Spartan-6 FPGA DSP48A1 Slice

User Guide

UG389 (v1.2) May 29, 2014

XILINX®

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/2009	1.0	Initial Xilinx release.
08/13/2009	1.1	• Added Table 1-1, page 7. • Updated path in VHDL and Verilog Instantiation Templates, page 13. • Removed inversion bubbles into adders and replaced adders with adder/subtracters in Figure 1-1, page 8, Figure 1-3, page 14, and Figure 2-1, page 39. • Revised Figure 2-2, page 40, Figure 2-3, page 41, and Figure 2-4, page 41 to show the individual slices. • Added a register just after the din input of Figure 2-3, page 41. • Removed inversion bubbles into subtracters and replaced subtracters with adder/subtracters in Figure 2-6, page 42 and Figure 2-7, page 43.
05/29/2014	1.2	Changed *pre-adder* to *pre-adder/subtracter* throughout document. Clarified second sentence in third paragraph under Architecture Highlights. Added reference to the *XST Hardware Description Language (HDL) Coding Techniques under* Architecture Highlights. Updated the following signal names throughout document: CIN to CARRYIN, RSTCY to RSTCARRYIN, RSTO to RSTOPMODE, CECY to CECARRYIN, CEO to CEOPMODE, MFOUT to M, CCOUT to CARRYOUT, and CFOUT to CARRYOUTF. Updated signal descriptions in Table 1-2 and pin functions in Table 1-3. Clarified attribute settings in DSP48A1 Slice Attributes. Updated location of instantiation templates in VHDL and Verilog Instantiation Templates. Clarified notes relevant to Figure 1-3. Clarified A, B, C, D, P, and M Port Logic. Added Magnitude Compare Use Model and Timing Parameters. Updated Figure 1-24. Updated F_{MAX} frequencies in Table 2-1. Updated Figure 1-24. Added *Spartan-6 FPGA GTP Transceivers User Guide* to Additional Documentation and *XST User Guide for Virtex-6 and Spartan-6 Devices* to Additional Documentation Resources in Appendix A.

Table of Contents

Chapter 1

Design Considerations

This user guide is a detailed functional description of the DSP48A1 slice in Spartan®-6 FPGAs. Complete and up-to-date documentation of the Spartan-6 family of FPGAs is available on the Xilinx website at http://www.xilinx.com/products/spartan6.

This chapter provides technical details for the digital signal processing (DSP) element in Spartan-6 FPGAs, the DSP48A1 slice. Each DSP48A1 slice forms the basis of a versatile, coarse-grained DSP architecture. The DSP48A1 is a subtle evolution of the DSP48A block that is part of the Extended Spartan-3A DSP family.

Many DSP designs follow a multiply with addition. In Spartan-6 devices, these elements are supported in dedicated circuits.

The DSP48A1 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 DSP48A1 slices to form wide math functions, DSP filters, and complex arithmetic without the use of general FPGA logic.

The DSP48A1 slices in all Spartan-6 family members support DSP algorithms and high levels of DSP integration similar to those DSP slices available in legacy Xilinx® FPGAs. Minimal use of general FPGA logic leads to low power, very high performance, and efficient silicon utilization.

This chapter contains the following sections:

- Introduction
- Architecture Highlights
- Performance
- DSP48A1 Slice Description
- DSP48A1 Slice in Detail
- Simplified DSP48A1 Slice Operation
- A, B, C, D, P, and M Port Logic
- FIR Filters
- Adder Cascade Versus Adder Tree
- DSP48A1 Slice Functional Use Models

Chapter 2, Using the DSP48A1 Pre-Adder describes how the pre-adder/subtracter increases the density, performance, and power efficiency of symmetric FIR filters and complex multipliers.

Introduction

The DSP48A1 is derived from the DSP48A slice in Extended Spartan-3A FPGAs (refer to *XtremeDSP™ DSP48A for Spartan-3A DSP FPGAs User Guide* [Ref 1] for more information).

Many DSP algorithms are supported with minimal use of the general-purpose FPGA logic, resulting in low power, high performance, and efficient device utilization. At first look, the DSP48A1 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. 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 DSP48A1 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 DSP48A1 slice utilization in symmetric filters.

Architecture Highlights

The DSP48A1 slices are organized as vertical DSP columns. Within the DSP column, a single DSP slice is combined with extra logic and routing. The vertical column configuration is ideal to connect a DSP slice to its two adjacent neighboring slices, hence cascading those blocks and facilitating the implementation of systolic DSP algorithms.

Each DSP48A1 slice has a selectable 18-bit pre-adder/subtracter. This pre-adder/subtracter accepts 18-bit, two's-complement inputs and generates an 18-bit, two's-complement result. The pre-adder/subtracter 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 36-bit output of the multiplier (or of the M register) can be routed directly to the FPGA logic. 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 DSP48A1 slices in a DSP48A1 column. Dedicated cascade paths include from BCOUT to BCIN, from PCOUT to PCIN, and from CARRYOUT to CARRYIN. 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 25.

Architecture highlights of the DSP48A1 slices are:

- Two-input pre-adder/subtracter 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 DSP48A1 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
- Advanced carry management (cascadable, register capable, and routable to the user logic)
- Direct 36-bit multiplier output to the user logic
- 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 DSP48A1 slice. The Xilinx ISE® software tools support DSP48A1 slice instantiations. Synthesis tools such as XST also support DSP48A1 inference. CORE Generator™, System Generator™ for DSP, and AccelDSP™ tools use the DSP48A1 block in designs targeting the Spartan-6 family, making it easier for a designer to quickly generate math, DSP, or similar functions using the DSP48A1 slices.

Refer to the *XST Hardware Description Language (HDL) Coding Techniques* chapter in the *XST User Guide for Virtex-6 and Spartan-6 Devices* [Ref 3], which includes VHDL and Verilog inference coding templates for various DSP functions.

Performance

To achieve the maximum performance when using the DSP48A1 slice, the users need to fully pipeline their designs. See the timing numbers in the DSP48A1 Switching Characteristics section of the *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics* for timing information regarding how the pipeline stages affect the performance. See Timing Parameters, page 37 for more description of the data sheet specifications.

DSP48A1 Slice Description

The Spartan-6 family offers a high ratio of DSP48A1 slices to logic, making it ideal for math-intensive applications. Table 1-1 shows the number of DSP48A1 slices for each device in the Spartan-6 family.

Table 1-1: **Number of DSP48A1 Slices per Family Member**

Device	Total DSP48A1 Slices per Device	Number of DSP48A1 Columns per Device	Number of DSP48A1 Slices per Column
XC6SLX4	8	1	8
XC6SLX9	16	1	16
XC6SLX16	32	2	16
XC6SLX25	38	2	18/20
XC6SLX45	58	2	30/28

Table 1-1: **Number of DSP48A1 Slices per Family Member *(Cont'd)***

Device	Total DSP48A1 Slices per Device	Number of DSP48A1 Columns per Device	Number of DSP48A1 Slices per Column
XC6SLX75	132	3	44/40/48
XC6SLX100	180	4	48/44/40/48
XC6SLX150	180	4	48/44/40/48
XC6SLX25T	38	2	18/20
XC6SLX45T	58	2	30/28
XC6SLX75T	132	3	44/40/48
XC6SLX100T	180	4	48/44/40/48
XC6SLX150T	180	4	48/44/40/48

Figure 1-1 is a top-level block diagram showing the basic functionality of the DSP48A1 slice.

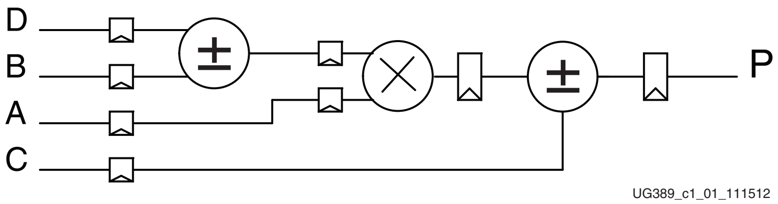


Figure 1-1: **Simplified DSP48A1 Slice with Pre-Adder**

DSP48A1 Slice Primitive

Figure 1-2 shows the DSP48A1 slice primitive.

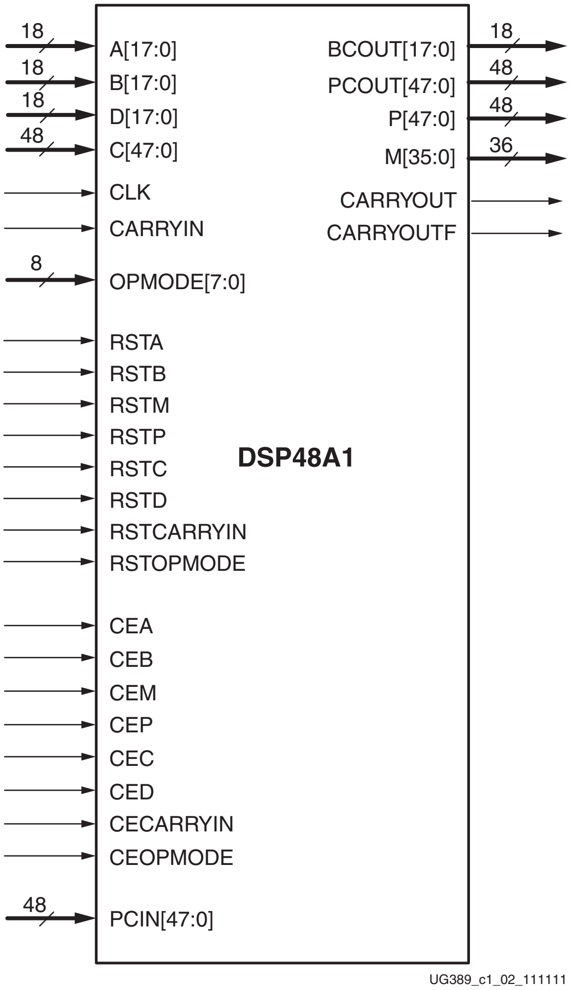


Figure 1-2: **DSP48A1 Slice Primitive**

Table 1-2 defines the available ports in the DSP48A1 slice primitive.

Table 1-2: **DSP48A1 Slice Port Descriptions**

Signal Name	Direction	Size	Function
Data Ports			
A	Input	18	18-bit data input to multiplier, and optionally to post-adder/subtracter depending on the value of OPMODE[1:0].
B	Input	18	18-bit data input to pre-adder/subtracter, to multiplier depending on OPMODE[4], or to post-adder/subtracter depending on OPMODE[1:0]. The DSP48A1 component uses this input when cascading BCOUT from an adjacent DSP48A1 slice. The tools then translate BCOUT cascading to the dedicated BCIN input and set the B_INPUT attribute for implementation.
C	Input	48	48-bit data input to post-adder/subtracter.

Table 1-2: **DSP48A1 Slice Port Descriptions *(Cont'd)***

Signal Name	Direction	Size	Function
CARRYOUT	Output	1	Cascade carry out signal from post-adder/subtracter. It can be registered in CARRYOUTREG = 1) or unregistered (CARRYOUTREG = 0). Dedicated cascade that connects only to CARRYIN of adjacent DSP48A1. If not used, leave this output unconnected.
CARRYOUTF	Output	1	Carry out signal from post-adder/subtracter for use in the FPGA logic. It is a copy of the CARRYOUT signal that can be routed to the user logic.
CARRYIN	Input	1	Cascaded, external carry input to the post-adder/subtracter. Should only be connected to the CARRYOUT pin of another DSP48A1 block.
D	Input	18	18-bit data input to pre-adder/subtracter. D[11:0] are concatenated with A and B and optionally sent to post-adder/subtracter depending on the value of OPMODE[1:0].
M	Output	36	36-bit buffered multiplier data output, routable to the FPGA logic. It is either the output of the M register (MREG = 1) or the direct output of the multiplier (MREG = 0). The M and P outputs should not be used at the same time due to potential routing congestion.
P	Output	48	Primary data output from the post-adder/subtracter. It is either the output of the P register (PREG = 1) or the direct output of the post-adder/subtracter (PREG = 0). The M and P outputs should not be used at the same time due to potential routing congestion.
Control Input Ports			
CLK	Input	1	DSP48A1 clock.
OPMODE	Input	8	Control input to select the arithmetic operations of the DSP48A1 slice (see Table 1-7).
Reset/Clock Enable Input Ports			
CEA	Input	1	Clock enable for the A port registers: A0REG = 1 or A1REG = 1. Tie to logic one if not used and A0REG = 1 or A1REG = 1. Tie to logic zero if A0REG = 0 and A1REG = 0.
CEB	Input	1	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	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.
CECARRYIN	Input	1	Clock enable for the carry-in register (CYI) and the carry-out register (CYO). Tie to logic one if not used and CARRYINREG = 1 or CARRYOUTREG = 1. Tie to a logic zero if CARRYINREG = 0 and CARRYOUTREG = 0.
CED	Input	1	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	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.
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.

Table 1-2: **DSP48A1 Slice Port Descriptions *(Cont'd)***

Signal Name	Direction	Size	Function
CEP	Input	1	Clock enable for the P output port registers (PREG = 1). Tie to logic one if not used and PREG = 1. Tie to a logic zero if PREG = 1.
RSTA	Input	1	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	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	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.
RSTCARRYIN	Input	1	Reset for the carry-in register CYI and the carry-out register CYO. 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	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	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.
RSTOPMODE	Input	1	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.
RSTP	Input	1	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.
Cascade Ports			
BCOUT	Output	18	Cascade output for Port B. If used, connect to the B port of the downstream cascaded DSP48A1. The tools translate the BCOUT cascade to the dedicated BCIN input and set the B_INPUT attribute for implementation. If not used, this port should be left unconnected.
PCIN	Input	48	Cascade input for Port P. If used, connect to the PCOUT of the upstream cascaded DSP48A1. If not used, this port should be tied to all zeros.
PCOUT	Output	48	Cascade output for Port P. If used, connect to the PCIN of the downstream cascaded DSP48A1. If not used, this port should be left 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 zeros (disable the post-adder/subtracter and propagate the Z result to P) 1 – Use the multiplier product 2 – Use the P output signal (accumulator) 3 – Use the concatenated D:A:B input signals
OPMODE[3:2]	**Specifies the source of the Z input to the post-adder/subtracter**
	0 – Specifies to place all zeros (disable the post-adder/subtracter and propagate the multiplier product or other X result to P) 1 – Use the PCIN 2 – Use the P output signal (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 the carry input of the carry-in register (CYI) or direct to the CIN 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 (D-B)
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 (Z-(X+CIN))

DSP48A1 Slice Attributes

The synthesis attributes for the DSP48A1 slice are described in detail throughout this section. With the exception of the 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). A0 and B0 are the first stages of the pipelines. A1 and B1 are the second stages of the pipelines. The B1 register is after the pre-adder/subtracter. See the A, B, C, D, P, and M 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 1-10, page 19) and at the output of the post-adder/subtracter (PREG) (shown in Figure 1-8, page 18). The PREG attribute controls both the dedicated PCOUT cascade and the P connection to the FPGA logic. The CREG attribute is used to select the pipeline register at the C input (shown in Figure 1-7, page 18). The DREG attribute is used to select the pipeline register at the D input to the pre-adder/subtracter (shown in Figure 1-6, page 17). CREG, DREG, MREG, and PREG all default to `1` (registered).

- The CARRYINREG, CARRYOUTREG, 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 CARRYOUTREG attribute controls both the dedicated CARRYOUT cascade and the CARRYOUTF connection to the FPGA logic. CARRYINREG, CARRYOUTREG, and OPMODEREG all default to `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 DSP48A1 slice) or dynamically controlled from the FPGA logic by the OPMODE[5] input. This attribute can be set to the string CARRYIN or OPMODE5. CARRYINSEL defaults to OPMODE5.
- 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 DSP48A1 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 the DSP48A1 slice. If the B port is connected to the BCOUT of another DSP48A1 slice, then the tools automatically set the attribute to "CASCADE", otherwise it is set to "DIRECT".
- The RSTTYPE attribute selects whether all resets for the DSP48A1 slice 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. Only reset is directly supported, not set. For this reason, a set (value equals logic `1` upon an applied signal) should not be coded around logic that can be implemented within a DSP48A1 slice.

VHDL and Verilog Instantiation Templates

The VHDL and Verilog instantiation templates for the DSP48A1 slice can be found in the ISE tool through the following choice of menus:

```
ISE → Edit → Language Templates → VHDL || Verilog →
Device Primitive Instantiation → Spartan-6 → Arithmetic
Functions → 48-bit Multi-Functional Arithmetic Block (DSP48A1)
```

DSP48A1 Slice in Detail

Figure 1-3 shows a DSP48A1 slice and its associated datapaths. 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).

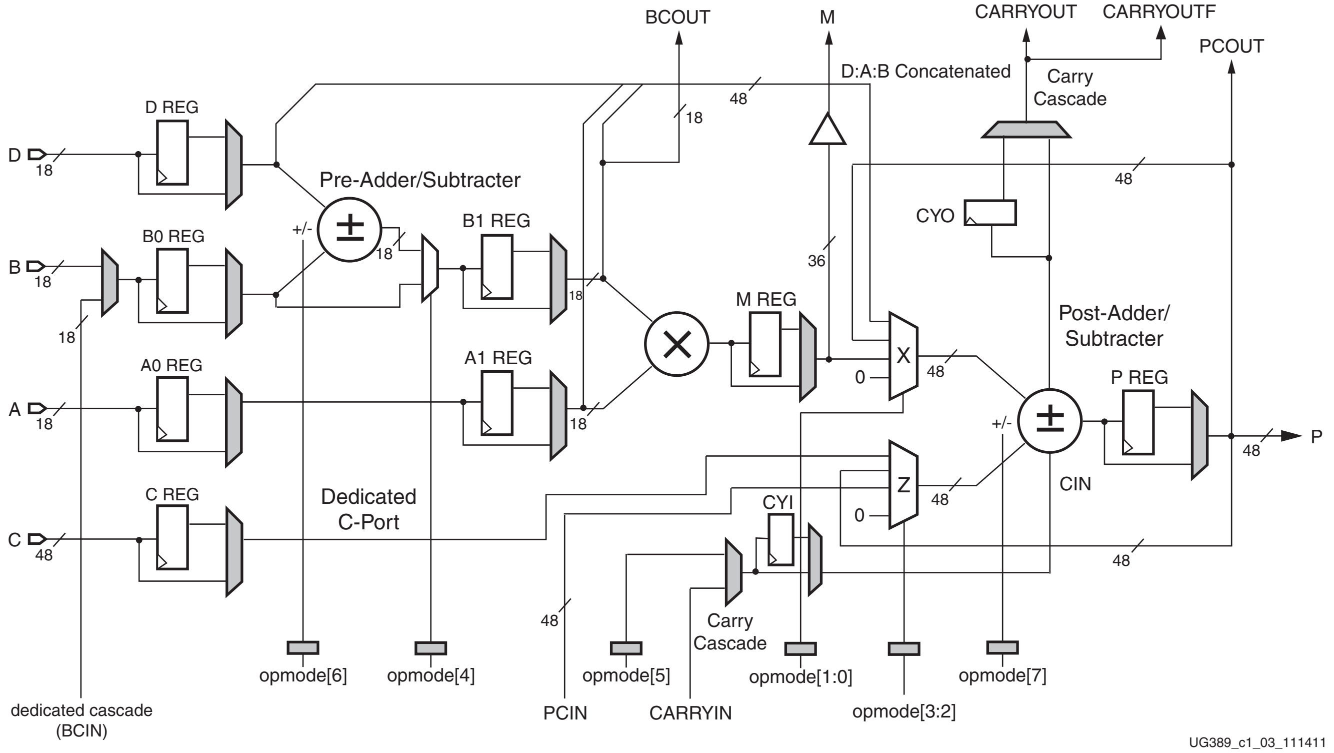


Figure 1-3: **DSP48A1 Slice**

Notes relevant to Figure 1-3:

1. The 18-bit A, B, and D buses 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 that can be optionally registered to the M register.
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. The clear multiplexers are controlled by OPMODE inputs, allowing dynamic changes to functionality.
6. The C register supports multiply-add or wide addition operations.
7. Enabling SUBTRACT implements Z – (X + CIN) 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. Enabling SUBTRACT implements D – B at the output of the pre-adder/subtracter.

9. Inputs (B and D) and the output of the pre-adder are 18 bits wide. Because two 18-bit inputs can produce an overflow, the user must take this situation into consideration.
10. CARRYOUTF is a copy of CARRYOUT but dedicated to applications in the FPGA logic, whereas CARRYOUT is the dedicated route to the adjacent DSP48A1 slice.
11. The registered output of the multiplier or its direct output can be routed to the FPGA logic through a 36-bit vector called M.
12. The BCIN input is the direct cascade from the adjacent DSP48A1 BCOUT.

Simplified DSP48A1 Slice Operation

The math portion of the DSP48A1 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 datapath multiplexers (with outputs X and Z) followed by a two-input, 48-bit post-adder/subtracter as shown in Figure 1-1, page 8.

The data and control inputs to the DSP48A1 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 uses the pipeline registers. Maximum frequency is shown in the Spartan-6 FPGA Data Sheet: DC and Switching Characteristics.

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 CIN by the post-adder/subtracter. The CIN and X multiplexer output are always added together. This combined result can be selectively added to or subtracted from the Z multiplexer output.

$$Adder\ Out = (Z \pm (X + CIN)) \qquad \text{Equation 1-1}$$

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.

$$Adder\ Out = C \pm (A \times B + CIN) \qquad \text{Equation 1-2}$$

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.

$$Adder\ Out = C \pm (A \times (D \pm B) + CIN) \qquad \text{Equation 1-3}$$

Figure 1-4 shows the DSP48A1 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 and CARRYINSEL control the function of the pre- and post-adder/subtracters. See Table 1-7 for more details regarding the opcode combinations.

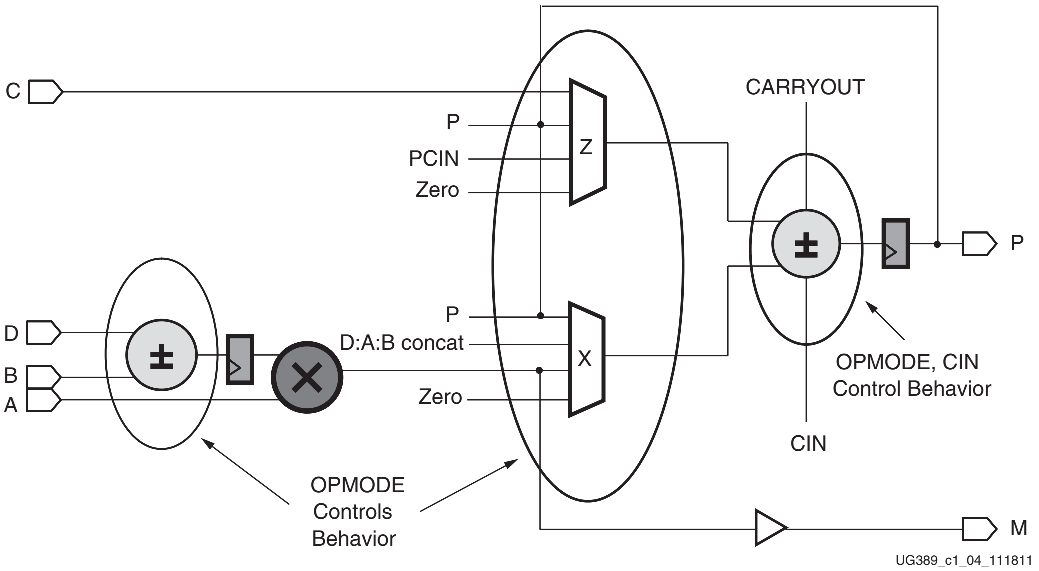


Figure 1-4: **Simplified DSP48A1 Slice Model**

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

The DSP48A1 input and output data ports support many common DSP and math algorithms. The DSP48A1 slice has three direct 18-bit input data ports labeled A, B, and D and a 48-bit input data port labeled C. Each DSP48A1 slice has one direct 48-bit output port labeled P. Each DSP48A1 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 DSP48A1 slices. The B cascade is a dedicated resource always connected to the adjacent DSP48A1; when used, the tools automatically select the path via the B_INPUT attribute. Also included in the DSP48A1 slice are a CARRYIN input, and a CARRYOUT output. These resources are used to facilitate the creation of parallel MACC operations in adjacent DSP48A1 slices. The PCIN cascade is a dedicated resource that is always connected to the adjacent DSP48A1 slice and can be dynamically selected via the Z MUX (OPMODE 3:2).

The M port is the raw 36-bit output of the multiplier that can be routed optionally to the FPGA logic.

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).

The 18-bit D and B inputs 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, page 9 and shown in Figure 1-2, page 9.

The logic for the A, B/D, C, and P ports is shown in Figure 1-5, Figure 1-6, Figure 1-7, and Figure 1-8, respectively.

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

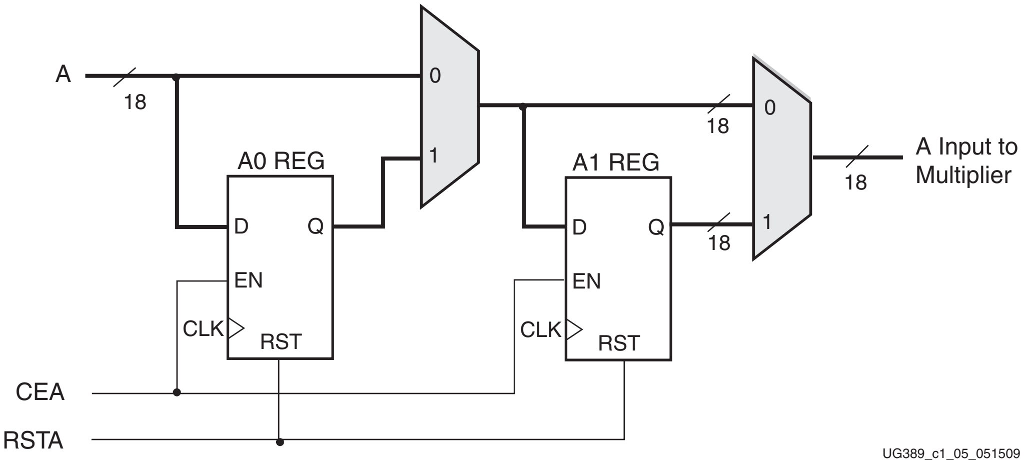


Figure 1-5: **A Input Logic**

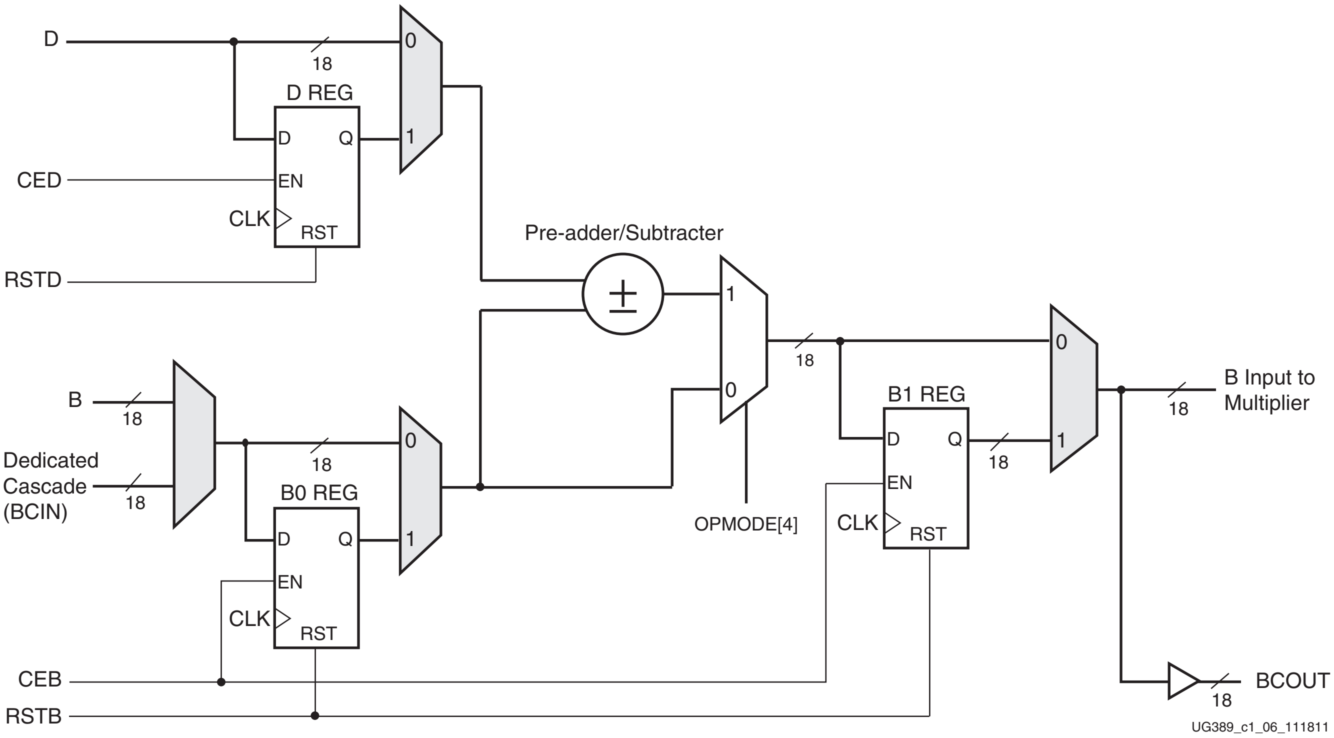


Figure 1-6: **B and D Input Logic**

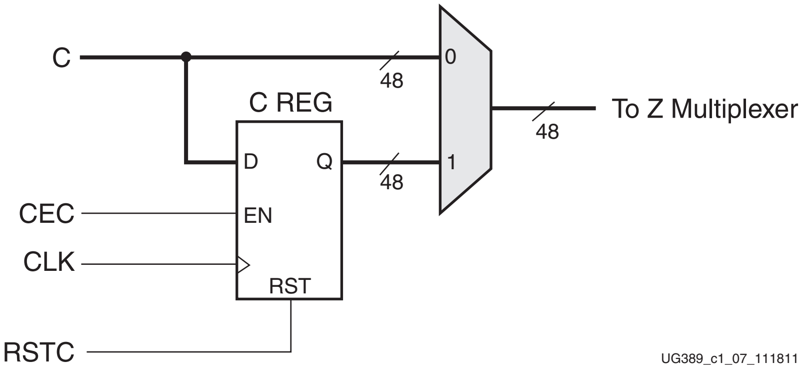


Figure 1-7: **C Input Logic**

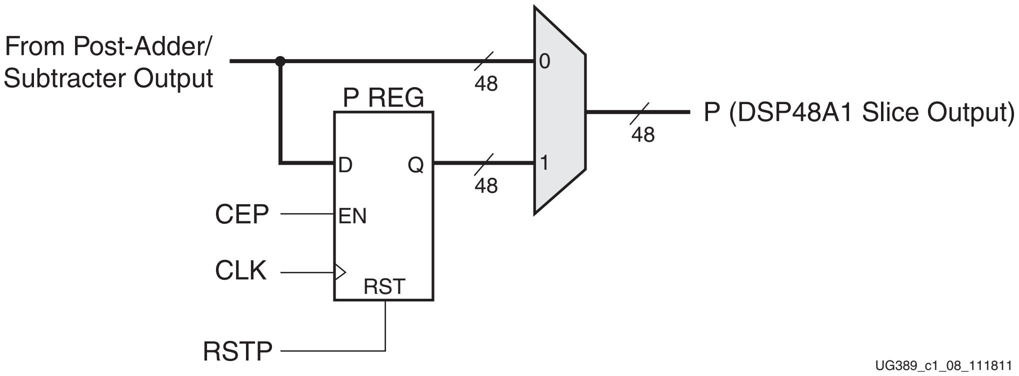


Figure 1-8: **P Output Logic**

OPMODE Port Logic

The OPMODE port logic supports flow through or registered input control signals. Similar to the datapaths, multiplexers controlled by configuration bits select flow through or optional registers (OPMODEREG = 1). 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 1-2, page 9. The OPMODE registers are reset by RSTOPMODE. The OPMODE register has a separate enable labeled CEOPMODE. Figure 1-9 shows the OPMODE port logic.

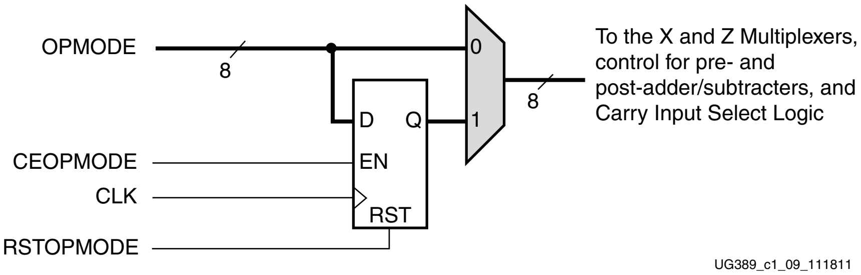


Figure 1-9: **OPMODE Port Logic**

Two's Complement Multiplier and M Output Port

The two's-complement multiplier inside the DSP48A1 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 36-bit result of the multiplier, being optionally registered, is routed to the X multiplexer input and optionally to the FPGA logic via the M port after being properly buffered. The 36-bit vector that feeds the X multiplexer can also feed the CLBs. Some buffering has been added, and the resulting buffered vector M can be routed to the FPGA logic. This feature ensures Spartan-6 devices have some downward compatibility with the MULT18X18 slice available in previous Spartan families. This is a direct route that bypasses the X multiplexer and the post-adder, hence reducing latency.

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.

Note: The M and P outputs should *not* be used at the same time due to potential routing congestion.

Figure 1-10 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."

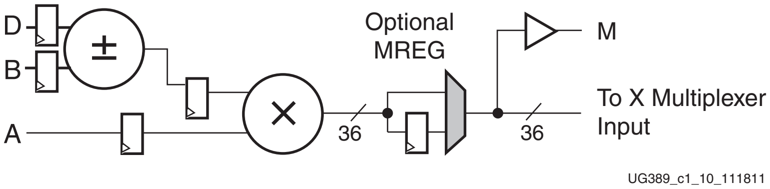


Figure 1-10: **Two's Complement Multiplier Followed by Optional M REG**

X and Z Multiplexers

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 DSP48A1 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 1-9).

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 CIN bit
OPMODE[4]	Use Pre-Adder/Subtracter for B input to multiplier 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 bits)
- Multiplier bypass bus consisting of A concatenated with B and D
- C bus (48 bits)
- Cascaded P bus (48 bits) from a neighboring DSP48A1 slice
- Registered P bus output (48 bits) for accumulator functions

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

Table 1-7 shows a complete description of all the DSP48A1 operating modes.

Table 1-7: **DSP48A1 Operating Modes**

OPMODE	OPMODE[7:0]								CARRYINSEL	Output
	Post-Adder/ Subtracter	Pre_Adder/ Subtracter	CIN	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 + CIN)[1][2]
D:A:B Select	0/1	0/1	0/1	0	0	0	1	1	0/1	± (D:A:B + CIN)
D:A:B Select with PreAdd/Subtract	0/1	0/1	0/1	1	0	0	1	1	0/1	± (D:A:(D ±B) + CIN)
Multiply	0/1	0/1	0/1	0	0	0	0	1	0/1	±(A × B + CIN)
PreAdd-Multiply	0/1	0	0/1	1	0	0	0	1	0/1	±(A × (D + B) + CIN)
PreSubtract-Multiply	0/1	1	0/1	1	0	0	0	1	0/1	± (A × (D – B) + CIN)
P Cascade Select	0/1	0/1	0/1	0/1	0	1	0	0	0/1	PCIN ±CIN
P Cascade Feedback Add/Subtract	0/1	0/1	0/1	0/1	0	1	1	0	0/1	PCIN ±(P + CIN)[2]
P Cascade Add/Subtract	0/1	0/1	0/1	0	0	1	1	1	0/1	PCIN ±(D:A:B + CIN)
P Cascade Add/Subtract with PreAdd/ Subtract	0/1	0/1	0/1	1	0	1	1	1	0/1	PCIN ± (D:A:(D ± B) + CIN)[2]
P Cascade Multiply Add/Subtract	0/1	0/1	0/1	0	0	1	0	1	0/1	PCIN ±(A × B + CIN)
P Cascade PreAdd-Multiply	0/1	0	0/1	1	0	1	0	1	0/1	PCIN ± (A × (D + B) + CIN)
P Cascade PreSubtract-Multiply	0/1	1	0/1	1	0	1	0	1	0/1	PCIN ± (A × (D – B) + CIN)
Feedback Carryin Add/Subtract	0/1	0/1	0/1	0/1	1	0	0	0	0/1	P ± CIN[2]
Double Feedback Add/Subtract	0/1	0/1	0/1	0/1	1	0	1	0	0/1	P ± (P + CIN)[2]
Feedback Add/Subtract	0/1	0/1	0/1	0	1	0	1	1	0/1	P ± (D:A:B + CIN)[2]
Feedback Add with PreAdd/Subtract	0/1	0/1	0/1	1	1	0	1	1	0/1	P ± (D:A:(D ± B) + CIN)[2]
Multiply-Accumulate	0/1	0/1	0/1	0	1	0	0	1	0/1	P ± (A × B + CIN)[2]
Feedback PreAdd-Multiply	0/1	0	0/1	1	1	0	0	1	0/1	P ± (A × (D + B) + CIN)[2]
Feedback PreSubtract-Multiply	0/1	1	0/1	1	1	0	0	1	0/1	P ± (A × (D – B) + CIN)[2]
C Select	0/1	0/1	0/1	0/1	1	1	0	0	0/1	C ± CIN
C Feedback Add/Subtracter	0/1	0/1	0/1	0/1	1	1	1	0	0/1	C ± (P + CIN)[2]
48-Bit Adder/Subtracter	0/1	0/1	0/1	0	1	1	1	1	0/1	C ± (D:A:B + CIN)
C Multiply-Add/Subtracter	0/1	0/1	0/1	0	1	1	0	1	0/1	C ± (A × B + CIN)
C PreAdd-Multiply	0/1	0	0/1	1	1	1	0	1	0/1	C ± (A × (D + B) + CIN)
C PreSubtract-Multiply	0/1	1	0/1	1	1	1	0	1	0/1	C ± (A × (D – B) + CIN)
48-Bit Adder/Subtracter with PreAdd/ Subtract	0/1	0/1	0/1	1	1	1	1	1	0/1	C ± (D:A:(D ± B) + CIN)

Notes:

1. For the remaining functions, CIN = CARRYIN from the upstream DSP48A1 block or OPMODE[5] depending upon programming of CARRYINSELECT.
2. Output mux must be selecting the register output.

Pre-Adder/Subtracter

The pre-adder/subtracter output is a function of OPMODE[6] and the B and D data inputs. The pre-adder/subtracter reduces resource utilization for symmetric filters. Add or subtract is defined by the OPMODE[6] control signal (OPMODE[6] = 1 is defined as “subtraction”, D - B.). The pre-adder/subtracter 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, select the inputs to the X or Z multiplexer directed to the associated post-adder/subtracter inputs.

Table 1-4 through Table 1-6 show OPMODE combinations and the resulting functions. The symbol ± in Table 1-7 means either add or subtract and is specified by the state of the 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 CIN is a function of the CARRYINSELECT configuration bit, OPMODE[5], and CARRYIN. The inputs to the carry input logic appear in Figure 1-11. 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 (CIN) is selected.

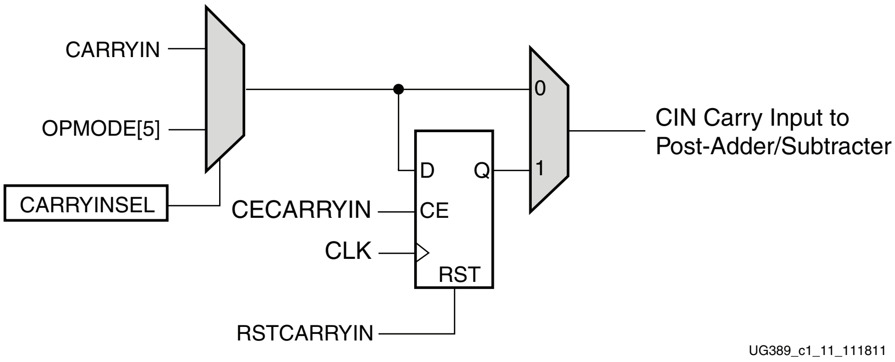


Figure 1-11: **Carry Input Logic Feeding the Post-Adder/Subtracter**

Figure 1-11 shows two inputs, selected by the CARRYINSEL programming bit. The first input CARRYIN (CARRYINSEL = CARRYIN) is driven from the carry-out of the upstream DSP48A1 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 DSP48A1 blocks. It can be optionally registered in CYO to match the pipeline delay of the MREG when used. This register delay is controlled by the CARRYINREG configuration bit.

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 DSP48A1 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 DSP48A1 slice. Each slice multiplies the samples with the corresponding coefficients within the DSP48A1 slice. The outputs of the multipliers are combined in the cascaded adders.

In Figure 1-12, 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.

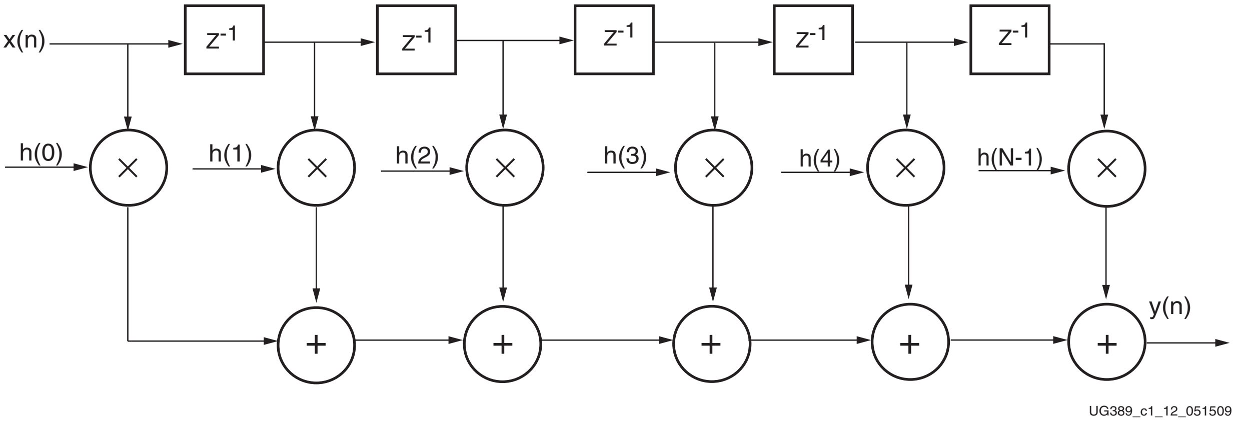


Figure 1-12: **Conventional Tapped Delay Line FIR Filter**

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 1-13 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.

$$x(n) = x_I(n) + jx_Q(n)$$ *Equation 1-5*

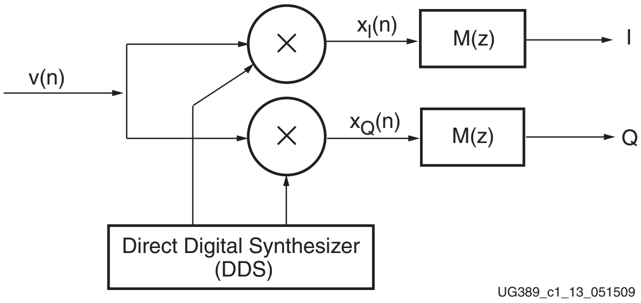


Figure 1-13: **Software-Defined Radio Digital Down Converter**

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 1-4, page 16, 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. 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 DSP48A1 slice shown in Figure 1-4, the X multiplexer is set to multiply and the cascaded input from another DSP48A1 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* [Ref 2] 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 1-14).

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 DSP48A1 column consisting entirely of dedicated silicon.

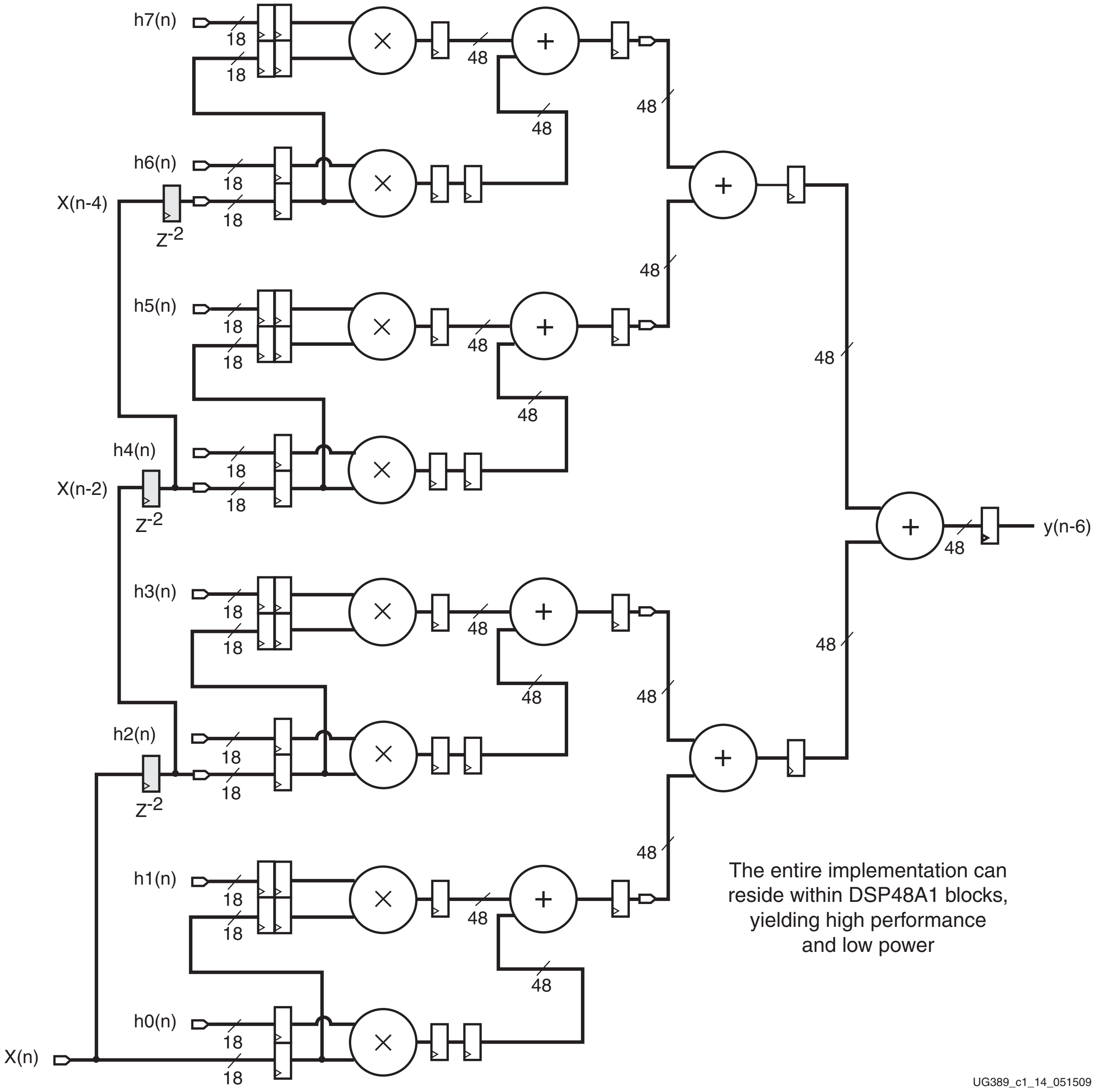


Figure 1-14: **FIR Filter Adder Tree Using DSP48A1 Slices**

The Spartan-6 FPGA 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 1-15. 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.

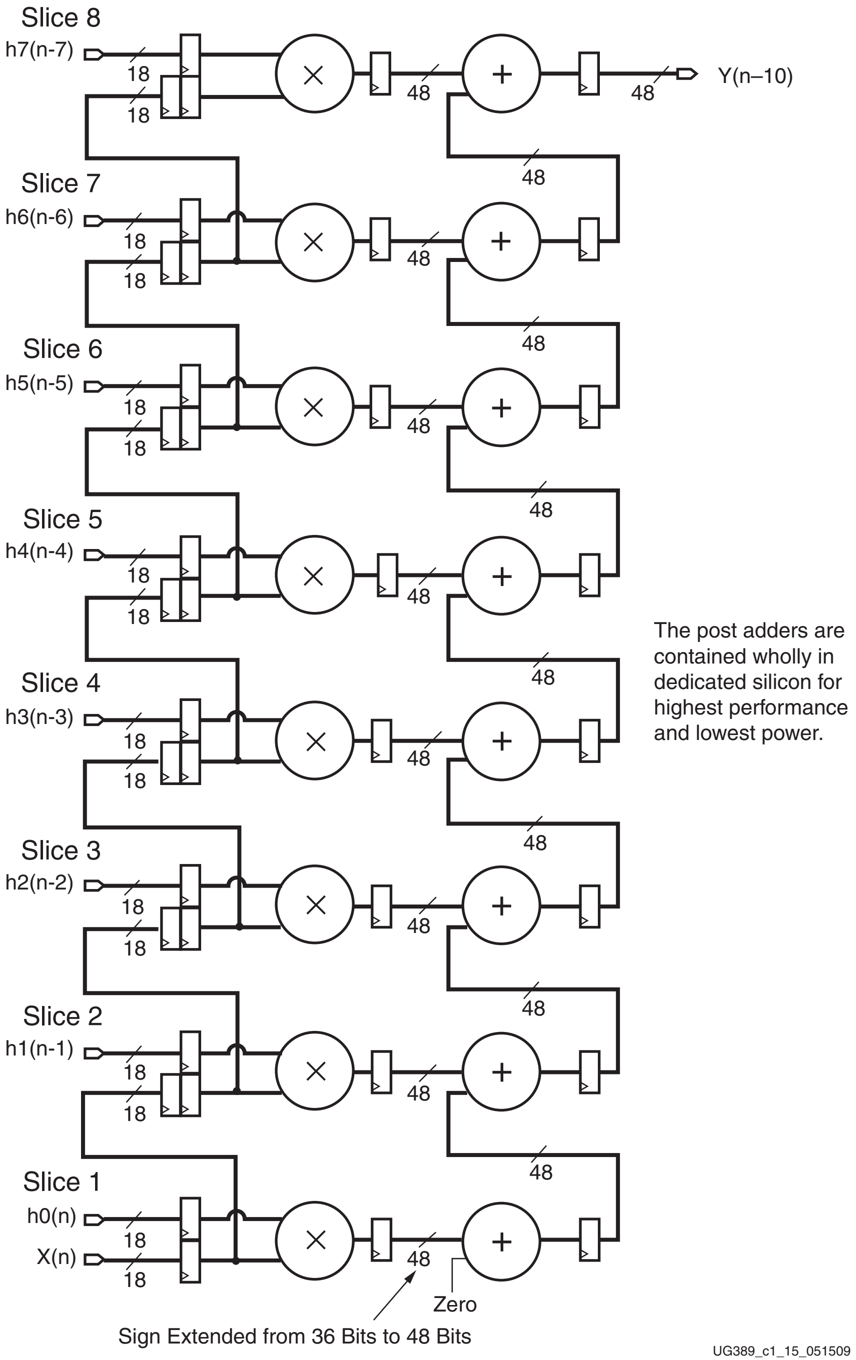


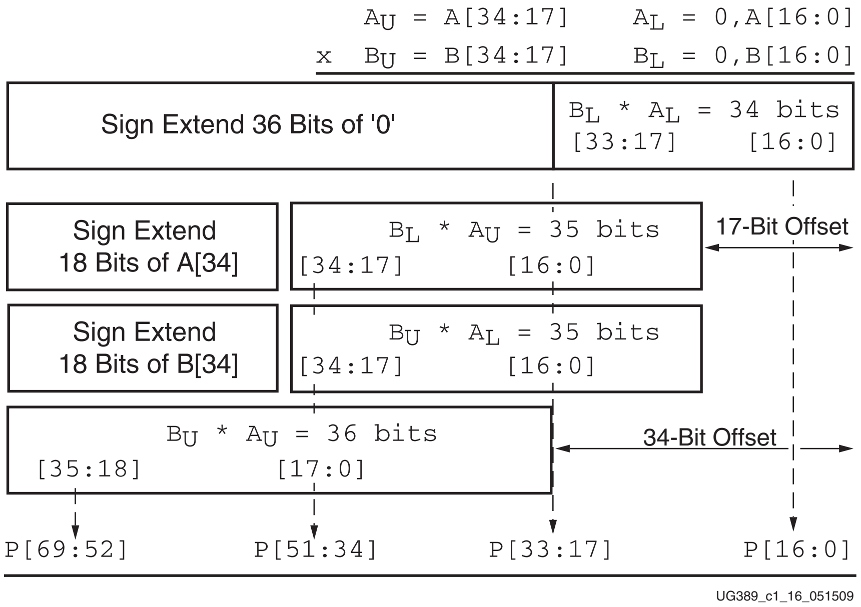
Figure 1-15: **Systolic FIR with Adder Cascade**

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).

DSP48A1 Slice Functional Use Models

Fully Pipelined, 35 x 35 Multiplier Use Model (Large Multiplier)

Figure 1-16 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.



UG389_c1_16_051509

Figure 1-16: **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 DSP48A1 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 1-16. 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 DSP48A1 slice, resulting in maximum performance with minimal power consumption.

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 1-17.

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 registers.

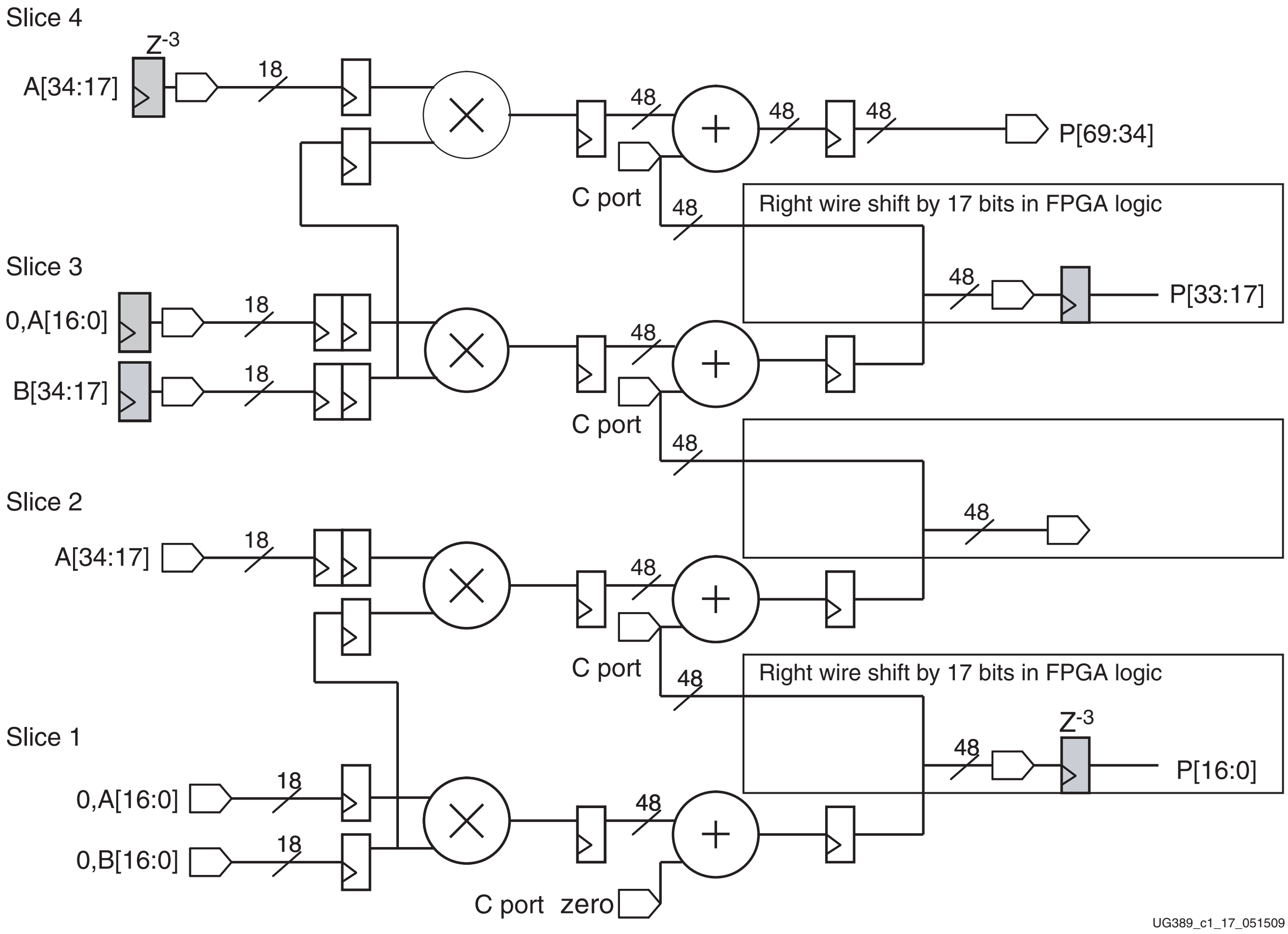


Figure 1-17: **Pipelined Version of 35x35 Multiplier Built from Cascaded DSP48A1 Slices**

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 1-17, 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.

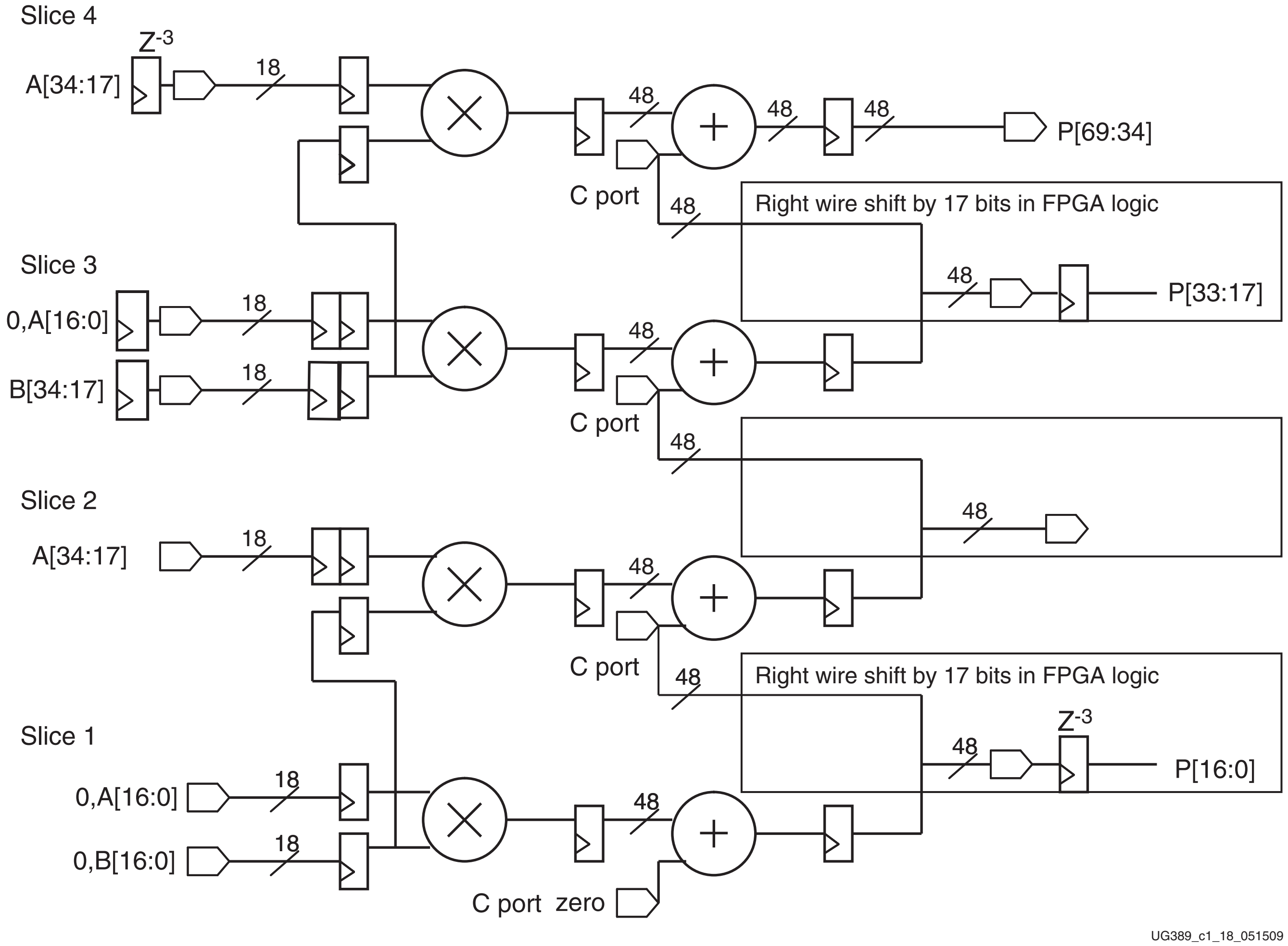


Figure 1-18: **Non-Pipelined Version of 35 x 35 Multiplier Built from Cascaded DSP48A1 Slices**

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 x B_real) – (A_imaginary x B_imaginary) = P_real *Equation 1-6*

(A_real x B_imaginary) + (A_imaginary x B_real) = P_imaginary *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 1-19 shows several DSP48A1 slices used as a complex, 18-bit x 18-bit multiplier.

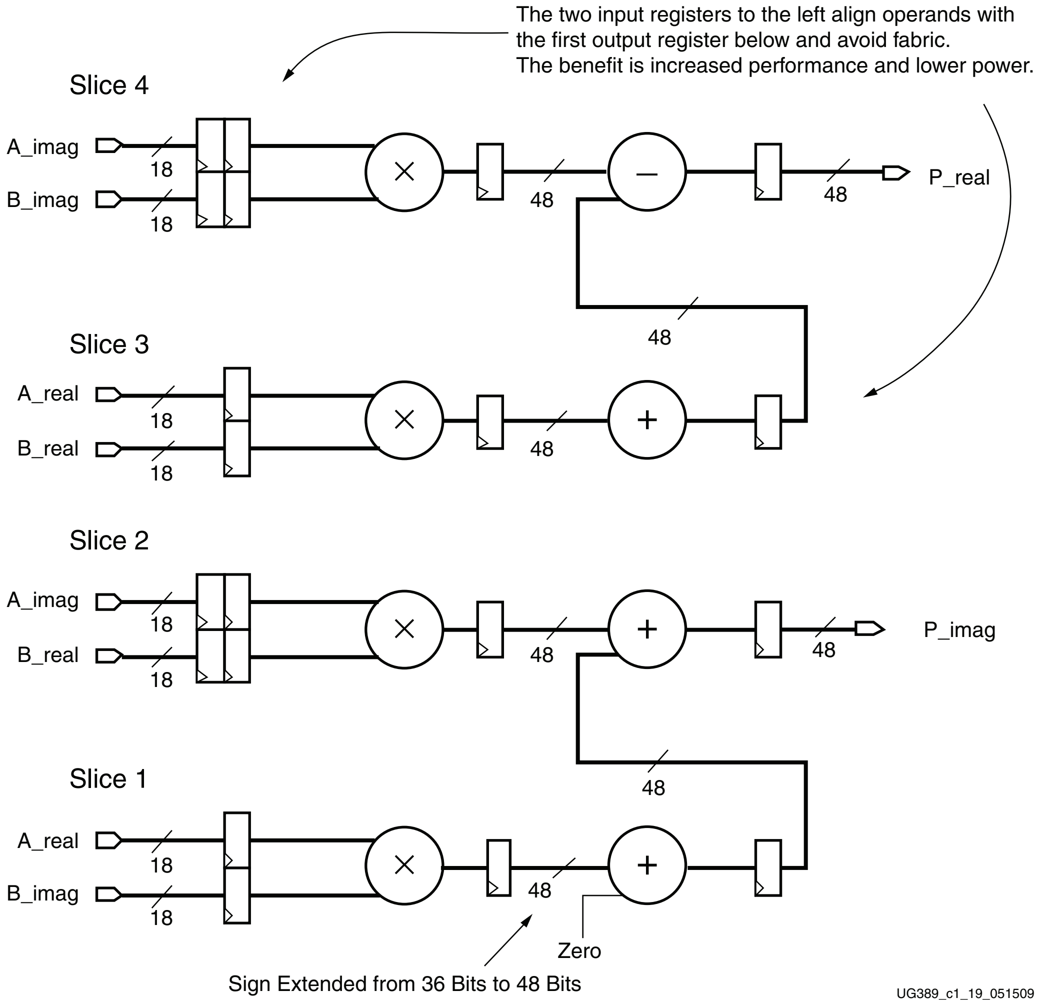

The two input registers to the left align operands with the first output register below and avoid fabric.
The benefit is increased performance and lower power.
Slice 4
A_imag
B_imag
18
18
48
48
P_real
48
Slice 3
A_real
B_real
18
18
48
Slice 2
A_imag
B_real
18
18
48
48
P_imag
48
Slice 1
A_real
B_imag
18
18
48
Zero
Sign Extended from 36 Bits to 48 Bits
UG389_c1_19_051509

Figure 1-19: **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 DSP48A1 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 1-20 and Figure 1-21, use the complex multiply implementation shown in Figure 1-22.

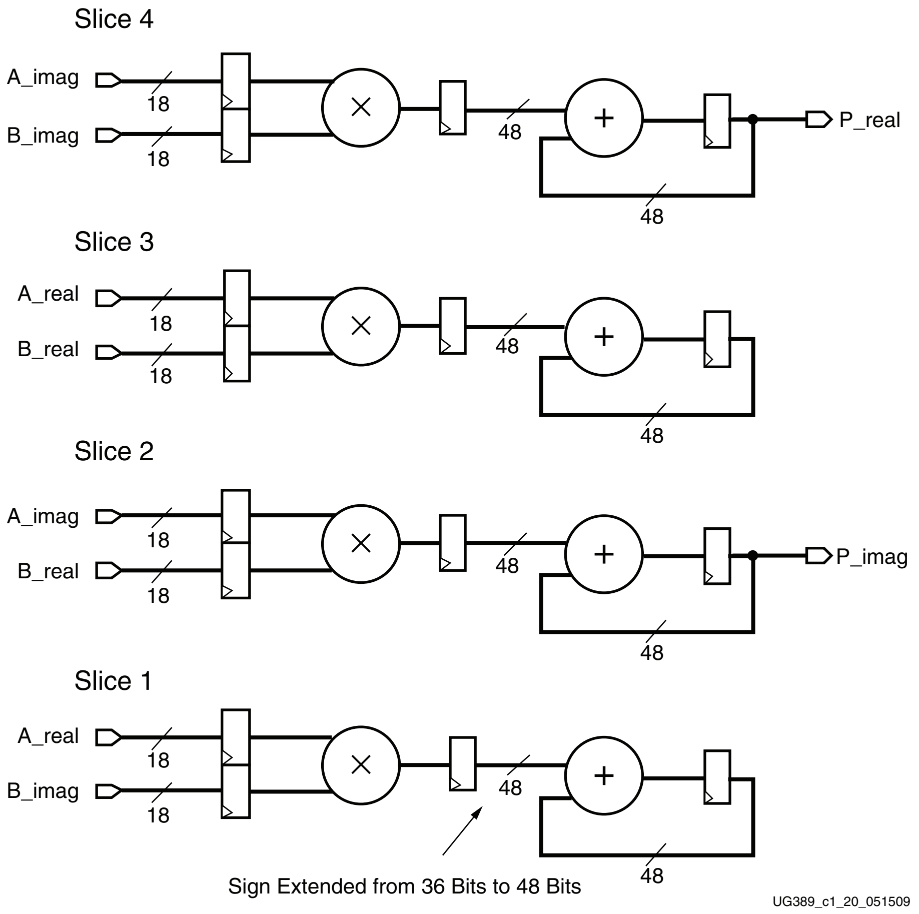


Figure 1-20: **Fully Pipelined, Complex, 18 x 18 MACC (N Cycles)**

In Figure 1-21, the N+1 cycle adds the accumulated products, and the input data stalls one cycle.

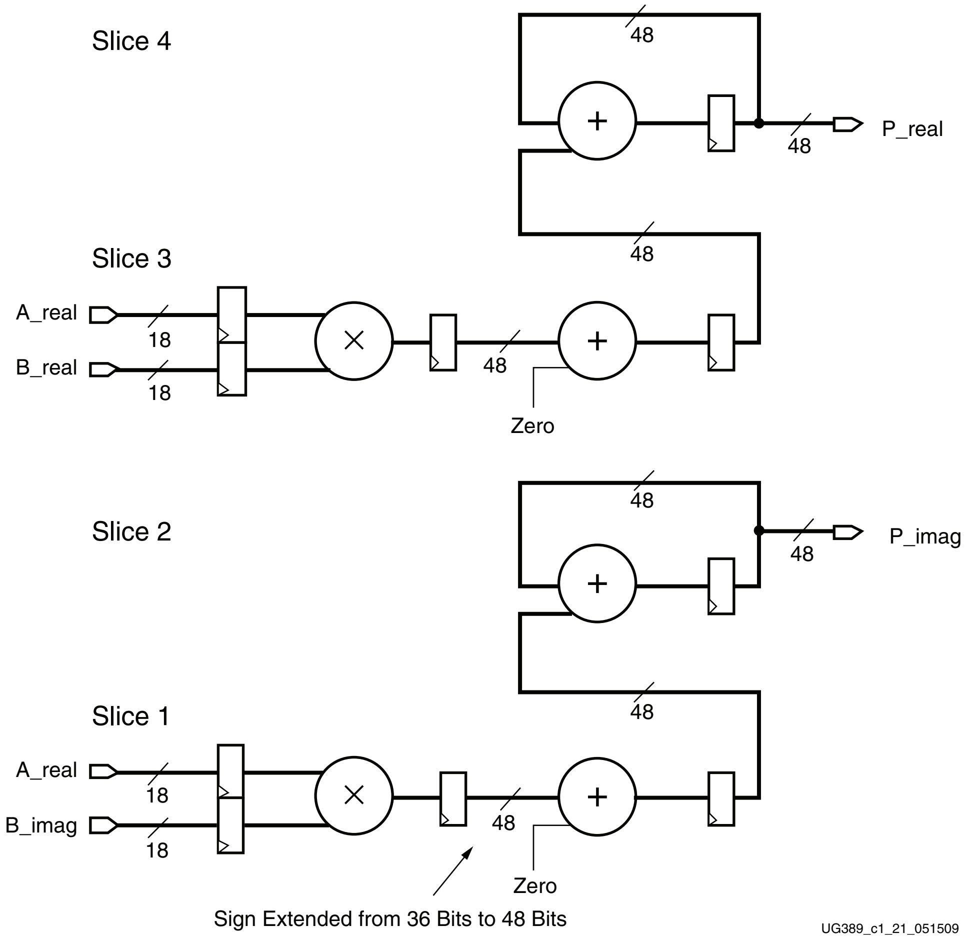


Figure 1-21: **Fully Pipelined, Complex, 18 x 18 MACC (Last or N + 1 Cycle)**

An additional slice used for the accumulation is shown in Figure 1-22. 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.

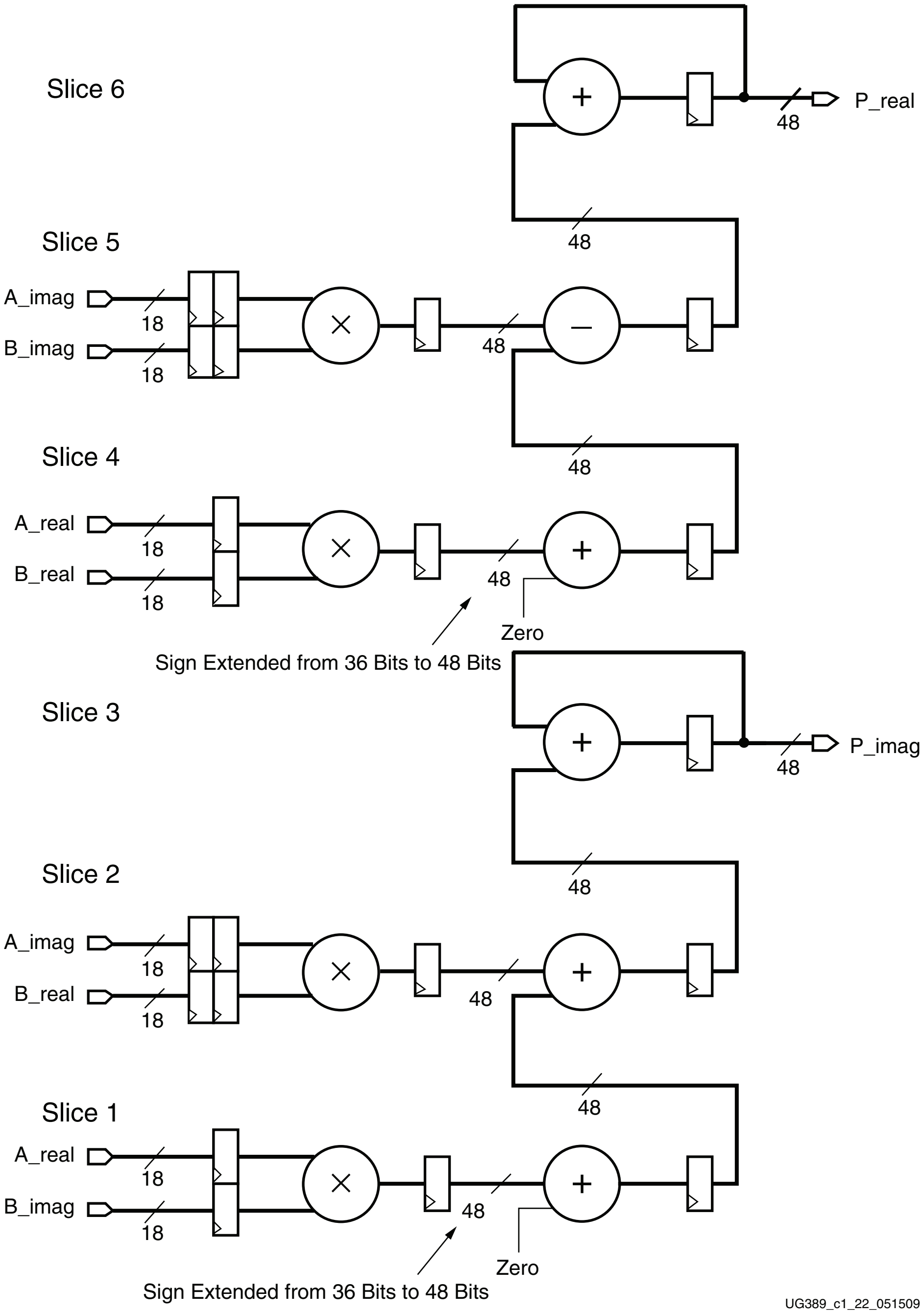


Figure 1-22: **Fully Pipelined, Complex, 18 x 18 MACC with Extra Slice**

Miscellaneous Functional Use Models

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

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

Use Model	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 DSP48A1 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. 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 1-23 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.

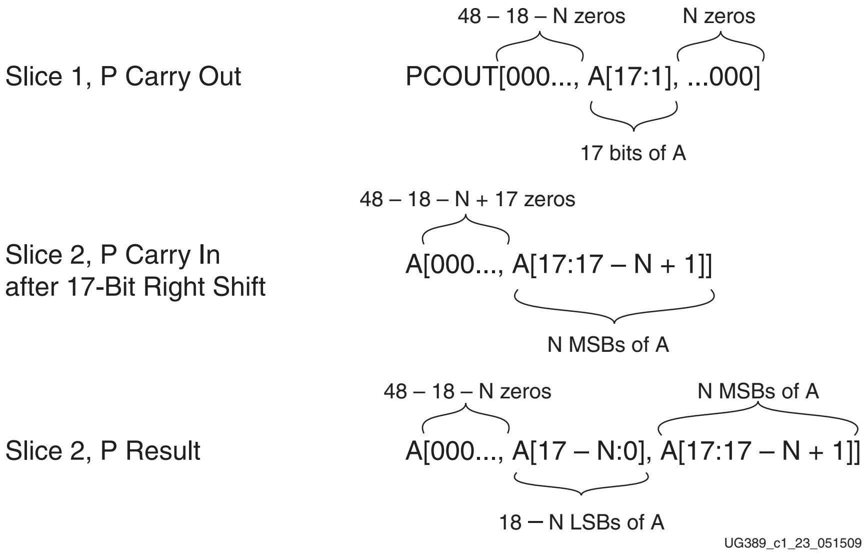


Figure 1-23: **Circular Barrel Shifter Equations**

Figure 1-24 shows the DSP48A1 slice used as 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.

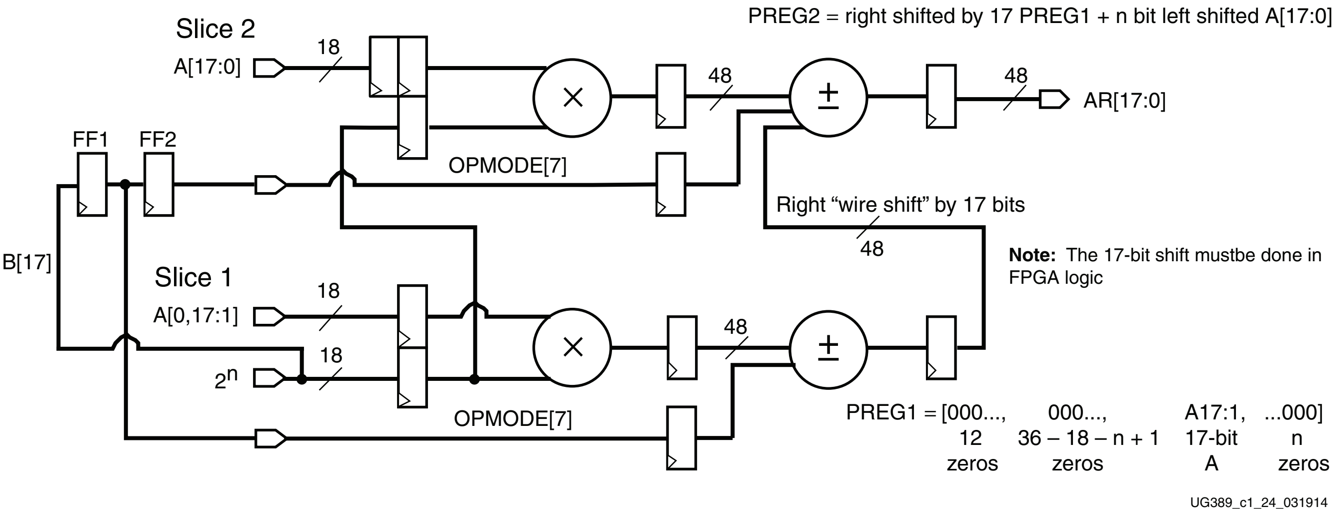

PREG2 = right shifted by 17 PREG1 + n bit left shifted A[17:0]
Slice 2
A[17:0]
18
48
48
AR[17:0]
FF1
FF2
OPMODE[7]
Right "wire shift" by 17 bits
48
B[17]
Note: The 17-bit shift mustbe done in FPGA logic
Slice 1
A[0,17:1]
18
18
2^n
48
OPMODE[7]
PREG1 = [000..., 12 zeros
000..., 36 – 18 – n + 1 zeros
A17:1, 17-bit A
...000] n zeros
UG389_c1_24_031914

Figure 1-24: **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 DSP48A1 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 DSP48A1 subtract input (which is also configured with internal subtract pipeline FF).
2. Feed the output of FF1 to the input of another fabric flip-flop called FF2. The output of FF2 then feeds the upper DSP48A1 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 inverted, which in essence is multiplying the result by –1, which effectively converts the -2^{17} value to a positive 2^{17} shift parameter.

Magnitude Compare Use Model

A magnitude comparison can be implemented by doing a subtraction. General purpose logic can then be used to check for a positive or negative result by checking the sign bit.

Timing Parameters

Understanding the naming conventions for DSP48A1 slice timing parameters used in the Spartan-6 FPGA data sheet and in the Timing Analyzer can help with design debugging and optimization. The general form of a DSP48A1 timing parameter is a T subscripted with DSP followed by the type of parameter, with underscores separating the starting and ending points for the paths. The data sheet has four types of parameters:

- Setup (T_{DSPDCK})
- Hold (T_{DSPCKD})
- Clock-to-Out (T_{DSPCKO})
- Propagation Delay (T_{DSPDO})

For example, the setup and hold times from the A input to the A1 register (with the A0 register bypassed) are $T_{DSPDCK\_A\_A1REG}$ and $T_{DSPCKD\_A\_A1REG}$. Clock-to-out delays are named with the output and then the source register. For example, the clock-to-out from the P register to the P output is $T_{DSPCKO\_P\_PREG}$. Combinatorial propagation delays are named according to the input followed by the output. For example, the delay from the A input to the P output, which assumes all registers are bypassed, is $T_{DSPDO\_A\_P}$.

Note that some parameter names can be used for multiple paths through the DSP48A1 slice. The pre-adder/subtracter, multiplier, and post-adder/subtracter can each be bypassed by certain inputs, and the OPMODE inputs can affect different parts of the DSP48A1 slice. The parameter name is based on starting and ending points, but the actual path used can result in a wide range of values. Therefore the data sheet defines multiple values for some parameters, depending on the specific elements in the path.

Not all paths are documented in the data sheet, which provides only a representative delay for selected paths. Use the Timing Analyzer reports for more specific and detailed timing information for a given design implementation.

Chapter 2

Using the DSP48A1 Pre-Adder

Introduction

The Spartan®-6 FPGA DSP48A1 slice 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.

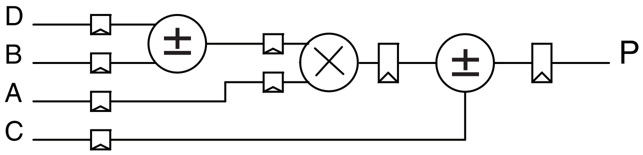


Figure 2-1: **DSP48A1 Slice with Pre-Adder**

Table 2-1: **DSP48A1 Slice with Pre-Adder**

Application	Pre-Adder	Density	F_{MAX}
N tap Symmetric FIR	No	1 FIR Tap/DSP48A1 Slice	See data sheet F_{MAX}
N tap Symmetric FIR	Yes	2 FIR Taps/DSP48A1 Slice	See data sheet F_{MAX}
Complex Multiplier	No	1 Complex Multiplier/4 DSP48A1 Slices	See data sheet F_{MAX}
Complex Multiplier	Yes	1 Complex Multiplier/3 DSP48A1 Slices	> 200 MHz[1]

Notes:

1. The listed frequency is based on -2 or faster speed grade devices.

The pre-adder enables the DSP48A1 slice to implement the equations in Table 2-2 as selected by the OPMODE.

Table 2-2: **DSP48A1 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 DSP48A1 slice.

Figure 2-2 shows a simplified diagram of a non-symmetric FIR filter architecture that uses four pipelined DSP48A1 slices, in addition to 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 as dotted are implemented in a single DSP48A1 slice.

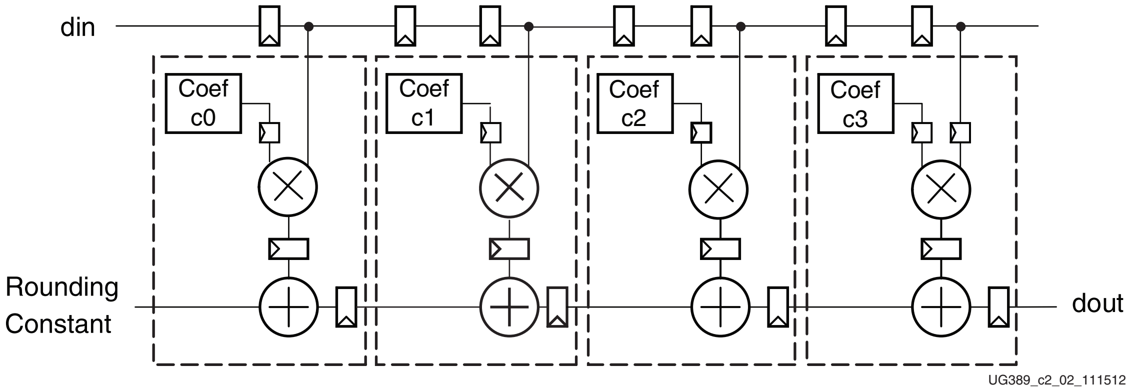


Figure 2-2: **Non-Symmetric FIR Filter Architecture (Simplified Diagram)**

If the number of taps (N) in the filter exceeds the number of multipliers (M) in the filter implementation, an 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 coefficient memories are used to store data and coefficient values. Data is written or copied from upstream memories to downstream memories at the data sample rate, once every N/M clocks. Data and coefficients are read from the 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. 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 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;
```

The two FIR filters shown in Figure 2-3 and Figure 2-4 are transposed convolution filters that eliminate registers in the reverse cascade path.

Figure 2-3 illustrates the transposed convolution, even tap Symmetric FIR filter architecture.

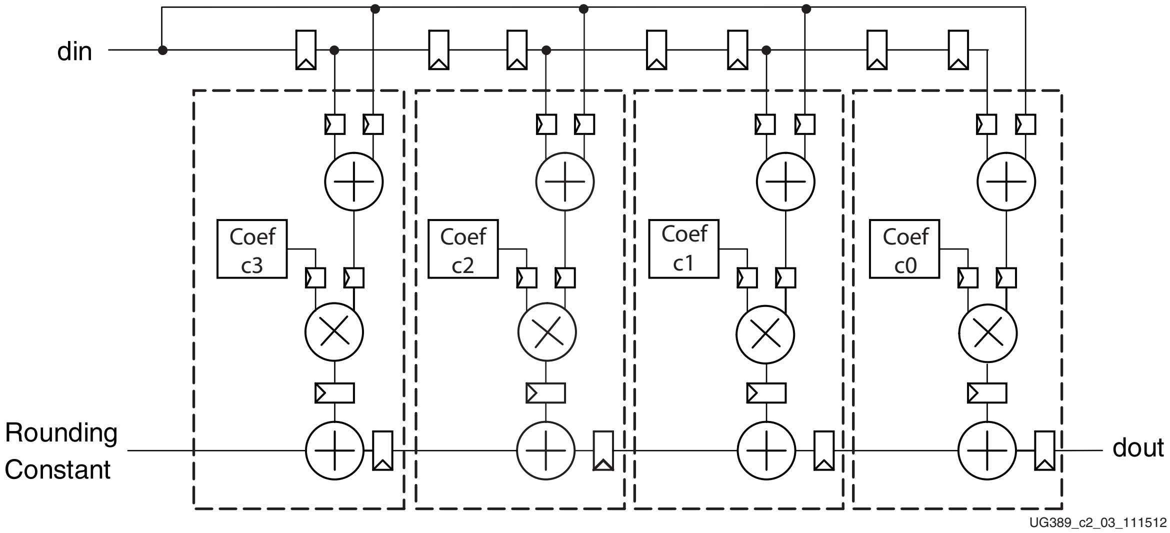


Figure 2-3: **Transposed Convolution, Even Tap Symmetric FIR Filter Architecture**

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

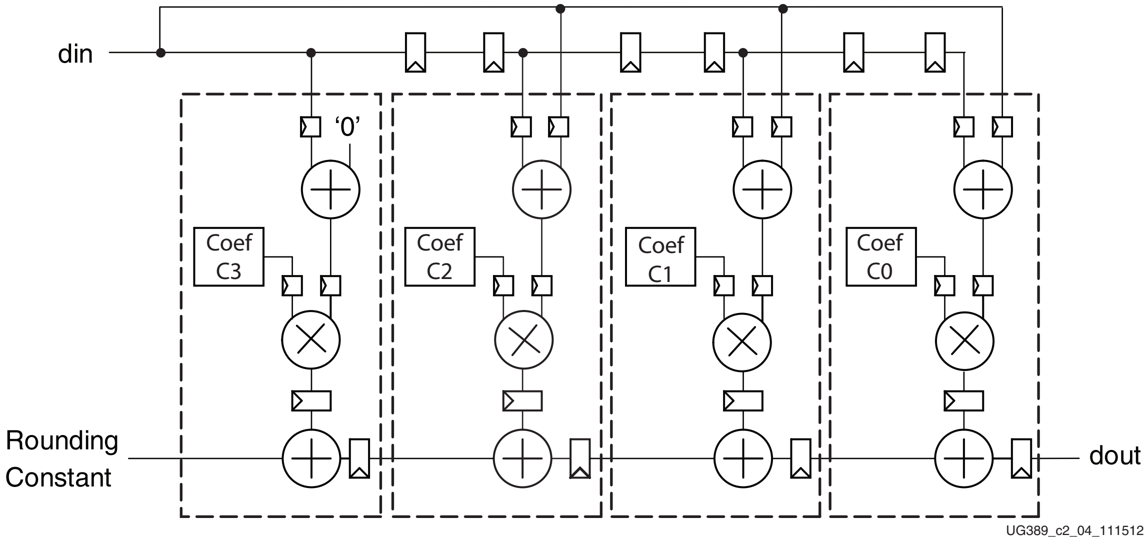


Figure 2-4: **Transposed Convolution, Odd Tap Symmetric FIR Filter Architecture**

Transposed convolution filters are fully supported in the ISE® tool, version 11.2 and later.

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 that uses 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 bold type is common to both multipliers and can be implemented with a single multiplier.

The four-multiplier complex version is shown in Figure 2-5.

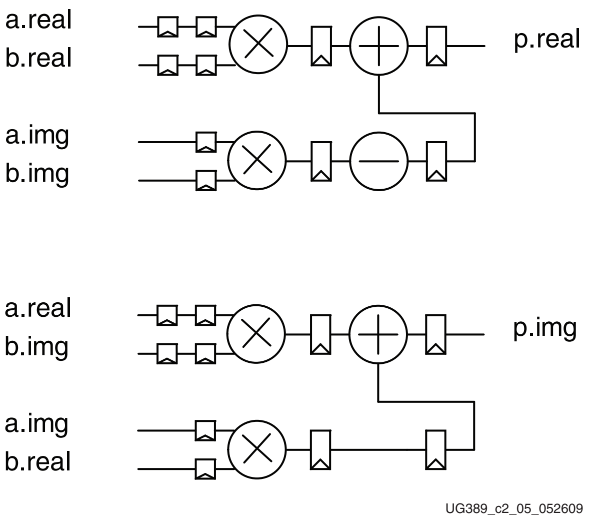


Figure 2-5: **Four-Multiplier Complex Version**

The three-multiplier complex version is shown in Figure 2-6.

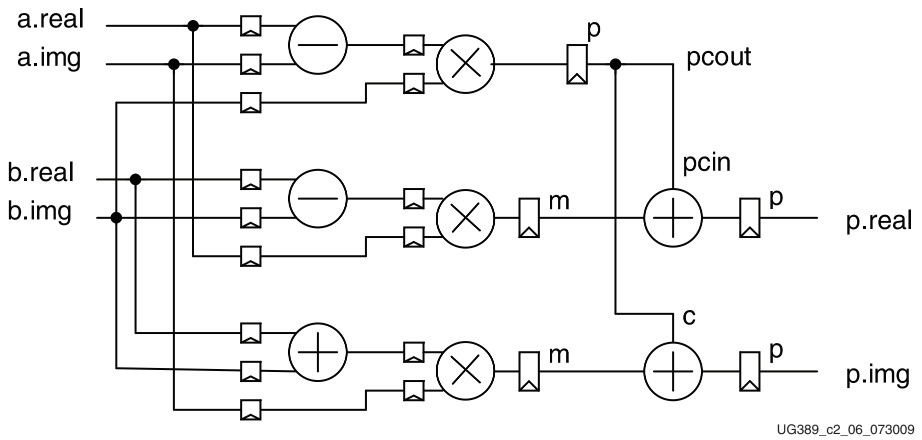


Figure 2-6: **Three-Multiplier Complex Version**

This implementation suffers from the fact that the shared term generated by the upper DSP48A1 slice is required to be routed to the C port of the lower DSP48A1 slice rather than using the PCIN port. Also, the MREG is not used in the upper DSP48A1 slice. The lack of pipelining reduces the speed of the three multiplier complex version using the DSP48A1 slice from the F_{MAX} value in the Spartan-6 FPGA data sheet to a frequency greater than 200 MHz. If this performance is not acceptable, a fully pipelined version can be implemented as in Figure 2-7.

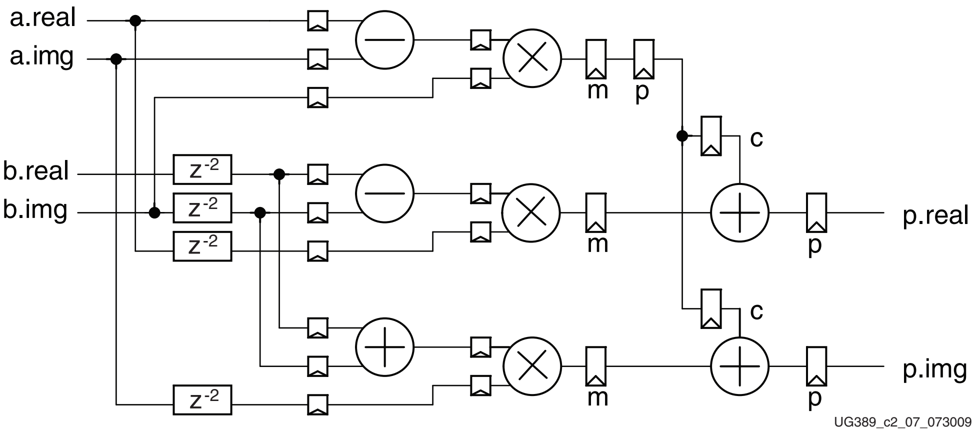


Figure 2-7: **Fully Pipelined Version**

The input setup time to the pre-adder without using input registers is sufficiently low to meet the data sheet F_{MAX} value without using input registers, which gives some flexibility in pipelining complex circuits that use the pre-adder.

Appendix A

Additional Resources

Additional Documentation

The following documents are also available for download at http://www.xilinx.com/products/spartan6.

- Spartan®-6 Family Overview

  This overview outlines the features and product selection of the Spartan-6 family.
- Spartan-6 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Spartan-6 family.
- Spartan-6 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.
- Spartan-6 FPGA Configuration User Guide

  This all-encompassing configuration guide includes chapters on configuration interfaces (serial and parallel), multi-bitstream management, bitstream encryption, boundary-scan and JTAG configuration, and reconfiguration techniques.
- Spartan-6 FPGA SelectIO Resources User Guide

  This guide describes the SelectIO™ resources available in all Spartan-6 devices.
- Spartan-6 FPGA Clocking Resources User Guide

  This guide describes the clocking resources available in all Spartan-6 devices, including the DCMs and the PLLs.
- Spartan-6 FPGA Block RAM Resources User Guide

  This guide describes the Spartan-6 device block RAM capabilities.
- Spartan-6 FPGA Configurable Logic Block User Guide

  This guide describes the capabilities of the configurable logic blocks (CLBs) available in all Spartan-6 devices.
- Spartan-6 FPGA Memory Controller User Guide

  This guide describes the Spartan-6 FPGA memory controller block, a dedicated embedded multi-port memory controller that greatly simplifies interfacing Spartan-6 FPGAs to the most popular memory standards.
- Spartan-6 FPGA GTP Transceivers User Guide

  This guide describes the GTP transceivers available in Spartan-6 LXT FPGAs.

- Spartan-6 FPGA PCB and Pin Planning Design Guide

  This guide provides information on PCB design for Spartan-6 devices, with a focus on strategies for making design decisions at the PCB and interface level.

- Spartan-6 FPGA Power Management User Guide

  This guide provides information on the various hardware methods of power management in Spartan-6 devices, primarily focusing on the suspend mode.

Additional Documentation Resources

The following documents provide useful supplemental material to this guide:

1. UG431, *XtremeDSP DSP48A for Spartan-3A DSP FPGAs User Guide*
2. UG073, *XtremeDSP for Virtex®-4 FPGAs*
3. UG687, *XST User Guide for Virtex-6 and Spartan-6 Devices*

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

http://www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.