# LatticeECP3 High-Speed I/O Interface

# Technical Note

FPGA-TN-02184-2.6

June 2023

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

## Contents

# Figures

# Tables

# Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| CSM | Clock Synchronization Module |
| DDR | Double Data Rate |
| DLL | Delay Locked Loops |
| DM | Data Mask |
| ECLK | Edge Clock |
| FPGA | Field Programmable Gate Array |
| I/O | Input/Output |
| IOL | Input/Output Logic |
| PAR | Place and Route |
| PCLK | Primary Clock |
| PIO | Programmable Input/Output |
| PLL | Phase Locked Loops |
| POR | Power On Reset |
| SCLK | System Clock |
| SDR | Single Data Rate |
| SDRAM | Synchronous Dynamic Random Access Memory |

# 1. Introduction

LatticeECP3™ devices support high-speed I/O interfaces, including Double Data Rate (DDR) and Single Data Rate (SDR) interfaces, using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance. LatticeECP3 I/O also have dedicated circuitry that is used along with the DDR I/O to support DDR, DDR2 and DDR3 SDRAM memory interfaces. Refer to the LatticeECP3 Family Data Sheet (FPGA-DS-02074) for a detailed description of the I/O logic architecture.

This document discusses how to utilize the capabilities of the LatticeECP3 "E" and "EA" devices to implement both the high-speed generic DDR interface and the DDR and DDR2 memory interfaces. Refer to the Implementing DDR/DDR2/DDR3 Memory Interfaces section of this document for more information.

There are some differences in high-speed interface architecture between the LatticeECP3 "EA" and "E" devices. The implementation differences between the two devices are indicated in the appropriate sections of this document.

This document assumes that version 8.0 of the ispLEVER® software is used for all interfaces. Please see exceptions under each section if you are using the ispLEVER 7.2 SP2.

## 1.1. Designing a High-Speed DDR Interface

The following steps must be followed to successfully design a high-speed DDR interface using this document.

- **Step 1: Determine the type of interface to implement**.
  Based on the external interface determine the type of high-speed interface to be built. See the Types of High-Speed DDR Interfaces section for details.
- Step 2: Use the ispLEVER IPexpress™ tool to build the interface.
  Once you have determined the type of interface to be built, use IPexpress to build the interface. See the section Using IPexpress to Build High-Speed DDR Interfaces.
- **Step 3: Understand design rules, clocking requirements for each interface**.
  Understand the architecture, interface rules and clocking requirements for each of the interfaces. See the High-Speed DDR Interface Details section for a detailed description of each interface. If multiple interfaces are used in one device, it is critical to follow these design rules to avoid resource conflicts between interfaces.
- **Step 4: Review pinout requirements for clock and data pins for each interface before making pin assignments**. It is critical that the interface clock and data pins follow the placement recommendations listed in the section Placement Guidelines for High-Speed DDR Interfaces. If using interfaces requiring DQ-DQS grouping, follow the rules described in the section DQ-DQS Grouping Rules.
- **Step 5: Assign timing preferences and clock preferences**.
  Follow the guidelines in the section Timing Analysis for High-Speed DDR Interfaces to assign timing preferences for the interfaces.
- **Step 6: Run software Place and Route without DRC and Trace**.
  It is important that the software Place and Route tool pass without any DRC errors. If it runs into DRC errors review the sections described above to make sure you are not violating any of the design/pinout rules for the interface. Run Trace to look at the static timing analysis results for all the timing preferences added to the design. Make design changes as necessary to assure that you are meeting your timing requirements.

## 1.2. Generating a Valid Pinout for a High-Speed DDR Interface

Due to the various design rules and pinout requirements for each interface, it is critical that the interfaces be created as described above and run through the software before designing the PCB. If it is necessary to determine pinouts for a PCB design before the complete design is in place, You must follow the steps listed below to create the pinouts.

- **Step 1: Determine the type of interface to implement**.
  Based on the external interface, determine the type of high-speed interface to be built. See the section Types of High-Speed DDR Interfaces for details.
- **Step 2: Use the ispLEVER IPexpress Tool to build the interface**.
  Once you have determined the type of interface to be built, IPexpress should be used to build the interface. See the section Using IPexpress to Build High-Speed DDR Interfaces.
- **Step 3: Build the complete I/O ring and clocking structure**.
  Some dummy logic may be required to assure that the data out of the DDR elements are not optimized out by the software. For example, if you are building an input and a corresponding output interface you may connect them using dummy register between the two interfaces. If you are building receive-only interfaces, the signals out of the receive interface need to be used in dummy logic or assigned to outputs. Similarly, if building a transmit- only interface you must provide input signals that are used in the transmit interface.
- **Step 3: Understand the design rules and clocking requirements for each interface**.
  Understand the architecture, interface rules and clocking requirements for each of the interfaces. See the High-Speed DDR Interface Details section for a detailed description of each interface. If multiple interfaces are used in one device, it is critical to follow these design rules to avoid resource conflicts between interfaces.
- **Step 4: Make pin assignments for clock and data pins for each interface following the design and pinout rules**.
  Assign the input and output pins to specific sites, banks or DQS groups. Review the pinout requirements for the clock and data pins for each interface before making pin assignments. It is critical that the interface clock and data pins follow the placement recommendations listed in the section Placement Guidelines for High-Speed DDR Interfaces. If using interfaces requiring a DQ-DQS grouping, you must follow the rules described in the section DQ-DQS Grouping Rules.
- **Step 5: Run the design through ispLEVER Place and Route and pass without any DRC errors**.
  If you run into errors, change pin assignments as required to pass all the DRC checks. When the design passes Place and Route, the pin assignments listed in the PAD report can be used on the board.

## 2. External Interface Description

This technical note uses two types of external interface definitions, centered and aligned. In a centered external interface, at the device pins, the clock is centered in the data opening. In an aligned external interface, at the device pins, the clock and data transition are aligned. This is also sometimes called “edge-on-edge”. Figure 2.1 shows the external interface waveform for SDR and DDR.

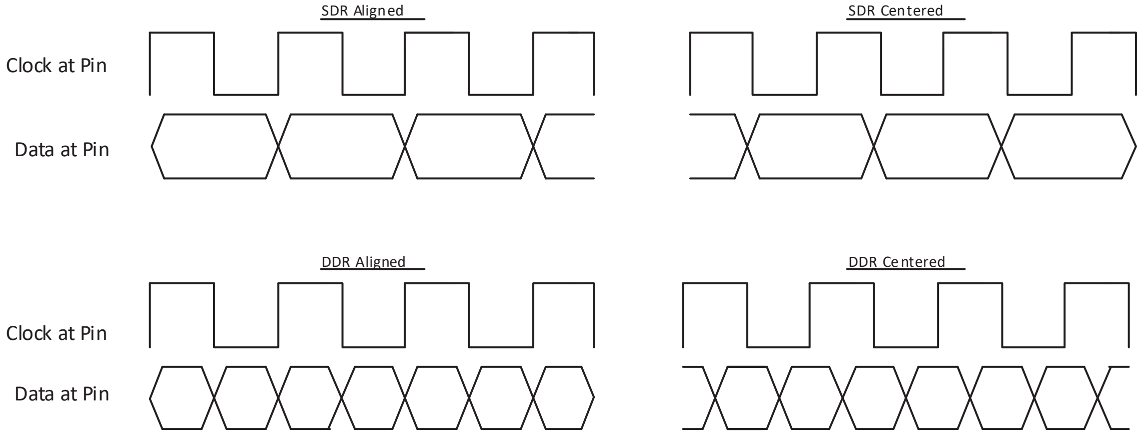


**Figure 2.1. External Interface Definition**

The interfaces described are referenced as centered or aligned interfaces. An aligned interface is needed to adjust the clock location to satisfy the capture flip-flop setup and hold times. A centered interface is needed to balance the clock and data delay to the first flip-flop to maintain the setup and hold already provided.

# 3. High-Speed I/O Interface Building Blocks

The LatticeECP3 device contains dedicated functions for building high-speed interfaces. This section describes when and how to use these functions. A complete description of the library elements including descriptions and attributes is provided at the end of this document.

Figure 3.1 shows a high-level diagram of the clocking resources available in the "E" and "EA" devices for building high-speed I/O interfaces.

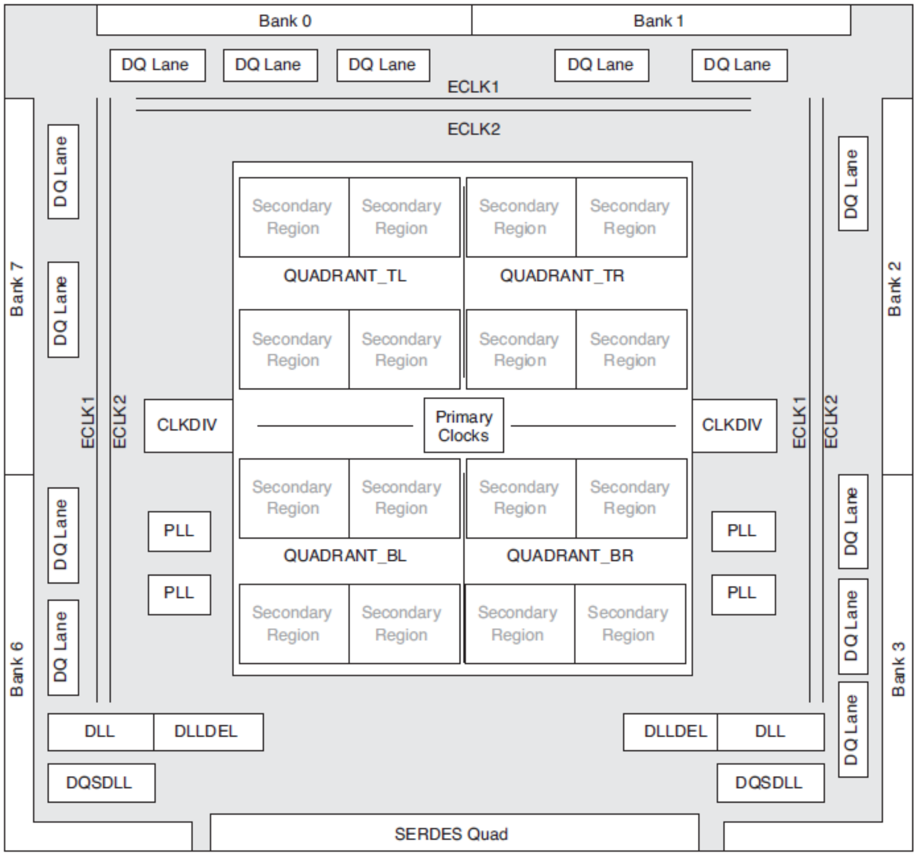


**Figure 3.1. LatticeECP3 Clocking Diagram (LatticeECP3-35 Shown)**

A complete description and of the LatticeECP3 clocking resources and clocking routing restrictions is available in LatticeECP3 sysCLOCK™ PLL/DLL Design and Usage Guide (FPGA-TN-02191).

Below is a brief description for each of the major elements used for building various high-speed interfaces. The DDR Software Primitives and Attributes section describes the library elements for these components.

## 3.1. ECLK

Edge clocks are high-speed, low-skew I/O dedicated clocks. They are arranged in groups of two on the left, right, and top sides of the device.

## 3.2. SCLK

SCLK refers to the system clock of the design. SCLK can use either primary or secondary clocks.

## 3.3. DQS Lane

A DQS Lane borrows its name from memory interfaces, but can be used for general-purpose high-speed interfaces. Each DQS Lane provides a clock (DQS) and up to 10 data bits on that clock. The number of DQS Lanes on the device is different for each device size. LatticeECP3 devices support DQS signals on the top, left and right sides of the device.

## 3.4. PLL

The PLL provides frequency synthesis as well as static and dynamic phase adjustment. Three output ports are provided: CLKOP, CLKOS and CLKOK. CLKOK provides a dedicated divide-by-2 function for use with the I/O logic gearing. The number of PLLs available on the device varies by device size from 2 to 10 PLLs per device.

## 3.5. DLL

The general-purpose DLL provides clock injection delay removal as well as 90° delay compensation when used with the DLLDEL element. There are two DLLs, one on each side of all LatticeECP3 devices.

## 3.6. DQSDLL

The DQSDLL is a dedicated DLL for creating a 90° clock delay. There are two DQSDLLs provided on all LatticeECP3 devices. There is one DQSDLL on each side of the device.

## 3.7. Input DDR (IDDR)

The input DDR function can be used in either x1 or x2 gearing modes. The x1 mode is supported by the IDDRXD element. This library element inputs a single DDR data input and clock (DQS, ECLK or SCLK) and provides a 2-bit wide data synchronized to the SCLK (system clock) to the FPGA fabric.

In the x2 mode, the IDDRX2D element is used. This element is useful for interfaces with greater than a 200MHz clock. It supports a 4-bit wide interface to the FPGA fabric. The clock input to the IDDRX2D is the high-speed ECLK. The SCLK clock is divided down to half of the ECLK input clock.

## 3.8. Output DDR (ODDR)

The output DDR function can also be supported in either x1 or x2 gearing modes. The output x1 DDR function is supported by the ODDRXD element. This library element provides a single DDR output and clock (SCLK) and accepts 2-bit wide data from the FPGA fabric.

The ODDR2XD element is used in the x2 mode and supports a 4-bit wide interface to the FPGA fabric. This element is useful for interfaces with greater than a 200MHz clock. The SCLK clock is divided down to half of the ECLK.

## 3.9. CLKDIV

The CLKDIV element is used to divide the high-speed ECLK by 2 to generate the SCLK when using x2 input or output gearing modes.

## 3.10. DELAY

There are three types of input data available. DELAYC provides a fixed value of delay to compensate for clock injection delay. DELAYC is used by default when configuring the interface in the software. The software configures the DELAYC with delay values based on the interface used. DELAYB provides dynamic or user-defined delay. DELAYC is also used for SDR interfaces where it provides clock injection delay.

## 3.11. ECLK/SCLK versus DQS Lanes

ECLK and SCLK span the entire side of the device and is useful for creating wide input bus interfaces. The DQS Lanes cover only 10 bits of data width and work well for narrow input bus interfaces. Each DQS Lane is serviced by a DQSBUF to control clock access and delay. The DQS Lane is supported by the DQSDLL for 90° clock delay. There is only one DQSDLL per side of the device, but this DQSDLL can be used for all of the DQS Lanes on the side. If several narrow input bus interfaces are required it is best to use the DQS Lanes instead of the ECLK or SCLK.

Due to architectural difference between “E” and “EA” devices, wider buses are supported using ECLK in “E” devices and SCLK in “EA” devices in x1 mode.

# 4. Building Generic High-Speed Interfaces

This section explains in detail how to build high-speed interfaces using the building blocks described above. The ispLEVER IPexpress tool builds these interfaces based on external interface requirements.

## 4.1. Types of High-Speed DDR Interfaces

This section describes the different types of high-speed DDR interfaces available in the LatticeECP3 device. Table 4.1 lists these interfaces. A description of each interface in the table is provided below the table.

**Table 4.1. Generic High-Speed I/O DDR Interfaces**

| Mode | Interface Name | Description | LatticeECP3 Device Support |
|---|---|---|---|
| RX SDR | GIREG_RX.SCLK | SDR Input register using SCLK | E, EA |
| RX DDRX1 Aligned | GDDRX1_RX.SCLK.Aligned/ GDDRX1_RX.SCLK.PLL.Aligned | DDR x1 Input using SCLK. Data is edge-to-edge with incoming clock. | EA |
| RX DDRX1 Aligned | GDDRX1_RX.DQS.Aligned | DDR x1 Input using DQS. Data is edge-to-edge with incoming clock. | E, EA |
| RX DDRX1 Aligned | GDDRX1_RX.ECLK.Aligned | DDR x1 Input using ECLK. Data is edge-to-edge with incoming clock. | E |
| RX DDRX2 Aligned | GDDRX2_RX.ECLK.Aligned/ GDDRX2_RX.ECLK.Aligned (no CLKDIV) | DDR x2 Input using ECLK. Data is edge-to-edge with incoming clock. | E, EA |
| RX DDRX2 Aligned | GDDRX2_RX.DQS.Aligned | DDR x2 Input using DQS. Data is edge-to-edge with incoming clock. | E, EA |
| RX DDRX1 Centered | GDDRX1_RX.DQS.Centered | DDR x1 Input using DQS. Clock is already centered in data window. | E, EA |
| RX DDRX1 Centered | GDDRX1_RX.ECLK.Centered | DDR x1 Input using ECLK. Clock is already centered in data window. | E |
| RX DDRX2 Centered | GDDRX2_RX.ECLK.Centered | DDR x2 Input using ECLK. Clock is already centered in data window. | E, EA |
| RX DDRX2 Centered | GDDRX2_RX.DQS.Centered | DDR x2 Input using DQS. Clock is already centered in data window. | E, EA |
| RX DDRX2 Dynamic | GDDRX2_RX.ECLK.Dynamic | DDR x2 Input with Dynamic Alignment using ECLK. | EA |
| RX DDRX2 Dynamic | GDDRX2_RX.DQS.Dynamic | DDR x2 Input with Dynamic Alignment using DQS. | EA |
| RX DDRX2 Dynamic | GDDRX2_RX.PLL.Dynamic | DDR x2 Input with Dynamic Alignment using ECLK. | EA |
| TX SDR | GOREG_TX.SCLK | SDR Output using SCLK. Clock is forwarded through ODDR. | E, EA |
| TX DDRX1 Centered | GDDRX1_TX.SCLK.Centered | DDR x1 Output using SCLK. Clock is centered using PLL with different SCLK. | E, EA |
| TX DDRX1 Centered | GDDRX1_TX.DQS.Centered | DDR x1 Output using DQS. Clock is centered using DQSDLL and ODDRDQS. | E, EA |
| TX DDRX1 Aligned | GDDRX1_TX.SCLK.Aligned | DDR x1 Output using SCLK. Data is edge-on- edge using same clock through ODDR. | E, EA |
| TX DDRX2 Aligned | GDDRX2_TX.Aligned | DDR x2 Output that is edge-on-edge. | EA |
| TX DDRX2 Centered | GDDRX2_TX.DQSDLL.Centered | DDR x2 Output that is pre-centered using DQS-DLL | EA |
| TX DDRX2 Centered | GDDRX2_TX.PLL.Centered | DDR x2 Output that is pre-centered using PLL | EA |

The following describes the naming conventions used for each of the interfaces listed in Table 4.1.

- G – Generic
- IREG – SDR input I/O register
- OREG – SDR output I/O register
- DDRX1 – DDR x1 I/O register
- DDRX2 – DDR x2 I/O register
- _RX – Receive interface
- _TX – Transmit interface
- .ECLK – Uses ECLK (edge clock) clocking resource
- .SCLK – Uses SCLK (primary clock) clocking resource
- .DQS – Uses DQS clocking resource
- .Centered – Clock is centered to the data when coming into the device
- .Aligned– Clock is aligned edge-on-edge to the data when coming into the device

### 4.1.1. Receive Interfaces

This section lists the receive interfaces can be implemented.

#### 4.1.1.1. Single Date Rate Interface (GIREG_RX.SCLK)

This interface is used when a simple input register is required for the design. The clock input to the input register can be optionally inverted if required. These interfaces always use SCLK.

#### 4.1.1.2. Input DDR 1x Interfaces

Input DDR 1x interfaces are used for DDR interfaces running at or below 200 MHz. The 1x interfaces can be further split into aligned and centered interfaces depending on the incoming clock-to-data relationship. In addition, for each of these interfaces the incoming data is delayed using the DELAYC element to compensate for the clock injection time.

**Aligned Interfaces**

These interfaces are used when the data and clock are aligned edge-on-edge when input to the device. The clock on the aligned interfaces is phase-shifted 90° using the on-chip DLL or DQSLL on each side of the device. These interfaces are further split into the following interfaces.

- Input DDR 1x Aligned Interface using SCLK (GDDRX1_RX.SCLK.Aligned/ GDDRX1_RX.SCLK.PLL.Aligned)
  This interface is used on "EA" devices when the clock and data are aligned edge-on-edge. The clock is shifted 90° using a DLL or PLL before it is used in the IDDRX1 element. The SCLK is used in this case to clock the IDDRX1 element on the "EA" device.
- Input DDR 1x Aligned Interface using ECLK (GDDRX1_RX.ECLK.Aligned)
  This interface is used on "E" devices when the clock and data are aligned edge-on-edge when coming into the device. ECLK is used to clock the IDDRX1 element on the "E" device. This clock is shifted 90° using DLL before it is routed to the ECLK.
- Input DDR 1x Aligned Interface using DQS (GDDRX1_RX.DQS.Aligned)
  This interface is used when the interface bus is narrow (<10 bits) and the clock and data are aligned edge-on-edge. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input. The 90° shift for the clock is generated using the DQS- DLL and the clock is delayed in the DQSBUF element. This delayed clock is used to clock the IDDRX1 element. This interface then uses SCLK to clock the data from the interface to the FPGA logic.

**Centered Interfaces**

These interfaces are used when the data and clock are centered when input to the device. Since the clock is centered to the data, it is not required to be phase-shifted for these interfaces. These interfaces are further split into the following interfaces.

- Input DDR 1x Centered Interface Using ECLK (GDDRX1_RX.ECLK.Centered)
  This interface is used on "E" devices when the clock and data are centered coming into the device. The ECLK is used to clock the input DDR element on the "E" device.
- Input DDR 1x Centered Interface Using SCLK (GDDRX1_RX.SCLK.Centered)

This interface is used on “EA” devices when the clock and data are centered coming into the device. The SCLK is used to clock the input DDR element on the “EA” device.

- Input DDR 1x Centered Interface using DQS (GDDRX1_RX.DQS.Centered)
  This interface is used when the interface bus is narrow (<10 bits) and the clock and data are centered. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input. DQSDLL is held in reset for this case as no clock shift is required. This interface then uses SCLK to clock the data from the interface to the FPGA logic.

### 4.1.1.3. Input DDR 2x Interfaces

Input DDR 2x interfaces are used for DDR interfaces running higher than 200 MHz. The 2x interfaces can be further split into aligned and centered interfaces depending on the incoming clock-to-data relationship. In addition, for each of these interfaces the incoming data is delayed using DELAYC element to compensate for the clock injection time.

**Aligned Interfaces**

These interfaces are used when the data and clock are aligned edge-on-edge when input to the device. The clock on the aligned interfaces is phase shifted 90° using the on-chip DLL or DQSLL on each side of the device. These interfaces are further split into the following interfaces.

- Input DDR 2x Aligned Interface using ECLK (GDDRX2_RX.ECLK.Aligned/ GDDRX2_RX.ECLK.Aligned (No CLKDIV))
  This interface is used when the clock and data are aligned edge-on-edge. The incoming clock is shifted 90° using the DLL. The output of the DLL is routed to the ECLK which is used to clock the IDDRX2 element.The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module or in the DLL module.
- Input DDR 2x Aligned Interface using DQS (GDDRX2_RX.DQS.Aligned)
  This interface is used when the interface bus is narrow (<10 bits) and the clock and data are aligned edge-on-edge. In this case, the DQS Lanes are used for each interface and DQS pin is used for the clock input. The 90° shift for clock is generated using the DQSDLL and the clock is delayed in the DQSBUF element. This delayed clock is used to clock the IDDRX2 element. The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module.

**Centered Interfaces**

These interfaces are used when the data and clock are centered when input to the device. Since the clock is centered to the data, it does not required to be phase-shifted for these interfaces. These interfaces are further split into the following interfaces.

- Input DDR 2x Centered Interface using ECLK (GDDRX2_RX.ECLK.Centered)
  This interface is used when clock and data are centered coming into the device. The clock is connected directly to the ECLK which is used to clock the IDDRX2 element. The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module.
- Input DDR 1x Centered Interface using DQS (GDDRX2_RX.DQS.Centered)
  This interface is used when the interface bus is narrow (<10 bits) and the clock and data are centered. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input going to the DQSBUF module. DQSDLL is held in reset for this case as no clock shift is required. The clock output of the DQSBUF is used to clock the IDDRX2 element. A PLL is used to generate the SCLK for this interface.
- Dynamic Interfaces (“EA” Devices Only)
  The data delay input on the data input of the input DDR 2x centered interfaces can optionally be controlled dynamically by the user logic using the DELAYB element. For dynamic control of the clock or data delay, one of following dynamic interfaces can be used. The dynamic interfaces are only available on “EA” devices.
- Input DDR 2x Centered Interface Using ECLK and Dynamic Delay (GDDRX2_RX.ECLK.Dynamic)
  This interface is similar to the GDDRX2_RX.ECLK.Centered interface described above but the input data delay is controlled by the user with the DELAYB element.
- Input DDR 2x Centered Interface Using DQS and Dynamic Delay (GDDRX2_RX.DQS.Dynamic)
  This interface is similar to the GDDRX2_RX.DQS.Centered interface described above but the input data delay is controlled by the user with the DELAYB element.
- Input DDR 2x Centered Interface Using PLL and Dynamic Delay (GDDRX2_RX.PLL.Dynamic)
  In this interface the PLL is used to dynamically shift the clock to adjust to the correct position to the data. This interface generates a bus-based delay for the interface.

**7:1 LVDS Interface**

This interface should be used when implementing a 7:1 LVDS interface. The 7:1 LVDS interface requires that the input clock is multiplied 3.5x before input to the IDDRX2 element to demux the data. Refer to 7:1 LVDS Video Interface Reference Design (RD1030) for further information.

### 4.1.2. Transmit Interfaces

This section lists the transmit interfaces can be implemented.

#### 4.1.2.1. Single Date Rate Interface (GOREG_TX.SCLK)

This interface is used for a SDR data output implementation with tight specifications on clock out to data out skew. This interface uses a simple output flip-flop for the data but forwards the clock using an ODDRX register. The same clock is used for both data and clock generation.

#### 4.1.2.2. Output DDR 1x Interfaces

Output DDR 1x interfaces are used for DDR interfaces running at or below 200 MHz. The 1x interfaces can further be spilt into aligned and centered interfaces depending on the relationship between the forwarded clock and data. The following are the different 1x interfaces.

**Aligned Interfaces**

These interfaces are used to provide data and clocks that are aligned edge-on-edge when leaving the device. These interfaces are further split into the following.

- Output DDR 1x Aligned Interface Using SCLK (GDDRX1_TX.SCLK.Aligned)
  This interface uses SCLK to generate clock and data that are aligned edge on edge

**Centered Interfaces**

These interfaces are used to provide data and clocks that are centered when leaving the device. The clock output is phase-shifted 90° using the on-chip PLL so that it can be centered to the data. These interfaces are further split as follows.

- Output DDR 1x Centered Interface Using SCLK (GDDRX1_TX.SCLK.Centered)
  In this case, the SCLK is used to generate the data and clock output. SCLK used to generate the clock is shifted 90° using a PLL so that it can be pre-centered to the data output.
- Output DDR 1x Centered Interface Using DQS (GDDRX1_TX.DQS.Centered)
  This interface is used when implementing interfaces that are <10 bits wide. The DQS Lanes are for data and clock assignments. The clock is pre-centered to the data using the DQSLL and the ODDRDQSA blocks.

#### 4.1.2.3. Output DDR 2x Interfaces

Output DDR 2x interfaces are used for DDR interfaces running higher than 200 MHz. The 2x interfaces can further be spilt into aligned and centered interfaces depending on the relationship between the forwarded clock and data. Output DDR 2x interfaces are supported only on “EA” devices. The following are the different 2x interfaces.

**Aligned Interfaces**

These interfaces are used to provide data and clocks that are aligned edge-on-edge when leaving the device. These interfaces are further split into the following.

- Output DDR 2x Aligned Interface (GDDRX2_TX.Aligned)
  ODDRX2 is used generate the clock and data that are aligned in phase.

### Centered Interfaces

These interfaces are used to provide data and clocks that are centered when leaving the device. The clock output is phase shifted 90° so that it can be centered to the data. These interfaces are further split into the following.

- Output DDR 2x Centered Interface using DQSDLL (GDDRX2_TX.DQSDLL.Centered)
  This interface is primarily used for interfaces that are <10 bits wide. In this case, a DQSDLL is used to generate the 90° shift required to pre-center the clock to the data output.
- Output DDR 2x Centered Interface using PLL (GDDRX2_TX.PLL.Centered)
  This interface is primarily used for wider interfaces. Here, a PLL is used to generate the 90° shift required to pre-center the clock to the data output.

### 7:1 LVDS Interface

This interface is used when implementing a 7:1 LVDS interface. The 7:1 LVDS interface requires that the clock output be multiplied 3.5x before going to the ODDRX2 module. Refer to 7:1 LVDS Video Interface Reference Design (RD1030) for further information.

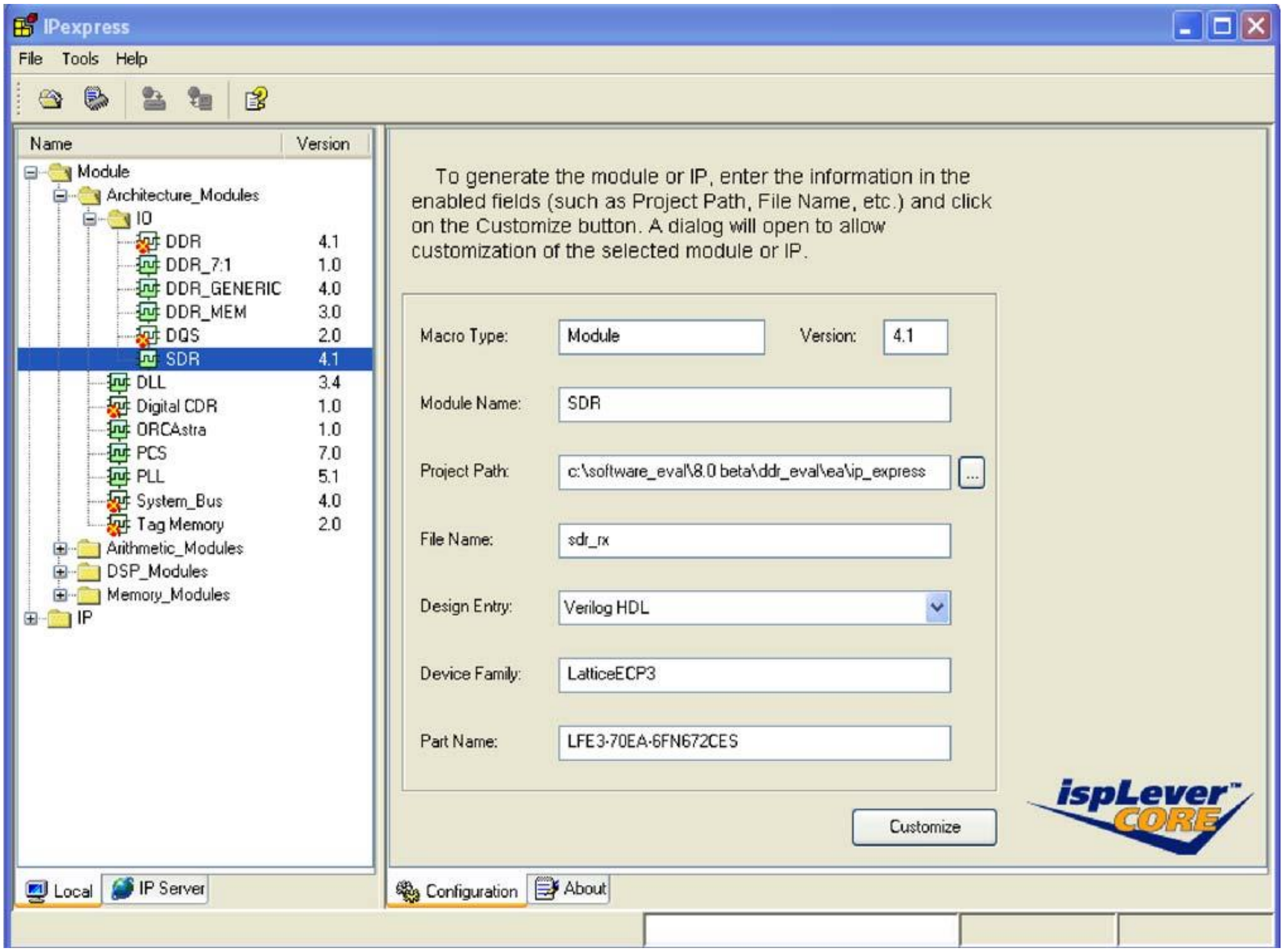
# 5. Using IPexpress to Build High-Speed DDR Interfaces

The IPexpress tool is used to configure and generate all the high-speed interfaces described above. IPexpress generates a complete HDL module including clocking requirements for each of the interfaces.

This section assumes that ispLEVER 8.0 is used for generation of the interfaces. If you are using ispLEVER 7.2 SP2, see Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2. If you are using Lattice Diamond® design software, see Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond.

For a detailed block diagram of each interface generated by IPexpress, see the High-Speed DDR Interface Details section.

IPexpress can be opened from the Tools menu in Project Navigator. All DDR modules are located under Architecture Modules -> IO. This section covers SDR and DDR_GENERIC. DDR_MEM is discussed in the Implementing DDR/DDR2/DDR3 Memory Interfaces section.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.1 |
| DDR_7:1 | 1.0 |
| DDR_GENERIC | 4.0 |
| DDR_MEM | 3.0 |
| DQS | 2.0 |
| SDR | 4.1 |
| DLL | 3.4 |
| Digital CDR | 1.0 |
| ORCAstra | 1.0 |
| PCS | 7.0 |
| PLL | 5.1 |
| System_Bus | 4.0 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| DSP_Modules | |
| Memory_Modules | |
| IP | |

To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: Module
Version: 4.1
Module Name: SDR
Project Path: c:\software_eval\8.0 beta\ddr_eval\ea\ip_express
File Name: sdr_rx
Design Entry: Verilog HDL
Device Family: LatticeECP3
Part Name: LFE3-70EA-6FN672CES

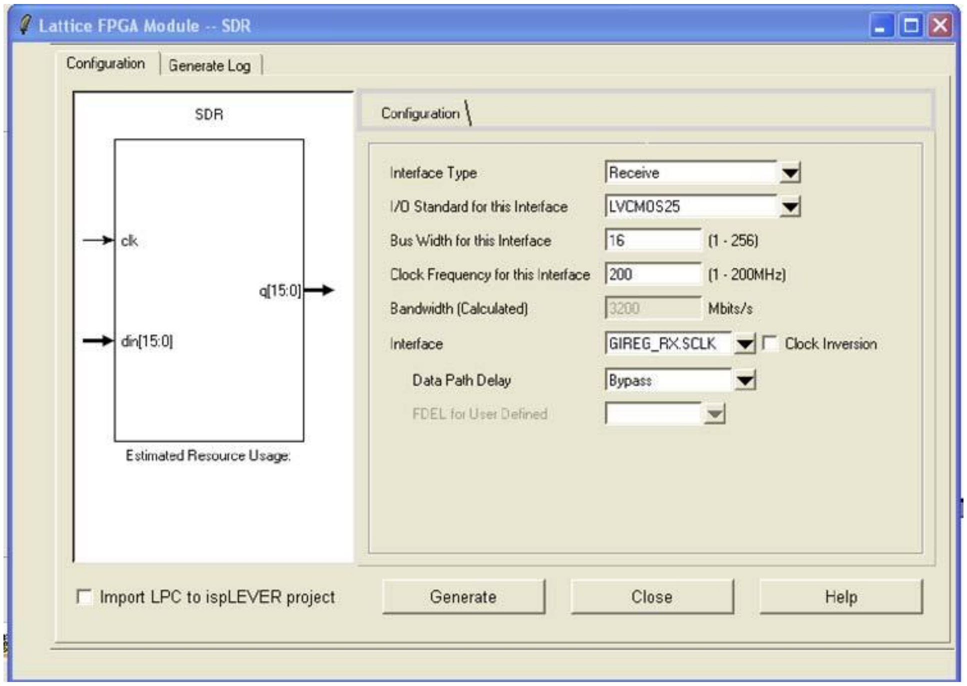
**Figure 5.1. IPexpress Main Window**

Select the type of interface you would like to build and enter the name of the module. Figure 5.1 shows the type of interface selected as "SDR" and module name entered. Each module can then be configured by clicking the Customize button.

## 5.1. Building SDR Modules

Choose interface type SDR, enter module name and click Customize to open the configuration tab.

Figure 5.2 shows the Configuration Tab for the SDR module in IPexpress. Table 5.1 lists the various configurations options available for SDR modules.


Lattice FPGA Module -- SDR
Configuration
Generate Log
SDR
clk
q[15:0]
din[15:0]
Estimated Resource Usage:
Configuration
□ Import LPC to ispLEVER project
Generate
Close
Help

Interface Type: Receive
I/O Standard for this Interface: LVCMOS25
Bus Width for this Interface: 16 (1 - 256)
Clock Frequency for this Interface: 200 (1 - 200MHz)
Bandwidth (Calculated): 3200 Mbits/s
Interface: GIREG_RX.SCLK □ Clock Inversion
Data Path Delay: Bypass
FDEL for User Defined:

**Figure 5.2. IPexpress Main Window**

**Table 5.1. SDR Configuration Parameters**

| User Interface Option | Description | Values | Default |
|---|---|---|---|
| Interface Type | Type of interface (transmit or receive) | Transmit, Receive | Receive |
| I/O Standard for this Interface | I/O standard to be used for the inter-face. | Transmit and Receive:<br>LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTL33<br>Transmit only:<br>RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE | LVCMOS25 |
| Bus Width for this Interface | Bus size for the interface. | 1 - 256 | 16 |
| Clock Frequency for this Inter- face | Speed at which the interface runs. | 1 - 200 | 200 |
| Bandwidth (Calculated) | Calculated from the clock frequency entered. | (Calculated) | (Calculated) |
| Interface | Interface selected based on previous entries. | Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default) | GIREG_RX.S CLK |

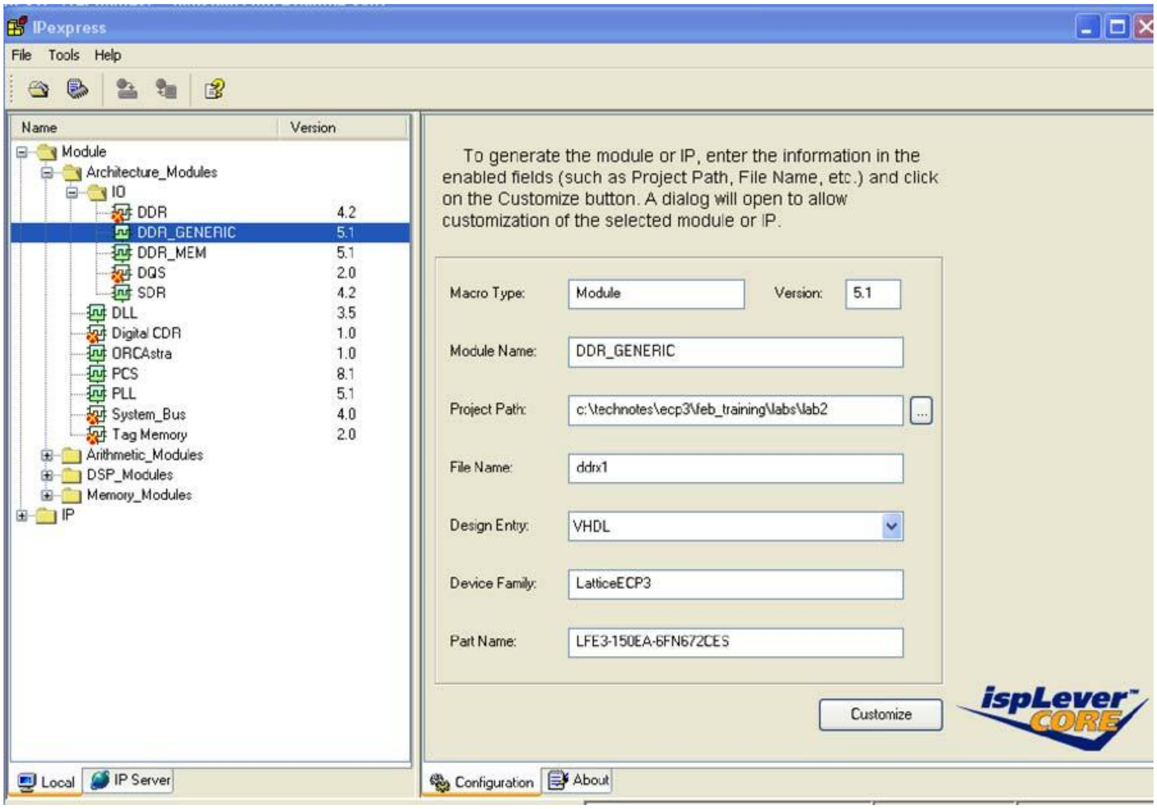
| User Interface Option | Description | Values | Default |
|---|---|---|---|
| Clock Inversion | Option to invert the clock input to the I/O register. | DISABLED, ENABLED | DISABLED |
| Data Path Delay | Data input can be optionally delayed using the DELAY block. | Bypass, Dynamic[1], User-Defined | Bypass |
| FDEL for User-Defined | If Delay type selected above is user-defined, delay values can be entered with this parameter. | 0 to 15[2] | 0 |

**Notes:**

1. When Delay type Dynamic is selected, the 16-step delay values must be controlled from your design.
2. A FDEL is a fine-delay value that is additive. The delay value for a FDEL can be found in the LatticeECP3 Family Data Sheet.

## 5.2. Building DDR Generic Modules

Choose interface type DDR_GENERIC, enter module name and click Customize to open the configuration tab.



To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: Module Version: 5.1

Module Name: DDR_GENERIC

Project Path: c:\technotes\ecp3\feb_training\labs\lab2

File Name: ddrx1

Design Entry: VHDL

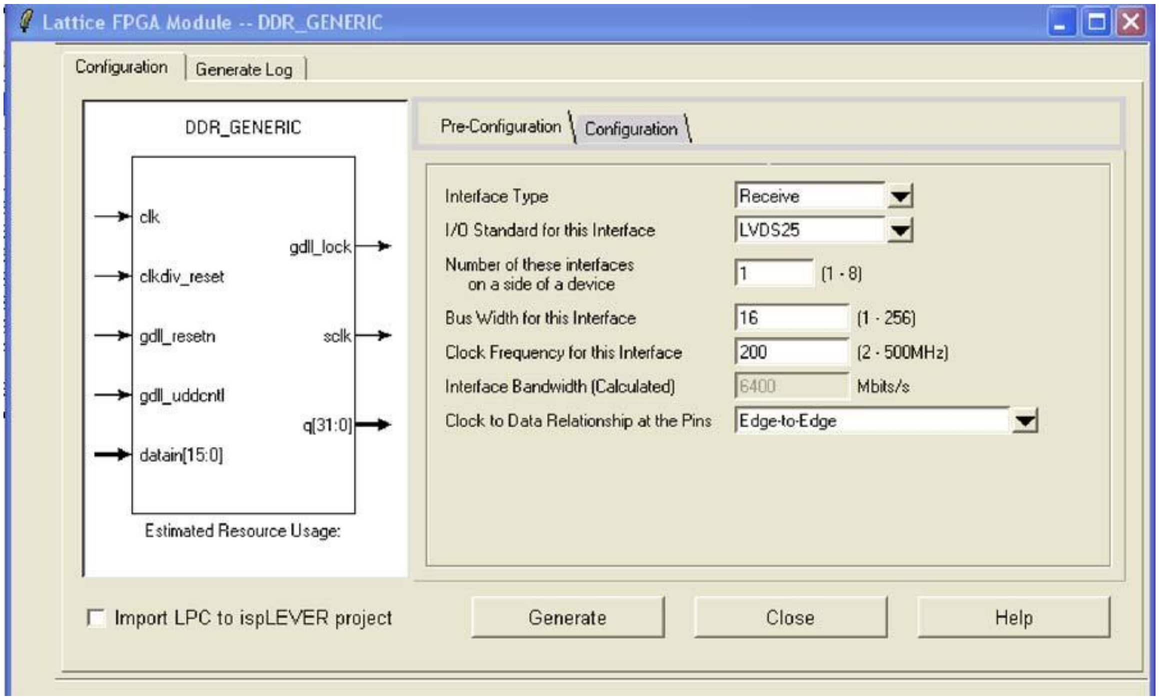
Device Family: LatticeECP3

Part Name: LFE3-150EA-6FN672CES

**Figure 5.3. "DDR_Generic" Selected in Main IPexpress Window**

When clicking Customize, DDR modules have a Pre-Configuration Tab and a Configuration" Tab. The Pre-Configuration Tab allows you to enter information about the type of interface to be built. Based on the entries in the Pre-configuration Tab, the Configuration Tab is populated with the best interface selection. You can also, if necessary, override the selection made for the interface in the Configuration Tab and customize the interface based on design requirements.

Figure 5.4 shows the Pre-Configuration Tab for DDR generic interfaces. Table 5.2 lists the various parameters in the tab.



Interface Type: Receive

I/O Standard for this Interface: LVDS25

Number of these interfaces on a side of a device: 1 (1 - 8)

Bus Width for this Interface: 16 (1 - 256)

Clock Frequency for this Interface: 200 (2 - 500MHz)

Interface Bandwidth (Calculated): 6400 Mbits/s

Clock to Data Relationship at the Pins: Edge-to-Edge

☐ Import LPC to ispLEVER project

Figure 5.4. DDR Generic Pre-Configuration Tab

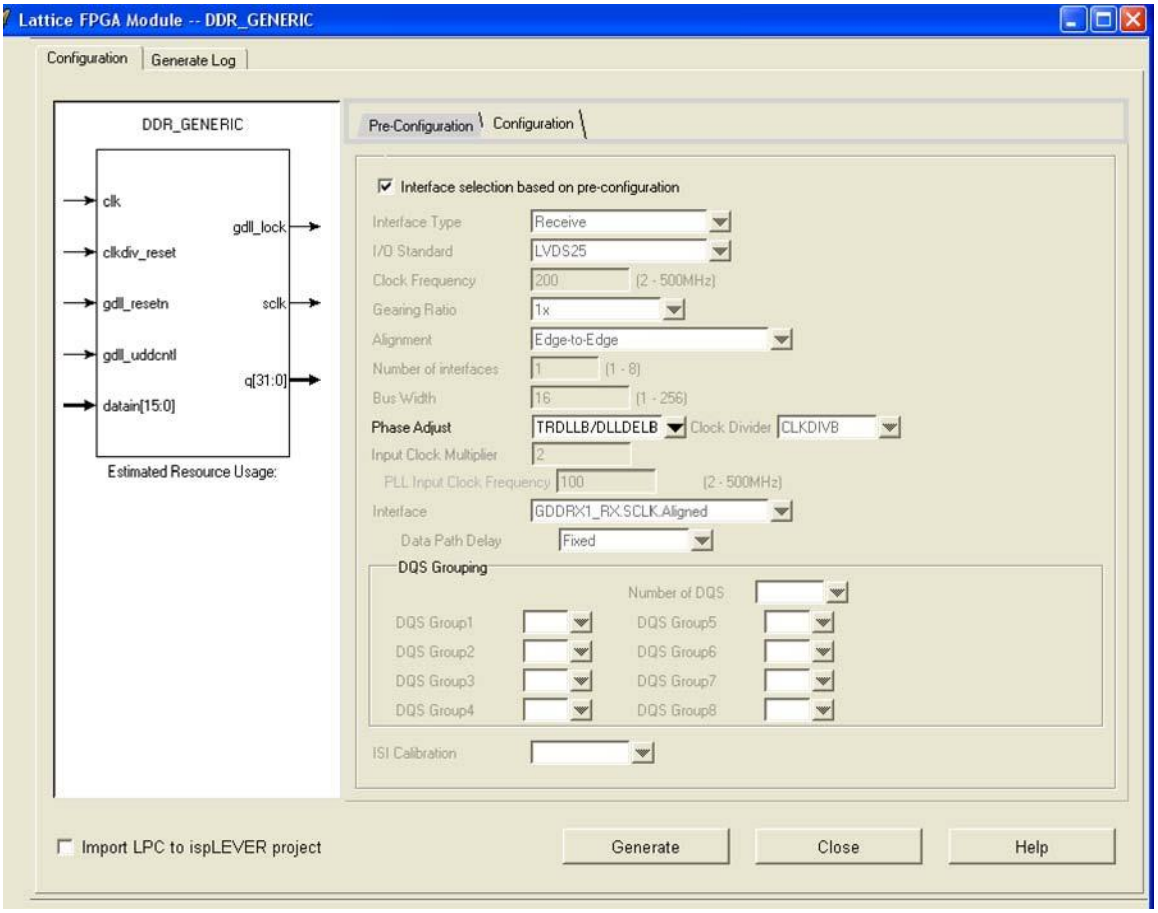
Table 5.2. Pre-Configuration Tab Settings

| User Interface Option | Description | Values |
|---|---|---|
| Interface Type (Transmit or Receive) | Type of interface (Receive or Transmit) | Transmit, Receive |
| I/O Standard for this Interface | I/O Standard used for the interface | Transmit and Receive:<br>LVCMOS25,LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTL33<br><br>Transmit only:<br>RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE |
| Number of interfaces on a side of a device | Number of interfaces to be implemented per side. This is used primarily for nar- row bus width interfaces (<10). Other- wise it is recommended to leave this at 1. | 1 to 8 |
| Bus Width for this Interface | Bus width for each interface. If the num- ber of interfaces per side is >1 then the bus width per interface is limited to 10. If number of interfaces per side is >1 and if using differential I/O standards then bus width is limited to 5. | 1-256 |
| Clock Frequency for this Interface | Interface speed | 2 - 500 MHz |
| Interface Bandwidth (Calculated) | Bandwidth is calculated from the clock frequency. | Calculated |

| User Interface Option | Description | Values |
|---|---|---|
| Clock to Data Relationship at the Pins | Relationship between clock and data. | Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required[1], Dynamic Clock Phase Alignment Required |

**Note:** Dynamic Phase Alignment is only available for x2 interfaces (for example, when the clock frequency is higher than 200 MHz).

Based on the selections made in the Pre-Configuration Tab, the Configuration Tab is populated with the selections. Figure 5.5 shows the Configuration Tab for the selections made in the Pre-Configuration Tab.



☑ Interface selection based on pre-configuration

Interface Type: Receive
I/O Standard: LVDS25
Clock Frequency: 200 (2 - 500MHz)
Gearing Ratio: 1x
Alignment: Edge-to-Edge
Number of interfaces: 1 (1 - 8)
Bus Width: 16 (1 - 256)
Phase Adjust: TRDLLB/DLLDELB Clock Divider: CLKDIVB
Input Clock Multiplier: 2
PLL Input Clock Frequency: 100 (2 - 500MHz)
Interface: GDDRX1_RX.SCLK.Aligned
Data Path Delay: Fixed
Number of DQS:
DQS Group1:
DQS Group2:
DQS Group3:
DQS Group4:
DQS Group5:
DQS Group6:
DQS Group7:
DQS Group8:
ISI Calibration:

☐ Import LPC to ispLEVER project

**Figure 5.5 DDR Generic Configuration Tab**

The checkbox at the top of this tab indicates that the interface is selected based on entries in the Pre-Configuration Tab. You can choose to change these values by disabling this entry. Note that IPexpress chooses the most suitable interface based on selections made in the Pre-Configuration Tab.

Table 5.3 lists the various parameters in the Configuration Tab.

**Table 5.3. Configuration Tab Settings**

| User Interface Option | Description | Values | Default Value |
|---|---|---|---|
| Interface Selection Based on Pre-configuration | Indicates interface is selected based on selection made in the Pre-configuration tab. Disabling this checkbox allows you to make changes if needed. | ENABLED, DISABLED | ENABLED |
| Interface Type | Type of interface (receive or transmit) | Transmit, Receive | Receive |
| I/O Standard | I/O standard used for the interface | All the ones listed in the Pre-configuration tab | LVCMOS25 |
| Clock Frequency | Speed of the interface | 2 to 500 MHz | 200 MHz |

| User Interface Option | Description | Values | Default Value |
|---|---|---|---|
| Gearing Ratio | DDR register gearing ratio (1x or 2x) | 1x, 2x | 1x |
| Alignment | Clock to data alignment | Edge-to-Edge,<br>Centered,<br>Dynamic Data Phase Alignment Required,<br>Dynamic Clock Phase Alignment Required | Edge-to-Edge |
| Number of Interfaces | Number of interfaces to be implemented per side. This is primarily used for narrow bus width interfaces (<10), otherwise it is recommended to leave this at 1. | 1 to 8 | 1 |
| Bus Width | Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If the number of interfaces per side is >1 and if using differential I/O standards then the bus width is limited to 5. | 1 to 256 | 10 |
| Phase Adjust | Module used for phase shifting input clock. | TRDLLB/DLLDELB, PLL[1] | TRDLLB/DLLDELB |
| Clock Divider | Module used for generation of SCLK from ECLK. | CLKDIVB, TRDLLB[2] | CLKDIVB |
| Interface | Shows list of all valid high-speed interfaces for a given configuration. | See Table 5.4 for interfaces available for a given configuration. | GDDRX1_RX.SCLK. Aligned (EAdevices);<br>GDDRX1_RX.ECLK. Aligned (E devices) |
| Data Path Delay | Data input can be optionally delayed using the DELAY block. Value is selected based on Interface Type. | Bypass, Fixed, Dynamic[3] | Fixed |
| Number of DQS Groups | Enabled when a DQS interface is selected in the Interface selection. | 1 to 8 | |
| Number of DQ:<br>DQS Group1 to DQS Group8 | This option can be used to change the number of DQ assigned to each DQS lane. Each DQS lane can support up to 10 DQ. | 1 to 10 | |

**Notes:**

1. Only available when using GDDRX2_RX.ECLK.Aligned interface.
2. Only available when using GDDRX2_RX.SCLK Aligned interface.
3. When Dynamic Delay is selected, the 16-step delay values must be controlled from your design.

Table 5.4 shows how the interfaces are selected by IPexpress based on the selections made in the Pre-Configuration Tab.

**Table 5.4. IPexpress Interface Selection**

| Device Selected | Interface Type | Gearing Ratio[1] | Alignment | Number of Interfaces | Interface |
|---|---|---|---|---|---|
| EA | Receive | 1x | Edge-to-Edge | 1 | GDDRX1_RX.SCLK.Aligned |
| EA | Receive | 1x | Centered | 1 | GDDRX1_RX.SCLK.Centered |
| E | Receive | 1x | Edge-to-Edge | 1 | GDDRX1_RX.ECLK.Aligned |
| E | Receive | 1x | Centered | 1 | GDDRX1_RX.ECLK.Centered |
| E, EA | Receive | 1x | Edge-to-Edge | >1 | GDDRX1_RX.DQS.Aligned |
| E, EA | Receive | 1x | Centered | >1 | GDDRX1_RX.DQS.Centered |
| E, EA | Receive | 2x | Edge-to-Edge | 1 | GDDRX2_RX.ECLK.Aligned |
| E, EA | Receive | 2x | Centered | 1 | GDDRX2_RX.ECLK.Centered |
| E, EA | Receive | 2x | Edge-to-Edge | >1 | GDDRX2_RX.DQS.Aligned |
| E, EA | Receive | 2x | Centered | >1 | GDDRX2_RX.DQS.Centered |
| EA | Receive | 2x | Dynamic | 1 | GDDRX2_RX.ECLK.Dynamic (Default) |

| Device Selected | Interface Type | Gearing Ratio[1] | Alignment | Number of Interfaces | Interface |
|---|---|---|---|---|---|
| EA | | | | | GDDRX2_RX.DQS.Dynamic[2] |
| EA | | | | | GDDRX2_RX.PLL.Dynamic[2] |
| E, EA | Transmit | 1x | Centered | 1 | GDDRX1_TX.SCLK.Centered |
| E, EA | Transmit | 1x | Edge-to-Edge | 1 | GDDRX1_TX.SCLK.Aligned |
| E, EA | Transmit | 1x | Centered | >1 | GDDRX1_TX.DQS.Centered |
| EA | Transmit | 2x | Edge-to-Edge | 1 | GDDRX2_TX.Aligned |
| EA | Transmit | 2x | Centered | >1 | GDDRX2_TX.DQSDLL.Centered |
| EA | Transmit | 2x | Centered | 1 | GDDRX2_TX.PLL.Centered |

**Notes:**

1. Gearing Ratio of 1x is selected for clock frequencies less than 200MHz. Gearing ratio of 2x is selected for frequencies above 200 MHz.
2. These interfaces can only be selected in the Configuration Tab.

The implementation for several of the interfaces described above differs between the “E” and “EA” devices. Refer to the High-Speed DDR Interface Details section to see implementation details for “E” and “EA” devices.

The Data Delay setting for each interface is predetermined and cannot be changed by the user. You can only control Data Delay values when using a dynamic interface.

**Note:** Some modules generated by IPexpress have a SCLK and ECLK output port. If present, this port must be used to drive logic outside the interface driven by the same signal. In these modules, the input buffer for the clock is inside the IPexpress module and therefore cannot be used to drive other logic in the top level.

# 6. High-Speed DDR Interface Details

This section describes each of the generic high-speed interfaces in detail including the clocking to be used for each interface. For detailed information about the LatticeECP3 clocking structure, refer to LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide (FPGA-TN-02191). The various interface rules and preferences listed under each interface should be followed to build these interfaces successfully. Each of the interfaces for the "EA" devices has an ID label associated with the interface. The interface ID is set when generating the interface using IPexpress. This ID is entered using an attribute called IDDRAPPS or ODDRAPPS. The software uses this ID to set appropriate the data delay for the DELAYC element used in each of the interfaces. It is required that every interface on the "EA" devices use this attribute to achieve the correct timing results in the software Trace report. Refer to the Timing Analysis for High-Speed DDR Interfaces section for more information about the timing analysis on these interfaces. It is also necessary to follow the interface rules and preferences listed for each of the interface descriptions below for the interfaces to work as described.

In order to achieve higher speeds, the guidelines described in the Placement Guidelines for High-Speed DDR Interfaces section should be strictly followed.

All the interfaces described below are supported using ispLEVER 8.0 software. Some of these interfaces do not work in versions of ispLEVER prior to version 8.0. Refer to the Generic DDR Design Guidelines section to see all other design guidelines.

## 6.1. GIREG_RX.SCLK

Generic SDR Receive Interface using SCLK.

Device Support: "E" and "EA" devices

### Description

This is a generic interface for single data rate (SDR) data. An optional inverter can be used to center the clock for aligned inputs. A PLL or DLL can be used to remove the clock injection delay or adjust the setup and hold times. There are a limited number of DLLs in the architecture and these should be saved for high-speed interfaces when necessary. This interface can either be built using IPexpress or inferred during synthesis.

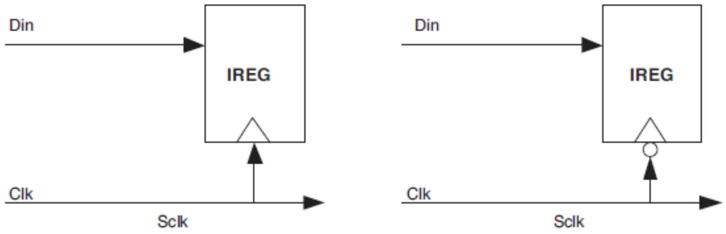


**Figure 6.1 GIREG_RX Interface ("E" and "EA" Devices)**

This interface also supports data delay on the data input. This delay value is set using a DELAYB or a DELAYC element. A DELAYC element provides a fixed delay to match the SCLK injection time.

DELAYB is used when you choose to update the delay dynamically or use user-defined static values. Figure 6.2 shows the DELAYB element connected to this interface.

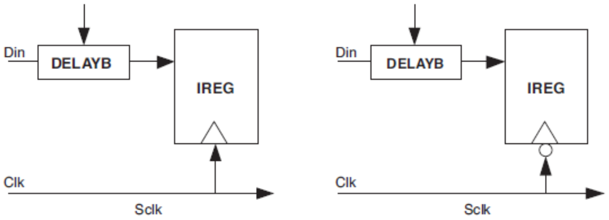


**Figure 6.2 GIREG_RX.SCLK with Delay Interface ("E" and "EA" Devices)**

**Interface Rules**

- The input clock must use a dedicated clock (PCLK) input pin.

## 6.2. GDDRX1_RX.ECLK.Aligned

Generic DDR Receive Interface using ECLK with Aligned External Interface Device Support: "E" devices only

**Description**

This DDR interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRXD. DLLDELB is used to delay the incoming clock by 90°. CLKDIV is used to generate a divide-by-1 clock that is connected to the SCLK. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

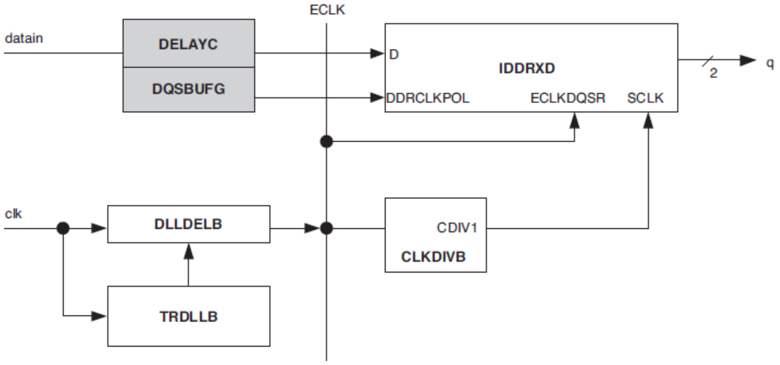


**Figure 6.3 GDDRX1_RX.ECLK.Aligned Interface ("E" Devices Only)**

**Interface Rules**

- The input clock must use a GPLLT_IN or GDLLT_IN pin. All data for the interface must all be on the same ECLK (same side).
- Since there is only one DLLDELB and one CLKDIVB per left and right sides of the device, you can implement only one such interface per side, or two total on the device.
- The clock net connected to SCLK should be on a primary clock net. You must assign the "USE PRIMARY NET<clk>" preference to assign the SCLK clock net to a primary clock.
- The pin assignments for data and clocks require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.

## 6.3. GDDRX1_RX.ECLK.Centered

Generic DDR Receive Interface using ECLK with Centered External Interface.

Device Support: "E" devices only

**Description**

This DDR interface uses the ECLK and DELAYC to match clock and data delay at the IDDRXD. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

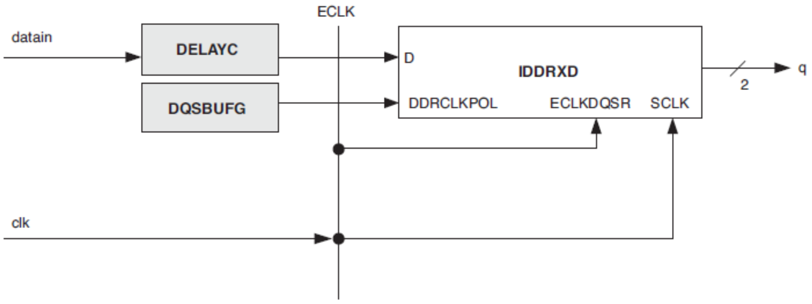


**Figure 6.4 GDDRX1_RX.ECLK.Centered Interface (“E” Devices Only)**

**Interface Rules**

- The input clock port must use a dedicated clock (PCLK) input pin. All data for the interface must be on the same ECLK (same side).
- The clock net connected to SCLK must be routed on a primary clock net. It is your responsibility to assign the SCLK clock net to a primary clock tree using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.

## 6.4. GDDRX1_RX.SCLK.Aligned

Generic DDR Receive Interface using SCLK with Aligned External Interface

Device Support: “EA” devices only

**Description**

This DDR interface uses the SCLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRXD1. This interface is useful for large data buses (>10 bits). A DELAYC is used to adjust data delay for the SCLK clock injection time. CLKDIV in divide-by-1 setting is used to generate the SCLK.

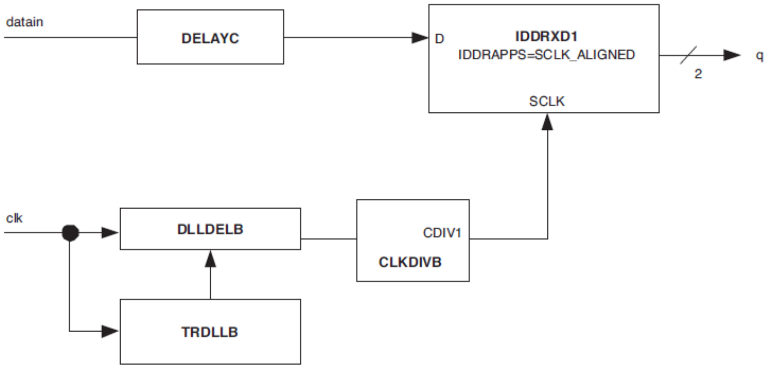


**Figure 6.5 GDDRX1_RX.SCLK.Aligned Interface (“EA” Devices Only)**

**Interface Rules**

- The clock input must use a dedicated GPLLT_IN or GDLLT_IN clock input pin (two pins per side). A dedicated PCLK pin on the top side can connect directly to the TRDLLB as well.
- There is only one DLLDELB per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- CLKDIV is required to generate SCLK.
- The clock net connected to SCLK should be on a primary clock net. You must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

## 6.5. GDDRX1_RX.SCLK.PLL.Aligned

Generic DDR Receive Interface using SCLK with Aligned External Interface

Device Support: “EA” devices only

### Description

This DDR interface uses the SCLK and a PLL to provide a 90° clock shift to center the clock at the IDDRXD1. This interface is useful for large data buses (>10 bits). A DELAYC is used to adjust data delay for the SCLK clock injection time. CLKDIV in divide by 1 setting is used to generate the SCLK.

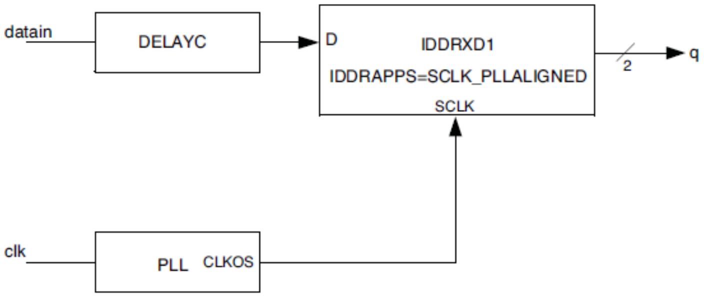


**Figure 6.6 GDDRX1_RX.SCLK.PLL.Aligned Interface (“EA” Devices Only)**

### Interface Rules

The clock input must use a dedicated GPLLT_IN clock input pin (two pins per side). CLKDIV is required to generate SCLK.

The clock net connected to SCLK should be on a primary clock net. You must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

## 6.6. GDDRX1_RX.SCLK.Centered

Generic DDR Receive Interface using SCLK with Centered External Interface

Device Support: “EA” devices only

### Description

This DDR interface uses the SCLK and DELAYC to match clock and data delay at the IDDRXD. This interface is useful for large data buses (>10 bits).

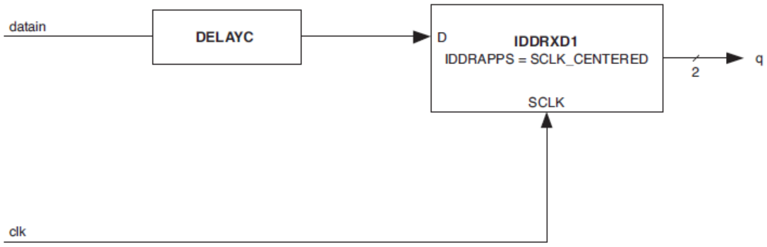


**Figure 6.7 GDDRX1_RX.SCLK.Centered Interface (“EA” Devices Only)**

### Interface Rules

- The clock input must use a dedicated clock (PCLK) input pin. The output of a PLL clock in bypass mode can be connected to the SCLK as well.
- The clock connected to SCLK should be on a primary clock net. You must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

## 6.7. GDDRX1_RX.DQS.Aligned

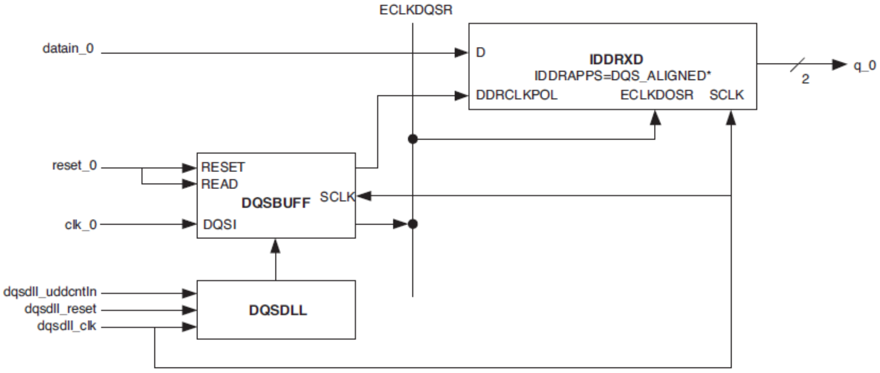
Generic DDR Receive Interface using DQS Lane with Aligned External Interface

Device Support: “E” and “EA” devices

**Description**

This DDR interface uses the DQS and the DQSDLL to provide a 90° clock shift to center the clock at the IDDRXD. This interface is uses a DQS lane and is useful for small data buses (< 11 bits). DQSDLL provide the 90° delay to the DQSBUFF which is used to delay the incoming clock.

A reference clock input “dqsdll_clk” running at the same frequency as DQS clock is required to be input to the DQSDLL to generate the delay.



* IDDRAPPS required for “EA” devices only.

**Figure 6.8 GDDRX1_RX.DQS.Aligned Interface (“E” and “EA” Devices)**

Any frequency-locked clock or the local clock from the input pin can be used for SCLK. The timing transfer between the ECLKDQSR and SCLK is handled in the hardware through the DDRCLKPOL.

**Interface Rules**

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The VREF1 for the selected DQS lane must be powered on the board.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section “DQSDLLB” on page 88 for the detailed requirements for the DQSDLL_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- “dqsdll_clk” and clock net connected to SCLK must use the primary clock tree. It is your responsibility to assign the “USE PRIMARY NET<clk>” preference on these nets to assign it to the primary clock.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.

## 6.8. GDDRX1_RX.DQS.Centered

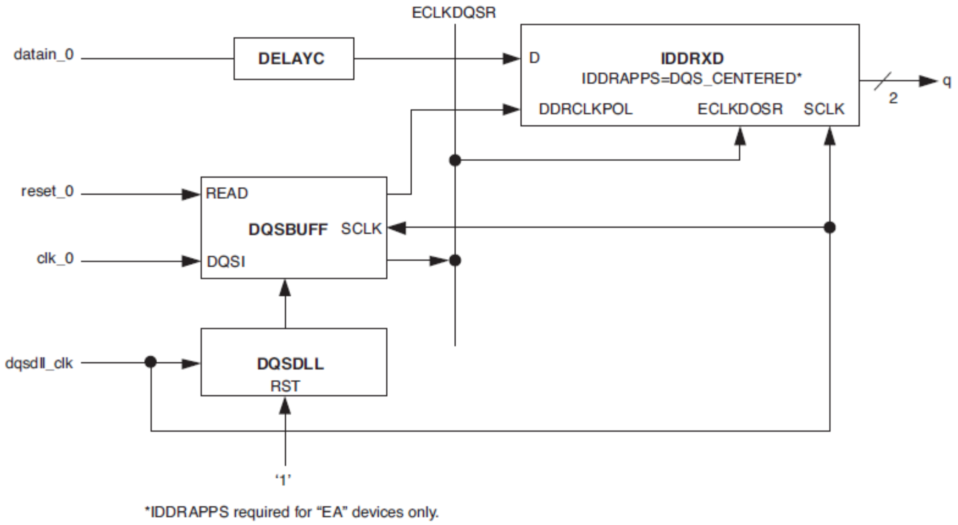
Generic DDR Receive Interface using DQS Lane with Centered External Interface

Device Support: "E" and "EA" devices

### Description

This DDR interface uses the DQS and DELAYC to match clock and data delay at the IDDRXD. This interface is useful for small data buses (<11 bits). Since a 90° shift is not required, the DQSDLL is held in reset for this interface.

You can use any frequency-locked clock for SCLK or the local clock from the input pin. The timing transfer between the ECLKDQSR and SCLK is handled in the hardware through the DDRCLKPOL



Notes:
1. The DELAYC is only applicable for "EA" devices. For "E" devices, change the DELAYC to DELAYB with a delay value of 7.
2. When retargeting from "E" to "EA" devices, manually update DELYAB to DELAYC, otherwise the software will error out.

**Figure 6.9 GDDRX1_RX.DQS.Centered Interface ("E" and "EA" Devices)**

### Interface Rules

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The VREF1 for the selected DQS lane must be powered on the board.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The clock net connected to SCLK must use the primary clock tree. It is your responsibility to assign "USE PRIMARY NET<clk>" preference on this SCLK clock net to assign it to the primary clock.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.

## 6.9. GDDRX2_RX.ECLK.Aligned

Generic DDR Receive Interface with x2 Gearing using ECLK with Aligned External Interface

Device Support: "E" and "EA" devices

### Description

This DDR x2 interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRX2D. DELAYC is used to delay data to match the ECLK injection delay. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

Note the difference in interface implementation between the "E" and "EA" devices. "E" devices require the use of a DQSBUFE which is not required on the "EA" devices.

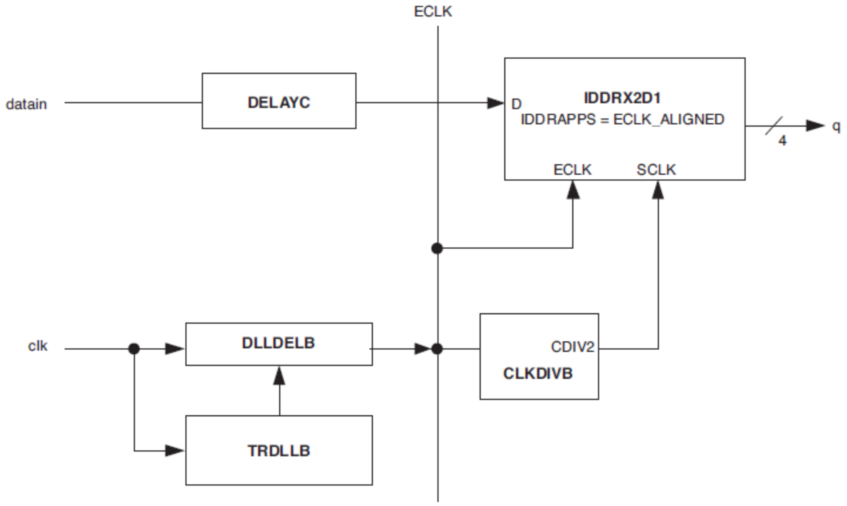


**Figure 6.10 GDDRX2_RX.ECLK.Aligned Interface ("EA" Devices)**

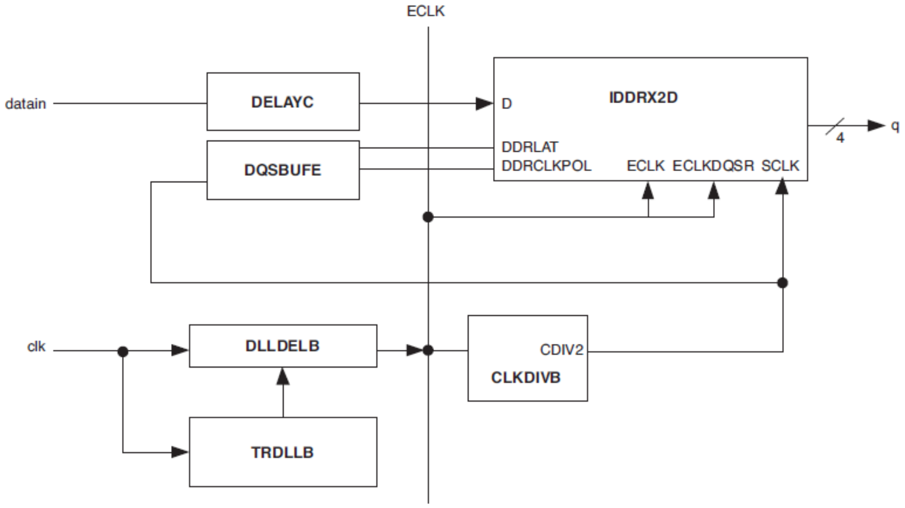


**Figure 6.11 GDDRX2_RX.ECLK.Aligned Interface ("E" Devices)**

**Interface Rules**

- It is recommended that a dedicated GDLL T_IN is used for the clock input. In "EA" devices, a GPLLT_IN or PCLK from the top side can also be used for the clock input.
- The clock net output of the CLKDIVB module is connected to SCLK and it must be routed on a primary clock. The "USE PRIMARY NET<clk>" preference must be used on this clock net as well.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The data/clock pin assignments for "E" devices require following the DQ-DQS group pinout guidelines. See the Generic DDR Design Guidelines for details. This is not required for "EA" devices.

## 6.10. GDDRX2_RX.ECLK.Aligned (No CLKDIV)

Generic DDR Receive Interface with x2 Gearing using ECLK with Aligned External Interface

Device Support: "EA" devices

### Description

This DDR x2 interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRX2D. DELAYC is used to delay data to match the ECLK injection delay. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

This interface uses x2 gearing with the IDDRX2D element. The CLKOS output of the TRDLLB can be set up to divide ECLK by 2 to generate the SCLK.

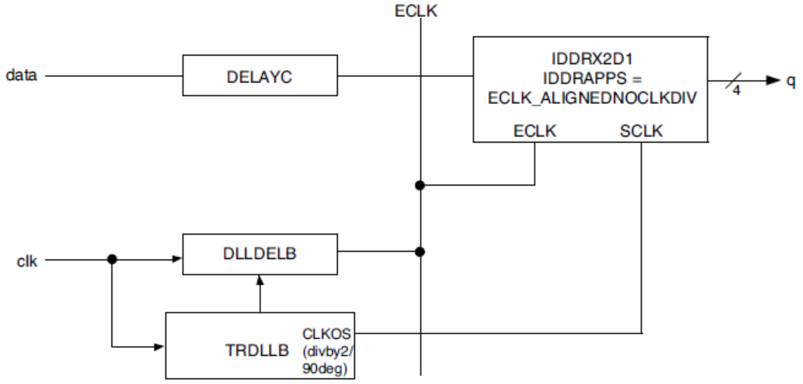


**Figure 6.12 GDDRX2_RX.ECLK.Aligned (No CLKDIV) Interface ("EA" Devices)**

### Interface Rules

It is recommended that a dedicated TGDLL_IN pin be used for the clock input. In an "EA" device a GPLLT_IN or PCLK from the top side can also be used for the clock input.

The clock CLKOS output of the TRDLLB module is connected to SCLK of IDDRX2D1 and it must be routed on a primary clock. The "USE PRIMARY NET<clk>" preference must be used on this clock net as well.

There is only one DLLDELB per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.

## 6.11. GDDRX2_RX.ECLK.Centered

Generic DDR Receive Interface with x2 Gearing using ECLK with Centered External Interface

Device Support: "E" and "EA" devices

### Description

This DDR x2 interface uses the ECLK and DELAYC to match clock and data delay at the IDDRX2D. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

Note the difference in interface implementation between the "E" and "EA" devices. "E" devices require the use of a DQSBUFE which is not required on the "EA" devices.

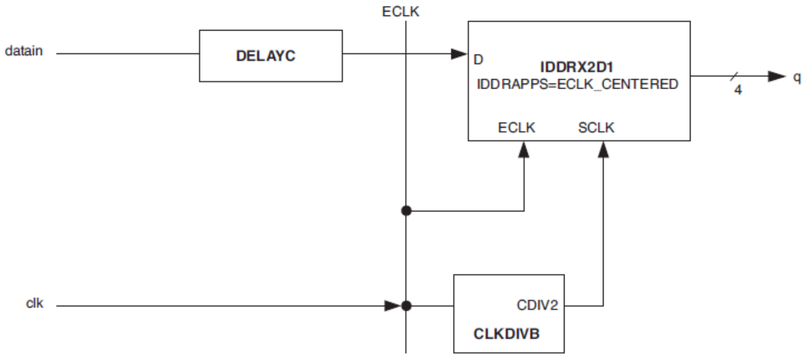


**Figure 6.13 GDDRX2_RX.ECLK.Centered Interface ("EA" Devices)**

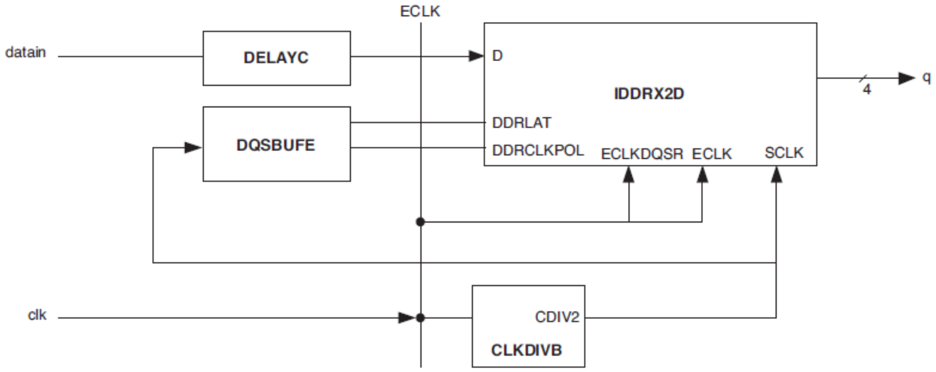


**Figure 6.14 GDDRX2_RX.ECLK.Centered Interface ("E" Devices)**

### Interface Rules

- Input clock port must use a dedicated clock (PCLK) input pin. All the data for the interface must be on the same ECLK tree (same side of the device). In "EA" devices, PLL output in Bypass mode can be connected to the ECLK as well.
- The clock net connected to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- There is only one CLKDIVB per side of the device, so the interface is limited to one per side.
- The data/clock pin assignments for "E" devices require following the DQ-DQS group pinout guidelines. See the Generic DDR Design Guidelines section for details. This is not required for "EA" devices.

## 6.12. GDDRX2_RX.DQS.Aligned

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Aligned External Interface

Device Support: "E" and "EA" devices

### Description

This DDR x2 interface uses the DQS and the DQSDLL to provide a 90° clock shift to center the clock at the IDDRXD. This interface uses a DQS lane and is useful for small data buses (<11 bits). There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.

This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

This interface requires the use of a secondary input pll_clk for the PLL input. The PLL output must run at the same rate as the DQSI clk input, but can be arbitrary phase.

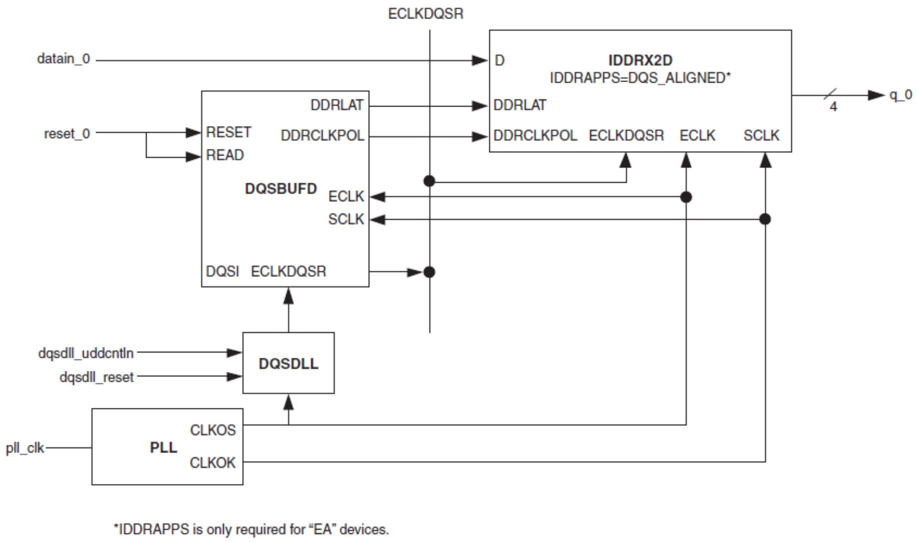


**Figure 6.15 GDDRX2_RX.DQS.Aligned Interface ("E" and "EA" Devices)**

**Interface Rules**

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The input pll_clk must use a GPLLT_IN or GDLLT_IN input pin or from a primary clock tree.
- The clock net connected to SCLK must also be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section "DQSDLLB" on page 88 for the detailed requirements for the DQSDLL_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- The VREF1 for the selected DQS lane must be powered on the board.

## 6.13. GDDRX2_RX.DQS.Centered

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Centered External Interface

Device Support: "E" and "EA" devices

### Description

This DDR x2 interface uses the DQS and DELAYC to match clock and data delay at the IDDRX2D. This interface is useful for small data buses (<11 bits). Since a 90° shift is not required, the DQSDLL is held in reset for this interface. An additional refclk input running at the same as the clk input to DQSI is provided to the DQSDLL, ECLK input of the IDDRX2D. It is also used in the CLKDIV module to generate the SCLK which is half the frequency as the input clock.

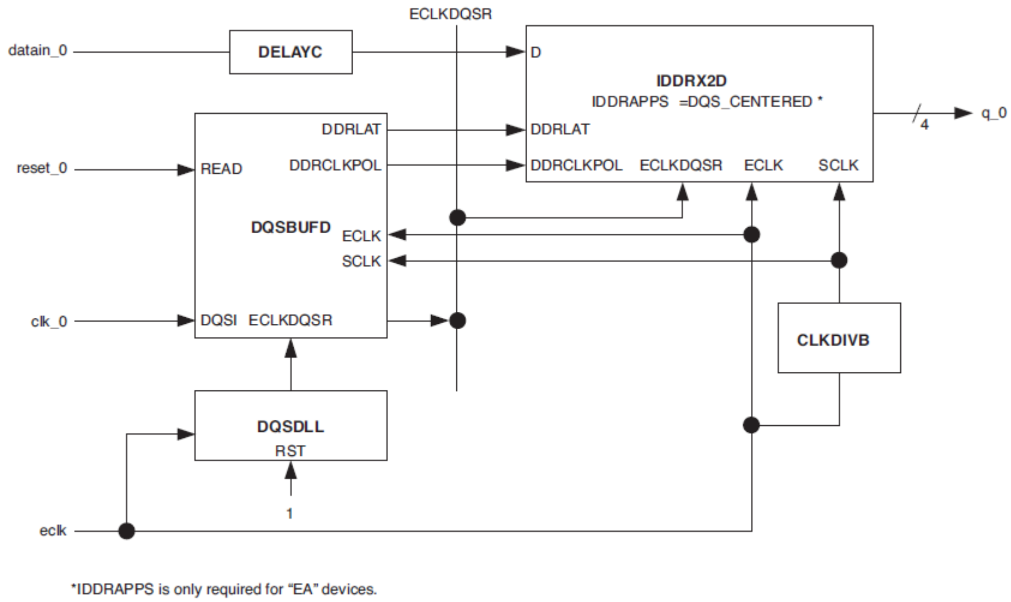


**Figure 6.16 GDDRX2_RX.DQS.Centered Interface ("E" and "EA" Devices)**

**Interface Rules**

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The ECLK clock input must use a dedicated GPLLT_IN input or PCLK input pin or from a primary clock tree. This second synchronous clock input is used for the DQSDLL, ECLK, and CLKDIV.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The clock net connected to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- The VREF1 for the selected DQS lane must be powered on the board.

## 6.14. GDDRX2_RX.ECLK.Dynamic

Generic DDR Receive Interface with x2 Gearing using ECLK with Centered External Interface and Dynamic Data

Delay Control

Device Support: "EA" devices

**Description**

This interface uses a DELAYB and ECLK for bit-level control of the alignment. User logic controls the inputs of the DELAYB delay module. The CLKDIV module is used to generate the SCLK which is half the frequency of ECLK. This interface should only be used when the input clock is centered to the data as this interface does not have phase-shift capability on the clock. This interface is similar to the GDDRX2_RX.ECLK.Centered, but in this version the data delay is controlled dynamically by the user.

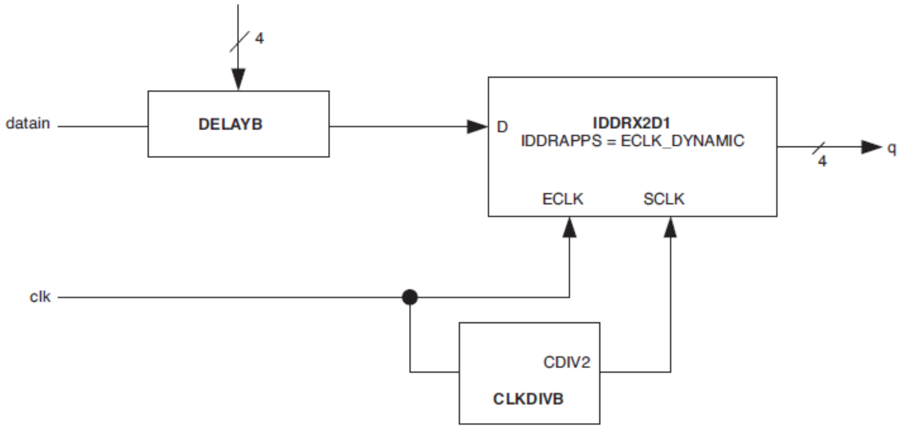


Figure 6.17 GDDRX2_RX.ECLK.Dynamic ("EA" Devices)

**Interface Rules**

- Input clock port must use a dedicated clock (PCLK) input pin. All the data for the interface must be on the same ECLK tree (same side of the device).
- The clock net connected to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- There is only one CLKDIVB per side of the device, so the interface is limited to one per side.

## 6.15. GDDRX2_RX.DQS.Dynamic

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Centered External Interface and Dynamic Data Delay Control

Device Support: "EA" devices

**Description**

This interface uses a DELAYB and DQS lane for bit-level control of the alignment. User logic controls the inputs of the DELAYB delay module. CLKDIV module is used to generate the SCLK which is half the frequency of ECLK. This interface should only be used when the input clock is centered to the data as this interface does not have phase-shift capability on the clock. This interface is similar to the GDDRX2_RX.DQS.Centered, but in this version the data delay is controlled dynamically by the user.

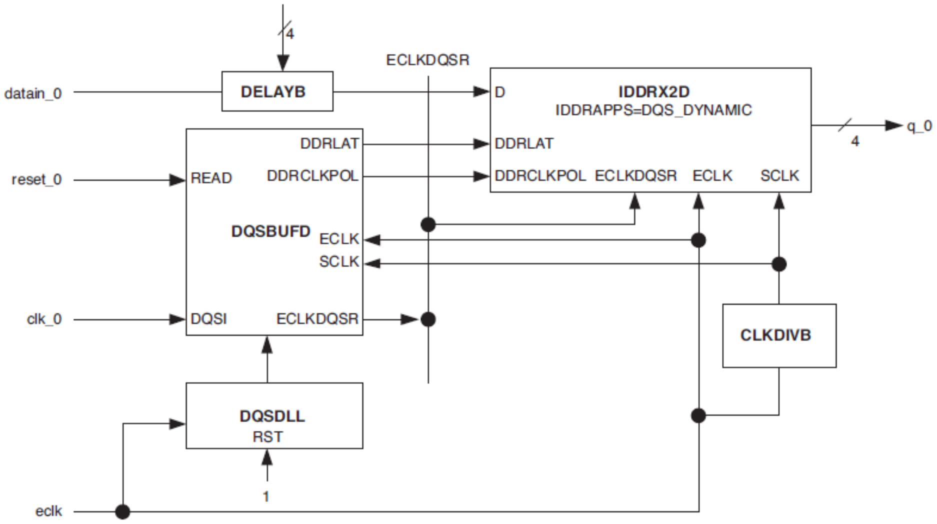


Figure 6.18 GDDRX2_RX.DQS.Dynamic ("EA" Devices)

### Interface Rules

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The eclk clock input must use a dedicated GPLLT_IN input or PCLK input pin. This second synchronous "eclk" input is used for the DQSDLL, ECLK, and CLKDIV.
- The clock net connected to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.

## 6.16. GDDRX2_RX.PLL.Dynamic

Generic DDR Receive Interface with x2 Gearing using ECLK with Dynamic control on ECLK phase using PLL Device Support: EA devices

### Description

This interface uses a PLL to delay the ECLK for bus-level control of the alignment. The benefit of the PLL is that an entire period of delay is provided. User logic controls the DPHASE input to the PLL.

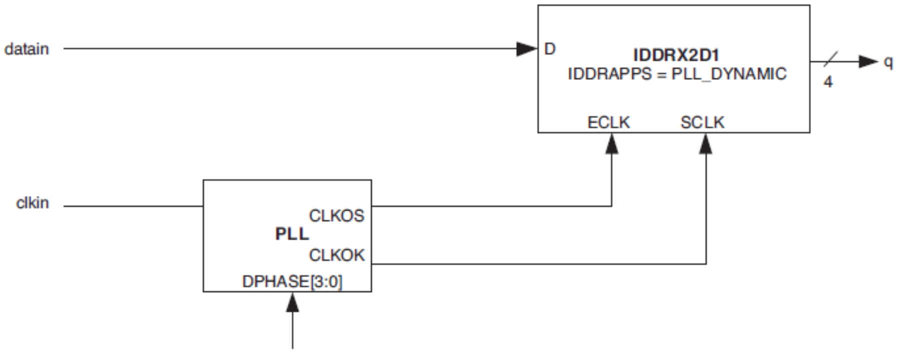


**Figure 6.19 GDDRX2_RX.PLL.Dynamic ("EA" Devices)**

### Interface Rules

- The input clock must use a dedicated GPLLT_IN clock input pin. The clock net connected to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.

## 6.17. GOREG_TX.SCLK

Generic SDR Transmit Interface using SCLK

Device Support: "E" and "EA" devices

### Description

This is a generic interface for SDR data and a forwarded clock. The ODDR used for the clock balances the clock path to match the data path. A PLL can also be used to clock the ODDRXD to phase shift the clock to provide a precise clock-to-data output.

On "E" devices, the sides (left and right) need to pass through a DQSBUFG before going to the ODDRXD element. The top does not require the DQSBUFG and can take SCLK directly.

The "EA" device does not require a DQSBUFG block.

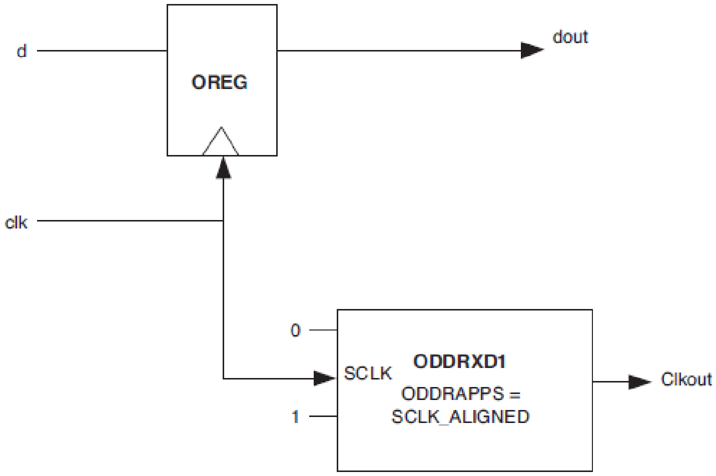


**Figure 6.20 GOREG_TX.SCLK Interface (“EA” Devices)**

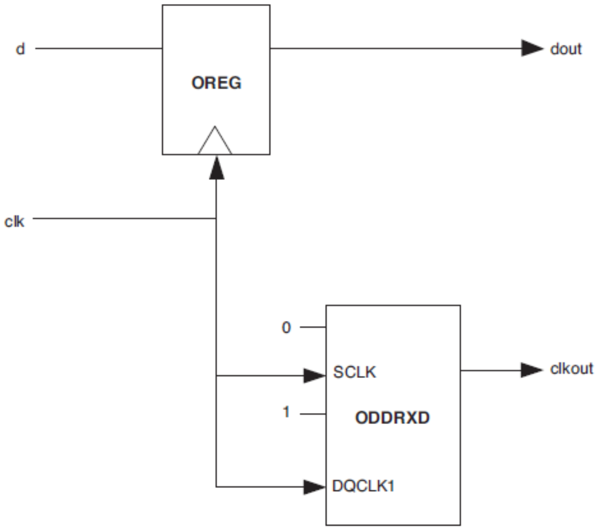


**Figure 6.21 GOREG_TX.SCLK Interface (“E” Devices, Top)**

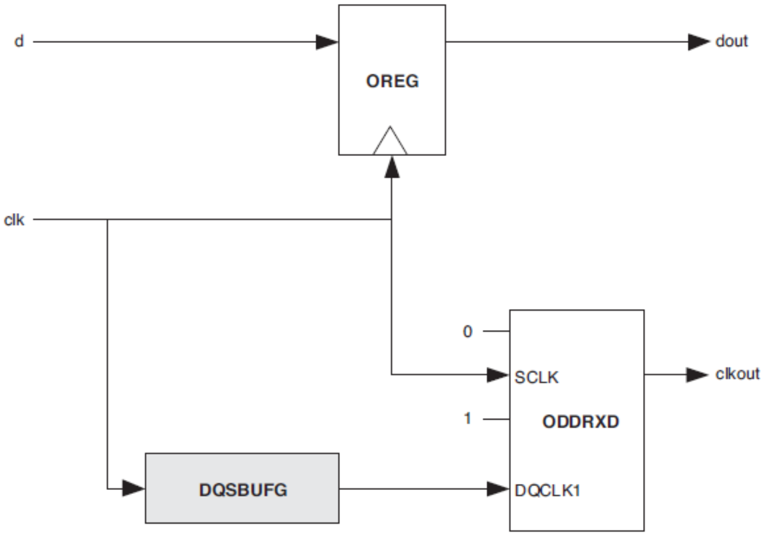


**Figure 6.22 GOREG_TX.SCLK Interface (“E” Devices, Left/Right)**

### Interface Rules

- SCLK must be routed on either primary or secondary clock resources using the USE PRIMARY NET<clk> or USE SECONDARY preferences.

## 6.18. GDDRX1_TX.SCLK.Centered

Generic DDR Transmit Interface using SCLK with Centered External Interface

Device Support: "E" and "EA" devices

### Description

This output DDR interface provides clock and data that are pre-centered using a PLL and two SCLKs.

On "E" devices, the left and right sides need to pass through a DQSBUFG before going to the ODDRXD element. The top side of the "E" device does not require the DQSBUFG and can take SCLK directly. When using the DQS- BUFG, the clock output needs to be in a different DQS lane than the data since there is only one DQSBUFG per lane.

The "EA" device does not require the DQSBUFG.

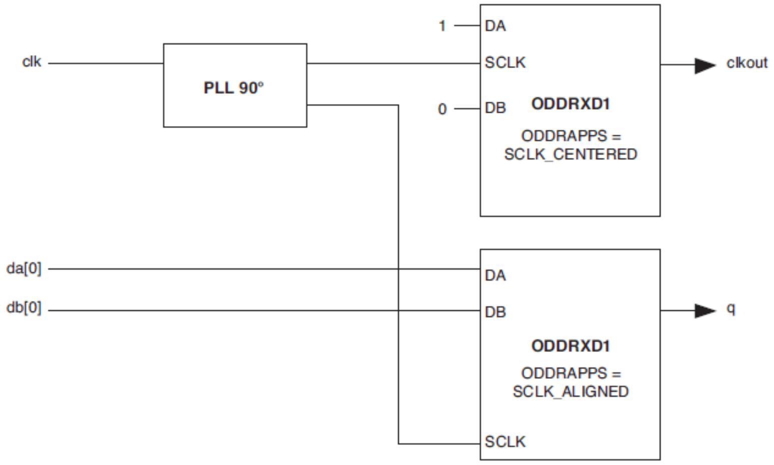


**Figure 6.23 GDDRX1_TX.SCLK.Centered Interface ("EA" Devices)**

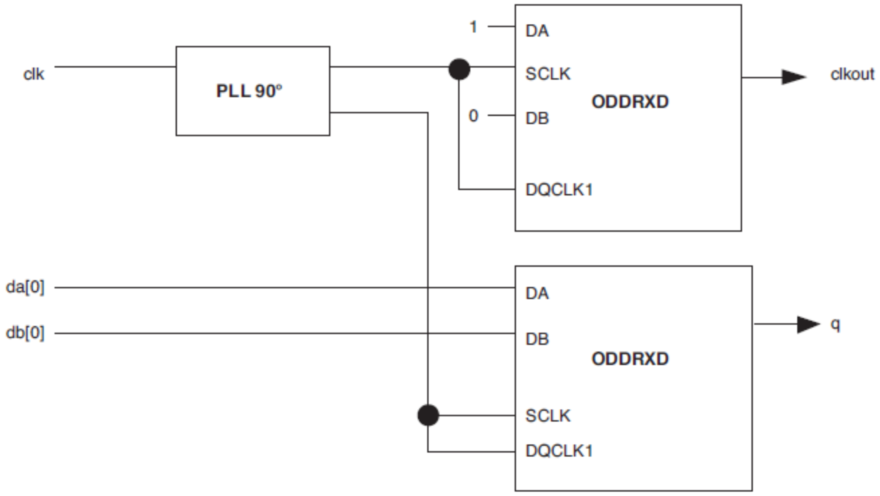


**Figure 6.24 GDDRX1_TX.SCLK.Centered Interface ("E" Devices, Top)**

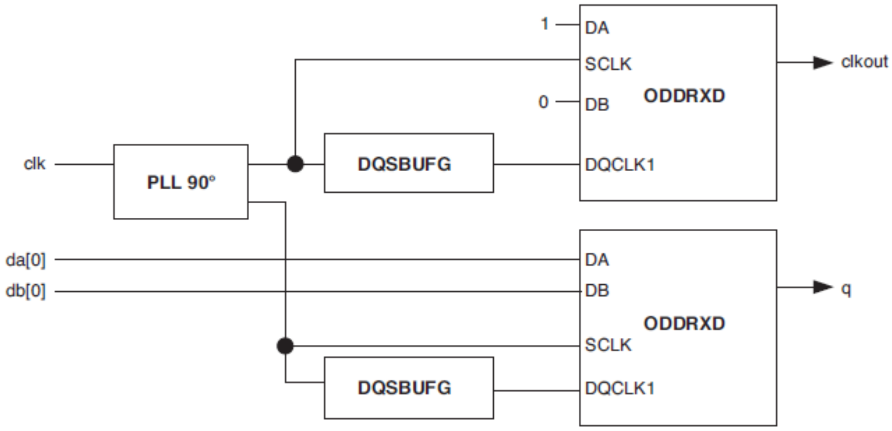


**Figure 6.25 GDDRX1_TX.SCLK.Centered Interface (“E” Devices, Left/Right)**

**Interface Rules**

- On “E” devices, the clock and data outputs need to be in different DQS lanes on the left and right sides since there is only one DQSBUFG per lane. Clock and data outputs can use the same DQS lane on top. Clock and data outputs cannot use the DQS site. They must use the DQ site.
- SCLK and 90° shifted SCLK should be assigned to a primary clock pin using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock require following the DQ-DQS lane assignments for the “E” device. Refer to the Generic DDR Design Guidelines section for details. “EA” devices do not have this requirement. The clock pin to the PLL path must be routed on a dedicated clock route. A dedicated GPLLT_IN pin must be used for input of this clock.

## 6.19. GDDRX1_TX.SCLK.Aligned

Generic DDR Transmit Interface using SCLK with Aligned External Interface

Device Support: “E” and “EA” devices

**Description**

This output DDR interface provides clock and data that are aligned using a single SCLK.

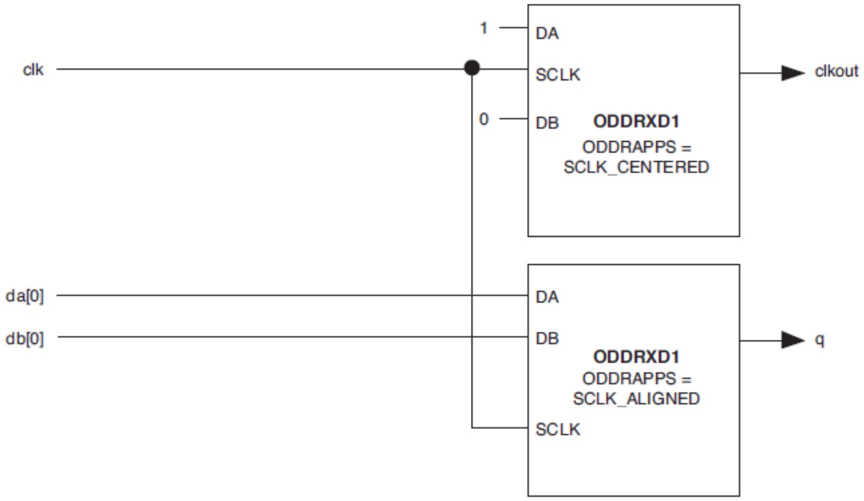


**Figure 6.26 GDDRX1_TX.SCLK.Aligned Interface (“EA” Devices)**

"E" devices require the use of DQSBUFG on the left and right sides of the device. If the clock and data bus can fit in the same DQS lane then a single DQSBUFG is all that is needed (<10 bits). For a wider data bus (>0 bits) it is required to use a DQSBUFG for clock and data and assign the clock to a different DQS lane.

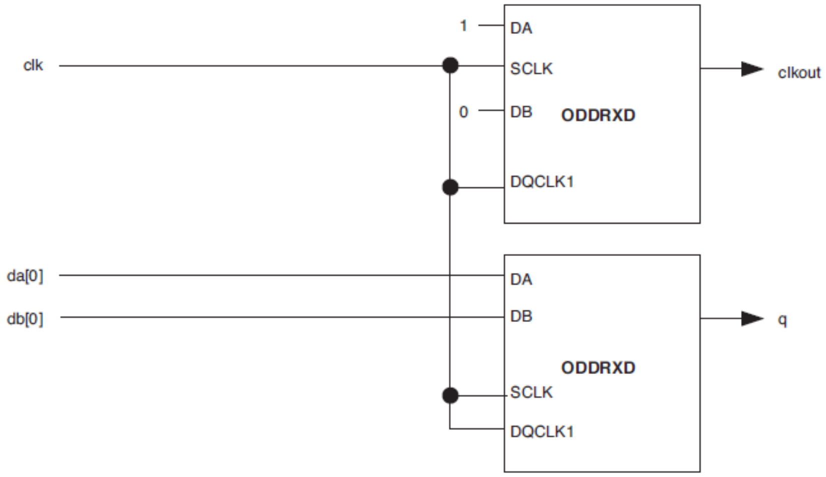


**Figure 6.27 GDDRX1_TX.SCLK.Aligned Interface ("E" Devices, Top Side))**

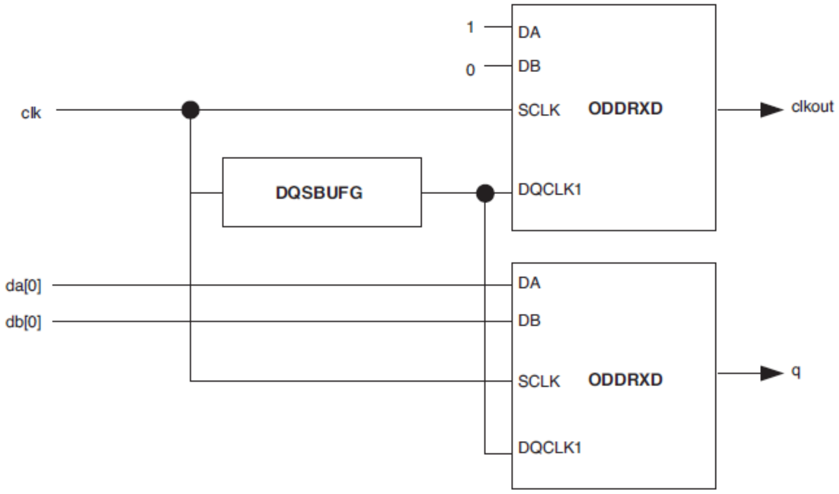


**Figure 6.28 GDDRX1_TX.SCLK.Aligned Interface ("E" Devices, Left/Right Sides) < 10 Bits**

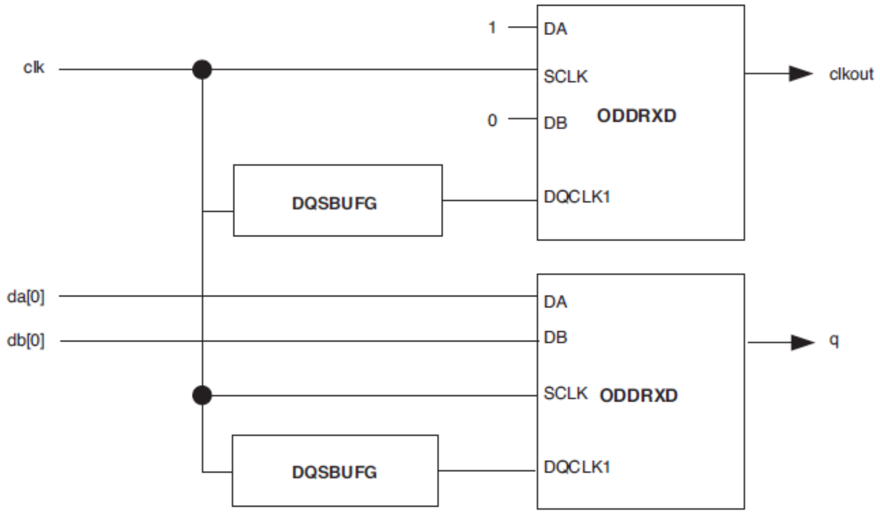


**Figure 6.29 GDDRX1_TX.SCLK.Aligned Interface (“E” Devices, Left/Right Sides) > 10 Bits**

### Interface Rules

- On “E” devices, the clock and data outputs need to be in different DQS lanes on the left and right sides since there is only one DQSBUFG per lane. Clock and data outputs can use the same DQS lane on top. Clock and data outputs cannot use the DQS site. They must use the DQ site.
- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock require following the DQ-DQS lane assignments for “E” devices. Refer to the Generic DDR Design Guidelines section for details. “EA” devices do not have this requirement.

## 6.20. GDDRX1_TX.DQS.Centered

Generic DDR Transmit Interface using DQS Lane with Centered External Interface

Device Support: “E” and “EA” devices

### Description

This output DDR x1 interface provides clock and data that is pre-centered using a DQSDLL and ODDRXDQSA. This is the same as the GDDRX1_TX.SCLK.Centered, but does not require a PLL. This interface can also be used multiple times using the same DQSDLL. This interface should be used for narrow data buses (<11 bits wide)

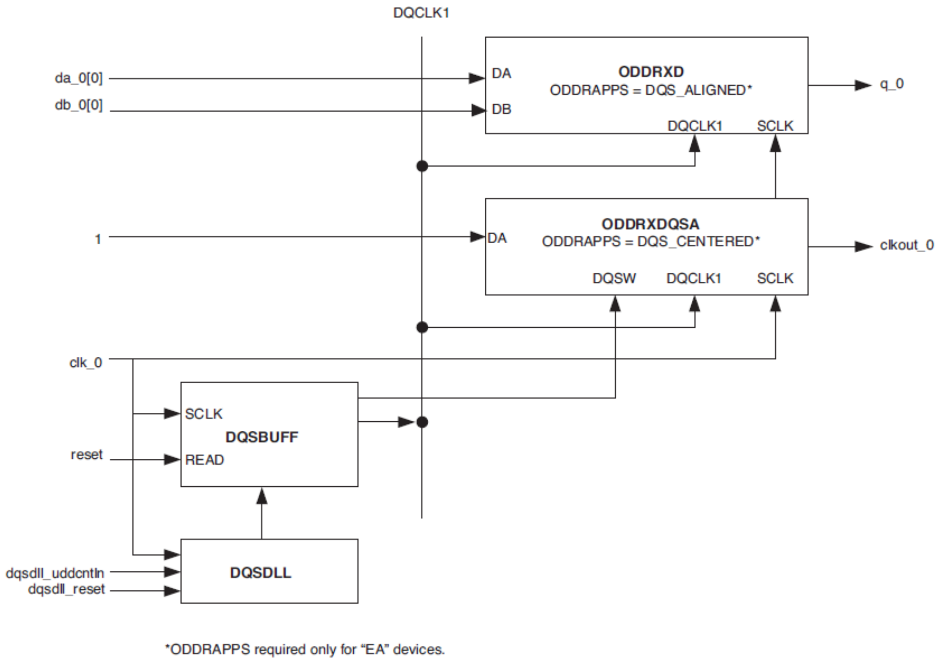


**Figure 6.30 GDDRX1_TX.DQS.Centered Interface ("E" and "EA" Devices)**

**Interface Rules**

- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- There is only one DQSDLL per side of the device. One of these interfaces can be placed in each DQS group, but they all need to run at the same rate for that side.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section "DQSDLLB" on page 88 for the detailed requirements for the DQSDLL_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- "clk_0" must use the primary clock tree. It is your responsibility to assign "USE PRIMARY NET<clk>" preference on this net to assign it to the primary clock.
- This interface cannot use the LVDS25 IO_TYPE for the clkout from the DQSI.

## 6.21. GDDRX2_TX.Aligned

Generic DDR x2 Transmit Interface with Aligned External Interface

Device Support: "EA" devices only

**Description**

This output DDR x2 interface provides clock and data that are aligned. A PLL is used to generate ECLK. A CLKDIV is use to generate the SCLK which is half the frequency of the ECLK. The PLL CLKOK can also be used to generate the SCLK. Additional soft logic is required for this interface to work as expected. This logic is used to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

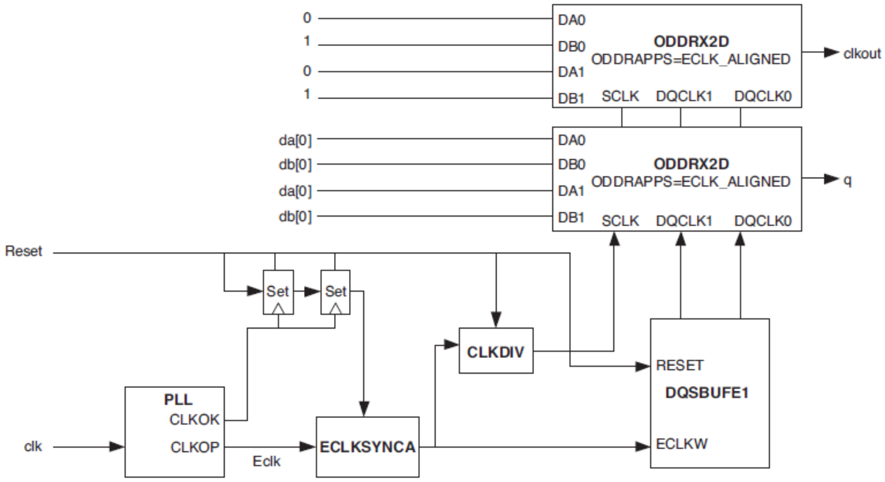


Figure 6.31 GDDRX2_TX.Aligned Interface ("EA" Devices)

### Interface Rules

- Clock input must use a dedicated GPLLT_IN input pin or from a primary clock tree.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- The additional soft logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.
- The clock to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.

## 6.22. GDDRX2_TX.DQSDLL.Centered

Generic DDR x2 Transmit Interface with Centered External Interface using DQSDLL

Device Support: "EA" devices

### Description

This output DDR x2 interface provides a clock and data that are centered. This interface uses a DQSDLL along with the DQSBUFD to generate the 90° delayed clock used to generate the clock output. ODDRDQSA module is used for clock generation. A CLKDIV is use to generate SCLK which is half the frequency of the ECLK. Additional logic is needed to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

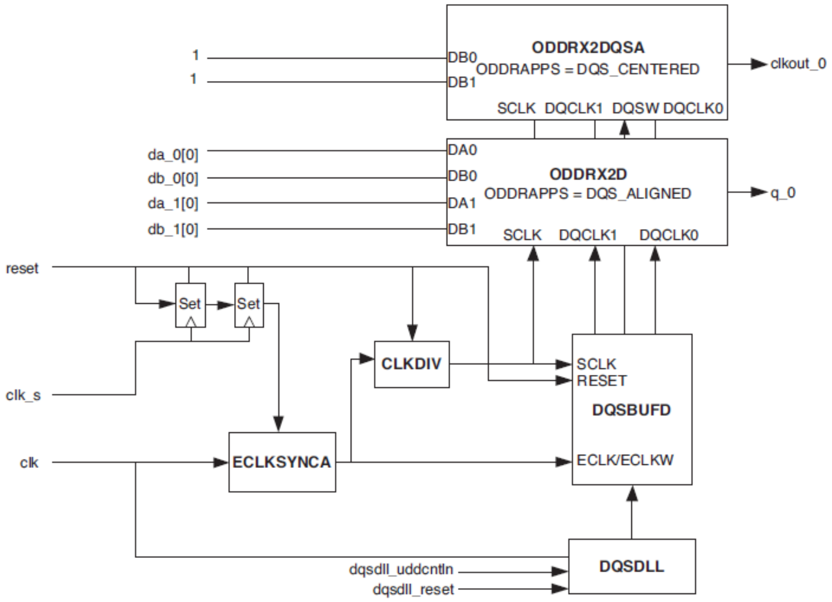


**Figure 6.32 GDDRX2_TX.DQSDLL.Centered Interface ("EA" Devices)**

### Interface Rules

- Clock inputs must come in from a primary clock tree.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- The additional logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.
- There is only one DQSDLL per side of the device. One of these interfaces can be placed in each DQS group, but they all need to run at the same rate for that side.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section "DQSDLLB" on page 88 for the detailed requirements for the DQSDLL_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- The clock to SCLK must be routed on a primary routing resource using the "USE PRIMARY NET<clk>" preference.
- CLKOUT must be assigned to a DQS pin. The DQS pins do not support True LVDS outputs, hence CLKOUT cannot use LVDS IO_TYPE.

## 6.23. GDDRX2_TX.PLL.Centered

Generic DDR x2 Transmit Interface with Centered External Interface using PLL

Device Support: "EA" devices only

### Description

This output DDR x2 interface provides a clock and data that are centered. This interface uses a PLL to generate the 90° phase shift required for the clock. A CLKDIV is used to generate SCLK which is half the frequency of the ECLK. Additional logic is required to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

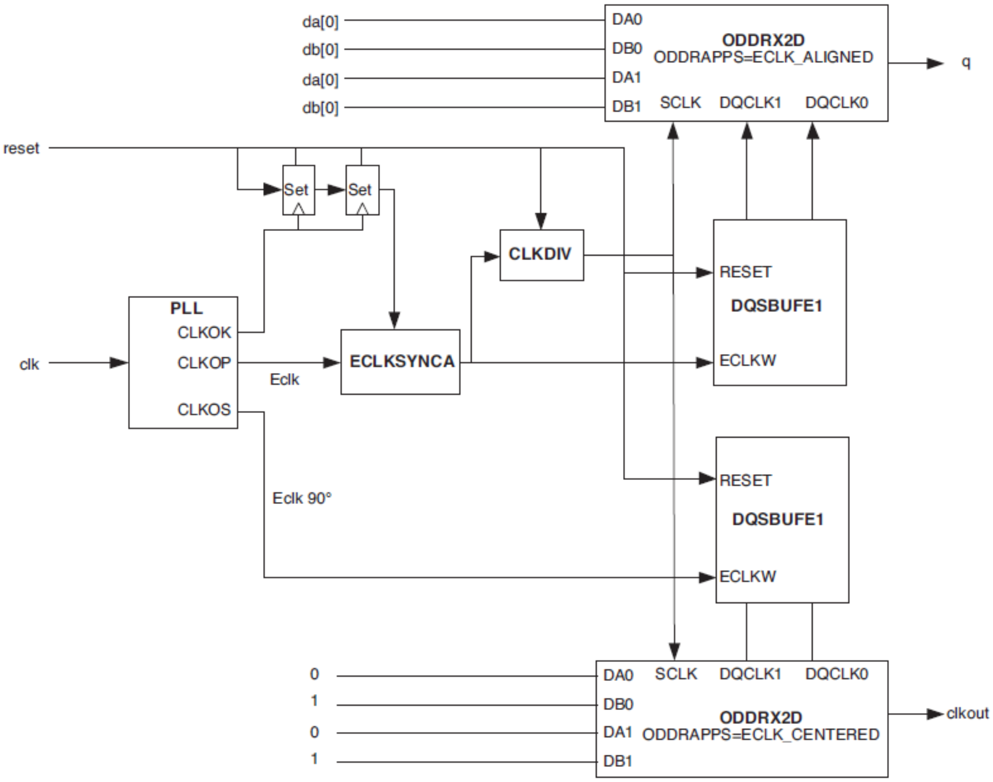


Figure 6.33 GDDRX2_TX.PLL.Centered (“EA” Devices)

**Interface Rules**

- Clock input must use a dedicated GPLLT_IN input pin or from a primary clock tree.
- The pin assignments for data and clock require following the DQ-DQS lane assignments. Refer to the Generic DDR Design Guidelines section for details.
- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>“preference.
- The additional soft logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.

## 6.24. 7:1 LVDS Implementation

It is recommended that the 7:1 LVDS Video Interface Reference Design provided on the lattice web site be used to implement all 7:1 LVDS designs.

# 7. Generic DDR Design Guidelines

This section describes the various design guidelines used for building generic high-speed DDR interfaces in LatticeECP3 FPGAs. In additional to these guidelines, it is also necessary to follow the interface rules for each interface type as described above.

## 7.1. Placement Guidelines for High-Speed DDR Interfaces

The following clock and data placement guidelines should be followed for high-speed design requirements. The software places the clock and data you specified. In order to achieve higher speeds, you must follow the placement rules listed in Table 7.1 for each interface type.

It is required that all clocks used to clock the DDR Interfaces use a dedicated clock path. No general routing should be used to route the clock pin. General routing used for a clock path causes duty cycle distortion as well as limit the operational frequency of the interface. It is your responsibility to assign clock inputs to dedicated clock pins and use preferences such as “USE PRIMARY NET<clk>” to route clock nets on dedicated clock paths.

Table 7.1 lists the various high-speed interfaces and the placement required for the clock and data.

**Table 7.1. Pin Placement Guidelines for High-Speed Interfaces**

| DDR Interface | LatticeECP3 “EA” Devices | | | LatticeECP3 “E” Devices | | |
|---|---|---|---|---|---|---|
| | DATA | CLK | CLK PIN | DATA | CLK | CLK PIN |
| GDDRX1_RX.SCLK.Aligned | L/R/T | L/R | GDLLT_IN | N/A | N/A | N/A |
| | L/R/T | L/R | GPLLT_IN | N/A | N/A | N/A |
| | L/R/T | T | PCLK | N/A | N/A | N/A |
| GDDRX1_RX.SCLK.Centered | L/R/T | L/R/T | PCLK | N/A | N/A | N/A |
| | L/R/T | L/R/T | GPLLT_IN | N/A | N/A | N/A |
| GDDRX1_RX.ECLK.Aligned | N/A | N/A | N/A | L/R | L/R | GDLLT_IN |
| GDDRX1_RX.ECLK.Centered | N/A | N/A | N/A | L/R | L/R | PCLK |
| GDDRX1_RX.DQS.Aligned | L/R/T | L/R/T | DQS | L/R | L/R | DQS |
| GDDRX1_RX.DQS.Centered | L/R/T | L/R/T | DQS | L/R | L/R | DQS |
| GDDRX2_RX.ECLK.Aligned | L/R/T | L/R | GDLLT_IN | L/R | L/R | GDLLT_IN |
| | L/R/T | L/R | GPLLT_IN | N/A | N/A | N/A |
| | L/R/T | T | PCLK | N/A | N/A | N/A |
| GDDRX2_RX.ECLK.Centered | L/R | L/R | PCLK | L/R | L/R | PCLK |
| | L/R/T | L/R | GPLLT_IN | N/A | N/A | N/A |
| GDDRX2_RX.DQS.Aligned | L/R | L/R | DQS | L/R | L/R | DQS |
| GDDRX2_RX.DQS.Centered | L/R | L/R | DQS | L/R | L/R | DQS |
| GDDRX2_RX.ECLK.Dynamic | L/R | L/R | PCLK | N/A | N/A | N/A |
| GDDRX2_RX.DQS.Dynamic | L/R | L/R | DQS | N/A | N/A | N/A |
| GDDRX2_RX.PLL.Dynamic | L/R | L/R | GPLLT_IN | N/A | N/A | N/A |
| | L/R/T | L | High Speed Bridge[2,3] | N/A | N/A | N/A |
| GDDRX2_RX.ECLK.DR | L/R | L/R | GPLLT_IN | N/A | N/A | N/A |
| GDDRX1_TX.SCLK.Centered | L/R/T | L/R/T | ANY | L/R | L/R | ANY |
| GDDRX1_TX.SCLK.Aligned | L/R/T | L/R/T | ANY | L/R | L/R | ANY |
| GDDRX1_TX.DQS.Centered | L/R/T | L/R/T | DQS | L/R | L/R | DQS |
| GDDRX2_TX.ECLK.Aligned | L/R | L/R | ANY | L/R | L/R | ANY |
| GDDRX2_TX.Centered (DQS) | L/R | L/R | DQS | L/R | L/R | DQS |
| GDDRX2_TX.Centered (PLL) | L/R | L/R | ANY | L/R | L/R | ANY |

**Notes:**

1. L, R and T refer to “Left”, “Right” and “Top” sides of the device.

2. The high-speed clock bridge can be accessed by using the "USE EDGE2EDGE < clk> "software preference. For preference details please see "ispLEVER Help" in the software.
3. High-speed bridge is only available on "EA" devices.
4. Top-side DDR is not supported on "E" devices.

## 7.2. High-Speed Clock Bridge ("EA" Devices)

The high-speed clock bridge is available only on "EA" devices on the GDDRX2.RX.PLL_Dynamic interface. The bridge enables a clock to route to a single edge clock or multiple edge clocks on the device using a three-way (left/right/top) bridge. It can only be used on clocks coming in from the left side dedicated GPLLT pin or PLL output. The software preference EDGE2EDGE is used to enable this route.

For example, USE EDGE2EDGE <pllin_c>; where "pllin_c" is the clock net coming from the dedicated GPLLT pin.

When this preference is placed on the clock coming in on the left side GPLLT pin, this clock is connected to the one of the ECLK on the left, right and top sides using a dedicated route. This ECLK on all three sides cannot be used for any other ECLK function. You have to make the following changes to the GDDRX2.RX.PLL_Dynamic module generated by IPExpress to incorporate the ECLKBRIDGE.

- A CLKDIV should be used to generate SCLK instead of using PLL CLKOK. The CLKOS output of the PLL should be connected to the input of the CLKDIV module. Output of CLKDIV is used as SCLK.
- You have to instantiate 2 ECLKSYNC modules to be used on either side of the device. Both of them should be connected to the CLKOS of the PLL used as ECLK.
- An EDGE2EDGE preference should be assigned to the CLKOS output of the PLL to be used as ECLKBRIDGE.

## 7.3. DQ-DQS Grouping Rules

Due to differences in architecture between the LatticeECP3 "E" and "EA" devices, the DQ-DQS grouping that is required for some interfaces in "E" devices is not required for "EA" devices. It is necessary to use the DQS grouping structure to group pins when either of the DQSBUFE/DQSBUFF/DQSBUFG modules is used in an interface. Refer to the section High-Speed DDR Interface Details to see the requirements for each device.

Below are some of the rules to be followed when locking DQS groups. The ispLEVER checks for these rules during MAP and Place and Route.

- Each DQS pin has a DQSBUF block which spans across 12 pins including the DQS and DQS# pins. Each DQSBUF sends out control logic to these pins.
- The DQS# I/O logic registers cannot be used to implement DDR registers. DQS pins can be used for IDDR implementation only. ODDR cannot be assigned to a DQS pin.
- The DQS and DQS# I/O logic registers can be used to implement SDR input and output registers. If the DQSBUF of that DQS group is used then it cannot be used for SDR functions unless the same clock going to the DQSBUF is used to clock the SDR register at the DQS/DQS# site.
- An IDDRX element used for a generic DDR interface cannot be mixed in a DQS group with an ODDRX element used for implementing a DDR memory interface.
- Similarly, an ODDRX element used for a generic DDR interface cannot be mixed in a DQS group with an IDDRX element used for a DDR memory interface.
- The ODDRXD and ODDRX2D elements in a given DQS group cannot share the same DQSBUF and therefore cannot be placed together within the same DQS group.
- The upper left corner of the LatticeECP3 device has a DQS group without a DQS pin. This group of I/O does not have a DQS function. It is recommended to use this group for pins in non-DQS interfaces.
- See Table 17.1 to see the availability on each side.

## 7.4. I/O Logic (IOL) Site Types/Names

Based on the functions they support, I/O logic blocks are divided into the following site types.

- IOLOGIC – This site supports IREG, OREG, IDDRX, IDDRX2, ODDRX and ODDRX2 functions. These are the IOLs on the left and right sides of the device.
- SIOLOGIC – This site supports IREG, OREG, IDDRX, IDDRX2 and ODDRX functions. There is no ODDRX2 support in this site. These are mainly the IOLs on the top side of the device.
- XSIOLOGIC – This site supports IREG and OREG only. These are primarily the bottom side IOLs and DQS# IOLs.
- DQSIOL – These are the DQS IOLs. They support IREG, OREG, IDDRXD, IDDRX2, ODDRXDQSA and ODDRX2DQSA (left and right sides only) functions
- SDQSIOL – These are DQS IOLs with support for IREG, OREG and DQS ODDRXDQSA functions. Compared to DQSIOL, there is no ODDRX2DQSA support in this site. These are mostly the DQS IOLs on the top side of the device.

The software issues an error message using these site names when an unsupported function is placed on one of these sites.

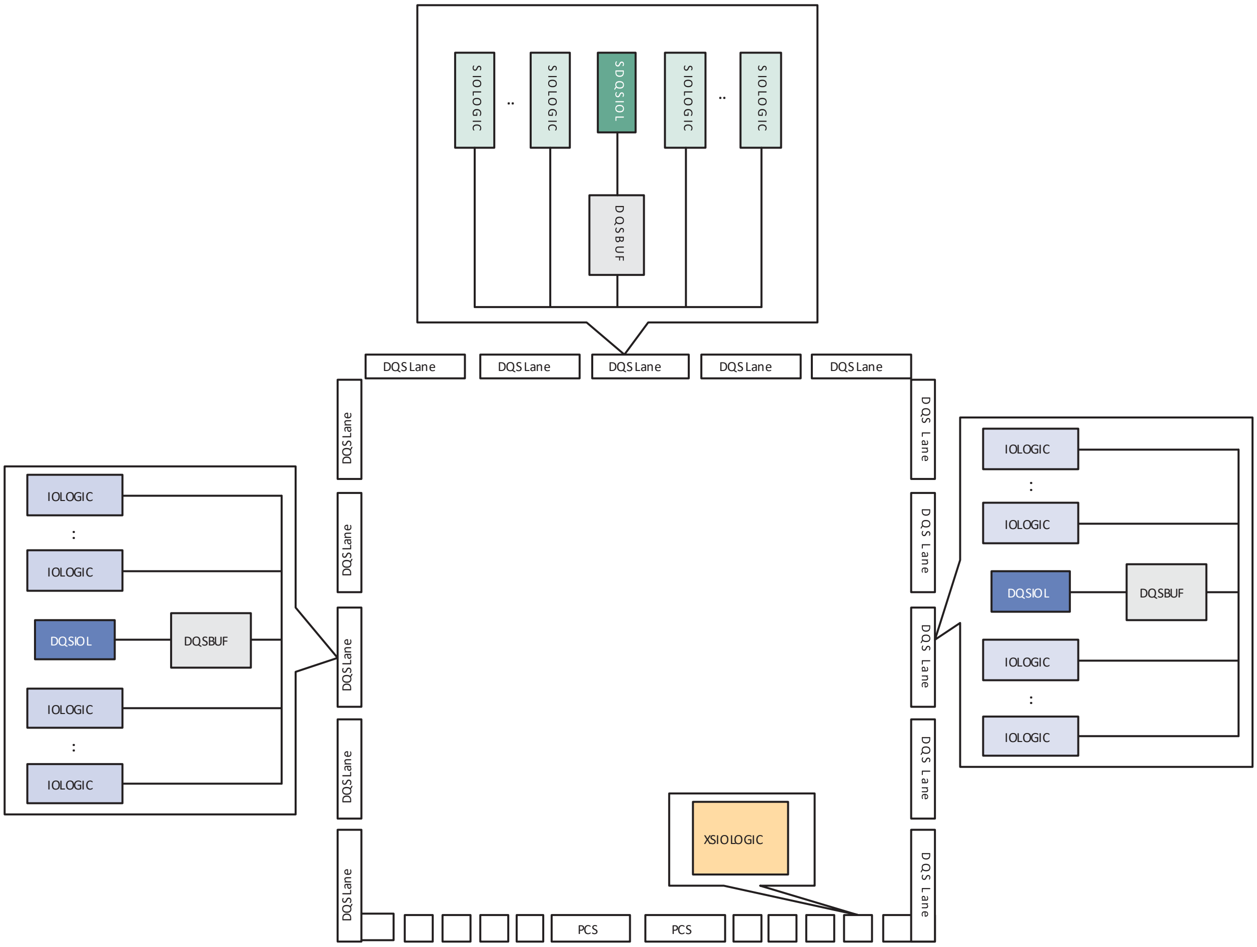


**Figure 7.1 IOLOGIC Site Types**

## 7.5. Design Rules for Fitting Multiple Interfaces into One Device

Rx Interfaces Running at High Speeds (>250 MHz)

Receive interfaces running at speeds higher than 250 MHz must use the x2 mode gearing DDR elements.

- To achieve high speeds these interfaces must be placed on the left and right sides of the device.
- If implementing a centered interface then the clock input must be locked to a primary clock (PCLK) input pin

- If implementing an aligned interface then the clock input must be locked to a dedicated GPLLT if interface is using a PLL GDLLT input pin if interface is using a DLL.
- Interfaces using the x2 gearing needs to use the edge clock resource. A single edge clock covers only one side of the device, hence all the data bits in the data bus should be assigned to one side of the device.
- There are two edge clocks per side. Two interfaces can be implemented per side of the device as long as they both do not require the DLL or CLKDIV module. See below:
  - There is only one CLKDIV and one DLL module available per side, hence two interfaces that use the CLK- DIV and/or the DLL module cannot be on the same side.
  - When implementing an aligned interface using the DLL module (GDDRX2_RX.ECLK.Aligned) only one interface can be implemented per side as there is only one DLL and one CLKDIV per side of the device.
  - When implementing an aligned interface using a PLL (GDDRX2_RX.PLL.Dynamic), up to two of these interfaces can be implemented per side. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - You can mix an aligned interface using a PLL (GDDRX2_RX.PLL.Dynamic) and a centered interface (GDDRX2_RX.ECLK.Centered) on the same side of the device. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - You can mix an aligned interface using a DLL (GDDRX2_RX.ECLK.Aligned) and a centered interface (GDDRX2_RX.ECLK.Centered) on the same side of the device. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - A GDDRX2_RX.ECLK.Aligned and GDDRX2_RX.PLL.Dynamic can be implemented on the same side of the device. In this case, the clock going to the PLL must be locked to a dedicated PLL clock input pin in the center of the device close to the DLL pin. For example, if the DLL pin is placed on the LUM0_GDLLT_IN_A pin then the clock input to the PLL must be placed on the LUM0_GPLLT_IN_A pin. It is recommended to allow the software to pick the best clock pins after locking the data inputs to the desired banks
  - For all interfaces using PLLs, refer to LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide (FPGA-TN-02191) to see which two dedicated PLL clock outputs can feed the ECLK tree
- When multiple receive interfaces with multiple input clocks are required, and if the data widths for each of these interfaces is <10 bits wide, then the DQS clock can be used to implement these interfaces (GDDRX2_RX.DQS.Aligned and GDDRX2_RX.DQS.Centered)
  - For these interfaces it is required to connect the input clock to the DQS input pin
  - Data bits of the bus must be locked to the corresponding data pins
  - See the section “DQ-DQS Grouping Rules” for pin assignment rules.

### Rx Interfaces Running at Low Speeds (<250 MHz)

- Receive interfaces running at speeds lower than 250 MHz can use the x1 mode gearing DDR elements.
- If implementing a centered interface then the clock input must be locked to a primary clock (PCLK) input pin
- If implementing an aligned interface then the clock input must be locked to a dedicated “GPLLT” if interface is using a PLL “GDLLT” input pin if interface is using a DLL.
- Interfaces using the x1 gearing uses the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.
  - If all interfaces are aligned and using a GDDRX1_RX.SCLK.Aligned interface that uses a DLL, then the number of interfaces is limited to two as there are two DLLs supported per device.
  - If using a GDDRX1_RX.SCLK.PLL.Aligned interface then the number of interfaces per device is limited by the number of PLLs available in that device.

### Tx Interface Running at High Speeds (> 250 MHz)

- Transmit Interfaces running at speeds higher than 250 MHz must use the x2 mode gearing DDR elements.
- To achieve high speeds these interfaces must be placed on the left and right sides of the device.
- Both edge clock and DQS clocks are used to implement the transmit side interfaces
- Data pins must be grouped into the DQS groups on these interfaces
- For interfaces requiring True LVDS outputs, the number of available True LVDS pins per DQS group must be calculated from the data sheet pinout tables since only a limited number of pins support True LVDS outputs. It may be required to modify the HDL generated from IPexpress to reduce the number of pins assigned to each DQS group

- All transmit interfaces using the x2 gearing mode use the CLKDIV module, hence only one TX interface can be implemented per side of the device.
- Inputs to the PLL used in the interface should be on primary clock routing or come in from a dedicated GPLLT input pin

**Tx Interface Running at High Speeds (<250 MHz)**

- Transmit interfaces running at speeds lower than 250 MHz can use the x1 mode gearing DDR elements
- Interfaces using the x1 gearing uses the primary clock resource. You can as many interfaces as the number of primary clocks supported in the device.
- If all interfaces are centered, then the number of interfaces depend on the number of available PLLs on the device.

## 7.6. Clocking Guidelines for Generic DDR Interfaces

- Edge clock and primary clock resources are used when implementing a x2 receive or transmit interface
- Only the primary clock (PCLK) resources are used when implementing x1 receive or transmit interfaces
- Each edge clock can only span one side of the device, hence all the data bits of the in the x2 interface must be locked to one side of the device
- When implementing x1 interfaces, the bus can span any two sides as primary clocks can access DDR registers on all sides
- There are two edge clocks on each left, right of top side of the device. The bottom side does not support DDR registers and does not have any edge clocks. There are up to eight primary clocks available on a LatticeECP3 device.
- For high-speed interfaces it is recommended to use the edge clocks on the left and right sides of the device instead of the top side
- See Design Rules for Fitting Multiple Interfaces into One Device for details on implementing multiple interfaces on one side of a device
- The ECLK to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DLL outputs, and GPLL input pins. See "LatticeECP3 sysCLOCK PLL/DLL Design & Usage Guide" for details
- Primary clock to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DLL outputs & CLKDIV outputs. See "LatticeECP3 sysCLOCK PLL/DLL Design & Usage Guide" for details
- None of the clocks going to the DDR registers can come from internal general routing.
- DQS clocking is mainly used for DDR memory interface implementation.
- The DQS clock spans every 12 I/O's include the DQS pins, but only 10 of these can be used for generic DDR implementation
- DQS clocking can also be used for Generic DDR Receive interfaces when the bus size is 10 or less
- DQS clocking is also used when implementing all Generic DDR x2 Transmit interfaces
- Refer to the "DQ-DQS Grouping Rule" section for pinout assignment rules when using DQS clocking

## 7.7. Common Software Errors and Solutions

- Placement error when implementing Transmit high speed True LVDS interface:
  "ERROR - par: DDR assignment finished unsuccessfully."
  DQS grouping is used when implementing 2x gearing on the transmit side. This error occurs then there aren't enough true LVDS buffers in one DQS group. IPexpress generates DQS groups correctly for emulated buffers. Since the number of true LVDS pins vary per device and package, you need to update the module generated by IPexpress to correct the number of pins per DQS group.
  - If more true LVDS pairs are required than available in a DQS group, consider instantiating another DQS buffer adjacent to the DQS buffer used by the IPExpress module and use the fourth LVDS pair.
- PAR Routing error when not using dedicated clock routing:
  "ERROR - par: netsanitycheck: the clock buf_clk on comp Inst4_DLLDELB port CLKI is driven by general routing through comp clk"
  This error usually occurs when general routing is used on any of clocks routed to any of the DDR modules.
  Check the following:

  - If using a DLL or PLL, the clock input to the DLL or PLL should be on a dedicated GDLLT or GPLLT pin. A PCLK pin can be used but a USE PRIMARY preference must be assigned on this clock route from the PCLK pin to a DLL or PLL.
  - If an interface requires the clock to be routed directly to the ECLK or PCLK tree, the PCLK pin should be used to input the clock. GPLLT or GDLLT pins do not have access to the PCLK or ECLK tree and cannot be used here.
  - If the clocks going to any of the DDR registers are not used in any logic inside the FPGA (like a mux function) this causes it to get on general route.
  - Refer to High-Speed DDR Interface Details to see the clock placement requirements for each type of interface.
- PAR error when assigning a DDR function to a non-DDR I/O pin:
  ERROR - par: Cannot place PIO comp "datain_2" on the proposed PIO site "PB11A / AE4" because the types of their IOLOGICs are incompatible: the associated IOLOGIC comp "datain_2_MGIOL" has been set to "IDDR_OREG" mode (of type IDDRIOL), while the IOLOGIC site is of type XSIOLOGIC and supports FF only.
  In this case an IDDRX2D1 function is placed on the bottom side pin which does not support this function, hence the error.
  See section I/O Logic (IOL) Site Types/Names to see the functions supported on each site type.
- Map Error on IDDRAPPS/ODDRAPPS function:
  ERROR - map: The 'IDDRAPPS' property is missing on IDDR instance 'Inst_IDDRX2D1_1_2'. Each IDDR component targeted for this device needs the 'IDDRAPPS' property identifying the interface being implemented. Refer to DDR usage documentation for details.
  All LatticeECP3 “EA” designs require an IDDRAPPS attribute assigned to the input DDR module and an ODD- RAPPS attribute assigned to the output DDR module. If these attributes are not assigned, then MAP errors out with the error message above. To fix the error, regenerate the module in IPexpress. IPexpress generates the module with all the required input and output DDR attributes.

## 7.8. Timing Analysis for High-Speed DDR Interfaces

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

The External Switching Characteristics section of the LatticeECP3 Family Data Sheet should be used along with this section. The data sheet specifies the actual values for these constraints for each of the interfaces.

### 7.8.1. Frequency Constraints

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

### 7.8.2. DDR Input Setup and Hold Time Constraints

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

#### 7.8.2.1. Receive Centered Interface

Figure 7.2 shows the data and clock relationship for a Receive Centered Interface. The clock is centered to the data, so it comes into the devices with a setup and hold time.

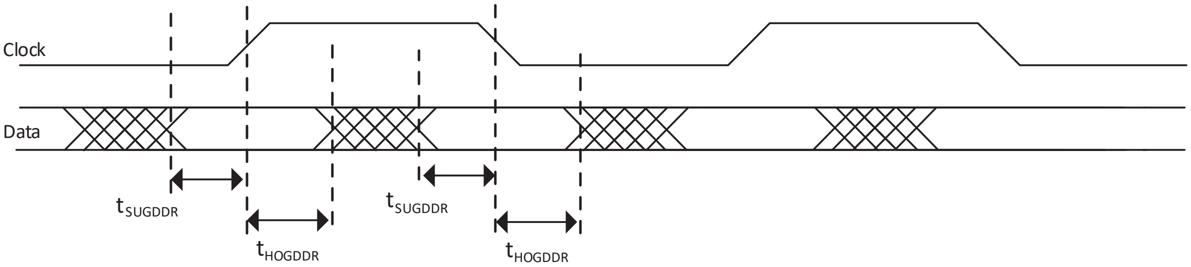


**Figure 7.2 RX Centered Interface Timing**

**Note:** $t_{SUGDDR}$ = Setup Time, $t_{HOGDDR}$ = Hold Time

In this case, you must specify in the software preference the amount of setup and hold time available. These parameters are listed in the figure as tSUGDDR and tHGDDR. These can be directly provided using the INPUT_SETUP and HOLD preference as:

```
INPUT_SETUP PORT "Data" <tSUGDDR> ns HOLD <tHOGDDR> ns CLKPORT "Clock";
```

where:

Data = Input Data Port Clock = Input Clock Port

The External Switching Characteristics section of the LatticeECP3 Family Data Sheet specifies the minimum setup and hold times required for each of the high-speed interfaces running at maximum speed. These values can be picked up from the data sheet if the interface is running at maximum speed.

Example:

For a GDDRX2_RX.ECLK.Centered interface on the left or right sides using a PCLK pin on the LatticeECP3-150EA-8 device when running at the maximum speed of 405MHz, the preference would be:

```
INPUT_SETUP PORT "Data" 0.321 ns HOLD 0.321 ns CLKPORT "Clock";
```

Note: Please check the LatticeECP3 Family Data Sheet for the latest tSUDDR and tHOGDDR numbers.

### 7.8.2.2. Receive Aligned Interface

Figure 7.3 shows the data and clock relationship for a Receive Aligned Interface. The clock is aligned edge-to-edge with the data.

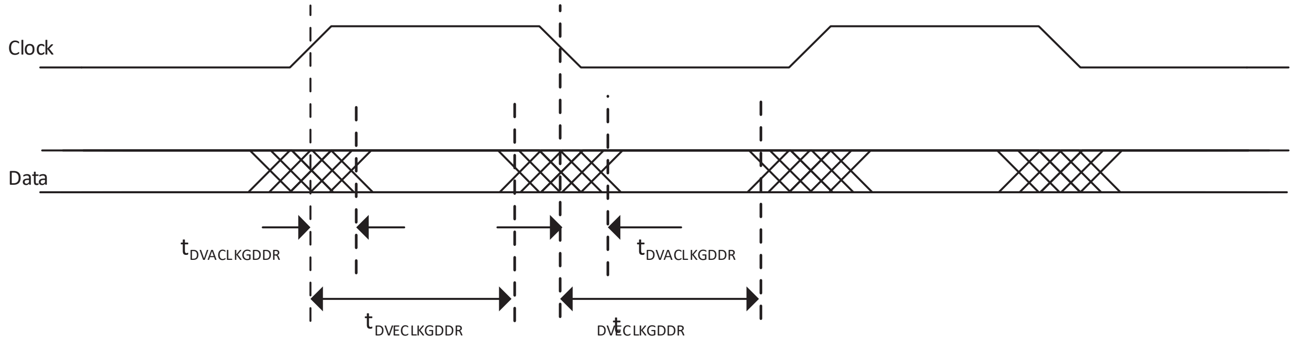


**Figure 7.3 RX Aligned Interface Timing**

**Note:** $t_{DVACLKGDDR}$ = Data Valid after CLK, $t_{DVECLKGDDR}$ = Data Hold After CLK

In this case, worst case data may occur after the clock edge resulting in a negative setup time when entering the device. The worst case setup is specified by the tDVACLKGDDR after the clock edge and the worst case hold time is specified as tDVECLKGDDR. The setup and hold time can be specified as:

```
INPUT_SETUP PORT "Data" <-tDVACLKGDDR > ns HOLD < tDVECLKGDDR> ns CLKPORT
"Clock";
```

where:

Data = Input Data Port

Clock = Input Clock Port

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

The External Switching Characteristics section of the LatticeECP3 Family Data Sheet specifies the minimum $t_{DVACLKGDDR}$ and $t_{DVECLKGDDR}$ values required for each of the high-speed interfaces running at maximum speed. These values can be picked up from the data sheet if the interface is running at maximum speed. The data sheet numbers for this preference are listed in UI (Unit Interface). 1UI is equal to one-half the clock period. Therefore, these numbers needs to be calculated from the clock period used.

Preference Example:

For a GDDRX2_RX.ECLK.Aligned (no CLKDIV) interface on the left or right side using DLLCLKPIN for clock input on a LatticeECP3-150EA-8 device running at the maximum speed of 460MHz (UI = 1.09ns), the preference would be:

$t_{DVACLKGDDR}$ = 0.225UI = 0.25ns, tDVECLKGDDR = 0.775UI = 0.84ns

The preference for this case is:

```
INPUT_SETUP PORT "Data" -0.250000 ns HOLD 0.840000 ns CLKPORT "Clock”;
```

Note: Please check the LatticeECP3 Family Data Sheet for the latest tDVACLKGDDR and tDVECLKGDDR numbers.

#### 7.8.2.3. Receive Dynamic Interfaces

Static Timing Analysis does not show timing for all the dynamic interface cases as the either the clock or data delay is dynamically updated at run time.

### 7.8.3. DDR Clock to Out Constraints for Transmit Interfaces

All of the transmit (TX) interfaces, both x1 and x2, can be constrained with clock-to-out constraints to detect the relationship between the clock and data when leaving the device.

Figure 7.4 shows how the clock-to-out is constrained in the software. Minimum $t_{CO}$ is the minimum time after the clock edge transition that the data does not transition. So any data transition must occur between the $t_{CO}$ minimum and maximum values.

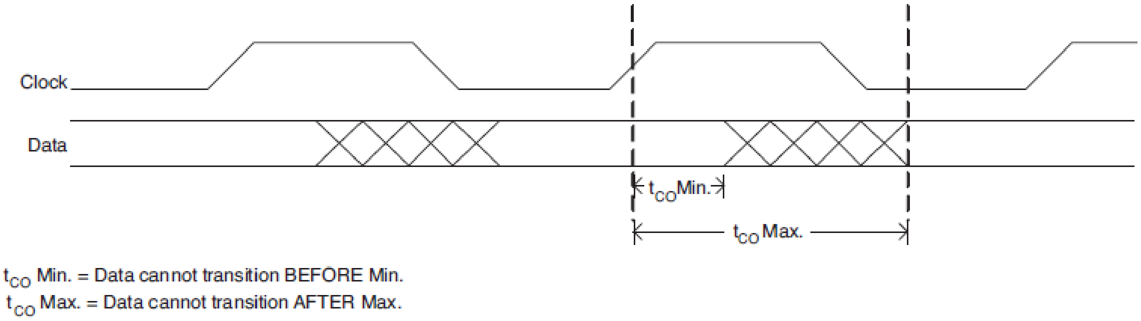

Clock
Data
$t_{CO}$Min.
$t_{CO}$Max.
$t_{CO}$ Min. = Data cannot transition BEFORE Min.
$t_{CO}$ Max. = Data cannot transition AFTER Max.

**Figure 7.4 $t_{CO}$ Minimum and Maximum Timing Analysis**

#### 7.8.3.1. Transmit Centered Interfaces

In this case, the transmit clock is expected to be centered with the data when leaving the device. Figure 7.5 shows the timing for a centered transmit interface.

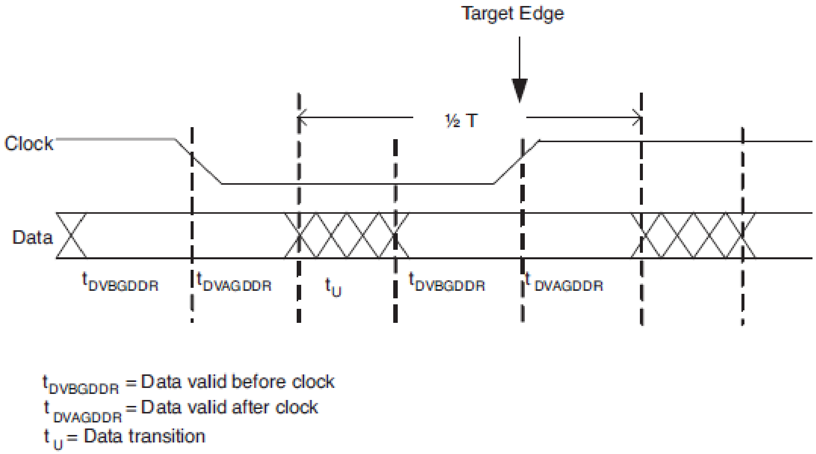


**Figure 7.5 Transmit Centered Interface Timing**

Figure 7.5 shows that the maximum value after which the data cannot transition is $t_{DVBCKGDDR}$. The minimum value before which the data cannot transition is $-(t_{U} + t_{VBCKGDDR})$. A negative sign is used because in this particular case where clock is forwarded centered-aligned to the data, these two conditions occur before the clock edge.

The LatticeECP3 Family Data Sheet specifies the tDVBCKGDDR and tDVACKGDDR values at maximum speed. But we do not know the $t_U$ value, so the minimum $t_{CO}$ can be calculated using the following equation.

$$t_{CO}\text{ Min.} = -(t_{DVBGDDR} + t_U)$$

$$½T = t_{DVAGDDR} + t_{DVBGDDR} + t_U$$

$$-(t_{DVBGDDR} + t_U) = t_{DVAGDDR} - ½T$$

$$t_{CO}\text{ Min.} = t_{DVAGDDR} - ½T$$

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

```
CLOCK_TO_OUT PORT “Data” MAX <-tDVBGDDR> MIN <tDVAGDDR -1/2 Clock Period>
CLKPORT “clk” CLKOUT PORT “Clock”;
```

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The values for tDVBCKGDDR and tDVACKGDDR can be found in the External Switching Characteristics section of the LatticeECP3 Family Data Sheet for the maximum speed.

Preference Example:

For a GDDRX1_TX.SCLK.Centered interface running on the LatticeECP3-150EA-8 device at 250 MHz, $t_{DVBGDDR} = t_{DVAGDDR} =$ 0.670 ns, the preference would be:

```
CLOCK_TO_OUT PORT "Data" MAX -0.670000 ns MIN -1.330000 ns CLKPORT "clk" CLKOUT
PORT "Clock”;
```

**Note:** Check the LatticeECP3 Family Data Sheet for the latest $t_{DVAGDDR}$ and $t_{DVBGDDR}$ numbers.

### 7.8.3.2. Transmit Aligned Interfaces

In this case, the clock and data are aligned when leaving the device. Figure 7.6 shows the timing diagram for this interface.

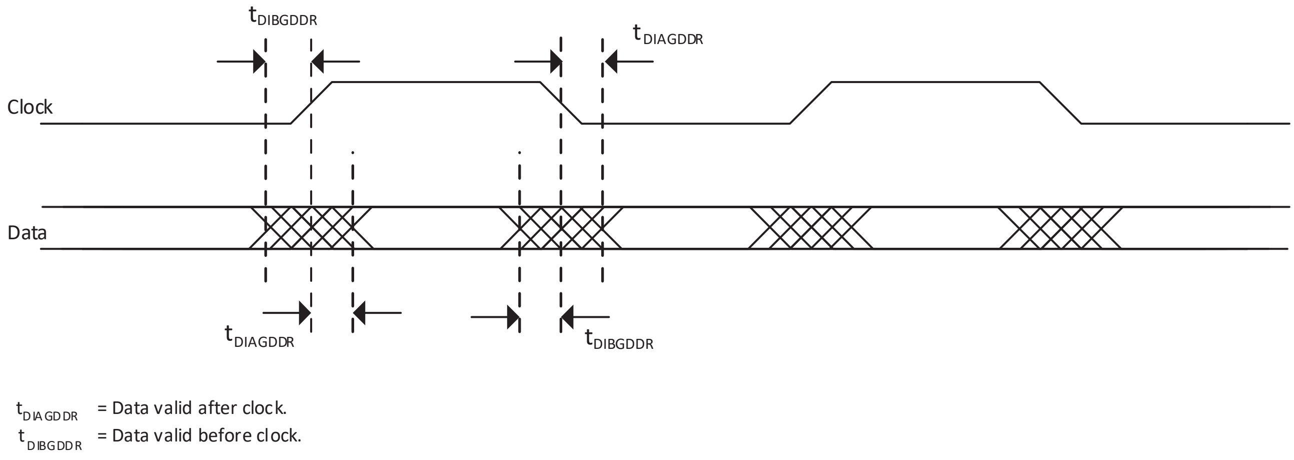

$t_{DIBGDDR}$
$t_{DIAGDDR}$
Clock
Data
$t_{DIAGDDR}$
$t_{DIBGDDR}$
$t_{DIAGDDR}$ = Data valid after clock.
$t_{DIBGDDR}$ = Data valid before clock.

**Figure 7.6 Transmit Aligned Interface Timing**

Figure 7.6 shows that maximum value after which the data cannot transition is $t_{DIAGDDR}$. The minimum value before which the data cannot transition is $t_{DIBGDDR}$. A negative sign is used for the minimum value because the minimum condition occurs before the clock edge.

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

```
CLOCK_TO_OUT PORT “Data” MAX <tDIAGDDR> MIN <-tDIBGDDR> CLKPORT “clk” CLK- OUT
PORT “Clock”;
```

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The $t_{DIAGDDR}$ and $t_{DIBGDDR}$ values are available in the External Switching Characteristics section of the LatticeECP3 Family Data Sheet for maximum speed.

Preference Example:

For a GDDRX2_TX.Aligned interface on the LatticeECP3-150EA-8 device running on the left or right sides at 500 MHz, $t_{DIAGDDR}$ = $t_{DIBGDDR}$ = 0.200 ns. The preference would be:

```
CLOCK_TO_OUT PORT "Data" MAX 0.200000 ns MIN -0.200000 ns CLKPORT "clk" CLKOUT
PORT "Clock";
```

**Note:** Please check the LatticeECP3 Family Data Sheet for the latest $t_{DIAGDDR}$ and $t_{DIBGDDR}$ numbers.

### 7.8.4. Timing Rule Check for Clock Domain Transfers

Clock Domain Transfers within the IDDR and ODDR modules are checked by Trace automatically when these elements are used in a design. Most clock domain transfers occur in the IDDRX2 and ODDRX2 modules where there are fast-speed and slow-speed clock inputs. For IDDRX2, there is a transfer from the fast-speed ECLK to the slow- speed SCLK. On the ODDRX2, the transfer happens from the slow-speed SCLK to the fast-speed ECLK.

For ispLEVER 8.0, no special preferences are needed to run this check. The clock domain transfer checks are automatically done by the software and reported in the Trace report under section called “Timing Rule Check”. The report lists the timing for the both the input and output DDR blocks where a clock domain transfer is done.

Figure 7.7 shows the transfer of data between the ECLK and SCLK for the IDDRX2 block. A cause of concern is the phase relationship between the ECLK and SCLK. As in the waveforms below, these two clocks have to maintain a certain amount of skew to transfer data successfully between the two clocks.

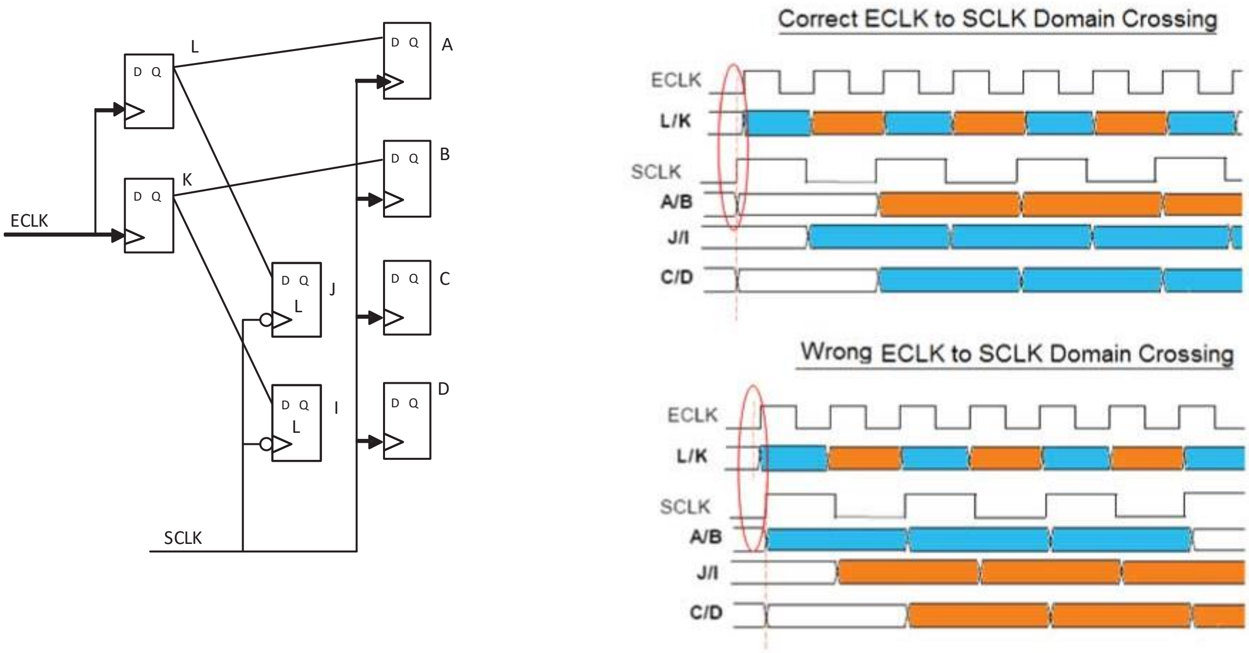

L
K
A
B
C
D
J
I
D Q
ECLK
SCLK
Correct ECLK to SCLK Domain Crossing
ECLK
L/K
SCLK
A/B
J/I
C/D
Wrong ECLK to SCLK Domain Crossing
ECLK
L/K
SCLK
A/B
J/I
C/D

**Figure 7.7 IDDRX2 ECLK to SCLK Transfer**

The skew between the two clocks is specified in terms of “min.” skew and “max.” skew. Figure 7.8 shows how the “min.” and “max.” skew is measured for the IDDRX2 block. It is required that the “min.” and “max.” skew be within the specified “min.” and “max.” specs.

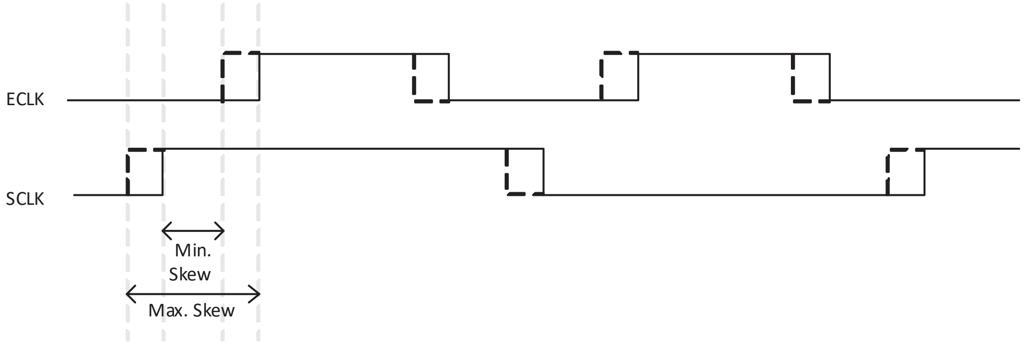


**Figure 7.8 IDDRX2 ECLK to SCLK Skew Calculation**

The following equations are used to check for valid skew between the ECLK and SCLK for an IDDRX2 block.

Max. Skew < - (0 + Internal ECLK to SCLK Setup Time)

Min. Skew > - (ECLK cycle + Internal ECLK to SCLK Hold Time)

The Trace report below shows an example IDDRX2 ECLK to SCLK Timing rule check.

```
Internal Preference: Timing Rule Check
32 items scored, 0 timing errors detected.
----------------------------------------
This section of the Trace report will identify any inherent timing rule violations
in the design. These rules may be independent of preferences.


Passed: din_i_13_MGIOL meets ECLK to CLK skew range from -2.489ns to 0.006ns

Max skew of -1.367ns meets timing requirement of 0.006ns by 1.373ns

Max ECLK:

Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     0.369         M4.PAD to       M4.PADDI clk_i
ROUTE        33     0.509       M4.PADDI to IOL_L43EA.ECLK iddr_inst/buf_clk (to sclk_o_c)
----
0.878   (42.0% logic, 58.0% route), 1 logic levels.

Min CLK:

Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     0.369         M4.PAD to       M4.PADDI clk_i
ROUTE        33     0.476       M4.PADDI to *V_R61C15.CLKI iddr_inst/buf_clk
CLKOUT_DEL  ---     0.353 *V_R61C15.CLKI to *_R61C15.CDIV2 iddr_inst/Inst3_CLKDIVB
ROUTE        33     1.047 *_R61C15.CDIV2 to  IOL_L43EA.CLK sclk_o_c
----
2.245   (32.2% logic, 67.8% route), 2 logic levels.

Min skew of -1.537ns meets timing requirement of -2.489ns by 0.952ns

Min ECLK:

Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     0.423         M4.PAD to       M4.PADDI clk_i
ROUTE        33     0.476       M4.PADDI to IOL_L43EA.ECLK iddr_inst/buf_clk (to sclk_o_c)
----
0.899   (47.1% logic, 52.9% route), 1 logic levels.

Max CLK:

Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     0.423         M4.PAD to       M4.PADDI clk_i
ROUTE        33     0.509       M4.PADDI to *V_R61C15.CLKI iddr_inst/buf_clk
CLKOUT_DEL  ---     0.353 *V_R61C15.CLKI to *_R61C15.CDIV2 iddr_inst/Inst3_CLKDIVB
ROUTE        33     1.151 *_R61C15.CDIV2 to  IOL_L43EA.CLK sclk_o_c
----
2.436   (31.9% logic, 68.1% route), 2 logic levels
```

**Note:** For paths that are common between the ECLK and CLK, the same delay is used. For example, since PADI_DEL is the shared path between the Max. ECLK and Min. CLK for "Max skew calculation" and between Min. ECLK and Max. CLK for "Min skew calculation" the value is the same in both cases.

The "ECLK to CLK skew range from -2.510 ns to -0.019 ns" for this case is an allowable skew range between the two clocks. The skew between the two clocks must fall within this range or the Trace fails on this preference. The allowable skew range is determined by the frequency at which the fast ECLK clock is running.

Similarly for the ODDRX2D, there is an internal transfer from SCLK to DQCLK0/1 which is listed in the Timing Rule Check section as well.

This internal rule check is required for normal data pins but is not required for the forwarded clock output itself where data inputs to the ODDRX2D are constants. There is no internal data transfer in this case. Users can ignore Trace reported errors on this Internal Rule Check only for forwarded clocks. These errors may be blocked by a preference like:

```
BLOCK NET "sclk_c" COMP "clkout_MGIOL";
```

Where sclk_c is the net on the CLK pin of the IOLOGIC component clkout_MGIOL, where clkout is the forwarded clock. Trace violations of this rule check on corresponding data outputs need to be understood and resolved.

### 7.8.5. Preferences for Specific Elements

1. **DLLDELB** – This element is used in to generate a 90° delay in all receive aligned interfaces. The 90° delay is calculated based on the input clock to the TRDLLB element. A frequency preference must be provided on the CLKI of the TRDLLB to allow trace to produce the 90° delay.
2. **DQSBUFx** – The DQSBUF element used in all the DQS interfaces delay the DQSI input by 90°. The 90° delay is calculated from the input clock to the DQSDLL element connected through the DQSDEL signal. A frequency preference must be provided on the CLK input of the DQSDLL to allow trace to produce the 90° delay.

   The DDR Software Primitives and Attributes section provides a detailed description of the DQSBUF elements.

**Note:** Some device I/O pin names end with an "E_A", "E_B", "E_C" or "E_D" (for example, PR43E_B). These pins are "Input Only" pins. These pins can be used only to implement receive interfaces. The clock delay to these pins is ~50 ps longer than the other pins, so you can see some difference in timing between these and other I/O pins.

### 7.8.6. Valid Window Calculation

You can calculate the available valid window using the transmitter device specifications to determine if it is possible to meet the receiver timing requirements at the LatticeECP3 device.

The data sheet numbers can be used to estimate the required valid window at the LatticeECP3 DDR inputs depending on the interface type.

For an Rx centered interface the total data valid window required = $t_{SUGDDR} + t_{HGDDR}$

For an Rx aligned interface the total data valid window required = $t_{DVECLKGDDR} - t_{DVACLKGDDR}$ For example, in the following case the data at the receiver looks like Figure 7.9.

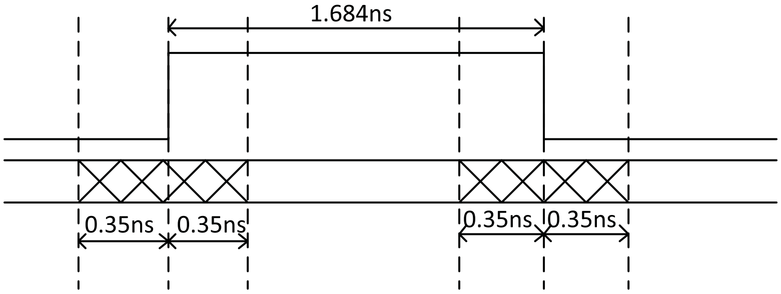


**Figure 7.9 Data at Receiver (Tx Data)**

In this case, the data at the LatticeECP3 input is an aligned interface, with the following timing:

Data Rate = 594 Mbps

$f_{MAX}$ = 297 MHz

Uncertainty around clock edges = 0.350 ns

UI = 1.684 ns

Since this is an aligned interface, we can calculate $t_{DVECLKGDDR}$ and $t_{DVACLKGDDR}$ to determine the available Data Valid Window.

$t_{DVECLKGDDR}$ = 0.350 ns

$t_{DVACLKGDDR}$ = 1.684 ns – 0.350 ns = 1.334 ns

Data Valid Window = $t_{DVECLKGDDR}$ - $t_{DVACLKGDDR}$ = 0.984 ns

On the LatticeECP3 receive side, you must use the GDDRX2_RX.ECLK.Aligned interface given the frequency and data to clock alignment of this interface.

For the GDDRX2_RX.ECLK.Aligned for the -6 speed grade, here are the requirements from the data sheet at 297 MHz.

$t_{DVECLKGDDR}$ = 0.225 UI = 0.379 ns $t_{DVACLKGDDR}$ = 0.775 UI = 1.305 ns

Data Valid Window = $t_{DVECLKGDDR}$ - $t_{DVACLKGDDR}$ = 0.926 ns

**Note:** Refer to the LatticeECP3 Family Data Sheet for the latest timing numbers.

The required data valid window for this example at the LatticeECP3 DDR input is 0.926 ns and the data valid window available on the interface is 0.984 ns. This interface meets the timing requirements at the LatticeECP3 device.

To calculate the worst corner case, the software I/O Timing Analysis should be run to see if the worst case required timing is met. I/O Timing Analysis runs the timing through all the speed grades of the device family and show the worst case results. The Trace Report provides the worst case timing on the same speed grade.

# 8. DDR/DDR2/DDR3 SDRAM Interfaces Overview

The DDR SDRAM interface transfers data at both the rising and falling edges of the clock. The DDR2 is the second generation of the DDR SRDRAM memory, whereas DDR3 is the third generation.

The DDR, DDR2 and DDR3 SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DDR SDRAM interface uses a single-ended DQS strobe signal and the DDR2 and DDR3 interfaces use a differential DQS strobe. The figures below show typical DDR and DDR2/DDR3 SDRAM interface signals. SDRAM interfaces are typically implemented with eight DQ data bits per DQS. So, a x16 interface uses two DQS signals, and each DQS is associated with eight DQ bits. Both the DQ and DQS are bi-directional ports and are used to read and write to the memory.

When reading data from the external memory device, data coming into the device is edge-aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90° before the FPGA logic can sample the read data. When writing to a DDR/DDR2/DDR3 SDRAM, the memory controller (FPGA) must shift the DQS by 90° to center-align with the data signals (DQ). A clock signal is also provided to the memory. This clock is provided as a differential clock (CLKP and CLKN) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read through a DLL inside the memory. Figure 8.1 and Figure 8.2 show DQ and DQS timing relationships for read and write cycles.

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

The state when the DQS is low before it goes into tri-state is the Postamble state. This is the state after the last valid data transition.

DDR SDRAM also requires a Data Mask (DM) signal to mask data bits during write cycles. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface has one strobe signal.

DDR SDRAM interfaces use the SSTL25 Class I/II I/O standards, DDR2 SDRAM interface uses the SSTL18 Class I/II and DDR3 SDRAM interface uses the SSTL15 I/O standards. Both the DDR2 and DDR3 SDRAM interfaces require differential DQS (DQS and DQS#). DDR2 has an option to use either single-ended or differential DQS.

In addition, the DDR3 memory module uses fly-by architecture for the data and strobe signals. This requires the memory controller to support read and write leveling to adjust for leveled delay on read and write data transfers.

**Table 8.1. DDR DDR2 and DDR3 Summary**

| | DDR | DDR2 | DDR3 |
|---|---|---|---|
| Data Rate | 200 to 400 Mbps | 250 Mbps to 532 Mbps | 600 to 800 Mbps |
| DQS | Single-Ended | Single-Ended /Differential | Differential |
| Interface | SSTL25 | SSTL18 | SSTL15 |
| Leveling | No | No | Yes |

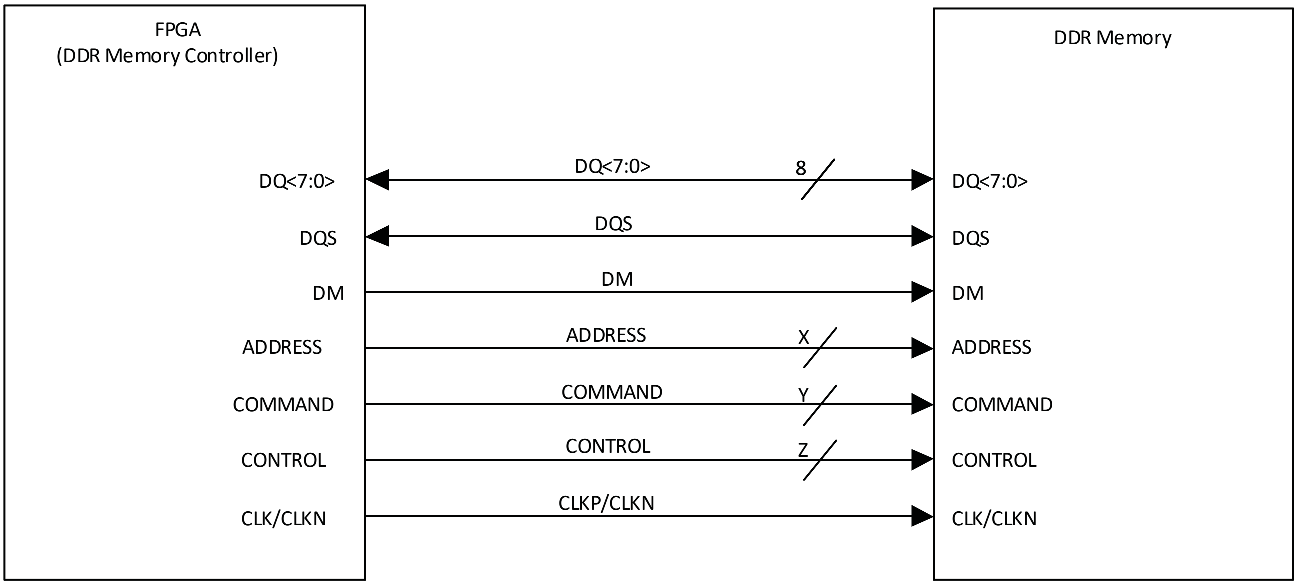


**Figure 8.1 Typical DDR SDRAM Interface**

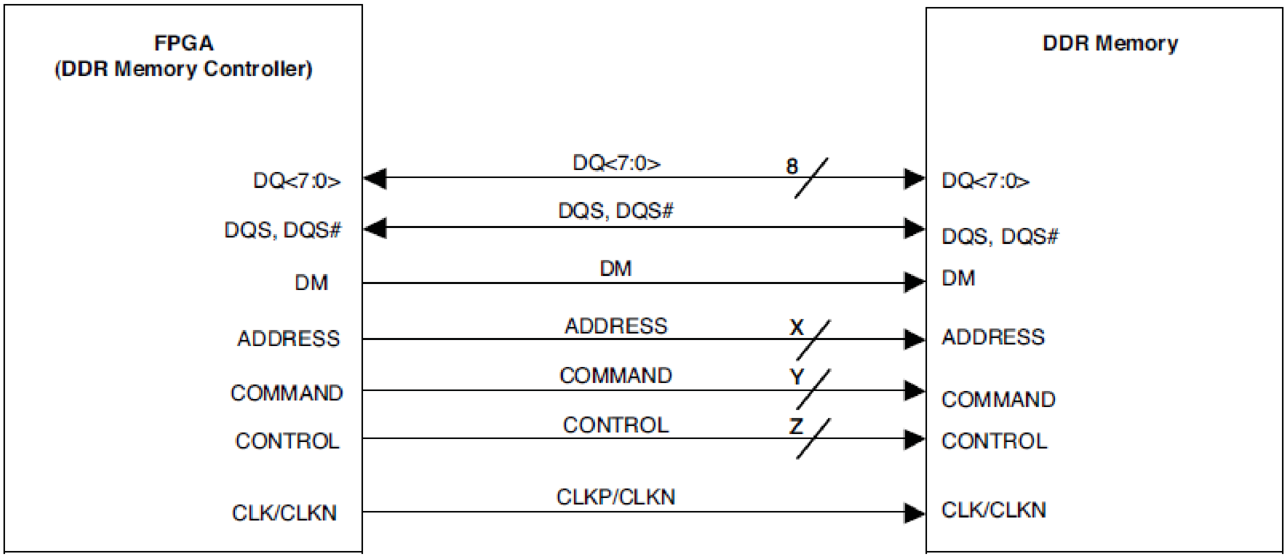


**Figure 8.2 Typical DDR2 and DDR3 SDRAM Interface**

Figure 8.3 and Figure 8.4 show the DQ and DQS relationship for memory read and write interfaces.

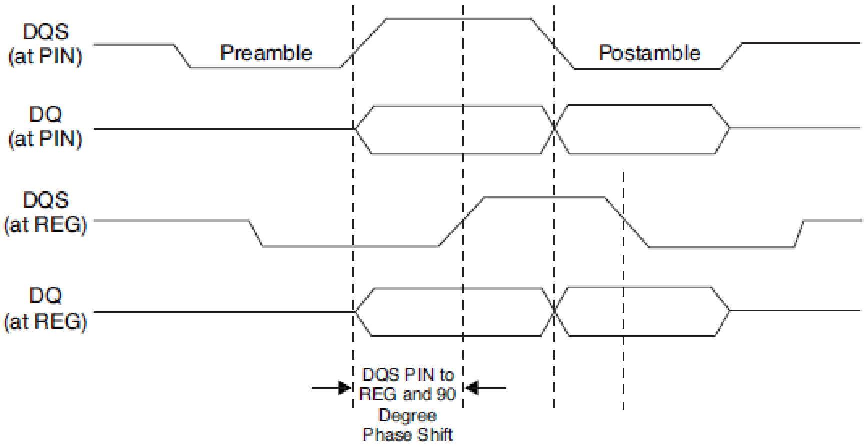


Figure 8.3 DQ-DQS During Read

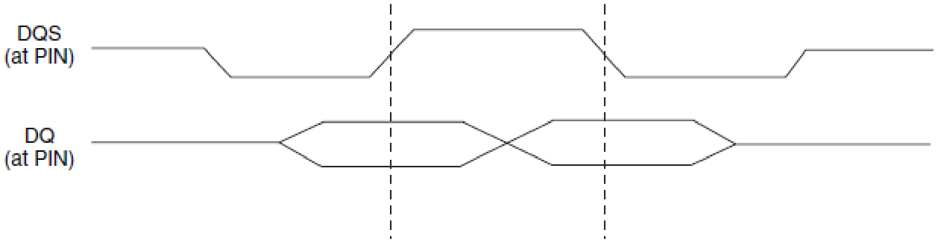


Figure 8.4 DQ-DQS During Write

# 9. Implementing DDR/DDR2/DDR3 Memory Interfaces

As described in the DDR/DDR2/DDR3 SDRAM Interfaces Overview section, all the DDR SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the LatticeECP3 device is edge-aligned with respect to the DQS signal. Therefore, the LatticeECP3 device needs to shift the DQS (90° phase shift) before using it to sample the read data. When writing to a DDR SDRAM from the memory controller, the LatticeECP3 device must generate a DQS signal that is center- aligned with the DQ, the data signals. This is accomplished by ensuring a DQS strobe is 90° shifted relative to the DQ data.

For DDR3 memory, the memory controller also needs to handle the read and write leveling required by the interface due to the newer fly-by topology.

LatticeECP3 devices have dedicated DQS support circuitry for generating appropriate phase-shifting for DQS. The DQS phase shift circuit uses a frequency reference DLL to generate delay control signals associated with each of the dedicated DQS pins, and is designed to compensate for process, voltage and temperature (PVT) variations. The frequency reference is provided through one of the global clock pins. The dedicated DDR support circuit is also designed to provide comfortable and consistent margins for the data sampling window.

This section describes how to implement the read and write sections of a DDR memory interface. It also provides details of the DQ and DQS grouping rules associated with the LatticeECP3 devices.

Both the LatticeECP3 “E” and “EA” devices can be used to implement DDR and DDR2 memory interfaces. DDR3 memory interface can only be implemented on the “EA” devices. There are some differences in the DDR memory implementation between the “EA” and “E” devices. These differences between the devices are indicated in the appropriate sections below.

See the LatticeECP3 DDR3 Memory Controller IP to see how the DDR3 memory interface can be implemented on LatticeECP3 “EA” devices. DDR and DDR2 memory interface implementations are described in the sections below.

## 9.1. DQS Grouping

Each DQS group generally consists of at least 10 I/O (one DQS, eight DQ and one DM) for an 8-bit DDR/DDR2 memory interface or 11 I/O (two DQS, eight DQ, one DM) to implement a complete 8-bit DDR3 memory interface. LatticeECP3 devices support DQS signals on the top, left and right sides of the device.

Each DQS signal spans across 12 I/O. Any 10 (for DDR/DDR2) or 11 (for DDR2/DDR3) of these 12 I/O spanned by the DQS can be used to implement an 8-bit DDR memory interface. In addition to the DQS grouping, you must also assign one reference voltage VREF1 for a given I/O bank.

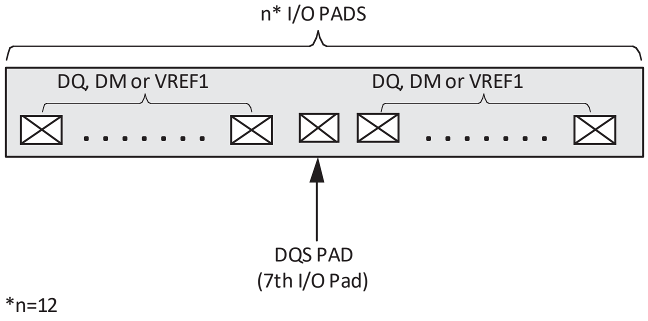


**Figure 9.1 DQ-DQS Grouping**

Figure 9.1 shows a typical DQ-DQS group for LatticeECP3 devices. The seventh I/O of this group of 12 I/O is the dedicated DQS pin. All six pads before of the DQS and five pads after the DQS are covered by this DQS bus span. If using DDR2 or DDR3 memory then the DQS is differential and the eighth pad is used by the DQS# signal. You can assign any eight of the other I/O pads to be DQ data pins. Therefore, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups.

When not interfacing with the memory, the dedicated DQS pin can be used as a general-purpose I/O. Note that the DQS/DQS# pads cannot be used for other DDR functions like DQ, DM or generic DDR. Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pinout information contained in the LatticeECP3 Family Data Sheet shows pin locations for the DQS pads and the corresponding DQ pads.

Some I/O on the left, right and top sides do not support DQS grouping. See the DDR Memory DQ/DQS Design Rules and Guidelines section of this document for further information.

## 9.2. Memory Read Implementation

The LatticeECP3 devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

- DLL-compensated DQS delay elements
- DDR input registers
- Automatic DQS to system/edge clock domain transfer circuitry
- Data Valid Module

## 9.3. DLL-Compensated DQS Delay Elements

The DQS from the memory is connected to the DQS Delay element. The DQS delay block receives a 7-bit delay control from the on-chip DQSDLL. LatticeECP3 devices support two DQSDLL, one on the left and the other on the right side of the device. The DQSDEL generated by the DQSDLL on the left side of the device is routed to all the DQS blocks on the left and top halves of the device. The delay generated by the DQSDLL on the right side of the device is distributed to all the DQS delay blocks on the right side and the top half of the device. There are no DQS pins on the bottom banks of the device. These digital delay control signals are used to delay the DQS from the memory by 90°.

The DQS received from the memory is delayed in each of the DQS delay blocks and this delay DQS is used to clock the first set stage DDR input registers.

## 9.4. Automatic DQS to System/Edge Clock Domain Transfer Circuitry

In a typical DDR memory interface design, the phase relation between the incoming delayed DQS strobe and the internal system clock (during the read cycle) is unknown. Prior to the read operation in DDR memories DQS is in tri-state. Coming out of tri-state, the DDR memory device drives DQS low in the preamble state. The DQS Transition Detect block detects this transition and generates a signal indicating the required polarity for the FPGA system clock (DDRCLKPOL). This signal is used to control the polarity of the clock to the synchronizing registers. For the DDR3 memory interface, this block generates two signals, DDRCLKPOL and DDRLAT. DDRCLKPOL is used to transfer data from the DDR registers to the synchronization registers clocked by ECLK and the DDRLAT signal is used to transfer data from the synchronization registers to the clock transfer registers.

## 9.5. Data Valid Module

The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmit- ted out the input DDR registers to the FPGA core. Note that the DATAVALID signal indicates only the start of valid read data. It is the memory controller’s responsibility to make the proper width of the read data valid signal.

## 9.6. DDR I/O Register Implementation

The first set of DDR registers is used to de-mux the DDR data at the positive and negative edges of the phase- shifted DQS signal. The latch that captures the positive-edge data is followed by a negative-edge triggered register. This register transfers the positive edge data from the first register to the negative edge of DQS so that both the positive and negative portions of the data are now aligned to the negative edge of DQS.

For DDR and DDR2 memory interfaces, the second stage synchronization registers are clocked by the FPGA clock. The polarity of this clock is selected by the DDR Clock Polarity (DDRCLKPOL) signal generated by the DQS Transition Detect Block. The FPGA clock clocks an additional set of clock transfer registers at the end before the data enters the FPGA core.

The DDR3 memory interface uses the gearing feature of the input registers. The synchronization registers are clocked by the fast edge clock (ECLK) input and are then transferred to the clock transfer registers clocked by the slower FPGA clock. The polarity of the ECLK is set by the DDRCLKPOL signal and the FPGA clock (SCLK) polarity is set by the DDRLAT signal.

The LatticeECP3 Family Data Sheet describes each of these circuit elements in more depth.

## 9.7. DDR/DDR2 Memory Read Implementation

The following sections explain the DDR/DDR2 read-ide implementation. LatticeECP3 devices support the DDR/DDR2 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer can generate the different modules required to read the data coming from the DDR/DDR2 memory. See the section DDR Memory Interface Generation Using IPexpress for details on the IPexpress interface. This section explains the read side module generated by IPexpress.

The DDR/DDR2 read side is implemented using the following three software elements. The DQSDLL represents the DLL used for calibration. The IDDRXD implements the input DDR registers. The DQSBUFF represents the DQS delay block, the clock polarity control logic and the Data Valid module. Figure 9.2 shows the read side implementation for both the “E” and “EA” devices. IPexpress should be used to generate this interface. See the section DDR Memory Interface Generation Using IPexpress for details.

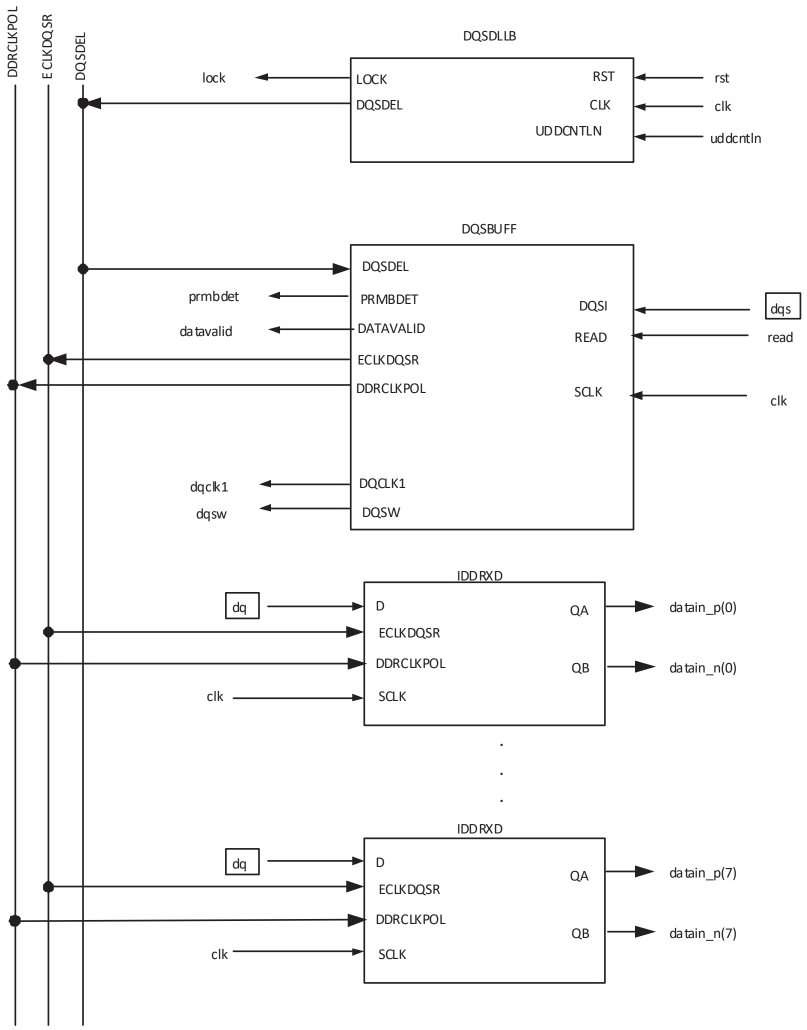


**Figure 9.2 DDR/DDR2 Read Side Implementation for “E” and “EA” Devices**

### 9.7.1. DDR3 Memory Read Implementation

The following sections explain the DDR3 read side implementation. LatticeECP3 devices support the DDR3 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer can generate the different modules required to read the data coming from the DDR3 memory. See the section DDR Memory Interface Generation Using IPexpress for details on the IPexpress interface. This section explains the read side module generated by IPexpress.

The DDR3 memory interface generated in IPexpress also includes a Clock Synchronization Module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality.

The DDR3 read side is implemented using three software elements. The DQSDLL represents the DLL used for calibration. The IDDRX2D implements the input DDR registers in x2 gearing mode required for DDR3 memory. The DQSBUFD represents the DQS delay block, the clock polarity control logic and the Data Valid module.

The ECLK, SCLK2X and SCLK are generated in a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. See the DDR3 Clock Synchronization Module section for details on the Clock Synchronization Module and internal clock generation

Figure 9.3 shows the read side implementation. IPexpress should be used to generate this interface.

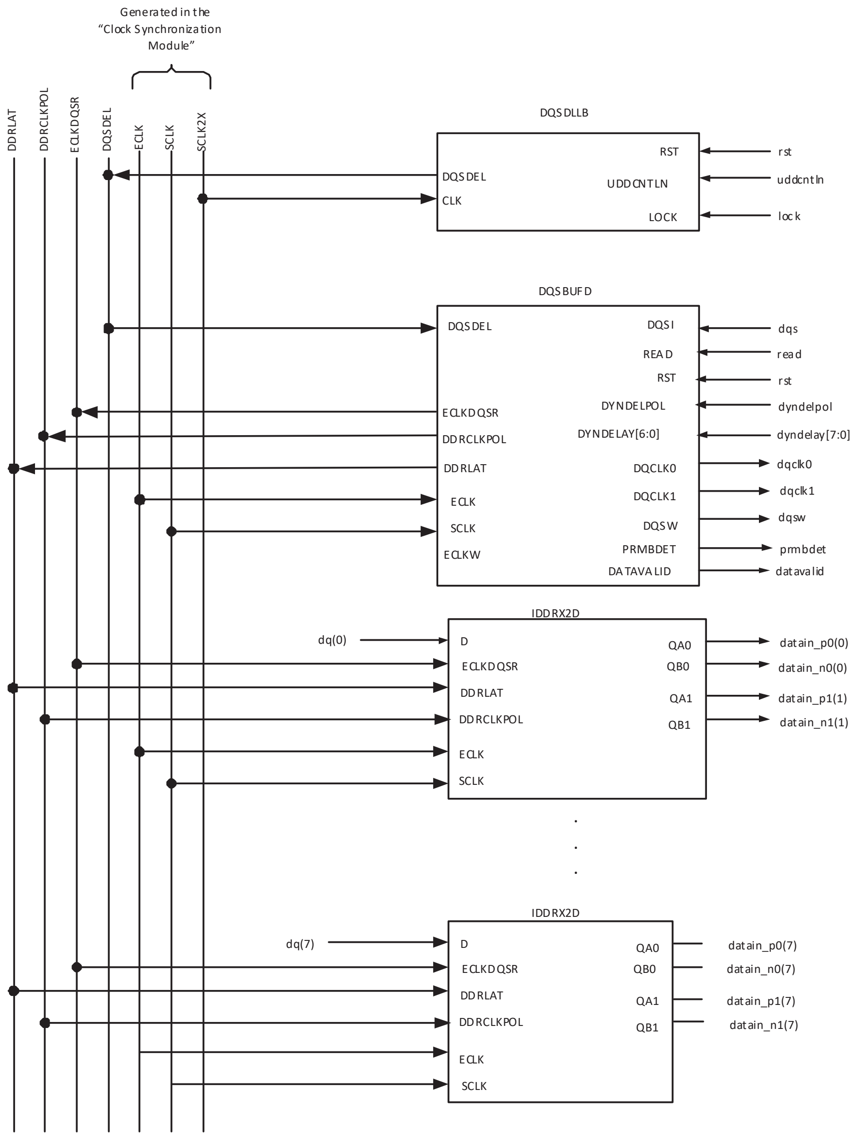


**Figure 9.3 DDR3 Read Side Implementation for "EA" Devices**

## 9.8. Memory Write Implementation

To implement the write portion of a DDR memory interface, two streams of single data rate data must be multi- plexed together with data transitioning on both edges of the clock. In addition, during a write cycle, DQS must arrive at the memory pins center-aligned with data, DQ. Along with the DQS strobe and DQ, CLKP, CLKN, Address/Command and Data Mask (DM) signals also need to be generated. IPexpress should be used to generate this interface. See the section DDR Memory Interface Generation Using IPexpress for details.

Challenges encountered during memory write:

- Differential CLK signals (CLKP and CLKN) need to be generated.
- Generate ADDR/CMD signal edge-aligned to CLKP falling edge.
- DQS needs to be edge-aligned to CLKP. In DDR3 interfaces where fly-by routing is used, write leveling should be used to compensate for skews between CLKP and DQS.
- DQ/DM needs to be center-aligned to DQS and therefore center-aligned to CLKP as well.
- The controller must meet the DDR interface specification for the tDSS and tDSH parameters, defined as DQS falling edge setup and hold time to/from CLKP rising edge, respectively. The skews, if caused by the fly-by topology, are compensated by write-leveling.
- The DDR output data must be muxed from two SDR streams into a single outgoing DDR data stream. For the DDR3 memory interface, this DDR output data is generated from four SDR streams.

The DDR/DDR2 memory write interface is implemented using the following modules:

- IDDRXD, ODDRXD, OFD1S3AX – Used for DDR2 DQ input, output and tri-state control. DDR2 DM uses ODD- RXD and OFD1S3AX only.
- ODDRXDQSA, ODDRTDQSA – Used for DDR2 DQS output and tri-state control.
- DQSBUFF – Dedicated for DDR2 memory interface application.
- DQSBUFG – Generic ODDRXD support for clocks on “E” devices.
- ODDRXD – Generic DDR mode for clocks on “E” devices.
- ODDRXD1 – Generic DDR mode for clocks on “EA” devices.

## 9.9. DDR/DDR2 Internal Clock Generation

LatticeECP3 devices require two clocks to implement a DDR/DDR2 memory interface. SCLK (k_clk) is used to generate the data and data strobe signals while a 270° shifted clock, SCLK1 (k1_clk), is needed to generate the memory clock and address/command signals. Figure 9.4 shows the generation of these clocks using a PLL. SCLK is also used for the read side implementation as described in Figure 9.2.

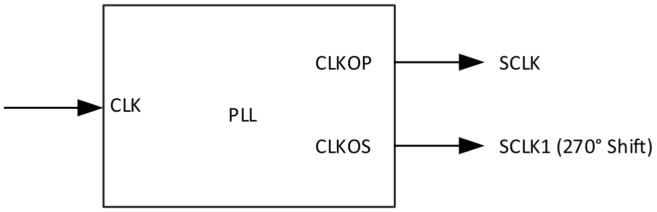


**Figure 9.4 DDR/DDR2 Internal Clock Generation**

## 9.10. DDR/DDR2 Memory Write Implementation

The following sections explain the DDR/DDR2 write side implementation. LatticeECP3 devices support the DDR/DDR2 memory interface function using the DDR memory mode module generated through the IPexpress tool.Using IPexpress, a designer may generate the Data (DQ), Strobe (DQS), Data Mask, Clock (CLKP/CLKN), Address/Command (ADDR/CMD) signals required when writing to the DDR memory. See the section DDR Memory Interface Generation Using IPexpress for details on the IPexpress Interface. This section explains the different Write Side modules generated by IPexpress.

### 9.10.1. DDR/DDR2 Write Side Data (DQ) and Strobe (DQS) Generation

Figure 9.5 shows the DDR2 write side for data and strobe generation. The DQCLK1 and DQSW are signals generated in the DQSBUFF module. DQCLK0 is not used for the DDR2 memory interface. For details on each element, refer to the DDR Software Primitives and Attributes section. The DDR2 memory interface does not require write leveling, therefore the DYNDELAY [6:0] and DYNDELPOL are not used here. The DDR write side uses the same implementation except that the DQS signal in this case is single-ended.

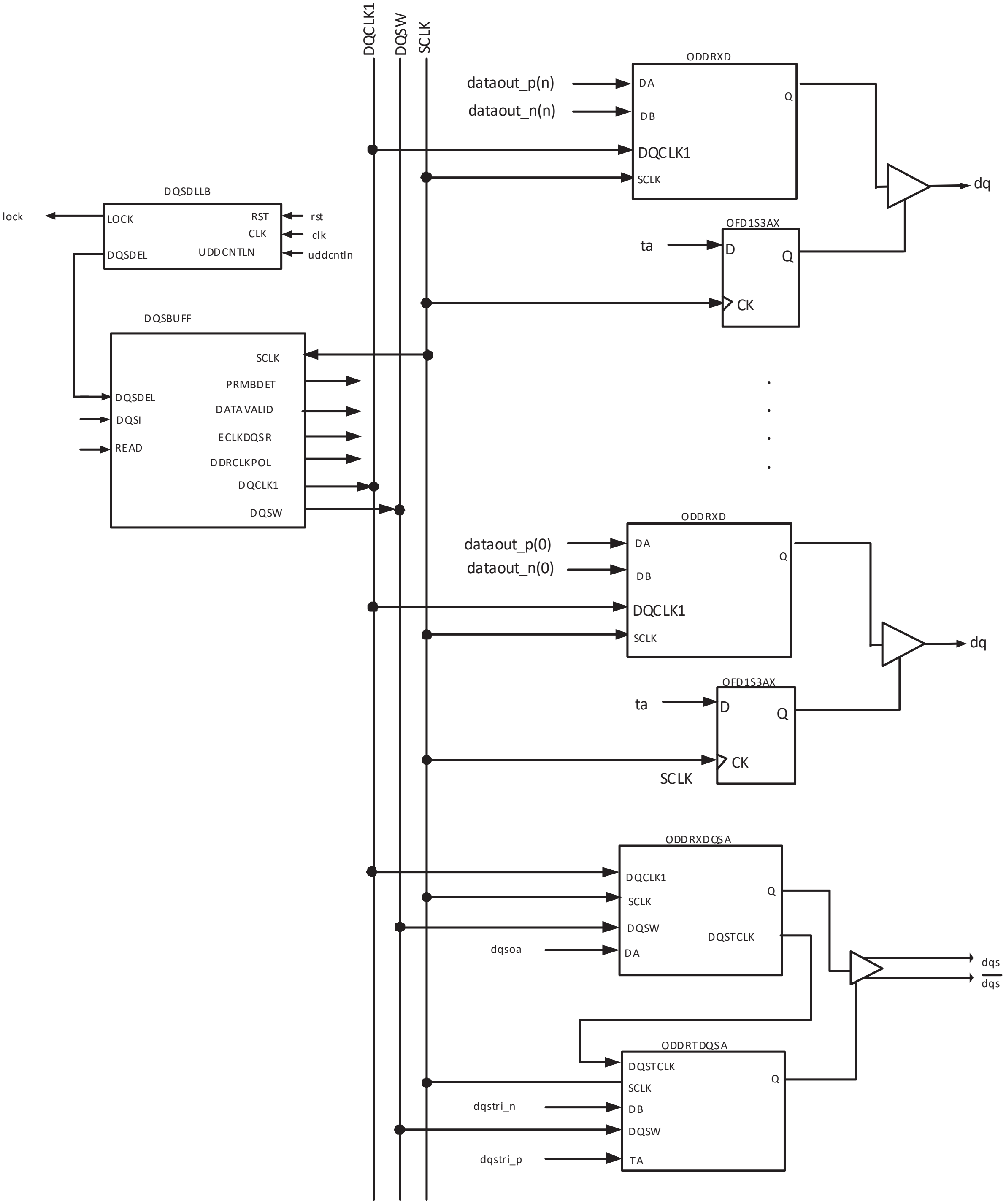


**Figure 9.5 DDR/DDR2 Memory Write DQ and DQS Generation (“E” and “EA” Devices)**

### 9.10.2. DDR/DDR2 Write Side Clock (CLKP/CLKN) Generation

The figures below show the DDR clock output (CLKP/CLKN) generation. The 270° shifted SCLK1 is used to generate the DDR clock outputs. See the section DDR/DDR2 Internal Clock Generation for details. On the LatticeECP3

"E" devices the ODDRXD primitive is used to generate the clocks. The left and right sides of the "E" devices also require the DQSBUFG which is used to generate the DQCLK1 required in the ODDRXD module. The top side of the "E" device does not require the use of DQSBUFG and the DQCLK1 input of the ODDRXD can be tied to SCLK. The "EA" device uses the ODDRXD1 element instead. "EA" devices do not require the use of the DQSBUFG element.

The inputs to the ODDRXD/ODDRXD1 are tied to constant values to generate a clock out of the ODD- RXD/ODDRXD1. When interfacing to the DDR SDRAM memory, CLKP should be connected to the SSTL25D I/O Standard and when interfacing to DDR2 memory it should be connected to SSTL18D I/O Standard to generate the differential clock outputs. Generating the CLKN in this manner prevents skew between the two signals.

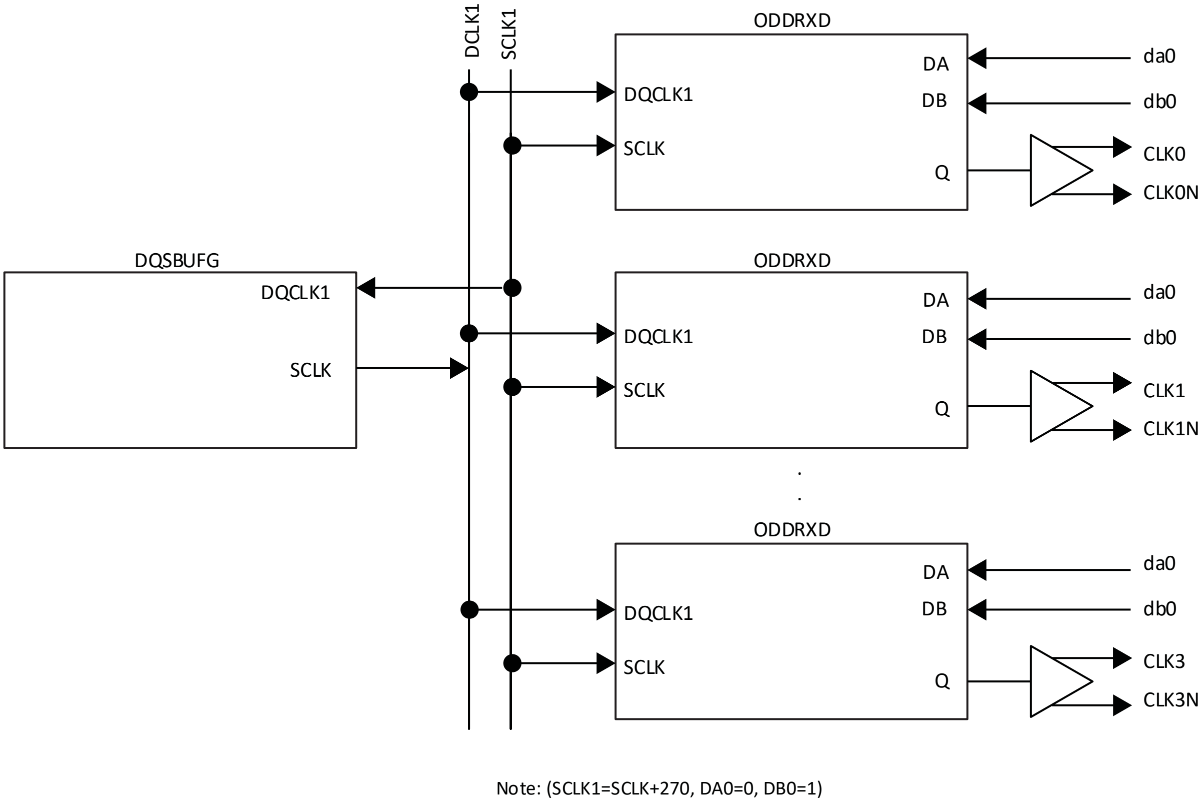
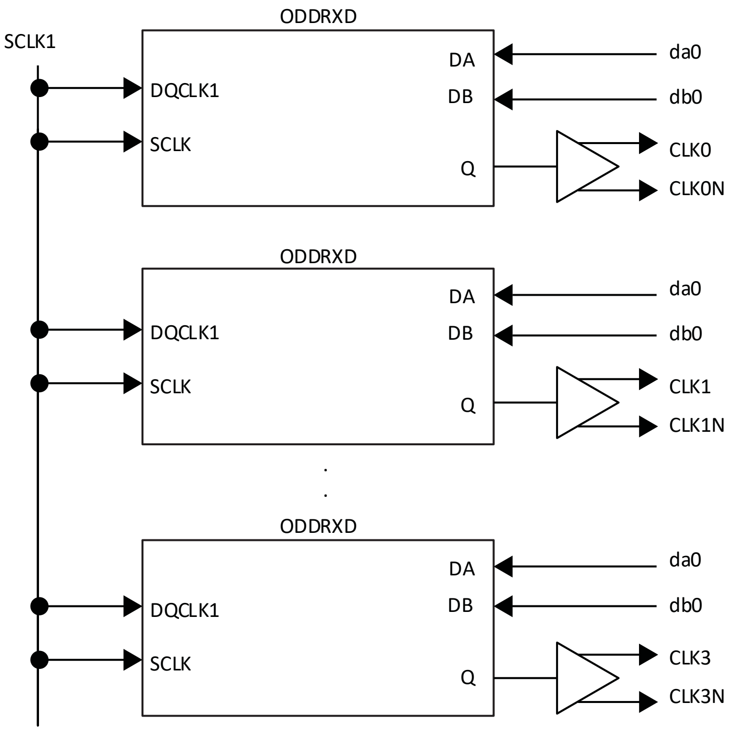
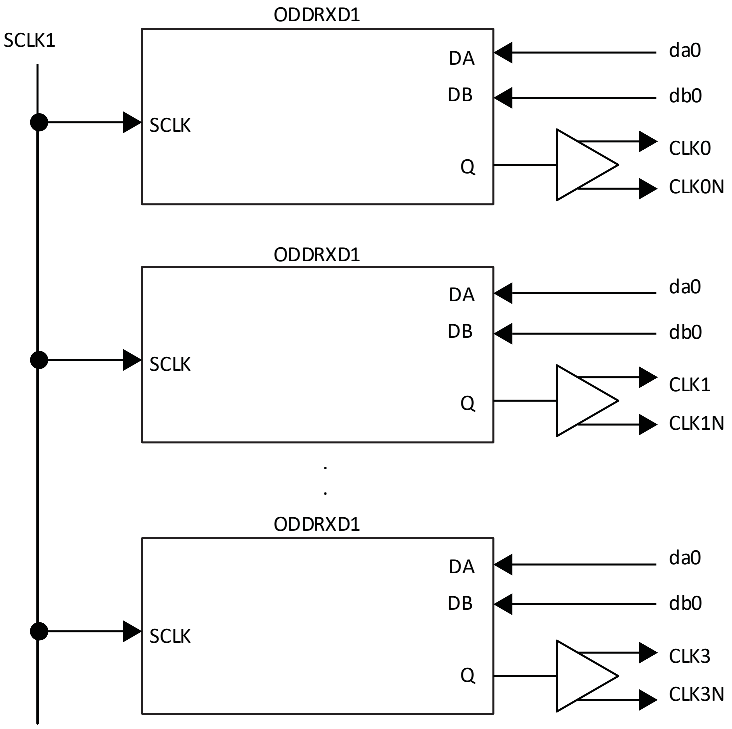


**Figure 9.6 DDR/DDR2 Write Clock Generation ("E" Devices, Left/Right/Top Sides)**



Note: (SCLK1=SCLK+270, DA0=0, DB0=1)

**Figure 9.7 DDR/DDR2 Write Clock Generation (“E” Devices, Top Side)**

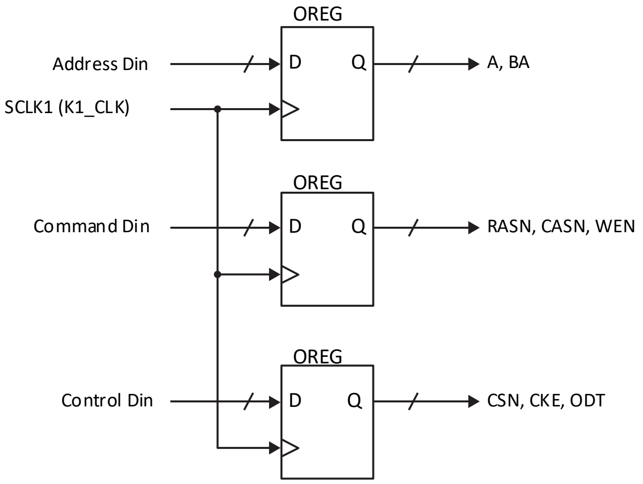


Note: (SCLK1=SCLK+270, DA0=0, DB0=1)

**Figure 9.8 DDR/DDR2 Write Clock Generation (“EA” Devices)**

### 9.10.3. DDR/DDR2 Address/Command Generation

Figure 9.9 shows the address and command generation for the LatticeECP3 “E” and “EA” devices. The 270° shifted SCLK is used for address and command generation. See the section DDR/DDR2 Internal Clock Generation for details. SDR registers are used to generate the address and command signals.



Note: A: Address; BA: Bank Address; RASN: RAS; CASN: CAS; WEN: Write Enable;
CSN: Chip Select; CKE: Clock Enable; ODT: On-Die Termination.

**Figure 9.9 DDR/DDR2 Address / Command Implementation (“E” and “EA” Devices)**

## 9.11. DDR3 Memory Write Implementation

The following sections explain the DDR3 write side implementation. LatticeECP3 devices support the DDR3 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer may generate the Data (DQ), Strobe (DQS), Data Mask, Clock (CLKP/CLKN) and Address/Command (ADDR/CMD) signals required when writing to the DDR3 memory. See the section DDR Memory Interface Generation Using IPexpress for details on the IPexpress Interface. This section explains the different write side modules generated by IPexpress.

The DDR3 memory interface generated in IPexpress also includes a Clock Synchronization Module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality.

### 9.11.1. DDR3 Data (DQ) and Strobe (DQS) Generation

Figure 9.10 shows the DDR3 write side implementation for data (DQ) and strobe (DQS) generation. The DQCLK0, DQCLK1 and DQSW are signals generated in the DQSBUFD module. ODDRX2D implements the output DDR registers in x2 gearing mode required for DDR3 DQ generation. The ODDRX2DQSA module is used to generate the DQS strobe output. For details on each element, refer to the DDR Software Primitives and Attributes section.

The DDR3 memory interface requires write leveling which is supported with the DYNDELAY [6:0] and DYNDELPOL inputs of DQSBUFD. Users can provide the delays for each DQS group using these ports. They are available to you as ports in the top-level module generated by IPexpress.

The ECLK, SCLK2X and SCLK are generated in a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. See the DDR3 Clock Synchronization Module section for details on the Clock Synchronization Module.

IPexpress is used to generate these signals. Figure 9.10 shows the module generated by IPexpress data (DQ) and strobe (DQS) generation.

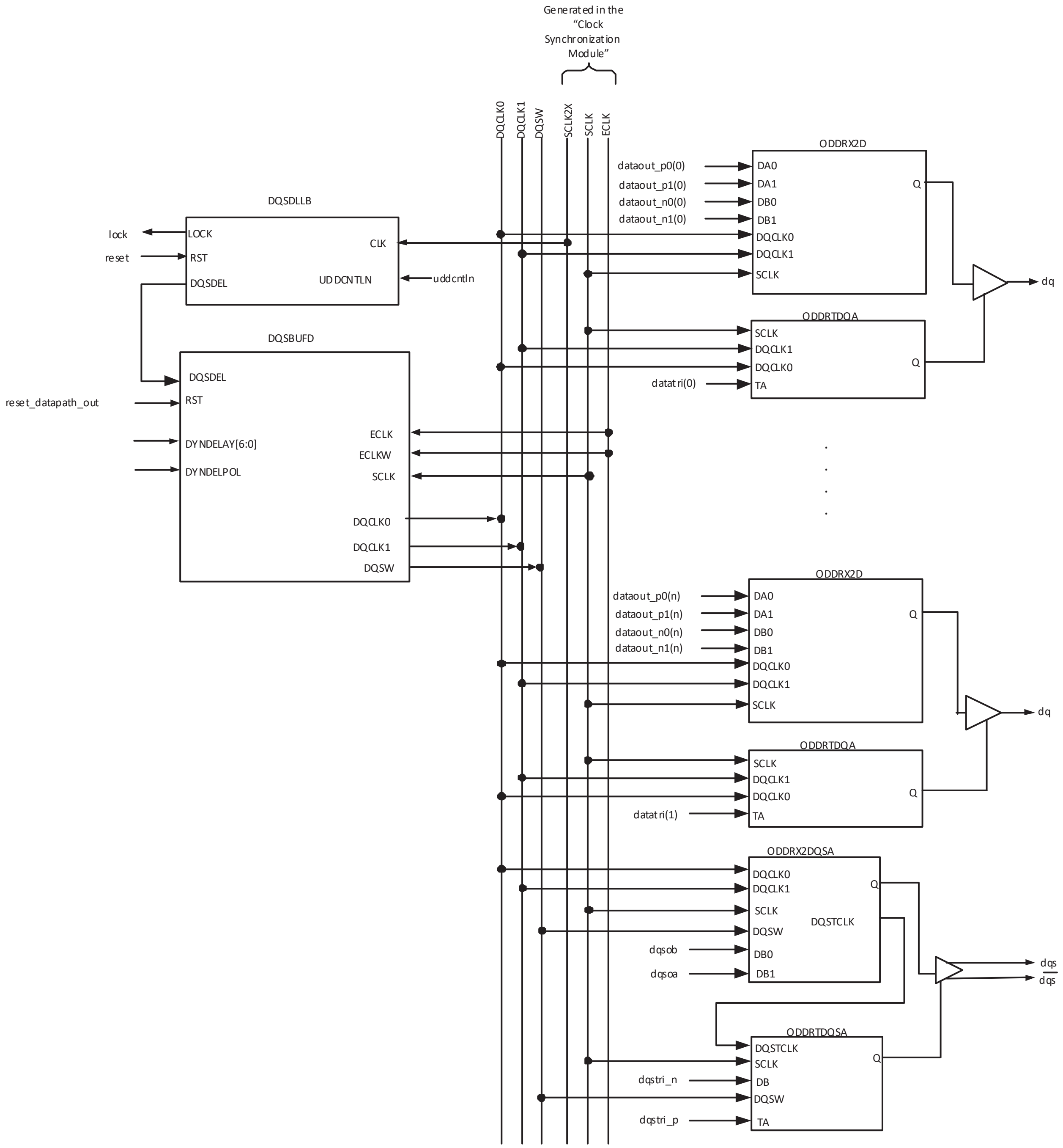


**Figure 9.10 DDR3 Write Side Implementation for DQ and DQS Generation**

### 9.11.2. DDR3 Write Side Clock (CLKP/CLKN) Generation

The SCLK2X out of the Clock Synchronization Module is used to generate the DDR clock outputs. See the section DDR3 Clock Synchronization Module for details.

The ODDRXD1 element is used to generate the clock. The inputs to the ODDRXD1 are tied to constant values to generate a clock out of the ODDRXD1. When interfacing to the DDR3 SDRAM memory, CLKP (ddrclkP) should be connected to the SSTL15D I/O standard to generate the differential clock outputs. Generating the CLKN (ddrclkN) in this manner prevents skew between the two signals

IPexpress is used to generate these signals. Figure 9.11 shows the module generated by IPexpress for DDR clock outputs (CLKP/CLKN).

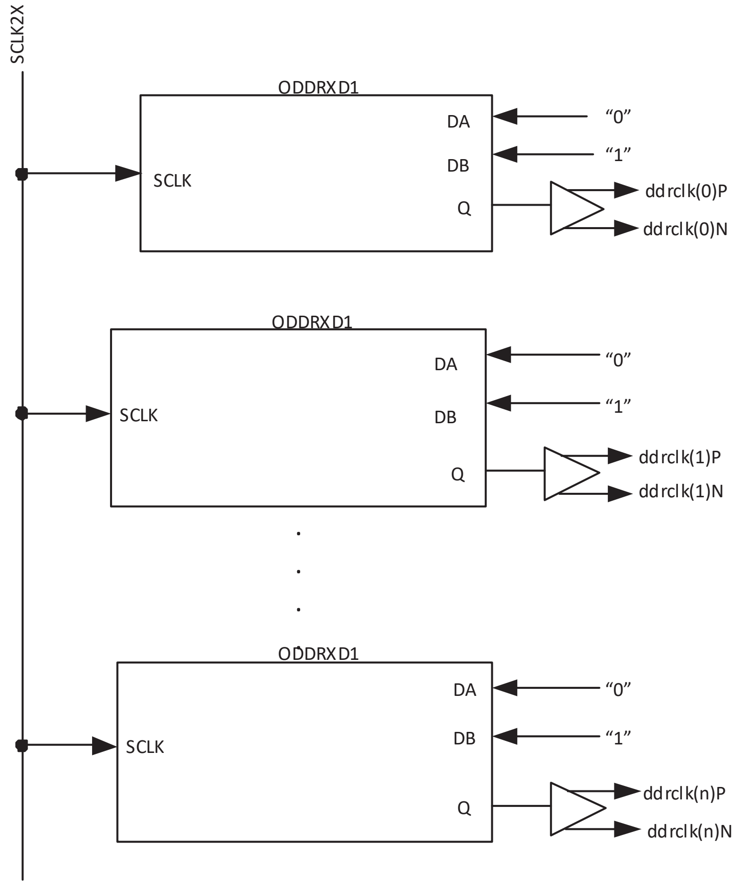


**Figure 9.11 DDR3 Write Side Clock Generation**

### 9.11.3. DDR3 Write Side Address/Command (ADDR/CMD) Generation

The SCLK output of the Clock Synchronization Module, along with the ODDRXD1 element, is used for address and command generation. See the section DDR3 Clock Synchronization Module for details on how the SCLK is generated.

IPexpress is used to generate these signals. Figure 9.12 shows the module generated by IPexpress for address and command signals.

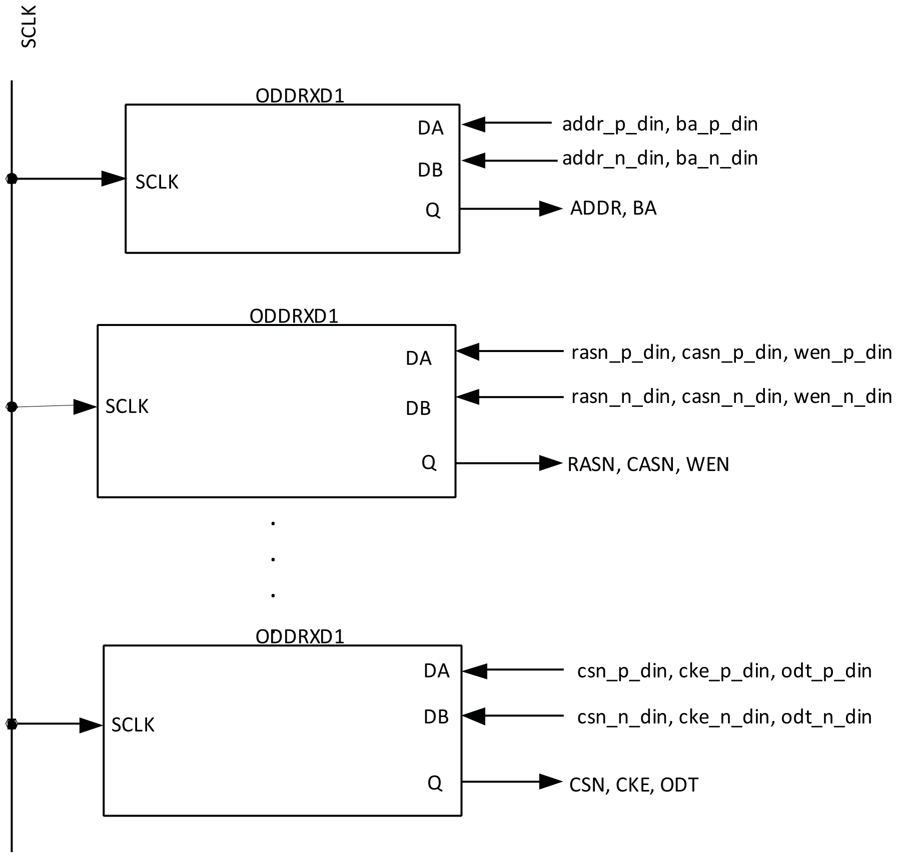

SCLK
ODDRXD1
SCLK
DA
DB
Q
addr_p_din, ba_p_din
addr_n_din, ba_n_din
ADDR, BA
ODDRXD1
SCLK
DA
DB
Q
rasn_p_din, casn_p_din, wen_p_din
rasn_n_din, casn_n_din, wen_n_din
RASN, CASN, WEN
ODDRXD1
SCLK
DA
DB
Q
csn_p_din, cke_p_din, odt_p_din
csn_n_din, cke_n_din, odt_n_din
CSN, CKE, ODT

**Figure 9.12 DDR3 Write Side Address/Command Generation**

## 9.12. DDR3 Clock Synchronization Module

The DDR3 memory interface generated in IPexpress includes a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. A DDR3 memory interface that is implemented in LatticeECP3 must use the CSM block for correct read data transfer from the ECLK to SCLK domain. It is also used to provide the proper clock relationship between SCLK and DQCLK0/1 used for write data generation.

The CSM block generates the ECLK, SCLK2X and SCLK clocks to be used in the read and write side implementations. It generates a clocking_good output to indicate that all the clocks for DDR3 operation have been synchronized and the device is ready to serve the DDR3 operation. The DDR3 memory interface and controller should wait until the clocking_good signal is detected as high.

The DDR3 mode supports the write leveling feature by providing the dynamic delay control ports inputs (DYNDE- LAY) of DQSBUFD to you. In order for the CSM block to support write leveling, the reset_datapath input port should be asserted high for at least one PLL input clock cycle (pll_clk) to reset the DQSBUFD blocks through the reset_datapath_out signal after a write leveling operation is finished. The reset_datapath_out signal is directly connected to the reset (RST) port of all DQSBUFD.

Figure 9.13 shows the block diagram of the CSM block.

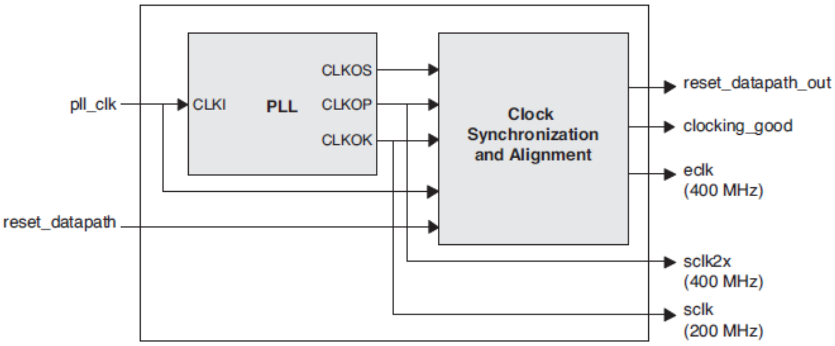


**Figure 9.13 DDR3 Clock Synchronization Module**

PLL Settings for Clock Synchronization Module: LatticeECP3 devices allow two dedicated PLL blocks per side to support the DDR3 memory interface applications. One of the four available PLLs (two PLLs in the left side and two PLLs in the right side) must be used to implement the CSM block. Since the PLL input clock is also used to control the synchronization logic, the use of a PLL input clock that is too fast may cause failures in PAR timing results. Therefore, it is recommended to use the 1:2:4 clock multiplication ratios for high-speed DDR3 memory interface applications as shown in the example of a 400 MHz/800 Mbps DDR3 memory interface below. A dedicated PLL input clock pad or a PCLK pad can be used to drive the selected PLL.

**Table 9.1. PLL Settings for 400 MHz DDR3 Operation**

| | PLL Pin Name | Clock Output | Speed (400MHz) | Speed (300MHz) |
|---|---|---|---|---|
| Input Clock | CLKI | pll_clk | 100 MHz | 75 MHz |
| Output Clocks | CLKOP | sclk2x | 400 MHz | 300 MHz |
| | CLKOS | eclk | 400 MHz | 300 MHz |
| | CLKOK | sclk | 200 MHz | 150 MHz |
| PLL Divider Setting | CLKOK_DIV=2, CLKOP_DIV=2, CLKFB_DIV=4, CLKI_DIV=1 | | | |

Timing Preferences for Clock Synchronization Module: Since the CSM block includes a few timing-sensitive nets that affect the performance of DDR3 I/O functions, the route delays on these nets should be tightly controlled. To consistently guarantee successful routing results, these nets should be constrained by the MAXDDELAY preferences. The nets shown in Table 9.2 include the net delay preferences to force the software Place and Route (PAR) tool to meet the required net delays.

**Table 9.2. MAXDELAY NET Preferences for Clock Synchronization Module**

| Preference | Net Name | -8 Device | -7 Device | -6 Device |
|---|---|---|---|---|
| MAXDELAY NET | eclk | 1.2 ns | 1.3 ns | 1.45 ns |
| MAXDELAY NET | stop | 0.8 ns | 0.85 ns | 0.9 ns |
| MAXDELAY NET | clkos (PLL 1) | 1.1 ns | 1.2 ns | 1.35 ns |
| MAXDELAY NET | clkos (PLL 2) | 0.65 ns | 0.75 ns | 0.8 ns |
| MAXDELAY NET | dqclk1bar_ff | 0.65 ns | 0.7 ns | 0.75 ns |

The preferences for the constrained PAR targets are located in the HDL module generated by IPexpress. It is automatically transferred to the project preference file (.prf) to achieve the optimum timing results.

The example shown below includes the target preferences with the following conditions:

- DDR3 module's instance name is "u_ddr3"
- Speed grade -8 device is targeted
- PLL location 1 is selected

```
MAXDELAY NET "[path]/eclk" 1.200000 nS;
MAXDELAY NET "[path]/u_ddr3/stop" 0.800000 nS;
MAXDELAY NET "[path]/u_ddr3/clkos" 1.100000 nS;
MAXDELAY NET "[path]/u_ddr3/Inst8_clk_phase/dqclk1bar_ff" 0.650000 nS;
```

Locate Preferences for Clock Synchronization Module: The CSM block also requires several manual locations for the clock resources in the module like PLL, ECLKSYNC etc. These modules are all generated by IPexpress and include an UGROUP attribute added to the modules. The corresponding PGROUP (physical grouping) should be locked to the specific sites as shown in Table 9.3.

**Table 9.3. DDR3 Clock and PGROUP Locations for CSM Support**

| | **LatticeECP3-150EA 1156** | | | | **LatticeECP3-150EA 672** | | | |
|---|---|---|---|---|---|---|---|---|
| | Left 1 | Left 2 | Right 1 | Right 2 | Left 1 | Left 2 | Right 1 | Right 2 |
| PLL input clock pad | U6 | Y9 | V34 | Y28 | M3 | U4 | T21 | V20 |
| PLL | R61C5 | R79C5 | R61C178 | R79C178 | R61C5 | R79C5 | R61C178 | R79C178 |
| ECLKSYNC | L2 | L1 | R2 | R1 | L2 | L1 | R2 | R1 |
| PGROUP clk_phase0 | R50C5D | R78C5D | R50C178D | R78C178D | R50C5D | R78C5D | R50C178D | R78C178D |
| PGROUP clk_phase1a | R60C2D | R60C2D | R60C181D | R60C181D | R60C2D | R60C2D | R60C181D | R60C181D |
| PGROUP clk_phase1b | R60C2D | R60C2D | R60C181D | R60C181D | R60C2D | R60C2D | R60C181D | R60C181D |
| PGROUP clk_stop | R60C2D | R60C2D | R60C180D | R60C180D | R60C2D | R60C2D | R60C180D | R60C180D |
| | **LatticeECP3-95/70EA 1156** | | | | **LatticeECP3-95/70EA 672** | | | |
| | Left 1 | Left 2 | Right 1 | Right 2 | Left 1 | Left 2 | Right 1 | Right 2 |
| PLL input clock pad | U6 | Y9 | V34 | Y28 | M3 | U4 | T21 | V20 |
| PLL | R43C5 | R61C5 | R43C142 | R61C142 | R43C5 | R61C5 | R43C142 | R61C142 |
| ECLKSYNC | L2 | L1 | R2 | R1 | L2 | L1 | R2 | R1 |
| PGROUP clk_phase0 | R32C5D | R60C5D | R32C142D | R60C142D | R32C5D | R60C5D | R32C142D | R60C142D |
| PGROUP clk_phase1a | R42C2D | R42C2D | R42C145D | R42C145D | R42C2D | R42C2D | R42C145D | R42C145D |
| PGROUP clk_phase1b | R42C2D | R42C2D | R42C145D | R42C145D | R42C2D | R42C2D | R42C145D | R42C145D |
| PGROUP clk_stop | R42C2D | R42C2D | R42C144D | R42C144D | R42C2D | R42C2D | R42C144D | R42C144D |
| | **LatticeECP3-95/70EA 484** | | | | **LatticeECP3-35EA 672** | | | |
| | Left 1 | Left 2 | Right 1 | Right 2 | Left 1 | Left 2 | Right 1 | Right 2 |
| PLL input clock pad | L5 | T3 | M18 | R17 | M3 | U4 | T21 | V20 |
| PLL | R43C5 | R61C5 | R43C142 | R61C142 | R35C5 | R53C5 | R35C70 | R53C70 |
| ECLKSYNC | L2 | L1 | R2 | R1 | L2 | L1 | R2 | R1 |
| PGROUP clk_phase0 | R32C5D | R60C5D | R32C142D | R60C142D | R24C5D | R52C5D | R24C70D | R52C70D |
| PGROUP clk_phase1a | R42C2D | R42C2D | R42C145D | R42C145D | R34C2D | R34C2D | R34C73D | R34C73D |
| PGROUP clk_phase1b | R42C2D | R42C2D | R42C145D | R42C145D | R34C2D | R34C2D | R34C73D | R34C73D |
| PGROUP clk_stop | R42C2D | R42C2D | R42C144D | R42C144D | R34C2D | R34C2D | R34C72D | R34C72D |
| | **LatticeECP3-35EA 484** | | | | **LatticeECP3-35EA 256** | | | |
| | Left 1 | Left 2 | Right 1 | Right 2 | Left 1 | Left 2 | Right 1 | Right 2 |
| PLL input clock pad | L5 | T3 | M18 | R17 | K3 | P2 | K14 | T15 |
| PLL | R35C5 | R53C5 | R35C70 | R53C70 | R35C5 | R53C5 | R35C70 | R53C70 |
| ECLKSYNC | L2 | L1 | R2 | R1 | L2 | L1 | R2 | R1 |
| PGROUP clk_phase0 | R24C5D | R52C5D | R24C70D | R52C70D | R24C5D | R52C5D | R24C70D | R52C70D |
| PGROUP clk_phase1a | R34C2D | R34C2D | R34C73D | R34C73D | R34C2D | R34C2D | R34C73D | R34C73D |
| PGROUP clk_phase1b | R34C2D | R34C2D | R34C73D | R34C73D | R34C2D | R34C2D | R34C73D | R34C73D |
| PGROUP clk_stop | R34C2D | R34C2D | R34C72D | R34C72D | R34C2D | R34C2D | R34C72D | R34C72D |
| | **LatticeECP3-17EA 484** | | | | **LatticeECP3-17EA 256** | | | |
| | Left 1 | Left 2 | Right 1 | Right 2 | Left 1 | Left 2 | Right 1 | Right 2 |
| PLL input clock pad | L5 | NA | M18 | NA | K3 | NA | K14 | NA |
| PLL | R26C5 | NA | R26C52 | NA | R26C5 | NA | R26C52 | NA |
| ECLKSYNC | L2 | NA | R2 | NA | L2 | NA | R2 | NA |
| PGROUP clk_phase0 | R15C5D | NA | R15C52D | NA | R15C5D | NA | R15C52D | NA |
| PGROUP clk_phase1a | R25C2D | NA | R25C55D | NA | R25C2D | NA | R25C55D | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PGROUP clk_phase1b | R25C2D | NA | R25C55D | NA | R25C2D | NA | R25C55D | NA |
| PGROUP clk_stop | R25C2D | NA | R25C54D | NA | R25C2D | NA | R25C54D | NA |

The PLL input pad, the corresponding PLL site and the legal ECLKSYNC site for the selected PLL are constrained in the user preference file (.lpf) as shown in the example below:

```
LOCATE COMP "clk_in” SITE "U6";
LOCATE COMP "[path]/u_ddr3/Inst1_EHXPLLF" SITE "PLL_R61C5";
LOCATE COMP "[path]/u_ddr3/Inst6_ECLKSYNCA" SITE "LECLKSYNC2";
```

Locating the defined PGROUPs is crucial for successful DDR3 operation. The PGROUPs should be manually located according to the table as shown below:

```
LOCATE PGROUP "clk_phase0" SITE "R59C3D";
LOCATE PGROUP "clk_phase1" SITE "R59C2D";
LOCATE PGROUP "clk_stop" SITE "R60C2D";
LOCATE PGROUP "rst_dp_out" SITE "R60C4D";
```

The following block paths should also be included to help avoid unnecessary domain crossing nets being reported in the trace report as false alarms.

[False paths for PAR and TRACE]

```
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "sclk" ;
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "*/clkos" ;
BLOCK PATH FROM CLKNET "sclk" TO CLKNET "pll_clk" ;
BLOCK PATH FROM CLKNET "*sclk2x" TO CLKNET "pll_clk" ;
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "*eclk" ;
BLOCK PATH FROM CLKNET "*/clkos" TO CLKNET "*eclk" ;
BLOCK PATH FROM CLKNET "*/clkos" TO CLKNET "sclk" ;
BLOCK PATH FROM CLKNET "*sclk2x" TO CLKNET "*/clkos" ;
```

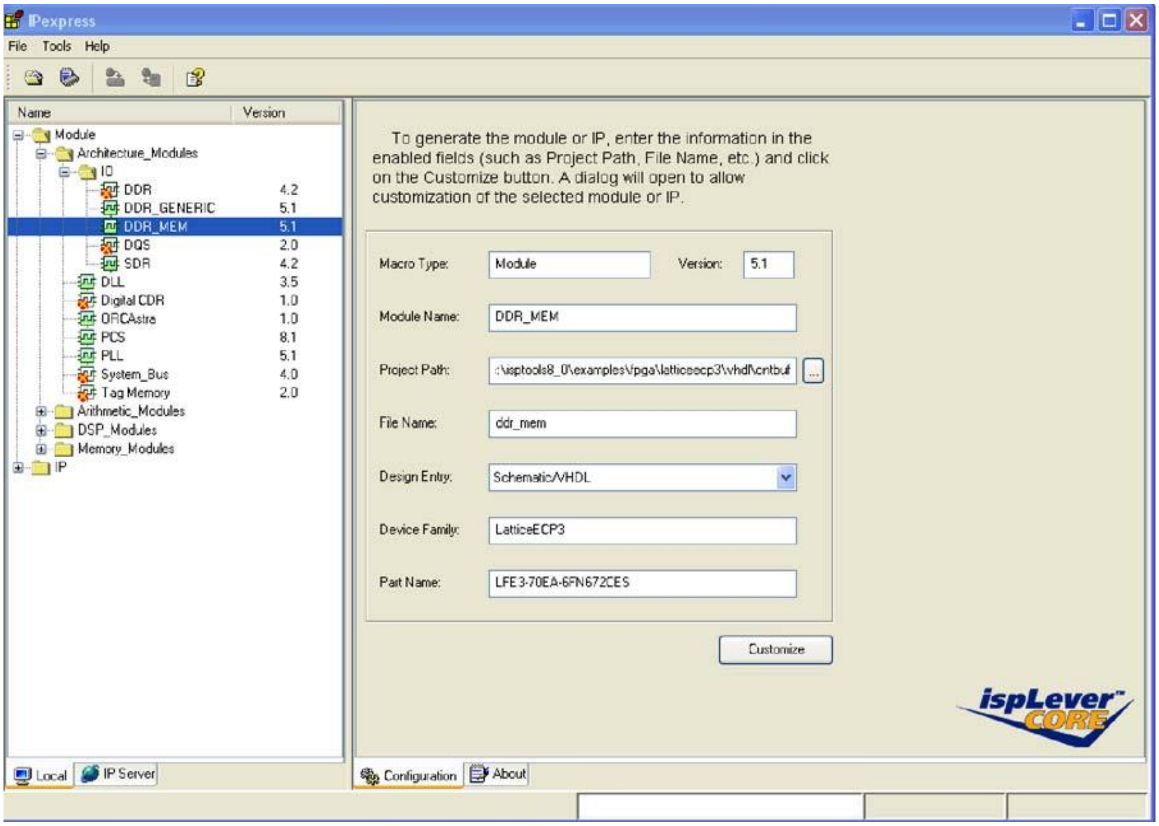
**Note:** The clock net names and hierarchy path names may be changed after synthesis. If this happens, updates on the preferences to follow the changed names are required.

# 10. DDR Memory Interface Generation Using IPexpress

The IPexpress tool is used to configure and generate the DDR, DDR2 and DDR3 memory interfaces. This section assumes that ispLEVER 8.0 SP1 is used for generation of the interfaces. If you are using ispLEVER 7.2 SP2, see Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2. If you are using Lattice Diamond design software, see Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond.

To see the detailed block diagram for each interface generated by IPexpress see the Memory Read Implementation and Memory Write Implementation sections.

IPexpress can be opened from the Tools menu in Project Navigator. All the DDR modules are located under Architecture Modules > IO. DDR_MEM is used to generate DDR memory interfaces.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.2 |
| DDR_GENERIC | 5.1 |
| DDR_MEM | 5.1 |
| DQS | 2.0 |
| SDR | 4.2 |
| DLL | 3.5 |
| Digital CDR | 1.0 |
| ORCAstra | 1.0 |
| PCS | 8.1 |
| PLL | 5.1 |
| System_Bus | 4.0 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| DSP_Modules | |
| Memory_Modules | |
| IP | |

To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

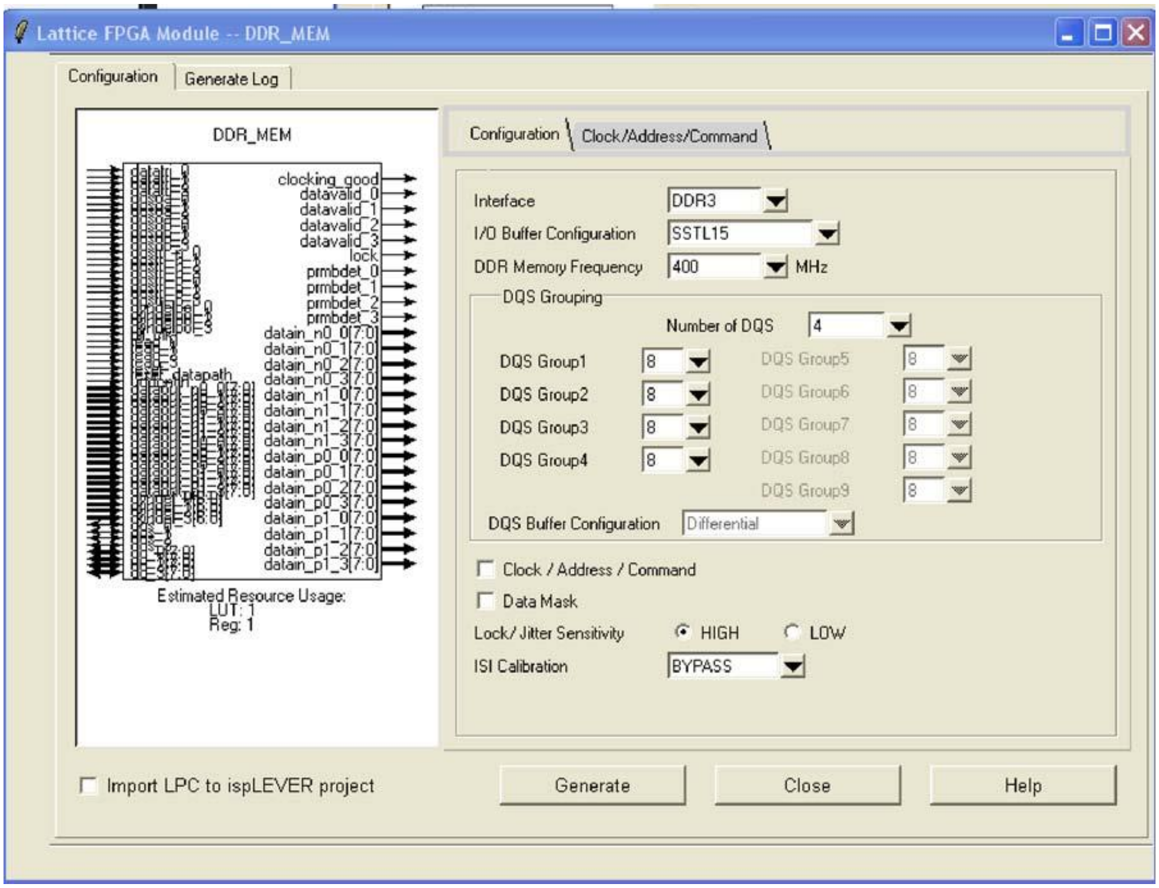
Macro Type: Module
Version: 5.1
Module Name: DDR_MEM
Project Path: :\isptools8_0\examples\fpga\latticeecp3\vhdl\cntbuf
File Name: ddr_mem
Design Entry: Schematic/VHDL
Device Family: LatticeECP3
Part Name: LFE3-70EA-6FN672CES

Customize

**Figure 10.1 IPexpress Main Window**

Figure 10.1 shows the IPexpress Main Window. To generate a DDR memory interface, select DDR_MEM, assign a module name and click on Customize to see the Configuration Tab.

Figure 10.2 shows the Configuration Tab for the DDR_MEM interface. You can choose to implement the DDR1_MEM, DDR2_MEM or DDR3_MEM interface.



Estimated Resource Usage:
LUT: 1
Reg: 1

Interface: DDR3
I/O Buffer Configuration: SSTL15
DDR Memory Frequency: 400 MHz

DQS Grouping

Number of DQS: 4

DQS Group1: 8
DQS Group2: 8
DQS Group3: 8
DQS Group4: 8
DQS Group5: 8
DQS Group6: 8
DQS Group7: 8
DQS Group8: 8
DQS Group9: 8

DQS Buffer Configuration: Differential

□ Clock / Address / Command
□ Data Mask
Lock/ Jitter Sensitivity ● HIGH ○ LOW
ISI Calibration: BYPASS

□ Import LPC to ispLEVER project

Figure 10.2 Configuration Tab for DDR_MEM

Table 10.1 describes the various settings shown in the Configuration Tab above.

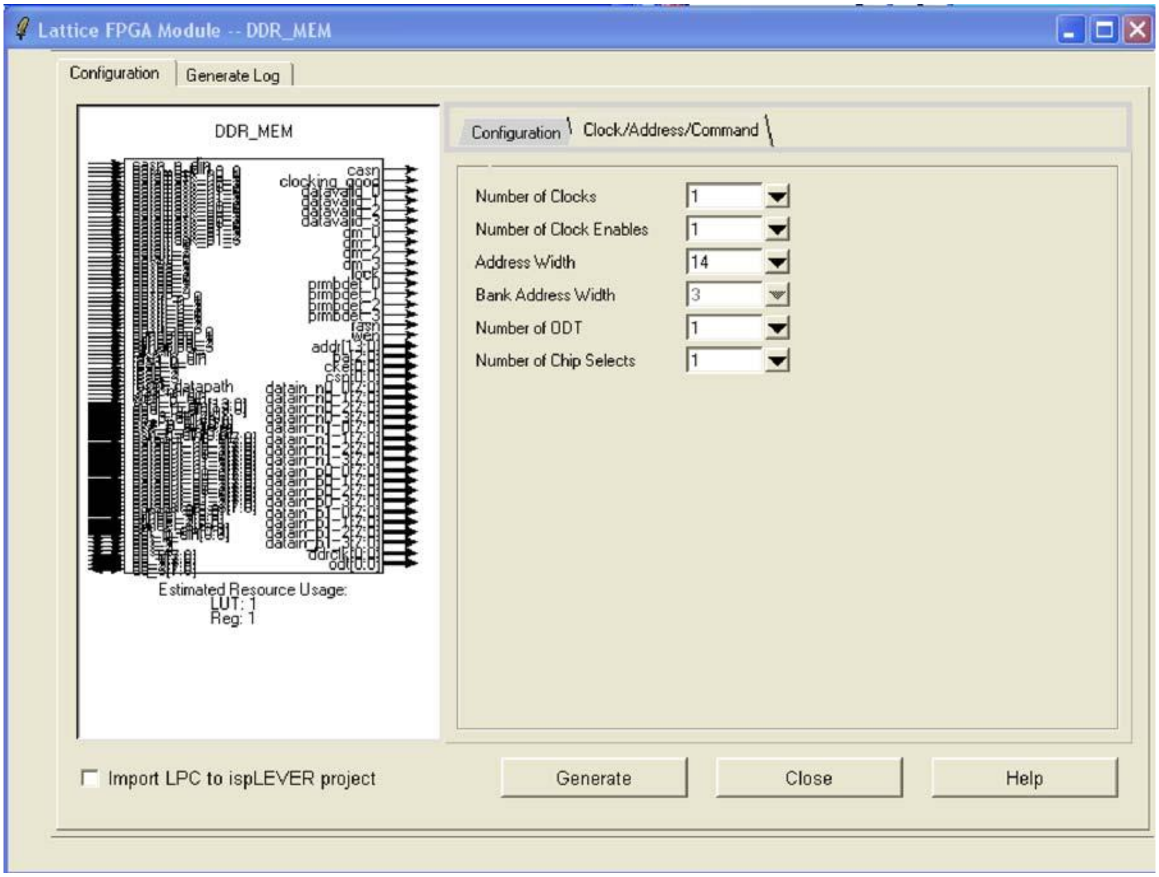
Table 10.1. Configuration Tab Settings for DDR_MEM

| User Interface Option | Description | Range | Default Value |
|---|---|---|---|
| Interface | DDR memory interface type | DDR, DDR2, DDR3 | DDR2 |
| I/O Buffer Configuration | I/O type configuration for DDR pins | SSTL25_I, SSTL25_II SSTL18_I, SSTL18_II, SSTL15 | DDR – SSTL25_I DDR2 – SSTL18_I DDR3 – SSTL15 |
| Number of DQS | Interface width (1 DQS per 8 bits of data) | 1 to 9 | 4 |
| DQS Group1 to DQS Group8 | Number of DQ per DQS pin | 1 to 8 | 8 |
| DQS Buffer Configuration | DQS buffer type | DDR: Single-ended<br>DDR2: Single-ended, Differential<br>DDR3: Differential | DDR – Single-ended<br>DDR2 – Single-ended<br>DDR3 – Differential |
| Clock/Address/Command | Clock/address/command interface is generated when this option is checked | ENABLED, DISABLED | DISABLED |
| Data Mask | Data mask signal is generated when this option is checked | ENABLED, DISABLED | DISABLED |

| User Interface Option | Description | Range | Default Value |
|---|---|---|---|
| Lock/Jitter Sensitivity | Lock Sensitivity attribute for DQSDLL* | HIGH, LOW | HIGH |
| DDR Memory Frequency | DDR Memory Interface Frequency | DDR – 87.5 MHz, 100 MHz, 133.33 MHz, 166.67 MHz, 200 MHz<br>DDR2 – 125 MHz, 200 MHz, 266.67 MHz<br>DDR3 – 300 MHz, 400 MHz | DDR – 200 MHz DDR2 – 200 MHz DDR3 – 400 MHz |
| ISI Calibration | ISI calibration is available for the DDR3 interface to adjust for inter-symbol inference adjustment per DQS group | BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7 | BYPASS |

*Note: It is recommended to set Lock Sensitivity to HIGH for DDR Memory Frequency higher than 133 MHz.

If you select to generate the Clock/Address/Command signals using IPexpress, then the settings in the Clock/Address/Command Tab are active and can be set up as required. Figure 10.3 shows the Clock/Address/Command Tab in the IPexpress for DDR2 Memory.



Lattice FPGA Module -- DDR_MEM

DDR_MEM

Estimated Resource Usage:
LUT: 1
Reg: 1

Number of Clocks: 1
Number of Clock Enables: 1
Address Width: 14
Bank Address Width: 3
Number of ODT: 1
Number of Chip Selects: 1

☐ Import LPC to ispLEVER project

**Figure 10.3 Clock/Address/Command Tab in the IPexpress for DDR_MEM**

Table 10.2 lists the values that can be used for the Clock/Address/Command settings.

**Table 10.2. Clock/Address/Command Settings for DDR_MEM**

| User Interface Option | Range | Default Value |
|---|---|---|
| Number of Clocks | 1, 2, 4 | 1 |
| Number of Clock Enables | 1, 2, 4 | 1 |
| Address Width | DDR: 12-14<br>DDR2: 13-16<br>DDR3: 13-16 | DDR: 13<br>DDR2: 13<br>DDR3: 14 |
| Bank Address Width | DDR: 2<br>DDR2: 2, 3<br>DDR3: 3 | DDR: 2<br>DDR2: 2<br>DDR3: 3 |
| Number of ODT | DDR: N/A DDR2: 1, 2, 4<br>DDR3: 1, 2, 4 | DDR: N/A DDR2: 1<br>DDR3: 1 |
| Number of Chip Selects | DDR: 1, 2, 4, 8<br>DDR2: 1, 2, 4<br>DDR3: 1, 2, 4 | DDR: 1<br>DDR2: 1<br>DDR3: 1 |

# 11. DDR Memory DQ/DQS Design Rules and Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in LatticeECP3 devices.

- LatticeECP3 devices have dedicated DQ-DQS banks. Please refer to the Logical Signal Connections tables in the LatticeECP3 Family Data Sheet before locking these pins.
- There are two DQSDLLs on the device, one for the left half and one for the right half of the device. Only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half, all the DDR memory interfaces on that half should run at the same frequency.
- Each DQSDLL generates 90° digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.
- The clock to the PLL used in the write implementation to generate the clocks for the outputs must be locked to the correct dedicated PLL pin input.
- When implementing a DDR SDRAM interface, all interface signals should be connected to the SSTL25 I/O standard.
- For the DDR2 SDRAM interface, the interface signal should be connected to the SSTL18 I/O standard.
- For the DDR3 SDRAM interface, these signals should be connected to the SSTL15 standard.
- The DDR, DDR2 and DDR3 require a differential clock signal. For these standards, the differential clock signals should be connected to SSTL25D (DDR), SSTL18D (DDR2) or SSTL15D (DDR3).
- DDR3 also requires differential DQS signal. The use of differential DQS is optional for DDR2. If differential DQS is used it should be connected to SSTL18D for DDR2 and SSTL15D for DDR3.
- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins. VREF1 should not be connected with VREF2 of the bank when implementing DDR memory interfaces.
- There is no DQS strobe support on the bottom of the device, so memory interfaces cannot be implemented on this side.
- Within a DQS-12 group, the IOLOGIC in the group’s DQS/DQS# pads cannot be used for DDR registers.
- If the register is implemented inside the FPGA fabric instead of the IOLOGIC, there is no restriction on using the DQS pad.
- The upper left corner of the LatticeECP3 device has a non-DQS DDR group. This group of I/O does not have a DQS function. This group of I/O can only be used for generic DDR implementations.
- IDDRX memory cannot be mixed in the same DQS group as an ODDRX generic implementation. Similarly, ODDRX memory cannot be mixed in the same DQS group with an IDDRX generic implementation.
- Generic DDR interfaces are not available on the top side of the LatticeECP3 “E” device only. The generic DDR required for DDR clock and control generation needs to be implemented on the left and right sides of the device. Generic DDR interfaces can be implemented on the top side of the “EA” devices, therefore this issue does not exist for “EA” devices.
- Table 11.1 summarizes what is available on each side of the LatticeECP3-70E, LatticeECP3-95E and LatticeECP3-150EA devices. Note that DDR registers are not available on the bottom of the device.

**Table 11.1. DDR Pin Limitations**

| Left | Right | Top |
|---|---|---|
| DQS12 group available | DQS12 available | DQS 12 with some restrictions |
| Input x1 DDR Generic Output x1DDR Generic Input x2 DDR Generic Output x2 DDR Generic DDR Memory<br>DDR2 Memory<br>DDR3 Memory ("EA" only) | Input x1 DDR Generic Output x1DDR Generic Input x2 DDR Generic Output x2 DDR Generic DDR Memory<br>DDR2 Memory<br>DDR3 Memory ("EA" only) | Input x1 DDR Generic ("EA" only) Output x1DDR Generic ("EA" only) Input x2 DDR Generic ("EA" only) DDR Memory ("E" and "EA") DDR2 Memory ("E" and "EA") |
| Upper left side has DDR function without DQS. | Upper right side I/O shared with sysCONFIG pins in bank 8. No DDR functions are available on these pins | Right part of top side has eight I/O shared with the sysCONFIG pins. No DDR functions are available. |

**Note**: See the LatticeECP3 Family Data Sheet Pinout tables to find DQS group assignments.

# 12. DDR/DDR2 Pinout Guidelines

- The DQS-DQ association rule must be followed.
  - All associated DQs (8 or 4) to a DQS must be in the same DQS-12 group.
- The data mask (DM) must be part of the corresponding DQS-12 group.
  - Example: DM[0] must be in the DQS-12 group that has DQ[7:0], DQS[0].
- DQS pad must be allocated to a dedicated DQS pad.
  - DQS# pad is used when differential DQS is selected.
- Do not assign any signal to the DQS# pad if SSTL18D is applied to the DQS pad.
  - The software automatically places DQS# when SSTL18D is applied.
- DQS/DQS# pads cannot be used for other DDR functions.
  - The DQS IOLOGIC structure is not compatible with non-DQS DDR IOLOGIC.
- The clock to the PLL used to generate the outputs must be assigned to use dedicated clock routing.
- Data group signals (DQ, DQS, DM) can use either the left, right or top edge of LatticeECP3.
- Locating memory clock signals:
  - For the LatticeECP3 "E" device, it is highly recommended to have all clock pads within a memory controller be in one DQS-12 group to minimize pinout restrictions.
- The bottom-side pads in Bank 6 and Bank 3 are also good candidates for address/command/control signals. Save pins on the left and right sides for DDR.
- VREF1 of the bank where DQs are located must not be taken by any signal.
  - VREF2 is OK to use.
  - Do not connect VREF1 and VREF2 together.
- External termination to VTT is required for DDR and DDR2 interfaces. All DQ and DQS pins must be terminated to VTT using an external termination resistor. VTT = ½ VCCIO (0.9 V for DDR2 and 1.25 V for DDR). It is recommended that SI Simulation be run to determine the best termination value. If signal integrity simulation is not available, parallel termination of 75 Ω to VTT should be used.
- It is required to provide a PCB connect resistor to the XRES pins. These pins cannot be used for other functions. See LatticeECP3 Hardware Checklist (FPGA-TN-02183), for detailed requirements on the XRES pin.

# 13. DDR3 Termination Guidelines

Proper termination of a DDR3 interface is an important part of implementation that ensures reliable data transactions at high speed. Below is the general termination guideline for the LatticeECP3 DDR3 interface.

## 13.1. Termination for DQ, DQS and DM

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.
- External termination to VTT is required for DDR3 interfaces at the FPGA side. Each DQ and DQS pin should be terminated to VTT using an external termination resistor. The termination resistor location is important. See the Layout Considerations for DDR3 section for the requirements.
- It is recommended that signal integrity (SI) simulation be run to determine the best termination value. If SI simulation is not available, parallel termination of 100 Ω to 120 Ω to VTT is recommended.
- Use of series termination resistors at the FPGA side is not recommended.

## 13.2. Termination for CK

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

- Locate a 100-Ω resistor between the positive and negative clock signal, or
- Connect one end of an Rtt resistor to the positive pin and one end of another Rtt to the negative pin of a CK pair, then connect the other ends of two Rtt resistors together and return to VDD through a Ctt capacitance. This is a JEDEC CK termination scheme defined in the DIMM specifications. JEDEC uses 36-Ω for Rtt with 0.1 uF Ctt for DIMM. 50-Ω Rtt can also be used for non-DIMM applications.
- Use of series termination resistors at the FPGA side is not recommended.
- When fly-by wiring is used, the CK termination resistor should be located after the last DDR3 SDRAM device.

## 13.3. Termination for Address, Commands and Controls

Parallel termination to VTT on address, command and control lines is typically required.

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

## 13.4. Termination for DDR3 DIMM

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

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

# 14. DDR3 Interface without Termination

When a wide DDR3 data bus is implemented and requires most of the pin resources in the assigned banks to be used as data lines, SSO impact usually becomes a designer’s concern. While proper use of termination resistors provides optimized signal integrity results, removing them may also provide improved noise margin in some cases due to increased eye height.

## 14.1. DQ, DQS and DM without Parallel VTT Termination

Although the external parallel VTT termination is typically required for read operations at the FPGA side, it can be removed if the following conditions are met:

- Point-to-point connection between FPGA and DDR3 SDRAM device.
- PCB trace length is maintained shorter than 3.0"
- SI simulation confirms that there is no significant reflection or ringing noise due to unmatched line impedance.

Either or both of the following considerations are suggested when you implement a DDR3 interface without external VTT termination:

- You can still keep the external VTT termination in your PCB layout without population if the board space allows. It gives you an opportunity to return to the external termination scheme without board re-spin work.
- You can connect the VTT input pads in the banks where the DDR3 interface’s data bus is implemented to the external VTT source. This allows you to use the LatticeECP3's internal on-chip termination on selective signals in case only a few have bad signal integrity results. This approach also allows you to use internal VTT termination only on DQS signals, which can be a useful termination option in some cases.

## 14.2. Address, Command and Control Signals without Parallel VTT Termination

As described, parallel termination to VTT on address, command and control lines is typically required. However, the parallel VTT termination on them can also be optionally removed to increase the noise margin or to avoid layout difficulties at the memory side. When they are removed, use of series termination resistors at the driver side (FPGA) is recommended to suppress overshoot and undershoot noise. The same condition as specified above for DQ, DQS and DM is applied to remove the parallel termination.

# 15. Layout Considerations for DDR3

- Placement of external discrete resistors or resistor packs (RPACKS) for parallel VTT termination is critical and must be placed within 0.6 inches (600-mil) of the FPGA ball.
  - 120 Ω BGA RPACKS (CTS RT2432B7 type) are recommended for the 64- and 32-bit interfaces due to better routing and density issues. Each RPACK contains 18 resistors in a very small BGA footprint. Note that only 120, 75 and 50 Ω values are available in this package type.
  - 4 × 1 RPACKS (CTS 741X083101JP type) can also be used in cases where a 100 Ω value is needed without routing/density issues.
- The termination resistor stubs should be kept minimal.
- All traces should be matched to 50 Ω.
- For SODIMM and UDIMM designs, write leveling should be used and all traces from the FPGA to the DIMM should be matched in length on the PC board similar to DDR2.
- If using a discrete DDR3 device, fly-by routing can be used for traces. If fly-by routing is used, the Write Leveling option must be enabled. If not using fly-by routing, the Write Leveling option must be disabled.
- Refer to LatticeECP3 Hardware Checklist (FPGA-TN-02183), DDR3 Interface Requirements section for complete DDR3 layout guidelines.

# 16. DDR3 Pinout Guidelines

The LatticeECP3 device contains dedicated I/O functions for supporting DDR3 memory interfaces. The following pinout rules must be followed to properly use the dedicated I/O functions.

- The DQS-DQ association rule must be followed.
  - All associated DQs (8 or 4) to a DQS must be in the same DQS-12 group.
- A data mask (DM) must be part of the corresponding DQS-12 group.
  - Example: DM[0] must be in the DQS-12 group that has DQ[7:0], DQS[0].
- A DQS pad must be allocated to a dedicated DQS True (+) pad.
  - A DQS# pad is auto-placed when a differential SSTL type (SSTL15D) is selected.
- Do not assign any signal to a DQS# pad if SSTL15D is applied to the DQS pad.
  - The software automatically places DQS# when SSTL15D is applied.
- DQS/DQS# pads cannot be used for other DDR functions.
  - The DQS IOLOGIC structure is not compatible with non-DQS DDR IOLOGIC.
  - Do not use DQS pads for any DDR3 signals except DQS and RST#. They can be used for other user logic signals if a DDR function is not required.
- Data group signals (DQ, DQS, DM) must use the left and right sides of the LatticeECP3 device.
  - Top-side pads do not have 2x gearing.
- Address, command, control and CK signals must be located on generic DDR-capable pads (ODDRXD).
  - Place the CK/CK# outputs on the same side where the DQ and DQS pads are located. The top side is not recommended for the high-speed DDR CK function.
  - Place the address, command and control signals either on the same side as where the DQ and DQS pads are located or on the top side. The bottom side cannot be used.
- RST# can be located anywhere an output is available as long as LVCMOS15 is applicable.
  - All DDR3 signals except RST# use DDR functions.
- The DDR3 input reference clock to the PLL must be assigned to use dedicated clock routing.
  - The dedicated PLL input pads are recommended while PCLK inputs can also be used.
  - Two PLLs that have a direct connection to ECLK in each side can be used for a DDR3 function. (Note that LatticeECP3-17EA devices have one PLL that supports DDR3 in each side.)
- Do not use the bottom-side pads in Bank 6 and Bank 3 for address, command and control signals.
  - No generic DDR support on the bottom side of the device.
- VREF1 of the bank where the DQ, DQS and DM pads are located must not be taken by any signal.
  - Do not PROHIBIT VREF1 in the preference file.
  - VREF2 can be used for a general purpose I/O signal.
  - Do not connect VREF1 and VREF2 together.
- Leave VTT pins unconnected when external VTT termination is implemented.
  - VTT pins can be optionally connected to the external VTT source when a DDR3 interface is implemented without external VTT termination. See the DDR3 Interface without Termination section for details.
  - VTT pins can be connected in series with a capacitor (around .01 uf) to ground when the bank has LVDS internal termination to suppress externally generated common mode noise. Internal VTT termination cannot be used in this case.
- It is required to provide a PCB connect resistor to the XRES pins. These pins cannot be used for other functions. See LatticeECP3 Hardware Checklist (FPGA-TN-02183), for detailed requirements on the XRES pin.

# 17. Pin Placement Considerations for Improved Noise Immunity

In addition to the general pinout guidelines, there are additional pinout considerations for minimizing simultaneous switching noise (SSN) impact. The following considerations are necessary to control SSN within the required level:

1. Properly terminated interface
2. SSN optimized PCB layout
3. SSN considered I/O pad assignment
4. Use of pseudo power pads

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

- Place the DQS groups for data implementation starting from the middle of the (right or left) edge of the LatticeECP3. Allow a corner DQS group to be used as a data group only when necessary to implement the required width.
- Locate a spacer DQS group between the data DQS groups if possible. A DQS group becomes a spacer DQS group if the I/O pads inside the group are not used as data pads (DQ, DQS or DM).
    - The pads in a spacer group can be used for address, command, control or CK pads as well as for user logic or the pseudo power pads.
    - No more than two consecutive-data DQS group placements is recommended in the middle of the edge. When a corner-side DQS group is used for data, locate a spacer DQS group right next to it.
    - If there is an incomplete DQS group (not the size of DQS-12) or enough space for more than 10 pads between two DQS groups, the following DQS group can be located next to the previous complete DQS group, both as data groups. The incomplete DQS groups, or the space between them, can be used as a spacer.
- It is recommended that you locate a few pseudo VCCIO/ground (GND) pads inside a spacer DQS group. An I/O pad becomes a pseudo power pad when it is configured to OUTPUT with its maximum driving strength (i.e., SSTL15, 10 mA for DDR3) and connected to the external VCCIO or ground power source on the PCB.
    - Your design needs to drive the pseudo power I/O pads according to the external connection. (i.e., you assign them as OUTPUT and let your design drive ‘1’ for pseudo VCCIO pads and ‘0’ for pseudo GND pads in your RTL coding.)
    - Locating four pseudo power pads in a spacer DQS group should be sufficient to efficiently suppress the SSN impact. At least two pseudo pads should be implemented in a spacer DQS group if more pins are needed for a user design.
    - Good candidate pads are two pads in both ends (the first and the last ones in the group) and/or two DQS (positive and negative) pads in the middle.
- You may have one (DDR2/DDR3) or two (DDR/DDR2) remaining pads in a data DQS group which are not assigned as a data pad in a DDR memory interface. Assign them to pseudo VCCIO or pseudo GND. The preferred location is in the middle of the group (right next to a DQS pad pair). Note that you may not have this extra pad in DDR2/DDR3 if the DQS group includes a VREF pad for the bank.
- Avoid fast switching signals located close to the XRES pad of a LatticeECP3 device. XRES requires an external resistor which is used to create the bias currents for the I/O. Since this resistor is used for a calibration reference for sensitive on-chip circuitry, careful pin assignment around the XRES pad is also necessary to produce less jittery PLL outputs for DDR memory interface operations.

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

- Assign the DM (data mask) pad in a data DQS group close to the other side of DQS pads where a pseudo power pad is located. If the data DQS group includes VREF, locate DM to the other side of VREF with respect to DQS. It can be used as an isolator due to its almost static nature in most applications.
- Other DQS groups (neither data nor spacer group) can be used for accommodating DDR3 address, command, control and clock pads. It is recommended that you still assign all or most DQS pads (both positive and negative) in these groups to pseudo power. Since LatticeECP3 DQS pads have a dedicated DDR function that cannot be shared with other DDR3 signals, they are good pseudo power pad candidates.

- You can assign more unused I/O pads to pseudo power if you want to increase the SSN immunity. Note that the SSN immunity does not get increased at the same rate as the increased number of pseudo power pads. The first few pseudo power pad placements described above are more crucial. Keep the total pseudo power pad ratio (VCCIO vs. GND) between 2:1 to 3:1.
- Although not necessary, it is slightly more effective to locate a pseudo VCCIO to a positive pad (A) and GND to a negative pad (B) of a PIO pair if possible.
- If a bank includes unused input-only pads such as dedicated PLL input pads, you can also connect them to VCCIO on your PCB. They are not as efficient as the pseudo power pads but can still be used as isolators, and the connections on the board would provide good shielding. No extra consideration is necessary for these pins in your design.
- It is a good idea to shield the VREF1 pad by locating pseudo power pads around it if the VREF1 pad is not located in a data DQS group.

Table 17.1 shows the recommended examples of DQS group allocations following the guidelines. If you have enough pin resources, following the best examples would provide you with maximized SSN immunity results. It is more practical in most applications to follow the examples in the “Allowed” columns. It is expected that the SSN and ground bounce impacts are considerably less than those cases where you do not include any consideration.

**Table 17.1. Recommended Examples of DQS Group Allocation**

| LatticeECP3-150EA | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **DQS** | **150-1156 Left** | | **150-1156 Right** | | **150-672 Left** | | **150-672 Right** | | | | | | | | | |
| | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | | | | | | | | |
| 1 | | D | | D | | D | | D* | | | | | | | | |
| 2 | D | | D | D* | D | | D | D | | | | | | | | |
| 3 | | D | | | | D | | | | | | | | | | |
| 4 | D | | D | D | D | D | D | D | | | | | | | | |
| 5 | | D | | D | | | D* | D* | | | | | | | | |
| 6 | D | D | D | | D | D | | | | | | | | | | |
| 7 | | | | D | D* | D* | | | | | | | | | | |
| 8 | D | D | D | D | | | | | | | | | | | | |
| 9 | | D | | | | | | | | | | | | | | |
| 10 | D | | D | D | | | | | | | | | | | | |
| 11 | | D | | | | | | | | | | | | | | |
| 12 | D | D | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | |
| Bus Size | 48 | 64 | 40 | 56 | 32 | 40 | 24 | 32 | | | | | | | | |
| **LatticeECP3-95/70EA** | | | | | | | | | | | | | | | | |
| **DQS** | **95/70-1156 Left** | | **95/70-1156 Right** | | **95/70-672 Left** | | **95/70-672 Right** | | **95/70-484 Left** | | **95/70-484 Right** | | | | | |
| | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | | | | |
| 1 | | D | | D | | D | | D* | | | | D | | | | |
| 2 | D | | D | | D | | D | D | D | D | D | | | | | |
| 3 | | D | | D | | D | | | | D | | D | | | | |
| 4 | D | D | D | D | D | D | D | D | D | | D | D | | | | |
| 5 | | | | | | | D* | D* | | D | | | | | | |
| 6 | D | D | D | D | D | D | | | | D* | | | | | | |
| 7 | | D | D* | D* | D* | D* | | | | | | | | | | |
| 8 | D | D* | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| DDR3 Bus | 32 | 48 | 32 | 40 | 32 | 40 | 24 | 32 | 16 | 32 | 16 | 24 | | | | |
| **LatticeECP3-35EA** | | | | | | | | | | | | | | | | |
| **DQS** | | | | | **35-672 Left** | | **35-672 Right** | | **35-484 Left** | | **35-484 Right** | | **35-256 Left** | | **35-256 Right** | |
| | | | | | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** |
| 1 | | | | | | D | | D | | | | D | D* | D* | D* | D* |
| 2 | | | | | D | | D | | D | D | D | | | D* | | |
| 3 | | | | | | D | | D | | D | | D | | | | |
| 4 | | | | | D | D | D | D | D | | D | D | | | | |
| 5 | | | | | | | | | | D | | | | | | |
| 6 | | | | | D | D | | D | | D* | | | | | | |
| 7 | | | | | | | | | | | | | | | | |
| DDR3 Bus | | | | | 24 | 32 | 16 | 32 | 16 | 32 | 16 | 24 | 8 | 16 | 8 | 8 |

| LatticeECP3-17EA | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **DQS** | | | | | | | | | **17-484 Left** | | **17-484 Right** | | **17-256 Left** | | **17-256 Right** | |
| | | | | | | | | | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** | **Best** | **Allowed** |
| 1 | | | | | | | | | | D | | | | D* | D* | D* |
| 2 | | | | | | | | | D | | D | D | D* | D* | | |
| 3 | | | | | | | | | | D | | D | | | | |
| 4 | | | | | | | | | D | D | | | | | | |
| 5 | | | | | | | | | | | | | | | | |
| DDR3 Bus | | | | | | | | | 16 | 24 | 8 | 16 | 8 | 16 | 8 | 8 |

**Notes**: DQS groups with a 'D' indicate the data DQS groups while the blank ones indicate the spacer DQS groups. Data DQS groups with an asterisk indicate that they have an incomplete DQS group or enough isolation in front. Shaded cells are not applicable to the selected device.

# 18. DDR Software Primitives and Attributes

This section describes the software primitives that can be used to implement all the DDR interfaces. These primitives are divided into ones that are used to implement the DDR data and ones for DDR Strobe signal or the Source Synchronous clock. The DQSBUF primitives are used to generate the signals required to correctly capture the data from the DDR memory.

**Table 18.1. DDR Software Primitives List**

| Type | Primitive | Usage |
|---|---|---|
| Data Input | IDDRXD | E and EA Generic DDRX1<br>E and EA DDR/DDR2 Memory |
| | IDDRX1D | EA Generic DDRX1 |
| | IDDRX2D | E Data Input Generic DDRX2 E and EA DDR3 Memory |
| | IDDRX2D1 | EA Generic DDRX2 |
| Data Output | ODDRXD | E and EA Generic DDRX1<br>E and EA DDR/DDR2/DDR3 Memory |
| | ODDRXD1 | EA Generic DDRX1 |
| | ODDRX2D | E and EA Generic DDRX2 E and EA DDR3 Memory |
| Data Tri-state | ODDRTDQA | E and EA Generic DDRX2 E and EA DDR3 Memory |
| | OFD1S3AX | E and EA Generic DDRX1<br>E and EA DDR/DDR2 Memory |
| DQS Output | ODDRXDQSA | E and EA DDR/DDR2 Memory |
| | ODDRX2DQSA | E and EA DDR3 Memory |
| DQS Tri-state | ODDRTDQSA | E and EA DDR/DDR2 Memory E and EA DDR3 Memory |
| DQSBUF Logic | DQSBUFD | E and EA DDR3 Memory, E and EA Generic DDRX2 (for bus widths <10 bits) |
| | DQSBUFF | E and EA DDR/DDR2 Memory, E and EA Generic DDRX1 (for bus widths <10 bits) |
| | DQSBUFE | E Generic DDRX2 |
| | DQSBUFE1 | EA Generic DDRX2 |
| | DQSBUFG | E Generic DDRX1 |
| DQSDLL | DQSDLL | DLL for Generic DDRX1/DDRX2 (for bus width <10 bits and multiple interfaces per side of the device) and DDR/DDR2/DD3 Memory |
| Input Delay | DELAYB | Delay block for Generic DDRX2 with Dynamic Control |
| | DELAYC | Delay block for Generic DDRX1/X2 with clock injection removal. The amount of Fixed Delay varies by interface. |
| ECLK Stop | ECLKSYNCA | EA Generic DDRX2 Output<br>EA ECLK Synchronization for DDR3 Memory |

## 18.1. DQSDLLB

The DQSDLLB generates the 90° phase shift required for the DQS signal. This primitive implements the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one-half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DLL generates the delay based on this clock frequency and the update control input to this block. The DLL updates the dynamic delay control to the DQS delay block when this update control (UDDCNTLN) input is asserted. Figure 18.1 shows the primitive symbol. The active low signal on UDDCNTLN updates the DQS phase alignment.

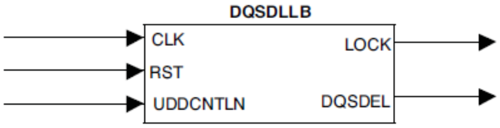


**Figure 18.1 DQSDLL Symbol**

Table 18.2 provides a description of the ports.

**Table 18.2. DQSDLLB Ports**

| Port Name | I/O | Definition |
|---|---|---|
| CLK | I | CLK should be at the frequency of the DDR interface. |
| RST | I | Resets the DQSDLLB. |
| UDDCNTLN | I | Provides update signal to the DLL that updates the dynamic delay. |
| LOCK | O | Indicates when the DLL is in phase. |
| DQSDEL | O | The digital delay generated by the DLL, should be connected to the DQSBUF primitive. |

### 18.1.1. DQSDLL Update Control

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

It is important to understand that the UDDCNTLN signal is a synchronous input to the DQSDLL CLK domain. When using DDR in 2x gearing, it is required to use clock domain transfer logic first to transfer UDDCNTLN from slow clock domain to fast clock domain before it is input to DQSDLL. You can use two- or three-stage pipeline registers to safely transfer the DQSDEL update control input to the DQSDLL block. The first stage register uses the local domain clock while the second and third registers use the DQSDLL CLK domain clock. The second to third stage pipelining is desirable because it can eliminate the placement and routing issue due to the increased clock rate (2x) for the net and also avoids any meta-stability issues.

### 18.1.2. DQSDLL Configuration

By default, this DLL generates a 90° phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL.

### 18.1.3. DQSBUF Logic Primitives for Generic DDR

The DQSBUF primitives (DQSBUFE and DQSBUFG for “E” devices) are used to generate the strobe logic and delay used in the input DDR modules to correctly demux the DDR data. The DQSBUFE is used for x2 interfaces and the DQSBUFG is used for x1 interfaces. The DQSBUFE1 is used only for output x2 interfaces in LatticeECP3 devices.

Figure 18.2 shows DQSBUF Generic DDR functions for the “E” and “EA” devices.

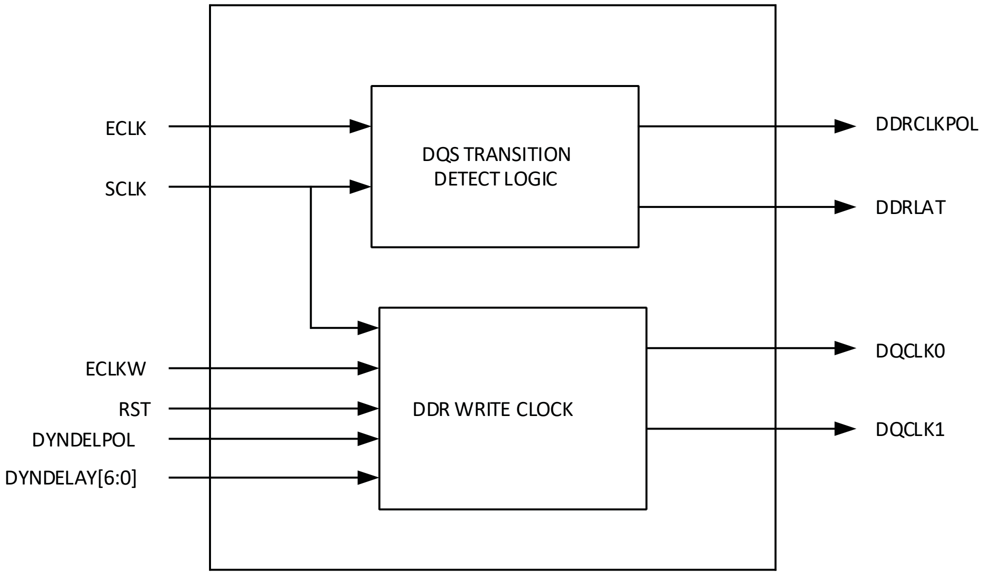


**Figure 18.2 DQSBUFE Function for Generic Input/Output DDR x2 Interfaces (“E” Devices)**

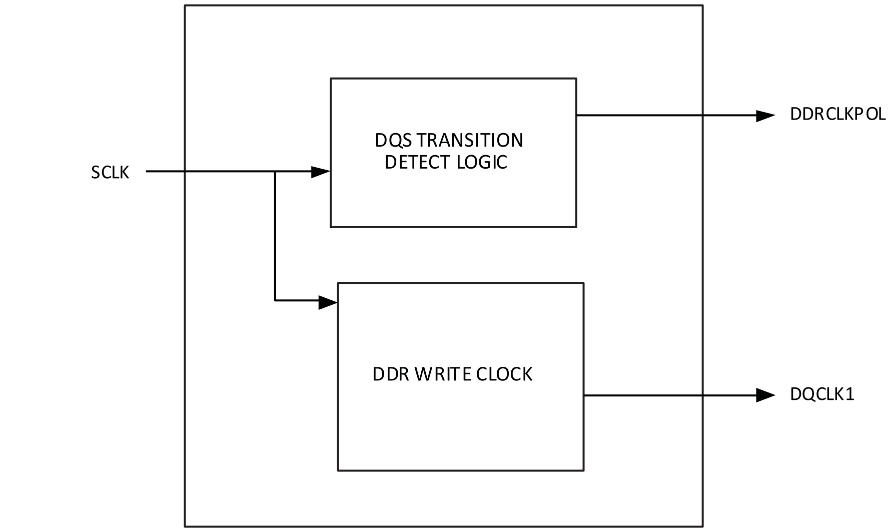


**Figure 18.3 DQSBUFG Function for Generic Input/Output DDR x 1 Interface ("E" Devices)**

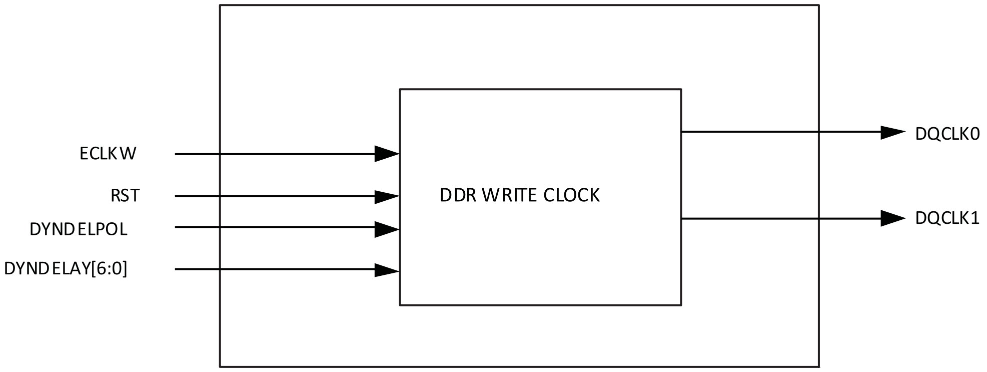


**Figure 18.4 DQSBUFE1 Function for Generic Output DDR x2 Interfaces ("EA" Devices)**

### 18.1.4. DQS Transition Detect

The DQS Transition Detect block inputs the fast ECLK and the slower SCLK (=1/2 ECLK) inputs and generates the DDRCLKPOL and the DDRLAT signals. These signals are generated based on the phase of the FPGA SCLK at the first ECLK transition. The DDRLAT signal is used in generic DDRX2 mode to transfer data from the ECLK to SCLK domain. These are only required to implement generic DDR on the LatticeECP3 "E" devices. LatticeECP3 "EA" devices do not require these signals.

### 18.1.5. DDR Write Clock

This block inputs the fast edge clock used for the write side and generates two control signals, DQCLK0 and DQCLK1. For Generic DDRX2 gearing, both DQCLK0 and DQCLK1 are generated. These control clocks run at a rate of one-half the fast edge clock, and DQCLK1 is offset delayed by 90° relative to DQCLK0. DQCLK1 output matches the phase of the SCLK input to the block. These two clocks toggle between different legs of a 4:1 mux in the output logic, which allows 4:1 gearing of data at twice the edge clock rate. When using generic DDRX1, instead of DDRX2 gearing, only the DQCLK1 is output, which toggles at the rate of FPGA clock provided by SCLK and matches the phase of the SCLK input.

The DDR write clock block also inputs DYNDELPOL and DYNDELAY [6:0] delay inputs. These inputs support the DDR3 memory interface to adjust delay among the various DQS groups. They can also be used for the generic DDR for the same purpose. The DYNDELAY [6:0] can input 128 possible delay step settings with each step generating approximately 25ps nominal delay. In addition, the DYNDELPOL can be used to invert the clock for a 180° skew using this delay. The DYNDELAY [6:0] and DYNDELPOL inputs should be generated using the FPGA core.

## 18.2. DQSBUFE

This primitive provides the control logic for Generic DDRX2 interface in the LatticeECP3 "E" devices. Figure 18.5 shows the primitive symbol.

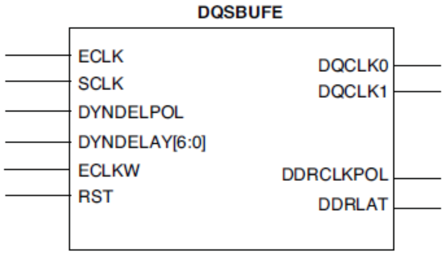


**Figure 18.5 DQSBUFE Symbol**

Table 18.3 provides a description of all the I/O ports associated with the DQSBUFE primitive.

**Table 18.3. DQSBUFE Primitive Ports**

| Port Name | I/O | Definition |
|---|---|---|
| ECLK | I | Edge CLK |
| SCLK | I | System CLK |
| ECLKW | I | Edge CLK used the DDR write side |
| RST | I | Reset input |
| DYNDELPOL | I | Input from user logic used to invert the clock polarity |
| DYNDELAY [6:0] | I | Input from user logic used to delay the ECLK |
| DQCLK0 | O | Clock output at frequency of SCLK used for output side gearing |
| DQCLK1 | O | Clock output at frequency of SCLK and matches the phase of SCLK using for output side gearing. DQCLK1 is 90° shifted from DQCLK0. |
| DDRCLKPOL | O | DDR clock polarity signal |
| DDRLAT | O | DDR latch control signal |

## 18.3. DQSBUFG

This primitive implements the strobe logic for Generic DDRX1 interface for LatticeECP3 "E" devices. Figure 18.6 shows the primitive symbol.

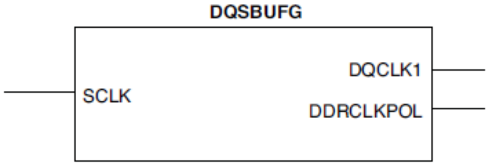


**Figure 18.6 DQSBUFG Symbol**

Table 18.4 provides a description of all the I/O ports associated with the DQSBUFG primitive.

**Table 18.4. DQSBUFG Primitive Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK |
| DQCLK1 | O | Clock output at frequency of SCLK, matches the phase of SCLK used for output gearing |
| DDRCLKPOL | O | DDR clock polarity signal |

## 18.4. DQSBUFE1

This primitive implements the strobe logic for Generic DDRX2 Output interfaces on the LatticeECP3 "EA" devices. Figure 18.7 shows the primitive symbol.

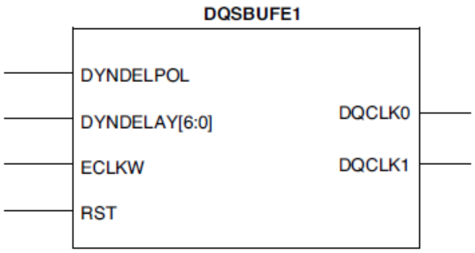


**Figure 18.7 DQSBUFE1 Symbol**

Table 18.5 provides a description of all the I/O ports associated with the DQSBUFE1 primitive.

**Table 18.5. DQSBUFE1 Primitive Ports**

| Port Name | I/O | Definition |
|---|---|---|
| ECLKW | I | Edge CLK used the DDR write side |
| RST | I | Reset input |
| DYNDELPOL | I | Input from user logic used to invert the clock polarity |
| DYNDELAY [6:0] | I | Input from user logic used to delay the ECLK |
| DQCLK0 | O | Clock output at frequency of SCLK used for output side gearing |

## 18.5. DQSBUF Logic Primitives for DDR Memory Interfaces

The DQSBUF primitives (DQSBUFD and DQSBUFF) are used to generate the DQS strobe logic and delay used in the input DDR modules to correctly demux the DDR data. The DQSBUF module used for the DDR memory interface is composed of DQS Delay, DQS Transition Detect, Data Valid Generation and the DDR Write Clock block, as shown in Figure 18.8.

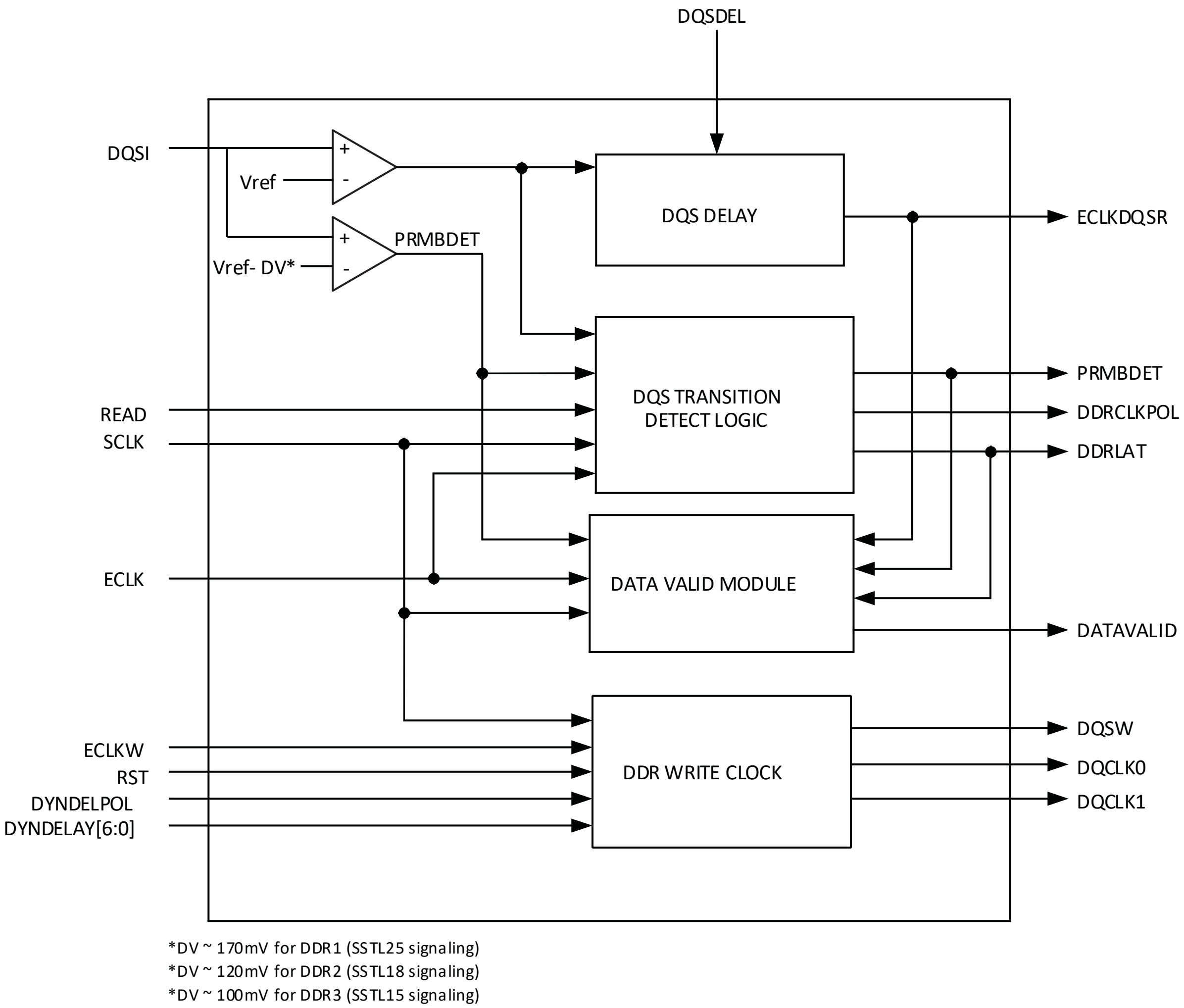


**Figure 18.8 DQSBUF Block for DDR Memory Interfaces**

## 18.5.1. DQS Delay Block

The DQS Delay block receives the digital control delay line (DQSDEL) coming from one of the two DQSDLL blocks. These control signals are used to delay the DQSI by 90°. ECLKDQSR is the delayed DQS and is connected to the clock input of the first set of input DDR registers.

## 18.5.2. DQS Transition Detect

The DQS Transition Detect block generates the DDR Clock Polarity (DDRCLKPOL) and DDR Latch Control (DDR- LAT) signal based on the phase of the FPGA clock (SCLK) and edge clock signal (ECLK) at the first DQS transition. The DDR READ control signal and FPGA CLK inputs to this block come from the FPGA core. The DDRLAT signal is used when implementing DDRX2 output gearing to transfer data from the ECLK to SCLK domain.

## 18.5.3. Data Valid Module

The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out of the input DDR registers to the FPGA core.

### 18.5.4. DDR Write Clock

This block inputs the fast edge clock used for the write side and generates two control signals, DQCLK0 and DQCLK1, and the clock used to generate the DQS clock (DQSW). DQSW is generated by applying the DQSDEL from the DQSDLL to delay the ECLK (DDR3) or SCLK (DDR and DDR2) inputs. For the DDR3 memory interface both DQCLK0 and DQCLK1 are generated. These control clocks run at a rate of one-half the edge clock, and DQCLK1 is offset delayed by 90° relative to DQCLK0. These two clocks toggle between different legs of a 4:1 mux allowing 4:1 gearing of data at twice the edge clock rate. When using output DDRX1 instead of DDRX2 gearing, only the DQCLK1 is output, which is the same as the FPGA clock.

The DDR write clock block also inputs DYNDELPOL and DYNDELAY [6:0] delay inputs. These are used to support the write leveling required for DDR3 memory interface. This delay can be used to adjust the delays among the DQS groups to account for any skew that may be introduced due to the DDR3 fly-by topology.

The DYNDELAY [6:0] can input 128 possible delay step settings with each step generating approximately 26 ps nominal delay. In addition, the DYNDELPOL can be used to invert the clock for a 180° shift of the incoming clock. The DYNDELAY [6:0] and DYNDELPOL inputs should be generated in the memory controller.

## 18.6. DQSBUFD

This primitive implements the strobe logic for DDR3 memory interface. Figure 18.9 shows the primitive symbol.

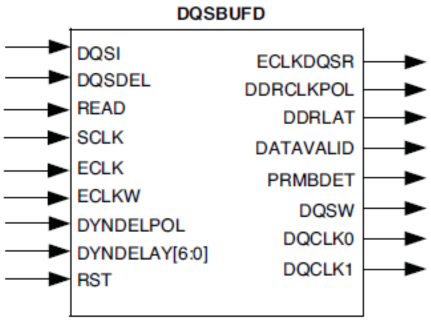


**Figure 18.9 DQSBUFD Symbol**

Table 18.6 provides a description of all the I/O ports associated with the DQSBUFD primitive.

**Table 18.6. DQSBUFD Primitive Ports**

| Port Name | I/O | Definition |
|---|---|---|
| DQSI | I | DQS strobe input from the memory |
| Read | I | Read signal generated from the FPGA core |
| ECLK | I | Edge CLK |
| SCLK | I | System CLK |
| DQSDEL | I | DQS delay signal from the DQSDLL module |
| ECLKW | I | Edge CLK used the DDR write side |
| RST | I | Reset input |
| DYNDELPOL | I | Input from user logic used to invert the clock polarity |
| DYNDELAY [6:0] | I | Input from user logic used to delay the ECLK |
| ECLKDQSR | O | Delay DQS used to capture the data |
| PRMBDET | O | Preamble detect signal, going to the FPGA core logic |
| DATAVALID | O | Signal indicating transmission of Valid data to the FPGA core |
| DDRCLKPOL | O | DDR Clock polarity signal |
| DDRLAT | O | DDR latch control signal |
| DQSW | O | Clock used to generate DQS on the write side |
| DQCLK0 | O | Clock Output at frequency SCLK used for output gearing |

| Port Name | I/O | Definition |
|---|---|---|
| DQCLK1 | O | Clock output at frequency and phase of SCLK used for output gearing |

## 18.7. DQSBUFF

This primitive implements the strobe logic for DDR and DDR2 memory interface. Figure 18.10 shows the primitive symbol.

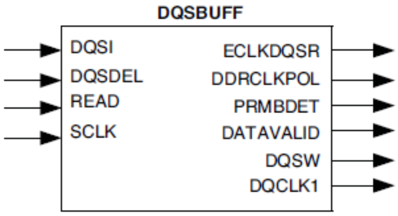


**Figure 18.10 DQSBUFF Symbol**

Table 18.7 provides a description of all the I/O ports associated with the DQSBUFF primitive.

**Table 18.7. DQSBUFF Primitive Ports**

| Port Name | I/O | Definition |
|---|---|---|
| DQSI | I | DQS strobe input from the memory |
| READ | I | Read signal generated from the FPGA core |
| SCLK | I | System CLK |
| DQSDEL | I | DQS delay signal from the DQSDLL module |
| ECLKDQSR | O | Delay DQS used to capture the data |
| PRMBDET | O | Preamble detect signal, going to the FPGA core logic |
| DATAVALID | O | Signal indicating transmission of valid data to the FPGA core |
| DDRCLKPOL | O | DDR Clock polarity signal |
| DQSW | O | Clock used to generate DQS on the write side |
| DQCLK1 | O | Clock output at frequency and phase of SCLK used for output gearing |

### 18.7.1. READ Pulse Generation

The READ signal to the DQSBUFF block is internally generated in the FPGA core. The READ signal goes high after the READ command to control the DDR-SDRAM is initially asserted. This should normally precede the DQS preamble by one cycle yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry requires the falling edge of the READ signal to be placed within the preamble stage.

Figure 18.11 shows a READ pulse timing example with respect to the PRMBDET signal.

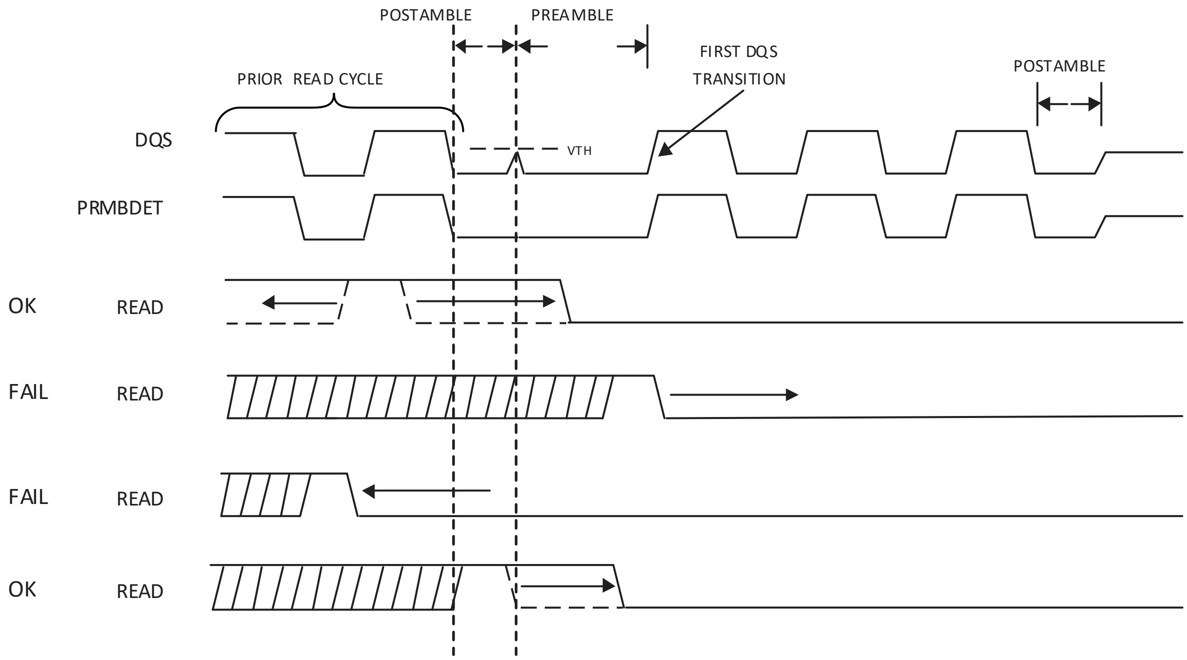


**Figure 18.11 READ Pulse Generation**

### 18.7.2. DQSBUF Attributes

Table 18.8 shows the attributes can be used with the DQSBUF primitives described above.

**Table 18.8. DQSBUF Attributes**

| Attribute | Description | Values | Software Default | Used in DQSBUF |
|---|---|---|---|---|
| DYNDEL_TYPE | Type of Static Delay input to the write control block<br>Normal: 0° phase shifted<br>Shifted: 180° phase shifted using clock inversion | NORMAL, SHIFTED | NORMAL | DQSBUFD, DQSBUFE |
| DYNDEL_VAL | Value of Static Delay to the write control block | 0-127 | 0 | DQSBUFD, DQSBUFE |
| DYNDEL_CNTL | Attribute to enable Static or Dynamic DYNDEL | STATIC, DYNAMIC | DYNAMIC | DQSBUFD, DQSBUFE |
| NRZMODE* | Attribute used to select NRZMODE for DDR3 Memory | DISABLED<br>ENABLED | DISABLED | DQSBUFD |

***Note:** NRZMODE is only used with the DDR3 memory interface. This attribute affects the read data valid signal. When enabled, the read data valid signal toggles to indicate valid data.

## 18.8. Input DDR Primitives

The input DDR primitives represent the input DDR module used to capture both the generic DDR data and the DDR data coming from a memory interface. There are two available modes for the DDR input registers, one is used to implement DDRX1 gearing and the other is for DDRX2 gearing. The signals connected to the inputs of the IDDR are different for the DDR memory interface.

### 18.8.1. IDDRXD

This primitive implements the input register block in x1 gearing mode. This mode is used to implement DDR/DDR2 memory interfaces in the LatticeECP3 “E” and “EA” devices. It is also used to capture the generic DDRX1 data on the LatticeECP3 “E” device.

DDR registers are designed to use edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFF) when implementing a DDR or

DDR2 memory interface. For generic source synchronous DDR applications, this signal should connect to the edge clock input. The SCLK input should be connected to the system (FPGA) clock. DDRCLKPOL is an input from the DQS clock polarity tree. This signal is generated by the DQS Transition detect circuit in the corresponding DQSBUF block. The DDRCLKPOL signal is used to choose the polarity of the SCLK to the synchronization registers.

Figure 18.12 shows the primitive symbol and all the I/O ports.

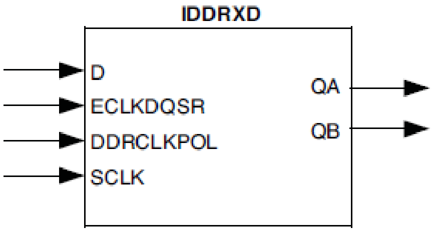


**Figure 18.12 IDDRXD Symbol**

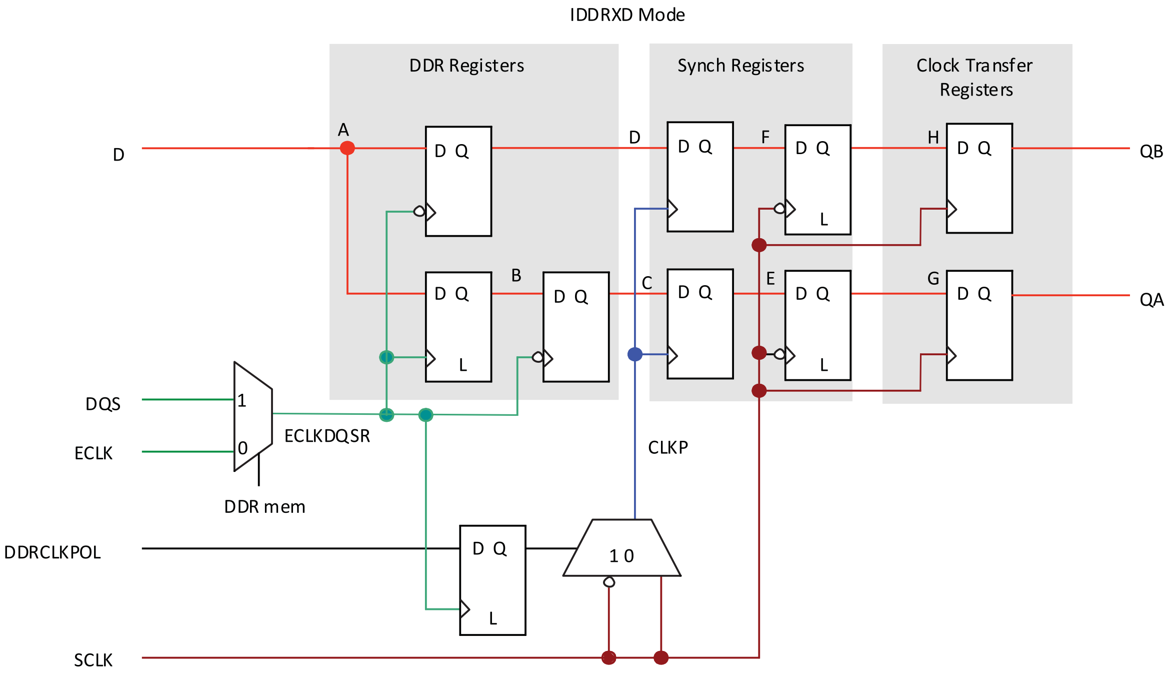
Table 18.9 provides a description of all I/O ports associated with the IDDRXD primitive.

**Table 18.9. IDDRXD Ports**

| Port Name | I/O | Definition |
|---|---|---|
| D | I | DDR data |
| ECLKDQSR | I | Phase-shifted DQS for DDR memory interfaces. ECLK for generic DDR interfaces. |
| SCLK | I | System clock |
| DDRCLKPOL | I | DDR clock polarity signal |
| QA | O | Data at positive edge of the clock |
| QB | O | Data at the negative edge of the clock |

**Note**: The DDRCLKPOL input to IDDRXD should be connected to the DDRCLKPOL output of the DQSBUFF or DQFBUFG modules.

Figure 18.13 shows the Input Register Block configured in the IDDRXD mode.



Note: Simplified diagram does not show CE, SET and RST details. All latches are transparent when low.

**Figure 18.13 Input Register Block in IDDRXD Mode**

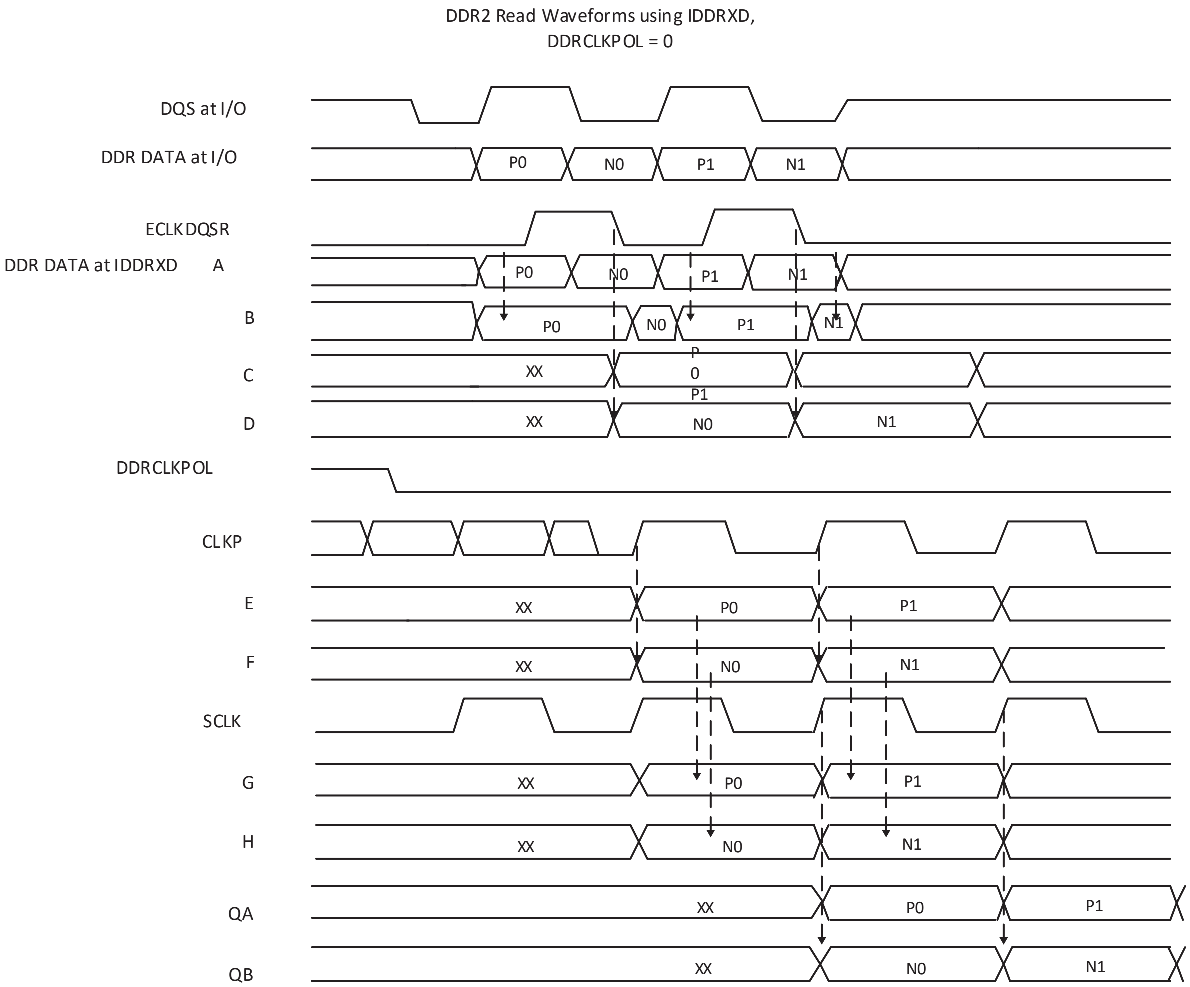


**Figure 18.14 IDDRXD Waveform DDRCLKPOL=0**

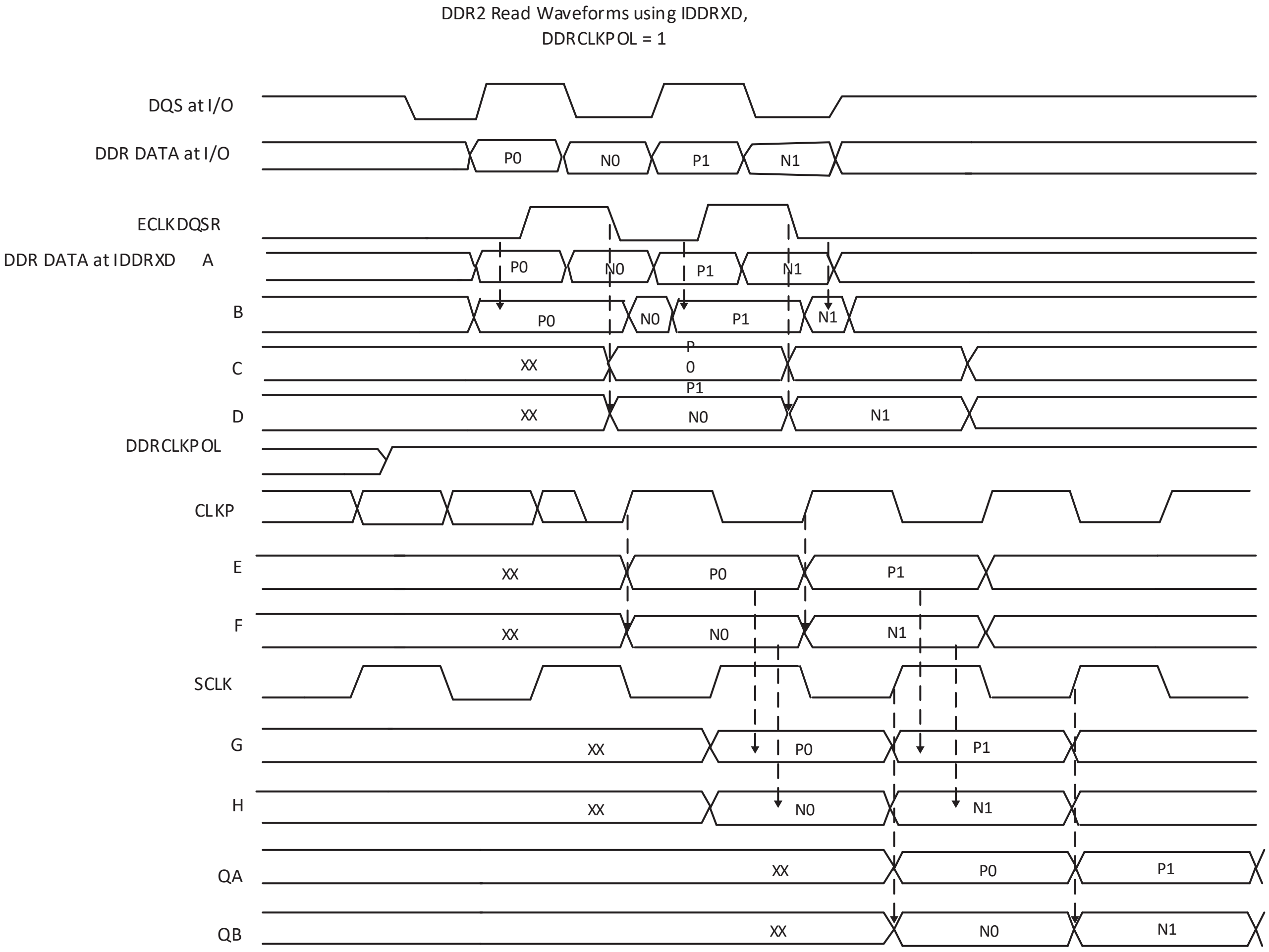


**Figure 18.15 IDDRXD Waveform DDRCLKPOL=1**

### 18.8.2. IDDRXD1

This primitive is a simplified version of IDDRXD without the DDRCLKPOL and ECLKDQSR signal for the "EA" devices. This also implements the input register block in x1 gearing mode for generic DDRX1 interfaces.

On "EA" devices, the DDR registers use the primary clock (SCLK) only. The SCLK input should be connected to the system (FPGA) clock. "EA" devices do not require the control signals from the DQSBUF module in the IDDRXD1 element, making it more flexible for placement than the "E" device.

Figure 18.16 shows the primitive symbol and all the I/O ports.

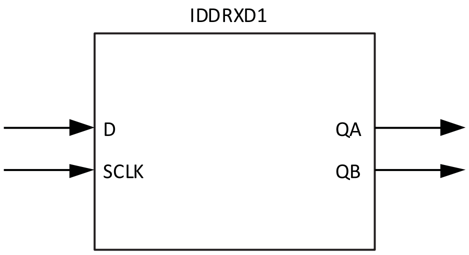


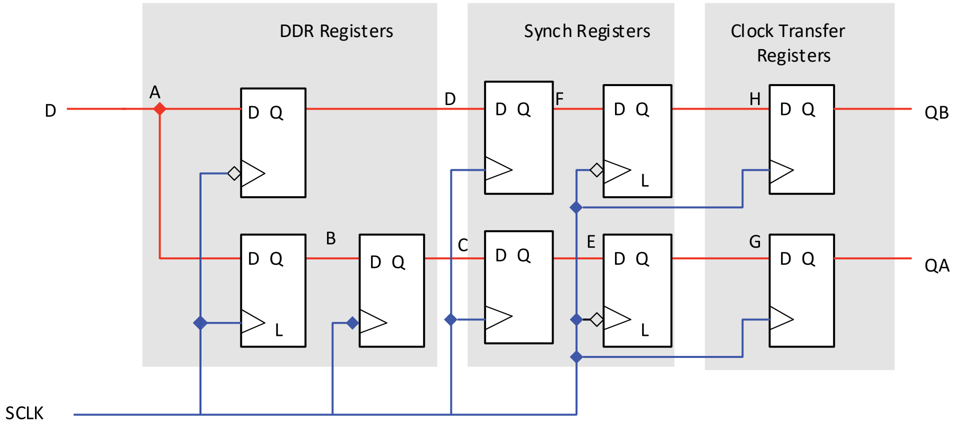
**Figure 18.16 IDDRXD1 Symbol ("EA" Devices)**

Table 18.10 provides a description of all I/O ports associated with the IDDRXD1 primitive.

**Table 18.10. IDDRXD1 Ports**

| Port Name | I/O | Definition |
|---|---|---|
| D | I | DDR data |
| SCLK | I | System clock |
| QA | O | Data at the positive edge of the clock |
| QB | O | Data at the negative edge of the clock |

Figure 18.17 shows the Input Register Block configured in the IDDRXD1 mode.



Note: Simplified version does not show CE/SET/RST details. All latches are transparent when LOW.

**Figure 18.17 Input Register Block in IDDRXD1 Mode ("EA" Devices)**

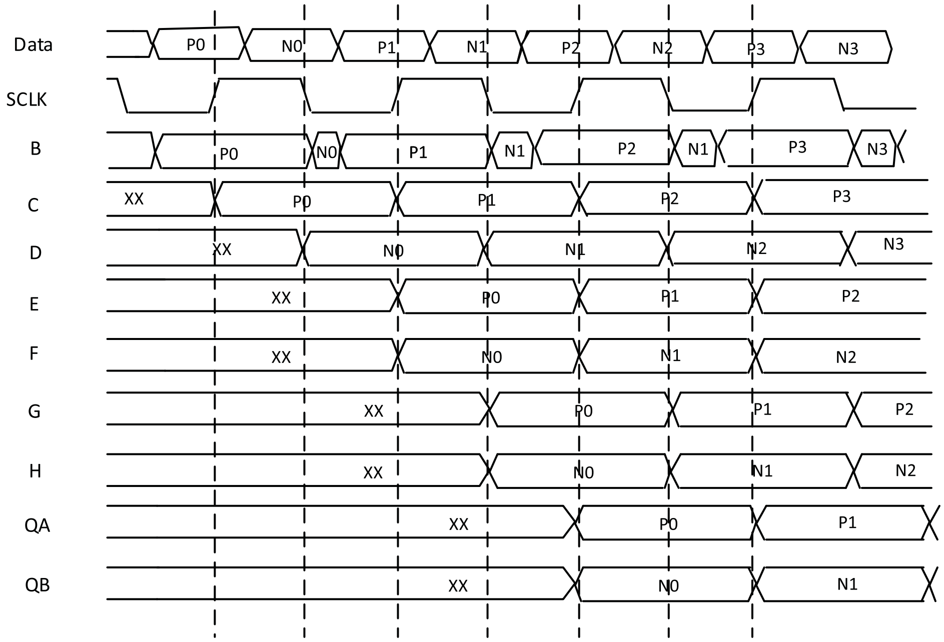


**Figure 18.18 IDDRXD1 Waveform**

### 18.8.3. IDDRX2D

This primitive implements the input register block in x2 gearing mode. This mode is used to implement DDR3 memory interface on the LatticeECP3 "E" and "EA" devices. It is also used on "E" devices to capture the generic DDRX2 Input data.

Figure 18.19 shows the IDDRX2D primitive symbol and all the I/O ports.

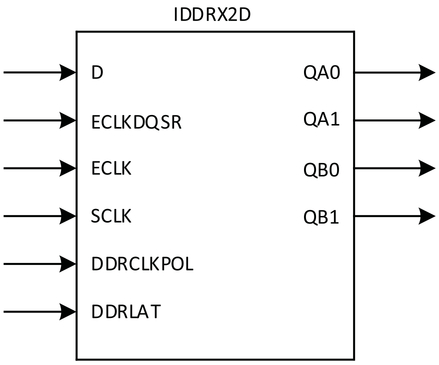


**Figure 18.19 IDDRX2D Symbol**

Table 18.11 provides a description of all I/O ports associated with the IDDRX2D primitive.

**Table 18.11. IDDRX2D Ports**

| Port Name | I/O | Definition |
|---|---|---|
| D | I | DDR Data |
| ECLKDQSR | I | Phase-shifted DQS for DDR memory interfaces. Edge clock for generic DDR interfaces. |
| ECLK | I | Edge Clock. Should be connected to DQS strobe for DDR3 memory interfaces. |
| SCLK | I | System clock running at one-half the ECLK or DQS signal. |
| DDRCLKPOL | I | DDR clock polarity signal |
| DDRLAT | I | DDR latch control signal |
| QA0, QA1 | O | Data at the positive edge of the clock. |
| QB0, QB1 | O | Data at the negative edge of the clock. |

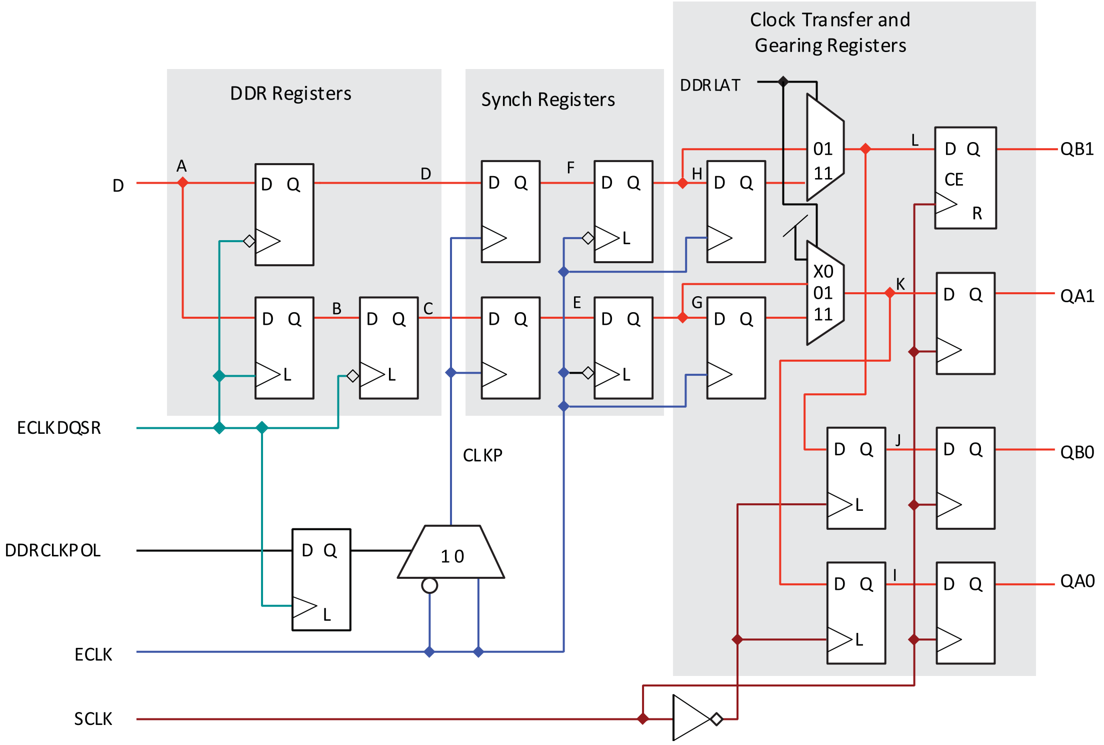
**Notes:**

1. The DDRCLKPOL input to IDDRX2D should be connected to the DDRCLKPOL output of the DQSBUFD for DDR3 memory interfaces or the DDRCLKPOL output of the DQSBUFE for generic DDRX2 interfaces.
2. The DDRLAT input to the IDDRX2D should be connected to the DDRLAT output of the DQSBUFD for DDR3 memory interfaces or the DDRLAT output of the DQSBUFE for generic DDRX2 interfaces.

Figure 18.20 shows the LatticeECP3 Input Register Block configured to function in the IDDRX2D mode.

The ECLKDQSR input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFD) when implementing a DDR3 memory interface. For generic source synchronous DDR applications, this signal should be connected to the high-speed source synchronous edge clock input. The ECLK input is connected to the edge clock. The SCLK input should be connected to the system (FPGA) clock. The SCLK should run at half the frequency of ECLK.

The DDRCLKPOL and DDRLAT inputs are generated by the DQS transition detect circuit in the corresponding DQSBUF block. The DDRCLKPOL signal is used to choose the polarity of the ECLK to the synchronization registers.DDRLAT is used to transfer data from the ECLK to the SCLK in the Clock Transfer Register block.



Notes:

1. Simplified version does not show CE/SET/RST details. All latches are transparent when LOW.
2. ECLKDQSR is connected to the DQS signal when in DDR memory mode. In DDR generic mode ECLKDQSR should be connected to the ECLK signal.

**Figure 18.20 Input Register Block in IDDRX2D Mode**

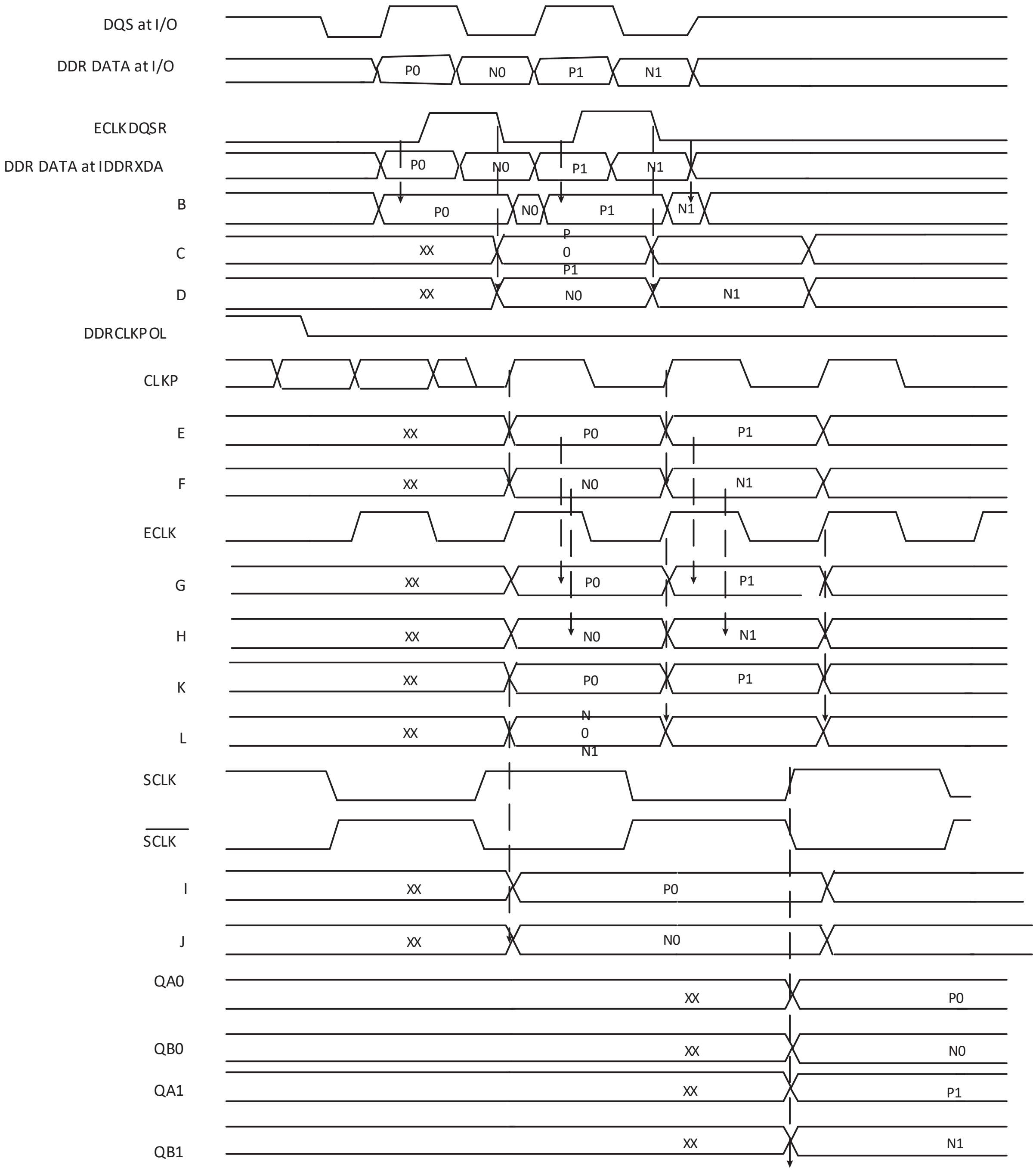


**Figure 18.21 IDDRX2D Waveform DDRLAT=0**

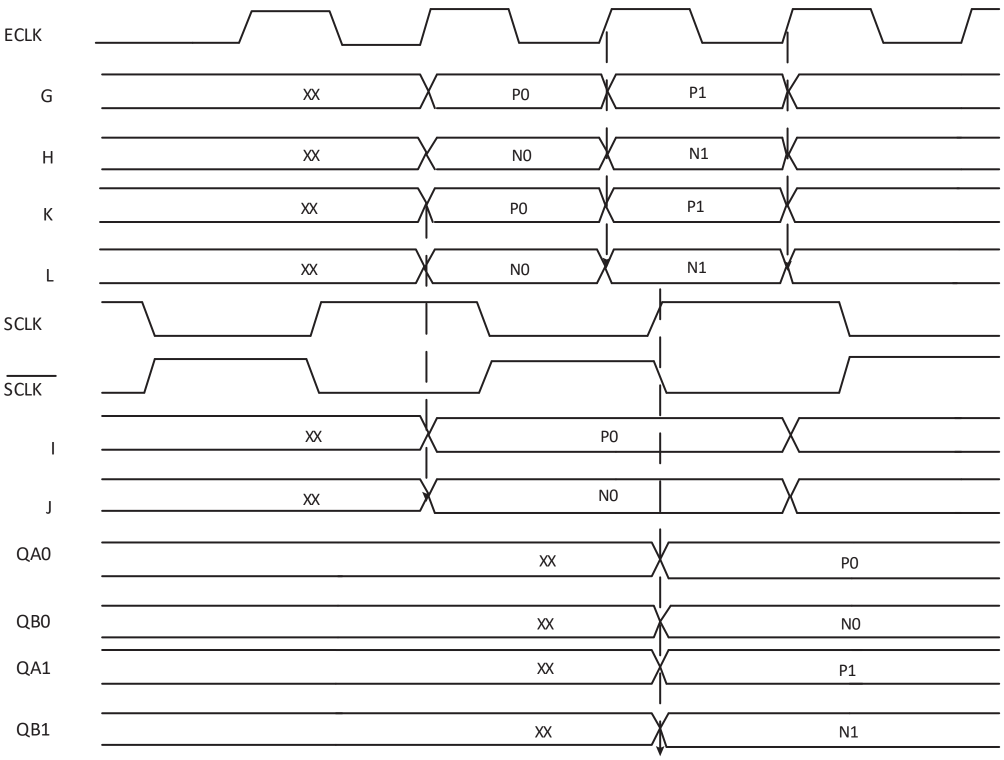


**Figure 18.22 IDDRX2D Waveform DDRLAT=1**

### 18.8.4. IDDRX2D1

This primitive is a simplified version of IDDRX2D without the DDRLAT, DDCLKPOL and the ECLKDQSR signals for the LatticeECP3 "EA" devices. It is used for input generic DDRX2 input data in "EA" devices.

In this case, the first stage of registers is clocked by the ECLK signal and the second stage is clocked by the SCLK signal. The "EA" device does not require the control signals from the DQSBUF module in the IDDRX2D1 element. This makes the "EA" device more flexible for placement than the "E" device.

Figure 18.23 shows the IDDRX2D1 primitive symbol and all the I/O ports.

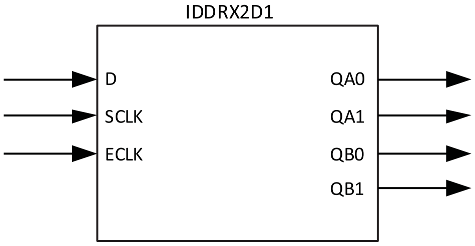


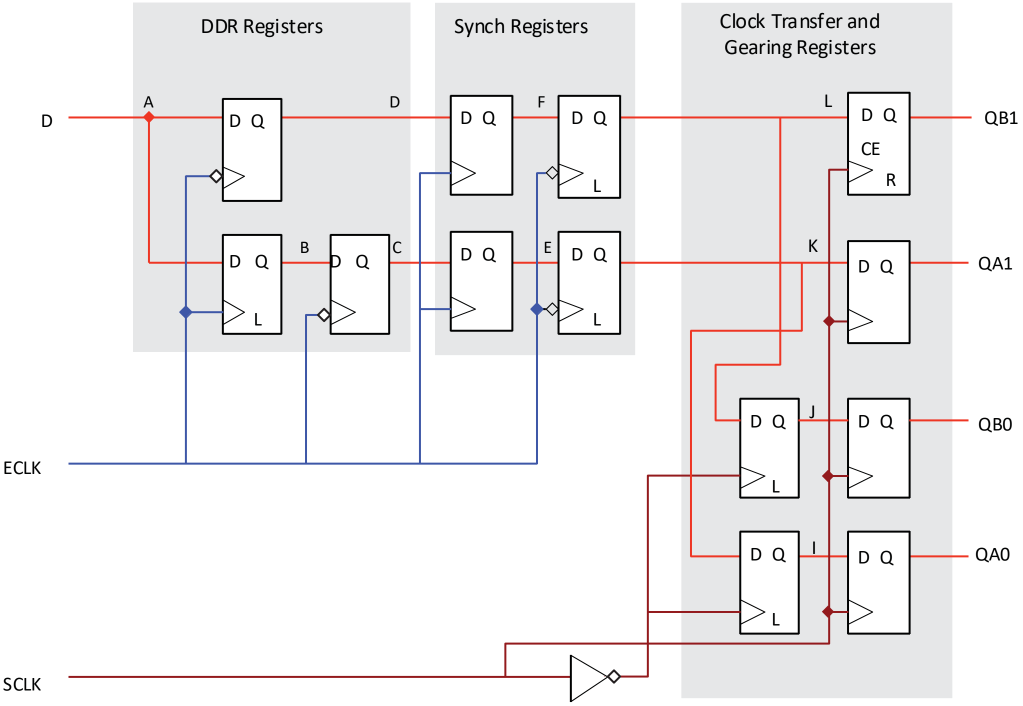
**Figure 18.23 IDDRX2D1 Symbol ("EA" Devices)**

Table 18.12 provides a description of all I/O ports associated with the IDDRX2D1 primitive.

**Table 18.12. IDDRX2D1 Ports**

| Port Name | I/O | Definition |
|---|---|---|
| D | I | DDR data |
| ECLK | I | Edge clock. Should be connected to DQS strobe for DDR3 memory interfaces. |
| SCLK | I | System clock running at one-half ECLK |
| QA0, QA1 | O | Data at the positive edge of the clock. |
| QB0, QB1 | O | Data at the negative edge of the clock. |

Figure 18.24 shows the LatticeECP3 Input Register Block configured to function in IDDRX2D1 mode. The ECLK input is connected to the edge clock. The SCLK input should be connected to the system (FPGA) clock. The SCLK should run at half the frequency of ECLK.



Note: Simplified version does not show CE/SET/RST details. All latches are transparent when LOW.

**Figure 18.24 Input Register Block in IDDRX2D1 Mode (“EA” Devices)**

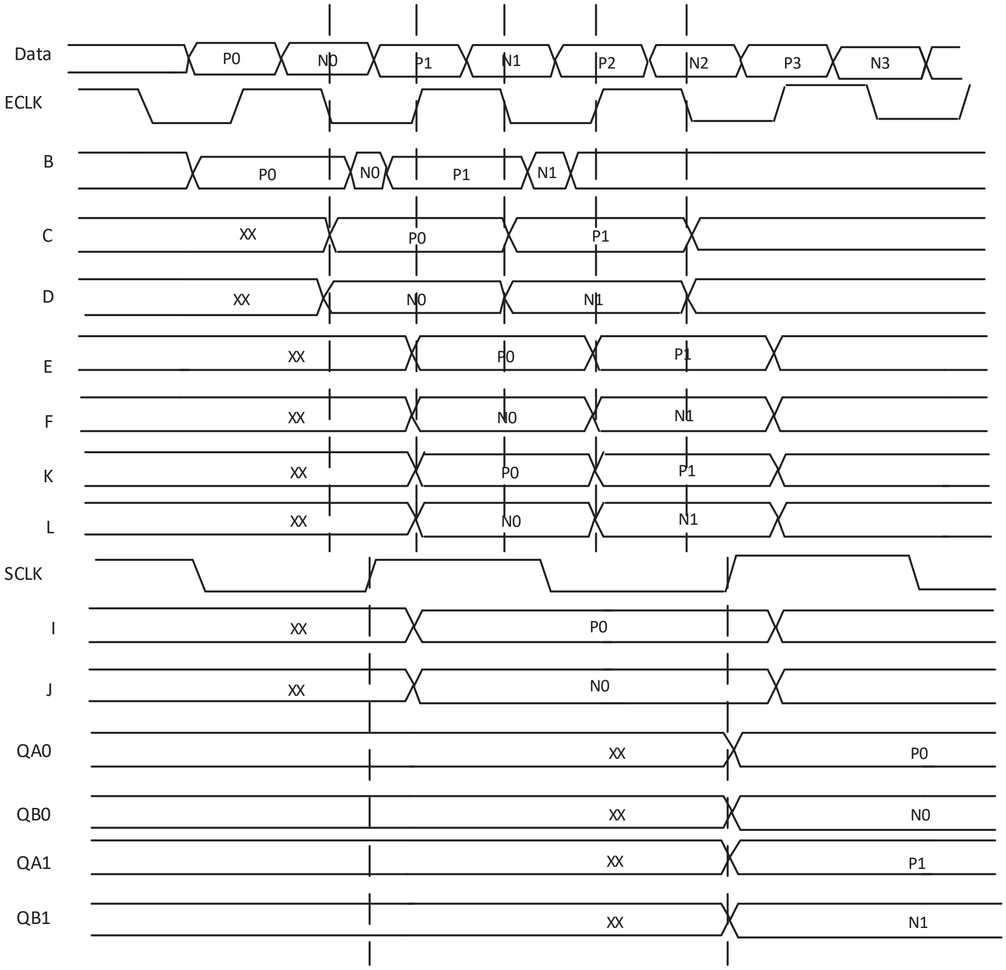


**Figure 18.25 IDDRX2D1 Waveform**

### 18.8.5. ECLKSYNCA

ECLKSYNCA is used in x2 gearing Tx interfaces to synchronize the signals generated from the ECLK after RESET. These signals include the SCLK, DQCLK0/DQCLK1 and DQSW for DDR memory interfaces.

This module will STOP the ECLK to the CLKDIV and DQSBUF modules until the RESET is released.

Asserting the STOP input of the ECLKSYNC stops the ECLK output. When the STOP signal is released, every clock toggling from the second rising edge of the ECLK input is output from this block. This block resides after the muxes for edge clock sources and before driving onto the actual edge clock.

Figure 18.26 shows the ECLKSYNCA primitive symbol.

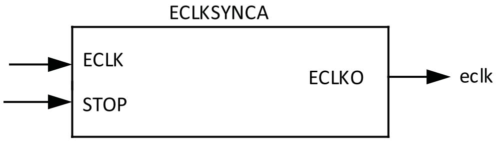


**Figure 18.26 ECLKSYNCA Symbol**

Table 18.13 lists the port descriptions of the ECLKSYNCA primitive.

**Table 18.13. ECLKSYNCA Port Descriptions**

| Port Name | I/O | Definition |
|---|---|---|
| ECLK | I | Edge clock input |
| STOP | I | Signal used to stop the edge clock |
| ECLKO | O | Edge clock output |

Figure 18.27 is a waveform that shows this operation.

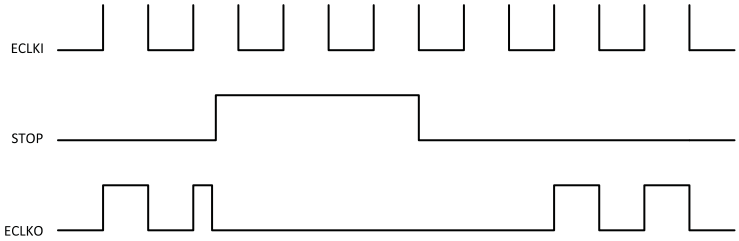


**Figure 18.27 ECLKSYNC Operation**

This stops the ECLK to the CLKDIV and DQSBUF until after these blocks are out if reset. By doing this, you can synchronize the ECLK, SCLK and the DQCLKs used in the ODDR module.

It is required that there is at least two clock cycles between the release of RESET and the release of the STOP input to ECLKSYNC. In the IPexpress-generated module, a soft IP consisting of two flip-flops is used to generate this delayed STOP signal to ECLKSYNC. The reset input to the CLKDIV and DQSBUF is used as an input to these two flip-flops. The clock input to these flip-flops must be slower than the ECLK.

Refer to the GDDRX2_TX.Aligned, GDDRX2_TX.DQSDLL.Centered and GDDRX2_TX.PLL.Centered interface descriptions in the High-Speed DDR Interface Details section.

## 18.8.6. DELAYC

Data going to the DDR registers can be optionally delayed using the delay block. The DELAYC block is used to compensate for clock injection delay times. The amount of the delay is determined by the software based on the type of interface implemented using the Interface ID attribute IDDRAPPS. Refer to Interface ID Attribute section for details. If an incorrect Interface ID is used for a given interface, then the DELAYC setting is incorrect. It is important that the correct Interface ID attribute be assigned for each interface to allow the software to set the correct value for DELAYC.

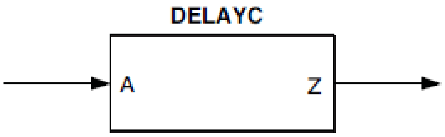


**Figure 18.28 DELAYC Symbol**

**Table 18.14. DELAYC Port Names**

| Port Name | I/O | Description |
|---|---|---|
| A | I | DDR input from sysIO™ buffer |
| Z | O | Delayed output |

### 18.8.7. DELAYB

Data going to the DDR registers can also be delayed the DELAYB block. Unlike the DELAYC block where the software controls the amount of data delay, the DELAYB block allows you to control the amount of data delay. This block receives 4-bit delay control. The 4-bit delay can be set by using static delay values or can be dynamically controlled by the user logic. DELAYB can only be used when the interface type is dynamic. See the Building Generic High-Speed Interfaces section for details.

The delay can be adjusted in the range of 38 ps to 45 ps per step. You can choose from two types of delay values:

- Dynamic – The delay value is controlled by the user logic using the inputs DEL[3:0] of the DELAYB block.
- User-Defined – In this mode, you can choose a static delay value from one of the 16 delay values. This ties the inputs DEL[3:0] of the DELAYB block to a fixed value depending on the value chosen.

Figure 18.29 shows the primitive symbol for the DELAYB mode.

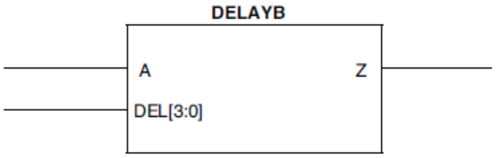


**Figure 18.29 DELAYB Symbol**

Table 18.15 lists the port names and descriptions for the DELAYB primitive.

**Table 18.15. DELAYB Port Names**

| Port Name | I/O | Definition |
|---|---|---|
| A | I | DDR input from the sysIO buffer |
| DEL (0:3) | I | Delay inputs |
| Z | O | Delay DDR data |

## 18.9. Output DDR Primitives

The output DDR primitives represent the output DDR module used to generate both the generic DDR output data and the DDR memory interface data. There are two available modes for DDR output registers. One is used to implement DDRX1 gearing and the other for DDRX2 gearing.

### 18.9.1. ODDRXD

This primitive implements the output register block in x1 gearing mode. This mode is used to implement DDR/DDR2 memory interfaces on the "E" and "EA" devices. It is also used to generate the generic DDRX1 data on "E" devices.

Figure 18.30 shows the ODDRXD primitive symbol and its I/O ports.

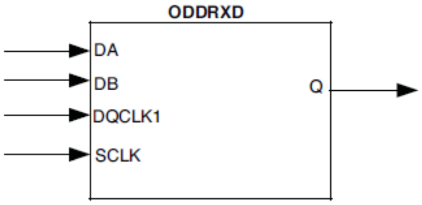


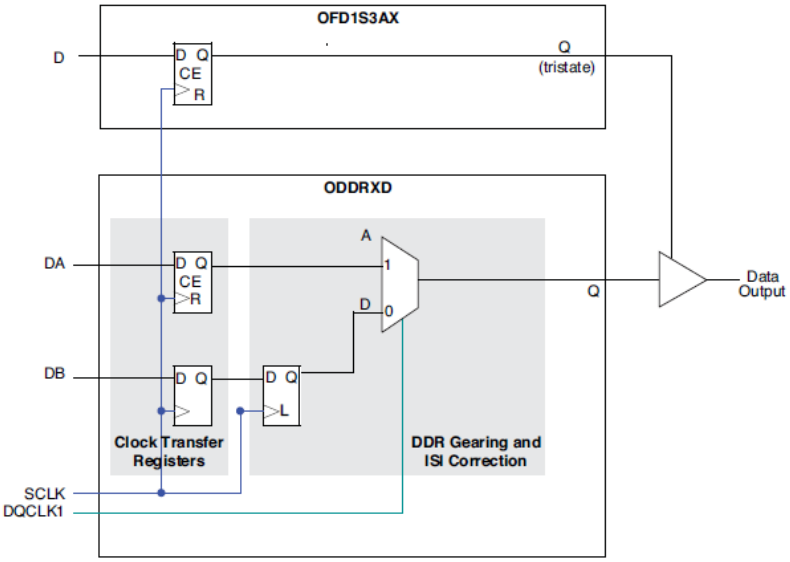
**Figure 18.30 ODDRXD Symbol**

Table 18.16 provides a description of all I/O ports associated with the ODDRXD primitive.

Table 18.16. ODDRXD Ports

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DA | I | Data at the positive edge of the clock |
| DB | I | Data at the negative edge of the clock |
| DQCLK1 | I | Clock output at frequency of SCLK used for output gearing |
| Q | O | DDR data output |

Figure 18.31 shows the LatticeECP3 Output Register Block configured in the ODDRXD mode.



Note: All latches are transparent when LOW.

Figure 18.31 Output Register Block in ODDRXD Mode

**Note:** Tri-state control for ODDRXD can only be implemented using the OFD1S3AX module. If this module is not implemented your design, then the software infers this module. The clock used in the OFD1SAX should be the same as the one used in the ODDRXD module. This module does not support tri-state inversion.

Figure 18.32 shows the ODDRXD timing waveform

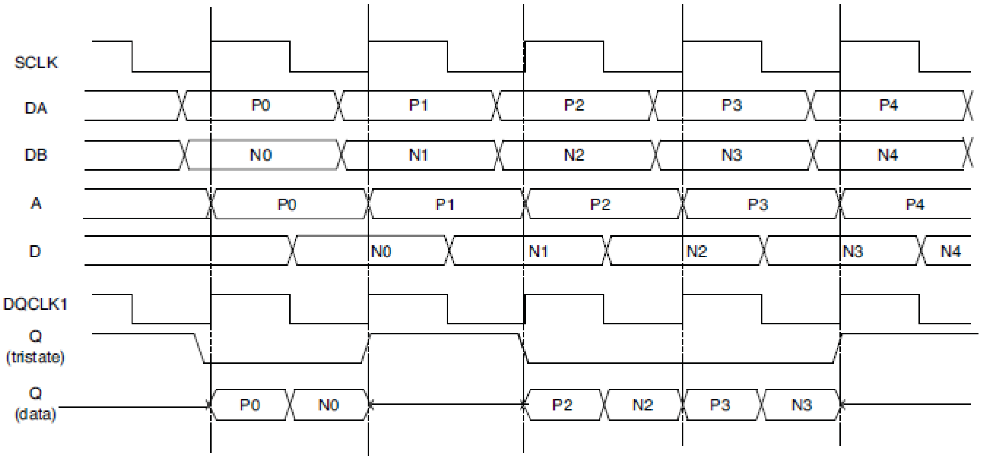


Figure 18.32 ODDRXD Waveform

### 18.9.2. ODDRXD1

This element is used to generate the generic DDRX1 data on "EA" devices. The ODDRXD1 in the "EA" device does not require the DQCLK1 control signal from the DQSBUF block. This makes the "EA" device more flexible for placement than "E" devices.

Figure 18.33 shows the ODDRXD1 primitive symbol and its I/O ports.

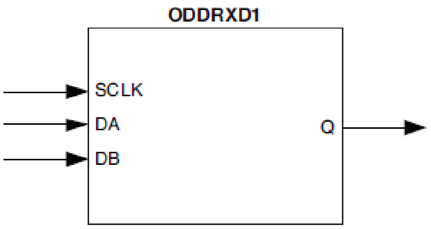


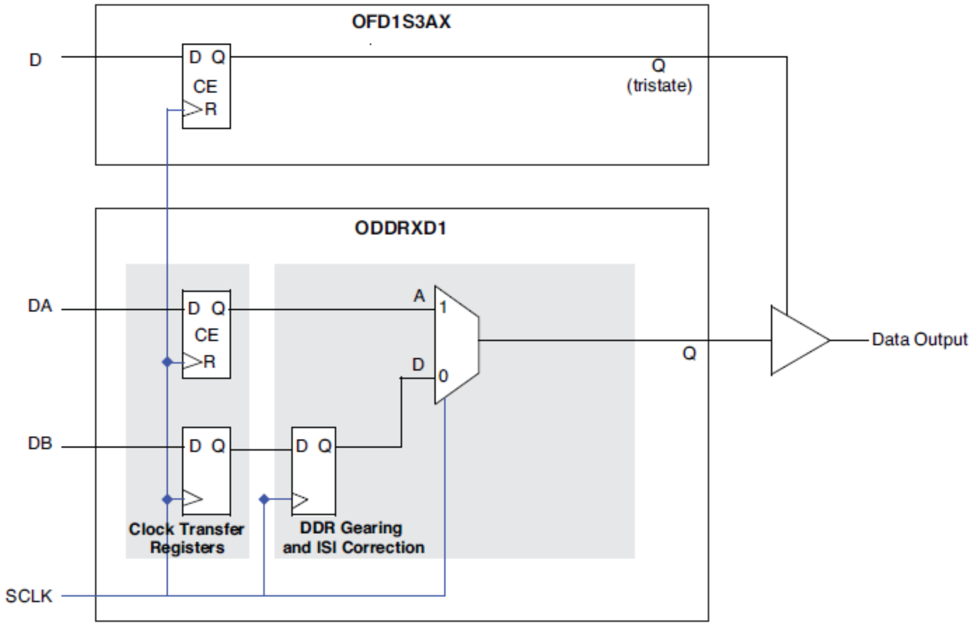
**Figure 18.33 ODDRXD1 Symbol ("EA" Devices)**

Table 18.17 provides a description of all I/O ports associated with the ODDRXD1 primitive.

**Table 18.17. ODDRXD1 Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DA | I | Data at the positive edge of the clock |
| DB | I | Data at the negative edge of the clock |
| Q | O | DDR data output |

Figure 18.34 shows the LatticeECP3 Output Register Block configured in the ODDRXD1 mode.



Note: All latches are transparent when LOW.

**Figure 18.34 Output Register Block in ODDRXD1 Mode ("EA" Devices)**

**Note:** Tri-state control for ODDRXD1 can only be implemented using the OFD1S3AX module. If this module is not implemented in your design, then the software infers this module. The clock used in the OFD1SAX should be the same as the one used in the ODDRXD1 module. This module does not support tri-state inversion.

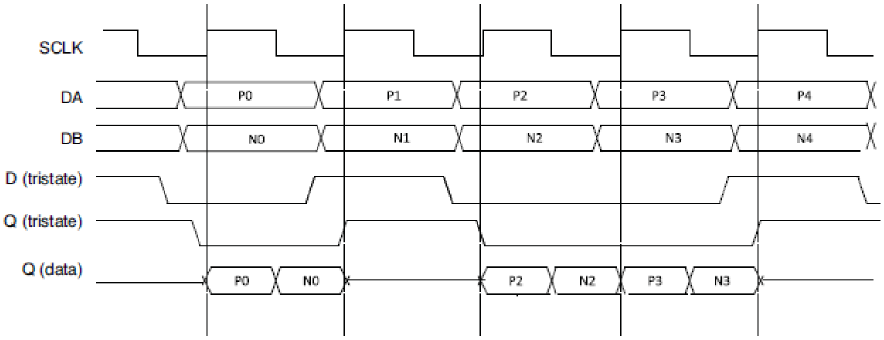


**Figure 18.35 ODDRXD1 Waveform**

### 18.9.3. ODDRX2D

The ODDRX2D primitive implements the output register for DDR3 memory and generic DDRX2 write functions. Figure 18.36 shows the ODDRX2D primitive symbol and its I/O ports.

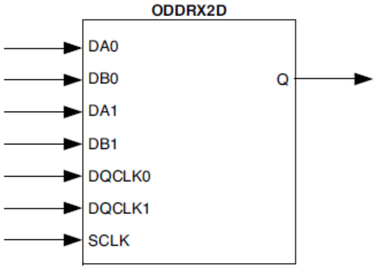


**Figure 18.36 ODDRX2D Symbol**

Table 18.18 provides a description of all I/O ports associated with the ODDRX2D primitive.

**Table 18.18. ODDRX2D Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DA0 | I | First data at the positive edge of the clock |
| DB0 | I | First data at the negative edge of the clock |
| DA1 | I | Second data at the positive edge of the clock |
| DB1 | I | Second data at the negative edge of the clock |
| DQCLK0 | I | Clock Output at frequency of SCLK used for output gearing |
| DQCLK1 | I | Clock output at frequency of SCLK used for output gearing (90° shifted from DQCLK0) |
| Q | O | DDR data output |

### 18.9.4. ODDRTDQA

The ODDRTDQA primitive implements the tri-state register block for DDR3 memory and generic x2 DDR write functions. Figure 18.37 shows the ODDRTDQA primitive symbol and its I/O ports.

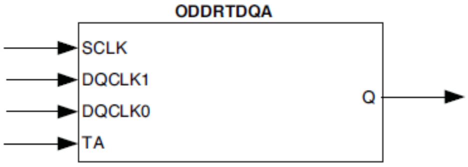


**Figure 18.37 ODDRTDQA Symbol**

Table 18.19 provides a description of all I/O ports associated with the ODDRTDQA primitive.

**Table 18.19. ODDRTDQA Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| TA | I | Tri-state input |
| DQCLK0 | I | Clock output at frequency of SCLK used for output gearing |
| DQCLK1 | I | Clock output at frequency of SCLK used for output gearing (90° shifted from DQCLK0) |
| Q | O | DDR tri-state output |

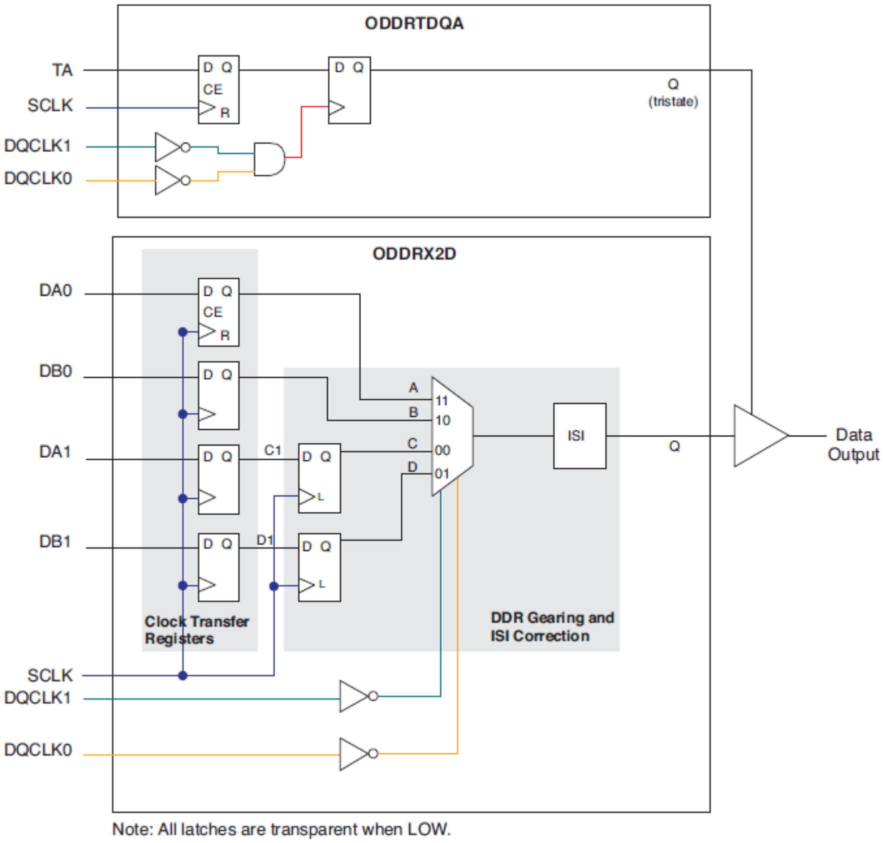


**Figure 18.38 Output Register Block in ODDRX2D/ODDRTDQA Mode**

Figure 18.38 shows the LatticeECP3 Output Register Block configured in the ODDRX2D and ODDRTDQA tri-state modes.

**Note:** Tri-state control for ODDRX2D can only be implemented using the ODDRTDQA module. The clock used in the ODDRTDQA should be the same as the one used in the ODDRX2D module. This module does not support tri-state inversion.

On the ODDRX2, it is required that the SCLK be in correct phase with DQCLK1 for the data to be captured correctly inside the ODDRX2. The figure below explains the correct relationship between the SCLK and DQCLK1. SCLK edge must be delayed to occur after the DQCLK1 edge for the data to be captured without any glitches.

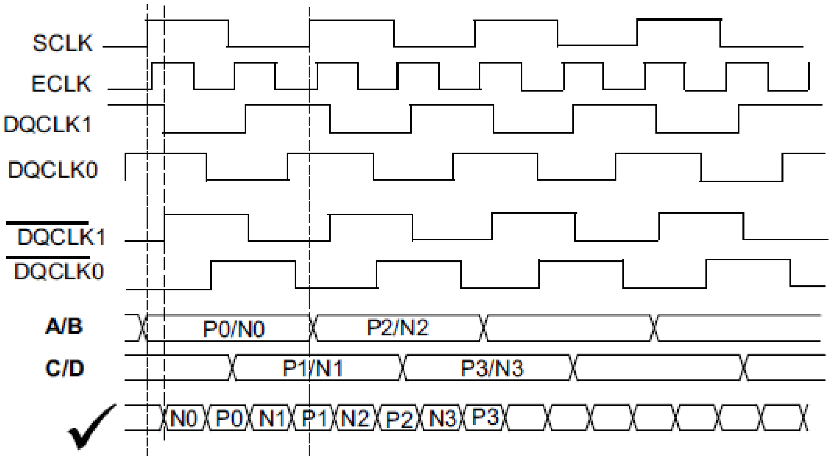


Figure 18.39 Correct DQCLK1 Polarity

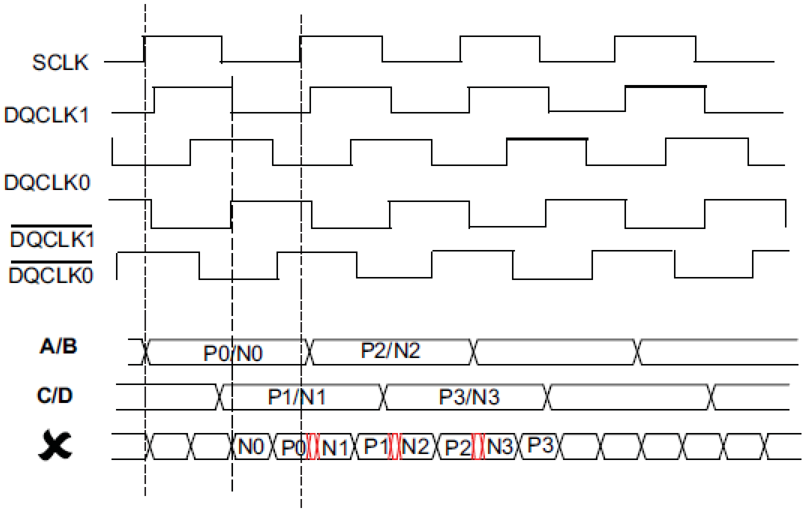


Figure 18.40 Incorrect DQCLK1 Polarity

The soft IP, ECLKSYNCA, CLKDIVB, DQSBUFE1 and the SCLK routing delay ensures the correct phase relationship between SCLK and DQCLK inside the ODDRX2 module. You must generate the interface using IPexpress to guarantee this delay is achieved.

### 18.9.5. ODDRXDQSA

The ODDRXDQSA primitive implements the output register for generating the DQS strobe signal for DDR and DDR2 memory.

Figure 18.41 shows the ODDRXDQSA primitive symbol and its I/O ports.

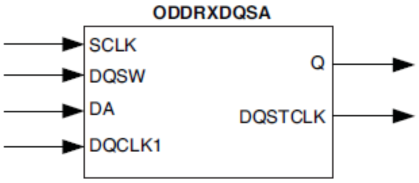


Figure 18.41 ODDRXDQSA Symbol

Table 18.20 provides a description of all I/O ports associated with the ODDRXDQSA primitive.

**Table 18.20. ODDRXDQSA Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DA | I | Data input |
| DQSW | I | DQS write clock |
| DQCLK1 | I | Clock output at frequency of SCLK used for output gearing |
| DQSTCLK | O | DQS tri-state clock |
| Q | O | DQS data output |

## 18.9.6. ODDRTDQSA

The ODDRTDQSA primitive implements the tri-state register block for DDR/DDR2 and DDR3 memory DQS output clock generation.

Figure 18.42 shows the ODDRTDQSA primitive symbol and its I/O ports.

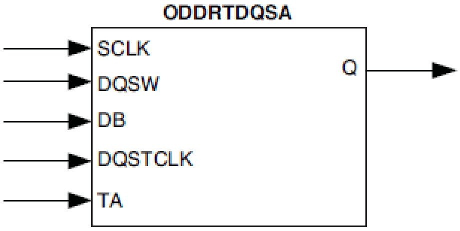


**Figure 18.42 ODDRTDQSA Symbol**

Table 18.21 provides a description of all I/O ports associated with the ODDRTDQSA primitive.

**Table 18.21. ODDRTDQSA Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DB | I | Data input |
| DQSW | I | DQS write clock |
| DQSTCLK | I | DQS tri-state Clock |
| TA | I | Tri-state input |
| Q | O | DQS tri-state output |

Figure 18.43 shows the LatticeECP3 Output Register Block configured in the ODDRXDQSA and ODDRTDQSA tri-state modes.

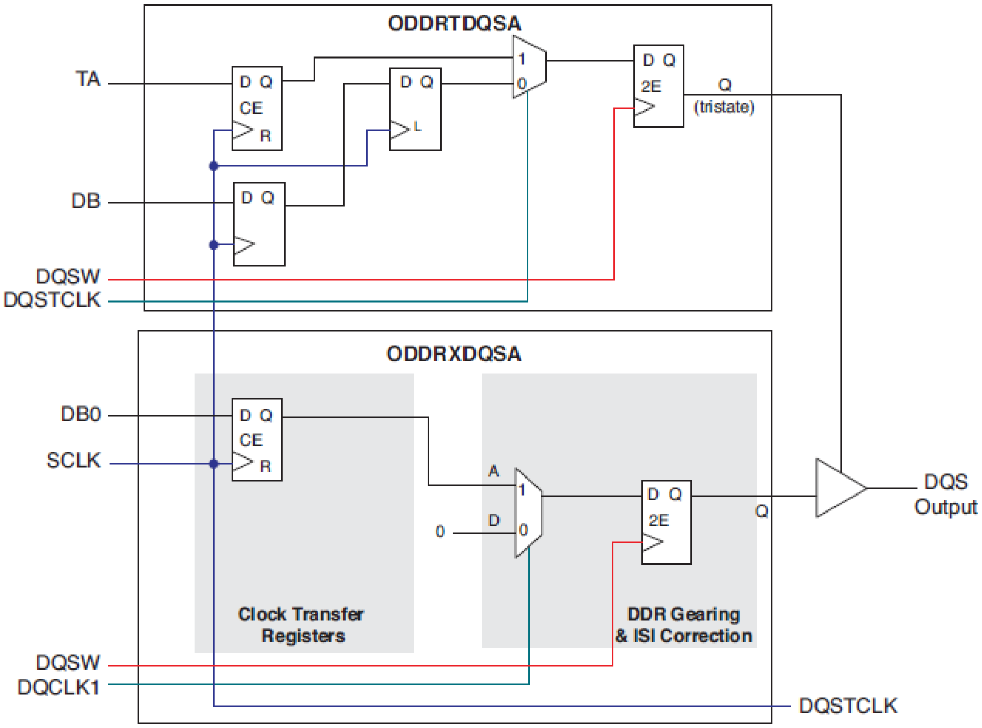


Figure 18.43 Output Register Block in ODDRXDQSA/ODDRTDQSA Mode

**Note:** Tri-state control for ODDRXDQSA can only be implemented using the ODDRTDQSA module. The clock used in the ODDRTDQSA should be the same as the one used in the ODDRXDQSA module. This module does not support tri-state inversion.

Figure 18.44 shows the ODDRXDQSA and ODDRTDQSA timing waveform.

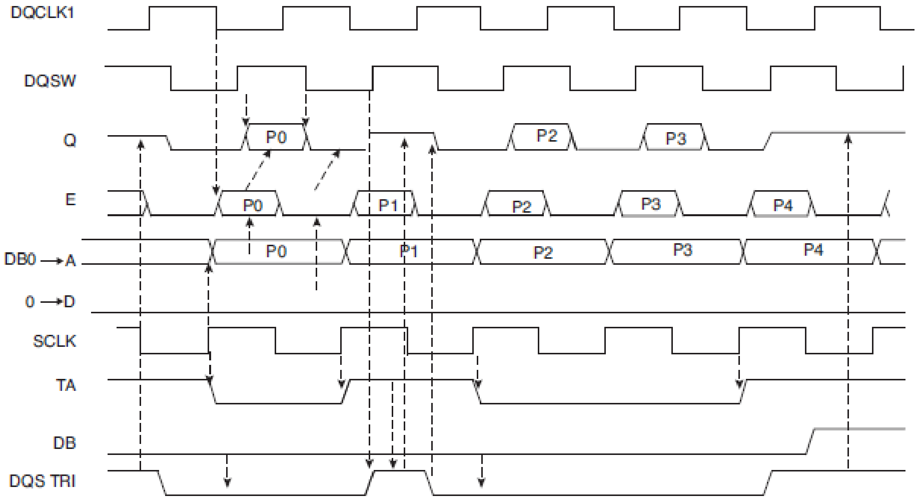


Figure 18.44 ODDRXDQSA/ODDRTDQSA Waveform

### 18.9.7. ODDRX2DQSA

The ODDRX2DQSA primitive implements the output register for generating the DQS strobe signal for DDR3 memory interfaces.

Figure 18.45 shows the ODDRX2DQSA primitive symbol and its I/O ports.

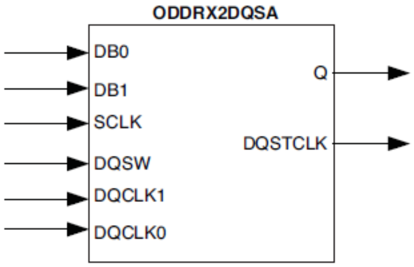


**Figure 18.45 ODDRX2DQSA Symbol**

Table 18.22 provides a description of all I/O ports associated with the ODDRX2DQSA primitive.

**Table 18.22. Table 33 ODDRX2DQSA Ports**

| Port Name | I/O | Definition |
|---|---|---|
| SCLK | I | System CLK or ECLK |
| DB0 | I | Data input |
| DB1 | I | Data input |
| DQSW | I | DQS write clock |
| DQCLK0 | I | Clock output at frequency of SCLK used for output gearing |
| DQCLK1 | I | Clock output at frequency of SCLK and shifted 90°, used for output gearing |
| DQSTCLK | O | DQS tri-state clock |
| Q | O | DDR data output |

Figure 18.46 shows the LatticeECP3 Output Register Block configured in the ODDRX2DQSA and ODDRTDQSA tri-state mode.

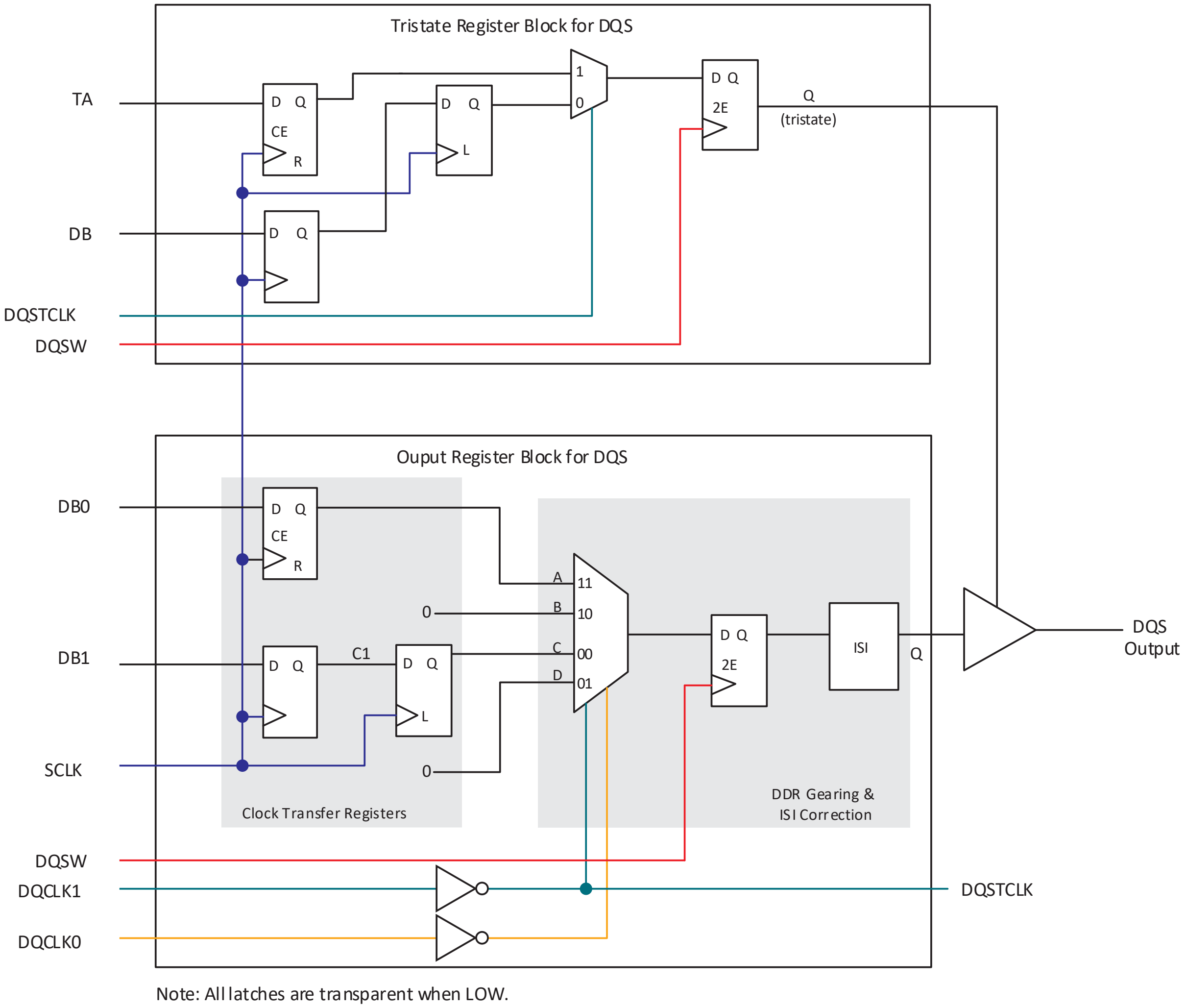


**Figure 18.46 Output Register Block in ODDRX2DQSA/ODDRTDQSA Mode**

**Note:** Tri-state control for ODDRX2DQSA can only be implemented using the ODDRTDQSA module. The clock used in the ODDRTDQSA should be the same as the one used in the ODDRX2DQSA module. This module does not support tri-state inversion.

Figure 18.47 shows the ODDRX2DQSA timing waveform.

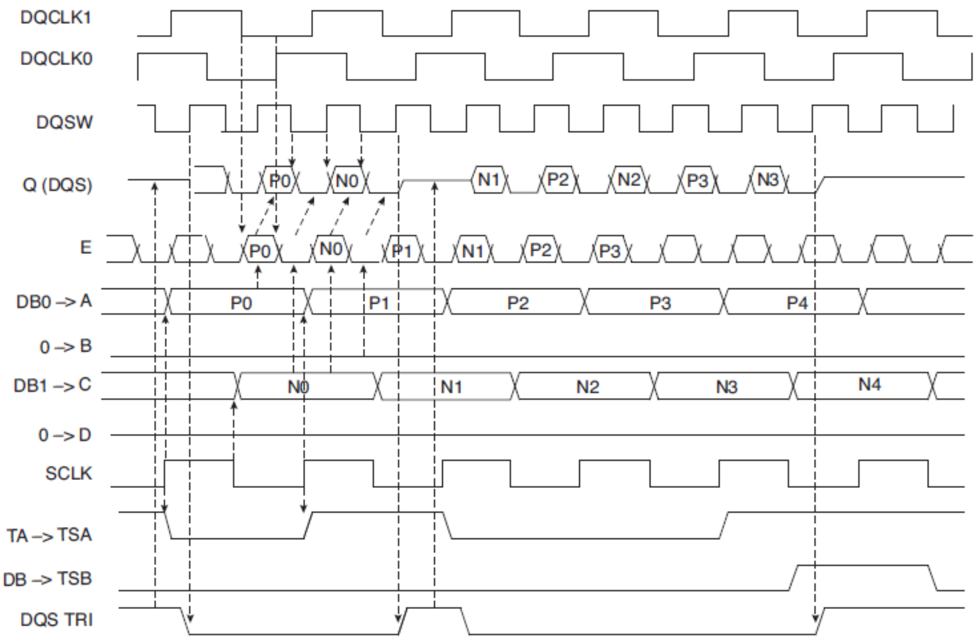


**Figure 18.47 ODDRX2DQSA/ODDRTDQSA Waveform**

## 18.10. Interface ID Attribute

This attribute provides an ID setting for each of the high-speed interfaces in the LatticeECP3 "EA" device. IDDRAPPS attribute is used for input interfaces and ODDRAPPS attribute is used for output interfaces. The value for these is pre-determined for each high-speed DDR interfaces. These attributes are set to the correct values when the interfaces are generated by IPexpress. If an IDDRAPPS or ODDRAPPS attribute is not set for a given interface, the software errors out. If an attribute value is not set correctly for a given interface, then the wrong data delay is used in the DELAYC element for that interface.

The IDDRAPPS and ODDRAPPS attributes are strings. They are only generated for "EA" devices.

**Table 18.23. Interface ID (IDDRAPPS/ODDRAPPS) Attribute Values**

| Interfaces Name | Interface ID (Primitive: ATTRIBUTE = Value)[1, 2] |
|---|---|
| **Generic Interfaces** | |
| GIREG_RX.SCLK | N/A |
| GDDRX1_RX.SCLK.Aligned | IDDRXD1: IDDRAPPS = SCLK_ALIGNED |
| GDDRX1_RX.SCLK.Centered | IDDRXD1: IDDRAPPS = SCLK_CENTERED |
| GDDRX1_RX.DQS.Aligned | IDDRXD: IDDRAPPS = DQS_ALIGNED |
| GDDRX1_RX.DQS.Centered | IDDRXD: IDDRAPPS = DQS_CENTERED |
| GDDRX2_RX.ECLK.Aligned | IDDRX2D1: IDDRAPPS = ECLK_ALIGNED |
| GDDRX2_RX.ECLK.Centered | IDDRX2D1: IDDRAPPS = ECLK_CENTERED |
| GDDRX2_RX.DQS.Aligned | IDDRX2D: IDDRAPPS = DQS_ALIGNED |
| GDDRX2_RX.DQS.Centered | IDDRX2D: IDDRAPPS = DQS_CENTERED |
| GDDRX2_RX.ECLK.Dynamic | IDDRX2D1: IDDRAPPS = ECLK_DYNAMIC |
| GDDRX2_RX.DQS.Dynamic | IDDRX2D: IDDRAPPS = DQS_DYNAMIC |
| GDDRX2_RX.PLL.Dynamic | IDDRX2D1: IDDRAPPS = PLL_DYNAMIC |
| GOREG_TX.SCLK | ODDRXD1: ODDRAPPS = SCLK_ALIGNED |
| GDDRX1_TX.SCLK.Centered | ODDRXD1: ODDRAPPS = SCLK_CENTERED (CLOCK) ODDRXD1: ODDRAPPS = SCLK_ALIGNED (DATA) |
| GDDRX1_TX.SCLK.Aligned | ODDRXD1: ODDRAPPS = SCLK_ALIGNED (CLOCK) ODDRXD1: ODDRAPPS = SCLK_ALIGNED (DATA) |

| Interfaces Name | Interface ID (Primitive: ATTRIBUTE = Value)[1, 2] |
|---|---|
| GDDRX1_TX.DQS.Centered | ODDRXDQSA: ODDRAPPS = DQS_CENTERED (CLOCK) ODDRXD: ODDRAPPS = DQS_ALIGNED (DATA) |
| GDDRX2_TX.Aligned | ODDRX2D: ODDRAPPS = ECLK_ALIGNED (CLOCK) ODDRX2D: ODDRAPPS = ECLK_ALIGNED (DATA) |
| GDDRX2_TX.DQSDLL.Centered | ODDRX2DQSA: ODDRAPPS = DQS_CENTERED (CLOCK) ODDRX2D: ODDRAPPS = DQS_ALIGNED (DATA) |
| GDDRX2_TX.PLL.Centered | ODDRX2D: ODDRAPPS = ECLK_CENTERED (CLOCK) ODDRX2D: ODDRAPPS = ECLK_ALIGNED (DATA) |
| GDDRX1_RX.ECLK.Aligned | N/A: not a LatticeECP3 “EA” configuration. |
| GDDRX1_RX.ECLK.Centered | N/A: not a LatticeECP3 “EA” configuration. |
| **DDR Memory Interfaces** | |
| DDR MEM | ODDRXDQSA: ODDRAPPS = DDR_MEM_DQS IDDRXD: IDDRAPPS = DDR_MEM_DQ ODDRXD: ODDRAPPS = DDR_MEM_DQ |
| DDR2 MEM | ODDRXDQSA: ODDRAPPS = DDR2_MEM_DQS IDDRXD: IDDRAPPS = DDR2_MEM_DQ ODDRXD: ODDRAPPS = DDR2_MEM_DQ |

**Notes:**

1. Attribute should be assigned on the corresponding IDDRX/ODDRX primitives.
2. Interface IDs are only valid for LatticeECP3 “EA” devices.

## 18.11. ISI Calibration

ISI correction is only available in the ODDRX2D or ODDRX2DQSA modes on the left and right sides of the device. ISI calibration settings exist once per output, so each I/O in a DQS-12 group may have a different ISI calibration setting.

The ISI Calibration is set using the ISI_CAL attribute. Table 18.24 shows the values that can be set for this attribute.

**Table 18.24. ISI Calibration Attribute**

| Attribute | Description | Values | Software Default |
|---|---|---|---|
| ISI_CAL | Used to set the ISI Correction values | BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7 | BYPASS |

The ISI block extends output signals at certain times, as a function of recent signal history, so it can be read at the output signal’s destination. If the output pattern consists of long strings of 0s to long strings of 1s, there are no delays on output signals. However, if there are quick, successive transitions from 010, the block stretches out the binary 1. This is because the long trail of 0s cause these symbols to interfere with the logic 1. Likewise, if there are quick, successive transitions from 101, the block stretches out the binary 0.

This block is controlled by a 3-bit delay “stretching” control, set in the DQS logic section. There are eight settings in the range from “BYPASS” to “DEL7”.

# 19. Migrating Designs from LatticeECP3 "E" to "EA"

This section lists the changes in design required when moving from LatticeCP3 "E" device to an "EA" device. The LatticeECP3-150EA device was designed as an "E" device in ispLEVER 7.2 SP2. These changes are also required if moving a LatticeECP3-150EA design from ispLEVER 7.2 SP2 to ispLEVER 8.0 or later versions. It is required that all LatticeECP3-150EA designs be implemented in ispLEVER 8.0 or newer software. Refer to the section Migrating Designs from ispLEVER 7.2 SP2 to ispLEVER 8.0 to see other changes required when you are moving designs from ispLEVER 7.2 SP2 to ispLEVER 8.0 or later versions of the software. The same changes apply if moving from ispLEVER 7.2 to the Lattice Diamond design software as well.

A summary of the differences between the "E" and "EA" devices are listed below:

- DDR x1 interfaces on "E" devices use ECLK (edge clock) as the clock input limiting the number of interfaces to 1 per side. DDR x1 interfaces on "EA" devices use the SCLK clock input, so you can have more than one interface per side of the device.
- "E" device requires that DQSBUF be used to implement both x1 and x2 input and output DDR functions. "EA" devices do not require the DQSBUF to implement the input and x1 output DDR functions. X2 output DDR functions require the use of DQSBUF.
- The "E" devices require the data pins to be grouped into DQS groups so that every 10 data bits are locked to a DQS group. This grouping is not required on the "EA" devices for inputs and 1x output interfaces. 2x output interfaces on "EA" would require DQS grouping.
- The primitives used for Generic DDR input and output functions are different between "E" and "EA".
- All "EA" designs require that the IDDRAPPS/ODDRAPPS be assigned to each interface which is not required on the "E" device. Refer to the section Interface ID Attribute for a description of this attribute. IPexpress-generated modules contain this attribute.
- In DDR and DDR2 memory, the implementation is mostly the same between "E" and "EA" devices. But since DDR and DDR2 use generic DDR to generate the output CLKP/CLKN signal, these must use new primitives on the "EA" device. In addition, all the primitives require the IDDRAPPS/ODDRAPPS attributes to be added for the "EA" device.

Refer to the section DDR Software Primitives and Attributes for a detailed description of the primitives. Table 19.1 lists the clocking differences between "E" and "EA" devices for each interface. Table 19.2 lists the differences in library elements used for each interfaces. Refer to the High-Speed DDR Interface Details section for block diagrams of each of the interfaces in the tables below.

**Table 19.1. "E" to "EA" Clocking Differences**

| "E" Interface | Clocking Resource | DQS Grouping for Pins | Equivalent "EA" Interface | Clocking Resource | DQS Grouping for Pins |
|---|---|---|---|---|---|
| GIREG_RX.SCLK | SCLK | No | GIREG_RX.SCLK | SCLK | No |
| GDDRX1_RX.ECLK.Aligned | ECLK | Yes | GDDRX1_RX.SCLK.Aligned | SCLK | No |
| GDDRX1_RX.ECLK.Centered | ECLK | Yes | GDDRX1_RX.SCLK.Centered | SCLK | No |
| GDDRX1_RX.DQS.Aligned | DQS Tree | Yes | GDDRX1_RX.DQS.Aligned | DQS Tree | Yes |
| GDDRX1_RX.DQS.Centered | DQS Tree | Yes | GDDRX1_RX.DQS.Centered | DQS Tree | Yes |
| GDDRX2_RX.ECLK.Aligned | ECLK | Yes | GDDRX2_RX.ECLK.Aligned | ECLK | No |

**Table 19.2. "E" to "EA" Primitive Changes**

| "E" Interface | Primitives Required | Equivalent "EA" Interface | Primitives Used |
|---|---|---|---|
| GIREG_RX.SCLK | IFS1P3DX | GIREG_RX.SCLK | IFS1P3DX |
| GDDRX1_RX.ECLK.Aligned | IDDRXD DQSBUFG TRDLLB DLLDELB CLKDIVB | GDDRX1_RX.SCLK.Aligned | IDDRX1D TRDLLB DLLDELB CLKDIVB |
| GDDRX1_RX.ECLK.Centered | IDDRXD DQSBUFG | GDDRX1_RX.SCLK.Centered | IDDRX1D |

| "E" Interface | Primitives Required | Equivalent "EA" Interface | Primitives Used |
|---|---|---|---|
| GDDRX1_RX.DQS.Aligned | IDDRXD DQSBUFF DQSDLL | GDDRX1_RX.DQS.Aligned | IDDRXD DQSBUFF DQSDLL |
| GDDRX1_RX.DQS.Centered | IDDRXD DQSBUFF DQSDLL | GDDRX1_RX.DQS.Centered | IDDRXD DQSBUFF DQSDLL |
| GDDRX2_RX.ECLK.Aligned | IDDRX2D DQSBUFE TRDLLB DLLDELB CLKDIV | GDDRX2_RX.ECLK.Aligned | IDDRX2D1 TRDLLB DLLDELB CLKDIVB |
| GDDRX2_RX.ECLK.Centered | IDDRX2D DQSBUFE CLKDIVB | GDDRX2_RX.ECLK.Centered | IDDRX2D1 CLKDIVB |
| GDDRX2_RX.DQS.Aligned | IDDRX2D DQSBUFD DQSDLL PLL | GDDRX2_RX.DQS.Aligned | IDDRX2D DQSBUFD DQSDLL PLL |
| GDDRX2_RX.DQS.Centered | IDDRX2D DQSBUFD DQSDLL CLKDIVB | GDDRX2_RX.DQS.Centered | IDDRX2D DQSBUFD DQSDLL CLKDIVB |
| GOREG_TX.SCLK | OFS1P3DX (data)<br>ODDRXD (clock)<br>DQSBUFG (left/right sides only) | GOREG_TX.SCLK | OFS1P3DX (data) ODDRXD1 |
| GDDRX1_TX.SCLK.Centered | ODDRXD<br>DQSBUFG (left/right sides only)<br>PLL | GDDRX1_TX.SCLK.Centered | ODDRXD1 PLL |
| GDDRX1_TX.SCLK.Aligned | ODDRXD DQSBUFG | GDDRX1_TX.SCLK.Aligned | ODDRX1D |
| GDDRX1_TX.DQS.Centered | ODDRXD (data) ODDRDQSA (clock) DQSBUFF DQSDLL | GDDRX1_TX.DQS.Centered | ODDRXD (data) ODDRDQSA (clock) DQSBUFF DQSDLL |

Due to the differences described above, some of the "E" interfaces must be regenerated in the software as an "EA" device. Table 19.3 can be used as a guide to determine which interfaces need to be regenerated. When necessary, interfaces should be regenerated using the IPexpress software tool.

**Table 19.3. "E" to "EA" Design Conversion Table**

| "E" Interface | Equivalent "EA" Interface | Regeneration Required[1] |
|---|---|---|
| GIREG_RX.SCLK | GIREG_RX.SCLK | No |
| GDDRX1_RX.ECLK.Aligned | GDDRX1_RX.SCLK.Aligned | Yes |
| GDDRX1_RX.ECLK.Centered | GDDRX1_RX.SCLK.Centered | Yes |
| GDDRX1_RX.DQS.Aligned | GDDRX1_RX.DQS.Aligned | No |
| GDDRX1_RX.DQS.Centered | GDDRX1_RX.DQS.Centered | No |
| GDDRX2_RX.ECLK.Aligned | GDDRX2_RX.ECLK.Aligned | Yes |
| GDDRX2_RX.ECLK.Centered | GDDRX2_RX.ECLK.Centered | Yes |
| GDDRX2_RX.DQS.Aligned | GDDRX2_RX.DQS.Aligned | No |
| GDDRX2_RX.DQS.Centered | GDDRX2_RX.DQS.Centered | No |
| GOREG_TX.SCLK | GOREG_TX.SCLK | Yes |
| GDDRX1_TX.SCLK.Centered | GDDRX1_TX.SCLK.Centered | Yes |
| GDDRX1_TX.SCLK.Aligned | GDDRX1_TX.SCLK.Aligned | Yes |
| GDDRX1_TX.DQS.Centered | GDDRX1_TX.DQS.Centered | No |

**Note:** All designs should be regenerated using IPexpress.

# 20. Migrating Designs from ispLEVER 7.2 SP2 to ispLEVER 8.0

This section lists changes that need to be made to existing designs when migrating from ispLEVER 7.2 SP2 to ispLEVER 8.0 and later versions of the software. The same changes apply if moving from ispLEVER 7.2 to the Lattice Diamond design software as well.

## 20.1. LatticeECP3-70E and LatticeECP3-90E designs:

- No HDL changes are required to move the LatticeECP3 "E" generic designs from ispLEVER 7.2 SP2 to isp- LEVER 8.0
- It is required to re-run the designs through the software Map, Place and Route. ispLEVER 8.0 has an added Design Rule Check that flags any general non-dedicated routed used on clock paths to DDR interfaces.
- If any of these errors occur, they must be fixed by either changing the clock pin assignment to a dedicated pin or by adding preferences to route the clock on dedicated clock routes.
- All the interface rules and placement guidelines specified for the interface must be followed.

## 20.2. LatticeECP3-150EA designs:

- All "EA" designs were implemented as "E" designs in ispLEVER 7.2 SP2. All "EA" generic DDR designs has to be regenerated when moving over to ispLEVER 8.0 except for the designs using DQS interfaces.
- The requirements listed in the section Migrating Designs from LatticeECP3 "E" to "EA" should be followed when moving "EA" designs from ispLEVER 7.2 SP2 to ispLEVER 8.0.
- All modules should be regenerated using IPexpress.
- "EA" designs require that an Interface ID attribute, IDDRAPPS or ODDRAPPS, be added to all the IDDR and ODDR elements to indicate the interface topology being used. The software errors out if it does not see this attribute on the DDR elements. These attributes are added automatically when IPexpress is used to generate the interface.

## 20.3. Other important migration rules:

- All DDR designs must only use one of the pre-defined topologies listed in this technical note.
- All DDR interfaces must be generated using the IPexpress software tool.
- This technical note must be strictly followed to understand the interface rules, placement guidelines and timing analysis for all interfaces.
- ispLEVER 7.2 SP2 allows the use of either the DELAYB or DELAYC element to delay the incoming data. In ispLEVER 8.0, all non-dynamic interfaces must only use the DELAYC element to delay the data input. DELAYB can only be used only for dynamic interfaces.
- A new Interface ID attribute IDDRAPPS and ODDRAPPS is required for "EA" devices. The software uses this attribute to determine the delay settings to be programmed into the DELAYC element. Refer to the DDR Software Primitives and Attributes section for the values to be used for this attribute. In ispLEVER 8.0, the software errors out if this attribute is not assigned for "EA" devices. "E" devices do not require this attribute.
- It is required that clocks connected to the DDR registers only use dedicated clock resources. No general routing should be used on these clocks. A new DRC check was added to ispLEVER 8.0 Place and Route that generates an error message if the clocks to any of the DDR elements are not using a dedicated clock route. This check also generates an error message if the clock going to the PLL/DLL that is generating the DDR clocks is not using a dedicated route. As a result, you may see designs passing all DRC checks in ispLEVER 7.2 SP2 fail in ispLEVER 8.0. It is recommended that this problem be fixed by changing either clock location or adding preferences to reroute the clock using a dedicated clock route.

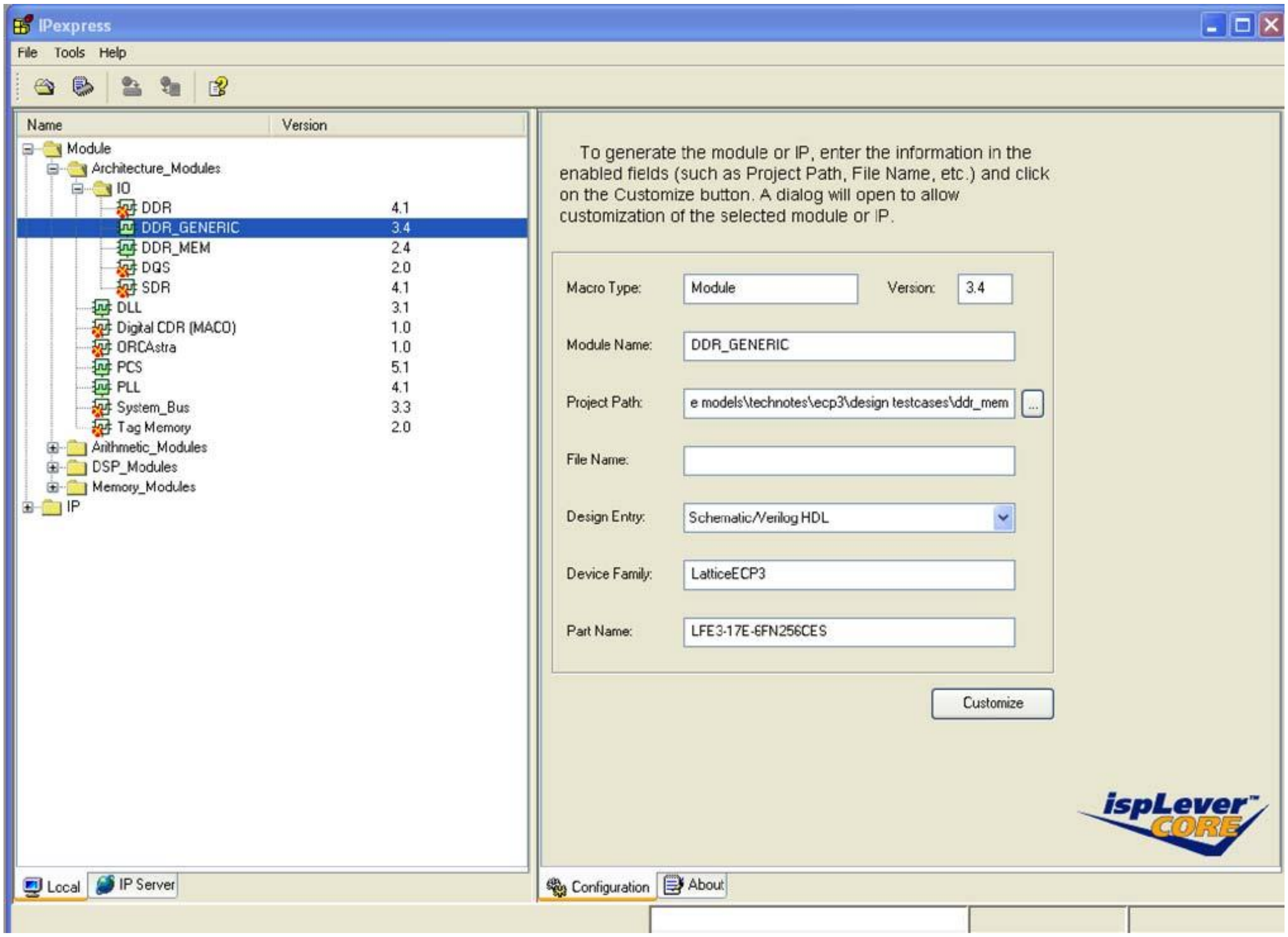
# Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2

This appendix describes the interface generation for DDR Generic and DDR Memory using IPexpress in ispLEVER 7.2 SP2. *It is highly recommended to update your software to the latest version*. This appendix is only for reference.

IPexpress can be used to configure and generate the DDR Memory Interface and Generic DDR Module. The tool generates an HDL module that contains the DDR primitives. This module can be used in the top-level design.

To implement the correct clocking structures to be used for high-speed source synchronous interfaces, see the High-Speed DDR Interface Details section.

Figure A.1 shows the main window of IPexpress. The DDR_Generic and DDR_MEM options are under Architecture.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.1 |
| DDR_GENERIC | 3.4 |
| DDR_MEM | 2.4 |
| DQS | 2.0 |
| SDR | 4.1 |
| DLL | 3.1 |
| Digital CDR (MACO) | 1.0 |
| ORCAstra | 1.0 |
| PCS | 5.1 |
| PLL | 4.1 |
| System_Bus | 3.3 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| DSP_Modules | |
| Memory_Modules | |
| IP | |

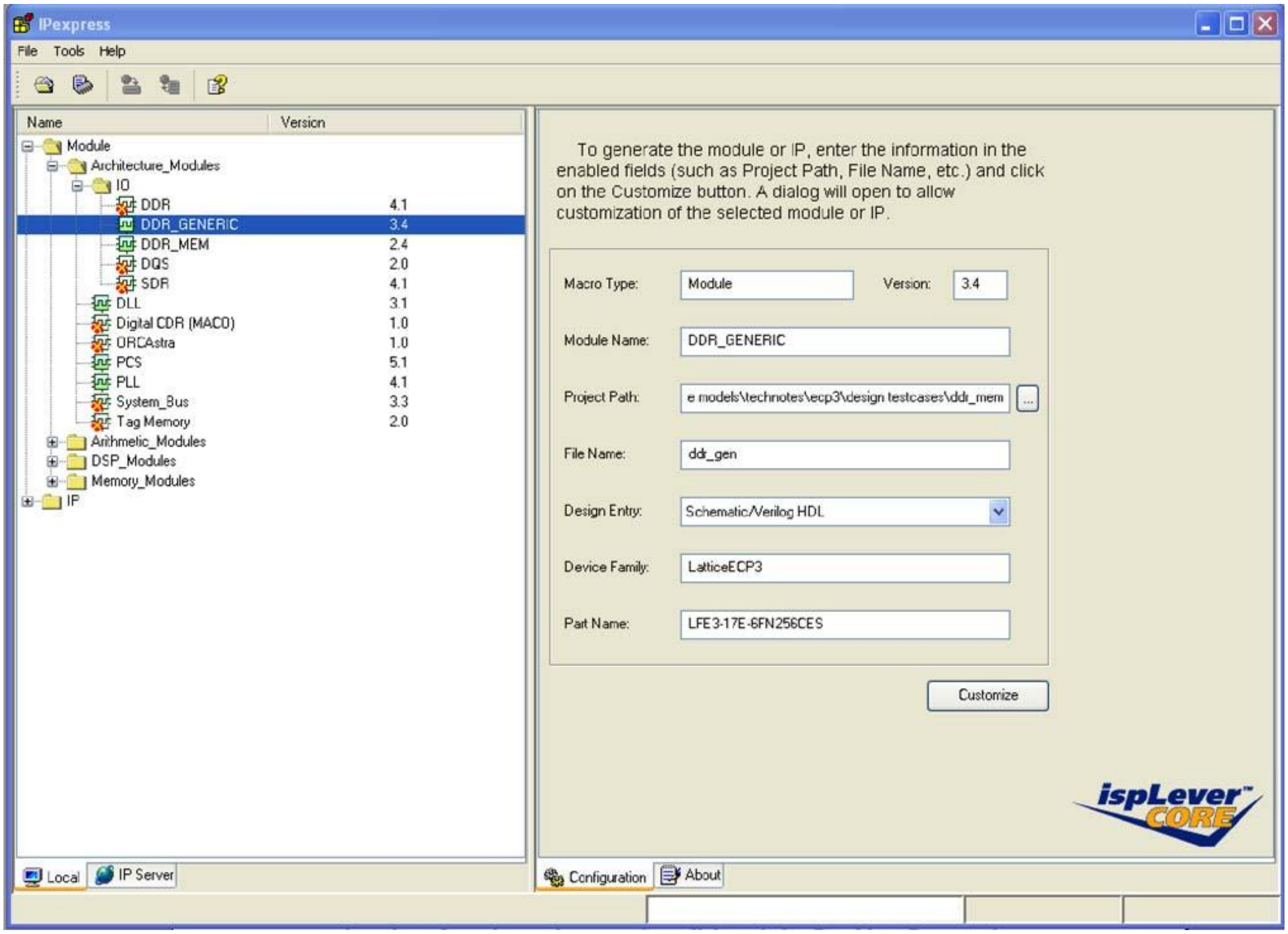
To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: Module
Version: 3.4
Module Name: DDR_GENERIC
Project Path: e models\technotes\ecp3\design testcases\ddr_mem
File Name:
Design Entry: Schematic/Verilog HDL
Device Family: LatticeECP3
Part Name: LFE3-17E-6FN256CES

**Figure A.1 IPexpress Main Window**

## A.1. DDR Generic

Figure A.1 shows the main window when DDR_GENERIC is selected. The only entry required in this window is the module name. Other entries are set to the project settings. You may change these entries if desired. After entering the module name, click on Customize to open the Configuration Tab window as shown in Figure A.2.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.1 |
| DDR_GENERIC | 3.4 |
| DDR_MEM | 2.4 |
| DQS | 2.0 |
| SDR | 4.1 |
| DLL | 3.1 |
| Digital CDR (MACO) | 1.0 |
| ORCAstra | 1.0 |
| PCS | 5.1 |
| PLL | 4.1 |
| System_Bus | 3.3 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| DSP_Modules | |
| Memory_Modules | |
| IP | |

To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: Module Version: 3.4

Module Name: DDR_GENERIC

Project Path: e models\technotes\ecp3\design testcases\ddr_mem

File Name: ddr_gen

Design Entry: Schematic/Verilog HDL

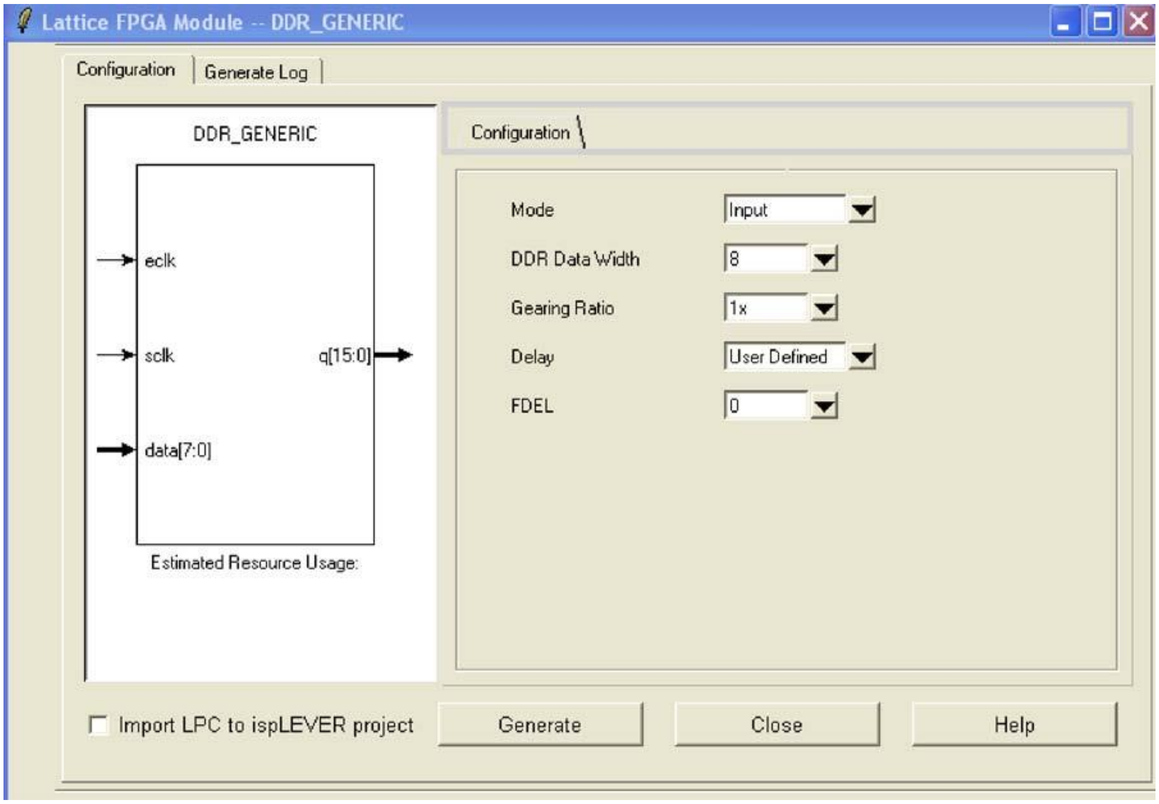
Device Family: LatticeECP3

Part Name: LFE3-17E-6FN256CES

**Figure A.2 IPexpress Main Window for DDR_Generic**

## A.2. Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click Generate to generate source and constraint files. You may choose to use the .lpc file to load parameters.


Lattice FPGA Module -- DDR_GENERIC
Configuration
Generate Log
DDR_GENERIC
eclk
sclk
q[15:0]
data[7:0]
Estimated Resource Usage:
Configuration
□ Import LPC to ispLEVER project
Generate
Close
Help

Mode: Input
DDR Data Width: 8
Gearing Ratio: 1x
Delay: User Defined
FDEL: 0

Figure A.3 Configuration Tab for DDR_Generic

You can change the Mode parameter to choose either Input or Output Tri-state DDR module.

The other configuration parameters change according to the mode selected. The delay and parameter are only available for Input modes.

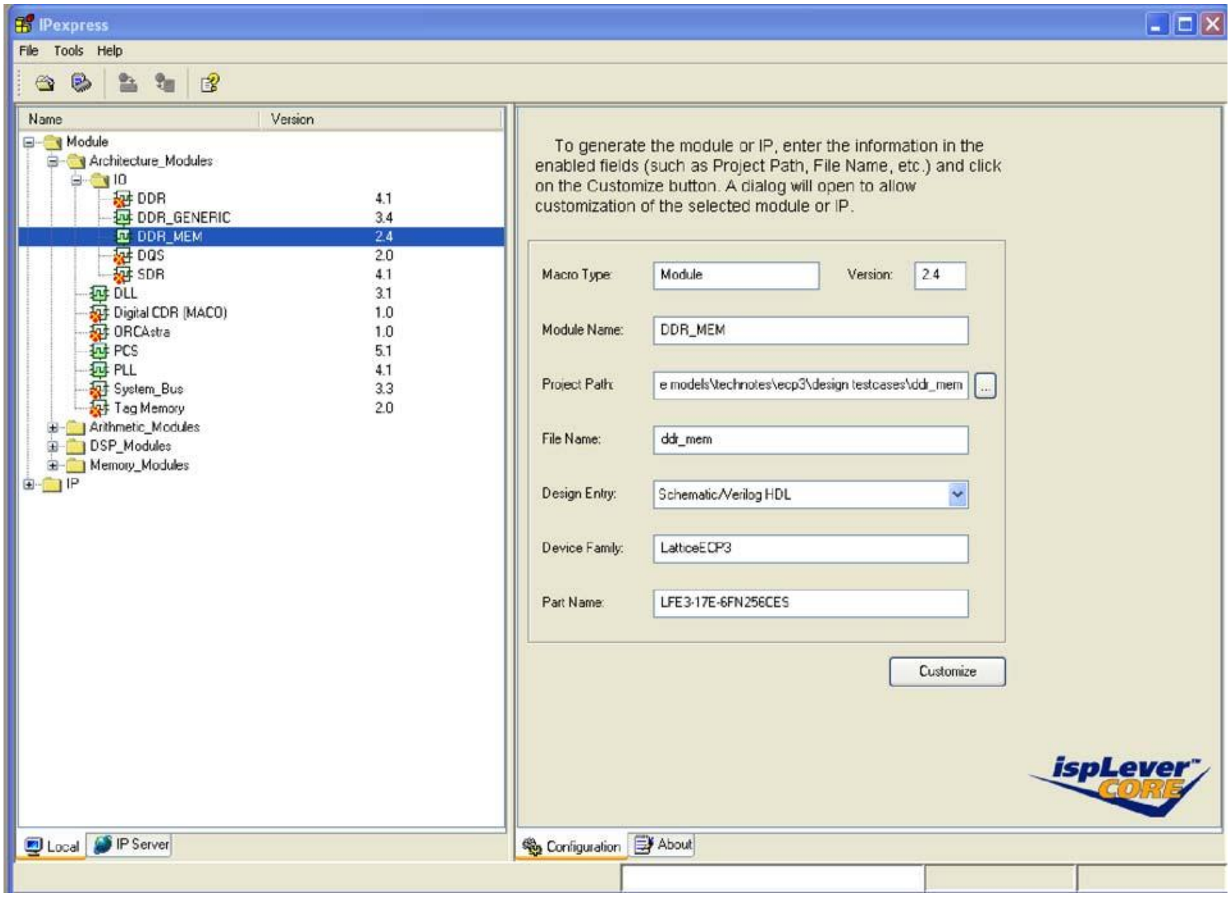
Table A.1 describes all user parameters in the IPexpress user interface and their usage.

**Table A.1. "E" to "EA" Design Conversion Table**

| User Parameter | Description | Values/Range | Default |
|---|---|---|---|
| Mode | Mode selection for the DDR block | Input, Output, Tri-state | Input |
| Data Width | Width of the data bus | 1-64 | 8 |
| Gearing Ratio | Gearing ratio selection | 1x, 2x | 1x |
| Delay | Input delay configuration | Dynamic, User-Defined, Fixed | User-Defined |
| FDEL | User-Defined delay values. Available only when Delay is configured to User-Defined. | 0-15 | 0 |

## A.3. DDR_MEM

Figure A.4 shows the main window when DDR_MEM is selected. Similar to the DDR_Generic, the only entry required here is the module name. Other entries are set to the project settings. You may change these entries if desired. After entering the module name, click on Customize to open the Configuration Tab window as shown in Figure A.4. Although you can generate the DDR3_MEM using IPexpress in ispLEVER 7.2 SP2, it is recommended that the LatticeECP3 DDR3 Memory Controller IP Core be referred to prior to building any DDR3 memory controllers with LatticeECP3 devices.



To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: Module Version: 2.4

Module Name: DDR_MEM

Project Path: e models\technotes\ecp3\design testcases\ddr_mem

File Name: ddr_mem

Design Entry: Schematic/Verilog HDL

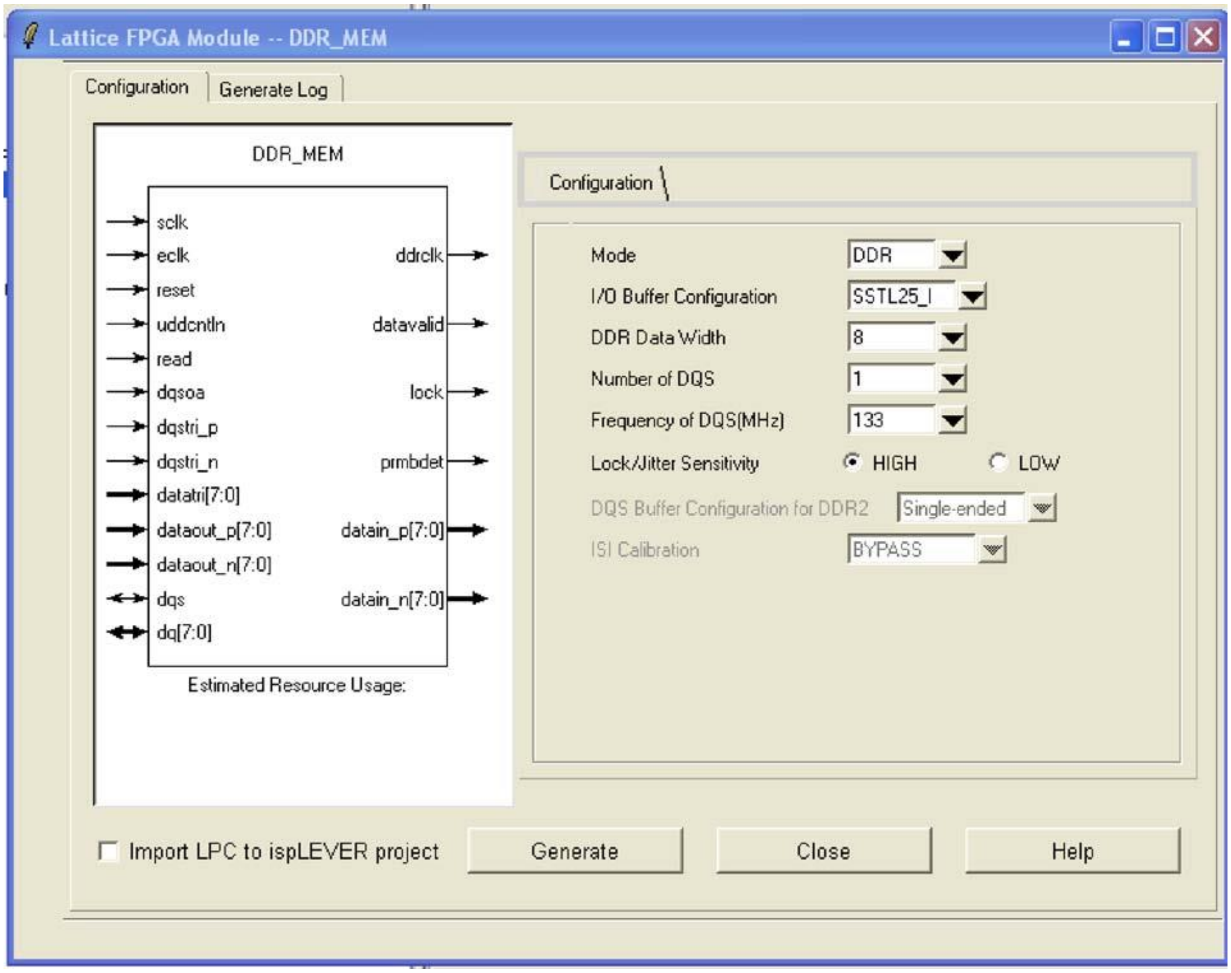
Device Family: LatticeECP3

Part Name: LFE3-17E-6FN256CES

**Figure A.4 IPexpress Main Window for DDR_MEM**

## A.4. Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click Generate to generate source and constraint files. You may choose to use the .lpc file to load parameters.



Mode: DDR

I/O Buffer Configuration: SSTL25_I

DDR Data Width: 8

Number of DQS: 1

Frequency of DQS(MHz): 133

Lock/Jitter Sensitivity: ● HIGH ○ LOW

DQS Buffer Configuration for DDR2: Single-ended

ISI Calibration: BYPASS

□ Import LPC to ispLEVER project

**Figure A.5 Configuration Tab for DDR_MEM**

You can change the Mode parameter to the DDR, DDR2 or DDR3 interface. The other configuration parameters change according to the mode selected. The Number of DQS parameter determines the number of DDR Interfaces. The software assumes there are eight data bits for every DQS. You can also choose the frequency of operation; the DDRDLL is configured to this frequency.

It is recommended that the Lock/Jitter be enabled if the DDR interface is running at 133 MHz or higher. ISI Calibration is only allowed for DDR3 configuration. The parameters available depend on the mode selected. Table A.2, Table A.3 and Table A.4 describe all user parameters in the IPexpress user interface and their usage for modes DDR, DDR2 and DDR3.

**Table A.2. User Parameters in the IPexpress User Interface when in DDR Mode**

| User Parameter | Description | Values/Range | Default |
|---|---|---|---|
| I/O Buffer Configuration | I/O standard used for the interface. This also depends on the mode selected. | SSTL25_I, SSTL25_II | SSTL25_I |
| Data Width | Width of the data bus. | 8-64 | 8 |
| Number of DQS | Number of DQS determines the number of DQS groups | 1, 2, 4, 8 | 1 |
| Frequency of DQS | DDR Interface Frequency. This is also input to the DDR DLL. The values depend on the mode selected. | 100 MHz, 133 MHz,<br>166 MHz, 200 MHz | 200 MHz |
| Lock/Jitter Sensitivity | DLL sensitivity to jitter. | High, Low | High |

### Table A.3. User Parameters in the IPexpress User Interface when in DDR2 Mode

| User Parameter | Description | Values/Range | Default |
|---|---|---|---|
| I/O Buffer Configuration | I/O standard used for the interface. This also depends on the mode selected. | SSTL18_I, SSTL18_II | SSTL18_I |
| Data Width | Width of the data bus. | 8-64 | 8 |
| Number of DQS | Number of DQS determines the number of DQS groups | 1, 2, 4, 8 | 1 |
| Frequency of DQS | DDR Interface Frequency. This is also input to the DDR DLL. The values depend on the mode selected. | 166 MHz, 200 MHz, 266 MHz | 200 MHz |
| Lock/Jitter Sensitivity | DLL sensitivity to jitter. | High, Low | High |
| DQS Buffer Configuration for DDR2 | DQS buffer can be optionally configured as Differential for DDR2. | On/Off | Off |

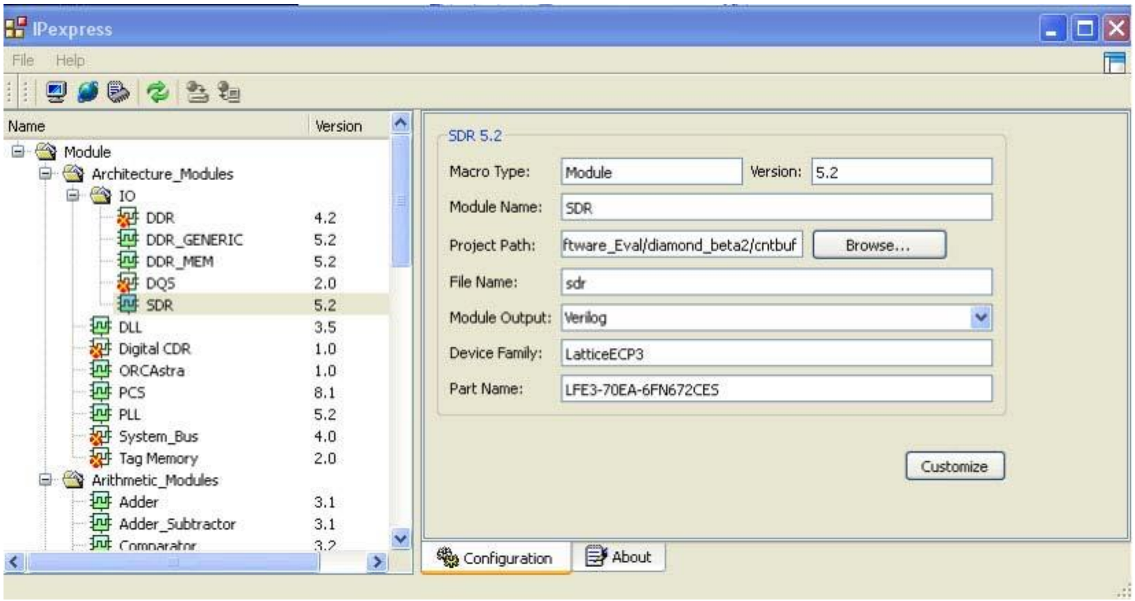
### Table A.4. User Parameters in the IPexpress User Interface when in DDR3 Mode

| User Parameter | Description | Values/Range | Default |
|---|---|---|---|
| I/O Buffer Configuration | I/O standard used for the interface. This also depends on the mode selected. | SSTL15_I, SSTL15_II | SSTL15_I |
| Data Width | Width of the data bus. | 8-64 | 8 |
| Number of DQS | Number of DQS determines the number of DQS groups | 1, 2, 4, 8 | 1 |
| Frequency of DQS | DDR Interface Frequency. This is also input to the DDR DLL. The values depend on the mode selected. | 400 MHz | 400 MHz |
| Lock/Jitter Sensitivity | DLL sensitivity to jitter. | High, Low | High |
| ISI Calibration | ISI calibration setting for DDR3 output. | BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7 | BYPASS |

# Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond

This appendix describes the interface generation for DDR Generic and DDR Memory using IPexpress in Lattice Diamond design software. The IPexpress tool is used to configure and generate all the high-speed interfaces described in this document. IPexpress generates a complete HDL module including clocking requirements for each of the interfaces.

For a detailed block diagram of each interface generated by IPexpress, see the section High-Speed DDR Interface Details. IPexpress can be opened from the Tools menu in Project Navigator. All DDR modules are located under Architecture Modules > IO. This section covers SDR and DDR_GENERIC. DDR_MEM as discussed in the Implementing DDR/DDR2/DDR3 Memory Interfaces section.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.2 |
| DDR_GENERIC | 5.2 |
| DDR_MEM | 5.2 |
| DQS | 2.0 |
| SDR | 5.2 |
| DLL | 3.5 |
| Digital CDR | 1.0 |
| ORCAstra | 1.0 |
| PCS | 8.1 |
| PLL | 5.2 |
| System_Bus | 4.0 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| Adder | 3.1 |
| Adder_Subtractor | 3.1 |
| Comparator | 3.2 |

SDR 5.2

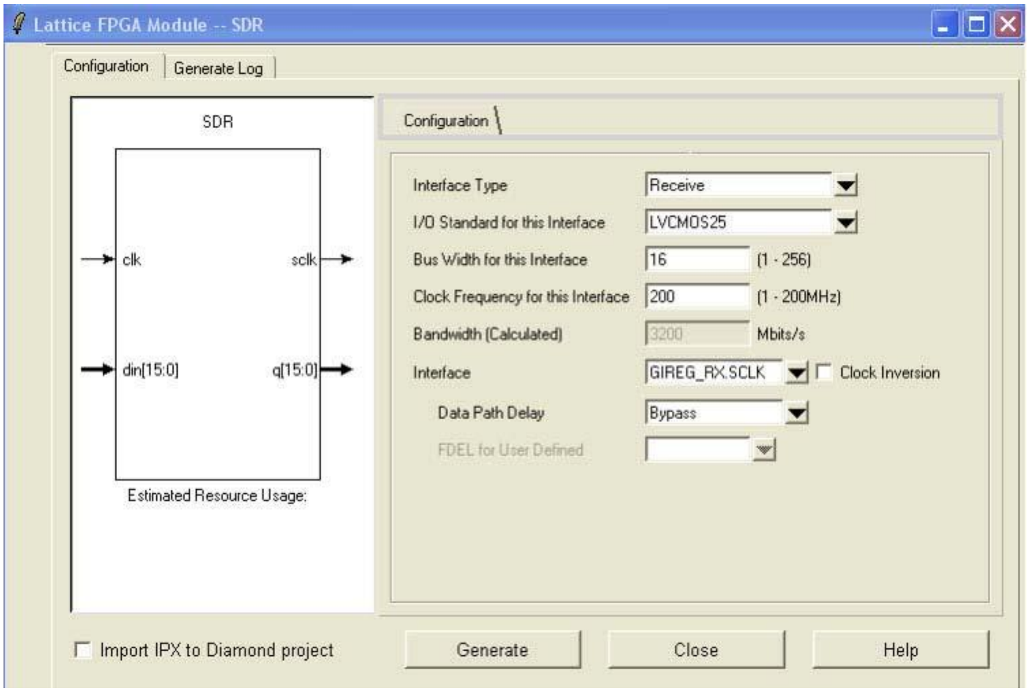
Macro Type: Module
Version: 5.2
Module Name: SDR
Project Path: ftware_Eval/diamond_beta2/cntbuf
File Name: sdr
Module Output: Verilog
Device Family: LatticeECP3
Part Name: LFE3-70EA-6FN672CES

**Figure B.6 IPexpress Main Window**

Select the type of interface you would like to build and enter the name of the module. Figure B.6 shows the type of interface selected as "SDR" and the module name entered. Each module can then be configured by clicking the Customize button.

## B.1. Building SDR Modules

Choose the interface type SDR, enter the module name and click the Customize to open the configuration tab Figure B.7 shows the Configuration Tab for the SDR module in IPexpress. Table B.5 lists the various configurations options available for SDR modules.



Lattice FPGA Module -- SDR
Configuration
Generate Log
SDR
clk
sclk
din[15:0]
q[15:0]
Estimated Resource Usage:
Configuration
□ Import IPX to Diamond project
Generate
Close
Help

Interface Type: Receive

I/O Standard for this Interface: LVCMOS25

Bus Width for this Interface: 16 (1 - 256)

Clock Frequency for this Interface: 200 (1 - 200MHz)

Bandwidth (Calculated): 3200 Mbits/s

Interface: GIREG_RX.SCLK □ Clock Inversion

Data Path Delay: Bypass

FDEL for User Defined:

**Figure B.7 SDR Configuration Tab**

**Table B.5. SDR Configuration Parameters**

| User Interface Option | Description | Values | Default |
|---|---|---|---|
| Interface Type | Type of interface (transmit or receive) | Transmit, Receive | Receive |
| I/O Standard for this Interface | I/O standard to be used for the interface | Transmit and Receive:<br>LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTL33<br>Transmit only:<br>RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE | LVCMOS25 |
| Bus Width for this Interface | Bus size for the interface | 1 - 256 | 16 |
| Clock Frequency for this Inter- face | Speed at which the interface runs | 1 - 200 | 200 |
| Bandwidth (Calculated) | Calculated from the clock frequency entered | (Calculated) | (Calculated) |
| Interface | Interface selected based on previous entries | Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default) | GIREG_RX.SCLK |
| Clock Inversion | Option to invert the clock input to the I/O register | DISABLED, ENABLED | DISABLED |
| Data Path Delay | Data input can be optionally delayed using the DELAY block | Bypass, Dynamic[1], User-Defined | Bypass |

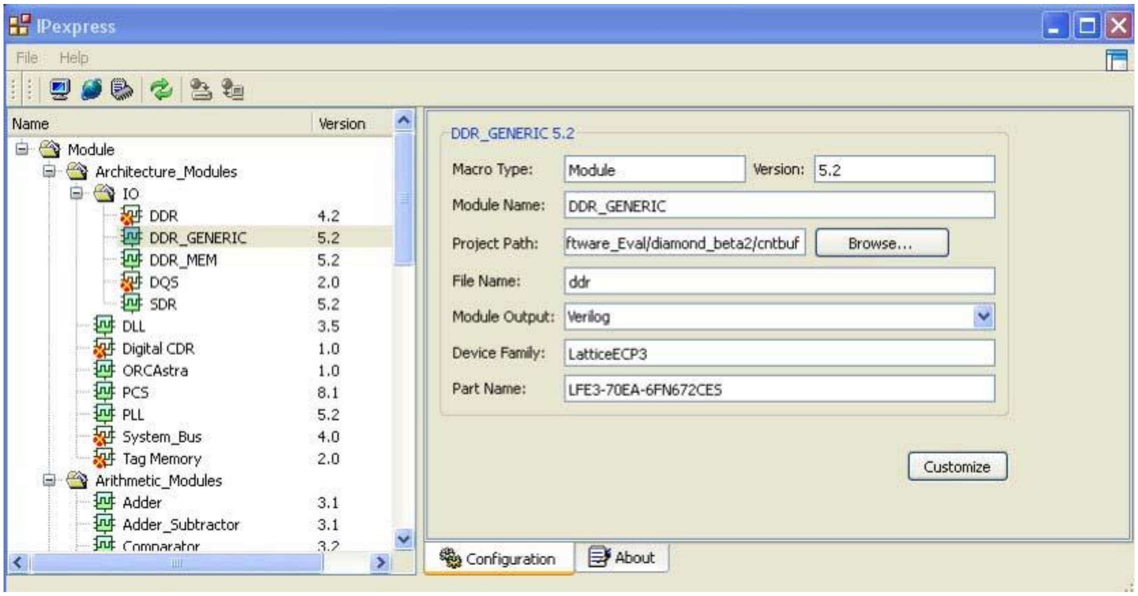
| User Interface Option | Description | Values | Default |
|---|---|---|---|
| FDEL for User-Defined | If Delay type selected above is user-defined, delay values can be entered with this parameter. | 0 to 15[2] | 0 |

**Notes:**

1. When Delay type Dynamic is selected, the 16-step delay values must be controlled from your design.
2. A FDEL is a fine-delay value that is additive. The delay value for a FDEL can be found in the LatticeECP3 Family Data Sheet.

## B.2. Building DDR Generic Modules

Choose interface type DDR_GENERIC, enter the module name and click Customize to open the configuration tab.



| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.2 |
| DDR_GENERIC | 5.2 |
| DDR_MEM | 5.2 |
| DQS | 2.0 |
| SDR | 5.2 |
| DLL | 3.5 |
| Digital CDR | 1.0 |
| ORCAstra | 1.0 |
| PCS | 8.1 |
| PLL | 5.2 |
| System_Bus | 4.0 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| Adder | 3.1 |
| Adder_Subtractor | 3.1 |
| Comparator | 3.2 |

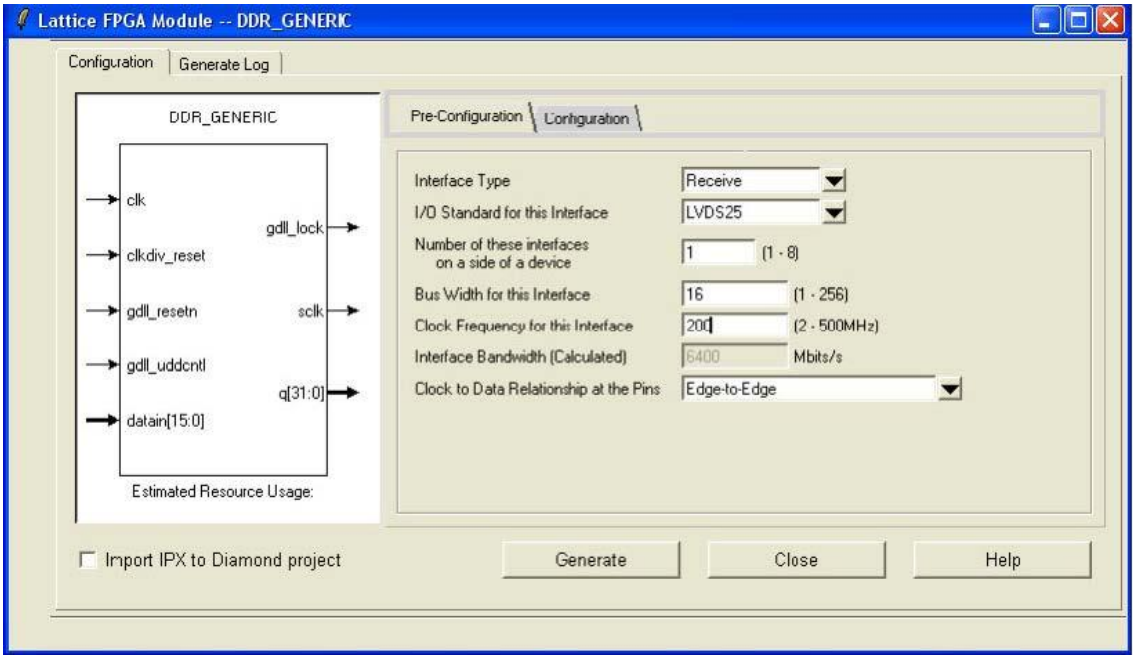
DDR_GENERIC 5.2

Macro Type: Module  Version: 5.2
Module Name: DDR_GENERIC
Project Path: ftware_Eval/diamond_beta2/cntbuf
File Name: ddr
Module Output: Verilog
Device Family: LatticeECP3
Part Name: LFE3-70EA-6FN672CES

**Figure B.8 “DDR_Generic” Selected in Main IPexpress Window**

When you click Customize, DDR modules have a Pre-Configuration tab and a Configuration tab. The Pre-Configuration tab allows you to enter information about the type of interface to be built. Based on the entries in the Pre-configuration tab, the Configuration tab is populated with the best interface selection. You can also, if necessary, override the selection made for the interface in the Configuration tab and customize the interface based on design requirements.

Figure B.9 shows the Pre-Configuration tab for DDR generic interfaces. Table B.6 lists the various parameters in the tab.



Interface Type: Receive

I/O Standard for this Interface: LVDS25

Number of these interfaces on a side of a device: 1 (1 - 8)

Bus Width for this Interface: 16 (1 - 256)

Clock Frequency for this Interface: 200 (2 - 500MHz)

Interface Bandwidth (Calculated): 6400 Mbits/s

Clock to Data Relationship at the Pins: Edge-to-Edge

□ Import IPX to Diamond project

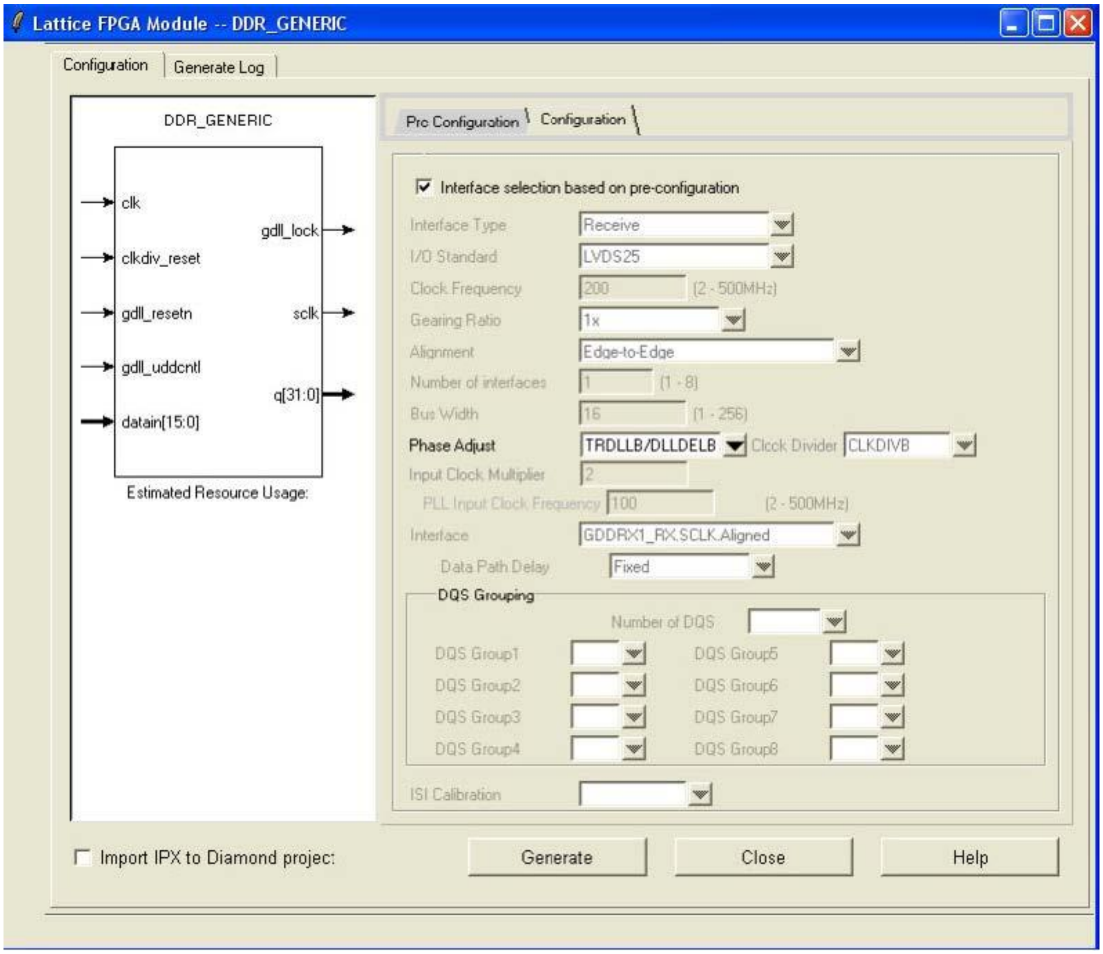
**Figure B.9 DDR Generic Pre-Configuration Tab**

**Table B.6. Pre-Configuration Tab Settings**

| User Interface Option | Description | Values |
|---|---|---|
| Interface Type (Transmit or Receive) | Type of interface (Receive or Transmit) | Transmit, Receive |
| I/O Standard for this Interface | I/O Standard used for the interface | Transmit and Receive: LVCMOS25,LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTL33<br>Transmit only:<br>RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE |
| Number of interfaces on a side of a device | Number of interfaces to be implemented per side. This is used primarily for narrow bus width interfaces (<10). Other- wise it is recommended to leave this at 1. | 1 to 8 |
| Bus Width for this Interface | Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If number of interfaces per side is >1 and if using differential I/O standards then bus width is limited to 5. | 1-256 |
| Clock Frequency for this Interface | Interface speed | 2 - 500 MHz |
| Interface Bandwidth (Calculated) | Bandwidth is calculated from the clock frequency. | Calculated |
| Clock to Data Relationship at the Pins | Relationship between clock and data. | Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required*, Dynamic Clock Phase Alignment Required |

*Note: Dynamic Phase Alignment is only available for x2 interfaces (i.e, when the clock frequency is higher than 200 MHz).

Based on the selections made in the Pre-Configuration Tab, the Configuration Tab is populated with the selections. Figure B.10 shows the Configuration Tab for the selections made in the Pre-Configuration Tab.



☑ Interface selection based on pre-configuration

Interface Type: Receive
I/O Standard: LVDS25
Clock Frequency: 200 (2 - 500MHz)
Gearing Ratio: 1x
Alignment: Edge-to-Edge
Number of interfaces: 1 (1 - 8)
Bus Width: 16 (1 - 256)
Phase Adjust: TRDLLB/DLLDELB Clock Divider: CLKDIVB
Input Clock Multiplier: 2
PLL Input Clock Frequency: 100 (2 - 500MHz)
Interface: GDDRX1_RX.SCLK.Aligned
Data Path Delay: Fixed

DQS Grouping
Number of DQS:
DQS Group1: DQS Group5:
DQS Group2: DQS Group6:
DQS Group3: DQS Group7:
DQS Group4: DQS Group8:

ISI Calibration:

☐ Import IPX to Diamond projec:

**Figure B.10 DDR Generic Configuration Tab**

The checkbox at the top of this tab indicates that the interface is selected based on entries in the Pre-Configuration tab. You can choose to change these values by disabling this entry. Note that IPexpress chooses the most suitable interface based on selections made in the Pre-Configuration tab.

Table B.7 lists the various parameters in the Configuration tab.

### Table B.7. Configuration Tab Settings

| User Interface Option | Description | Values | Default Value |
|---|---|---|---|
| Interface Selection Based on Pre-configuration | Indicates interface is selected based on selection made in the Pre-configuration tab. Disabling this checkbox allows you to make changes if needed. | ENABLED, DISABLED | ENABLED |
| Interface Type | Type of interface (receive or transmit) | Transmit, Receive | Receive |
| I/O Standard | I/O standard used for the interface | All the ones listed in the Pre-configuration tab | LVCMOS25 |
| Clock Frequency | Speed of the interface | 2 to 500 MHz | 200 MHz |
| Gearing Ratio | DDR register gearing ratio (1x or 2x) | 1x, 2x | 1x |
| Alignment | Clock to data alignment | Edge-to-Edge,<br>Centered,<br>Dynamic Data Phase Alignment Required,<br>Dynamic Clock Phase Alignment Required | Edge-to-Edge |
| Number of Interfaces | Number of interfaces to be implemented per side. This is primarily used for narrow bus width interfaces (<10), otherwise it is recommended to leave this at 1. | 1 to 8 | 1 |
| Bus Width | Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If the number of inter- faces per side is >1 and if using differential I/O standards then the bus width is limited to 5. | 1 to 256 | 10 |
| Phase Adjust | Module used for phase shifting input clock. | TRDLLB/DLLDELB, PLL[1] | TRDLLB/DLLDELB |
| Clock Divider | Module used for generation of SCLK from ECLK. | CLKDIVB, TRDLLB[2] | CLKDIVB |
| Interface | Shows list of all valid high-speed interfaces for a given configuration. | See Table 5.4 for interfaces available for a given configuration. | GDDRX1_RX.SCLK. Aligned (EAdevices);<br>GDDRX1_RX.ECLK. Aligned (E devices) |
| Data Path Delay | Data input can be optionally delayed using the DELAY block. Value is selected based on Inter-face Type. | Bypass, Fixed, Dynamic[3] | Fixed |
| Number of DQS Groups | Enabled when a DQS interface is selected in the Interface selection. | 1 to 8 | |
| Number of DQ:<br>DQS Group1 to DQS Group8 | This option can be used to change thenumber of DQ assigned to each DQS lane. Each DQS lane can support up to 10 DQ. | 1 to 10 | |

**Notes:**

1. Only available when using GDDRX2_RX.ECLK.Aligned interface.
2. Only available when using GDDRX2_RX.SCLK Aligned interface.
3. When Dynamic Delay is selected, the 16-step delay values must be controlled from your design.

Table B.8 shows how the interfaces are selected by IPexpress based on the selections made in the Pre-Configuration tab.

Table B.8. IPexpress Interface Selection

| Device Selected | Interface Type | Gearing Ratio[1] | Alignment | Number of Interfaces | Interface |
|---|---|---|---|---|---|
| EA | Receive | 1x | Edge-to-Edge | 1 | GDDRX1_RX.SCLK.Aligned |
| EA | Receive | 1x | Centered | 1 | GDDRX1_RX.SCLK.Centered |
| E | Receive | 1x | Edge-to-Edge | 1 | GDDRX1_RX.ECLK.Aligned |
| E | Receive | 1x | Centered | 1 | GDDRX1_RX.ECLK.Centered |
| E, EA | Receive | 1x | Edge-to-Edge | >1 | GDDRX1_RX.DQS.Aligned |
| E, EA | Receive | 1x | Centered | >1 | GDDRX1_RX.DQS.Centered |
| E, EA | Receive | 2x | Edge-to-Edge | 1 | GDDRX2_RX.ECLK.Aligned |
| E, EA | Receive | 2x | Centered | 1 | GDDRX2_RX.ECLK.Centered |
| E, EA | Receive | 2x | Edge-to-Edge | >1 | GDDRX2_RX.DQS.Aligned |
| E, EA | Receive | 2x | Centered | >1 | GDDRX2_RX.DQS.Centered |
| EA | Receive | 2x | Dynamic | 1 | GDDRX2_RX.ECLK.Dynamic (Default) |
| EA | | | | | GDDRX2_RX.DQS.Dynamic[2] |
| EA | | | | | GDDRX2_RX.PLL.Dynamic[2] |
| E, EA | Transmit | 1x | Centered | 1 | GDDRX1_TX.SCLK.Centered |
| E, EA | Transmit | 1x | Edge-to-Edge | 1 | GDDRX1_TX.SCLK.Aligned |
| E, EA | Transmit | 1x | Centered | >1 | GDDRX1_TX.DQS.Centered |
| EA | Transmit | 2x | Edge-to-Edge | 1 | GDDRX2_TX.Aligned |
| EA | Transmit | 2x | Centered | >1 | GDDRX2_TX.DQSDLL.Centered |
| EA | Transmit | 2x | Centered | <1 | GDDRX2_TX.PLL.Centered |

**Notes:**

1. Gearing Ratio of 1x is selected for clock frequencies less than 200 MHz. Gearing ratio of 2x is selected for frequencies above 200 MHz.
2. These interfaces can only be selected in the Configuration Tab.

The implementation for several of the interfaces described above differs between the “E” and “EA” devices. Refer to the High-Speed DDR Interface Details section to see implementation details for “E” and “EA” devices.

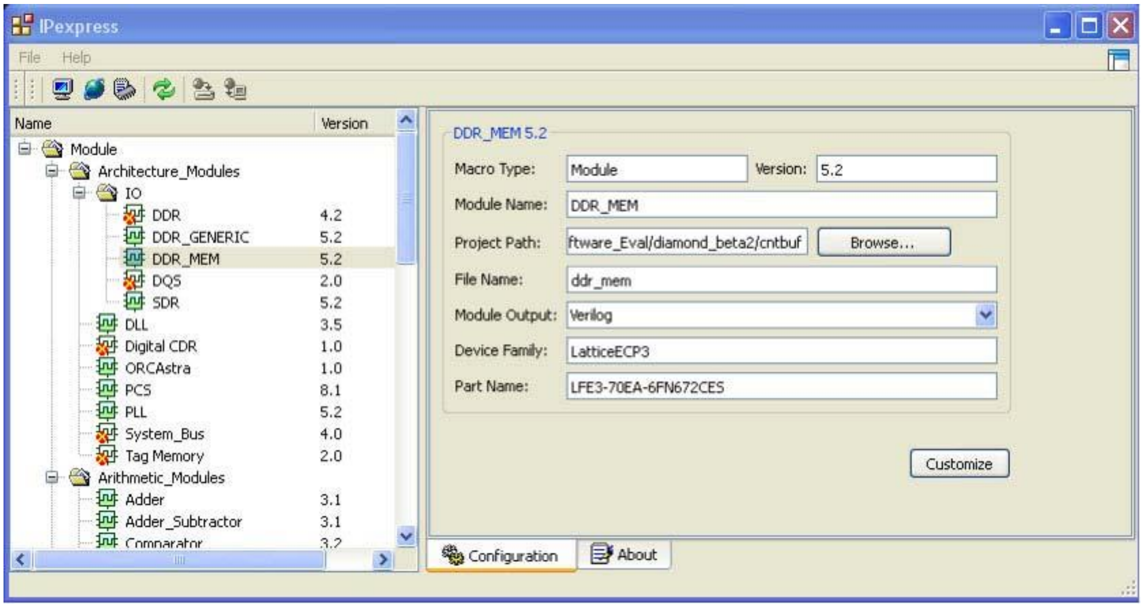
The Data Delay setting for each interface is predetermined and cannot be changed by the user. You can only control Data Delay values when using a dynamic interface.

**Note:** Some modules generated by IPexpress have a SCLK and ECLK output port. If present, this port must be used to drive logic outside the interface driven by the same signal. In these modules, the input buffer for the clock is inside the IPexpress module and therefore cannot be used to drive other logic in the top level.

## B.3. Building DDR Memory Interfaces

The IPexpress tool is used to configure and generate the DDR, DDR2 and DDR3 memory interfaces.

To see the detailed block diagram for each interface generated by IPexpress see the Memory Read Implementation and Memory Write Implementation sections. IPexpress can be opened from the Tools menu in Project Navigator. All the DDR modules are located under Architecture Modules > IO. DDR_MEM is used to generate DDR memory interfaces.



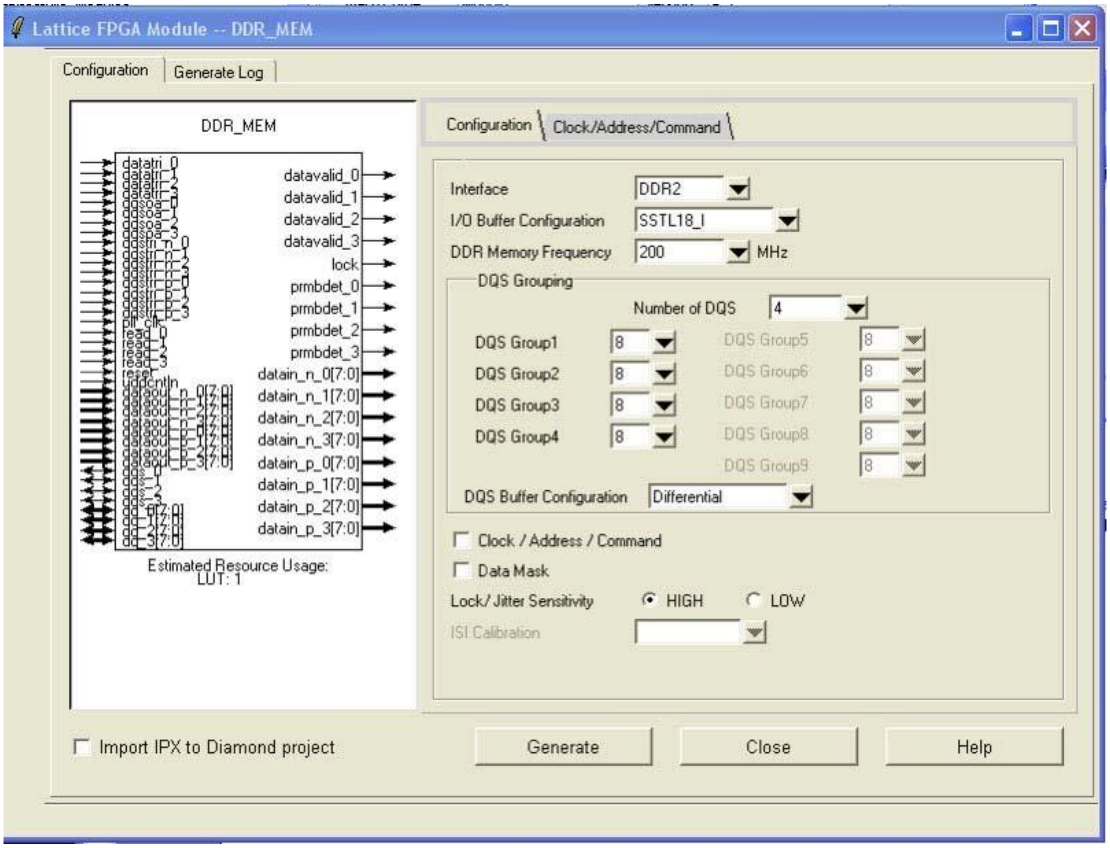
| Name | Version |
|---|---|
| Module | |
| Architecture_Modules | |
| IO | |
| DDR | 4.2 |
| DDR_GENERIC | 5.2 |
| DDR_MEM | 5.2 |
| DQS | 2.0 |
| SDR | 5.2 |
| DLL | 3.5 |
| Digital CDR | 1.0 |
| ORCAstra | 1.0 |
| PCS | 8.1 |
| PLL | 5.2 |
| System_Bus | 4.0 |
| Tag Memory | 2.0 |
| Arithmetic_Modules | |
| Adder | 3.1 |
| Adder_Subtractor | 3.1 |
| Comparator | 3.2 |

DDR_MEM 5.2

Macro Type: Module
Version: 5.2
Module Name: DDR_MEM
Project Path: ftware_Eval/diamond_beta2/cntbuf
File Name: ddr_mem
Module Output: Verilog
Device Family: LatticeECP3
Part Name: LFE3-70EA-6FN672CES

**Figure B.11 "DDR_MEM" Selected in Main IPexpress Window**

Figure B.11 shows the IPexpress Main Window. To generate a DDR memory interface, select **DDR_MEM**, assign a module name and click on Customize to see the Configuration tab. Figure B.12 shows the Configuration tab for the DDR_MEM interface. You can choose to implement the DDR1_MEM, DDR2_MEM or DDR3_MEM interface.



Interface: DDR2
I/O Buffer Configuration: SSTL18_I
DDR Memory Frequency: 200 MHz

DQS Grouping

Number of DQS: 4

| | | | |
|---|---|---|---|
| DQS Group1 | 8 | DQS Group5 | 8 |
| DQS Group2 | 8 | DQS Group6 | 8 |
| DQS Group3 | 8 | DQS Group7 | 8 |
| DQS Group4 | 8 | DQS Group8 | 8 |
| | | DQS Group9 | 8 |

DQS Buffer Configuration: Differential

☐ Clock / Address / Command
☐ Data Mask
Lock/ Jitter Sensitivity ● HIGH ○ LOW
ISI Calibration

☐ Import IPX to Diamond project

**Figure B.12 Configuration Tab for DDR_MEM**

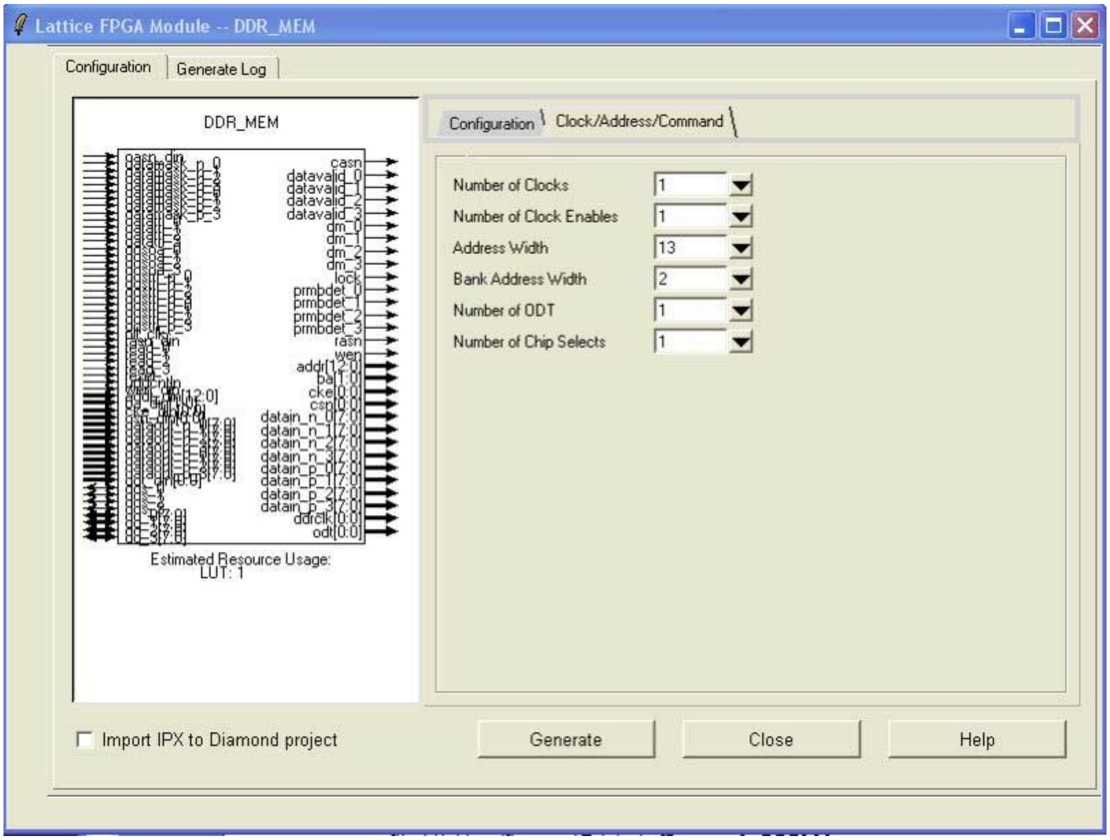
Table B.9 describes the various settings shown in the Configuration tab above.

**Table B.9. Configuration Tab Settings for DDR_MEM**

| User Interface Option | Description | Range | Default Value |
|---|---|---|---|
| Interface | DDR memory interface type | DDR, DDR2, DDR3 | DDR2 |
| I/O Buffer Configuration | I/O type configuration for DDR pins | SSTL25_I, SSTL25_II SSTL18_I, SSTL18_II, SSTL15 | DDR – SSTL25_I DDR2 – SSTL18_I DDR3 – SSTL15 |
| Number of DQS | Interface width (1 DQS per 8 bits of data) | 1 to 9 | 4 |
| DQS Group1 to DQS Group8 | Number of DQ per DQS pin | 1 to 8 | 8 |
| DQS Buffer Configuration for DDR2 | DQS buffer type | Single-ended, Differential | DDR – Single-ended<br>DDR2 – Single-ended<br>DDR3 – Differential |
| Clock/Address/Command | Clock/address/command interface is generated when this option is checked | ENABLED, DISABLED | DISABLED |
| Data Mask | Data mask signal is generated when this option is checked | ENABLED, DISABLED | DISABLED |
| Lock/Jitter Sensitivity | Lock Sensitivity attribute for DQSDLL* | HIGH, LOW | HIGH |
| DDR Memory Frequency | DDR Memory Interface Frequency | DDR – 87.5 MHz, 100 MHz, 133.33 MHz, 166.67 MHz, 200 MHz<br>DDR2 – 125 MHz, 200 MHz, 266.67 MHz<br>DDR3 – 150 MHz, 200 MHz, 300 MHz, 400 MHz | DDR – 200 MHz DDR2 – 200 MHz DDR3 – 400 MHz |
| ISI Calibration | ISI calibration is available for the DDR3 interface to adjust for inter-symbol inference adjustment per DQS group | BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7 | BYPASS |

*__Note:__ It is recommended to set Lock Sensitivity to HIGH for DDR Memory Frequency higher than 133 MHz.

If you choose to generate the Clock/Address/Command signals using IPexpress, then the settings in the Clock/Address/Command Tab are active and can be set up as required. Figure B.13 shows the Clock/Address/Command Tab in the IPexpress for DDR2 Memory.



Number of Clocks: 1
Number of Clock Enables: 1
Address Width: 13
Bank Address Width: 2
Number of ODT: 1
Number of Chip Selects: 1

☐ Import IPX to Diamond project

**Figure B.13 Clock/Address/Command Tab in the IPexpress for DDR_MEM**

Table B.10 lists the values that can be used for the Clock/Address/Command settings.

**Table B.10. Clock/Address/Command Settings for DDR_MEM**

| User Interface Option | Range | Default Value |
|---|---|---|
| Number of Clocks | 1, 2, 4 | 1 |
| Number of Clock Enables | 1, 2, 4 | 1 |
| Address Width | DDR: 12-14<br>DDR2: 13-16<br>DDR3: 13-16 | DDR: 13<br>DDR2: 13<br>DDR3: 14 |
| Bank Address Width | DDR: 2<br>DDR2: 2, 3<br>DDR3: 3 | DDR: 2<br>DDR2: 2<br>DDR3: 3 |
| Number of ODT | DDR: N/A DDR2: 1, 2, 4<br>DDR3: 1, 2, 4 | DDR: N/A DDR2: 1<br>DDR3: 1 |
| Number of Chip Selects | DDR: 1, 2, 4, 8<br>DDR2: 1, 2, 4<br>DDR3: 1, 2, 4 | DDR: 1<br>DDR2: 1<br>DDR3: 1 |

# 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

### Revision 2.6, June 2023

| Section | Change Summary |
|---|---|
| Pin Placement Considerations for Improved Noise Immunity | Fixed table headers in Table 17.1. Recommended Examples of DQS Group Allocation. |
| Technical Support Assistance | Added a link to the Lattice Support Database. |

### Revision 2.5, November 2021

| Section | Change Summary |
|---|---|
| All | Minor adjustments in formatting across the document. |
| Acronyms in This Document | Added this section. |
| DDR Software Primitives and Attributes | Updated delay information from 35 ps value to 38 to 45 ps in Input DDR Primitives section. |

### Revision 2.4, October 2021

| Section | Change Summary |
|---|---|
| All | • Changed document number from TN1180 to FPGA-TN-02184.<br>• Updated document template. |
| Disclaimers | Added this section. |
| Generic DDR Design Guidelines | Added a workaround option in the subsection Common Software Errors and Solutions for designs which require more true LVDS pairs in a DQS group than available. |

### Revision 2.3, April 2013

| Section | Change Summary |
|---|---|
| High-Speed DDR Interface Details | Updated GDDRX2_RX.DQS.Aligned Interface (“E” and “EA” Devices) figure. |
| Building Generic High-Speed Interfaces | Updated Interface Rules for GDDRX1_RX.DQS.Centered, GDDRX1_RX.DQS.Aligned, GDDRX2_RX.DQS.Aligned, GDDRX2_RX.DQS.Centered, GDDRX2_TX.DQSDLL.Centered, and GDDRX2_TX.PLL.Centered. |
| Generic DDR Design Guidelines | • Update the Pin Placement Guidelines for High-Speed Interfaces table.<br>• Added information to the High-Speed Clock Bridge (“EA” Devices) section. |

### Revision 2.2, November 2012

| Section | Change Summary |
|---|---|
| High-Speed DDR Interface Details | Updated GDDRX1_RX.DQS.Centered Interface (“E” and “EA” Devices) diagram. |
| DDR Software Primitives and Attributes | • Updated DQSDLL Configuration text section and removed DQSDLL Attributes table.<br>• Updated IDDRXD1 Waveform diagram.<br>• Updated ODDRXD Waveform diagram. |

### Revision 2.0, February 2012

| Section | Change Summary |
|---|---|
| All | Updated document with new corporate logo. |

### Revision 1.9, December 2011

| Section | Change Summary |
|---|---|
| Generic DDR Design Guidelines | Updated Generic DDR Design Guidelines with new sections on Design guidelines, Clocking guidelines, Common software errors and valid window calculation. |
| Implementing DDR/DDR2/DDR3 Memory Interfaces | Updated DDR3 Clock Synchronization Module with new table for DDR3 clock and PGROUP locations. |
| DDR3 Pinout Guidelines | Updated DDR3 Pinout Guidelines with a new sections on DDR3 pin placements for Improved Noise Immunity and table for DQS Group Allocation. |
| DDR Software Primitives and | • Updated ECLKSYNCA & ODDRX2D in the DDR Software Primitives and Attributes |

| | |
|---|---|
| Attributes | section.<br>- Updated DQSDLLB with domain transfer consideration on the UDDCNTLN input. |
| High-Speed DDR Interface Details | Updated GDDRX2_TX.DQSDLL.Centered Interface with the requirement that CLKOUT must be assigned to DQS pin and this pin cannot support LVDS IO Standard. |

### Revision 1.8, September 2011

| Section | Change Summary |
|---|---|
| DDR3 Pinout Guidelines | Updated the DDR3 Pinout Guidelines. |
| DDR3Termination Guidelines | Added the DDR3 Termination Guidelines and Layout Considerations. |
| DDR Software Primitives and Attributes | Updated the DQSDLL Update Control section. |

### Revision 1.7, April 2011

| Section | Change Summary |
|---|---|
| Multiple | Updated to include additional clarification in the Generic Timing Analysis, DDR Primitive and Attribute and DDR3 Clock Synchronization modules. |

### Revision 1.6, June 2010

| Section | Change Summary |
|---|---|
| Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond | Added Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond. |

### Revision 1.5, June 2010

| Section | Change Summary |
|---|---|
| All | Updated to reflect all the Generic DDR and DDR memory enhancements made in ispLEVER 8.0 Service Pack 1. |

### Revision 1.4, March 2010

| Section | Change Summary |
|---|---|
| DDR3 Termination Guidelines | Updated the termination scheme for DDR3 to external VTT termination. |

### Revision 1.3, November 2009

| Section | Change Summary |
|---|---|
| DDR3 Termination Guidelines | Updated DDR3 termination and pin assignments. |

### Revision 1.2, November 2009

| Section | Change Summary |
|---|---|
| Multiple | Updated for the LatticeECP3 “EA” device and for ispLEVER 8.0 software support. Includes new methodology to implement DDR interfaces in LatticeECP3 devices. |

### Revision 1.1, June 2009

| Section | Change Summary |
|---|---|
| All | Updated for ispLEVER 7.2 SP2. Some of the implementation listed here may not be valid if using ispLEVER 7.2 SP1. |

### Revision 1.0, February 2009

| Section | Change Summary |
|---|---|
| All | Initial release. |

www.latticesemi.com