

Ipm_shiftreg Megafunction



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

Quartus II Software Version: 6.0
Document Version: 2.0
Document Date: August 2006

Copyright © 2006 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-033105-2.0



Contents

About this User Guide

Revision History	v
How to Contact Altera	v
Typographic Conventions	vi

Chapter 1. About this Megafunction

Device Family Support	1-1
Introduction	1-2
Features	1-2
General Description	1-2
Common Applications	1-3
Resource Utilization & Performance	1-4

Chapter 2. Getting Started

System Requirements	2-1
MegaWizard Plug-In Manager Customization	2-1
MegaWizard Page Descriptions	2-1
Inferring Megafunctions from HDL Code	2-7
Instantiating Megafunctions in HDL Code	2-8
Identifying a Megafunction after Compilation	2-8
Simulation	2-8
Quartus II Simulation	2-8
EDA Simulation	2-9
SignalTap II Embedded Logic Analyzer	2-9
Design Example: Configurable 8-Bit SIPO or PISO Shift Register	2-10
Design Files	2-10
Example 1	2-10
Generate a Configurable 8-Bit SIPO or PISO Shift Register	2-10
Implement the Configurable 8-Bit SIPO or PISO Shift Register	2-15
Functional Results—Simulate the 8-Bit Shift Register in Quartus II	2-17
Functional Results—Simulate the 8-Bit Shift Register in ModelSim-Altera	2-19
Design Example: Time Delay	2-21
Design Files	2-21
Example 2	2-21
Generate the Time Delay Design	2-21
Implement the Time Delay Design	2-27
Functional Results—Simulate the Time Delay Design in Quartus II	2-28
Functional Results—Simulate the Time Delay Design in ModelSim-Altera	2-30

Conclusion 2-31

Chapter 3. Specifications

Ports & Parameters 3-1



About this User Guide

Revision History The table below displays the revision history for the chapters in this User Guide.

Chapter	Date	Version	Changes Made
All	August 2006	2.0	Updated for Quartus II 6.0 software release.
All	July 2005	4.2	Updated for Quartus II 4.2 software release.
All	March 2005	1.0	Initial release.

How to Contact Altera

For the most up-to-date information about Altera® products, go to the Altera worldwide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	(408) 544-7000 (1) (7:00 a.m. to 5:00 p.m. Pacific Time)
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com (1)	literature@altera.com (1)
Non-technical customer service	(800) 767-3753 (7:00 a.m. to 5:00 p.m. Pacific Time)	(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)
FTP site	ftp.altera.com	ftp.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 below.

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} , qdesigns 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 + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pdf file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets are used in a list of items when the sequence of the items is not important.
	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Device Family Support

Megafunctions provide either full or preliminary support for target Altera® device families, as described below:

- *Full support* means the megafunction meets all functional and timing requirements for the device family and may be used in production designs.
- *Preliminary support* means the megafunction meets all functional requirements, but may still be undergoing timing analysis for the device family. It may be used in production designs with caution.

Table 1–1 shows level of support for each Altera device family.

Table 1–1. Device Family Support	
Device Family	Support
Stratix® II	Full
Stratix II GX	Full
Stratix GX	Full
Stratix	Full
Cyclone™ II	Full
Cyclone	Full
HardCopy® II	Full
HardCopy Stratix	Full
MAX® II	Full
MAX 7000	Full
MAX 3000	Full
APEX™ II	Full
APEX 20KC	Full
APEX 20KE	Full
FLEX® 10K	Full
FLEX 10KA	Full
FLEX 10KE	Full
FLEX 6000	Full
ACEX® 1K	Full

Introduction

As design complexities increase, use of vendor-specific IP blocks has become common design methodology. Altera provides parameterizable megafunctions optimized for Altera device architectures. Using megafunctions instead of coding your own logic saves valuable design time, offering more efficient logic synthesis and device implementation. Scale the megafunction's size by simply setting parameters.

Features

The `lpm_shiftreg` megafunction implements a shift register and offers many additional features, including:

- Fully parameterizable
- Synchronous or asynchronous inputs to shift register
- Synchronous parallel load
- Left/right register shifting
- Optional inputs, including clock enable input, serial shift data input, and parallel input
- Optional outputs, including data output and serial shift data output

General Description

The `lpm_shiftreg` megafunction is a storage megafunction provided in the Quartus II® MegaWizard® Plug-In Manager. Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All the flip-flops are driven by a common clock and are set or reset simultaneously. A shift register is useful for converting parallel signals to serial signals and vice versa.

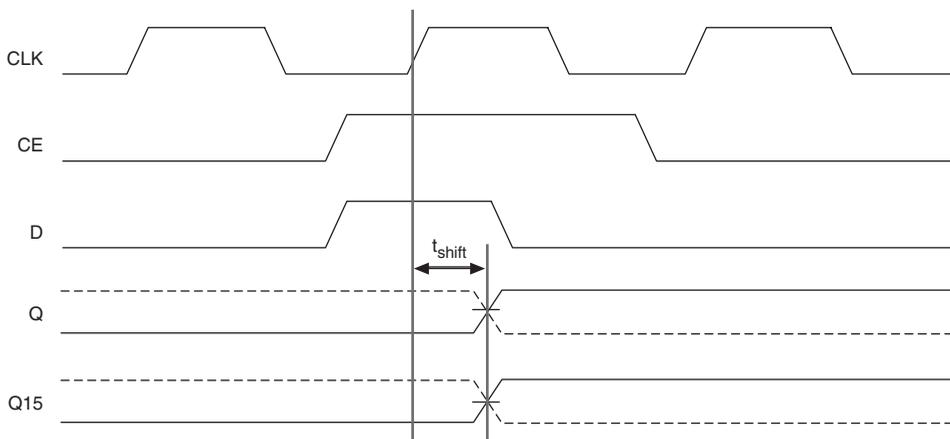
Shift register megafunction provided by Altera is a very versatile parameterizable block of logic. Thus long delay chains can be instantiated using this megafunction. This megafunction provides for either left shift or right shift of the input data bits. Data that has to be shifted is either loaded in parallel into the registers synchronously or in serial through the 'shiftin' input of the megafunction. The loaded data is then shifted with the rising edge of clock input.

The shift operation is a single clock-edge operation with an active-high clock enable feature. When enable is High, the input (D) is loaded into the first bit of the shift register, and each bit is shifted to the next highest bit position. [Figure 1-1](#) illustrates the shift operations. Cascading of shift registers is another way of using the `lpm_shiftreg` megafunction to achieve higher shift count or bit count.

Optional inputs are available to asynchronously clear or set the registers, or synchronously clear or set the registers. Using this feature, you can either set the initial value of all the registers to 1, or to a desired value.

Parallel output $q[]$ is used to read parallel data from the shift register. Parallel data is always available on the $q[]$ outputs at every clock. When data is shifted serially with every clock, you get the MSB of the $q[]$ output on the 'shiftout' pin. A shift operation is shown in [Figure 1-1](#).

Figure 1-1. Shift Operation



Common Applications

Use the `lpm_shiftreg` megafunction to replace all other types of shift register functions.

Shift registers enable the development of efficient designs for applications that require delay or latency compensation. Shift registers are also useful in synchronous FIFO and content addressable memory (CAM) designs. Shift registers are often used as the state register in a sequential device. Usually, the next state is determined by shifting right and inserting a primary input or output into the next position (for example, a finite memory machine). They are very effective for sequence detectors. Shift registers are used for Serial interconnection of systems that keeps interconnection cost low with serial interconnect. Shift registers are used for Bit Serial Operations. Bit serial operations can be performed quickly through device iteration.

The `lpm_shiftreg` megafunction finds applications where there is a need to shift the data in or out of digital systems. Serial to Parallel Conversion, Parallel to Serial Conversion, and delay generation for multistage pipeline stages are some of the common applications of a shift register.

Resource Utilization & Performance

This megafunction uses one logic cell per bit. Table 1–2 shows the resource usage for each Altera device family.

Table 1–2. *lpm_shiftreg* Resource Usage

Device Family	Optimization (1)	Width	Logic Elements
Stratix II	Balanced	8-Bit	8 ALUT
Stratix GX	Balanced	8-Bit	8 logic elements
Stratix	Balanced	8-Bit	8 logic elements
Cyclone II	Balanced	8-Bit	8 logic elements
Cyclone	Balanced	8-Bit	8 logic elements
MAX II	Balanced	8-Bit	8 logic elements
MAX 7000	Balanced	8-Bit	8 registers
MAX 3000	Balanced	8-Bit	8 registers
APEX II	Balanced	8-Bit	8 registers
APEX 20KC	Balanced	8-Bit	8 registers
APEX 20KE	Balanced	8-Bit	8 registers
FLEX 10K	Balanced	8-Bit	8 registers
FLEX 10KA	Balanced	8-Bit	8 registers
FLEX 10KE	Balanced	8-Bit	8 registers
FLEX 6000	Balanced	8-Bit	8 registers
ACEX 1K	Balanced	8-Bit	8 registers

Note for Table 1–2:

- (1) Choose a design implementation that balances high performance with minimal logic usage. This setting is available for Cyclone series, MAX II, Stratix, and Stratix II devices only. The balanced optimization logic option can be set on the Assignments menu in **Analysis and Synthesis settings**.

The MegaWizard® Plug-In Manager reports approximate resource utilization based on user specification and parameters, available in the lower left corner of the MegaWizard Plug-In Manager screen.

System Requirements

The instructions in this section require the following hardware and software:

- A PC running the Windows 2000/XP, Red Hat Linux Enterprise 3 or 4, or a Sun workstation running the Solaris 8 or 9 operating system
- The Quartus® II software version 6.0 or higher

MegaWizard Plug-In Manager Customization

Use the MegaWizard® Plug-In Manager to specify the `lpm_shiftreg` megafunction features for each shift register function in your design.

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

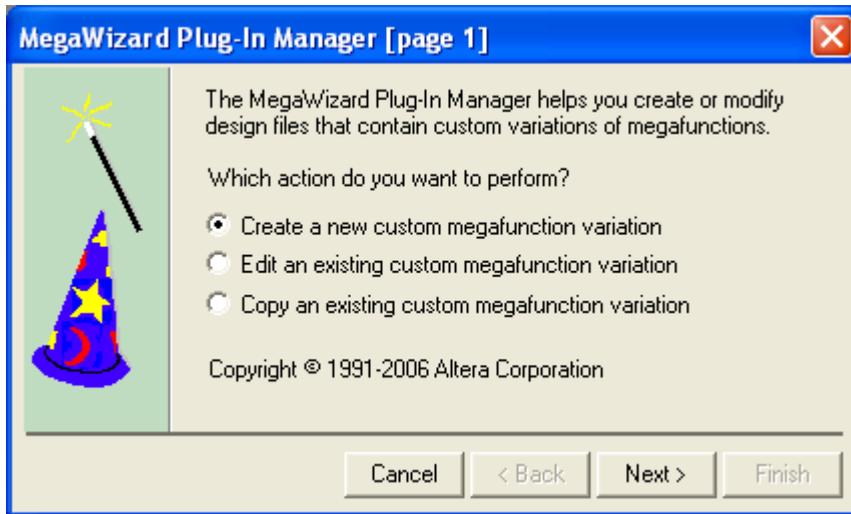
- On the Tools menu, click the **MegaWizard Plug-In Manager** command.
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** dialog box.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:
`qmegawiz`↵

MegaWizard Page Descriptions

This section provides descriptions of the options available on the individual pages of the `lpm_shiftreg` wizard.

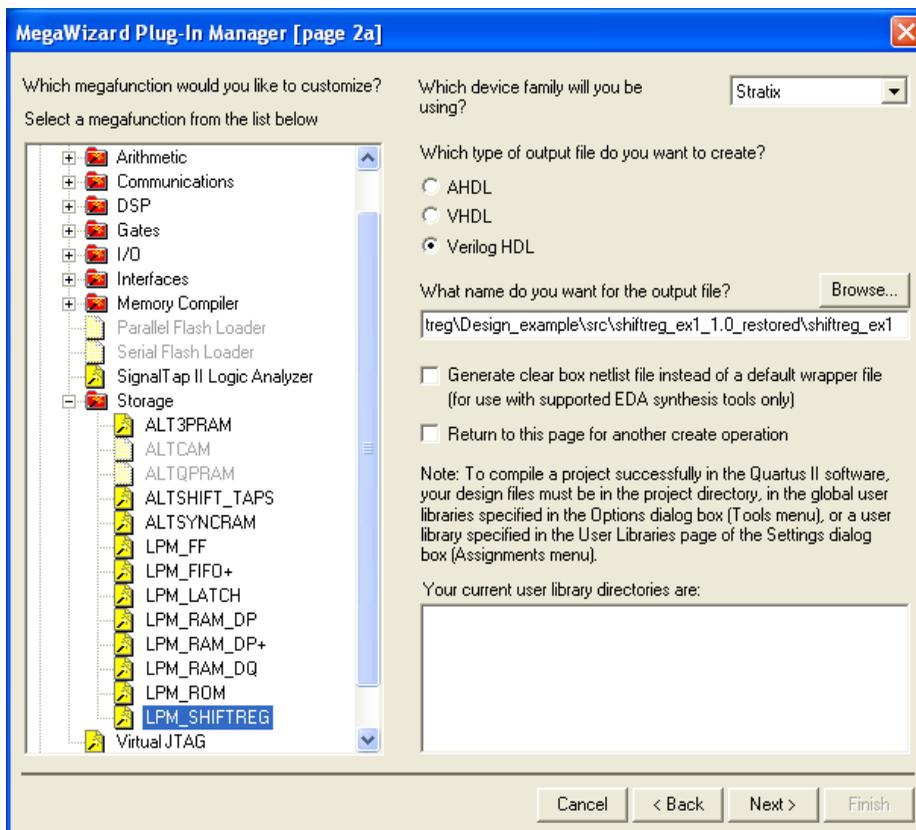
On page 1 of the MegaWizard Plug-In Manager, you can choose to **Create a new custom megafunction variation**, **Edit an existing megafunction variation**, or **Copy an existing custom megafunction variation** (Figure 2-1).

Figure 2-1. MegaWizard Plug-In Manager [page 1]



On page 2a of the wizard, specify plug-in, select device family, output file type, and name of output file (Figure 2-2). Choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v). You can also create a clearbox instantiation for third-party EDA tools.

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



On page 3 of the wizard, select the width of the output bus, specify the shift direction, shift register output, and optional inputs. (Figure 2-3).

Figure 2-3. MegaWizard Plug-In Manager [page 3]

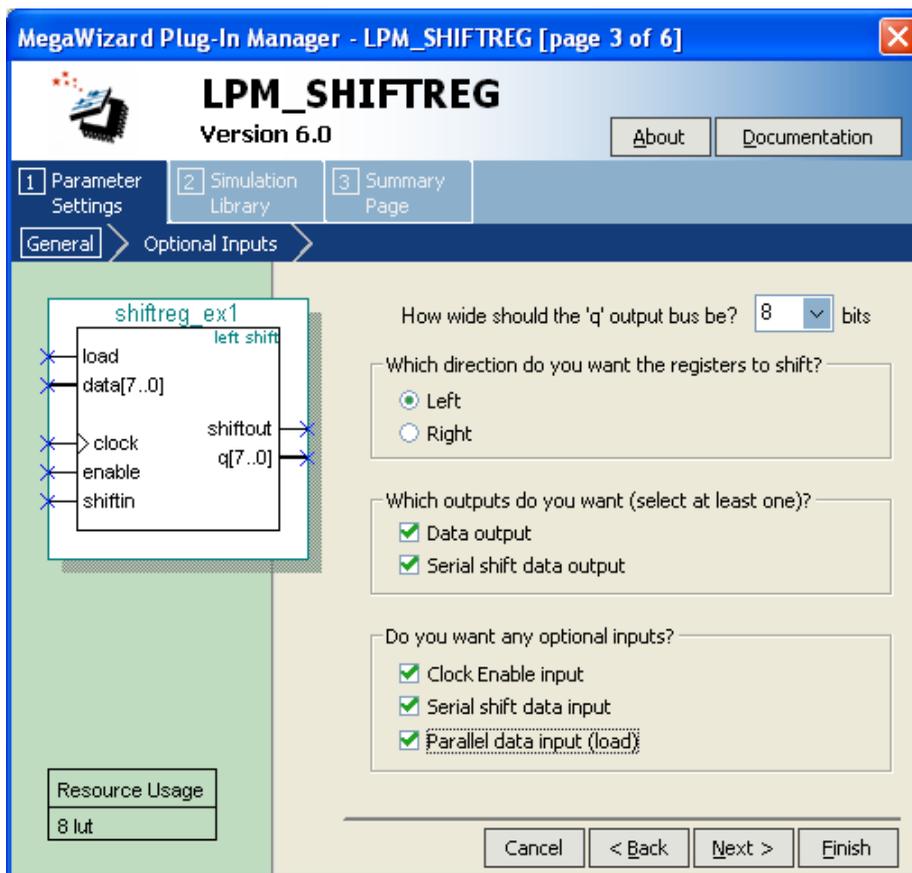


Table 2-1 describes the options on page 3 of the `lpm_shiftreg` wizard.

Table 2-1. *lpm_shiftreg* MegaWizard Plug-in Manager Page 3 Options (Part 1 of 2)

Function	Description
How wide should the 'q' output bus be?	Select the width for the 'q' output bus. The maximum size of the 'q' output bus can be 256 bits. Manually enter widths greater than 256.
Which direction do you want the registers to shift?	Select 'left' or 'right' to define the direction of data shift.

Table 2–1. lpm_shiftreg MegaWizard Plug-in Manager Page 3 Options (Part 2 of 2)

Function	Description
Which outputs do you want (select at least one)?	Select 'Data output' (parallel output), 'Serial shift data output', or select both.
Do you want any optional inputs?	Select optional inputs to shift register. Select 'Clock Enable input' to provide enable function to clock input. Load shift register with parallel data. Use serial shift data input feature to put serial data into shift register.

On page 4, specify synchronous and asynchronous inputs (Figure 2–4).

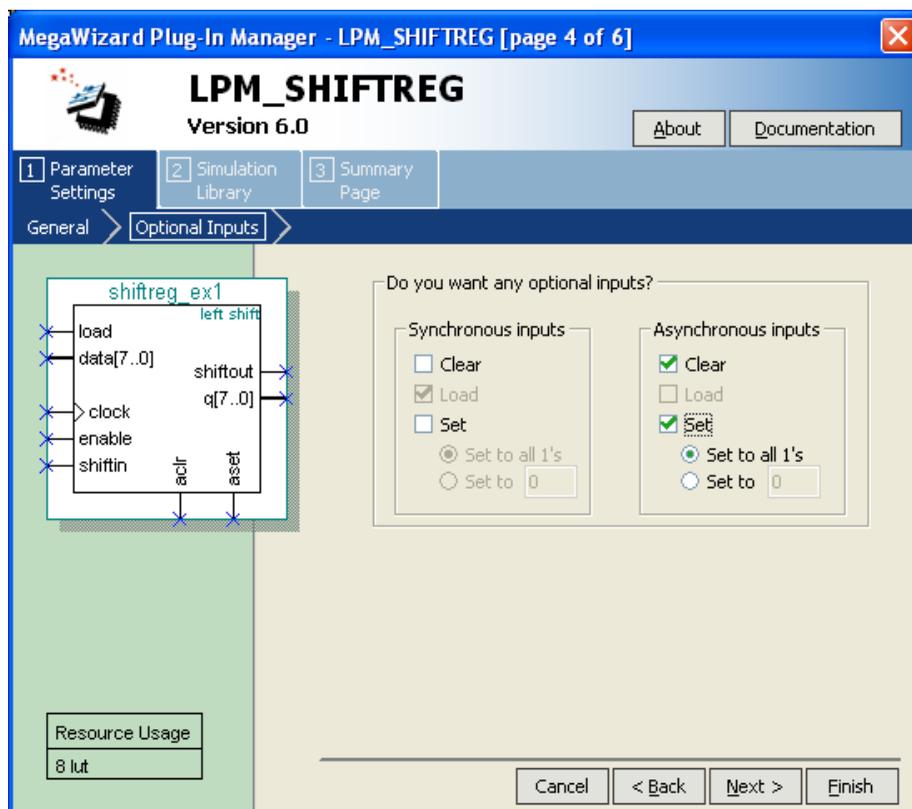
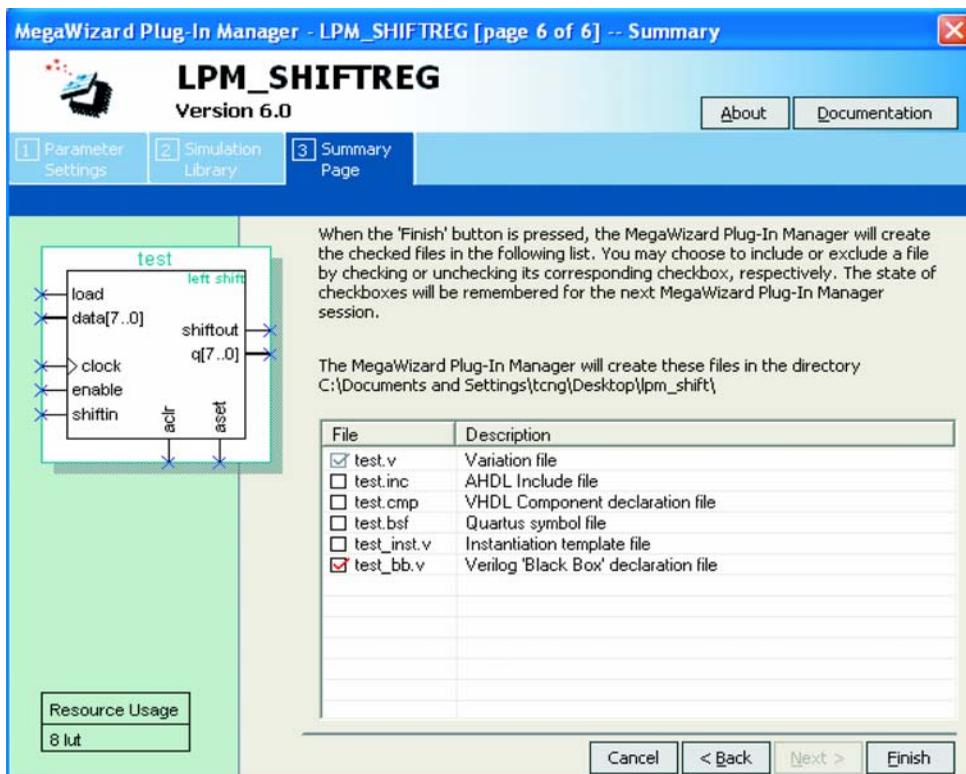
Figure 2–4. lpm_shiftreg Wizard [page 4]

Table 2–2. *lpm_shiftreg* MegaWizard Plug-in Manager Page 4 Options

Function	Description
Do you want any optional inputs?	Select synchronous and/or asynchronous inputs. Use synchronous or asynchronous set and clear inputs of the shift register as optional features.
Synchronous inputs	Shift register has optional synchronous clear and set inputs. Use 'clear' input to clear all registers synchronously. Use 'set' input to either set all <code>q[]</code> outputs to 1's or to a particular value specified in 'Set to' field. The <code>sclr</code> signal affects <code>q[]</code> outputs before polarity is applied to ports. If both <code>sset</code> and <code>sclr</code> are used and both are asserted, <code>sclr</code> is dominant.
Asynchronous inputs.	Shift register has optional asynchronous clear and set options. Use 'clear' to asynchronously clear all <code>q[]</code> outputs. <code>acclr</code> signal affects <code>q[]</code> outputs before polarity is applied to ports. Use 'set' input to either set all <code>q[]</code> outputs to 1's or to a particular value specified in 'Set to' field. The <code>acclr</code> signal affects <code>q[]</code> outputs before polarity is applied to ports. If both <code>aset</code> and <code>acclr</code> are used and both are asserted, <code>acclr</code> is dominant.

On page 6 of the wizard, specify the types of files to be generated. Choose from HDL wrapper file, (`<function name>.v` | `.vhd` | `.tdf`), Block Symbol file (`.bsf`), Instantiation template file (`<function name>_inst.v`), or Verilog Black Box declaration file (`<function name>_bb.v`) (Figure 2–5).

Figure 2-5. *lpm_shiftreg Wizard [page 6] -- Summary*

For more information about the ports for the `lpm_shiftreg` megafunction, refer to the *Specifications* chapter in this User Guide.

Inferring Megafunctions from HDL Code

Synthesis tools, including the Quartus II integrated synthesis, recognize specific types of HDL code, automatically inferring the appropriate megafunction when a megafunction will provide optimal results. The Quartus II software uses the Altera® megafunction code when compiling your design, even if it was not specifically instantiated. The Quartus II software infers megafunctions which are optimized for Altera devices, so the area and/or performance may be better than generic HDL code. Use megafunctions to access Altera architecture-specific features, such as memory, DSP blocks, and shift registers, providing improved performance compared with basic logic elements.

Refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook* for more information.

Instantiating Megafunctions in HDL Code

When you use the MegaWizard Plug-In Manager to set up and parameterize a megafunction, it creates either a VHDL or Verilog HDL wrapper file that instantiates the megafunction (a black-box methodology). For some megafunctions, you can generate a fully synthesizable netlist for improved results with EDA synthesis tools, such as Synplify and Precision RTL Synthesis (a clear-box methodology). Both clear-box and black-box methodologies are described in the third-party synthesis support chapters in the *Quartus II Handbook*.

- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*
- *Synplicity Synplify & Synplify Pro Support* chapter in volume 1 of the *Quartus II Handbook*
- *Mentor Graphics Precision RTL Synthesis Support* chapter in volume 1 of the *Quartus II Handbook*

Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration is performed to build the structure of your design. To locate your megafunction in the Project Navigator window, expand the compilation hierarchy and find the megafunction by its name.

To search for node names within the megafunction (using the Node Finder), click **Browse (...)** in the **Look in** box and select the megafunction in the **Hierarchy** box.

Simulation

The Quartus II Simulation tool provides an easy-to-use, integrated solution for performing simulations. The following sections describe the simulation options.

Quartus II Simulation

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. The functional simulation enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, you can view signals that exist before synthesis. You can find these signals with the Registers: pre-synthesis, Design Entry, or Pin filters in the Node Finder. The top-level ports of megafunctions are found using these three filters.

In contrast, timing simulation in the Quartus II software verifies the operation of your design with annotated timing information. This simulation is performed using the post place-and-route netlist. When performing a timing simulation, you are able to view signals that exist after place-and-route. These signals are found with the Post-Compilation filter of the Node Finder. During synthesis and place-and-route, the names of RTL signals change. Therefore, it might be difficult to find signals from your megafunction instantiation in the Post-Compilation filter. To preserve the names of your signals during the synthesis and place-and-route stages, use the synthesis attributes `keep` or `preserve`. These are Verilog and VHDL synthesis attributes that direct analysis & synthesis to keep a particular wire, register, or node intact. Use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation. Refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

EDA Simulation

Depending on the third-party simulation tool you are using, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*. These tool-specific chapters show you how to perform functional and gate-level timing simulations including megafunctions, and the necessary files and file directories.

SignalTap II Embedded Logic Analyzer

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

To monitor signals from your megafunctions, first configure the SignalTap II embedded logic analyzer in the Quartus II software, and include the analyzer as part of your project. The Quartus II software seamlessly embeds the analyzer 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: Configurable 8-Bit SIPO or PISO Shift Register

This design example uses the `lpm_shiftreg` megafunction to implement a configurable 8-bit serial in parallel out (SIPO) or parallel in serial out (PISO) shift register.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site:

<http://www.altera.com/support/examples/quartus/quartus.html>

Select the “Examples for `lpm_shiftreg` Megafunction User Guide” link from the examples page to download the design files.

Example 1

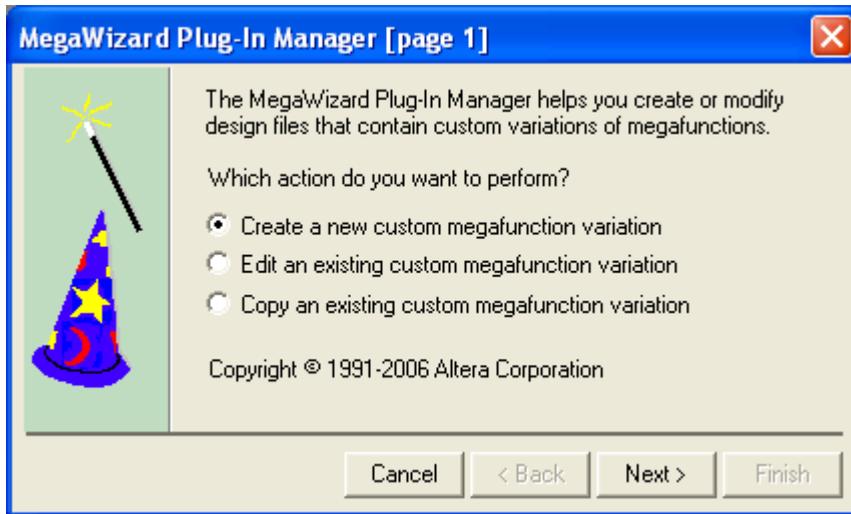
In this example, you perform the following tasks:

- Create an 8-bit shift register using the `lpm_shiftreg` megafunction and the MegaWizard Plug-in Manager
- Implement design and assign the EP1S10F780C6 device to the project
- Compile and simulate the design

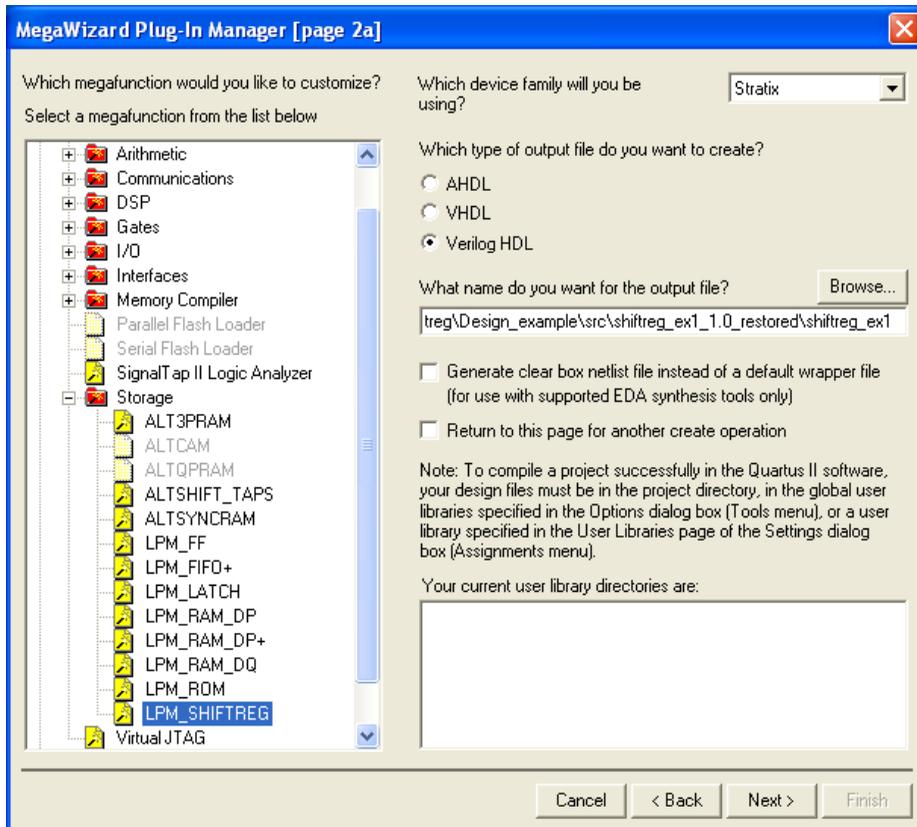
Generate a Configurable 8-Bit SIPO or PISO Shift Register

1. In the Quartus II software, open the `lpm_shiftreg_DesignExample_ex1.qar` project.
2. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears (Figure 2–6).

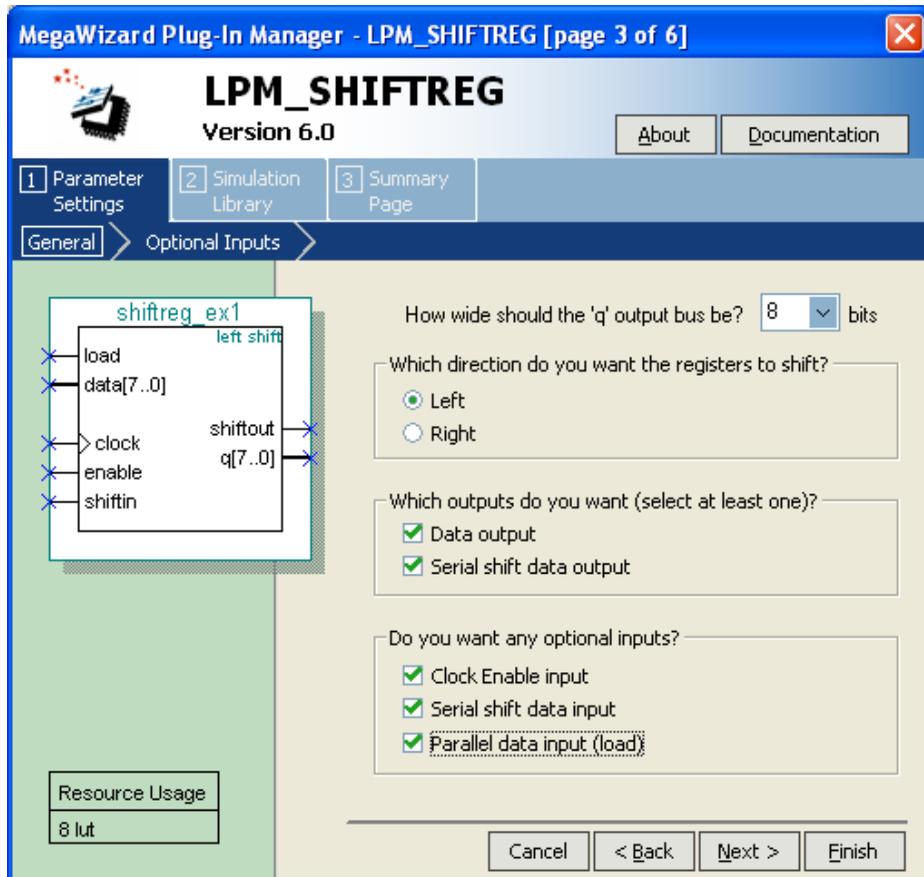
Figure 2–6. MegaWizard Plug-In Manager [page 1]



3. Select **Create a new custom megafunction variation**, and click **Next**. Page 2a of the **MegaWizard Plug-In Manager** appears (Figure 2–7).

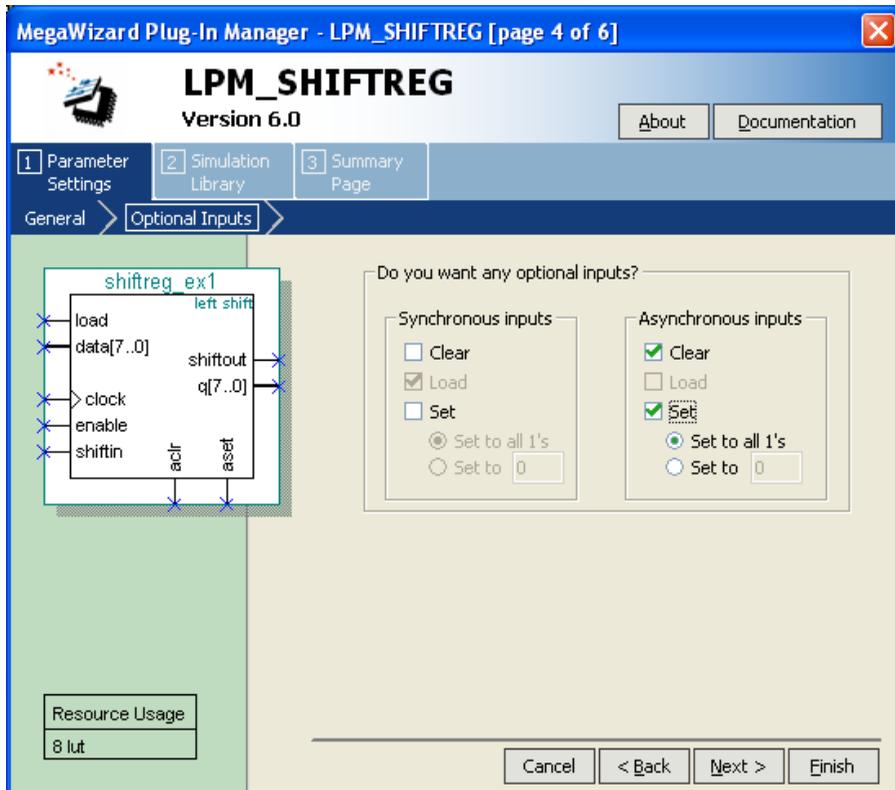
Figure 2–7. *lpm_shiftreg Wizard [page 2a]*

4. From the **Which device family will you be using?** list, select **Stratix**.
5. From the **Which type of output file do you want to create?** option, click **Verilog HDL**.
6. In the **Storage** folder, select **LPM_SHIFTREG**. Specify the output file **shiftreg_ex1**.
7. Click **Next**. Page 3 appears (Figure 2–8).

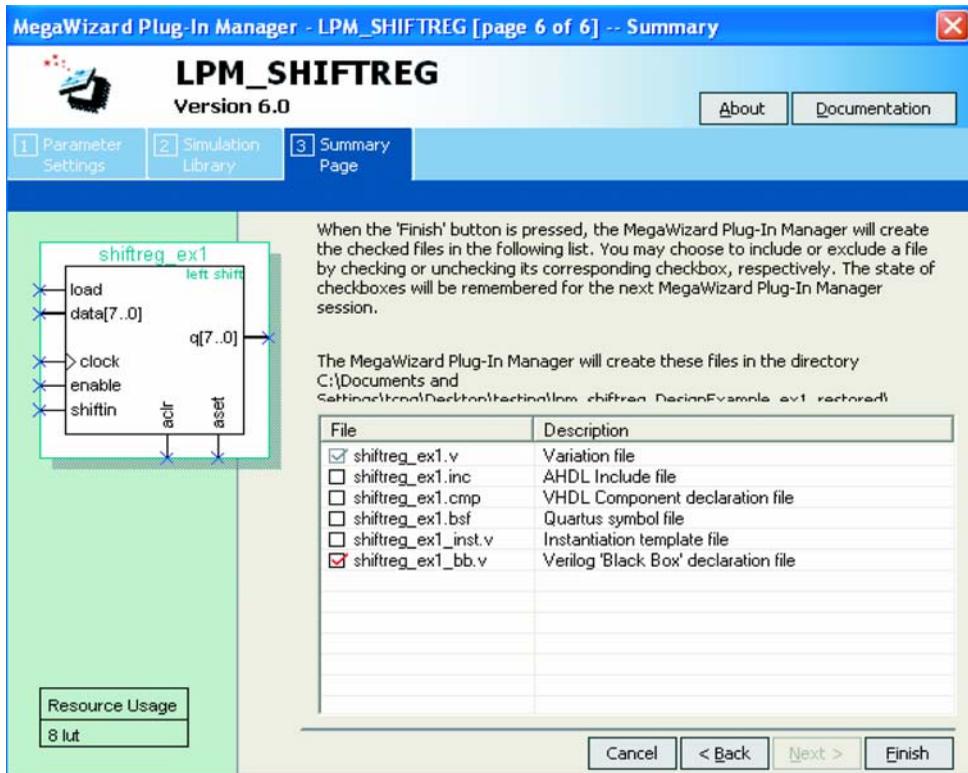
Figure 2–8. *lpm_shiftreg Wizard [page 3 of 6]*

8. In the **How wide should the 'q' output bus be?** list, select 8.
9. Under **What direction do you want the registers to shift?**, select **Left**.
10. Under **What outputs do you want (select at least one)?**, turn on both the **Data output** and **Serial shift data output** options.
11. Under **Do you want any optional inputs?**, turn on all three options.
12. Click **Next**. Page 4 appears (Figure 2–9).

Figure 2–9. *lpm_shiftreg Wizard [page 4 of 6]*



13. Under **Synchronous inputs**, turn off **Clear** and **Set**.
14. Under **Asynchronous inputs**, turn on **Clear** and **Set**.
15. Click **Finish**. Page 6 appears (Figure 2–10).

Figure 2–10. *lpm_shiftreg* Wizard [page 6 of 6] -- Summary

16. Turn on Verilog 'Black Box' declaration file.
17. Turn off AHDL Include file, VHDL Component declaration file, Quartus symbol file, and Instantiation template file, and click Finish.

The `lpm_shiftreg` module is now built.

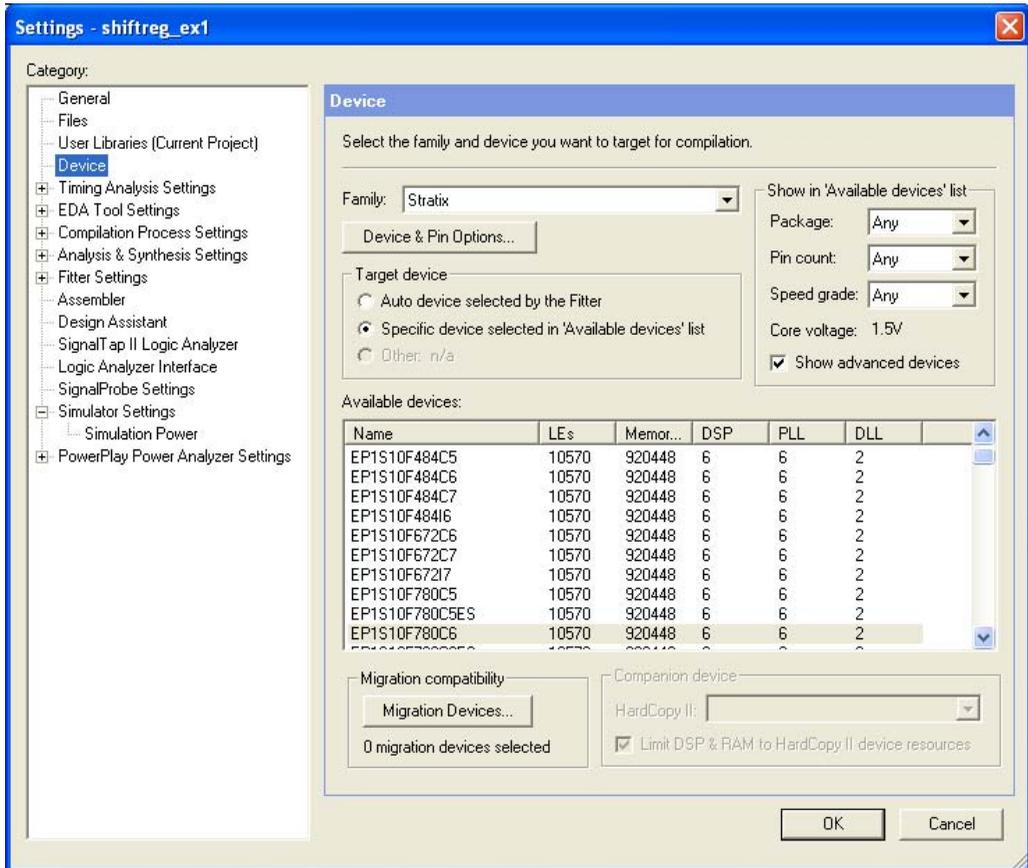
Implement the Configurable 8-Bit SIPO or PISO Shift Register

In this example, you assign the EP1S10F780C6 device to the project and compile the project.

1. In the Quartus II software, on the Assignments menu, click **Settings**. The Settings dialog box appears.

- Under **Category**, select **Device** (Figure 2–11).

Figure 2–11. Device Settings Dialog Box



- In the **Family** list, select **Stratix**.
- Under **Target device**, click **Specific device selected in 'Available devices' list**.
- In the **Available devices** list, select **EP1S10F780C6**.
- Leave the other options in the default state and click **OK**.
- On the Processing menu, click **Start Compilation**.

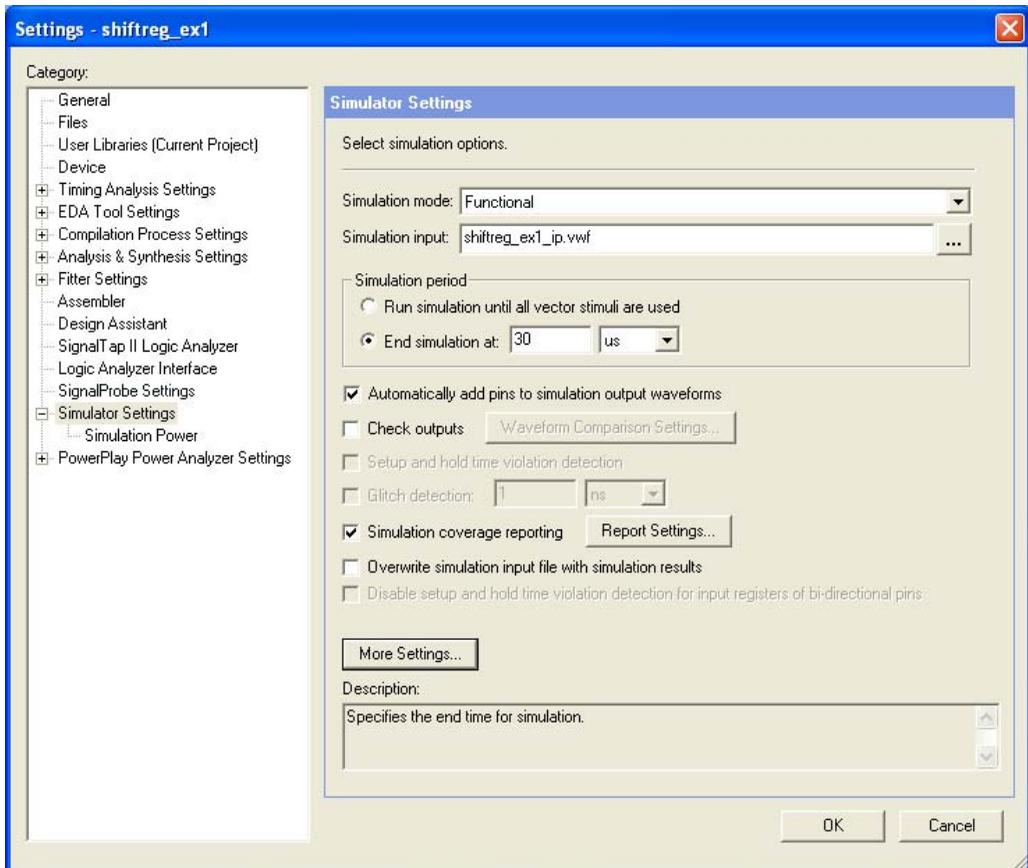
8. When the **Full Compilation was successful** box appears, click **OK**.

Functional Results—Simulate the 8-Bit Shift Register in Quartus II

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. In the Quartus II software, on the Processing menu, select **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful message** box appears, click **OK**.
3. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
4. In the **Category** list, select **Simulator Settings** (Figure 2-12).

Figure 2–12. Simulator Settings Dialog Box



5. In the **Simulation mode** list, select **Functional**.
6. Type **shiftreg_ex1_ip.vwf** in the **Simulation input** box, or click **Browse (...)** to select the file in the project folder.
7. Select **End simulation at**; type **30**, and select **us** from the list.
8. Turn on the **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting** options.
9. Turn off **Check points** and **Overwrite simulation input file with simulation results** and click **OK**.

10. On the Processing menu, click **Start Simulation**.
11. When the **Simulation was successful** message box appears, click **OK**.
12. The **Simulation Report** window displays. Verify the simulation waveform results (Figure 2–13 and Figure 2–14).

Figure 2–13. Simulation Waveform, SIPO

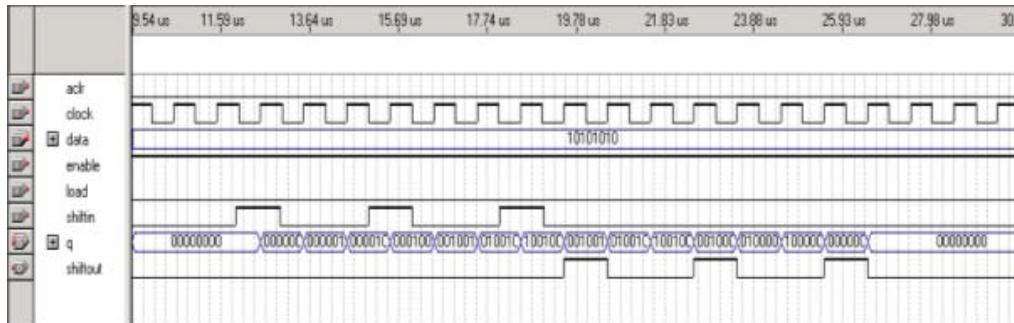
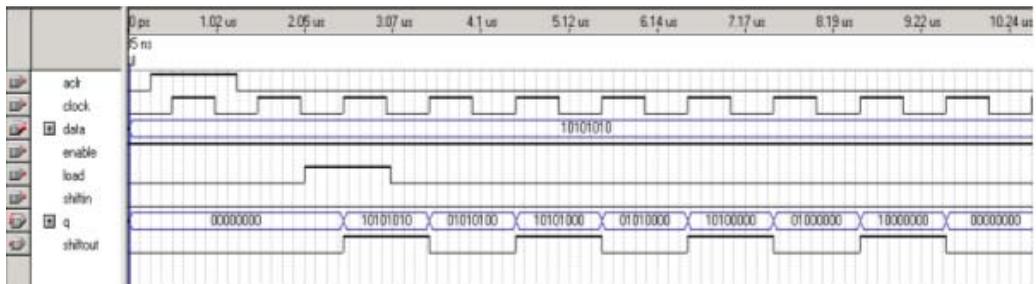


Figure 2–14. Simulation Waveform, PISO



Functional Results—Simulate the 8-Bit Shift Register 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 at:

Example 1

<http://www.altera.com/support/software/products/modelsim/mod-modelsim.html>. The support page has links to topics such as installation, usage, and troubleshooting.

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

1. Unzip the `lpm_shiftreg_ex1_msim.zip` file to any working directory on your PC.
2. Start Modelsim-Altera.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.
5. On the Tools menu, click **Execute Macro**.
6. Select the `shiftreg_ex1.do` file and click **Open**. This is a script file for ModelSim that automates all necessary settings for the simulation.
7. Verify the results shown in the Waveform Viewer window.

You can rearrange signals, remove signals, add signals and change the radix by modifying the script in `shiftreg_ex1.do` accordingly to suit the results in the Quartus II Simulator.

Figure 2–15 and Figure 2–16 show the expected simulation results in ModelSim.

Figure 2–15. ModelSim Simulation Results, PISO

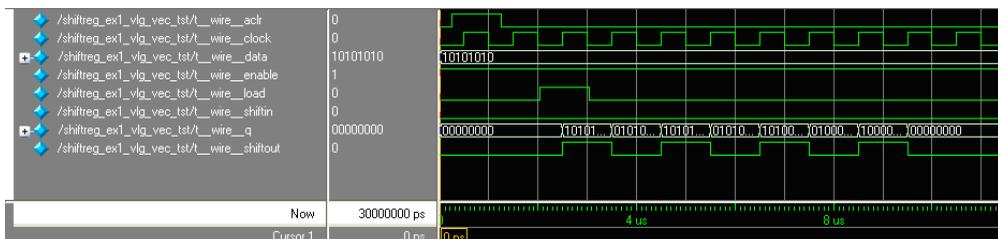
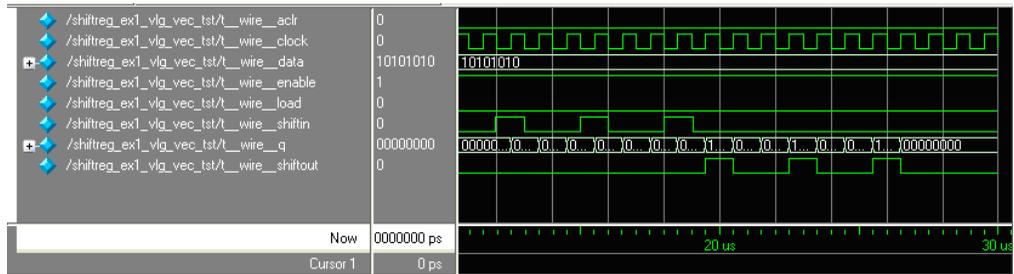


Figure 2–16. ModelSim Simulation Results, SIPO

Design Example: Time Delay

This design example uses the `lpm_shiftreg` megafunction to implement time delay functionality.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site:

<http://www.altera.com/support/examples/quartus/quartus.html>

Select the “Examples for `lpm_shiftreg` Megafunction User Guide” link from the examples page to download the design files.

Example 2

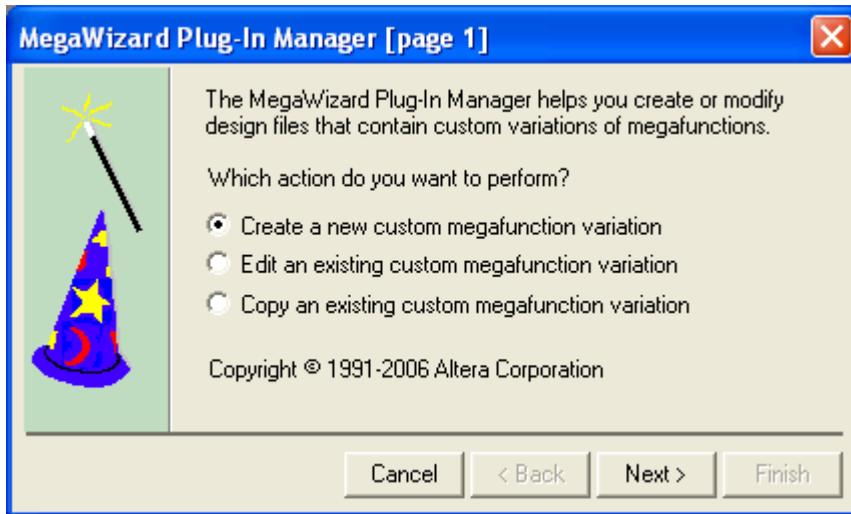
In this example, you perform the following tasks:

- Create a time delay
- Implement design and assign the EP1S10B672C6 device to project
- Compile and simulate the design

Generate the Time Delay Design

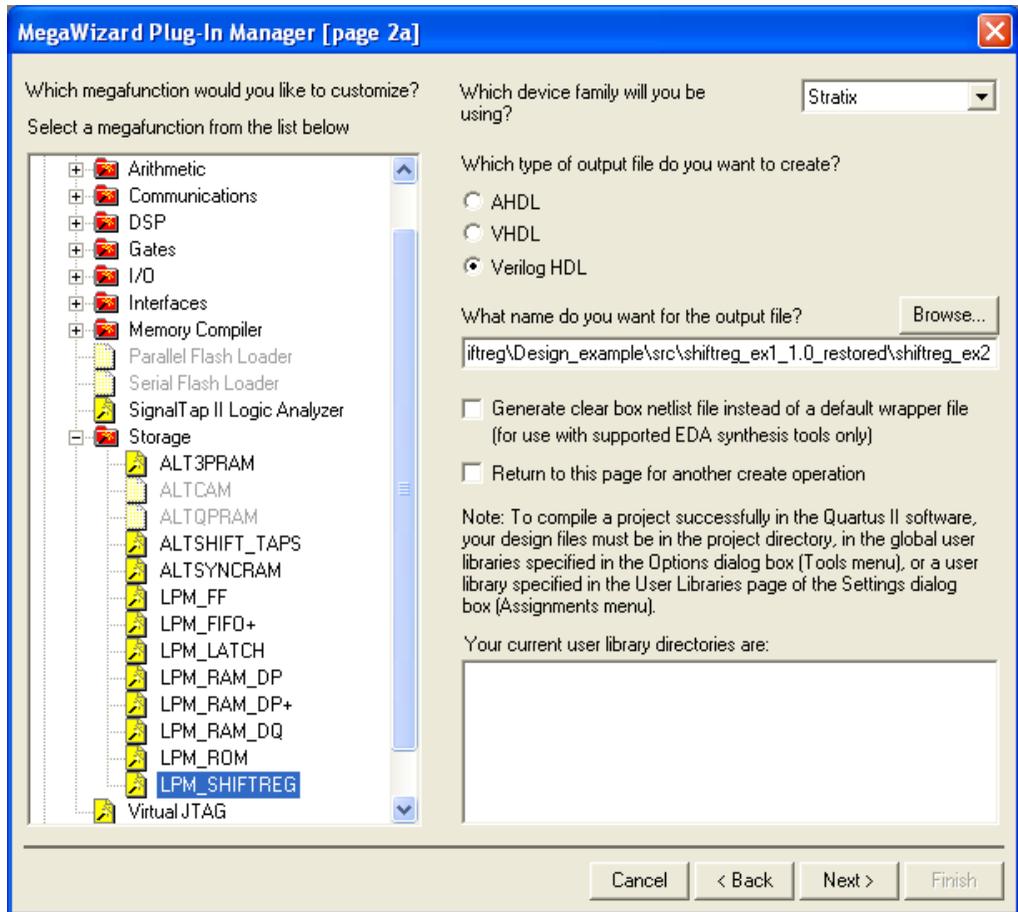
1. From the Quartus II software, open the `lpm_shiftreg_DesignExample_ex2.qar` project.
2. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears (Figure 2–17).

Figure 2–17. MegaWizard Plug-In Manager [page 1]

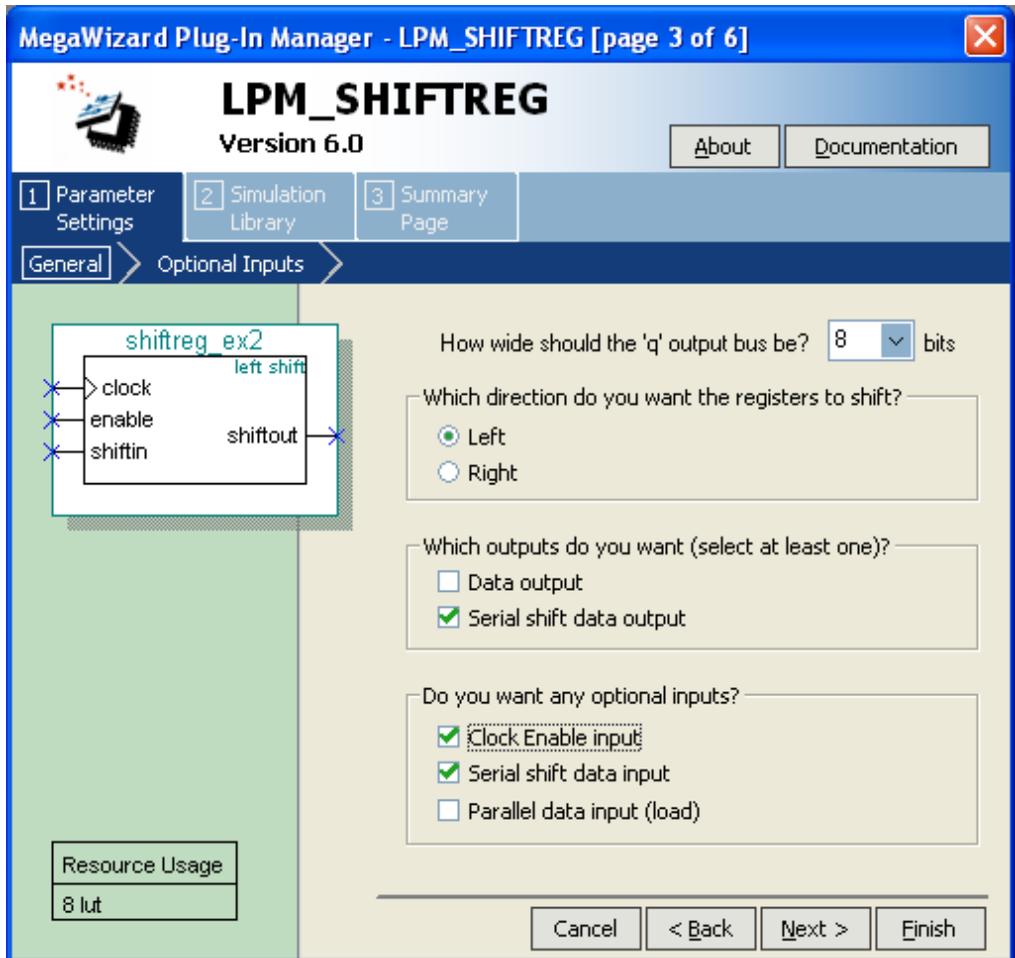


3. Select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page 2a displays (Figure 2–18).

Figure 2–18. MegaWizard Plug-In Manager [page 2a]



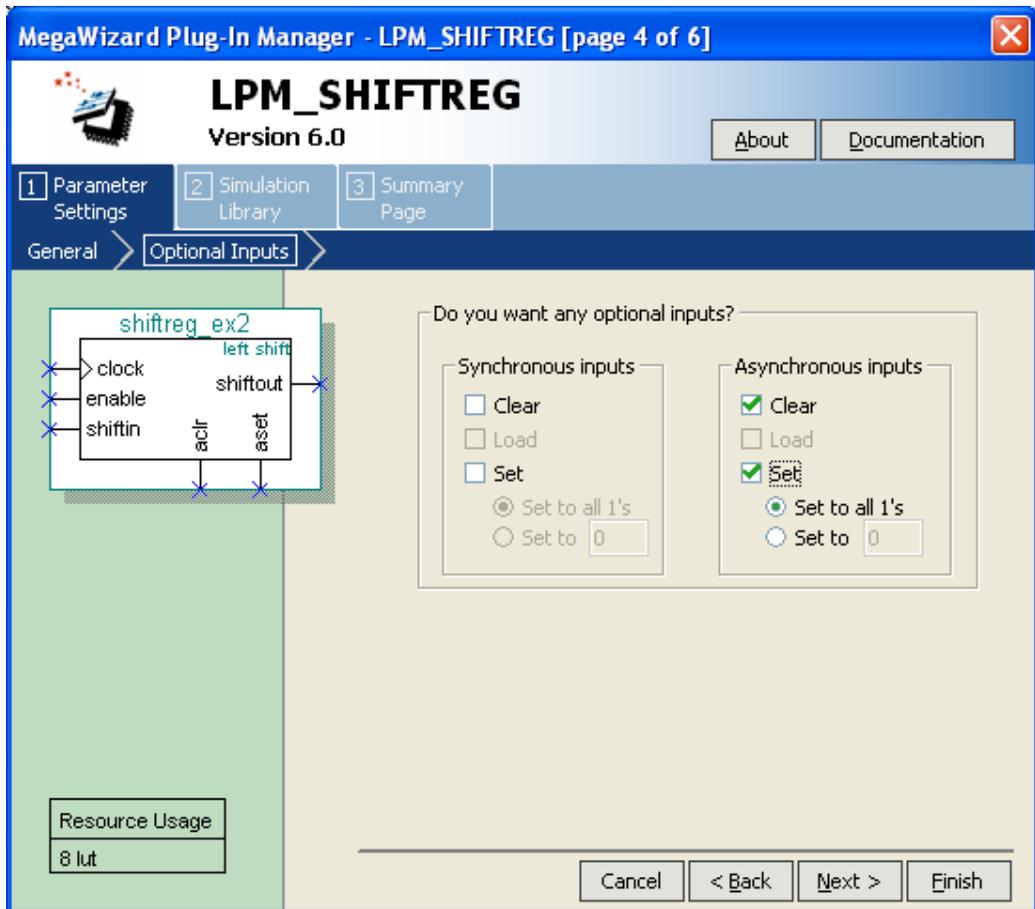
- In the **Storage** folder, select **LPM_SHIFTREG**. Specify the output file as **shiftreg_ex2**.
- In the **Which device family will you be using?** list, select **Stratix**.
- For **Which type of output file do you want to create?**, select **Verilog HDL**.
- Click **Next**. Page 3 appears (Figure 2–19).

Figure 2–19. *lpm_shiftreg Wizard [page 3 of 6]*

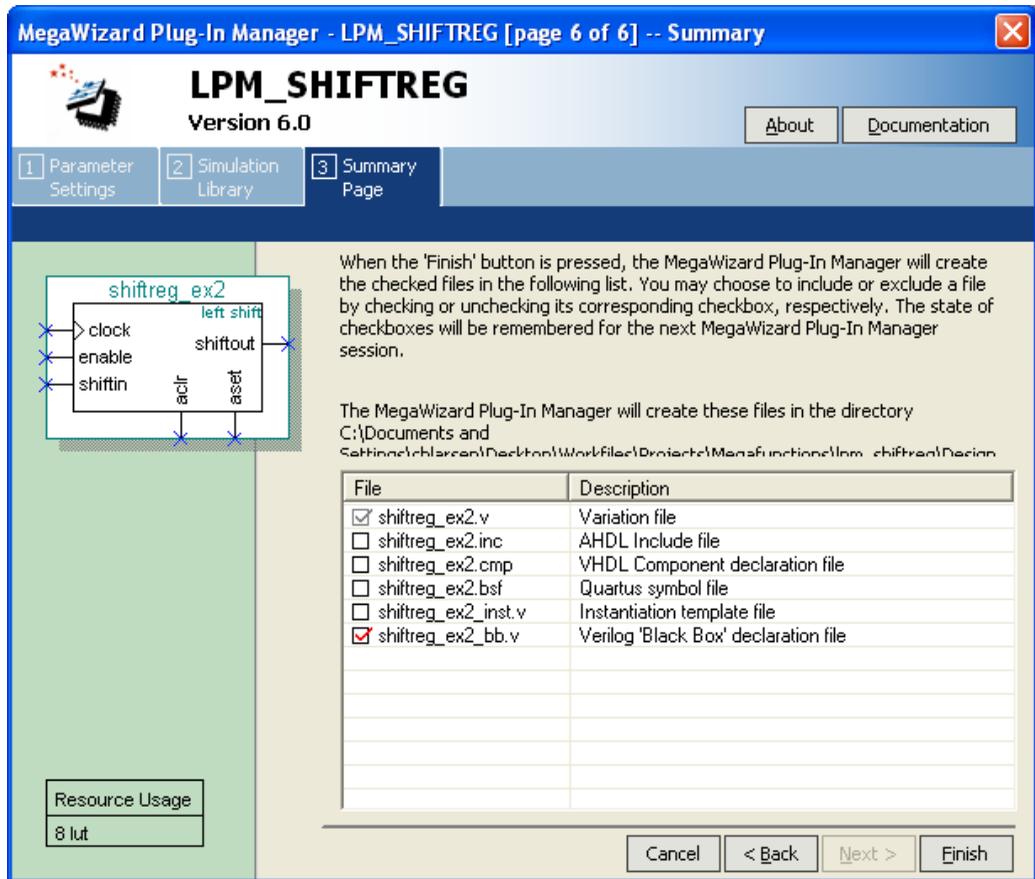
8. In the **How wide should the 'q' output bus be?** list, select 8.
9. Under **What direction do you want the registers to shift?**, select **Left**.
10. Under **Which outputs do you want (select at least one)?**, turn off **Data output** and turn on **Serial shift data output**.

11. Under **Do you want any optional inputs?**, turn on **Clock Enable input** and **Serial shift data input**, and turn off **Parallel data input (load)**.
12. Click **Next**. Page 4 appears (Figure 2–20).

Figure 2–20. *lpm_shiftreg Wizard [page 4 of 6]*



13. Under **Synchronous inputs**, turn off **Clear** and **Set**.
14. Under **Asynchronous inputs**, turn on **Clear** and turn off **Set**.
15. Click **Finish**. Page 6 appears (Figure 2–21).

Figure 2–21. *lpm_shiftreg Wizard [page 6 of 6] -- Summary*

16. Turn on the Verilog 'Black Box' declaration file option.
17. Turn off AHDL Include file, VHDL Component declaration file, Quartus symbol file, and Instantiation template file, click Finish.

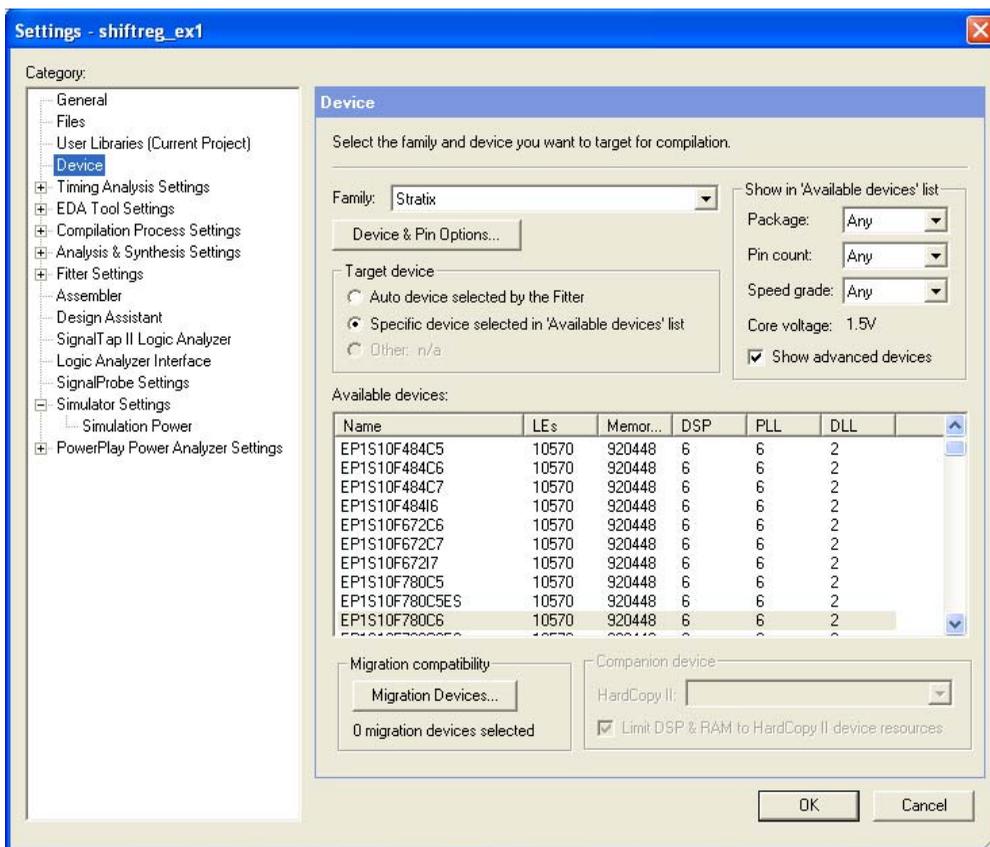
The `lpm_shiftreg` module is now built.

Implement the Time Delay Design

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

1. From the Quartus II software, on the Assignments menu, select **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Device**. The **Device Settings** dialog box appears (Figure 2–22).

Figure 2–22. Device Settings Dialog Box



3. In the **Family** list, select **Stratix**.

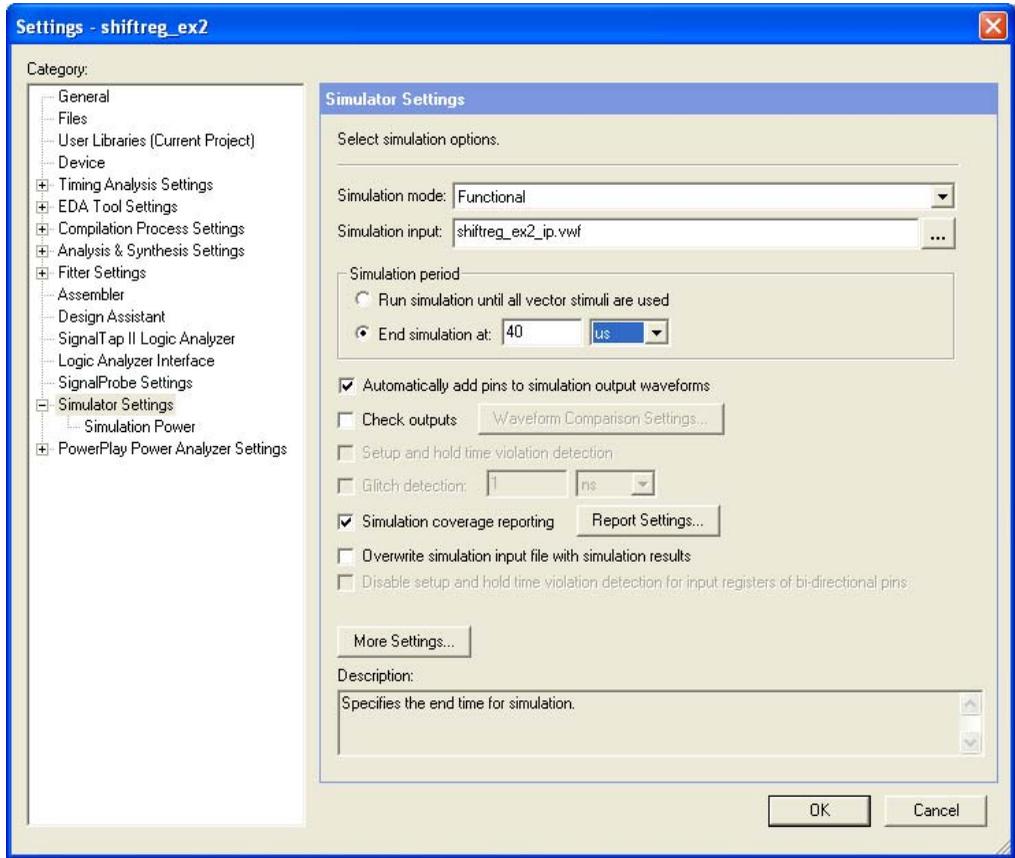
4. In the **Target device** list, click **Specific device selected in 'Available devices'** list.
5. Under **Show in 'Available devices' list**, select **EP1S10F780C6**.
6. Leave the other options in the default state and click **OK**.
7. On the Processing menu, click **Start Compilation**.
8. When the **Full compilation was successful** message box appears, click **OK**.

Functional Results—Simulate the Time Delay Design in Quartus II

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. From the Quartus II software, on the Processing menu, click **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful message** box appears, click **OK**.
3. On the Assignments menu, click **Settings**. The **Functional Simulation Settings** window appears.
4. In the **Category** list, select **Simulator Settings**. The **Simulator Settings** dialog box appears (Figure 2-23).

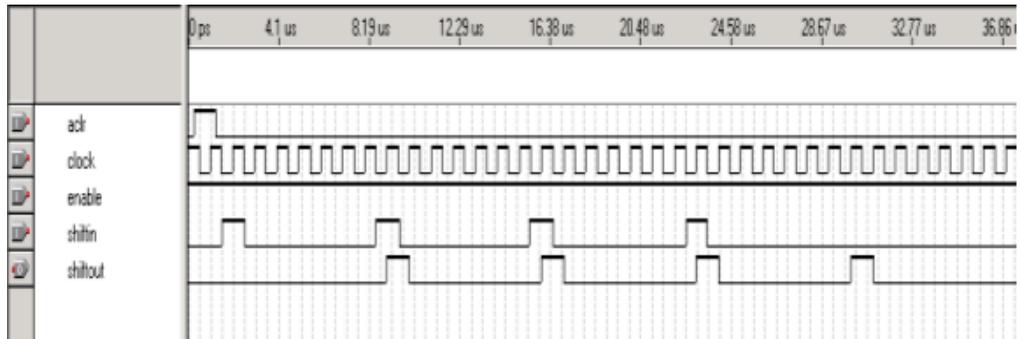
Figure 2–23. Simulator Settings Dialog Box



5. In the **Simulation mode** list, select **Functional**.
6. Type **shiftreg_ex2_ip.vwf** in the **Simulation input** box, or click **Browse (...)** to select the file in the project folder.
7. Turn on the **End simulation at:** option, type **40** and select **us**.
8. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting** options.
9. Turn off **Overwrite simulation input file with simulation results**, and click **OK**.
10. On the Processing menu, click **Start Simulation**.

11. When the **Simulation was successful** message box appears, click **OK**. The **Simulation Report** window displays. Verify the simulation waveform results (Figure 2-24).

Figure 2-24. Functional Simulation Waveform



Functional Results—Simulate the Time Delay 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 at: <http://www.altera.com/support/software/products/modelsim/mod-modelsim.html>. The support page has links to topics such as installation, usage, and troubleshooting.

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

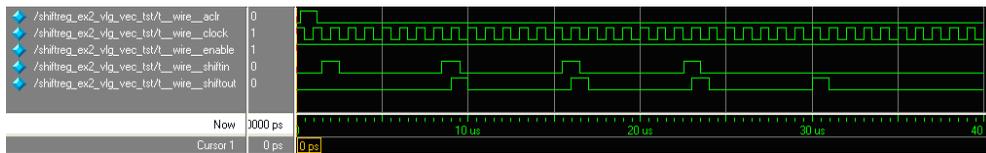
1. Unzip the **lpm_shiftreg_ex2_msim.zip** file to any working directory on your PC.
2. Start Modelsim-Altera.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.
5. On the Tools menu, click **Execute Macro**.

6. Select the **shiftreg_ex2.do** file and click **Open**. This is a script file for ModelSim that automates all necessary settings for the simulation.
7. Verify the results shown in the Waveform Viewer window.

You can rearrange signals, remove signals, add signals and change the radix by modifying the script in **shiftreg_ex2.do** accordingly to suit the results in the Quartus II Simulator.

Figure 2–25 shows the expected simulation results in ModelSim.

Figure 2–25. ModelSim Simulation Results



Conclusion

The Quartus II software provides parameterizable megafunctions ranging from simple arithmetic units, such as adders and counters, to advanced phase-locked loop (PLL) blocks, multipliers, and memory structures. These megafunctions 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. Altera recommends using these functions during design implementation so you can consistently meet your design goals.

Ports & Parameters

The options listed in this section describe all of the ports and parameters available for each device to customize the `lpm_shiftreg` megafunction according to your application. Figure 3–1 shows ports and parameters for the `lpm_shiftreg` megafunction.

Figure 3–1. Port and Parameter Description

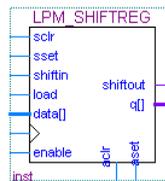


Table 3–1 shows input ports, Table 3–2 shows output ports, and Table 3–3 shows parameters.

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 the user of the MegaWizard Plug-In Manager interface.



Refer to the latest version of the Quartus® II Help for the most current information on the ports and parameters for this megafunction.

Table 3–1. `lpm_shiftreg` Megafunction Input Ports (Part 1 of 2)

Port Name	Required	Description	Comments
<code>data[]</code>	No	Data input to the shift register.	Input port <code>LPM_WIDTH</code> wide. At least one of <code>data</code> , <code>aset</code> , <code>aclr</code> , <code>sset</code> , <code>sclr</code> and/or <code>shiftin</code> ports must be used.
<code>clock</code>	Yes	Positive-edge-triggered clock.	

Table 3–1. *lpm_shiftreg* Megafunction Input Ports (Part 2 of 2)

Port Name	Required	Description	Comments
enable	No	Clock enable input.	Shift options use enable input for clock enable. Enable must be high (1) or unconnected for serial operation. Load must be high (1) and enable must be high or unconnected for parallel load operation.
Shiftin	No	Serial shift data input.	At least one of data, aset, aclr, sset, sclr and/or shiftin ports must be used. Default value is VCC.
load	No	Synchronous parallel load. High (1): load operation; low (0): shift operation.	Default is low (0) shift operation. For parallel load operation, load must be high (1) and enable must be high or unconnected.
sclr	No	Synchronous clear input.	If both sset and sclr are used and both are asserted, sclr is dominant. sclr signal affects q[] outputs before polarity is applied to ports.
sset	No	Synchronous set input.	Sets q outputs to value specified by LPM_SVALUE, if that value is present, or sets the q outputs to all 1s. If both sset and sclr are used and asserted, sclr is dominant. sset signal affects q[] outputs before polarity is applied to ports.
aclr	No	Asynchronous clear input.	If both aset and aclr are used and both are asserted, aclr is dominant. aclr signal affects the q[] outputs before polarity is applied to the ports.
aset	No	Asynchronous set input.	Sets q[] outputs to the value specified by LPM_AVALUE, if that value is present, or sets the q[] outputs to all 1s. If both aset and aclr are used and both are asserted, aclr is dominant. aset signal affects q[] outputs before polarity is applied to ports.

Table 3–2. *lpm_shiftreg* Megafunction Output Ports

Port Name	Required	Description	Comments
q[]	No	Data output from the shift register.	Output port LPM_WIDTH wide. Either q[] or shiftout or both must be used.
shiftout	No	Serial shift data output.	Either q[] or shiftout or both must be used. shiftout port value is equal to q[LPM_WIDTH-1] when LPM_DIRECTION="LEFT". When LPM_DIRECTION="RIGHT", shiftout equals q[0].

Table 3–3. *lpm_shiftreg* Megafunction Parameters

Parameter	Type	Required	Comments
LPM_WIDTH	Integer	Yes	Width of the <code>data[]</code> and <code>q</code> ports.
LPM_DIRECTION	String	No	Values are “LEFT”, “RIGHT”, and “UNUSED”. If omitted, default is “LEFT”. MSB is the leftmost bit, LSB is rightmost bit. The MSB is <code>q[LPM_WIDTH-1]</code> .
LPM_AVALUE	Integer / String	No	Constant value loaded when <code>aset</code> is high. If omitted, defaults to all 1s. The LPM_AVALUE parameter is limited to a maximum of 32 bits. Altera recommends that you specify this value as a decimal number for AHDL designs.
LPM_SVALUE	Integer / String	No	Constant value that is loaded on the rising edge of clock when <code>sset</code> is high. If omitted, defaults to all 1s. Altera recommends that you specify this value as a decimal number for AHDL designs.
LPM_HINT	String	No	Allows you to specify Altera-specific parameters in VHDL Design Files (<code>.vhd</code>). The default is “UNUSED”.
LPM_TYPE	String	No	Identifies library of parameterized modules (LPM) entity name in VHDL Design Files.

