

altshift_taps Megafunction

User Guide



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

Document Version: 1.0
Document Date: September 2004

Copyright © 2004 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001

UG-MF9204-1.0

About this User Guide	v
Revision History	v
How to Contact Altera	v
Typographic Conventions	vi
Chapter 1. About this Megafunction	
Device Family Support	1-1
Introduction	1-1
Features	1-2
General Description	1-2
Common Applications	1-2
Resource Utilization & Performance	1-2
Chapter 2. Getting Started	
System Requirements	2-1
Mega Wizard Plug-In Manager Customization	2-1
Using the MegaWizard Plug-In Manager	2-1
Inferring Megafunctions from HDL Code	2-4
Instantiating Megafunctions in HDL Code	2-4
Identifying a Megafunction after Compilation	2-4
Simulation	2-4
Quartus II Simulation	2-4
EDA Simulation	2-5
SignalTap II Embedded Logic Analyzer	2-5
Design Example: Shift Register with Taps	2-6
Design Files	2-6
Example	2-6
Generate a Shift Register with Taps	2-6
Implement the Shift Register with Taps	2-9
Functional Results - Simulate the Shift Register with Taps	2-10
Conclusion	2-11
Chapter 3. Specifications	
Ports & Parameters	3-1



About this User Guide

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

Chapter	Date	Document Version	Changes Made
1	September 2004	1.0	• Initial release
2	September 2004	1.0	• Initial release
3	September 2004	1.0	• Initial release

How to Contact Altera

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








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

Note to table:

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

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: f_{MAX} , lqdesigns directory, d: drive, chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pdf file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets are used in a list of items when the sequence of the items is not important.
	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Device Family Support

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

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

Table 1–1 shows the level of support offered by the `altshift_taps` megafunction to each Altera device family.

Device Family	Support
Stratix® II	Full
Stratix	Full
Stratix GX	Full
HardCopy Stratix	Full
Cyclone™	Full
Mercury™	Full
Excalibur™	Full
ACEX® 1K	Full
APEX™ II	Full
APEX 20KE & APEX 20KC	Full
APEX 20K	Full
HardCopy™ APEX	Full
FLEX 10K®	Full
Other device families	No support

Introduction

As design complexities increase, 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.

Features

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

- Ability to specify the width of the `shiftin` and `shiftout` ports
- Ability to specify the number of taps
- Ability to specify the distance between taps

General Description

The `altshift_taps` megafunction is one of the storage megafunctions provided in the Quartus® II MegaWizard® Plug-In Manager. 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. The tap points must be evenly spaced. You can attach additional logic to the tap outputs for further applications.

Common Applications

A shift register with taps is useful for applications such as a Linear Feedback Shift Register (LFSR) and Finite Impulse Response (FIR) filters. LFSR is a shift register whose outputs are combined in exclusive-or (XOR) configuration to form a feedback mechanism. LFSR is used to generate pseudo-random bit sequences (PRBS) found in many different applications, including cryptography, spread spectrum, and bit error rate (BER) measurements.

Resource Utilization & Performance

The `altshift_taps` megafunction is implemented in all supported device families with simple dual-port RAM. The `altshift_taps` megafunction is implemented in the device memory blocks. The width and depth of the memory block depends on the parameters (`TAP_DISTANCE`, `NUMBER_OF_TAPS`, and `WIDTH`) of the `altshift_taps` megafunction.

For Stratix II, Stratix, and Stratix GX devices, the `altshift_taps` megafunction is implemented in either M512 or M4K memory blocks. For Cyclone II and Cyclone devices, the `altshift_taps` megafunction is implemented in M4K memory blocks.

Table 1–2 summarizes the resource usage for an `altshift_taps` function that is used to implement a shift register with taps.

Device Family	Width	Number of Taps	Tap Distance	Logic Usage
Stratix II, Stratix, Stratix GX, Cyclone II, and Cyclone	8 bits	8	4	130 memory blocks
Stratix II, Stratix, Stratix GX, Cyclone II, and Cyclone	8 bits	32	8	1530 memory blocks



For more information on using memory blocks in Stratix II devices, see “TriMatrix Embedded Memory Blocks in Stratix II Devices” in Volume 2 of the *Stratix II Device Handbook*.

For more information on using memory blocks in Stratix and Stratix GX devices, refer to “TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices” in Volume 2 of the *Stratix Device Handbook*.

For more information on using memory blocks in Cyclone II devices, refer to the “Cyclone II Memory Blocks” chapter in the *Cyclone II Device Handbook*.

For more information on using memory blocks in Cyclone devices, refer to the “On-Chip Memory Implementations Using Cyclone Memory Blocks” chapter in Volume 1 of the *Cyclone Device Handbook*.

System Requirements

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

- A PC running either Windows NT/2000/XP, Red Hat Linux 7.3 or 8.0, Red Hat Linux Enterprise 3, *or* an HP workstation running the HP-UX version 11.0 operating system, *or* a Sun workstation running the Solaris 7 or 8 operating system
- Quartus® II software version 4.1 or later

Mega Wizard Plug-In Manager Customization

You can use the MegaWizard® Plug-In Manager to set the `altshift_taps` megafunction features for each multiplier in the design.

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

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

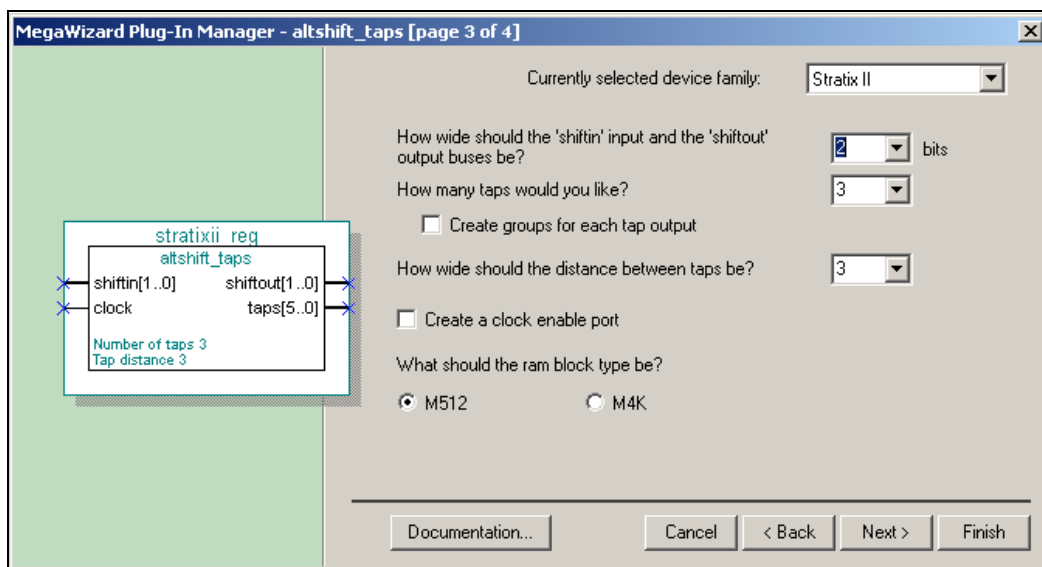
Using the MegaWizard Plug-In Manager

This section provides descriptions for the options available in the `altshift_taps` megafunction.

Page 3 of the `altshift_taps` megafunction wizard is where you determine the width of the 'shiftin' input bus and the 'shiftout' output bus, set the number of taps, create groups for each tap output, and set the width of the distance between the taps. The minimum width is 3 bits. You can also create a clock enable, if applicable to your design and select the type of memory block to use, M512 or M4K. Cyclone II and Cyclone devices are used in M4K memory blocks only.

Figure 2-1 below shows page 3 of the `altshift_taps` megafunction wizard.

Figure 2–1. MegaWizard Plug-In Manager - altshift_taps [page 3 of 4]



Starting on Page 3 of the altshift_taps megafunction wizard, you can launch the Quartus II Help for the altshift_taps megafunction by selecting the **Quartus II Megafunction Reference** option from the **Documentation** button.

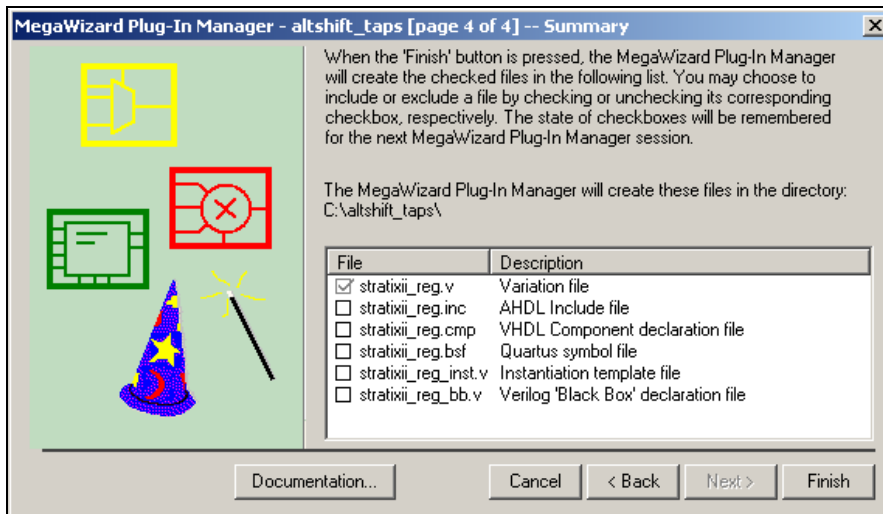
Table 2–1 below shows the features and settings of the altshift_taps megafunction. Use this table, along with the hardware descriptions for the features, to determine the appropriate settings.

Function	Description
How wide should the 'shiftin' input bus and the 'shiftout' output bus be?	Specify the width of the data input and output buses.
How many taps would you like to use? Do you want to create groups of taps? What is the distance between the taps?	Specify which options you want to perform.
Do you want to create a clock enable port?	Turn on this option to set a clock enable port.
What type of ram block do you want?	Choose the type of RAM block based on the device type.

Page 4 of the `altshift_taps` megafunction wizard lets you choose the types of files to be generated by the wizard. You can choose from the HDL wrapper file (`<function name>.v` | `.vhd` | `.tdf`), Block Symbol file (`.bsf`), Instantiation template file (`<function name>_inst.v` | `.vhd` | `.tdf`), or Verilog Black Box declaration file (`<function name>_bb.v`).

Figure 2-2 shows page 4 of the `altshift_taps` megafunction wizard.

Figure 2-2. MegaWizard Plug-In Manager - `altshift_taps` [page 4 of 4] -- Summary



Inferring Megafunctions from HDL Code

Synthesis tools, including the Quartus II integrated synthesis, recognize certain types of HDL code and automatically infer the appropriate megafunction when a megafunction will provide optimal results. That is, the Quartus II software uses the Altera megafunction code when compiling your design, even though you did not specifically instantiate the megafunction. The Quartus II software infers megafunctions because they are optimized for Altera devices, so the area and/or performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features—such as memory, DSP blocks, and shift registers—that generally provide improved performance compared with basic logic elements.



Refer to the “Recommended HDL Coding Styles” chapter in Volume 1 of the *Quartus II Handbook* for specific information about your particular megafunction.

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 3rd party synthesis support chapters in the Synthesis section 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. You can locate your megafunction in the Project Navigator window by expanding the compilation hierarchy and locating the megafunction by its name.

Similarly, 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. A functional simulation in the Quartus II program 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 add only 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 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 your RTL signals will change. Therefore, it might be difficult to find signals from your megafunction instantiation in the **Post-Compilation** filter. However, if you want to preserve the names of your signals during the synthesis and place and route stages, you must use the synthesis attributes `keep` or `preserve`. These are Verilog and VHDL synthesis attributes that direct analysis and synthesis to keep a particular wire, register, or node intact. You can use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation. More information on these attributes is available in the “Integrated Synthesis” chapter of the *Quartus II Handbook*.

EDA Simulation

Depending on the 3rd party simulation tool you are using, refer to the appropriate chapter in the Simulation section in Volume 3 of the *Quartus II Handbook*. The *Quartus II Handbook* chapters show you how to perform functional and gate-level timing simulations that include the megafunctions, with details on the files that are needed and the directories where those files are located.

You can generate a clear-box model for this megafunction that can be synthesized with 3rd party EDA synthesis tools.

SignalTap II Embedded Logic Analyzer

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

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



For more information on using the SignalTap II embedded logic analyzer, refer to the “*Design Debugging Using the SignalTap II Embedded Logic Analyzer*” chapter of the *Quartus II Handbook*.

Design Example: Shift Register with Taps

This section presents a design example that uses the `altshift_taps` megafunction to generate a shift register with taps. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into your overall project.

Design Files

The example design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site at the following URL: www.altera.com.

Example

In this example, you perform the following activities:

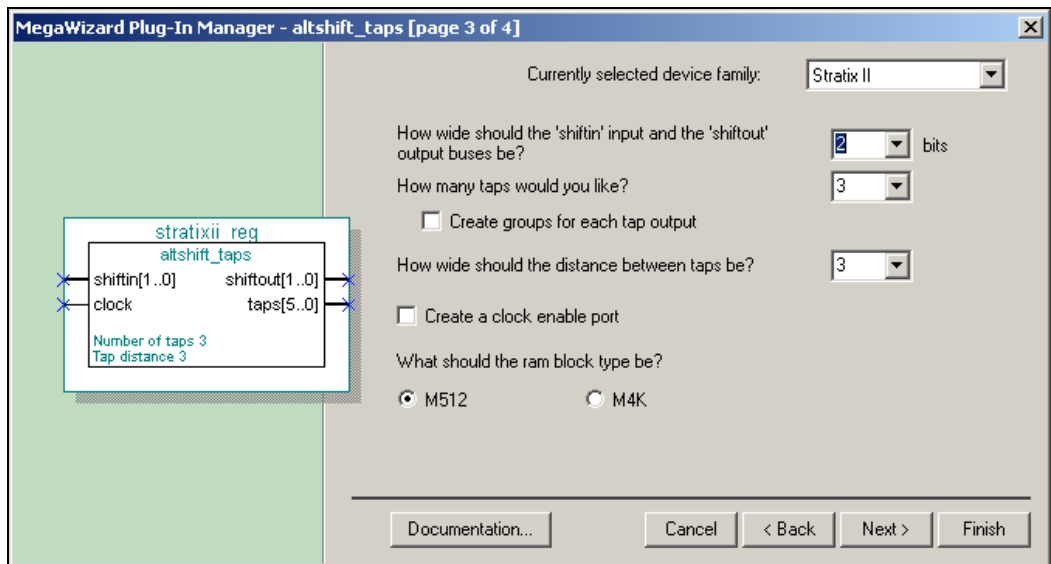
- Generate a shift register with taps using the `altshift_taps` megafunction in the MegaWizard Plug-In Manager
- Implement the shift register design in an Altera device by assigning the EP2S30F484C3 device and compiling the project
- Simulate the shift register design

Generate a Shift Register with Taps

1. Open the project file `\altshift_taps\stratixii_reg.qpf`.
2. Open the top-level file `stratixii_reg.bdf`. This is an incomplete file that you will complete as part of this example.
3. Double-click on a blank area in the block design file (`.bdf`).
4. Choose **MegaWizard Plug-In Manager** in the **Symbol** window.
5. In the window that appears, turn on **Create a new custom megafunction variation** in the **What action do you want to perform?** section.
6. Click **Next**.
7. On page 2a, expand the Arithmetic folder and select the `altshift_taps` megafunction.

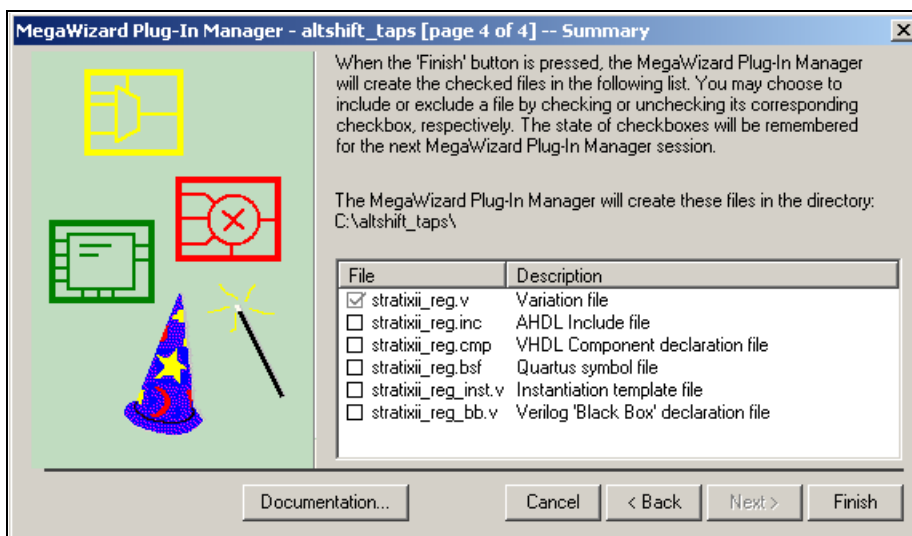
8. Select Stratix II in the **Which device family will you be using?** list.
9. Set the output file name to `<project directory>\stratixii_reg`.
10. Click **Next**.
11. On page 3 of the `altshift_taps` megafunction wizard, select 2 bits for the **How wide should the 'shiftin' input and 'shiftout' output buses be?** section (Figure 2–3).
12. Select 3 in the **How many taps would you like?** section.
13. Set 3 in the **How wide should the distance between taps be?** section.
14. Select M512 under the **What should the ram block type be?** section.
15. Click **Next**.

Figure 2–3. MegaWizard Plug-In Manager - altshift_taps [page 3 of 4]



- On Page 4, turn on the option for a Verilog Variation file (.v) so it is created for the project (Figure 2-4).
- Click **Finish**.
- Move your mouse to place the shift register with taps symbol in between the input and output ports of the `stratixii_reg.bdf` file. Click the left mouse button to place the symbol. You have now completed the design file.
- Select **Save (File menu)** to save the design.

Figure 2-4. MegaWizard Plug-In Manager - altshift_taps [page 4 of 4]-- Summary



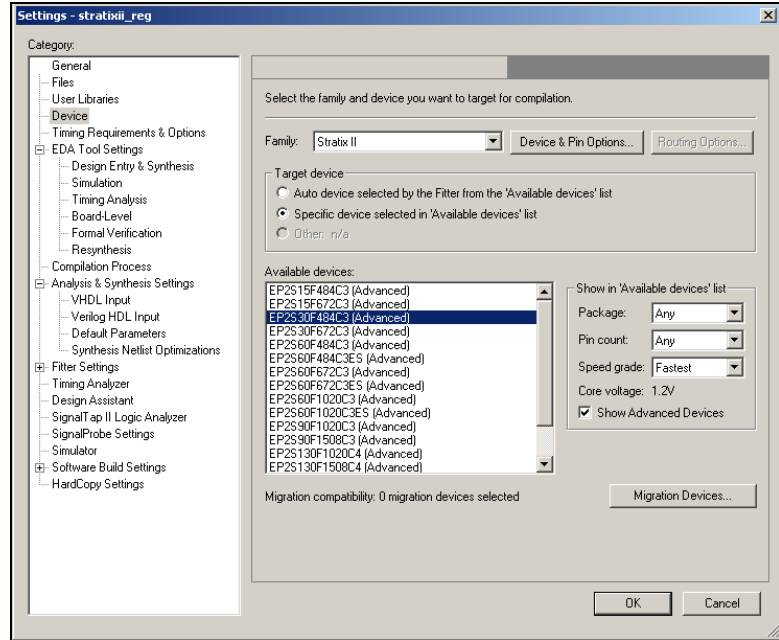
Implement the Shift Register with Taps

Next, assign the EP2S30F484C3 device to the project and compile the project.

- Select **Settings (Assignments menu)** to open the **Settings** dialog box.
- Click the **Devices** category. Stratix II should already be selected in the Family field. If Stratix II is not selected, select it.

3. Turn on **Specific device selected in 'Available devices' list** under the **Target device** section (Figure 2-5).
4. Select **EP2S30F484C3** in the **Available devices** list.
5. Click **OK**.

Figure 2-5. Settings- stratixii_reg



6. Select **Start Compilation (Processing menu)** to compile the design.
7. Click **OK** when the **Full compilation was successful** message box appears.

Functional Results - Simulate the Shift Register with Taps

Finally, simulate the design to verify the results. Set up the Quartus II Simulator by performing the following steps:

1. Select **Generate Functional Simulation Netlist (Processing menu)**.
2. Click **OK**.


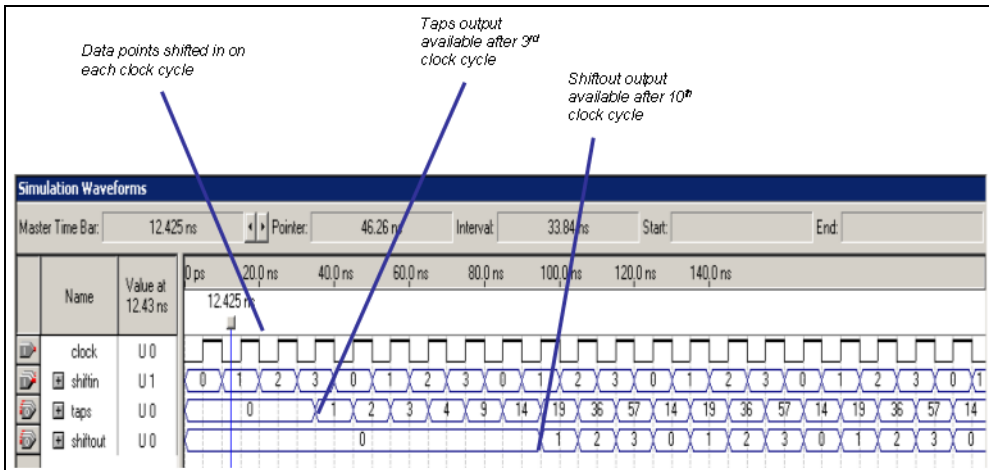
3. Select **Wizards > Simulator Setting Wizard** (**Assignments** menu).
4. In the **Category** section, highlight the **Simulator** category.
5. On Page 1, select **Functional** under **Simulation mode**.
6. Click **Next**.
7. On Page 2, turn on **Yes, use this file:** and specify the path to the **stratixii_reg.vwf** file. Leave all the other options in their default settings.
8. Click **Next**.
9. On Page 3, select **Run simulation until all vector stimuli have been used** for the **How do you want to determine the end time for the simulation?** section.
10. Click **Next** until you reach the **Summary** Page and then click **Finish**.
11. Click the **Start Simulation** (**Processing** menu), or press **Ctrl+I**, or click the  **Simulation** button to run a simulation.
12. Click **OK** when the **Simulation was successful** message box appears.
13. In the **Simulation Report** window, look at the simulation output waveform and verify the results. [Figure 2-6](#) shows the expected simulation results of shift register with taps function.

Figure 2–6. Simulation Waveforms



Conclusion

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

Ports & Parameters



Figure 3–1 below shows the ports and parameters for the `altshift_taps` megafunction. Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the `altshift_taps` megafunction parameters.

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

The parameter details are only relevant for users who by-pass 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. `altshift_taps` Port and Parameter Description Symbol



Name	Required	Description	Comment
<code>shiftin[]</code>	Yes	Data input to the shifter.	Input port WIDTH bits wide.
<code>clock</code>	Yes	Positive-edge triggered clock.	
<code>clken</code>	No	Clock enable for the clock port.	

Table 3–2. altshift_taps Megafunction Output Ports

Name	Required	Description	Comment
shiftout[]	No	Output from the end of the shift register.	Output port WIDTH bits wide.
taps[]	No	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. altshift_taps Megafunction Parameters

Name	Type	Required	Comment
NUMBER_OF_TAPS	Integer	No	Specifies the number of regularly spaced taps along the shift register.
TAP_DISTANCE	Integer	No	Specifies the distance between the regularly spaced taps in clock cycles. This number translates to the number of RAM words that will be used.
WIDTH	Integer	No	Specifies the width of the input pattern.