MAX 10 Embedded Multipliers User Guide

Subscribe

Send Feedback

UG-M10DSP
2017.02.21

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

ALTERA® now part of Intel

Contents

MAX® 10Embedded Multiplier Block Overview..... 1-1

MAX® 10Embedded Multipliers Features and Architecture..... 2-1
Embedded Multipliers Architecture..... 2-1
Input Register..... 2-1
Multiplier Stage..... 2-2
Output Register..... 2-2
Embedded Multipliers Operational Modes..... 2-3
18-Bit Multipliers..... 2-3
9-Bit Multipliers..... 2-4

MAX 10 Embedded Multipliers Implementation Guides..... 3-1
Files Generated by IP Cores..... 3-1
Verilog HDL Prototype Location..... 3-1
VHDL Component Declaration Location..... 3-2

LPM_MULT (Multiplier) IP Core References for MAX 10..... 4-1
LPM_MULT Parameter Settings..... 4-1
Signals..... 4-3

ALTMULT_ACCUM (Multiply-Accumulate) IP Core References for MAX 10..... 5-1
ALTMULT_ACCUM Parameter Settings..... 5-1
ALTMULT_ACCUM Ports..... 5-8

ALTMULT_ADD (Multiply-Adder) IP Core References for MAX 10..... 6-1
ALTMULT_ADD Parameter Settings..... 6-1
ALTMULT_ADD Ports..... 6-8

ALTMULT_COMPLEX (Complex Multiplier) IP Core References for MAX 10..... 7-1
ALTMULT_COMPLEX Parameter Settings..... 7-1
Signals..... 7-2

MAX 10 Embedded Multipliers User Guide Archives..... A-1

Additional Information for MAX 10 Embedded Multipliers User Guide.....B-1
Document Revision History for MAX 10 Embedded Multipliers User Guide B-1

MAX® 10Embedded Multiplier Block Overview

1

2017.02.21

UG-M10DSP Subscribe Send Feedback

The embedded multiplier is configured as either one 18 x 18 multiplier or two 9 x 9 multipliers. For multiplications greater than 18 x 18, the Quartus® Prime software cascades multiple embedded multiplier blocks together. There are no restrictions on the data width of the multiplier but the greater the data width, the slower the multiplication process.

Figure 1-1: Embedded Multipliers Arranged in Columns with Adjacent LABS

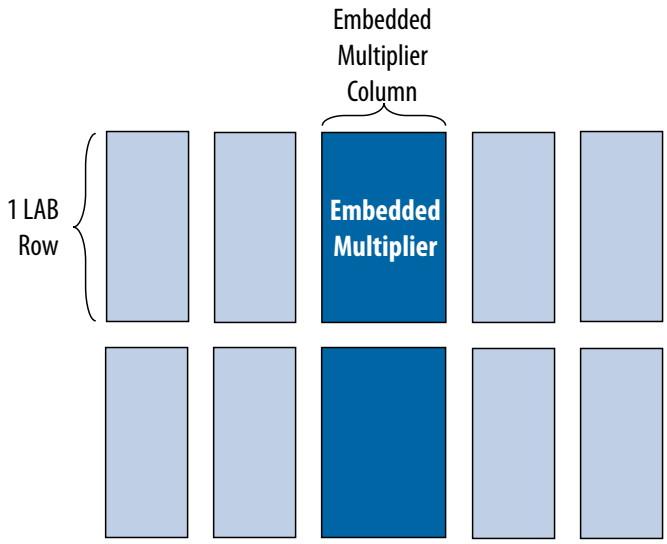


Table 1-1: Number of Embedded Multipliers in the MAX® 10 Devices

Device	Embedded Multipliers	9 x 9 Multipliers (1)	18 x 18 Multipliers (1)
10M02	16	32	16
10M04	20	40	20
10M08	24	48	24
10M16	45	90	45

(1) These columns show the number of 9 x 9 or 18 x 18 multipliers for each device. The total number of multipliers for each device is not the sum of all the multipliers.

Device	Embedded Multipliers	9 x 9 Multipliers [1]	18 x 18 Multipliers [1]
10M25	55	110	55
10M40	125	250	125
10M50	144	288	144

You can implement soft multipliers by using the M9K memory blocks as look-up tables (LUTs). The LUTs contain partial results from multiplying input data with coefficients implementing variable depth and width high-performance soft multipliers for low-cost, high-volume DSP applications. The availability of soft multipliers increases the number of available multipliers in the device.

Table 1-2: Number of Multipliers in the MAX® 10 Devices

Device	Embedded Multipliers	Soft Multipliers (16 x 16)[2]	Total Multipliers [3]
10M02	16	12	28
10M04	20	21	41
10M08	24	42	66
10M16	45	61	106
10M25	55	75	130
10M40	125	140	265
10M50	144	182	326

Related Information

MAX 10 Embedded Multipliers User Guide Archives on page 8-1
Provides a list of user guides for previous versions of the LPM_MULT, ALTMULT_ACCUM, ALTMULT_ADD, and ALTMULT_COMPLEX IP cores.

(1) These columns show the number of 9 x 9 or 18 x 18 multipliers for each device. The total number of multipliers for each device is not the sum of all the multipliers.
(2) Soft multipliers are implemented in sum of multiplication mode. M9K memory blocks are configured with 18-bit data widths to support 16-bit coefficients. The sum of the coefficients requires 18-bits of resolution to account for overflow.
(3) The total number of multipliers may vary, depending on the multiplier mode you use.

MAX® 10Embedded Multipliers Features and Architecture

2

2017.02.21

UG-M10DSP Subscribe Send Feedback

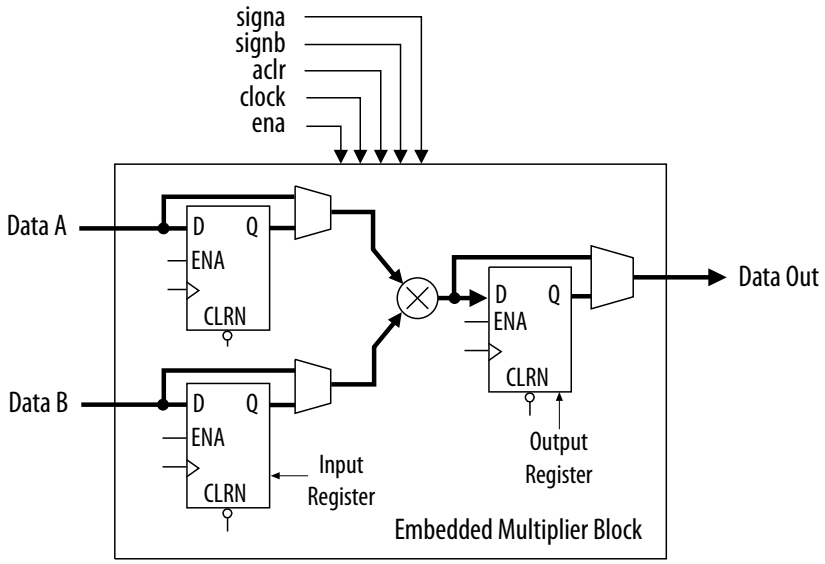
Each embedded multiplier consists of three elements. Depending on the application needs, you can use an embedded multiplier block in one of two operational modes.

Embedded Multipliers Architecture

Each embedded multiplier consists of the following elements:

- Multiplier stage
- Input and output registers
- Input and output interfaces

Figure 2-1: Multiplier Block Architecture



Input Register

Depending on the operational mode of the multiplier, you can send each multiplier input signal into either one of the following:

- An input register
- The multiplier in 9- or 18-bit sections

Each multiplier input signal can be sent through a register independently of other input signals. For example, you can send the multiplier `Data A` signal through a register and send the `Data B` signal directly to the multiplier.

The following control signals are available to each input register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

Multiplier Stage

The multiplier stage of an embedded multiplier block supports 9 × 9 or 18 × 18 multipliers and other multipliers in between these configurations. Depending on the data width or operational mode of the multiplier, a single embedded multiplier can perform one or two multiplications in parallel.

Each multiplier operand is a unique signed or unsigned number. Two signals, `signa` and `signb`, control an input of a multiplier and determine if the value is signed or unsigned. If the `signa` signal is high, the `Data A` operand is a signed number. If the `signa` signal is low, the `Data A` operand is an unsigned number.

The following table lists the sign of the multiplication results for the various operand sign representations. The results of the multiplication are signed if any one of the operands is a signed value.

Data A		Data B		Result
`signa` Value	Logic Level	`signb` Value	Logic Level	
Unsigned	Low	Unsigned	Low	Unsigned
Unsigned	Low	Signed	High	Signed
Signed	High	Unsigned	Low	Signed
Signed	High	Signed	High	Signed

You can dynamically change the `signa` and `signb` signals to modify the sign representation of the input operands at run time. You can send the `signa` and `signb` signals through a dedicated input register. The multiplier offers full precision, regardless of the sign representation.

When the `signa` and `signb` signals are unused, the Quartus Prime software sets the multiplier to perform unsigned multiplication by default.

Output Register

You can register the embedded multiplier output using output registers in either 18- or 36-bit sections. This depends on the operational mode of the multiplier. The following control signals are available for each output register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

Embedded Multipliers Operational Modes

You can use an embedded multiplier block in one of two operational modes, depending on the application needs:

- One 18-bit x 18-bit multiplier
- Up to two 9-bit x 9-bit independent multipliers

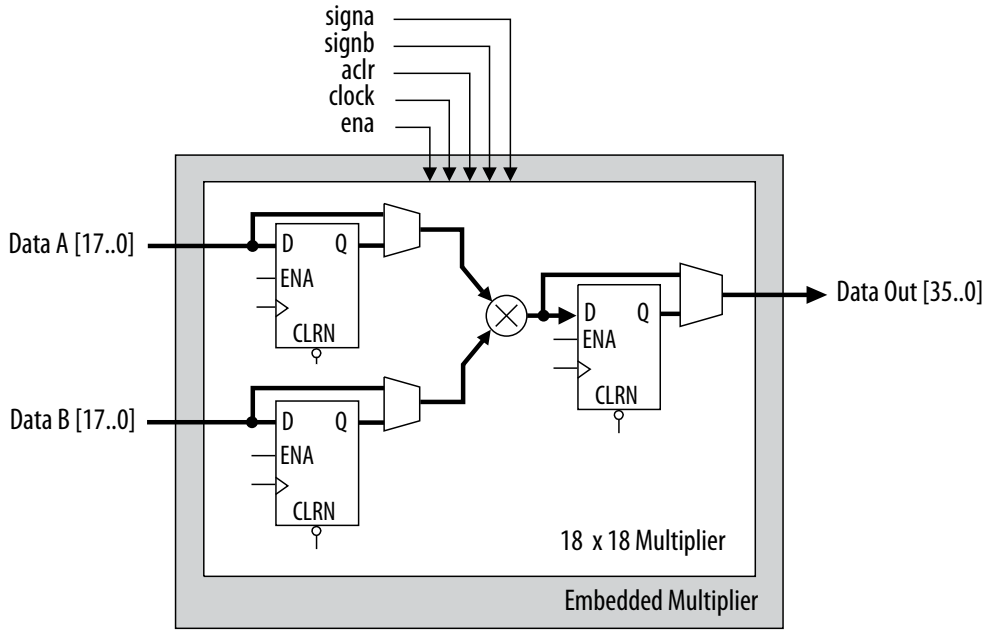
You can also use embedded multipliers of the MAX® 10 devices to implement multiplier adder and multiplier accumulator functions. The multiplier portion of the function is implemented using embedded multipliers. The adder or accumulator function is implemented in logic elements (LEs).

18-Bit Multipliers

You can configure each embedded multiplier to support a single 18 x 18 multiplier for input widths of 10 to 18 bits.

The following figure shows the embedded multiplier configured to support an 18-bit multiplier.

Figure 2-2: 18-Bit Multiplier Mode



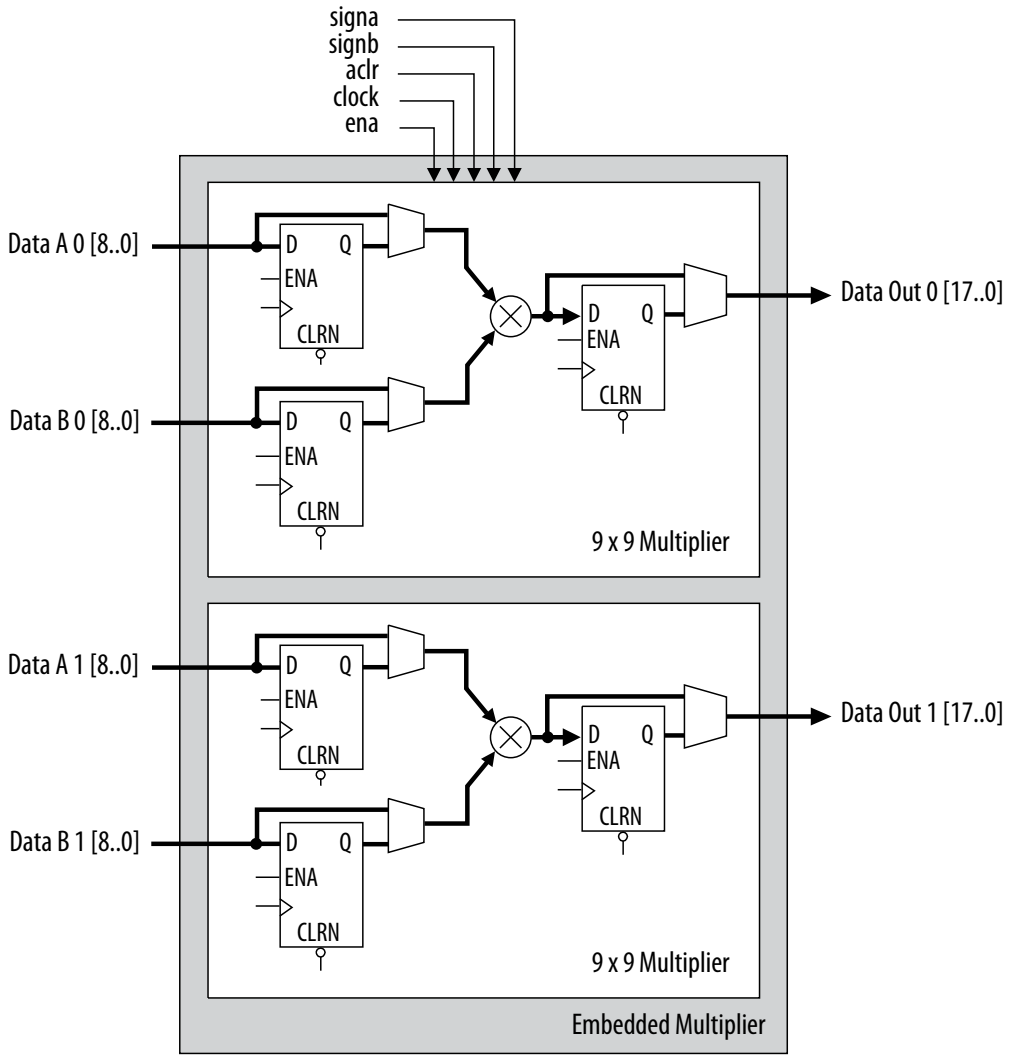
All 18-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both. Also, you can dynamically change the `signa` and `signb` signals and send these signals through dedicated input registers.

9-Bit Multipliers

You can configure each embedded multiplier to support two 9 × 9 independent multipliers for input widths of up to 9 bits.

The following figure shows the embedded multiplier configured to support two 9-bit multipliers.

Figure 2-3: 9-Bit Multiplier Mode



All 9-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both.

Each embedded multiplier block has only one `signa` and one `signb` signal to control the sign representation of the input data to the block. If the embedded multiplier block has two 9 × 9 multipliers the following applies:

- The `Data A` input of both multipliers share the same `signa` signal
- The `Data B` input of both multipliers share the same `signb` signal

MAX 10 Embedded Multipliers Implementation Guides 3

2017.02.21

UG-M10DSP

The Quartus Prime software contains tools for you to create and compile your design, and configure your device.

You can prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores using the Quartus Prime software.

Related Information

- **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.

Files Generated by IP Cores

The following integer arithmetic IP cores use the MAX 10 device embedded multipliers block:

- LPM_MULT
- ALTMULT_ACCUM (MAC)
- ALTMULT_ADD
- ALTMULT_COMPLEX

Verilog HDL Prototype Location

You can view the Verilog HDL prototype for the IP cores in the following Verilog Design Files (**.v**):

Table 3-1: Verilog HDL Prototype Location

Integer Arithmetic Megafunctions	Directory	Verilog Design File (.v)
LPM_MULT	**<Quartus Prime installation directory>\eda\synthesis**	**lpm.v**

Integer Arithmetic Megafunctions	Directory	Verilog Design File (.v)
• ALTMULT_ACCUM • ALTMULT_ADD • ALTMULT_COMPLEX	**<Quartus Prime installation directory>\eda\synthesis**	**altera_mf.v**

VHDL Component Declaration Location

You can view the VHDL component declaration for the IP cores in the following VHDL Design Files (**.vhd**):

Integer Arithmetic Megafunctions	Directory	VHDL Design File (.vhd)
LPM_MULT	**<Quartus Prime installation directory>\libraries\vhdl\lpm**	**LPM_PACK.vhd**
• ALTMULT_ACCUM • ALTMULT_ADD • ALTMULT_COMPLEX	**<Quartus Prime installation directory>\libraries\vhdl\altera_mf**	**altera_mf_components.vhd**

LPM_MULT (Multiplier) IP Core References for MAX 10

4

2017.02.21

UG-M10DSP Subscribe Send Feedback

LPM_MULT Parameter Settings

There are three groups of options: **General**, **General2**, and **Pipeling**.

Table 4-1: LPM_MULT Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Multiplier configuration	—	—	• Multiply 'dataa' input by 'datab' input • Multiply 'dataa' input by itself (squaring operation)	Specifies the multiplier configuration.
How wide should the 'dataa' input be?	LPM_WIDTHA	—	1–256	Specifies the width of the dataa[] port.
How wide should the 'datab' input be?	LPM_WIDTHB	—	1–256	Specifies the width of the datab[] port.
How should the width of the 'result' output be determined?	LPM_WIDTHP	—	• Automatically calculate the width • Restrict the width to [] bits	Specifies how the result width is determined.
How should the width of the 'result' output be determined? > Restrict the width to [] bits	LPM_WIDTHP	How should the width of the 'result' output be determined? > Restrict the width to [] bits = On	1–256	You can set the result width.

Table 4-2: LPM_MULT Parameters - General2

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Does the 'datab' input bus have a constant value?	—	—	• No • Yes, the value is []	You can specify the constant value of the 'datab' input bus, if any.
Which type of multiplication do you want?	LPM_REPRESENTATION	—	• Unsigned • Signed	Specifies the type of multiplication performed.
Which multiplier implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use default implementation • Use the dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier implementation.

Table 4-3: LPM_MULT Parameters - Pipeling

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Do you want to pipeline the function?	LPM_PIPELINE	—	• No • Yes, I want output latency of [] clock cycles	You can add extra latency to the outputs, if any.
Create an 'aclr' asynchronous clear port	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies asynchronous clear for the complex multiplier. Clears the function asynchronously when aclr port is asserted high.
Create a 'clken' clock enable clock	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies active high clock enable for the clock port of the complex multiplier
What type of optimization do you want?	MAXIMIZE_SPEED	—	• Default • Speed • Area	You can specify if the type of optimization is determined by Quartus Prime, speed, or area.

Signals

Table 4-4: LPM_MULT Input Signals

Signal Name	Required	Description
dataa[]	Yes	Data input. The size of the input signal depends on the LPM_WIDTHA parameter value.
datab[]	Yes	Data input. The size of the input signal depends on the LPM_WIDTHB parameter value.
clock	No	Clock input for pipelined usage. For LPM_PIPELINE values other than 0 (default), the clock signal must be enabled.
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 4-5: LPM_MULT Output signals

signal Name	Required	Description
result[]	Yes	Data output. For Stratix V, Arria V and Cyclone V, 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.

ALTMULT_ACCUM (Multiply-Accumulate) IP Core References for MAX 10

5

2017.02.21

UG-M10DSP

ALTMULT_ACCUM Parameter Settings

There are four groups of options: **General**, **Extra Modes**, **Multipliers**, and **Accumulator**.

Table 5-1: ALTMULT_ACCUM Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1	By default, only 1 multiplier is supported.
All multipliers have similar configurations	—	—	On	By default all multipliers have similar configurations
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
Create a 4th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned • Variable	Specifies the representation format for A inputs.

GUI Parameter	Parameter	Condition	Value	Description
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGN_REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_ PIPELINE_ REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_ PIPELINE_ ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned • Variable	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register

GUI Parameter	Parameter	Condition	Value	Description
Input Register > What is the source for clock input?	SIGN_REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_ PIPELINE_ REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_ PIPELINE_ ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

Table 5-2: ALTMULT_ACCUM Parameters - Extra Modes

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from A input of the last multiplier.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from B input of the last multiplier.
Add extra register(s) at the output	—	—	On	By default, output register must be enabled for accumulator.
What is the source for clock input?	OUTPUT_REG	Outputs Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the registers on the outputs.
What is the source for asynchronous clear input?	OUTPUT_ACLR	Outputs Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear signal for the registers on the outputs.
Add [] extra latency to the output	—	Outputs Configuration > More Options	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	Specifies the extra latency to add to the output.

GUI Parameter	Parameter	Condition	Value	Description
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use the default implementation • Use dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier-adder implementation.

Table 5-3: ALTMULT_ACCUM Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REG_A	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the clock port for the `dataa[]` port.
What is the source for asynchronous clear input?	INPUT_ACLR_A	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear port for the `dataa[]` port.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for clock input?	INPUT_REG_B	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the clock port for the `datab[]` port.
What is the source for asynchronous clear input?	INPUT_ACLR_B	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear port for the `datab[]` port.
What is the input A of the multiplier connected to?	—	—	Multiplier input	By default, input A of the multiplier is always connected to the multiplier's input.
What is the input B of the multiplier connected to?	—	—	Multiplier input	By default, input B of the multiplier is always connected to the multiplier's input.
Register output of the multiplier	—	—	On or Off	Turn on to enable register output of the multiplier.
What is the source for clock input?	MULTIPLIER_REG	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the register that immediately follows the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	MULTIPLIER_ACLR	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear signal of the register that follows the corresponding multiplier.

Table 5-4: ALTMULT_ACCUM Parameters - Accumulator

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create an 'accum_sload' input port	—	—	On or off	Dynamically specifies whether the accumulator value is constant. If the accum_sload port is high, then the multiplier output is loaded into the accumulator.
Register 'accum_sload' input	—	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	On or off	Turn on to enable register 'accum_sload' input.
Add an extra pipeline register	—	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	On or off	Turn on this option if you want to enable the extra pipeline register

GUI Parameter	Parameter	Condition	Value	Description
Input Register > What is the source for clock input?	ACCUM_SLOAD_REG	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	Clock0–Clock3	Specifies the clock signal for the accum_sload port.
Input Register > What is the source for asynchronous clear input?	ACCUM_SLOAD_ACLR	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear source for the first register on the accum_sload input.
Pipeline Register > What is the source for clock input?	ACCUM_SLOAD_PIPELINE_REG	• • Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ACCUM_SLOAD_PIPELINE_ACLR	• Accumulator > Create an 'accum_soad' input port = On • Accumulator > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Create an 'overflow' output port	—	—	On or Off	Overflow port for the accumulator
Add [] extra latency to the multiplier output	EXTRA_MULTIPLIER_LATENCY	—	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	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.

ALTMULT_ACCUM Ports

Table 5-5: ALTMULT_ACCUM IP Core 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.
ena3	No	Clock enable for the clock3 port.

Port Name	Required	Description
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 5-6: ALTMULT_ACCUM IP Core Output Ports

Port Name	Required	Description
overflow	No	Overflow port for the accumulator.
result[]	Yes	Accumulator output port. The size of the output port depends on the WIDTH_RESULT parameter value.
scanouta[]	No	Output of the first shift register. The size of the output port depends on the WIDTH_A parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanouta[] port to shiftouta port.
scanoutb[]	No	Output of the second shift register. The size of the input port depends on the WIDTH_B parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanoutb[] port to shiftoutb port.

ALTMULT_ADD (Multiply-Adder) IP Core References for MAX 10

6

2017.02.21

UG-M10DSP Subscribe Send Feedback

ALTMULT_ADD Parameter Settings

There are three groups of options: **General**, **Extra Modes**, and **Multipliers**.

Table 6-1: ALTMULT_ADD Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1, 2, 3, or 4	Specifies the number of multipliers. You can specify up to four multipliers.
All multipliers have similar configurations	—	—	On or Off	Turn on this option if you want all multipliers to have similar configurations.
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
Create a 4th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.

GUI Parameter	Parameter	Condition	Value	Description
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned • Variable	Specifies the representation format for A inputs.
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned • Variable	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input

GUI Parameter	Parameter	Condition	Value	Description
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

Table 6-2: ALTMULT_ADD Parameters - Extra Modes

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on to create a signal from A input.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on to create a signal from B input.
Register output of the adder unit	—	—	On or Off	Turn on to create a register output of the adder unit.
What is the source for clock input?	OUTPUT_REGISTER	• Outputs Configuration > Register output of the adder unit = On • Outputs Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the output register.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	OUTPUT_ACLR	• Outputs Configuration > Register output of the adder unit = On • Outputs Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the first pair of multipliers?	MUTIPLIER1_DIRECTION	General > What is the number of multipliers? = 2, 3, or 4	• Add • Subtract • Variable	Specifies whether the second multiplier adds or subtracts its value from the sum. Values are add and subtract. If Variable is selected the `addnsub1` port is used.
'addnsub1' input controls the operation (1 add/0 sub)	—	Adder Operation > What operation should be performed on outputs of the first pair of multipliers? = Variable	More Options	High 'addnsub1' input indicates add and low 'addnsub1' input indicates subtract.
Register 'addnsub1' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addnsub1' input
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDSUB_MULTIPLIER_ACLR[1]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.

GUI Parameter	Parameter	Condition	Value	Description
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ ACLR[1]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the second pair of multipliers?	MUTIPLIER3_ DIRECTION	General > What is the number of multipliers? = 4	—	Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are add and subtract. If Variable is selected, the `addnsub3` port is used.
'addnsub3' input controls the sign (1 add/0 sub) - More Options	—	—	—	High 'addnsub3' input indicates add and low 'addnsub3' input indicates subtract.
Register 'addnsub3' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addnsub3' input.
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register.
Input Register > What is the source for clock input?	ADDNSUB_ MULTIPLIER_ REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDSUB_ MULTIPLIER_ ACLR[3]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ ACLR[3]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

GUI Parameter	Parameter	Condition	Value	Description
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use the default implementation • Use dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier-adder implementation.

Table 6-3: ALTMULT_ADD Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REGISTER_A[0..3]	• Input Configuration > Register input A of the multiplier = On • • Input Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_A[0..3]	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for clock input?	INPUT_REGISTER_B[0..3]	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_B[0..3]	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the input A of the multiplier connected to?	INPUT_SOURCE_A[0..3]	—	• Multiplier input • Shiftin input	Specifies the input A of the multiplier is connected to either multiplier input or shiftin input.
What is the input B of the multiplier connected to?	INPUT_SOURCE_B[0..3]	—	• Multiplier input • Shiftin input	Specifies the input B of the multiplier is connected to either multiplier input or shiftin input.
Register output of the multiplier	—	—	On or Off	Turn on to enable the register for output of the multiplier.
What is the source for clock input?	MULTIPLIER_REGISTER[]	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	MULTIPLIER_ACLR[]	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

ALTMULT_ADD Ports

Table 6-4: ALTMULT_ADD IP Core 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 6-5: ALTMULT_ADD IP Core 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.
scanoutb[]	No	Output of scan chain B. Output port [WIDTH_B - 1..0] wide.

ALTMULT_COMPLEX (Complex Multiplier) IP Core References for MAX 10

7

2017.02.21

UG-M10DSP

ALTMULT_COMPLEX Parameter Settings

There are two groups of options: **General** and **Implementation Style/Pipelining**.

Table 7-1: ALTMULT_COMPLEX Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned	Specifies the representation format for A inputs.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned	Specifies the representation format for B inputs.

Table 7-2: ALTMULT_COMPLEX Parameters - Implementation Style/Pipelining

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Which implementation style should be used?	IMPLEMENTATION_STYLE	—	Automatically select a style for best trade-off for the current settings	By default automatic selection for MAX 10 device is selected. Quartus Prime software will determine the best implementation based on the selected device family and input width.
Output latency [] clock cycles	PIPELINE	—	0–14	Specifies the number of clock cycles for output latency.
Create an asynchronous Clear input	—	—	On or off	Specifies synchronous clear for the complex multiplier. Clears the function asynchronously when the `aclr` port is asserted high.
Create clock enable input	—	—	On or off	Specifies active high clock enable for the clock port of the complex multiplier.

Signals

Table 7-3: 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.
`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.
`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.

Signal	Required	Description
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 7-4: 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.

MAX 10 Embedded Multipliers User Guide Archives

A

2017.02.21

UG-M10DSP

Subscribe Send Feedback

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

IP Core Version	User Guide
15.1	MAX 10 Embedded Multipliers User Guide
14.1	MAX 10 Embedded Multipliers User Guide

Additional Information for MAX 10 Embedded Multipliers User Guide

B

2017.02.21

UG-M10DSP Subscribe Send Feedback

Document Revision History for MAX 10 Embedded Multipliers User Guide

Date	Version	Changes
February 2017	2017.02.021	• Rebranded as Intel.
May 2016	2016.05.02	• Updated MAX 10 to each chapter in the user guide. • Added MAX 10 Embedded Multipliers User Guide Archives chapter.
November 2015	2015.11.02	• Changed instances of *Quartus II* to *Quartus Prime*. • Removed topics on generating IP cores and added links to Introduction to Altera IP Cores, Creating Version-Independent IP and Qsys Simulation Scripts, and Project Management Best Practices.
September 2014	2014.09.22	Initial release.