6. DSP Blocks in Stratix & Stratix GX Devices

S52006-2.2

Introduction

Traditionally, designers had to make a trade-off between the flexibility of off-the-shelf digital signal processors and the performance of custom-built devices. Altera® Stratix® and Stratix GX devices eliminate the need for this trade-off by providing exceptional performance combined with the flexibility of programmable logic devices (PLDs). Stratix and Stratix GX devices have dedicated digital signal processing (DSP) blocks, which have high-speed parallel processing capabilities, that are optimized for DSP applications. DSP blocks are ideal for implementing DSP applications that need high data throughput.

The most commonly used DSP functions are finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, discrete cosine transform (DCT) functions, and correlators. These functions are the building blocks for more complex systems such as wideband code division multiple access (W-CDMA) basestations, voice over Internet protocol (VoIP), and high-definition television (HDTV).

Although these functions are complex, they all use similar building blocks such as multiply-adders and multiply-accumulators. Stratix and Stratix GX DSP blocks combine five arithmetic operations—multiplication, addition, subtraction, accumulation, and summation—to meet the requirements of complex functions and to provide improved performance.

This chapter describes the Stratix and Stratix GX DSP blocks, and explains how you can use them to implement high-performance DSP functions. It addresses the following topics:

- Architecture
- Operational Modes
- Software Support

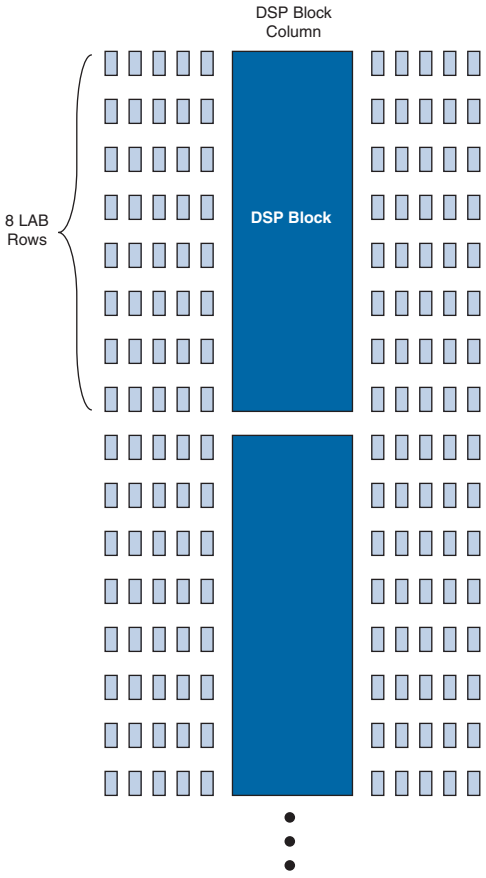
See the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* and the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* for more information on Stratix and Stratix GX devices, respectively.

DSP Block Overview

Each Stratix and Stratix GX device has two columns of DSP blocks that efficiently implement multiplication, multiply accumulate (MAC), and filtering functions. Figure 6–1 shows one of the columns with surrounding LAB rows. You can configure each DSP block to support:

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

Figure 6–1. DSP Blocks Arranged in Columns



The multipliers can then feed an adder or an accumulator block, depending on the DSP block operational mode. Additionally, you can use the DSP block input registers as shift registers to implement applications such as FIR filters efficiently. The number of DSP blocks per column

increases with device density. Tables 6–1 and 6–2 describe the number of DSP blocks in each Stratix and Stratix GX device, respectively, and the multipliers that you can implement.

Table 6–1. Number of DSP Blocks in Stratix Devices Note (1)

Device	DSP Blocks	9 × 9 Multipliers	18 × 18 Multipliers	36 × 36 Multipliers
EP1S10	6	48	24	6
EP1S20	10	80	40	10
EP1S25	10	80	40	10
EP1S30	12	96	48	12
EP1S40	14	112	56	14
EP1S60	18	144	72	18
EP1S80	22	176	88	22

Table 6–2. Number of DSP Blocks in Stratix GX Devices Note (1)

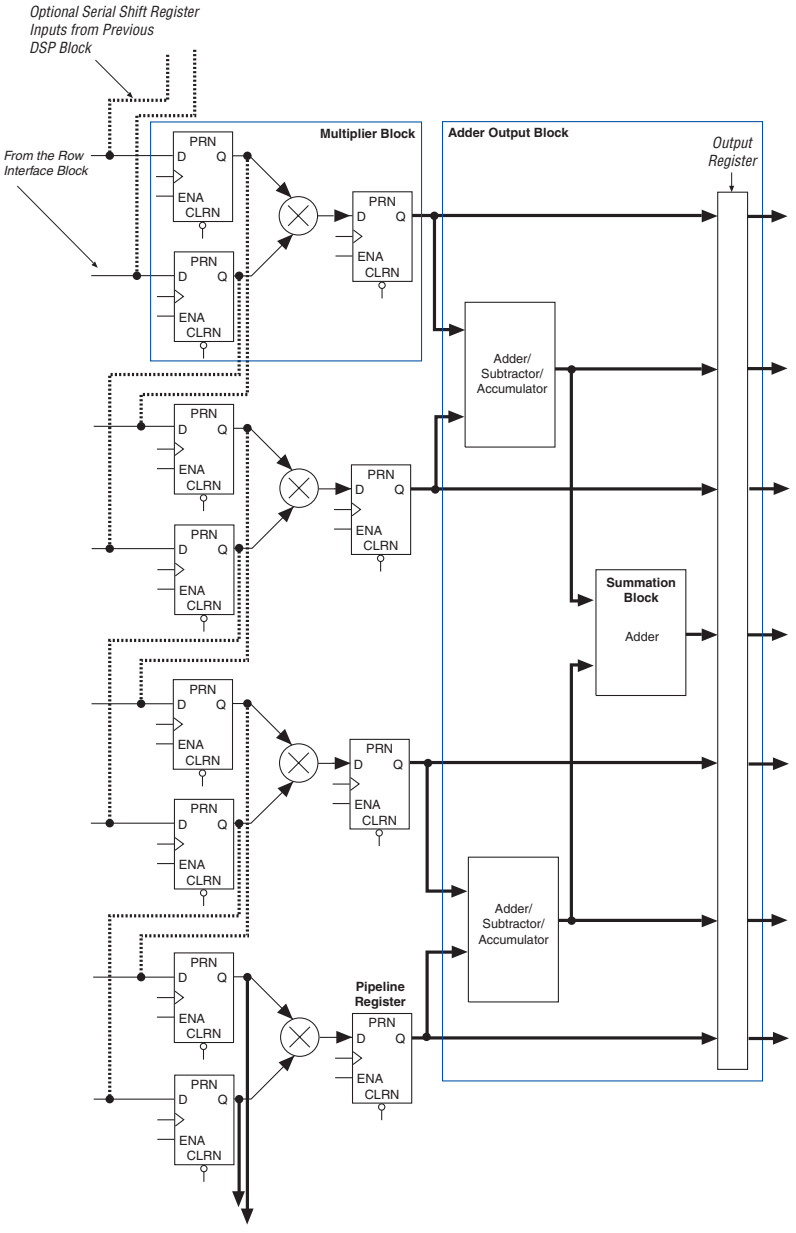
Device	DSP Blocks	9 × 9 Multipliers	18 × 18 Multipliers	36 × 36 Multipliers
EP1SGX10C	6	48	24	6
EP1SGX10D	6	48	24	6
EP1SGX25C	10	80	40	10
EP1SGX25D	10	80	40	10
EP1SGX25F	10	80	40	10
EP1SGX40D	14	112	56	14
EP1SGX40G	14	112	56	14

Note to Tables 6–1 and 6–2:

(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.

Figure 6–2 shows the DSP block operating as an 18 × 18 multiplier.

Figure 6–2. DSP Block in 18 × 18 Mode



Architecture

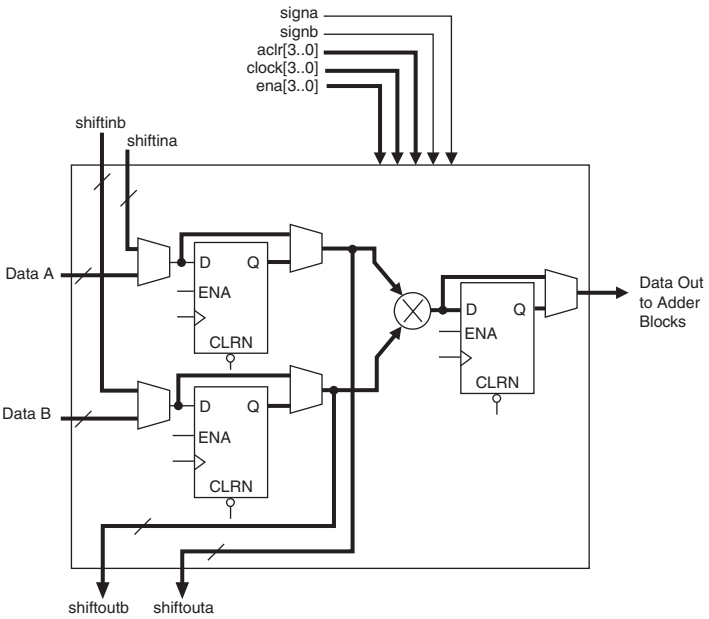
The DSP block consists of the following elements:

- A multiplier block
- An adder/subtractor/accumulator block
- A summation block
- An output interface
- Output registers
- Routing and control signals

Multiplier Block

Each multiplier block has input registers, a multiplier stage, and a pipeline register. See Figure 6–3.

Figure 6–3. Multiplier Block Architecture



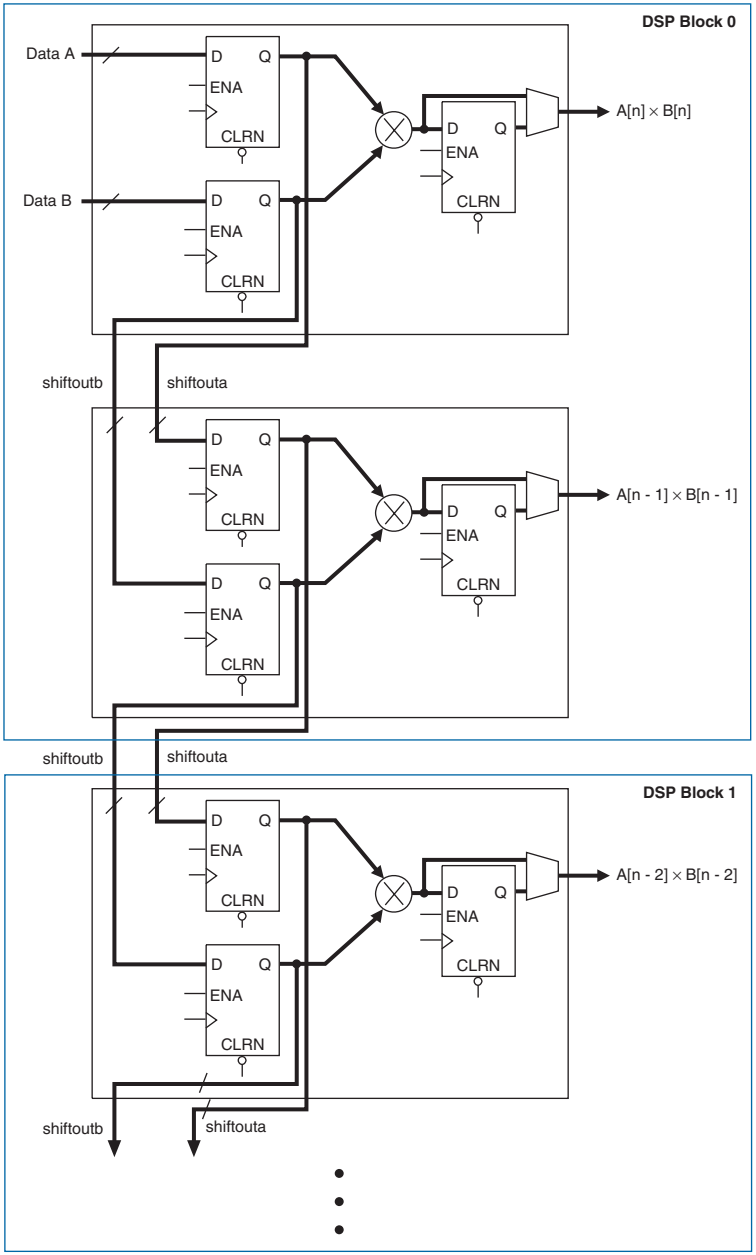
Input Registers

Each operand feeds an input register or the multiplier directly. The DSP block has the following signals (one of each controls every input and output register):

- `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 shift outputs from one multiplier block directly feed the adjacent multiplier block in the same DSP block (or the next DSP block), as shown in Figure 6–4 on page 6–7, to form a shift register chain. This chain can terminate in any block, i.e., you can create any length of shift register chain up to 224 registers. A shift register is useful in DSP applications 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 greatly reduces the required LE count and routing resources, and produces repeatable timing.

Figure 6–4. Shift Register Chain



Multiplier Stage

The multiplier stage supports 9 × 9, 18 × 18, or 36 × 36 multiplication. (The multiplier stage also support smaller multipliers. See "Operational Modes" on page 6–18 for details.) Based on the data width, a single DSP block can perform many multiplications in parallel.

The multiplier operands can be signed or unsigned numbers. Two signals, `signa` and `signb`, indicate the representation of the two operands. For example, a logic 1 on the `signa` signal indicates that data A is a signed number; a logic 0 indicates an unsigned number. The result of the multiplication is signed if any one of the operands is a signed number, as shown in Table 6–3.

Table 6–3. Multiplier Signed Representation

Data A	Data B	Result
Unsigned	Unsigned	Unsigned
Unsigned	Signed	Signed
Signed	Unsigned	Signed
Signed	Signed	Signed

The `signa` and `signb` signals affect the entire 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.

By default, the Altera Quartus® II software sets the multiplier to perform unsigned multiplication when the `signa` and `signb` signals are not used.

Pipeline Registers

The output from the multiplier can feed a pipeline register or be bypassed. You can use pipeline registers for any multiplier size; pipelining is useful for increasing the DSP block performance, particularly when using subsequent adder stages.

In the DSP block, pipelining improves the performance of 36 × 36 multipliers. For 18 × 18 multipliers and smaller, pipelining adds latency but does not improve performance.

Adder/Output Block

The adder/output block has the following elements (See Figure 6–5 on page 6–10):

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

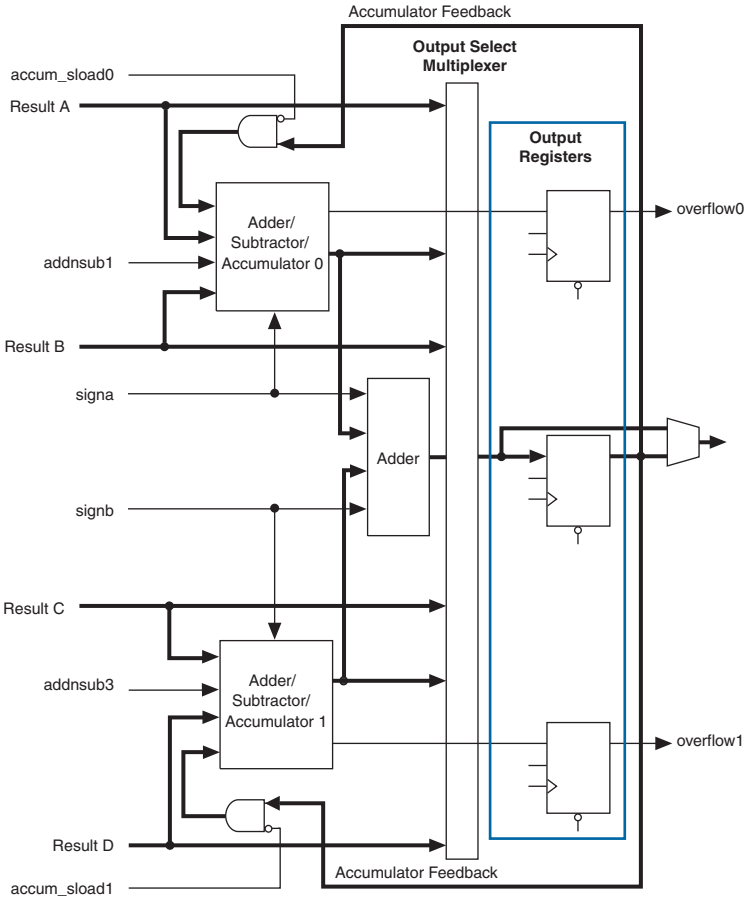
You can configure the adder/output block as:

- A pure output interface
- An accumulator
- A simple 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

The output select multiplexer sets the output of the DSP block. You can register the adder/output block's output using the output registers.

You cannot use the adder/output block independently of the multiplier.

Figure 6–5. Adder/Output Block



Adder/Subtractor/Accumulator Block

The adder/subtractor/accumulator is the first level of the adder/output block. You can configure the block 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 6–5. You can use the

accum_sload[1..0] signals to clear the accumulator asynchronously. This action is not the same as resetting the output registers. You can clear the accumulation and begin a new one without losing any clock cycles.

The overflow signal goes high on the positive edge of the clock when the accumulator overflows or underflows. In the next clock cycle, however, the overflow signal resets to zero even though an overflow (or underflow) occurred in the previous clock cycle. Use a latch to preserve the overflow condition indefinitely (until the latch is cleared).

Adder/Subtractor

The addnsub[1..0] signals select addition or subtraction: high for addition and low for subtraction. You can control the addnsub[1..0] signals using external logic; therefore, the first-level block can switch from an adder to a subtractor dynamically, simply by changing the addnsub[1..0] signals. If the first stage is configured as a subtractor, the output is A - B and C - D.

The adder/subtractor also uses two signals, signa and signb, like the multiplier block. These signals indicate the sign representation of both operands together. You can register the signals with a latency of 1 or 2 clock cycles.

Summation Block

The output from the adder/subtractor feeds to an optional summation block, which is an adder block that sums the outputs of the adder/subtractor. The summation block is important in applications such as FIR filters.

Output Select Multiplexer

The outputs from the various elements of the adder/output block are routed through an output select multiplexer. Based on the DSP block operational mode, the outputs of the multiplier block, adder/subtractor/accumulator, or summation block feed straight to the output, bypassing the remaining blocks in the DSP block.

The output select multiplier configuration is configured automatically by software.

Output Registers

You can use the output registers to register the DSP block output. Like the input registers, the output registers are controlled by the four clock[3..0], aclr[3..0], and ena[3..0] signals. You can use the output registers in any DSP block operational mode.

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

Routing Structure & Control Signals

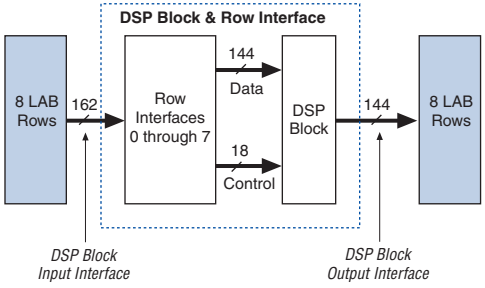
This section describes the interface between the DSP blocks and the row interface blocks. It also describes how the DSP block generates control signals and how the signals route from the row interface to the DSP block.

DSP Block Interface

The DSP blocks are organized in columns, which provides efficient horizontal communication between the blocks and the column-based memory blocks. The DSP block communicates with other parts of the device through an input and output interface. Each DSP block, including the input and output interface, is 8 logic array blocks (LABs) long.

The DSP block and row interface blocks consist of eight blocks that connect to eight adjacent LAB rows on the left and right. Each of the eight blocks has two regions: right and left, one per row. The DSP block receives 144 data input signals and 18 control signals for a total of 162 input signals. This block drives out 144 data output signals; 2 of the data signals can be used as overflow signals (`overflow`). Figure 6–6 provides an overview of the DSP block and its interface to adjacent LABs.

Figure 6–6. DSP Block Interface to Adjacent LABs



Input Interface
The DSP block input interface has 162 input signals from adjacent LABs; 18 data signals per row and 18 control signals per block.

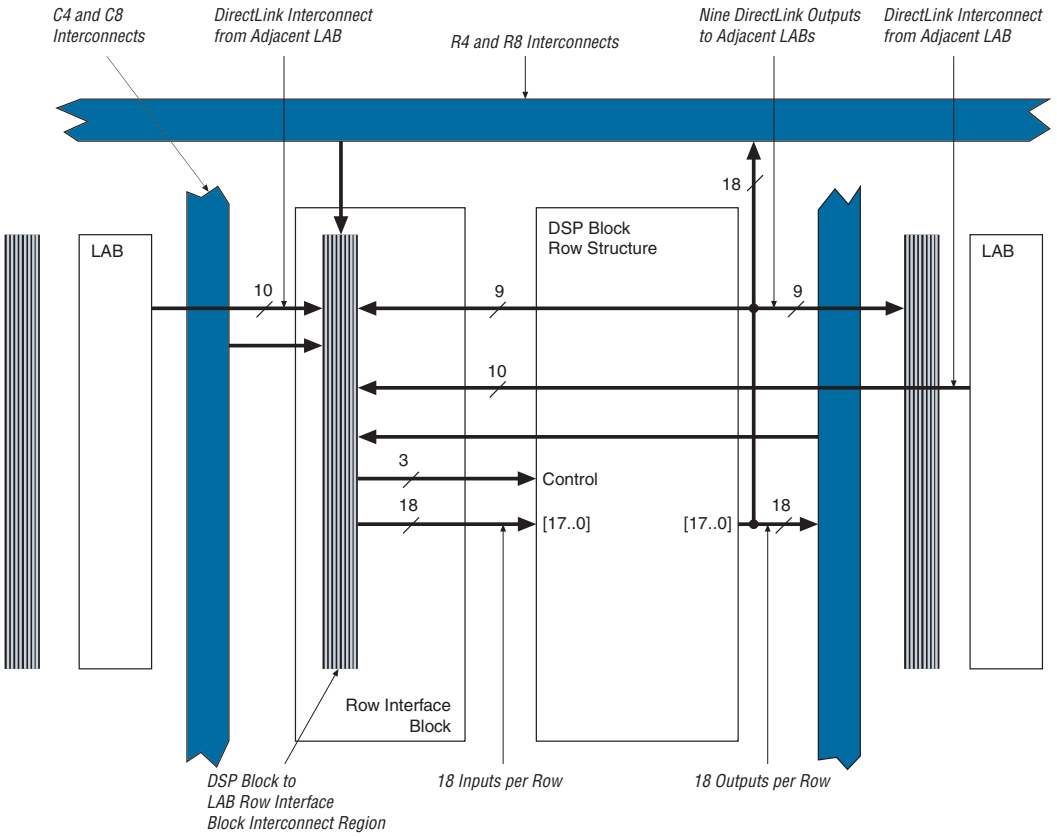
Output Interface
The DSP block output interface drives 144 outputs to adjacent LABs, 18 signals per row from 8 rows.

Because the DSP block outputs communicate horizontally, and because each DSP block row has more outputs than an LAB (18 from the DSP block compared to 10 from an LAB), the DSP block has double the number of row channel drivers compared to an LAB. The DSP block has the same number of row channels, but the row channels are staggered as if there were two LABs within each block. The DSP blocks have the same number of column channels as LABs because DSP blocks communicate primarily through row channels.

Row Interface Block

Each row interface block connects to the DSP block row structure with 21 signals. Because each DSP block has eight row interface blocks, this block receives 162 signals from the eight row interfaces. Of the 162 signals, 144 are data inputs and 18 are control signals. Figure 6–7 on page 6–14 shows one row block within the DSP block.

Figure 6–7. DSP Row Interface Block



Control Signals in the Row Interface Block

The DSP block has a set of input registers, a pipeline register, and an output register. Each register is grouped in banks that share the same clock and clear resources:

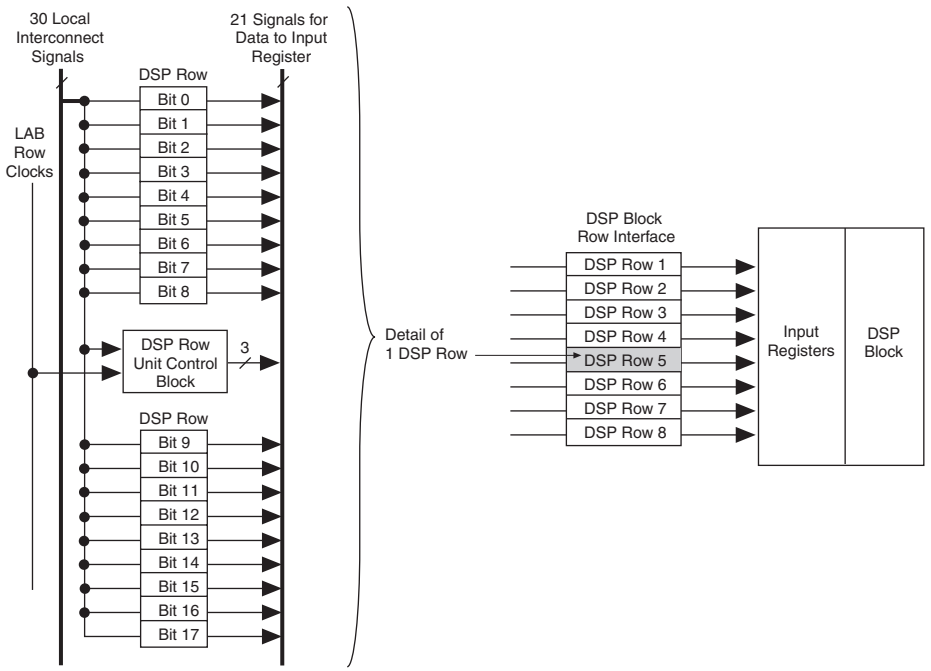
- 1- to 9-bit banks for the input register
- 1- to 18-bit banks for the pipeline register
- 18 bits for the output register

The row interface block generates the control signals and routes them to the DSP block. Each DSP block has 18 control signals:

- Four clock signals (`clock[3..0]`), which are available to each bank of DSP blocks
- Four clear signals (`aclr[3..0]`), which are available to each bank of DSP blocks
- Four clock enable signals (`ena[3..0]`), which the whole DSP block can use
- `signa` and `signb`, which are specific to each DSP block
- `addnsub[1..0]` signals
- `accum_sload[1..0]` signals

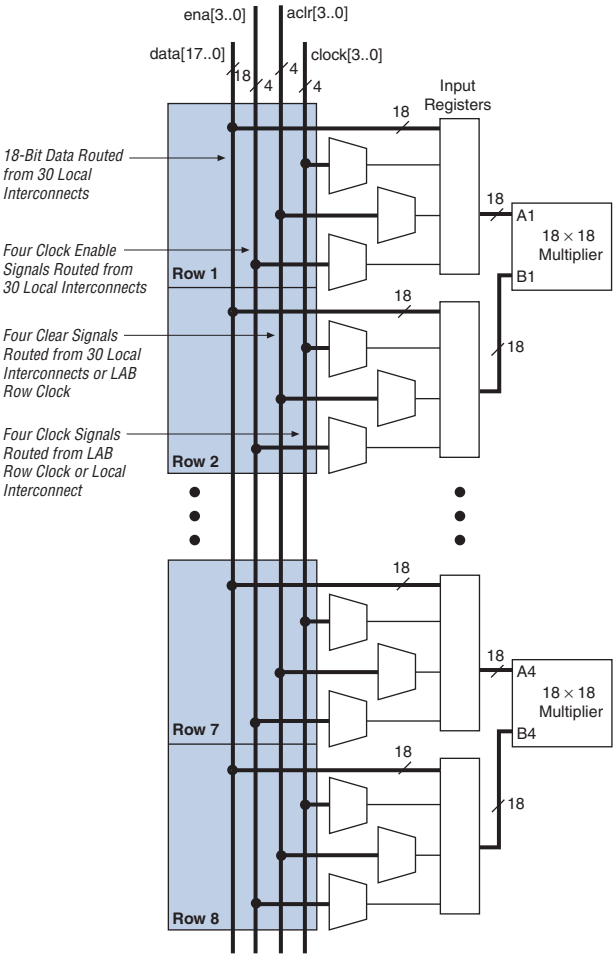
The `signa`, `signb`, and `addnsub[1..0]`, `accum_sload[1..0]` signals have independent clocks and clears and can be registered individually. When each 18 × 18 multiplier in the DSP block splits in half to two 9 × 9 multipliers, each 9 × 9 multiplier has independent control signals. Figure 6–8 shows the DSP block row interface and shows how it generates the data and control signals.

Figure 6–8. DSP Block Row Interface



The DSP block interface generates the clock signals from LAB row clocks or the local interconnect. The clear signals are generated from the local interconnects within each DSP block row interface or from LAB row clocks. The four clock enable signals are generated from the 30 local interconnects from the same LAB rows that generate the clock signals. The clock enable is paired with the clock because the enable logic is implemented at the interface. Figure 6–9 shows the signal distribution within the row interface block.

Figure 6–9. DSP Block Row Interface Signal Distribution



Each row block provides 18 bits of data to the multiplier (i.e., one of the operands to the multiplier), which are routed through the 30 local interconnects within each DSP row interface block. Any signal in the device can be the source of the 18-bit multiplier data, by connecting to the local row interconnect through any row or column.

Each control signal routes through one of the eight rows of the DSP block. Table 6–4 shows the 18 control signals and the row to which each one routes.

Table 6–4. Control Signals in DSP Block

Signal Name	Row	Description
signa	1	DSP block-wide signed and unsigned control signals for all multipliers. The multiplier outputs are unsigned only if both signa and signb are low.
signb	6	
addnsub1	3	Controls addition or subtraction of the two one-level adders. The addnsub0 signal controls the top two one-level adders; the addnsub1 signal controls the bottom two one-level adders. A high indicates addition; a low indicates subtraction.
addnsub3	7	
accum_sload0	2	Resets the feedback input to the accumulator. The signal asynchronously clears the accumulator and allows new accumulation to begin without losing any clock cycles. The accum_sload0 controls the top two one-level adders, and the accum_sload1 controls the bottom two one-level adders. A low is for normal accumulation operations and a high is for zeroing the accumulator.
accum_sload1	7	
clock0	3	DSP block-wide clock signals.
clock1	4	
clock2	5	
clock3	6	
aclr0	1	DSP block-wide clear signals.
aclr1	4	
aclr2	5	
aclr3	7	
ena[3..0]	Same rows as the Clock Signals	DSP block-wide clock enable signals.

Input/Output Data Interface Routing

The 30 local interconnects generate the 18 inputs to the row interface blocks. The 21 outputs of the row interface block are the inputs to the DSP row block (see Figure 6–7 on page 6–14).

The row interface block has DirectLink™ connections that connect the DSP block input or output signals to the left and right adjacent LABs at each row. (The DirectLink connections provide interconnects between LABs and adjacent blocks.) The DirectLink connection reduces the use of row and column interconnects, providing higher performance and flexibility.

Each row interface block receives 10 DirectLink connections from the right adjacent LABs and 10 from the left adjacent LABs. Additionally, the row interface block receives signals from the DSP block, making a total of 30 local interconnects for each row interface block. All of the row and column resources within the DSP block can access this interconnect region (see Figure 6–7 on page 6–14).

A DSP block has nine outputs that drive the right adjacent LAB and nine that drive the left adjacent LAB through DirectLink interconnects. All 18 outputs drive any row or column.

Operational Modes

You can use the DSP block in one of four operational modes, depending on your application needs (see Table 6–4). The Quartus II software has built-in megafunctions that you can use to control the mode. After you have made your parameter settings using the megafunction's MegaWizard® Plug-In, the Quartus II software automatically configures the DSP block.

Table 6–5. DSP Block Operational Modes

Mode	9 × 9	18 × 18	36 × 36
Simple multiplier	Eight multipliers with eight product outputs	Four multipliers with four product outputs	One multiplier
Multiply accumulator	Two 34-bit multiply-accumulate blocks	Two 52-bit multiply-accumulate blocks	–
Two-multiplier adder	Four two-multiplier adders	Two two-multiplier adders	–
Four-multiplier adder	Two four-multiplier adders	One four-multiplier adder	–

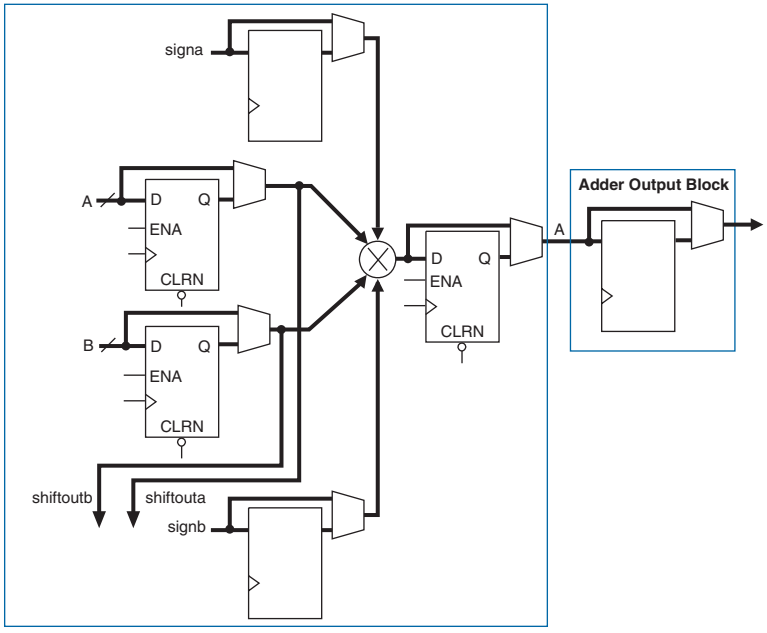
Simple Multiplier Mode

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

9- & 18-Bit Multipliers

You can configure each DSP block multiplier for 9 or 18 bits. A single DSP block can support up to 8 individual 9-bit or smaller multipliers, or up to 4 individual multipliers with operand widths between 10- and 18-bits. Figure 6–10 shows the simple multiplier mode.

Figure 6–10. Simple Multiplier Mode



The multiplier operands can accept signed integers, unsigned integers, or a combination. The `signa` and `signb` signals are dynamic and can be registered in the DSP block. Additionally, you can register the multiplier inputs and results independently. Pipelining the result, using the pipeline registers in the block, increases the performance of the DSP block.

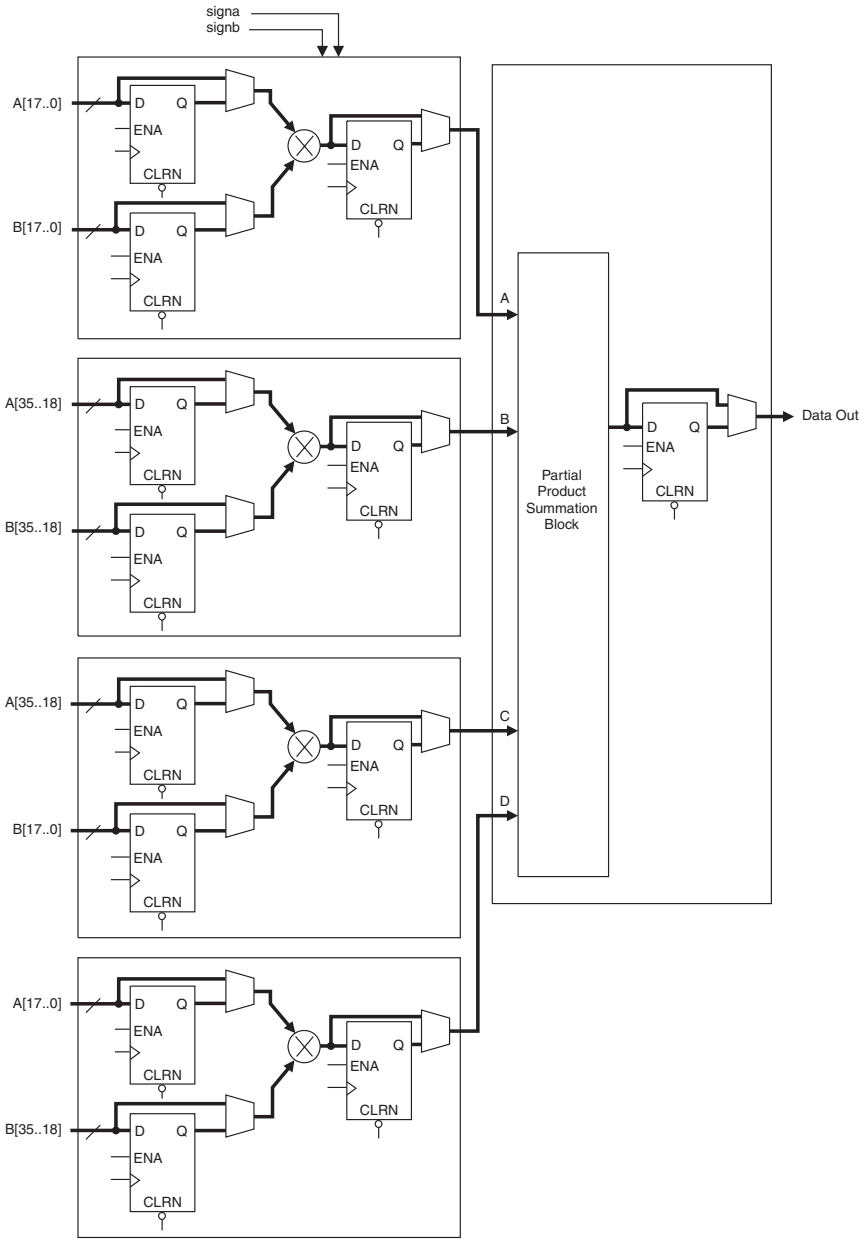
36-Bit Multiplier

The 36-bit multiplier is a subset of the simple multiplier mode. It uses the entire DSP block to implement one 36 × 36-bit multiplier. The four 18-bit multipliers are fed part of each input, as shown in Figure 6–11 on page 6–21. The adder/output block adds the partial products using the

summation block. You can use pipeline registers between the multiplier stage and the summation block. The 36 × 36-bit multiplier supports signed and unsigned operation.

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

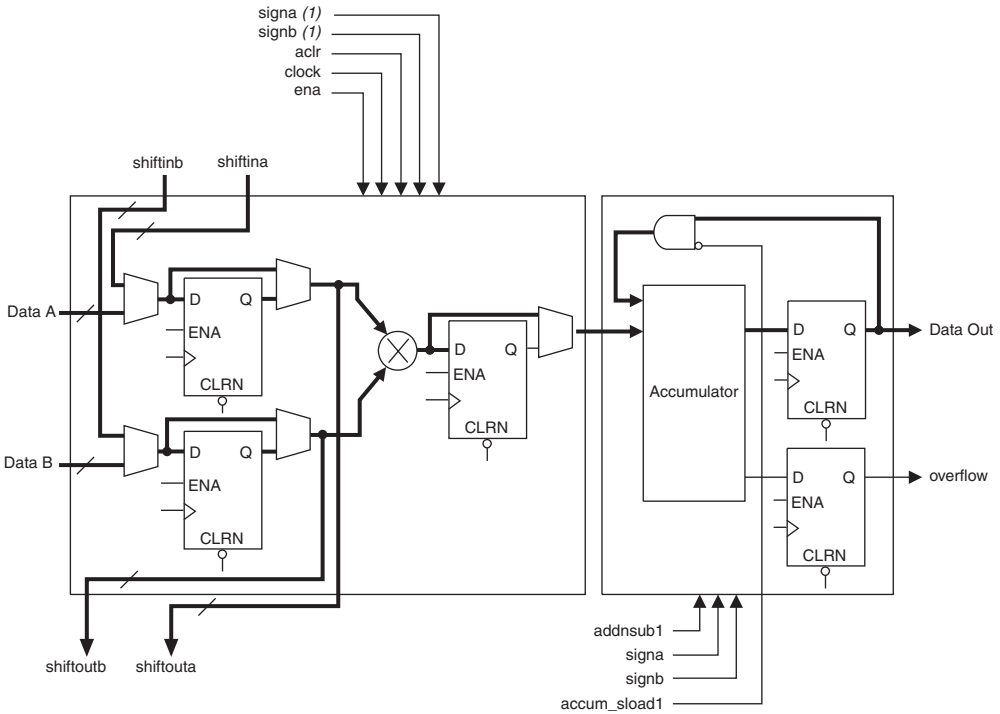
Figure 6–11. 36-Bit Multiplier



Multiply Accumulator Mode

In multiply accumulator mode, the output of the multiplier stage feeds the adder/output block, which is configured as an accumulator or subtractor (see Figure 6–12). You can implement up to two independent 18-bit multiply accumulators in one DSP block. The Quartus II software implements smaller multiplier-accumulators by tying the unused low-order bits of an 18-bit multiplier to ground.

Figure 6–12. Multiply Accumulator Mode



Note to Figure 6–12:
(1) The `signa` and `signb` signals are the same in the multiplier stage and the adder/output block.

The multiply accumulator output can be up to 52 bits wide for a maximum 36-bit 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[1..0]` signal synchronously loads the multiplier result to the accumulator output. This signal can be unregistered or registered once or twice. The DSP block can then begin a new accumulation without losing any clock cycles. The `overflow` signal indicates an overflow or underflow in the accumulator. This signal is

cleared for the next accumulation cycle, and you can use an external latch to preserve the signal. You can use the `addnsub[1..0]` signals to perform accumulation or subtraction dynamically.

If you want to use DSP blocks and your design only has an accumulator, you can use a multiply by one followed by an accumulator to force the software to implement the logic in the DSP block.

Two-Multiplier Adder Mode

The two-multiplier adder mode uses the adder/output block to add or subtract the outputs of the multiplier block, which is useful for applications such as FFT functions and complex FIR filters. Additionally, in this mode, the DSP block outputs two sums or differences for multipliers up to 18 bits, or 4 sums or differences for 9-bit or smaller multipliers. A single DSP block can implement one 18 × 18-bit complex multiplier or two 9 × 9-bit complex multipliers.

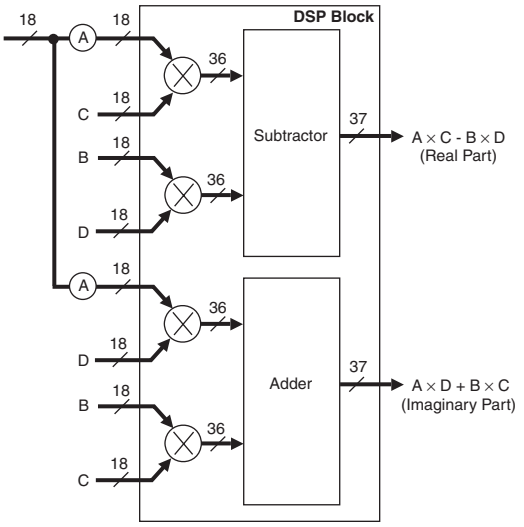
A complex multiplication can be written as:

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

In this mode, a single DSP block calculates the real part ($a \times c - b \times d$) using one adder/subtractor/accumulator and the imaginary part ($a \times d + b \times c$) using another adder/subtractor/accumulator for data up to 18 bits. Figure 6–13 shows an 18-bit complex multiplication. For data widths up to 9 bits, the DSP block can perform two complex multiplications using four one-level adders. Resources outside of the DSP block route each input to the two multiplier inputs.

You can only use the adder block if it follows multiplication operations.

Figure 6–13. Complex Multiplier Implemented Using Two-Multiplier Adder Mode



Four-Multiplier Adder Mode

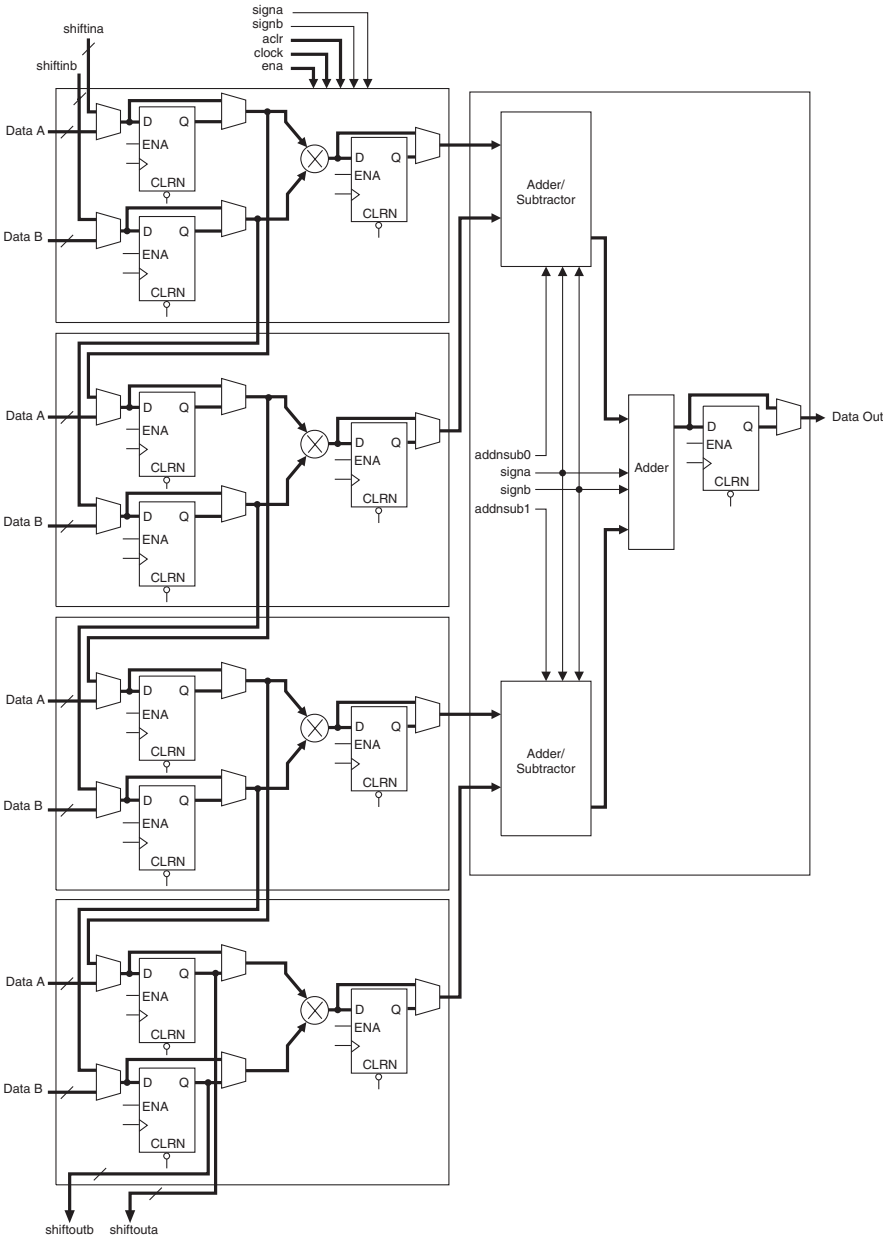
In the four-multiplier adder mode, which you can use for 1-dimensional and 2-dimensional filtering applications, the DSP block adds the results of two adder/subtractor/accumulators in a final stage (the summation block).

You can only use the adder block if it follows multiplication operations.

9- & 18-Bit Summation Blocks

A single DSP block can implement one 18 × 18 or two 9 × 9 summation blocks (see Figure 6–14 on page 6–25). The multiplier product widths must be the same size.

Figure 6–14. Four-Multiplier Adder Mode



FIR Filters

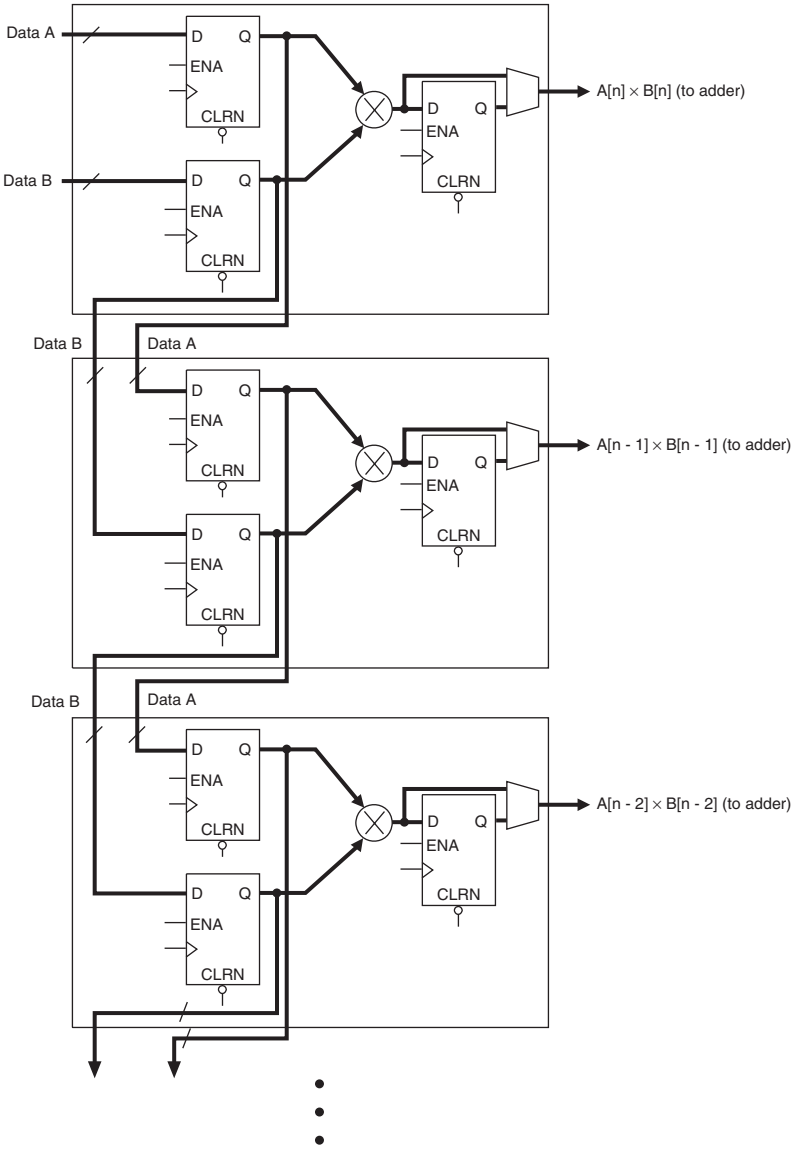
The four-multiplier adder mode can be used for FIR filter and complex FIR filter applications. The DSP block combines a four-multiplier adder with the input registers configured as shift registers. One set of shift inputs contains the filter data, while the other holds the coefficients, which can be loaded serially or in parallel (see Figure 6–15).

The input shift register eliminates the need for shift registers external to the DSP block (e.g., implemented in device logic elements). This architecture simplifies filter design and improves performance because the DSP block implements all of the filter circuitry.

Serial shift inputs in 36-bit simple multiplier mode require external registers.

One DSP block can implement an entire 18-bit FIR filter with up to four taps. For FIR filters larger than four taps, you can cascade DSP blocks with additional adder stages implemented in logic elements.

Figure 6–15. Input Shift Registers Configured for a FIR Filter



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 an 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 the *Implementing High-Performance DSP Functions in Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook, Volume 2* or the *Stratix GX Device Handbook, Volume 2* for more information on using DSP blocks to implement high-performance DSP functions such as FIR filters, IIR filters, and discreet cosine transforms (DCTs).

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

For more information on DSP block inference support, see the *Recommended HDL Coding Styles* chapter of the *Quartus II Development Software Handbook v4.1, Volume 1*.

Conclusion

The Stratix and Stratix GX device DSP blocks are optimized to support DSP applications that need high data throughput, such as FIR filters, FFT functions, and encoders. These DSP blocks are flexible and can be configured in one of four operational modes to suit any application need. The DSP block's adder/subtractor/accumulator and the summation blocks minimize the amount of logic resources used and provide efficient routing. This efficiency results in improved performance and data throughput for DSP applications. The Quartus II software, together with the LeonardoSpectrum and Synplify software, provides a complete and easy-to-use flow for implementing functionality in the DSP block.