DDR Generic Module

User Guide

FPGA-IPUG-02057-1.9

June 2024

Disclaimers

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

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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
FPGA	Field Programmable Gate Array
GDDR	Generic Double Data Rate
I/O	Input/Output
RTL	Register Transfer Level
SDR	Single Data Rate

1. Introduction

The Lattice Semiconductor Generic Double Data Rate Input/Output (GDDR I/O) Module is designed to be used in a wide range of applications in which high-speed data transfer is required.

1.1. Features

The key features of Generic Double Data Rate Input/Output (GDDR I/O) Module include:

- Receive and Transmit Interface up to 1500 Mbps
- Supported gearing: X1, X2, X4, X5
- Selectable I/O type
  - Single-ended or Differential Signaling
- 1-bit to 256-bit data bus width
- 100 MHz to 750 MHz clock frequency
  - 100 MHz to 250 MHz for X1 Gearing
  - 100 MHz to 500 MHz for X2 Gearing
  - 100 MHz to 750 MHz for X4 and X5 Gearing
- Clock-data relationship options:
  - Edge-to-edge
  - Centered
- Data Path Delay that includes following options:
  - Bypass
  - Static Default (Receive Interface only)
  - Dynamic Default (Receive Interface only)
  - Static User-defined
  - Dynamic User-defined
- Includes GDDR_SYNC soft IP logic to be used by the following configurations:
  - For Receive, Centered, X2, X4, X5
  - For Transmit, X2, X4, X5
- Includes RX_SYNC soft IP logic to be used by the following configurations:
  - For Receive, Aligned, X2, X4, X5
- Tri-state control (Transmit Interface only)

1.2. Conventions

1.2.1. Nomenclature

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

1.2.2. Signal Names

Signal names that end with:

- _*n* are active low
- _*i* are input signals
- _*o* are output signals
- _*io* are bidirectional input/output signals

2. Functional Description

2.1. Overview

The GDDR I/O Module is directly connected to the memory interface providing all required DDR ports for memory access. It converts the single data rate (SDR) data to DDR data for write operations and performs the DDR to SDR conversion for read operations. This I/O module utilizes the dedicated FPGA DDR I/O logic and is designed to reliably drive and capture data on the memory interface. DDR interfaces capture data on both rising and falling edges of the clock, thus doubling performance. Table 2.1 provides a summary of GDDR I/O Interface. Refer to Refer to FPGA-TN-02244 CertusPro-NX High-Speed I/O Interface for more details.

Table 2.1. Available GDDR I/O Module Interfaces

Feature	Description	Comments
GDDRX1_RX.SCLK.Centered	Generic DDR 2:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay.
GDDRX1_RX.SCLK.Aligned	Generic DDR 2:1 Receive Aligned Interface	Supports bypassed, static, and dynamic data path delay. Supports dynamic clock path delay. Required RX_SYNC support soft logic. Using DDRDLL and DLLDEL.
GDDRX2_RX.ECLK.Centered	Generic DDR 4:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. Required GDDR_SYNC support soft logic.
GDDRX2_RX.ECLK.Aligned	Generic DDR 4:1 Receive Aligned Interface	Supports bypassed, static, and dynamic data path delay. Supports dynamic clock path delay. Required RX_SYNC support soft logic. Using DDRDLL and DLLDEL.
GDDRX4_RX.ECLK.Centered	Generic DDR 8:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. Required GDDR_SYNC support soft logic.
GDDRX4_RX.ECLK.Aligned	Generic DDR 8:1 Receive Aligned Interface	Supports bypassed, static, and dynamic data path delay. Supports dynamic clock path delay. Required RX_SYNC support soft logic. Using DDRDLL and DLLDEL.
GDDRX5_RX.ECLK.Centered	Generic DDR 10:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. Required GDDR_SYNC support soft logic.
GDDRX5_RX.ECLK.Aligned	Generic DDR 10:1 Receive Aligned Interface	Supports bypassed, static, and dynamic data path delay. Supports dynamic clock path delay. Required RX_SYNC support soft logic. Using DDRDLL and DLLDEL.
GDDRX1_TX.SCLK.Centered	Generic DDR 2:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control.
GDDRX1_TX.SCLK.Aligned	Generic DDR 2:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control.
GDDRX2_TX.ECLK.Centered	Generic DDR 4:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.
GDDRX2_TX.ECLK.Aligned	Generic DDR 4:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.
GDDRX4_TX.ECLK.Centered	Generic DDR 8:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.
GDDRX4_TX.ECLK.Aligned	Generic DDR 8:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.
GDDRX5_TX.ECLK.Centered	Generic DDR 10:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.
GDDRX5_TX.ECLK.Aligned	Generic DDR 10:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. Required GDDR_SYNC support soft logic.

Table 2.2 shows the summary of GDDR Support Soft Logic.

The following Soft Logic Modules can be utilized by the Generic DDR upon startup for well-defined synchronization. For interface with gearing greater than X1, GDDR_SYNC is required for all Receive Centered and Transmit interfaces; while, RX_SYNC is required for all Receive Aligned Interfaces.

Table 2.2. Summary of GDDR Support Soft Logic

Feature	Description
GDDR_SYNC	Needed to tolerate large skew between stop and reset input.
RX_SYNC	Used to break up the DDRDLL to DLLDEL clock loop for aligned interface.

The following notes apply to Table 2.1 and Table 2.2:

- G – Generic
- _RX – Receive interface
- _TX – Transmit interface
- .SCLK – Uses SCLK (primary clock) clocking resource
- .ECLK – Uses ECLK (edge clock) clocking resource
- DDRX1 – DDR X1 gearing I/O Register
- DDRX2 – DDR X2 gearing I/O Registers
- DDRX4 – DDR X4 gearing I/O Registers
- DDRX5 – DDR X5 gearing I/O Registers
- .Centered – Clock is centered to the data when coming into the device
- .Aligned – The clock is coming in edge aligned to the Data

Figure 2.1 illustrates top-level design of GDDR I/O Soft IP.

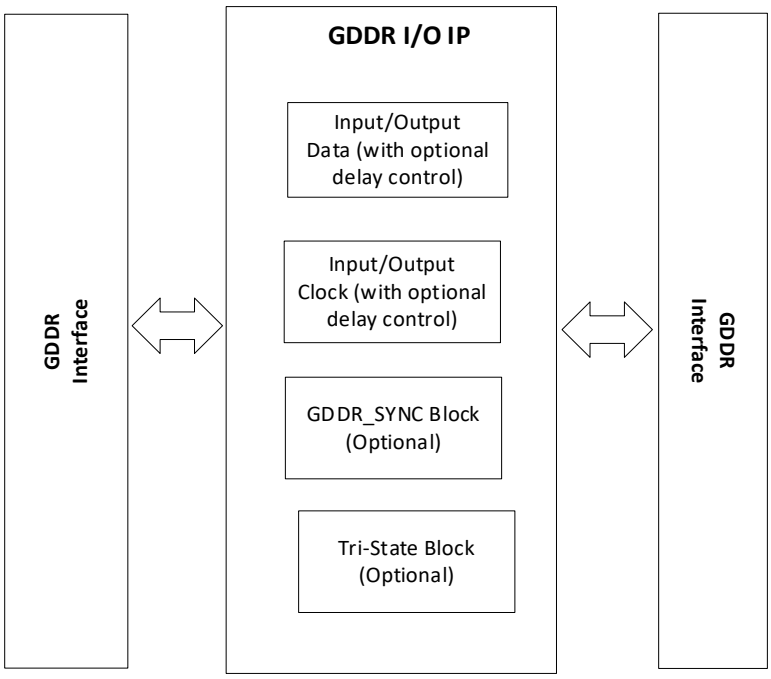


Figure 2.1. GDDR I/O Module Top-level Block Diagram

2.2. Functional Diagrams

2.2.1. GDDRX1_RX.SCLK.Centered

This is a generic receive interface using X1 gearing and SCLK. The input clock is centered relative to the data. This interface can be used for DDR data rates below 400 Mb/s.

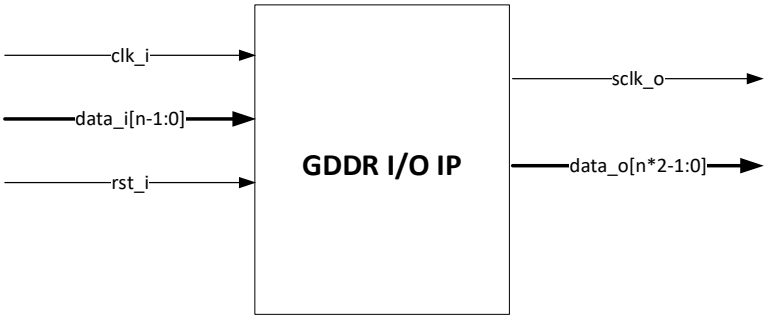


Figure 2.2. GDDRX1_RX.SCLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram

Based on Figure 2.2, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Bypass and Static Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Static User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Output sampling clock, sclk_o, is routed from clk_i.

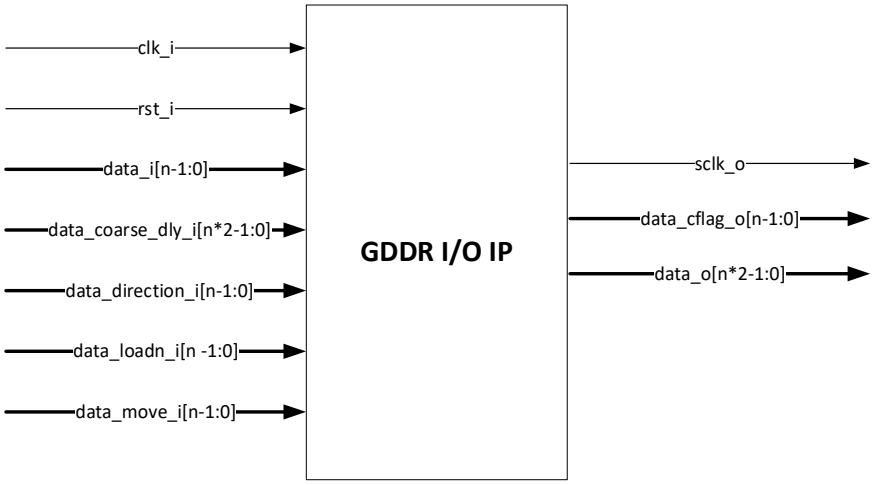


Figure 2.3. GDDRX1_RX.SCLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram

Based on Figure 2.3, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- Dynamic default data path delay setting can be used to control the delay on data dynamically through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals. Choosing dynamic user-defined data path delay adds fine delay value on the data path delay.
- Output sampling clock, sclk_o, is routed from clk_i.

2.2.2. GDDRX1_RX.SCLK.Aligned

This is a generic receive interface using X1 gearing and SCLK. The input clock is edge aligned to the data. This interface can be used for DDR data rates up to 400 Mb/s.

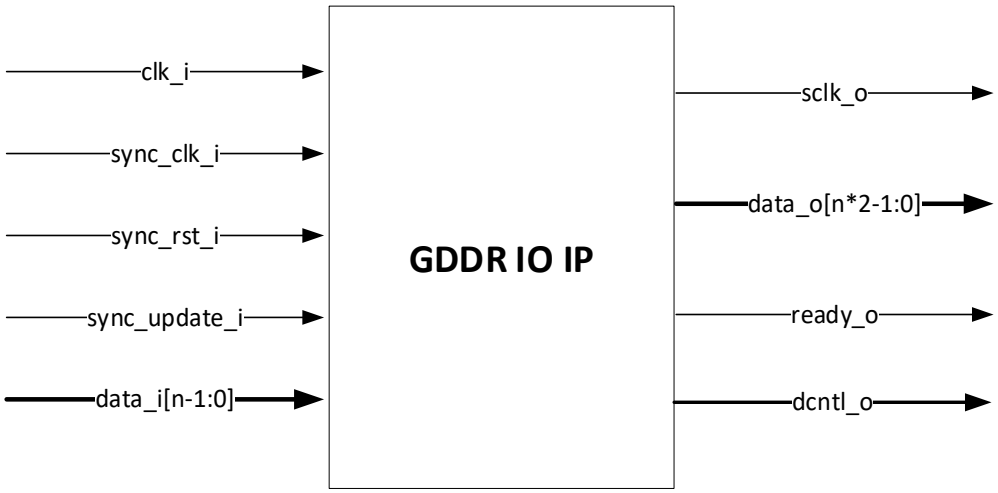


Figure 2.4. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram

Based on Figure 2.4, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Bypass and Static Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Static User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Clock delay component is used on this interface to phase shift the incoming clk_i to produce the sclk_o.
- RX_SYNC Soft IP module used on this interface generates reset sequence on all the IP components for proper synchronization. The ready_o signal is then asserted to indicate that the interface is ready to operate.

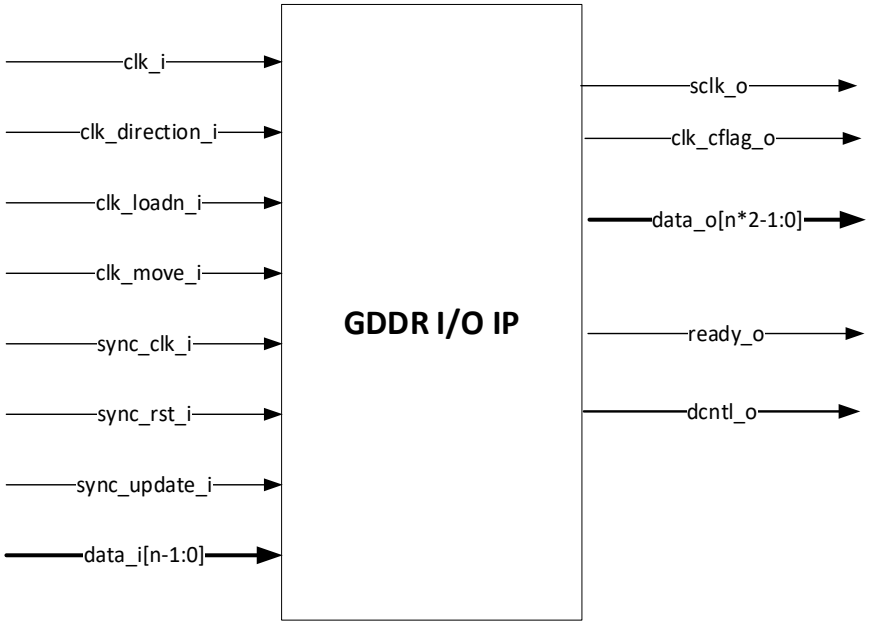


Figure 2.5. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User Defined Delay with Dynamic (Clock) Delay Block Diagram

Based on Figure 2.5, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Bypass and Static Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Static User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Clock delay component is used on this interface to phase shift the incoming clk_i to produce the sclk_o.
- RX_SYNC Soft IP module used on this interface generates reset sequence on all the IP components for proper synchronization. The ready_o signal is then asserted to indicate that the interface is ready to operate.
- Dynamic clock path delay setting can be used to control the delay on clock dynamically through clk_direction_i, clk_loadn_i, and clk_move_i input signals.

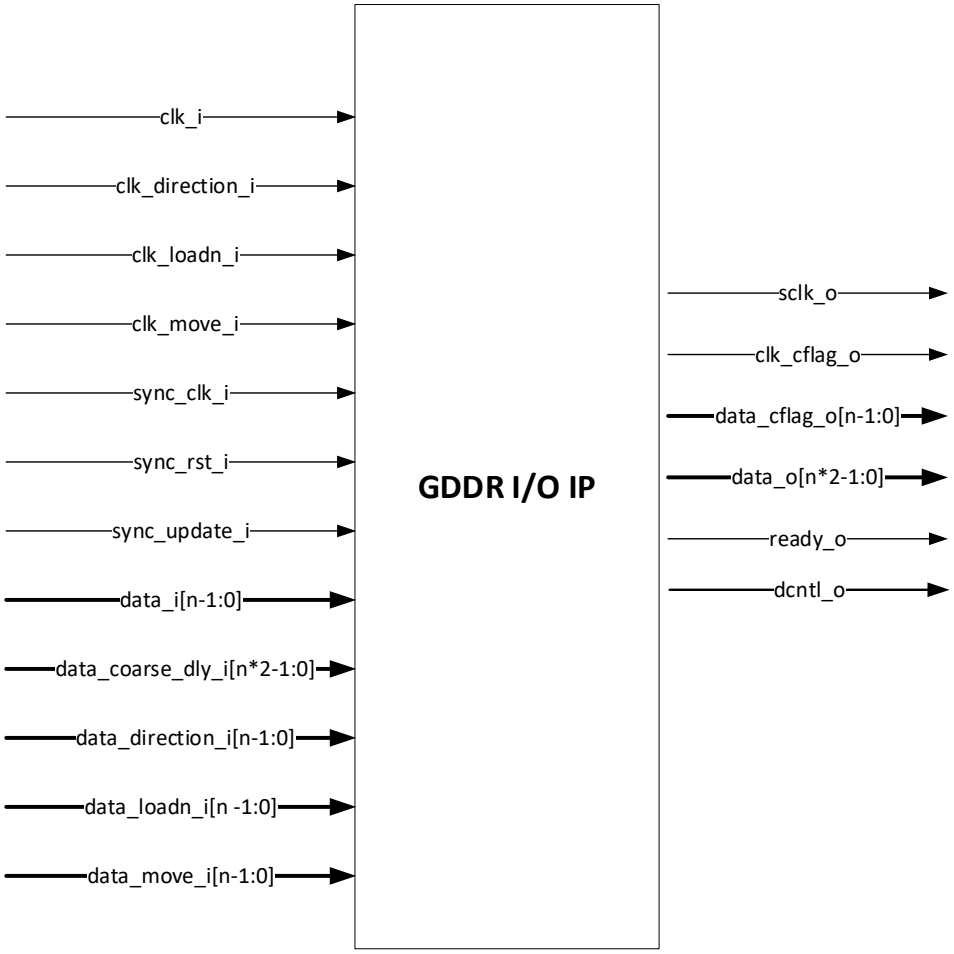


Figure 2.6. GDDRX1_RX.SCLK.Aligned Dynamic Default (Data)/Dynamic User-defined Delay (Clock and Data) Block Diagram

Based on Figure 2.6, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Dynamic Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Dynamic User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Clock delay component is used on this interface to phase shift the incoming clk_i to produce the sclk_o.
- RX_SYNC Soft IP module used on this interface generates reset sequence on all the IP components for proper synchronization. The ready_o signal is then asserted to indicate that the interface is ready to operate.
- Dynamic default data path delay setting can be used to control the delay on data dynamically through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals. Choosing dynamic user-defined data path delay adds coarse delay value on the data path delay.
- Dynamic clock path delay setting can be used to control the delay on clock dynamically through clk_direction_i, clk_loadn_i, and clk_move_i input signals.

2.2.3. GDDRX2/4/5_RX.ECLK.Centered

These are generic receive interfaces using X2, X4, or X5 gearing and Edge Clock Tree (ECLK). The input clock is centered relative to the data. These interfaces must be used for DDR data rates above 400 Mb/s.

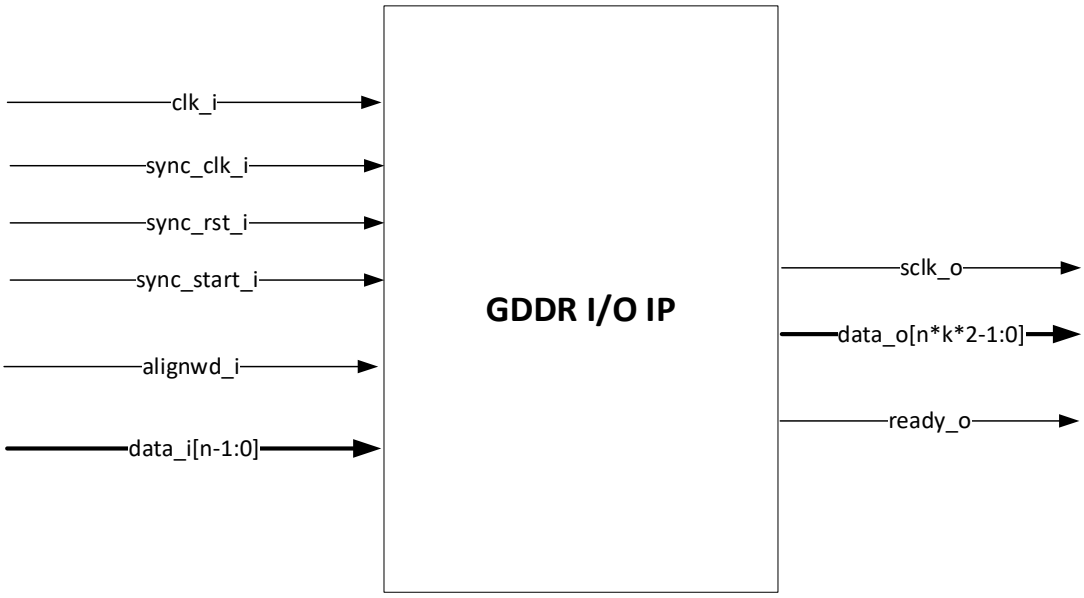


Figure 2.7. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram

Based on Figure 2.7, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. Static User-defined interface has a fine delay value setting to override the data path delay.
- The incoming clock, clk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- GDDR_SYNC is automatically included in this configuration which is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data components. The ready_o signal is asserted after synchronization is done to indicate that the interface is ready to operate.

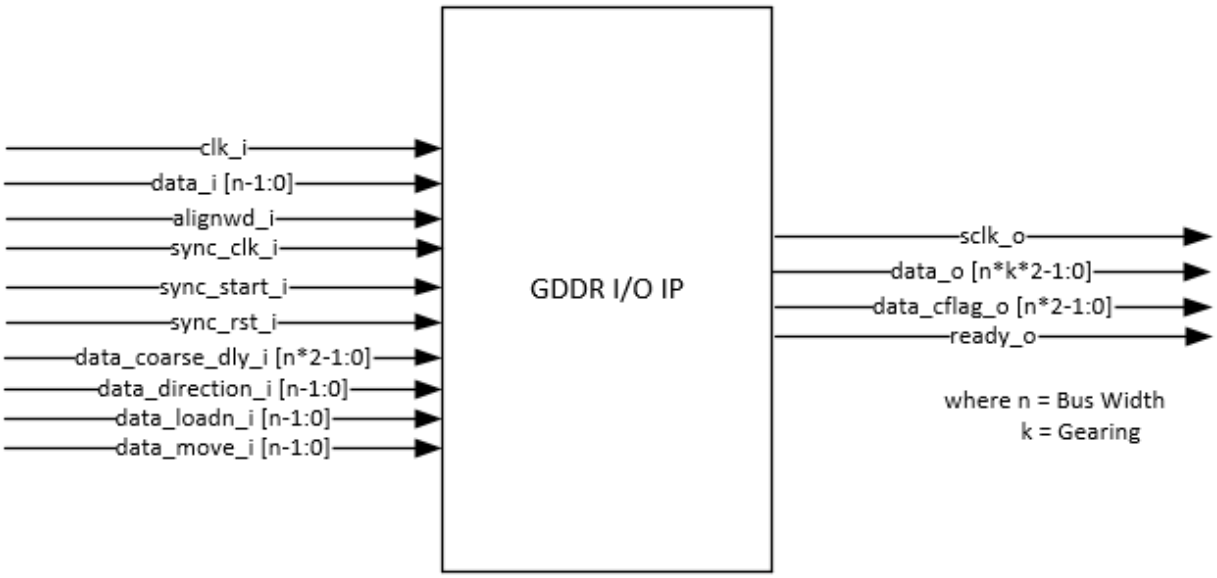


Figure 2.8. GDDRX2/4/5_RX.ECLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram

Based on Figure 2.8, interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- Dynamic default data path delay setting can be used to control the delay on data dynamically through data_direction_i, data_loadn_i, and data_move_i input signals. Choosing dynamic user-defined data path delay adds fine delay value on the data path delay.
- The incoming clock, clk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- GDDR_SYNC is automatically included in this configuration which is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data components. The ready_o signal is asserted after synchronization is done to indicate that the interface is ready to operate.

2.2.4. GDDRX2/4/5_RX.ECLK.Aligned

These are generic receive interfaces using X2, X4, or X5 gearing and Edge Clock Tree (ECLK). The input clock is edge aligned to the data. These interfaces must be used for DDR data rates above 400 Mb/s.

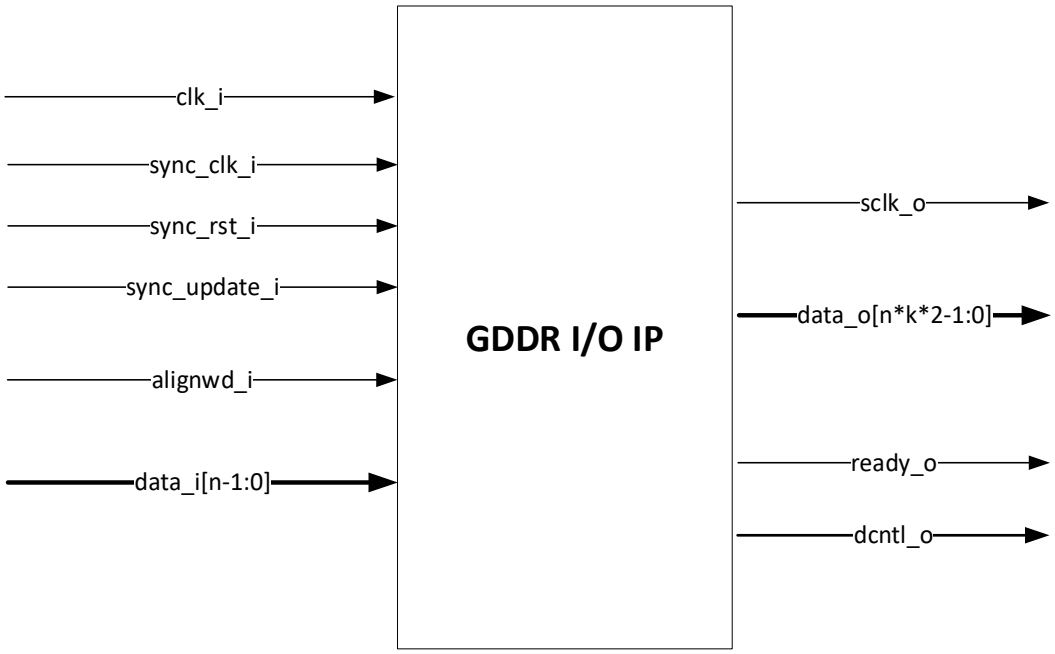


Figure 2.9. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram

Based on Figure 2.9, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Bypass and Static Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Static User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Clock delay component is used on this interface to phase shift the incoming clk_i. Then, it is divided into gearing setting on the clock divider component to generate the sclk_o.
- RX_SYNC Soft IP module used on this interface generates reset sequence on all the IP components for proper synchronization. The ready_o signal is then asserted to indicate that the interface is ready to operate.

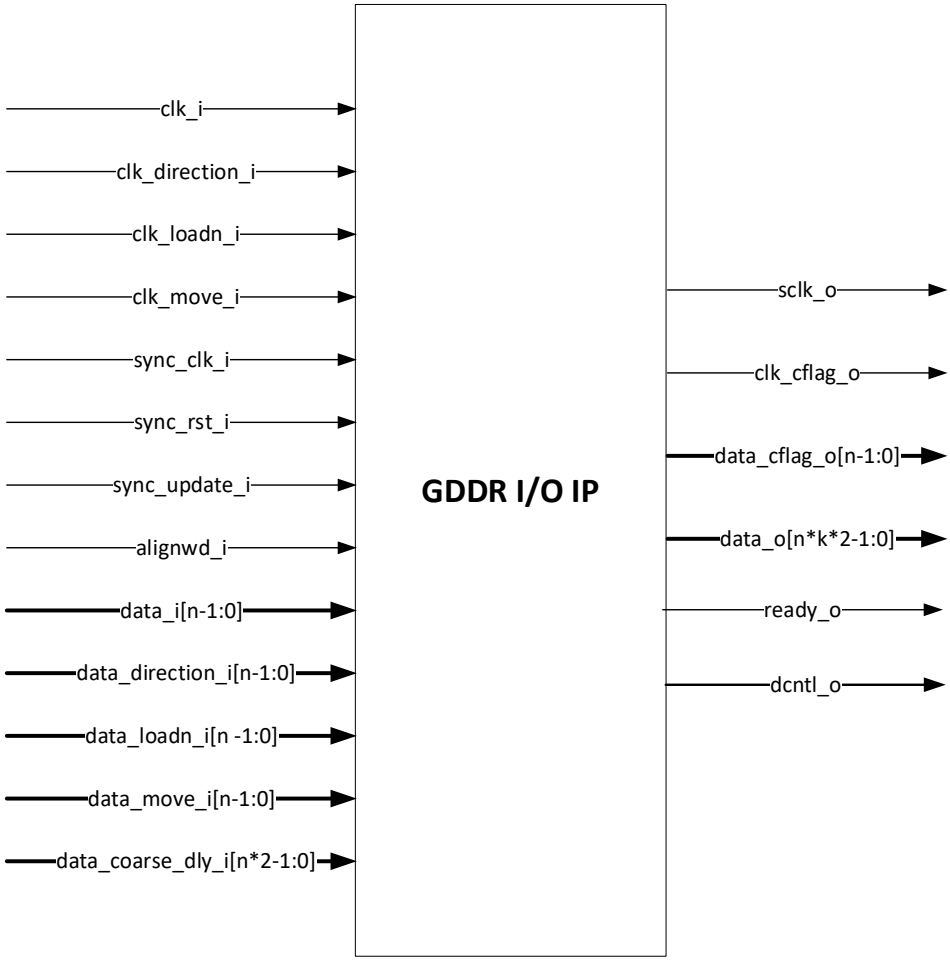


Figure 2.10. GDDRX2/4/5_RX.ECLK.Aligned Dynamic Default/Dynamic User-defined Delay Block Diagram

Based on Figure 2.10, interface function is described below:

- On this interface, the input data, data_i, is being captured initially by an IP delay component and then processed by DDR data component.
- The delay component behaves depending on the selected data path delay setting. For Dynamic Default, fine delay value is automatically set to 0 and the coarse delay value is set to 0 ns, while Dynamic User-defined interface has a fine delay value and a coarse delay value setting to override the data path delay.
- Clock delay component is used on this interface to phase shift the incoming clk_i. Then, it is divided into gearing setting on the clock divider component to generate the sclk_o.
- RX_SYNC Soft IP module used on this interface generates reset sequence on all the IP components for proper synchronization. The ready_o signal is then asserted to indicate that the interface is ready to operate.
- Dynamic default data path delay setting can be used to control the delay on data dynamically through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals. Choosing dynamic user-defined data path delay adds coarse delay value on the data path delay.
- Dynamic clock path delay setting can be used to control the delay on clock dynamically through clk_direction_i, clk_loadn_i, and clk_move_i input signals.

2.2.5. GDDRX1_TX.SCLK.Centered

This is a generic transmit interface using X1 gearing and SCLK. The input clock is centered relative to the data. This interface can be used for DDR data rates below 400 Mb/s.

The clock used to generate the clock output is delayed 90 degrees to center the data at the output side.

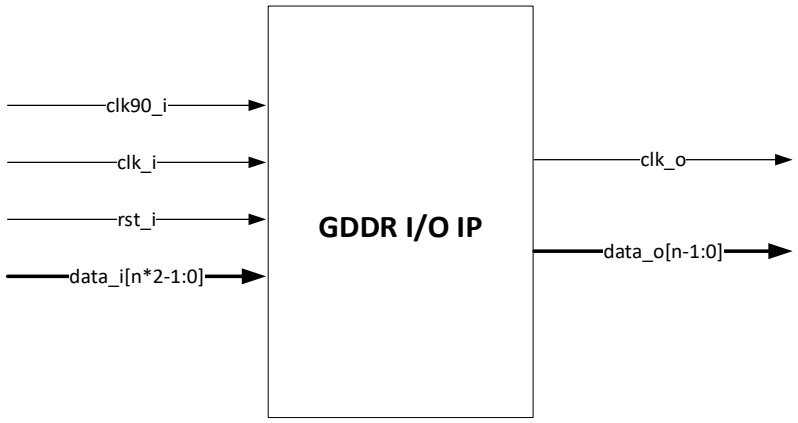


Figure 2.11. GDDRX1_TX.SCLK.Centered Bypass/Static User-defined Delay Block Diagram

Based on Figure 2.11, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value setting to override the delay of the data path.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk90_i.

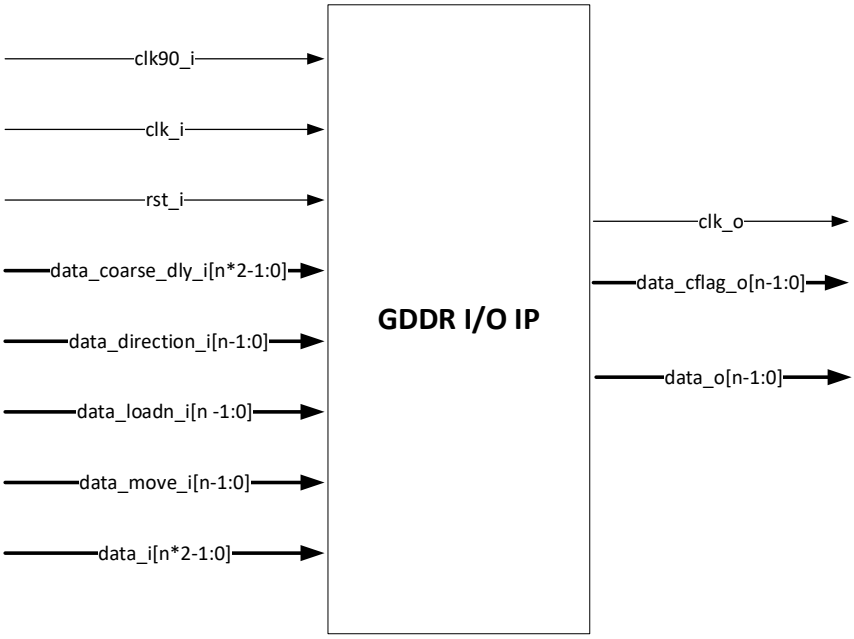


Figure 2.12. GDDRX1_TX.SCLK.Centered Dynamic User-defined Delay Block Diagram

- On this interface, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component.
- Dynamic user-defined data path delay adds fine delay value on the data path delay. Data delay can also be dynamically controlled through data_direction_i, data_loadn_i, and data_move_i input signals.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk90_i. Data is sampled by clk_i.

2.2.6. GDDRX1_TX.SCLK.Aligned

This is a generic transmit interface using X1 gearing and SCLK. The input clock is edge aligned to the data. This interface can be used for DDR data rates up to 400 Mb/s.

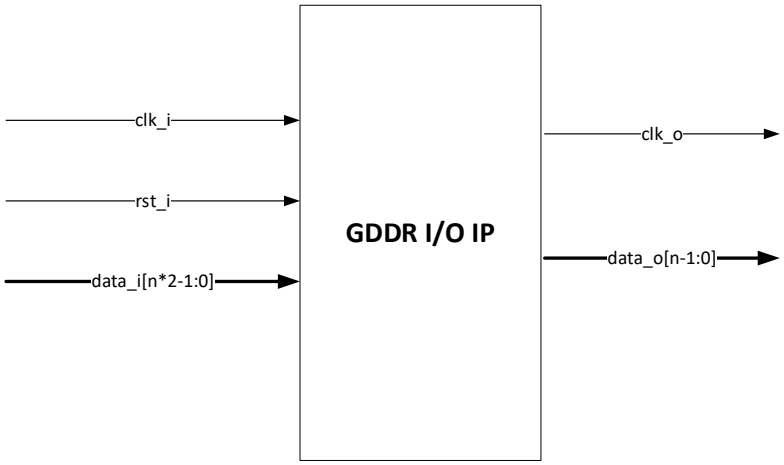


Figure 2.13. GDDRX1_TX.SCLK.Aligned Bypass/Static User-defined Delay Block Diagram

Based on Figure 2.13, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value setting to override the data path delay.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk_i. Both data and clock DDR components share common input clk_i.

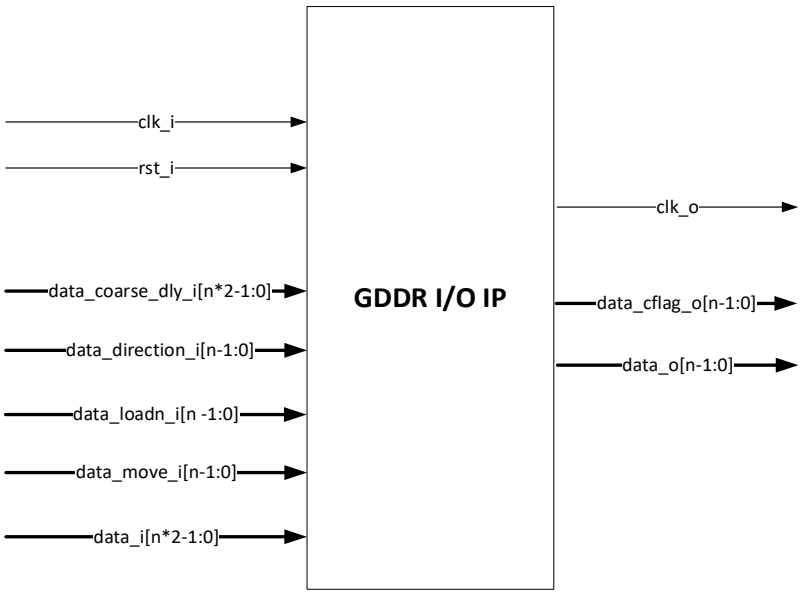


Figure 2.14. GDDRX1_TX.SCLK.Aligned Dynamic User-defined Delay Block Diagram

Based on Figure 2.14, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component.

- Dynamic user-defined data path delay adds fine delay value on the data path delay. Data delay can also be dynamically controlled through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk_i. Both data and clock DDR components share common input clk_i.

2.2.7. GDDRX2/4/5_TX.ECLK.Centered

These are generic transmit interfaces using X2, X4, or X5 gearing and Edge Clock Tree (ECLK). The input clock is centered relative to the data. These interfaces must be used for DDR data rates above 400 Mb/s.

The clock used to generate the clock output is delayed 90 degrees to center the data at the output side.

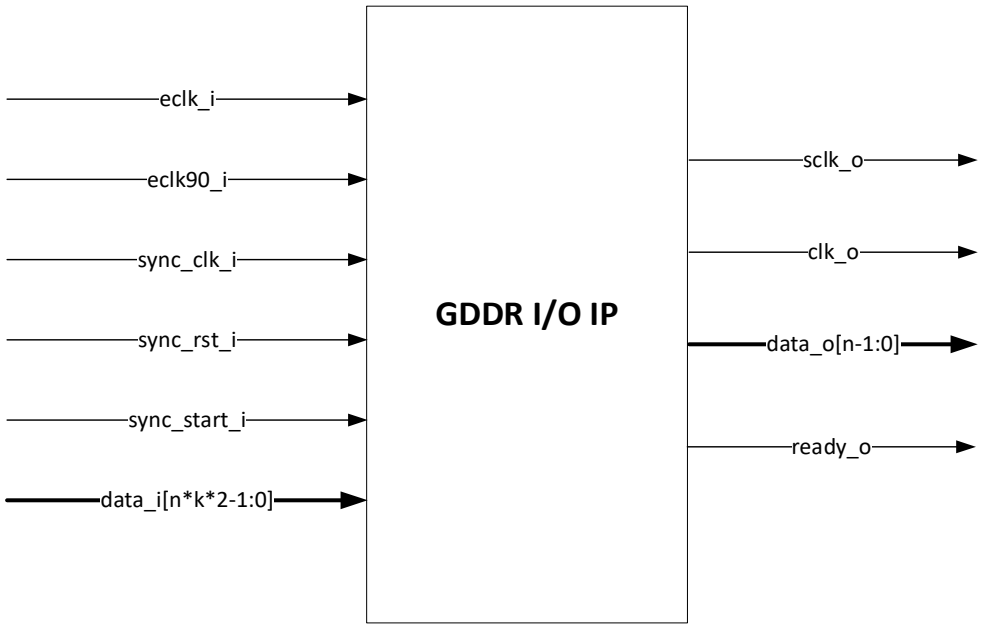


Figure 2.15. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay Block Diagram

Based on Figure 2.15, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value and a coarse delay value setting to override the delay of the data path.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from eclk90_i.
- The incoming clock, eclk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- The clock, eclk_i also serves as the clock used for capturing data of the DDR data components.
- GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components. The ready_o signal is asserted after the synchronization is done to indicate that the interface is ready to operate.

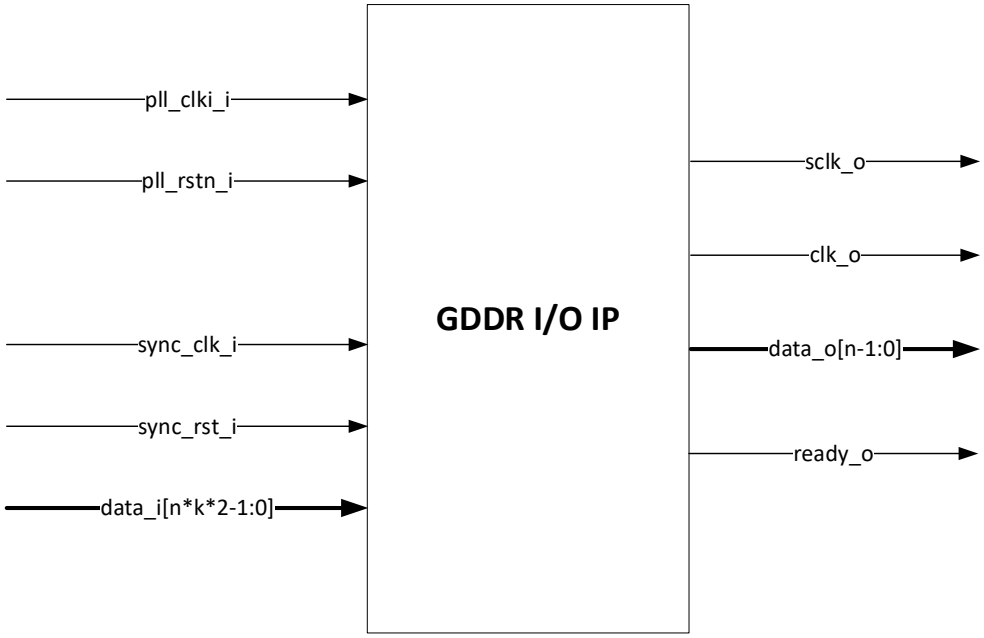


Figure 2.16. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay with PLL Block Diagram

Based on Figure 2.16, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value and a coarse delay value setting to override the delay of the data path.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from generated 90-degree phase shifted PLL clock output based on pll_clk_i.
- The incoming interface clock generated by PLL from pll_clk_i is divided into gearing setting on the clock divider component to generate the sclk_o.
- GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components. The ready_o signal is asserted after the synchronization is done to indicate that the interface is ready to operate.

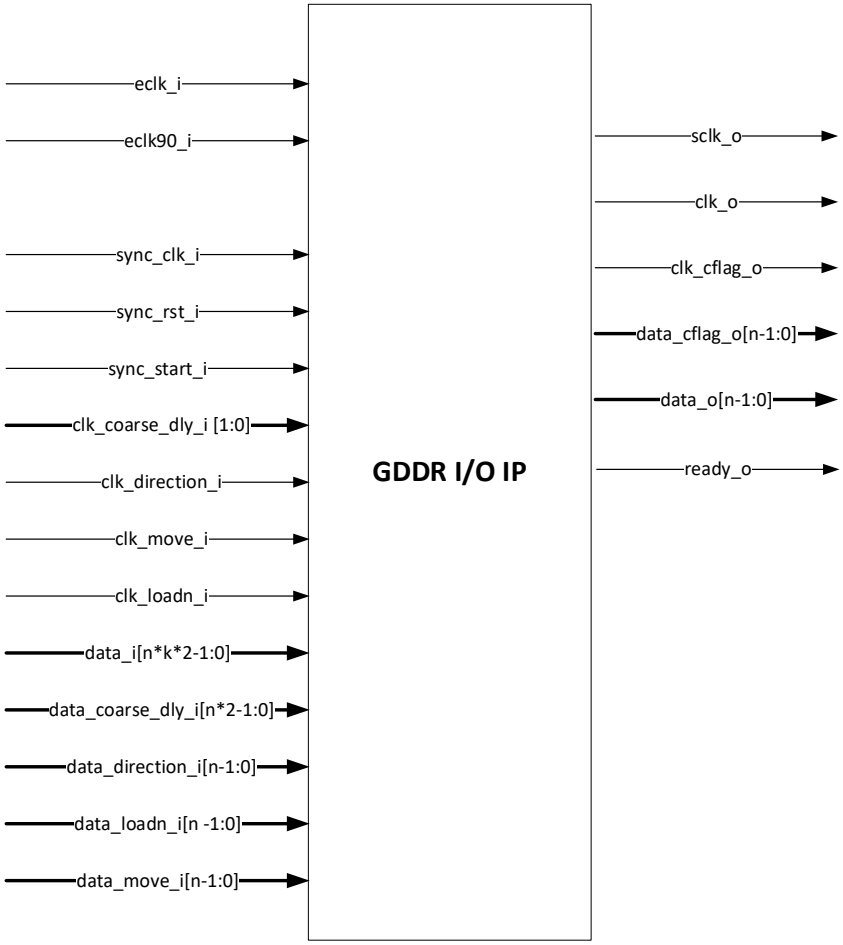


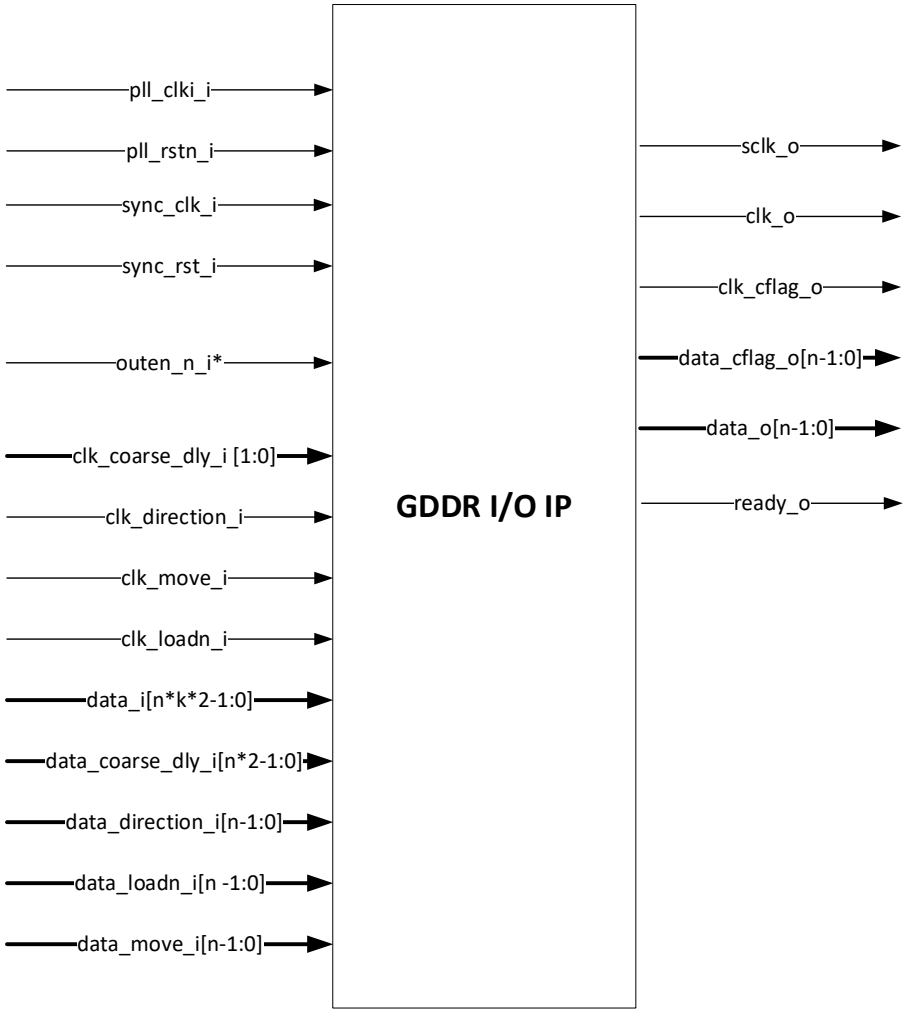
Figure 2.17. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram

The diagram presented in Figure 2.17 is true for both; Tristate Control enabled and disabled cases (with outen_n_i present in the Tristate Control enabled case).

The interface function is described below:

- On this interface, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component.
- Dynamic user-defined data path delay adds fine delay value on the data path delay. Data delay can also be dynamically controlled through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals.
- Clock delay can also be dynamically controlled through clk_coarse_dly_i, clk_direction_i, clk_loadn_i, and clk_move_i input signals.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from eclk90_i.
- The incoming clock, eclk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- The data and clock outputs, data_o and clk_o, can be optionally tri-stated using an I/O register.

GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components. The ready_o signal is asserted after the synchronization is done to indicate that the interface is ready to operate.



***Note:** Present only for Tristate Control Enabled case.

Figure 2.18. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay with PLL

Based on Figure 2.18, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component.
- Dynamic user-defined data path delay adds fine delay value on the data path delay. Data delay can also be dynamically controlled through data_coarse_dly_i, data_direction_i, data_loadn_i, and data_move_i input signals.
- Clock delay can also be dynamically controlled through clk_coarse_dly_i, clk_direction_i, clk_loadn_i, and clk_move_i input signals.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from generated 90-degree phase shifted PLL clock output based on pll_clk_i.
- The incoming interface clock generated by PLL from pll_clk_i is divided into gearing setting on the clock divider component to generate the sclk_o.
- The data and clock outputs, data_o and clk_o, can be optionally tri-stated using an I/O register.
- GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components. The ready_o signal is asserted after the synchronization is done to indicate that the interface is ready to operate.

2.2.8. GDDRX2/4/5_TX.ECLK.Aligned

These are generic transmit interfaces using X2, X4, or X5 gearing and Edge Clock Tree (ECLK). The input clock is edge aligned to the data. These interfaces must be used for DDR data rates above 400 Mb/s.

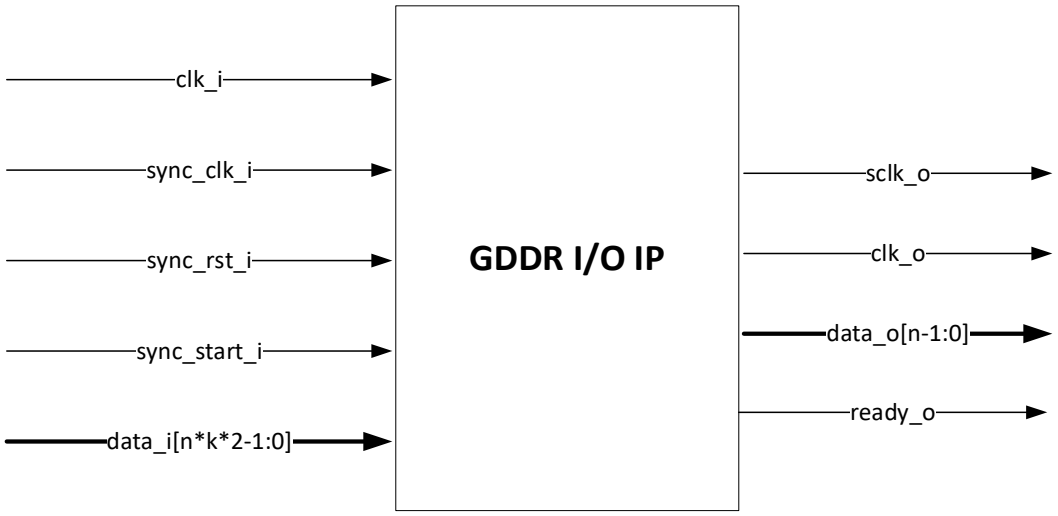


Figure 2.19. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay (with GDDR_SYNC) Block Diagram

Based on Figure 2.19, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value and a coarse delay value setting to override the delay of the data path.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk_i.
- The incoming clock, clk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components. The ready_o signal is asserted after the synchronization is done to indicate that the interface is ready to operate.

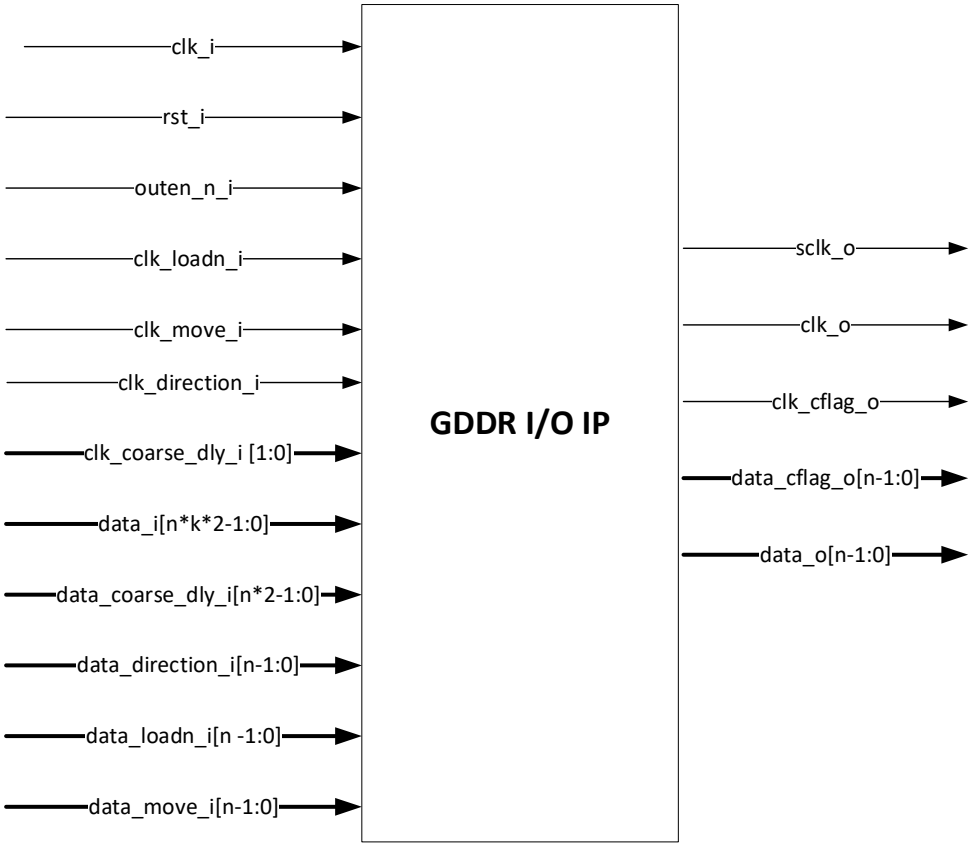


Figure 2.20. GDDRX2/4/5_TX.ECLK.Aligned Dynamic User-defined Delay Block Diagram

Based on Figure 2.20, the interface function is described below:

- On this interface, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component.
- Dynamic user-defined data path delay adds fine delay value and coarse delay value on the data path delay. Data delay can also be dynamically controlled through data_coarse_dly_i, data_direction_i, data_loadn_i and data_move_i input signals.
- Clock delay can also be dynamically controlled through clk_coarse_dly_i, clk_direction_i, clk_loadn_i, and clk_move_i input signals.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk_i.
- The incoming clock, clk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.

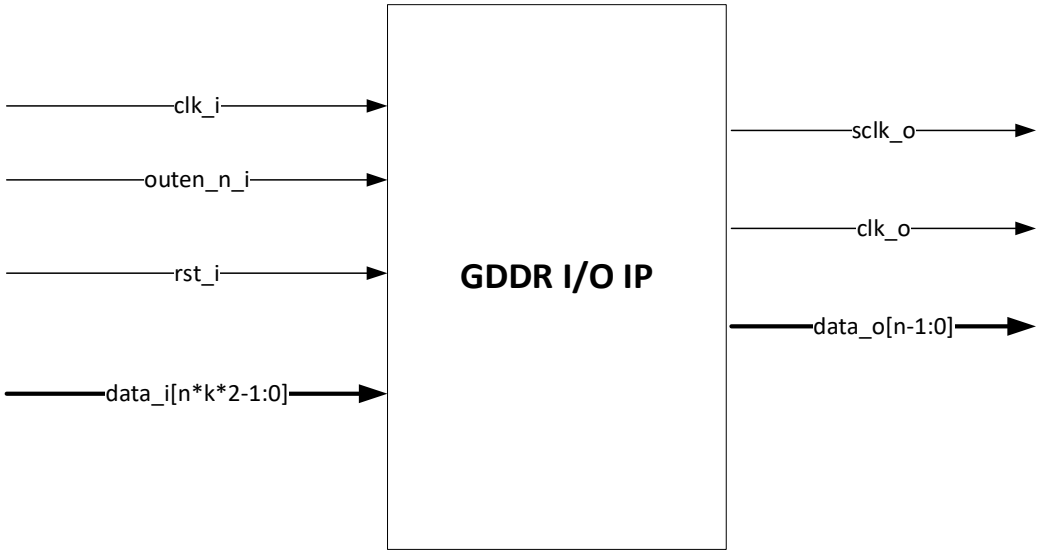


Figure 2.21. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay Tristate Control Enabled Block Diagram

Based on Figure 2.21, the interface function is described below:

- For Bypass interface, the input data, data_i, is being captured directly by DDR data component.
- For Static User-defined, the input data, data_i, is being captured initially by DDR data component and then processed by data delay component. This interface has a fine delay value setting to override the delay of the data path.
- Output sampling clock, clk_o, is internally routed through a DDR clock component from clk_i.
- The incoming clock, clk_i, is divided into gearing setting on the clock divider component to generate the sclk_o.
- The data and clock outputs, data_o and clk_o, can be optionally tri-stated using an I/O register.
- GDDR_SYNC is a startup synchronization Soft IP module used on this interface to handle the reset of the clock and DDR data and clock components.

2.3. GDDR Simulation Behavior

2.3.1. RX Output Data Mapping

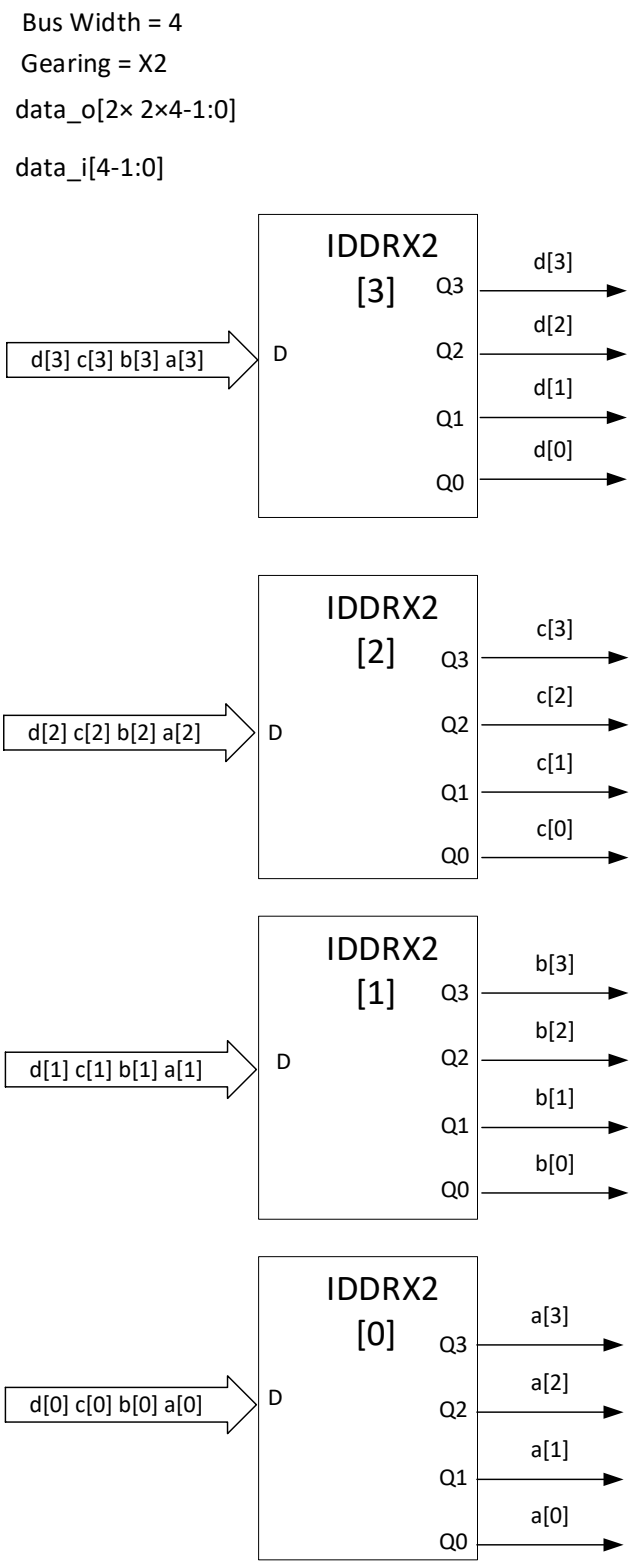


Figure 2.22. Rx Output Data Mapping Illustration

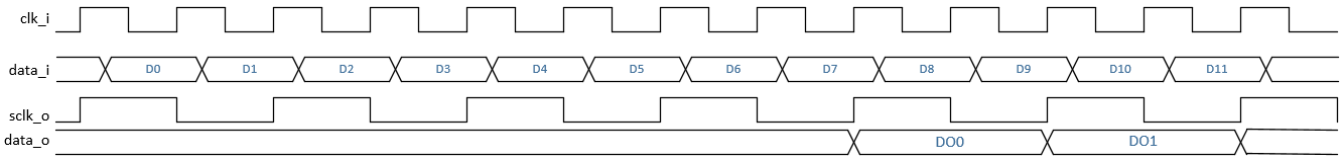


Figure 2.23. Rx (Centered) Timing Diagram

As seen in Figure 2.22, for GDDR Rx configuration, when Bus Width is 4 bits, the IP generates four IDDRX2 modules (in our example for X2 gearing, the number of data_i batches is four.) The number of data_i batches is dependent on the Bus Width of the configuration set by the user. Refer to FPGA-TN-02244 CertusPro-NX High-Speed I/O Interface for more details.

First batch of incoming data_i[a3:a0] is captured on the rising edge of the fast clock. The next batch of data_i[b3:b0] is captured on the falling edge of the fastest clock and so on. The fast clock for this case is eclk_i since it has the higher frequency.

In Figure 2.23, this translates to the batch of the input data's slowest bits data_i[d3:d0] being placed on the data_o vector's highest bits [15:12]. Similarly, data_i[c3:c0] bits are placed on data_o[11:8] and so on. The speed of the bit is determined by which output of the IDDRX2A module it came from - with those coming from Q3 as the slowest, followed by Q2, and so on. The concatenation of all the IDDRX2s outputs is a data_o[15:0] vector.

Figure 2.23 shows the timing diagram for such example.

2.3.2. TX Input Data Mapping

Bus Width = 4

Gearing = X2

data_i [2× 2×4-1:0]

data_o [4-1:0]

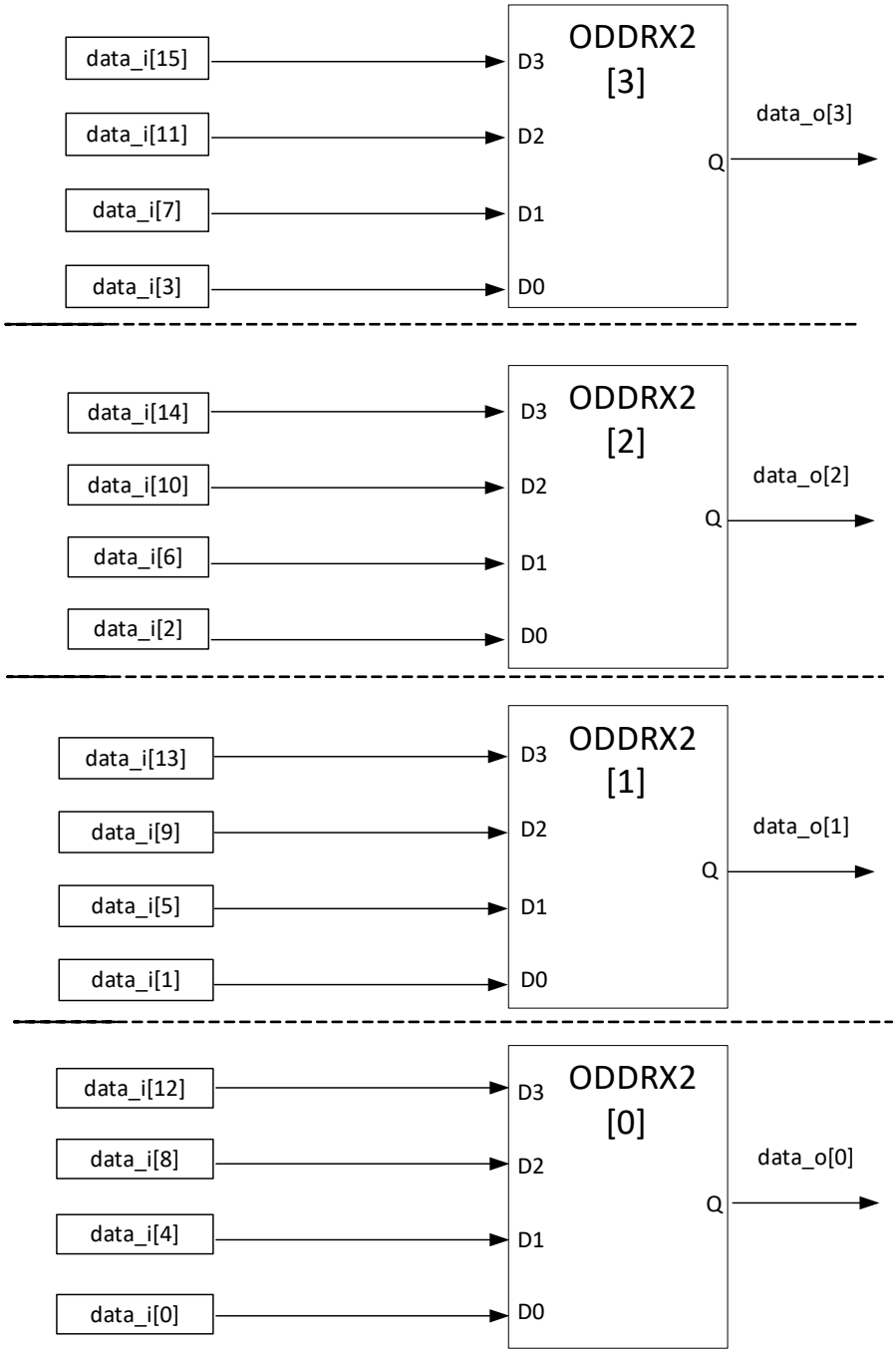


Figure 2.24. Tx Input Data Mapping Illustration

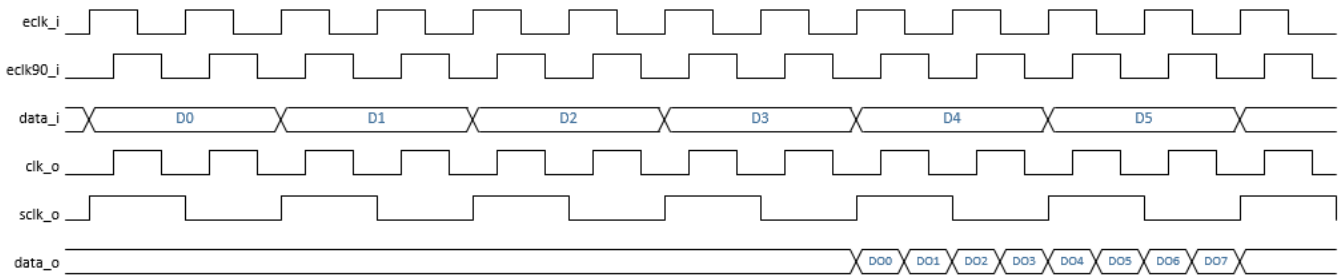


Figure 2.25. Tx (Centered) Timing Diagram

As seen in Figure 2.24, for GDDR Tx configuration, when Bus Width is 4 bits, the IP generates four ODDRX2 modules. The entire input data is being broken down into four groups, each group the size of (gearing*2) bits. The group of the fastest bits [12, 8, 4, 0] is being transmitted first. and so on, as shown in the picture above. Figure 2.25 shows the timing diagram for Figure 2.24. Refer to FPGA-TN-02244 CertusPro-NX High-Speed I/O Interface for more details.

2.3.3. RTL versus Post PAR Simulation Result

RTL simulation always uses the value/setting based on user interface setting. However, some of these settings will not produce the same expected result from RTL simulation due to unavoidable difference behavior between simulation model and silicon. Therefore, different setting derived from silicon characterization data is applied to primitives used such as DLLDEL, DELAYA and DELAYB during the Radiant implementation flow to ensure that expected result can be correctly produced in silicon. This might cause the post-routed netlist not being able to produce the expected output on simulation with the testbench that comes with the IP. This applies to the following GDDR configurations:

- On DLLDEL used on:
  - RX X1/X2/X4/X5 Aligned Bypass/Static Default/Dynamic Default/Static User Defined/Dynamic User Defined
- On DELAYB used on:
  - RX X1/X2/X4/X5 Aligned Bypass
  - RX X1 Aligned Dynamic Clock Bypass Data
  - RX X2/X4/X5 Aligned/Centered Static Default
  - RX X1 Centered Bypass
  - RX X2/X4/X5 Centered Bypass with or without GDDR_SYNC
- On DELAYA used on:
  - RX X2/X4/X5 Aligned/Centered Dynamic Default

The generated post PAR netlist added a default delay on the data input path, but clock path delay remains 0. To compensate for this, an artificial clock delay element is added to the IP testbench to model the clock injection delay on the chip. These delay parameters on the IP testbench, which are the *ENABLE_POST_PAR_DLY*, *COARSE_DLY* and *FINE_DLY_STEP*, need to be set to the right value that corresponds to the IP configuration. Set *ENABLE_POST_PAR_DLY* to 1 or 0 to enable or disable adding of the post PAR netlist default delay value to testbench respectively. Table 2.3 summarizes the setting for *COARSE_DLY* and *FINE_DLY_STEP* settings.

Table 2.3. Value Setting for Testbench Parameter during Post PAR Simulation

Family/Device	Configuration	*COARSE_DLY*	*FINE_DLY_STEP*
LFCPNX	*GEARING == X1* and *DATA_PATH_DELAY == Bypass*	“1P6NS”	“126”
LIFCL, LFD2NX	*GEARING == X1*, *DATA_PATH_DELAY == Bypass* and *CLOCK_DATA_RELATION == Edge-to-Edge*	“1P6NS”	“31”
LIFCL, LFD2NX	*GEARING == X1*, *DATA_PATH_DELAY == Bypass* and *CLOCK_DATA_RELATION == Centered*	“1P6NS”	“33”
LFMXO5-25	*GEARING == X1, DATA_PATH_DELAY == Bypass and CLOCK_DATA_RELATION == Edge-to-Edge*	"0P8NS"	"40"
LFMXO5-25	*GEARING == X1, DATA_PATH_DELAY == Bypass and CLOCK_DATA_RELATION == Centered*	"0P8NS"	“39”

Note that no input data delay nor clock injection delay is accounted for in RTL and post synthesis simulation.

2.3.4. DLLDEL on Dynamic Clock Delay Configuration

DLLDEL offers 512 steps each with 12.5 ps delay. Initially, DLLDEL receives code from DDRDLL which is used to create a 90-degree phase shifted clock. During dynamic clock delay, DLLDEL.MOVE can be asserted several times to increase the clock delay. DLLDEL.COUT asserts when 512 DLLDEL.MOVE assertion is applied while incrementing delay. However, overflow indicator does not include DDRDLL.CODE. To use the DLLDEL dynamic clock delay configuration efficiently, include the DDRDLL.CODE value on the maximum 512 steps.

Example calculation of maximum DLLDEL.MOVE assertion:

$$Input\ Clock\ Frequency\ =\ 100\ MHz$$

$$DDRDLL.CODE\ =\ (((1/100MHz)/4)/12.5ps)\ =\ 200$$

$$Maximum\ DLLDEL.MOVE\ assertion\ =\ 511 - 200\ =\ 311$$

2.4. Signal Description

Table 2.4. GDDR I/O Module Receive Signal Description

Port Name	Direction	Width (Bits)	Description
Clock and Reset			
clk_o	OUT	1	Received data sampling clock
rst_i	IN	1	Active HIGH reset signal
sclk_o	OUT	1	Clock output for receive interface
sync_clk_i	IN	1	RX_SYNC/GDDR_SYNC Startup clock. A low speed continuously running clock input. Its frequency is independent to the input clock, but it must be significantly lower.
sync_rst_i	IN	1	RX_SYNC/GDDR_SYNC active HIGH reset signal. Only utilized by the RX_SYNC/GDDR_SYNC and can be asserted asynchronously but it is recommended to be de-asserted synchronously with sync_clk_i.
User Interface			
data_o	OUT	n * k	Received input data to fabric
alignwd_i	IN	1	This signal is used for word alignment. It shifts word by one bit. Only available for X2, X4, and X5 gearing ratio.
data_loadn_i	IN	n	Active LOW signal to reset data path delay setting to default. Only available during dynamic user-defined data path delay.
data_move_i	IN	n	Increments or decrements data path delay setting depending on data_direction_i. Only available during dynamic user-defined data path delay.
data_coarse_dly_i	IN	2n	Coarse delay value for data path set by user. Only available during dynamic user-defined data path delay.
data_direction_i	IN	n	*1* to decrease data path delay; *0* to increase data path delay. Only available during dynamic user-defined data path delay.
data_cflag_o	OUT	n	Underflow or overflow flag to indicate minimum or maximum data path delay adjustment is reached. Only available during dynamic user-defined data path delay.
clk_loadn_i	IN	1	Active LOW signal to reset clock delay setting for clock to default. Only available during dynamic clock delay enabled.
clk_move_i	IN	1	Increments or decrements clock delay setting depending on clk_direction_i. Only available during dynamic clock delay enabled.
clk_direction_i	IN	1	*1* to decrease clock delay; *0* to increase clock delay. Only available during dynamic clock delay enabled.
clk_coarse_dly_i	IN	2	Coarse delay value for clock set by user. Only available during dynamic clock delay enabled.
clk_cflag_o	OUT	1	Underflow or overflow flag to indicate minimum or maximum clock delay adjustment is reached. Only available during dynamic clock delay enabled.
sync_update_i	IN	1	Used to restart sync process. READY goes LOW and waits for the sync process to finish before going high again. Must not be asserted while data transfer is active.
sync_start_i	IN	1	Start the sync process. Must be HIGH during all synchronization process.
ready_o	OUT	1	Indicate that startup is finished, and RX circuit is ready to operate.
I/O Pad Interface			
clk_i	IN	1	Clock input signal from I/O
data_i	IN	n	Data input signal from I/O. Bus width value range is [1, 256].

Note: n = number of lanes, k = 2(X1), 4(X2), 8(X4), and 10(X5).

Table 2.5. GDDR I/O Module Transmit Signal Description

Port Name	Direction	Width (Bits)	Description
Clock and Reset			
rst_i	IN	1	Active HIGH reset signal
clk_i	IN	1	Transmit data sampling clock. Only available when PLL instantiation is disabled. For centered X1 or aligned X1, X2, X4, X5 configuration.
clk90_i	IN	1	90-degree shifted for transmit clock generation. For centered X1 configuration only.
sclk_o	OUT	1	Clock output for transmit interface. For X2, X4, and X5 configurations only.
eclk_i	IN	1	Transmit data sampling clock. Only available when PLL instantiation is disabled. For centered X2, X4, and X5 configurations.
eclk90_i	IN	1	90-degree shifted for transmit clock generation. For centered X2, X4 and X5 configurations.
sync_clk_i	IN	1	GDDR_SYNC Startup clock. A low speed continuously running clock input. Its frequency is independent to the input clock but it must be significantly lower.
sync_rst_i	IN	1	GDDR_SYNC reset signal which can be asserted asynchronously to the input clock, but it is recommended to be de-asserted synchronously with sync_clk_i.
pll_clki_i	IN	1	PLL reference clock.
pll_rstn_i	IN	1	PLL reset signal.
User Interface			
data_i	IN	n × k	Transmit output data going to I/O
outen_n_i	IN	1	Active LOW signal output enables
data_loadn_i	IN	n	Active LOW signal to reset data path delay setting to default. Only available during dynamic user-defined data path delay.
data_move_i	IN	n	Increments or decrements data path delay setting depending on data_direction_i. Only available during dynamic user-defined data path delay.
data_coarse_dly_i	IN	2n	Coarse delay value for data path set by user. Only available during dynamic user-defined data path delay.
data_direction_i	IN	n	*1* to decrease data path delay; *0* to increase data path delay. Only available during dynamic user-defined data path delay.
data_cflag_o	OUT	n	Underflow or overflow flag to indicate minimum or maximum data path delay adjustment is reached. Only available during dynamic user-defined data path delay.
clk_loadn_i	IN	1	Active LOW signal to reset clock delay setting for clock to default. Only available during dynamic clock delay enabled.
clk_move_i	IN	1	Increments or decrements clock delay setting depending on clk_direction_i. Only available during dynamic clock delay enabled.
clk_direction_i	IN	1	*1* to decrease clock delay; *0* to increase clock delay. Only available during dynamic clock delay enabled.
clk_coarse_dly_i	IN	2	Coarse delay value for clock set by user. Only available during dynamic clock delay enabled.
lock_o	OUT	1	PLL lock output signal. Available when PLL instantiation is enabled.
ready_o	OUT	1	Indicate that startup is finished, and TX circuit is ready to operate.
sync_start_i	IN	1	Used to re-start sync process. READY goes low and wait for sync process to finish before going high again.
I/O Pad Interface			
clk_o	OUT	1	Clock output signal to I/O
data_o	OUT	n	Data output signal to I/O. Value range is [1, 256]

Note: n = number of lanes, k = 2(X1), 4(X2), 8(X4), and 10(X5).

2.5. Attribute Summary

Table 2.6 provides a list of user configurable attributes for the GDDR I/O Module. Attribute settings are specified using GDDR I/O Module Configuration user interface in Lattice Radiant.

Table 2.6. Attributes Table

Attribute	Selectable Values	Default	Dependency on Other Attributes
Interface Type	Transmit, Receive	Receive	—
I/O Standard for this Interface	(Legal Combination Table)	LVDS	—
Gearing Ratio	X1, X2, X4, X5	X1	When *Interface Type ==Receive*, X1 is available if *Clock Frequency <=150 MHz* and family is LFCPNX or *Clock Frequency <=250 MHz* and family is not LFCPNX When Interface *Type == Transmit*, X1 is available if *Clock Frequency <=250 MHz* X2 is available if *Clock Frequency <= 500 MHz* X4 and X5 are available if *Clock Frequency <= 750 MHz*
Bus Width for this Interface	1 - 256	8	—
Clock to Data Relations on the Pins	Edge-to-Edge, Centered	Edge-to-Edge	—
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 GDDRX5_RX.ECLK.Aligned GDDRX5_RX.ECLK.Centered GDDRX5_TX.ECLK.Aligned GDDRX5_TX.ECLK.Centered	GDDRX1_RX.SCLK.Aligned	For display information only.
Data Path Delay	Bypass, Static Default, Dynamic Default, Static User-defined, Dynamic User-defined	Bypass	*Static Default* and *Dynamic Default* are supported for *Interface Type == Receive*
Fine Delay Value for User-defined	0-126	0	*Data Path Delay == Static User-defined* or *Dynamic User-defined*
Coarse Delay Value for User-defined	0NS 0P8NS 1P6NS	0NS	*Data Path Delay == Static User-defined* or *Dynamic User-defined*
Clock Path Delay	Fixed Dynamic	Fixed	Should be **Dynamic** if: *Data Path Delay == Dynamic Default* or *Data Path Delay == Dynamic User-defined*
Enable Tristate Control	Checked, Not checked	Not checked	Optional if: *Interface Type == Transmit*, *Clock to Data Relationship on the Pins == Centered*, *Gearing Ratio* != *X1* and *Data Path Delay == Dynamic*

Attribute	Selectable Values	Default	Dependency on Other Attributes
			User-defined and *Clock Path Delay == Dynamic* Automatically Checked if: *Interface Type == Transmit*, *Clock to Data Relationship on the Pins == Edge-to-edge*, *Gearing Ratio != X1* and *Data Path Delay != Dynamic Default nor Dynamic User-defined* and *Clock Path Delay != Dynamic*
Clock Frequency for this Interface (MHz)	100-750	150	Selectable maximum value is based on Gearing setting.
Bandwidth for this Interface (Mbits/s)	Calculated	2400	2*(Clock Frequency for this Interface) *(Bus Width for this Interface)
Enable PLL Instantiation	Checked, Not checked	Not checked	Optional if: *Interface Type == Transmit,'Clock to Data Relationship on the Pins == Centered, Gearing Ratio* != *X1*
PLL Input Clock Frequency (MHz)	10 – Max Value	25	Enable PLL Instantiation is checked. Max Value = (Clock Frequency for this Interface)
PLL Output Clock Frequency Actual Value (MHz)	Calculated	150	—
PLL Reference Clock from I/O Pin	Checked, Not checked	Not checked	Enable PLL Instantiation is checked.
I/O Standard for Reference Clock	(Legal Combination Table)	LVDS	PLL Reference Clock from I/O Pin is checked.
PLL Output Clock Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	Enable PLL Instantiation is checked.

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

Table 2.7 presents attribute description.

Table 2.7. Attribute Description

Attribute	Description
Interface Type	Selects interface type as Receive or Transmit.
I/O Standard for this Interface	List of Single-ended or Differential I/O supported.
Gearing Ratio	Selects gearing ratio as X1, X2, X4, or X5.
Bus Width for this Interface	Total number of lanes/bus width.
Clock to Data Relations on the Pins	Selects clock to data relationship as Edge-to-edge or Centered.
Interface	Summarizes the interface to be used.
Data Path Delay	Selects among Bypass, Static Default, Static User-defined, Dynamic Default or Dynamic User-defined. Allows you to set default data path coarse and fine delay values when selected as user-defined.
Fine Delay Value for User-defined	Default data path fine delay setting. Only valid when Data Path Delay is Static User-defined or Dynamic User-defined. 7-bit binary string, 12.5 ps per step max_fine_delay = 127 * 12.5 ps = 1.5875 ns
Coarse Delay Value for User Defined	Default data path coarse delay setting. Only valid when Data Path Delay is Static User Defined or Dynamic User Defined. For STATIC mode, value overrides the default delay setting. For DYNAMIC mode, this is the initial value of the coarse delay element. 2'b00 = *0NS* 2'b01 = *0P8NS* 2'b10 = *1P6NS* Computation: total_delay = coarse_delay + fine_delay coarse_delay_val == 2'b00: total_delay(min) = 0 ns, total_delay(max) = 1.5875 ns coarse_delay_val == 2'b01: total_delay(min) = 0.8 ns, total_delay(max) = 2.3875 ns coarse_delay_val == 2'b10: total_delay(min) = 1.6 ns, total_delay(max) = 3.1875 ns
Clock Path Delay	Fixed or dynamic addition of clock path delay.
Include GDDR_SYNC	Enables usage of GDDR_SYNC support soft logic.
Enable Tristate Control	Enable Tri-State Control instantiation.
Clock Frequency for this Interface (MHz)	Clock frequency to be used in selected interface.
Enable PLL Instantiation	Enables usage of existing PLL module.
PLL Input Clock Frequency (MHz)	Value of PLL input clock frequency
PLL Output Clock Frequency Actual Value (MHz)	Value of PLL output clock frequency
PLL Reference Clock from I/O Pin	Enables the use of I/O supported
I/O Standard for Reference Clock	List of Single-ended or Differential I/O supported.
PLL Output Clock Tolerance (%)	Percentage on allowable difference of the actual to desired value.

3. IP Generation, Simulation, and Validation

This section provides information on how to generate the IP using the Lattice Radiant software, and how to run simulation, synthesis, and hardware evaluation. For more details on the Lattice Radiant software, refer to the Lattice Radiant software user guide.

3.1. Generating the IP

The Lattice Radiant software allows you to customize and generate modules and IPs and integrate them into the device's architecture. The procedure for generating the GDDR I/O Module in Lattice Radiant software is described below.

To generate GDDR I/O Module:

1. Create a new Lattice Radiant Software project or open an existing project.
2. In the **IP Catalog** tab, double-click on **DDR_Generic** under **Module, Architecture_Modules, IO** category. The **Module/IP Block Wizard** opens as shown in Figure 3.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

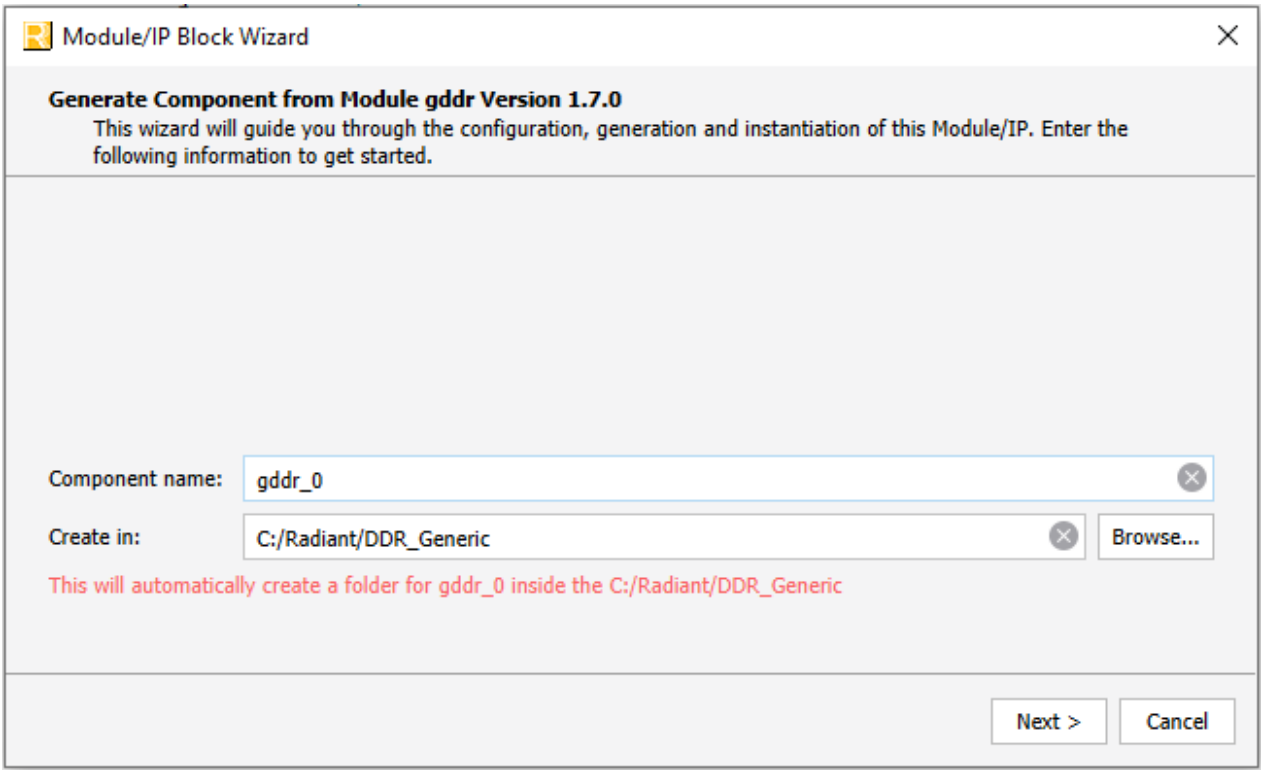


Figure 3.1. Module/IP Block Wizard

3. In the module's dialog box of the **Module/IP Block Wizard** window, customize the selected GDDR I/O Module using drop-down menus and check boxes. As a sample configuration, see Figure 3.2. For configuration options, see the Attribute Summary section.

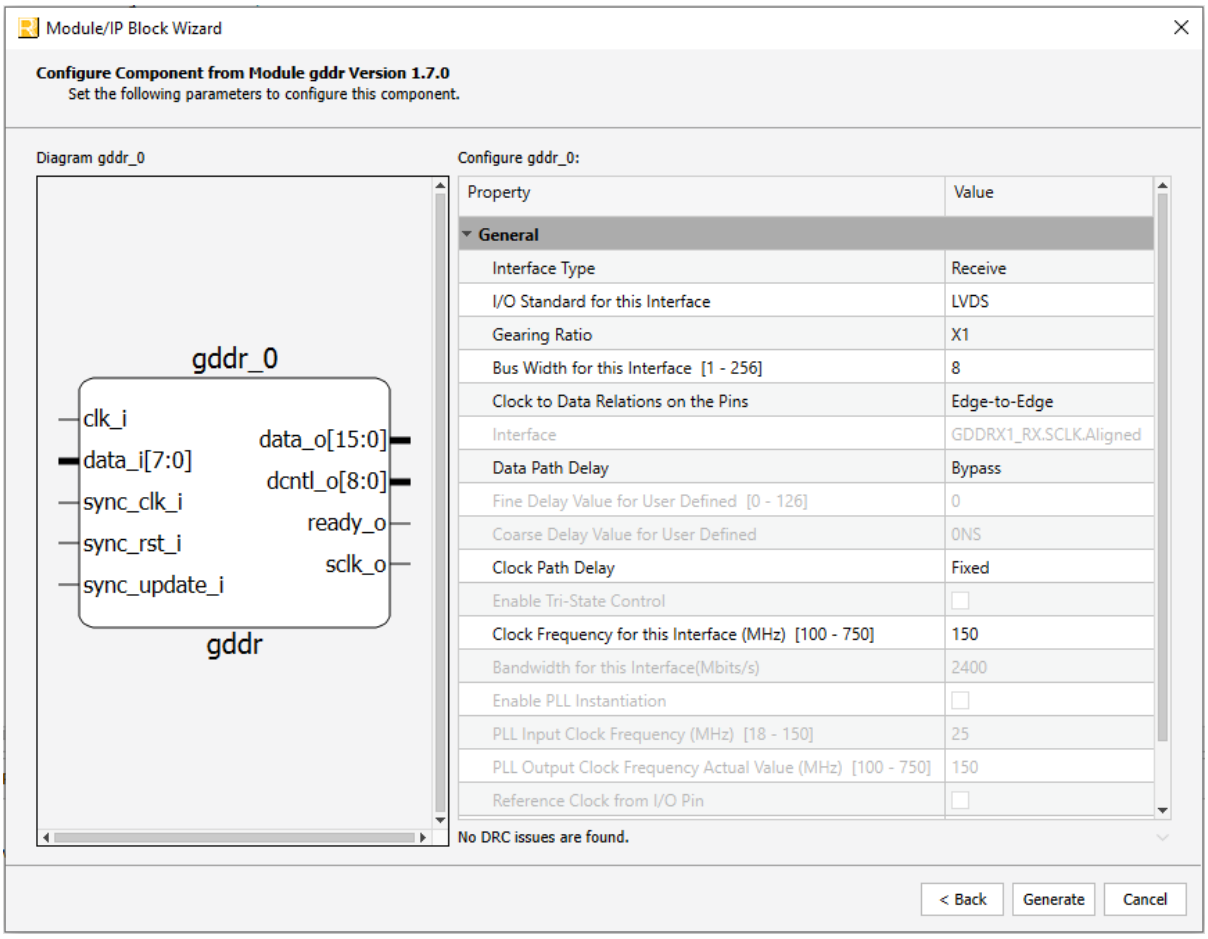


Figure 3.2. Configure Block of GDDR I/O Module

4. Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 3.3.

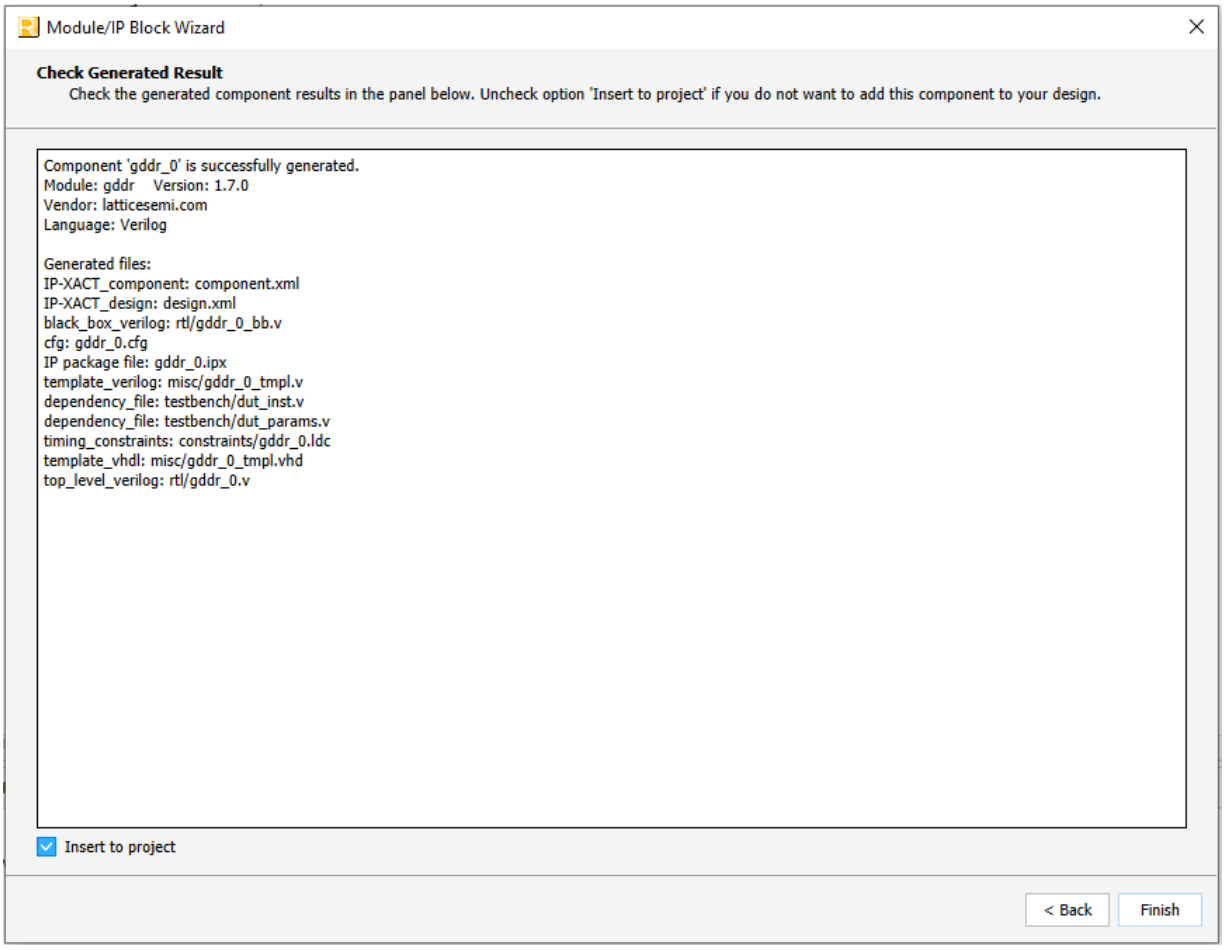


Figure 3.3. Check Generated Result

5. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 3.1.

The generated GDDR I/O module package includes the closed-box (<Instance Name>_bb.v) and instance templates (<Instance Name>_tmpl.v/vhd) that can be used to instantiate the module in a top-level design. An example RTL top-level reference source file (<Instance 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 3.1.

Table 3.1. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file 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/<Instance Name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Instance Name>_bb.v	This file provides the synthesis closed-box.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the module.

3.2. Running Functional Simulation

After the IP is generated, running functional simulation can be performed using different available simulators. The default simulator already has pre-compiled libraries ready for simulation. Choosing a non-default simulator, however, may require additional steps.

To run functional simulation using the default simulator:

1. Click the  button located on the **Toolbar** to initiate **Simulation Wizard**, as shown in Figure 3.4.

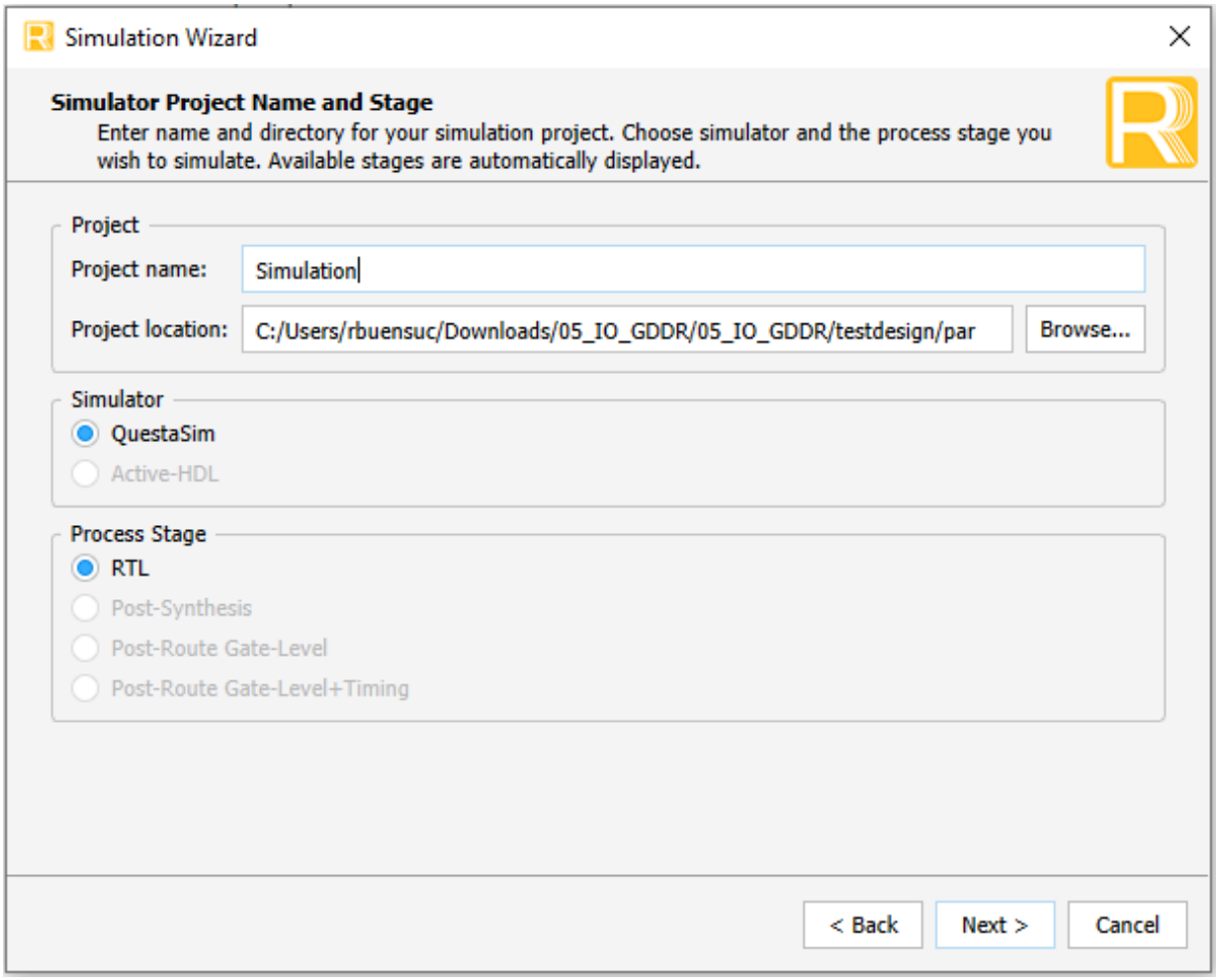


Figure 3.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.5.

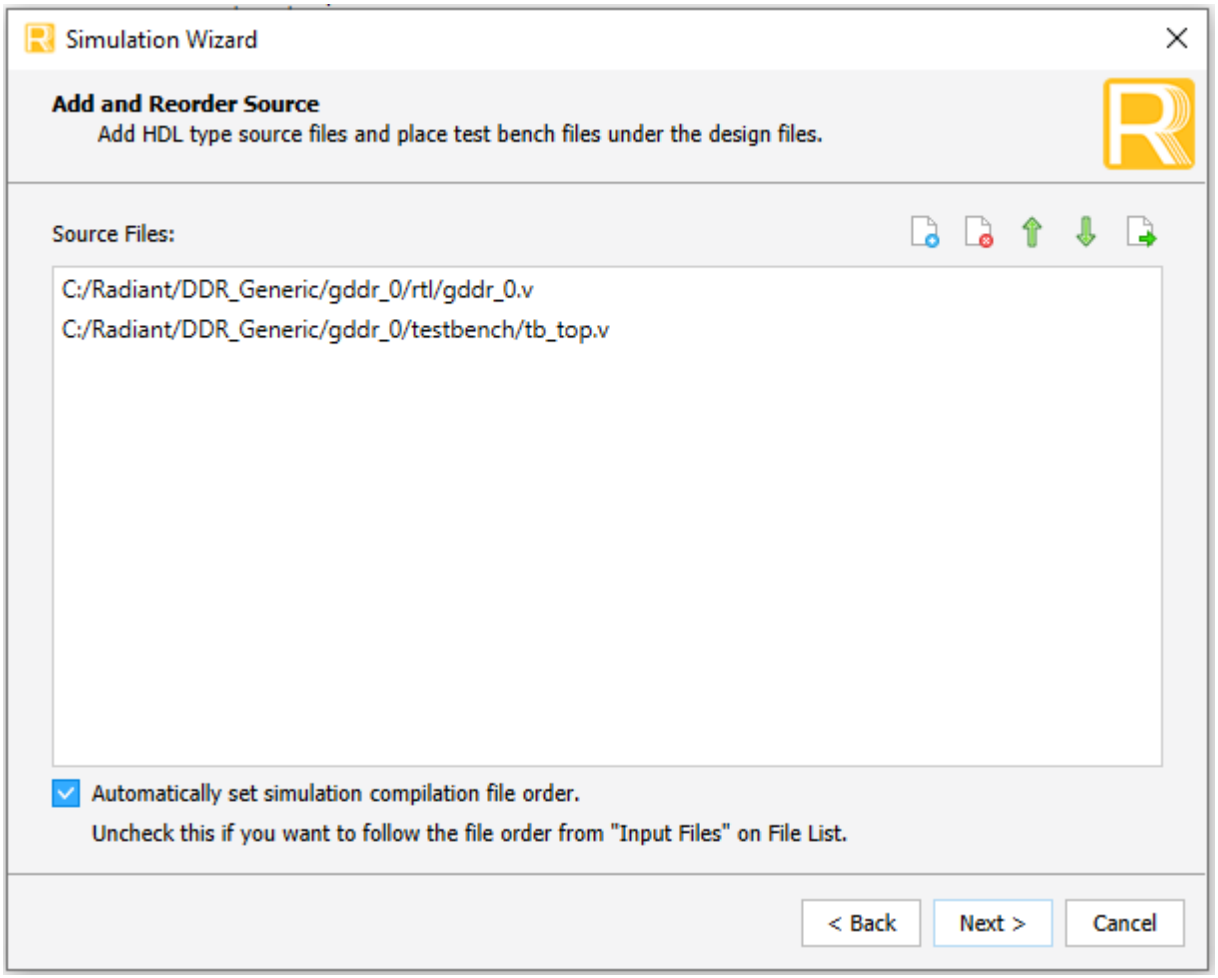


Figure 3.5. Adding and Reordering Source

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

Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant Software Suite.

The results of the simulation in our example are provided in Figure 3.6.

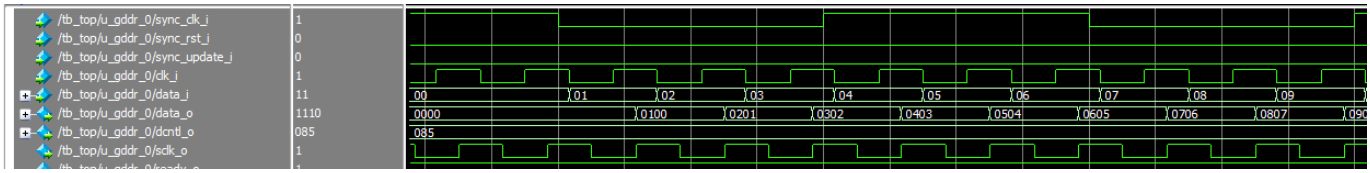


Figure 3.6. Simulation Waveform

3.3. IP Evaluation

There is no restriction on the IP evaluation of this module.

Appendix A. Resource Utilization

Table A.1 and Table A.2 show resource utilization of DDR Generic configurations for LIFCL-40-9BG400C and LFCPNX-100-9LFG672C using Lattice Synthesis Engine of Lattice Radiant software 3.1. Default configuration is used, and some attributes are changed from the default values to show the effect on the resource utilization.

Table A.1. Resource Utilization (LIFCL)

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
8-bit Receive X1 Edge-to-Edge Bypass	200	32	101	0	0
8-bit Receive X2 Centered Bypass	200	0	0	0	0
16-Receive X4 Edge-to-Edge Static Default	200	32	94	0	0
4-bit Receive X5 Centered Dynamic Default	200	0	1	0	0
8-bit Transmit X1 Edge-to-Edge Bypass	200	0	1	0	0
16-bit Transmit X2 Centered Bypass	200	12	33	0	0
16-bit Transmit X4 Centered Static User Defined	200	12	33	0	0
4-bit Transmit X5 Centered Dynamic User Defined	200	12	33	0	0

Notes:

1. Fmax is generated when the FPGA design only contains Generic DDR module, and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among *logic*, *distributed RAM*, and *ripple logic*.

Table A.2. Resource Utilization (LFCPNX)

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
8-bit Receive X1 Edge-to-Edge Bypass	150	32	101	0	0
8-bit Receive X2 Centered Bypass	250	0	0	0	0
16-Receive X4 Edge-to-Edge Static Default	250	32	94	0	0
4-bit Receive X5 Centered Dynamic Default	250	0	1	0	0
8-bit Transmit X1 Edge-to-Edge Bypass	250	0	1	0	0
16-bit Transmit X2 Centered Bypass	250	12	33	0	0
16-bit Transmit X4 Centered Static User Defined	250	12	33	0	0
4-bit Transmit X5 Centered Dynamic User Defined	250	12	33	0	0

Notes:

1. Fmax is generated when the FPGA design only contains the Generic DDR module, and the target frequency is 250 MHz. FPGA can run up to 250 MHz but Generic DDR module in Receive X1 mode can be used up to 150 MHz only. These values may be reduced when user logic is added to the FPGA design.
2. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among *logic*, *distributed RAM*, and *ripple logic*.

References

- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- iCE40 UltraPlus web page
- MachXO5-NX web page
- Lattice Radiant Software web page
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Insights web page 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, refer to the Lattice Answer Database at https://www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.9, June 2024

Section	Change Summary
All	• Updated document title from *DDR Generic Module - Lattice Radiant Software* to *DDR Generic Module*. • Made editorial fixes.
Disclaimers	Updated this section.
Inclusive Language	Added boilerplate.

Introduction	• Updated the Features section:
	  • Added device configuration for *GDDR_SYNC soft IP logic*.
	  • Added *RX_SYNC soft IP logic* with device configuration.

Functional Description	• Updated the Overview section:
	  • Added information of Synchronization Modules in Table 2.2. Summary of GDDR Support Soft Logic.
	• Updated the Functional Diagrams section:
	  • Removed *Figure 2.7 GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram*.
	  • Added interface function descriptions for the following figures:
	  • *Figure 2.2. GDDRX1_RX.SCLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram* – *Figure 2.21. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay Tristate Control Enabled Block Diagram.*
	  • Renamed *Figure 2.5 GDDRX1_RX.SCLK.Aligned Dynamic Default/Dynamic User-defined Delay (Clock) Block Diagram* to Figure 2.5. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User Defined Delay with Dynamic (Clock) Delay Block Diagram.
	  • Renamed *Figure 2.9. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC Block Diagram* to Figure 2.7. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram.
	  • Updated the Figure 2.8. GDDRX2/4/5_RX.ECLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram.
	  • Added information of *Centered clock generation* for sections GDDRX1_TX.SCLK.Centered and GDDRX2/4/5_TX.ECLK.Centered.
	  • Removed the note sections for Figure 2.11. GDDRX1_TX.SCLK.Centered Bypass/Static User-defined Delay Block Diagram, Figure 2.12. GDDRX1_TX.SCLK.Centered Dynamic User-defined Delay Block Diagram, Figure 2.15. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay Block Diagram, and Figure 2.17. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram.
	• Updated the GDDR Simulation Behavior sections:
	  • Updated description for Figure 2.23. Rx (Centered) Timing Diagram and Figure 2.24. Tx Input Data Mapping Illustration.
	  • Added caption for Figure 2.25. Tx (Centered) Timing Diagram.
	• Updated the description of the following ports in Table 2.4. GDDR I/O Module Receive Signal Description:
	  • *sync_clk_i*
	  • *sync_rst_i*
	  • *sync_update_i*
	• Updated the description of the following ports in Table 2.5. GDDR I/O Module Transmit Signal Description:
	  • *clk_i*
	  • *clk90_i*
	  • *sync_clk_i*

Section	Change Summary

	• *sync_rst_i*
	• Added new ports in Table 2.5. GDR I/O Module Transmit Signal Description:
	  • *pll_clki_i*
	  • *pll_rstn_i*
	• Updated the description of the following ports in Table 2.6. Attributes Table:
	  • *Clock Path Delay.*
	  • *Enable Tristate Control.*
	  • *Enable PLL Instantiation.*
	  • Removed *Include GDDR_SYNC.*

IP Generation, Simulation and Validation	• Updated the following figures in the Generating the IP section:
	  • Figure 3.1. Module/IP Block Wizard.
	  • Figure 3.2. Configure Block of GDDR I/O Module.
	  • Figure 3.3. Check Generated Result.
	• Updated the following figures in the Running Functional Simulation section:
	  • Figure 3.4. Simulation Wizard.
	  • Figure 3.5. Adding and Reordering Source.

Reference	Updated this section.
Technical Support Assistance	Updated this section.

Revision 1.8, November 2022

Section	Change Summary
Functional Description	• Added LFMXO5-25 Family/Device to Table 2.3.
IP Generation, Simulation, and Validation	• Updated the heading of Section 3 from “IP Generation and Evaluation” to “IP Generation, Simulation, and Validation”. • Removed “Licensing the IP” section. • Updated the heading of Section 3.1 from “Generation and Synthesis” to “Generating the IP”. • Updated Figure 3.1. Module/IP Block Wizard, Figure 3.2. Configure Block of GDDR I/O Module, Figure 3.3. Check Generated Result, Figure 3.4. Simulation Wizard, and Figure 3.6. Simulation Waveform. • Updated the heading of Section 3.3 from “Hardware Evaluation” to “IP Evaluation”.

Revision 1.7, November 2021

Section	Change Summary
Functional Description	• In RTL versus Post PAR Simulation Result, added Table 2.3. Value Setting for Testbench Parameter during Post PAR Simulation. • Updated Table 2.6. Attributes Table. Revised Gearing Ratio - Dependency on Other Attributes
Appendix A. Resource Utilization	Added Table A.1. Resource Utilization (LIFCL) and updated Table A.2. Resource Utilization (LFCPNX).

Revision 1.6, June 2021

Section	Change Summary
Introduction	Updated content including removing Quick Facts section.
Appendix A. Resource Utilization	Updated Table A.1.
References	Updated section content.

Revision 1.5, March 2021

Section	Change Summary
Functional Description	Added DLLDEL on Dynamic Clock Delay Configuration section.

Revision 1.4, December 2020

Section	Change Summary
Functional Description	Added *Interface* attribute in Attribute Summary.
IP Generation and Evaluation	Updated figures.
Appendix A. Resource Utilization	Added this section.
References	Updated this section.

Revision 1.3, October 2020

Section	Change Summary
Functional Description	In RTL versus Post PAR Simulation Result: • Corrected part of statement to "… expected output *on* simulation…". • Added paragraph.
IP Generation and Evaluation	Updated introductory paragraph in Running Functional Simulation sub-section.

Revision 1.2, June 2020

Section	Change Summary
Introduction	• Added Certus-NX support. • Updated Table 1.1 to add LFD2NX-40 as targeted device. • Updated Lattice Implementation to Lattice Radiant 2.1. • Updated Synopsis Synplify Pro version.
Functional Description	• Moved contents of the RX Output Data Mapping and the TX Input Data Mapping sections. • Added GDDR Simulation Behavior sub-section.
All	Updated references to Lattice Radiant Software 2.1 User Guide.

Revision 1.1, February 2020

Section	Change Summary
Acronyms in this Document	Updated the table.
Introduction	• Updated Table 1.1 to add LIFCL-17 as targeted device. • Added subsection for 100 MHz to 750 MHz clock frequency in Features section.
Functional Description	Updated Table 2.5.
IP Generation and Evaluation	Updated content of the Generation and Synthesis section.
Appendix	Removed this section.

Revision 1.0, December 2019

Section	Change Summary
All	Changed document status from Preliminary to final.
Quick Facts	Updated the table details.
Attribute Summary	Edited Fine Delay Value for user-defined selectable value range.
IP Generation and Evaluation	• Updated details and illustrations. • Removed Licensing and Evaluation and merged contents with this section.
All	Updated table and figure captions.
Appendix	Added this section.

Revision 0.80, October 2019

Section	Change Summary
All	Preliminary release.

www.latticesemi.com