RAM-Based Shift Register (ALTSHIFT_TAPS) Megafunction

User Guide

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

UG-01009-2.1

Document last updated for Altera Complete Design Suite version: 10.1
Document publication date: November 2010

Subscribe

Contents

1. About this Megafunction

Device Family Support

The ALTSHIFT_TAPS megafunction supports the following target Altera® device families:

- Arria® GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® II
- HardCopy Stratix®
- Stratix IV
- Stratix III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX
- ACEX® 1K
- APEX™ II
- APEX 20KC
- APEX 20KE
- FLEX® 10K
- FLEX 10KA
- FLEX 10KE

Introduction

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

Altera provides a RAM-based shift register megafunction called ALTSHIFT_TAPS that contains additional features not found in a conventional shift register. Traditional shift registers implemented with standard flip-flops use many logic cells for large shift registers. The ALTSHIFT_TAPS megafunction is implemented in the device memory blocks, saving logic cells and routing resources. In a complicated design such as a digital signal processing (DSP) application that requires local data storage, it is more efficient to implement an ALTSHIFT_TAPS megafunction as the shift register.

The ALTSHIFT_TAPS megafunction is a parameterized shift register with taps. The taps provide data outputs from the shift register at certain points in the shift register chain. You can add additional logic that uses the output from these taps for further applications. The megafunction's output tap feature is useful for applications such as the Linear Feedback Shift Register (LFSR) and Finite Impulse Response (FIR) filters.

Features

The ALTSHIFT_TAPS megafunction implements a shift register with taps and offers additional features, which include:

- Selectable RAM block type
- A wide range of widths for the `shiftin` and `shiftout` ports
- Support for output taps at certain points in the shift register chain
- Selectable distance between taps

General Description

The ALTSHIFT_TAPS megafunction can be easily configured and built through the Shift Register (RAM-based) MegaWizard® Plug-In Manager in the Quartus® II software.

Chapter 2, Getting Started guides you through each page of the MegaWizard Plug-In Manager with explanations for each of the options.

The ALTSHIFT_TAPS megafunction is implemented in the embedded memory block of all supported device families with simple dual-port RAM. You can select the RAM block type according to the capacity you require. The capacity that is represented by the width and the depth of the memory block depends on the `TAP_DISTANCE`, `NUMBER_OF_TAPS`, and `WIDTH` parameters of the ALTSHIFT_TAPS megafunction.

For the features and capacities of the typical memory block, refer to the chapter of your device handbook that contains information about TriMatrix embedded memory blocks.

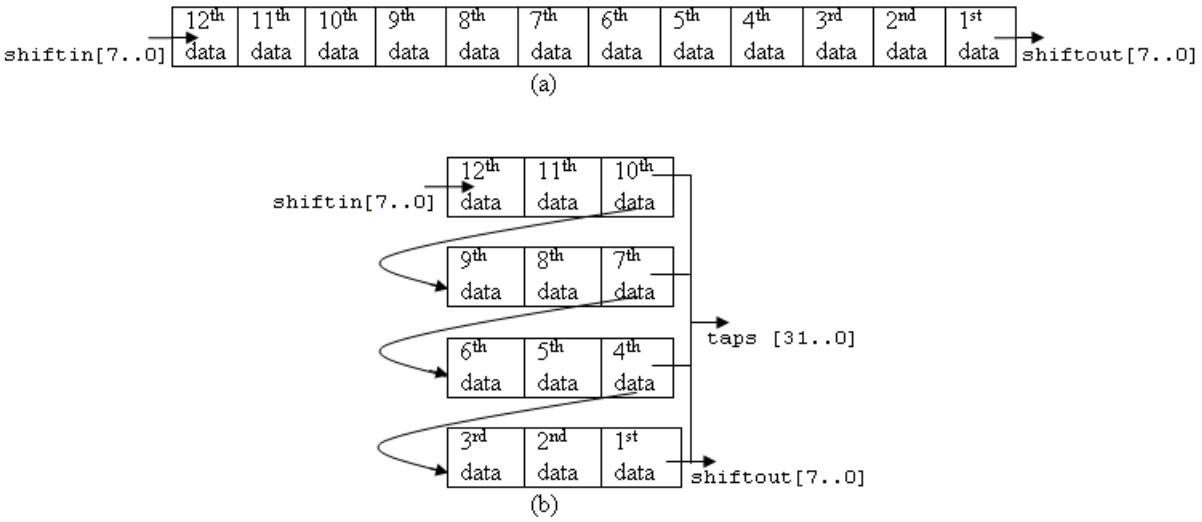
The ALTSHIFT_TAPS megafunction supports single-bit and multiple-bit data shifting at one clock cycle, depending on the width of the `shiftin` and `shiftout` ports. For example, if the `shiftin` and `shiftout` ports are single-bit data, only one bit is shifted per clock cycle. If the `shiftin` and `shiftout` ports are multiple-bit data, such as one-word data (8-bit), the whole word is shifted per clock cycle.

The megafunction also supports output taps at certain points in the shift register chain, but the tap points must be evenly spaced. The space between taps is set by the `TAP_DISTANCE` parameter in the MegaWizard Plug-in Manager.

For more information about setting the option for the distance between taps in the MegaWizard Plug-in Manager, refer to Chapter 2, Getting Started. For information about the TAP_DISTANCE parameter, refer to Chapter 3, Specifications.

Figure (a) in Figure 1–1 shows a traditional 12-word-depth shift register. Figure (b) shows how the data in the shift register chain are being tapped at even spaces (1st, 4th, 7th, and 10th) at the output taps of the ALTSHIFT_TAPS megafunction.

Figure 1–1. Tapping Data at Certain Points of the Shift Register Chain *(Note 1)*, *(2)*, *(3)*



Notes for Figure 1–1

(1) The ALTSHIFT_TAPS megafunction depicted here has TAP_DISTANCE = 3 and NUMBER_OF_TAPS = 4.

(2) The tapped data is output to taps[31..0]. Note that taps[31..0] is a 32-bit output because it taps four words at one time. The first word from the MSB of the taps (taps[31..24]) represents the first data and is followed by the 4th data, 7th data, and 10th data.

(3) The shiftout[7..0] word is equivalent to taps[31..24].

System Requirements

The instructions in this section require the following software:

- The Quartus® II software version 8.0 or later
- For operating system support information, refer to www.altera.com/support/software/os_support/oss-index.html

MegaWizard Plug-In Manager Customization

The MegaWizard® Plug-In Manager creates or modifies design files that contain custom megafunction variations, which can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the ALTSHIFT_TAPS megafunction features in your design.

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

- On the Tools menu, click **MegaWizard Plug-In Manager**.
- In the Block Editor, on the Edit menu, click **Insert Symbol as Block**, or right-click in the Block Editor, point to **Insert**, and click **Symbol as Block**. In the Symbol window, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:

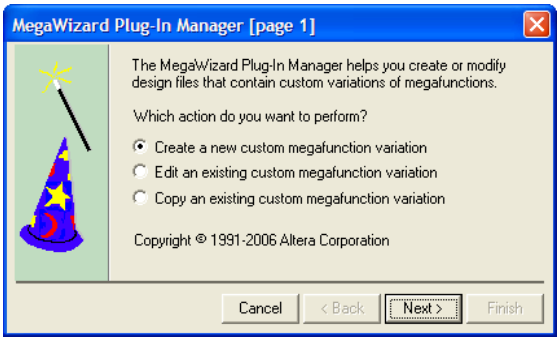
```
qmegawiz↵
```

MegaWizard Plug-In Manager Page Descriptions

This section provides descriptions for the options available on the individual pages of the Shift Register (RAM-based) MegaWizard plug-in.

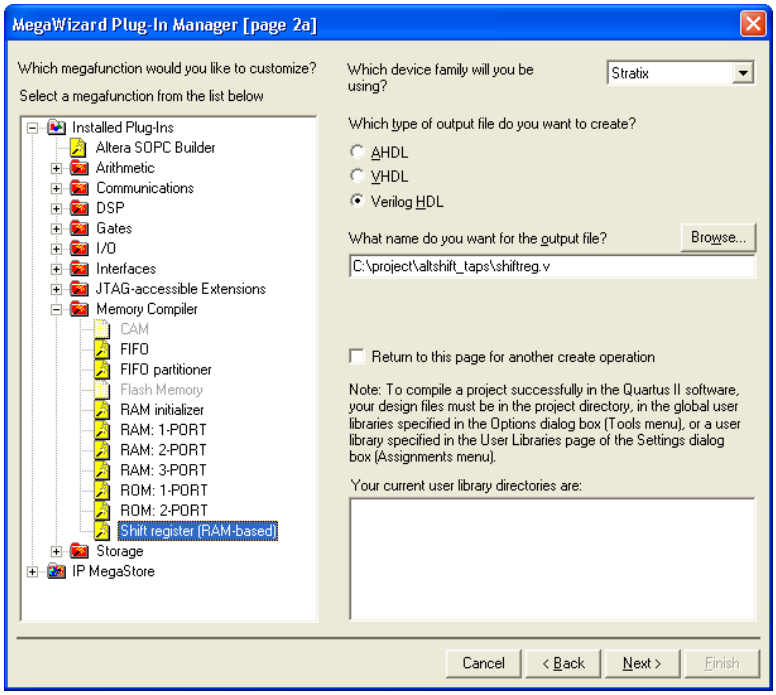
On page 1 of the MegaWizard Plug-In Manager, you can select **Create a new custom megafunction variation**, **Edit an existing custom megafunction variation**, or **Copy an existing custom megafunction variation** (Figure 2–1).

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



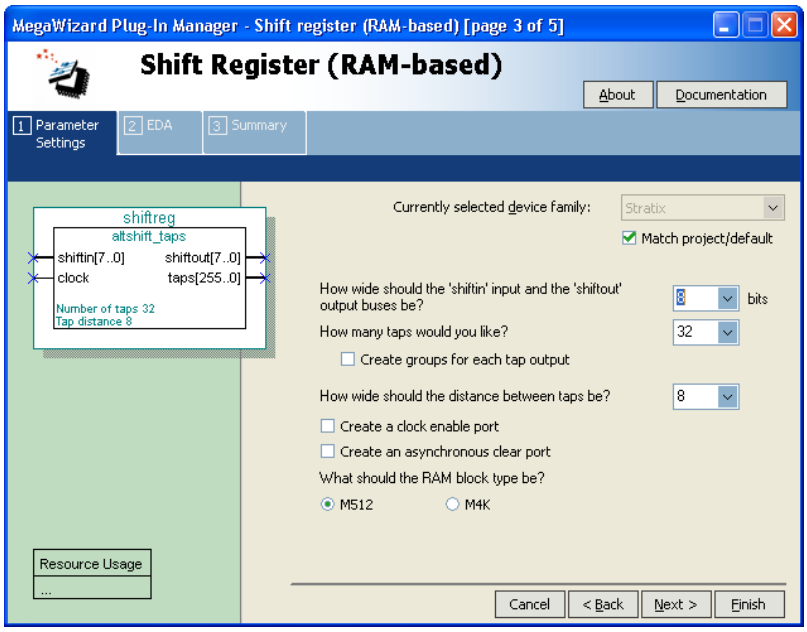
On page 2a of the MegaWizard Plug-In Manager, specify the megafunction, device family to use, type of output file to create, and the name of the output file (Figure 2–2). Choose AHDL (**.tdf**), VHDL (**.vhd**), or Verilog HDL (**.v**) as the output file type.

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



On page 3 of the MegaWizard Plug-In Manager, specify the width of the `shiftin` input bus and the `shiftout` output bus, specify the number of taps, create groups for each tap output, and specify the distance between the taps. You can also create a clock-enable port and an asynchronous-clear port, if applicable to your design, and select the type of RAM block type to use (Figure 2–3).

Figure 2–3. MegaWizard Plug-In Manager – Shift Register (RAM-based) [page 3 of 5]



Starting on page 3 of the Shift Register (RAM-based) MegaWizard Plug-In Manager, you can launch the *Shift Register (RAM-based) (ALTSHIFT_TAPS) Megafunction User Guide*, the ALTSHIFT_TAPS megafunction online help, or generate sample waveforms by clicking the **Documentation** button.

Table 2–1 shows the options available on page 3 of the Shift Register (RAM-based) MegaWizard Plug-In Manager. Use this table, along with the hardware descriptions, to determine the appropriate settings for the features.

Table 2–1. Shift Register (RAM-based) MegaWizard Plug-in Manager [page 3] Options (Part 1 of 2)

Configuration Setting	Description
How wide should the '`shiftin`' input and the '`shiftout`' output buses be?	Specify the width of the data input and output buses. This value is represented by the term *w* in the Shift Register Memory Configuration shown in Figure 2–4. *(1)*
How many taps would you like?	Specify the number of taps. This value is represented by the term *n* in the Shift Register Memory Configuration shown in Figure 2–4. *(2)*
Create groups for each tap output	Turn on this option to create separate groups for output data tapped from the register chain. *(3)*
How wide should the distance between taps be?	Specify the distance between taps. This value is represented by the term *m* in the Shift Register Memory Configuration shown in Figure 2–4. *(4)*

Table 2–1. Shift Register (RAM-based) MegaWizard Plug-in Manager [page 3] Options (Part 2 of 2)

Configuration Setting	Description
Create a clock enable port	Turn on this option to create an enable signal for register ports. The register ports are always enabled if this option is not turned on. *(5)*
Create an asynchronous clear port	Turn on this option to create an asynchronous clear signal. When asserted, the outputs of the shift register are immediately cleared.
What should the RAM block type be?	Choose the type of memory block that supports the feature, memory configuration, and capacity for your application. *(6)*

Notes for Table 2–1

(1) The widths of the `shiftin` input bus and `shiftout` output bus are identical, and they are not registered. However, the output data can be considered synchronous with the clock because the internal read address to the memory block is synchronous to the clock.

(2) The width of the output taps is the multiplication of *w* (width of input data) and *n* (number of taps). Also, the word from the MSB of the output taps is equivalent to the `shiftout` output bus.

(3) The combination of these groups represent the `taps[wn-1:0]` bus.

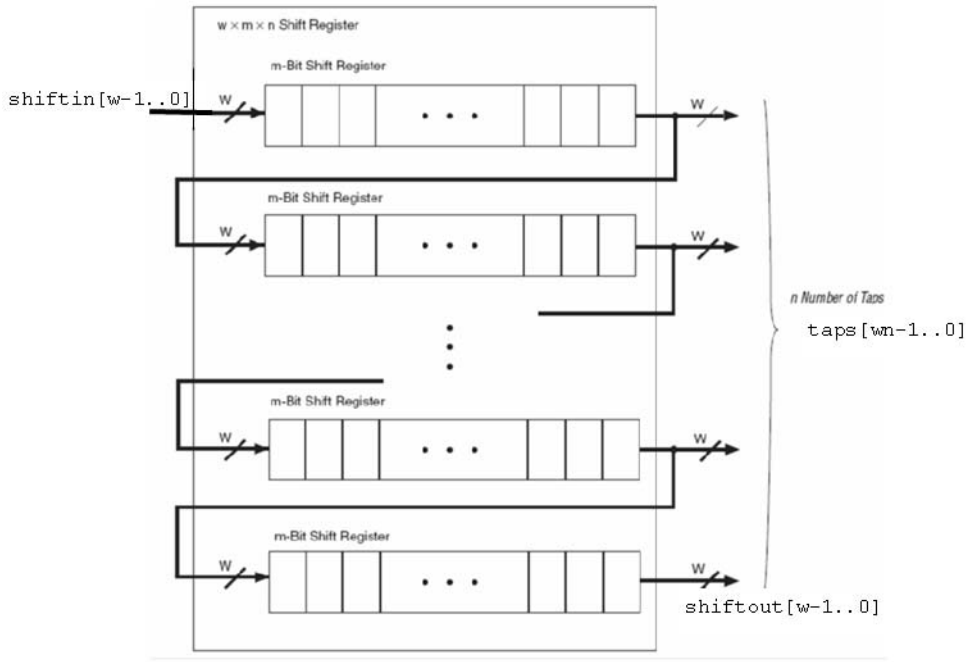
(4) The distance between taps, *m*, must be at least 3.

(5) The registered port is referred to as the internal register at the memory address ports. The `shiftin` and `shiftout` ports are not registered.

(6) For information about the chosen memory block type, refer to the TriMatrix Embedded Memory Block chapter of your target device handbook. You can also choose `AUTO` if you are not particular about the RAM block type used. With the `AUTO` option, the memory block type is determined by the Quartus II software synthesizer or Fitter at compile time. To determine the type of memory block used, check the Quartus II Fitter Report.

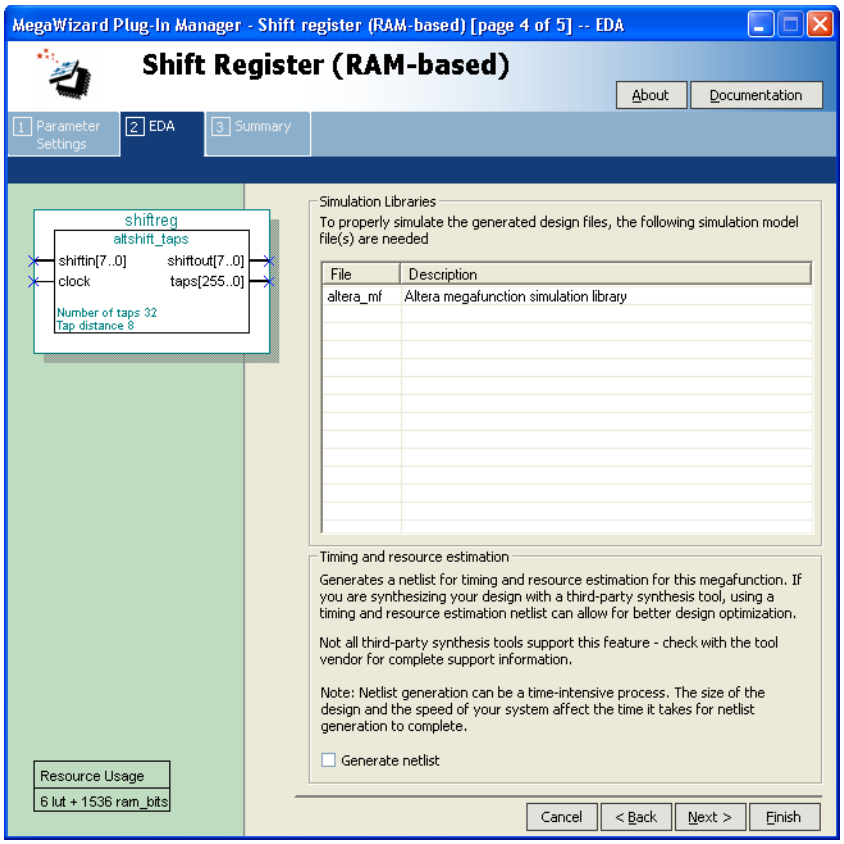
Figure 2–4 shows an example of a shift register chain.

Figure 2–4. Shift Register Chain Example



Page 4 of the Shift Register (RAM-based) MegaWizard Plug-In Manager lists the files needed to properly simulate the generated design files (Figure 2–5).

Figure 2–5. MegaWizard Plug-In Manager – Shift Register (RAM-based) [page 4 of 5]

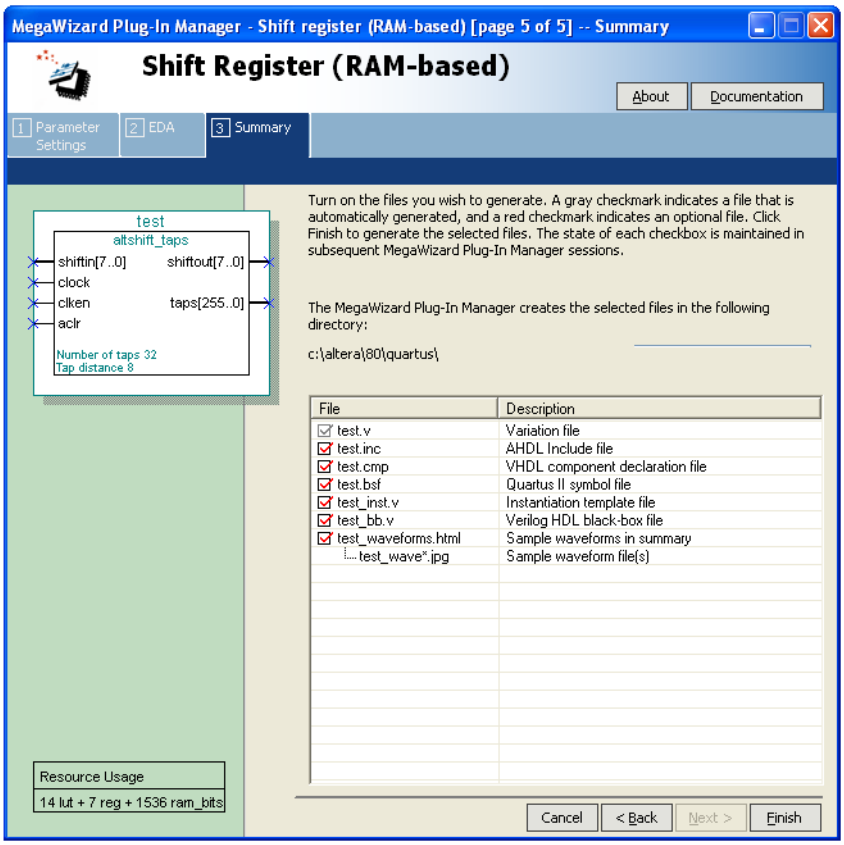


Page 5 of the Shift Register (RAM-based) MegaWizard Plug-In Manager displays the types of files to be generated. The Variation file, which is automatically generated, contains wrapper code in the language you specified on page 2a. On page 5 of the MegaWizard Plug-In Manager, specify the types of files to be generated. You can choose from the following types of files:

- AHDL Include file (<*function name*>**.inc**)
- VHDL component declaration file (<*function name*>**.cmp**)
- Quartus II symbol file (<*function name*>**.bsf**)
- Instantiation template file (<*function name*>**_inst.v**)
- Verilog HDL black-box file (<*function name*>**_bb.v**)

If you selected **Generate netlist** on page 4 of the MegaWizard Plug-In Manager, the file for that netlist is also available. A gray checkmark indicates a file that is automatically generated and a red checkmark indicates an optional file (Figure 2–6).

Figure 2–6. MegaWizard Plug-In Manager - Shift Register (RAM-based) [page 5 of 5]



Instantiating Megafunctions in HDL Code or Schematic Designs

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

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

- A sample instantiation template for the language of the variation file (**_inst.v**, **_inst.vhd**, or **inst.tdf**)
- Component Declaration File (**.cmp**) that can be used in VHDL Design Files
- AHDL Include File (**.inc**) that can be used in Text Design Files (**.tdf**)
- Quartus II Block Symbol File (**.bsf**) that can be used in schematic designs
- Verilog HDL module declaration file that can be used when instantiating the megafunction as a black box in a third-party synthesis tool (**_bb.v**)

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

Generating a Netlist for EDA Tool Use

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

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

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

Using the Port and Parameter Definitions

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

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

For a list of the megafunction ports and parameters, refer to Chapter 3, Specifications.

Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration are performed to build the structure of your design. You can locate your megafunction in the Project Navigator window by expanding the compilation hierarchy and locating 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 Simulator provides an easy-to-use, integrated solution for performing simulations. The following sections describe the simulation options.

Quartus II Software Simulator

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. A 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, add only signals that exist before synthesis. You can find these signals in the Node Finder by using any of the following Filter options: Registers: pre-synthesis, Design Entry, or Pins. 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, add only 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 the RTL signals change. Therefore, it may 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 HDL and VHDL synthesis attributes that direct analysis and 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.

For more information about these attributes, refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

EDA Simulator

The *Quartus II Handbook* chapters describe how to perform functional and gate-level timing simulations that include the megafunctions, with details about the files that are needed and the directories where the files are located.

Depending on which simulation tool you are using, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*.

Design Example: Shift Register with Taps

The objective of this design example is to implement and instantiate an ALTSHIFT_TAPS megafunction built through the Shift Register (RAM-Based) MegaWizard Plug-In Manager. This example uses a shift register with a data width, *w*, of 8 bits, a taps distance, *m*, of 3, and the number of taps, *n*, equal to 4. It also demonstrates how you can tap the data at specific points from the shift register chain.

Design Files

The example design files are available in the User Guides section on the Literature page of the Altera® website (www.altera.com).

Configuration Settings

In page 3 of the Shift Register (RAM-based) MegaWizard Plug-In Manager, select or verify the configuration settings shown in Table 2–2. Click **Next** to advance to the next page.

Table 2–2. Shift Register (RAM-Based) MegaWizard Plug-In Manager Configuration Settings

Configuration Setting	Value
Currently selected device family	Stratix III
How wide should the 'shiftin' input and the 'shiftout' output buses be?	8 bits
How many taps would you like?	4
Create groups for each tap output	Selected
How wide should the distance between taps be?	3
Create a clock enable port	Selected
Create an asynchronous clear port	Selected
What should the RAM block type be?	Auto

Functional Simulation in the ModelSim-Altera Simulator

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

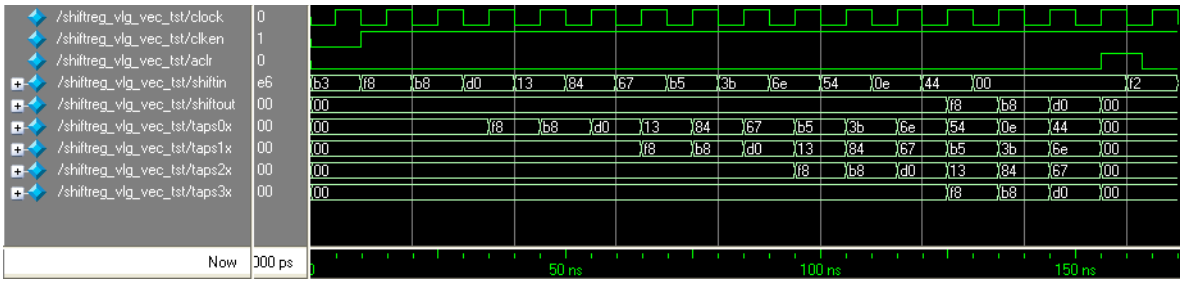
You should be familiar with the ModelSim-Altera software before trying the design example. If you are unfamiliar with the ModelSim-Altera software, refer to the support page for software products on the Altera website (www.altera.com). On the support page, there are links to such topics as installation, usage, and troubleshooting.

Set up and simulate the design in the ModelSim-Altera software by performing the following steps.

1. Unzip the **DE_ALTSHIFT_TAPS.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**.
6. On the Tools menu, click **Execute Macro**.
7. Select the **DE_ALTSHIFT_TAPS.do** file and click **Open**. The **DE_ALTSHIFT_TAPS.do** file is a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.

View the simulation results in the Wave window. Figure 2–7 shows the expected simulation results in the ModelSim-Altera software.

Figure 2–7. Simulation Waveform for Shift Register with Taps Design Example



Understanding the Simulation Results

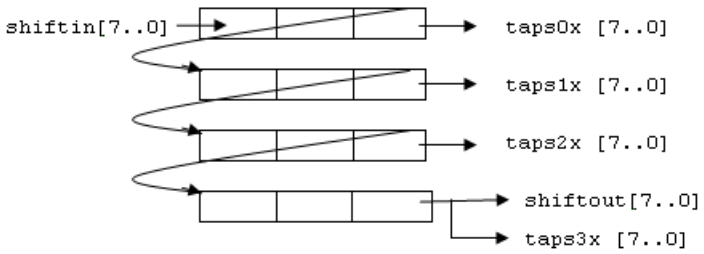
In this example, you configured the shift register to have the following properties:

- 8-bit data width
- Distance between taps (taps length) equals to 3
- Number of taps equals to 4
- Created groups for each tap output
- Created a clock-enable signal and an asynchronous-clear signal

This example shows how you can tap the 1st-4th-7th-10th data words simultaneously (followed by the 2nd-5th-8th-11th and 3rd-6th-9th-12th) when all 12 words of data are shifted into the shift register.

Figure 2–8 shows the shift register chain that is analogous to the configuration you set in the ALTSHIFT_TAPS megafunction in this example.

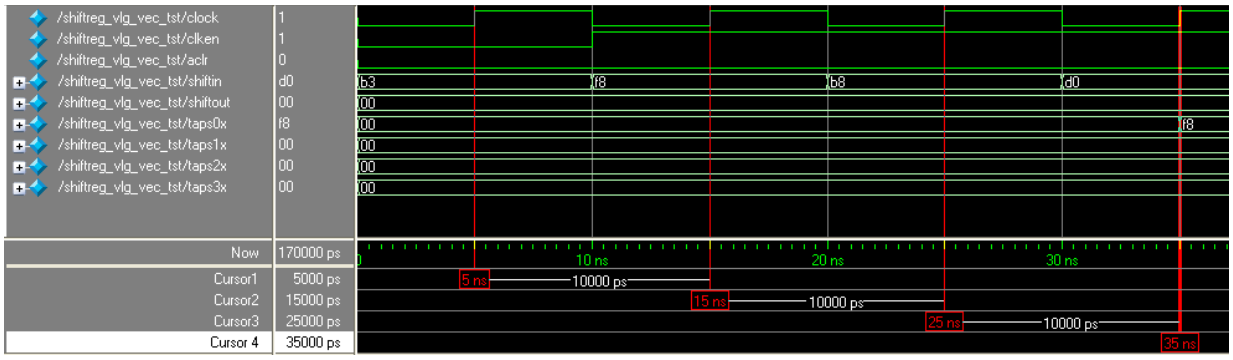
Figure 2–8. Shift Register Chain Analogy to the Configured ALTSHIFT_TAPS Megafunction



The next section uses this shift register chain to explain the shifting operation and the output operation of the ALTSHIFT_TAPS megafunction.

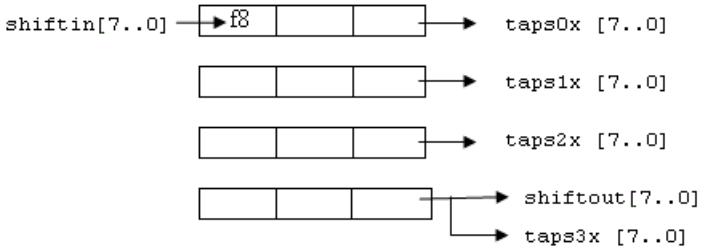
Figure 2–9 shows the first three data words written into the shift register chain, shifted in the register chain, and the first data shown at the `taps0x` output.

Figure 2–9. First Three Data Written and Shifted in the Shift Register



At 5 ns, the `clken` signal is low and therefore no operation is executed. You can consider 15 ns to be the first rising clock edge, as this is when the operation begins. The first data `F8` is shifted into the shift register as shown in Figure 2–10. All outputs show `00` because no data is being shifted to any of the outputs.

Figure 2–10. Content of the Shift Register Chain at 15 ns



At 25 ns and 35 ns, the second data `B8` and the third data `D0` are shifted into the shift register, respectively.

The existing data in the shift register chain are shifted right before the shift-in of new data.

Figure 2–11 shows the content in the shift register chain at 35 ns. All of the outputs show `00` except `taps0x`, which shows the first data, `F8`.

None of the input and output data ports are registered. Only the address ports of the memory block within the shift register are registered. Therefore, when the data are shifted to any of the output ports, the data are shown immediately at the respective output ports.

Figure 2–11. Content of the Shift Register Chain at 35 ns

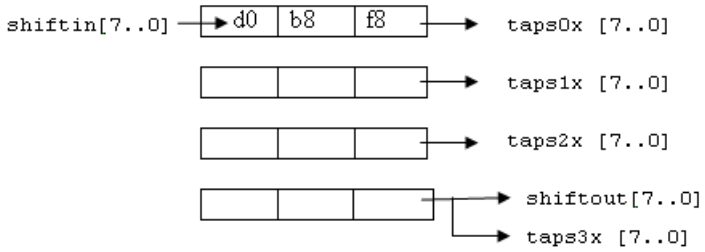
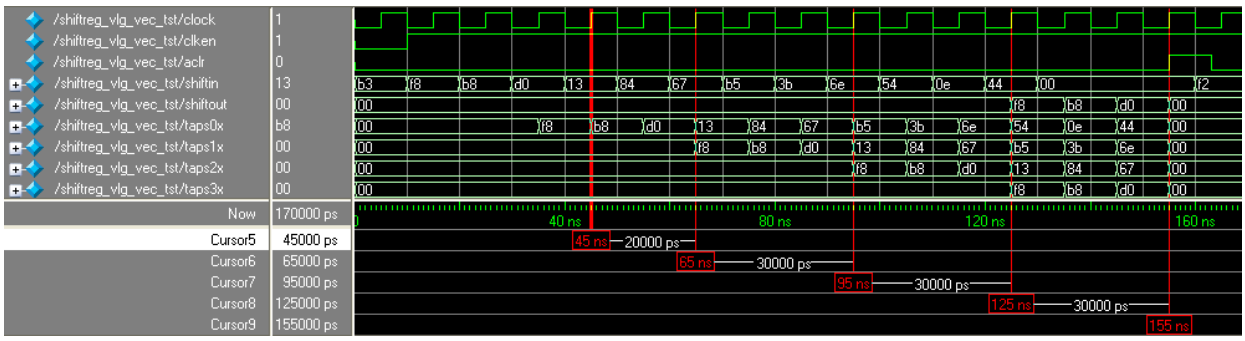


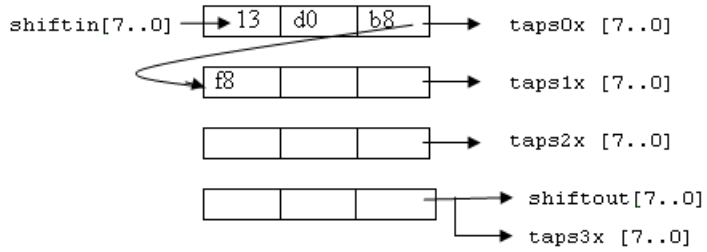
Figure 2–12 shows the data-shifting and output-tapping from the shift register chain at evenly-spaced intervals.

Figure 2–12. Data-Shifting and Output-Tapping



At 45 ns, the first data `F8` is shifted to the next row of taps and the second data `B8` is shifted to `taps0x`, as shown in Figure 2–13. Other output ports continue to show `00`. Also, at the same rising clock edge, the new data `13` is shifted into the shift register.

Figure 2–13. Content of the Shift Register Chain at 45 ns

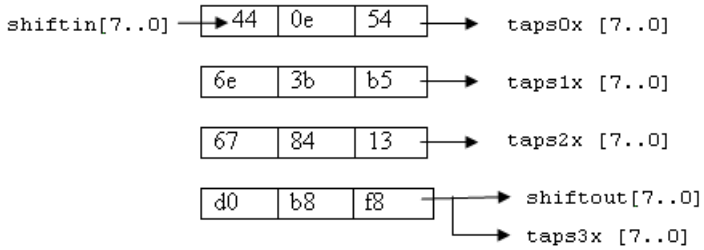


At 65 ns, the first data F8, and the fourth data 13 are shifted to taps1x and taps0x, respectively. At 95 ns, the first data F8, the fourth data 13, and the seventh data B5 are shifted to taps2x, taps1x, and taps0x, respectively. Finally, at 125 ns, all twelve data words are shifted into the shift register. You can then start to tap the 1st-4th-7th-10th data words simultaneously, from taps3x, taps2x, taps1x, and taps0x, respectively.

The shiftout output port is equivalent to taps3x and both ports generate the same output data.

At the following rising clock edge, you can tap the 2nd-5th-8th-11th data words, followed by the 3rd-6th-9th-12th data words at the next rising edge. Figure 2–14 shows the contents for the shift register chain when all twelve words are being shifted into the shift register.

Figure 2–14. Content of the Shift Register Chain at 125 ns



After you have tapped out all the data at 155 ns, you can assert the aclr signal to immediately clear all the data at the output ports and the contents of the shift register. You can then start to shift in another twelve words of data.

This design example shows you how the shifting and tapping operation works. It is not meant to show a specific application usage. You can use the tapping feature with additional logic to suit your needs.

Conclusion

The shift register is widely used in digital signal processing (DSP) applications. Compared to traditional shift registers that are implemented with standard flip-flops, the ALTSHIFT_TAPS megafunction is more suitable for DSP applications, because the megafunction is implemented using the embedded memory block, which saves logic cells and routing resources, and provides a bigger memory capacity.

Also, the ALTSHIFT_TAPS megafunction is equipped with taps capabilities that allow you to tap the data at certain fixed points. The selectable input data width, the length of the taps, and the number of taps provide a flexible configuration of the shift register you require.

3. Specifications

This chapter describes the prototypes, declarations, ports, and parameters of the ALTSHIFT_TAPS megafunction. You can use the ports and parameters to customize the ALTSHIFT_TAPS megafunction according to your application.

Verilog HDL Prototype for the ALTSHIFT_TAPS Megafunction

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

```
module    altshift_taps
#(        parameter         intended_device_family = "unused",
          parameter         number of taps = 1,
          parameter         power_up_state = "CLEARED",
          parameter         taps_distance = 1,
          parameter         width = 1,
          parameter         lpm_type = "altshift_taps",
          parameter         lpm_hint = "unused")
(         input wire        aclr,
          input wire        clken,
          input wire        clock,
          input wire        [width-1:0]    shiftin,
          output wire       [width-1:0]    shiftout,
          output wire       [width*number_of_taps-1:0]    taps)/*synthesis syn_black_box=1 */;
endmodule \\altshift_taps
```

VHDL Component Declaration for the ALTSHIFT_TAPS Megafunction

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

```
component altshift_taps
        generic (
                intended_device_family :    string := "unused";
                number_of_taps :       natural;
                power_up_state :       string := "CLEARED";
                tap_distance   :       natural;
                width  :        natural;
                lpm_hint       :       string := "UNUSED";
                lpm_type       :       string := "altshift_taps"
        );
        port(
                aclr   :        in std_logic := '0';
                clken  :        in std_logic := '1';
                clock  :        in std_logic;
                shiftin:        in std_logic_vector(width-1 downto 0);
                shiftout       :       out std_logic_vector(width-1 downto 0);
                taps   :       out std_logic_vector(width*number_of_taps-1 downto 0)
        );
end component;
```

VHDL Library-Use Declaration

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

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

Ports and Parameters for the ALTSHIFT_TAPS Megafunction

Figure 3–1 below shows the ports and parameters for the ALTSHIFT_TAPS megafunction.

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

Figure 3–1. Shift Register (RAM-based) Ports and Parameters

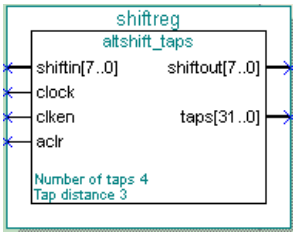


Table 3–1 shows the input ports of the ALTSHIFT_TAPS megafunction.

Table 3–1. Shift Register (RAM-based) MegaWizard Plug-In Manager Input Ports

Name	Required	Description
shiftin[]	Yes	Data input to the shifter. Input port WIDTH bits wide.
clock	Yes	Positive-edge triggered clock.
clken	No	Clock enable for the clock port. clken defaults to V_{CC}.
aclr	No	Asynchronously clears the contents of the shift register chain. The shiftout outputs are cleared immediately upon the assertion of the aclr signal.

Table 3–2 shows the output ports of the ALTSHIFT_TAPS megafunction.

Table 3–2. Shift Register (RAM-based) MegaWizard Plug-In Manager Output Ports

Name	Required	Description
shiftout[]	Yes	Output from the end of the shift register. Output port WIDTH bits wide.
taps[]	Yes	Output from the regularly spaced taps along the shift register. Output port WIDTH * NUMBER_OF_TAPS wide. This port is an aggregate of all the regularly spaced taps (each WIDTH bits) along the shift register.

Table 3–3 shows the ALTSHIFT_TAPS megafunction parameters.

Table 3–3. Shift Register (RAM-based) MegaWizard Plug_In Parameters

Name	Type	Required	Description	
NUMBER_OF_TAPS	Integer	Yes	Specifies the number of regularly spaced taps along the shift register.	
TAP_DISTANCE	Integer	Yes	Specifies the distance between the regularly spaced taps in clock cycles. This number translates to the number of RAM words that will be used. TAP_DISTANCE must be at least 3.	
WIDTH	Integer	Yes	Specifies the width of the input pattern.	
POWER_UP_STATE	String	No	Specifies the shift register contents at power-up. Values are CLEARED and DONT_CARE. If omitted, the default is CLEARED.	
			Value	**Description**
			CLEARED	Zero content. For Stratix and Stratix II device families, you must use M512 or M4K RAM blocks.
			DONT_CARE	Unknown contents. M-RAM blocks can be used with this setting.

This chapter provides additional information about the document and Altera.

Document Revision History

The following table shows the revision history for this document.

Date	Document Version	Changes Made
November 2010	2.1	■ Updated ports and parameters ■ Added prototype and component declarations
July 2008	2.0	■ Updated the list of device families supported by this megafunction ■ Created a new design example with explanations showing the features and behaviors of the megafunction ■ Added the description for the new input pin, `aclr` ■ Reorganized the whole document
March 2007	1.2	Added Cyclone® III support
December 2006	1.1	Added Stratix® III support
September 2006	1.0	Initial release

How to Contact Altera

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

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

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

Typographic Conventions

The following table shows the typographic conventions this document uses.

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