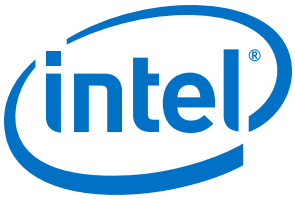


Intel FPGA Integer Arithmetic IP Cores User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**


Subscribe


Send Feedback

UG-01063 | 2020.04.26
Latest document on the web: **PDF** | **HTML**

Contents

1. Intel FPGA Integer Arithmetic IP Cores 5

2. LPM_COUNTER (Counter) IP Core 7
2.1. Features 7
2.2. Verilog HDL Prototype 8
2.3. VHDL Component Declaration 8
2.4. VHDL LIBRARY_USE Declaration 9
2.5. Ports 9
2.6. Parameters 10

3. LPM_DIVIDE (Divider) Intel FPGA IP Core 12
3.1. Features 12
3.2. Verilog HDL Prototype 12
3.3. VHDL Component Declaration 13
3.4. VHDL LIBRARY_USE Declaration 13
3.5. Ports 13
3.6. Parameters 14

4. LPM_MULT (Multiplier) IP Core 16
4.1. Features 16
4.2. Verilog HDL Prototype 17
4.3. VHDL Component Declaration 17
4.4. VHDL LIBRARY_USE Declaration 17
4.5. Signals 18
4.6. Parameters for Stratix V, Arria V, Cyclone V, and Intel Cyclone 10 LP Devices 18
4.6.1. General Tab 18
4.6.2. General 2 Tab 19
4.6.3. Pipelining Tab 19
4.7. Parameters for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices 20
4.7.1. General Tab 20
4.7.2. General 2 Tab 20
4.7.3. Pipelining 21

5. LPM_ADD_SUB (Adder/Subtractor) 22
5.1. Features 22
5.2. Verilog HDL Prototype 23
5.3. VHDL Component Declaration 23
5.4. VHDL LIBRARY_USE Declaration 23
5.5. Ports 23
5.6. Parameters 24

6. LPM_COMPARE (Comparator) 26
6.1. Features 26
6.2. Verilog HDL Prototype 27
6.3. VHDL Component Declaration 27
6.4. VHDL LIBRARY_USE Declaration 27
6.5. Ports 27
6.6. Parameters 28

7. ALTECC (Error Correction Code: Encoder/Decoder) IP Core..... 30
7.1. ALTECC Encoder Features..... 31
7.2. Verilog HDL Prototype (ALTECC_ENCODER)..... 32
7.3. Verilog HDL Prototype (ALTECC_DECODER)..... 32
7.4. VHDL Component Declaration (ALTECC_ENCODER)..... 33
7.5. VHDL Component Declaration (ALTECC_DECODER)..... 33
7.6. VHDL LIBRARY_USE Declaration..... 33
7.7. Encoder Ports..... 33
7.8. Decoder Ports..... 34
7.9. Encoder Parameters..... 34
7.10. Decoder Parameters..... 35

8. Intel FPGA Multiply Adder IP Core..... 36
8.1. Features..... 37
8.1.1. Pre-adder..... 38
8.1.2. Systolic Delay Register..... 40
8.1.3. Pre-load Constant..... 43
8.1.4. Double Accumulator..... 43
8.2. Verilog HDL Prototype..... 44
8.3. VHDL Component Declaration..... 44
8.4. VHDL LIBRARY_USE Declaration..... 44
8.5. Signals..... 44
8.6. Parameters..... 46
8.6.1. General Tab..... 46
8.6.2. Extra Modes Tab..... 46
8.6.3. Multipliers Tab..... 48
8.6.4. Preadder Tab..... 50
8.6.5. Accumulator Tab..... 52
8.6.6. Systolic/Chainout Tab..... 54
8.6.7. Pipelining Tab..... 55

9. ALTMEMMULT (Memory-based Constant Coefficient Multiplier) IP Core..... 57
9.1. Features..... 57
9.2. Verilog HDL Prototype..... 58
9.3. VHDL Component Declaration..... 58
9.4. Ports..... 59
9.5. Parameters..... 59

10. ALTMULT_ACCUM (Multiply-Accumulate) IP Core..... 61
10.1. Features..... 62
10.2. Verilog HDL Prototype..... 62
10.3. VHDL Component Declaration..... 63
10.4. VHDL LIBRARY_USE Declaration..... 63
10.5. Ports..... 63
10.6. Parameters..... 64

11. ALTMULT_ADD (Multiply-Adder) IP Core..... 69
11.1. Features..... 71
11.2. Verilog HDL Prototype..... 72
11.3. VHDL Component Declaration..... 72
11.4. VHDL LIBRARY_USE Declaration..... 72

11.5. Ports..... 72
11.6. Parameters..... 73

12. ALTMULT_COMPLEX (Complex Multiplier) IP Core..... 86
12.1. Complex Multiplication..... 86
12.2. Canonical Representation..... 87
12.3. Conventional Representation..... 87
12.4. Features..... 88
12.5. Verilog HDL Prototype..... 88
12.6. VHDL Component Declaration..... 89
12.7. VHDL LIBRARY_USE Declaration..... 89
12.8. Signals..... 89
12.9. Parameters..... 90

13. ALTSQRT (Integer Square Root) IP Core..... 92
13.1. Features..... 92
13.2. Verilog HDL Prototype..... 92
13.3. VHDL Component Declaration..... 93
13.4. VHDL LIBRARY_USE Declaration..... 93
13.5. Ports..... 93
13.6. Parameters..... 94

14. PARALLEL_ADD (Parallel Adder) IP Core..... 95
14.1. Feature..... 95
14.2. Verilog HDL Prototype..... 95
14.3. VHDL Component Declaration..... 96
14.4. VHDL LIBRARY_USE Declaration..... 96
14.5. Ports..... 96
14.6. Parameters..... 97

15. Integer Arithmetic IP Cores User Guide Document Archives..... 98

16. Document Revision History for Intel FPGA Integer Arithmetic IP Cores User Guide.... 99

1. Intel FPGA Integer Arithmetic IP Cores

You can use the Intel® FPGA integer IP cores to perform mathematical operations in your design.

These functions offer more efficient logic synthesis and device implementation than coding your own functions. You can customize the IP cores to accommodate your design requirements.

Intel integer arithmetic IP cores are divided into the following two categories:

- Library of parameterized modules (LPM) IP cores
- Intel-specific (ALT) IP cores

The following table lists the integer arithmetic IP cores.

Table 1. List of IP Cores

IP Cores	Function Overview	Supported Device
LPM IP cores		
LPM_COUNTER	Counter	Arria® II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone® IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, Intel Cyclone 10 GX, MAX® II, MAX V, MAX 10, Stratix® IV, Stratix V
LPM_DIVIDE	Divider	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, Intel Cyclone 10 GX, MAX II, MAX V, MAX 10, Stratix IV, Stratix V, Intel Stratix 10
LPM_MULT	Multiplier	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, Intel Cyclone 10 GX, MAX II, MAX V, MAX 10, Stratix IV, Stratix V, Intel Stratix 10
LPM_ADD_SUB	Adder or subtractor	Arria II GX, Arria II GZ, Arria V, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, MAX 10, MAX II, MAX V, Stratix IV, Stratix V
LPM_COMPARE	Comparator	Arria II GX, Arria II GZ, Arria V, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, MAX 10, MAX II, MAX V, Stratix IV, Stratix V
Intel-specific (ALT) IP cores		
ALTECC	ECC Encoder/Decoder	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V,Intel Cyclone 10 LP, Intel Cyclone 10 GX, MAX II, MAX V, MAX 10, Stratix IV, Stratix V

continued...

IP Cores	Function Overview	Supported Device
Intel FPGA Multiply Adder or ALTERA_MULT_ADD	Multiplier-Adder	Arria V, Stratix V, Cyclone V, Intel Stratix 10, Intel Arria 10, Intel Cyclone 10 GX
ALTMEMMULT	Memory-based Constant Coefficient Multiplier	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, MAX II, MAX V, MAX 10, Stratix IV, Stratix V
ALTMULT_ACCUM	Multiplier-Accumulator	Arria II GX, Arria II GZ, Cyclone IV E, Cyclone IV GX, Intel Cyclone 10 LP, MAX 10, MAX II, MAX V, Stratix IV
ALTMULT_ADD	Multiplier-Adder	Arria II GX, Arria II GZ, Cyclone IV E, Cyclone IV GX,Intel Cyclone 10 LP, MAX 10, MAX II, MAX V, Stratix IV
ALTMULT_COMPLEX	Complex Multiplier	Arria II GX, Arria II GZ, Intel Arria 10, Arria V, Arria V GZ, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 GX, Intel Cyclone 10 LP, MAX 10, Stratix V, Intel Stratix 10
ALTSQRT	Integer Square-Root	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP, Intel Cyclone 10 GX, MAX II, MAX V, MAX 10, Stratix IV, Stratix V
PARALLEL_ADD	Parallel Adder	Arria II GX, Arria II GZ, Arria V, Intel Arria 10, Cyclone IV E, Cyclone IV GX, Cyclone V, Intel Cyclone 10 LP,Intel Cyclone 10 GX, MAX II, MAX V, MAX 10, Stratix IV, Stratix V

Related Information

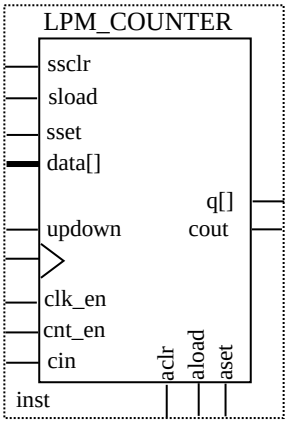
- Intel FPGA IP Release Notes
- Introduction to Intel FPGA IP Cores
  Provides more information about Intel FPGA IP Cores.
- Floating Point IP Cores User Guide
  Provides more information about Intel FPGA Floating-Point IP cores.
- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Qsys Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.
- Integer Arithmetic IP Cores User Guide Document Archives on page 98
  Provides a list of user guides for previous versions of the Integer Arithmetic IP cores.

2. LPM_COUNTER (Counter) IP Core

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

The following figure shows the ports for the LPM_COUNTER IP core.

Figure 1. LPM_COUNTER Ports



2.1. Features

The LPM_COUNTER IP core 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

2.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the <Intel Quartus® Prime 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
```

2.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the <Intel Quartus Prime 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;
```

2.4. 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;
```

2.5. Ports

The following tables list the input and output ports for the LPM_COUNTER IP core.

Table 2. LPM_COUNTER 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.
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 3. LPM_COUNTER 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 in the parameter editor because the parameter only supports AHDL. 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.

2.6. Parameters

The following table lists the parameters for the LPM_COUNTER IP core.

Table 4. LPM_COUNTER Parameters

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. Intel 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 the sset port is asserted high. Intel recommends that you specify this value as a decimal number for AHDL designs.
LPM_HINT	String	No	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (.vhd), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES" The default value is UNUSED.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL design files.

continued...

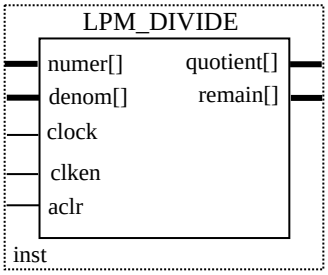
Parameter Name	Type	Required	Description
INTENDED_DEVICE_FAMILY	String	No	This parameter is used for modeling and behavioral simulation purposes. This parameter is used for modeling and behavioral simulation purposes. The parameter editor calculates the value for this parameter.
CARRY_CNT_EN	String	No	Intel-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.
LABWIDE_SCLR	String	No	Intel-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 Intel 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.

3. LPM_DIVIDE (Divider) Intel FPGA IP Core

The LPM_DIVIDE Intel FPGA IP core implements a divider to divide a numerator input value by a denominator input value to produce a quotient and a remainder.

The following figure shows the ports for the LPM_DIVIDE IP core.

Figure 2. LPM_DIVIDE Ports



3.1. Features

The LPM_DIVIDE IP core 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.

3.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the `<Intel Quartus Prime 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
```

3.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the <Intel Quartus Prime 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;
```

3.4. 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;
```

3.5. Ports

The following tables list the input and output ports for the LPM_DIVIDE IP core.

Table 5. LPM_DIVIDE 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.

continued...

Port Name	Required	Description
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 6. LPM_DIVIDE 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.

3.6. Parameters

The following table lists the parameters for the LPM_DIVIDE Intel FPGA IP core.

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	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (**.vhd**), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES" The default value is UNUSED.
LPM_REMAINDERPOSITIVE	String	No	Intel-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

continued...

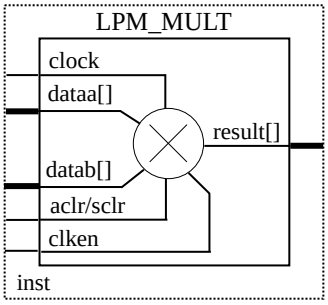
Parameter Name	Type	Required	Description
			than or equal to zero. If this parameter is set to TRUE, then the value of the `remain[]` port is either zero, or the value is the same sign, either positive or negative, as the value of the `numer` port. In order to reduce area and improve speed, Intel recommends setting this parameter to TRUE in operations where the remainder must be positive or where the remainder is unimportant.
`MAXIMIZE_SPEED`	Integer	No	Intel-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 Intel Quartus Prime 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` IP core 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. The parameter editor calculates 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 IP core. Specify the number of leading GND on the quotient output to this parameter.

4. LPM_MULT (Multiplier) IP Core

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

The following figure shows the ports for the LPM_MULT IP core.

Figure 3. LPM_Mult Ports



Related Information

Features on page 71

4.1. Features

The LPM_MULT IP core 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)

  Note: When building multipliers larger than the natively supported size there may/ will be a performance impact resulting from the cascading of the DSP blocks.
- Supports optional asynchronous clear and clock enable input ports
- Supports optional synchronous clear for Intel Stratix 10, Intel Arria 10 and Intel Cyclone 10 GX devices

4.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the `<Intel Quartus Prime 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
```

4.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the `<Intel Quartus Prime 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;
```

4.4. 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;
```

4.5. Signals

Table 7. LPM_MULT Input Signals

Signal Name	Required	Description
dataa[]	Yes	Data input. For Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices, the size of the input signal depends on the **Dataa width** parameter value. For older and Intel Cyclone 10 LP devices, the size of the input signal depends on the **LPM_WIDTHA** parameter value.
datab[]	Yes	Data input. For Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices, the size of the input signal depends on the **Datab width** parameter value. For older and Intel Cyclone 10 LP devices, the size of the input signal depends on the LPM_WIDTHB parameter value.
clock	No	Clock input for pipelined usage. For older and Intel Cyclone 10 LP devices, the clock signal must be enabled for LPM_PIPELINE values other than 0 (default). For Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices, the clock signal must be enabled if **Latency** value is other than **1** (default).
clken	No	Clock enable for pipelined usage. When the clken signal 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 signal used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value.
sclr	No	Synchronous clear signal used at any time to reset the pipeline to all 0s, synchronously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value.

Table 8. LPM_MULT Output signals

signal Name	Required	Description
result[]	Yes	Data output. For older and Intel Cyclone 10 LP devices, the size of the output signal 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. For Intel Stratix 10, Intel Arria 10 and Intel Cyclone 10 GX, the size of the output signals depends on the **Result width** parameter.

4.6. Parameters for Stratix V, Arria V, Cyclone V, and Intel Cyclone 10 LP Devices

4.6.1. General Tab

Table 9. General Tab

Parameter	Value	Default Value	Description
Multiplier Configuration	**Multiply 'dataa' input by 'datab' input**	**Multiply 'dataa' input by 'datab' input**	Select the desired configuration for the multiplier.

continued...

Parameter	Value	Default Value	Description
	Multiply 'dataa' input by itself (squaring operation)		
How wide should the 'dataa' input be?	1 - 256 bits	8 bits	Specify the width of the `dataa[]` port.
How wide should the 'datab' input be?	1 - 256 bits	8 bits	Specify the width of the `datab[]` port.
How should the width of the 'result' output be determined?	**Automatically calculate the width** **Restrict the width**	**Automatically calculate the width**	Select the desired method to determine the width of the `result[]` port.
Restrict the width	1 - 512 bits	16 bits	Specify the width of the `result[]` port. This value will only be effective if you select **Restrict the width** in the **Type** parameter.

4.6.2. General 2 Tab

Table 10. General 2 Tab

Parameter	Value	Default Value	Description
Datab Input			
Does the 'datab' input bus have a constant value?	**No** **Yes**	**No**	Select **Yes** to specify the constant value of the 'datab' input bus, if any.
Multiplication Type			
Which type of multiplication do you want?	**Unsigned** **Signed**	**Unsigned**	Specify the representation format for both `dataa[]` and `datab[]` inputs.
Implementation			
Which multiplier implementation should be used?	**Use the default implementation** **Use the dedicated multiplier circuitry (Not available for all families)** **Use logic elements**	**Use the default implementation**	Select the desired method to determine the width of the `result[]` port.

4.6.3. Pipelining Tab

Table 11. Pipelining Tab

Parameter	Value	Default Value	Description
Do you want to pipeline the function?	**No** **Yes**	**No**	Select **Yes** to enable pipeline register to the multiplier's output and specify the desired output latency in clock cycle. Enabling the pipeline register adds extra latency to the output.
Create an 'aclr' asynchronous clear port	—	Unchecked	Select this option to enable `aclr` port to use asynchronous clear for the pipeline register.

continued...

Parameter	Value	Default Value	Description
Create a 'clken' clock enable clock	—	Unchecked	Specifies active high clock enable for the clock port of the pipeline register
Optimization			
What type of optimization do you want?	**Default** **Speed** **Area**	**Default**	Specify the desired optimization for the IP core. Select **Default** to let Intel Quartus Prime software to determine the best optimization for the IP core.

4.7. Parameters for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices

4.7.1. General Tab

Table 12. General Tab

Parameter	Value	Default Value	Description
Multiplier Configuration			
Type	**Multiply 'dataa' input by 'datab' input** **Multiply 'dataa' input by itself (squaring operation)**	**Multiply 'dataa' input by 'datab' input**	Select the desired configuration for the multiplier.
Data Port Widths			
Dataa width	1 - 256 bits	8 bits	Specify the width of the `dataa[]` port.
Datab width	1 - 256 bits	8 bits	Specify the width of the `datab[]` port.
How should the width of the 'result' output be determined?			
Type	**Automatically calculate the width** **Restrict the width**	**Automatically calculate the width**	Select the desired method to determine the width of the `result[]` port.
Value	1 - 512 bits	16 bits	Specify the width of the `result[]` port. This value will only be effective if you select **Restrict the width** in the **Type** parameter.
Result width	1 - 512 bits	—	Displays the effective width of the `result[]` port.

4.7.2. General 2 Tab

Table 13. General 2 Tab

Parameter	Value	Default Value	Description
Datab Input			
Does the 'datab' input bus have a constant value?	**No** **Yes**	**No**	Select **Yes** to specify the constant value of the 'datab' input bus, if any.

continued...

Parameter	Value	Default Value	Description
Value	Any value greater than 0	0	Specify the constant value of `datab[]` port.
Multiplication Type			
Which type of multiplication do you want?	**Unsigned** **Signed**	**Unsigned**	Specify the representation format for both `dataa[]` and `datab[]` inputs.
Implementation Style			
Which multiplier implementation should be used?	**Use the default implementation** **Use the dedicated multiplier circuitry** **Use logic elements**	**Use the default implementation**	Select the desired method to determine the width of the `result[]` port.

4.7.3. Pipelining

Table 14. Pipelining Tab

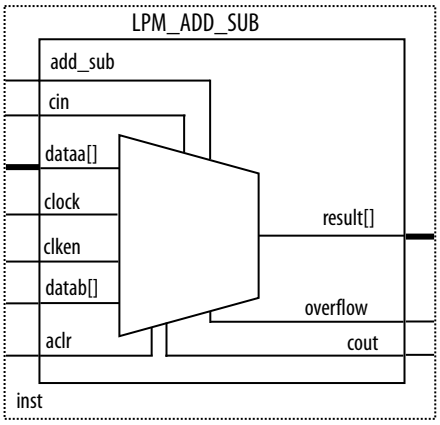
Parameter	Value	Default Value	Description
Do you want to pipeline the function?			
Pipeline	**No** **Yes**	**No**	Select **Yes** to enable pipeline register to the multiplier's output. Enabling the pipeline register adds extra latency to the output.
Latency	Any value greater than 0.	1	Specify the desired output latency in clock cycle.
Clear Signal Type	**NONE** **ACLR** **SCLR**	**NONE**	Specify the type of reset for the pipeline register. Select **NONE** if you do not use any pipeline register. Select **ACLR** to use asynchronous clear for the pipeline register. This will generate ACLR port. Select **SCLR** to use synchronous clear for the pipeline register. This will generate SCLR port.
Create a 'clken' clock enable clock	—	—	Specifies active high clock enable for the clock port of the pipeline register
What type of optimization do you want?			
Type	**Default** **Speed** **Area**	**Default**	Specify the desired optimization for the IP core. Select **Default** to let Intel Quartus Prime software to determine the best optiomization for the IP core.

5. LPM_ADD_SUB (Adder/Subtractor)

The LPM_ADD_SUB IP core 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.

The following figure shows the ports for the LPM_ADD_SUB IP core.

Figure 4. LPM_ADD_SUB Ports



5.1. Features

The LPM_ADD_SUB IP core 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.

5.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) lpm.v in the `<Intel Quartus Prime 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 = "SIGNED";
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
```

5.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the `<Intel Quartus Prime 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;
```

5.4. 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;
```

5.5. Ports

The following tables list the input and output ports for the LPM_ADD_SUB IP core.

Table 15. LPM_ADD_SUB IP Core 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. Intel 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 16. LPM_ADD_SUB IP Core 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.

5.6. Parameters

The following table lists the LPM_ADD_SUB IP core parameters.

Table 17. LPM_ADD_SUB IP Core 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. Intel 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.

continued...

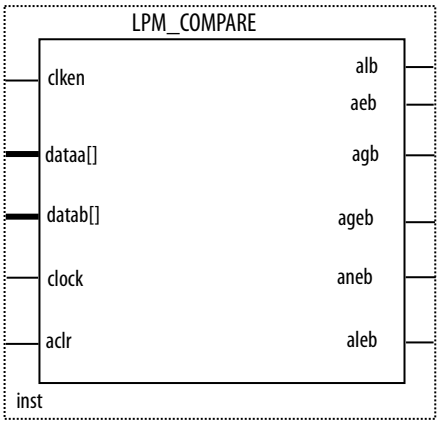
Parameter Name	Type	Required	Description
LPM_REPRESENTATION	String	No	Specifies the type of addition performed. Values are SIGNED and UNSIGNED. If omitted, the default value is SIGNED. When this parameter is set to SIGNED, the adder/subtractor interprets the data input as signed two's complement.
LPM_PIPELINE	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 Intel-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	Intel-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.
MAXIMIZE_SPEED	Integer	No	Intel-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 Intel Quartus Prime 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 IP core 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. The parameter editor calculates the value for this parameter.

6. LPM_COMPARE (Comparator)

The LPM_COMPARE IP core 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.

The following figure shows the ports for the LPM_COMPARE IP core.

Figure 5. LPM_COMPARE Ports



6.1. Features

The LPM_COMPARE IP core 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

6.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the `<Intel Quartus Prime 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
```

6.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **LPM_PACK.vhd** in the `<Intel Quartus Prime 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;
```

6.4. 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;
```

6.5. Ports

The following tables list the input and output ports for the LMP_COMPARE IP core.

Table 18. LPM_COMPARE IP core 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 19. LPM_COMPARE IP core Output Ports

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.

6.6. Parameters

The following table lists the parameters for the LPM_COMPARE IP core.

Table 20. LPM_COMPARE IP core 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 Intel-specific parameters in VHDL design files (.vhd). The default value is UNUSED.

continued...

Parameter Name	Type	Required	Description
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. The parameter editor calculates the value for this parameter.
ONE_INPUT_IS_CONSTANT	String	No	Intel-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.

7. ALTECC (Error Correction Code: Encoder/Decoder) IP Core

Intel provides the ALTECC IP core to implement the ECC functionality. ECC detects 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 single error correction double error detection (SECDED).

You can configure encoder and decoder functions of the ALTECC IP core. The data input to the encoder 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 decoder module for decoding just before reaching its destination block. The decoder generates a syndrome vector to determine if there is any error in the received code word. The decoder corrects 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 decoder also has flag signals to show the status of the data received and the action taken by the decoder, if any.

The following figures show the ports for the ALTECC IP core.

Figure 6. ALTECC Encoder Ports

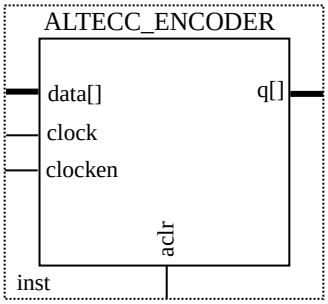
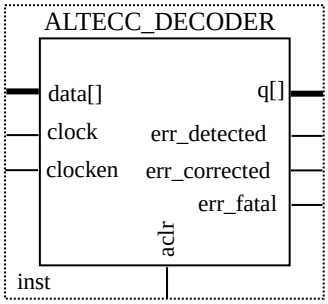


Figure 7. ALTECC Decoder Ports



7.1. ALTECC Encoder Features

The ALTECC encoder IP core 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 IP core 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.

The following table 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.

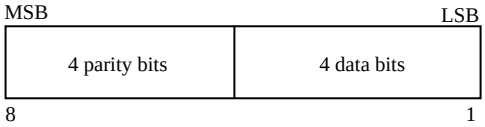
Table 21. Number of Parity Bits and Code Word According to Data Width

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 the table as +1) is appended to the parity bits as the 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.

The following figure shows the generated code word and the arrangement of the parity bits and data bits in an 8-bit data input.

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



The ALTECC encoder IP core 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 Intel devices, generate outputs of 18 bits, 36 bits, and 72 bits respectively. You can control the bit-selection limitation in the parameter editor.

7.2. Verilog HDL Prototype (ALTECC_ENCODER)

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the `<Intel Quartus Prime 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
```

7.3. Verilog HDL Prototype (ALTECC_DECODER)

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **lpm.v** in the `<Intel Quartus Prime 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
```

7.4. VHDL Component Declaration (ALTECC_ENCODER)

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the `<Intel Quartus Prime 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;
```

7.5. VHDL Component Declaration (ALTECC_DECODER)

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the `<Intel Quartus Prime 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);
err_corrected    :    out std_logic;
err_detected    :    out std_logic;
q:out std_logic_vector(width_dataword-1 downto 0);
syn_e    :    out std_logic);
end component;
```

7.6. 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;
```

7.7. Encoder Ports

The following tables list the input and output ports for the ALTECC encoder IP core.

Table 22. ALTECC Encoder 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 23. ALTECC Encoder 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.

7.8. Decoder Ports

The following tables list the input and output ports for the ALTECC decoder IP core.

Table 24. ALTECC Decoder 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 25. ALTECC Decoder 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.
syn_e	No	An output signal which will go high whenever a single-bit error is detected on the parity bits.

7.9. Encoder Parameters

The following table lists the parameters for the ALTECC encoder IP core.

Table 26. ALTECC Encoder 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.

7.10. Decoder Parameters

The following table lists the ALTECC decoder IP core parameters.

Table 27. ALTECC Decoder Parameters

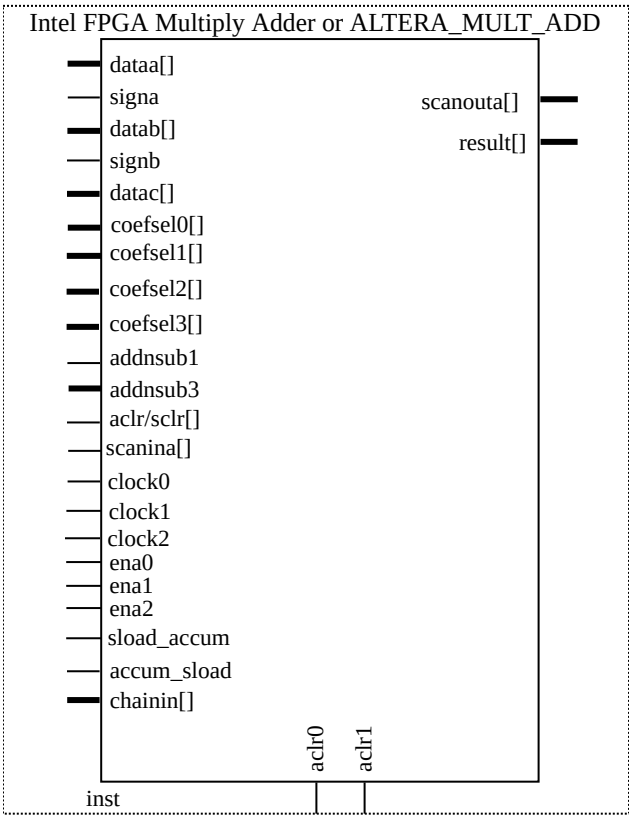
Parameter Name	Type	Required	Description
WIDTH_DATAWORD	Integer	Yes	Specifies the width of the raw data. Values are 2 to 64. The default value is 8.
WIDTH_CODEWORD	Integer	Yes	Specifies the width of the corresponding code word. Values are 6 to 72, excluding 9, 17, 33, and 65. If omitted, the default 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.
Create a 'syn_e' port	Integer	No	Turn on this parameter to create a syn_e port.

8. Intel FPGA Multiply Adder IP Core

The Intel FPGA Multiply Adder (Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices) or ALTERA_MULT_ADD (Arria V, Stratix V, and Cyclone V devices) IP core allows you to implement a multiplier-adder.

The following figure shows the ports for the Intel FPGA Multiply Adder or ALTERA_MULT_ADD IP core.

Figure 9. Intel FPGA Multiply Adder or ALTERA_MULT_ADD Ports



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

If all of the input data widths are 9-bits wide or smaller, the function uses the 9 x 9 bit input multiplier configuration in the DSP block for devices which support 9 x 9 configuration. If not, the DSP block uses 18 × 18-bit input multipliers to process data with widths between 10 bits and 18 bits. If multiple Intel FPGA Multiply Adder or ALTERA_MULT_ADD IP cores 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 V, and Arria V device series, refer to the DSP Blocks chapter of the respective handbooks on the Literature and Technical Documentation page.

Related Information

AN306: Implementing Multipliers in FPGA Devices
Provides more information about implementing multipliers using DSP and memory blocks in Intel FPGA devices.

8.1. Features

The Intel FPGA Multiply Adder or ALTERA_MULT_ADD IP core offers the following features:

- Generates a multiplier to perform multiplication operations of two complex numbers

  Note: When building multipliers larger than the natively supported size there may/ will be a performance impact resulting from the cascading of the DSP blocks.
- Supports data widths of 1– 256 bits
- Supports signed and unsigned data representation format
- Supports pipelining with configurable input latency
- Provides an option to dynamically switch between signed and unsigned data support
- Provides an option to dynamically switch between add and subtract operation
- Supports optional asynchronous and synchronous clear and clock enable input ports
- Supports systolic delay register mode
- Supports pre-adder with 8 pre-load coefficients per multiplier
- Supports pre-load constant to complement accumulator feedback

8.1.1. Pre-adder

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

There are five pre-adder modes:

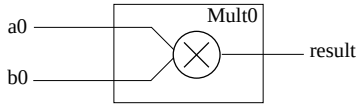
- Simple mode
- Coefficient mode
- Input mode
- Square mode
- Constant mode

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

8.1.1.1. 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 10. Pre-adder Simple Mode



8.1.1.2. 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]`.

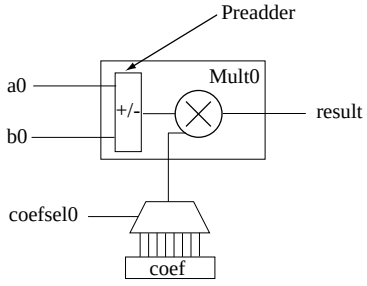
This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} (a_n + b_n) \times coef_n$$

where $k =$ number of multipliers

The following shows the pre-adder coefficient mode of a multiplier.

Figure 11. Pre-adder Coefficient Mode



8.1.1.3. Pre-adder Input Mode

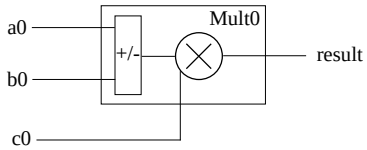
In this mode, one multiplier operand derives from the pre-adder, and the other operand derives from the `datac[]` input port.

This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} (a_n + b_n) \times c_n$$
$$\text{where } k = \text{ number of multipliers}$$

The following shows the pre-adder input mode of a multiplier.

Figure 12. Pre-adder Input Mode



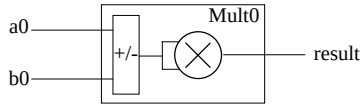
8.1.1.4. Pre-adder Square Mode

This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} (a_n + b_n)^2$$
$$\text{where } k = \text{ number of multipliers}$$

The following shows the pre-adder square mode of two multipliers.

Figure 13. Pre-adder Square Mode



8.1.1.5. Pre-adder Constant Mode

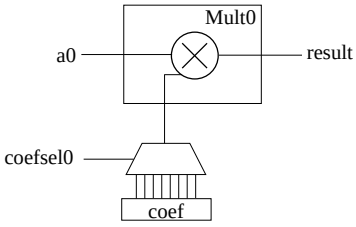
In this mode, one multiplier operand derives from the input port, and the other operand derives from the internal coefficient storage. The coefficient storage allows up to 8 preset constants. The coefficient selection signals are `coefsel[0..3]`.

This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} a_0 \times coef$$
$$\text{where } k = \text{ number of multipliers}$$

The following figure shows the pre-adder constant mode of a multiplier.

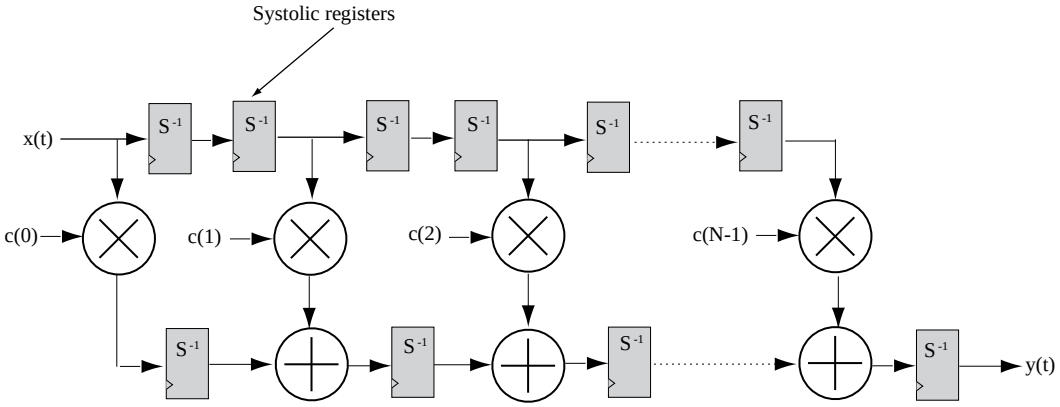
Figure 14. Pre-adder Constant Mode



8.1.2. 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 15. 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*-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 the following equation.

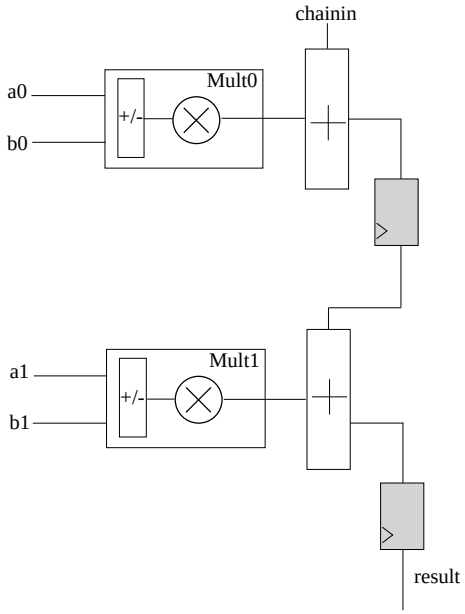
$$y(t) = \sum_{i=0}^{N-1} B(i)A(t-i)$$

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*.

The systolic register architecture is available only for sum-of-2 and sum-of-4 modes. For both systolic register architecture modes, the first `chainin` signal needs to be tied to 0.

The following figure shows the systolic delay register implementation of 2 multipliers.

Figure 16. Systolic Delay Register Implementation of 2 Multipliers

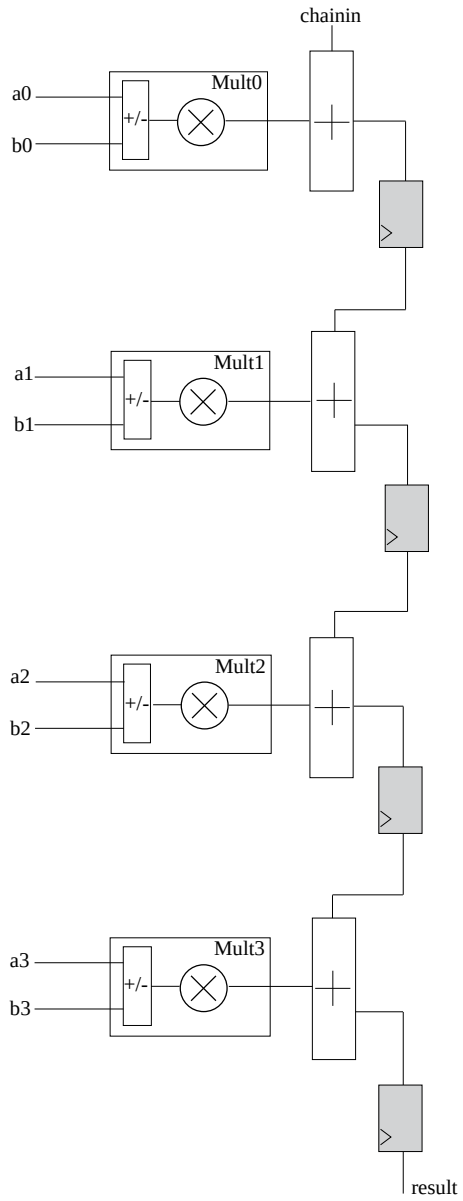


The sum of two multipliers is expressed in the following equation.

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

The following figure shows the systolic delay register implementation of 4 multipliers.

Figure 17. Systolic Delay Register Implementation of 4 Multipliers



The sum of four multipliers is expressed in the following equation.

Figure 18. Sum of 4 Multipliers

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

The following lists the advantages of systolic register implementation:

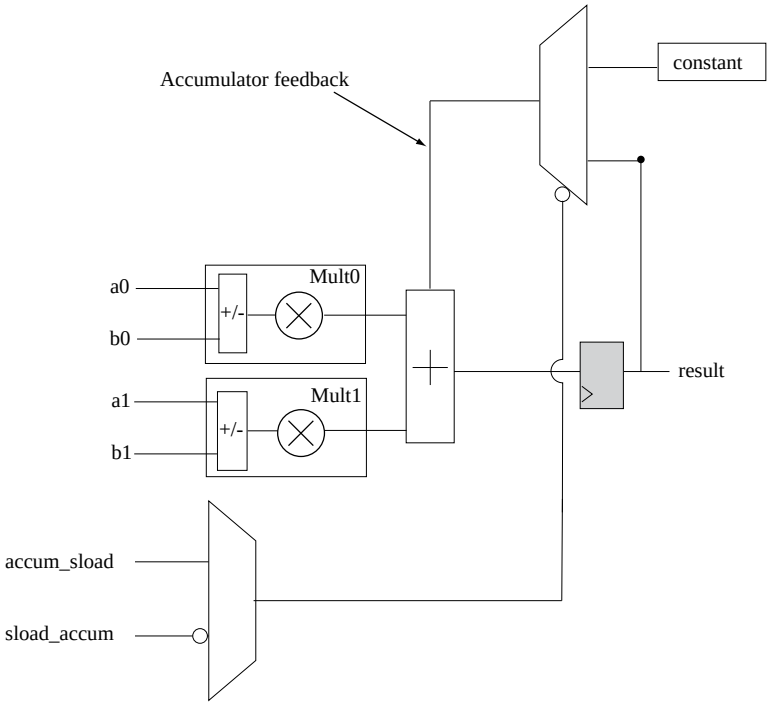
- Reduces DSP resource usage
- Enables efficient mapping in the DSP block using the chain adder structure

8.1.3. 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.

The following figure shows the pre-load constant implementation.

Figure 19. Pre-load Constant



Refer to the following IP cores for other multiplier implementations:

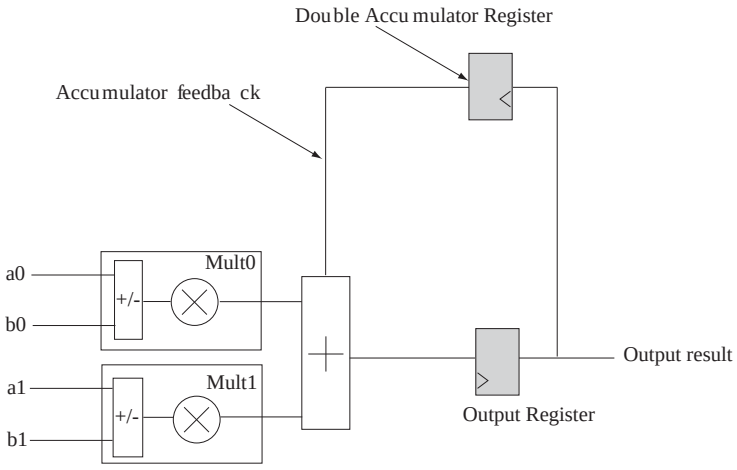
- ALTMULT_ACCUM
- ALTMEMMULT
- LPM_MULT

8.1.4. 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.

The following figure shows the double accumulator implementation.

Figure 20. Double Accumulator



8.2. Verilog HDL Prototype

You can find the Intel FPGA Multiply Adder or ALTERA_MULT_ADD Verilog HDL prototype file (altera_mult_add_rtl.v) in the `<Intel Quartus Prime installation directory>\libraries\megafunctions` directory.

8.3. VHDL Component Declaration

The VHDL component declaration is located in the altera_lnsim_components.vhd in the `<Intel Quartus Prime installation directory>\libraries\vhdl\altera_lnsim` directory.

8.4. 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;
```

8.5. Signals

The following tables list the input and output signals of the Multiply Adder Intel FPGA IPor ALTERA_MULT_ADD IP core.

Table 28. Multiply Adder Intel FPGA IPor ALTERA_MULT_ADD Input Signals

Signal	Required	Description
`dataa_0[]/dataa_1[]/ dataa_2[]/dataa_3[]`	Yes	Data input to the multiplier. Input port `[NUMBER_OF_MULTIPLIERS * WIDTH_A - 1 … 0]` wide
`datab_0[]/datab_1[]/ datab_2[]/datab_3[]`	Yes	Data input to the multiplier. Input signal `[NUMBER_OF_MULTIPLIERS * WIDTH_B - 1 … 0]` wide

continued...

Signal	Required	Description
datac_0[] /datac_1[]/ datac_2[]/datac_3[]	No	Data input to the multiplier. Input signal [NUMBER_OF_MULTIPLIERS * WIDTH_C - 1, … 0] wide Select **INPUT** for **Select preadder mode** parameter to enable these signals.
clock[1:0]	No	Clock input port to the corresponding register. This signal can be used by any register in the IP core.
aclr[1:0]	No	Asynchronous clear input to the corresponding register.
sclr[1:0]	No	Synchronous clear input to the corresponding register.
ena[1:0]	No	Enable signal input to the corresponding register.
signa	No	Specifies the numerical representation of the multiplier input A. If the signa signal is high, the multiplier treats the multiplier input A signal as a signed number. If the signa signal is low, the multiplier treats the multiplier input A signal as an unsigned number. Select **VARIABLE** for **What is the representation format for Multipliers A inputs** parameter to enable this signal.
signb	No	Specifies the numerical representation of the multiplier input B signal. If the signb signal is high, the multiplier treats the multiplier input B signal as a signed two's complement number. If the signb signal is low, the multiplier treats the multiplier input B signal as an unsigned number.
scanina[]	No	Input for scan chain A. Input signal [WIDTH_A - 1, ... 0] wide. When the INPUT_SOURCE_A parameter has a value of SCANA, the scanina[] signal is required.
accum_sload	No	Dynamically specifies whether the accumulator value is constant. If the accum_sload signal is low, then the multiplier output is loaded into the accumulator. Do not use accum_sload and sload_accum simultaneously.
sload_accum	No	Dynamically specifies whether the accumulator value is constant. If the sload_accum signal is high, then the multiplier output is loaded into the accumulator. Do not use accum_sload and sload_accum simultaneously.
chainin[]	No	Adder result input bus from the preceding stage. Input signal [WIDTH_CHAININ - 1, … 0] wide.
addnsub1	No	Perform addition or subtraction to the outputs from the first pair of multipliers. Input 1 to addnsub1 signal to add the outputs from the first pair of multipliers. Input 0 to addnsub1 signal to subtract the outputs from the first pair of multipliers.
addnsub3	No	Perform addition or subtraction to the outputs from the first pair of multipliers. Input 1 to addnsub3 signal to add the outputs from the second pair of multipliers. Input 0 to addnsub3 signal to subtract the outputs from the first pair of multipliers.
coefsel0[]	No	Coefficient input signal[0:3] to the first multiplier.
coefsel1[]	No	Coefficient input signal[0:3]to the second multiplier.
coefsel2[]	No	Coefficient input signal[0:3]to the third multiplier.
coefsel3[]	No	Coefficient input signal [0:3] to the fourth multiplier.

Table 29. Multiply Adder Intel FPGA IP Output Signals

Signal	Required	Description
`result []`	Yes	Multiplier output signal. Output signal `[WIDTH_RESULT - 1 … 0]` wide
`scanouta []`	No	Output of scan chain A. Output signal `[WIDTH_A - 1..0]` wide. Select more than 2 for numbers of multipliers and choose **Scan chain input** for **What is the input A of the multiplier connected to** parameter to enable this signal.

8.6. Parameters

8.6.1. General Tab

Table 30. General Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
What is the number of multipliers?	number_of_multipliers	1 - 4	1	Number of multipliers to be added together. Values are 1 up to 4.
How wide should the A input buses be?	width_a	1 - 256	16	Specify the width of the `dataa[]` port.
How wide should the B input buses be?	width_b	1 - 256	16	Specify the width of the `datab[]` port.
How wide should the 'result' output bus be?	width_result	1 - 256	32	Specify the width of the `result[]` port.
Create an associated clock enable for each clock	gui_associated_clock_enable	On Off	Off	Select this option to create clock enable for each clock.

8.6.2. Extra Modes Tab

Table 31. Extra Modes Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Outputs Configuration				
Register output of the adder unit	gui_output_register	On Off	Off	Select this option to enable output register of the adder module.
What is the source for clock input?	gui_output_register_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the clock source for output registers. You must select **Register output of the adder unit** to enable this parameter.
What is the source for asynchronous clear input?	gui_output_register_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the adder output register. You must select **Register output of the adder unit** to enable this parameter.
What is the source for synchronous clear input?	gui_output_register_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the adder output register. You must select **Register output of the adder unit** to enable this parameter.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
Adder Operation				
What operation should be performed on outputs of the first pair of multipliers?	gui_multiplier1_direction	**ADD**, **SUB**, **VARIABLE**	**ADD**	Select addition or subtraction operation to perform for the outputs between the first and second multipliers. • Select **ADD** to perform addition operation. • Select **SUB** to perform subtraction operation. • Select **VARIABLE** to use addnsub1 port for dynamic addition/subtraction control. When **VARIABLE** value is selected: • Drive addnsub1 signal to high for addition operation. • Drive addnsub1 signal to low for subtraction operation. You must select more than two multipliers to enable this parameter.
Register 'addnsub1' input	gui_addnsub_multiplier_register1	On Off	Off	Select this option to enable input register for addnsub1 port. You must select **VARIABLE** for **What operation should be performed on outputs of the first pair of multipliers** to enable this parameter.
What is the source for clock input?	gui_addnsub_multiplier_register1_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to specify the input clock signal for addnsub1 register. You must select **Register 'addnsub1' input** to enable this parameter.
What is the source for asynchronous clear input?	gui_addnsub_multiplier_aclr1	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the addnsub1 register. You must select **Register 'addnsub1' input** to enable this parameter.
What is the source for synchronous clear input?	gui_addnsub_multiplier_sclr1	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the addnsub1 register. You must select **Register 'addnsub1' input** to enable this parameter.
What operation should be performed on outputs of the second pair of multipliers?	gui_multiplier3_direction	**ADD**, **SUB**, **VARIABLE**	**ADD**	Select addition or subtraction operation to perform for the outputs between the third and fourth multipliers. • Select **ADD** to perform addition operation. • Select **SUB** to perform subtraction operation. • Select **VARIABLE** to use addnsub1 port for dynamic addition/subtraction control. When **VARIABLE** value is selected: • Drive addnsub1 signal to high for addition operation. • Drive addnsub1 signal to low for subtraction operation. You must select the value **4** for **What is the number of multipliers?** to enable this parameter.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
Register 'addnsub3' input	gui_addnsub_multiplier_register3	On Off	Off	Select this option to enable input register for addnsub3 signal. You must select **VARIABLE** for **What operation should be performed on outputs of the second pair of multipliers** to enable this parameter.
What is the source for clock input?	gui_addnsub_multiplier_register3_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to specify the input clock signal for addnsub3 register. You must select **Register 'addnsub3' input** to enable this parameter.
What is the source for asynchronous clear input?	gui_addnsub_multiplier_aclr3	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the addnsub3 register. You must select **Register 'addnsub3' input** to enable this parameter.
What is the source for synchronous clear input?	gui_addnsub_multiplier_sclr3	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the addnsub3 register. You must select **Register 'addnsub3' input** to enable this parameter.
Polarity				
Enable 'use_subadd'	gui_use_subnadd	On Off	Off	Select this option to reverse the function of addnsub input port. Drive addnsub to high for subtraction operation. Drive addnsub to low for addition operation.

8.6.3. Multipliers Tab

Table 32. Multipliers Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
What is the representation format for Multipliers A inputs?	gui_representation_a	**SIGNED**, **UNSIGNED**, **VARIABLE**	**UNSIGNED**	Specify the representation format for the multiplier A input.
Register 'signa' input	gui_register_signa	On Off	Off	Select this option to enable signa register. You must select **VARIABLE** value for **What is the representation format for Multipliers A inputs?** parameter to enable this option.
What is the source for clock input?	gui_register_signa_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the input clock signal for signa register. You must select **Register 'signa' input** to enable this parameter.
What is the source for asynchronous clear input?	gui_register_signa_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the signa register. You must select **Register 'signa' input** to enable this parameter.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
What is the source for synchronous clear input?	gui_register_signa_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `signa` register. You must select **Register 'signa' input** to enable this parameter.
What is the representation format for Multipliers B inputs?	gui_representation_b	**SIGNED**, **UNSIGNED**, **VARIABLE**	**UNSIGNED**	Specify the representation format for the multiplier B input.
Register 'signb' input	gui_register_signb	On Off	Off	Select this option to enable `signb` register. You must select **VARIABLE** value for **What is the representation format for Multipliers B inputs?** parameter to enable this option.
What is the source for clock input?	gui_register_signb_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the input clock signal for `signb` register. You must select **Register 'signb' input** to enable this parameter.
What is the source for asynchronous clear input?	gui_register_signb_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `signb` register. You must select **Register 'signb' input** to enable this parameter.
What is the source for synchronous clear input?	gui_register_signb_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `signb` register. You must select **Register 'signb' input** to enable this parameter.
Input Configuration				
Register input A of the multiplier	gui_input_register_a	On Off	Off	Select this option to enable input register for `dataa` input bus.
What is the source for clock input?	gui_input_register_a_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the register input clock signal for `dataa` input bus. You must select **Register input A of the multiplier** to enable this parameter.
What is the source for asynchronous clear input?	gui_input_register_a_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the register asynchronous clear source for the `dataa` input bus. You must select **Register input A of the multiplier** to enable this parameter.
What is the source for synchronous clear input?	gui_input_register_a_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the register synchronous clear source for the `dataa` input bus. You must select **Register input A of the multiplier** to enable this parameter.
Register input B of the multiplier	gui_input_register_b	On Off	Off	Select this option to enable input register for `datab` input bus.
What is the source for clock input?	gui_input_register_b_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the register input clock signal for `datab` input bus. You must select **Register input B of the multiplier** to enable this parameter.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
What is the source for asynchronous clear input?	gui_input_register_b_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the register asynchronous clear source for the `datab` input bus. You must select **Register input B of the multiplier** to enable this parameter.
What is the source for synchronous clear input?	gui_input_register_b_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the register synchronous clear source for the `datab` input bus. You must select **Register input B of the multiplier** to enable this parameter.
What is the input A of the multiplier connected to?	gui_multiplier_a_input	**Multiplier input** **Scan chain input**	**Multiplier input**	Select the input source for input A of the multiplier. Select **Multiplier input** to use `dataa` input bus as the source to the multiplier. Select **Scan chain input** to use `scanin` input bus as the source to the multiplier and enable the `scanout` output bus. This parameter is available when you select **2**, **3** or **4** for **What is the number of multipliers?** parameter.
Scanout A Register Configuration				
Register output of the scan chain	gui_scanouta_register	On Off	Off	Select this option to enable output register for `scanouta` output bus. You must select **Scan chain input** for **What is the input A of the multiplier connected to?** parameter to enable this option.
What is the source for clock input?	gui_scanouta_register_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the register input clock signal for `scanouta` output bus. You must turn on **Register output of the scan chain** parameter to enable this option.
What is the source for asynchronous clear input?	gui_scanouta_register_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the register asynchronous clear source for the `scanouta` output bus. You must turn on **Register output of the scan chain** parameter to enable this option.
What is the source for synchronous clear input?	gui_scanouta_register_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the register synchronous clear source for the `scanouta` output bus. You must select **Register output of the scan chain** parameter to enable this option.

8.6.4. Preadder Tab

Table 33. Preadder Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Select preadder mode	preadder_mode	**SIMPLE**, **COEF**, **INPUT**, **SQUARE**, **CONSTANT**	**SIMPLE**	Specifies the operation mode for preadder module. **SIMPLE**: This mode bypass the preadder. This is the default mode.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				COEF: This mode uses the output of the preadder and `coefsel` input bus as the inputs to the multiplier. **INPUT**: This mode uses the output of the preadder and `datac` input bus as the inputs to the multiplier. **SQUARE**: This mode uses the output of the preadder as both the inputs to the multiplier. **CONSTANT**: This mode uses `dataa` input bus with preadder bypassed and `coefsel` input bus as the inputs to the multiplier.
Select preadder direction	gui_preadder_direction	**ADD**, **SUB**	**ADD**	Specifies the operation of the preadder. To enable this parameter, select the following for **Select preadder mode**: • **COEF** • **INPUT** • **SQUARE** or • **CONSTANT**
How wide should the C input buses be?	width_c	1 - 256	16	Specifies the number of bits for C input bus. You must select **INPUT** for **Select preadder mode** to enable this parameter.
Data C Input Register Configuration				
Register datac input	gui_datac_input_register	On Off	On	Select this option to enable input register for `datac` input bus. You must set **INPUT** to **Select preadder mode** parameter to enable this option.
What is the source for clock input?	gui_datac_input_register_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to specify the input clock signal for `datac` input register. You must select **Register datac input** to enable this parameter.
What is the source for asynchronous clear input?	gui_datac_input_register_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `datac` input register. You must select **Register datac input** to enable this parameter.
What is the source for synchronous clear input?	gui_datac_input_register_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `datac` input register. You must select **Register datac input** to enable this parameter.
Coefficients				
How wide should the coef width be?	width_coef	1 - 27	18	Specifies the number of bits for `coefsel` input bus. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
Coef Register Configuration				
Register the coefsel input	gui_coef_register	On Off	On	Select this option to enable input register for `coefsel` input bus.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
What is the source for clock input?	gui_coef_register_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to specify the input clock signal for `coefsel` input register. You must select **Register the coefsel input** to enable this parameter.
What is the source for asynchronous clear input?	gui_coef_register_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `coefsel` input register. You must select **Register the coefsel input** to enable this parameter.
What is the source for synchronous clear input	gui_coef_register_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `coefsel` input register. You must select **Register the coefsel input** to enable this parameter.
Coefficient_0 Configuration	coef0_0 to coef0_7	0x00000 – 0xFFFFFFF	0x00000000	Specifies the coefficient values for this first multiplier. The number of bits must be the same as specified in **How wide should the coef width be?** parameter. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
Coefficient_1 Configuration	coef1_0 to coef1_7	0x00000 – 0xFFFFFFF	0x00000000	Specifies the coefficient values for this second multiplier. The number of bits must be the same as specified in **How wide should the coef width be?** parameter. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
Coefficient_2 Configuration	coef2_0 to coef2_7	0x00000 – 0xFFFFFFF	0x00000000	Specifies the coefficient values for this third multiplier. The number of bits must be the same as specified in **How wide should the coef width be?** parameter. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
Coefficient_3 Configuration	coef3_0 to coef3_7	0x00000 – 0xFFFFFFF	0x00000000	Specifies the coefficient values for this fourth multiplier. The number of bits must be the same as specified in **How wide should the coef width be?** parameter. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.

8.6.5. Accumulator Tab

Table 34. Accumulator Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Enable accumulator?	accumulator	**YES**, **NO**	**NO**	Select **YES** to enable the accumulator.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				You must select **Register output of adder unit** when using accumulator feature.
What is the accumulator operation type?	accum_direction	**ADD**, **SUB**	**ADD**	Specifies the operation of the accumulator: • **ADD** for addition operation • **SUB** for subtraction operation. You must select **YES** for **Enable accumulator?** parameter to enable this option.
Preload Constant				
Enable preload constant	gui_ena_preload_const	On Off	Off	Enable the `accum_sload` or `sload_accum` signals and register input to dynamically select the input to the accumulator. When `accum_sload` is low or `sload_accum`, the multiplier output is feed into the accumulator. When `accum_sload` is high or `sload_accum`, a user specified preload constant is feed into the accumulator. You must select **YES** for **Enable accumulator?** parameter to enable this option.
What is the input of accumulate port connected to?	gui_accumulate_port_select	**ACCUM_SLOAD**, **SLOAD_ACCUM**	**ACCUM_SLOAD**	Specifies the behavior of `accum_sload`/`sload_accum` signal. **ACCUM_SLOAD**: Drive `accum_sload` low to load the multiplier output to the accumulator. **SLOAD_ACCUM**: Drive `sload_accum` high to load the multiplier output to the accumulator. You must select **Enable preload constant** option to enable this parameter.
Select value for preload constant	loadconst_value	0 - 64	64	Specify the preset constant value. This value can be 2^N where N is the preset constant value. When N=64, it represents a constant zero. You must select **Enable preload constant** option to enable this parameter.
What is the source for clock input?	gui_accum_sload_register_clock	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to specify the input clock signal for `accum_sload/sload_accum` register. You must select **Enable preload constant** option to enable this parameter.
What is the source for asynchronous clear input?	gui_accum_sload_register_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `accum_sload/sload_accum` register.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				You must select **Enable preload constant** option to enable this parameter.
What is the source for synchronous clear input?	gui_accum_sload_register_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `accum_sload/sload_accum` register. You must select **Enable preload constant** option to enable this parameter.
Enable double accumulator	gui_double_accum	On Off	Off	Enables the double accumulator register.

8.6.6. Systolic/Chainout Tab

Table 35. Systolic/Chainout Adder Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Enable chainout adder	chainout_adder	**YES**, **NO**	**NO**	Select **YES** to enable chainout adder module.
What is the chainout adder operation type?	chainout_adder_direction	**ADD**, **SUB**	**ADD**	Specifies the chainout adder operation. For subtraction operation, **SIGNED** must be selected for **What is the representation format for Multipliers A inputs?** and **What is the representation format for Multipliers B inputs?** in the Multipliers Tab.
Enable 'negate' input for chainout adder?	Port_negate	**PORT_USED**, **PORT_UNUSED**	**PORT_UNUSED**	Select **PORT_USED** to enable `negate` input signal. This parameter is invalid when chainout adder is disabled.
Register 'negate' input?	negate_register	**UNREGISTERED**, **CLOCK0**, **CLOCK1**, **CLOCK2**, **CLOCK3**	**UNREGISTERED**	To enable the input register for `negate` input signal and specifies the input clock signal for `negate` register. Select **UNREGISTERED** if the `negate` input register to is not needed This parameter is invalid when you select: • **NO** for **Enable chainout adder** or • **PORT_UNUSED** for **Enable 'negate' input for chainout adder?** parameter or
What is the source for asynchronous clear input?	negate_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `negate` register. This parameter is invalid when you select: • **NO** for **Enable chainout adder** or • **PORT_UNUSED** for **Enable 'negate' input for chainout adder?** parameter or
What is the source for synchronous clear input?	negate_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `negate` register. This parameter is invalid when you select:

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				• **NO** for **Enable chainout adder** or • **PORT_UNUSED** for **Enable 'negate' input for chainout adder?** parameter or
Systolic Delay				
Enable systolic delay registers	gui_systolic_delay	On Off	Off	Select this option to enable systolic mode. This parameter is available when you select **2**, or **4** for **What is the number of multipliers?** parameter. You must enable the **Register output of the adder unit** to use the systolic delay registers.
What is the source for clock input?	gui_systolic_delay_clock	**CLOCK0**, **CLOCK1**, **CLOCK2**,	**CLOCK0**	Specifies the input clock signal for systolic delay register. You must select **enable systolic delay registers** to enable this option.
What is the source for asynchronous clear input?	gui_systolic_delay_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the systolic delay register. You must select **enable systolic delay registers** to enable this option.
What is the source for synchronous clear input?	gui_systolic_delay_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the systolic delay register. You must select **enable systolic delay registers** to enable this option.

8.6.7. Pipelining Tab

Table 36. Pipelining Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Pipelining Configuration				
Do you want to add pipeline register to the input?	gui_pipelining	**No**, **Yes**	**No**	Select **Yes** to enable an additional level of pipeline register to the input signals. You must specify a value greater than 0 for **Please specify the number of latency clock cycles** parameter.
Please specify the number of latency clock cycles	latency	Any value greater than 0	0	Specifies the desired latency in clock cycles. One level of pipeline register = 1 latency in clock cycle. You must select **YES** for **Do you want to add pipeline register to the input?** to enable this option.
What is the source for clock input?	gui_input_latency_clock	**CLOCK0**, **CLOCK1**, **CLOCK2**	**CLOCK0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the pipeline register input clock signal.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				You must select **YES** for **Do you want to add pipeline register to the input?** to enable this option.
What is the source for asynchronous clear input?	gui_input_latency_aclr	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the register asynchronous clear source for the additional pipeline register. You must select **YES** for **Do you want to add pipeline register to the input?** to enable this option.
What is the source for synchronous clear input?	gui_input_latency_sclr	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the register synchronous clear source for the additional pipeline register. You must select **YES** for **Do you want to add pipeline register to the input?** to enable this option.

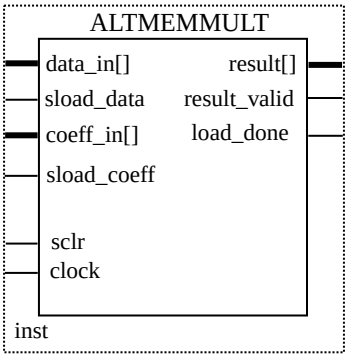
9. ALTMEMMULT (Memory-based Constant Coefficient Multiplier) IP Core

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

The ALTMEMMULT IP core is a synchronous function that requires a clock. The ALTMEMMULT IP core implements a multiplier with the smallest throughput and latency possible for a given set of parameters and specifications.

The following figure shows the ports for the ALTMEMMULT IP core.

Figure 21. ALTMEMMULT Ports



Related Information

Features on page 71

9.1. Features

The ALTMEMMULT IP core offers the following features:

- Creates only memory-based multipliers using on-chip memory blocks found in Intel 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

9.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the <Intel Quartus Prime 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
```

9.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <Intel Quartus Prime 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;
```

9.4. Ports

The following tables list the input and output ports for the ALTMEMMULT IP core.

Table 37. ALTMEMMULT 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 38. ALTMEMMULT 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.

9.5. Parameters

The following table lists the parameters for the ALTMEMMULT IP core.

Table 39. ALTMEMMULT Parameters

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.

continued...

Parameter Name	Type	Required	Description
COEFFICIENT0	Integer	Yes	Specifies value of the first fixed coefficient.
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 data_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.
LPM_HINT	String	No	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (.vhd), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES" 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.

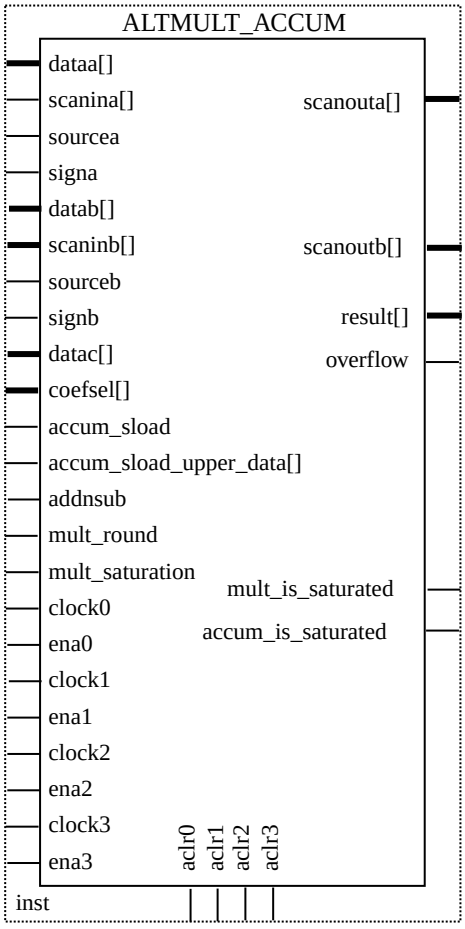
10. ALTMULT_ACCUM (Multiply-Accumulate) IP Core

The ALTMULT_ACCUM IP core allows you to implement a multiplier-adder.

Note: This IP core is not supported in Arria V, Intel Arria 10, Cyclone V, Intel Cyclone 10 GX, and Stratix V devices and will be replaced by Intel FPGA Multiply Adder or ALTERA_MULT_ADD IP core.

The following figure shows the ports for the ALTMULT_ACCUM IP core.

Figure 22. ALTMULT_ACCUM Ports



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 the following equation.

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

Where *N* is the number of cycles of data that has been entered into the accumulator.

Related Information

Features on page 71

10.1. Features

The ALTMULT_ACCUM IP core 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)

  Note: When building multipliers larger than the natively supported size there may/will be a performance impact resulting from the cascading of the DSP blocks.
- 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 optional asynchronous clear and clock enable input ports

Related Information

- Intel FPGA Multiply Adder IP Core on page 36
- ALTMEMMULT (Memory-based Constant Coefficient Multiplier) IP Core on page 57
- LPM_MULT (Multiplier) IP Core on page 16

10.2. Verilog HDL Prototype

To view the Verilog HDL prototype for the IP core, refer to the Verilog Design File (**.v**) **altera_mf.v** in the `<Intel Quartus Prime installation directory>\eda\synthesis` directory.

10.3. VHDL Component Declaration

To view the VHDL component declaration for the IP core, refer to the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <Intel Quartus Prime installation directory>\libraries\vhdl\altera_mf directory.

10.4. 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;
```

10.5. Ports

The following tables list the input and output ports for the ALTMULT_ACCUM IP core.

Table 40. ALTMULT_ACCUM 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 IP core.
clock1	No	Specifies the second clock input, usable by any register in the IP core.
clock2	No	Specifies the third clock input, usable by any register in the IP core.
clock3	No	Specifies the fourth clock input, usable by any register in the IP core.
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.

continued...

Port Name	Required	Description
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.

Table 41. ALTMULT_ACCUM Input Ports (HardCopy Devices Only)

Port Name	Required	Description
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.

Table 42. ALTMULT_ACCUM Input Ports (Stratix IV Devices Only)

Port Name	Required	Description
accum_round	No	Enables accumulator rounding.

Table 43. ALTMULT_ACCUM 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. The parameter editor 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. The parameter editor renames the scanoutb[] port to shiftoutb port.

10.6. Parameters

The following table lists the parameters for the ALTMULT_ACCUM IP core.

Table 44. ALTMULT_ACCUM Parameters

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

continued...

Parameter Name	Type	Required	Description
			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.
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

continued...

Parameter Name	Type	Required	Description
			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. The parameter editor calculates the value for this parameter.
LPM_HINT	String	No	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (.vhd), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES" The default value is UNUSED.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL design files.
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. You must enable the output registers in order to use accumulator.
PORT_ADDNSUB	String	No	Specifies the usage of the addnsub input port. Values are: PORT_USED , PORT_UNUSED, and PORT_CONNECTIVITY (port usage is determined by checking the port connectivity.) If omitted the default value is PORT_CONNECTIVITY.

continued...

Parameter Name	Type	Required	Description
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.
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.

Table 45. ALTMULT_ACCUM Parameters (HardCopy Devices Only)

Parameter Name	Type	Required	Description
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.

Table 46. ALTMULT_ACCUM Parameters (Stratix IV, Arria GX, and HardCopy Devices Only)

Parameter Name	Type	Required	Description
MULTIPLIER_SATURATION	String	No	Specifies multiplier saturation. Values are NO, YES, and VARIABLE. If omitted the default value is NO.

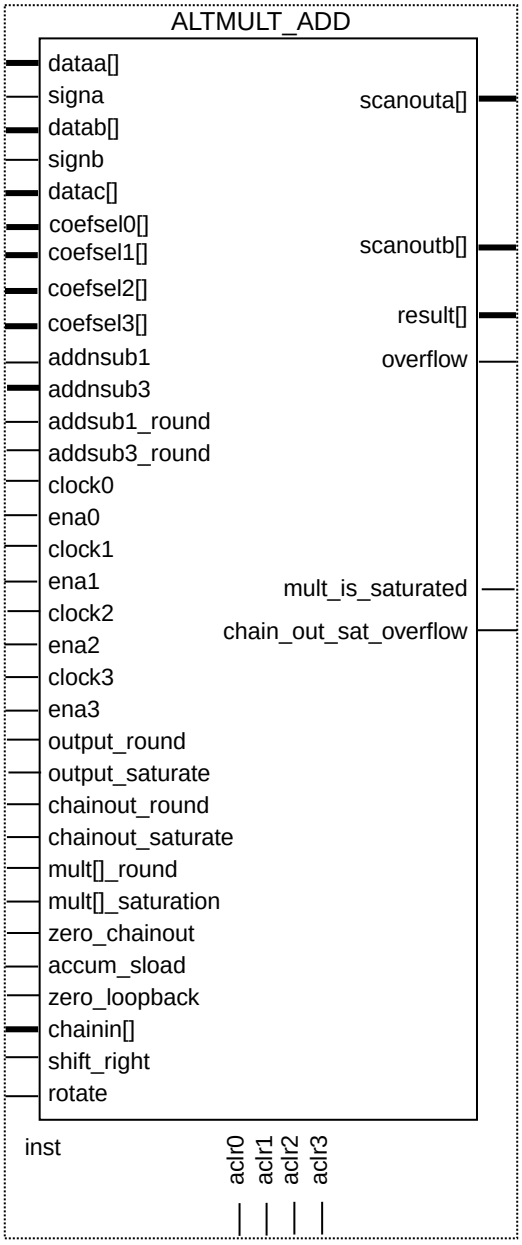
11. ALTMULT_ADD (Multiply-Adder) IP Core

The ALTMULT_ADD IP core allows you to implement a multiplier-adder.

Note: This IP core is not supported in Arria V, Intel Arria 10, Cyclone V, Intel Cyclone 10 GX, and Stratix V devices. For the mentioned devices, refer to Intel FPGA Multiply Adder or ALTERA_MULT_ADD (Multiply-Adder) IP core.

The following figure shows the ports for the ALTMULT_ADD IP core.

Figure 23. ALTMULT_ADD Ports



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

The ALTMULT_ADD IP core also offers many variations in dedicated DSP block circuitry. 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 IV device families use two MAC blocks (mac_mult and mac_out) to form DSP operations, multiply and add.

The multipliers and adders of the ALTMULT_ADD IP core are placed in the dedicated DSP block circuitry of the Stratix IV 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 IP cores 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.

11.1. Features

The ALTMULT_ADD IP core 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)

  Note: When building multipliers larger than the natively supported size there may/ will be a performance impact resulting from the cascading of the DSP blocks.
- 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

Related Information

- ALTMULT_ACCUM (Multiply-Accumulate) IP Core on page 61
- ALTMEMMULT (Memory-based Constant Coefficient Multiplier) IP Core on page 57

- LPM_MULT (Multiplier) IP Core on page 16

11.2. Verilog HDL Prototype

To view the Verilog HDL prototype for the IP core, refer to the Verilog Design File (**.v**) **altera_mf.v** in the <Intel Quartus Prime installation directory>\eda\synthesis directory.

11.3. VHDL Component Declaration

To view the VHDL component declaration for the IP core, refer to the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <Intel Quartus Prime installation directory>\libraries\vhdl\altera_mf directory.

11.4. 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;
```

11.5. Ports

The following tables list the input and output ports for the ALTMULT_ADD IP core.

Table 47. ALTMULT_ADD Input Ports

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 IP core.
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.

Table 48. ALTMULT_ADD Input Ports (Stratix IV Devices Only)

Port Name	Required	Description
Ports Available in 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.

Table 49. ALTMULT_ADD 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. Do not use scanina[] and scaninb[] simultaneously.
scanoutb[]	No	Output of scan chain B. Output port [WIDTH_B - 1..0] wide. Do not use scanina[] and scaninb[] simultaneously.

Table 50. ALTMULT_ADD Output Ports (Stratix IV Devices Only)

Port Name	Required	Description
chainout_sat_overflow	No	Overflow flag for the chainout saturation.

11.6. Parameters

The following table lists the parameters for the ALTMULT_ADD IP core.

Note: For 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 51. ALTMULT_ADD Parameters

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_A[0 ... 3]	String	No	Specifies the clock port for the dataa[] operand of the multiplier. Values are CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0.
INPUT_REGISTER_B[0 ... 3]	String	No	Specifies the clock port for the datab[] operand of the first multiplier. Values are CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0.
INPUT_ACLR_A[0 ... 3]	String	No	Specifies the asynchronous clear for the dataa[] operand of the first multiplier. Values are ACLR0, ACLR1, ACLR2, ACLR3, and NONE. If omitted , the default value is NONE. The INPUT_ACLR_A[1 ... 3] values must be set similar to the value of INPUT_ACLR_A0.
INPUT_ACLR_B[0 ... 3]	String	No	Specifies the asynchronous clear for the datab[] operand of the first multiplier. Values are ACLR0, ACLR1, ACLR2, ACLR3, and NONE. If omitted, the default value is NONE. The INPUT_ACLR_B[1 ... 3] values must be set similar to the value of INPUT_ACLR_B0.
INPUT_SOURCE_A[0 ... 3]	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.
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.
INPUT_SOURCE_B1	String	No	Specifies the data source of the second multiplier. Values are DATAB and SCANB. If this parameter is set to

continued...

Parameter Name	Type	Required	Description
			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.
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.
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.
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.
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.
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. If the

continued...

Parameter Name	Type	Required	Description
			corresponding sign[] port value is UNUSED, this parameter is ignored. If omitted, the default value is CLOCK0.
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. 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.
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.
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.
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.
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. 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. If the addnsub3 port is used, this parameter is ignored. If omitted, the default value is ADD.

continued...

Parameter Name	Type	Required	Description
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. 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 Intel-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. Instantiate the ALTMULT_ADD IP core through the IP Catalog to calculate the value for this parameter.
DSP_BLOCK_BALANCING	String	No	If omitted, the default value is AUTO.
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. If omitted, the default value is AUTO.

Table 52. ALTMULT_ADD Parameters (Stratix IV Devices Only)

Parameter Name	Type	Required	Description
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. 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. 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. If omitted, the default value is NO.
ACCUMULATOR	String	No	Specifies the accumulator mode of the final adder stage. Values are YES and NO. If omitted, the default value is NO. When value is set to YES, rounding is dynamic and you must initialize the accumulator while rounded data is acquired.
WIDTH_CHAININ	Integer	No	Width of the chainin[] port. WIDTH_CHAININ equals WIDTH_RESULT if port chainin is used. If omitted, the default value is 1.
OUTPUT_ROUNDING	String	No	Enables rounding handling at second adder 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.
OUTPUT_ROUND_TYPE	String	No	Specifies the rounding mode. Values are NEAREST_EVEN and NEAREST_INTEGER. A value of NEAREST_EVEN specifies round-to-nearest-even. A value of NEAREST_INTEGER specifies round-to-nearest-integer. 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. If omitted, the default value is CLOCK0.

continued...

Parameter Name	Type	Required	Description
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. 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. 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. If omitted and OUTPUT_ROUND_PIPELINE_REGISTER is used, the default value is ACLR3.
OUTPUT_SATURATION	String	No	Enables saturation handling at second adder 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. 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. 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. 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. 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. If omitted and OUTPUT_SATURATE_PIPELINE_REGISTER is used, the default value is ACLR3.

continued...

Parameter Name	Type	Required	Description
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. If the value of CHAINOUT_ROUNDING is YES, the symmetric saturation at the second adder output stage is not allowed. 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. 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. If omitted and CHAINOUT_ROUND_REGISTER is used, the default value is ACLR3.
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. 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. 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. 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. 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

continued...

Parameter Name	Type	Required	Description
			VARIABLE allows dynamically controlled saturation handling. 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. 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. 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. 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. 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. If omitted and CHAINOUT_SATURATE_OUTPUT_REGISTER is used, the default value is ACLR3.
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. 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. 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. 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

continued...

Parameter Name	Type	Required	Description
			ACLR0, ACLR1, ACLR2, and ACLR3. 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. 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. 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. 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. 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. 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. If omitted and ACCUM_SLOAD_REGISTER is used, the default value is ACLR3.
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. 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. If omitted and ACCUM_SLOAD_PIPELINE_REGISTER is used, the default value is ACLR3.

continued...

Parameter Name	Type	Required	Description
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. If omitted, the default value is NO. 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. 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. 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. 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. 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. 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. If omitted and ROTATE_OUTPUT_REGISTER is used, the default value is ACLR3.
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. 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. If omitted and SHIFT_RIGHT_REGISTER is used, the default value is ACLR3.

continued...

Parameter Name	Type	Required	Description
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. 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. 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. 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. 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. 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. If omitted, the default value is PORT_UNUSED.
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.
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.
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,

Parameter Name	Type	Required	Description
			ACLR2, and ACLR3. If omitted and CHAINOUT_REGISTER is used, the default value is ACLR3.

12. ALTMULT_COMPLEX (Complex Multiplier) IP Core

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

- Canonical

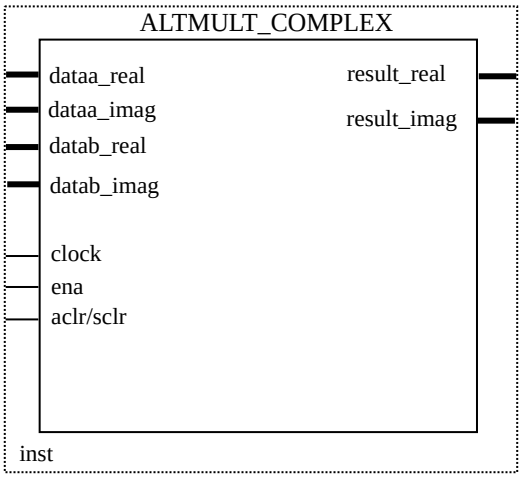
  You can use the canonical representation for all supported Intel devices prior to Stratix III devices with input data widths of less than 18 bits.

- Conventional

  You can use the ALTMULT_ADD IP core to implement the complex multiplier by instantiating two multipliers.

The following figure shows the ports for the ALTMULT_COMPLEX IP core.

Figure 24. ALTMULT_COMPLEX Ports



12.1. 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:√-1

Two complex numbers, *x* = *a* + *ib* and *y* = *c* + *id* are multiplied, as shown in the following equations.

Figure 25. Equation for Two Complex Numbers Multiplication

$$x * y = (a + ib)(c + id)$$
$$= ac + ibc + iad - bd$$
$$= (ac - bd) + i(ad + bc)$$

Related Information

Canonical Representation on page 87

12.2. Canonical Representation

From Complex Multiplication equation in Figure 25 on page 87, the multiplication of two complex numbers can be represented in two parts: real and imaginary.

The following equation shows that the *xy_real* variable represents real representation.

Figure 26. Real Representation

$$xy\_real = ac - bd$$
$$= ac - bd + (ad - bc) - (ad - bc)$$
$$= (ac - ad + bc - bd) + (ad - bc)$$
$$= ((a + b)(c - d)) + (ad - bc)$$

$$xy\_real \text{ represents the real part}$$

The following equation shows that the *xy_imaginary* variable represents imaginary representation.

Figure 27. Imaginary Representation

$$xy\_imaginary = ad + bc$$

$$xy\_imaginary \text{ represents imaginary}$$

Both equations derived from Complex Multiplication equation.

Note: The canonical representation is available for all supported Intel devices prior to Stratix III devices.

Related Information

Complex Multiplication on page 86

12.3. Conventional Representation

The multiplication of two complex numbers can be represented in two parts, real and imaginary. The *xy_real* variable in the following equation represents the real part:

xy_real = ac – bd

The *xy_imaginary* variable in the following equation represents the imaginary part.

xy_imaginary = ad + bc

12.4. Features

The ALTMULT_COMPLEX IP core offers the following features:

- Generates a multiplier to perform multiplication operations of two complex numbers

  Note: When building multipliers larger than the natively supported size there may/will be a performance impact resulting from the cascading of the DSP blocks.
- 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
- Supports optional synchronous clear for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices
- Provides an option to dynamically switch between 36 × 36 normal mode and 18 × 18 complex mode (for Stratix V devices only)

12.5. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the `<Intel Quartus Prime 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
```

12.6. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <Intel Quartus Prime 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;
```

12.7. 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;
```

12.8. Signals

Table 53. ALTMULT_COMPLEX Input Signals

Signal	Required	Description
aclr	No	Asynchronous clear for the complex multiplier. When the aclr signal is asserted high, the function is asynchronously cleared.
sclr	No	Synchronous clear for the complex multiplier. When the sclr signal is asserted high, the function is asynchronously cleared. Only available for Intel Stratix 10, Intel Arria 10 and Intel Cyclone 10 GX devices.
clock	Yes	Clock input to the ALTMULT_COMPLEX function.
dataa_imag[]	Yes	Imaginary input value for the data A signal of the complex multiplier. The size of the input signal depends on the WIDTH_A parameter value.
dataa_real[]	Yes	Real input value for the data A signal of the complex multiplier. The size of the input signal depends on the WIDTH_A parameter value.

continued...

Signal	Required	Description
datab_imag[]	Yes	Imaginary input value for the data B signal of the complex multiplier. The size of the input signal depends on the WIDTH_B parameter value.
datab_real[]	Yes	Real input value for the data B signal of the complex multiplier. The size of the input signal depends on the WIDTH_B parameter value.
ena	No	Active high clock enable for the clock signal of the complex multiplier.
complex	No	Optional input to enable dynamic switching between 36 × 36 normal model and 18 × 18 complex mode. This input is only available in Stratix V devices. In the GUI, this parameter is referred as Dynamic Complex Mode.

Table 54. ALTMULT_COMPLEX Output Signals

Signal	Required	Description
result_imag	Yes	Imaginary output value of the multiplier. The size of the output signal depends on the WIDTH_RESULT parameter value.
result_real	Yes	Real output value of the multiplier. The size of the output signal depends on the WIDTH_RESULT parameter value.

12.9. Parameters

The following table lists the parameters for the ALTMULT_COMPLEX IP core.

Table 55. ALTMULT_COMPLEX Parameters

Parameter	IP Generated Parameter	Value	Default Value	Description
General				
How wide should the A input buses be?	WIDTH_A	1–256	18	Specifies the number of bits for A input buses.
How wide should the B input buses be?	WIDTH_B	1–256	18	Specifies the number of bits for B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	1–256	36	Specifies the number of bits for 'result' output bus.
Input Representation				
What is the representation format for A inputs?	REPRESENTATION_A	**Signed**, **Unsigned**	**Signed**	Specifies the representation format for A inputs. Arria V, Intel Arria 10, Cyclone V, Intel Cyclone 10 GX, Intel Stratix 10, and Stratix V devices support only signed input representation format.
What is the representation format for B inputs?	REPRESENTATION_B	**Signed**, **Unsigned**	**Signed**	Specifies the representation format for B inputs. Arria V, Intel Arria 10, Cyclone V, Intel Cyclone 10 GX, Intel Stratix 10, and Stratix V devices support only signed input representation format.
Complex Multiplier Option				
Dynamic Complex Mode	GUI_DYNAMIC_COMPLEX	—	Unchecked	Enable dynamic switching between 36 x 36 normal mode and 18 x 18 complex mode.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				Available only in Stratix V devices.
Implementation Style				
Which implementation style should be used?	IMPLEMENTATION_STYLE	**Automatically select a style for best trade-off for the current settings** **Canonical (Minimize the number of simple multipliers)** **Conventional (Minimize the use of logic cells)**	**Automatically select a style for best trade-off for the current settings**	Arria V, Intel Arria 10, Cyclone V, Intel Cyclone 10 GX, Intel Stratix 10, and Stratix V devices support only **Automatically select a style for best trade-off for the current settings** style. The Intel Quartus Prime software determines the best implementation based on the selected device family and input width.
Pipelining (Only available for Arria 10 and Cyclone 10 GX devices)				
Output latency	PIPELINE	0–11	4	Specifies the number of clock cycles for output latency.
Create a Clear input?	CLEAR_TYPE	**NONE** **ACLR** **SCLR**	**NONE**	Select this option to create `aclr` or `sclr` signal for the complex multiplier.
Create a Clock Enable input?	GUI_USE_CLKEN	—	Unchecked	Select this option to create `ena` signal for the complex multiplier clock.

Related Information

Does the ALTMULT_COMPLEX IP support unsigned operation in V-series and 10-series device families?

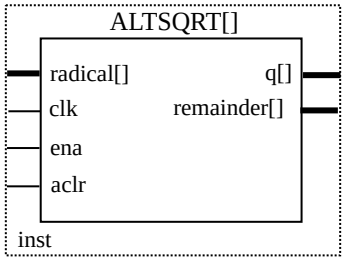
Solution to enable V-series and 10-series devices to support signed operation.

13. ALTSQRT (Integer Square Root) IP Core

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

The following figure shows the ports for the ALTSQRT IP core.

Figure 28. ALTSQRT Ports



13.1. Features

The ALTSQRT IP core 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

13.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the `<Intel Quartus Prime 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
```

13.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <Intel Quartus Prime 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;
```

13.4. 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;
```

13.5. Ports

The following tables list the input and output ports for the ALTSQRT IP core.

Table 56. ALTSQRT 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 IP core. 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 57. ALTSQRT 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.

13.6. Parameters

The following table lists the parameters for the ALTSQRT IP core.

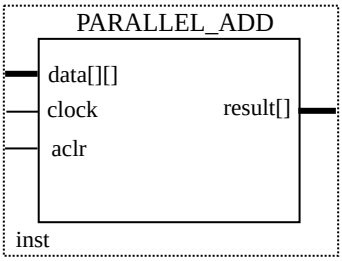
Parameter Name	Type	Required	Description
WIDTH	Integer	Yes	Specifies the widths of the `radical[]` input port.
Q_PORT_WIDTH	Integer	Yes	Specifies the width of the `q[]` output port.
R_PORT_WIDTH	Integer	Yes	Specifies the width of the `remainder[]` output port.
PIPELINE	Integer	No	Specifies the number of clock cycles of latency to add.
LPM_HINT	String	No	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (`.vhd`), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: `LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES"` The default value is UNUSED.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL design files.

14. PARALLEL_ADD (Parallel Adder) IP Core

The PARALLEL_ADD IP core 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.

The following figure shows the ports for the PARALLEL_ADD IP core.

Figure 29. PARALLEL_ADD Ports



14.1. Feature

The PARALLEL_ADD IP core 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

14.2. Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (**.v**) **altera_mf.v** in the `<Intel Quartus Prime 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
```

14.3. VHDL Component Declaration

The VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the `<Intel Quartus Prime 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;
```

14.4. 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;
```

14.5. Ports

The following tables list the input and output ports of the PARALLEL_ADD IP core.

Table 58. PARALLEL_ADD 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 59. PARALLEL_ADD Output Ports

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

14.6. Parameters

The following table lists the parameters for the PARALLEL_ADD IP core.

Table 60. PARALLEL_ADD 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. The parameter editor calculates this value.
LPM_HINT	String	No	When you instantiate a library of parameterized modules (LPM) function in a VHDL Design File (.vhd), you must use the LPM_HINT parameter to specify an Intel-specific parameter. For example: LPM_HINT = "CHAIN_SIZE = 8, ONE_INPUT_IS_CONSTANT = YES" The default value is UNUSED.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL design files.

15. Integer Arithmetic IP Cores User Guide Document Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
17.0	Integer Arithmetic IP Cores User Guide
16.0	Integer Arithmetic IP Cores User Guide
15.1	Integer Arithmetic IP Cores User Guide
14.1	Integer Arithmetic IP Cores User Guide

16. Document Revision History for Intel FPGA Integer Arithmetic IP Cores User Guide

Document Version	Intel Quartus Prime Version	Changes
2020.04.30	17.1	• Updated values for **Which multiplier implementation should be used?** parameter for LPM_MULT IP core in the *Parameters for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices* topic.

Date	Version	Changes
November 2017	2017.11.24	• Rebranded ALTERA_MULT_ADD IP core name to Intel FPGA Multiply Adder for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices. • Reverted the behavior of `sload_accum` and `accum_sload` signals in *Intel FPGA Multiply Adder or ALTERA_MULT_ADD Input Signals*. • Added Intel Stratix 10 devices in the *List of IP Cores* table. • Removed *Design Example Files* section. • Added a note for the first multiplier `chainin` signal in Systolic Delay Register on page 40.
June 2017	2017.06.19	• Rebranded as Intel. • Added support for Cyclone 10 GX and Cyclone 10 LP devices. • Removed outdated design example information. • Updated `sload_accum` and `accum_sload` signals behavior in the ALTERA_MULT_ADD Input Signals table.
June 2016	2016.06.10	• Added separate parameters table for Arria 10 devices. • Replaced ip-generated parameter names with GUI parameter names for LPM_MULT, ALTERA_MULT_ADD and ALTMULT_COMPLEX IP cores. • Added synchronous clear support for input, pipeline, and output registers LPM_MULT in Arria 10 devices, ALTERA_MULT_ADD for all devices and ALTMULT_COMPLEX for Arria 10 devices. • Added new parameters in LPM_MULT and ALTMULT_COMPLEX IP cores (Arria 10 devices) and ALTERA_MULT_ADD (for all devices) to enable users to select synchronous clear feature. • Removed resource tables for all Integer Arithmetic IP cores.
November 2015	2015.11.18	• Corrected LPM_COUNTER VHDL component declaration. • Added device support list to List of IP Cores table. • Removed Stratix V, Arria V and Cyclone V devices support for ALTMULT_ACCUM and ALTMULT_ADD IP cores. • Removed Cyclone II, Cyclone III, Stratix II, and Stratix III because these devices are no longer supported for all the integer arithmetic IP cores. • Change `COEFFSEL[]_REGISTER` parameter name to `COEFSEL[]_REGISTER` and `COEFFSEL[]_ACLR` parameter name to `COEFSEL[]_ACLR`. • Added description in `REPRESENTATION_A` and `REPRESENTATION_B` parameters to clarify only signed input representation is supported for Stratix V, Arria V, Cyclone V, and Arria 10 devices. • Added LPM_ADD_SUB and LPM_COMPARE IP cores information. • Added links to Introduction to Altera IP Cores, Creating Version-Independent IP and Qsys Simulation Scripts, and Project Management Best Practices. • Changed instances of Quartus II to Quartus Prime.

continued...

Date	Version	Changes
December, 2014	2014.12.19	• Removed the LPM_ADD_SUB and LPM_COMPARE IPs because these IPs are no longer supported. • Added a note to clarify that when building multipliers larger than the natively supported size there may be a performance impact resulting from the cascading of the DSP blocks in LPM_MULT, ALTERA_MULT_ADD, ALTMULT_ACCUM, ALTMULT_ADD, and ALTMULT_COMPLEX IP cores. • Added information about the **Create a 'sync_e' port** parameter and the sync_e signal for ALTECC_DECODER IP core. • Removed sconst port information as the port is no longer available for LPM_COUNTER IP core. • Provided an example to use the LPM_HINT parameter.
August, 2014	2014.08.18	• Updated parameterization steps for legacy and latest parameter editors. • Added note for IP cores that do not support Arria 10 designs. • Added device migration information.
June 2014	5.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor.
June 2013	4.0	• Added Intel FPGA Multiply Adder IP Core on page 36 section. • Removed the following obsoleted megafunctions: LPM_ABS, ALTACCUMULATE, ALTMULT_ACCUM, ALTMULT_ADD. • Updated ALTMULT_ACCUM (Multiply-Accumulate) on page 10-1 to include an obsolescence note and remove Arria V, Cyclone V, and Stratix V devices information. • Updated ALTMULT_ADD (Multiply-Adder) on page 11-1 to include an obsolescence note and remove Arria V, Cyclone V, and Stratix V devices information.
February 2013	3.1	• Updated Table 52 on page 63 to include Stratix V information for accum_sload port. • Updated Table 54 on page 65 to include Stratix V information for PORT_SIGNA and PORT_SIGNB parameters.
February 2012	3.0	• Added Arria V and Cyclone V device support. • Updated the parameter description for the following section: — ALTMULT_ACCUM (Multiply-Accumulate) — ALTMULT_ADD (Multiply-Add) • Added the Double Accumulator section.
July 2010	2.0	• Updated architecture information for the following sections: — ALTMULT_ACCUM (Multiply-Accumulate) — ALTMULT_ADD (Multiply-Add) — ALTMULT_COMPLEX (Complex Multiplier) • Added specification information for all megafunctions
November 2009	1.0	Initial release.