LPM_SHIFTREG Megafunction

2013.03.05
UG-033105

Subscribe Feedback

This document describes the Altera®-provided megafunction IP core 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:

- 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 memory compilation IP core accessible from the Quartus II® MegaWizard® Plug-In Manager.

Shift registers are a type of sequential logic circuit, that mainly store digital data. These cores are comprised of 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. 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. Most of the registers possess no characteristic internal sequence of states.

The shift register megafunction is highly parameterizable block of logic. You can use this megafunction to implement long delay chains. The megafunction provides for either left shift or right shift of the input data bits. Shifted data 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. 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 theq[] pin.

Common Applications

Use the optimized LPM_SHIFTREG megafunction to replace all other types of shift register functions in your design.

You can perform the following functions with shift registers:

- 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).
- Shift registers 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 is effective at shifting the data in or out of digital systems. Some of the common applications of a shift register include serial to parallel conversion, parallel to serial conversion, and delay generation for multistage pipeline stages.

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.

The table below shows level of support for each Altera device family.

Table 1: Device Family Support

Device Family	Support
Stratix® V (E, GX, GT, GZ)	Preliminary
Stratix IV (E, GX, GT)	Full
Stratix III (L and E)	Full
Stratix II	Full
Stratix II GX	Full
Stratix GX	Full

Device Family	Support
Stratix	Full
Cyclone® V (E, GX, GT, SE, SX)	Preliminary
Cyclone IV (E and GX)	Full
Cyclone III	Full
Cyclone II	Full
Cyclone	Full
Arria® V (GX, GT, GZ, SX, ST)	Preliminary
Arria II (GX and GZ)	Full
Arria GX	Full
HardCopy® IV (E and GX)	Full
HardCopy III	Full
HardCopy II	Full
HardCopy Stratix	Full
MAX® II	Full
MAX 7000	Full
MAX 3000A	Full

Resource Utilization

The LPM_SHIFTREG megafunction uses one unit of logic per bit.

The **MegaWizard® Plug-In Manager** reports approximate resource utilization based on your specification and parameters. Resource Utilization is available in the lower left corner of the MegaWizard Plug-In Manager screen.

Customizing Megafunctions in the GUI

The MegaWizard Plug-In Manager GUI allows you to define and instantiate a custom variation of an Altera megafunction. You can edit megafunctions at any time. Megafunctions defined in your project appear in the Project Navigator. To edit a megafunction, double-click the megafunction file in the Project Navigator or Block Editor to display the MegaWizard GUI. To customize a megafunction using the MegaWizard Plug-In Manager GUI, follow these steps:

1. Launch the MegaWizard Plug-In Manager using any of the following methods:

- In the Quartus II software, click **Tools** > **MegaWizard Plug-In Manager**.
- In the Project Navigator, right-click a megafunction file and click **MegaWizard Plug-In Manager**.
- In the Block Editor, click **Edit** > **Insert Symbol as Block**. In the Symbol editor, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager GUI by typing the following command at the command prompt:

`qmegawiz` (for GUI or command-line mode)

or

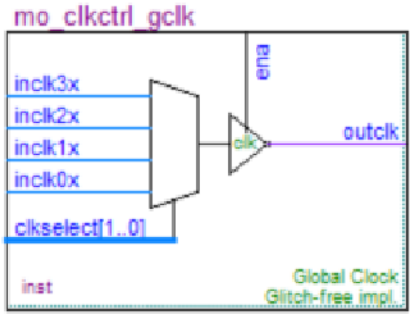
`qmegawizq` (for GUI only)

2. Specify **Create**, **Edit**, or **Copy** a megafunction.
3. In **Which device family will you be using?** select your target device family.

   Only megafunctions available for the target device are available in **Which megafunction would you like to customize?** Unsupported megafunctions are grayed out.
4. Specify the name and file format of the output file. Click **Next**
5. Parameterize the megafunction by specifying options in the wizard. Click **Next** .
6. If the wizard includes **EDA** and **Summary** tabs, follow these steps:
   - Some third-party synthesis tools can use a grey box netlist that contains the structure of an IP core without detailed logic to optimize timing and performance of the design containing it. To use this feature, turn on **Generate Netlist** to generate a netlist file for area and timing estimation instead of a wrapper file.
   - On the **Summary** tab, select the files you want to generate. A gray checkmark indicates a file that is automatically generated. All other files are optional. This step instantiates the megafunction into your HDL code and creates a wrapper file.
7. Click **Finish**. The megafunction variation is generated along with the files you specify.
8. To view the megafunction schematic, open the generated block symbol file (**.bsf**) located in your project directory. The megafunction block symbol appears in the Symbol window.

   You can edit megafunctions at any time. Megafunctions defined in your project appear in the Project Navigator. To edit a megafunction, double-click the megafunction file in the Project Navigator or Block Editor to display the MegaWizard GUI.

Figure 1: Example Parameterized Global Clock Control Module



Related Information

Creating a System with Qsys

Parameters in the MegaWizard Plug-In Manager GUI

The LPM_SHIFTREG wizard guides you through the process of specifying parameters for the megafunction. The following table lists parameters specific to the LPM_SHIFTREG megafunction:

Table 2: LPM_SHIFTREG MegaWizard Plug-in Manager Parameters

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.
Which outputs do you want (select at least one)?	Select **Data output**, **Serial shift data output**, or select both.
Do you want any optional inputs? (MegaWizardpage 3)	Select one or more optional inputs to shift register. • Select **Serial shift data input** to put serial data into the shift register. This is the default setting. • Select **Clock Enable input** to provide enable function to clock input. • Select **Parallel data input (load)** to load shift register with parallel data.
Do you want any optional inputs? (MegaWizard page 4)	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. Turn on **Set** to either select **Set all to 1's** or to a particular value specified in **Set to** field. **Note:** The `sclr` signal affects `q[]` outputs before polarity is applied to ports. If both `sset` and `sclr` are used and both are asserted, `sclr` is dominant.
Asynchronous inputs.	Shift register has optional asynchronous clear and set options. Use **Clear** input to clear all registers asynchronously. Turn on **Set** to either select **Set all to 1's** or to a particular value specified in **Set to** field. **Note:** The aclr signal affects`q[]` outputs before polarity is applied to ports. If both `aset` and `aclr` are used and both are asserted, `aclr` is dominant.

Function	Description
Turn on the files you wish to generate.	Some files are automatially selected by the MegaWizard Plug-In Manager. The choices are: • Variation file, (<*function name*>**.v**/**.vhd**/**.tdf**) • Block Symbol file (**.bsf**) • Instantiation template file (<*function name*>**_inst.v**) • VHDL component declaration file (<*function name*>**.cmp**) • Verilog Black Box declaration file (<*function name*>**_bb.v**)

Related Information

Ports and Parameters on page 12

Infer 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.

Related Information

Recommended HDL Coding Styles

Instantiate Megafunctions in HDL Code

You can instantiate a megafunction directly in your Verilog HDL, or VHDL, code by calling the megafunction and setting its parameters as you would in any other module, component, or subdesign. When instantiating a megafunction in VHDL, be sure to include the correct libraries.

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.

Related Information

Recommended HDL Coding Styles

Quartus II Integrated Synthesis

Synopsys Synplify Support

Mentor Graphics Precision Synthesis Support

LPM_SHIFTREG AHDL Function Prototype

The following AHDL function prototype is located in the AHDL Include File (**.inc**) **lpm_shiftreg.inc** in the ***<Quartus II installation directory>*\libraries\megafunctions** directory.

Note: Port name and order also apply when used in Verilog HDL.

```
FUNCTION lpm_shiftreg (data[LPM_WIDTH-1..0], clock, enable,
                       shiftin, load,
                       sclr, sset,
                       aclr, aset)
                  WITH (LPM_WIDTH, LPM_DIRECTION, LPM_AVALUE, LPM_SVALUE)

                  RETURNS(q[LPM_WIDTH-1..0], shiftout);
```

LPM_SHIFTREG VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the ***<Quartus II installation directory>*\libraries\vhd\lpm** directory.

```
component LPM_SHIFTREG

   generic (LPM_WIDTH : natural;    -- MUST be greater than 0

            LPM_AVALUE : string := "UNUSED";
            LPM_SVALUE : string := "UNUSED";
            LPM_PVALUE : string := "UNUSED";
            LPM_DIRECTION: string := "UNUSED";
            LPM_TYPE: string := L_SHIFTREG;
            LPM_HINT : string := "UNUSED");

   port (DATA : in std_logic_vector(LPM_WIDTH-1 downto 0)
         := (OTHERS => '0');

            CLOCK : in std_logic;
            ENABLE : in std_logic := '1';
            SHIFTIN : in std_logic := '1';
            LOAD : in std_logic := '0';
            SCLR : in std_logic := '0';
            SSET : in std_logic := '0';
            ACLR : in std_logic := '0';
            ASET : in std_logic := '0';
            Q : out std_logic_vector(LPM_WIDTH-1 downto 0);
             SHIFTOUT : out std_logic);

end component;
```

The following VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE lpm.lpm_components.all;
```

Simulating Megafunctions

Simulation verifies design behavior before device programming. The Quartus II software supports RTL and gate level design simulation of megafunction IP cores in other EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation. Altera provides various tools to help you quickly setup and run simulation. You can use the Quartus II NativeLink feature to automatically generate simulation files and scripts. NativeLink launches your preferred simulator a from within the Quartus II software.

Use a custom flow for more control over all aspects of simulation file generation. Alternatively, the Simulation Library Compiler automatically compiles and stores the correct simulation model libraries for functional and gate-level timing simulation of your design.

Related Information

Simulating Altera Designs

Debugging with the 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.

Related Information

Design Debugging Using the SignalTap II Logic Analyzer

Design Example: Configurable 8Bit 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. In this example, you create an 8-bit SIPO or PISO shift register.

- Generating an 8-bit shift register using the LPM_SHIFTREG megafunction and the MegaWizard Plug-in Manager.
- Implement design and assign the **5SGMD4E1H29C1** Stratix V GS device to the project.
- Compile and simulate the design.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site: Select the “Examples for lpm_shiftreg Megafunction User Guide” link from the examples page to download the design files.

Related Information

Design Examples

Generating a Configurable 8-Bit SIPO or PISO Shift Register

To build and configure the LPM_SHIFTREG megafunction with the configurable 8-bit SIPO or PISO shift register design example, perform the following steps:

1. In the Quartus II software, open the **lpm_shiftreg_DesignExample_ex1.qar** project.
2. On the Tools menu, click MegaWizard Plug-In Manager.
3. On page 1 of the MegaWizard Plug-In Manager, select **Create a new custom megafunction variation**, and click **Next**.
4. On Page 2a of the MegaWizard Plug-In Manager select **Stratix V** from the **Which device family will you be using?** list.
5. Click **Verilog HDL** under **Which type of output file do you want to create?**.
6. Expand the **Memory Compiler** folder and select **LPM_SHIFTREG**. Specify the output file **shiftreg_ex1**.
7. Click **Next**.
8. On Page 3, set the width of the output bus in he **How wide should the 'q' output bus be?** list to **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**.
13. On page 4, under **Synchronous inputs**, turn off **Clear** and **Set**.
14. Under **Asynchronous inputs**, turn on **Clear** and **Set**.
15. Click **Finish**.
16. Turn on **Verilog HDL black-box 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.

Implementing the Design Example

In this example, you assign the 5SGMD4E1H29C1 device to the project and compile the project.

1. In the Quartus II software, click **Assignments** > **Device**.
2. Under **Device family**, select **Stratix V (GS/GT/GX/E)** from the **Family** list.
3. Select **Stratix V GS Mainstream** on the**Devices** list.
4. Select **Specific device selected in 'Available devices' list**.
5. In the **Available devices** list, select **5SGMD4E1H29C1**.
6. Leave the other options in the default state and click **OK**.
7. On the Processing menu, click **Start Compilation**.

   A green check mark appears next to **Compile Design** in the **Tasks** window when compilation is complete.

Simulating the 8-Bit Shift Register in ModelSim-Altera

You can simulate the 8-bit shift register example in ModelSim-Altera.

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_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**.

Note: This is a script file for ModelSim that automates all necessary settings for the simulation.

You can rearrange signals, remove signals, add signals and change the radix by modifying the script in **shiftreg_ex1.do** accordingly to adjust the results in your preferred simulator.

Design Example: Time Delay

This design example uses the LPM_SHIFTREG megafunction to implement time delay functionality. In this example you implement a time delay with the LPM_SHIFTREG megafunction.

- Generating a time delay module.
- Implement design and assign the **5SGMD4E1H29C1** Stratix V GS device to the project.
- Compile and simulate the design.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site: Select the "Examples for lpm_shiftreg Megafunction User Guide" link from the examples page to download the design files.

Related Information

Design Examples

Generating the Time Delay Design

To build and configure the LPM_SHIFTREG megafunction with the time delay design example, perform the following steps:

1. In the Quartus II software, open the **lpm_shiftreg_DesignExample__ex2.qar** project.
2. On the Tools menu, click. **MegaWizard Plug-In Manager**.
3. On page 1 of the MegaWizard Plug-In Manager, select **Create a new custom megafunction variation**, and click **Next**.
4. On Page 2a of the MegaWizard Plug-In Manager, select **Stratix V** from the **Which device family will you be using?** list.
5. Click **Verilog HDL** under **Which type of output file do you want to create?**.
6. Expand the **Memory Compiler** folder and select **LPM_SHIFTREG**. Specify the output file **shiftreg_ex2**.
7. Click **Next**.
8. On Page 3, set the width of the output bus in the **How wide should the 'q' output bus be?** list to **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 off the **Data output** option and turn on the **Serial shift data output** option.
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**.
13. On page 4, under **Synchronous inputs**, turn off the**Clear** and **Set** options.
14. Under **Asynchronous inputs**, turn on the**Clear** option and turn off the **Set** option.
15. Click **Finish**.
16. Turn on **Verilog HDL black-box 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.

Implementing the Design Example

In this example, you assign the 5SGMD4E1H29C1 device to the project and compile the project.

1. In the Quartus II software, click **Assignments** > **Device**.
2. Under **Device family**, select **Stratix V (GS/GT/GX/E)** from the **Family** list.
3. Select **Stratix V GS Mainstream** on the**Devices** list.
4. Select **Specific device selected in 'Available devices' list**.
5. In the **Available devices** list, select **5SGMD4E1H29C1**.
6. Leave the other options in the default state and click **OK**.
7. On the Processing menu, click **Start Compilation**.

A green check mark appears next to **Compile Design** in the **Tasks** window when compilation is complete.

Simulating the Time Delay Design in ModelSimAltera

You can simulate the time delay example in ModelSim-Altera. 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**.

Note: This is a script file for ModelSim that automates all necessary settings for the simulation.

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 your preferred simulator.

Ports and 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.

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.

LPM_SHIFTREG Megafunction Input Ports on page 12

LPM_SHIFTREG Megafunction Output Ports on page 13

LPM_SHIFTREG Megafunction Parameters on page 13

LPM_SHIFTREG Megafunction Input Ports

Table 3: Input Ports for the LPM_SHIFTREG Megafunction

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

Port Name	Required	Description	Comments
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.

LPM_SHIFTREG Megafunction Output Ports

Table 4: Output Ports for the LPM_SHIFTREG Megafunction

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].

LPM_SHIFTREG Megafunction Parameters

Table 5: Parameters for the LPM_SHIFTREG Megafunction

Parameter	Type	Required	Comments
LPM_WIDTH	Integer	Yes	Width of the data[] and q 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 q[LPM_WIDTH-1].

Parameter	Type	Required	Comments
LPM_AVALUE	Integer/ String	No	Constant value loaded when aset 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 sset 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 (.vhd)**. The default is "UNUSED".
LPM_TYPE	String	No	Identifies library of parameterized modules (LPM) entity name in VHDL Design Files.

Revision History

Date	Version	Changes
May 2013	2013.05.06	New output format. Updated for Arria V, Cyclone V, and Stratix V devices. Removed outdated content.
December 2006	3.0	Updated table 1-1 to include Stratix III device support.
August 2006	2.0	Updated for Quartus II 6.0 software release.
July 2005	1.1	Updated for Quartus II 4.2 software release.
March 2005	1.0	Initial release.