Avant DDR Generic Module

IP Version: v2.4.0

User Guide

FPGA-IPUG-02188-1.3

April 2026

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

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	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 1800 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 900 MHz clock frequency
  - 100 MHz to 250 MHz for X1 Gearing
  - 100 MHz to 600 MHz for X2 Gearing
  - 100 MHz to 900 MHz for X4 and X5 Gearing
- Clock-data relationship options:
  - Edge-to-edge
  - Centered
- Data Path Delay that includes the 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 the performance. Table 2.1 provides a summary of GDDR I/O Interface. Refer to FPGA-TN-02300 Lattice Avant High-Speed I/O and External Memory Interface User Guide for more details on User Primitives and Attributes.

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. Requires RX_SYNC support soft logic, which uses DDRDLL and DLLDEL.
GDDRX2_RX.ECLK.Centered	Generic DDR 4:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. With optional support to GDDR_SYNC 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. Requires RX_SYNC support soft logic, which uses DDRDLL and DLLDEL.
GDDRX4_RX.ECLK.Centered	Generic DDR 8:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. With optional support to GDDR_SYNC 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. Requires RX_SYNC support soft logic, which uses DDRDLL and DLLDEL.
GDDRX5_RX.ECLK.Centered	Generic DDR 10:1 Receive Centered Interface	Supports bypassed, static, and dynamic data path delay. With optional support to GDDR_SYNC 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. Requires RX_SYNC support soft logic, which uses 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. With optional support to GDDR_SYNC soft logic.
GDDRX2_TX.ECLK.Aligned	Generic DDR 4:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. With optional support to GDDR_SYNC soft logic.
GDDRX4_TX.ECLK.Centered	Generic DDR 8:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. With optional support to GDDR_SYNC soft logic.
GDDRX4_TX.ECLK.Aligned	Generic DDR 8:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. With optional support to GDDR_SYNC soft logic.
GDDRX5_TX.ECLK.Centered	Generic DDR 10:1 Transmit Centered Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. With optional support to GDDR_SYNC soft logic.
GDDRX5_TX.ECLK.Aligned	Generic DDR 10:1 Transmit Aligned Interface	Supports bypassed and dynamic data path delay. Supports tri-state control. With optional support to GDDR_SYNC soft logic.

Note:

The simulation initialization stage for RX and Aligned mode configurations may require more time than other modes due to the DDRDLL lock time requirement.

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 optional but recommended 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 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 optional but recommended for all Receive Centered and Transmit interfaces; while, RX_SYNC is required for all Receive Aligned Interfaces.

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.

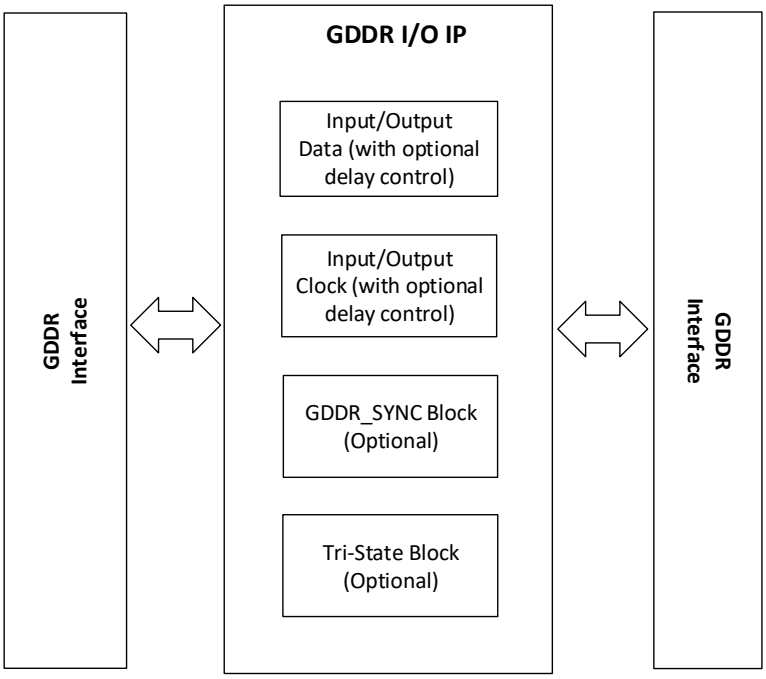


Figure 2.1. GDDR I/O Module Top-level Block Diagram

2.2. Functional Diagrams

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 500 Mb/s.

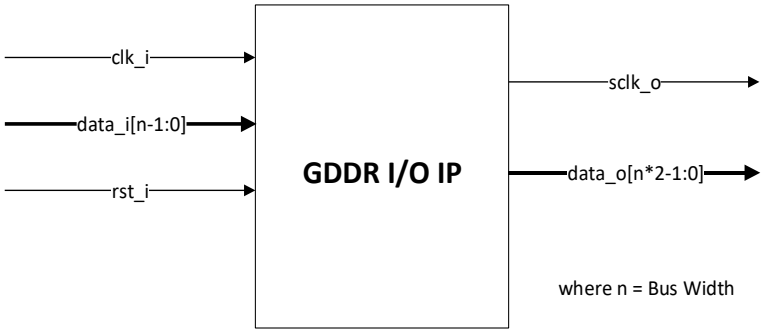


Figure 2.2. GDDRX1_RX.SCLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram

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, delay value is automatically set to 0, while Static User-defined interface has a fine delay value setting to override the data path delay.
- Output sampling clock, sclk_o, is routed from clk_i.

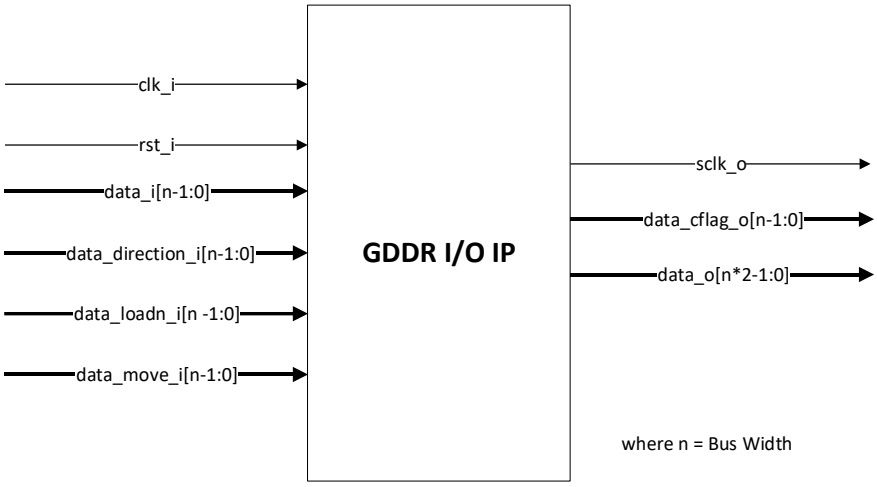


Figure 2.3. GDDRX1_RX.SCLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram

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_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.

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 500 Mb/s.

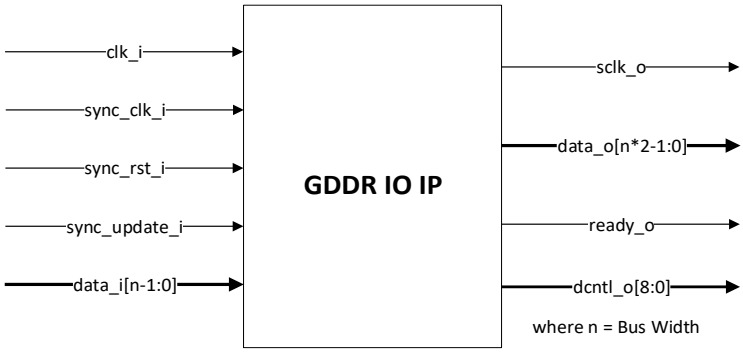


Figure 2.4. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram

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 this configuration, the delay value is set to 0. Static User-defined interface has a fine 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.

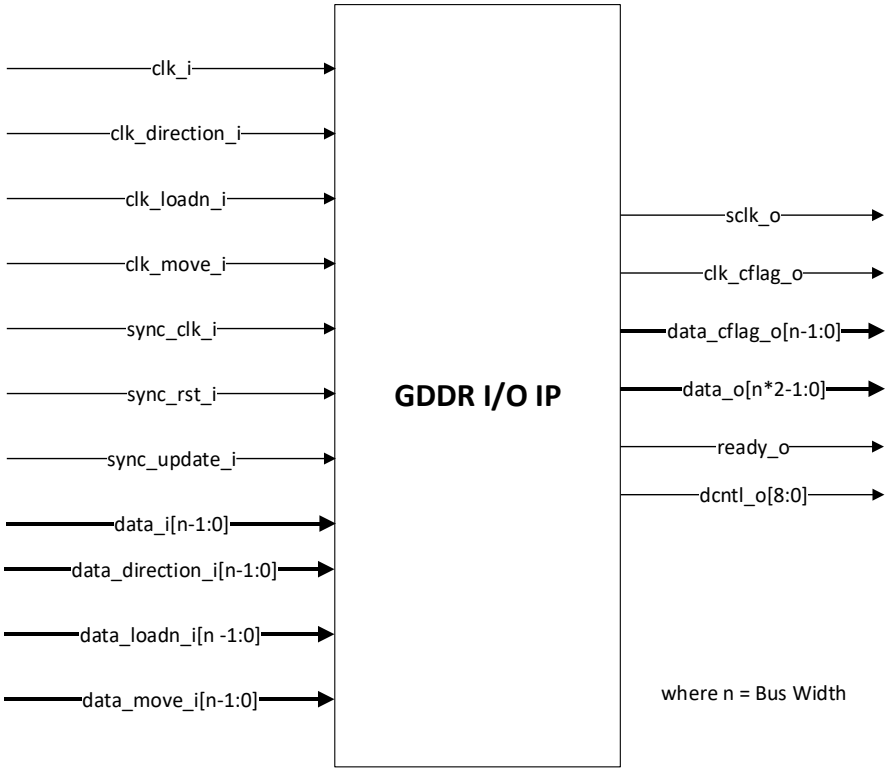


Figure 2.5. GDDRX1_RX.SCLK.Aligned Dynamic Default/User-Defined (Data) and Dynamic Delay (Clock) Block Diagram

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.
- 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_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.
- 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.

2.2.1. 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 500 Mb/s.

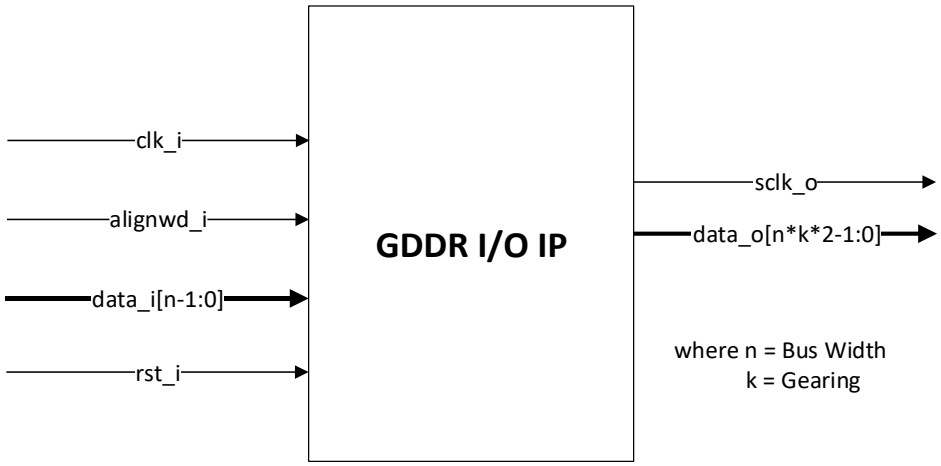


Figure 2.6. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram

Based on Figure 2.6, 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 this configuration, the delay value is set to 0. 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.

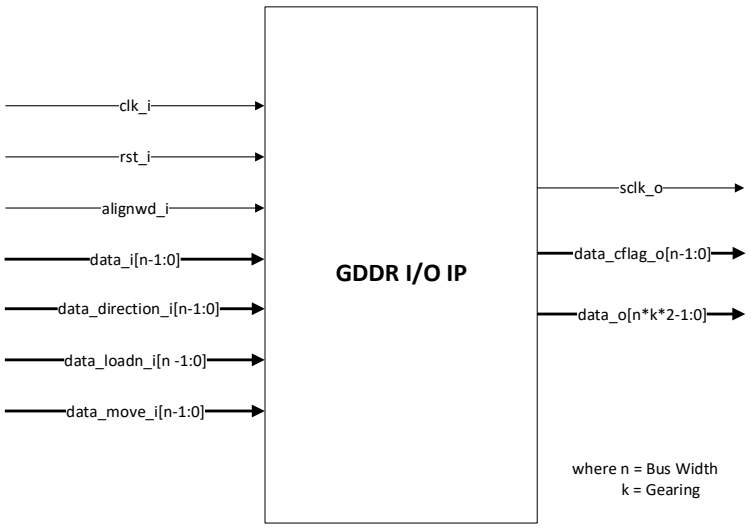


Figure 2.7. GDDRX2/4/5_RX.ECLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram

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.
- 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.

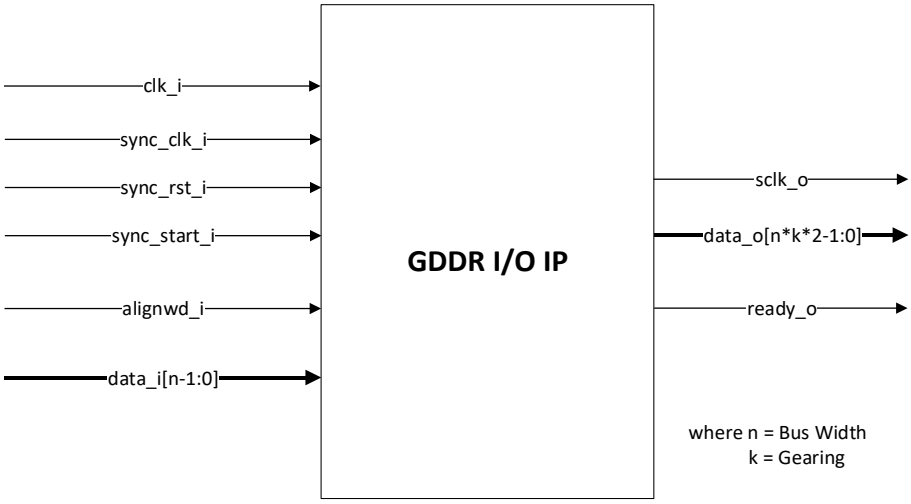


Figure 2.8. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC Block Diagram

Based on Figure 2.8, 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 this configuration, the delay value is set to 0. 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 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.

2.2.2. 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 500 Mb/s.

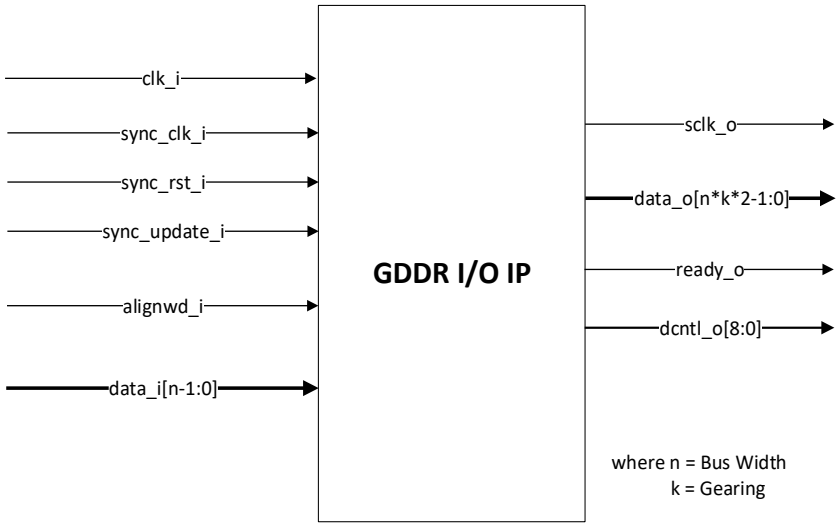


Figure 2.9. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram

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 this configuration, the delay value is set to 0. Static User-defined interface has a fine 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.

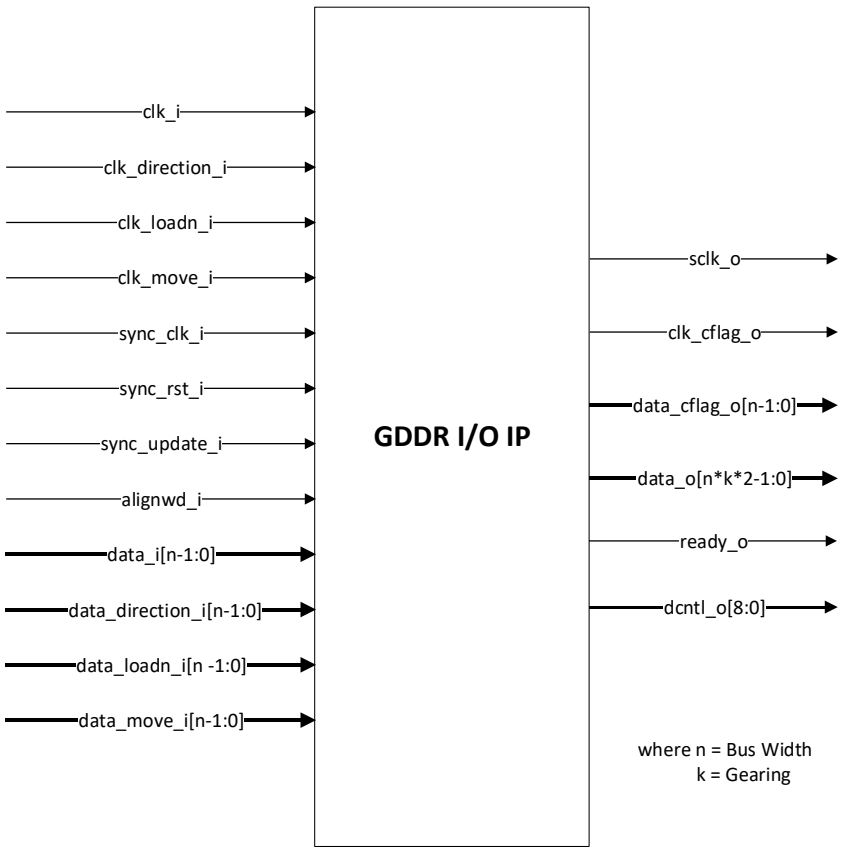


Figure 2.10. GDDRX2/4/5_RX.ECLK.Aligned Dynamic Default/Dynamic User-defined Delay Block Diagram

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_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.
- 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.

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 500 Mb/s.

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

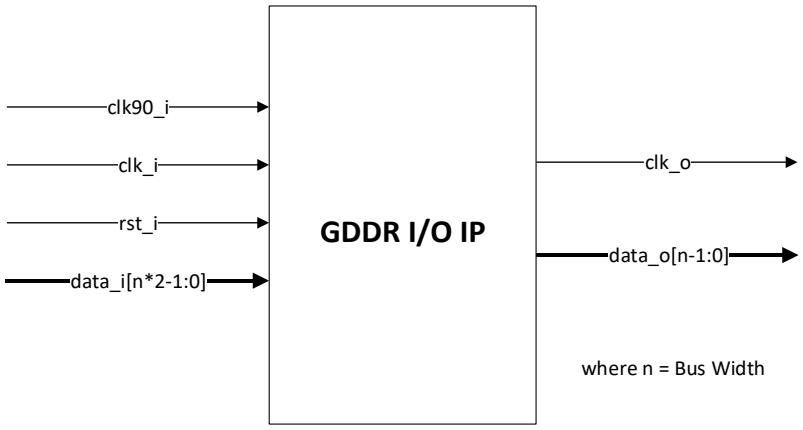


Figure 2.11. GDDRX1_TX.SCLK.Centered Bypass/Static User-defined Delay Block Diagram

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.

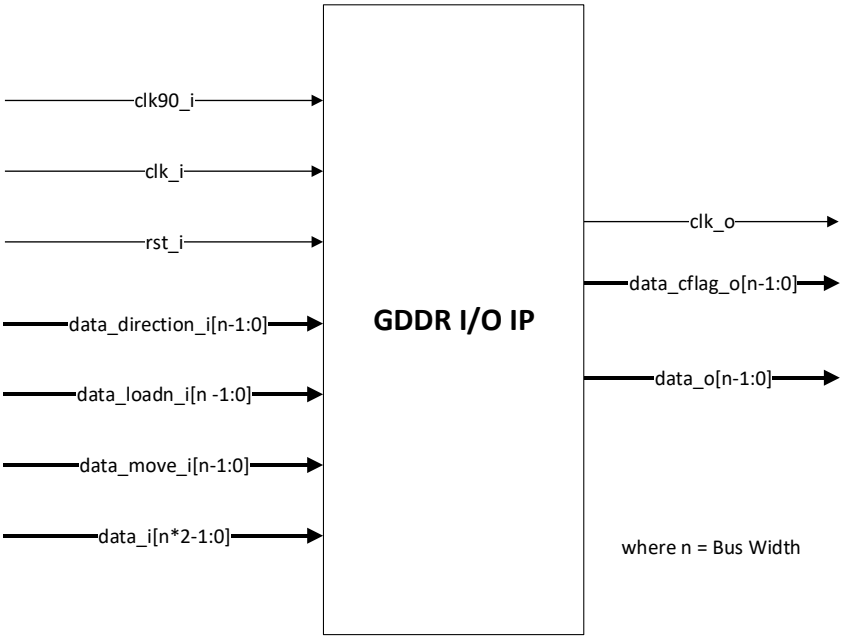


Figure 2.12. GDDRX1_TX.SCLK.Centered Dynamic User-defined Delay Block Diagram

Based on Figure 2.12, 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_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.

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 500 Mb/s.

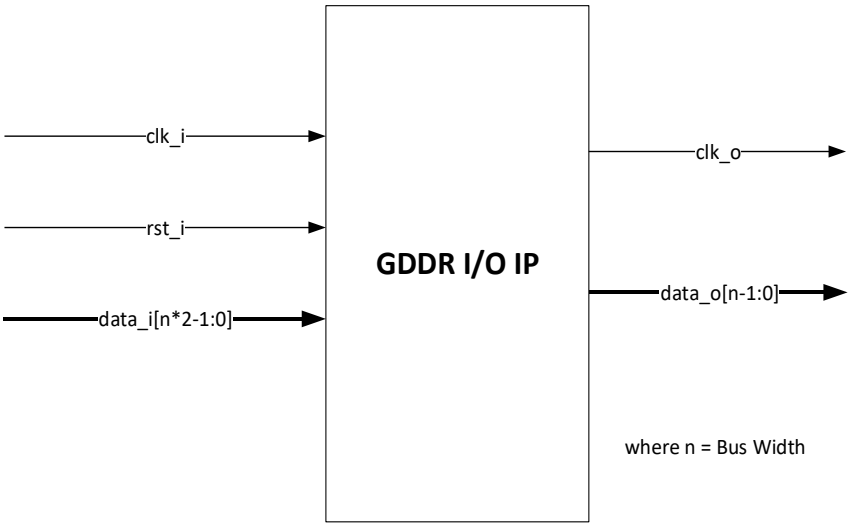


Figure 2.13. GDDRX1_TX.SCLK.Aligned Bypass/Static User-defined Delay Block Diagram

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.

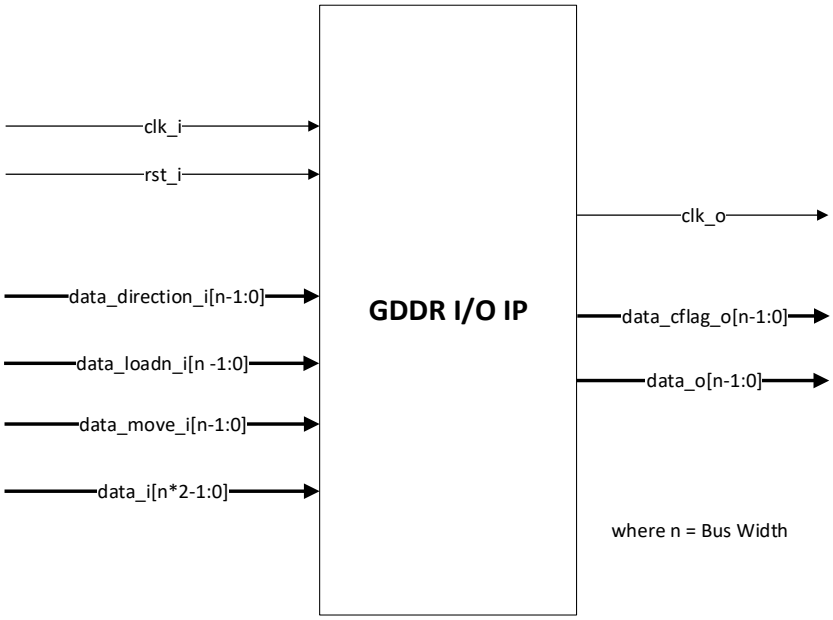


Figure 2.14. GDDRX1_TX.SCLK.Aligned Dynamic User-defined Delay Block Diagram

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_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.

2.2.3. 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 500 Mb/s.

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

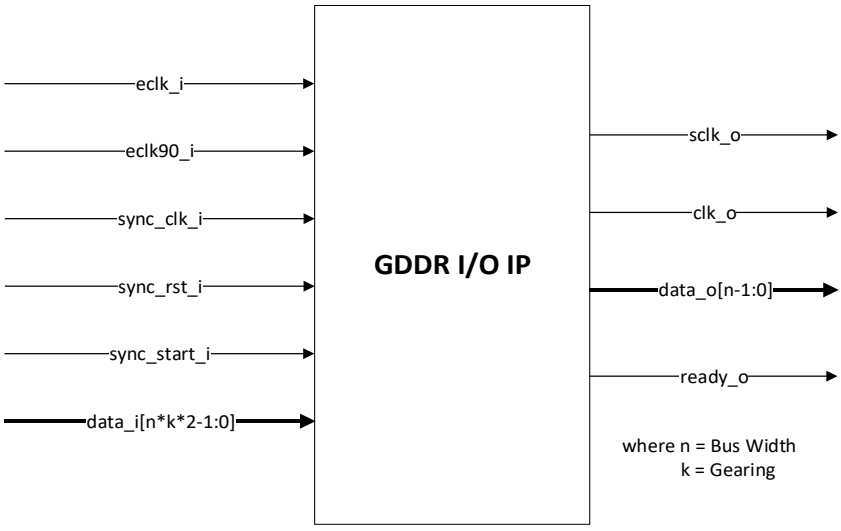


Figure 2.15. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay (with GDDR_SYNC) Block Diagram

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 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.

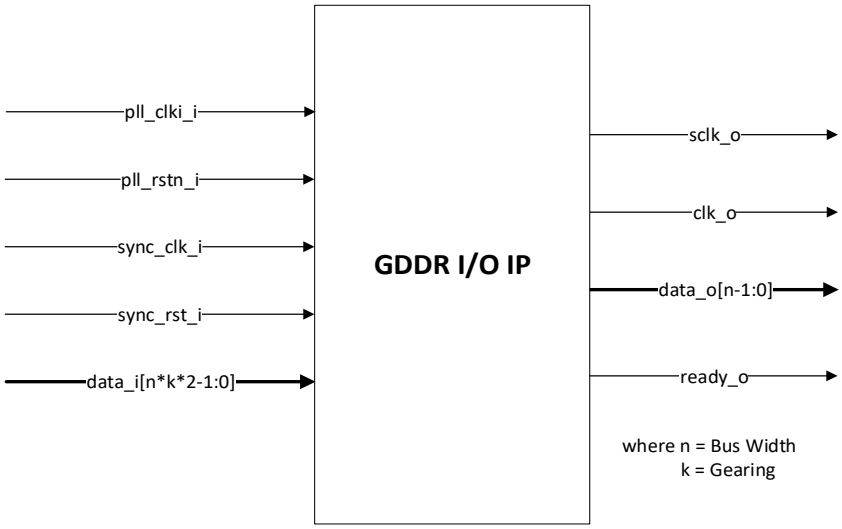


Figure 2.16. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay with PLL Block Diagram

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 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.

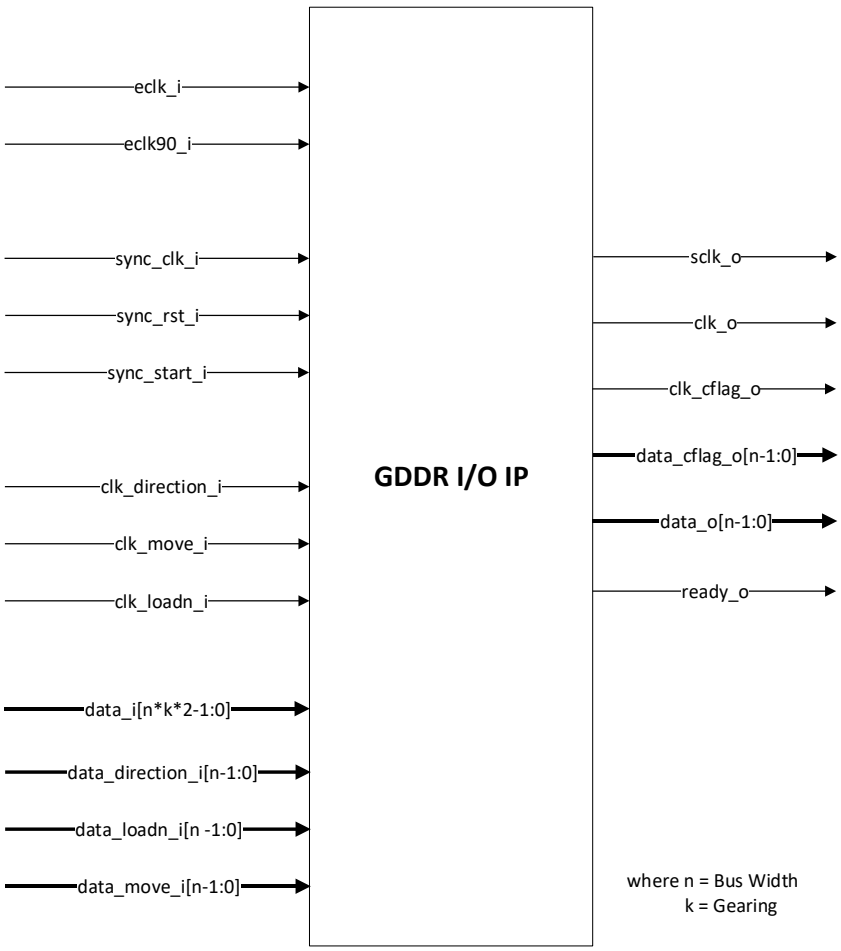
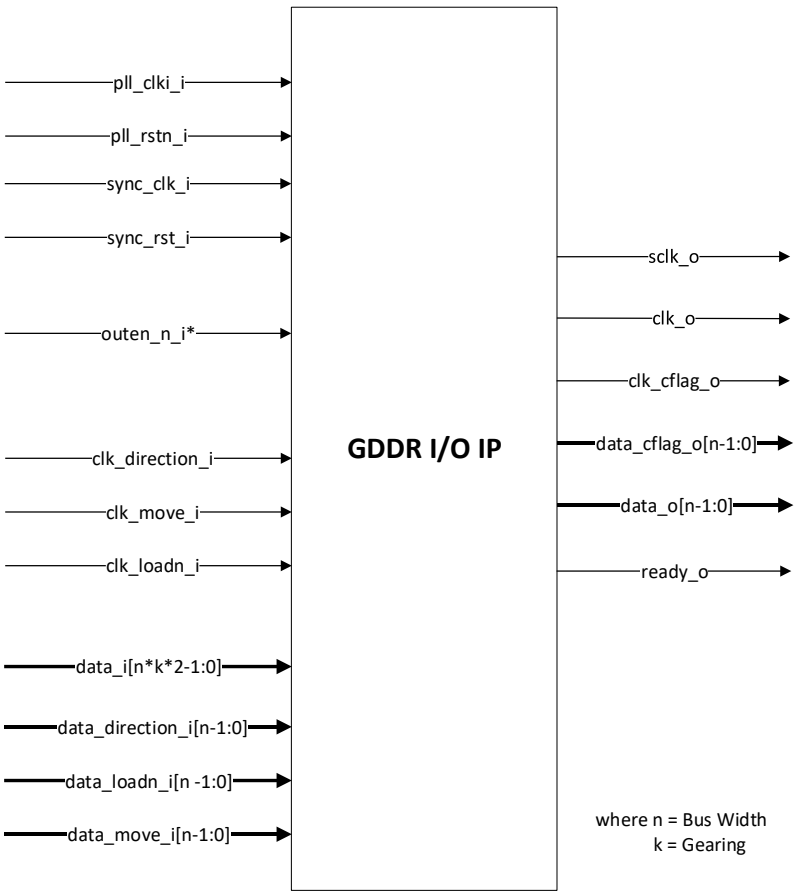


Figure 2.17. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram

The diagram presented in Figure 2.17 is true for both: Tri-state Control enabled and disabled cases, with outen_n_i present in the Tri-state 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_direction_i, data_loadn_i, and data_move_i input signals.
- Clock delay can also be dynamically controlled through 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 clock, eclk_i also serves as the clock used for capturing data of the DDR data components.
- 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_direction_i, data_loadn_i, and data_move_i input signals.
- Clock delay can also be dynamically controlled through 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.

2.2.4. 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 500 Mb/s.

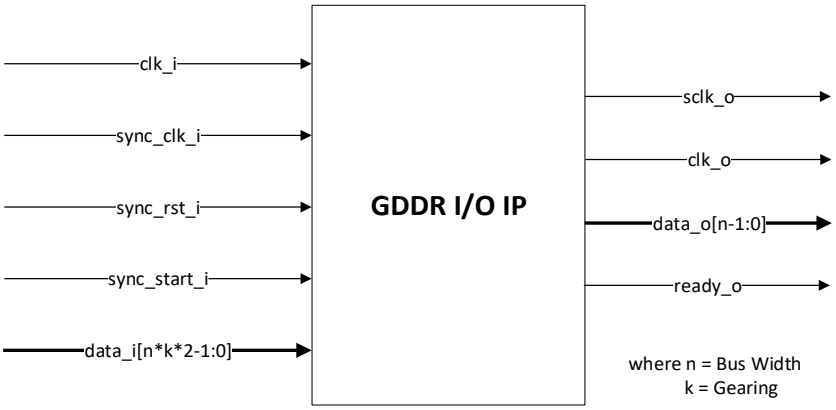


Figure 2.19. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay GDDR_SYNC Enabled Block Diagram

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 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.

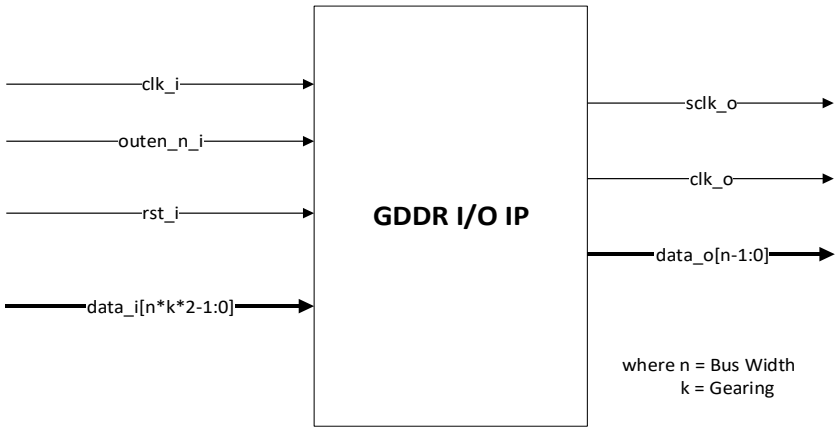


Figure 2.20. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay Tri-state Control Enabled Block Diagram

Based on Figure 2.20, 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.

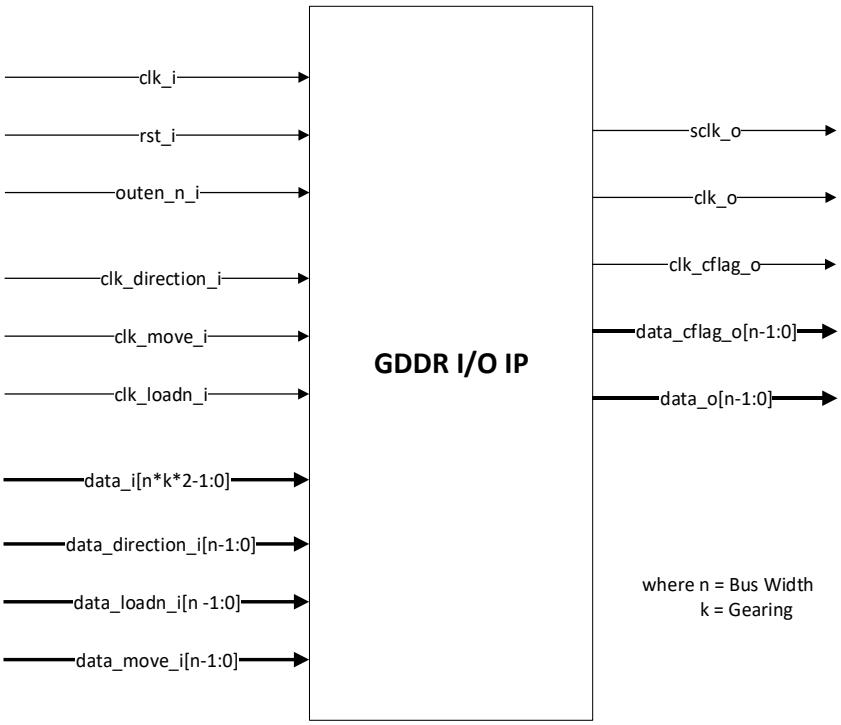


Figure 2.21. GDDRX2/4/5_TX.ECLK.Aligned Dynamic User-defined Delay Block Diagram

Based on Figure 2.21, 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_direction_i, data_loadn_i and data_move_i input signals.
- Clock delay can also be dynamically controlled through 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.

2.3. GDDR Behavior

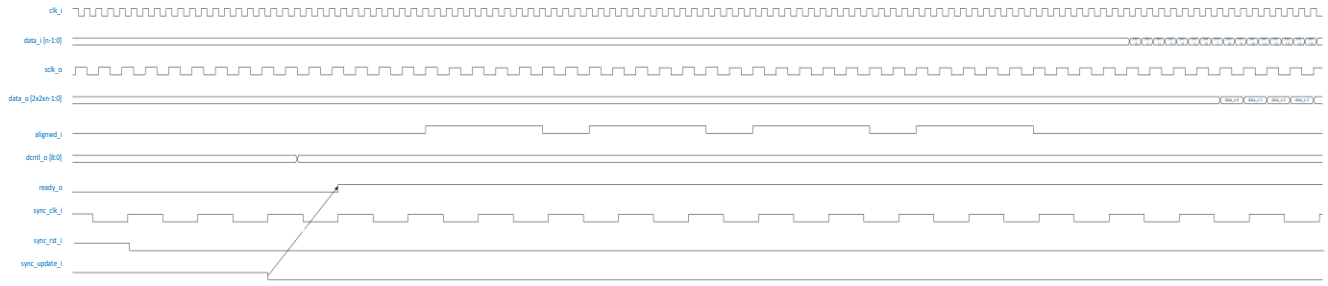


Figure 2.22. GDDR Timing Diagram (Receive Aligned)

Figure 2.22 shows the timing diagram of the signals of the Generic DDR IP and their relationship with one another. From the figure above, you can observe that input data, data_i is sampled using in the input clock, clk_i. After at least 3 SCLK cycles, the first output data, data_o appears which is the result of the first input data. The output data, data_o updates every sclk_o cycle and follows a first in first out condition which means that the first output data came from the first input data and so on. The signal alignwd_i is present when gearing is greater than X1 and is used for word alignment. This process is done internally in the ECLKDIVA primitive. Additionally, when Data Path Delay is set to Dynamic, the ports data_direction_i, data_loadn_i, data_move_i, and data_cflag_o are added. These signals are used to control the data dynamically and is done internally in the Delay Primitive (DELAYD, DELAYE, or DELAYF) depending on the configuration. Also, when Clock Path Delay is Dynamic, the ports clk_direction_i, clk_loadn_i, and clk_move_i are added. These signals are used to control the clock dynamically and is done internally in the DLLDELA primitive, hence this feature is only available when the Clock to Data Relations on the Pins is set to Aligned or Edge-to-Edge. For more information on the primitives mentioned, please refer to: FPGA-TN-02300 Lattice Avant High-Speed I/O and External Memory Interface User Guide

The configuration above uses the Module RX_SYNC upon startup for synchronization, hence the signals sync_clk_i, sync_rst_i, sync_update_i, dcntl_o, and ready_o is present. These signals follow the behavior defined in the module and is a separate entity from the IP having its own clock and reset with the assertion of the ready_o signal as a mark to start the transactions; hence, the IP waits for the signal ready_o to assert start the processes which usually takes 44 sync_clk_i cycles after the release of the reset of this module. As mentioned, the IP supports two soft logic modules upon start up with the GDDR_SYNC being the other which takes around 24 sync_clk_i cycles after the release of reset to assert the ready signal. Note that there is a gap between the deassertion of the signal sync_update_i and the assertion of ready_o which is not in scale in the figure above and that the timing behavior may vary on actual board.

2.3.1. RX Output Data Mapping

Bus Width = 4

Gearing = X2

data_i[4-1:0] = a[3:0], b[3:0], c[3:0], d[3:0]

data_o[2× 2×4-1:0] = {d[3:0], c[3:0], b[3:0], a[3:0]}

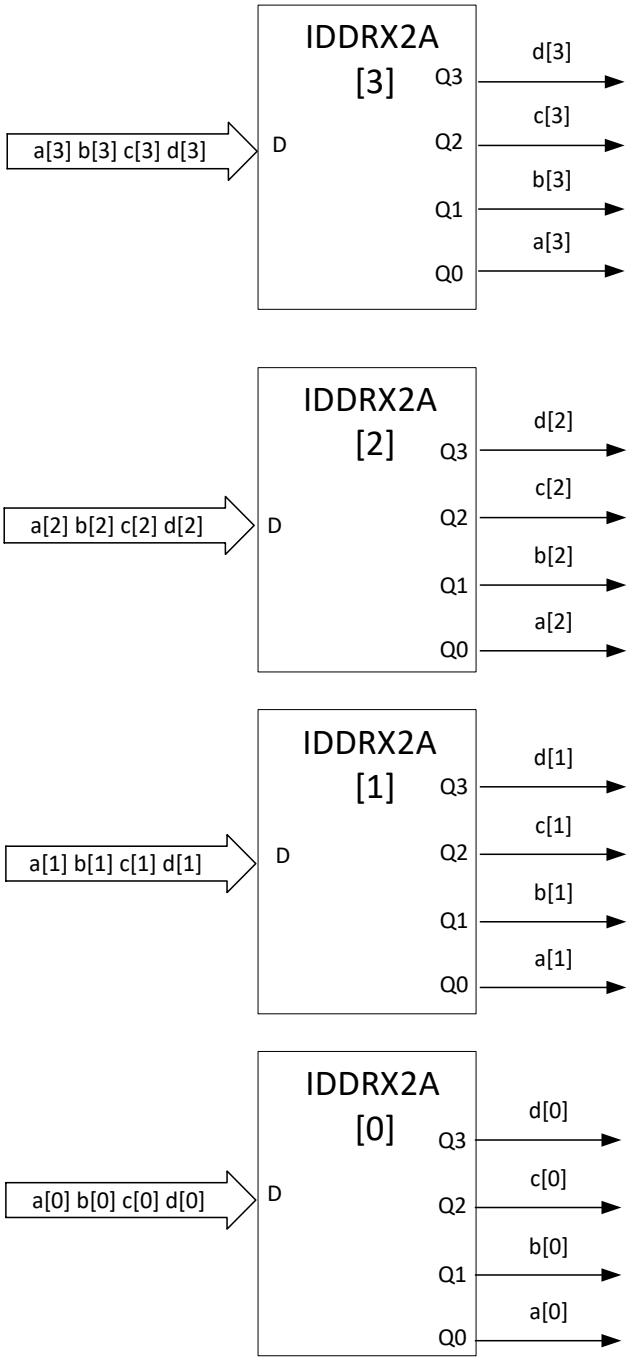


Figure 2.23. Rx Output Data Mapping Illustration

As shown in Figure 2.23, for GDDR Rx configuration, when Bus Width is 4 bits, the IP generates four IDDRX2A 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-02300 Lattice Avant High-Speed I/O and External Memory Interface User Guide for more details on User Primitives and Attributes.

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 IDDRX2As outputs is a data_o[15:0] vector.

Figure 2.24. shows the timing diagram for such example.

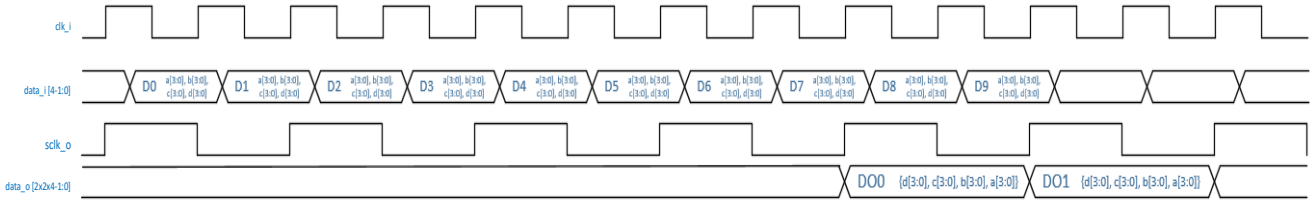


Figure 2.24. Rx (Centered) Timing Diagram

2.3.2. TX Input Data Mapping

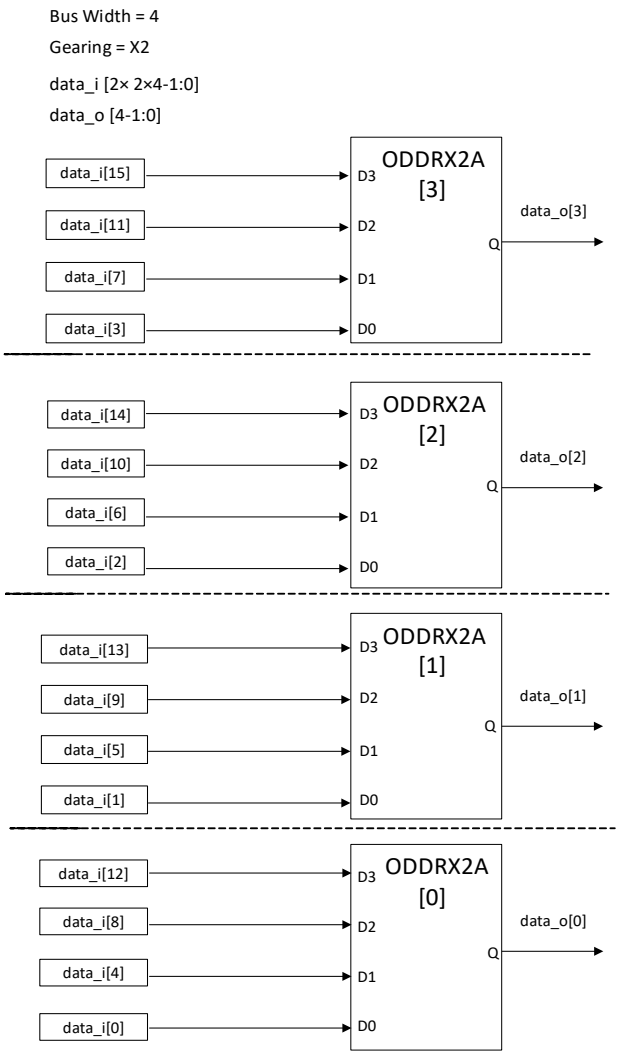


Figure 2.25. Tx Input Data Mapping Illustration

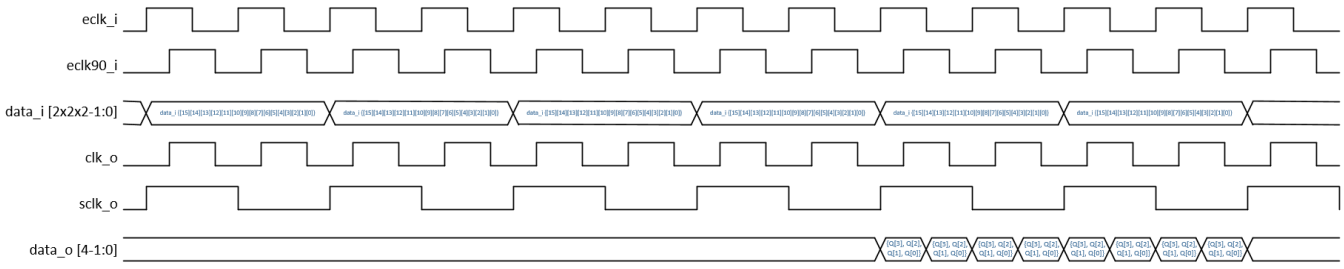


Figure 2.26. Tx (Centered) Timing Diagram

As shown in Figure 2.25, for GDDR Tx configuration, when Bus Width is 4 bits, the IP generates four ODDRX2A 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 data_i[3:0] is being transmitted first, followed by data_i[7:4] and so on. Figure 2.26 shows the timing diagram for Figure 2.25. Refer to FPGA-TN-02300 Lattice Avant High-Speed I/O and External Memory Interface User Guide for more details on User Primitives and Attributes.

2.3.3. RTL versus Post PAR Simulation Result

RTL simulation always uses the values based on the user interface settings. However, some of these settings will not produce the same expected result from RTL simulation due to difference in behavior between simulation model and silicon. Therefore, different settings derived from silicon characterization data are applied to primitives used such as DLLDEL, DELAYD, DELAYE, and DELAYF 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 in simulation. 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 DELAYD used on:
  - RX, X1, Aligned, Dynamic Data Path Delay, Dynamic Clock Path Delay
  - TX, X1, Aligned, Dynamic Data Path Delay, Fixed Clock Path Delay
  - RX, X1, Centered, Dynamic Data Path Delay, Fixed Clock Path Delay
  - TX, X1, Centered, Dynamic Data Path Delay
- On DELAYE used on:
  - RX, X1, Aligned, Bypass/Static Default/Dynamic Default/Static User Defined/Dynamic User Defined Data Path Delay, Fixed Clock Path Delay
  - RX, X1, Centered, Bypass/Static Default/Static User Defined Data Path Delay, Fixed Clock Path Delay
  - TX, X1, Aligned/Centered, Bypass/Static Default/Static User Defined Data Path Delay, Fixed Clock Path Delay
- On DELAYF used on:
  - TX/RX X2/X4/X5 Aligned/Centered

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*, *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	*FINE_DLY_STEP*
LAV-AT	*INTERFACE_TYPE_INPUT == RECEIVE, GEARING == X1, CLOCK_DATA_RELATION == Aligned,* and *DATA_PATH_DELAY == Bypass*	191
LAV-AT	*INTERFACE_TYPE_INPUT == RECEIVE, GEARING == X1, CLOCK_DATA_RELATION == Centered,* and *DATA_PATH_DELAY == Bypass*	191

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

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_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_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_o 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	Starts the sync process. Must be high during all synchronization process.
Ready_o	OUT	1	Indicates 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

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 jt 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_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.
lock_o	OUT	1	PLL lock output signal. Only available when PLL instantiation is enabled.
ready_o	OUT	1	Indicates that startup is finished, and TX circuit is ready to operate.
sync_start_i	IN	1	Used to re-start sync process. ready_o 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

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™ software.

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	X1 is available depending on the Clock Frequency and I/O Standard. • If I/O Standard is LVDS, HSUL12D, and SSTL135D the maximum frequency for X1 Configuration is 300 MHz. • If I/O Standard is LVCMOS33, LVCMOS25, and LVCMOS18 the maximum frequency for X1 Configuration is 200 MHz. • If I/O Standard is LVCMOS12 the maximum frequency for X1 Configuration is 100 MHz. X2 is available if *Clock Frequency <= 600 MHz* X4 and X5 are available if *Clock Frequency <= 900 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 - Maximum Fine Delay	0	*Data Path Delay == Static User-defined* or *Dynamic User-defined* Maximum Fine Delay = (1/ *Clock Frequency*)/14ps Note: In silicon, targeted delay step is 12.5 ps. In functional simulation, delay step that can be observed is 10 ps. Meanwhile 14 ps is the observed delay step on timing characterization.

Attribute	Selectable Values	Default	Dependency on Other Attributes
Clock Path Delay	Fixed Dynamic	Depends on the Configuration	***Dynamic*** *if:* *Interface Type == Transmit, Gearing != X1, Data Path Delay == Dynamic User Defined* *or* *Interface Type == Receive, Clock to Data Relations on the Pins == Edge-to-Edge, Data Path Delay == Dynamic Default or Dynamic User Defined* ***With options*** *if:* *Interface Type == Receive, Gearing == X1, Clock to Data Relations on the Pins == Edge-to-Edge, Data Path Delay == Bypass, Static Default, or Static User Defined* ***Fixed*** *for all other configuration not mentioned*
Include GDDR_SYNC	Checked, Not checked	Not checked	*Interface Type == Receive, Clock to Data Relationship on the Pins == Centered, Gearing Ratio != X1* and *Data Path Delay != Dynamic Default* nor *Dynamic User-defined* *Interface Type == Transmit, Clock to Data Relationship on the Pins == Edge-to-edge, Gearing Ratio != X1, Data Path Delay != Dynamic Default* nor *Dynamic User-defined* and *Tristate Control == Disabled*
Enable Tri-state Control	Checked, Not checked	Not checked	*Interface Type == Transmit, Clock to Data Relationship on the Pins == Centered, Gearing Ratio != X1* and *Data Path Delay == Dynamic User-defined* and *Clock Path Delay == Dynamic* *Interface Type == Transmit, Clock to Data Relationship on the Pins == Edge-to-edge, Gearing Ratio != X1* and *Data Path Delay != Dynamic User-defined* and *Clock Path Delay != Dynamic*
Clock Frequency for this Interface (MHz)	100 – 900	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	*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* if *Clock Frequency <= 800 MHz* else Max Value= 800 MHz
PLL Output Clock Tolerance (%)	0 – 10	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 the 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. It allows you to set default data path fine delay value 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. Ideally, a 9-bit binary string, 12.5 ps per step max_fine_delay = 511 × 12.5 ps = 6387.5 ps Note: The silicon is designed to have 12.5 ps delay per step but on simulation, only 10 ps can be observed.
Clock Path Delay	Fixed or dynamic addition of clock path delay
Include GDDR_SYNC	Enables usage of GDDR_SYNC support soft logic.
Enable Tri-state Control	Enables 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 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.

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**.

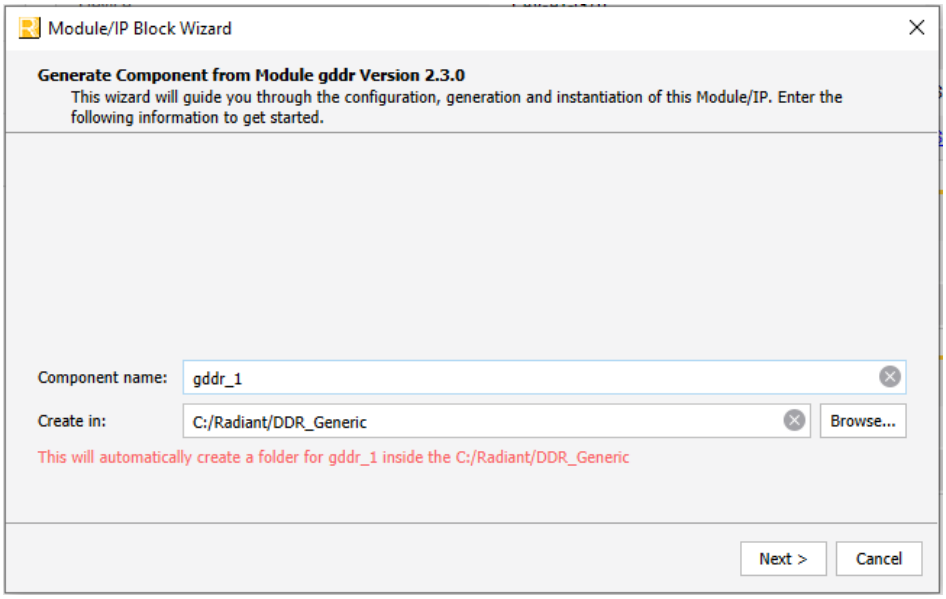


Figure 3.1. Module/IP Block Wizard

2. 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.

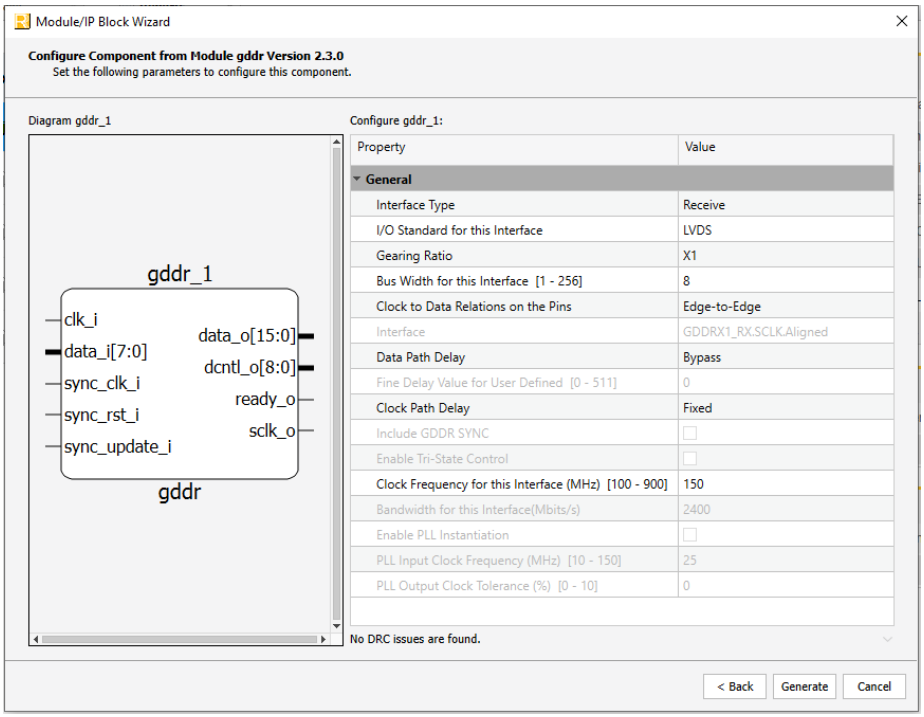


Figure 3.2. Configure Block of GDDR I/O Module

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

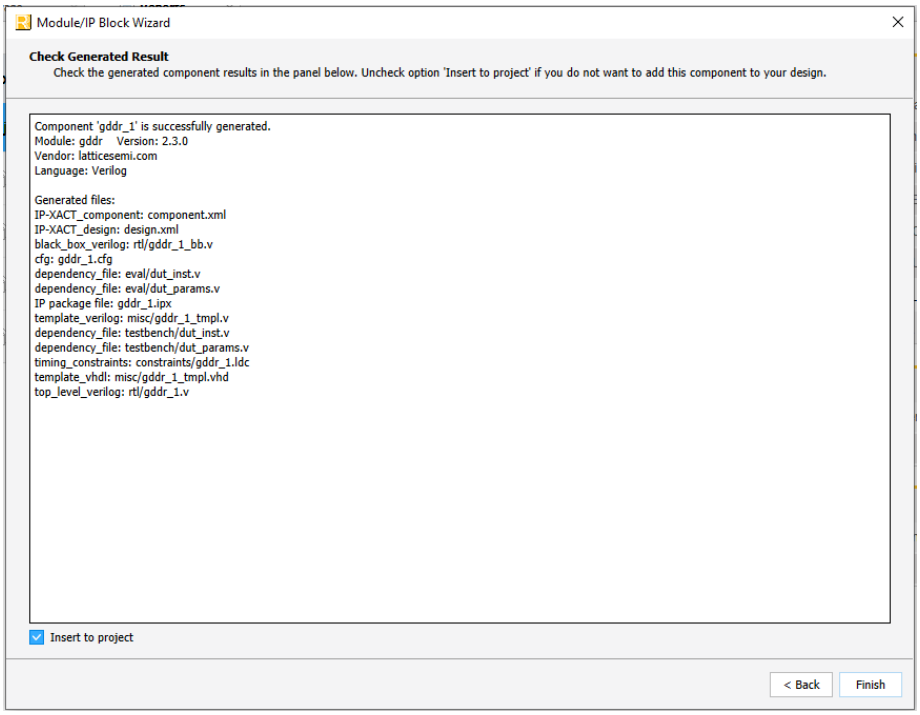


Figure 3.3. Check Generated Result

4. 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 black 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. user 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 black box.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the module.
eval/constraints.pdc	IP constraint file.
testbench/tb_top.v	Test bench template. You can edit this to match the specific needs.
testbench/dut_params.v	Instantiated version of the <IP_name>.v file for simulation use
testbench/dut_ints.v	Top level parameters of the generated RTL file

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

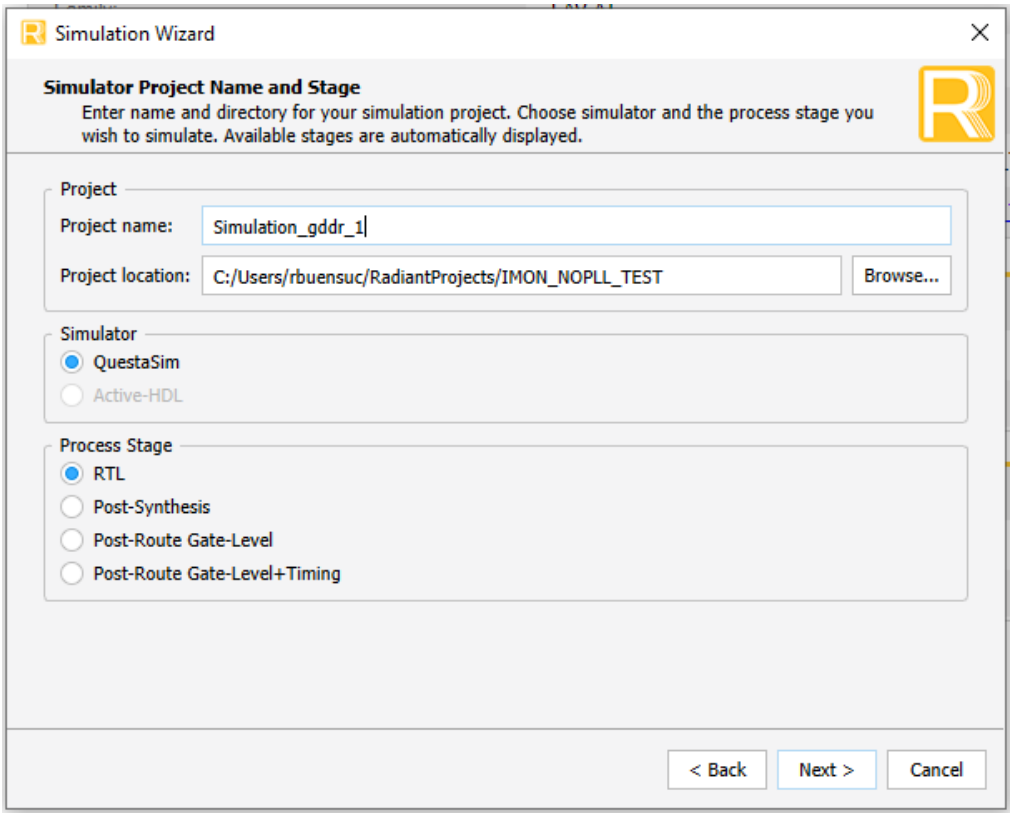


Figure 3.4. Simulation Wizard

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

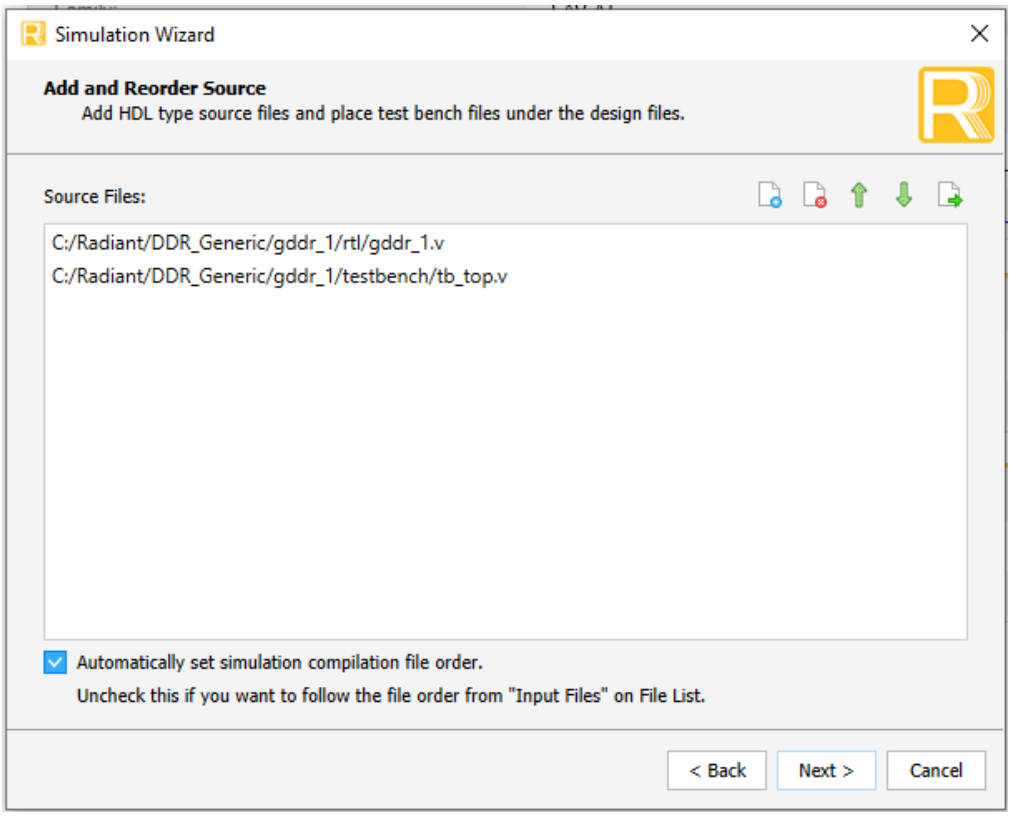


Figure 3.5. Adding and Reordering Source

3. Click **Next**. The Summary window is shown. Click **Finish** to run the simulation. The results of the simulation in our example are provided in figure below.

Notes:

- It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant Software Suite
- For configurations with RX and Aligned mode, the DDRDLL requires additional time to achieve lock during simulation startup, which impacts simulation runtime. For a configuration using 100 MHz, the DDRDLL lock time is approximately 334 µs in simulation. At higher frequencies, the DDRDLL lock time is shorter

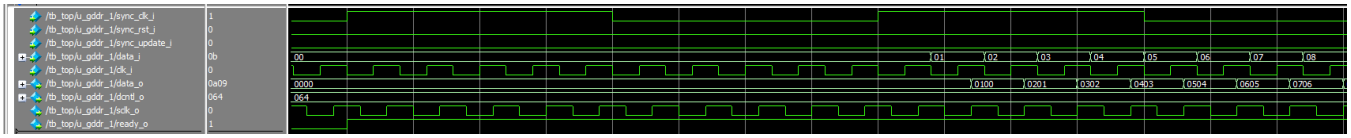


Figure 3.6. Simulation Waveform

3.3. Constraining the IP

You need to provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints:

<IP_Instance_Path>/<IP_Instance_Name>/eval/constraints.pdc.

The constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.

To use this constraint file, copy the content of *constraints.pdc* to the top-level design constraint for post-synthesis.

Refer to Lattice Radiant Timing Constraints Methodology for details on how to constraint your design.

3.4. IP Evaluation

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

Appendix A. Resource Utilization

Table A.1 and Table A.2 shows resource utilization of DDR Generic configurations using LAV-AT-E70-3LFG1156I and LAV-AT-E70-1LFG1156I devices with Synplify Pro of Lattice Radiant software. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
8-bit Receive X1 Edge-to-Edge Bypass	250	44	108	0	0
8-bit Receive X2 Centered Bypass	300	0	0	0	0
16-bit Receive X4 Edge-to-Edge Static Default	225	44	112	0	0
4-bit Receive X5 Centered Dynamic Default	180	0	4	0	0
8-bit Transmit X1 Edge-to-Edge Bypass	250	0	0	0	0
16-bit Transmit X2 Centered Bypass	300	12	25	0	0
16-bit Transmit X4 Centered Static User Defined with 10 Fine Delay value	225	12	25	0	0
4-bit Transmit X5 Centered Dynamic User Defined	180	12	30	0	0

Notes:

1. Fmax is generated when the FPGA design only contains the Generic DDR module and the target frequency is 200 MHz for X1, X2, and X4 gearing and 180 MHz for X5 gearing. Generic DDR module supports SCLK frequency up to 250 MHz for X1, 300 MHz for X2, 225 MHz for X4 and 180 MHz for X5 gearing. The obtained Fmax 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 using LAV-AT-E70-1LFG1156I

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
8-bit Receive X1 Edge-to-Edge Bypass	250	44	108	0	0
8-bit Receive X2 Centered Bypass	300	0	0	0	0
16-bit Receive X4 Edge-to-Edge Static Default	225	44	112	0	0
4-bit Receive X5 Centered Dynamic Default	180	0	4	0	0
8-bit Transmit X1 Edge-to-Edge Bypass	250	0	0	0	0
16-bit Transmit X2 Centered Bypass	300	12	25	0	0
16-bit Transmit X4 Centered Static User Defined with 10 Fine Delay value	225	12	25	0	0
4-bit Transmit X5 Centered Dynamic User Defined	180	12	30	0	0

Notes:

1. Fmax is generated when the FPGA design only contains the Generic DDR module and the target frequency is 200 MHz for X1, X2, and X4 gearing and 180 MHz for X5 gearing. Generic DDR module supports SCLK frequency up to 250 MHz for X1, 300 MHz for X2, 225 MHz for X4 and 180 MHz for X5 gearing. The obtained Fmax 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

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, see the Lattice Radiant software user guide.

- Lattice Avant-E FPGA web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

For frequently asked questions, refer to the Lattice Answer Database at 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.

Revision 1.3, April 2026

Section	Change Summary
All	Add IP version on the cover page.
Abbreviations in This Document	Renamed section title.
Functional Description	Added the following table note to Table 2.1. Available GDDR I/O Module Interfaces: *The simulation initialization stage for RX and Aligned mode configurations may require more time than other modes due to DDRDLL lock time requirement.*
IP Generation, Simulation, and Validation	Added the following note: *For configurations with RX and Aligned mode, the DDRDLL requires additional time to achieve lock during simulation startup, which impacts simulation runtime. For a configuration using 100 MHz, the DDRDLL lock time is approximately 334 µs in simulation. At higher frequencies, the DDRDLL lock time is shorter.*

Revision 1.2, June 2024

Section	Change Summary
All	Made editorial fixes.
Introduction	• Added device configuration for *GDDR_SYNC*. • Added *RX_SYNC soft IP logic* with device configuration.

Functional Description	• Updated the comments of the following features in Table 2.1. Available GDDR I/O Module Interfaces:
	  • *GDDRX1_RX.SCLK.Aligned*
	  • *GDDRX2_RX.ECLK.Centered*
	  • *GDDRX2_RX.ECLK.Aligned*
	  • *GDDRX4_RX.ECLK.Centered*
	  • *GDDRX4_RX.ECLK.Aligned*
	  • *GDDRX5_RX.ECLK.Centered*
	  • *GDDRX5_RX.ECLK.Aligned*
	  • *GDDRX2_TX.ECLK.Centered*
	  • *GDDRX2_TX.ECLK.Aligned*
	  • *GDDRX4_TX.ECLK.Centered*
	  • *GDDRX4_TX.ECLK.Aligned*
	  • *GDDRX5_TX.ECLK.Centered*
	  • *GDDRX5_TX.ECLK.Aligned*
	• Added information of *Synchronization Modules* for Table 2.2. Summary of GDDR Support Soft Logic.
	• Updated the *interface function* description for the following figures:
	  • Figure 2.2. GDDRX1_RX.SCLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram
	  • Figure 2.4. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram
	  • Figure 2.5. GDDRX1_RX.SCLK.Aligned Dynamic Default/User-Defined (Data) and Dynamic Delay (Clock) Block Diagram
	  • Figure 2.6. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram
	  • Figure 2.8. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC Block Diagram
	  • Figure 2.9. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram

Section	Change Summary
	• Figure 2.10. GDDRX2/4/5_RX.ECLK.Aligned Dynamic Default/Dynamic 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 (with GDDR_SYNC) Block Diagram • Figure 2.17. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram • Updated the description for the following figures Figure 2.22. Rx Output Data Mapping Illustration and Figure 2.24. Tx Input Data Mapping Illustration: • Added Figure 2.23. Rx (Centered) Timing Diagram and Figure 2.25. Tx (Centered) Timing Diagram • Added section 2.3.3 RTL versus Post PAR Simulation Result. • Updated the description of the following ports in Table 2.3. GDDR I/O Module Receive Signal Description: • *sync_clk_i* • *sync_rst_i* • *sync_update_i* • *ready_o* • Updated the description of the following ports in Table 2.4. GDDR I/O Module Transmit Signal Description: • *clk_i* • *clk90_i* • *sync_clk_i* • *sync_rst_i* • Added new ports in Table 2.4. GDDR I/O Module Transmit Signal Description: • *pll_clki_i* • *pll_rstn_i* • Updated the *Dependency on Other Attributes* and *Selected Values* of the following attribute in Table 2.5. Attributes Table: • *Gearing Ratio* • *Clock Path Delay* • *Enable Tristate Control* • *PLL Output Clock Tolerance* • Removed the *PLL Output Clock Frequency Actual Value (MHz)* attribute in Table 2.5. Attributes Table and Table 2.6. Attribute Description.
IP Generation, Simulation, and Validation	Updated the following figures from Figure 3.1. Module/IP Block Wizard - Figure 3.5. Adding and Reordering Source.

Revision 1.1, January 2024

Section	Change Summary
All	Updated the document title to *Avant DDR Generic Module*.
Disclaimers	Updated this section.
Inclusive Language	Newly added this section.
Functional Description	• Newly added corresponding interface function descriptions for the following figures: Figure 2.2. GDDRX1_RX.SCLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram; Figure 2.3. GDDRX1_RX.SCLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram; Figure 2.4. GDDRX1_RX.SCLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram; Figure 2.5. GDDRX1_RX.SCLK.Aligned Dynamic Default/User-Defined (Data) and Dynamic User-defined Delay (Clock) Block Diagram; Figure 2.6. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Block Diagram;

	Figure 2.7. GDDRX2/4/5_RX.ECLK.Centered Dynamic Default/Dynamic User-defined Delay Block Diagram; Figure 2.8. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC Block Diagram; Figure 2.9. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Delay Block Diagram; Figure 2.10. GDDRX2/4/5_RX.ECLK.Aligned Dynamic Default/Dynamic User-defined Delay Block Diagram; 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 2.13. GDDRX1_TX.SCLK.Aligned Bypass/Static User-defined Delay Block Diagram; Figure 2.14. GDDRX1_TX.SCLK.Aligned Dynamic User-defined Delay Block Diagram; Figure 2.15. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay (with GDDR_SYNC) Block Diagram; Figure 2.16. GDDRX2/4/5_TX.ECLK.Centered Bypass/Static User-defined Delay with PLL Block Diagram; Figure 2.17. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram; Figure 2.18. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay with PLL; Figure 2.19. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay GDDR_SYNC Enabled Block Diagram; Figure 2.20. GDDRX2/4/5_TX.ECLK.Aligned Bypass/Static User-defined Delay Tri-state Control Enabled Block Diagram; Figure 2.21. GDDRX2/4/5_TX.ECLK.Aligned Dynamic User-defined Delay Block Diagram. Removed the following figures of the previous version: Figure 2.3, Figure 2.5, Figure 2.7, Figure 2.9, Figure 2.11, Figure 2.13, Figure 2.15, Figure 2.17, Figure 2.19, Figure 2.21, Figure 2.22, Figure 2.24, Figure 2.26, Figure 2.27, Figure 2.29, Figure 2.31, Figure 2.32, Figure 2.34, Figure 2.35, Figure 2.36, Figure 2.38, Figure 2.40, Figure 2.42, Figure 2.43, Figure 2.45, Figure 2.46, Figure 2.48.
IP Generation, Simulation, and Validation	• Updated Figure 3.1. Module/IP Block Wizard, Figure 3.2. Configure Block of GDDR I/O Module, and Figure 3.3. Check Generated Result. In Table 3.1. Generated File List, newly added the *eval/constraints.pdc* attribute. • Newly added the Constraining the IP section.
Appendix A. Resource Utilization	• Changed device name to *LAV-AT-E70-3LFG1156I* in the description and caption of Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I. • Changed device name to *LAV-AT-E70-1LFG1156I* in the description and caption of Table A.2. Resource Utilization using LAV-AT-E70-1LFG1156I.
References	Newly added the link to Lattice Insights.

Revision 1.0.1, June 2023

Section	Change Summary
Functional Description	Updated below figures: • Figure 2.11. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Interface • Figure 2.13. GDDRX2/4/5_RX.ECLK.Centered Dynamic Default/Dynamic User-defined Delay Interface • Figure 2.15. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC Interface • Figure 2.17. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Interface • Figure 2.19. GDDRX2/4/5_RX.ECLK.Aligned Dynamic Default/Dynamic User-defined Delay Interface • Figure 2.31. GDDRX2/4/5_TX.ECLK.Centered Bypass (with GDDR_SYNC) Interface • Figure 2.32. GDDRX2/4/5_TX.ECLK.Centered Static User-defined Delay (with GDDR_SYNC) Interface • Figure 2.34. GDDRX2/4/5_TX.ECLK.Centered Bypass for PLL Interface • Figure 2.35. GDDRX2/4/5_TX.ECLK.Centered Static User-defined Delay for PLL Interface • Figure 2.38. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay (with GDDR_SYNC) Block Diagram • Figure 2.40. GDDRX2/4/5_TX.ECLK.Centered Dynamic User-defined Delay for PLL Interface

Section	Change Summary
	• Figure 2.42. GDDRX2/4/5_TX.ECLK.Aligned Bypass with GDDR_SYNC Enabled Interface • Figure 2.43. GDDRX2/4/5_TX.ECLK.Aligned Static User-defined Delay with GDDR_SYNC Enabled Interface • Figure 2.45. GDDRX2/4/5_TX.ECLK.Aligned Bypass Tristate Control Enabled Interface • Figure 2.46. GDDRX2/4/5_TX.ECLK.Aligned Static User-defined Delay Tristate Control Enabled Interface • Figure 2.48. GDDRX2/4/5_TX.ECLK.Aligned Dynamic User-defined Delay Interface
IP Generation, Simulation, and Validation	Updated below figures for updating gddr version from 2.0.0 to 2.1.0: • Figure 3.1. Module/IP Block Wizard • Figure 3.2. Configure Block of GDDR I/O Module • Figure 3.3. Check Generated Result
Technical Support Assistance	Added Lattice FAQ website link in Technical Support Assistance section.
Reference	Added link for Lattice Avant-E.

Revision 1.0, October 2022

Section	Change Summary
Functional Description	Updated below figures: • Figure 2.11. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay Interface • Figure 2.15. GDDRX2/4/5_RX.ECLK.Centered Bypass/Static Default/Static User-defined Delay with GDDR_SYNC • Interface • Figure 2.17. GDDRX2/4/5_RX.ECLK.Aligned Bypass/Static Default/Static User-defined Interface • Figure 2.32. GDDRX2/4/5_TX.ECLK.Centered Static User-defined Delay (with GDDR_SYNC) Interface • Figure 2.35. GDDRX2/4/5_TX.ECLK.Centered Static User-defined Delay for PLL Interface • Figure 2.43. GDDRX2/4/5_TX.ECLK.Aligned Static User-defined Delay with GDDR_SYNC Enabled • Figure 2.46. GDDRX2/4/5_TX.ECLK.Aligned Static User-defined Delay Tristate Control Enabled
IP Generation, Simulation, and Validation	Updated below figures: • Figure 3.1. Module/IP Block Wizard • Figure 3.2. Configure Block of GDDR I/O Module • Figure 3.3. Check Generated Result
Resource Utilization	Added Appendix A. Resource Utilization section.

Revision 0.8, May 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com