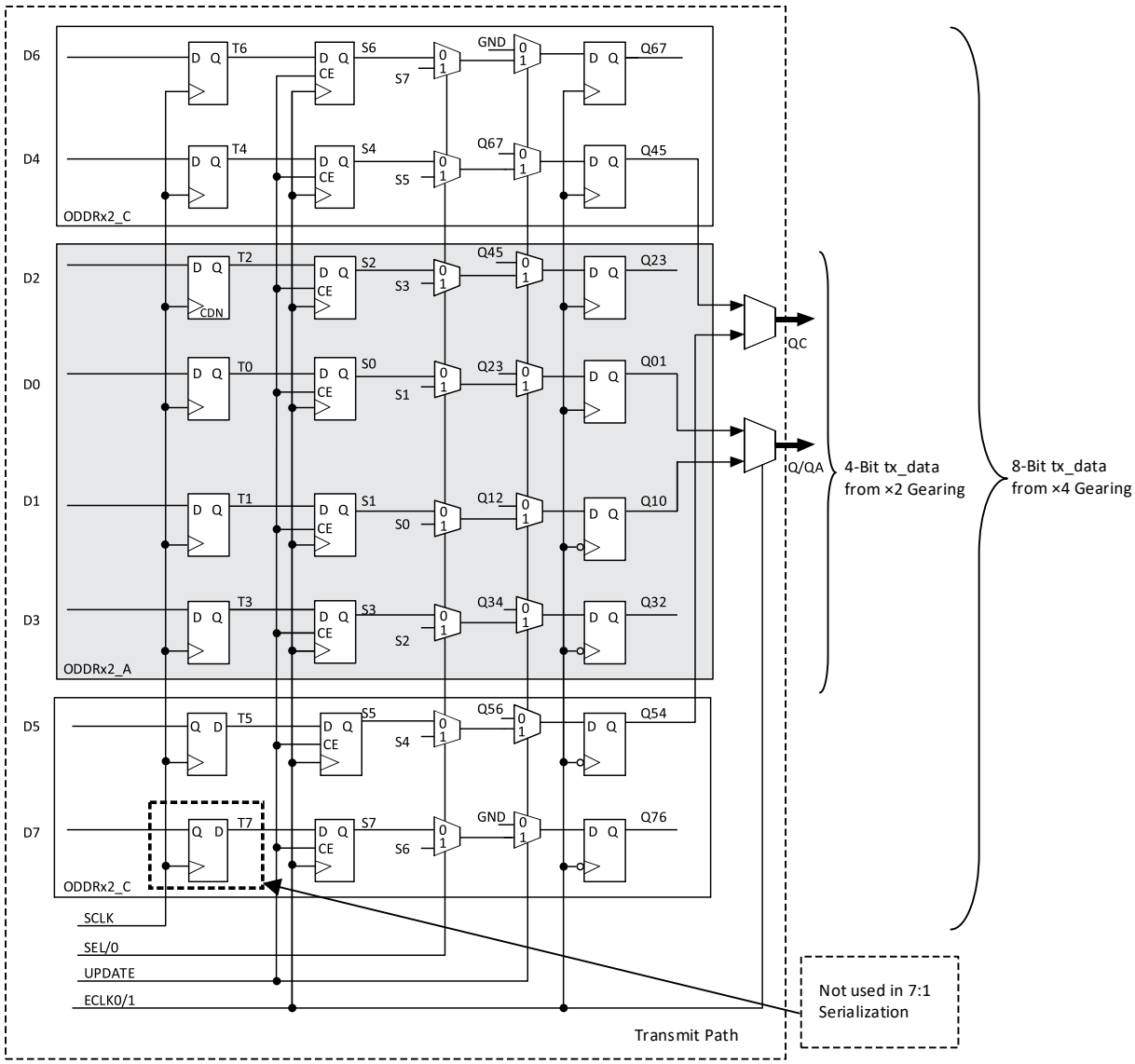


Figure 2.2. Video PIO Cell for ×2/×4 and 7:1 Applications in MachXO4

2.3. Clock Domain Transfer at PIO Cells

The MachXO4 gearing logic performs serializing and de-serializing of high-speed data in the PIO cells. The clock domain transfer for the data from the high-speed edge clock (ECLK) to the low-speed system clock (SCLK) is guaranteed by design through two internal signals, UPDATE and SEL.

The SEL signal toggles between 0 and 1 to sample three bits or four bits of data at a time for the 7:1 gearing. It remains static during the ×2/×4 gearings.

The UPDATE signal behaves the same for all the gearings to update the register with the correct byte of data. This data is then clocked by the SCLK for downstream processing. Figure 2.2 illustrates the architecture of ×2 /×4 input gearing logic.

MachXO4 devices provide logic to support word alignment with minimal FPGA resources. The word alignment results in a shift to the UPDATE, SEL and the SCLK signals. It can be activated by providing an alignment request signal to the ALIGNWD port of the high-speed interface components. ALIGNWD can be asynchronous to the ECLK domain, but it must be at least two ECLK cycles wide. For the 7:1 gearing, ALIGNWD must be pulsed seven times to loop through a

maximum of seven combinations of word orders. For the ×2/×4 gearings, ALIGNWD must be pulsed eight times to step through maximum eight possible word orders.

The MachXO4 PIO receive deserializer primitives contains a fundamental logic error for ×2 and ×4 applications. Odd word alignments (SEL = 1) do not present the correct data at the Q3 output port (×2), or Q7 output port (×4). The timing diagrams below correctly show the actual behavior of the output port, with the actual output data bit highlighted in BOLD. The designer must be aware of this behavior if instantiating the primitive directly into his design, and take appropriate steps to correct the temporal misalignment. Note that the 7:1 gearing primitives are not impacted by this issue.

The temporal misalignment mentioned above is corrected when using the Generic IDDRX4 and IDDRX2 logic interfaces (GDDRX2/X4_RX.ECLK.Centered/Aligned) generated using Lattice Radiant™ IP Catalog tool, and which are described in the Generic High-Speed DDR Interfaces section. As a result of the fix, the Q bus output is delayed by one SCLK cycle with respect to the primitive output, and with respect to previous versions of the GDDRX2/X4_RX~ interfaces. In most applications, the additional delay is negligible.

Figure 2.3 through Figure 2.7 provide a timing relationship of UPDATE, SEL, ECLK, and SCLK signals under different gearing requirements. Figure 2.8 through Figure 2.10 show the word alignment procedure for various gearing ratios. The discussion of gearing logic is applicable to both receive and transmit sides of the high-speed interfaces.

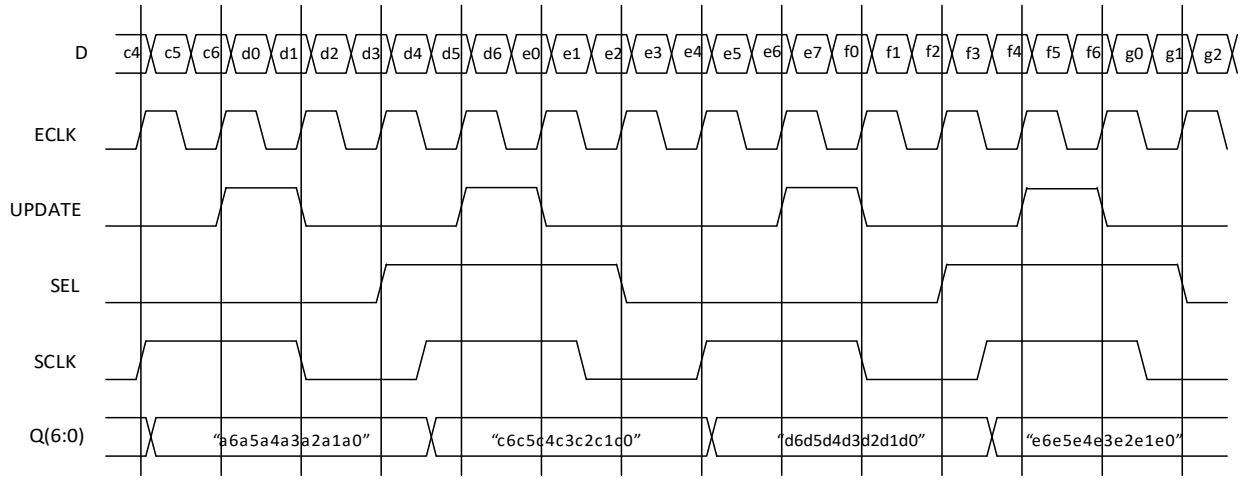


Figure 2.3. 7:1 Deserializer Timing in MachXO4 Device

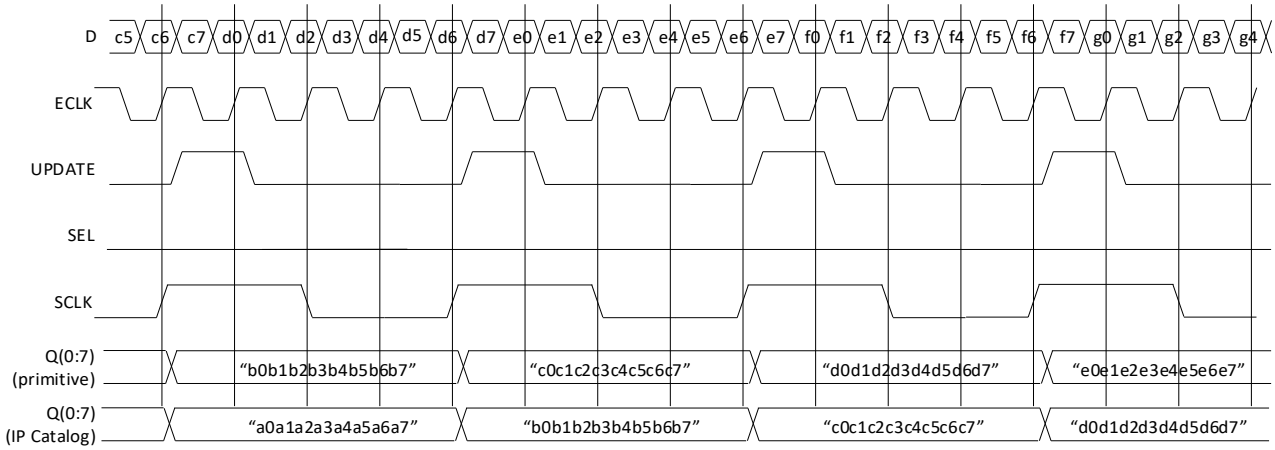


Figure 2.4. ×4 Deserializer Timing (Even Phases, SEL=0)

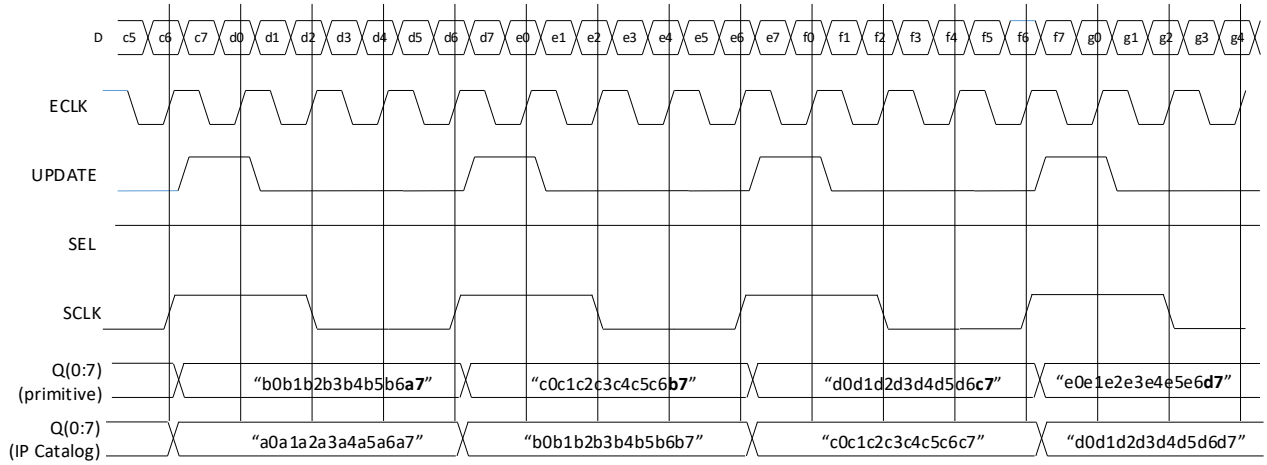


Figure 2.5. ×4 Deserializer Timing (Odd Phases, SEL=1)

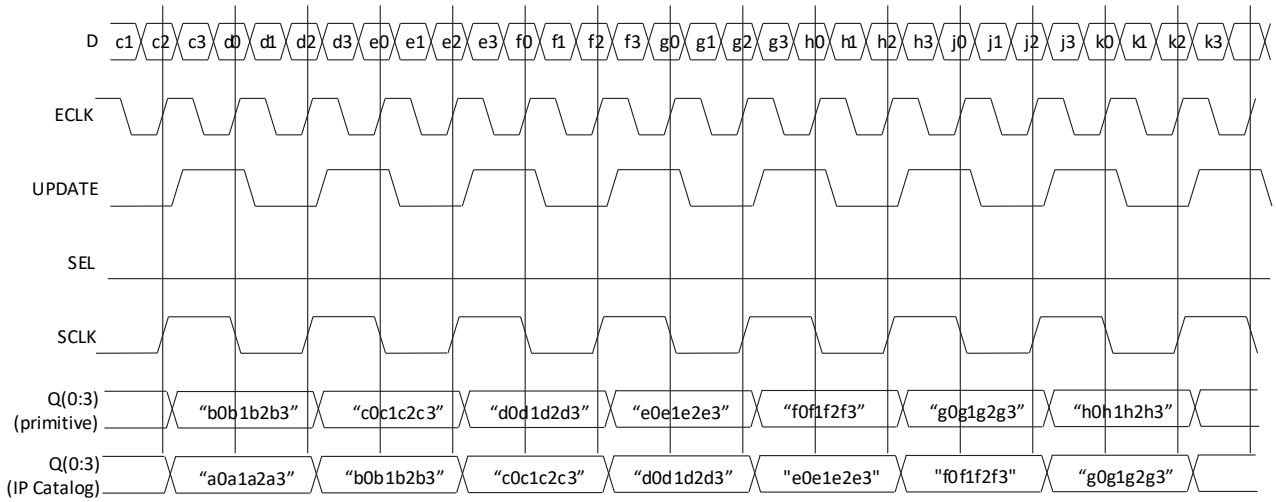


Figure 2.6. ×2 Deserializer Timing (Even Phases, SEL=0)

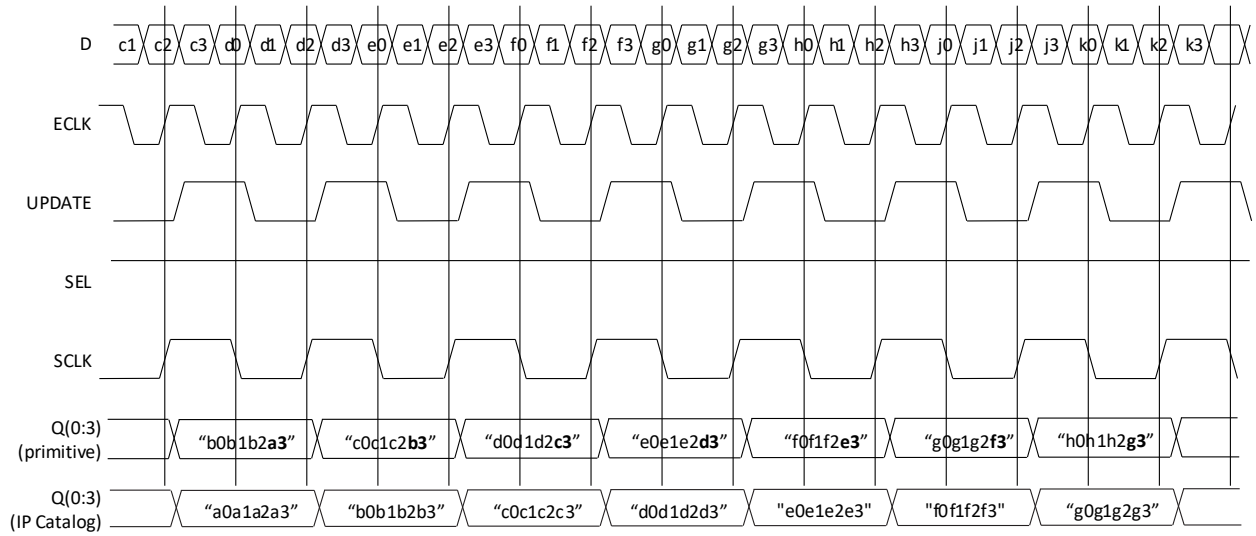


Figure 2.7. ×2 Deserializer Timing (Odd Phases, SEL=1)

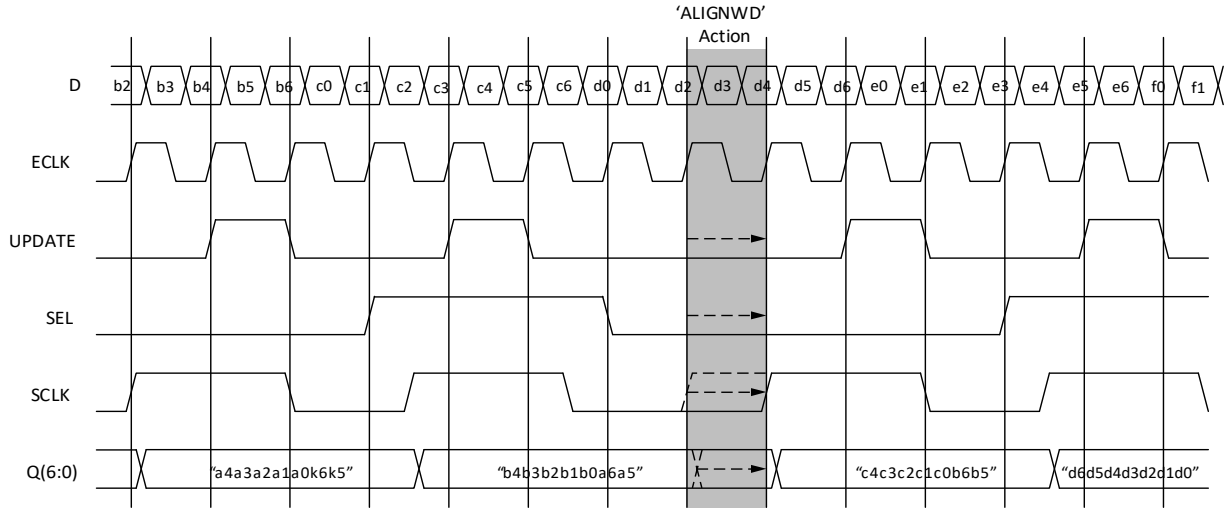


Figure 2.8. 7:1 Deserializer Timing in Response to ALIGNWD in MachXO4 Device

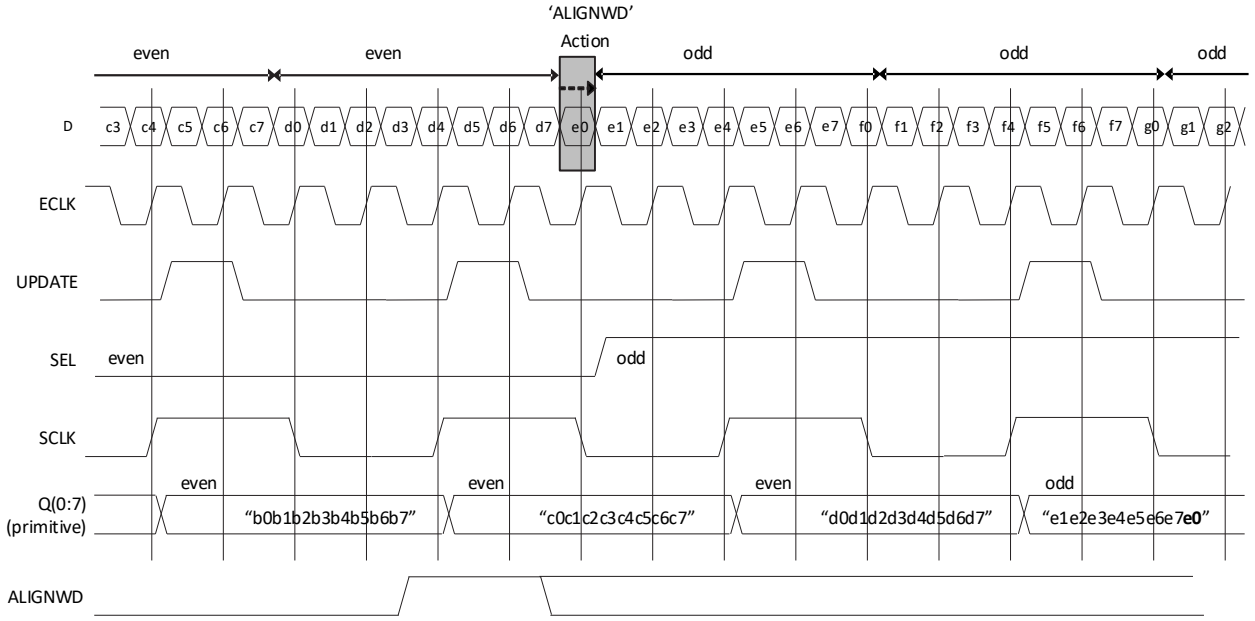


Figure 2.9. ×4 Deserializer Timing in Response to ALIGNWD (Even-to-Odd Phase)

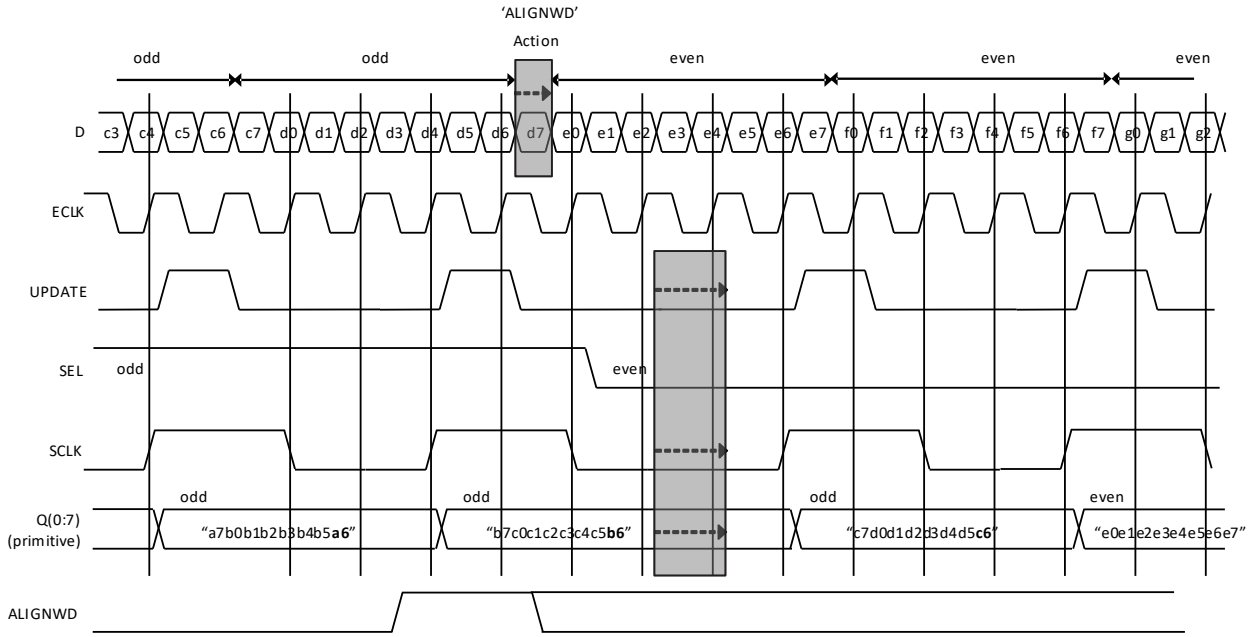


Figure 2.10. ×4 Deserializer Timing in Response to ALIGNWD (Odd-to-Even Phase)

3. External High-Speed Interface Description

There are two types of external high-speed interface definitions that can be used with the MachXO4 devices: centered and aligned. In a centered external interface, at the device pins, the clock is centered in the data opening. In an aligned external interface, the clock and data transition are aligned at the device pins. This is sometimes called as *edge-on-edge*.

Figure 3.1 shows external interface waveforms for SDR and DDR. At the receive side, an aligned interface requires clock delay adjustment to position the clock edge at the middle of the data opening to ensure that the capture flip-flop setup and hold times are not violated. Similarly, a centered interface at the transmit side requires a clock delay adjustment to position the clock at the center of the data opening for transmission.

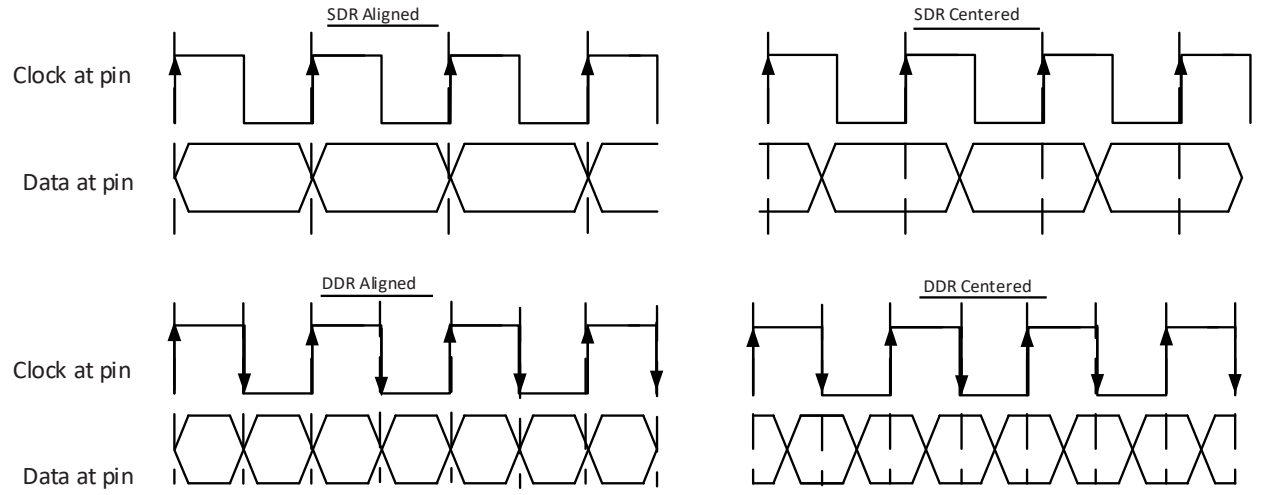


Figure 3.1. External Interface Definition

4. High-Speed Interface Building Blocks

MachXO4 devices provide dedicated logic blocks for building high-speed interfaces, with each block performing a unique function. Combining various blocks gives ultimate performance of a specific interface. The hardware components in the device are described in this section. The DDR Software Primitives and Attributes section describes the library elements for these components. Refer to MachXO4 sysCLOCK PLL Design and Usage Guide (FPGA-TN-02391), for an in-depth discussion of clocking and PLL architectures.

4.1. ECLK

Edge clocks are high-speed, low-skew I/O dedicated clocks. Two edge clocks (ECLK) are available on each of the top and bottom sides. The primary clock nets (PCLK) have direct connectivity to ECLKs. The bottom PCLK pins also have minimal routing to PLLs for video applications.

4.2. ECLKSYNC

This is the ECLK synchronization block. Each ECLK has its own ECLKSYNC component to synchronize the clock domain transfer of data. This block can also be used to dynamically disable an edge clock to save power during operation.

4.3. CLKDIV

Clock dividers are used to generate low-speed system clocks from a high-speed edge clock. The ECLK frequency can be divided down by 2, by 3.5, or by 4 through the CLKDIV component.

4.4. SCLK

SCLK refers to the system clock of the design. SCLK must use primary clock pins or primary clock nets for high speed interfaces. There are eight primary clock pins (PCLK) available for the MachXO4 device, and eight primary clock nets in the MachXO4 devices.

4.5. PLL

A maximum of two PLLs are available in the MachXO4 devices. The number of PLLs varies with the logic density.

The MachXO4-010, MachXO4-015 and MachXO4-025 have one PLL. MachXO4-050, MachXO4-080 and MachXO4-110 devices have two PLLs. There are pre-assigned dual-purpose I/O pins that drive to PLLs as reference clock inputs.

4.6. DQSDLL

A maximum of two DQSDLLs are available. The top-right DQSDLL controls the top and right banks. The bottom-left DQSDLL can be used by the bottom and left banks. The DQSDLL, together with the clock target delay cell (DLLDEL), is used to create a 90° clock shift/delay for aligned receiver interfaces.

4.7. Input DDR (IDDR)

Generic input DDR components support ×1, ×2, ×4, and 7:1 gearing ratios at the receiving side of the PIO cells.

The ×1 gearing is supported by IDDRX, or the basic PIO cell. It receives 1-bit DDR data and outputs 2-bit wide parallel data synchronized to the SCLK. There is no clock domain transfer involved in the ×1 gearing.

The ×2 gearing is supported by IDDRX2. It receives 1-bit DDR data synchronized to the ECLK and outputs four bits of parallel data synchronized to the SCLK. The same function applies to the IDDRX4, which receives a single bit of DDR data synchronized to the ECLK and outputs eight bits of parallel data synchronized to the SCLK.

The 7:1 gearing shares the same structure as the ×4 gearing. The 7:1 gearing outputs seven bits of parallel data instead of eight. The generic high-speed interface gearings are supported by the video PIO cells.

4.8. Output DDR (ODDR)

Generic output DDR components support ×1, ×2, ×4, and 7:1 gearing ratios at the transmit side of the PIO cells.

The ×1 gearing is supported by ODDRX, or the basic PIO cell. It serializes the 2-bit data based on SCLK. There is no clock domain transfer involved in ×1 gearing. The ×2 gearing is supported by ODDRX2. The 4-bit parallel data is clocked by SCLK and is serialized using ECLK. The ×4 gearing is supported by ODDRX4. The 8-bit parallel data is clocked by SCLK and is serialized using ECLK. The 7:1 gearing shares the same structure as the ×4 gearing. The 7-bit parallel data is serialized by the ECLK. The generic high speed interface gearings are supported by the video PIO cells.

4.9. Delays

There are two types of delay available for high-speed interfaces, I/O logic delay and clock target delay cell (DLLDEL).

The first type, the I/O logic delay can be applied on the input data paths, as shown in the block diagram of PIO architectures at the beginning of the document.

Although the 32-tap I/O logic delay can be static or dynamic, only the bottom side of the MachXO4 supports dynamic data path delay. The static I/O logic delay (DELAYH) is used by default when configuring the interface in the Lattice design software. Software applies fixed delay values based on the interface used.

Dynamic delay (DELAYG) at the bottom-side input data path provides dynamic or user-defined delay. Dynamic delay requires extra ports available on the module to be connected to user logic for delay control. The I/O logic delay is used to achieve the SDR zero hold timing, or match primary clock injection for ×1 gearing, or match the edge clock injection for ×2/×4 gearings.

The second type of delay, the clock target delay cell (DLLDEL), which delays the incoming clock by 90° to place the clock in the middle of the data opening. This block is digitally controlled by the DQSDLL through 7-bit control code. There is one clock target delay cell per primary clock pin. Its input comes from the primary clock pins and its output can drive the primary clock net for the ×1 aligned interface or ECLK for ×2/×4 aligned interfaces.

5. Generic High-Speed DDR Interfaces

Generic high-speed interfaces, or Generic DDR (GDDR), are supported in MachXO4 devices using the dedicated logic blocks. This section will discuss the GDDR types, the interface logic, and the software to support the GDDR capability in the silicon.

5.1. High-Speed GDDR Interface Types

The GDDR interfaces supported by the MachXO4 devices are pre-defined in the software and characterized in the silicon. Table 5.1 lists all the supported interfaces and gives a brief description of each interface.

Table 5.1. Generic High-Speed I/O DDR Interfaces

Mode	Interface Name	Description	Supporting Device & Sides
RX SDR	GIREG_RX.SCLK	SDR input using SCLK	All sides
RX GDDRx1 Aligned	GDDRX1_RX.SCLK.Aligned	DDRx1 input using SCLK, data is edge-to-edge with incoming clock	All sides
RX GDDRx1 Centered	GDDRX1_RX.SCLK.Centered	DDRx1 input using SCLK, incoming clock is centered at the data opening	All sides
RX GDDRx2 Aligned	GDDRX2_RX.ECLK.Aligned	DDRx2 input using ECLK, data is edge-to-edge with incoming clock	Bottom
RX GDDRx2 Centered	GDDRX2_RX.ECLK.Centered	DDRx2 input using ECLK, incoming clock is centered at the data opening	Bottom
RX GDDRx4 Aligned	GDDRX4_RX.ECLK.Aligned	DDRx4 input using ECLK, data is edge-to-edge with incoming clock	Bottom
RX GDDRx4 Centered	GDDRX4_RX.ECLK.Centered	DDRx4 input using ECLK, incoming clock is centered at the data opening	Bottom
RX GDDR71	GDDR71_RX.ECLK.7:1	GDDR 7:1 input using ECLK	Bottom
TX SDR	GOREG_TX.SCLK	SDR output using SCLK	All sides
TX GDDRx1 Aligned	GDDRX1_TX.SCLK.Aligned	DDRx1 output using SCLK, data is edge-to-edge with outgoing clock	All sides
TX GDDRx1 Centered	GDDRX1_TX.SCLK.Centered	DDRx1 output using SCLK, outgoing clock is centered at the data opening	All sides
TX GDDRx2 Aligned	GDDRX2_TX.ECLK.Aligned	DDRx2 output using ECLK, data is edge-to-edge with outgoing clock	Top
TX GDDRx2 Centered	GDDRX2_TX.ECLK.Centered	DDRx2 output using ECLK, outgoing clock is centered at the data opening	Top
TX GDDRx4 Aligned	GDDRX4_TX.ECLK.Aligned	DDRx4 output using ECLK, data is edge-to-edge with outgoing clock	Top
TX GDDRx4 Centered	GDDRX4_TX.ECLK.Centered	DDRx4 output using ECLK, outgoing clock is centered at the data opening	Top
TX GDDR71	GDDR71_TX.ECLK.7:1	GDDR 7:1 output using ECLK	Top

Note: MIPI D-PHY Receive can be built using GDDRX4_RX.ECLK.Centered interface, while MIPI D-PHY Transmit can be built using GDDRX4_TX.ECLK.Centered interface. Refer to MIPI D-PHY Interface IP (FPGA-RD-02040) for details.
Note that, while this reference design was originally targeted for MachXO3 and ECP5 devices, the concept can still be applied for MachXO4 but additional steps may be required. Contact Technical Support Assistance for more details.

The following describes the naming conventions used for each of the interfaces listed in Table 5.1.

- G – Generic
- IREG – SDR input I/O register
- OREG – SDR output I/O register
- DDRX1 – DDR ×1 I/O register
- DDRX2 – DDR ×2 I/O register
- DDRX4 – DDR ×4 I/O register

- DDR71 – DDR 7:1 I/I register
- _RX – Receive interface
- _TX – Transmit interface
- ECLK – Uses ECLK (edge clock) clocking resource at the GDDR interface
- SCLK – Uses SCLK (primary clock) clocking resource at the GDDR interface
- Centered – Clock is centered to the data when coming into the device
- Aligned – Clock is aligned edge-on-edge to the data when coming into the device

5.2. High-Speed GDDR 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 MachXO4 clocking structure, refer to MachXO4 sysCLOCK PLL Design and Usage Guide (FPGA-TN-02391).

As listed in Table 5.1, each interface is supported in specific bank locations of the MachXO4 devices. It is important to follow the architecture and various interface rules and preferences listed under each interface to build these interfaces successfully. The discussion of each component can be found in the DDR Software Primitives and Attributes section of this document.

5.2.1. Receive Interfaces

There are eight receive interfaces pre-defined and supported through the Lattice Radiant software. It is recommended to generate interfaces using IP Catalog instead of manually instantiating primitives as the parameters will be properly configured using the IP Catalog.

5.2.1.1. GIREG_RX.SCLK

This is a generic interface for SDR data. The standard I/O register in the basic PIO cell as shown in Figure 2.1 is used for the implementation. An optional inverter can be used to center the clock for aligned inputs. PLLs or DLLs can be used to remove the clock injection delay or adjust the setup and hold times. There are a limited number of DLLs in the architecture, and these should be saved for high-speed interfaces when necessary. This interface can either be built using IP Catalog, instantiating an I/O register element, or inferred during synthesis.

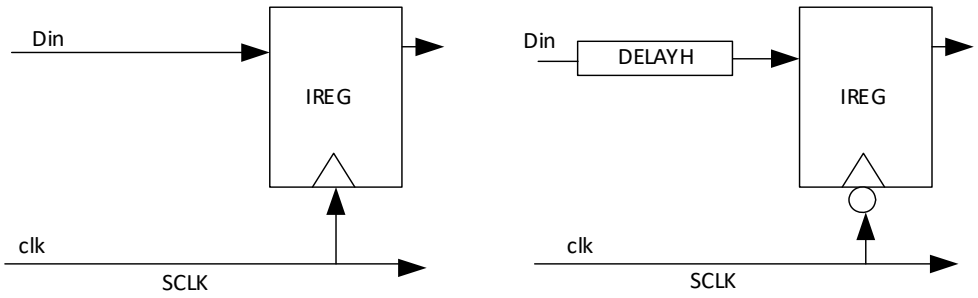


Figure 5.1. GIREG_RX Interface

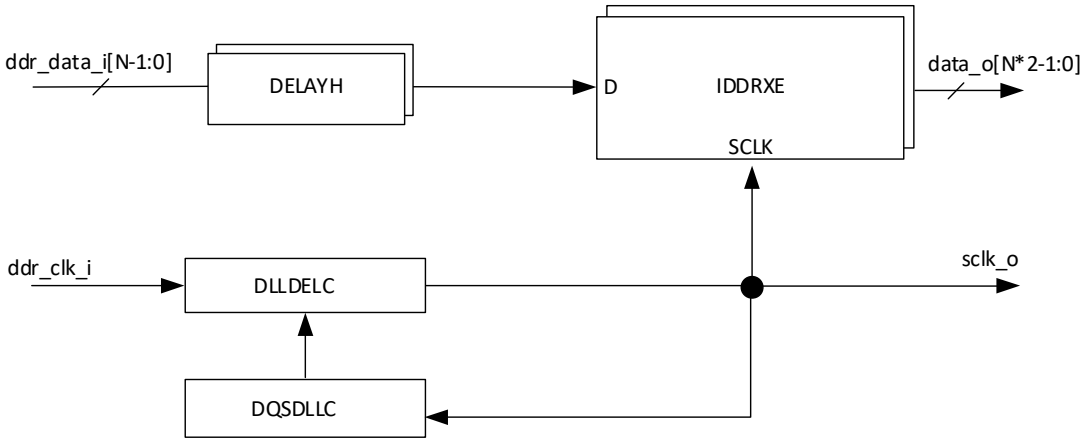
The input data path delay cells can be used on the Din path of the interface. A DELAYH element provides a fixed delay to match the SCLK injection time. The dynamic input delay, DELAYG, is not available for this interface. Figure 5.1 shows possible implementations of this interface. For detailed information, refer to MachXO4 SDR Module User Guide (FPGA-IPUG-02317).

Interface rule:

Must use a dedicated clock pin PCLK as the clock source

5.2.1.2. GDDRX1_RX.SCLK.Aligned

This DDR interface uses the SCLK and the DQSDLL to provide a 90° clock shift to center the clock at the IDDRXE. A DELAYH element is used to adjust data delay for the SCLK clock injection time. The DELAYG is not available for the ×1 interface. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Note: N = Number of lanes or DDR bus width

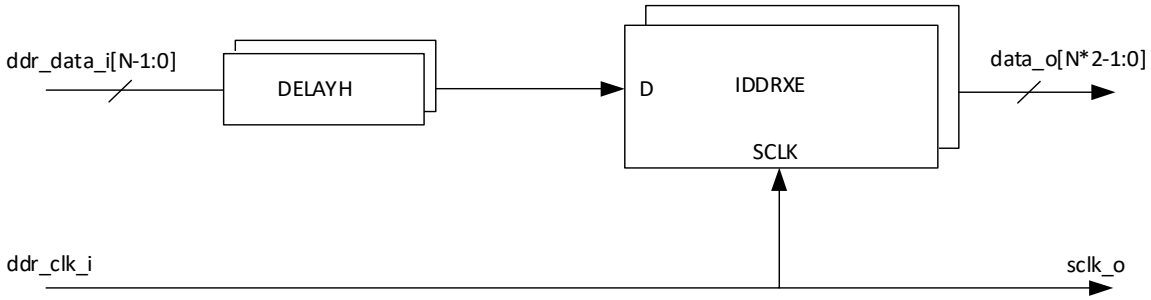
Figure 5.2. GDDRX1_RX.SCLK.Aligned Interface Using DQSDLL

Interface rules:

- Must use a dedicated clock pin PCLK as the clock source for DLLDELC
- A primary clock net must be used to connect DLL outputs to the SCLK port
- The DELAYH value should be set to SCLK_ALIGNED for the best timing
- There are up to two DQSDLLCs per device. This limits the interface to a maximum of two clock frequencies per device.

5.2.1.3. GDDRX1_RX.SCLK.Centered

This DDR interface uses DELAYH to match the SCLK delay at the IDDRXE. DELAYG is not available for the ×1 interface. Since it is a centered interface, the clock edge is already in the middle of the data opening. There is no logic required to shift the clock. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Note: N = Number of lanes or DDR bus width

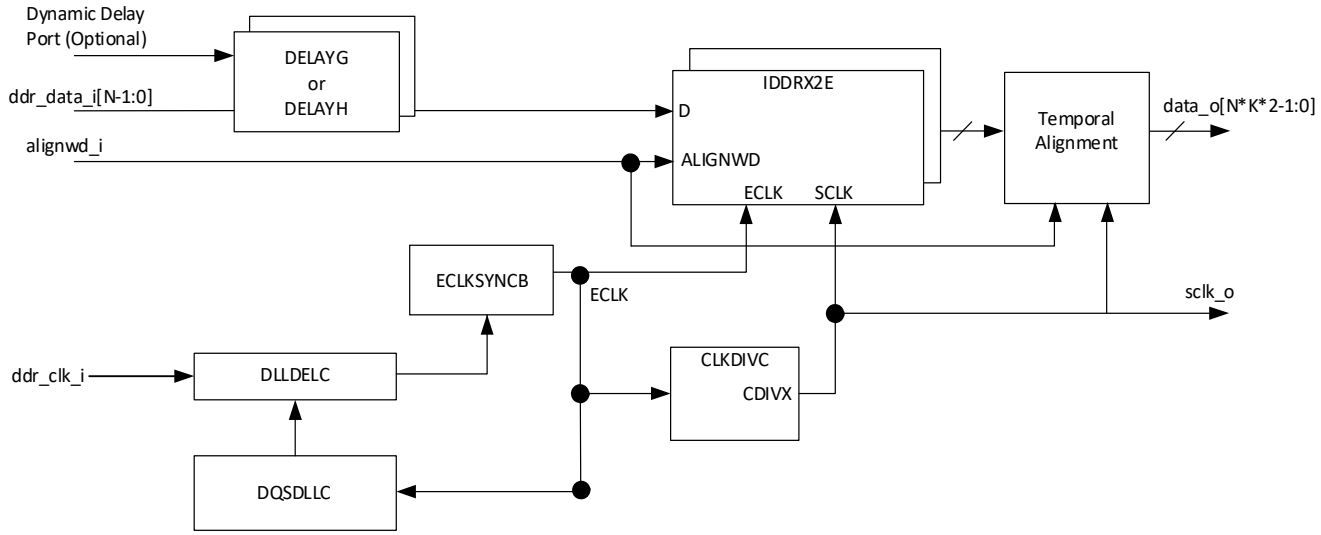
Figure 5.3. GDDRX1_RX.SCLK.Centered

Interface rules:

- Must use a dedicated clock pin PCLK as the clock source
- DELAYH value should be set to SCLK_CENTERED for the best timing
- The clock connected to SCLK should be on a primary clock net

5.2.1.4. GDDRX2_RX.ECLK.Aligned

This DDR ×2 interface uses the DQSDLL to provide a 90° clock shift to center the clock at the IDDRX2E buffer. DELAYH is used to delay data to match the ECLK injection delay. DELAYG can also be used to control the delay dynamically. This interface uses ×2 gearing with the IDDRX2E element. This requires the use of a CLKDIVC to provide the SCLK which is half the frequency of the ECLK. The ECLKSYNCB element is associated with the ECLK and must be used to drive the ECLK. The port ALIGNWD can be used for word alignment at the interface. The Temporal Alignment block introduces a one SCLK period delay and corrects for the temporal misalignment of IDDRX2E Q3 data output. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:

- N = Number of lanes or DDR bus width.
- K = 2 (for x1 gearing), 4 (for x2 gearing), and 8 (for x4 gearing)

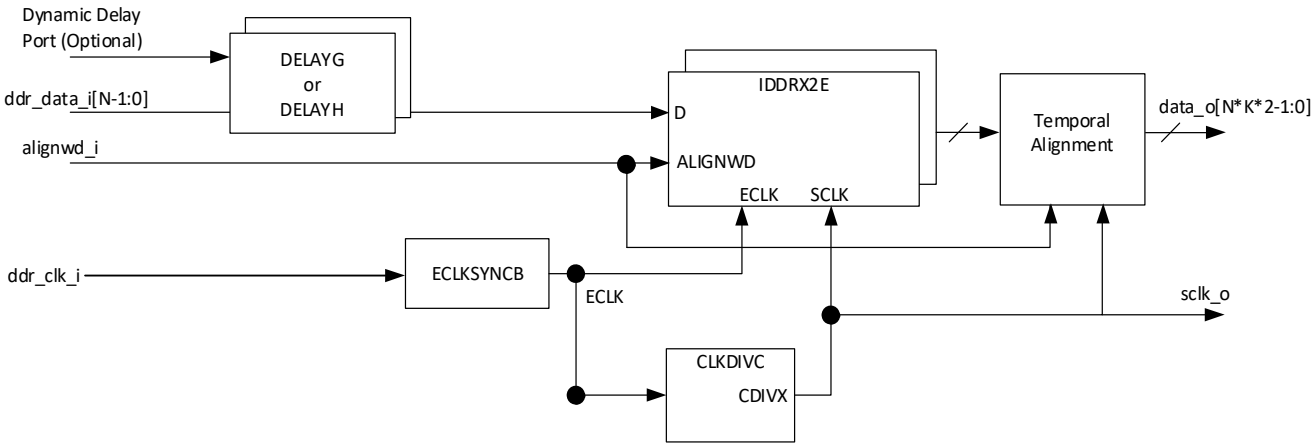
Figure 5.4. GDDRX2_RX.ECLK.Aligned Interface

Interface rules:

- Must use a dedicated clock pin PCLK as the clock source for DLLDELC
- Clock net routed to SCLK must use primary clock net
- There are up to two DQSDLLC per device. It limits this interface to a maximum of two clock frequencies per device
- DELAYG should be set to ECLK_ALIGNED.
- When DELAYG is used, only one dynamic delay port is needed for the entire bus
- This interface is supported at the bottom side of the devices

5.2.1.5. GDDRX2_RX.ECLK.Centered

This DDR ×2 interface uses DELAYH or DELAYG to match edge clock delay at the IDDRX2E. Since this interface uses the ECLK, it can be extended to support large data bus sizes for the entire side of the device. This interface uses ×2 gearing with the IDDRX2D element. This requires the use of a CLKDIVC to provide the SCLK which is half the frequency of the ECLK. The port ALIGNWD can be used for word alignment at the interface. The Temporal Alignment block introduces a one SCLK period delay and corrects for the temporal misalignment of IDDRX2E Q3 data output. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:
- N = Number of lanes or DDR bus width.
- K = 2 (for x1 gearing), 4 (for x2 gearing), and 8 (for x4 gearing)

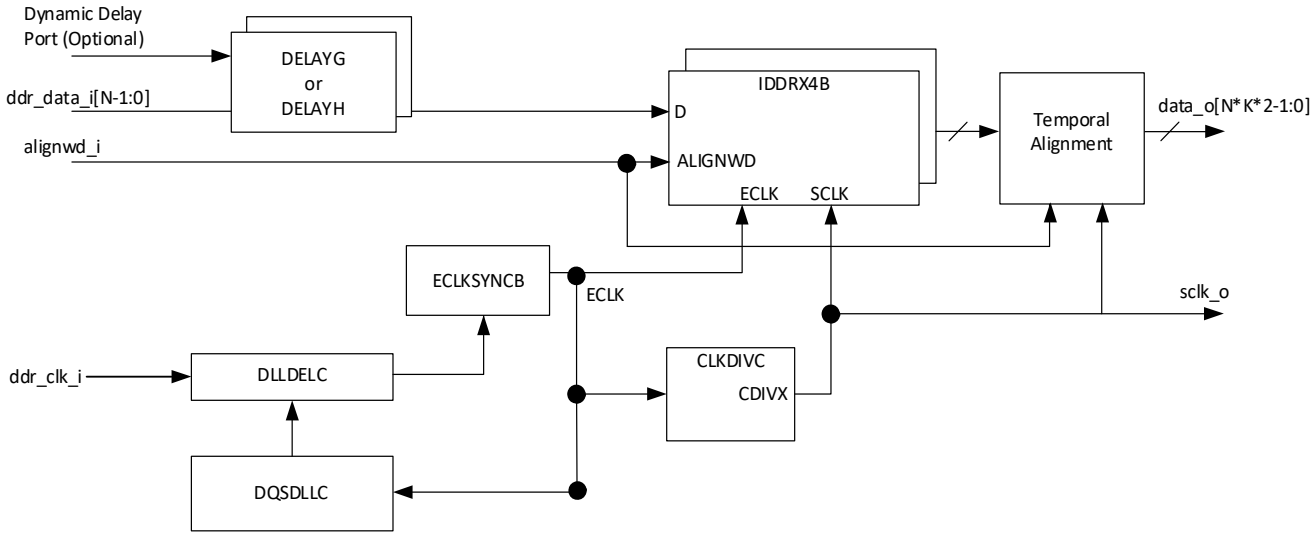
Figure 5.5. GDDRX2_RX.ECLK.Centered Interface

Interface rules:

- Must use a dedicated clock pin PCLK as the clock source for ECLKSYNCB
- Clock net routed to SCLK must use primary clock net
- DELAYH should be set to ECLK_CENTERED
- When DELAYG is used, only one dynamic delay port is needed for the entire bus
- This interface is supported at the bottom side of the devices

5.2.1.6. GDDRX4_RX.ECLK.Aligned

This DDR ×4 interface uses the DQSDLL to provide a 90° clock shift to center the edge clock at the IDDRX4B buffer. DELAYH is used to delay data to match the ECLK injection delay. DELAYG can also be used to control the delay dynamically. Since this interface uses the ECLK, it can be extended to support large data bus sizes for the entire side of the device. This interface uses ×4 gearing with the IDDRX4B element. This requires the use of a CLKDIVC to provide the SCLK which is one quarter of the ECLK frequency. ECLKSYNCB element is associated with the ECLK and must be used to drive the ECLK. The port ALIGNWD can be used for word alignment at the interface. The Temporal Alignment block introduces a one SCLK period delay and corrects for the temporal misalignment of IDDRX4B Q7 data output. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:
- N = Number of lanes or DDR bus width.
- K = 2 (for x1 gearing), 4 (for x2 gearing), and 8 (for x4 gearing)

Figure 5.6. GDDRX4_RX.ECLK.Aligned Interface

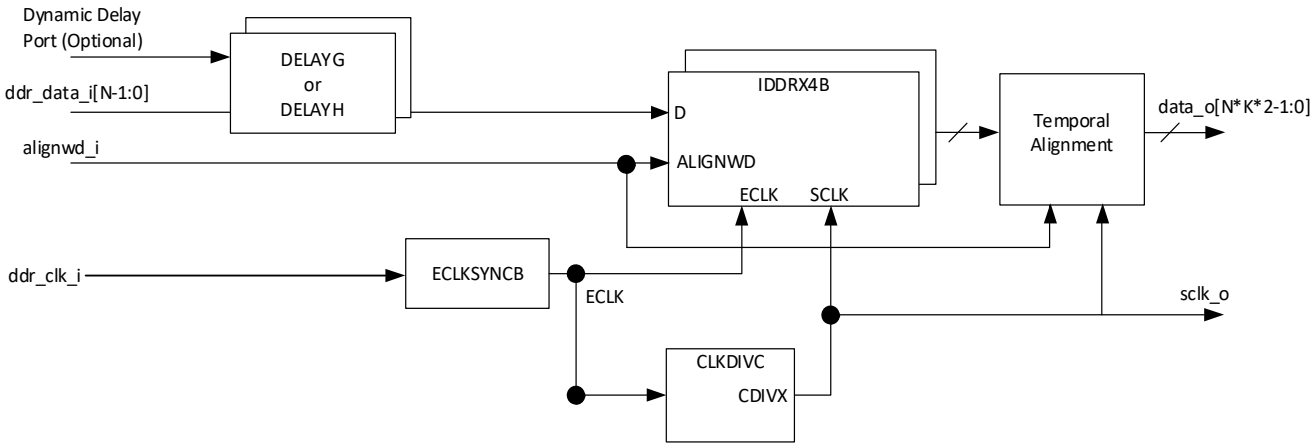
Interface rules:

- Must use a dedicated clock pin PCLK as the clock source for DLLDELC
- Clock net routed to SCLK must use primary clock net
- There are up to two DQSDLLC per device. It limits this interface to have maximum of two clock frequencies per device.
- Data input must use A/B pair of the I/O logic cells for ×4 gearing
- DELAYG should be set to ECLK_ALIGNED
- When DELAYG is used, only one dynamic delay port is needed for the entire bus
- This interface is supported in bank 2

Note: The GDDRX4_RX.ECLK.Centered interface is used to build MIPI D-PHY Receive Interface. Refer to MIPI D-PHY Interface IP (FPGA-RD-02040) for details. Note that, while this reference design was originally targeted for MachXO3 and ECP5 devices, the concept can still be applied for MachXO4, but additional steps may be required. Contact Technical Support Assistance for more details.

5.2.1.7. GDDRX4_RX.ECLK.Centered

This DDR ×4 interface uses DELAYH or DELAYG to match edge clock delay at the IDDRX4B. Since this interface uses the ECLK, it can be extended to support large data bus sizes for the entire side of the device. This interface uses ×4 gearing with the IDDRX4B element. This requires the use of a CLKDIVC to provide the SCLK, which is one quarter of the ECLK frequency. The port ALIGNWD can be used for word alignment at the interface. The Temporal Alignment block introduces a one SCLK period delay and corrects for the temporal misalignment of IDDRX4B Q7 data output. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:
- N = Number of lanes or DDR bus width.
- K = 2 (for x1 gearing), 4 (for x2 gearing), and 8 (for x4 gearing)

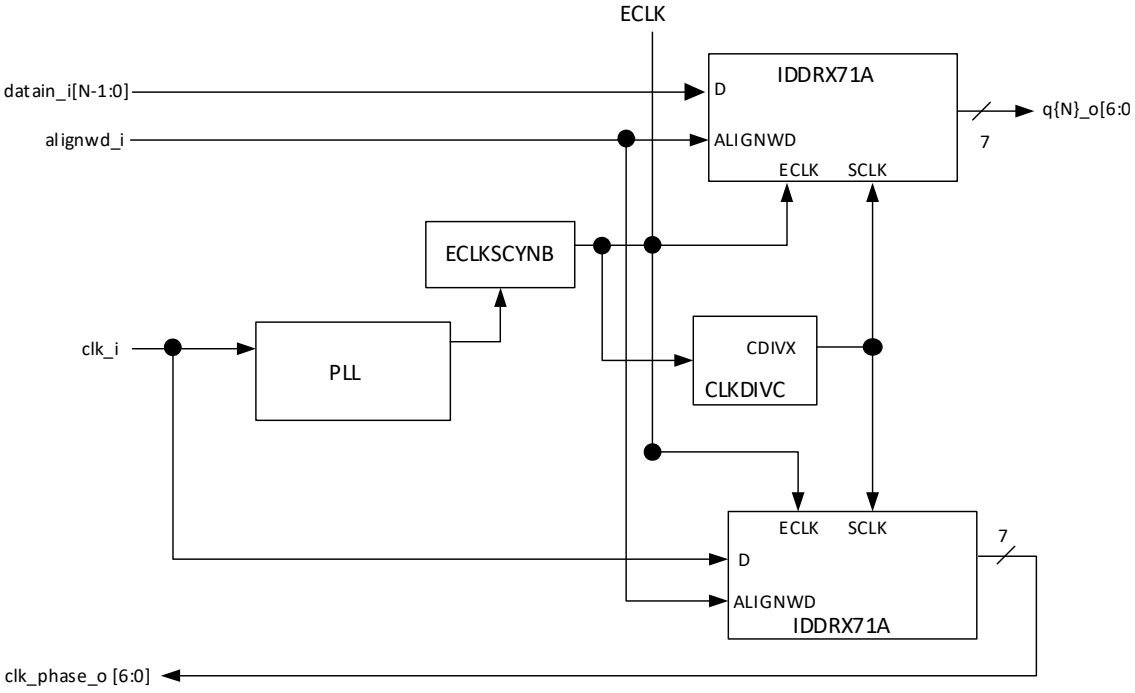
Figure 5.7. GDDRX4_RX.ECLK.Centered Interface

Interface rules:

- Must use a dedicated clock pin PCLK as the clock source for ECLKSYNCB
- Clock net routed to SCLK must use primary clock net
- Data input must use A/B pair of the I/O logic for ×4 gearing
- DELAYH should be set to ECLK_CENTERED
- When DELAYH is used, one dynamic delay port is needed for the entire bus
- This interface is supported at the bottom side of the devices

5.2.1.8. GDDR71_RX.ECLK.7:1

The GDDR 7:1 receive interface is unique among the supported high-speed DDR interfaces. It uses the PLL to search for the best clock edge to the data opening position during bit alignment process. The PLL steps through the 16 phases to give eight sampling points per data. The data path delay is not used in this interface. CLKDIVC is used to divide down the ECLK by 3.5 due to the nature of the 1:7 deserializing requirement. This means the SCLK is running seven times slower than the incoming data rate. ECLKSYNCB element is associated with the ECLK and must be used to drive the ECLK. The complete 7:1 LVDS video display application requires bit alignment and word alignment blocks to be built in the FPGA resources in addition to the built-in I/O gearing logic and alignment logic. The CLK_PHASE signal is sent to the FPGA side to build the bit alignment logic. For detailed information, refer to MachXO4 DDR 7:1 Module User Guide (FPGA-IPUG-02315).



Note: n = number of lanes. The range of *n* is from 1 to 21.

Figure 5.8. GDDR71_RX.ECLK.7:1 Interface

Interface rules:

- Must use a dedicated clock pin PCLK at the bottom side as the clock source for PLL
- Clock net routed to SCLK must use primary clock net
- There are up to two PLLs per device. It limits this interface to have maximum of two clock frequencies per device.
- The data input must use A/B pair of the I/O logic cell for 7:1 gearing
- This interface is supported at the bottom side of the devices

5.2.2. Transmit Interfaces

There are eight transmit interfaces pre-defined and supported through IP Catalog in Lattice Radiant software.

5.2.2.1. GOREG_TX.SCLK

This is a generic interface for SDR data and a forwarded clock. The standard register in the basic PIO cell is used to implement this interface. The ODDRXE used for the output clock balances the clock path to match the data path. A PLL can also be used to clock the ODDRXE to phase shift the clock to provide a precise clock to data output. There are a limited number of PLLs in the architecture, and these should be saved for high-speed interfaces when necessary. This interface can either be built using IP Catalog, instantiating an I/O register element, or inferred during synthesis. For detailed information, refer to MachXO4 SDR Module User Guide (FPGA-IPUG-02317).

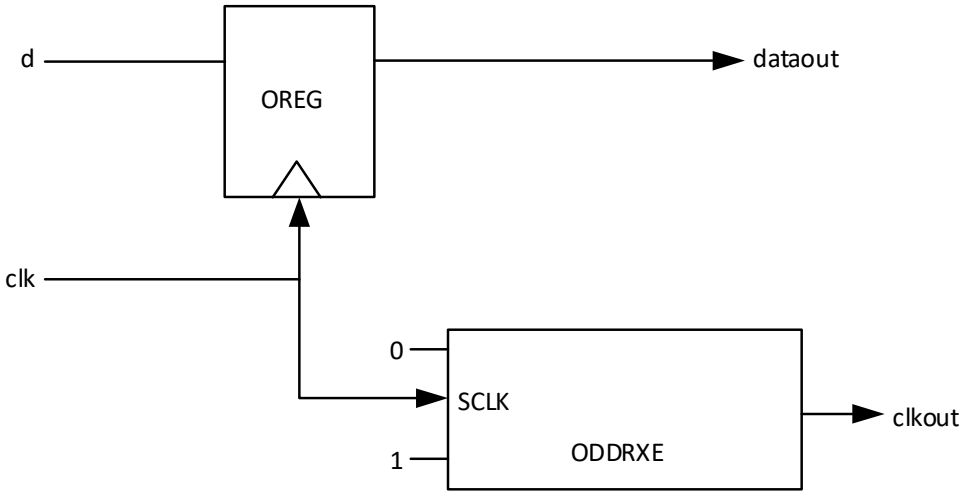


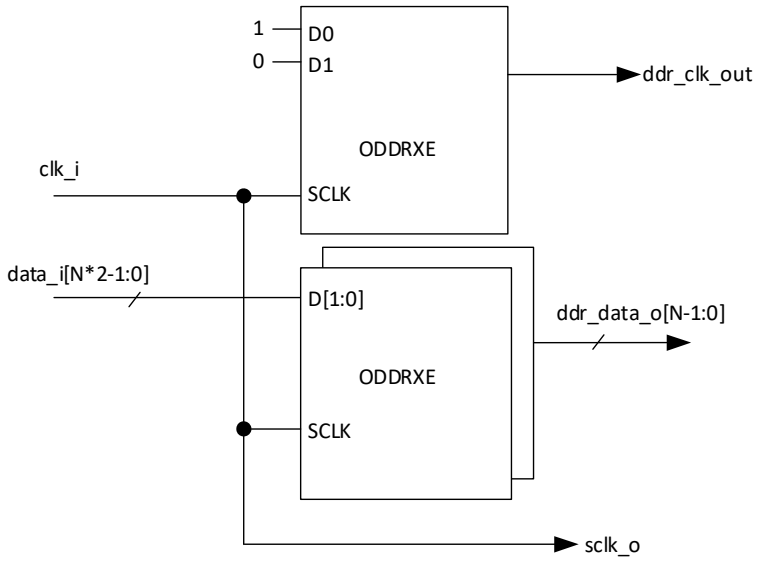
Figure 5.9. GOREG_TX.SCLK Interface

Interface rule:

The clock source for SCLK must be routed on a primary clock net

5.2.2.2. GDDRX1_TX.SCLK.Aligned

This output DDR interface provides clock and data that are aligned using a single SCLK. The ODDRXE used for the output clock balances the clock path to match the data path. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Note: N = number of lanes or DDR bus width

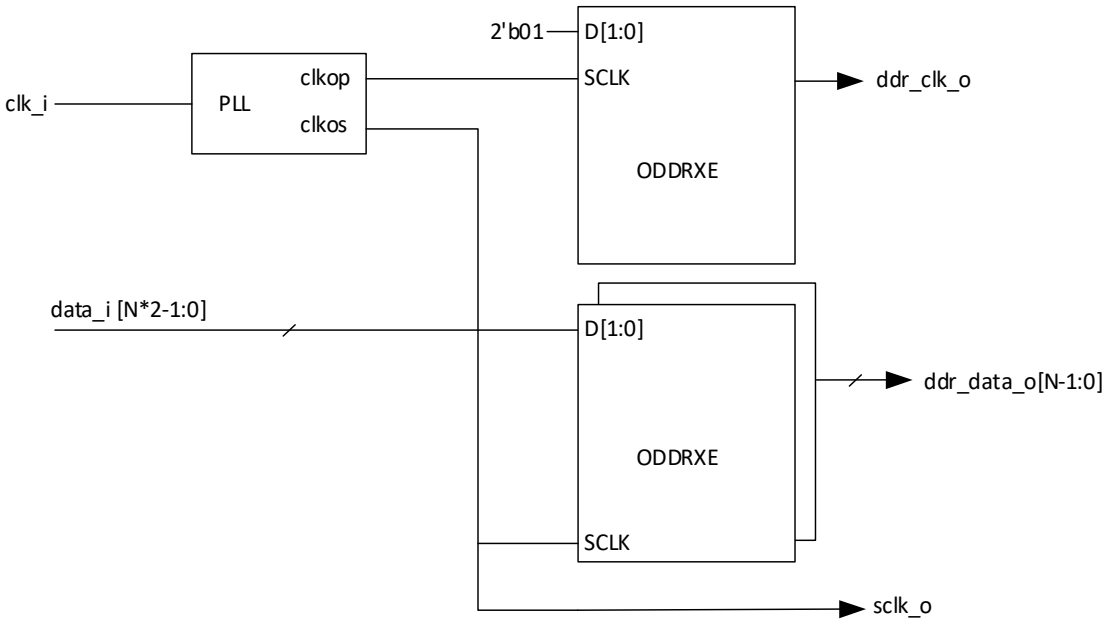
Figure 5.10. GDDRX1_TX.SCLK.Aligned Interface

Interface rule:

The clock source for SCLK must be routed on a primary clock net

5.2.2.3. GDDRX1_TX.SCLK.Centered

This output DDR interface provides clock and data that are pre-centered. PLL uses clkop and clkos ports to provide the 90° phase difference between the data and the clock. It requires two SCLK resources to drive the output data I/O cell and the output clock I/O cell. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Note: N = number of lanes of DDR bus width

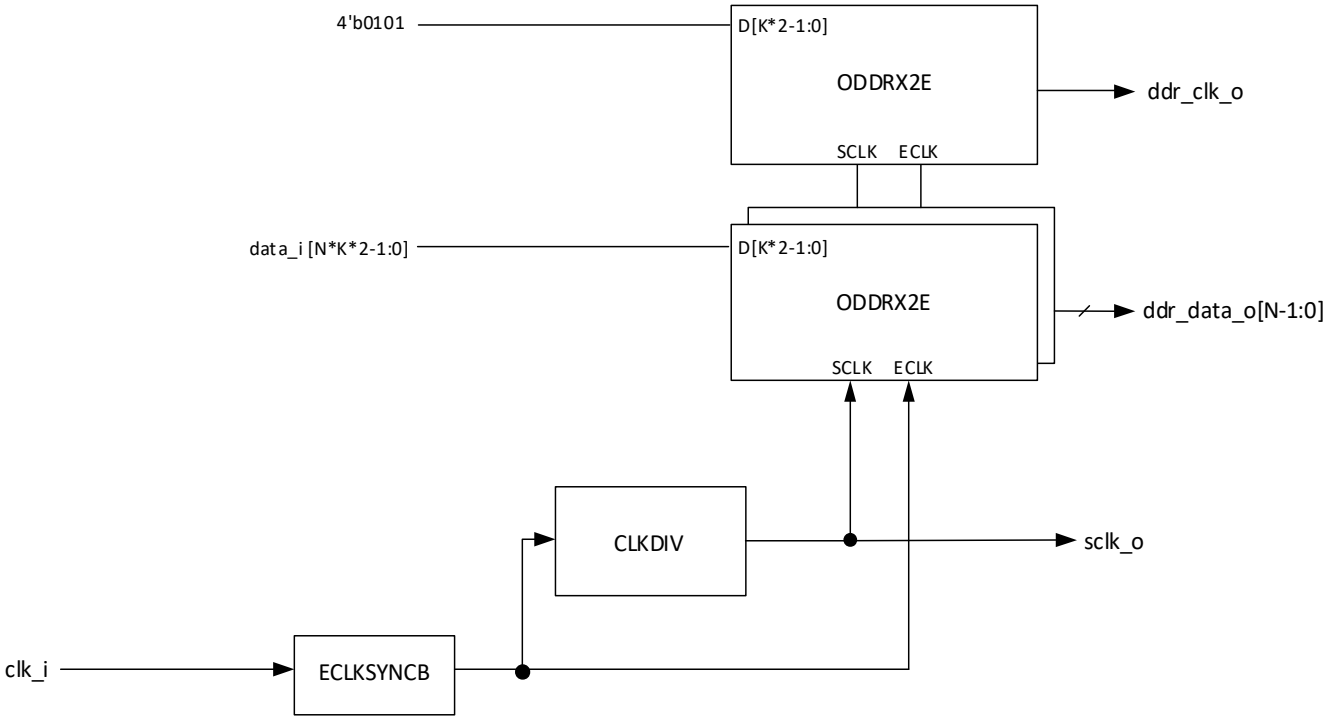
Figure 5.11. GDDRX1_TX.SCLK.Centered Interface

Interface rule:

SCLK and 90°-shifted SCLK must be routed on primary clock nets

5.2.2.4. GDDRX2_TX.ECLK.Aligned

This output DDR ×2 interface provides clock and data that are aligned. A CLKDIV is used to generate the SCLK which is half of the ECLK frequency. The ECLKSYNC element is used on the ECLK path for data synchronization. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:

- N = Number of lanes or DDR bus width
- K = 2 (for x1 gearing), 4 (for x2 gearing), and 8 (for x4 gearing)

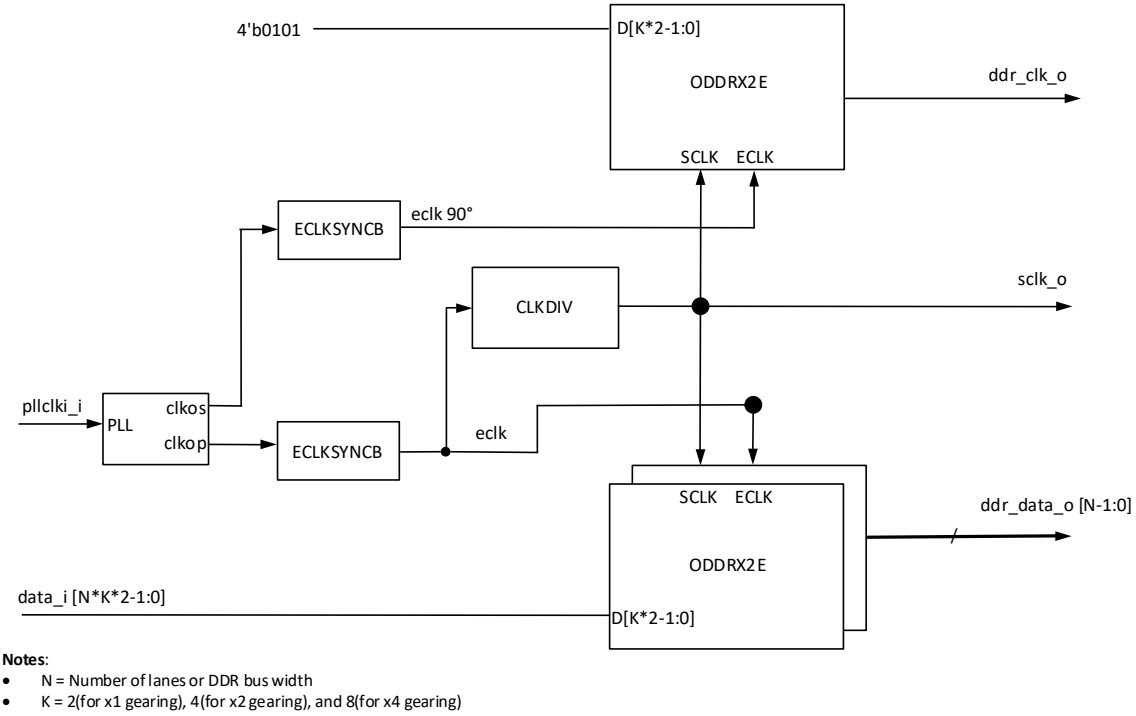
Figure 5.12. GDDRX2_TX.ECLK.Aligned Interface

Interface rules:

- Must use edge clock routing resources for the ECLK
- The routing of SCLK must use primary clock net
- This interface is supported in the bank 0

5.2.2.5. GDDRX2_TX.ECLK.Centered

This output DDR ×2 interface provides a clock that is centered at the data opening. The PLL uses clkop and clkos ports to provide the 90° phase difference between the data and the clock. Two ECLK routing resources are used in this interface to drive the output data and the output clock. A CLKDIV is used to generate the SCLK which is half of the ECLK frequency. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).



Notes:
- N = Number of lanes or DDR bus width
- K = 2(for x1 gearing), 4(for x2 gearing), and 8(for x4 gearing)

Figure 5.13. GDDRX2_TX.ECLK.Centered Interface

Interface rules:

- Must use edge clock routing resources for the ECLK
- Since two ECLKs are used on this interface, maximum one bus of this interface can be implemented at a time
- The routing of the SCLK must use primary clock net
- This interface is supported at the top side of the device

5.2.2.6. GDDRX4_TX.ECLK.Aligned

This output DDR ×4 interface provides clock and data that are aligned. A CLKDIV is used to generate the SCLK which is a quarter of the ECLK frequency. The ECLKSYNC element is used on the ECLK path for data synchronization. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).

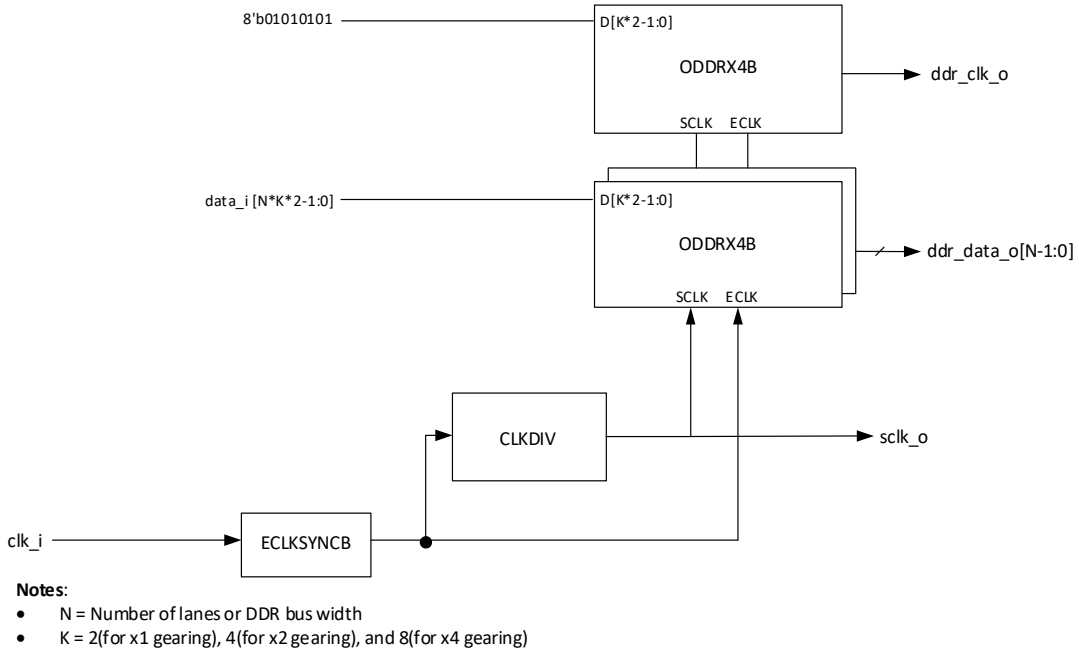


Figure 5.14. GDDRX4_TX.ECLK.Aligned Interface

Interface rules:

- Must use edge clock routing resources for the ECLK
- The routing of SCLK must use primary clock net
- Data output must use A/B pair of the I/O logic for ×4 gearing
- This interface is supported at the top side of the device

5.2.2.7. GDDRX4_TX.ECLK.Centered

This output DDR ×4 interface provides a clock that is centered at the data opening. The PLL uses clkop and clkos ports to provide the 90° phase difference between the data and the clock. Two ECLK routing resources are used in this interface to drive the output data and the output clock. A CLKDIV is used to generate the SCLK which is one-quarter of the ECLK frequency. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).

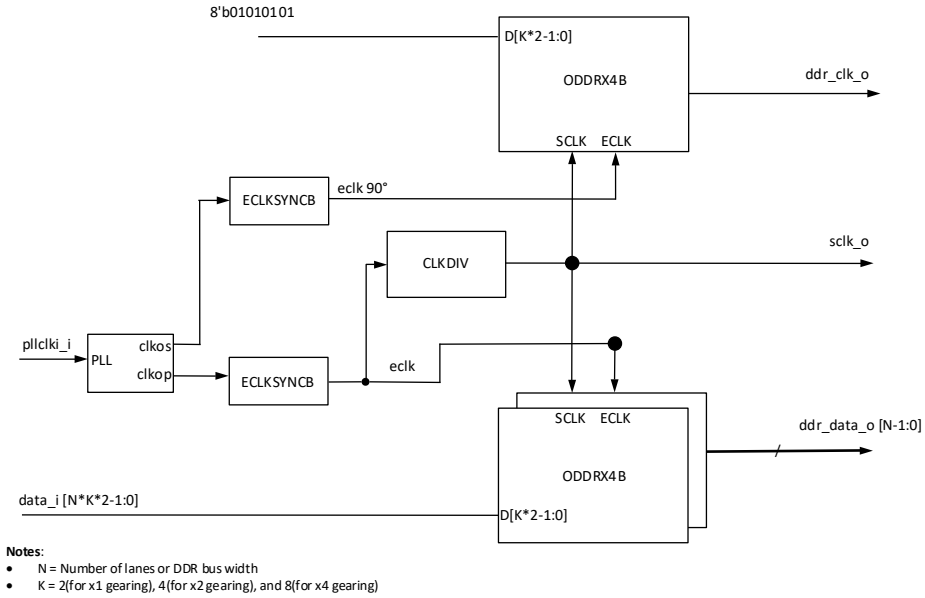


Figure 5.15. GDDRX4_TX.ECLK.Centered Interface

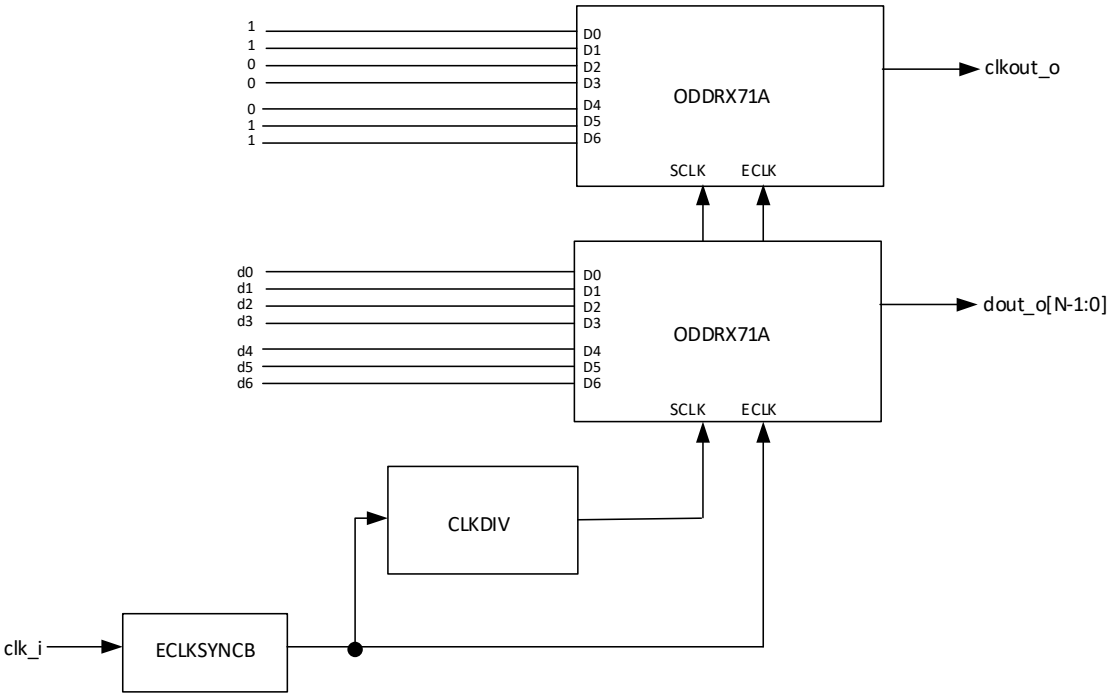
Interface rules:

- Must use edge clock routing resources for the ECLK
- Since two ECLKs are used on this interface, a maximum of one bus of this interface can be implemented at a time
- The routing of the SCLK must use primary clock net
- Data output must use A/B pair of the I/O logic for ×4 gearing
- This interface is supported at the top side of the device

Note: The GDDRX4_TX.ECLK.Centered interface is used to build MIPI D-PHY Receive Interface. Refer to MIPI D-PHY Interface IP (FPGA-RD-02040) for details. Note that, while this reference design was originally targeted for MachXO3 and ECP5 devices, the concept can still be applied for MachXO4, but additional steps may be required. Contact Technical Support Assistance for more details.

5.2.2.8. GDDR71_TX.ECLK.7:1

The GDDR 7:1 transmit interface is unique among the supported high-speed DDR interfaces. It uses a specific pattern to generate the output clock, known as pixel clock. The CLKDIVC is used to divide down the ECLK by 3.5 due to the nature of the 7:1 serializing requirement. This means the SCLK is running 7 times slower than the transmit data rate. ECLKSYNCB element is associated with the ECLK and must be used to drive the ECLK. For detailed information, refer to MachXO4 DDR 7:1 Module User Guide (FPGA-IPUG-02315).



Note: n = number of lanes. The range of n is from 1 to 21.

Figure 5.16. GDDR71_TX.ECLK.7:1 Interface

Interface rules:

- Must use edge clock routing resources for the ECLK
- The routing of SCLK must use primary clock net
- Data output must use A/B pair of the I/O logic for 7:1 gearing
- This interface is supported at the top side of the device

6. Using IP Catalog to Build High-Speed DDR Interfaces

The IP Catalog in Lattice Radiant software can be used to configure and generate all the generic high-speed interfaces described above. IP Catalog will generate a complete HDL module including clocking requirements for each of the interfaces described above.

All the DDR modules are located under **Architecture Modules > IO**. 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

6.1. Configuring the SDR Modules

As shown in Figure 6.1, in IP Catalog, choose and double-click SDR. This launches **Module/IP Block Wizard** as shown in Figure 6.2. Enter **Component Name** and choose the directory where the IP will be created. Click **Next** to proceed to SDR configuration window. For detailed information, refer to MachXO4 SDR Module User Guide (FPGA-IPUG-02317).

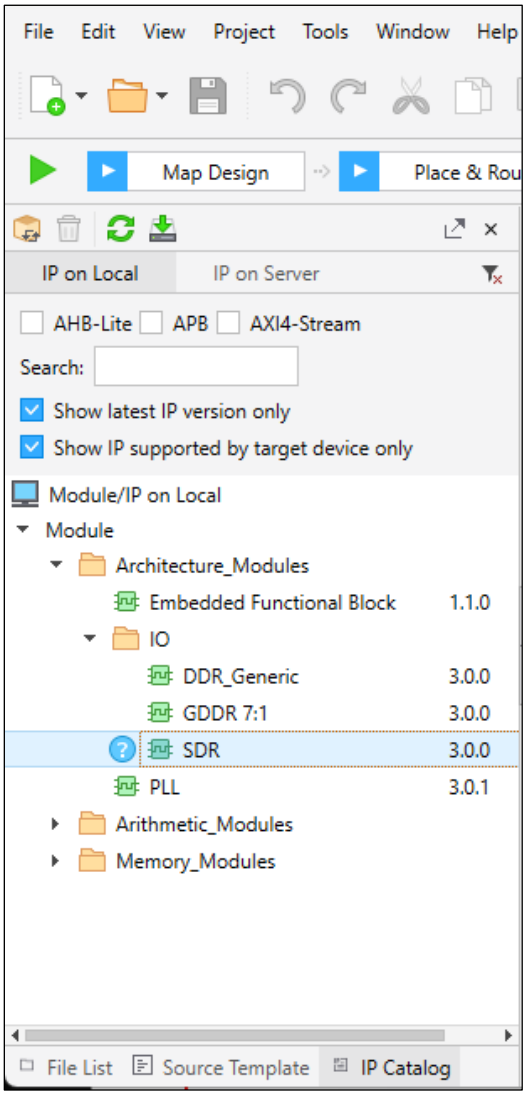


Figure 6.1. SDR Module Selection at the IP Catalog Main Window

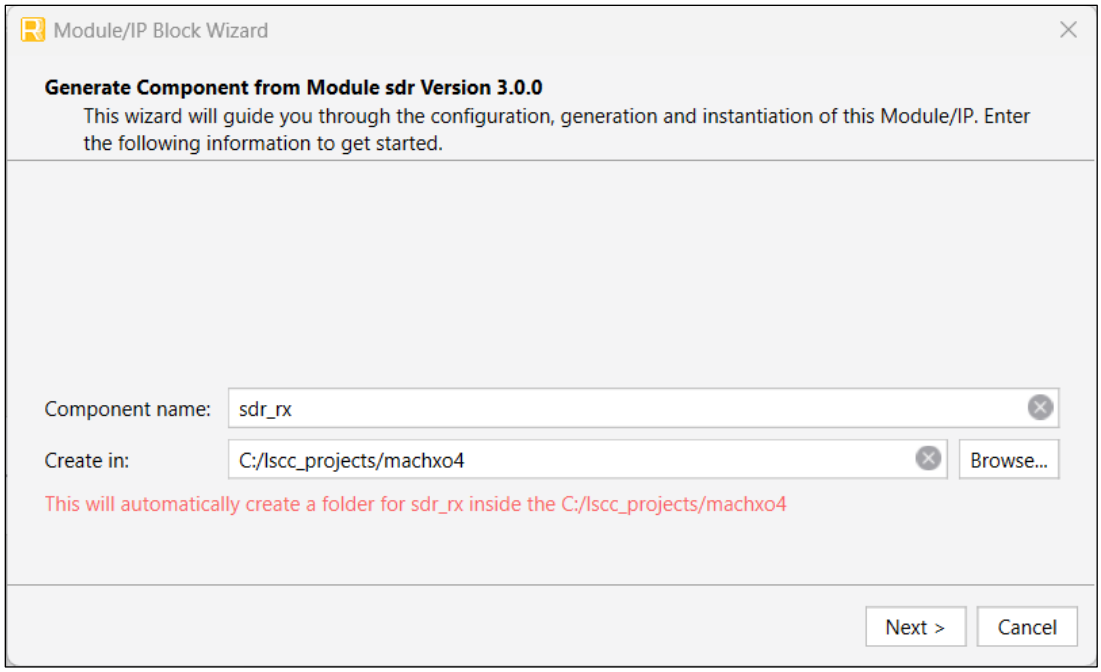


Figure 6.2. SDR Module Customization in Module/IP Block Wizard

Figure 6.3 shows the configuration window for the SDR Module in **Module/IP Block Wizard**.

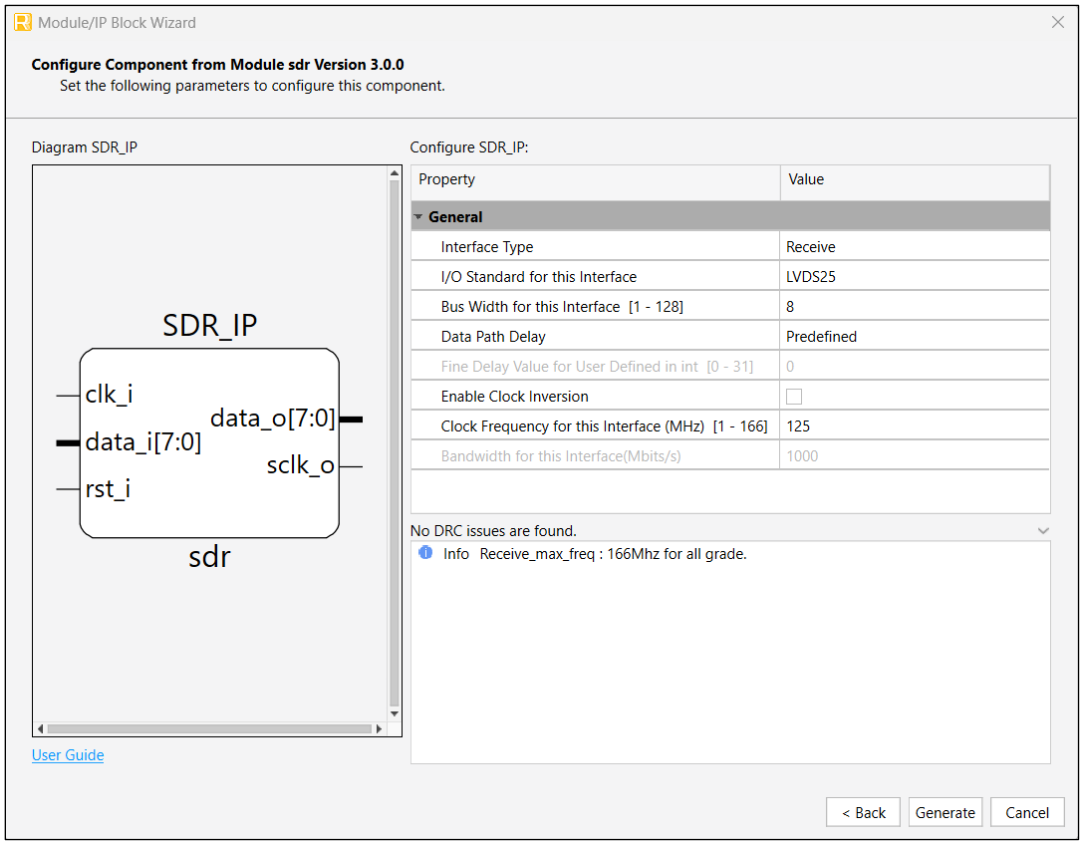


Figure 6.3. SDR Module Configuration in Module/IP Block Wizard

Table 6.1 lists the various configurations options available for SDR module. Wherever applicable, default values are in bold.

Table 6.1. SDR Module General Attributes

Attribute	Selectable Values	Default	Dependency on other Attributes
Interface Type	Receive, Transmit	**Receive**	—
I/O Standard for this Interface	Common: LVDS25, LVTTL33, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12. For Transmit only: LVDS25E, BLVDS25E, MLVDS25E, LVPECL33E, LVTTL33D, LVCMOS33D, LVCMOS25D. For Receive only: BLVDS25, MLVDS25, LVPECL33, LVTTL33D, LVCMOS33D, LVCMOS25D, LVCMOS10R25, LVCMOS10R33, LVCMOS12R25, LVCMOS12R33.	**LVDS25**	—
Bus Width for this Interface	1 – 128	**8**	—
Data Path Delay	Bypass Bypass, Predefined, User Defined	**Bypass** **Predefined**	*Interface Type* == *Transmit* *Predefined* and *User Defined*, only supported when *Interface Type* == *Receive*
Fine Delay Value for User Defined	0 – 31	**0**	*Data Path Delay* == *User Defined*
Enable Clock Inversion	Checked, Not checked	**Not checked**	*Interface Type* == *Receive*
Clock Frequency for this Interface (MHz)	1 – 166 1 – 150 1 – 125	**125** **125** **125**	*Interface Type* == *Receive* *Interface Type* == *Transmit* (grade-6 dev) *Interface Type* == *Transmit* (grade-5 dev) and below
Bandwidth for this Interface (Mbits/s)	Calculated	**1000**	Clock Frequency × Bus Width. Display for information only

Note: All attributes can be configured from the General tab of the Lattice Radiant Software user interface.

Table 6.2. SDR Module Attributes Description

Attribute Name	Description
Interface Type	Select the interface type, such as Receive or Transmit.
I/O Standard for this Interface	Select an I/O standard from the list of supported Single-ended and Differential types.
Bus Width for this Interface	Select the total number of bus lanes.
Data Path Delay	Selects between *Bypass*, *Predefined*, or *User Defined* modes.
Fine Delay Value for User Defined	Defines the fine delay setting for the data path. This setting is only applicable when the *Data Path Delay* mode is configured as *User Defined*. Provides 32 delay steps, from *DELAY0* to *DELAY31*. *DELAY0* equals 0 ps, while each subsequent step (*DELAY1* to *DELAY31*) adds 105 ps. The maximum fine delay is 31 × 105ps = 3.255 ns.
Enable Clock Inversion	Enables clock inversion for the *Receive* interface type only. When enabled, the sampling clock for received data is inverted.

Attribute Name	Description
Clock Frequency for this Interface (MHz)	Specifies the clock frequency used by the selected interface.

6.2. Configuring DDR Generic Modules

As shown in Figure 6.4, in IP Catalog, choose and double-click DDR_Generic, This will launch **Module/IP Block Wizard** as shown in Figure 6.2. Enter **Component Name** and choose the directory where the IP will be created. Click **Next** to proceed to DDR_Generic configuration window. For detailed information, refer to MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314).

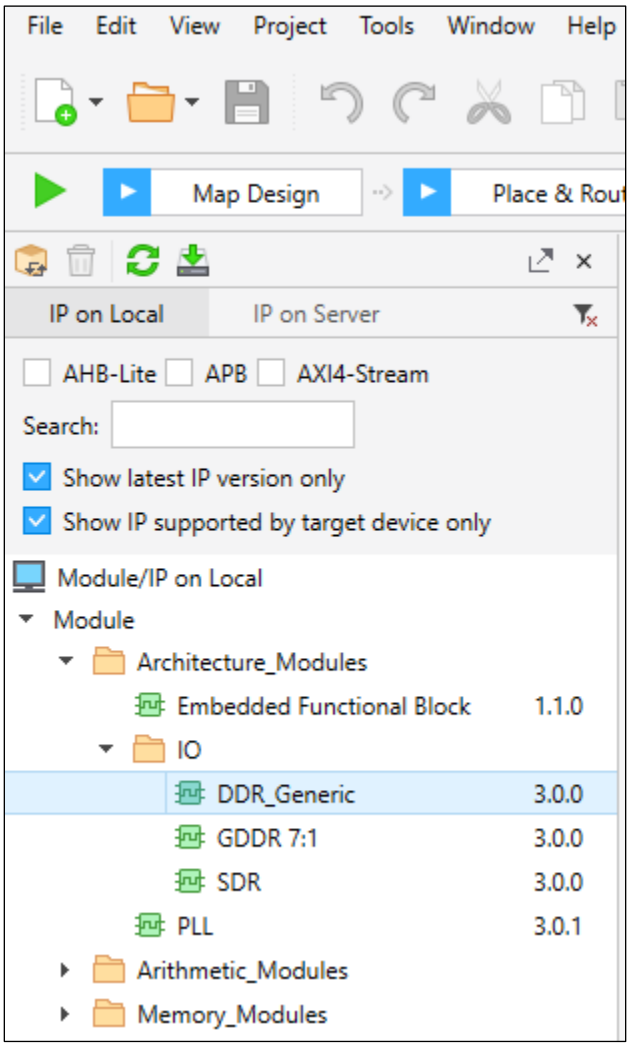


Figure 6.4. DDR_Generic Module Selection at the IP Catalog Main Window

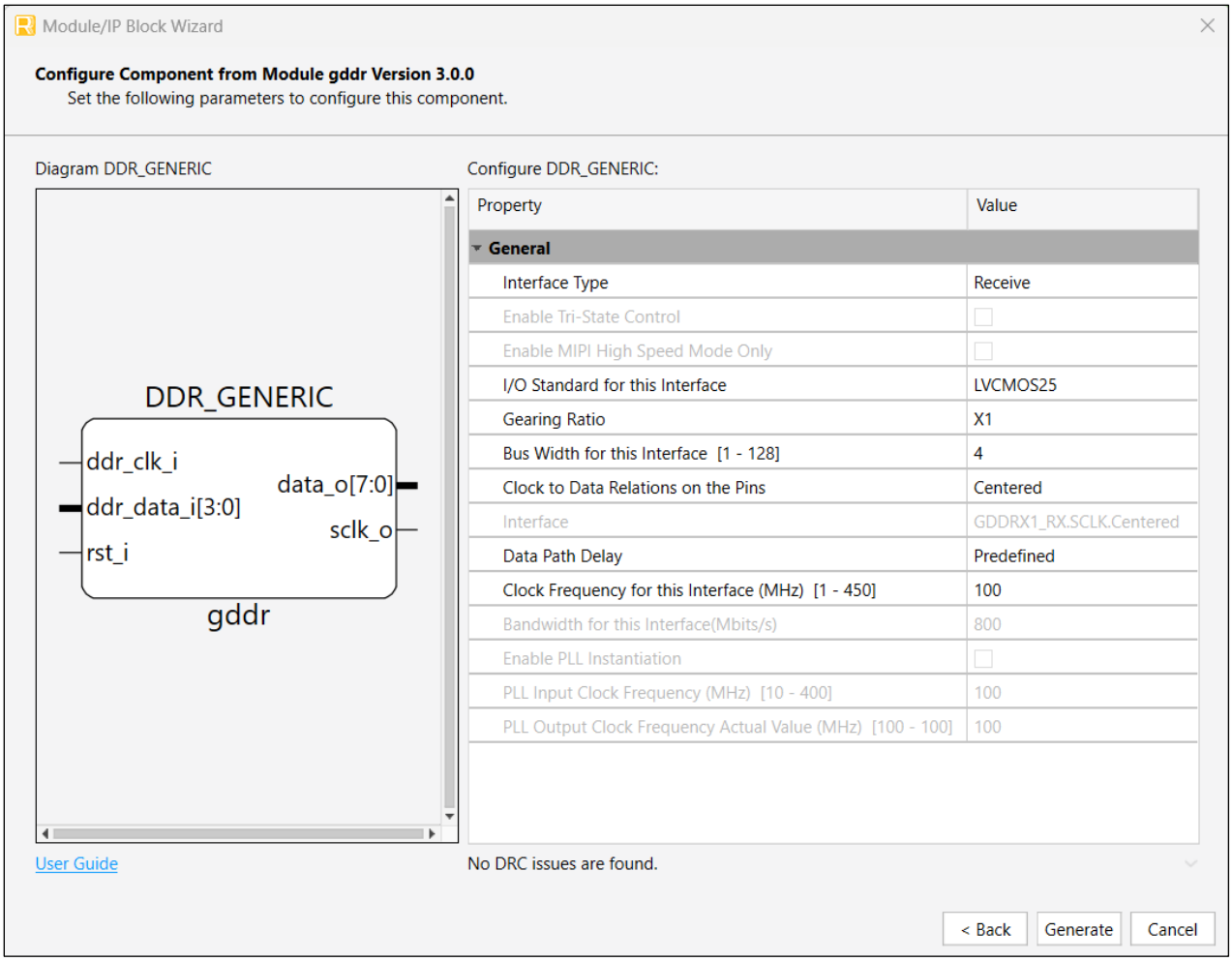


Figure 6.5. DDR Generic Module Configuration in Module/IP Block Wizard

Table 6.3 list the various configuration options available for DDR_Generic module. Wherever applicable, default values are in bold.

Table 6.3. DDR_Generic Module General Attributes

Attribute	Selectable Values	Description
Interface Type	• **Receive (Default)** • Transmit • Receive MIPI • Transmit MIPI	Specify the DDR interface type.
Enable Tri-state Control	• **Unchecked** • Checked	Enable to instantiate Tri-State I/O buffer on both DDR data and clock. **Dependency on other attributes:** Available when *(Interface Type) == Transmit or Transmit MIPI*
Enable MIPI High Speed Mode Only	• **Unchecked** • Checked	Enable to use only High Speed MIPI interface. **Dependency on other attributes:** Available when *(Interface Type) == Transmit MIPI or Receive MIPI*
I/O Standard for this Interface	(Legal Combination Table)	Refer to IP GUI.

Attribute	Selectable Values	Description
Gearing Ratio [K]	• **x1** • x2 • x4	Specify gearing ratio.
Bus Width for this Interface [N]	Range : 1 – 128 **Default : 4**	Specify the total number of lanes or bus width for DDR data interface. The maximum number of lanes varies with Gearing Ratio: x1 : 128 x2 : 32 x4 : 21
Clock to Data Relations on the Pins	• Edge-to-Edge • **Centered**	Specify the clock to data relationship of DDR interface at FPGA pins.
Interface	• GDDRX1_RX.SCLK.Aligned • **GDDRX1_RX.SCLK.Centered** • GDDRX1_TX.SCLK.Aligned • GDDRX1_TX.SCLK.Centered • GDDRX2_RX.ECLK.Aligned • GDDRX2_RX.ECLK.Centered • GDDRX2_TX.ECLK.Aligned • GDDRX2_TX.ECLK.Centered • GDDRX4_RX.ECLK.Aligned • GDDRX4_RX.ECLK.Centered • GDDRX4_TX.ECLK.Aligned • GDDRX4_TX.ECLK.Centered	For display information only.
Data Path Delay	• Bypass, • **Pre-defined** • User-defined • Dynamic	Specify the delay implementation on data path delay. For Transmit, this option is fixed to Bypass. **Bypass**: No delay component added DDR data path. For Receive, DDR data are fed directly to IDDR component. For Transmit, DDR data are sent out from FPGA directly. **Pre-defined**: Added delay component (DELAYH) with predefined value to delay the incoming DDR data before fed to IDDR block. **User-defined**: Added delay component (DELAYH) on incoming DDR data path and allow user to configure how much delay needed. **Dynamic**: Added dynamic delay component (DELAYG) on incoming DDR data path. User can dynamically control the delay value through signals.
Fine Delay Value for User-defined	Range : DELAY0 to DELAY31 **Default : DELAY0**	Specify the delay step apply on DELAYH. Each step is about ~105ps **Dependency on other attributes:** Available when *(Data Path Delay) == User-defined*
Clock Frequency for this Interface (MHz)	Range : 1-450 **Default : 100**	Specify the clock frequency of DDR interface
Bandwidth for this Interface (Mbits/s)	Derived	Bandwidth of this DDR interface. This option is only for display. **Formula**: 2*(Clock Frequency for this Interface) *(Bus Width for this Interface)

Attribute	Selectable Values	Description
Enable PLL Instantiation	• **Unchecked** • Checked	Enable to add PLL. This PLL is used to generate the DDR clock and its 90° phase shifted DDR clock. **Dependency on other attributes:** *Available when (Interface Type) == (Transmit || Transmit MIPI) && Clock to Data Relationship on the Pins == Centered*
PLL Input Clock Frequency (MHz)	Range :10 – 400 **Default : 100**	Specify the reference clock of PLL. **Dependency on other attributes:** Available when (*PLL Instantiation*) == Checked
PLL Output Clock Frequency Actual Value (MHz)	Derived	Only for display.

6.3. Configuring Generic DDR 7:1 Modules

As shown in Figure 6.6, in IP Catalog, choose and double-click GDDR 7:1, This will launch **Module/IP Block Wizard** as shown in Figure 6.2. Click **Next** to proceed to GDDR 7:1 configuration window. The DDR 7:1 interface is a very specific application, so the interface options are relatively simple. Most of the necessary logic is built into the software for ease of use. For detailed information, refer to MachXO4 DDR 7:1 Module User Guide (FPGA-IPUG-02315).

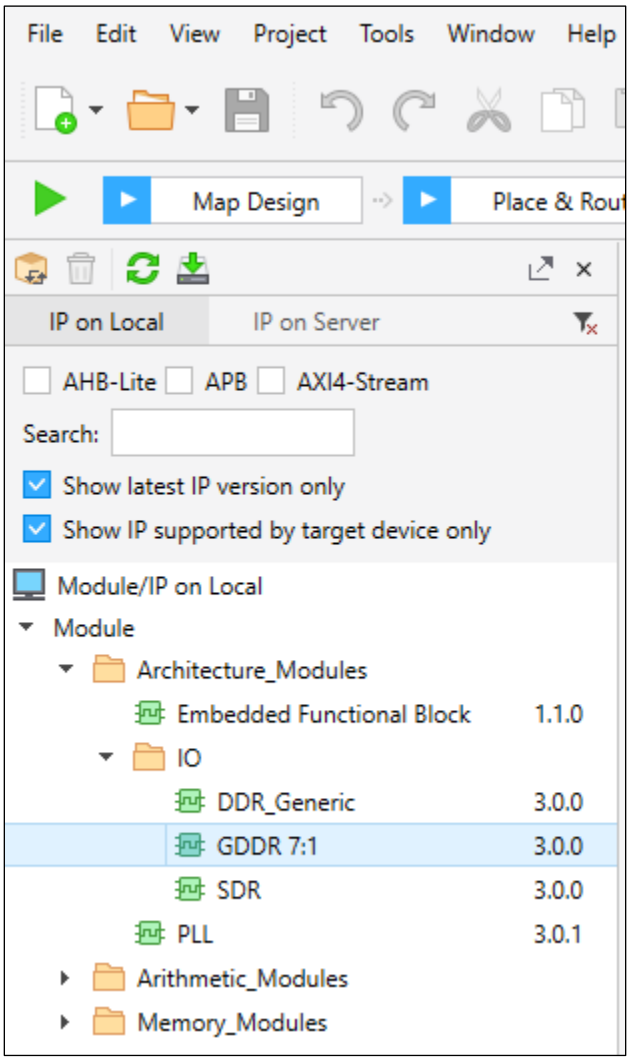


Figure 6.6. GDDR 7:1 Module Selection at the IP Catalog Main Window

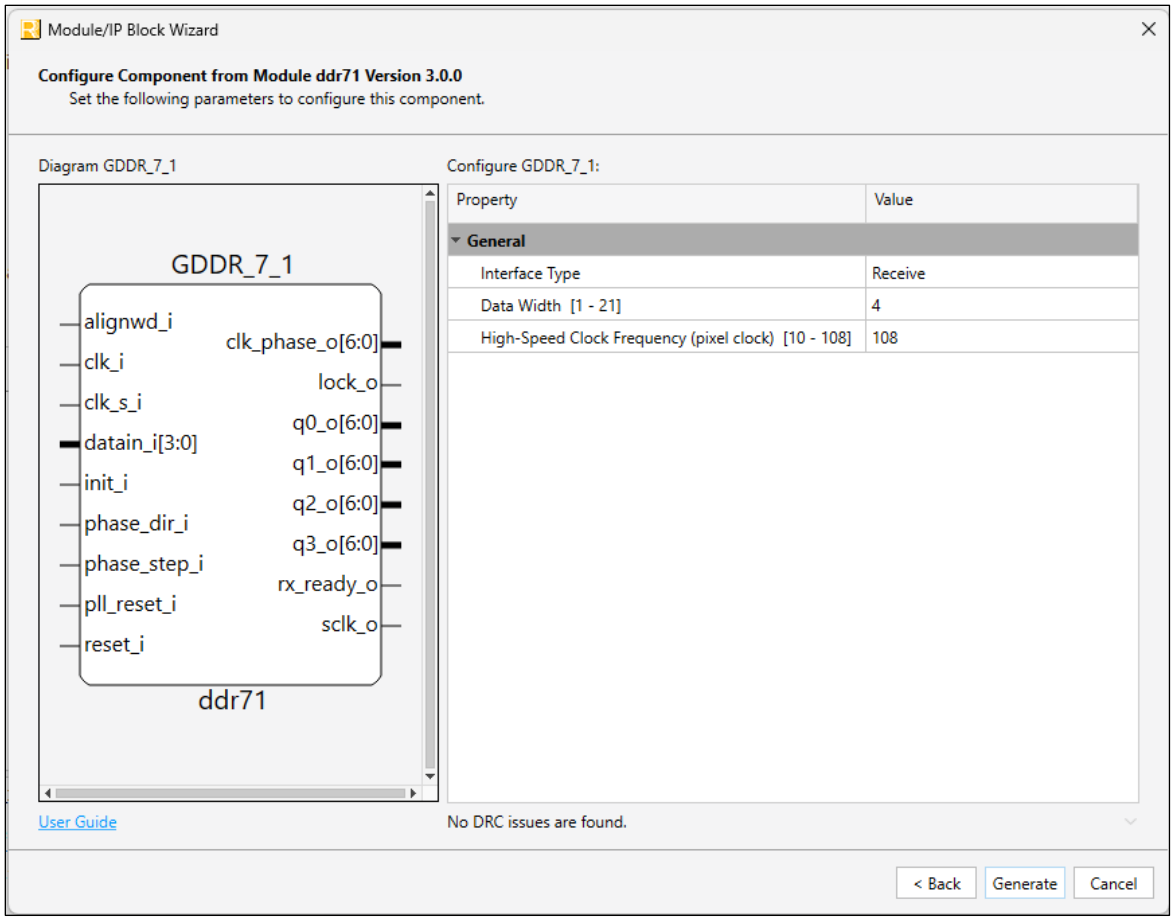


Figure 6.7. Configuration Tab of GDDR 7:1 Module

Table 6.4 list the various configuration options available for GDDR 7:1 module. Wherever applicable, default values are in bold.

Table 6.4. GDDR 7:1 General Attributes

Attribute	Selectable Values	Description
Interface type	Transmit, **Receive**	Type of interface
Data width	1-21	The number of incoming lanes (n). Default is **4.**
High Speed Clock Frequency (Pixel Clock)	10 MHz – **108 MHz**	Pixel clock frequency for GDDR 7:1 Module.

7. Generic High-Speed DDR Design Guidelines

7.1. I/O Logic Cells and Gearing Logic

Each Programmable I/O Cell (PIC) has four programmable I/Os (PIOs), which form two pairs of I/O buffers. Each PIO by itself can support a ×1 gearing ratio. A pair of PIOs, either the A/B pair or C/D pair, can support a ×2 gearing ratio. Support of a ×4 or 7:1 gearing ratio will take up all four PIOs in one PIC block. The ×4 or 7:1 gearing ratio can only be supported when the A/B pair pins are available in the package, and are independent of the availability of the C/D pins. The total number of ×2 interfaces available in a specific package is determined by the total number of A/B and C/D pairs. The total number of ×4/7:1 interfaces available in a specific package is determined by the total number of A/B pairs.

Table 7.1. Gearing Logic Supported by Mixed Mode of I/O Logic Cells for MachXO4 Devices

	I/O Logic A	I/O Logic B	I/O Logic C	I/O Logic D
×2 gearing (A/B pair)	IDDRX2 or ODDRX2	Not available	Basic I/O registers or ×1 gearing	Basic I/O registers or ×1 gearing
×2 gearing (C/D pair)	Basic I/O registers or ×1 gearing	Basic I/O registers or ×1 gearing	IDDRX2 or ODDRX2	Not available
×4 gearing	IDDRX4 or ODDRX4	Not available	Basic I/O registers or ×1 gearing	Basic I/O registers or ×1 gearing
7:1 gearing	IDDRX71 or ODDRX71	Not available	Basic I/O registers or ×1 gearing	Basic I/O registers or ×1 gearing

7.2. High-Speed ECLK Bridge

The high-speed ECLK bridge is used to enhance communication of ECLKs across the MachXO4 device and is mainly used for high-speed video applications. The bridge allows a clock source to drive the edge clocks on the top and bottom edges of the device with minimal skew. The inputs to the bridge include primary clock pins from the top and bottom sides, PLL outputs from both sides, and clock tree routings.

Two bridge muxes are available in the ECLK bridge: one for each ECLK on the same side of the device. These muxes allow dynamic switching between two edge clocks. Refer to MachXO4 sysCLOCK PLL Design and Usage Guide (FPGA-TN-02391), for ECLK bridge connectivity details.

The ECLK bridge supports all the generic high-speed interfaces except the high-speed ×2 and ×4 receive interfaces. The ECLK bridge component must be instantiated in the design to use the bridge function or to use it for routing purpose.

7.2.1. Reset Synchronization Requirement

The generic DDR interfaces are built with multiple dedicated circuits that are optimized for high-speed applications. It is therefore necessary to make sure all the components, such as CLKDIV and IDDR/ODDR, start with the same high-speed edge clock cycle to maintain the clock domain crossing margin between ECLK and SCLK, and to avoid bus bit-order scrambling due to the various delays of the reset pulse.

The ECLKSYNCB component and a particular reset sequence are required to guarantee a successful system implementation for interfaces using ×2, ×4, and 7:1 gearings. Figure 7.1 and Figure 7.2 show the timing requirements for receive interfaces and transmit interfaces. The RX_STOP or TX_STOP signal must be connected to the STOP port of the ECLKSYNCB component. The RX_RST or the TX_RST should be connected to the reset port of the ODDR/IDDR and CLKDIV components. The RX_ECLK or TX_ECLK are the outputs of the ECLKSYNCB components. It is necessary to have a minimum of two ECLK cycles between the RST and STOP signals as shown in the figures below. The receive interface reset process should not start until the transmit interface reset process is complete in a loopback implementation. The clock signal used to generate the minimum two ECLK delay can be any clock that is slower than the ECLK frequency.

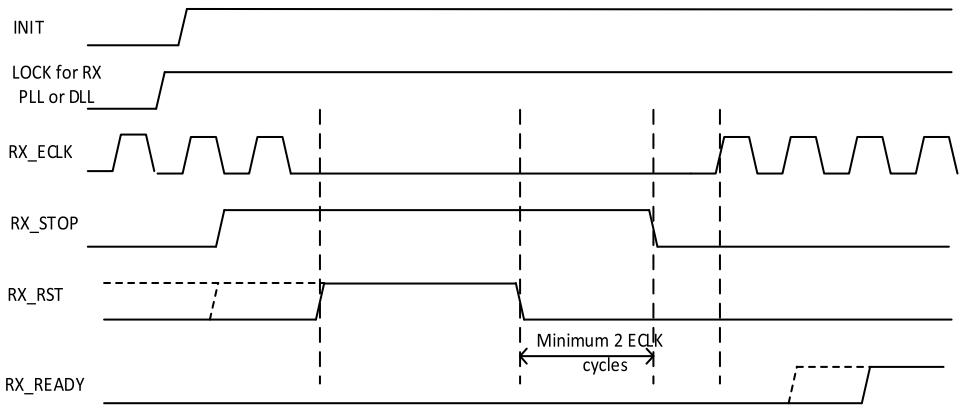


Figure 7.1. Reset Synchronization for Receive Interfaces

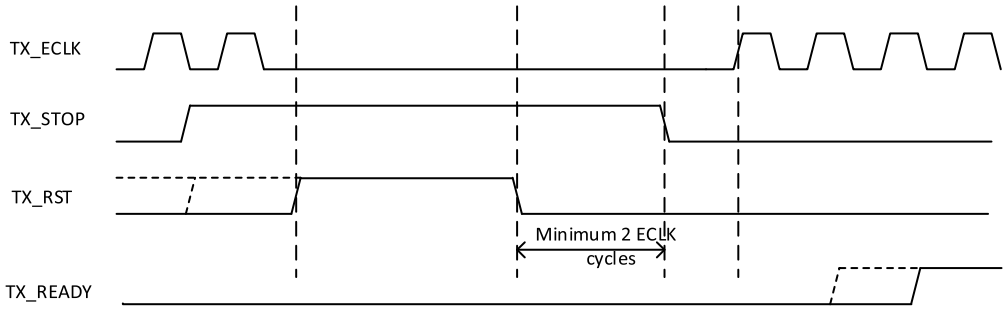


Figure 7.2. Reset Synchronization for Transmit Interfaces

These timing requirements are built into the generic DDR ×2/×4/7:1 modules when they are generated by IP Catalog. The RX_STOP/TX_STOP, RX_RST/TX_RST, and RX_ECLK/TX_ECLK are internal signals when the modules are generated by IP Catalog. The reset timing requirements must be followed and implemented in RTL code when the generic DDR interfaces are built outside of IP Catalog.

8. DDR Software Primitives and Attributes

Software primitives used for all the generic DDR interfaces implementation are discussed in this section. The primitives are divided according to their usage. Some of them are used for generic DDR interfaces and others are control functions. The DDR input primitives will be discussed first, followed by the DDR output primitives, then the DDR control logic primitives.

Table 8.1. MachXO4 DDR Software Primitives

Type	Primitive	Usage
Data Input	IDDRXE	Generic DDR ×1
	IDDRX2E	Generic DDR ×2
	IDDRX4B	Generic DDR ×4
	IDDR71A	Generic DDR 7:1
Data Output	ODDRXE	Generic DDR ×1
	ODDRX2E	Generic DDR ×2
	ODDRX4B	Generic DDR ×4
	ODDR71A	Generic DDR 7:1
DLL	DQSDLLC	Controller DLL for Generic ×2 and ×4
Input Delay	DELAYG	Delay block with dynamic control for Generic ×2, ×4
	DELAYH	Delay block with fixed delays for Generic DDR ×1, ×2, ×4
	DLLDELC	Clock target delay cell for Generic DDR ×2, ×4

8.1. Input DDR Primitives

The input DDR primitives represent the modules used to capture both the GDDR data. There are several modes for the DDR input registers to implement different gearings for GDDR interfaces. For all the data ports of the input primitives, Q0 of the parallel data is the first bit received.

8.1.1. IDDRXE

The primitive implements the input register block in ×1 gearing mode. This is used only for the generic DDR ×1 interface (gearing ratio 1:2) available on all sides of the MachXO4 devices. It uses a single clock source, SCLK, for the entire primitive so it does not involve a clock domain transfer.

Figure 8.1. IDDRXE Symbol

The internal register structure for this primitive is based on the basic PIO cell, as shown in Figure 2.1. The first set is the DDR register to capture the data at both edges of the SCLK. The second set is the synchronization registers to transfer the captured data to the FPGA core.

8.1.2. IDDRX2E

The primitive implements the input register block in ×2 gearing mode. This is used only for the generic DDR ×2 interface (gearing ratio 1:4) on the bottom side of the device. Its registers are designed to use edge clock routing on the GDDR interface and the system clock for FPGA core. The edge clock is connected to the ECLK port, while the system clock is connected to the SCLK port. This primitive can be used on both A/B or C/D pairs of I/O cells at the bottom side of the device.

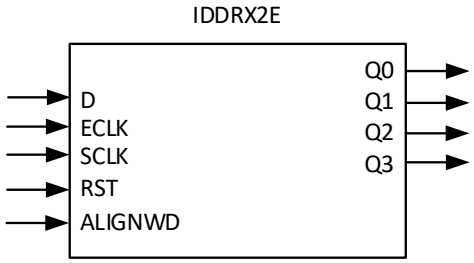


Figure 8.2. IDDRX2E Symbol

The internal register structure for this primitive is based on the video PIO cell, as shown in receive path of Figure 2.2.

The first set of the registers is the DDR register to capture the data at both edges of the ECLK. The second set is the synchronization registers to hold the data ready for clock domain transfer. The third set of registers performs the clock domain transfer from ECLK to SCLK.

The IDDRX2E primitive contains a fundamental logic error. Odd word alignments (SEL = 1) do not present the correct data at the Q3 output port. You must be aware of this behavior if instantiating the primitive directly into your design, and take appropriate steps to correct the temporal misalignment.

In Lattice Radiant, the temporal misalignment is corrected when using the Generic IDDRX2 logic interfaces (GDDRX2_RX.ECLK.Centered/Aligned) generated with the IP Catalog, and which are described in the Generic High-Speed DDR Interfaces section.

8.1.3. IDDRX4B

The primitive implements the input register block in ×4 gearing mode. This is used only for the generic DDR ×4 interface (gearing ratio 1:8) on the bottom side of the device. Its registers are designed to use edge clock routing on the GDDR interface and the system clock on the FPGA side. The edge clock is connected to the ECLK port, while the system clock is connected to the SCLK port. This primitive can only be used on the A/B pair of the I/O cells at the bottom side of the device.

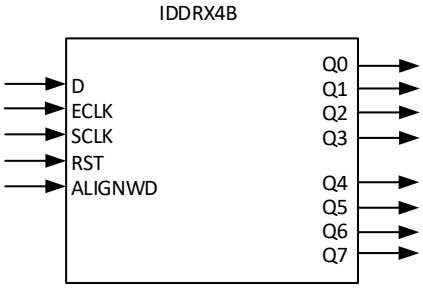


Figure 8.3. IDDRX4B Symbol

The 1:8 gearing of IDDRX4B uses two of the 1:4 gearing and shares the basic architecture with IDDRX2E. The internal register structure for this primitive is based on the video PIO cell, as shown in receive path of Figure 2.2. The first set of registers is the DDR register to capture the data at both edges of the ECLK. The second set of registers is synchronization registers to hold the data ready for clock domain transfer. The third set of registers performs the clock domain transfer from ECLK to SCLK.

The IDDRX4B primitive contains a fundamental logic error. Odd word alignments (SEL = 1) do not present the correct data at the Q7 output port. You must be aware of this behavior if instantiating the primitive directly into your design, and take appropriate steps to correct the temporal misalignment.

In Radiant, the temporal misalignment is corrected when using the Generic IDDRX4 logic interfaces (GDDRX4_RX.ECLK.Centered/Aligned) generated with the IP Catalog, which are described in the Generic High-Speed DDR Interfaces section.

8.1.4. IDDRX71A

The primitive implements the input register block in 7:1 gearing mode. This is used only for the generic DDR 71 interface (gearing ratio 1:7) on the bottom side of the device. Its registers are designed to use edge clock routing on the GDDR interface and the system clock on the FPGA side. The edge clock is connected to the ECLK port, while the system clock is connected to the SCLK port. This primitive, like the input ×4 gearing primitive, can only be used on the A/B pair of the PIO cell at the bottom side of the device.

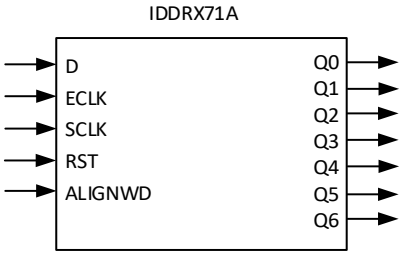


Figure 8.4. IDDRX71A Symbol

The 1:7 gearing of IDDRX71A shares the same architecture as the 1:8 gearing of the IDDRX4B primitive. It depends on an internal control signal to select three bits or four bits data at a time. The internal register structure for this primitive is based on the video PIO cell, as shown in the receive path of Figure 2.2. The first set of the registers is the DDR register to capture the data at both edges of ECLK. The second set is the synchronization registers to hold the data ready for clock domain transfer. The third set of registers performs the clock domain transfer from ECLK to SCLK.

8.2. Output DDR Primitives

The output DDR primitives represent the output DDR module used to multiplex two data streams before sending them out to the GDDR interface.

8.2.1. ODDRXE

This primitive will implement the output register block in ×1 gearing mode. It can be used in all sides of the MachXO4 devices. A single primary clock source, SCLK from the FPGA core, is used for this primitive.

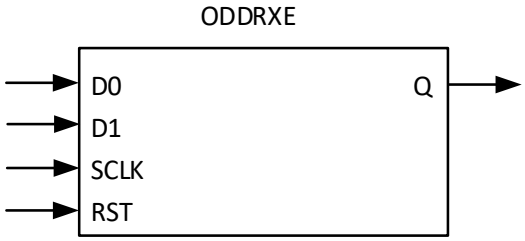


Figure 8.5. ODDRXE Symbol

The internal register structure for this primitive is based on the basic PIO cell, as shown in Figure 2.1. The SCLK is used to multiplex between the 2-bit parallel data to generate a serial data stream.

8.2.2. ODDRX2E

The primitive implements the output register block in ×2 gearing mode. This is used only for the generic DDR ×2 interface (gearing ratio 4:1) on the top side of the device. Its registers are designed to use the system clock on the FPGA side and the edge clock at the DDR interface. The edge clock is connected to ECLK port, while the system clock is connected to SCLK port. This primitive can be used on both the A/B or C/D pairs of PIO cells at the top side of the device.

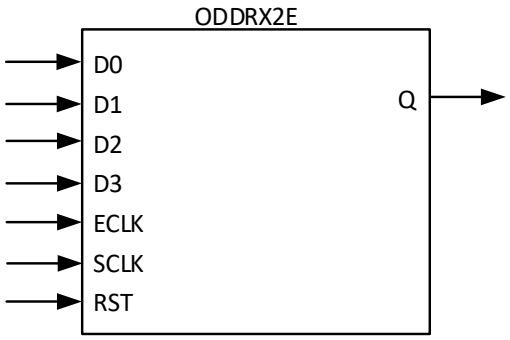


Figure 8.6. ODDRX2E Symbol

The internal register structure for this primitive is based on the video PIO cell, as shown in transmit path of Figure 2.2. The parallel data is registered by SCLK at the first set of registers. At each update, the parallel data is clocked in by SCLK and is held by the second set of registers. The third set of registers has the data ready to be multiplexed out as serial data by the ECLK.

8.2.3. ODDRX4B

The primitive implements the output register block in ×4 gearing mode. This is used only for the generic DDR ×4 interface (gearing ratio 8:1) on the top side of the device. Its registers are designed to use the system clock on the FPGA side and the edge clock at the GDDR interface. The edge clock is connected to the ECLK port, while the system clock is connected to the SCLK port. This primitive can only be used on the A/B pair of I/O cells at the top side of the device.

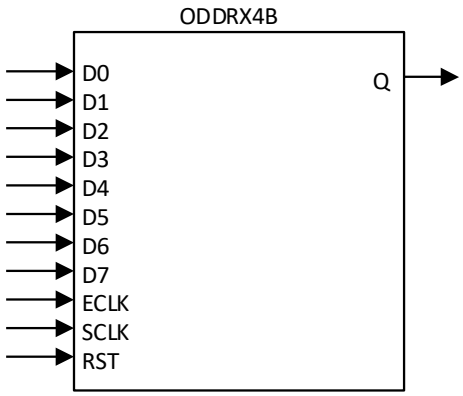


Figure 8.7. ODDRX4B Symbol

The internal register structure for this primitive is based on the video PIO cell, as shown in the transmit path of Figure 2.2. The parallel data is registered by SCLK at the first set of registers. At each update, the parallel data is clocked in by SCLK and is held by the second set of registers. The third set of registers has the data ready to be multiplexed out as serial data by ECLK.

8.2.4. ODDRX71A

The primitive implements the output register block in 7:1 gearing mode. This is used only for the generic DDR 71 interface (gearing ratio 7:1) on the top side of the device. Its registers are designed to use the system clock on the FPGA side and the edge clock routing on the GDDR interface. The edge clock is connected to the ECLK port, while the system clock is connected to the SCLK port. This primitive can only be used on the A/B pair of I/O cells at the top side of the device.

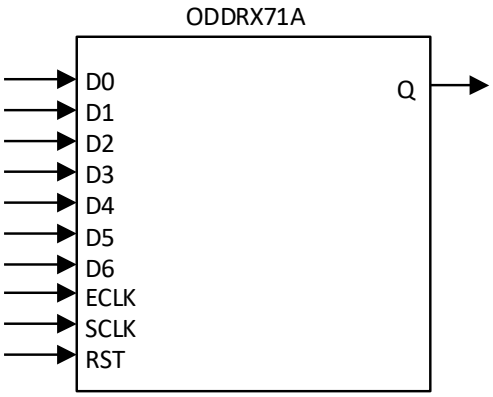


Figure 8.8. ODDRX71A Symbol

The 7:1 gearing of ODDRX71A shares the same architecture as the 8:1 gearing of the ODDRX4B primitive. It depends on the internal control signal to select the three or four bits of data at a time for transmission. The internal register structure for this primitive is based on the video PIO cell, as shown in the transmit path of Figure 2.2. The parallel data is registered by SCLK at the first set of registers. At each update, the parallel data is clocked in by SCLK and is held by the second set of registers. The third set of registers has the data ready to be multiplexed out as serial data by ECLK.

8.3. DDR Control Logic Primitives

The DDR primitives discussed below include the DLL and the delay elements. The delay elements are for data paths or the clock target delay paths.

8.3.1. DQSDLLC

The DQSDLLC is the on-chip DLL, which generates the 90° phase shift required for the DQS signal. Only one DQSDLLC can be used for the DDR implementations on one-half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DQSDLLC generates the delay based on this clock frequency and the update control input to this block. The DQSDLLC updates the dynamic delay control code (DQSDEL) to the DLLDELC block when this update control (UDDCNTLN) input is asserted. Otherwise, the update will be in the hold condition. The active low signal on UDDCNTLN updates the clock phase using DQSDEL delay.

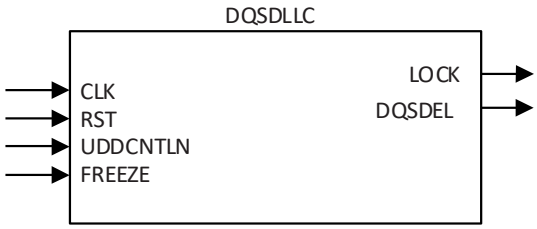


Figure 8.9. DQSDLLC Symbol

Table 8.2. DQSDLLC Signals

Signal	I/O	Description
CLK	I	Input clock to the DLL, same frequency as DDR interface
RST	I	DLL reset control
UDDCNTLN	I	Update/hold control to delay code before adjustment. Active low signal updates the delay code.
FREEZE	I	Use to freeze or release DLL input CLK
LOCK	O	DLL lock signal
DQSDEL	O	DLL delay control code to target delay

The DQS delay can be updated for PVT variation using the UDDCNTLN input. The DQSDEL is updated when the UDDCNTLN is held low. The DQSDEL can be updated when variations are expected. It can be updated anytime except during a READ or WRITE operation.

The FREEZE input port of this component is used to freeze or release the DLL. When FREEZE goes high, the device will freeze the DLL to save power while the delay code is preserved. When FREEZE goes low, it will release the DLL to resume operation. FREEZE must be applied to the DQSDLLC before the clock stops.

By default, this DLL will generate a 90° phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. You can control the sensitivity to jitter by using the LOCK_SENSITIVITY attribute. This configuration bit can be programmed to be either HIGH or LOW. Lock_sensitivity HIGH means more sensitive to jitter. It is recommended that the bit be programmed LOW.

The DQSDLLC supports a wide range of frequencies up to 400 MHz. The FIN attribute associated with this primitive allows you to set the DLL frequency. It is possible to bypass the DLL locking process when the frequency becomes very low. The attribute FORCE_MAX_DELAY can be used for this purpose. When FORCE_MAX_DELAY is set in the software, the DLL will not go through the locking process. Instead, DLL will be locked to maximum delay steps. The effect of the FORCE_MAX_DELAY attribute will not be reflected in the simulation model. The simulation model always models the 90° phase shift for DLL. Refer to MachXO4 Family Data Sheet (FPGA-DS-02125) for the range of frequencies when the FORCE_MAX_DELAY becomes effective.

Table 8.3. Attribute for DQSDLLC

Attribute	Description	Values	Software Default
LOCK_SENSITIVITY	Jitter sensitivity	HIGH, LOW	LOW
FIN	Input clock frequency of DLL	Range supported by DLL	100 MHz
FORCE_MAX_DELAY	Bypass DLL locking procedure at low frequency and sets the maximum delay setting	YES, NO	NO

8.3.2. DELAYH

Input data going to the DDR registers can optionally be delayed using the delay block, DELAYH. The 32-tap DELAYH block is used to compensate for clock injection delay times. The amount of the delay is determined by the software based on the type of interface implemented using the DEL_MODE attribute. You are allowed to set the delay by choosing the USER_DEFINED mode for the block. When in USER_DEFINED mode, you must manually set the number of delay steps to be used. Each delay stay would generate ~105ps of delay. It is recommended to use the PREDEFINED mode for all generic DDR interfaces. If an incorrect attribute value is used for a given interface, the DELAYH setting will be incorrect, and the performance of the DDR interface will not be optimal. The DELAYH block is applicable to the receive mode of the DDR interfaces. It is available for all input register paths at all sides of a MachXO4 device.

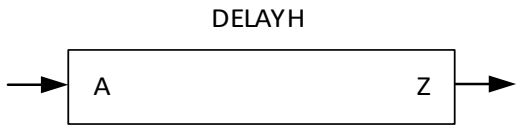


Figure 8.10. DELAYH Symbol

Table 8.4. DELAYH Signals

Signal	I/O	Description
A	I	DDR input from sysIO buffer
Z	O	Output with delay

Table 8.5. DELAYH Attributes

Attribute	Description	Values	Software Default
DEL_MODE	Fixed delay value depending on interface and user-defined delay values	SCLK_ZEROHOLD ECLK_ALIGNED ECLK_CENTERED SCLK_ALIGNED SCLK_CENTERED USER_DEFINED	USER_DEFINED
DEL_VALUE	User-defined value	DELAY0…DELAY31	DELAY0

Table 8.6. DEL_MODE Values Corresponding to the GDDR Interface

Interface Name	DEL_MODE Values
GIREG_RX.SCLK	SCLK_ZERHOLD
GDDRX1_RX.SCLK.Aligned	SCLK_ALIGNED
GDDRX1_RX.SCLK.Centered	SCLK_CENTERED
GDDRX2_RX.ECLK.Aligned	ECLK_ALIGNED
GDDRX2_RX.ECLK.Centered	ECLK_CENTERED
GDDRX4_RX.ECLK.Aligned	ECLK_ALIGNED
GDDRX4_RX.ECLK.Centered	ECLK_CENTERED
GDDRX71_RX.ECLK.71	Bypass
GOREG_TX.SCLK	N/A
GDDRX1_TX.SCLK.Centered	N/A
GDDRX1_TX.SCLK.Aligned	N/A
GDDRX2_TX.ECLK.Aligned	N/A
GDDRX2_TX.ECLK.Centered	N/A
GDDRX4_TX.ECLK.ALIGNED	N/A
GDDRX4_TX.ECLK.CENTERED	N/A
GDDRX_TX.ECLK.7:1	N/A

8.3.3. DELAYG

At the bottom side of the device, input data going to the DDR registers can also be delayed by the DELAYG block. Unlike the DELAYH block where the delay is determined during the operation of the device, the DELAYG block allows you to control the amount of data delay while the device is in operation. This block receives 5-bit (32 taps) delay control. The 5-bit delay is dynamically controlled by the user logic through the delay port. Each delay step would generate ~105 ps of delay.

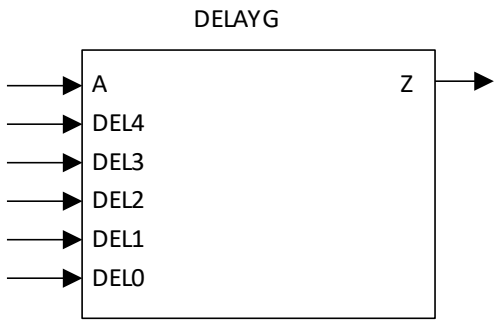


Figure 8.11. DELAYG Symbol

Table 8.7. DELAYG Signals

Signal	I/O	Description
A	I	DDR input from I/O buffer
DEL4, DEL3, DEL2, DEL1, DEL0	I	Dynamic delay input port from FPGA logic
Z	O	Output with delay

8.3.4. DLLDELC

This is the clock target delay cell, which is used to generate a 90° delay in all receive aligned interfaces. The 90° delay is calculated based on the input clock to the DQSDLLC element. The amount of delay required is based on the delay control code, DQSDEL, generated from the DQSDLLC.

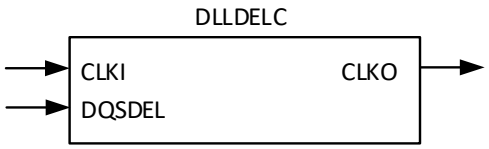


Figure 8.12. DLLDELC Symbol

Table 8.8. DLLDELC Signals

Signal	I/O	Description
CLKI	I	Data Input from I/O buffer
DQSDEL	I	Dynamic delay inputs from DQSDLLC
CLKO	O	Output with delay

References

- MachXO4 Family Data Sheet (FPGA-DS-02125)
- MachXO4 sysCLOCK PLL Design and Usage Guide (FPGA-TN-02391)
- MIPI D-PHY Interface IP (FPGA-RD-02040)
- MachXO4 SDR Module User Guide (FPGA-IPUG-02317)
- MachXO4 DDR Generic Module User Guide (FPGA-IPUG-02314)
- MachXO4 DDR 7:1 Module User Guide (FPGA-IPUG-02315)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- MachXO4 web page
- Lattice Radiant Software web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 1.0, December 2025

Section	Change Summary
All	Initial release

www.latticesemi.com