Floating-Point Megafunctions

User Guide

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

UG-01058-7.0

Document last updated for Altera Complete Design Suite version: 13.1
Document publication date: November 2013

Subscribe

1. About Floating-Point Megafunctions

This user guide provides information about the Altera® floating-point megafunctions, which enable you to perform floating-point arithmetic in FPGAs through optimized parameterizable functions for Altera device architectures. You can customize the megafunctions by configuring various parameters to accommodate your needs.

Table 1–1 lists the floating-point megafunctions described in this user guide.

Table 1–1. List of Megafunctions

Megafunction Name	Function Overview
Operator Functions	
ALTFP_ADD_SUB	Adder/Subtractor
ALTFP_DIV	Divider
ALTFP_MULT	Multiplier
ALTFP_SQRT	Square Root
Algebraic and Trancendental Functions	
ALTFP_EXP	Exponential
ALTFP_INV	Inverse
ALTFP_INV_SQRT	Inverse Square Root
ALTFP_LOG	Natural Logarithm
Trigonometric Functions	
ALTFP_ATAN	Arctangent
ALTFP_SINCOS	Trigonometric Sine/Cosine
Other Functions	
ALTFP_ABS	Absolute value
ALTFP_COMPARE	Comparator
ALTFP_CONVERT	Converter
ALTERA_FP_ACC_CUSTOM	An Application Specific Accumulator
ALTERA_FP_FUNCTIONS	A Collection of Floating-Point Functions
Complex Functions	
ALTFP_MATRIX_INV	Matrix Inverse
ALTFP_MATRIX_MULT	Matrix Multiplier

This user guide assumes that you are familiar with megafunctions and know how to instantiate and use these megafunctions. If you are unfamiliar with megafunctions, refer to the *Megafunction Overview User Guide*.

Altera also offers the single-precision floating-point option in the FFT MegaCore. For more information about the FFT MegaCore functions, refer to the *FFT MegaCore Function User Guide*.

Device Family Support

All Altera floating-point megafunctions, except for the complex IP megafunctions, support all device families in the Arria®, Cyclone®, and Stratix® series of devices. The complex IP megafunctions, ALTFP_MATRIX_INV and ALTFP_MATRIX_MULT, support Arria II GX, Stratix IV, and Stratix V devices.

General Features

All Altera floating-point megafunctions offer the following features:

- Support for floating-point formats.
- Input support for not-a-number (NaN), infinity, zero, and normal numbers.
- Optional asynchronous input ports including asynchronous clear (`aclr`) and clock enable (`clk_en`).
- Support for round-to-nearest-even rounding mode.
- Compute results of any mathematical operations according to the IEEE-754 standard compliance with a maximum of 1 unit in the last place (u.l.p.) error. This assumption is applied to all floating-point megafunctions excluding complex matrix multiplication and inverse operations (for example, ALTFP_MATRIX_MULTI and ALFP_MATRIX_INV), where a slight increase in errors is observed due to the accumulation of errors during the mathematical operation.

Altera floating-point megafunctions do not support denormal number inputs. If the input is a denormal value, the megafunction forces the value to zero and treats the value as a zero before going through any operation.

IEEE-754 Standard for Floating-Point Arithmetic

The floating-point megafunctions implement the following representations in the IEEE-754 standard:

- Floating-point numbers
- Special values (zero, infinity, denormal numbers, and NaN bit combinations)
- Single-precision, double-precision, and single-extended precision formats for floating-point numbers

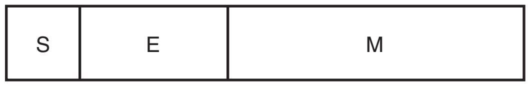
Floating-Point Formats

All floating-point formats have binary patterns. In Figure 1–1, *S* represents a sign bit, *E* represents an exponent field, and *M* is the mantissa (part of a logarithm, or fraction) field.

For a normal floating-point number, a leading 1 is always implied, for example, binary 1.0011 or decimal 1.1875 is stored as 0011 in the mantissa field. This format saves the mantissa field from using an extra bit to represent the leading 1. However, the leading bit for a denormal number can be either 0 or 1. For zero, infinity, and NaN, the mantissa field does not have an implied leading 1 nor any explicit leading bit.

Figure 1–1 shows a floating-point format.

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



Single-Precision Format

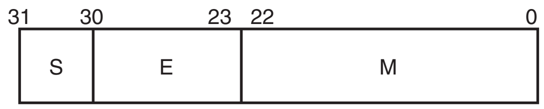
The single-precision format contains the following binary patterns:

- The MSB holds the sign bit.
- The next 8 bits hold the exponent bits.
- 23 LSBs hold the mantissa.

The total width of a floating-point number in the single-precision format is 32 bits. The bias for the single-precision format is 127.

Figure 1–2 shows a single-precision representation.

Figure 1–2. Single-Precision Representation



Double-Precision Format

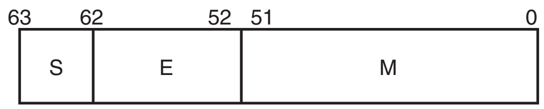
The double-precision format contains the following binary patterns:

- The MSB holds the sign bit.
- The next 11 bits hold the exponent bits.
- 52 LSBs hold the mantissa.

The total width of a floating-point number in the double-precision format is 64 bits. The bias for the double-precision format is 1023.

Figure 1–3 shows a double-precision representation.

Figure 1–3. Double-Precision Representation



Single-Extended Precision Format

The single-extended precision format contains the following binary patterns:

- The MSB holds the sign bit.
- The exponent and mantissa fields do not have fixed widths.
- The minimum exponent field width is 11 bits and must be less than the width of the mantissa field.
- The width of the mantissa field must be a minimum of 31 bits.

The sum of the widths 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 the single-extended precision format is unspecified in the IEEE-754 standard. In these megafunctions, a bias of $2^{(\texttt{WIDTH\_EXP-1})}-1$ is assumed for the single-extended precision format.

Special Case Numbers

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

Table 1–2. Special Case Numbers in IEEE-754 Representation

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

Rounding

The IEEE-754 standard defines four types of rounding modes: round-to-nearest-even, round-toward-zero, round-toward-positive-infinity, and round-toward-negative-infinity. Altera floating-point megafunctions support only the most commonly used rounding mode, which is the round-to-nearest-even mode (`TO_NEAREST`). With round-to-nearest-even, the megafunction rounds the result to the nearest floating-point number. If the result is exactly halfway between two floating-point numbers, the megafunction rounds the result so that the LSB becomes a zero, which is even.

Non-IEEE-754 Standard Format

Only the ALTFP_CONVERT and ALTERA_FP_FUNCTIONS (when the convert function is selected) support the fixed point format.

The fixed-point data type is similar to the conventional integer data type, except that the fixed-point data carries a predetermined number of fractional bits. If the width of the fraction is 0, the data becomes a normal signed integer.

The notation for fixed-point format numbers in this user guide is *Qm.f*, where *Q* designates that the number is in *Q* format notation, *m* is the number of bits used to indicate the integer portion of the number, and *f* is the number of bits used to indicate the fractional portion of the number.

For example, Q4.12 describes a number with 4 integer bits and 12 fractional bits in a 16-bit word.

Figure 1–4 and Figure 1–5 show the difference between the signed-integer format and the fixed-point format for a 32-bit number.

Figure 1–4. Signed-Integer Format

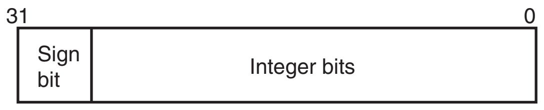
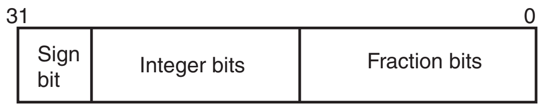


Figure 1–5. Fixed-Point Format



Output Latency

The megafunctions measure the output latency in clock cycles and is different for each megafunction. In some megafunctions, the precision modes determine the number of clock cycles between the input and output result. When you select a mode, the options for latency are fixed for that mode.

For specific details about latency options, refer to the *Output Latency* section of your selected megafunction in this user guide.

Design Flow

Altera strongly recommends that you use the MegaWizard™ Plug-In Manager flow for complex megafunctions. Using the MegaWizard Plug-In Manager flow ensures that you set all megafunction ports and parameters properly.

If you are an expert user, and choose to configure the megafunction directly through parameterized instantiation in your design, refer to the port and parameter details. The details of these ports and parameters are hidden in the MegaWizard Plug-In Manager.

Design Example Files

The design examples for each megafunction in this user guide use the MegaWizard Plug-In Manager in the Quartus II software.

The design example files are available for download from the following locations:

- On the Quartus II Development and Software Literature page, under **Using Megafunctions** in the **Arithmetic** section
- On the Literature: User Guides webpage, with this user guide

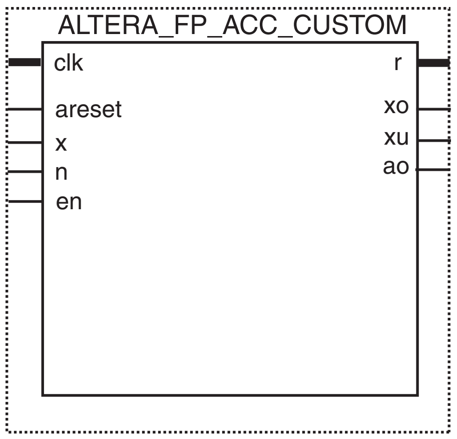
Simulate the designs in the ModelSim®-Altera software to generate a waveform display of the device behavior. You must be familiar with the ModelSim-Altera software before trying out the design examples. If you are unfamiliar with the ModelSim-Altera software, refer to the ModelSim-Altera Software Support page on the Altera website. The support page includes links topics such as installation, usage, and troubleshooting.

For more details, refer to the *Design Example* section of your selected megafunction in this user guide.

2. ALTERA_FP_ACC_CUSTOM

Figure 2–1 shows the ports for the ALTERA_FP_ACC_CUSTOM megafunction.

Figure 2–1. ALTERA_FP_ACC_CUSTOM Ports



Features

The ALTERA_FP_ACC_CUSTOM megafunction offers the following features:

- Supports frequency driven cores.
- Supports VHDL RTL generation.
- Supports customization of the required range of the input and output values.

Output Latency

The amount of latency is driven by the target frequency and the selected device family. You must set the desired frequency and the target device before generating the megafunction. The megafunction reports the latency when you set the parameters and when you generate the megafunction. Then, use the reported latency to incorporate the megafunction into your design.

Resource Utilization and Performance

Table 2–1 lists the resource utilization and performance information for the ALTERA_FP_ACC_CUSTOM megafunction. The information was derived using the Quartus II software version 13.1.

Table 2–1. ALTERA_FP_ACC_CUSTOM Resource Utilization and Performance

Device Family	Input Data		Accumulator Size		Target Frequency (MHz)	Latency	ALMs	DSP Blocks	Logic Registers		M10K	M20K	f_{MAX}
	Floating Point Format	MaxMSBX	MSBA	LSBA					Primary	Secondary			
Arria V *(1)*	double	24	40	-52	270	15	866	0	1,166	106	0	--	265
Cyclone V *(2)*	double	24	40	-52	230	15	830	0	1,102	32	0	--	198
Stratix V *(3)*	double	24	40	-52	400	15	968	0	1,655	27	--	0	426
Arria V *(1)*	single	12	20	-26	270	12	337	0	588	52	0	--	309
Cyclone V *(2)*	single	12	20	-26	230	12	383	0	494	28	0	--	225
Stratix V *(3)*	single	12	20	-26	400	13	475	0	903	20	--	0	450

Notes to Table 2–1:

(1) The device is 5AGXFB3H4F40C5.

(2) The device is 5CGXFC7D6F31C7.

(3) The device is 5SGXEA7K2F40C2.

For more information about Quartus II resource utilization reporting, refer to Fitter Resources Reports in Quartus Help.

Specifications

This section describes the ports and parameters of the ALTERA_FP_ACC_CUSTOM megafunction. The ports and parameters are available to customize the ALTERA_FP_ACC_CUSTOM megafunction according to your application.

Ports and Parameters

Table 2–2 lists the input ports of the ALTERA_FP_ACC_CUSTOM megafunction.

Table 2–2. ALTERA_FP_ACC_CUSTOM Megafunction Input Ports

Port Name	Required	Description
`clk`	Yes	All input signals, otherwise explicitly stated, must be synchronous to this clock
`areset`	Yes	Asynchronous active-high reset. Deassert this signal synchronously to the input clock to avoid metastability issues.
`en`	No	Global enable signal. This port is optional.
`x`	Yes	Data input port.
`n`	Yes	Boolean port which signals the beginning of a new data set to be accumulated. This should go high together with the first element in the new data set and should go low the next cycle. The data sets may be of variable length and a new data set may be started at any time. The accumulation result for an input will be available after the reported latency.

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

Table 2–3. ALTERA_FP_ACC_CUSTOM Megafunction Output Ports

Port Name	Required	Description
`r`	Yes	The running value of the accumulation.
`xo`	Yes	The overflow flag for port `x`. The signal goes high when the exponent of the input `x` is larger than maxMSBX. The signal remains high for the entire data set. This flag invalidates port `r`. You should consider increasing maxMSBX. This flag also indicate infinity and NaN.
`xu`	Yes	The underflow flag for port `x`. The signal goes high when the exponent of the input `x` is smaller than LSBA. The signal remains high for the entire data set. This flag does not invalidate port `r`. You should consider lowering LSBA.
`ao`	Yes	The overflow flag for Accumulator. The signal goes high when the exponent of the accumulated value is larger than MSBA. The signal remains high for the entire data set. This flag invalidates port `r`. You should consider increasing MSBA.

Table 2–4 lists the parameters of the ALTERA_FP_ACC_CUSTOM megafunction.

Table 2–4. ALTERA_FP_ACC_CUSTOM Megafunction Parameters

Category	Parameter	Values	Description
Input Data	Floating point format	**single**, **double**	Choose the floating point format of the input data values. The output data values of the accumulator is in the same format. The default is **single**.
	maxMSBX	—	The maximum weight of the MSB of an input. For example, when adding probabilities in the 0 to 1 range set this weight to ceil($\log_2(1)$)=0. The `xo` output signal goes high when the MSB of an input value has a weight larger than maxMSBX. The result of the accumulation is then invalid. If you are unsure about the range of the inputs, then set the **maxMSBX** parameter to MSBA, at the possible expense of increased resource usage. The default value is **12**.
Accumulator Size	MSBA	—	The weight of the MSB of the accumulator. For example, in a financial simulation, if the value of a stock cannot exceed 100,000 dollars, use a value of ceil($\log_2(100000)$)=17. In a circuit simulation where the circuit adds numbers in the 0 to 1 range, for one year, at 400 MHz, use a value of ceil($\log_2(365 \times 60 \times 60 \times 24 \times 400 \times 10^6)$)=54. The `ao` output signal goes high when the MSB of the accumulated value has a weight larger than MSBA. The result of the accumulation is then invalid. Altera recommends adding a few guard bits to avoid possible accumulator overflow. A few guard bits have little impact on the accumulator size. The default value is **20**.
	LSBA	—	The weight of the LSB of the accumulator and the accuracy of the accumulator. Because an N term accumulation can invalidate the $\log_2(N)$ LSBs of the accumulator, you must consider the length of the accumulation and the range of the inputs when setting this parameter. For example, if a 2^{-30} accuracy is required over an accumulation of 1024 numbers, then set the LSBA to: $(-30 - \log_2(1024)) = -40$. Any input $2^e \times 1.F$, where F is the mantissa and e is less than the LSBA will be shifted out of the accumulator. The `au` output signal goes high to indicate this situation. The default value is **-26**.
Required Performance	Target frequency	Any positive integer value.	Choose the frequency in MHz at which this core is expected to run. This together with the target device family will determine the amount of pipelining in the core. The default value is **200** MHz.

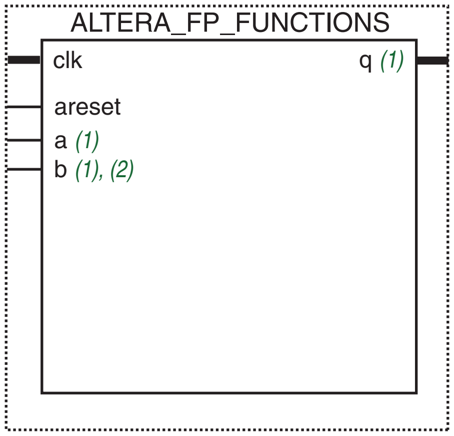
Table 2–4. ALTERA_FP_ACC_CUSTOM Megafunction Parameters

Category	Parameter	Values	Description
Optional	Generate an enable port	—	Choose if the accumulator should have an enable signal. This parameter is disabled by default.
Report	—	—	Reports the latency of the device, which is the number of cycles it takes for an accumulation to propagate through the block from input to output.

3. ALTERA_FP_FUNCTIONS

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

Figure 3–1. ALTERA_FP_FUNCTIONS Ports



Notes to Figure 3–1:

(1) The floating point and fixed point data widths determine the port width of this port.

(2) This port is not relevant for convert and square root functions.

Features

The ALTERA_FP_FUNCTIONS megafunction offers the following features:

- Supports both latency and frequency driven cores.
- Supports VHDL code generation.
- Supports the following functions:
  - Add
  - Compare
  - Convert
  - Max
  - Min
  - Multiply
  - Square Root
  - Subtract

Output Latency

If you require a specific latency, follow these steps:

1. In the MegaWizard Plug-in Manager for the ALTERA_FP_FUNCTIONS megafunction, click **Basic** tab.
2. Under the Performance category, in the **Goal** option, select **latency**.
3. In the **Target** field, set your desired latency (cycles).
4. Then, click **Check Performance**.

Target Frequency

If you require a specific frequency, follow these steps:

1. In the MegaWizard Plug-in Manager for the ALTERA_FP_FUNCTIONS megafunction, click **Basic** tab.
2. Under the Performance category, in the **Goal** option, select **frequency**.
3. In the **Target** field, set your desired frequency (MHz).
4. The megafunction reports the latency for the instance that it will generate in the Report category.

☞ You must verify the frequency by running the TimeQuest Timing Analyzer.

Resource Utilization and Performance

Table 3–1 lists the resource utilization and performance information for the ALTERA_FP_FUNCTIONS megafunction. The information was derived using the Quartus II software version 13.1.

Table 3–1. ALTERA_FP_FUNCTIONS Resource Utilization and Performance

Device Family	Function	Precision	Target Latency	ALMs	DSP Blocks	Logic Registers		M10k	M20k	f_{MAX}
						Primary	Secondary			
Arria V (1)	Add	Double	14	847	0	1,352	139	0	—	211
			7	844	0	723	34	0	—	144
		Single	14	438	0	767	30	0	—	278
			7	380	0	438	22	0	—	194
	Compare EQ	Double	3	121	0	118	0	0	—	416
		Single	3	62	0	60	0	0	—	570
	Compare LT	Double	3	173	0	139	8	0	—	343
		Single	3	88	0	65	0	0	—	423
	Convert	16.16 to Double	6	180	0	220	11	0	—	240
		16.16 to Single	6	163	0	199	8	0	—	223
		32.32 to Double	6	343	0	386	30	0	—	215
		32.32 to Single	6	239	0	274	9	0	—	208
		Double to 16.16	6	173	0	219	3	0	—	387
		Double to 32.32	6	316	0	396	17	0	—	339
		Double to int32	6	173	0	219	3	0	—	387
		Double to int64	6	316	0	396	17	0	—	339
		int32 to Double	6	180	0	220	11	0	—	240
		int32 to Single	6	163	0	199	8	0	—	223
		int64 to Double	6	343	0	386	30	0	—	215
		int64 to Single	6	239	0	274	9	0	—	208
		Single to 16.16	6	150	0	211	3	0	—	386
		Single to 32.32	6	286	0	370	13	0	—	347

Table 3–1. ALTERA_FP_FUNCTIONS Resource Utilization and Performance

Device Family	Function	Precision	Target Latency	ALMs	DSP Blocks	Logic Registers		M10k	M20k	f_{MAX}
						Primary	Secondary			
Arria V *(1)*	Convert	Single to int32	6	150	0	211	3	0	—	386
		Single to int64	6	286	0	370	13	0	—	347
	Max	Double	2	273	0	390	1	0	—	549
		Single	2	143	0	169	1	0	—	369
	Min	Double	2	273	0	390	0	0	—	540
		Single	2	140	0	169	0	0	—	416
	Multiply	Double	11	605	4	1,239	2	0	—	250
			5	357	4	363	10	0	—	195
		Single	11	218	1	468	7	0	—	310
			5	159	1	194	1	0	—	232
	Square Root	Double	29	890	14	1,802	72	8	—	210
		Single	12	192	2	301	60	3	—	310
	Subtract	Double	14	823	0	1,361	111	0	—	212
			7	854	0	750	35	0	—	156
		Single	14	445	0	760	28	0	—	275
			7	385	0	433	31	0	—	201
Cyclone V *(2)*	Add	Double	14	824	0	1,432	45	0	—	165
			7	836	0	736	34	0	—	104
		Single	14	456	0	767	19	0	—	209
			7	378	0	440	14	0	—	163
	Compare EQ	Double	3	122	0	118	0	0	—	333
		Single	3	65	0	60	0	0	—	455
	Compare LT	Double	3	171	0	133	0	0	—	327
		Single	3	87	0	65	0	0	—	359
	Convert	16.16 to Double	6	176	0	220	13	0	—	196
		16.16 to Single	6	167	0	196	11	0	—	182
		32.32 to Double	6	345	0	388	10	0	—	157
		32.32 to Single	6	240	0	269	18	0	—	179
		double to 16.16	6	165	0	216	2	0	—	293
		Double to 32.32	6	309	0	394	10	0	—	251
		Double to int32	6	165	0	216	2	0	—	293

Table 3–1. ALTERA_FP_FUNCTIONS Resource Utilization and Performance

Device Family	Function	Precision	Target Latency	ALMs	DSP Blocks	Logic Registers		M10k	M20k	f_{MAX}
						Primary	Secondary			
Cyclone V (2)	Convert	Double to int64	6	309	0	394	10	0	—	251
		int32 to Double	6	176	0	220	13	0	—	196
		int32 to Single	6	167	0	196	11	0	—	182
		int64 to Double	6	345	0	388	10	0	—	157
		int64 to Single	6	240	0	269	18	0	—	179
		Single to 16.16	6	144	0	208	2	0	—	300
		Single to 32.32	6	278	0	364	9	0	—	258
		Single to int32	6	144	0	208	2	0	—	300
		Single to int64	6	278	0	364	9	0	—	258
	Max	Double	2	274	0	390	0	0	—	650
		Single	2	142	0	169	0	0	—	358
	Min	Double	2	273	0	390	0	0	—	650
		Single	2	141	0	169	1	0	—	360
	Multiply	Double	11	499	4	1,024	30	0	—	218
			5	370	4	396	3	0	—	158
		Single	11	217	1	469	2	0	—	303
			5	154	1	166	2	0	—	223
	Square Root	Double	29	902	14	1,742	57	8	—	187
		Single	12	183	2	305	41	3	—	275
	Subtract	Double	14	807	0	1,414	55	0	—	154
			7	827	0	729	28	0	—	112
		Single	14	451	0	760	28	0	—	220
			7	377	0	425	33	0	—	163

Table 3–1. ALTERA_FP_FUNCTIONS Resource Utilization and Performance

Device Family	Function	Precision	Target Latency	ALMs	DSP Blocks	Logic Registers		M10k	M20k	f_{MAX}
						Primary	Secondary			
Stratix V (3)	Add	Double	14	912	0	1,625	60	—	0	387
			7	759	0	938	27	—	0	354
		Single	14	481	0	976	44	—	0	450
			7	372	0	534	13	—	0	429
	Compare EQ	Double	3	122	0	118	0	—	0	717
		Single	3	61	0	60	0	—	0	717
	Compare LT	Double	3	165	0	57	0	—	0	717
		Single	3	91	0	31	0	—	0	717
	Convert	16.16 to Double	6	185	0	234	19	—	0	550
		16.16 to Single	6	159	0	210	7	—	0	530
		32.32 to Double	6	347	0	391	34	—	0	366
		32.32 to Single	6	233	0	276	14	—	0	430
		Double to 16.16	6	235	0	371	17	—	0	717
		Double to 32.32	6	396	0	722	14	—	0	612
		Double to int32	6	235	0	371	17	—	0	717
		Double to int64	6	396	0	722	14	—	0	612
		int32 to Double	6	185	0	234	19	—	0	550
		int32 to Single	6	159	0	210	7	—	0	530
	Convert	int64 to Double	6	347	0	391	34	—	0	366
		int64 to Single	6	233	0	276	14	—	0	430
		Single to 16.16	6	196	0	334	58	—	0	657
		Single to 32.32	6	339	0	659	7	—	0	592
		Single to int32	6	196	0	334	58	—	0	657
		Single to int64	6	339	0	659	7	—	0	592
	Max	Double	2	318	0	394	0	—	0	717
		Single	2	139	0	171	2	—	0	717

Table 3–1. ALTERA_FP_FUNCTIONS Resource Utilization and Performance

Device Family	Function	Precision	Target Latency	ALMs	DSP Blocks	Logic Registers		M10k	M20k	f_{MAX}
						Primary	Secondary			
Stratix V *(3)*	Min	Double	2	319	0	394	1	—	0	717
		Single	2	138	0	171	0	—	0	717
	Multiply	Double	11	482	4	1,025	37	—	0	391
		Double	5	310	4	378	16	—	0	278
		Single	11	230	1	504	10	—	0	500
		Single	5	156	1	186	2	—	0	403
	Square Root	Double	29	839	10	1,739	62	—	8	438
		Single	12	204	2	391	3	—	3	450
	Subtract	Double	14	922	0	1,507	124	—	0	418
			7	773	0	948	20	—	0	311
		Single	14	469	0	920	77	—	0	450
			7	373	0	536	12	—	0	419

Notes to Table 3–1:

(1) The device is 5AGXFB3H4F40C5.
(2) The device is 5CGXFC7D6F31C7.
(3) The device is 5SGXEA7K2F40C2.

For more information about Quartus II resource utilization reporting, refer to Fitter Resources Reports in Quartus Help.

Specifications

This section describes the ports and parameters of the ALTERA_FP_FUNCTIONS megafunction. The ports and parameters are available to customize the ALTERA_FP_FUNCTIONS megafunction according to your application.

Ports and Parameters

Table 3–2 lists the input ports of the ALTERA_FP_FUNCTIONS megafunction.

Table 3–2. ALTERA_FP_FUNCTIONS Megafunction Input Ports

Port Name	Required	Description
`clk`	Yes	All input signals must be synchronous to this clock.
`areset`	Yes	Asynchronous active-high reset. Deassert this signal synchronously to the input clock to avoid metastability issues.
`en`	No	Global enable signal. This port is optional.
`a`	Yes	Data input port.
`b`	Yes	Data input port (where applicable).

Table 3–3 lists the output ports of the ALTERA_FP_FUNCTIONS megafunction.

Table 3–3. ALTERA_FP_FUNCTIONS Megafunction Output Ports

Port Name	Required	Description
`q`	Yes	Data output port.

Table 3–4 lists the parameters of the ALTERA_FP_FUNCTIONS megafunction.

Table 3–4. ALTERA_FP_FUNCTIONS Megafunction Parameters

Tab	Category	Parameter	Values	Description
Basic	Function	Name	■ Add ■ Compare ■ Convert ■ Max ■ Min ■ Multiply ■ Square Root ■ Subtract.	Allows you to choose your desired function. The default value is **Add**.
		Type	■ Fixed to Floating ■ Floating to Fixed ■ LT ■ LE ■ EQ ■ GE ■ GT ■ NEQ	Allows you to choose the type of comparator or conversion. This parameter is only available when the **Name** parameter is set to either **Compare** or **Convert**. When you set the **Name** parameter to **Convert**, the values are: ■ **Fixed to Floating** (default) ■ **Floating to Fixed** When you set the **Name** parameter to **Compare**, the values are: ■ **LT** (Less than) (default) ■ **LE** (Less than or equals) ■ **EQ** (Equals) ■ **GE** (Greater than or equals) ■ **GT** (Greater than) ■ **NEQ** (Not equals)

Table 3–4. ALTERA_FP_FUNCTIONS Megafunction Parameters

Tab	Category	Parameter	Values	Description
Basic	Floating Point Data	Format	■ **single** ■ **double** ■ **custom**	Allows you to choose the floating point format of the data values. The default value is **single**.
		Exponent	5 to 8	Allows you to specify the width of the exponent. This parameter is only available when the **Format** parameter is set to **custom**. The default value is 8.
		Mantissa	10 to 52	Allows you to specify the width of the mantissa. This parameter is only available when the **Format** parameter is set to **custom**. The default value is 23.
	Fixed Point Data	Width	16 to 128	The bit width of the fixed point data port. This parameter is only available when the **Name** parameter is set to **Convert**.
		Fraction	0-fixed_point_data_width	The bit width of the fraction. This parameter is only available when the **Name** parameter is set to **Convert**.
		Sign	■ signed ■ unsigned	Chose if the fixed point data is signed or unsigned. This parameter is only available when the **Name** parameter is set to **Convert**. The default value is **signed**.
	Performance	Goal	■ frequency ■ latency	Allows you to specify the performance goal. The default value is **frequency**.
		Target	Any positive integer value.	If the **Goal** is the frequency, then the **Target** is the desired frequency in MHz. This, together with the target device family, will determine the amount of pipelining. If the **Goal** is latency, then the **Target** is the desired latency. When you set the **Goal** parameter to **frequency**, the default value is 200 MHz When you set the **Goal** parameter to **latency**, the default value is 2.
	Report	Report	—	If the **Goal** is the frequency, then the latency of the megafunction will be reported. If the **Goal** is the latency, then the **Check Performance** button will appear. If the requested latency is achievable, then the megafunction will report the estimated frequency when you click **Check Performance**. Otherwise, the nearest achievable latency will be reported.

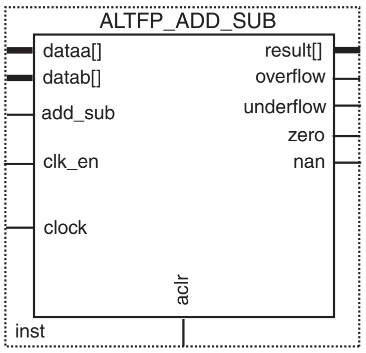
Table 3–4. ALTERA_FP_FUNCTIONS Megafunction Parameters

Tab	Category	Parameter	Values	Description
Advanced	Rounding	Mode	■ nearest with tie breaking to even ■ toward zero ■ nearest with tie breaking away from zero	The rounding mode. The **nearest with tie breaking away from zero** and **toward zero** values are only available when you set the Name parameter to **Convert** and the Type parameter to **Floating to Fixed**. The default value is **nearest with tie breaking to even.**
		Relax rounding to round up or down to reduce resource usage	—	Choose if the nearest rounding mode should be relaxed to faithful rounding, where the result may be rounded up or down, to reduce resource usage.
	Ports	Generate an enable port	—	Choose if the ALTERA_FP_FUNCTION megafunction should have an enable signal.

4. ALTFP_ADD_SUB

Figure 4–1 shows the ports for the ALTFP_ADD_SUB megafunction.

Figure 4–1. ALTFP_ADD_SUB Ports



Features

The ALTFP_ADD_SUB megafunction offers the following features:

- Dynamically configurable adder and subtracter functions.
- Optional exception handling output ports such as `zero`, `overflow`, `underflow`, and `nan`.
- Optimization of speed and area.

Output Latency

The output latency options for the ALTFP_ADD_SUB megafunction are the same for all three precision formats—single, double, and single-extended. The options available are 7, 8, 9, 10, 11, 12, 13, and 14 clock cycles.

Truth Table

Figure 4–2 lists the truth table for the addition/subtraction operations.

Figure 4–2. Truth Table for Addition/Subtraction Operations

DATAA[]	DATAB[]	SIGN BIT	RESULT[]	Overflow	Underflow	Zero	NaN
Normal	Normal	0	Zero	0	0	1	0
Normal	Normal	0/1	Normal	0	0	0	0
Normal	Normal	0/1	Denormal	0	1	1	0
Normal	Normal	0/1	Infinity	1	0	0	0
Normal	Denormal	0/1	Normal	0	0	0	0
Normal	Zero	0/1	Normal	0	0	0	0
Normal	Infinity	0/1	Infinity	1	0	0	0
Normal	NaN	X	NaN	0	0	0	1
Denormal	Normal	0/1	Normal	0	0	0	0
Denormal	Denormal	0/1	Normal	0	0	0	0
Denormal	Zero	0/1	Zero	0	0	1	0
Denormal	Infinity	0/1	Infinity	1	0	0	0
Denormal	NaN	X	NaN	0	0	0	1
Zero	Normal	0/1	Normal	0	0	0	0
Zero	Denormal	0/1	Zero	0	0	1	0
Zero	Zero	0/1	Zero	0	0	1	0
Zero	Infinity	0/1	Infinity	1	0	0	0
Zero	NaN	X	NaN	0	0	0	1
Infinity	Normal	0/1	Infinity	1	0	0	0
Infinity	Denormal	0/1	Infinity	1	0	0	0
Infinity	Zero	0/1	Infinity	1	0	0	0
Infinity	Infinity	0/1	Infinity	1	0	0	0
Infinity	NaN	X	NaN	0	0	0	1
NaN	Normal	X	NaN	0	0	0	1
NaN	Denormal	X	NaN	0	0	0	1
NaN	Zero	X	NaN	0	0	0	1
NaN	Infinity	X	NaN	0	0	0	1
NaN	NaN	X	NaN	0	0	0	1

Resource Utilization and Performance

Table 4–1 and Table 4–2 list the resource utilization and performance information for the ALTFP_ADD_SUB megafunction. The information was derived using the Quartus II software version 10.0.

Table 4–1. ALTFP_ADD_SUB Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Optimization	Output latency	Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	f_{MAX} (MHz)
Stratix III	single	speed	7	592	376	386	218
			14	688	727	509	442
		area	7	576	345	372	218
			14	599	603	427	416
	double	speed	7	1,193	687	821	180
			14	996	1,607	1,085	345
		area	7	1,104	630	765	182
			14	903	1,518	1,013	212
Stratix IV	single	speed	7	594	376	385	228
			14	674	686	498	495
		area	7	576	345	375	227
			14	596	603	421	484
	double	speed	7	1,198	687	824	187
			14	997	1,607	1,080	398
		area	7	1,106	630	762	189
			14	904	1,518	1,013	265

Table 4–2. ALTFP_ADD_SUB Resource Utilization and Performance for the Cyclone Series of Devices

Device Family	Precision	Optimization	Output latency	Logic Elements (LEs)	Dedicated Logic Registers (DLRs)	Logic Usage	f_{MAX} (MHz)
Cyclone II	single	speed	7	831	346	831	153
		area	7	785	317	785	143
	double	speed	7	1,764	648	1,764	119
		area	7	1,656	591	1,656	101
Cyclone III	single	speed	7	822	346	822	154
		area	7	775	317	775	149
	double	speed	7	1,743	648	1,743	128
		area	7	1,631	591	1,631	116

Design Example: Addition of Double-Precision Format Numbers

This design example uses the ALTFP_ADD_SUB megafunction to perform the addition of double-precision format numbers using the MegaWizard Plug-In Manager flow in the Quartus II software.

Design Files

The following files are related to the ALTFP_ADD_SUB megafunction:

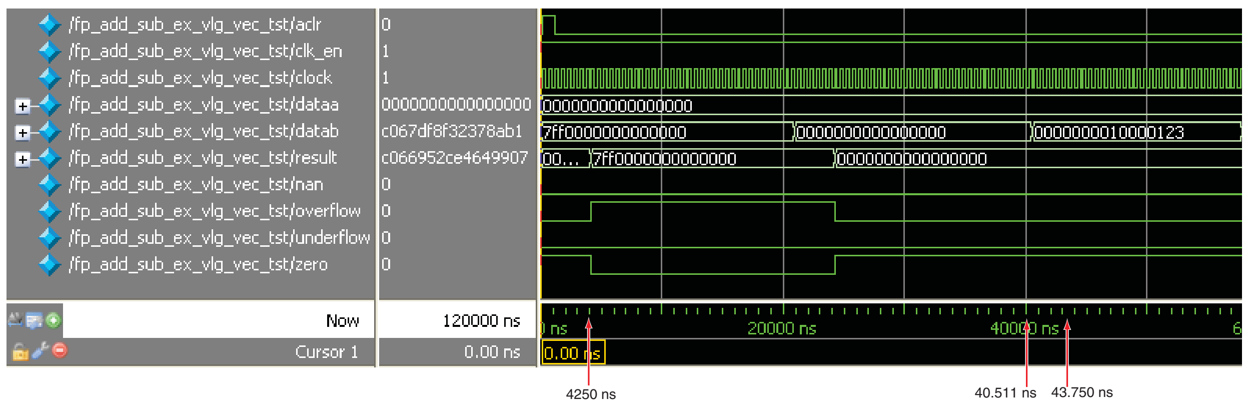
- **altfp_add_sub_DesignExample.zip** (Quartus II design files)
- **altfp_add_sub_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 4–3 shows the expected simulation results in ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 4–3. ALTFP_ADD_SUB Simulation Waveform


/fp_add_sub_ex_vlg_vec_tst/aclr
/fp_add_sub_ex_vlg_vec_tst/clk_en
/fp_add_sub_ex_vlg_vec_tst/clock
/fp_add_sub_ex_vlg_vec_tst/dataa
/fp_add_sub_ex_vlg_vec_tst/datab
/fp_add_sub_ex_vlg_vec_tst/result
/fp_add_sub_ex_vlg_vec_tst/nan
/fp_add_sub_ex_vlg_vec_tst/overflow
/fp_add_sub_ex_vlg_vec_tst/underflow
/fp_add_sub_ex_vlg_vec_tst/zero
0
1
1
0000000000000000
c067df8f32378ab1
c066952ce4649907
0
0
0
0
0000000000000000
7ff0000000000000
0000000000000000
0000000010000123
00...
7ff0000000000000
0000000000000000
Now
120000 ns
Cursor 1
0.00 ns
0 ns
20000 ns
40000 ns
0.00 ns
4250 ns
40.511 ns
43.750 ns

This design example implements a floating-point adder for the addition of double-precision format numbers. All the optional input ports (`clk_en` and `aclr`) and optional output ports (`overflow`, `underflow`, `zero`, and `nan`) are enabled.

In this example, the output latency of the multiplier is set to 7 clock cycles. Every addition result appears at the `result[]` port 7 clock cycles after the input values are captured on the `dataa[]` and `datab[]` ports.

Table 4–3 lists the inputs and corresponding outputs obtained from the simulation in Figure 4–3.

Table 4–3. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`dataa[]` value: 0000 0000 0000 0000h `datab[]` value: 7FF0 0000 0000 0000h Output value: All values seen on the output port before the 7th clock cycle are merely due to the behavior of the system during startup and should be disregarded.
4250 ns	Output value: 7FF0 0000 0000 0000h Exception handling ports: `overflow` asserts The addition of zero at the input port `dataa[]`, and infinity value at the input port `datab[]` results in infinity value.
40,511 ns	`dataa[]` value: 0000 0000 0000 0000h `datab[]` value: 0000 0000 1000 0123h The is the addition of a zero and a denormal value.
43,750 ns	Output value: 0000 0000 0000 0000h Exception handling ports: `zero` remains asserted. Denormal inputs are not supported and are forced to zero before addition takes place.This results in a zero.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_ADD_SUB megafunction. The ports and parameters are available to customize the ALTFP_ADD_SUB megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_add_sub
    generic (
        denormal_support        :       string := "YES";
        intended_device_family  :       string := "unused";
        direction       :       string := "ADD";
        exception_handling      :       string := "YES";
        optimize        :       string := "SPEED";
        pipeline        :       natural := 11;
        reduced_functionality   :       string := "NO";
        rounding        :       string := "TO_NEAREST";
        speed_optimized :       string := "STRATIX_ONLY";
        width_exp       :       natural := 8;
        width_man       :       natural := 23;
        lpm_hint        :       string := "UNUSED";
        lpm_type        :       string := "altfp_add_sub"
    );
    port(
        aclr    :       in std_logic := '0';
        add_sub :       in std_logic := '1';
        clk_en   :       in std_logic := '1';
        clock    :       in std_logic;
        dataa    :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
        datab    :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
        denormal         :       out std_logic;
        indefinite       :       out std_logic;
        nan      :       out std_logic;
        overflow         :       out std_logic;
        result   :       out std_logic_vector(width_exp+width_man+1-1 downto 0);
        underflow        :       out std_logic;
        zero     :       out std_logic
    );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 4–4 lists the input ports of the ALTFP_ADD_SUB megafunction.

Table 4–4. ALTFP_ADD_SUB Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear input for floating-point adder or subtractor. The source is asynchronously reset when the `aclr` signal is asserted high.
`add_sub`	No	Optional input port to enable dynamic switching between the adder and subtractor functions. The `add_sub` port must be used when the `DIRECTION` parameter is set to `VARIABLE`. When the `add_sub` port is high, `result[] = dataa[] + datab[]`, otherwise, `result[] = dataa[] - datab[]`.
`clk_en`	No	Clock enable to the floating-point adder or subtractor. This port allows addition or subtraction to occur when asserted high. When asserted low, no operations occur and the outputs are unchanged.
`clock`	Yes	Clock input to the megafunction.
`dataa[]`	Yes	Data input to the floating-point adder or subtractor. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa bits. The size of this port is the total width of the sign bit, the exponent bits, and the mantissa bits.
`datab[]`	Yes	Data input to the floating-point adder or subtractor. This port is configured in the same way as `dataa[]`.

Table 4–5 lists the output ports of the ALTFP_ADD_SUB megafunction.

Table 4–5. ALTFP_ADD_SUB Megafunction Output Ports

Port Name	Required	Description
`nan`	Yes	NaN exception output. Asserted when an illegal addition or subtraction occurs, such as infinity minus infinity. When an invalid addition or subtraction occurs, a NaN value is output to the `result[]` port. Any adding or subtracting involving NaN values also produces a NaN value.
`overflow`	Yes	Overflow exception port. Asserted when the result of the addition or subtraction, after rounding, exceeds or reaches infinity. Infinity is defined as a number in which the exponent exceeds $2^{\text{WIDTH\_EXP} -1}$.
`result[]`	Yes	Floating-point output result. Like the input values, the MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`underflow`	Yes	Underflow port for the adder or subtractor. Asserted when the result of the addition or subtraction, after rounding, the value is zero and the inputs are not equal. The `underflow` port is also asserted when the result is a denormalized number.
`zero`	No	Zero port for the adder or subtractor. Asserted when the `result[]` port is zero.

Table 4–6 lists the parameters of the ALTFP_ADD_SUB megafunction.

Table 4–6. ALTFP_ADD_SUB Megafunction Parameters (Part 1 of 2)

Parameter Name	Type	Required	Description
`DIRECTION`	String	Yes	Specifies addition or subtraction operations. Values are `ADD`, `SUB`, or `VARIABLE`. If this parameter is not specified, the default is `ADD`. When the value is `VARIABLE`, the `add_sub` port determines whether the operation is addition or subtraction. The `add_sub` port must be connected if the `DIRECTION` parameter is set to `VARIABLE`. If the value is `ADD` or `SUB`, the `add_sub` port is ignored.
`PIPELINE`	Integer	No	Specifies the latency in clock cycles used in the ALTFP_ADD_SUB megafunction. The `PIPELINE` parameter supports values of `7` through `14`. If this parameter is not specified, the default value is `11`. In general, a higher pipeline value produces better f_{MAX} performance.
`ROUNDING`	String	Yes	Specifies the rounding mode. The default value is `TO_NEAREST`. Other rounding modes are currently not supported.
`OPTIMIZE`	String	No	Defines the design preference, whether the design is optimized for speed (faster f_{MAX}), or optimized for area (lower resource count). Values are `SPEED` and `AREA`. If this parameter is not specified, the default is `SPEED`.

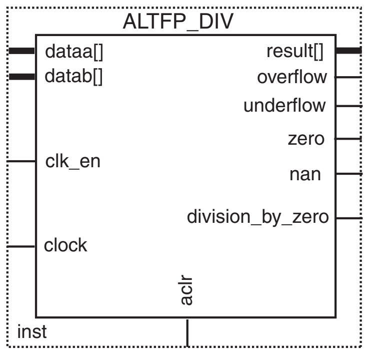
Table 4–6. ALTFP_ADD_SUB Megafunction Parameters (Part 2 of 2)

Parameter Name	Type	Required	Description
WIDTH_EXP	Integer	No	Specifies the precision of the exponent. The bias of the exponent is always set to $2^{(WIDTH\_EXP-1)}$ -1 (that is, 127 for single-precision format and 1023 for double-precision format). The WIDTH_EXP parameter must be 8 for the single-precision mode and 11 for the double-precision mode, or a minimum of 11 for the single-extended precision mode. The WIDTH_EXP parameter must be less than the WIDTH_MAN parameter. The sum of WIDTH_EXP and the WIDTH_MAN parameters must be less than 64. If this parameter is not specified, the default is 8.
WIDTH_MAN	Integer	No	Specifies the precision of the mantissa. The WIDTH_MAN parameter must be 23 (to comply with the IEEE-754 standard for the single-precision mode) when the WIDTH_EXP parameter is 8. Otherwise, the WIDTH_MAN parameter must have a value that is greater than or equal to 31. The WIDTH_MAN parameter must be greater than the WIDTH_EXP parameter. The sum of the WIDTH_EXP and WIDTH_MAN parameters must be less than 64. If this parameter is not specified, the default is 23.

5. ALTFP_DIV

Figure 5–1 shows the ports for the ALTFP_DIV megafunction.

Figure 5–1. ALTFP_DIV Ports



Features

The ALTFP_DIV megafunction offers the following features:

- Division functions.
- Optional exception handling output ports such as `zero`, `division_by_zero`, `overflow`, `underflow`, and `nan`.
- Optimization of speed and area.
- Low latency option.

Output Latency

The output latency options for the ALTFP_DIV megafunction differs depending on the precision selected, the width of the mantissa, or both. You have the choice of selecting the smaller figures of clock cycles delay in your design if the low latency option is desired.

Table 5–1 lists the output latency options that can be selected for each precision format.

Table 5–1. Latency Options for Each Operation (Part 1 of 2)

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	6, 14, 33
Double	52	10, 24, 61

Table 5–1. Latency Options for Each Operation (Part 2 of 2)

Precision	Mantissa Width	Latency (in clock cycles)
Single Extended	31 – 32	8, 18, 41
	33 – 34	8, 18, 43
	35 – 36	8, 18, 45
	37 – 38	8, 18, 47
	39 – 40	8, 18, 49
	41	10, 24, 41
	42	10, 24, 51
	43 – 44	10, 24, 53
	45 – 46	10, 24, 55
	47 – 48	10, 24, 57
	49 – 50	10, 24, 59
	51 – 52	10, 24, 61

Truth Table

Table 5–2 lists the truth table for the division operation.

Table 5–2. Truth Table for Division Operations (Part 1 of 2)

DATAA[]	DATAB[]	SIGN BIT	RESULT[]	Overflow	Underflow	Zero	Division-by-zero	NaN
Normal	Normal	0/1	Normal	0	0	0	0	0
Normal	Normal	0/1	Denormal	0	0	1	0	0
Normal	Normal	0/1	Infinity	1	0	0	0	0
Normal	Normal	0/1	Zero	0	1	1	0	0
Normal	Denormal	0/1	Infinity	0	0	0	1	0
Normal	Zero	0/1	Infinity	0	0	0	1	0
Normal	Infinity	0/1	Zero	0	0	1	0	0
Normal	NaN	X	NaN	0	0	0	0	1
Denormal	Normal	0/1	Zero	0	0	1	0	0
Denormal	Denormal	0/1	NaN	0	0	0	0	1
Denormal	Zero	0/1	NaN	0	0	0	0	1
Denormal	Infinity	0/1	Zero	0	0	1	0	0
Denormal	NaN	X	NaN	0	0	0	0	1
Zero	Normal	0/1	Zero	0	0	1	0	0
Zero	Denormal	0/1	NaN	0	0	0	0	1
Zero	Zero	0/1	NaN	0	0	0	0	1
Zero	Infinity	0/1	Zero	0	0	1	0	0
Zero	NaN	X	NaN	0	0	0	0	1
Infinity	Normal	0/1	Infinity	0	0	0	0	0
Infinity	Denormal	0/1	Infinity	0	0	0	0	0
Infinity	Zero	0/1	Infinity	0	0	0	0	0

Table 5–2. Truth Table for Division Operations (Part 2 of 2)

DATAA[]	DATAB[]	SIGN BIT	RESULT[]	Overflow	Underflow	Zero	Division-by-zero	NaN
Infinity	Infinity	0/1	NaN	0	0	0	0	1
Infinity	NaN	X	NaN	0	0	0	0	1
NaN	Normal	X	NaN	0	0	0	0	1
NaN	Denormal	X	NaN	0	0	0	1	1
NaN	Zero	X	NaN	0	0	0	1	1
NaN	Infinity	X	NaN	0	0	0	0	1
NaN	NaN	X	NaN	0	0	0	0	1

Resource Utilization and Performance

Table 5–3 lists the resource utilization and performance information for the ALTFP_DIV megafunction. The information was derived using the Quartus II software version 10.0.

Table 5–3. ALTFP_DIV Resource Utilization and Performance for the Stratix Series of Devices

Device family	Precision	Optimization	Output latency	Logic Usage				f_{MAX}(MHz)
				Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-bit DSP	
Stratix III	Single	Speed	33	3,658	3,374	2,510	—	283
		Area	33	1,722	2,097	1,467	—	278
	Double	Speed	61	14,191	13,298	10,501	—	264
		Area	61	5,425	7,515	5,021	—	216
Stratix IV	Single	Speed	33	3,593	3,351	2,500	—	313
		Area	33	1,646	2,074	1,441	—	308
	Double	Speed	61	13,867	13,143	10,196	—	292
		Area	61	5,125	7,360	4,842	—	267
Low Latency Option								
Stratix III	Single	—	6	207	304	208	16	136
		—	14	295	331	262	16	296
	Double	—	10	714	1,077	768	44	133
		—	24	1,104	1,592	963	44	195
Stratix IV	Single	—	6	207	304	212	16	154
		—	14	253	638	385	16	358
	Double	—	10	714	1,077	779	44	151
		—	24	765	2,488	1,397	44	238

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

This design example uses the ALTFP_DIV megafunction to implement a floating-point divider for the division of single-precision format numbers with low latency. This example uses the MegaWizard Plug-In Manager flow in the Quartus II software.

Design Files

The following files are related to the ALTFP_DIV megafunction:

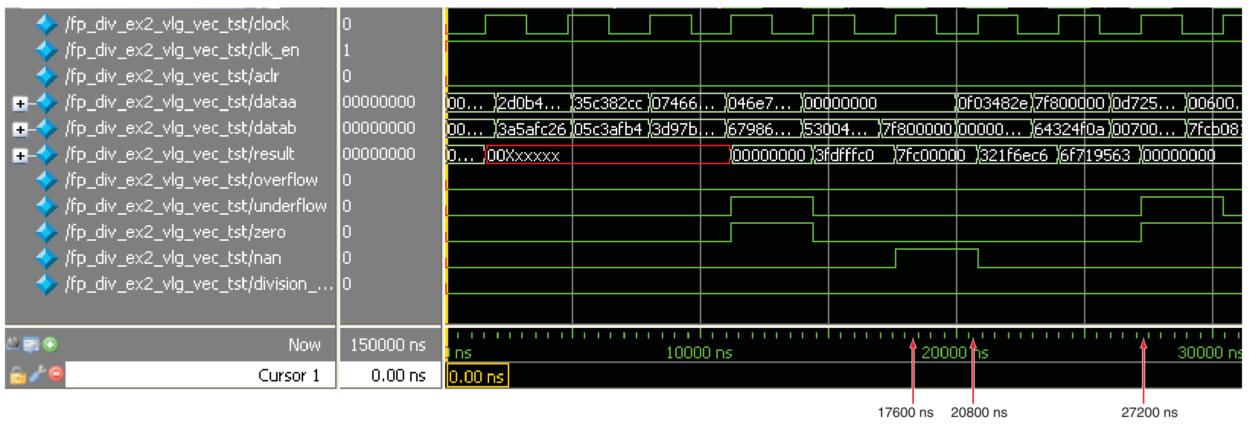
- **altfp_div_DesignExample.zip** (Quartus II design files)
- **altfp_div_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 1 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 1. ALTFP_DIV Simulation Waveform


/fp_div_ex2_vlg_vec_tst/clock 0
/fp_div_ex2_vlg_vec_tst/clk_en 1
/fp_div_ex2_vlg_vec_tst/aclr 0
/fp_div_ex2_vlg_vec_tst/dataa 00000000
/fp_div_ex2_vlg_vec_tst/datab 00000000
/fp_div_ex2_vlg_vec_tst/result 00000000
/fp_div_ex2_vlg_vec_tst/overflow 0
/fp_div_ex2_vlg_vec_tst/underflow 0
/fp_div_ex2_vlg_vec_tst/zero 0
/fp_div_ex2_vlg_vec_tst/nan 0
/fp_div_ex2_vlg_vec_tst/division_... 0
00... 2d0b4... 35c382cc 07466... 046e7... 00000000 0f03482e 7f800000 0d725... 00600.
00... 3a5afc26 05c3afb4 3d97b... 67986... 53004... 7f800000 00000... 64324f0a 00700... 7fcb08
0... 00Xxxxxx 00000000 3fdfffc0 7fc00000 321f6ec6 6f719563 00000000
Now 150000 ns
Cursor 1 0.00 ns
ns 10000 ns 20000 ns 30000 ns
0.00 ns
17600 ns
20800 ns
27200 ns

This design example implements a floating-point divider for the division of single-precision numbers with a low latency option. The output latency is 6, hence every division generates the output result 6 clock cycles later.

Table 5–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 1.

Table 5–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`dataa[]` value: 0000 0000h `datab[]` value: 0000 0000h Output value: The undefined value is seen on the `result[]` port, which is ignored. All values seen on the output port before the 6th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
17600 ns	Output value: 7FC0 0000h Exception handling ports: `nan` asserts The division of zeros result in a NaN.
2000 ns	`dataa[]` value: 2D0B 496Ah `datab[]` value: 3A5A FC26h Both inputs hold normal values.
20800 ns	Output result: 321F 6EC6h Exception output ports: `nan` deasserts The division of two normal value results in a normal value.
11000 ns	`dataa[]` value: 046E 78BCh `datab[]` value: 6798 698Bh Both inputs hold normal values.
27200 ns	Output value: 0h Exception handling ports: `underflow` and `zero` asserts The division of the two normal values results in a denormal value. As denormal values are not supported, the result is zero and the `underflow` port asserts. The `zero` port is also asserted to indicate that the result is zero.
2600 ns	`dataa[]` value: 0D72 54A8h `datab[]` value: 0070 0000h The input port `dataa[]` holds a normal value while the input port `datab[]` holds a denormal value.
36800 ns	Output value: 7F80 0000h Exception handling ports: `division_by_zero` asserts Denormal numbers are forced-zero values, therefore, attempts to divide a normal value with a zero result in an infinity value.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_DIV megafunction. The ports and parameters are available to customize the ALTFP_DIV megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_div
    generic (
        intended_device_family  :      string := "unused";
        exception_handling      :      string := "YES";
        optimize        :       string := "SPEED";
        pipeline        :       natural := 32;
        rounding        :       string := "TO_NEAREST";
        width_exp       :       natural := 8;
        width_man       :       natural := 23;
        lpm_hint        :       string := "UNUSED";
        lpm_type        :       string := "altfp_div"
    );
    port(
        aclr    :       in std_logic := '0';
        clk_en  :       in std_logic := '1';
        clock   :       in std_logic;
        dataa   :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
        datab   :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
        division_by_zero        :       out std_logic;
        nan     :       out std_logic;
        overflow        :       out std_logic;
        result  :       out std_logic_vector(width_exp+width_man+1-1 downto 0);
        underflow       :       out std_logic;
        zero    :       out std_logic
);
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
```

```
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 5–5 lists the input ports of the ALTFP_DIV megafunction.

Table 5–5. ALTFP_DIV Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear input for the floating-point divider. The source is asynchronously reset when the `aclr` signal is asserted high.
`clock`	Yes	Clock input to the megafunction.
`clk_en`	No	Clock enable to the floating-point divider. This port enables division. This signal is active high. When this signal is low, no division takes place and the outputs remain the same.
`dataa[]`	Yes	Numerator data input. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits and mantissa bits.
`datab[]`	Yes	Denominator data input.The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits and mantissa bits.

Table 5–6 lists the output ports of the ALTFP_DIV megafunction.

Table 5–6. ALTFP_DIV Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	Divider output port. The division result (after rounding). As with the input values, the MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`overflow`	No	Overflow port for the divider. Asserted when the result of the division (after rounding) exceeds or reaches infinity. Infinity is defined as a number in which the exponent exceeds $2^{\text{WIDTH\_EXP}-1}$.
`underflow`	No	Underflow port for the divider. 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.
`zero`	No	Zero port for the divider. Asserted when the value of `result[]` is zero.
`division_by_zero`	No	Division-by-zero output port for the divider. Asserted when the value of `datab[] is a zero.`
`nan`	No	NaN port. Asserted when an invalid division occurs, such as infinity dividing infinity or zero dividing zero. A NaN value appears as output at the `result[]` port. Any division of a NaN value causes the `nan` output port to be asserted.

Table 5–7 lists the parameters of the ALTFP_DIV megafunction.

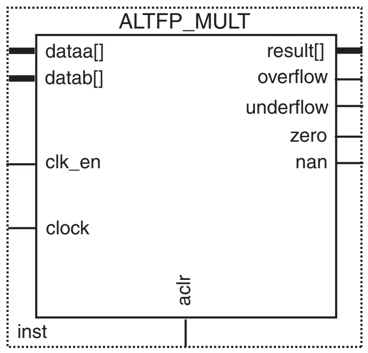
Table 5–7. ALTFP_DIV Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to (2 ^ (`WIDTH_EXP` - 1)) - 1, that is, 127 for single precision and 1023 for double precision. The value of `WIDTH_EXP` must be `8` for single precision, `11` for double precision, and a minimum of 11 for single extended precision. The value of `WIDTH_EXP` must be less than the value of `WIDTH_MAN`, and the sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is `23`. When `WIDTH_EXP` is `8` and the floating-point format is the single-precision format, the `WIDTH_MAN` value must be `23`. Otherwise, the value of `WIDTH_MAN` must be a minimum of `31`. The value of `WIDTH_MAN` must be greater than the value of `WIDTH_EXP`, and the sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`ROUNDING`	String	Yes	Specifies the rounding mode. The default value is `TO_NEAREST`. The floating-point divider does not support other rounding modes.
`OPTIMIZE`	String	No	Specifies whether to optimize for area or for speed. Values are `AREA` and `SPEED`. A value of `AREA` optimizes the design using less total logic utilization or resources. A value of `SPEED` optimizes the design for better performance. If this parameter is not specified, the default value is `SPEED`.
`PIPELINE`	Integer	No	Specifies the number of clock cycles needed to produce the result. For the single-precision format, the latency options are `33`, `14` or `6`. For the double-precision format, the latency options are `61`, `24` or `10`. For the single-extended precision format, the value ranges from a minimum of `41` to a maximum of `61`. For the low-latency option, the latency is determined from the mantissa width. For a mantissa width of `31` to `40` bits, the value is `8` or `18`. For a mantissa width of `41` bits or more, the value is `10` or `24`.

6. ALTFP_MULT

Figure 6–1 shows the ports for the ALTFP_MULT megafunction.

Figure 6–1. ALTFP_MULT Ports



Features

The ALTFP_MULT megafunction offers the following features:

- Multiplication functions.
- Optional exception handling output ports such as `zero`, `overflow`, `underflow`, and `nan`.
- Optional dedicated multiplier circuitries in Cyclone and Stratix series.

Output Latency

The output latency options for the ALTFP_MULT megafunction are similar for all precisions.

Table 6–1 lists the output latency options required for each precision format.

Table 6–1. Latency Options for Each Precision Format

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	5, 6, 10,11
Double	52	5, 6, 10,11
Single-Extended	31–52	5, 6, 10,11

Truth Table

Table 6–2 lists the truth table for the multiplier operation.

Table 6–2. Truth Table for Multiplier Operations

DATAA[]	DATAB[]	RESULT[]	Overflow	Underflow	Zero	NaN
Normal	Normal	Normal	0	0	0	0
Normal	Normal	Denormal	0	1	1	0
Normal	Normal	Infinity	1	0	0	0
Normal	Normal	Zero	0	1	1	0
Normal	Denormal	Zero	0	0	1	0
Normal	Zero	Zero	0	0	1	0
Normal	Infinity	Infinity	1	0	0	0
Normal	NaN	NaN	0	0	0	1
Denormal	Normal	Zero	0	0	1	0
Denormal	Denormal	Zero	0	0	1	0
Denormal	Zero	Zero	0	0	1	0
Denormal	Infinity	NaN	0	0	0	1
Denormal	NaN	NaN	0	0	0	1
Zero	Normal	Zero	0	0	1	0
Zero	Denormal	Zero	0	0	1	0
Zero	Zero	Zero	0	0	1	0
Zero	Infinity	NaN	0	0	0	1
Zero	NaN	NaN	0	0	0	1
Infinity	Normal	Infinity	1	0	0	0
Infinity	Denormal	NaN	0	0	0	1
Infinity	Zero	NaN	0	0	0	1
Infinity	Infinity	Infinity	1	0	0	0
Infinity	NaN	NaN	0	0	0	1
NaN	Normal	NaN	0	0	0	1
NaN	Denormal	NaN	0	0	0	1
NaN	Zero	NaN	0	0	0	1
NaN	Infinity	NaN	0	0	0	1
NaN	NaN	NaN	0	0	0	1

Resource Utilization and Performance

Table 6–3 and Table 6–4 list the resource utilization and performance information for the ALTFP_MULT megafunction. The information was derived using the Quartus II software version 10.0.

Table 6–3. ALTFP_MULT Resource Utilization and Performance for the Stratix Series of Devices with Dedicated Multiplier Circuitry

Device Family	Precision	Output latency	Logic usage				f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-bit DSP	
Stratix III	Single	5	138	148	101	4	240
		11	195	301	206	4	466
	Double	5	306	367	272	10	214
		11	416	523	347	10	312
Stratix IV	Single	5	138	148	100	4	274
		11	185	301	190	4	445
	Double	5	306	367	272	10	255
		11	419	523	348	10	395

Table 6–4. ALTFP_MULT Resource Utilization and Performance for the Cyclone Series of Devices with Dedicated Multiplier Circuitry

Device Family	Precision	Output latency	Logic usage			f_{MAX} (MHz)
			Logic Elements (LEs)	Dedicated Logic Registers (DLRs)	Total Logic Utilization	
Cyclone II	Single	5	296	221	296	188
	Double	5	688	412	688	101
Cyclone III	Single	5	295	221	295	209
	Double	5	687	412	687	122

Design Example: Multiplication of Double-Precision Format Numbers

This design example uses the ALTFP_MULT megafunction to compute the multiplication results of two double-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_MULT megafunction:

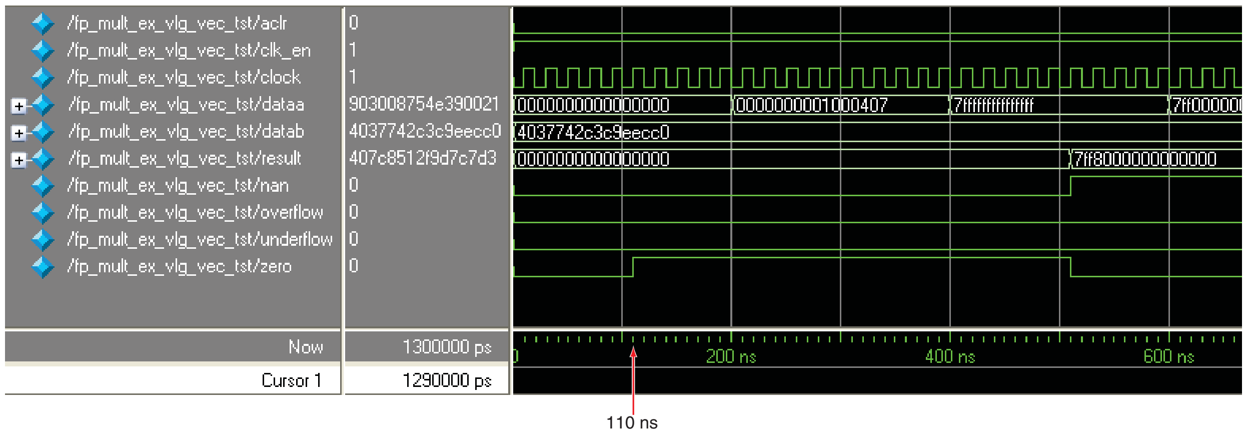
- **altfp_mult_DesignExample.zip** (Quartus II design files)
- **altfp_mult_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 6–2 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 6–2. ALTFP_MULT Simulation Waveform


/fp_mult_ex_vlg_vec_tst/aclr 0
/fp_mult_ex_vlg_vec_tst/clk_en 1
/fp_mult_ex_vlg_vec_tst/clock 1
/fp_mult_ex_vlg_vec_tst/dataa 903008754e390021
/fp_mult_ex_vlg_vec_tst/datab 4037742c3c9eecc0
/fp_mult_ex_vlg_vec_tst/result 407c8512f9d7c7d3
/fp_mult_ex_vlg_vec_tst/nan 0
/fp_mult_ex_vlg_vec_tst/overflow 0
/fp_mult_ex_vlg_vec_tst/underflow 0
/fp_mult_ex_vlg_vec_tst/zero 0
0000000000000000
0000000001000407
7fffffffffffffff
7ff000000
4037742c3c9eecc0
0000000000000000
7ff8000000000000
Now 1300000 ps
Cursor 1 1290000 ps
0
200 ns
400 ns
600 ns
110 ns

This design example implements a floating-point multiplier for the multiplication of double-precision format numbers. All the optional input ports (`clk_en` and `aclr`) and output ports (`overflow`, `underflow`, `zero`, and `nan`) are enabled.

In this example, the latency is set to 6 clock cycles. Therefore, every multiplication result appears at the result port 6 clock cycles later.

Table 6–5 lists the inputs and corresponding outputs obtained from the simulation in Figure 6–2.

Table 6–5. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`dataa[]` value: 0000 0000 0000 0000h `datab[]` value: 4037 742C 3C9E ECC0h Output value: All values seen on the output port before the 6th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
110 ns	Output value: 0000 0000 0000 0000h Exception handling ports: `zero` asserts The multiplication of zero at the input port `dataa[]`, and a non-zero value at the input port `datab[]` results in a zero.
600 ns	`dataa[]` value: 7FF0 0000 0000 0000h `datab[]` value: 4037 742C 3C9E ECC0h This is the multiplication of an infinity value and a normal value.
710 ns	Output value: 7FF0 0000 0000 0000h Exception handling ports: `overflow` asserts The multiplication of an infinity value and a normal value results in infinity. All multiplications with an infinity value results in infinity except when infinity is multiplied with a zero.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_MULT megafunction. The ports and parameters are available to customize the ALTFP_MULT megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_mult
        generic (
                dedicated_multiplier_circuitry  :       string := "AUTO";
                intended_device_family  :       string := "unused";
                pipeline        :       natural := 5;
                width_exp       :       natural := 8;
                width_man       :       natural := 23;
                lpm_hint        :       string := "UNUSED";
                lpm_type        :       string := "altfp_mult"
        );
        port(
                aclr    :       in std_logic := '0';
                clk_en  :       in std_logic := '1';
                clock   :       in std_logic;
                dataa   :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
                datab   :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
                nan     :       out std_logic;
                overflow        :       out std_logic;
                result   :        out std_logic_vector(width_exp+width_man+1-1 downto 0);
                underflow         :        out std_logic;
                zero     :        out std_logic
        );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 6–6 lists the input ports of the ALTFP_MULT megafunction.

Table 6–6. ALTFP_MULT Megafunction Input Ports

Port Name	Required	Description
`clock`	Yes	Clock input to the megafunction.
`clk_en`	No	Clock enable. Allows multiplication to take place when asserted high. When signal is asserted low, no multiplication occurs and the outputs remain unchanged.
`aclr`	No	Synchronous clear. Source is asynchronously reset when asserted high.
`dataa[]`	Yes	Floating-point input data input to the multiplier. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.
`datab[]`	Yes	Floating-point input data to the multiplier. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.

Table 6–7 lists the output ports of the ALTFP_MULT megafunction.

Table 6–7. ALTFP_MULT Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	Output port for the multiplier. The floating-point result after rounding. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa.
`overflow`	No	Overflow port for the multiplier. Asserted when the result of the multiplication, after rounding, exceeds or reaches infinity. Infinity is defined as a number in which the exponent exceeds $2^{WIDTH\_EXP}-1$.
`underflow`	No	Underflow port for the multiplier. Asserted when the result of the multiplication (after rounding) is `0` while none of the inputs to the multiplication is `0`, or asserted when the result is a denormalized number.
`zero`	No	Zero port for the multiplier. Asserted when the value of `result[]` is `0`.
`nan`	No	NaN port for the multiplier. This port is asserted when an invalid multiplication occurs, such as the multiplication of infinity and zero. In this case, a NaN value is the output generated at the `result[]` port. The multiplication of any value and NaN produces NaN.

Table 6–8 lists the parameters of the ALTFP_MULT megafunction.

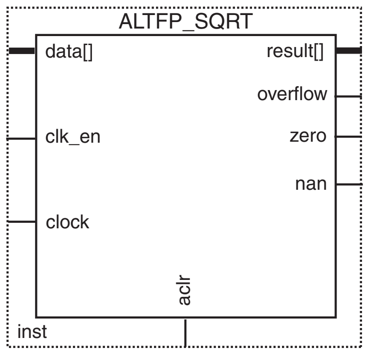
Table 6–8. ALTFP_MULT Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	No	Specifies the value of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always `2`$^{(\texttt{WIDTH\_EXP-1})}$`-1` (that is, `127` for the single-precision format and `1023` for the double-precision format). `WIDTH_EXP` must be `8` for the single-precision format or a minimum of `11` for the double-precision format and the single-extended precision format. `WIDTH_EXP` must less than `WIDTH_MAN`. The sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`WIDTH_MAN`	Integer	No	Specifies the value of the mantissa. If this parameter is not specified, the default is `23`. When `WIDTH_EXP` is `8` and the floating-point format is single-precision, the `WIDTH_MAN` value must be `23`; otherwise, the value of `WIDTH_MAN` must be a minimum of `31`. The `WIDTH_MAN` value must always be greater than the `WIDTH_EXP` value. The sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`DEDICATED_MULTIPLIER_ CIRCUITRY`	String	No	Specifies whether to use dedicated multiplier circuitry. Values are `AUTO`, `YES`, or `NO`. If this parameter is not specified, the default is `AUTO`. If a device does not have dedicated multiplier circuitry, the `DEDICATED_MULTIPLIER_CIRCUITRY` parameter has no effect and defaults to `NO`.
`PIPELINE`	Integer	No	Specifies the number of clock cycles needed to produce the multiplied result. Values are `5`, `6`, `10`, and `11`. If this parameter is not specified, the default is `5`.

7. ALTFP_SQRT

Figure 7–1 shows the ports for the ALTFP_SQRT megafunction.

Figure 7–1. ALTFP_SQRT Ports



Features

The ALTFP_SQRT megafunction offers the following features:

- Square root functions.
- Optional exception handling output ports such as `zero`, `overflow`, and `nan`.

Output Latency

The output latency options for the ALTFP_SQRT megafunction differs depending on the precision selected, the width of the mantissa, or both.

Table 7–1 lists the output latency options required for each precision format.

Table 7–1. Latency Options for Each Precision Format (Part 1 of 2)

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	16, 28
Double	52	30, 57

Table 7–1. Latency Options for Each Precision Format (Part 2 of 2)

Precision	Mantissa Width	Latency (in clock cycles)
Single-extended	31	20, 36
	32	20, 37
	33	21, 38
	34	21, 39
	35	22, 40
	36	22, 41
	37	23, 42
	38	23, 43
	39	24, 44
	40	24, 45
	41	25, 46
	42	25, 47
	43	26, 48
	44	26, 49
	45	27, 50
	46	27, 51
	47	28, 52
	48	28, 53
	49	29, 54
	50	29, 55
	51	30, 56

Truth Table

Table 7–2 lists the truth table for square root operations.

Table 7–2. Truth Table for Square Root Operations

DATA[]	SIGN BIT	RESULT[]	NaN	Overflow	Zero
Normal	0	Normal	0	0	0
Denormal	0/1	Zero	0	0	1
Positive Infinity	0	Infinity	0	1	0
Negative Infinity	1	All 1’s	1	0	0
Positive NaN	0	All 1’s	1	0	0
Negative NaN	1	All 1’s	1	0	0
Zero	0/1	Zero	0	0	1
Normal	1	All 1’s	1	0	0

Resource Utilization and Performance

Table 7–3 lists the resource utilization and performance information for the ALTFP_SQRT megafunction. The information was derived using the Quartus II software version 10.0.

Table 7–3. ALTFP_SQRT Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output latency	Logic usage			f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Login Registers (DLRs)	Adaptive Logic Modules (ALMs)	
Stratix III	Single	28	526	942	536	396
	Double	57	2,311	3,815	2,311	283
Stratix IV	Single	28	502	932	528	472
	Double	57	2,177	3,725	2,202	366

Design Example: Square Root of Single-Precision Format Numbers

This design example uses the ALTFP_SQRT megafunction to compute the square root of single-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_SQRT megafunction:

- **altfp_sqrt_DesignExample.zip** (Quartus II design files)
- **altfp_sqrt_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 7–2 and Figure 7–3 show the expected simulation results in the ModelSim-Altera software.

☞ The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 7–2. ModelSim Simulation Waveform (Input Data)

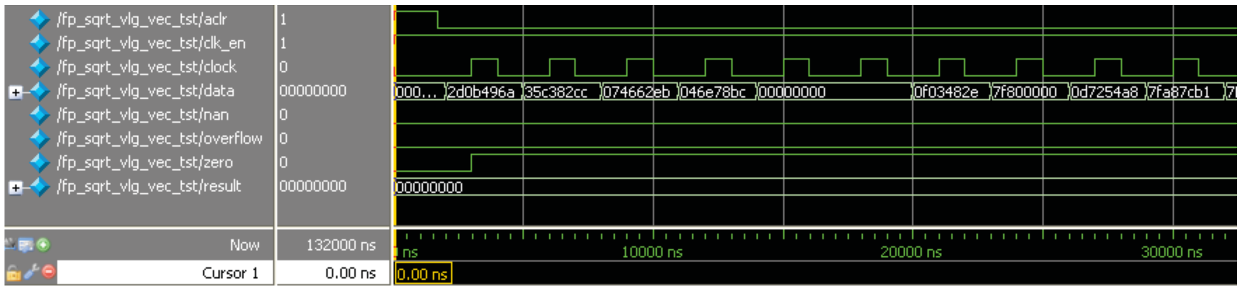
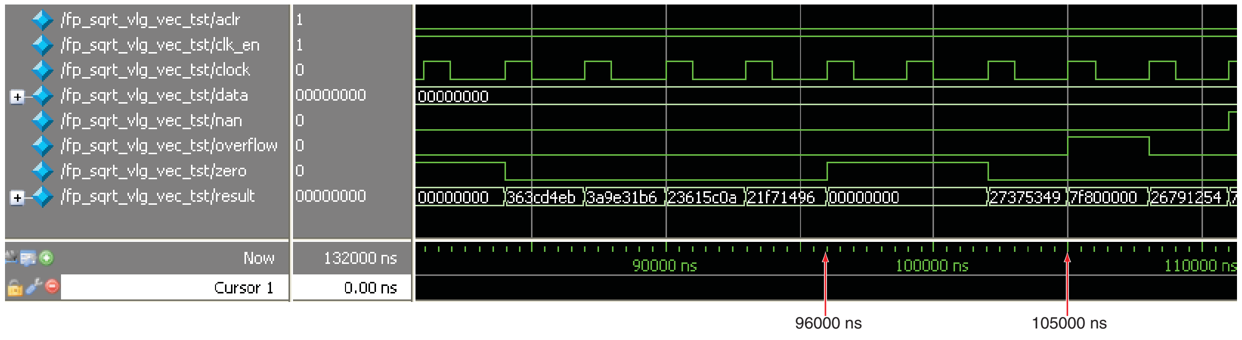


Figure 7–3. ModelSim Simulation Waveform (Output Data)



This design example implements a floating-point square root function for single-precision format 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.

Table 7–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 7–2 on page 7–4 and Figure 7–3.

Table 7–4. Summary of Input Values and Corresponding Outputs (Part 1 of 2)

Time	Event
0 ns, start-up	Output value: All values seen on the output port before the 28th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
2 000 ns	`data[]` value: 2D0B 496Ah The data input is a normal number.
84 000 ns	Output value: 363C D4EBh The square root computation of a normal input results in a normal output.
14 000 ns	`data[]` value: 0000 0000h
96 000 ns	Output value: 0000 0000h Exception handling ports: `zero` asserts The square root computation of zero results in a zero.

Table 7–4. Summary of Input Values and Corresponding Outputs (Part 2 of 2)

Time	Event
23 000 ns	`data[]` value: 7F80 0000h The input is infinity.
105 000 ns	Output value: 7F80 0000h Exception handling ports: `overflow` asserts

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_SQRT megafunction. The ports and parameters are available to customize the ALTFP_SQRT megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_sqrt
        generic (
                intended_device_family  :       string := "unused";
                exception_handling      :       string := "YES";
                pipeline        :       natural := 28;
                rounding        :       string := "TO_NEAREST";
                width_exp       :       natural := 8;
                width_man       :       natural := 23;
                lpm_hint        :       string := "UNUSED";
                lpm_type        :       string := "altfp_sqrt"
        );
        port(
                aclr   :       in std_logic := '0';
                clk_en :       in std_logic := '1';
                clock  :       in std_logic;
                data   :       in std_logic_vector(width_exp+width_man+1-1 downto 0);
                nan    :       out std_logic;
                overflow       :       out std_logic;
                result :       out std_logic_vector(width_exp+width_man+1-1 downto 0);
                zero   :       out std_logic
        );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
```

```
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 7–5 lists the input ports of the ALTFP_SQRT megafunction.

Table 7–5. ALTFP_SQRT Megafunction Input Ports

Port Name	Required	Description
`clock`	Yes	Clock input to the megafunction.
`clk_en`	No	Clock enable that allows square root operations when the port is asserted high. When the port is asserted low, no operation occurs and the outputs remain unchanged.
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously reset.
`data[]`	Yes	Floating-point input data. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.

Table 7–6 lists the output ports of the ALTFP_SQRT megafunction.

Table 7–6. ALTFP_SQRT Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	Square root output port for the floating-point result. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`overflow`	Yes	Overflow port. Asserted when the result of the square root (after rounding) exceeds or reaches infinity. Infinity is defined as a number in which the exponent exceeds $2^{WIDTH\_EXP}-1$.
`zero`	Yes	Zero port. Asserted when the value of the `result[]` port is `0`.
`nan`	Yes	NaN port. Asserted when an invalid square root occurs, such as negative numbers or NaN inputs.

Table 7–7 lists the parameters of the ALTFP_SQRT megafunction.

Table 7–7. ALTFP_SQRT Megafunction Parameters (Part 1 of 2)

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to $2^{(WIDTH\_EXP -1)}-1$, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of the `WIDTH_EXP` parameter must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of the `WIDTH_EXP` parameter must be less than the value of the `WIDTH_MAN` parameter, and the sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the value of the mantissa. If this parameter is not specified, the default is `23`. When the `WIDTH_EXP` parameter is `8` and the floating-point format is single-precision, the `WIDTH_MAN` parameter value must be `23`. Otherwise, the value of the `WIDTH_MAN` parameter must be a minimum of `31`. The value of the `WIDTH_MAN` parameter must be greater than the value of the `WIDTH_EXP` parameter. The sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.

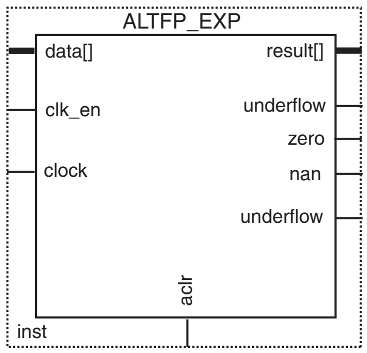
Table 7–7. ALTFP_SQRT Megafunction Parameters (Part 2 of 2)

Parameter Name	Type	Required	Description
`ROUNDING`	String	Yes	Specifies the rounding mode. The default value is `TO_NEAREST`. Other rounding modes are not supported.
`PIPELINE`	Integer	Yes	Specifies the number of clock cycles for the square root results of the `result[]` port. Values are `WIDTH_MAN` + `5` and ((`WIDTH_MAN` + `5/2`)+`2`) as specified by truncating the radix point.

8. ALTFP_EXP

Figure 8–1 shows the ports for the ALTFP_EXP megafunction.

Figure 8–1. ALTFP_EXP Ports



Features

The ALTFP_EXP megafunction offers the following features:

- Exponential value of a given input.
- Optional exception handling output ports such as `zero`, `overflow`, `underflow`, and `nan`.

Output Latency

The output latency options for the ALTFP_EXP megafunction differs depending on the precision selected, the width of the mantissa, or both.

Table 8–1 lists the output latency options required for each precision format.

Table 8–1. Latency Options for Each Precision Format

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	17
Double	52	25
Single-extended	31 – 38	22
	39 – 52	25

Truth Table

Table 8–2 lists the truth table for the exponential operation.

Table 8–2. Truth Table for Exponential Operations

DATAA[]	Calculation	RESULT[]	NaN	Overflow	Underflow	Zero
Normal	e^{data}	Normal	0	0	0	0
Normal	e^{data}	Infinity	0	1	0	0
Normal (numbers of small magnitude)	e^{data}	1	0	0	1	0
Normal (negative numbers of large magnitude)	e^{data}	0	0	0	1	0
Denormal *(1)*	e^0	1	0	0	0	0
Zero	e^0	1	0	0	0	0
Infinity (+)	e^+	Infinity	0	0	0	0
Infinity (-)	e^-	0	0	0	0	1
NaN	—	NaN	1	0	0	0

Note to Table 8–2:

(1) Any denormal input is treated as a zero before going through the exponential process.

Resource Utilization and Performance

Table 8–3 lists the resource utilization and performance information for the ALTFP_EXP megafunction. The information was derived using the Quartus II software version 10.0.

Table 8–3. ALTFP_EXP Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-bit DSP	
Stratix III	Single	17	631	521	445	19	275
	Double	25	4,138	2,022	2,959	46	257
Stratix IV	Single	17	631	521	448	19	284
	Double	25	4,104	2,007	2,939	46	279

Design Example: Exponential of Single-Precision Format Numbers

This design example uses the ALTFP_EXP megafunction to compute the exponential value of single-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_EXP megafunction:

- **altfp_exp_DesignExample.zip** (Quartus II design files)
- **altfp_exp_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 8–2 and Figure 8–3 show the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 8–2. ModelSim Simulation Waveform (Input Data)

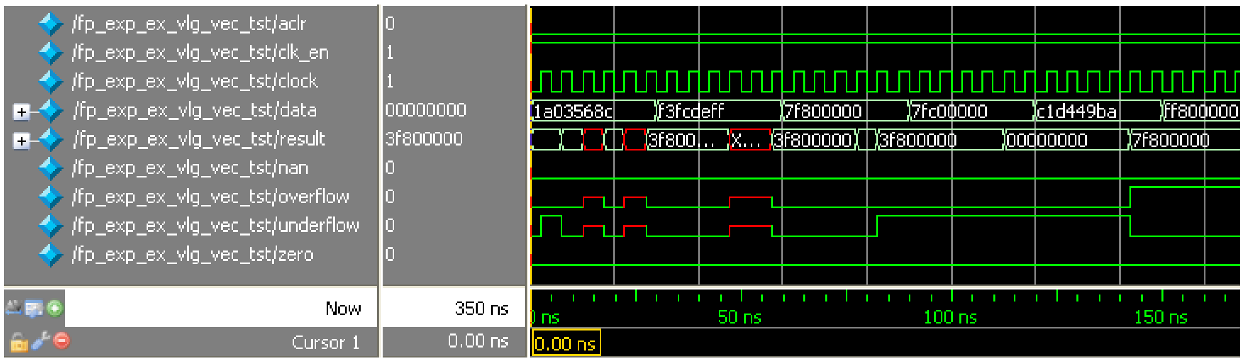
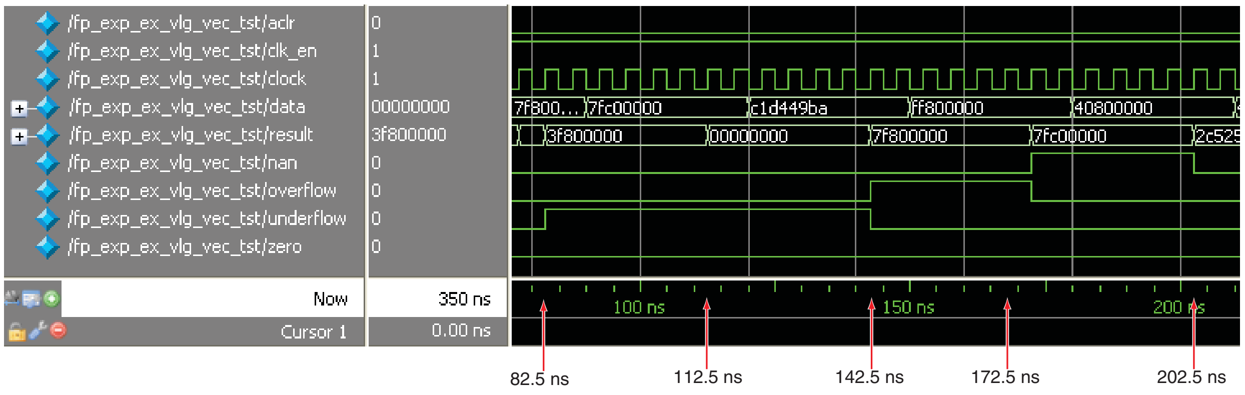

```
/fp_exp_ex_vlg_vec_tst/aclr        0
/fp_exp_ex_vlg_vec_tst/clk_en      1
/fp_exp_ex_vlg_vec_tst/clock       1
/fp_exp_ex_vlg_vec_tst/data        00000000   1a03568c  f3fcdeff  7f800000  7fc00000  c1d449ba  ff800000
/fp_exp_ex_vlg_vec_tst/result      3f800000   3f800..  X...  3f800000  3f800000  00000000  7f800000
/fp_exp_ex_vlg_vec_tst/nan         0
/fp_exp_ex_vlg_vec_tst/overflow    0
/fp_exp_ex_vlg_vec_tst/underflow   0
/fp_exp_ex_vlg_vec_tst/zero        0
Now                                350 ns     0 ns  50 ns  100 ns  150 ns
Cursor 1                           0.00 ns    0.00 ns
```

Figure 8–3. ModelSim Simulation Waveform (Output Data)


```
/fp_exp_ex_vlg_vec_tst/aclr        0
/fp_exp_ex_vlg_vec_tst/clk_en      1
/fp_exp_ex_vlg_vec_tst/clock       1
/fp_exp_ex_vlg_vec_tst/data        00000000   7f800...  7fc00000  c1d449ba  ff800000  40800000
/fp_exp_ex_vlg_vec_tst/result      3f800000   3f800000  00000000  7f800000  7fc00000  2c525
/fp_exp_ex_vlg_vec_tst/nan         0
/fp_exp_ex_vlg_vec_tst/overflow    0
/fp_exp_ex_vlg_vec_tst/underflow   0
/fp_exp_ex_vlg_vec_tst/zero        0
Now                                350 ns     100 ns  150 ns  200 ns
Cursor 1                           0.00 ns
```

This design example implements a floating-point exponential for the single-precision format numbers. The optional input ports (`clk_en` and `aclr`) and all four exception handling output ports (`nan`, `overflow`, `underflow`, and `zero`) are enabled.

For single-precision format numbers, the latency is fixed at 17 clock cycles. Therefore, every exponential operation outputs the results 17 clock cycles later.

Table 8–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 8–2 on page 8–3 and Figure 8–3 on page 8–3.

Table 8–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`data[]` value: 1A03 568Ch Output value: An undefined value is seen on the `result[]` port, which is ignored. All values seen on the output port before the 17th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
82.5 ns	Output value: 3F80 0000h As the input value of 1A03568Ch is a very small number, it is seen as a value that is approaching zero, and the result approaches 1 (which is represented by 3F800000). Exponential operations carried out on numbers of very small magnitudes result in a 1 and assert the `underflow` flag. Exception handling ports: `underflow` asserts
30 ns	`data[]` value: F3FC DEFFh This is a normal negative value of a very large magnitude.
112.5 ns	Output value: 0000 0000h The outcome of exponential operations on negative numbers of very large magnitudes approaches zero. Exception handling ports: `underflow` remains asserted
60 ns	`data[]` value: 7F80 0000h This is a positive infinite value.
142.5 ns	Output value: 7F80 0000h The operation on positive infinite values results in infinity. Exception handling ports: `underflow` deasserts, `overflow` asserts
90 ns	`data[]` value: 7FC0 0000h This is a NaN.
172.5 ns	Output value: 7FC0 0000h The exponential of a NaN results in a NaN. Exception handling ports: `nan` asserts
120 ns	`data[]` value: C1D4 49BAh This is a normal value.
202.5 ns	Output value: 2C52 5981h The result is a normal value. Exception handling ports: `nan` deasserts

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_EXP megafunction. The ports and parameters are available to customize the ALTFP_EXP megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_exp
   generic (
      intended_device_family : string := "unused";
      pipeline : natural := 0;
      rounding : string := "TO_NEAREST";
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_exp"
   );
   port(
      aclr : in std_logic := '0';
      clk_en : in std_logic := '1';
      clock : in std_logic;
      data : in std_logic_vector(width_exp+width_man+1-1 downto 0);
      nan : out std_logic;
      overflow : out std_logic;
      result : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      underflow : out std_logic;
      zero : out std_logic
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
```

```
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 8–5 lists the input ports of the ALTFP_EXP megafunction.

Table 8–5. ALTFP_EXP Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high the function is asynchronously reset.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, an exponential value operation takes place. When this signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction
`data[]`	Yes	Floating-point input data. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits.

Table 8–6 lists the output ports of the ALTFP_EXP megafunction.

Table 8–6. ALTFP_EXP Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The floating-point exponential result of the value at `data[]`. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`overflow`	No	Overflow exception output. Asserted when the result of the operation (after rounding) is infinite.
`underflow`	No	Underflow exception output. Asserted when the result of the exponential approaches `1` (from numbers of very small magnitude), or when the result approaches `0` (from negative numbers of very large magnitudes).
`zero`	No	Zero exception output. Asserted when the value in the `result[]` port is zero.
`nan`	No	NaN exception output. Asserted when an invalid operation occurs. Any operation involving NaN also asserts the `nan` port.

Table 8–7 lists the parameters of the ALTFP_EXP megafunction.

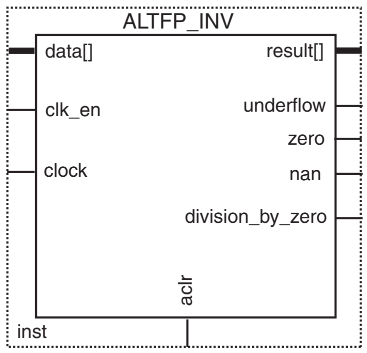
Table 8–7. ALTFP_EXP Megafunction Parameters

Parameter Name	Type	Required	Description
WIDTH_EXP	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is 8. The bias of the exponent is always set to $2^{(WIDTH\_EXP -1)}-1$, that is, 127 for the single-precision format and 1023 for the double-precision format. The value of the WIDTH_EXP parameter must be 8 for the single-precision format, 11 for the double-precision format, and a minimum of 11 for the single-extended precision format. The value of the WIDTH_EXP parameter must be less than the value of the WIDTH_MAN parameter, and the sum of the WIDTH_EXP and WIDTH_MAN parameters must be less than 64.
WIDTH_MAN	Integer	Yes	Specifies the value of the mantissa. If this parameter is not specified, the default is 23. When the WIDTH_EXP parameter is 8 and the floating-point format is single-precision, the WIDTH_MAN parameter value must be 23. Otherwise, the value of the WIDTH_MAN parameter must be a minimum of 31. The value of the WIDTH_MAN parameter must be greater than the value of the WIDTH_EXP parameter. The sum of the WIDTH_EXP and WIDTH_MAN parameters must be less than 64.
PIPELINE	Integer	Yes	Specifies the amount of latency, expressed in clock cycles, used in the ALTFP_EXP megafunction. Acceptable pipeline values are 17, 22, and 25 cycles of latency. Create the ALTFP_EXP megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter.
ROUNDING	String	Yes	Specifies the rounding mode. The default value is TO_NEAREST. Other rounding modes are not supported.

9. ALTFP_INV

Figure 9–1 shows the ports for the ALTFP_INV megafunction.

Figure 9–1. ALTFP_INV Ports



Features

The ALTFP_INV megafunction offers the following features:

- Inverse value of a given input.
- Optional exception handling output ports such as `zero`, `division_by_zero`, `underflow`, and `nan`.

Output Latency

The output latency options for the ALTFP_INV megafunction differs depending on the precision selected, the width of the mantissa, or both.

Table 9–1 lists the output latency options required for each precision format.

Table 9–1. Latency Options for Each Precision Format

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	20
Double	52	27
Single Extended	31 – 39	20
	40 – 52	27

Truth Table

Table 9–2 lists the truth table for the inverse operation.

Table 9–2. Truth Table for Inverse Operations

DATA[]	SIGN BIT	RESULT[]	Underflow	Zero	Division_by_zero	NaN
Normal	0/1	Normal	0	0	0	0
Normal	0/1	Denormal *(1)*	1	1	0	0
Normal	0/1	Infinity	0	0	0	0
Normal	0/1	Zero	1	1	0	0
Denormal *(2)*	0/1	Infinity	0	0	1	0
Zero	0/1	Infinity	0	0	1	0
Infinity	0/1	Zero	0	1	0	0
NaN	X	NaN	0	0	0	1

Notes to Table 9–2:

(1) Any calculated or computed **denormal** output is replaced by a zero and asserts the zero and underflow flags.

(2) Any denormal input is treated as a zero before going through the inverse process.

Resource Utilization and Performance

Table 9–3 lists the resource utilization and performance information for the ALTFP_INV megafunction. The information was derived using the Quartus II software version 10.0.

Table 9–3. ALTFP_INV Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-Bit DSP	
Stratix III	Single	20	413	640	376	16	427
	Double	27	1,049	1,574	990	48	187
Stratix IV	Single	20	401	616	373	16	412
	Double	27	939	1,386	912	48	203

Design Example: Inverse of Single-Precision Format Numbers

This design example uses the ALTFP_INV megafunction to compute the inverse of single-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_INV megafunction:

- **altfp_inv_DesignExample.zip** (Quartus II design files)
- **altfp_inv_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 9–2 and Figure 9–3 show the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 9–2. ModelSim Simulation Waveform (Input Data)

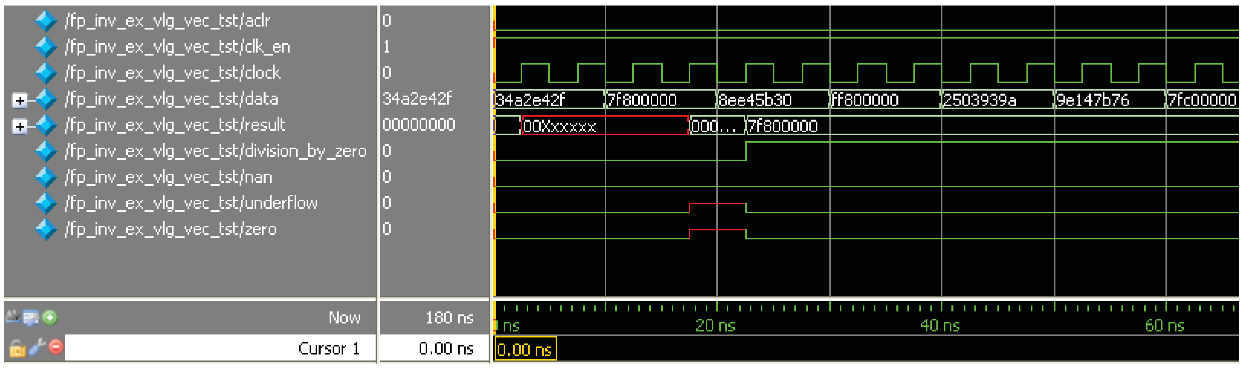
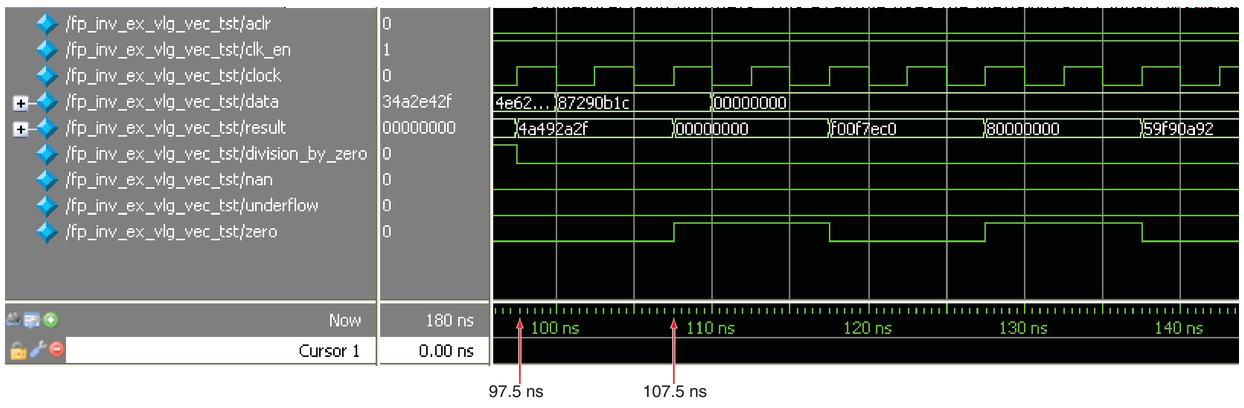

/fp_inv_ex_vlg_vec_tst/aclr 0
/fp_inv_ex_vlg_vec_tst/clk_en 1
/fp_inv_ex_vlg_vec_tst/clock 0
/fp_inv_ex_vlg_vec_tst/data 34a2e42f
/fp_inv_ex_vlg_vec_tst/result 00000000
/fp_inv_ex_vlg_vec_tst/division_by_zero 0
/fp_inv_ex_vlg_vec_tst/nan 0
/fp_inv_ex_vlg_vec_tst/underflow 0
/fp_inv_ex_vlg_vec_tst/zero 0
34a2e42f
7f800000
8ee45b30
ff800000
2503939a
9e147b76
7fc00000
00Xxxxxx
000...
7f800000
Now 180 ns
Cursor 1 0.00 ns
ns
20 ns
40 ns
60 ns
0.00 ns

Figure 9–3. ModelSim Simulation Waveform (Output Data)


/fp_inv_ex_vlg_vec_tst/aclr 0
/fp_inv_ex_vlg_vec_tst/clk_en 1
/fp_inv_ex_vlg_vec_tst/clock 0
/fp_inv_ex_vlg_vec_tst/data 34a2e42f
/fp_inv_ex_vlg_vec_tst/result 00000000
/fp_inv_ex_vlg_vec_tst/division_by_zero 0
/fp_inv_ex_vlg_vec_tst/nan 0
/fp_inv_ex_vlg_vec_tst/underflow 0
/fp_inv_ex_vlg_vec_tst/zero 0
4e62...
87290b1c
00000000
4a492a2f
00000000
f00f7ec0
80000000
59f90a92
Now 180 ns
Cursor 1 0.00 ns
100 ns
110 ns
120 ns
130 ns
140 ns
97.5 ns
107.5 ns

This design example implements a floating-point inverse for single-precision format numbers. The optional input ports (`clk_en` and `aclr`) and all four exception handling output ports (`division_by_zero`, `nan`, `zero`, and `underflow`) are enabled.

The latency is fixed at 20 clock cycles; therefore, every inverse operation outputs results 20 clock cycles later.

Table 9–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 9–2 on page 9–3 and Figure 9–3 on page 9–3.

Table 9–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`data[]` value: 34A2 E42Fh Output value: An undefined value is seen on the `result[]` port, which is ignored. All values seen on the output port before the 20th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
97.5 ns	Output value: 4A49 2A2Fh Exception handling ports: `division_by_zero` deasserts The inverse of a normal number results in a normal value.
10 ns	`data[]` value: 7F80 0000h This is an infinity value.
107.5 ns	Output value: 0000 0000h Exception handling ports: `zero` asserts The inverse of an infinity value produces a zero.
60 ns	`data[]` value: 7FC0 0000h This is a NaN.
157.5 ns	Output value: 7FC0 0000h Exception handling ports: `nan` asserts The inverse of a NaN results in a NaN
70 ns	`data[]` value: 0000 1000h This is a denormal number.
167.5 ns	Output value: 7F80 0000h Exception handling ports: `nan` deasserts, `division_by_zero` asserts Denormal numbers are forced-zero values, therefore, the inverse of a zero results in infinity.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_INV megafunction. The ports and parameters are available to customize the ALTFP_INV megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_inv
   generic (
      intended_device_family : string := "unused";
      pipeline : natural := 20;
      rounding : string := "TO_NEAREST";
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_inv"
   );
   port(
      aclr : in std_logic := '0';
      clk_en : in std_logic := '1';
      clock : in std_logic;
      data : in std_logic_vector(width_exp+width_man+1-1 downto 0);
      division_by_zero : out std_logic;
      nan : out std_logic;
      result : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      underflow : out std_logic;
      zero : out std_logic
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;

USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 9–5 lists the input ports of the ALTFP_INV megafunction.

Table 9–5. ALTFP_INV Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, an inversion value operation takes place. When signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits.

Table 9–6 lists the output ports of the ALTFP_INV megafunction.

Table 9–6. ALTFP_INV Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The floating-point inverse result of the value at the `data[]` input port. The MSB is the sign, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`underflow`	No	Underflow exception output. Asserted when the result of the inversion (after rounding) is a denormalized number.
`zero`	No	Zero exception output. Asserted when the value at the `result[]` port is a zero.
`division_by_zero`	No	Division-by-zero exception output. Asserted when the denominator input is a zero.
`nan`	No	NaN exception output. Asserted when an invalid inversion occurs, such as the inversion of NaN. In this case, a NaN value is output to the `result[]` port. Any operation involving NaN also asserts the `nan` port.

Table 9–7 lists the parameters of the ALTFP_INV megafunction.

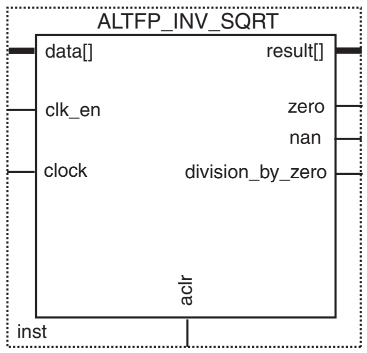
Table 9–7. ALTFP_INV Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to `2` $^{(\texttt{WIDTH\_EXP -1})}$`-1`, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of the `WIDTH_EXP` parameter must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of the `WIDTH_EXP` parameter must be less than the value of the `WIDTH_MAN` parameter, and the sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the value of the mantissa. If this parameter is not specified, the default is `23`. When the `WIDTH_EXP` parameter is `8` and the floating-point format is single-precision, the `WIDTH_MAN` parameter value must be `23`. Otherwise, the value of the `WIDTH_MAN` parameter must be a minimum of `31`. The value of the `WIDTH_MAN` parameter must be greater than the value of the `WIDTH_EXP` parameter. The sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.
`PIPELINE`	Integer	Yes	Specifies the amount of latency in clock cycles used in the ALTFP_INV megafunction. Create the ALTFP_INV megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter.
`ROUNDING`	String	No	Specifies the rounding mode. The default value is `TO_NEAREST`. Other rounding modes are not supported.

10. ALTFP_INV_SQRT

Figure 10–1 shows the ports for the ALTFP_INV_SQRT megafunction.

Figure 10–1. ALTFP_INV_SQRT Ports



Features

The ALTFP_INV_SQRT megafunction offers the following features:

- Inverse square root value of a given input.
- Optional exception handling output ports such as `zero`, `division_by_zero`, and `nan`.

Output Latency

The output latency options for the ALTFP_INV_SQRT megafunction differs depending on the precision selected, the width of the mantissa, or both.

Table 10–1 lists the output latency options required for each precision format.

Table 10–1. Latency Options for Each Precision Format

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	26
Double	52	36
Single-Extended	31– 39	26
	40 – 52	36

Truth Table

Table 10–2 lists the truth table for the inverse square root operation.

Table 10–2. Truth Table for Inverse Square Root Operations

DATA[]	SIGN BIT	RESULT[]	Zero	Division_by_zero	NaN
Normal	0	Normal	0	0	0
Normal	1	NaN	0	0	1
Denormal *(1)*	0/1	Infinity	0	1	0
Zero	0/1	Infinity	0	1	0
Infinity	0/1	Zero	1	0	0
NaN	X	NaN	0	0	1

Note to Table 10–2:

(1) Any denormal input is treated as a zero before going through the inverse process.

Resource Utilization and Performance

Table 10–3 lists the resource utilization and performance information for the ALTFP_INV_SQRT megafunction. The information was derived using the Quartus II software version 10.0.

Table 10–3. ALTFP_INV_SQRT Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
			Adaptive Look-up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-Bit DSP	
Stratix III	Single	26	510	674	439	22	428
	Double	36	1,345	1,866	1,215	78	193
Stratix IV	Single	26	502	658	430	22	413
	Double	36	1,324	1,855	1,209	78	209

Design Example: Inverse Square Root of Single-Precision Format Numbers

This design example uses the ALTFP_INV_SQRT megafunction to compute the inverse square root of single-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_INV_SQRT megafunction:

- **altfp_inv_sqrt_DesignExample.zip** (Quartus II design files)
- **altfp_inv_sqrt_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 10–2 and Figure 10–3 show the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 10–2. ModelSim Simulation Waveform (Input Data)

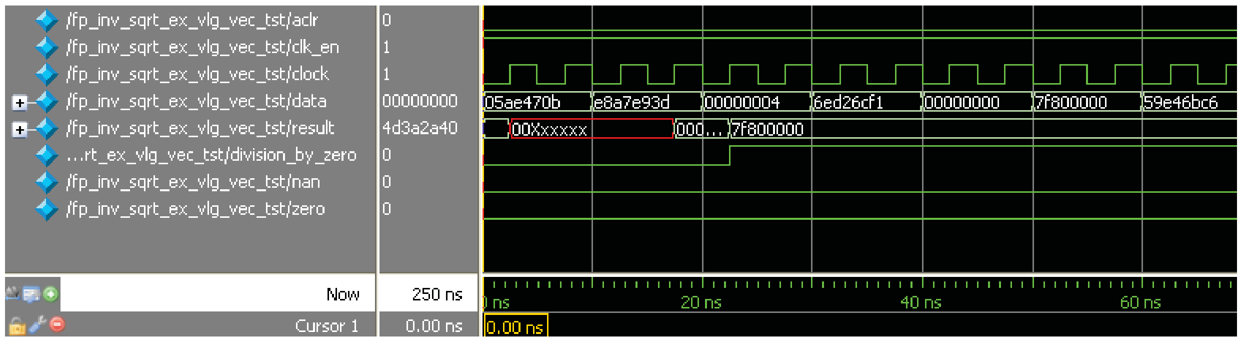
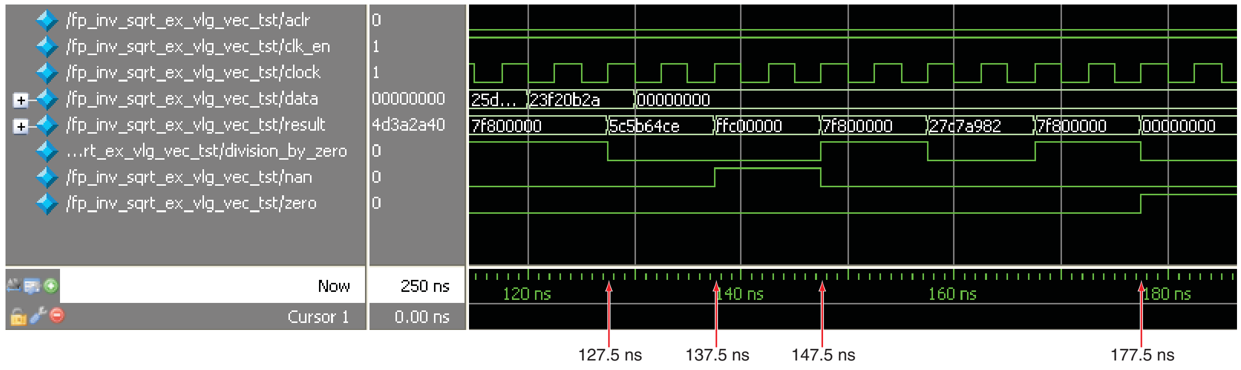

/fp_inv_sqrt_ex_vlg_vec_tst/aclr 0
/fp_inv_sqrt_ex_vlg_vec_tst/clk_en 1
/fp_inv_sqrt_ex_vlg_vec_tst/clock 1
/fp_inv_sqrt_ex_vlg_vec_tst/data 00000000
/fp_inv_sqrt_ex_vlg_vec_tst/result 4d3a2a40
...rt_ex_vlg_vec_tst/division_by_zero 0
/fp_inv_sqrt_ex_vlg_vec_tst/nan 0
/fp_inv_sqrt_ex_vlg_vec_tst/zero 0
05ae470b
e8a7e93d
00000004
6ed26cf1
00000000
7f800000
59e46bc6
00Xxxxxx
000...
7f800000
Now 250 ns
Cursor 1 0.00 ns
0 ns
20 ns
40 ns
60 ns
0.00 ns

Figure 10–3. ModelSim Simulation Waveform (Output Data)


/fp_inv_sqrt_ex_vlg_vec_tst/aclr 0
/fp_inv_sqrt_ex_vlg_vec_tst/clk_en 1
/fp_inv_sqrt_ex_vlg_vec_tst/clock 1
/fp_inv_sqrt_ex_vlg_vec_tst/data 00000000
/fp_inv_sqrt_ex_vlg_vec_tst/result 4d3a2a40
...rt_ex_vlg_vec_tst/division_by_zero 0
/fp_inv_sqrt_ex_vlg_vec_tst/nan 0
/fp_inv_sqrt_ex_vlg_vec_tst/zero 0
25d...
23f20b2a
00000000
7f800000
5c5b64ce
ffc00000
7f800000
27c7a982
7f800000
00000000
Now 250 ns
Cursor 1 0.00 ns
120 ns
140 ns
160 ns
180 ns
127.5 ns
137.5 ns
147.5 ns
177.5 ns

This design example implements a floating-point inverse square root for single-precision format numbers. The optional input ports (`clk_en` and `aclr`) and all three exception handling output ports (`division_by_zero`, `nan`, and `zero`) are enabled.

The latency is fixed at 26 clock cycles. Therefore, every inverse square root operation outputs the results 26 clock cycles later.

Table 10–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 10–2 on page 10–3 and Figure 10–3 on page 10–3.

Table 10–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`data[]` value: 05AE 470Bh Output value: An undefined value is seen on the `result[]` port, which can be ignored. All values seen on the output port before the 26th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
127.5 ns	Output value: 5C5B 64CEh The inverse square root of a normal number results in a normal value.
10 ns	`data[]` value: E8A7 E93Dh This is a negative normal value.
137.5 ns	Output value: FFC0 0000h Exception handling ports: `nan` asserts The inverse square root of a negative value produces a NaN.
20 ns	`data[]` value: 0000 0004h The is a denormal value.
147.5 ns	Output value: 7F80 0000h Denormal numbers are forced-zero values, therefore the inverse square root of zero results in infinity. Exception handling ports: `nan` deasserts, `division_by_zero` asserts
50 ns	`data[]` value: 7F80 0000h This is an infinity value.
177.5 ns	Output value: 0000 0000h The inverse square root of an infinity value produces a zero. Exception handling ports: `zero` asserts

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_INV_SQRT megafunction. The ports and parameters are available to customize the ALTFP_INV_SQRT megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_inv_sqrt
   generic (
      intended_device_family : string := "unused";
      pipeline : natural := 26;
      rounding : string := "TO_NEAREST";
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_inv_sqrt"
   );
   port(
      aclr : in std_logic := '0';
      clk_en : in std_logic := '1';
      clock : in std_logic;
      data : in std_logic_vector(width_exp+width_man+1-1 downto 0);
      division_by_zero : out std_logic;
      nan : out std_logic;
      result : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      zero : out std_logic
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 10–5 lists the input ports of the ALTFP_INV_SQRT megafunction.

Table 10–5. ALTFP_INV_SQRT Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, an inversion value operation takes place. When signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits.

Table 10–6 lists the output ports of the ALTFP_INV_SQRT megafunction.

Table 10–6. ALTFP_INV_SQRT Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The floating-point inverse result of the value at the `data[]` input port. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. The size of this port is the total width of the sign bit, exponent bits, and mantissa bits.
`zero`	No	Zero exception output. Asserted when the value at the `result[]` port is a zero.
`division_by_zero`	No	Division-by-zero exception output. Asserted when the denominator input is a zero.
`nan`	No	NaN exception output. Asserted when an invalid inversion of square root occurs, such as the square root of a negative number. In this case, a NaN value is output to the `result[]` output port. Any operation involving a NaN will also produce a NaN.

Table 10–7 lists the parameters of the ALTFP_INV_SQRT megafunction.

Table 10–7. ALTFP_INV_SQRT Megafunction Parameters (Part 1 of 2)

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to $2^{(WIDTH\_EXP -1)}-1$, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of the `WIDTH_EXP` parameter must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of the `WIDTH_EXP` parameter must be less than the value of the `WIDTH_MAN` parameter, and the sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the value of the mantissa. If this parameter is not specified, the default is `23`. When the `WIDTH_EXP` parameter is `8` and the floating-point format is single-precision, the `WIDTH_MAN` parameter value must be `23`. Otherwise, the value of the `WIDTH_MAN` parameter must be a minimum of `31`. The value of the `WIDTH_MAN` parameter must be greater than the value of the `WIDTH_EXP` parameter. The sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.

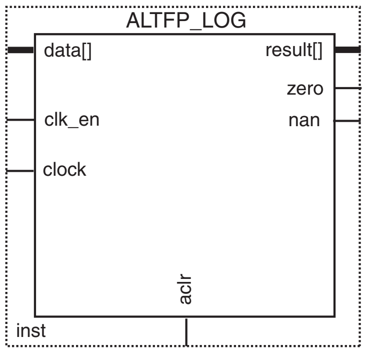
Table 10–7. ALTFP_INV_SQRT Megafunction Parameters (Part 2 of 2)

Parameter Name	Type	Required	Description
`PIPELINE`	Integer	Yes	Specifies the amount of latency, expressed in clock cycles, used in the ALTFP_INV_SQRT megafunction. Create the ALTFP_INV_SQRT megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter.
`ROUNDING`	String	No	Specifies the rounding mode. The default value is `TO_NEAREST`. Other rounding modes are not supported.

11. ALTFP_LOG

Figure 11–1 shows the ports for the ALTFP_LOG megafunction.

Figure 11–1. ALTFP_LOG Ports



Features

The ALTFP_LOG megafunction offers the following features:

- Natural logarithm functions.
- Optional exception handling output ports such as `zero` and `nan`.

Output Latency

The output latency options for the ALTFP_LOG megafunction differs depending on the precision selected, the width of the mantissa, or both.

Table 11–1 lists the output latency options required for each precision format.

Table 11–1. Latency Options for Each Precision Format

Precision	Mantissa Width	Latency (in clock cycles)
Single	23	21
Double	52	34
Single Extended	31–36	25
	37–42	28
	43–48	31
	49–52	34

Truth Table

Table 11–2 lists the truth table for the natural logarithm operation.

Table 11–2. Truth Table for Natural Logarithm Operations

DATA[]	SIGN BIT	RESULT[]	Zero	NaN
Normal	0	Normal	0	0
Normal	1	NaN (1)	0	1
1 (2)	0	Zero	1	0
Denormal (3)	0	Negative Infinity	0	0
Zero (4)	0/1	Negative Infinity	0	0
Infinity	0	Positive Infinity	1	0
NaN	X	NaN	0	1

Notes to Table 11–2:

(1) The natural logarithm of a negative value is invalid. Therefore, the output produced is a NaN.

(2) The “1” in this case is equivalent to *ln* 1.

(3) The value of positive denormalized numbers is a value that approximates zero, and the output produced is a negative infinity number.

(4) The zero in this case represents zero special case of the IEEE standard. It is not equivalent to *ln* 0, but instead approximates to it.

Resource Utilization and Performance

Table 11–3 lists the resource utilization and performance information for the ALTFP_LOG megafunction. The information was derived using the Quartus II software version 10.0.

Table 11–3. ALTFP_LOG Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-Bit DSP	
Stratix III	Single	21	1,983	1,904	1,387	8	338
	Double	34	5,475	6,049	4,171	64	160
Stratix IV	Single	21	1,950	1,864	1,378	8	385
	Double	34	5,451	6,031	4,151	64	211

Design Example: Natural Logarithm of Single-Precision Format Numbers

This design example uses the ALTFP_LOG megafunction to compute the natural logarithm of single-precision format numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following file is related to the ALTFP_LOG megafunction:

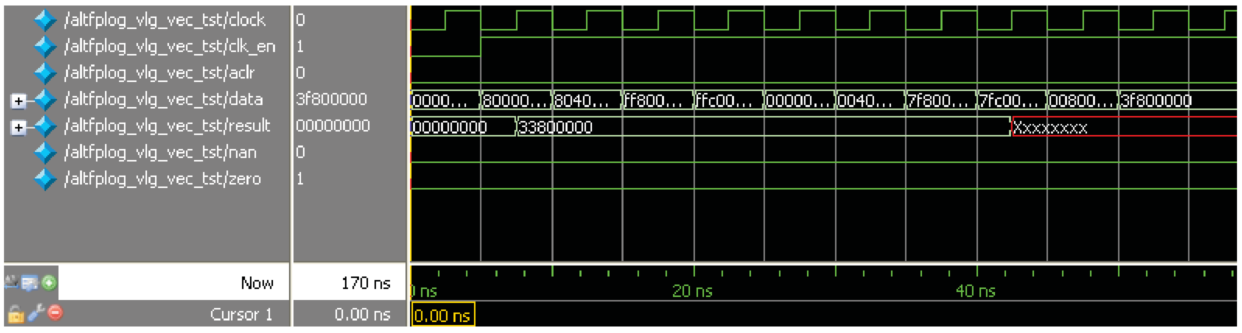
- **altfp_log_DesignExample.zip** (Quartus II design files)
- **altfp_log_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 11–2 and Figure 11–3 show the expected simulation results in the ModelSim-Altera software.

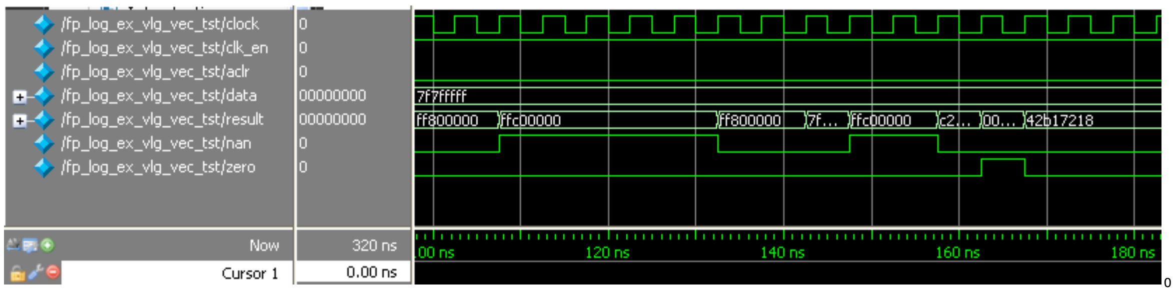
The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 11–2. ModelSim Simulation Waveform (Input Data)



```
/altfplog_vlg_vec_tst/clock      0
/altfplog_vlg_vec_tst/clk_en     1
/altfplog_vlg_vec_tst/aclr       0
/altfplog_vlg_vec_tst/data       3f800000
/altfplog_vlg_vec_tst/result     00000000
/altfplog_vlg_vec_tst/nan        0
/altfplog_vlg_vec_tst/zero       1
Now       170 ns
Cursor 1  0.00 ns
```

Figure 11–3. ModelSim Simulation Waveform (Output Data)



```
/fp_log_ex_vlg_vec_tst/clock     0
/fp_log_ex_vlg_vec_tst/clk_en    0
/fp_log_ex_vlg_vec_tst/aclr      0
/fp_log_ex_vlg_vec_tst/data      00000000
/fp_log_ex_vlg_vec_tst/result    00000000
/fp_log_ex_vlg_vec_tst/nan       0
/fp_log_ex_vlg_vec_tst/zero      0
Now       320 ns
Cursor 1  0.00 ns
```

This design example includes the input of special cases to show the exception handling of the megafunction, such as the smallest valid input and the input value of "1".

In this example, the output delay is set to 21 clock cycles. Therefore, the result is only shown at the output port after the 21st clock cycle at 102.5 ns.

Table 11–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 11–2 on page 11–3 and Figure 11–3 on page 11–3.

Table 11–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`data[]` value: 0000 0000h Output value: An undefined value is seen on the `result[]` port, which is ignored. All values seen on the output port before the 21st clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
102.5 ns	Output value: FF80 0000h The natural logarithm of zero is negative infinity.
5 ns	`data[]` value: 8000 0000h This is a negative number.
107.5 ns	Output value: FFC0 0000h Exception handling ports: `nan` asserts The natural logarithm of a negative value is invalid. Therefore, the output produced is a NaN.
30 ns	`data[]` value: 0040 0000h The is a denormal value.
132.5 ns	Output value: FF80 0000h As denormal numbers are not supported, the input is forced to zero before going through the logarithm function. The natural logarithm of zero is negative infinity.
45 ns	`data[]` value: 0080 0000h This is the smallest valid input. All the input bits are 0 except the LSB of the exponent field.
147.5 ns	Output value: C2AE AC50h
60 ns	`data[]` value: 3F80 0000h The input value 3F80 0000h is equivalent to the actual value, $1.0 \times 2^0 = 1$.
152.5 ns	Output value: 0000 0000h Exception handling ports: `zero` asserts Since *ln* 1 results in zero, it produces an output of zero.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_LOG megafunction. The ports and parameters are available to customize the ALTFP_LOG megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_log
   generic (
      intended_device_family : string := "unused";
      pipeline : natural := 21;
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_log"
);
   port(
      aclr : in std_logic := '0';
      clk_en : in std_logic := '1';
      clock : in std_logic;
      data : in std_logic_vector(width_exp+width_man+1-1 downto 0);
      nan : out std_logic;
      result : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      zero : out std_logic
);
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;

USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 11–5 lists the input ports of the ALTFP_LOG megafunction.

Table 11–5. ALTFP_LOG Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, a natural logarithm operation takes place. When signal is asserted low, no operation occurs and the outputs remain unchanged. Deasserting `clk_en` halts operation until it is asserted again. Assert the `clk_en` signal for the number of clock cycles equivalent to the required output latency (`PIPELINE` parameter value) for the results to be shown at the output.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits. For single precision, the width is fixed to 32 bits. For double precision, the width is fixed to 64 bits. For single extended precision, you can choose a width in the range from 43 to 64 bits.

Table 11–6 lists the output ports of the ALTFP_LOG megafunction.

Table 11–6. ALTFP_LOG Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The natural logarithm of the value on input data. The natural logarithm of the `data[]` input port, shown in floating-point format. The widths of the `result[]` output port and `data[]` input port are the same.
`zero`	No	Zero exception output. Asserted when the exponent and mantissa of the output port are zero. This occurs when the actual input value is `1` because *ln* `1 = 0`.
`nan`	No	NaN exception output. Asserted when the exponent and mantissa of the output port are all 1's and non-zero, respectively. This occurs when the input is a negative number or NaN.

Table 11–7 describes the parameters of the ALTFP_LOG megafunction.

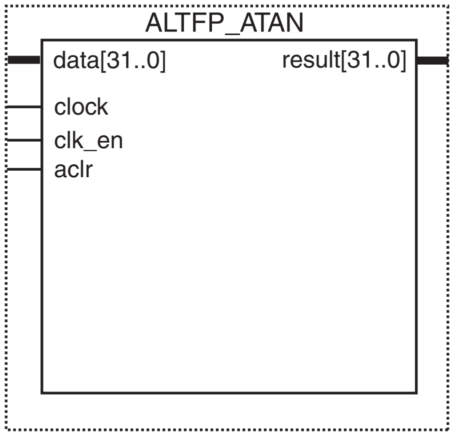
Table 11–7. ALTFP_LOG Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to `2` $^{(\texttt{WIDTH\_EXP -1})}$`-1`, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of the `WIDTH_EXP` parameter must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of the `WIDTH_EXP` parameter must be less than the value of the `WIDTH_MAN` parameter, and the sum of the `WIDTH_EXP` and `WIDTH_MAN` parameters must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is `23`. The value of `WIDTH_MAN` must be `23` for the single-precision format, and `52` for the double-precision format. For the single-extended precision format, the valid value ranges from `31` to `52`. The value of `WIDTH_MAN` must be greater than the value of `WIDTH_EXP`, and the sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`PIPELINE`	Integer	Yes	Specifies the amount of latency in clock cycles used in the ALTFP_LOG megafunction. Create the ALTFP_LOG megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter.

12. ALTFP_ATAN

Figure 12–1 shows the ports for the ALTFP_ATAN megafunction.

Figure 12–1. ALTFP_ATAN Ports



Features

The ALTFP_ATAN megafunction offers the following features:

- Arctangent value of a given angle, θ in unit radian.
- Support for single-precision floating point format.
- Support for optional input ports such as asynchronous clear (`aclr`) and clock enable (`clk_en`) ports.

Output Latency

The output latency option for the ALTFP_ATAN megafunction have a fixed latency level for single-precision format.

Table 12–1 lists the output latency option required for single precision format.

Table 12–1. Latency Option

Trigonometric Function	Precision	Mantissa Width	Latency (in clock cycles)
Arctangent	Single	23	34

Resource Utilization and Performance

Table 12–2 lists the resource utilization and performance information for the ALTFP_ATAN megafunction. The information was derived using the Quartus II software version 11.0.

Table 12–2. ALTFP_ATAN Resource Utilization and Performance

Device Family	Function	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
				Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-Bit DSP	
Stratix V	ArcTangent	Single	36	2,454	1,010	1,303	27	255.49

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_ATAN megafunction. The ports and parameters are available to customize the ALTFP_ATAN megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_atan
   generic (
      intended_device_family:string := "unused";
      pipeline:natural := 36;
      rounding:string := "TO_NEAREST";
      width_exp:natural := 8;
      width_man:natural := 23;
      lpm_hint:string := "UNUSED";
      lpm_type:string := "altfp_atan"
   );
   port(
      aclr:  in std_logic := '0';
      clk_en:in std_logic := '0';
      clock: in std_logic := '0';
      data:  in std_logic_vector(width_exp+width_man+1-1 downto 0) := (others => '0');
      result:out std_logic_vector(width_exp+width_man+1-1 downto 0)
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 12–3 lists the input ports of the ALTFP_ATAN megafunction.

Table 12–3. ALTFP_ATAN Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, division takes place. When the signal is deasserted, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits.

Table 12–4 lists the output ports of the ALTFP_ATAN megafunction.

Table 12–4. ALTFP_ATAN Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The result of the trigonometric function in floating-point format. The widths of the `result[]` output port and `data[]` input port are the same.

Table 12–5 lists the parameters of the ALTFP_ATAN megafunction.

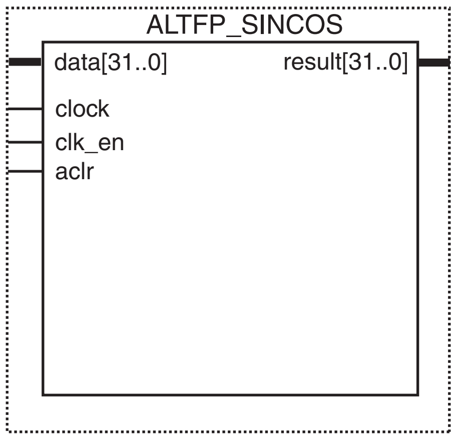
Table 12–5. ALTFP_ATAN Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. The bias of the exponent is always set to `2`(`WIDTH_EXP`-1) -1 (that is, `127` for single-precision format). The value of `WIDTH_EXP` must be `8` for single-precision format. The default value for `WIDTH_EXP` is `8`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. The value of `WIDTH_MAN` must be `23` when `WIDTH_EXP` is `8`. The default value for `WIDTH_MAN` is `23`.
`PIPELINE`	Integer	Yes	The number of pipeline is fixed for the mantissa width and some internal parameter. For the correct settings, refer to Table 12–1 on page 12–1.
`ROUNDING`	Integer	No	Specifies the rounding mode. The default value is `TO_NEAREST`. Other rounding modes are not supported.

13. ALTFP_SINCOS

Figure 13–1 shows the ports for the ALTFP_SINCOS megafunction.

Figure 13–1. ALTFP_SINCOS Ports



Features

The ALTFP_SINCOS megafunction offers the following features:

- Implements sine and cosine calculations.
- Support for single-precision floating point format.
- Support for optional input ports such as asynchronous clear (`aclr`) and clock enable (`clk_en`) ports.

Output Latency

The output latency options for the ALTFP_SINCOS megafunction have a fixed latency level for sine and cosine functions.

Table 13–1 lists the output latency options required for each precision format.

Table 13–1. Latency Options for Each Precision Format

Trigonometric Function	Precision	Mantissa Width	Latency (in clock cycles)
Sine	Single	23	36
Cosine	Single	23	35

Resource Utilization and Performance

Table 13–2 lists the resource utilization and performance information for the ALTFP_SINCOS megafunction. The information was derived using the Quartus II software version 10.1.

Table 13–2. ALTFP_SINCOS Resource Utilization and Performance

Device Family	Function	Precision	Output Latency	Logic usage				f_{MAX} (MHz)
				Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	18-Bit DSP	
Stratix IV	Sine	Single	36	2,859	2,190	1,830	16	292.96
	Cosine	Single	35	2,753	2,041	1,745	16	258.26

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_SINCOS megafunction. The ports and parameters are available to customize the ALTFP_SINCOS megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_sincos
   generic (
      cordic_depth:natural := 0;
      cordic_width:natural := 0;
      intended_device_family:string := "unused";
      indexpoint:natural := 0;
      operation:string;
      pipeline:natural := 20;
      rounding:string := "TO_NEAREST";
      width_exp:natural := 8;
      width_man:natural := 23;
      lpm_hint:string := "UNUSED";
      lpm_type:string := "altfp_sincos"
   );
   port(
      aclr:  in std_logic := '0';
      clk_en:in std_logic := '1';
      clock: in std_logic;
      data:  in std_logic_vector(width_exp+width_man+1-1 downto 0);
      nan :  out std_logic;
      result:out std_logic_vector(width_exp+width_man+1-1 downto 0);
      zero:  out std_logic
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;

USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 13–3 lists the input ports of the ALTFP_SINCOS megafunction.

Table 13–3. ALTFP_SINCOS Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, sine or cosine operation takes place. When the signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of the sign bit, exponent bits, and mantissa bits.

Table 13–4 lists the output ports of the ALTFP_SINCOS megafunction.

Table 13–4. ALTFP_SINCOS Megafunction Output Ports

Port Name	Required	Description
`result[]`	Yes	The trigonemetric of the `data[]` input port in floating-point format. The widths of the `result[]` output port and `data[]` input port are the same.

Table 13–5 lists the parameters of the ALTFP_SINCOS megafunction.

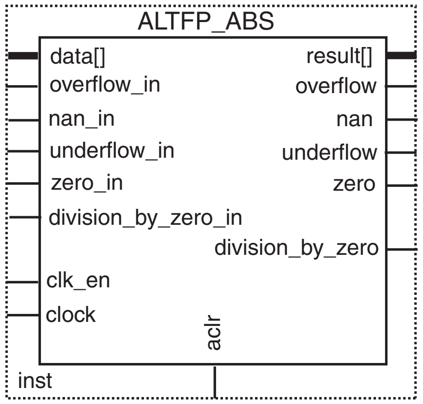
Table 13–5. ALTFP_SINCOS Megafunction Parameters

Parameter Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. The bias of the exponent is always set to `2`(`WIDTH_EXP`-1) -1 (that is, `127` for single-precision format). The value of `WIDTH_EXP` must be `8` for single-precision format and must be less than `WIDTH_MAN`. The available value for `WIDTH_EXP` is `8`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. The value of `WIDTH_MAN` must be `23` when `WIDTH_EXP` is `8`. Otherwise, `WIDTH_MAN` must be a minimum of `31`. The value of `WIDTH_MAN` must be greater than `WIDTH_EXP`. The available value for `WIDTH_MAN` is `23`.
`PIPELINE`	Integer	Yes	The number of pipeline is fixed for the mantissa width and some internal parameter. For the correct settings, refer to Table 13–1 on page 13–1.

14. ALTFP_ABS

Figure 14–1 shows the ports for the ALTFP_ABS megafunction.

Figure 14–1. ALTFP_ABS Ports



Features

The ALTFP_ABS megafunction offers the following features:

- Absolute value of a given input.
- Optional exception handling output ports such as `zero`, `division_by_zero`, `overflow`, `underflow`, and `nan`.
- Carry-through exception ports from other floating-point modules that act as inputs to the ALTFP_ABS megafunction.

Output Latency

The output latency options for the ALTFP_ADD_SUB megafunction are the same for all three precision formats—single, double, and single-extended. The options available are zero without pipeline, and 1 clock cycle.

Resource Utilization and Performance

Table 14–1 lists the resource utilization and performance information for the ALTFP_ABS megafunction. The information was derived using the Quartus II software version 10.0.

Table 14–1. ALTFP_ABS Resource Utilization and Performance for the Stratix III Device Family

Precision	Output Latency	Logic usage				f_{MAX} (MHz)
		Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	18-Bit DSP	Memory	
Single	0	0	0	0	0	The f_{MAX} of this megafunction depends on the speed of the selected device
	1	0	36	0	0	
Double	0	0	0	0	0	
	1	0	68	0	0	

Design Example: Absolute Value of Multiplication Results

This design example uses the ALTFP_ABS megafunction to compute the absolute value of the multiplication result of single-precision format numbers. This example incorporates the ALTFP_MULT megafunction and uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_ABS megafunction:

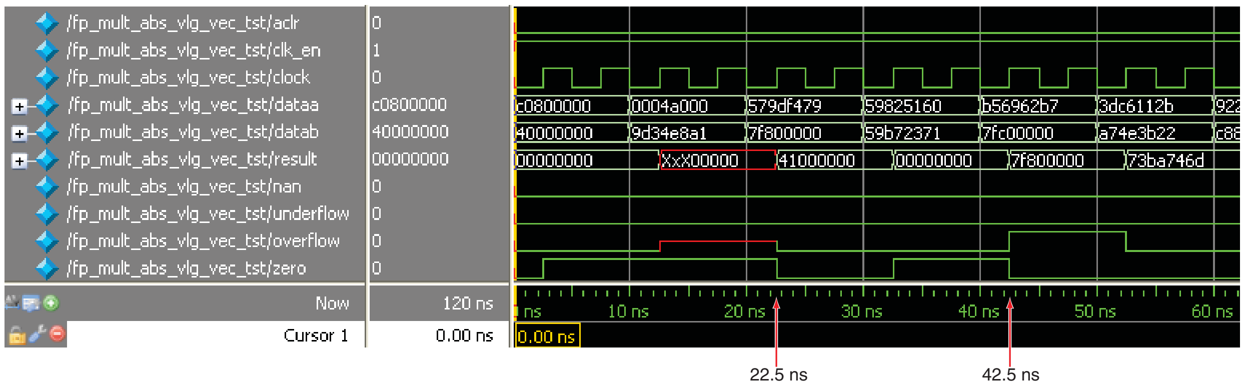
- **altfp_mult_abs_DesignExample.zip** (Quartus II design files)
- **altfp_mult_abs_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Table 14–2 on page 14–3 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Table 14–2. ALTFP_ABS Simulation Waveform



```
/fp_mult_abs_vlg_vec_tst/aclr           0
/fp_mult_abs_vlg_vec_tst/clk_en         1
/fp_mult_abs_vlg_vec_tst/clock          0
/fp_mult_abs_vlg_vec_tst/dataa          c0800000
/fp_mult_abs_vlg_vec_tst/datab          40000000
/fp_mult_abs_vlg_vec_tst/result         00000000
/fp_mult_abs_vlg_vec_tst/nan            0
/fp_mult_abs_vlg_vec_tst/underflow      0
/fp_mult_abs_vlg_vec_tst/overflow       0
/fp_mult_abs_vlg_vec_tst/zero           0
Now                                     120 ns
Cursor 1                                0.00 ns
```

c0800000 0004a000 579df479 59825160 b56962b7 3dc6112b 922
40000000 9d34e8a1 7f800000 59b72371 7fc00000 a74e3b22 c88
00000000 XxX00000 41000000 00000000 7f800000 73ba746d
ns 10 ns 20 ns 30 ns 40 ns 50 ns 60 ns
0.00 ns
22.5 ns
42.5 ns

This design example produces a floating-point absolute value function for the multiplication results of single-precision format numbers. All the optional input ports (`clk_en` and `aclr`) and optional output ports (`overflow`, `underflow`, `zero`, `division_by_zero`, and `nan`) are enabled.

In this example, the latency of the multiplier is set to five clock cycles, while none is being set for the absolute value function. Thus, the absolute value result only appears at the `result[]` port five cycles after the input values are captured on the input ports.

The `dataa[]` and `datab[]` values in the simulation waveform above portray the two input values that are being fed to the multiplier. The value in the `result[]` port depicts the multiplication result that has gone through the absolute value operation.

Table 14–3 lists the inputs and corresponding outputs obtained from the simulation in Table 14–2.

Table 14–3. Summary of Input Values and Corresponding Outputs (Part 1 of 2)

Time	Event
0 ns, start-up	`dataa[]` value: C080 0000h `datab[]` value: 4000 0000h Output value: All values seen on the output port before the 5th clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
22.5 ns	Output value: 4100 0000h The multiplication of a negative number with a positive number results in a negative number. The absolute value of the result is reflected on the `result[]` port.

Table 14–3. Summary of Input Values and Corresponding Outputs (Part 2 of 2)

Time	Event
20 ns	`dataa[]` value: 579D F479h `datab[]` value: 7F80 0000h The value of `dataa[]` is normal while the value of `datab[]` is infinity.
42.5 ns	Output value: 7F80 0000h Exception handling ports: `overflow` asserts The multiplication of a normal value with infinity results in infinity and sets the `overflow` port in the multiplier. The absolute value of the output is infinity and the `overflow` port is also set as this assertion of the port is being carried through from the corresponding `overflow` port in the multiplier.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_ABS megafunction. The ports and parameters are available to customize the ALTFP_ABS megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_abs
   generic (
      intended_device_family : string := "unused";
      pipeline : natural := 0;
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_abs"
   );
port(
      aclr : in std_logic := '0';
      clk_en : in std_logic := '1';
      clock : in std_logic := '0';
      data : in std_logic_vector(width_exp+width_man+1-1 downto 0);
      division_by_zero : out std_logic;
      division_by_zero_in : in std_logic := '0';
      nan : out std_logic;
      nan_in : in std_logic := '0';
      overflow : out std_logic;
      overflow_in : in std_logic := '0';
      result : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      underflow : out std_logic;
      underflow_in : in std_logic := '0';
      zero : out std_logic;
      zero_in : in std_logic := '0'
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 14–4 lists the input ports of the ALTFP_ABS megafunction.

Table 14–4. ALTFP_ABS Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clk_en`	No	Clock enable. When the `clk_en` port is asserted high, an absolute value operation takes place. When the signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`data[]`	Yes	Floating-point input data. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.
`zero_in`	No	Zero exception input. Carry-through exception input port from other floating-point modules.
`nan_in`	No	NaN exception input. Carry-through exception input port from other floating-point modules.
`overflow_in`	No	Overflow exception input. Carry-through exception input port from other floating-point modules.
`underflow_in`	No	Underflow exception input. Carry-through exception input port from other floating-point modules.
`division_by_zero_in`	No	Division-by-zero exception input. Carry-through exception input port from other floating-point modules.

Table 14–5 lists the output ports of the ALTFP_ABS megafunction.

Table 14–5. ALTFP_ABS Megafunction Output Ports (Part 1 of 2)

Port Name	Required	Description
`result[]`	Yes	The absolute value result of the input data. The size of this port corresponds to the size of the input `data[]` port.
`zero`	No	Zero exception output carried from the input. Asserted if the corresponding carry-through port from the input is asserted.
`nan`	No	NaN output carried from the input. Asserted if the corresponding carry-through port from the input is asserted.
`overflow`	No	Overflow exception output carried from the input. Asserted if the corresponding carry-through port from the input is asserted.

Table 14–5. ALTFP_ABS Megafunction Output Ports (Part 2 of 2)

Port Name	Required	Description
`underflow`	No	Underflow exception output carried from the input. Asserted if the corresponding carry-through port from the input is asserted.
`division_by_zero`	No	Division-by-zero exception output carried from the input. Asserted if the corresponding carry-through port from the input is asserted.

Table 14–6 lists the parameters of the ALTFP_ABS megafunction.

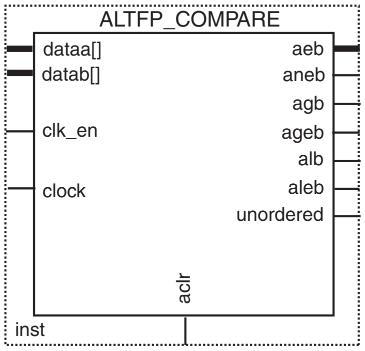
Table 14–6. ALTFP_ABS Megafunction Parameters

Port Name	Type	Required	Description
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to $2^{(\texttt{WIDTH\_EXP} - 1)} - 1$, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of `WIDTH_EXP` must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of `WIDTH_EXP` must be less than the value of `WIDTH_MAN`, and the sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is `23`. When `WIDTH_EXP` 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 value of `WIDTH_MAN` must be greater than the value of `WIDTH_EXP`, and the sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than `64`.
`PIPELINE`	Integer	Yes	Specifies the amount of latency, expressed in clock cycles, used in the ALTFP_ABS megafunction. Create the ALTFP_ABS megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter.

15. ALTFP_COMPARE

Figure 15–1 shows the ports for the ALTFP_COMPARE megafunction.

Figure 15–1. ALTFP_COMPARE Ports



Features

The ALTFP_COMPARE megafunction offers the following features:

- Comparison functions between two inputs.
- Seven status output ports:
  - `aeb` (input A is equal to input B).
  - `aneb` (input A is not equal to input B).
  - `agb` (input A is greater than input B).
  - `ageb` (input A is greater than or equal to input B).
  - `alb` (input A is less than input B).
  - `aleb` (input A is less than or equal to input B).
  - `unordered` (used as an output to flag if one or both input ports are NaN).

Output Latency

The output latency options for the ALTFP_COMPARE megafunction are the same for all three precision formats—single, double, and single-extended. The options available are 1, 2, and 3 clock cycles.

Resource Utilization and Performance

Table 15–1 lists the resource utilization and performance information for the ALTFP_COMPARE megafunction. The information was derived using the Quartus II software version 10.0.

Table 15–1. ALTFP_COMPARE Resource Utilization and Performance for the Stratix Series of Devices

Device Family	Precision	Output Latency	Logic usage			f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Look-Up Modules (ALMs)	
Stratix III	single	3	68	33	49	764
	double	3	121	47	84	708
Stratix IV	single	3	68	33	47	794
	double	3	121	47	87	680

Design Example: Comparison of Single-precision Format Numbers

This design example uses the ALTFP_COMPARE megafunction to implement the comparison of single-precision format numbers using the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_COMPARE megafunction:

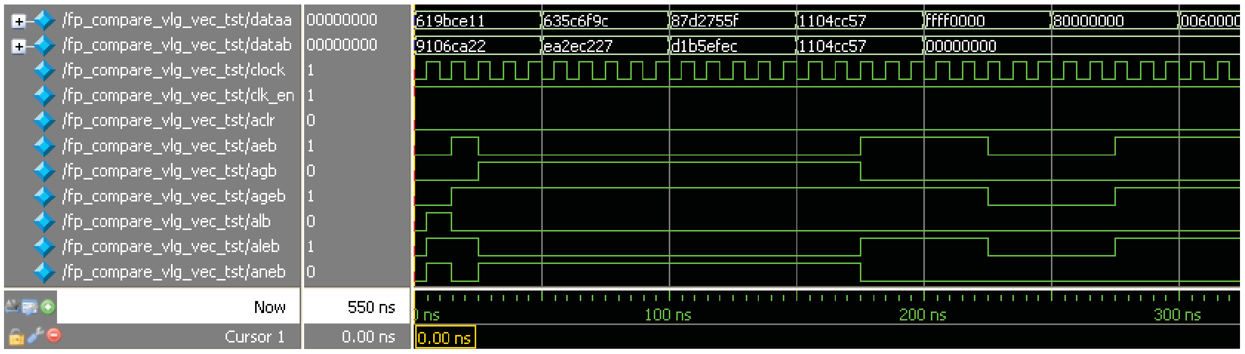
- **altfp_compare_DesignExample.zip** (Quartus II design files)
- **altfp_compare_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 15–2 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 15–2. ALTFP_COMPARE Simulation Waveform



/fp_compare_vlg_vec_tst/dataa 00000000
/fp_compare_vlg_vec_tst/datab 00000000
/fp_compare_vlg_vec_tst/clock 1
/fp_compare_vlg_vec_tst/clk_en 1
/fp_compare_vlg_vec_tst/aclr 0
/fp_compare_vlg_vec_tst/aeb 1
/fp_compare_vlg_vec_tst/agb 0
/fp_compare_vlg_vec_tst/ageb 1
/fp_compare_vlg_vec_tst/alb 0
/fp_compare_vlg_vec_tst/aleb 1
/fp_compare_vlg_vec_tst/aneb 0
619bce11 635c6f9c 87d2755f 1104cc57 ffff0000 80000000 0060000
9106ca22 ea2ec227 d1b5efec 1104cc57 00000000
Now 550 ns
Cursor 1 0.00 ns
0 ns 100 ns 200 ns 300 ns
0.00 ns

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

The chosen output latency is 3. Therefore, the comparison operation generates the output result 3 clock cycles later.

Table 15–2 lists the inputs and corresponding outputs obtained from the simulation in Figure 15–2.

Table 15–2. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`dataa[]` value: 619B CE11h `datab[]` value: 9106 CA22h Output value: An undefined value is seen on the `result[]` port, which is ignored. All values seen on the output port before the 3rd clock cycle are merely due to the behavior of the system during start-up and should be disregarded.
25 ns	Output ports: `ageb`, `aneb`, and `agb` assert
350 ns	`dataa[]` value: 0060 0000h `datab[]` value: 0070 0000h Both input values are denormal numbers.
375 ns	Output ports: `aeb`, `ageb`, and `aleb` assert Denormal inputs are not supported and are forced to zero before comparison takes place, which results in the `dataa[]` value being equal to `datab[]`.
460 ns	The `aclr` signal is set for 1 clock cycle.
495.5 ns	The comparisons of subsequent data inputs are performed 3 clock cycles after the `aclr` signal deasserts.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_COMPARE megafunction. The ports and parameters are available to customize the ALTFP_COMPARE megafunction according to your application.

VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 15–3 lists the input ports of the ALTFP_COMPARE megafunction.

Table 15–3. ALTFP_COMPARE Megafunction Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear. The source is asynchronously reset when asserted high.
`clk_en`	No	Clock enable. When this port is asserted high, a compare operation takes place. When signal is asserted low, no operation occurs and the outputs remain unchanged.
`clock`	Yes	Clock input to the megafunction.
`dataa[]`	Yes	Data input. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.
`datab[]`	Yes	Data input. The MSB is the sign bit, the next MSBs are the exponent, and the LSBs are the mantissa. This input port size is the total width of sign bit, exponent bits, and mantissa bits.

Table 15–4 lists the output ports of the ALTFP_COMPARE megafunction.

Table 15–4. ALTFP_COMPARE Megafunction Output Ports

Port Name	Required	Description
aeb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port equals the value of the datab[] port.
agb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port is greater than the value of the datab[] port.
ageb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port is greater than or equal to the value of the datab[] port.
alb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port is less than the value of the datab[] port.
aleb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port is less than or equal to the value of the datab[] port.
aneb	Yes	Output port for the comparator. Asserted if the value of the dataa[] port is not equal to the value of the datab[] port.
unordered	Yes	Output port for the comparator. Asserted when either the dataa[] port and the datab[] port is set to NaN, or if both the dataa[] port and the datab[] port are set to NaN.

Table 15–5 lists the parameters of the ALTFP_COMPARE megafunction.

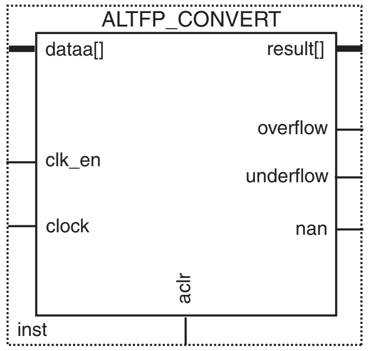
Table 15–5. ALTFP_COMPARE Megafunction Parameters

Port Name	Type	Required	Description
WIDTH_EXP	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is 8. The bias of the exponent is always set to $2^{(WIDTH\_EXP - 1)} - 1$, that is, 127 for the single-precision format and 1023 for the double-precision format. The value of WIDTH_EXP must be 8 for the single-precision format, 11 for the double-precision format, and a minimum of 11 for the single-extended precision format. The value of WIDTH_EXP must be less than the value of WIDTH_MAN, and the sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
WIDTH_MAN	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is 23. When WIDTH_EXP 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 value of WIDTH_MAN must be greater than the value of WIDTH_EXP, and the sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
PIPELINE	Integer	Yes	Specifies the latency in clock cycles used in the ALTFP_COMPARE megafunction. The pipeline values are 1, 2, and 3 latency in clock cycles.

16. ALTFP_CONVERT

Figure 16–1 shows the ports for the ALTFP_CONVERT megafunction.

Figure 16–1. ALTFP_CONVERT Ports



Features

The ALTFP_CONVERT megafunction offers the following features:

- Conversion functions for the following formats:
  - Integer-to-Float.
  - Float-to-Integer.
  - Float-to-Float.
  - Fixed-to-Float.
  - Float-to-Fixed.
- Support for signed and unsigned integers.
- Optional exception handling output ports such as `overflow`, `underflow`, and `nan`.

  Table 16–1 lists the conversion operations supported by each exception port.

Table 16–1. Supported Operations and Exception Ports

Operation	Supported Exception Ports
Integer-to-Float	Not supported
Float-to-Integer	`overflow`, `underflow`, and `nan`
Float-to-Float	`overflow`, `underflow`, and `nan`
Fixed-to-Float	Not supported
Float-to-Fixed	`overflow`, `underflow`, and `nan`

Conversion Operations

The following sections describes the features of each conversion operation.

Integer-to-Float Conversion

- Converts integers to the IEEE-754 standard floating-point representation.
- Supports conversions of signed integers to floating-point numbers in single, double, and single-extended precision formats.

Float-to-Integer Conversion

- Converts IEEE-754 standard floating-point representations to the integer-bit format.
- Supports conversions of signed integers to floating-point numbers in single, double, and single-extended precision formats.

Float-to-Float Conversion

- Converts between IEEE-754 standard floating-point representations.
- Supports conversions of signed integers to floating-point numbers in single, double, and single-extended precision formats.
- This operation offers the following modes:
  - Single-precision format to single-extended precision format or double-precision format.
  - Double-precision format to single-precision format or single-extended precision format.
  - Single-extended precision format to single-precision or double-precision format.

Fixed-to-Float Conversion

- Converts fixed-point format data to the IEEE-754 standard floating-point representation.
- Supports conversions of fixed-point format data to floating-point numbers in single, double, and single-extended precision formats.

Float-to-Fixed Conversion

- Converts IEEE-754 standard floating-point representations to the fixed-point format.
- Supports conversion of floating-point numbers in single, double, and single-extended precision formats.

Output Latency

The output latency options for the all the conversion operations in the ALTFP_CONVERT megafunction are fixed, except for the Float-to-Float operation.

Table 16–2 lists the output latency options required for each operation.

Table 16–2. Latency Options for Each Operation

Operation		Latency (in clock cycles)
Integer-to-Float		6
Float-to-Integer		6
Float-to-Float	Conversion from:	
	Single-precision format	2
	Double-precision format	3
	Single-extended precision format	3
Fixed-to-Float		6
Float-to-Fixed		6

Resource Utilization and Performance

Table 16–3 lists the resource utilization and performance information for the ALTFP_CONVERT megafunction. The information was derived using the Quartus II software version 10.0.

Table 16–3. ALTFP_CONVERT Resource Utilization and Performance for the Stratix III Device Family (Part 1 of 2)

Operation	Format	Pipeline	Logic usage			f_{MAX} (MHz)
			Adaptive Look-Up Tables (ALUTs)	Dedicated Logic Registers (DLRs)	Adaptive Logic Modules (ALMs)	
Integer to-Float	32-bit integer to single-precision	6	182	238	157	515
	32-bit integer to double-precision	6	150	139	123	510
	64-bit integer to single-precision	6	385	371	296	336
	64-bit integer to single-precision	6	393	461	344	336
Float-to-Integer	Single-precision to 32-bit integer	6	256	255	176	455
	Single-precision to 64-bit integer	6	417	361	257	311
	Double-precision to 32-bit integer	6	406	387	273	409
	Double-precision to 64-bit integer	6	535	480	362	309

Table 16–3. ALTFP_CONVERT Resource Utilization and Performance for the Stratix III Device Family (Part 2 of 2)

Float-to-Float	Single-precision to double-precision	2	44	73	40	868
	Double-precision to single-precision	3	103	140	89	520
Fixed-to-Float	16.16 fixed-point to double-precision	6	182	238	155	519
	16.16 fixed-point to double-precision	6	150	139	122	513
	32.32 fixed-point to single-precision	6	384	371	296	334
	32.32 fixed-point to single-precision	6	393	461	336	333
Float-to-Fixed	Single-precision to 16.16 fixed-point	6	319	261	210	438
	Single-precision to 32.32 fixed-point	6	469	367	288	315
	Double-precision to 16.16 fixed-point	6	579	393	402	365
	Double-precision to 32.32 fixed-point	6	695	486	474	306

Design Example: Convert Double-Precision Floating-Point Format Numbers to 64-bit Integers

This design example uses the ALTFP_CONVERT megafunction to convert double-precision floating-point format numbers to 64-bit integers. This design example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_CONVERT megafunction:

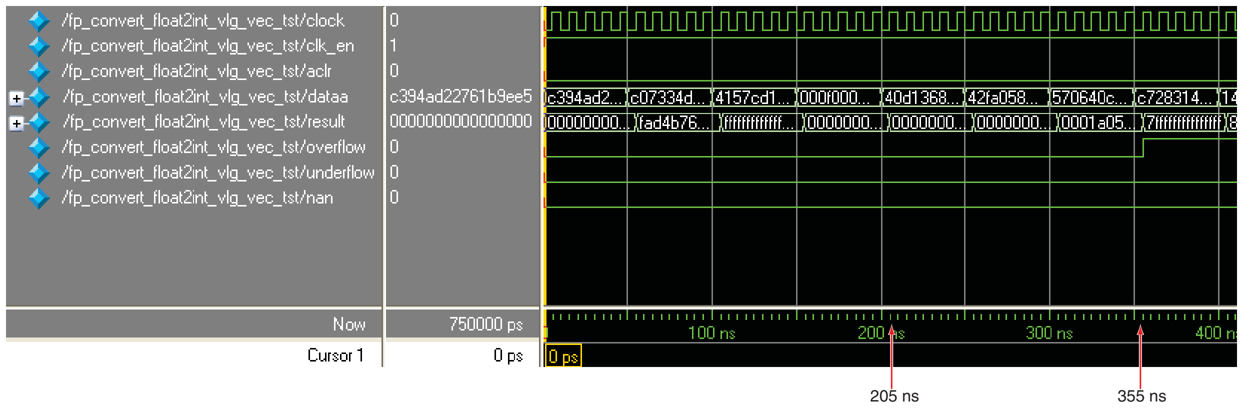
- **altfp_convert_DesignExample.zip** (Quartus II design files)
- **altfp_convert_float2int_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 16–2 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 16–2. ALTFP_CONVERT Simulation Waveform


/fp_convert_float2int_vlg_vec_tst/clock
/fp_convert_float2int_vlg_vec_tst/clk_en
/fp_convert_float2int_vlg_vec_tst/aclr
/fp_convert_float2int_vlg_vec_tst/dataa
/fp_convert_float2int_vlg_vec_tst/result
/fp_convert_float2int_vlg_vec_tst/overflow
/fp_convert_float2int_vlg_vec_tst/underflow
/fp_convert_float2int_vlg_vec_tst/nan
0
1
0
c394ad22761b9ee5
0000000000000000
0
0
0
c394ad2... c07334d... 4157cd1... 000f000... 40d1368... 42fa058... 570640c... c728314... 14
00000000... fad4b76... ffffffffffff... 0000000... 0000000... 0000000... 0001a05... 7fffffffffffff 8
Now
750000 ps
Cursor 1
0 ps
100 ns
200 ns
300 ns
400 ns
0 ps
205 ns
355 ns

This design example implements a float-to-integer converter for converting double-precision floating-point format numbers to 64-bit integers. In this operation, the optional exception ports of `overflow`, `underflow`, and `nan` are available apart from the `result[]` port.

The latency for the float-to-integer operation is six clock cycles. Therefore, each conversion generates the output result six clock cycles after receiving the input value.

Table 16–4 lists the inputs and corresponding outputs obtained from the simulation in Figure 16–2 on page 16–5.

Table 16–4. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns, start-up	`dataa[]` value: C394 AD22 761B 9EE5h Output value: The `result[]` port displays 0 regardless of what the input value is. This value seen on the output port before the 6th clock cycle is merely due to the behavior of the system during start-up and should be disregarded.
55 ns	Output value: FAD4 B762 7918 46C0h
150 ns	`dataa[]` value: 000F 0000 5555 1111h This value is a denormal number.
205 ns	Denormal inputs are not supported and are forced to zero before conversion takes place.
300 ns	`dataa[]` value: 5706 40CF OEC6 1176h
355 ns	Output value: 7FFF FFFF FFFF FFFFh Exception handling ports: `overflow` asserts. The `overflow` flag is triggered because the width of the resulting integer is more than the maximum width allowed, and the value seen on the `result[]` port is the standard value used to represent a positive overflow number.
350 ns	`dataa[]` value: C728 3147 8444 1F75h
405 ns	Output value: 8000 0000 0000 0000h Exception handling ports: `overflow` remains asserted. This is a standard value to represent a negative overflow number.
400 ns	`dataa[]` value: 145A 257C 895A B309h
455 ns	Output value: 0000 0000h Exception handling ports: `underflow` asserts. The input value triggers the `underflow` port because the exponent of the input value is less than the exponent bias of 1023.
500 ns	`dataa[]` value: FFFF 0000 DDDD 5555h This value is a NaN.
555 ns	Output value: 0000 0000h Exception handling ports: `nan` asserts.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_CONVERT megafunction. The ports and parameters are available to customize the ALTFP_CONVERT megafunction according to your application.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_convert
        generic (
                intended_device_family  :        string := "unused";
                operation       :       string := "INT2FLOAT";
                rounding        :       string := "TO_NEAREST";
                width_data      :       natural := 32;
                width_exp_input :       natural := 8;
                width_exp_output        :       natural := 8;
                width_int       :       natural := 32;
                width_man_input :       natural := 23;
                width_man_output        :       natural := 23;
                width_result    :       natural := 32;
                lpm_hint        :       string := "UNUSED";
                lpm_type        :       string := "altfp_convert"
        );
        port(
                aclr    :       in std_logic := '0';
                clk_en  :       in std_logic := '1';
                clock   :       in std_logic;
                dataa   :       in std_logic_vector(width_data-1 downto 0);
                nan     :       out std_logic;
                overflow        :       out std_logic;
                result  :       out std_logic_vector(width_result-1 downto 0);
                underflow       :       out std_logic
        );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;

USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 16–5 lists the input ports of the ALTFP_CONVERT megafunction.

Table 16–5. ALTFP_CONVERT Megafunction Input Ports

Port Name	Required	Description
`clock`	Yes	The clock input to the ALTFP_CONVERT megafunction.
`clk_en`	No	Clock enable that allows conversions to take place when asserted high. When asserted low, no operation occurs and the outputs are unchanged.
`aclr`	No	Asynchronous clear. The source is asynchronously reset when the `aclr` signal is asserted high.
`dataa[]`	Yes	Data input. The size of this input port depends on the `WIDTH_DATA` parameter value. If the operation mode value is `INT2FLOAT` or `FIXED2FLOAT`, the data on the input bus is an integer. If the operation mode value is `FLOAT2INT` or `FLOAT2FIXED`, the input bus is the IEEE floating-point representation. In the single-precision format, the input bus width value is `32`. In the double-precision format, the input bus width value is `64`. In the single-extended precision format, the input bus range is from `43` to `64`. If the operation mode value is `FLOAT2FLOAT`, the input bus value is the IEEE floating-point representation. In the single-precision format, the input bus width value is `32`. In the double-precision format, the input bus width value is `64`. In the single-extended precision format, the input bus range is from `43` to `64`.

Table 16–6 describes the output ports of the ALTFP_CONVERT megafunction.

Table 16–6. ALTFP_CONVERT Megafunction Output Ports (Part 1 of 2)

Port Name	Required	Description
`result[]`	Yes	Output for the floating-point converter. The size of this output port depends on the `WIDTH_RESULT` parameter value. If the operation mode value is `FLOAT2INT` or `FLOAT2FIXED`, the output bus is an IEEE floating-point representation. If the operation mode is `FLOAT2INT`, the output bus is an integer representation. If the selected precision is the single-precision format, the output bus width value is `32`. If the selected precision is the double-precision format, the output bus width value is `64`. If the selected precision is the single-extended precision format, the input bus range is from `43` to `64`. If the operation mode value is `FLOAT2FLOAT`, the output bus is an IEEE floating-point representation. If the selected precision is the single-precision format, the output bus is in the 64-bit double-precision format. If the selected precision is the double-precision format, the output bus is in the 32-bit single-precision format. If the selected precision is the single-extended precision format, the output bus ranges from `43` to `64`.
`overflow`	No	Optional overflow exception output. This port is available only when the operation mode values are `FLOAT2FIXED`, `FLOAT2INT`, or `FLOAT2FLOAT`. Asserted when the result of the conversion (after rounding), exceeds the maximum width of the `result[]` port, or when the `dataa[]` input is infinity.

Table 16–6. ALTFP_CONVERT Megafunction Output Ports (Part 2 of 2)

Port Name	Required	Description
`underflow`	No	Optional underflow exception output. This port is available only when the operation mode values are `FLOAT2FIXED`, `FLOAT2INT`, or `FLOAT2FLOAT`. Asserted when the result of the conversion, after rounding, is fractional. In `FLOAT2INT` operations, this port is asserted when the exponent value of the floating-point input is smaller than the exponent bias. In `FLOAT2FLOAT` operations, this port is asserted when the floating-point input has a value smaller than the lowest exponent limit of the target floating-point format.
`nan`	No	Optional NaN exception output. This port is available only when the operation mode values are `FLOAT2INT`, `FLOAT2FLOAT`, or `FLOAT2FIXED`. Asserted when the input port is a NaN representation. If the operation mode value is `FLOAT2INT` or `FLOAT2FIXED`, the `result[]` port is set to zero. If the operation mode value is `FLOAT2FLOAT`, the `result[]` port is set to a NaN representation.

Table 16–7 lists the parameters of the ALTFP_CONVERT megafunction.

Table 16–7. ALTFP_CONVERT Megafunction Parameters (Part 1 of 3)

Port Name	Type	Required	Description
`WIDTH_EXP_INPUT`	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is `8`. The bias of the exponent is always set to $2^{(\texttt{WIDTH\_EXP} - 1)} - 1$, that is, `127` for the single-precision format and `1023` for the double-precision format. The value of `WIDTH_EXP_INPUT` must be `8` for the single-precision format, `11` for the double-precision format, and a minimum of `11` for the single-extended precision format. The value of `WIDTH_EXP_INPUT` must be less than the value of `WIDTH_MAN_INPUT`, and the sum of `WIDTH_EXP_INPUT` and `WIDTH_MAN_INPUT` must be less than `64`. These settings apply only to the `FLOAT2FIXED`, `FLOAT2INT`, and `FLOAT2FLOAT` operation modes.
`WIDTH_MAN_INPUT`	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is `23`. When `WIDTH_EXP_INPUT` is `8` and the floating-point format is single-precision, the `WIDTH_MAN_INPUT` value must be `23`. Otherwise, the value of `WIDTH_MAN_INPUT` must be a minimum of `31`. The value of `WIDTH_MAN_INPUT` must be greater than the value of `WIDTH_EXP_INPUT`, and the sum of `WIDTH_EXP_INPUT` and `WIDTH_MAN_INPUT` must be less than `64`. These settings apply only to the `FLOAT2FIXED`, `FLOAT2INT`, and `FLOAT2FLOAT` operation modes.
`WIDTH_INT`	Integer	Yes	Specifies the integer width. If the operation is `FIXED2FLOAT` or `INT2FLOAT`, this parameter defines the integer width on the input side. If the operation is `FLOAT2INT` or `FLOAT2FIXED`, this parameter defines the result width on the output side. The available settings are `32` bits, `64` bits or *n* bits. For *n* bits settings, the range is from `4` bits to `64` bits.

Table 16–7. ALTFP_CONVERT Megafunction Parameters (Part 2 of 3)

Port Name	Type	Required	Description
WIDTH_DATA	Integer	Yes	Specifies the input data width. If the operation is INT2FLOAT, the WIDTH_DATA is also WIDTH_INT. If the operation is FIXED2FLOAT, the data width value is WIDTH_INT + fractional width. If the operation is FLOAT2FIXED, FLOAT2INT or FLOAT2FLOAT, the data width value is WIDTH_EXP_INPUT + WIDTH_MAN_INPUT + 1. The available settings are 32 bits, 64 bits or *n* bits. For *n* bits settings, the range is from 4 bits to 64 bits.
WIDTH_EXP_OUTPUT	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the default is 8. The bias of the exponent is always set to $2^{(\mathtt{WIDTH\_EXP} - 1)} - 1$, that is, 127 for the single-precision format and 1023 for the double-precision format. The value of WIDTH_EXP_OUTPUT must be 8 for the single-precision format, 11 for the double-precision format, and a minimum of 11 for the single-extended precision format. The value of WIDTH_EXP_OUTPUT must be less than the value of WIDTH_MAN_OUTPUT, and the sum of WIDTH_EXP_OUTPUT and WIDTH_MAN_OUTPUT must be less than 64. These settings apply only to the FLOAT2FIXED, FLOAT2INT, and FLOAT2FLOAT operation modes.
WIDTH_MAN_OUTPUT	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is 23. When WIDTH_EXP_OUTPUT is 8 and the floating point format is single-precision, the WIDTH_MAN_OUTPUT value must be 23. Otherwise, the value of WIDTH_MAN_OUTPUT must be a minimum of 31. The value of WIDTH_MAN_OUTPUT must be greater than the value of WIDTH_EXP_OUTPUT, and the sum of WIDTH_EXP_OUTPUT and WIDTH_MAN_OUTPUT must be less than 64. These settings apply only to the FLOAT2FIXED, FLOAT2INT, and FLOAT2FLOAT operation modes.
WIDTH_RESULT	Integer	Yes	Specifies the width of the output result. In an INT2FLOAT, FLOAT2FLOAT, or FIXED2FLOAT operation, the result width is WIDTH_EXP_OUTPUT + WIDTH_MAN_OUTPUT + 1. In a FLOAT2INT operation, the result width is the value of the WIDTH_INT parameter. In a FLOAT2FIXED operation, this parameter is the result width. The available settings are 32 bits, 64 bits or *n* bits. For *n* bits settings, the range is from 4 bits to 64 bits.

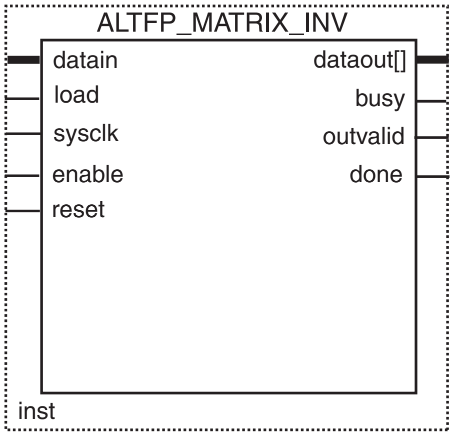
Table 16–7. ALTFP_CONVERT Megafunction Parameters (Part 3 of 3)

Port Name	Type	Required	Description
`ROUNDING`	Integer	Yes	Specifies the rounding mode. The default value is `TO_NEAREST`. Other modes are not supported.
`OPERATION`	Integer	Yes	Specifies the operating mode. Values are `INT2FLOAT`, `FLOAT2INT`, `FLOAT2FLOAT`, `FLOAT2FIXED`, and `FIXED2FLOAT`. If this parameter is not specified, the default value is `INT2FLOAT`. When set to `INT2FLOAT`, the conversion of an integer input to an IEEE floating-point representation output takes place. When set to `FLOAT2INT`, the conversion of an IEEE floating-point representation input to an integer output takes place. When set to `FLOAT2FLOAT`, the conversion between IEEE floating-point representations input and output takes place. When set to `FIXED2FLOAT`, the conversion of a fixed point input to an IEEE floating-point representation output takes place. When set to `FLOAT2FIXED`, the IEEE floating-point input conversion to fixed point representation output takes place.

17. ALTFP_MATRIX_INV

Figure 17–1 shows the ports for the ALTFP_MATRIX_INV megafunction.

Figure 17–1. ALTFP_MATRIX_INV Ports



Features

The ALTFP_MATRIX_INV megafunction offers the following features:

- Inversion of a matrix.
- Support for floating-point format in single precision.
- Support for VHDL and Verilog HDL languages.
- Support for matrix sizes up to are 4 × 4, 6 × 6, 8 × 8, 16 ×16, 32 × 32, and 64 × 64.
- Use of control signal, `load`.
- Use of handshaking signals: `busy`, `outvalid`, and `done`.

Output Latency

The ALTFP_MATRIX_INV megafunction does not have a fixed output latency. Instead, it uses handshaking signals to interface with external circuitry.

Resource Utilization and Performance

Table 17–1 and Table 17–2 list the resource utilization and performance information for the ALTFP_MATRIX_INV megafunction. The information was derived using the Quartus II software version 10.0

Table 17–1. ALTFP_MATRIX_INV Resource Utilization and Performance for the Stratix III Device Family

Precision	Matrix Size	Blocks	Logic usage					Latency	Throughput (kb/s)	Giga Floating-Point Operations per Second (GFLOPS)	f_{MAX} (MHz)
			Adaptive Logic Modules (ALMs)	DSP Usage (18 x 18 DSPs)	M9K	M144K	Memory (Bits)				
Single	4× 4	2	22151	222	116	—	18227	Pending	Pending	Pending	219
	6 × 6	2	62872	574	1	—	Pending	Pending	Pending	Pending	162
	8 × 8	2	6,160	63	39	—	53,736	2,501	3,987	14.33	312
	16 × 16	4	9,923	95	64	—	138,051	11,057	855	27.76	295
	32 × 32	8	17,980	159	150	—	699,164	52,625	165	51.58	271
	64 × 64	16	34,973	287	151	33	4,770,369	281,505	25	83.34	218

Table 17–2. ALTFP_MATRIX_INV Resource Utilization and Performance for the Stratix IV Device Family

Precision	Matrix Size	Blocks	Logic usage					Latency	Throughput (kb/s)	Giga Floating-Point Operations per Second (GFLOPS)	f_{MAX} (MHz)
			Adaptive Logic Modules (ALMs)	DSP Usage (18 x 18 DSPs)	M9K	M144K	Memory (Bits)				
Single	4× 4	2	21159	222	139	—	19919	Pending	Pending	Pending	221
	6 × 6	2	59827	574	90	—	15759	Pending	Pending	Pending	170
	8 × 8	2	5,538	63	49	—	53,736	2,501	3,987	15.26	332
	16 × 16	4	8,865	95	80	—	138,051	11,057	855	30.93	329
	32 × 32	8	15,655	159	193	—	699,164	52,625	165	55.12	290
	64 × 64	16	29,940	287	386	22	4,770,369	281,505	25	83.16	218

Functional Description

A matrix inversion function is composed of the following components:

- Cholesky decomposition function.

  The Cholesky decomposition function generates a lower triangular matrix.
- Triangular matrix inversion function.

  The triangular matrix inversion process then generates the inverse of the lower triangular using backward substitution.
- Matrix multiplication function.

  The matrix multiplier multiplies the transpose of the inverse triangular matrix with the inverse triangular matrix.

In linear algebra, the Cholesky decomposition states that every positive definite matrix A is decomposed as $A = L \times L^T$

where, L is a lower triangular matrix, and L^T denotes the transpose of L.

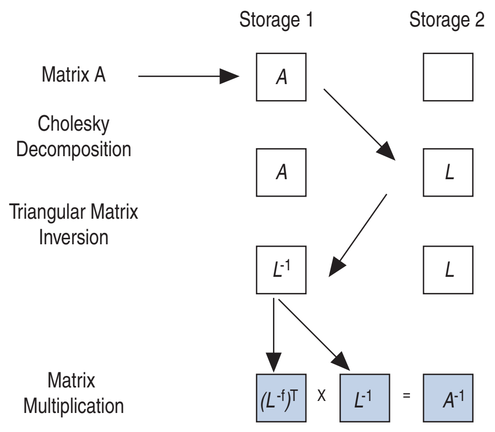
The property of invertible matrices states that $(X \times Y)^{-1} = X^{-1} \times Y^{-1}$ and the property of transpose states that $(X^T)^{-1} = (X^{-1})^T$. Combining these two properties, the following equation represents a derivation of a matrix inversion using the Cholesky decomposition method:

$$\begin{aligned} A^{-1} &= (L \times L^T)^{-1} \\ &= (L^T)^{-1} \times L^{-1} \\ &= (L^{-1})^T \times L^{-1} \end{aligned}$$

where a Cholesky decomposition function is needed to obtain L, a triangular matrix inversion is needed to obtain L^{-1}, and a matrix multiplication is needed for $(L^{-1})^T \times L^{-1}$.

Figure 17–2 shows the flow diagram of the matrix inversion.

Figure 17–2. Flow Diagram



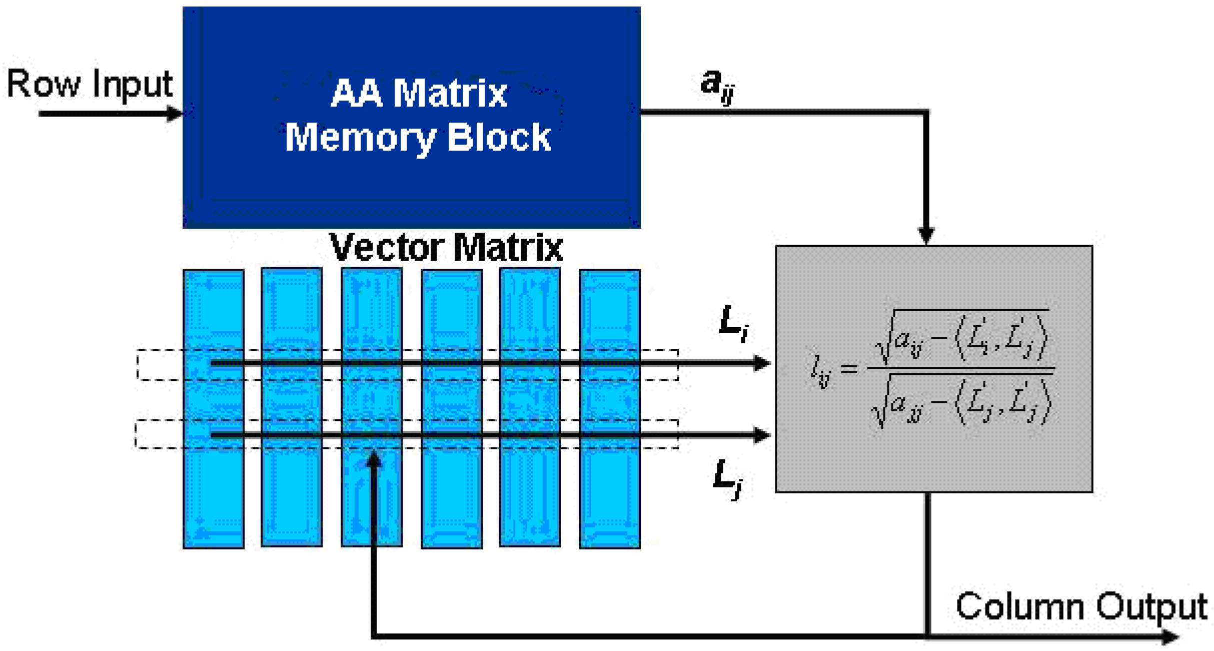
Cholesky Decomposition Function

The functions consists of two memory and two processing blocks. One of the memory blocks is the input matrix memory block and is loaded with the input matrix in a row order, one element at a time. However, during processing, this block is read in a column order, one element at a time when required.

The other memory block is the processing matrix block which consists of multiple column memories to enable an entire row to be read at once. During the loading of the input memory, the FPC datapath preprocesses the input elements to generate the first column of the resulting triangular matrix. The top element of the first column, l00, is the square root of the input matrix value `a00`. The rest of the first column, `li0` is the input value `ai0` divided by l00. This preprocessing step introduces latency into the load, during which the `INIT_BUSY` signal is asserted. The `CALCULATE` signal initiates and starts processing after the `INIT_BUSY` signal is deasserted.

Figure 17–3 shows the top-level architecture of the Cholesky decomposition function, where the monolithic input memory and the column-wise processing memory, also known as the vector matrix, are shown. The gray block is the FPC datapath section.

Figure 17–3. Cholesky Decomposition Function Top-level Diagram


Row Input
AA Matrix Memory Block
a_{ij}
Vector Matrix
L_i
L_j
$l_{ij} = \frac{\sqrt{a_{ij} - \langle L_i, L_j \rangle}}{\sqrt{a_{jj} - \langle L_j, L_j \rangle}}$
Column Output

Although the Cholesky decomposition algorithm only operates on the lower triangular matrix, the core requires the entire matrix to be loaded, during which the processing or vector memory is initialized.

The FPC datapath is split into two sections. The first section, also known as the vector section, takes the inner product of two vectors and subtracts it from the input matrix element, a_{ij}. The second section, also known as the root section, calculates square roots and performs division by the square root. The first element is loaded into both inputs of the root section and the outcome is its own square root. The first element continues to stay latched in the left input field of the root section while all the other elements of the first column are loaded into the right input field. The resulting output is the value of the respective column element divided by the value of the first element of the Cholesky decomposition matrix.

During processing, two rows from the processing matrix are loaded. For the first element in each new column, both rows have the same index; hence contain the same values. The first row is latched into the input register of the vector section. For the rest of the column, the row index is increased, and a new a_{ij} element and triangular matrix vector, L_j is loaded. The first result out of the vector section is latched onto the left register of the root section. All results from the column, including the first result, are loaded into the right register of the root section. The root section generates the square root of the first vector result, while for the other results coming from the vector section, the number is divided by the square root of the first result.

All calculated values are written to another memory block for further processing. The first column values are output singly during preprocessing, while the values of other columns are burst out during processing.

There are only minor differences between the architectures for real and complex matrices. For the complex matrix, both the input and processing memory blocks contain complex values. Similarly, all values going into the vector section are complex numbers. The complex conjugate of the latched register is obtained by simply inverting the sign bit. As for the root section, the structure is simplified by the nature of the positive definite matrix. The diagonal value, which is the first value at the top of each column in the decomposition, is always a real number so that the result from the inverse square root calculation is always a real number. The complex multiplier in the root section is therefore a real scalar, so only two real multipliers are required.

Triangular Matrix Inversion Algorithm

The triangular matrix, L, obtained from the Cholesky decomposition function is computed using the triangular matrix inversion algorithm to get its inversion. The following MatLab pseudo code shows how the inversion is carried out:

```
for j = n:-1:1,
  X(j,j) = 1/L(j, j);
  for k = j+1:n
    for i = j+1:n
      X(k, j) = X(k, j) + X(k, i)*L(i, j);
    end;
  end;
  for k = j+1:n
    X(k, j) = -X(j, j)*X(k, j);
  end;
```

The pseudo code is converted into an RTL file. The result, L-1 is stored in the input matrix storage in the Cholesky decomposition function.

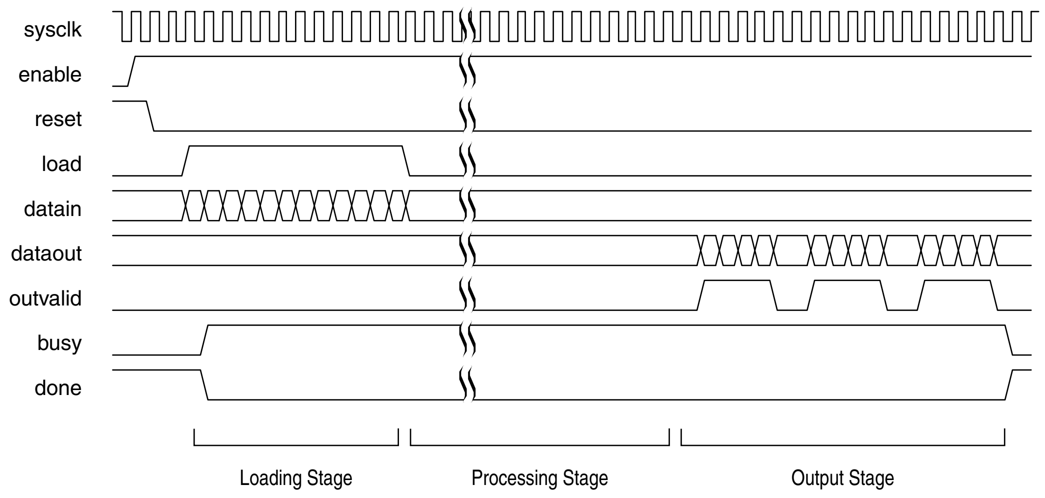
Matrix Multiplication

The final stage of the matrix inversion process involves multiplying the transpose of the inverse triangular matrix with the inverse triangular matrix using the Altera Floating-Point Matrix Multiplier. The original version of the matrix multiplier is modified for this purpose. As there are memory blocks already available for the storage of the input matrices in the Cholesky decomposition function, the memory blocks in the matrix multiplier are redundant and can be removed. Data is instead fed directly from the results stored at the end stage of the triangular matrix inversion algorithm.

Matrix Inversion Operation

Figure 17–4 shows the timing diagram for a matrix inversion operation.

Figure 17–4. Matrix Inversion Timing Diagram



The following sequence describes the matrix inversion operation:

1. The operation begins when the `enable` signal is asserted and the `reset` signal is deasserted.
2. The `load` signal is asserted to load data from the `loaddata[]` port for the input matrix. As long as the `load` signal is high, data is loaded continuously for the input matrix.
3. The `busy` signal is asserted and the done signal is deasserted for a few clock cycles after the `datain[]` signal is asserted.
4. The `outvalid` signal is asserted multiple times to signify the availability of valid data on the `dataout[]` port. The number of times this signal is asserted equals the number of rows found in the output matrix.

5. The `busy` and `done` signals are asserted when the last row of the output matrix has been burst out. This assertion signifies the end of the matrix inversion operation on the first set of data.

Design Example: Matrix Inverse of Single-Precision Format Numbers

This design example uses the ALTFP_MATRIX_INV megafunction to show the matrix inversion operation. The input matrix applied is an 8 × 8 matrix with a block size of 2. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

Design Files

The following files are related to the ALTFP_MATRIX_INV megafunction:

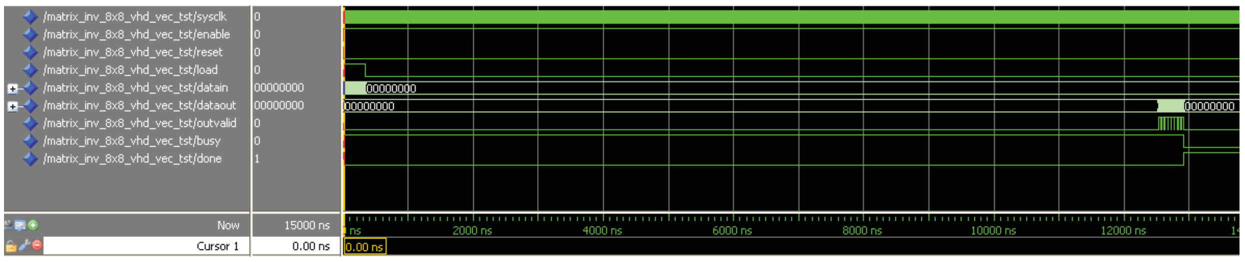
- **altfp_matrix_inv_DesignExample.zip** (Quartus II design files)
- **altfp_matrix_inv_ex_msim.zip** (ModelSim-Altera files)

Understanding the Simulation Results

Figure 17–5 shows the expected simulation results in the ModelSim-Altera software.

The simulation waveform in this design example is not shown in its entirety. Run the design example files in the ModelSim-Altera software to see the complete simulation waveforms.

Figure 17–5. ModelSim Simulation Waveform (Input Data)


/matrix_inv_8x8_vhd_vec_tst/sysclk 0
/matrix_inv_8x8_vhd_vec_tst/enable 0
/matrix_inv_8x8_vhd_vec_tst/reset 0
/matrix_inv_8x8_vhd_vec_tst/load 0
/matrix_inv_8x8_vhd_vec_tst/datain 00000000
/matrix_inv_8x8_vhd_vec_tst/dataout 00000000
/matrix_inv_8x8_vhd_vec_tst/outvalid 0
/matrix_inv_8x8_vhd_vec_tst/busy 0
/matrix_inv_8x8_vhd_vec_tst/done 1
00000000
00000000
00000000
Now 15000 ns
Cursor 1 0.00 ns
ns
2000 ns
4000 ns
6000 ns
8000 ns
10000 ns
12000 ns
0.00 ns

This design example implements a floating-point matrix inversion to calculate the inverse value of matrices in single-precision formats. The optional input ports (`enable` and `reset`) are enabled.

Table 17–3 on page 17–8 lists the inputs and corresponding outputs obtained from the simulation in Figure 17–5. The number of clock cycles obtained for each stage is based on the particular matrix size and parameter settings used in this design example.

Table 17–3. Summary of Input Values and Corresponding Outputs

Time	Event
0 ns – 10 ns	Start sequence: ■ The `reset` signal deasserts. ■ The `enable` signal asserts.
19.86 ns – 340 ns	Matrix input data load: ■ The `load` signal asserts and remains high for 80 clock cycles. ■ As long as the `load` signal is high, data for the input matrix is loaded row by row. ■ Input data is burst in regularly, one at every clock cycle. ■ The `load` signal deasserts at 340 ns. The deassertion of the `load` signal signifies the completion of the data load operation for the matrix.
27.5 ns	Processing stage: ■ The `busy` signal asserts while the `done` signal deasserts. ■ The assertion of the `busy` signal and the deassertion of the `done` signal indicate that the matrix inversion core is processing the input data. ■ There are about 2500 clock cycles between the beginning of the processing stage and the first available output value.
12527.5 – 12922.5 ns	Output stage: ■ The `outvalid` signal asserts in intervals of 8 clock cycles. These series of assertions signify the availability of valid data for the output matrix on the `outdata[]` port. ■ The output is an 8 x 8 matrix. Data is burst out regularly, row by row. ■ At 12922.5 ns, the `busy` signal is asserted and the `done` signal is deasserted. ■ The assertion of the `busy` signal and the deassertion of the `done` signal indicate that the final output is written and a new matrix can be processed.

Sample Matrix Data

This section shows the random test data assigned to the input matrices and the results obtained from the matrix inversion operation.

The following two sets of results are computed:

- PC-based results—these are results obtained from running the simulation in Matlab.
- FPGA-based results—these are results obtained from running the simulation in ModelSim.

Table 17–4 lists the input and output data values presented in IEEE-754 Floating-point format.

Table 17–4. Input and Output Data

Matrix	Data
Input Matrix	40c89c6c 40b16187 40e21dfb 40847306 40c00d1d 40bbf0c4 40be4fc1 40953a30
	40b16187 41244acb 410e61b9 40defe3a 40f8e982 40eff916 410e0ff4 41121d78
	40e21dfb 410e61b9 41217d87 40d7f5f4 40fd78fa 410618c0 41060327 40ff4517
	40847306 40defe3a 40d7f5f4 40b10427 40b6be88 40bbff4a 40d12685 40ca69f9
	40c00d1d 40f8e982 40fd78fa 40b6be88 41146829 40ee188a 40fa2d80 40cf065c
	40bbf0c4 40eff916 410618c0 40bbff4a 40ee188a 40ecbddf 40e3aa3a 40d60773
	40be4fc1 410e0ff4 41060327 40d12685 40fa2d80 40e3aa3a 4111ed09 40ecd83c
	40953a30 41121d78 40ff4517 40ca69f9 40cf065c 40d60773 40ecd83c 410847da
PC-based Output Matrix	42148e03 42f5794f 421b33f4 430e0587 41ff0d66 c2f579a3 c2df1c28 c2f945bc
	42f5794f 43d60be5 430944db 43f2dd63 42da2dd0 c3d1dd59 c3bff960 c3d98c47
	421b33f4 430944db 424b067c 43204d17 421907da c3107054 c2fc035b c30d24b3
	430e0587 43f2dd63 43204d17 440cc66b 43002bbb c3f4e779 c3dcd667 c3f7e3f3
	41ff0d66 42da2dd0 421907da 43002bbb 41f5048b c2e44480 c2c91e6d c2df60c9
	c2f579a3 c3d1dd59 c3107054 c3f4e779 c2e44480 43d89b61 43c003b9 43d685d3
	c2df1c28 c3bff960 c2fc035b c3dcd667 c2c91e6d 43c003b9 43ae19b0 43c37f99
	c2f945bc c3d98c47 c30d24b3 c3f7e3f3 c2df60c9 43d685d3 43c37f99 43ddb1bc
FPGA-based Output Matrix	42148d06 42f5773e 421b32c4 430e0484 41ff0bb7 c2f577f4 c2df1a71 c2f943b1
	42f5773e 43d609cf 430943a0 43f2db4a 42da2c09 c3d1db95 c3bff79e c3d98a34
	421b32c4 430943a0 424b0515 43204be2 421906da c3106f53 c2fc014f c30d237c
	430e0484 43f2db4a 43204be2 440cc563 43002adf c3f4e5c0 c3dcd4a7 c3f7e1df
	41ff0bb7 42da2c09 421906da 43002adf 41f50322 c2e44314 c2c91cf5 c2df5f08
	c2f577f4 c3d1db95 c3106f53 c3f4e5c0 c2e44314 43d899f3 43c00242 43d68414
	c2df1a71 c3bff79e c2fc014f c3dcd4a7 c2c91cf5 43c00242 43ae1837 43c37dda
	c2f943b1 c3d98a34 c30d237c c3f7e1df c2df5f08 43d68414 43c37dda 43ddafad

The difference between each result element of the PC-based and FPGA-based output matrices are as shown:

Result differences (in decimal)

253	529	304	259	431	431	439	523
529	534	315	537	455	452	450	531
304	315	359	309	256	257	524	311
259	537	309	264	220	441	448	532
431	455	256	220	361	364	376	449
431	452	257	441	364	366	375	447
439	450	524	448	376	375	377	447
523	531	311	532	449	447	447	527

The difference between the two output matrices are due to the following reasons:

- Method of processing—Matlab uses sequential processing while Modelsim uses parallel processing.
- Method of conversion—Matlab first computes in double-precision format, and then only converts the result into single-precision format. During this conversion, some units in the last place (ulp) are expected to be lost.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_MATRIX_INV megafunction. The ports and parameters are available to customize the ALTFP_MATRIX_INV megafunction according to your application.

☞ You must manually include the **altfpc.v/vhd** library file to your project.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_matrix_inv
generic (
blocks : natural := 2;
cluster : natural := 8;
intended_device_family : string := "unused";
dimension : natural := 8;
width_exp : natural := 8;
width_man : natural := 23;
lpm_hint : string := "UNUSED";
lpm_type : string := "altfp_matrix_inv"
);
port(
busy : out std_logic;
datain : in std_logic_vector(width_exp+width_man+1-1 downto 0) := (others => '0');
dataout : out std_logic_vector(width_exp+width_man+1-1 downto 0);
done : out std_logic;
enable : in std_logic := '1';
load : in std_logic := '0';
outvalid : out std_logic;
reset : in std_logic := '0';
sysclk : in std_logic
);
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 17–5 lists the input ports of the ALTFP_MATRIX_INV megafunction.

Table 17–5. ALTFP_MATRIX_INV Megafunction Input Ports

Port Name	Required	Description
`sysclk`	Yes	The clock input to the ALTFP_MATRIX_INV megafunction. This is the main system clock. All operations occur on the rising edge.
`enable`	No	Optional port. Allow calculation to take place when asserted. When deasserted, no operation will take place and the outputs are unchanged.
`reset`	No	Optional port. The core resets asynchronously when the `reset` signal is asserted.
`load`	Yes	When asserted, loads the `LOADDATA` bus into the memory.
`loaddata`	Yes	Single-precision 32-bit matrix input value. Matrices load row by row.

Table 17–6 lists the output ports of the ALTFP_MATRIX_INV megafunction.

Table 17–6. ALTFP_MATRIX_INV Megafunction Output Ports

Port Name	Required	Description
`ready`	Yes	When asserted, the core preprocesses the input data. The calculate signal cannot be asserted until the `ready` signal is low.
`outdata`	Yes	Single-precision 32-bit matrix result value. The matrix result value is written out row by row.
`outvalid`	Yes	When asserted, a valid output data is available. An entire row of the result matrix is written out as a burst. There is a gap between row outputs, which will depend on the parameters.
`done`	Yes	When asserted, the last output has been written. A new matrix multiply can be started with `calculate`. `done` will follow `ready` by some fixed amount, depending on the parameters.

Table 17–7 lists the parameters of the ALTFP_MATRIX_INV megafunction.

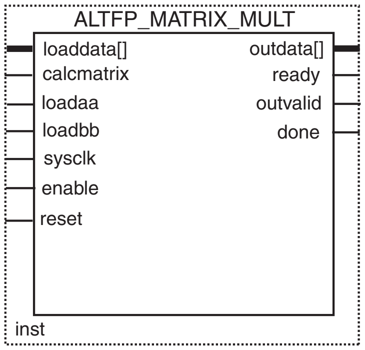
Table 17–7. ALTFP_MATRIX_INV Megafunction Parameters

Port Name	Type	Required	Description
`BLOCKS`	Integer	No	The number of memory blocks for the double-buffered storage of matrix multiplication. The allowable range is from `2` to `16`.
`DIMENSION`	Integer	Yes	The number of rows in the matrix. As the matrix is square, this is also the number of columns in the matrix. The supported dimensions are 4 x 4, 6 x 6, 8 x 8, 16 x 16, 32 x 32, and 64 x 64. The maximum supported input dimension is 64 × 64. This parameter also acts as the VECTORSIZE when calling the ALTFP_MATRIX_MULT megafunction internally.
`WIDTH_EXP`	Integer	Yes	Specifies the precision of the exponent. The bias of the exponent is always set to $2^{(WIDTH\_EXP-1)}-1$ (that is, `127` for single-precision format). `WIDTH_EXP` must be `8` for single-precision format and must be less than `WIDTH_MAN`. The available value for `WIDTH_EXP` is `8`.
`WIDTH_MAN`	Integer	Yes	Specifies the precision of the mantissa. `WIDTH_MAN` must be 23 when `WIDTH_EXP` is 8. Otherwise, `WIDTH_MAN` must be a minimum of 31. `WIDTH_MAN` must be greater than `WIDTH_EXP`. The sum of `WIDTH_EXP` and `WIDTH_MAN` must be less than 64. Current available value for `WIDTH_MAN` is only 23 for single precision.

18. ALTFP_MATRIX_MULT

Figure 18–1 shows the ports for the ALTFP_MATRIX_MULT megafunction.

Figure 18–1. ALTFP_MATRIX_MULT Ports



Features

The ALTFP_MATRIX_INV megafunction offers the following features:

- Multiplication of two matrices.
- Support for floating-point formats in single and double precisions.
- Support for VHDL and Verilog HDL languages.
- Support for complex numbers in single-precision format.
- Use of control signals: `loadaa`, `loadbb`, and `calcmatrix`.
- Use of handshaking signals: `ready`, `outvalid`, and `done`.

Output Latency

The ALTFP_MATRIX_MULT megafunction does not have a fixed output latency. Instead, it uses handshaking signals to interface with external circuitry.

Resource Utilization and Performance

Table 18–1 and Table 18–2 list the resource utilization and performance information for the ALTFP_MATRIX_MULT megafunction. The information was derived using the Quartus II software version 10.0.

Table 18–1. ALTFP_MATRIX_MULT Resource Utilization and Performance for the Stratix III Device Family

Precision	MatrixAA Size	MatrixBB Size	Blocks	Vectorsize	Logic usage					Latency	Throughput (kb/s)	Giga Floating-Point Operations per Second (GFLOPS)	f_{MAX} (MHz)
					Adaptive Logic Modules (ALMs)	DSP Usage (18 x 18 DSPs)	M9K	M144K	Memory (Bits)				
Single	8 × 8	8 × 8	2	8	3,698	32	22	—	14,986	209	63,333	6.20	414
	16 × 16	16 × 16	2	8	3,992	32	20	—	55,562	611	21,573	6.18	412
	32 × 32	32 × 32	4	16	7,173	64	61	—	339,718	2,172	5,972	12.57	405
	64 × 64	64 × 64	8	32	13,803	128	41	16	2,382,318	8,353	1,454	23.91	380
Double	8 × 8	8 × 8	2	8	9,026	112	34	—	29,762	213	90,967	4.54	303
	16 × 16	16 × 16	2	8	9,234	112	39	—	110,756	615	32,658	4.71	314
	32 × 32	32 × 32	4	16	18,142	224	109	—	679,302	2,178	8,791	9.27	299
	64 × 64	64 × 64	8	32	35,839	448	77	32	4,765,120	8,359	2,176	17.91	284
Single (Complex)	8 × 8	8 × 8	2	8	9,996	128	59	—	22,666	220	114,411	12.80	413
	16 × 16	16 × 16	2	8	10,344	128	64	—	79,139	624	39,743	12.52	404
	32 × 32	32 × 32	4	16	20,005	256	146	—	420,519	2,181	10,937	24.99	397
	64 × 64	64 × 64	8	32	39,068	512	216	16	2,674,289	8,362	2,594	45.68	360

Table 18–2. ALTFP_MATRIX_MULT Resource Utilization and Performance for the Stratix IV Device Family

Precision	MatrixAA Size	MatrixBB Size	Blocks	Vectorsize	Logic usage					Latency	Throughput (kb/s)	Giga Floating-Point Operations per Second (GFLOPS)	f_{MAX} (MHz)
					Adaptive Logic Modules (ALMs)	DSP Usage (18 x 18 DSPs)	M9K	M144K	Memory (Bits)				
Single	8 × 8	8 × 8	2	8	3,367	32	26	—	14,986	209	63,333	6.30	420
	16 × 16	16 × 16	2	8	3,585	32	27	—	55,562	611	21,573	6.32	421
	32 × 32	32 × 32	4	16	6,301	64	76	—	339,718	2,172	5,972	13.00	419
	64 × 64	64 × 64	8	32	11,822	128	80	16	2,382,318	8,353	1,454	24.45	388
Double	8 × 8	8 × 8	2	8	9,026	112	34	—	29,762	213	90,967	4.54	303
	16 × 16	16 × 16	2	8	9,234	112	39	—	110,756	615	32,658	4.71	314
	32 × 32	32 × 32	4	16	18,142	224	109	—	679,302	2,178	8,791	9.27	299
	64 × 64	64 × 64	8	32	35,839	448	77	32	4,765,120	8,359	2,176	17.91	284
Single (Complex)	8 × 8	8 × 8	2	8	9,996	128	59	—	22,666	220	114,411	12.80	413
	16 × 16	16 × 16	2	8	10,344	128	64	—	79,139	624	39,743	12.52	404
	32 × 32	32 × 32	4	16	20,005	256	146	—	420,519	2,181	10,937	24.99	397
	64 × 64	64 × 64	8	32	39,068	512	216	16	2,674,289	8,362	2,594	45.68	360

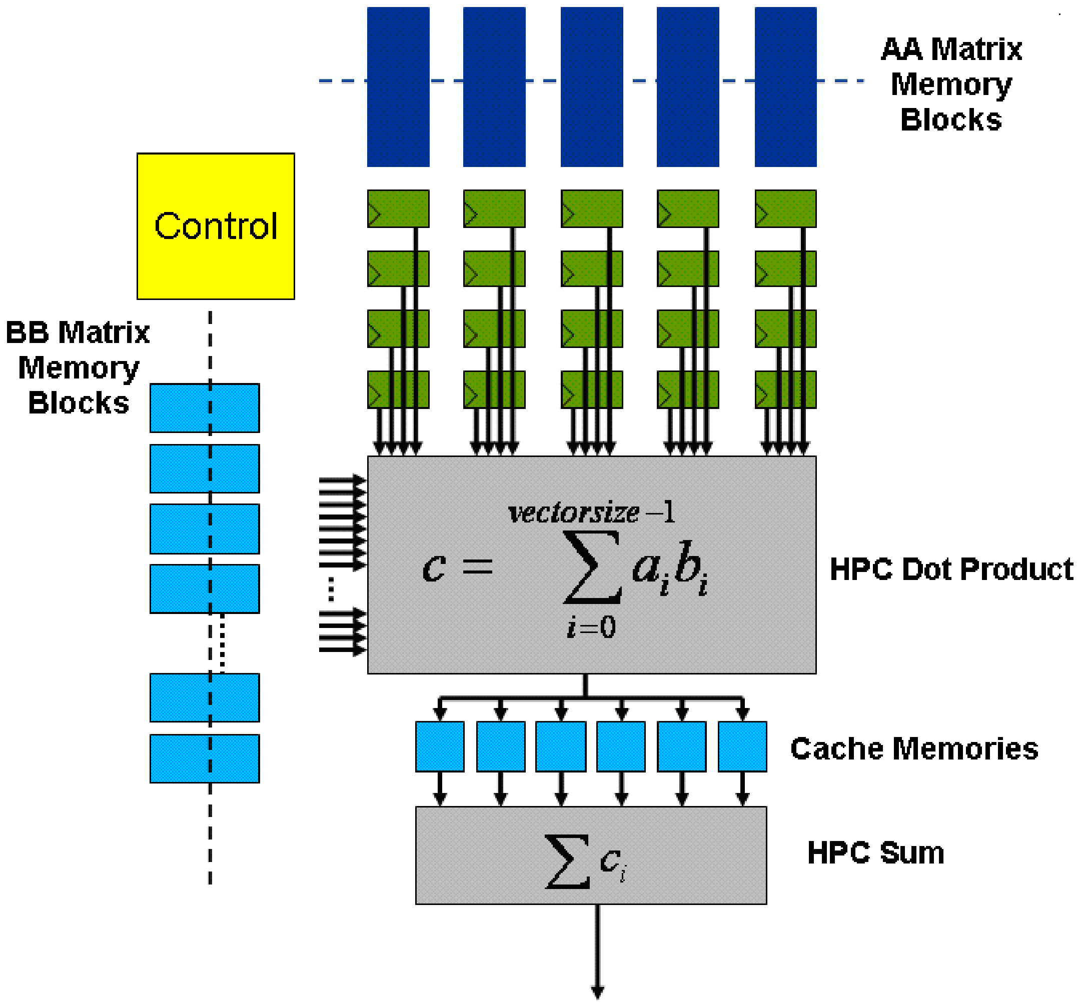
Functional Description

The matrix multiplier in the ALTFP_MATRIX_INV megafunction multiplies two matrices: AA matrix and BB matrix. The cores for the real and complex matrices are almost identical.

Figure 18–2 shows the top-level view of the matrix multiplier block. The matrix multiplier block comprises the following components:

- Memory blocks for the AA matrix storage
- Memory blocks for the BB matrix storage
- Local registers
- An FPC vector calculator unit
- Cache memories
- A parallel FPC adder unit

Figure 18–2. Top-Level View of the Matrix Multiplier Block



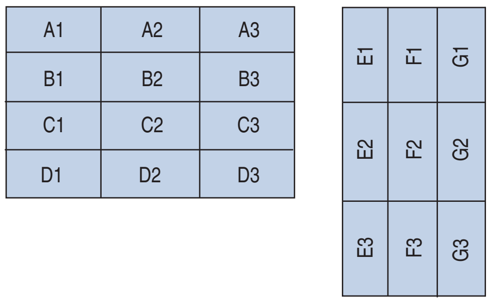
The elements of the AA matrix are loaded from the memory blocks and stored in local registers before passed through to the FPC vector calculator along with the elements of the BB matrix from another set of memory blocks. The results obtained from multiplying these elements are written to the cache memories. When all the elements for a row are written to the cache, the parallel floating-point adder sums up the elements as the output matrix. The contents of the output matrix are then burst out on a row by row basis.

AA and BB Matrix Storage

The AA and BB matrices are stored in M144K or M9K memory blocks. The Fitter tool in the Quartus II software assigns the type of memory to these matrices based on the available resources.

Each AA matrix location stores two consecutive words for the real matrix core and a single word for the complex core. Because the AA matrix is allocated with fewer memory blocks, it has less bandwidth compared to the BB matrix. You must expand the bandwidth of the AA matrix by storing multiple loads from the AA matrix in local registers. Figure 18–3 shows the logical storage of the AA and BB matrix vectors.

Figure 18–3. Logical Storage of the AA and BB Matrix Vectors



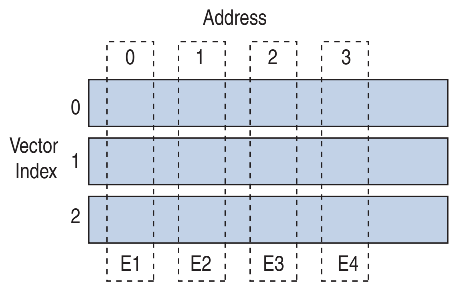
For larger matrix sizes, the matrix data is split into several sections of equal widths. The number of elements in each section is also the width side. You can obtain the size of the width by dividing `COLUMNSAA` with `VECTORSIZE`. The sections in the AA matrix are known as sub-rows and the sections in the BB matrix are known as sub-columns. In Figure 18–3, the AA matrix has three sub-rows and the BB matrix has three sub-columns.

Vectors A1, A2, and A3 are the sub-rows of the first row of the AA matrix, while vectors E1, E2, and E3 are the sub-columns of the first column of the BB matrix. To load vector A1, you require multiple reads which are then held in local registers, while sub-column vectors E1, F1, and G1 are loaded from the BB matrix storage. These sub-column vectors from the BB matrix are loaded at one clock cycle per vector.

When a sub-column of the BB matrix is processed, the results are stored in the cache memory, and the AA matrix row index and BB matrix column index are increased by the `VECTORSIZE` value. Then, a new AA matrix sub-row and a BB matrix sub-column are loaded and processing starts. The results are once again stored in the cache memory, but in the next memory location.

Figure 18–4 shows the physical storage of the BB matrix.

Figure 18–4. Physical Storage of the BB Matrix



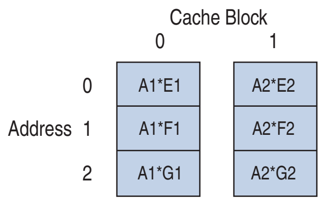
For a real matrix core, the number of memory blocks for the BB matrix is the same as the value of `VECTORSIZE`. For a complex matrix core, you need twice the number of memory blocks. These blocks are all read in parallel. The logical and physical storages for the BB matrix are different.

The entries for the BB matrix are written to the storage row by row, for example, Vector index 0/ Address 0, Vector index 0/ Address 3, Vector index 0/ Address 6, Vector index 1/ Address 0 and so on. The vectors are read in the following order: E1, F1, G1, E2 and so on, and the parallel access follows an address sequence of 0, 3, 6, 1, 4, 7, 2, 5, and 8.

Cache Storage

Figure 18–5 shows the cache addressing scheme.

Figure 18–5. Cache Addressing Scheme

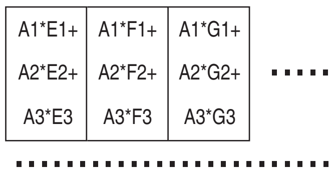


The cache storage consists of memory blocks—one memory block for each real core and two for each complex core. There are six cache memories, which are written to sequentially but read from in parallel.

The depth of each cache memory is the same as the value of `COLUMNSAA`. In Figure 18–5, the cache memory has a depth of three. The number of cache memories used is determined by the number of iterations per element (`COLUMNSAA`/ `VECTORSIZE`). Any outputs from unused memories are zeroed. Cache memories are not double-buffered because the start of the memory is read before the writing of the partial vector products of the current matrix is complete.

Figure 18–6 shows that the first element in the result matrix is the sum of the vector products of A1*E1, A2*E2 and A3*E3. However, because of the bandwidth differences between the memory storages of the AA and BB matrices, the sequence of the vector operations is A1*E1, A1*F1, A1*G1, A2*E2, and so on. The products from these operations are stored in Cache 0/Address 0, Cache 0/Address 1, Cache 0/Address 2, and Cache 1/Address 0.

Figure 18–6. Output Element Components

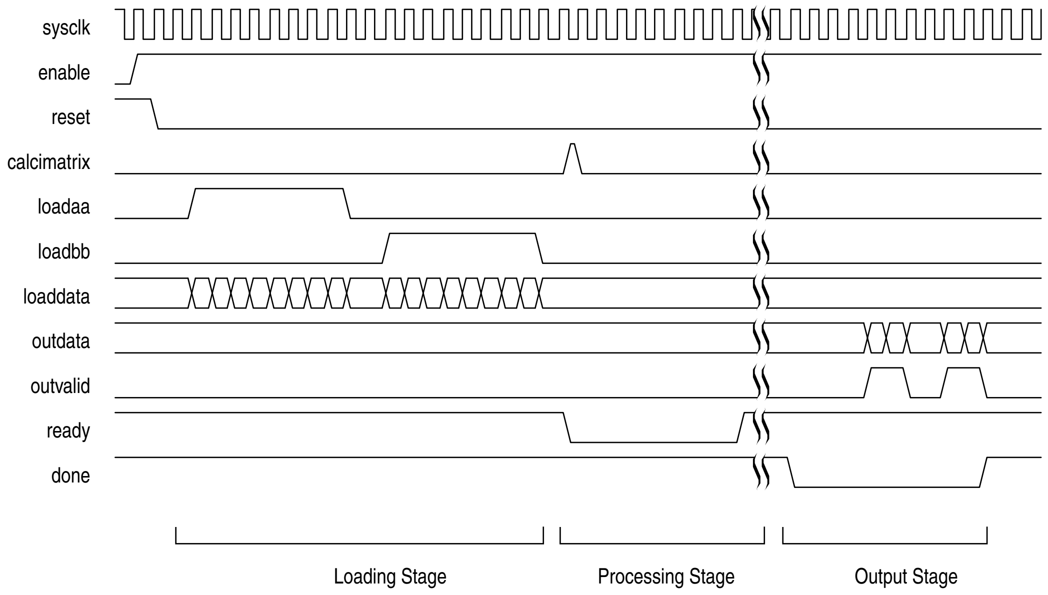


After each cache memory is filled up, all six cache blocks are read in parallel, starting with address location 0. The contents from each cache block are then added and the results are burst out in an entire row to produce the output matrix.

Matrix Multiplication Operation

Figure 18–7 shows the timing diagram for a single matrix multiplication operation.

Figure 18–7. Matrix Multiplication Timing Diagram



The following sequence describes the matrix multiplication operation:

1. The operation begins when the enable signal asserts and the reset signal deasserts.
2. The `loadaa` signal is asserted to load data from the `loaddata[]` port for the AA matrix. As long as the `loadaa` signal is high, data is loaded continuously for the AA matrix.
3. The `loadaa` signal deasserts.

4. The `loadbb` signal is asserted to load data from the `loaddata[]` port for the BB matrix.
5. The `calcmatrix` signal is asserted for approximately one clock cycle. This assertion triggers the deassertion of the `ready` signal which indicates the start of the processing stage.
6. The `ready` signal is asserted a few clock cycles later to allow data to load for a new set of matrix.
7. The `done` signal deasserts.
8. The `outvalid` signal is asserted multiple times to signify the availability of valid data on the `outdata[]` port. The number of times this signal is asserted equals the number of rows found in the output matrix.
9. The `done` signal is asserted when the last row of the output matrix has been burst out. This assertion signifies the end of the matrix multiplication operation.

For a best system performance, the cycles required to load a matrix set must be less than or equal to the number of cycles needed to multiply the matrix. This equation ensures that the core is able to process continuously without having to wait for the interface to catch up. Because the core usually takes a longer time to load than processing the matrix multiplication operation, the performance analysis are made in terms of:

System duty cycle = $min(1, t_{process}/t_{load})$

The naïve number of operations is the number of elements in the result matrix multiplied by the number of operations required to calculate the element, or `COLUMNSAA * 2`, as the dot product calculation for every element involves a multiplication and an addition operation. For example:

t_{naive} = `ROWSAA * COLUMNSBB * COLUMNSAA * 2`

The time to load the core is the same as the total number of elements in the two-input matrices, or

t_{load} = `(COLUMNSAA * ROWSAA) + (COLUMNSAA * COLUMNSBB)`

The time to process a core depends on the following matrix size and core parameters:

- There are `ROWSAA` rows to process.
- Each row consists of `COLUMNSBB` elements.
- Each element in the row requires `COLUMNSAA/VECTORSIZE` iterations to calculate.
- Each iteration requires `VECTORSIZE/(2*BLOCKS)` cycles to set up.

Based on these requirements, the time to process is

$t_{process}$ = `ROWSAA * [COLUMNSBB * (COLUMNSAA/ VECTORSIZE) + (COLUMNSAA/ VECTORSIZE) * VECTORSIZE/(2*BLOCKS)] + LATENCYTOP + LATENCYBOT`

or

$t_{process}$ = `ROWSAA * COLUMNSAA/ VECTORSIZE * [COLUMNSBB + VECTORSIZE/(2*BLOCKS)] + LATENCYTOP + LATENCYBOT`

The actual efficiency of the core is ($t_{naïve}$/`VECTORSIZE * 2 *` $t_{process}$) `*` system duty cycle.

The following section provides examples to carry out the matrix multiplication operation.

Example 1

The AA matrix and the BB matrix have the following parameters:

- `COLUMNSAA` = 64
- `ROWSAA` = 20
- `COLUMNSBB` = 15
- `VECTORSIZE` = 32
- `BLOCKS` = 4

t_{load} = (64 * 20) + (64 * 15) = 2,240 clocks

$t_{process}$ = 20 * (64 / 32) * [15 + 32 / (2 * 4)] + 41 + 24 = 825 clocks

The system duty cycle is therefore, (825 / 2,240) = 0.37

$t_{naïve}$ = 64 * 20 * 15 * 2 = 38,400 clocks

Efficiency = [38,400 / (32 * 2 * 825)] * 0.37 = 27%

For high performance, the matrix sizes must be closer to a square.

Example 2

The AA matrix and the BB matrix have the following parameters:

- `COLUMNSAA` = 192
- `ROWSAA` = 192
- `COLUMNSBB` = 192
- `VECTORSIZE` = 96
- `BLOCKS` = 24

t_{load} = 2 * 192 * 192 = 73,728 blocks

$t_{process}$ = 192 * (192 / 96) * [192 + 96 / (2 * 24)] + 51 + 24 = 74,571 clocks

The system duty cycle is therefore, 74,571 / 73,728 = 1

$t_{naïve}$ = 192 * 192 * 192 * 2 = 14,155,776

Efficiency = [14,155,776 / (96 * 2 * 74,571)] * 0.1 = 99%

At a system speed of 250 MHz, the core has an equivalent performance of 250 * 0.99 * 2 * 96 = 47.52 GFLOPs.

Specifications

This section describes the component declarations, ports, and parameters of the ALTFP_MATRIX_MULT megafunction. The ports and parameters are available to customize the ALTFP_MATRIX_MULT megafunction according to your application.

You must manually include the **altfpc.v/vhd** library file to your project.

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altfp_matrix_mult
   generic (
      blocks : natural := 0;
      cluster : natural := 8;
      columnsaa : natural := 0;
      columnsbb : natural := 0;
      intended_device_family : string := "unused";
      rowsaa : natural := 0;
      vectorsize : natural := 0;
      width_exp : natural := 8;
      width_man : natural := 23;
      lpm_hint : string := "UNUSED";
      lpm_type : string := "altfp_matrix_mult"
   );
   port(
      calcmatrix : in std_logic := '0';
      done : out std_logic;
      enable : in std_logic := '1';
      loadaa : in std_logic := '0';
      loadbb : in std_logic := '0';
     loaddata : in std_logic_vector(width_exp+width_man+1-1 downto 0) := (others => '0');
      outdata : out std_logic_vector(width_exp+width_man+1-1 downto 0);
      outvalid : out std_logic;
      ready : out std_logic;
      reset : in std_logic := '0';
      sysclk : in std_logic
   );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf_altera_mf_components.all;
```

Ports and Parameters

Table 18–3 lists the input ports of the ALTFP_MATRIX_MULT megafunction.

Table 18–3. ALTFP_MATRIX_MULT Megafunction Input Ports

Port Name	Required	Description
`sysclk`	Yes	The clock input port for the ALTFP_MATRIX_MULT megafunction.
`reset`	No	Asynchronous reset. The ALTFP_MATRIX_MULT megafunction is asynchronously reset when the `reset` signal is asserted.
`enable`	No	Clock enable that allows calculation to take place when asserted. When this signal is deasserted, no operation occurs and the outputs remain unchanged.
`calcmatrix`	Yes	Control input. Asserted for at least one clock cycle to start processing of the matrices.
`loadaa`	Yes	Control input. When asserted, the megafunction loads the `loaddata[]` bus into the memory for the AA matrix.
`loadbb`	Yes	Control input. When asserted, the megafunction loads the `loaddata[]` bus into the memory for the BB matrix.
`loaddata[]`	Yes	Control input. 32-bit single-precision format or 64-bit double-precision format matrix input value. Matricies are loaded row by row.
`loaddatareal[]`	No	Data input for real values. 32-bit single-precision format, real value matrix input. Matrices are loaded row by row. This port must be declared when using the complex number mode.
`loaddataimag[]`	No	Data input for imaginary values. 32-bit single-precision format, imaginary value matrix input. Matrices are loaded row by row. This port must be declared when using the complex number mode.

Table 18–4 lists the output ports of the ALTFP_MATRIX_MULT megafunction.

Table 18–4. ALTFP_MATRIX_MULT Megafunction Output Ports (Part 1 of 2)

Port Name	Required	Description
`outdata[]`	Yes	Floating-point output result. 32-bit single-precision format or 64-bit double-precision format matrix output value. The result matrix is written out row by row. This port must be declared when using the normal number mode.
`outdatareal[]`	Yes	Data output for real result values. 32-bit single-precision format, real result value matrix output. Matrices are loaded row by row. This port must be declared when using the complex number mode.
`outdataimag[]`	Yes	Data output for real result values. 32-bit single-precision format, imaginary result value matrix output. Matrices are loaded row by row. This port must be declared when using the complex number mode.

Table 18–4. ALTFP_MATRIX_MULT Megafunction Output Ports (Part 2 of 2)

Port Name	Required	Description
outvalid	Yes	Handshaking signal. When asserted, valid output data is available on the outdata[] port. An entire row of the result matrix is written out as a burst. There is a parameter-determined gap between the row outputs.
done	Yes	Handshaking signal. When asserted, the last output is written. A new matrix multiply can be started with the calcmatrix port. Deassertion of the done port follows assertion of the ready port by some fixed amount, depending on the parameters.
ready	Yes	Handshaking signal. When asserted, the core has completed reading both matrices for processing, and the loading of a new set of matrices can begin. After reset, a second pair of AA and BB matrices can be loaded, even if the ready port is low, after which the double buffer is full. Further matrix loads must wait for the ready port.

Table 18–5 lists the parameters of the ALTFP_MATRIX_MULT megafunction.

Table 18–5. ALTFP_MATRIX_MULT Megafunction Parameters

Port Name	Type	Required	Description
WIDTH_EXP	Integer	Yes	Specifies the precision of the exponent. If this parameter is not specified, the value is 8. The bias of the exponent is always set to $2^{(\text{WIDTH\_EXP} - 1)} - 1$, that is, 127 for the single-precision format and 1023 for the double-precision format. The value of WIDTH_EXP must be 8 for the single-precision format, 11 for the double-precision format, and a minimum of 11 for the single-extended precision format. The value of WIDTH_EXP must be less than the value of WIDTH_MAN, and the sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
WIDTH_MAN	Integer	Yes	Specifies the precision of the mantissa. If this parameter is not specified, the default is 23. When WIDTH_EXP 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 value of WIDTH_MAN must be greater than the value of WIDTH_EXP, and the sum of WIDTH_EXP and WIDTH_MAN must be less than 64.
COLUMNSAA	Integer	Yes	The number of columns in matrix AA, and therefore also the number of rows in matrix BB. This value must be an even multiple of 2 x BLOCKS.
ROWSAA	Integer	Yes	Number of rows in matrix AA.
COLUMNSBB	Integer	Yes	Number of columns in matrix BB.
BLOCKS	Integer	No	The number of M144 memory blocks for the double buffered storage of matrix AA. The allowable range is from 2 to 48.
VECTORSIZE	String	No	The number of elements in the dot product core. The core library supports lengths of 8, 16, 32, 64, 96, and 128.

Document Revision History

Table 18–6 lists the revision history for this user guide.

Table 18–6. Document Revision History (Part 1 of 2)

Date	Document Version	Changes Made
November 2013	7.0	■ Added "ALTERA_FP_FUNCTIONS" on page 3–1. ■ Added "ALTERA_FP_ACC_CUSTOM" on page 2–1. ■ Updated Table 1–1 on page 1–1 to list ALTERA_FP_FUNCTIONS and ALTERA_FP_ACC_CUSTOM. ■ Updated the "ALTFP_MATRIX_INV" on page 17–1 section to include 4 x 4 and 6 x 6 dimensions. ■ Updated "Rounding" on page 1–4 to clarify that the code for round-to-nearest-even mode is `TO_NEAREST`. ■ Removed Design Example section for "ALTFP_MATRIX_MULT" on page 18–1. ■ Removed device family support for HardCopy III, HardCopy IV, Stratix II, and Stratix II GX devices from "Device Family Support" on page 1–2.
November 2011	6.0	Updated "General Features" on page 1–2.
May 2011	5.0	Added "ALTFP_ATAN" on page 12–1.
January 2011	4.0	Added "ALTFP_SINCOS" on page 13–1.
July 2010	3.0	■ Updated architecture information for the following sections: ■ ALTFP_MATRIX_MULT ■ ALTFP_MATRIX_INV. ■ Added specification information in all sections.

Table 18–6. Document Revision History (Part 2 of 2)

Date	Document Version	Changes Made
November 2009	2.0	■ Updated resource utilization information for the following sections: ■ ALTFP_ADD_SUB ■ ALTFP_DIV ■ ALTFP_MULT ■ ALTFP_SQRT ■ ALTFP_EXP ■ ALTFP_INV ■ ALTFP_INV_SQRT ■ ALTFP_LOG ■ ALTFP_COMPARE ■ ALTFP_CONVERT ■ ALTFP_MATRIX_MULT ■ Added the ALTFP_MATRIX_INV section. ■ Updated the Ports and Parameters section for all floating-point megafunctions.
March 2009	1.0	Initial release.