XtremeDSP DSP48A for Spartan-3A DSP FPGAs

User Guide

UG431 (v1.3) July 15, 2008

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/02/07	1.0	Initial Xilinx release.
05/29/07	1.1	Modified the title and figures in Chapter 2.
11/05/07	1.2	• Figure 1-1: New DSP48A layout figure for the XC3SD1800A device. • Figure 1-2: New DSP48A layout figure for the XC3SD3400A device. • Table 1-2: Edited the Function Description for signal B and BCIN. • Figure 1-5: Added new note 9 to the Notes below the figure. • Figure 1-8: Connected CEB pin to EN pin of B1 register. • Table 1-8: Changed silicon utilization for 48-bit adder/subtracter to 1 DSP slice. • Chapter 1: Cosmetic edits. • Updated "VHDL and Verilog Instantiation Templates."
07/15/08	1.3	• Added "Performance" in Chapter 1.

Table of Contents

Schedule of Figures

Schedule of Tables

About This Guide

This user guide is a detailed functional description of the XtremeDSP™ DSP48A slice for Spartan™-3A DSP FPGAs.

Guide Contents

This manual contains the following chapters:

- Chapter 1, "XtremeDSP Design Considerations," introduces the DSP48A slice, its elements, and its applications.
- Chapter 2, "Using the DSP48A Pre-Adder," describes how the pre-adder increases the density, performance, and power efficiency of symmetric FIR filters and complex multipliers.

Additional Resources

To find additional documentation, see the Xilinx website at:

www.xilinx.com/literature.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support Web Case, see the Xilinx website at:

www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*

Convention	Meaning or Use	Example
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr ={on|off}`**
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis . . .	Repetitive material that has been omitted	**`allow block`** *block_name loc1 loc2 ... locn;*

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Platform FPGA User Guide.*
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

Chapter 1

XtremeDSP Design Considerations

This chapter provides technical details for the XtremeDSP™ Digital Signal Processing (DSP) element, the DSP48A slice.

The DSP48A slice is unique to the Spartan™-3A DSP family of FPGAs. Each XtremeDSP slice contains a single DSP48A slice to form the basis of a versatile coarse-grained DSP architecture. Many DSP designs follow a multiply with addition. In Spartan-3A DSP devices, these elements are supported in dedicated circuits.

The DSP48A slices support many independent functions, including multiplier, multiplier-accumulator (MACC), pre-adder/subtracter followed by a multiply-accumulator, multiplier followed by an adder, wide bus multiplexers, magnitude comparator, or wide counter. The architecture also supports connecting multiple DSP48A slices to form wide math functions, DSP filters, and complex arithmetic without the use of general FPGA fabric.

The DSP48A slices available in all Spartan-3A DSP family members support new DSP algorithms and higher levels of DSP integration than previously available in FPGAs. Minimal use of general FPGA fabric leads to low power, very high performance, and efficient silicon utilization.

This chapter contains the following sections:

Introduction

The DSP48A slices facilitate higher levels of DSP integration than previously possible in FPGAs. The DSP48A is derived from the Virtex®-4 DSP48 slice (refer to UG073, *XtremeDSP for Virtex-4 FPGAs User Guide*, for more information).

Many DSP algorithms are supported with minimal use of the general-purpose FPGA fabric, resulting in low power, high performance, and efficient device utilization. At first look, the DSP48A slice contains an 18-bit input pre-adder followed by an 18 x 18 bit two's complement multiplier and a 48-bit sign-extended adder/subtracter/accumulator, a function that is widely used in digital signal processing (DSP). A second look reveals many subtle features that enhance the usefulness, versatility, and speed of this arithmetic building block. Programmable pipelining of input operands, intermediate products, and accumulator outputs enhances throughput. The 48-bit internal bus allows for practically unlimited aggregation of DSP slices.

One of the most important features is the ability to cascade a result from one XtremeDSP Slice to the next without the use of general fabric routing. This path provides high-performance and low-power post addition for many DSP filter functions of any tap length. Another key feature for filter composition is the ability to cascade an input stream from slice to slice.

The C input port allows the formation of many 3-input mathematical functions, such as 3-input addition by cascading the pre-adder with the post -adder and 2-input multiplication with a single addition. The D input allows a second argument to be used with the pre-adder to reduce XtremeDSP slice utilization in Symmetric filters.

Architecture Highlights

The Spartan-3A DSP slices are organized as vertical DSP columns. Within the DSP column, a single DSP slice is combined with extra logic and routing. The Spartan-3A DSP slice is four CLBs tall.

Each DSP48A slice has a selectable 18 bit pre-adder. This pre-adder accepts 18-bit, two's-complement inputs and generates an 18-bit, two's-complement result. The pre-adder is followed by a two-input multiplier, multiplexers and a two-input adder/subtracter. The multiplier accepts two 18-bit, two's complement operands producing a 36-bit, two's complement result. The result is sign extended to 48 bits and can optionally be fed to the adder/subtracter. The adder/subtracter accepts two 48-bit, two's-complement operands, and produces a 48-bit two's complement result.

Higher-level DSP functions are supported by cascading individual DSP48A slices in a DSP48A column. One input (cascade B input bus) and the DSP48A slice output (cascade P output bus) provide the cascade capability. For example, a Finite Impulse Response (FIR) filter design can use the cascading input to arrange a series of input data samples and the cascading output to arrange a series of partial output results. For details on this technique, refer to "Adder Cascade Versus Adder Tree," page 34.

Architecture highlights of the DSP48A slices are:

- Two-input pre-adder for efficient implementation of symmetric filters
- 18-bit x 18-bit, two's-complement multiplier with a full-precision 36-bit result, sign extended to 48 bits
- Two-input, flexible 48-bit post-adder/subtracter with optional registered accumulation feedback
- Dynamic user-controlled operating modes to adapt DSP48A slice functions from clock cycle to clock cycle
- Cascading 18-bit B bus, supporting input sample propagation
- Cascading 48-bit P bus, supporting output propagation of partial results

- Performance enhancing pipeline options for control and data signals are selectable by configuration bits
- Input port *C* typically used for multiply-add operation, large two-operand addition, or flexible rounding mode
- Separate reset and clock enable for control and data registers
- I/O registers, ensuring maximum clock performance and highest possible sample rates with no area cost

A number of software tools support the DSP48A slice. The Xilinx® ISE® software tools support DSP48A slice instantiations. Synthesis tools such as XST also support DSP48A inference. CORE Generator™, System Generator™ for DSP, and AccelDSP™ use the DSP48A in designs targeting the Spartan®-3A DSP family, making it easier for a designer to quickly generate math, DSP, or similar functions using the Spartan-3A DSP DSP48A slices.

Performance

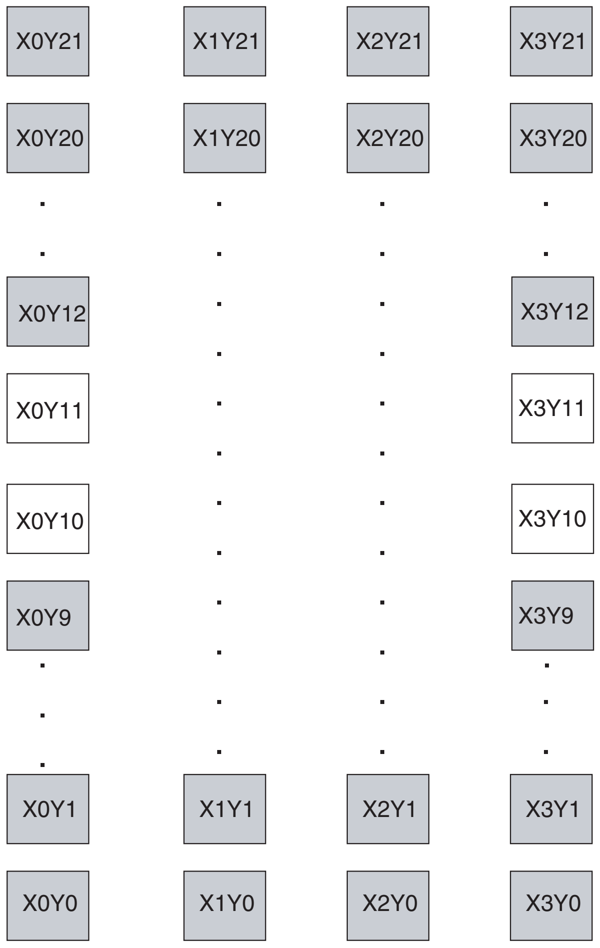
To achieve the maximum performance when using the DSP48A Slice, the users need to fully pipeline their designs. See the timing numbers in the XtremeDSP Switching Characteristics section of the *Spartan-3A DSP FPGA Family: Complete Data Sheet*, for timing information regarding how the pipeline stages affect the performance.

Number of DSP48A Slices per Spartan-3A DSP Device

Table 1-1 shows the number of DSP48A slices for each device in the Spartan-3A DSP families. The Spartan-3A DSP family offers a high ratio of DSP48A slices to logic, making it ideal for math-intensive applications. Figure 1-1 shows the DSP48A layout for the XC3SD1800A FPGA and Figure 1-2 shows the DSP48A layout for the XC3SD3400A FPGA.

Table 1-1: **Number of DSP48A Slices per Family Member**

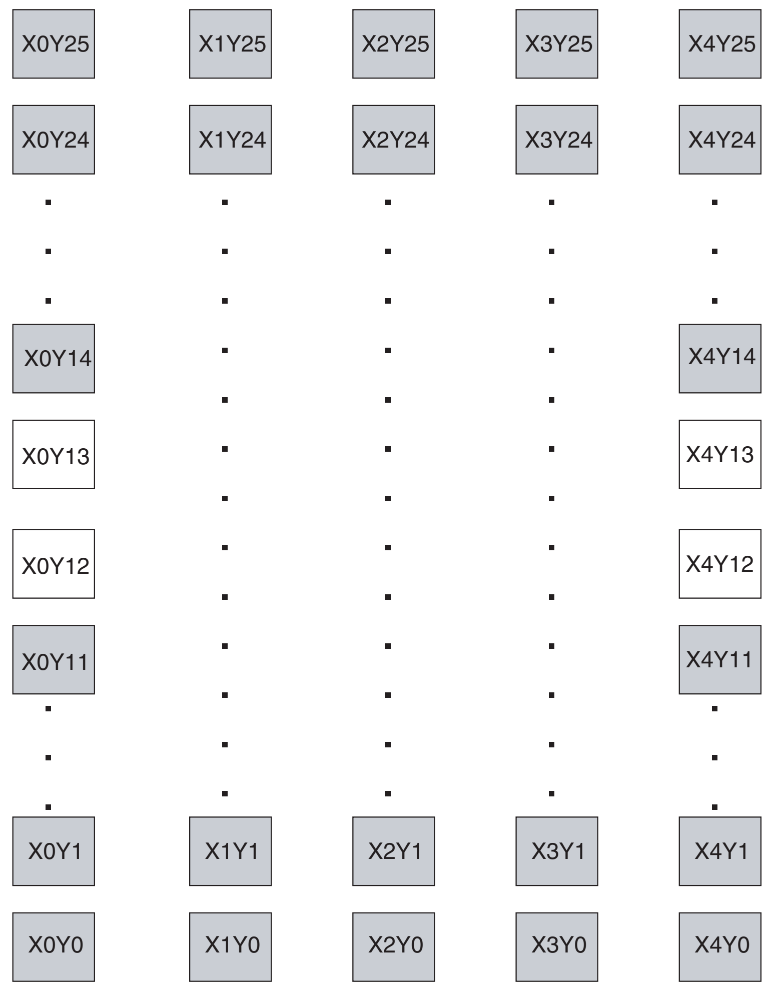
Device	DSP48A	Columns
XC3SD1800A	84	4
XC3SD3400A	126	5



Notes:
1. Gray positions are populated with DSP48As.
2. Clear positions do not contain DSP48As. DSP48A carry chains are also broken in these positions. There are continuous carry chains from X0Y0 to X0Y9, and then from X0Y12 to X0Y21 and from X3Y12 to X3Y21.

ug431_ch1_01_110207

Figure 1-1: **DSP48A Layout for the XC3SD1800A FPGA**



Notes:
1. Gray positions are populated with DSP48As.
2. Clear positions do not contain DSP48As. DSP48A carry chains are also broken in these positions. There are continuous carry chains from X0Y0 to X0Y11, and then from X0Y14 to X0Y25 and from X4Y0 to X4Y11 and from X4Y14 to X4Y25.

ug431_ch1_02_110207

Figure 1-2: **DSP48A Layout for the XC3SD3400A FPGA**

DSP48A Slice Primitive

Figure 1-3 shows the DSP48A slice primitive.

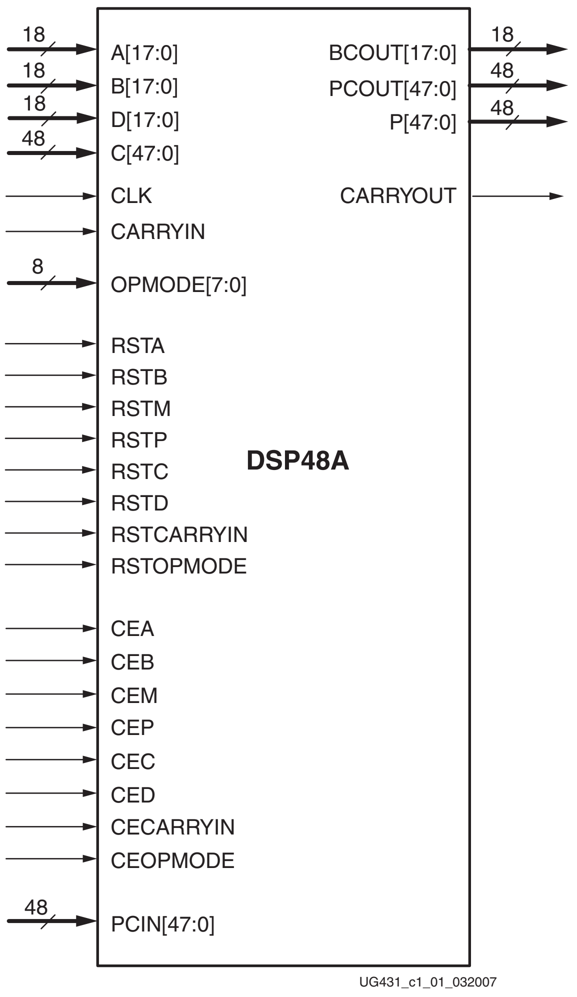


Figure 1-3: **DSP48A Slice Primitive**

Table 1-2 lists the available ports in the DSP48A slice primitive.

Table 1-2: **DSP48A Slice Port List and Definitions**

Signal Name	Direction	Size	Function
Data Ports			
A	Input	18	18-bit data input to multiplier or post adder/subtracter depending on the value of OPMODE[3:0].
B	Input	18	18-bit data input to multiplier, pre-adder/subtracter and, perhaps, a post-adder/subtracter depending on the value of OPMODE[4]. It should also be noted that the DSP48A UNISIM component uses this input when cascading the BCOUT from an adjacent DSP48A. The tools will then translate BCOUT cascade to the BCIN input and set the B_INPUT attribute for implementation.

Table 1-2: **DSP48A Slice Port List and Definitions *(Continued)***

Signal Name	Direction	Size	Function
C	Input	48	48-bit data input to post-adder/subtracter.
D	Input	18	18-bit data input to pre-adder/subtracter.
CARRYIN	Input	1	External carry input to the post-adder/subtracter. Should only be connected to the CARRYOUT pin of another DSP48A block.
P	Output	48	Primary data output.
CARRYOUT	Output	1	Carry out signal for post-adder/subtracter. Should only be connected to the CARRYIN pin of another DSP48A.
Control Input Ports			
CLK	Input	1	DSP48A clock.
OPMODE	Input	8	Control input to select the arithmetic operations of the DSP48A (see OPMODE, Table 1-7).
Reset/Clock Enable Input Ports			
RSTA	Input	1	Active-High reset for the A port registers: A0REG = 1 or A1REG = 1 Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTB	Input	1	Active-High reset for the B port registers: B0REG = 1 or B1REG = 1 Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTC	Input	1	Active-High reset for the C input registers (CREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTD	Input	1	Active-High reset for the D port registers (DREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTM	Input	1	Active-High reset for the multiplier registers (MREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTP	Input	1	Active-High reset for the P output registers (PREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTCARRYIN	Input	1	Active-High reset for the carry-in register (CARRYINREG =1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.
RSTOPMODE	Input	1	Active-High reset for the OPMODE registers (OPMODEREG=1). Tie to logic zero if not used. This reset is configurable to be synchronous or asynchronous depending on the value of the RSTTYPE attribute.

Table 1-2: **DSP48A Slice Port List and Definitions *(Continued)***

Signal Name	Direction	Size	Function
CEA	Input	1	Active-High clock enable for the A port registers: A0REG = 1, A1REG = 1 Tie to logic one if not used and: A0REG = 1, A1REG = 1. Tie to logic zero if A0REG = 0 and A1REG = 0.
CEB	Input	1	Active-High clock enable for the B port registers: B0REG = 1 or B1REG = 1. Tie to logic one if not used and B0REG = 1 or B1REG = 1. Tie to logic zero if B0REG = 0 and B1REG = 0
CEC	Input	1	Active-High, clock enable for the C port registers (CREG=1). Tie to logic one if not used and CREG=1. Tie to a logic zero if CREG=0.
CED	Input	1	Active-High, clock enable for the D port registers (DREG=1). Tie to logic one if not used and DREG=1. Tie to a logic zero if DREG=0.
CEM	Input	1	Active-High, clock enable for the multiplier registers (MREG=1). Tie to logic one if not used and MREG=1. Tie to a logic zero if MREG=0.
CEP	Input	1	Active-High, clock enable for the output port registers (PREG=1). Tie to logic one if not used and PREG=1. Tie to a logic zero if PREG=0.
CECARRYIN	Input	1	Active-High, clock enable for the carry-in registers (CARRYINREG=1). Tie to logic one if not used and CARRYINREG=1. Tie to a logic zero if CARRINREG=0.
CEOPMODE	Input	1	Clock enable for the OPMODE input registers (OPMODEREG=1). Tie to logic one if not used and OPMODEREG=1. Tie to a logic zero if OPMODEREG=0.
Cascade Ports			
PCIN	Input	48	Cascade input for Port P. If used, connect to PCOUT of upstream cascaded DSP48A. If not used, tie port to all zeros.
BCIN	Input	18	Cascade input for Port B. If used, connect to BCOUT of the upstream cascaded DSP48A. If not used, tie port to all zeros. It should also be noted that the UNISIM component does not have this port. Instead the BCOUT should be connected to the B port on the UNISIM component and the tools will translate the BCOUT cascade to the BCIN input and set the B_INPUT attribute for implementation.
PCOUT	Output	48	Cascade output for Port P. If used, connect to PCIN of downstream cascaded DSP48A. If not used, leave unconnected.
BCOUT	Output	18	Cascade output for Port B. If used, connect to the B port of downstream cascaded DSP48A. If not used, leave unconnected.

OPMODE Pin Descriptions

Table 1-3 shows the OPMODE pin descriptions.

Table 1-3: **OPMODE Pin Descriptions**

Port Name	Function
OPMODE[1:0]	**Specifies the source of the X input to the post-adder/subtracter** 0 – Specifies to place all zeroes (disable the post-adder/subtracter) 1 – Use the multiplier product 2 – Use the P output signal (accumulator) 3 – Use the concatenated D, B, A input signals
OPMODE[3:2]	**Specifies the source of the Z input to the post-adder/subtracter** 0 – Specifies to place all zeroes (disable the post-adder/subtracter and propagate the multiplier product to P) 1 – Use the PCIN 2 – Use the P port (accumulator) 3 – Use the C port
OPMODE[4]	**Specifies the use of the pre-adder/subtracter** 0 – Bypass the pre-adder supplying the data on port B directly to the multiplier 1 – Selects to use the pre-adder adding or subtracting the values on the B and D ports prior to the multiplier
OPMODE[5]	**Forces a value on carry-in to the post-adder. Only applicable when CARRYINSEL = OPMODE5**
OPMODE[6]	**Specifies whether the pre-adder/subtracter is an adder or subtracter** 0 – Specifies pre-adder/subtracter to perform an addition operation 1 – Specifies pre-adder/subtracter to perform a subtraction operation
OPMODE[7]	**Specifies whether the post-adder/subtracter is an adder or subtracter** 0 – Specifies post-adder/subtracter to perform an addition operation 1 – Specifies post-adder/subtracter to perform a subtraction operation

DSP48A Slice Attributes

The synthesis attributes for the DSP48A slice are described in detail throughout this section. With the exception of the B_INPUT, RSTTYPE, and CARRYINSEL attributes, all other attributes call out pipeline registers in the control and data paths. The value of the attribute sets the number of pipeline registers.

The attribute settings are as follows:

- The A0REG, A1REG, B0REG, and B1REG attributes can take values of 0 or the values define the number of pipeline registers in the A and B input paths.A0REG defaults to 0 (no register). A1REG defaults to 1 (register). B0REG defaults to 0 (no register) B1REG defaults to 1 (register). See the "A, B, C, D, and P Port Logic" section for more information.
- The CREG, DREG, MREG, and PREG attributes can take a value of 0 or 1. The value defines the number of pipeline registers at the output of the multiplier (MREG) (shown in Figure 1-12) and at the output of the post-adder/subtracter (PREG) (shown

in Figure 1-10). The CREG attribute is used to select the pipeline register at the C input (shown in Figure 1-9). CREG, DREG, MREG, and PREG all default to 1 (registered).

- The CARRYINREG and OPMODEREG attributes take a value of 0 if no pipelining register is on these paths, and they take a value of 1 if there is one pipeline register in their path. The CARRYINSEL and OPMODEREG paths are shown in Figure 1-11, and the CARRYINREG path is shown in Figure 1-13. CARRYINREG and OPMODEREG both take a default value of 1 (registered).
- The CARRYINSEL attribute selects whether the post adder/subtracter carry-in signal should be sourced from the CARRYIN pin (connected to the CARRYOUT of another DSP48A) or dynamically controlled from the FPGA fabric by the OPMODE[5] input. This attribute can be set to the string CARRYIN or OPMODE5. CARRYINSEL defaults to CARRYIN.
- The B_INPUT attribute defines whether the input to the B port is routed from the parallel input (attribute: DIRECT) or the cascaded input from the previous slice (attribute: CASCADE). The attribute is only used by place and route tools and is not necessary for the users to set for synthesis. The attribute is determined by the connection to the B port of DSP48A. If the B port is connected to the BCOUT of anotherDSP48A, then the tools will automatically set the attribute to CASCADE, otherwise it will be set as DIRECT.
- The RSTTYPE attribute selects whether all resets for the DSP48A should have a synchronous or asynchronous reset capability. This attribute can be set to ASYNC or SYNC. Due to improved timing and circuit stability, it is recommended to always have this set to SYNC unless asynchronous reset is absolutely necessary. RSTTYPE defaults to SYNC.

VHDL and Verilog Instantiation Templates

The VHDL and Verilog instantiation templates for the DSP48A slice can be found in the ISE tool through the following choice of menus:

```
ISE -> Edit -> Language Templates -> VHDL || Verilog -> Device
Primtive Instantiation -> Arithmetic Functions -> Spartan-3A DSP
-> DSP BLock (DSP48A)
```

DSP48A Slice and Interconnect

The DSP48A slices stack vertically in a DSP48A column. Refer to Table 1-1. The height of a DSP48A slice is the same as four CLBs and also matches the height of one block RAM. This "regularity" enhances the routing of wide data paths. The smaller Spartan-3A DSP (XC3SD1800A) family member has four DSP48A columns, while the larger Spartan-3A DSP (XC3SD3400A) family member has five DSP48A columns.

The multipliers and block RAM share interconnect resources in the Virtex-II and Virtex-II Pro architectures. Spartan-3A DSP devices have independent routing for the DSP48A slices and block RAM, effectively doubling the available data bandwidth between the elements.

Figure 1-4 shows the DSP48A interconnect and relative dedicated element sizes.

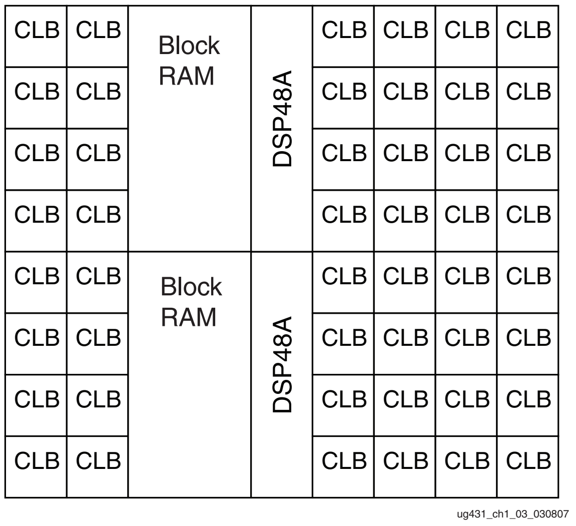


Figure 1-4: **DSP48A Layout**

Figure 1-5 shows a DSP48A slice and its associated data paths stacked vertically in a DSP48A column. The inputs to the shaded multiplexers are selected by configuration control signals. These attributes are set in the HDL source code or by the User Constraint File (UCF).

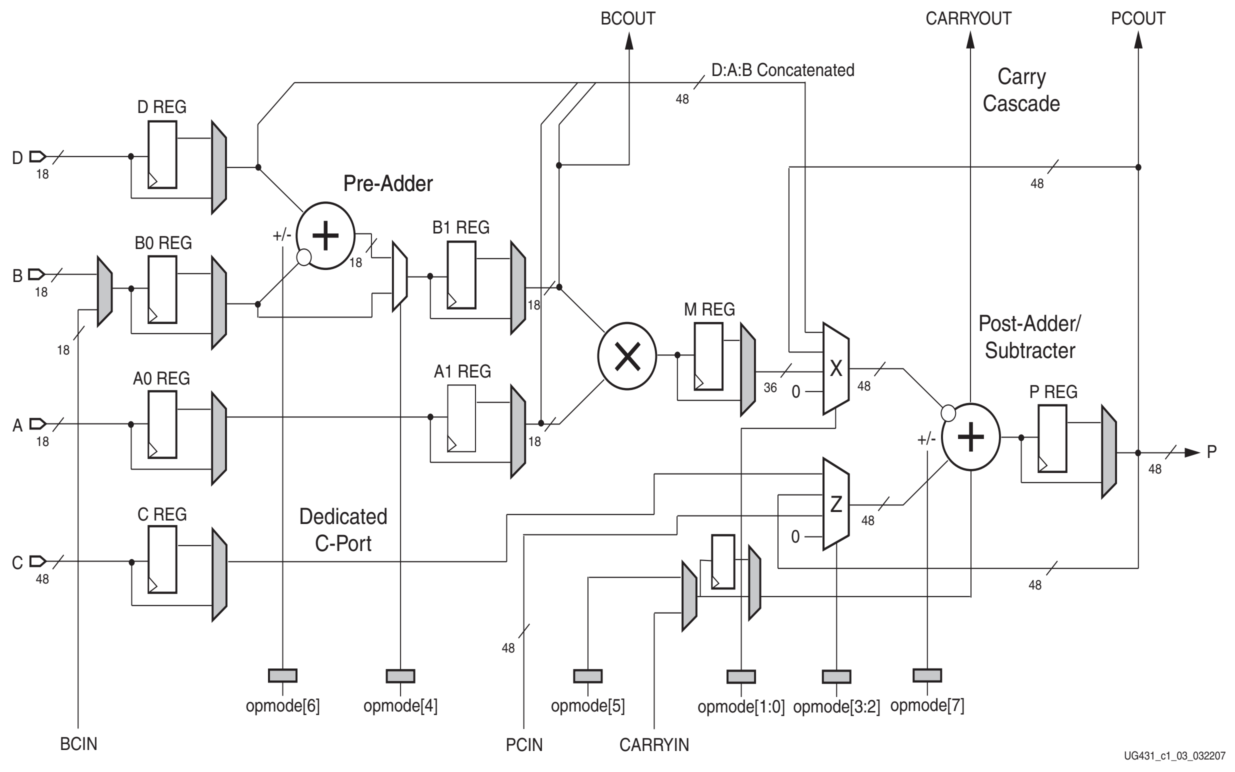


Figure 1-5: **A DSP48A Slice**

Notes:

1. The 18-bit A, B, and D bus are concatenated in the following order: D[11:0], A[17:0], B[17:0].
2. The X and Z multiplexers are 48-bit designs. Selecting any of the 36-bit inputs provides a 48-bit sign-extended output.
3. The multiplier outputs a 36-bit result.
4. The multiply-accumulate path for P is through the Z multiplexer. The P feedback through the X multiplexer enables accumulation of P cascade when the multiplier is not used.
5. The gray-colored multiplexers are programmed at configuration time.
6. The C register supports multiply-add, or wide addition. The C register is dedicated to the DSP48A Block. This differs from the C register in the Virtex DSP48 block, where a single C register is shared between two DSP48 blocks.
7. Enabling SUBTRACT implements Z – (X + CARRYIN) at the output of the post-adder/ subtracter.
8. B input can be added or subtracted from the D input using the pre-adder/subtracter.
9. Inputs (B & D) and output of pre-adder are 18-bit wide. Since two 18-bit inputs can produce an overflow, the user must take this situation into consideration.

Simplified DSP48A Slice Operation

The math portion of the DSP48A slice consists of an 18 bit pre-adder followed by an 18-bit x 18-bit, two's complement multiplier followed by two 48-bit data path multiplexers (with outputs X and Z) followed by a two-input, 48-bit post-adder/subtracter.

The data and control inputs to the DSP48A slice feed the arithmetic portions directly or are optionally registered one or two times to assist the construction of different, highly pipelined, DSP application solutions. The data inputs A and B can be registered once or twice. The other data inputs and the control inputs can be registered once. Full-speed operation is 250 MHz when using the pipeline registers.

Logic Equation

In its most basic form, the output of the post-adder/subtracter is a function of its inputs. The inputs are driven by the upstream multiplexers, carry select logic, and multiplier array. Functional Equation 1-1 summarizes the combination of X, Z, and CARRYIN by the post-adder/subtracter. The CARRYIN and X multiplexer output are always added together. This combined result can be selectively added to or subtracted from the Z multiplexer output.

$$\mathit{Adder\ Out} = (Z \pm (X + \mathit{CARRYIN}))$$ Equation 1-1

Functional Equation 1-2 describes a typical use where A and B are multiplied, and the result is added to or subtracted from the C register. More detailed operations based on control and data inputs are described in later sections. Selecting the multiplier function consumes the X multiplexer output to feed the adder. The 36-bit product from the multiplier is sign extended to 48 bits before being sent to the post-adder/subtracter.

$$\mathit{Adder\ Out} = C \pm (A \text{ ¥ } B + \mathit{CARRYIN})$$ Equation 1-2

Functional Equation 1-3 describes a use where B and D are added initially through the pre-adder/subtracter, and the result of the pre-adder is then multiplied against A, with the result of the multiplication being added to the C input. This equation facilitates efficient use for symmetric filters.

$$\mathit{Adder\ Out} = C \pm (A \text{ ¥ } (D \pm B) + \mathit{CARRYIN})$$ Equation 1-3

Figure 1-6 shows the DSP48A slice in a very simplified form. The eight OPMODE bits control the selection of the 48-bit data paths of the multiplexers feeding each of the two inputs to the post-adder/subtracter, and the B input to the multiplier. OPMODE bits also control add or subtract on the pre-adder/subtracter, or the post-adder/subtracter. In all cases, the 36-bit input data to the multiplexers is sign extended, forming 48-bit input data paths to the post-adder/subtracter. Based on 36-bit operands and a 48-bit accumulator output, the number of *guard bits* (i.e., bits available to guard against overflow) is 12. Therefore, the number of multiply accumulations possible before overflow occurs is 4096. Combinations of OPMODE, CARRYINSEL, and CARRYIN control the function of the pre- and post-adder/subtracters. See Table 1-5 for more details regarding the op-code combinations.

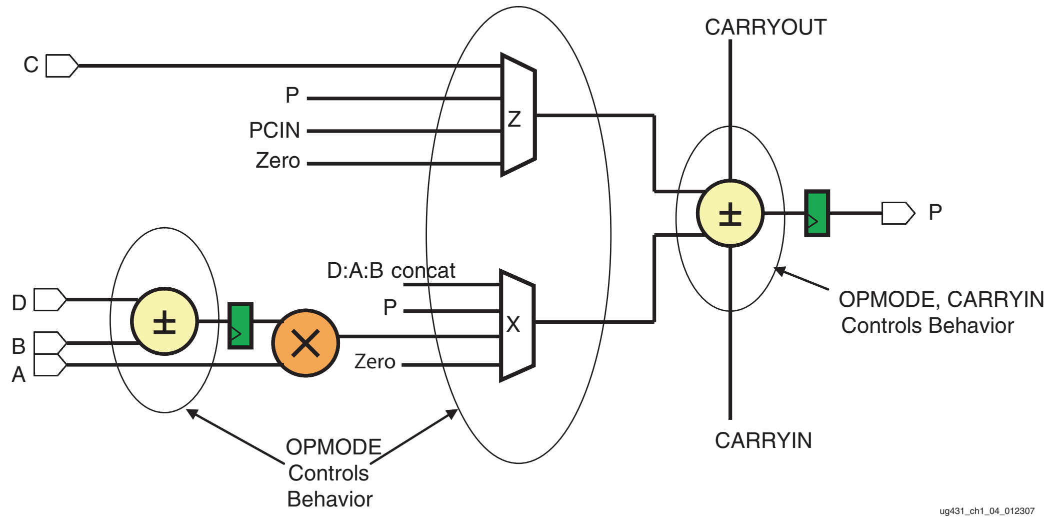


Figure 1-6: **Simplified DSP48A Slice Model**

A, B, C, D, and P Port Logic

The DSP48A input and output data ports support many common DSP and math algorithms. The DSP48A slice has three direct 18-bit input data ports labeled A, B, and D and a 48-bit input data port labeled C. Each DSP48A slice has one direct 48-bit output port labeled P. Each DSP48A also has a cascaded input data path (B cascade) and a cascaded output data path (P cascade), providing a cascaded input and output stream between adjacent DSP48A slices. The B cascade is a dedicated resource always connected to the adjacent DSP48A selected via the B_INPUT attribute. Also included in the DSP48A slice are a CARRYIN input, and a CARRYOUT output. These resources are used to facilitate the creation of parallel MACC operations in adjacent DSP48A slices. The PCIN cascade is a dedicated resource that is always connected to the adjacent DSP48A and can be dynamically selected via the Z MUX (OPMODE 3:2).

Applications benefiting from these features include FIR filters, complex multiplication, multi-precision multiplication, complex MACs, adder cascade, and adder trees (the final summation of several multiplier outputs) support.

The 18-bit D and B input can supply input data to the 18-bit pre-adder/subtracter. The pre-adder/subtracter can be bypassed if the user desires. The output of the pre-adder can feed one input of the multiplier, if desired, using OPMODE[4].

The 18-bit A and B port can supply input data to the 18-bit x 18-bit, two's complement multiplier. When concatenated, D, A and B can bypass the multiplier and feed the X multiplexer input directly. The 48-bit C port is used as a general input to the Z multiplexer to perform add or subtract.

Multiplexers are controlled by configuration bits to select flow-through paths, optional registers, or cascaded inputs. The data port registers allow users to trade off increased clock frequency (i.e., higher performance) vs. data latency. Also, a configuration controlled pipeline register between the multiplier and adder/subtracter is known as the M register. The registers have independent clock enables and resets, described in Table 1-2 and shown in Figure 1-3.

The A, B, C, and P port logics are shown in Figure 1-7, Figure 1-8, Figure 1-9, and Figure 1-10, respectively.

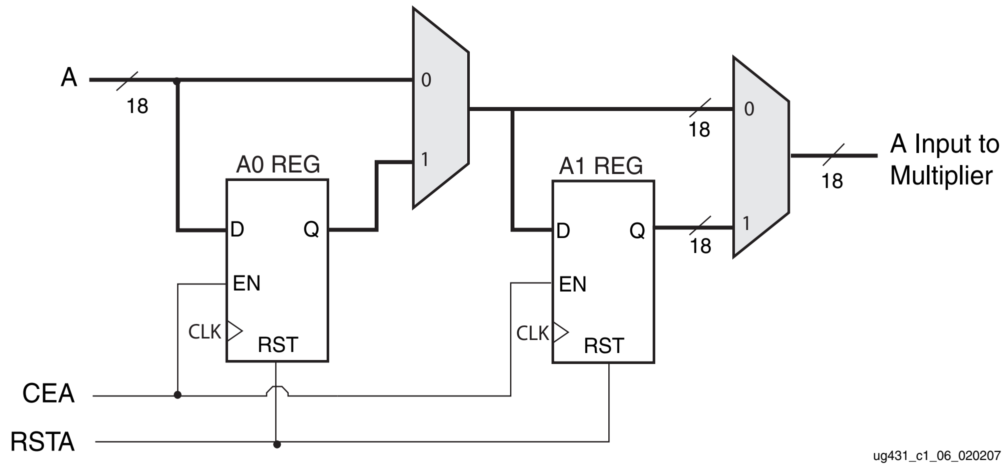


Figure 1-7: **A Input Logic**

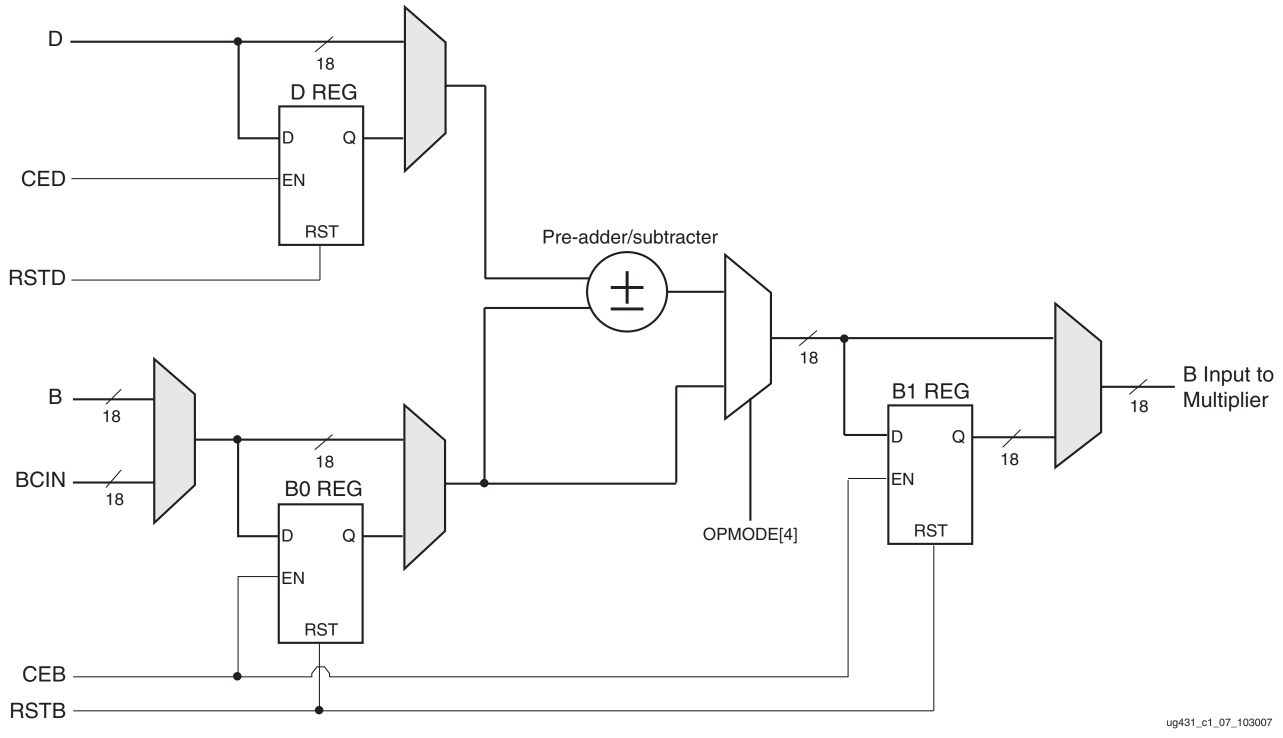


Figure 1-8: **B and D Input Logic**

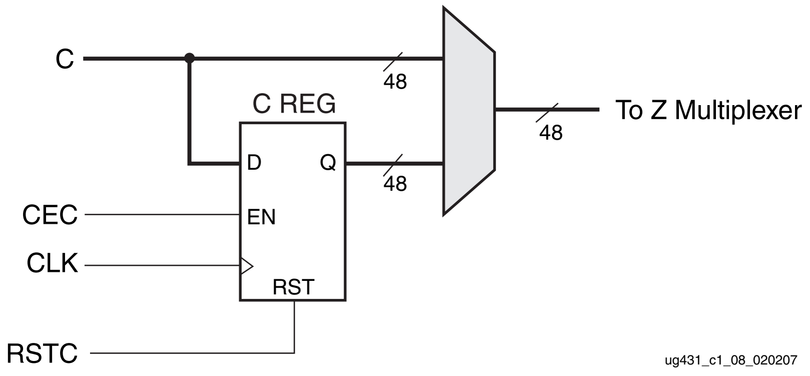


Figure 1-9: **C Input Logic**

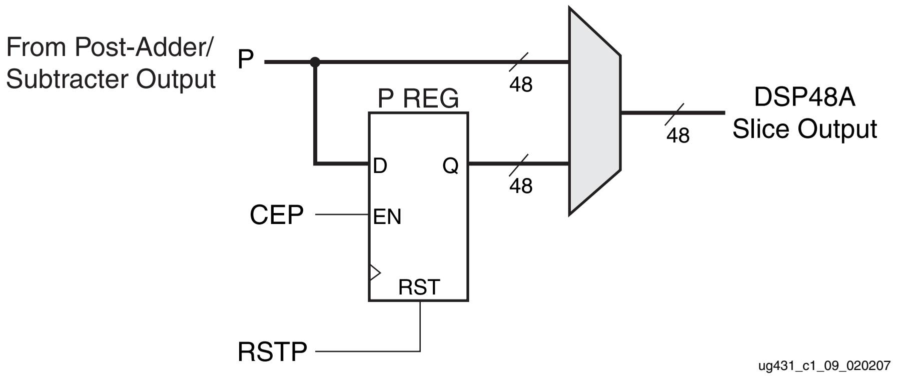


Figure 1-10: **P Output Logic**

Note: Grey multiplexers are multiplexers that are programmed at configuration time. Clear multiplexers are dynamic and, therefore, can be changed using the OPMODE bits.

OPMODE Port Logic

The OPMODE port logic supports flow through or registered input control signals. Similar to the data paths, multiplexers controlled by configuration bits select flow through or optional registers. The control port registers allow users to trade off increased clock frequency (for example, higher performance) versus data latency.

The registers have independent clock enables and resets, described in Table 1-2 and shown in Figure 1-3. The OPMODE registers are reset by RSTO. The OPMODE register has a separate enable labeled CEO. Figure 1-11 shows the OPMODE port logic.

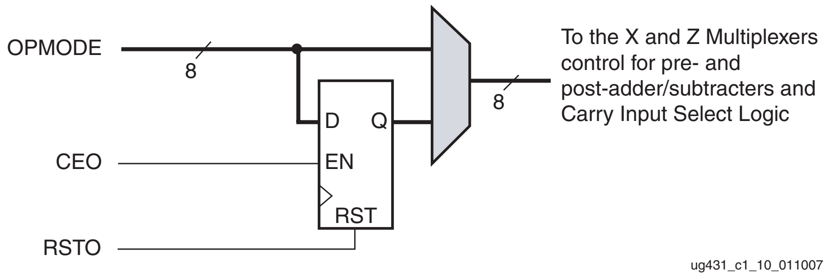


Figure 1-11: **OPMODE Port Logic**

Two's Complement Multiplier

The two's-complement multiplier inside the DSP48A slice accepts two 18-bit, two's complement inputs and produces a 36-bit, two's complement result. Cascading of multipliers to achieve larger products is supported with a 17-bit right-shift implemented in the fabric. This bus is then input to the post-adder/subtracter to right justify partial products by the correct number of bits. MACC functions can also right justify intermediate results for multi-precision. The multiplier can emulate unsigned math by setting the MSB of an 18-bit operand to zero.

The output of the multiplier consists of a 36-bit product. The product is sign extended to 48 bits prior to being input to the adder/subtracter. Selecting the output of the multiplier consumes the X multiplexer.

Figure 1-12 shows an optional pipeline register (M REG) for the output of the multiplier. Using the register provides increased performance with a single clock cycle of increased latency. The gray multiplexer indicates "selected at configuration time by configuration bits."

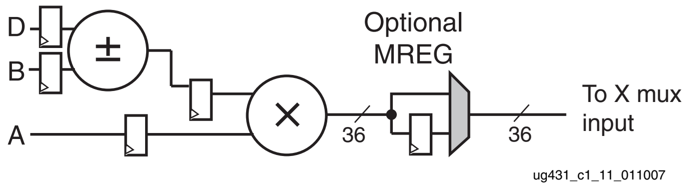


Figure 1-12: **Two's Complement Multiplier Followed by Optional M REG**

X and Z Multiplexer

The OPMODE inputs provide a way for the design to change its functionality from clock cycle to clock cycle (for example, when altering the initial or final state of the DSP48A relative to the middle part of a given calculation). The OPMODE bits can be optionally registered under the control of the configuration memory cells (as denoted by the gray MUX symbol in Figure 1-11).

Table 1-4 and Table 1-5 list the possible values of OPMODE and the resulting function at the outputs of the two multiplexers (X and Z multiplexers). The multiplexer outputs supply three operands to the following adder/subtracter. If the multiplier output is selected, then the X multiplexer is used, supplying the multiplier output to the post-adder/subtracter.

Table 1-6 identifies OPMODE[7:4] and their associated functions.

Table 1-4: **OPMODE Control Bits Selecting X Multiplexer Outputs**

OPMODE Binary		X Multiplexer Output Fed to Post-Adder/Subtracter
Z OPMODE[3:2]	X OPMODE[1:0]	
XX	00	ZERO (Default)
XX	01	Multiplier Output
XX	10	P
XX	11	D[11:0] concatenated with A[17:0] concatenated with B[17:0]

Table 1-5: **OPMODE Control Bits Selecting Z Multiplexer Outputs**

OPMODE Binary		Z Multiplexer Output Fed to Post-Adder/Subtracter
Z OPMODE[3:2]	X OPMODE[1:0]	
00	XX	ZERO (Default)
01	XX	PCIN
10	XX	P
11	XX	C

Table 1-6: **Functional Description of OPMODE[7:4]**

OPMODE	Functional Description
OPMODE[7]	Post-Adder/Subtracter select 1 = Subtract
OPMODE[6]	Pre-Adder/Subtracter select 1 = Subtract
OPMODE[5]	User CARRYIN bit
OPMODE[4]	Use Pre-Adder/Subtracter with B input 1 = Use the Pre-Adder/Subtracter

The possible non-zero operands for the post-adder/subtracter as selected by the two multiplexers are listed below:

- Multiplier output (36-bit)
- Multiplier bypass bus consisting of A concatenated with B and D
- C bus, 48 bits
- Cascaded P bus, 48 bits, from a neighbor DSP48A slice
- Registered P bus output, 48 bits, for accumulator functions
- Registered P bus output for accumulator functions

Note: All 36-bit operands are sign extended to 48 bits.

Table 1-7 shows a complete description of all the DSP48A operating modes.

Table 1-7: **DSP48A Operating Modes**

OPMODE	OPMODE[7:0]								CARRYINSEL	Output
	Post-Adder/ Subtracter	Pre_Adder/ Subtracter	CARRYIN	Pre-Adder/ Subtracter Select	Z		X			
	7	6	5	4	3	2	1	0		
Zero + Carryin	0	0/1	0/1	0/1	0	0	0	0	0	+ Carryin
Zero – Carryin	1	0/1	0/1	0/1	0	0	0	0	0	– Carryin
Zero + OpMode<5>	0	0/1	0/1	0/1	0	0	0	0	1	+ OpMode[5]
Zero – OpMode<5>	1	0/1	0/1	0/1	0	0	0	0	1	– OpMode[5]
Hold P	0/1	0/1	0/1	0/1	0	0	1	0	0/1	±(P + CARRYIN)
D:A:B Select	0/1	0/1	0/1	0	0	0	1	1	0/1	± (D:A:B + CARRYIN)
D:A:B Select: with PreAdd	0/1	0/1	0/1	1	0	0	1	1	0/1	± (D:A:(D ±B) + CARRYIN)
Multiply	0/1	0/1	0/1	0	0	0	0	1	0/1	±(A × B + CARRYIN)
PreAdd-Multiply	0/1	0	0/1	1	0	0	0	1	0/1	±(A × (D + B) + CARRYIN)
PreSubtract-Multiply	0/1	1	0/1	1	0	0	0	1	0/1	± (A × (D – B) + CARRYIN)
P Cascade Select	0/1	0/1	0/1	0/1	0	1	0	0	0/1	PCIN ±CIN
P Cascade Feedback Add	0/1	0/1	0/1	0/1	0	1	1	0	0/1	PCIN ±(P + CARRYIN)
P Cascade Add	0/1	0/1	0/1	0	0	1	1	1	0/1	PCIN ±(D:A:B + CARRYIN)
P Cascade Add with PreAdd	0/1	0/1	0/1	1	0	1	1	1	0/1	PCIN ± (D:A:(D ± B) + CARRYIN
P Cascade Multiply Add	0/1	0/1	0/1	0	0	1	0	1	0/1	PCIN ±(A × B + CARRYIN)
Cascade PreAdd-Multiply	0/1	0	0/1	1	0	1	0	1	0/1	PCIN ± (A × (D + B) + CARRYIN)
Cascade PreSubtract-Multiply	0/1	1	0/1	1	0	1	0	1	0/1	PCIN ± (A × (D – B) + CARRYIN)
Feedback Carryin Add	0/1	0/1	0/1	0/1	1	0	0	0	0/1	P ± CARRYIN
Double Feedback Add	0/1	0/1	0/1	0/1	1	0	1	0	0/1	P ± (P + CARRYIN)
Feedback Add	0/1	0/1	0/1	0	1	0	1	1	0/1	P ± (D:A:B + CARRYIN)
Feedback Addwith PreAdd	0/1	0/1	0/1	1	1	0	1	1	0/1	P ± (D:A:(D ± B) + CARRYIN)
Multiply-Accumulate	0/1	0/1	0/1	0	1	0	0	1	0/1	P ± (A × B + CARRYIN)
Feedback PreAdd-Multiply	0/1	0	0/1	1	1	0	0	1	0/1	P ± (A × (D + B) + CARRYIN)
Feedback PreSubtract-Multiply	0/1	1	0/1	1	1	0	0	1	0/1	P ± (A × (D – B) + CARRYIN)
C Select	0/1	0/1	0/1	0/1	1	1	0	0	0/1	C ± CARRYIN
Feedback Add	0/1	0/1	0/1	0/1	1	1	1	0	0/1	C ± (P + CARRYIN)
48-Bit Adder	0/1	0/1	0/1	0	1	1	1	1	0/1	C ± (D:A:B + CARRYIN)
Multiply-Add	0/1	0/1	0/1	0	1	1	0	1	0/1	C ± (A × B + CARRYIN)
C PreAdd-Multiply	0/1	0	0/1	1	1	1	0	1	0/1	C ± (A × (D + B) + CARRYIN)
C PreSubtract-Multiply	0/1	1	0/1	1	1	1	0	1	0/1	C ± (A × (D – B) + CARRYIN)

Table 1-7: **DSP48A Operating Modes *(Continued)***

OPMODE	OPMODE[7:0]								CARRYINSEL	Output
	Post-Adder/ Subtracter	Pre_Adder/ Subtracter	CARRYIN	Pre-Adder/ Subtracter Select	Z		X			
	7	6	5	4	3	2	1	0		
48-Bit Adder with PreAdd	0/1	0/1	0/1	1	1	1	1	1	0/1	C ± (D:A:(D ± B) + CARRYIN)

Notes:
1. For the remaining functions, CARRYIN = CARRYIN output from the upstream DSP48A block or OPMODE[5] depending upon programming of CARRY SEL shown in this section.
2. Output mux must be selecting register output.
3. Added for all 1s case.

Pre-Adder/Subtracter

The pre-adder/subtracter output is a function of OPMODE[6] and the B and D data inputs. The pre-adder reduces resource utilization for symmetric filters. OPMODE[6] control signal (OPMODE[6] = 1 is defined as "subtraction"). It can also be combined with the post-adder for three-input adder functions.

Post-Adder/Subtracter/Accumulator

The post-adder/subtracter/accumulator output is a function of OPMODE and data inputs. OPMODE[3:0], as shown in the previous section, selects the inputs to the X or Z multiplexer directed to the associated post-adder/subtracter inputs.

Table 1-4 and Table 1-5 shows OPMODE combinations and the resulting functions. The symbol ± in the table means either add or subtract and is specified by the state of OPMODE[7] control signal (OPMODE[7] = 1 is defined as "subtraction"). The outputs of the X multiplexer and CARRYIN (or OPMODE[5] depending on the programming of the Carry Select bit) are always added together. This result is then added to or subtracted from the output of the Z multiplexer.

Carry Input Logic

The carry input logic result is a function of the CARRYIN Select configuration bit, OPMODE[5], and CARRYIN. The inputs to the carry input logic appear in Figure 1-13. Carry inputs used to form results for adders and subtracters are always in the critical path. High performance is achieved by implementing this logic in the diffused silicon. The possible carry inputs to the carry logic are "gathered" prior to the outputs of the X and Z multiplexers. In a sense, the X and Z multiplexer function is duplicated for the carry inputs to the carry logic. Both OPMODE and CARRYINSEL must be in the correct state to ensure the correct carry input (CARRYIN) is selected.

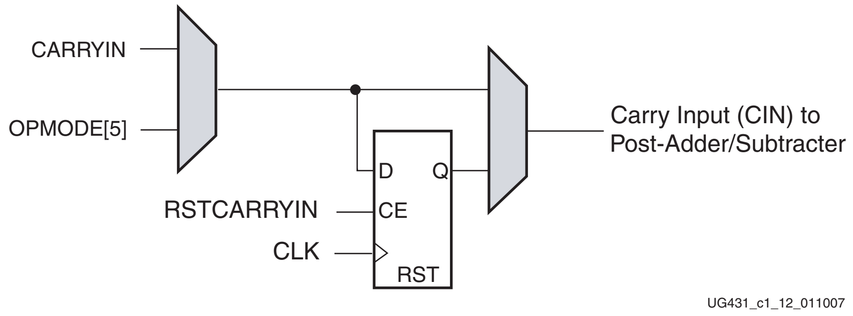


Figure 1-13: **Carry Input Logic Feeding the Post-Adder/Subtracter**

Figure 1-13 shows two inputs, selected by the CARRYINSEL programming bit. The first input CARRYIN (CARRYINSEL is equal to `0`) is driven from the carry-out of the upstream DSP48A block. The second input OPMODE[5] can be driven from general logic. This option allows implementation of a carry function based on user logic, or from implementations residing entirely inside DSP48A blocks. It can be optionally registered to match the pipeline delay of the MREG when used. This register delay is controlled by the CARRYINREG configuration bit.

Forming Larger Multipliers

Figure 1-14 illustrates the formation of a 35 x 35-bit multiplication from smaller 18 x 18-bit multipliers. The notation "0,B[16:0]" denotes B has a leading zero followed by 17 bits, forming a positive two's complement number.

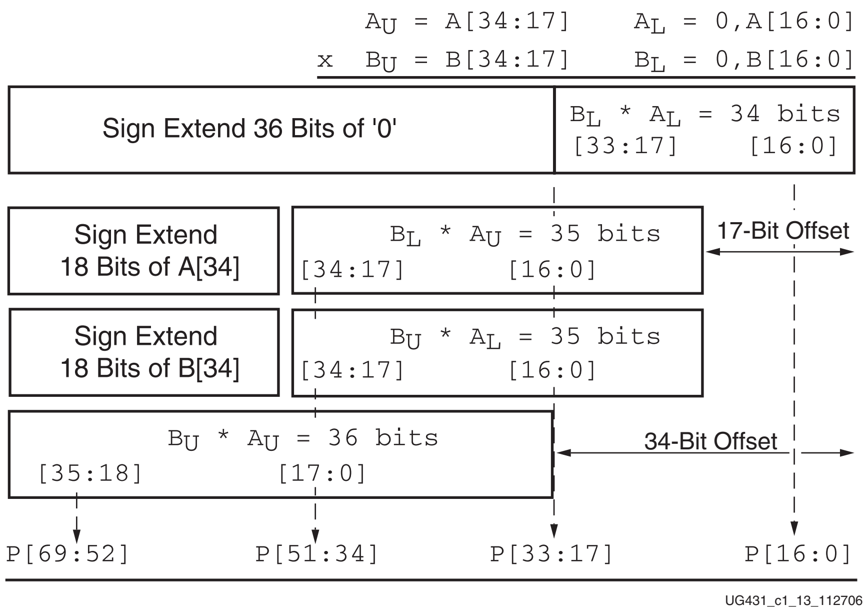


Figure 1-14: **35x35-Bit Multiplication from 18x18-bit Multipliers**

When separating two's complement numbers into two parts, only the most-significant part carries the original sign. The least-significant part must have a *forced zero* in the sign position meaning they are positive operands. While it seems logical to separate a positive number into the sum of two positive numbers, it can be counter intuitive to separate a negative number into a negative most-significant part and a positive least-significant part.

However, after separation, the most-significant part becomes "more negative" by the amount the least-significant part becomes "more positive." The 36-bit input operands include a forced zero sign bit in the least-significant part. So the valid number of bits in the input operands is only 35-bits.

The DSP48A slices with 18 x 18 multipliers and post-adder/subtracter can now be used to implement the sum of the four partial products shown in Figure 1-14. The lessor significant partial products must be right-shifted by 17 bit positions before being summed with the next most-significant partial products. This is accomplished with a *wire shift* implemented in user logic, applied to the C port. The entire process of multiplication, shifting, and addition using adder cascade to form the 70-bit result can remain in the dedicated silicon of the DSP48A slice, resulting in maximum performance with minimal power consumption.

FIR Filters

Basic FIR Filters

FIR filters are used extensively in video broadcasting and wireless communications. DSP filter applications include, but are not limited to, the following:

- Wireless Communications
- Image Processing
- Video Filtering
- Multimedia Applications
- Portable Electrocardiogram (ECG) Displays
- Global Positioning Systems (GPS)

Equation 1-4 shows the basic equation for a single-channel FIR filter.

$$y(n) = \sum_{k=0}^{k=N-1} h(k)x(n-k)$$

Equation 1-4

The terms in the equation can be described as input samples, output samples, and coefficients. Imagine x as a continuous stream of input samples and y as a resulting stream (i.e., a filtered stream) of output samples. The n and k in the equation correspond to a particular instant in time, so to compute the output sample y(n) at time n, a group of input samples at N different points in time, or x(n), x(n-1), x(n-2), … x(n-N+1) is required. The group of N input samples are multiplied by N coefficients and summed together to form the final result y.

The main components used to implement a digital filter algorithm include adders, multipliers, storage, and delay elements. The DSP48A slice includes all of the above elements, making it ideal to implement digital filter functions. All of the input samples from the set of n samples are present at the input of each DSP48A slice. Each slice multiplies the samples with the corresponding coefficients within the DSP48A slice. The outputs of the multipliers are combined in the cascaded adders.

In Figure 1-15, the sample delay logic is denoted by Z^{-1}, where the –1 represents a single clock delay. The delayed input samples are supplied to one input of the multiplier. The coefficients (denoted by *h0* to *h(N-1)*) are supplied to the other input of the multiplier through individual ROMs, RAMs, registers, or constants. Y(n) is merely the summation of a set of input samples, and in time, multiplied by their respective coefficients.

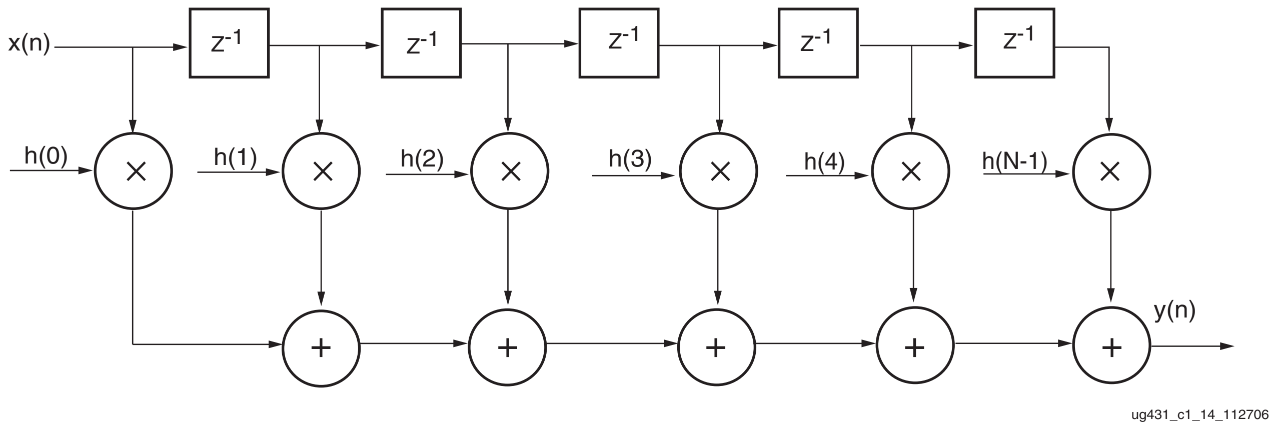


Figure 1-15: **Conventional Tapped Delay Line FIR Filter**

Multichannel FIR Filters

Multichannel filters are used to filter multiple data streams of input signals. The channels can either use the same set of coefficients or different coefficients.

A common example of a multichannel filter is a radio receiver digital down converter. Equation 1-5 shows the equation, and Figure 1-16 shows the block diagram. A digitized baseband signal is applied to a matched low-pass filter M(z) to reduce the data rate from the input sample rate to the bit rate. The resulting in-phase and quadrature components are each processed by the same filter and, therefore, could be processed by a single, multichannel filter running at twice the sample rate.

$$x(n) = x_I(n) + jx_Q(n) \qquad \textit{Equation 1-5}$$

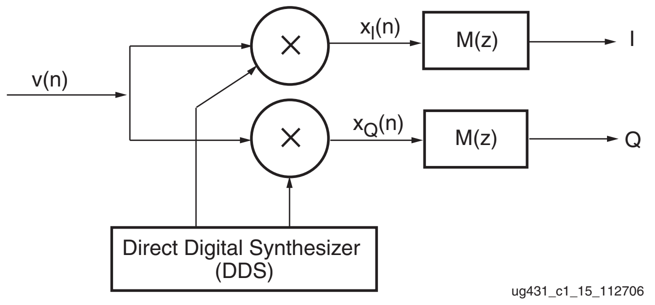


Figure 1-16: **Software-Defined Radio Digital Down Converter**

Some video applications use multi-channel implementations for multiple components of a video stream. Typical video components are red, green, and blue (RGB) or luma, chroma red, and chroma blue (YCrCb). The different video components can have the same coefficient sets or different coefficient sets for each channel by simply changing the coefficient ROM structure.

Creating FIR Filters

Referring to Figure 1-6, Table 1-4, and Table 1-5, an inner product MACC operation starts by loading the first operand into the P register. The output of the multiplier is passed through the X multiplexer, added to zero, and loaded into the P register. Note the load operation OPMODE with value `00000001` selects zero to be output on the Z multiplexer supplying one of the post-adder/subtracter inputs. A previous MACC inner product can exit via the P bus during this clock cycle.

In subsequent clock cycles, the MACC operation requires the X multiplexer to supply the multiplier output and the Z multiplexer to supply the output of the P register to the post-adder/subtracter. The OPMODE for this operation is 00001001 .

To create a simple multiply-add processing element using the DSP48A slice shown in Figure 1-6, the X multiplexer is set to multiply and the cascaded input from another DSP48A output (PCIN) is selected as the Z MUX input to the arithmetic unit. For a normal multiply-add operation, the OPMODE value is set to `00000101`.

Refer to the *XtremeDSP for Virtex-4 FPGAs User Guide* (UG073) for additional examples of MACC FIR structures and Parallel FIR structures.

Adder Cascade Versus Adder Tree

In typical direct form FIR filters, an input stream of samples is presented to one data input of separate multipliers where coefficients supply the other input to the multipliers. An adder tree follows the multipliers where the outputs from many multipliers are combined (see Figure 1-17).

One difficulty of the adder tree concept is defining the size. Filters come in various lengths and consume a variable number of adders, forming an adder tree. Placing a fixed number of adder tree components in silicon displaces other elements or requires a larger FPGA, thereby increasing the cost of the design. In addition, the adder tree structure with a fixed number of additions forces the designer to use logic resources when the fixed number of additions is exceeded. Using logic resources dramatically reduces performance and increases power consumption. The key to maximizing performance and lowering power for DSP math is to remain inside the DSP48A column consisting entirely of dedicated silicon.

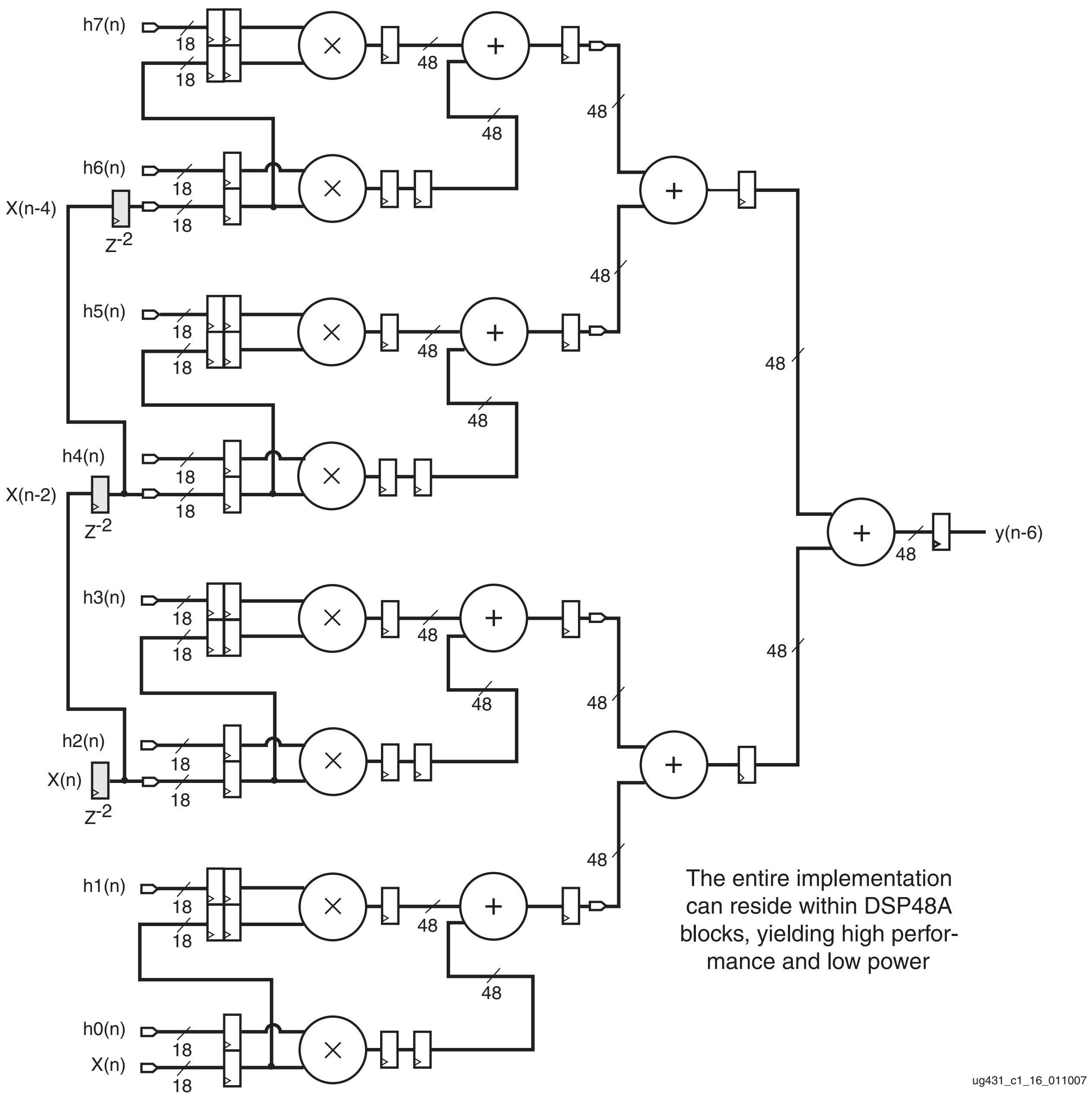

h7(n)
h6(n)
X(n-4)
Z-2
h5(n)
h4(n)
X(n-2)
Z-2
h3(n)
h2(n)
X(n)
Z-2
h1(n)
h0(n)
X(n)
18
48
y(n-6)
The entire implementation can reside within DSP48A blocks, yielding high performance and low power
ug431_c1_16_011007

Figure 1-17: **FIR Filter Adder Tree Using DSP48A Slices**

The Spartan-3A DSP solution accomplishes the post-addition process while guaranteeing no wasted silicon resources. The solution involves computing the additive result incrementally utilizing a cascaded approach, illustrated in Figure 1-18. The cascaded approach is a systolic version of a direct form FIR with a latency of 10 clocks, versus an adder tree latency of 6 clocks.

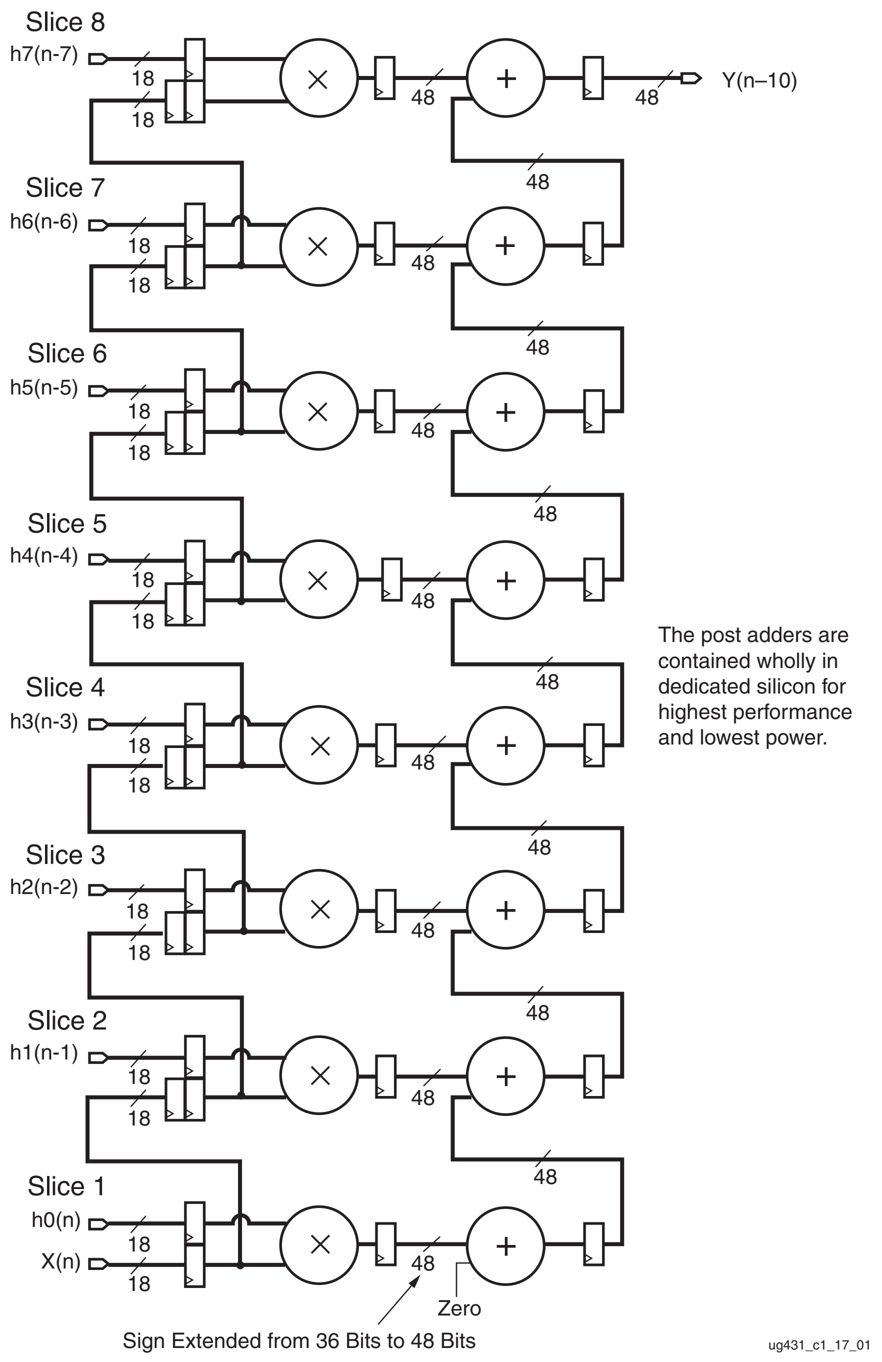


Figure 1-18: **Systolic FIR with Adder Cascade**

Note: To ensure correct operation, the input sample delay and the coefficients must be balanced with the cascaded adder. The adaptive coefficients are staggered in time (wave coefficients).

DSP48A Slice Functional Use Models

Fully Pipelined, 35 x 35 Multiplier Use Model

Similar to the 35 x 18-bit example, this fully pipelined design can present inputs every clock cycle. An output is also computed every single clock cycle. Once again, no particular path becomes the timing bottleneck. The single slice version of the 35 x 35-bit multiply uses four clock cycles. In each clock cycle, the slice is presented with different operands, and switching the OPMODE bits modifies the behavior. The fully pipelined version connects separate slices with fixed behavior. See Figure 1-19.

Note: If only one input register is being used to obtain the highest internal multiplier frequency, use the A1 and B1 registers instead of the A0 and B0.

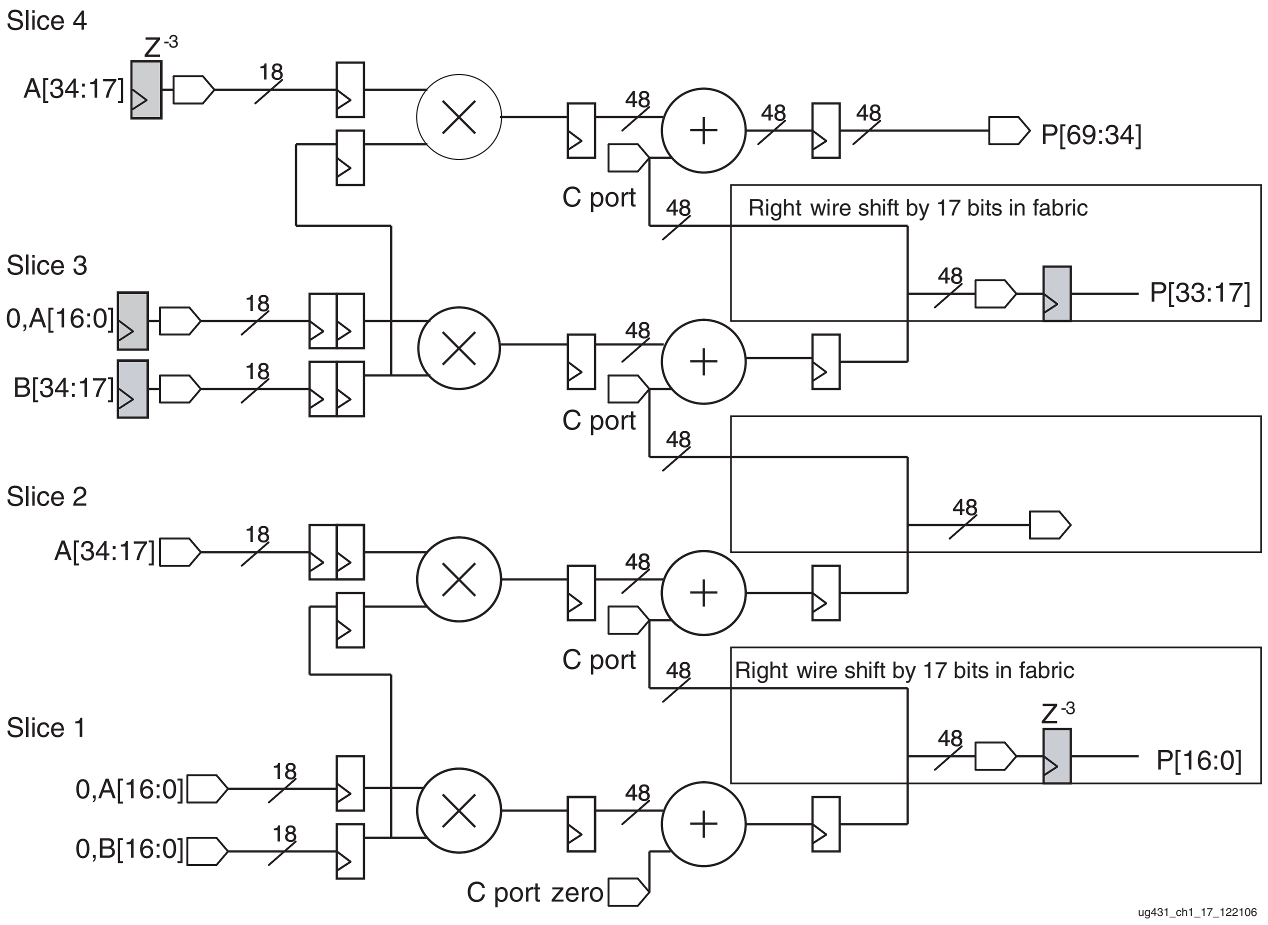


Figure 1-19: **Pipelined Version of 35x35 Multiplier Built from Cascaded DSP48A Slices**

In the single slice versions of this algorithm, partial products are computed sequentially and summed in the adder. For the fully pipelined version of the algorithm, the same partial products are computed in parallel and summed in the last slice, producing a result and consuming new input operands every clock cycle.

As in the 35 x 18-bit example, there are additional register stages placed in the input paths to delay input data until the needed cascading results arrive. In Figure 1-19, the block diagram for the fully pipelined, 35 x 35 multiply shows where additional input register stages are placed. The 35 x 35-bit multiplier has additional output registers outside of the slice to align the output data. The notation Z^{-3} is in the external register to signify that the data must be delayed by three clock cycles. If the delay is only one cycle, then registers are typically used. When the delay is larger than one, an SRL16 followed by the associated CLB flip-flop achieves maximum design performance.

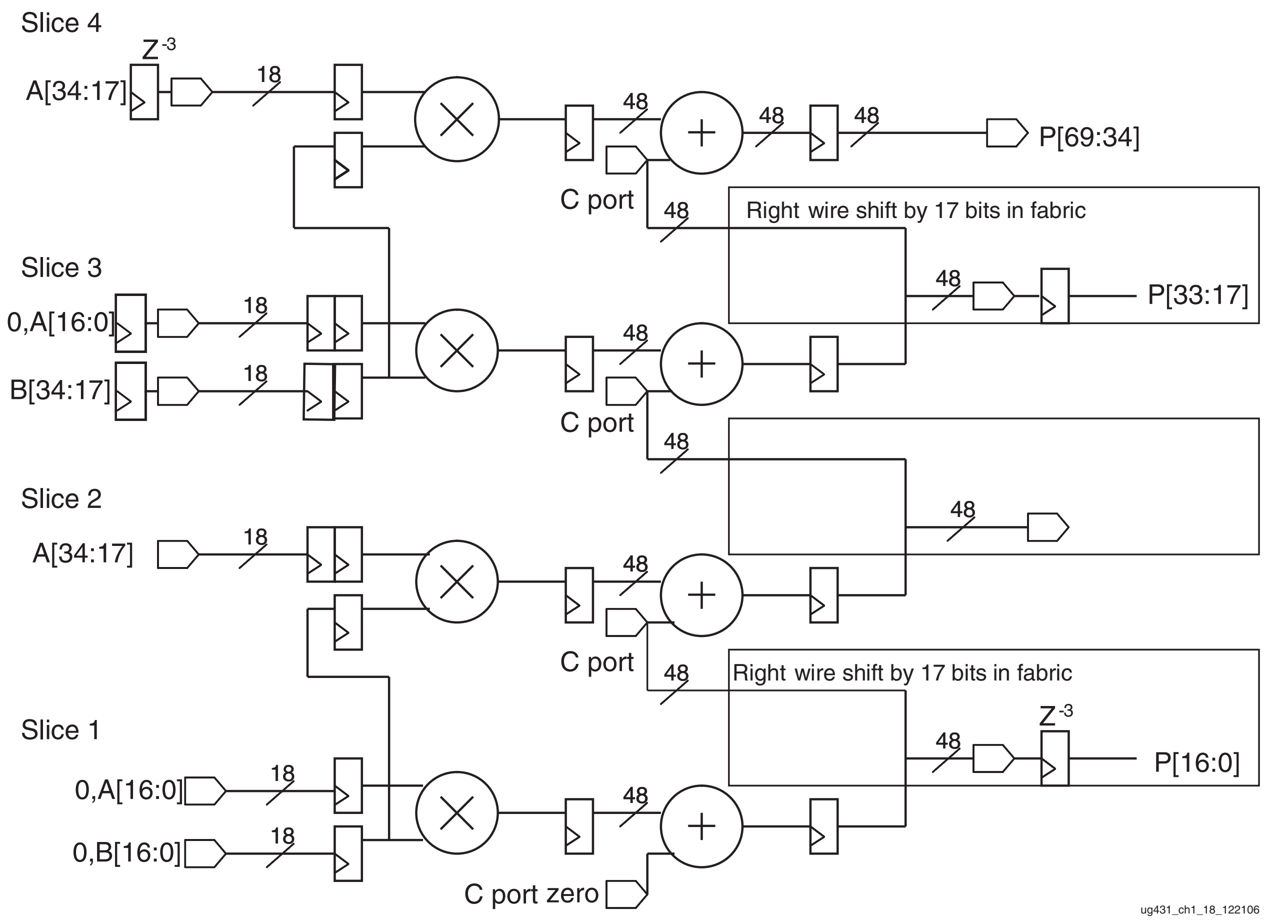


Figure 1-20: **Non-Pipelined Version of 35x35 Multiplier Built from Cascaded DSP48A Slices**

Fully Pipelined, Complex, 18 x 18 Multiplier Use Model

Complex multiplication used in many DSP applications combines operands having both real and imaginary parts into results with real and imaginary parts. Two operands A and B, each having real and imaginary parts, are combined as shown in the following equations:

$$(A\_real \times B\_real) - (A\_imaginary \times B\_imaginary) = P\_real \qquad \textit{Equation 1-6}$$

$$(A\_real \times B\_imaginary) + (A\_imaginary \times B\_real) = P\_imaginary \qquad \textit{Equation 1-7}$$

The real and imaginary results use the same slice configuration with the exception of the post-adder/subtracter. The post-adder/subtracter performs subtraction for the real result and addition for the imaginary result.

Figure 1-21 shows several DSP48A slices used as a complex, 18-bit x 18-bit multiplier.

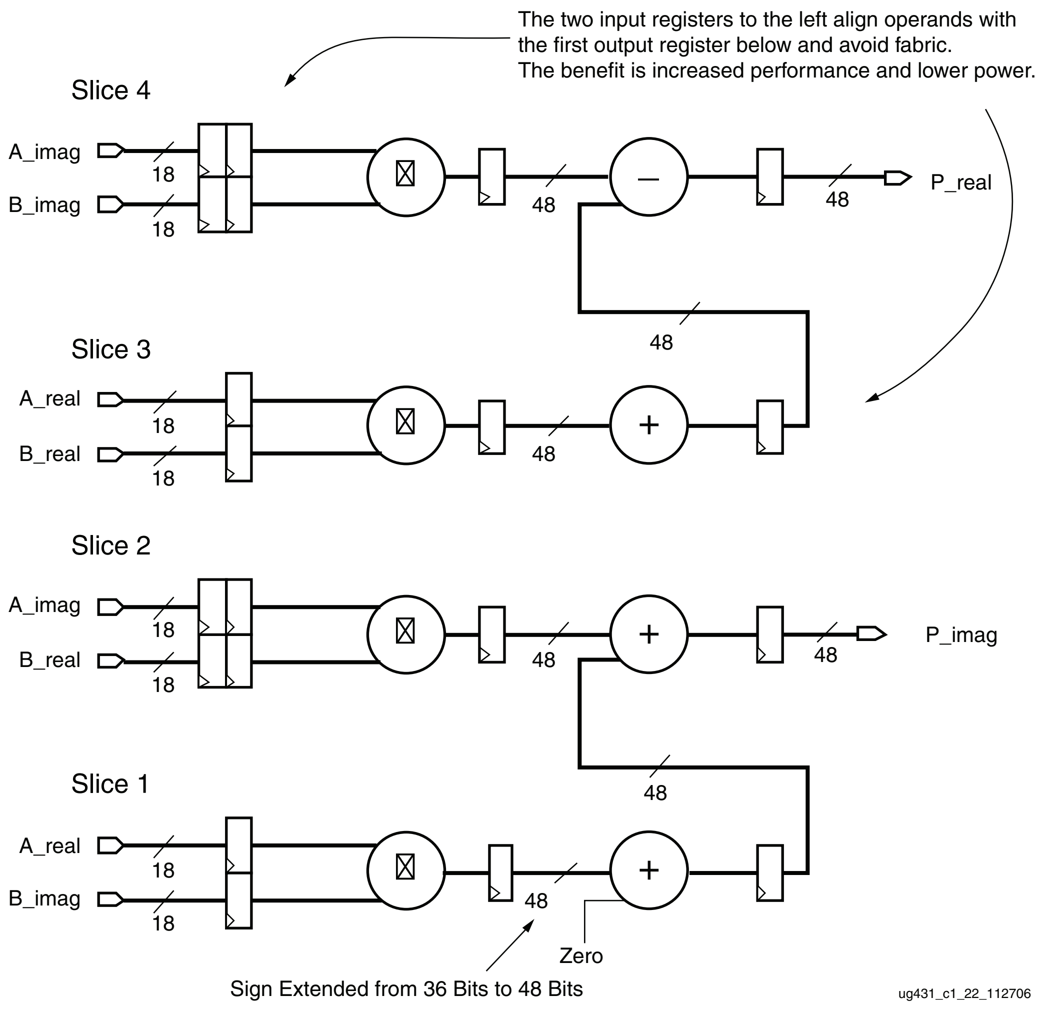

The two input registers to the left align operands with the first output register below and avoid fabric. The benefit is increased performance and lower power.
Slice 4
A_imag
B_imag
18
18
48
48
P_real
48
Slice 3
A_real
B_real
18
18
48
Slice 2
A_imag
B_real
18
18
48
48
P_imag
48
Slice 1
A_real
B_imag
18
18
48
Zero
Sign Extended from 36 Bits to 48 Bits
ug431_c1_22_112706

Figure 1-21: **Pipelined, Complex, 18 x 18 Multiply**

Note: The real and the imaginary computations are functionally similar using different input data. The real output subtracts the multiplied terms, and the imaginary output adds the multiplied terms.

Fully Pipelined, Complex, 18 x 18 MACC Use Model

The differences between complex multiply and complex MACC implementations using several DSP48A slices is illustrated in the next set of equations. As shown, the addition and subtraction of the terms only occur after the desired number of MACC operations.

For N Cycles:

Slice 1 = (A_real × B_imaginary) accumulation
Slice 2 = (A_imaginary × B_real) accumulation
Slice 3 = (A_real × B_real) accumulation
Slice 4 = (A_imaginary × B_imaginary) accumulation

Last Cycle:

Slice 1 + Slice 2 = P_imaginary
Slice 3 – Slice 4 = P_real

During the last cycle, the input data must stall while the final terms are added. To avoid having to stall the data, instead of using the complex multiply implementation shown in Figure 1-22 and Figure 1-23, use the complex multiply implementation shown in Figure 1-24.

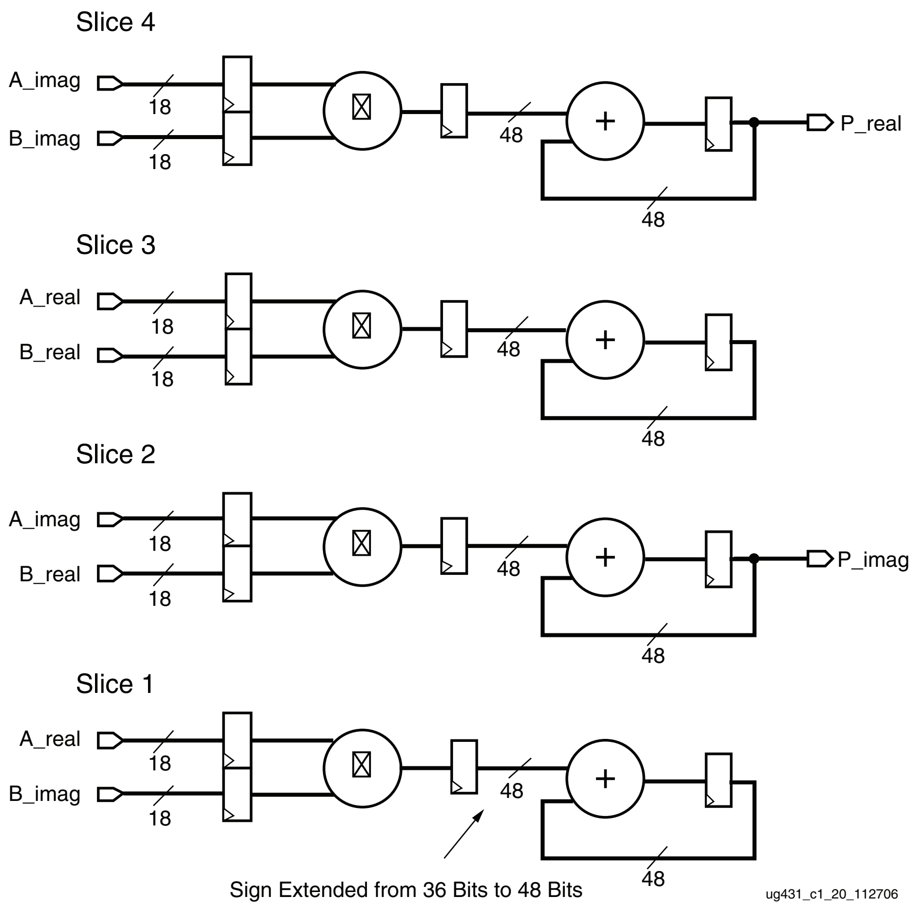


Figure 1-22: **Fully Pipelined, Complex, 18 x 18 MACC (N Cycles)**

In Figure 1-23, the N+1 cycle adds the accumulated products, and the input data stalls one cycle.

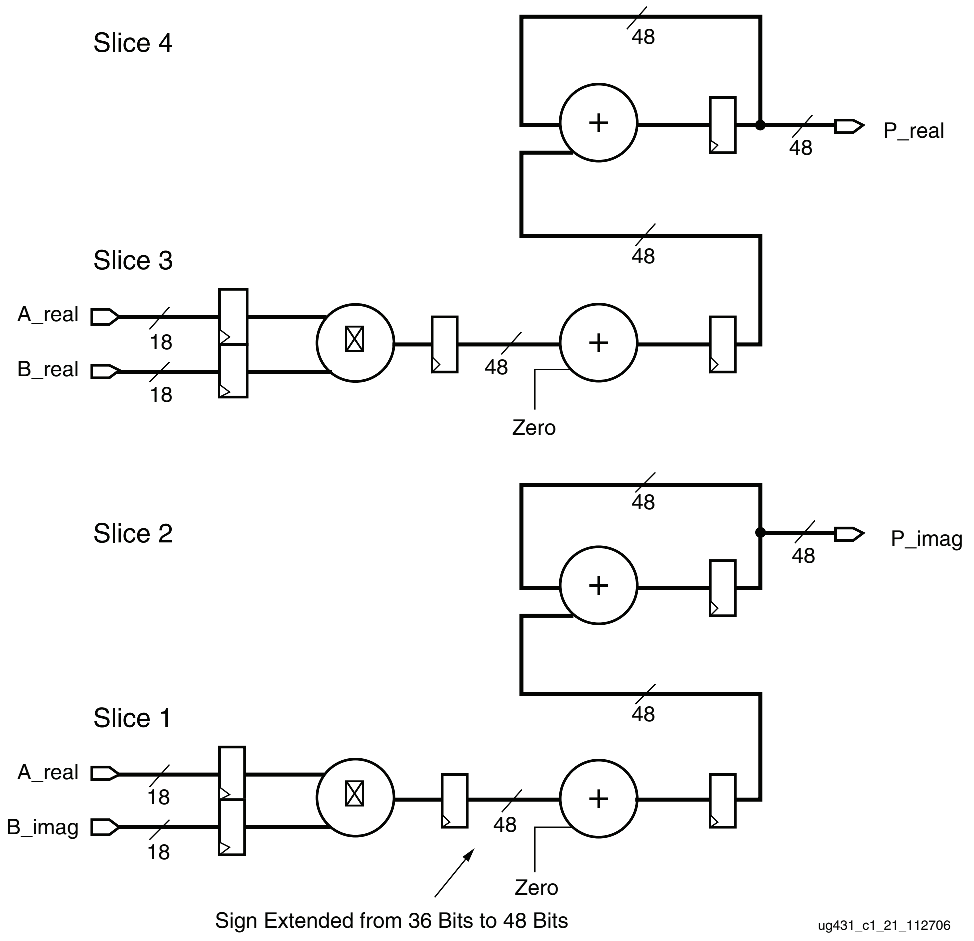


Figure 1-23: **Fully Pipelined, Complex, 18 x 18 MACC (Last or N+1 Cycle)**

An additional slice used for the accumulation is shown in Figure 1-24. The extra slice prevents the input data from stalling on the last cycle. The capability of accumulating the P cascade through the X MUX feedback eliminates the pipeline stall.

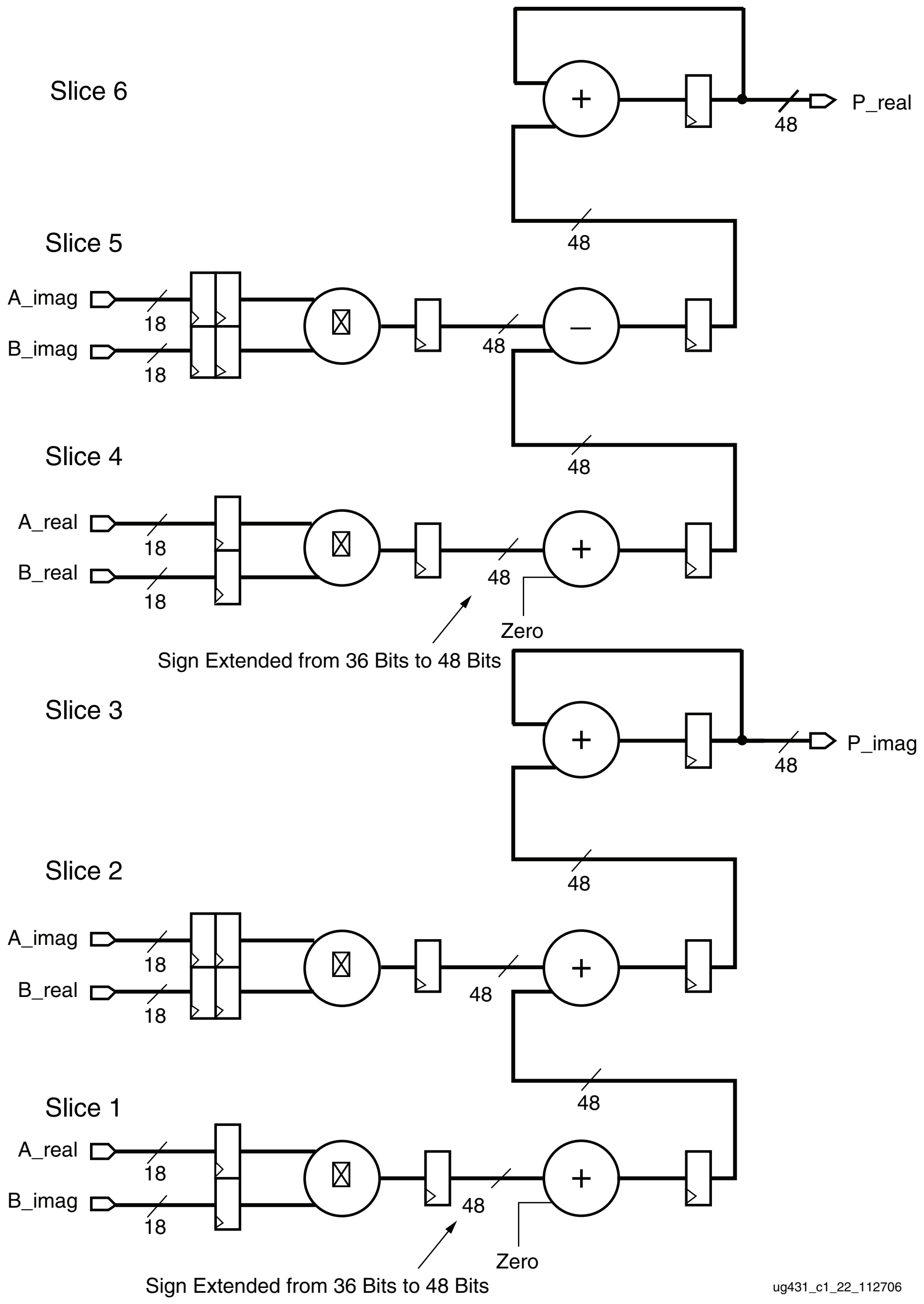


Figure 1-24: **Fully Pipelined, Complex, 18 x 18 MACC with Extra Slice**

Miscellaneous Functional Use Models

Table 1-8 summarizes a few common functional use models.

Table 1-8: **Miscellaneous Functional Use Models**

Miscellaneous	Silicon Utilization	OPMODE
18-bit Barrel Shifter	2 DSP slices	Static
48-bit Add Subtract	1 DSP slice	Static
36-bit Add Subtract Cascade	n DSP slices	Static
n word MUX, 48-bit words	2n DSP slices	Dynamic
n word MUX, 36-bit words	n DSP slices	Dynamic
48-bit Counter	1 DSP slice	Static
Magnitude Compare	1 DSP slice, logic	Static
Equal to Zero Compare	1 DSP slice, logic	Static
24 2-input ANDs	1 DSP slice	Static
24 2-input XORs	1 DSP slice	Static
Up to 48-bit AND	1 DSP slice	Static

Dynamic, 18-Bit Circular Barrel Shifter Use Model

Using two DSP48A slices and external fabric for shifting, an 18-bit circular barrel shifter can be implemented. The barrel shift function is useful when trying to quickly realign data. Using two DSP48A slices, an 18-bit circular barrel shifter can be implemented. This implementation shifts 18 bits of data left by the number of bit positions represented by n. The bits shifted out of the most-significant part reappear in the lower significant part of the answer, completing the circular shift. The equations in Figure 1-25 describe the value carried out of the first slice, what this value looks like after shifting right 17 bits, and finally what is visible as a result.

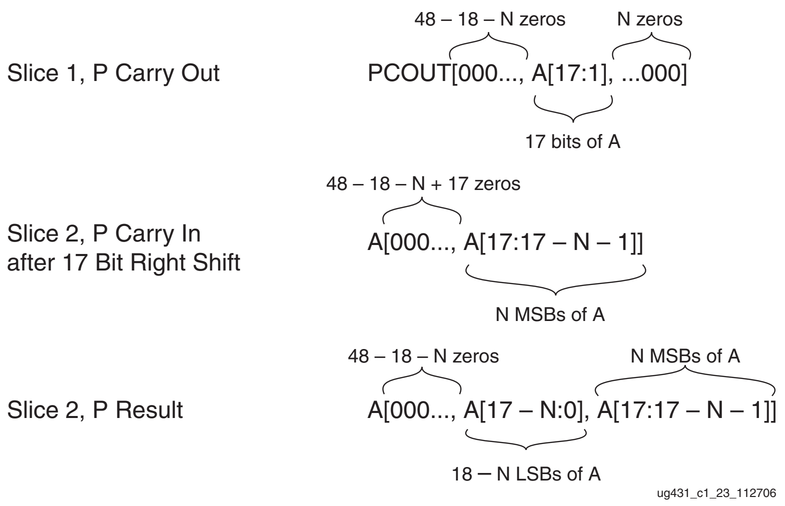


Figure 1-25: **Circular Barrel Shifter Equations**

Figure 1-26 shows the DSP48A used an 18-bit circular barrel shifter. The P register for slice 1 contains leading zeros in the MSBs, followed by the most-significant 17 bits of A, followed by n trailing zeros. If n equals zero, then are no trailing zeros and the P register contains leading zeros followed by 17 bits of A.

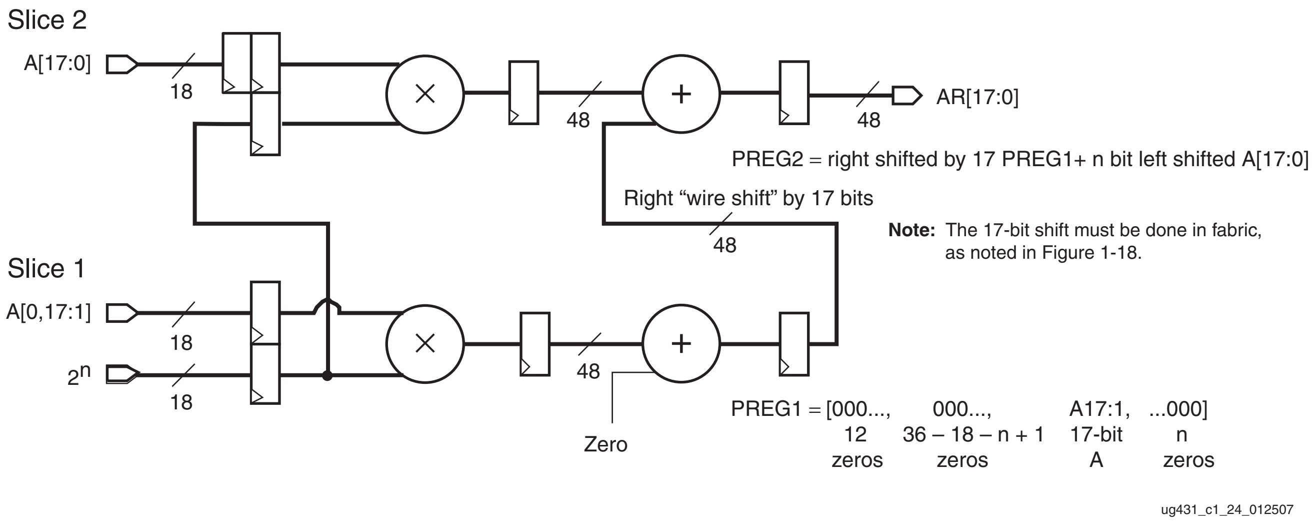

Slice 2
A[17:0]
18
48
48
AR[17:0]
PREG2 = right shifted by 17 PREG1+ n bit left shifted A[17:0]
Right "wire shift" by 17 bits
48
Note: The 17-bit shift must be done in fabric, as noted in Figure 1-18.
Slice 1
A[0,17:1]
18
2^n
18
48
Zero
PREG1 = [000..., 12 zeros, 000..., 36 – 18 – n + 1 zeros, A17:1, 17-bit A, ...000] n zeros
ug431_c1_24_012507

Figure 1-26: **Dynamic 18-Bit Barrel Shifter**

In the case of n equal to zero (i.e., no shift), the P register of slice 1 is passed to slice 2 with 17 bits of right shift. All 48 bits of the P carry input are effectively equal to zero because A[17:1] shifted toward the least-significant direction. If there is a positive shift amount, then P carry out of slice 1 contains A[17:1] padded in front by 48 – 17– n zeros and in back by n zeros. After the right shift by 17, only the n most-significant bits of A remain in the lower 48 bits of the P carry input.

This n-bit guaranteed positive number is added to the A[17:0], left shifted by n bits. In the n least-significant bits, there are zeros. The end result contained in A[17:0] of the second slice P register is A[17 – n:n, 17:17 – n + 1] or a barrel shifted A[17:0]. The design is fully pipelined and can generate a new result every clock cycle at the maximum DSP48A clock rate.

There is a corner case that requires two fabric flip-flops. This corner case only occurs when n=17. The following steps describe how to work around this corner case.

1. Take the MSB of the input scaling bus B[17] and route that to a fabric flip-flop called FF1, which then feeds the lower DSP48A subtract input (which is also configured with internal subtract pipeline FF).
2. Take the output of FF1 and feed that to the input of another fabric flip-flop called FF2. The output of FF2 then feeds the upper DSP48A subtract input (which is also configured with internal subtract pipeline FF).

Whenever a 17-bit left barrel shift is performed, then B[17] equals one. This is a negative 2**17 number instead of positive 2**17. However, with the various pipeline alignment FFs, the outputs of the multipliers are subtracted, which in essence is multiplying the result by -1, which effectively converts the -2**17 value to positive 2**17 shift parameter.

Additional Information

- For additional information on MACC FIR filters, etc., see the *XtremeDSP for Virtex-4 FPGAs* (UG073).
- For more information on the Spartan-3A Libraries, see the *Spartan-3A Libraries Guide for HDL Designs*.

Chapter 2

Using the DSP48A Pre-Adder

Introduction

The DSP48A includes a pre-adder before the multiplier. The hard pre-adder significantly increases the density, performance, and power efficiency of symmetric FIR filters and complex multipliers. See Figure 2-1 and Table 2-1.

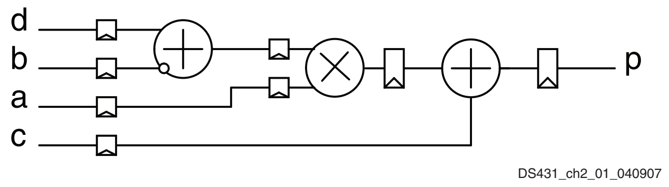


Figure 2-1: **DSP48A with Pre-Adder**

Table 2-1: **DSP48A with Pre-Adder**

Application	Pre-Adder	Density	Fmax
N tap Symmetric FIR	No	1 FIR Tap/DSP48A	250 MHz
N tap Symmetric FIR	Yes	2 FIR Tap/DSP48A	250 MHz
Complex Multiplier	No	1 Complex Multiplier/4 DSP48A	250 MHz
Complex Multiplier	Yes	1 Complex Multiplier/3 DSP48A	210 MHz

The pre-adder enables the DSP48A to implement the following equations (in Table 2-2) as selected by the OPMODE.

Table 2-2: **DSP48A Equations**

Operation	Function	OPMODE[6]	OPMODE[4]
Multiply	M = A * B	0	0
Pre-add	M = A * (D+B)	0	1
Pre-subtract	M = A * (D-B)	1	1

Symmetric FIR Filters

A *symmetric FIR filter* refers to a filter with symmetric coefficients. For example, an odd-symmetric filter contains an odd number of coefficients, such as [c0,c1,c2,c3,c2,c1,c0]. An even symmetric FIR filter has an even number of coefficients, such as [c0,c1,c2,c2,c1,c0]. Both types of FIR filter can be efficiently implemented using the DSP48A.

Figure 2-2 shows a simplified diagram of a non-symmetric FIR filter architecture that uses five pipelined DSP48As, in addition to delay and coefficient memories. This basic architecture supports multi-channel data streams, as well as interpolating and decimating multi-rate filters. It also supports the addition of a constant for rounding. The components shown in red are implemented in a single DSP48A.

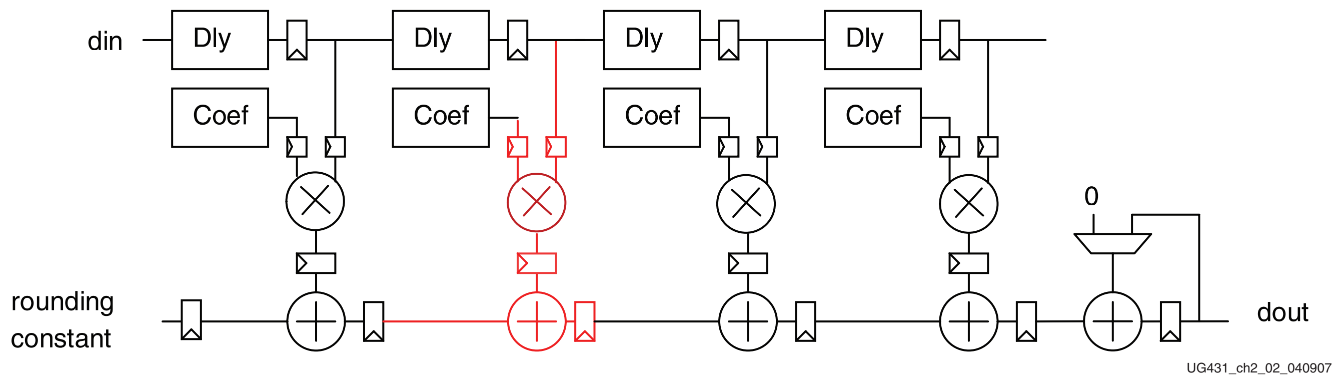


Figure 2-2: **Non-Symmetric FIR Filter Architecture- Simplified Diagram**

If the number of taps (N) in the filter exceeds the number of multipliers (M) in the filter implementation, the accumulator at the end of the FIR filter cascade is used to sum partial results for each set of taps. For this case, each data sample requires N/M clocks to complete. The delay and coefficient memories are used to store data and coefficient values. Data is written or copied from upstream delay memories to downstream memories at the data sample rate, once every N/M clocks. Data and coefficients are read from the delay and coefficient memories on each clock. The individual memories can be implemented using flip-flops (FFs), addressable shift registers, a single ported RAM or a simple dual ported RAM. Either SRLs, LUTRAM or block RAM can be used, but memory depths less than 16 generally map better to SRLs or LUTRAM than block RAM.

Symmetric FIR Filter Implementation

If the filter implements symmetric coefficients, the number of multipliers can be reduced from N multipliers to (N/2 +1) multipliers. This is possible because the FIR filter equation can be refactored to sum the delay data pairs with matching coefficients before multiplication, rather than after. For instance, the FIR filter with symmetric coefficients [c0 c1 c2 c1 c0] can be refactored from

dout = dly[0]*c0 + dly[1]*c1 + dly[2]*c2 + dly[3]*c1 dly[4]*c0;

to

dout = (dly[0]+dly[4])*c0 +(dly[1]+dly[3])*c1 + dly[2]*c2;

Figure 2-3 illustrates the basic even tap Symmetric FIR filter architecture.

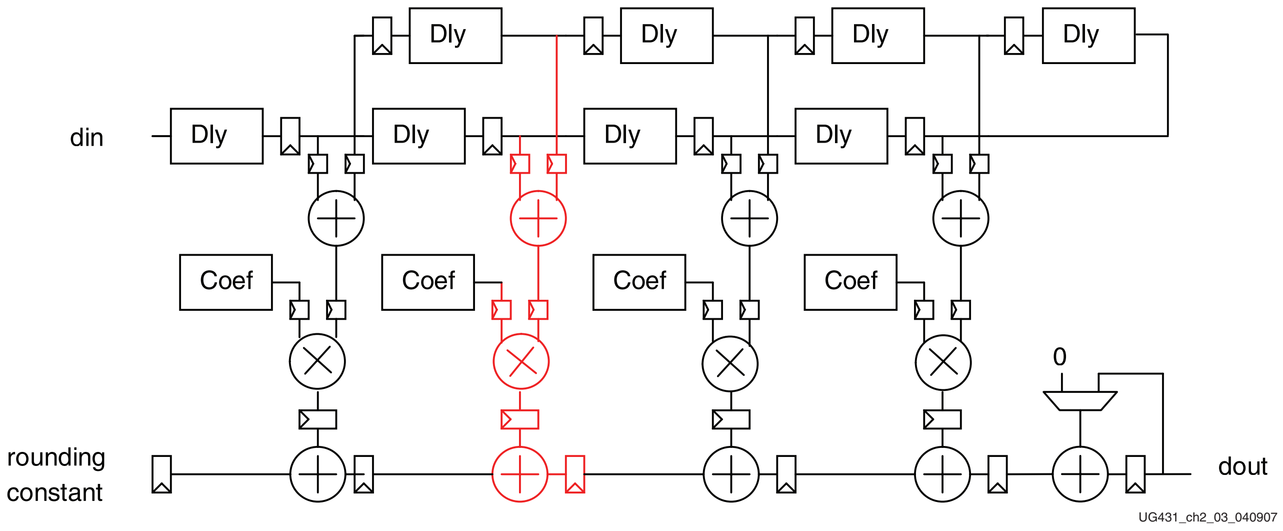


Figure 2-3: **Basic Even Tap Symmetric FIR Filter Architecture**

The odd tap implementation is shown in Figure 2-4.

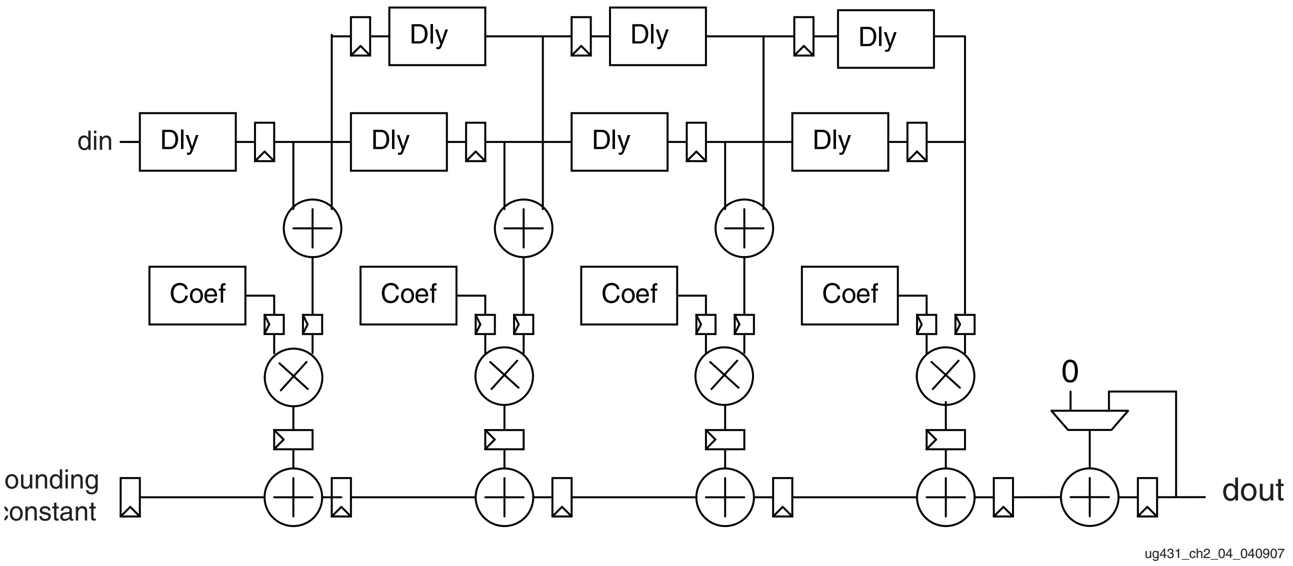


Figure 2-4: **Basic Odd Tap Symmetric FIR Filter Architecture**

For the symmetric FIR filter implementation, the FIR filter requires an additional set of delay memories which transfer data from back to front.

Pipelining the Reverse Delay Cascade

The reverse delay cascade is similar to the forward delay cascade, but pipeline balancing has a special issue when pipelining the reverse delay cascade. For the forward case, it is easy to add pipeline registers to match the pipeline registers inserted into the adder cascade. The additional pipeline register can be folded into a block RAM output register, or a SLICE FF can be used with LUTRAM memories or SRLs. These registers are also required for full speed operation. For the reverse cascade, however, a delay must be removed. For full rate or unfolded, single-channel FIR filters, the reverse pipeline can be removed entirely as shown in Figure 2-5.

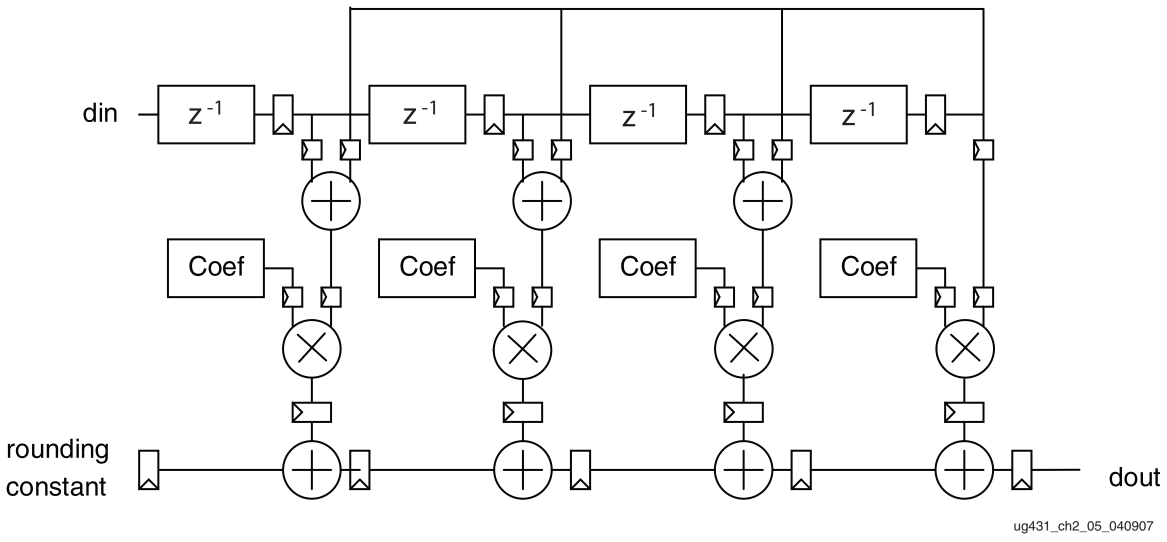


Figure 2-5: **Reverse Pipeline Removed**

For folded and multichannel FIR filters where the number of taps is greater than the number of multipliers, data delays and coefficient are stored in memories. Removing a delay for pipeline balancing can be accomplished with an addressing and control signal.

A special case is encountered when using single port memories as the delay blocks. Single port memories can be used as delay blocks if used in *read first* mode. This is advantageous because a true dual port 18-Kbit block RAM can be used to implement two independent 18-bit wide 9-Kbit single port memories. For non-symmetric FIRs, this enables a single block RAM to be used with each DSP48A, providing both coefficients and data delay memories. When using single port memories for the reverse cascade, the negative problem can be avoided by adding an additional *capture register* at the input of each delay memory. as shown in Figure 2-6.

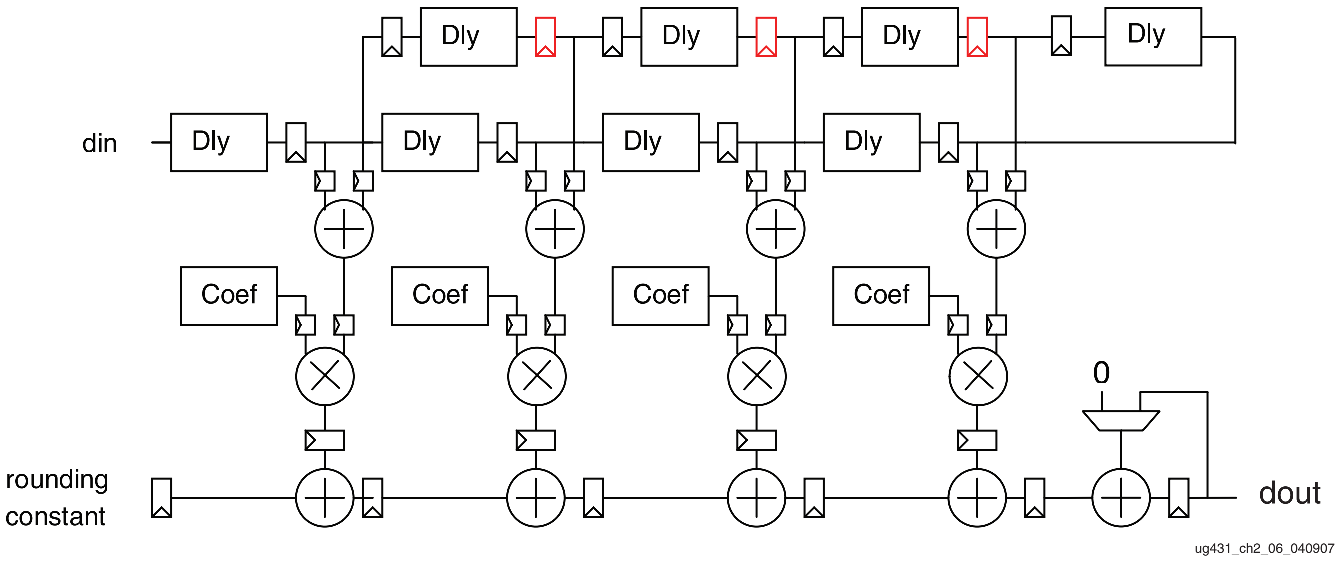


Figure 2-6: **Adding an Additional Capture Register**

The scheme works because in a folded FIR architecture the individual data samples are read out multiple times before being transferred to the downstream memories. The capture register is enabled using a clock enable to capture the required data, before it is needed. It is then written to the downstream memory when needed.

Complex Multipliers

Complex multipliers can be implemented using three or four multipliers. The four-multiplier equation is:

p.real = a.real * b.real - a.img * b.img;

p.img = a.real * b.img + a.img * b.real;

This can be refactored into a three-multiplier version which makes use of a pre-add function:

p.real = (a.real - a.img)* b.img + (b.real - b.img)* a.real;

cp.img = (a.real - a.img)* b.img + (b.real + b.img)* a.img;

The factor in red is common to both multipliers and can be implemented with a single multiplier.

The four-multiplier complex version is in Figure 2-7.

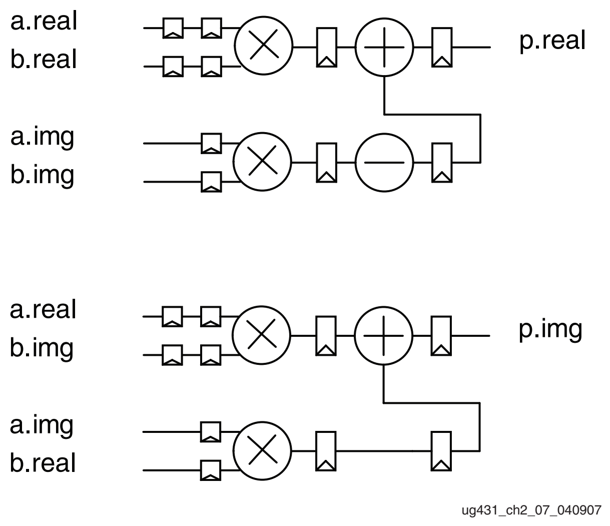


Figure 2-7: **Four Multiplier Complex Version**

The three multiplier complex version is shown in Figure 2-8.

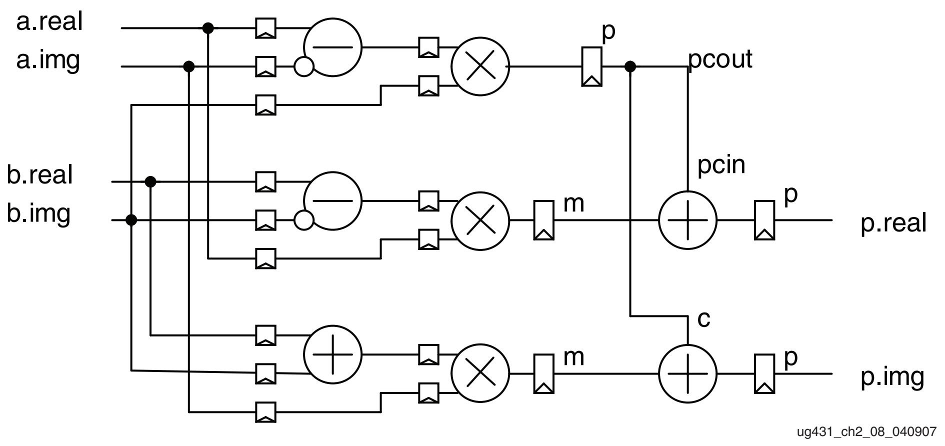


Figure 2-8: **Three Multiplier Complex Version**

This implementation suffers from the fact that the shared term generated by the upper DSP48A is required to be routed to the C-port of the lower DSP48A rather than using the PCIN port. Also, the MREG is not used in the upper DSP48A. The lack of pipelining reduces the speed of the three multiplier complex version using the DSP48A from 250 MHz to something greater than 200 MHz. If this performance is not acceptable, a fully pipelined version can be implemented as in Figure 2-9.

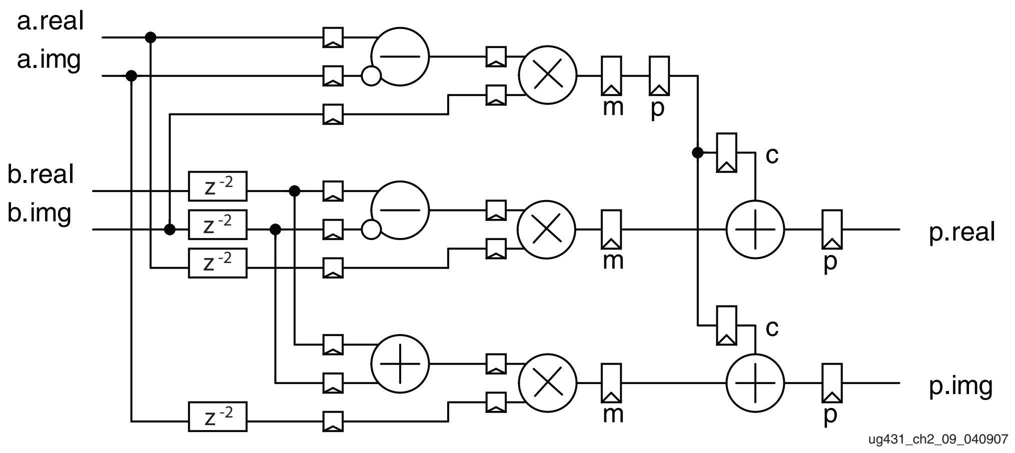


Figure 2-9: **Fully Pipelined Version**

Note that the input setup time to the pre-adder without using input registers is less than 2 ns. In many cases, this setup time is sufficiently low to meet a 250 MHz clock rate without using input registers, which gives some flexibility in pipelining complex circuits that use the pre-adder.