



Floating Point Multiplier (ALTFP_MULT)

Megafunction User Guide



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

Quartus II Software Version: 8.0
Document Version: 3.0
Document Date: June 2008

Copyright © 2008 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.



UG-052404-3.0

Chapter 1. About this Megafunction

Device Family Support	1-1
Introduction	1-1
Features	1-1
General Description	1-2
The Format	1-2
Special Case Numbers	1-3
Rounding	1-3
Algorithm for Floating-Point Calculation	1-4
Exception Handling	1-4
Common Applications	1-5
Resource Utilization	1-5

Chapter 2. Getting Started

Software and System Requirements	2-1
MegaWizard Customization	2-1
Instantiating Megafunctions in HDL Code or Schematic Designs	2-9
Generating a Netlist for EDA Tool Use	2-9
Using the Port and Parameter Definitions	2-10
Identifying a Megafunction after Compilation	2-10
Simulation	2-10
Quartus II Simulation	2-10
EDA Tool Simulation	2-11
Design Files	2-12
Example	2-12
Generate the Multiplier	2-12
Implement the Multiplier	2-13
Functional Simulation in the ModelSim-Altera Simulator	2-15
Understanding the Simulation Results	2-16
Truth Table	2-18
Conclusion	2-19

Chapter 3. Specifications

Ports and Parameters	3-1
----------------------------	-----

Additional Information

Revision History	Info-1
Referenced Documents	Info-2
How to Contact Altera	Info-2
Typographic Conventions	Info-3

Device Family Support

The ALTFP_MULT 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

Introduction

As design complexities increase, the use of vendor-specific intellectual property (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. The Altera-provided functions offer more efficient logic synthesis and device implementation. You can scale the megafunction's size by setting parameters.

Features

The ALTFP_MULT megafunction implements multiplier functions and offers the following additional features:

- Multiplication of numbers in single precision, single-extended precision, and double-precision
- Dedicated multiplier circuitry in Stratix series, Cyclone series, and HardCopy series devices
- Support for input of normal numbers, infinity, zero, and not-a-number (NaN)
- Optional input ports such as the asynchronous-clear and clock-enable ports
- Optional exception-handling output ports such as the zero, overflow, underflow, and nan ports

General Description

The ALTFP_MULT megafunction follows the IEEE-754 Standard for Binary Floating-Point Arithmetic and defines the following:

- The formats for representing floating-point numbers
- The representation of special values (zero, infinity, denormal numbers, and bit combinations that do not represent a number)
- Exception signals, rounding modes, and a set of operations that work identically on any conforming system

The IEEE-754 standard defines four formats for floating-point numbers. The four formats are: single precision, double precision, single-extended precision, and double-extended precision. The most commonly used floating-point formats are single precision and double precision. The ALTFP_MULT megafunction supports only three formats: single-precision, double-precision, and single-extended precision.

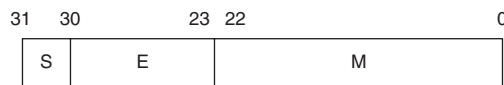
The Format

The single-precision, double-precision, and single-extended precision formats are available to represent floating-point numbers.

Single-Precision Format

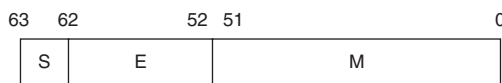
In the single-precision format, the most significant bit (MSB) is a sign bit, followed by 8 intermediate bits to represent an exponent, and 23 least significant bits (LSBs) to represent the mantissa. As a result, the total width of single-precision numbers is 32 bits. The bias for the single-precision format is 127. Refer to [Figure 1-1](#).

Figure 1-1. Single-Precision Representation



Double-Precision Format

In the double-precision format, the MSB is a sign bit, followed by 11 intermediate bits to represent an exponent, and 52 LSBs to represent the mantissa. As a result, the total width of double-precision numbers is 64 bits. The bias for double-precision format is 1023. Refer to [Figure 1-2](#).

Figure 1–2. Double-Precision Representation

Single-Extended Precision Format

In single-extended precision format, the MSB is a sign bit. However, there are no fixed widths for the exponent and mantissa fields. The exponent field has a minimum of 11 bits and its width must be less than the width of the mantissa field. The mantissa field has a minimum of 31 bits. The sum of the widths of the sign bit, the exponent field, and the mantissa field is a minimum of 43 bits and a maximum of 64 bits. The bias for the single-extended precision format is unspecified in the IEEE-754 standard. In this multiplier, a bias of $2^{(\text{WIDTH_EXP}-1)}-1$ is assumed for single-extended precision.

Special Case Numbers

Table 1–1 shows the special case numbers defined by the IEEE-754 1985 Standard for Binary Arithmetic and their data bit representations.

Table 1–1. Representation of Special Case Numbers in IEEE 754-1985 Standard

Meaning	Sign Field	Exponent Field	Mantissa Field
Zero	Don't care	All 0's	All 0's
Positive Denormalized	0	All 0's	Non-zero
Negative Denormalized	1	All 0's	Non-zero
Positive Infinity	0	All 1's	All 0's
Negative Infinity	1	All 1's	All 0's
Not-a-Number (NaN)	Don't care	All 1's	Non-zero

Rounding

In the IEEE-754 standard, there are four types of rounding modes: round-to-nearest-even, round-toward-zero, round-toward-positive-infinity, and round-toward-negative-infinity. The most commonly used rounding mode is round-to-nearest-even. This multiplier uses only the round-to-nearest-even mode. With the round-to-nearest-even mode, the result is rounded to the nearest

floating-point number. If the result is exactly halfway between two floating-point numbers, it is rounded so that the LSB becomes zero, which is even.

Algorithm for Floating-Point Calculation

Given two decimal inputs, A and B , the result R of the multiplication is as follows:

$$R = (M_a \times 2^{Ea}) \times (M_b \times 2^{Eb}) = (M_a \times M_b) \times 2^{Ea+Eb}$$

where:

- Ea is the exponent bit of A
- Eb is the exponent bit of B
- M_a is the mantissa bit of A
- M_b is the mantissa bit of B

Therefore, R has the following values:

- Sign = (sign bit of A) XOR (sign bit of B)
- Exponent = exponent of A + exponent of B – bias
- Mantissa = mantissa of A × mantissa of B

The exponent of A and the exponent of B are stored with bias adjustments in the IEEE-754 floating-point number. Therefore, when the two exponents are added together, the extra exponent must be removed.

The following calculations show how the addition of two bias numbers causes an extra bias:

- $\text{Exp_A_bias} = \text{Exp_A_actual} + \text{bias}$
- $\text{Exp_B_bias} = \text{Exp_B_actual} + \text{bias}$
- $\text{Exp_A_bias} + \text{Exp_B_bias}$
= $\text{Exp_A_actual} + \text{bias} + \text{Exp_B_actual} + \text{bias}$
= $\text{Exp_A_actual} + \text{Exp_B_actual} + \text{bias} + \text{bias}$

For an IEEE-754 standard floating-point number, only one bias adjustment is needed. Adding two bias numbers together causes an extra bias on the result of the calculation. The extra bias must be removed to obtain the correct result.

Exception Handling

The ALTFP_MULT megafuction provides four optional exception signals: zero, overflow, underflow, and nan.

For more information on the exception ports, refer to [Table 3–2 on page 3–2](#).

Common Applications

The advantage of floating-point numbers is that they can represent a much larger range of values. In a fixed-point number representation, the radix point is always at the same location. While the convention simplifies numeric operations and conserves memory, it places a limit on the magnitude and precision of the number representation. In situations that require a large range of numbers or high resolution, a relocatable radix point is desirable. Very large and very small numbers can be represented in a floating-point format. Multiplication of floating-point numbers is also commonly required in DSP-based applications.

Resource Utilization

[Table 1–2](#) summarizes the resource usage of the ALTFP_MULT megafunction for Stratix II and Stratix III devices with dedicated multiplier circuitry.

Device Family	Precision	Output Latency	Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Logic Usage	f _{MAX}
Stratix II	Single	5	126	148	173	228
	Double	5	383	410	480	122
Stratix III	Single	5	126	148	172	246
	Double	5	296	356	398	242

Note for Table 1–2:

- (1) You can get the performance information from the ALTFP_MULT MegaWizard® Plug-In Manager. The information in this table is valid and accurate in the Quartus® II software version 8.0.

Table 1–3 summarizes the resource usage of the ALTFP_MULT megafunction for Cyclone II and Cyclone III devices with dedicated multiplier circuitry.

Table 1–3. ALTFP_MULT Resource Usage for Cyclone II and Cyclone III Devices with Dedicated Multiplier Circuitry *Note (1)*

Device Family	Precision	Output Latency	Logic Elements (LEs)	Dedicated Logic Registers (DLRs)	Logic Usage	f _{MAX}
Cyclone II	Single	5	296	221	296	188
	Double	5	688	412	688	101
Cyclone III	Single	5	295	221	295	209
	Double	5	687	412	687	122

Note for Table 1–3:

- (1) You can get the performance information from the ALTFP_MULT MegaWizard Plug-In Manager. The information in this table is valid and accurate in the Quartus II software version 8.0.

Software and System Requirements

The instructions in this section require the following hardware and 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 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 Floating Point Multiplier (ALTFP_MULT) megafunction features in your design.

Start the MegaWizard Plug-In Manager using one of the following methods:

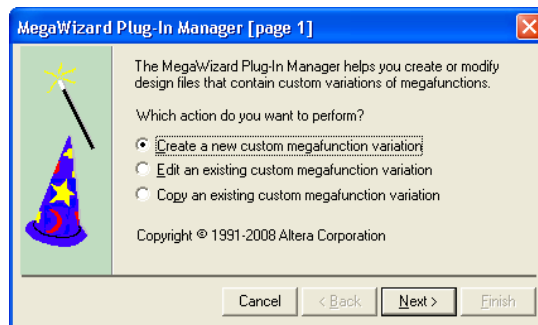
- On the Tools menu, click **MegaWizard Plug-In Manager**.
- When working in the Block Editor, from the Edit menu, click **Insert Symbol as Block**, or right-click in the Block Editor, point to Insert, and click **Symbol as Block**. In the **Symbol** dialog box, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:
`qmegawiz`↵

MegaWizard Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALTFP_MULT MegaWizard Plug-In Manager.

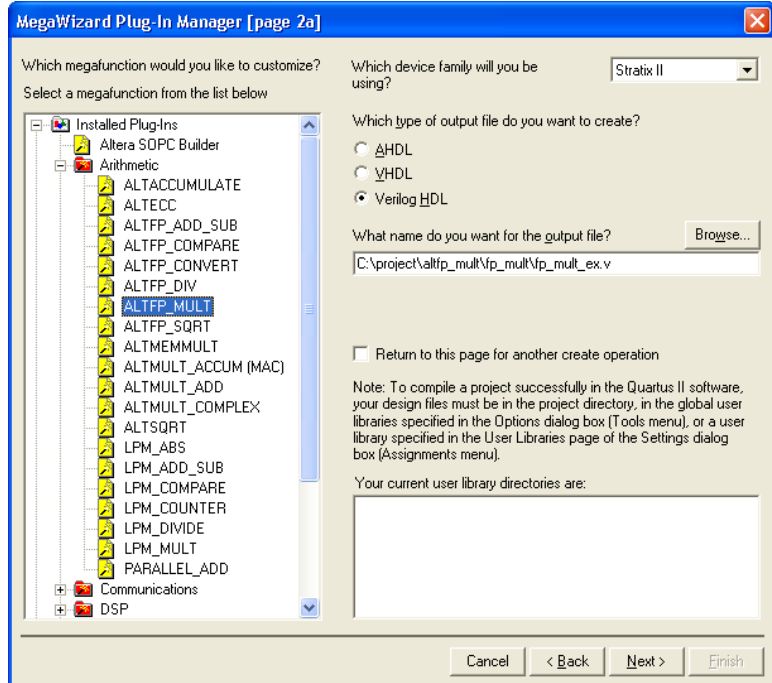
On page 1 of the MegaWizard Plug-In Manager, click **Create a new custom megafunction variation**, **Edit an existing custom megafunction variation**, or **Copy an existing custom megafunction variation** (Figure 2-1).

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



On page 2a of the MegaWizard Plug-In Manager, specify the megafunction device family, type of output file to create, and output file name (Figure 2–2). You can 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 ALTFP_MULT MegaWizard Plug-In Manager, select the type of precision, verify the widths of the input and output buses, verify the widths of the exponent and mantissa, and specify the output latency in clock cycles. There is also an option to use the dedicated multiplier circuitry (Figure 2–3).

Figure 2–3. ALTFP_MULT MegaWizard Plug-In Manager [page 3 of 6]

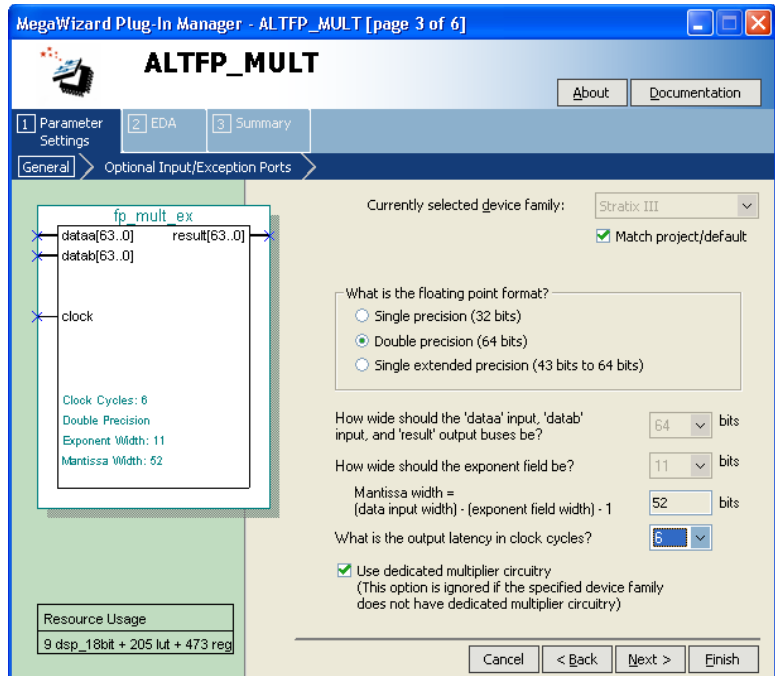
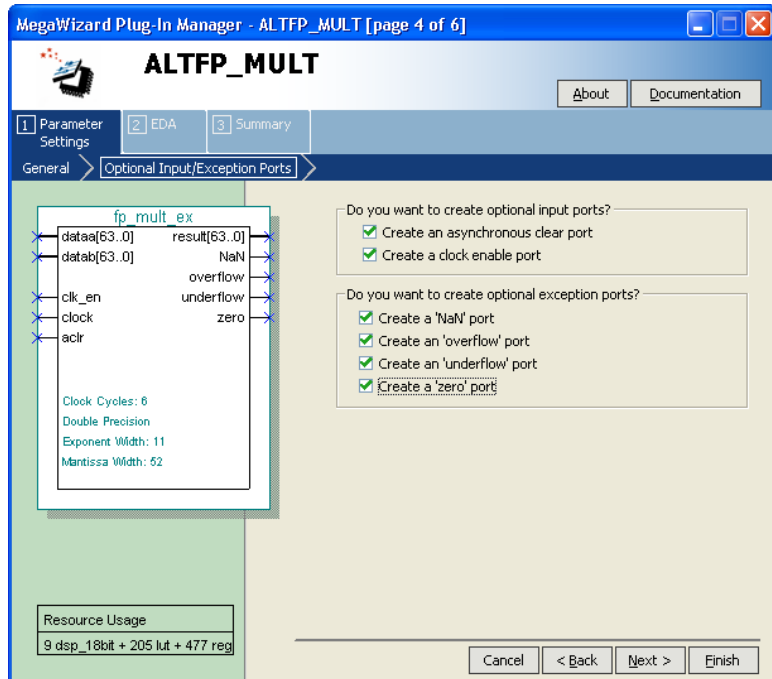


Table 2–1 shows the options available on page 3 of the ALTFP_MULT MegaWizard Plug-In Manager.

Table 2–1. ALTFP_MULT MegaWizard Plug-In Manager (page 3) Options	
Configuration Setting	Description
Currently selected device family	Specify the device family you want to use.
Match project/default	Select this option to ensure that the device selected matches the device family chosen on the previous page.
What is the floating point format?	Select Single precision for 32 bits, Double precision for 64 bits, or Single-extended precision for 43 to 64 bits.
How wide should the 'dataa' input, 'datab' input, and 'result' output buses be?	Specify the widths of the buses. The maximum width is 32 bits for single precision, 64 bits for double precision, and 64 bits for single-extended precision.
How wide should the exponent field be?	Specify the width of the exponent field. The maximum width is 30 bits.
Mantissa width = (data input width) – (exponent field width) – 1	This is not a user-specified option. The value is automatically calculated when the widths of the exponent field and input buses are specified.
What is the output latency in clock cycles?	Specify the latency for the result output in clock cycles. The valid values are 5, 6, 10, and 11. The default value is 5.
Use dedicated multiplier circuitry (This option is ignored if the specified device family does not have dedicated multiplier circuitry)	This option is available for the Stratix® series, Cyclone® series, and HardCopy® series devices that have dedicated multipliers. If this option is not selected, the multiplier is implemented with logic elements.

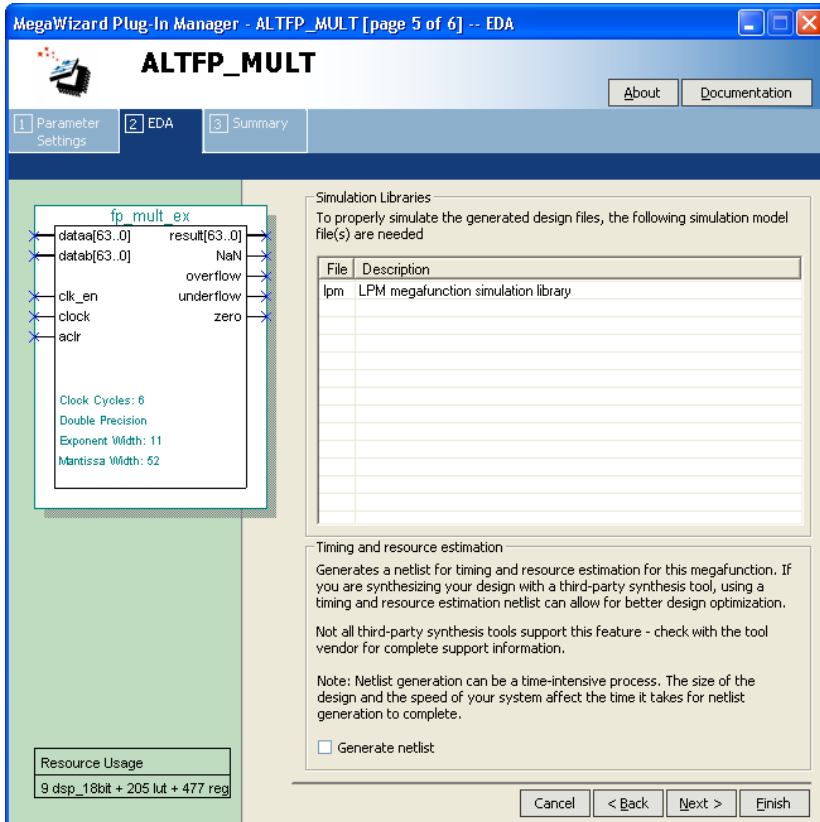
On page 4 of the ALTFP_MULT MegaWizard Plug-In Manager, you can create optional input ports and select optional exception ports for your design (Figure 2-4).

Figure 2-4. ALTFP_MULT MegaWizard Plug-In Manager [page 4 of 6]



On page 5 of the ALTFP_MULT MegaWizard Plug-In Manager, you can choose to generate a netlist for your third-party EDA synthesis tool to estimate the timing and resource usage of the megafunction (Figure 2-5).

Figure 2-5. ALTFP_MULT MegaWizard Plug-In Manager [page 5 of 6]

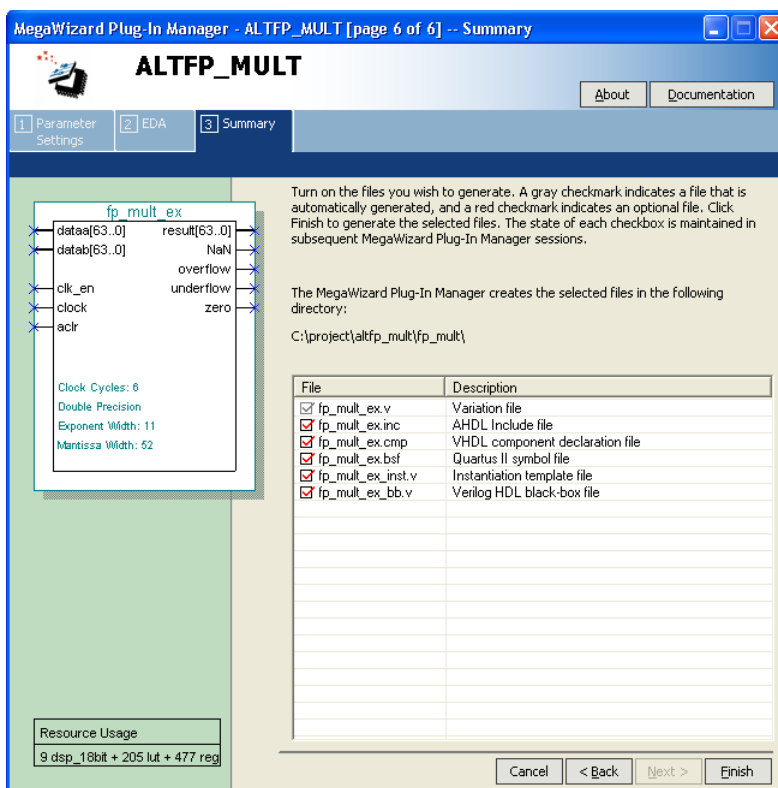


Page 6 of the ALTFP_MULT MegaWizard Plug-In Manager displays the types of files to be generated. The Variation file, which is automatically generated, contains wrapper code in the language you specified on page 2a. On page 6 of the ALTFP_MULT MegaWizard Plug-In Manager, specify the types of files to be generated. 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 5, 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. ALTFP_MULT MegaWizard Plug-In Manager [page 6 of 6]



For more information about the ports and parameters for this megafunction, refer to [Chapter 3, Specifications](#).

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 MegaWizard 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
- ADHL 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 MegaWizard 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 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 the MegaWizard Plug-In Manager, you can instantiate the megafunction directly in your Verilog HDL, VHDL, or AHDL code by calling the megafunction and settings 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. To locate your megafunction in the Project Navigator window, expand the compilation hierarchy and find the megafunction by its name.

To search for node names within the megafunction (using the Node Finder), click **Browse** in the **Look in** box and select the megafunction in the **Hierarchy** box.

Simulation

The Quartus II Simulator 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 enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, you can view the

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 can view the signals that exist after place-and-route. These signals are found with the post-compilation filter of the Node Finder. During synthesis and place-and-route, the names of RTL signals change. Therefore, it 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 Tool Simulation

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: Multiplication of Double- Precision Numbers

This design example uses the ALTFP_MULT megafunction to implement a floating-point multiplier for the multiplication of double-precision numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

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

Example

In this example, the following tasks are performed:

- Create a floating-point multiplier using the ALTFP_MULT megafunction and the MegaWizard Plug-in Manager.
- Assign the EP2S60F484C3 device to the project. Then, compile and simulate the design.

Generate the Multiplier

1. Open the `altfp_mult_DesignExample_ex.zip` file and extract `fp_mult_ex.qar`.
2. In the Quartus II software, open `fp_mult_ex.qar` and restore the archive file into your working directory.
3. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
4. Select **Create a new custom megafunction variation**, and click **Next**. Page 2a appears.

- In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in [Table 2-2](#). Click **Next** to advance from one page to the next.

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Select a megafunction	Under Arithmetic, select ALTFP_MULT
	Which device family will you be using?	Stratix II
	Which type of output file do you want to create?	Verilog HDL
	What name do you want for the output file?	fp_mult_ex
3	Currently selected family	Stratix II
	Match project/default	Selected
	What is the floating point format?	Double precision (64 bits)
	How wide should the 'dataa' input, 'datab' input, and 'result' output buses be?	64 bits
	How wide should the exponent field be?	11 bits
	Mantissa width = (data input width) – (exponent field width) – 1	52 bits
	What is the output latency in clock cycles?	6
	Use dedicated multiplier circuitry	Selected
4	Do you want to create optional input ports?	Select all
	Do you want to create optional exception ports?	Select all
5	Generate netlist	Not selected
6	Variation file	Selected
	Instantiation template file	Not selected
	Verilog HDL black-box file	Selected
	AHDL Include file	Not selected
	VHDL component declaration file	Not selected
	Quartus II symbol file	Selected

Implement the Multiplier

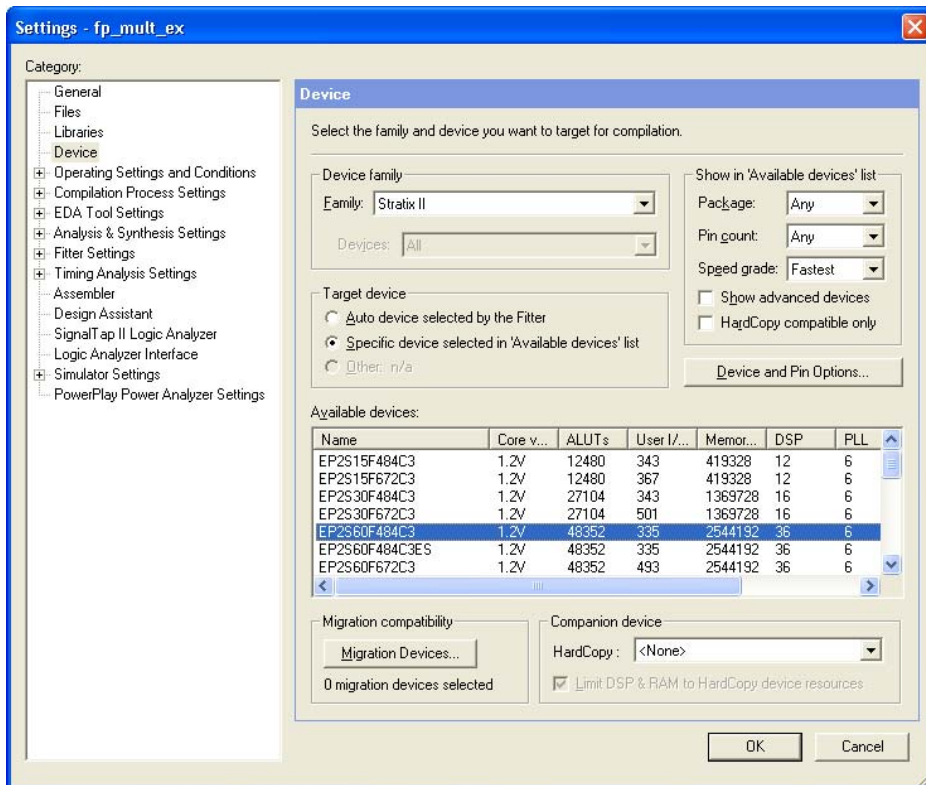
Next, perform the following steps to assign the EP2S60F484C3 device to the project and compile the project.

- On the Assignments menu, click **Settings**. The **Settings** dialog box appears.

2. Under Category, select **Device**.
3. In the Family list, select **Stratix II**.
4. Under Target Device, select **Specific device selected in 'Available devices' list**.
5. In the **Available devices** list, select **EP2S60F484C3**.

Figure 2-7 shows the **Device Settings** dialog box after these selections are made.

Figure 2-7. Device Settings Dialog Box



6. Click **OK**.
7. On the Processing menu, click **Start Compilation**, or click **Start Compilation** on the toolbar, to compile the design.

8. When the **Full Compilation was successful** message box appears, click **OK**.

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 out 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:

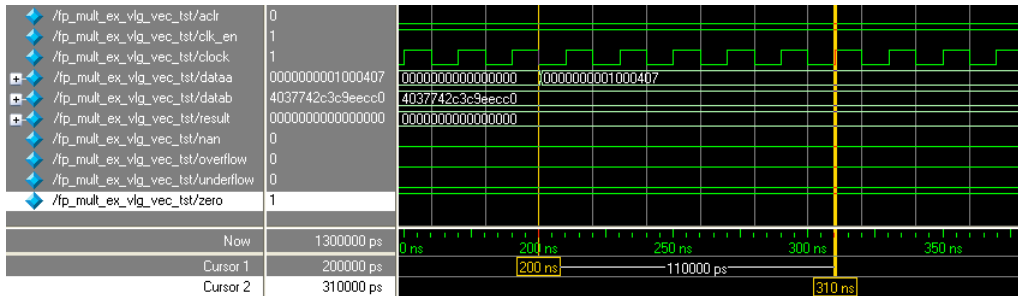
1. Unzip the **alftp_mult_ex_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**.
6. On the Tools menu, click **Execute Macro**.
7. Select the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

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in the **fp_mult_ex.do** file.

Figure 2–8 shows the expected simulation results in the ModelSim-Altera software. For more information about the simulation results, refer to “Understanding the Simulation Results” on page 2–16.

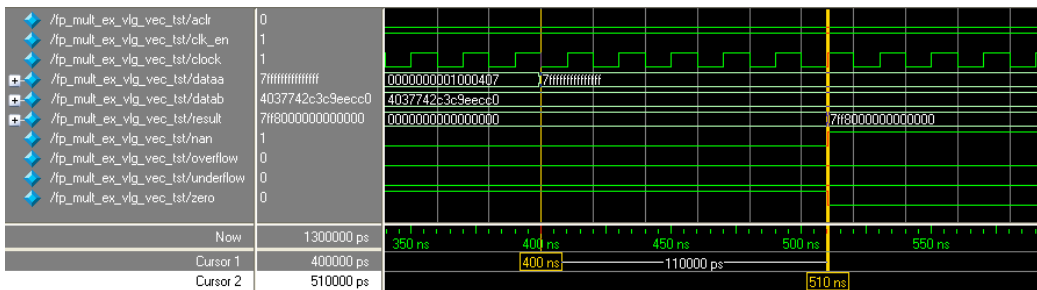
The second multiplication occurs between a normal value and a denormal value at 200 ns. The result is sent to the `result` port 6 clock cycles later at 310 ns. As a denormal number is a forced-zero value, this multiplication results in a zero, thus causing the assertion of the `zero` port at 310 ns. Refer to [Figure 2–10](#).

Figure 2–10. Multiplication of a Normal and a Denormal Value Results in a Zero



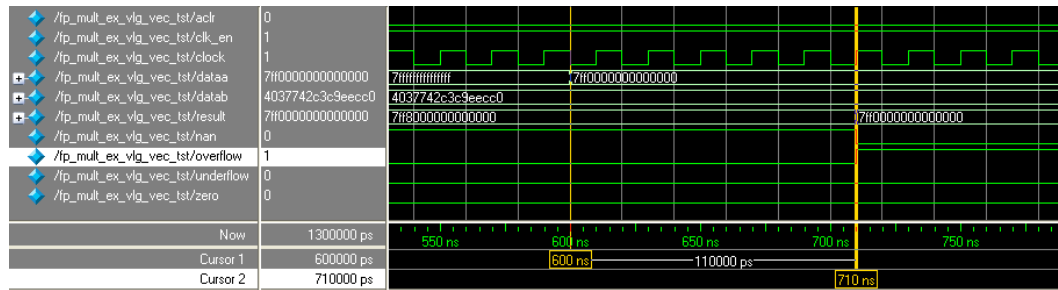
[Figure 2–11](#) shows the multiplication of a NaN and a normal value from the `dataa` and `datab` ports, respectively, at 400 ns. The result is sent to the `result` port 6 clock cycles later at 510 ns, where the previous `zero` port is deasserted. Any multiplication with a NaN results in a NaN.

Figure 2–11. Multiplication of a NaN and a Normal Value Results in a NaN



[Figure 2–12](#) shows that the multiplication of an infinity value and a normal value results in an infinity. The `overflow` port is asserted at 710 ns. All multiplications with an infinity value results in an infinity except when infinity is multiplied with zero.

Figure 2–12. Multiplication of an Infinity Value and a Normal Value Results in an Overflow



Truth Table

Table 2–3 shows the truth table for the ALTFP_MULT megafunction.

Table 2–3. Truth Table for ALTFP_MULT Megafunction (Part 1 of 2)

dataa[]	datab[]	result[]	overflow	underflow	zero	nan
Normal	Normal	Normal	0	0	0	0
Normal	Normal	Denormal (1)	0	1	1	0
Normal	Normal	Infinity	1	0	0	0
Normal	Normal	Zero	0	1	1	0
Normal	Denormal	Zero	0	0	1	0
Normal	Zero	Zero	0	0	1	0
Normal	Infinity	Infinity	1	0	0	0
Normal	NaN	NaN	0	0	0	1
Denormal	Normal	Zero	0	0	1	0
Denormal	Denormal	Zero	0	0	1	0
Denormal	Zero	Zero	0	0	1	0
Denormal	Infinity	NaN	0	0	0	1
Denormal	NaN	NaN	0	0	0	1
Zero	Normal	Zero	0	0	1	0
Zero	Denormal	Zero	0	0	1	0
Zero	Zero	Zero	0	0	1	0
Zero	Infinity	NaN	0	0	0	1
Zero	NaN	NaN	0	0	0	1
Infinity	Normal	Infinity	1	0	0	0
Infinity	Denormal	NaN	0	0	0	1

Table 2–3. Truth Table for ALTFP_MULT Megafunction (Part 2 of 2)

dataa[]	datab[]	result[]	overflow	underflow	zero	nan
Infinity	Zero	NaN	0	0	0	1
Infinity	Infinity	Infinity	1	0	0	0
Infinity	NaN	NaN	0	0	0	1
NaN	Normal	NaN	0	0	0	1
NaN	Denormal	NaN	0	0	0	1
NaN	Zero	NaN	0	0	0	1
NaN	Infinity	NaN	0	0	0	1
NaN	NaN	NaN	0	0	0	1

Note to Table 2–3:

- (1) When the `zero` and `underflow` flags are asserted, any denormal output value is replaced with a zero.

Conclusion

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

Ports and Parameters

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



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

Figure 3–1 shows the input and output ports for the ALTFP_MULT megafunction.

Figure 3–1. Input and Output Ports of the ALTFP_MULT Megafunction

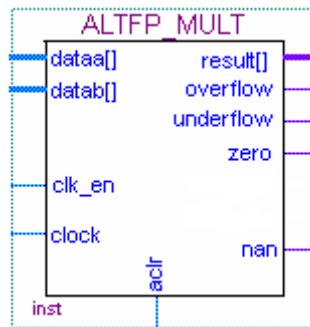


Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameters for the ALTFP_MULT megafunction.

Table 3–1. ALTFP_MULT Megafunction Input Ports (Part 1 of 2)			
Port Name	Required	Description	Comments
clock	Yes	Clock input to the multiplier.	—
clk_en	No	Clock enable for the multiplier.	Allows multiplication to take place when <code>clk_en</code> is asserted high. When this signal is low, no multiplication takes place and the outputs remain the same.

Table 3–1. ALTFP_MULT Megafunction Input Ports (Part 2 of 2)

Port Name	Required	Description	Comments
<code>aclr</code>	No	Asynchronous clear for the multiplier.	The core is asynchronously reset when the <code>aclr</code> signal is asserted high.
<code>dataa[]</code>	Yes	Data input to the multiplier.	Input port that is $[\text{WIDTH_EXP} + \text{WIDTH_MAN}.0]$ wide.
<code>datab[]</code>	Yes	Data input to the multiplier.	Input port that is $[\text{WIDTH_EXP} + \text{WIDTH_MAN}.0]$ wide.

Table 3–2. ALTFP_MULT Megafunction Output Ports

Port Name	Required	Comments
<code>result</code>	Yes	The floating-point result (after rounding). As with the input values, the MSB is the sign, the next MSB is the exponent, and the mantissa occupies the LSB. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
<code>overflow</code>	No	This port is asserted when the result of the multiplication (after rounding) exceeded or reached infinity. Infinity is defined as a number in which the exponent exceeds $\pm (2^{\text{WIDTH_EXP}-1})$.
<code>underflow</code>	No	This port is asserted when the result of the multiplication (after rounding) is zero while none of the multiplication inputs is zero. This port is also asserted when the result is a denormalized number.
<code>zero</code>	No	This port is asserted when the value of <code>result []</code> is zero.
<code>nan</code>	No	This port is asserted when an invalid multiplication occurs, such as the multiplication of infinity and zero. In this case, a NaN is the output generated at the <code>result []</code> port. The multiplication of any value and a NaN produces a NaN.

Table 3–3. ALTFP_MULT Megafunction Parameters

Parameter Name	Type	Required	Description
WIDTH_EXP	Integer	No	Specifies the value of the exponent. If omitted, the default is 8. The bias of the exponent is always set to $2 * \text{WIDTH_EXP} - 1$. For example, the bias is 127 for a single-precision floating-point format and 1023 for a double-precision floating-point format. The WIDTH_EXP value must be 8 for a floating-point format in single-precision or a minimum of 11 for floating-point formats in double precision and single-extended precision. The WIDTH_EXP value must be less than the WIDTH_MAN value. The sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
WIDTH_MAN	Integer	No	Specifies the value of the mantissa. If omitted, the default is 23. When the WIDTH_EXP value is 8 and the floating-point format is single precision, the WIDTH_MAN value must be 23. Otherwise, the value of WIDTH_MAN must be a minimum of 31. The WIDTH_MAN value must always be greater than the WIDTH_EXP value. The sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
DEDICATED_MULTIPLIER_CIRCUITRY	String	No	Specifies whether to use dedicated multiplier circuitry. Values are AUTO, YES, or NO. If omitted, the default is AUTO. If the selected device does not have dedicated multiplier circuitry, the DEDICATED_MULTIPLIER_CIRCUITRY parameter has no effect and defaults to NO.
PIPELINE	Integer	No	Specifies the number of clock cycles needed to produce the multiplied result. Values are 5, 6, 10, and 11. If omitted, the default is 5.
ROUNDING	String	Yes	Specifies the rounding mode. The default is TO_NEAREST. Other rounding modes are not supported.



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

Date & Document Version	Changes Made	Summary of Changes
June 2008 v3.0	<p>Added the following:</p> <ul style="list-style-type: none">● Table 2–2 and Table 2–3● “Design Example: Multiplication of Double-Precision Numbers” section <p>Updated the following:</p> <ul style="list-style-type: none">● “Referenced Documents” section● “Features” section● “Exception Handling” section● “Resource Utilization” section● “MegaWizard Page Descriptions” section <p>Removed the following:</p> <ul style="list-style-type: none">● Output ports and parameters that are no longer supported● Table on “Exception Handling Output Ports”● “Reduced Functionality” section● “Denormal Input and Result Handling” section● “SignalTap® II Embedded Logic Analyzer” section● “Design Example 1: Multiplication of Double-Precision Numbers with Full Functionality” section● “Design Example 2: Multiplication of Double-Precision Numbers with Reduced Functionality” section	Updated document to support the Quartus® II software version 8.0.
June 2007 v2.3	<p>Updated the following for the Quartus II software version 7.1:</p> <ul style="list-style-type: none">● Added support for Arria™ GX devices.● Added new megafunction parameters in “General Description” on page 1–2.● Modified the “Resource Utilization” section with data for Stratix® III, Stratix II, and Cyclone® III devices.● Consolidated the full-functionality tables at the end of the design example sections, in Tables 2–2 and 2–3.● Added “Referenced Documents” section.	Updated document for the Quartus II software version 7.1 by adding support for Arria GX and Stratix III devices.
March 2007 v2.2	Removed Table 1-1 and added a bulleted list.	Updated document for the Quartus II software version 7.0 by adding support for Cyclone® III devices.

Date & Document Version	Changes Made	Summary of Changes
December 2006 v2.1	Updated Table to include Stratix III support.	—
August 2006 v2.0	Updated for Quartus II software version 6.0.	—
July 2005 v1.0	Initial release.	—

Referenced Documents

This user guide references the following documents:

- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*
- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Simulation* section in volume 3 of the *Quartus II Handbook*
- *Synthesis* section in volume 1 of the *Quartus II Handbook*

How to Contact Altera

For the most up-to-date information about Altera® products, refer to the following table.








Contacts	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

This document uses the following typographic conventions:

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} , qdesigns 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>.pof 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. For example, 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 (for example, the AHDL keyword SUBDESIGN), as well as logic function names (for example, TRI) are shown in courier font.
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.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information about a particular topic.

