# 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
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.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Date</th>
<th>Version</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>December</td>
<td>3.0</td>
<td>Updated Tables 1–1 and 1–2 to include Stratix III device support.</td>
</tr>
<tr>
<td>All</td>
<td>August 2006</td>
<td>2.0</td>
<td>Updated for Quartus II 6.0 software release.</td>
</tr>
<tr>
<td>All</td>
<td>July 2005</td>
<td>4.2</td>
<td>Updated for Quartus II 4.2 software release.</td>
</tr>
<tr>
<td>All</td>
<td>March 2005</td>
<td>1.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Information Type</th>
<th>USA &amp; Canada</th>
<th>All Other Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td><a href="http://www.altera.com/mysupport/">www.altera.com/mysupport/</a></td>
<td>altera.com/mysupport/</td>
</tr>
<tr>
<td></td>
<td>(800) 800-EPLD (3753)</td>
<td>(408) 544-7000 (1)</td>
</tr>
<tr>
<td></td>
<td>(7:00 a.m. to 5:00 p.m. Pacific Time)</td>
<td>(7:00 a.m. to 5:00 p.m. Pacific Time)</td>
</tr>
<tr>
<td>Product literature</td>
<td><a href="http://www.altera.com">www.altera.com</a></td>
<td><a href="http://www.altera.com">www.altera.com</a></td>
</tr>
<tr>
<td>Altera literature services</td>
<td><a href="mailto:literature@altera.com">literature@altera.com</a> (1)</td>
<td><a href="mailto:literature@altera.com">literature@altera.com</a> (1)</td>
</tr>
<tr>
<td>Non-technical customer service</td>
<td>(800) 767-3753</td>
<td>(408) 544-7000</td>
</tr>
<tr>
<td></td>
<td>(7:00 a.m. to 5:00 p.m. Pacific Time)</td>
<td>(7:30 a.m. to 5:30 p.m. Pacific Time)</td>
</tr>
<tr>
<td>FTP site</td>
<td>ftp.altera.com</td>
<td>ftp.altera.com</td>
</tr>
</tbody>
</table>

Note to table:
(1) You can also contact your local Altera sales office or sales representative.
This document uses the typographic conventions shown below.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: <em>Save As</em> dialog box.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: <em>fMAX</em>, <em>\qdesigns</em> directory, <em>d:</em> drive, <em>chiptrip.gdf</em> file.</td>
</tr>
<tr>
<td><strong>Italic Type with Initial Capital Letters</strong></td>
<td>Document titles are shown in italic type with initial capital letters. Example: <em>AN 75: High-Speed Board Design.</em></td>
</tr>
<tr>
<td><strong>Italic type</strong></td>
<td>Internal timing parameters and variables are shown in italic type. Examples: <em>tPIA</em>, <em>n + 1.</em></td>
</tr>
<tr>
<td></td>
<td>Variable names are enclosed in angle brackets (&lt; &gt;) and shown in italic type. Example: <em>&lt;file name&gt;</em>, <em>&lt;project name&gt;pof</em> file.</td>
</tr>
<tr>
<td><strong>Initial Capital Letters</strong></td>
<td>Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.</td>
</tr>
<tr>
<td><strong>“Subheading Title”</strong></td>
<td>References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”</td>
</tr>
<tr>
<td><strong>Courier type</strong></td>
<td>Signal and port names are shown in lowercase Courier type. Examples: <em>data1</em>, <em>tdi</em>, <em>input</em>. Active-low signals are denoted by suffix <em>n</em>, e.g., <em>resetn.</em></td>
</tr>
<tr>
<td></td>
<td>Anything that must be typed exactly as it appears is shown in Courier type. For example: <em>c:\qdesigns\tutorial\chiptrip.gdf</em>. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword <em>SUBDESIGN</em>), as well as logic function names (e.g., <em>TRI</em>) are shown in Courier.</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., etc.</td>
<td>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.</td>
</tr>
<tr>
<td>■ ●</td>
<td>Bullets are used in a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td>✔</td>
<td>The checkmark indicates a procedure that consists of one step only.</td>
</tr>
<tr>
<td>!</td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td>!</td>
<td>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.</td>
</tr>
<tr>
<td>!</td>
<td>The warning indicates information that should be read prior to starting or continuing the procedure or processes.</td>
</tr>
<tr>
<td>←</td>
<td>The angled arrow indicates you should press the Enter key.</td>
</tr>
<tr>
<td>✈</td>
<td>The feet direct you to more information on a particular topic.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Device Family</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix® III</td>
<td>Full</td>
</tr>
<tr>
<td>Stratix II</td>
<td>Full</td>
</tr>
<tr>
<td>Stratix II GX</td>
<td>Full</td>
</tr>
<tr>
<td>Stratix GX</td>
<td>Full</td>
</tr>
<tr>
<td>Stratix</td>
<td>Full</td>
</tr>
<tr>
<td>Cyclone™ II</td>
<td>Full</td>
</tr>
<tr>
<td>Cyclone</td>
<td>Full</td>
</tr>
<tr>
<td>HardCopy® II</td>
<td>Full</td>
</tr>
<tr>
<td>HardCopy Stratix</td>
<td>Full</td>
</tr>
<tr>
<td>MAX® II</td>
<td>Full</td>
</tr>
<tr>
<td>MAX 7000</td>
<td>Full</td>
</tr>
<tr>
<td>MAX 3000</td>
<td>Full</td>
</tr>
<tr>
<td>APEX™ II</td>
<td>Full</td>
</tr>
<tr>
<td>APEX 20KC</td>
<td>Full</td>
</tr>
<tr>
<td>APEX 20KE</td>
<td>Full</td>
</tr>
<tr>
<td>FLEX® 10K</td>
<td>Full</td>
</tr>
<tr>
<td>FLEX 10KA</td>
<td>Full</td>
</tr>
<tr>
<td>FLEX 10KE</td>
<td>Full</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLEX 6000</td>
<td>Full</td>
</tr>
<tr>
<td>ACEX® 1K</td>
<td>Full</td>
</tr>
</tbody>
</table>
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 \texttt{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 \texttt{q[]} is used to read parallel data from the shift register. Parallel data is always available on the \texttt{q[]} outputs at every clock. When data is shifted serially with every clock, you get the MSB of the \texttt{q[]} output on the 'shiftout' pin. A shift operation is shown in Figure 1–1.

\textbf{Figure 1–1. Shift Operation}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1.jpg}
\caption{Shift Operation}
\end{figure}

\textbf{Common Applications}

Use the \texttt{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.

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

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Optimization (1)</th>
<th>Width</th>
<th>Logic Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix III</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 ALUT</td>
</tr>
<tr>
<td>Stratix II</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 ALUT</td>
</tr>
<tr>
<td>Stratix GX</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 logic elements</td>
</tr>
<tr>
<td>Stratix</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 logic elements</td>
</tr>
<tr>
<td>Cyclone II</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 logic elements</td>
</tr>
<tr>
<td>Cyclone</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 logic elements</td>
</tr>
<tr>
<td>MAX II</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 logic elements</td>
</tr>
<tr>
<td>MAX 7000</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>MAX 3000</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>APEX II</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>APEX 20KC</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>APEX 20KE</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>FLEX 10K</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>FLEX 10KA</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>FLEX 10KE</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>FLEX 6000</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
<tr>
<td>ACEX 1K</td>
<td>Balanced</td>
<td>8-Bit</td>
<td>8 registers</td>
</tr>
</tbody>
</table>

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.
Chapter 2. Getting Started

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).
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)**

![MegaWizard Plug-In Manager](image)

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)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>How wide should the 'q' output bus be?</td>
<td>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.</td>
</tr>
<tr>
<td>Which direction do you want the registers to shift?</td>
<td>Select 'left' or 'right' to define the direction of data shift.</td>
</tr>
</tbody>
</table>
On page 4, specify synchronous and asynchronous inputs (Figure 2–4).

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which outputs do you want (select at least one)?</td>
<td>Select 'Data output' (parallel output), 'Serial shift data output', or select both.</td>
</tr>
<tr>
<td>Do you want any optional inputs?</td>
<td>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.</td>
</tr>
</tbody>
</table>
On page 6 of the wizard, specify the types of files to be generated. Choose from HDL wrapper file, (\texttt{<function name>}.v | .vhd | .tdf), Block Symbol file (.bsf), Instantiation template file (\texttt{<function name>}_inst.v), or Verilog Black Box declaration file (\texttt{<function name>}_bb.v) (Figure 2–5).
For more information about the ports for the lpm_shiftreg megafunction, refer to the Specifications chapter in this User Guide.

**Inferring Megafonctions 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 megafonctions which are optimized for Altera devices, so the area and/or performance may be better than generic HDL code. Use megafonctions 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](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).
3. Select **Create a new custom megafunction variation**, and click **Next**. Page 2a of the **MegaWizard Plug-In Manager** appears (Figure 2–7).
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**).
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).
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**).
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.
2. Under **Category**, select **Device** (Figure 2–11).

**Figure 2–11. Device Settings Dialog Box**

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

4. Under **Target device**, click **Specific device selected in ‘Available devices’ list**.

5. In the **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** 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)*.
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:
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.


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
Design Example: Time Delay

This design example uses the `lpm_shiftreg` megafuction 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 Megafuction 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).
3. Select **Create a new custom megafuction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page 2a displays (Figure 2–18).
4. In the **Storage** folder, select **LPM_SHIFTREG**. Specify the output file as **shiftreg_ex2**.

5. In the **Which device family will you be using?** list, select **Stratix**.

6. For **Which type of output file do you want to create?**, select **Verilog HDL**.

7. Click **Next**. Page 3 appears (Figure 2–19).
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).
Design Example: Time Delay

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).

3. In the Family list, select Stratix.
4. In the Target device list, click Specific device selected in ‘Available devices’ list.


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.


4. In the Category list, select Simulator Settings. The Simulator Settings dialog box appears (Figure 2–23).
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.


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.
Conclusion
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>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>data[]</td>
<td>No</td>
<td>Data input to the shift register.</td>
<td>Input port LPM_WIDTH wide. At least one of data, aset, aclr, sset, sclr and/or shiftin ports must be used.</td>
</tr>
<tr>
<td>clock</td>
<td>Yes</td>
<td>Positive-edge-triggered clock.</td>
<td></td>
</tr>
</tbody>
</table>
### Ports & Parameters

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

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>No</td>
<td>Clock enable input.</td>
<td>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.</td>
</tr>
<tr>
<td>Shiftin</td>
<td>No</td>
<td>Serial shift data input.</td>
<td>At least one of data, aset, aclr, sset, sclr and/or shiftin ports must be used. Default value is VCC.</td>
</tr>
<tr>
<td>load</td>
<td>No</td>
<td>Synchronous parallel load. High (1): load operation; low (0): shift operation.</td>
<td>Default is low (0) shift operation. For parallel load operation, load must be high (1) and enable must be high or unconnected.</td>
</tr>
<tr>
<td>sclr</td>
<td>No</td>
<td>Synchronous clear input.</td>
<td>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.</td>
</tr>
<tr>
<td>sset</td>
<td>No</td>
<td>Synchronous set input.</td>
<td>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.</td>
</tr>
<tr>
<td>aclr</td>
<td>No</td>
<td>Asynchronous clear input.</td>
<td>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.</td>
</tr>
<tr>
<td>aset</td>
<td>No</td>
<td>Asynchronous set input.</td>
<td>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.</td>
</tr>
</tbody>
</table>

#### Table 3–2. *lpm_shiftreg* Megafunction Output Ports

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>q[]</td>
<td>No</td>
<td>Data output from the shift register.</td>
<td>Output port LPM_WIDTH wide. Either q[] or shiftout or both must be used.</td>
</tr>
<tr>
<td>shiftout</td>
<td>No</td>
<td>Serial shift data output.</td>
<td>Either q[] or shiftout or both must be used. shiftout port value is equal to q[LPM_WIDTH-1] when LPM_DIRECTION=&quot;LEFT&quot;. When LPM_DIRECTION=&quot;RIGHT&quot;, shiftout equals q[0].</td>
</tr>
</tbody>
</table>
### Table 3–3. *lpm_shiftreg* Megafunction Parameters

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