

altddio Megafunction

User Guide



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

Software Version:	7.1
Document Version:	4.2
Document Date:	June 2007

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. 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 Corporation. 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.



UG-DDRMGAFTN-4.2

About This User Guide

Revision History	v
Referenced Documents	v
How to Contact Altera	vi
Typographic Conventions	vi

Chapter 1. About These Megafunctions

Device Family Support	1-1
Introduction	1-1
Features	1-2
General Description	1-2
Common Applications	1-2
DDR SDRAM, DDR2 SDRAM and RLDRAM II Memory Interfaces	1-3
QDR SRAM and QDRII SRAM Memory Interfaces	1-3
High-Speed Interface Applications	1-3
DDR Device Configuration	1-4
Input Configuration	1-4
Output Configuration	1-5
Bidirectional Configuration	1-8
DDR I/O Timing	1-10
Resource Utilization and Performance	1-11

Chapter 2. Getting Started

Software and System Requirements	2-1
MegaWizard Plug-In Manager Customization	2-1
DDR MegaWizard Plug-Ins Page Descriptions	2-2
altdio_in Megafunction Configuration	2-4
altdio_out Megafunction Configuration	2-6
altdio_bidir Megafunction Configuration	2-8
altdio_in, altdio_out and altdio_bidir Simulation Libraries	2-11
Instantiating Megafunctions in HDL Code or Schematic Designs	2-13
Generating a Netlist for EDA Tool Use	2-14
Using the Port and Parameter Definitions	2-14
SignalTap II Embedded Logic Analyzer	2-15
Design Example 1: 8-Bit DDR Divider Using altdio_in and altdio_out	2-16
Design Files	2-16
Summary	2-16
Generate a Divider Using altdio_in and altdio_out	2-16
Create the altdio_in Module	2-16
Create the altdio_out Module	2-22
Create the lpm_divide Module	2-26

Create a Divider	2-32
Implement the Divider Design	2-33
Functional Results—Simulate the Design in Quartus II Simulator	2-34
Functional Results—Simulate the Divider Design in ModelSim-Altera	2-35
Design Example 2: 8-Bit DDR Divider Using altddio_bidir	2-37
Design Files	2-37
Summary	2-37
Generate a Divider Using altddio_bidir	2-37
Create the altddio_bidir Module	2-37
Create the lpm_divide Module	2-42
Create a Divider	2-43
Implement the Divider Design	2-44
Functional Results—Simulate the Design in Quartus II Simulator	2-45
Functional Results—Simulate the Divider Design in ModelSim-Altera	2-47
Conclusion	2-48

Chapter 3. Specifications

Ports and Parameters	3-1
Ports and Parameters for the altddio_in Megafunction	3-1
Ports and Parameters for the altddio_out Megafunction	3-3
Ports and Parameters for the altddio_bidir Megafunction	3-6



About This User Guide

Revision History The following table shows the revision history for this user guide.

Date and Document Version	Changes Made	Summary of Changes
June 2007 v4.2	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 “Referenced Documents” section.• Updated “Revision History” and “How to Contact Altera” sections.	Updated for Quartus II version 7.1 by adding information about Arria GX and Cyclone III devices.
March 2007 v4.1	Added Cyclone III device to list of supported devices.	Updated for Quartus II version 7.0 by adding support for Cyclone III device.
July 2006 v4.0	Updated to reflect Quartus II 6.0 release, added ModelSim simulation information, updated design examples	
March 2005 v3.0	Updated to reflect new GUI changes	
December 2004 v2.0	Updated to reflect new document organization and GUI changes	

Referenced Documents

This user guide references the following documents:

- *External Memory Interfaces in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook*
- *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*
- *Stratix II Architecture* chapter in volume 1 of the *Stratix II Device Handbook*
- *APEX II Programmable Logic Device Family Data Sheet*
- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Synthesis* section in volume 1 of the *Quartus II Handbook*
- *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*

How to Contact Altera

For 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/mysupport
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Altera literature services	Email	literature@altera.com
Non-technical 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

This document uses the typographic conventions shown in the following table.

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, filenames, filename extensions, and software utility names are shown in bold type. Examples: f_{MAX} , lqdesigns directory, d: drive, chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n</i> + 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."

Visual Cue	Meaning
Courier type	Signal and port names are shown in lowercase Courier type. Examples: <code>data1</code> , <code>tdi</code> , <code>input</code> . Active-low signals are denoted by suffix <code>n</code> , e.g., <code>resetn</code> . Anything that must be typed exactly as it appears is shown in Courier type. For example: <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword <code>SUBDESIGN</code>), as well as logic function names (e.g., <code>TRI</code>) 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.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	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 about a particular topic.



1. About These Megafunctions

Device Family Support

The altdio megafunctions—altdio_in, altdio_out, and altdio_bidir—support the following target Altera® device families:

- Arria™ GX
- Stratix® III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® Stratix
- HardCopy II
- APEX™ II

Introduction

As design complexities increase, use of vendor-specific intellectual property (IP) blocks has become a common design methodology. Altera provides parameterizable megafunctions that are optimized for Altera device architectures. Using megafunctions instead of coding your own logic saves valuable design time. The Altera-provided functions offer more efficient logic synthesis and device implementation. You can scale the size of the megafunction by setting various parameters.

Features

The altdio megafunctions implement a DDR interface and offer many additional features, which include:

- The altdio_in megafunction receives data on both edges of the reference clock
- The altdio_out megafunction transmits data on both edges of the reference clock
- The altdio_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 altdio_bidir megafunction
- An output enable input for the altdio_out and altdio_bidir megafunctions

General Description

The Altera DDR I/O megafunctions let you configure the DDR I/O registers in Arria GX, Stratix series, Cyclone series, HardCopy II, HardCopy Stratix, and APEX II 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 altdio_in megafunction implements the interface for DDR inputs. The altdio_out megafunction implements the interface for DDR outputs. The altdio_bidir megafunction implements the interface for bidirectional DDR inputs and outputs.

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, RDRAM II, QDR SRAM, and QDRII 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 RDRAM II memory interfaces
- QDR SRAM and QDRII 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 DDRII SDRAM controller is available by downloading the Altera DDR SDRAM Controller MegaCore® function from **www.altera.com**.

QDR SRAM and QDRII SRAM Memory Interfaces

The QDR and QDRII SRAM standard is defined jointly by Cypress Semiconductor Corporation, Integrated Device Technology, Inc., and Micron Technology, Inc. QDR and QDRII 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. QDRII 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 resources:

- *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*

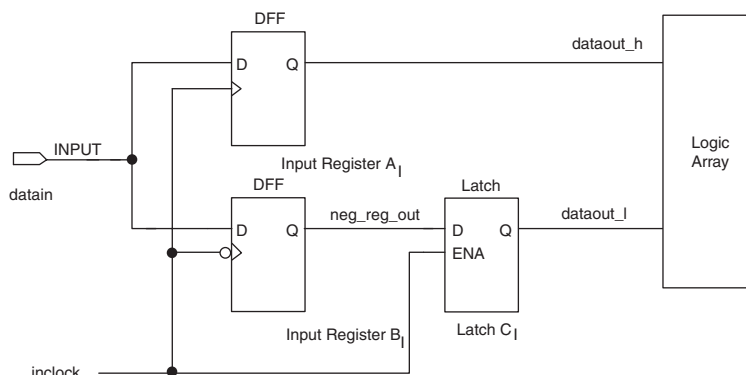
DDR Device Configuration

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 1–1** shows an IOE configured for DDR inputs for a Stratix series or APEX II device.

Figure 1–1. Input DDR I/O Path Configuration for a Stratix Series or APEX II Device *Note (1)*

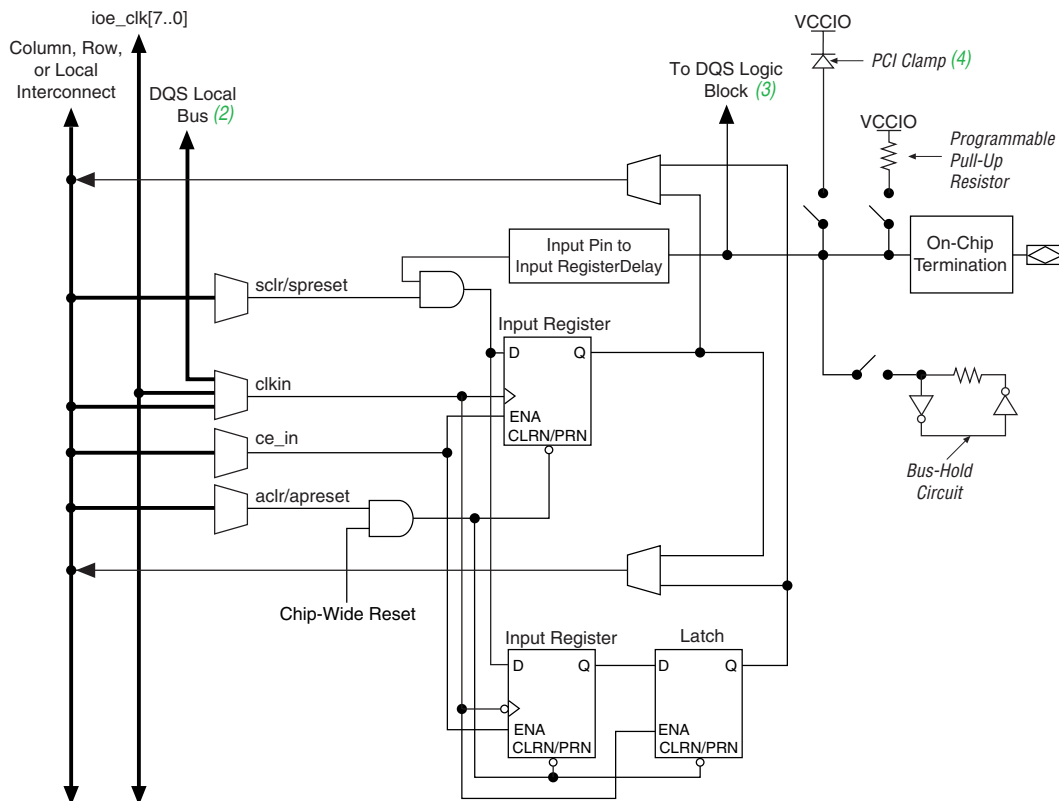


Note to Figure 1–1:

- (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 1-2 shows an IOE configured for DDR inputs for a Stratix or Stratix II device.

Figure 1-2. Stratix II IOE in DDR Input I/O Configuration Notes (1), (2), (3)



Notes to Figure 1-2:

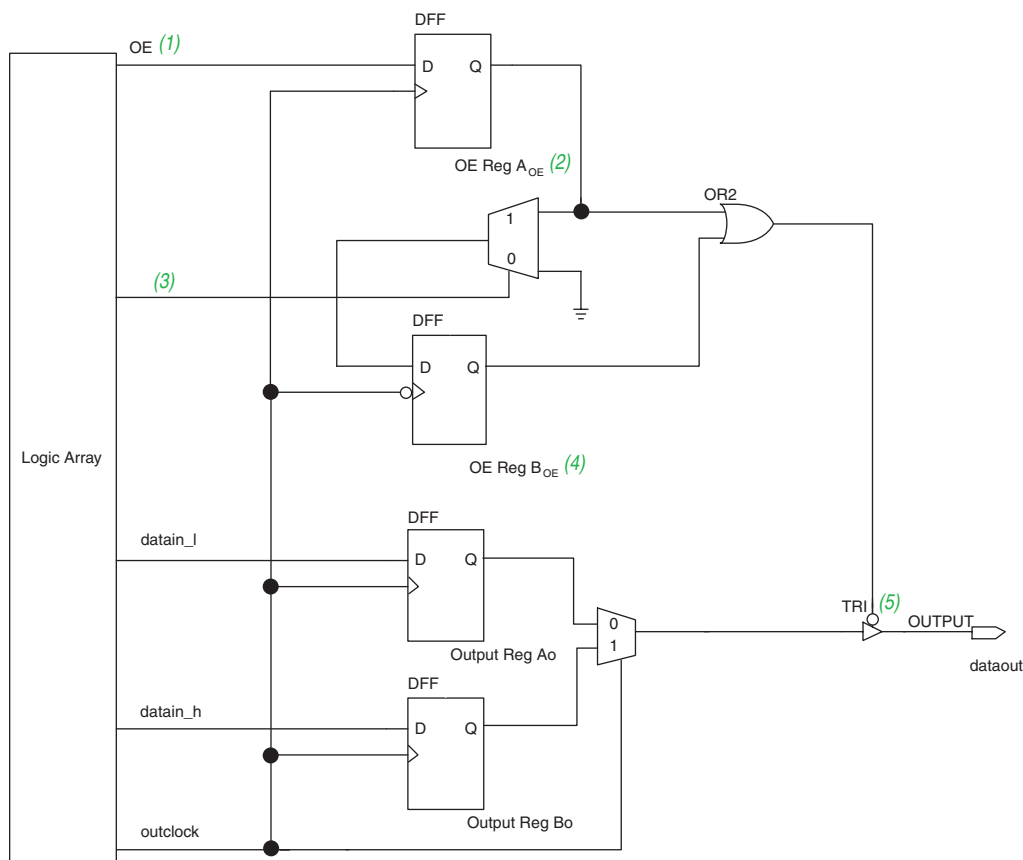
- (1) All input signals to the IOE can be inverted at the IOE.
- (2) This signal connection is only allowed on dedicated DQ function pins.
- (3) This signal is for dedicated DQS function pins only.
- (4) The optional PCI clamp is only available on column I/O pins.

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 1-3 shows the IOE configuration for DDR outputs in Stratix series and APEX II devices.

Figure 1–3. Output DDR I/O Path Configuration for Stratix Series and APEX II Devices

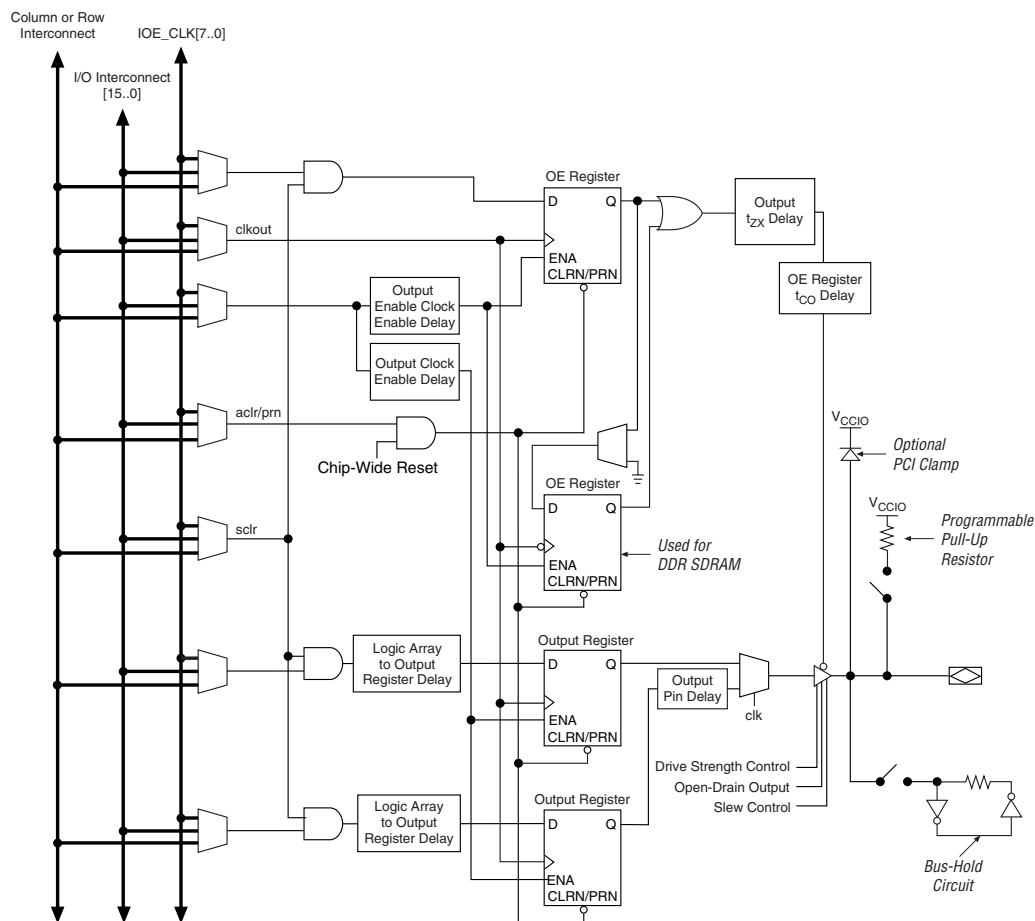
**Notes to Figure 1–3:**

- (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 1-4 shows the IOE configuration for DDR outputs in Stratix series devices.

Figure 1-4. 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.

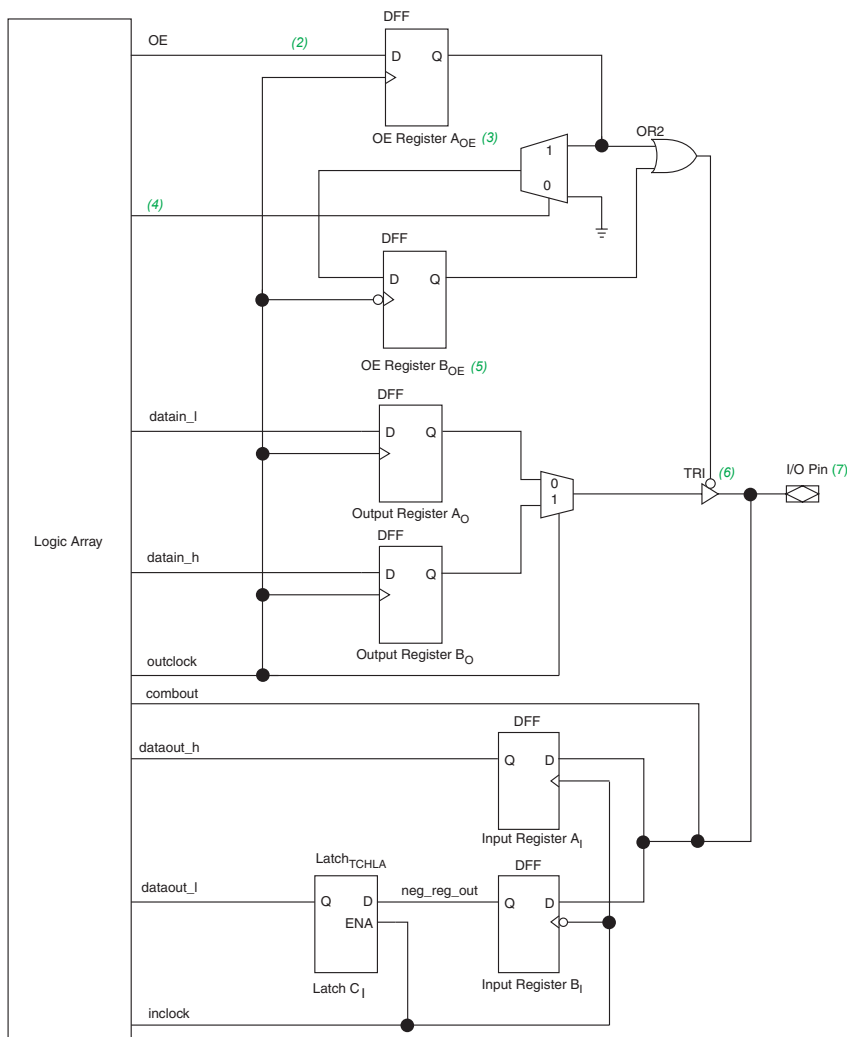
The bidirectional path consists of two data flow paths:

- 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 1-1 on page 1-4](#). 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 1-3 on page 1-6](#). 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 1-5](#) shows the bidirectional DDR I/O configuration for Stratix series and APEX II devices.

Figure 1–5. Bidirectional DDR I/O Path Configuration *Note (1)*



Notes to Figure 1–5:

- (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 sources:

- 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 1-6 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_h`, and the output of register A_1 is `dataout_l`. `dataout_h` and `dataout_l` feed the logic array and show the conversion of the data from a DDR implementation to positive-edge triggered data.

Figure 1-6. DDR I/O Input Timing Waveform

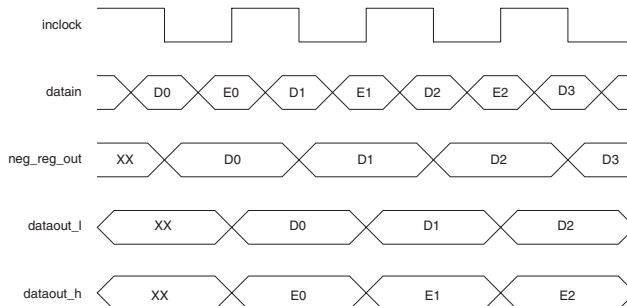
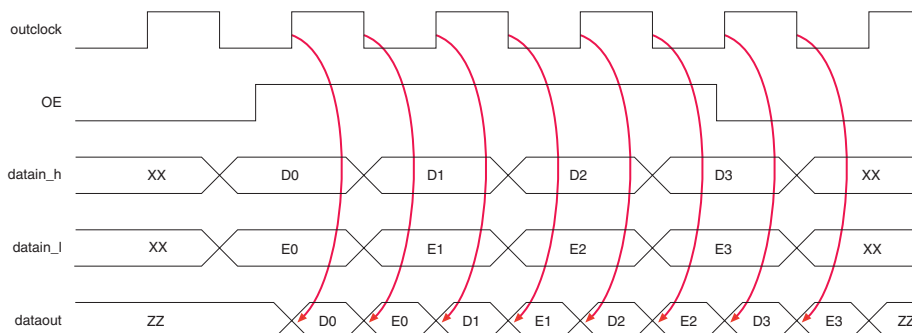


Figure 1-7 shows a functional timing waveform example for the output path with the output enable registered. In this example, the delay switch-on by a half clock cycle is not turned on, so the second output enable register (B_{OE}) is not used. The output enable signal `OE` is active high and can be 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 1–7. DDR I/O Output Timing Waveform



The waveform in [Figure 1–7](#) 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.

Resource Utilization and Performance

For precise details about the resource utilization of the `altdio_in`, `altdio_out`, and `altdio_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.

Software and System Requirements

The instructions in this section require the following software:

- For operating system support information, refer to:
[//www.altera.com/support/software/os_support/oss-index.html](http://www.altera.com/support/software/os_support/oss-index.html)
- Quartus® II software

MegaWizard Plug-In Manager Customization

The MegaWizard® Plug-In Manager allows you to create or modify design files containing custom megafunction variations, which you can then instantiate in a design file. With the MegaWizard Plug-In Manager, you can specify custom megafunction variation options. The MegaWizard Plug-In Manager provides a wizard for each of the `altdio_in`, `altdio_out`, `altdio_bidir` megafunctions that allows you to set the megafunction's features—parameter values and optional ports—in the design.

Start the MegaWizard Plug-In Manager in one of the following ways:

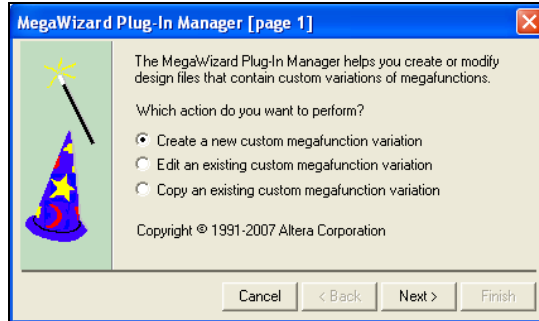
- On the Tools menu, choose the **MegaWizard Plug-In Manager** command.
- When working in the Block Editor, from the Edit menu, click **Insert Symbol as Block**, or right-click in the Block Editor, point to **Insert**, and click **Symbol as Block**. In the **Symbol** dialog box, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the **MegaWizard Plug-In Manager** by typing the following command at the command prompt:
`qmegawiz` ↵

DDR MegaWizard Plug-Ins Page Descriptions

This section provides descriptions of the options available on the individual pages of the DDR MegaWizard Plug-Ins, also known as wizards.

In page 1 of the MegaWizard Plug-In Manager, you can create, edit, or copy a custom megafunction variation ([Figure 2-1](#)).

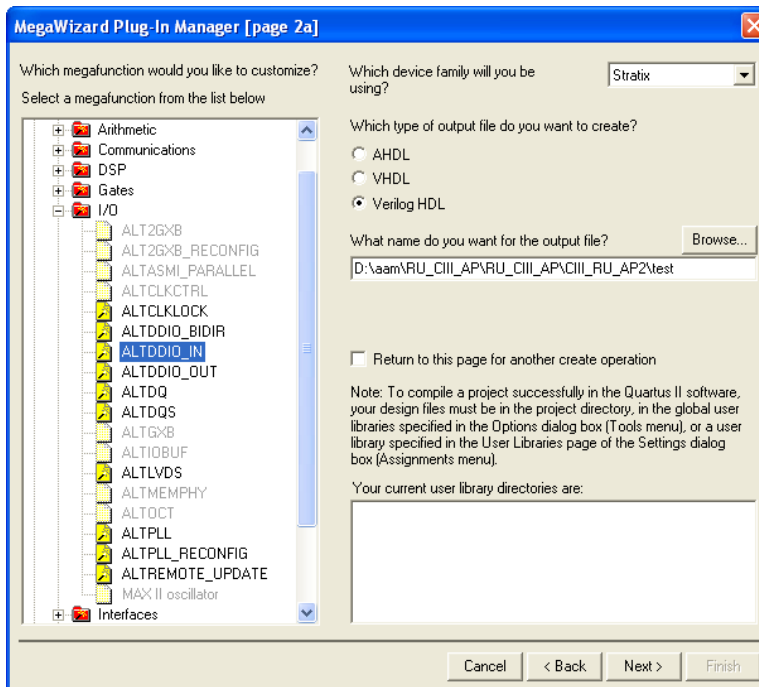
Figure 2-1. Create, Edit, or Copy a Megafunction Variation



On page 2a ([Figure 2-2](#)), specify the device family you want to use, type of output file to create, and the name of the output file.

Select the megafunction you're creating from the list on the left of the window. Choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type ([Figure 2-2](#)).

Figure 2–2. MegaWizard Plug-In Manager [page 2a]



Generate the required variation of each DDR megafunction using the wizard. Click **Documentation** (Figure 2–3) for more information.

Each DDR megafunction has its own sequence of pages in the wizard pages where you specify the required configuration details. These pages are described in the following sections.

altddio_in Megafunction Configuration

Use pages 3 and 4 of the wizard to specify the initial altddio_in megafunction parameters. On page 3 of the altddio_in wizard, specify customizable parameters for the device family, data bus width, and type of asynchronous reset (Figure 2–3). A description of each function on page 3 is shown in Table 2–1.

Figure 2–3. MegaWizard Plug-In Manager—altddio_in [page 3 of 6]

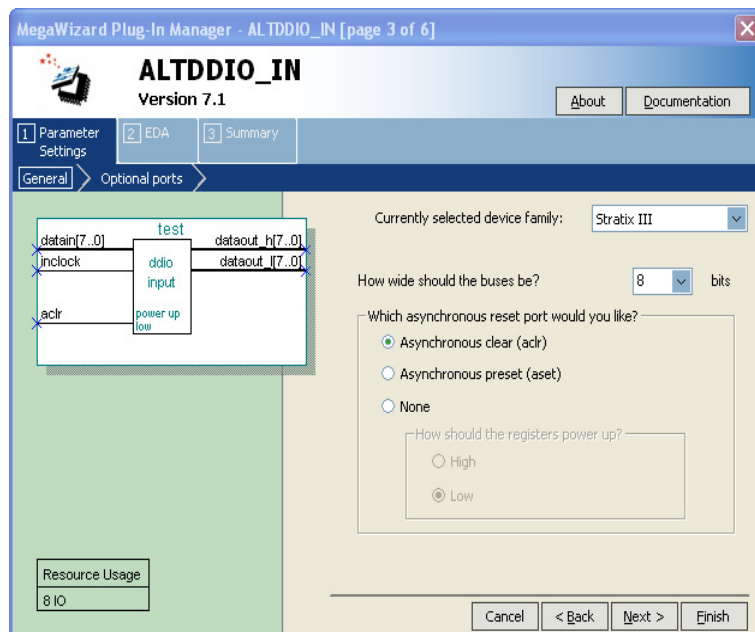


Table 2–1. altddio_in Wizard Page 3 Options

Function	Description
Currently selected device family	Specify the Altera® device family you are using. The Arria™ GX, Stratix® III, Stratix II, Stratix II GX, Stratix, Stratix GX, Cyclone® III, Cyclone II, Cyclone, Hardcopy® II, HardCopy Stratix, and APEX™ II device families are available.
How wide should the buses be?	Specify the width of the data buses.
Which asynchronous reset port would you like?	Use asynchronous clear (aclr) or asynchronous preset (aset) as asynchronous reset. If you do not use either clear option, you must specify whether registers should power up high or low.

On page 4 of the `altdio_in` wizard, specify whether to create a clock enable port, whether to invert the input clock, and the type of synchronous reset (Figure 2–4). A description of each function is shown in Table 2–2.

Figure 2–4. MegaWizard Plug-In Manager—`altdio_in` [page 4 of 6]

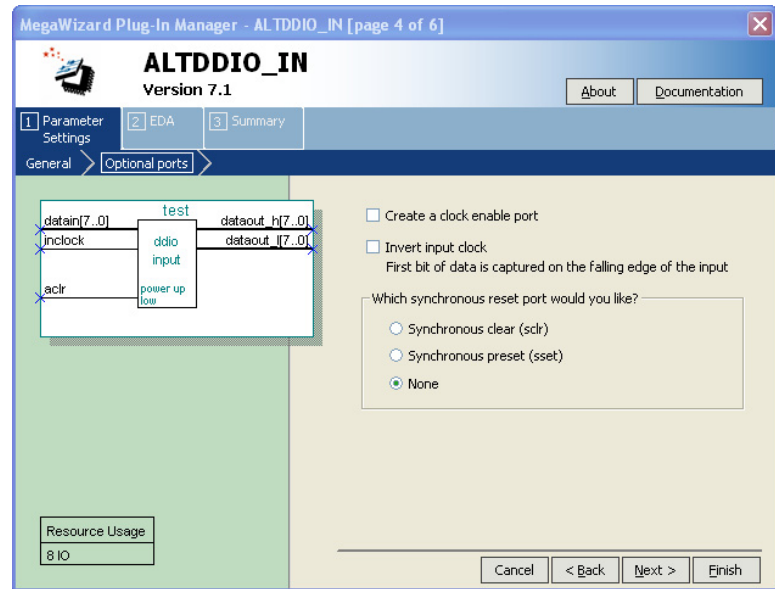


Table 2–2. `altdio_in` Wizard Page 4 Options

Function	Description
Create a clock enable port	You can add a clock enable port to control when data 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.
Which synchronous reset port would you like?	Use synchronous clear (<code>sclr</code>) or synchronous preset (<code>sset</code>) as synchronous reset. If you do not use either clear option, you must specify whether registers should power up high or low. 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.

altd dio_out Megafunction Configuration

Use pages 3 and 4 of the wizard to specify the initial altd dio_out megafunction parameters. On page 3 of the altd dio_out wizard, specify customizable parameters for the device family, the data bus width, the type of asynchronous reset, and the clock enable port (Figure 2–5). A description of each function on page 3 is shown in Table 2–3.

Figure 2–5. MegaWizard Plug-In Manager—altd dio_out [page 3 of 6]

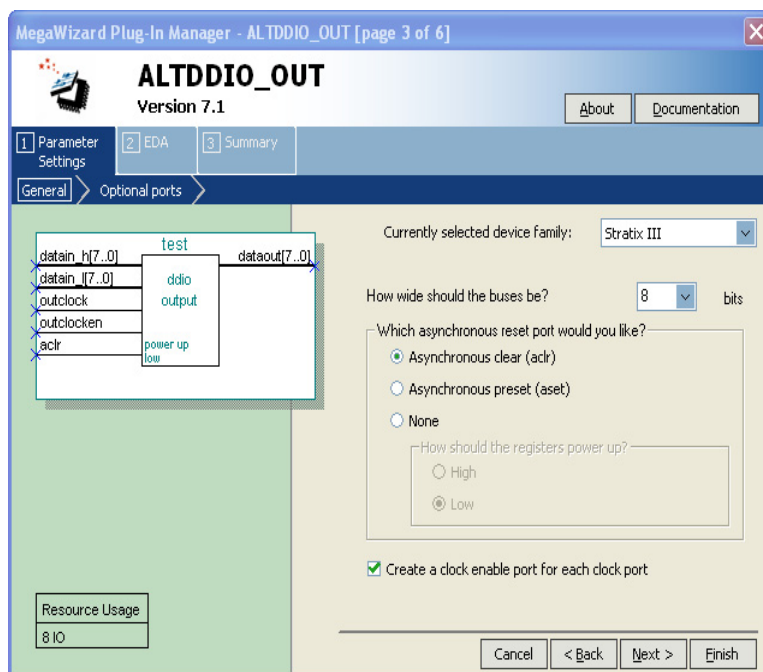


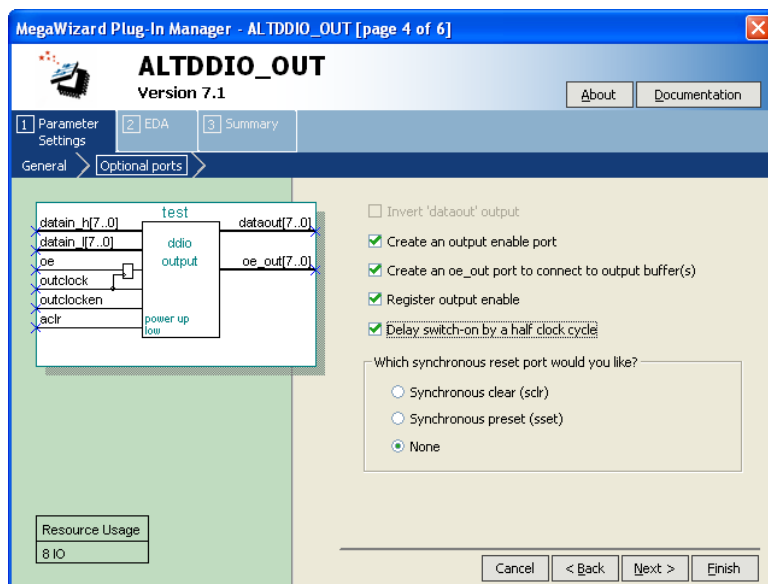
Table 2–3. altd dio_out Wizard Page 3 Options (Part 1 of 2)

Function	Description
Currently selected device family	Specify the Altera device family you are using. The Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, Cyclone III, Cyclone II, Cyclone, Hardcopy II, HardCopy Stratix, and APEX II device families are available.
How wide should the buses be?	Specify the width of the data buses.

Table 2–3. *altdio_out Wizard Page 3 Options (Part 2 of 2)*

Function	Description
Which asynchronous reset port would you like?	Use asynchronous clear (<code>ac1r</code>) or asynchronous preset (<code>aset</code>) as asynchronous reset. If you do not use either option, you must specify whether the registers should power up high or low.
Create a clock enable port for each clock port	You can add a clock enable port to control when data is clocked in. This signal prevents data from being passed through.

On page 4 of the `altdio_out` wizard, specify whether to invert the output data, whether to create an output enable port with the options to connect the port to output buffers, register it, and extend the high-impedance state for a half clock cycle, and the type of synchronous reset (Figure 2–6). A description of each function is shown in Table 2–4.

Figure 2–6. MegaWizard Plug-In Manager—*altdio_out* [page 4 of 6]Table 2–4. *altdio_out Wizard Page 4 Options (Part 1 of 2)*

Function	Description
Invert 'dataout' output	Specifies whether to invert the <code>dataout[]</code> output port. This option is available for Cyclone III and Cyclone II devices only.
Create an output enable port	Specify whether to create an output enable input port (<code>oe</code>) to control when the data is set out to the <code>dataout</code> port.

Table 2–4. *altdio_out Wizard Page 4 Options (Part 2 of 2)*

Function	Description
Create an oe_out port to 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 output enable	When enabled, this option registers the output-enable (<code>oe</code>) input port.
Delay switch-on by a half clock cycle	Specifies whether an additional <code>oe</code> register should be used. When this second <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.
Which synchronous reset port would you like?	Use synchronous clear (<code>sc1r</code>) or synchronous preset (<code>sset</code>) as synchronous reset. If you do not use either clear option, you must specify whether registers should power up high or low. 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.

altdio_bidir Megafunction Configuration

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.

Use pages 3 and 4 of the `altdio_bidir` wizard to specify the initial `altdio_bidir` megafunction parameters. On page 3 of the `altdio_bidir` wizard, specify customizable parameters for the device family, bus width, and type of asynchronous reset, and whether to invert the `padio` port, create an output enable port, create an `oe_out` port to connect to the bidirectional buffers, register the output enable signal and delay switch on by a half cycle (Figure 2–7). A description of each function on page 3 is shown in Table 2–5.

Figure 2-7. MegaWizard Plug-In Manager—altdio_bidir [page 3 of 6]

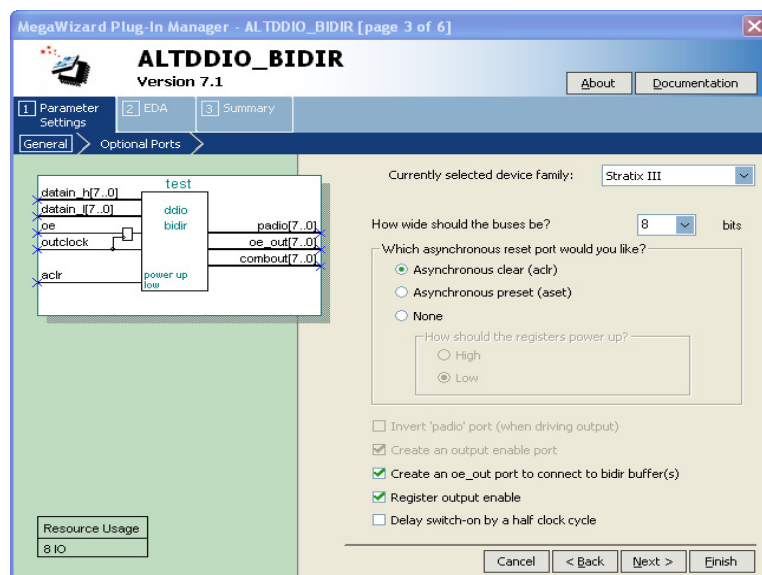


Table 2-5 shows the features and settings on page 3 of the altdio_bidir wizard.

Table 2-5. altdio_bidir Wizard Page 3 Options (Part 1 of 2)

Function	Description
Currently selected device family	Specify the Altera device family you are using. The Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, Cyclone III, Cyclone II, Cyclone, Hardcopy II, HardCopy Stratix, and APEX II device families are available.
How wide should the buses be?	Specify the width of the data buses.
Which asynchronous reset port would you like?	You can use asynchronous clear (aclr) or asynchronous preset (aset) as asynchronous reset. If you do not use either clear option, you must specify whether the should power up high or low.
Invert 'padio' port (when driving output)	The 'padio' port is inverted whenever driven as an output. This option is available for Cyclone III and Cyclone II devices only.
Create an output enable port	Specify whether to create an output enable input port (oe) to control when the data is set out to the dataout port.
Create an oe_out port to connect to bidir buffer(s)	Output enable for the bidirectional padio port. This port is available for Stratix III and Cyclone III devices only.

Table 2–5. *altddio_bidir* Wizard Page 3 Options (Part 2 of 2)

Function	Description
Register output enable	When enabled, this option registers the output-enable (oe) input port.
Delay switch-on by a half clock cycle	Specifies whether an additional oe register should be used. When this second oe register is used, the output pin is held at high impedance for an extra half clock cycle after the oe port goes high.

On page 4 of the *altddio_bidir* wizard, specify whether to create a clock enable port for every clock port, add unregistered output ('combout'), use the *dqsundelayedout* output port (for use with dedicated DDR circuitry), use the *dataout_h* and *dataout_l* ports, and implement input registers in LEs, and the type of synchronous reset (Figure 2–8). A description of each function on page 4 is shown in Table 2–6.

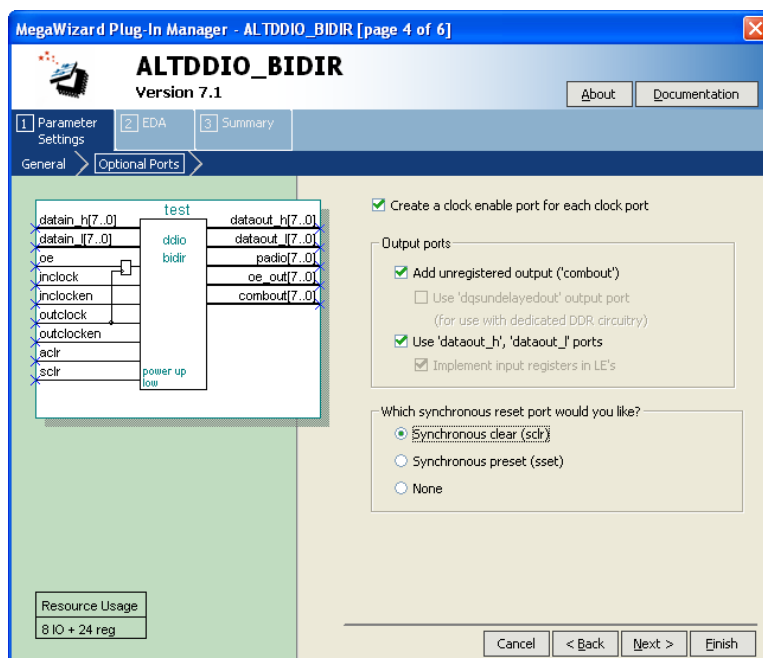
Figure 2–8. MegaWizard Plug-In Manager—*altddio_bidir* [page 4 of 6]

Table 2-6 shows the features and settings on page 4 of the `altdio_bidir` wizard.

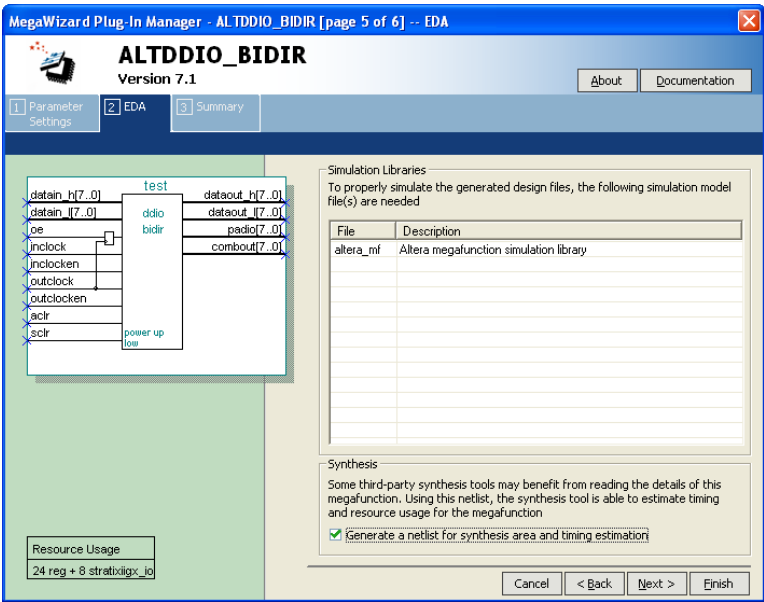
Table 2-6. <i>altdio_bidir</i> Wizard Page 4 Options	
Function	Description
Create a clock enable port for each clock port	You can add a clock enable port to control when data is clocked in. This signal prevents data from being passed through.
Output ports: Add unregistered output ('combout')	Use the 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' output port (for use with dedicated DDR circuitry)	Creates undelayed output from the <code>DQS</code> pins. If you use the <code>altdio_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.
Output ports: Use 'dataout_h', 'dataout_l' ports	Enables the <code>dataout_h</code> and <code>dataout_l</code> ports.
Output ports: Implement input registers in LE's	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.
Which synchronous reset port would you like?	Use synchronous clear (<code>sc1r</code>) or synchronous preset (<code>sset</code>) as synchronous reset. If you do not use either clear option, you must specify whether registers should power up high or low. 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.

`altdio_in`, `altdio_out` and `altdio_bidir` Simulation Libraries

In the `altdio_in`, `altdio_out`, and `altdio_bidir` wizards, page 5 shows a list of the libraries needed to properly simulate the design files (Figure 2-9).

On page 5 of each `altdio` wizard, you can enable the Quartus II software to generate a synthesis area and timing estimation netlist for the megafunction for use by third-party tools.

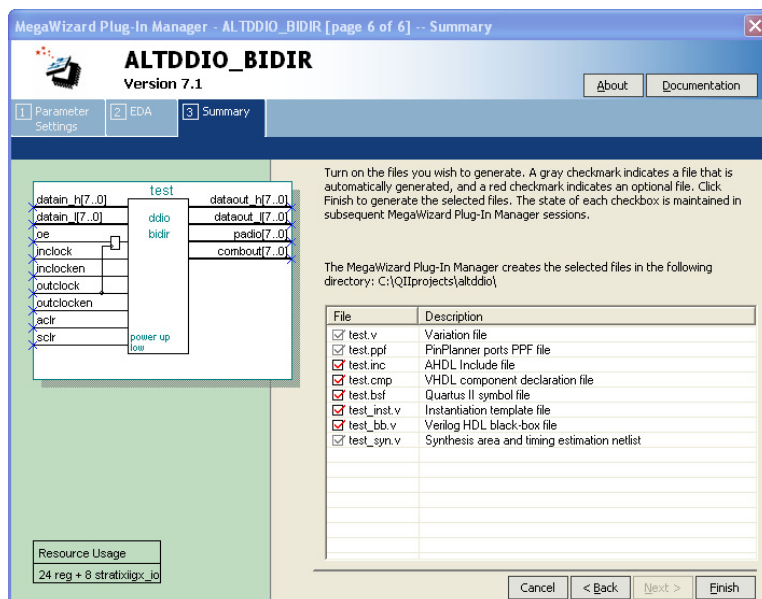
Figure 2–9. MegaWizard Plug-In Manager—altdddio_in, altdddio_out, or altdddio_bidir [page 5 of 6] — EDA



On the final page of the wizard, 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 red check mark) (Figure 2–10).

Click **Finish** to create an instance of the megafunction.

Figure 2–10. MegaWizard Plug-In Manager -- Final Page



Instantiating Megafunctions in HDL Code or Schematic Designs

When you use the MegaWizard Plug-In Manager to customize and parameterize a megafunction, it creates a set of output files that allow you to instantiate the customized function in your design. Depending on the language you choose in the MegaWizard Plug-In Manager, it instantiates the megafunction with the correct parameter values and generates a megafunction variation file (wrapper file) in Verilog HDL (.v), VHDL (.vhd), or AHDL (.tdf), along with other supporting files.

The MegaWizard Plug-In Manager provides options to create the following files:

- A sample instantiation template for the language of the variation file (`_inst.v`, `_inst.vhd`, or `_inst.tdf`)

- Component Declaration File (**.cmp**) that can be used in VHDL Design Files
- AHDL Include File (**.inc**) that can be used in Text Design Files (**.tdf**)
- Quartus II Block Symbol File (**.bsf**) that can be used in schematic designs
- Verilog HDL module declaration file that can be used when instantiating the megafunction as a black box in a third-party synthesis tool (**_bb.v**)



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

Generating a Netlist for EDA Tool Use

If you use a third-party EDA synthesis tool, you can instantiate the megafunction variation file as a black box for synthesis. Use the VHDL component declaration or Verilog module declaration black-box file to define the function in your synthesis tool, and then include the megafunction variation file in your Quartus II project.

If you enable the option to generate a synthesis area and timing estimation netlist in the wizard, it generates an additional netlist file (**_syn.v**). The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of the architectural elements in the megafunction but may not represent true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to focus timing-driven optimizations and improve the quality of results.



For more information about using megafunctions in your third-party synthesis tool, refer to the appropriate chapter in the *Synthesis* section in volume 1 of the *Quartus II Handbook*.

Using the Port and Parameter Definitions

Instead of the MegaWizard Plug-In Manager, you can instantiate the megafunction directly in your Verilog HDL, VHDL, or AHDL code by calling the megafunction and settings its parameters as you would any other module, component, or subdesign.



Altera strongly recommends that you use the MegaWizard Plug-In Manager for complex megafunctions. The MegaWizard Plug-In Manager ensures that all megafunction parameters are set properly.

Refer to [Chapter 3, Specifications](#) for a list of the megafunction ports and parameters.

SignalTap II Embedded Logic Analyzer

The SignalTap® II embedded logic analyzer provides you with a method of debugging all of the Altera megafunctions within your design. With the SignalTap II embedded logic analyzer, you can capture and analyze data samples for the top-level ports of the Altera megafunctions in your design while your system is running at full speed.

To monitor signals from your Altera megafunctions, you must first configure the SignalTap II embedded logic analyzer in the Quartus II software, and then include the analyzer as part of your project. The Quartus II software will then seamlessly embed the analyzer along with your design in the selected device.



For more information about using the SignalTap II embedded logic analyzer, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Design Example 1: 8-Bit DDR Divider Using altd dio_in and altd dio_out

This section presents a design example that uses the altd dio_in and altd dio_out megafunctions 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 the overall project.

Design Files

The design files are available with this user guide in the Quartus II Project section and in the User Guides section of the Altera website (www.altera.com).

Summary

In this example, you perform the following activities:

- Create a divider using the altd dio_in, altd dio_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 altd dio_in and altd dio_out

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

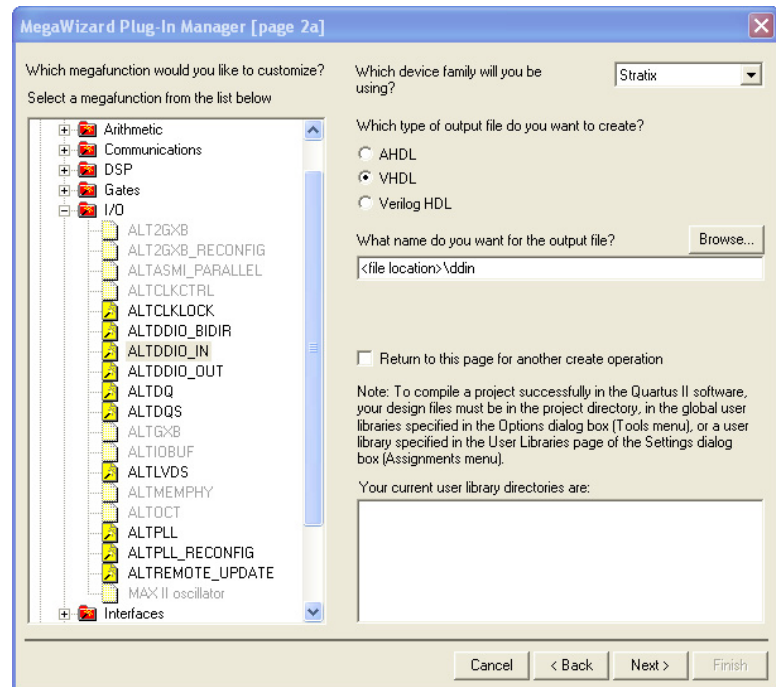
Create the altd dio_in Module

Follow these steps to create the altd dio_in module:

1. Unzip the **altd dio_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 ([Figure 2-2](#)).
5. To answer **Which megafunction would you like to customize?**, in the **I/O** folder, select **ALTDDIO_IN**.
6. To answer **Which device family will you be using?**, select **Stratix**.

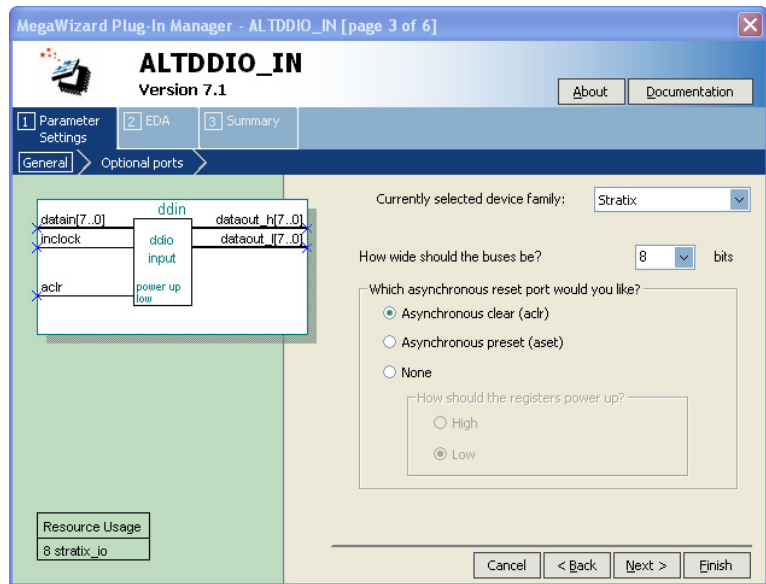
7. To answer **Which type of output file do you want to create?**, select **VHDL**.
8. Specify the output file **ddin**. **Figure 2-11** shows the wizard after you have made these selections.

Figure 2-11. MegaWizard Plug-In Manager [page 2a]



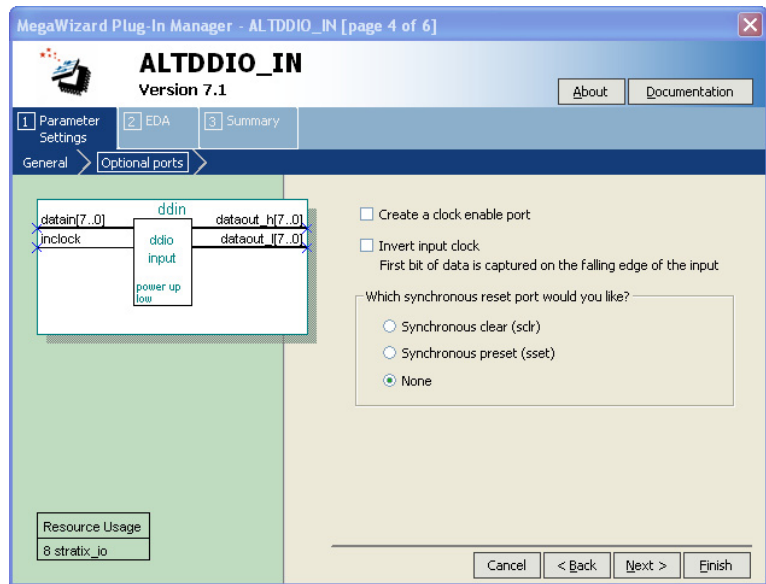
9. Click **Next**. Page 3 appears (**Figure 2-12**).

Figure 2–12. MegaWizard Plug-In Manager—altddiv_in [page 3 of 6]



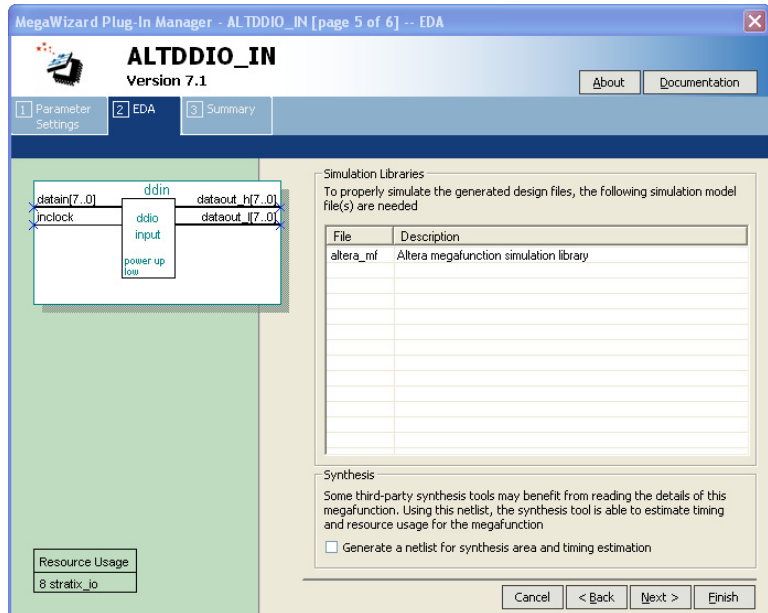
10. To answer **How wide should the buses be?** list, select **8**.
11. To answer **Which asynchronous reset port would you like?**, click **None**.
12. To answer **How should the registers power up?**, click **Low**.
13. Click **Next**. Page 4 appears (Figure 2–13).

Figure 2-13. MegaWizard Plug-In Manager—altdio_in [page 4 of 6]



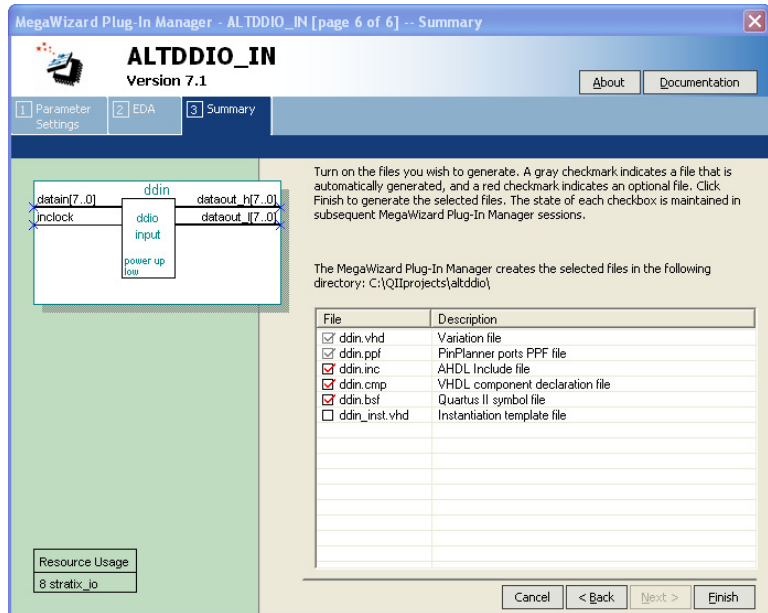
14. Turn off **Create a clock enable port** and **Invert input clock**.
15. To answer **Which synchronous reset port would you like?**, click **None**.
16. Click **Next**. Page 5 appears (Figure 2-14).

Figure 2-14. MegaWizard Plug-In Manager—altdio_in [page 5 of 6]



17. Turn off **Generate a netlist for synthesis area and timing estimation**.
18. Click **Next**. Page 6 appears (Figure 2-15).

Figure 2-15. MegaWizard Plug-In Manager—altdddio_in [page 6 of 6]



19. Turn on **VHDL component declaration file** and **Instantiation template file**.
20. Turn off **Quartus II symbol file** and **AHDL Include file**.
21. Click **Finish**.

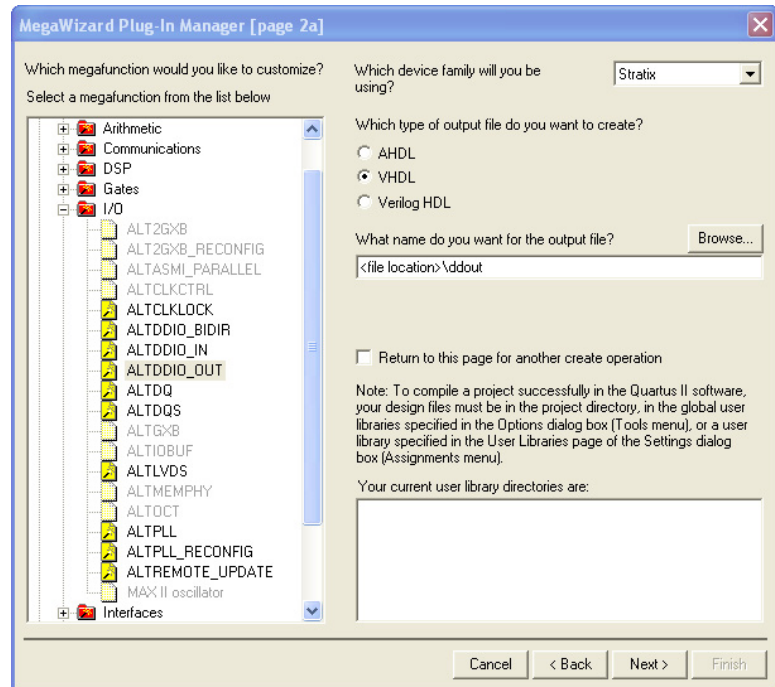
The altdddio_in module is now built.

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 (Figure 2–16).

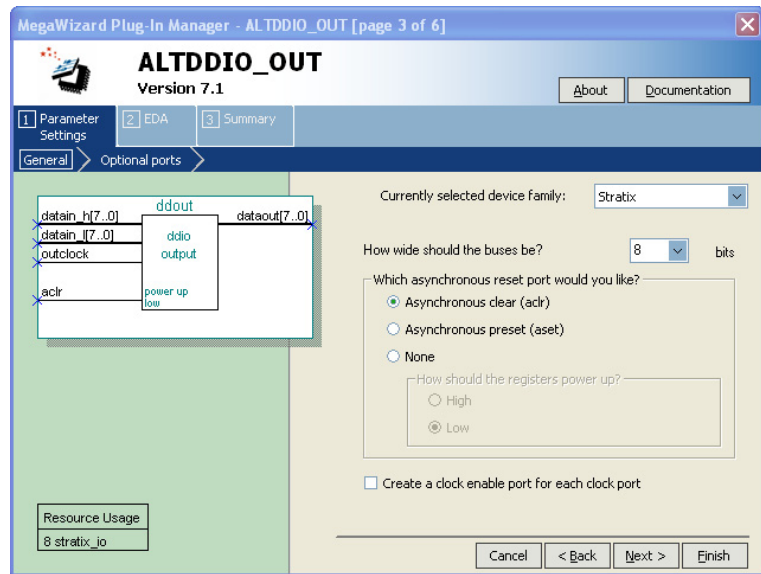
Figure 2–16. MegaWizard Plug-In Manager [page 2a]



3. To answer **Which megafunction would you like to customize?**, in the **I/O** folder, select **ALTDIO_OUT**.
4. To answer **Which device family will you be using?**, select **Stratix**.
5. To answer **Which type of output file do you want to create?**, select **VHDL**.
6. Specify the output file **ddout**.

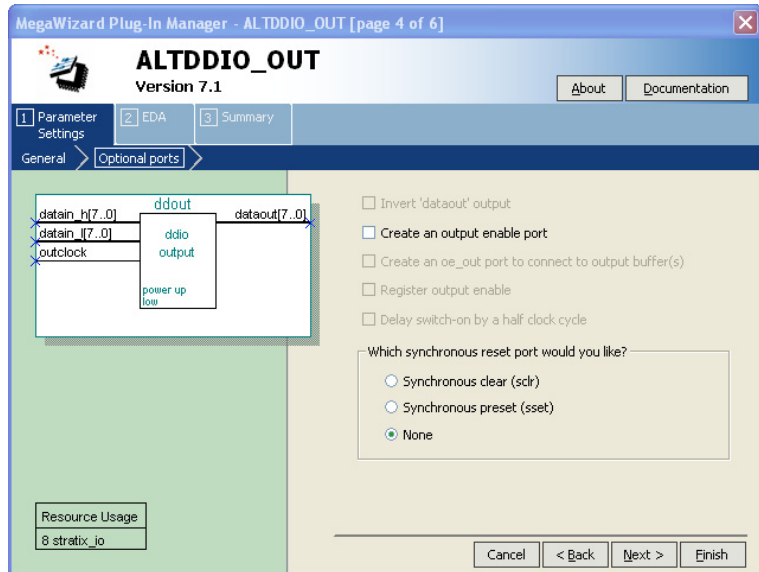
7. Click **Next**. Page 3 appears (Figure 2–17).

Figure 2–17. MegaWizard Plug-In Manager—*altdio_out* [page 3 of 6]



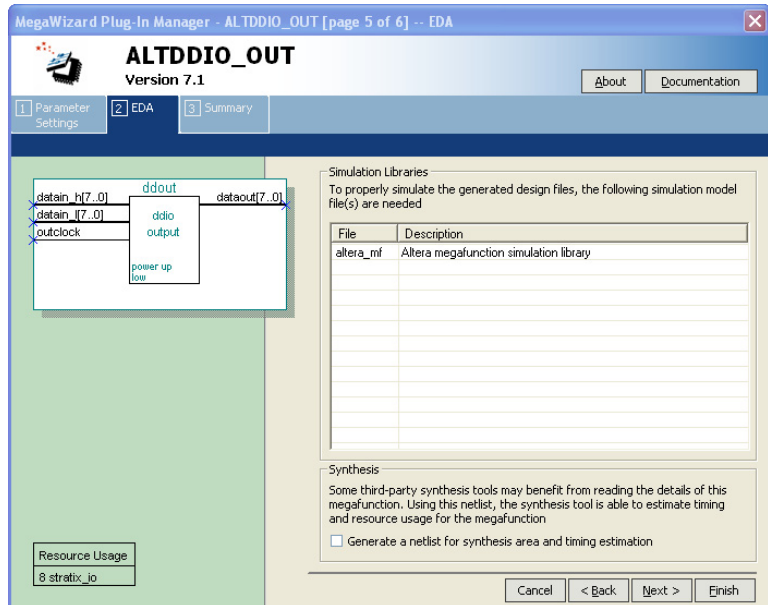
8. To answer **How wide should the buses be?**, select **8**.
9. To answer **Which asynchronous reset port would you like?**, select **None**.
10. To answer **How should the registers power up?**, select **Low**.
11. Turn off the **Create a clock enable port for each clock port** option.
12. Click **Next**. Page 4 appears (Figure 2–18).

Figure 2–18. MegaWizard Plug-In Manager—altdddio_out [page 4 of 6]



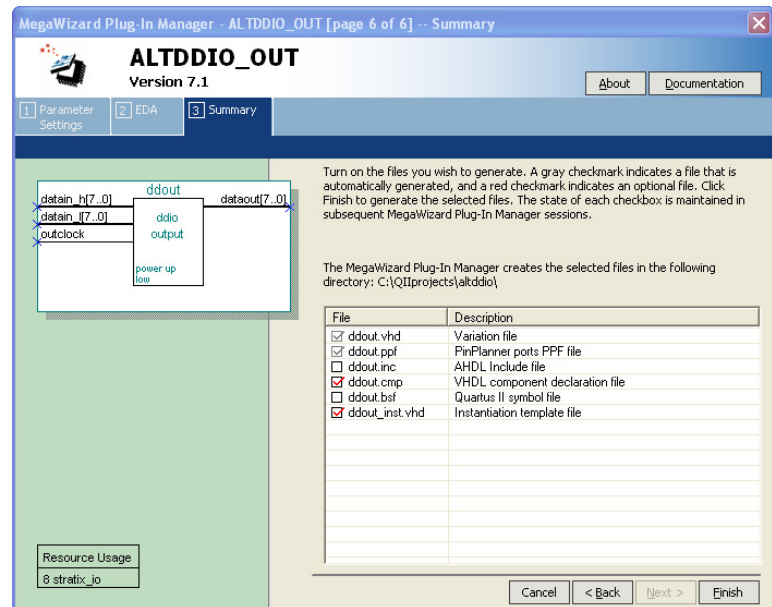
13. Turn off **Create an output enable port**.
14. To answer **Which synchronous reset port would you like?**, click **None**.
15. Click **Next**. Page 5 appears (Figure 2–19).

Figure 2-19. MegaWizard Plug-In Manager—altdddio_out [page 5 of 6]



16. Turn off **Generate a netlist for synthesis area and timing estimation**.
17. Click **Next**. Page 6 appears (Figure 2-20).

Figure 2–20. MegaWizard Plug-In Manager—altdio_out [page 6 of 6]



18. Turn on the **VHDL component declaration file** and **Instantiation template file** options.
19. Turn off the **Quartus II symbol file** and **AHDL Include file** options.
20. Click **Finish**.

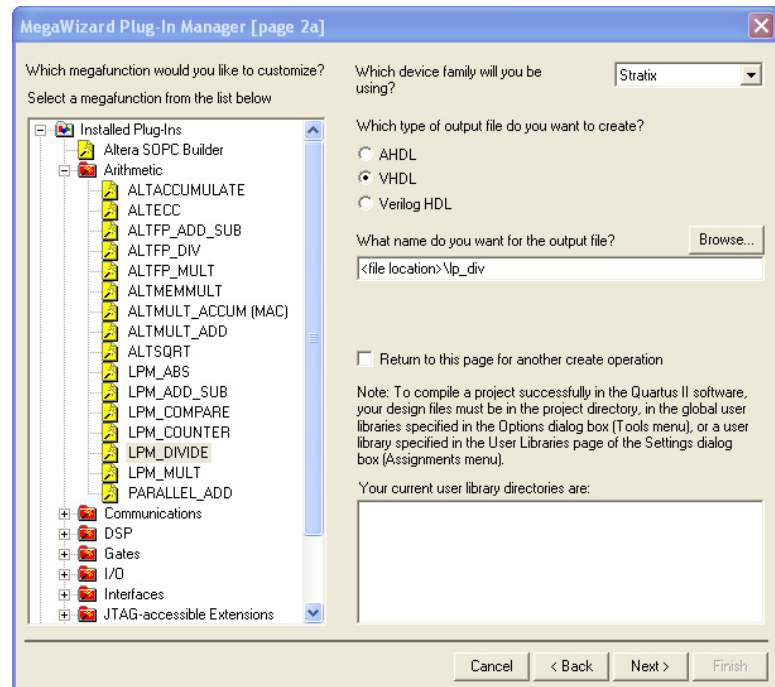
The `altdio_out` module is now built.

Create the `lpm_divide` Module

Follow these steps to create the `lpm_divide` module:

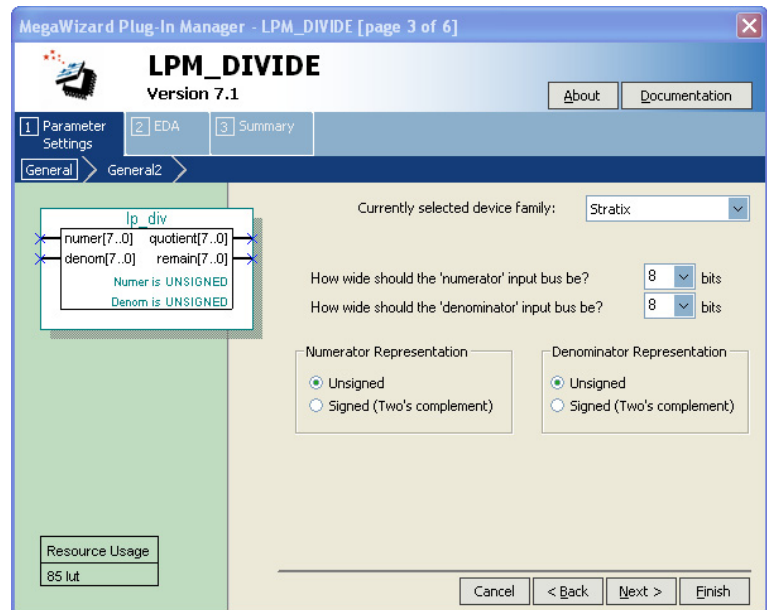
1. On the Tools menu, select **MegaWizard Plug-In Manager**.
2. In the MegaWizard Plug-In manager dialog box, select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays (Figure 2–21).

Figure 2–21. MegaWizard Plug-In Manager [page 2a]



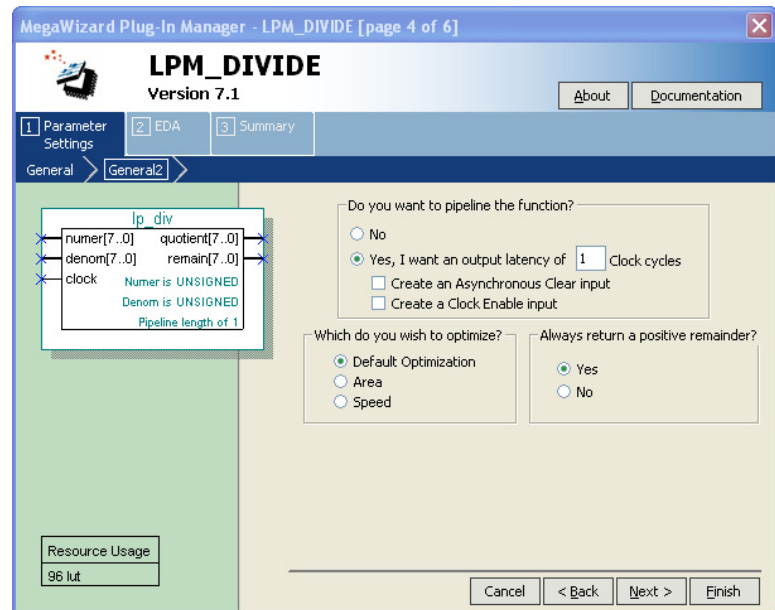
3. To answer **Which megafunction would you like to customize?**, in the **Arithmetic** folder, select **LPM_DIVIDE**.
4. To answer **Which device family will you be using?**, select **Stratix**.
5. To answer **Which type of output file do you want to create?**, select **VHDL**.
6. Specify the output file **lp_div**.
7. Click **Next**. Page 3 appears (Figure 2–22).

Figure 2–22. MegaWizard Plug-In Manager—lpm_divide [page 3 of 6]



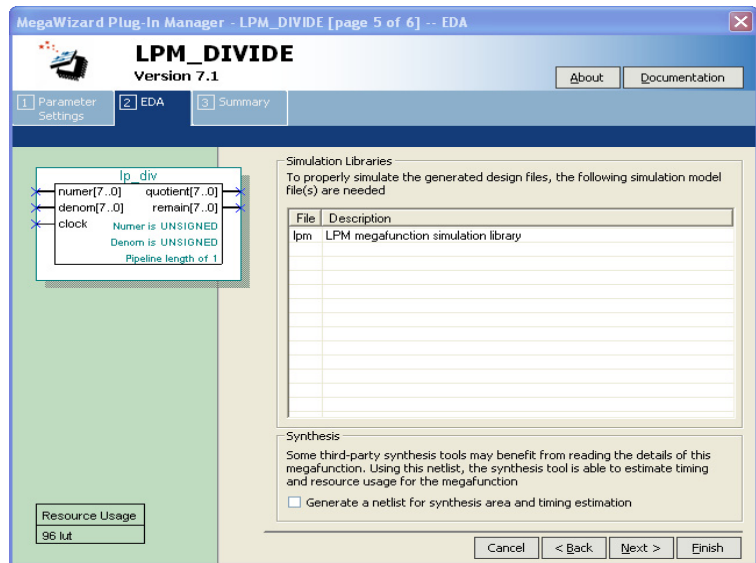
8. To answer the questions **How wide should the 'numerator' input bus be?** and **How wide should the 'denominator' input bus be?**, select **8**.
9. Under both the **Numerator Representation** and **Denominator Representation** options, select **Unsigned**.
10. Click **Next**. Page 4 appears (Figure 2–23).

Figure 2–23. MegaWizard Plug-In Manager—lpm_divide [page 4 of 6]



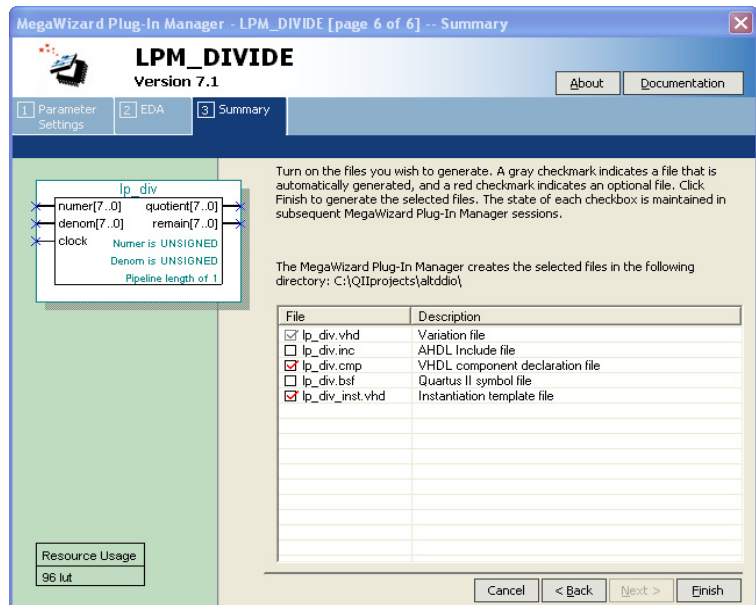
11. To answer **Do you want to pipeline the function?**, select **Yes, I want an output latency of** and type **1** in the **Clock cycles** box.
12. Turn off both the **Create an Asynchronous Clear input** and **Create a Clock Enable input** options.
13. To answer **Which do you wish to optimize?**, select **Default Optimization**.
14. To answer **Always return a positive remainder?**, select **Yes**.
15. Click **Next**. Page 5 appears (Figure 2–24).

Figure 2-24. MegaWizard Plug-In Manager—lpm_divide [page 5 of 6]



16. Turn off **Generate a netlist for synthesis area and timing estimation**.
17. Click **Next**. Page 6 appears (Figure 2-25).

Figure 2–25. MegaWizard Plug-In Manager—lpm_divide [page 6 of 6]



18. Turn on the **VHDL component declaration file** and **Instantiation template file** options.
19. Turn off the **Quartus II Symbol file** and **AHDL Include file** options.
20. Click **Finish**.

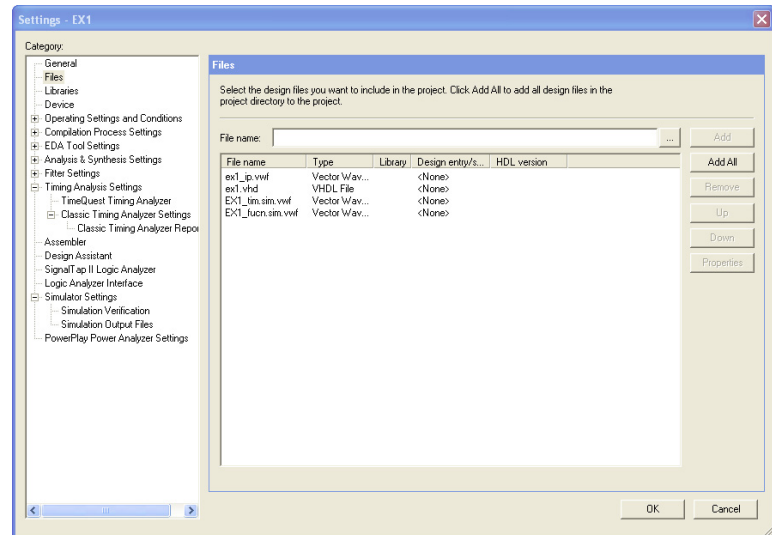
The lpm_divide module is now built.

Create a Divider

Use the following steps to combine the `altdddio_in`, `altdddio_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 (Figure 2–26).

Figure 2–26. File Settings Dialog Box



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

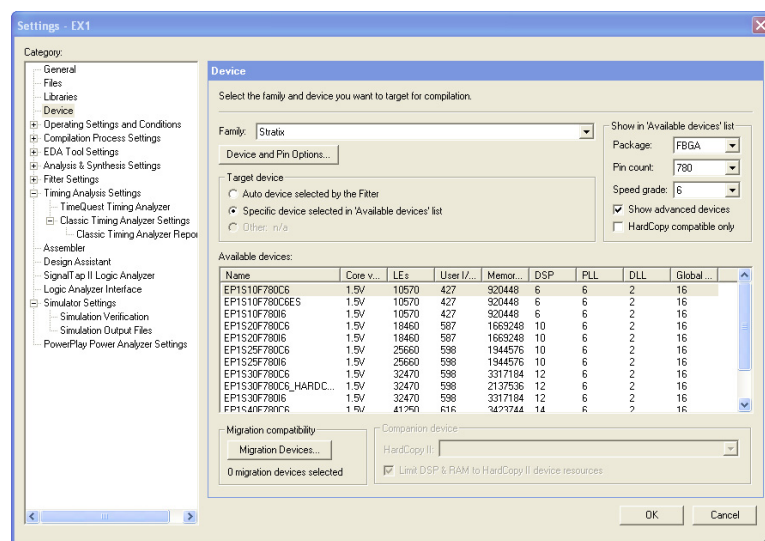
remainder through a one-stage pipeline. The quotient and remainder are fed into the `ddout` module and also to `REG_DDROUT8_IN_H[7..0]` and `REG_DDROUT8_IN_L[7..0]`. The `ddout` module then drives the data out through pins `DDR_IN8_OUT[7..0]` at double the data rate.

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 (Figure 2–27).

Figure 2–27. Device Selection Dialog Box



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**.

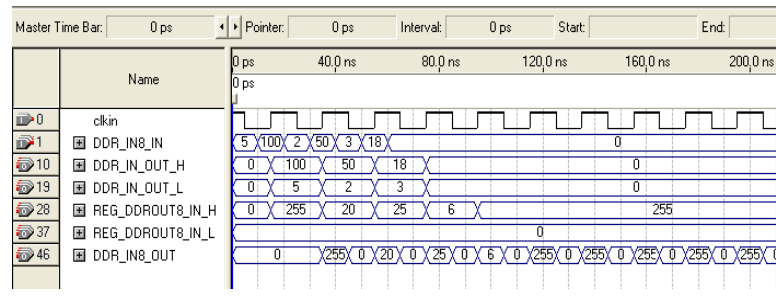
Functional Results—Simulate the Design in Quartus II Simulator

This section describes how to verify the design example you just created by simulating the design using the Quartus II Simulator.

To set up the Quartus II Simulator, perform the following steps:

1. On the Processing menu, click **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful** box displays, click **OK**.
3. On the Assignments menu, click **Settings**. The Settings dialog box displays.
4. In the Category list, select **Simulator Settings**.
5. In the Simulation mode list, select **Functional**.
6. In the Simulation input box, type `EX1_ip.vwf`, or click **Browse (...)** to select the file in the project folder.
7. Turn on the **Run simulation until all vector stimuli are used** option.
8. Click **OK**.
9. On the Processing menu, click **Start Simulation**.
10. When the **Simulator was successful** box displays, click **OK**.
11. The Simulation Report window displays. Verify the simulation waveform results (Figure 2-28).

Figure 2–28. Simulation Waveform



For Figure 2–28, the numerator (100) and denominator (5) are captured at 100 Mbps through pin `DDR_IN8_IN`.

- On the rising edge of `clk` at 15 ns, the numerator (100) drives onto the signal `DDR_IN_OUT_H` and the denominator (5) drives onto the signal `DDR_IN_OUT_L`.
- At 35 ns the quotient (20) and the remainder (0) are calculated and driven onto the signals `REG_DDROUT8_IN_H` and `REG_DDROUT8_IN_L`.
- The high level of `clk`, starting at 55 ns, selects the quotient (20) to drive the `DDR_IN8_OUT` pins, and the low level of `clk` selects the remainder (0) to drive the same pins.

The waveform shown in Figure 2–28 contains calculations for two more sets of numbers. The latency exists because the divider includes a one clock-cycle pipeline.

Functional Results—Simulate the Divider Design in ModelSim-Altera

Simulate the design in ModelSim to compare the results of both simulators. This user guide assumes that you are familiar with using ModelSim-Altera before trying out the design example. If you are unfamiliar with ModelSim-Altera, refer to the support page for software products on the Altera website (www.altera.com). On the support page, there are various links to topics such as installation, usage, and troubleshooting.

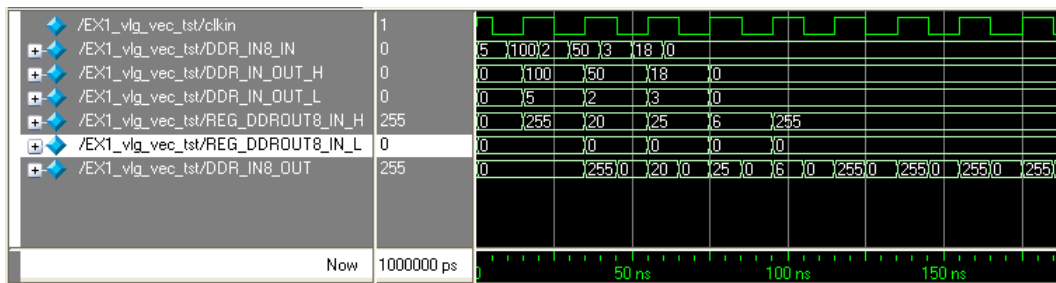
Set up the ModelSim-Altera simulator by performing the following steps.

1. Unzip the **altdio_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 **altdddio_ex1.do** file in a text editor.
3. In line 1 of the **altdddio_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 **altdddio_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 may need to rearrange signals, remove redundant signals, and change the radix to match the appearance of the results in the Quartus II Simulator. [Figure 2–29](#) shows the expected simulation results in ModelSim.

Figure 2–29. 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.

Design Files

The design files are available with this user guide in the Quartus II Project section and in the User Guides section of the Altera website (www.altera.com).

Summary

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.

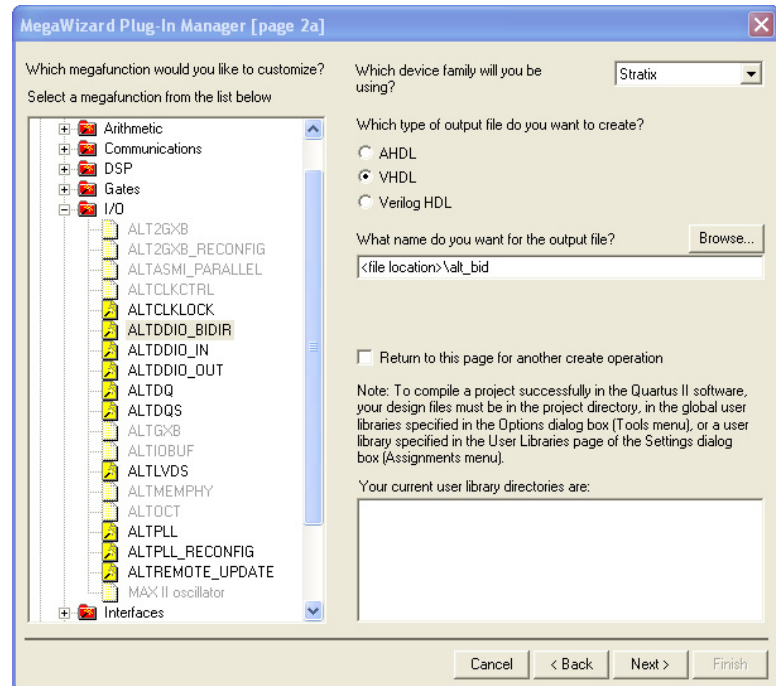
Create the `altddio_bidir` Module

Follow these steps to create the `altddio_bidir` module:

1. Unzip the **altddio_DesignExample_ex2.zip** file to any working directory on your PC.
1. In the Quartus II software, open the **ex2.qar** project .
2. On the Tools menu, select **MegaWizard Plug-In Manager**.
3. 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 ([Figure 2-2](#)).
4. To answer **Which megafunction would you like to customize?**, in the **I/O** folder, select **ALTDDIO_BIDIR**.
5. To answer **Which device family will you be using?**, select **Stratix**.

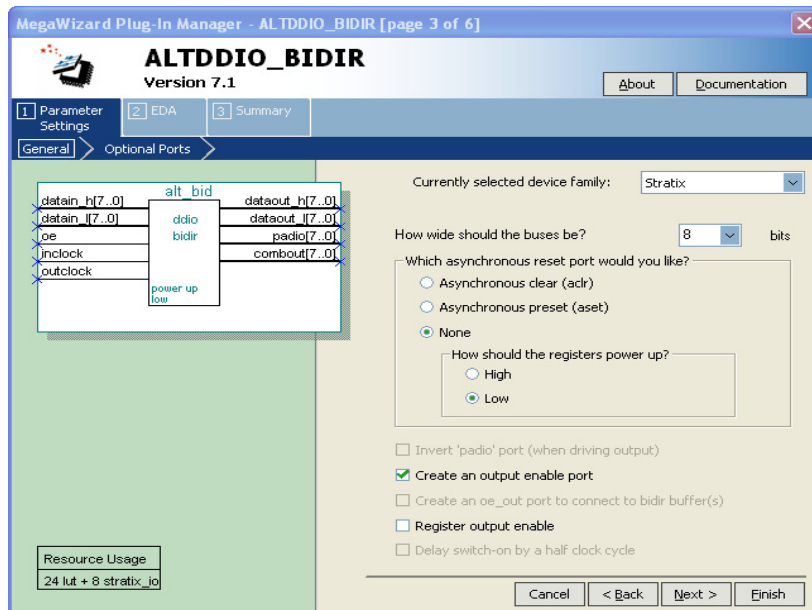
6. To answer **Which type of output file do you want to create?**, select **VHDL**.
7. Specify the output file **alt_bid**. **Figure 2–30** shows the wizard after you have made these selections.

Figure 2–30. MegaWizard Plug-In Manager



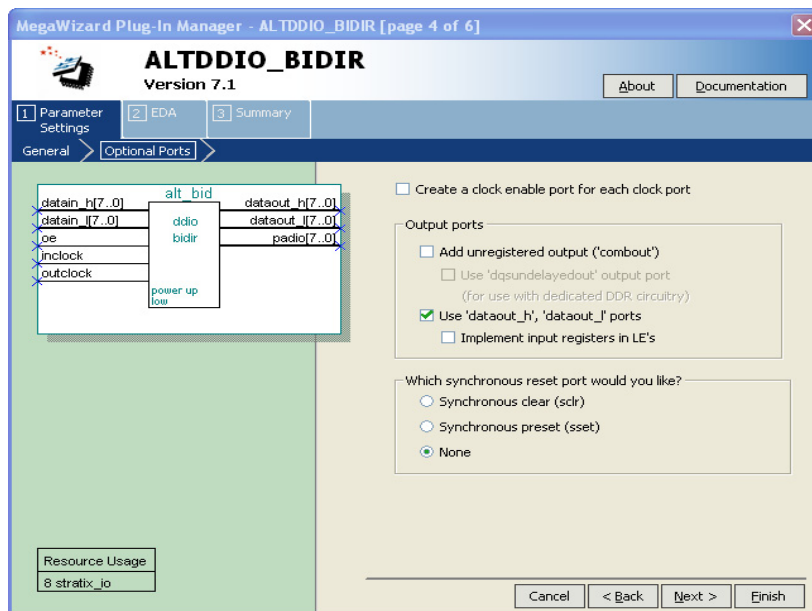
8. Click **Next**. Page 3 appears (**Figure 2–31**).

Figure 2–31. MegaWizard Plug-In Manager—altddio_bidir [page 3 of 6]



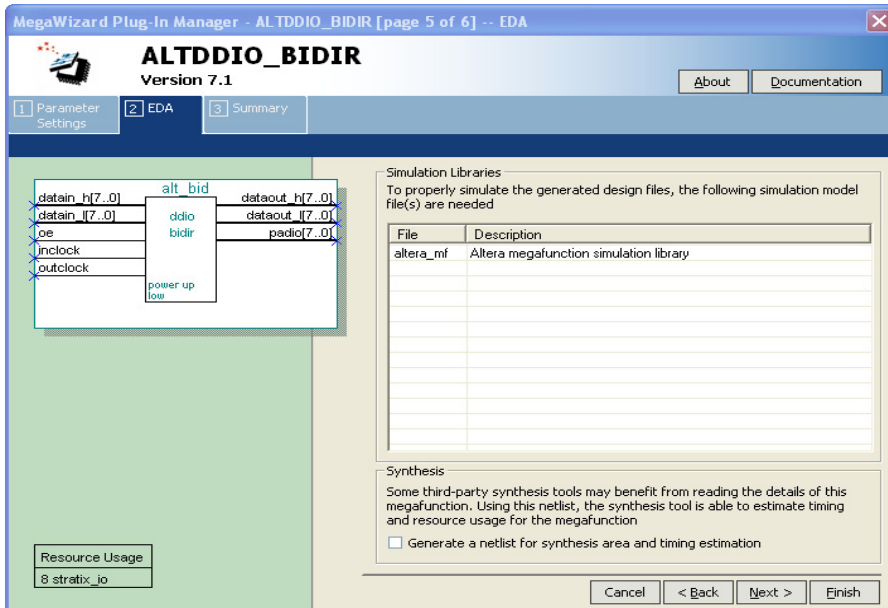
9. To answer **How wide should the buses be?**, select **8**.
10. To answer **Which asynchronous reset port would you like?**, select **None**.
11. To answer **How should the registers power up?**, select **Low**.
12. Turn on **Create an output enable port**.
13. Turn off **Register output enable**.
14. Click **Next**. Page 4 appears (Figure 2–32).

Figure 2–32. MegaWizard Plug-In Manager—altdio_bidir [page 4 of 6]



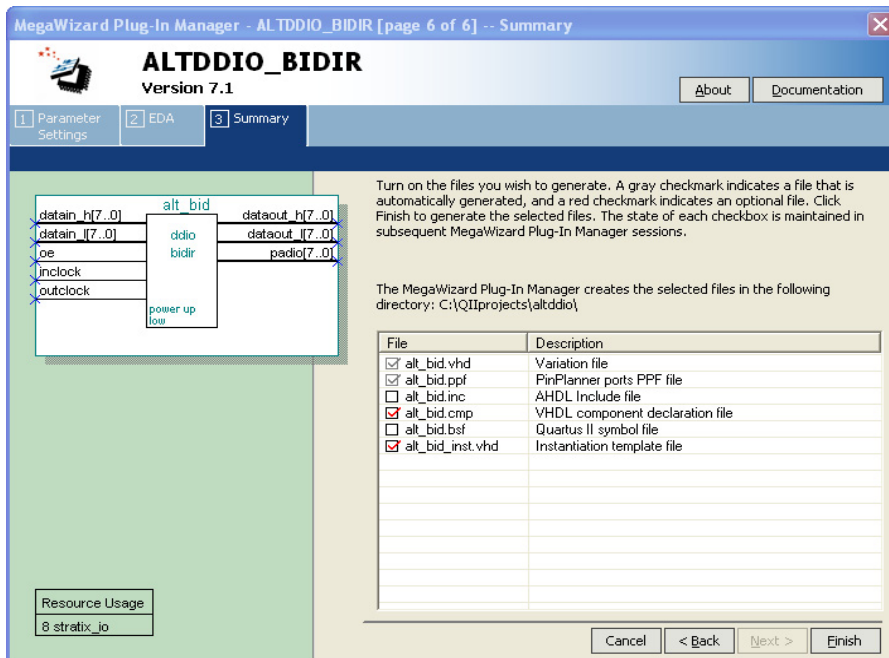
15. Turn off the **Create a clock enable port for each clock port** option.
16. Under **Output ports**, turn off **Add unregistered output ('combout')**, turn on **Use dataout_h, dataout_l ports** and turn off **Implement input registers in LE's**.
17. To answer **Which synchronous reset port would you like?**, select **None**.
18. Click **Next**. Page 5 appears (Figure 2–33).

Figure 2–33. MegaWizard Plug-In Manager—altddio_bidir [page 5 of 6] -- EDA



19. Turn off **Generate a netlist for synthesis area and timing estimation**.
20. Click **Next**. Page 6 appears (Figure 2–34).

Figure 2–34. MegaWizard Plug-In Manager—altdio_bidir [page 6 of 6] -- Summary



21. Turn on the **VHDL component declaration file** and **Instantiation template file** options.
22. Turn off the **Quartus II symbol file** and **AHDL Include file** options.
23. Click **Finish**.

The altdio_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 2–26.

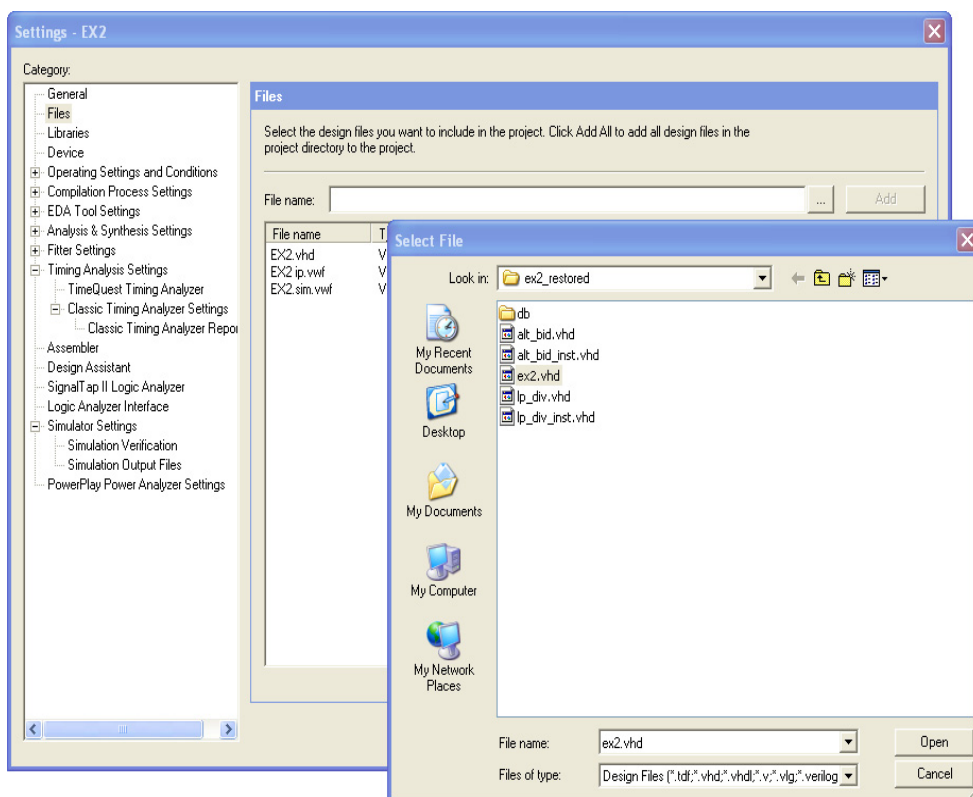
Create a Divider

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 ([Figure 2–35](#)).

Figure 2–35. File Settings



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 `altddividir_in` and `altddividir_out` modules is implemented in a single megafunction, `altddividir`. The bidirectional pins `DDR_BIDIR8[7..0]` receive data at double the clock rate. The `DDR_BIDIR8_OUT_H[7..0]` signals are the numerator and the `DDR_BIDIR8_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 `altddividir` megafunction. The `altddividir` 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**.

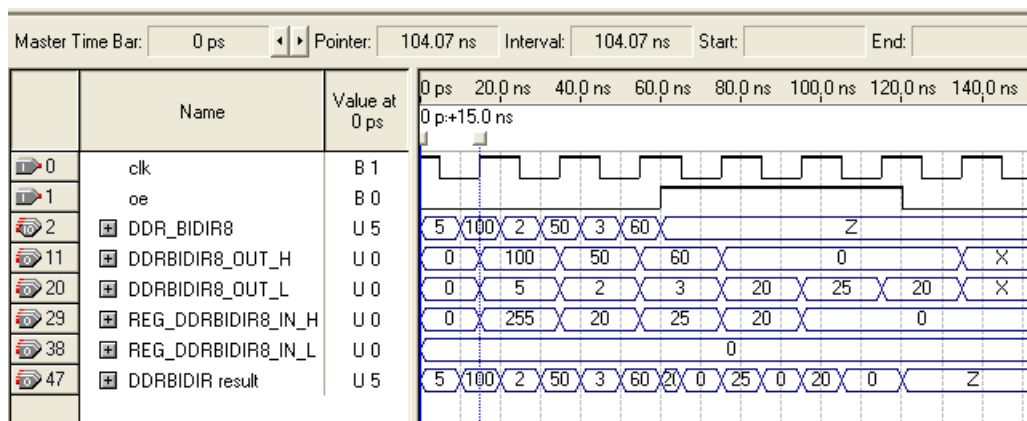
Functional Results—Simulate the Design in Quartus II Simulator

This section describes how to verify the design example you just created by simulating the design using the Quartus II Simulator.

To set up the Quartus II Simulator, perform the following steps:

1. On the Processing menu, click **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful** box displays, click **OK**.
3. On the Assignments menu, click **Settings**. The **Settings** dialog box displays.
4. In the **Category** list, select **Simulator Settings**.
5. In the **Simulation mode** list, select **Functional**.
6. In the **Simulation input** box, type **ex2_ip.vwf**, or click **Browse (...)** to select the file in the project folder.
7. Turn on the **Run simulation until all vector stimuli are used** option.
8. Click **OK**.
9. On the Processing menu, click **Start Simulation**.
10. When the **Simulator was successful** box displays, click **OK**.
11. The **Simulation Report** window displays. Verify the simulation waveform results ([Figure 2-36 on page 2-46](#)).

Figure 2–36. Simulation Waveform



In Figure 2–36, the numerator (100) and denominator (5) are captured at 100 Mbps through pin DDRBIDIR8.

- On the rising edge of `clk` at 15 ns, the numerator (100) drives onto the signal `DDR_BIDIR8_OUT_H` and the denominator (5) drives onto the signal `DDR_BIDIR8_OUT_L`.
- At 35 ns the quotient (20) and the remainder (0) are calculated and driven onto the signals `REG_DDRBIDIR8_IN_H` and `REG_DDRBIDIR8_IN_L`.
- After the clock rises at 55 ns, when the `oe` signal is asserted at 60 ns, the quotient (20) drives onto the `DDR_BIDIR` pins. The following low level of `clk` selects the remainder (0) to drive the same pins.

The waveform shown in Figure 2–36 contains calculations for two more sets of numbers. The latency exists because the divider includes a one-stage pipeline.

To allow the data to be driven out of the bidirectional pin in the simulation, make sure the input signal part of the bidirectional pin is set to a weak unknown, allowing the simulation to overwrite the value at the specific time interval. The Quartus II software creates an additional signal to emulate the output part of the bidirectional pin. This signal is named **<pin name> result**.

Functional Results—Simulate the Divider Design in ModelSim-Altera

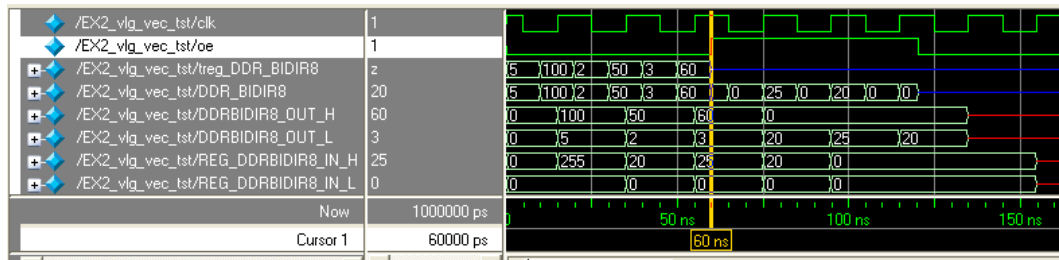
Simulate the design in ModelSim to compare the results of both simulators. This user guide assumes that you are familiar with using ModelSim-Altera before trying out the design example. If you are unfamiliar with ModelSim-Altera, refer to the support page for software products on the Altera website (www.altera.com). On the support page, there are various links to topics such as installation, usage, and troubleshooting.

Set up the ModelSim-Altera simulator by performing the following steps.

1. Unzip the **altd dio_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 **altd dio_ex2.do** file in a text editor.
3. In line 1 of the **altd dio_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 **altd dio_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 may need to rearrange signals, remove redundant signals, and change the radix to match the appearance of the results in the Quartus II Simulator. [Figure 2-37](#) shows the expected simulation results in ModelSim.

Figure 2–37. ModelSim Simulation Results



Conclusion

The Quartus II software provides parameterizable megafuncions ranging from simple arithmetic units, such as adders and counters, to advanced phase-locked loop (PLL) blocks, multipliers, and memory structures. These megafuncions are performance-optimized for Altera devices, and therefore provide more efficient logic synthesis and device implementation, because they automate the coding process and save valuable design time. You should use these functions during design implementation so you can consistently meet your design goals.

Ports and Parameters

The Quartus® II software provides three megafunctions that support DDR functionality: altdio_in, altdio_out, and altdio_bidir. This chapter describes the ports and parameters for the DDR megafunctions.

The parameter details are only relevant for users who bypass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in their design. The details of these parameters are hidden from MegaWizard Plug-In Manager interface users.



Refer to the latest version of the Quartus II Help for the most current information about the ports and parameters for these megafunctions.

Ports and Parameters for the altdio_in Megafunction

Figure 3-1 shows the ports for the altdio_in megafunction.

Figure 3-1. altdio_in Ports

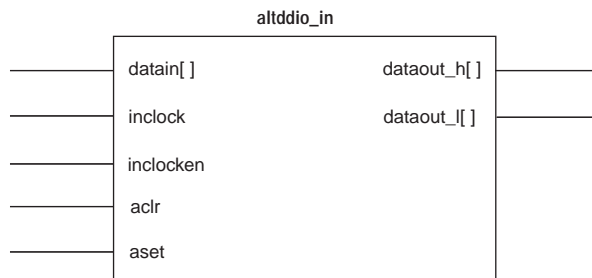


Table 3-1 shows the input ports, Table 3-2 shows the output ports, and Table 3-3 shows the altdio_in megafunction parameters. The options listed in these tables are valid for Arria™ GX, Stratix® III, Stratix II,

Stratix II GX, Stratix, Stratix GX, Cyclone® III, Cyclone II, Cyclone, HardCopy® II, HardCopy Stratix, and APEX™ II devices except where noted.

Table 3–1. altddio_in Input Ports

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

Table 3–2. altddio_in Output Ports

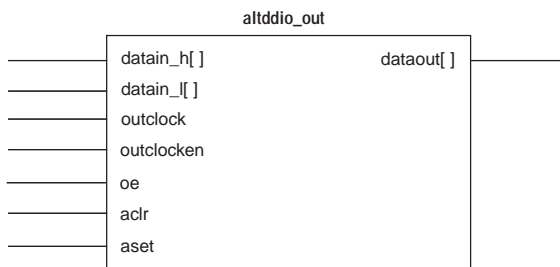
Name	Required	Description
dataout_h[]	Yes	Data sampled from <code>datain[]</code> port at the rising edge of the <code>inclock</code> signal.
dataout_l[]	Yes	Data sampled from <code>datain[]</code> port at the falling edge of the <code>inclock</code> signal.

Table 3–3. altdio_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. Available values are "Arria GX", "Stratix III", "Stratix II", "Stratix II GX", "Stratix", "Stratix GX", "Cyclone III", "Cyclone II", "Cyclone", "HardCopy II", "HardCopy Stratix", and "APEX II".
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.

Ports and Parameters for the altdio_out Megafunction

Ports for the altdio_out megafunction are shown in [Figure 3–2](#).

Figure 3–2. altdio_out Ports

[Table 3–4](#) shows the input ports, [Table 3–5](#) shows the output ports, and [Table 3–6](#) shows the altdio_out megafunction parameters. The options listed in these tables are valid for Arria GX, Stratix III, Stratix II,

Stratix II GX, Stratix, Stratix GX, Cyclone III, Cyclone II, Cyclone, HardCopy II, HardCopy Stratix, and APEX II devices, except where noted.

Table 3–4. *altddio_out* Input Ports

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

Table 3–5. *altddio_out* Output Ports

Name	Required	Description	Comments
<code>dataout[]</code>	Yes	DDR output data port.	Output port <code>WIDTH</code> wide. <code>dataout</code> port should directly feed an output pin in top-level design.
<code>oe_out</code>	No	Output enable for the bidirectional <code>padio</code> port.	Output port [<code>WIDTH</code> –1.0] wide. This port is available for Stratix III and Cyclone III devices only.

Table 3–6. altdio_out Parameters

Parameter	Type	Required	Comments
WIDTH	Integer	Yes	Sets the 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. Available values are “Arria GX”, “Stratix III”, “Stratix II”, “Stratix II GX”, “Stratix”, “Stratix GX”, “Cyclone III”, “Cyclone II”, “Cyclone”, “HardCopy II”, “HardCopy Stratix”, and “APEX II”.
OE_REG	String	No	Specifies whether the <code>oe</code> port is registered. Values are “REGISTERED”, “UNREGISTERED”, and “UNUSED”. The default setting is “UNUSED”.
EXTEND_OE_DISABLE	String	No	This 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”.
LPM_HINT	String	No	Allows you to assign Altera®-specific parameters in VHDL Design Files (<code>.vhd</code>). The default is “UNUSED”.
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.

Ports and Parameters for the altdio_bidir Megafunction

Figure 3–3 shows the ports for the altdio_bidir megafunction.

Figure 3–3. altdio_bidir Ports

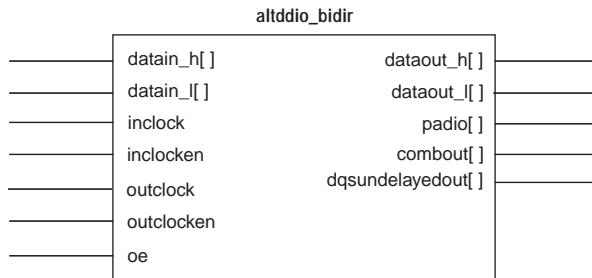


Table 3–7 shows the input ports, Table 3–8 shows the output ports, Table 3–9 shows the bidirectional port, and Table 3–10 shows the altdio_bidir megafunction parameters. The options listed in these tables are valid for Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, Stratix GX, Cyclone III, Cyclone II, Cyclone, HardCopy II, HardCopy Stratix, and APEX II devices.

Table 3–7. altdio_bidir Input Ports (Part 1 of 2)

Name	Required	Description	Comments
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.

Table 3–7. *altddio_bidir* Input Ports (Part 2 of 2)

Name	Required	Description	Comments
oe	No	Output enable for the bidirectional <code>padio</code> port.	If the <code>oe</code> port is not connected, then the <code>padio</code> port is an output port.
sclr	No	Synchronous clear input.	The <code>sclr</code> and <code>sset</code> ports cannot be connected at the same time. The <code>sclr</code> 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 <code>sclr</code> and <code>sset</code> ports cannot be connected at the same time. The <code>sset</code> 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. *altddio_bidir* Output Ports

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

Notes to Table 3–8:

- (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 3–9. *altddio_bidir* Bidirectional Port

Name	Required	Description	Comments
<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 [(WIDTH) – (1) .. 0] wide.

Table 3–10. altdio_bidir Parameters

Parameter	Type	Required	Comments
WIDTH	Integer	Yes	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. Available values are “Arria GX”, “Stratix III”, “Stratix II”, “Stratix II GX”, “Stratix”, “Stratix GX”, “Cyclone III”, “Cyclone II”, “Cyclone”, “HardCopy II”, “HardCopy Stratix”, and “APEX II”.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
OE_REG	String	No	Specifies whether the oe 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 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”.
INVERT_OUTPUT	String	No	Specifies whether to invert the dataout[] output port. Values are “ON” or “OFF”. This parameter is available for Cyclone III and Cyclone II devices only.