



# **MachXO4 DDR Generic Module**

IP Version: v3.0.0

## **User Guide**

FPGA-IPUG-02314-1.0

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

Contents .....	3
Abbreviations in This Document.....	5
1. Introduction .....	6
1.1. Overview of the IP .....	6
1.2. Quick Facts .....	6
1.3. Features .....	6
1.4. Licensing and Ordering Information .....	7
1.5. Minimum Device Requirements .....	7
1.6. Naming Conventions .....	7
1.6.1. Nomenclature.....	7
1.6.2. Signal Names .....	7
2. Functional Description.....	8
2.1. IP Architecture Overview .....	8
2.1.1. GDDR1_RX.SCLK.Aligned .....	10
2.1.2. GDDR1_RX.SCLK.Centered .....	11
2.1.3. GDDR2/4_RX.ECLK.Aligned .....	12
2.1.4. GDDR2/4_RX.ECLK.Centered.....	14
2.1.5. GDDR4_RX.MIPI .....	15
2.1.6. GDDR1_TX.SCLK.Aligned .....	16
2.1.7. GDDR1_TX.SCLK.Centered .....	17
2.1.8. GDDR2/4_TX.ECLK.Aligned.....	18
2.1.9. GDDR2/4_TX.ECLK.Centered .....	20
2.1.10. GDDR4_TX.MIPI.....	21
2.2. Clocking .....	22
2.2.1. Receive Interfaces Clocking .....	22
2.2.2. Transmit Interfaces Clocking .....	23
2.3. Reset .....	24
2.3.1. Reset Sequence for Receive Interfaces .....	24
2.3.2. Reset Sequence for Transmit Interfaces .....	25
3. IP Parameter Description.....	26
3.1. General.....	26
4. Signal Description .....	28
5. Designing with the IP .....	32
5.1. Generating and Instantiating the IP .....	32
5.1.1. Generated Files and File Structure .....	34
5.2. Design Implementation .....	34
5.3. Timing Constraints .....	34
5.4. Physical Constraints .....	35
5.5. Specifying the Strategy.....	35
5.6. Running Functional Simulation .....	35
5.6.1. Simulation Results .....	37
Appendix A. Resource Utilization .....	38
References .....	39
Technical Support Assistance .....	40
Revision History.....	41

## Figures

Figure 2.1. GDDR I/O Module Block Diagram .....	8
Figure 2.2. GDDR <sub>X1</sub> _RX.SCLK.Aligned Block Diagram .....	10
Figure 2.3. DDR Data Output Arrangement for GDDR <sub>X1</sub> .RX.SCLK.Aligned Interface .....	10
Figure 2.4. GDDR <sub>X1</sub> _RX.SCLK.Centered Block Diagram .....	11
Figure 2.5. GDDR <sub>X2/4</sub> _RX.ECLK.Aligned Block Diagram .....	12
Figure 2.6. Data output order for GDDR <sub>X2</sub> .RX.ECLK.Aligned Interface .....	12
Figure 2.7. Data output order for GDDR <sub>X4</sub> .RX.ECLK.Aligned interface .....	13
Figure 2.8. GDDR <sub>X2/4</sub> _RX.ECLK.Centered Block Diagram .....	14
Figure 2.9. GDDR <sub>X4</sub> _RX.MIPI Block Diagram .....	15
Figure 2.10. GDDR <sub>X1</sub> _TX.SCLK.Aligned Block Diagram .....	16
Figure 2.11. Data Output Order for GDDR <sub>X1</sub> .TX.SCLK.Aligned Interface .....	16
Figure 2.12. GDDR <sub>X1</sub> _TX.SCLK.Centered Block Diagram .....	17
Figure 2.13. GDDR <sub>X2/4</sub> _TX.ECLK.Aligned Block Diagram .....	18
Figure 2.14. Data Output Order for GDDR <sub>X2</sub> .TX.ECLK.Aligned Interface .....	18
Figure 2.15. Data Output Order for GDDR <sub>X4</sub> .TX.ECLK.Aligned Interface .....	19
Figure 2.16. GDDR <sub>X2/4</sub> _TX.ECLK.Centered Block Diagram .....	20
Figure 2.17. GDDR <sub>X4</sub> _TX.MIPI Block Diagram .....	21
Figure 2.18. Clocking Network for Receive Interfaces .....	22
Figure 2.19. Clocking Network for Transmit Interfaces .....	23
Figure 2.20. Reset Sequence of X1 Receive Interfaces .....	24
Figure 2.21. Reset Sequence of X2/X4 Receive Interfaces .....	24
Figure 2.22. Reset Sequence of X1 Transmit Interfaces .....	25
Figure 2.23. Reset Sequence of X2/X4 Transmit Interfaces .....	25
Figure 5.1. Module/IP Block Wizard .....	32
Figure 5.2. IP Configuration .....	33
Figure 5.3. Check Generated Result .....	33
Figure 5.4. Simulation Wizard .....	35
Figure 5.5. Add and Reorder Source .....	36
Figure 5.6. Select simulation Top module .....	36
Figure 5.7. Simulation Waveform .....	37
Figure 5.8. Simulation Results .....	37

## Tables

Table 1.1. Summary of the GDDR I/O Module .....	6
Table 1.2. Minimum Device Requirements for GDDR I/O Module .....	7
Table 2.1. Source Synchronous DDR Interfaces .....	9
Table 3.1. General Attributes .....	26
Table 4.1. GDDR I/O Module Signals for Receive Interfaces .....	28
Table 4.2. GDDR I/O Module Signals for Transmit Interfaces .....	30
Table 5.1. Generated File List .....	34
Table A.1. Resource Utilization .....	38

## Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviations	Definition
GDDR	Generic Double Data Rate
I/O	Input/Output
IP	Intellectual Property
MIPI	Mobile Industry Processor Interface
BIDIR	Bidirectional
HS	High Speed
LP	Low Power
ECLK	Edge clock
PCLK	Primary clock
SCLK	System clock
Edge-aligned	Data is edge aligned with clock
Center-aligned	Data is center aligned with clock
LSB	Least Significant Bit
MSB	Most Significant Bit

# 1. Introduction

## 1.1. Overview of the IP

The Lattice Semiconductor Generic Double Data Rate Input/Output (GDDR I/O) Module core is designed for high-speed data transfer across a wide range of applications. It supports both edge-aligned and center-aligned source synchronous interfaces for both receive and transmit.

## 1.2. Quick Facts

**Table 1.1. Summary of the GDDR I/O Module**

IP Requirements	Supported Devices	MachXO4™
	IP Changes <sup>1</sup>	For a list of changes to the IP, refer to the <a href="#">MachXO4 DDR Generic Module Release Notes (FPGA-RN-02094)</a> .
Resource Utilization	Supported User Interface	Data : Native
	Resources	Refer <a href="#">Appendix A. Resource Utilization</a> .
Design Tool Support	Lattice Implementation	IP Core v3.0.0 – Lattice Radiant™ Software 2025.2.
	Synthesis	Synopsys® Synplify Pro® for Lattice. Lattice Synthesis Engine (LSE)
	Simulation	For a list of supported simulators, refer to the <a href="#">Lattice Radiant Software User Guide</a>

**Note:**

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

## 1.3. Features

The key features of GDDR I/O Module include:

- Support interfaces:
  - Receive
  - Receive MIPI
  - Transmit
  - Transmit MIPI
- Supported gearing: x1, x2, x4
- Selectable I/O type:
  - Single-ended
  - Differential Signaling
- 1-bit to 128-bit data bus width
- Supports 1 MHz to 450 MHz DDR clock frequency
  - 1 MHz to 150 MHz for x1 Gearing
  - 1 MHz to 332 MHz for x2 Gearing
  - 1 MHz to 400 MHz for x4 Gearing
  - Up to 450MHz for MIPI interface mode
- Clock-data relationship options:
  - Edge-to-edge
  - Centered
- Data Path Delay that includes following options:
  - Bypass
  - Predefined (Receive Interface only)
  - User-defined (Receive Interface only)
  - Dynamic (Receive Interface only)

- Tri-state control (Transmit Interface only)

## 1.4. Licensing and Ordering Information

The GDDR I/O Module is provided at no additional cost with the Lattice Radiant software.

## 1.5. Minimum Device Requirements

The minimum device requirements for the GDDR I/O Module are as follows:

**Table 1.2. Minimum Device Requirements for GDDR I/O Module**

Device	Speed Grade	Interface	Gearing	Data Rate (Mbps) Per lane
MachXO4	6	Receive, Transmit	x1	2 to 300
			x2	2 to 664
			x4	2 to 800
		Receive MIPI, Transmit MIPI	x4	2 to 900
	5	Receive, Transmit	x1	2 to 250
			x2	2 to 554
			x4	2 to 630
		Receive MIPI, Transmit MIPI	x4	2 to 900

## 1.6. Naming Conventions

### 1.6.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

### 1.6.2. Signal Names

- `_n` are active low (asserted when value is logic 0)
- `_i` are input signals
- `_o` are output signals

## 2. Functional Description

### 2.1. IP Architecture Overview

The GDDR I/O Module uses dedicated FPGA DDR I/O primitives to implement double-data-rate (DDR) functionality.

Figure 2.1 shows a high-level block diagram of the GDDR I/O Module.

For Receive, incoming DDR data goes through *Data Delay Control* block, which includes DELAYH and DELAYG primitives. The block allows the incoming DDR data to be adjusted with a static delay value or dynamically via signals. The DDR data flows to *IDDR(x)* block for data sampling. The IDDR(x) block includes of IDDRX1, IDDRX2, and IDDRX4 primitives to support different gearing.

Meanwhile, the incoming DDR clock passes through the *Clock Delay Control* block, which includes DQSDLL and DLLDEL primitives. This block generates a 90° phase-shifted DDR clock used by IDDR(x) for data sampling. The *rx\_sync* block is a reset synchronization soft logic that ensure all components start on the same clock cycle. This prevents bus bit-order scrambling caused by reset pulse delay.

For Transmit, the *ODDR(x)* block that consists of ODDRX1, ODDRX2, and ODDRX4 primitives. The block sends DDR clock and DDR data to the external FPGA. The PLL generates the DDR clock and its 90° phase shifted DDR clock. The MIPI LP BIDIR IO block presents is both receive and transmit block is used only in MIPI mode. It functions as a bi-direction I/O buffer for MIPI Low Power (LP) signals.

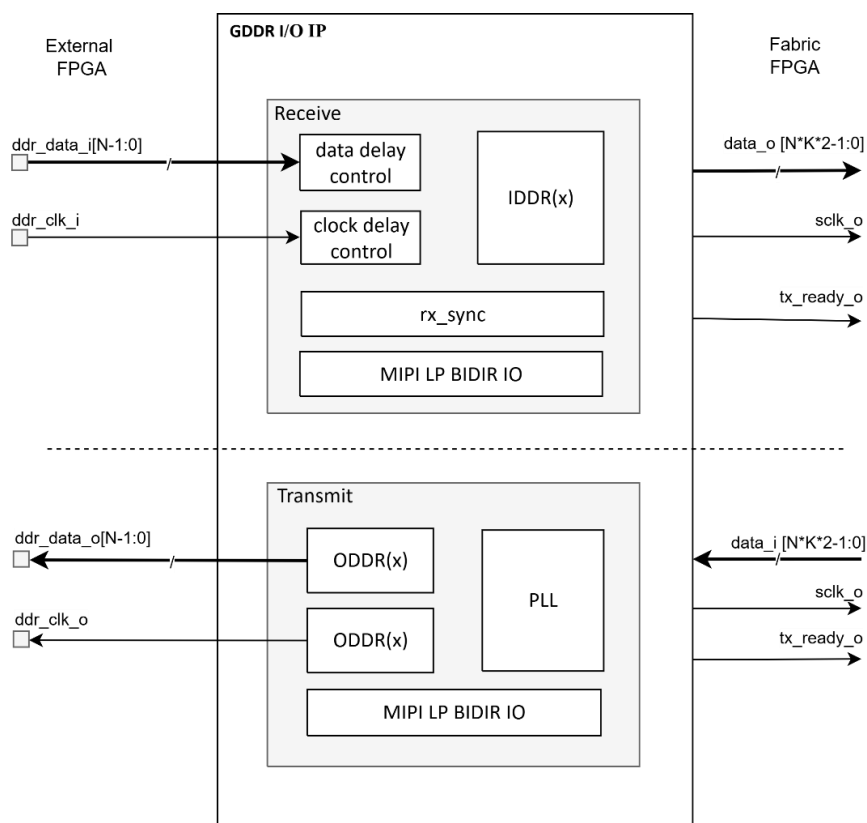


Figure 2.1. GDDR I/O Module Block Diagram

**Note:** N = number of lanes/DDR bus width, K= 2(for x1 gearing), 4(for x2 gearing), and 8(for x4 gearing).

Table 2.1 provides a summary of supported source synchronous DDR interfaces for GDDR I/O Module.



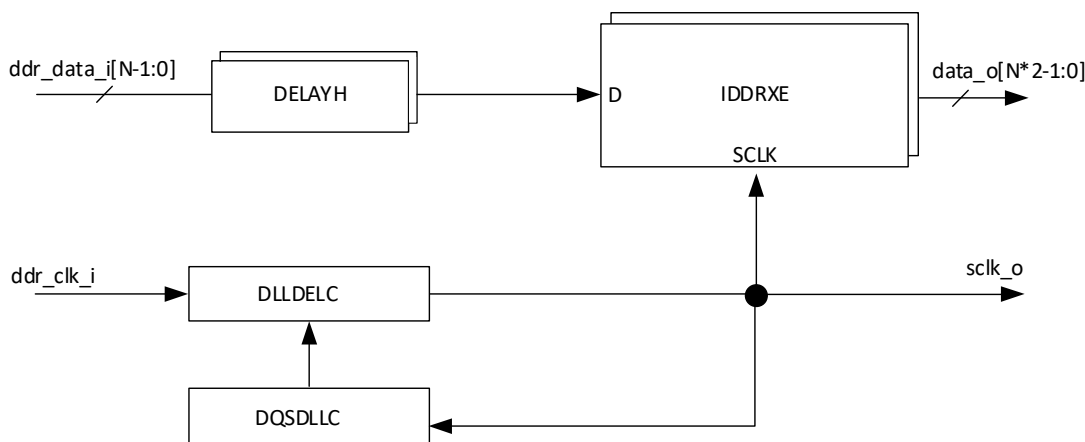
**Table 2.1. Source Synchronous DDR Interfaces**

Interface	Descriptions
<b>Receive</b>	
GDDR1_RX.SCLK.Aligned	<ul style="list-style-type: none"> <li>x1 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data.</li> <li>Uses DQS DLL &amp; DLL DEL to generate 90° shifted DDR clock</li> <li>Support bypass, predefined and user-defined data delay control</li> <li>Refer to <a href="#">2.1.1 GDDR1_RX.SCLK.Aligned</a> for details</li> </ul>
GDDR1_RX.SCLK.Centered	<ul style="list-style-type: none"> <li>x1 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Support bypass, predefined and user-defined data delay control</li> <li>Refer to <a href="#">2.1.2 GDDR1_RX.SCLK.Centered</a> for details</li> </ul>
GDDR2_RX.ECLK.Aligned	<ul style="list-style-type: none"> <li>x2 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data</li> <li>Uses DQS DLL &amp; DLL DEL to generate 90° shifted DDR clock</li> <li>Support bypass, predefined, user-defined and dynamic data delay control</li> <li>Refer to <a href="#">2.1.3 GDDR2/4_RX.ECLK.Aligned</a> for details</li> </ul>
GDDR2_RX.ECLK.Centered	<ul style="list-style-type: none"> <li>x2 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Support bypass, predefined, user-defined and dynamic data delay control</li> <li>Refer to <a href="#">2.1.4 GDDR2/4_RX.ECLK.Centered</a> for details</li> </ul>
GDDR4_RX.ECLK.Aligned	<ul style="list-style-type: none"> <li>x4 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data</li> <li>Uses DQS DLL &amp; DLL DEL to generate 90° shifted DDR clock</li> <li>Support bypass, predefined, user-defined and dynamic data delay control</li> <li>Refer to <a href="#">2.1.3 GDDR2/4_RX.ECLK.Aligned</a> for details</li> </ul>
GDDR4_RX.ECLK.Centered	<ul style="list-style-type: none"> <li>x4 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Support bypass, predefined, user-defined and dynamic data delay control</li> <li>Refer to <a href="#">2.1.4 GDDR2/4_RX.ECLK.Centered</a> for details</li> </ul>
GDDR4_RX.MIPI (HS&LP mode and HS-only mode)	<ul style="list-style-type: none"> <li>MIPI RX Interface</li> <li>x4 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Support bypass, predefined, user-defined and dynamic data delay control</li> <li>Support HS&amp;LP mode or HS-only mode</li> <li>Refer to <a href="#">2.1.5 GDDR4_RX.MIPI</a> for details</li> </ul>
<b>Transmit</b>	
GDDR1_TX.SCLK.Aligned	<ul style="list-style-type: none"> <li>x1 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data</li> <li>Refer to <a href="#">2.1.6 GDDR1_TX.SCLK.Aligned</a> for details</li> </ul>
GDDR1_TX.SCLK.Centered	<ul style="list-style-type: none"> <li>x1 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Refer to <a href="#">2.1.7 GDDR1_TX.SCLK.Centered</a> for details</li> </ul>
GDDR2_TX.ECLK.Aligned	<ul style="list-style-type: none"> <li>x2 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data</li> <li>Refer to <a href="#">2.1.8 GDDR2/4_TX.ECLK.Aligned</a> for details</li> </ul>
GDDR2_TX.ECLK.Centered	<ul style="list-style-type: none"> <li>x2 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Refer to <a href="#">2.1.9 GDDR2/4_TX.ECLK.Centered</a> for details</li> </ul>
GDDR4_TX.ECLK.Aligned	<ul style="list-style-type: none"> <li>x4 gearing</li> <li>Edge-aligned interface. DDR clock is aligned with DDR data</li> </ul>

	<ul style="list-style-type: none"> <li>Refer to <a href="#">2.1.8 GDDR2/4_TX.ECLK.Aligned</a> for details</li> </ul>
GDDR4_TX.ECLK.Centered	<ul style="list-style-type: none"> <li>x4 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Refer to <a href="#">2.1.9 GDDR2/4_TX.ECLK.Centered</a> for details</li> </ul>
GDDR4_TX.MIPI (HS&LP mode & HS-only mode)	<ul style="list-style-type: none"> <li>MIPI TX Interface</li> <li>x4 gearing</li> <li>Center-aligned interface. DDR clock is in center of DDR data</li> <li>Support HS&amp;LP mode or HS-only mode</li> <li>Refer to <a href="#">2.1.10 GDDR4_TX.MIPI</a> for details</li> </ul>

### 2.1.1. GDDR1\_RX.SCLK.Aligned

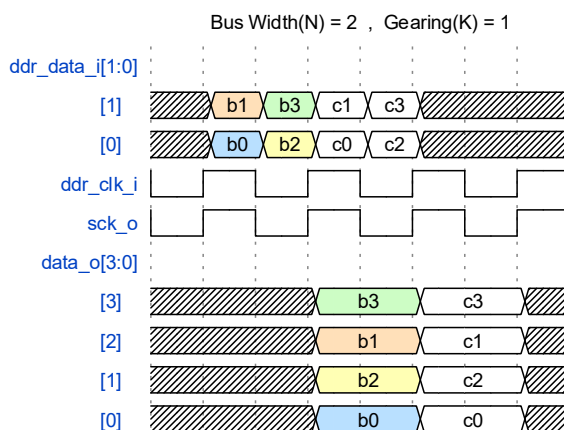
This section describes the implementation of *GDDR1\_RX.SCLK.Aligned* interface. *DELAYH* delays the data to match the *ECLK* injection delay. The DDR input clock (*ddr\_clk\_i*) is fed to *DLL* primitives to delay the DDR clock by 90° and is used as sampling clock for the *IDDRX* primitive.



**Note:** N = Number of lanes or DDR bus width

**Figure 2.2. GDDR1\_RX.SCLK.Aligned Block Diagram**

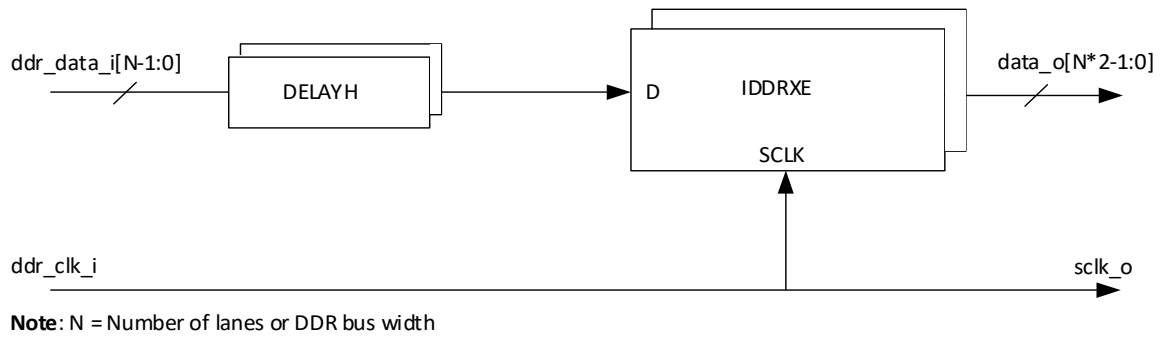
Figure 2.3 shows the order of output data (*data\_o*) in relative to the input data (*ddr\_data\_i*), where the data width (*N*) = 2 and X1 gearing. The even bit of *data\_o* are the data captured on the positive edge of *ddr\_clk\_i*, and odd bits of *data\_o* are the data captured on the negative edge of *ddr\_clk\_i*.



**Figure 2.3. DDR Data Output Arrangement for GDDR1\_RX.SCLK.Aligned Interface**

### 2.1.2. GDDR1\_RX.SCLK.Centered

This section describes the implementation of *GDDR1\_RX.SCLK.Centered* interface. *DELAYH* delays the data to match the *ECLK* injection delay. In this configuration, the *DDR* clock (*ddr\_clk\_i*) is directly fed to *IDDR* primitive as sampling clock.



**Figure 2.4. GDDR1\_RX.SCLK.Centered Block Diagram**

The output data arrangement for *GDDR1\_RX.SCLK.Centered* interface is same as that of *GDDR1\_RX.SCLK.Aligned* interface. Refer to [Figure 2.3](#) for details.

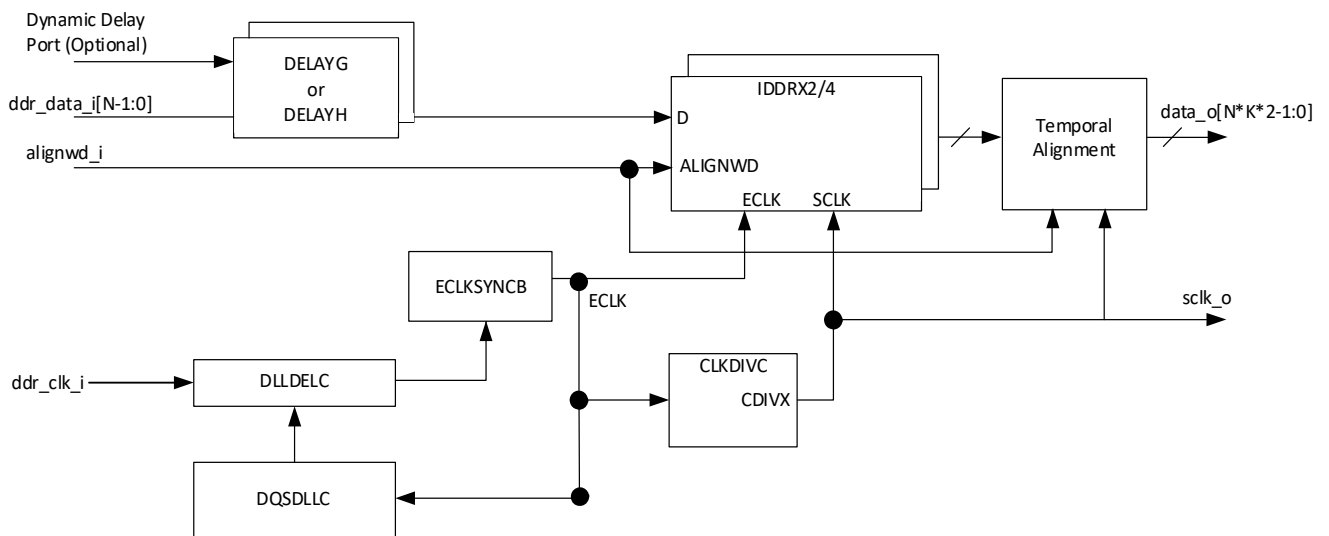
### 2.1.3. GDDR2/4\_RX.ECLK.Aligned

This section describes the implementation of *GDDR2\_RX.ECLK.Aligned* and *GDDR4\_RX.ECLK.Aligned* interfaces. The *DQS DLL* provides a 90° clock phase shift to center the clock at the *IDDR2/4* primitive. *DELAYH* delays the data to match the *ECLK* injection delay. *DELAYG* can also be used to control the delay dynamically.

The *GDDR2\_RX.ECLK.Aligned* interface uses x2 gearing with *IDDR2E* primitive. The interface requires the use of a *CLKDIVC* to provide the *SCLK* which is half the frequency of the *ECLK*.

The *GDDR4\_RX.ECLK.Aligned* interface uses x4 gearing with *IDDR4B* primitive. The interface requires the use of a *CLKDIVC* to provide the *SCLK* which is quarter the frequency of the *ECLK*.

The *ECLKSYNCB* element is associated with the *ECLK* and must be used to drive the it. The port *alignwd\_i* can be used for word alignment at the interface. The *Temporal Alignment* block introduces a one *SCLK* period delay to correct the temporal misalignment of *DDR2E Q3* data output and *DDR4B Q7* data outputs.

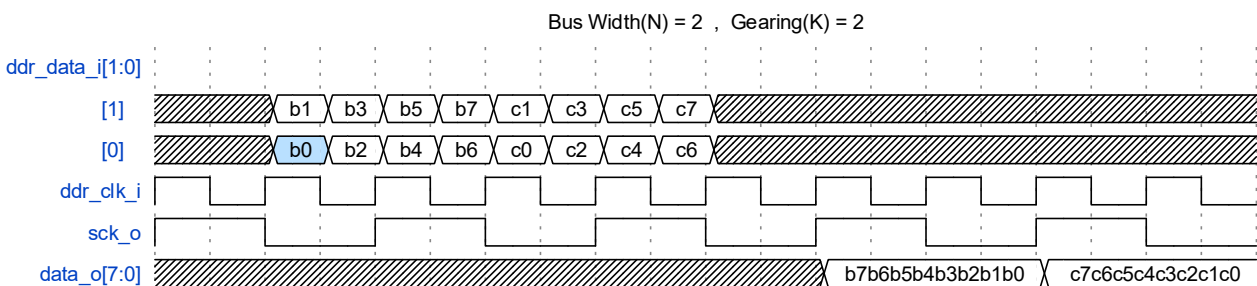


**Notes:**

- N = Number of lanes or DDR bus width.
- K = 2(for x1 gearing), 4(for x2 gearing), and 8(for X4 gearing)

**Figure 2.5. GDDR2/4\_RX.ECLK.Aligned Block Diagram**

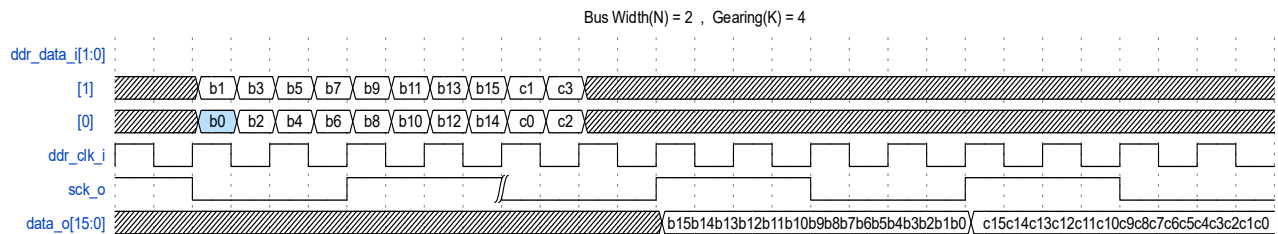
Figure 2.6 shows the order of output data (*data\_o*) in relative to the incoming DDR input data (*ddr\_data\_i*) for the *GDDR2\_RX.ECLK.Aligned* interface, where the data width (N) is equal to 2. The *data\_o* is arranged in LSB to MSB order. The lower N\*2 bits represent *ddr\_data\_i* captured by the first *ddr\_clk\_i*, and upper N\*2 bits represent *ddr\_data\_i* captured by the second *ddr\_clk\_i*. This is different from X1 gearing interface because the *Temporal Alignment* block rearranges the output data of *IDDR2*.



**Figure 2.6. Data output order for GDDR2.RX.ECLK.Aligned Interface**

Figure 2.7 shows the order of DDR output data (*data\_o*) relative to the incoming DDR input data (*ddr\_data\_i*) for *GDDR4\_RX.ECLK.Aligned* interface, where the data width (N) is equal to 2. The first bit of DDR input data (*b0*) shifted into the IP is captured at LSB bit of *data\_o*. This is different from x1 gearing interface because the *Temporal Alignment* block helped to rearrange the output data of *IDDRX4*.

The *data\_o* is arranged in LSB to MSB order. The first N\*2 bits represent *ddr\_data\_i* captured by the first *ddr\_clk\_i*, and the next upper N\*2 bits represent data captured by the second *ddr\_clk\_i* and continue up to every N\*2 bits until the fourth *ddr\_clk\_i*. This is different from X1 gearing interface because the *Temporal Alignment* block rearranges the output data of *IDDRX4*.



**Figure 2.7. Data output order for GDDR4\_RX.ECLK.Aligned interface**

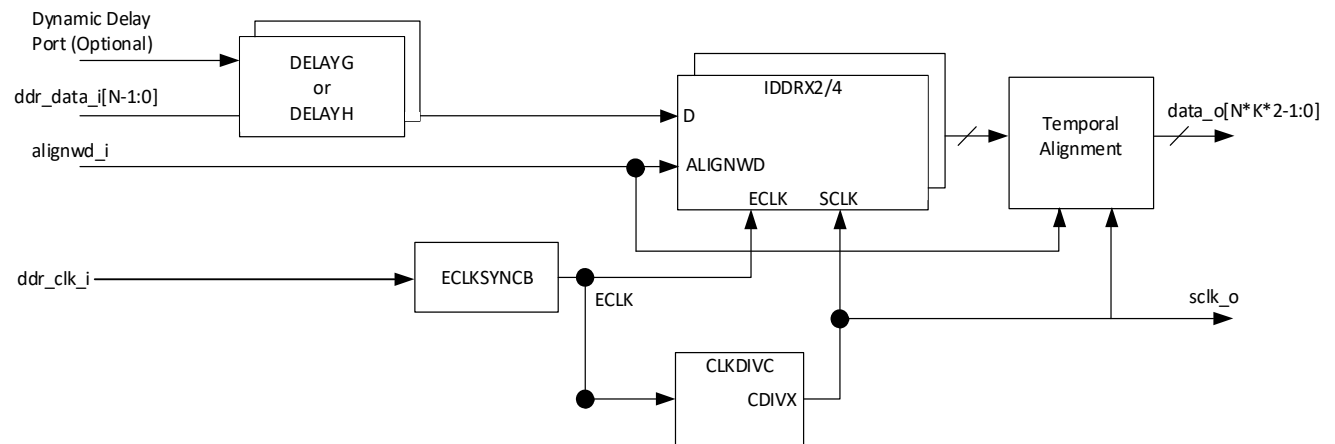
### 2.1.4. GDDR2/4\_RX.ECLK.Centered

This section describes the implementation of *GDDR2\_RX.ECLK.Centered* and *GDDR4\_RX.ECLK.Centered* interfaces. *DELAYH* or *DELAYG* is used to match edge clock delay at the *IDDR2/4*.

The *GDDR2\_RX.ECLK.Centered* interface uses x2 gearing with *IDDR2E* primitive. This requires the use of *CLKDIVC* to provide the *SCLK* which is half of *ECLK*.

The *GDDR4\_RX.ECLK.Centered* interface uses x4 gearing with the *IDDR4B* primitive. This requires the use of a *CLKDIVC* to provide the *SCLK* which is one quarter of the *ECLK* frequency.

The *ECLKSYNCB* element is associated with the *ECLK* and must be used to drive the *ECLK*. The port *alignwd\_i* can be used for word alignment at the interface. The *Temporal Alignment* block introduces a one *SCLK* period delay to correct the temporal misalignment of *DDR2E* Q3 data output and *DDR4B* Q7 data outputs.



**Notes:**

- N = Number of lanes or DDR bus width
- K = 2(for x1 gearing), 4(for x2 gearing), and 8(for X4 gearing)

**Figure 2.8. GDDR2/4\_RX.ECLK.Centered Block Diagram**

The output data arrangement for *GDDR2\_RX.ECLK.Centered* interface is the same as that of the *GDDR2\_RX.ECLK.Aligned* interface. Refer to [Figure 2.6](#) for details. Similarly, the output data arrangement for *GDDR4\_RX.ECLK.Centered* interface matches that of the *GDDR4\_RX.ECLK.Aligned* interface. Refer to [Figure 2.7](#) for details.

### 2.1.5. GDDR4\_RX.MIPI

The *GDDR4\_RX.MIPI* interface reuses *GDDR4\_RX.ECLK.Centered* interface for MIPI High-Speed (HS) signals and adds bidirectional I/O buffers for MIPI Low-Power (LP) signals as shown in Figure 2.9. It supports both HS&LP mode, as well as HS-only mode.

In HS&LP mode, the *ddr\_data\_i* and *ddr\_clk\_i* signals from the *GDDR4\_RX.ECLK.Centered* interface are used as MIPI HS signals, with the *IO\_TYPE* set to MIPI. The MIPI LP signals are set to *LP\_MIPI*. The following list the MIPI LP signals:

- *mipi\_buf\_dout*
- *mipi\_clk\_lp0\_io*
- *mipi\_clk\_lp1\_io*
- *mipi\_data\_lp0\_io[N\*K-1:0]*
- *mipi\_data\_lp1\_io[N\*K-1:0]*

In HS-only mode, the MIPI LP signals are removed, and only the MIPI HS signals (*ddr\_data\_i* and *ddr\_clk\_i*) are retained. The output data arrangement for *GDDR4\_RX.MIPI* interface is same as that of the *GDDR4\_RX.SCLK.Aligned* interface. Refer to Figure 2.7 for details.

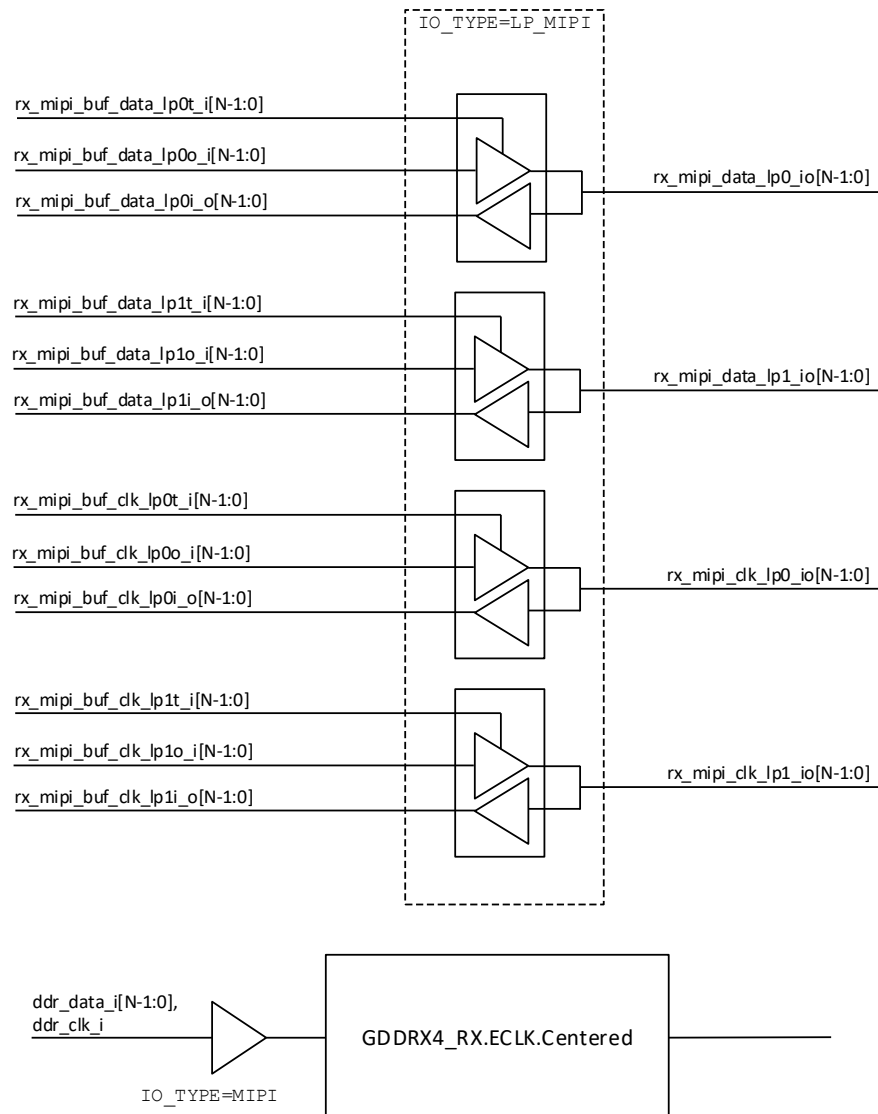
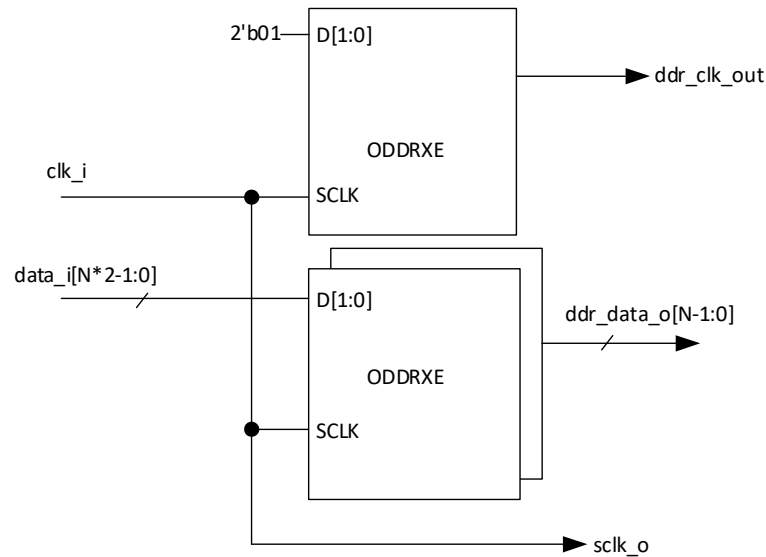


Figure 2.9. GDDR4\_RX.MIPI Block Diagram

### 2.1.6. GDDR1\_TX.SCLK.Aligned

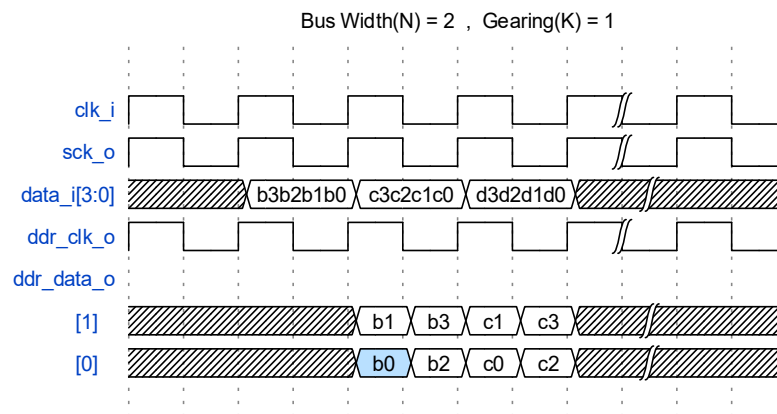
The *GDDR1\_TX.SCLK.Aligned* interface provides clock and data that are edge-aligned. An additional *ODDRXE* primitive is used for the DDR output clock (*ddr\_clk\_o*) to minimize the skew difference between the clock path (*ddr\_clk\_o*) and the data path (*ddr\_data\_o*). The *SCLK* is exposed to IP top level as *sclk\_o* signal, which you can use to align the input parallel data (*data\_i*) with *sclk\_o*.



**Note:** N = number of lanes or DDR bus width

**Figure 2.10. GDDR1\_TX.SCLK.Aligned Block Diagram**

Figure 2.11 shows the order of DDR output data (*ddr\_data\_o*) relative to the input parallel data (*data\_i*) for *GDDR1\_TX.SCLK.Aligned*, interface where the data width (N) is equal to 2. The *ddr\_data\_o* transmits *data\_i* in LSB first order per bus width (N), with *b0* and *b1* sent first.

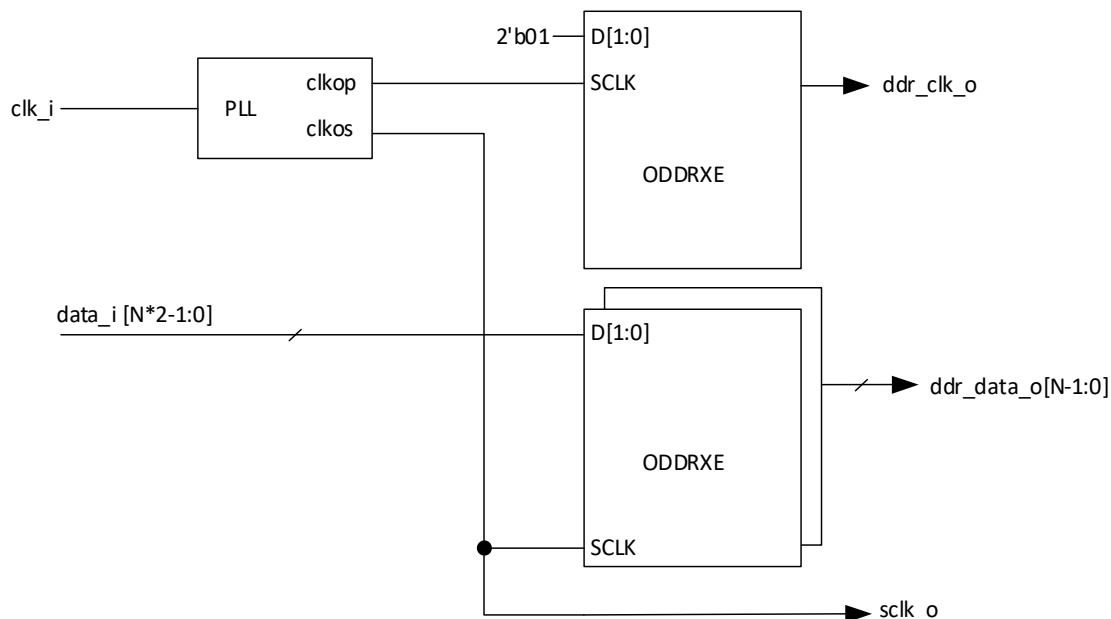


**Figure 2.11. Data Output Order for GDDR1\_TX.SCLK.Aligned Interface**



### 2.1.7. GDDR1\_TX.SCLK.Centered

The *GDDR1\_TX.SCLK.Centered* interface provides clock and data that are center-aligned. A PLL is required to generate the 90° phase difference clocks. The *clkop* signal drives the output data *ODDRX* primitive, while *clkos* (shifted by 90°) drives the output clock *ODDRX* primitive. There is an IP GUI option to include or exclude PLL instantiation within the IP. The SCLK is exposed to IP top level as *sclk\_o* signal, which you can use to align the input parallel data (*data\_i*) with *sclk\_o*.



**Note:** N = number of lanes of DDR bus width

**Figure 2.12. GDDR1\_TX.SCLK.Centered Block Diagram**

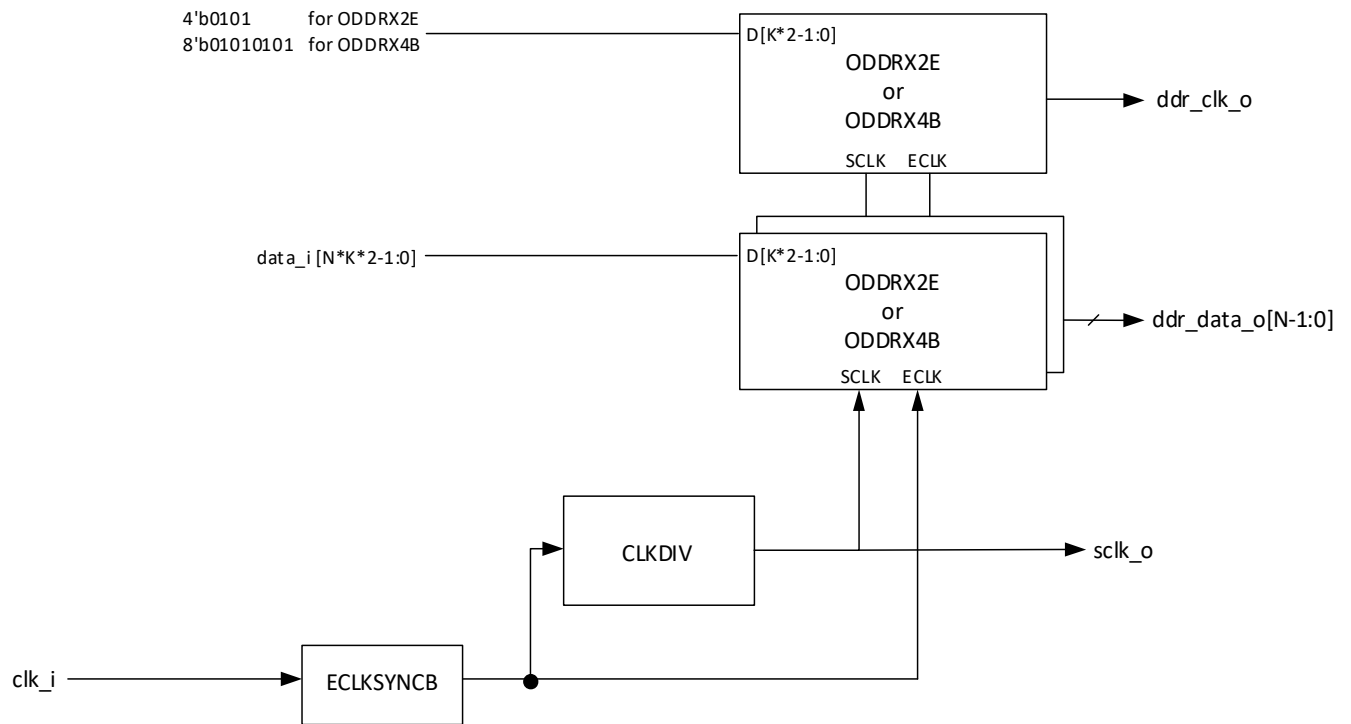
### 2.1.8. GDDR2/4\_TX.ECLK.Aligned

This section describes the implementation of *GDDR2\_TX.ECLK.Aligned* and *GDDR4\_TX.ECLK.Aligned* interfaces. These interfaces provide edge-aligned clock and data.

For *GDDR2\_TX.ECLK.Aligned* interface, a *CLKDIV* generates the *SCLK* at half of the *ECLK* frequency.

For *GDDR4\_TX.ECLK.Aligned* interface, the *SCLK* is a quarter of the *ECLK* frequency.

The *ECLKSYNCB* element is used on the *ECLK* path for data synchronization. The *SCLK* is exposed to IP top level as *sclk\_o* signal, which you can use to align the input parallel data (*data\_i*) with *sclk\_o*.

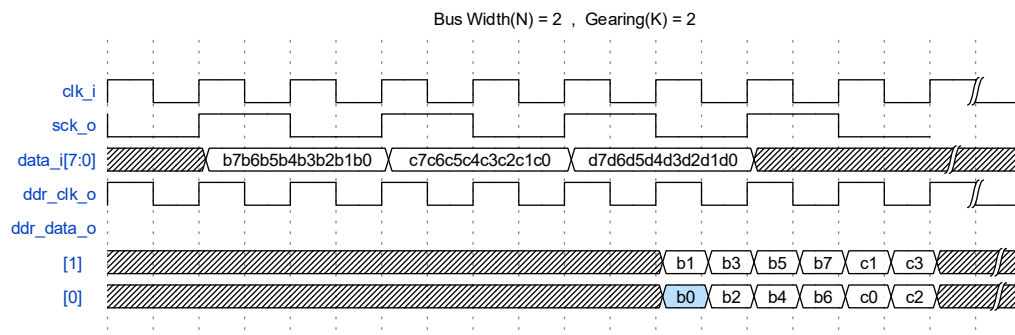


**Notes:**

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

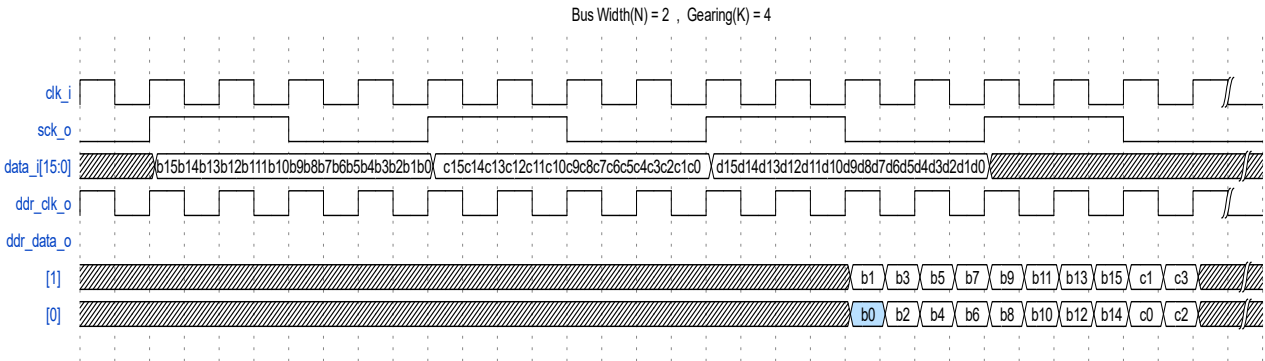
**Figure 2.13. GDDR2/4\_TX.ECLK.Aligned Block Diagram**

Figure 2.14 shows the order of DDR output data (*ddr\_data\_o*) relative to the input parallel data (*data\_i*) for *GDDR2\_TX.ECLK.Aligned* interface, where the data width (N) is equal to 2. The *ddr\_data\_o* transmits the *data\_i* in LSB first order per bus width (N), with *b0* and *b1* are sent first.



**Figure 2.14. Data Output Order for GDDR2.TX.ECLK.Aligned Interface**

Figure 2.15 shows the order of DDR output data (*ddr\_data\_o*) relative to the input parallel data (*data\_i*) for *GDDR4\_TX.ECLK.Aligned* interface where the bus width (N) is equal to 2. The DDR output data (*ddr\_data\_o*) transmits *data\_i* in LSB first order per bus width(N), with *b0* and *b1* sent first.



**Figure 2.15. Data Output Order for GDDR4.TX.ECLK.Aligned Interface**

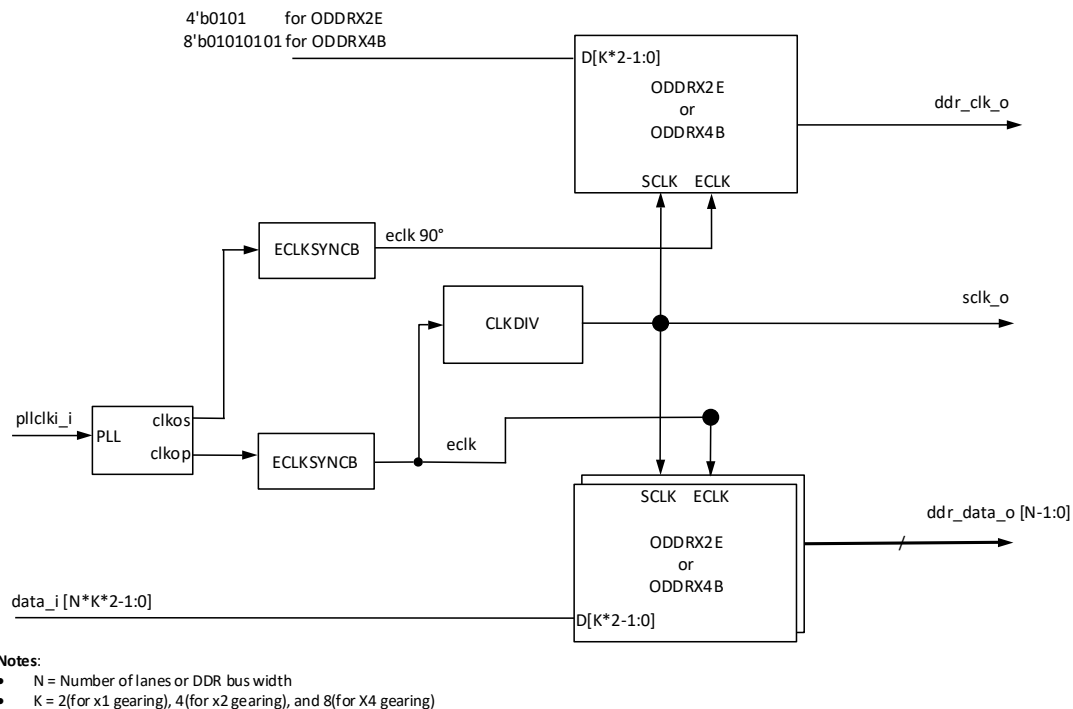
### 2.1.9. GDDR2/4\_TX.ECLK.Centered

This section describes the implementation of *GDDR2\_TX.ECLK.Centered* and *GDDR4\_TX.ECLK.Centered* interfaces. These interfaces provide center-aligned clock and data. A PLL is required to generate clocks with a 90° phase difference. The *clkop* signal drives *ODDR2/4* primitive, while *clkos* (shifted by 90°) drives the output clock *ODDR2/4* primitive. There is IP GUI option for to include or exclude the PLL instantiation within the IP.

For *GDDR2\_TX.ECLK.Centered* interface, a *CLKDIV* generates the *SCLK* at half of the *ECLK* frequency.

For *GDDR4\_TX.ECLK.Centered* interface, the *SCLK* is a quarter of the *ECLK* frequency.

Two *ECLKSYN*B primitives are used on the *ECLK* path for data synchronization. The *SCLK* is exposed to IP top level as *sclk\_o* signal, which you can use to align the input parallel data (*data\_i*) with *sclk\_o*.



**Figure 2.16. GDDR2/4\_TX.ECLK.Centered Block Diagram**

The output data arrangement for *GDDR2\_TX.ECLK.Centered* interface is same as that of the *GDDR2\_TX.ECLK.Aligned* interface. Refer to [Figure 2.14](#) for details. Similarly, the output data arrangement for *GDDR4\_TX.ECLK.Centered* interface is same that of the *GDDR4\_TX.ECLK.Aligned* interface. Refer to [Figure 2.15](#) for details.

### 2.1.10. GDDR4\_TX.MIPI

The **GDDR4\_TX.MIPI** interface reuse the **GDDR4\_TX.ECLK.Centered** interface for MIPI High-Speed (HS) signals and adds bidirectional I/O buffers for MIPI Low-Power (LP) signals, as shown in Figure 2.17 . It supports HS&LP mode and HS-only mode.

In HS&LP mode, the **ddr\_data\_o** and **ddr\_clk\_o** signals from **GDDR4\_TX.ECLK.Centered** interface are used as MIPI HS signals, with the **IO\_TYPE** are set to MIPI. The MIPI LP signals are set to **LP\_MIPI**. The following list the MIPI LP signals:

- **mipi\_buf\_dout**
- **mipi\_clk\_lp0\_io**
- **mipi\_clk\_lp1\_io**
- **mipi\_data\_lp0\_io[N\*K-1: 0]**
- **mipi\_data\_lp1\_io[N\*K-1: 0]**

In HS-only mode, the MIPI LP signals are removed, and only the MIPI HS signals (**ddr\_data\_o** and **ddr\_clk\_o**) remain.

The DDR output data order for **GDDR4\_TX.MIPI** interface is same as that of the as **GDDR4\_TX.ECLK.Centered** interface. Refer to Figure 2.16 for details.

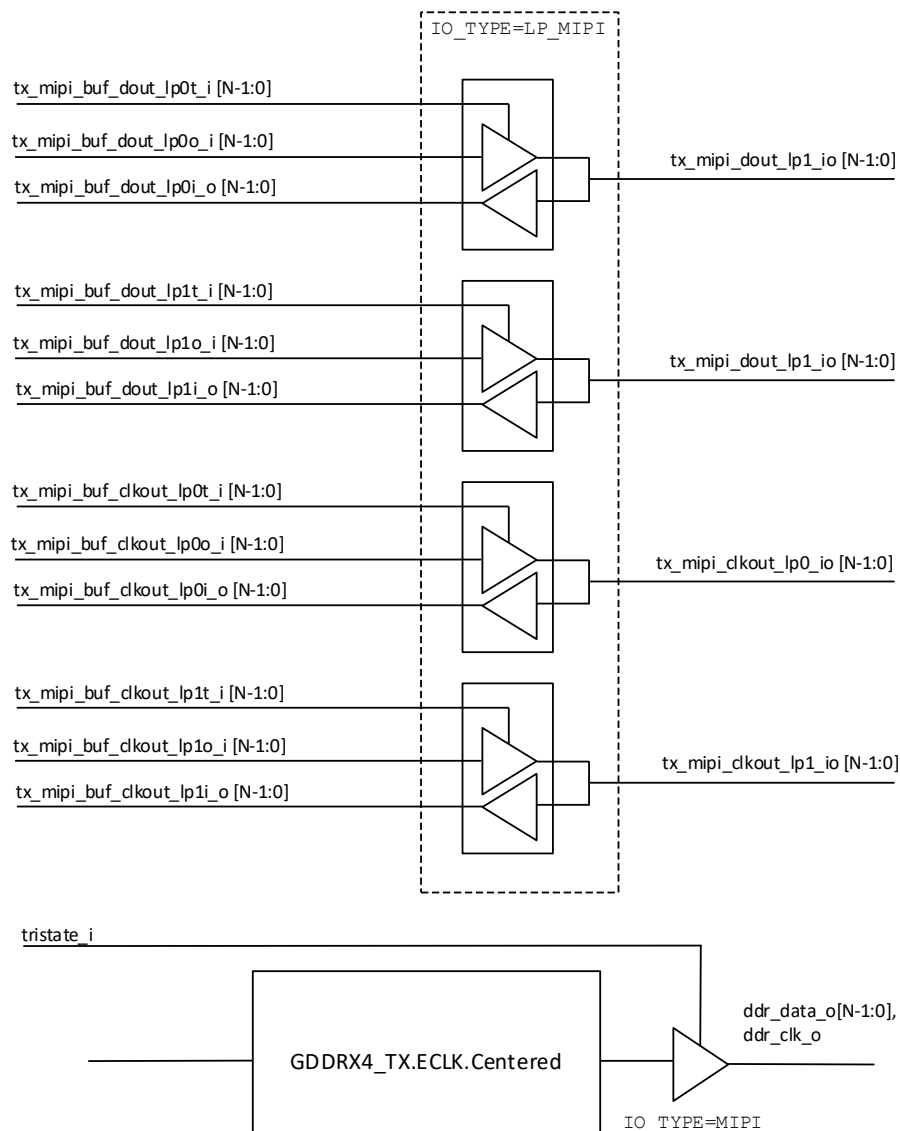


Figure 2.17. GDDR4\_TX.MIPI Block Diagram

## 2.2. Clocking

This section explains the clocking networks for the Receive and Transmit interfaces of GDDR I/O Module.

### 2.2.1. Receive Interfaces Clocking

Figure 2.18 shows an overview of the clocking networks of all receive interfaces for this IP. The incoming DDR clock (*ddr\_clk\_i*) must use a dedicated FPGA clock pin that has PCLK routing. Note that the incoming DDR clock must be a continuous clock.

In edge-aligned mode, the DLL shifts DDR clock by 90°. Otherwise, this will be bypassed. *ECLKSYNC* synchronizes the *PCLK* and *ECLK* networks. The output of *ECLKSYNC* is fed to *IDDR* and used as *DDR* sampling clock for *IDDR* blocks.

In parallel, output of *ECLKSYNC* is also fed to *CLKDIV* to generate the parallel clock (*sclk\_o*). The parallel data (*data\_o*) from the output of *IDDR* blocks is synchronized to this parallel clock (*sclk\_o*).

As shown in Figure 2.18, you must provide the *sync\_clk\_i* signal. The cross-clock-domain (CDC) handling for the *sync\_clk\_i* is managed within the IP. You only need to ensure that its frequency is slower than the *DDR* clock (*ddr\_clk\_i*).

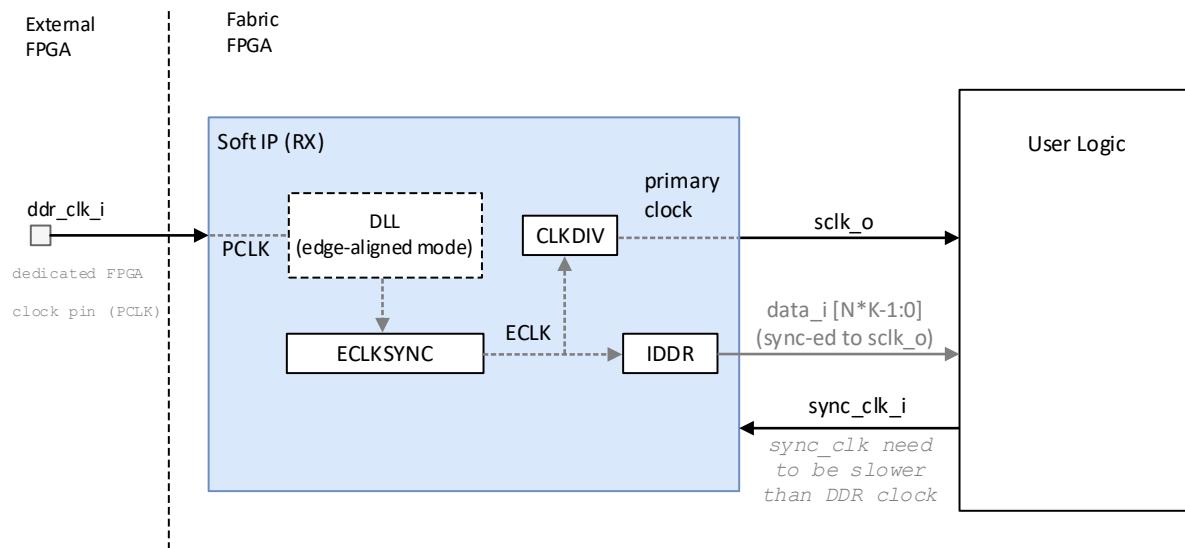


Figure 2.18. Clocking Network for Receive Interfaces

## 2.2.2. Transmit Interfaces Clocking

Figure 2.19 shows overview clocking network of all transmit interfaces for this IP. The outgoing DDR clock (*ddr\_clk\_o*) must use a dedicated FPGA clock pin that has PCLK routing. The DDR clock is also generated from the ODDR to ensure minimum skew between outgoing DDR data and DDR clock.

In edge-aligned mode, you must provide the main DDR clock (*clk\_i*) from your logic. In center-aligned mode, you can enable PLL instantiation in this IP. For this configuration, you only need to provide the reference clock for PLL (*pll\_clk\_i*). You may also disable PLL instantiation in the IP and generate the DDR clocks (*clk\_i* and *clk90\_i*) from your own PLL logic.

The parallel data (*data\_i*) must be synchronized to parallel clock (*sclk\_o*) generated from this IP. As shown in Figure 2.19, you must also provide *sync\_clk\_i* signal. The cross-clock-domain (CDC) for *sync\_clk\_i* is managed within the IP. You only need to ensure that its frequency is slower than the DDR clock (*clk\_i*).

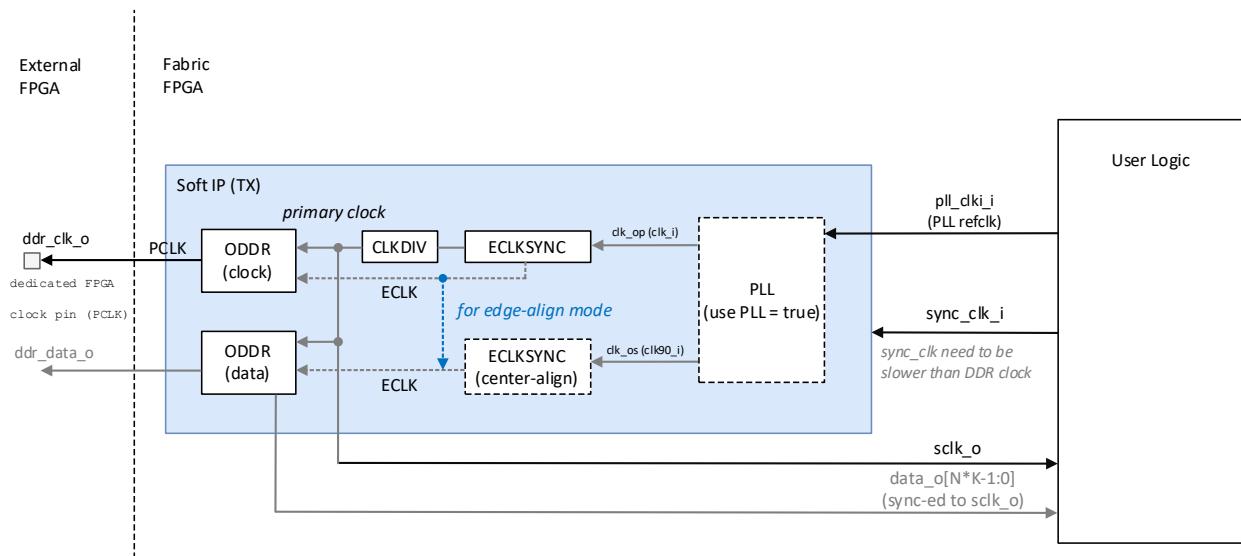


Figure 2.19. Clocking Network for Transmit Interfaces

## 2.3. Reset

The reset sequence of the GDDR I/O Module is divided into Receive and Transmit interfaces.

### 2.3.1. Reset Sequence for Receive Interfaces

Figure 2.20 shows the reset sequence of all x1 Receive interfaces.

1. Assert *rst\_i* for at least one DDR clock (*ddr\_clk\_i*) duration.
2. Wait for *rx\_lock\_o* to go HIGH. **Skip this step for center-aligned interfaces.**
3. Assert *uddcntl\_n\_i* to LOW for at least two *ddr\_clk\_i* period. This instructs the DQSDLL to generate 90° phase shifted DDR clock.
4. At the next available positive edge of *ddr\_clk\_i* clock, the IP is ready for operation.

Note: For x1 interface, there is no reset synchronization circuit and has only 1 reset signal (*rst\_i*).

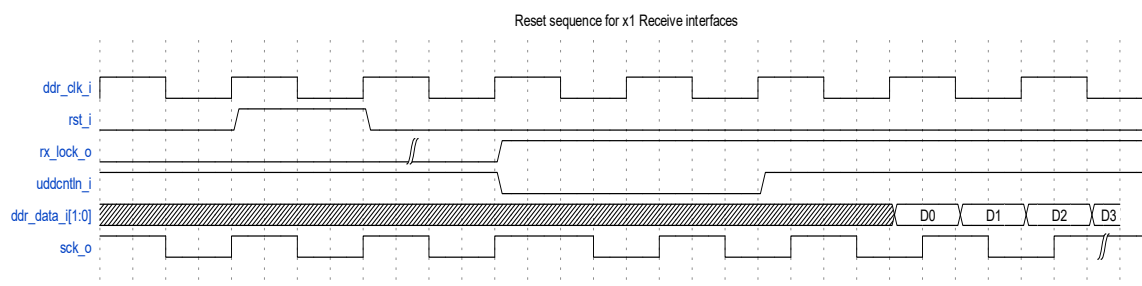


Figure 2.20. Reset Sequence of x1 Receive Interfaces

Figure 2.21 shows the reset sequence of all x2/x4 Receive interfaces.

1. Assert *dqsdll\_rst\_i* for at least one *sck\_o* duration. **Skip this step for center-aligned interface.**
2. Wait for *rx\_lock\_o* to go HIGH.
3. Assert *rst\_i* for at least one *sync\_clk\_i* duration.
4. Assert *sync\_init\_i*. This starts the reset synchronization circuit.
5. Wait for *rx\_ready\_o* to go HIGH. This indicates that the IP is ready to operate.

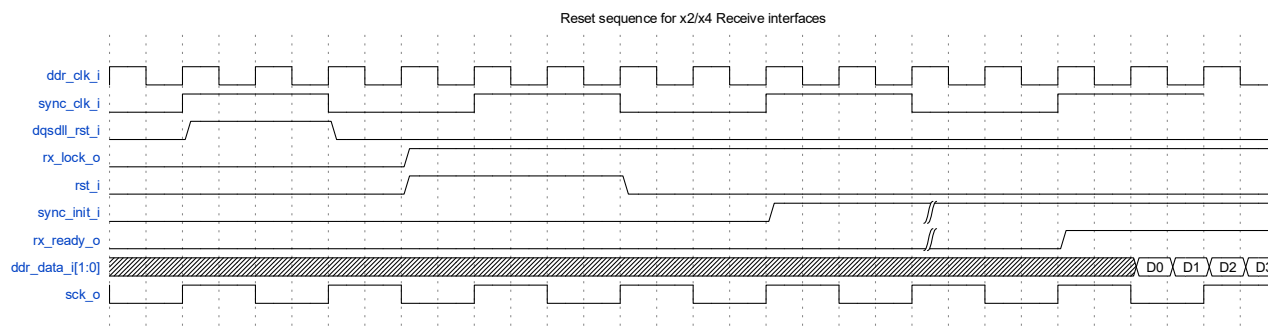


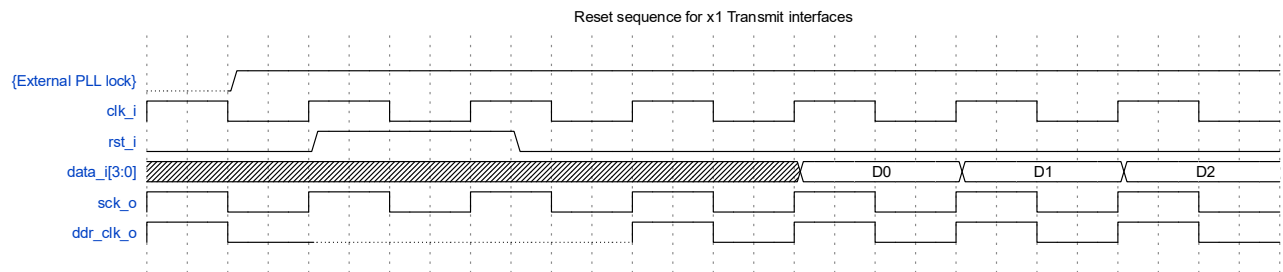
Figure 2.21. Reset Sequence of x2/x4 Receive Interfaces



## 2.3.2. Reset Sequence for Transmit Interfaces

Figure 2.22 shows the reset sequence of all x1 Transmit interfaces.

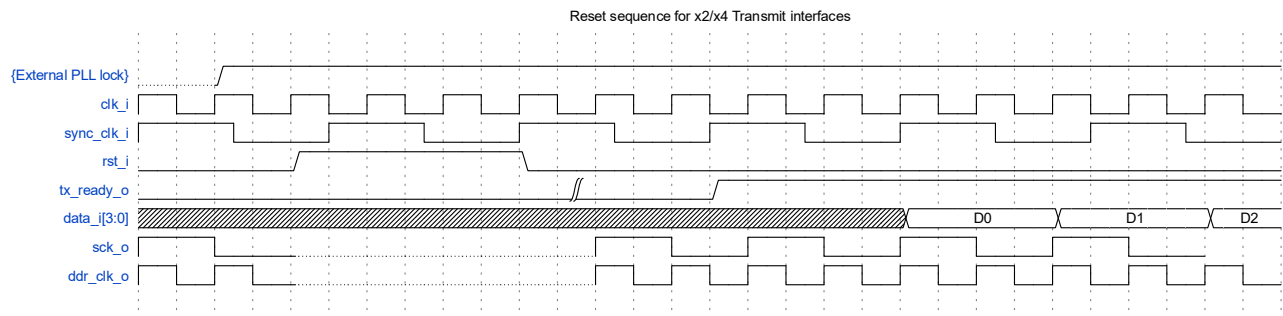
1. Ensure all input clocks are stable and locked.
2. Assert *rst\_i* for at least one input clock (*clk\_i*) duration.
3. Wait for about approximately one *sclk\_o* duration.
4. At the next available positive edge of *sclk\_o* clock, the IP is ready for operation.



**Figure 2.22. Reset Sequence of x1 Transmit Interfaces**

Figure 2.23 shows the reset sequence of all x2 and x4 Transmit interfaces.

1. Ensure all input clocks are stable and locked.
2. Assert *rst\_i* for at least one *sync\_clk\_i* duration.
3. Wait for *tx\_ready\_o* go HIGH. This indicates that the IP is ready for operation.



**Figure 2.23. Reset Sequence of x2/x4 Transmit Interfaces**

## 3. IP Parameter Description

The configurable attributes of the GDDR I/O Module are listed in the following table. You can configure the IP by setting these attributes accordingly in the IP Catalog's Module/IP Wizard in the Lattice Radiant software.

Default values are shown in bold where applicable.

### 3.1. General

**Table 3.1. General Attributes**

Attribute	Selectable Values	Description
Interface Type	<ul style="list-style-type: none"> <li>• <b>Receive (Default)</b></li> <li>• Transmit</li> <li>• Receive MIPI</li> <li>• Transmit MIPI</li> </ul>	Specify the DDR interface type.
Enable Tri-state Control	<ul style="list-style-type: none"> <li>• <b>Unchecked</b></li> <li>• Checked</li> </ul>	<p>Enable to instantiate Tri-State I/O buffer on both DDR data and clock.</p> <p><b>Dependency on other attributes:</b> Available when <i>(Interface Type) == Transmit or Transmit MIPI</i></p>
Enable MIPI High Speed Mode Only	<ul style="list-style-type: none"> <li>• <b>Unchecked</b></li> <li>• Checked</li> </ul>	<p>Enable to use only High Speed MIPI interface.</p> <p><b>Dependency on other attributes:</b> Available when <i>(Interface Type) == Transmit MIPI or Receive MIPI</i></p>
I/O Standard for this Interface	(Legal Combination Table)	Refer to IP GUI.
Gearing Ratio [K]	<ul style="list-style-type: none"> <li>• <b>x1</b></li> <li>• x2</li> <li>• x4</li> </ul>	Specify gearing ratio.
Bus Width for this Interface [N]	<p>Range : 1 – 128</p> <p><b>Default : 4</b></p>	<p>Specify the total number of lanes or bus width for DDR data interface.</p> <p>The maximum number of lanes varies with Gearing Ratio: x1 : 128 x2 : 32 x4 : 21</p>
Clock to Data Relations on the Pins	<ul style="list-style-type: none"> <li>• Edge-to-Edge</li> <li>• <b>Centered</b></li> </ul>	Specify the clock to data relationship of DDR interface at FPGA pins.
Interface	<ul style="list-style-type: none"> <li>• GDDR1_RX.SCLK.Aligned</li> <li>• <b>GDDR1_RX.SCLK.Centered</b></li> <li>• GDDR1_TX.SCLK.Aligned</li> <li>• GDDR1_TX.SCLK.Centered</li> <li>• GDDR2_RX.ECLK.Aligned</li> <li>• GDDR2_RX.ECLK.Centered</li> <li>• GDDR2_TX.ECLK.Aligned</li> <li>• GDDR2_TX.ECLK.Centered</li> <li>• GDDR4_RX.ECLK.Aligned</li> <li>• GDDR4_RX.ECLK.Centered</li> <li>• GDDR4_TX.ECLK.Aligned</li> <li>• GDDR4_TX.ECLK.Centered</li> </ul>	For display information only.
Data Path Delay	<ul style="list-style-type: none"> <li>• Bypass,</li> <li>• <b>Pre-defined</b></li> </ul>	Specify the delay implementation on data path delay. For Transmit, this option is fixed to Bypass.

Attribute	Selectable Values	Description
	<ul style="list-style-type: none"> <li>User-defined</li> <li>Dynamic</li> </ul>	<p><b>Bypass:</b> 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.</p> <p><b>Pre-defined:</b> Added delay component (DELAYH) with predefined value to delay the incoming DDR data before fed to IDDR block.</p> <p><b>User-defined:</b> Added delay component (DELAYH) on incoming DDR data path and allow user to configure how much delay needed.</p> <p><b>Dynamic:</b> Added dynamic delay component ( DELAYG ) on incoming DDR data path. User can dynamically control the delay value through signals.</p>
Fine Delay Value for User-defined	Range : DELAY0 to DELAY31 <b>Default : DELAY0</b>	Specify the delay step apply on DELAYH. Each step is about ~105ps  <b>Dependency on other attributes:</b> Available when (Data Path Delay) == User-defined
Clock Frequency for this Interface (MHz)	Range : 1-450 <b>Default : 100</b>	Specify the clock frequency of DDR interface
Bandwidth for this Interface (Mbits/s)	Derived	Bandwidth of this DDR interface. This option is display only.  <b>Formula:</b> $2 * (\text{Clock Frequency for this Interface}) * (\text{Bus Width for this Interface})$
Enable PLL Instantiation	<ul style="list-style-type: none"> <li><b>Unchecked</b></li> <li>Checked</li> </ul>	Enable to add PLL. This PLL is use to generate the DDR clock and its 90° phase shifted DDR clock.  <b>Dependency on other attributes:</b> Available when (Interface Type) == (Transmit    Transmit MIPI) && Clock to Data Relationship on the Pins == Centered
PLL Input Clock Frequency (MHz)	Range :10 – 400 <b>Default : 100</b>	Specify the reference clock of PLL.  <b>Dependency on other attributes:</b> Available when (PLL Instantiation) == Checked
PLL Output Clock Frequency Actual Value (MHz)	Derived	Only for display.

## 4. Signal Description

This section describes the ports of the GDDR I/O Module.

**Table 4.1. GDDR I/O Module Signals for Receive Interfaces**

Port Name	Clock Domain	Direction	Description
<b>DDR interface at I/O Pad</b>			
ddr_clk_i	—	Input	Serial DDR Clock input signal of DDR input interface.
ddr_data_i[N*K-1: 0]	ddr_clk_i	Input	Serial DDR Data input signal of DDR input interface. Bus width value range is [1, 128].
<b>Clock and Reset</b>			
rst_i	asynchronous	Input	Main reset signal. This signal is active HIGH.
dqsdll_rst_i	asynchronous	Input	Reset signal for DQSDLL. This signal is active HIGH. Only available for X2 and X4 aligned interfaces.
sclk_o	—	Output	System clock or parallel clock for parallel output data ( <i>data_o</i> ) to FPGA fabric. Use this clock to sample the parallel output data ( <i>data_o</i> ) of Receive interfaces.
sync_clk_i	—	Input	Slow clock for reset synchronization. The frequency is independent of other clocks, but it must be significantly lower than DDR input clock ( <i>ddr_clk_i</i> ).
<b>User Interface</b>			
alignwd_i	sclk_o	Input	Active HIGH word alignment control signal. This shifts word by one bit. Only available for X2 and X4 gearing ratio.
data_del_dyn_i[4:0]	asynchronous	Input	Dynamic delay control for DDR data path. Only available during dynamic data path delay. Each step is about ~105ps.
freeze_i	asynchronous	Input	Active HIGH DLL freeze signal. Set HIGH to freeze DLL component, and LOW to release DLL component.
sync_init_i	sync_clk_i	Input	Active HIGH init signal to initialize reset synchronization.
uddcntln_i	asynchronous	Input	Active LOW signal to hold or update control of delay code. For X1 gearing, you must assert this signal to LOW to instruct the DLL to delay the input DDR clock by 90° phase. For X2 and X4 gearing, <i>sync_init_i</i> signal helps to generate the <i>uddcntln</i> pulse internally to DLL block. This is optional for X2 and X4 gearing and you may tie this to HIGH. Only available for edge-aligned interfaces.
data_o [N*K-1: 0]	sclk_o	Output	Parallel data output of Receive interfaces.
rx_lock_o	ddr_clk_i	Output	Indicate DLL lock. Only available for edge-aligned interfaces.
rx_ready_o	sync_clk_i	Output	Indicate that the startup is completed, and RX circuit is ready to operate.
<b>MIPI User Interface</b>			
mipi_buf_clk_lp0o_i	—	Input	Input MIPI Low Power (LP) clock 0 from I/O. This signal is the input signal to bidirectional (BIDIR) I/O buffer of rx_mipi_clk_lp0_io.
mipi_buf_clk_lp0t_i	—	Input	Control signal of BIDIR I/O port of the rx_mipi_clk_lp0_io. This set the BIDIR I/O as input or output. 0 - Transmit

Port Name	Clock Domain	Direction	Description
			1 - Receive
mipi_buf_clk_lp1o_i	—	Input	Input MIPI LP clock 1 from I/O. This signal is the input signal to BIDIR I/O buffer of rx_mipi_clk_lp1_io.
mipi_buf_clk_lp1t_i	—	Input	Control signal of BIDIR I/O port of the rx_mipi_clk_lp1_io. This set the BIDIR I/O as input or output. 0 - Transmit 1 - Receive
mipi_buf_data_lp0o_i[N*K-1: 0]	—	Input	Input MIPI LP data 0 from I/O. This signal is the input signal to BIDIR I/O buffer of rx_mipi_data_lp0_io.
mipi_buf_data_lp0t_i[N*K-1: 0]	—	Input	Control signal of BIDIR I/O buffer of rx_mipi_data_lp0_io to set the BIDIR I/O buffer as input or output.
mipi_buf_data_lp1o_i[N*K-1: 0]	—	Input	Input MIPI LP data 1 from I/O. This signal is the input signal to BIDIR I/O buffer of rx_mipi_data_lp1_io.
mipi_buf_data_lp1t_i[N*K-1: 0]	—	Input	Control signal of BIDIR I/O buffer of rx_mipi_data_lp1_io to set the BIDIR I/O buffer as input or output.
mipi_buf_clk_lp0i_o	—	Output	Output MIPI LP clock 0 from I/O. This signal is the output signal to BIDIR I/O buffer of rx_mipi_clk_lp0_io.
mipi_buf_clk_lp1i_o	—	Output	Output MIPI LP clock 1 from I/O. This signal is the output signal to BIDIR I/O buffer of rx_mipi_clk_lp1_io.
mipi_buf_data_lp0i_o[N*K-1: 0]	—	Output	Output MIPI LP data 0 from I/O. This signal is the output signal to BIDIR I/O buffer of rx_mipi_data_lp0_io.
mipi_buf_data_lp1i_o[N*K-1: 0]	—	Output	Output MIPI LP data 1 from I/O. This signal is the output signal to BIDIR I/O buffer of rx_mipi_data_lp1_io.
mipi_clk_lp0_io	—	Bidirectional	MIPI Low Power (LP) clock 0 from or to I/O pad
mipi_clk_lp1_io	—	Bidirectional	MIPI LP clock 1 from or to I/O pad
mipi_data_lp0_io[N*K-1: 0]	—	Bidirectional	MIPI LP data 0 from or to I/O pad
mipi_data_lp1_io[N*K-1: 0]	—	Bidirectional	MIPI LP data 1 from or to I/O pad

**Note:** N = number of lanes/DDR bus width, K= 2(for x1 gearing), 4(for x2 gearing) and 8(for x4 gearing).

**Table 4.2. GDDR I/O Module Signals for Transmit Interfaces**

Port Name	Clock Domain	Direction	Description
<b>DDR interface at I/O Pad</b>			
ddr_clk_o	—	Output	Serial DDR clock output signal for DDR output interface.
ddr_data_o[N*K-1:0]	ddr_clk_o	Output	Serial DDR data output signal for DDR output interface. Bus width value range is [1, 128].
<b>Clock and Reset</b>			
rst_i	asynchronous	Input	Main reset signal. This signal is active HIGH.
clk_i	—	Input	Input DDR clock. Only available when PLL instantiation is disabled.
clk90_i	—	Input	90° shifted of input DDR clock ( <i>clk_i</i> ) signal. Only available for center-aligned interface and PLL instantiation is disable.
sclk_o	—	Output	System clock or parallel clock for input parallel data ( <i>data_i</i> ). Use this clock to sample the input parallel data.
sync_clk_i	—	Input	Slow clock for reset synchronization. The frequency is independent of the input clock, but it must be significantly lower.
pll_clki_i	—	Input	PLL reference clock. Only available when PLL instantiation is enabled
pll_rst_i	asynchronous	Input	Active HIGH PLL reset signal. Only available when PLL instantiation is enabled.
<b>User Interface</b>			
data_i[N*K-1:0]	sclk_o	Input	Input parallel data of the DDR interfaces.
lock_chk_i	asynchronous	Input	PLL lock signal. Connect this to PLL lock signal. Only available when centered mode is used with PLL instantiation is disabled.
tristate_i	asynchronous	Input	Control signal to tristate the DDR output interface. Set HIGH to tristate the DDR output interface. Set LOW when FPGA is driving the DDR output interface.
tx_lock_o	asynchronous	Output	PLL lock output signal. Only available when PLL instantiation is enabled.
tx_ready_o	sync_clk_i	Output	Indicate that startup is finished, and TX circuit is ready to operate.
<b>MIPI User Interface</b>			
mipi_buf_clkout_lp0o_i	—	Input	Input MIPI Low Power (LP) clock 0 from I/O. This signal is the input signal to bidirectional (BIDIR) I/O buffer of rx_mipi_clk_lp0_io.
mipi_buf_clkout_lp0t_i	—	Input	Control signal of BIDIR I/O port of the rx_mipi_clk_lp0_io. This set the BIDIR I/O as input or output. 0 - Transmit 1 - Receive
mipi_buf_clkout_lp1o_i	—	Input	Input MIPI LP clock 1 from I/O. This signal is the input signal to BIDIR IO buffer of rx_mipi_clk_lp1_io.
mipi_buf_clkout_lp1t_i	—	Input	Control signal of BIDIR I/O port of the rx_mipi_clk_lp1_io. This set the BIDIR I/O as input or output. 0 - Transmit 1 - Receive
mipi_buf_dout_lp0o_i [n*k-1:0]	—	Input	Input MIPI LP data 0 from I/O. This signal is the input signal to BIDIR IO buffer of rx_mipi_data_lp0_io.
mipi_buf_dout_lp0t_i[N*K-1:0]	—	Input	Control signal of BIDIR IO buffer of rx_mipi_data_lp0_io to set the BIDI IO buffer as input or output.

Port Name	Clock Domain	Direction	Description
mipi_buf_dout_lp1o_i[N*K-1: 0]	—	Input	Input MIPI LP data 1 from I/O. This signal is the input signal to BIDIR IO buffer of rx_mipi_data_lp1_io.
mipi_buf_dout_lp1t_i[N*K-1: 0]	—	Input	Control signal of BIDIR IO buffer of rx_mipi_data_lp1_io to set the BIDIR IO buffer as input or output.
mipi_buf_clkout_lp0i_o	—	Output	Output MIPI LP clock 0 from I/O. This signal is the output signal to BIDIR IO buffer of rx_mipi_clk_lp0_io.
mipi_buf_clkout_lp1i_o	—	Output	Output MIPI LP clock 1 from I/O. This signal is the output signal to BIDIR IO buffer of rx_mipi_clk_lp1_io.
mipi_buf_dout_lp0i_o[N*K-1: 0]	—	Output	Output MIPI LP data 0 from I/O. This signal is the output signal to BIDIR IO buffer of rx_mipi_data_lp0_io.
mipi_buf_dout_lp1i_o[N*K-1: 0]	—	Output	Output MIPI LP data 1 from I/O. This signal is the output signal to BIDIR IO buffer of rx_mipi_data_lp1_io.
mipi_clkout_lp0_io	—	Bidirectional	MIPI Low Power (LP) clock 0 from/to I/O pad
mipi_clkout_lp1_io	—	Bidirectional	MIPI LP clock 1 from or to I/O pad
mipi_dout_lp0_io[N*K-1: 0]	—	Bidirectional	MIPI LP data 0 from or to I/O pad
mipi_dout_lp1_io[N*K-1: 0]	—	Bidirectional	MIPI LP data 1 from or to I/O pad

**Note:** N = number of lanes/DDR bus width, K = 2(for x1 gearing), 4(for x2 gearing), and 8(for X4 gearing).

## 5. Designing with the IP

This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the [Lattice Radiant Software User Guide](#).

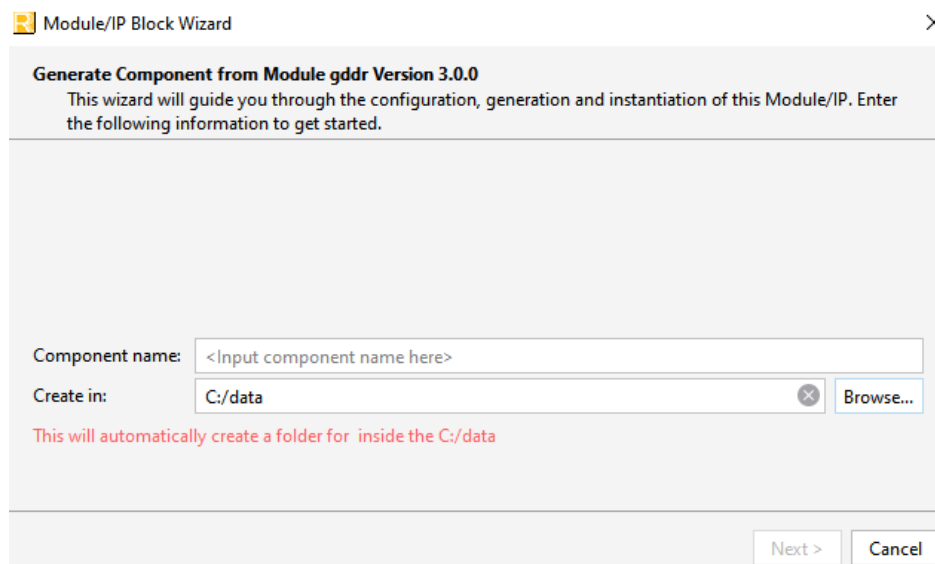
**Note:** The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

### 5.1. Generating and Instantiating the IP

You can use Lattice Radiant software to generate IP modules and integrate them into the device architecture. The steps below describe how to generate the GDDR I/O Module in the Lattice Radiant software.

To generate the GDDR I/O Module:

1. Create a new Lattice Radiant software project or open an existing project.
2. Click the **IP Catalog** button to view the **IP Catalog** pane.
3. On the **IP on Local** tab, double-click **GDDR\_Generic** under **Module, Architecture\_Modules, IO** category. The **Module/IP Block Wizard** opens as shown in [Figure 5.1](#). Enter values in the **Component name** and the **Create in** fields and click **Next**.



**Figure 5.1. Module/IP Block Wizard**

4. In the next **Module/IP Block Wizard** window, customize the selected IP using drop-down lists and check boxes. [Figure 5.2](#) shows an example configuration of the GDDR I/O Module. For details on the configuration options, refer to the Table 3.1. General Attributes section.



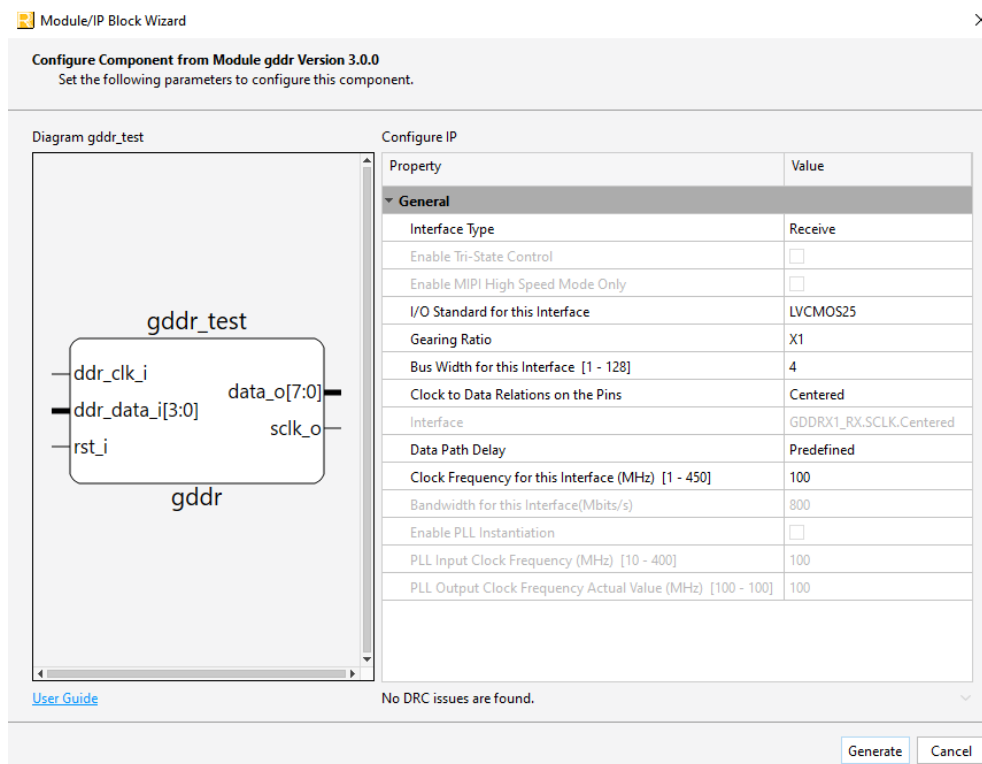


Figure 5.2. IP Configuration

- Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 5.3.

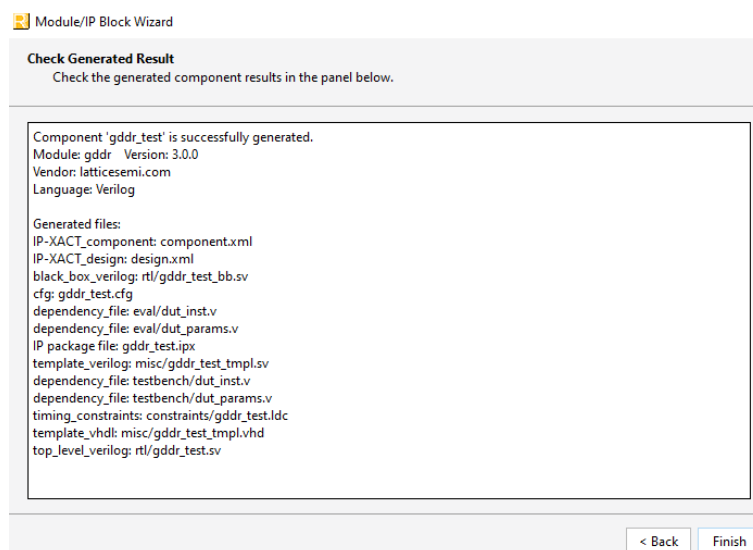


Figure 5.3. Check Generated Result

- Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 5.1.

### 5.1.1. Generated Files and File Structure

The generated GDDR I/O Module package includes the closed-box (<Component name>\_bb.v) and instance templates (<Component name>\_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in [Table 5.1](#).

**Table 5.1. Generated File List**

Attribute	Description
<Component name>.ipx	Contains the information on the files associated to the generated IP.
<Component name>.cfg	Contains the parameter values used in IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	Provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	Provides the synthesis closed-box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	Provide instance templates for the module.
testbench/tb_top.v	Top testbench module for IP simulation.
constraint/<Component name>.ldc	IP constraint file.

## 5.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC File.

Post-Synthesis constraint files (.pdc) contain both timing and non-timing *constraint.pdc* source files for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the [Lattice Radiant Software User Guide](#) for more information on how to create or edit constraints and how to use the Device Constraint Editor.

## 5.3. Timing Constraints

User must define the clock constraints for these clock signals: *sync\_clk\_i*, *pll\_clki\_i*, *clk\_i*, *clk90i*, and *ddr\_clk\_i* in user active PDC constraint file. Refer to example below.

```
#For sync_clk_i with 10MHz
# - IF gearing > 1
create_clock -name {sync_clk_i} -period 100 [get_ports sync_clk_i]

#For pll_clki_i with 100MHz
# - IF PLL instantiation enabled
create_clock -name {pll_clki_i} -period 10 [get_ports pll_clki_i]

# For clk_i and clk90_i with 100MHz
# - IF TX || TX_MIIPI interface
create_clock -name {clk_i} -period 10 [get_ports clk_i]
# - IF center-aligned interface
create_clock -name {clk90_i} -period 10 -waveform [list 2.5 7.5] [get_ports {clk90_i}]

# For ddr_clk_i with 100MHz
# - IF RX || RX_MIIPI interface
create_clock -name {ddr_clk_i} -period 10 [get_ports {ddr_clk_i}]
```

## 5.4. Physical Constraints

The I/O standard constraints for DDR input and output interface are automatically set through IP *constraint.ldc* file based on selected I/O Standard attribute in IP GUI. For *Interface Type == Receive MIPI or Transmit MIPI*, MIPI and MIPI\_LP I/O standard will be applied. Refer to IP's *constraint.ldc* for details.


## 5.5. Specifying the Strategy

The Radiant software provides two predefined strategies: Area and Timing. It also enables you to create customized strategies. For details on how to create a new strategy, refer to the Strategies section of the [Lattice Radiant Software User Guide](#).

## 5.6. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in [Figure 5.4](#).

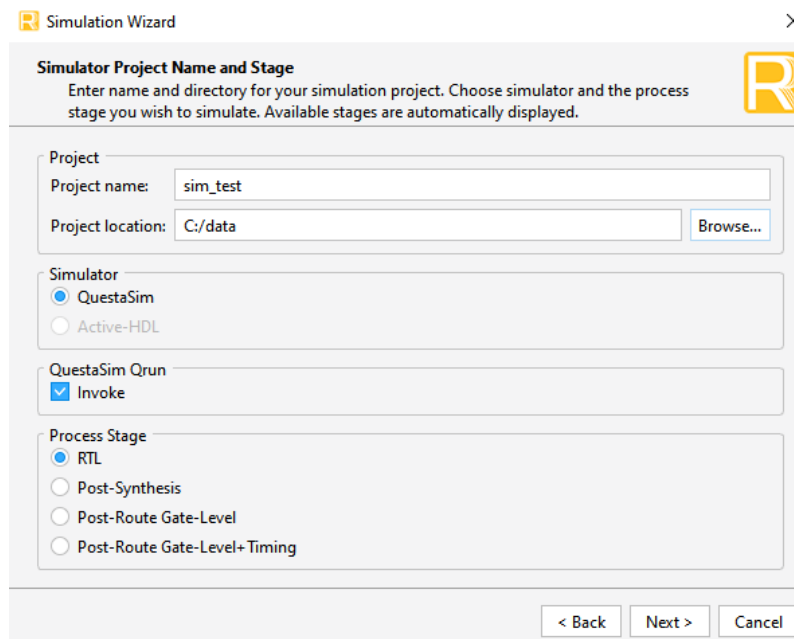


Figure 5.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in [Figure 5.5](#).

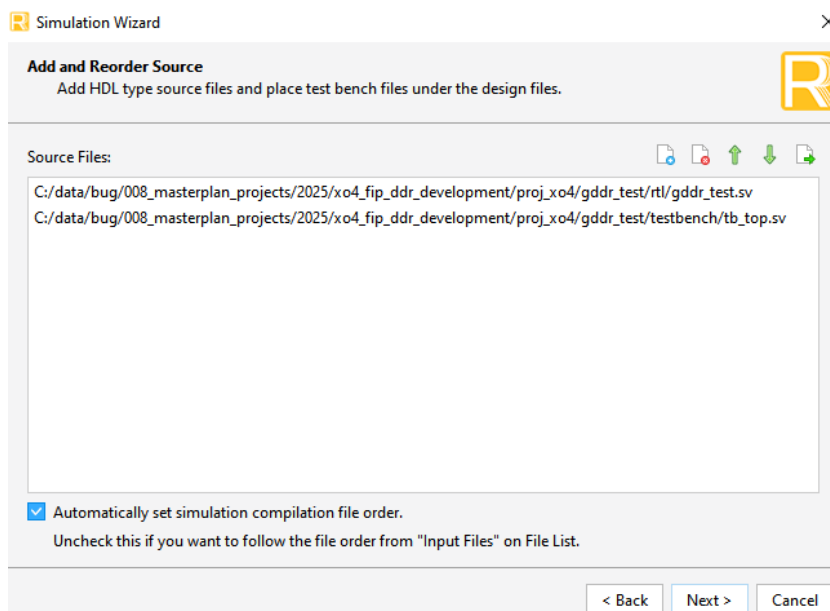


Figure 5.5. Add and Reorder Source

3. Select **tb\_top** as simulation top module. Then click **Next**.

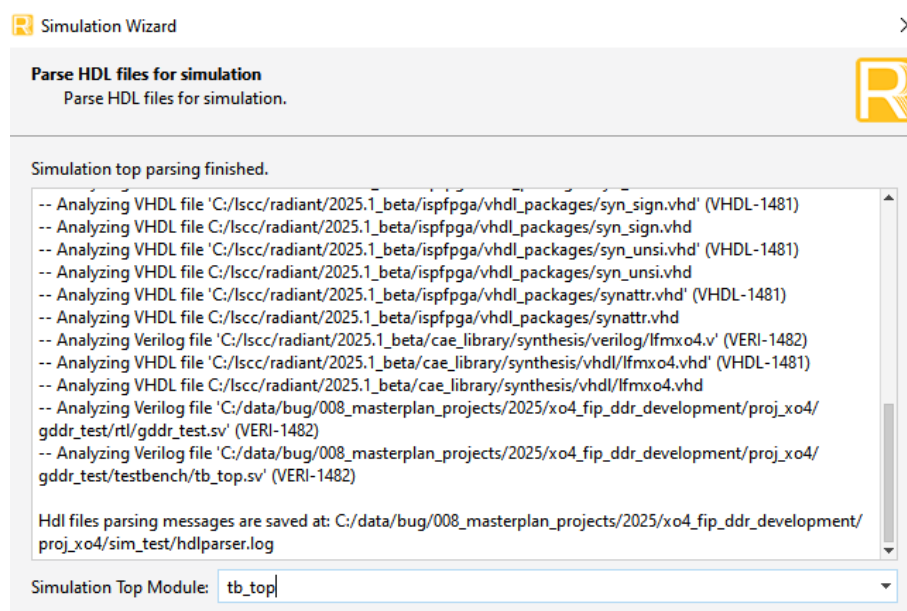


Figure 5.6. Select simulation Top module

4. Click **Next**. The **Summary** window is shown.
5. Click **Finish** to run the simulation.

### 5.6.1. Simulation Results

```

# +-----+
# TEST[0] - Data Comparison
# +-----+
# check if send and received data are matching
#
# Flow - Generating Data
#
#         Number of data generated = 10
#
# Flow - Transmitting Data
# Flow - Sampling Data
# Flow - Comparing Actual Received Data vs Expected Data
#
#         DATA[0] expected: 0x24 received: 0x24 -- OK
#         DATA[1] expected: 0x81 received: 0x81 -- OK
#         DATA[2] expected: 0x09 received: 0x09 -- OK
#         DATA[3] expected: 0x63 received: 0x63 -- OK
#         DATA[4] expected: 0x0d received: 0x0d -- OK
#         DATA[5] expected: 0x8d received: 0x8d -- OK
#         DATA[6] expected: 0x65 received: 0x65 -- OK
#         DATA[7] expected: 0x12 received: 0x12 -- OK
#         DATA[8] expected: 0x01 received: 0x01 -- OK
#         DATA[9] expected: 0x0d received: 0x0d -- OK
#
# +-----+
# TEST[1] - RX SYNC INIT Signal
# +-----+
# check if rx_ready_o signal is de-assert when rx_sync_init is set to 0
#
# Flow - Set sync_init_i signal to LOW (sync_init_i = 0 )
#
#         rx_ready_o goes LOW -- OK
#
# ***** SIMULATION PASSED *****

```

**Note:** IP simulation is expected to fail for gate-level simulation. This is because routing delay is not taken account in the IP testbench. Thus, the sampled data for data comparison might be incorrect.

## Appendix A. Resource Utilization

Table A.1 shows a sample resource utilization of the GDDR I/O Module for MachXO4 device (LFMXO4-110HE-5BBG484C) using Radiant 2025.2 with Synopsys® Synplify Pro® for Lattice.

**Table A.1. Resource Utilization**

IP Configuration	LUTs	Registers	Others
Receive x1 (Alignment = Centered, Bus width = 4 )	0	0	4 IDDRX1
Receive x2 (Alignment = Centered, Bus width = 4 )	7	23	4 IDDRX2 1 CLKDIV 1 ECLKSYNC
Receive x4 (Alignment = Centered, Bus width = 4 )	7	39	4 IDDRX4 1 CLKDIV 1 ECLKSYNC
Transmit x1 (Alignment = Centered, Bus width = 4 )	2	0	5 ODDR1
Transmit x2 (Alignment = Centered, Bus width = 4 )	3	4	5 ODDR2 1 CLKDIV 2 ECLKSYNC
Transmit x4 (Alignment = Centered, Bus width = 4 )	3	4	5 ODDR4 1 CLKDIV 2 ECLKSYNC

## References

- [MachXO4 DDR Generic Module Release Notes \(FPGA-RN-02094\)](#)
- [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#)
- [MachXO4](#) web page
- [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#)
- [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](http://www.latticesemi.com/techsupport).

For frequently asked questions, please refer to the Lattice Answer Database at [www.latticesemi.com/Support/AnswerDatabase](http://www.latticesemi.com/Support/AnswerDatabase).



## Revision History

**Note:** In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

### Revision 1.0, IP v3.0.0, September 2025

Section	Change Summary
All	Initial release.



[www.latticesemi.com](http://www.latticesemi.com)