Chapter 1. About this Megafunction

Device Family Support ................................................................. 1–1
Introduction ..................................................................................... 1–1
Features ................................................................................................. 1–1
General Description ............................................................................ 1–2
  Floating-Point Formats ................................................................. 1–3
    Single Precision ........................................................................ 1–3
    Double Precision ...................................................................... 1–3
    Single-Extended Precision ..................................................... 1–3
  Special Case Numbers .................................................................. 1–4
  Rounding ......................................................................................... 1–4
  Algorithm for Floating-Point Divider Calculation ...................... 1–4
  Exception Handling ....................................................................... 1–5
  Optimization .................................................................................... 1–6
  Low-Latency Option ...................................................................... 1–6
  Common Applications .................................................................... 1–6
Resource Utilization and Performance ............................................. 1–7

Chapter 2. Getting Started

System Requirements ......................................................................... 2–1
MegaWizard Plug-In Manager Customization .................................. 2–1
MegaWizard Page Descriptions ..................................................... 2–1
Instantiating Megafucntions 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 Software Simulation ................................................. 2–10
  EDA Tool Simulation ................................................................... 2–11
Design Example 1: Division of Single-Precision Numbers ............... 2–11
  Design Files ................................................................................... 2–11
  Example ....................................................................................... 2–11
  Generate the Single-Precision Divider ....................................... 2–12
  Implement the Single-Precision Divider ..................................... 2–13
  Simulate the Single-Precision Divider in the ModelSim-Altera Software .............................................................................................................. 2–15
Understanding the Simulation Results ........................................... 2–16
Design Example 2: Division of Single-Precision Numbers with Low Latency ......................................................... 2–21
  Design Files ................................................................................... 2–21
  Example ....................................................................................... 2–21
  Generate the Single-Precision Divider ....................................... 2–21
  Implement the Single-Precision Divider ..................................... 2–23
Contents

Simulate the Single-Precision Divider in the ModelSim-Altera Software ............................. 2–24
Understanding the Simulation Results ....................................................................................... 2–26
Truth Table .......................................................................................................................... 2–27
Conclusion ............................................................................................................................................ 2–28

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
Chapter 1. About this Megafunction

Device Family Support

The ALTFP_DIV 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 size of the megafunction by setting various parameters.

Features

The ALTFP_DIV megafunction implements division functions and offers many additional features, including:

- Division of single-precision, double-precision, and single-extended precision numbers
- Input support for not-a-number (NaN), normal, infinity, and zero data types
- Status output signals such as nan, normal, infinity, zero, and division_by_zero
- Optional ports including asynchronous clear (aclr) and clock enable (clk_en)
- Support for rounding modes of round-to-nearest-even and round-to-nearest
- Optimization feature to enhance performance in speed or area
General Description

- Optional exception-handling output ports such as zero, overflow, underflow, and nan
- Low-latency implementation

The ALTFP_DIV 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)). However, this megafunction does not support denormal inputs and converts such inputs to zeros.
- The five exceptions and 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 division megafunction 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
- \( M \) is for 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, 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, a denormal number does not have an implied leading 1. The far left 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, 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

```
31 30 23 22 0
S  E  M
```

Double Precision

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

```
63 62 52 51 0
S  E  M
```

Single-Extended Precision

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 less than the width of the mantissa field. The width for the mantissa field must be a minimum of 31 bits. The sum of widths for 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 megafuction, 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 standard and their 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 0’s</td>
<td>All 0’s</td>
</tr>
<tr>
<td>Positive Denormalized</td>
<td>0</td>
<td>All 0’s</td>
<td>Non-zero</td>
</tr>
<tr>
<td>Negative Denormalized</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>

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 megafunction 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 Divider Calculation

The following equations represent two decimal inputs, $A$ and $B$:

\[
(1) \quad A = (-1)^{S_a} \times 2^{E_a} \times M_a
\]

\[
(2) \quad B = (-1)^{S_b} \times 2^{E_b} \times M_b
\]

where:

- $S_a$ and $S_b$ are sign bits
- $E_a$ and $E_b$ are exponent values
- $M_a$ and $M_b$ are mantissa bits
The result, $R$, from the division of inputs $A$ and $B$, is given in the following equation:

$$R = \frac{(-1)^{S_a} \times 2^{E_a} \times 1.M_a}{(-1)^{S_b} \times 2^{E_b} \times 1.M_b}$$

Therefore, the floating-point division has the following algorithm:

1. Sign of $R = \text{Sign bit of } A \oplus \text{Sign bit of } B$
2. Exponent of $R = \text{Exponent of } A - \text{Exponent of } B + \text{Bias}$
3. Mantissa = $\frac{\text{Mantissa of } A}{\text{Mantissa of } B}$

The exponent of $A$ and the exponent of $B$ are stored with bias adjustments in the IEEE-754 floating-point number. Therefore, in the division, the exponent $A$ must subtract exponent $B$ to remove the bias, as shown in Equation 5.

$$\text{Exp}_A\_bias = \text{Exp}_A\_actual + \text{Bias}$$

$$\text{Exp}_B\_bias = \text{Exp}_B\_actual + \text{Bias}$$

$$\text{Exp}_A\_bias - \text{Exp}_B\_bias = (\text{Exp}_A\_actual + \text{Bias}) - (\text{Exp}_B\_actual + \text{Bias})$$

$$= (\text{Exp}_A\_actual - \text{Exp}_B\_actual) + (\text{Bias} - \text{Bias})$$

$$= (\text{Exp}_A\_actual - \text{Exp}_B\_actual)(\text{without Bias})$$

For an IEEE-754 standard floating-point number, only one bias adjustment is needed. Because 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

Five optional exception signals are provided: division_by_zero, nan, overflow, underflow, and zero.

For more details on the exception ports, refer to Table 3–2 on page 3–2.
Optimization

Two optimization options are available for the floating-point divider:

- Optimize for area—If you select this option, the divider is optimized to save resources, but its performance might not be optimal.
- Optimize for speed—If you select this option, additional resources are used to ensure the divider is optimized to improve timing performance.

Low-Latency Option

The low-latency option provides an alternative process with lower latency. With this option, the megafunction takes a shorter time to produce the output, thus requiring a shorter compile time. This option also uses less logic elements.

However, the $f_{\text{MAX}}$ performance of designs using the low-latency implementation is generally lower. Also, this option produces less precise rounding results and involves more DSP usage.

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.

Division of floating-point numbers is commonly required in DSP-based applications, which involve complex calculations based on this basic arithmetic function.
Table 1–2 summarizes the resource usage and performance of the ALTFP_DIV megafunction for Stratix II and Stratix III devices.

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Precision</th>
<th>Optimization</th>
<th>Output Latency (Cycles)</th>
<th>Adaptive Look-Up Tables (ALUTs)</th>
<th>Dedicated Logic Registers (DLRs)</th>
<th>Logic Usage</th>
<th>fMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix II</td>
<td>Single</td>
<td>Speed</td>
<td>33</td>
<td>3379</td>
<td>3131</td>
<td>4200</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>33</td>
<td>1442</td>
<td>1854</td>
<td>2313</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>Speed</td>
<td>61</td>
<td>13,382</td>
<td>12,652</td>
<td>17,750</td>
<td>188</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>4661</td>
<td>6845</td>
<td>8280</td>
<td>166</td>
</tr>
<tr>
<td>Stratix III</td>
<td>Single</td>
<td>Speed</td>
<td>33</td>
<td>3565</td>
<td>3352</td>
<td>4488</td>
<td>279</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>33</td>
<td>1626</td>
<td>2075</td>
<td>2576</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>Speed</td>
<td>61</td>
<td>13,830</td>
<td>13,144</td>
<td>17,677</td>
<td>254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>5120</td>
<td>7361</td>
<td>9055</td>
<td>220</td>
</tr>
</tbody>
</table>

Low-Latency Option

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Precision</th>
<th>Optimization</th>
<th>Output Latency (Cycles)</th>
<th>Adaptive Look-Up Tables (ALUTs)</th>
<th>Dedicated Logic Registers (DLRs)</th>
<th>Logic Usage</th>
<th>fMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix II</td>
<td>Single</td>
<td>—</td>
<td>6</td>
<td>198</td>
<td>270</td>
<td>353</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>—</td>
<td>10</td>
<td>657</td>
<td>1027</td>
<td>1271</td>
<td>124</td>
</tr>
<tr>
<td>Stratix III</td>
<td>Single</td>
<td>—</td>
<td>6</td>
<td>216</td>
<td>305</td>
<td>378</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>—</td>
<td>10</td>
<td>683</td>
<td>1073</td>
<td>1317</td>
<td>142</td>
</tr>
</tbody>
</table>

Note to Table 1–2:
(1) You can get the 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.
Table 1–3 summarizes the resource usage and performance of the ALTFP_DIV megafunction for Cyclone II and Cyclone III devices.

Table 1–3. ALTFP_DIV Resource Usage for Cyclone II and Cyclone III Devices  Note (1)

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Precision</th>
<th>Optimization</th>
<th>Output Latency (Cycles)</th>
<th>Logic Elements (LEs)</th>
<th>Dedicated Logic Registers (DLRs)</th>
<th>Logic Usage</th>
<th>f&lt;sub&gt;MAX&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclone II</td>
<td>Single</td>
<td>Speed</td>
<td>33</td>
<td>4935</td>
<td>3131</td>
<td>4935</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>2892</td>
<td>1854</td>
<td>2892</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>Speed</td>
<td>33</td>
<td>19,690</td>
<td>12,619</td>
<td>19,960</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>10,412</td>
<td>6836</td>
<td>10,412</td>
<td>113</td>
</tr>
<tr>
<td>Cyclone III</td>
<td>Single</td>
<td>Speed</td>
<td>33</td>
<td>4902</td>
<td>3131</td>
<td>4902</td>
<td>151</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>2876</td>
<td>1854</td>
<td>2876</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>Speed</td>
<td>33</td>
<td>19,782</td>
<td>12,619</td>
<td>19,782</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area</td>
<td>61</td>
<td>10,305</td>
<td>6836</td>
<td>10,305</td>
<td>128</td>
</tr>
</tbody>
</table>

Low-Latency Option

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Precision</th>
<th>Optimization</th>
<th>Output Latency (Cycles)</th>
<th>Logic Elements (LEs)</th>
<th>Dedicated Logic Registers (DLRs)</th>
<th>Logic Usage</th>
<th>f&lt;sub&gt;MAX&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclone II</td>
<td>Single</td>
<td>—</td>
<td>6</td>
<td>361</td>
<td>212</td>
<td>361</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>—</td>
<td>10</td>
<td>1495</td>
<td>1006</td>
<td>1495</td>
<td>78</td>
</tr>
<tr>
<td>Cyclone III</td>
<td>Single</td>
<td>—</td>
<td>6</td>
<td>356</td>
<td>212</td>
<td>356</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>—</td>
<td>10</td>
<td>1487</td>
<td>1006</td>
<td>1487</td>
<td>85</td>
</tr>
</tbody>
</table>

Note to Table 1–3:

(1) You can get the 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.
Chapter 2. Getting Started

System Requirements

The instructions in this section require the following software:

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

MegaWizard Plug-In Manager Customization

Use the 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 ALTFP_DIV 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

MegaWizard Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALTFP_DIV 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).
On page 2a of the MegaWizard Plug-In Manager, you can specify the megafuction, the device family to use, the type of output file to create, and the name of the output file (Figure 2–2). The ALTFP_DIV megafuction appears in the Arithmetic category. You can choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type.
On page 3 of the ALTFP_DIV MegaWizard Plug-In Manager, select the floating-point format. Then select the widths of the input and output buses, verify the width of the exponent field, and select the output latency in clock cycles (Figure 2–3).

**Figure 2–3. ALTFP_DIV MegaWizard Plug-In Manager (page 3 of 7)**

![ALTFP_DIV MegaWizard Plug-In Manager](image)

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

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currently selected device family</td>
<td>Specifies the device family you chose on page 2a.</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 in 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>
</tbody>
</table>
How wide should the ‘dataa’ input, ‘datab’ input, and ‘result’ output buses be?

How wide should the exponent field be?

Mantissa width = (data input width) – (exponent field width) – 1

Output latency (clock cycles)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>How wide should the ‘dataa’ input, ‘datab’ input, and ‘result’ output buses be?</td>
<td>Specify the widths of the buses. The value is predefined for single-precision and double-precision formats. The maximum width is 32 bits for single precision, and 64 bits for double precision and single-extended precision. 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 value is predefined for single-precision and double-precision formats. The width is 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. The value is automatically calculated when the widths of the exponent field and input buses are specified.</td>
</tr>
<tr>
<td>Output latency (clock cycles)</td>
<td>Specify the output latency. For single-precision format, the pipeline value is 33 (6 for low-latency option). For double-precision format, the pipeline value is 61 (10 for the low-latency option). For single-extended precision format, the pipeline value ranges from a minimum of 41 to a maximum of 61, while for the low-latency option, the pipeline is determined from the mantissa width. For a mantissa width of between 31 and 40 bits, the pipeline value for the low-latency option is 8. For a mantissa width that is 41 bits or more, the pipeline value for the low-latency option is 10.</td>
</tr>
</tbody>
</table>
On page 4 of the ALTFP_DIV 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_DIV MegaWizard Plug-In Manager [page 4 of 7]**

![ALTFP_DIV MegaWizard Plug-In Manager](image-url)
On page 5 of the ALTFP_DIV MegaWizard Plug-In Manager, you can select the type of optimization to implement (Figure 2–5).

**Figure 2–5. ALTFP_DIV MegaWizard Plug-In Manager [page 5 of 7]**

![MegaWizard Plug-In Manager](image)

On page 6 of the ALTFP_DIV MegaWizard Plug-In Manager, you can choose to generate a synthesis area and timing estimation netlist (Figure 2–6).
Page 7 of the ALTFP_DIV 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 7 of the ALTFP_DIV MegaWizard Plug-In Manager, specify the 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 declaration file (<function name>_bb.v)
If you selected **Generate netlist** on page 6, 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 2–7).

**Figure 2–7. ALTFP_DIV MegaWizard Plug-In Manager (page 7 of 7)**

For more information about the ports and parameters for the ALTFP_DIV megafunction, refer to Chapter 3, Specifications.
Instantiating Megafonctions in HDL Code or Schematic Designs

When you use the MegaWizard Plug-In Manager to customize and parameterize a megafonction, 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 megafonction with the correct parameter values and generates a megafonction 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 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 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 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.

Alterna 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 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
the 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 1: Division of Single-Precision Numbers

This design example uses the ALTFP_DIV megafuction to implement a floating-point divider for the division of single-precision numbers.

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).

Example

In this example, you complete the following tasks:

1. Create a floating-point divider using the ALTFP_DIV megafuction and the MegaWizard Plug-In Manager.
2. Assign the EP2S60F484C3 device to the project.
3. Compile and simulate the design.
Design Example 1: Division of Single-Precision Numbers

Generate the Single-Precision Divider

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

1. Open the altfp_div_DesignExample_ex1.zip file and extract fp_div_ex1.qar.

2. In the Quartus II software, open fp_div_ex1.qar and restore the archive file into your working directory.


4. Select Create a new custom megafunction 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. Click Next to advance from one page to the next.

<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 megafunction</td>
<td>ALTFP_DIV</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_div_ex1</td>
</tr>
<tr>
<td>3</td>
<td>Currently selected device 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, 'datab' input and 'result' output 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 width = (data input width) – (exponent field width) – 1</td>
<td>23 bits</td>
</tr>
<tr>
<td></td>
<td>What is the output latency in clock cycles?</td>
<td>33 bits</td>
</tr>
<tr>
<td>4</td>
<td>Do you want to create optional input ports?</td>
<td>Select all</td>
</tr>
<tr>
<td></td>
<td>Do you want to create optional exception ports?</td>
<td>Select all</td>
</tr>
</tbody>
</table>
7. Click Finish. The ALTFP_DIV module is now built.

**Implement the Single-Precision Divider**

Next, assign the EP2S60F484C3 device to the project and compile the project.

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

2. In the Category list, select Device.

3. In the Family list, select Stratix II.

4. Under Target Device, ensure that Specific device selected in 'Available devices' list is selected.

5. In the Available devices list, select EP2S60F484C3.

6. Leave the rest of the variables at the default settings.

   Figure 2–8 shows the Settings dialog box after you have made these selections.
7. Click OK.

8. To compile the design, on the Processing menu, click **Start Compilation**, or click the **Start Compilation** button on the toolbar.

9. When the **Full Compilation was successful** message box appears, click **OK**. The design is now assigned to the EP2S60F484C3 device and is compiled.
Simulate the Single-Precision Divider in the ModelSim-Altera Software

Simulate the design in the ModelSim®-Altera software to generate a waveform display of the device behavior.

This user guide assumes that you are 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 the ModelSim-Altera software by performing the following steps:

1. Unzip the altfp_div_ex1_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_div_ex1.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_div_ex1.do.

Figures 2–9 and 2–10 show the expected simulation results in the ModelSim-Altera software.

Due to latency at the pipeline registers, the results are not visible immediately when you simulate the design. Figure 2–9 shows the initial output data, and Figure 2–10 shows the output data for each input data after 33 clock cycles.
Understanding the Simulation Results

Design Example 1 implements a floating-point divider for the division of single-precision numbers. All the optional input ports (clk_en and aclr) and output ports (division_by_zero, overflow, underflow, zero, and nan) are enabled.

For this design example, the output latency is set to 33. Every division result appears at the result port only 33 clock cycles later.

For more information about the calculation of the PIPELINE parameter, refer to Table 3–3 on page 3–3.

At 0 μs, the power-up values of the data_a and data_b input ports are assumed to be zero. Thus, during the first clock pulse at 3 μs, the result port displays NaN as the division result of zeros. The hexadecimal representation of NaN is 7FC00000. If both the inputs are zero, the division results in a NaN and therefore, the nan port asserts (Figure 2–11).
At 2.0 μs, the input ports of dataa and datab hold normal values (Figure 2–12).

Because of the 33 clock cycles of latency, the result of the division appears on the result port at approximately 99.0 μs. The division of two normal values results in a normal value and causes the deassertion of the nan port (Figure 2–13).
**Design Example 1: Division of Single-Precision Numbers**

**Figure 2–13. Design Example 1: Division of Normal Values Results in a Normal Output at 99.0 μs**

At 11.0 μs, the input ports of \( \text{dataa} \) and \( \text{datab} \) hold normal values (Figure 2–14).

**Figure 2–14. Design Example 1: Division of Normal Values at 11.0 μs**

After 33 clock cycles, the result of the division is displayed on the result port, at approximately 108.0 μs. The division of these two normal values results in a denormal value. As denormal values are not supported by the ALTFP_DIV megafunction, the result that appears on the result port is zero and the underflow port asserts. The zero port is also asserted to indicate that the result is zero (Figure 2–15).
At 26.0 μs, the input port dataa holds a normal value, while the input port datab holds a denormal value (Figure 2–16).

As denormal numbers are forced-zero values, the division divides a normal value with a zero, which results in an infinity value and causes the assertion of the division_by_zero port at 123.0 μs (Figure 2–17).
Lastly, at 29.0 \(\mu s\), the input port \(data_a\) holds a denormal value, while the input port \(data_b\) holds a normal value (Figure 2–18).

Again, as denormal numbers are forced-zero values, the division of a normal value with a zero produces a zero at 129.0 \(\mu s\) and causes the assertion of the \texttt{zero} port (Figure 2–19).
For more information about the floating-point division and the behavior of the output ports, refer to the “Truth Table” on page 2–27.

Design Example 2: Division of Single-Precision Numbers with Low Latency

This design example uses the ALTFP_DIV megafunction to implement a floating-point divider for the division of single-precision numbers.

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).

Example

In this example, you complete the following tasks:

1. Create a floating-point divider using the ALTFP_DIV 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 Divider

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

1. Open the altfp_div_DesignExample_ex2.zip file and extract fp_div_ex2.qar.
Design Example 2: Division of Single-Precision Numbers with Low Latency

2. In the Quartus II software, open `fp_div_ex2.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 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 2–3**. Click **Next** to advance from one page to the next.

<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_DIV</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><code>fp_div_ex2</code></td>
</tr>
<tr>
<td>3</td>
<td>Currently selected device 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, 'datab' input and 'result' output 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 width = (data input width) – (exponent field width) – 1</td>
<td>23 bits</td>
</tr>
<tr>
<td></td>
<td>What is the output latency in clock cycles?</td>
<td>6 bits</td>
</tr>
<tr>
<td>4</td>
<td>Do you want to create optional input ports?</td>
<td>Select all</td>
</tr>
<tr>
<td></td>
<td>Do you want to create optional exception ports?</td>
<td>Select all</td>
</tr>
<tr>
<td>5</td>
<td>What is the optimization you want to use?</td>
<td>Speed</td>
</tr>
<tr>
<td>6</td>
<td>Generate netlist</td>
<td>Not selected</td>
</tr>
</tbody>
</table>
Getting Started

Table 2–3. Design Example 2: Configuration Settings (Part 2 of 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>7</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>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>

7. Click Finish. The ALTFP_DIV module is now built.

Implement the Single-Precision Divider

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. From the Category list, select Device.

3. In the Family list, select Stratix II.

4. Under Target Device, ensure that Specific device selected in ‘Available devices’ list is selected.

5. In the Available devices list, select EP2S60F484C3.

6. Leave the rest of the variables at the default settings.

Figure 2–20 shows the Device Settings dialog box after you have made these selections.
7. Click OK.

8. To compile the design, on the Processing menu, click **Start Compilation**, or click the **Start Compilation** button on the toolbar.

9. When the **Full Compilation was successful** message box appears, click **OK**. The design is now assigned to the EP2S60F484C3 device and is compiled.

**Simulate the Single-Precision Divider in the ModelSim-Altera Software**

Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.
This user guide assumes that you are 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 the ModelSim-Altera simulator by performing the following steps:

1. Unzip the `altfp_div_ex2_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_div_ex2.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_div_ex2.do`.

Figure 2–21 shows the expected simulation results in the ModelSim-Altera software.

Due to latency at the pipeline registers, the results are not visible immediately when you simulate the design.
Design Example 2: Division of Single-Precision Numbers with Low Latency

Understanding the Simulation Results

Design Example 2 implements a floating-point divider for the division of single-precision numbers with low-latency option.

For single-precision numbers, the output latency is 6. Every division generates the output result 6 clock cycles later.

For more information about the calculation of the PIPELINE parameter, refer to Table 3–3 on page 3–3.

The values seen on the result port and the assertion of several output ports from the first until the sixth clock cycle (0 µs until 17.6 µs) can be ignored as they are just start-up values (Figure 2–22).

For more details on the division results of various types of inputs, refer to “Design Example 1: Division of Single-Precision Numbers” on page 2–11. For more information on floating-point division and the behavior of the output ports, refer to the “Truth Table” on page 2–27.
### Truth Table

Table 2–4 shows the truth table for the ALTFP_DIV megafunction.

<table>
<thead>
<tr>
<th>data []</th>
<th>datab []</th>
<th>Sign Bit</th>
<th>result []</th>
<th>overflow</th>
<th>underflow</th>
<th>zero</th>
<th>division by zero</th>
<th>nan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>0/1</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>0/1</td>
<td>Denormal</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>0/1</td>
<td>Infinity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Denormal</td>
<td>Normal</td>
<td>0/1</td>
<td>Infinity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Denormal</td>
<td>Normal</td>
<td>0/1</td>
<td>Denormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Denormal</td>
<td>Normal</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Denormal</td>
<td>NaN</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Zero</td>
<td>Normal</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Zero</td>
<td>Denormal</td>
<td>0/1</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Zero</td>
<td>Denormal</td>
<td>0/1</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Zero</td>
<td>NaN</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Infinity</td>
<td>Normal</td>
<td>0/1</td>
<td>Infinity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Infinity</td>
<td>Denormal</td>
<td>0/1</td>
<td>Infinity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Infinity</td>
<td>Denormal</td>
<td>0/1</td>
<td>Zero</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Infinity</td>
<td>Denormal</td>
<td>0/1</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NaN</td>
<td>Normal</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NaN</td>
<td>Normal</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NaN</td>
<td>Denormal</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NaN</td>
<td>Denormal</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NaN</td>
<td>Denormal</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NaN</td>
<td>NaN</td>
<td>X</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note to Table 2–4:**

(1) Any calculated or computed denormal output is replaced with a zero. The zero and underflow flags are also asserted.
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 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.
The Quartus® II software provides the ALTFP_DIV megafunction that supports floating-point division. This chapter describes the ports and parameters of the ALTFP_DIV megafunction. These ports and parameters are available to customize the ALTFP_DIV 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_DIV 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_DIV megafunction.

### Table 3–1. ALTFP_DIV 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 divider.</td>
<td>—</td>
</tr>
<tr>
<td>clk_en</td>
<td>No</td>
<td>Clock enable for the divider.</td>
<td>Allows division to take place when clk_en is asserted high. When this signal is low, no division takes place and the outputs remain the same.</td>
</tr>
<tr>
<td>aclr</td>
<td>No</td>
<td>Asynchronous clear for the divider.</td>
<td>The core is asynchronously reset when the aclr signal is asserted high.</td>
</tr>
<tr>
<td>dataa</td>
<td>Yes</td>
<td>Numerator data input to the divider.</td>
<td>The MSB is the sign, the next most significant bits are the exponent, and the mantissa occupies the least significant bits. 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>Denominator data input to the divider.</td>
<td>The MSB is the sign, the next most significant bits are the exponent, and the mantissa occupies the least significant bits. 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 lists the output ports of the ALTFP_DIV megafuction.

### Table 3–2. ALTFP_DIV Megafunction Output Ports (Part 1 of 2)

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>Yes</td>
<td>Output port for the divider.</td>
<td>The division result (after rounding). As with the input values, the MSB is the sign, the next most significant bits are the exponent, and the mantissa occupies the least significant bits. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.</td>
</tr>
<tr>
<td>overflow</td>
<td>No</td>
<td>Overflow port for the divider.</td>
<td>Asserted when the result of the division (after rounding) exceeds or reaches infinity. Infinity is defined as a number in which the exponent exceeds $2^{\text{WIDTH}_\text{EXP}}-1$.</td>
</tr>
<tr>
<td>underflow</td>
<td>No</td>
<td>Underflow port for the divider.</td>
<td>Asserted when the result of the division (after rounding) is zero even though neither of the inputs to the divider is zero, or when the result is a denormalized number.</td>
</tr>
<tr>
<td>zero</td>
<td>No</td>
<td>Zero port for the divider.</td>
<td>Asserted when the value of result is zero.</td>
</tr>
</tbody>
</table>
Specifications

Table 3–3 lists the parameters for the ALTFP_DIV megafuction.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH_EXP</td>
<td>Integer</td>
<td>Yes</td>
<td>Specifies the precision of the exponent. If omitted, the default value for WIDTH_EXP is 8. The bias of the exponent is always set to $2^{(\text{WIDTH}_\text{EXP} - 1)} - 1$ (for example, 127 for single-precision floating-point format and 1023 for double-precision floating-point format). WIDTH_EXP must be 8 for single-precision floating-point format and 11 for double-precision or a minimum of 11 for single-extended precision floating-point format. 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.</td>
<td></td>
</tr>
<tr>
<td>WIDTH_MAN</td>
<td>Integer</td>
<td>Yes</td>
<td>Specifies the precision of the mantissa. If omitted, the default value for WIDTH_MAN 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.</td>
<td></td>
</tr>
<tr>
<td>ROUNINDING</td>
<td>String</td>
<td>Yes</td>
<td>Specifies the rounding mode. The default value is TO_NEAREST. The floating-point divider does not support other rounding modes.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3–3. ALTFP_DIV Megafonction Parameters (Part 2 of 2)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIPELINE</td>
<td>Integer</td>
<td>No</td>
<td>Specifies the number of clock cycles needed to produce the result. You can select a single-precision latency of either 33 (or 6 for low latency), double-precision latency of either 61 (or 10 for low latency), or single-extended precision. For the single-extended precision format, the pipeline value ranges from a minimum of 41 to a maximum of 61 while for the low-latency option, the pipeline is determined from the mantissa width. For a mantissa width of 31 to 40 bits, the pipeline value for the low-latency option is 8. For a mantissa width that is 41 bits or more, the pipeline value for the low-latency option is 10.</td>
</tr>
<tr>
<td>OPTIMIZE</td>
<td>String</td>
<td>No</td>
<td>Specifies the optimization method for the divider. Valid values are SPEED and AREA. If OPTIMIZE is set to SPEED, the megafonction is optimized for speed at the expense of area. If OPTIMIZE is set to AREA, the megafonction is optimized for area at normal speed.</td>
</tr>
</tbody>
</table>
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>
| July 2008 v2.0            | - Added support for Stratix® IV devices.  
- Updated “Features” section.  
- Updated “General Description” section.  
- Updated “Resource Utilization and Performance” section.  
- Removed support for reduced functionality version of the megafunction.  
- Removed support for parameters indefinite and denormal.  
- Removed “Design Example 1: Division for Single-Precision Numbers in Full Functionality Mode” section.  
- Removed “Design Example 2: Division for Single-Precision Numbers in Reduced Functionality Mode” section.  
- Added “Design Example 1: Division of Single-Precision Numbers” section.  
- Added “Design Example 2: Division of Single-Precision Numbers with Low Latency” section.  
- Updated Table 3–3. | Updated document for the Quartus® II software version 8.0 |
| October 2007 v1.2         | Updated for the Quartus II software version 7.2, including:  
- Added support for Arria® GX devices.  
- Updated pipeline value options in “MegaWizard Page Descriptions” and in Table 3–3.  
- Added two new megafunction parameters—EXCEPTION_HANDLING and OPTIMIZE.  
- Removed obsolete megafunction parameter “DECODER_SUPPORT”.  
- Modified the “Resource Utilization and Performance” section with updated data for Stratix III and Stratix II devices.  
- Updated the design example explanations for the current ModelSim-Altera waveforms.  
- Added “Referenced Documents” section.  
- Updated “How to Contact Altera” section. | Updated for the Quartus II software version 7.2 by adding support for Arria GX devices and expanding resource usage information for Stratix III and Stratix II devices. |
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*
- *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/mysupport">www.altera.com/mysupport</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>
<tr>
<td>Non-technical support</td>
<td>Email</td>
<td><a href="mailto:nacomp@altera.com">nacomp@altera.com</a></td>
</tr>
<tr>
<td>(General)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Software Licensing)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
</tbody>
</table>

*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.

<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: ( f_{\text{MAX}} ), <code>\qdesigns</code> directory, <code>d:</code> drive, <code>chiptrip.gdf</code> 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: ( t_{\text{PIA}}, n + 1 ). Variable names are enclosed in angle brackets (&lt; &gt;) and shown in italic type. Example: <code>&lt;file name&gt;, &lt;project name&gt;.pof</code> 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: <code>data1</code>, <code>tdi</code>, <code>input</code>. Active-low signals are denoted by suffix <code>n</code>, e.g., <code>resetn</code>.</td>
</tr>
<tr>
<td><strong>1., 2., 3., and a., b., c., etc.</strong></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><img src="hand.png" alt="Hand Pointing" /></td>
<td>The hand points to information that requires special attention.</td>
</tr>
<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>🔽</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>