Floating Point Square Root (ALTFP_SQRT) Megafunction User Guide

Alterna Corporation

Copyright © 2008 Altera Corporation. All rights reserved. Alterna, 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. Alterna 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. Alterna 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-01026-2.0
Contents

About this User Guide  ................................................................. v
   Revision History ........................................................................................................ v
   How to Contact Altera ................................................................................................ v
   Typographic Conventions ......................................................................................... vi

Chapter 1. About this Megafunction
   Device Family Support ............................................................................................ 1–1
   Introduction ................................................................................................................ 1–1
   Features ..................................................................................................................... 1–1
   General Description ................................................................................................. 1–2
   Common Applications .............................................................................................. 1–4
   Resource Utilization & Performance ...................................................................... 1–4

Chapter 2. Getting Started
   System Requirements .............................................................................................. 2–1
   Mega Wizard Plug-In Manager Customization ..................................................... 2–1
   Using the MegaWizard Plug-In Manager ............................................................. 2–1
   Inferring Megafuctions from HDL Code ............................................................... 2–5
   Instantiating Megafuctions in HDL Code .............................................................. 2–5
   Identifying a Megafuction after Compilation ....................................................... 2–5
   Simulation ................................................................................................................ 2–5
   Quar tus II Simulation ............................................................................................... 2–6
   EDA Simulation .......................................................................................................... 2–6
   SignalTap II Embedded Logic Analyzer ............................................................... 2–6
   Design Example: 9-bit Square Root ........................................................................ 2–7
   Design Files .............................................................................................................. 2–7
   Example .................................................................................................................... 2–7
   Generate a 9-bit Square Root .................................................................................. 2–7
   Implement the Square Root Function ...................................................................... 2–11
   Functional Results—Simulate the 9-bit Square Root Design in Quartus II .......... 2–12
   Functional Results—Simulate the 9-bit Square Root Design in ModelSim-Altera .... 2–13
   Conclusion ................................................................................................................ 2–14

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>
</table>
| May 2008 v2.0             | Updated the following sections to support the Quartus® II software version 8.0 and address ADoQS issue 10001512:  
  ● “Exception Handling” on page 1–6  
  ● “MegaWizard Plug-In Manager Page Descriptions” on page 2–2  
  ● “Design Example 1: Square Root of Single-Precision Numbers” on page 2–11  
  ● “Ports and Parameters” on page 3–1 | Updated document to support the Quartus II software version 8.0 and address ADoQS issue. |
| November 2007 v1.0        | Initial release. | — |

Referenced Documents

This user guide references the following documents:

- Design Debugging Using the SignalTap II Embedded Logic Analyzer chapter in volume 3 of the Quartus II Handbook
- 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
- 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

Typographic Conventions

This document uses the typographic conventions shown in the following table.

<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, : 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><strong>Initial Capital Letters</strong></td>
<td>Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.</td>
</tr>
<tr>
<td><strong>“Subheading Title”</strong></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:\qdesign\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>
<tr>
<td>Visual Cue</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>❞</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>❞</td>
<td>A warning calls attention to a condition or possible situation that can cause injury to the user.</td>
</tr>
<tr>
<td>↘</td>
<td>The angled arrow indicates you should press the Enter key.</td>
</tr>
<tr>
<td>↘</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 Square Root (ALTFP_SQRT) megafunction supports the following target Altera® device families:

- Arria™ GX
- Stratix® IV
- Stratix III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® II
- HardCopy Stratix

Introduction

As design complexities increase, 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. Additionally, the Altera-provided functions may offer more efficient logic synthesis and device implementation. You can scale the size of the megafunction by setting parameters.

Features

The ALTFP_SQRT megafunction implements a floating-point square-root function and offers many additional features, including:

- Square-root operations for single-precision, single-extended, and double-precision numbers
- Status outputs such as not-a-number (NaN), overflow, and zero
- Overflow conditions indicated by overflow outputs

The square-root operation on single- and double-precision floating-point numbers is commonly used for signal processing.
General Description

The ALTFP_SQRT megafunction follows the IEEE-754 standard for floating-point division 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 five exceptions, four rounding modes, and a set of operations that work identically on any conforming system

The IEEE-754 standard also 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. This square-root megafuction only supports three formats: single precision, double precision, and single-extended precision.

All of the floating-point formats are implemented as shown in Figure 1–1. In this figure:

- $S$ represents a sign bit
- $E$ represents an exponent field
- $M$ represents the mantissa (part of a logarithm, or fraction) field

**Figure 1–1. IEEE-754 Floating-Point Format**

<table>
<thead>
<tr>
<th>S</th>
<th>E</th>
<th>M</th>
</tr>
</thead>
</table>

For a normal floating-point number, the leading 1 is always implied (for example, the binary 1.0011 of decimal 1.1875 is stored as 0011 in the mantissa field). This can save the mantissa field from using an extra bit to represent the leading 1.

However, for a denormalized number, the leading bit can be 0 or 1. Therefore, a denormalized number does not have an implied leading 1. For a denormalized number, the left-most bit of the mantissa field is the leading number, either 0 or 1. For zero, infinity, and NaN, there is no implied leading 1 or explicit leading bit for the mantissa field.

This square-root megafuction does not support denormal number inputs. Denormal inputs result in zero values. It is impossible to get denormal results from normal inputs.
Table 1–1 shows the data bit or bits of the sign, exponent, and mantissa fields for floating-point numbers.

<table>
<thead>
<tr>
<th>Precision Mode</th>
<th>Sign Bit (S)</th>
<th>Exponent Bits (E)</th>
<th>Mantissa Bits (M)</th>
<th>Total Bits</th>
<th>Bias of $2^{*(E-1)}-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>1</td>
<td>8</td>
<td>23</td>
<td>32</td>
<td>127</td>
</tr>
<tr>
<td>Double</td>
<td>1</td>
<td>11</td>
<td>52</td>
<td>64</td>
<td>1023</td>
</tr>
<tr>
<td>Single Extended</td>
<td>1</td>
<td>$\geq 11$</td>
<td>$\geq 31$</td>
<td>$\geq 43$</td>
<td>Unspecified. Assume $2^{*(E-1)}-1$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$&lt;$ M Width</td>
<td>$\geq 31$</td>
<td>$\leq 64$</td>
<td>Unspecified. Assume $2^{*(E-1)}-1$</td>
</tr>
</tbody>
</table>

### Single-Precision Floating-Point Format

In single precision, 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 for single precision is 32 bits. The bias for single precision is 127. Refer to Figure 1–2.

**Figure 1–2. Single-Precision Representation**

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>23</th>
<th>22</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
<td>M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### Double-Precision Floating-Point Format

In double precision, the MSB is a sign bit. It is followed by 11 intermediate bits to represent an exponent, and 52 LSBs 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.

**Figure 1–3. Double-Precision Representation**

```
<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>52</th>
<th>51</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
<td>M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
General Description

Single-Extended Precision Floating-Point Format

In single-extended precision, the MSB is a sign bit. However, the exponent and mantissa fields do not have fixed widths. The width of the exponent field must be a minimum of 11 bits and must be less than the width of the mantissa field. The width for the mantissa field must be a minimum of 31 bits. The sum of the width of the sign bit, exponent field, and mantissa field must be 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 this square-root megafunction, a bias of 2^{(\text{WIDTH}_\text{EXP}–1)}–1 is assumed for single-extended precision.

Special Case Numbers

Table 1–2 shows the special case numbers defined by the IEEE-754 standard and the data bit representations.

<table>
<thead>
<tr>
<th>Meaning</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 zeros</td>
<td>All zeros</td>
</tr>
<tr>
<td>Positive Denormalized</td>
<td>0</td>
<td>All zeros</td>
<td>Non-zero</td>
</tr>
<tr>
<td>Negative Denormalized</td>
<td>1</td>
<td>All zeros</td>
<td>Non-zero</td>
</tr>
<tr>
<td>Positive Infinity</td>
<td>0</td>
<td>All ones</td>
<td>All zeros</td>
</tr>
<tr>
<td>Negative Infinity</td>
<td>1</td>
<td>All ones</td>
<td>All zeros</td>
</tr>
<tr>
<td>Not a Number (NaN)</td>
<td>Don't care</td>
<td>All ones</td>
<td>Non-zero</td>
</tr>
</tbody>
</table>

Rounding

In the IEEE-754 standard, there are four types of rounding modes:

- Round-to-nearest-even
- Round-toward-zero
- Round-toward-positive-infinity
- Round-toward-negative-infinity

The most commonly used rounding mode is round-to-nearest-even. This square-root megafunction uses only the round-to-nearest-even mode. In 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 Square Root

Given floating-point input, $A$, as shown in Equation 1:

$$A = (-1)^{S_a} \times 2^{E_a} \times 1.M_a$$

In these equations, the values are:

- $S_a$ is the sign bit
- $E_a$ is the exponent bit
- $M_a$ is the mantissa bit

The result, $R$, from the square-root computation of input $A$ is shown in Equation 2:

$$R = ((-1)^{S_a} \times 2^{E_a} \times 1.M_a)^{1/2}$$

Therefore, the floating-point square root has the following algorithms:

- Sign bit of $R$ = Sign bit of $A$
- Exponent of $R$ = $\frac{(\text{Exponent of } A - \text{bias}) + \text{bias}}{2}$
- Exponent of $R$ = $\frac{(\text{Exponent of } A + \text{bias})}{2}$
- Mantissa of $R$ = $\left(\text{Mantissa of } A\right)^{1/2}$

The exponent of $A$ is stored with bias adjustment in the IEEE-754 floating-point number. Therefore, in the square-root operation, both the exponent $A$ and the bias must be divided by 2. The equation is derived as shown in Equation 3:

$$\frac{\text{Exp}_A \text{ bias}}{2} = \frac{\text{Exp}_A \text{ actual} + \text{bias}}{2}$$

$$\frac{\text{Exp}_A \text{ bias}}{2} = \frac{(\text{Exp}_A \text{ actual})}{2} + \frac{\text{bias}}{2} \quad (\text{lack of half-bias value})$$

$$\frac{\text{Exp}_A \text{ bias}}{2} + \frac{\text{bias}}{2} = \frac{(\text{Exp}_A \text{ actual})}{2} + \frac{\text{bias}}{2} + \frac{\text{bias}}{2}$$
General Description

The division of an even exponent value results in a remainder value that must be adjusted into the mantissa value. The result of the division is then added with a half-bias to re-establish the IEEE format.

Check for exceptions and set the output flags accordingly:

- The nan signal is set when the input is NaN or negative.
- The zero signal is set when the input is zero or denormal.
- The overflow signal is set when the input is infinity.
- When the input is a positive- or negative-denormal number, the result is a positive-zero or negative-zero, respectively.
- When the input is a positive- or negative-zero, the result is a positive-zero or negative-zero, respectively.

Exception Handling

The exceptions for the square-root operation are as follows:

- Set zero signal when the input is a zero, positive-, or negative-denormal number.
- Set overflow signal when the input is positive-infinity.
- Set nan signal when the input is NaN, negative, or negative-infinity.

The exception ports are zero, overflow, and nan. You can instantiate any of these output ports with the MegaWizard® Plug-In Manager. (Figure 2–4 on page 2–5).

You can select the output port to instantiate only when the value of the EXCEPTION_HANDLING parameter is set to YES.

Common Applications

The advantage of using 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.

Calculating the square root of floating-point numbers is commonly required in DSP-based applications, which involve complex calculations based on this basic building block.
About this Megafunction

Resource Usage

Table 1–3 summarizes the resource usage of the ALTFP_SQRT megafunction with full functionality.

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Optimization (2)</th>
<th>Precision (3)</th>
<th>Output Latency (4)</th>
<th>Logic Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ALUTs/LEs</td>
</tr>
<tr>
<td>Stratix III</td>
<td>Balance</td>
<td>Single</td>
<td>28</td>
<td>459</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double</td>
<td>57</td>
<td>1765</td>
</tr>
<tr>
<td>Cyclone III</td>
<td>Balance</td>
<td>Single</td>
<td>28</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double</td>
<td>57</td>
<td>—</td>
</tr>
</tbody>
</table>

Notes to Table 1–3:
(1) You can get the resource information from the MegaWizard Plug-In Manager. The information in this table is valid and accurate in the Quartus II software version 7.2.
(2) Choose a design implementation that balances high performance with minimal logic usage. This setting is available for Cyclone, Cyclone II, Cyclone III, Stratix, Stratix II, and Stratix III devices only. The balanced optimization logic option is set in Analysis and Synthesis Settings on the Assignments menu.
(3) Specify the floating-point format on page 3 of the ALTFP_SQRT megafunction in the MegaWizard Plug-In Manager.
(4) The output latency in clock cycles is determined on page 3 of the ALTFP_SQRT megafunction in the MegaWizard Plug-In Manager.
General Description
Chapter 2. Getting Started

Software and System Requirements

The instructions in this section require the following software:

- For operating system support information, refer to:  
  www.altera.com/support/software/os_support/oss-index.html
- Quartus® II software version 7.2 or later

MegaWizard Plug-In Manager Customization

Use the Altera® MegaWizard® Plug-In Manager to create or modify 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 Square Root (ALT_FP_SQRT) megafunction features in your design.

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

- On the Tools menu, click MegaWizard Plug-In Manager.
- 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
This section provides descriptions of the options available on the individual pages of the ALTFP_SQRT wizard.

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).
Getting Started

On page 2a of the MegaWizard Plug-In Manager, specify the megafunction, 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. The ALTFP_SQRT megafunction appears under the Arithmetic category.

Figure 2–2. ALTFP_SQRT MegaWizard Plug-In Manager [Page 2a]
On page 3 of the ALTFP_SQRT MegaWizard Plug-In Manager, select the floating-point format, specify the input and output widths, and specify the output latency (Figure 2–3).

Figure 2–3. ALTFP_SQRT MegaWizard Plug-In Manager (Page 3 of 6)

Table 2–1 shows the options available on page 3 of the ALTFP_SQRT 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 target device family.</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 ‘data’ input and ‘result’ output buses be?</td>
<td>Specify the width of the buses. This is not an option. The value has been predefined. The maximum width is 32 bits for single precision, and 64 bits for double precision and single-extended precision.</td>
</tr>
</tbody>
</table>
**Getting Started**

On page 4 of the ALTFP_SQRT MegaWizard Plug-In Manager, specify the exception handling and optional inputs (Figure 2–4).

---

### Table 2–1. ALTFP_SQRT MegaWizard Plug-In Manager (Page 3) Options (Part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>How wide should the exponent field be?</td>
<td>Specify the width of the exponent field. This is not an option. The value has been predefined. The maximum width is 8 bits for single precision, 11 bits for double precision, and 31 bits for single-extended precision (depending on the width of the output buses).</td>
</tr>
<tr>
<td>Mantissa width = (data input width) – (exponent field width) –1</td>
<td>This is not an option. The value is automatically calculated when the widths of the exponent field and input buses are specified.</td>
</tr>
<tr>
<td>Output latency in clock cycles</td>
<td>Valid values are ((\text{WIDTH_MAN} + 5)) or (([\text{WIDTH_MAN} + 5] / 2) + 2) with the radix point truncated. For single precision, the latency is fixed at 16 or 28 clock cycles. For double-precision, the latency is fixed at 30 or 57 clock cycles.</td>
</tr>
</tbody>
</table>

---

![Figure 2–4. ALTFP_SQRT MegaWizard Plug-In Manager (Page 4 of 6)](image)
Table 2–2 shows the options available on page 4 of the ALTFP_SQRT MegaWizard Plug-In Manager.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you want to create optional input ports?</td>
<td>Select Create an asynchronous clear port to reset the megafunction asynchronously with the aclr port. Select Create a clock enable port to enable the clock used by the megafunction with the clk_en port.</td>
</tr>
<tr>
<td>Do you want to create optional exception ports?</td>
<td>Select the exception ports for the megafunction design.</td>
</tr>
</tbody>
</table>
Getting Started

On page 5 of the ALTFP_SQRT MegaWizard Plug-In Manager, you can choose to generate a netlist (Figure 2–5).

Figure 2–5. ALTFP_SQRT MegaWizard Plug-In Manager [Page 5 of 6]

On page 6 of the ALTFP_SQRT MegaWizard Plug-In Manager, specify the types of files to be generated. Choose from the following file types:

- 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).
Instantiating Megafunctions in HDL Code

For more information about the ports and parameters for the ALTFP_SQRT megafunction, refer to Chapter 3, Specifications.

When you use the MegaWizard Plug-In Manager to customize and parameterize a megafunction, it creates a set of output files that allow 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.
Getting Started

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
- ADI 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 megafonction 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 megafonction 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 megafonction 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 megafonction, 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 megafonctions in your third-party synthesis tool, refer to the appropriate chapter in the Synthesis section in volume 1 of the Quartus II Handbook.

Using the Port and Parameter Definitions

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

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.

The Quartus II Simulator

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. A functional simulation enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, add only signals that exist before synthesis. You can find these signals 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
Getting Started

and synthesis to keep a particular wire, register, or node intact. Use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation.

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

EDA Simulator

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

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

SignalTap II Embedded Logic Analyzer

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

To monitor signals from Altera megafunctions, configure the SignalTap II Embedded Logic Analyzer in the Quartus II software, and include the analyzer as part of your Quartus II project. The Quartus II software then embeds the analyzer in your design seamlessly in the selected device.

For more information about using the SignalTap II Embedded Logic Analyzer, refer to the Design Debugging Using the SignalTap II Embedded Logic Analyzer chapter in volume 3 of the Quartus II Handbook.

Design Example 1: Square Root of Single-Precision Numbers

This design example uses the MegaWizard Plug-In Manager to configure an ALTFP_SQRT megafunction. The design uses a Stratix III device to implement a square root for single-precision numbers. The ModelSim®-Altera software is used for simulation purposes.

Design Files

The design files are available with this user guide in the Quartus II Project section and in the User Guides section of the Altera website (www.altera.com).
Design Example 1: Square Root of Single-Precision Numbers

Configure ALTFP_SQRT for Single-Precision Square Root

Table 2–3 shows the MegaWizard Plug-In Manager configuration settings that were used to create this design example, a single-precision square root.

<table>
<thead>
<tr>
<th>MegaWizard Configuration Settings</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Family</td>
<td>Stratix III</td>
</tr>
<tr>
<td>Output file type</td>
<td>Verilog HDL</td>
</tr>
<tr>
<td>Floating point format</td>
<td>Single Precision (32 bits)</td>
</tr>
<tr>
<td>‘data’ input and ‘result’ output bus width</td>
<td>32 bits</td>
</tr>
<tr>
<td>Exponent field width</td>
<td>8 bits</td>
</tr>
<tr>
<td>Mantissa width</td>
<td>23 bits</td>
</tr>
<tr>
<td>Output latency in clock cycles</td>
<td>28 clock cycles</td>
</tr>
<tr>
<td>Create an asynchronous clear port</td>
<td>Yes</td>
</tr>
<tr>
<td>Create a clock enable port</td>
<td>Yes</td>
</tr>
<tr>
<td>Create a ‘NaN’ port</td>
<td>Yes</td>
</tr>
<tr>
<td>Create an ‘overflow’ port</td>
<td>Yes</td>
</tr>
<tr>
<td>Create a ‘zero’ port</td>
<td>Yes</td>
</tr>
</tbody>
</table>

You can download the design examples (for the Quartus II software and ModelSim-Altera software) from the Altera website (www.altera.com), and run functional simulations in the following sections.

Functional Simulation in the Quartus II Software

After you have downloaded a design example, you can run the Quartus II functional simulation to display its functional behavior waveform. The examples are already configured and compiled. Set up and run the Quartus II Simulator by performing the following steps:

1. Open the altfp_sqrt_DesignExample.zip project and extract fp_sqrt.qar. In the Quartus II software, open fp_sqrt.qar and restore the archive file into your working directory.

2. On the Processing menu, click Generate Functional Simulation Netlist.

3. When the Functional Simulation Netlist Generation was successful message box appears, click OK.
Getting Started

4. On the Processing menu, click **Start Simulation**, or, on the toolbar, click the **Start Simulation** button.

5. When the **Simulator was successful** message box appears, click **OK**.

6. In the Simulation Report window, view the simulation output waveforms to verify the results (Figure 2–7 and Figure 2–8).

**Figure 2–7. Quartus II Simulation (Input Data)**

<table>
<thead>
<tr>
<th>Name</th>
<th>3 ps</th>
<th>30 ps</th>
<th>60 ps</th>
<th>90 ps</th>
<th>120 ps</th>
<th>150 ps</th>
<th>180 ps</th>
<th>210 ps</th>
<th>240 ps</th>
<th>270 ps</th>
<th>300 ps</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clk_en</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2–8. Quartus II Simulation Results (Output Data)**

<table>
<thead>
<tr>
<th>Name</th>
<th>3 ps</th>
<th>30 ps</th>
<th>60 ps</th>
<th>90 ps</th>
<th>120 ps</th>
<th>150 ps</th>
<th>180 ps</th>
<th>210 ps</th>
<th>240 ps</th>
<th>270 ps</th>
<th>300 ps</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clk_en</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Functional Simulation in the ModelSim-Altera Software**

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 such as installation, usage, and troubleshooting.

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

1. Unzip the `altfp_sqrt_msim.zip` file to any working directory on your PC.
Design Example 1: Square Root of Single-Precision Numbers

2. Start the ModelSim-Altera software.


4. Select the folder in which you unzipped the files.

5. Click OK.


7. Select the fp_sqrt.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_sqrt.do file.

Figure 2–9 and Figure 2–10 show the expected simulation results in the ModelSim-Altera software. For more information about the simulation results, refer to “Understanding the Simulation Results” on page 2–15.

**Figure 2–9. Simulation Waveforms in the ModelSim-Altera Software for Input Data**

**Figure 2–10. Simulation Waveforms in the ModelSim-Altera Software for Output Data**
Getting Started

Understanding the Simulation Results

This design example implements a floating-point square-root function for single-precision numbers with all the exception output ports instantiated. The output ports include overflow, zero, and nan. The output latency is 28 clock cycles. Every square-root computation generates the output result 28 clock cycles later.

For more information about the calculation of the parameter pipeline, refer to “Ports and Parameters” on page 3–1.

At 3.0 μs, the data input is a normal number (Figure 2–11).

Figure 2–11. Floating-Point Square-Root Results—Normal Input at 3 μs

Notes to Figure 2–11:
(1) During the rising edge of the clock at 3 μs, the data input is normal. Computation result is available 28 clock cycles later (Figure 2–12).
(2) The clk_en port is asserted high throughout the simulation.
Design Example 1: Square Root of Single-Precision Numbers

Figure 2–12. Floating-Point Square-Root Results—Normal Output at 84 μs

Notes to Figure 2–12:
(1) The square-root computation of a normal input results in a normal output.
(2) The zero port deasserts to indicate a non-zero result.

The input is a denormal number and zero at 12 μs and 15 μs, respectively (Figure 2–13).

Figure 2–13. Floating-Point Square-Root Results—Zero Input at 15 μs

Note to Figure 2–13:
(1) A denormal input is treated as zero. Computation results are available at 96 μs and 99 μs (Figure 2–14).
Getting Started

Figure 2–14. Floating-Point Square-Root Results—Normal Output at 96 μs

Notes to Figure 2–14:
(1) At 96 μs, the square-root computation of a zero input results in a zero output.
(2) The zero port is asserted high.

The input is infinity at 24 μs (Figure 2–15).

Figure 2–15. Floating-Point Square-Root Results—Infinity Input at 24 μs

Note to Figure 2–15:
(1) The square-root computation of an infinity results in an infinity. Computation results are available at 105 μs (Figure 2–16).
Design Example 1: Square Root of Single-Precision Numbers

Figure 2–16. Floating-Point Square-Root Results—Infinity Input at 105 μs

Note to Figure 2–16:
(1) At 105 μs, the overflow port asserts high to indicate an infinity result.

Truth Table
Table 2–4 shows the truth table for the ALTFP_SQRT megafunction.

Table 2–4. Truth Table

<table>
<thead>
<tr>
<th>DATA[]</th>
<th>SIGN BIT</th>
<th>RESULT[]</th>
<th>NaN</th>
<th>Overflow</th>
<th>Zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Denormal</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Positive Infinity</td>
<td>0</td>
<td>Infinity</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Negative Infinity</td>
<td>1</td>
<td>All One</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Positive NaN</td>
<td>0</td>
<td>All One</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Negative NaN</td>
<td>1</td>
<td>All One</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Zero</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>1</td>
<td>All One</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The Quartus II software provides parameterizable megafuunctions 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, 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.
Chapter 3. Specifications

Ports and Parameters

The Quartus® II software provides the Floating Point Square Root (ALTFP_SQRT) megafunction that supports a floating-point square root function. This chapter describes the ports and parameters of the ALTFP_SQRT megafunction. These ports and parameters are available to customize the ALTFP_SQRT megafunction according to your application.

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 user 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_SQRT megafunction.

Figure 3–1. ALTFP_SQRT Megafonction Ports

![ALTFP_SQRT Ports Diagram]
Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameters for the ALTFP_SQRT megafunction.

**Table 3–1. ALTFP_SQRT 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.</td>
<td>—</td>
</tr>
<tr>
<td>clk_en</td>
<td>No</td>
<td>Clock enable.</td>
<td>Allows square-root operation to take place when asserted high. When de-asserted low, no operation takes place and the outputs are unchanged.</td>
</tr>
<tr>
<td>aclr</td>
<td>No</td>
<td>Asynchronous clear.</td>
<td>The core is asynchronously reset when the aclr signal is asserted high.</td>
</tr>
<tr>
<td>data[]</td>
<td>Yes</td>
<td>Data input.</td>
<td>Input port ([WIDTH_EXP + WIDTH_MAN + 1)...0] wide. The MSB is the sign, the next MSB is the exponent, and the mantissa occupies the LSB.</td>
</tr>
</tbody>
</table>

Table 3–2 lists the output ports of the ALTFP_SQRT megafunction.

**Table 3–2. ALTFP_SQRT Megafunction Output Ports**

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result[]</td>
<td>Yes</td>
<td>Output port ([WIDTH_EXP + WIDTH_MAN + 1)...0] wide. Specifies the floating-point result after rounding. The MSB is the sign, the next MSB is the exponent, and the mantissa occupies the LSB.</td>
</tr>
<tr>
<td>overflow</td>
<td>Yes</td>
<td>Asserted when the result of the square-root operation (after rounding) exceeded or reached infinity. Infinity is defined as a number in which the exponent exceeds (2^WIDTH_EXP−1).</td>
</tr>
<tr>
<td>zero</td>
<td>Yes</td>
<td>Asserted when the value in RESULT[] is zero.</td>
</tr>
<tr>
<td>nan</td>
<td>Yes</td>
<td>Asserted when there is an invalid square-root result, such as a negative number or NaN.</td>
</tr>
</tbody>
</table>
Table 3–3 lists the parameters for the ALTFP_SQRT megafuction.

<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}-1}) (that is, 127 for single precision and 1023 for double precision). The WIDTH_EXP value must be 8 for single precision, 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. If omitted, 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 (in compliance with the IEEE-754 standard for single-precision floating-point numbers) 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>ROUNDING</td>
<td>String</td>
<td>Yes</td>
<td>Defines the rounding mode. The default is TO_NEAREST. Other rounding modes are not supported.</td>
</tr>
<tr>
<td>PIPELINE</td>
<td>Integer</td>
<td>Yes</td>
<td>Defines the number of clock cycles for the square-root result of the result[] port. Values are (\text{WIDTH_MAN} + 5) and (\frac{(\text{WIDTH_MAN} + 5)}{2} + 2) with the radix point truncated.</td>
</tr>
<tr>
<td>DEVICE_FAMILY</td>
<td>String</td>
<td>Yes</td>
<td>Device family setting used for modeling and behavioral simulation. Create the ALTFP_SQRT megafuction with the MegaWizard Plug-in Manager to calculate the value for this parameter. Although the ALTFP_SQRT megafuction is not a family-dependent module, this parameter is required.</td>
</tr>
</tbody>
</table>