



Double Data Rate I/O (ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR) Megafunctions

User Guide



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

UG-DDRMGAFCTN-6.1



© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



Chapter 1. About These Megafunctions

Features	1-1
Device Support	1-1
Common Applications	1-2
DDR SDRAM, DDR2 SDRAM and RLDRAM II Memory Interfaces	1-2
QDR SRAM and QDRII SRAM Memory Interfaces	1-2
High-Speed Interface Applications	1-2
Resource Utilization and Performance	1-3

Chapter 2. Parameter Settings

MegaWizard Parameter Settings	2-1
Command Line Interface Parameters	2-4

Chapter 3. Functional Description

DDR Device Configuration	3-1
Input Configuration	3-1
Output Configuration	3-3
Bidirectional Configuration	3-4
DDR I/O Timing	3-7
Design Example Files	3-8
Design Example 1: 8-Bit DDR Divider Using ALTDDIO_IN and ALTDDIO_OUT	3-9
Generate a Divider Using ALTDDIO_IN and ALTDDIO_OUT	3-9
Create the ALTDDIO_IN Module	3-9
Create the ALTDDIO_OUT Module	3-10
Create the LPM_DIVIDE Module	3-12
Create a Divider	3-13
Implement the Divider Design	3-13
Functional Results—Simulate the Divider Design in the ModelSim-Altera Software	3-14
Design Example 2: 8-Bit DDR Divider Using ALTDDIO_BIDIR	3-15
Generate a Divider Using ALTDDIO_BIDIR	3-15
Create the ALTDDIO_BIDIR Module	3-15
Create the lpm_divide Module	3-17
Create a Divider	3-17
Implement the Divider Design	3-17
Functional Results—Simulate the Divider Design in the ModelSim-Altera Software	3-18
ALTDDIO_IN Megafunction Ports	3-19
ALTDDIO_OUT Megafunction Ports	3-20
ALTDDIO_BIDIR Megafunction Ports	3-21
Prototypes and Component Declarations	3-23
Verilog HDL Prototype for the ALTDDIO_IN Megafunction	3-23
VHDL Component Declaration for the ALTDDIO_IN Megafunction	3-23
Verilog HDL Prototype for the ALTDDIO_OUT Megafunction	3-24
VHDL Component Declaration for the ALTDDIO_OUT Megafunction	3-25
Verilog HDL Prototype for the ALTDDIO_BIDIR Megafunction	3-25
VHDL Component Declaration for the ALTDDIO_BIDIR Megafunction	3-26
VHDL LIBRARY-USE Declaration	3-27

Additional Information

Document Revision History Info-1
How to Contact Altera Info-1
Typographic Conventions Info-2

The Altera® DDR I/O megafunctions configure the DDR I/O registers in APEX™ II, Arria® series, Cyclone® series, HardCopy® series, and Stratix® series devices. You can also use the megafunctions to implement DDR registers in the logic elements (LEs). In Arria GX, Stratix series, HardCopy II, HardCopy Stratix, and APEX II devices, the DDR registers are implemented in the I/O element (IOE). In Cyclone series devices, the megafunctions automatically implement the DDR registers in the LEs closest to the pin. The ALTDDIO_IN megafunction implements the interface for DDR inputs. The ALTDDIO_OUT megafunction implements the interface for DDR outputs. The ALTDDIO_BIDIR megafunction implements the interface for bidirectional DDR inputs and outputs.

Features

The ALTDDIO megafunctions implement a DDR interface and offer the following additional features:

- The ALTDDIO_IN megafunction receives data on both edges of the reference clock
- The ALTDDIO_OUT megafunction transmits data on both edges of the reference clock
- The ALTDDIO_BIDIR megafunction transmits and receives data on both edges of the reference clock
- Asynchronous clear and asynchronous set input options available
- Synchronous clear and synchronous set input options available for Arria GX and Stratix series devices.
- inclock signal to sample the DDR input
- outclock signal to register the data output
- Clock enable signals
- Bidirectional port for the ALTDDIO_BIDIR megafunction
- An output enable input for the ALTDDIO_OUT and ALTDDIO_BIDIR megafunctions

Device Support

The ALTDDIO megafunctions support APEX II, Arria, Cyclone, HardCopy, and Stratix device series.


Common Applications

DDR registers capture and/or send data at twice the rate of the clock or data strobe to interface with a memory device or other high-speed interface application in which the data is clocked at both edges of the clock. The DDR registers interface with DDR SDRAM, DDR2 SDRAM, RLDRAM II, QDR SRAM, and QDR II SRAM memory devices. You can also use the DDR I/O registers as a SERDES bypass mechanism in LVDS applications. This section provides information about the following DDR I/O applications:

- DDR SDRAM, DDR2 SDRAM, and RLDRAM II memory interfaces
- QDR SRAM and QDR II SRAM memory interfaces
- High-speed interface applications


DDR SDRAM, DDR2 SDRAM and RLDRAM II Memory Interfaces

DDR SDRAM, DDR2 SDRAM, and RLDRAM II write and read data at twice the clock rate by capturing data on both the positive and negative edge of a clock. DDR and DDR2 SDRAM are JEDEC standards. RLDRAM II devices have minimal latency to support designs that require fast response times. These DDR memory interfaces use a variety of I/O standards, such as SSTL-II, 1.8-V HSTL, LVTTTL, and LVCMOS.

-  The DDR and DDR II SDRAM controller is available by downloading the Altera DDR SDRAM Controller MegaCore[®] function from www.altera.com.


QDR SRAM and QDR II SRAM Memory Interfaces

The QDR and QDR II SRAM standard is defined jointly by Cypress Semiconductor Corporation, Integrated Device Technology, Inc., and Micron Technology, Inc. QDR and QDR II SRAMs have separate DDR read and write ports that pass data concurrently. The combination of concurrent transactions and DDR signaling allows data to be passed four times faster than by conventional SRAMs. The I/O standards used for QDR SRAM devices are 1.5-V HSTL class I and II. QDR II SRAMs use both 1.5-V and 1.8-V HSTL class I.

-  The QDR II SDRAM controller is available by downloading the Altera QDR II SDRAM Controller MegaCore function from www.altera.com.

High-Speed Interface Applications

High-speed interface applications use various differential standards, such as LVDS, LVPECL, PCML, or HyperTransport technology to transfer data. These standards often use DDR data. Stratix series devices implement high-speed standards either by using the dedicated differential I/O SERDES blocks or by bypassing SERDES and using the DDR I/O circuitry in SERDES bypass mode. DDR megafunctions, PLLs, and shift registers are all used in SERDES functionality.

-  For more information, refer to the following documents:

- The *External Memory Interfaces in Stratix II and Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook*

- The *Implementing Double Data Rate I/O Signaling in Cyclone Devices* chapter in volume 1 of the *Cyclone Device Handbook*
- *AN 167: Using Flexible-LVDS I/O Pins in APEX II Devices*

Resource Utilization and Performance

For details about the resource utilization of the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions in various devices, and the performance of devices that include these megafunctions, refer to the MegaWizard Plug-In Manager and the compilation reports for each device.

This section describes the parameter settings for the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions.

You can parameterize the megafunctions using the MegaWizard™ Plug-In Manager or the command-line interface (CLI). Altera recommends that you configure the megafunctions using the MegaWizard Plug-In Manager.

MegaWizard Parameter Settings


 The options on pages 1 and 2a of the parameter editor are the same for all supported device families. For more information, refer to the [Introduction to Megafunctions User Guide](#).

Table 2–1 lists the parameter settings for the ALTDDIO_IN megafunction.

Table 2–1. ALTDDIO_IN Parameter Settings

Option	Description
Currently selected device family	Specify the Altera® device family you are using.
Width: (bits)	Specify the width of the data buses.
Asynchronous clear and asynchronous set ports	Select Use 'aclr' port for asynchronous clear (aclr). Select Use 'aset' port for asynchronous preset (aset). If you are not using any of the asynchronous clear options, select Not used and specify whether registers should power up high or low by turning on/off Registers power up high .
Synchronous clear and synchronous set ports	Select Use 'sclr' port for synchronous clear (sclr). Select Use 'sset' port for synchronous preset (sset). If you are not using any of the synchronous clear options, select Not used . The synchronous reset option is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only.
Use 'inclocken' port	Turn on this option to add a clock enable port that controls when data input is clocked in. This signal prevents data from being passed through.
Invert input clock	When enabled, the first bit of data is captured on the rising edge of the input clock. If not enabled, the first bit of data is captured on the falling edge of the input clock.

Table 2–2 lists the parameter settings for the ALTDDIO_OUT megafunction.

Table 2–2. ALTDDIO_OUT Parameter Settings

Option	Description
Currently selected device family	Specify the Altera device family you are using.
Width: (bits)	Specify the width of the data buses.

Table 2-2. ALTDIO_OUT Parameter Settings

Option	Description
Asynchronous clear and asynchronous set ports	Select Use 'aclr' port for asynchronous clear (aclr). Select Use 'aset' port for asynchronous preset (aset). If you are not using any of the asynchronous clear options, select Not used and specify whether registers should power up high or low by turning on/off Registers power up high .
Use 'outclocken' port	Turn on this option to add a clock enable port to control when data output is clocked in. This signal prevents data from being passed through.
Invert 'dataout' output	Turn on this option to invert the <code>dataout []</code> output port. This option is available for Cyclone III and Cyclone II devices only.
Use output enable port	Turn on this option to create an output enable input port (<code>oe</code>) to control when the data is set out to the <code>dataout</code> port.
Use 'oe_out' port to connect to tri-state output buffer(s)	Turn on this option to create an output enable port for the bidirectional <code>padio</code> port. This port is available for Stratix III and Cyclone III devices only.
Register 'oe' port	Turn on this option to register the output-enable (<code>oe</code>) input port.
Delay switch-on by half a clock cycle	Turn on this option to use an additional <code>oe</code> register. When the additional <code>oe</code> register is used, the output pin is held at high impedance for an extra half clock cycle after the <code>oe</code> port goes high.
Synchronous clear and synchronous set ports	Select Use 'sclr' port for synchronous clear (sclr). Select Use 'sset' port for synchronous preset (sset). If you are not using any of the synchronous clear options, select Not used . The synchronous reset option is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only.

Table 2-3 lists the parameter settings for the ALTDIO_BIDIR megafunction.

The ALTDIO_BIDIR megafunction combines the ALTDIO_IN and ALTDIO_OUT megafunction functionality into a single megafunction, which instantiates bidirectional DDR ports.



In Stratix II and Stratix devices, if you use the ALTDIO_BIDIR megafunction for your DQS signal in an external memory interface, the undelayed DQS signal is routed to the LE.

Table 2-3. ALTDIO_BIDIR Parameter Settings (Part 1 of 2)

Option	Description
Currently selected device family	Specify the Altera device family you are using.
Width: (bits)	Specify the width of the data buses.
Asynchronous clear and asynchronous set ports	Select Use 'aclr' port for asynchronous clear (aclr). Select Use 'aset' port for asynchronous preset (aset). If you are not using any of the asynchronous clear options, select Not used and specify whether registers should power up high or low by turning on/off Registers power up high .
Synchronous clear and synchronous set ports	Select Use 'sclr' port for synchronous clear (sclr). Select Use 'sset' port for synchronous preset (sset). If you are not using any of the synchronous clear options, select Not used . The synchronous reset option is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only.

Table 2-3. ALTDDIO_BIDIR Parameter Settings (Part 2 of 2)

Option	Description
Invert 'padio' port	The 'padio' port is inverted whenever driven as an output. This option is available for Cyclone III and Cyclone II devices only.
Use 'inclocken' and 'outclocken' ports	Turn on this option to add a clock enable port to control when data input and output are clocked in. This signal prevents data from being passed through.
Use output enable port	Turn on this option to create an output enable input port (oe) to control when the data is set out to the <code>dataout</code> port.
Use oe_out port to connect to tri-state output buffer(s)	Output enable for the bidirectional <code>padio</code> port. This port is available for Stratix III and Cyclone III devices only.
Register 'oe' port	Turn on this option to register the output-enable (oe) input port.
Delay switch-on by a half clock cycle	Turn on this option to use an additional <code>oe</code> register. When the additional <code>oe</code> register is used, the output pin is held at high impedance for an extra half clock cycle after the <code>oe</code> port goes high.
Use 'combout' port	Use the optional data port <code>combout</code> . The <code>combout</code> port sends data to the core, bypassing the DDR I/O input registers. For bidirectional operation, you must enable the <code>dataout_h</code> and <code>dataout_l</code> ports, the <code>combout</code> port, or both.
Use 'dqsundelayedout' port	Creates undelayed output from the <code>DQS</code> pins. If you use the <code>ALTDDIO_BIDIR</code> megafunction for your <code>DQS</code> signal in an external memory interface, you route the undelayed <code>DQS</code> signal to the LE, in Stratix II and Stratix devices. This option is available in Stratix, Stratix GX, and HardCopy Stratix devices only.
Use 'dataout_h' and 'dataout_l' ports	Enables the <code>dataout_h</code> and <code>dataout_l</code> ports.
Implement input registers in LEs	Implements the input path in logic elements. This option is available only if the <code>dataout_h</code> and <code>dataout_l</code> ports are enabled.

In the `ALTDDIO_IN`, `ALTDDIO_OUT`, and `ALTDDIO_BIDIR` parameter editor, page 5 shows a list of the libraries needed to properly simulate the design files. You can also enable the Quartus II software to generate a synthesis area and timing estimation netlist for the megafunction for use by third-party tools.

On the final page of the parameter editor, specify the files you wish to have generated for your custom megafunction. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (as indicated by a green check mark). Click **Finish** to create an instance of the megafunction.



For more information about the wizard-generated files, refer to Quartus II Help or to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Command Line Interface Parameters

Expert users can choose to instantiate and parameterize the megafunction through the command-line interface using the clear box generator command. This method requires you to have command-line scripting knowledge.


 For more information about using the clear box generator, refer to the *Introduction to Megafunctions User Guide*.

Table 2-4 lists the parameters for the ALTDDIO_IN megafunction.

Table 2-4. ALTDDIO_IN Parameters

Parameter	Type	Required	Description
width	Integer	Yes	Width of datain, dataout_h, and dataout_l ports.
power_up_high	String	No	When both aset and aclr ports are unused, power_up_high parameter is available to specify power-up state of output ports. Values are "ON" and "OFF". The default setting is "OFF".
intended_device_family	String	No	This parameter is used for modeling and behavioral simulation.
invert_input_clocks	String	No	Values are ON or OFF . If omitted, the default value is OFF . If this parameter is set to ON , the inclock port must be inverted at the connection.
implement_input_in_lcell	String	No	Specifies whether the input channels should be implemented using logic cells. Values are ON , OFF , and UNUSED . If omitted, the default is UNUSED . This parameter is available for all supported devices.

Table 2-5 lists the parameters for the ALTDDIO_OUT megafunction.

Table 2-5. ALTDDIO_OUT Parameters (Part 1 of 2)

Parameter	Type	Required	Comments
width	Integer	Yes	Sets the width of the datain_h, datain_l, and dataout ports.
power_up_high	String	No	When both the aset and aclr ports are unused, the power_up_high parameter is available to specify the power-up state of the output ports. Values are ON and OFF . The default setting is OFF .
intended_device_family	String	No	This parameter is used for modeling and behavioral simulation.
oe_reg	String	No	Specifies whether the oe port is registered. Values are REGISTERED , UNREGISTERED , and UNUSED . The default setting is UNUSED .
extend_oe_disable	String	No	This specifies whether the second oe register should be used. When the second oe register is used, the output pin is held at high impedance an extra half clock cycle after the oe port goes high. Values are ON , OFF , and UNUSED . The default setting is UNUSED .
lpm_hint	String	No	Allows you to assign Altera-specific parameters in VHDL Design Files (.vhd). The default is UNUSED .

Table 2-5. ALTDDIO_OUT Parameters (Part 2 of 2)

Parameter	Type	Required	Comments
lpm_type	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
invert_output	String	No	Specifies whether to invert the <code>dataout []</code> output port. Values are ON or OFF . This parameter is available for Cyclone III and Cyclone II devices only.

Table 2-6 lists the parameters for the ALTDDIO_BIDIR megafunction.

Table 2-6. ALTDDIO_BIDIR Parameters

Parameter	Type	Required	Comments
width	Integer	Yes	Width of the <code>datain_h</code> , <code>datain_l</code> , and <code>dataout</code> ports.
power_up_high	String	No	When both the <code>aset</code> and <code>aclr</code> ports are unused, the <code>power_up_high</code> parameter is available to specify the power-up state of the output ports. Values are ON and OFF . The default setting is OFF .
intended_device_family	String	No	This parameter is used for modeling and behavioral simulation.
lpm_type	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
oe_reg	String	No	Specifies whether the <code>oe</code> port is registered. Values are REGISTERED , UNREGISTERED , and UNUSED . The default setting is UNUSED .
implement_input_in_lcell	String	No	Specifies whether the input channels should be implemented using logic cells. Values are ON , OFF , and UNUSED . The default is UNUSED .
extend_oe_disable	String	No	Specifies whether the second <code>oe</code> register should be used. When the second <code>oe</code> register is used, the output pin is held at high impedance an extra half clock cycle after the <code>oe</code> port goes high. Values are ON , OFF , and UNUSED . The default setting is UNUSED .
invert_output	String	No	Specifies whether to invert the <code>dataout []</code> output port. Values are ON or OFF . This parameter is available for Cyclone III and Cyclone II devices only.

This chapter describes the functional description and the design examples of the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions. This section also includes prototypes, component declarations and ports of the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions. You can use the ports to customize the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions according to your application.

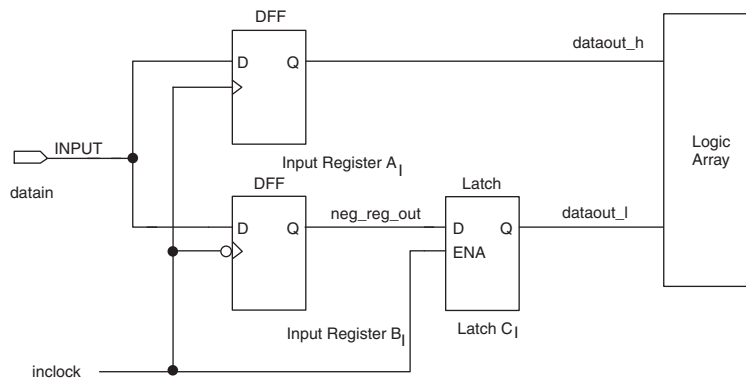
DDR Device Configuration

The following sections describe how the DDR registers are configured in the Stratix series and APEX II devices.

Input Configuration

When the IOE is configured as an input pin, input registers A_I and B_I and latch C_I implement the input path for DDR I/O. Figure 3–1 shows an IOE configured for DDR inputs for a Stratix series or APEX II device.

Figure 3–1. Input DDR I/O Path Configuration for a Stratix Series or APEX II Device

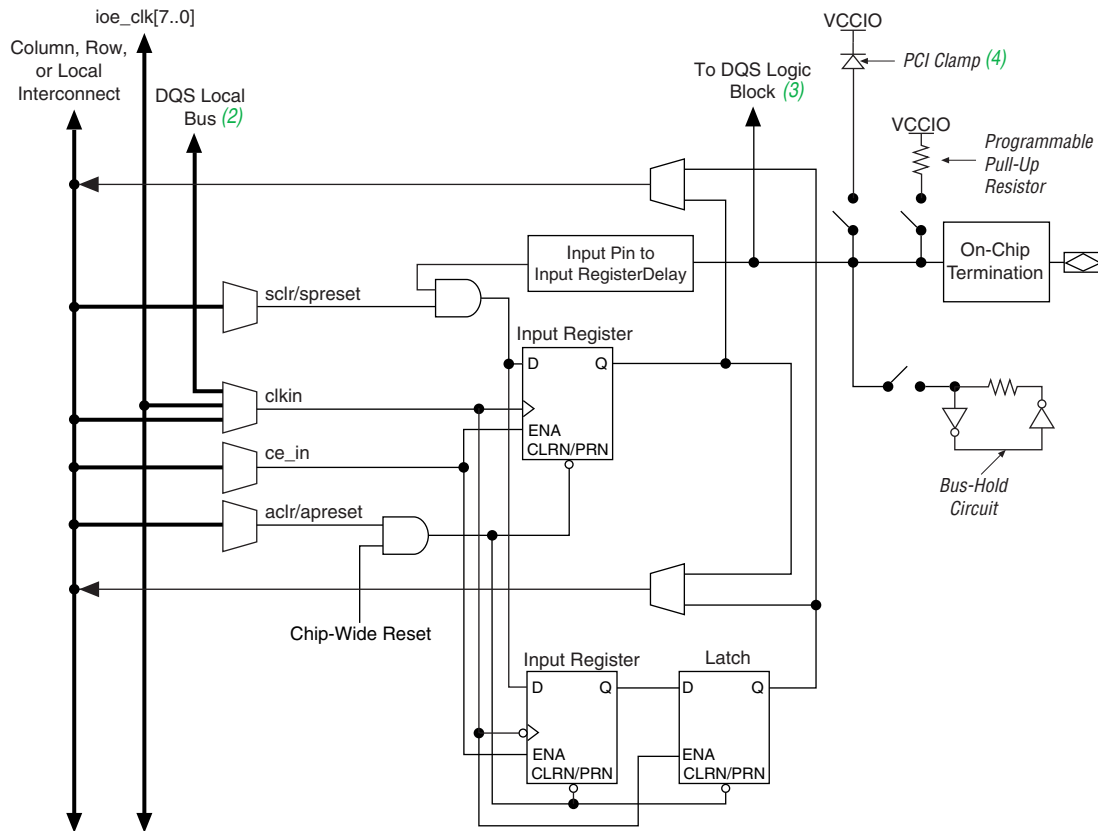


Note to Figure 3–1:

- (1) On the falling edge of the clock, the negative-edge triggered register B_I acquires the first data bit. On the corresponding rising edge of the clock, the positive-edge triggered register A_I acquires the second data bit. For a successful data transfer to the logic array, the latch C_I synchronizes the data from register B_I to the positive edge of the clock.

Figure 3-2 shows an IOE configured for DDR inputs for a Stratix or Stratix II device.

Figure 3-2. Stratix II IOE in DDR Input I/O Configuration



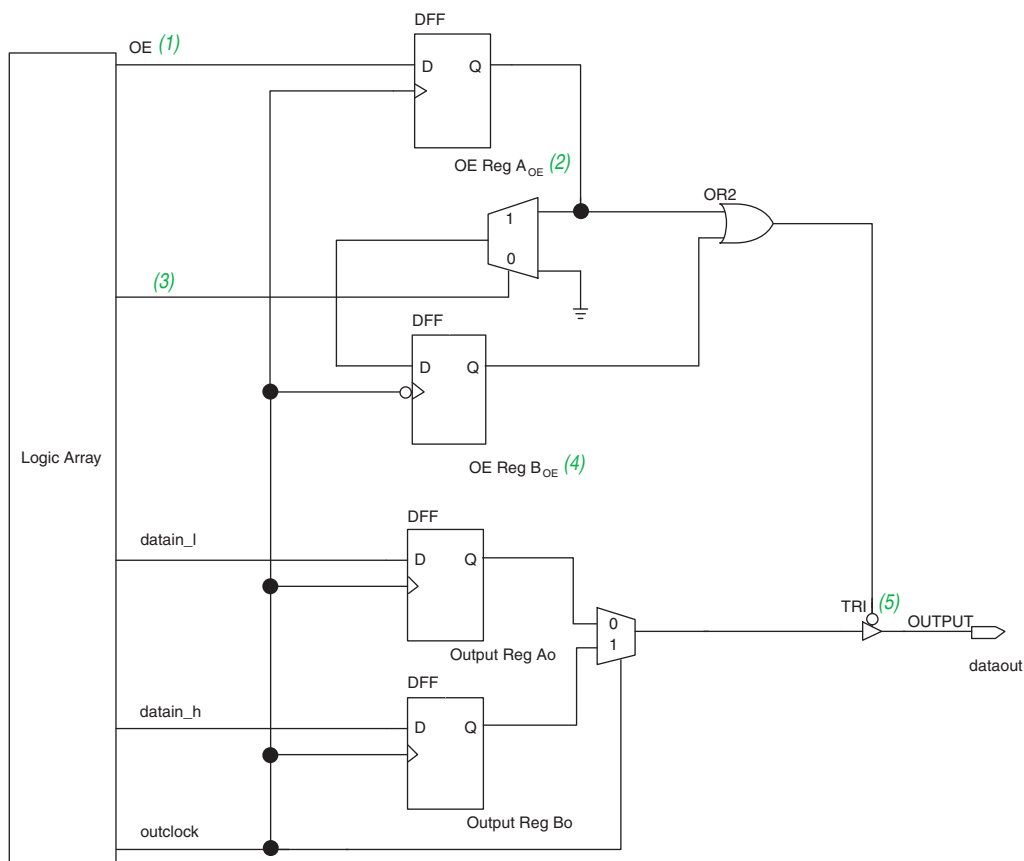
Notes to Figure 3-2:

- (1) All input signals to the IOE can be inverted at the IOE.
- (2) This signal connection is only allowed on dedicated DQ function pins.
- (3) This signal is for dedicated DQS function pins only.
- (4) The optional PCI clamp is only available on column I/O pins.

Output Configuration

The dedicated output registers for Stratix series and APEX II devices are labeled A_O and B_O . These positive-edge triggered registers and a multiplexer are used to implement the output path for DDR I/O. Figure 3-3 shows the IOE configuration for DDR outputs in Stratix series and APEX II devices.

Figure 3-3. Output DDR I/O Path Configuration for Stratix Series and APEX II Devices



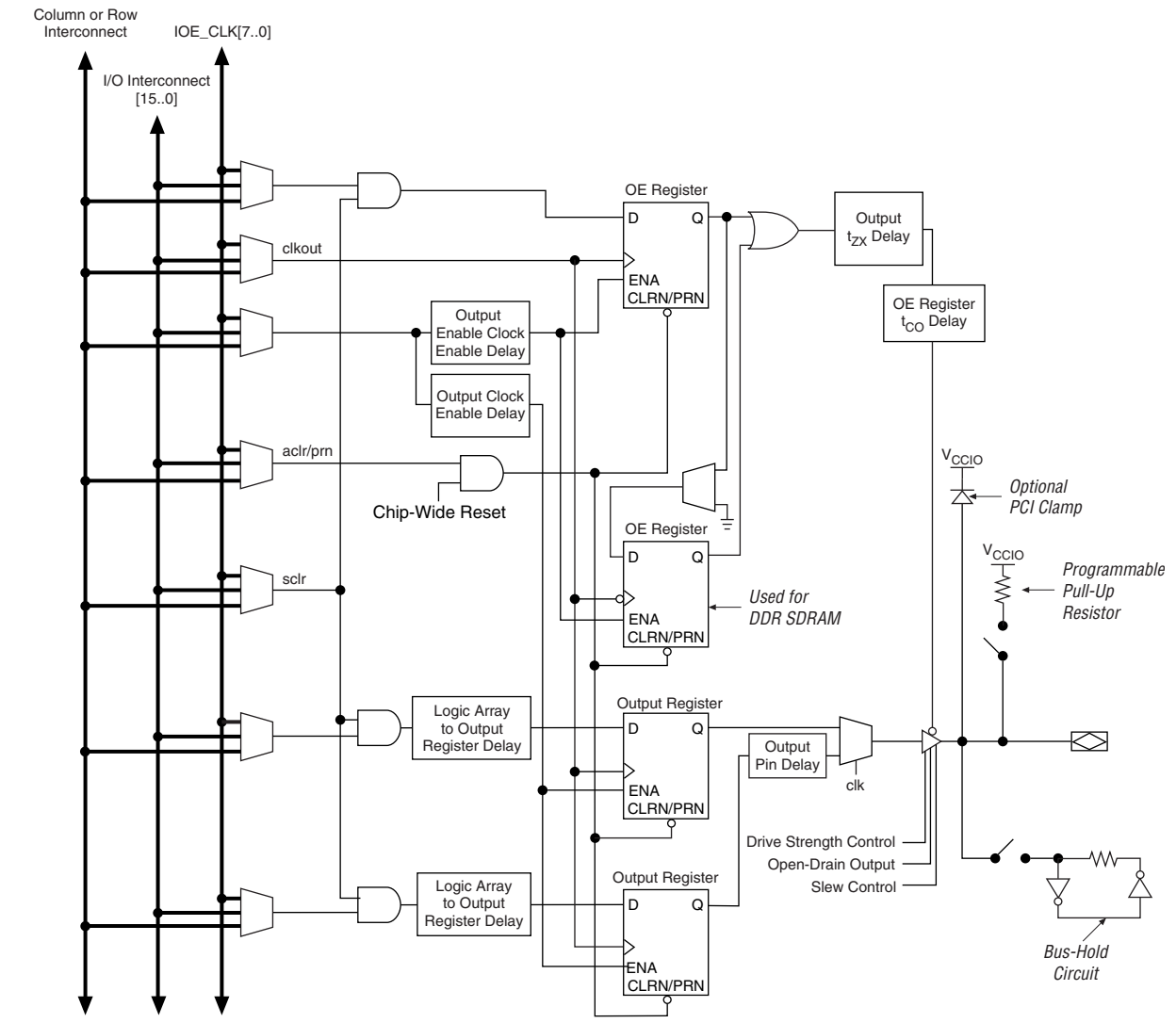
Notes to Figure 3-3:

- (1) The OE is active low, but the Quartus® II software implements this as active high and automatically adds an inverter before the input to the A_{OE} register during compilation. If desired, you can change the OE back to active low.
- (2) Register A_{OE} generates the enable signal for general-purpose DDR I/O applications.
- (3) This select line corresponds to the delay switch-on by a half clock cycle option in the MegaWizard® Plug-In Manager.
- (4) Register B_{OE} generates the delayed enable signal for DDR SDRAM applications.
- (5) The tri-state is active high by default. However, you can design it to be active low.

On the positive edge of the clock, a high data bit and a low data bit are captured in registers A_O and B_O . The outputs of these two registers are fed to the input of a 2-to-1 multiplexer, which uses the output register clock as its control signal. A high clock selects the data in register B_O , and a low level of the clock selects the data in register A_O . This process doubles the data at the I/O pin.

Figure 3-4 shows the IOE configuration for DDR outputs in Stratix series devices.

Figure 3-4. Stratix IOE in DDR Output I/O Configuration



Bidirectional Configuration

Input and output registers are independent of each other, enabling the bidirectional DDR I/O path to be implemented entirely in the I/O element for Stratix, Stratix GX, and APEX II devices. The bidirectional configuration includes an input path, an output path, and two output enable registers.

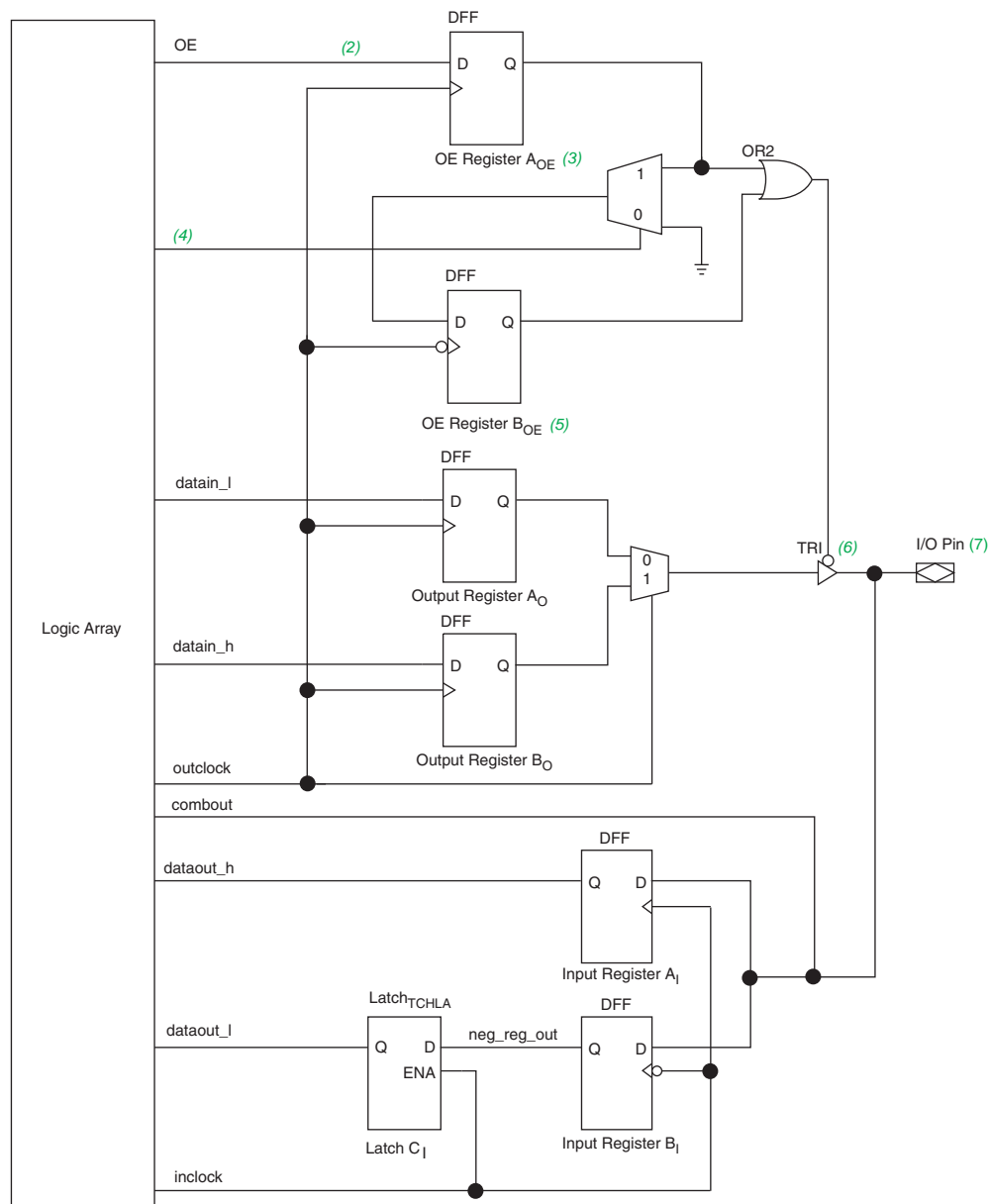
The bidirectional path consists of two data flow paths:

- Input path active
- Output path active

When the input path is active, the output enable disables the tri-state buffer, which prevents data from being sent out on the output path. Disabling the tri-state buffer prevents contention at the I/O pin. The input path behaves like the input configuration as shown in [Figure 3-1 on page 3-1](#). When the output path is active, the output enable register AOE controls the flow of data from the output registers. During outgoing transactions, the bidirectional configuration behaves like the output configuration as shown in [Figure 3-3 on page 3-3](#). The second output enable register (B_{OE}) is used for DDR SDRAM interfaces. This negative-edge register extends the high-impedance state of the pin by a half clock cycle. This option is useful to provide the write preamble for the DQS strobe in the DDR SDRAM interfaces. This feature is enabled by using the **Delay switch-on by a half clock cycle** option in the ALTDDIO_BIDIR megafunction in the Quartus II software. You can bypass the input registers and latch to get a combinational output (combout) from the pin going into the APEX II or Stratix series device. Furthermore, the input data ports (dataout_h and dataout_l) can be disabled. These features are especially useful for generating data strobes like DQS.


Figure 3-5 shows the bidirectional DDR I/O configuration for Stratix series and APEX II devices.


Figure 3-5. Bidirectional DDR I/O Path Configuration



Notes to Figure 3-5:

- (1) All control signals can be inverted at the I_{OE} .
- (2) The OE signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before input to the A_{OE} register during compilation. If desired, you can change the OE back to active low.
- (3) The A_{OE} register generates the enable signal for general-purpose DDR I/O applications.
- (4) This line selects whether the OE signal should be delayed by half a clock cycle.
- (5) The B_{OE} register generates the delayed enable signal for the write strobes or write clocks for memory interfaces.
- (6) The tri-state enable is by default active low. You can, however, design it to be active high.
- (7) You can also have combinational output to the I/O pin. This path is not shown in the diagram.

- 
 For more information about clock signals and output enable signals for Stratix series or APEX II devices, refer to the following documents:
 - The *Stratix II Architecture* chapter in volume 1 of the *Stratix II Device Handbook*
 - *APEX II Programmable Logic Device Family Data Sheet*

- 
 For more information about the DDR registers in Cyclone devices, refer to the *Implementing Double Data Rate I/O Signaling in Cyclone Devices* chapter in volume 1 of the *Cyclone Device Handbook*.

DDR I/O Timing

Figure 3-6 shows the functional timing waveform for the input path. The signal names are the port names used in the ALTDDIO_IN megafunction. The `datain` signal is the input from the pin to the DDR circuitry. The output of register B_1 is `neg_reg_out`. The output of latch C_1 is `dataout_1`, and the output of register A_1 is `dataout_h`. `dataout_h` and `dataout_1` feed the logic array and show the conversion of the data from a DDR implementation to positive-edge triggered data.

Figure 3-6. DDR I/O Input Timing Waveform

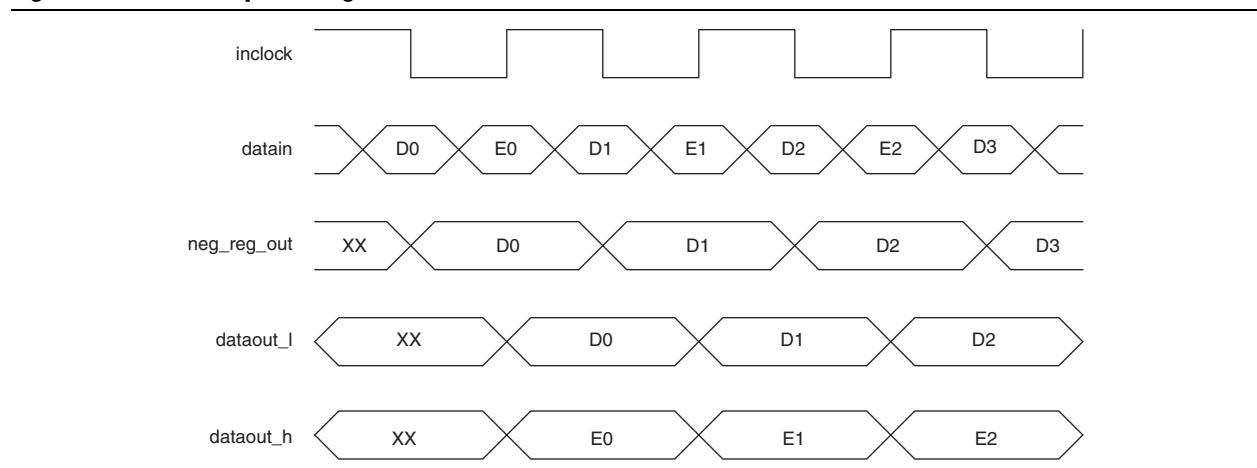
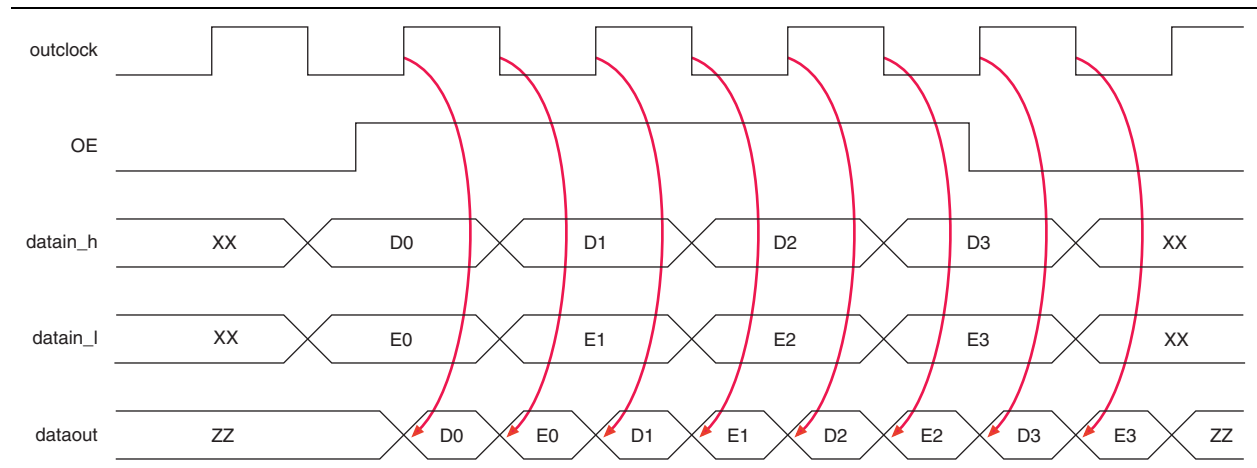


Figure 3-7 shows a functional timing waveform example for the output path with the output enable registered. In this example, the **delay switch-on by a half clock cycle** is not turned on, so the second output enable register (B_{OE}) is not used. The output enable signal OE is active high and can be driven from a pin or internal logic. The data signals $datain_l$ and $datain_h$ are driven from the logic array to output registers A_O and B_O . The $dataout$ signal is the output from the DDR circuitry to the pin.

Figure 3-7. DDR I/O Output Timing Waveform



The waveform in Figure 3-7 reflects the software simulation results. The OE signal is active low in silicon; however, the Quartus II software implements this as active high and automatically adds an inverter before the D input of the OE register A_{OE} . You can change the OE back to active low, if desired.

Design Example Files

The design examples for the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions in this user guide use the MegaWizard Plug-In Manager in the Quartus II software. The design example files are available for download from the following locations:

- On the [Documentation: Quartus II Development Software](#) page, expand the **Using Megafunctions** section and then expand the **I/O** section.
- On the [Documentation: User Guides](#) section of the Altera website.

The designs are simulated in the ModelSim®-Altera software to generate a waveform display of the device behavior. You should be familiar with the ModelSim-Altera software before using the design examples. To get started with the ModelSim-Altera software, refer to the [ModelSim-Altera Software Support](#) page on the Altera website. The support page includes links to such topics as installation, usage, and troubleshooting.

Design Example 1: 8-Bit DDR Divider Using ALTDDIO_IN and ALTDDIO_OUT

This section presents a design example that uses the ALTDDIO_IN and ALTDDIO_OUT megafunctions to generate a divider. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

In this example, you perform the following activities:

- Create a divider using the ALTDDIO_IN, ALTDDIO_OUT, and LPM_DIVIDE megafunctions and the MegaWizard Plug-in Manager
- Implement the design and assign the Stratix EP1S10F780C6 device to the project
- Compile and simulate the design

Generate a Divider Using ALTDDIO_IN and ALTDDIO_OUT

The new megafunction created in this example is added to the top-level file in your Quartus II project.

Create the ALTDDIO_IN Module

Follow these steps to create the ALTDDIO_IN module:

1. Unzip the ALTDDIO_DesignExample_ex1.zip file to any working directory on your PC.
2. In the Quartus II software, open the ex1.qar project.
3. On the Tools menu, select **MegaWizard Plug-In Manager**.
4. In the MegaWizard Plug-In manager dialog box, select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays.
5. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in [Table 3-9](#). Click **Next** to advance from one page to the next.

Table 3-1. Configuration Settings for Design Example 1 (ALTDDIO_IN) (Part 1 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Which megafunction would you like to customize	In the I/O folder, select ALTDDIO_IN
	Which device family will you be using?	Stratix
	Which type of output file do you want to create?	VHDL
	What name do you want for the output file?	ddin
	Return to this page for another create operation	Turned off

Table 3-1. Configuration Settings for Design Example 1 (ALTDDIO_IN) (Part 2 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	Currently selected device family	Stratix IV
	Match project/default	Turned on
	Width: (bits)	8
	Use 'aclr' port	Turned off
	Use 'aset' port	Turned off
	Not used	Turned on
	Registers power up high	Turned off
	Use 'sclr' port	Turned off
	Use 'sset' port	Turned off
	Not used	Turned on
	Use 'inclocken' port	Turned off
	Invert input clock	Turned off
	4	Generate netlist
5	Variation file	Turned on
	Quartus II IP file	Turned on
	Quartus II symbol file (.bsf)	Turned off
	Instantiation template file	Turned on
	Verilog HDL black box file (_bb.v)	Turned on
	AHDL Include file (.inc)	Turned off
	VHDL component declaration file (.cmp)	Turned on
	PinPlanner ports file (.PPF)	Turned on

6. Click **Finish**.

The ALTDDIO_IN module is now built.

Create the ALTDDIO_OUT Module

Follow these steps to create the ALTDDIO_OUT module:

1. On the Tools menu, select **MegaWizard Plug-In Manager**.
2. In the MegaWizard Plug-In manager dialog box, select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays.

3. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3-9. Click **Next** to advance from one page to the next.

Table 3-2. Configuration Settings for Design Example 1 (ALTDDIO_OUT)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Which megafunction would you like to customize	In the I/O folder, select ALTDDIO_OUT
	Which device family will you be using?	Stratix
	Which type of output file do you want to create?	VHDL
	What name do you want for the output file?	ddout
	Return to this page for another create operation	Turned off
3	Currently selected device family	Stratix IV
	Match project/default	Selected
	Width: (bits)	8
	Use 'aclr' port	Turned off
	Use 'aset' port	Turned off
	Not used	Turned on
	Registers power up high	Turned off
	Not used	Turned on
	Use 'outclocken' port	Turned off
	Invert 'dataout' output	Turned off
	Use output enable port	Turned off
	Use 'oe_out' port to connect to tri-state output buffer(s)	Turned off
	Register 'oe' port	Turned off
	Delay switch-on by half a clock cycle	Turned off
	Use 'sclr' port	Turned off
Use 'sset' port	Turned off	
Not used	Turned on	
4	Generate netlist	Turned off
5	Variation file	Turned on
	Quartus II IP file	Turned on
	Quartus II symbol file (.bsf)	Turned off
	Instantiation template file	Turned on
	Verilog HDL black box file (_bb.v)	Turned on
	AHDL Include file (.inc)	Turned off
	VHDL component declaration file (.cmp)	Turned on
PinPlanner ports file (.PPF)	Turned on	

4. Click **Finish**.

The ALTDDIO_OUT module is now built.

Create the LPM_DIVIDE Module

Follow these steps to create the `lpm_divide` module:

1. On the Tools menu, select **MegaWizard Plug-In Manager**.
2. In the MegaWizard Plug-In manager dialog box, select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays.
3. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in [Table 3-9](#). Click **Next** to advance from one page to the next.

Table 3-3. Configuration Settings for Design Example 1 (LPM_DIVIDE)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Which megafunction would you like to customize	In the Arithmetic folder, select LPM_DIVIDE
	Which device family will you be using?	Stratix
	Which type of output file do you want to create?	VHDL
	What name do you want for the output file?	lp_div
	Return to this page for another create operation	Turned off
3	Currently selected device family	Stratix IV
	Match project/default	Turned on
	How wide should the 'numerator' input bus be?	8
	How wide should the 'denominator' input bus be?	8
	Numerator Representation	Select Unsigned
Denominator Representation	Select Unsigned	
4	Do you want to pipeline the function?	Select Yes, I want an output latency of 1 clock cycle
	Create an Asynchronous Clear input	Turned off
	Create a Clock Enable input	Turned off
	Which do you wish to optimize?	Select Default Optimization
	Always return a positive remainder?	Select Yes
5	Generate netlist	Turned off
6	Variation file	Turned on
	Quartus II IP file	Turned on
	Quartus II symbol file (.bsf)	Turned off
	Instantiation template file	Turned on
	Verilog HDL black box file (_bb.v)	Turned on
	AHDL Include file (.inc)	Turned off
	VHDL component declaration file (.cmp)	Turned on
PinPlanner ports file (.PPF)	Turned on	

4. Click **Finish**.

The `lpm_divide` module is now built.

Create a Divider

Use the following steps to combine the ALTDDIO_IN, ALTDDIO_OUT, and lpm_divide modules to create a divider.

1. In the Quartus II software, open the file **ex1.vhd**.
2. On the Project menu, click **Add/Remove Files in Project**. The **File Settings** dialog box displays.
3. In the **File Settings** dialog box, click (...) after **File name** and browse for **ex1.vhd** in the project folder.
4. Select **ex1.vhd** and click **Add**.
5. Click **OK**.

The top-level file is added to the project. You have now created the complete design file.

In the complete divider design, the DDR_IN_OUT_H[7..0] signals are the numerator and the DDR_IN_OUT_L[7..0] signals are the denominator. The DDR_IN_OUT_H[7..0] and DDR_IN_OUT_L[7..0] sets of signals are passed into the lp_div function where the quotient and remainder are calculated. The divider calculates the quotient and remainder through a one-stage pipeline. The quotient and remainder are fed into the ddout module and also to REG_DDROUT8_IN_H[7..0] and REG_DDROUT8_IN_L[7..0]. The ddout module then drives the data out through pins DDR_IN8_OUT[7..0] at double the data rate.

Implement the Divider Design

This section describes how to assign the Stratix EP1S10F780C6 device to the project and how to compile the project.

1. With the **ex1.qar** project open, on the Assignments menu, click **Settings**. The **Settings** dialog box displays.
2. In the **Category** list, select **Device**. The **Device** dialog box displays.
3. From the **Family** list, select **Stratix**.
4. Under **Target device**, select **Specific device selected in 'Available devices' list**.
5. Under **Show in 'Available devices' list**, select **FBGA** as the **Package**, **Pin count of 780**, **Speed grade of 6**, and turn on **Show advanced devices**.
6. Click **OK**.
7. On the Processing menu, click **Start Compilation**.
8. When the **Full Compilation was successful** box displays, click **OK**.

Functional Results—Simulate the Divider Design in the ModelSim-Altera Software

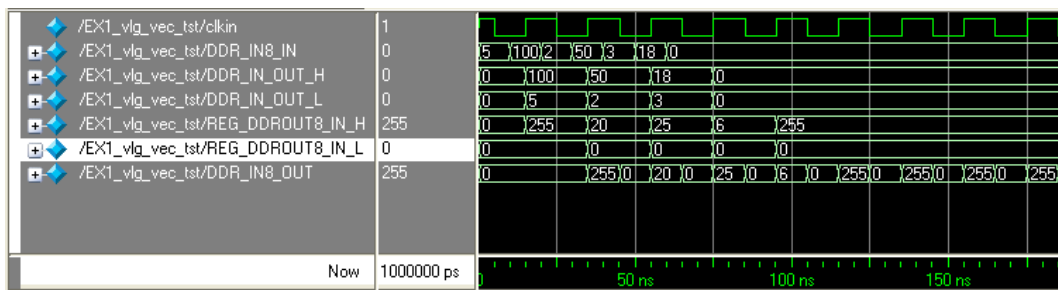
Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.

To set up the ModelSim-Altera software, follow these steps:

1. Unzip the `ALTDDIO_ex1_msim.zip` file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files and open the `ALTDDIO_ex1.do` file in a text editor.
3. In line 1 of the `ALTDDIO_ex1.do` file, replace `<insert_directory_path_here>` with the directory path of the appropriate library files. For example,
`C:/altera/71/modelsim_ae/altera/verilog/stratix`
4. On the File menu, click **Save**.
5. Start **ModelSim-Altera**.
6. On the File menu, click **Change Directory**.
7. Select the folder in which you unzipped the files. Click **OK**.
8. On the Tools menu, click **Execute Macro**.
9. Select the `ALTDDIO_ex1.do` file and click **Open**. This is a script file for ModelSim that automates all the necessary settings for the simulation.
10. Verify the results by looking at the **Waveform Viewer** window.

You can rearrange signals, remove redundant signals, and change the radix by modifying the script in the `ALTDDIO_ex1.do` file. [Figure 3-8](#) shows the expected simulation results in ModelSim-Altera software.

Figure 3-8. ModelSim Simulation Results



Design Example 2: 8-Bit DDR Divider Using ALTDDIO_BIDIR

This section presents a design example that uses the ALTDDIO_BIDIR megafunction to generate a divider. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into your overall project.

In this example, you perform the following tasks:

- Create a divider using the ALTDDIO_BIDIR and lpm_divide megafunctions and the MegaWizard Plug-in Manager
- Implement the design and assign the Stratix EP1S10F780C6 device to the project
- Compile and simulate the design

Generate a Divider Using ALTDDIO_BIDIR

The new megafunction created in this example is added to the top-level file in your Quartus II project.

Create the ALTDDIO_BIDIR Module

Follow these steps to create the ALTDDIO_BIDIR module:

1. Unzip the ALTDDIO_DesignExample_ex2.zip file to any working directory on your PC.
2. In the Quartus II software, open the ex2.qar project .
3. On the Tools menu, select **MegaWizard Plug-In Manager**.
4. In the MegaWizard Plug-In manager dialog box, select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays.
5. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in [Table 3-9](#). Click **Next** to advance from one page to the next.

Table 3-4. Configuration Settings for Design Example 1 (ALTDDIO_BIDIR) (Part 1 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Which megafunction would you like to customize	In the I/O folder, select ALTDDIO_BIDIR
	Which device family will you be using?	Stratix
	Which type of output file do you want to create?	VHDL
	What name do you want for the output file?	alt_bid
	Return to this page for another create operation	Turned off

Table 3–4. Configuration Settings for Design Example 1 (ALTDDIO_BIDIR) (Part 2 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	Currently selected device family	Stratix IV
	Match project/default	Turned on
	Width: (bits)	8
	Use 'aclr' port	Turned off
	Use 'aset' port	Turned off
	Not used	Turned on
	Registers power up high	Turned off
	Use 'sclr' port	Turned off
	Use 'sset' port	Turned off
	Not used	Turned on
	Invert 'padio' port	Turned off
	4	Use 'inclocken' and 'outclocken' ports
Use output enable port		Turned on
Use 'oe_out' port to connect to tri-state output buffer(s)		Turned off
Register 'oe' port		Turned off
Delay switch-on by half a clock cycle		Turned off
Use 'combout' port		Turned off
Use 'dqsundelayedout' port		Turned off
Use 'dataout_h' and 'dataout_l' ports		Turned on
Implement input registers in LEs	Turned off	
5	Generate netlist	Turned off
6	Variation file	Turned on
	Quartus II IP file	Turned on
	Quartus II symbol file (.bsf)	Turned off
	Instantiation template file	Turned on
	Verilog HDL black box file (_bb.v)	Turned on
	AHDL Include file (.inc)	Turned off
	VHDL component declaration file (.cmp)	Turned on
PinPlanner ports file (.PPF)	Turned on	

6. Click **Finish**.

The ALTDDIO_BIDIR module is now built.

Create the lpm_divide Module

To generate a divider, follow the steps shown in “[Create the LPM_DIVIDE Module](#)” on page 3–12.

Create a Divider

Use the following steps to combine the ALTDDIO_BIDIR and lpm_divide modules to create a divider.

Follow these steps to create a top-level VHDL file:

1. In the Quartus II software, with the **ex2.qar** project open, open the file **ex2.vhd**.
2. On the Project menu, click **Add/Remove File in Project**. The **File Settings** page displays.
3. In the **File Settings** window, click (...) after **File name** and browse for **ex2.vhd** in the project folder.
4. Select **ex2.vhd** and click **Add**.
5. Click **OK**.

The top-level file is added to the project. You have now created the complete design file.

This design implements the same divider as that in Design Example 1, but the functionality of the ALTDDIO_IN and ALTDDIO_OUT modules is implemented in a single megafunction, ALTDDIO_BIDIR. The bidirectional pins DDR_BIDIR8 [7..0] receive data at double the clock rate. The DDRBIDIR8_OUT_H [7..0] signals are the numerator and the DDRBIDIR8_OUT_L [7..0] signals are the denominator. These two sets of signals are passed to the lpm_divide module where the quotient and remainder are calculated. The divider calculates the quotient and remainder with a one-stage pipeline. The quotient and remainder are then fed via signals quotient [7..0] and remain [7..0] back to the ALTDDIO_BIDIR megafunction. The ALTDDIO_BIDIR megafunction then drives the data out through pins DDR_BIDIR8 [7..0] at double the data rate.

Implement the Divider Design

This section describes how to assign the Stratix EP1S10F780C6 device to the project and compile the project.

1. With the **ex2.qar** project open, on the Assignments menu, click **Settings**. The **Settings** dialog box displays.
2. In the **Category** list, select **Device**.
3. To answer **Which device family will you be using?**, select **Stratix**.
4. Under **Target device**, select **Specific device selected in ‘Available devices’ list**.
5. In the **Available devices** list, select **EP1S10F780C6**.
6. Under **Show in ‘Available devices’ list**, select **FBGA** as the **Package**, **Pin count of 780**, **Speed grade of 6**, and turn on **Show Advanced Devices**.
7. Click **OK**.

8. On the Processing menu, click **Start Compilation**.
9. When the **Full Compilation was successful** box displays, click **OK**.

Functional Results—Simulate the Divider Design in the ModelSim-Altera Software

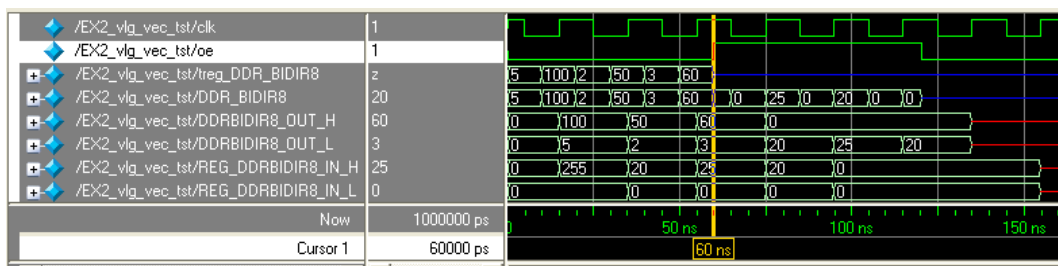
Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.

To set up the ModelSim-Altera software, follow these steps:

1. Unzip the **ALTDDIO_ex2_msim.zip** file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files and open the **ALTDDIO_ex2.do** file in a text editor.
3. In line 1 of the **ALTDDIO_ex2.do** file, replace *<insert_directory_path_here>* with the directory path of the appropriate library files. For example,
C:/altera/71/modelsim_ae/altera/verilog/stratix
4. On the File menu, click **Save**.
5. Start **ModelSim-Altera**.
6. On the File menu, click **Change Directory**.
7. Select the folder in which you unzipped the files. Click **OK**.
8. On the Tools menu, click **Execute Macro**.
9. Select the **ALTDDIO_ex2.do** file and click **Open**. This is a script file for ModelSim that automates all the necessary settings for the simulation.
10. Verify the results by looking at the **Waveform Viewer** window.

You can rearrange signals, remove redundant signals, and change the radix by modifying the script in the **ALTDDIO_ex2.do** file. [Figure 3-9](#) shows the expected simulation results in ModelSim-Altera software.

Figure 3-9. ModelSim Simulation Results



ALTDDIO_IN Megafunction Ports

Figure 3-10 shows the ports for the ALTDDIO_IN megafunction.

Figure 3-10. ALTDDIO_IN Ports

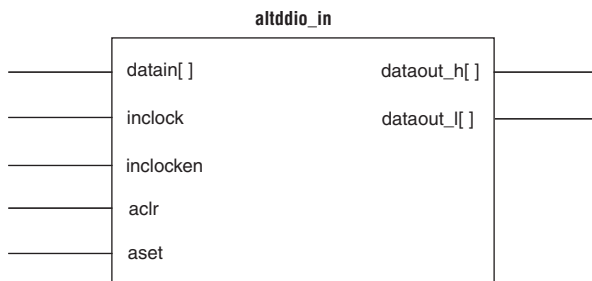


Table 3-5 and Table 3-6 list the input and output ports for the ALTDDIO_IN megafunction.

Table 3-5. ALTDDIO_IN Input Ports

Name	Required	Description
datain[]	Yes	DDR input data port. Input port WIDTH wide. The datain port should be directly fed from an input pin in the top-level design.
inclock	Yes	Clock signal to sample the DDR input. The datain port is sampled on each clock edge of the inclock signal.
inclocken	No	Clock enable for the data clock
aclr	No	Asynchronous clear input. The aclr and aset ports cannot be connected at the same time.
aset	No	Asynchronous set input. The aclr and aset ports cannot be connected at the same time.
sclr	No	Synchronous clear input. The sclr and sset ports cannot be connected at the same time. The sclr port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only. ⁽¹⁾
sset	No	Synchronous set input. The sclr and sset ports cannot be connected at the same time. The sset port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only. ⁽¹⁾

Note to Table 3-5:

(1) When designing with Stratix III devices, when sclr is asserted, it synchronously presets both the input path and resynchronization register.

Table 3-6. ALTDDIO_IN Output Ports

Name	Required	Description
dataout_h[]	Yes	Data sampled from datain[] port at the rising edge of the inclock signal.
dataout_l[]	Yes	Data sampled from datain[] port at the falling edge of the inclock signal.

ALTDIO_OUT Megafunction Ports

Figure 3-11 shows the ports for the ALTDIO_OUT megafunction.

Figure 3-11. ALTDIO_OUT Ports

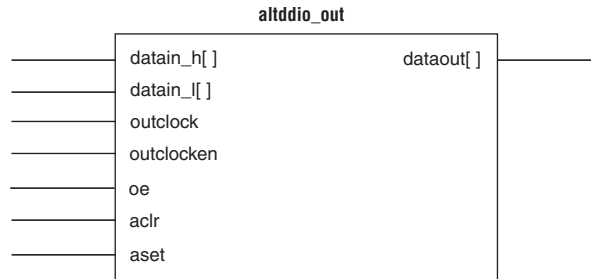


Table 3-7 and Table 3-8 list the input and output ports for the ALTDIO_OUT megafunction.

Table 3-7. ALTDIO_OUT Input Ports

Name	Required	Description
datain_h[]	Yes	Input data for rising edge of outclock port. Input port WIDTH wide.
datain_l[]	Yes	Input data for falling edge of outclock port. Input port WIDTH wide.
outclock	Yes	Clock signal to register data output. dataout port outputs DDR data on each level of outclock signal.
outclocken	No	Clock enable for outclock port.
aclr	No	Asynchronous clear input. The aclr and aset ports cannot be connected at the same time.
aset	No	Asynchronous set input. The aclr and aset ports cannot be connected at the same time.
oe	No	Output enable for the dataout port. Active-high signal. You can add an inverter if you need an active-low oe.
sclr	No	Synchronous clear input. The sclr and sset ports cannot be connected at the same time. The sclr port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only.
sset	No	Synchronous set input. The sclr and sset ports cannot be connected at the same time. The sset port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only.

Table 3-8. ALTDIO_OUT Output Ports

Name	Required	Description	Comments
dataout[]	Yes	DDR output data port.	Output port WIDTH wide. dataout port should directly feed an output pin in top-level design.
oe_out	No	Output enable for the bidirectional padio port.	Output port [WIDTH-1..0] wide. This port is available for Stratix III and Cyclone III devices only.

ALTDDIO_BIDIR Megafunction Ports

Figure 3-12 shows the ports for the ALTDDIO_BIDIR megafunction.

Figure 3-12. ALTDDIO_BIDIR Ports

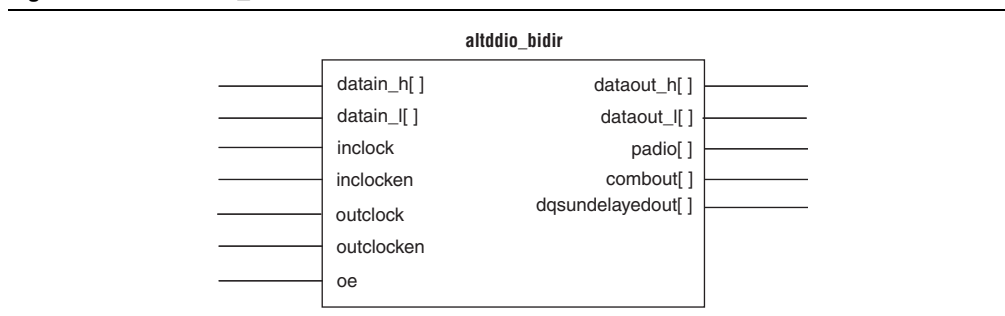


Table 3-9 lists the input ports, Table 3-10 list the output ports, and Table 3-11 lists the bidirectional ports for the ALTDDIO_BIDIR megafunction.

Table 3-9. ALTDDIO_BIDIR Input Ports

Name	Required	Description
datain_h[]	Yes	Input data to be output to the padio port at the rising edge of the outclock port. Input port [(WIDTH) - (1)..0] wide.
datain_l[]	Yes	Input data to be output to the padio port at the falling edge of the outclock port. Input port [(WIDTH) - (1)..0] wide.
inclock	Yes	Clock signal to sample the DDR input. The padio port is sampled on each clock edge of the inclock signal.
inclocken	No	Clock enable for the inclock port.
outclock	Yes	Clock signal to register the data output. The padio port outputs the DDR data on each edge of the outclock signal.
outclocken	No	Clock enable for the outclock port.
aclr	No	Asynchronous clear input. The aclr and aset ports cannot be connected at the same time.
aset	No	Asynchronous set input. The aclr and aset ports cannot be connected at the same time.
oe	No	Output enable for the bidirectional padio port. If the oe port is not connected, then the padio port is an output port.
sclr	No	Synchronous clear input. The sclr and sset ports cannot be connected at the same time. The sclr port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only. ⁽¹⁾
sset	No	Synchronous set input. The sclr and sset ports cannot be connected at the same time. The sset port is available for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, HardCopy II, and HardCopy Stratix devices only. ⁽¹⁾

Note to Table 3-9:

(1) When designing with Stratix III devices, when sclr is deasserted, it synchronously presents both the input path and resynchronization register.

Table 3-10. ALTDDIO_BIDIR Output Ports

Name	Required	Description
dataout_h[]	Yes	Data sampled from the <code>padio</code> port at the rising edge of the <code>inclock</code> signal. Output port <code>[WIDTH-1..0]</code> wide.
dataout_l[]	Yes	Data sampled from the <code>padio</code> port at the falling edge of the <code>inclock</code> signal. Output port <code>[WIDTH-1..0]</code> wide.
combout [] ⁽¹⁾	No	Combinational output directly fed by the <code>padio</code> port.
dqsundelayedout []	No	Undelayed output from the DQS pins. Output port <code>[WIDTH-1..0]</code> wide. ⁽²⁾
oe_out	No	Output enable for the bidirectional <code>padio</code> port. Output port <code>[WIDTH-1..0]</code> wide. This port is available for Stratix III and Cyclone III devices only.

Notes to Table 3-10:

(1) This port is available for Stratix series, HardCopy Stratix, Cyclone series, and APEX II devices only.

(2) This port is available for Stratix and HardCopy Stratix devices only.

Table 3-11. ALTDDIO_BIDIR Bidirectional Port

Name	Required	Description
<code>padio</code> []	Yes	Bidirectional DDR port that should directly feed a bidirectional pin in the top-level design. The DDR data is transmitted and received on this bidirectional port. Bidirectional port <code>[(WIDTH) - (1) .. 0]</code> wide.

Prototypes and Component Declarations

This section describes the prototypes and component declarations of the ALTDDIO_IN, ALTDDIO_OUT, and ALTDDIO_BIDIR megafunctions.

Verilog HDL Prototype for the ALTDDIO_IN Megafunction

You can locate the following Verilog HDL prototype in the Verilog Design File (.v) `altera_mf.v` in the `<Quartus II installation directory>\eda\synthesis` directory.

```
module altddio_in
#( parameter intended_device_family = "unused",
  parameter implement_input_in_lcell = "ON",
  parameter invert_input_clocks = "OFF",
  parameter power_up_high = "OFF",
  parameter width = 1,
  parameter lpm_type = "altddio_in",
  parameter lpm_hint = "unused")
( input wire aclr,
  input wire areset,
  input wire [width-1:0] datain,
  output wire [width-1:0] dataout_h,
  output wire [width-1:0] dataout_l,
  input wire inclock,
  input wire inclocken,
  input wire sclr,
  input wire sreset) /* synthesis syn_black_box=1 */;
endmodule //altddio_in
```

VHDL Component Declaration for the ALTDDIO_IN Megafunction

You can locate the following VHDL Design File (.vhd) `altera_mf.vhd` in the `<Quartus II installation directory>\libraries\vhdl\altera_mf` directory.

```
component altddio_in
  generic (
    intended_device_family:string := "unused";
    implement_input_in_lcell:string := "ON";
    invert_input_clocks:string := "OFF";
    power_up_high:string := "OFF";
    width: natural;
    lpm_hint:string := "UNUSED";
    lpm_type:string := "altddio_in"
  );
  port (
```

```

    aclr: in std_logic := '0';
    aset: in std_logic := '0';
    datain:in std_logic_vector(width-1 downto 0);
    dataout_h:out std_logic_vector(width-1 downto 0);
    dataout_l:out std_logic_vector(width-1 downto 0);
    inclock:in std_logic;
    inclocken:in std_logic := '1';
    sclr: in std_logic := '0';
    sset: in std_logic := '0'
);
end component;

```

Verilog HDL Prototype for the ALTDDIO_OUT Megafunction

You can locate the following Verilog HDL prototype in the Verilog Design File (.v) **altera_mf.v** in the *<Quartus II installation directory>\eda\synthesis* directory.

```

modulealtddio_out
#( parameterintended_device_family = "unused",
  parameterextend_oe_disable = "OFF",
  parameterinvert_output = "OFF",
  parameteroe_reg = "UNREGISTERED",
  parameterpower_up_high = "OFF",
  parameterwidth = 1,
  parameterlpm_type = "altddio_out",
  parameterlpm_hint = "unused")
( inputwireaclr,
  inputwireaset,
  inputwire[width-1:0]datain_h,
  inputwire[width-1:0]datain_l,
  outputwire[width-1:0]dataout,
  inputwireoe,
  outputwire[width-1:0]oe_out,
  inputwireoutclock,
  inputwireoutclocken,
  inputwiresclr,
  inputwiresset)/* synthesis syn_black_box=1 */;
endmodule //altddio_out

```

VHDL Component Declaration for the ALTDDIO_OUT Megafunction

You can locate the following VHDL Design File (.vhd) `altera_mf.vhd` in the `<Quartus II installation directory>\libraries\vhdl\altera_mf` directory.

```
component altddio_out
  generic (
    intended_device_family:string := "unused";
    extend_oe_disable:string := "OFF";
    invert_output:string := "OFF";
    oe_reg:string := "UNREGISTERED";
    power_up_high:string := "OFF";
    width:natural;
    lpm_hint:string := "UNUSED";
    lpm_type:string := "altddio_out"
  );
  port (
    aclr: in std_logic := '0';
    aset: in std_logic := '0';
    datain_h:in std_logic_vector(width-1 downto 0);
    datain_l:in std_logic_vector(width-1 downto 0);
    dataout:out std_logic_vector(width-1 downto 0);
    oe : in std_logic := '1';
    oe_out:out std_logic_vector(width-1 downto 0);
    outclock:in std_logic;
    outclocken:in std_logic := '1';
    sclr: in std_logic := '0';
    sset: in std_logic := '0'
  );
end component;
```

Verilog HDL Prototype for the ALTDDIO_BIDIR Megafunction

You can locate the following Verilog HDL prototype in the Verilog Design File (.v) `altera_mf.v` in the `<Quartus II installation directory>\eda\synthesis` directory.

```
modulealtddio_bidir
#( parameterintended_device_family = "unused",
  parameterextend_oe_disable = "OFF",
  parameterimplement_input_in_lcell = "OFF",
  parameterinvert_output = "OFF",
  parameteroe_reg = "UNREGISTERED",
  parameterpower_up_high = "OFF",
```

```

parameterwidth = 1,
parameterlpm_type = "altdio_bidir",
parameterlpm_hint = "unused")
( inputwireaclr,
  inputwireaset,
  outputwire [width-1:0] combout,
  inputwire [width-1:0] datain_h,
  inputwire [width-1:0] datain_l,
  outputwire [width-1:0] dataout_h,
  outputwire [width-1:0] dataout_l,
  outputwire [width-1:0] dqsundelayedout,
  inputwireinclock,
  inputwireinclocken,
  inputwireoe,
  outputwire [width-1:0] oe_out,
  inputwireoutclock,
  inputwireoutclocken,
  inoutwire [width-1:0] padio,
  inputwiresclr,
  inputwiresset)/* synthesis syn_black_box=1 */;
endmodule //altdio_bidir

```

VHDL Component Declaration for the ALTDDIO_BIDIR Megafunction

You can locate the following VHDL Design File (.vhd) `altera_mf.vhd` in the `<Quartus II installation directory>\libraries\vhdl\altera_mf` directory.

```

component altdio_bidir
  generic (
    intended_device_family:string := "unused";
    extend_oe_disable:string := "OFF";
    implement_input_in_lcell:string := "OFF";
    invert_output:string := "OFF";
    oe_reg:string := "UNREGISTERED";
    power_up_high:string := "OFF";
    width: natural;
    lpm_hint:string := "UNUSED";
    lpm_type:string := "altdio_bidir"
  );
  port (
    aclr: in std_logic := '0';
    aset: in std_logic := '0';

```



```
    combout:out std_logic_vector(width-1 downto 0);
    datain_h:in std_logic_vector(width-1 downto 0);
    datain_l:in std_logic_vector(width-1 downto 0);
    dataout_h:out std_logic_vector(width-1 downto 0);
    dataout_l:out std_logic_vector(width-1 downto 0);
    dqsundelayedout:out std_logic_vector(width-1 downto 0);
    inclock:in std_logic := '0';
    inclocken:in std_logic := '1';
    oe : in std_logic := '1';
    oe_out:out std_logic_vector(width-1 downto 0);
    outclock:in std_logic := '0';
    outclocken:in std_logic := '1';
    padio: inout std_logic_vector(width-1 downto 0);
    sclr: in std_logic := '0';
    sset: in std_logic := '0'
);
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL component declaration.

```
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
```


This chapter provides additional information about the document and Altera.

Document Revision History

The following table lists the revision history for this document.

Date	Version	Changes
January 2013	6.1	Updated to correct content error in “DDR I/O Timing” on page 3–7.
February 2012	6.0	Updated to refelect new GUI changes.
September 2010	5.0	Added ports and parameters.
June 2007	4.2	Updated for Quartus II software version 7.1: <ul style="list-style-type: none"> ■ Updated for Arria GX and Cyclone III devices. ■ Updated and renamed “DDR MegaWizard Plug-Ins Page Descriptions” section. ■ Added parameter to the ALTDDIO_IN megafunction. ■ Added “Referenced Documents” section. Updated “Revision History” and “How to Contact Altera” sections.
March 2007	4.1	Added Cyclone III device to list of supported devices.
July 2006	4.0	Updated to reflect Quartus II 6.0 release, added ModelSim simulation information, updated design examples.
March 2005	3.0	Updated to reflect new GUI changes.
December 2004	2.0	Updated to reflect new document organization and GUI changes.

How to Contact Altera

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











Contact ⁽¹⁾	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Nontechnical support (general) (software licensing)	Email	nacomp@altera.com
	Email	authorization@altera.com

Note to Table:

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

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, Save As dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, <code>\qdesigns</code> directory, D: drive, and <code>chiptrip.gdf</code> file.
<i>Italic Type with Initial Capital Letters</i>	Indicate document titles. For example, <i>Stratix IV Design Guidelines</i> .
<i>italic type</i>	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, <file name> and <project name>.pdf file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, <code>data1</code> , <code>tdi</code> , and <code>input</code> . The suffix <code>n</code> denotes an active-low signal. For example, <code>resetn</code> . Indicates command line commands and anything that must be typed exactly as it appears. For example, <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword <code>SUBDESIGN</code>), and logic function names (for example, <code>TRI</code>).
	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	The question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	The multimedia icon directs you to a related multimedia presentation.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.
	The feedback icon allows you to submit feedback to Altera about the document. Methods for collecting feedback vary as appropriate for each document.