10. DSP Blocks in Arria GX Devices

AGX52010-1.2

Introduction

Arria™ GX devices have dedicated digital signal processing (DSP) blocks optimized for DSP applications requiring high data throughput. These DSP blocks combined with the flexibility of programmable logic devices (PLDs), provide you with the ability to implement various high performance DSP functions easily. Complex systems such as CDMA2000, voice over Internet protocol (VoIP), and high-definition television (HDTV) require high performance DSP blocks to process data. These system designs typically use DSP blocks as finite impulse response (FIR) filters, complex FIR filters, fast Fourier transform (FFT) functions, discrete cosine transform (DCT) functions, and correlators.

Arria GX DSP blocks consist of a combination of dedicated blocks that perform multiplication, addition, subtraction, accumulation, and summation operations. You can configure these blocks to implement arithmetic functions like multipliers, multiply-adders and multiply-accumulators which are necessary for most DSP functions.

Along with the DSP blocks, the TriMatrix™ memory structures in Arria GX devices also support various soft multiplier implementations. The combination of soft multipliers and dedicated DSP blocks increases the number of multipliers available in Arria GX devices and provides you with a wide variety of implementation options and flexibility when designing your systems.

For more information about Arria GX devices respectively, see the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

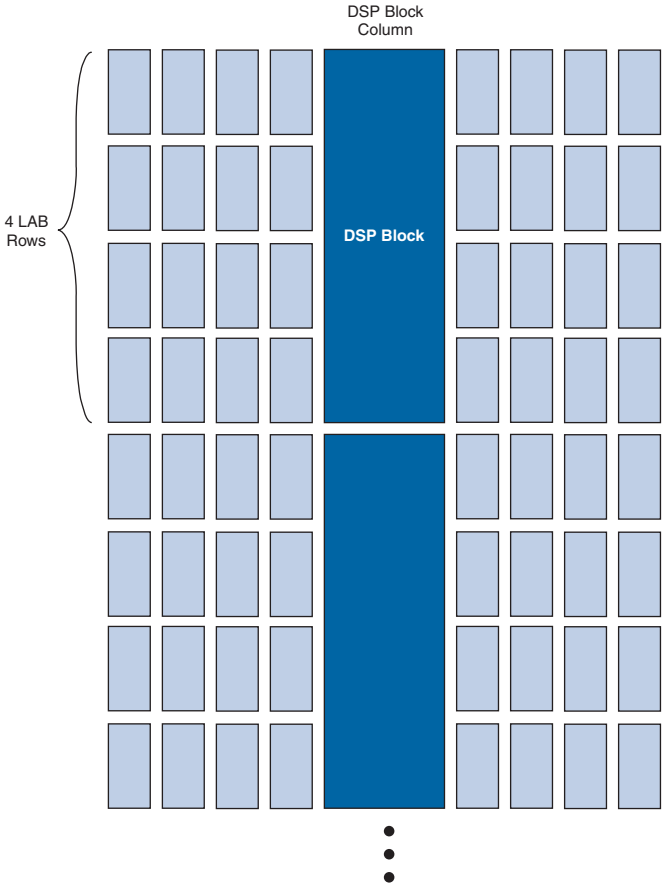
This chapter contains the following sections:

DSP Block Overview

Each Arria GX device has two to four columns of DSP blocks that efficiently implement multiplication, multiply-accumulate (MAC) and multiply-add functions. Figure 10–1 shows the arrangement of one of the DSP block columns with the surrounding LABs. Each DSP block can be configured to support:

- Eight 9 × 9-bit multipliers
- Four 18 × 18-bit multipliers
- One 36 × 36-bit multiplier

Figure 10–1. DSP Blocks Arranged in Columns with Adjacent LABs



The multipliers then feed an adder or accumulator block within the DSP block. Arria GX device multipliers support rounding and saturation on Q1.15 input formats. The DSP block also has input registers that can be configured to operate in a shift register chain for efficient implementation of functions such as FIR filters. The accumulator within the DSP block can be initialized to any value and supports rounding and saturation on Q1.15 input formats to the multiplier. A single DSP block can be broken down to operate different configuration modes simultaneously.

For more information on Q1.15 formatting, see the section "Saturation and Rounding" on page 10–11.

The number of DSP blocks per column and the number of columns available increases with device density.

Table 10–1 shows the number of DSP blocks in each Arria GX device and the multipliers that you can implement.

Table 10–1. Number of DSP Blocks in Arria GX Devices Note (1)

Device	DSP Blocks	9 × 9 Multipliers	18 × 18 Multipliers	36 × 36 Multipliers
EP1AGX20C	10	80	40	10
EP1AGX35C/D	14	112	56	14
EP1AGX50C/D	26	208	104	26
EP1AGX60C/D/E	32	256	128	32
EP1AGX90E	44	352	176	44

Note to Table 10–1:

(1) Each device has either the number of 9 × 9-, 18 × 18-, or 36 × 36-bit multipliers shown. The total number of multipliers for each device is not the sum of all the multipliers.

In addition to the DSP block multipliers, you can use the Arria GX device's TriMatrix memory blocks for soft multipliers. The availability of soft multipliers increases the number of multipliers available within the device. Table 10–2 shows the total number of multipliers available in Arria GX devices using DSP blocks and soft multipliers.

Table 10–2. Number of Multipliers in Arria GX Devices

Device	DSP Blocks (18 × 18)	Soft Multipliers (16 × 16) *(1)*, *(2)*	Total Multipliers *(3)*, *(4)*
EP1AGX20C	40	102	142 (3.55)
EP1AGX35C/D	56	122	178 (3.18)
EP1AGX50C/D	104	202	306 (2.94)
EP1AGX60C/D/E	128	211	339 (2.65)
EP1AGX90E	176	324	500 (2.84)

Notes to Table 10–2:

(1) Soft multipliers implemented in sum of multiplication mode. RAM blocks are configured with 18-bit data widths and sum of coefficients up to 18-bits.

(2) Soft multipliers are only implemented in M4K and M512 TriMatrix memory blocks, not M-RAM blocks.

(3) The number in parentheses represents the increase factor, which is the total number of multipliers with soft multipliers divided by the number of 18 × 18 multipliers supported by DSP blocks only.

(4) The total number of multipliers may vary according to the multiplier mode used.

Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for more information about Arria GX TriMatrix memory blocks.

Refer to *AN 306: Implementing Multipliers in FPGA Devices* for more information on soft multipliers.

Figure 10–2 shows the DSP block configured for 18 × 18 multiplier mode.

Figure 10–2. DSP Block in 18 × 18 Mode

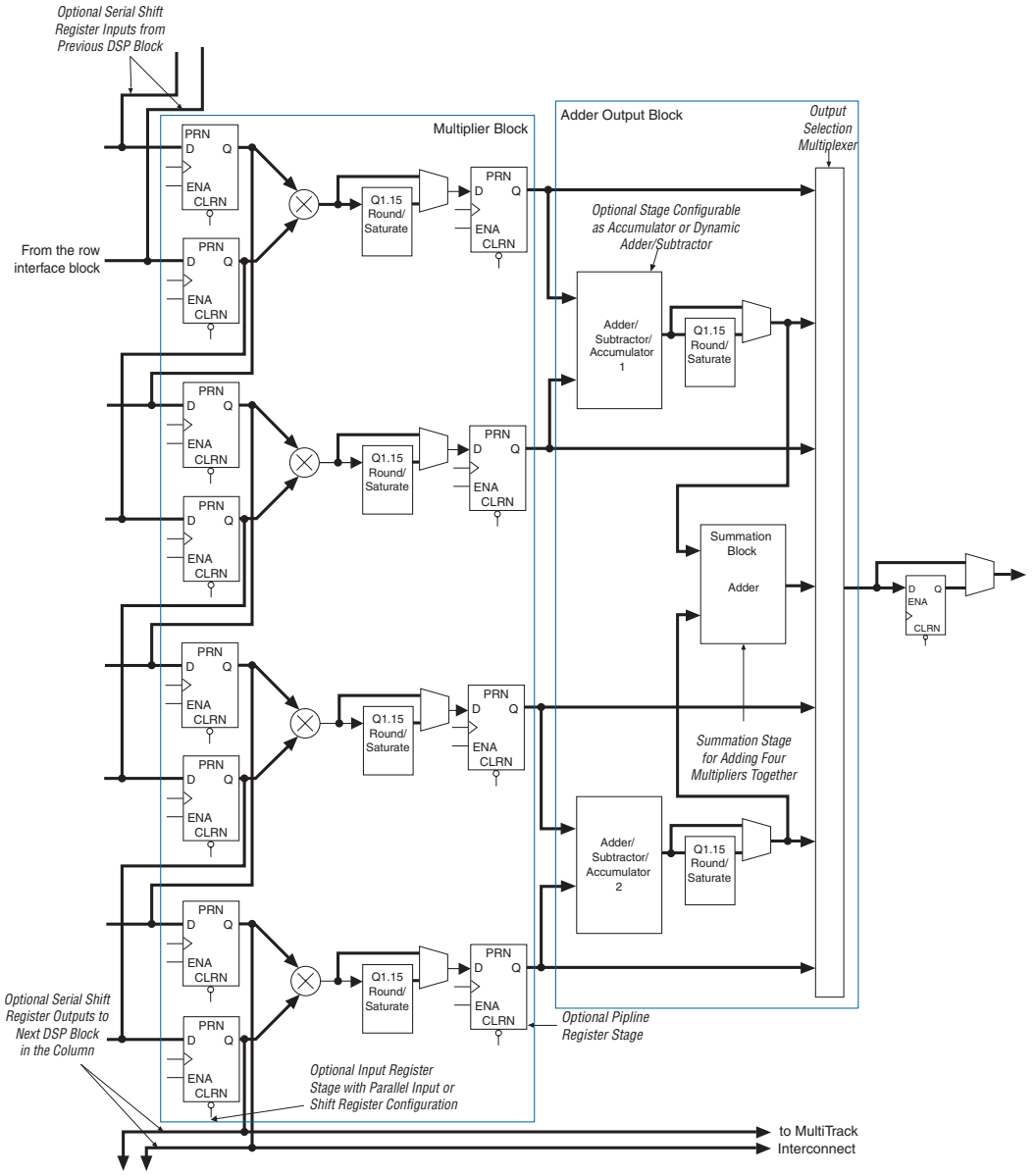
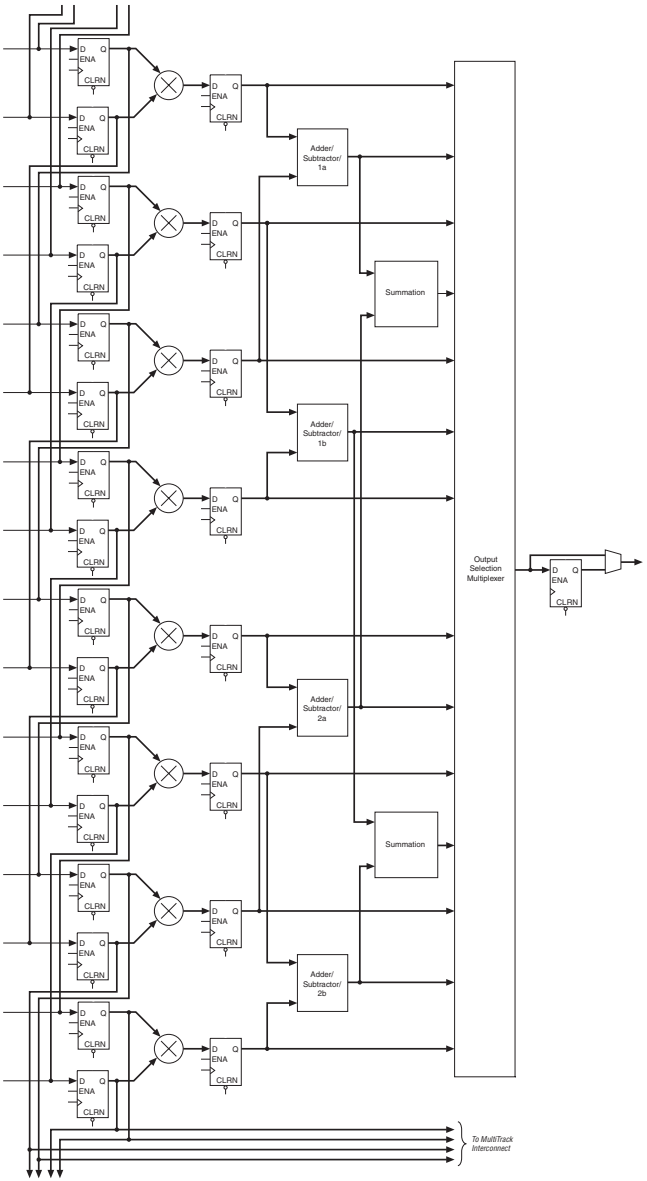


Figure 10–3 shows the 9 × 9 multiplier configuration of the DSP block.

Figure 10–3. DSP Block in 9 × 9 Mode



Architecture

The DSP block consists of the following elements:

- A multiplier block
- An adder/subtractor/accumulator block
- A summation block
- Input and output interfaces
- Input and output registers

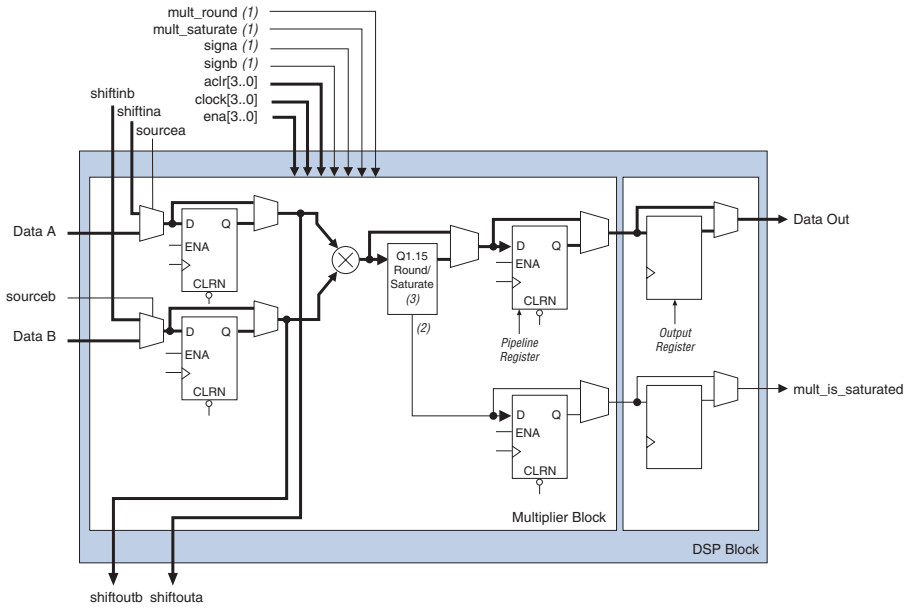
Multiplier Block

Each multiplier block has the following elements:

- Input registers
- A multiplier block
- A rounding and/or saturation stage for Q1.15 input formats
- A pipeline output register

Figure 10–4 shows the multiplier block architecture.

Figure 10–4. Multiplier Block Architecture



Notes to Figure 10–4:

(1) These signals are not registered or registered once to match the data path pipeline.
(2) You can send these signals through either one or two pipeline registers.
(3) The rounding and/or saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

Input Registers

Each multiplier operand can feed an input register or directly to the multiplier. The following DSP block signals control each input register within the DSP block:

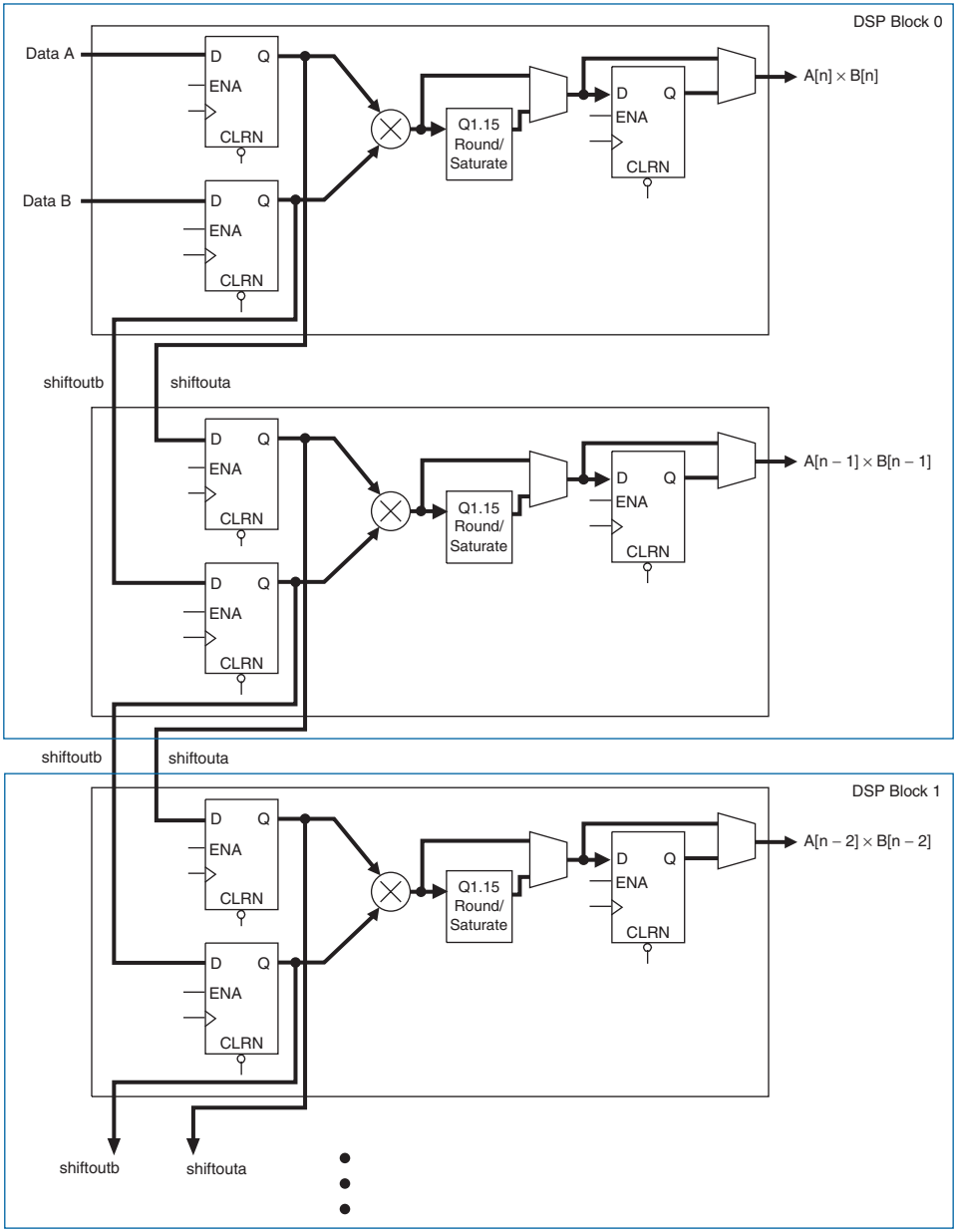
- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

The input registers feed the multiplier and drive two dedicated shift output lines, `shiftouta` and `shiftoutb`. The dedicated shift outputs from one multiplier block directly feed input registers of the adjacent multiplier below it within the same DSP block or the first multiplier in the next DSP block to form a shift register chain, as shown in Figure 10–5. The dedicated shift register chain spans a single column but longer shift register chains requiring multiple columns can be implemented using regular FPGA routing resources. Therefore, this shift register chain can be of any length up to 768 registers in the largest member of the Arria GX device family.

Shift registers are useful in DSP functions such as FIR filters. When implementing 9 × 9 and 18 × 18 multipliers, you do not need external logic to create the shift register chain because the input shift registers are internal to the DSP block. This implementation significantly reduces the LE resources required, avoids routing congestion, and results in predictable timing.

Arria GX DSP blocks allow you to dynamically select whether a particular multiplier operand is fed by regular data input or the dedicated shift register input using the `sourcea` and `sourceb` signals. A logic 1 value on the `sourcea` signal indicates that data A is fed by the dedicated scan-chain; a logic 0 value indicates that it is fed by regular data input. This feature allows the implementation of a dynamically loadable shift register where the shift register operates normally using the scan-chains and can also be loaded dynamically in parallel using the data input value. Figure 10–5 shows the shift register chain.

Figure 10–5. Shift Register Chain Note (1)



Note to Figure 10–5:

(1) Either Data A or Data B input can be set to a parallel input for constant coefficient multiplication.

Table 10–3 shows the summary of input register modes for the DSP block.

Table 10–3. Input Register Modes

Register Input Mode	9 × 9	18 × 18	36 × 36
Parallel input	✓	✓	✓
Shift register input	✓	✓	-

Multiplier Stage

The multiplier stage supports 9 × 9, 18 × 18, or 36 × 36 multipliers as well as other smaller multipliers in between these configurations. See "Operational Modes" on page 10–18 for details. Depending on the data width of the multiplier, a single DSP block can perform many multiplications in parallel.

Each multiplier operand can be a unique signed or unsigned number. Two signals, `signa` and `signb`, control the representation of each operand respectively. A logic 1 value on the `signa` signal indicates that data A is a signed number while a logic 0 value indicates an unsigned number. Table 10–4 shows the sign of the multiplication result for the various operand sign representations. The result of the multiplication is signed if any one of the operands is a signed value.

Table 10–4. Multiplier Sign Representation

Data A (signa Value)	Data B (signb Value)	Result
Unsigned (logic 0)	Unsigned (logic 0)	Unsigned
Unsigned (logic 0)	Signed (logic 1)	Signed
Signed (logic 1)	Unsigned (logic 0)	Signed
Signed (logic 1)	Signed (logic 1)	Signed

There is only one `signa` and one `signb` signal for each DSP block. Therefore, all of the data A inputs feeding the same DSP block must have the same sign representation. Similarly, all of the data B inputs feeding the same DSP block must have the same sign representation. The multiplier offers full precision regardless of the sign representation.

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

Saturation and Rounding

The DSP blocks have hardware support to perform optional saturation and rounding after each 18 × 18 multiplier for Q1.15 input formats.

Designs must use 18 × 18 multipliers for the saturation and rounding options because the Q1.15 input format requires 16-bit input widths.

Q1.15 input format multiplication requires signed multipliers. The most significant bit (MSB) in the Q1.15 input format represents the value's sign bit. Use signed multipliers to ensure the proper sign extension during multiplication.

The Q1.15 format uses 16 bits to represent each fixed point input. The MSB is the sign bit, and the remaining 15-bits are used to represent the value after the decimal place (or the fractional value). This Q1.15 value is equivalent to an integer number representation of the 16-bits divided by 2^{15}, as shown in the following equations.

$$-\frac{1}{2} = 1\ 100\ 0000\ 0000\ 0000 = -\frac{0x4000}{2^{15}}$$

$$\frac{1}{8} = 0\ 001\ 0000\ 0000\ 0000 = \frac{0x1000}{2^{15}}$$

All Q1.15 numbers are between –1 and 1.

When performing multiplication, even though the Q1.15 input only uses 16 of the 18 multiplier inputs, the entire 18-bit input bus is transmitted to the multiplier. This is similar to a 1.17 input, where the two least significant bits (LSBs) are always 0.

The multiplier output will be a 2.34 value (36 bits total) before performing any rounding or saturation. The two MSBs are sign bits. Since the output only requires one sign bit, you can ignore one of the two MSBs, resulting in a Q1.34 value before rounding or saturation.

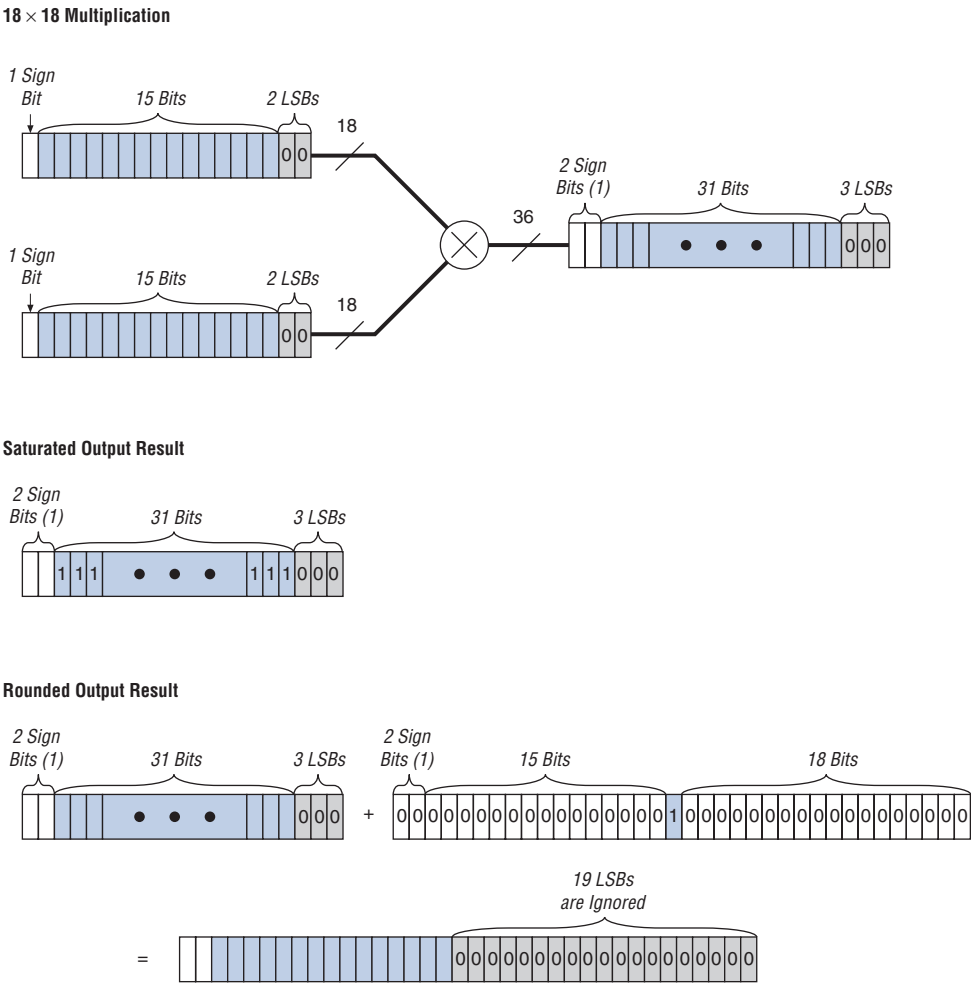
When the design performs saturation, the multiplier output gets saturated to 0x7FFFFFFF in a 1.31 format. This uses bits [34..3] of the overall 36-bit multiplier output. The three LSBs are set to 0.

The DSP block obtains the `mult_is_saturated` or `accum_is_saturated` overflow signal value from the LSB of the multiplier or accumulator output. Therefore, whenever saturation occurs, the LSB of the multiplier or accumulator output sends a `1` to the

mult_is_saturated or accum_is_saturated overflow signal. At all other times, this overflow signal is 0 when saturation is enabled or reflects the value of the LSB of the multiplier or accumulator output.

When the design performs rounding, it adds 0x00008000 in 1.31 format to the multiplier output, and it only uses bits [34..15] of the overall 36-bit multiplier output. Adding 0x00008000 in 1.31 format to the 36-bit multiplier result is equivalent to adding 0x0 0004 0000 in 2.34 format. The 16 LSBs are set to 0. Figure 10–6 shows which bits are used when the design performs rounding and saturation for the multiplication.

Figure 10–6. Rounding and Saturation Bits



Note to Figure 10–6:

(1) Both sign bits are the same. The design only uses one sign bit, and the other one is ignored.

If the design performs a `multiply_accumulate` or `multiply_add` operation, the multiplier output is input to the adder/subtractor/accumulator blocks as a 2.31 value, and the three LSBs are 0.

Pipeline Registers

The output from the multiplier can feed a pipeline register or this register can be bypassed. Pipeline registers may be implemented for any multiplier size and increase the DSP block's maximum performance, especially when using the subsequent DSP block adder stages. Pipeline registers split up the long signal path between the adder/subtractor/accumulator block and the adder/output block, creating two shorter paths.

Adder/Output Block

The adder/output block has the following elements:

- An adder/subtractor/accumulator block
- A summation block
- An output select multiplexer
- Output registers

The adder/output block can be configured as:

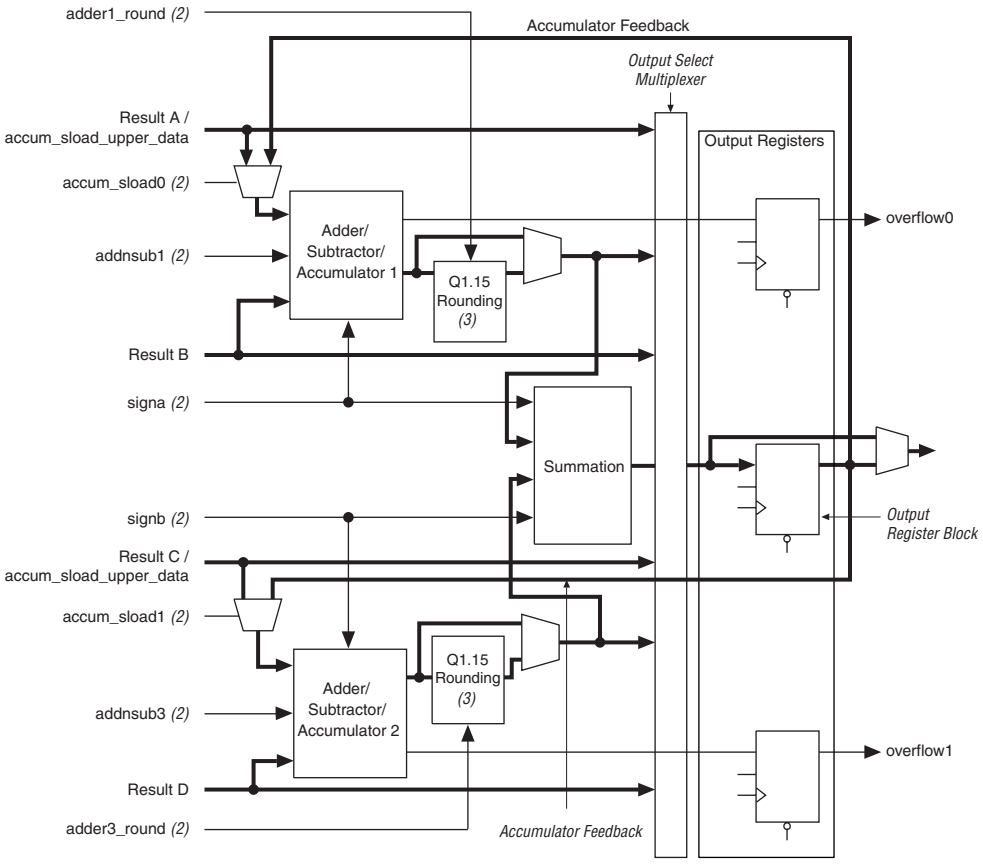
- An output interface
- An accumulator which can be optionally loaded
- A one-level adder
- A two-level adder with dynamic addition/subtraction control on the first-level adder
- The final stage of a 36-bit multiplier, 9 × 9 complex multiplier, or 18 × 18 complex multiplier

The output select multiplexer sets the output configuration of the DSP block. The output registers can be used to register the output of the adder/output block.

The adder/output block cannot be used independently from the multiplier.

Figure 10–7 shows the adder/output block architecture.

Figure 10–7. Adder/Output Block Architecture



Notes to Figure 10–7:

(1) The adder/output block is in 18 × 18 mode. In 9 × 9 mode, there are four adder/subtractor blocks and two summation blocks.

(2) You can send these signals through a pipeline register. The pipeline length can be set to 1 or 2.

(3) Q1.15 inputs are not available in 9 × 9 or 36 × 36 modes.

Adder/Subtractor/Accumulator Block

The adder/subtractor/accumulator block is the first level adder stage of the adder/output block. This block can be configured as an accumulator or as an adder/subtractor.

Accumulator

When the adder/subtractor/accumulator is configured as an accumulator, the output of the adder/output block feeds back to the accumulator as shown in Figure 10–7. The accumulator can be set up to perform addition only, subtraction only or the addnsub signal can be used to dynamically control the accumulation direction. A logic 1 value on the addnsub signal indicates that the accumulator is performing addition while a logic 0 value indicates subtraction.

Each accumulator can be cleared by either clearing the DSP block output register or by using the accum_sload signal. The accumulator clear using the accum_sload signal is independent from the resetting of the output registers so the accumulation can be cleared and a new one can begin without losing any clock cycles. The accum_sload signal controls a feedback multiplexer that specifies that the output of the multiplier should be summed with a zero instead of the accumulator feedback path.

The accumulator can also be initialized/preloaded with a non-zero value using the accum_sload signal and the accum_sload_upper_data bus with one clock cycle latency. Preloading the accumulator is done by adding the result of the multiplier with the value specified on the accum_sload_upper_data bus. As in the case of the accumulator clearing, the accum_sload signal specifies to the feedback multiplexer that the accum_sload_upper_data signal should feed the accumulator instead of the accumulator feedback signal. The accum_sload_upper_data signal only loads the upper 36-bits of the accumulator. To load the entire accumulator, the value for the lower 16-bits must be sent through the multiplier feeding that accumulator with the multiplier set to perform a multiplication by one.

The overflow signal will go high on the positive edge of the clock when the accumulator detects an overflow or underflow. The overflow signal will stay high for only one clock cycle after an overflow or underflow is detected even if the overflow or underflow condition is still present. A latch external to the DSP block has to be used to preserve the overflow signal indefinitely or until the latch is cleared.

The DSP blocks support Q1.15 input format saturation and rounding in each accumulator. The following signals are available that can control if saturation or rounding or both is performed to the output of the accumulator:

- accum_round
- accum_saturation
- accum_is_saturated output

Each DSP block has two sets of accum_round and accum_saturation signals which control if rounding or saturation is performed on the accumulator output respectively (one set of signals for each accumulator). Rounding and saturation of the accumulator output is only available when implementing a 16 × 16 multiplier-accumulator to conform to the bit widths required for Q1.15 input format computation. A logic 1 value on the accum_round and accum_saturation signal indicates that rounding or saturation is performed while a logic 0 indicates that no rounding or saturation is performed. A logic 1 value on the accum_is_saturated output signal tells you that saturation has occurred to the result of the accumulator.

Figure 10–10 shows the DSP block configured to perform multiplier-accumulator operations.

Adder/Subtractor

The addnsub1 or addnsub3 signals specify whether you are performing addition or subtraction. A logic 1 value on the addnsub1 or addnsub3 signals indicates that the adder/subtractor is performing addition while a logic 0 value indicates subtraction. These signals can be dynamically controlled using logic external to the DSP block. If the first stage is configured as a subtractor, the output is A – B and C – D.

The adder/subtractor block share the same signa and signb signals as the multiplier block. The signa and signb signals can be pipelined with a latency of one or two clock cycles or not.

The DSP blocks support Q1.15 input format rounding (not saturation) after each adder/subtractor. The addnsub1_round and addnsub3_round signals determine if rounding is performed to the output of the adder/subtractor.

The addnsub1_round signal controls the rounding of the top adder/subtractor and the addnsub3_round signal controls the rounding of the bottom adder/subtractor. Rounding of the adder output is only available when implementing a 16 × 16 multiplier-adder to conform to the bit widths required for Q1.15 input format computation. A logic 1 value on the addnsub_round signal indicates that rounding is performed while a logic 0 indicates that no rounding is performed.

Summation Block

The output of the adder/subtractor block feeds an optional summation block, which is an adder block that sums the outputs of both adder/subtractor blocks. The summation block is used when more than two multiplier results are summed. This is useful in applications such as FIR filtering.

Output Select Multiplexer

The outputs of the different elements of the adder/output block are routed through an output select multiplexer. Depending on the operational mode of the DSP block, the output multiplexer selects whether the outputs of the DSP blocks comes from the outputs of the multiplier block, the outputs of the adder/subtractor/accumulator, or the output of the summation block. The output select multiplier configuration is set automatically by software, based on the DSP block operational mode you specify.

Output Registers

You can use the output registers to register the DSP block output. The following signals can control each output register within the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

The output registers can be used in any DSP block operational mode.

The output registers form part of the accumulator in the multiply-accumulate mode.

Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for more information on the DSP block routing and interface.

Operational Modes

The DSP block can be used in one of four basic operational modes, or a combination of two modes, depending on the application needs. Table 10–5 shows the four basic operational modes and the number of multipliers that can be implemented within a single DSP block depending on the mode.

Table 10–5. DSP Block Operational Modes			
Mode	**Number of Multipliers**		
	9 × 9	**18 × 18**	**36 × 36**
Simple multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier
Multiply accumulate	-	Two 52-bit multiply-accumulate blocks	-
Two-multiplier adder	Four two-multiplier adder (two 9 9 complex multiply)	Two two-multiplier adder (one 18 18 complex multiply)	-
Four-multiplier adder	Two four-multiplier adder	One four-multiplier adder	-

The Quartus II software includes megafunctions used to control the mode of operation of the multipliers. After you make the appropriate parameter settings using the megafunction's MegaWizard® Plug-In Manager, the Quartus II software automatically configures the DSP block.

Arria GX DSP blocks can operate in different modes simultaneously. For example, a single DSP block can be broken down to operate a 9 × 9 multiplier as well as an 18 × 18 multiplier-adder where both multiplier's input a and input b have the same sign representations. This increases DSP block resource efficiency and allows you to implement more multipliers within an Arria GX device. The Quartus II software automatically places multipliers that can share the same DSP block resources within the same block.

Additionally, you can set up each Arria GX DSP block to dynamically switch between the following three modes:

- Up to four 18-bit independent multipliers
- Up to two 18-bit multiplier-accumulators
- One 36-bit multiplier

Each half of an Arria GX DSP block has separate mode control signals, which allows you to implement multiple 18-bit multipliers or multiplier-accumulators within the same DSP block and dynamically switch them independently (if they are in separate DSP block halves). If the design requires a 36-bit multiplier, you must switch the entire DSP block to accommodate it since the multiplier requires the entire DSP block. The smallest input bit width that supports dynamic mode switching is 18 bits.

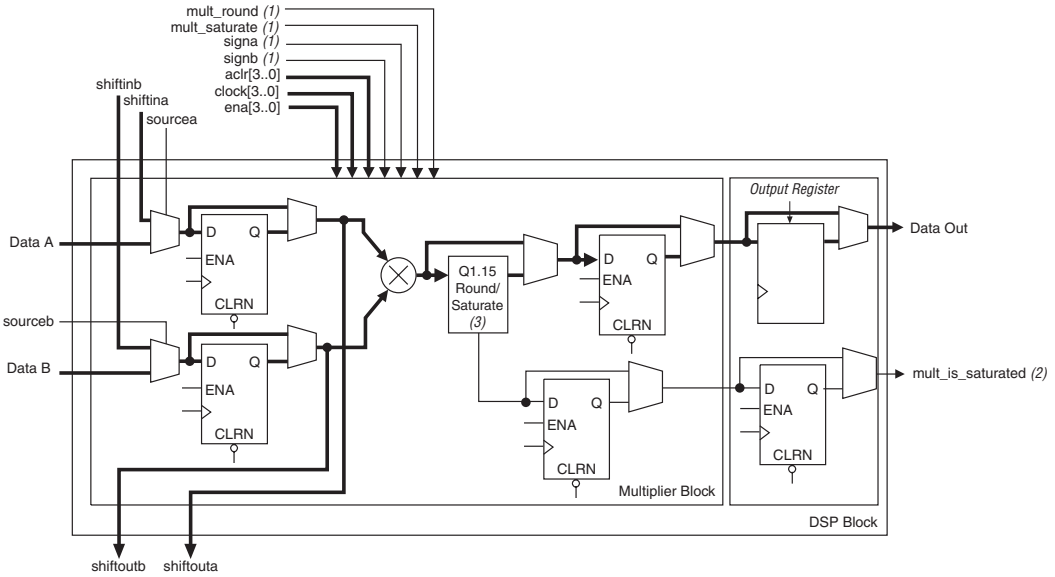
Simple Multiplier Mode

In simple multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers and for applications such as computing equalizer coefficient updates which require many individual multiplication operations.

9- and 18-Bit Multipliers

Each DSP block multiplier can be configured for 9- or 18-bit multiplication. A single DSP block can support up to eight individual 9 × 9 multipliers or up to four individual 18 × 18 multipliers. For operand widths up to 9-bits, a 9 × 9 multiplier will be implemented and for operand widths from 10- to 18-bits, an 18 × 18 multiplier will be implemented. Figure 10–8 shows the DSP block in the simple multiplier operation mode.

Figure 10–8. Simple Multiplier Mode



Notes to Figure 10–8:

(1) These signals are not registered or registered once to match the data path pipeline.

(2) This signal has the same latency as the data path.

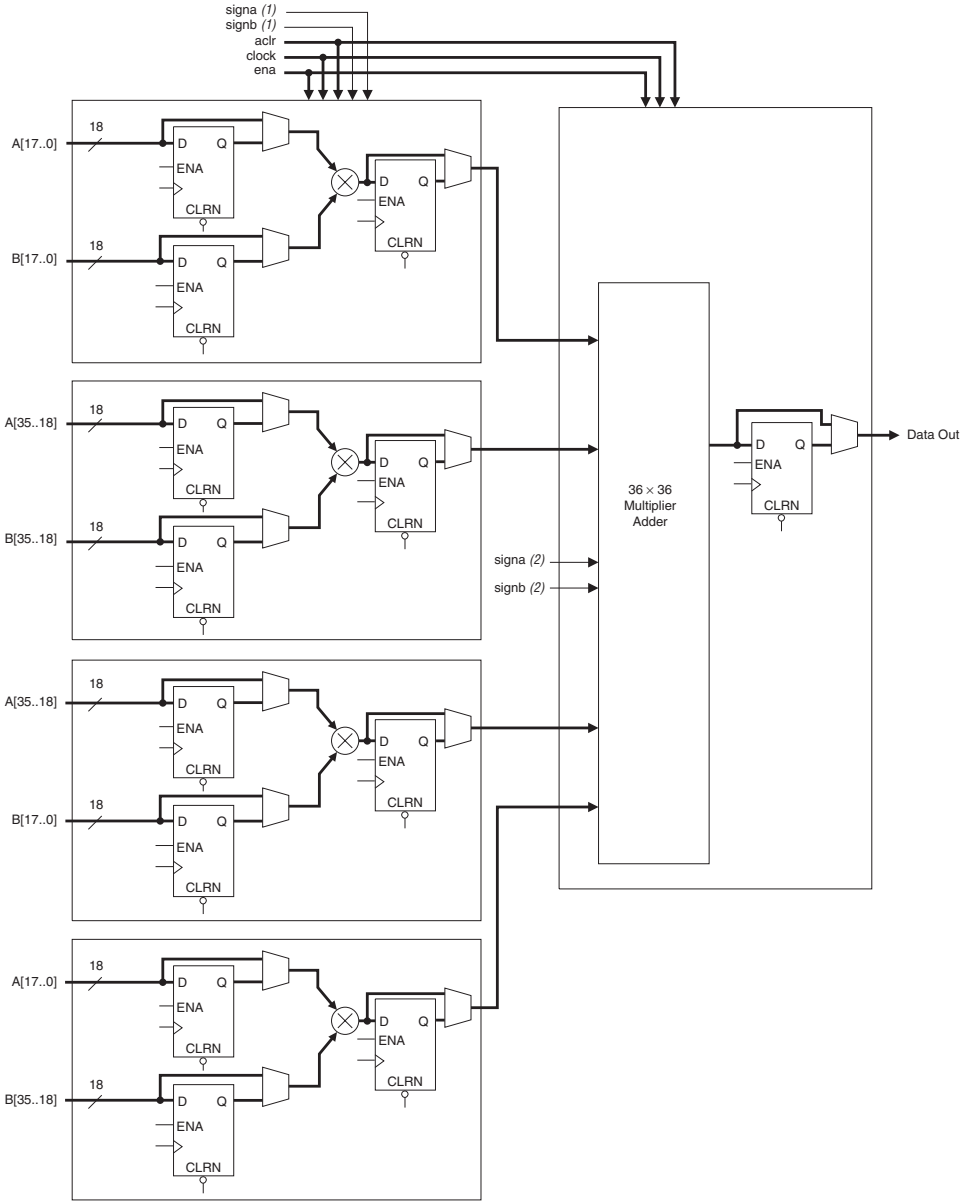
(3) The rounding and saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

The multiplier operands can accept signed integers, unsigned integers or a combination of both. The `signa` and `signb` signals can be changed dynamically and can be registered in the DSP block. Additionally, the multiplier inputs and result can be registered independently. The pipeline registers within the DSP block can be used to pipeline the multiplier result, increasing the performance of the DSP block.

36-Bit Multiplier

The 36-bit multiplier is also a simple multiplier mode but uses the entire DSP block, including the adder/output block to implement the 36 × 36-bit multiplication operation. The device inputs 18-bit sections of the 36-bit input into the four 18-bit multipliers. The adder/output block adds the partial products obtained from the multipliers using the summation block. Pipeline registers can be used between the multiplier stage and the summation block to speed up the multiplication. The 36 × 36-bit multiplier supports signed, unsigned as well as mixed sign multiplication. Figure 10–9 shows the DSP block configured to implement a 36-bit multiplier.

Figure 10–9. 36-Bit Multiplier



Notes to Figure 10–9:

(1) These signals are either not registered or registered once to match the pipeline.

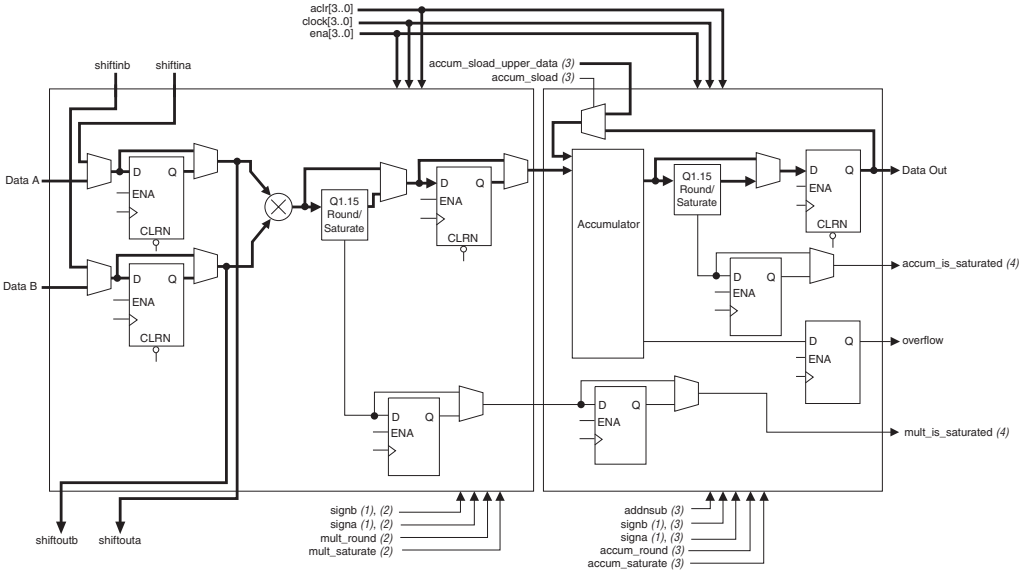
(2) These signals are either not registered, registered once, or registered twice to match the data path pipeline.

The 36-bit multiplier is useful for applications requiring more than 18-bit precision, for example, for mantissa multiplication of precision floating-point arithmetic applications.

Multiply Accumulate Mode

In multiply accumulate mode, the output of the multiplier stage feeds the adder/output block which is configured as an accumulator or subtractor. Figure 10–10 shows the DSP block configured to operate in multiply accumulate mode.

Figure 10–10. Multiply Accumulate Mode



Notes to Figure 10–10:

(1) The `signa` and `signb` signals are the same in the multiplier stage and the adder/output block.
(2) These signals are not registered or registered once to match the data path pipeline.
(3) You can send these signals through either one or two pipeline registers.
(4) These signals match the latency of the data path.

A single DSP block can implement up to two independent 18-bit multiplier accumulators. The Quartus II software implements smaller multiplier accumulators by tying the unused lower-order bits of the 18-bit multiplier to ground.

The multiplier accumulator output can be up to 52-bits wide to account for a 36-bit multiplier result with 16-bits of accumulation. In this mode, the DSP block uses output registers and the accum_sload and overflow signals. The accum_sload signal can be used to clear the accumulator so that a new accumulation operation can begin without losing any clock cycles. This signal can be unregistered or registered once or twice. The accum_sload signal can also be used to preload the accumulator with a value specified on the accum_sload_upper_data signal with a one clock cycle penalty. The accum_sload_upper_data signal only loads the upper 36-bits (bits [51..16] of the accumulator). To load the entire accumulator, the value for the lower 16-bits (bits [15..0]) must be sent through the multiplier feeding that accumulator with the multiplier set to perform a multiplication by one. Bits [17..16] are overlapped by both the accum_sload_upper_data signal and the multiplier output. Either one of these signals can be used to load bits [17..16].

The overflow signal indicates an overflow or underflow in the accumulator. This signal gets updated every clock cycle due to a new accumulation operation every cycle. To preserve the signal, an external latch can be used. The addnsub signal can be used to specify if an accumulation or subtraction is performed dynamically.

The DSP block can implement just an accumulator (without multiplication) by specifying a multiply by one at the multiplier stage followed by an accumulator to force the Quartus II software to implement the function within the DSP block.

Multiply Add Mode

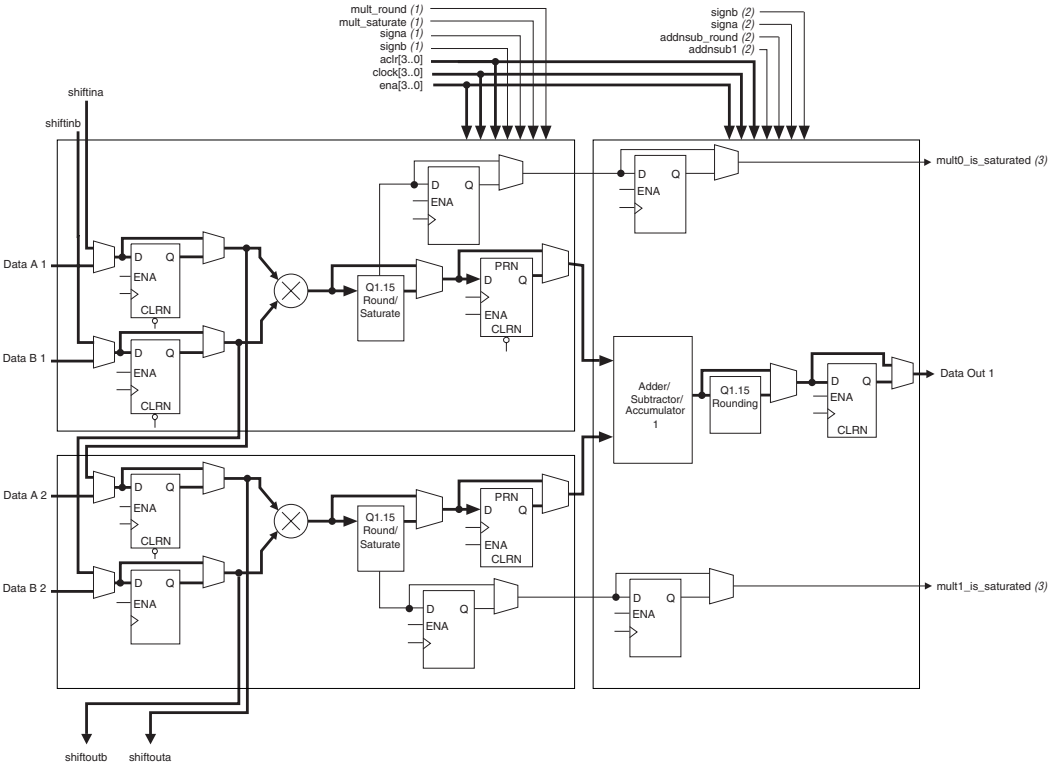
In multiply add mode, the output of the multiplier stage feeds the adder/output block which is configured as an adder or subtractor to sum or subtract the outputs of two or more multipliers. The DSP block can be configured to implement either a two-multiply add (where the outputs of two multipliers are added/subtracted together) or a four-multiply add function (where the outputs of four multipliers are added or subtracted together).

The adder block within the DSP block can only be used if it follows multiplication operations.

Two-Multiplier Adder

In the two-multiplier adder configuration, the DSP block can implement four 9-bit or smaller multiplier adders or two 18-bit multiplier adders. The adders can be configured to take the sum of both multiplier outputs or the difference of both multiplier outputs. You have the option to vary the summation/subtraction operation dynamically. These multiply add functions are useful for applications such as FFTs and complex FIR filters. Figure 10–11 shows the DSP block configured in the two-multiplier adder mode.

Figure 10–11. Two-Multiplier Adder Mode



Notes to Figure 10–11:

(1) These signals are not registered or registered once to match the data path pipeline.
(2) You can send these signals through a pipeline register. The pipeline length can be set to 1 or 2.
(3) These signals match the latency of the data path.

Complex Multiply

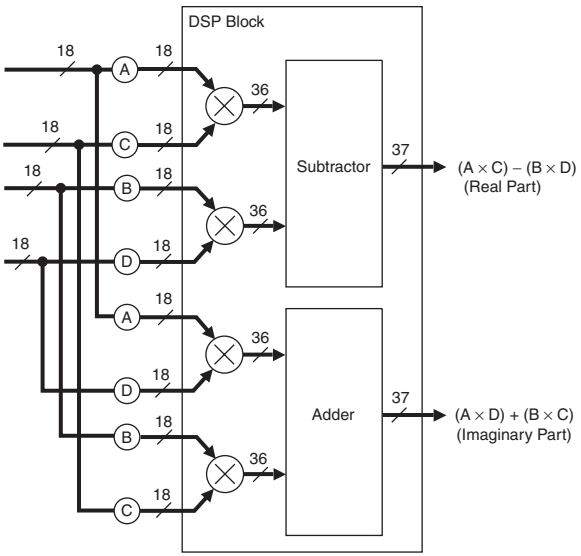
The DSP block can be configured to implement complex multipliers using the two-multiplier adder mode. A single DSP block can implement one 18 × 18-bit complex multiplier or two 9 × 9-bit complex multipliers.

A complex multiplication can be written as:

$$(a + jb)\ (c + jd) = ((ac) - (bd)) + j\ ((ad) + (bc))$$

To implement this complex multiplication within the DSP block, the real part ((*ac*) – (*bd*)) is implemented using two multipliers feeding one subtractor block while the imaginary part ((*ad*) + (*bc*)) is implemented using another two multipliers feeding an adder block, for data up to 18-bits. Figure 10–12 shows an 18-bit complex multiplication. For data widths up to 9-bits, a DSP block can perform two separate complex multiplication operations using eight 9-bit multipliers feeding four adder/subtractor/accumulator blocks. Resources external to the DSP block must be used to route the correct real and imaginary input components to the appropriate multiplier inputs to perform the correct computation for the complex multiplication operation.

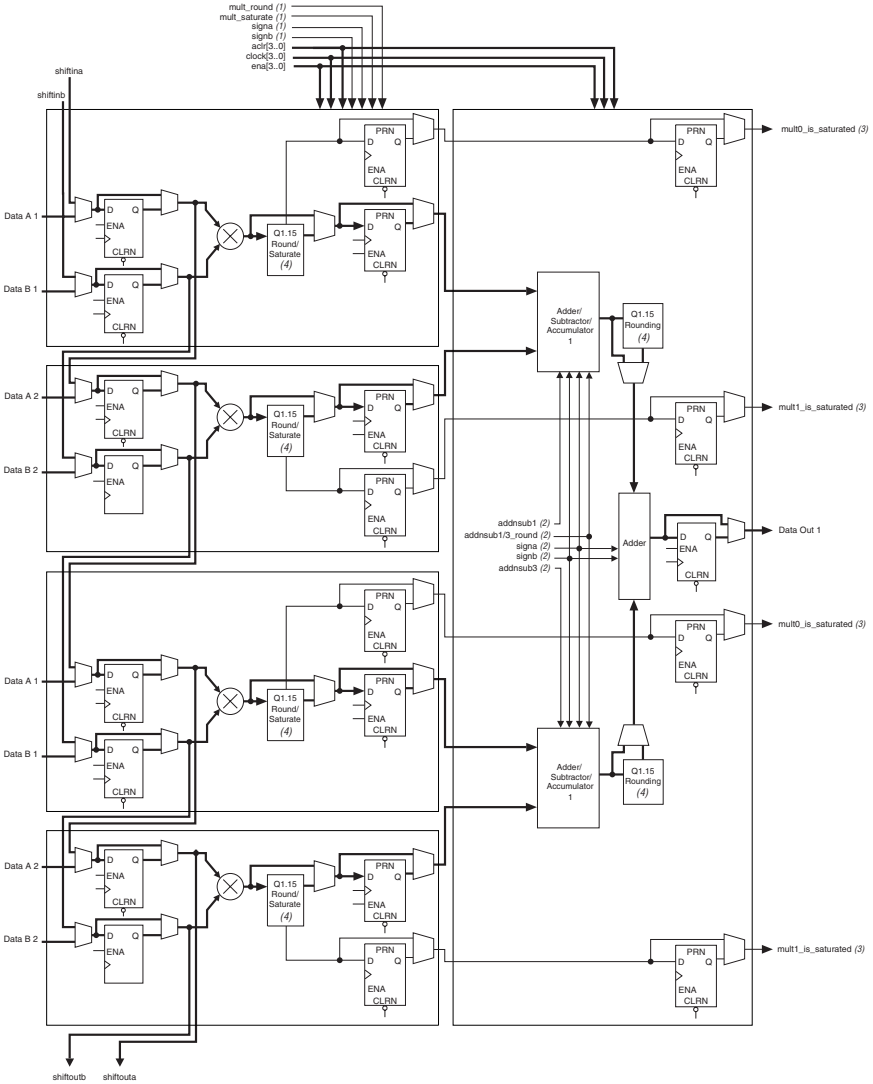
Figure 10–12. Complex Multiplier Using Two-Multiplier Adder Mode



Four-Multiplier Adder

In the four-multiplier adder configuration, the DSP block can implement one 18 × 18 or two individual 9 × 9 multiplier adders. These modes are useful for implementing one-dimensional and two-dimensional filtering applications. The four-multiplier adder is performed in two addition stages. The outputs of two of the four multipliers are initially summed in the two first-stage adder/subtractor/accumulator blocks. The results of these two adder/subtractor/accumulator blocks are then summed in the final stage summation block to produce the final four-multiplier adder result. Figure 10–13 shows the DSP block configured in the four-multiplier adder mode.

Figure 10–13. Four-Multiplier Adder Mode



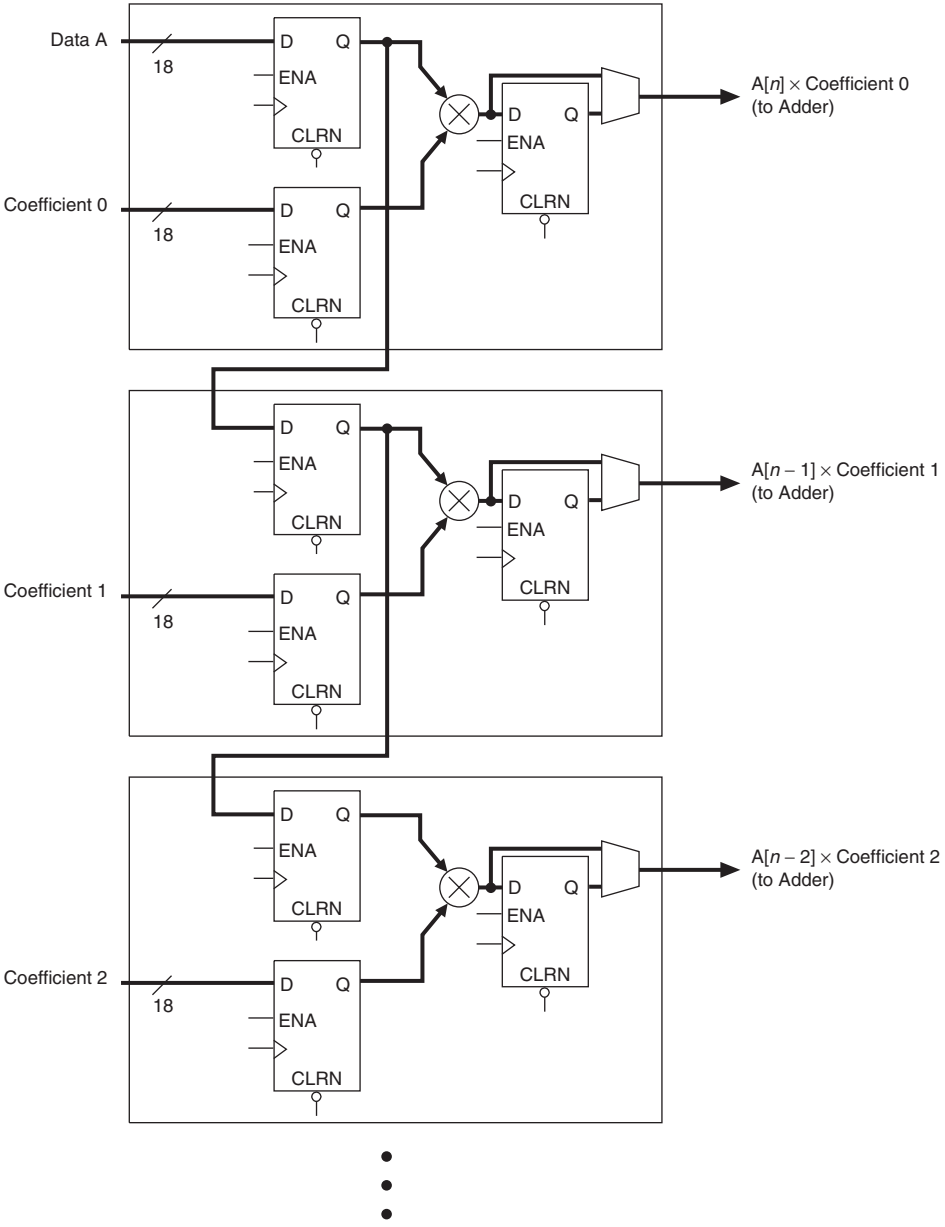
Notes to Figure 10–13:

(1) These signals are not registered or registered once to match the data path pipeline.
(2) You should send these signals through the pipeline register to match the latency of the data path.
(3) These signals match the latency of the data path.
(4) The rounding and saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

FIR Filter

The four-multiplier adder mode can be used to implement FIR filter and complex FIR filter applications. To do this, the DSP block is set up in a four-multiplier adder mode with one set of input registers configured as shift registers using the dedicated shift register chain. The set of input registers configured as shift registers will contain the input data while the inputs configured as regular inputs will hold the filter coefficients. Figure 10–14 shows the DSP block configured in the four-multiplier adder mode using input shift registers.

Figure 10–14. FIR Filter Implemented Using the Four-Multiplier Adder Mode with Input Shift Registers



The built-in input shift register chain within the DSP block eliminates the need for shift registers externally to the DSP block in logic elements (LEs). This architecture feature simplifies the filter design and improves the filter performance because all the filter circuitry is localized within the DSP block.

Input shift registers for the 36-bit simple multiplier mode have to be implemented using external registers to the DSP block.

A single DSP block can implement a four tap 18-bit FIR filter. For filters larger than four taps, the DSP blocks can be cascaded with additional adder stages implemented using LEs.

Software Support

Altera provides two distinct methods for implementing various modes of the DSP block in your design: instantiation and inference. Both methods use the following three Quartus II megafunctions:

- `lpm_mult`
- `altmult_add`
- `altmult_accum`

You can instantiate the megafunctions in the Quartus II software to use the DSP block. Alternatively, with inference, you can create a HDL design and synthesize it using a third-party synthesis tool like LeonardoSpectrum™ or Synplify or Quartus II Native Synthesis that infers the appropriate megafunction by recognizing multipliers, multiplier adders, and multiplier accumulators. Using either method, the Quartus II software maps the functionality to the DSP blocks during compilation.

See Quartus II On-Line Help for instructions on using the megafunctions and the MegaWizard Plug-In Manager.

For more information, see the *Design and Synthesis* section in volume 1 of the *Quartus II Development Software Handbook*.

Conclusion

The Arria GX device DSP blocks are optimized to support DSP applications requiring high data throughput such as FIR filters, FFT functions and encoders. These DSP blocks are flexible and can be configured to implement one of several operational modes to suit a particular application. The built-in shift register chain, adder/subtractor/accumulator block and the summation block minimizes the amount of external logic required to implement these functions, resulting in efficient resource utilization and improved performance and data throughput for DSP applications. The Quartus II

software, together with the LeonardoSpectrum™ and Synplify software provide a complete and easy-to-use flow for implementing these multiplier functions in the DSP blocks.

Referenced Documents

This chapter references the following documents:

- *AN 306: Implementing Multipliers in FPGA Devices*
- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*
- *Design and Synthesis* section in volume 1 of the *Quartus II Development Software Handbook*

Document Revision History

Table 10–6 shows the revision history for this chapter.

Table 10–6. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2008, v1.2	Minor text edits.	—
August 2007, v1.1	Added the “Referenced Documents” section.	—
	Minor text edits.	—
May 2007, v1.0	Initial Release	—