ALTDLL and ALTDQ_DQS

Megafunctions User Guide

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Software Version: 9.1
Document Version: 5.0
Document Date: February 2012

UG-01032-5.0

Contents

Appendix A. Clear Box Generator

Additional Information

1. About these Megafunctions

The ALTDLL and ALTDQ_DQS megafunctions provide a custom external memory interface solution to access an FPGA's architecture and allow you to build your own custom external memory interface physical layer (PHY) blocks.

Altera® recommends that you use the ALTDLL and ALTDQ_DQS megafunctions when implementing a specialized or customized intellectual property (IP) for an Altera-supported external memory interface that is not supported in Altera's IP or a proprietary interface that is not supported by Altera.

The ALTDLL and ALTDQ_DQS custom external memory interface solution offers more efficient logic synthesis and device implementation, and saves valuable design time if you choose to code your own logic. The ALTDLL megafunction configures the dedicated DQS phase-shift circuitry, and the ALTDQ_DQS megafunction implements the read and write PHY required for the interface.

While the ALTDLL and ALTDQ_DQS custom external memory interface solution is primarily for building custom memory interface PHY blocks, you can also use this solution to interface with any external device, such as ASIC, ASSP or another FPGA, through the double data rate (DDR) interface.

The ALTDLL and ALTDQ_DQS megafunctions are specifically for memory interfaces that support memory burst lengths of two. For common memory interfaces that support memory burst lengths of four, Altera recommends that you use the ALTMEMPHY- or UniPHY-based memory controllers to take advantage of the benefits of Altera's IP and timing closure methodologies.

For more information about the ALTMEMPHY- or UniPHY-based memory controllers that Altera offers, refer to the volume 3 of the *External Memory Interface Handbook*.

Device Support

The ALTDLL and ALTDQ_DQS megafunctions support the following Altera device families:

- Arria® II GX
- HardCopy® III
- HardCopy IV
- Stratix® III
- Stratix IV

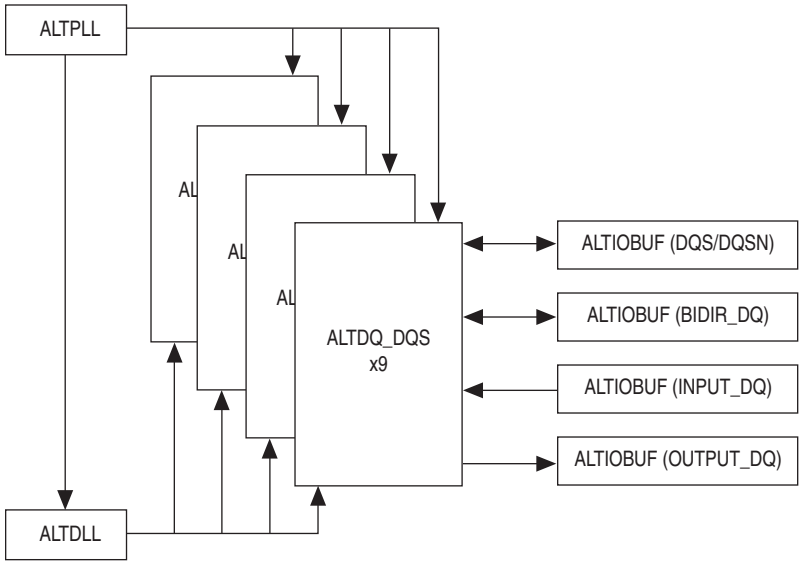
Features

The ALTDLL and ALTDQ_DQS megafunctions offer the following features:

- ALTDLL
  - A delay-locked loop (DLL) block to center-align the read strobe with read data.
  - Phase offset control blocks to fine-tune the delay time on the read strobe using static or dynamic offset.
- ALTDQ_DQS
  - Supports RLDRAM II memory interface.
  - DDR registers on the input and output paths to read or write to an external DDR interface.
  - Half-rate registers to enable successful data transfers between the I/O registers and the core logic.
  - Access to dynamic on-chip termination (OCT) controls to switch between parallel termination during reads to series termination during writes.
  - Access to I/O delay chains to fine-tune delays on the data or strobe signals statically or dynamically.

Figure 1–1 shows a high-level overview of how you can connect the ALTDQ_DQS megafunction with other megafunctions such as ALTPLL, ALTDLL, and ALTIOBUF, to create a full custom external memory interface. Figure 1–1 shows a 36-bit interface created with ALTDQ_DQS instantiations, where each instantiation is configured in the ×9 mode.

Figure 1–1. System-Level View

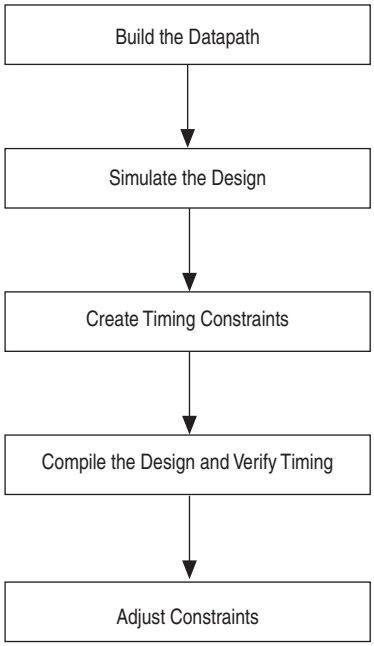


Design Flow

This chapter describes the FPGA design flow to implement a custom memory interface datapath using the ALTDLL and ALTDQ_DQS megafunctions and Altera's FPGA hardware features.

Figure 2–1 shows the design flow for creating a custom memory datapath system with the ALTDLL and ALTDQ_DQS megafunctions and the Quartus® II software.

Figure 2–1. Design Flowchart



Build the Datapath

After you identify the requirements for your custom external memory interface, the first stage is to build a datapath to interface with the memory blocks.

To build the datapath, you must perform the following steps:

1. Create a project in the Quartus II software that targets the preferred Altera device.
2. Instantiate the ALTPLL megafunction to provide the required clocking scheme for the custom PHY.

   For more information about instantiating megafunctions and the clocking scheme, refer to *Instantiate the ALTPLL Megafunction* section in volume 5 of the *External Memory Interface Handbook*. For more information about using PLLs, refer to the *ALTPLL Megafunction User Guide*.

3. Instantiate the ALTDLL megafunction to implement the DLL.

4. Instantiate the ALTDQ_DQS megafunction to implement the read and write PHY required for the interface.
5. Integrate the custom PHY with user logic, and a custom or third party memory controller if needed.
6. Instantiate the ALTIOBUF megafunction to use the I/O buffers for pin connections. This megafunction enables dynamic OCT capabilities for the respective interface pins.

   For more information about the pin connections, refer to "ALTIOBUF Megafunction and Delay Chains Integration" on page 4–18. For more information about the ALTIOBUF megafunction, refer to *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

7. Connect all the instances of ALTPLL, ALTDLL, ALTDQ_DQS, ALTIOBUF, and other custom memory controllers in the Quartus II software.

The following sections discuss other megafunctions or customized controller logic that are used in some cases.

ALTOCT Megafunction

If you use the OCT capabilities in the targeted devices, you eliminate the need for external series or parallel termination resistors, and you simplify the design of a PCB. If the I/O in your design uses calibrated series, parallel, or dynamic termination, your design requires a calibration block. This block requires a pair of R_{UP} and R_{DN} pins located in a bank that shares the same V_{CCIO} voltage as your memory interface. This calibration block is not required to be in the same bank or side of the device as the I/O elements it is serving. To use these capabilities in the FPGA, you must turn on the **Use dynamic OCT path** option when parameterizing the ALTDQ_DQS megafunction, and instantiate the ALTOCT megafunction.

For more information about the OCT capabilities in the DQ/DQS path, refer to "DQ/DQS OCT Path" on page 4–14.

Customized Controller Logic

In some cases, you require a customized controller logic to control the PHY created with the ALTDLL and ALTDQ_DQS instances. You must create a controller logic for the following instances:

- Controller logic for data, data_valid, and strobe pins for the custom external memory interface.
- If you use calibrated termination, controller logic for all pins in the ALTOCT instances associated with the custom external memory interface.

For more information about calibrated termination, refer to *Dynamic Calibrated On-Chip Termination (ALTOCT) Megafunction User Guide*.

Simulate the Design

After instantiating the megafunctions, the Quartus II software generates design source files and Verilog or VHDL simulation model files. Simulate these files in Modelsim-AE, Modelsim SE, or other third-party functional simulator tools.

For information about functional and gate-level timing simulations, refer to *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Create Timing Constraints

The ALTDLL and ALTDQ_DQS megafunctions do not provide automatic timing scripts for custom external memory interfaces. You must create your own timing constraints for the following paths and clocks:

- Timing paths from FPGA I/O to external device.
- Timing paths from I/O registers to core logic.
- PLL and other clock constraints.

After creating your constraints, perform the timing analysis using the TimeQuest timing analyzer in the Quartus II software.

Because the timing analysis for custom external memory interfaces are the same as the timing analysis for source-synchronous interfaces, refer to the *Timing Analysis* section in volume 3 of the *Quartus II Handbook* and *AN 433: Constraining and Analyzing Source-Synchronous Interfaces*.

The ALTDLL and ALTDQ_DQS custom PHY solution supports timing analysis using the TimeQuest timing analyzer with Synopsys Design Constraints (SDC) assignments. You can derive the timing constraints from the external device data sheet and tolerances from the board layout.

For more information about timing constraints, refer to "Appendix D: Interface Timing Analysis" section in *AN 328: Interfacing DDR2 SDRAM with Stratix II, Stratix II GX, and Arria GX Devices*.
For more information about creating timing constraints in SDC format for the TimeQuest timing analyzer, refer to the *The Quartus II TimeQuest Timing Analyzer* chapter of the *Quartus II Handbook*. Depending on which simulation tool you are using, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*.

Compile the Design and Verify Timing

After constraining your design, compile your design in the Quartus II software to generate timing reports to verify whether timing has been met.

After compiling your design in the Quartus II software, run the verifying timing script to produce the timing report for different paths, such as write data, read data, address and command, and core (entire interface) timing paths in your design.

The timing analyzer reports margins on the following paths:

- Address and command setup and hold margin
- Half-rate address and command setup and hold margin
- Core setup and hold margin
- Core reset and removal setup and hold margin
- Write setup and hold margin
- Read capture setup and hold margin

For more information about timing analysis and reporting using the ALTDLL and ALTDQ_DQS external memory solution, refer to the *Analyzing Timing of Memory IP* chapter in volume 2 of the *External Memory Interface Handbook*.

Adjust Constraints

The timing report shows the worst case setup and hold margin for the different paths in your design. If the setup and hold margin do not meet timing requirements, adjust the phase setting of the clocks that latch the data.

For example, the address and command outputs are clocked by an address and command clock that may be different than the system clock, which is 0°. The system clock clocks the clock outputs going to the memory. If the report timing script indicates that using the default phase setting for the address and command clock results in more hold time than setup time, adjust the address and command clock to be less negative than the default phase setting to ensure that there is less hold margin. Similarly, adjust the address and command clock to be more negative than the default phase setting if there is more setup margin.

Design Example: Implementing Read Paths Using Stratix III Devices

This section provides a walkthrough of a simple design example. The design example demonstrates a Stratix III device reading from an external DDR2 SDRAM. The DDR2 external memory interface is implemented using the ALTDLL and ALTDQ_DQS megafunctions. This design requires 1 DQS and 8 DQ input pins. The DQS frequency for the design is 150 MHz and the data rate is 300 Mbps.

For a more complex design example, refer to "Design Example: Implementing Half-Rate DDR2 Interface in Stratix III Devices" on page 4–49.

The design examples are available next to the *ALTDLL and ALTDQ_DQS Megafunctions User Guide* on the Documentation: User Guides page of the Altera website.

Generate the Megafunctions

Create a Quartus II project and generate the following megafunctions:

- ALTPLL megafunction
- ALTDLL megafunction
- ALTDQ_DQS megafunction
- ALTIOBUF megafunction

Create a Quartus II Project

Create a project in the Quartus II software that targets the EP3SL150F1152-C2 device for the DDR2 SDRAM by performing the following steps:

1. Open the **altdll_altdq_dqs_DesignExample_ex1.zip** file and extract the **altdll_altdq_dqs_design_ex1.qar** file.
2. In the Quartus II software, restore the **altdll_altdq_dqs_design_ex1.qar** file into your working directory.
3. Open the **altdll_altdq_dqs_design_ex1.bdf** file.

Generate the ALTPLL Megafunction

Before generating the ALTDLL and ALTDQ_DQS megafunctions, you must generate the ALTPLL megafunction first by performing the following steps:

1. Double-click anywhere on the Block Editor window. The Symbol window appears.
2. Click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard™ Plug-In Manager appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.
5. Select **Create a new custom megafunction variation**.
6. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTPLL**, and **Verilog HDL**, and type the file name as **PLL_50MHz.v**.
7. On the **Parameter Settings** tab, on the **General/Modes** page, specify the parameters as shown in Table 2–1. These parameters configure the general settings for the ALTPLL instance.

Table 2–1. ALTPLL Parameter Settings

Settings	Value
Currently selected device family	**Stratix III**
Match project/default	Turned on.
What is the frequency of the inclock0 input?	**50 MHz**
How will the PLL outputs be generated?	**With no compensation** This option is selected because the PLL is used to clock the ALTDLL instance only.

8. On the **Output Clocks** tab, on the **clk c0** page, specify the parameters as shown in Table 2–2. You don't have to parameterize the other pages on the **Output Clocks** tab because you only use one clock for this design.

Table 2–2. ALTPLL Output Clocks/clk c0 Settings

Settings	Value
Use this clock	Turned on
Enter output clock frequency	**150 Mhz**
Clock phase shift	**0 deg**
Clock duty cycle (%)	**50**

9. Click **Finish**.
10. Click **Finish**. The ALTPLL instance is generated.
11. Click **OK** to close the Symbol window.
12. Place the instance on the **altdll_altdq_dqs_design_ex1.bdf** Block Editor.

Generate the ALTDLL Megafunction

To generate the ALTDLL megafunction, perform the following steps:

1. Double-click anywhere on the Block Editor window. The Symbol window appears.
2. Click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.
5. Select **Create a new custom megafunction variation**.
6. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTDLL**, and **Verilog HDL**, and type the file name as **dll_150MHz.v**.
7. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in Table 2–3. These parameters configure the general settings for the ALTDLL instance.

Table 2–3. ALTDLL GeneraL Settings

Settings	Value
Currently selected device family	**Stratix III**
Match project/default	Turned on.
Number of Delay Chains	**12** Refer to *Stratix III Device Datasheet: DC and Switching Characteristics of Stratix III Devices* chapter in the *Stratix III Device Handbook*, and pick a DLL mode that supports 150 MHz and find the DLL setting.

Table 2–3. ALTDLL GeneraL Settings

Settings	Value
DQS Delay Buffer Mode	**Low** Refer to *Stratix III Device Datasheet: DC and Switching Characteristics of Stratix III Devices* chapter in the *Stratix III Device Handbook*, and pick a DLL mode that supports 150 MHz and find the DLL setting.
Input Clock Frequency	**150 MHz**
Turn on jitter reduction	Turned off.

8. On the **DLL Offset Controls/Optional Ports** page, specify the parameters as shown in Table 2–4.

Table 2–4. ALTDLL Parameter Settings/DLL Offset Controls/Optional Ports Settings

Settings	Value
DLL Phase Offset Control A **Instantiate dll_offset_ctrl block**	Turned off. The design is intended to run slow, so you do not need to select this parameter. However, if the read timing is unbalanced, you can fine-tune the DQS phase shift using this parameter.
DLL Phase Offset Control B **Instantiate dll_offset_ctrl block**	Turned off.
Optional Ports **Create a dll_aload port**	Turned off.
Optional Ports **Create a dll_dqsupdate port**	Turned off.

9. Click **Finish**.
10. Click **Finish**. The ALTDLL instance is generated.
11. Click **OK** to close the Symbol window.
12. Place the instance on the Block Editor.

Generate the ALTDQ_DQS Megafunction

To generate the ALTDQ_DQS megafunction, perform the following steps:

1. Double-click anywhere on the Block Editor window. The Symbol window appears.
2. Click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTDQ_DQS**, and **Verilog HDL**, and type the file name as **dq_dqs_input_path.v**.
5. On the **Parameter Settings** page, specify the parameters as shown in Table 2–5. These parameters configure the general settings for the ALTDQ_DQS instance.

Table 2–5. Parameter Settings

Parameter	Value
RLDRAMII Mode	**NONE**
Number of bidirectional DQ	**0**
Number of input DQ	**8**
Number of output DQ	**0**
Number of stages in dqs_delay_chain	**3**
DQS input frequency	**150 MHz**
Use half-rate components	Turned off. The design uses full-rate memory components, so you do not select this option.
Use Dynamic OCT	Turned off. Dynamic OCT is not used for input paths.
Add memory interface specific fitter grouping assignments	Turned on.

6. On the **Advanced Options** tab, on the **DQS IN** page, specify the parameters as shown in Table 2–6. These parameters configure the DQS input path of the ALTDQ_DQS instance.

Table 2–6. Advance Options (DQS IN)

Parameter	Sub-options	Value
Enable DQS Input Path	—	Turned on.
Enable dqs_delay_chain	—	Selected.
Advanced delay chain options	**Select dynamically using configuration registers**	Turned off.
	DQS delay chain 'delayctrlin' port source	**DLL** The DQS delay-chain settings is based on the DLL.
	DQS Delay Buffer Mode	**Low** Use the same mode selected in the DLL settings.
	DQS Phase Shift	**9000..** Specify a 90° DQS phase shift. The phase-shift value must inter-relate with the selected dqs_delay_chain stage.
	Enable DQS offset control	Turned off. Disable DQS delay fine-tuning using offset feature.
	Enable DQS delay chain latches	Turned off.

Table 2–6. Advance Options (DQS IN)

Parameter	Sub-options	Value
Enable DQS busout delay chain	—	Turned on.
Enable DQS enable block	—	Turned on.

7. On the **DQS OUT/OE** page, turn off the **Enable DQS output path** option. When you deselect the **Enable DQS output path** option, the other options on this page are disabled.
8. On the **DQ IN** page, specify the parameters as shown in Table 2–7. These parameters configure the DQ input path of the ALTDQ_DQS instance.

Table 2–7. Advance Options (DQ IN)

Options	Value
DQ input register mode	**DDIO** Select DDIO to enable double data rate capture for DQ.
DQ input register clock source	**dqs_bus_out port** and turn off **Connect DDIO clkn to DQS_BUS from complementary DQSn**
Use DQ input phase alignment	Turned off. The feature is for half-rate components; the design uses full-rate memory components.
Use DQ input delay chain	Turned on.

9. On the **DQ OUT/OE** page, all the options are automatically disabled because the design is not using output DQ. The parameters on this page configure the DQ output and OE paths of the ALTDQ_DQS instance.
10. On the **Half-rate** page, for the **IO Clock Divider Invert Phase** parameter, turn on **Never** because the design requires full-rate components. The other options are automatically disabled. The parameters on this page configure the half-rate settings of the ALTDQ_DQS instance.
11. On the **OCT Path** page, all the options are automatically disabled because the design is not using input and output DQS or bidirectional DQ. The parameters on this page configure the OCT path of the ALTDQ_DQS instance.
12. On the **DQSn I/O** page, turn off the **Use DQSn I/O** option because the design is not using DQSn. When you turn off the **Use DQSn I/O** option, the other options on this page are disabled.
13. In the **Reset/Config Ports** tab, tun off all the parameters.
14. Click **Finish**.
15. Click **Finish**. The ALTDQ_DQS instance is generated.
16. Click **OK** to close the Symbol window.
17. Place the instance on the Block Editor.

Generate the ALTIOBUF Megafunction

You must generate the ALTIOBUF megafunction to set the following I/O buffer settings:

- 1 input buffer for input DQS pin
- 8 input buffers for input DQ pins

To generate the ALTIOBUF megafunction, perform the following steps:

1. Double-click anywhere on the Block Editor window. The Symbol window appears.
2. Click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTIOBUF**, and **Verilog HDL**, and type the file name as **ibuf_input_dqs.v** (for DQS pin) or **ibuf_input_dq.v** (for DQ pins).
5. On the **Parameter Settings** page, specify the parameters as shown in Table 2–8. These parameters configure the general settings for the ALTIOBUF instance.

Table 2–8. ALTIOBUF General Settings

Settings	Value	
	1 input buffer for the input DQS pins	8 input buffer for the input DQ pins
Currently selected device family	Stratix III	Stratix III
How do you want to configure this module?	As an input buffer	As an input buffer
What is the number of buffers to be instantiated?	1	8
Use bus hold circuitry	Turned off.	Turned off.
Use differential mode	Turned off.	Turned on.
Use open drain output	Turned off.	Turned off.
Use output enable port	Turned off.	Turned off.
Use dynamic termination control	Turned off.	Turned off.
Use series and parallel termination control	Turned off.	Turned off.

6. On the **Dynamic Delay Chains** page, specify the parameters as shown in Table 2–9.

Table 2–9. ALTIOBUF Dynamic Delay Chain Settings

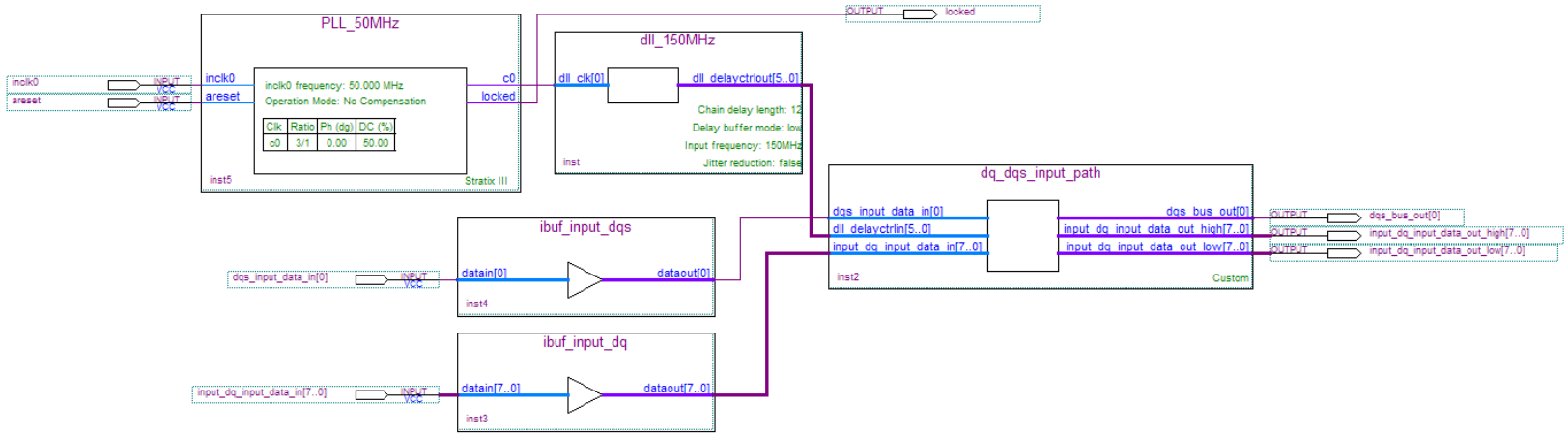
Settings	Value	
	1 input buffer for the input DQS pins	8 input buffer for the input DQ pins
Enable input buffer dynamic delay chain	Turned off.	Turned off.
Enable output buffer dynamic delay chain 1	Turned off.	Turned off.
Enable output buffer dynamic delay chain 2	Turned off.	Turned off.
Create a 'clkena' port	Turned off	Turned off.

7. Click **Finish**.
8. Click **Finish**. The ALTIOBUF instance is generated.
9. Click **OK** to close the Symbol window.
10. Place the instance on the Block Editor.

For more information about connecting all the instances, refer to "Integrate the I/O Buffer Modules with the ALTDQ_DQS modules" on page 4–55.

11. On the Block Editor, connect all the instances as shown in Figure 2–2 on page 2–11.

Figure 2–2. Block Diagram of the Design Example



Compile and Simulate the Design

On the Processing menu, click **Start Compilation** to compile the design. After the design is compiled, you can view the implemention in the RTL Viewer. You can also view the resource usage in the Compilation Report.

After you compile your design, simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior. Set up and simulate the design in the ModelSim-Altera software by performing the following steps:

1. Unzip the **altdll_altdq_dqs_ex1_msim.zip** file to your preferred working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files in the **altdll_altdq_dqs_ex1_msim.zip** folder.
5. Click **OK**.
6. On the Tools menu, point to **Tcl** and click **Execute Macro**.
7. Select the **altdll_altdq_dqs_ex1_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.
8. Verify the results with the simulation waveform.

You can rearrange, remove and add signals, and change the radix by modifying the script in the **altdll_altdq_dqs_ex1_msim.do** file.

3. Parameter Settings

The Quartus II software provides the MegaWizard Plug-In Manager that helps you quickly customize your megafunction variation. The parameter editor provides a list of megafunctions and available options for each variation.

Altera recommends that you use the parameter editor to instantiate the ALTDLL and ALTDQ_DQS megafunctions. However, for advanced users, if you want to bypass the MegaWizard Plug-In Manager and use the megafunctions as directly parameterized instantiations in your design, you can use the clear box generator. For more information about the clear box generator, refer to Appendix A, Clear Box Generator.

Some advanced parameters can only be modified through the clear box parameters.

ALTDLL Parameter Editor

This section provides information about the ALTDLL MegaWizard parameters.

For advanced users who may use the clearbox generator, the clearbox parameter names are provided for the corresponding MegaWizard parameters.

The ALTDLL **Parameter Settings** page in the ALTDLL parameter editor allows you to configure the parameters in the following pages:

- **General**
- **DLL Offset Controls/Optional Ports**

Table 3–1 shows the options available on the **General** page.

Table 3–1. Options on General Settings Page

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Number of Delay Chains	**6**, **8**, **10**, **12**, or **16**	DELAY_CHAIN_LENGTH	Represents the number of delay buffers in the delay loop. The DLL consists of 6, 8, 10, 12, or 16 DLL-controlled delay buffers chained together. The total delay in the DLL delay chain is computed with the following equation: delay = delay_chain_length x delay_buffer_delay The DLL uses the delay chain to implement a 360° phase shift. By comparing the incoming clock to the 360°-shifted clock, the DLL determines the delay setting to implement an actual 360° phase shift in its delay chain. Because each delay buffer is identical, each buffer in the delay chain implements a phase shift that is equal to (360/delay_chain_length)°. The default value is **12**.
DQS Delay Buffer Mode	**Low** or **High**	DELAY_BUFFER_MODE	Specifies the frequency mode for the variable delay buffers. If you select **Low**, the dll_offset_ctrl_a_offsetctrlout [5..0] or dll_offset_ctrl_b_offsetctrlout [5..0] output is limited to a maximum value of 63. If you select **High**, the output is limited to a maximum value of 31. The default value is **Low**.
Input Clock Frequency		INPUT_FREQUENCY	Specifies the frequency of the clock (in MHz) that is connected to the clk input port. This frequency must be within the valid range for the device you are using. You can specify a duration in ps. The value is in floating-point format with no decimal point limit. The default value is **300** MHz. For information about the clock range for the Altera devices, refer to the respective device handbook.
Turn on jitter reduction	—	JITTER_REDUCTION	Enables the jitter reduction circuit. Jitter affects the signal integrity of the clock signal from a PLL clock source or an external clock pin. If you turn on this parameter, the jitter reduction circuit is enabled on the dll_delayctrlout[5..0] and dll_offset_ctrl_a_offsetctrlout [5..0], or the dll_offset_ctrl_b_offsetctrlout [5..0] output port. When the jitter reduction circuit is enabled, the DLL may require up to 1,024 clock cycles to lock. When the jitter reduction circuit is disabled, the DLL requires only up to 256 clock cycles to lock.

The **DLL Offset Controls/Optional Ports** page allows you to instantiate the DLL offset control blocks (A and B), specify whether to use static offset, and create the `dll_aload` and `dll_dqsupdate` optional ports. Table 3–2 shows the options available on **DLL Offset Controls/Optional Ports** page.

Table 3–2. Options on DLL Offset Controls/Optional Ports Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DLL Phase Offset Control A **Instantiate dll_offset_ctrl block**	**Set statically to** or **Set dynamically using offset input port**	USE_DLL_OFFSET_CTRL_A	Instantiates DLL_OFFSET_CTRL_A block. The block can be placed either at the top, bottom, or side of the FPGA device, depending on how the Quartus II Fitter places it. If you turn on this parameter, you must specify whether you want to set the blocks statically or dynamically.
	–63 to **63**	DLL_OFFSET_CTRL_A_STATIC_OFFSET	The **Set statically to** option is a signed integer. Turn on this option if you want a fixed offset value, and key in the value you want. This fixed value is added to the DLL feedback counter and the output is generated on the `dll_offset_ctrl_a_offsetctrlout [5..0]` output port. The default value is **0**.
	—	DLL_OFFSET_CTRL_A_USE_OFFSET	The **Set dynamically using offset input port** option determines the output of the `dll_offset_ctrl_a_offsetctrlout [5..0]` output port. Turn on this option if you want a dynamic offset value. If you turn on this option, depending on whether the `dll_offset_ctrl_a_addnsub` signal is asserted or not, the phase offset specified on the offset input bus is added or subtracted from the DLL feedback counter output to get the `dll_offset_ctrl_a_offsetctrlout [5..0]` output.

Table 3–2. Options on DLL Offset Controls/Optional Ports Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DLL Phase Offset Control B **Instantiate dll offset_ctrl block**	**Set statically to** or **Set dynamically using offset input port**	USE_DLL_OFFSET_CTRL_B	Instantiates DLL_OFFSET_CTRL_B block. The block can be placed either at the top, bottom, or side of the FPGA device, depending on how the Quartus II Fitter places it. If you turn on this option, you must specify whether you want to set the blocks statically or dynamically.
	–63 to **63**	DLL_OFFSET_CTRL_B_STATIC_OFFSET	The **Set statically to** option is a signed integer. Turn on this option if you want a fixed offset value, and key in the value you want. This fixed value is added to the DLL feedback counter and the output is generated on the dll_offset_ctrl_b_offsetctrlout[5..0]output port. The default value is **0**.
	—	DLL_OFFSET_CTRL_B_USE_OFFSET	The **Set dynamically using offset input port** option determines the output of the dll_offset_ctrl_b_offsetctrlout [5..0] output bus. Turn on this option if you want a dynamic offset value. If you turn on this option, depending on whether the dll_offset_ctrl_b_addnsub signal is asserted or not, the phase offset specified on the offset input bus is added or subtracted from the DLL feedback counter output to get the dll_offset_ctrl_b_offsetctrlout [5..0]output.
Optional Ports **Create a dll_aload port**	—	DLL_ALOAD	Enables the asynchronous-load signal for the DLL up or down counter. When the dll_aload signal is high, the counter is asynchronously loaded with the initial delay setting of 16 in low-frequency mode when you select **Low** for the **DQS Delay Buffer Mode** parameter, or 32 in high-frequency mode when you select **High** for the **DQS Delay Buffer Mode** parameter. This input defaults to GND.
Optional Ports **Create a 'dll_dqsupdate' port**	—	DLL_DQSUPDATE	Enables the update-enable signal for the delay-setting latches in the DQS pins. This signal only feeds the dqsupdateen port of the ALTDQ_DQS megafunction. To use the dll_dqsupdate signal, you must turn on the **Enable DQS delay chain latches** option on the **DQS IN** page in the ALTDQ_DQS parameter editor.

The **Simulation Model** page allows you to optionally generate simulation model files. The **Summary** page displays a list of the types of files to be generated. The automatically generated variation file contains wrapper code in the language you specified earlier. On this page, you can specify additional types of files to be generated.

Choose from the following file types:

- Quartus II IP file (<*function name*>.**qip**)
- Instantiation template file (<*function name*>.**v**)
- Verilog HDL black box file (<*function name*>**_bb.v**)
- AHDL Include file (<*function name*>.inc)
- VHDL component declaration file (<*function name*>.**cmp**)
- Quartus II symbol file (<*function name*>.**bsf**)

If you select **Generate netlist** on the **Simulation Model** page, the file for that netlist is also available. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates generation of an optional file

ALTDQ_DQS Parameter Editor

This section provides information about the ALTDQ_DQS MegaWizard parameters.

For advanced users who may use the clearbox generator, the clearbox parameter names are provided for the corresponding MegaWizard parameters.

The **Parameter Settings** page in the ALTDQ_DQS parameter editor allows you to configure the parameters in Table 3–3.

Table 3–3. Options on Parameter Settings Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
RLDRAMII mode	**NONE**, **x9**, **x18**, or **x36**	RLDRAMII_MODE	Enables RLDRAM II support for ALTDQ_DQS instance. If you select **x9** or **x18** mode, the DK pins do not have group assignments, but they must be placed in the same bank or chip edge as the other pins in the interface. If you select **x36** mode, the DK/DK# pins must be placed manually in DQS locations. If you select **x18** mode, place the DM pins in either group 0 or group 1, which forces QVLD to the other group. If you select **x36** mode, place the DM pins in group 0 or 1, and QVLD to be in group 0 or 1. All combinations are allowed. Not supported in Arria II GX.
Data mask pin group	**NONE**, **GROUP0**, or **GROUP1**	DM_LOC	Specifies the group assignment for the DM pin group. If you select **NONE** for the **RLDRAMII mode** option, then this option defaults to **NONE**. If you select **x9** for the **RLDRAMII mode** option, then this option defaults to **NONE**. If you select **x18** for the **RLDRAMII mode** option, then for this option you can select either **NONE**, **GROUP0**, or **GROUP1**. If you select **GROUP0**, then **GROUP1** is used for the **Q valid signal group** option, and if you select **GROUP1**, then **GROUP0** is used for the **Q valid signal group** option. If you select **x36** for the **RLDRAMII mode** option, then for this option you can select either **NONE**, **GROUP0**, or **GROUP1**. Not supported in Arria II GX devices.
Q valid signal group	**NONE**, **GROUP0**, or **GROUP1**	QVLD_LOC	Specifies the group assignment for the Q valid signal group. If you select **NONE** for the **RLDRAMII mode** option, then this option defaults to **NONE**. If you select **x9** for the **RLDRAMII mode** option, then this option defaults to **GROUP0**. If you select **x18** for the **RLDRAMII mode** option, then this option depends on the **Data mask pin group** option. If you select **GROUP0** for the **Data mask pin group** option, then **GROUP1** is defaulted for this option, and if you select **GROUP1** for the **Data mask pin group** option, then **GROUP0** is defaulted for this option. If you select **x36** for the **RLDRAMII mode** option, then for this option you can select either **NONE**, **GROUP0**, or **GROUP1**. Not supported in Arria II GX devices.
Number of bidirectional DQ	**0–48**	NUMBER_OF_BIDIR_DQ	Specifies the number of bidirectional DQ ports used in the ALTDQ_DQS instance.

Table 3–3. Options on Parameter Settings Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Number of input DQ	**0–48**	NUMBER_OF_INPUT_DQ	Specifies the number of input DQ ports used in the ALTDQ_DQS instance.
Number of output DQ	**0–48**	NUMBER_OF_OUTPUT_DQ	Specifies the number of output DQ ports used in the ALTDQ_DQS instance.
Number of stages in dqs_delay_chain	**1**, **2**, **3**, and **4**	DQS_DELAY_CHAIN_PHASE_SETTING	Specifies the stages of DQS_DELAY_CHAIN. The number of stages depends on the intended phase shift that you want to clock for <*IO*>_DDIO_IN block in the DQ input path. The bigger the value you specify, the longer the delay. The coarse phase shift depends on this option. For example, in Stratix IV devices, if you set the frequency mode to **1**, you will get a phase shift of 20°, 60°, 90°, or 120°. If you set **Number of stages in dqs_delay_chain** value to **2**, you will get 60° phase shift and if you set the **Number of stages in dqs_delay_chain** value to **1**, you will get 30° phase shift.
DQS input frequency	—	DQS_INPUT_FREQUENCY	Specifies the input frequency of the DQS strobe in MHz. The input frequency must match the DLL (ALTDLL) input frequency.
Use half rate components	—	USE_HALF_RATE	Instantiates the half-rate blocks in the ALTDQ_DQS instance. This parameter is used only when the external memory interface requires half-rate mode. Not supported in Arria II GX devices.
Use dynamic OCT path	—	USE_DYNAMIC_OCT	Instantiates the dynamic OCT blocks in the ALTDQ_DQS instance. This parameter enables access to dynamic OCT paths on both DQ and DQS paths. The dynamic OCT features enable parallel termination (R_t) during reads from the external memory and disable R_t during writes to the external memory. Not supported in Arria II GX devices.
Add memory interface specific fitter grouping assignments	—	ADD_MEM_FITTER_GROUP_ASSIGNMENTS	Enables the Quartus II Fitter to automatically assign the memory interface I/O ports to the memory interface I/O pins on the FPGA.

The **Advanced Options** page allows you to configure the parameters in the following pages:

- **DQS IN**
- **DQS OUT/OE**
- **DQ IN**
- **DQ OUT/OE**
- **Half-rate**
- **OCT Path**
- **DQSn I/O**

- **Reset/Config Ports**

Table 3–4 describes the options available on the **DQS IN** page. This page allows you to configure the DQS input path. For more information about the DQS input path, refer to "DQS Input Path" on page 4–6.

Table 3–4. Options on DQS IN Page (Part 1 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Enable DQS Input Path	—	USE_DQS_INPUT_PATH	Instantiates the DQS input path.
Enable DQS Input Path	—	USE_DQS_INPUT_PATH	Instantiates the DQS input path.
Delay chain usage: **Enable dynamic delay chain**	—	USE_DQS_INPUT_DELAY_CHAIN	Enables *<IO>*_INPUT_DELAY_CHAIN (D1) on the DQS input path. If you turn on this parameter, DQS_DELAY_CHAIN block in the path is disabled. D1 is a run-time adjustable delay chain. To configure delay chains dynamically, refer to "Delay Chains" on page 4–15.
Delay chain usage: **Enable dqs_delay_chain**	—	USE_DQS_DELAY_CHAIN	Enables DQS_DELAY_CHAIN block. The DQS delay chain is a DLL-controlled delay chain used to phase shift the DQS read clock.
Enable DQS busout delay chain	—	USE_DQSBUSOUT_DELAY_CHAIN	Enables DQSBUSOUT_DELAY_CHAIN (Da). This busout delay chain fine-tunes the outputs of DQS_DELAY_CHAIN block so that the DQS strobe timing matches the DQS enable signal. The DQS strobe has 15 steppable delays, with each step having 50 ps of delay. Da is a run-time adjustable delay chain.
Enable DQS enable block	—	USE_DQS_ENABLE	Enables DQS_ENABLE block. This block grounds the DQS input strobe when the strobe goes to high impedance state (Z) after a DDR read postamble.
Enable DQS enable control block	—	USE_DQS_ENABLE_CTRL	Enables DQS_ENABLE_CTRL block that controls a DQS enable circuitry. You must determine an efficient working resync_postamble_clk clock phase which clocks this block to ensure smooth data transfer. The ALTDQ_DQS megafunction cannot determine the phase for the data transfer. Use round trip delay (RTD) analysis or create a custom data training circuitry to write and read back a training pattern to and from the memory device and then dynamically adjust the PLL's resyncronization clock phase to find an efficient working phase. Even though this block controls the DQS enable signal, the megafunction does not consider the necessary timing for this signal. Refer to the external memory interface requirements for the necessary timing.

Table 3–4. Options on DQS IN Page (Part 2 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Enable DQS enable block delay chain	—	—	Enables DQS_ENABLE_DELAY_CHAIN (Db) that fine-tunes the outputs of DQS_ENABLE_CTRL block so that the DQS enable signal timing matches the DQS strobe. Db is a run-time adjustable delay chain.
Advanced Delay Chain Options **Set dynamically using configuration registers**	—	USE_DQS_DELAY_CHAIN_PHASECTRLIN	Determines the phasectrlin input for the phase setting. If you turn on this option, it dynamically chooses the phase applied to the dqsbusout output during the FPGA run time. If you turn off this option, the phase setting is determined by the **Number of stages in dqs_delay_chain** option in the **Parameter Settings** page. This delay chain fine-tunes the DQS strobe signal.
Advanced Delay Chain Options **DQS delay chain delayctrlin port source**	**DLL** or **Core**	DQS_DELAY_CHAIN_DELAYCTRLIN_SOURCE	Determines whether you want the delayctrlin port to be controlled by **DLL** (outputs) or from the **Core** (FPGA). If you select **DLL**, the dll_delayctrlin[5..0] port is connected to the dll_delayctrlout[5..0] port of the DLL. The **DLL** option adjusts the delay setting in DQS_DELAY_CHAIN block across pressure, volume, and temperature (PVT). Altera recommends that you always select **DLL** to optimize the read capture at the DQ input register. If you select **Core**, the core_delayctrlin port is fed by the core.
Advanced Delay Chain Options **DQS Delay Buffer Mode**	**Low** or **High**	DELAY_BUFFER_MODE	Specifies whether the variable delay buffers in the DQS_DELAY_CHAIN work in low-frequency or high-frequency mode. The frequency mode must match the frequency mode you select for the **DQS Delay Buffer Mode** parameter on the **Parameter Settings** page in the ALTDLL parameter editor.
Advanced Delay Chain Options **DQS Phase Shift**	**0–36,000**	DQS_PHASE_SHIFT	Specifies the phase shift between the delayed DQS signal and the input DQS signal in units of hundreds of degrees, for example, a 90° phase shift is represented as 9,000. Use this parameter for static timing analysis only because timing analysis cannot determine the phase shift through the delayctrlin[5..0], phasectrlin[2..0], and offsetctrlin[5..0] ports on the megafunction the way a simulation can. This is an optional field and defaults to 0.
Advanced Delay Chain Options **Enable DQS offset control**	—	DQS_OFFSETCTRL_ENABLE	Enables offset values to be added to DQS_DELAY_CHAIN block. If you turn on this option, make sure that the ALTDLL instance is set to use the DLL offset control blocks. This option connects the outputs from the DLL offset control blocks to the DQS delay chain block. This parameter is optional and turned off by default.

Table 3–4. Options on DQS IN Page (Part 3 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Advanced Delay Chain Options **Enable DQS delay chain latches**	—	DQS_CTRL_LATCHES_ENABLE	Enables the delayctrlin[5..0] and offsetctrlin[5..0] inputs to be registered by the dqsupdateen signal. The DLL continues changing its delay settings value due to the feedback system. These DLL values are propagated through the delayctrlout and offsetctrlout signals of the DLL and DLL offset control blocks to DQS_DELAY_CHAIN block to calibrate the necessary delay settings. These values are updated based on the dll_dqsupdate port from the DLL, which is connected to the dqsupdateen port. To use this option, you must turn on the **Create a 'use dll_dqsupdate' port** option on the **DLL Offset Controls/Optional Ports** page in the ALTDLL parameter editor.
Advanced Enable Control Options **DQS Enable Control Phase Setting**	**Set statically to** or **Set dynamically using configuration registers**	DQS_ENABLE_CTRL_PHASE_SETTING	If you turn on the **Set statically to** option, you can select the phase setting for the delay chains from 0 up to 4 to fine-tune the DQS enable signal. If you turn on the **Select dynamically using configuration registers** option, the phase setting is determined by the phasectrlin input for the delay chains.
Advanced Enable Control Options **DQS Enable Control Invert Phase**	**Always**, **Never**, or **Based on configuration registers**	DQS_ENABLE_CTRL_INVERT_PHASE	If you turn on **Always**, the phase output is inverted. If you turn on **Never**, the phase output is not inverted. If you turn on **Based on configuration registers**, the phaseinvertctrl input determines whether or not the inverter is used. The inverter can be used to increase the number of available phases. This is an optional field and defaults to **Never**.
Enable DQS enable block delay chain	—	USE_DQSENABLE_DELAY_CHAIN	Enables DQS_ENABLE_DELAY_CHAIN. This delay chain fine-tunes the outputs of DQS_ENABLE_CTRL block so that the DQS enable signal timing matches the DQS strobe. This delay chain is a run-time adjustable delay chain.

Table 3–5 describes options available on the **DQS OUT/OE** page. This page allows you to configure the DQS output and output enable (OE) paths. For more information about the DQS output and OE paths, refer to "DQS Output/OE Path" on page 4–12.

Table 3–5. Options on DQS OUT/OE Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Enable DQS output path	—	USE_DQS_OUTPUT_PATH	Instantiates the DQS output path.
Enable DQS output path	—	USE_DQS_OUTPUT_PATH	Instantiates the DQS output path.

Table 3–5. Options on DQS OUT/OE Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DQS Output Path Options **Enable DQS output delay chain1**	—	USE_DQS_OUTPUT_DELAY_CHAIN1	Enables DQS_OUTPUT_DELAY_CHAIN1 (D5) in the DQS output path. This parameter is used for deskew purposes or SSN reduction. D5 is a run-time adjustable delay chain.
DQS Output Path Options **Enable DQS output delay chain2**	—	USE_DQS_OUTPUT_DELAY_CHAIN2	Enables DQS_OUTPUT_DELAY_CHAIN2 (D6) in the DQS output path. This parameter is used for deskew purposes or SSN reduction. D6 is a run-time adjustable delay chain.
DQS Output Path Options **DQS output register mode**	**Not used**, **FF**, or **DDIO**	DQS_OUTPUT_REG_MODE	Enables the DQS_OUTPUT_FF or DQS_OUTPUT_DDIO_OUT output registers. Select **FF** if you want flip-flop output registers or **DDIO** if you want double data rate I/O registers.
DQS Output Enable Options **Enable DQS output enable**	—	USE_DQS_OE_PATH	Instantiates DQS output enable path.
DQS Output Enable Options **Enable DQS output enable delay chain1**	—	USE_DQS_OE_DELAY_CHAIN1	Enables DQS_OUTPUT_DELAY_CHAIN1 (D5) in the DQS OE path. This parameter is used for deskew purposes or SSN reduction. D6 is a run-time adjustable delay chain.
DQS Output Enable Options **Enable DQS output enable delay chain2**	—	USE_DQS_OE_DELAY_CHAIN2	Enables DQS_OUTPUT_DELAY_CHAIN2 (D6) in the DQS OE path. This parameter is used for deskew purposes or SSN reduction. D6 is a run-time adjustable delay chain.
DQS Output Enable Options **DQS output enable register mode**	**Not used**, **FF**, or **DDIO**	DQS_OE_REG_MODE	Enables the DQS_OUTPUT_FF or DQS_OUTPUT_DDIO_OUT output registers. Select **FF** if you want flip-flop registers or **DDIO** if you want double data rate I/O registers.

Table 3–6 describes options available on the **DQ IN** page. This page allows you to configure the DQ input path. For more information about the DQ input path, refer to "DQ Input Path" on page 4–8.

Table 3–6. Options on DQ IN Page (Part 1 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DQ Input Register Options **DQ input register mode**	**Not used**, **FF**, or **DDIO**	DQ_INPUT_REG_MODE	Enables the DQ input registers (*<IO>*_INPUT_FF or *<IO>*_DDIO_IN registers). Select **FF** if you want flip-flop registers or **DDIO** if you want double data rate I/O registers.
DQ Input Register Options **DQ input register clock source**	**'dqs_bus_out' port**, **Inverted 'dqs_bus_out' port**, or **Core**	DQ_INPUT_REG_CLK_SOURCE	Specifies how the DQ input registers should be clocked. You can either clock it from the **'dqs_bus_out' port** (DQS input path), the **Inverted 'dqs_bus_out' port** (DQS input path), or directly from the **Core** (FPGA). Altera recommends that you turn on the **'dqs_bus_out' port** option to clock the DQ input register. When reading from the external memory, the DQ data that comes into the DDIO must be center-aligned with the DQS strobe that goes through the DQS input path and comes out the dqs_bus_out port. By center-aligning the DDIO with DQS strobe, you maximize the setup and hold margins at the DQ input register. You can also connect the dqs_bus_out port to the full-rate DQ input register for complementary clocking purpose as used in QDR and QDR II applications. You can connect the dqs_bus_out port by turning on the **Connect DDIO clkn to DQS_BUS from complementary DQSn** option.
DQ Input Register Options **Use DQ input phase alignment**	—	USE_DQ_IPA	Enables the input phase alignment (*<IO>*_IPA_LOW or *<IO>*_IPA_HIGH) blocks. The input phase alignment blocks represent the circuitry required to phase-shift the input signal the DQ data for resynchronization and alignment purpose. The resynchronization and alignment are done to match the arrival delay of the DQS (triggered by the fly-by clock on a DDR-DIMM) to the latest arrival delay of a DQS from the DIMM. Because this block is meant for resynchronization, the ALTDQ_DQS megafunction does not consider the clocking requirements of this block. You must figure the clocking requirements using the RTD analysis or create a custom data training circuitry to read or write back a training pattern to and from the memory device, and then dynamically adjust the PLL's resyncronization clock phase to find a good working phase. For more component information about the available alignment and resynchronization registers in this block, refer to the "I/O Element (IOE) Registers" section in the *External Memory Interface* chapter of the respective device handbooks. For the available levelling delay chains in this block, refer to the "Leveling Circuitry" section in the *External Memory Interface* chapter of the respective device handbooks.

Table 3–6. Options on DQ IN Page (Part 2 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DQ Input Register Options **Use DQ resync register**	—	DQ_RESYNC_REG_MODE	Enables the DQ resynchronization register. Supported in Arria II GX devices only.
DQ Input Register Options **Use DQ half rate 'dataoutbypass' port**	—	DQ_HALF_RATE_USE_DATAOUTBYPASS	If you turn on this parameter, the dataoutbypass input dynamically routes the directin input to the dataout output for *<IO>*_HALF_RATE_INPUT block. Using this parameter, you can bypass the half-rate registers in *<IO>*_HALF_RATE_INPUT block dynamically during the FPGA run-time. Not supported in Arria II GX devices.
Advanced DQ IPA Options **DQ Input Phase Alignment Phase Setting**	**Set statically to** or **Set dynamically using configuration registers**	DQ_IPA_PHASE_SETTING	If you turn on the **Set statically to** option, the phase setting can be selected from values **0** to **7** for the delay chains. If you turn on the **Select dynamically using configuration registers** option, the phase setting is determined by the phasectrlin input for the delay chains. This parameter fine-tunes the resynchronization phase for the DQ input data. The phase settings are also called the levelling delay chains that handle the fly-by clock topology in DDR3 interfaces.
Advanced DQ IPA Options **Add DQ Input Phase Alignment Input Cycle Delay**	**Always**, **Never**, or **Based on configuration registers**	DQ_IPA_ADD_INPUT_CYCLE_DELAY	If you turn on **Always**, a single cycle delay is added to the input path. If you turn on **Never**, no delay is added. If you turn on **Based on configuration registers**, the enainputcycledelaysetting input controls whether or not a single cycle delay is added to the input path.
Advanced DQ IPA Options **Invert DQ Input Phase Alignment Phase**	**Always**, **Never**, or **Based on configuration registers**	DQ_IPA_INVERT_PHASE	If you turn on **Always**, the phase output is inverted. If you turn on **Never**, the phase output is not inverted. If you turn on **Based on configuration registers**, the phaseinvertctrl input determines whether or not the inverter is used. The inverter is used to increase the number of available phases.
Advanced DQ IPA Options **Register DQ input phase alignment bypass output**	—	DQ_IPA_BYPASS_OUTPUT_REGISTER	Controls the output register in the DQ input path. If you turn on this option, the output data bypasses the output register. If you turn off this option, then the data goes through the output register.

Table 3–6. Options on DQ IN Page (Part 3 of 3)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Advanced DQ IPA Options **Register DQ input phase alignment add phase transfer**	—	DQ_IPA_ADD_PHASE_TRANSFER_REG	If you turn on this option, a negative edge-triggered register is added in the data path for the clock phase transfer. If you turn off this option, no register is added. The negative-edge register is used to guarantee the setup and hold time for a phase transfer.
Use DQ input delay chain	—	USE_DQ_INPUT_DELAY_CHAIN	Enables *<IO>*_INPUT_DELAY_CHAIN (D1). This parameter is used for deskew purposes or SSN reduction on the DQ input path. Not supported in Arria II GX devices. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.

Table 3–7 describes options available on the **DQ OUT/OE** page. This page allows you to configure the DQ output and OE paths. For more information about the DQ output and OE paths, refer to "DQ Output/OE Path" on page 4–10.

Table 3–7. Options on DQ OUT/OE Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DQ Output Path Options **Enable DQ output delay chain1**	—	USE_DQ_OUTPUT_DELAY_CHAIN1	Enables *<IO>*_OUTPUT_DELAY_CHAIN1 (D5) in the DQ output path. This parameter is used for deskew purposes or SSN reduction. D5 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.
DQ Output Path Options **Enable DQ output delay chain2**	—	USE_DQ_OUTPUT_DELAY_CHAIN2	Enables *<IO>*_OUTPUT_DELAY_CHAIN2 (D6) in the DQ output path. This parameter is used for deskew purposes or SSN reduction. D6 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.
DQ Output Path Options **DQ output register mode**	**Not used**, **FF**, or **DDIO**	DQ_OUTPUT_REG_MODE	Enables the full-rate DQ output registers (*<IO>*_OUTPUT_FF or *<IO>*_OUTPUT_DDIO_OUT registers).
DQ Output Enable Options **Enable DQ output enable**	—	USE_DQ_OE_PATH	Instantiates the DQ output enable path.
DQ Output Enable Options **Enable DQ output enable delay chain1**	—	USE_DQ_OE_DELAY_CHAIN1	Enables *<IO>*_OE_DELAY_CHAIN1 (D5) in the DQ OE path. This parameter is used for deskew purposes or SSN reduction. D5 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.

Table 3–7. Options on DQ OUT/OE Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
DQ Output Enable Options **Enable DQ output enable delay chain2**	—	USE_DQ_OE_DELAY_CHAIN2	Enables *<IO>*_OE_DELAY_CHAIN2 (D6) in the DQ OE path. This parameter is used for deskew purposes or SSN reduction. D6 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.
DQ Output Enable Options **DQ output enable register mode**	**Not used**, **FF**, or **DDIO**	DQ_OE_REG_MODE	Enables the full-rate DQ output-enable registers (*<IO>*_OE_FF or *<IO>*_OE_DDIO_OE registers). Select **FF** if you want flip-flop registers or **DDIO** if you want double data rate I/O registers.

Table 3–8 describes the options available on the **Half-rate** page.

Table 3–8. Options on Half-Rate Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
IO Clock Divider Source	**Core**, **'dqs_bus_out' port**, or **Inverted 'dqs_bus_out' port**	IO_CLOCK_DIVIDER_CLK_SOURCE	Specifies the I/O clock divider clock source which can be from the **Core** (FPGA), the **'dqs_bus_out' port** (DQS input path), or the **Inverted 'dqs_bus_out' port** (DQS input path). Altera recommends that you turn on the **'dqs_bus_out' port** option to clock the DQ input register. When reading from the external memory, the DQ data that comes from the full-rate DQ input registers must be synchronized to the half-rate input block, if half-rate interfaces are used. If the full-rate DQ input registers are clocked by the DQS input path via the dqs_bus_out port, then the I/O clock divider (and other clock source settings) must also be clocked via the dqs_bus_out port.
Create 'io_clock_divider_masterin' input port	—	USE_IO_CLOCK_DIVIDER_MASTERIN	Enables the masterin input to synchronize this divider with another I/O clock divider. If you turn off this option, this divider operates independently. This mode is meant for the master divider of a group of dividers. Turn on this parameter when you chain the I/O clock divider blocks from multiple ALTDQ_DQS instances.
Create 'io_clock_divider_clkout' output port	—	—	Divides the clock output signal by two. The clock out signal can be connected to the clock input of a half-rate Input block or fed to the FPGA core.

Table 3–8. Options on Half-Rate Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Create 'io_clock_divider _slaveout' output port	—	USE_IO_CLOCK_ DIVIDER_SLAVEOUT	Enables the output of the divider's D flip-flop (DFF). The output signal can only be connected to the masterin input of another I/O clock divider block and it cannot have more than one fan-out. Turn on this parameter when you chain the I/O clock divider blocks from multiple ALTDQ_DQS instances.
IO Clock Divider Invert Phase	**Always**, **Never**, or **Based on register configuration**	IO_CLOCK_DIVIDER_ INVERT_PHASE	If you turn on **Always**, the phase output is inverted. If you turn on **Never**, the phase output is not inverted. If you turn on **Based on register configuration**, the phaseinvertctrl input determines whether or not the inverter is used. The inverter can be used to increase the number of available phases.

Table 3–9 on page 3–16 describes the options available on the **OCT Path** page. This page allows you to configure the DQ and DQS OCT paths. For more information about the DQ and DQS OCT paths, refer to "DQ/DQS OCT Path" on page 4–14.

Table 3–9. Options on OCT Path Page

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Dynamic Termination Control Options **Enable Dynamic Delay-chain1**	—	USE_OCT_DELAY_ CHAIN1	Enables *<IO>*_OCT_DELAY_CHAIN1 (D5) on both the DQ and DQS dynamic OCT paths. The external memory interfaces synchronize the timing of the turning on and off of the parallel termination during reads and writes from both the DQ and DQS pins, and to improve overall timing margins. D5 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.
Dynamic Termination Control Options **Enable Dynamic Delay-chain2**	—	USE_OCT_DELAY_ CHAIN2	Enables *<IO>*_OCT_DELAY_CHAIN2 (D6) on both the DQ and DQS dynamic OCT paths. The external memory interfaces synchronize the timing of turning on and off of the parallel termination during reads and writes from both the DQ and DQS pins, and to improve overall timing margins. D6 is a run-time adjustable delay chain. For more information about configuring delay chains dynamically, refer to "Delay Chains" on page 4–15.
OCT register mode	**Not used**, **FF**, or **DDIO**	OCT_REG_MODE	Enables the full-rate dynamic OCT registers (*<IO>*_OCT_FF or *<IO>*_OCT_DDIO registers) on both the DQ and DQS dynamic OCT paths. Select **FF** if you want flip-flop registers or **DDIO** if you want double data rate I/O registers. For more component information about this block, refer to the "Dynamic On-Chip Termination Control" section in the *External Memory Interface* chapter of the respective device handbooks.

Table 3–10 describes the options available on the **DQSn I/O** page. This page allows you to configure the DQS and DQSn I/O pins for the ALTDQ_DQS instance. These options are used for memory interfaces that need differential or complementary strobes.

Table 3–10. Options on DQS/DQSn I/O Page

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Use DQSn I/O	—	DQS_DQSN_MODE	Enables access to the DQS I/O that is configured as either differential or complementary. Altera recommends that you use differential DQS for DDR3 interfaces to improve signal integrity. If the DQSn I/O is disabled, the value for the DQS_DQSN_MODE parameter is none. When enabled, the value may either **Complementary pair** or **Differential pair**.
DQS and DQSn IO Configuration mode	**Differential pair** or **Complementary pair**	DQS_DQSN_MODE	If you turn on the **Differential pair** option, the DQSn I/O pin is configured in a differential pair along with the DQS I/O pin. This means that the OE and OCT paths are configured for the DQSn I/O pin, which is similar to the DQS I/O pin. The input and output paths are shared with the DQS I/O pin. This mode is used mainly for DDR2 and DDR3 SDRAM, and RLDRAM II applications. If you turn on the **Complementary pair** option, the DQSn I/O pin is configured in a complementary pair along with the DQS I/O pin. In this mode, the DQSn I/O pin is configured similarly to the DQS I/O pin. This mode is used mainly for QDR/QDR II applications.

Table 3–11 describes the options available on the **Reset/Config Ports** page. For more information about reset and config ports, refer to "ALTDQ_DQS Megafunction Ports" on page 4–33.

Table 3–11. Options on Reset/Config Ports Page (Part 1 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Reset ports **Create 'dqs_areset' input port**	—	—	Enables the asynchronous reset port that asynchronously resets all registers in the DQS output or DQS OE path.
Reset ports **Create 'dqs_sreset' input port**	—	—	Enables synchronous reset port that synchronously resets all registers in the DQS output or DQS OE path.
Reset ports **Create 'input_dq_areset' input port**	—	—	Enables asynchronous reset port that asynchronously resets all registers in the DQ input path.

Table 3–11. Options on Reset/Config Ports Page (Part 2 of 2)

Parameter Name	Legal Value	Clear Box Parameter Name	Description
Reset ports **Create 'input_dq_sreset' input port**	—	—	Enables synchronous reset port that synchronously resets all registers in the DQ input path.
Reset ports **Create 'output_dq_areset' input port**	—	—	Enables asynchronous reset port that asynchronously resets all registers in the DQ output or DQ OE path.
Reset ports **Create 'output_dq_sreset' input port**	—	—	Enables synchronous reset port synchronously resets all registers in the DQ output or DQ OE path.
Reset ports **Create 'bidir_dq_areset' input port**	—	—	Enables asynchronous reset port that asynchronously resets all registers in the bidirectional DQ I/O path.
Reset ports **Create 'bidir_dq_sreset' input port**	—	—	Enables synchronous reset port that synchronously resets all registers in the bidirectional DQ I/O path.
Config ports **Create 'config_clk' input port**	—	—	Enables input clock port that feeds `IO_CONFIG` block for user-driven dynamic delay chain. This input port is used as the clock signal of the shift register block. The maximum frequency for this clock is 30 MHz.
Config ports **Create 'config_datain' input port**	—	—	Enables input port that feeds the input data to the serial load shift register in `IO_CONFIG` block for user-driven dynamic delay chain.
Config ports **Create 'config_update' input port**	—	—	Enables input port that feeds `IO_CONFIG` block update port for user-driven dynamic delay chain. When asserted, the serial load shift register bits feed the parallel load register.

The **Simulation Model** page allows you to optionally generate simulation model files. The **Summary** page displays a list of the types of files to be generated. The automatically generated variation file contains wrapper code in the language you specified earlier. On this page, you can specify additional types of files to be generated. Choose from the AHDL Include file (<*function name*>**.inc**), VHDL component declaration file, <*function name*>**.cmp**), Quartus II symbol file (<*function name*>**.bsf**), Instantiation template file (<*function name*>**.v**), and Verilog HDL black box file (<*function name*>**_bb.v**). If you select **Generate netlist** on the **Simulation Model** page, the file for that netlist is also available. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates generation of an optional file.

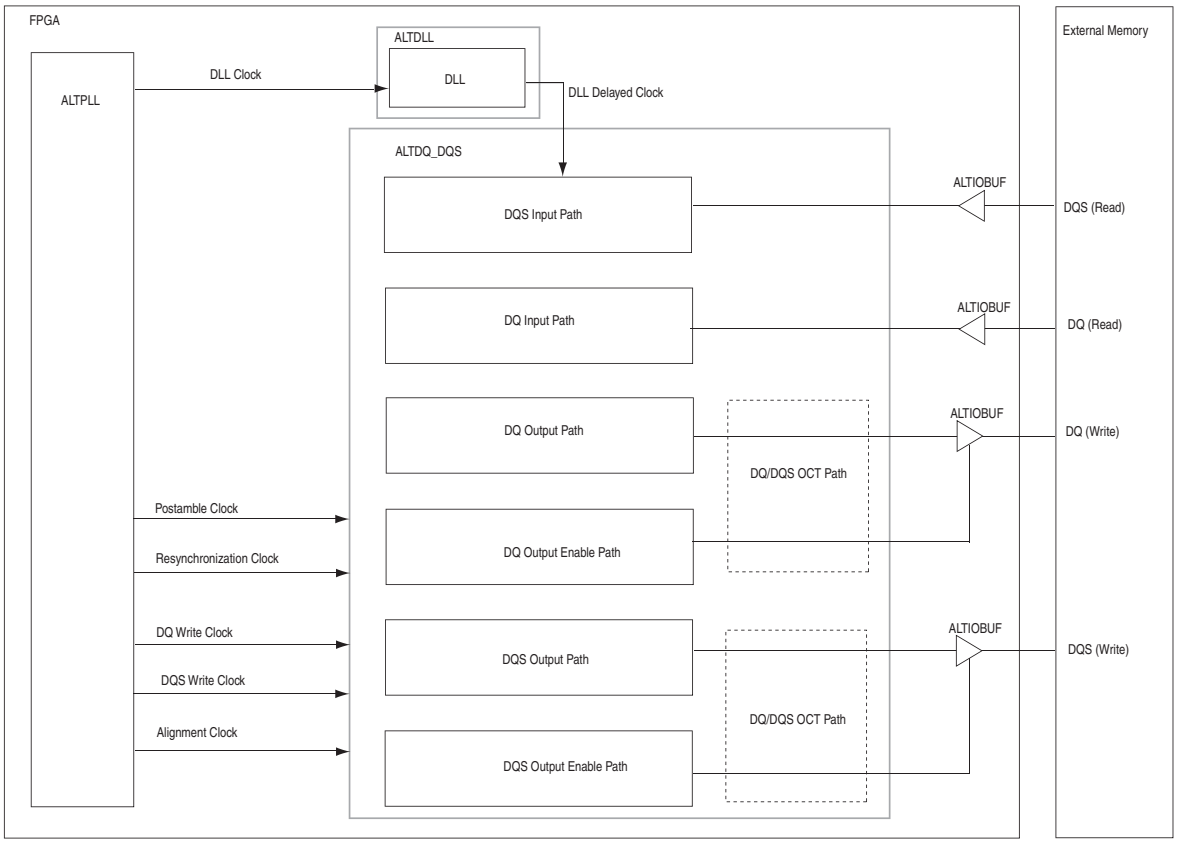
This section describes the functionality of the various blocks and ports in the ALTDLL and ALTDQ_DQS megafunctions. This section also describes the use of delay chains to achieve better timing margins. This section also includes an implementation example showing these megafunctions in a custom external memory interface.

Custom External Memory Interface Datapaths Overview

This section describes the functionality of the various blocks in the external memory datapaths that the megafunctions control.

Figure 4–1 shows the mapping of the ALTDLL and ALTDQ_DQS megafunctions to the dedicated I/O element for external memory interfaces in Stratix IV devices.

Figure 4–1. Mapping of ALTDLL and ALTDQ_DQS Megafunctions to the Dedicated I/O Circuitry



The following blocks are not available in Arria II GX devices:

- Dynamic OCT blocks
- Half-rate blocks
- I/O and DQS configuration blocks

For more information about the blocks available in the datapaths for your target device family, refer to the following chapters in the device handbook:

- *External Memory Interfaces in HardCopy III Devices* chapter in volume 1 of the *HardCopy III Device Handbook*
- *External Memory Interfaces in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*
- *External Memory Interfaces in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*
- *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*
- *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*

The DQ/DQS read and write signals in Figure 4–1 may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

Table 4–2 lists the megafunction blocks in Figure 4–1:

Table 4–1. Megafunction Blocks

Megafunction Block	Description
ALTDLL	The ALTDLL megafunction controls the DLL and DLL offset blocks. For more information about the DLL blocks, refer to "ALTDLL Megafunction" on page 4–3.
ALTDQ_DQS	The ALTDQ_DQS megafunction controls the following memory interface datapaths: ■ DQS Input Path ■ DQ Input Path ■ DQ Output/OE Path ■ DQS Output/OE Path ■ DQ/DQS OCT Path For more information about the datapaths, refer to "ALTDQ_DQS Megafunction" on page 4–4.
ALTPLL	The ALTPLL megafunction block provides the clocking scheme used in the custom external memory interface for half-rate or full-rate interface. For more information about using PLLs, refer to the ALTPLL Megafunction User Guide.
ALTIOBUF	The ALTIOBUF megafunction provides I/O buffer variations to connect the ALTDQ_DQS instance to the FPGA pins and to support dynamic OCT feature.

ALTDLL Megafunction

This section describes the DLL block and the DLL offset control blocks associated with the ALTDLL megafunction.

DLL block and DLL offset control block

The ALTDLL megafunction controls the DLL and its two associated phase-offset control blocks. The ALTDLL megafunction also controls the delay-chain settings to achieve a compensated delay for PVT. For example, a DQS read strobe/clock that is edge-aligned to its associated read data can be used to clock the data into I/O registers if the data is delayed before reaching the register.

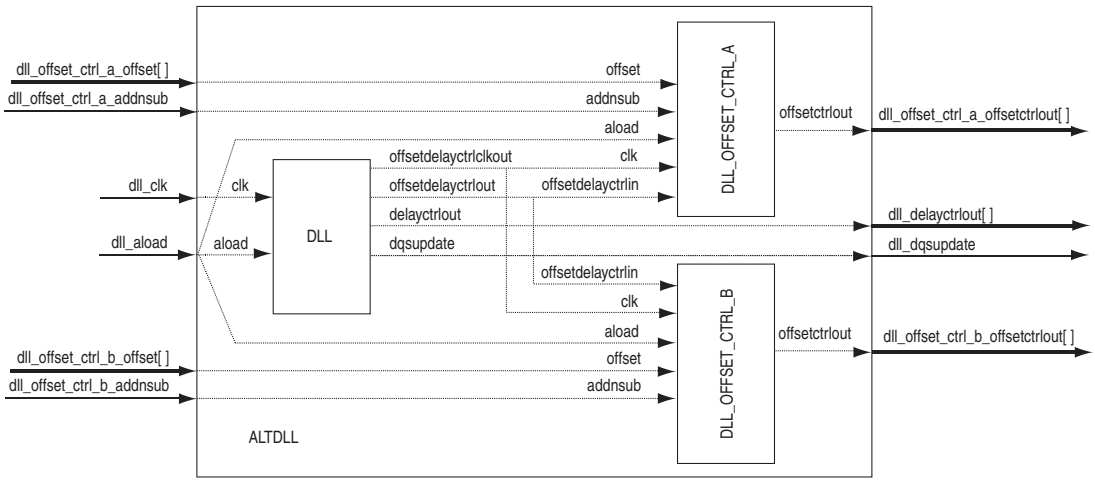
The DLL consists of two phase-offset control blocks— one for each edge adjacent to the DLL, which resides in the corner of the device. Both phase-offset control blocks cannot feed the same edge.

The DLL block computes the necessary delay settings by comparing the period of an input reference clock to the delay through an internal delay chain. You can then use the DLL offset control block to fine-tune the delay setting.

At a minimum, the DLL has a single input that is connected to a dedicated PLL output or input pin, and six gray-coded outputs that are connected to the DQS delay chain block, which is part of the ALTDQ_DQS megafunction.

Figure 4–2 shows the components of the ALTDLL megafunction. The `DLL_OFFSET_CTRL_A` block is the first phase-offset control block, and the `DLL_OFFSET_CTRL_B` block is the second phase-offset control block. These two phase-offset control blocks are connected together to form the ALTDLL megafunction. Each offset control block can only control the DQS delay chains on one edge of the device. To feed the same offset to the DQS delay chains on two edges, you must use both phase-offset control blocks.

Figure 4–2. ALTDLL Megafunction



The names DLL_OFFSET_CTRL_A and DLL_OFFSET_CTRL_B are logical and do not denote the placement of the actual phase-offset blocks. With location assignments, you can assign these blocks to the top, bottom, or side of the FPGA, depending on which DLL your design uses. If location assignments are not used, the Quartus II Fitter places these blocks on the top, bottom, or side of the FPGA device.

The DLL and DLL offset blocks in the DQS phase shift circuitry generate the control signals to shift the DQS delay chain delays to center align the DQS strobe with the incoming DQ data at the IOE registers. This is common when reading from external memory interfaces. For more information about the DLL offset control blocks in the DQS phase shift circuitry, refer to the *DQS Phase Shift Circuitry* section in the respective device handbooks.

For more information about the ALTDLL megafunction ports, refer to "ALTDLL Megafunction Ports" on page 4–31.

ALTDQ_DQS Megafunction

This section describes the DQ/DQS datapaths and the associated blocks of the ALTDQ_DQS megafunction. The figures in the subsequent sections show the megafunction blocks used to construct the datapath and their connections of the top-level ports with the blocks that configure the paths. You must set the appropriate parameters using the parameter editor to enable the blocks and the desired configurations in the paths.

Table 4–2 list the common blocks that are used in the DQ/DQS input and output paths:

The value for <*IO*> depends on your selection in the parameter editor. The possible values are BIDIR_DQ and INPUT_DQ.

Table 4–2. Common Blocks in the DQ/DQS Input and Output Paths (Part 1 of 2)

Block	Name	Description
DQS_DELAY_CHAIN DQS_INPUT_DELAY_CHAIN (D1) DQSBUSOUT_DELAY_CHAIN (Da) DQS_ENABLE_DELAY_CHAIN (Db) <*IO*>_INPUT_DELAY_CHAIN (D1) <*IO*>_OUTPUT_DELAY_CHAIN1 (D5) <*IO*>_OUTPUT_DELAY_CHAIN2 (D6) DQS_OUTPUT_DELAY_CHAIN1 (D5) DQS_OUTPUT_DELAY_CHAIN2 (D6) <*IO*>_OE_DELAY_CHAIN1 (D5) <*IO*>_OE_DELAY_CHAIN2 (D6) DQS_OE_DELAY_CHAIN1 (D5) DQS_OE_DELAY_CHAIN2 (D6) <*IO*>_OCT_DELAY_CHAIN1 (D5 OCT) <*IO*>_OCT_DELAY_CHAIN2 (D6 OCT)	Delay Chains	Represents the delay chains used to delay signals. For more information about the DQS delay chain block, refer to the *DQS Delay Chain* section of the respective device handbooks. For more information about the delay chain types and settings, refer to "Delay Chains" on page 4–15.

Table 4–2. Common Blocks in the DQ/DQS Input and Output Paths (Part 2 of 2)

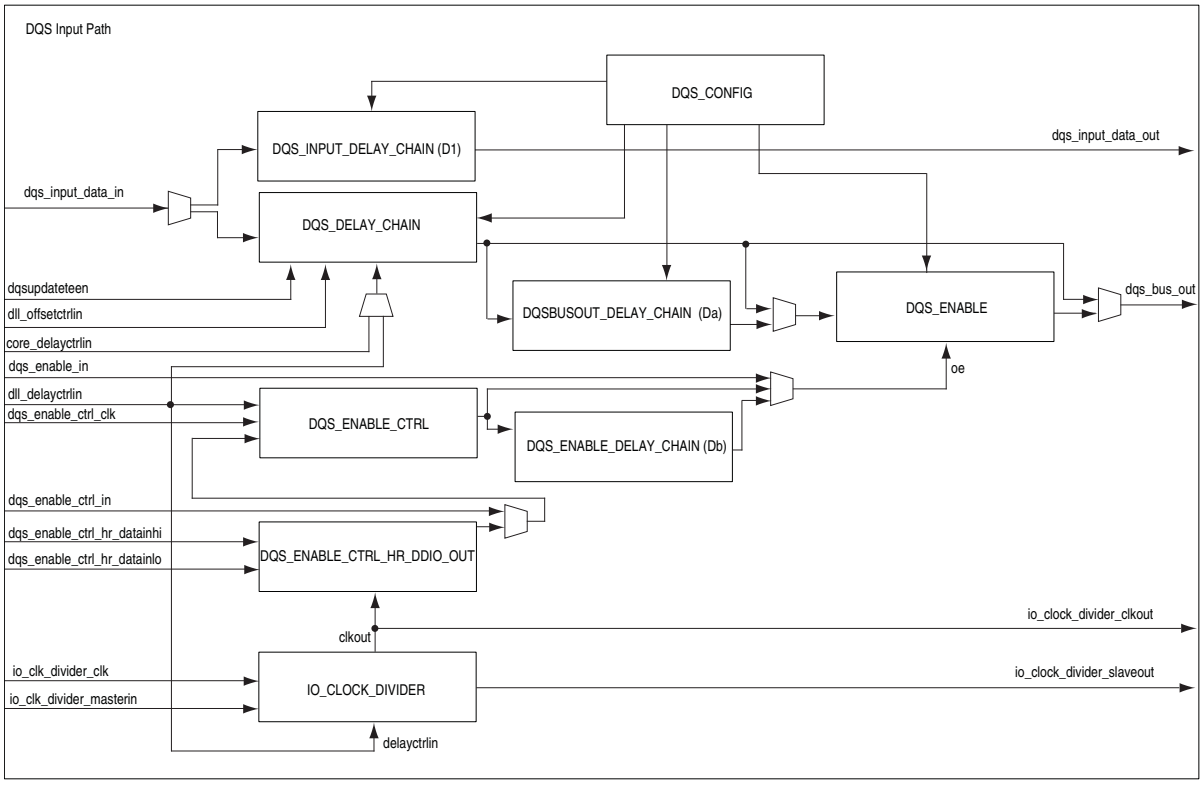
Block	Name	Description
DQS_CONFIG	DQS Configuration Block	A shift register that dynamically changes the settings of various device configuration bits. The shift registers power up low. The IO_CONFIG block is used to configure the settings for all I/O pins. The IO_CONFIG block cannot configure the dynamic delay chains on the OCT path or on the DQS input path (D2, D3_0, D3_1, D4,D5 OCT, and D6 OCT) that are controlled by the DQS_CONFIG block. The DQS_CONFIG block is used to configure the settings of the DQ/DQS I/O pins. Note that these blocks are only available for Stratix III and Stratix IV devices. For more information about the DQS_CONFIG/IO_CONFIG blocks, refer to "DQS_CONFIG / IO_CONFIG Block" on page 4–22.
IO_CONFIG	I/O Configuration Block	
IO_CLOCK_DIVIDER	I/O Clock Divider Block	Represents a divide-by-2 clock divider for transferring data to the core at one half the speed of the I/O input or output clock. Each divider feeds up to six pins (a ×4 DQS group) in the device. To feed wider DQS groups, you need to chain multiple clock dividers together by feeding the slaveout output of one divider to the masterin input of the neighboring pins' divider. The IO_CLOCK_DIVIDER block is used in the DQ and DQS input paths when you enable the **Use half-rate components** option in the parameter editor. Note that this block is only available for Stratix III and Stratix IV devices. For more information about this block, refer to the *I/O Element (IOE) Registers* section in the *External Memory Interfaces* chapter of the respective device handbooks.

DQS Input Path

This path receives the DQS strobe signal from the external memory during read operations.

Figure 4–3 shows the available blocks in the DQS input path.

Figure 4–3. DQS Input Path *(Note 1)*, *(2)*



Notes to Figure 4–3:

(1) The `dqs_input_data_in` port must be connected to the output port of the input buffer.

(2) The `dll_offsetctrlin`, `dll_delayctrlin`, and `dqsupdateen` ports must be connected to the DLL.

The DQS input path consists of the following blocks:

Table 4–3. DQS Input Path

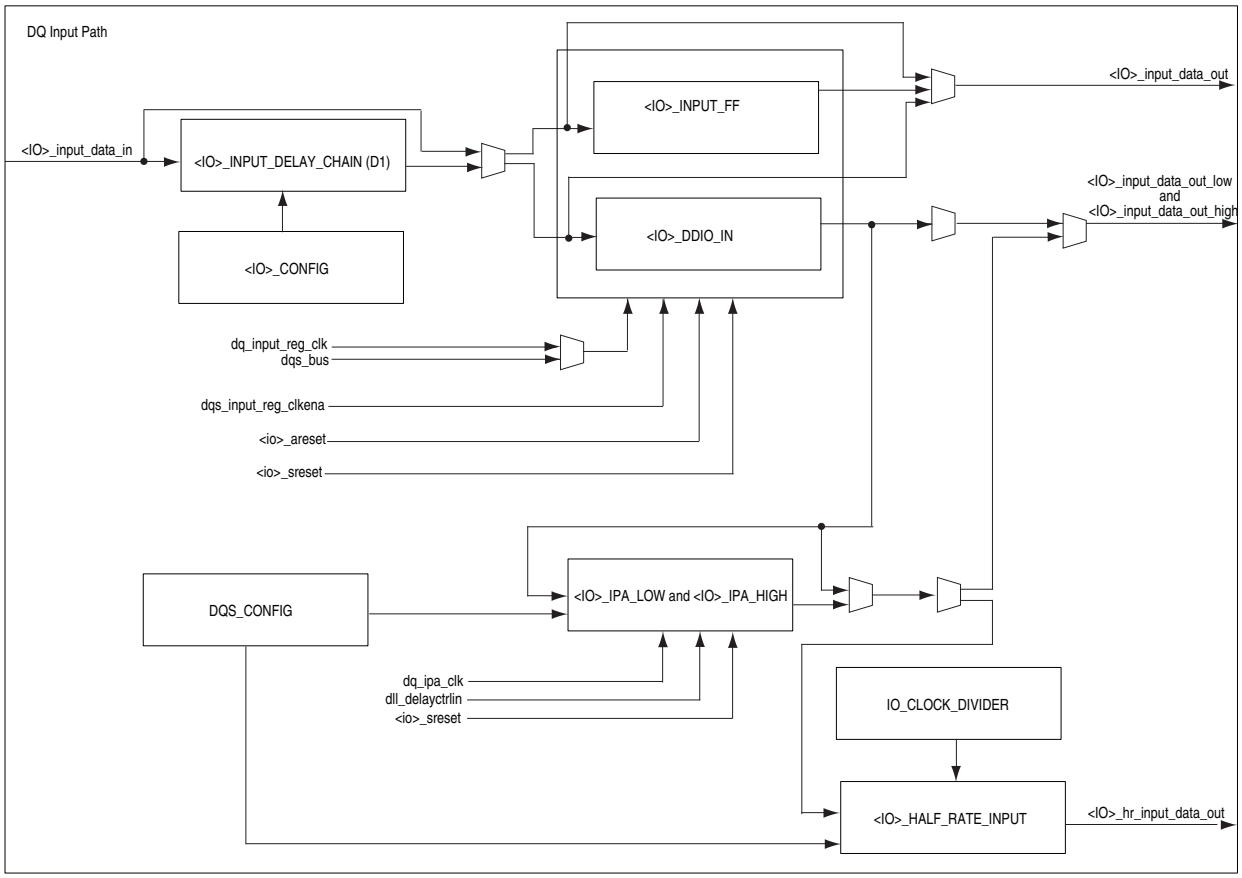
Block	Name	Description
DQS_ENABLE_CTRL	DQS Enable Control Block	Represents the circuitry to control the DQS enable block. Each DQS enable block can be controlled by a DQS enable control block. For more information about the DQS enable control, refer to the *DQS Postamble Circuitry* section in the *External Memory Interface* chapter of the respective device handbooks.
DQS_ENABLE_CTRL_HR_DDIO_OUT	DQS Enable Control Half Rate Block	Represents the circuitry to transfer input to the DQS_ENABLE_CTRL block from a half-rate clock to a full-rate clock.
DQS_ENABLE	DQS Enable Block	Represents the AND-gate control on the DQS input used to ground the DQS input strobe when the strobe goes to Z after a DDR read postamble. The DQS_ENABLE block enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change. For more information about the DQS enable block, refer to the *Update Enable Circuitry* section in the *External Memory Interfaces* chapter of the respective device handbooks.
DQS_DELAY_CHAIN	DQS Delay Chain Block	For more information about these delay chains, refer to Table 4–2 on page 4–4.
DQSBUSOUT_DELAY_CHAIN	DQS Busout Delay Chain	
DQS_ENABLE_DELAY_CHAIN	DQS Enable Delay Chain	
DQS_CONFIG	DQS Configuration Blocks	For more information about DQS_CONFIG block, refer to Table 4–2 on page 4–4.
IO_CLOCK_DIVIDER	I/O Clock Divider Block	For more information about I/O clock divider block, refer to Table 4–2 on page 4–4.

DQ Input Path

This path receives the DQ signal from the external memory during read operations. Instantiate this path for all input-only and bidirectional DQ I/O pins. Figure 4–4 shows the available blocks in the DQ input path and the connections with the ALTDQ_DQS ports.

The value for <*IO*> depends on your selection in the parameter editor. The possible values are `BIDIR_DQ` and `INPUT_DQ`.

Figure 4–4. DQ Input Path *(Note 1)*, *(2)*, *(3)*



Notes to Figure 4–4:

(1) The <*IO*>`_input_data_in` port must be connected to the output port of the input buffer.
(2) The `dll_delayctrlin` port must be connected to the DLL.
(3) The `IO_CLOCK_DIVIDER`, <*IO*>`_HALF_RATE_INPUT`, <*IO*>`_IPA_LOW`, and <*IO*>`_IPA_HIGH` blocks are half-rate components.

The DQ input path consists of the following blocks:

Table 4–4. DQ Input Path

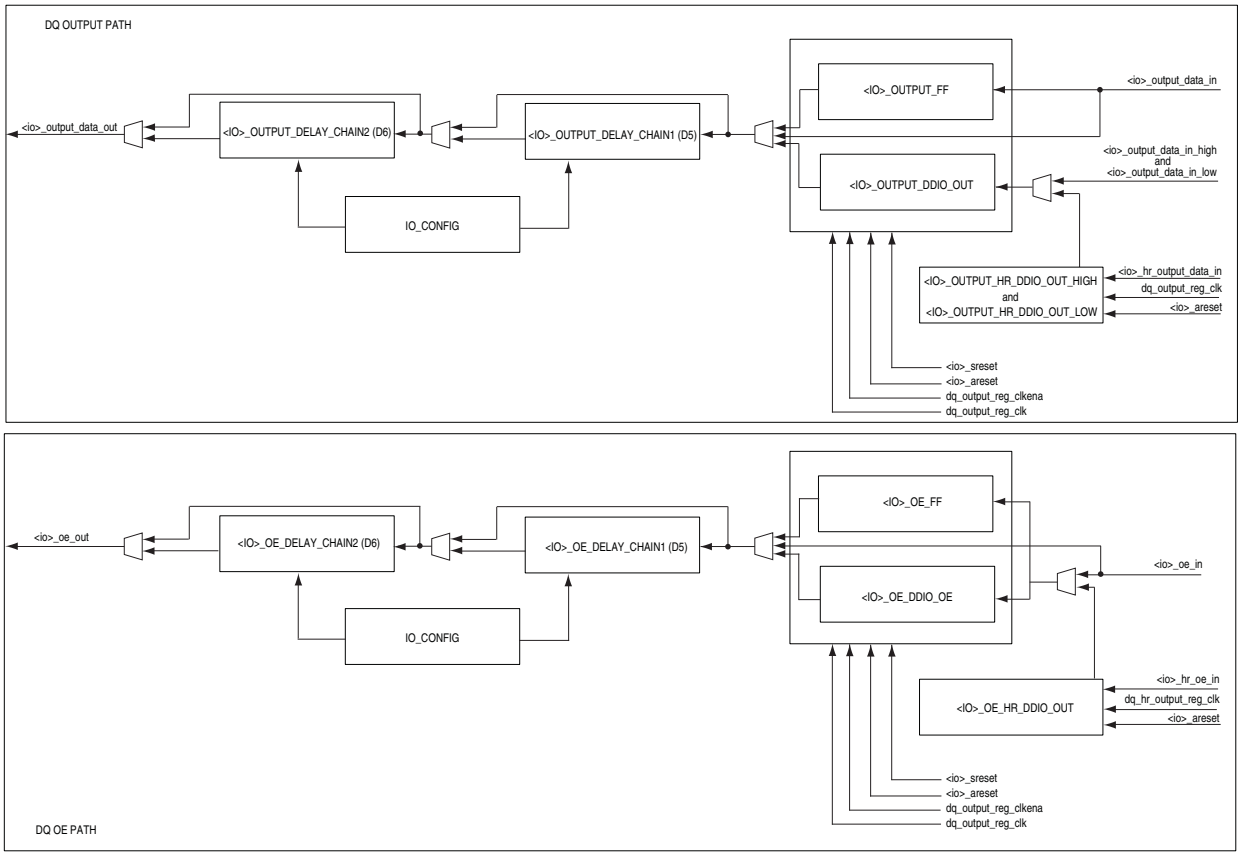
Block	Name	Description
<*IO*>_INPUT_FF <*IO*>_DDIO_IN	DQ Input register blocks	Samples the DQ signal during a read operation. These blocks are clocked by the core or by a clock pin. The <*IO*>_INPUT_FF block represents a group of flip-flops registers in the DQ input path. The <*IO*>_DDIO_IN represents a group of double data rate input registers in the DQ input path.
<*IO*>_IPA_LOW and <*IO*>_IPA_HIGH	Input Phase Alignment (IPA) Block	Represents the circuitry required to phase shift the input signal. This is primarily used to match the arrival delay of the DQS (triggered by the fly-by clock on a DDR3-DIMM) to the latest arrival delay of a DQS from the DIMM. The input phase alignment block levels or aligns the DQ group signals in the core using different phase shifts. For more information about input phase alignment, refer to the *Leveling Circuitry* section in the *External Memory Interface* chapter of the respective device handbooks.
<*IO*>_HALF_RATE_INPUT	Half-rate input registers block	Represents the circuitry required to transfer the input signal from a full-rate clock to a half-rate clock. Note that this block is only available in Stratix III and Stratix IV devices.
INPUT_DELAY_CHAIN	Input Delay Chain	For more information about the input delay chain, refer to Table 4–2 on page 4–4.
DQS_CONFIG	DQS Configuration Block	For more information about the DQS_CONFIG block, refer to Table 4–2 on page 4–4.
IO_CONFIG	I/O/ Configuration Block	For more information about the IO_CONFIG block, refer to Table 4–2 on page 4–4.
IO_CLOCK_DIVIDER	I/O Clock Divider Block	For more information about I/O clock divider block, refer to Table 4–2 on page 4–4.

DQ Output/OE Path

This path sends the DQ signal to the external memory for writing operations. Figure 4–5 shows the available blocks in the DQ Output and OE path and the connections with the ALTDQ_DQS ports.

The value for <*IO*> depends on your selection in the parameter editor. The possible values are `BIDIR_DQ` and `OUTPUT_DQ`.

Figure 4–5. DQ Output and OE Path *(Note 1)*, *(2)*, *(3)*



Notes to Figure 4–5:

(1) The *<IO>*`_output_data_out` port must be connected to the input port of the output buffer.

(2) The *<IO>*`_oe_out` port must be connected to the output enable port of the output buffer.

(3) The *<IO>*`_OE_HR_DDIO_OUT`, *<IO>*`_OUTPUT_HR_DDIO_OUT_HIGH` and *<IO>*`_OUTPUT_HR_DDIO_OUT_LOW` blocks are half-rate components.

The DQ output and OE path consist of the following blocks:

Table 4–5. DQ Output and OE Path

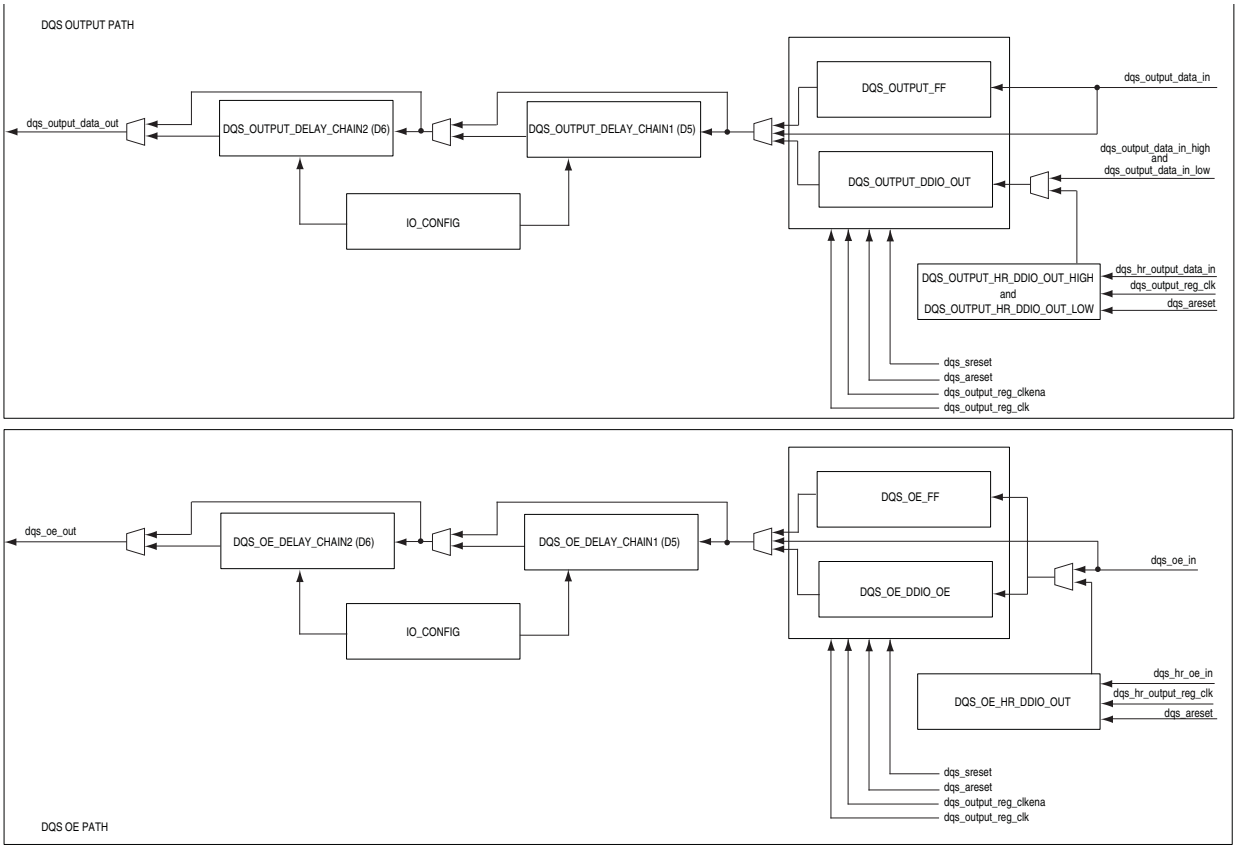
Block	Name	Description
<*IO*>_OUTPUT_FF	DQ output register blocks	Sends data directly to the external memory DQ pins during a write operation through the output buffer. These blocks are clocked by the DQ write clock. The <*IO*>_OUTPUT_FF block represents a group of flip-flop registers in the DQ output path. The <*IO*>_OUTPUT_DDIO_OUT represents a group of double data rate output registers in the DQ output path.
<*IO*>_OUTPUT_DDIO_OUT		
<*IO*>_OE_FF	DQ output enable register blocks	Sends output enable signal to the output buffer. These blocks are clocked by the DQ write clock. The <*IO*>_OE_FF block represents a group of flip-flop registers in the DQ OE path. The <*IO*>_OE_DDIO_OE represents a group of double data rate registers in the DQ OE path.
<*IO*>_OE_DDIO_OE		
<*IO*>_OUTPUT_HR_DDIO_OUT_HIGH and <*IO*>_OUTPUT_HR_DDIO_OUT_LOW	Half-rate output register block	Represents the DDIO registers that are used to transfer DQ signals from the core during half-rate write operation. These blocks are clocked by the DQ write clock.
<*IO*>_OE_HR_DDIO_OUT	Half-rate output enable register block	Represents the DDIO registers that are used to transfer half-rate DQ output enable signals to the output buffer.
<*IO*>_OUTPUT_DELAY_CHAIN1(D5)	DQ output delay chains	For more information about the DQ output and OE delay chains, refer to Table 4–2 on page 4–4.
<*IO*>_OUTPUT_DELAY_CHAIN2(D6)		
<*IO*>_OE_DELAY_CHAIN1 (D5)	DQ OE delay chains	
<*IO*>_OE_DELAY_CHAIN2 (D6)		
IO_CONFIG	I/O Configuration Block	For more information about the IO_CONFIG block, refer to Table 4–2 on page 4–4.

DQS Output/OE Path

This path sends the DQS strobe signal to the external memory for writing operations.

Figure 4–6 shows the available blocks in the DQS output and OE path and the connections with the ALTDQ_DQS ports.

Figure 4–6. DQS Output and OE Path *(Note 1)*, *(2)*, *(3)*



Notes to Figure 4–6:

(1) The `dqs_output_data_out` port must be connected to the input port of the output buffer.

(2) The `dqs_oe_out` port must be connected to the output enable port of the output buffer.

(3) The `DQS_OE_HR_DDIO_OUT`, `DQS_OUTPUT_HR_DDIO_OUT_HIGH` and `DQS_OUTPUT_HR_DDIO_OUT_LOW` blocks are half-rate components.

The DQS output and OE path consist of the following blocks:

Table 4–6. DQS Output and OE Path

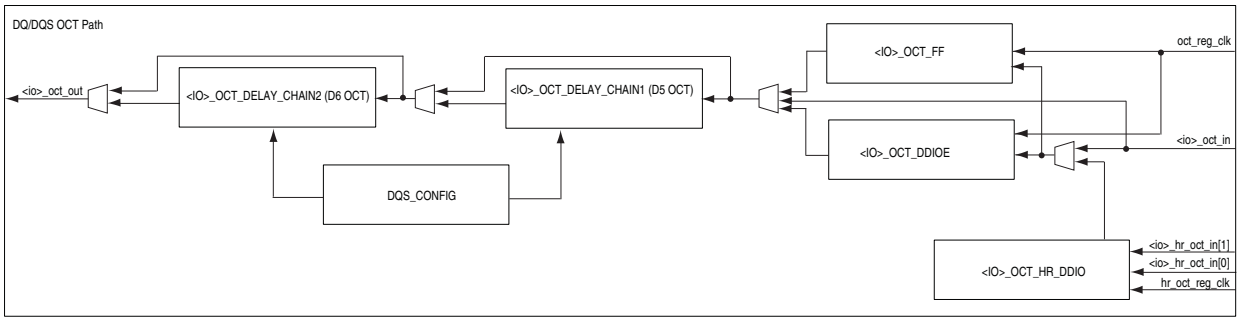
Block	Name	Description
DQS_OUTPUT_FF	DQS output register blocks	Sends data directly to the external memory DQs pins during a write operation through the output buffer. These blocks are clocked by the DQS write clock. The DQS_OUTPUT_FF block represents a group of flip-flop registers in the DQS output path. The DQS_OUTPUT_DDIO_OUT represents a group of double data rate output registers in the DQS output path.
DQS_OUTPUT_DDIO_OUT		
DQS_OE_FF	DQS output enable register blocks	Sends output enable signal to the output buffer. These blocks are clocked by the DQS write clock. The DQS_OE_FF block represents a group of flip-flop registers in the DQS OE path. The DQS_OE_DDIO_OE represents a group of double data rate registers in the DQS OE path.
DQS_OE_DDIO_OE		
DQS_OUTPUT_HR_DDIO_OUT_HIGH and DQS_OUTPUT_HR_DDIO_OUT_LOW	Half-rate output register block	Represents the DDIO registers that are used to transfer DQS signals from the core during half-rate write operation. These blocks are clocked by the DQS write clock.
DQS_OE_HR_DDIO_OUT	Half-rate output enable register block	Represents the DDIO registers that are used to transfer half-rate DQS output enable signals to the output buffer.
DQS_OUTPUT_DELAY_CHAIN1(D5)	DQS output delay chains	For more information about the DQS output and OE delay chains, refer to Table 4–2 on page 4–4.
DQS_OUTPUT_DELAY_CHAIN2(D6)		
DQS_OE_DELAY_CHAIN1 (D5)	DQS OE delay chains	
DQS_OE_DELAY_CHAIN2 (D6)		
IO_CONFIG	I/O Configuration Block	For more information about the IO_CONFIG block, refer to Table 4–2 on page 4–4.

DQ/DQS OCT Path

Figure 4–7 shows the available blocks in the DQ/DQS OCT paths and the connections with the ALTDQ_DQS ports. Use this path to utilize OCT capabilities at the DQ and DQS output paths.

The <*IO*> value depends on your selection in the parameter editor. The possible values are `DQS`, `DQSn`, `BIDIR_DQ`, and `OUTPUT_DQ`.

Figure 4–7. DQ/DQS OCT Path *(Note)*



Notes to Figure 4–7:

(1) The <*IO*>`_oct_out` port must be connected to the input port of the output buffer.
(2) The <*IO*>`_OCT_HR_DDIO` block is a half-rate component.

The DQ/DQS OCT path consists of the following blocks:

Table 4–7. DQ/DQS OCT Path

Block	Name	Description
<*IO*>`_OCT_FF`	OCT register blocks	The <*IO*>`_OCT_FF` block represents a group of flip-flop registers in the DQ/DQS OCT output path.
<*IO*>`_OCT_DDIOE`		The <*IO*>`_OCT_DDIOE` represents a group of DDIO registers in the DQ/DQS OCT output path.
<*IO*>`_OCT_HR_DDIO`	Half -rate OCT block	Represents a group of DDIO registers required to transfer the calibrated output signal in half-rate mode.
<*IO*>`_OCT_DELAY_CHAIN1` (D5 OCT)	OCT delay chain blocks	For more information about the OCT output delay chain blocks, refer to Table 4–2 on page 4–4
<*IO*>`_OCT_DELAY_CHAIN2` (D6 OCT)		
`DQS_CONFIG`	DQS Configuration Block	For more information about the `DQS_CONFIG` block, refer to Table 4–2 on page 4–4.

For more information about using the dynamic calibration blocks for termination, refer to *Dynamic Calibrated On-Chip Termination (ALTOCT) Megafunction User Guide*. For more information about implementing calibrated dynamic OCT, refer to *AN 465: Implementing OCT Calibration in Stratix III Devices*.

Delay Chains

The ALTDQ_DQS megafunction uses various types of delay chains. You can control delay chains dynamically to provide a better sampling window for external memory interfaces.

Table 4–8 shows the delay chain type and their respective settings.

Table 4–8. Delay Elements and Settings

Delay Chain Type	Function	Possible Settings	Step Value (ps)	Maximum Delay Value (ps)
D1	Tunes the DQ delay (read calibration) in DDR applications.	There are 16 possible settings for this delay chain because the delay control in the chain is 4 bits wide.	50	0
D5 and D5 OCT	D5 is the output register-to-I/O buffer delay. D5 OCT is the OCT to I/O buffer delay. These delay chains are for write calibration in DDR applications. D5 is cascaded together with D6 to generate the sum of delays.	There are 16 possible settings for this delay chain because the delay control in the chain is 4 bits wide.	50	0
D6 and D6 OCT	D6 is the output register-to-I/O buffer delay. D6 OCT is the OCT to I/O buffer delay. This delay chain is used to reduce simultaneous switching noise (SSN). These delay chains can be adjusted on a group basis for non-DDR3 applications. This delay chain works with a write-leveling clock to adjust the delay among groups for DDR3 applications. D6 is cascaded together with D5 to generate the sum of delays. For more information about reducing SSN, refer to "Deskew Delay Chains" on page 4–16.	There are 8 possible settings for this delay chain because the delay control in the chain is 3 bits wide.	50	0

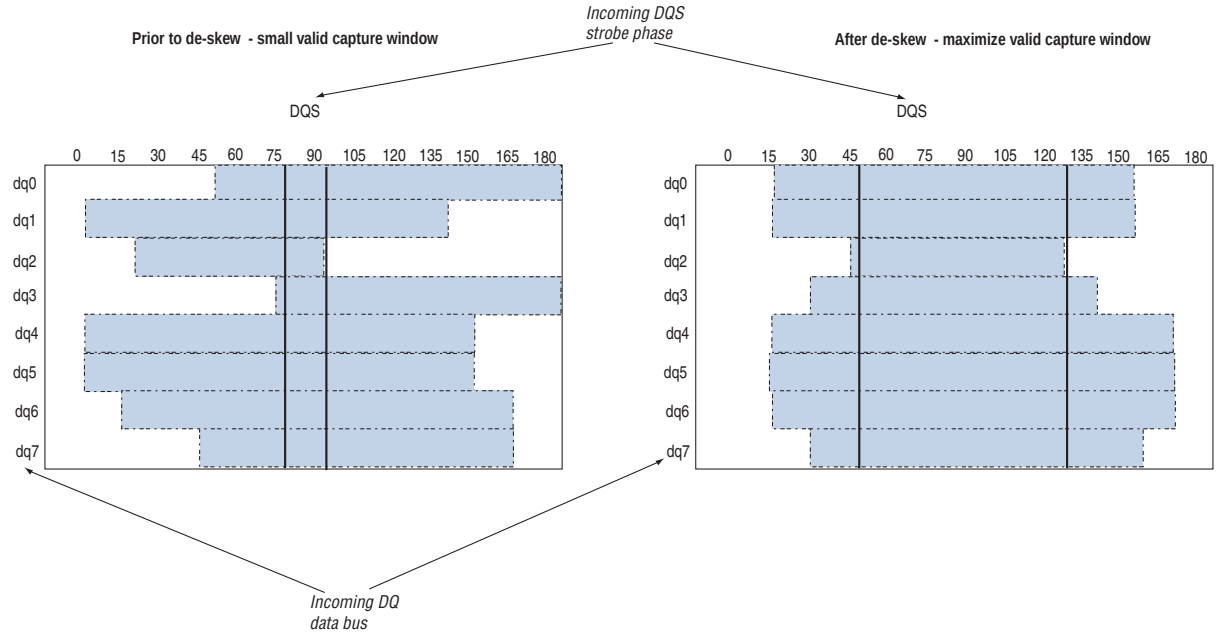
Each step value is either 50 or 400 ps. Setting the number of stages in the delay chain to 10 means 10 × 50 ps = 500 ps of delay.

The minimum delay value factors in only variable delays, but not the intrinsic delay present in the delay chain. For more information about intrinsic delays, refer to the respective Arria II GX, HardCopy III, HardCopy IV, Stratix III, and Stratix IV device handbook or data sheet.

Deskew Delay Chains

The deskew delay chain feature in Stratix III or Stratix IV devices is useful in external memory interfaces, such as DDR or DDR2 external memory interfaces. Refer to Figure 4–8.

Figure 4–8. Deskew Delay Chains



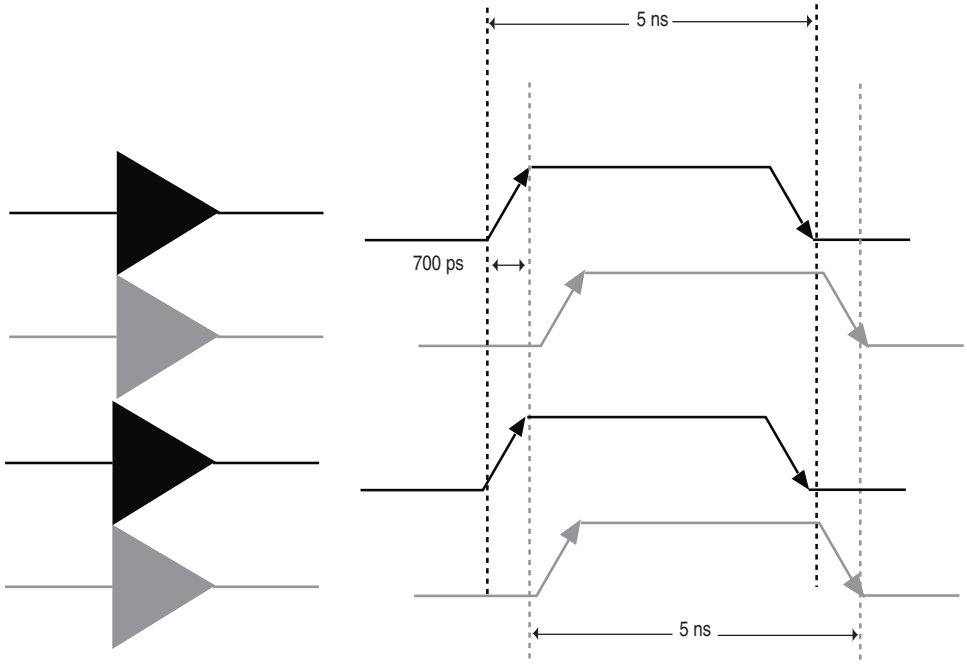
This feature is useful in deskewing the DQ bus for board trace mismatches between the FPGA and external memory interface.

The graph on the left is obtained when no deskew delay chains are used. The capture window is small because of the board trace delays.

The graph on the right is obtained when deskew delay chains are used to deskew the DQ bus appropriately, based on the board trace delays, to maximize the capture window.

The deskew delay chains reduce SSN by delaying the DQ bus by small amounts of delay compared to the period of the signal on adjacent DQ pins. Refer to Figure 4–8.

Figure 4–9. Reduce SSN Using Deskew Delay Chains



The SSN is induced when adjacent pins in a DQ bus that toggle at the same time (especially at a high frequency) induces noise that affects signal integrity. To ensure that the adjacent pins in a DQ bus are not toggled at the same time, deskew delay chains are used to provide small amounts of delay. Refer to Figure 4–9.

You can access these delay chains in the ALTDQ_DQS megafunction for the DQ Input Path (D1) and DQ Output Path (D5 and D6). These 50 ps step delay chains provide small amounts of delay.

You must create a custom calibration circuit to control these delay chains to reduce SSN.

ALTIOBUF Megafunction and Delay Chains Integration

You must instantiate the ALTIOBUF megafunction separately to configure the input buffer block, output buffer block, and differential output buffer block that are used together with the ALTDQ_DQS megafunction. These I/O buffers are used so that the impedance between the system and the external circuitry matches. This implementation maximizes the power transfer and minimizes reflections from the external circuitry.

CAUTION

The ALTIOBUF megafunction must not be used to configure any dynamic delay chains. The ALTIOBUF must only be used to configure the I/O buffers to avoid conflict between the dynamic configuration and delay chain circuitry in the ALTDQ_DQS megafunction.

The dynamic delay chains are controlled by the configuration circuitry encapsulated in the ALTDQ_DQS megafunction. Each instance of the I/O buffer uses the D1, D5, and D6 delay chains. These delay chains are dynamically configured by the `IO_CONFIG` and `DQS_CONFIG` blocks. The `IO_CONFIG` and `DQS_CONFIG` blocks are a shift registers that change the delay settings in the I/O buffers that are connected to the I/O pins and DQ and DQS I/O pins, respectively. The `IO_CONFIG` block cannot configure the dynamic delay chains on the OCT path or the DQS input path because these delay chains are configured by the `DQS_CONFIG` block.

For more information about the `IO_CONFIG` and `DQS_CONFIG` blocks, refer to "DQS_CONFIG / IO_CONFIG Block" on page 4–22.

For more information about input buffer, output buffer, or bidirectional buffer, refer to the *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

Figure 4–10 through Figure 4–17 show the various configurations of the ALTDQ_DQS megafunction when combined with the ALTIOBUF megafunction. These configurations apply to both the DQ and DQS I/O pins. The use of the `datain` and `datout` signals in these figures are generic. These signals represent either data, clock, or strobe in external memory interfaces.

Figure 4–10. Input Only—Single-Ended

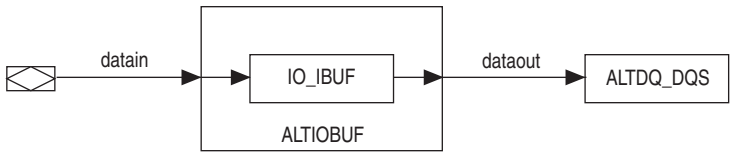


Figure 4–11. Input Only—Differential

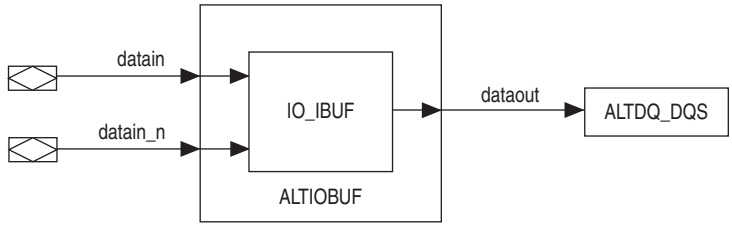


Figure 4–12. Input Only—Complementary

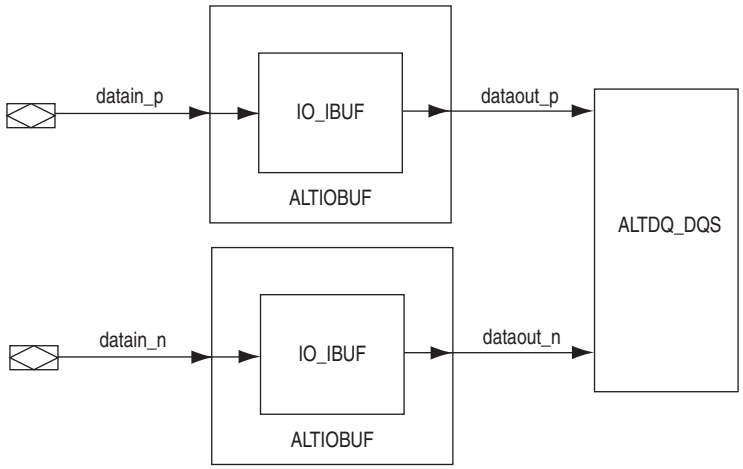
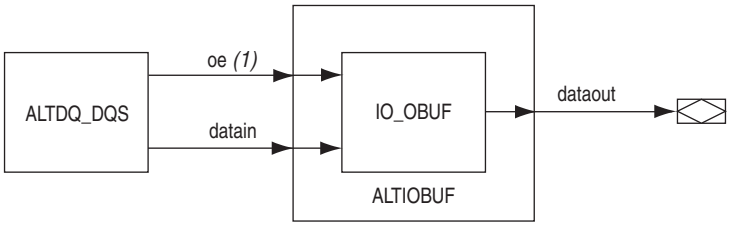


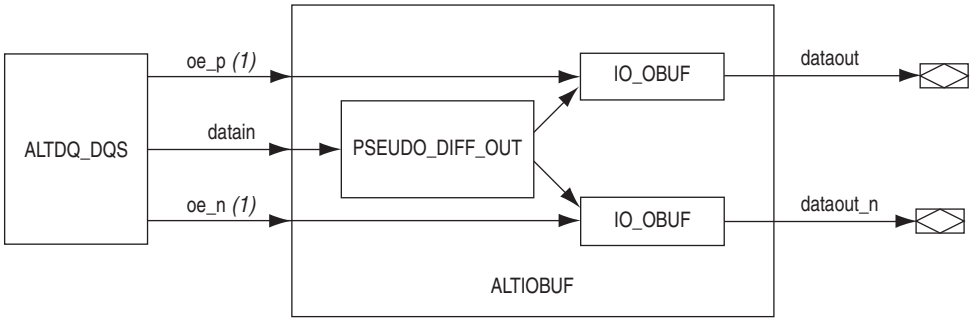
Figure 4–13. Output Only—Single-Ended



Note to Figure 4–13:

(1) The oe port is optional.

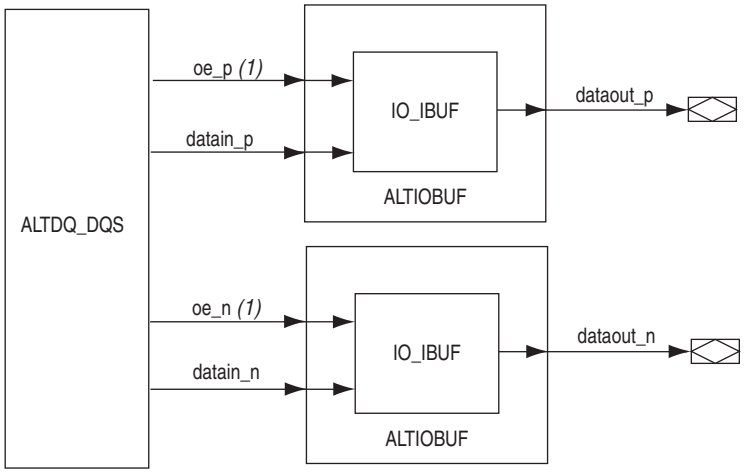
Figure 4–14. Output Only—Differential



Note to Figure 4–14:

(1) The oe_p and oe_n ports are optional.

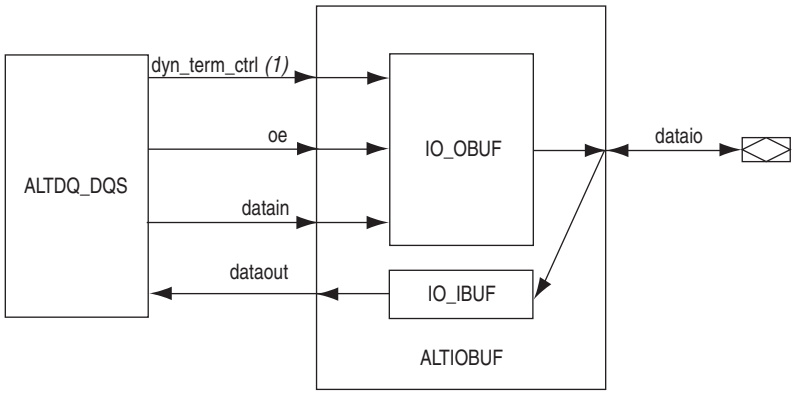
Figure 4–15. Output Only— Complementary



Note to Figure 4–15:

(1) The oe_p and oe_n ports are optional.

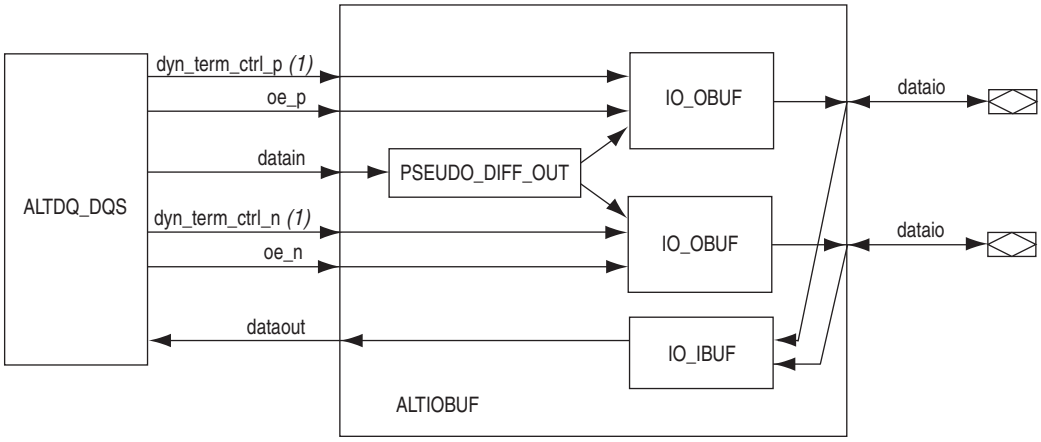
Figure 4–16. Bidirectional— Single-Ended



Note to Figure 4–16:

(1) The dyn_term_ctrl port is optional.

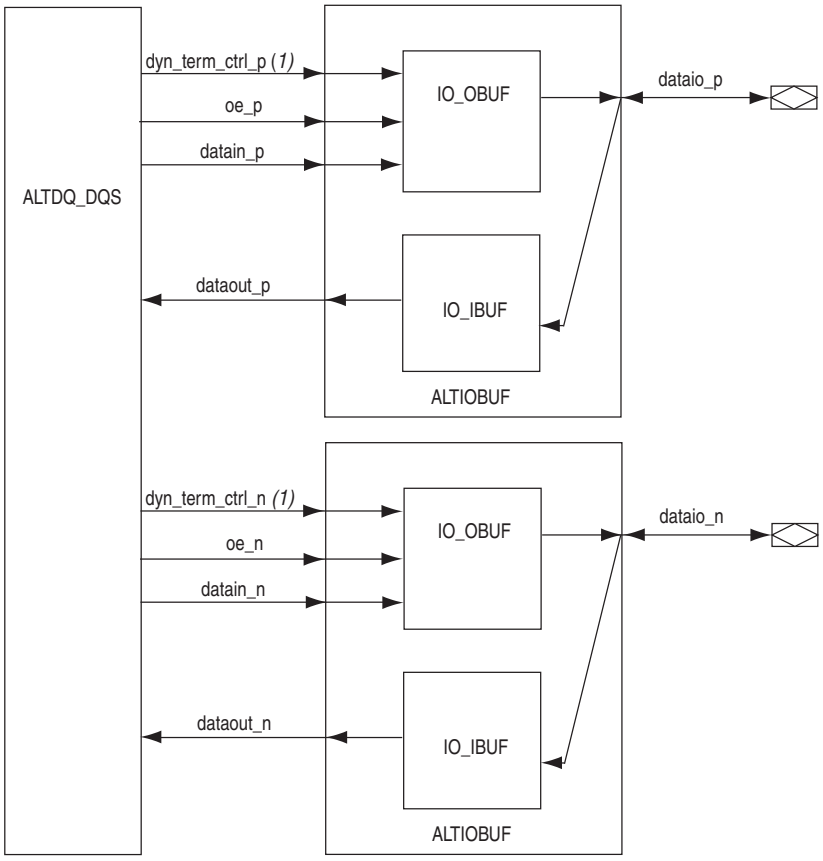
Figure 4–17. Bidirectional— Differential



Note to Figure 4–17:

(1) The dyn_term_ctrl_p and dyn_term_ctrl_n ports are optional.

Figure 4–18. Bidirectional— Complementary



Note to Figure 4–17:

(1) The dyn_term_ctrl_p and dyn_term_ctrl_n ports are optional.

DQS_CONFIG / IO_CONFIG Block

The DQS_CONFIG and IO_CONFIG blocks dynamically change the settings of various configuration bits. One IO_CONFIG block is configured per I/O, whereas one DQS_CONFIG block is configured per x4 group of I/Os (similar to IO_CLOCK_DIVIDERs). These blocks share the datain, clk, and update signals eventhough they have individual enable signals.

When dynamic delay chains are enabled, two key blocks are used together with the I/O buffer block (input buffer, output buffer, or bidirectional buffer), the I/O config block and the delay chain block.

The IO_CONFIG block controls the configuration of the necessary delay settings. The necessary delay settings are set into the respective delay chain block (D1, D5, and D6). These delay settings delay data that passes through the delay chain before going through the I/O buffer block.

The ALTDQ_DQS megafunction allows you to control the delay chain using the following I/O config signals:

- config_datain
- config_clk
- config_update
- <*xxx*>_io_config_ena. <*xxx*> depends on which I/O pin is controlled—input, output, bidirectional, DQS, or DQSn I/O.

For more information about the DQS block or the DQSn I/O block and the sequence of the shift registers, refer to the *I/O Configuration Block and DQS Configuration Block* section in *Chapter 7: External Memory Interfaces in Stratix IV Devices* of the *Stratix IV Devices Handbook*.

For more information about these ports, refer to the "DQS_CONFIG/IO_CONFIG Megafunction Ports" on page 4–45.

Configuring Dynamic Delay Chains Using the IO_CONFIG Block

The IO_CONFIG block serially shifts the value of config_datain only when <*xxx*>_io_config_ena is asserted, during which you shift in the value of config_datain to a shift register. Because a 11-bit shift register is used in the IO_CONFIG block, you must hold <*xxx*>_io_config_ena asserted for 11 configuration clock cycles (config_clk). When the shift registers are fully loaded, the shift register has its bits arranged in correspondence with the values for datain:

- datain values set during the first four configuration clock cycles corresponds to the 4–bit input delay chain values (D1).
- datain values set during the next three configuration clock cycles corresponds to the 3–bit output delay chain values (D6).
- datain values set during the last four configuration clock cycles corresponds to the 4–bit output delay chain values (D5).

In all cases, the most significant bit (MSB) of the delay chain values is shifted in first and the least significant bit (LSB) is shifted in last. For example, in the first four configuration clock cycles, the first configuration clock cycle corresponds to the MSB of the input delay chain value and the fourth configuration clock cycle corresponds to the LSB.

The delay only takes effect when the `config_update` signal is asserted for one configuration clock cycle, in which all the bits in the serial shift register feeds an 11–bit, parallel-loaded register. Right after the signal is deasserted, you can observe the delay from `datain` (of the delay chain primitive) to `dataout` (of the delay chain block).

For all delay chains, each delay setting increment adds approximately 50 ps of delay (the actual value depends on the device speed grade); therefore, the total delay value is equal to the number of stages in the delay chain ×50 ps. For example, if you set the number of stages in the delay chain to five, then the total delay value is five times 50 ps, which is 250 ps.

Figure 4–19 through Figure 4–23 are simulation examples that show the results of varying the delay at the input delay chain (D1).

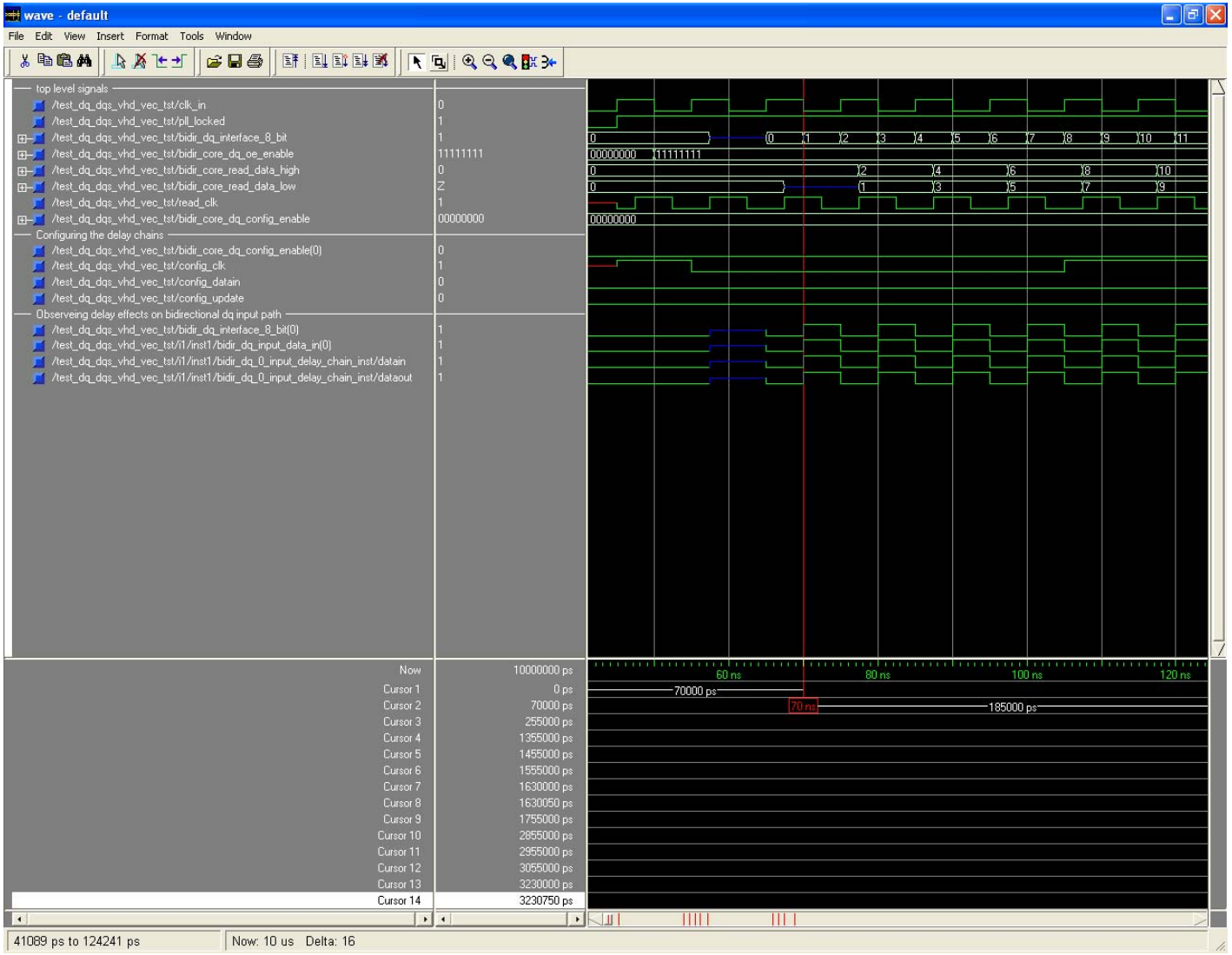
For more information about controlling these delay chains and how to vary the output delay chains (D5 and D6), refer to the *Design Example 1: Dynamically Changing Delay Chains in Output Buffer of Stratix III* section of the *I/O Buffer ALTIOBUF Megafunction User Guide*.

Setting the Input Delay Chain (D1) to Zero Delay (default)

Figure 4–19 shows that there are no timing difference with the cursor at 70 ns when D1 is set to zero delay. The cursor at 70 ns represents the path from the bidirectional buffer (`bidir_dq_input_data_in`) through the input delay chain (`bidir_dq_0_input_delay_chain_inst`). You can view the effects of the delay chain by comparing the `datain` port and the `dataout` port of the `bidir_dq_0_input_delay_chain_inst`.

Figure 4–19 shows the simulation results for D1 with zero delay.

Figure 4–19. Simulation Results—D1 is Set to Zero Delay (Default)



Setting the Input Delay Chain to 50 ps Delay

In Figure 4–20, cursor 3 (255 ns) to cursor 4 (1355 ns) show that the delay chain is configured to 50 ps.

The `config_clock` takes 11 (1,100 ns) clock cycles to load the intended delay values into the `IO_CONFIG` block because of the first four clock cycles (for the input delay chain, D1), the next three clock cycles (for the output delay chain 2, D6) and the last 4 clock cycles (for the output delay chain 1, D5).

The following steps describe how the input delay chain changes:

1. Because there is a 11-bit shift register in the `IO_CONFIG` block, `bidir_core_dq_confiq_enable(0)` is asserted for 11 clock cycles. When the shift registers are fully loaded, the shift registers have their bits arranges to correspond with datain values.
2. The `config_datain` signal is asserted at the 4th clock cycle to change the input delay chain value.
3. The delay only takes effect when the `config_update` signal is asserted for one clock cycle at 1455,000 ps (Cursor 5).
4. After the `config_update` signal is deasserted, the delay from `bidir_dq_0_input_delay_chain_inst/datain` at 1630,000 ps (Cursor 7) to `bidir_dq_0_input_delay_chain_inst/dataout` at 1630,050 ps (Cursor 8) is noticeable, which is 50 ps. Refer to Figure 4–21.

Figure 4–20 shows the first part of the simulation when you set the input delay chain to 50 ps delay.

Figure 4–20. First Part of the Simulation Results—D1 is set to 50 ps Delay

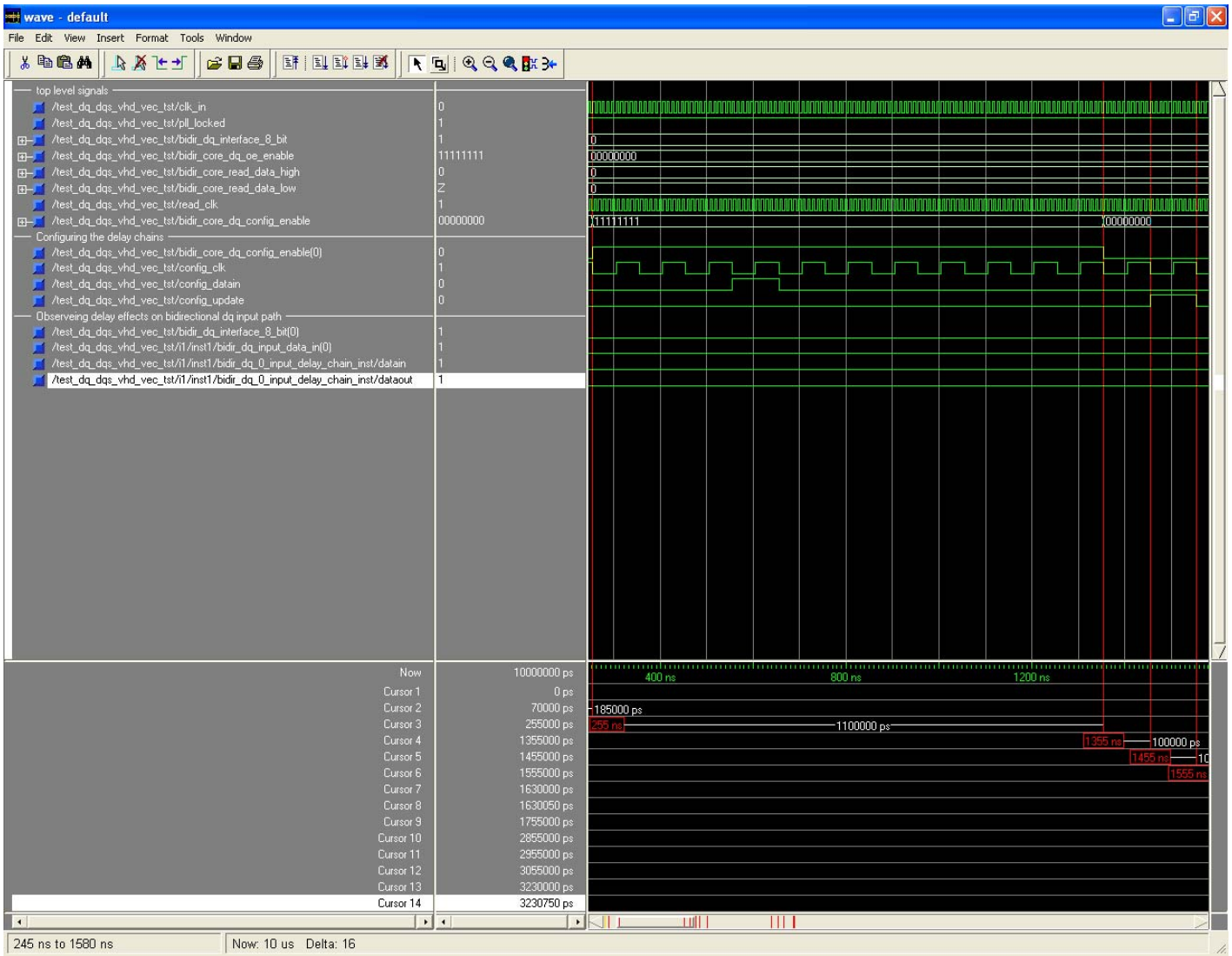
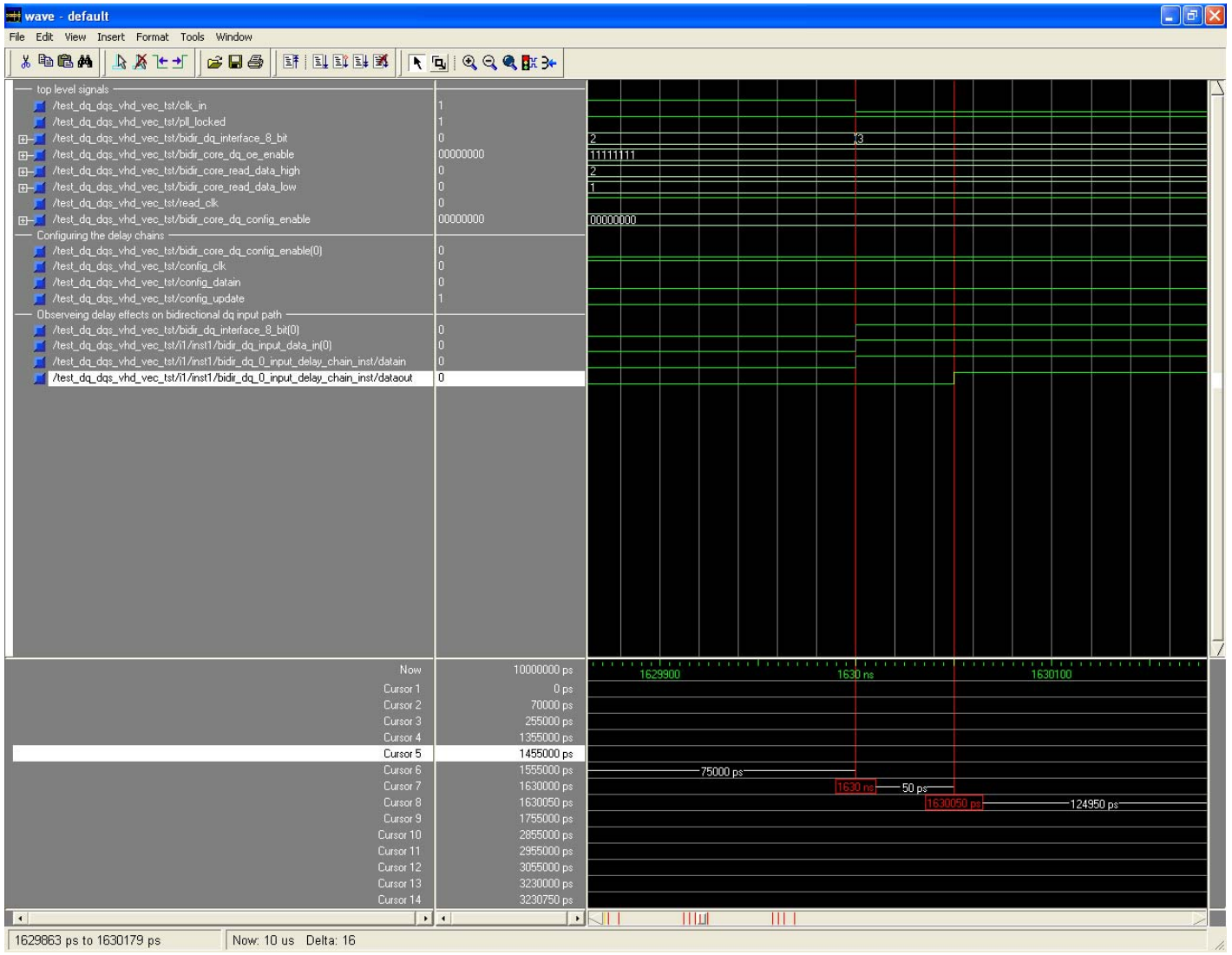


Figure 4–21 shows the second part of the simulation results when the effects of the 50 ps delay has been propagated.

Figure 4–21. Second Part of the Simulation Results—D1 is Set to 50 ps



Setting the Input Delay Chain to 750 ps Delay

Cursor 9 (1,755 ns) to cursor 10 (2,855 ns) in Figure 4–22 show that the input delay chain is configured to 750 ps.

The `config_clock` takes 11 (1,100 ns) clock cycles to load the intended delay values into the `IO_CONFIG` block because of the first four clock cycles (for the input delay chain, D1), the next three clock cycles (for the output delay chain 2, D6) and the last four clock cycles (for the output delay chain 1, D5).

The following steps describe how the input delay chain changes:

1. Because there is a 11-bit shift register in the `IO_CONFIG` block, `bidir_core_dq_confiq_enable(0)` is asserted for 11 clock cycles. When the shift registers are fully loaded, the shift registers have their bits arranges to correspond with datain values.
2. The `config_datain` signal is asserted at the next 4 clock cycles to change the input delay chain value.
3. The delay only takes effect when the `config_update` signal is asserted for one clock cycle at 2955,000 ps (Cursor 11).
4. After the `config_update` signal is deasserted, the delay from `bidir_dq_0_input_delay_chain_inst/datain` at 3230,000 ps (Cursor 13) to `bidir_dq_0_input_delay_chain_inst/dataout` at 3230,750 ps (Cursor 14) is noticeable, which is 750 ps. Refer to Figure 4–23.

Figure 4–21 shows the third part of the simulation results when you set the input delay chain to 750 ps delay.

Figure 4–22. Third Part of the Simulation Results—Input Delay Chain is set to 750 ps Delay

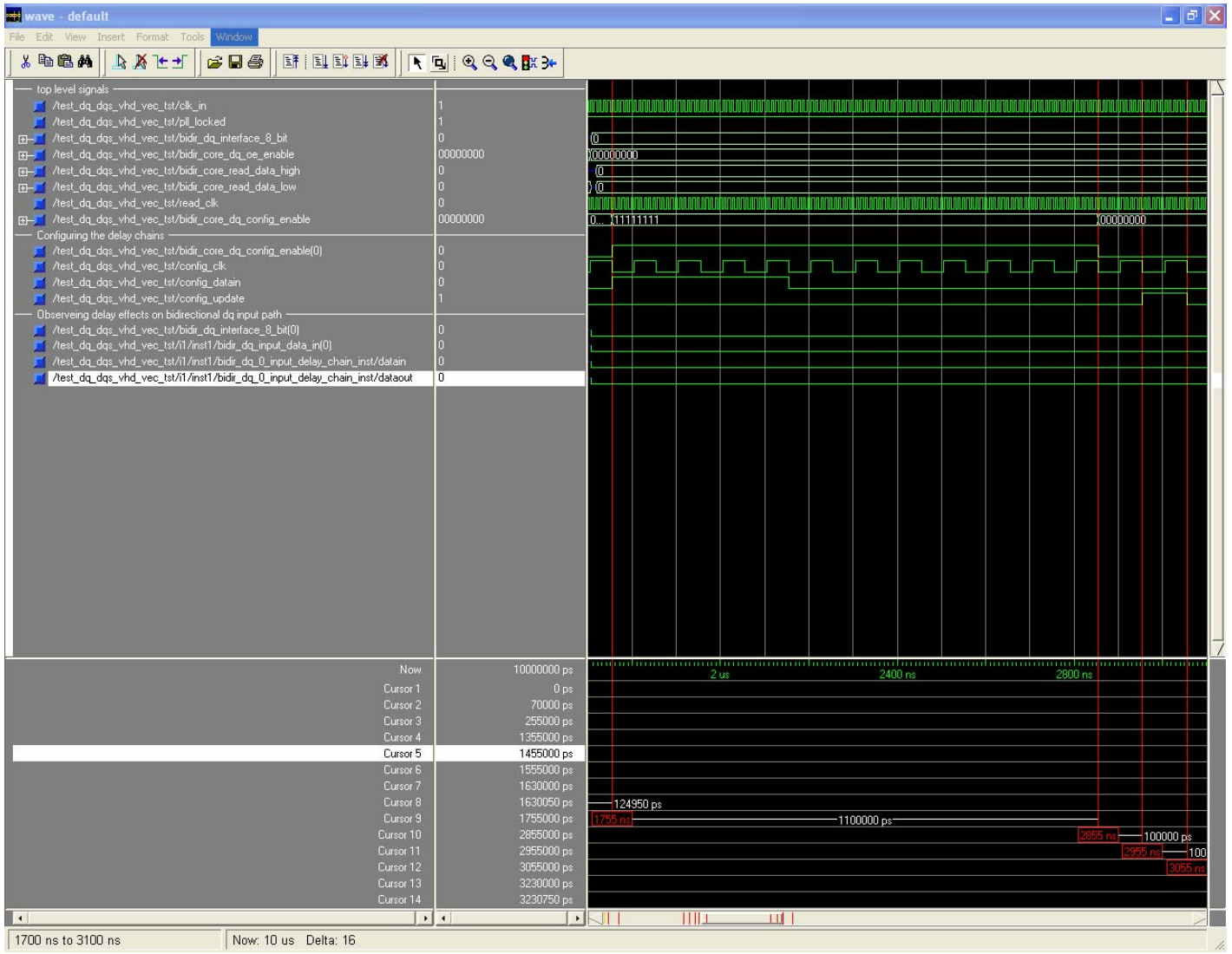
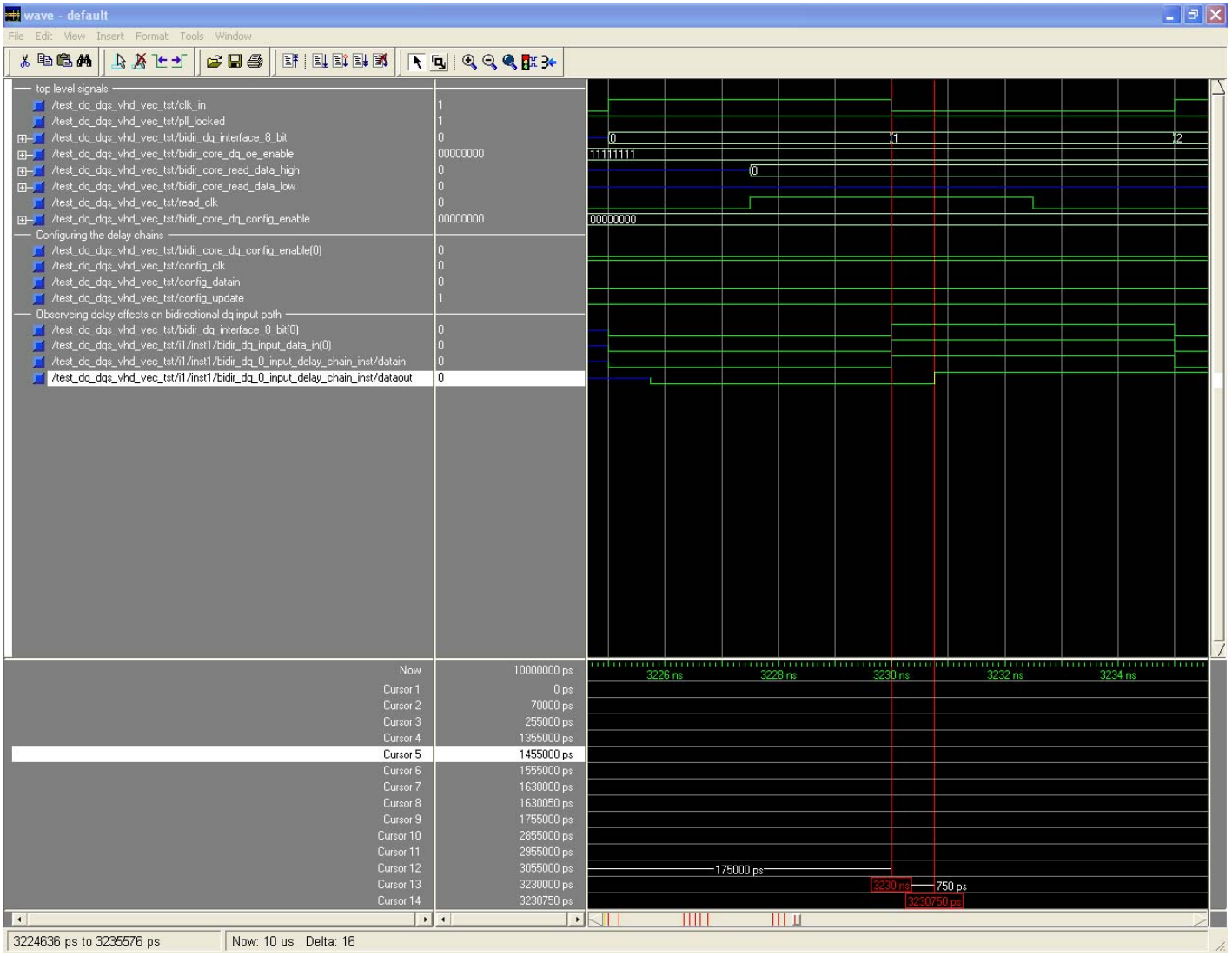


Figure 4–23 shows the fourth part of the simulation results when the effects of the 750 ps delay has been propagated.

Figure 4–23. Fourth Part of the Simulation Results—D1 is Set to 750 ps Delay



ALTDLL Megafunction Ports

This section describes the ports of the ALTDLL megafunction.

Table 4–9 lists the input ports for the ALTDLL megafunction.

Table 4–9. ALTDLL Megafunction Input Ports

Port Name	Optional/ Required	Default	Description
dll_aload	Optional	GND	Asynchronous load signal for the DLL counter. When dll_aload is HIGH, the counter is asynchronously loaded with the initial delay setting of 16 in low-frequency mode (when the parameter DELAY_BUFFER_MODE is set to LOW), or 32 in high-frequency mode (when the parameter DELAY_BUFFER_MODE is set to HIGH).
dll_clk	Required	GND	DLL reference clock that matches the frequency of the DQS clock used to determine the delay for the phase shift. Feed this input by an input pin or a PLL output. This input must match the polarity of its source and cannot be inverted.
dll_offset_ctrl_a_addnsub	Optional	V_{CC}	Addition/subtraction control port for DLL_OFFSET_CTRL_A block. This port controls whether the delay-offset setting A is added or subtracted. Ignore this input if the DLL_OFFSET_CTRL_A_USE_OFFSET parameter is set to FALSE. If the input is V_{CC}, the offset is added; if it is GND, the offset is subtracted.
dll_offset_ctrl_a_offset[5..0]	Optional	0	This is the offset input setting for DLL_OFFSET_CTRL_A block. This is a Gray-coded offset added or subtracted from the current value of the DLL's delay setting to get the dll_offset_ctrl_a_offsetctrlout result. Ignore this input if the DLL_OFFSET_CTRL_A_USE_OFFSET parameter is set to FALSE. The offset is limited to a minimum value of 0 and a maximum value of 63 in low-frequency mode, and a maximum value of 31 in high-frequency mode.
dll_offset_ctrl_b_addnsub	Optional	V_{CC}	This is the addition/subtraction control port for DLL_OFFSET_CTRL_B block. This port controls whether the delay-offset setting B is added or subtracted. Ignore this input if the DLL_OFFSET_CTRL_B_USE_OFFSET parameter is set to FALSE. If the input is V_{CC}, the offset is added; if it is GND, the offset is subtracted. This input defaults to V_{CC}.
dll_offset_ctrl_b_offset[5..0]	Optional	0	This is the offset input setting for DLL_OFFSET_CTRL_B block. This is a Gray-coded offset added or subtracted from the current value of the DLL's delay setting to get the dll_offset_ctrl_b_offsetctrlout result. Ignore this input if the DLL_OFFSET_CTRL_B_USE_OFFSET parameter is set to FALSE. The offset is limited to a minimum value of 0 and a maximum value of 63 in low-frequency mode, and a maximum value of 31 in high-frequency mode.

Table 4–10 lists the output ports of the ALTDLL megafunction.

Table 4–10. ALTDLL Megafunction Output Ports

Port Name	Optional/ Required	Default	Function
dll_delayctrlout [5..0]	Required	—	This is the DLL's delay setting output. This is a 1-cycle-delayed value of the current delay chain setting of the DLL. This signal is Gray-coded to minimize jitter due to toggling.This signal can feed the dll_delayctrlin input port of the ALTDQ_DQS megafunction or the core logic. This output is available for SignalTap® II Embedded Logic Analyzer.
dll_dqsupdate	Optional	—	This is an update-enable signal for the delay-setting latches of the DQS pins. This signal can feed the dqsupdateen input port of the ALTDQ_DQS megafunction. This output is not available for SignalTap II Embedded Logic Analyzer.
dll_offset_ctrl_a_ offsetctrlout[5..0]	Optional	—	This is the offsetctrlout output setting for DLL_OFFSET_CTRL_A block. This is a registered Gray-coded value of the delay-offset setting A. This output can be adjusted based on the value of the dll_offset_ctrl_a_use_offset parameter. This signal can feed the offsetctrlin input port of the ALTDQ_DQS megafunction. This signal is not available for SignalTap II Embedded Logic Analyzer.
dll_offset_ctrl_b_ offsetctrlout[5..0]	Optional	—	This is the offsetctrlout output setting for DLL_OFFSET_CTRL_B block. This is a registered Gray-coded value of the delay-offset setting B. This output can be adjusted based on the value of the dll_offset_ctrl_b_use_offset parameter. This signal can feed the offsetctrlin input port of the ALTDQ_DQS megafunction. This signal is not available for SignalTap II Embedded Logic Analyzer.

ALTDQ_DQS Megafunction Ports

Table 4–11 to Table 4–19 describes the ports of the ALTDQ_DQS megafunction that you can use to configure the DQS input path, DQS output path, DQS OE path, DQ/DQS OCT path, DQ input path, DQ output path, DQ OE path, DQSN IO path, and DQS_CONFIG/IO_CONFIG path.

DQS Input Path Megafunction Ports

Table 4–11 summarizes all the ports on the megafunction to configure the DQS input path.

n_b = number of bidirectional DQ
n_o = number of output DQ
n_i = number of input DQ
n_c = number of clock divider

Table 4–11. Megafunction Ports to Configure DQS Input Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
`core_delayctrlin[5..0]`	Input	Optional	GND	This port receives the Gray-coded delay chain setting for the DQS read path from the FPGA core. This port does not need to match the polarity of its source and can be inverted.
`dll_delayctrlin[5..0]`	Input	Optional	GND	This port receives the Gray-coded delay chain setting for the DQS read path from the `ALTDLL:delayctrlout[5..0]` port. This port must match the polarity of its source and cannot be inverted.
`dqs_bus_out`	Output	Optional	—	This port receives the possibly delayed DQS output signal from the `DQS_ENABLE:dqsbusout`, `DQSBUSOUT_DELAY_CHAIN:dataout`, or `DQS_DELAY_CHAIN:dqsbusout` port.
`dqs_enable_ctrl_clk`	Input	Optional	V_{CC}	This port is connected to the `DQS_ENABLE_CTRL:clk` port that is used to capture the `DQS_ENABLE_CTRL:dqsenablein` signal.
`dqs_enable_ctrl_hr_datainhi`	Input	Optional	GND	This port is connected to the `DQS_ENABLE_CTRL_HR_DDIO_OUT:datainhi` port. This port receives the half-rate data for the rising edge of the `IO_CLOCK_DIVIDER:clkout` signal.
`dqs_enable_ctrl_hr_datainlo`	Input	Optional	GND	This port is connected to the `DQS_ENABLE_CTRL_HR_DDIO_OUT:datainlo` port. This port receives the half-rate data for the falling edge of the `IO_CLOCK_DIVIDER:clkout` signal.
`dqs_enable_ctrl_in`	Input	Optional	V_{CC}	This active-high port is connected to the `DQS_ENABLE_CTRL:dqsenablein` port that is used to enable or disable the `DQS_ENABLE_CTRL:dqsenableout` port.

Table 4–11. Megafunction Ports to Configure DQS Input Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
dqs_enable_in	Input	Optional	V_{CC}	This active-high port is connected to the DQS_ENABLE:dqsenable that is used to enable or disable the DQS_ENABLE:dqsbusout port. When the dqs_enable_in port is connected to GND, the DQS_ENABLE:dqsbusout signal is GND on the next falling edge of the DQS_ENABLE:dqsin signal. The DQS_ENABLE:dqsbusout is connected directly to the dqs_bus_out port.
dqs_input_data_in	Input	Optional	GND	This port receives the incoming DQS signal for the DQS input path
dqs_input_data_out	Output	Optional	—	This port receives the outgoing DQS signal from the DQS_INPUT_DELAY_CHAIN:busout port, or directly from the dqs_input_data_in port
dqsupdateen	Input	Optional	GND	This active-high port is connected to the DQS_DELAY_CHAIN:dqsupdateen port that is used to latch the DQS_DELAY_CHAIN:delayctrlin[5..0] and DQS_DELAY_CHAIN:offsetctrlin[5..0] signals. The dqsupdateen port is fed by the ALTDLL:dll_dqsupdate port, or the core.
Io_clock_divider_clk	Input	Optional	GND	This port is connected to the IO_CLOCK_DIVIDER:clk port that is the clock input port for that block.
Io_clock_divider_ clkout[n_c-1..0]	Output	Optional	—	This port is connected to the IO_CLOCK_DIVIDER:clkout port that is used to output clock signal that is half the frequency of the IO_CLOCK_DIVIDER:clk signal.
Io_clock_divider_ masterin	Input	Optional	GND	This port is connected to the IO_CLOCK_DIVIDER:masterin port that is used when you need to chain multiple clock dividers together to feed wider DQS groups.
Io_clock_divider_ slaveout	Output	Optional	—	This port is connected to the IO_CLOCK_DIVIDER:slaveout port that is used when you need to chain multiple clock dividers together to feed wider DQS groups. This port must not have more than one fan-out and must only be connected to the io_clock_divider_masterin port of another ALTDQ_DQS megafunction.
offsetctrlin[5..0]	Input	Optional	GND	This port receives the Gray-coded fine-tune delay chain setting for the DQS output path from the ALTDLL:dll_offset_ctrl_a_offsetctrlout[5..0] port or ALTDLL:dll_offset_ctrl_b_offsetctrlout[5..0]. This port must match the polarity of its source and cannot be inverted.

DQS Output Path Megafunction Ports

Table 4–12 summarizes all the ports on the megafunction that configure the DQS output path.

Table 4–12. Megafunction Ports to Configure DQS Output Path

Port Name	Type	Optional/ Required	Default	Description
dqs_areset	Input	Optional	GND	This port is connected to the DQS_OUTPUT_FF:clrn, DQS_OUTPUT_DDIO_OUT:areset, and DQS_OUTPUT_HR_DDIO_OUT_HIGH/_LOW:areset ports that is used to asynchronously reset all registers in those blocks.
dqs_hr_output_data_in[3..0]	Input	Optional	GND	This port feeds the half-rate data to the DQS_OUTPUT_HR_DDIO_OUT_HIGH:datainhi / datainlo and DQS_OUTPUT_HR_DDIO_OUT_LOW:datainhi / datainlo ports.
dqs_hr_output_reg_clk	Input	Optional	GND	This port feeds the clock signal for the DQS_OUTPUT_HR_DDIO_OUT_HIGH:clkh / clklo / muxsel and DQS_OUTPUT_HR_DDIO_OUT_LOW:clkhi / clklo / muxsel ports.
dqs_output_data_in	Input	Optional	GND	This port feeds the DQS_OUTPUT_FF:d, DQS_OUTPUT_DELAY_CHAIN1:datain, DQS_OUTPUT_DELAY_CHAIN2:datain, or dqs_output_data_out port.
dqs_output_data_in_high	Input	Optional	GND	This port feeds the DQS_OUTPUT_DDIO_OUT:datainhi port that is the full-rate data for the rising edge.
dqs_output_data_in_low	Input	Optional	GND	This port feeds the DQS_OUTPUT_DDIO_OUT:datainlo port that is the full-rate data for the falling edge.
dqs_output_data_out	Output	Optional	—	This port can be driven by the DQS_OUTPUT_DELAY_CHAIN2:dataout, DQS_OUTPUT_DELAY_CHAIN1:dataout, DQS_OUTPUT_FF:q, DQS_OUTPUT_DDIO_OUT:dataout, or dqs_output_data_in port.
dqs_output_reg_clk	Input	Optional	GND	This port is connected to the DQS_OUTPUT_FF:clk and the DQS_OUTPUT_DDIO_OUT:clkhi/clklo/muxsel ports that is used to clock the registers in those blocks.
dqs_output_reg_clkena	Input	Optional	V_{CC}	This port is connected to the DQS_OUTPUT_FF:ena and the DQS_OUTPUT_DDIO_OUT:ena ports that is used as output enable for the registers in those block.
dqs_sreset	Input	Optional	GND	This port is connected to the DQS_OUTPUT_FF:sclr and DQS_OUTPUT_DDIO_OUT:sreset ports that is used to synchronously reset all registers in those blocks.

DQS OE Path Megafunction Ports

Table 4–13 summarizes all the ports on the megafunction that configure the DQS OE path.

Table 4–13. Megafunction Ports to Configure DQS OE Path

Port Name	Type	Optional/ Required	Default	Description
dqs_areset	Input	Optional	GND	This port is connected to the DQS_OE_FF:clrn, DQS_OE_DDIO_OE:areset, and DQS_OE_HR_DDIO_OUT:areset ports that is used to asynchronously reset all registers in those blocks.
dqs_hr_oe_in[1..0]	Input	Optional	GND	This 2-bit port is connected to the DQS_OE_HR_DDIO_OUT:datainhi /datainlo port that is used as the output enable for the half-rate registers in that block.
dqs_hr_output_reg_clk	Input	Optional	GND	This port is connected to the DQS_OE_HR_DDIO_OUT:clkhi / clklo / muxsel ports that is used to clock the half-rate registers in those blocks.
dqs_oe_in	Input	Optional	GND	This port feeds the DQS_OE_FF:d, DQS_OE_DDIO_OE:oe, DQS_OE_DELAY_CHAIN1:datain, DQS_OE_DELAY_CHAIN2:datain, or dqs_oe_out port. For information about how to enable these blocks, refer to "Parameter Settings" on page 3–1.
dqs_oe_out	Output	Optional	—	This port receives the output signal from the DQS_OE_DELAY_CHAIN2:dataout, DQS_OE_DELAY_CHAIN1:dataout, DQS_OE_FF:q, DQS_OE_DDIO_OE:dataout, or dqs_oe_in port. For information about how to enable these blocks, refer to "Parameter Settings" on page 3–1.
dqs_output_reg_clk	Input	Optional	GND	This port is connected to the DQS_OE_FF:clk and the DQS_OE_DDIO_OE:clk ports that is used to clock the registers in those blocks.
dqs_output_reg_clkena	Input	Optional	V_{CC}	This port is connected to the DQS_OE_FF:ena and the DQS_OE_DDIO_OE:ena ports that is used as output enable for the registers in those block.
dqs_sreset	Input	Optional	GND	This port is connected to the DQS_OE_FF:sclr and DQS_OE_DDIO_OE:sreset ports that is used to synchronously reset all registers in those blocks.

DQ/DQS OCT Path Megafunction Ports

Table 4–14 summarizes all the ports on the megafunction that configure the OCT path. The possible values for <*IO*> are DQS, DQSn, BIDIR_DQ, and OUTPUT_DQ.

Table 4–14. Megafunction Ports to Configure OCT Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
bidir_dq_hr_oct_in [2*n_b-1..0]	Input	Optional	GND	This port feeds the half-rate bidirectional DQ signal for the BIDIR_DQ_OCT_HR_DDIO_OUT:datainhi / datainlo ports.
bidir_dq_oct_in [n_b-1..0]	Input	Optional	GND	This port feeds the full-rate bidirectional DQ signal for the BIDIR_DQ_OCT_FF:d, BIDIR_DQ_OCT_DDIO_OE:oe, BIDIR_DQ_OCT_DELAY_CHAIN1:datain, BIDIR_DQ_OCT_DELAY_CHAIN2:datain, or bidir_dq_oct_out port.
bidir_dq_oct_out [n_b-1..0]	Output	Optional	—	This port outputs signal from the BIDIR_DQ_OCT_DELAY_CHAIN2:dataout, BIDIR_DQ_OCT_DELAY_CHAIN1:dataout, BIDIR_DQ_OCT_FF:q, BIDIR_DQ_OCT_DDIO_OE:dataout, or bidir_dq_oct_in port.
dqs_hr_oct_in[1..0]	Input	Optional	GND	This port feeds the half-rate DQS signal for the DQS_OCT_HR_DDIO_OUT:datainhi / datainlo ports.
dqs_oct_in	Input	Optional	GND	This port feeds the full-rate DQS signal for the DQS_OCT_FF:d, DQS_OCT_DDIO_OE:oe, DQS_OCT_DELAY_CHAIN1:datain, DQS_OCT_DELAY_CHAIN2:datain, or dqs_oct_out port.
dqs_oct_out	Output	Optional	—	This port outputs signal from the DQS_OCT_DELAY_CHAIN2:dataout, DQS_OCT_DELAY_CHAIN1:dataout, DQS_OCT_FF:q, DQS_OCT_DDIO_OE:dataout, or dqs_oct_in port.
dqsn_hr_oct_in[1..0]	Input	Optional	GND	This port feeds the half-rate DQSn signal for the DQSN_OCT_HR_DDIO_OUT:datainhi / datainlo ports.
dqsn_oct_in	Input	Optional	GND	This port feeds the full-rate DQSn signal for the DQSN_OCT_FF:d, DQSN_OCT_DDIO_OE:oe, DQSN_OCT_DELAY_CHAIN1:datain, DQSN_OCT_DELAY_CHAIN2:datain, or dqsn_oct_out port.
dqsn_oct_out	Output	Optional	—	This port outputs signal from the DQSN_OCT_DELAY_CHAIN2:dataout, DQSN_OCT_DELAY_CHAIN1:dataout, DQSN_OCT_FF:q, DQSN_OCT_DDIO_OE:dataout, or dqsn_oct_in port.
hr_oct_reg_clk	Input	Optional	GND	This port feeds the half-rate clock signal for the <*IO*>_OCT_HR_DDIO_OUT:clkhi/clklo/muxsel.

Table 4–14. Megafunction Ports to Configure OCT Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
oct_reg_clk	Input	Optional	GND	This port feeds the full-rate clock signal to the <*IO*>_OCT_FF:clk and <*IO*>_OCT_DDIO_OE:clk ports.
output_dq_hr_oct_in [2*n_o-1..0]	Input	Optional	GND	This port feeds the half-rate output DQ signal for the OUTPUT_DQ_OCT_HR_DDIO_OUT:datainhi / datainlo ports.
output_dq_oct_in [n_o-1..0]	Input	Optional	GND	This port feeds the full-rate output DQ signal for the OUTPUT_DQ_OCT_FF:d, OUTPUT_DQ_OCT_DDIO_OE:oe, OUTPUT_DQ_OCT_DELAY_CHAIN1:datain, OUTPUT_DQ_OCT_DELAY_CHAIN2:datain, or output_dq_oct_out port.
output_dq_oct_out [n_o-1..0]	Output	Optional	—	This port outputs signal from the OUTPUT_DQ_OCT_DELAY_CHAIN2:dataout, OUTPUT_DQ_OCT_DELAY_CHAIN1:dataout, OUTPUT_DQ_OCT_FF:q, OUTPUT_DQ_OCT_DDIO_OE:dataout, or output_dq_oct_in port.

DQ Input Path Megafunction Ports

Table 4–15 summarizes all the ports on the megafunction that configure the DQ input path. The possible values for <*IO*> are BIDIR_DQ and INPUT_DQ.

Table 4–15. Megafunction Ports to Configure DQ Input Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
bidir_dq_areset [n_b-1..0]	Input	Optional	GND	This port is connected to all areset port in the bidirectional DQ IO primitives that is used to asynchronously reset the registers in those primitives.
bidir_dq_hr_input_ data_out[4*n_b-1..0]	Output	Optional	—	This port outputs the half-rate DDR bidirectional DQ signal from the BIDIR_DQ_HALF_RATE_INPUT:dataout port.
bidir_dq_input_data_in [n_b-1..0]	Input	Optional	GND	This port feeds the bidirectional DQ signal for the BIDIR_DQ_INPUT_DELAY_CHAIN:datain, BIDIR_DQ_INPUT_FF:d, BIDIR_DQ_DDIO_IN:datain, or bidir_dq_input_data_out port.
bidir_dq_input_data_ou t_high[n_b-1..0]	Output	Optional	—	This port outputs the full-rate DDR bidirectional DQ signal (rising edge) from the BIDIR_DQ_IPA_HIGH:dataout or BIDIR_DQ_DDIO_IN:regouthi.
bidir_dq_input_data_ou t_low[n_b-1..0]	Output	Optional	—	This port outputs the full-rate DDR bidirectional DQ signal (falling edge) from the BIDIR_DQ_IPA_LOW:dataout or BIDIR_DQ_DDIO_IN:regoutlo.

Table 4–15. Megafunction Ports to Configure DQ Input Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
bidir_dq_input_data_out[n_b-1..0]	Output	Optional	—	This port outputs the bidirectional DQ signal from the BIDIR_DQ_INPUT_DELAY_CHAIN:dataout, BIDIR_DQ_INPUT_FF:q, or bidir_dq_input_data_in port.
bidir_dq_sreset n_b-1..0]	Input	Optional	GND	This port is connected to all sreset port in the bidirectional DQ IO primitives that is used to synchronously reset the registers in those primitives.
dll_delayctrlin 5..0]	Input	Optional	GND	This port receives the Gray-coded delay chain setting for the DQ read path from the delayctrlout[5..0] port of the ALTDLL.
dq_input_reg_clk	Input	Optional	GND	This port feeds the clock signal for the *<IO>*_INPUT_FF:clk and *<IO>*_DDIO_IN:clk ports.
dq_input_reg_clkena	Input	Optional	V_{CC}	This port feeds the output enable signal for the *<IO>*_INPUT_FF:ena and *<IO>*_DDIO_IN:ena ports.
dq_ipa_clk	Input	Optional	GND	This port feeds the clock signal for the *<IO>*_IPA_HIGH:clk and *<IO>*_IPA_LOW:clk ports.
input_dq_areset [n_i-1..0]	Input	Optional	GND	This port is connected to all areset port in the input DQ IO primitives that is used to asynchronously reset the registers in those primitives.
input_dq_hr_input_data_out[4*n_i-1..0]	Output	Optional	—	This port outputs the half-rate DDR input DQ signal from the INPUT_DQ_HALF_RATE_INPUT:dataout port.
input_dq_input_data_in [n_i-1..0]	Input	Optional	GND	This port feeds the input DQ signal for the INPUT_DQ_INPUT_DELAY_CHAIN:datain, INPUT_DQ_INPUT_FF:d, INPUT_DQ_DDIO_IN:datain, or input_dq_input_data_out port.
input_dq_input_data_out_high[n_i-1..0]	Output	Optional	—	This port outputs the full-rate DDR input DQ signal (rising edge) from the INPUT_DQ_IPA_HIGH:dataout or INPUT_DQ_DDIO_IN:regouthi.
input_dq_input_data_out_low[n_i-1..0]	Output	Optional	—	This port outputs the full-rate DDR input DQ signal (falling edge) from the INPUT_DQ_IPA_LOW:dataout or INPUT_DQ_DDIO_IN:regoutlo.
input_dq_input_data_out[n_i-1..0]	Output	Optional	—	This port outputs the input DQ signal from the INPUT_DQ_INPUT_DELAY_CHAIN:dataout, INPUT_DQ_INPUT_FF:q, or input_dq_input_data_in port.
input_dq_sreset [n_i-1..0]	Input	Optional	GND	This port is connected to all sreset ports in the input DQ IO primitives that is used to synchronously reset the registers in those primitives.

DQ Output Path Megafunction Ports

Table 4–16 summarizes all the ports on the megafunction that configure the DQ Output path. The possible values for <*IO*> are `BIDIR_DQ` and `OUTPUT_DQ`.

Table 4–16. Megafunction Ports to Configure DQ Output Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
`bidir_dq_areset [`n_b`-1..0]`	Input	Optional	GND	This port is connected to all `areset` ports in the bidirectional DQ IO primitives that is used to asynchronously reset the registers in those primitives.
`bidir_dq_hr_output_data_in[4*`n_b`-1..0]`	Input	Optional	GND	This port feeds the half-rate DDR bidirectional DQ signal for the `BIDIR_DQ_OUTPUT_HR_DDIO_OUT_HIGH:datainhi` / `datainlo` and `BIDIR_DQ_OUTPUT_HR_DDIO_OUT_LOW:datainhi` / `datainlo` ports.
`bidir_dq_output_data_in [`n_b`-1..0]`	Input	Optional	GND	This port feeds the bidirectional DQ signal for the `BIDIR_DQ_OUTPUT_FF:d`, `BIDIR_DQ_OUTPUT_DELAY_CHAIN1:datain`, `BIDIR_DQ_OUTPUT_DELAY_CHAIN2:datain`, or `bidir_dq_output_data_out` port
`bidir_dq_output_data_in_high[`n_b`-1..0]`	Input	Optional	GND	This port feeds the full-rate DDR bidirectional DQ signal (rising edge) for the `BIDIR_DQ_OUTPUT_DDIO_OUT:datainhi` port.
`bidir_dq_output_data_in_low[`n_b`-1..0]`	Input	Optional	GND	This port feeds the full-rate DDR bidirectional DQ signal (falling edge) for the `BIDIR_DQ_OUTPUT_DDIO_OUT:datainlo` port.
`bidir_dq_output_data_out[`n_b`-1..0]`	Output	Optional	—	This port outputs the bidirectional DQ signal from the `BIDIR_DQ_OUTPUT_DELAY_CHAIN2:dataout`, `BIDIR_DQ_OUTPUT_DELAY_CHAIN1:dataout`, `BIDIR_DQ_OUTPUT_FF:q`, `BIDIR_DQ_OUTPUT_DDIO_OUT:dataout`, or `bidir_dq_output_data_in` port.
`bidir_dq_sreset [`n_b`-1..0]`	Input	Optional	GND	This port is connected to all `sreset` port in the bidirectional DQ IO primitives that is used to synchronously reset the registers in those primitives.
`dq_hr_output_reg_clk`	Input	Optional	GND	This port feeds the output enable signal for the <*IO*>`_OUTPUT_FF:ena` and <*IO*>`_OUTPUT_DDIO_OUT:ena` ports.
`dq_output_reg_clk`	Input	Optional	GND	This port feeds the clock signal for the <*IO*>`_OUTPUT_FF:clk` and <*IO*>`_OUTPUT_DDIO_OUT:clkhi` / `clklo` / `muxsel` ports.

Table 4–16. Megafunction Ports to Configure DQ Output Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
dq_output_reg_clkena	Input	Optional	V_{CC}	This port feeds the output enable signal for the <*IO*>_OUTPUT_FF:ena and <*IO*>_OUTPUT_DDIO_OUT:ena ports.
output_dq_areset [n_o-1..0]	Input	Optional	GND	This port is connected to all areset port in the output DQ IO primitives that is used to asynchronously reset the registers in those primitives.
output_dq_hr_output_ data_in[4*n_o-1..0]	Input	Optional	GND	This port feeds the half-rate DDR output DQ signal for the OUTPUT_DQ_OUTPUT_HR_DDIO_OUT_HIGH :datainhi / datainlo and OUTPUT_DQ_OUTPUT_HR_DDIO_OUT_LOW: datainhi / datainlo ports.
output_dq_output_data_ in_high[n_o-1..0]	Input	Optional	GND	This port feeds the full-rate DDR output DQ signal (rising edge) for the OUTPUT_DQ_OUTPUT_DDIO_OUT:datainh i port.
output_dq_output_data_ in_low[n_o-1..0]	Input	Optional	GND	This port feeds the full-rate DDR output DQ signal (falling edge) for the OUTPUT_DQ_OUTPUT_DDIO_OUT:datainl o port.
output_dq_output_data_ in[n_o-1..0]	Input	Optional	GND	This port feeds the output DQ signal for the OUTPUT_DQ_OUTPUT_FF:d, OUTPUT_DQ_OUTPUT_DELAY_CHAIN1:dat ain, OUTPUT_DQ_OUTPUT_DELAY_CHAIN2:dat ain, or output_dq_output_data_out port.
output_dq_output_data_ out[n_o-1..0]	Output	Optional	—	This port outputs the output DQ signal from the OUTPUT_DQ_OUTPUT_DELAY_CHAIN2:dat aout, OUTPUT_DQ_OUTPUT_DELAY_CHAIN1:dat aout, OUTPUT_DQ_OUTPUT_FF:q, OUTPUT_DQ_OUTPUT_DDIO_OUT:dataout, or output_dq_output_data_in port.
output_dq_sreset [n_o-1..0]	Input	Optional	GND	This port is connected to all sreset ports in the output DQ IO primitives that is used to synchronously reset the registers in those primitives.

DQ OE Path Megafunction Ports

Table 4–17 summarizes all the ports on the megafunction that configure the DQ OE path. The possible values for <*IO*> are `BIDIR_DQ` and `OUTPUT_DQ`.

Table 4–17. Megafunction Ports to Configure DQ OE Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
`bidir_dq_areset` [n_b-1..0]	Input	Optional	GND	This port is connected to all `areset` port in the bidir DQ IO primitives that is used to asynchronously reset the registers in those primitives.
`bidir_dq_hr_oe_in` [2*n_b-1..0]	Input	Optional	GND	This port feeds the half-rate bidirectional DQ OE signal for the `BIDIR_DQ_OE_HR_DDIO_OUT:datainhi` / `datainlo` ports.
`bidir_dq_oe_in` [n_b-1..0]	Input	Optional	GND	This port feeds the bidirectional DQ OE signal for the `BIDIR_DQ_OE_FF:d`, `BIDIR_DQ_OE_DDIO_OE:oe`, `BIDIR_DQ_OE_DELAY_CHAIN1:datain`, `BIDIR_DQ_OE_DELAY_CHAIN2:datain`, or `bidir_dq_oe_out` port.
`bidir_dq_oe_out` [n_b-1..0]	Output	Optional	—	This port is driven by the `BIDIR_DQ_OE_DELAY_CHAIN2:dataout`, `BIDIR_DQ_OE_DELAY_CHAIN1:dataout`, `BIDIR_DQ_OE_FF:q`, `BIDIR_DQ_OE_DDIO_OE:dataout`, or `bidir_dq_oe_in` port.
`bidir_dq_sreset` [n_b-1..0]	Input	Optional	GND	This port is connected to all `sreset` port in the bidir DQ IO primitives that is used to synchronously reset the registers in those primitives.
`dq_hr_output_reg_clk`	Input	Optional	GND	This port feeds the half-rate clock signal for the <*IO*>`_OE_HR_DDIO_OUT:clkhi/clklo/muxsel` ports. The clock signal is for the half-rate DDIO registers.
`dq_output_reg_clk`	Input	Optional	GND	This port feeds the clock signal for the <*IO*>`_OE_FF:clk` and <*IO*>`_OE_DDIO_OE:clk` ports.
`dq_output_reg_clkena`	Input	Optional	V_{CC}	This port feeds the output enable signal for the <*IO*>`_OE_FF:ena` and <*IO*>`_OE_DDIO_OE:ena` ports.
`output_dq_areset` [n_o-1..0]	Input	Optional	GND	This port is connected to all `areset` ports in the output DQ IO primitives that is used to asynchronously reset the registers in those primitives.
`output_dq_hr_oe_in` [2* n_o-1..0]	Input	Optional	GND	This port feeds the half-rate output DQ OE signal for the `OUTPUT_DQ_OE_HR_DDIO_OUT:datainhi` / `datainlo` ports.

Table 4–17. Megafunction Ports to Configure DQ OE Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
output_dq_oe_in [n_o-1..0]	Input	Optional	GND	This port feeds the bidirectional DQ OE signal for the OUTPUT_DQ_OE_FF:d, OUTPUT_DQ_OE_DDIO_OE:oe, OUTPUT_DQ_OE_DELAY_CHAIN1:datain, OUTPUT_DQ_OE_DELAY_CHAIN2:datain, or output_dq_oe_out port.
output_dq_oe_out [n_o-1..0]	Output	Optional	—	This port is driven by the OUTPUT_DQ_OE_DELAY_CHAIN2:dataout , OUTPUT_DQ_OE_DELAY_CHAIN1:dataout , OUTPUT_DQ_OE_FF:q, OUTPUT_DQ_OE_DDIO_OE:dataout, or output_dq_oe_in port.
output_dq_sreset [n_o-1..0]	Input	Optional	GND	This port is connected to all sreset ports in the output DQ IO primitives that is used to synchronously reset the registers in those primitives.

DQSn I/O Path Ports

Table 4–18 summarizes all the ports that are specific to the DQSn I/O. All other ports are shared with the DQS IO.

Table 4–18. Megafunction Ports to Configure DQSN IO Path (Part 1 of 2)

Port Name	Type	Optional/ Required	Default	Description
dqsn_areset	Input	Optional	GND	This port is connected to all areset ports in the DQSn IO primitives that is used to asynchronously reset the registers in those primitives.
dqsn_bus_out	Output	Optional	—	This port outputs the signal from DQSN_ENABLE:dqsbusout, DQSNBUSOUT_DELAY_CHAIN:dataout, or DQSN_DELAY_CHAIN:dqsbusout port.
dqsn_hr_oe_in[1..0]	Input	Optional	GND	This port feeds the half-rate DDR signal to the DQSn OE path. This port is connected to the DQSN_OE_HR_DDIO_OUT:datainhi / datainlo port.
dqsn_hr_output_data_in [3..0]	Input	Optional	GND	This port feeds the half-rate DDR input signal to the DQSn output path. This port is connected to the DQSN_OUTPUT_HR_DDIO_OUT_HIGH: datainhi / datainlo and DQSN_OUTPUT_HR_DDIO_OUT_LOW: datainhi / datainlo ports.
dqsn_input_data_in	Input	Optional	GND	This port feeds the input signal to the DQSn input path. This port is connected to the DQSN_DELAY_CHAIN:dqsin, DQSN_INPUT_DELAY_CHAIN:datain, or dqsn_input_data_out port.

Table 4–18. Megafunction Ports to Configure DQSN IO Path (Part 2 of 2)

Port Name	Type	Optional/ Required	Default	Description
dqsn_input_data_out	Output	Optional	—	This port outputs the signal from the DQSn input path. This port is connected to the DQSN_INPUT_DELAY_CHAIN:dataout or dqsn_input_data_in port.
dqsn_oe_in	Input	Optional	GND	This port feeds the input signal for the DQSn OE path. This port is connected to the DQSN_OE_FF:d, DQSN_OE_DDIO_OE:oe, DQSN_OE_DELAY_CHAIN1:datain, DQSN_OE_DELAY_CHAIN2:datain, dqsn_oe_out port.
dqsn_oe_out	Output	Optional	—	This port is fed by the output signal from DQSn OE path. This port can be driven by the DQSN_OE_DELAY_CHAIN2:dataout, DQSN_OE_DELAY_CHAIN1:dataout, DQSN_OE_FF:q, DQSN_OE_DDIO_OE:dataout, dqsn_oe_in port.
dqsn_output_data_in	Input	Optional	GND	This port feeds the input signal for DQSn output path. This port is connected to the DQSN_OUTPUT_FF:d, DQSN_OUTPUT_DELAY_CHAIN1:datain, DQSN_OUTPUT_DELAY_CHAIN2:datain, or dqsn_output_data_out port.
dqsn_output_data_in_high	Input	Optional	GND	This port feeds the full-rate DDR input signal (rising edge) to the DQSn output path. This port is connected to the DQSN_OUTPUT_DDIO_OUT:datainhi port.
dqsn_output_data_in_low	Input	Optional	GND	This port feeds the full-rate DDR input signal (falling edge) to the DQSn output path. This port is connected to the DQSN_OUTPUT_DDIO_OUT:datainlo port.
dqsn_output_data_out	Output	Optional	—	This port outputs the output signal from the DQSn output path. This port can be driven by DQSN_OUTPUT_DELAY_CHAIN2:dataout, DQSN_OUTPUT_DELAY_CHAIN1:dataout, DQSN_OUTPUT_FF:q, DQSN_OUTPUT_DDIO_OUT:dataout, or dqsn_output_data_in port.
dqsn_sreset	Input	Optional	GND	This port is connected to all sreset ports in the DQSn IO primitives that is used to synchronously reset the registers in those primitives.

DQS_CONFIG/IO_CONFIG Megafunction Ports

Table 4–19 summarizes all the ports on the megafunction that configure the DQS_CONFIG/IO_CONFIG path.

Table 4–19. Megafunction Ports to Configure DQS_CONFIG/IO_CONFIG Path

Port Name	Type	Optional/ Required	Default	Description
bidir_dq_io_config_ena [n_b-1..0]	Input	Optional	V_{CC}	Enable signal for the BIDIR_DQ_IO_CONFIG block.
config_clk	Input	Optional	GND	Clock signal for the DQS_CONFIG and IO_CONFIG blocks.
config_datain	Input	Optional	GND	Input signal for the DQS_CONFIG and IO_CONFIG blocks.
config_update	Input	Optional	GND	Update signal for the DQS_CONFIG and IO_CONFIG blocks.
dqs_config_ena	Input	Optional	V_{CC}	Enable signal for the DQS_CONFIG block.
dqs_io_config_ena	Input	Optional	V_{CC}	Enable signal for the DQS_IO_CONFIG block.
dqsn_io_config_ena	Input	Optional	V_{CC}	Enable signal for the DQSN_IO_CONFIG block.
input_dq_io_config_ena [n_b-1..0]	Input	Optional	V_{CC}	Enable signal for the INPUT_DQ_IO_CONFIG block.
output_dq_io_config_ena [n_b-1..0]	Input	Optional	V_{CC}	Enable signal for the OUTPUT_DQ_IO_CONFIG block.

Correct Settings for External Memory Interfaces

Table 4–20 shows the correct settings required for the ALTDLL and ALTDQ_DQS megafunctions to work in the DDR, QDR, and RLDRAM interfaces.

n represents the number of pins in a path. The value of *n* ranges from 0 to 48, but varies according to the memory interface used. To determine the value of *n* for a particular memory interface, the *External Memory Interface* chapter of the respective device handbooks.

Table 4–20. Correct Settings for DDR, QDR, and RLDRAM Interfaces

Parameter	DDR	QDR		RLDRAM	
		read	write	read	write
RLDRAMII mode	Unused	Unused		Turned on. Refer to the "ALTDQ_DQS Parameter Editor" on page 3–5.	
Data mask pin group	Unused	Unused			
Q valid signal group	Unused	Unused			
Number of bidirectional DQ	*n*	0	0	*n*	
Number of input DQ	0	*n*	0	0	
Number of output DQ	0	0	*n*	0	
Enable DQ output enable path	Turned on	Turned off	Turned on	Turned on	
Use half-rate components	For full-rate controller: Turned off			For half-rate controller: Turned on	
Use dynamic OCT path	Turned on			Turned on	
Enable DQS input path	Turned on	Turned on	Turned off	Turned on	Turned off
Enable DQS output path	Turned on	Turned off		Turned off	
Enable DQS OE path	Turned on	Turned off		Turned off	
DQS/DQSn IO configuration mode	If used: **Differential pair** If single_ended: Turned off	**Complementary pair**		**Differential pair**	
DQS input frequency	<*x*> MHz (e.g. 400 MHz)				
DQS delay chain phase setting	<*n*> = `phase_shift` / 360 x `DLL_delay_chain_length`				
DQS delay chain 'delayctrlin' port source	**DLL**				
Enable DQS input delay chain	Default:Turned off If used: Turned on				
Delay buffer mode	**High** or **Low** (depending on the ALTDLL instantiation settings)				
Enable DQS delay chain	Turned on				

Table 4–2 shows the correct port use for DDR, QDR, and RLDRAM interfaces.

Table 4–21. Correct Port Use for DDR, QDR and RLDRAM Interfaces

Port Name	Controllers	
	Full-Rate	Half-Rate
INPUT_DQ_INPUT_DATA_IN	Used	Used
INPUT_DQ_INPUT_DATA_OUT_HIGH	Used	Unused
INPUT_DQ_INPUT_DATA_OUT_LOW	Used	Unused
INPUT_DQ_HR_INPUT_DATA_OUT	Unused	Used
OUTPUT_DQ_OUTPUT_DATA_OUT	Used	Used
OUTPUT_DQ_OUTPUT_DATA_IN_LOW	Used	Unused
OUTPUT_DQ_OUTPUT_DATA_IN_HIGH	Used	Unused
OUTPUT_DQ_HR_OUTPUT_DATA_IN	Unused	Used
OUTPUT_DQ_HR_OE_IN	Unused	Used
OUTPUT_DQ_OE_IN	Used	Unused
OUTPUT_DQ_OE_OUT	Used	Used
BIDIR_DQ_INPUT_DATA_IN	Used	Used
BIDIR_DQ_HR_INPUT_DATA_OUT	Unused	Used
BIDIR_DQ_OUTPUT_DATA_OUT	Used	Used
BIDIR_DQ_HR_OUTPUT_DATA_IN	Unused	Used
DQS_INPUT_DATA_IN	Used	Used
DQS_HR_OUTPUT_DATA_IN	Unused	Used
DQSN_INPUT_DATA_IN	Used	Used
DQSN_HR_OUTPUT_DATA_IN	Unused	Used
DQS_BUS_OUT	Used	Used
DQS_OUTPUT_DATA_OUT	Used	Used
DQ_INPUT_REG_CLK	Used	Used
DQ_OUTPUT_REG_CLK	Used	Unused
DQ_HR_OUTPUT_REG_CLK	Unused	Used
DLL_DELAYCTRLIN	Used	Used
IO_CLOCK_DIVIDER_CLK	Used	Used

Table 4–22 shows the correct OCT parameter settings for the DDR, QDR, and RLDRAM interfaces.

Table 4–22. General OCT Parameter Settings for DDR, QDR, and RLDRAM Interfaces

Parameter	Controller	
	Full-Rate	Half-Rate
Enable Dynamic OCT	Turned on	Turned on
Enable OCT delay chain 1	Turned on / Turned off	Turned on / Turned off
Enable OCT delay chain 2	Turned on / Turned off	Turned on / Turned off
OCT register mode	FF	FF

Table 4–23 shows the correct OCT port use for the DDR, QDR, and RLDRAM interfaces.

Table 4–23. General OCT Ports for DDR, QDR, and RLDRAM Interfaces

Parameter	Controller	
	Full-Rate	Half-Rate
DQS_OCT_IN	Used	Used
DQSN_OCT_IN	Used	Used
BIDIR_DQ_OCT_IN	Used if DQ pin is bidirectional	Used if DQ pin is bidirectional
INPUT_OCT_IN	Used for DQ pin as input	Used for DQ pin as input
OUTPUT_OCT_IN	Used for DQ pin as output	Used for DQ pin as output
DQS_HR_OCT_IN	Used	Unused
DQSN_HR_OCT_IN	Used	Unused
BIDIR_DQ_HR_OCT_IN	Used	Unused
INPUT_DQ_HR_OCT_IN	Used	Unused
OUTPUT_DQ_HR_OCT_IN	Used	Unused
OCT_REG_CLK	Used	Used
HR_OCT_REG_CLK	Used if controller is at half-rate	Unused
DQS_OCT_OUT	Used	Used
DQSN_OCT_OUT	Used	Used
BIDIR_DQ_OCT_OUT	Used if DQ pin is bidirectional	Used if DQ pin is bidirectional
INPUT_DQ_OCT_OUT	Used for DQ pin as input	Used if DQ pin is bidirectional
OUTPUT_DQ_OCT_OUT	Used for DQ pin as output	Used if DQ pin is bidirectional

Design Example: Implementing Half-Rate DDR2 Interface in Stratix III Devices

This section describes a design example that uses the DLL and DQ/DQS circuitry with half-rate DDR2 external memory interface in Stratix III devices. The memory interface is running at 333.333 MHz with 8-bit bidirectional DQ pins, a 1-bit output DQ pin, and a 1-bit differential DQS pin.

The design examples are available next to the ALTDLL and ALTDQ_DQS Megafunction User Guides on the Documentation: User Guides page of the Altera website.

Procedure

This example describes the following steps:

- Instantiate the ALTDLL Megafunction
- Instantiate the ALTDQ_DQS Megafunction
- Instantiate the ALTIOBUF Megafunction
- Simulate the Design

Instantiate the ALTDLL Megafunction

To instantiate the ALTDLL megafunction, perform the following steps:

1. Open the **altdll_altdq_dqs_DesignExample_ex2.zip** project and extract the **altdll_altdq_dqs_design_ex2.qar** file.
2. In the Quartus II software, open the **altdll_altdq_dqs_design_ex2.qar** file and restore the archived file into your working directory.
3. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
4. Select **Create a new custom megafunction variation**.
5. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTDLL**, and **Verilog HDL**, and type the file name as **dll_inst.v**.
6. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in Table 4–24. These parameters configure the general settings for the ALTDLL instance.

Table 4–24. General Settings (Part 1 of 2)

Settings	Value
Currently selected device family	**Stratix III**
Match project/default	Turned on
Number of Delay Chains	**10**
DQS Delay Buffer Mode	**High**
Input Clock Frequency	**333 MHz**
Turn on jitter reduction	Turned off

Table 4–24. General Settings (Part 2 of 2)

Settings	Value
DLL Phase Offset Control A **Instantiate dll_offset_control block**	Turned off
DLL Phase Offset Control B **Instantiate dll_offset_control block**	Turned off
Optional Ports **Create a dll_aload port**	Turned off
Optional Ports **Create a dll_dqsupdate port**	Turned off

7. On the **DLL Offset Controls/Optional Ports** page, specify the parameters as shown in Table 4–25.

Table 4–25. ALTDLL Parameter Settings/DLL Offset Controls/Optional Ports Settings

Settings	Value
DLL Phase Offset Control A **Instantiate dll_offset_control block**	Turned off
DLL Phase Offset Control B **Instantiate dll_offset_control block**	Turned off
Optional Ports **Create a dll_aload port**	Turned off
Optional Ports **Create a dll_dqsupdate port**	Turned off

8. Click **Finish**.

When you are prompted to add the Quartus II IP file (**.qip**) to your project, click **Yes**.

The ALTDLL instance is now generated.

9. Browse to your working directory and open the **input.txt** file.
10. Change the value of the `CBX_OUTPUT_DIRECTORY` parameter to the path of your working directory.
11. Save the file.
12. Copy the **input.txt** file to the **<*quartusii_install_dir*>\quartus\bin** directory.

Instantiate the ALTDQ_DQS Megafunction

To instantiate the ALTDQ_DQS megafunction, perform the following steps:

1. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
2. Select **Create a new custom megafunction variation**.
3. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears. Select **ALTDQ_DQS**, and **Verilog HDL**, and type the file name as **dq_dqs_inst.v**.
4. On the **Parameter Settings** page of the ALTDQ_DQS parameter editor, specify the parameters as shown in Table 4–26.

Table 4–26. Parameter Settings

Parameter	Value
RLDRAM II Mode	**NONE**
Data mask pin group	**NONE**
Q valid signal group	**NONE**
Number of bidirectional DQ	**8**
Number of input DQ	**0**
Number of output DQ	**1**
Number of stages in dqs_delay_chain	**2**
DQS Input Frequency	**333 MHz**
Use half-rate components	Turned on
Use Dynamic OCT	Turned off
Add memory interface specific fitter grouping assignments	Turned off

5. In the **Advanced Options** tab of the ALTDQ_DQS parameter editor, on the **DQS IN** page, specify the parameters as shown in Table 4–27. These parameters configure the DQS input path of the ALTDQ_DQS instance.

Table 4–27. Advanced Options (DQS IN) (Part 1 of 2)

Parameter	Sub-options	Value
Enable DQS Input Path	—	Turned on
Enable Dynamic Delay Chain	—	Not selected
Enable dqs_delay_chain	—	Selected
Advanced delay chain options	**Select dynamically using configuration registers**	Turned off
	DQS delay chain 'delayctrlin' port source	**DLL**
	DQS Delay Buffer Mode	**HIGH**
	DQS Phase Shift	**9000..**
	Enable DQS offset Control	Turned off
	Enable DQS delay chain latches	Turned off
Enable DQS busout delay chain	—	Turned on
Enable DQS enable block	—	Turned on
Enable DQS enable control block	—	Turned on

Table 4–27. Advanced Options (DQS IN) (Part 2 of 2)

Parameter	Sub-options	Value
Advanced enable control options	**DQS Enable Control Phase setting**	**Set Statically to '0'**
	DQS Enable Control Invert Phase	**Never**
Enable DQS enable block delay chain	—	Turned on

6. On the **DQS OUT/OE** page, specify the parameters as shown in Table 4–28. These parameters configure the DQS OUTPUT and DQS OE path of the ALTDQ_DQS instance.

Table 4–28. Advance Options (DQS OUT/OE)

Parameter	Value
Enable DQS Output Path	Turned on
Enable DQS output delay chain1	Turned on
Enable DQS output delay chain2	Turned on
]DQS output register mode	**DDIO**
Enable DQS output enable	Turned on
Enable DQS output enable delay chain1	Turned on
Enable DQS output enable delay chain2	Turned on
DQS output enable register mode	**DDIO**

7. On the **DQ IN** page, specify the parameters as shown in Table 4–29. These parameters configure the DQ input path of the ALTDQ_DQS instance.

Table 4–29. Advance Options (DQ IN) (Part 1 of 2)

Parameter	Sub-options	Value
DQ input register mode	—	**DDIO**
DQ Input Register Options	—	**'dqs_bus_out' port**
DQ input register clock source	—	Turned off **Connect DDIO clkn to DQS_BUS from complementary DQSn**
Use DQ input phase alignment	—	Turned on
Advanced DQ IPA Options	**DQ Input Phase Alignment Phase Setting**	**Set statically to '0'**
	Add DQ Input Phase Alignment Input Cycle Delay	**Never**
	Invert DQ Input Phase Alignment Phase	**Never**
	Register DQ input phase alignment bypass output	Turned on
	Register DQ input phase alignment add phase transfer	Turned off
Use DQ resync register	—	Turned off

Table 4–29. Advance Options (DQ IN) (Part 2 of 2)

Parameter	Sub-options	Value
Use DQ half rate 'dataoutbypass' port	—	Turned off
Use DQ input delay chain	—	Turned on

8. On the **DQ OUT/OE** page, specify the parameters as shown in Table 4–30. These parameters configure the DQ OUTPUT and DQ OE path of the ALTDQ_DQS instance.

Table 4–30. Advance Options (DQ OUT/OE)

Parameter	Value
Enable DQ output delay chain1	Turned on
Enable DQ output delay chain2	Turned on
DQ output register mode	**DDIO**
Enable DQ output enable	Turned on
Enable DQ output enable delay chain1	Turned on
Enable DQ output enable delay chain2	Turned on
DQ output enable register mode	**DDIO**

9. On the **Half-rate** page, specify the parameters as shown in Table 4–31. These parameters configure the half-rate settings of the ALTDQ_DQS instance.

Table 4–31. Advance Options (Half-Rate)

Parameter	Value
IO Clock Divider Source	**Core**
Create 'io_clock_divider_masterin' input port	Turned off
Create 'io_clock_divider_clkout' output port	Turned on
Create 'io_clock_divider_slaveout' output port	Turned off
IO Clock Divider Invert Phase	**Never**

10. On the **DQSn I/O** page, specify the parameters as shown in Table 4–32. .

Table 4–32. Advanced Options (DQS/DQSn IO)

Parameter	Value
Use DQSn IO	Turned on
DQS and DQSn IO Configuration mode	**Differential Pair**

11. On the **Reset/Config Ports** page, specify the parameters as shown in Table 4–33.

Table 4–33. Advanced Options (Reset and Config Ports) (Part 1 of 2)

Parameter	Value
Create 'dqs_areset' input port	Turned on
Create 'dqs_sreset' input port	Turned on
Create 'input_dq_areset' input port	Turned off
Create 'input_dq_sreset' input port	Turned off
Create 'output_dq_areset' input port	Turned off

Table 4–33. Advanced Options (Reset and Config Ports) (Part 2 of 2)

Parameter	Value
Create 'output_dq_sreset' input port	Turned off
Create 'bidir_dq_areset' input port	Turned on
Create 'bidir_dq_sreset' input port	Turned on
Create 'config_clk' input port	Turned on
Create 'config_datain' input port	Turned on
Create 'config_update' input port	Turned on

12. Click **Finish**. The dq_dqs_inst module (**dq_dqs_inst.v)** is generated.

Instantiate the ALTIOBUF Megafunction

After instantiating the ALTDLL and ALTDQ_DQS megafunctions, you must instantiate the ALTIOBUF megafunction with the following I/O buffer settings:

- 1 bidirectional buffer for the differential DQS pins
- 1 output buffer for the output DQ pins
- 8 bidirectional buffers for the bidirectional DQ pins

To instantiate these three types of I/O buffers, perform the following steps:

1. In the Quartus II software, on the Tools menu, click **MegaWizard Plug-In Manager**.
2. On page 1, select **Create a new custom megafunction variation**. Click **Next**. Page 2a appears.
3. On page 2a, select or verify the configuration settings shown in Table 4–34. Click **Next** to advance from one page to the next.

Table 4–34. ALTIOBUF Configuration Settings

Settings	Value		
	1 bidirectional buffer for the differential DQS pins	**1 output buffer for the output DQ pins**	**8 bidirectional buffers for the bidirectional DQ pins**
Which device family will you be using?	**Stratix III**	**Stratix III**	**Stratix III**
Which megafunction would you like to customize?	**ALTIOBUF**	**ALTIOBUF**	**ALTIOBUF**
Which type of output file do you want to create?	**Verilog HDL**	**Verilog HDL**	**Verilog HDL**
What name do you want for the output file?	**dqs_iobuf_inst.v**	**output_dq_iobuf_inst.v**	**bidir_dq_iobuf_inst.v**

4. On the **Parameter Settings** page, specify the parameters as shown in Table 4–35. These parameters configure the general settings for the ALTIOBUF instance.

Table 4–35. ALTIOBUF General Settings

Settings	Value		
	1 bidirectional buffer for the differential DQS pins	1 output buffer for the output DQ pins	8 bidirectional buffers for the bidirectional DQ pins
Currently selected device family	Stratix III	Stratix III	Stratix III
How do you want to configure this module?	As bidirectional buffer	As output buffer	As bidirectional buffer
What is the number of buffers to be instantiated?	1	1	8
Use bus hold circuitry	Turned off	Turned off	Turned off
Use differential mode	Turned on	Turned off	Turned off
Use open drain output	Turned off	Turned off	Turned off
Use output enable port	Turned off	Turned on	Turned on
Use dynamic termination control	Turned off	Turned off	Turned off
Use series and parallel termination control	Turned off	Turned off	Turned off

5. On the **Dynamic Delay Chains** page, specify the parameters as shown in Table 4–36.

Table 4–36. ALTIOBUF Dynamic Delay Chain Settings

Settings	Value		
	1 bidirectional buffer for the differential DQS pins	1 output buffer for the output DQ pins	8 bidirectional buffers for the bidirectional DQ pins
Enable input buffer dynamic delay chain	Turned off	Turned off	Turned off
Enable output buffer dynamic delay chain 1	Turned off	Turned off	Turned off
Enable output buffer dynamic delay chain 2	Turned off	Turned off	Turned off
Create a 'clkena' port	Turned off	Turned off	Turned off

6. Click **Finish**. The I/O buffer module (**dqs_iobuf_inst.v/output_dq_iobuf_inst.v/bidir_dq_iobuf_inst.v**) is generated.
7. On the File menu, click **Save**.

Integrate the I/O Buffer Modules with the ALTDQ_DQS modules

To integrate the I/O buffer modules with the ALTDQ_DQS modules, perform the following steps:

1. Open the **test_dq_dqs.bdf** file in the Quartus II Block Editor software.
2. To insert the I/O buffer modules, double-click on the Block Editor window. The Symbol window appears.
3. Under **Name**, browse to the I/O buffer **dqs_iobuf_inst.bsf** file.
4. Click **OK**. The I/O buffer module is inserted into the Block Editor window.

5. Repeat steps 1 to 4 to insert other I/O buffer modules.
6. Use the appropriate connectors from the Block Editor toolbar to connect the I/O buffer modules to the **dq_dqs_inst.v** module as shown in Figure 4–24.

For more information about the Quartus II Block Editor, refer to "Using the Block Editor" in the Quartus II Help.

Figure 4–24 shows a block diagram of the design example, which consists of six blocks.

Figure 4–24. Block Diagram of Design Example

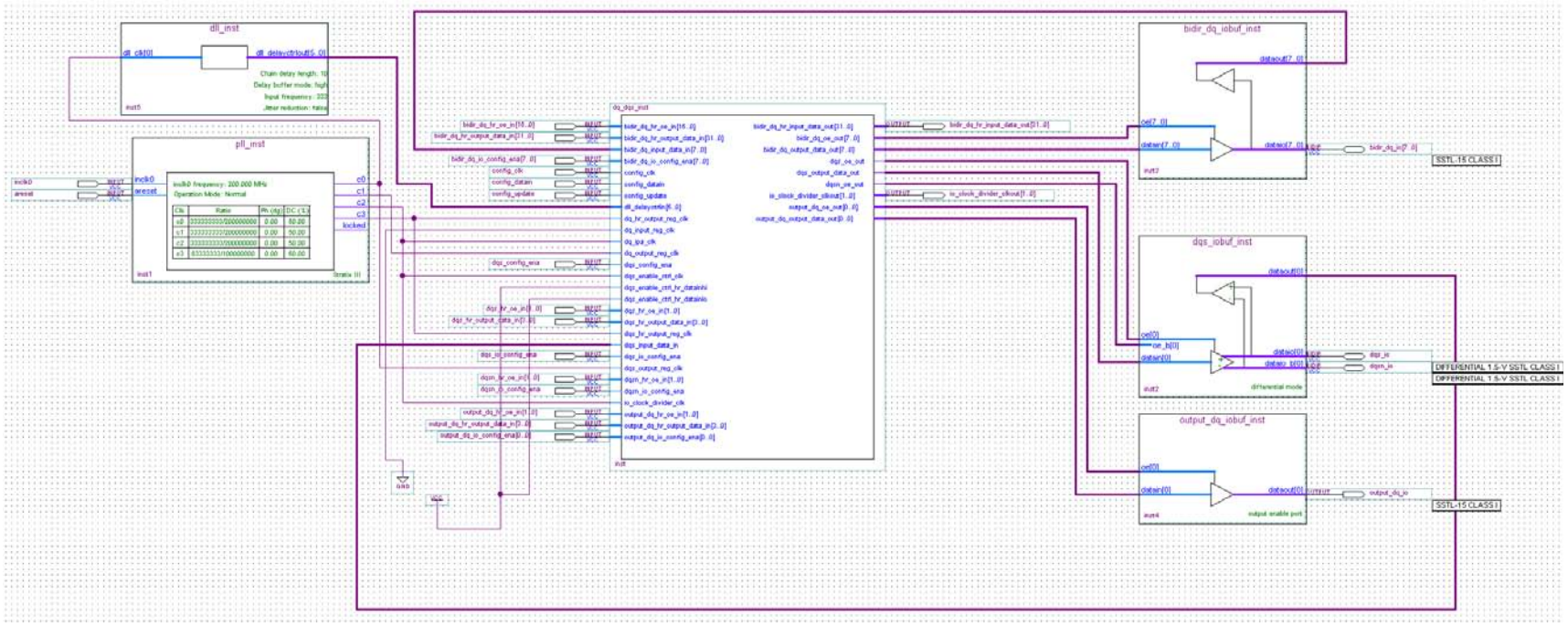


Table 4–37 provides the description for each block in the design example.

Table 4–37. Blocks in Design Example

Block Name	Description
`pll_inst:inst1`	This block represents the Stratix III PLL with the following settings: ■ `inclk` = 200 MHz ■ `c0` = 3,000 ps, 50% duty cycle ■ `c1` = 3,000 ps, 50% duty cycle ■ `c2` = 3,000 ps, 50% duty cycle ■ `c3` = 6,000 ps, 50% duty cycle
`dll_inst:inst5`	This block represents the DLL circuitry used during a read from the external memory. This block is clocked by the PLL with the following settings: ■ delay chain length = **10** ■ delay buffer mode = **High** ■ input frequency = **333** MHz ■ jitter reduction = Turned off
`dq_dqs_inst:inst`	This block represents the DQ and DQS circuitry that interfaces with the external memory. The settings are specified in the **input.txt** file. The block is customized for a half-rate operation and represents the interface between the FPGA core and the I/O buffers that are connected to the external memory pins.
`dqs_iobuf_inst:inst2`	This block represents the bidirectional I/O buffer that is used as the DQS strobe/clock signal for interfacing with the external memory. This block is in differential mode and is 1 bit wide. It is connected to the `dq_dqs_inst` block.
`bidir_dq_iobuf_inst:inst3`	This block represents the bidirectional I/O buffer that is used as the DQ data signals for interfacing with the external memory. This block is 8 bits wide. It is connected to the `dq_dqs_inst` block.
`output_dq_iobuf_inst:inst4`	This block represents the output I/O buffer that is used as the DQ data signals for interfacing with the external memory. This block is 1 bit wide. It is connected to the `dq_dqs_inst` block.

Simulate the Design

After instantiating the megafunctions, perform the following steps to compile your design.

1. In the Quartus II software, on the Project menu, click **Add/Remove Files in Project**.
2. In the **Category** list, select **Files**.
3. Next to the **File name** box, click ... to browse to your working directory. Select the **dll_inst.v** file and click **Open**.
4. Click **Add** to add the **dll_inst.v** file to your project.
5. Repeat steps 3 and 4 to add the **dq_dqs_inst.v** and **test_dq_dqs.bdf** files.
6. Click **OK**.
7. On the File menu, click **Save**.
8. On the Processing menu, click **Start Compilation** to compile the design. After the design is compiled, you can view implementation in the RTL Viewer. You can also view the resource usage in the Compilation Report.

After you compile your design, simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior. Set up and simulate the design in the ModelSim-Altera software by performing the following steps:

1. Unzip the **altdll_altdq_dqs_ex2_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**.
6. On the Tools menu, point to **TCL** and click **Execute Macro**.
7. Select the **altdll_altdq_dqs_ex2_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.
8. Verify the results with the waveform.

You can rearrange signals, remove signals and add signals, and change the radix by modifying the script in the **altdll_altdq_dqs_ex2_msim.do** file.

Understanding the Simulation Results

This section describes the simulation results of "Design Example: Implementing Half-Rate DDR2 Interface in Stratix III Devices" on page 4–49.

Writing Data to the External Memory

The following sequence describes the transferring of data from the FPGA core to the bidirectional DQ pins with various delay chain settings (refer to Figure 4–25 on page 4–63):

1. The simulation begins when the PLL is locked, as indicated by the assertion of the `locked` signal at 225,000 ps (refer to Figure 4–25). At this point, the PLL input frequency, as indicated by the `inclk0` signal, is 200 MHz.
2. The `c0`, `c1`, and `c2` ports generate a 333.333-MHz clock output while the `c3` port generates a 166.666-MHz clock output.

   This design example uses the half-rate option, which means that the FPGA core sends and receives data from the external memory interface at a half-rate of 166.666 MHz. The pin that interfaces with the memory toggles at 333.333 MHz. However, because this pin is also toggled by a `DDIO_OUT` signal, the data throughput is 666.666 Mbps.

3. The output path from the FPGA core to the bidirectional DQ pin is represented by a 32-bit input, `bidir_dq_hr_output_data_in[31:0]`. The input path from the bidirectional pin to the FPGA core is represented by a 32-bit output, `bidir_dq_hr_input_data_out[31:0]`. The OE path from the FPGA core to the bidirectional buffer, `bidir_dq_hr_oe_in[15:0]`, is 16 bits wide and is active-low.
4. For the DQ output pin, the output path in the FPGA core to the bidirectional DQ pin is represented by a 4-bit input, `output_dq_hr_output_data_in [3:0]`. The OE path is 2 bits wide from the FPGA core to the bidirectional buffer, `output_dq_hr_oe_in[1:0]`.

   In the first part of the simulation, only output paths are used; therefore, `bidir_dq_hr_oe_in[15:0]` = 16'b0 and `dqs_hr_oe_in [1:0]` = 2'b0.

5. For `bidir_dq_hr_output_data_in[31:0]`, each bit is toggled with a 10-MHz data signal from 100 ns to 300 ns. The toggling behavior of `bidir_dq_hr_output_data_in[31:0]`is represented in the waveform in groups of 4-bit signals (for example, `bidir_dq_hr_output_data_in[3:0]`), as the four input paths are connected to the `bidir_dq_io[0]` pin.
6. The `bidir_dq_hr_output_data_in[3]` and `bidir_dq_hr_output_data_in[2]`signals go through the `DDIO_OUT` port, which is clocked at 166.666 MHz by the `c3` PLL clock output. At the same time, the `bidir_dq_hr_output_data_in[1]` and `bidir_dq_hr_output_data_in[0]` signals go through another `DDIO_OUT` port, which is clocked at 166.666 MHz by the `c3` PLL clock output.

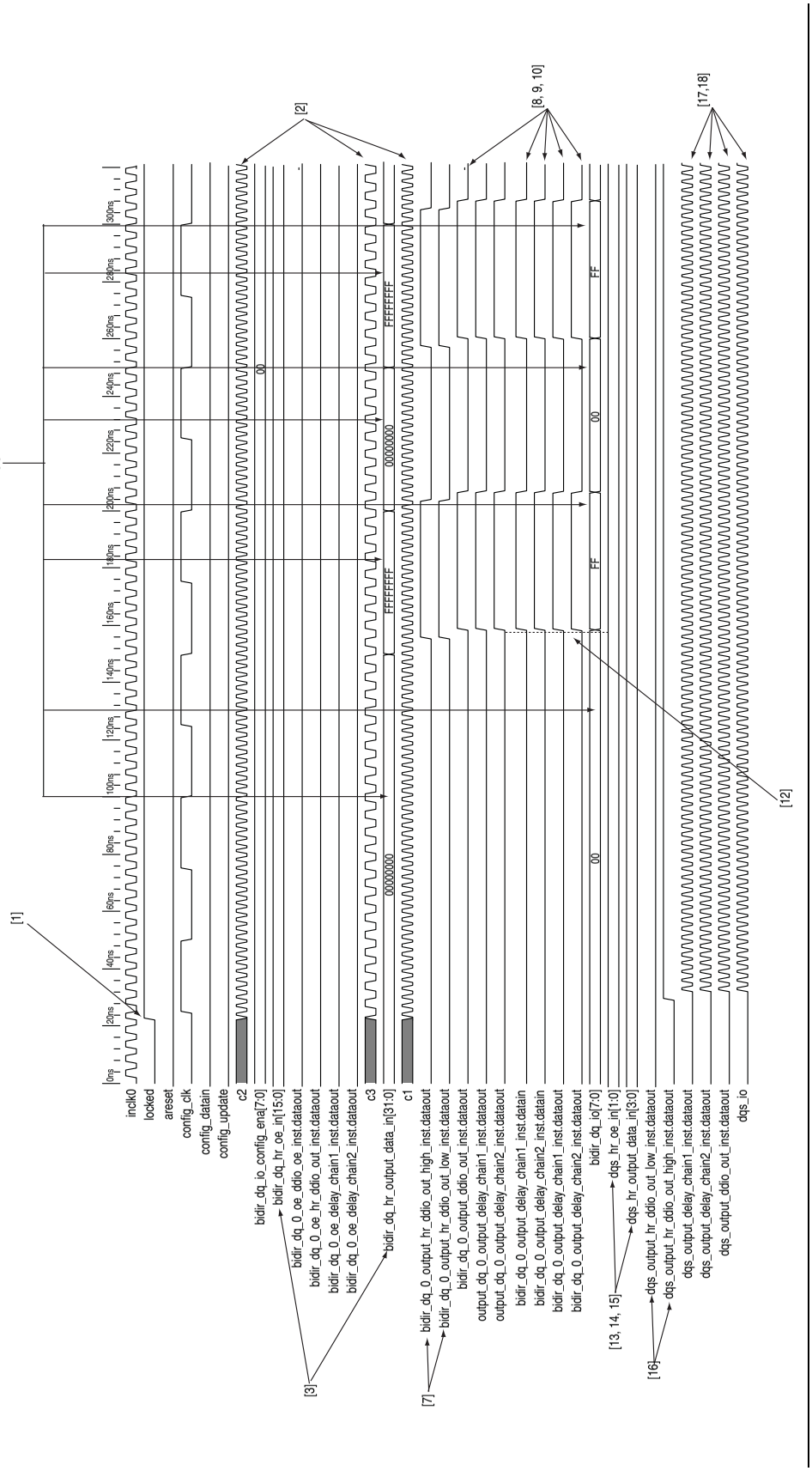
7. Both outputs (`bidir_dq_0_output_hr_ddio_out_high_inst/dataout` and `bidir_dq_0_output_hr_ddio_out_low_inst/dataout`) of the previous `DDIO_OUT` ports are channeled into another `DDIO_OUT` port, which is clocked at 333.333 MHz by the `c1` PLL clock output.
8. The output `bidir_dq_0_output_ddio_out_inst/dataout` is then connected to the bidirectional DQ output delay chain 1.
9. The output `bidir_dq_0_output_delay_chain1_inst/dataout` is connected to the bidirectional DQ output delay chain 2, and the output `bidir_dq_0_output_delay_chain2_inst/dataout` is connected to the `bidir_dq_io[0]` pin.
10. The same data is propagated through the other inputs of `bidir_dq_hr_output_data_in[31:4]`, which causes the `bidir_dq_io[7:1]` pins to toggle in the same manner.
11. The throughput of data going out on each pin to the external memory is 666.666 Mbps.
12. The output delay chains are disabled. The `bidir_dq_0_output_delay_chain1_inst/datain`, `bidir_dq_0_output_delay_chain2_inst/datain`, `bidir_dq_0_output_delay_chain1_inst/dataout`, and `bidir_dq_0_output_delay_chain2_inst/dataout signals are aligned, which indicates that there's no delay settings on the two output delay chains.`
    The same write sequence applies to writing data with different delay chain values activated on the two output delay chains. You can obtain the difference in the delay chain values by analyzing the timing paths of the following signals:
    - `bidir_dq_0_output_delay_chain1_inst/datain`
    - `bidir_dq_0_output_delay_chain2_inst/datain`
    - `bidir_dq_0_output_delay_chain1_inst/dataout`
    - `bidir_dq_0_output_delay_chain2_inst/dataout`
    - `bidir_dq_0_output_hr_ddio_out_high_inst/dataout`
    - `bidir_dq_0_output_hr_ddio_out_low_inst/dataout`
    - `bidir_dq_0_output_ddio_out_inst/dataout`
    - `bidir_dq_io[0]`

    
    For more information about how to analyze the timing paths to obtain the delay chain values, refer to the timing diagrams in "DQS_CONFIG / IO_CONFIG Block" on page 4–22.
13. The output path from the FPGA core to the bidirectional DQS pin is represented by a 4-bit input, `dqs_hr_output_data_in[3:0]`. The OE path is 2 bits wide from the FPGA core to the bidirectional buffer, `dqs_hr_oe_in [1:0]`. The input path of the DQS pin goes through a specialized circuitry to clock the 8-bit bidirectional DQ pin input paths.

14. The dqs_hr_output_data_in[3:0], dqs_hr_output_data_in[3] and dqs_hr_output_data_in[2] signals are toggled with a constant value of 1'b1. After that, the dqs_hr_output_data_in[1] and dqs_hr_output_data_in[0] signals are toggled with a constant value of 1'b0. The signals are toggled at a constant rate to generate the necessary DQS write strobe/clock signals, which are sent together with the DQ write data to the external memory.
15. As the throughput of the data is sent at 666.666 Mbps, the DQS write strobe/clock signal is a 333.333-MHz DDR clock signal. To obtain such a signal, the dqs_hr_output_data_in[3] and dqs_hr_output_data_in[2] signals go through a DDIO_OUT port, which is clocked at 166.666 MHz by the c3 PLL clock output. At the same time, the dqs_hr_output_data_in[1] and dqs_hr_output_data_in[0] signals go through another DDIO_OUT port, which is clocked at 166.666 MHz by the c3 PLL clock output.
16. Both outputs (dqs_output_hr_ddio_out_high_inst/dataout and dqs_output_hr_ddio_out_low_inst/dataout) of the previous DDIO_OUT ports are channeled into another DDIO_OUT port, which is clocked at 333.333 MHz by the c1 PLL clock output.
17. The output dqs_output_ddio_out_inst/dataout is then connected to output_delay_chain_1. The output dqs_output_delay_chain1_inst/dataout is connected to output_delay_chain_2.
18. The output dqs_output_delay_chain2_inst/dataout is connected to the dqs_io pin, which acts as a 333.333-MHz DQS write strobe/clock signal.

For details about changing the delay chain values dynamically, refer to the *I/O Buffer (ALTIOBUF) Megafunction User Guide*.

Figure 4–25. Data Transfer From the FPGA Core to the Bidirectional DQ Pin with No Delay Chains Activated



Reading Data from the External Memory

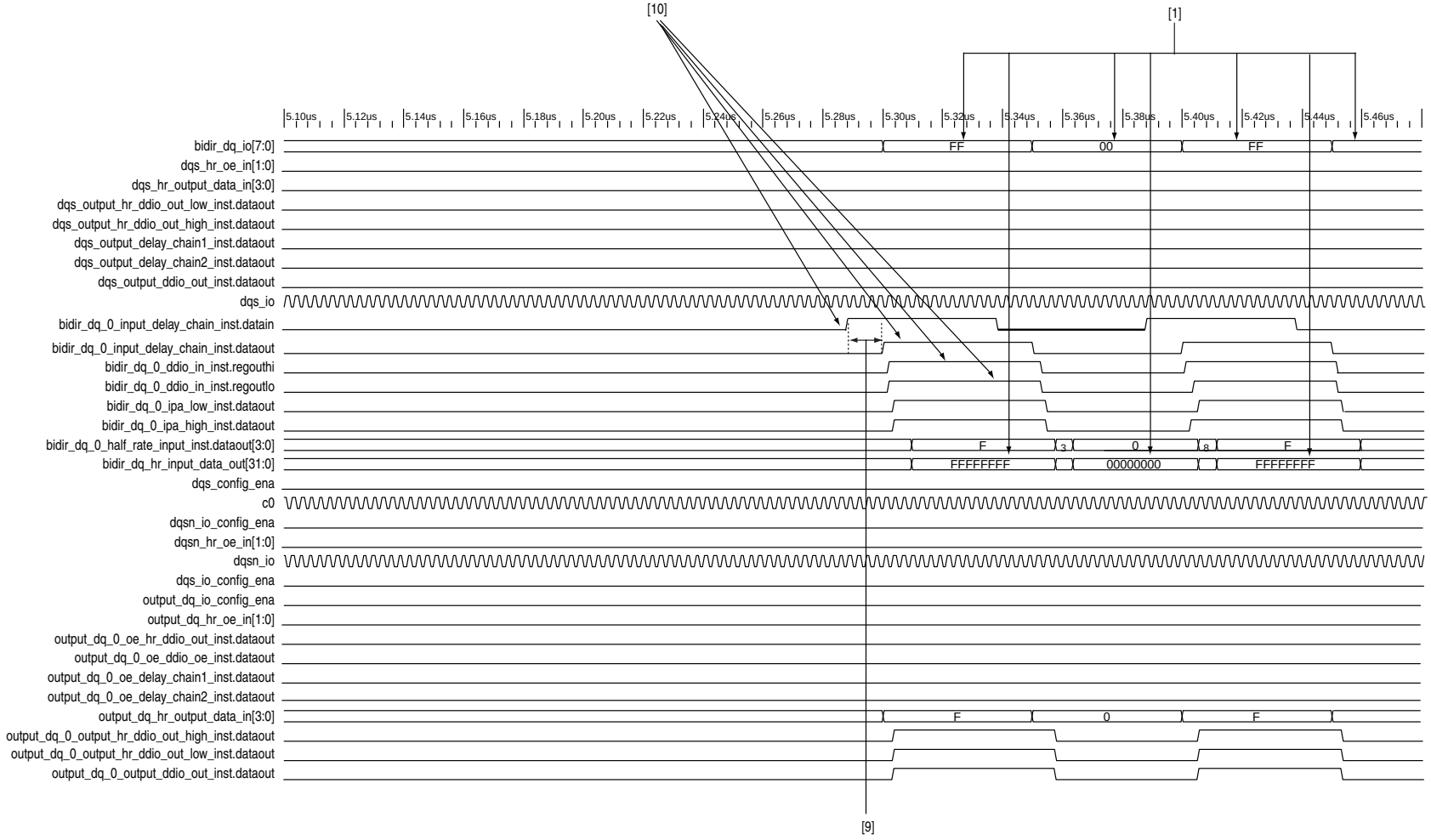
The following sequence describes the transferring of data from the bidirectional DQ pins to the FPGA core with various delay chain settings (refer to Figure 4–26 on page 4–65):

The interface to the external memory has a throughput of 666.666 Mbps during the read process.

In Figure 4–26, only the input paths are used; therefore, `bidir_dq_hr_oe_in[15:0]` = 16'b1 and `dqs_hr_oe_in[1:0]` = 2'b1 from 5 µs onwards.

1. Each bit in the `bidir_dq_io[7:0]` pin is toggled with a 10-MHz data signal from 5.25 µs to 5.45 µs. The pin behavior is represented in the waveform in groups of 4-bit signals because the `bidir_dq_io[0]` input is connected to the `bidir_dq_hr_input_data_out[3:0]` outputs.
2. The `bidir_dq_io[0]` pin is connected to the input delay chain.
3. The output `bidir_dq_0_input_delay_chain_inst/dataout` of the delay chain is connected to the input of the `DDIO_IN` port, which is clocked by a specialized DQS circuitry that uses the DLL.
4. The outputs (`bidir_dq_0_ddio_in_inst/regouthi` and `bidir_dq_0_ddio_in_inst/regoutlo`) of the previous `DDIO_IN` ports are channeled to two input phase alignment blocks, respectively. These input phase alignment blocks are clocked at 333.333 MHz by the `c2` clock output of the PLL.
5. The outputs of the two IPAs, `bidir_dq_0_ipa_high_inst/dataout` and `bidir_dq_0_ipa_low_inst/dataout`, are channeled to a half-rate input block, which is clocked by the `IO_CLOCK_DIVIDER` blocks.
6. The output `bidir_dq_0_half_rate_input_inst/dataout[3:0]` of this block is then connected to the `bidir_dq_hr_input_data_out[3:0]` outputs.
7. The same data is propagated through the other bidirectional pins of `bidir_dq_io[7:1]`, which causes the `bidir_dq_hr_input_data_out[31:4]` outputs to toggle in the same manner.
8. The throughput of the data in the output ports are at a half-rate of 166.666 MHz.
9. The input delay chain is enabled.
   The `bidir_dq_0_input_delay_chain_inst/datain` and `bidir_dq_0_input_delay_chain_inst/dataout` signals are not aligned, which indicates that there is a delay on the input delay chain.
   The same read sequence applies to reading data with different chain values activated on the input delay chain. You can obtain the difference in the delay chain values by analyzing the timing paths of the following signals:
   - `bidir_dq_io[0]`
   - `bidir_dq_0_input_delay_chain_inst/datain`
   - `bidir_dq_0_input_delay_chain_inst/dataout`
   - `bidir_dq_0_ddio_in_inst/regouthi`
   - `bidir_dq_0_ddio_in_inst/regoutlo`

Figure 4–26. Data Transfer from the Bidirectional DQ Pin to the FPGA Core with 50-ps Delay Chain Activated



Using Clear Box Generator

You can use the clear box generator, a command-line executable, to configure parameters that are not available in the ALTDQ_DQS parameter editor. The clear box generator creates or modifies design files that contain custom megafunction variations, which can then be instantiated in a design file.

To run the clear box generator, perform the following steps:

1. Type the following command at the command prompt of your operating system:

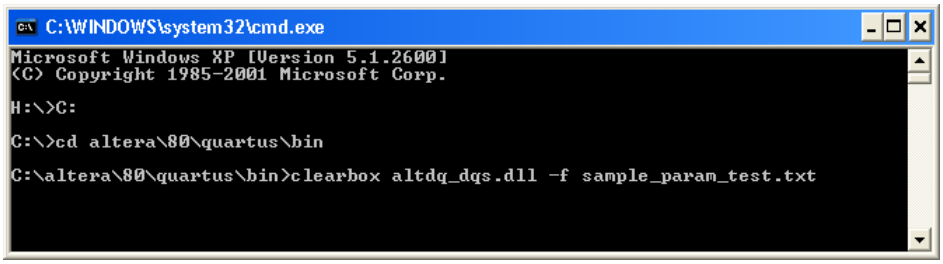
   <*quartusii_install_dir*>`\quartus\bin\`

2. The executable name is **clearbox.exe**. To use the executable, type the following command:

   ```
   clearbox altdq_dqs.dll -f *.txt
   ```

   where ***.txt** represents one or more text files containing the ports and parameters that you want to generate, refer to Figure A–1.

Figure A–1. Accessing the Clear Box Generator



Clear Box Generator Options

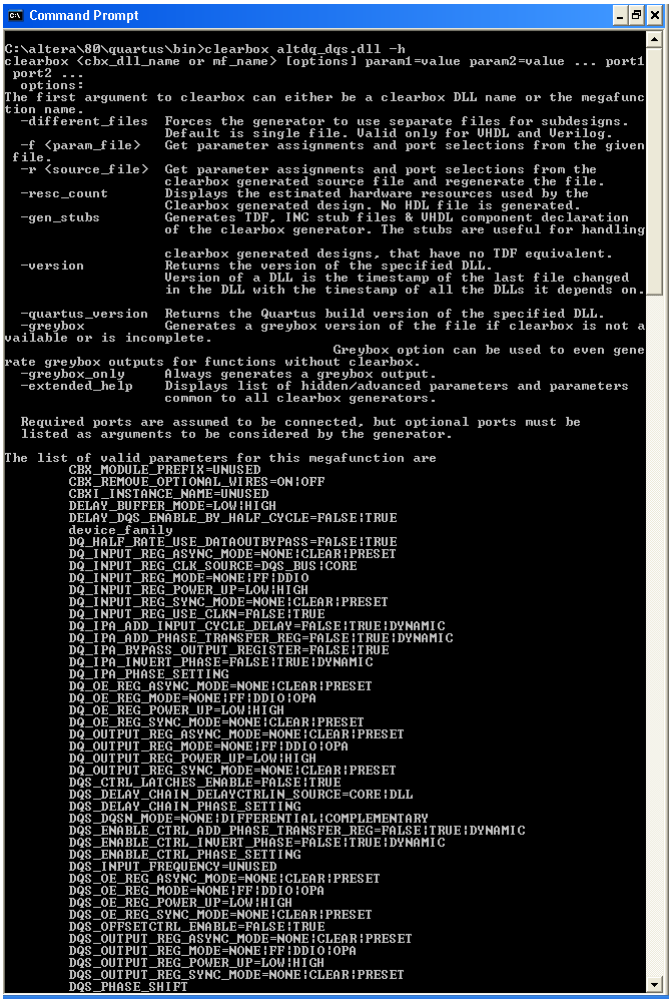
This section describes the options available when you generate the ALTDQ_DQS megafunction with the clear box generator.

To find out the available ports and parameters for this megafunction, type the following command at the command prompt of your operating system:

```
clearbox altdq_dqs.dll -h
```

Figure A–2 shows a sample listing of the available ports and parameters for the ALTDQ_DQS megafunction.

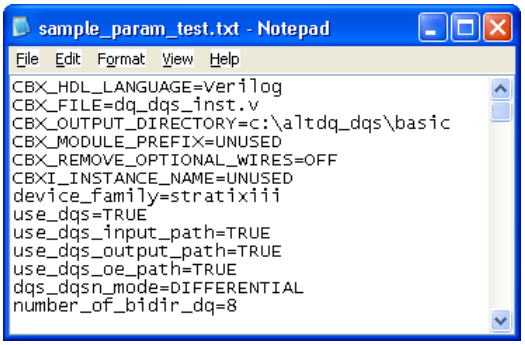
Figure A–2. Available Ports and Parameters for the ALTDQ_DQS Megafunction



To efficiently generate output files for the ALTDQ_DQS megafunction, Altera recommends that you use a text file to pass the required ports and parameters to the clear box generator. This method promotes reusability and provides an easier way to customize the megafunction.

Figure A–3 shows a sample text file used for the clear box generator.

Figure A–3. Sample Text File for Clear Box Generator



With the text file, you can generate output files using the following command:

```
clearbox altdq_dqs.dll –f sample_param_test.txt
```

After the output files are generated, you can instantiate the megafunction module into either a HDL file or a block diagram file in the Quartus II software.

To determine the resource usage for a particular configuration in the ALTDQ_DQS megafunction, type the following command:

```
clearbox altdq_dqs.dll -f sample_param_test.txt -resc_count
```

Clear Box Parameters

Table A–1 lists the Clear Box parameters for the ALTDLL megafunction.

Table A–1. ALTDLL Megafunction Parameters (Part 1 of 3)

Parameter Name	Type	Required	Description
DELAY_BUFFER_MODE	String	No	Determines whether the DLL delay buffers are working in low-frequency mode or high-frequency mode. Available values are LOW and HIGH. The default value is LOW.
DELAY_CHAIN_LENGTH	Integer	No	This parameter represents the number of delay buffers in the delay loop. The available values are 6, 8, 10, 12, and 16. This parameter defaults to 12.
DLL_OFFSET_CTRL_A_ STATIC_OFFSET	String	No	This is a Gray-coded signed integer expressed as a string with a range from –63 to 63. If the DLL_OFFSET_CTRL_A_USE_OFFSET parameter is set to FALSE, the value is added to the DLL delay-setting value and appears as output on the dll_offset_ctrl_a_offsetctrlout[5..0] output bus. If the DLL_OFFSET_CTRL_A_USE_OFFSET parameter is set to TRUE, ignore this value. The default value is 0.

Table A–1. ALTDLL Megafunction Parameters (Part 2 of 3)

Parameter Name	Type	Required	Description
DLL_OFFSET_CTRL_A_USE_OFFSET	String	No	Available values are TRUE and FALSE. It determines the output of the dll_offset_ctrl_a_offsetctrlout[5..0] output bus. If set to TRUE, then depending on whether the dll_offset_ctrl_a_addnsub input is asserted or not, the phase offset specified on the dll_offset_ctrl_a_offset[5..0]input bus is added or subtracted from the DLL delay setting output to get the dll_offset_ctrl_a_offsetctrlout[5..0] output. If set to FALSE, the phase offset specified by the DLL delay setting to get the dll_offset_ctrl_a_offsetctrlout[5..0] output. If omitted, the default is FALSE.
DLL_OFFSET_CTRL_B_STATIC_OFFSET	String	No	This is a Gray-coded signed integer expressed as a string with a range from –63 to 63. If the DLL_OFFSET_CTRL_B_USE_OFFSET parameter is set to FALSE, the value is added to the DLL delay-setting value and appears as output on the dll_offset_ctrl_a_offsetctrlout[5..0] output bus. If the DLL_OFFSET_CTRL_B_USE_OFFSET parameter is set to TRUE, ignore this value. The default value is 0.
DLL_OFFSET_CTRL_B_USE_OFFSET	String	No	Available values are TRUE and FALSE. It determines the output of the dll_offset_ctrl_b_offsetctrlout[5..0] output bus. If set to TRUE, then depending on whether the dll_offset_ctrl_b_addnsub input is asserted or not, the phase offset specified on the dll_offset_ctrl_b_offset[5..0]input bus is added or subtracted from the DLL delay setting output to get the dll_offset_ctrl_b_offsetctrlout[5..0] output. If set to FALSE, the phase offset specified by the DLL delay setting to get the dll_offset_ctrl_b_offsetctrlout[5..0] output. If omitted, the default is FALSE
INPUT_FREQUENCY	String	Yes	This is the frequency of the clock connected to the clk input port. Check this parameter value to ensure that it falls within a valid range. This field is required and defaults to 0. You can specify a duration by placing a time unit after the value (for example, 2.5 ns). The value is in floating-point format with no decimal point limit.
JITTER_REDUCTION	String	No	Available values are TRUE and FALSE. If set to TRUE, the jitter reduction circuit is enabled on the dll_delayctrlout[5..0], and dll_offset_ctrl_a_offsetctrlout[5..0]or dll_offset_ctrl_b_offsetctrlout[5..0] outputs and the DLL may require up to 1,024 clock cycles to lock. If set to FALSE, the jitter reduction circuit is disabled and the DLL only requires up to 256 clock cycles to lock. If omitted, the default is FALSE.

Table A–1. ALTDLL Megafunction Parameters (Part 3 of 3)

Parameter Name	Type	Required	Description
USE_DLL_OFFSET_CTRL_A	String	No	Specifies whether to instantiate DLL_OFFSET_CTRL_A block. Values are TRUE and FALSE. If omitted, the default is FALSE.
USE_DLL_OFFSET_CTRL_B	String	No	Specifies whether to instantiate DLL_OFFSET_CTRL_B block. Values are TRUE and FALSE. If omitted, the default is FALSE.

Table A–2 lists the high-level Clear Box parameters for the ALTDQ_DQS megafunction.

Table A–2. ALTDQ_DQS Megafunction High-Level Configurations

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQS	Optional	FALSE	FALSE, TRUE	Instantiates DQS IO if TRUE
USE_DQS_INPUT_PATH	Optional	FALSE	FALSE, TRUE	Instantiates DQS input path if TRUE
USE_DQS_OUTPUT_PATH	Optional	FALSE	FALSE, TRUE	Instantiates DQS output path if TRUE
USE_DQS_OE_PATH	Optional	FALSE	FALSE, TRUE	Instantiates DQS OE path if TRUE
DQS_DQSN_MODE	Optional	NONE	NONE, DIFFERENTIAL, COMPLEMENTARY	Instantiates DQS and DQSn I/Os as a differential pair if DIFFERENTIAL, Instantiate DQS and DQSn I/Os as a complementary pair if COMPLEMENTARY, Instantiate only a DQS IO if NONE
NUMBER_OF_BIDIR_DQ	Optional	0	0, 1, 2 .. 48	Referred as n_b
NUMBER_OF_OUTPUT_DQ	Optional	0	0, 1, 2 .. 48	Referred as n_o
NUMBER_OF_INPUT_DQ	Optional	0	0, 1, 2 .. 48	Referred as n_i
USE_DQ_OE_PATH	Optional	FALSE	FALSE, TRUE	Instantiates OE path for OUTPUT_DQ IOs
USE_HALF_RATE	Optional	FALSE	FALSE, TRUE	Instantiates half-rate components
USE_DYNAMIC_OCT	Optional	FALSE	FALSE, TRUE	Instantiates dynamic OCT components
NUMBER_OF_CLK_DIVIDER	Optional	0	0,1, 2 .. 8	Referred as n_c

Table A–3 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQS input path.

Table A–3. Megafunction Parameters to Configure DQS Input Path (Part 1 of 4)

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQS_INPUT_DELAY_CHAIN	Optional	FALSE	FALSE, TRUE	Instantiates DQS_INPUT_DELAY_CHAIN if TRUE.
USE_DQS_DELAY_CHAIN	Optional	FALSE	FALSE, TRUE	Instantiates DQS_DELAY_CHAIN if TRUE.
USE_DQS_ENABLE	Optional	FALSE	FALSE, TRUE	Instantiates DQS_ENABLE if TRUE.
USE_DQS_ENABLE_CTRL	Optional	FALSE	FALSE, TRUE	Instantiates DQS_ENABLE_CTRL if TRUE.
USE_DQSBUSOUT_DELAY_CHAIN	Optional	FALSE	FALSE, TRUE	Instantiates DQSBUSOUT_DELAY_CHAIN if TRUE.
USE_DQSENABLE_DELAY_CHAIN	Optional	FALSE	FALSE, TRUE	Instantiates DQSENABLE_DELAY_CHAIN if TRUE.
DQS_INPUT_FREQUENCY	Optional	UNUSED	—	This parameter is set to the frequency of the DQS strobe/clock input expressed as a string and, if *<phase_setting>* is not set to 0, and *<use_phasectrlin>* is not set to FALSE, this needs to match the input_frequency parameter of the DLL feeding the delayctrlin[5..0] input. This is an optional field and defaults to UNUSED.
DQS_DELAY_CHAIN_ DELAYCTRLIN_SOURCE	Optional	Core, DLL	DLL	This parameter is the Gray-coded delay chain setting for the DQS read path. This is an optional input and defaults to GND. You can ignore this input if the *<use_phasectrlin>* parameter is set to FALSE and *<phase_setting>* is set to 0. You can feed this input by the delayctrlout output of a DLL or the core. This input does not need to match the polarity of its source and can be inverted (unless fed by the delayctrlout output of a DLL).

Table A–3. Megafunction Parameters to Configure DQS Input Path (Part 2 of 4)

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQS_DELAY_CHAIN_ PHASECTRLIN	OptionalO	FALSE	FALSE, TRUE	This parameter is the signal used to choose the phase applied to the dqsbusout output when *<use_phasectrlin>* is set to TRUE; otherwise, the phase setting is determined by the *<phase_setting>* parameter. Only feed this port by the dqsinputphasesetting output port of the DQS config. You must connect this port if *<use_phasectrlin>* is set to TRUE; otherwise, this port is optional and defaults to GND. This input must match the polarity of its source and cannot be inverted.
DQS_DELAY_CHAIN_PHASE_ SETTING	Optional	0	0, 1, 2, 3, 4	This parameter is a delay chain setting. If *<use_phasectrlin>* is set to FALSE, this parameter sets the number of DQS delay buffers that the dqsin signal should travel through to the dqsbusout port; otherwise, you can ignore this setting. When it is set to 0, the dqsbusout signal bypasses the DQS delay chain. When it is set to 1, 2, 3, or 4, the dqsbusout signal goes through 1, 2, 3, or 4 delay buffers, respectively, that are controlled by delayctrlin[5..0]. The phase-shift implemented is determined by the ratio of the DQS delay buffers to DLL delay buffers: phase_shift = phase_setting/DLL delay_chain_length * 360
DELAY_BUFFER_MODE	Optional	LOW	LOW, HIGH	This parameter determines whether the variable delay buffers are working in low-frequency mode or high-frequency mode.
DQS_PHASE_SHIFT	Optional	0	0..36,000	This parameter indicates the phase shift between the delayed DQS signal and the input DQS signal in units of hundreds of degrees (for example, a 90° phase shift is represented as 9,000). This parameter is applicable only for static timing analysis because the phase shift through the delayctrlin[5..0], phasectrlin[2..0], and offsetctrlin[5..0] ports cannot be determined.
DQS_OFFSETCTRL_ENABLE	Optional	FALSE	FALSE, TRUE	This parameter describes whether the offsetctrlin[5..0] inputs are used.

Table A–3. Megafunction Parameters to Configure DQS Input Path (Part 3 of 4)

Parameter Name	Optional/ Required	Default	Legal Values	Description
DQS_CTRL_LATCHES_ENABLE	Optional	FALSE	FALSE, TRUE	This parameter describes whether the delayctrlin[5..0] and offsetctrlin[5..0] inputs are latched or not. If set to TRUE, only a DLL feeds the delayctrlin[5..0] input bus.
USE_DQS_ENABLE_CTRL_PHASE_CTRLIN	Optional	TRUE	FALSE, TRUE	If set to TRUE, the phase setting is determined by the phasectrlin input. If set to FALSE, the phase setting is determined by the *<phase_setting>* parameter.
DQS_ENABLE_CTRL_PHASE_SETTING	Optional	0	0..7	This parameter determines the phase shift implemented by the delay chain if *<use_phasectrlin>* is set to FALSE; otherwise, you can ignore this setting.
LEVEL_DQS_ENABLE	OptionalO	FALSE	FALSE, TRUE	This is an optional field and defaults to FALSE.
DELAY_DQS_ENABLE_BY_HALF_CYCLE	Optional	FALSE	FALSE, TRUE	This is an optional field and defaults to FALSE.
DQS_ENABLE_CTRL_ADD_PHASE_TRANSFER_REG	Optional	FALSE	FALSE, TRUE, DYNAMIC	If set to TRUE, a negative edge-triggered register is added in data path for the clock phase transfer. If set to FALSE, no register is added. If it is set to DYNAMIC, the enaphasetransferreg input determines whether the register is added or not. You can use the negative-edge register to guarantee the setup and hold time for a phase transfer.
DQS_ENABLE_CTRL_INVERT_PHASE	Optional	FALSE	FALSE, TRUE, DYNAMIC	If set to TRUE, the phase output is inverted. If set to FALSE, the phase output is not inverted. If it is set to DYNAMIC, the phaseinvertctrl input determines whether the inverter is used or not. Use the inverter to increase the number of available phases.
USE_IO_CLOCK_DIVIDER_PHASECTRLIN	Optional	TRUE	FALSE, TRUE	If set to TRUE, the phase setting is determined by the phasectrlin input. If set to FALSE, the phase setting is determined by the *<phase_setting>* parameter.
IO_CLOCK_DIVIDER_PHASE_SETTING	Optional	0	0..7	This parameter determines the phase shift implemented by the delay chain if *<use_phasectrlin>* is set to FALSE; otherwise, ignore this setting.

Table A–3. Megafunction Parameters to Configure DQS Input Path (Part 4 of 4)

Parameter Name	Optional/ Required	Default	Legal Values	Description
IO_CLOCK_DIVIDER_INVERT_PHASE	Optional	FALSE	FALSE, TRUE, DYNAMIC	If set to TRUE, the phase output is inverted. If set to FALSE, the phase output is not inverted. If it is set to DYNAMIC, the phaseinvertctrl input determines whether the inverter is used or not. Use the inverter to increase the number of available phases.
USE_IO_CLOCK_DIVIDER_MASTERIN	Optional	FALSE	FALSE, TRUE	If set to TRUE, then the masterin input is used to synchronize this divider with another IO_CLOCK_DIVIDER. If set to FALSE, this divider operates independently (this mode is meant for the master divider for a group of dividers).
IO_CLOCK_DIVIDER_CLK_SOURCE	Optional	Core	Core, dqs_bus, inverted_dqs_bus	If Core, IO_CLOCK_DIVIDER:clk port on the primitive is fed by io_clock_divider_clk port on the megafunction. If dqs_bus, IO_CLOCK_DIVIDER:clk port on the primitive is fed by dqs_bus_out port on the megafunction. If inverted_dqs_bus, IO_CLOCK_DIVIDER:clk port on the primitive is fed by!dqs_bus_out port on the megafunction.

Table A–4 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQS output path.

Table A–4. Megafunction Parameters to Configure DQS Output Path (Part 1 of 2)

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQS_OUTPUT_DELAY_CHAIN1	Optional	FALSE	FALSE, TRUE	Instantiates DQS_OUTPUT_DELAY_CHAIN1 if TRUE.
USE_DQS_OUTPUT_DELAY_CHAIN2	Optional	FALSE	FALSE, TRUE	Instantiates DQS_OUTPUT_DELAY_CHAIN2 if TRUE.
DQS_OUTPUT_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates DQS_OUTPUT_FF if FF. Instantiates DQS_OUTPUT_DDIO_OUT if DDIO.
DQS_OUTPUT_REG_POWER_UP	Optional	LOW	LOW, HIGH	This parameter describes the power-up condition of all registers in the primitive.

Table A–4. Megafunction Parameters to Configure DQS Output Path (Part 2 of 2)

Parameter Name	Optional/ Required	Default	Legal Values	Description
DQS_OUTPUT_REG_ASYNC_ MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the areset port clears, presets, or has no effect on the DDIO register(s). The areset port is required if this parameter is set to CLEAR or PRESET.
DQS_OUTPUT_REG_SYNC_ MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the sreset port clears, presets, or has no effect on the DDIO register(s). The sreset port is required if this parameter is set to CLEAR or PRESET.

Table A–5 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQS OE path.

Table A–5. Megafunction Parameters to Configure DQS OE Path

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQS_OE_DELAY_ CHAIN1	Optional	FALSE	FALSE, TRUE	Instantiates DQS_OE_DELAY_CHAIN1 if TRUE
USE_DQS_OE_DELAY_ CHAIN2	Optional	FALSE	FALSE, TRUE	Instantiates DQS_OE_DELAY_CHAIN2 if TRUE
DQS_OE_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates DQS_OE_FF if FF Instantiates DQS_OE_DDIO_OE if DDIO
DQS_OE_REG_POWER_UP	Optional	LOW	LOW, HIGH	This parameter describes the power-up condition of all registers in the primitive.
DQS_OE_REG_ASYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the areset port clears, presets, or has no effect on the DDIO register(s). The areset port is required if this parameter is set to CLEAR or PRESET.
DQS_OE_REG_SYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the sreset port clears, presets, or has no effect on the DDIO register(s). The sreset port is required if this parameter is set to CLEAR or PRESET.

Table A–6 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the OCT path. The possible values for <*IO*> are DQS, DQSn, BIDIR_DQ and OUTPUT_DQ.

Table A–6. Megafunction Parameters to Configure OCT Path

Parameter Namer	Optional/ Required	Default	Legal Values	Mapping to Sub-Blocks
USE_OCT_DELAY_CHAIN1	Optional	FALSE	FALSE, TRUE	Instantiates <*IO*>_OCT_DELAY_CHAIN1 if TRUE
USE_OCT_DELAY_CHAIN2	Optional	FALSE	FALSE, TRUE	Instantiates <*IO*>_OCT_DELAY_CHAIN2 if TRUE
OCT_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates <*IO*>_OCT_FF if FF Instantiates <*IO*>_OCT_DDIO_OE if DDIO

Table A–7 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQ input path. The possible values for <*IO*> are BIDIR_DQ and INPUT_DQ.

Table A–7. Megafunction Parameters to Configure DQ Input Path (Part 1 of 3)

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQ_INPUT_DELAY_ CHAIN	Optional	FALSE	FALSE, TRUE	Instantiates <*IO*>_INPUT_DELAY_CHAIN if TRUE
DQ_INPUT_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates <*IO*>_INPUT_FF if FF Instantiates <*IO*>_DDIO_IN if DDIO
DQ_INPUT_REG_POWER_UP	Optional	LOW	LOW, HIGH	This parameter describes the power-up condition of all registers in the primitive.
DQ_INPUT_REG_ASYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the areset port clears, presets, or has no effect on the DDIO register(s). The areset port is required if this parameter is set to CLEAR or PRESET.
DQ_INPUT_REG_SYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the sreset port clears, presets, or has no effect on the DDIO register(s). The sreset port is required if this parameter is set to CLEAR or PRESET.

Table A–7. Megafunction Parameters to Configure DQ Input Path (Part 2 of 3)

Parameter Name	Optional/ Required	Default	Legal Values	Description
DQ_INPUT_REG_CLK_SOURCE	Optional	dqs_bus	dqs_bus, Core	If Core, <IO>_INPUT_FF:clk / <IO>_DDIO_IN:clk port on the primitive is fed by dq_input_reg_clk port on the megafunction If dqs_bus, <IO>_INPUT_FF:clk / <IO>_DDIO_IN:clk port on the primitive is fed by dqs_bus_out port on the megafunction
DQ_INPUT_REG_CLK_USE_CLKN	Optional	FALSE	FALSE, TRUE	If TRUE, <IO>_DDIO_IN:clk port on the primitive is fed by dqs_bus_out port on the megafunction and <IO>_DDIO_IN:clkn port on the primitive is fed by dqsn_bus_out port on the megafunction
USE_DQ_IPA	Optional	FALSE	FALSE, TRUE	Instantiates <IO>_IPA_HIGH / <IO>_IPA_LOW if TRUE
USE_DQ_IPA_PHASECTRLIN	Optional	FALSE	FALSE, TRUE	<IO>_IPA_HIGH.use_phasectrlin port on the primitive <IO>_IPA_LOW.use_phasectrlin port on the primitive
DQ_IPA_PHASE_SETTING	Optional	0	0..7	<IO>_IPA_HIGH.phase_setting port on the primitive <IO>_IPA_LOW.phase_setting port on the primitive
DQ_IPA_ADD_INPUT_CYCLE_DELAY	Optional	FALSE	FALSE, TRUE, DYNAMIC	<IO>_IPA_HIGH.add_input_cycle_delay port on the primitive <IO>_IPA_LOW.add_input_cycle_delay port on the primitive
DQ_IPA_BYPASS_OUTPUT_REGISTER	Optional	FALSE	FALSE, TRUE	<IO>_IPA_HIGH.bypass_output_register port on the primitive <IO>_IPA_LOW.bypass_output_register port on the primitive
DQ_IPA_ADD_PHASE_TRANSFER_REG	Optional	FALSE	FALSE, TRUE, DYNAMIC	<IO>_IPA_HIGH.add_phase_transfer_reg port on the primitive <IO>_IPA_LOW.add_phase_transfer_reg port on the primitive

Table A–7. Megafunction Parameters to Configure DQ Input Path (Part 3 of 3)

Parameter Name	Optional/ Required	Default	Legal Values	Description
DQ_IPA_INVERT_PHASE	OptionalIO	FALSE	FALSE, TRUE, DYNAMIC	*<IO>*_IPA_HIGH.invert_phase port on the primitive *<IO>*_IPA_LOW.invert_phase port on the primitive
DQ_HALF_RATE_USE_ DATAOUTBYPASS	Optional	FALSE	FALSE, TRUE	*<IO>*_HALF_RATE_INPUT.use_ dataoutbypass port on the primitive

Table A–8 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQ output path. The possible values for *<IO>* are BIDIR_DQ and OUTPUT_DQ.

Table A–8. Megafunction Parameters to Configure DQ Output Path

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQ_OUTPUT_DELAY_ CHAIN1	Optional	FALSE	FALSE, TRUE	Instantiates *<IO>*_OUTPUT_DELAY_CHAIN1 if TRUE
USE_DQ_OUTPUT_DELAY_ CHAIN2	Optional	FALSE	FALSE, TRUE	Instantiates *<IO>*_OUTPUT_DELAY_CHAIN2 if TRUE
DQ_OUTPUT_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates port on the primitive if FF Instantiates *<IO>*_OUTPUT_DDIO_OUT if DDIO
DQ_OUTPUT_REG_POWER_UP	Optional	LOW	LOW, HIGH	This parameter describes the power-up condition of all registers in the primitive.
DQ_OUTPUT_REG_ASYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the areset port clears, presets, or has no effect on the DDIO register(s). The areset port is required if this parameter is set to CLEAR or PRESET.
DQ_OUTPUT_REG_SYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the sreset port clears, presets, or has no effect on the DDIO register(s). The sreset port is required if this parameter is set to CLEAR or PRESET.

Table A–9 summarizes the Clear Box parameters for the ALTDQ_DQS megafunction to configure the DQ OE path. The possible values for <*IO*> are BIDIR_DQ and OUTPUT_DQ.

Table A–9. Megafunction Parameters to Configure DQ OE Path

Parameter Name	Optional/ Required	Default	Legal Values	Description
USE_DQ_OE_DELAY_ CHAIN1	Optional	FALSE	FALSE, TRUE	Instantiates <*IO*>_OE_DELAY_CHAIN1 if TRUE
USE_DQ_OE_DELAY_ CHAIN2	Optional	FALSE	FALSE, TRUE	Instantiates <*IO*>_OE_DELAY_CHAIN2 if TRUE
DQ_OE_REG_MODE	Optional	NONE	NONE, FF, DDIO	Instantiates <*IO*>_OE_FF if FF Instantiates <*IO*>_OE_DDIO_OE if DDIO
DQ_OE_REG_POWER_UP	Optional	LOW	LOW, HIGH	This parameter describes the power-up condition of all registers in the primitive.
DQ_OE_REG_ASYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the areset port clears, presets, or has no effect on the DDIO register(s). The areset port is required if this parameter is set to CLEAR or PRESET.
DQ_OE_REG_SYNC_MODE	Optional	NONE	CLEAR, PRESET, NONE	This parameter determines if the sreset port clears, presets, or has no effect on the DDIO register(s). The sreset port is required if this parameter is set to CLEAR or PRESET.

Revision History

The following table lists the revision history for this user guide.

Date	Version	Changes Made
February 2012	v.5.0	Update input frequency.
May 2010	v4.0	Updated the ALTDLL and ALTDQ_DQS Megafunctions User Guide. ■ Added the new ALTDLL and ALTDQ_DQS parameter editor options. ■ Added Appendix for clear box Generator. ■ Added new design example. ■ Removed repetitive and redundant information.
September 2009	v3.0	Updated and reorganised Chapter 1 and Chapter 2.
December 2008	v2.0	Added: ■ HardCopy III in "Features" on page 1–2 ■ Added full-rate mode and half-rate mode terms and descriptions in Figure 1–3 on page 1–9 ■ Last sentence in"Step 1: Understand the Available External Memory Dedicated Circuitry in Devices" on page 1–5 ■ "ALTDLL Features" on page 1–9 ■ Note *(2)*, *(3)*, *(4)* in Figure 1–3 on page 1–6 ■ Added Megafunction Ports and Parameters tables in "About these Megafunctions" ■ Added Sub-Blocks Parameters and Ports in "About these Megafunctions" ■ Added Advanced Options tables in "Getting Started" ■ Removed figures in "Getting Started" ■ Added new tables in "Specifications" ■ Added "Primitives" on page 1–51
July 2008	v1.0	Initial release

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com

Contact *(Note 1)*	Contact Method	Address
Non-technical support (Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table lists the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, file names, file name extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
WARNING	A warning calls attention to a condition or possible situation that can cause injury to the user.
↵	The angled arrow indicates you should press the Enter key.
👣	The feet direct you to more information on a particular topic.