# Integer Arithmetic Megafunctions



The Altera® integer arithmetic megafunctions offer you the convenience of performing mathematical operations on FPGAs through parameterizable functions that are optimized for Altera device architectures. These functions offer efficient logic synthesis and device implementation. You can customize the megafunctions by configuring various parameters to accommodate your needs.

Altera's integer arithmetic megafunctions are divided into the following two categories:

- Library of parameterized modules (LPM) megafunctions
- Altera-specific (ALT) megafunctions

Table 1 lists the integer arithmetic megafunctions described in this user guide:

**Table 1. List of Megafunctions**

| Megafunction Name | Function Overview |
|---|---|
| **LPM Megafunctions** | |
| LPM_ABS | Absolute value |
| LPM_ADD_SUB | Adder/Subtractor |
| LPM_COMPARE | Comparator |
| LPM_COUNTER | Counter |
| LPM_DIVIDE | Divider |
| LPM_MULT | Multiplier |
| **Altera-specific (ALT) Megafunctions** | |
| ALTACCUMULATE | Accumulator |
| ALTECC | ECC Encoder/Decoder |
| ALTMEMMULT | Memory-based Constant Coefficient Multiplier |
| ALTMULT_ACCUM | Multiply-Accumulator |
| ALTMULT_ADD | Multiply-Adder |
| ALTMULT_COMPLEX | Complex Multiplier |
| ALTSQRT | Integer Square-Root |
| PARALLEL_ADD | Parallel Adder |

The LPM_ABS, LPM_ADD_SUB, LPM_COMPARE, LPM_COUNTER, LPM_DIVIDE, ALTACCUMULATE, and ALTSQRT megafunctions are not available in the MegaWizard™ Plug-In Manager because they are not supported for new designs in the Quartus® II 10.0 software. However, you can still compile existing designs in the new Quartus II software.

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

This user guide assumes that you are familiar with megafunctions and how to create them. If you are unfamiliar with megafunctions, refer to the *Introduction to Megafunctions User Guide*.

Altera also provides floating-point megafunctions. For more information about the floating-point megafunctions, refer to the *Floating-Point Megafunctions User Guide* on the Altera website.

## Device Family Support

All Altera integer arithmetic megafunctions are available for Cyclone®, Stratix®, Arria®, and HardCopy® device series.

## Design Flow

Altera 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. For more details, refer to the "Ports and Parameters" section for your selected megafunction.

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

The designs are simulated in the ModelSim®-Altera software to generate a waveform display of the device behavior. You should be familiar with the ModelSim-Altera software before using the design examples. To get started with the ModelSim-Altera software, refer to the ModelSim-Altera Software Support page on the Altera website. The support page includes links to such topics as installation, usage, and troubleshooting. For more details about the design example for a specific megafunction, refer to the "Design Example" section for that megafunction.

Design examples are provided only for the ALT megafunctions in this user guide. No design examples are available for the LPM megafunctions because of the simple and self-explanatory nature of these megafunctions.

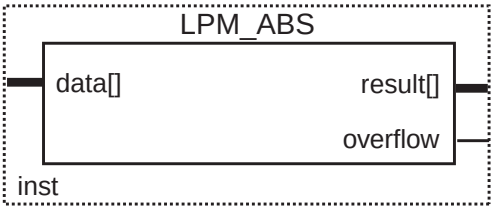
# LPM_ABS (Absolute Value)

The LPM_ABS megafunction implements an absolute value function that calculates the absolute value of its input. The overflow output indicates that the positive equivalent of the current input value does not exist.

A two's complement number system represents the input. In the two's complement system, there is one more negative number than there are positive numbers. Because of this asymmetric range, the negative number $-2^{(\text{data bit width}-1)}$ does not have a positive equivalent. For example, the possible values for a 4-bit data width ranges from –8 or $(-2^{(4-1)})$ to +7 or $(2^{(4-1)}-1)$. The LPM_ABS megafunction generates an overflow indication for an input value of –8. All other negative numbers have equivalent positive representations.

Figure 1 shows the ports for the LPM_ABS megafunction.

**Figure 1. LPM_ABS Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_ABS megafunction offers the following features:

- Calculates the absolute value of an input
- Supports data width of 2–256 bits
- Supports optional overflow output port

## Resource Utilization and Performance

Table 2 lists the resource utilization and performance information for the LPM_ABS megafunction.

**Table 2. LPM_ABS Resource Utilization and Performance (1)**

| Device family | Input data width | Logic Usage | | | $f_{MAX}$ (MHz) |
|---|---|---|---|---|---|
| | | Adaptive Look-Up Table (ALUTs) | Dedicated Logic Register (DLRs) | Adaptive Logic Module (ALMs) | |
| Stratix III | 8 | 9 | 0 | 0 | N/A |
| | 16 | 17 | 0 | 0 | |
| | 32 | 33 | 0 | 0 | |

**Note to Table 2:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_abs ( result, overflow, data );
parameter lpm_type = "lpm_abs";
parameter lpm_width = 1;
parameter lpm_hint = "UNUSED";
input  [lpm_width-1:0] data;
output [lpm_width-1:0] result;
output overflow;
endmodule
```

### VHDL Component Declaration

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

```
component LPM_ABS
        generic (LPM_WIDTH : natural;
LPM_TYPE: string := L_ABS;
LPM_HINT : string := "UNUSED");
port (DATA : in std_logic_vector(LPM_WIDTH-1 downto 0);
RESULT : out std_logic_vector(LPM_WIDTH-1 downto 0);
OVERFLOW : 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 lpm;
USE lpm.lpm_components.all;
```

## Ports and Parameters

Table 3 lists the input ports, Table 4 lists the output ports, and Table 5 lists the LPM_ABS megafunction parameters.

**Table 3. LPM_ABS Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | Yes | Data input to the absolute value function. The size of the input port depends on the LPM_WIDTH parameter value. |

**Table 4. LPM_ABS Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Data output from the absolute value function. The size of the output port depends on the LPM_WIDTH parameter value. |
| overflow | No | Optional overflow exception output. The overflow port asserts to indicate that no positive equivalent exists for a particular input data. |

**Table 5. LPM_ABS Megafunction Parameters**

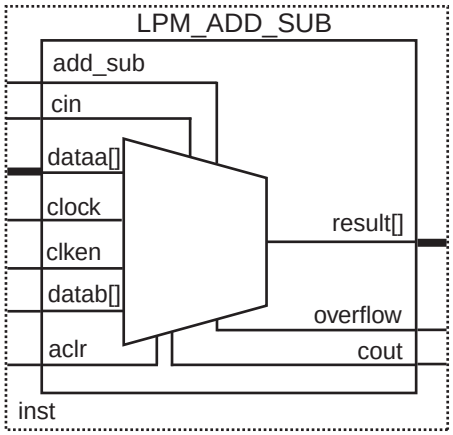
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTH | Integer | Yes | Specifies the widths of the data[] and result[] ports. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| IGNORE_CARRY_BUFFERS | String | No | Specifies whether to ignore carry buffers. |

# LPM_ADD_SUB (Adder/Subtractor)

The LPM_ADD_SUB megafunction lets you implement an adder or a subtractor to add or subtract sets of data to produce an output containing the sum or difference of the input values.

Figure 2 shows the ports for the LPM_ADD_SUB megafunction.

**Figure 2. LPM_ADD_SUB Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_ADD_SUB megafunction offers the following features:

- Generates adder, subtractor, and dynamically configurable adder/subtractor functions
- Supports data width of 1–256 bits
- Supports data representation format such as signed and unsigned
- Supports optional carry-in (borrow-out), asynchronous clear, and clock enable input ports
- Supports optional carry-out (borrow-in) and overflow output ports
- Assigns either one of the input data buses to a constant
- Supports pipelining with configurable output latency

## Resource Utilization and Performance

Table 6 lists the resource utilization and performance information for the LPM_ADD_SUB megafunction.

**Table 6. LPM_ADD_SUB Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 20 | 2 | 21 | 0 | 11 | 784 |
| | 96 | 5 | 509 | 0 | 267 | 420 |
| | 256 | 10 | 2669 | 0 | 1420 | 372 |
| Stratix IV | 20 | 2 | 21 | 0 | 11 | 863 |
| | 96 | 5 | 509 | 0 | 270 | 569 |
| | 256 | 10 | 2669 | 0 | 1423 | 455 |

**Notes to Table 6:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_add_sub ( result, cout, overflow,add_sub, cin, dataa, datab, clock, clken,
aclr );
parameter lpm_type = "lpm_add_sub";
parameter lpm_width = 1;
parameter lpm_direction  = "UNUSED";
parameter lpm_representation = "UNSIGNED";
parameter lpm_pipeline = 0;
parameter lpm_hint = "UNUSED";
input  [lpm_width-1:0] dataa, datab;
input  add_sub, cin;
input  clock;
input  clken;
input  aclr;
output [lpm_width-1:0] result;
output cout, overflow;
endmodule
```

## VHDL Component Declaration

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

```
component LPM_ADD_SUB
        generic (LPM_WIDTH : natural;
LPM_DIRECTION : string := "UNUSED";
LPM_REPRESENTATION: string := "SIGNED";
                 LPM_PIPELINE : natural := 0;
LPM_TYPE : string := L_ADD_SUB;
LPM_HINT : string := "UNUSED");
port (DATAA : in std_logic_vector(LPM_WIDTH-1 downto 0);
DATAB : in std_logic_vector(LPM_WIDTH-1 downto 0);
ACLR : in std_logic := '0';
CLOCK : in std_logic := '0';
CLKEN : in std_logic := '1';
CIN : in std_logic := 'Z';
ADD_SUB : in std_logic := '1';
RESULT : out std_logic_vector(LPM_WIDTH-1 downto 0);
COUT : out std_logic;
OVERFLOW : 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 lpm;
USE lpm.lpm_components.all;
```

## Ports and Parameters

Table 7 lists the input ports, Table 8 lists the output ports, and Table 9 lists the LPM_ADD_SUB megafunction parameters.

**Table 7. LPM_ADD_SUB Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| cin | No | Carry-in to the low-order bit. For addition operations, the default value is 0. For subtraction operations, the default value is 1. |
| dataa[] | Yes | Data input. The size of the input port depends on the LPM_WIDTH parameter value. |
| datab[] | Yes | Data input. The size of the input port depends on the LPM_WIDTH parameter value. |
| add_sub | No | Optional input port to enable dynamic switching between the adder and subtractor functions. If the LPM_DIRECTION parameter is used, add_sub cannot be used. If omitted, the default value is ADD. Altera recommends that you use the LPM_DIRECTION parameter to specify the operation of the LPM_ADD_SUB function, rather than assigning a constant to the add_sub port. |
| clock | No | Input for pipelined usage. The clock port provides the clock input for a pipelined operation. For LPM_PIPELINE values other than 0 (default), the clock port must be enabled. |
| clken | No | Clock enable for pipelined usage. When the clken port is asserted high, the adder/subtractor operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear for pipelined usage. The pipeline initializes to an undefined (X) logic level. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. |

**Table 8. LPM_ADD_SUB Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Data output. The size of the output port depends on the LPM_WIDTH parameter value. |
| cout | No | Carry-out (borrow-in) of the most significant bit (MSB). The cout port has a physical interpretation as the carry-out (borrow-in) of the MSB. The cout port detects overflow in UNSIGNED operations. The cout port operates in the same manner for SIGNED and UNSIGNED operations. |
| overflow | No | Optional overflow exception output. The overflow port has a physical interpretation as the XOR of the carry-in to the MSB with the carry-out of the MSB. The overflow port asserts when results exceed the available precision, and is used only when the LPM_REPRESENTATION parameter value is SIGNED. |

**Table 9. LPM_ADD_SUB Megafunction Parameters**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTH | Integer | Yes | Specifies the widths of the dataa[], datab[], and result[] ports. |
| LPM_DIRECTION | String | No | Values are ADD, SUB, and UNUSED. If omitted, the default value is DEFAULT, which directs the parameter to take its value from the add_sub port. The add_sub port cannot be used if LPM_DIRECTION is used. Altera recommends that you use the LPM_DIRECTION parameter to specify the operation of the LPM_ADD_SUB function, rather than assigning a constant to the add_sub port. |
| LPM_REPRESENTATION | String | No | Specifies the type of addition performed. Values are SIGNED and UNSIGNED. If omitted, the default value is SIGNED. When this parameter is set to SIGNED, the adder/subtractor interprets the data input as signed two's complement. |
| LPM_PIPELINE | Integer | No | Specifies the number of latency clock cycles associated with the result[] output. A value of zero (0) indicates that no latency exists, and that a purely combinational function will be instantiated. If omitted, the default value is 0 (non-pipelined). |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| ONE_INPUT_IS_CONSTANT | String | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the ONE_INPUT_IS_CONSTANT parameter in VHDL design files. Values are YES, NO, and UNUSED. Provides greater optimization if one input is constant. If omitted, the default value is NO. |

**Table 9. LPM_ADD_SUB Megafunction Parameters**

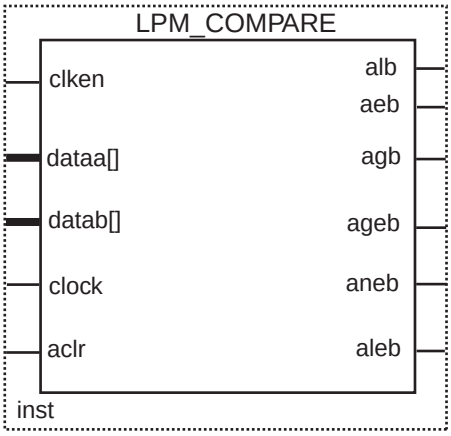
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MAXIMIZE_SPEED | Integer | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the MAXIMIZE_SPEED parameter in VHDL design files. You can specify a value between 0 and 10. If used, the Quartus II software attempts to optimize a specific instance of the LPM_ADD_SUB function for speed rather than routability, and overrides the setting of the Optimization Technique logic option. If MAXIMIZE_SPEED is unused, the value of the Optimization Technique option is used instead. If the setting for MAXIMIZE_SPEED is 6 or higher, the Compiler optimizes the LPM_ADD_SUB megafunctions for higher speed using carry chains; if the setting is 5 or less, the Compiler implements the design without carry chains. This parameter must be specified for Cyclone, Stratix, and Stratix GX devices only when the add_sub port is not used. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the LPM_ADD_SUB megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |

# LPM_COMPARE (Comparator)

The LPM_COMPARE megafunction compares the value of two sets of data to determine the relationship between them. In its simplest form, you can use an exclusive-OR gate to determine whether two bits of data are equal.

Figure 3 shows the ports for the LPM_COMPARE megafunction.

**Figure 3. LPM_COMPARE Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_COMPARE megafunction offers the following features:

- Generates a comparator function to compare two sets of data
- Supports data width of 1–256 bits
- Supports data representation format such as signed and unsigned
- Produces the following output types:
  - `alb` (input A is less than input B)
  - `aeb` (input A is equal to input B)
  - `agb` (input A is greater than input B)
  - `ageb` (input A is greater than or equal to input B)
  - `aneb` (input A is not equal to input B)
  - `aleb` (input A is less than or equal to input B)
- Supports optional asynchronous clear and clock enable input ports
- Assigns the `datab[]` input to a constant
- Supports pipelining with configurable output latency

## Resource Utilization and Performance

Table 10 provides resource utilization and performance information for the LPM_COMPARE megafunction.

**Table 10. LPM_COMPARE Resource Utilization and Performance** *(1)*

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) *(2)* |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 8 | 2 | 4 | 0 | 3 | 547 |
| | 96 | 5 | 64 | 0 | 71 | 357 |
| | 256 | 10 | 171 | 0 | 185 | 342 |
| Stratix IV | 8 | 2 | 4 | 0 | 3 | 548 |
| | 96 | 5 | 64 | 0 | 67 | 363 |
| | 256 | 10 | 171 | 0 | 185 | 313 |

**Notes to Table 10:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_compare ( alb, aeb, agb, aleb, aneb, ageb, dataa, datab,
clock, clken, aclr );
parameter lpm_type = "lpm_compare";
parameter lpm_width = 1;
parameter lpm_representation = "UNSIGNED";
parameter lpm_pipeline = 0;
parameter lpm_hint = "UNUSED";
input  [lpm_width-1:0] dataa, datab;
input  clock;
input  clken;
input  aclr;
output alb, aeb, agb, aleb, aneb, ageb;
endmodule
```

## VHDL Component Declaration

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

```
component LPM_COMPARE
        generic (LPM_WIDTH : natural;
LPM_REPRESENTATION : string := "UNSIGNED";
LPM_PIPELINE : natural := 0;
LPM_TYPE: string := L_COMPARE;
LPM_HINT : string := "UNUSED");
port (DATAA : in std_logic_vector(LPM_WIDTH-1 downto 0);
DATAB : in std_logic_vector(LPM_WIDTH-1 downto 0);
ACLR : in std_logic := '0';
CLOCK : in std_logic := '0';
CLKEN : in std_logic := '1';
AGB : out std_logic;
AGEB : out std_logic;
AEB : out std_logic;
ANEB : out std_logic;
ALB : out std_logic;
ALEB : 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 lpm;
USE lpm.lpm_components.all;
```

## Ports and Parameters

Table 11 lists the input ports, Table 12 lists the output ports, and Table 13 lists the LPM_COMPARE megafunction parameters.

**Table 11. LPM_COMPARE Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| dataa[] | Yes | Data input. The size of the input port depends on the LPM_WIDTH parameter value. |
| datab[] | Yes | Data input. The size of the input port depends on the LPM_WIDTH parameter value. |
| clock | No | Clock input for pipelined usage. The clock port provides the clock input for a pipelined operation. For LPM_PIPELINE values other than 0 (default), the clock port must be enabled. |
| clken | No | Clock enable for pipelined usage. When the clken port is asserted high, the comparison operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear for pipelined usage. The pipeline initializes to an undefined (X) logic level. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. |

**Table 12. LPM_COMPARE Megafunction Output Ports** *(1)*

| Port Name | Required | Description |
|---|---|---|
| alb | No | Output port for the comparator. Asserted if input A is less than input B. |
| aeb | No | Output port for the comparator. Asserted if input A is equal to input B. |
| agb | No | Output port for the comparator. Asserted if input A is greater than input B. |
| ageb | No | Output port for the comparator. Asserted if input A is greater than or equal to input B. |
| aneb | No | Output port for the comparator. Asserted if input A is not equal to input B. |
| aleb | No | Output port for the comparator. Asserted if input A is less than or equal to input B. |

**Note to Table 12:**

(1) One of these outputs is required. The rest of the outputs are optional.

**Table 13. LPM_COMPARE Megafunction Parameters**

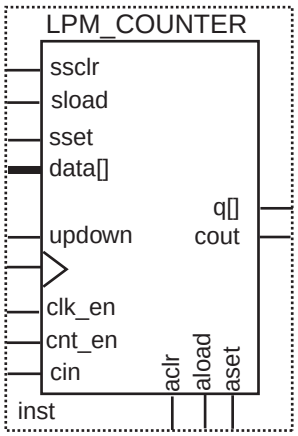
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTH | Integer | Yes | Specifies the widths of the dataa[] and datab[] ports. |
| LPM_REPRESENTATION | String | No | Specifies the type of comparison performed. Values are SIGNED and UNSIGNED. If omitted, the default value is UNSIGNED. When this parameter value is set to SIGNED, the comparator interprets the data input as signed two's complement. |
| LPM_PIPELINE | Integer | No | Specifies the number of clock cycles of latency associated with the alb, aeb, agb, ageb, aleb, or aneb output. A value of zero (0) indicates that no latency exists, and that a purely combinational function will be instantiated. If omitted, the default value is 0 (non-pipelined). |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**) . The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the LPM_COMPARE megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| ONE_INPUT_IS_CONSTANT | String | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the ONE_INPUT_IS_CONSTANT parameter in VHDL design files. Values are YES, NO, or UNUSED. Provides greater optimization if an input is constant. If omitted, the default value is NO. |

# LPM_COUNTER (Counter)

The LPM_COUNTER megafunction is a binary counter that creates up counters, down counters and up or down counters with outputs of up to 256 bits wide.

Figure 4 shows the ports for the LPM_COUNTER megafunction.

**Figure 4. LPM_COUNTER Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_COUNTER megafunction offers the following features:

- Generates up, down, and up/down counters
- Generates the following counter types:
  - Plain binary— the counter increments starting from zero or decrements starting from 255
  - Modulus—the counter increments to or decrements from the modulus value specified by the user and repeats
- Supports optional synchronous clear, load, and set input ports
- Supports optional asynchronous clear, load, and set input ports
- Supports optional count enable and clock enable input ports
- Supports optional carry-in and carry-out ports

## Resource Utilization and Performance

Table 14 provides resource utilization and performance information for the LPM_COUNTER megafunction.

**Table 14. LPM_COUNTER Resource Utilization and Performance** *(1)*

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) *(2)* |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 4 | – | 9 | 4 | 6 | 723 |
| | 8 | – | 9 | 8 | 5 | 808 |
| | 16 | – | 17 | 16 | 9 | 705 |
| | 24 | – | 25 | 24 | 13 | 583 |
| | 32 | – | 33 | 32 | 17 | 489 |
| | 64 | – | 65 | 64 | 33 | 329 |
| Stratix IV | 4 | – | 9 | 4 | 6 | 768 |
| | 8 | – | 9 | 8 | 5 | 896 |
| | 16 | – | 17 | 16 | 9 | 825 |
| | 24 | – | 25 | 24 | 13 | 716 |
| | 32 | – | 33 | 32 | 17 | 639 |
| | 64 | – | 65 | 64 | 33 | 470 |

**Notes to Table 14:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_counter ( q, data, clock, cin, cout, clk_en, cnt_en, updown,
aset, aclr, aload, sset, sclr, sload, eq );
parameter lpm_type = "lpm_counter";
parameter lpm_width = 1;
parameter lpm_modulus = 0;
parameter lpm_direction = "UNUSED";
parameter lpm_avalue = "UNUSED";
parameter lpm_svalue = "UNUSED";
parameter lpm_pvalue = "UNUSED";
parameter lpm_port_updown = "PORT_CONNECTIVITY";
parameter lpm_hint = "UNUSED";
output [lpm_width-1:0] q;
```

```
output cout;
output [15:0] eq;
input  cin;
input  [lpm_width-1:0] data;
input  clock, clk_en, cnt_en, updown;
input  aset, aclr, aload;
input  sset, sclr, sload;
endmodule
```

### VHDL Component Declaration

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

```
component LPM_COUNTER
        generic (LPM_WIDTH : natural;
LPM_MODULUS : natural := 0;
LPM_DIRECTION : string := "UNUSED";
LPM_AVALUE : string := "UNUSED";
LPM_SVALUE : string := "UNUSED";
LPM_PORT_UPDOWN : string := "PORT_CONNECTIVITY";
LPM_PVALUE : string := "UNUSED";
LPM_TYPE: string := L_COUNTER;
LPM_HINT : string := "UNUSED");
port (DATA : in std_logic_vector(LPM_WIDTH-1 downto 0):= (OTHERS => '0');
CLOCK : in std_logic ;
CLK_EN : in std_logic := '1';
CNT_EN : in std_logic := '1';
UPDOWN : in std_logic := '1';
SLOAD : in std_logic := '0';
SSET : in std_logic := '0';
SCLR : in std_logic := '0';
ALOAD : in std_logic := '0';
ASET : in std_logic := '0';
ACLR : in std_logic := '0';
CIN : in std_logic := '1';
COUT : out std_logic := '0';
Q : out std_logic_vector(LPM_WIDTH-1 downto 0);
EQ : out std_logic_vector(15 downto 0));
end component;
```

### VHDL LIBRARY_USE Declaration

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

```
LIBRARY lpm;
USE lpm.lpm_components.all;
```

### Ports and Parameters

Table 15 lists the input ports, Table 16 lists the output ports, and Table 17 lists the LPM_COUNTER megafunction parameters.

**Table 15. LPM_COUNTER Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | No | Parallel data input to the counter. The size of the input port depends on the LPM_WIDTH parameter value. |
| clock | Yes | Positive-edge-triggered clock input. |

**Table 15. LPM_COUNTER Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| clk_en | No | Clock enable input to enable all synchronous activities. If omitted, the default value is 1. |
| cnt_en | No | Count enable input to disable the count when asserted low without affecting sload, sset, or sclr. If omitted, the default value is 1. |
| updown | No | Controls the direction of the count. When asserted high (1), the count direction is up, and when asserted low (0), the count direction is down. If the LPM_DIRECTION parameter is used, the updown port cannot be connected. If LPM_DIRECTION is not used, the updown port is optional. If omitted, the default value is up (1). |
| cin | No | Carry-in to the low-order bit. For up counters, the behavior of the cin input is identical to the behavior of the cnt_en input. If omitted, the default value is 1 (VCC). |
| aclr | No | Asynchronous clear input. If both aset and aclr are used and asserted, aclr overrides aset. If omitted, the default value is 0 (disabled). |
| aset | No | Asynchronous set input. Specifies the q[] outputs as all 1s, or to the value specified by the LPM_AVALUE parameter. If both the aset and aclr ports are used and asserted, the value of the aclr port overrides the value of the aset port. If omitted, the default value is 0, disabled. |
| aload | No | Asynchronous load input that asynchronously loads the counter with the value on the data input. When the aload port is used, the data[] port must be connected. If omitted, the default value is 0, disabled. |
| sclr | No | Synchronous clear input that clears the counter on the next active clock edge. If both the sset and sclr ports are used and asserted, the value of the sclr port overrides the value of the sset port. If omitted, the default value is 0, disabled. |
| sset | No | Synchronous set input that sets the counter on the next active clock edge. Specifies the value of the q outputs as all 1s, or to the value specified by the LPM_SVALUE parameter. If both the sset and sclr ports are used and asserted, the value of the sclr port overrides the value of the sset port. If omitted, the default value is 0 (disabled). |
| sload | No | Synchronous load input that loads the counter with data[] on the next active clock edge. When the sload port is used, the data[] port must be connected. If omitted, the default value is 0 (disabled). |

**Table 16. LPM_COUNTER Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| q[] | No | Data output from the counter. The size of the output port depends on the LPM_WIDTH parameter value. Either q[] or at least one of the eq[15..0] ports must be connected. |
| eq[15..0] | No | Counter decode output. The eq[15..0] port is not accessible using the MegaWizard Plug-In Manager as it is for AHDL use only<br>Either the q[] port or eq[] port must be connected. Up to c eq ports can be used (0 <= c <= 15). Only the 16 lowest count values are decoded. When the count value is c, the eqc output is asserted high (1). For example, when the count is 0, eq0 = 1, when the count is 1, eq1 = 1, and when the count is 15, eq 15 = 1. Decoded output for count values of 16 or greater require external decoding. The eq[15..0] outputs are asynchronous to the q[] output. |
| cout | No | Carry-out port of the counter's MSB bit. It can be used to connect to another counter to create a larger counter. |

**Table 17. LPM_COUNTER Megafunction Parameters (Part 1 of 2)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTH | Integer | Yes | Specifies the widths of the data[] and q[] ports, if they are used. |
| LPM_DIRECTION | String | No | Values are UP, DOWN, and UNUSED. If the LPM_DIRECTION parameter is used, the updown port cannot be connected. When the updown port is not connected, the LPM_DIRECTION parameter default value is UP. |
| LPM_MODULUS | Integer | No | The maximum count, plus one. Number of unique states in the counter's cycle. If the load value is larger than the LPM_MODULUS parameter, the behavior of the counter is not specified. |
| LPM_AVALUE | Integer/ String | No | Constant value that is loaded when aset is asserted high. If the value specified is larger than or equal to <modulus>, the behavior of the counter is an undefined (X) logic level, where <modulus> is LPM_MODULUS, if present, or 2 ^ LPM_WIDTH. Altera recommends that you specify this value as a decimal number for AHDL designs. |
| LPM_SVALUE | Integer/ String | No | Constant value that is loaded on the rising edge of the clock port when either the sset port or the sconst port is asserted high. The LPM_SVALUE parameter must be used when the sconst port is used. Altera recommends that you specify this value as a decimal number for AHDL designs. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the LPM_COUNTER megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| CARRY_CNT_EN | String | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the CARRY_CNT_EN parameter in VHDL design files. Values are SMART, ON, OFF, and UNUSED. Enables the LPM_COUNTER function to propagate the cnt_en signal through the carry chain. In some cases, the CARRY_CNT_EN parameter setting might have a slight impact on the speed, so you might want to turn it off. The default value is SMART, which provides the best trade-off between size and speed. |

**Table 17. LPM_COUNTER Megafunction Parameters (Part 2 of 2)**

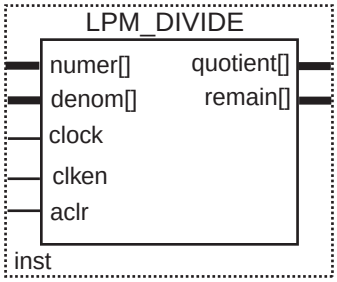
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LABWIDE_SCLR | String | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the LABWIDE_SCLR parameter in VHDL design files. Values are ON, OFF, or UNUSED. The default value is ON. Allows you to disable the use of the LAB-wide sclr feature found in obsoleted device families. Turning this option off increases the chances of fully using the partially filled LABs, and thus may allow higher logic density when SCLR does not apply to a complete LAB. This parameter is available for backward compatibility, and Altera recommends you not to use this parameter. |
| LPM_PORT_UPDOWN | String | No | Specifies the usage of the updown input port. If omitted the default value is PORT_CONNECTIVITY. When the port value is set to PORT_USED, the port is treated as used. When the port value is set to PORT_UNUSED, the port is treated as unused. When the port value is set to PORT_CONNECTIVITY, the port usage is determined by checking the port connectivity. |

# LPM_DIVIDE (Divider)

The LPM_DIVIDE megafunction implements a divider to divide a numerator input value by a denominator input value to produce a quotient and a remainder.

Figure 5 shows the ports for the LPM_DIVIDE megafunction.

**Figure 5. LPM_DIVIDE Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_DIVIDE megafunction offers the following features:

- Generates a divider that divides a numerator input value by a denominator input value to produce a quotient and a remainder
- Supports data width of 1–256 bits
- Supports signed and unsigned data representation format for both the numerator and denominator values
- Supports area or speed optimization
- Provides an option to specify a positive remainder output
- Supports pipelining configurable output latency
- Supports optional asynchronous clear and clock enable ports

## Resource Utilization and Performance

Table 18 provides resource utilization and performance information for the LPM_DIVIDE megafunction.

**Table 18. LPM_DIVIDE Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 10 | 1 | 131 | 0 | 70 | 133 |
| | 30 | 5 | 1017 | 0 | 635 | 71 |
| | 64 | 10 | 4345 | 0 | 2623 | 41 |

**Table 18. LPM_DIVIDE Resource Utilization and Performance** *(1)*

| | | | | | | |
|---|---|---|---|---|---|---|
| Stratix IV | 10 | 1 | 131 | 0 | 70 | 138 |
| | 30 | 5 | 1018 | 0 | 642 | 82 |
| | 64 | 10 | 4347 | 0 | 2634 | 48 |

**Note to Table 18:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_divide ( quotient, remain, numer, denom, clock, clken, aclr);
parameter lpm_type = "lpm_divide";
parameter lpm_widthn = 1;
parameter lpm_widthd = 1;
parameter lpm_nrepresentation = "UNSIGNED";
parameter lpm_drepresentation = "UNSIGNED";
parameter lpm_remainderpositive = "TRUE";
parameter lpm_pipeline = 0;
parameter lpm_hint = "UNUSED";
input  clock;
input  clken;
input  aclr;
input  [lpm_widthn-1:0] numer;
input  [lpm_widthd-1:0] denom;
output [lpm_widthn-1:0] quotient;
output [lpm_widthd-1:0] remain;
endmodule
```

### VHDL Component Declaration

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

```
component LPM_DIVIDE
        generic (LPM_WIDTHN : natural;
                 LPM_WIDTHD : natural;
LPM_NREPRESENTATION : string := "UNSIGNED";
LPM_DREPRESENTATION : string := "UNSIGNED";
LPM_PIPELINE : natural := 0;
LPM_TYPE : string := L_DIVIDE;
LPM_HINT : string := "UNUSED");
port (NUMER : in std_logic_vector(LPM_WIDTHN-1 downto 0);
DENOM : in std_logic_vector(LPM_WIDTHD-1 downto 0);
ACLR : in std_logic := '0';
CLOCK : in std_logic := '0';
CLKEN : in std_logic := '1';
QUOTIENT : out std_logic_vector(LPM_WIDTHN-1 downto 0);
REMAIN : out std_logic_vector(LPM_WIDTHD-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 lpm;
USE lpm.lpm_components.all;
```

## Ports and Parameters

Table 19 lists the input ports, Table 20 lists the output ports, and Table 21 lists the LPM_DIVIDE megafunction parameters.

**Table 19. LPM_DIVIDE Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| numer[] | Yes | Numerator data input. The size of the input port depends on the LPM_WIDTHN parameter value. |
| denom[] | Yes | Denominator data input. The size of the input port depends on the LPM_WIDTHD parameter value. |
| clock | No | Clock input for pipelined usage. For LPM_PIPELINE values other than 0 (default), the clock port must be enabled. |
| clken | No | Clock enable pipelined usage. When the clken port is asserted high, the division operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear port used at any time to reset the pipeline to all '0's asynchronously to the clock input. |

**Table 20. LPM_DIVIDE Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| quotient[] | Yes | Data output. The size of the output port depends on the LPM_WIDTHN parameter value. |
| remain[] | Yes | Data output. The size of the output port depends on the LPM_WIDTHD parameter value. |

**Table 21. LPM_DIVIDE Megafunction Parameters (Part 1 of 2)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTHN | Integer | Yes | Specifies the widths of the numer[] and quotient[] ports. Values are 1 to 64. |
| LPM_WIDTHD | Integer | Yes | Specifies the widths of the denom[] and remain[] ports. Values are 1 to 64. |
| LPM_NREPRESENTATION | String | No | Sign representation of the numerator input. Values are SIGNED and UNSIGNED. When this parameter is set to SIGNED, the divider interprets the numer[] input as signed two's complement. |
| LPM_DREPRESENTATION | String | No | Sign representation of the denominator input. Values are SIGNED and UNSIGNED. When this parameter is set to SIGNED, the divider interprets the denom[] input as signed two's complement. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files (**.vhd**). |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters, for example, LPM_REMAINDERPOSITIVE and MAXIMIZE_SPEED, in VHDL design files. The default value is UNUSED. |

**Table 21. LPM_DIVIDE Megafunction Parameters (Part 2 of 2)**

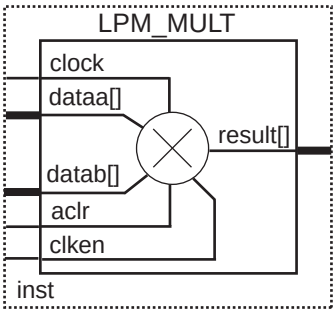
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_REMAINDERPOSITIVE | String | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the LPM_REMAINDERPOSITIVE parameter in VHDL design files. Values are TRUE or FALSE. If this parameter is set to TRUE, then the value of the remain[] port must be greater than or equal to zero. If this parameter is set to TRUE, then the value of the remain[] port is either zero, or the value is the same sign, either positive or negative, as the value of the numer port. In order to reduce area and improve speed, Altera recommends setting this parameter to TRUE in operations where the remainder must be positive or where the remainder is unimportant. |
| MAXIMIZE_SPEED | Integer | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the MAXIMIZE_SPEED parameter in VHDL design files. Values are [0..9]. If used, the Quartus II software attempts to optimize a specific instance of the LPM_DIVIDE function for speed rather than routability, and overrides the setting of the Optimization Technique logic option. If MAXIMIZE_SPEED is unused, the value of the Optimization Technique option is used instead. If the value of MAXIMIZE_SPEED is 6 or higher, the Compiler optimizes the LPM_DIVIDE megafunctions for higher speed by using carry chains; if the value is 5 or less, the compiler implements the design without carry chains. |
| LPM_PIPELINE | Integer | No | Specifies the number of clock cycles of latency associated with the quotient[] and remain[] outputs. A value of zero (0) indicates that no latency exists, and that a purely combinational function is instantiated. If omitted, the default value is 0 (non-pipelined). You cannot specify a value for the LPM_PIPELINE parameter that is higher than LPM_WIDTHN. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the LPM_DIVIDE megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| SKIP_BITS | Integer | No | Allows for more efficient fractional bit division to optimize logic on the leading bits by providing the number of leading GND to the LPM_DIVIDE megafunction. Specify the number of leading GND on the quotient output to this parameter. |

# LPM_MULT (Multiplier)

The LPM_MULT megafunction implements a multiplier to multiply two input data values to produce a product as an output.

Figure 6 shows the ports for the LPM_MULT megafunction.

**Figure 6. LPM_MULT Ports Shown in the MegaWizard Plug-In Manager**




## Features

The LPM_MULT megafunction offers the following features:

- Generates a multiplier that multiplies two input data values
- Supports data width of 1–256 bits
- Supports signed and unsigned data representation format
- Supports area or speed optimization
- Supports pipelining with configurable output latency
- Provides an option for implementation in dedicated digital signal processing (DSP) block circuitry or logic elements (LEs)
- Supports optional asynchronous clear and clock enable input ports

Refer to the following megafunctions in this user guide for other multiplier implementations:

- Multiplier-Adder Megafunction (ALTMULT_ADD)
- Memory-based Constant Coefficient Multiplier (ALTMEMMULT)
- Multiplier-Accumulator Megafunction (ALTMULT_ACCUM)

## Resource Utilization and Performance

The LPM_MULT megafunction can be implemented using either logic resources or dedicated multiplier circuitry in Altera devices. Typically, the LPM_MULT megafunction is translated to the dedicated multiplier circuitry when it is available because it provides better performance and resource utilization. If all of the input data widths are smaller than or equal to nine bits, the function uses the 9 × 9 multiplier configuration in the dedicated multiplier. Otherwise, 18 × 18 multipliers are used to process data with widths between 10 bits and 18 bits.

For information about the architecture of the DSP blocks and embedded multipliers, and for detailed information about the hardware conversion process, refer to the DSP block and embedded multiplier chapters in the Stratix device series, Stratix II, Stratix III, and Cyclone II handbooks on the Literature and Technical Documentation page.

Table 22 provides resource utilization and performance information for the LPM_MULT megafunction.

**Table 22. LPM_MULT Resource Utilization and Performance** *(1)*

| Device family | Input data width | Output latency | Logic Usage | | | | $f_{MAX}$ (MHz) *(2)* |
|---|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | 18-bit DSP | |
| Stratix III | 8 × 8 | 0 | 0 | 0 | 0 | 1 | N/A |
| | 16 × 16 | 0 | 0 | 0 | 0 | 2 | |
| | 32 × 32 | 0 | 0 | 0 | 0 | 4 | |
| | 16 × 16 | 3 | 0 | 0 | 0 | 2 | 645 |
| | 32 × 32 | 3 | 0 | 0 | 0 | 4 | 454 |
| | 64 × 64 | 3 | 92 | 128 | 82 | 16 | 191 |

**Notes to Table 22:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module lpm_mult ( result, dataa, datab, sum, clock, clken, aclr )
parameter lpm_type = "lpm_mult";
parameter lpm_widtha = 1;
parameter lpm_widthb = 1;
parameter lpm_widths = 1;
parameter lpm_widthp = 1;
parameter lpm_representation  = "UNSIGNED";
parameter lpm_pipeline  = 0;
parameter lpm_hint = "UNUSED";
input  clock;
input  clken;
input  aclr;
input  [lpm_widtha-1:0] dataa;
input  [lpm_widthb-1:0] datab;
input  [lpm_widths-1:0] sum;
output [lpm_widthp-1:0] result;
endmodule
```

### VHDL Component Declaration

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

```
component LPM_MULT
        generic ( LPM_WIDTHA : natural;
                  LPM_WIDTHB : natural;
                  LPM_WIDTHS : natural := 1;
                  LPM_WIDTHP : natural;
LPM_REPRESENTATION : string := "UNSIGNED";
LPM_PIPELINE : natural := 0;
LPM_TYPE: string := L_MULT;
LPM_HINT : string := "UNUSED");
port ( DATAA : in std_logic_vector(LPM_WIDTHA-1 downto 0);
DATAB : in std_logic_vector(LPM_WIDTHB-1 downto 0);
ACLR : in std_logic := '0';
CLOCK : in std_logic := '0';
CLKEN : in std_logic := '1';
SUM : in std_logic_vector(LPM_WIDTHS-1 downto 0) := (OTHERS => '0');
RESULT : out std_logic_vector(LPM_WIDTHP-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 lpm;
USE lpm.lpm_components.all;
```

## Ports and Parameters

Table 23 lists the input ports, Table 24 lists the output ports, and Table 25 lists the LPM_MULT megafunction parameters.

**Table 23. LPM_MULT Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| dataa[] | Yes | Data input. The size of the input port depends on the LPM_WIDTHA parameter value. |
| datab[] | Yes | Data input. The size of the input port depends on the LPM_WIDTHB parameter value. |
| clock | No | Clock input for pipelined usage. For LPM_PIPELINE values other than 0 (default), the clock port must be enabled. |
| clken | No | Clock enable for pipelined usage. When the clken port is asserted high, the adder/subtractor operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear port used at any time to reset the pipeline to all 0s, asynchonously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value. |

**Table 24. LPM_MULT Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Data output. The size of the output port depends on the LPM_WIDTHP parameter value. If LPM_WIDTHP < max (LPM_WIDTHA + LPM_WIDTHB, LPM_WIDTHS) or (LPM_WIDTHA + LPM_WIDTHS), only the LPM_WIDTHP MSBs are present. |

**Table 25. LPM_MULT Megafunction Parameters (Part 1 of 3)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| LPM_WIDTHA | Integer | Yes | Specifies the width of the dataa[] port. |
| LPM_WIDTHB | Integer | Yes | Specifies the width of the datab[] port. |
| LPM_WIDTHP | Integer | Yes | Specifies the width of the result[] port. |
| LPM_REPRESENTATION | String | No | Specifies the type of multiplication performed. Values are SIGNED and UNSIGNED. If omitted, the default value is UNSIGNED. When this parameter value is set to SIGNED, the multiplier interprets the data input as signed two's complement. |
| LPM_PIPELINE | String | No | Specifies the number of latency clock cycles associated with the result[] output. A value of zero (0) indicates that no latency exists, and that a purely combinational function will be instantiated. For Stratix and Stratix GX devices, if the design uses DSP blocks, you can increase the performance of the design when the value of the LPM_PIPELINE parameter is 3 or less. |
| INPUT_A_IS_CONSTANT | String | No | You must use the LPM_HINT parameter to specify the INPUT_A_IS_CONSTANT parameter in VHDL design files. Values are YES, NO, and UNUSED. If dataa [] is connected to a constant value, setting INPUT_A_IS_CONSTANT to YES optimizes the multiplier for resource usage and speed. If omitted, the default value is NO. |

**Table 25. LPM_MULT Megafunction Parameters (Part 2 of 3)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_B_IS_CONSTANT | String | No | You must use the LPM_HINT parameter to specify the INPUT_B_IS_CONSTANT parameter in VHDL design files. Values are YES, NO, and UNUSED. If datab[] is connected to a constant value, setting INPUT_B_IS_CONSTANT to YES optimizes the multiplier for resource usage and speed. The default value is NO. |
| USE_EAB | String | No | Specifies RAM block usage. Values are ON and OFF. Setting the USE_EAB parameter to ON allows the Quartus II software to use embedded array blocks (EABs) to implement 4 x 4 or (8 x const value) building blocks in some obsoleted devices. Altera recommends that you set USE_EAB to ON only when LCELLS are in short supply. This parameter is not available for simulation with other EDA simulators.<br>If you wish to use this parameter when you instantiate the function in a Block Design File (**.bdf**), you must specify it by entering the parameter name and value manually with the Parameters tab in the **Symbol Properties** dialog box or in the **Block Properties** dialog box. You can also use this parameter name in a Text Design File (**.tdf**) or a Verilog Design File (**.v**). You must use the LPM_HINT parameter to specify the USE_EAB parameter in VHDL design files. |
| MAXIMIZE_SPEED | Integer | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the MAXIMIZE_SPEED parameter in VHDL design files. You can specify a value between 0 and 10. If used, the Quartus II software attempts to optimize a specific instance of the LPM_MULT function for speed rather than area, and overrides the setting of the Optimization Technique logic option. If MAXIMIZE_SPEED is unused, the value of the Optimization Technique option is used instead. For a SIGNED multiplier with no inputs being a constant, if the setting for MAXIMIZE_SPEED is 9-10, the Compiler optimizes the LPM_MULT megafunction for larger area, these settings are for backward compatibility only; if the setting is between 6-8, the Compiler optimizes for larger area and higher speed; if the setting is between 1-5, the Compiler optimizes for smaller area and high speed. If the setting is 0, the smallest and, generally, slowest design results. For designs with LPM_WIDTHB parameters that are non-power-of-2, the default setting is 1-5. For designs with LPM_WIDTHB parameters that are a power-of-2, the default value is 6-8. For an UNSIGNED multiplier with no inputs being a constant, if the setting for MAXIMIZE_SPEED is 6 or higher, the Compiler optimizes for larger area and higher speed; if the setting is 0 up to 5, which is the default value, the Compiler optimizes for smaller area.<br>Note that specifying a value for MAXIMIZE_SPEED has an effect only if LPM_REPRESENTATION is set to SIGNED. |

**Table 25. LPM_MULT Megafunction Parameters (Part 3 of 3)**

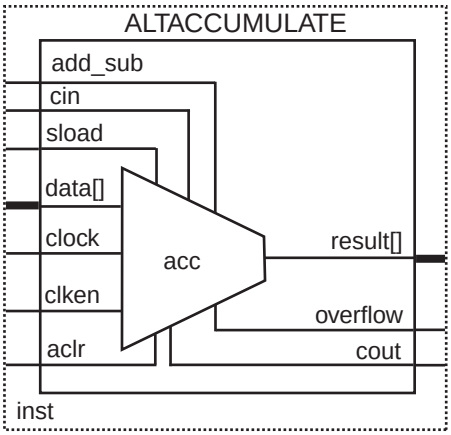
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| DEDICATED_MULTIPLIER_CIRCUITRY | String | No | Specifies whether to use the default dedicated multiplier circuitry implementation. Values are AUTO, YES, NO, and FIRM. If omitted, the default value is AUTO. For Stratix and Stratix GX devices, the value of AUTO specifies that the Quartus II software determines whether to use the dedicated multiplier circuitry based on the multiplier width. If a device does not have dedicated multiplier circuitry, the DEDICATED_MULTIPLIER_CIRCUITRY parameter has no effect and the value defaults to NO. |
| DSP_BLOCK_BALANCING | String | No | Specifies whether to use a dedicated multiplier circuitry implementation. Values are UNUSED, AUTO, DSP BLOCKS, and LOGIC ELEMENTS. If omitted, the default value is UNUSED. This parameter is available for all Altera devices except Cyclone, HardCopy, MAX II, MAX 3000, and MAX 7000 devices. |
| LOGIC_ELEMENTS | String | No | Specifies whether to use a logic element implementation based on the selected device family. When implemented in LEs, the LPM_MULT megafunction uses a variation on the Booth algorithm for all device families. Values are OFF, SIMPLE 18-BIT MULTIPLIERS, SIMPLE MULTIPLIERS, WIDTH 18-BIT MULTIPLIERS, and LOGIC ELEMENTS. |
| DEDICATED_MULTIPLIER_MIN_INPUT_WIDTH_FOR_AUTO | Integer | No | Altera-specific parameter. You must use the LPM_HINT parameter to specify the DEDICATED_MULTIPLIER_MIN_OUTPUT_WIDTH_FOR_AUTO parameter in VHDL design files. If the DEDICATED_MULTIPLIER_CIRCUITRY parameter setting is AUTO, this parameter specifies the minimum value of the sum of the LPM_WIDTHA and LPM_WIDTHB parameters in order for the multiplier to be built using dedicated circuitry. |
| INPUT_A_FIXED_VALUE | String | No | Specifies the value for the dataa[] port. This parameter is used when the INPUT_A_IS_CONSTANT parameter is set to FIXED. For example, to pass a four bit value of 3 to the dataa[] port, the INPUT_A_FIXED_VALUE parameter must be set to B0011. |
| INPUT_B_FIXED_VALUE | String | No | Specifies the value for the datab[] port. This parameter is used when the INPUT_B_IS_CONSTANT parameter is set to FIXED. For example, to pass a four bit value of 3 to the datab[] port, the INPUT_B_FIXED_VALUE parameter must be set to B0011. |

# ALTACCUMULATE (Accumulator)

The ALTACCUMULATE megafunction implements an accumulator that adds its input data to the registered result of its previous operation, allowing the accumulator to store the sum of a serial data stream.

Figure 7 shows the ports for the ALTACCUMULATE megafunction.

**Figure 7. ALTACCUMULATE Ports Shown in the MegaWizard Plug-In Manager**




## Features

The ALTACCUMULATE megafunction offers the following features:

- Generates an accumulator to store the sum of a serial data stream
- Supports data width of 1– 64 bits
- Supports signed and unsigned data representation format
- Support pipelining with configurable output latency
- Provides an option to dynamically switch between add and subtract operations
- Supports optional asynchronous clear, clock enable, and carry-in input ports
- Supports optional overflow and carry-out output ports

## Resource Utilization and Performance

Table 26 provides resource utilization and performance information for the ALTACCUMULATE megafunction.

**Table 26. ALTACCUMULATE Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 8 | 0 | 24 | 21 | 18 | 609 |
| | 8 | 7 | 36 | 87 | 51 | 661 |
| | 64 | 0 | 74 | 132 | 77 | 330 |
| | 64 | 7 | 253 | 362 | 226 | 429 |
| Stratix IV | 8 | 0 | 24 | 21 | 18 | 665 |
| | 8 | 7 | 36 | 87 | 51 | 730 |
| | 64 | 0 | 74 | 132 | 84 | 415 |
| | 64 | 7 | 253 | 362 | 230 | 480 |

**Notes to Table 26:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: 9-Bit Multiplier Accumulator

Accumulators are common basic building blocks that are useful in DSP functions. This section provides a design example that incorporates the LPM_MULT and ALTACCUMULATE megafunctions to generate a basic multiplier accumulator. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

## Design Files

The following design files can be found in **altaccumulate_DesignExample.zip**:

- **mult_accum.qar** (archived Quartus II design files)
- **altaccumulate_ex_msim** (ModelSim-Altera files)

## Understanding the Simulation Results

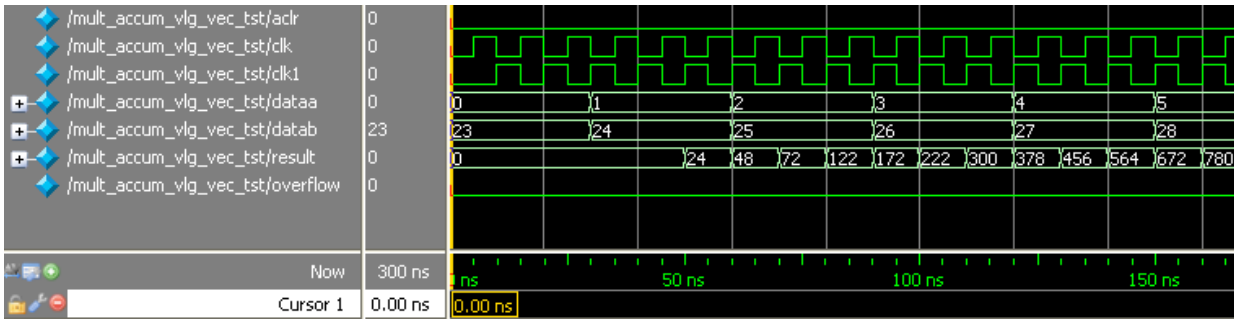
The following settings are observed in this example:

- The `dataa[]` and `datab[]` input widths are both set to 9 bits.
- The `result[]` output port width is set to 36 bits.
- The lpm_mult module and altaccumulate module operate on separate clocks.
- The asynchronous clear (`aclr`) input port is enabled.
- The `overflow` output port is enabled.

- The latency is set to one clock cycle for the multiplier and one clock cycle for the accumulator, resulting in a total output latency of two clock cycles. Hence, the result is seen on the `result[]` port two clock cycles after the input data is available

Figure 8 shows the expected simulation results in the ModelSim-Altera software.

**Figure 8. ALTACCUMULATE Simulation Results**




## Specifications

This section describes the prototypes, component declarations, ports, and parameters of the ALTACCUMULATE megafunction. These ports and parameters are available to customize the ALTACCUMULATE megafunction according to your application. The parameter details in this section are only relevant if you bypass the MegaWizard Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in your design. The details of these parameters are hidden from you in the MegaWizard Plug-In Manager interface.

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altaccumulate
#( parameter carry_chain = "MANUAL",
parameter carry_chain_length = 32,
parameter intended_device_family = "unused",
parameter extra_latency = 0,
parameter lpm_representation = "UNSIGNED",
parameter right_shift_distance = 0,
parameter use_wys = "ON",
parameter width_in = 1,
parameter width_out = 1,
parameter lpm_type = "altaccumulate",
parameter lpm_hint = "unused")
( input wire aclr,
input wire add_sub,
input wire cin,
input wire clken,
input wire clock,
output wire cout,
input wire [width_in-1:0] data,
output wire overflow,
output wire [width_out-1:0] result,
input wire sign_data,
input wire sload);
endmodule
```

## 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 altaccumulate
generic (
carry_chain:string := "MANUAL";
carry_chain_length:natural := 32;
intended_device_family:string := "unused";
extra_latency:natural := 0;
lpm_representation:string := "UNSIGNED";
right_shift_distance:natural := 0;
use_wys:string := "ON";
width_in:natural;
width_out:natural;
lpm_hint:string := "UNUSED";
lpm_type:string := "altaccumulate");
port(
aclr:in std_logic := '0';
add_sub:in std_logic := '1';
cin:in std_logic := '0';
clken:in std_logic := '1';
clock:in std_logic;
cout:out std_logic;
data:in std_logic_vector(width_in-1 downto 0);
overflow:out std_logic;
result:out std_logic_vector(width_out-1 downto 0);
sign_data:in std_logic := '0';
sload: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 27 lists the input ports, Table 28 lists the output ports, and Table 29 lists the ALTACCUMULATE megafunction parameters.

**Table 27. ALTACCUMULATE Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | Yes | Data input to the accumulator. The size of the input port depends on the WIDTH_IN parameter value. |
| clock | Yes | Clock input for pipelined operation. |
| cin | No | Carry-in to the low-order bit for add operations. Borrow-in to the low-order bit for subtract operations. If omitted, the default value is 0 for add operations and 1 for subtract operations. |
| clken | No | Clock enable for the clock port. When the clk_en is asserted high, the operation takes place. When the signal is low, no operation occurs. |
| aclr | No | Asynchronous clear input. When the aclr port is asserted high, the function is asynchronously cleared. |

**Table 27. ALTACCUMULATE Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| add_sub | No | Controls the operation of the accumulator. If signal is high, the accumulator performs an add function. If the signal is low, the accumulator performs a subtract function. If this parameter is not used, the accumulator performs add functions only. |
| sign_data | No | Specifies the numerical representation of the data[] port. If the sign_data port is high, the accumulator treats the data[] port as signed two's complement. If the sign_data port is low, the accumulator treats the data[] port as unsigned. |
| sload | No | Synchronous load input that loads the accumulator with the value of the data[] port on the next active clock edge. |

**Table 28. ALTACCUMULATE Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Accumulator output port. The size of the output port depends on the WIDTH_OUT parameter value. |
| cout | No | Carry-out (borrow-in) of the MSB that detects overflow in unsigned operations. The cout port operates in the same manner for signed and unsigned operations. |
| overflow | No | Overflow port for the accumulator that asserts when the function results in a value larger than the maximum value supported by the result[] port. |

**Table 29. ALTACCUMULATE Megafunction Parameters**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH_IN | Integer | Yes | Specifies the width of the data[] port. |
| WIDTH_OUT | Integer | Yes | Specifies the width of the result[] port. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTACCUMULATE megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| LPM_HINT | String | No | Allows you to assign Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| EXTRA_LATENCY | Integer | No | Specifies the number of clock cycles of latency. |

# ALTECC (Error Correction Code: Encoder/Decoder)

The error correction code (ECC) is a error detection and correction method in digital data transmission. Its primary purpose is to detect corrupted data that occurs at the receiver side during data transmission. This error correction method is best suited for situations where errors occur at random rather than in bursts.

The ECC detects errors through the process of data encoding and decoding. For example, when the ECC is applied in a transmission application, data read from the source are encoded before being sent to the receiver. The output (code word) from the encoder consists of the raw data appended with the number of parity bits. The exact number of parity bits appended depends on the number of bits in the input data. The generated code word is then transmitted to the destination.

The receiver receives the code word and decodes it. Information obtained by the decoder determines whether an error is detected. The decoder detects single-bit and double-bit errors, but can only fix single-bit errors in the corrupted data. This type of ECC is called a single error correction double error detection (SECDED).

Altera provides two megafunctions, the ALTECC_ENCODER and ALTECC_DECODER, to implement the ECC functionality. The data input to the ALTECC_ENCODER megafunction is encoded to generate a code word that is a combination of the data input and the generated parity bits. The generated code word is transmitted to the ALTECC_DECODER megafunction for decoding just before reaching its destination block. The ALTECC_DECODER megafunction generates a syndrome vector to determine if there is any error in the received code word. It fixes the data only if the single-bit error is from the data bits. No signal is flagged if the single-bit error is from the parity bits. The megafunction also has flag signals to show the status of the data received and the action taken by the ALTECC_DECODER megafunction, if any.

Figure 9 and Figure 10 show the ports for the ALTECC megafunction.

**Figure 9. ALTECC_ENCODER Ports Shown in the MegaWizard Plug-In Manager**

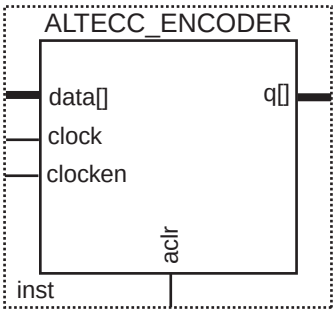
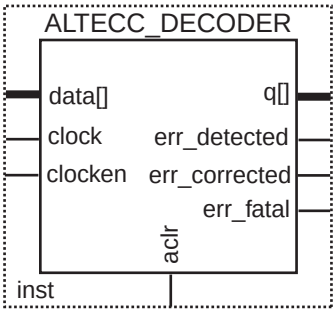



**Figure 10. ALTECC_DECODER Ports Shown in the MegaWizard Plug-In Manager**




# Features

This section describes the features of the two megafunctions, the ALTECC_ENCODER and ALTECC_DECODER that implements the ECC functionality.

## ALTECC_ENCODER

The ALTECC_ENCODER megafunction offers the following features:

- Performs data encoding using the Hamming Coding scheme
- Supports data width of 2–64 bits
- Supports signed and unsigned data representation format
- Support pipelining with output latency of either one or two clock cycles
- Supports optional asynchronous clear and clock enable ports

The ALTECC_ENCODER megafunction takes in and encodes the data using the Hamming Coding scheme. The Hamming Coding scheme derives the parity bits and appends them to the original data to produce the output code word. The number of parity bits appended depends on the width of the data.

Table 30 lists the number of parity bits appended for different ranges of data widths. The **Total Bits** column represents the total number of input data bits and appended parity bits.

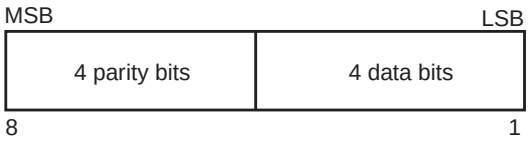
**Table 30. Number of Parity Bits and Code Word According to Data Width**

| Data Width | Number of Parity Bits | Total Bits (Code Word) |
|---|---|---|
| 2-4 | 3+1 | 6-8 |
| 5-11 | 4+1 | 10-16 |
| 12-26 | 5+1 | 18-32 |
| 27-57 | 6+1 | 34-64 |
| 58-64 | 7+1 | 66-72 |

The parity bit derivation uses an even-parity checking. The additional 1 bit (shown in Table 30 as +1) is appended to the parity bits as the most significant bit (MSB) of the code word. This ensures that the code word has an even number of 1's. For example, if the data width is 4 bits, 4 parity bits are appended to the data to become a code word with a total of 8 bits. If 7 bits from the LSB of the 8-bit code word have an odd number of 1's, the 8th bit (MSB) of the code word is 1 making the total number of 1's in the code word even.

Figure 11 shows the generated code word and the arrangement of the parity bits and data bits in an 8-bit data input.

**Figure 11. Parity Bits and Data Bits Arrangement in an 8-Bit Generated Code Word**




The ALTECC_ENCODER megafunction accepts only input widths of 2 to 64 bits at one time. Input widths of 12 bits, 29 bits, and 64 bits, which are ideally suited to Altera devices, generate outputs of 18 bits, 36 bits, and 72 bits respectively. The bit-selection limitation is controlled by the MegaWizard Plug-In Manager.

## ALTECC_DECODER

The ALTECC_DECODER megafunction offers the following features:

- Performs data decoding using the Single Error Correction Double Error Detection (SECDED) method
- Supports data width of 6-72 (excluding 9, 17, 33, and 65) bits
- Supports pipelining with output latency of either one or two clock cycles
- Uses flag signals to reflect the status of received data
- Supports optional asynchronous clear and clock enable input ports

The ALTECC_DECODER megafunction accepts a wide range of code word widths, from 6 bits to 72 bits except 9-, 17-, 33-, and 65-bit widths. To calculate the width of the output data from the decoder, refer to Table 30 on page 38.

The ALTECC_DECODER megafunction decodes the input data (code word) by extracting the parity bits and data bits from the code word. The parity bits and data bits are recalculated based on the Hamming Coding scheme to generate a syndrome code. The generated syndrome code provides the status of the data received. The ECC detects single-bit and double-bit errors, but only single-bit errors are corrected.

The ALTECC_DECODER megafunction has flag signals (`err_detected`, `err_corrected`, and `err_fatal`) that reflect the status of the data received. From these signals, you can determine whether the data you receive is corrupted, or if a single-bit error has been corrected.

Table 31 lists the syndrome codes and their respective flag signals.

**Table 31. Description of General Syndrome Codes and Their Respective Flag Signals**

| Syndrome Code | Description | Flag Signals |
|---|---|---|
| All-zero | No error occurs | `err_detected = 0`<br>`err_corrected = 0`<br>`err_fatal = 0` |
| Non-zero and the MSB is 1 | A single-bit error is detected, and the corrupted bit is flipped. The value of the syndrome vector (except the MSB) indicates the corrupted bit position within the code word *(1)* | `err_detected = 1`<br>`err_corrected = 1`<br>`err_fatal = 0` |
| Non-zero and the MSB is 0 | A double-bit error is detected. No correction is made and the output data is incorrect | `err_detected = 1`<br>`err_corrected = 0`<br>`err_fatal = 1` |

**Note to Table 31:**

(1) Even if the generated syndrome code indicates a single-bit error, the `err_detected` and `err_corrected` signals are asserted only if the corrupted bit is from the data bits and not from the parity bits.

If the syndrome bits show all 0's, no single-bit or double-bit error occurred. However, there is a slight possibility that the generated syndrome code shows all 0's even if errors occur. Therefore, the ECC is not used to detect errors of more than 2 bits.

## Resource Utilization and Performance

Table 32 and Table 33 provide resource utilization and performance information for the ALTECC megafunction.

**Table 32. ALTECC Resource Utilization and Performance for Stratix III Devices (1)**

| Configuration | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| ALTECC_ENCODER | 12 | 0 | 8 | 0 | 4 | 1161 |
| | 29 | 0 | 21 | 0 | 13 | 1076 |
| | 32 | 0 | 19 | 0 | 12 | 979 |
| | 64 | 0 | 40 | 0 | 27 | 758 |
| | 12 | 2 | 8 | 30 | 19 | 1188 |
| | 29 | 2 | 20 | 65 | 36 | 1021 |
| | 32 | 2 | 19 | 71 | 40 | 1013 |
| | 64 | 2 | 39 | 136 | 79 | 926 |
| ALTECC_DECODER | 12 | 0 | 8 | 0 | 4 | 1161 |
| | 29 | 0 | 21 | 0 | 13 | 1076 |
| | 32 | 0 | 19 | 0 | 12 | 979 |
| | 64 | 0 | 40 | 0 | 27 | 758 |
| | 12 | 2 | 8 | 30 | 19 | 1188 |
| | 29 | 2 | 20 | 65 | 36 | 1021 |
| | 32 | 2 | 19 | 71 | 40 | 1013 |
| | 64 | 2 | 39 | 136 | 79 | 926 |

**Table 33. ALTECC Resource Utilization and Performance for Stratix IV Devices (1)**

| Configuration | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| ALTECC_ENCODER | 12 | 0 | 8 | 0 | 4 | 1128 |
| | 29 | 0 | 21 | 0 | 13 | 1072 |
| | 32 | 0 | 19 | 0 | 12 | 1054 |
| | 64 | 0 | 40 | 0 | 27 | 901 |
| | 12 | 2 | 8 | 30 | 19 | 1104 |
| | 29 | 2 | 20 | 65 | 38 | 1082 |
| | 32 | 2 | 19 | 71 | 42 | 1061 |
| | 64 | 2 | 39 | 136 | 78 | 905 |

**Table 33. ALTECC Resource Utilization and Performance for Stratix IV Devices (1)**

| | | | | | | |
|---|---|---|---|---|---|---|
| ALTECC_DECODER | 12 | 0 | 8 | 0 | 4 | 1128 |
| | 29 | 0 | 21 | 0 | 13 | 1072 |
| | 32 | 0 | 19 | 0 | 12 | 1054 |
| | 64 | 0 | 40 | 0 | 27 | 901 |
| | 12 | 2 | 8 | 30 | 19 | 1104 |
| | 29 | 2 | 20 | 65 | 38 | 1082 |
| | 32 | 2 | 19 | 71 | 42 | 1061 |
| | 64 | 2 | 39 | 136 | 78 | 905 |

**Notes to Table 32 and Table 33:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

# Design Example 1: ALTECC_ENCODER

This design example uses the ECC encoder to encode an 8-bit wide input data to generate 13 bits of output code word. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

## Design Files

The following design files can be found in **altecc_DesignExample1.zip**:

- **altecc_encode.qar** (archived Quartus II design files)
- **altecc_encode_ex_msim** (ModelSim-Altera files)

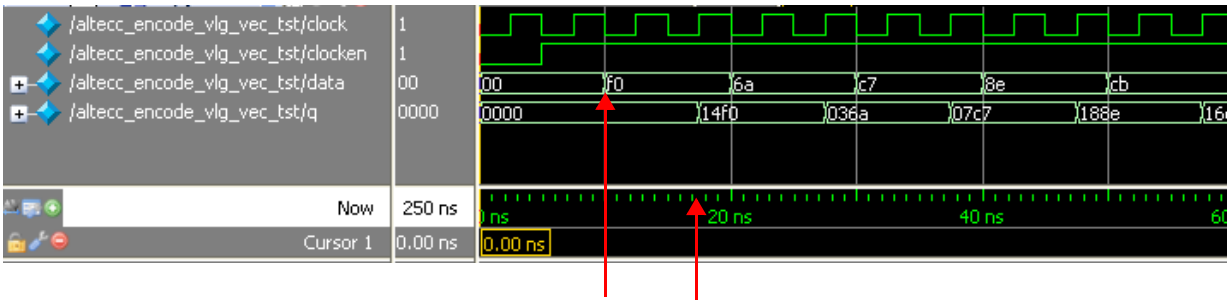
# Understanding the Simulation Results

The following settings are observed in this example:

- The `data[]` input width is set to 8 bits
- The output port, `q[]` has a width of 13 bits
- The clock enable (`clocken`) signal is enabled
- Pipelining is enabled, with an output latency of 2 clock cycles. Hence, the result is seen on the `q[]` port two clock cycles after the input data is available

Figure 12 shows the expected simulation results in the ModelSim-Altera software.

**Figure 12. Design Example 1: Simulation Waveform for the ECC Encoder**




The following sequence corresponds with the numbered items in Figure 12:

1. Data F0 is fed to the ECC encoder. As pipelining is enabled to have an output latency of 2 clock cycles, the result of the encoding operation appears at the output port `q[]` 2 clock cycles later. The 8-bit input data (F0) is encoded to generate a 13-bit output code word (14F0). The input data is appended with 5 parity bits.

   The ECC encoder encodes the data based on the Hamming Code scheme. The following steps describe the Hamming Code algorithm and explain how the ECC encoder encodes input data F0 to generate the output code word of 14F0:

   a. In a 13-bit code word, there are 13 locations (bit positions), and each location holds 1 bit. There are 8 bits of original data, and the appended 5 parity bits. The locations (bit positions) for the bits must be defined – bit positions that are powers of 2 are used as parity bits (positions 1, 2, 4, 8 …). Table 34 lists the bit positions, and the position of the parity bits of a 13-bit code word. P5* is the extra parity bit added. The prefix P denotes parity.

**Table 34. Design Example 1: Position of Parity Bits for a 13-Bit Code Word**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits | P1 | P2 | | P3 | | | | P4 | | | | | P5* |

   b. All other bit positions are for the data to be encoded. The least significant bit (LSB) of the data bit fills the lowest bit position. In this case, starting from the LSB of the data, F0 (1111 0000 in binary) fills the empty bit positions, starting from position (3), as shown in Table 35. The prefixes P and D denote parity and data, respectively.
   For the standard Hamming Code algorithm, the MSB of the data bit fills the lowest bit position, unlike the Altera ECC megafunction, which fills up the lowest bit position starting with the LSB. This bit order reduces the complexity of the circuit design.

**Table 35. Design Example 1: Filling of Data Bits (1111 0000) for a 13-Bit Code Word**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits and data bits | P1 | P2 | D1<br>0 | P3 | D2<br>0 | D3<br>0 | D4<br>0 | P4 | D5<br>1 | D6<br>1 | D7<br>1 | D8<br>1 | P5* |

c. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.
The following section list the sequence of bits that each parity bit checks:

Parity bit 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit… (1, 3, 5, 7, 9, 11)
Parity bit 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits… (2, 3, 6, 7, 10, 11)
Parity bit 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits… (4, 5, 6, 7,1 2)
Parity bit 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits… (8, 9, 10, 11, 12)

**Table 36. Design Example 1: Calculation of Parity Bits**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits and data bits | P1 | P2 | D1<br>0 | P3 | D2<br>0 | D3<br>0 | D4<br>0 | P4 | D5<br>1 | D6<br>1 | D7<br>1 | D8<br>1 | P5* |
| Calculate P1 | 0 | | 0 | | 0 | | 0 | | 1 | | 1 | | |
| Calculate P2 | | 0 | 0 | | | 0 | 0 | | | 1 | 1 | | |
| Calculate P3 | | | | 1 | 0 | 0 | 0 | | | | | 1 | |
| Calculate P4 | | | | | | | | 0 | 1 | 1 | 1 | 1 | |
| Calculate P5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

d. Calculate the additional parity bits using an even parity checking on all the bits, including the calculated parity bits. The additional parity bit P5* is calculated with an even parity checking on all the bits from position (1) to position (12), as shown in Table 36.

e. The generated code word is rearranged so that the data is at the LSB and the parity bits are at the MSB. In this example, the generated code word is rearranged as shown in Figure 13.

**Figure 13. Design Example 1: Complete Generated Code Word after Rearrangement**

MSB LSB

| P5* | P4 | P3 | P2 | P1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

2. The encoded input data for F0 is 14F0 (1 0100 1111 0000 in binary), as seen on the output port, `q[]`, at 17.5 ns.

## Design Example 2: ALTECC_DECODER

This design example uses the ECC decoder to decode input code words of 13-bit widths to generate 8 bits of output data. An asynchronous clear signal is also used to illustrate how the signal affects the registered ports. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **altecc_DesignExample2.zip**:

- **altecc_decode.qar** (archived Quartus II design files)
- **altecc_decode_ex_msim** (ModelSim-Altera files)

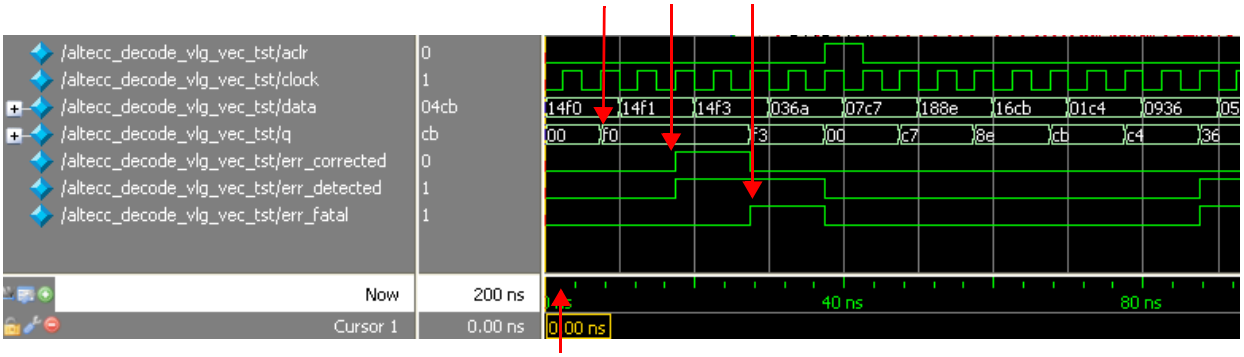
## Understanding the Simulation Results

The following settings are observed in this example:

- The `data[]` input width is set to 13 bits
- The output port, `q[]` has a width of 8 bits
- The asynchronous clear (`aclr`) signal is enabled
- Pipelining is enabled, with an output latency of 2 clock cycles. Hence, the result is seen on the `q[]` port two clock cycles after the input data is available

Figure 14 shows the expected simulation results in the ModelSim-Altera software.

**Figure 14. Design Example 1: Simulation Waveform for the ECC Decoder**




The following sequence corresponds with the numbered items in Figure 14:

1. The decoder decodes the code word 14F0 at the first rising edge of the clock at 2.5 ns. In this case, the input code word is not corrupted. The 13-bit input code word 14F0 (1 0100 1111 0000 in binary) is decoded to generate an 8-bit output data of F0. Table 37 lists the arrangement of parity bits and data bits in the code word 14F0. The prefixes P and D denote parity and data respectively.

**Table 37. Design Example 2: Arrangement of Parity Bits and Data Bits in Code Word 14F0**

| MSB | | | | | | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Table 37. Design Example 2: Arrangement of Parity Bits and Data Bits in Code Word 14F0**

| P5* | P4 | P3 | P2 | P1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

The ECC decoder decodes the code word based on the Hamming Code scheme. The following steps describe the Hamming Code algorithm and explain how the ECC decoder decodes input code word 14F0 to generate output data F0:

a. All bits have their bit positions, and bit positions that are powers of 2 are used as parity bits (positions 1, 2, 4, 8 …). Table 38 lists the bit positions and the positions of the parity bits in a 13-bit code word.

**Table 38. Design Example 2: Position of Parity Bits for a 13-Bit Code Word**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits and data bits | P1<br>0 | P2<br>0 | | P3<br>1 | | | | P4<br>0 | | | | | P5*<br>1 |

b. All other bit positions are for the data bits. The LSB of the data bit fills the lowest bit position. In this case, starting from the LSB of the data, F0 (1111 0000 in binary) fills the empty bit positions, starting from position (3), as shown in Table 39.

**Table 39. Design Example 2: Filling of Data Bits (1111 0000) for a 13-Bit Code Word**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits and data bits | P1<br>0 | P2<br>0 | D1<br>0 | P3<br>1 | D2<br>0 | D3<br>0 | D4<br>0 | P4<br>0 | D5<br>1 | D6<br>1 | D7<br>1 | D8<br>1 | P5*<br>1 |

c. Recalculate parity bits to generate the syndrome code. Each syndrome bit calculates the parity (even parity) for some of the bits in the code word. Table 40 lists how the syndrome bits are derived.

**Table 40. Design Example 2: Calculation of Parity Bits**

| Position | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parity Bits and data bits | P1 | P2 | D1<br>0 | P3 | D2<br>0 | D3<br>0 | D4<br>0 | P4 | D5<br>1 | D6<br>1 | D7<br>1 | D8<br>1 | P5* | Syndrome Code |
| Calculate P1 | 0 | | 0 | | 0 | | 0 | | 1 | | 1 | | | S1=0 |
| Calculate P2 | | 0 | 0 | | | 0 | 0 | | | 1 | 1 | | | S2=0 |
| Calculate P3 | | | | 1 | 0 | 0 | 0 | | | | | 1 | | S3=0 |
| Calculate P4 | | | | | | | | 0 | 1 | 1 | 1 | 1 | | S4=0 |
| Calculate P5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | S5*=0 |

d. Calculate the additional syndrome bit using an even parity checking on all the bits in the code word. In this example, the additional syndrome bit S5* is calculated using an even parity checking on all the bits from position (1) to position (13) as shown in Table 40. The generated syndrome code gives the status of the data, whether an error has occurred, and if so, whether it is a single-bit or double-bit error.

2. In this case, the syndrome code is zero (S5*S4S3S2S1=0 0000). No error is detected and no correction is needed on the retrieved data F0 (D8D7D6D5D4D3D2D1=1111 0000) based on the generated syndrome code. Therefore, the flag signals `err_detected`, `err_corrected`, and `err_fatal` are deasserted, indicating that the data is not corrupted. The decoding for 14F0 is F0 (1111 0000 in binary).

   For more information about the generated syndrome code, refer to Table 31 on page 39.

   Even if the generated syndrome code indicates a single-bit error, the `err_detected` and `err_corrected` signals are asserted only if the corrupted bit is from the data bits and not from the parity bits.

3. At 10 ns, a single-bit error occurred in the input code word that changes the code word to 14F1. In this case, assume that one of the data bits, the LSB, is corrupted and is inverted from 0 to 1. This causes the code word to become 14F1.

   With the same method of decoding using the Hamming Code scheme, the generated syndrome code is 1 0011. S5* equals to 1 (single error detected), and S4S3S2S1 equals to 0011 (the bit at position 3 is corrupted).

   Because only one of the data bits is corrupted, the decoder is able to correct it by flipping the error bit. Therefore, the corrupted data F1 is decoded as F0. When F0 is shown at the output port at the next rising edge of the clock at 17.5 ns, the `err_detected` and `err_corrected` signals are asserted to show that an error is detected and the single-bit error is corrected.

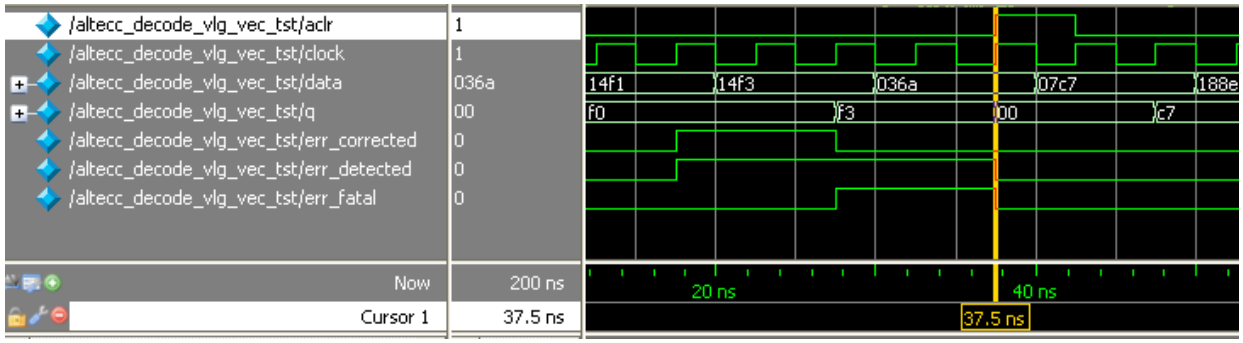
4. At 20 ns, a double-bit error in the input code word changes the code word to 14F3.

   In this case, assume that two of the data bits (bit-0 and bit-1) are corrupted and are inverted from 0 to 1. This causes the code word to become 14F3.

   The decoder decodes the code word 14F3 at 20 ns and shows the data F3 at 27.5 ns. The ECC decoder performs only SECDED, therefore it does not fix the corrupted data that contains double-bit errors. Instead, the `err_fatal` signal is asserted together with the `err_detected` signal.

Figure 15 shows the effects of the asynchronous-clear signal on the registered ports.

**Figure 15. Design Example 2: Asynchronous-Clear Feature of ECC**



If you do not want to use the corrupted data when the `err_fatal` signal is asserted, you can assert the asynchronous-clear signal (`aclr`) to clear the output port `q` and other status signals that are registered. You must enable the pipelining option in the MegaWizard Plug-In Manager to use this feature.

Figure 15 shows that when the `aclr` signal is asserted at 37.5 ns, the output and status signals are cleared immediately.

## Specifications

This section describes the prototypes, component declarations, ports, and parameters of the ALTECC_ENCODER and ALTECC_DECODER megafunctions. These ports and parameters are available to customize the ALTECC_ENCODER and ALTECC_DECODER megafunctions according to your application.

### Verilog HDL Prototype (ALTECC_ENCODER)

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altecc_encoder
#( parameter intended_device_family = "unused",
parameter lpm_pipeline = 0,
parameter width_codeword = 8,
parameter width_dataword = 8,
parameter lpm_type = "altecc_encoder",
parameter lpm_hint = "unused")
( input wire aclr,
input wire clock,
input wire clocken,
input wire [width_dataword-1:0] data,
output wire [width_codeword-1:0] q);
endmodule
```

### Verilog HDL Prototype (ALTECC_DECODER)

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altecc_decoder
#( parameter intended_device_family = "unused",
parameter lpm_pipeline = 0,
parameter width_codeword = 8,
parameter width_dataword = 8,
parameter lpm_type = "altecc_decoder",
parameter lpm_hint = "unused")
( input wire aclr,
input wire clock,
input wire clocken,
input wire [width_codeword-1:0] data,
output wire err_corrected,
output wire err_detected,
outut wire err_fatal,
output wire [width_dataword-1:0] q);
endmodule
```

### VHDL Component Declaration (ALTECC_ENCODER)

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 altecc_encoder
generic (
intended_device_family:string := "unused";
lpm_pipeline:natural := 0;
width_codeword:natural := 8;
width_dataword:natural := 8;
lpm_hint:string := "UNUSED";
lpm_type:string := "altecc_encoder");
port(
aclr:in std_logic := '0';
clock:in std_logic := '0';
clocken:in std_logic := '1';
data:in std_logic_vector(width_dataword-1 downto 0);
q:out std_logic_vector(width_codeword-1 downto 0));
end component;
```

### VHDL Component Declaration (ALTECC_DECODER)

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 altecc_decoder
generic (
intended_device_family:string := "unused";
lpm_pipeline:natural := 0;
width_codeword:natural := 8;
width_dataword:natural := 8;
lpm_hint:string := "UNUSED";
lpm_type:string := "altecc_decoder");
port(
aclr:in std_logic := '0';
clock:in std_logic := '0';
clocken:in std_logic := '1';
data:in std_logic_vector(width_codeword-1 downto 0);
q:out std_logic_vector(width_dataword-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 41 lists the input ports, Table 42 lists the output ports, and Table 43 lists the ALTECC_ENCODER megafunction parameters.

**Table 41. ALTECC_ENCODER Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | Yes | Data input port. The size of the input port depends on the WIDTH_DATAWORD parameter value. The data[] port contains the raw data to be encoded. |
| clock | Yes | Clock input port that provides the clock signal to synchronize the encoding operation. The clock port is required when the LPM_PIPELINE value is greater than 0. |
| clocken | No | Clock enable. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear input. The active high aclr signal can be used at any time to asynchronously clear the registers. |

**Table 42. ALTECC_ENCODER Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| q[] | Yes | Encoded data output port. The size of the output port depends on the WIDTH_CODEWORD parameter value. |

**Table 43. ALTECC_ENCODER Megafunction Parameters**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH_DATAWORD | Integer | Yes | Specifies the width of the raw data. Values are from 2 to 64. If omitted, the default value is 8. |
| WIDTH_CODEWORD | Integer | Yes | Specifies the width of the corresponding code word. Valid values are from 6 to 72, excluding 9, 17, 33, and 65. If omitted, the default value is 13. |
| LPM_PIPELINE | Integer | No | Specifies the pipeline for the circuit. Values are from 0 to 2. If the value is 0, the ports are not registered. If the value is 1, the output ports are registered. If the value is 2, the input and output ports are registered. If omitted, the default value is 0. |

Table 44 lists the input ports, Table 45 lists the output ports, and Table 46 lists the ALTECC_DECODER megafunction parameters.

**Table 44. ALTECC_DECODER Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | Yes | Data input port. The size of the input port depends on the WIDTH_CODEWORD parameter value. |
| clock | Yes | Clock input port that provides the clock signal to synchronize the encoding operation. The clock port is required when the LPM_PIPELINE value is greater than 0. |
| clocken | No | Clock enable. If omitted, the default value is 1. |
| aclr | No | Asynchronous clear input. The active high aclr signal can be used at any time to asynchronously clear the registers. |

**Table 45. ALTECC_DECODER Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| q[] | Yes | Decoded data output port. The size of the output port depends on the WIDTH_DATAWORD parameter value. |
| err_detected | Yes | Flag signal to reflect the status of data received and specifies any errors found. |
| err_corrected | Yes | Flag signal to reflect the status of data received. Denotes single-bit error found and corrected. You can use the data because it has already been corrected. |
| err_fatal | Yes | Flag signal to reflect the status of data received. Denotes double-bit error found, but not corrected. You must not use the data if this signal is asserted. |

**Table 46. ALTECC_DECODER Megafunction Parameters**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH_DATAWORD | Integer | Yes | Specifies the width of the raw data. Values are 2 to 64. The default value is 8. |
| WIDTH_CODEWORD | Integer | Yes | Specifies the width of the corresponding code word. Values are 6 to 72, excluding 9, 17, 33, and 65. If omitted, the default value is 13. |
| LPM_PIPELINE | Integer | No | Specifies the register of the circuit. Values are from 0 to 2. If the value is 0, no register is implemented. If the value is **1**, the output is registered. If the value is 2, both the input and the output are registered. If the value is greater than 2, additional registers are implemented at the output for the additional latencies. If omitted, the default value is 0. |

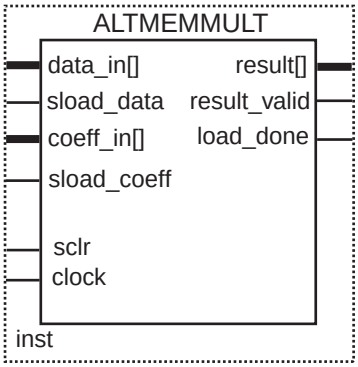
# ALTMEMMULT (Memory-based Constant Coefficient Multiplier)

The ALTMEMMULT megafunction is used to create memory-based multipliers using the on-chip memory blocks found in Altera FPGAs (with M512, M4K, M9K, and MLAB memory blocks). This megafunction is useful if you do not have sufficient resources to implement the multipliers in logic elements (LEs) or dedicated multiplier resources.

The ALTMEMMULT megafunction is a synchronous function that requires a clock. The ALTMEMMULT megafunction and the MegaWizard Plug-In Manager create a multiplier with the smallest throughput and latency possible for a given set of parameters and specifications.

Figure 16 shows the ports for the ALTMEMMULT megafunction.

**Figure 16. ALTMEMMULT Ports Shown in the MegaWizard Plug-In Manager**




## Features

The ALTMEMMULT megafunction offers the following features:

- Creates only memory-based multipliers using on-chip memory blocks found in Altera FPGAs
- Supports data width of 1–512 bits
- Supports signed and unsigned data representation format
- Supports pipelining with fixed output latency
- Stores multiples constants in random-access memory (RAM)
- Provides an option to select the RAM block type
- Supports optional synchronous clear and load-control input ports

Refer to the following megafunctions in this user guide for other multiplier implementations:

- Multiplier-Adder Megafunction (ALTMULT_ADD)
- Multiplier Megafunction (LPM_MULT)
- Multiplier-Accumulator Megafunction (ALTMULT_ACCUM)

For more information about implementing multipliers in Altera FPGAs, refer to *AN 306: Implementing Multipliers in FPGA Devices*.

## Resource Utilization and Performance

Table 47 provides resource utilization and performance information for the ALTMEMMULT megafunction.

**Table 47. ALTMEMMULT Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | data(4) × coeff(4) | 2 | 61 | 62 | 41 | 445 |
| | data(8) × coeff(8) | 7 | 65 | 88 | 55 | 567 |
| | data(16) × coeff(16) | 7 | 151 | 175 | 107 | 445 |
| Stratix IV | data(4) × coeff(4) | 2 | 43 | 55 | 36 | 623 |
| | data(8) × coeff(8) | 7 | 65 | 88 | 56 | 605 |
| | data(16) × coeff(16) | 7 | 109 | 156 | 96 | 570 |

**Notes to Table 47:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: 8 × 8 Multiplier

This design example uses the ALTMEMMULT megafunction to generate a basic multiplier using RAM blocks to determine the 16-bit product of two unsigned 8-bit numbers. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **altmemmult_DesignExample.zip**:

- **memmult_ex.qar** (archived Quartus II design files)
- **altmemmult_ex_msim** (ModelSim-Altera files)

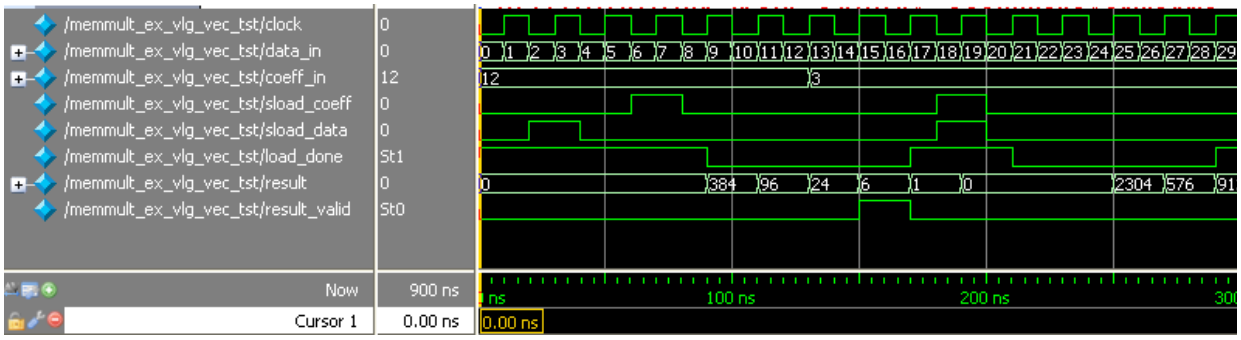
## Understanding the Simulation Results

The following settings are observed in this example:

- The `data_in[]` and `coeff_in[]` input widths are both set to 8 bits
- The output port, `result[]` is set to a width of 16 bits
- The initial coefficient is 2
- The output latency is fixed to seven clock cycles based on the input widths set

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

**Figure 17. ALTMEMMULT Simulation Results**




This design example implements a multiplier for unsigned 8-bit numbers. If the value of the MAX_CLOCK_CYCLES_PER_RESULT parameter is more than 1, the sload_data signal indicates a new multiplication and the result_valid signal indicates the validity of the multiplication result.

If the value of the MAX_CLOCK_CYCLES_PER_RESULT parameter is 1, the sload_data signal is not used and every positive clock edge starts a new multiplication.

In this design example, with the MAX_CLOCK_CYCLES_PER_RESULT parameter set to 4, the design requires no less than four clock cycles to compute the multiplication. The sload_data signal is used to indicate a new multiplication.

Altera recommends that you do not pull the sload_data signal high during the four clock cycles when the multiplication is taking place to avoid getting unpredictable results.

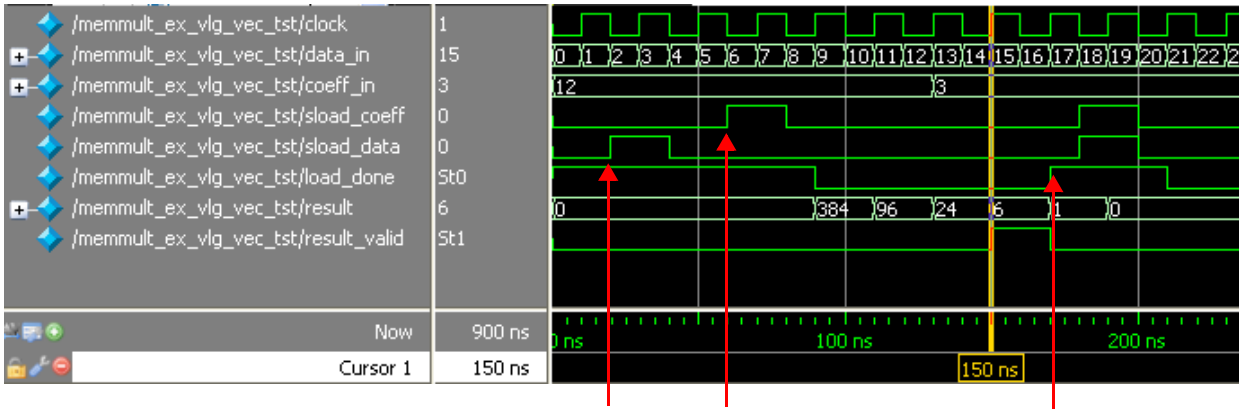
The megafunction only receive new inputs after four clock cycles. With the TOTAL_LATENCY parameter set to 7, all multiplication results require seven clock cycles to appear at the result[] port. The COEFFICIENT0 parameter holds the value of the first fixed coefficient, which is set to 2 (COEFFICIENT0=2) for this design example. The megafunction uses the latest coefficient value for every multiplication.

The sload_data signal asserts when a new coefficient value is written into the register. The load_done signal pulls low one clock cycle after the sload_data signal deasserts. When the load_done signal is low, the new coefficient value is reprogrammed into the RAM look-up table. Until the load_done signal pulls high, no other coefficient value can be loaded into the memory (regardless of whether the sload_data signal asserts anytime in between). The load_done signal asserts when programming completes.

The load time required to write a new coefficient value into the register is the same for any instance of the ALTMEMMULT megafunction. However, the load time can vary depending on the size of the RAM used.

Figure 18 shows the simulation results for the multiplication implementation with coefficient of 2.

**Figure 18. Multiplication with Coefficient of 2**

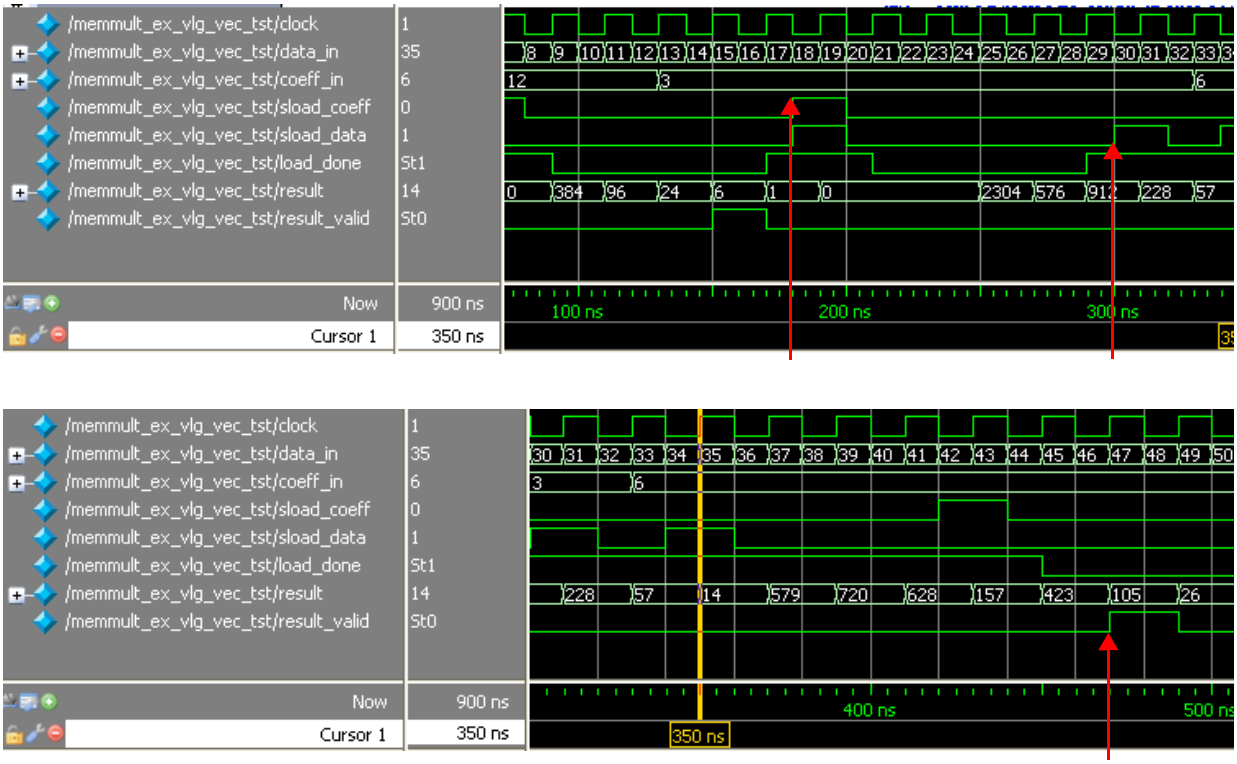



The following sequence corresponds with the numbered items in Figure 18:

1. At 30 ns, the `sload_data` signal asserts and triggers the first multiplication between the `data_in` value of 3 and the `COEFFICIENT0` value of 2. The result is sent to the result port seven clock cycles later at 150 ns. The `result_valid` signal asserts to indicate that the multiplication is valid.
2. At 70 ns, the `sload_coeff` signal asserts to register a new coefficient value of 12 into the register.
3. The `load_done` signal pulls low to begin loading the new value into the memory, and pulls high at 170 ns when loading is complete. In this example, the load time is five clock cycles.

Figure 19 shows the simulation results for the multiplication implementation with coefficient of 3.

**Figure 19. Multiplication with Coefficient of 3**




The following sequence corresponds with the numbered items in Figure 19:

1. At 190 ns, the `sload_coeff` signal asserts to register a new coefficient value of 3. The `sload_data` signal asserts and triggers a new multiplication. The latest value of the coefficient loaded into the memory is 12. Multiplication occurs between the `data_in` value of 19 and a coefficient of 12. At the same time, at 190 ns, the `sload_coeff` signal asserts and triggers the programming of coefficient 3. Although the multiplication result of 228 at 310 ns is valid, the `result_valid` signal does not pull high.
2. At 300 ns, the `sload_data` signal asserts and triggers a new multiplication. However, at 350 ns (less than four clock cycles after 300 ns), the `sload_data` signal pulls high again and cancels the previous multiplication process.
3. Multiplication finally occurs between the `data_in` value of 35 and a coefficient of 3 at 350 ns.The valid result, 105, of the computation is displayed at 470 ns.

Altera recommends that you do not assert both the `sload_coeff` and `sload_data` signals at the same time to prevent the programming and computation processes from occurring simultaneously.

# Specifications

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

## Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altmemmult
#( parameter coeff_representation = "SIGNED",
parameter coefficient0 = "UNUSED",
parameter data_representation = "SIGNED",
parameter intended_device_family = "unused",
parameter max_clock_cycles_per_result = 1,
parameter number_of_coefficients = 1,
parameter ram_block_type = "AUTO",
parameter total_latency = 1,
parameter width_c = 1,
parameter width_d = 1,
parameter width_r = 1,
parameter width_s = 1,
parameter lpm_type = "altmemmult",
parameter lpm_hint = "unused")
( input wire clock,
input wire [width_c-1:0]coeff_in,
input wire [width_d-1:0] data_in,
output wire load_done,
output wire [width_r-1:0] result,
output wire result_valid,
input wire sclr,
input wire [width_s-1:0] sel,
input wire sload_coeff,
input wire sload_data)/* synthesis syn_black_box=1 */;
endmodule
```

## 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 altmemmult
generic (
coeff_representation:string := "SIGNED";
coefficient0:string := "UNUSED";
data_representation:string := "SIGNED";
intended_device_family:string := "unused";
max_clock_cycles_per_result:natural := 1;
number_of_coefficients:natural := 1;
ram_block_type:string := "AUTO";
total_latency:natural;
width_c:natural;
width_d:natural;
width_r:natural;
width_s:natural := 1;
lpm_hint:string := "UNUSED";
lpm_type:string := "altmemmult");
port(
clock:in std_logic;
coeff_in:in std_logic_vector(width_c-1 downto 0) := (others => '0');
```

```
data_in:in std_logic_vector(width_d-1 downto 0);
load_done:out std_logic;
result:out std_logic_vector(width_r-1 downto 0);
result_valid:out std_logic;
sclr:in std_logic := '0';
sel:in std_logic_vector(width_s-1 downto 0) := (others => '0');
sload_coeff:in std_logic := '0';
sload_data:in std_logic := '0');
end component;
```

## Ports and Parameters

Table 48 lists the input ports, Table 49 lists the output ports, and Table 50 lists the ALTMEMMULT megafunction parameters.

**Table 48. ALTMEMMULT Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| clock | Yes | Clock input to the multiplier. |
| coeff_in[] | No | Coefficient input port for the multiplier. The size of the input port depends on the WIDTH_C parameter value. |
| data_in[] | Yes | Data input port to the multiplier. The size of the input port depends on the WIDTH_D parameter value. |
| sclr | No | Synchronous clear input. If unused, the default value is active high. |
| sel[] | No | Fixed coefficient selection. The size of the input port depends on the WIDTH_S parameter value. |
| sload_coeff | No | Synchronous load coefficient input port. Replaces the current selected coefficient value with the value specified in the coeff_in input. |
| sload_data | No | Synchronous load data input port. Signal that specifies new multiplication operation and cancels any existing multiplication operation. If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the sload_data input port is ignored. |

**Table 49. ALTMEMMULT Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Multiplier output port. The size of the input port depends on the WIDTH_R parameter value. |
| result_valid | Yes | Indicates when the output is the valid result of a complete multiplication. If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the result_valid output port is not used. |
| load_done | No | Indicates when the new coefficient has finished loading. The load_done signal asserts when a new coefficient has finished loading. Unless the load_done signal is high, no other coefficient value can be loaded into the memory. |

**Table 50. ALTMEMMULT Megafunction Parameters (Part 1 of 2)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH_D | Integer | Yes | Specifies the width of the data_in[] port. |
| WIDTH_C | Integer | Yes | Specifies the width of the coeff_in[] port. |
| WIDTH_R | Integer | Yes | Specifies the width of the result[] port. |
| WIDTH_S | Integer | No | Specifies the width of the sel[] port. |
| COEFFICIENT0 | Integer | Yes | Specifies value of the first fixed coefficient. |

**Table 50. ALTMEMMULT Megafunction Parameters (Part 2 of 2)**

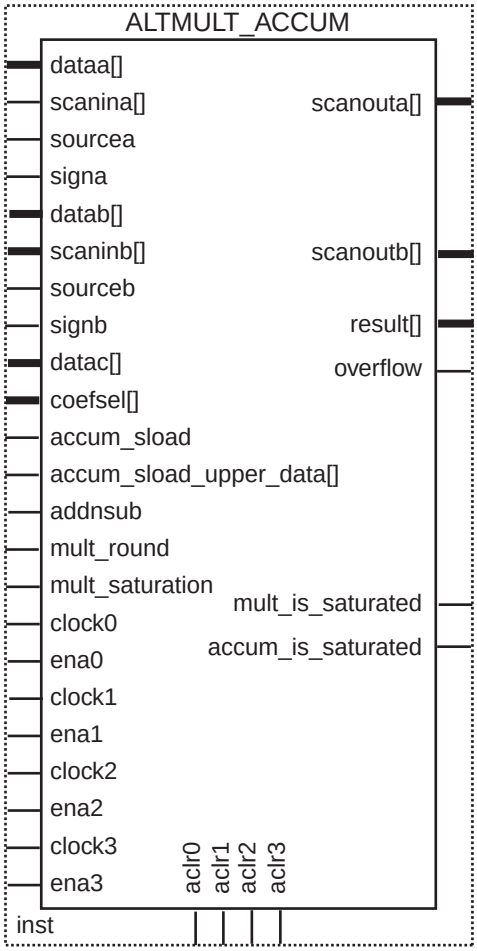
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| TOTAL_LATENCY | Integer | Yes | Specifies the total number of clock cycles from the start of a multiplication to the time the result is available at the output. |
| DATA_REPRESENTATION | String | No | Specifies whether the coeff_in[] input port and the pre-loaded coefficients are signed or unsigned. |
| COEFF_REPRESENTATION | String | No | Specifies whether the coeff_in[] input port and the pre-loaded coefficients are signed or unsigned. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTMEMMULT megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| MAX_CLOCK_CYCLES_PER_ RESULT | Integer | No | Specifies the number of clock cycles per result. |
| NUMBER_OF_COEFFICIENTS | Integer | No | Specifies the number of coefficients that are stored in the lookup table. |
| RAM_BLOCK_TYPE | String | No | Specifies the ram block type. Values are AUTO, SMALL, MEDIUM, M512, and M4K. If omitted, the default value is AUTO. |

# ALTMULT_ACCUM (Multiply-Accumulate)

The ALTMULT_ACCUM megafunction allows you to implement a multiplier-accumulator.

Figure 20 shows the ports for the ALTMULT_ACCUM megafunction.

**Figure 20. ALTMULT_ACCUM Ports Shown in the MegaWizard Plug-In Manager**




A multiplier-accumulator accepts a pair of inputs, multiplies the two inputs together, and feeds their result into an accumulator to be added to or subtracted from its previous registered result. This function is expressed in Equation 1.

**Equation 1.**

$$y = \sum_{i-0}^{N-1} (\pm 1) \times A_i \times B_i$$

In Equation 1, $N$ is the number of cycles of data that has been entered into the accumulator.

## Features

The ALTMULT_ACCUM megafunction offers the following features:

- Generates a multiplier-accumulator
- Supports data widths of 1–256 bits
- Supports signed and unsigned data representation format
- Supports pipelining with configurable output latency
- Provides a choice of implementation in dedicated DSP block circuitry or logic elements (LEs)
- Provides an option to dynamically switch between add and subtract operations in the accumulator
- Provides an option to dynamically switch between signed and unsigned data support
- Provides an option to set up data shift register chains
- Supports hardware saturation and rounding (for Stratix III and Stratix IV devices only)
- Supports optional asynchronous clear and clock enable input ports
- Supports systolic delay register mode (for Arria V, Cyclone V, and Stratix V devices only)
- Supports pre-adder with 8 coefficients per multiplier (for Arria V, Cyclone V, and Stratix V devices only)
- Supports pre-load constant to complement accumulator feedback (for Arria V, Cyclone V, and Stratix V devices only)

Refer to the following megafunctions in this user guide for other multiplier implementations:

- Multiplier-Adder Megafunction (ALTMULT_ADD)
- Memory-based Constant Coefficient Multiplier (ALTMEMMULT)
- Multiplier Megafunction (LPM_MULT)

## Resource Utilization and Performance

Table 51 provides resource utilization and performance information for the ALTMULT_ACCUM megafunction.

**Table 51. ALTMULT_ACCUM Resource Utilization and Performance (1)**

| Device family | Input data width | Number of multipliers | Logic Usage | | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | 18-bit DSP | |
| Stratix III | 16 ×16 | 1 | 0 | 0 | 0 | 4 | 481 |
| | 16 ×16 | 2 | 0 | 0 | 0 | 4 | 481 |
| Stratix IV | 16 ×16 | 1 | 0 | 0 | 0 | 4 | 443 |
| | 16 ×16 | 2 | 0 | 0 | 0 | 4 | 443 |

**Notes to Table 51:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

In the Stratix, Stratix GX, and Arria GX device series, the multiplier and the accumulator of the ALTMULT_ACCUM megafunction are placed in the DSP block circuitry. For Stratix V devices, the multiplier blocks and adder/accumulator block (mac_mult and mac_out) are combined into a single multiplier accumulator (MAC) block. The DSP blocks use the 18-bit × 18-bit input multiplier to process data with widths of up to 18 bits.

The registers and extra pipeline registers for the following signals are also placed inside the DSP block:

- Data input
- Signed or unsigned select
- Add or subtract select
- Synchronous load
- Products of multipliers

In the case of the output result, the first register is placed in the DSP block. The extra latency registers are placed in logic elements outside the block.

Cyclone II and Cyclone III devices have embedded multiplier blocks. When the ALTMULT_ACCUM megafunction is implemented in Cyclone II and Cyclone III devices, the multiplier is implemented in the embedded multiplier blocks, while the accumulator is put in LEs. In Cyclone devices, both the multiplier and accumulator are placed in LEs.

For more information about DSP blocks in any of the Stratix, Stratix GX, and Arria GX device series, refer to the *DSP Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information about the embedded memory blocks in any of the Stratix, Stratix GX, and Arria GX device series, refer to the *TriMatrix Embedded Memory Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information about embedded multiplier blocks in the Cyclone II and Cyclone III devices, refer to the *DSP Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information about implementing multipliers using DSP and memory blocks in Altera FPGAs, refer to *AN 306: Implementing Multipliers in FPGA Devices*.

## Design Example: Shift Accumulator

A multiplier-accumulator can be used to implement FIR filters. This design example uses the ALTMULT_ACCUM megafunction to implement a serial FIR filter, in which both the data and coefficient are shifted serially into the multiplier and then summed in the accumulator. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **altmult_accum_DesignExample.zip**:

- **serial_fir.qar** (archived Quartus II design files)
- **altmult_accum_ex_msim** (ModelSim-Altera files)

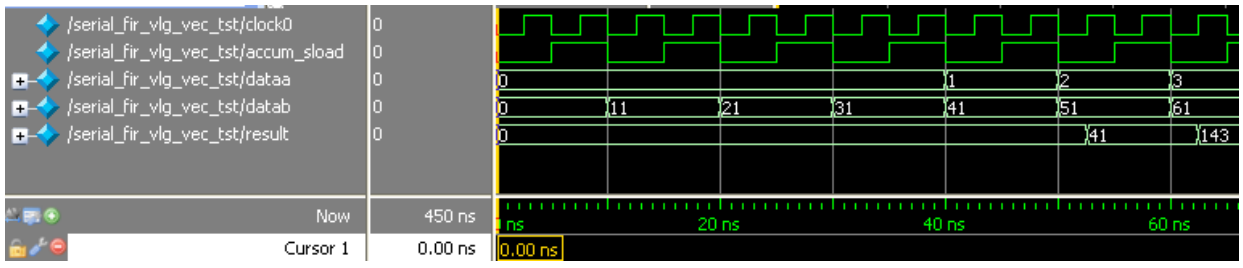
## Understanding the Simulation Results

The following settings are observed in this example:

- The `dataa[]` and `datab[]` input widths are both set to 16 bits
- The output port, `result[]` is set to a width of 33 bits
- The `accum_sload` input is enabled

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

**Figure 21. ALTMULT_ACCUM Simulation Results**



# Specifications

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

## Verilog HDL Prototype

To view the Verilog HDL prototype for ALTMULT_ACCUM megafunction, refer to the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

## VHDL Component Declaration

To view the VHDL component declaration for ALTMULT_ACCUM megafunction, refer to 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 52 and Table 53 list the input and output ports of the ALTMULT_ACCUM megafunction.

For Arria V, Cyclone V, and Stratix V devices, each register can select between 2 asynchronous clear signals (ACLR0, ACLR1) and 3 clock/enable pairs (CLOCK0/ENA0, CLOCK1/ENA1, CLOCK2/ENA2).

**Table 52. ALTMULT_ACCUM Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| accum_sload | No | Causes the value on the accumulator feedback path to go to zero (0) or to accum_sload_upper_data when concatenated with 0. If the accumulator is adding and the accum_sload port is high, then the multiplier output is loaded into the accumulator. If the accumulator is subtracting, then the opposite (negative value) of the multiplier output is loaded into the accumulator. |
| aclr0 | No | The first asynchronous clear input. The aclr0 port is active high. |
| aclr1 | No | The second asynchronous clear input. The aclr1 port is active high. |
| aclr2 | No | The third asynchronous clear input. The aclr2 port is active high. |
| aclr3 | No | The fourth asynchronous clear input. The aclr3 port is active high. |
| addnsub | No | Controls the functionality of the adder. If the addnsub port is high, the adder performs an add function; if the addnsub port is low, the adder performs a subtract function. |
| clock0 | No | Specifies the first clock input, usable by any register in the megafunction. |
| clock1 | No | Specifies the second clock input, usable by any register in the megafunction. |

**Table 52. ALTMULT_ACCUM Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| clock2 | No | Specifies the third clock input, usable by any register in the megafunction. |
| clock3 | No | Specifies the fourth clock input, usable by any register in the megafunction. |
| dataa[] | Yes | Data input to the multiplier. The size of the input port depends on the WIDTH_A parameter value. |
| datab[] | Yes | Data input to the multiplier. The size of the input port depends on the WIDTH_B parameter value. |
| ena0 | No | Clock enable for the clock0 port. |
| ena1 | No | Clock enable for the clock1 port. |
| ena2 | No | Clock enable for the clock2 port. |
| ena3 | No | Clock enable for the clock3 port. |
| signa | No | Specifies the numerical representation of the dataa[] port. If the signa port is high, the multiplier treats the dataa[] port as signed two's complement. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number. |
| signb | No | Specifies the numerical representation of the datab[] port. If the signb port is high, the multiplier treats the datab[] port as signed two's complement. If the signb port is low, the multiplier treats the datab[] port as an unsigned number. |
| **Ports Available in Stratix II devices only** | | |
| accum_saturation | No | Enables accumulator saturation. |
| mult_round | No | Enables multiplier rounding. |
| mult_saturation | No | Enables multiplier saturation. |
| **Ports Available in Stratix II and Cyclone II devices only** | | |
| accum_sload_ upper_data[] | No | Input for accumulator upper data bits during a synchronous load. |
| scanina[] | No | Input for scan chain A. |
| scaninb[] | No | Input for scan chain B. |
| **Ports Available in Stratix II, Cyclone II, and HardCopy devices only** | | |
| sourcea | No | Input source for scan chain A and dynamically controls whether the scanina[] and dataa[] ports are fed to the multiplier. |
| sourceb | No | Input source for scan chain B. |
| **Ports Available in Stratix III and Stratix IV devices only** | | |
| accum_round | No | Enables accumulator rounding. |
| **Ports Available in Arria V, Cyclone V, and Stratix V devices only** | | |
| coefsel0[] | No | Coefficient input. |
| coefsel1[] | No | Coefficient input. |
| coefsel2[] | No | Coefficient input. |
| coefsel3[] | No | Coefficient input. |
| datac[] | Yes | Data input to the multiplier. |

**Table 53. ALTMULT_ACCUM Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| overflow | No | Overflow port for the accumulator. |
| result[] | Yes | Accumulator output port. The size of the output port depends on the WIDTH_RESULT parameter value. |
| scanouta[] | No | Output of the first shift register. The size of the output port depends on the WIDTH_A parameter value. When instantiating the ALTMULT_ACCUM megafunction with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanouta[] port to shiftouta port. |
| scanoutb[] | No | Output of the second shift register. The size of the input port depends on the WIDTH_B parameter value. When instantiating the ALTMULT_ACCUM megafunction with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanoutb[] port to shiftoutb port. |
| **Ports Available in Stratix II devices only** | | |
| accum_is_saturated | No | Signal that indicates when accumulator saturation occurs. This port is available when the PORT_ACCUM_IS_SATURATED parameter is set to USED. |
| mult_is_saturated | No | Signal that indicates when multiplier saturation occurs. This port is available when the PORT_MULT_IS_SATURATED parameter is set to USED. |

Table 54 lists the ALTMULT_ACCUM megafunction parameters.

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 1 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| ACCUM_DIRECTION | String | No | Specifies whether the accumulator performs an add or subtract function. Values are ADD and SUB. When this parameter is set to ADD, the accumulator adds the product to the current accumulator value. When this parameter is set to SUB, the accumulator subtracts the product from the current accumulator value. If omitted the default value is ADD. This parameter is ignored if the addnsub port is used. |
| ACCUM_SLOAD_ACLR | String | No | Specifies the asynchronous clear signal for the accum_sload port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. This parameter is ignored if the accum_sload port is unused. |
| ACCUM_SLOAD_PIPELINE_ACLR | String | No | Specifies the asynchronous clear signal for the second register on the accum_sload port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. This parameter is ignored if the accum_sload port is unused. |
| ACCUM_SLOAD_PIPELINE_REG | String | No | Specifies the clock signal for the second register on the accum_sload port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the accum_sload port is unused. |
| ACCUM_SLOAD_REG | String | No | Specifies the clock signal for the accum_sload port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the accum_sload port is unused. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 2 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| ADDNSUB_ACLR | String | No | Specifies the asynchronous clear for the addnsub port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR0. This parameter is ignored if the addnsub port is unused. |
| ADDNSUB_PIPELINE_ACLR | String | No | Specifies the asynchronous clear for the second register on the addnsub port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR0. This parameter is ignored if the addnsub port is unused. |
| ADDNSUB_PIPELINE_REG | String | No | Specifies the clock for the second register on the addnsub port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the addnsub port is unused. |
| ADDNSUB_REG | String | No | Specifies the clock for the addnsub port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the addnsub port is unused. |
| DSP_BLOCK_BALANCING | String | No | Specifies whether to use DSP block balancing. Values are UNUSED, Auto, DSP blocks, Logic Elements, Off, Simple 18-bit Multipliers, Simple Multipliers, and Width 18-bit Multipliers. |
| EXTRA_ACCUMULATOR_ LATENCY | String | No | Adds the number of clock cycles of latency specified by the OUTPUT_REG parameter to the accumulator portion of the DSP block. |
| EXTRA_MULTIPLIER_ LATENCY | Integer | No | Specifies the number of clock cycles of latency for the multiplier portion of the DSP block. If the MULTIPLIER_REG parameter is specified, then the specified clock port is used to add the latency. If the MULTIPLIER_REG parameter is set to UNREGISTERED, then the clock0 port is used to add the latency. |
| INPUT_ACLR_A | String | No | Specifies the asynchronous clear port for the dataa[] port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. |
| INPUT_ACLR_B | String | No | Specifies the asynchronous clear port for the datab[] port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. |
| INPUT_REG_A | String | No | Specifies the clock port for the dataa[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| INPUT_REG_B | String | No | Specifies the clock port for the datab[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. If omitted, the default value is CLOCK0. |
| INTENDED_DEVICE_ FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTMULT_ACCUM megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 3 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MULTIPLIER_ACLR | String | No | Specifies the asynchronous clear signal for the register immediately following the multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. |
| MULTIPLIER_REG | String | No | Specifies the clock signal for the register that immediately follows the multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| OUTPUT_ACLR | String | No | Specifies the asynchronous clear signal for the registers on the outputs. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. |
| OUTPUT_REG | String | No | Specifies the clock signal for the registers on the outputs. Values are CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted the default value is CLOCK0. |
| PORT_ADDNSUB | String | No | Specifies the usage of the addnsub input port. Values are: PORT_USED = Port is treated as used. PORT_UNUSED = Port is treated as unused. PORT_CONNECTIVITY = Port usage is determined by checking the port connectivity. If omitted the default value is PORT_CONNECTIVITY. |
| PORT_SIGNA | String | No | Specifies the usage of the signa input port. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. If omitted the default value is PORT_CONNECTIVITY. |
| PORT_SIGNB | String | No | Specifies the usage of the signb input port. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. If omitted the default value is PORT_CONNECTIVITY. |
| REPRESENTATION_[] | String | No | Parameter [A,B]. Specifies the numerical representation of the corresponding data[] port. Values are UNSIGNED and SIGNED. When this parameter is set to SIGNED, the accumulator interprets the dataa input as signed two's complement. If omitted, the default value is UNSIGNED. This parameter is ignored if the signa port is used. |
| SIGN_ACLR_[] | String | No | Parameter [A,B]. Specifies the asynchronous clear signal for the first register on the corresponding sign[] port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. This parameter is ignored if the corresponding sign[] port is unused. |
| SIGN_PIPELINE_ACLR_[] | String | No | Parameter [A,B]. Specifies the asynchronous clear signal for the second register on the corresponding sign[] port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted the default value is ACLR3. This parameter is ignored if the corresponding sign[] port is unused. |
| SIGN_PIPELINE_REG_[] | String | No | Parameter [A,B]. Specifies the clock signal for the second register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the corresponding sign[] port is unused. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 4 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| SIGN_REG_[] | String | No | Parameter [A,B]. Specifies the clock signal for the first register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. This parameter is ignored if the corresponding sign[] port is unused. |
| WIDTH_A | Integer | Yes | Specifies the width of the dataa[] port. |
| WIDTH_B | Integer | Yes | Specifies the width of the datab[] port. |
| WIDTH_RESULT | Integer | No | Specifies the width of the result[] port. |
| **Parameters Available in Stratix II devices only** | | | |
| ACCUM_ROUND_ACLR | String | No | Specifies the asynchronous clear port for the accum_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_ROUND_PIPELINE_ACLR | String | No | Specifies the asynchronous clear port for the "second stage" accum_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_ROUND_PIPELINE_REG | String | No | Specifies the clock port for the "second stage" accum_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| ACCUM_ROUND_REG | String | No | Specifies the clock port for the accum_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| ACCUM_SATURATION_ACLR | String | No | Specifies the asynchronous clear port for the accum_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_SATURATION_PIPELINE_ACLR | String | No | Specifies the asynchronous clear port for the "second stage" accum_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_SATURATION_PIPELINE_REG | String | No | Specifies the clock port for the "second stage" accum_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| ACCUM_SATURATION_REG | String | No | Specifies the clock port for the accum_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| ACCUMULATOR_ROUNDING | String | No | Specifies accumulator rounding. Values are NO, YES, and VARIABLE. If omitted the default value is NO. |
| ACCUMULATOR_SATURATION | String | No | Specifies accumulator saturation. Values are NO, YES, and VARIABLE. If omitted the default value is NO. |
| MULT_ROUND_ACLR | String | No | Specifies the asynchronous clear port for the mult_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is UNREGISTERED. |
| MULT_ROUND_REG | String | No | Specifies the clock port for the mult_round port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 5 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MULT_SATURATION_ACLR | String | No | Specifies the asynchronous clear port for the mult_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is UNREGISTERED. |
| MULT_SATURATION_REG | String | No | Specifies the clock port for the mult_saturation port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| MULTIPLIER_ROUNDING | String | No | Specifies multiplier rounding. Values are NO, YES, and VARIABLE. If omitted the default value is NO. |
| PORT_ACCUM_IS_SATURATED | String | No | Specifies whether to use the mult_is_saturated output port. Values are UNUSED and USED. If omitted the default value is UNUSED. |
| PORT_MULT_IS_SATURATED | String | No | Specifies whether to use the accum_is_saturated output port. Values are UNUSED and USED. If omitted the default value is UNUSED. |
| **Parameters Available in Stratix II and Cyclone II devices only** | | | |
| ACCUM_SLOAD_UPPER_DATA_ACLR | String | No | Asynchronous clear port for the accum_sload_upper_data port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_SLOAD_UPPER_DATA_PIPELINE_ACLR | String | No | Specifies the asynchronous clear port for the "second stage" accum_sload_upper_data port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. |
| ACCUM_SLOAD_UPPER_DATA_PIPELINE_REG | String | No | Specifies the asynchronous clear port for the "second stage" accum_sload_upper_data port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| ACCUM_SLOAD_UPPER_DATA_REG | String | No | Specifies the clock port for the accum_sload_upper_data port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| INPUT_SOURCE_A | String | No | Specifies the input port for the dataa[] port. Values are DATAA, SCANA, and VARIABLE. If omitted, the default value is DATAA. |
| INPUT_SOURCE_B | String | No | Specifies the input port for the datab[] port. Values are DATAB, SCANB, and VARIABLE. If omitted, the default value is DATAB. |
| WIDTH_UPPER_DATA | Integer | No | Specifies the width of the accum_sload_upper_data[] port. |
| **Parameters Available in Stratix, Stratix GX, Stratix II, Cyclone II, and HardCopy devices only** | | | |
| DEDICATED_MULTIPLIER_CIRCUITRY | String | No | Specifies whether to use dedicated multiplier circuitry. Values are AUTO, ON, and OFF. If omitted, the default value is AUTO. |
| **Parameters Available in Stratix II, Stratix II GX, Stratix III, Stratix IV, Arria GX, and HardCopy devices only** | | | |
| MULTIPLIER_SATURATION | String | No | Specifies multiplier saturation. Values are NO, YES, and VARIABLE. If omitted the default value is NO. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 6 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| **Parameters Available in Arria V, Cyclone V, and Stratix V devices only** | | | |
| INPUT_ACLR_C0 | String | No | Specifies the asynchronous clear for the datac[] operand of the first multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C0 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_C1 | String | No | Specifies the asynchronous clear for the datac[] operand of the second multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C1 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_C2 | String | No | Specifies the asynchronous clear for the datac[] operand of the third multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C2 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_C3 | String | No | Specifies the asynchronous clear for the datac[] operand of the fourth and corresponding multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C3 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_REGISTER_C0 | String | No | Specifies the clock port for the datac[] operand of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C1 | String | No | Specifies the clock port for the datac[] operand of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C2 | String | No | Specifies the clock port for the datac[] operand of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C3 | String | No | Specifies the clock port for the datac[] operand of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 7 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MUTIPLIER1_DIRECTION | String | No | Specifies whether the second multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If the addnsub1 port is used, this parameter is ignored.<br>If omitted, the default value is ADD. |
| MUTIPLIER3_DIRECTION | String | No | Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are ADD and SUB.<br>If the addnsub3 port is used, this parameter is ignored. If omitted, the default value is ADD. |
| NUMBER_OF_MULTIPLIERS | Integer | Yes | Number of multipliers to be added together. Values are 1 up to 4. |
| WIDTH_C | Integer | Yes | Specifies the width of the datac[] port. |
| WIDTH_COEF | Integer | Yes | Specifies the width of the coefsel[] port. |
| LOADCONST_VALUE | Integer | No | Pre-load constant value to complement accumulator mode. Values are 2^*N* where 0 < *N* < 64. |
| PREADDER_MODE | String | No | Specifies the mode of pre-adder settings to be used. Values are SIMPLE, COEF, INPUT, SQUARE, and CONSTANT. The default value is SIMPLE. |
| PREADDER_DIRECTION_0 | String | No | Specifies whether the pre-adder of the first multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_1 | String | No | Specifies whether the pre-adder of the second multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_2 | String | No | Specifies whether the pre-adder of the third multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_3 | String | No | Specifies whether the pre-adder of the fourth and corresponding multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| COEFFSEL0_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL1_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL2_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL3_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 8 of 9)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| COEFFSEL_A_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the first multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_B_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the second multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_C_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the third multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and the corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_D_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the fourth and corresponding multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and the corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| SYSTOLIC_DELAY1 | String | No | Specifies the clock source for the systolic register inputs of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| SYSTOLIC_DELAY3 | String | No | Specifies the clock source for the systolic register inputs of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| SYSTOLIC_ACLR1 | String | No | Specifies the asynchronous clear source for the systolic register inputs of the first multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and the corresponding SYSTOLIC_DELAY[]is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |
| SYSTOLIC_ACLR3 | String | No | Specifies the asynchronous clear source for the systolic register inputs of the third multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and the corresponding SYSTOLIC_DELAY[]is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |
| COEF0_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the first multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| COEF1_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the second multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |

**Table 54. ALTMULT_ACCUM Megafunction Parameters (Part 9 of 9)**

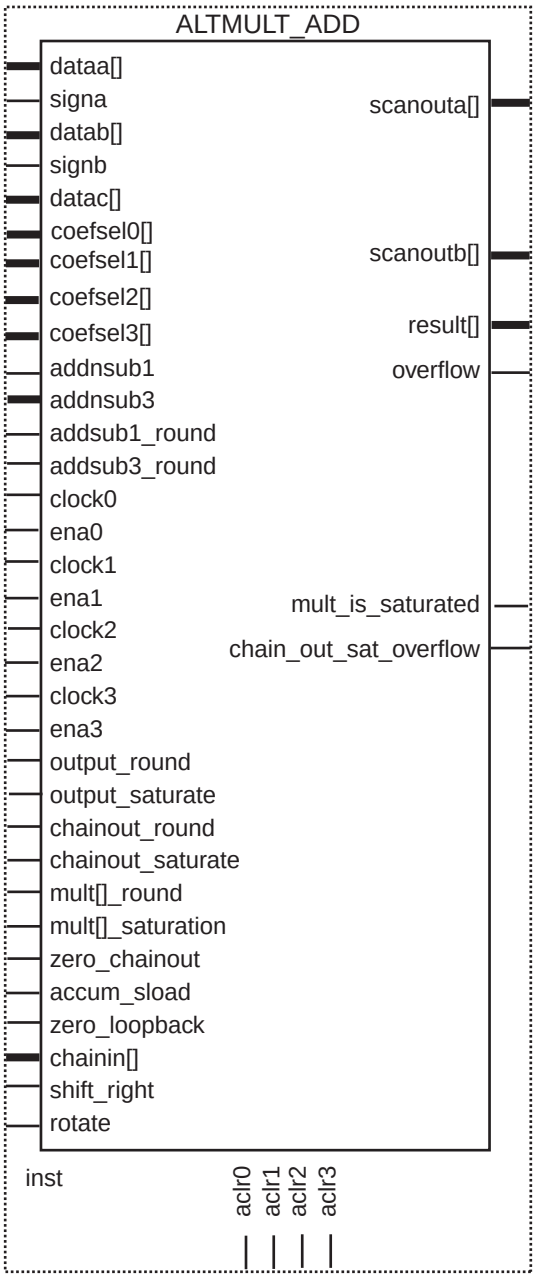
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| COEF2_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the third multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| COEF3_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the fourth multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |

# ALTMULT_ADD (Multiply-Add)

The ALTMULT_ADD megafunction allows you to implement a multiplier-adder.

Figure 22 shows the ports for the ALTMULT_ADD megafunction.

**Figure 22. ALTMULT_ADD Ports Shown in the MegaWizard Plug-In Manager**




A multiplier-adder accepts pairs of inputs, multiplies the values together and then adds to or subtracts from the products of all other pairs.

Apart from that, the ALTMULT_ADD megafunction offers many variations in dedicated DSP block circuitry. Data input sizes of up to 18 bits are accepted. Because the DSP blocks allow for one or two levels of 2-input add or subtract operations on the product, this function creates up to four multipliers.

Stratix III and Stratix IV device families use two MAC blocks (mac_mult and mac_out) to form DSP operations, multiply and add. For Stratix V devices, the multiplier blocks and adder/accumulator block is combined in a single MAC block.

The multipliers and adders of the ALTMULT_ADD megafunction are placed in the dedicated DSP block circuitry of the Stratix devices. If all of the input data widths are 9-bits wide or smaller, the function uses the 9 × 9-bit input multiplier configuration in the DSP block. If not, the DSP block uses 18 × 18-bit input multipliers to process data with widths between 10 bits and 18 bits. If multiple ALTMULT_ADD megafunctions occur in a design, the functions are distributed to as many different DSP blocks as possible so that routing to these blocks is more flexible. Fewer multipliers per DSP block allow more routing choices into the block by minimizing paths to the rest of the device.

The registers and extra pipeline registers for the following signals are also placed inside the DSP block:

- Data input
- Signed or unsigned select
- Add or subtract select
- Products of multipliers

In the case of the output result, the first register is placed in the DSP block. However the extra latency registers are placed in logic elements outside the block. Peripheral to the DSP block, including data inputs to the multiplier, control signal inputs, and outputs of the adder, use regular routing to communicate with the rest of the device. All connections in the function use dedicated routing inside the DSP block. This dedicated routing includes the shift register chains when you select the option to shift a multiplier's registered input data from one multiplier to an adjacent multiplier.

For more information about DSP blocks in any of the Stratix, Stratix GX, and Arria GX device series, refer to the *DSP Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information about the embedded memory blocks in any of the Stratix, Stratix GX, and Arria GX device series, refer to the *TriMatrix Embedded Memory Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information on embedded multiplier blocks in the Cyclone II and Cyclone III devices, refer to the *DSP Blocks* chapter of the respective handbooks on the Literature and Technical Documentation page.

For more information about implementing multipliers using DSP and memory blocks in Altera FPGAs, refer to *AN 306: Implementing Multipliers in FPGA Devices*.

# Features

The ALTMULT_ADD megafunction offers the following features:

- Generates a multiplier to perform multiplication operations of two complex numbers
- Supports data widths of 1– 256 bits
- Supports signed and unsigned data representation format
- Supports pipelining with configurable output latency
- Provides a choice of implementation in dedicated DSP block circuitry or logic elements (LEs)
- Provides an option to dynamically switch between signed and unsigned data support
- Provides an option to dynamically switch between add and subtract operation
- Provides an option to set up data shifting register chains
- Supports hardware saturation and rounding (for selected device families only)
- Supports optional asynchronous clear and clock enable input ports
- Supports systolic delay register mode (for Arria V, Cyclone V, and Stratix V devices only)
- Supports pre-adder with 8 pre-load coefficients per multiplier (for Arria V, Cyclone V, and Stratix V devices only)
- Supports pre-load constant to complement accumulator feedback (for Arria V, Cyclone V, and Stratix V devices only)

In Arria V, Cyclone V, and Stratix V devices, the pre-adder, coefficient storage and systolic delay register features are added to maximize flexibility. The following sections describe the new features.

## Pre-adder

With pre-adder, additions or subtractions are done prior to feeding the multiplier.

There are five pre-adder modes:

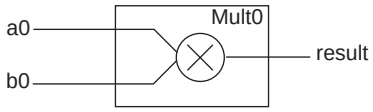
- Simple mode
- Coefficient mode
- Input mode
- Square mode
- Constant mode

When pre-adder is used (pre-adder coefficient/input/square mode), all data inputs to the multiplier must have the same clock setting.

### Pre-adder Simple Mode

In this mode, both operands derive from the input ports and pre-adder is not used or bypassed. This is the default mode.

**Figure 23. Pre-adder Simple Mode**




### Pre-adder Coefficient Mode

In this mode, one multiplier operand derives from the pre-adder, and the other operand derives from the internal coefficient storage. The coefficient storage allows up to 8 preset constants. The coefficient selection signals are `coefsel[0..3]`.

The following settings are applied in this mode:

- The width of the `dataa[]` input (`WIDTH_A`) must be less than or equals to 25 bits
- The width of the `datab[]` input (`WIDTH_B`) must be less than or equals to 25 bits
- The width of the coefficient input must be less than or equals to 27 bits

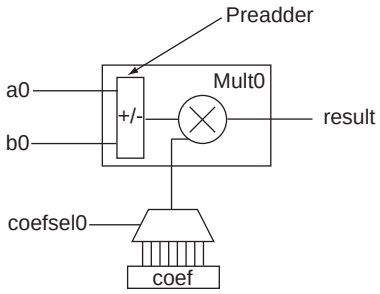
This mode is expressed in Equation 2.

**Equation 2. Equation for Pre-adder Coefficient Mode**

$$y = (a + b) \times coef$$

Figure 24 shows the pre-adder coefficient mode of a multiplier.

**Figure 24. Pre-adder Coefficient Mode**




### Pre-adder Input Mode

In this mode, one multiplier operand derives from the pre-adder, and the other operand derives from the `datac[]` input port.

The following settings are applied in this mode:

- The width of the `dataa[]` input (`WIDTH_A`) must be less than or equals to 25 bits
- The width of the `datab[]` input (`WIDTH_B`) must be less than or equals to 25 bits
- The width of the `datac[]` input (`WIDTH_C`) must be less than or equals to 22 bits
- The number of multipliers must be set to 1
- All input registers must be registered with the same clock

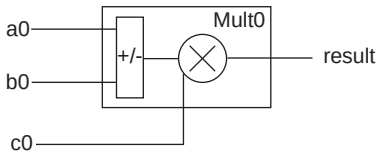
This mode is expressed in Equation 3.

**Equation 3. Equation for Pre-adder Input Mode**

$$y = (a + b) \times c$$

Figure 25 shows the pre-adder input mode of a multiplier.

**Figure 25. Pre-adder Input Mode**




### Pre-adder Square Mode

In this mode, both multiplier operands derive from the pre-adder.

The following settings are applied in this mode:

- The width of the `dataa[]` input (`WIDTH_A`) must be less than or equals to 17 bits
- The width of the `datab[]` input (`WIDTH_B`) must be less than or equals to 17 bits
- The number of multipliers must be set to 2

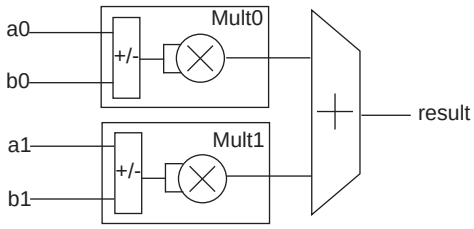
This mode is expressed in Equation 4.

**Equation 4. Equation for Pre-adder Square Mode**

$$y = (a0 + b0)^2 + (a1 + b1)^2$$

Figure 26 shows the pre-adder square mode of two multipliers.

**Figure 26. Pre-adder Square Mode**




### Pre-adder Constant Mode

In this mode, one multiplier operand derives from the input port, and the other operand derives from the internal coefficient storage. The coefficient storage allows up to 8 preset constants. The coefficient selection signals are `coefsel[0..3]`.

The following settings are applied in this mode:

- The width of the `dataa[]` input (`WIDTH_A`) must be less than or equals to 27 bits
- The width of the coefficient input must be less than or equals to 27 bits

- The `datab[]` port must be disconnected

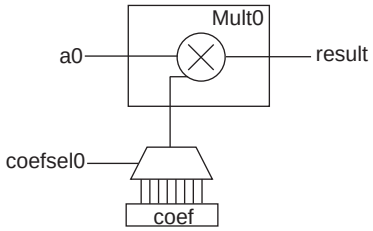
This mode is expressed in Equation 5.

**Equation 5. Equation for Constant Mode**

$$y = a0 \times coef$$

Figure 27 shows the pre-adder constant mode of a multiplier.

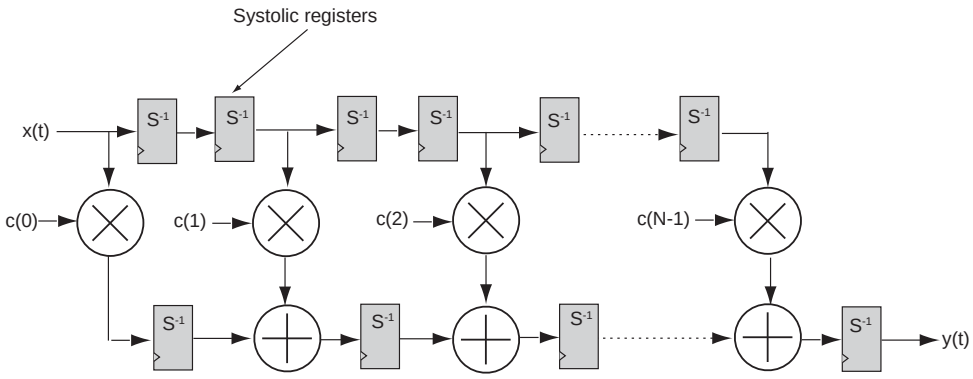
**Figure 27. Pre-adder Constant Mode**




## Systolic Delay Register

In a systolic architecture, the input data is fed into a cascade of registers acting as a data buffer. Each register delivers an input sample to a multiplier where it is multiplied by the respective coefficient. The chain adder stores the gradually combined results from the multiplier and the previously registered result from the `chainin[]` input port to form the final result. Each multiply-add element must be delayed by a single cycle so that the results synchronize appropriately when added together. Each successive delay is used to address both the coefficient memory and the data buffer of their respective multiply-add elements. For example, a single delay for the second multiply add element, two delays for the third multiply-add element, and so on.

**Figure 28. Systolic Registers**



$x(t)$ represents the results from a continuous stream of input samples and $y(t)$ represents the summation of a set of input samples, and in time, multiplied by their respective coefficients. Both the input and output results flow from left to right. The $c(0)$ to $c(N\text{-}1)$ denotes the coefficients. The systolic delay registers are denoted by $S^{-1}$, whereas the $^{-1}$ represents a single clock delay. Systolic delay registers are added at the inputs and outputs for pipelining in a way that ensures the results from the multiplier operand and the accumulated sums stay in synch. This processing element is replicated to form a circuit that computes the filtering function. This function is expressed in Equation 6.

$N$ represents the number of cycles of data that has entered into the accumulator, $y(t)$ represents the output at time $t$, $A(t)$ represents the input at time $t$, and $B(i)$ are the coefficients. The $t$ and $i$ in the equation correspond to a particular instant in time, so to compute the output sample $y(t)$ at time $t$, a group of input samples at $N$ different points in time, or A(n), A(n-1), A(n-2), … A(n-N+1) is required. The group of N input samples are multiplied by $N$ coefficients and summed together to form the final result $y$.

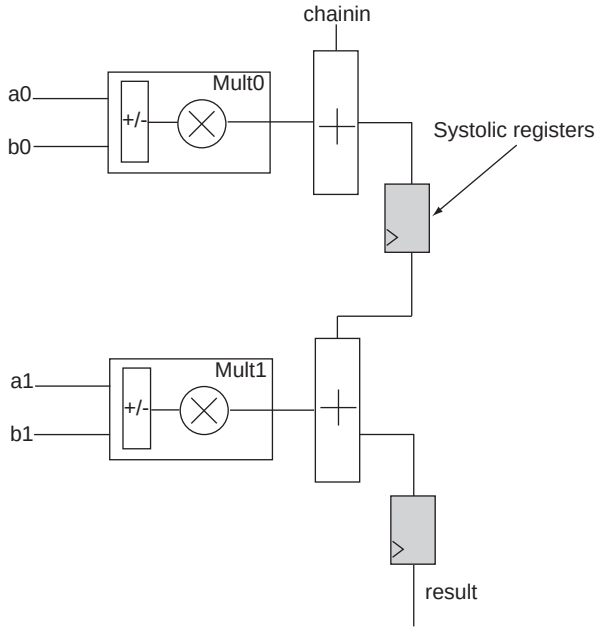
**Equation 6. Equation for Filtering Function**

$$y(t) = \sum_{i=0}^{N-1} B(i)A(t-i)$$

The systolic register architecture is available only for sum-of-2 and sum-of-4 modes.

Figure 29 shows the systolic delay register implementation of 2 multipliers.

**Figure 29. Systolic Delay Register Implementation of 2 Multipliers**



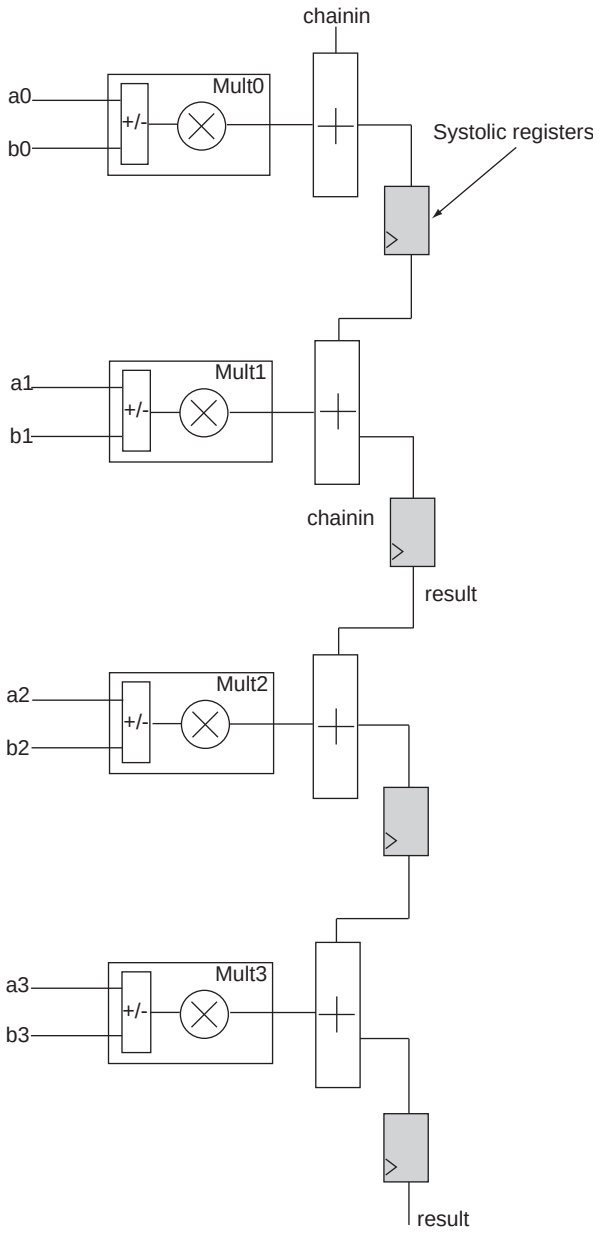
The sum of two multipliers is expressed in Equation 7.

**Equation 7. Sum of 4 Multipliers**

$$y(t) = [a1(t) \times b1(t)] + [a0(t-1) \times b0(t-1)]$$

Figure 30 shows the systolic delay register implementation of 4 multipliers.

**Figure 30. Systolic Delay Register Implementation of 4 Multipliers**



The sum of four multipliers is expressed in Equation 7.

**Equation 8. Sum of 4 Multipliers**

$$y(t) = [a3(t) \times b3(t)] + [a2(t-1) \times b2(t-1)] + [a1(t-2) \times b1(t-2)] + [a0(t-3) \times b0(t-3)]$$

The following lists the advantages of systolic register implementation:

- Reduces DSP resource usage
- Enables efficient mapping in the DSP block using the chain adder structure

The systolic delay implementation is only available for the following pre-adder modes:

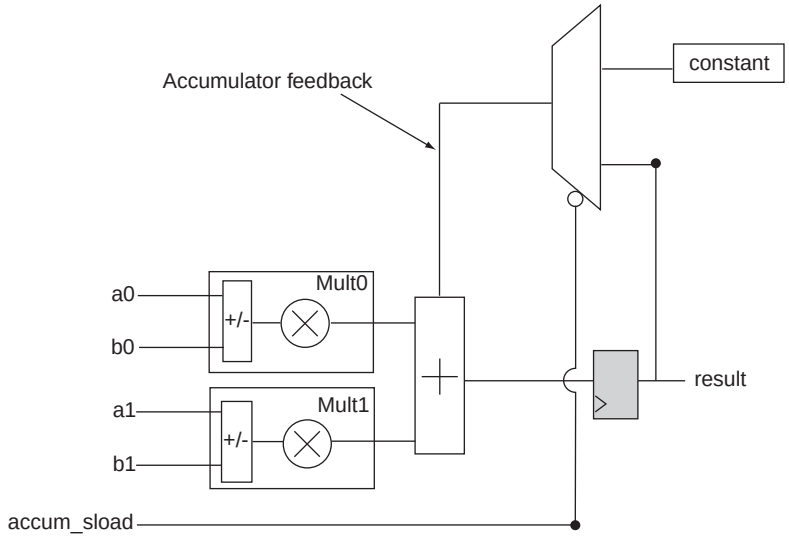
- Pre-adder coefficient mode
- Pre-adder simple mode
- Pre-adder constant mode

## Pre-load Constant

The pre-load constant controls the accumulator operand and complements the accumulator feedback. The valid LOADCONST_VALUE ranges from 0–64. The constant value is equal to $2^N$, where $N$ = LOADCONST_VALUE. . When the LOADCONST_VALUE is set to 64, the constant value is equal to 0. This function can be used as biased rounding.

Figure 31 shows the pre-load constant implementation.

**Figure 31. Pre-load Constant**




Refer to the following megafunctions in this user guide for other multiplier implementations:

- Multiplier-Accumulator Megafunction (ALTMULT_ACCUM)
- Memory-based Constant Coefficient Multiplier (ALTMEMMULT)

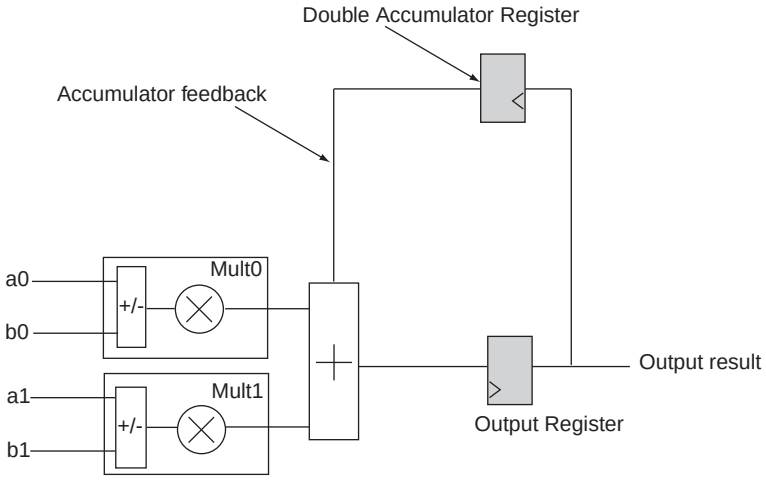
- Multiplier Megafunction (LPM_MULT)

### Double Accumulator

The double accumulator feature adds an additional register in the accumulator feedback path. The double accumulator register follows the output register, which includes the clock, clock enable, and aclr. The additional accumulator register returns result with a one-cycle delay. This feature enables you to have two accumulator channels with the same resource count.

Figure 31 shows the double accumulator implementation.

**Figure 32. Double Accumulator**




## Resource Utilization and Performance

Table 55 provides resource utilization and performance information for the ALTMULT_ADD megafunction.

**Table 55. ALTMULT_ADD Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | 18-bit DSP | |
| Stratix III | 16 x 16 | 3 | 0 | 0 | 0 | 2 | 645 |
| | 32 x 32 | 3 | 0 | 0 | 0 | 4 | 454 |
| | 64 x 64 | 3 | 217 | 128 | 146 | 16 | 145 |

**Notes to Table 55:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: Implementing a Simple Finite Impulse Response (FIR) Filter

This design example uses the ALTMULT_ADD megafunction to implement a simple FIR filter as shown in Equation 9. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

**Equation 9.**

$$y(t) = \sum_{i=0}^{n-1} A(t-i)B(i)$$

In Equation 9, $n$ represents the number of taps, $A(t)$ represents the sequence of input samples, and $B(i)$ represents the filter coefficients.

The number of taps ($n$) can be any value, but this example is of a simple FIR filter with $n = 4$, which is called a 4-tap filter. To implement this filter, the coefficients of data $B$ is loaded into the $B$ registers in parallel and a `shiftin` register moves data $A(0)$ to $A(1)$ to $A(2)$, and so on. With a 4-tap filter, at a given time ($t$), the sum of four products is computed. This function is implemented using the shift register chain option in the ALTMULT_ADD megafunction.

With reference to Equation 9, input $B$ represents the coefficients and data $A$ represents the data that is shifted into. The $A$ input (data) is shifted in with the main clock, named `clock0`. The $B$ input (coefficients) is loaded at the rising edge of `clock1` with the enable signal held high.

### Design Files

The following design files can be found in **altmult_add_DesignExample.zip**:

- **fir_fourtap.qar** (archived Quartus II design files)
- **altmult_add_ex_msim** (ModelSim-Altera files)

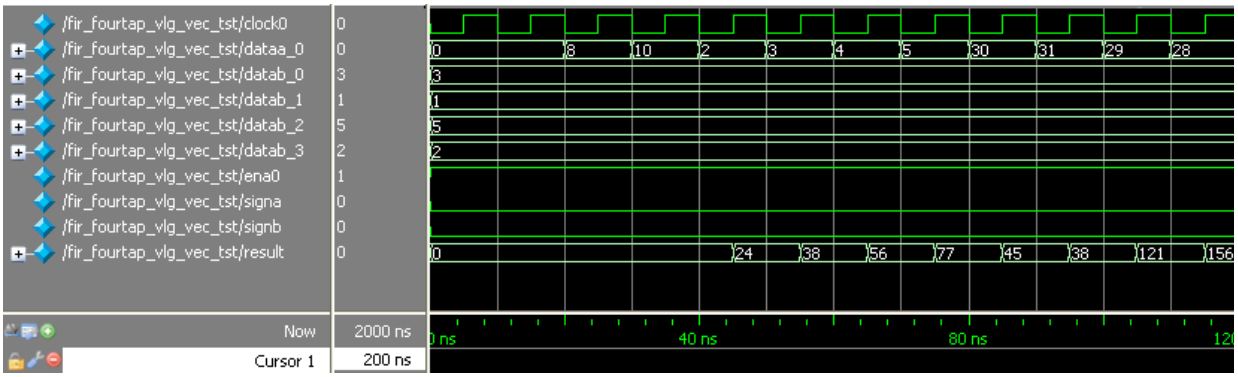
## Understanding the Simulation Results

The following settings are observed in this example:

- The widths of the data inputs are all set to 16 bits
- The width of the output port, `result[]`, is set to 34 bits
- The input registers are all operating on the same clock

Figure 33 shows the expected simulation results in the ModelSim-Altera software.

**Figure 33. ALTMULT_ADD Simulation Results**




## Specifications

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

### Verilog HDL Prototype

To view the Verilog HDL prototype for ALTMULT_ADD megafunction, refer to the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

### VHDL Component Declaration

To view the VHDL component declaration for ALTMULT_ADD megafunction, refer to 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 56 and Table 57 list the input and output ports for the ALTMULT_ADD megafunction.

For Arria V, Cyclone V, and Stratix V devices, each register can only select between 2 asynchronous signals (ACLR0, ACLR1) and 3 clock/enable pairs (CLOCK0/ENA0, CLOCK1/ENA1, CLOCK2/ENA2).

**Table 56. ALTMULT_ADD Megafunction Input Ports (Part 1 of 2)**

| Port Name | Required | Description |
|---|---|---|
| dataa[] | Yes | Data input to the multiplier. Input port [NUMBER_OF_MULTIPLIERS * WIDTH_A - 1..0] wide. |
| datab[] | Yes | Data input to the multiplier. Input port [NUMBER_OF_MULTIPLIERS * WIDTH_B - 1..0] wide. |
| clock[] | No | Clock input port [0..3] to the corresponding register. This port can be used by any register in the megafunction. |
| aclr[] | No | Input port [0..3]. Asynchronous clear input to the corresponding register. |
| ena[] | No | Input port [0..3]. Clock enable for the corresponding clock[] port. |
| signa | No | Specifies the numerical representation of the dataa[] port. If the signa port is high, the multiplier treats the dataa[] port as a signed two's complement number. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number. |
| signb | No | Specifies the numerical representation of the datab[] port. If the signb port is high, the multiplier treats the datab[] port as a signed two's complement number. If the signb port is low, the multiplier treats the datab[] port as an unsigned number. |
| **Ports Available in Stratix II devices only** | | |
| scanina[] | No | Input for scan chain A. Input port [WIDTH_A - 1..0] wide. When the INPUT_SOURCE_A parameter has a value of SCANA or VARIABLE, the scanina[] port is required. Do not use scanina[] and scaninb[] simultaneously. |
| scaninb[] | No | Input for scan chain B. Input port [WIDTH_B - 1..0] wide. When the INPUT_SOURCE_A parameter has a value of SCANB or VARIABLE, the scaninb[] port is required. Do not use scanina[] and scaninb[] simultaneously. |
| sourcea | No | Input source for scan chain A. |
| sourceb | No | Input source for scan chain B. |
| addnsub1 | No | Controls the functionality of the adder. If the addnsub1 port is high, the adder performs an add function. If the addnsub1 port is low, the adder performs a subtract function. |
| addnsub1_round | No | Enables addition or subtraction for the second multiplier. |
| addnsub3 | No | Controls the functionality of the adder. If the addnsub3 port is high, the adder performs an add function. If the addnsub3 port is low, the adder performs a subtract function. |
| addnsub3_round | No | Enables addition or subtraction for the fourth multiplier. |
| mult[]_round | No | Enables rounding for the first and second multiplier [01], or the third and fourth multiplier [23]. Port is required when the corresponding MULTIPLIER[]_ROUNDING parameter has a value of VARIABLE. |

**Table 56. ALTMULT_ADD Megafunction Input Ports (Part 2 of 2)**

| Port Name | Required | Description |
|---|---|---|
| mult[]_saturation | No | Enables saturation for the first and second multiplier [01], or the third and fourth multiplier [23]. Port is required when the corresponding MULTIPLIER[]_SATURATION parameter has a value of VARIABLE. |
| **Ports Available in Stratix III and Stratix IV devices only** | | |
| output_round | No | Enables dynamically controlled output rounding. When OUTPUT_ROUNDING is set to VARIABLE, output_round enables the final adder stage of rounding. |
| output_saturate | No | Enables dynamically controlled output saturation. When OUTPUT_SATURATION is set to VARIABLE, output_saturate enables the final adder stage of saturation. |
| chainout_round | No | Enables dynamically controlled chainout stage rounding. When CHAINOUT_ROUNDING is set to VARIABLE, chainout_round enables the chainout stage of rounding. |
| chainout_saturate | No | Enables dynamically controlled chainout stage saturation. When CHAINOUT_SATURATION is set to VARIABLE, chainout_saturate enables the chainout stage of saturation. |
| zero_chainout | No | Dynamically specifies whether the chainout value is zero. |
| zero_loopback | No | Dynamically specifies whether the loopback value is zero. |
| accum_sload | No | Dynamically specifies whether the accumulator value is zero. |
| chainin | No | Adder result input bus from the preceding stage. Input port [WIDTH_CHAININ - 1..0] wide. |
| rotate | No | Specifies dynamically controlled port rotation in shift mode. |
| shift_right | No | Specifies dynamically controlled port shift right or left in shift mode. Values are 0 and 1. A value of 0 specifies a shift to the left, a value of 1 specifies a shift to the right. |
| **Ports Available in Arria V, Cyclone V, and Stratix V devices only** | | |
| datac[] | Yes | Data input to the multiplier. Input port [NUMBER_OF_MULTIPLIERS * WIDTH_C - 1..0] wide. |
| coefsel0[] | No | Coefficient input port[0..3] to the first multiplier. |
| coefsel1[] | No | Coefficient input port[0..3]to the second multiplier. |
| coefsel2[] | No | Coefficient input port[0..3]to the third multiplier. |
| coefsel3[] | No | Coefficient input port[0..3]to the fourth multiplier. |

**Table 57. ALTMULT_ADD Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Multiplier output port. Output port [WIDTH_RESULT - 1..0] wide. |
| overflow | No | Overflow flag. If output_saturation is enabled, overflow flag is set. |
| scanouta[] | No | Output of scan chain A. Output port [WIDTH_A - 1..0] wide. When designing with Stratix III devices, port cannot be selected when scaninb[] is in use.<br>Do not use scanina[] and scaninb[] simultaneously. |
| scanoutb[] | No | Output of scan chain B. Output port [WIDTH_B - 1..0] wide. When designing with Stratix III devices, port cannot be selected when scanina[] is in use.<br>Do not use scanina[] and scaninb[] simultaneously. |

**Table 57. ALTMULT_ADD Megafunction Output Ports**

| | | |
|---|---|---|
| **Ports Available in Stratix II devices only** | | |
| mult0_is_saturated | No | Signal indicating saturation of the first multiplier. This port is required when PORT_MULT0_IS_SATURATED has a value of USED. |
| mult1_is_saturated | No | Signal indicating saturation of the second multiplier. This port is required when PORT_MULT1_IS_SATURATED has a value of USED. |
| mult2_is_saturated | No | Signal indicating saturation of the third multiplier. This port is required when PORT_MULT2_IS_SATURATED has a value of USED. |
| mult3_is_saturated | No | Signal indicating saturation of the fourth multiplier. This port is required when PORT_MULT3_IS_SATURATED has a value of USED. |
| **Ports Available in Stratix III and Stratix IV devices only** | | |
| chainout_sat_overflow | No | Overflow flag for the chainout saturation. |

Table 58 lists the ALTMULT_ADD megafunction parameters.

For Stratix III, Stratix IV, and Arria II GX devices, when the output result is > 36 bits (for example, when you set width_a=18 and width_b=18), the option for rounding and saturation is disabled. This is because additional logic is used to generate the MSB.

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 1 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| NUMBER_OF_MULTIPLIERS | Integer | Yes | Number of multipliers to be added together. Values are 1 up to 4. |
| WIDTH_A | Integer | Yes | Width of the dataa[] port. |
| WIDTH_B | Integer | Yes | Width of the datab[] port. |
| WIDTH_RESULT | Integer | Yes | Width of the result[] port. Value includes all bits before rounding and saturation. |
| INPUT_REGISTER_A0 | String | No | Specifies the clock port for the dataa[] operand of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, INPUT_REGISTER_A0 must have similar values with INPUT_REGISTER_A[1..3]. |
| INPUT_REGISTER_A1 | String | No | Specifies the clock port for the dataa[] operand of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_A[1..3] must be set similar to the value of INPUT_REGISTER_A0. |
| INPUT_REGISTER_A2 | String | No | Specifies the clock port for the dataa[] operand of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_A[1..3] must be set similar to the value of INPUT_REGISTER_A0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 2 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_REGISTER_A3 | String | No | Specifies the clock port for the dataa[] operand of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_A[1..3] must be set similar to the value of INPUT_REGISTER_A0. |
| INPUT_REGISTER_B0 | String | No | Specifies the clock port for the datab[] operand of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, INPUT_REGISTER_B0 must have similar values with INPUT_REGISTER_B[1..3]. |
| INPUT_REGISTER_B1 | String | No | Specifies the clock port for the datab[] operand of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_B[1..3] must be set similar to the value of INPUT_REGISTER_B0. |
| INPUT_REGISTER_B2 | String | No | Specifies the clock port for the datab[] operand of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_B[1..3] must be set similar to the value of INPUT_REGISTER_B0. |
| INPUT_REGISTER_B3 | String | No | Specifies the clock port for the datab[] operand of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. For Stratix III and Stratix V devices, the values for INPUT_REGISTER_B[1..3] must be set similar to the value of INPUT_REGISTER_B0. |
| INPUT_ACLR_A0 | String | No | Specifies the asynchronous clear for the dataa[] operand of the first multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_A[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_A0 value must be set similar to the values of INPUT_ACLR_A[1..3]. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 3 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_ACLR_A1 | String | No | Specifies the asynchronous clear for the dataa[] operand of the second multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_A[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_A1 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_A2 | String | No | Specifies the asynchronous clear for the dataa[] operand of the third multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_A[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_A2 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_A3 | String | No | Specifies the asynchronous clear for the dataa[] operand of the fourth and corresponding multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_A[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_A3 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_B0 | String | No | Specifies the asynchronous clear for the datab[] operand of the first multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_B[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_B0 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_B1 | String | No | Specifies the asynchronous clear for the datab[] operand of the second multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_B[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_B1 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_B2 | String | No | Specifies the asynchronous clear for the datab[] operand of the third multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_B[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_B2 value must be set similar to the value of INPUT_ACLR_A0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 4 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_ACLR_B3 | String | No | Specifies the asynchronous clear for the datab[] operand of the fourth and corresponding multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_B[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0.<br>The INPUT_ACLR_B3 value must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_SOURCE_A0 | String | No | Specifies the data source to the first multiplier. Values are DATAA and SCANA. If this parameter is set to DATAA, the adder uses the values from the dataa[] port. If this parameter is set to SCANA, the adder uses values from the scan chain. If omitted, the default value is DATAA.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. |
| INPUT_SOURCE_A1 | String | No | Specifies the data source to the second multiplier. Values are DATAA and SCANA. If this parameter is set to DATAA, the adder uses the values from the dataa[] port. If this parameter is set to SCANA, the adder uses values from the scan chain. If omitted, the default value is DATAA.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. |
| INPUT_SOURCE_A2 | String | No | Specifies the data source to the third multiplier. Values are DATAA and SCANA. If this parameter is set to DATAA, the adder uses the values from the dataa[] port. If this parameter is set to SCANA, the adder uses values from the scan chain. If omitted, the default value is DATAA.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 5 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_SOURCE_A3 | String | No | Specifies the data source to the fourth and corresponding multiplier. Values are DATAA and SCANA. If this parameter is set to DATAA, the adder uses the values from the dataa[] port. If this parameter is set to SCANA, the adder uses values from the scan chain. If omitted, the default value is DATAA.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. |
| INPUT_SOURCE_B0 | String | No | Specifies the data source of the first multiplier. Values are DATAB and SCANB. If this parameter is set to DATAB, then the adder uses the values from the datab[] port. If this parameter is set to SCANB, then the adder uses values from the scan chain. If omitted, the default value is DATAB.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Stratix III and Stratix V devices in sum 2 (sum of two) mode, a value of LOOPBACK is available.<br>For Arria V, Cyclone V, and Stratix VV devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. Arria V, Cyclone V, and Stratix V devices also support values of DATAC and COEF. |
| INPUT_SOURCE_B1 | String | No | Specifies the data source of the second multiplier. Values are DATAB and SCANB. If this parameter is set to DATAB, then the adder uses the values from the datab[] port. If this parameter is set to SCANB, then the adder uses values from the scan chain. If omitted, the default value is DATAB.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Stratix III and Stratix V devices in sum 2 (sum of two) mode, a value of LOOPBACK is available.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. Stratix V devices also support values of COEF. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 6 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_SOURCE_B2 | String | No | Specifies the data source of the third multiplier. Values are DATAB and SCANB. If this parameter is set to DATAB, then the adder uses the values from the datab[] port. If this parameter is set to SCANB, then the adder uses values from the scan chain. If omitted, the default value is DATAB.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Stratix III and Stratix V devices in sum 2 (sum of two) mode, a value of LOOPBACK is available.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. Stratix V devices also support values of DATAC and COEF. |
| INPUT_SOURCE_B3 | String | No | Specifies the data source of the fourth and corresponding multiplier. Values are DATAB and SCANB. If this parameter is set to DATAB, then the adder uses the values from the datab[] port. If this parameter is set to SCANB, then the adder uses values from the scan chain. If omitted, the default value is DATAB.<br>For Stratix II devices, a value of VARIABLE is available for the adder to perform rounding and saturation on the data source before feeding the result to the multiplier.<br>For Stratix III and Stratix V devices in sum 2 (sum of two) mode, a value of LOOPBACK is available.<br>For Arria V, Cyclone V, and Stratix V devices, a value of PREADDER is available for the adder to perform addition or subtraction on the data source before feeding the result to the multiplier. Stratix V devices also support values of COEF. |
| REPRESENTATION_A | String | No | Specifies the numerical representation of the multiplier input A. Values are UNSIGNED, SIGNED and VARIABLE. When this parameter is set to SIGNED, the adder interprets the multiplier input A as a signed two's complement number. When this parameter is set to UNSIGNED, the adder interprets the multiplier input A as an unsigned number. If omitted, the default value is UNSIGNED. Use the VARIABLE setting to access the SIGNED_REGISTER_A and the SIGNED_PIPELINE_REGISTER_A parameter options for the signa input port.<br>For Arria V, Cyclone V, and Stratix V devices, a value of UNUSED is available for dynamic control of the representation through the signa input port. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 7 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| REPRESENTATIONS_B | String | No | Specifies the numerical representation of the multiplier input B port. Values are UNSIGNED, SIGNED, and VARIABLE. When this parameter is set to UNSIGNED, the adder interprets the multiplier input B as an unsigned number. When this parameter is set to SIGNED, the adder interprets the multiplier input B as a signed two's complement number. If omitted, the default value is UNSIGNED. Use the VARIABLE setting to access the SIGNED_REGISTER_B and the SIGNED_PIPELINE_REGISTER_B parameter options for the signb input port.<br>For Arria V, Cyclone V, and Stratix V devices, a value of UNUSED is available for dynamic control of the representation through the signb input port. |
| SIGNED_REGISTER_[] | String | No | Parameter [A,B]. Specifies the clock signal for the first register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If the corresponding sign[] port value is UNUSED, this parameter is ignored.<br>If omitted, the default value is CLOCK0.<br>For Arria V, Cyclone V, and Stratix V devices, the value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| SIGNED_PIPELINE_ REGISTER_[] | String | No | Parameter [A,B]. Specifies the clock signal for the second register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If the corresponding sign[] port value is UNUSED, this parameter is ignored.<br>If omitted, the default value is CLOCK0. |
| SIGNED_ACLR_[] | String | No | Parameter [A,B]. Specifies the asynchronous clear signal for the first register on the corresponding sign[] port. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding SIGNED_REGISTER_[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| SIGNED_PIPELINE_ACLR_[] | String | No | Parameter [A,B]. Specifies the asynchronous clear signal for the second register on the corresponding sign[] port. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and the corresponding SIGNED_PIPELINE_REGISTER_[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix VV devices, the default value is ACLR0. The value must be set similar to the value of MULTIPLIER_ACLR0. |
| MULTIPLIER_REGISTER[] | String | No | Parameter [0..3]. Specifies the clock source of the register that follows the corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 8 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MULTIPLIER_ACLR[] | String | No | Parameter [0..3]. Specifies the asynchronous clear signal of the register that follows the corresponding multiplier. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding MULTIPLIER_REGISTER[] is used, the default value is ACLR3. For Arria V, Cyclone V, and Stratix V devices, the default value is ACLR0. |
| MUTIPLIER1_DIRECTION | String | No | Specifies whether the second multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If the addnsub1 port is used, this parameter is ignored.<br>If omitted, the default value is ADD. |
| MUTIPLIER3_DIRECTION | String | No | Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are ADD and SUB.<br>If the addnsub3 port is used, this parameter is ignored. If omitted, the default value is ADD. |
| ACCUM_DIRECTION | String | No | Specifies whether to use the accumulator and whether the accumulator adds or subtracts its value from the sum. Values are ADD and SUB.<br>If omitted, the default value is ADD. |
| OUTPUT_REGISTER | String | No | Specifies the clock signal for the second adder register. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. |
| OUTPUT_ACLR | String | No | Specifies the asynchronous clear signal for the second adder register. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted, the default value is ACLR3. |
| PORT_SIGN[] | String | No | Parameter [A,B]. Specifies the corresponding sign[] input port usage. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. If omitted, the default value is PORT_CONNECTIVITY. |
| CHAINOUT_ROUND_TYPE | String | No | Specifies the rounding mode at the chainout stage. Values are BIASED and UNBIASED. A value of BIASED specifies round-to-nearest-integer. A value of UNBIASED specifies round-to-nearest-even. |
| EXTRA_LATENCY | String | No | Specifies the number of clock cycles of latency. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTMULT_ADD megafunction with the MegaWizard Plug-in Manager to calculate the value for this parameter. |
| DSP_BLOCK_BALANCING | String | No | If omitted, the default value is AUTO. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 9 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| DEDICATED_MULTIPLIER_CIRCUITRY | String | No | Specifies whether to use the DSP block to implement the circuit. Values are YES, NO, and AUTO. The circuit is implemented using the DSP block when the value is set to YES.<br>If omitted, the default value is AUTO. |
| **Parameters Available in Stratix II devices only** | | | |
| ADDNSUB_MULTIPLIER_REGISTER[] | String | No | Parameter [1,3]. Specifies the clock signal for the first register on the corresponding addnsub[] input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If the corresponding addnsub[] port is UNUSED, this parameter is ignored.<br>If omitted, the default value is CLOCK0. |
| ADDSUB_MULTIPLIER_ACLR[] | String | No | Parameter [1,3]. Specifies the asynchronous clear signal for the first register on the corresponding addnsub[] input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If the corresponding addnsub[] port value is UNUSED, this parameter is ignored.<br>If omitted and corresponding ADDNSUB_MULTIPLIER_REGISTER[] is used, the default value is ACLR3. |
| ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[] | String | No | Parameter [1,3]. Specifies the clock signal for the second register on the corresponding addnsub[] input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If the corresponding addnsub[] port is UNUSED, this parameter is ignored.<br>If omitted, the default value is CLOCK0. |
| ADDNSUB_MULTIPLIER_PIPELINE_ACLR[] | String | No | Parameter [1,3]. Specifies the asynchronous clear signal for the second register on the corresponding addnsub[] input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and corresponding ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[] is used, the default value is ACLR3. |
| PORT_ADDNSUB[] | String | No | Parameter [1,3]. Specifies the usage of the corresponding addnsub[] input port. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. A value of PORT_CONNECTIVITY specifies the port usage by checking port connectivity.<br>If omitted, the default value is PORT_CONNECTIVITY. |
| MULTIPLIER[]_ROUNDING | String | No | Parameter [01,23]. Specifies rounding for the first and second multiplier [01], or the third and fourth multiplier [23]. Values are NO, YES, and VARIABLE.<br>If omitted, the default value is NO. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 10 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MULTIPLIER[]_SATURATION | String | No | Parameter [01,23]. Specifies saturation for the first and second multiplier [01], or the third and fourth multiplier [23]. Values are NO, YES, and VARIABLE.<br>If omitted, the default value is NO. |
| ADDER[]_ROUNDING | String | No | Parameter [1,3]. Specifies adder rounding for the first multiplier [1], or the third multiplier [3]. Values are NO, YES, and VARIABLE.<br>If omitted, the default value is NO. |
| PORT_MULT[]_IS_SATURATED | String | No | Parameter [0..3]. Specifies whether to use the corresponding mult[]_is_saturated output port. Values are NO and YES.<br>If omitted, the default value is NO. |
| WIDTH_MSB | Integer | No | Specifies the fractional rounding width. The value is determined by counting the bits from the MSB (before saturation) to the LSB (after rounding). Values are calculated according to the following modes: WIDTH_A, WIDTH_B, and WIDTH_RESULT. Value must be an unsigned integer, and must be less than WIDTH_RESULT.<br>If a positive number is unavailable, no saturation is allowed in your input/output width and mode setting.<br>If omitted, the default value is 17, which is compatible with Stratix II device settings. |
| ADDNSUB[]_ROUND_ACLR | String | No | Parameter [1,3]. Specifies the asynchronous clear source for the first register on the corresponding addnsub[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and corresponding ADDNSUB[]_ ROUND_REGISTER is used, the default value is ACLR3. |
| ADDNSUB[]_ROUND_ PIPELINE_ACLR | String | No | Parameter [1,3]. Specifies the asynchronous clear source for the second register on the corresponding addnsub[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and corresponding ADDNSUB_[]_ ROUND_PIPELINE_REGISTER is used, the default value is ACLR3. |
| ADDNSUB[]_ROUND_ PIPELINE_REGISTER | String | No | Parameter [1, 3]. Specifies the clock source for the second register on the corresponding addnsub[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ADDNSUB[]_ROUND_REGISTER | String | No | Parameter [1,3]. Specifies the clock source for the first register on the corresponding addnsub[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 11 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| MULT[]_ROUND_ACLR | String | No | Parameter [01,23]. Specifies the asynchronous clear source for the second register on the corresponding mult[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and corresponding MULT[]_ROUND_REGISTER is used, the default value is ACLR3. |
| MULT[]_ROUND_REGISTER | String | No | Parameter [01,23]. Specifies the clock source for the register on the corresponding mult[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| MULT[]_SATURATION_ACLR | String | No | Parameter [01,23]. Specifies the asynchronous clear source for the register on the corresponding mult[]_saturation input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and corresponding MULT[]_ SATURATION_REGISTER is used, the default value is ACLR3. |
| MULT[]_SATURATION_REGISTER | String | No | Parameter [01,23]. Specifies the clock source for the register on the corresponding mult[]_saturation input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| **Parameters Available in Stratix II, Stratix III, and Stratix IV devices only** | | | |
| OUTPUT_SATURATE_TYPE | String | No | Specifies the saturation mode. Values are SYMMETRIC and ASYMMETRIC. A value of SYMMETRIC specifies the absolute value of the maximum negative number equal to the maximum positive number. A value of ASYMMETRIC specifies the maximum negative number is larger than the maximum positive number.<br>If omitted, the default value is ASYMMETRIC. |
| WIDTH_SATURATE_SIGN | String | No | Specifies the saturation position. The value is determined by counting the bits that become the sign bits after saturation. Values are calculated according to the following modes— WIDTH_A, WIDTH_B, and WIDTH_RESULT.. Value must be an unsigned integer. If a positive number is unavailable, no saturation is allowed in your input/output width and mode setting.<br>If omitted, the default value is 1. |
| CHAINOUT_ADDER | String | No | Specifies the chainout mode of the final adder stage. Values are YES and NO.<br>If omitted, the default value is NO. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 12 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| **Parameters Available in Stratix II, Stratix III, Stratix IV, and Stratix V devices only** | | | |
| ACCUMULATOR | String | No | Specifies the accumulator mode of the final adder stage. Values are YES and NO.<br>If omitted, the default value is NO.<br>When value is set to YES, rounding is dynamic and you must initialize the accumulator while rounded data is acquired. |
| **Parameters Available in Stratix III and Stratix IV devices only** | | | |
| WIDTH_CHAININ | Integer | No | Width of the chainin[] port. WIDTH_CHAININ equals WIDTH_RESULT if port chainin is used.<br>If omitted, the default value is 1. |
| OUTPUT_ROUNDING | String | No | Enables rounding handling at second adder stage. If original design uses a Stratix II device, in some cases this parameter can be derived from the Stratix II rounding settings. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling. |
| OUTPUT_ROUND_TYPE | String | No | Specifies the rounding mode. Values are NEAREST_EVEN and NEAREST_INTEGER.<br>A value of NEAREST_EVEN specifies round-to-nearest-even.<br>A value of NEAREST_INTEGER specifies round-to-nearest-integer.<br>If omitted, the default value is NEAREST_INTEGER. |
| OUTPUT_ROUND_REGISTER | String | No | Specifies the clock source for the first register on the output_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| OUTPUT_ROUND_ACLR | String | No | Specifies the asynchronous clear source for the first register on the output_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and OUTPUT_ROUND_REGISTER is used, the default value is ACLR3. |
| OUTPUT_ROUND_PIPELINE_ REGISTER | String | No | Specifies the clock source for the second register on the output_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| OUTPUT_ROUND_PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the output_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and OUTPUT_ROUND_PIPELINE_ REGISTER is used, the default value is ACLR3. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 13 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| OUTPUT_SATURATION | String | No | Enables saturation handling at second adder stage. If original design uses a Stratix II device, in some cases this parameter can be derived from the Stratix II rounding settings. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling.<br>If omitted, the default value is NO. |
| OUTPUT_SATURATE_REGISTER | String | No | Specifies the clock source for the first register on the output_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is UNREGISTERED. |
| OUTPUT_SATURATE_ACLR | String | No | Specifies the asynchronous clear source for the first register on the output_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and OUTPUT_SATURATE_REGISTER is used, the default value is ACLR3. |
| OUTPUT_SATURATE_ PIPELINE_REGISTER | String | No | Specifies the clock source for the second register on the output_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| OUTPUT_SATURATE_ PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the output_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and OUTPUT_SATURATE_ PIPELINE_REGISTER is used, the default value is ACLR3. |
| CHAINOUT_ROUNDING | String | No | Enables rounding handling at the chainout stage. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling.<br>Note: If the value of CHAINOUT_ROUNDING is YES, the symmetric saturation at the second adder output stage is not allowed.<br>If omitted, the default value is NO. |
| CHAINOUT_ROUND_REGISTER | String | No | Specifies the clock source for the first register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_ROUND_ACLR | String | No | Specifies the asynchronous clear source for the first register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_ROUND_REGISTER is used, the default value is ACLR3. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 14 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| CHAINOUT_ROUND_ PIPELINE_REGISTER | String | No | Specifies the clock source for the second register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_ROUND_ PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_ROUND_ PIPELINE_REGISTER is used, the default value is ACLR3. |
| CHAINOUT_ROUND_OUTPUT_ REGISTER | String | No | Specifies the clock source for the third register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_ROUND_OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the third register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_ROUND_ OUTPUT_REGISTER is used, the default value is ACLR3. |
| CHAINOUT_SATURATION | String | No | Enables saturation handling at the chainout stage. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling.<br>If omitted, the default value is NO. |
| CHAINOUT_SATURATE_REGISTER | String | No | Specifies the clock source for the first register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_SATURATE_ACLR | String | No | Specifies the asynchronous clear source for the first register on the chainout_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_SATURATE_REGISTER is used, the default value is ACLR3. |
| CHAINOUT_SATURATE_ PIPELINE_REGISTER | String | No | Specifies the clock source for the second register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_SATURATE_ OUTPUT_REGISTER | String | No | Specifies the clock source for the third register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| CHAINOUT_SATURATE_ OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the third register on the chainout_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_SATURATE_ OUTPUT_REGISTER is used, the default value is ACLR3. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 15 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| ZERO_CHAINOUT_OUTPUT_ REGISTER | String | No | Specifies the clock source for the first register on the zero_chainout input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ZERO_CHAINOUT_OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the first register on the zero_chainout input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ZERO_CHAINOUT_OUTPUT_ REGISTER is used, the default value is ACLR3. |
| ZERO_LOOPBACK_REGISTER | String | No | Specifies the clock source for the first register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ZERO_LOOPBACK_ACLR | String | No | Specifies the asynchronous clear source for the first register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ZERO_LOOPBACK_ PIPELINE_REGISTER is used, the default value is ACLR3. |
| ZERO_LOOPBACK_PIPELINE_ REGISTER | String | No | Specifies the clock source for the second register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ZERO_LOOPBACK_PIPELINE_ ACLR | String | No | Specifies the asynchronous clear source for the second register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ZERO_LOOPBACK_ PIPELINE_REGISTER is used, the default value is ACLR3. |
| ZERO_LOOPBACK_OUTPUT_ REGISTER | String | No | Specifies the clock source for the third register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ZERO_LOOPBACK_OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the third register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ZERO_LOOPBACK_OUTPUT_ REGISTER is used, the default value is ACLR3. |
| ACCUM_SLOAD_REGISTER | String | No | Specifies the clock source for the first register on the accum_sload input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ACCUM_SLOAD_ACLR | String | No | Specifies the asynchronous clear source for the first register on the accum_sload input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ACCUM_SLOAD_REGISTER is used, the default value is ACLR3. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 16 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| ACCUM_SLOAD_PIPELINE_ REGISTER | String | No | Specifies the clock source for the second register on the accum_sload input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ACCUM_SLOAD_PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the accum_sload input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ACCUM_SLOAD_PIPELINE_ REGISTER is used, the default value is ACLR3. |
| SHIFT_MODE | String | No | Specifies the shift mode. Values are NO, LEFT, RIGHT, ROTATION, and VARIABLE. If VARIABLE is selected, rotate and shift_right are used to specify shift left, shift right, or rotation.<br>If omitted, the default value is NO.<br>Note that this parameter is supported only when inputs equal 32 bits each, output equals 32 bits, and the number of multipliers equals 1. |
| ROTATE_REGISTER | String | No | Specifies the clock source for the first register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ROTATE_ACLR | String | No | Specifies the asynchronous clear source for the first register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ROTATE_REGISTER is used, the default value is ACLR3. |
| ROTATE_PIPELINE_REGISTER | String | No | Specifies the clock source for the second register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ROTATE_PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ROTATE_PIPELINE_REGISTER is used, the default value is ACLR3. |
| ROTATE_OUTPUT_REGISTER | String | No | Specifies the clock source for the third register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| ROTATE_OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the third register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and ROTATE_OUTPUT_REGISTER is used, the default value is ACLR3. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 17 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| SHIFT_RIGHT_REGISTER | String | No | Specifies the clock source for the first register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| SHIFT_RIGHT_ACLR | String | No | Specifies the asynchronous clear source for the first register on the shift_right input. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and SHIFT_RIGHT_REGISTER is used, the default value is ACLR3. |
| SHIFT_RIGHT_PIPELINE_ REGISTER | String | No | Specifies the clock source for the second register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| SHIFT_RIGHT_PIPELINE_ACLR | String | No | Specifies the asynchronous clear source for the second register on the shift_right input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and SHIFT_RIGHT_PIPELINE_ REGISTER is used, the default value is ACLR3. |
| SHIFT_RIGHT_OUTPUT_ REGISTER | String | No | Specifies the clock source for the third register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3.<br>If omitted, the default value is CLOCK0. |
| SHIFT_RIGHT_OUTPUT_ACLR | String | No | Specifies the asynchronous clear source for the third register on the shift_right input. Values are ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and SHIFT_RIGHT_OUTPUT_REGISTER is used, the default value is ACLR3. |
| PORT_OUTPUT_IS_OVERFLOW | String | No | Specifies port usage. Values are PORT_UNUSED and PORT_USED. When the value is set to PORT_USED, output pin overflow is added.<br>If omitted, the default value is PORT_UNUSED. |
| PORT_CHAINOUT_SAT_IS_ OVERFLOW | String | No | Specifies port usage. Values are PORT_UNUSED and PORT_USED. When the value is set to PORT_USED, output pin chainout_sat_overflow is added.<br>If omitted, the default value is PORT_UNUSED. |
| **Parameters Available in Stratix III, Stratix IV, and Stratix V devices only** | | | |
| SCANOUTA_REGISTER | String | No | Specifies the clock source for the scanouta data bus registers. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is UNREGISTERED. |
| SCANOUTA_ACLR | String | No | Specifies the asynchronous clear source for the scanouta data bus registers. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and SCANOUTA_REGISTER is used, the default value is ACLR3. For Stratix V devices, the default value is ACLR0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 18 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| CHAINOUT_REGISTER | String | No | Specifies the clock source for the chainout mode result register. This is an additional stage after the second adder. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. For Stratix V devices, this parameter adds additional latency before the output register. |
| CHAINOUT_ACLR | String | No | Specifies the asynchronous clear for the chainout mode result register. This is an additional stage after the second adder. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3.<br>If omitted and CHAINOUT_REGISTER is used, the default value is ACLR3. |
| **Parameters Available in Arria V, Cylcone V, and Stratix V devices only** | | | |
| WIDTH_C | Integer | Yes | Width of the datac[] port. |
| WIDTH_COEF | Integer | Yes | Specifies the width of the coefsel[] port. |
| INPUT_REGISTER_C0 | String | No | Specifies the clock port for the datac[] operand of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C1 | String | No | Specifies the clock port for the datac[] operand of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C2 | String | No | Specifies the clock port for the datac[] operand of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_REGISTER_C3 | String | No | Specifies the clock port for the datac[] operand of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2.<br>If omitted, the default value is CLOCK0. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| INPUT_ACLR_C0 | String | No | Specifies the asynchronous clear for the datac[] operand of the first multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C0 must be set similar to the value of INPUT_ACLR_A0. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 19 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| INPUT_ACLR_C1 | String | No | Specifies the asynchronous clear for the datac[] operand of the second multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C1 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_C2 | String | No | Specifies the asynchronous clear for the datac[] operand of the third multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C2 must be set similar to the value of INPUT_ACLR_A0. |
| INPUT_ACLR_C3 | String | No | Specifies the asynchronous clear for the datac[] operand of the fourth and corresponding multiplier. Values are ACLR0 and ACLR1.<br>If omitted and corresponding INPUT_REGISTER_C[] is used, the default value is ACLR0. The value for INPUT_ACLR_C3 must be set similar to the value of INPUT_ACLR_A0. |
| LOADCONST_VALUE | Integer | No | Pre-load constant value to complement accumulator mode. Values are 2^*N* where 0 < *N* < 64. |
| PREADDER_MODE | String | No | Specifies the mode of pre-adder settings to be used. Values are SIMPLE, COEF, INPUT, SQUARE, and CONSTANT. The default value is SIMPLE. |
| PREADDER_DIRECTION_0 | String | No | Specifies whether the pre-adder of the first multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_1 | String | No | Specifies whether the pre-adder of the second multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_2 | String | No | Specifies whether the pre-adder of the third multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| PREADDER_DIRECTION_3 | String | No | Specifies whether the pre-adder of the fourth and corresponding multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default value is ADD. |
| COEFFSEL0_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL1_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the second multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 20 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| COEFFSEL2_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL3_REGISTER | String | No | Specifies the clock source for the coefficient inputs of the fourth and corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of INPUT_REGISTER_A0 or set as UNREGISTERED. |
| COEFFSEL_A_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the first multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_B_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the second multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_C_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the third multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| COEFFSEL_D_ACLR | String | No | Specifies the asynchronous clear source for the coefficient inputs to the fourth and corresponding multiplier. Values are NONE, ACLR0 and ACLR1. If omitted and corresponding COEFFSEL[]_REGISTER is used, the default value is ACLR0. The value must be set similar to the value of INPUT_ACLR_A0. |
| SYSTOLIC_DELAY1 | String | No | Specifies the clock source for the systolic register inputs of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| SYSTOLIC_DELAY3 | String | No | Specifies the clock source for the systolic register inputs of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| SYSTOLIC_ACLR1 | String | No | Specifies the asynchronous clear source for the systolic register inputs of the first multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and corresponding SYSTOLIC_DELAY[]is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |

**Table 58. ALTMULT_ADD Megafunction Parameters (Part 21 of 21)**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| SYSTOLIC_ACLR3 | String | No | Specifies the asynchronous clear source for the systolic register inputs of the third multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and corresponding SYSTOLIC_DELAY[] is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |
| CHAININ_REGISTER1 | String | No | Specifies the chainin register input of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| CHAININ_REGISTER3 | String | No | Specifies the chainin register input of the third multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, and CLOCK2. The value must be set similar to the value of OUTPUT_REGISTER or set as UNREGISTERED. |
| CHAININ_ACLR1 | String | No | Specifies the asynchronous clear source for the chainin register inputs to the first multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and corresponding CHAININ_REGISTER[] is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |
| CHAININ_ACLR3 | String | No | Specifies the asynchronous clear source for the chainin register inputs to the third multiplier. Values are NONE, ACLR0, and ACLR1. If omitted and corresponding CHAININ_REGISTER[] is used, the default value is ACLR0. The value must be set similar to the value of OUTPUT_ACLR. |
| COEF0_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the first multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| COEF1_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the second multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| COEF2_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the third multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| COEF3_[] | Integer | No | Specifies the coefficient value [0..7] for the inputs of the fourth multiplier. The number of coefficient bits must be set similar to the value of WIDTH_COEF. |
| DOUBLE_ACCUM | String | No | Enables the double accumulator register. For more information, refer to "Double Accumulator" on page 83. |

# ALTMULT_COMPLEX (Complex Multiplier)

The ALTMULT_COMPLEX megafunction implements the multiplication of two complex numbers and offers the following two implementation modes:

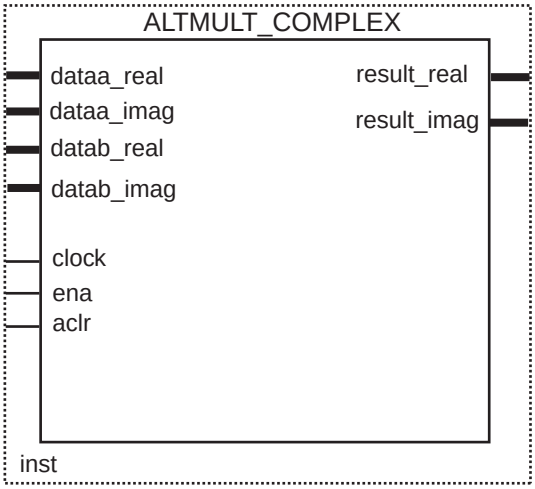
- Canonical
  You can use the canonical representation for the following:
  - All supported Altera devices prior to Stratix III devices. The canonical representation is no longer supported from Stratix III onwards
  - Input data widths of less than 18 bits
- Conventional
  You can use the conventional representation for the following:
  - All supported Altera devices
  - Input data widths of any size

With the conventional representation, you can use the ALTMULT_ADD megafunction to implement the complex multiplier by instantiating two multipliers.

For more information about the ALTMULT_ADD megafunction, refer to the Multiplier-Adder Megafunction section (ALTMULT_ADD).

Figure 34 shows the ports for the ALTMULT_COMPLEX megafunction.

**Figure 34. ALTMULT_COMPLEX Ports Shown in the MegaWizard Plug-In Manager**




## Complex Multiplication

Complex numbers are numbers in the form of the following equation:

$a + ib$

where:

- $a$ and $b$ are real numbers
- $i$ is an imaginary unit that equals the square root of -1: $\sqrt{-1}$

Two complex numbers $x = a + ib$ and $y = c + id$ are multiplied, as shown in Equation 10 and Equation 11:

**Equation 10.**

$$\begin{aligned} xy &= (a + ib)(c + id) \\ &= ac + ibc + iad - bd \end{aligned}$$

**Equation 11.**

$$= (ac - bd) + i(ad + bc)$$

## Canonical Representation

From Equation 10, the multiplication of two complex numbers can be represented in two parts: real and imaginary. The *xy_real* variable in Equation 12 represents the real part, and the *xy_imaginary* variable in Equation 13 represents the imaginary part. Both equations are derived from Equation 11.

**Equation 12.**

$$\begin{aligned} xy\_real &= ac - bd \\ &= ac - bd + (ad - bc) - (ad - bc) \\ &= (ac - ad + bc - bd) + (ad - bc) \\ &= ((a + b)(c - d)) + (ad - bc) \end{aligned}$$

**Equation 13.**

$$xy\_imaginary = ad + bc$$

The canonical representation is available for all supported Altera devices prior to Stratix III devices.

## Conventional Representation

From Equation 10, the multiplication of two complex numbers can be represented in two parts, real and imaginary. The *xy_real* variable in Equation 14 represents the real part, and the *xy_imaginary* variable in Equation 15 represents the imaginary part. Both equations are derived from Equation 11.

**Equation 14.**

$$xy\_real = ac - bd$$

**Equation 15.**

$$xy\_imaginary = ad + bc$$

## Features

The ALTMULT_COMPLEX megafunction offers the following features:

- Generates a multiplier to perform multiplication operations of two complex numbers
- Supports data width of 1–256 bits
- Supports signed and unsigned data representation format
- Supports canonical and conventional implementation modes
- Supports pipelining with configurable output latency
- Supports optional asynchronous clear and clock enable input ports
- Provides an option to dynamically switch between 36 × 36 normal mode and 18 × 18 complex mode (for Stratix V devices only)

## Resource Utilization and Performance

Table 59 provides resource utilization and performance information for the ALTMULT_COMPLEX megafunction.

**Table 59. ALTMULT_COMPLEX Resource Utilization and Performance** *(1)*

| Device family | Input data width | Output latency | Logic Usage | | | | $f_{MAX}$ (MHz) *(2)* |
|---|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | 18-bit DSP | |
| Stratix III | 8 | 0 | 0 | 0 | 0 | 4 | 529 |
| | 16 | 0 | 0 | 0 | 0 | 4 | 531 |
| | 32 | 0 | 73 | 0 | 37 | 16 | 291 |
| | 64 | 0 | 73 | 0 | 37 | 16 | 292 |
| | 8 | 14 | 19 | 10 | 10 | 4 | 492 |
| | 16 | 14 | 19 | 10 | 10 | 4 | 502 |
| | 32 | 14 | 91 | 8 | 47 | 16 | 265 |
| | 64 | 14 | 91 | 8 | 47 | 16 | 268 |
| Stratix IV | 8 | 0 | 0 | 0 | 0 | 4 | 487 |
| | 16 | 0 | 0 | 0 | 0 | 4 | 487 |
| | 32 | 0 | 73 | 0 | 37 | 16 | 293 |
| | 64 | 0 | 73 | 0 | 37 | 16 | 293 |
| | 8 | 14 | 19 | 10 | 10 | 4 | 489 |
| | 16 | 14 | 20 | 10 | 10 | 4 | 493 |
| | 32 | 14 | 91 | 8 | 47 | 16 | 292 |
| | 64 | 14 | 91 | 8 | 47 | 16 | 291 |

**Notes to Table 59:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: Multiplication of 8-bit Complex Numbers Using Canonical Representation

This design example uses the ALTMULT_COMPLEX megafunction to implement a complex multiplier with an 8-bit input data width using the canonical representation. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **altmult_complex_DesignExample.zip**:

- **complex_canonical.qar** (archived Quartus II design files)
- **altmult_complex_ex_msim** (ModelSim-Altera files)

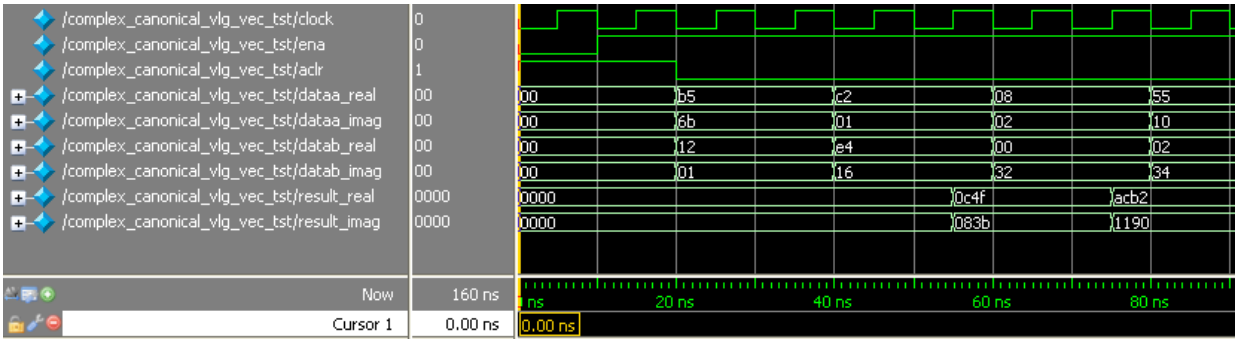
## Understanding the Simulation Results

The following settings are observed in this example:

- The widths of the data inputs are all set to 8 bits
- The widths of the output ports are set to 16 bits
- The asynchronous clear (`aclr`) and clock enable (`ena`) signals are enabled
- Pipelining is enabled with an output latency of four clock cycles. Hence, the result is seen on the output ports four clock cycles after the input data is available

Figure 35 shows the expected simulation results in the ModelSim-Altera software.

**Figure 35. ALTMULT_COMPLEX Simulation Results**



## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altmult_complex
# (parameter intended_device_family = "unused",
parameter implementation_style = "AUTO",
parameter pipeline = 4,
parameter representation_a = "SIGNED",
parameter representation_b = "SIGNED",
parameter width_a = 1,
parameter width_b = 1,
parameter width_result = 1,
parameter lpm_type = "altmult_complex",
parameter lpm_hint = "unused")
(input wire aclr,
input wire clock,
input wire complex,
input wire [width_a-1:0] dataa_imag,
input wire [width_a-1:0] dataa_real,
input wire [width_b-1:0] datab_imag,
input wire [width_b-1:0] datab_real,
input wire ena,
output wire [width_result-1:0] result_imag,
output wire [width_result-1:0] result_real;
endmodule
```

### 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 altmult_complex
generic (
intended_device_family:string := "unused";
implementation_style:string := "AUTO";
pipeline:natural := 4;
representation_a:string := "SIGNED";
representation_b:string := "SIGNED";
width_a:natural;
width_b:natural;
width_result:natural;
lpm_hint:string := "UNUSED";
lpm_type:string := "altmult_complex");
port(
aclr:in std_logic := '0';
clock:in std_logic := '0';
complex:in std_logic := '1';
dataa_imag:in std_logic_vector(width_a-1 downto 0);
dataa_real:in std_logic_vector(width_a-1 downto 0);
datab_imag:in std_logic_vector(width_b-1 downto 0);
datab_real:in std_logic_vector(width_b-1 downto 0);
ena:in std_logic := '1';
result_imag:out std_logic_vector(width_result-1 downto 0);
result_real:out std_logic_vector(width_result-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 60 lists the input ports, Table 61 lists the output ports, and Table 62 lists the ALTMULT_COMPLEX megafunction parameters.

**Table 60. ALTMULT_COMPLEX Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| aclr | No | Asynchronous clear for the complex multiplier. When the aclr port is asserted high, the function is asynchronously cleared. |
| clock | Yes | Clock input to the ALTMULT_COMPLEX function. |
| dataa_imag[] | Yes | Imaginary input value for the data A port of the complex multiplier. The size of the input port depends on the WIDTH_A parameter value. |
| dataa_real[] | Yes | Real input value for the data A port of the complex multiplier. The size of the input port depends on the WIDTH_A parameter value. |
| datab_imag[] | Yes | Imaginary input value for the data B port of the complex multiplier. The size of the input port depends on the WIDTH_B parameter value. |
| datab_real[] | Yes | Real input value for the data B port of the complex multiplier. The size of the input port depends on the WIDTH_B parameter value. |
| ena | No | Active high clock enable for the clock port of the complex multiplier. |
| **Ports Available in Stratix V devices only** | | |
| complex | No | Optional input port to enable dynamic switching between 36 × 36 normal mode and 18 ×18 complex mode. Values are 0 and 1. A value of 0 specifies a 36 × 36 normal mode, a value of 1 specifies a 18 ×18 complex mode. |

**Table 61. ALTMULT_COMPLEX Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result_imag | Yes | Imaginary output value of the multiplier. The size of the output port depends on the WIDTH_RESULT parameter value. |
| result_real | Yes | Real output value of the multiplier. The size of the output port depends on the WIDTH_RESULT parameter value. |

**Table 62. ALTMULT_COMPLEX Megafunction Parameters**

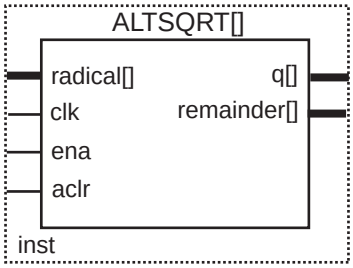
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| IMPLEMENTATION_STYLE | String | Yes | Specifies the representation algorithm and the number of bits per channel. Values are AUTO, CANONICAL, and CONVENTIONAL. If omitted, the default value is AUTO. When set to AUTO, the Quartus II software determines the best implementation based on the selected device family and input width. A value of CANONICAL is available for input widths that are less than 18 bits and for all supported devices, except for Stratix III devices. A value of CONVENTIONAL is available for all supported device families for all input ranges (1 to 256 bits). |
| PIPELINE | Integer | Yes | Specifies the amount of latency, in clock cycles, needed to produce the result. Values are [0..14]. If omitted, the default value is 4. If the value of IMPLEMENTATION_STYLE is CANONICAL, the maximum value of PIPELINE is 14, and if the value of the IMPLEMENTATION_STYLE parameter is CONVENTIONAL, the maximum value of PIPELINE is 11. |
| REPRESENTATION_A | String | Yes | Specifies the number representation of data A. Values are UNSIGNED and SIGNED. If omitted, the default value is UNSIGNED. The data A inputs are interpreted as unsigned numbers when the value is set to UNSIGNED, and as two's complement when the value is set to SIGNED. |
| REPRESENTATION_B | String | Yes | Specifies the number representation of data B. Values are UNSIGNED and SIGNED. If omitted, the default value is UNSIGNED. The data B inputs are interpreted as unsigned numbers when the value is set to UNSIGNED, and as two's complement when the value is set to SIGNED. |
| WIDTH_A | Integer | Yes | Specifies the width of the dataa_real[] and dataa_imag[] ports. Value must be 256 bits or less. If omitted, the default value is 18. |
| WIDTH_B | Integer | Yes | Specifies the width of the datab_real[] and datab_imag[] ports. Value must be 256 bits or less. If omitted, the default value is 18. |
| WIDTH_RESULT | Integer | Yes | Specifies the width of the result_real[] and result_imag[] ports. Value must be 256 bits or less. If omitted, the default value is 36. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTMULT_COMPLEX megafunction with the MegaWizard Plug-in Manager to calculate the value for this parameter. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |

# ALTSQRT (Integer Square Root)

The ALTSQRT megafunction implements a square root function that calculates the square root and remainder of an input.

Figure 36 shows the ports for the ALTSQRT megafunction.

**Figure 36. ALTSQRT Ports Shown in the MegaWizard Plug-In Manager**




## Features

The ALTSQRT megafunction offers the following features:

- Calculates the square root and the remainder of an input
- Supports data width of 1–256 bits
- Supports pipelining with configurable output latency
- Supports optional asynchronous clear and clock enable input ports

## Resource Utilization and Performance

Table 63 provides resource utilization and performance information for the ALTSQRT megafunction.

**Table 63. ALTSQRT Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 8 | 1 | 28 | 0 | 14 | 547 |
| | 20 | 5 | 131 | 0 | 90 | 321 |
| | 30 | 3 | 256 | 0 | 152 | 71 |
| Stratix IV | 8 | 1 | 28 | 0 | 14 | 350 |
| | 20 | 5 | 131 | 0 | 94 | 267 |
| | 30 | 3 | 256 | 0 | 154 | 66 |

**Notes to Table 63:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: 9-bit Square Root

This design example uses the ALTSQRT megafunction to generate a 9-bit square root. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **altsqrt_DesignExample.zip**:

- **altsqrt.qar** (archived Quartus II design files)
- **altsqrt_ex_msim** (ModelSim-Altera files)

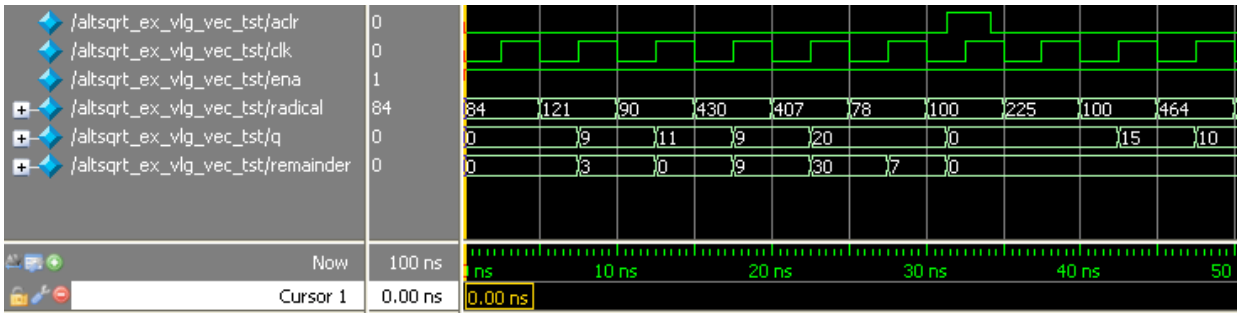
## Understanding the Simulation Results

The following settings are observed in this example:

- The width of the input port, `radical[]`, is set to 9 bits.
- The widths of the output ports, `q[]` and `remainder[]`, are set to 5 bits and 6 bits respectively.
- The asynchronous clear (`aclr`) and clock enable (`ena`) input ports are enabled.
- The output latency is set to two clock cycles. Hence, the result is seen on the `q[]` port two clock cycles after the input data is available.

Figure 37 shows the expected simulation results in the ModelSim-Altera software.

**Figure 37. ALTSQRT Simulation Results**



## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altsqrt
# (parameter lpm_hint = "UNUSED",
parameter lpm_type = "altsqrt",
parameter pipeline = 0,
parameter q_port_width = 1,
parameter r_port_width = 1,
parameter width = 1)
(input wire aclr,
input wire clk,
input wire ena,
output wire [q_port_width-1:0] q,
input wire [width-1:0] radical,
output wire [r_port_width-1:0] remainder);
endmodule
```

### 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 altsqrt
generic (
lpm_hint:string := "UNUSED";
lpm_type:string := "altsqrt";
pipeline:natural := 0;
q_port_width:natural := 1;
r_port_width:natural := 1;
width:natural);
port(
aclr:in std_logic := '0';
clk:in std_logic := '1';
ena:in std_logic := '1';
q:out std_logic_vector(Q_PORT_WIDTH-1 downto 0);
radical:in std_logic_vector(WIDTH-1 downto 0);
remainder:out std_logic_vector(R_PORT_WIDTH-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 64 lists the input ports, Table 65 lists the output ports, and Table 66 lists the ALTSQRT megafunction parameters.

**Table 64. ALTSQRT Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| radical[] | Yes | Data input port. The size of the input port depends on the WIDTH parameter value. |
| ena | No | Active high clock enable input port. |
| clk | No | Clock input port that provides pipelined operation for the ALTSQRT megafunction. For the values of PIPELINE parameter other than 0 (default value), the clock port must be connected. |
| aclr | No | Asynchronous clear input port. that can be used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. |

**Table 65. ALTSQRT Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| remainder[] | Yes | The square root of the radical. The size of the remainder[] port depends on the R_PORT_WIDTH parameter value. |
| q[] | Yes | Data output. The size of the q[] port depends on the Q_PORT_WIDTH parameter value. |

**Table 66. ALTSQRT Megafunction Parameters**

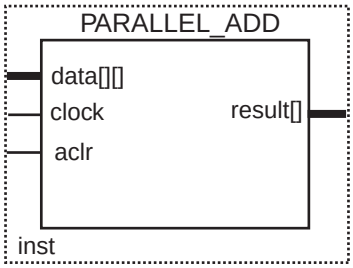
| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH | Integer | Yes | Specifies the widths of the radical[] input port. |
| Q_PORT_WIDTH | Integer | Yes | Specifies the width of the q[] output port. |
| R_PORT_WIDTH | Integer | Yes | Specifies the width of the remainder[] output port. |
| PIPELINE | Integer | No | Specifies the number of clock cycles of latency to add. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |

# PARALLEL_ADD (Parallel Adder)

The PARALLEL_ADD megafunction performs add or subtract operations on a selected number of inputs to produce a single sum result. You can add or subtract more than two operands and automatically shift the input operands upon entering the function. The method of shifting input operands is useful for serial FIR filter structures requiring a shift-and-accumulate of the partial products.

Figure 38 shows the ports for the PARALLEL_ADD megafunction.

**Figure 38. PARALLEL_ADD Ports Shown in the MegaWizard Plug-In Manager**




## Features

The PARALLEL_ADD megafunction offers the following features:

- Performs add or subtract operations on a number of inputs to produce a single sum result
- Supports data width of 8–128 bits
- Supports signed and unsigned data representation format
- Supports pipelining with configurable output latency
- Supports shifting data vectors
- Supports addition or subtraction of the most-significant input operands
- Supports optional asynchronous clear and clock enable ports

## Resource Utilization and Performance

Table 67 provides resource utilization and performance information for the PARALLEL_ADD megafunction.

**Table 67. PARALLEL_ADD Resource Utilization and Performance (1)**

| Device family | Input data width | Output latency | Logic Usage | | | $f_{MAX}$ (MHz) (2) |
|---|---|---|---|---|---|---|
| | | | Adaptive Look-Up Table (ALUT) | Dedicated Logic Register (DLR) | Adaptive Logic Module (ALM) | |
| Stratix III | 8 | 1 | 40 | 0 | 21 | 793 |
| | 32 | 5 | 142 | 0 | 102 | 378 |
| | 64 | 10 | 283 | 0 | 189 | 280 |

**Table 67. PARALLEL_ADD Resource Utilization and Performance** *(1)*

| | | | | | | |
|---|---|---|---|---|---|---|
| Stratix IV | 8 | 1 | 40 | 0 | 21 | 854 |
| | 32 | 5 | 142 | 0 | 103 | 472 |
| | 64 | 10 | 283 | 0 | 199 | 346 |

**Notes to Table 67:**

(1) The information in this table is valid and accurate in the Quartus II software version 9.1.

(2) The performance of the megafunction is dependant on the value of the maximum allowable ceiling $f_{MAX}$ that the selected device can achieve. Therefore, results may vary from the numbers stated in this column.

## Design Example: Shift Accumulator

This design example uses the LPM_MULT and PARALLEL_ADD megafunctions to generate a shift accumulator. This function implements the shift-and-accumulate operation after the multiplication process in a design block, such as a serial FIR filter. This example uses the MegaWizard Plug-In Manager in the Quartus II software.

### Design Files

The following design files can be found in **parallel_adder_DesignExample.zip**:

- **shift_accum.qar** (archived Quartus II design files)
- **parallel_adder_ex_msim** (ModelSim-Altera files)

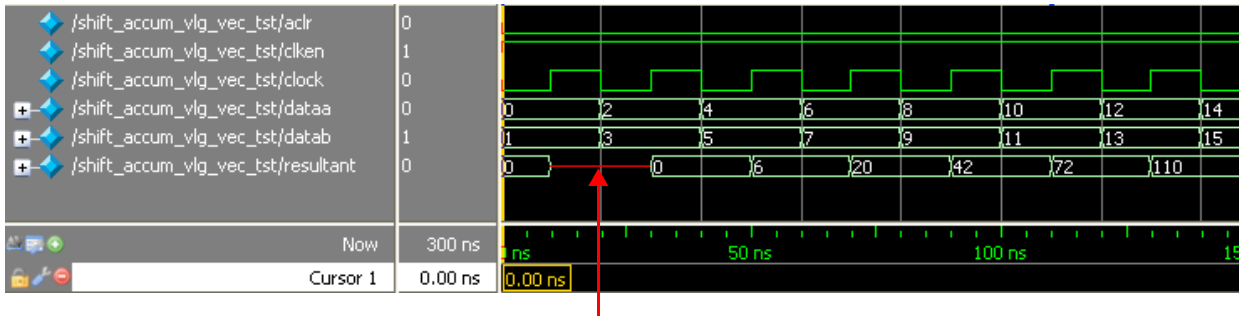
## Understanding the Simulation Results

The following settings are observed in this example:

- The widths of the input ports, `dataa[]` and `datab[]`, are set to 9 bits
- The width of the output port, `resultant[]`, is set to 10 bits
- The asynchronous clear (`aclr`) and clock enable (`clocken`) input ports are enabled
- The latency is set to one clock cycle for the multiplier and one clock cycle for the parallel adder, resulting in a total output latency of two clock cycles. Hence, the result is seen on the `resultant[]` port two clock cycles after the input data is available

Figure 39 shows the expected simulation results in the ModelSim-Altera software.

**Figure 39. PARALLEL_ADDER Simulation Results**




**Note to Figure 39:**

(1) At start up, an undefined value is seen on the `resultant[]` port, but this value is merely due to the behavior of the system during start-up and hence, can be ignored.

## Specifications

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

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module parallel_add (
    data,
    clock,
    aclr,
    clken,
    result);
    parameter width = 4;
    parameter size = 2;
    parameter widthr = 4;
    parameter shift = 0;
    parameter msw_subtract = "NO";  // or "YES"
    parameter representation = "UNSIGNED";
    parameter pipeline = 0;
    parameter result_alignment = "LSB"; // or "MSB"
    parameter lpm_type = "parallel_add";
    input [width*size-1:0] data;
    input clock;
    input aclr;
    input clken;
    output [widthr-1:0] result;
endmodule
```

## 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 parallel_add
   generic (
      width : natural := 4;
      size : natural := 2;
      widthr : natural := 4;
      shift : natural := 0;
      msw_subtract : string  := "NO";
      representation : string  := "UNSIGNED";
      pipeline : natural := 0;
      result_alignment : string  := "LSB";
      lpm_hint: string  := "UNUSED";
      lpm_type : string  := "parallel_add");
   port (
      data:in altera_mf_logic_2D(size - 1 downto 0,width- 1 downto 0);
      clock : in std_logic := '1';
      aclr : in std_logic := '0';
      clken : in std_logic := '1';
      result : out std_logic_vector(widthr - 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 68 lists the input ports, Table 69 lists the output ports, and Table 70 lists the PARALLEL_ADD megafunction parameters.

**Table 68. PARALLEL_ADD Megafunction Input Ports**

| Port Name | Required | Description |
|---|---|---|
| data[] | Yes | Data input to the parallel adder. Input port [SIZE - 1 DOWNTO 0, WIDTH-1 DOWNTO 0] wide. |
| clock | No | Clock input to the parallel adder. This port is required if the PIPELINE parameter has a value of greater than 0. |
| clken | No | Clock enable to the parallel adder. If omitted, the default value is 1. |
| aclr | No | Active high asynchronous clear input to the parallel adder. |

**Table 69. PARALLEL_ADD Megafunction Output Ports**

| Port Name | Required | Description |
|---|---|---|
| result[] | Yes | Adder output port. The size of the output port depends on the WIDTHR parameter value. |

**Table 70. PARALLEL_ADD Megafunction Parameters**

| Parameter Name | Type | Required | Description |
|---|---|---|---|
| WIDTH | Integer | Yes | Specifies the width of the data[] input port. |
| SIZE | Integer | Yes | Specifies the number of inputs to add. |
| WIDTHR | Integer | Yes | Specifies the width of the result[] output port. |
| SHIFT | Integer | Yes | Specifies the relative shift of the data vectors. |
| NEW_SUBTRACT | String | No | Specifies whether to add or subtract the most significant input word bit. Values are NO or YES. If omitted, the default value is NO. |
| REPRESENTATION | String | No | Specifies whether the input is signed or unsigned. Values are UNSIGNED or SIGNED. If omitted, the default value is UNSIGNED. |
| PIPELINE | Integer | No | Specifies the value, in clock cycles, of the output latency. |
| RESULT_ALIGNMENT | String | No | Specifies the alignment of the result port. Values are MSB or LSB. If omitted, the default value is LSB. |
| INTENDED_DEVICE_FAMILY | String | No | This parameter is used for modeling and behavioral simulation purposes. Create the ALTCDR_RX megafunction with the MegaWizard Plug-In Manager to calculate the value for this parameter. |
| LPM_HINT | String | No | Allows you to specify Altera-specific parameters in VHDL design files (**.vhd**). The default value is UNUSED. |
| LPM_TYPE | String | No | Identifies the library of parameterized modules (LPM) entity name in VHDL design files. |

# Document Revision History

Table 71 lists the revision history for this document.

**Table 71. Document Revision History**

| Date | Version | Changes |
|---|---|---|
| February 2012 | 3.0 | ■ Added Arria V and Cyclone V device support.<br>■ Updated the parameter description for the following section:<br>■ ALTMULT_ACCUM (Multiply-Accumulate)<br>■ ALTMULT_ADD (Multiply-Add)<br>■ Added the Double Accumulator section. |
| July 2010 | 2.0 | ■ Updated architecture information for the following sections:<br>■ ALTMULT_ACCUM (Multiply-Accumulate)<br>■ ALTMULT_ADD (Multiply-Add)<br>■ ALTMULT_COMPLEX (Complex Multiplier)<br>■ Added specification information for all megafunctions |
| November 2009 | 1.0 | Initial release. |