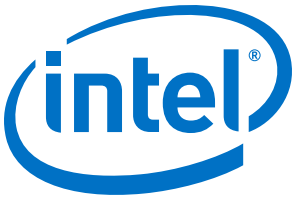
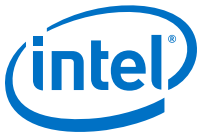


Intel® Agilex™ Variable Precision DSP Blocks User Guide

UG-20213 | 2019.04.02
Latest document on the web: PDF | HTML

Contents



1. Intel® Agilex™ Variable Precision DSP Blocks Overview

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

1.1. Features

The Intel Agilex 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 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 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

Intel Agilex Device Data Sheet—DSP Block Specifications
Refer to the *Intel Agilex Device Data Sheet* for more information on the DSP block performance.

1.2. Supported Operational Modes in Intel Agilex 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
	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

(1) The Intel Quartus® Prime software will determine 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.

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 x9 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

- Intel Agilex Device Overview—Intel Agilex FPGAs Family Plan
  Refer to the *Intel Agilex FPGAs Family Plan* in the Intel Agilex Device Overview for more information on the variable precision DSP blocks resources.
- Pre-adder for Fixed-Point Arithmetic on page 17
- Internal Coefficient for Fixed-Point Arithmetic on page 17
- Accumulator, Chainout Adder, and Preload Constant for Fixed-Point Arithmetic on page 18
- Input Cascade for Fixed-Point Arithmetic on page 64
- Intel Agilex Variable Precision DSP Blocks Design Considerations on page 62

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
	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 Device Overview—Intel Agilex FPGAs Family Plan
  Refer to the *Intel Agilex FPGAs Family Plan* in the Intel Agilex Device Overview for more information on the variable precision DSP blocks resources.
- Operational Modes for Floating-Point Arithmetic on page 40
- Chainout Adder on page 72

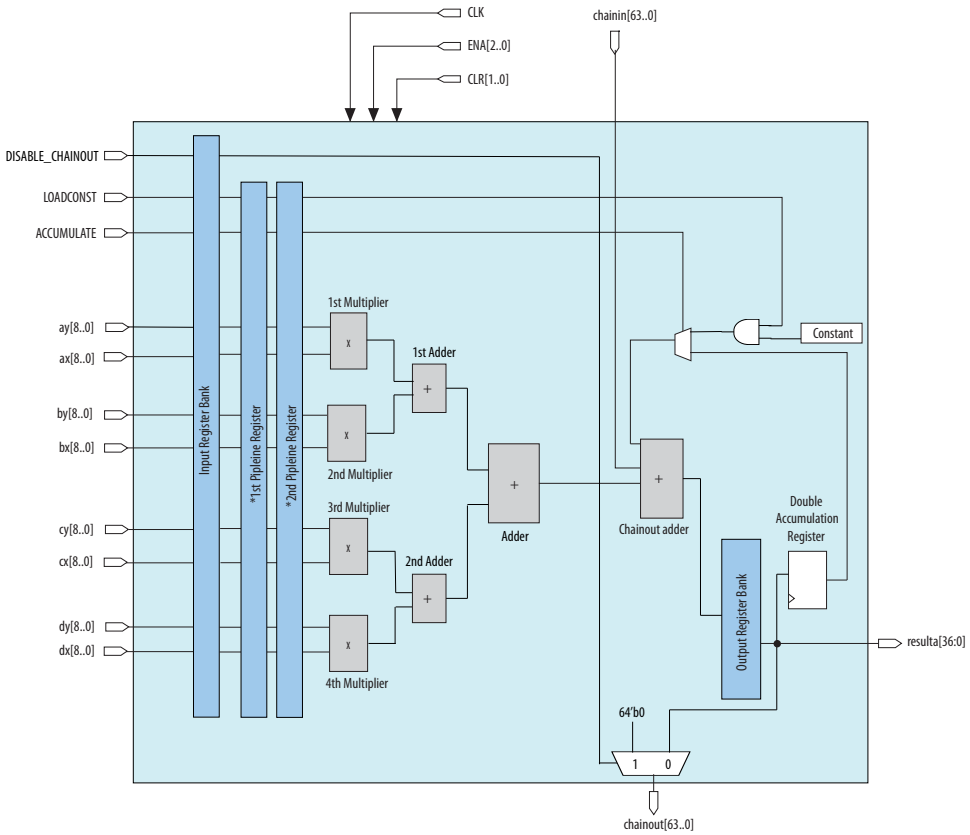
2. Intel Agilex Variable Precision DSP Blocks Architecture

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

Table 5. Block Architecture

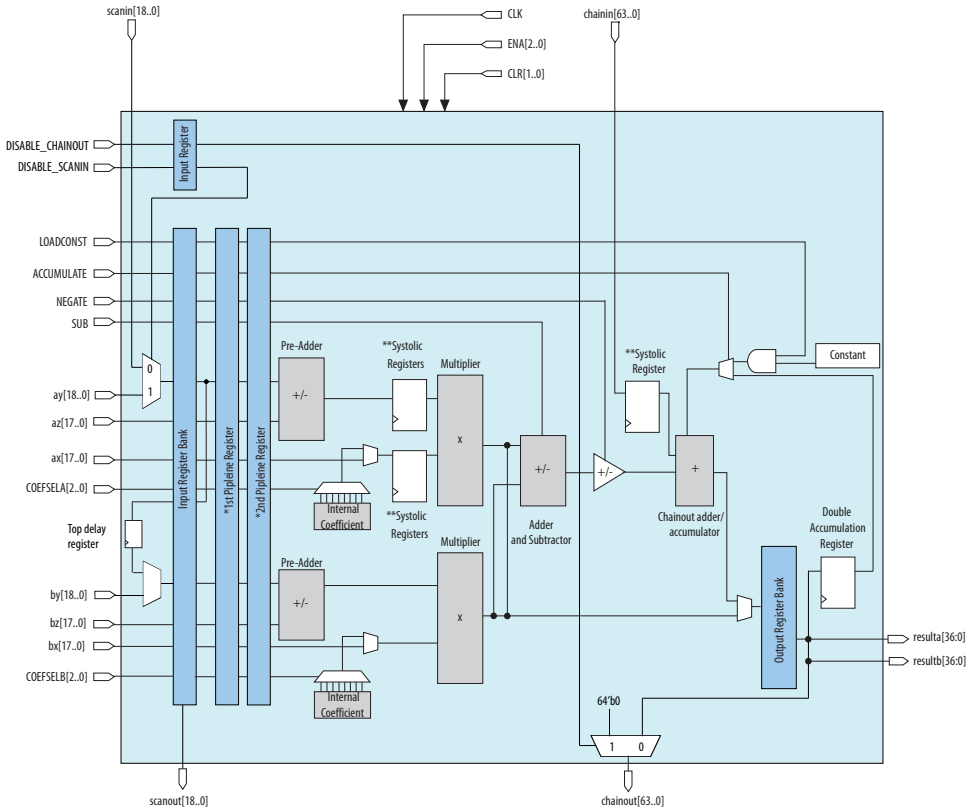
DSP Implementations	Block Architecture
Fixed-Point Arithmetic	• Input register bank • First and second pipeline registers • Pre-adder/subtract • Internal coefficient • Multipliers • Adder and Subtractor • Accumulator, chainout adder, and Preload Constant • Systolic registers • Double accumulation register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



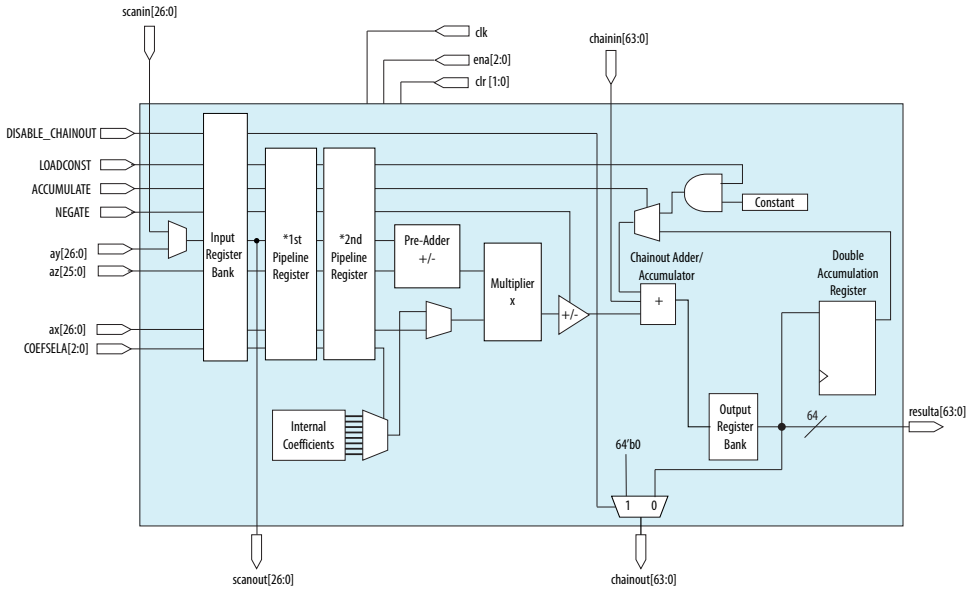
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

Figure 2. Fixed-Point Arithmetic 18 x 19 Mode



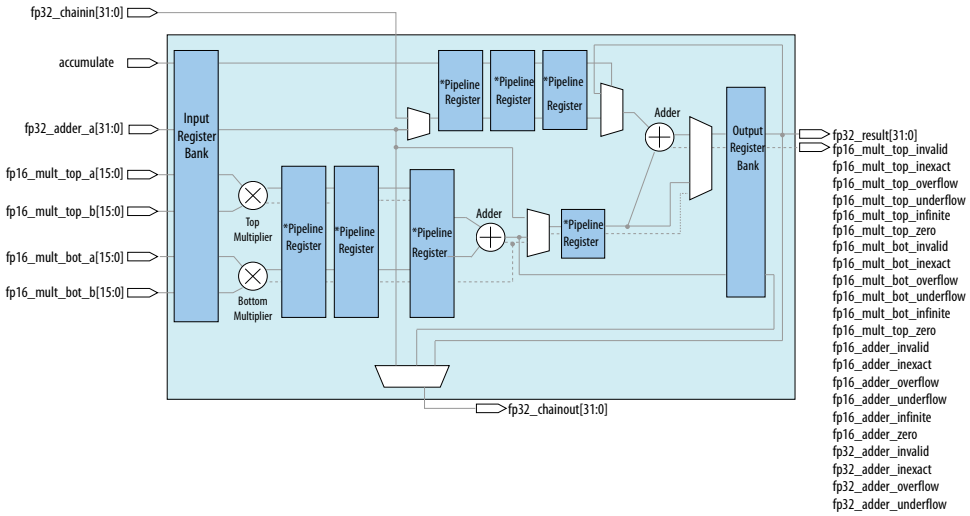
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.
**Systolic registers are enabled in systolic mode only.

Figure 3. Fixed-Point Arithmetic 27 x 27 Mode



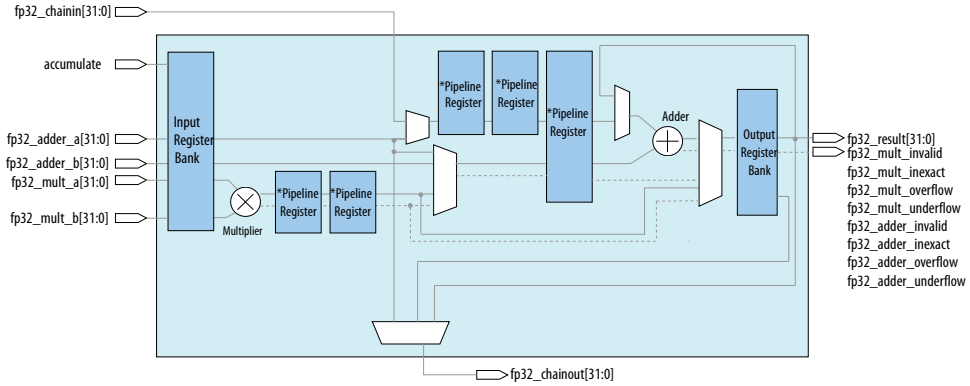
*This block diagram shows the functional representation of the DSP block. The pipeline registers are embedded within the various circuits of the DSP block.

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 and cleared on power up. 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

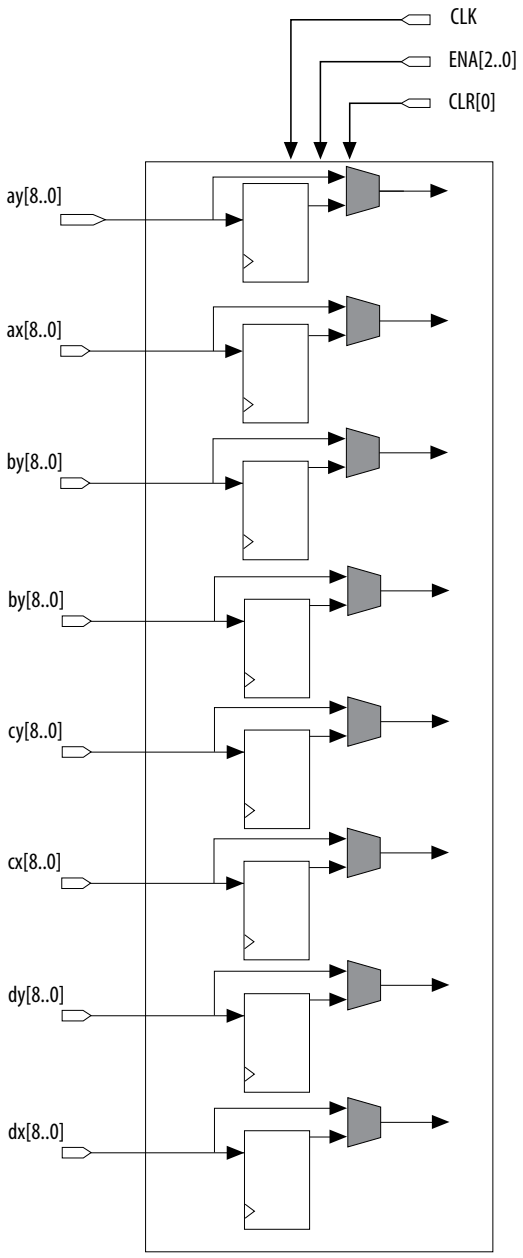


Figure 7. Data Input Registers in Fixed-Point Arithmetic 18 x 19 Mode

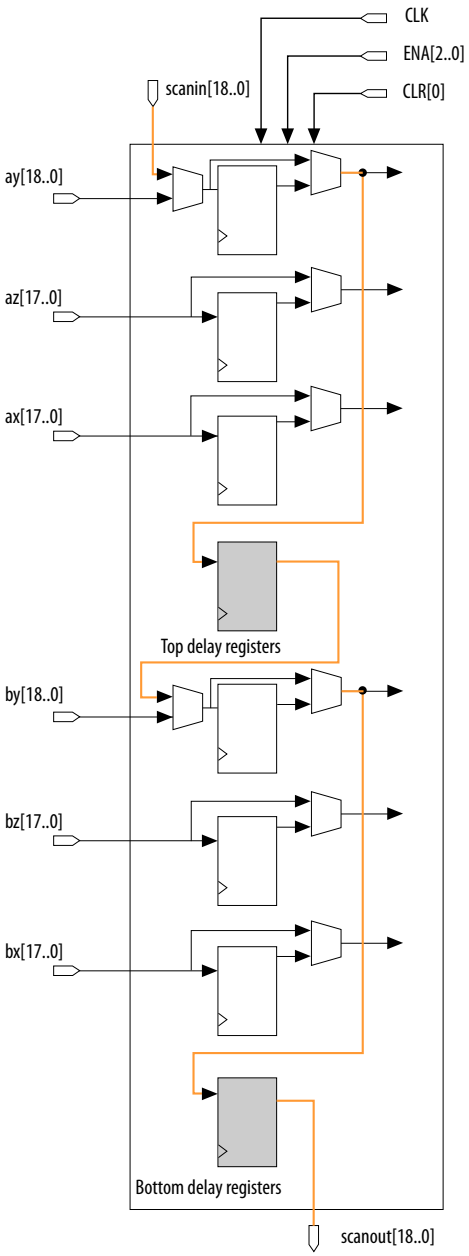
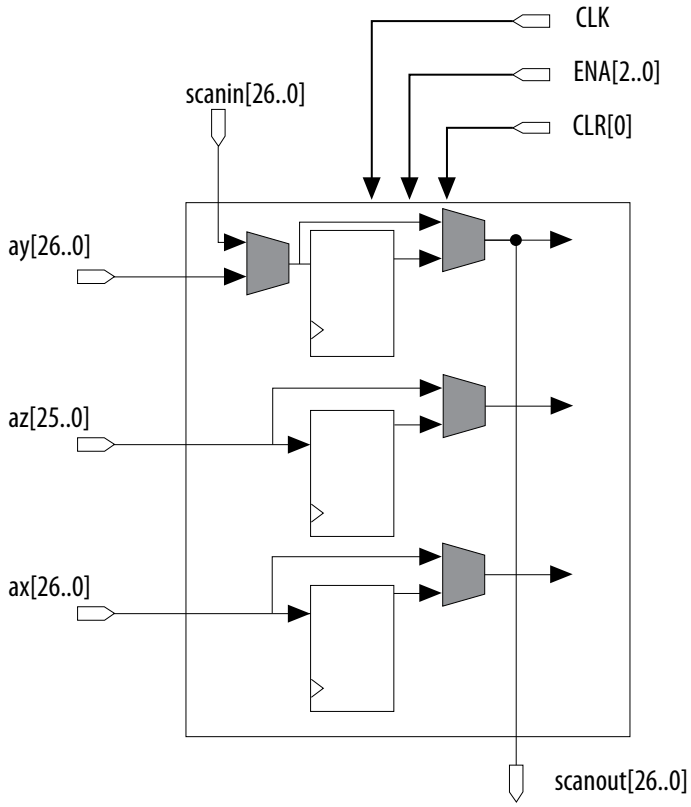


Figure 8. Data Input Registers in Fixed-Point Arithmetic 27 x 27 Mode



Related Information

Configurations for Input, Pipeline, and Output Registers on page 62
Provides information about restrictions on fixed-point arithmetic input 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 62
Provides information about restrictions on fixed-point arithmetic pipeline 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 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, C0EFSELA/C0EFSELB 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 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 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

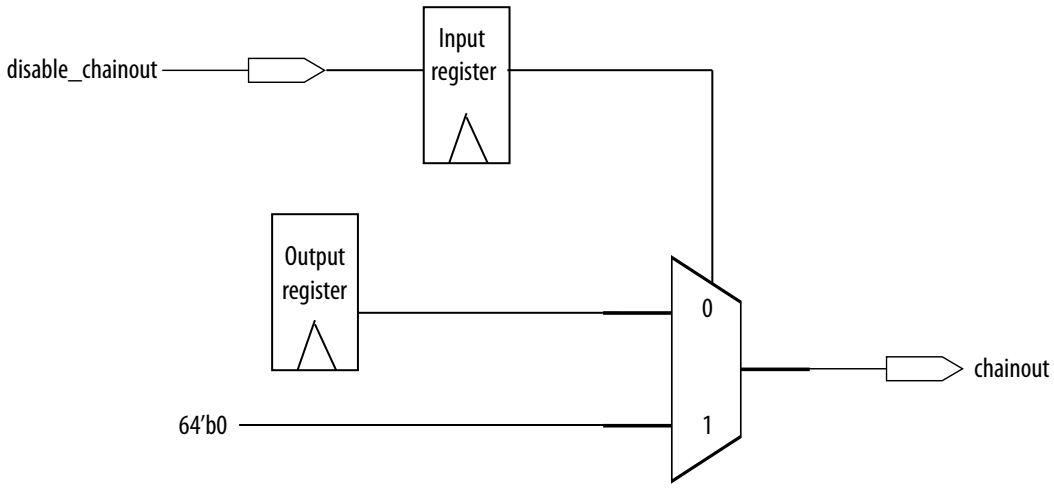


Table 8. DISABLE_CHAINOUT Signal Behavior

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 input and 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 and is cleared after power up.

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 62
Provides information about restrictions on fixed-point arithmetic 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

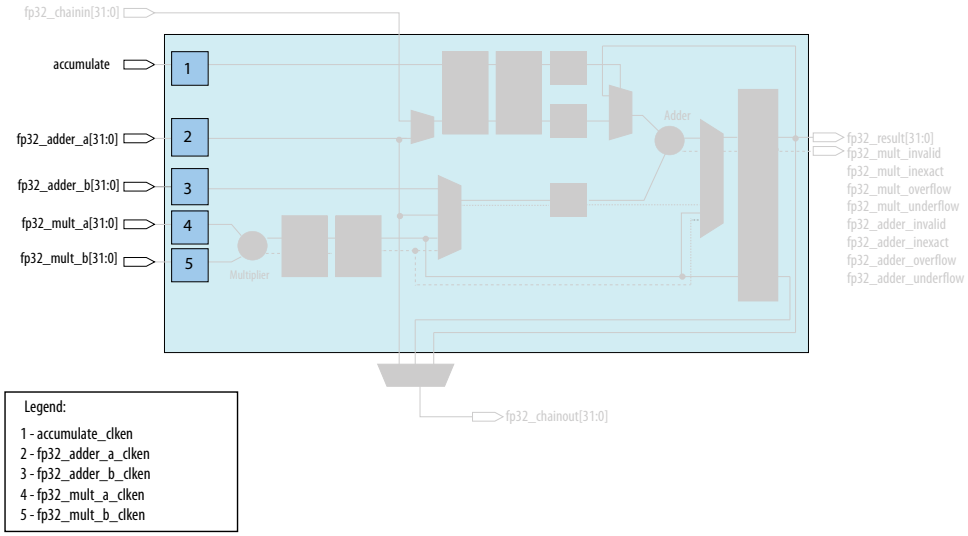
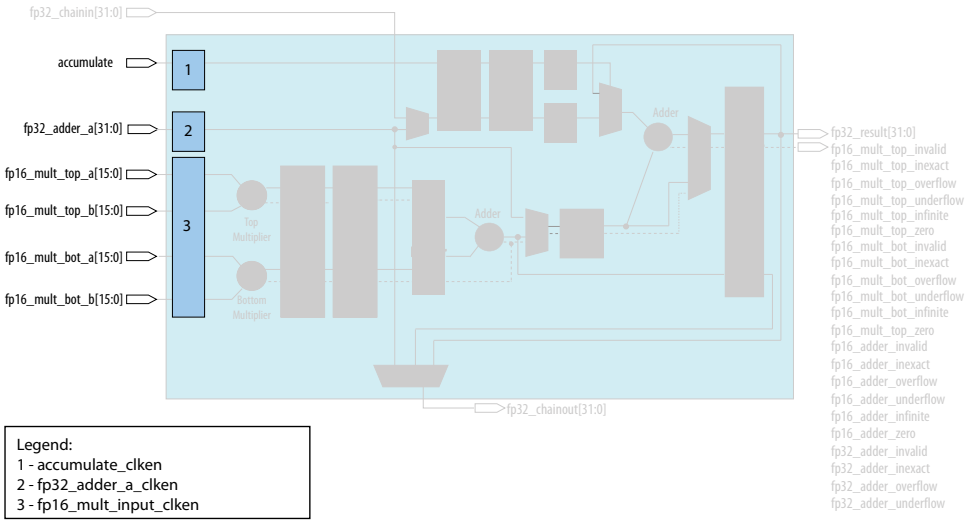


Figure 11. Location of Input Register for FP16 Operation Modes



All the registers in the DSP blocks are positive-edge triggered and cleared on power up. 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]`

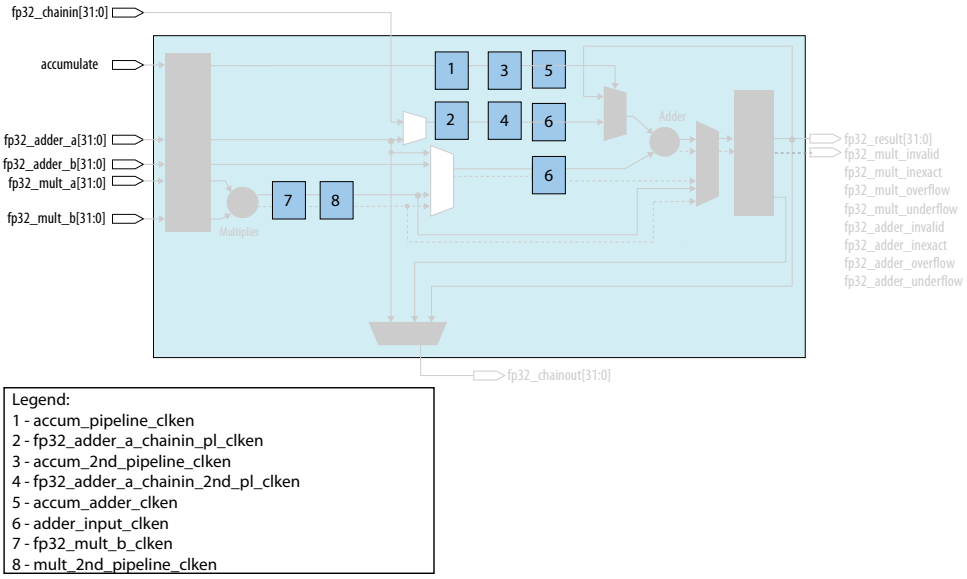
Related Information

Configurations for Input, Pipeline, and Output Registers on page 67
Provides information about restrictions on floating-point arithmetic input registers.

2.2.2. Pipeline Registers for Floating-Point Arithmetic

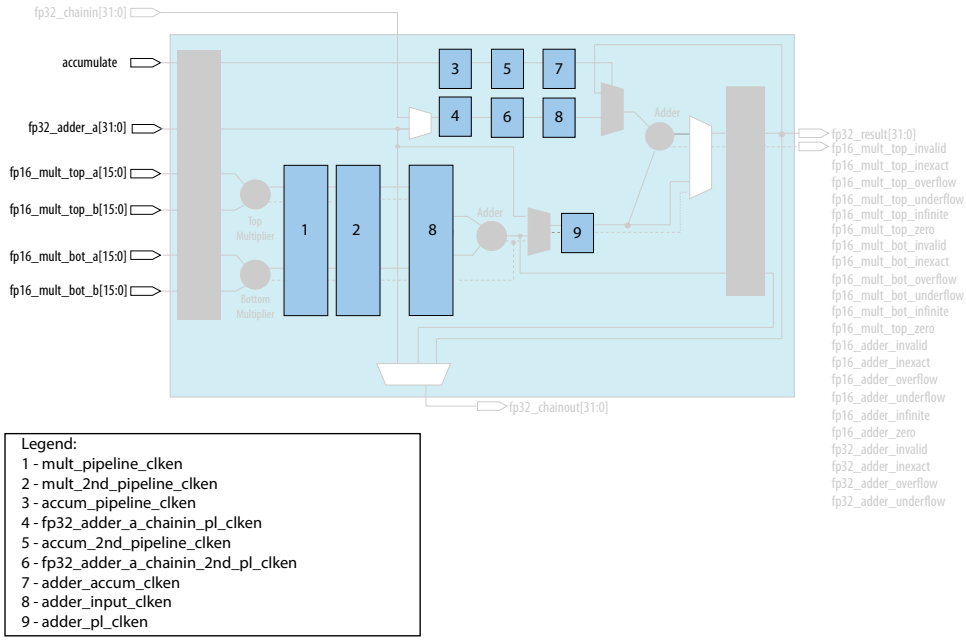
Floating-point arithmetic has 3 latency layers of pipeline registers. You can bypass all latency layers of the pipeline registers or use any one, two or three layers of pipeline registers.

Figure 12. Location of Pipeline Register for FP32 Operation Modes



Legend:
1 - accum_pipeline_clken
2 - fp32_adder_a_chainin_pl_clken
3 - accum_2nd_pipeline_clken
4 - fp32_adder_a_chainin_2nd_pl_clken
5 - accum_adder_clken
6 - adder_input_clken
7 - fp32_mult_b_clken
8 - mult_2nd_pipeline_clken

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 67
Provides information about restrictions on floating-point arithmetic pipeline 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 and is cleared after power up.

Figure 14. Location of Output Register for FP32 Operation Modes

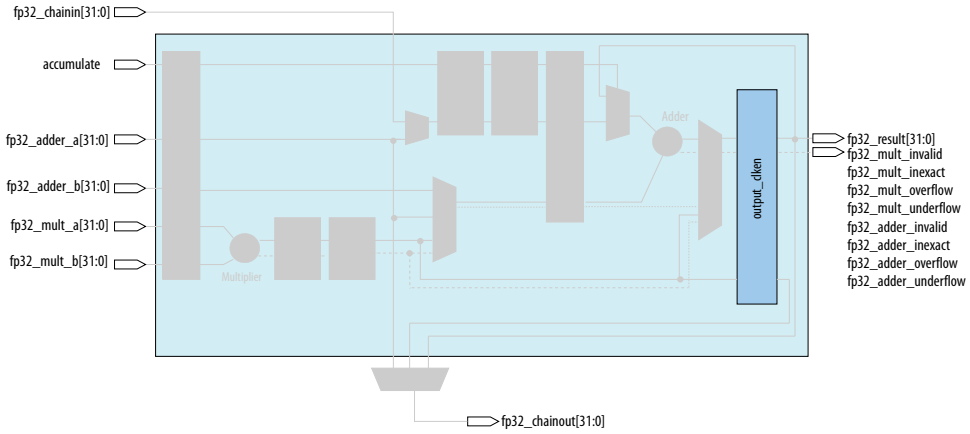
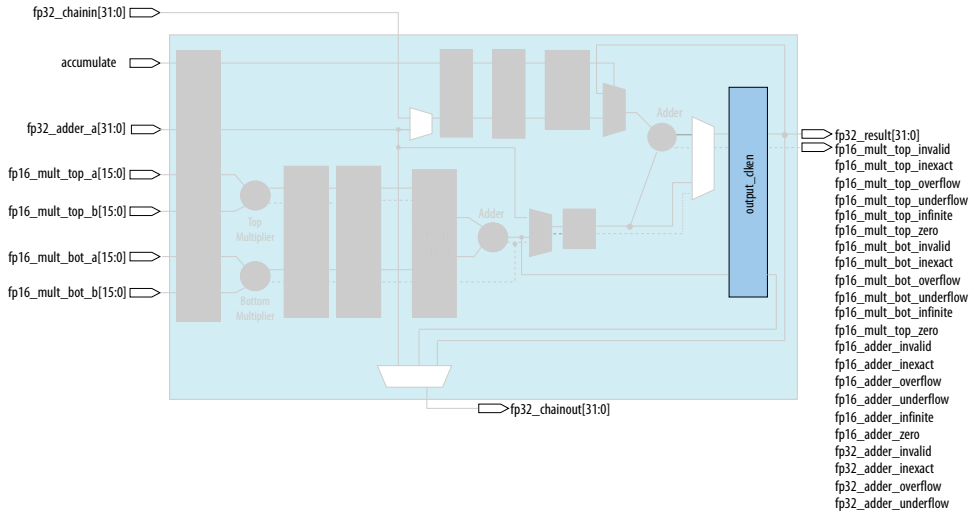


Figure 15. Location of Output Register for FP16 Operation Modes



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 67
Provides information about restrictions on floating-point arithmetic output registers.

2.2.6. Exception Handling for Floating-Point Arithmetic

The Intel Agilex 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		
	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.

continued...

Floating-Point Format	Exception Flags	Width	Description
			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.
	`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.

continued...

Floating-Point Format	Exception Flags	Width	Description
			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
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

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	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 Variable Precision DSP Blocks Operational Modes

This section describes how you can configure the Intel Agilex 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:

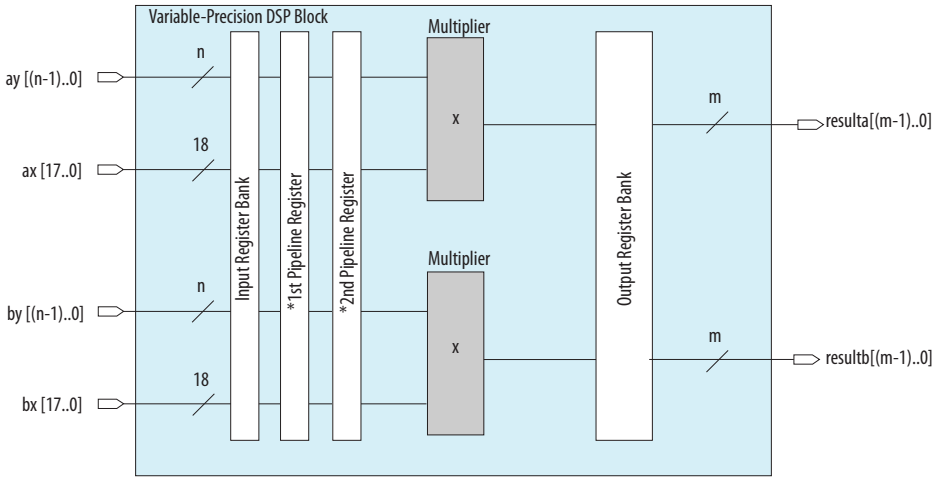
resulta = ax * ay

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



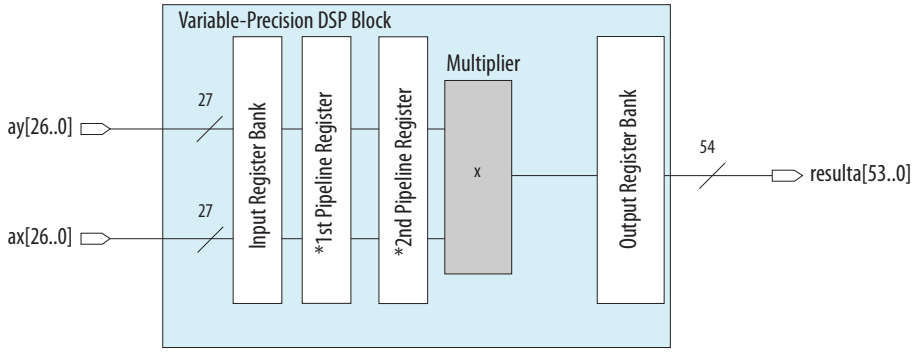
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

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 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

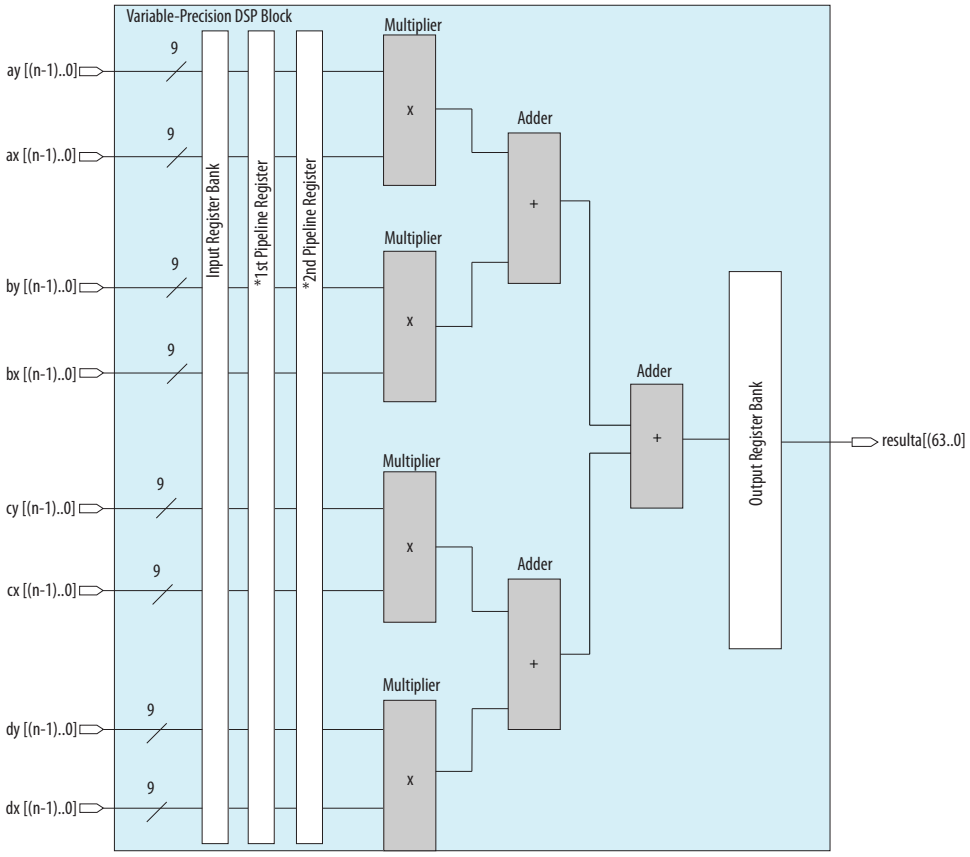
The 8 x 8 (unsigned) or 9 x 9 sum of 4 mode uses the following equations:

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

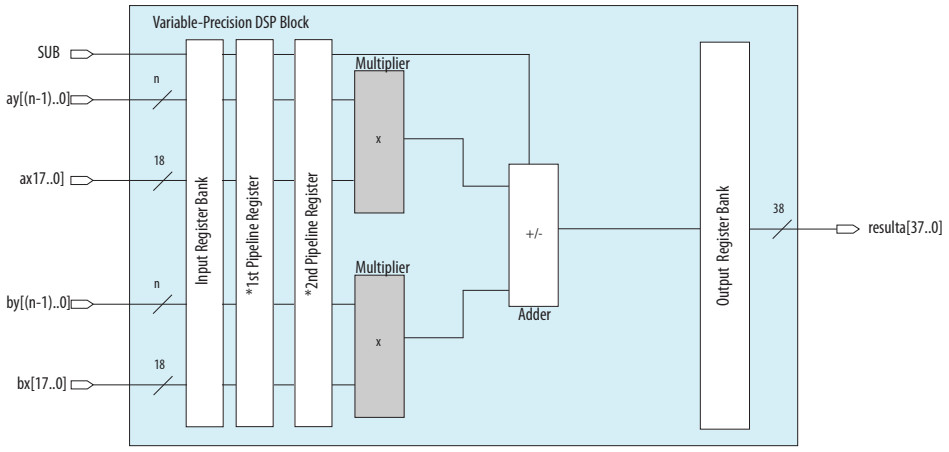
The multiplier adder sum mode uses the equations:

- 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 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

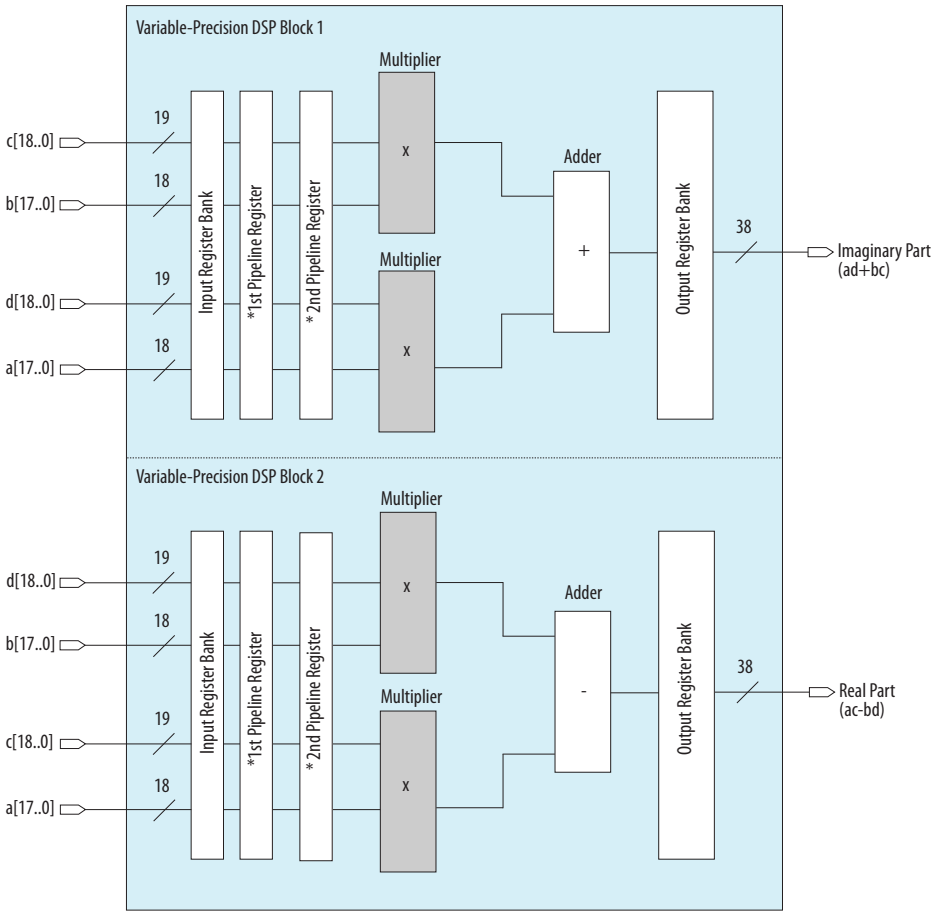
The Intel Agilex devices support the 18 × 19 complex multiplier mode using two fixed-point arithmetic multiplier adder sum mode.

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 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 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.

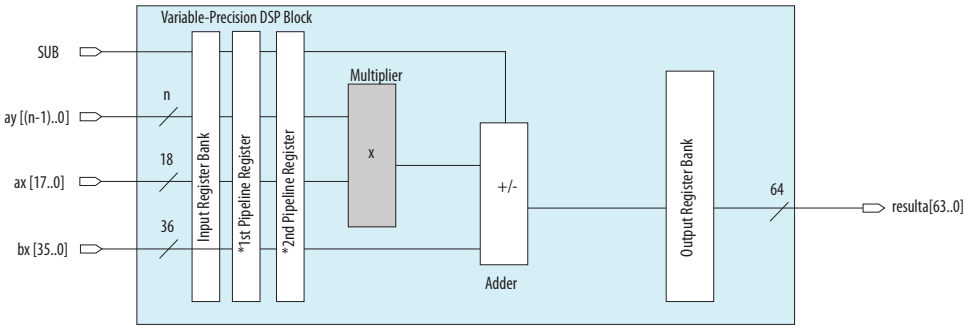
Use the upper multiplier to provide the input for an 18 × 19 multiplication, while the bottom multiplier is bypassed. The `bx[35..0]` signals the 36-bit input operand.

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 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.

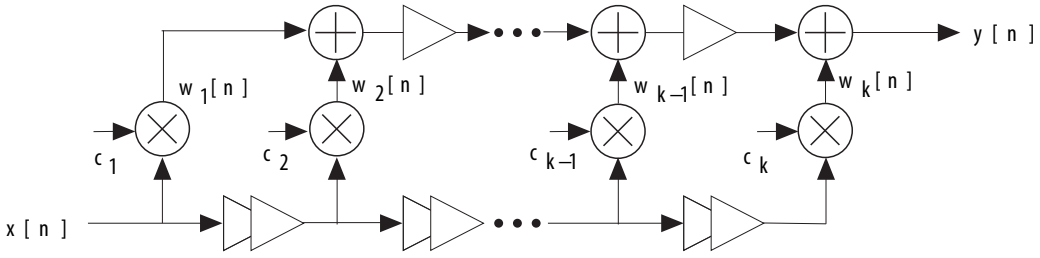
Figure 23. Basic FIR Filter Equation

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

$$\text{Where } i \text{ start from 1,} \qquad w_i[n] = c_i x[n-2i+2]$$

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 variable precision DSP blocks support the following systolic FIR structures:

- 18-bit
- 27-bit

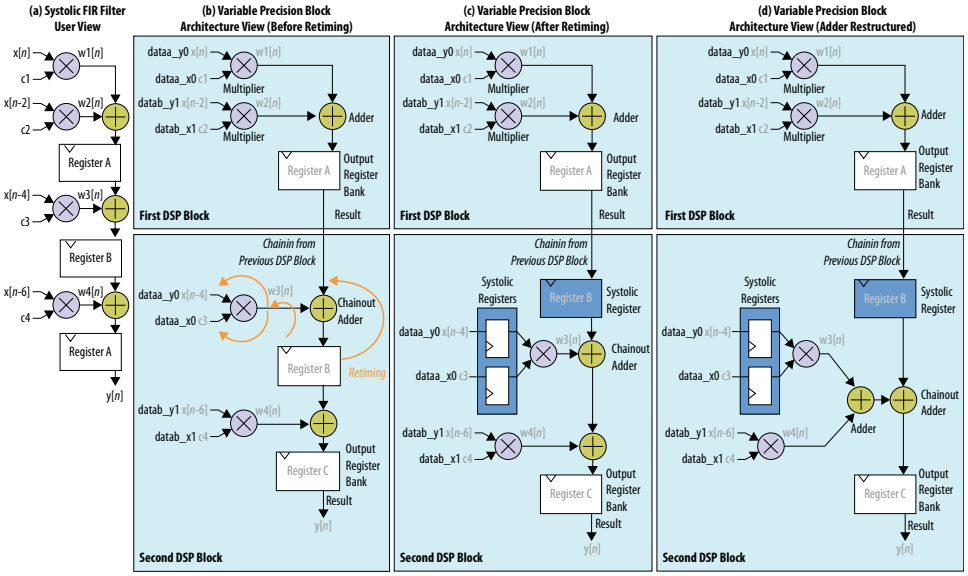
In systolic FIR mode, the input of the multiplier can come from four different sets of sources:

- 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 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).

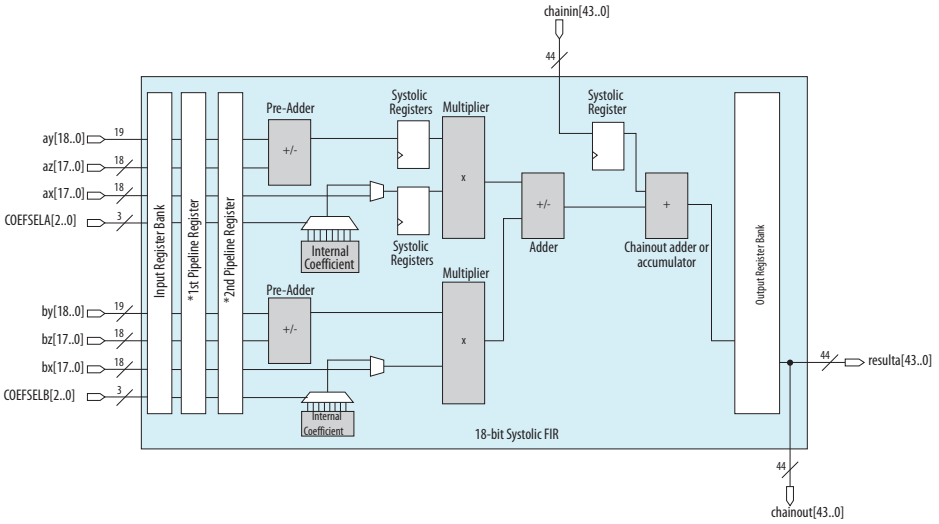
Figure 25. Mapping Systolic Mode User View to Variable Precision Block Architecture View



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. This allows a total sixteen 18 x 19 multipliers or eight Intel Agilex variable precision DSP blocks to be cascaded as systolic FIR structure.

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



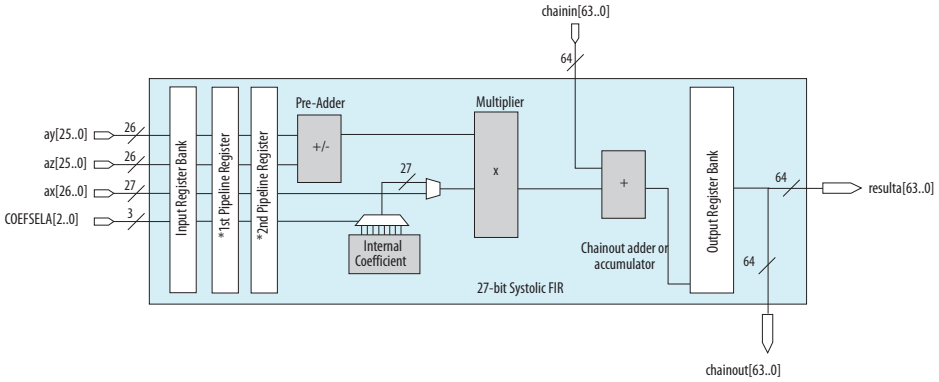
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

3.1.5.3. 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). This allows a total of eleven 27 x 27 multipliers or eleven Intel Agilex variable precision DSP blocks to be cascaded as systolic FIR structure.

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 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. 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

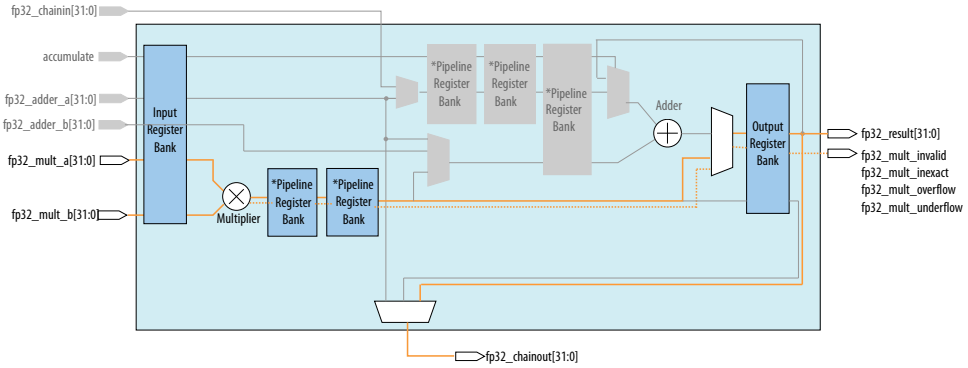
This mode allows you to apply basic floating-point multiplication equation:

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:

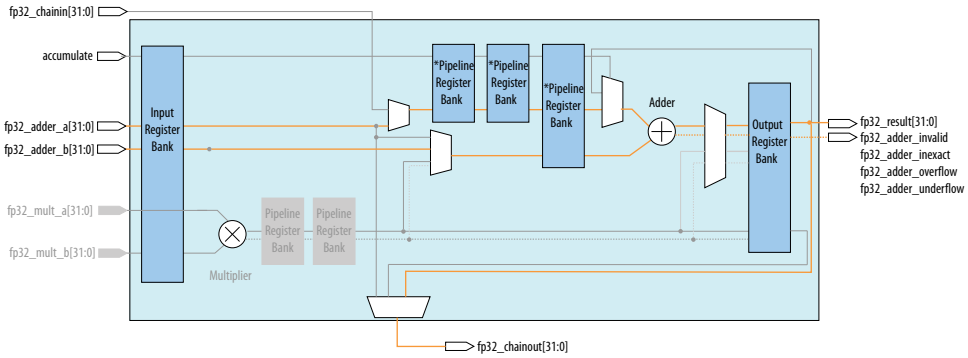
fp32_result = fp32_adder_b+fp32_adder_a

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



*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.

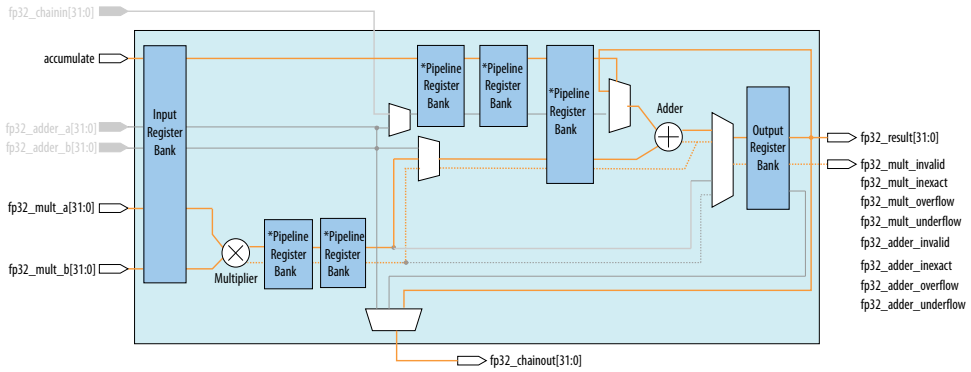
When ACCUMULATE signal is high, this mode uses the equation of fp32_result(t) = [fp32_mult_a(t-1)*fp32_mult_b(t-1)] +/- fp32_result(t-1).

When ACCUMULATE signal is low, this mode uses the equation of fp32_result = (ay*az).

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 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.

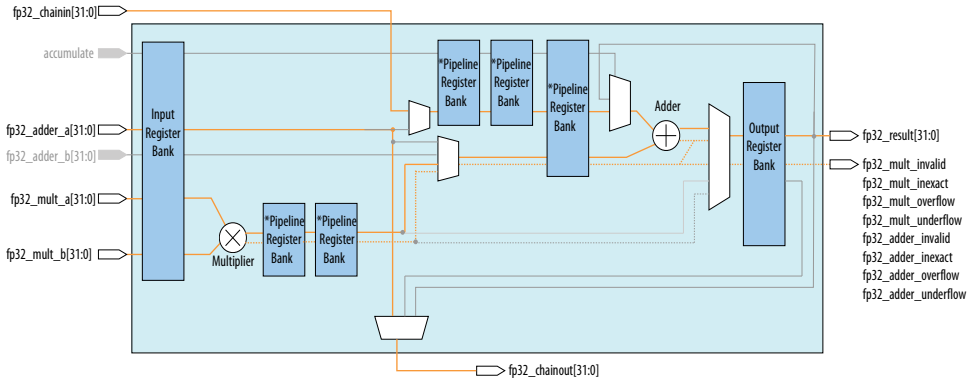
Table 16. Equations Applied to FP32 Vector One Mode

Chainin Parameter	Vector One with Floating-Point Addition	Vector One with Floating-Point Subtraction
Disable	result = `fp32_mult_a * fp32_mult_b` Chainout = `fp32_adder_a`	result = `fp32_mult_a * fp32_mult_b` Chainout = `fp32_adder_a`
Enable	result = (`fp32_mult_a * fp32_mult_b`) + `fp32_chainin` Chainout = `fp32_adder_a`	result = (`fp32_mult_a * fp32_mult_b`) - `fp32_chainin` 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.

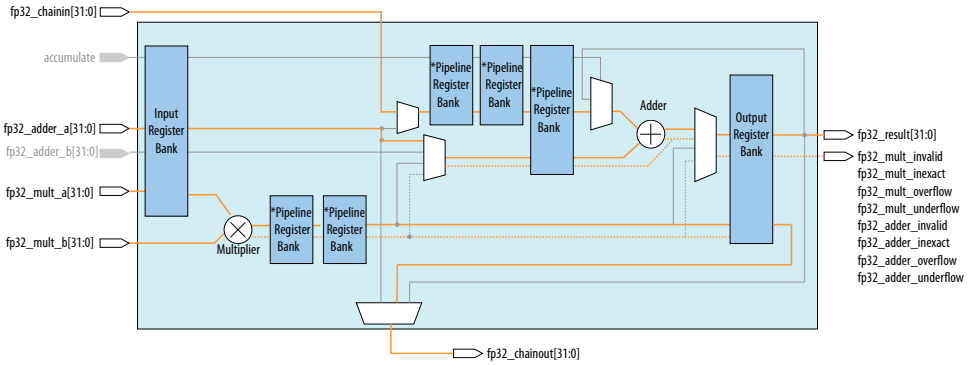
Table 17. Equations Applied to FP32 Vector Two Mode

Chainin Parameter	Vector Two with Floating-Point Addition	Vector Two with Floating-Point Subtraction
Disable	result = `fp32_adder_a` Chainout = `fp32_mult_a * fp32_mult_b`	result = `fp32_adder_a` Chainout = `fp32_mult_a * fp32_mult_b`
Enable	result = `fp32_adder_a + fp32_chainin` Chainout = `fp32_mult_a * fp32_mult_b`	result = `fp32_adder_a - fp32_chainin` 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:

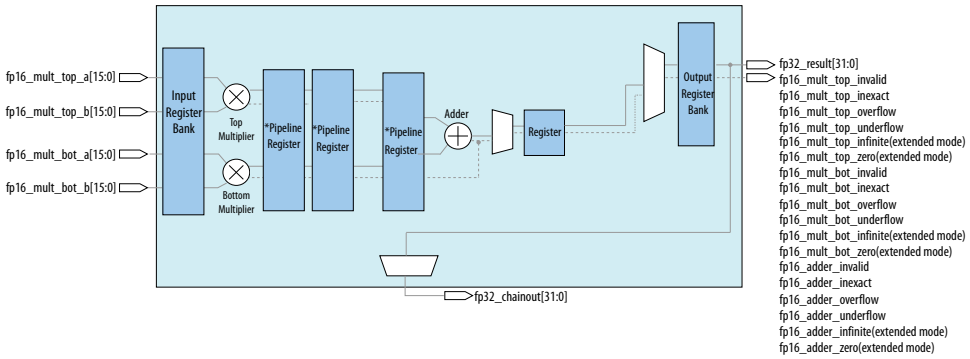
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`

- 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:

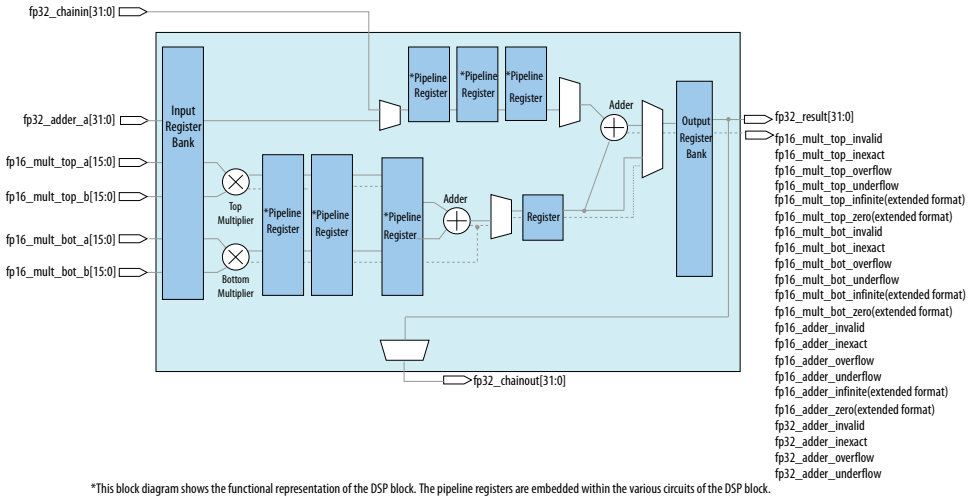
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`
- `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 = [fp16_mult_top_a * fp16_mult_top_b] + [fp16_mult_bot_a * fp16_mult_bot_b] + previous value of fp32_result

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

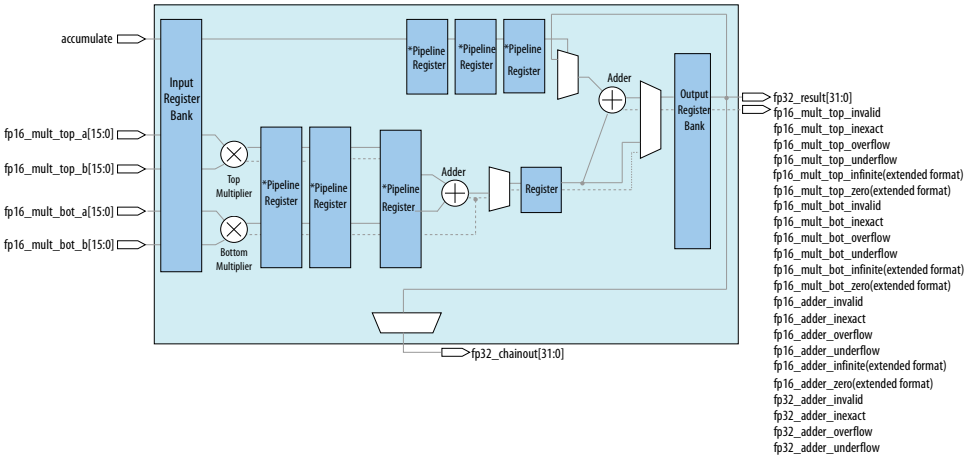
- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_overflow
- fp16_mult_top_underflow
- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_overflow
- fp16_mult_bot_underflow
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_overflow
- fp16_adder_underflow
- fp32_adder_invalid

- 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) Chainout = fp32_adder_a	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) 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 Chainout = fp32_adder_a	fp32_result = (fp16_mult_top_a * fp16_mult_top_b) - (fp16_mult_bot_a * fp16_mult_bot_b) - fp32_chainin Chainout = fp32_adder_a

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

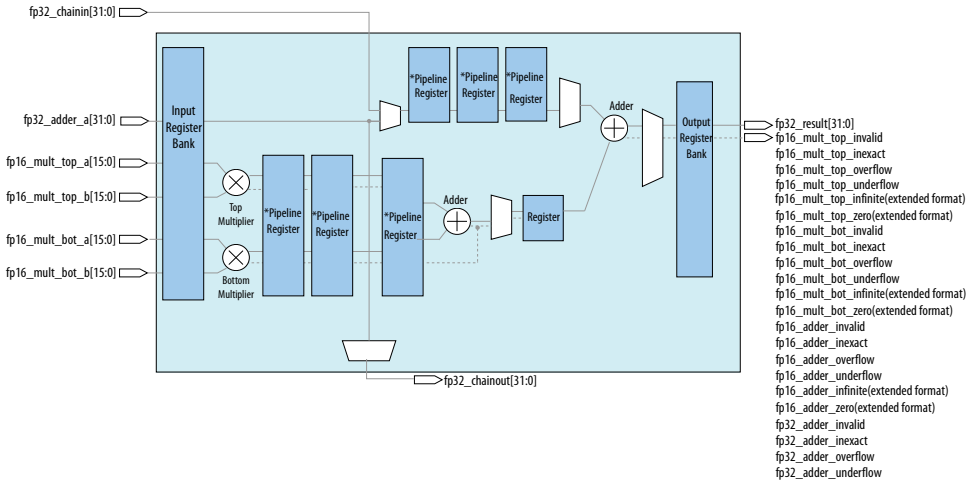
- `fp16_mult_top_invalid`
- `fp16_mult_top_inexact`
- `fp16_mult_top_overflow`
- `fp16_mult_top_underflow`
- `fp16_mult_bot_invalid`
- `fp16_mult_bot_inexact`
- `fp16_mult_bot_overflow`
- `fp16_mult_bot_underflow`
- `fp16_adder_invalid`
- `fp16_adder_inexact`
- `fp16_adder_overflow`
- `fp16_adder_underflow`
- `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_result = fp32_adder_a - fp32_chainin`

continued...

Chainin Parameter	Vector Two with Floating-Point Addition	Vector Two with Floating-Point Subtraction
	fp32_chainout = (fp16_mult_top_a * fp16_mult_top_b) + (fp16_mult_bot_a * fp16_mult_bot_b)	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:

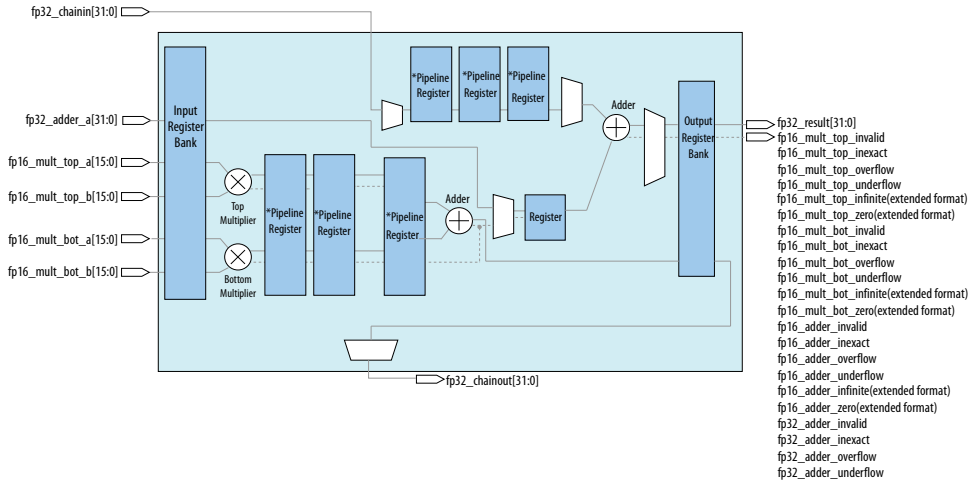
- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_overflow
- fp16_mult_top_underflow
- fp16_mult_bot_invalid
- fp16_mult_bot_inexact
- fp16_mult_bot_overflow
- fp16_mult_bot_underflow
- fp16_adder_invalid
- fp16_adder_inexact
- fp16_adder_overflow
- fp16_adder_underflow
- fp32_adder_invalid
- 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 = 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_result(t-1) fp32_chainout = {fp16_mult_top_a * fp16_mult_top_b} + {fp16_mult_bot_a * fp16_mult_bot_b}	fp32_result = fp32_adder_a - 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:

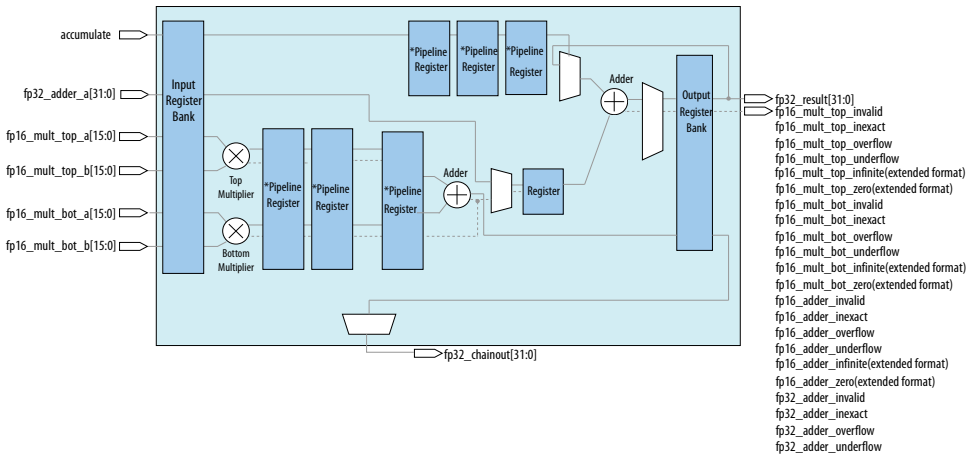
- fp16_mult_top_invalid
- fp16_mult_top_inexact
- fp16_mult_top_overflow
- fp16_mult_top_underflow
- fp16_mult_bot_invalid

- 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

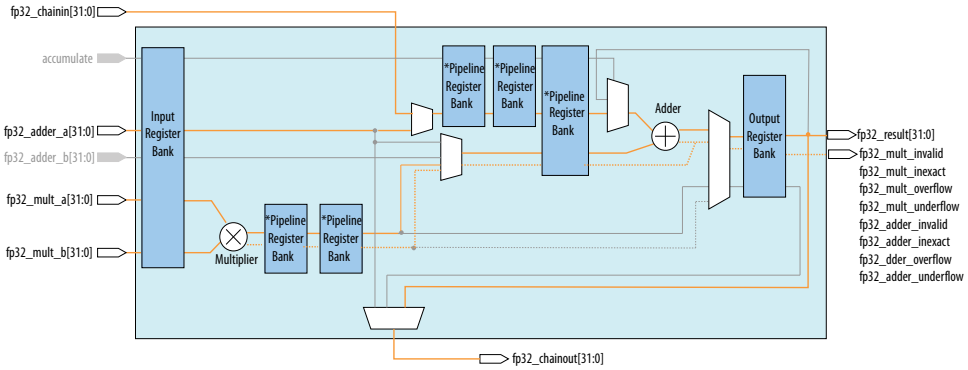
Chainin Parameter	Multiply-Add Mode	Multiply-Subtract Mode
Disable	fp32_result = (fp32_mult_a*fp32_mult_b) + fp32_adder_a	fp32_result = (fp32_mult_a*fp32_mult_b) - fp32_adder_a
Enable	fp32_result = (fp32_mult_a*fp32_mult_b) + fp32_chainin	fp32_result = (fp32_mult_a*fp32_mult_b) - fp32_chainin

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 Devices



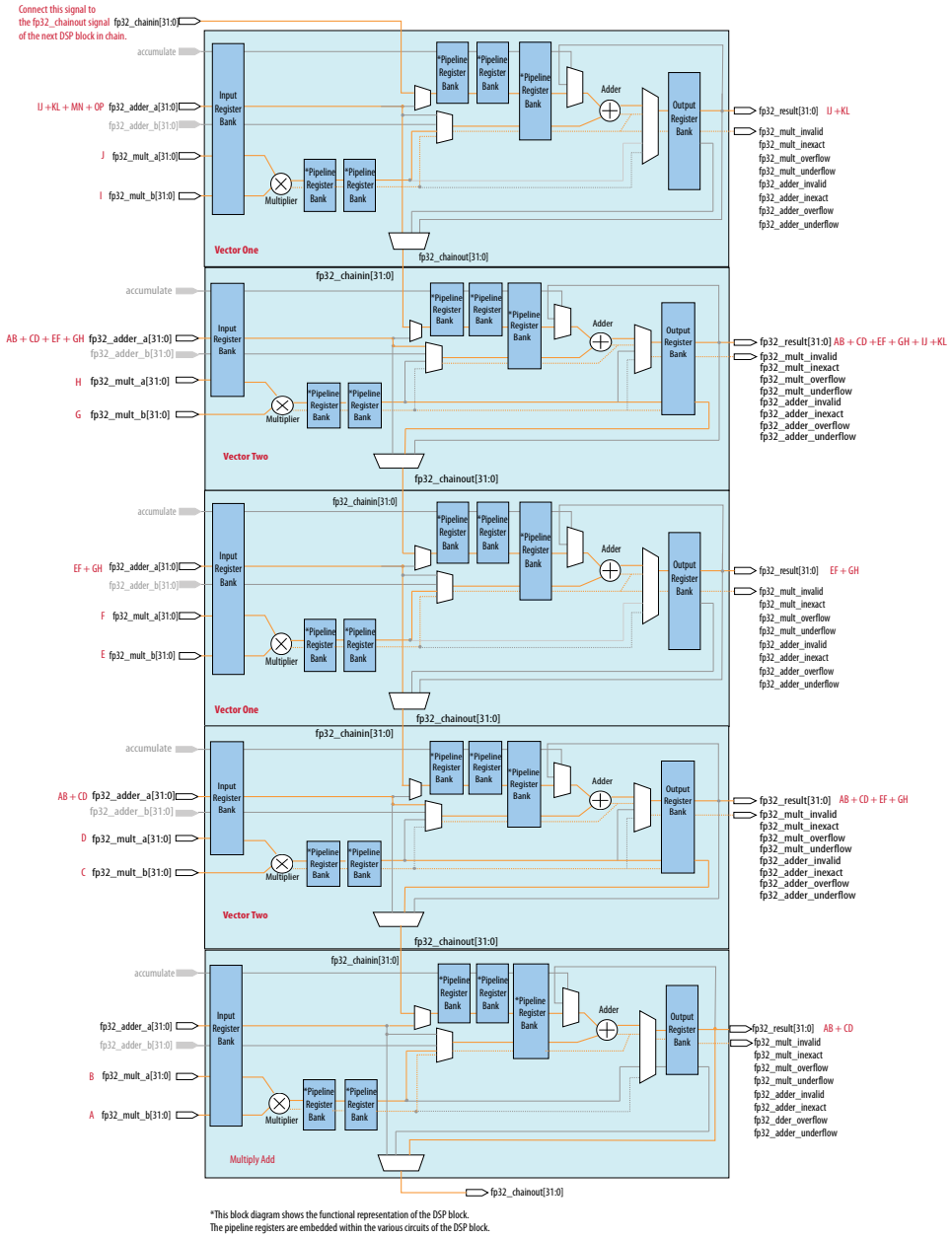
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

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

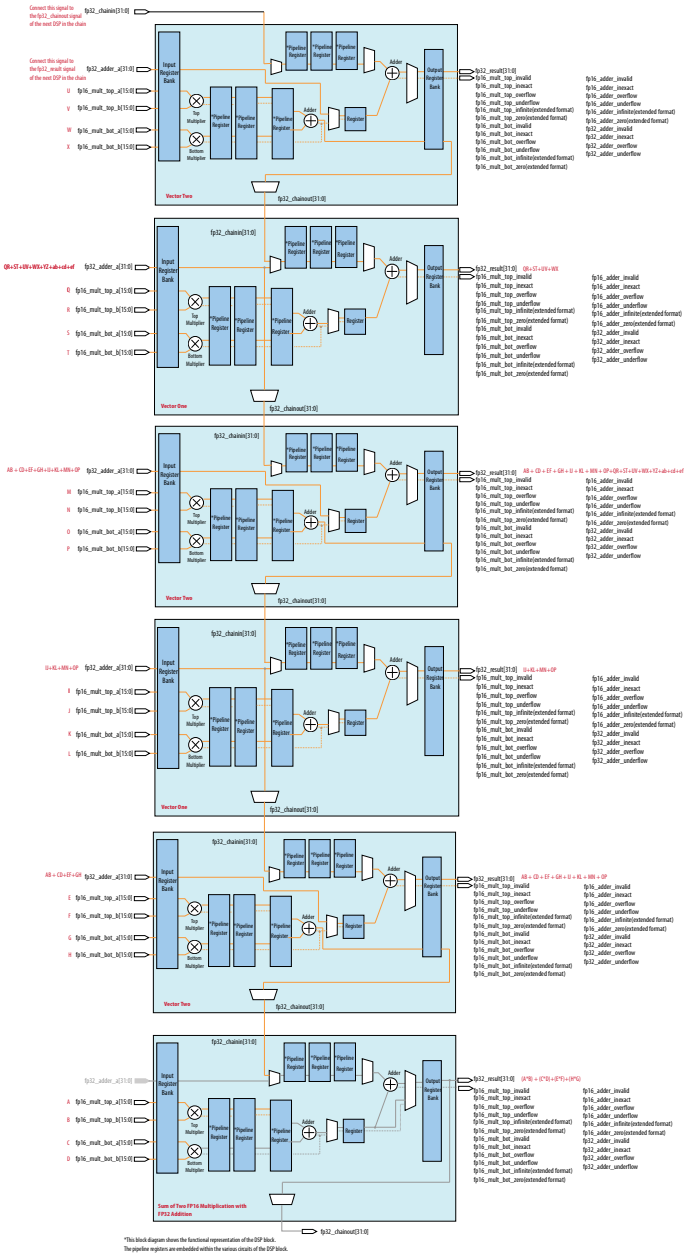
Figure 40. Direct Vector Dot Product Using FP32 Single-Precision Floating-Point Arithmetic



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

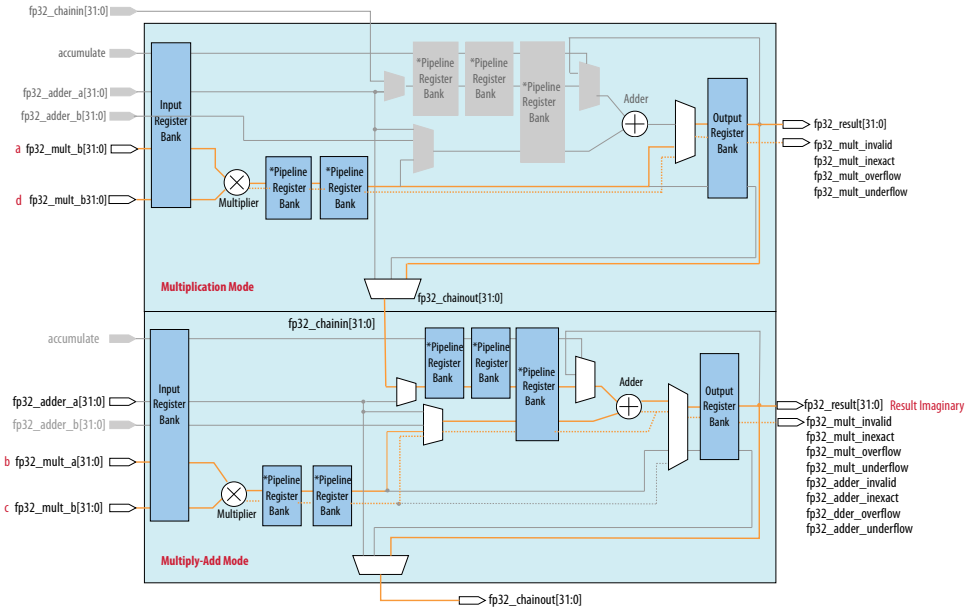
The Intel Agilex devices support the floating-point arithmetic single precision complex multiplier using four Intel Agilex variable-precision DSP blocks.

Figure 42. Sample of Complex Multiplication Equation

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

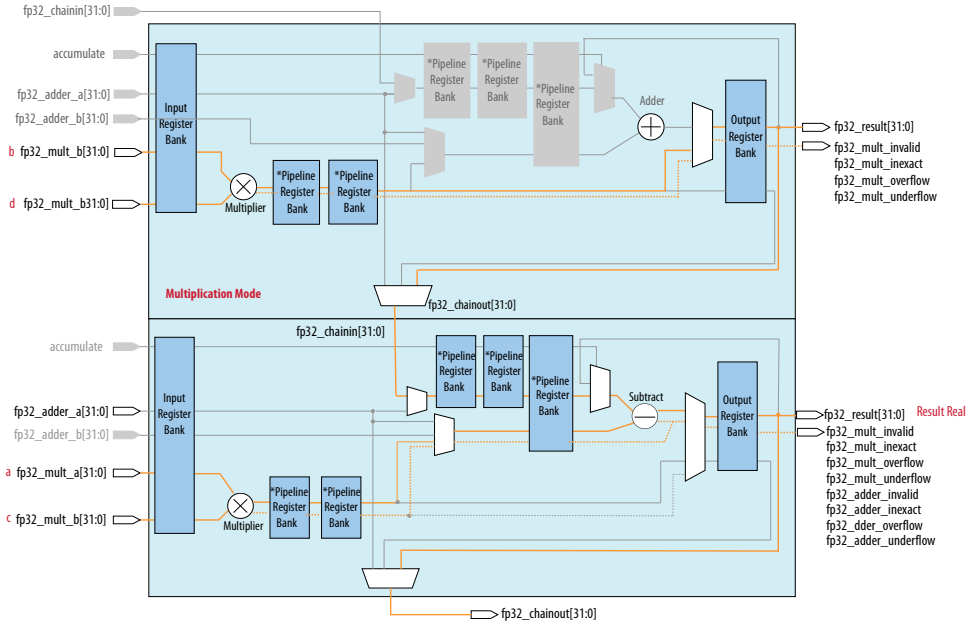
The imaginary part [(a × d) + (b × c)] is implemented in the first two variable-precision DSP blocks, while the real part [(a × c) - (b × d)] is implemented in the next two variable-precision DSP blocks.

Figure 43. Complex Multiplication with Imaginary Result Using FP32 Single-Precision Floating-Point Arithmetic



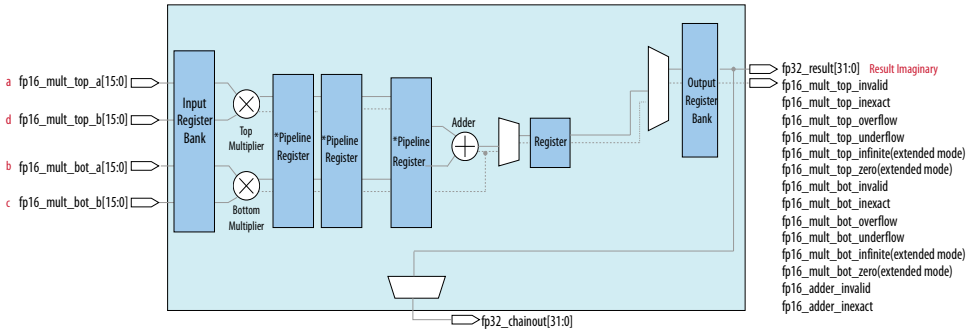
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

Figure 44. Complex Multiplication with Result Real Using FP32 Single-Precision Floating-Point Arithmetic



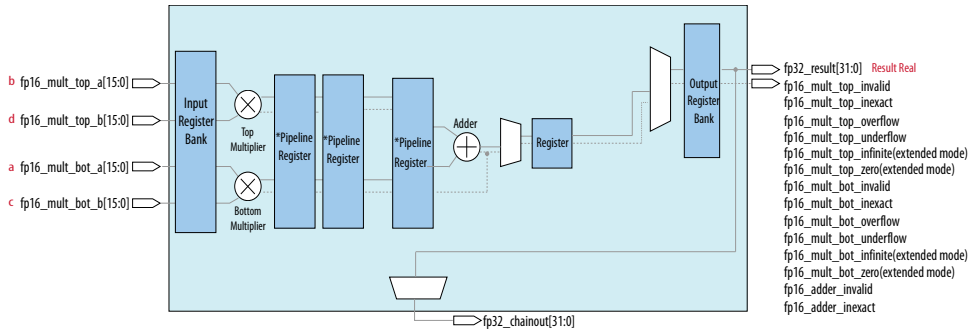
*This block diagram shows the functional representation of the DSP block.
The pipeline registers are embedded within the various circuits of the DSP block.

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 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 devices. Therefore these registers only support certain configurations.

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.

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.

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
	3	Enable	Disable	Enable	Enable
	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
	3(6)	Enable	Enable	Disable	Enable
	3(7)	Enable	Disable	Enable	Enable
	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
	3(6)	Enable	Enable	Disable	Enable
	3(7)	Enable	Disable	Enable	Enable
	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
	3	Enable	Disable	Enable	Enable
	4	Enable	Enable	Enable	Enable

continued...

(5) When Accumulator and/or Output Chainin are 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
18 x 19 systolic mode	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
	3(6)	Enable	Enable	Disable	Enable
	3(7)	Enable	Disable	Enable	Enable
	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 Intel Agilex 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 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

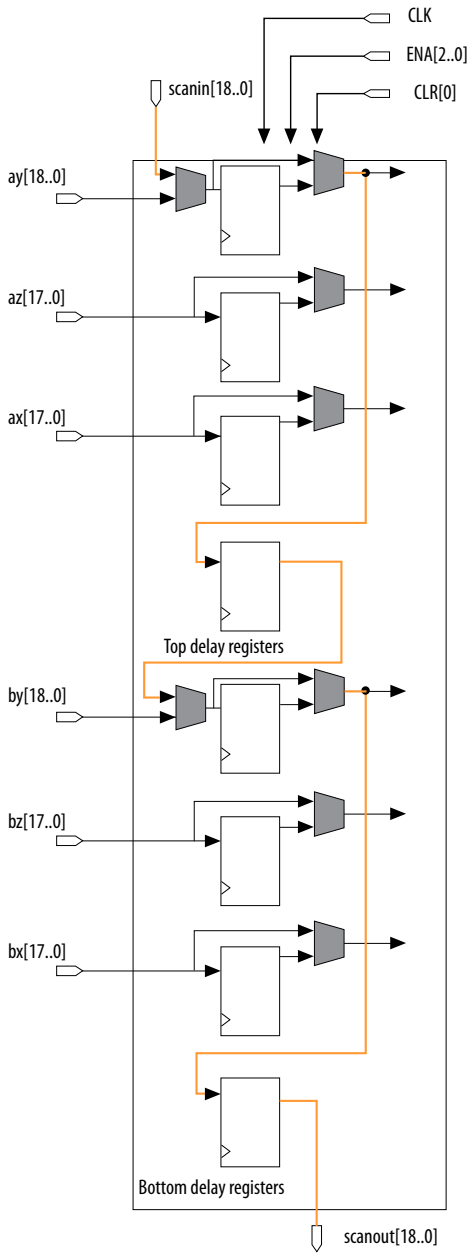
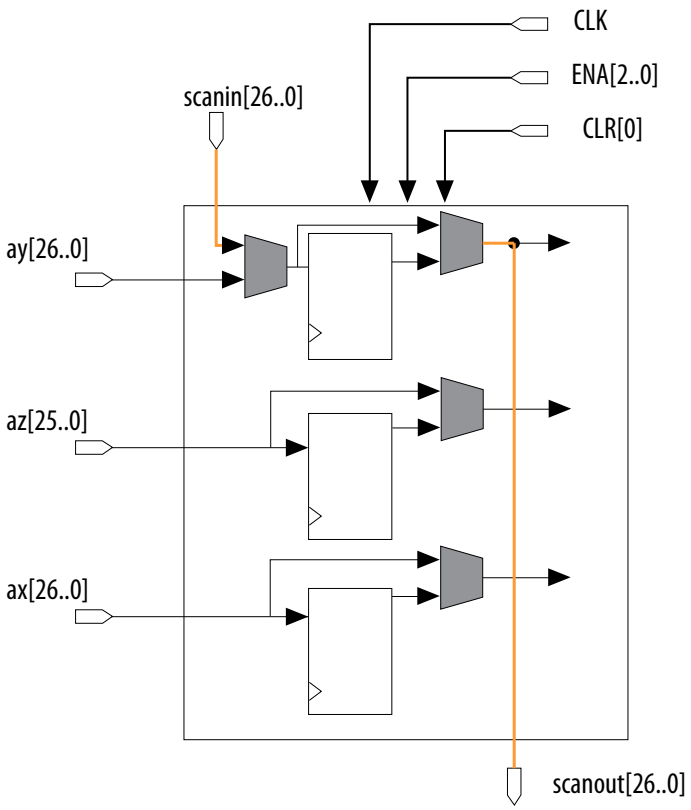


Figure 48. Input Cascade in Fixed-Point Arithmetic 27 x 27 Mode



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

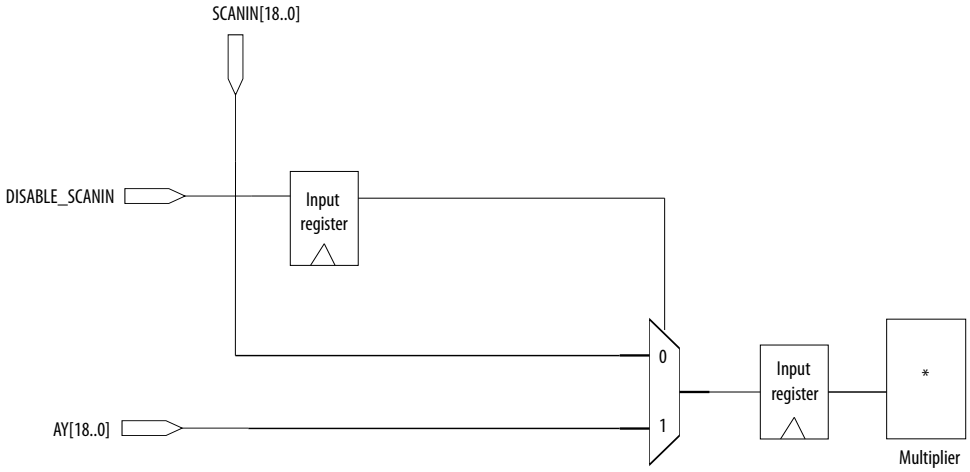


Table 25. DISABLE_SCANIN Signal Behavior

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 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
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		Enable
2	Enable	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

continued...

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
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

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

Document Version	Changes
2019.04.02	Initial release.