Intel Agilex® 7 Variable Precision DSP Blocks User Guide

Updated for Intel® Quartus® Prime Design Suite: **23.3**

Online Version
Send Feedback

UG-20213

683037
2023.10.02

Contents

1. Intel Agilex® 7 Variable Precision DSP Blocks Overview

The variable precision digital signal processing (DSP) blocks in Intel Agilex® 7 devices can support fixed-point arithmetic, single-precision, and half-precision floating-point arithmetic operations. The Intel Agilex 7 DSP blocks provide high design flexibility and are optimized to support high-performance DSP applications.

1.1. Features

The Intel Agilex 7 fixed-point arithmetic features include:

- High-performance, power-optimized, and fully registered multiplication operations
- 9-bit, 18-bit, and 27-bit word lengths
- Two 18 x 19 multipliers or one 27 x 27 multiplier per DSP block
- Built-in addition, subtraction, and 64-bit double accumulation register to combine multiplication results
- Cascading 19-bit or 27-bit and cascading 18-bit when pre-adder is used to form the tap-delay line for filtering applications
- Cascading 64-bit output bus to propagate output results from one block to the next block without external logic support
- Hard pre-adder supported in 18-bit and 27-bit DSP operation modes for symmetric filters
- Internal coefficient register bank in both 18-bit and 27-bit modes for filter implementation
- 18-bit and 27-bit systolic finite impulse response (FIR) filters with distributed output adder
- Biased rounding support
- Dynamically enable and disable scanin and chainout features

The Intel Agilex 7 floating-point arithmetic is a completely hardened architecture. Features for floating-point arithmetic include :

- Single-precision (32-bit arithmetic) and half-precision (16-bit arithmetic) modes
- Operational mode for flushed, extended, and bfloat16 (Brain Floating Point) floating-point format
- Multiplication, addition, subtraction, multiply-add, and multiply-subtract
- Multiplication with accumulation capability and a dynamic accumulator reset control
- Multiplication with cascade summation and subtraction capability
- Complex multiplication
- Direct vector dot product

- Systolic vector dot product
- Sequential vector dot product
- Exception handling support using exception flags:-
  - 8-bit exception flags for 32-bit arithmetic
  - 16-bit exception flags for 16-bit arithmetic
- Subnormal values handling

Related Information

- DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
  The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.
- DSP Block Specifications, Intel Agilex 7 FPGAs and SoCs Device Data Sheet: F-Series and I-Series
  Provides more information about DSP block performance.

1.2. Supported Operational Modes in Intel Agilex 7 Devices

1.2.1. Fixed-point Arithmetic

Table 1. Supported Combinations of Operational Modes and Features

Variable-precision DSP Block Resource	Operation Mode	Supported Operation Instance	Pre-adder Support	Coefficient Support	Input Cascade Support	Chainin/ Chainout Support
1 variable precision DSP block	Fixed-point independent 18 x 19 multiplication	2 (1)	Yes	Yes	Yes (2)	No
	Fixed-point independent 27 x 27 multiplication	1	Yes	Yes	Yes (3)	Yes
	Fixed-point two 18 x 19 multiplier adder mode	1	Yes	Yes	Yes (2)	Yes
	Fixed-point 18 x 18 multiplier adder summed with 36-bit input	1	No	No	No	Yes

continued...

(1) The Intel® Quartus® Prime software determines the merging of two independent multiplication automatically when there are not enough DSP blocks on the device or within a Logic Lock (Standard) region.

(2) Each of the two inputs to a pre-adder has a maximum width of 18-bit. When the input cascade is used to feed one of the pre-adder inputs, the maximum width for the input cascade is 18-bit.

(3) When you enable the pre-adder feature, the input cascade support is not available.

Variable-precision DSP Block Resource	Operation Mode	Supported Operation Instance	Pre-adder Support	Coefficient Support	Input Cascade Support	Chainin/ Chainout Support
	Fixed-point 18 x 19 systolic mode	1	Yes	Yes	Yes(2)	Yes
	Fixed-point four 9 x 9 multiplier adder mode	1	No	No	No	Yes
2 Variable precision DSP blocks	Fixed-point complex 18x19 multiplication	1	No	No	Yes(2)	No

Table 2. Supported Combinations of Operational Modes and Dynamic Control Features

Variable-Precision DSP Block Resource	Operation Mode	Dynamic ACCUMULATE	Dynamic LOADCONST	Dynamic SUB	Dynamic NEGATE	Dynamic Scanin	Dynamic Chainout
1 variable precision DSP block	Fixed-point four 9 x 9 multiplier adder mode	Yes	Yes	No	No	No	Yes
	Fixed-point independent 18 x 19 multiplication	No	No	No	No	Yes	No
	Fixed-point independent 27 x 27 multiplication	Yes	Yes	No	Yes	No	Yes
	Fixed-point two 18 x 19 multiplier adder mode	Yes	Yes	Yes	Yes	Yes	Yes
	Fixed-point 18 x 18 multiplier adder summed with 36-bit input	Yes	Yes	Yes	Yes	No	Yes
	Fixed-point 18 x 19 systolic mode	Yes	Yes	Yes	Yes	Yes	Yes
2 variable precision DSP blocks	Fixed-point complex 18 x 19 multiplication	No	No	No	No	No	No

Related Information

- DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
  The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.
- Pre-adder for Fixed-point Arithmetic on page 18
- Internal Coefficient for Fixed-point Arithmetic on page 18
- Accumulator, Chainout Adder, and Preload Constant for Fixed-point Arithmetic on page 19
- Input Cascade for Fixed-point Arithmetic on page 67

- Intel Agilex 7 Variable Precision DSP Blocks Design Considerations on page 64
- Intel Agilex 7 FPGAs and SoCs Family Plan, Intel Agilex 7 FPGAs and SoCs Device Overview
  Provides more information about the variable precision DSP blocks resources in Intel Agilex 7 devices.

1.2.2. Floating-point Arithmetic

Table 3. Supported Combinations of Operational Modes and Features

Variable-Precision DSP Block Resource	Operation Mode	Supported Operation Instance	Chainin Support	Chainout Support
1 variable precision DSP block	FP32 multiplication mode	1	No	Yes
	FP32 addition or subtraction mode	1	No	Yes
	FP32 multiplication with addition or subtraction mode	1	Yes	Yes
	FP32 multiplication with accumulation mode	1	No	Yes
	FP32 vector one mode	1	Yes	Yes
	FP32 vector two mode	1	Yes	Yes
	Sum of two FP16 multiplication mode	1	No	Yes
	Sum of two FP16 multiplication with FP32 addition mode	1	Yes	Yes
	Sum of two FP16 multiplication with accumulation mode	1	No	Yes
	FP16 vector one mode	1	Yes	Yes
	FP16 vector two mode	1	Yes	Yes
	FP16 vector three	1	No	Yes
4 Variable precision DSP blocks	Floating-point complex multiplication	1	No	No

Table 4. Supported Combinations of Operational Modes and Dynamic Control Features

Variable-Precision DSP Block Resource	Operation Mode	Dynamic ACCUMULATE
1 variable precision DSP block	FP32 multiplication mode	No
	FP32 adder or subtract mode	No
	FP32 multiplier adder or subtract mode	No
	FP32 multiplier accumulate mode	Yes
	FP32 vector one mode	No
	FP32 vector two mode	No
	Sum of two FP16 multiplication mode	No
	Sum of two FP16 multiplication with FP32 addition mode	No

continued...

Variable-Precision DSP Block Resource	Operation Mode	Dynamic ACCUMULATE
	Sum of two FP16 multiplication with accumulation mode	Yes
	FP16 vector one mode	No
	FP16 vector two mode	No
	FP16 vector three	Yes
4 Variable precision DSP blocks	Floating-point complex multiplication	No

Related Information

- Intel Agilex 7 FPGAs and SoCs Family Plan, Intel Agilex 7 FPGAs and SoCs Device Overview
  Provides more information about the variable precision DSP blocks resources in Intel Agilex 7 devices.
- Operational Modes for Floating-point Arithmetic on page 42
- Chainout Adder on page 76

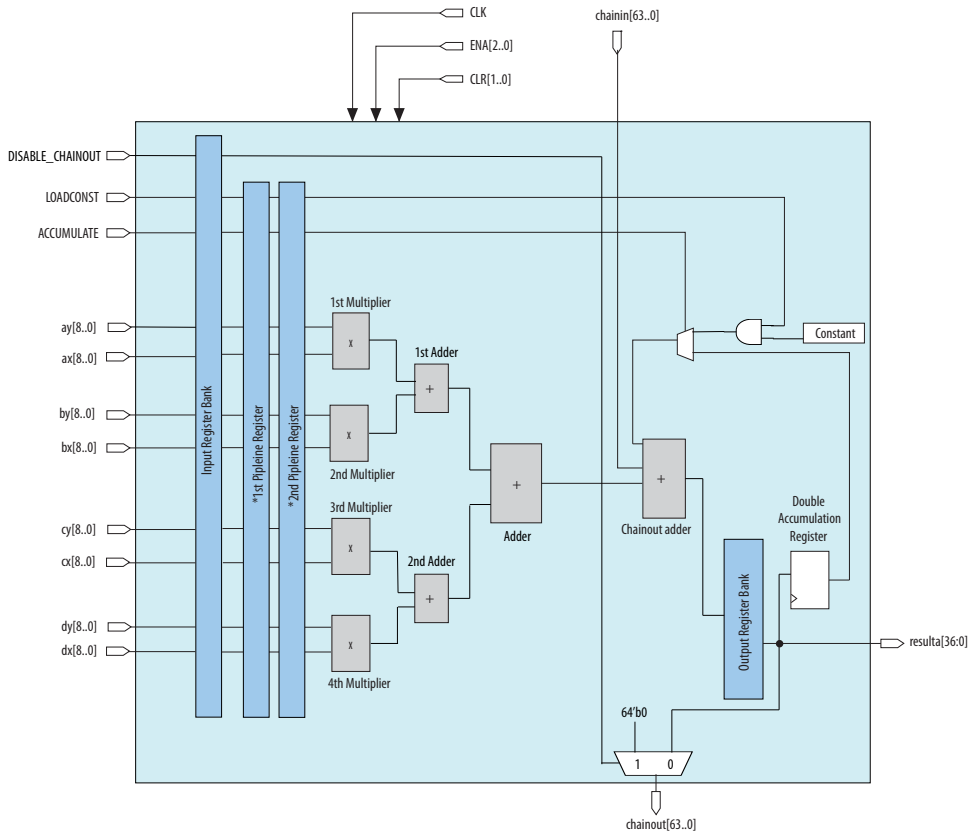
2. Intel Agilex 7 Variable Precision DSP Blocks Architecture

The Intel Agilex 7 variable precision DSP consists of the following blocks:

Table 5. Block Architecture

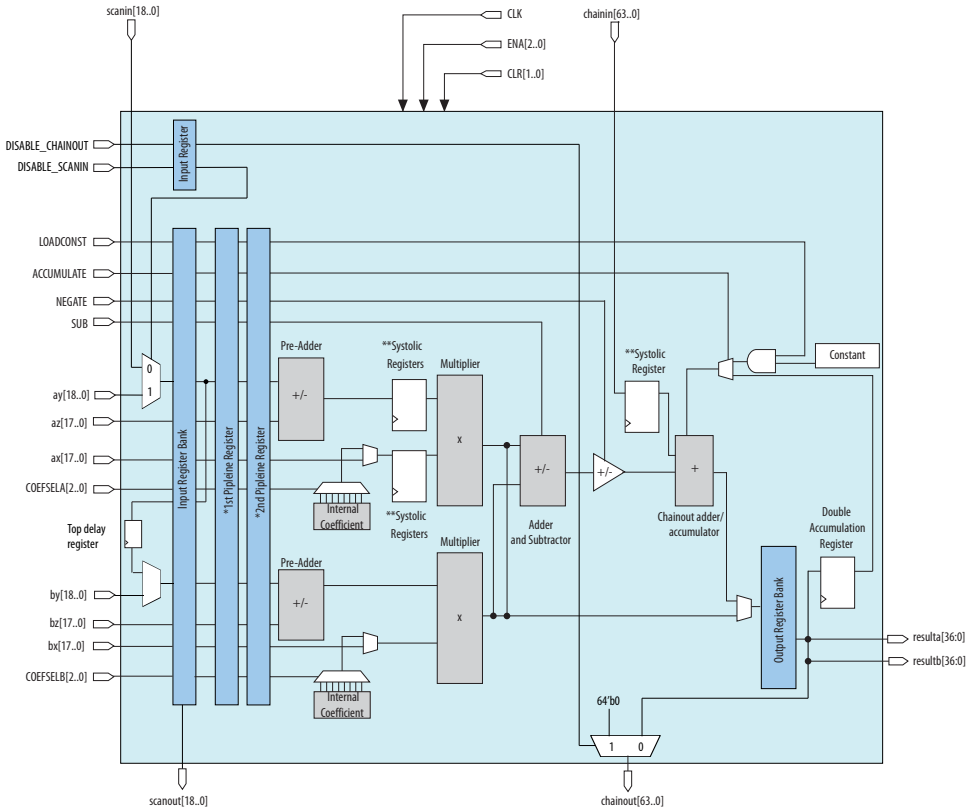
DSP Implementations	Block Architecture
Fixed-point Arithmetic	• Input register bank • First and second pipeline registers • Pre-adder/subtract • Internal coefficient • Multipliers • Adder and Subtractor • Accumulator, chainout adder, and Preload Constant • Systolic registers • Double accumulation register • Output register bank
Floating-point Arithmetic	• Input register bank • First and second pipeline registers • Multipliers • Adder • Accumulator • Output register bank • Exception Handling

Figure 1. Fixed-point Arithmetic 9 x 9 Mode



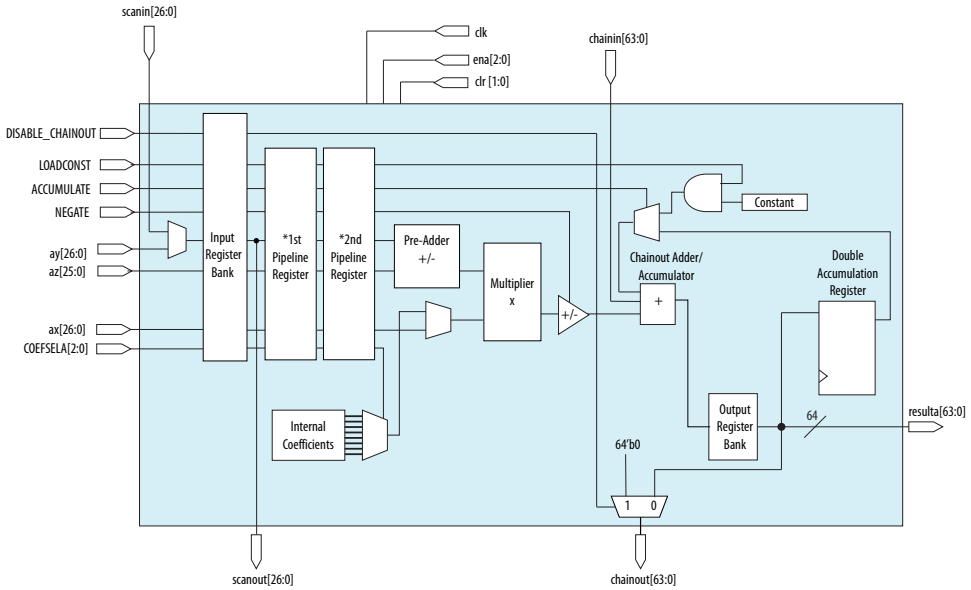
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

Figure 2. Fixed-point Arithmetic 18 x 19 Mode



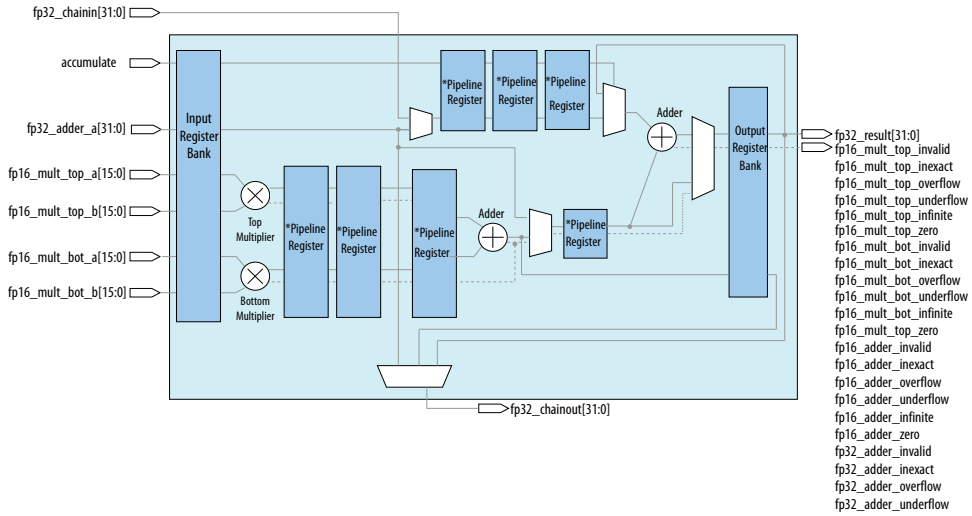
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.
**Systolic registers are enabled in systolic mode only.

Figure 3. Fixed-point Arithmetic 27 x 27 Mode



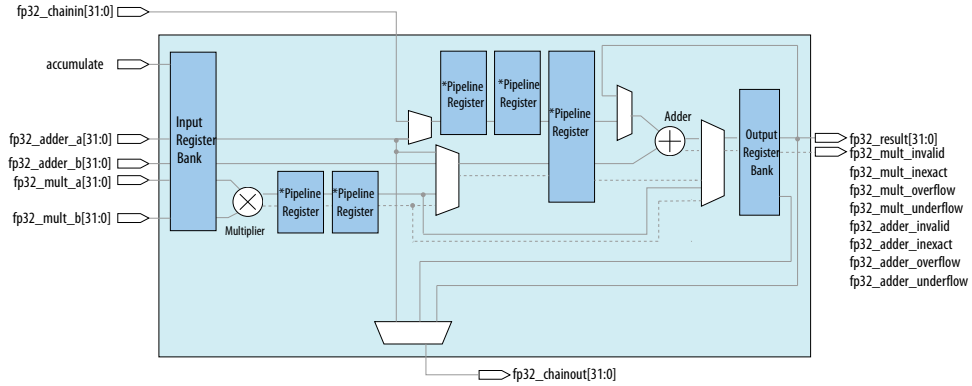
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

Figure 4. Floating-point Arithmetic 16-bit Half-Precision Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

Figure 5. Floating-point Arithmetic 32-bit Single-Precision Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

2.1. Fixed-point Arithmetic

2.1.1. Input Register Bank for Fixed-point Arithmetic

The input register banks for fixed-point DSP blocks are available for the following input signals:

- Data
- Dynamic control signals
  - NEGATE
  - LOADCONST
  - ACCUMULATE
  - SUB
  - Dynamic Scanin
  - Dynamic Chainout

All the registers in the DSP blocks are positive-edge triggered. These registers are not reset after power up and may hold unwanted data. Assert the CLR signal to clear the registers before starting an operation.

Each multiplier operand can feed an input register or a multiplier directly, bypassing the input registers.

The following variable precision DSP block signals control the input registers within the variable precision DSP block:

- CLK
- ENA[2..0]
- CLR[0]

Figure 6. Data Input Registers in Fixed-point Arithmetic 9 x 9 Mode

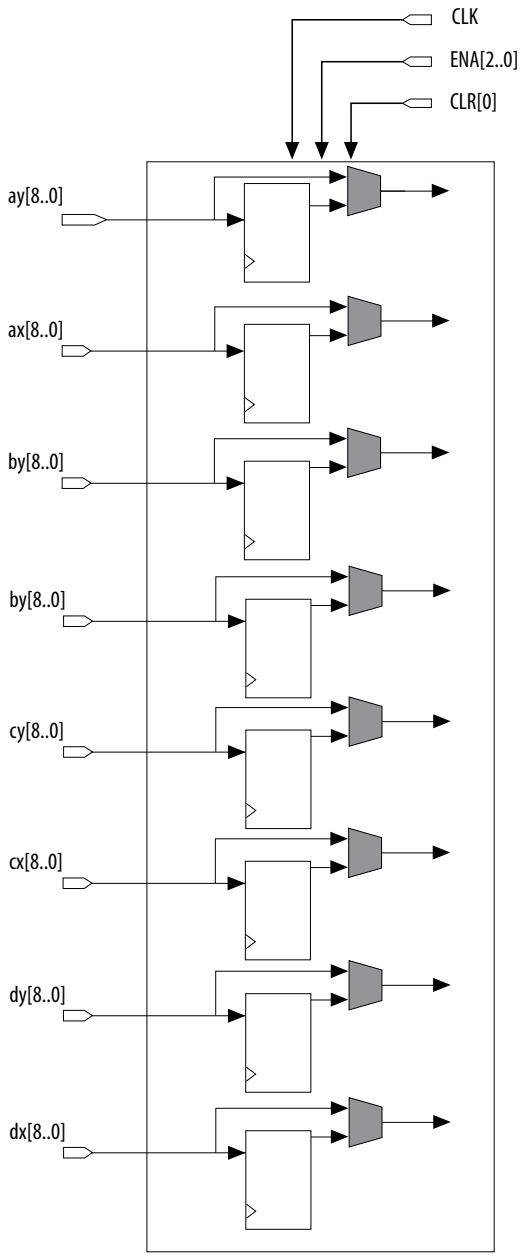


Figure 7. Data Input Registers in Fixed-point Arithmetic 18 x 19 Mode

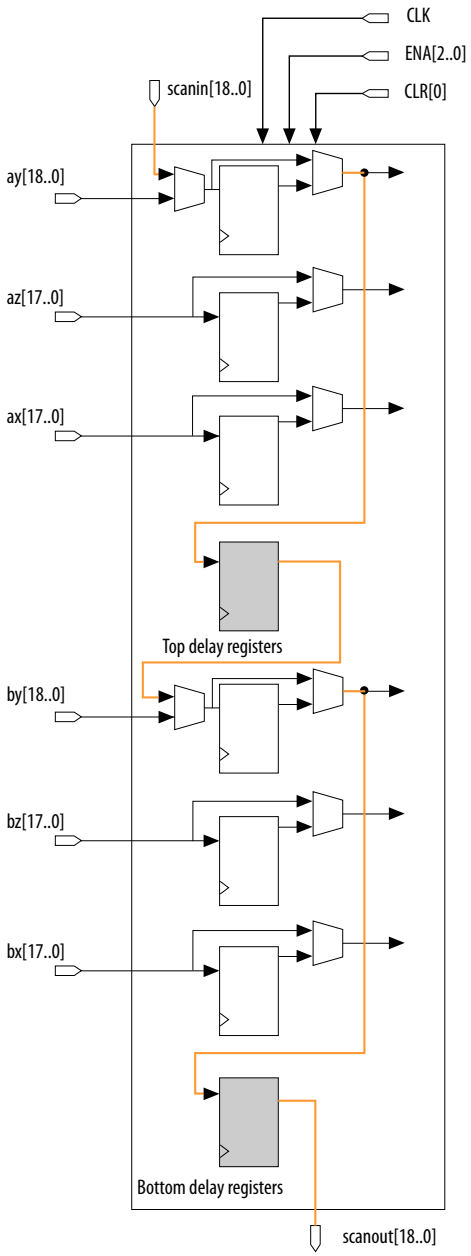
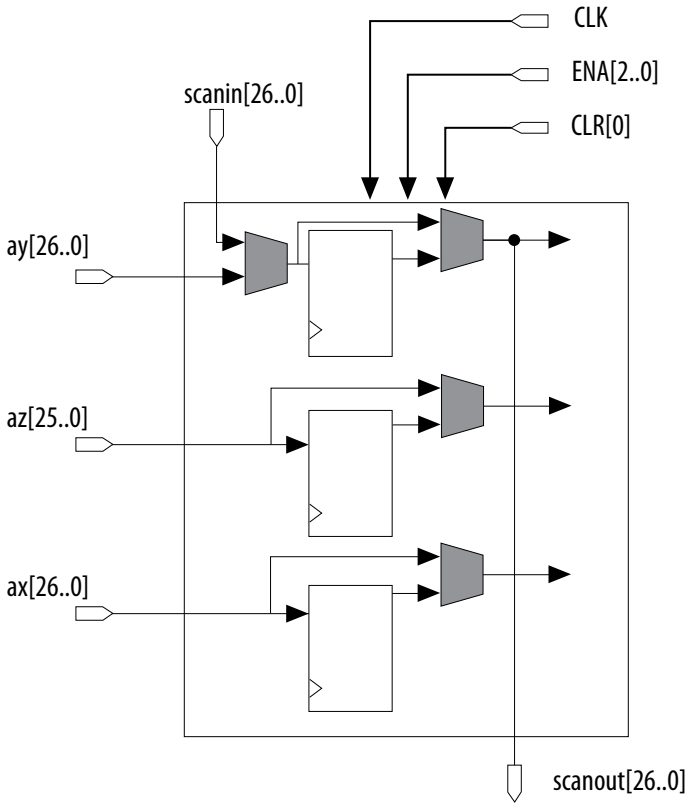


Figure 8. Data Input Registers in Fixed-point Arithmetic 27 x 27 Mode



Related Information

Configurations for Input, Pipeline, and Output Registers on page 64
Provides information about restrictions on fixed-point arithmetic input, pipeline, and output registers.

2.1.2. Pipeline Registers for Fixed-point Arithmetic

In addition to the input and output registers, there are 2 columns of pipeline registers for fixed-point arithmetic. Pipeline registers are used to get the maximum Fmax performance. The pipeline registers can be bypassed if high Fmax is not needed.

The following variable precision DSP block signals control the pipeline registers within the variable precision DSP block:

- CLK
- ENA[2..0]
- CLR[1]

Related Information

Configurations for Input, Pipeline, and Output Registers on page 64
Provides information about restrictions on fixed-point arithmetic input, pipeline, and output registers.

2.1.3. Pre-adder for Fixed-point Arithmetic

Each variable precision DSP block has two 19-bit pre-adders. You can configure these pre-adders in the following configurations:

- 18-bit (signed or unsigned) addition or 18-bit (signed) subtraction for 18 x 19 mode
- 26-bit addition or subtraction for 27 x 27 mode

For 18 x 19 mode, when both pre-adders within the same DSP block are used, they must share the same operation type (either addition or subtraction).

2.1.4. Internal Coefficient for Fixed-point Arithmetic

The Intel Agilex 7 variable precision DSP block has the flexibility of selecting the multiplicand from either the dynamic input or the internal coefficient.

The internal coefficient can support up to eight constant coefficients for the multiplicands in 18-bit and 27-bit modes. When you enable the internal coefficient feature, COEFSELA/COEFSELB are used to control the selection of the coefficient multiplexer.

2.1.5. Multipliers for Fixed-point Arithmetic

A single-variable precision DSP block can perform many multiplications in parallel, depending on the data width of the multiplier and implementation.

There are two multipliers per variable precision DSP block. You can configure these two multipliers in several operational modes:

- Four 9 (signed) x 9 (signed) multipliers or four 8 (unsigned) x 8 (unsigned) multipliers
- Two 18 (signed or unsigned) x 19 (signed) multipliers
- One 27 (signed) x 27 (signed) multiplier

2.1.6. Adder or Subtractor for Fixed-point Arithmetic

Depending on the operational mode, you can use the adder or subtractor as one 38-bit adder for fixed-point arithmetic addition or subtraction between two multipliers within a DSP block.

Use the dynamic SUB port to select the adder to perform addition or subtraction operation.

Table 6. Adder Operations with SUB Dynamic Control Signal

Operation	Description	SUB Signal
Addition	Adds the results of the two multipliers within one DSP block.	0
Subtraction	Subtracts the results between two multipliers within the same DSP block.	1

2.1.7. Accumulator, Chainout Adder, and Preload Constant for Fixed-point Arithmetic

The Intel Agilex 7 variable precision DSP block supports accumulator and adder up to 64 bits for fixed-point arithmetic.

The following signals can dynamically control the function of the accumulator and the chainout adder:

- NEGATE
- LOADCONST
- ACCUMULATE
- DISABLE_CHAINOUT

The accumulator and chainout adder features are not available in two fixed-point arithmetic independent 18 x 19 modes.

Table 7. Accumulator Functions and Dynamic Control Signals

Function	Description	NEGATE	LOADCONST	ACCUMULATE
Zeroing	Disables the accumulator.	0	0	0
Preload	The result is always added to the preload value. Only one bit of the 64-bit preload value can be "1". You can use this function to round the DSP result to any position of the 64-bit result.	0	1	0
Accumulation	Adds the current result to the previous accumulate result.	0	X	1
Decimation + Accumulation	This function takes the current result, converts it into two's complement, and adds it to the previous result.	1	X	1
Decimation + Chainout Adder	This function takes the current result, converts it into two's complement, and adds it to the output of previous DSP block.	1	0	0

2.1.7.1. Dynamic Chainout

Intel Agilex 7 devices support CHAINOUT port which can be dynamically disabled or enabled. In this feature, the input register is always enabled for the DISABLE_CHAINOUT signal.

Figure 9. Dynamic Chainout

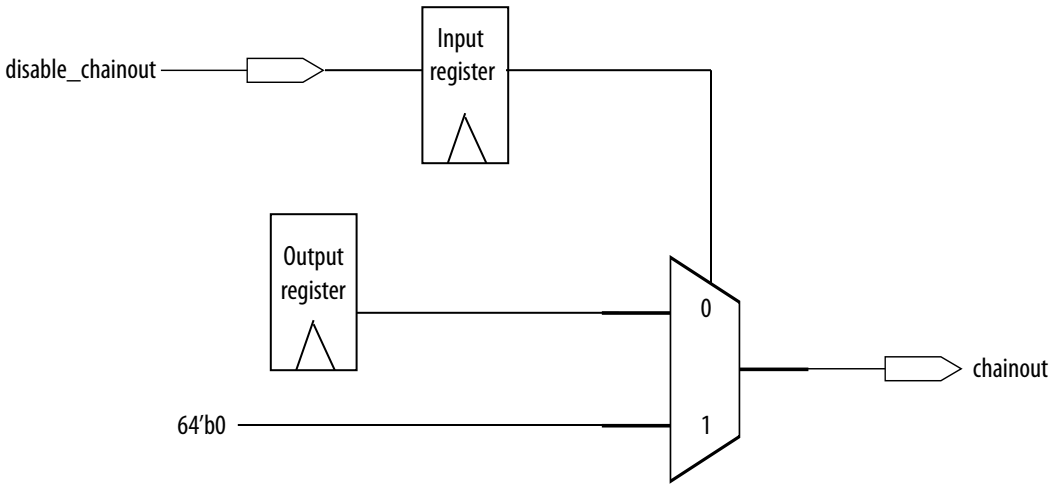


Table 8. DISABLE_CHAINOUT Signal Behavior

DISABLE_CHAINOUT Signal	Description
Low (0)	Chainout = result from output register
High (1)	Chainout = 0. Chainin to the next variable precision DSP block is disabled.

2.1.8. Systolic Register for Fixed-point Arithmetic

There are two sets of systolic registers per variable precision DSP block and each set supports up to 44 bits chain in and chain out adder. If the variable precision DSP block is not configured in fixed-point arithmetic systolic FIR mode, both sets of systolic registers are bypassed.

The first set of systolic registers consists of 18-bit and 19-bit registers that are used to register the 18-bit and 19-bit inputs of the upper multiplier, respectively.

The second set of systolic registers are used to delay the chainin input from the previous variable precision DSP block.

Below are the guidelines when implementing systolic registers in your design:

- The output register must be enabled when using systolic registers.
- First and second pipeline registers are optional when using systolic registers. If second pipeline is enabled, use the same clock enable as the input systolic register.
- The chainin systolic register always has the same clock enable as the output register.

2.1.9. Double Accumulation Register for Fixed-point Arithmetic

The accumulator supports double accumulation by enabling the 64-bit double accumulation registers located between the output register bank and the accumulator feedback path.

If the double accumulation register is enabled, an extra clock cycle delay is added into the feedback path of the accumulator.

This register has the same settings as the output register bank.

By enabling this register, you can have two accumulator channels using the same number of variable precision DSP block. This is useful when processing interleaved complex data (I, Q).

2.1.10. Output Register Bank for Fixed-point Arithmetic

The positive edge of the clock signal triggers the 74-bit bypassable output register bank. The output register bank is not reset after power up and may hold unwanted data. Assert the CLR signal to clear the register before starting an operation.

The following variable precision DSP block signals control the output register per variable precision DSP block:

- `CLK`
- `ENA[2..0]`
- `CLR[1]`

Related Information

Configurations for Input, Pipeline, and Output Registers on page 64
Provides information about restrictions on fixed-point arithmetic input, pipeline, and output registers.

2.2. Floating-point Arithmetic

2.2.1. Input Register Bank for Floating-point Arithmetic

The input register banks for floating-point DSP blocks are available for the following input signals:

- `fp32_adder_a`
- `fp32_adder_b`
- `fp32_mult_a`
- `fp32_mult_b`
- `fp16_mult_top_a`
- `fp16_mult_top_b`
- `fp16_mult_bot_a`
- `fp16_mult_bot_b`
- Dynamic ACCUMULATE control signal

Figure 10. Location of Input Register for FP32 Operation Modes

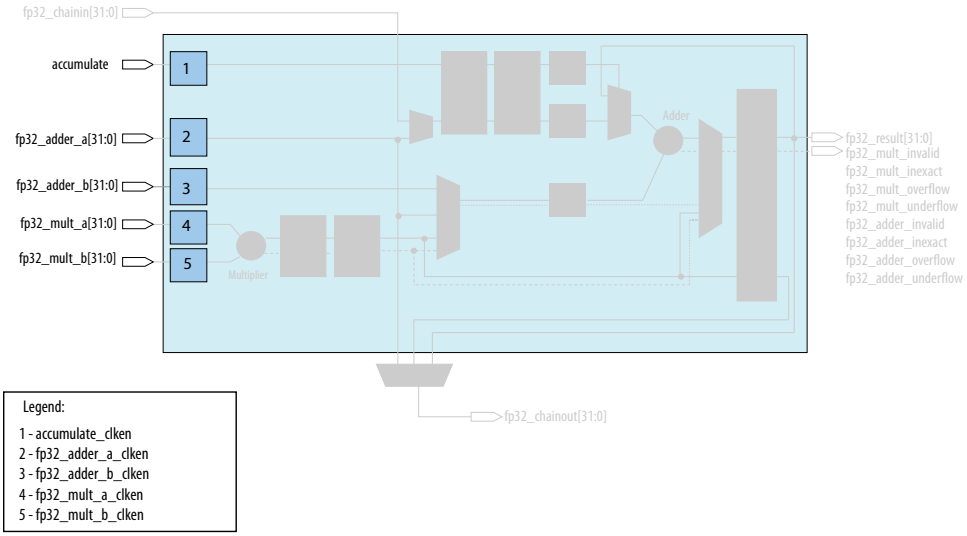
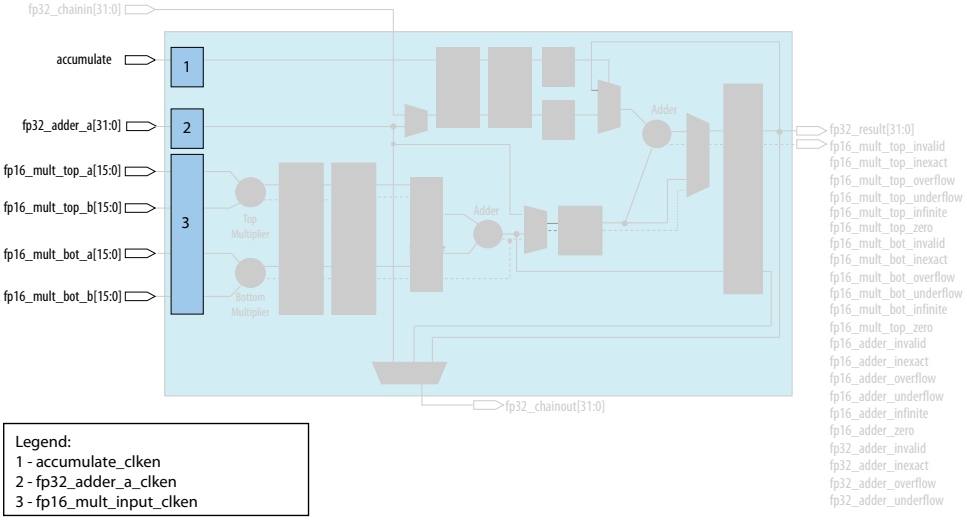


Figure 11. Location of Input Register for FP16 Operation Modes



All the registers in the DSP blocks are positive-edge triggered. These registers are not reset after power up and may hold unwanted data. Assert the CLR signal to clear the registers before starting an operation.

Each multiplier operand can feed an input register or a multiplier directly, bypassing the input registers.

The following variable precision DSP block signals control the input registers within the variable precision DSP block:

- CLK
- ENA[2..0]
- CLR[0]

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides information about restrictions on floating-point arithmetic input, pipeline, and output registers.

2.2.2. Pipeline Registers for Floating-point Arithmetic

Floating-point arithmetic has 3 latency layers of pipeline registers. You can bypass all latency layers of the pipeline registers or use any one, two or three layers of pipeline registers.

Figure 12. Location of Pipeline Register for FP32 Operation Modes

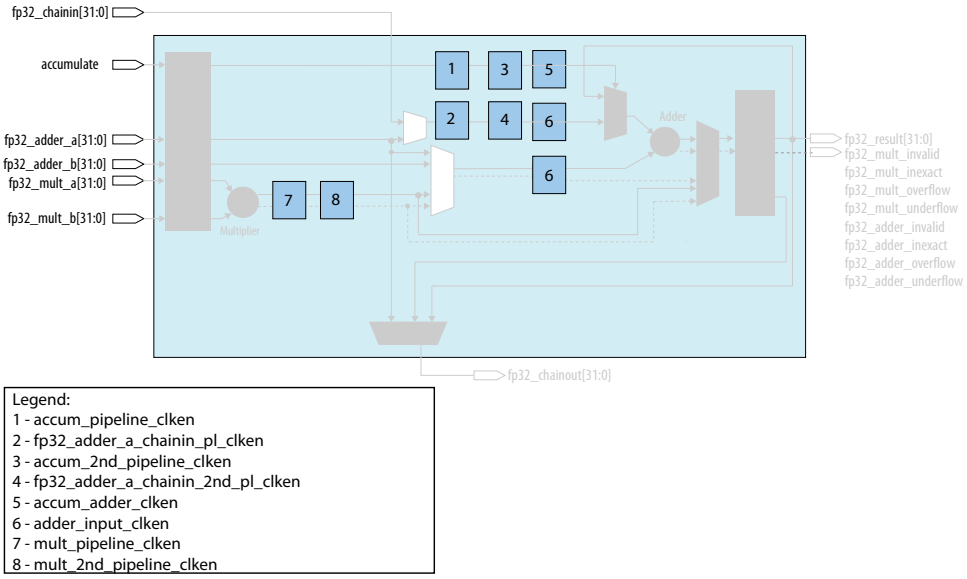
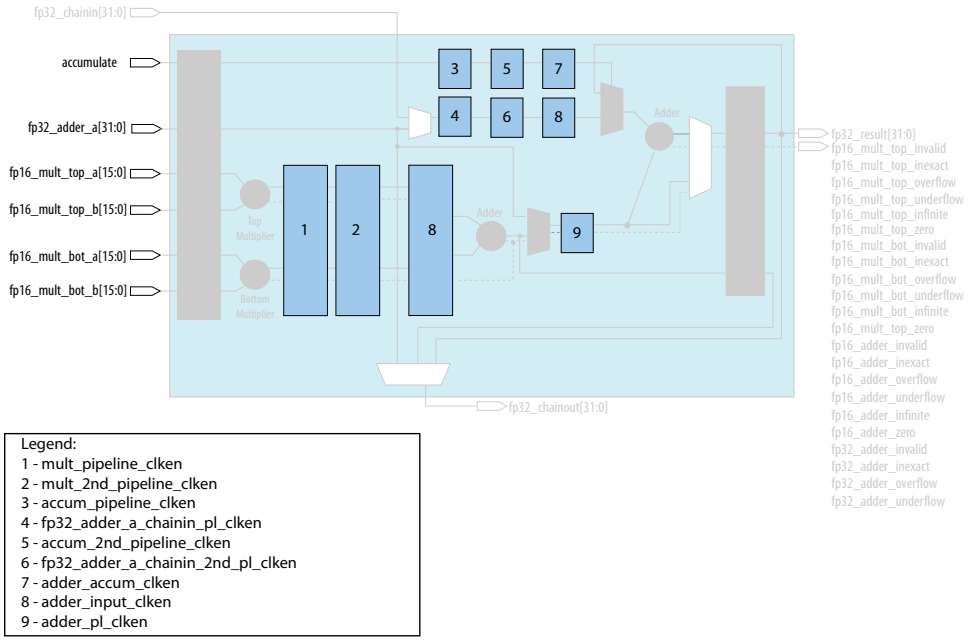


Figure 13. Location of Pipeline Register for FP16 Operation Modes



The following variable precision DSP block signals control the pipeline registers within the variable precision DSP block:

- CLK
- ENA[2..0]
- CLR[1]

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides information about restrictions on floating-point arithmetic input, pipeline, and output registers.

2.2.3. Multipliers for Floating-point Arithmetic

A single-variable precision DSP block can perform many multiplications in parallel, depending on the data width of the multiplier and implementation.

You can configure these two multipliers in several operational modes:

- One floating-point arithmetic single-precision multiplier
- Two floating-point arithmetic half-precision multiplier

2.2.4. Adder or Subtractor for Floating-point Arithmetic

Depending on the operational mode, you can use the adder or subtractor as

- A single precision addition/subtraction
- A single-precision multiplication with addition/subtraction
- Summation/subtraction of two half-precision multiplications with single precision result
- Summation/subtraction of two half-precision multiplications and addition/subtraction with single precision result
- Summation/subtraction of two half-precision multiplications accumulated into a single precision result

2.2.5. Output Register Bank for Floating-point Arithmetic

The positive edge of the clock signal triggers the 48-bit (32 bits data and 16 bits exception flags) bypassable output register bank. This register is not reset after power up and may hold unwanted data. Use the CLR signal to reset the register before starting an operation.

Figure 14. Location of Output Register for FP32 Operation Modes

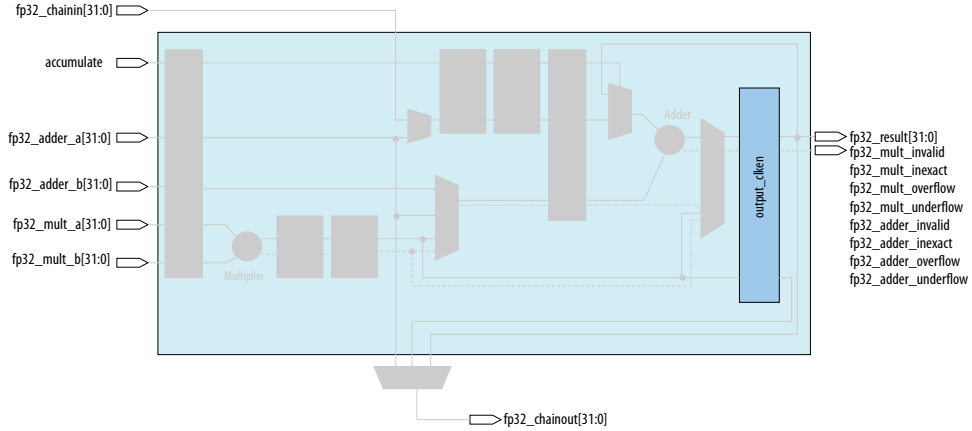
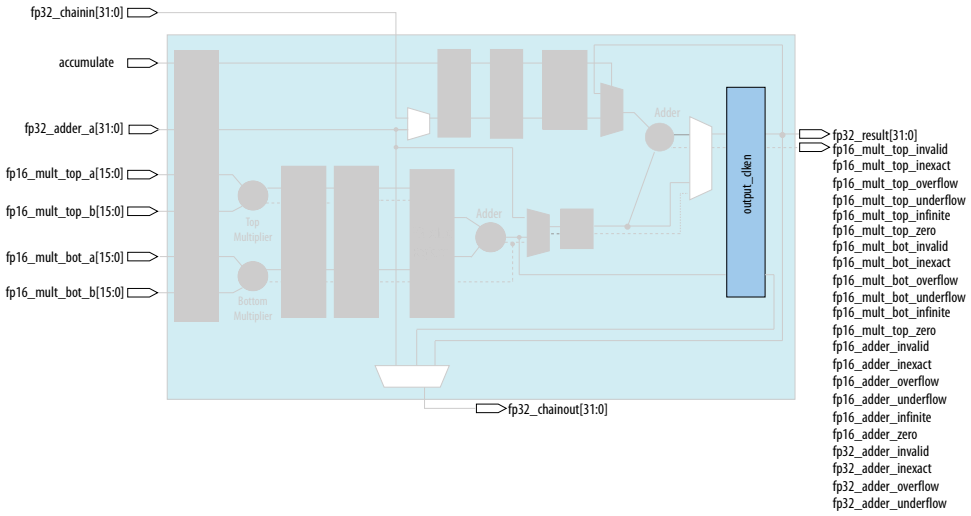


Figure 15. Location of Output Register for FP16 Operation Modes



The following variable precision DSP block signals control the output register per variable precision DSP block:

- `CLK`
- `ENA[2..0]`
- `CLR[1]`

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides information about restrictions on floating-point arithmetic input, pipeline, and output registers.

2.2.6. Exception Handling for Floating-point Arithmetic

The Intel Agilex 7 floating-point arithmetic supports exception handling for the multiplier and adder blocks.

Table 9. Supported Exception Flags

Floating-point Format	Exception Flags	Width	Description
Single precision	Multiplication		
	`fp32_mult_overflow`	1	This signal indicates if the multiplier result is a larger value than the maximum presentable value. 1: If the multiplier result is a larger value than the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value. This signal is not available in **Adder or Subtract Mode**.
	`fp32_mult_underflow`	1	This signal indicates if the multiplier result is a smaller value than the minimum presentable value.

continued...

Floating-point Format	Exception Flags	Width	Description
			1: If the multiplier result is a smaller value than the minimum representable non-zero absolute value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value. This signal is not available in **Adder or Subtract Mode**.
	`fp32_mult_inexact`	1	This signal indicates if the multiplier result is not accurately represented. 1: If the multiplier result is: • a rounded value • a smaller value than the minimum representable value or • a larger value than the maximum representable value. 0: If the multiplier result does not meet any of the criteria above. This signal is not available in **Adder or Subtract Mode**.
	`fp32_mult_invalid`	1	This signal indicates if the multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number. This signal is not available in **Adder or Subtract Mode**.
	Addition		
	`fp32_adder_overflow`	1	This signal indicates if the adder result is a larger value than the maximum representable value. 1: If the adder result is a larger value than the maximum presentable value and the result is cast to infinity. 0: If the adder result is not larger than the maximum presentable value. This signal is not available in **Multiplication Mode**.
	`fp32_adder_underflow`	1	This signal indicates if the adder result is a smaller value than the minimum presentable value. 1: If the adder result is a smaller value than the minimum representable non-zero absolute value and the result is flushed to zero. 0: If the adder result is a larger than the minimum representable value. This signal is not available in **Multiplication Mode**.
	`fp32_adder_inexact`	1	This signal indicates if the adder result is not accurately represented. 1: If the adder result is: • a rounded value • a smaller value than the minimum representable value or • a larger value than the maximum representable value. 0: If the adder result does not meet any of the criteria above. This signal is not available in **Multiplication Mode**.
	`fp32_adder_invalid`	1	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the adder result is invalid and cast to qNaN. 0: If the adder result is not an invalid number. This signal is not available in **Multiplication Mode**.
Half precision	Multiplication		

continued...

Floating-point Format	Exception Flags	Width	Description
	fp16_mult_top_overflow fp16_mult_bot_overflow	1	This signal indicates if the top or bottom multiplier result is a larger value than the maximum presentable value. 1: If the multiplier result is a larger value than the maximum representable value and the result is cast to infinity. 0: If the multiplier result is smaller than the maximum presentable value. This signal is not available in **Adder or Subtract Mode** and **Extended** format.
	fp16_mult_top_underflow fp16_mult_bot_underflow	1	This signal indicates if the top or bottom multiplier result is a smaller value than the minimum presentable value. 1: If the multiplier result is a smaller value than the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value. This signal is not available in **Adder or Subtract Mode** and **Extended** format.
	fp16_mult_top_inexact fp16_mult_bot_inexact	1	This signal indicates if the top or bottom multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value • a smaller value than the minimum representable value or • a larger value than the maximum representable value. 0: If the multiplier result does not meet any of the criteria above. This signal is not available in **Adder or Subtract Mode**.
	fp16_mult_top_invalid fp16_mult_bot_invalid	1	This signal indicates if the multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number. This signal is not available in **Adder or Subtract Mode**.
	fp16_mult_top_infinite fp16_mult_bot_infinite	1	This signal indicates if the top or bottom multiplier result is a positive or negative infinity. 1: If the result is infinite 0: If the result is normalized float or in the appropriate infinity range This signal is only available for **Extended** format.
	fp16_mult_top_zero fp16_mult_bot_zero	1	This signal indicates if the top or bottom multiplier result is a positive or negative zero. 1: If the result is zero 0: If the result is not a zero This signal is only available for **Extended** format.
	Addition		
	fp16_adder_overflow	1	This signal indicates if the adder result is a larger value than the maximum representable value. 1: If the adder result is a larger value than the maximum presentable value and the result is cast to infinity. 0: If the adder result is not larger than the maximum presentable value. This signal is not available in **Multiplication Mode Extended** format.

continued...

Floating-point Format	Exception Flags	Width	Description
	fp16_adder_underflow	1	This signal indicates if the adder result is a smaller value than the minimum presentable value. 1: If the adder result is a smaller value than the minimum representable value and the result is flushed to zero. 0: If the adder result is a larger than the minimum representable value. This signal is not available in **Multiplication Mode Extended** format.
	fp16_adder_inexact	1	This signal indicates if the adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value than the minimum representable value or • a larger value than the maximum representable value. 0: If the adder result does not meet any of the criteria above. This signal is not available in **Multiplication Mode**.
	fp16_adder_invalid	1	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the adder result is invalid and cast to qNaN. 0: If the adder result is not an invalid number. This signal is not available in **Multiplication Mode**.
	fp16_adder_infinite	1	This signal indicates if the adder result is a positive or negative infinity. 1: If the result is infinite 0: If the result is normalized float or in the appropriate infinity range This signal is only available for **Extended** format.
	fp16_adder_zero	1	This signal indicates if the adder result is a positive or negative zero. 1: If the result is zero 0: If the result is not a zero This signal is only available for **Extended** format.

Table 10. Multiplier Exception Handling Possible Results for FP32 Multiplication, FP16 Flushed, and FP16 Bfloat16 Modes

Input A	Input B	Result	(4) Flags Overflow/Underflow/ Inexact/Invalid
Normalized	Normalized	Normalized value	0/0/0/0
		Normalized (rounded) value	0/0/1/0
		Positive/negative infinity value	1/0/1/0
		Subnormal (denormal) value	0/1/1/0
0 or Subnormal (denormal)	Normalized	0 value	0/0/0/0

continued...

(4) Output exception flags. These flags do not change if exceptions are at input value.

Input A	Input B	Result	(4) Flags Overflow/Underflow/ Inexact/Invalid
Positive/negative infinity	Normalized	Positive/negative infinity value	0/0/0/0
Quiet Not A Number (qNaN)	Normalized	qNaN value	0/0/0/0
0 or Subnormal (denormal)	0 or Subnormal (denormal)	0 value	0/0/0/0
Positive/negative infinity	0 or Subnormal (denormal)	qNaN value	0/0/0/1
Quiet Not A Number (qNaN)	0 or Subnormal (denormal)	qNaN value	0/0/0/0
Positive/negative infinity	Positive/negative Infinity	Positive/negative infinity value	0/0/0/0
Quiet Not A Number (qNaN)	Positive/negative Infinity	qNaN value	0/0/0/0
Quiet Not A Number (qNaN)	Quiet Not A Number (qNaN)	qNaN value	0/0/0/0

Table 11. Adder Exception Handling Possible Results for FP32 Addition/Subtraction, FP16 Flushed, and FP16 Bfloat16 Modes

Input A	Input B	Result :	(4) Flags Overflow/Underflow/ Inexact/Invalid
Normalized	Normalized	Normalized value	0/0/0/0
		Normalized (rounded) value	0/0/1/0
		Positive/negative infinity value	1/0/1/0
		0 value Sign bit = 0	0/0/0/0
		Subnormal (denormal) value The sign is preserved	0/1/1/0
0 or Subnormal (denormal)	Normalized	Input b	0/0/0/0
Positive/negative infinity	Normalized	Positive/negative infinity value	0/0/0/0
Quiet Not A Number (qNaN)	Normalized	qNaN value	0/0/0/0
0 or Subnormal (denormal)	0 or Subnormal (denormal)	0 value For (-0 + (-0)) equation, sign bit = 1. For any other equation, sign bit = 0.	0/0/0/0
Positive/negative infinity	0 or Subnormal (denormal)	Positive/negative infinity value	0/0/0/0
Quiet Not A Number (qNaN)	0 or Subnormal (denormal)	qNaN value	0/0/0/0
Positive/negative infinity	Positive/negative infinity	qNaN value for invalid cases Positive/negative infinity value for valid cases	0/0/0/1 for invalid cases 0/0/0/0 for valid cases Valid cases are:

continued...

(4) Output exception flags. These flags do not change if exceptions are at input value.

Input A	Input B	Result :	(4) Flags Overflow/Underflow/ Inexact/Invalid
			• Positive infinity value + positive infinity value • Negative infinity value + negative infinity value • Negative infinity value - positive infinity value • Positive infinity value - negative infinity value
Quiet Not A Number (qNaN)	Positive/negative infinity	qNaN value	0/0/0/0
Quiet Not A Number (qNaN)	Quiet Not A Number (qNaN)	qNaN value	0/0/0/0

Table 12. Multiplication Exception Handling Possible Results for FP16 Extended Modes

Input A	Input B	Result:	(4) Flags Infinite/Zero/Inexact/ Invalid
Normalized/Subnormalized	Normalized/Subnormalized	Normalized/Subnormalized	0/0/x/0
0 value	Normalized/Subnormalized	0 value	0/1/0/0
Positive/negative infinity	Normalized/Subnormalized	Positive/negative infinity value	1/0/0/0
Quiet Not A Number (qNaN)	Normalized/Subnormalized	qNaN value	0/0/0/1 Mantissa = {100...00}
0 value	0 value	0 value	0/1/0/0
Positive/negative infinity	0 value	qNaN value	0/0/0/1 Mantissa = {100...00}
Quiet Not A Number (qNaN)	0 value	qNaN value	0/0/0/1 Mantissa = {100...00}
Positive/negative infinity	Positive/negative infinity	Positive/negative infinity value	1/0/0/0
Quiet Not A Number (qNaN)	Positive/negative infinity	qNaN value	0/0/0/1 Mantissa = {100...00}
Quiet Not A Number (qNaN)	Quiet Not A Number (qNaN)	qNaN value	0/0/0/1 Mantissa = {100...00}

Table 13. Addition Exception Handling Possible Results for FP16 Extended Modes

Input A	Input B	Result:	(4) Flags Infinite/Zero/Inexact/ Invalid
Normalized/Subnormalized	Normalized/Subnormalized	Normalized/Subnormalized	0/0/x/0
		0 value Sign bit = 0	0/0/0/0
0 value	Normalized/Subnormalized	Input b	0/0/0/0

continued...

Input A	Input B	Result:	(4) Flags Infinite/Zero/Inexact/ Invalid
Positive/negative infinity	Normalized/Subnormalized	Positive/negative infinity value	1/0/0/0
Quiet Not A Number (qNaN)	Normalized/Subnormalized	qNaN value	0/0/0/1 Mantissa = {100...00}
0 value	0 value	0 value For (-0 + (-0)) equation, sign bit = 1. For any other equation, sign bit = 0.	0/0/0/0
Positive/negative infinity	0 value	Positive/negative infinity value	1/0/0/0
Quiet Not A Number (qNaN)	0 value	qNaN value	0/0/0/1 Mantissa = {100...00}
Positive/negative infinity	Positive/negative infinity	qNaN value for invalid cases Positive/negative infinity value for valid cases	0/0/0/1 for invalid cases Mantissa = {100...00} 1/0/0/0 for valid cases Valid cases are: • Positive infinity value + positive infinity value • Negative infinity value + negative infinity value • Negative infinity value - positive infinity value • Positive infinity value - negative infinity value
Quiet Not A Number (qNaN)	Positive/negative infinity	qNaN value	0/0/0/1 Mantissa = {100...00}
Quiet Not A Number (qNaN)	Quiet Not A Number (qNaN)	qNaN value	0/0/0/1 Mantissa = {100...00}

3. Intel Agilex 7 Variable Precision DSP Blocks Operational Modes

This section describes how you can configure the Intel Agilex 7 variable precision DSP block to efficiently support the fixed-point arithmetic and floating-point arithmetic operational modes.

Table 14. Operational Modes

Fixed-point Arithmetic	Floating-point Arithmetic
• Independent multiplier mode • Multiplier adder sum mode • Independent complex multiplier • 18 × 18 multiplication summed with 36-Bit input mode • 18 × 18 systolic FIR mode	• FP32 single-precision multiplication mode • FP32 single-precision addition or subtraction mode • FP32 single-precision multiply-add or multiply-subtract mode • FP32 single-precision multiply accumulate mode • Sum of two FP16 multiplication mode • Sum of two FP16 multiplication with FP32 addition mode • Sum of two FP16 multiplication with accumulation mode • FP32 single-precision and FP16 half-precision vector one mode • FP32 single-precision and FP16 half-precision vector two mode • FP32 single-precision and FP16 half-precision direct vector dot product • FP32 single-precision and FP16 half-precision complex multiplication

3.1. Operational Modes for Fixed-point Arithmetic

3.1.1. Independent Multiplier Mode

In independent input and output multiplier mode, the variable precision DSP blocks perform individual multiplication operations for general purpose multipliers.

Table 15. Supported Independent Multiplier Modes

Configuration	Multipliers per Block
18 (unsigned) x 18 (unsigned)	2
18 (signed) x 19 (signed)	2
27 (signed or unsigned) x 27 (signed or unsigned)	1

3.1.1.1. 18 × 18 or 18 × 19 Independent Multiplier

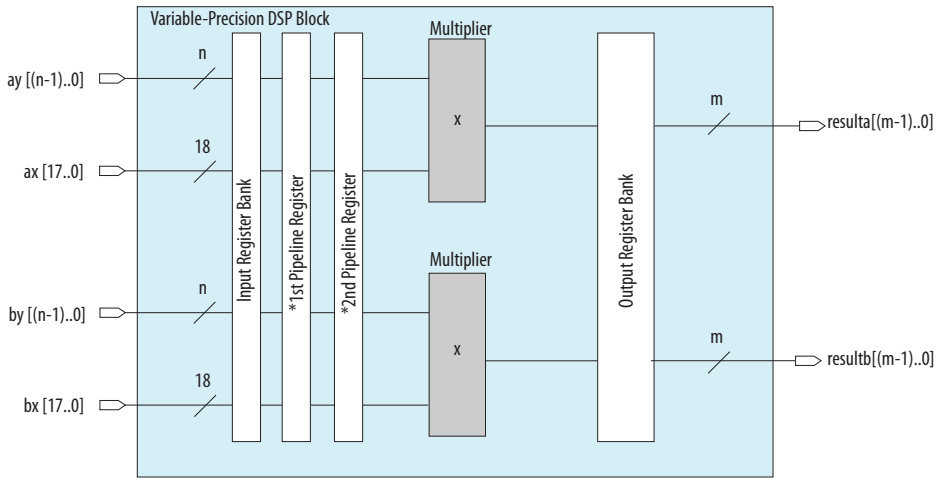
The 18 × 18 or 18 × 19 independent multiplier mode uses the following equations:

- resulta = ax × ay
- resultb = bx × by

Figure 16. Two 18 × 18 or 18 × 19 Independent Multiplier per Variable Precision DSP Block

In this figure, the variables are defined as follows:

- n = 19 and m = 37 for 18 × 19 signed operands
- n = 18 and m = 36 for 18 × 18 unsigned operands



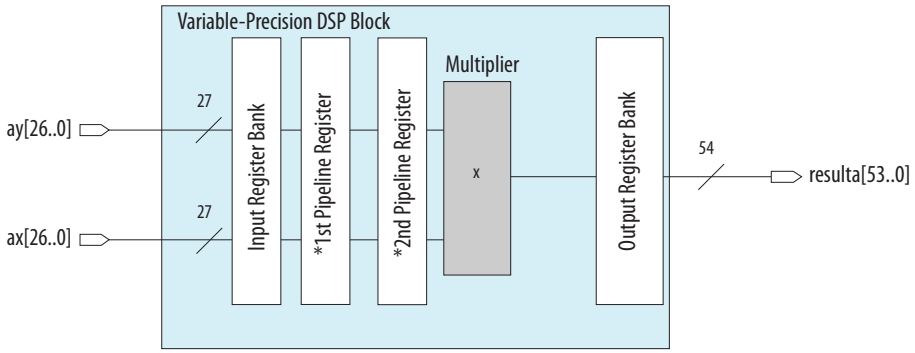
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.1.2. 27 × 27 Independent Multiplier

The 27 x 27 independent multiplier mode uses the equation of resulta = ay × ax.

Figure 17. One 27 × 27 Independent Multiplier Mode per Variable Precision DSP Block for Intel Agilex 7 Devices

In this mode, the `resulta` can be up to 64 bits when combined with a chainout adder or accumulator.



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.2. 8 x 8 (Unsigned) or 9 x 9 (Signed) Sum of 4 Mode

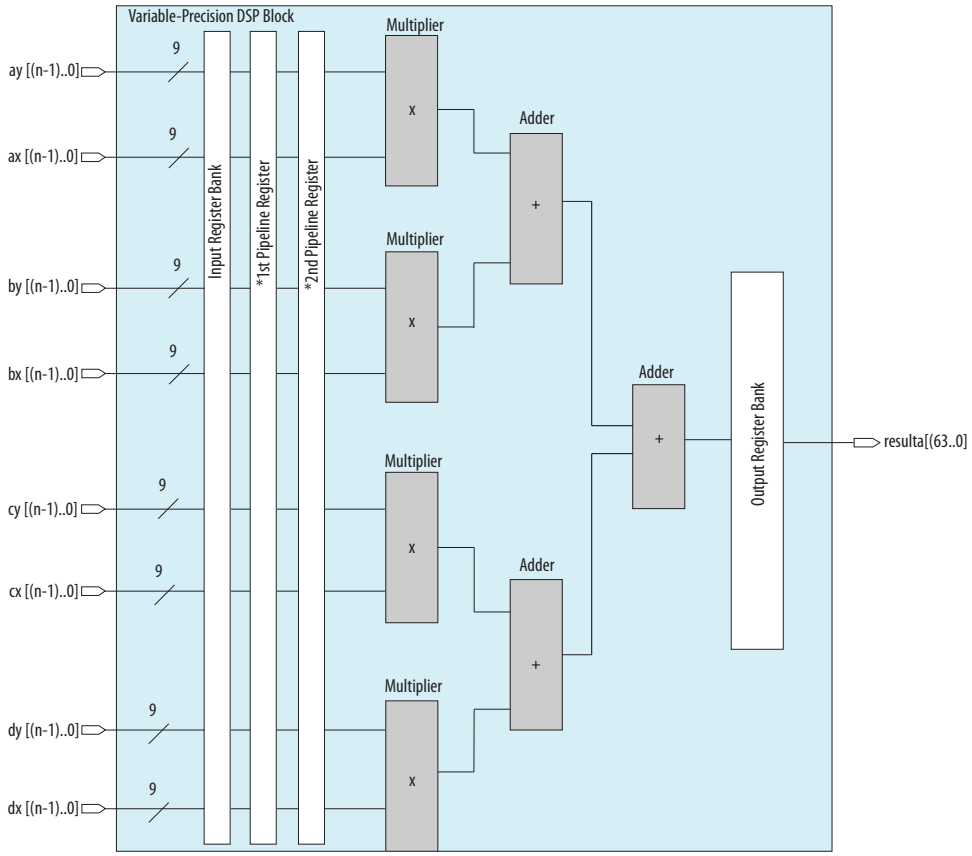
The 8 x 8 (unsigned) or 9 x 9 sum of 4 mode uses the following equation:

resulta = (ax × ay) + (bx × by) + (cx × cy) + (dx × dy)

Figure 18. 9 × 9 Sum of 4

In this figure, the variables are defined as follows:

- n = 8 and m = 8 for 8 x 8 unsigned operands
- n = 9 and m = 9 for 9 x 9 signed operands



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.3. Multiplier Adder Sum Mode

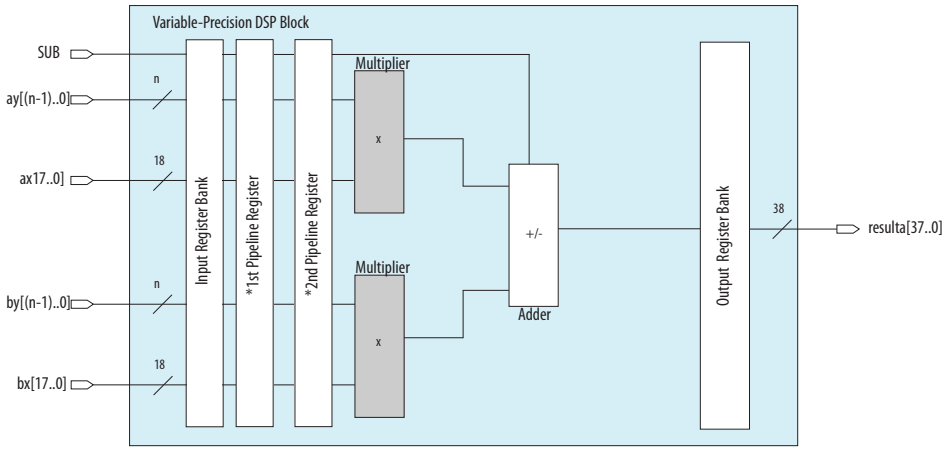
The multiplier adder sum mode uses these equations:

- resulta = (bx × by) + (ax × ay) to calculate the sum of the two 18 x 19 multiplications.
- resulta = (bx × by) – (ax × ay) to calculate the difference of the two 18 x 19 multiplications.

Figure 19. One Sum of Two 18 x 18 or 18 × 19 Multipliers with One Variable Precision DSP Block for Intel Agilex 7 Devices

In this figure, the variable is defined as follows:

- n = 19 for 18 × 19 signed operands
- n = 18 for 18 × 18 unsigned operands



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Set the SUB dynamic control signal to high to calculate the difference of the two 18 × 19 multiplications.

3.1.4. Independent Complex Multiplier

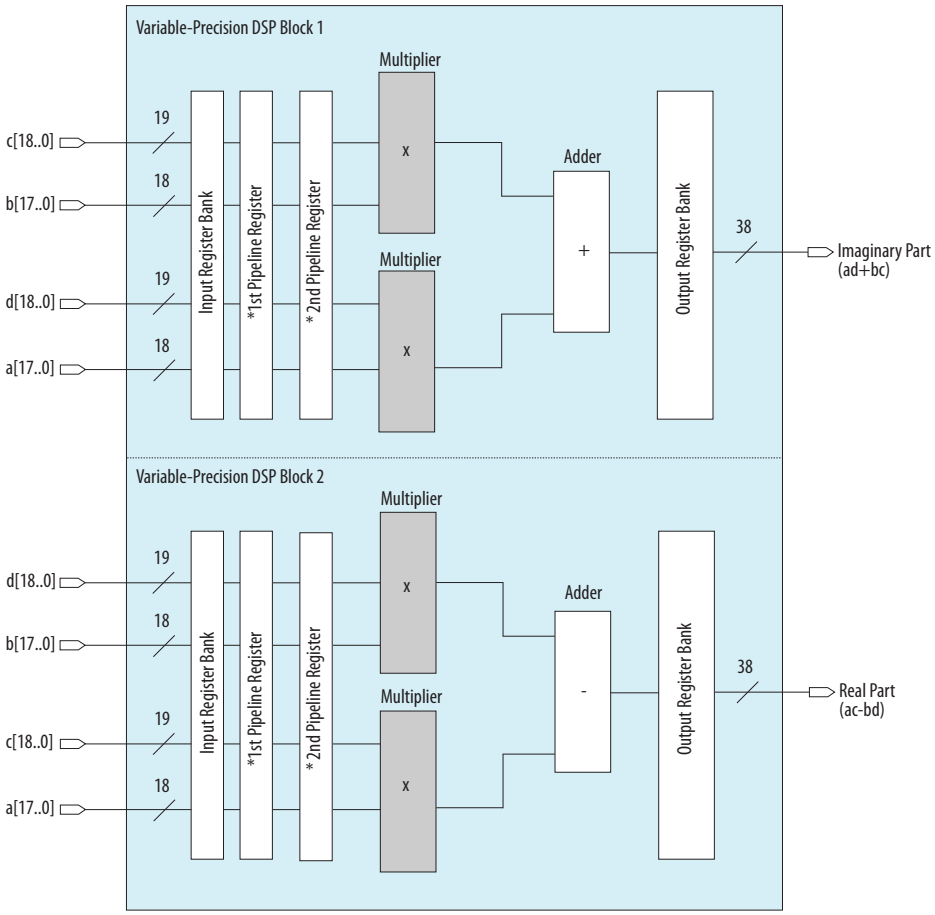
The Intel Agilex 7 devices support the 18 × 19 complex multiplier mode using two fixed-point arithmetic multiplier adder sum mode.

Figure 20. Sample of Complex Multiplication Equation

$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j[(a \times d) + (b \times c)]$$

The imaginary part [(a × d) + (b × c)] is implemented in the first variable-precision DSP block, while the real part [(a × c) - (b × d)] is implemented in the second variable-precision DSP block.

Figure 21. One 18 × 19 Complex Multiplier with Two Variable Precision DSP Blocks for Intel Agilex 7 Devices



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.4.1. 18 × 19 Multiplication Summed with 36-Bit Input Mode

Intel Agilex 7 variable precision DSP blocks support one 18 × 19 multiplication summed to a 36-bit input.

The 18 × 19 multiplication summed with 36-bit input mode uses the equations:

- resulta = (ax * ay) + bx to sum the 18 x 19 multiplication with 36-bit input.
- resulta = (ax * ay) - bx to subtract the 18 x 19 multiplication with 36-bit input.

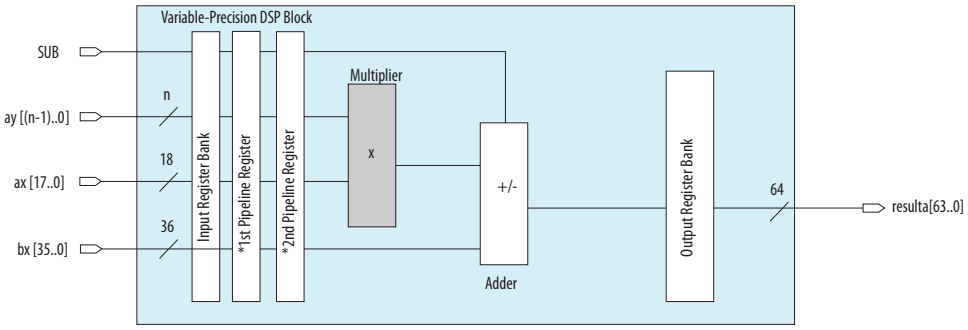
Use the upper multiplier to provide the input for an 18 × 19 multiplication, while the bottom multiplier is bypassed. The `bx[35..0]` signals the 36-bit input operand.

Use the SUB dynamic control signal to control the adder to perform addition or subtraction operation.

Figure 22. One 18 x 19 Multiplication Summed with 36-Bit Input Mode for Intel Agilex 7 Devices

In this figure, the variable is defined as follows:

- n = 19 for 18 × 19 signed operands
- n = 18 for 18 × 18 unsigned operands



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.5. Systolic FIR Mode

The basic structure of a FIR filter consists of a series of multiplications followed by an addition.

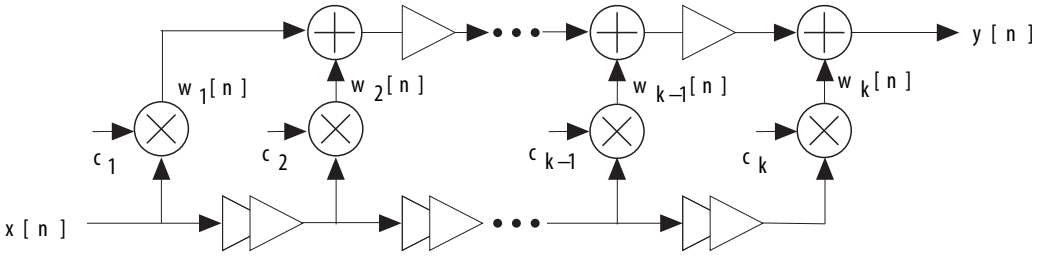
Figure 23. Basic FIR Filter Equation

$$y[n] = (\sum_{i=1}^{k} w_i[n-k+i] + w_1[n-k+2])$$

Where i start from 1, $w_i[n] = c_i x[n-2i+2]$

Depending on the number of taps and the input sizes, the delay through chaining a high number of adders can become quite large. To overcome the delay performance issue, the systolic form is used with additional delay elements placed per tap to increase the performance at the cost of increased latency.

Figure 24. Systolic FIR Filter Equivalent Circuit



Intel Agilex 7 variable precision DSP blocks support the following systolic FIR structures:

- 18-bit
- 27-bit

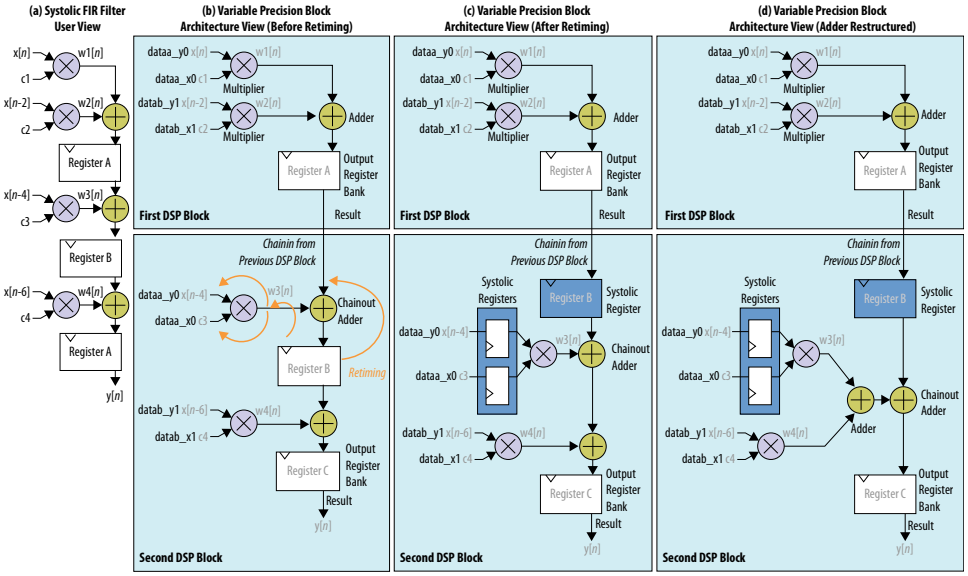
In systolic FIR mode, the input of the multiplier can come from four different sets of sources:

- Two dynamic inputs
- One dynamic input and one coefficient input
- One coefficient input and one pre-adder output
- One dynamic input and one pre-adder output

3.1.5.1. Mapping Systolic Mode User View to Variable Precision Block Architecture View

The following figure shows implementation of the systolic FIR filter (a) using the Intel Agilex 7 variable precision DSP blocks (d) by retiming the register and restructuring the adder. Register B can be retimed into systolic registers at the chainin, ay and ax input paths as shown in (b). The end result of the register retiming is shown in (c). The location of the adder is then restructured to sum both the multipliers output. The adder result is send to chainout adder to sum with the chainin value from the previous DSP block as shown in (d).

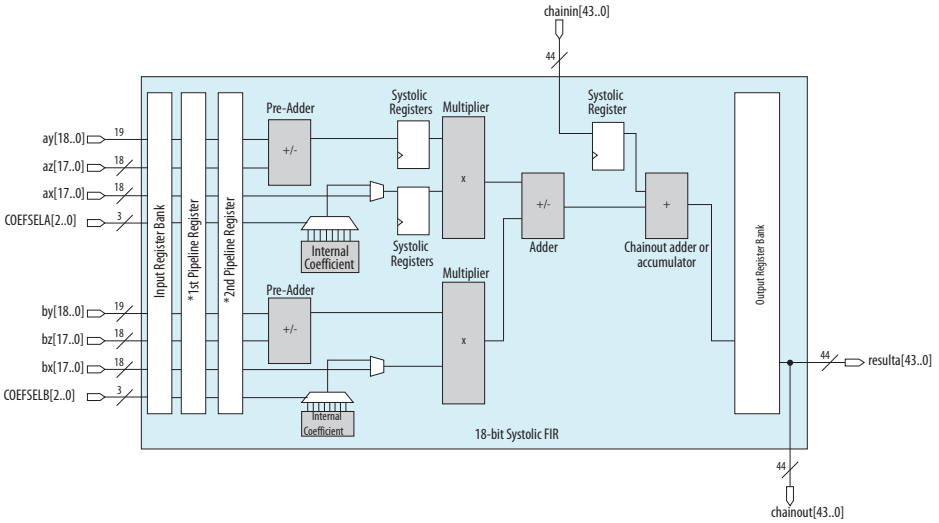
Figure 25. Mapping Systolic Mode User View to Variable Precision Block Architecture View



3.1.5.2. 18-bit Systolic FIR Mode

In 18-bit systolic FIR mode, the adders are configured as dual 44-bit adders, thereby giving 7 bits of overhead when using an 18 x 19 operation mode, resulting 37-bit result.

Figure 26. 18-Bit Systolic FIR Mode for Intel Agilex 7 Devices



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Related Information

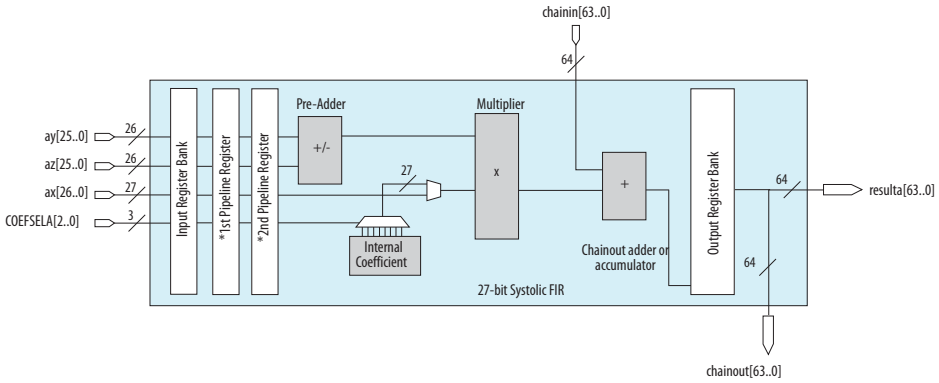
DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.

3.1.5.3. 27-Bit Systolic FIR Mode

In 27-bit systolic FIR mode, the chainout adder or accumulator is configured for a 64-bit operation, providing 10 bits of overhead when using a 27-bit data (54-bit products).

The 27-bit systolic FIR mode allows the implementation of one stage systolic filter per DSP block. Systolic registers are not required in this mode.

Figure 27. 27-Bit Systolic FIR Mode for Intel Agilex 7 Devices



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Related Information

DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.

3.2. Operational Modes for Floating-point Arithmetic

3.2.1. FP32 Single-precision Floating-point Arithmetic Functions

The FP32 single-precision floating-point arithmetic DSP can perform the following:

- FP32 multiplication
- FP32 addition or subtraction
- FP32 multiplication with addition or subtraction
- FP32 multiplication with accumulation
- FP32 vector one
- FP32 vector two

3.2.1.1. FP32 Multiplication Mode

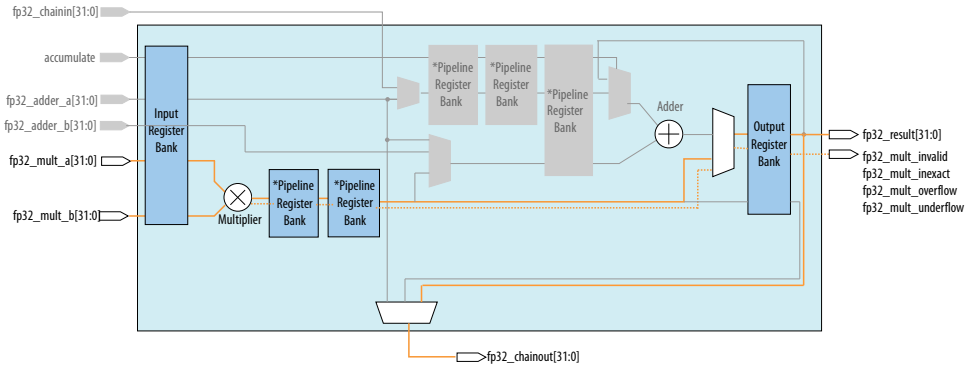
This mode allows you to apply basic floating-point multiplication equation:

fp32_result = fp32_mult_a*fp32_mult_b

The floating-point multiplication mode supports the following exception flags:

- `fp32_mult_invalid`
- `fp32_mult_inexact`
- `fp32_mult_overflow`
- `fp32_mult_underflow`

Figure 28. FP32 Multiplication Mode



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.1.2. Adder or Subtract Mode

This mode allows you to apply following equations:

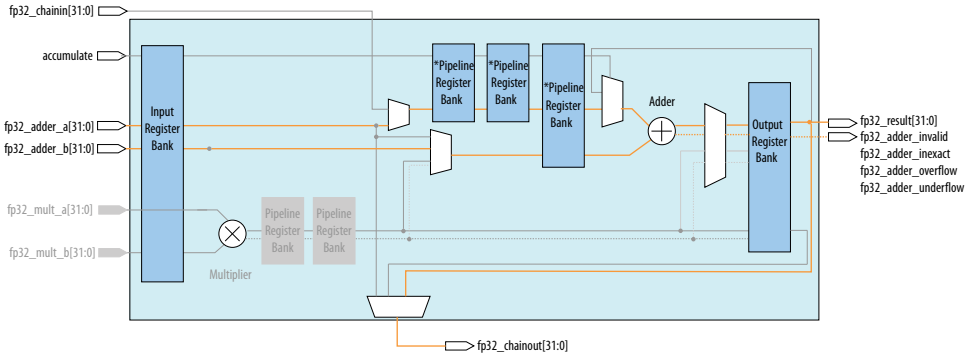
fp32_result = fp32_adder_b + fp32_adder_a

fp32_result = fp32_adder_b - fp32_adder_a

The floating-point adder or subtract mode supports the following exception flags:

- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

Figure 29. Adder or Subtract Mode for Intel Agilex 7



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.1.3. Multiply Accumulate Mode

This mode performs floating-point multiplication followed by floating-point addition or subtraction with the previous multiplication result.

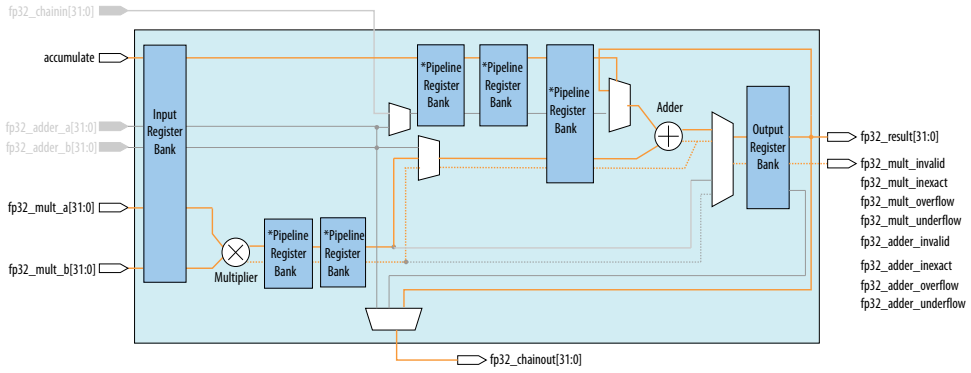
When ACCUMULATE signal is high, this mode uses the equation of fp32_result(t) = [fp32_mult_a(t)*fp32_mult_b(t)] +/- fp32_result(t-1).

When ACCUMULATE signal is low, this mode uses the equation of fp32_result = fp32_mult_a*fp32_mult_b.

The floating-point multiply accumulate mode supports the following exception flags:

- `fp32_mult_invalid`
- `fp32_mult_inexact`
- `fp32_mult_overflow`
- `fp32_mult_underflow`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

Figure 30. Multiply Accumulate Mode for Intel Agilex 7 Devices



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.1.4. FP32 Vector One Mode

This mode performs floating-point multiplication followed by floating-point addition or subtraction with the chainin input from the previous variable DSP Block. Input `fp32_adder_a` is directly fed into chainout.

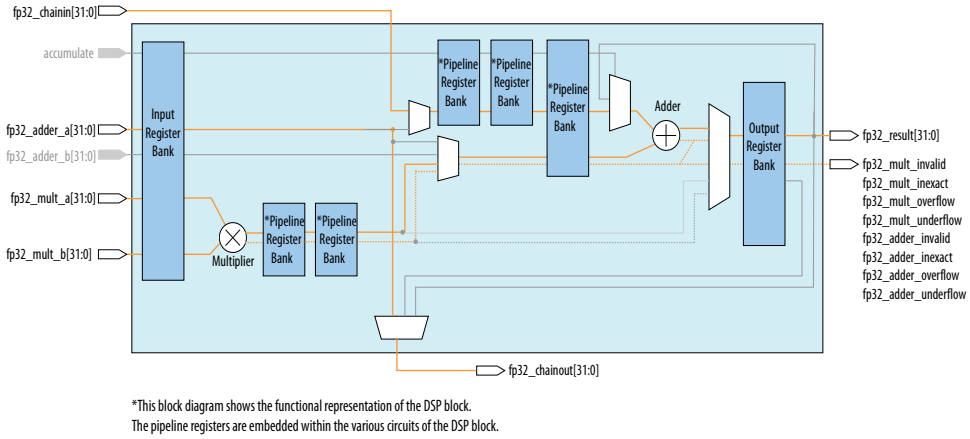
Table 16. Equations Applied to FP32 Vector One Mode

Chainin Parameter	Vector One with Floating-point Addition	Vector One with Floating-point Subtraction
Disable	fp32_result = fp32_mult_a * fp32_mult_b fp32_chainout = fp32_adder_a	fp32_result = fp32_mult_a * fp32_mult_b fp32_chainout = fp32_adder_a
Enable	fp32_result = (fp32_mult_a * fp32_mult_b) + fp32_chainin fp32_chainout = fp32_adder_a	fp32_result = (fp32_mult_a * fp32_mult_b) - fp32_chainin fp32_chainout = fp32_adder_a

The FP32 vector one mode supports the following exception flags:

- `fp32_mult_invalid`
- `fp32_mult_inexact`
- `fp32_mult_overflow`
- `fp32_mult_underflow`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

Figure 31. Vector One Mode



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.1.5. FP32 Vector Two Mode

This mode performs single-precision floating-point multiplication for input `fp32_mult_a` and input `fp32_mult_b`, and direct the result to chainout. The chainin input from the previous variable DSP Block is then added or subtracted from input `fp32_adder_a` as the output result.

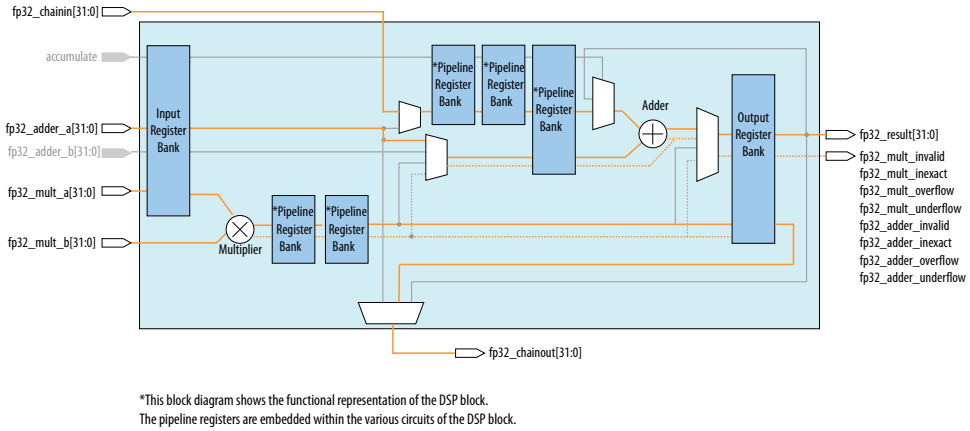
Table 17. Equations Applied to FP32 Vector Two Mode

Chainin Parameter	Vector Two with Floating-point Addition	Vector Two with Floating-point Subtraction
Disable	fp32_result = fp32_adder_a fp32_chainout = fp32_mult_a * fp32_mult_b	fp32_result = fp32_adder_a fp32_chainout = fp32_mult_a * fp32_mult_b
Enable	fp32_result = fp32_adder_a + fp32_chainin fp32_chainout = fp32_mult_a * fp32_mult_b	fp32_result = fp32_adder_a - fp32_chainin fp32_chainout = fp32_mult_a * fp32_mult_b

The FP32 vector two mode supports the following exception flags:

- fp32_mult_invalid
- fp32_mult_inexact
- fp32_mult_overflow
- fp32_mult_underflow
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

Figure 32. FP32 Vector Two Mode



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2. FP16 Half-precision Floating-point Arithmetic Functions

The FP16 half-precision floating-point arithmetic DSP can perform the following:

- Sum of two multiplication
- Sum of two multiplication with addition
- Sum of two multiplication with accumulation
- Vector one
- Vector two
- Vector three

Each of the functions supports:

- Extended precision format
- Flushed precision format
- Bfloat16 and bfloat+ formats

3.2.2.1. FP16 Supported Precision Formats

The FP16 half-precision floating-point arithmetic functions support the following formats:

- Flushed - use IEEE-754 half-precision format (binary16) for multiplier inputs and FP16 multiplication/addition/subtraction operations.
- Extended - use IEEE-754 half-precision format (binary16) for multiplier inputs. Use extended format for FP16 multiplication/addition/subtraction operations.
- Bfloat16 - multiplier inputs can be configured to accept 16-bit bfloat16 format or 19-bit extended bfloat16+ format. Use extended format for FP16 multiplication/addition/subtraction operations.

The following table shows the differences between the formats:

Table 18. Differences between Flushed, Extended, and Bfloat Formats

Features	Flushed	Extended	Bfloat16/Bfloat 16+
Input format (sign.exponent.mantissa)	1.5.10	1.5.10	1.8.7 or 1.8.10 (Bfloat16+)
FP16 operation format (sign.exponent.mantissa)	1.5.10	1.8.10	1.8.10
Input width	16 bit	16 bit	16 or 19 bit (Bfloat16+)
Minimum representable exponent	5'h01 - 5'h0f = -14	8'h01 - 8'h7f = -126	8'h01 - 8'h7f = -126
FP16 Subnormal	No support for subnormal. Subnormal result is flushed to zero.	Subnormal results can be represented as normal numbers	No support for subnormal. Subnormal result is flushed to zero.
Exception flags	Overflow, underflow, inexact, and invalid	Infinite, zero, inexact, and invalid	Overflow, underflow, inexact, and invalid
Invalid flag behavior	Asserted when there is an ill-defined operation	Asserted when there is an ill-defined operation or a qNaN input	Asserted when there is an ill-defined operation
Rounding	Round to nearest even (RNE)	RNE: • if both FP16 operands are normal numbers • if one of the FP16 operands is a subnormal number and mantissa product is ≥ 1 • if one of the FP16 operands is a subnormal number and mantissa product = "0.1111111111| 1xxxxxxxxx" • when using adder/ subtractor operations Round to zero(RZ) • if both FP16 operands are subnormal numbers • if one of the FP16 operands is a subnormal number and mantissa product is ≤ 1	RZ

3.2.2.2. Sum of Two FP16 Multiplication Mode

This mode performs a summation of two half-precision multiplication and provide a single-precision result:

fp32_result = (fp16_mult_top_a*fp16_mult_top_b) +
(fp16_mult_bot_a*fp16_mult_bot_b)

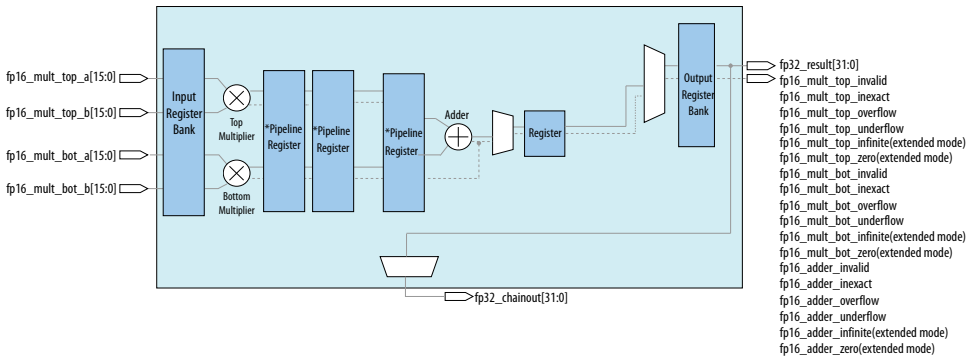
The following are exception flags supported in flushed and bfloat16 formats:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`

The following are exception flags supported in extended format:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_infinite`
- `fp16_mult_top_zero`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_infinite`
- `fp16_mult_bot_zero`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_infinite`
- `fp16_adder_zero`

Figure 33. Sum of Two FP16 Multiplication Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2.3. Sum of Two FP16 Multiplication with FP32 Addition Mode

This mode performs a summation of two half-precision multiplication, provide a 32-bit result, and add with a single-precision number:

fp32_result = (fp16_mult_top_a*fp16_mult_top_b) +
(fp16_mult_bot_a*fp16_mult_bot_b) + fp32_adder_a

The following are exception flags supported in flushed and bfloat16 formats:

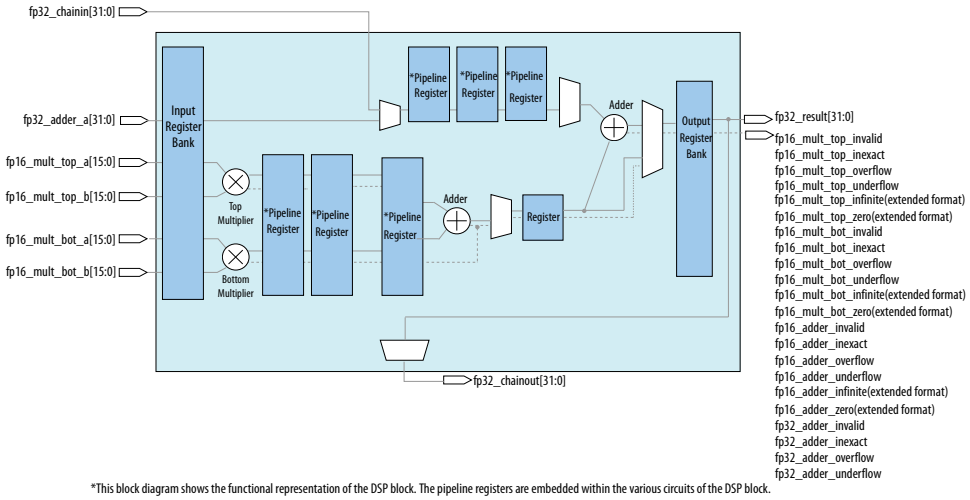
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

The following are exception flags supported in extended format:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_infinite`

- fp16_mult_top_zero
- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_infinite
- fp16_mult_bot_zero
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_infinite
- fp16_adder_zero
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

Figure 34. Sum of Two FP16 Multiplication with FP32 Addition Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2.4. Sum of Two FP16 Multiplication with Accumulation Mode

This mode performs a summation of two half-precision multiplication and accumulate the value into single-precision format:

fp32_result(t) = [fp16_mult_top_a(t) * fp16_mult_top_b(t)] + [fp16_mult_bot_a(t) * fp16_mult_bot_b(t)] + fp32_result(t-1)

The following are exception flags supported in flushed and bfloat16 formats:

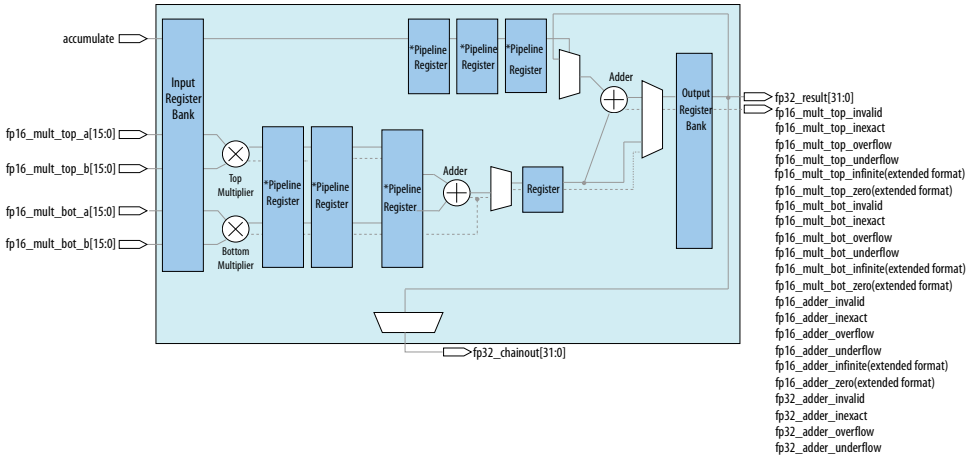
- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_overflow
- fp16_mult_top_underflow

- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_overflow
- fp16_mult_bot_underflow
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_overflow
- fp16_adder_underflow
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

The following are exception flags supported in extended format:

- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_infinite
- fp16_mult_top_zero
- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_infinite
- fp16_mult_bot_zero
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_infinite
- fp16_adder_zero
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

Figure 35. Sum of Two FP16 Multiplication with Accumulation Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2.5. FP16 Vector One Mode

This mode performs a summation of two half-precision multiplications with the chainin input from the previous variable DSP Block. The output is a single-precision floating-point value which is fed into chainout.

Table 19. Equations Applied to FP16 Vector One Mode

Chainin Parameter	Vector One with Floating-point Addition	Vector One with Floating-point Subtraction
Disable	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) fp32_chainout = fp32_adder_a	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) fp32_chainout = fp32_adder_a
Enable	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) + fp32_chainin fp32_chainout = fp32_adder_a	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) - fp32_chainin fp32_chainout = fp32_adder_a

The following are exception flags supported in flushed and bfloat16 formats:

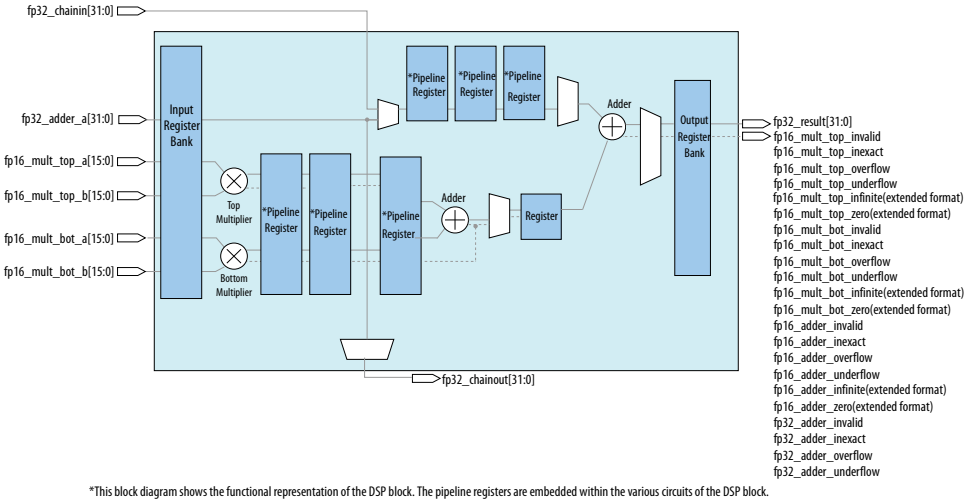
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`

- fp16_adder_underflow
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

The following are exception flags supported in extended format:

- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_infinite
- fp16_mult_top_zero
- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_infinite
- fp16_mult_bot_zero
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_infinite
- fp16_adder_zero
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

Figure 36. FP16 Vector One Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2.6. FP16 Vector Two Mode

This mode performs a summation of two half precision multiplication and fed to chainout. The chainin input from the previous variable DSP Block is then added or subtracted from input `fp32_adder_a` as the output result.

Table 20. Equations Applied to FP16 Vector Two Mode

Chainin Parameter	Vector Two with Floating-point Addition	Vector Two with Floating-point Subtraction
Disable	fp32_result = fp32_adder_a fp32_chainout = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b)	fp32_result = fp32_adder_a fp32_chainout = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b)
Enable	fp32_result = fp32_adder_a + fp32_chainin fp32_chainout = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b)	fp32_result = fp32_adder_a - fp32_chainin fp32_chainout = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b)

The following are exception flags supported in flushed and bfloat16 formats:

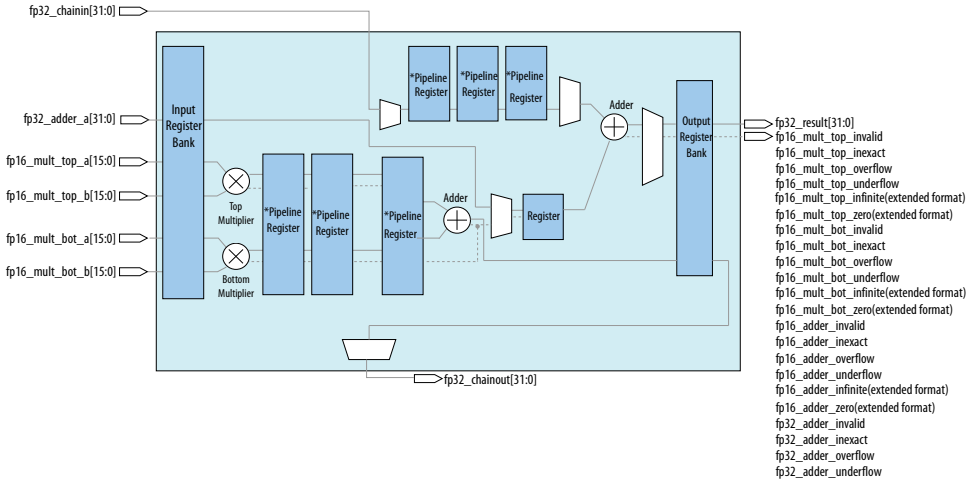
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

The following are exception flags supported in extended format:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_infinite`
- `fp16_mult_top_zero`
- `fp16_mult_bot_invalid`

- fp16_mult_bot_inexact
- fp16_mult_bot_infinite
- fp16_mult_bot_zero
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_infinite
- fp16_adder_zero
- fp32_adder_invalid
- fp32_adder_inexact
- fp32_adder_overflow
- fp32_adder_underflow

Figure 37. FP16 Vector Two Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.2.7. FP16 Vector Three Mode

This mode performs a single-precision accumulation and a summation of two half-precision multiplications.

Table 21. Equations Applied to Vector Three Mode

Accumulate Input	Vector Three with Floating-point Addition	Vector Three with Floating-point Subtraction
Disable	fp32_result(t) = fp32_adder_a(t) fp32_chainout = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b}	fp32_result(t) = fp32_adder_a(t) fp32_chainout = {fp16_mult_top_a * fp16_mult_top_b} - {fp16_mult_bot_a * fp16_mult_bot_b}
Enable	fp32_result(t) = fp32_adder_a(t) + fp32_result(t-1) fp32_chainout = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b}	fp32_result(t) = fp32_adder_a(t) - fp32_result(t-1) fp32_chainout = {fp16_mult_top_a * fp16_mult_top_b} - {fp16_mult_bot_a * fp16_mult_bot_b}

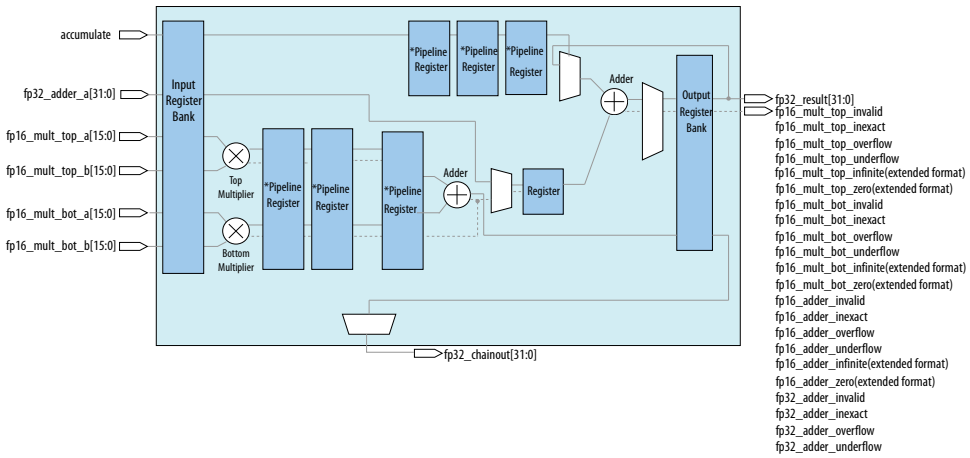
The following are exception flags supported in flushed and bfloat16 formats:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

The following are exception flags supported in extended format:

- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_infinite`
- `fp16_mult_top_zero`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_infinite`
- `fp16_mult_bot_zero`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_infinite`
- `fp16_adder_zero`
- `fp32_adder_invalid`
- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

Figure 38. FP16 Vector Three Mode



*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

3.2.3. Multiple Floating-point Variable DSP Blocks Functions

Two or more floating-point DSP blocks can perform the following:

- Multiply-add or multiply-subtract mode which uses single floating-point arithmetic DSP if the chainin parameter is turn off
- Direct vector dot product
- Complex multiplication

3.2.3.1. Multiply-Add or Multiply-Subtract Mode

This mode performs floating-point multiplication followed by floating-point addition or floating-point subtraction. The chainin parameter allows you to enable a multiple-chain mode.

Table 22. Equations Applied to Multiply-Add or Multiply-Subtract Mode

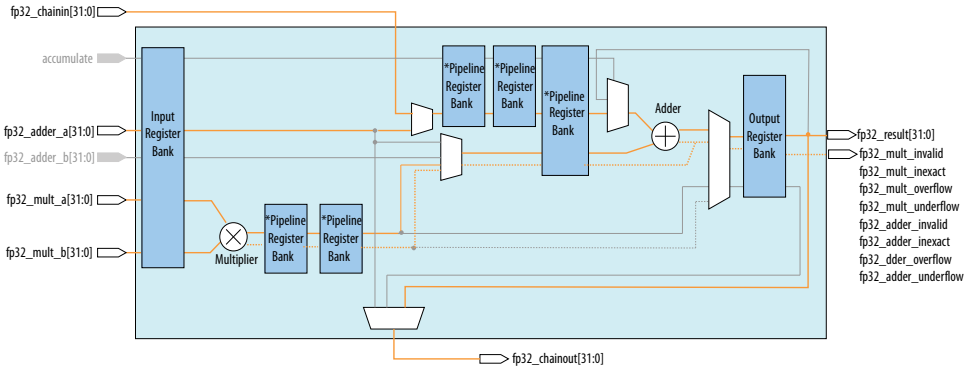
Chainin Parameter	Multiply-Add Mode	Multiply-Subtract Mode
Disable	fp32_result = (fp32_mult_a*fp32_mult_b) + fp32_adder_a	fp32_result = (fp32_mult_a*fp32_mult_b) - fp32_adder_a
Enable	fp32_result = (fp32_mult_a*fp32_mult_b) + fp32_chainin	fp32_result = (fp32_mult_a*fp32_mult_b) - fp32_chainin

The floating-point multiply-adder or multiply-subtract mode supports the following exception flags:

- `fp32_mult_invalid`
- `fp32_mult_inexact`
- `fp32_mult_overflow`
- `fp32_mult_underflow`
- `fp32_adder_invalid`

- `fp32_adder_inexact`
- `fp32_adder_overflow`
- `fp32_adder_underflow`

Figure 39. Multiply-Add or Multiply-Subtract Mode for Intel Agilex 7 Devices



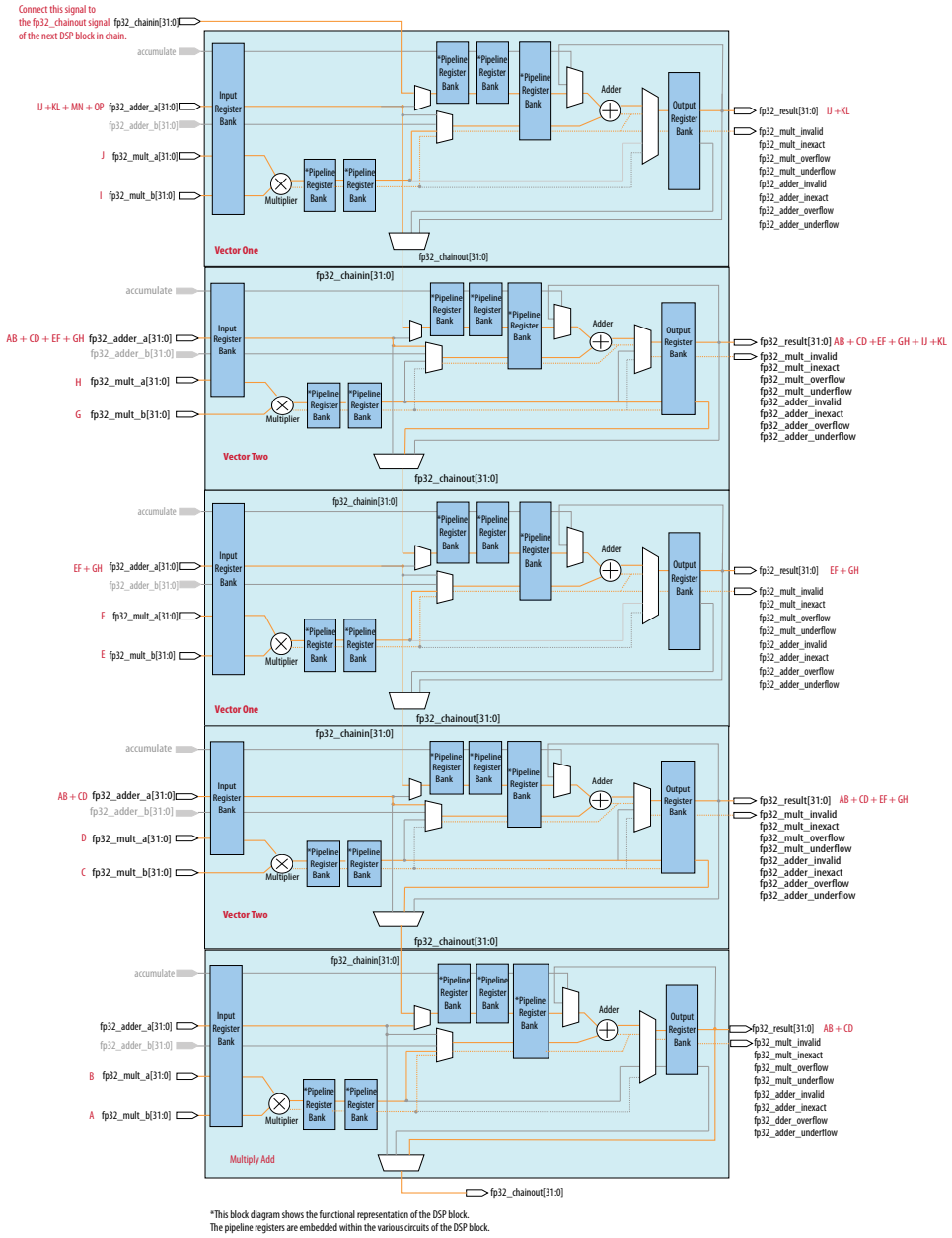
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.2.3.2. Direct Vector Dot Product

The following figures shows the combination of DSP blocks to create direct vector dot product. For FP32 single-precision floating-point arithmetic, the direct vector dot product consists of:

- Multiply-add and subtract mode with chainin parameter turned on
- Vector one
- Vector two

Figure 40. Direct Vector Dot Product Using FP32 Single-precision Floating-point Arithmetic

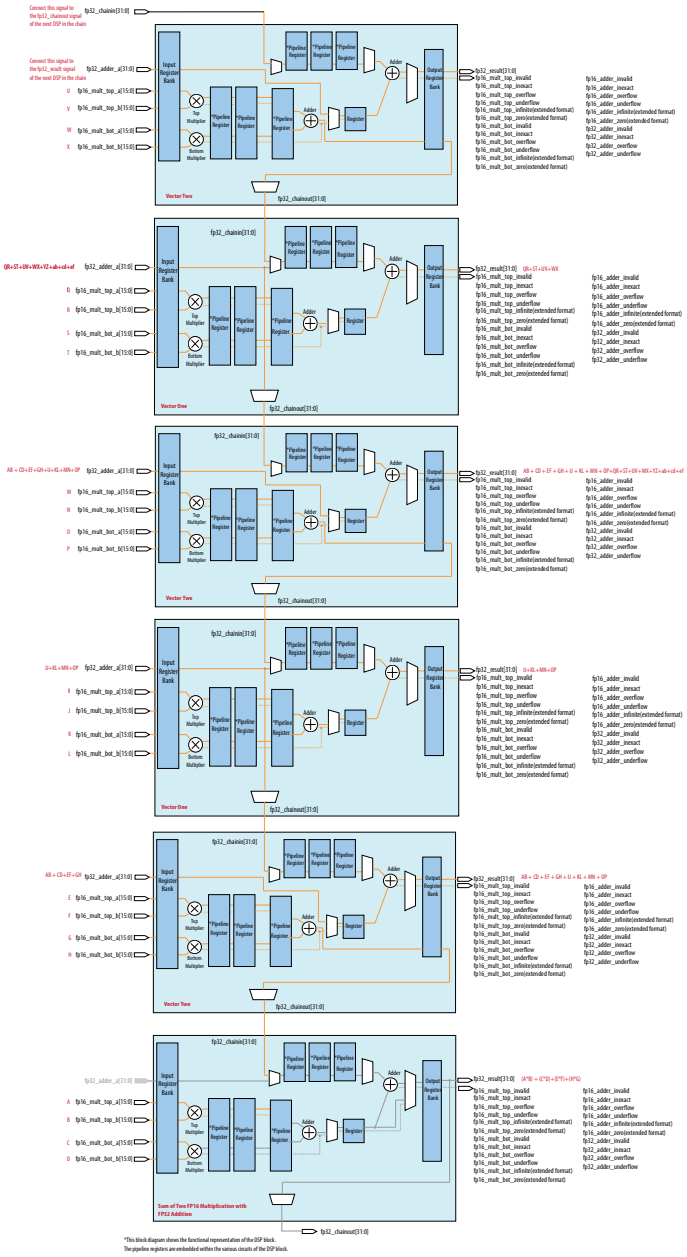


*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

For FP16 half-precision floating-point arithmetic, the direct vector dot product consists of:

- Sum of two multiplication with FP32 addition mode with chainin feature enabled
- Vector one
- Vector two

Figure 41. Direct Vector Dot Product Using FP16 Half-precision Floating-point Arithmetic



3.2.3.3. Complex Multiplication

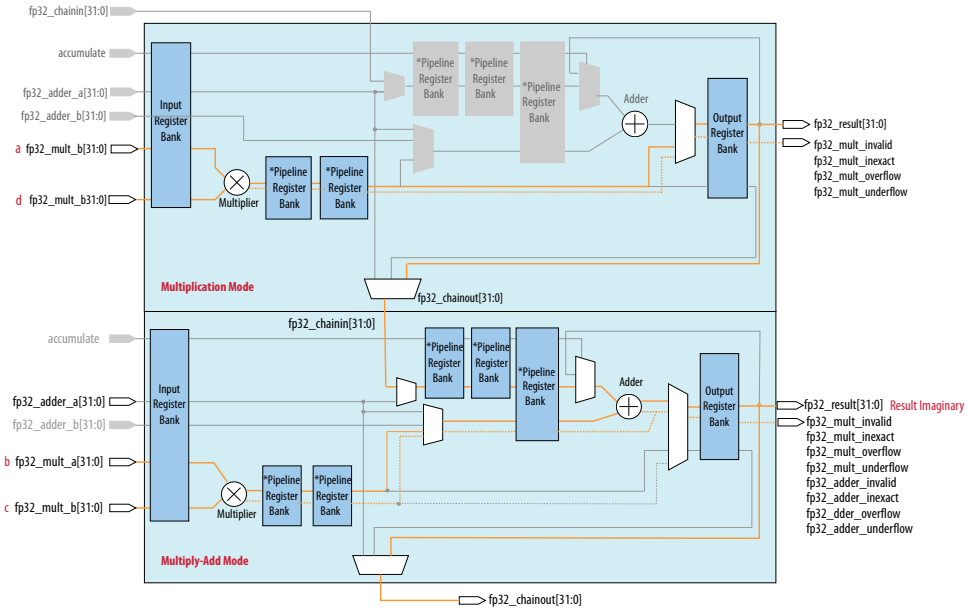
The devices support the floating-point arithmetic single precision complex multiplier using four variable-precision DSP blocks.

Figure 42. Sample of Complex Multiplication Equation

$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j[(a \times d) + (b \times c)]$$

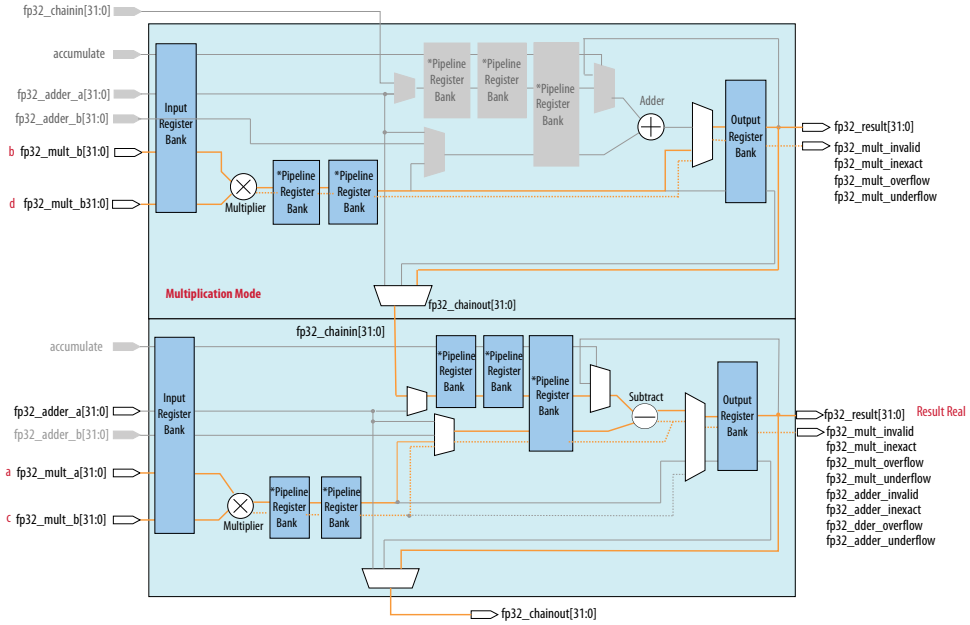
The imaginary part [(a × d) + (b × c)] is implemented in the first two variable-precision DSP blocks, while the real part [(a × c) - (b × d)] is implemented in the next two variable-precision DSP blocks.

Figure 43. Complex Multiplication with Imaginary Result Using FP32 Single-precision Floating-point Arithmetic



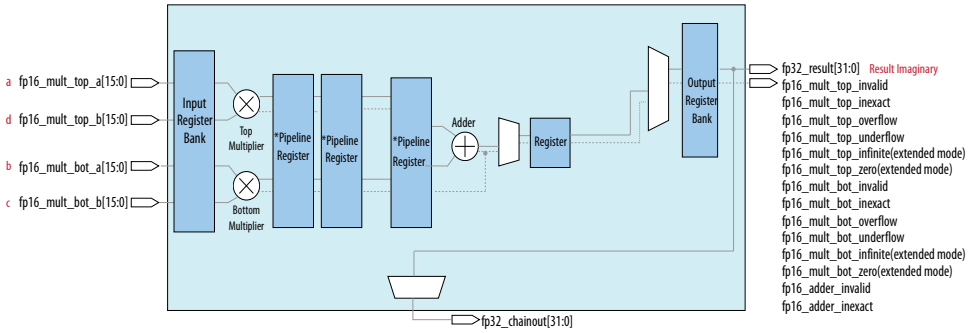
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Figure 44. Complex Multiplication with Result Real Using FP32 Single-precision Floating-point Arithmetic



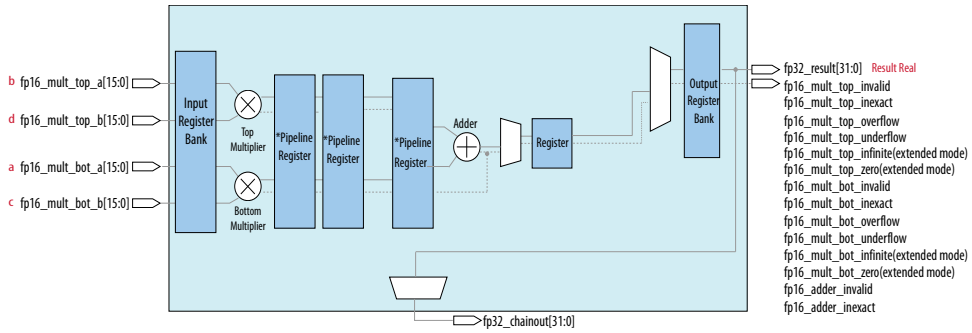
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Figure 45. Complex Multiplication with Imaginary Result Using FP16 Half-precision Floating-point Arithmetic



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Figure 46. Complex Multiplication with Result Real Using FP16 Half-precision Floating-point Arithmetic



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

4. Intel Agilex 7 Variable Precision DSP Blocks Design Considerations

You should consider the following elements in your design:

Table 23. Design Considerations

DSP Functions	Design Elements
Fixed-point arithmetic	• Operational modes • Input, pipeline, and output registers • Internal coefficient and pre-adder • Accumulator • Chainout adder • Input cascade
Floating-point arithmetic	• Input, pipeline, and output registers • Operational modes • Chainout adder

4.1. Fixed-point Arithmetic

4.1.1. Configurations for Input, Pipeline, and Output Registers

The configurations for the input, pipeline, and output registers are restricted due to the timing model in Intel Agilex 7 devices. Therefore these registers only support certain configurations.

4.1.1.1. Restrictions for Input Registers

The following are the clock enable restrictions for input registers:

- When using 9 x 9 sum of 4 operational mode, the following input signal pairs must use the same clock enable settings:
  - ax and bx
  - ay and by
  - cx and dx
  - cy and dy
- If the input registers for SUB, NEGATE, ACCUMULATE, and LOADCONST signals are enabled, these registers must use the same clock enable settings.
- Disable the input registers for SUB, NEGATE, ACCUMULATE, and LOADCONST signals if these signals are driven by a constant value.

4.1.1.2. Restrictions for Pipeline Registers

The following are the clock enable restrictions for pipeline registers:

- When the pipeline registers for LOADCONST or ACCUMULATE signals are enabled, the pipeline registers for all the multiplier inputs must be enabled and use the same clock enable settings.
- Disable the pipeline registers for LOADCONST or ACCUMULATE signals if these signals are driven by a constant value.

4.1.1.3. Supported Register Configurations per Operation Modes

Table 24. Supported Register Configurations per Operation Modes

Operation Mode	Register Level	Input Register	Pipeline Register	2nd Pipeline Register	Output Register
9 x 9 Sum of 4 Mode	0	Disable	Disable	Disable	Disable
	1	Enable	Disable	Disable	Disable
	1(5)	Disable	Disable	Disable	Enable
	2	Enable	Disable	Disable	Enable
	2	Enable	Disable	Enable	Disable
	3	Enable	Disable	Enable	Enable
	3	Enable	Enable	Enable	Disable
	4	Enable	Enable	Enable	Enable
Independent 18 x 19 multiplication	0	Disable	Disable	Disable	Disable
	1	Enable	Disable	Disable	Disable
	2	Enable	Disable	Disable	Enable
	2	Enable	Disable	Enable	Disable
	3(6)	Enable	Enable	Disable	Enable
	3(7)	Enable	Disable	Enable	Enable
	3	Enable	Enable	Enable	Disable
	4	Enable	Enable	Enable	Enable
Two 18 x 19 multiplier adder mode	0	Disable	Disable	Disable	Disable
	1	Enable	Disable	Disable	Disable
	1(5)	Disable	Disable	Disable	Enable
	2	Enable	Disable	Disable	Enable
	2	Enable	Disable	Enable	Disable
	3(6)	Enable	Enable	Disable	Enable

continued...

(5) When Accumulator is enabled

(6) When Pre-Adder and/or Coefficient are enabled

(7) When Pre-Adder and/or Coefficient are disabled

Operation Mode	Register Level	Input Register	Pipeline Register	2nd Pipeline Register	Output Register
	3[7]	Enable	Disable	Enable	Enable
	3	Enable	Enable	Enable	Disable
	4	Enable	Enable	Enable	Enable
18 x 18 multiplier adder summed with 36-bit input	0	Disable	Disable	Disable	Disable
	1	Enable	Disable	Disable	Disable
	1[5]	Disable	Disable	Disable	Enable
	2	Enable	Disable	Disable	Enable
	2	Enable	Disable	Enable	Disable
	3	Enable	Disable	Enable	Enable
	3	Enable	Enable	Enable	Disable
	4	Enable	Enable	Enable	Enable
18 x 19 systolic mode	1	Disable	Disable	Disable	Enable
	2	Enable	Disable	Disable	Enable
	3[6]	Enable	Enable	Disable	Enable
	3[7]	Enable	Disable	Enable	Enable
	4	Enable	Enable	Enable	Enable
Independent 27 x 27 multiplication	0	Disable	Disable	Disable	Disable
	1	Enable	Disable	Disable	Disable
	1[5]	Disable	Disable	Disable	Enable
	2	Enable	Disable	Disable	Enable
	2	Enable	Disable	Enable	Disable
	3[6]	Enable	Enable	Disable	Enable
	3[7]	Enable	Disable	Enable	Enable
	3	Enable	Enable	Enable	Disable
	4	Enable	Enable	Enable	Enable

4.1.2. Internal Coefficient and Pre-Adder for Fixed-point Arithmetic

In both 18-bit and 27-bit modes, you can use the coefficient feature and pre-adder feature independently.

When pre-adder feature is enabled in 18-bit modes, you must enable both top and bottom pre-adder.

When internal coefficient feature is enabled in 18-bit modes, you must enable both top and bottom coefficient.

4.1.3. Accumulator for Fixed-point Arithmetic

The accumulator in the devices supports double accumulation by enabling the 64-bit double accumulation registers located between the output register bank and the accumulator.

4.1.4. Input Cascade for Fixed-point Arithmetic

The input register bank in Intel Agilex 7 variable precision DSP block supports input cascade feature. This feature provides the capability of cascading the input bus within a DSP block and to another DSP block.

When you enable the input cascade feature in 18 x 19 mode:

- The top multiplier Y input drives the bottom multiplier Y input within a DSP block
- The bottom multiplier Y input of the first DSP block drives the top multiplier Y input of the subsequent DSP block

For 27 × 27 mode, the multiplier Y input of the first DSP block drives the multiplier Y input of the subsequent DSP block. This feature is not supported with pre-adder enabled.

There are two delay registers that you can use to balance the latency requirements when you use both the input cascade and chainout features in fixed-point arithmetic 18 x 19 mode. These are the top delay registers and bottom delay registers. The ay input register must be enabled when top delay register is enabled. The clock enable for both registers must be the same. Similarly, the by input register must be enabled when bottom delay register is enabled. The clock enable for both registers must be the same.

The delay registers are only supported in 18 x 18 or 18 x 19 independent multiplier, multiplier adder sum mode and 18-bit systolic FIR mode.

Figure 47. Input Cascade in Fixed-point Arithmetic 18 x 19 Mode

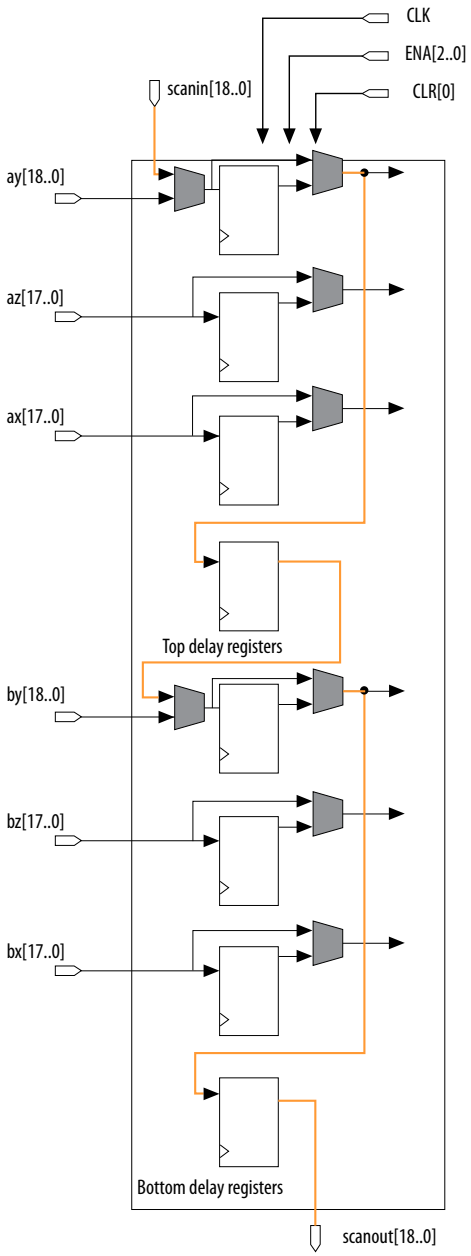
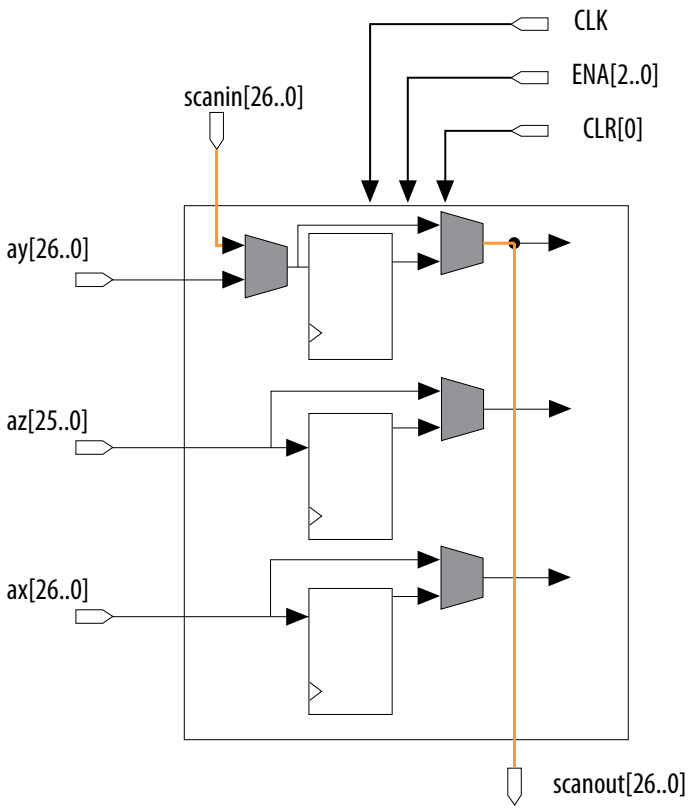


Figure 48. Input Cascade in Fixed-point Arithmetic 27 x 27 Mode



Related Information

DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.

4.1.4.1. Dynamic Scanin

When input cascade is used, the source of top multiplier can be dynamically switched between SCANIN and AY by asserting/de-asserting DISABLE_SCANIN input.

Figure 49. Dynamic Scanin

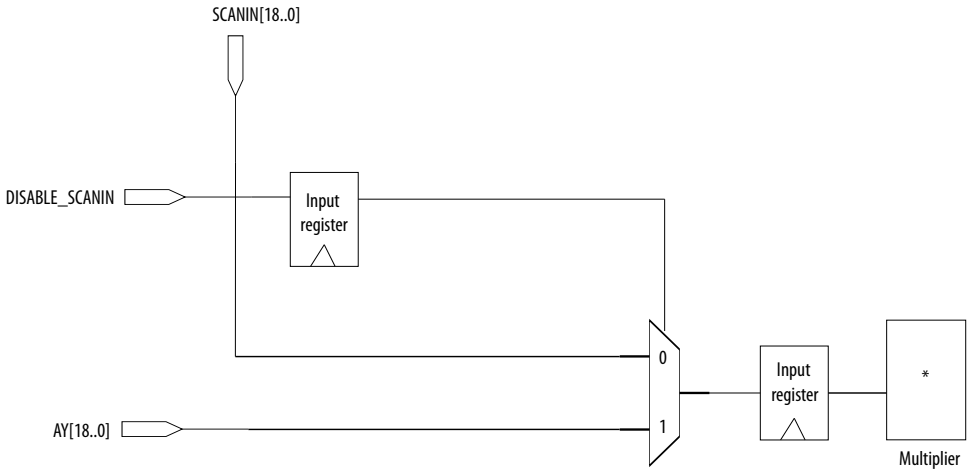


Table 25. DISABLE_SCANIN Signal Behavior

DISABLE_CHAINOUT Signal	Description
Low (0)	Source of multiplier input is from SCANIN input.
High (1)	Source of multiplier input is switched from SCANIN to AY.

When DISABLE_SCANIN port is used, the input register for this signal will be enabled. The register is driven by free running clock and there is no clock enable or clock clear signal to control this register.

4.1.5. Chainout Adder

You can use the output chaining path to add results from another DSP block. The output chainout port can be dynamically disable by asserting the DISABLE_CHAINOUT signal.

The chainout adder support all operational modes except for 18 x 18 or 18 x 19 independent multiplier mode.

When DISABLE_CHAINOUT port is used, the input register for this signal will be enabled. The register is driven by free running clock and there is no clock enable or clock clear signal to control this register.

4.2. Floating-point Arithmetic

4.2.1. Configurations for Input, Pipeline, and Output Registers

The configurations for the input, pipeline, and output registers are restricted due to the timing model in Intel Agilex 7 devices. Therefore these registers only support certain configurations.

You must enable all registers within the same register level but you can use different clock enables. However, when port `accumulate` is connected to constant VCC, the register settings for `accumulate_clken`, `accum_pipeline_clken`, `accum_2nd_pipeline_clken`, and `accum_adder_clken` should be disabled to avoid register clear signal interrupting the constant VCC.

The following registers should have the same clock enable settings:

- Registers `adder_input_clken` and `accum_adder_clken` when operation_mode is set to FP32 multiplication with accumulation mode, sum of two FP16 multiplication with accumulation mode, or FP16 vector three mode.
- Registers `fp16_mult_input_clken` and `fp32_adder_a_clken` when in all FP16 operation modes except FP16 vector three mode.

4.2.1.1. FP32 Operation Modes Supported Register Configurations

Table 26. Supported Register Configurations For FP32 Multiplication Mode

Latency	Input Register		Pipeline Register		Output Register
	fp32_mult_a_clken	fp32_mult_b_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Enable	Enable
≥3	Disable	Enable	Disable, enable	Enable	Enable

Table 27. Supported Register Configurations For FP32 Addition or Subtraction Mode

Latency	Data Input Register		Pipeline Register		Adder Input Register	Output Register
	fp32_adder_a_clken	fp32_adder_b_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_2nd_pl_clken	adder_input_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Disable, enable	Disable, enable	Enable	Enable

Table 28. Supported Register Configurations For FP32 Multiplication with Addition or Subtraction Mode

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Output Register
	fp32_adder_a_clken	fp32_mult_a_clken	fp32_mult_b_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_2nd_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable
≥4	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable

Table 29. Supported Register Configurations For FP32 Multiplication with Accumulation Mode

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register		Output Register
	accumulate_clken	fp32_mult_a_clken	fp32_mult_b_clken	accum_pipeline_clken	accum_2nd_pipeline_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	accum_adder_clken	adder_input_clken	output_clken
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable
≥4	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable	Enable

Table 30. Supported Register Configurations For FP32 Vector One Mode

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Output Register
	fp32_adder_a_clken	fp32_mult_a_clken	fp32_mult_b_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable

continued...

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Output Register
	fp32_adder_a_clken	fp32_mult_a_clken	fp32_mult_b_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	output_clken
2	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable
≥4	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable

Table 31. Supported Register Configurations For FP32 Vector Two Mode

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Output Register
	fp32_adder_a_clken	fp32_mult_a_clken	fp32_mult_b_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable

4.2.1.2. FP16 Operation Mode Supported Register Configurations

Table 32. Supported Register Configurations For Sum of Two FP16 Multiplication Mode

Latency	Data Input Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Adder Pipeline Register	Output Register
	fp16_mult_input_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	adder_pl_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Disable	Disable	Disable	Disable	Enable
3	Enable	Disable	Disable	Enable	Disable	Enable
4	Enable	Disable	Disable	Enable	Enable	Enable
≥5	Enable	Disable, enable	Enable	Enable	Enable	Enable

Table 33. Supported Register Configurations For Sum of Two FP16 Multiplication with FP32 Addition Mode

Latency	Data Input Register		Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Adder Pipeline Register	Output Register
	fp32_adder_a_clken	fp16_mult_input_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_2nd_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	adder_pl_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Disable	Enable
≥4	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable
≥5	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable	Enable

Table 34. Supported Register Configurations For Sum of Two FP16 Multiplication with Accumulation Mode

Latency	Data Input Register		Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register		Adder Pipeline Register	Output Register
	accumulate_clken	fp16_mult_input_clken	accum_pipeline_clken	accum_2nd_pipeline_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	accum_adder_clken	adder_input_clken	adder_pl_clken	output_clken
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Disable	Enable
≥4	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable	Enable
≥5	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable	Enable	Enable

Table 35. Supported Register Configurations For FP16 Vector One Mode

Latency	Data Input Register		Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Adder Pipeline Register	Output Register
	fp32_adder_a_clken	fp16_mult_input_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_2nd_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	adder_pl_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Disable	Enable
≥4	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable
≥5	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable

Table 36. Supported Register Configurations For FP16 Vector Two Mode

Latency	Data Input Register		Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register	Adder Pipeline Register	Output Register
	fp32_adder_a_clken	fp16_mult_input_clken	fp32_adder_a_chainin_pl_clken	fp32_adder_a_chainin_2nd_pl_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	adder_input_clken	adder_pl_clken	output_clken
0	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable
≥4	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable

Table 37. Supported Register Configurations For FP16 Vector Three Mode

Latency	Data Input Register			Adder 1st Pipeline Register	Adder 2nd Pipeline Register	Multiplier 1st Pipeline Register	Multiplier 2nd Pipeline Register	Adder Input Register		Adder Pipeline Register	Output Register
	accumulate_clken	fp32_adder_a_clken	fp16_mult_input_clken	accum_pipeline_clken	accum_2nd_pipeline_clken	mult_pipeline_clken	mult_2nd_pipeline_clken	accum_adder_clken	adder_input_clken	adder_pl_clken	output_clken
1	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
2	Enable	Enable	Enable	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Enable
≥3	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable	Disable	Enable	Enable	Enable	Enable
≥4	Enable	Enable	Enable	Disable, enable	Disable, enable	Disable, enable	Enable	Enable	Enable	Enable	Enable

4.2.2. Chainout Adder

You can use the output chaining path to add results from another DSP block.

Support for certain operation modes:

- Multiply-add or multiply-subtract mode
- Vector one mode
- Vector two mode

4.3. DSP Block Cascade Limit in Intel Agilex 7 Devices

The sector size limits the number of DSP blocks cascade. For Intel Agilex 7 devices, you can cascade up to 41 DSP blocks.

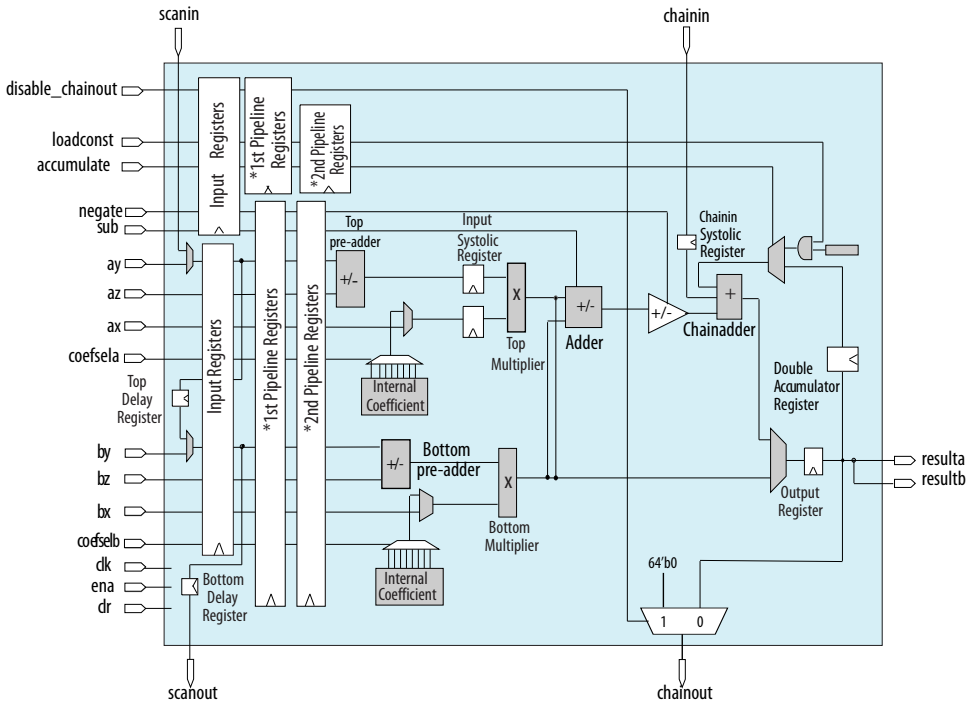
5. Native Fixed Point DSP Intel Agilex FPGA IP Core References

The Native Fixed Point DSP Intel Agilex FPGA IP core instantiates and controls a single Intel Agilex 7 Variable Precision DSP block.

Operational modes supported in this IP core include:

- 9 × 9 sum-of-4 mode
- 18 × 18 full mode
- 18 × 18 sum-of-2 mode
- 18 × 18 plus 36 mode
- 18 × 18 systolic mode
- 27 × 27 mode

Figure 50. Native Fixed Point DSP Intel Agilex FPGA IP Core Functional Block Diagram



*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

5.1. Native Fixed Point DSP Intel Agilex FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 38. Native Fixed Point DSP Intel Agilex FPGA IP Release Information

Item	Description
IP Version	19.1.1
Intel Quartus Prime Version	21.2
Release Date	2021.06.23

5.2. Supported Operational Modes

Table 39. Operational Modes Supported by Native Fixed Point DSP Intel Agilex FPGA IP Core

Operational Modes	Description
9 × 9 Sum of 4 Mode	This mode operates as sum of four 9 (signed) × 9 (signed) or 8 (unsigned) × 8 (unsigned) multipliers with 20 to 64 bits output when chainout adder or accumulator is enabled. This mode applies the following equations: • resulta = (ax * ay)+(bx * by)+(cx * cy)+(dx * dy)
18 × 18 Full Mode	This mode operates as two independent 18 (signed) x 19 (signed) or 18 (unsigned) x 18 (unsigned) multipliers with 37-bit output. This mode applies the following equation: • resulta = ax * ay • resultb = bx * by
18 × 18 Sum of Two Mode	This mode operates as sum of two 18 × 19 multiplication. This mode applies the equations of: • resulta = [(bx * by) + (ax * ay)] when sub signal is driven low. • resulta = [(bx * by) - (ax * ay)] when sub signal is driven high. The `resulta` output bus can support up to 64 bits when you enable accumulator or chainout adder.
18 × 18 Plus 36 Mode	This mode operates as one 18 × 19 multiplication summed to a 36-bit input. This mode applies the equation of resulta = (ax * ay) + (bx * by). When the input bus is less than 36-bit in this mode, you are required to provide the necessary signed extension to fill up the 36-bit input. When you enable the accumulator, the `resulta` output bus can support up to 64 bits.
18 × 18 Systolic Mode	This mode operates as 18-bit systolic FIR.

continued...

Operational Modes	Description
	Enable the input systolic register and the output register when using this operational mode. When you enable the chainout adder, the chainout and chainin width can support up to 44 bits. When you enable the accumulator, the `resulta` output bus can support up to 64 bits.
27 × 27 Mode	This mode operates as one independent 27(signed/unsigned) × 27(signed/unsigned) multiplier. This mode applies the equation of resulta = ax * ay. The `resulta` output bus can support up to 64 bits when you enable accumulator or chainout adder.

5.3. Maximum Input Data Width for Fixed-point Arithmetic

Table 40. Maximum Input Data Width for 9 x 9 Sum of 4 Operational Mode

ax	ay	bx	by	cx	cy	dx	dy	chainin
9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	9 (signed) 8 (unsigned)	64

Table 41. Maximum Input Data Width for 18 x 18 Fixed-point Arithmetic Operational Modes

Operation Mode	Maximum Input Data Width						
	ax	ay	az	bx	by	bz	chainin
Without Pre-adder or Internal Coefficient							
m18×18_full	18 (signed) 18 (unsigned) (8)	19 (signed) 18 (unsigned)	Not used	18 (signed) 18 (unsigned)	19 (signed) 18 (unsigned)	Not used	Not used
m18×18_sumof2	18 (signed) 18 (unsigned)(8)	19 (signed) 18 (unsigned)	Not used	18 (signed) 18 (unsigned)(8)	19 (signed) 18 (unsigned)	Not used	64
m18×18_systolic	18 (signed) 18 (unsigned)(8)	19 (signed) 18 (unsigned)	Not used	18 (signed) 18 (unsigned)(8)	19 (signed) 18 (unsigned)	Not used	44
m18x18_plus36	18 (signed) 18 (signed)	19 (signed) 18 (unsigned)	Not used	36 (signed) 36 (signed)	Not used	Not used	64
With Pre-adder Feature Only							
m18×18_full	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	Not used
m18×18_sumof2	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	64
m18×18_systolic	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	18 (signed) 18 (unsigned)(8)	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	44
With Internal Coefficient Feature Only							
m18×18_full	Not used	19 (signed) 18 (unsigned)	Not used	Not used	19 (signed) 18 (unsigned)	Not used	Not used
m18×18_sumof2	Not used	19 (signed) 18 (unsigned)	Not used	Not used	19 (signed) 18 (unsigned)	Not used	64
m18×18_systolic	Not used	19 (signed) 18 (unsigned)	Not used	Not used	19 (signed) 18 (unsigned)	Not used	44

continued...

(8) When using `negate` port, maximum width for this port is 17.

Operation Mode	Maximum Input Data Width						
	ax	ay	az	bx	by	bz	chainin
With Pre-adder and Internal Coefficient Features							
m18×18_full	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	Not used
m18×18_sumof2	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	64
m18×18_systolic	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	Not used	18 (signed) 17 (unsigned)	18 (signed) 17 (unsigned)	44

Table 42. Maximum Input Data Width for 27 x 27 Fixed-point Arithmetic Operational Mode

Operation Mode	Maximum Input Data Width						
	ax	ay	az	bx	by	bz	chainin
Without Pre-adder or Internal Coefficient							
m27×27	27 (signed) 27 (unsigned)[9]	27 (signed) 27 (unsigned)	Not used	Not used	Not used	Not used	64
With Pre-adder Feature Only							
m27×27	27 (signed) 27 (unsigned)[9]	26 (signed) 26 (unsigned)	26 (signed) 26 (unsigned)	Not used	Not used	Not used	64
With Internal Coefficient Feature Only							
m27×27	Not used	27 (signed) 27 (unsigned)	Not used	Not used	Not used	Not used	64
With Pre-adder and Internal Coefficient Features							
m27×27	Not used	26 (signed) 26 (unsigned)	26 (signed) 26 (unsigned)	Not used	Not used	Not used	64

5.3.1. Using Less Than 36-Bit Operand In 18 x 18 Plus 36 Mode Example

This example shows how to configure the Native Fixed Point DSP Intel Agilex FPGA IP to use 18 × 18 Plus 36 operational mode with a signed 12-bit input data of 101010101010 (binary) instead of a 36-bit operand.

1. Set **Representation format for bottom multiplier x operand** to **signed**.
2. Set **Representation format for bottom multiplier y operand** to **unsigned**.
3. Set **'bx' input bus width** to 18.
4. Set **'by' input bus width** to 18.
5. Provide 18-bit signed representation data, example,**'111111111111111111'**, to bx input bus.

[9] When using `negate` port, the maximum width for this port is 26.

This step is to perform sign extension. The initial 12 bits input is extended to 36 bits with bx representing the most significant 18 bits.

6. **'111111101010101010'**, to Provide data 18-bit signed representation data, example, by input bus.

5.4. Maximum Output Data Width for Fixed-point Arithmetic

Table 43. Maximum Output Data Width for 9 x 9 Sum of 4 Operational Mode

Operation Mode	Maximum Output Data Width	
	resulta	chainout
m9x9_sumof4	64	64

Table 44. Maximum Output Data Width for 18 x 18 Fixed-point Arithmetic Operational Modes

Operation Mode	Maximum Output Data Width						
	resulta	resultb	scanout				chainout
			Without Input Cascade Feature	When Input Cascade is Enabled for ay Input	When Input Cascade is Enabled for by Input	When Input Cascade is Enabled for ay and by Input	
m18×18_full	37	37	Use the same width as by port width.	Use the same width as by port width.	Use the same width as ay or scanin port width.	Use the same width as ay or scanin port width.	Not used
m18×18_sumof2	64	Not used	Use the same width as by.	Use the same width as by port width.	Use the same width as ay or scanin port width.	Use the same width as ay or scanin port width.	64
m18×18_systolic	44	37	Use the same width as by.	Use the same width as by port width.	Use the same width as ay or scanin port width.	Use the same width as ay or scanin port width.	44

Table 45. Maximum Output Data Width for 27 x 27 Fixed-point Arithmetic Operational Mode

Operation Mode	Maximum Output Data Width		
	resulta	scanout	chainout
m27×27	64	Use the same width as ay or scanin port width.	64

5.5. Parameterizing Native Fixed Point DSP IP

1. In Intel Quartus Prime Pro Edition, create a new project that targets the Intel Agilex 7 device.
2. In IP Catalog, click **Library ➤ DSP ➤ Primitive DSP ➤ Native Fixed Point DSP Intel Agilex FPGA IP**.
   The Native Fixed Point DSP IP parameter editor opens.
3. In the **New IP Variation** dialog box, enter an **Entity Name** and click **OK**.

4. Under **Parameters**, select the operation mode, multiplier configuration, clear signal, port width, and internal coefficient configurations according to the variant of your IP core
5. Click **Generate HDL**.
6. Click **Finish**.

5.5.1. Operation Mode Tab

Table 46. Operation Mode Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Please choose the operation mode	operation_mode	**m9x9_sum of4** **m18×18_full** **m18×18_sumof2** **m18×18_plus36** **m18×18_systolic** **m27×27**	**m18×18_full**	Select the desired operational mode.
1st Multiplier Configuration				
Representation format for 'ax' operand	signed_max	**unsigned** **signed**	**unsigned**	Specify the representation format for the first multiplier x operand.
Representation format for 'ay' operand	signed_may	**unsigned** **signed**	**unsigned**	Specify the representation format for the first multiplier y operand.
'ax' input bus width	ax_width	0–27	—	Specify the width of ax input bus. For more information about supported input width, refer to the related information.
Enable 'ax' input register	ax_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for ax input register. For more information about clock enable restrictions for input registers, refer to the related information.
'ay' or 'scanin' bus width	ay_scan_in_width	1–27	18	Specify the width of ay or scanin input bus. For more information about supported input width, refer to the related information.
Enable 'ay' or 'scanin' input register	ay_scan_in_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for ay or scanin input register. For more information about clock enable restrictions for input registers, refer to the related information.
2nd Multiplier Configuration				
Representation format for 'bx' operand	signed_mbx	**unsigned** **signed**	**unsigned**	Specify the representation format for second multiplier x operand.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
Representation format for 'by' operand	signed_mby	**unsigned** **signed**	**unsigned**	Specify the representation format for second multiplier y operand. Always select **unsigned** for **m18×18_plus36**.
'bx' input bus width	bx_width	0–36	18	Specify the width of bx input bus. For more information about supported input width, refer to the related information.
Enable 'bx' input register	bx_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for bx input register. For more information about clock enable restrictions for input registers, refer to the related information.
'by' input bus width	by_width	0–19	18	Specify the width of by input bus. For more information about supported input width, refer to the related information.
Enable 'by' input register	by_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for by input register. For more information about clock enable restrictions for input registers, refer to the related information.
3rd Multiplier Configuration				
Representation format for 'cx' operand	signed_mcx	**unsigned** **signed**	**unsigned**	Specify the representation format for third multiplier x operand. Only **m9x9_sumof4** operational mode supports this parameter.
Representation format for 'cy' operand	signed_mcy	**unsigned** **signed**	**unsigned**	Specify the representation format for third multiplier y operand. Only **m9x9_sumof4** operational mode supports this parameter.
'cx' input bus width	cx_width	0–9	0	Specify the width of cx input bus. Only **m9x9_sumof4** operational mode supports this parameter. For more information about supported input width, refer to the related information.
Enable 'cx' input register	cx_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for cx input register. Only **m9x9_sumof4** operational mode supports this parameter. For more information about clock enable restrictions for input registers, refer to the related information.
'cy' input bus width	cy_width	0–9	0	Specify the width of cy input bus. Only **m9x9_sumof4** operational mode supports this parameter. For more information about supported input width, refer to the related information.
Enable 'cy' input register	cy_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for cy input register. Only **m9x9_sumof4** operational mode supports this parameter.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				For more information about clock enable restrictions for input registers, refer to the related information.
4th Multiplier Configuration				
Representation format for 'dx' operand	signed_mdx	**unsigned** **signed**	**unsigned**	Specify the representation format for fourth multiplier x operand. Only **m9x9_sumof4** operational mode supports this parameter.
Representation format for 'dy' operand	signed_mdy	**unsigned** **signed**	**unsigned**	Specify the representation format for fourth multiplier y operand. Only **m9x9_sumof4** operational mode supports this parameter.
'dx' input bus width	dx_width	0–9	0	Specify the width of dx input bus. Only **m9x9_sumof4** operational mode supports this parameter. For more information about supported input width, refer to the related information.
Enable 'dx' input register	dx_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for dx input register. Only **m9x9_sumof4** operational mode supports this parameter. For more information about clock enable restrictions for input registers, refer to the related information.
'dy' input bus width	dy_width	0–9	0	Specify the width of dy input bus. Only **m9x9_sumof4** operational mode supports this parameter. For more information about supported input width, refer to the related information.
Enable 'dy' input register	dy_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for dy input register. Only **m9x9_sumof4** operational mode supports this parameter. For more information about clock enable restrictions for input registers, refer to the related information.
Sub Configuration				
Enable 'sub' port	enable_sub	**No** **Yes**	**No**	Select to enable sub port. The sub port is an input signal that can be used dynamically to subtract the output of the top multiplier from the output of the bottom multiplier. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** For more information about the sub port, refer to the related information.
Enable 'sub' input register	sub_clken	**no_reg** **ena0**	**no_reg**	Specify the clock enable signal for sub input register.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
		ena1 **ena2**		Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** For more information about clock enable restrictions for input registers, refer to the related information.
Output 'result' Configuration				
'resulta' output bus width	result_a_width	1–64	37	Specify the width of `resulta` output bus.
'resultb' output bus width	result_b_width	0–37	37	Specify the width of `resultb` output bus. Only available for **m18x18_full** operation mode.
Enable output register	output_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for `resulta` and `resultb` output register.

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.2. Input Cascade Tab

Table 47. Input Cascade Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Enable input cascade for 'ay' input	ay_use_scan_in	**No** **Yes**	**No**	Select to enable input cascade feature for first multiplier. When you enable input cascade, the multiplier uses `scanin` port instead of ay input bus as input data. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_systolic** • **m27×27**
Enable input cascade for 'by' input	by_use_scan_in	**No** **Yes**	**No**	Select to enable input cascade feature for second multiplier.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				When you enable input cascade, the multiplier uses ay input bus instead of by input bus as input data. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_systolic**
Enable 'disable_scanin'	disable_scanin	**No** **Yes**	**No**	Select to enable `disable_scanin` port. The `disable_scanin` port is an input signal that can be used dynamically to disable the input cascade feature for top multiplier by disabling the `scanin` input port. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_systolic** For more information about enabling and disabling `scanin` port dynamically, refer to the related information.
Scanout				
Enable data ay delay register	delay_scan_out_ay	**No** **Yes**	**No**	Select to enable delay register between ay and by input data.
Enable data by delay register	delay_scan_out_by	**No** **Yes**	**No**	Select to enable delay register between by and `scanout` input data.
Enable 'scanout' port	enable_scanout	**No** **Yes**	**No**	Select to enable `scanout` port. The `scanout` port is an output data bus of the input cascade module. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_systolic** • **m27×27**
'scanout' output bus width	scan_out_width	0–27	18	Specify the width of `scanout` output bus. Only available for the following operation modes: • **m18×18_full** • **m18×18_sumof2** • **m18×18_systolic** • **m27×27**

Related Information

- DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
  The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.
- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82

- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Dynamic Scanin on page 69
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.3. Pre-adder Tab

Table 48. Pre-adder Tab

These parameters are only available in **m18×18_full**, **m18×18_sumof2**, **m18×18_systolic**, and, **m27×27** operational modes.

Parameter	IP Generated Parameter	Value	Default Value	Description
'ay' operand source	operand_source_may	**Input** **Preadder**	**Input**	Select the operand source for ay input bus. To enable pre-adder block, select **Preadder**.
'by' operand source	operand_source_mby	**Input** **Preadder**	**Input**	Select the operand source for by input bus. To enable pre-adder block, select **Preadder**.
Set top pre-adder operation to subtraction	preadder_subtract_a	**No** **Yes**	**No**	Specify the operation for top pre-adder. Select **Yes** to use top pre-adder as a subtractor. Select **No** to use top pre-adder as an adder.
Set bottom pre-adder operation to subtraction	preadder_subtract_b	**No** **Yes**	**No**	Specify the operation for bottom pre-adder. Select **Yes** to use bottom pre-adder as a subtractor. Select **No** to use bottom pre-adder as an adder.
Data 'z' Configuration				
'az' input bus width	az_width	0–26	0	Specify the width of az input bus.
Enable 'az' input register	az_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for az input register.
'bz' input bus width	bz_width	0–18	0	Specify the width of bz input bus.
Enable 'bz' input register	bz_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for bz input register.

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.4. Internal Coefficient Tab

Table 49. Internal Coefficient Configuration

These parameters are only available in **m18×18_full**, **m18×18_sumof2**, **m18×18_systolic**, and **m27×27** operational modes.

Parameter	IP Generated Parameter	Value	Default Value	Description
'ax' operand source	operand_source_max	**input** **coef**	**input**	Specify the operand source for ax input bus. Select **coef** to use ax input bus to provide constant coefficients to the top multiplier.
'bx' operand source	operand_source_mbx	**input** **coef**	**input**	Specify the operand source for bx input bus. Select **coef** to use ax input bus to provide constant coefficients to the bottom multiplier.
'coefsel' Input Register Configuration				
Enable 'coefsela' input register	coef_sel_a_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `coefsela` input register.
Enable 'coefselb' input register	coef_sel_b_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `coefselb` input register.
Coefficient Storage Configuration				
coef_a_0	coef_a_0	Integer	0	Specify the coefficient values for ax input bus. For 18-bit operation mode, the maximum input value is 2^{18} - 1. For 27-bit operation, the maximum value is 2^{27} - 1.
coef_a_1	coef_a_1			
coef_a_2	coef_a_2			
coef_a_3	coef_a_3			
coef_a_4	coef_a_4			
coef_a_5	coef_a_5			
coef_a_6	coef_a_6			
coef_a_7	coef_a_7			
coef_b_0	coef_a_0	Integer	0	Specify the coefficient values for bx input bus. Set coefficient values to more than 67108864 when operand is set to **unsigned** and `negate` is enabled. These parameters are not available in **m27×27** operational mode.
coef_b_1	coef_a_1			
coef_b_2	coef_a_2			
coef_b_3	coef_a_3			
coef_b_4	coef_a_4			
coef_b_5	coef_a_5			
coef_b_6	coef_a_6			
coef_b_7	coef_a_7			

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.5. Accumulator/Output Chaining

Table 50. Accumulator/Output Chaining Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Accumulator				
Enable accumulate port	enable_accumulate	**No** **Yes**	**No**	Select to enable `accumulate` port. Only available for the following operational modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Enable 'accumulate' input register	accumulate_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `accumulate` input register. Only available for the following operational modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27** For more information about clock enable restrictions for input registers, refer to the related information.
Enable double accumulator	enable_double_accum	**No** **Yes**	**No**	Select to enable the double accumulator feature. Only available for the following operational modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Negate				
Enable 'negate' port	enable_negate	**No** **Yes**	**No**	Select to enable `negate` port. Only available for the following operational modes: • **m18×18_sumof2** • **m18×18_systolic** • **m27×27**

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
Enable 'negate' input register	negate_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `negate` input register. Only available for the following operational modes: • **m18×18_sumof2** • **m18×18_systolic** • **m27×27** For more information about clock enable restrictions for input registers, refer to the related information.
Loadconst				
Enable 'loadconst' port	enable_loadconst	**No** **Yes**	**No**	Select to enable `loadconst` port. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Enable 'loadconst' input register	load_const_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `loadconst` input register. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27** For more information about clock enable restrictions for input registers, refer to the related information.
N value of preset constant	load_const_value	0–63	0	Specify the preset constant value. This value can be 2^N where N is the preset constant value. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Chainin/Chainout				
Enable chainin port	use_chainadder	**No** **Yes**	**No**	Select to enable `chainin` port. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Enable chainout port	enable_chainout	**No** **Yes**	**No**	Select to enable `chainout` port. Only available for the following operation modes:

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
				• **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Enable disable_chainout	disable_chainout	**No** **Yes**	**No**	Select to enable `disable_chainout` port. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**
Set the chainin and chainout width	chain_inout_width	**0** **44** **64**	**0**	Specify the width of `chainin` and `chainout` buses. Only available for the following operation modes: • **m9×9_sumof4** • **m18×18_sumof2** • **m18×18_plus36** • **m18×18_systolic** • **m27×27**

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.6. Pipelining

Table 51. Pipelining Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Input Pipeline Register				
Enable input pipeline register to the input data signal (x/y/z/ coefsel)	input_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the first pipeline register clock enable signal for `x`, `y`, `z`, and `coefsel` ports. Select **no_reg** to disable the register.
Enable 2nd input pipeline register to the input data signal (x/y/z/ coefsel)	second_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the second pipeline register clock enable signal for `x`, `y`, `z`, and `coefsel` ports. Select **no_reg** to disable the register.
Accumulator Pipeline Register				

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
Enable 'accumulate' input pipeline register	accum_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the first pipeline register clock enable signal for `accumulate` port. Select **no_reg** to disable the register.
Enable 'accumulate' 2nd input pipeline register	accum_2nd_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the first pipeline register clock enable signal for `accumulate` port. Select **no_reg** to disable the register.
Loadconst Pipeline Registers				
Enable 'loadconst' input pipeline register	load_const_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the first pipeline register clock enable signal for `loadconst` port. Select **no_reg** to disable the register.
Enable 'loadconst' 2nd input pipeline register	load_const_2nd_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the second pipeline register clock enable signal for `loadconst` port. Select **no_reg** to disable the register.
Systolic Configuration				
Enable input systolic register	input_systolic_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for the input systolic register. Select **no_reg** to disable the register.

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.5.7. Clear Signal

Table 52. Clear Signal Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Type of clear signal	clear_type	**none** **aclr** **sclr**	**none**	Specify the clear signal behavior for all registers in the fixed-point DSP block • **none**: Select to not use any clear signal. • **aclr**: Select to use asynchronous clear signal type for all registers. • **sclr**: Select to use synchronous clear signal type for all registers.
Enable clr0 for all input registers	enable_clr0	**No** **Yes**	**No**	Select **Yes** to enable `clr[0]` signal for all input registers.
Enable clr1 for output and pipeline registers	enable_clr1	**No** **Yes**	**No**	Select **Yes** to enable `clr[1]` signal for output and pipeline registers

Related Information

- Maximum Input Data Width for Fixed-point Arithmetic on page 80
  Provides more information about supported input width.
- Maximum Output Data Width for Fixed-point Arithmetic on page 82
- Native Fixed Point DSP Intel Agilex FPGA IP Signals on page 94
  Provides more information about the sub port.
- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.

5.6. Native Fixed Point DSP Intel Agilex FPGA IP Signals

The following are the input and output signals of the Native Fixed Point DSP Intel Agilex FPGA IP for each operational mode.

5.6.1. 9 × 9 Sum of 4 Mode Signals

Figure 51. 9 × 9 Sum of 4 Mode Signals

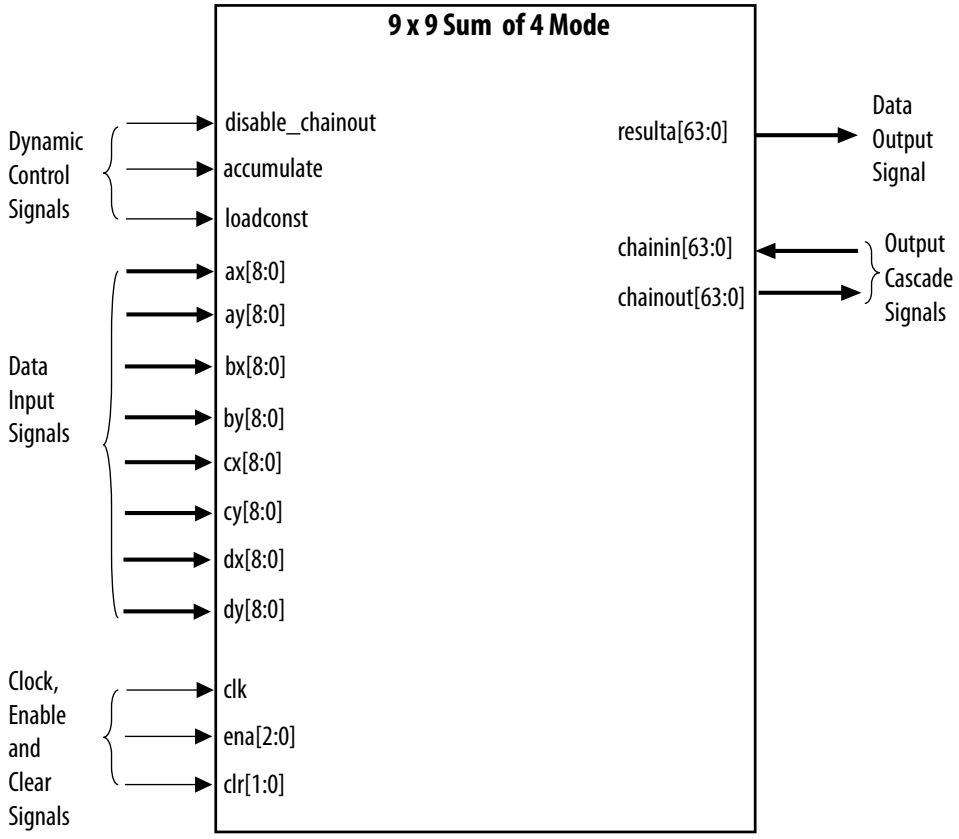


Table 53. Input and Output Data Signals

Signal Name	Type	Width	Description
ax[8:0]	Input	9	Input data bus to first multiplier.
ay[8:0]	Input	9	Input data bus to first multiplier. When pre-adder is enabled, these signals are served as input to the top pre-adder.
bx[8:0]	Input	9	Input data bus to second multiplier.
by[17:0]	Input	9	Input data bus to second multiplier. When pre-adder is enabled, these signals are served as input to the bottom pre-adder.
cx[8:0]	Input	9	Input data bus to third multiplier.
cy[8:0]	Input	9	
dx[8:0]	Input	9	Input data bus to third multiplier
dy[8:0]	Input	9	
resulta[63:0]	Output	64	Output data bus.

Table 54. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 55. Dynamic Control Signals

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_chainout	Input	1	Dynamic input signal to enable dynamic chainout feature. You can change the value of this signal during run-time. You must connect the chainout output bus to the next DSP block in order to use this signal. • 0: Send the chainout output to the next DSP block. Default value. • 1: Do not send the chainout output to the next DSP block. The chainout output is all zero.
accumulate	Input	1	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 0: Generate the current result without accumulating the previous result. Default value. • 1: Add the current result to the previous result.
loadconst	Input	1	Input signal to enable or disable the load constant feature. You can change the value of this signal during run-time. • 0: Disable the load constant feature. Default value. • 1: Add a preload constant to the result to perform a biased rounding.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.6.2. 18 × 18 Full Mode Signals

Figure 52. 18 × 18 Full Mode Signals

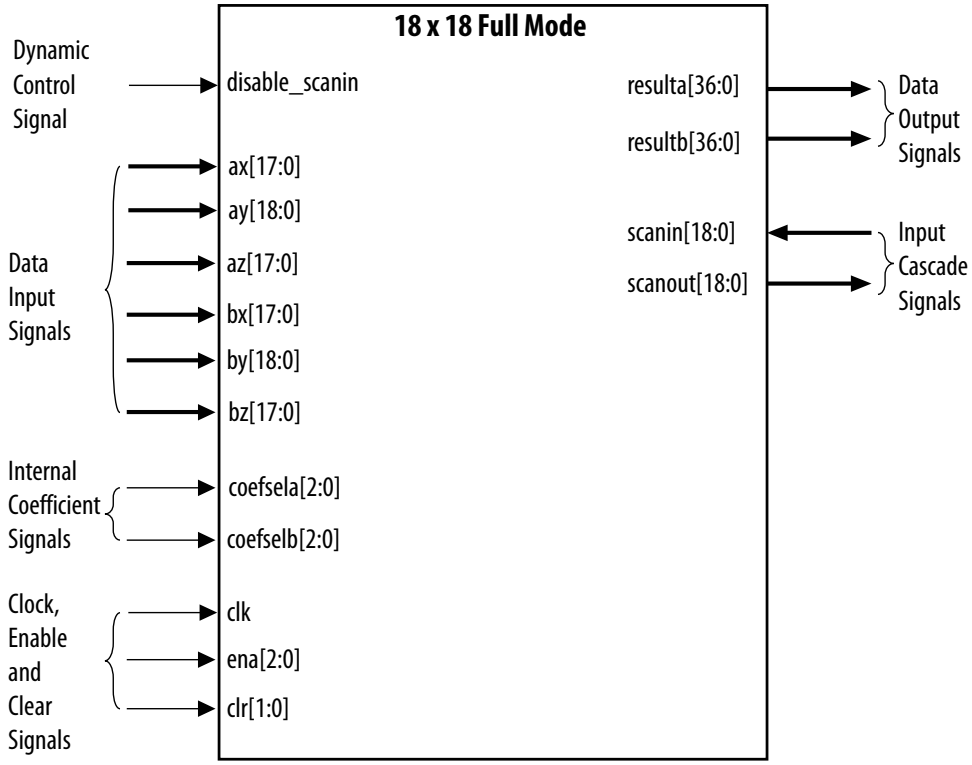


Table 56. Data Input and Output Signals

Signal Name	Type	Width	Description
ax[17:0]	Input	18	Input data bus to top multiplier. This signal is not available when internal coefficient feature is enabled.
ay[18:0]	Input	19	Input data bus to top multiplier. When pre-adder is enabled, these signals are served as input to the top pre-adder.
az[17:0]	Input	18	These signal are input to the top pre-adder. These signals are only available when pre-adder is enabled.
bx[17:0]	Input	18	Input data bus to bottom multiplier.
by[18:0]	Input	19	Input data bus to bottom multiplier. When pre-adder is enabled, these signals serve as input signals to the bottom pre-adder.
bz[17:0]	Input	18	These signals are input signals to the bottom pre-adder. These signals are only available when pre-adder is enabled.
resulta[36:0]	Output	37	Output data bus from top multiplier.
resultb[36:0]	Output	37	Output data bus from bottom multiplier.

Table 57. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 58. Dynamic Control Signal

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_scanin	Input	1	Dynamic input signal to enable dynamic scanin feature. You can change the value of this signal during run-time. This signal is available when you Set **Enable 'disable scanin** parameter to **Yes**. You must set **Enable input cascade for 'ay' input** parameter to **Yes** to use this signal. • 0: Switch the input of the top multiplier to use scanin input. • 1: Switch the input of the top multiplier to use ay input.

Table 59. Internal Coefficient Ports

For a summary of supported features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
coefsela[2:0]	Input	3	Input selection signals for 8 coefficient values defined by user for the top multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_a_0** to **coef_a_7**. • coefsela[2:0] = 000 refers to **coef_a_0** • coefsela[2:0] = 001 refers to **coef_a_1** • coelsela[2:0] = 010 refers to **coef_a_2** and so forth. These signals are only available when the internal coefficient feature is enabled.
coefselb[2:0]	Input	3	Input selection signals for 8 coefficient values defined by user for the bottom multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_b_0** to **coef_b_7**. • coefselb[2:0] = 000 refers to **coef_b_0** • coefselb[2:0] = 001 refers to **coef_b_1** • coelselb[2:0] = 010 refers to **coef_b_2** and so forth. These signals are only available when the internal coefficient feature is enabled.

Table 60. Input Cascade Signals

Signal Name	Type	Width	Description
scanin[26:0]	Input	27	Input data bus for input cascade module.

continued...

Signal Name	Type	Width	Description
			Connect these signals to the scanout signals from the preceding DSP core.
scanout[26:0]	Output	27	Output data bus of the input cascade module. Connect these signals to the scanin signals of the next DSP core.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.6.3. 18 × 18 Sum of Two Mode Signals

Figure 53. 18 × 18 Sum of Two Mode Signals

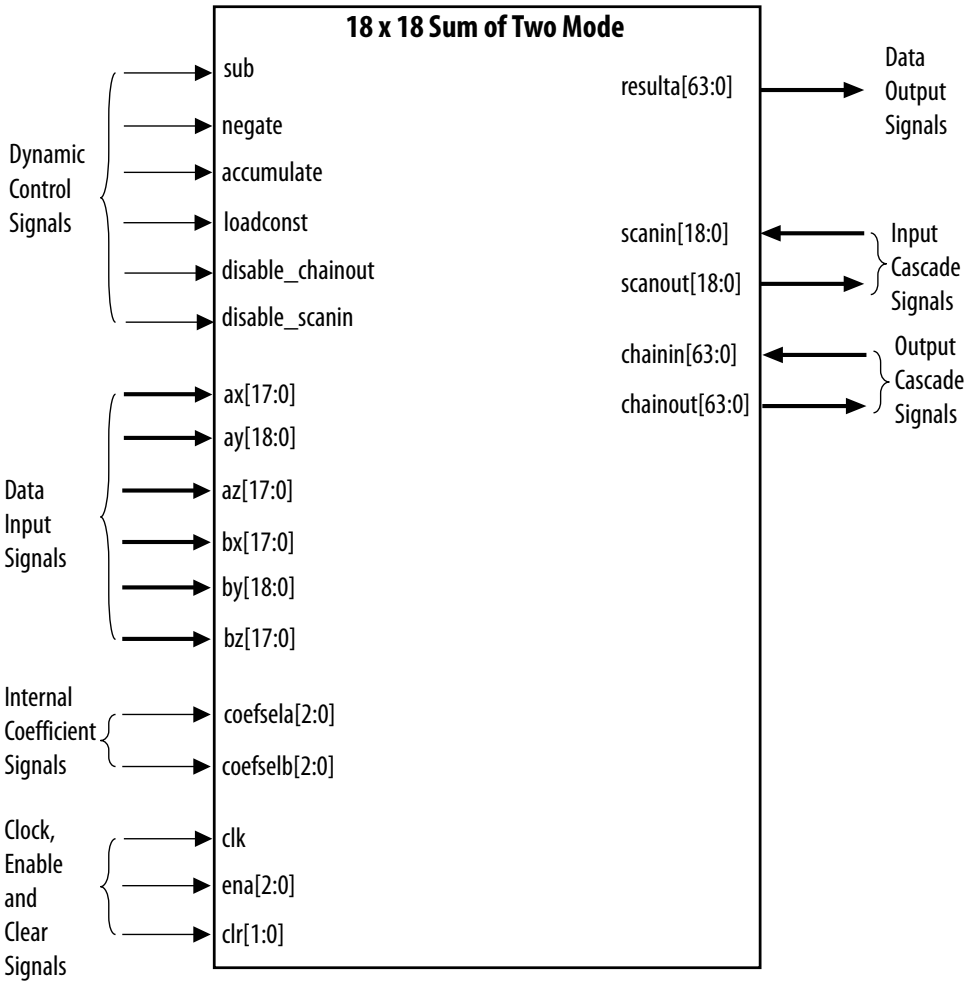


Table 61. Data Input and Output Signals

Signal Name	Type	Width	Description
ax[17:0]	Input	18	Input data bus to top multiplier. This signal is not available when internal coefficient feature is enabled.
ay[18:0]	Input	19	Input data bus to top multiplier. When pre-adder is enabled, these signals are served as input to the top pre-adder.
az[17:0]	Input	18	These signal are input to the top pre-adder. These signals are only available when pre-adder is enabled.
bx[17:0]	Input	18	Input data bus to bottom multiplier.
by[18:0]	Input	19	Input data bus to bottom multiplier. When pre-adder is enabled, these signals serve as input signals to the bottom pre-adder.
bz[17:0]	Input	18	These signals are input signals to the bottom pre-adder. These signals are only available when pre-adder is enabled.
resulta[63:0]	Output	37	Output data bus from top multiplier.

Table 62. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 63. Dynamic Control Signals

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_chainout	Input	1	Dynamic input signal to enable dynamic chainout feature. You can change the value of this signal during run-time. You must connect the chainout output bus to the next DSP block in order to use this signal. • 0: Send the chainout output to the next DSP block. Default value. • 1: Do not send the chainout output to the next DSP block. The chainout output is all zero.
disable_scanin	Input	1	Dynamic input signal to enable dynamic scanin feature. You can change the value of this signal during run-time. This signal is available when you Set **Enable 'disable scanin** parameter to **Yes**.

continued...

Signal Name	Type	Width	Description
			You must set **Enable input cascade for 'ay' input** parameter to **Yes** to use this signal. • 0: Switch the input of the top multiplier to use `scanin` input. • 1: Switch the input of the top multiplier to use `ay` input.
`accumulate`	Input	1	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 0: Generate the current result without accumulating the previous result. Default value. • 1: Add the current result to the previous result.
`loadconst`	Input	1	Input signal to enable or disable the load constant feature. You can change the value of this signal during run-time. • 0: Disable the load constant feature. Default value. • 1: Add a preload constant to the result to perform a biased rounding.
`sub`	Input	1	Dynamic input signal to control the operation of the adder module. You can change the value of this signal during run-time. • 0: Add the output of the top multiplier with the output of the bottom multiplier. Default value. • 1: Subtract the output of the top multiplier from the output of the bottom multiplier.
`negate`	Input	1	Dynamic input signal to control the operation of the chainout adder module. You can change the value of this signal during run-time. • 0: Add the sum of the top and bottom multipliers with the chainin data input bus and accumulate loopback data. Default value. • 1: Subtract the sum of the top and bottom multipliers from the chainin data input bus and accumulate loopback data.

Table 64. Internal Coefficient Ports

For a summary of supported features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
`coefsela[2:0]`	Input	3	Input selection signals for 8 coefficient values defined by user for the top multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_a_0** to **coef_a_7**. • coefsela[2:0] = 000 refers to **coef_a_0** • coefsela[2:0] = 001 refers to **coef_a_1** • coelsela[2:0] = 010 refers to **coef_a_2** and so forth. These signals are only available when the internal coefficient feature is enabled.
`coefselb[2:0]`	Input	3	Input selection signals for 8 coefficient values defined by user for the bottom multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_b_0** to **coef_b_7**. • coefselb[2:0] = 000 refers to **coef_b_0** • coefselb[2:0] = 001 refers to **coef_b_1** • coelselb[2:0] = 010 refers to **coef_b_2** and so forth. These signals are only available when the internal coefficient feature is enabled.

Table 65. Input Cascade Signals

Signal Name	Type	Width	Description
`scanin[18:0]`	Input	19	Input data bus for input cascade module.

continued...

Signal Name	Type	Width	Description
			Connect these signals to the scanout signals from the preceding DSP core.
scanout[18:0]	Output	19	Output data bus of the input cascade module. Connect these signals to the scanin signals of the next DSP core.

Table 66. Output Cascade Signals

Signal Name	Type	Width	Description
chainin[63:0]	Input	64	Input data bus for output cascade module. Connect these signals to the chainout signals from the preceding DSP core.
chainout[63:0]	Output	64	Output data bus of the output cascade module. Connect these signals to the chainin signals of the next DSP core.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.6.4. 18 × 18 Plus 36 Mode Signals

Figure 54. 18 × 18 Plus 36 Mode Signals

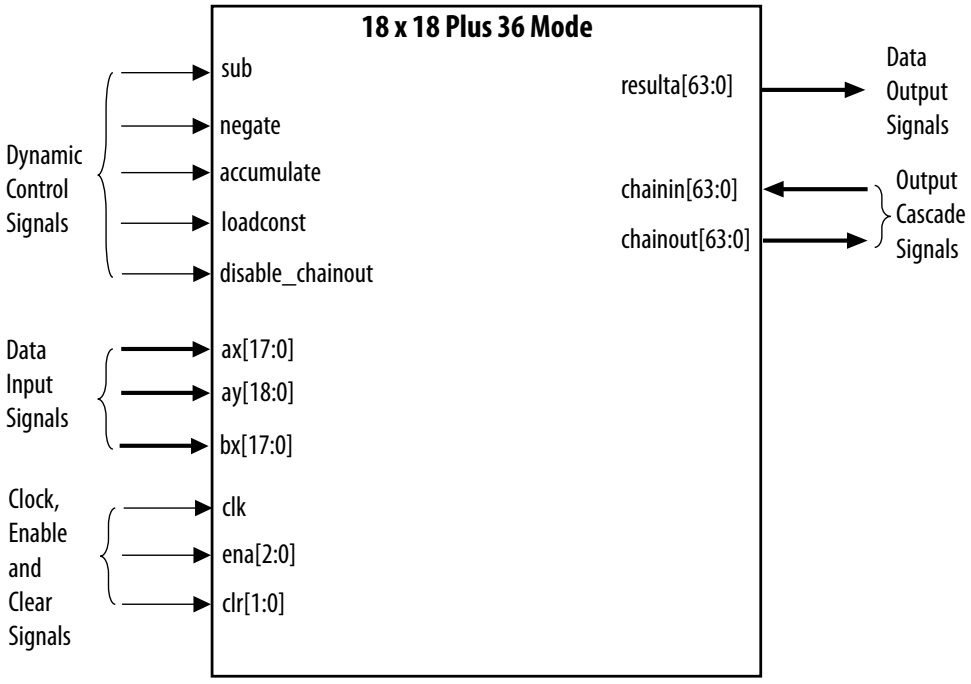


Table 67. Data Input and Output Signals

Signal Name	Type	Width	Description
ax[17:0]	Input	18	Input data bus to top multiplier. This signal is not available when internal coefficient feature is enabled.
ay[18:0]	Input	19	Input data bus to top multiplier. When pre-adder is enabled, these signals are served as input to the top pre-adder.
bx[17:0]	Input	18	Input data bus to bottom multiplier.
resulta[63:0]	Output	37	Output data bus from top multiplier.

Table 68. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 69. Dynamic Control Signals

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_chainout	Input	1	Dynamic input signal to enable dynamic chainout feature. You can change the value of this signal during run-time. You must connect the chainout output bus to the next DSP block in order to use this signal. • 0: Send the chainout output to the next DSP block. Default value. • 1: Do not send the chainout output to the next DSP block. The chainout output is all zero.
accumulate	Input	1	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 0: Generate the current result without accumulating the previous result. Default value. • 1: Add the current result to the previous result.
loadconst	Input	1	Input signal to enable or disable the load constant feature. You can change the value of this signal during run-time. • 0: Disable the load constant feature. Default value. • 1: Add a preload constant to the result to perform a biased rounding.
sub	Input	1	Dynamic input signal to control the operation of the adder module. You can change the value of this signal during run-time.

continued...

Signal Name	Type	Width	Description
			• 0: Add the output of the top multiplier with the output of the bottom multiplier. Default value. • 1: Subtract the output of the top multiplier from the output of the bottom multiplier.
negate	Input	1	Dynamic input signal to control the operation of the chainout adder module. You can change the value of this signal during run-time. • 0: Add the sum of the top and bottom multipliers with the chainin data input bus and accumulate loopback data. Default value. • 1: Subtract the sum of the top and bottom multipliers from the chainin data input bus and accumulate loopback data.

Table 70. Output Cascade Signals

Signal Name	Type	Width	Description
chainin[63:0]	Input	64	Input data bus for output cascade module. Connect these signals to the chainout signals from the preceding DSP core.
chainout[63:0]	Output	64	Output data bus of the output cascade module. Connect these signals to the chainin signals of the next DSP core.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.6.5. 18 × 18 Systolic Mode Signals

Figure 55. 18 × 18 Systolic Mode Signals

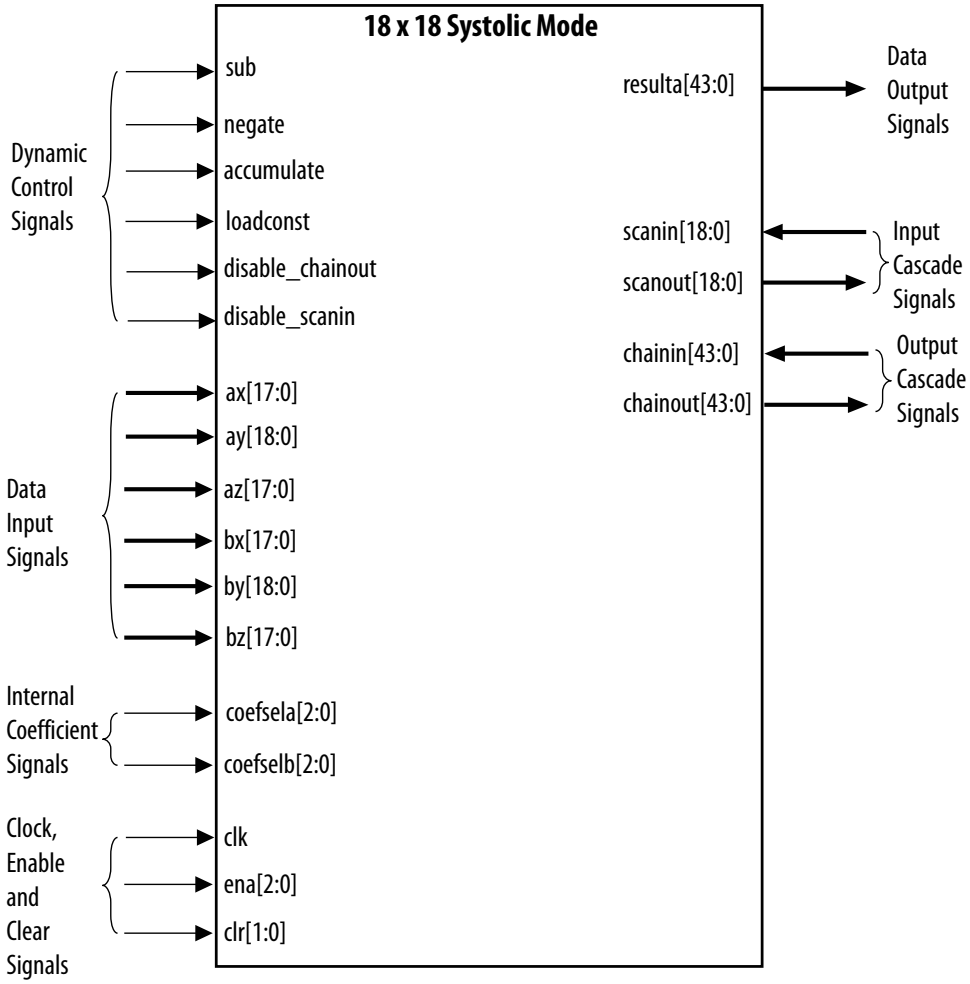


Table 71. Data Input and Output Signals

Signal Name	Type	Width	Description
ax[17:0]	Input	18	Input data bus to top multiplier. This signal is not available when internal coefficient feature is enabled.
ay[18:0]	Input	19	Input data bus to top multiplier. When pre-adder is enabled, these signals are served as input to the top pre-adder.
az[17:0]	Input	18	These signal are input to the top pre-adder. These signals are only available when pre-adder is enabled.
bx[17:0]	Input	18	Input data bus to bottom multiplier.
by[18:0]	Input	19	Input data bus to bottom multiplier.

continued...

Signal Name	Type	Width	Description
			When pre-adder is enabled, these signals serve as input signals to the bottom pre-adder.
bz[17:0]	Input	18	These signals are input signals to the bottom pre-adder. These signals are only available when pre-adder is enabled.
resulta[43:0]	Output	44	Output data bus from top multiplier.

Table 72. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 73. Dynamic Control Signals

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_chainout	Input	1	Dynamic input signal to enable dynamic chainout feature. You can change the value of this signal during run-time. You must connect the chainout output bus to the next DSP block in order to use this signal. • 0: Send the chainout output to the next DSP block. Default value. • 1: Do not send the chainout output to the next DSP block. The chainout output is all zero.
disable_scanin	Input	1	Dynamic input signal to enable dynamic scanin feature. You can change the value of this signal during run-time. This signal is available when you Set **Enable 'disable scanin** parameter to **Yes**. You must set **Enable input cascade for 'ay' input** parameter to **Yes** to use this signal. • 0: Switch the input of the top multiplier to use scanin input. • 1: Switch the input of the top multiplier to use ay input.
accumulate	Input	1	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 0: Generate the current result without accumulating the previous result. Default value. • 1: Add the current result to the previous result.
loadconst	Input	1	Input signal to enable or disable the load constant feature. You can change the value of this signal during run-time.

continued...

Signal Name	Type	Width	Description
			• 0: Disable the load constant feature. Default value. • 1: Add a preload constant to the result to perform a biased rounding.
sub	Input	1	Dynamic input signal to control the operation of the adder module. You can change the value of this signal during run-time. • 0: Add the output of the top multiplier with the output of the bottom multiplier. Default value. • 1: Subtract the output of the top multiplier from the output of the bottom multiplier.
negate	Input	1	Dynamic input signal to control the operation of the chainout adder module. You can change the value of this signal during run-time. • 0: Add the sum of the top and bottom multipliers with the chainin data input bus and accumulate loopback data. Default value. • 1: Subtract the sum of the top and bottom multipliers from the chainin data input bus and accumulate loopback data.

Table 74. Internal Coefficient Ports

For a summary of supported features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
coefsela[2:0]	Input	3	Input selection signals for 8 coefficient values defined by user for the top multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_a_0** to **coef_a_7**. • coefsela[2:0] = 000 refers to **coef_a_0** • coefsela[2:0] = 001 refers to **coef_a_1** • coelsela[2:0] = 010 refers to **coef_a_2** and so forth. These signals are only available when the internal coefficient feature is enabled.
coefselb[2:0]	Input	3	Input selection signals for 8 coefficient values defined by user for the bottom multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_b_0** to **coef_b_7**. • coefselb[2:0] = 000 refers to **coef_b_0** • coefselb[2:0] = 001 refers to **coef_b_1** • coelselb[2:0] = 010 refers to **coef_b_2** and so forth. These signals are only available when the internal coefficient feature is enabled.

Table 75. Input Cascade Signals

Signal Name	Type	Width	Description
scanin[18:0]	Input	19	Input data bus for input cascade module. Connect these signals to the scanout signals from the preceding DSP core.
scanout[18:0]	Output	19	Output data bus of the input cascade module. Connect these signals to the scanin signals of the next DSP core.

Table 76. Output Cascade Signals

Signal Name	Type	Width	Description
chainin[43:0]	Input	44	Input data bus for output cascade module.

continued...

Signal Name	Type	Width	Description
			Connect these signals to the chainout signals from the preceding DSP core.
chainout[43:0]	Output	44	Output data bus of the output cascade module. Connect these signals to the chainin signals of the next DSP core.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.6.6. 27 × 27 Mode Signals

Figure 56. 27 × 27 Mode Signals

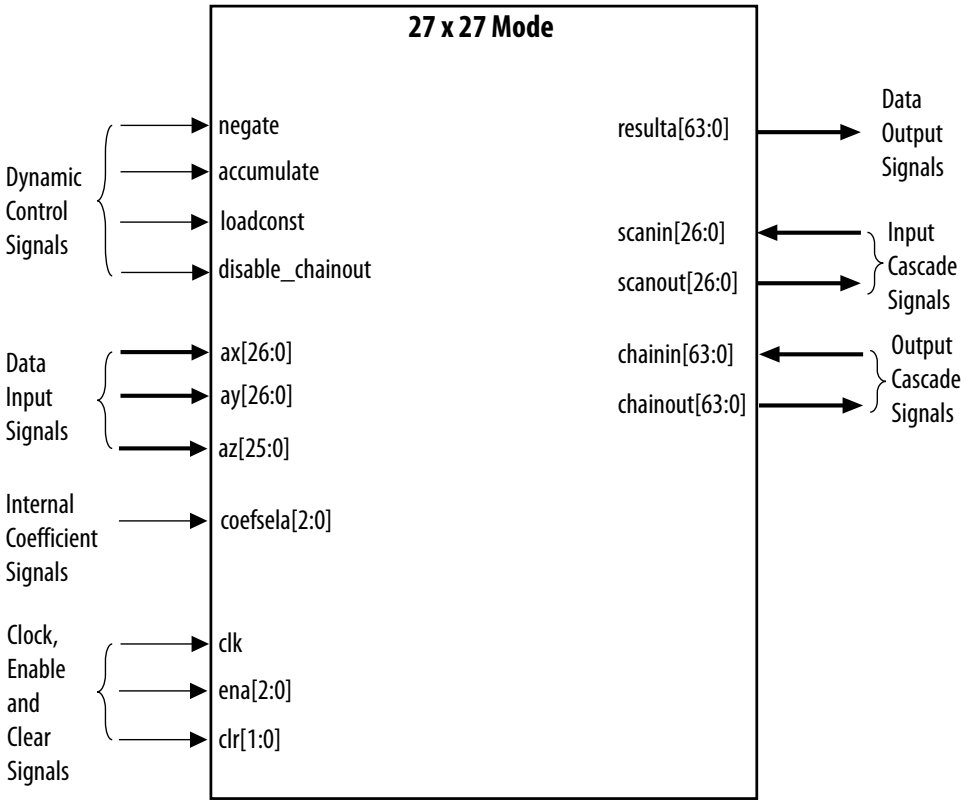


Table 77. Data Input and Output Signals

Signal Name	Type	Width	Description
ax[26:0]	Input	27	Input data bus to the multiplier. This signal is not available when internal coefficient feature is enabled.
ay[26:0]	Input	27	Input data bus to the multiplier. When pre-adder is enabled, these signals are served as input to the pre-adder.
az[25:0]	Input	26	These signal are input to the pre-adder. These signals are only available when pre-adder is enabled.
resulta[63:0]	Output	64	Output data bus from the multiplier.

Table 78. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Description
clk[0]	Input	1	Input clock for all registers.
ena[2:0]	Input	3	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. By default, this signal is low. For more information about clock enable restrictions for input registers, refer to the related information.

Table 79. Dynamic Control Signals

For a summary of supported dynamic control features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
disable_chainout	Input	1	Dynamic input signal to enable dynamic chainout feature. You can change the value of this signal during run-time. You must connect the chainout output bus to the next DSP block in order to use this signal. • 0: Send the chainout output to the next DSP block. Default value. • 1: Do not send the chainout output to the next DSP block. The chainout output is all zero.
accumulate	Input	1	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 0: Generate the current result without accumulating the previous result. Default value. • 1: Add the current result to the previous result.
loadconst	Input	1	Input signal to enable or disable the load constant feature. You can change the value of this signal during run-time. • 0: Disable the load constant feature. Default value. • 1: Add a preload constant to the result to perform a biased rounding.
negate	Input	1	Dynamic input signal to control the operation of the chainout adder module. You can change the value of this signal during run-time.

continued...

Signal Name	Type	Width	Description
			• 0: Add the sum of the top and bottom multipliers with the chainin data input bus and accumulate loopback data. Default value. • 1: Subtract the sum of the top and bottom multipliers from the chainin data input bus and accumulate loopback data.

Table 80. Internal Coefficient Ports

For a summary of supported features for each operational mode, refer to the related information.

Signal Name	Type	Width	Description
coefsela[2:0]	Input	3	Input selection signals for 8 coefficient values defined by user for the top multiplier. The coefficient values are stored in the internal memory and specified by parameters **coef_a_0** to **coef_a_7**. • coefsela[2:0] = 000 refers to **coef_a_0** • coefsela[2:0] = 001 refers to **coef_a_1** • coelsela[2:0] = 010 refers to **coef_a_2** and so forth. These signals are only available when the internal coefficient feature is enabled.

Table 81. Input Cascade Signals

Signal Name	Type	Width	Description
scanin[26:0]	Input	27	Input data bus for input cascade module. Connect these signals to the scanout signals from the preceding DSP core.
scanout[26:0]	Output	27	Output data bus of the input cascade module. Connect these signals to the scanin signals of the next DSP core.

Table 82. Output Cascade Signals

Signal Name	Type	Width	Description
chainin[63:0]	Input	64	Input data bus for output cascade module. Connect these signals to the chainout signals from the preceding DSP core.
chainout[63:0]	Output	64	Output data bus of the output cascade module. Connect these signals to the chainin signals of the next DSP core.

Related Information

- Configurations for Input, Pipeline, and Output Registers on page 64
  Provides more information about clock enable restrictions for input registers.
- Fixed-point Arithmetic on page 6
  Provides a summary of supported features and dynamic control features for each operational mode.

5.7. IP Migration

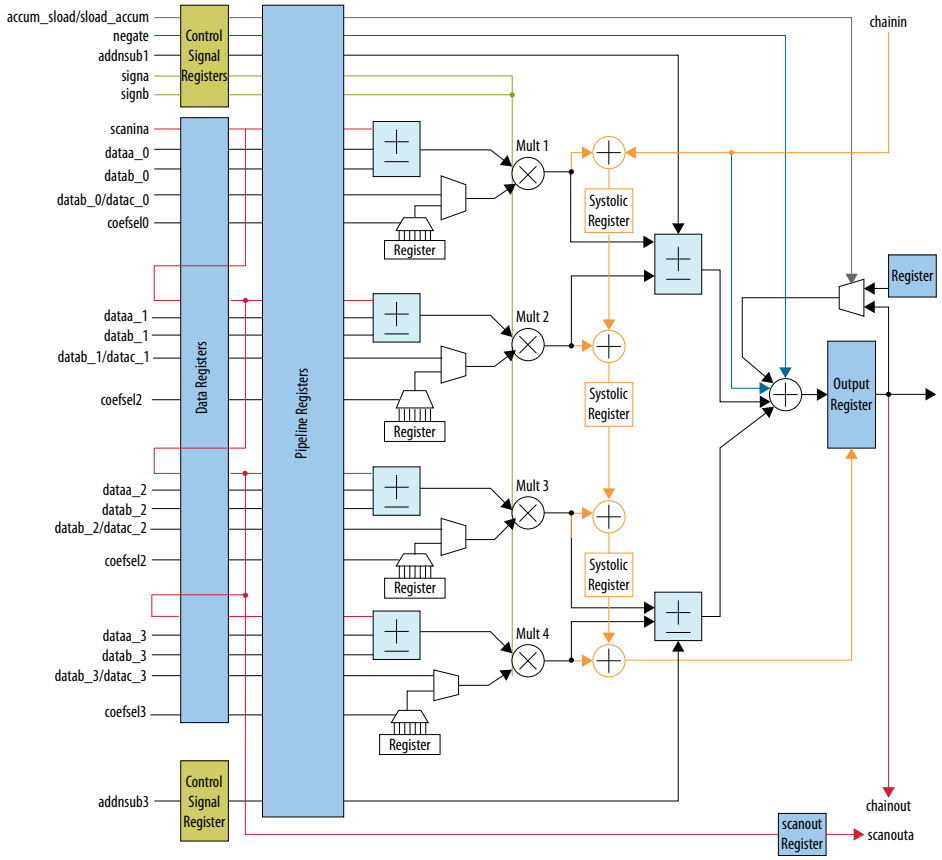
The Native Fixed Point DSP Intel Agilex Intel FPGA IP does not support family migration.

6. Multiply Adder Intel FPGA IP Core References

The Multiply Adder Intel FPGA IP core allows you to implement a multiplier-adder. This is a family independent IP.

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

Figure 57. Multiply Adder Intel FPGA IP 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 DSP block uses 18 × 19-bit input multipliers to process data with widths up to 18 bits and 27 × 27 bit input multipliers to process data with widths between 18 to 27 bits. For data with widths more than 27 bits, the DSP block uses partial products algorithm to process the data and 27 × 27-bit input multiplier to process data with widths between 18 to 27 bits.

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

- Data input
- Signed or unsigned select
- Add or subtract select
- 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.

6.1. Multiply Adder Intel FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 83. Multiply Adder Intel FPGA IP Release Information

Item	Description
IP Version	19.2.0
Intel Quartus Prime	21.2
Release Date	2021.06.23

6.2. Features

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

- Generates a multiplier to perform multiplication operations of two numbers

  Note: When building multipliers larger than the natively supported size there may/ will be a performance impact resulting from the partial production implementation.
- 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

6.2.1. Pre-adder

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

There are five pre-adder modes:

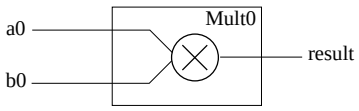
- Simple mode
- Coefficient mode
- Input mode
- Square mode
- Constant mode

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

6.2.1.1. 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 58. Pre-adder Simple Mode



6.2.1.2. 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]`.

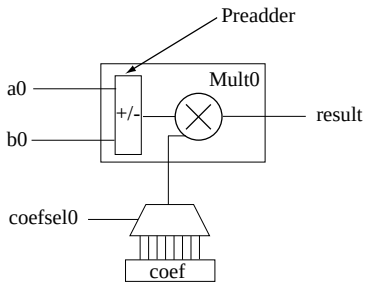
This mode is expressed in the following equation.

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

where $k =$ number of multipliers

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

Figure 59. Pre-adder Coefficient Mode



6.2.1.3. 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.

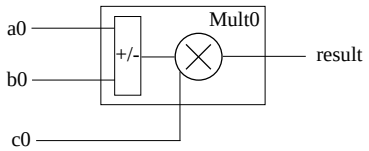
This mode is expressed in the following equation.

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

where $k =$ number of multipliers

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

Figure 60. Pre-adder Input Mode



6.2.1.4. Pre-adder Square Mode

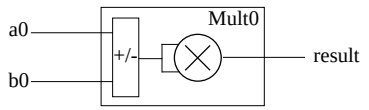
This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} (a_n + b_n)^2$$

where $k =$ number of multipliers

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

Figure 61. Pre-adder Square Mode



6.2.1.5. 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]`.

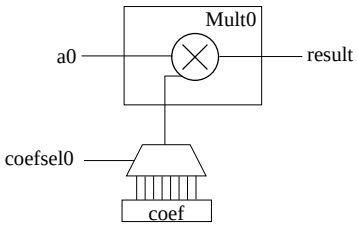
This mode is expressed in the following equation.

$$result = \sum_{n=0}^{k-1} a_0 \times coef$$

where $k =$ number of multipliers

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

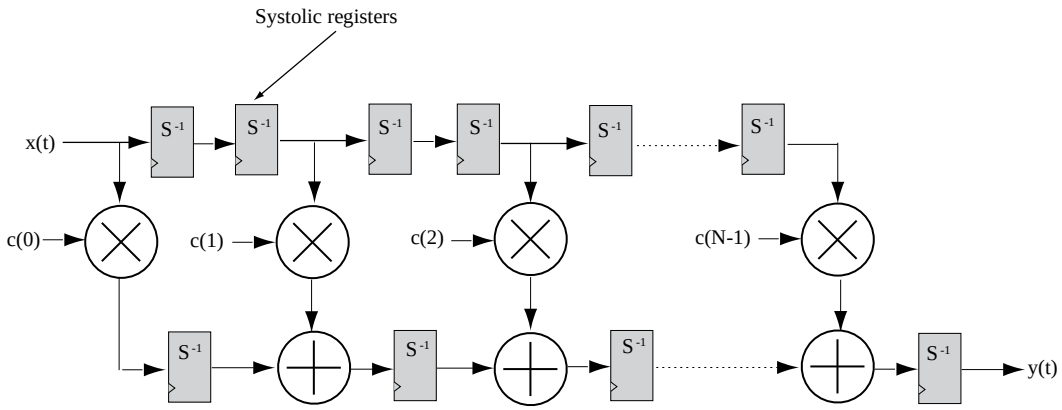
Figure 62. Pre-adder Constant Mode



6.2.2. 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 63. 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 sync. This processing element is replicated to form a circuit that computes the filtering function. This function is expressed in the following equation.

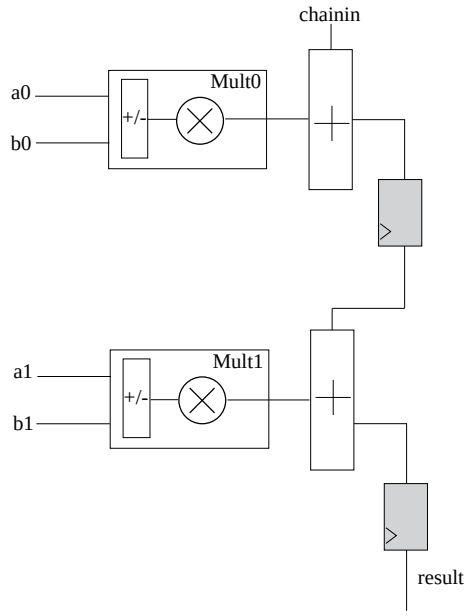
$$y(t) = \sum_{i=0}^{N-1} B(i)A(t-i)$$

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.

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

Figure 64. Systolic Delay Register Implementation of 2 Multipliers

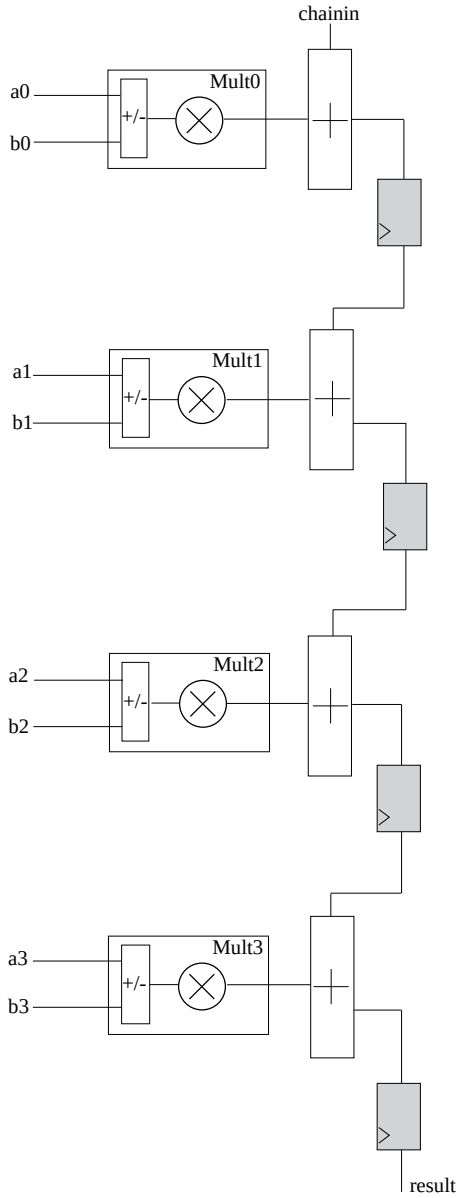


The sum of two multipliers is expressed in the following equation.

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

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

Figure 65. Systolic Delay Register Implementation of 4 Multipliers



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

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

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

Related Information

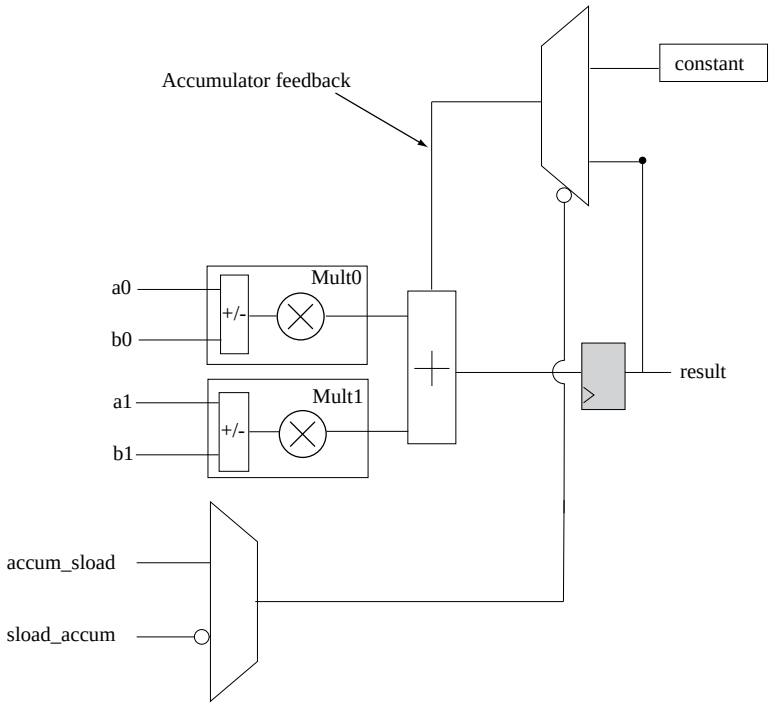
DSP Block Cascade Limit in Intel Agilex 7 Devices on page 76
The sector size limits the number of DSP blocks you can cascade in Intel Agilex 7 devices.

6.2.3. 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 66. Pre-load Constant

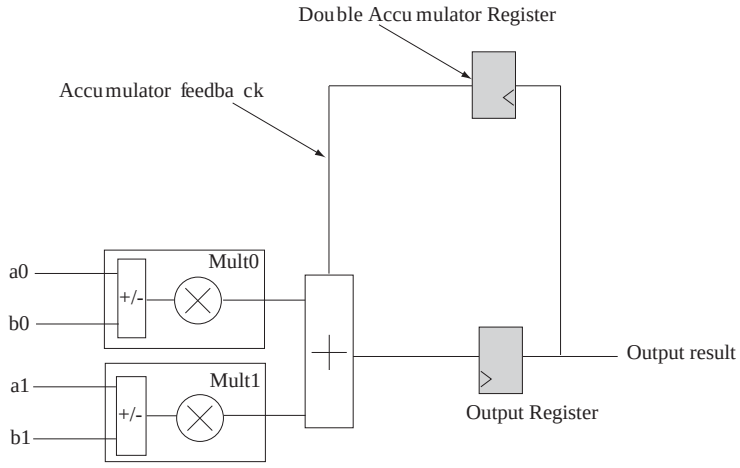


6.2.4. Double Accumulator

The double accumulator feature adds an additional register in the accumulator feedback path that process the interleaved complex data (I, Q) . 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 67. Double Accumulator



6.3. Parameters

You can customize the Multiply Adder Intel FPGA IP core by specifying the parameters using the parameter editor in the Intel Quartus Prime software.

6.3.1. General Tab

Table 84. General Tab

Parameter	Value	Default Value	Description
What is the 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?	1 - 256	16	Specify the width of the `dataa[]` port.
How wide should the B input buses be?	1 - 256	16	Specify the width of the `datab[]` port.
How wide should the 'result' output bus be?	1 - 256	32	Specify the width of the `result[]` port.
Create an associated clock enable for each clock	On Off	Off	Select this option to create clock enable for each clock.

6.3.2. Extra Modes

Table 85. Extra Modes Tab

Parameter	Value	Default Value	Description
Outputs Configuration			
Register output of the adder unit	On Off	Off	Turn on this option to enable output register of the adder module.

continued...

Parameter	Value	Default Value	Description
What is the source for clock input?	**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?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
Adder Operation			
What operation should be performed on outputs of the first pair of multipliers?	**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	On Off	Off	Turn on 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?	**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?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the addnsub1 register. You must select **Register 'addnsub1' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the addnsub1 register. You must select **Register 'addnsub1' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.

continued...

Parameter	Value	Default Value	Description
What operation should be performed on outputs of the second pair of multipliers?	**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.
Register 'addnsub3' input	On Off	Off	Turn on 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?	**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?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the addnsub3 register. You must select **Register 'addnsub3' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the addnsub3 register. You must select **Register 'addnsub3' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
Polarity			
Enable 'use_subadd'	On Off	Off	Turn on this option to reverse the function of addnsub input port. When this option is turned on, do the following: • drive addnsub to high for subtraction operation • drive addnsub to low for addition operation

6.3.3. Multipliers Tab

Table 86. Multipliers Tab

Parameter	Value	Default Value	Description
What is the representation format for Multipliers A inputs?	**SIGNED**, **UNSIGNED**,	**UNSIGNED**	Specify the representation format for the multiplier A input.

continued...

Parameter	Value	Default Value	Description
	VARIABLE		
Register 'signa' input	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?	**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?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `signa` register. You must select **Register 'signa' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `signa` register. You must select **Register 'signa' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the representation format for Multipliers B inputs?	**SIGNED**, **UNSIGNED**, **VARIABLE**	**UNSIGNED**	Specify the representation format for the multiplier B input.
Register 'signb' input	On Off	Off	Turn on 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?	**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?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `signb` register. You must select **Register 'signb' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `signb` register. You must select **Register 'signb' input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
Input Configuration			
Register input A of the multiplier	On Off	Off	Turn on this option to enable input register for `dataa` input bus.
What is the source for clock input?	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the register input clock signal for `dataa` input bus.

continued...

Parameter	Value	Default Value	Description
			You must select **Register input A of the multiplier** to enable this parameter.
What is the source for asynchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
Register input B of the multiplier	On Off	Off	Turn on this option to enable input register for `datab` input bus.
What is the source for clock input?	**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.
What is the source for asynchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the input A of the multiplier connected to?	**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	On Off	Off	Turn on 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?	**Clock0** **Clock1** **Clock2**	**Clock0**	Select **Clock0** , **Clock1** or **Clock2** to enable and specify the register input clock signal for `scanouta` output bus.

continued...

Parameter	Value	Default Value	Description
			You must turn on **Register output of the scan chain** parameter to enable this option.
What is the source for asynchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**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. The IP core supports either asynchronous or synchronous clear but not both.

6.3.4. Preadder Tab

Table 87. Preadder Tab

Parameter	Value	Default Value	Description
Select 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. **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	**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?	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	On Off	On	Turn on 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?	**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.

continued...

Parameter	Value	Default Value	Description
What is the source for asynchronous clear input?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `datac` input register. You must select **Register datac input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous clear source for the `datac` input register. You must select **Register datac input** to enable this parameter. The IP core supports either asynchronous or synchronous clear but not both.
Coefficients			
How wide should the coef width be?	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	On Off	Checked	Select this option to enable input register for `coefsel` input bus. You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
What is the source for clock input?	**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?	**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. The IP core supports either asynchronous or synchronous clear but not both.
What is the source for synchronous clear input	**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. The IP core supports either asynchronous or synchronous clear but not both.
Coefficient_0 Configuration	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	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	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.

continued...

Parameter	Value	Default Value	Description
			You must select **COEF** or **CONSTANT** for preadder mode to enable this parameter.
Coefficient_3 Configuration	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.

6.3.5. Accumulator Tab

Table 88. Accumulator Tab

Parameter	Value	Default Value	Description
Enable accumulator?	**YES**, **NO**	**NO**	Select **YES** to enable the accumulator. You must select **Register output of adder unit** when using accumulator feature.
What is the accumulator operation type?	**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	On Off	Off	Enable the `accum_sload` or `sload_accum` signals and the registers input to dynamically select the input to the accumulator. When `accum_sload` is low or `sload_accum` is high, the multiplier output is feed into the accumulator. When `accum_sload` is high or `sload_accum` is low, 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?	**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	0 - 64	64	Specify the preset constant value. This value can be 2^N where N is the preset constant value. N=64 represents a constant zero. You must select **Enable preload constant** option to enable this parameter.
What is the source for clock input?	**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.

continued...

Parameter	Value	Default Value	Description
What is the source for asynchronous clear input?	**NONE** **ACLR0** **ACLR1**	**NONE**	Specifies the asynchronous clear source for the `accum_sload/sload_accum` register. You must select **Enable preload constant** option to enable this parameter.
What is the source for synchronous clear input?	**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	**TRUE** **FALSE**	**FALSE**	To enable or disable the double accumulator feature.

6.3.6. Systolic/Chainout Tab

Table 89. Systolic/Chainout Adder Tab

Parameter	Value	Default Value	Description
Enable chainout adder	**YES**, **NO**	**NO**	Select **YES** to enable chainout adder module.
What is the chainout adder operation type?	**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_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?	**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
What is the source for asynchronous clear input?	**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
What is the source for synchronous clear input?	**NONE** **SCLR0** **SCLR1**	**NONE**	Specifies the synchronous 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
Systolic Delay			

continued...

Parameter	Value	Default Value	Description
Enable systolic delay registers	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?	**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?	**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?	**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.

6.3.7. Pipelining Tab

Table 90. 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. 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.

6.4. Signals

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

Table 91. Multiply Adder Intel FPGA IP 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
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.

continued...

Signal	Required	Description
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.

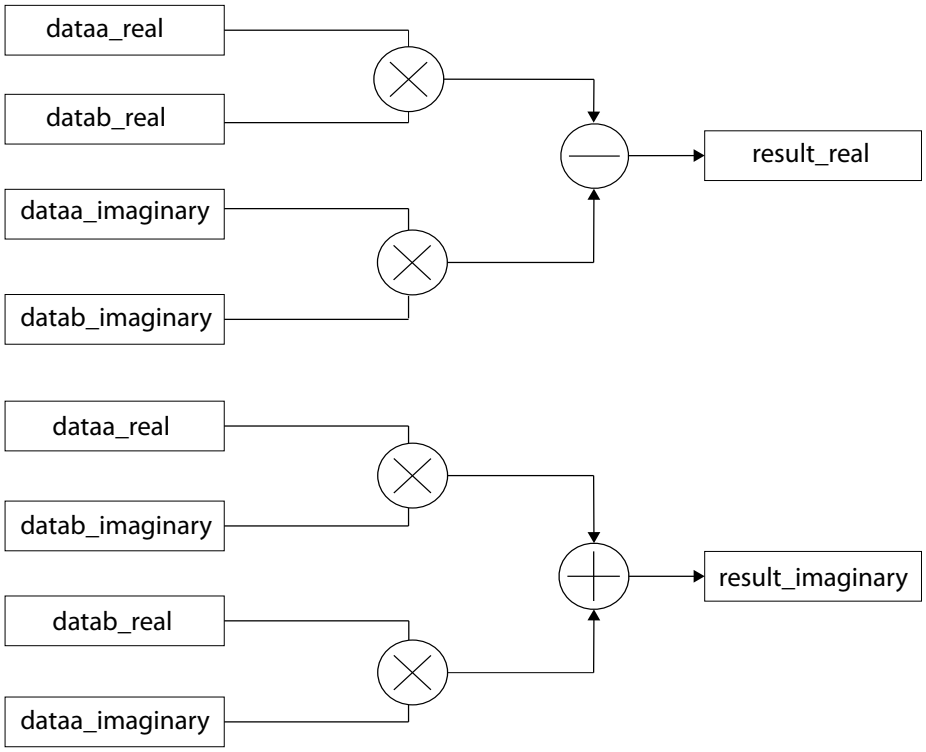
Table 92. Multiply Adder Intel FPGA IP Output Signals

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.

7. ALTMULT_COMPLEX Intel FPGA IP Core References

You can use the ALTMULT_COMPLEX Intel FPGA IP core to implement the complex multiplier by instantiating two multipliers. This is a family independent IP.

Figure 68. ALTMULT_COMPLEX Intel FPGA IP Block Diagram



7.1. ALTMULT_COMPLEX Intel FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 93. ALTMULT_COMPLEX Intel FPGA IP Release Information

Item	Description
IP Version	19.1.0
Intel Quartus Prime Version	21.2
Release Date	2021.06.23

7.2. Features

The ALTMULT_COMPLEX Intel FPGA 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 partial products calculations..
- Supports data width of 1–256 bits
- Supports signed and unsigned data representation format
- Supports pipelining with configurable output latency
- Supports optional asynchronous and synchronous clear and clock enable input ports

7.3. 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.

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

Figure 69. Equation for Two Complex Numbers Multiplication

$$\begin{aligned} x * y &= (a + ib)(c + id) \\ &= ac + ibc + iad - bd \\ &= (ac - bd) + i(ad + bc) \end{aligned}$$

7.4. Parameters

Table 94. ALTMULT_COMPLEX Intel FPGA IP Parameters

Parameter	Value	Default Value	Description
General			
How wide should the A input buses be?	1–256	18	Specifies the number of bits for `dataa_imag` and `dataa_real` input buses.
How wide should the B input buses be?	1–256	18	Specifies the number of bits for `datab_imag` and `datab_real` input buses.
How wide should the 'result' output bus be?	1–256	36	Specifies the number of bits for 'result' output bus.
Input Representation			
What is the representation format for A inputs?	**Signed**, **Unsigned**	**Signed**	Specifies the representation format for A inputs. Only **Signed** representation format is supported in Intel Agilex 7 devices.
What is the representation format for B inputs?	**Signed**, **Unsigned**	**Signed**	Specifies the representation format for B inputs. Only **Signed** representation format is supported in Intel Agilex 7 devices.
Implementation Style			
Which implementation style should be used?	**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**	Intel Agilex 7 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			
Output latency	0 - 11	4	Specifies the number of clock cycles for output latency.
Create a Clear input?	**NONE** **ACLR** **SCLR**	**NONE**	Select this option to create `aclr` or `sclr` signal for the complex multiplier.
Create a Clock Enable input?	On Off	Off	Select this option to create `ena` signal for the complex multiplier clock.

7.5. Signals

Table 95. ALTMULT_COMPLEX Intel FPGA IP 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 **How wide should the A input buses be?** 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 **How wide should the A input buses be?** 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 **How wide should the B input buses be?** 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 **How wide should the B input buses be?** parameter value.
ena	No	Active high clock enable for the clock signal of the complex multiplier.

Table 96. ALTMULT_COMPLEX Intel FPGA IP Output Signals

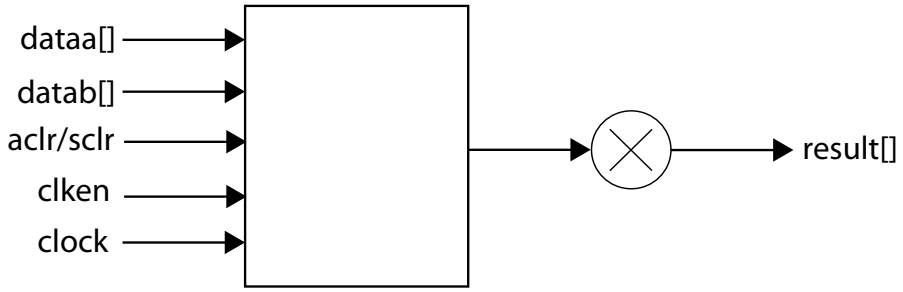
Signal	Required	Description
result_imag	Yes	Imaginary output value of the multiplier. The size of the output signal depends on the WIDTH_RESULT parameter value.
result_real	Yes	Real output value of the multiplier. The size of the output signal depends on the WIDTH_RESULT parameter value.

8. LPM_MULT Intel FPGA IP Core References

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

This is a family independent IP.

Figure 70. LPM_MULT Intel FPGA IP Core Architecture



8.1. LPM_MULT Intel FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 97. LPM_MULT Intel FPGA IP Release Information

Item	Description
IP Version	19.2.0
Intel Quartus Prime Version	21.2
Release Date	2021.06.23

8.2. Features

The LPM_MULT 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 and synchronous clear and clock enable input ports

8.3. Parameters

You can customize the LPM_MULT Intel FPGA IP core by specifying the parameters using the IP Parameter Editor in the Intel Quartus Prime software.

8.3.1. General Tab

Table 98. 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.

8.3.2. General 2 Tab

Table 99. 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.
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. When **SCLR** is selected for **Clear Signal Type** parameter, only **Use the dedicated multiplier circuitry** option is available.

8.3.3. Pipelining Tab

Table 100. 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 generates ACLR port. Select **SCLR** to use synchronous clear for the pipeline register. This generates SCLR port.
Create a 'clken' clock enable clock	Off On	Off	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 optimization for the IP core.

8.4. Signals

Table 101. LPM_MULT Intel FPGA IP Core Input Signals

Signal Name	Required	Description
dataa[]	Yes	Data input. The size of the input signal depends on the **Dataa width** parameter value.
datab[]	Yes	Data input. The size of the input signal depends on the **Datab width** parameter value.
clock	No	Clock input for pipelined usage. For **Latency** values other than **1** (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 102. LPM_MULT Intel FPGA IP Output signals

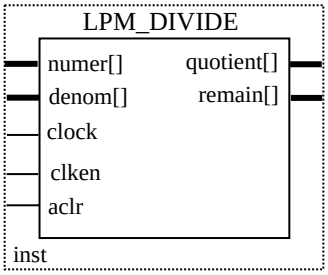
signal Name	Required	Description
result[]	Yes	Data output. The size of the output signals depends on the **Result width** parameter.

9. LPM_DIVIDE Intel FPGA IP Core References

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 Intel FPGA IP core.

Figure 71. LPM_DIVIDE Ports



9.1. LPM_DIVIDE Intel FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 103. LPM_DIVIDE Intel FPGA IP Release Information

Item	Description
IP Version	19.1.0
Intel Quartus Prime Version	21.2
Release Date	2021.06.23

9.2. 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.

9.3. 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
```

9.4. 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;
```

9.5. 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;
```

9.6. Ports

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

Table 104. 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.
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 105. 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.

9.7. Parameters

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

9.7.1. General Tab

Parameter Name	Value	Default Value	Description
How wide should the 'numerator' input bus be?	1–64	8	Specifies the widths of the `numer[]` and `quotient[]` ports.
How wide should the 'denominator' input bus be?	1–64	8	Specifies the widths of the `denom[]` and `remain[]` ports. Values are 1 to 64.
Numerator Representation	• **Unsigned** • **Signed**	**Unsigned**	Sign representation of the numerator input. When this parameter is set to **Signed**, the divider interprets the `numer[]` input as signed two's complement.
Denominator Representation	• **Unsigned** • **Signed**	**Unsigned**	Sign representation of the denominator input. When this parameter is set to **Signed**, the divider interprets the `denom[]` input as signed two's complement.

9.7.2. General1 Tab

Parameter Name	Value	Default Value	Description
Pipelining			
Output latency	0–14	0	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 **Output latency** parameter that is higher than the value specified in the **How wide should the 'numerator' input bus be?** parameter.
Create an asynchronous Clear input?	• On • Off	Off	Select this option to create `aclr` signal.
Create a Clock Enable Input?	• On • Off	Off	Select this option to create `clken` signal for the IP clock.
Optimization			
Which do you wish to optimize?	• **Default Optimization** • **Area** • **Speed**	**Default Optimization**	Specify type of optimization for a specific instance of the IP.

continued...

Parameter Name	Value	Default Value	Description
			• **Default Optimization**: Select this option to use Intel Quartus Prime software to optimize using default optimization technique logic for a specific instance of the IP. • **Area**: Select this option to use Intel Quartus Prime software to optimize routability for a specific instance of the IP. • **Speed**: Select this option to use Intel Quartus Prime software to optimize speed by using carry chains for a specific instance of the IP.
Remainder			
Always return a positive remainder?	• **Yes** • **No**	**Yes**	In order to reduce area and improve speed, Intel recommends setting this parameter to **Yes** in operations where the remainder must be positive or unimportant.

10. Native Floating Point DSP Intel Agilex FPGA IP References

The Native Floating Point DSP Intel Agilex FPGA IP instantiates and controls a single Intel Agilex 7 Variable Precision DSP block.

10.1. Native Floating Point DSP Intel Agilex FPGA IP Release Information

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 106. Native Floating Point DSP Intel Agilex FPGA IP Release Information

Item	Description
IP Version	19.1.0
Intel Quartus Prime Version	21.2
Release Date	2021.06.23

10.2. Native Floating Point DSP Intel Agilex FPGA IP Core Supported Operational Modes

Table 107. Operational Modes Supported by Native Floating Point DSP Intel Agilex FPGA IP Core

Operational Modes	Description	Supported Exception Flags
FP32 multiplication mode	This mode performs single precision multiplication operation. This mode applies the following equation: • **fp32_result** = fp32_result = fp32_mult_a*fp32_mult_b	• `fp32_mult_overflow` • `fp32_mult_underflow` • `fp32_mult_inexact` • `fp32_mult_invalid`
FP32 addition or subtraction mode	This mode performs single precision addition or subtraction operation. This mode applies the following equations: • **fp32_result** = fp32_adder_b+fp32_adder_a • **fp32_result** = fp32_adder_b-fp32_adder_a	• `fp32_adder_overflow` • `fp32_adder_underflow` • `fp32_adder_inexact` • `fp32_adder_invalid`
FP32 multiplication with addition or subtraction mode	This mode performs single precision multiplication, followed by addition or subtraction operations. This mode applies the following equations: • When chainin feature is enabled: — **fp32_result** = (fp32_mult_a*fp32_mult_b) + fp32_chainin — **fp32_result** = (fp32_mult_a*fp32_mult_b) - fp32_chainin • When chainin feature is disabled: — **fp32_result** = (fp32_mult_a*fp32_mult_b) + fp32_adder_a — **fp32_result** = (fp32_mult_a*fp32_mult_b) - fp32_adder_a	• `fp32_mult_overflow` • `fp32_mult_underflow` • `fp32_mult_inexact` • `fp32_mult_invalid` • `fp32_adder_overflow` • `fp32_adder_underflow` • `fp32_adder_inexact` • `fp32_adder_invalid`
FP32 multiplication with accumulation mode	This mode performs floating-point multiplication followed by floating-point addition or subtraction with the previous multiplication result. This mode applies the following equations: • When `accumulate` signal is driven high: — **fp32_result(t)** = [fp32_mult_a(t)*fp32_mult_b(t)] + fp32_result(t-1) — **fp32_result(t)** = [fp32_mult_a(t)*fp32_mult_b(t) - fp32_result(t-1) • When `accumulate` signal is driven low: — **fp32_result** = fp32_mult_a*fp32_mult_b.	
FP32 vector one mode	This mode performs floating-point multiplication followed by floating-point addition or subtraction with the chainin input from the previous variable DSP Block. This mode applies the following equations: • When chainin feature is enabled: — **fp32_result** = (fp32_mult_a * fp32_mult_b) + fp32_chainin, **fp32_chainout** = fp32_adder_a — **fp32_result** = (fp32_mult_a * fp32_mult_b) - fp32_chainin, **fp32_chainout** = fp32_adder_a • When chainin feature is disabled: — **fp32_result** = fp32_mult_a * fp32_mult_b, **fp32_chainout** = fp32_adder_a	

continued...

Operational Modes	Description	Supported Exception Flags
FP32 vector two mode	This mode performs floating-point multiplication where the multiplication result is directly fed to `chainout`. The `chainin` input from the previous variable DSP Block is then added or subtracted from input Ax as the output result. This mode applies the following equations: • When chainin feature is enabled: — **fp32_result** = fp32_adder_a + fp32_chainin, **fp32_chainout** = fp32_mult_a * fp32_mult_b — **fp32_result** = fp32_adder_a - fp32_chainin, **fp32_chainout** = fp32_mult_a * fp32_mult_b • When chainin feature is disabled: — **fp32_result** = fp32_adder_a, fp32_chainout = fp32_mult_a * fp32_mult_b	
Sum of two FP16 multiplication mode	This mode performs a summation of two half-precision multiplication and provide a single-precision result. This mode applies the following equations: • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) + (fp16_mult_bot_a*fp16_mult_bot_b) • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) - (fp16_mult_bot_a*fp16_mult_bot_b)	Exception flags supported in flushed and bfloat16 formats: • `fp16_mult_top_invalid` • `fp16_mult_top_inexact` • `fp16_mult_top_overflow` • `fp16_mult_top_underflow` • `fp16_mult_bot_invalid` • `fp16_mult_bot_inexact` • `fp16_mult_bot_overflow` • `fp16_mult_bot_underflow` • `fp16_adder_invalid` • `fp16_adder_inexact` • `fp16_adder_overflow` • `fp16_adder_underflow` Exception flags supported in extended format: • `fp16_mult_top_invalid` • `fp16_mult_top_inexact` • `fp16_mult_top_infinite` • `fp16_mult_top_zero` • `fp16_mult_bot_invalid` • `fp16_mult_bot_inexact` • `fp16_mult_bot_infinite` • `fp16_mult_bot_zero` • `fp16_adder_invalid` • `fp16_adder_inexact` • `fp16_adder_infinite` • `fp16_adder_zero`
Sum of two FP16 multiplication with FP32 addition mode	This mode performs a summation of two half-precision multiplication and provide a single-precision result. This mode applies the following equations: • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) + (fp16_mult_bot_a*fp16_mult_bot_b) - fp32_adder_a • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) - (fp16_mult_bot_a*fp16_mult_bot_b) - fp32_adder_a • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) + (fp16_mult_bot_a*fp16_mult_bot_b) + fp32_adder_a • **fp32_result** = (fp16_mult_top_a*fp16_mult_top_b) - (fp16_mult_bot_a*fp16_mult_bot_b) + fp32_adder_a	Exception flags supported in flushed and bfloat16 formats: • `fp16_mult_top_invalid` • `fp16_mult_top_inexact` • `fp16_mult_top_overflow` • `fp16_mult_top_underflow` • `fp16_mult_bot_invalid` • `fp16_mult_bot_inexact` • `fp16_mult_bot_overflow` • `fp16_mult_bot_underflow` • `fp16_adder_invalid`

continued...

Operational Modes	Description	Supported Exception Flags
		• fp16_adder_inexact • fp16_adder_overflow • fp16_adder_underflow • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow Exception flags supported in extended format: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_infinite • fp16_mult_top_zero • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_infinite • fp16_mult_bot_zero • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_infinite • fp16_adder_zero • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow
Sum of two FP16 multiplication with accumulation mode	This mode performs a summation of two half-precision multiplication and accumulate the value into single-precision format. This mode applies the following equations: • When accumulate signal is driven high: — **fp32_result (t)** = [fp16_mult_top_a(t) * fp16_mult_top_b(t)] + [fp16_mult_bot_a(t) * fp16_mult_bot_b(t)] + fp32_result(t-1) — **fp32_result (t)** = [fp16_mult_top_a(t) * fp16_mult_top_b(t)] - [fp16_mult_bot_a(t) * fp16_mult_bot_b(t)] + fp32_result(t-1) — **fp32_result (t)** = [fp16_mult_top_a(t) * fp16_mult_top_b(t)] + [fp16_mult_bot_a(t) * fp16_mult_bot_b(t)] - fp32_result(t-1) — **fp32_result (t)** = [fp16_mult_top_a(t) * fp16_mult_top_b(t)] - [fp16_mult_bot_a(t) * fp16_mult_bot_b(t)] - fp32_result(t-1) • When accumulate signal is driven low: — **fp32_result** = [fp16_mult_top_a * fp16_mult_top_b] + [fp16_mult_bot_a * fp16_mult_bot_b] — **fp32_result** = [fp16_mult_top_a * fp16_mult_top_b] - [fp16_mult_bot_a * fp16_mult_bot_b]	Exception flags supported in flushed and bfloat16 formats: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_overflow • fp16_mult_top_underflow • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_overflow • fp16_mult_bot_underflow • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_overflow • fp16_adder_underflow • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow Exception flags supported in extended format: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_infinite • fp16_mult_top_zero • fp16_mult_bot_invalid

continued...

Operational Modes	Description	Supported Exception Flags
		• fp16_mult_bot_inexact • fp16_mult_bot_infinite • fp16_mult_bot_zero • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_infinite • fp16_adder_zero • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow
FP16 vector one mode	This mode performs a summation of two half-precision multiplications with the chainin input from the previous variable DSP Block. The output is a single-precision floating-point value which is fed into chainout. This mode applies the following equation: • When chainin feature is enabled: — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) + fp32_chainin, **fp32_chainout** = fp32_adder_a — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) + fp32_chainin, **fp32_chainout** = fp32_adder_a — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) - fp32_chainin, **fp32_chainout** = fp32_adder_a — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) - fp32_chainin, **fp32_chainout** = fp32_adder_a • When chainin feature is disabled: — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b), **fp32_chainout** = fp32_adder_a — **fp32_result** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b), **fp32_chainout** = fp32_adder_a	Exception flags supported in flushed and bfloat16 formats: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_overflow • fp16_mult_top_underflow • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_overflow • fp16_mult_bot_underflow • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_overflow • fp16_adder_underflow • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow Exception flags supported in extended format: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_infinite • fp16_mult_top_zero • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_infinite • fp16_mult_bot_zero • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_infinite • fp16_adder_zero • fp32_adder_invalid

continued...

Operational Modes	Description	Supported Exception Flags
		• fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow
FP16 vector two mode	This mode performs a summation of two half precision multiplication and fed to chainout. The chainin input from the previous variable DSP Block is then added or subtracted from input fp32_adder_a as the output result. This mode applies the following equation: • When chainin feature is enabled: — **fp32_result** = fp32_adder_a + fp32_chainin, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) — **fp32_result** = fp32_adder_a - fp32_chainin, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) — **fp32_result** = fp32_adder_a + fp32_chainin, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) — **fp32_result** = fp32_adder_a - fp32_chainin, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) • When chainin feature is disabled: — **fp32_result** = fp32_adder_a, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b) — **fp32_result** = fp32_adder_a, **fp32_chainout** = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b)	Exception flags supported in flushed and bfloat16 formats: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_overflow • fp16_mult_top_underflow • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_overflow • fp16_mult_bot_underflow • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_overflow • fp16_adder_underflow • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow Exception flags supported in extended format: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_infinite • fp16_mult_top_zero • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_infinite • fp16_mult_bot_zero • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_infinite • fp16_adder_zero • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow
FP16 vector three	This mode performs a single-precision accumulation and a summation of two half-precision multiplications. This mode applies the following equation:	Exception flags supported in flushed and bfloat16 formats: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_overflow • fp16_mult_top_underflow • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_overflow • fp16_mult_bot_underflow • fp16_adder_invalid • fp16_adder_inexact

Operational Modes	Description	Supported Exception Flags
	• When accumulate is driven high: — **fp32_result(t)** = fp32_adder_a(t) + fp32_result(t-1), **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b} — **fp32_result(t)** = fp32_adder_a(t) - fp32_result(t-1), **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b} — **fp32_result(t)** = fp32_adder_a(t) + fp32_result(t-1), **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} - {fp16_mult_bot_a * fp16_mult_bot_b} — **fp32_result(t)** = fp32_adder_a(t) - fp32_result(t-1), **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} - {fp16_mult_bot_a * fp16_mult_bot_b} • When accumulate is driven low: — **fp32_result** = fp32_adder_a, **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b} — **fp32_result** = fp32_adder_a, **fp32_chainout** = {fp16_mult_top_a * fp16_mult_top_b} - {fp16_mult_bot_a * fp16_mult_bot_b}	• fp16_adder_overflow • fp16_adder_underflow • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow Exception flags supported in extended format: • fp16_mult_top_invalid • fp16_mult_top_inexact • fp16_mult_top_infinite • fp16_mult_top_zero • fp16_mult_bot_invalid • fp16_mult_bot_inexact • fp16_mult_bot_infinite • fp16_mult_bot_zero • fp16_adder_invalid • fp16_adder_inexact • fp16_adder_infinite • fp16_adder_zero • fp32_adder_invalid • fp32_adder_inexact • fp32_adder_overflow • fp32_adder_underflow

10.3. Parameterizing the Native Floating Point DSP Intel Agilex FPGA IP

Select different parameters to create an IP core suitable for your design.

1. In Intel Quartus Prime Pro Edition,create a new project that targets a Intel Agilex 7 device.
2. In IP Catalog, click **Library ➤ DSP ➤ Primitive DSP ➤ Native Floating Point DSP Intel Agilex FPGA IP**.
   The Native Floating Point DSP Intel Agilex FPGA IP Core IP parameter editor opens.
3. In the **New IP Variation** dialog box, enter an **Entity Name** and click **OK**.
4. Under **Parameters**, select the operation mode, features, and register configurations according to the variant of your IP core
5. Click **Generate HDL**.
6. Click **Finish**.

10.3.1. General Tab

Table 108. General Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Operation Mode				
Choose the operation mode	operation_mode	**fp32_mult** **fp32_add** **fp32_mult_add** **fp32_mult_acc** **fp32_vector1** **fp32_vector2** **fp16_sumof2mult** **fp16_sumof2mult_add_fp32** **fp16_sumof2mult_acc** **fp16_vector1** **fp16_vector2** **fp16_vector3**	**fp32_mult_add**	Select the desired floating-point operation mode.
Enable fp32_chainin	use_chainin	**No** **Yes**	**No**	Select to enable chainin feature. When you enable the chainin feature, the result from the multiplier is added or subtracted by the input from `chainin` port.
Enable fp32_chainout	enable_chainout	**No** **Yes**	**No**	Select to enable the `chainout` port.
FP32 Operation				
Perform subtraction in fp32_adder	fp32_adder_subtract	**No** **Yes**	**No**	Select **Yes** to set FP32 adder to perform subtraction. Select **No** to set FP32 adder to perform addition.
FP16 Representation/Operation				
Select the mode for fp16	fp16_mode	**FLUSHED** **EXTENDED** **BFLOAT16**	**FLUSHED**	Select the precision format for FP16 operation modes.
Select the width size for fp16 (Only for bfloat16 mode)	fp16_input_width	**16** **19**	**16**	Specify the width of FP16 data input bus.
Perform subtraction in fp16_adder	fp16_adder_subtract	**No** **Yes**	**No**	Select **Yes** to set FP16 adder to perform subtraction. Select **No** to set FP16 adder to perform addition.
Exception Flag				
Enable exception flag	enable_exception_flag	**No** **Yes**	**No**	Select to enable exception flag feature.

10.3.2. Registers Tab

Table 109. Registers Tab

Parameter	IP Generated Parameter	Value	Default Value	Description
Clear Signal Setting				
Type of clear signal	clear_type	**none** **aclr** **sclr**	none	Specify the clear signal behavior for all registers in the floating-point DSP block. • **none**: Select to not use any clear signal. • **aclr**: Select to use asynchronous clear signal type for all registers. • **sclr**: Select to use synchronous clear signal type for all registers.
Enable clr0 signal for all input registers	enable_clr0	**No** **Yes**	**No**	Select **Yes** to enable `clr[0]` signal for all input registers.
Enable clr1 for output and pipeline registers	enable_clr1	**No** **Yes**	**No**	Select **Yes** to enable `clr[1]` signal for output and pipeline registers.
Input Registers				
Enable for input 'accumulate'	accumulate_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `accumulate` input register. Select **no_reg** to disable the register.
Enable for input 'fp32_adder_a'	fp32_adder_a_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for `fp32_adder_a` input register. Select **no_reg** to disable the register.
Enable for input 'fp32_adder_b'	fp32_adder_b_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `fp32_adder_b` input register. Select **no_reg** to disable the register.
Enable for input 'fp32_mult_a'	fp32_mult_a_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for `fp32_mult_a` input register. Select **no_reg** to disable the register.
Enable for input 'fp32_mult_b'	fp32_mult_b_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for `fp32_mult_b` input register. Select **no_reg** to disable the register.
Enable for input 'fp16_mult_input'	fp16_mult_input_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for `fp16_mult_input` input register. Select **no_reg** to disable the register.
Output Registers				
Enable output register	output_clken	**no_reg** **ena0**	**ena0**	Specify the clock enable signal for output register.

continued...

Parameter	IP Generated Parameter	Value	Default Value	Description
		ena1 **ena2**		Select **no_reg** to disable the register.
Pipeline Registers				
Enable 'accum_adder' register	accum_adder_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for accum_adder pipeline register. Select **no_reg** to disable the register.
Enable 'adder_input' register	adder_input_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for adder_input pipeline register. Select **no_reg** to disable the register.
Enable 'adder_pl' register	adder_pl_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for adder_pl pipeline register. Select **no_reg** to disable the register.
Enable 'fp32_adder_a_chainin_pl' register	fp32_adder_a_chainin_pl_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for fp32_adder_a_chainin_pl pipeline register. Select **no_reg** to disable the register.
Enable 'accum_pipeline' register	accum_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for accum_pipeline register. Select **no_reg** to disable the register.
Enable 'mult_pipeline' register	mult_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for mult_pipeline register. Select **no_reg** to disable the register.
Enable 'fp32_adder_a_chainin_2nd_pl' register	fp32_adder_a_chainin_2nd_pl_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for fp32_adder_a_chainin_2nd_pl pipeline register. Select **no_reg** to disable the register.
Enable 'accum_2nd_pipeline' register	accum_2nd_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**no_reg**	Specify the clock enable signal for accum_2nd_pipeline register. Select **no_reg** to disable the register.
Enable 'mult_2nd_pipeline' register	mult_2nd_pipeline_clken	**no_reg** **ena0** **ena1** **ena2**	**ena0**	Specify the clock enable signal for mult_2nd_pipeline register. Select **no_reg** to disable the register.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input, pipeline, and output registers.

10.4. Native Floating Point DSP Intel Agilex FPGA IP Core Signals

10.4.1. FP32 Multiplication Mode Signals

Figure 72. FP32 Multiplication Mode Signals

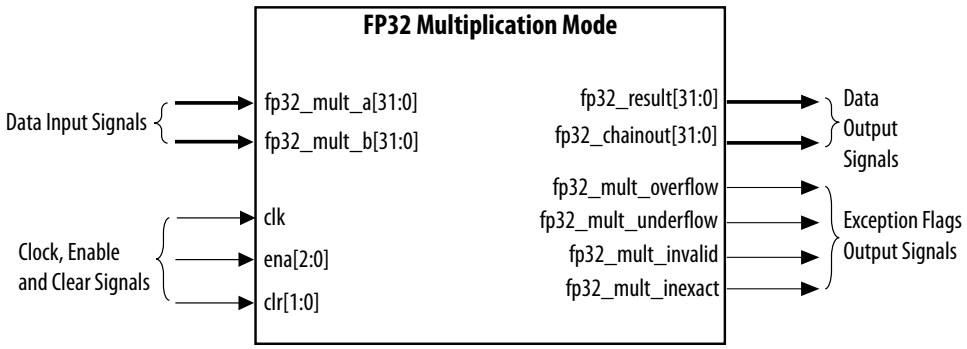


Table 110. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
fp32_mult_a[31:0]	Input	32	Low	Input data bus to the multiplier.
fp32_mult_b[31:0]	Input	32	Low	Input data bus to the multiplier.
fp32_result[31:0]	Output	32	—	Output data bus from IP core.
fp32_chainout[31:0]	Output	32	—	Connect these signals to the chainin signals of the next floating-point DSP IP core.

Table 111. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	Low	Input clock for all registers.
ena[2:0]	Input	3	High	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 112. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp32_mult_overflow	Output	1	—	This signal indicates if the FP32 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity.

continued...

Signal Name	Type	Width	Default	Description
				0: If the multiplier result is not larger than the maximum presentable value.
fp32_mult_underflow	Output	1	—	This signal indicates if the FP32 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_mult_inexact	Output	1	—	This signal indicates if the FP32 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_mult_invalid	Output	1	—	This signal indicates if the FP32 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.2. FP32 Addition or Subtraction Mode Signals

Figure 73. FP32 Addition or Subtraction Mode Signals

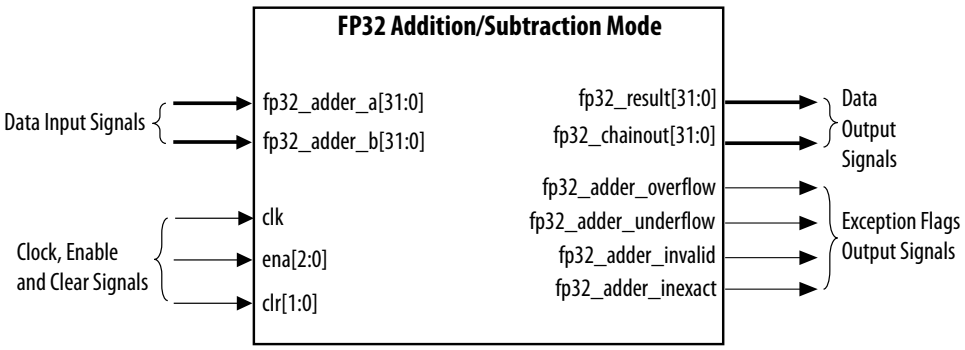


Table 113. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
fp32_adder_a[31:0]	Input	32	Low	Input data bus to the adder.
fp32_adder_b[31:0]	Input	32	Low	Input data bus to the adder.
fp32_result[31:0]	Output	32	—	Output data bus from IP core.
fp32_chainout[31:0]	Output	32	—	Connect these signals to the chainin signals of the next floating-point DSP IP core.

Table 114. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 115. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp32_adder_overflow	Output	1	—	This signal indicates if the adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the adder result is not larger than the maximum presentable value.
fp32_adder_underflow	Output	1	—	This signal indicates if the adder result is a smaller value compared to the minimum presentable value. 1: If the adder result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the adder result is a larger than the minimum representable value.
fp32_adder_inexact	Output	1	—	This signal indicates if the adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the adder result does not meet any of the criteria above.
fp32_adder_invalid	Output	1	—	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the adder result is invalid and cast to qNaN. 0: If the adder result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.3. FP32 Multiplication with Addition or Subtraction Mode Signals

Figure 74. FP32 Multiplication with Addition or Subtraction Mode Signals

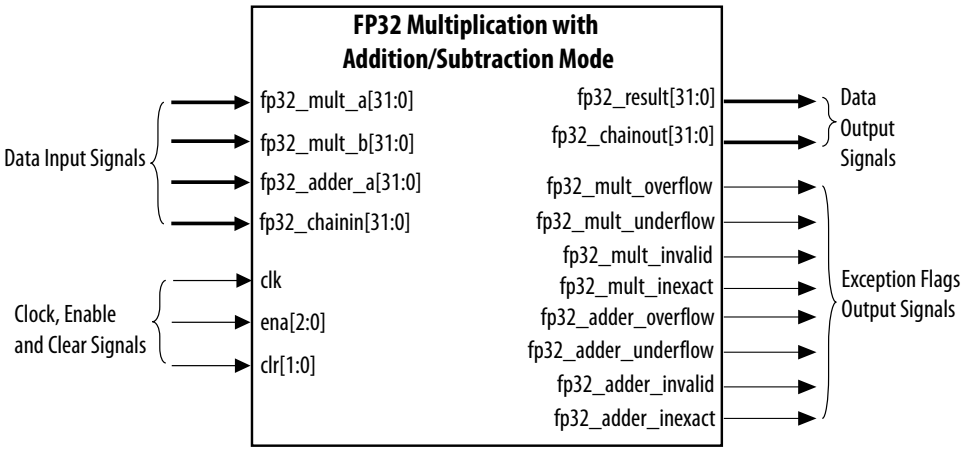


Table 116. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
`fp32_mult_a[31:0]`	Input	32	Low	Input data bus to the multiplier.
`fp32_mult_b[31:0]`	Input	32	Low	Input data bus to the multiplier.
`fp32_adder_a[31:0]`	Input	32	Low	Input data bus to the adder.
`fp32_chainin[31:0]`	Input	32	Low	Connect these signals to the `chainout` signals from the preceding floating-point DSP IP core.
`fp32_result[31:0]`	Output	32	—	Output data bus from IP core.
`fp32_chainout[31:0]`	Output	32	—	Connect these signals to the `chainin` signals of the next floating-point DSP IP core.

Table 117. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
`clk[0]`	Input	1	—	Input clock for all registers.
`ena[2:0]`	Input	3	—	Clock enable signals for all registers. These signals are active-High.
`clr[1:0]`	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 118. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp32_mult_overflow	Output	1	—	This signal indicates if the FP32 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp32_mult_underflow	Output	1	—	This signal indicates if the FP32 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_mult_inexact	Output	1	—	This signal indicates if the FP32 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_mult_invalid	Output	1	—	This signal indicates if the FP32 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp32_adder_overflow	Output	1	—	This signal indicates if the adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp32_adder_underflow	Output	1	—	This signal indicates if the adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_adder_inexact	Output	1	—	This signal indicates if the adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value.

continued...

Signal Name	Type	Width	Default	Description
				0: If the multiplier result does not meet any of the criteria above.
fp32_adder_invalid	Output	1	—	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.4. FP32 Multiplication with Accumulation Mode Signals

Figure 75. FP32 Multiplication with Accumulation Mode Signals

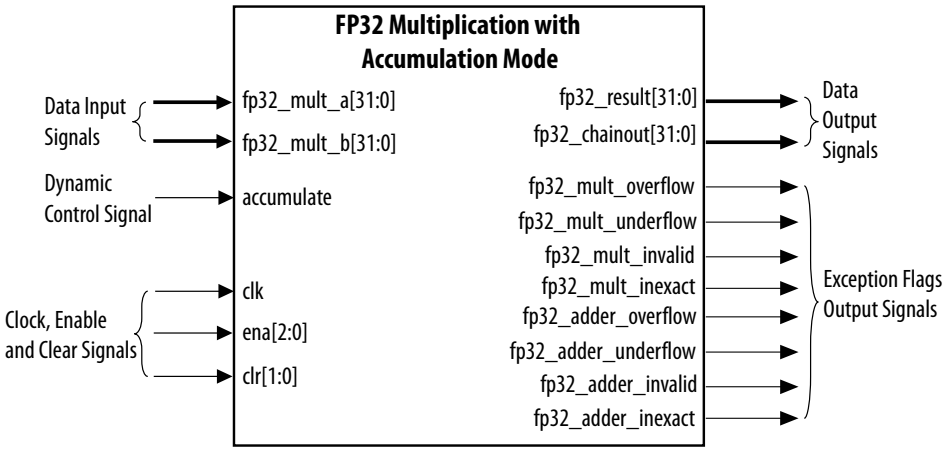


Table 119. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
fp32_mult_a[31:0]	Input	32	Low	Input data bus to the multiplier.
fp32_mult_b[31:0]	Input	32	Low	Input data bus to the multiplier.
fp32_result[31:0]	Output	32	—	Output data bus from IP core.
fp32_chainout[31:0]	Output	32	—	Connect these signals to the chainin signals of the next floating-point DSP IP core.

Table 120. Dynamic Control Signal

Signal Name	Type	Width	Default	Description
accumulate	Input	1	Low	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 1: Enable feedback the adder's output. • 0: Disable the feedback mechanism.

Table 121. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 122. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp32_mult_overflow	Output	1	—	This signal indicates if the FP32 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp32_mult_underflow	Output	1	—	This signal indicates if the FP32 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_mult_inexact	Output	1	—	This signal indicates if the FP32 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_mult_invalid	Output	1	—	This signal indicates if the FP32 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp32_adder_overflow	Output	1	—	This signal indicates if the adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.

continued...

Signal Name	Type	Width	Default	Description
fp32_adder_underflow	Output	1	—	This signal indicates if the adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_adder_inexact	Output	1	—	This signal indicates if the adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_adder_invalid	Output	1	—	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.5. FP32 Vector One and Vector Two Modes Signals

Figure 76. FP32 Vector One and Vector Two Modes Signals

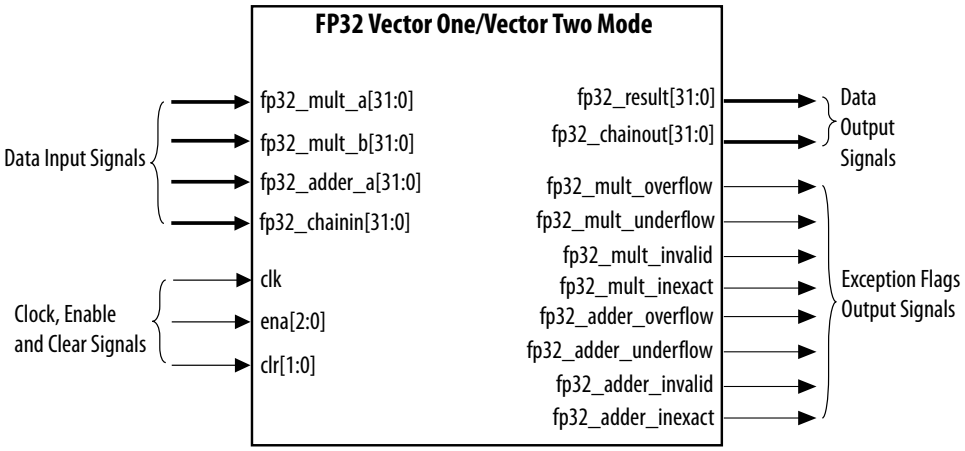


Table 123. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
fp32_mult_a[31:0]	Input	32	Low	Input data bus to the multiplier.
fp32_mult_b[31:0]	Input	32	Low	Input data bus to the multiplier.

continued...

Signal Name	Type	Width	Default	Description
fp32_add_a[31:0]	Input	32	Low	Input data bus to the adder.
fp32_chainin[31:0]	Input	32	Low	Connect these signals to the chainout signals from the preceding floating-point DSP IP core.
fp32_result[31:0]	Output	32	—	Output data bus from IP core.
fp32_chainout[31:0]	Output	32	—	Connect these signals to the chainin signals of the next floating-point DSP IP core.

Table 124. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 125. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp32_mult_overflow	Output	1	—	This signal indicates if the FP32 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp32_mult_underflow	Output	1	—	This signal indicates if the FP32 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_mult_inexact	Output	1	—	This signal indicates if the FP32 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_mult_invalid	Output	1	—	This signal indicates if the FP32 multiplier operation is ill-defined and produces an invalid result.

continued...

Signal Name	Type	Width	Default	Description
				1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp32_adder_overflow	Output	1	—	This signal indicates if the adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp32_adder_underflow	Output	1	—	This signal indicates if the adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp32_adder_inexact	Output	1	—	This signal indicates if the adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp32_adder_invalid	Output	1	—	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.6. Sum of Two FP16 Multiplication Mode Signals

Figure 77. Sum of Two FP16 Multiplication Mode Signals

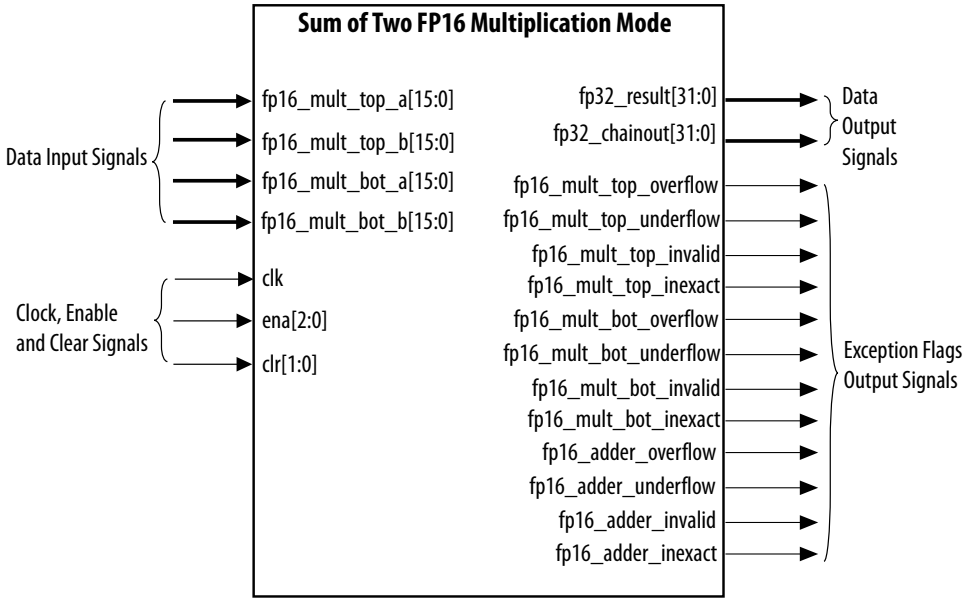


Table 126. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
`fp16_mult_top_a[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_top_b[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_bot_a[15:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp16_mult_bot_b[16:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp32_result[31:0]`	Output	32	—	Output data bus from IP core.
`fp32_chainout[31:0]`	Output	32	—	Connect these signals to the `chainin` signals of the next floating-point DSP IP core.

Table 127. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
`clk[0]`	Input	1	—	Input clock for all registers.
`ena[2:0]`	Input	3	—	Clock enable signals for all registers. These signals are active-High.
`clr[1:0]`	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 128. Exception Flag Signals

Signal Name	Type	Width	Default	Description
`fp16_mult_top_overflow/ fp16_mult_bot_overflow`	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
`fp16_mult_top_underflow/ fp16_mult_bot_underflow`	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
`fp16_mult_top_inexact/ fp16_mult_bot_inexact`	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
`fp16_mult_top_invalid/ fp16_mult_bot_invalid`	Output	1	—	This signal indicates if the top/bottom fp16 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
`fp16_adder_overflow`	Output	1	—	This signal indicates if the adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
`fp16_adder_underflow`	Output	1	—	This signal indicates if the adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
`fp16_adder_inexact`	Output	1	—	This signal indicates if the adder result is an exact representation. 1: If the adder result is:

continued...

Signal Name	Type	Width	Default	Description
				• a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_adder_invalid	Output	1	—	This signal indicates if the adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.7. Sum of Two FP16 Multiplication with FP32 Addition Mode Signals

Figure 78. Sum of Two FP16 Multiplication with FP32 Addition Mode Signals

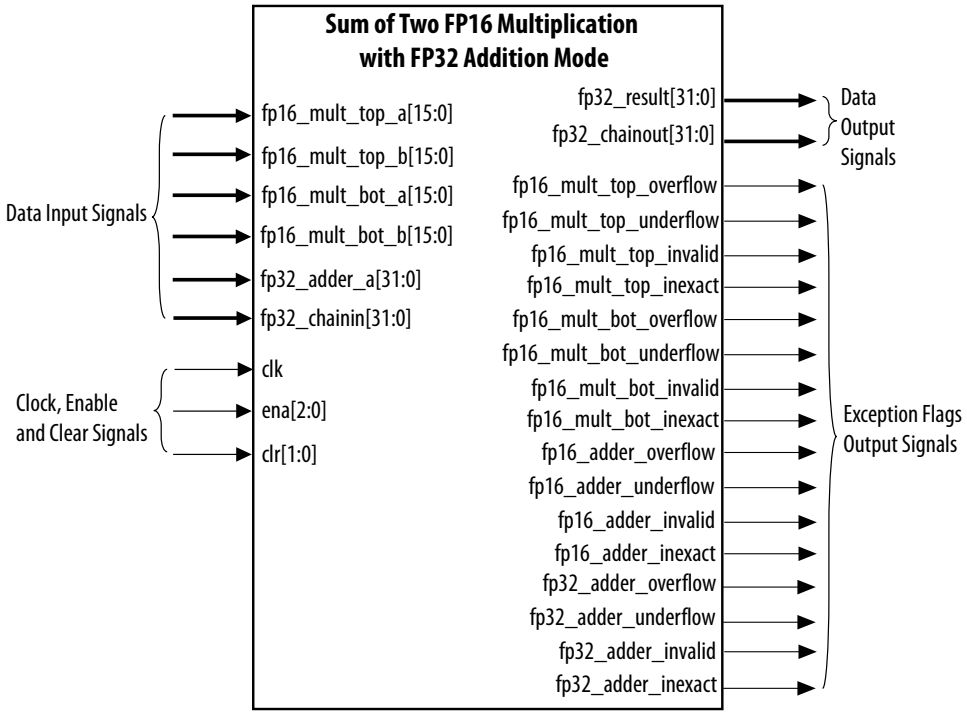


Table 129. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
fp32_adder_a[31:0]	Input	32	Low	Input data bus to the FP32 adder.
fp16_mult_top_a[15:0]	Input	16	Low	Input data bus to the top FP16 multiplier.

continued...

Signal Name	Type	Width	Default	Description
fp16_mult_top_b[15:0]	Input	16	Low	Input data bus to the top FP16 multiplier.
fp16_mult_bot_a[15:0]	Input	16	Low	Input data bus to the bottom FP16 multiplier.
fp16_mult_bot_b[16:0]	Input	16	Low	Input data bus to the bottom FP16 multiplier.
fp32_chainin[31:0]	Input	32	Low	Connect these signals to the chainout signals from the preceding floating-point DSP IP core.
fp32_result[31:0]	Output	32	—	Output data bus from IP core.
fp32_chainout[31:0]	Output	32	—	Connect these signals to the chainin signals of the next floating-point DSP IP core.

Table 130. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 131. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp16_mult_top_overflow/ fp16_mult_bot_overflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp16_mult_top_underflow/ fp16_mult_bot_underflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_mult_top_inexact/ fp16_mult_bot_inexact	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.

continued...

Signal Name	Type	Width	Default	Description
fp16_mult_top_invalid/ fp16_mult_bot_invalid	Output	1	—	This signal indicates if the top/bottom fp16 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp16_adder_overflow/ fp32_adder_overflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp16_adder_underflow/ fp32_adder_underflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_adder_inexact/ fp32_adder_inexact	Output	1	—	This signal indicates if the FP16/FP32 adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_adder_invalid/ fp32_adder_invalid	Output	1	—	This signal indicates if the FP16/FP32 adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.8. Sum of Two FP16 Multiplication with Accumulation Mode Signals

Figure 79. Sum of Two FP16 Multiplication with Accumulation Mode Signals

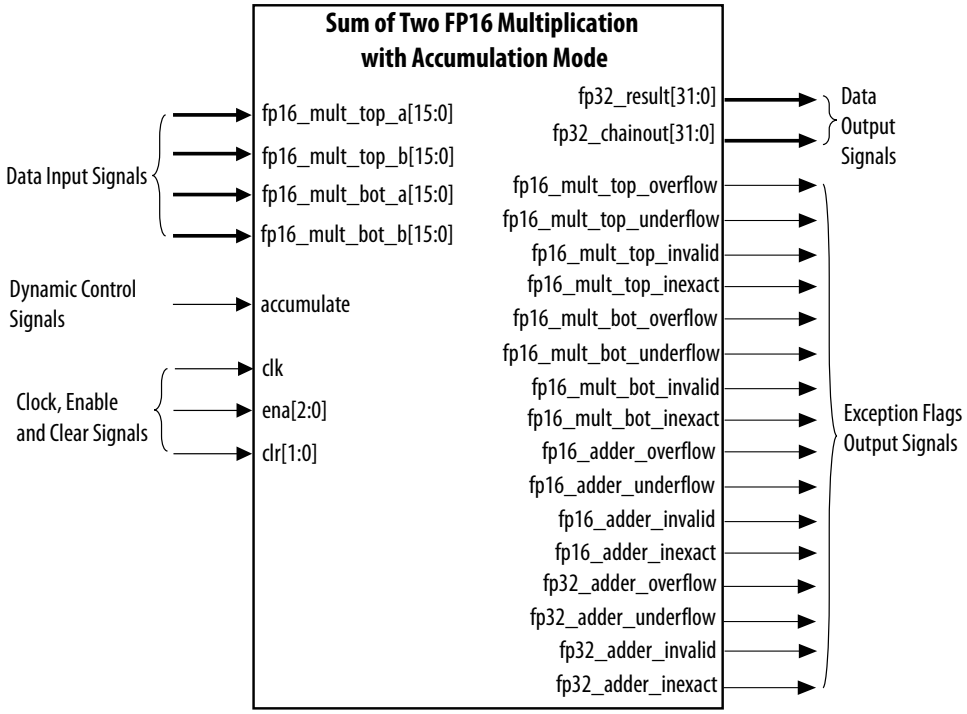


Table 132. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
`fp16_mult_top_a[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_top_b[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_bot_a[15:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp16_mult_bot_b[16:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp32_result[31:0]`	Output	32	—	Output data bus from IP core.
`fp32_chainout[31:0]`	Output	32	—	Connect these signals to the `chainin` signals of the next floating-point DSP IP core.

Table 133. Dynamic Control Signal

Signal Name	Type	Width	Default	Description
`accumulate`	Input	1	Low	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 1: Enable feedback the adder's output. • 0: Disable the feedback mechanism.

Table 134. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 135. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp16_mult_top_overflow/ fp16_mult_bot_overflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp16_mult_top_underflow/ fp16_mult_bot_underflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_mult_top_inexact/ fp16_mult_bot_inexact	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_mult_top_invalid/ fp16_mult_bot_invalid	Output	1	—	This signal indicates if the top/bottom fp16 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp16_adder_overflow/ fp32_adder_overflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.

continued...

Signal Name	Type	Width	Default	Description
`fp16_adder_underflow/ fp32_adder_underflow`	Output	1	—	This signal indicates if the FP16/FP32 adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
`fp16_adder_inexact/ fp32_adder_inexact`	Output	1	—	This signal indicates if the FP16/FP32 adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
`fp16_adder_invalid/ fp32_adder_invalid`	Output	1	—	This signal indicates if the FP16/FP32 adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.9. FP16 Vector One and Vector Two Modes Signals

Figure 80. FP16 Vector One and Vector Two Modes Signals

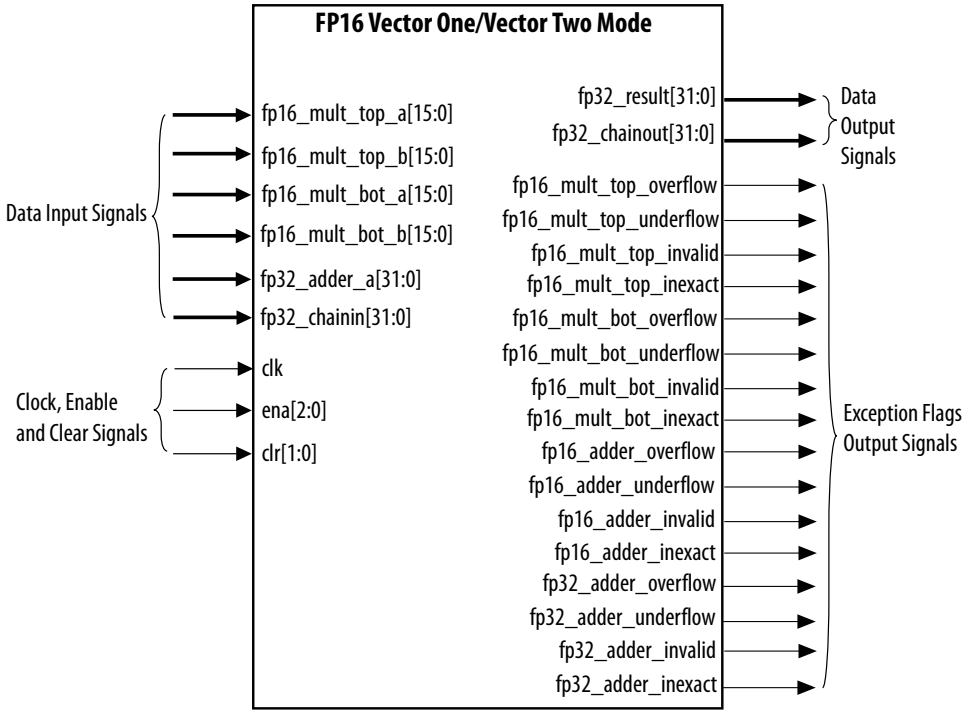


Table 136. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
`fp32_adder_a[31:0]`	Input	32	Low	Input data bus to the FP32 adder.
`fp16_mult_top_a[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_top_b[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_bot_a[15:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp16_mult_bot_b[16:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp32_chainin[31:0]`	Input	32	Low	Connect these signals to the `chainout` signals from the preceding floating-point DSP IP core.
`fp32_result[31:0]`	Output	32	—	Output data bus from IP core.
`fp32_chainout[31:0]`	Output	32	—	Connect these signals to the `chainin` signals of the next floating-point DSP IP core.

Table 137. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
`clk[0]`	Input	1	—	Input clock for all registers.
`ena[2:0]`	Input	3	—	Clock enable signals for all registers.

continued...

Signal Name	Type	Width	Default	Description
				These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 138. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp16_mult_top_overflow/ fp16_mult_bot_overflow	Output	1	—	1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity.This signal indicates if the top/bottom fp16 multiplier result is a larger value compared to the maximum presentable value. This signal indicates if the top/bottom fp160: If the multiplier result is not larger than the maximum presentable value.
fp16_mult_top_underflow/ fp16_mult_bot_underflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_mult_top_inexact/ fp16_mult_bot_inexact	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_mult_top_invalid/ fp16_mult_bot_invalid	Output	1	—	This signal indicates if the top/bottom fp16 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp16_adder_overflow/ fp32_adder_overflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp16_adder_underflow/ fp32_adder_underflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a smaller value compared to the minimum presentable value.

continued...

Signal Name	Type	Width	Default	Description
				1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_adder_inexact/ fp32_adder_inexact	Output	1	—	This signal indicates if the FP16/FP32 adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_adder_invalid/ fp32_adder_invalid	Output	1	—	This signal indicates if the FP16/FP32 adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.4.10. FP16 Vector Three Mode Signals

Figure 81. FP16 Vector Three Mode Signals

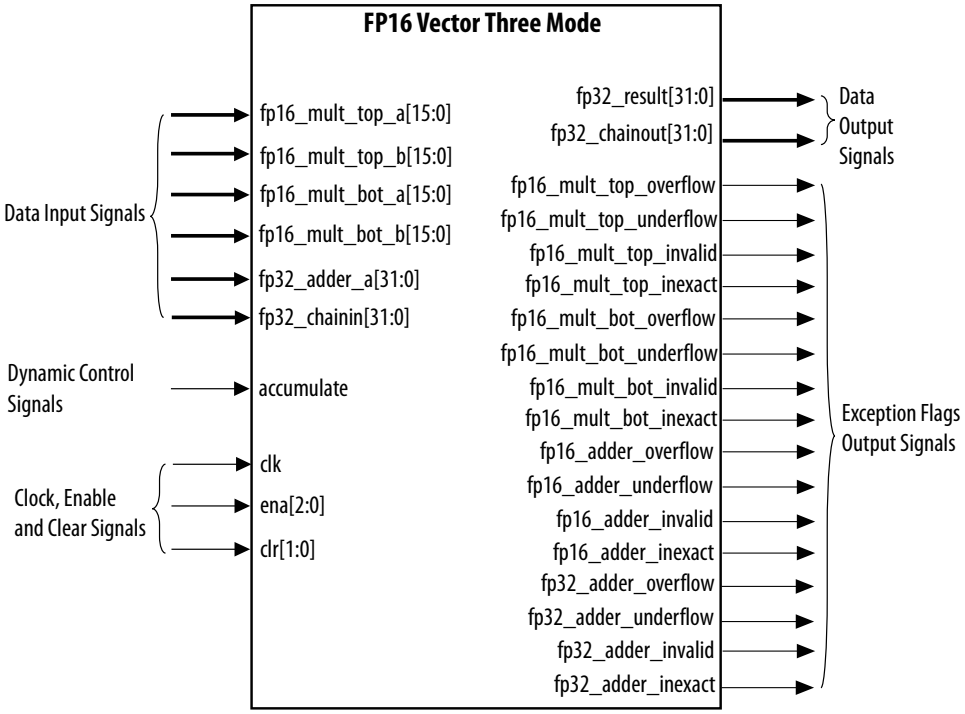


Table 139. Data Input and Output Signals

Signal Name	Type	Width	Default	Description
`fp32_adder_a[31:0]`	Input	32	Low	Input data bus to the FP32 adder.
`fp16_mult_top_a[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_top_b[15:0]`	Input	16	Low	Input data bus to the top FP16 multiplier.
`fp16_mult_bot_a[15:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp16_mult_bot_b[16:0]`	Input	16	Low	Input data bus to the bottom FP16 multiplier.
`fp32_chainin[31:0]`	Input	32	Low	Connect these signals to the `chainout` signals from the preceding floating-point DSP IP core.
`fp32_result[31:0]`	Output	32	—	Output data bus from IP core.
`fp32_chainout[31:0]`	Output	32	—	Connect these signals to the `chainin` signals of the next floating-point DSP IP core.

Table 140. Dynamic Control Signal

Signal Name	Type	Width	Default	Description
`accumulate`	Input	1	Low	Input signal to enable or disable the accumulator feature. You can change the value of this signal during run-time. • 1: Enable feedback the adder's output. • 0: Disable the feedback mechanism.

Table 141. Clock, Enable, and Clear Signals

Signal Name	Type	Width	Default	Description
clk[0]	Input	1	—	Input clock for all registers.
ena[2:0]	Input	3	—	Clock enable signals for all registers. These signals are active-High.
clr[1:0]	Input	2	Low	These signals can be asynchronous or synchronous clear input signals for all registers. You may select the type of clear input signal using **Type of clear signal** parameter. These signals are active-High. For more information about clock enable restrictions for input registers, refer to the related information.

Table 142. Exception Flag Signals

Signal Name	Type	Width	Default	Description
fp16_mult_top_overflow/ fp16_mult_bot_overflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a larger value compared to the maximum presentable value. 1: If the multiplier result is a larger value compared to the maximum representable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.
fp16_mult_top_underflow/ fp16_mult_bot_underflow	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_mult_top_inexact/ fp16_mult_bot_inexact	Output	1	—	This signal indicates if the top/bottom fp16 multiplier result is an exact representation. 1: If the multiplier result is: • a rounded value or • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_mult_top_invalid/ fp16_mult_bot_invalid	Output	1	—	This signal indicates if the top/bottom fp16 multiplier operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.
fp16_adder_overflow/ fp32_adder_overflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a larger value compared to the maximum representable value. 1: If the adder result is a larger value compared to the maximum presentable value and the result is cast to infinity. 0: If the multiplier result is not larger than the maximum presentable value.

continued...

Signal Name	Type	Width	Default	Description
fp16_adder_underflow/ fp32_adder_underflow	Output	1	—	This signal indicates if the FP16/FP32 adder result is a smaller value compared to the minimum presentable value. 1: If the multiplier result is a smaller value compared to the minimum representable value and the result is flushed to zero. 0: If the multiplier result is a larger than the minimum representable value.
fp16_adder_inexact/ fp32_adder_inexact	Output	1	—	This signal indicates if the FP16/FP32 adder result is an exact representation. 1: If the adder result is: • a rounded value • a smaller value compared to the minimum representable value or • a larger value compared to the maximum representable value. 0: If the multiplier result does not meet any of the criteria above.
fp16_adder_invalid/ fp32_adder_invalid	Output	1	—	This signal indicates if the FP16/FP32 adder operation is ill-defined and produces an invalid result. 1: If the multiplier result is invalid and cast to qNaN. 0: If the multiplier result is not an invalid number.

Related Information

Configurations for Input, Pipeline, and Output Registers on page 70
Provides more information about clock enable restrictions for input registers.

10.5. IP Migration

The Native Floating Point DSP Intel Agilex Intel FPGA IP does not support family migration.

11. Intel Agilex 7 Variable Precision DSP Blocks User Guide Archives

For the latest and previous versions of this user guide, refer to Intel Agilex® 7 Variable Precision DSP Blocks User Guide. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

12. Document Revision History for the Intel Agilex 7 Variable Precision DSP Blocks User Guide

Document Version	Intel Quartus Prime Version	Changes
2023.10.02	23.3	• Added *IP Migration* topics in the following sections: — *Native Fixed Point DSP Intel Agilex FPGA IP Core References* — *Native Floating Point DSP Intel Agilex FPGA IP References* • Added description about the family independent IP in the following topics: — *Multiply Adder Intel FPGA IP Core References* — *ALTMULT_COMPLEX Intel FPGA IP Reference* — *LPM_MULT Intel FPGA IP References* • Made minor editorial edits throughout the document.
2023.04.11	23.1	• Updated product family name to Intel Agilex 7.
2022.11.17	21.2	Corrected legend 7 in the *Location of Pipeline Register for FP32 Operation Modes* figure from "fp32_mult_b_clken" to "mult_pipeline_clken".
2021.08.13	21.2	• Added the *DSP Block Cascade Limit in Intel Agilex™ Devices* topic. • Removed the statements about the number of DSP blocks you can cascade as systolic FIR structure in the following topics: — *18-bit Systolic FIR Mode* — *27-Bit Systolic FIR Mode* • In the *27-Bit Systolic FIR Mode* topic, removed the "Systolic registers are not required in this mode" statement. The registers are not available in the 27-bit systolic FIR mode. • Fixed broken links and updated related information sections throughout the document.
2021.02.05	20.3	Clarified that bfloat16 is Brain Floating Point in the *Features* topic.
2020.09.28	20.3	• Updated the supported registers for 18 × 19 systolic mode in the *Supported Register Configurations per Operation Modes* table. • Updated guidelines when using systolic register in fixed-point arithmetic in the *Systolic Register for Fixed-point Arithmetic* topic.
2020.04.26	20.1	• Updated values for **Which multiplier implementation should be used?** parameter for the LPM_MULT IP core.
2020.04.13	20.1	• Removed chainin output feature from footnote (5) in the *Supported Register Configurations per Operation Modes* table.
2019.09.30	19.3	• Clarified that input and output registers for fixed-point arithmetic are not reset after power up and users need to clear the registers manually before starting an operation. • Updated equation for the following operation modes: — FP32 Multiplication with Accumulation — Sum of Two FP16 Multiplication with Accumulation • Updated the *Supported Register Configurations per Operation Modes* table in the *Configurations for Input, Pipeline, and Output Registers* topic for fixed-point arithmetic.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Added information for Native Fixed Point DSP Intel Agilex FPGA IP version 19.1.1. • Added information for Native Floating Point DSP Intel Agilex FPGA IP version 19.1.0 • Added information for ALTMULT_COMPLEX Intel FPGA IP version 19.1.0 • Added information for LPM_DIVIDE Intel FPGA IP version 19.1. • Added information for LPM_MULT Intel FPGA IP version 19.1.0 • Added information for Multiply Adder Intel FPGA IP version 19.1.0
2019.04.02	19.1	Initial release.