## Contents

### About this User Guide
- Revision History
- Referenced Documents
- How to Contact Altera
- Typographic Conventions

### Chapter 1. About this Megafuction
- Device Family Support
- Introduction
- Features
- General Description
  - Floating Point Formats
    - Single Precision
    - Double Precision
    - Single-Extended Precision
  - Special Case Numbers
- Algorithm for Floating-Point Comparator
- Common Applications
- Resource Utilization and Performance

### Chapter 2. Getting Started
- Software and System Requirements
- MegaWizard Plug-In Manager Customization
- Instantiating Megafuctions in HDL Code or Schematic Designs
  - Generating a Netlist for EDA Tool Use
  - Using the Port and Parameter Definitions
- Identifying a Megafuction after Compilation
- Simulation
  - Quartus II Software Simulation
  - EDA Simulation
- Design Example: Comparison of Single-Precision Signed Numbers
  - Design Files
  - Example
  - Generate the Single-Precision Comparator
  - Implement the Single-Precision Comparator
  - Functional Results—Simulate the Single-Precision Comparator in the ModelSim-Altera
    - Software
    - Understanding the Simulation Results
- Conclusion
Chapter 3. Specifications

Ports and Parameters .......................................................... 3–1
About this User Guide

Revision History

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

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2008 v2.0</td>
<td>Updated the following to support the Quartus® II software version 8.0:</td>
<td>Updated document to support the Quartus II software version 8.0.</td>
</tr>
<tr>
<td></td>
<td>● Resource usage and performance data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Design example description, procedure, and screenshots</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Simulation results for the ModelSim®-Altera software</td>
<td></td>
</tr>
<tr>
<td>October 2007 v1.0</td>
<td>Initial release.</td>
<td>—</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Contact (1)</th>
<th>Contact Method</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td>Website</td>
<td><a href="http://www.altera.com/support">www.altera.com/support</a></td>
</tr>
<tr>
<td>Technical training</td>
<td>Website</td>
<td><a href="http://www.altera.com/training">www.altera.com/training</a></td>
</tr>
<tr>
<td></td>
<td>Email</td>
<td><a href="mailto:custrain@altera.com">custrain@altera.com</a></td>
</tr>
<tr>
<td>Product literature</td>
<td>Website</td>
<td><a href="http://www.altera.com/literature">www.altera.com/literature</a></td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: <strong>Save As</strong> dialog box.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: \qdesigns directory, d: drive, chiptrip.gdf file.</td>
</tr>
<tr>
<td><strong>Italic Type with Initial Capital Letters</strong></td>
<td>Document titles are shown in italic type with initial capital letters. Example: <em>AN 75: High-Speed Board Design</em>.</td>
</tr>
<tr>
<td><strong>Italic type</strong></td>
<td>Internal timing parameters and variables are shown in italic type. Examples: tPIA, n + 1. Variable names are enclosed in angle brackets (&lt; &gt;) and shown in italic type. Example: &lt;file name&gt;, &lt;project name&gt;.pof file.</td>
</tr>
<tr>
<td>Initial Capital Letters</td>
<td>Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.</td>
</tr>
<tr>
<td>“Subheading Title”</td>
<td>References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”</td>
</tr>
<tr>
<td><strong>Courier type</strong></td>
<td>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: \qdesigna \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.</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., etc.</td>
<td>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.</td>
</tr>
<tr>
<td>■ ● </td>
<td>Bullets are used in a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td>✓</td>
<td>The checkmark indicates a procedure that consists of one step only.</td>
</tr>
<tr>
<td>✎</td>
<td>The hand points to information that requires special attention.</td>
</tr>
</tbody>
</table>

---

Note to table:

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

This document uses the typographic conventions shown in the following table.
## About this User Guide

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="caution.png" alt="Caution" /></td>
<td>A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.</td>
</tr>
<tr>
<td><img src="warning.png" alt="Warning" /></td>
<td>A warning calls attention to a condition or possible situation that can cause injury to the user.</td>
</tr>
<tr>
<td><img src="arrow.png" alt="Angled Arrow" /></td>
<td>The angled arrow indicates you should press the Enter key.</td>
</tr>
<tr>
<td><img src="feet.png" alt="Feet" /></td>
<td>The feet direct you to more information about a particular topic.</td>
</tr>
</tbody>
</table>
Chapter 1. About this Megafunction

Device Family Support

The Floating Point Compare (ALTFP_COMPARE) 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 megafonctions that are optimized for Altera device architectures. Using megafonctions instead of coding your own logic saves valuable design time. The Altera-provided functions offer more efficient logic synthesis and device implementation. Scale the megafonction’s size by setting parameters.
Features

The ALTFP_COMPARE megafunction implements comparison functions and offers the following additional features:

- Single precision, double precision, and single-extended precision
- Input support for normal numbers, infinity, zero, denormal numbers, and not-a-number (NaN).
- Seven status output ports:
  - aeb (input A is equal to input B)
  - aneb (input A is not equal to input B)
  - agb (input A is greater than input B)
  - ageb (input A is greater than or equal to input B)
  - alb (input A is less than input B)
  - aleb (input A is less than or equal to input B)
  - unordered (used as an output to flag if one or both input ports are NaN)
- Optional input ports, including asynchronous clear (aclr) and clock enable (clk_en).

General Description

The ALTFP_COMPARE megafunction follows the IEEE-754 standard for floating-point comparison and defines the following:

- The formats for representing floating-point numbers
- The representations of special values (zero, infinity, denormal numbers, and bit combinations that do not represent a number (NaN))

The IEEE-754 standard also defines four formats for floating-point numbers:

- Single precision
- Double precision
- Single-extended precision
- Double-extended precision

The most commonly used floating-point formats are single precision and double precision. This comparison megafunction only supports three formats: single precision, double precision, and single-extended precision.

All floating-point formats have binary patterns as shown in Figure 1–1, where S represents a sign bit, E represents an exponent field, and M represents the mantissa field. For a normal floating-point number, the leading 1 is always implied (for example, binary 1.0011 of decimal 1.1875 is stored as 0011 in the mantissa field). This saves the mantissa field from using an extra bit to represent the leading 1. However, a denormal number does not have an implied leading 1. The left-most bit of the mantissa field can be either 0 or 1.
Floating Point Formats

This section describes the formats for single precision, double precision, and single-extended precision.

Single Precision

In 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 (LSB) to represent the mantissa. As a result, the total width for single precision is 32 bits. The bias for single precision is 127. Refer to Figure 1–2.

Double Precision

In double-precision format, the MSB is a sign bit, followed by 11 intermediate bits to represent an exponent, and 52 LSB to represent the mantissa. As a result, the total width for double precision is 64 bits. The bias for double precision is 1023. Refer to Figure 1–3.
General Description

Single-Extended Precision

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

Special Case Numbers

Table 1–1 shows the special case numbers as defined by the IEEE-754 standard and their data bit representations.

<table>
<thead>
<tr>
<th></th>
<th>Sign Field</th>
<th>Exponent Field</th>
<th>Mantissa Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>Don’t care</td>
<td>All 0’s</td>
<td>All 0’s</td>
</tr>
<tr>
<td>Positive Denormal</td>
<td>0</td>
<td>All 0’s</td>
<td>Non-zero</td>
</tr>
<tr>
<td>Negative Denormal</td>
<td>1</td>
<td>All 0’s</td>
<td>Non-zero</td>
</tr>
<tr>
<td>Positive Infinity</td>
<td>0</td>
<td>All 1’s</td>
<td>All 0’s</td>
</tr>
<tr>
<td>Negative Infinity</td>
<td>1</td>
<td>All 1’s</td>
<td>All 0’s</td>
</tr>
<tr>
<td>Not a Number (NaN)</td>
<td>Don’t care</td>
<td>All 1’s</td>
<td>Non-zero</td>
</tr>
</tbody>
</table>

Algorithm for Floating-Point Comparator

Two floating-point numbers, A and B, are shown in the following equations:

\[
A = (-1)^{S_a} \times 2^{E_a} \times 1.M_a \\
B = (-1)^{S_b} \times 2^{E_b} \times 1.M_b
\]

These equations have the following values:

- $S_a$ and $S_b$ are sign bits
- $E_a$ and $E_b$ are exponent values
- $M_a$ and $M_b$ are mantissa bits

The outputs of the floating-point comparator are obtained from the results of comparing input A and input B using the following equations:
### About this Megafunction

\[
a_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad \Rightarrow \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

\[
ag_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad > \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

\[
al_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad < \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

\[
av_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad \neq \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

\[
ag_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad \geq \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

\[
ale_{eb} = (-1)^{Sa} \times 2^{Ea} \times 1.M_a \quad \leq \quad (-1)^{Sb} \times 2^{Eb} \times 1.M_b \\
\]

unordered = NaN_a || NaN_b

### 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.

The comparison of floating-point numbers is commonly required in scientific calculation applications such as a math coprocessor, embedded arithmetic coprocessor, data processor, data controller, and DSP algorithms.
Resource Utilization and Performance

Tables 1–2, 1–3, and 1–4 provide resource usage and performance information for the ALTFP_COMPARE megafuction.

<table>
<thead>
<tr>
<th>Table 1–2. ALTFP_COMPARE Megafuction Resource Usage for Stratix II and Stratix III Devices Note (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Family</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Stratix II</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Stratix III</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Notes to Table 1–2:
(1) You can get the usage and performance information by compiling your design with the Quartus® II software. The information in this table is valid and accurate in the Quartus II software version 8.0.
(2) Specify the floating-point format on page 1 of the ALTFP_COMPARE megafuction MegaWizard® Plug-In Manager.
(3) Specify the output latency in clock cycles on page 1 of the ALTFP_COMPARE megafuction MegaWizard Plug-In Manager.

<table>
<thead>
<tr>
<th>Table 1–3. ALTFP_COMPARE Megafuction Resource Usage for Cyclone II and Cyclone III Devices Note (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Family</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Cyclone II</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cyclone III</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Notes to Table 1–3:
(1) You can get the usage and performance information by compiling your design with the Quartus II software. The information in this table is valid and accurate in the Quartus II software version 8.0.
(2) Specify the floating-point format on page 1 of the ALTFP_COMPARE megafuction MegaWizard Plug-In Manager.
(3) Specify the output latency in clock cycles on page 1 of the ALTFP_COMPARE megafuction MegaWizard Plug-In Manager.
### Table 1–4. Performance of ALTFP_COMPARE Megafuntion Note (1)

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Precision (2)</th>
<th>Output Latency (3)</th>
<th>f_{MAX} (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix II</td>
<td>Single</td>
<td>3</td>
<td>644</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>3</td>
<td>627</td>
</tr>
<tr>
<td>Stratix III</td>
<td>Single</td>
<td>3</td>
<td>898</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>3</td>
<td>856</td>
</tr>
<tr>
<td>Cyclone II</td>
<td>Single</td>
<td>3</td>
<td>534</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>3</td>
<td>445</td>
</tr>
<tr>
<td>Cyclone III</td>
<td>Single</td>
<td>3</td>
<td>570</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>3</td>
<td>473</td>
</tr>
</tbody>
</table>

Notes to Table 1–4:

1. You can get the usage and performance information by compiling your design with the Quartus II software. The information in this table is valid and accurate in the Quartus II software version 8.0.
2. Specify the floating-point format on page 1 of the ALTFP_COMPARE megafuntion MegaWizard Plug-In Manager.
3. Specify the output latency in clock cycles on page 1 of the ALTFP_COMPARE megafuntion MegaWizard Plug-In Manager.
Chapter 2. Getting Started

Software and System Requirements

The instructions in this section require the following software:

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

MegaWizard Plug-In Manager Customization

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

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

- 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 window, click MegaWizard Plug-In Manager.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: qmegawiz

MegaWizard Plug-In Manager Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALTFP_COMPARE wizard. Click Next on each page to move to the next page.

On page 1 of the MegaWizard Plug-In Manager, you can select Create a new custom megafunction variation, Edit an existing custom megafunction variation, or Copy an existing custom megafunction variation (Figure 2–1).
On page 2a of the MegaWizard Plug-In Manager, you can select the
megafuction, specify the device family to use, the type of output file to
create, and the name of the output file (Figure 2–2). Choose AHDL (.tdf),
VHDL (.vhd), or Verilog HDL (.v) as the output file type.
Getting Started

On page 3 of the ALTFP_COMPARE MegaWizard Plug-In Manager, you can select the type of precision, specify the widths of the input buses, specify the width of the exponent, verify the width of the mantissa, and specify the output latency in clock cycles (Figure 2–3).

Figure 2–3. ALTFP_COMPARE MegaWizard Plug-In Manager [page 3 of 6]
Table 2–1 describes the options available on page 3 of the ALTFP_COMPARE MegaWizard Plug-In Manager.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currently selected device family</td>
<td>Specify the device family you want to use.</td>
</tr>
<tr>
<td>Match project/default</td>
<td>Select this option to ensure that the device selected in 'Currently selected device family' matches the device family chosen on the previous page.</td>
</tr>
<tr>
<td>What is the floating point format?</td>
<td>Select single precision for 32 bits, double precision for 64 bits, and single-extended precision for 43 to 64 bits.</td>
</tr>
<tr>
<td>How wide should the 'dataa' input and 'datab' input buses be?</td>
<td>Specify the width of the buses. The value is predefined for single-precision and double-precision formats. The maximum width is 32 bits for single-precision format and 64 bits for double-precision format. For single-extended precision, you can specify a value between 43 and 64 bits.</td>
</tr>
<tr>
<td>How wide should the exponent field be?</td>
<td>Specify the width of the exponent. The predefined values are 8 bits for single precision and 11 bits for double precision. For single-extended precision, you can specify a value between 11 and 31 bits.</td>
</tr>
<tr>
<td>Mantissa width = (data input width) – (exponent field width) – 1</td>
<td>Verify the width of the mantissa. This is not an option. The value is calculated automatically when the widths of the exponent field and input buses are specified.</td>
</tr>
<tr>
<td>What is the output latency in clock cycles?</td>
<td>Select the latency. The latency value ranges between 1 and 3 clock cycles.</td>
</tr>
</tbody>
</table>
On page 4 of the ALTFP_COMPARE MegaWizard Plug-In Manager, you can select the output ports needed and create optional input ports (Figure 2–4).

Figure 2–4. ALTFP_COMPARE MegaWizard Plug-In Manager [page 4 of 6]
On page 5 of the ALTFPCOMPARE MegaWizard Plug-In Manager, you can choose to generate a synthesis area and timing estimation netlist (Figure 2–5).

Figure 2–5. ALTFPCOMPARE MegaWizard Plug-In Manager [page 5 of 6]
Page 6 of the ALTFP_COMPARE MegaWizard Plug-In Manager displays a list of the types of files to be generated. The automatically generated Variation file contains wrapper code in the language you specified on page 2a. On page 6, you can specify additional types of files to be generated. You can choose from the following types of files:

- AHDL include file (<function name>.inc)
- VHDL component declaration file, <function name>.cmp
- Quartus II symbol file (<function name>.bsf)
- Instantiation template file (<function name>.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_COMPARE MegaWizard Plug-In Manager [page 6 of 6]
Instantiating Megafunctions in HDL Code or Schematic Designs

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

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

- A sample instantiation template for the language of the variation file (_inst.v, _inst.vhd, or _inst.tdf)
- Component Declaration File (.cmp) that can be used in VHDL Design Files
- 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 wizard generates an additional netlist file (_syn.v). The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of the architectural elements in the megafunction but may not represent 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.
Getting Started

Using the Port and Parameter Definitions

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

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

Refer to Chapter 3, Specifications for a list of the megafunction ports and parameters.

Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration is 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 Software 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, 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, the timing simulation in the Quartus II software verifies the operation of your design with annotated timing information. This simulation is performed using the post place-and-route netlist. When performing a timing simulation, add only signals that exist after place-and-route. These signals are found with the Post-Compilation filter of the Node Finder. During synthesis and place-and-route, the names of 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 Simulation

Depending on the 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 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.

Design Example: Comparison of Single-Precision Signed Numbers

This design example uses the ALTFP_COMPARE megafunction to implement a floating-point comparator for the comparison of single-precision signed numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described.

Design Files

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

The design files are also available in the Quartus II Projects section on the Design Examples page at: http://www.altera.com/support/examples/quartus/quartus.html

Example

In this example, the following tasks are performed:

1. Create a floating-point comparator using the ALTFP_COMPARE megafunction and the MegaWizard Plug-In Manager.

2. Assign the EP2S60F484C3 device to the project.

3. Compile and simulate the design.
Generate the Single-Precision Comparator

To generate the single-precision comparator, perform the following steps:

1. Open altfp_compare_DesignExample.zip and extract fp_compare_ex.qar.

2. In the Quartus II software, open fp_compare_ex.qar and restore the archive file into your working directory.


4. Select Create a new custom megafuction variation.

5. Click Next. Page 2a of the MegaWizard Plug-In Manager appears.

6. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 2–2.

<table>
<thead>
<tr>
<th>MegaWizard Plug-In Manager Page</th>
<th>MegaWizard Plug-In Manager Configuration Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>Select a megafuction</td>
<td>ALTFP_COMPARE</td>
</tr>
<tr>
<td></td>
<td>Which device family will you be using?</td>
<td>Stratix II</td>
</tr>
<tr>
<td></td>
<td>Which type of output file do you want to create?</td>
<td>Verilog HDL</td>
</tr>
<tr>
<td></td>
<td>What name do you want for the output file?</td>
<td>fp_compare</td>
</tr>
<tr>
<td>3</td>
<td>Currently selected family</td>
<td>Stratix II</td>
</tr>
<tr>
<td></td>
<td>Match project/default</td>
<td>Selected</td>
</tr>
<tr>
<td></td>
<td>What is the floating point format?</td>
<td>Single precision (32 bits)</td>
</tr>
<tr>
<td></td>
<td>How wide should the ‘dataa’ input and ‘datab’ input buses be?</td>
<td>32 bits</td>
</tr>
<tr>
<td></td>
<td>How wide should the exponent field be?</td>
<td>8 bits</td>
</tr>
<tr>
<td></td>
<td>Mantissa field width</td>
<td>23 bits</td>
</tr>
<tr>
<td></td>
<td>What is the output latency in clock cycles?</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Which outputs do you want?</td>
<td>Select all outputs</td>
</tr>
<tr>
<td></td>
<td>Create an asynchronous clear port</td>
<td>Selected</td>
</tr>
<tr>
<td></td>
<td>Create a clock enable port</td>
<td>Selected</td>
</tr>
<tr>
<td>5</td>
<td>Generate netlist</td>
<td>Not selected</td>
</tr>
</tbody>
</table>
7. Click Finish.

The ALTFP_COMPARE module is now built.

**Implement the Single-Precision Comparator**

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

1. 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 Settings dialog box after these selections are made.

<table>
<thead>
<tr>
<th>MegaWizard Plug-In Manager Page</th>
<th>MegaWizard Plug-In Manager Configuration Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Variation file</td>
<td>Selected</td>
</tr>
<tr>
<td></td>
<td>AHDL include file</td>
<td>Not selected</td>
</tr>
<tr>
<td></td>
<td>VHDL component declaration file</td>
<td>Not selected</td>
</tr>
<tr>
<td></td>
<td>Quartus II symbol file</td>
<td>Not selected</td>
</tr>
<tr>
<td></td>
<td>Instantiation template file</td>
<td>Not selected</td>
</tr>
<tr>
<td></td>
<td>Verilog HDL black box file</td>
<td>Selected</td>
</tr>
</tbody>
</table>
6. Click OK.

7. To compile the design, on the Processing menu, click Start Compilation. Alternatively, you can click the Start Compilation button on the toolbar.

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

The design is now assigned to the EP2S60F484C3 device and is compiled.
Functional Results—Simulate the Single-Precision Comparator in the ModelSim-Altera Software

Next, 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 ModelSim-Altera software support page on the Altera website (www.altera.com). On the support page, there are links to such topics as installation, usage, and troubleshooting.

Set up the ModelSim-Altera software by performing the following steps:

1. Unzip the altfp_compare_ex_msim.zip file to any working directory on your PC.
2. Start the ModelSim-Altera software.
4. Select the folder in which you unzipped the files.
5. Click OK.
7. Select the fp_compare_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 fp_compare_ex.do.

Figure 2–8 shows the expected simulation results in the ModelSim-Altera software.
Understanding the Simulation Results

The design example implements a floating-point comparator for single-precision numbers. Both the optional input ports (clk_en and aclr) and all seven output ports (ageb, aeb, agb, aneb, alb, aleb, and unordered) are enabled.

The output latency chosen is 3, which causes the comparison operation to generate the output result only 3 clock cycles later.

Figure 2–9 shows that after power-up, the aneb, alb, and aleb signals are set on the next rising edge of the clock at 5 ns. The remaining signals (ageb, aeb, agb, and unordered) are not set during this time. At the subsequent rising edge of the clock, the aneb and alb signals are deasserted while the aleb signal remains asserted, followed by the aeb and ageb signals.
Design Example: Comparison of Single-Precision Signed Numbers

Because the output latency is set to 3, the result of the comparison operation is seen only 3 clock cycles after data is available on the input ports. During the third rising edge of the clock at 25 ns, the output ports of ageb, aneb, and agb are asserted as a result of the comparison of the input values. The remaining output ports are low. Refer to Figure 2–10.

Figure 2–10. Floating-Point Comparison at 25 ns

At 375 ns, a denormal number is seen on both the dataa and datab input ports. As denormal inputs are not supported, the data of the input ports are forced to zero before comparison takes place, which results in the dataa value being equal to the datab value. This result causes the aeb, ageb andaleb ports to be asserted. Refer to Figure 2–11.

Figure 2–11. Floating-Point Comparison of Denormal Numbers at 375 ns
Getting Started

At 460 ns, the aclr signal is set for 1 clock cycle. The result of the assertion of the aclr signal is shown in Figure 2–12. The signal patterns are the same as those seen during power-up (Figure 2–3 on page 2–3). The comparison of subsequent data inputs is performed after the aclr signal deasserts.

Figure 2–12. Floating-Point Comparison after Deassertion of aclr Signal at 460 ns

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, divisions, and memory structures. These megafunctions are performance-optimized for Altera devices, and 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.
Chapter 3. Specifications

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 the users of the MegaWizard Plug-In Manager interface.

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

Figure 3–1 shows the ports for the ALTFP_COMPARE megafunction.

![Figure 3–1. ALTFP_COMPARE Megafucntion Ports](image)

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

Table 3–1. ALTFP_COMPARE Megafunction Input Ports

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>Yes</td>
<td>Clock input to the comparator</td>
<td>—</td>
</tr>
<tr>
<td>clk_en</td>
<td>No</td>
<td>Clock enable for the comparator</td>
<td>Allows the comparator megafunction to operate when asserted (HIGH). When deasserted (LOW), no operation takes place and the outputs are unchanged.</td>
</tr>
<tr>
<td>aclr</td>
<td>No</td>
<td>Asynchronous clear for the comparator</td>
<td>The core is asynchronously reset when the aclr signal is asserted (HIGH).</td>
</tr>
<tr>
<td>dataa</td>
<td>Yes</td>
<td>Data input to the comparator</td>
<td>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.</td>
</tr>
<tr>
<td>datab</td>
<td>Yes</td>
<td>Data input to the comparator</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 3–2. ALTFP_COMPARE Megafunction Output Ports

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>aeb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is equal to input B.</td>
</tr>
<tr>
<td>aneb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is not equal to input B.</td>
</tr>
<tr>
<td>agb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is greater than input B.</td>
</tr>
<tr>
<td>ageb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is greater than or equal to input B.</td>
</tr>
<tr>
<td>alb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is less than input B.</td>
</tr>
<tr>
<td>aleb</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted if input A is less than or equal to input B.</td>
</tr>
<tr>
<td>unordered</td>
<td>(1)</td>
<td>Output port for the comparator</td>
<td>Asserted when either one or both of the inputs (the dataa port and/or datab port) are NaN.</td>
</tr>
</tbody>
</table>

Note to Table 3–2: (1) One of these outputs is required; the rest are optional.
Specifications

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH_EXP</td>
<td>Integer</td>
<td>Yes</td>
<td>Defines the precision of the exponent. The bias of the exponent is always set to (2^{\text{WIDTH_EXP}-11}-1) (that is, 127 for single precision and 1023 for double precision). The WIDTH_EXP value must be 8 for single precision and 11 for double precision, or a minimum of 11 for 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. The default value of WIDTH_EXP is 8.</td>
</tr>
<tr>
<td>WIDTH_MAN</td>
<td>Integer</td>
<td>Yes</td>
<td>Defines the precision of the mantissa. The WIDTH_MAN value must be 23 (compliant with the IEEE-754 standard for single-precision floating-point format) when the WIDTH_EXP value is 8. Otherwise, the WIDTH_MAN value must be a minimum of 31. The WIDTH_MAN value must be greater than the WIDTH_EXP value. The sum of WIDTH_EXP and WIDTH_MAN must be less than 64. The default value of WIDTH_MAN is 23.</td>
</tr>
<tr>
<td>PIPELINE</td>
<td>Integer</td>
<td>Yes</td>
<td>Defines the number of pipelines used in the ALTFP_COMPARE megafunction. For the Quartus II software version 8.0, there are three pipeline options—1, 2, and 3.</td>
</tr>
<tr>
<td>DEVICE_FAMILY</td>
<td>String</td>
<td>Yes</td>
<td>Defines the device family setting. Although the ALTFP_COMPARE megafunction is not a family-dependent module, this parameter is required.</td>
</tr>
</tbody>
</table>
Ports and Parameters