



Introduction to Megafunctions

User Guide



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

Copyright © 2011 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, and specific device designations are trademarks and/or service marks of Altera Corporation in the U.S. and other countries. All other words and logos identified as trademarks and/or service marks are the property of Altera Corporation or their respective owners. 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. 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.



Megafunctions are vendor-specific intellectual property (IP) blocks that are parameterizable and optimized for Altera® device architectures. Altera provides a library of megafunctions, including the Library of Parameterized Modules (LPM) functions and other parameterized functions, which offer more efficient logic synthesis and device implementation.

You can generate megafunctions using the following methods:

- [Generating Megafunctions Using the MegaWizard Plug-In Manager](#)
- [Generating Megafunctions Using the Command-Line Tools](#)

Altera recommends that you configure megafunctions using the MegaWizard™ Plug-In Manager.

Megafunction Types and Features

Table 1–1 lists the features and types of Altera-provided megafunctions and LPM functions. Some megafunctions are based on the Hardware Component Description File (`_hw.tcl`) infrastructure while others are non-`_hw.tcl`-based. A `_hw.tcl`-based megafunction uses a Tcl script that defines everything the MegaWizard Plug-In Manager and Qsys require about the name and location of component design files, including files for simulation and constraint files.

 For more information about the megafunctions in Table 1–1, refer to the respective megafunction user guides on the [Documentation: IP and Megafunctions](#) page of the Altera website.

Table 1–1. Megafunction Types and Features

Megafunction Name	Features	Infrastructure
Arithmetic Functions		
LPM_ABS	Absolute value	non- <code>_hw.tcl</code> -based
LPM_ADD_SUB	Adder/Subtractor	non- <code>_hw.tcl</code> -based
LPM_COMPARE	Comparator	non- <code>_hw.tcl</code> -based
LPM_COUNTER	Counter	non- <code>_hw.tcl</code> -based
LPM_DIVIDE	Divider	non- <code>_hw.tcl</code> -based
LPM_MULT	Multiplier	non- <code>_hw.tcl</code> -based
ALTACCUMULATE	Accumulator	non- <code>_hw.tcl</code> -based
ALTECC	ECC Encoder/Decoder	non- <code>_hw.tcl</code> -based
ALTMEMMULT	Memory-based Constant Coefficient Multiplier	non- <code>_hw.tcl</code> -based
ALTMULT_ACCUM	Multiply-Accumulator	non- <code>_hw.tcl</code> -based
ALTMULT_ADD	Multiply-Adder	non- <code>_hw.tcl</code> -based
ALTMULT_COMPLEX	Complex Multiplier	non- <code>_hw.tcl</code> -based
ALTSQRT	Integer Square-Root	non- <code>_hw.tcl</code> -based

Table 1–1. Megafunction Types and Features

Megafunction Name	Features	Infrastructure
PARALLEL_ADD	Parallel Adder	non- _hw.tcl -based
Floating-Point		
ALTFP_ADD_SUB	Adder/Subtractor	non- _hw.tcl -based
ALTFP_DIV	Divider	non- _hw.tcl -based
ALTFP_MULT	Multiplier	non- _hw.tcl -based
ALT_SQRT	Square Root	non- _hw.tcl -based
ALTFP_EXP	Exponential	non- _hw.tcl -based
ALTFP_INV	Inverse	non- _hw.tcl -based
ALTFP_INV_SQRT	Inverse Square Root	non- _hw.tcl -based
ALTFP_LOG	Natural Logarithm	non- _hw.tcl -based
ALTFP_ATAN	Arctangent	non- _hw.tcl -based
ALTFP_SINCOS	Trigonometric Sine/Cosine	non- _hw.tcl -based
ALTFP_ABS	Absolute value	non- _hw.tcl -based
ALTFP_COMPARE	Comparator	non- _hw.tcl -based
Gates functions		
LPM_GLSHIFT	Combinatorial logic shifter	non- _hw.tcl -based
LPM_CONSTANT	Constant generator	non- _hw.tcl -based
LPM_DECODE	Decoder	non- _hw.tcl -based
LPM_MUX	Multiplexer	non- _hw.tcl -based
I/O functions		
ALTLVDS	LVDS transmitter and receiver	non- _hw.tcl -based
ALTPLL	Phase-Locked Loop (PLL)	non- _hw.tcl -based
ALTPLL_RECONFIG	PLL reconfiguration	non- _hw.tcl -based
ALTERA_PLL	Phase-Locked Loop for Stratix® V devices only	_hw.tcl -based
ALTREMOTE_UPDATE	Remote Update	non- _hw.tcl -based
ALTGXB	Gigabit transceiver block (GXB)	non- _hw.tcl -based
ALTOCT	On-Chip Termination	non- _hw.tcl -based
ALTCLKCTRL	Clock Control	non- _hw.tcl -based
ALTDIO	Double Data Rate I/O	non- _hw.tcl -based
ALTIOBUF	I/O Buffer	non- _hw.tcl -based
ALTTEMP_SENSE	Temperature Sensor	non- _hw.tcl -based
ALTDLL	Delay Locked Loop (DLL)	non- _hw.tcl -based
ALTDQ and ALTDQS	DQ and DQS	non- _hw.tcl -based
ALTDQ_DQS2	DQ and DQS for Stratix V	_hw.tcl -based
ALTASMI_PARALLEL	Active Serial Memory Interface	non- _hw.tcl -based

Table 1-1. Megafunction Types and Features

Megafunction Name	Features	Infrastructure
Memory Compiler		
RAM and ROM	Internal memories	non- <code>_hw.tcl</code> -based
SCFIFO and DCFIFO	Single clock FIFO and Dual Clock FIFO	non- <code>_hw.tcl</code> -based
FIFO Partitioner	First-In First-Out Partitioner	non- <code>_hw.tcl</code> -based
LPM_SHIFTREG	Shift-register	non- <code>_hw.tcl</code> -based
ALTSHIFT_TAPS	RAM-Based Shift Register	non- <code>_hw.tcl</code> -based
ALT_UFM	User Flash Memory	non- <code>_hw.tcl</code> -based
ALTOTP	One-Time Programmable function	non- <code>_hw.tcl</code> -based
JTAG Accessible Extensions		
PFL	Parallel Flash Loader	non- <code>_hw.tcl</code> -based
SLD_VIRTUAL_JTAG	Virtual JTAG	non- <code>_hw.tcl</code> -based


Design Flow

The megafunction design flow involves the following steps:

1. Determine the type of device family variant that best suit your megafunction design requirements. For more information, refer to [“Selecting a Device” on page 2-2](#).
2. Decide whether to instantiate megafunctions or infer megafunctions from HDL code. Instantiation allows you to have better control over how the megafunctions are used while inference provides portable codes that you can use for various designs. You can parameterize and instantiate the megafunction using either the Quartus® II software parameter editor or command line.
 - When you set the instantiate and parameterize through the parameter editor, you are given an option to automatically generate either a wrapper file or a netlist file. For more information, refer to [“Instantiating Megafunctions” on page 2-4](#)

You can generate megafunctions in the Quartus II software through the following methods:

- Generate megafunction instances automatically using the MegaWizard Plug-In Manager. For more information, refer to [“Generating Megafunctions Using the MegaWizard Plug-In Manager” on page 2-1](#).

 For more information, refer to [Chapter 10: Recommended HDL Coding Styles](#) in volume 1 of the *Quartus II Handbook*.

- Create megafunction instances using the block editor.
 - For more information, refer to “Working with Blocks and Symbols in the Block Editor” section in the Quartus II Help.
- Instantiate the megafunctions directly into the HDL code using the ports and parameters definition. For more information, refer to “Instantiating Megafunctions Using the Port and Parameter Definition” on page 2-8.
- Create megafunctions using Qsys.
 - For more information about creating megafunctions using Qsys, refer to *Creating a System with Qsys* chapter in volume 1 of the *Quartus II Handbook*.



Altera recommends that you use the Quartus II software to generate megafunctions. However, if you are an expert user, you can generate megafunctions using command-line executables.

- If you want to use command line to generate the megafunctions, you can use one of the following methods:
 - Using the qmegawiz command-line executable. For more information, refer to “Using qmegawiz Command-Line Executable” on page 2-12.
 - Using the IP-generate command-line executable. For more information, refer to “Using IP-generate Command-Line Executable” on page 2-13.

On the other hand, you can infer megafunctions by writing codes for a function that you would like to achieve and then the synthesis tool interprets the HDL code to determine the components to be used. For more information, refer to “Inferring Megafunctions from HDL Code” on page 2-10.

3. If you generate more than one megafunction variations, connect the megafunction variations through the following methods:
 - Using the block editor
 - Using Qsys

For more information, refer to “Connecting Megafunctions” on page 2-3.

4. After instantiating the megafunction variations, the Quartus II software generates design source files and Verilog or VHDL simulation model files. Simulate these files in ModelSim®-AE, ModelSim SE, or other third-party simulation tools. For more information about simulating megafunctions, refer to [Chapter 3, Simulating Megafunctions](#).

This section describes how to generate megafunctions using the MegaWizard Plug-In Manager and command-line tools.

Generating Megafunctions Using the MegaWizard Plug-In Manager

The MegaWizard Plug-In Manager provides a GUI-based flow to create and modify design files that contain custom megafunction variations, which you can then instantiate in a design file. These custom megafunction variations are based on Altera-provided megafunctions. You can use the MegaWizard Plug-In Manager to create Altera megafunctions, LPM functions, and IP functions for use in designs in the Quartus II software, the EDA design entry, and synthesis tools.

You can start the MegaWizard Plug-In Manager through one of the following ways:

- From the Quartus II software, on the Tools menu, click **MegaWizard Plug-In Manager**.
- When you are working in the Block Editor, from the Edit menu, click **Insert Symbol 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 GUI-based MegaWizard Plug-In Manager by typing the following command at the command prompt:

```
qmegawiz ←
```

or

```
qmegawizq ←
```



`qmegawizq` and `qmegawiz` are command-line executables to launch the GUI-based MegaWizard Plug-In Manager without going through the Quartus II software. The difference between the two stand-alone executables is that you can use the `qmegawiz` executable to either launch the MegaWizard Plug-In Manager in command-line mode or through the GUI whereas the `qmegawizq` executable launches only the GUI-based MegaWizard-Plug-In Manager.

To use the `qmegawiz` executable in command-line mode, at the command prompt, type `qmegawiz` followed by the parameter and values as command-line options.

For more information about the `qmegawiz`, refer to [“Using qmegawiz Command-Line Executable”](#) on page 2-12.

Selecting a Device

To determine the device families supported for a particular megafunction, follow these steps:

1. Select a megafunction from the **Which megafunction would you like to customize?** list on page 2a of the MegaWizard Plug-In Manager.

The device families supported for that megafunction are automatically shown in the **Which device family will you be using?** pull-down list on the same page of the MegaWizard Plug-In Manager.

2. Select a device family from the **Which device family will you be using?** pull-down list on page 2a of the MegaWizard Plug-In Manager.

The megafunctions that are supported for that family are shown in the **Which megafunction would you like to customize?** list on the same page of the MegaWizard Plug-In Manager. The device families that are not supported are grayed out.

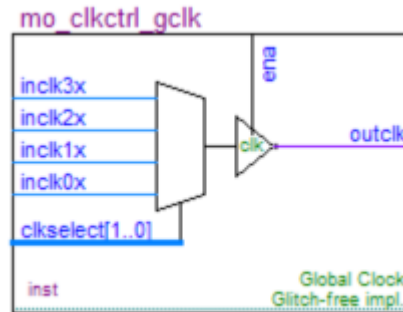
Creating a New Megafunction

To create a megafunction using the MegaWizard Plug-In Manager, follow these steps:

1. On the Tools menu, select **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
2. Select **Create a new custom megafunction variation**.
3. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.
4. Select a megafunction from the list.
5. Select your target device.
6. Select whether to create a Verilog HDL, VHDL, or a AHDL output file.
7. Specify a name for your output file. Click **Next**.
8. Customize and parameterize the megafunction.
9. If the parameter editor includes **EDA** and **Summary** tabs, follow these steps:
 - a. Some third-party synthesis tools can use a 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 if your synthesis tools support it, turn on **Generate Netlist** to generate a netlist file for area and timing estimation instead of a wrapper file. For more information, refer to [“Creating a Netlist for Other Synthesis Tools”](#) on page 2-7.
 - b. On the **Summary** tab, if available, select the files you want to generate. A gray check mark 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. For more information, [“Instantiating Megafunctions Using the MegaWizard Plug-In Manager”](#) on page 2-5.
10. Click **Finish**. The megafunction module is generated along with the files you specified in Step b.

11. To view the megafunction module in schematic form, open the generated Block Symbol File (.bsf) located in your project directory. The megafunction block symbol appears in the Symbol window.

Figure 2-1. mo_clkctrl_gclk Module



Connecting Megafunctions

You connect megafunctions using the block editor. The block editor allows you to create, edit, and integrate graphic design information, in the form of schematic block diagrams or block symbols. These symbols represent megafunctions, primitives or design files.

You can also connect `_hw.tcl`-based megafunctions that are available in Qsys (for example, `Altera_PLL`) using the Qsys design flow.

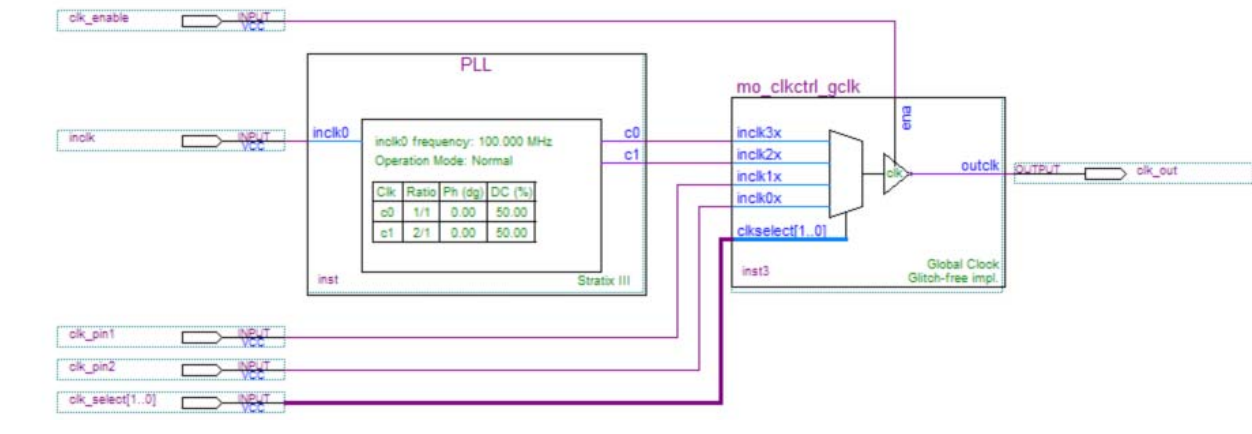
- For more information about connecting `_hw.tcl`-based megafunction components using Qsys, refer to “Connecting Components” section in the *Creating a System with Qsys* chapter of the *Quartus II Handbook*.

To connect two or more megafunctions using the block editor, follow these steps:

1. Open the first megafunction module `<megafunction name>.bsf` in the Quartus II Block Editor software.
2. To insert the second megafunction module, right-click on the Block Editor window, select **Insert**, and click **Symbol**. The Symbol window appears.
3. Under **Name**, browse to the second megafunction module `<megafunction name>.bsf`.
4. Click **OK**. The second megafunction module is inserted into the Block Editor window.

- Use the appropriate connectors from the Block Editor tool bar to connect the two megafunction modules. Figure 2-2 shows an example of a completed design file.

Figure 2-2. Complete Design Schematics



- In the File menu, click **Save**.
- Run a full compilation.

Identifying a Megafunction After Compilation

During compilation, the Quartus II software performs analysis and elaboration to build the structure of your design. To locate your megafunction in the Project Navigator window, expand the compilation hierarchy and find the megafunction by its name. To search for node names in the megafunction (using the Node Finder), click **Browse** in the **Look in** box and select the megafunction in the **Hierarchy** box.

Instantiating Megafunctions

You can instantiate Altera megafunctions and LPM functions in the Quartus II software through the following methods:

- “Instantiating Megafunctions Using the MegaWizard Plug-In Manager” on page 2-5. You can use the MegaWizard Plug-In Manager to parameterize the function and create a wrapper file.
- “Creating a Netlist for Other Synthesis Tools” on page 2-7. You can optionally create a netlist file instead of a wrapper file.
- “Instantiating Megafunctions Using the Port and Parameter Definition” on page 2-8. You can instantiate the function directly into your HDL code.

Instantiating Megafunctions Using the MegaWizard Plug-In Manager

After you have created and parameterized the megafunction, depending on which language you choose, the MegaWizard Plug-In Manager instantiates the megafunction with the correct parameters and generates a megafunction variation file (wrapper file) in Verilog HDL (.v), VHDL (.vhd), or AHDL (.tdf), along with other supporting files.

For non-`_hw.tcl`-based megafunctions, the MegaWizard Plug-In Manager provides options to create the files listed in [Table 2-1](#).

Table 2-1. MegaWizard Plug-In Manager Generated Files for non-`_hw.tcl`-based Megafunctions

File	Description
<code><variant_name>.v</code> (1)	<i>Verilog HDL Variation Wrapper File</i> This is a megafunction wrapper file for instantiation in a Verilog HDL design.
<code><variant_name>.vhd</code> (1)	<i>VHDL Variation Wrapper File</i> This is a megafunction wrapper file for instantiation in a VHDL design.
<code><variant_name>.tdf</code> (1)	<i>AHDL Variation Wrapper File</i> This is a megafunction wrapper file for instantiation in an AHDL design.
<code><variant_name>.inc</code>	<i>AHDL Include File</i> This file is used in AHDL designs.
<code><variant_name>.cmp</code>	<i>Component Declaration File</i> This file is used in VHDL designs.
<code><variant_name>.bsf</code>	<i>Block Symbol File</i> This file is for schematic designs and is used in Quartus II Block Design Files (.bdf).
<code><variant_name>_inst.v</code>	<i>Verilog HDL Instantiation Template</i> This file is a sample Verilog HDL instantiation of the module in the megafunction wrapper file.
<code><variant_name>_inst.vhd</code>	<i>VHDL Instantiation Template</i> This is a sample VHDL instantiation of the entity in the megafunction wrapper file.
<code><variant_name>_inst.tdf</code>	<i>Text Design File Instantiation Template</i> This is a sample AHDL instantiation of the subdesign in the megafunction wrapper file.
<code><variant_name>_bb.v</code>	<i>Black box Verilog HDL Module Declaration</i> This is a module declaration file that is used when instantiating the megafunction as a black box in a third-party synthesis tool.
<code><variant_name>_syn.v</code> (2)	<i>Synthesis area and timing estimation netlist</i> If you enable the option to generate a synthesis area and timing estimation netlist, the MegaWizard Plug-In Manager generates this additional synthesis netlist file. This file is the megafunction netlist used by certain third-party synthesis tools to improve their area and timing estimates.

Notes to Table 2-1:

- (1) The MegaWizard Plug-In Manager generates a Verilog HDL, VHDL, or AHDL Variation Wrapper File, depending on the language you select for the output file on the megafunction-selection page of the parameter editor.
- (2) The MegaWizard Plug-In Manager generates this file only if you turn on the **Generate a synthesis area and timing estimation netlist** option on the EDA page of the parameter editor. This file is generated in Verilog HDL format regardless of the HDL language selected.

For `_hw.tcl`-based megafunctions, the following files are generated automatically when you instantiate the megafunction.

Table 2–2. MegaWizard Plug-In Manager Generated Files for `_hw.tcl`-based Megafunctions (Part 1 of 2)

Directory and Filename	Description
<code><working directory>\<variant_name>.v (1)</code>	<i>Verilog HDL Variation Wrapper File</i> This is a megafunction top-level wrapper file for instantiation in a Verilog HDL design.
<code><working directory>\<variant_name>.vhd (1)</code>	<i>VHDL Variation Wrapper File</i> This is a megafunction top-level wrapper file for instantiation in a VHDL design.
<code><working directory>\<variant_name>.bsf</code>	<i>Block Symbol File</i> This file is for schematic designs and is used in Quartus II Block Design Files (<code>.bdf</code>).
<code><working directory>\<variant_name>.qip</code>	QIP file which refers to all generated files in the synthesis fileset.
<code><working directory>\<variant_name>\<variant_name>_0002.v</code>	Wrapper file.
<code><working directory>\<variant_name>_sim\<variant_name>.vo</code>	Verilog simulation model.
<code><working directory>\<variant_name>_sim\<variant_name>.vho</code>	VHDL simulation model.
<code><working directory>\<variant_name>_sim\mentor\msim_setup.tcl</code>	<i>Mentor Simulation Script</i> The simulation script for Mentor Graphics simulators ModelSim, ModelSim-Altera, and QuestaSim. The script creates alias commands to compile the required device libraries and system design files in the correct order and elaborate or load the top-level design for simulation.
<code><working directory>\<variant_name>_sim\synopsys\vcs_setup.sh (2)</code> <code><working directory>\<variant_name>_sim\synopsys\vcsmx_setup.sh</code>	<i>Synopsys VCS and VCS MX Simulation Shell Script</i> The simulation scripts for Synopsys VCS and VCS MX that contain shell commands to compile the required device libraries and system design files in the correct order, and elaborate the top-level design for simulation and run the simulation for 100 time units by default.

Table 2-2. MegaWizard Plug-In Manager Generated Files for `_hw.tcl`-based Megafunctions (Part 2 of 2)

Directory and Filename	Description
<code><working directory>\<variant_name>_sim\cadence\ncsim_setup.sh</code>	<p><i>Cadence Incisive Enterprise Simulation Shell Script</i></p> <p>The simulation script for the Cadence Incisive Enterprise Simulation Shell Script that contains shell commands to compile the required device libraries and system design files in the correct order, and elaborate the top-level design for simulation and run the simulation for 100 time units by default.</p>

Notes to Table 2-2:

- (1) The MegaWizard Plug-In Manager generates a Verilog HDL, VHDL, or AHDL Variation Wrapper File, depending on the language you select for the output file on the megafunction-selection page of the parameter editor.
- (2) The simulation script for Synopsys VCS is not generated for VHDL designs.

Creating a Netlist for Other Synthesis Tools

When you use non `hw.tcl`-based megafunctions with third-party EDA synthesis tools (tools other than the Quartus II Integrated Synthesis), you can optionally create a netlist for area and timing estimation instead of a wrapper file.

The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of architectural elements in the megafunction but might not represent true functionality (gray box). 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 on timing-driven optimizations and improve the quality of results.

To generate the netlist, turn on the **Generate a synthesis area and timing estimation netlist** option on the EDA page of the MegaWizard Plug-In Manager. The netlist file is called `<variant name>_syn.v` file. This file is in Verilog HDL format, regardless of the HDL language selected in the MegaWizard Plug-In Manager. If you use this netlist for synthesis, you must include the megafunction wrapper file `<variant name>.v` or `<variant name>.vhd` in your Quartus II project for placement and routing.


Your synthesis tool can call the Quartus II software in the background to generate this netlist, so you are not required to perform the extra step of turning on this option.



For information about support for area and timing estimation netlists in your synthesis tool, refer to the vendor's documentation or the appropriate chapter in the *Synthesis* section in volume 1 of the *Quartus II Handbook*.

Instantiating Megafunctions Using the Port and Parameter Definition

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 in any other module, component, or subdesign.

-  For a list of the megafunction ports and parameters, refer to the specific megafunction user guide. The megafunction user guide also provides a sample VHDL component declaration and AHDL function prototype for each megafunction.

The Verilog HDL and VHDL code samples shown in [Example 2-1](#) and [Example 2-2](#) instantiate an `ALTFP_MULT` in the top level module with one of the input connected to a 2:1 multiplexer. The parameters for the instance are defined directly in the code as well. For the correct megafunction name to be used in the instantiation, refer to the Quartus II Help.

-  When instantiating a megafunction in VHDL, be sure to include the correct libraries.

Example 2-1. Instantiating `ALTFP_MULT` in Verilog HDL

```
module MF_top (a, b, sel, datab, clock, result);
    input [31:0] a, b, datab;
    input clock, sel;
    output [31:0] result;
    wire [31:0] wire_dataa;

    assign wire_dataa = (sel)? a : b;
    altfp_mult inst1 (.dataa(wire_dataa), .datab(datab), .clock(clock),
    .result(result));

    defparam
        inst1.pipeline = 11,
        inst1.width_exp = 8,
        inst1.width_man = 23,
        inst1.exception_handling = "no";
endmodule
```

Example 2-2. Instantiating ALTFP_MULT in VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library altera_mf;
use altera_mf.altera_mf_components.all;

entity MF_top is
    port (clock, sel : in std_logic;
          a, b, datab : in std_logic_vector(31 downto 0);
          result      : out std_logic_vector(31 downto 0));
end entity;

architecture arch_MF_top of MF_top is
    signal wire_dataaa : std_logic_vector(31 downto 0);
begin

    wire_dataaa <= a when (sel = '1') else b;

    inst1 : altfp_mult
        generic map(
            pipeline => 11,
            width_exp => 8,
            width_man => 23,
            exception_handling => "no")
        port map (
            dataaa => wire_dataaa,
            datab => datab,
            clock => clock,
            result => result);
end arch_MF_top;
```

Inferring Megafunctions from HDL Code

Synthesis tools, including Quartus II Integrated synthesis, recognize certain types of HDL code and automatically infer the appropriate megafunction when a megafunction can provide optimal results. 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 area usage and performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features within memory, DSP blocks, and shift registers. These features provide improved performance when compared to basic logic elements (LEs).

You can access code examples or templates for inferred RAMs, ROMs, shift registers, arithmetic functions, and DSP functions optimized for Arria V, Cyclone V, and Stratix V devices in the Quartus II software.

To access these templates, follow these steps:

1. Open a file in the text editor.
2. On the Edit menu, click **Insert Template**.



You can also right-click in the text editor and click **Insert Template**.

3. In the **Insert Template** dialog box, depending on the language you use, click the + icon to expand either the **Verilog HDL** category or the **VHDL** category.
4. Under **Full Designs** category, expand the navigation tree until you see the type of functions you would like to infer.
5. Select the function to display the code for the selected template in the Preview pane, and click **Insert**.



For new DSP features optimized for Arria V, Cyclone V, and Stratix V devices, expand the **Arithmetic** category, and then expand the **DSP features (Stratix-V, Arria-V and Cyclone-V)** category.



[Example 2-3](#) and [Example 2-4](#) are Verilog HDL code examples for an unsigned and a signed multiplier that synthesis tools can infer as an LPM_MULT or ALTMULT_ADD megafunction. Each example fits into one DSP block 9-bit element. In addition, when register packing occurs, no extra logic cells for registers are required.

Example 2-3. Verilog HDL Unsigned Multiplier

```
module unsigned_mult (out, a, b);  
    output [15:0] out;  
    input [7:0] a;  
    input [7:0] b;  
    assign out = a * b;  
endmodule
```

Example 2-4. Verilog HDL Signed Multiplier with Input and Output Registers (Pipelining = 2)

```
module signed_mult (out, clk, a, b);  
    output [15:0] out;  
    input clk;  
    input signed [7:0] a;  
    input signed [7:0] b;  
    reg signed [7:0] a_reg;  
    reg signed [7:0] b_reg;  
    reg signed [15:0] out;  
    wire signed [15:0] mult_out;  
    assign mult_out = a_reg * b_reg;  
    always @ (posedge clk)  
    begin  
        a_reg <= a;  
        b_reg <= b;  
        out <= mult_out;  
    end  
endmodule
```

-  For more information about sample codes, refer to the [Recommended HDL Coding Styles](#) chapter in volume 1 of the *Quartus II Handbook*.
-  For more information about the synthesis options to control the megafunction inference, refer to the [Quartus II Integrated Synthesis](#) chapter in volume 1 of the *Quartus II Handbook*.

Generating Megafunctions Using the Command-Line Tools

This section describes the steps to generate megafunctions using command-line tools. You can generate megafunctions using the command-line through the following methods:

- Using qmegawiz command-line executable
- Using IP-generate command-line executable

Using qmegawiz Command-Line Executable

qmegawiz is a command-line executable version of the MegaWizard Plug-In Manager that allows you to modify, update, or create variation files without using the GUI.

To do this, use the following syntax:

```
qmegawiz [options] [module=<module name>] | wizard=<wizard name>]
[<param>=<value>...<port>=<used>|<unused>...] <variation file name>
```

Table 2-4 lists the description of the arguments that are commonly used.

Table 2-3. qmegawiz Syntax Options and Arguments

Options/Arguments	Description
-silent	Runs the MegaWizard Plug-In Manager in command-line mode, without displaying the GUI.
-f:<parameter file>	Specifies a .txt name that contains all the parameter and port values.
-p:<working directory>	Specifies the default working directory that qmegawiz uses when it generates files.
module=<module name>	Specifies the module or wizard name. When there are multiple wizard names that correspond to one module name, use the wizard option to specify one wizard. When there are multiple module names that correspond to one wizard name, use the module option to specify one module.
wizard=<wizard name>	
<param>=<value>	Specifies the parameter values.
<port>=<used> <unused>	Specifies whether the ports are used.
<variation file name>	Specifies a variation file name. Valid extensions are .v , .vhd , or .tdf .

For example,

```
qmegawiz -silent module=altlvds_rx wizard=altlvds common_rx_tx_pll=ON
tx_coreclock=used lvds_sample.v ←
```



For more information about the qmegawiz command-line options, refer to [Chapter 2: Command-Line Scripting](#) in volume 2 of the *Quartus II Handbook*.

Using IP-generate Command-Line Executable

You can use **ip-generate.exe**, a command-line executable, to configure parameters. The command creates or modifies custom megafunction variations, which you can then instantiate in a design file. The ip-generate command is normally used for **_hw.tcl**-based megafunctions such as the Altera_PLL and ALTDQ_DQS2 megafunctions.

To run the ip-generate command, follow these steps:

1. Type the following command at the command prompt of your operating system:

```
<ACDS installation directory>\quartus\sopc_builder\bin
```

2. The executable name is **ip-generate.exe**. To run the executable file, type the following command:

```
ip-generate
```

3. To instantiate the megafunction using the executable file, type the following syntax:

```
ip-generate --component-name=altdq_dqs2 --component-system-  
param=DEVICE_FAMILY="Stratix V" --file-set=QUARTUS_SYNTH --output-  
name[=<file_name>] --component-  
param[=<parameter_name>] [=<parameter_value>]
```

Table 2-4 lists the description of the arguments that are commonly used.

Table 2-4. IP-Generate Arguments

Arguments	Description
--component-name=<megafunction name>	Specifies the megafunction instance name.
--file-set=QUARTUS_SYNTH --output-name [=<file_name>]	A parameter that is used by the ip-generator to specify an output file. Valid extensions are .v , .sv , .vhd , or .tdf .
--component-system-param=DEVICE_FAMILY="<device family name>"	Specifies the target device family.
--component-param[=<parameter_name>] [=<parameter_value>]	Specifies the parameters and values.


This section describes the simulation guidelines and the simulation libraries for megafunctions.

Simulation Guidelines

To simulate megafunctions, follow these guidelines:


1. Determine whether your megafunction is based on the Hardware Component Description File (`_hw.tcl`) infrastructure or non-`_hw.tcl`-based. The type of infrastructure your megafunction design is based on determines the simulation tool flows.
2. The simulation flow depends on the following types of megafunctions instantiated in the design:
 - Designs that contain `_hw.tcl`-based megafunctions only

Use the simulation script templates provided with your megafunction to set up the simulation environment for your chosen simulator. The scripts compile the required device libraries and system design files in the correct order and elaborate or load the top-level design for simulation.

 For more information, refer to “Perform functional simulation with IP cores Based on `_hw.tcl`” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.
 - Designs that contain one or more non-`_hw.tcl`-based megafunctions

Automatically create scripts to set up and launch an EDA simulator with the Quartus II NativeLink feature, manually set up EDA simulation using the Simulation Library Compiler, or manually set up EDA simulation by writing your own ModelSim script (for ModelSim- Altera software only) to set up the simulation environment. For more information about writing scripts, refer to “Writing a ModelSim Simulation Script” on page 3–4.

The NativeLink feature launches the supported simulator of your choice from within the Quartus II software and automates the compilation and simulation of testbenches. The Simulation Library Compiler automatically compiles all libraries required for functional and gate-level timing simulation, and then you manually store the compiled libraries in the directory you specify.

 For more information about using the NativeLink feature, refer to “Launching the EDA Simulator with the NativeLink Feature” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

- For more information about the Simulation Library Compiler, refer to “EDA Simulation Library Compiler” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.
 - Designs that contain a combination of `_hw.tcl`-based and non-`_hw.tcl`-based megafunctions
Manually set up simulation in your EDA simulator using the Simulation Library Compiler or write your own simulation script to manually set up the simulation environment.
3. Determine the third party simulation tools to run the simulation. You can simulate your megafunction design using the following third party simulation tools:
- ModelSim-Altera Software—the ModelSim-Altera software includes precompiled simulation libraries for Altera-provided models. You do not have to create simulation libraries and compile simulation models for the precompiled Altera libraries.
 - For more information about simulating megafunctions in the ModelSim-Altera software, refer to “Simulating with the ModelSim-Altera software” section in the *Mentor Graphics ModelSim and QuestaSim Support* chapter of the *Quartus II Handbook*.
 - Synopsys VCS and VCS MX
 - For more information, refer to *Synopsys VCS and VCS MX Support* chapter in volume 3 of the *Quartus II Handbook*.
 - Cadence Incisive Enterprise Simulator
 - For more information, refer to *Cadence Incisive Enterprise Simulator* chapter in volume 3 of the *Quartus II Handbook*.
 - Aldec ActiveHDL and Riviera-Pro Support
 - For more information, refer to *Aldec ActiveHDL and Riviera-Pro Support* chapter in volume 3 of the *Quartus II Handbook*.


Simulation Model Libraries


To properly simulate the generated design files, you must include the simulation model file or files in your simulation tool. You can find the information about the file or files in the EDA tab in the MegaWizard Plug-In Manager. The Quartus II simulation libraries are located at the `<ACDS installation directory>\eda\sim_lib` directory.

Table 3–1 lists the commonly used simulation model files for megafunction designs written in Verilog and VHDL.

Table 3–1. Megafunction Types and Features

Megafunction Type	Simulation Model Files	
	Verilog	VHDL
LPM functions	220model.v	<ul style="list-style-type: none"> ■ 220model.vhd for VHDL-93 compliant designs ■ 220model_87.vhd for VHDL-87 compliant designs ■ 220pack.vhd—Libraries that contain VHDL Component Declarations. <p>For designs coded in VHDL, the simulation model libraries (<code><library_name>.vhd</code> or <code><library_name>_atoms.vhd</code>) contain functional descriptions or atom declarations, and these simulation model libraries must be simulated together with PACK file (<code><library_name>_pack.vhd</code>) or COMPONENT file (<code><library_name>_components.vhd</code>)</p>
Altera-specific functions	<ul style="list-style-type: none"> ■ altera_mf.v ■ altera_primitives.v (include this file when your designs contain Altera-specific primitives) 	<ul style="list-style-type: none"> ■ altera_mf.vhd for VHDL-93 compliant designs ■ altera_mf_87.vhd for VHDL-87 compliant designs ■ altera_primitives.v (include this file when your designs contain Altera-specific primitives)

 For a list of all functional simulation libraries files in the Quartus II directory, refer to [Altera Functional Simulation Libraries](#) in the Quartus II Help.

 The ModelSim-Altera software includes precompiled simulation libraries for Altera-provided models. For a list of precompiled library names for all functional simulation models, refer to [ModelSim-Altera Precompiled Libraries](#) in the Quartus II Help.

Writing a ModelSim Simulation Script

For non-`_hw.tcl`-based megafunctions that do not have simulation script templates provided with them, you can manually set up the simulation environment by writing your own ModelSim simulation script to compile the simulation libraries, load the design, add signals to the Wave window, provide stimulus to the signals, and run the simulation.

Example 3-1 shows a sample simulation script to simulate a megafunction design file in the ModelSim-Altera software.

Example 3-1. ModelSim Simulation Script

```
# Creates the working directory
vlib work

# Reads the simulation library
vlog /quartus/eda/sim_lib/<library_name>.v

# Compiles the Verilog HDL or VHDL design files
#For Verilog HDL designs:
vlog -work work <design name>.v

# For VHDL designs:
vcom -work work <design name>.vhd

# Loads and simulate the design:
# For VHDL designs:
vsim -t ps work.<top level design entity>

#For Verilog HDL designs:
vsim -L altera_mf_ver -L lpm_ver <top-level design module>

# Adds port signals to the waveform
add wave /<top level design entity>/*

# Runs the simulation for 1000 ns
run 1000 ns
```

You can execute the ModelSim simulation script from within the GUI or you can run them from the ModelSim transcript window.

To execute the script from the ModelSim transcript window, type the following command at the VSIM> prompt:

```
do <script_filename>.do
```



For steps to run the script from the GUI, refer to “Simulating with the ModelSim-Altera software” section in the *Mentor Graphics ModelSim and QuestaSim Support* chapter of the *Quartus II Handbook*.

This chapter provides additional information about the document and Altera.

Document Revision History

The following table shows the revision history for this user guide.

Date	Version	Changes
November 2011	2.0	The reorganized version replacing the <i>Megafunction Overview User Guide</i> .

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) (Software Licensing)	Email	nacomp@altera.com
	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, <code>\qdesigns</code> directory, D: drive, and <code>chiptrip.gdf</code> file.
<i>Italic Type with Initial Capital Letters</i>	Indicate document titles. For example, <i>Stratix IV Design Guidelines</i> .
<i>italic type</i>	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.

Visual Cue	Meaning
"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, <code>data1</code> , <code>tdi</code> , and <code>input</code> . The suffix <code>n</code> denotes an active-low signal. For example, <code>resetn</code> . Indicates command line commands and anything that must be typed exactly as it appears. For example, <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword <code>SUBDESIGN</code>), and logic function names (for example, <code>TRI</code>).
	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.