UltraScale Architecture DSP Slice

User Guide

UG579 (v1.10) September 22, 2020

XILINX

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/22/2020	1.10	Added VU23P and VU57P to Table 1-2. Updated XOR input bits for XOR24_48_96 in Table 2-11. Added delay element to DSP slices in Figure 4-6.
09/20/2019	1.9	Added VU19P, VU45P, and VU47P to Table 1-2.
05/14/2019	1.8	In Device Resources, updated Tcl command and added note. In Table 1-2, updated total column for VU11P and VU13P, and added VU27P, VU29P, VU31P, VU33P, VU35P, and VU37P devices.
06/04/2018	1.7	Added description of ALUMODE after Figure 5-3, and added Table 5-1. Updated ALUMODE settings in Adder/Subtracter-only Operation.
04/05/2018	1.6	In Figure 2-2, connected upper input of INMODE[4]-controlled multiplexer in B input path to configured output selection after B2 stage.
10/18/2017	1.5	Added output of P/C multiplexer in Figure 1-1. Added sentence about cascading across clock regions in paragraph after Figure 1-2. Updated X multiplexer inputs in list after Equation 2-1.
06/01/2017	1.4	Updated link to the UltraScale architecture documentation suite in last paragraph of Introduction to UltraScale Architecture, page 6. Removed duplication of the word term in the bulleted list item starting with *Pattern detector:*. Revised last paragraph of Differences from Previous Generations, page 8. Updated Table 1-2 by changing total for KU3P and removing row containing KU7P. Updated second bullet after Equation 2-1. Updated pre-adder/multiplier function column in Table 2-1. Updated multiplier A and B port columns in Table 2-2. Revised first sentence under Embedded Functions, page 36 by adding the embedded function pre-adder. Added new paragraph at the end of Overflow and Underflow Logic, page 42. Added IS_RSTINMODE_INVERTED, IS_RSTM_INVERTED, and IS_RSTP_INVERTED to Table 3-3. All figures have been replaced in this version.
11/24/2015	1.3	Under Introduction to UltraScale Architecture, page 6, added new introductory text for UltraScale+ devices. Under Device Resources, page 10, added new first paragraph, original first paragraph becomes second paragraph, original second paragraph is deleted, and new third paragraph is added. Updated Figure 1-2. Added Table 1-1 and Table 1-2. Under MULTSIGNOUT and CARRYCASCOUT, page 71, revised CARRYINSEL to CARRYINSELREG in fifth paragraph. Reorganized and updated References, page 75, and added UltraScale+ device references.
01/12/2015	1.2	Removed Table 1-2 and added reference to *UltraScale Architecture and Product Overview* (DS890) on page 9. Changed INMODE[3] value from 0 to 0/1 in third row of Table 2-2. Added reference to *Vivado Design Suite Reference Guide: Model-Based DSP Design Using System Generator* (UG958) on page 49. Added reference to Vivado High-Level Synthesis webpage on page 49 and in Appendix A. Added reference to UltraScale Architecture Libraries Guide (UG974) on page 50. Added reference to UltraScale device data sheets on page 59. Added reference to Vivado High-Level Synthesis, DSP Solution, Vivado Video Tutorials, and Xilinx DSP Training web pages in Appendix A.

Date	Version	Revision
07/15/2014	1.1	Deleted section Differences in Devices Using SSI Technology on page 8. Added Table 1-2. Added multiplexer INMODE[0] values used to select each input in Figure 2-5. Added multiplexer INMODE[4] values used to select each input in Figure 2-6. Added note 3 to Table 2-2. Added DSP48E2 Operation Modes in Chapter 2. Revised description for CEA1, CEA2, CEB1, CEB2, and INMODE in Table 3-2. Revised description for AREG, and BREG in Table 3-3. Added [Ref 7] and [Ref 8] to References.
12/10/2013	1.0	Initial Xilinx release.

Table of Contents

Appendix A: Additional Resources and Legal Notices

Overview

Introduction to UltraScale Architecture

The Xilinx® UltraScale™ architecture is the first ASIC-class All Programmable architecture to enable multi-hundred gigabit-per-second levels of system performance with smart processing, while efficiently routing and processing data on-chip. UltraScale architecture-based devices address a vast spectrum of high-bandwidth, high-utilization system requirements by using industry-leading technical innovations, including next-generation routing, ASIC-like clocking, 3D-on-3D ICs, multiprocessor SoC (MPSoC) technologies, and new power reduction features. The devices share many building blocks, providing scalability across process nodes and product families to leverage system-level investment across platforms.

Virtex® UltraScale+™ devices provide the highest performance and integration capabilities in a FinFET node, including both the highest serial I/O and signal processing bandwidth, as well as the highest on-chip memory density. As the industry's most capable FPGA family, the Virtex UltraScale+ devices are ideal for applications including 1+Tb/s networking and data center and fully integrated radar/early-warning systems.

Virtex UltraScale devices provide the greatest performance and integration at 20 nm, including serial I/O bandwidth and logic capacity. As the industry's only high-end FPGA at the 20 nm process node, this family is ideal for applications including 400G networking, large scale ASIC prototyping, and emulation.

Kintex® UltraScale+ devices provide the best price/performance/watt balance in a FinFET node, delivering the most cost-effective solution for high-end capabilities, including transceiver and memory interface line rates as well as 100G connectivity cores. Our newest mid-range family is ideal for both packet processing and DSP-intensive functions and is well suited for applications including wireless MIMO technology, Nx100G networking, and data center.

Kintex UltraScale devices provide the best price/performance/watt at 20 nm and include the highest signal processing bandwidth in a mid-range device, next-generation transceivers, and low-cost packaging for an optimum blend of capability and cost-effectiveness. The family is ideal for packet processing in 100G networking and data centers applications as well as DSP-intensive processing needed in next-generation medical imaging, 8k4k video, and heterogeneous wireless infrastructure.

Zynq® UltraScale+ MPSoC devices provide 64-bit processor scalability while combining real-time control with soft and hard engines for graphics, video, waveform, and packet processing. Integrating an Arm®-based system for advanced analytics and on-chip programmable logic for task acceleration creates unlimited possibilities for applications including 5G Wireless, next generation ADAS, and Industrial Internet-of-Things.

This user guide describes the UltraScale architecture DSP Slice resources and is part of the UltraScale architecture documentation suite available at: www.xilinx.com/documentation.

UltraScale Architecture DSP Slice Overview

Programmable logic devices are efficient for digital signal processing (DSP) applications because they can implement custom, fully parallel algorithms. DSP applications use many binary multipliers and accumulators that are best implemented in dedicated DSP resources. The UltraScale devices have many dedicated low-power DSP slices, combining high speed with small size while retaining system design flexibility. The DSP resources enhance the speed and efficiency of many applications beyond digital signal processing, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O registers. The DSP slice in the UltraScale architecture is defined using the DSP48E2 primitive and the slice is referred to as either DSP or DSP48E2 in the Xilinx tools. The basic functionality of the DSP48E2 slice is shown in Figure 1-1. For complete details, refer to Chapter 2, DSP48E2 Functionality.

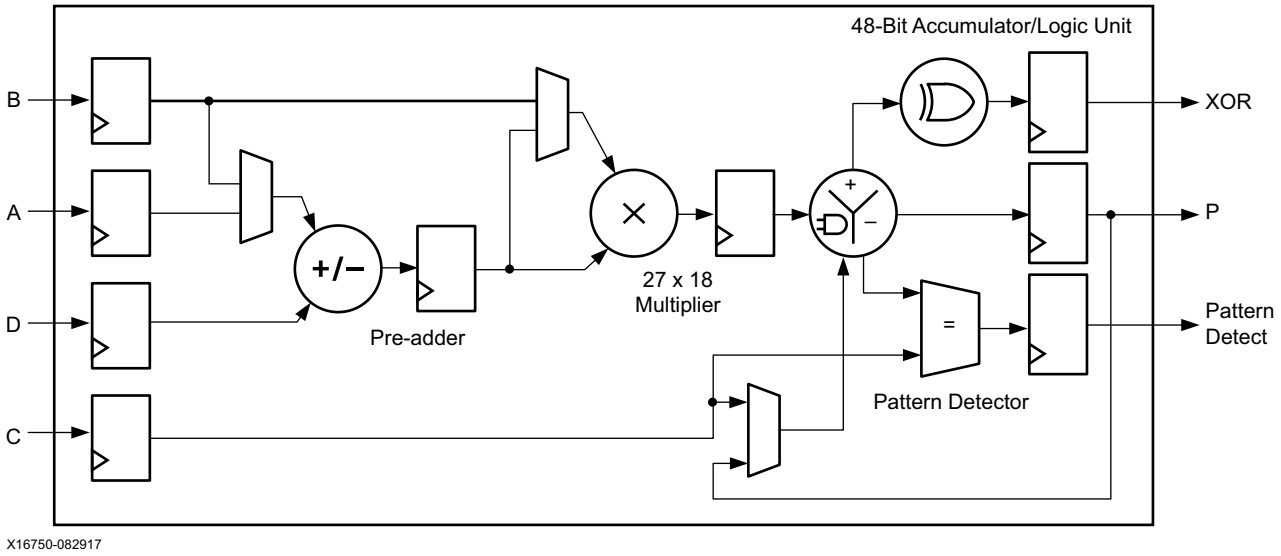


Figure 1-1: **Basic DSP48E2 Functionality**

Some highlights of the DSP slice functionality include:

- 27 × 18 two's complement multiplier with dynamic bypass
- Power saving 27-bit pre-adder: optimizes symmetrical filter applications and reduces DSP logic requirements
- 48-bit accumulator that can be cascaded to build 96-bit and larger accumulators, adders, and counters
- Single-instruction-multiple-data (SIMD) arithmetic unit: dual 24-bit or quad 12-bit add/subtract/accumulate
- 48-bit logic unit: bitwise AND, OR, NOT, NAND, NOR, XOR, and XNOR
- Pattern detector: terminal counts, overflow/underflow, convergent/symmetric rounding support, and 96-bit wide AND/NOR when combined with logic unit
- Optional pipeline registers and dedicated buses for cascading multiple DSP slices in a column for hierarchical/composite functions like Systolic FIR filters

The DSP48E2 slice supports both sequential and cascaded operations due to the dynamic OPMODE and cascade capabilities. Applications of the DSP slice include:

- Fixed and floating point Fast Fourier Transform (FFT) functions
- Systolic FIR filters
- MultiRate FIR filters
- CIC filters
- Wide real/complex multipliers/accumulators

Differences from Previous Generations

The UltraScale architecture DSP48E2 slice is backwards compatible with the 7 series FPGA DSP48E1 slice. The DSP48E2 slice is effectively a superset of the DSP48E1 slice with these differences:

- Wider functionality in DSP48E2 than DSP48E1 slice:
  - Multiplier width is improved from 25 x 18 in the DSP48E1 to 27 x 18 in the DSP48E2
  - Pre-adder increased from 25 bits to 27 bits:
    - D input and register to pre-adder increased to 27 bits
    - AD register result from pre-adder increased to 27 bits
- More flexibility in pre-adder:
  - A or B can be selected as input to the pre-adder

  - Output of the pre-adder can be squared
- Added fourth operand to ALU with WMUX:
  - Support adding two other input operands with the multiplier's two partial products, instead of only one in DSP48E1
  - Enable four-operand add in second stage
  - Add a memory-cell based rounding constant while freeing the C input for the following function: A x B + C + RND
  - WMUX provides another accumulator feedback path to reduce the size of the complex multiply-accumulate (MACC) or a semi-parallel FIR filter.
- Wide XOR of X, Y, Z multiplexers
  - 48 3-bit XOR at first level feeds XOR tree to create octal 12-bit XOR, quad 24-bit XOR, dual 48-bit XOR, single 96-bit XOR
  - Cascading two DSP48E2 slices in Wide XOR mode creates octal 24-bit XOR, quad 48-bit XOR, dual 96-bit XOR, or single 192-bit XOR. Cascade depth is limited to DSP column size
  - Sequentially create wider XOR via XOR accumulation feedback with a single DSP48E2, extending the XOR width by 96 bits every clock cycle
- Unique features in DSP48E2:
  - Programmable inversion added to reset inputs for flexibility
  - Clock enable can have priority over autoreset of counter/accumulator in P register

The DSP48E2 blocks use a signed arithmetic implementation. To best match the resource capabilities and, in general, to get the most efficient mapping, write code using signed values in the HDL source. Designs created for the 25 x 18 multiplier in the 7 series FPGAs may need to be sign-extended for the 27 x 18 multiplier in the UltraScale architecture. For more details on migration and design methodologies, see *UltraScale Architecture Migration Methodology Guide* (UG1026) [Ref 1]. When migrating designs with many cascaded DSP slices, the number of DSP slices per column in the new target device should be taken into consideration.

Device Resources

The DSP resources are optimized and scalable across the UltraScale portfolio, providing a common architecture that improves implementation efficiency, IP implementation, and design migration. Migration between UltraScale families does not require any design changes for the DSP48E2 slice.

Two DSP48E2 slices with a dedicated interconnect form each DSP tile (see Figure 1-2). The DSP tiles stack vertically in a DSP48E2 column. The height of a DSP tile is the same as five configurable logic blocks (CLBs) and also matches the height of one 36K block RAM. The block RAM can be split into two 18K block RAMs. Each DSP48E2 slice aligns horizontally with an 18K block RAM, providing optimal connectivity between resources.

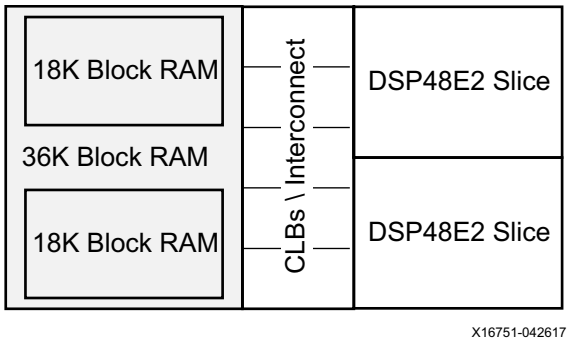


Figure 1-2: **DSP Tile**

The DSP48E2 column is 12 tiles tall per clock region, therefore providing 24 DSP48E2 slices per column per clock region. The number of clock regions per column can be found in the Vivado Device view, in the UltraScale Architecture and Product Overview (DS890) [Ref 2], or in the bank diagrams in *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) [Ref 3]. DSP48E2 slices can be cascaded across clock regions up to the boundary of the device or of a super logic region (SLR) in 3D ICs based on SSI technology. In the UltraScale+ low-voltage devices (V_{CCINT} = 0.72V), cascading across a clock region might impact performance. The number of cascadeable DSP48E2 slices in a column can be found with the Tcl command:

```
llength [get_sites DSP48E2_X3Y* -of_objects [get_slrs SLR0]]
```

Note: The maximum might be less in different DSP columns (if PS limits the DSP cascade height) or SLRs (SLR with HBM interface vs. no HBM interface).

Table 1-1 shows the maximum number of DSP48E2 slices that can be directly cascaded vertically in a column, and the total number of DSP48E2 slices, for the UltraScale FPGAs.

Table 1-1: **Maximum Number of Cascadable DSP Slices in UltraScale FPGAs**

	Max Cascade	Total
Kintex UltraScale		
KU025	72	1,152
KU035	120	1,700
KU040	120	1,920
KU060	120	2,760
KU085	120(1)	4,100
KU095	192	768
KU115	120	5,520
Virtex UltraScale		
VU065	120	600
VU080	192	672
VU095	192	768
VU125	120	1,200
VU160	120(2)	1,560
VU190	120	1,800
VU440	120	2,880

Notes:
1. KU085 max cascade is 96 in SLR1.
2. VU160 max cascade is 96 in SLR0.

Table 1-2 shows the same information for the UltraScale+ FPGAs.

Table 1-2: **Maximum Number of Cascadable DSP Slices in UltraScale+ FPGAs**

	Max Cascade	Total
Kintex UltraScale+		
KU3P	96	1,368
KU5P	96	1,824
KU9P	168	2,520
KU11P	192	2,928
KU13P	168	3,528
KU15P	264	1,968
Virtex UltraScale+		
VU3P	120	2,280
VU5P	120	3,474

Table 1-2: **Maximum Number of Cascadable DSP Slices in UltraScale+ FPGAs**

	Max Cascade	Total
VU7P	120	4,560
VU9P	120	6,840
VU11P	96	9,216
VU13P	96	12,288
VU19P	120	3,840
VU23P	264	1,320
VU27P	96	9,216
VU29P	96	12,288
VU31P	90	2,880
VU33P	90	2,880
VU35P	96	5,952
VU37P	96	9,024
VU45P	96	5,952
VU47P	96	9,024
VU57P	96	9,024

Recommended Design Flow

Many DSP designs are well suited for UltraScale architecture-based devices. To obtain best use of the architecture, the underlying features and capabilities need to be understood so that design entry code can take advantage of these resources. DSP48E2 resources are used automatically for most DSP functions and many arithmetic functions. In most cases, DSP resources should be inferred. See your preferred synthesis tool documentation for guidelines to ensure proper inference of the DSP48E2 slice. Instantiation of the DSP48E2 primitive can be used to directly access specific features. Recommendations for using DSP48E2 slices include:

- Use signed values in HDL source
- Pipeline for performance and lower power, both in the DSP48E2 slice and in programmable logic
- Use configurable logic block (CLB) shift register LUTs (SRLs), CLB distributed RAM, and/or block RAM to store filter coefficients
- Set USE_MULT to NONE when using only the adder/logic unit to save power
- Cascade using the dedicated resources rather than general-purpose interconnect, keeping usage to one column for highest performance and lowest power
- Consider using time multiplexing if resources are limited in a lower-speed application

- Use the CLB carry logic to implement small multipliers, adders, and counters

For more information on design techniques, see Chapter 4, DSP48E2 Usage Guidelines.

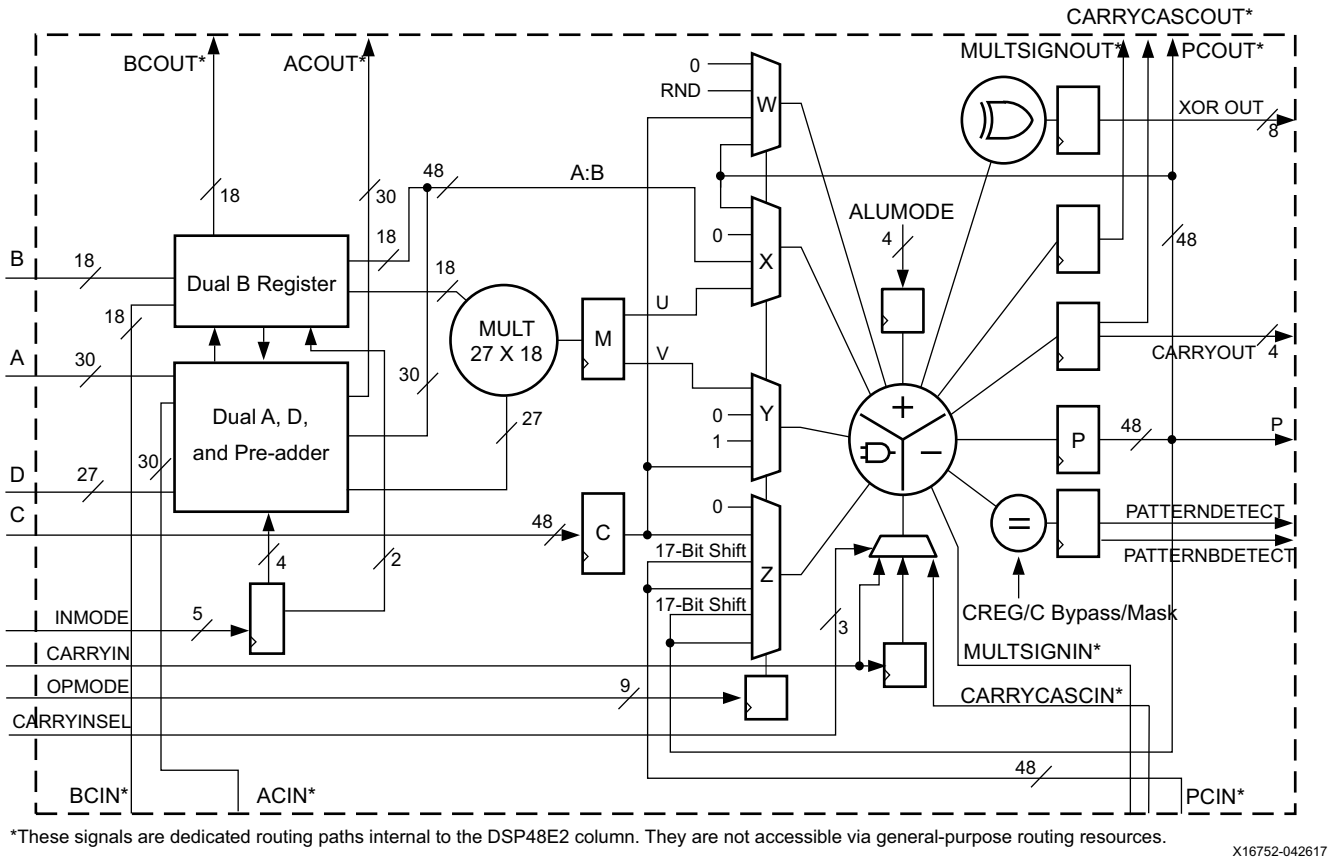
Pinout Planning

DSP usage has little effect on pinouts because DSP48E2 slices are distributed throughout the device. The best approach is to let the tools choose the DSP48E2 and I/O locations based on the implementation requirements. Results can be adjusted if necessary for board layout considerations. The timing constraints should be set so that the tools can choose optimal placement to meet the specific design requirements. The only directional consideration in the DSP structure is that DSP48E2 slices cascade vertically up a column, allowing wide buses to drive a vertical orientation to other logic, including I/O. The I/O columns typically provide 13 I/O in the same vertical space as every six DSP48E2 slices, with every clock region defined in height by a bank of 52 I/O and 12 DSP tiles (24 DSP slices).

DSP48E2 Functionality

Overview

This chapter provides technical details of the DSP48E2 element. The DSP48E2 slice consists of a 27-bit pre-adder, 27 x 18 multiplier and a flexible 48-bit ALU that serves as a post-adder/subtracter, accumulator, or logic unit (see Figure 2-1).



*These signals are dedicated routing paths internal to the DSP48E2 column. They are not accessible via general-purpose routing resources.

X16752-042617

Figure 2-1: **Detailed DSP48E2 Functionality**

The DSP48E2 supports many independent functions. These functions include:

- Multiply
- Multiply accumulate (MACC)
- Multiply add
- Four-input add
- Barrel shift
- Wide-bus multiplexing
- Magnitude comparator
- Bitwise logic functions
- Wide XOR
- Pattern detect
- Wide counter

The architecture also supports cascading multiple DSP48E2 slices to form wide math functions, DSP filters, and complex arithmetic without the use of general logic.

DSP48E2 Features

The features in the DSP48E2 slice are:

- 27-bit pre-adder with D register to enhance the capabilities of the A or B path
- A or B can be selected as pre-adder input to allow for wider multiplication coefficients
- The result of the pre-adder can be sent to both inputs of the multiplier to provide squaring capability
- INMODE control supports balanced pipelining when dynamically switching between multiply (A*B) and add operations (A+B)
- 27 x 18 multiplier
- 30-bit A input of which the lower 27 bits feed the A input of the multiplier, and the entire 30-bit input forms the upper 30 bits of the 48-bit A:B concatenated internal bus
- Cascading A and B input:
  - Semi-independently selectable pipelining between direct and cascade paths
  - Separate clock enables for two-deep A and B set of input registers
- Independent C input and C register with independent reset and clock enable

- CARRYCASCIN and CARRYCASCOUT internal cascade signals to support 96-bit accumulators/adders/subtracters in two DSP48E2 slices, and to support cascading more than two DSP slices
- MULTSIGNIN and MULTSIGNOUT internal cascade signals with special OPMODE setting to support a 96-bit MACC extension
- Single Instruction Multiple Data (SIMD) Mode for four-input adder/subtracter, which precludes use of multiplier in first stage:
  - Dual 24-bit SIMD adder/subtracter/accumulator with two separate CARRYOUT signals
  - Quad 12-bit SIMD adder/subtracter/accumulator with four separate CARRYOUT signals
- 48-bit logic unit:
  - Bitwise logic operations—two-input AND, OR, NOT, NAND, NOR, XOR, and XNOR
  - Logic unit mode dynamically selectable via ALUMODE
- 96-bit wide XOR logic selectable from eight 12-bit XORs to one 96-bit XOR
- Pattern detector:
  - Overflow/underflow support
  - Convergent rounding support
  - Terminal count detection support and auto resetting: auto resetting can give priority to clock enable
- Cascading 48-bit P bus supports internal low-power adder cascade: 48-bit P bus allows for 12-bit quad or 24-bit dual SIMD adder cascade support
- Optional 17-bit right shift to enable wider multiplier implementation
- Dynamic user-controlled operating modes:
  - 9-bit OPMODE control bus provides W, X, Y, and Z multiplexer select signals
  - 5-bit INMODE control bus provides selects for 2-deep A and B registers, pre-adder add-sub control as well as mask gates for pre-adder multiplexer functions
  - 4-bit ALUMODE control bus selects logic unit function and accumulator add-sub control
- Carry in for the second stage adder:
  - Support for rounding
  - Support for wider add/subtracts
  - 3-bit CARRYINSEL multiplexer
- Carry out for the second stage adder:

  - Support for wider add/subtracts
  - Available for each SIMD adder (up to four)
  - Cascaded CARRYCASCOUT and MULTSIGNOUT allows for MACC extensions up to 96 bits
- Single clock for synchronous operation
- Optional input, pipeline, and output/accumulate registers
- Optional registers for control signals (OPMODE, ALUMODE, and CARRYINSEL)
- Independent clock enable and synchronous resets with programmable polarity for greater flexibility
- Internal multiplier and XOR logic can be gated off when unused to save power

The DSP slice consists of a multiplier followed by an accumulator. At least three pipeline registers are required for both multiply and multiply-accumulate operations to run at full speed. The multiply operation in the first stage generates two partial products that need to be added together in the second stage.

When only one or two registers exist in the multiplier design, the M register should always be used to save power and improve performance.

Add/Sub and Logic Unit operations require at least two pipeline registers (input, output) to run at full speed.

The cascade capabilities of the DSP slice are extremely efficient at implementing high-speed pipelined filters built on the adder cascades instead of adder trees.

Multiplexers are controlled with dynamic control signals, such as OPMODE, ALUMODE, and CARRYINSEL, enabling a great deal of flexibility. Designs using registers and dynamic opmodes are better equipped to take advantage of the DSP slice's capabilities than combinatorial multiplies.

In general, the DSP slice supports both sequential and cascaded operations due to the dynamic OPMODE and cascade capabilities. Fast Fourier Transforms (FFTs), floating point, computation (multiply, add/sub, divide), counters, and large bus multiplexers are some applications of the DSP slice.

Additional capabilities of the DSP slice include synchronous resets and clock enables, dual A input pipeline registers, pattern detection, Logic Unit functionality, single instruction/multiple data (SIMD) functionality, and MACC and Add-Acc extension to 96 bits. The DSP slice supports convergent and symmetric rounding, terminal count detection and auto-resetting for counters, and overflow/underflow detection for sequential accumulators. A 96-bit wide XOR function can be implemented as eight 12-bit wide XOR, four 24-bit wide XOR, or two 48-bit wide XOR.

Architectural Highlights of the DSP48E2 Slice

The DSP48E2 slice contains a pre-adder after the A and B registers with a 27-bit input vector called D. The D register can be used either as the pre-adder register or an alternate input to the multiplier. The DSP48E2 specific features are highlighted in Figure 2-2.

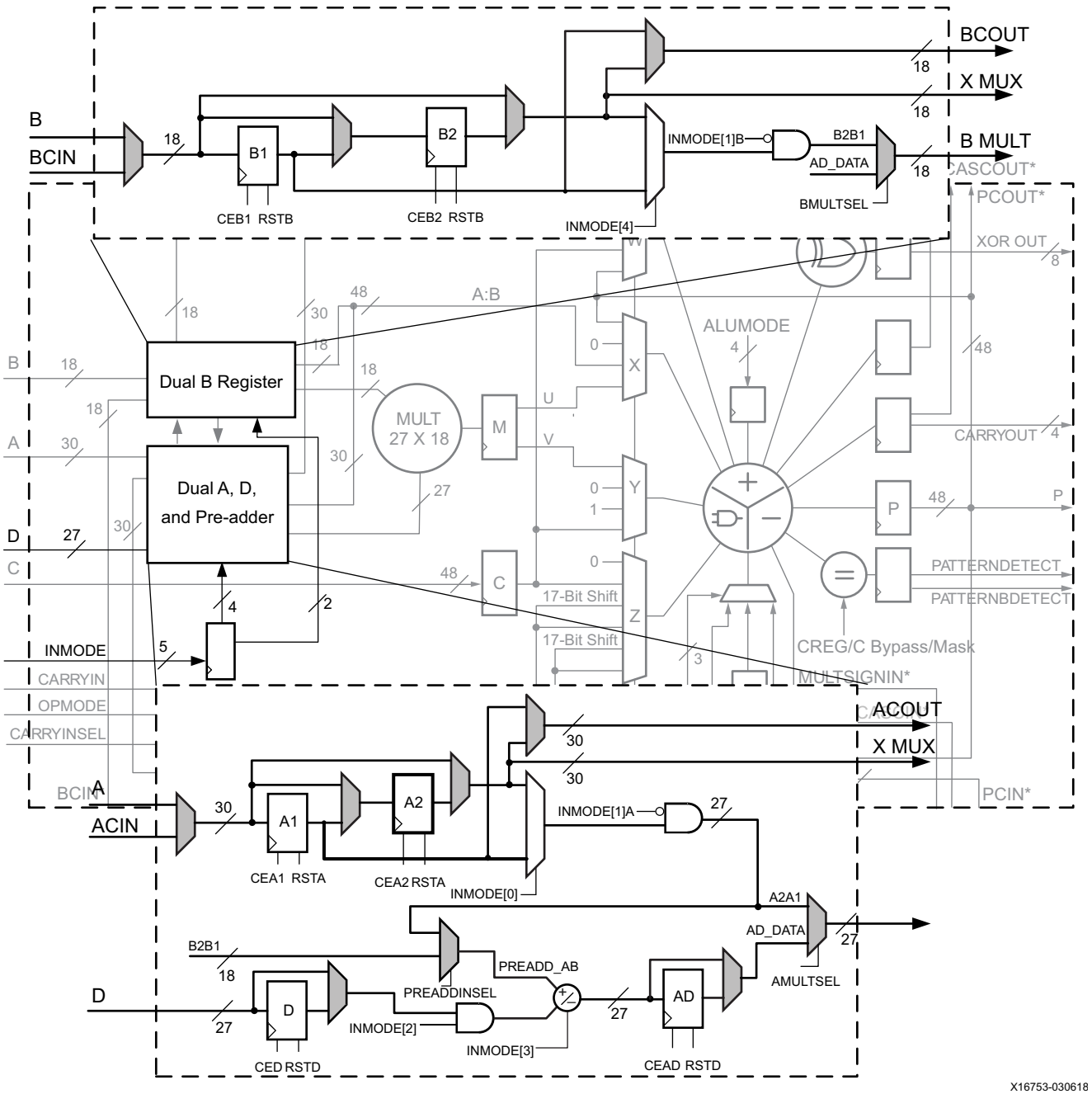


Figure 2-2: **Hierarchical View of the DSP48E2 Slice Input Registers and Pre-adder**

Each DSP48E2 slice has a two-input multiplier followed by multiplexers and a four-input adder/subtracter/accumulator. The DSP48E2 multiplier has asymmetric inputs and accepts an 18-bit two's complement operand and a 27-bit two's complement operand. The multiplier stage produces a 45-bit two's complement result in the form of two partial products. These partial products are sign-extended to 48 bits in the X multiplexer and Y multiplexer and fed into four-input adder for final summation. This results in a 45-bit multiplication output, which has been sign-extended to 48 bits. Therefore, when the multiplier is used, the adder effectively becomes a three-input adder.

The second stage adder/subtracter accepts four 48-bit, two's complement operands and produces a 48-bit, two's complement result when the multiplier is bypassed by setting USE_MULT attribute to NONE and with the appropriate OPMODE setting. In SIMD mode, the 48-bit adder/subtracter also supports dual 24-bit or quad 12-bit SIMD arithmetic operations with CARRYOUT bits. In this configuration, bitwise logic operations on two 48-bit binary numbers (and three 48-bit binary numbers in the special XOR3 case) are also supported with dynamic ALUMODE control signals.

Higher level DSP functions are supported by cascading individual DSP48E2 slices in a DSP48E2 column. Two datapaths (ACOUT and BCOUT) and the DSP48E2 slice outputs (PCOUT, MULTSIGNOUT, and CARRYCASCOUT) provide the cascade capability. The ability to cascade datapaths is useful in filter designs. For example, a Finite Impulse Response (FIR) filter design can use the cascading inputs to arrange a series of input data samples and the cascading outputs to arrange a series of partial output results. The ability to cascade provides a high-performance and low-power implementation of DSP filter functions because the general routing in the fabric is not used.

The C input allows the formation of many 3-input mathematical functions, such as 3-input addition or 2-input multiplication with an addition. One subset of this function is the valuable support of symmetrically rounding a multiplication toward zero or toward infinity. The C input together with the pattern detector also supports convergent rounding.

For multi-precision arithmetic, the DSP48E2 slice provides a right wire shift by 17. Thus, a partial product from one DSP48E2 slice can be right justified and added to the next partial product computed in an adjacent DSP48E2 slice. Using this technique, the DSP48E2 slices can be used to build bigger multipliers.

Programmable pipelining of input operands, intermediate products, and accumulator outputs enhances throughput. The 48-bit internal bus (PCOUT/PCIN) allows for aggregation of DSP slices in a single column. CLB logic is needed when spanning multiple columns.

The pattern detector at the output of the DSP48E2 slice provides support for convergent rounding, overflow/underflow, block floating point, and support for accumulator terminal count (counter auto reset). The pattern detector can detect if the output of the DSP48E2 slice matches a pattern, as qualified by a mask.

Simplified DSP48E2 Slice Operation

The math portion of the DSP48E2 slice consists of a 27-bit pre-adder, a 27-bit by 18-bit two's complement multiplier followed by four 48-bit datapath multiplexers (with outputs W, X, Y, and Z). This is followed by a four-input adder/subtracter or two-input logic unit (see Figure 2-4). When using two-input logic unit, the multiplier cannot be used.

The data and control inputs to the DSP48E2 slice feed the arithmetic and logic stages. The A and B data inputs can optionally be registered one or two times to assist the construction of different, highly pipelined, DSP application solutions. The D path and the AD path can each be registered once. The other data inputs and the control inputs can be optionally registered once. Maximum frequency operation as specified in the UltraScale and UltraScale+ device data sheets [Ref 2] is achieved by using pipeline registers.

In its most basic form, the output of the adder/subtracter/logic unit is a function of its inputs. The inputs are driven by the upstream multiplexers, carry select logic, and multiplier array.

Equation 2-1 summarizes the combination of W, X, Y, Z, and CIN by the adder/subtracter. The CIN, W multiplexer output, X multiplexer output, and Y multiplexer output are always added together. This combined result can be selectively added to or subtracted from the Z multiplexer output. The second option is obtained by setting the ALUMODE to `0001`.

$$\text{Adder/Subtracter Out} = (Z \pm (W + X + Y + CIN)) \text{ or } (-Z + (W + X + Y + CIN) - 1) \qquad \textit{Equation 2-1}$$

A typical use of the slice is where A and B inputs 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 both X and Y multiplexer outputs to feed the adder. The two 45-bit partial products from the multiplier are sign extended to 48 bits before being sent to the adder/subtracter.

When not using the first stage multiplier, the 48-bit, dual input, bit-wise logic function implements AND, OR, NOT, NAND, NOR, XOR, and XNOR. The inputs to these functions are:

- All 0s on the W multiplexer
- A:B or P on the X multiplexer
- Either all 1s or all 0s on the Y multiplexer depending on logic operation
- Either C, P, or PCIN on the Z multiplexer.

Since PCIN is a cascade input from a lower DSP slice, creating even wider logic operations is feasible via this cascade path. A 48-bit, triple input, bit-wise XOR3 logic operation is supported when the Y multiplexer selects the C input and ALUMODE[3:0] = `0100`.

The output of the adder/subtracter or logic unit feeds the pattern detector logic. The pattern detector allows the DSP48E2 slice to support Convergent Rounding, Counter

Autoreset when a count value has been reached, and Overflow/Underflow/Saturation in accumulators. In conjunction with the logic unit, the pattern detector can be extended to perform a 48-bit dynamic comparison of two 48-bit fields. This enables functions such as A:B NAND C = = 0, or A:B (bit-wise logic) C = = Pattern to be implemented.

Figure 2-3 shows the DSP48E2 slice in a very simplified form. The nine OPMODE bits control the selects of W, X, Y, and Z multiplexers, feeding the inputs to the adder/subtracter or logic unit. In all cases, the 45-bit partial product data from the multiplier to the X and Y multiplexers is sign extended, forming 48-bit input datapaths to the adder/subtracter. Based on 45-bit operands and a 48-bit accumulator output, the number of *guard bits* (i.e., bits available to guard against overflow) is 3. To extend the number of MACC operations, the MACC_EXTEND feature should be used, which allows the MACC to extend to 96 bits with two DSP48E2 slices. If A is limited to 18 bits (sign-extended to 27), then there are 12 guard bits for the MACC. The CARRYOUT bits are invalid during multiply operations. Combinations of OPMODE, ALUMODE, CARRYINSEL, and CARRYIN control the function of the adder/subtracter or logic unit.

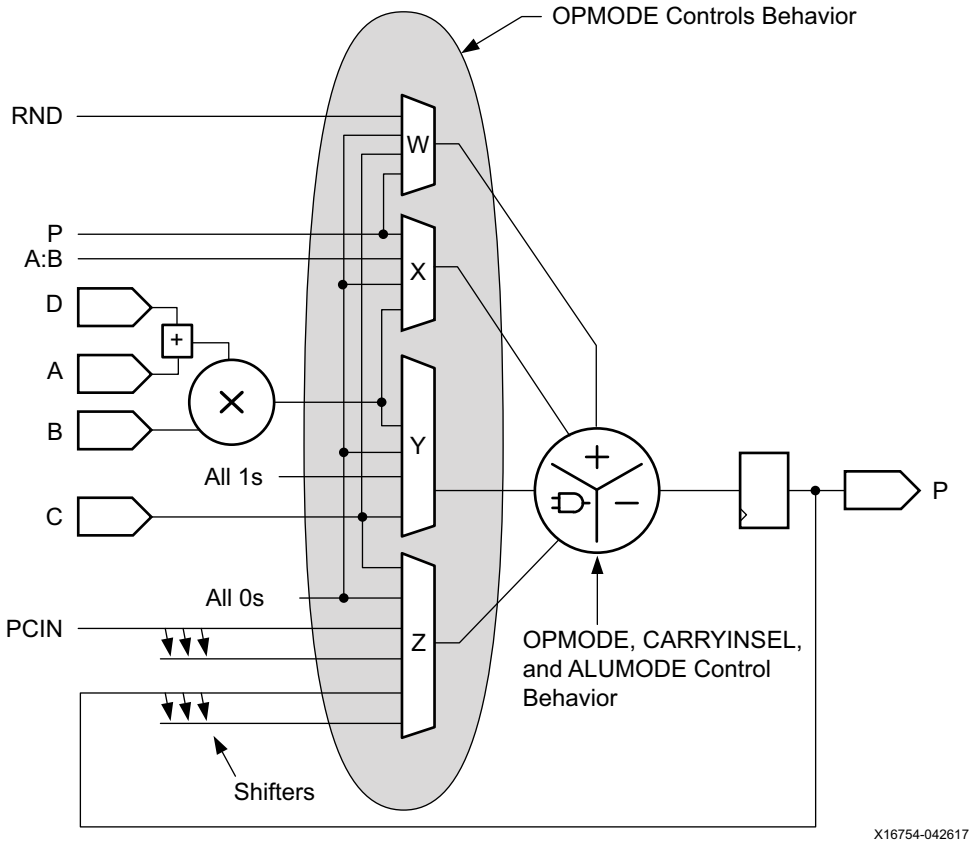


Figure 2-3: **Simplified DSP Slice Operation**

Input Ports

This section describes the input ports of the DSP48E2 slice in detail. The input ports of the DSP48E2 slice are highlighted in Figure 2-4.

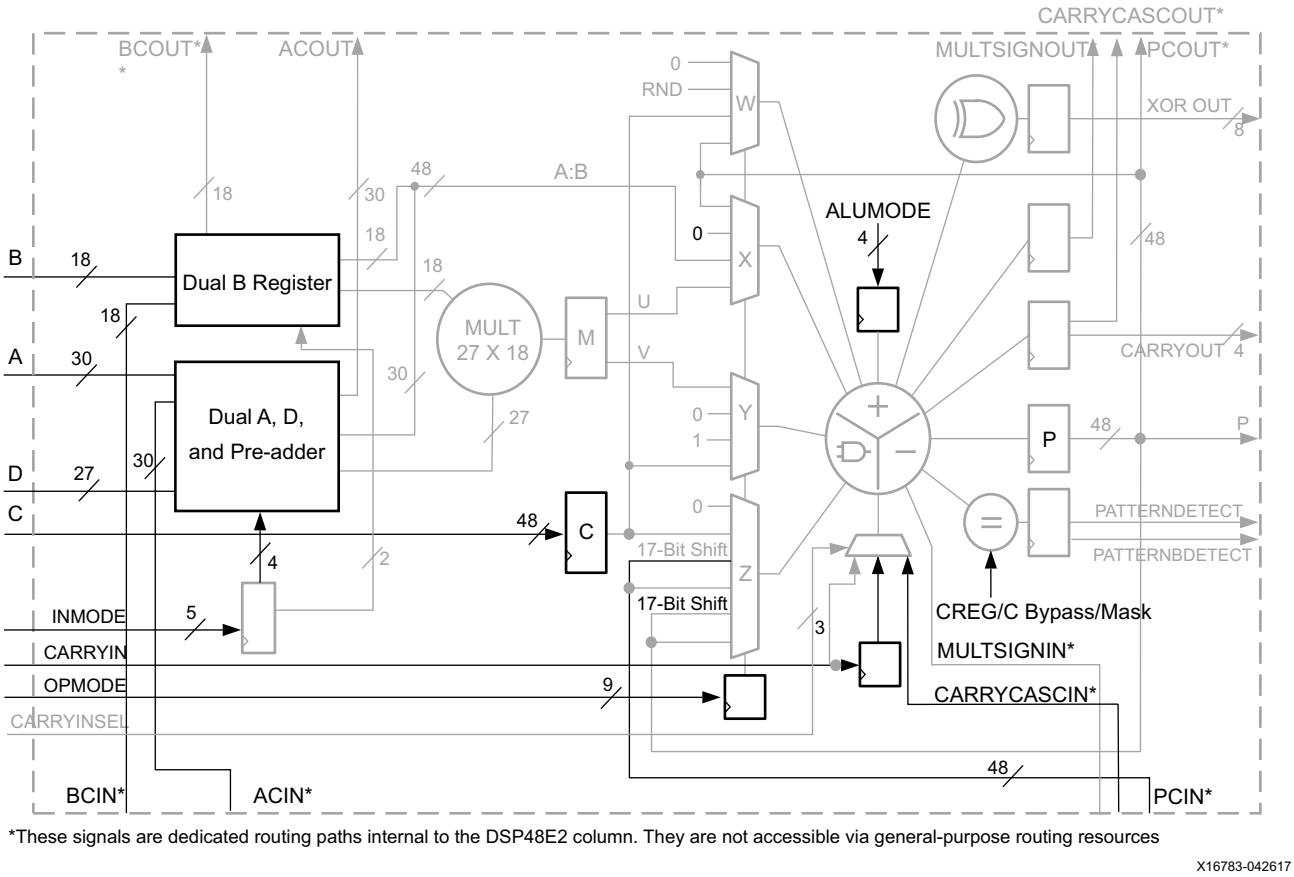


Figure 2-4: **Input Ports in the DSP48E2 Slice**

A, B, C, and D Ports

The DSP48E2 slice input data ports support many common DSP and math algorithms. The DSP48E2 slice has four direct input data ports labeled A, B, C, and D. The A data port is 30 bits wide, the B data port is 18 bits wide, the C data port is 48 bits wide, and the pre-adder D data port is 27 bits wide.

The 27-bit A (A[26:0]) and 18-bit B ports supply input data to the 27-bit by 18-bit, two's complement multiplier. With independent C port, each DSP48E2 slice is capable of Multiply-Add, Multiply-Subtract, and Multiply-Round operations.

Concatenated A and B ports (A:B) bypass the multiplier and feed the X multiplexer input. The 30-bit A input port forms the upper 30 bits of A:B concatenated datapath, and the 18-bit B input port forms the lower 18 bits of the A:B datapath. The A:B datapath, together with the C input port, enables each DSP48E2 slice to implement a full 48-bit

adder/subtracter provided the multiplier is not used, which is achieved by setting USE_MULT to NONE (or DYNAMIC).

Each DSP48E2 slice also has two cascaded input datapaths (ACIN and BCIN), providing a cascaded input stream between adjacent DSP48E2 slices. The cascaded path is 30 bits wide for the A input and 18 bits wide for the B input. Applications benefiting from this feature include FIR filters, complex multiplication, multi-precision multiplication and complex MACCs.

The A and B input port and the ACIN and BCIN cascade port can have 0, 1, or 2 pipeline stages in its datapath. The dual A, D, and pre-adder port logic is shown in Figure 2-5. The dual B register port logic is shown in Figure 2-6. The different pipestages are set using attributes. Attributes AREG and BREG are used to select the number of pipeline stages for A and B direct inputs to the X multiplexer to the ALU, and INMODE[0] can dynamically change the number of pipeline stages to the multiplier. Attributes ACASCREG and BCASCREG select the number of pipeline stages in the ACOUT and BCOUT cascade datapaths. The allowed attribute settings are shown in Table 3-3, page 53. Multiplexers controlled by configuration bits select flow through paths, optional registers, or cascaded inputs. The data port registers allow users to typically trade off increased clock frequency (i.e., higher performance) vs. data latency.

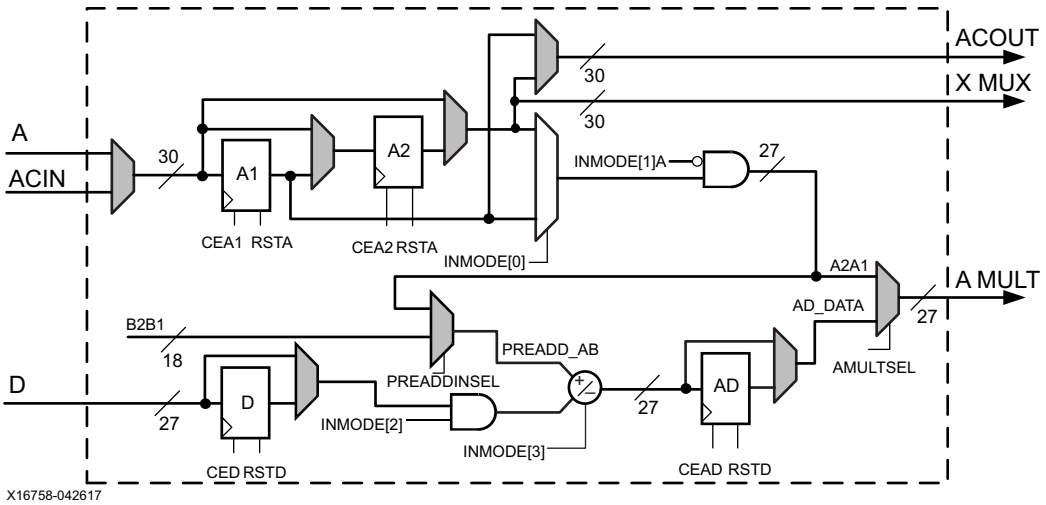


Figure 2-5: **Dual A, D, and Pre-Adder Logic**

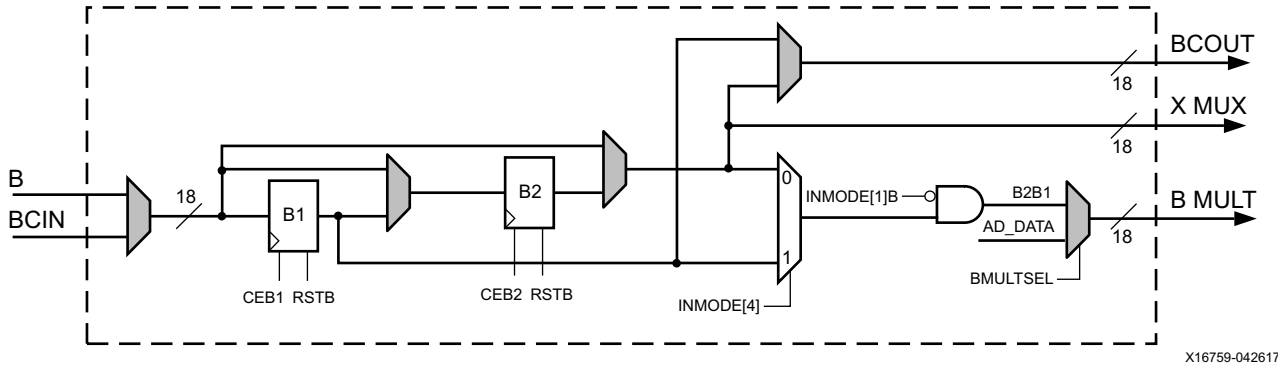


Figure 2-6: **Dual B Register Logic**

Table 2-1 and Table 2-2 shows the encoding for the INMODE[4:0] dynamic control bits and AMULTSEL, BMULTSEL, and PREADDINSEL static control bits. Note that the DSP48E2 attribute AMULTSEL has replaced the DSP48E1 attribute USE_DPORT due to increased functionality of the pre-adder.

These bits select the functionality of the pre-adder, the A, B, and D input registers. AMULTSEL must be set to AD to enable the pre-adder functions described in Table 2-1 and Table 2-2.

In summary, the INMODE dynamic control signals along with AMULTSEL, BMULTSEL, and PREADDINSEL static attributes control the pre-adder functionality and A, B, and D register bus multiplexers that precede the multiplier. The DSP48E2 supports two-deep A or B sourcing the pre-adder as well as a pre-adder squaring function.

Table 2-1: **INMODE[4:0] Functions with Legacy Options (AREG/BREG = 1 or 2)**

INMODE[4]	INMODE[3]	INMODE[2]	INMODE[1]	INMODE[0]	BMULTSEL	AMULTSEL (USE_DPORT)	Multiplier A Port	Multiplier B Port	Pre-Adder/ Multiplier Function
0/1	0	0	0	0	B	A (FALSE)	A2	B2/B1	A2 * B
0/1	0	0	0	1	B	A (FALSE)	A1	B2/B1	A1 * B
0/1	0	0	1	0	B	A (FALSE)	Zero	B2/B1	B * Zero
0/1	0	0	1	1	B	A (FALSE)	Zero	B2/B1	B * Zero
0/1	0	0	0	0	B	AD (TRUE)	A2	B2/B1	A2 * B
0/1	0	0	0	1	B	AD (TRUE)	A1	B2/B1	A1 * B
0/1	0	0	1	0	B	AD (TRUE)	Zero	B2/B1	B * Zero
0/1	0	0	1	1	B	AD (TRUE)	Zero	B2/B1	B * Zero
0/1	0	1	0	0	B	AD (TRUE)	D + A2(1)	B2/B1	(D + A2) * B
0/1	0	1	0	1	B	AD (TRUE)	D + A1(1)	B2/B1	(D + A1) * B
0/1	0	1	1	0	B	AD (TRUE)	D	B2/B1	D * B
0/1	0	1	1	1	B	AD (TRUE)	D	B2/B1	D * B
0/1	1	0	0	0	B	AD (TRUE)	–A2	B2/B1	–(A2 * B)
0/1	1	0	0	1	B	AD (TRUE)	–A1	B2/B1	–(A1 * B)
0/1	1	0	1	0	B	AD (TRUE)	Zero	B2/B1	B * Zero
0/1	1	0	1	1	B	AD (TRUE)	Zero	B2/B1	B * Zero
0/1	1	1	0	0	B	AD (TRUE)	D – A2(1)	B2/B1	(D – A2) * B
0/1	1	1	0	1	B	AD (TRUE)	D – A1(1)	B2/B1	(D – A1) * B
0/1	1	1	1	0	B	AD (TRUE)	D	B2/B1	D * B
0/1	1	1	1	1	B	AD (TRUE)	D	B2/B1	D * B

Notes:

1. Set the data on the D and the A ports so the pre-adder, which does not support saturation, does not overflow or underflow. See Pre-Adder, page 36.

Table 2-2: **INMODE[4:0] Functions with New Pre-Adder Options**

INMODE[4]	INMODE[3]	INMODE[2]	INMODE[1]	INMODE[1]A[1]	INMODE[1]B[1]	INMODE[0]	PREADDINSEL	BMULTSEL	AMULTSEL (USE_DPORT)	Multiplier A Port	Multiplier B Port[3]	Pre-Adder/Multiplier Function
0/1	0	0	0	0	0	0/1	A	B	A (FALSE)	A2/A1	B2/B1	A * B
0/1	0	0	0	0	0	0/1	B	B	A (FALSE)	A2/A1	B2/B1	A * B
0/1	0/1	1	0	0	0	0/1	A	B	AD (TRUE)	D ± A2/A1[2]	B2/B1	(D ± A) * B
0/1	0	0	1	1	0	X	A	B	A (FALSE)	Zero	B2/B1	B * Zero
0/1	0	0	1	0	1	X	B	B	A (FALSE)	A2/A1	Zero	A * Zero
X	0/1	1	0	0	0	0/1	A	AD	AD (TRUE)	D ± A2/A1[2]	D ± A2/A1[2]	$(D \pm A)^2$
X	0	1	1	1	0	X	A	AD	AD (TRUE)	D	D	D^2
X	0/1	0	0	0	0	0/1	A	AD	AD (TRUE)	± A2/A1	± A2/A1	A^2
X	0/1	0	0	0	0	0/1	A	AD	A (FALSE)	A2/A1	± A2/A1	$\pm (A^2)$
X	0/1	1	0	0	0	0/1	A	AD	A (FALSE)	A2/A1	D ± A2/A1[2]	(D ± A) * A
0/1	0/1	1	0	0	0	0/1	B	AD	A (FALSE)	A2/A1	D ± B2/B1[2]	(D ± B) * A
X	0	1	1	0	1	0/1	B	AD	A (FALSE)	A2/A1	D	D * A
0/1	0/1	0	0	0	0	0/1	B	AD	A (FALSE)	A2/A1	± B2/B1	(± B) * A
0/1	0/1	1	0	0	0	0/1	B	AD	AD (TRUE)	D ± B2/B1[2]	D ± B2/B1[2]	$(D \pm B)^2$
0/1	0/1	0	0	0	0	0/1	B	AD	AD (TRUE)	± B2/B1	± B2/B1	B^2
0/1	0/1	0	0	0	0	0/1	B	B	AD (TRUE)	± B2/B1	B2/B1	$\pm (B^2)$
0/1	0/1	1	0	0	0	0/1	B	B	AD (TRUE)	D ± B2/B1[2]	B2/B1	(D ± B) * B

Notes:

1. INMODE[1]A and INMODE[1]B are internal signals defined by the user settings of PREADDINSEL and INMODE[1]. If PREADDINSEL=A, INMODE[1]A (see Figure 2-5, page 23) is INMODE[1] and INMODE[1]B (see Figure 2-6, page 24) is 0. If PREADDINSEL=B, INMODE[1]B is INMODE[1] and INMODE[1]A is 0.
2. Set the data on the D and the A or B ports so the pre-adder, which does not support saturation, does not overflow or underflow. See Pre-Adder, page 36.
3. A or D are limited to 18 bits when provided through the B port, and are limited to 17-bit two's complement sign-extended numbers when the pre-adder is used.

INMODE[0] selects between A1 (INMODE[0] = 1) and the A2 mux controlled by AREG (INMODE[0] = 0).

INMODE[1] may be used to gate the A or B datapath to use the pre-adder to create a 2:1 bus multiplexer along with the INMODE[2] control signal.

When INMODE[2] = 0, the D input to the pre-adder is 0. INMODE[1] and INMODE[2] enable multiplexing between the D register and the A or B registers, without having to use resets to force them to zero.

INMODE[3] provides pre-adder subtract control, where INMODE[3] = 1 indicates subtract and INMODE[3] = 0 indicates add of A or B to D. When D is gated off, this dynamic inversion can provide the absolute value of A or B.

INMODE[4] selects between B1 (INMODE[4] = 1) and the B2 mux controlled by BREG (INMODE[4] = 0).

The 48-bit C port is used as a general input to the W, Y and Z multiplexers to perform add, subtract, four-input add/subtract, and logic functions. The C input is also connected to the pattern detector for rounding function implementations. The C port logic is shown in Figure 2-7. The CREG attribute selects the number of pipestages for the C input datapath.

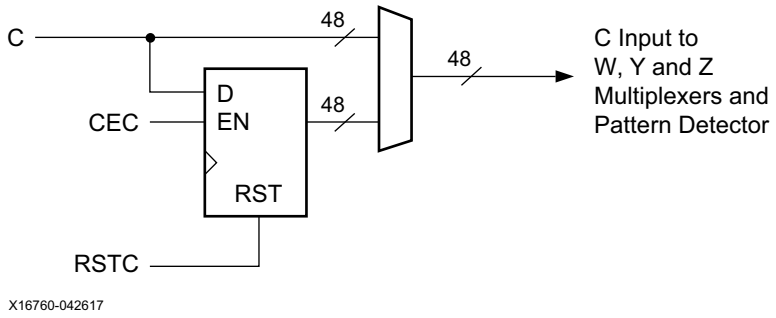


Figure 2-7: **C Port Logic**

OPMODE, ALUMODE, and CARRYINSEL Port Logic

The OPMODE, ALUMODE, and CARRYINSEL port logic supports flow through or registered input control signals. Multiplexers controlled by configuration bits select flow through or optional registers. The control port registers allow you to trade off increased clock frequency (i.e., higher performance) vs. data latency. The registers have independent clock enables and resets. The OPMODE and CARRYINSEL registers are reset by RSTCTRL. The ALUMODE is reset by RSTALUMODE. The clock enables and the OPMODE, ALUMODE, and CARRYINSEL port logic are shown in Figure 2-8.

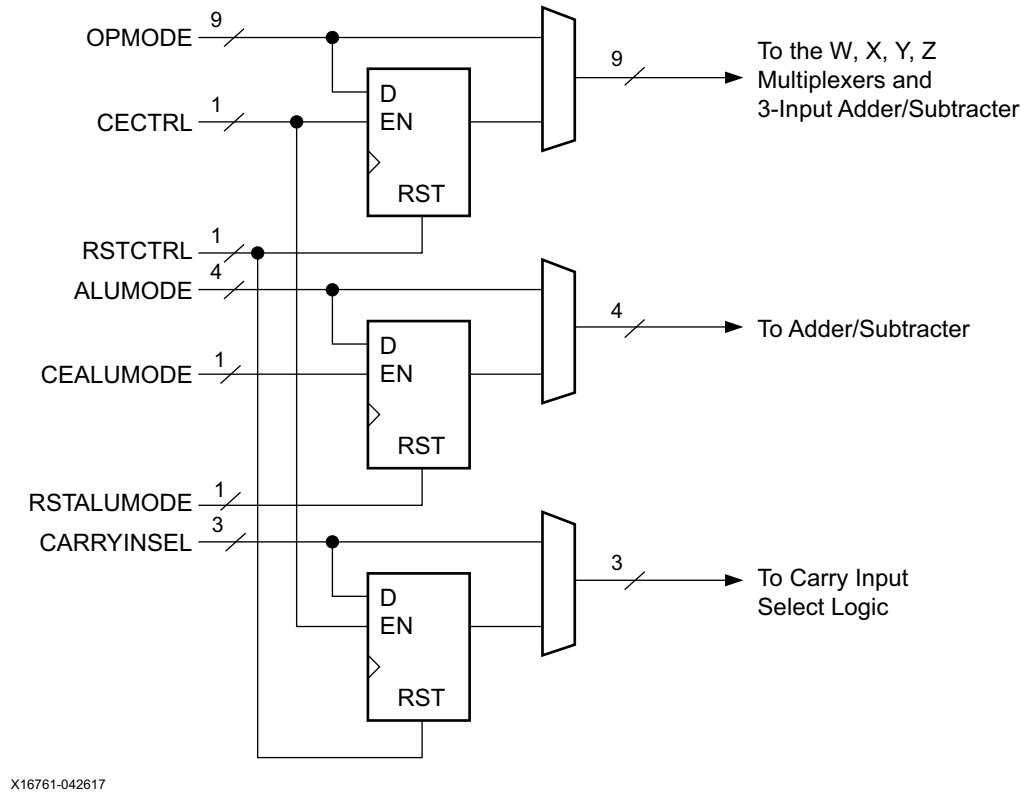


Figure 2-8: **OPMODE, ALUMODE, and CARRYINSEL Port Logic**

W, X, Y, and Z Multiplexers

The OPMODE (Operating Mode) control input contains fields for W, X, Y, and Z multiplexer selects.

The OPMODE input provides a way for you to dynamically change DSP48E2 functionality from clock cycle to clock cycle (e.g., when altering the internal datapath configuration of the DSP48E2 slice relative to a given calculation sequence).

The OPMODE bits can be optionally registered using the OPMODEREG attribute (as noted in Table 3-4).

Table 2-3, Table 2-4, Table 2-5, and Table 2-6 list the possible values of OPMODE and the resulting function at the outputs of the four multiplexers (W, X, Y, and Z multiplexers). The multiplexer outputs supply four operands to the following adder/subtracter. Not all possible combinations for the multiplexer select bits are allowed. Some are marked in the tables as “illegal selection” and give undefined results. If the multiplier output is selected, then both the X and Y multiplexers are used to supply the multiplier partial products to the adder/subtracter.

Table 2-3: **OPMODE Control Bits Select W Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	W Multiplexer Output	Notes
00	xxx	xx	xx	0	Default, must select for logic operations
01	xxx	xx	xx	P	Requires PREG = 1
10	xxx	xx	xx	RND	-
11	xxx	xx	xx	C	-

Table 2-4: **OPMODE Control Bits Select X Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	X Multiplexer Output	Notes
xx	xxx	xx	00	0	Default
xx	xxx	01	01	M	Must select with OPMODE[3:2] = 01
xx	xxx	xx	10	P	Requires PREG = 1
xx	xxx	xx	11	A:B	48-bits wide

Table 2-5: **OPMODE Control Bits Select Y Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	Y Multiplexer Output	Notes
xx	xxx	00	xx	0	Default
xx	xxx	01	01	M	Must select with OPMODE[1:0] = 01
xx	xxx	10	xx	48'FFFFFFFFFFFF	Used mainly for logic unit bitwise operations on the X and Z multiplexers
xx	xxx	11	xx	C	

Table 2-6: **OPMODE Control Bits Select Z Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	Z Multiplexer Output	Notes
xx	000	xx	xx	0	Default
xx	001	xx	xx	PCIN	-
xx	010	xx	xx	P	Requires PREG = 1
xx	011	xx	xx	C	-
00	100	10	00	P	Use for MACC extend only. Requires PREG = 1
xx	101	xx	xx	17-bit Shift (PCIN)	-
xx	110	xx	xx	17-bit Shift (P)	Requires PREG = 1
xx	111	xx	xx	xx	Illegal selection

ALUMODE Inputs

The 4-bit ALUMODE controls the behavior of the second stage add/sub/logic unit. ALUMODE = `0000` selects add operations of the form Z + (W + X + Y + CIN). CIN is the output of the CARRYIN mux (see Figure 2-9). ALUMODE = `0011` selects subtract operations of the form Z – (W + X + Y + CIN). ALUMODE = `0001` can implement –Z + (W + X + Y + CIN) – 1. ALUMODE = `0010` can implement –(Z + W + X + Y + CIN) – 1, which is equivalent to not (Z + W + X + Y + CIN). The negative of a two's complement number is obtained by performing a bitwise inversion and adding one, e.g., –k = not (k) + 1. Other subtract and logic operations can also be implemented with the enhanced add/sub/logic unit. See Table 2-7.

Table 2-7: **Four-Input ALUMODE Operations**

DSP Operation	OPMODE[8:0]	ALUMODE[3:0]			
		3	2	1	0
Z +W + X + Y + CIN	Any legal OPMODE	0	0	0	0
Z – (W + X + Y + CIN)	Any legal OPMODE	0	0	1	1
–Z + (W + X + Y + CIN) – 1 = not (Z) + W + X + Y + CIN	Any legal OPMODE	0	0	0	1
not (Z +W + X + Y + CIN) = –Z – W - X – Y – CIN - 1	Any legal OPMODE	0	0	1	0

Notes:
1. In two's complement: –Z = not (Z) + 1.

See Table 2-10, page 38 for two-input ALUMODE operations and Table 5-3, page 70.

Carry Input Logic

The carry input logic result is a function of a 3-bit CARRYINSEL signal. The inputs to the carry input logic appear in Figure 2-9. 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 silicon. The possible carry inputs to the carry logic are "gathered" prior to the outputs of the W, X, Y, and Z multiplexers. CARRYIN has no dependency on the OPMODE selection.

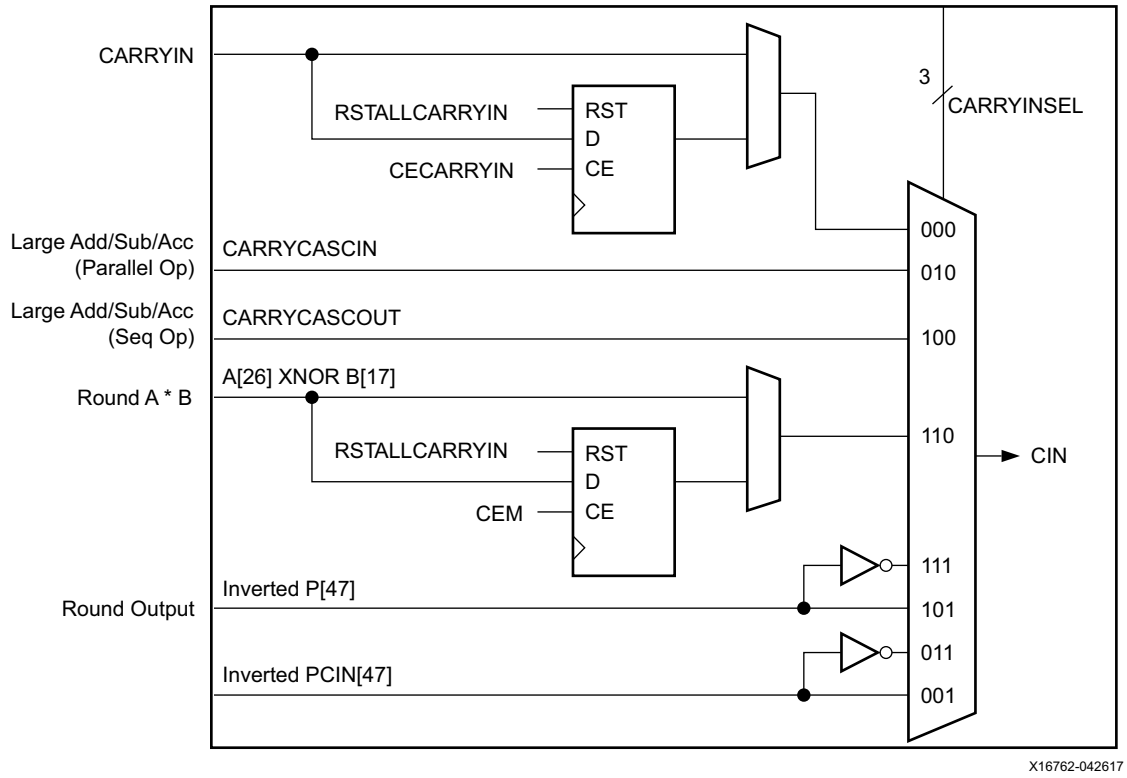


Figure 2-9: **CARRYINSEL Port Logic**

Figure 2-9 shows eight inputs selected by the 3-bit CARRYINSEL control. The first input, CARRYIN (CARRYINSEL set to binary 000), is driven from general logic. This option allows implementation of a carry function based on user logic. CARRYIN can be optionally registered. The next input, (CARRYINSEL is equal to binary 010) is the CARRYCASCIN input from an adjacent DSP48E2 slice. The third input (CARRYINSEL is equal to binary 100) is the CARRYCASCOUT from the same DSP48E2 slice, fed back to itself. Refer to Table 3-4, page 57 for internal register definitions pertaining to carry logic.

The fourth input (CARRYINSEL is equal to binary 110) is A[26] XNOR B[17] for symmetrically rounding multiplier outputs. This signal can be optionally registered to match the MREG pipeline delay. The fifth and sixth inputs (CARRYINSEL is equal to binary 111 and 101) selects the true or inverted P output MSB P[47] for symmetrically rounding the P output. The seventh and eight inputs (CARRYINSEL is equal to binary 011 and 001) selects the true or inverted cascaded P input MSB PCIN[47] for symmetrically rounding the cascaded P input.

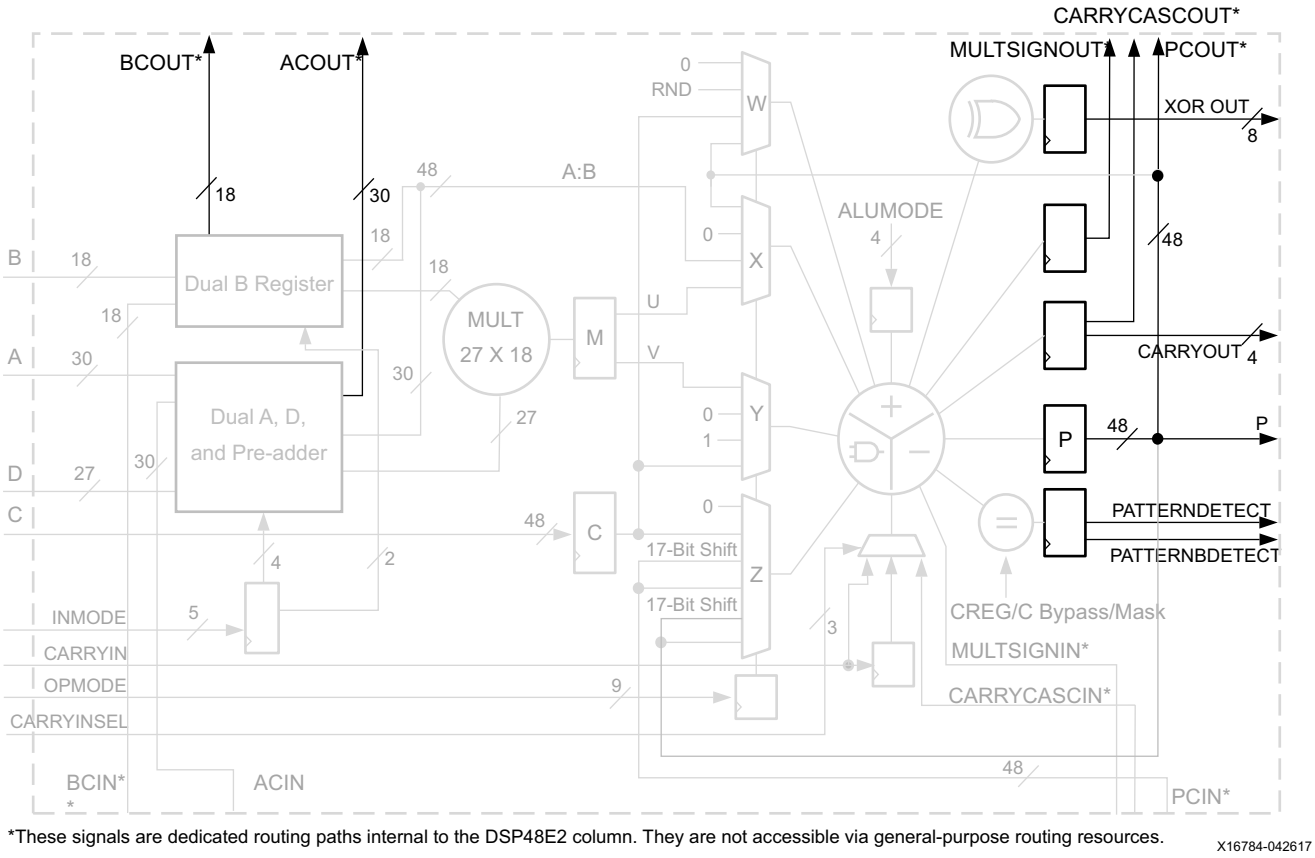
Table 2-8 lists the possible values of the three carry input select bits (CARRYINSEL) and the resulting carry inputs or sources.

Table 2-8: **CARRYINSEL Control Carry Source**

CARRYINSEL			Select	Notes
2	1	0		
0	0	0	CARRYIN	General interconnect
0	0	1	~PCIN[47]	Rounding PCIN (round towards infinity)
0	1	0	CARRYCASCIN	Larger add/sub/acc (parallel operation)
0	1	1	PCIN[47]	Rounding PCIN (round towards zero)
1	0	0	CARRYCASCOUT	For larger add/sub/acc (sequential operation via internal feedback). Requires PREG = 1
1	0	1	~P[47]	Rounding P (round towards infinity). Requires PREG = 1
1	1	0	A[26] XNOR B[17]	Rounding A x B
1	1	1	P[47]	For rounding P (round towards zero). Requires PREG = 1

Output Ports

This section describes the output ports of the DSP48E2 slice in detail. The output ports of the DSP48E2 slice are shown in Figure 2-10.

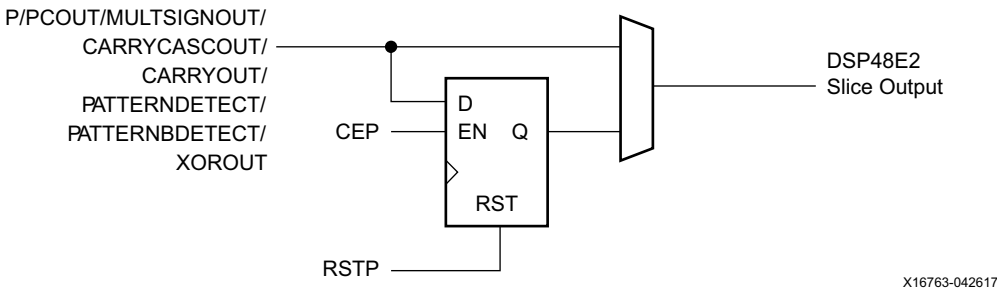


*These signals are dedicated routing paths internal to the DSP48E2 column. They are not accessible via general-purpose routing resources.

X16784-042617

Figure 2-10: **Output Ports in the DSP48E2 Slice**

All the output ports except ACOUT and BCOUT are reset by RSTP and enabled by CEP (see Figure 2-11). ACOUT and BCOUT are reset by RSTA and RSTB, respectively (shown in Figure 2-5 and Figure 2-6).



X16763-042617

Figure 2-11: **Output Port Logic**

P Port

Each DSP48E2 slice has a 48-bit output port P. This output can be connected (cascaded connection) to the adjacent DSP48E2 slice internally through the PCOUT path. The PCOUT connects to the input of the Z multiplexer (PCIN) in the adjacent DSP48E2 slice. This path provides an output cascade stream between adjacent DSP48E2 slices.

CARRYCASCOUT and CARRYOUT Ports

The carry out from each DSP48E2 slice can be sent to the logic resources using the CARRYOUT port. This port is 4 bits wide. CARRYOUT[3] is the valid carry output for a two-input 48-bit adder/subtracter or one-input accumulator. In this case, USE_SIMD = ONE48 is the default setting and represents a non-SIMD configuration. When a two-input adder/subtracter or one-input accumulator is used in SIMD mode, such as TWO24 or FOUR12, the valid CARRYOUT signals are listed in Table 2-9. The CARRYOUT signals are not valid if three-input (or four-input) adder/subtracter (e.g., A:B + C + PCIN) or two-input (or three-input) accumulator (e.g., A:B + C + P) configurations are used or if the multiplier is used.

Table 2-9: **CARRYOUT Bit Associated with Different SIMD Modes**

SIMD Mode	Adder Bit Width	Corresponding CARRYOUT
FOUR12	P[11:0]	CARRYOUT[0]
	P[23:12]	CARRYOUT[1]
	P[35:24]	CARRYOUT[2]
	P[47:36]	CARRYOUT[3]
TWO24	P[23:0]	CARRYOUT[1]
	P[47:24]	CARRYOUT[3]
ONE48	P[47:0]	CARRYOUT[3]

See also Table 2-7, page 30 for 4-input ALUMODE operations.

The CARRYOUT signal is cascaded to the next adjacent DSP48E2 slice using the CARRYCASCOUT port. Larger add, subtract, ACC, and MACC functions can be implemented in the DSP48E2 slice using the CARRYCASCOUT output. The 1-bit CARRYCASCOUT signal corresponds to CARRYOUT[3], but is not identical. The CARRYCASCOUT signal is also fed back into the same DSP48E2 slice via the CARRYINSEL multiplexer.

The CARRYOUT[3] signal should be ignored when the multiplier or a 3-input (or 4-input) add/subtract operation is used. Because a MACC operation includes a three-input adder in the accumulator stage, the combination of MULTSIGNOUT and CARRYCASCOUT signals is required to perform a 96-bit MACC, spanning two DSP48E2 slices. The second DSP48E2 slice's OPMODE must be MACC_EXTEND (001001000) to use both CARRYCASCOUT and MULTSIGNOUT, thereby eliminating the ternary adder carry restriction for the upper DSP48E2 slice. The actual hardware implementation of CARRYOUT/CARRYCASCOUT and the

differences between them are described in Chapter 5, Cascading: CARRYOUT, CARRYCASCOUT, and MULTSIGNOUT.

MULTSIGNOUT Logic

MULTSIGNOUT is a software abstraction of the hardware signal. It is modeled as the MSB of the multiplier output and used only in MACC extension applications to build a 96-bit MACC. The actual hardware implementation of MULTSIGNOUT is described in Chapter 5, Cascading: CARRYOUT, CARRYCASCOUT, and MULTSIGNOUT.

The MSB of a multiplier output is cascaded to the next DSP48E2 slice using the MULTSIGNIN signal and can be used only in MACC extension applications to build a 96-bit accumulator. The actual hardware implementation of MULTSIGNOUT is described in Chapter 5, Cascading: CARRYOUT, CARRYCASCOUT, and MULTSIGNOUT.

PATTERNDETECT and PATTERNBDETECT Logic

A pattern detector on the output of the DSP48E2 slice detects if the P bus matches a specified pattern or if it exactly matches the complement of the pattern. The PATTERNDETECT output goes High if the output of the adder matches a set pattern. The PATTERNBDETECT output goes High if the output of the adder matches the complement of the set pattern.

A mask field can also be used to hide certain bit locations in the pattern detector. PATTERNDETECT computes ((P == pattern)||mask) on a bitwise basis and then ANDs the results to a single output bit. Similarly, PATTERNBDETECT can detect if ((P == ~pattern)||mask). The pattern and the mask fields can each come from a distinct 48-bit configuration field or from the (registered) C input. When the C input is used as the PATTERN, the OPMODE should be set to select a 0 at the input of the Z multiplexer. If all the registers are reset, PATTERNDETECT is High for one clock cycle immediately after the RESET is deasserted.

The pattern detector allows the DSP48E2 slice to support convergent rounding and counter auto reset when a count value has been reached as well as support overflow, underflow, and saturation in accumulators.

Overflow and Underflow Logic

The dedicated OVERFLOW and UNDERFLOW outputs of the DSP48E2 slice use the pattern detector to determine if the operation in the DSP48E2 slice has overflowed beyond the P[N] bit where N is between 1 and 46. The P register must be enabled while using OVERFLOW and UNDERFLOW. This is further described in the Embedded Functions section.

Embedded Functions

The embedded functions include a pre-adder, 27 x 18 multiplier, adder/subtracter/logic unit, and pattern detector logic (see Figure 2-12).

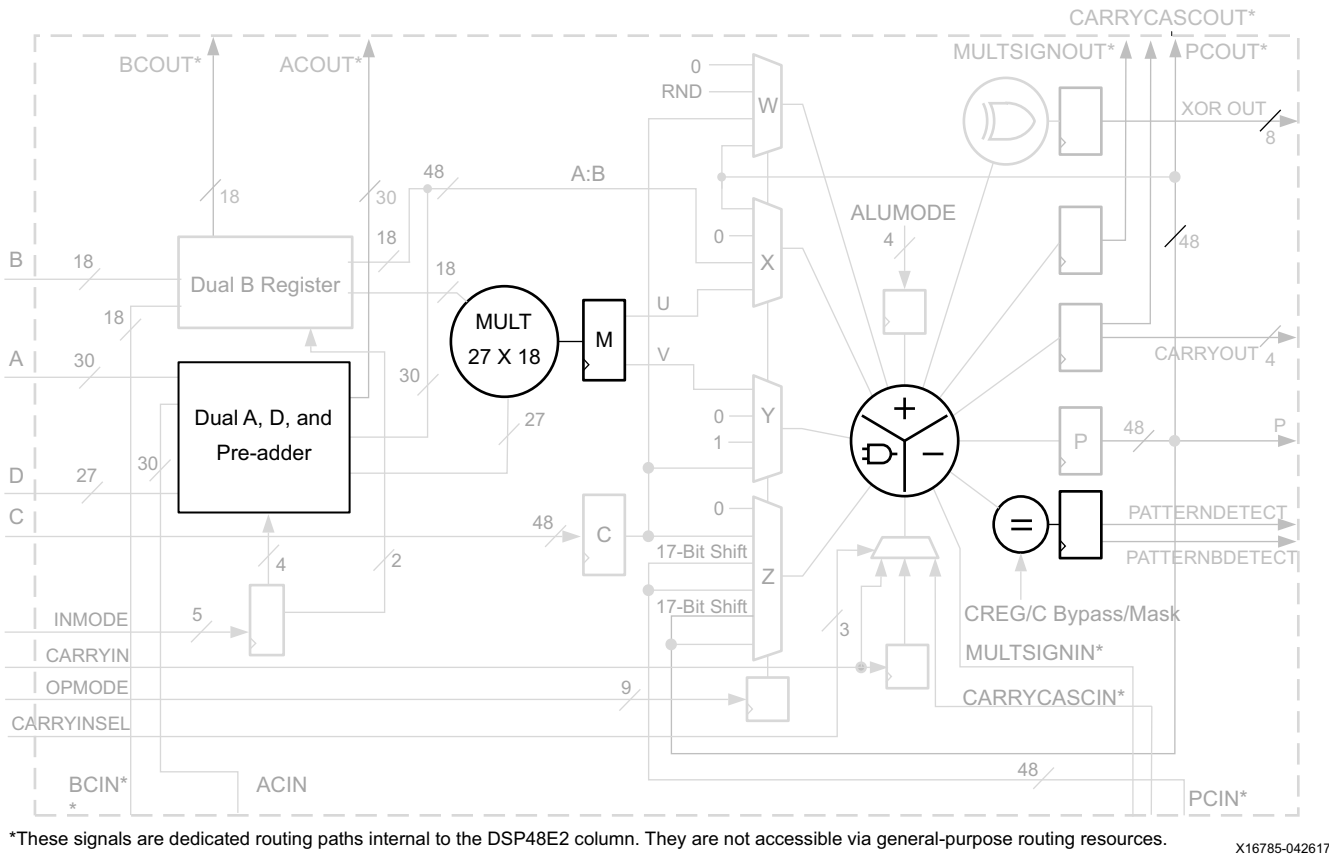


Figure 2-12: **Embedded Functions in a DSP48E2 Slice**

Pre-Adder

The DSP slice has a 27-bit pre-adder, which is inserted in the A or B register path (shown in Figure 2-12 with an expanded view in Figure 2-5, page 23). With the pre-adder, pre-additions or pre-subtractions are possible prior to feeding the multiplier. Since the pre-adder does not contain saturation logic, designers should limit input operands to 26-bit (or 17-bit for the B path) two's complement sign-extended data to avoid overflow or underflow during arithmetic operations. Optionally, the pre-adder can be bypassed, making D the new input path to the multiplier. When the D path is not used, the output of the A or B pipeline can be negated prior to driving the multiplier. There are up to 15 operating modes, including pre-adder squaring, making this pre-adder block very flexible.

In Equation 2-2, A (or B) and D are added initially through the pre-adder/subtracter. The result of the pre-adder is then multiplied against B (or A), with the result of the

multiplication being added to the C input. This equation facilitates efficient symmetric filters.

$$\text{Final Adder/Subtracter Output} = \begin{array}{l} C \pm (B \times (D \pm A) + W + C_{IN}) \\ C \pm (A \times (D \pm B) + W + C_{IN}) \\ C \pm (B \times (D \pm B) + W + C_{IN}) \\ C \pm (A \times (D \pm A) + W + C_{IN}) \\ C \pm ((D \pm B)^2 + W + C_{IN}) \\ C \pm ((D \pm A)^2 + W + C_{IN}) \end{array} \quad \text{Equation 2-2}$$

Two's Complement Multiplier

The two's complement multiplier in the DSP48E2 slice in Figure 2-12 accepts a 27-bit two's complement input and an 18-bit two's complement input. The multiplier produces two 45-bit partial products. The two partial products together give an 90-bit result at the output of the multiplier, as shown in Figure 2-13. Cascading of multipliers to achieve larger products is supported with a 17-bit, right-shifted, cascaded output bus. The right shift is used to right justify the partial products by the correct number of bits. This cascade path feeds into the Z multiplexer, which is connected to the adder/subtracter of an adjacent DSP48E2 slice. The multiplier can emulate unsigned math by setting the MSB of an input operand to zero.

Figure 2-13 shows an optional pipeline register (MREG) for the output of the multiplier. Using the register provides increased performance with an increase of one clock latency.

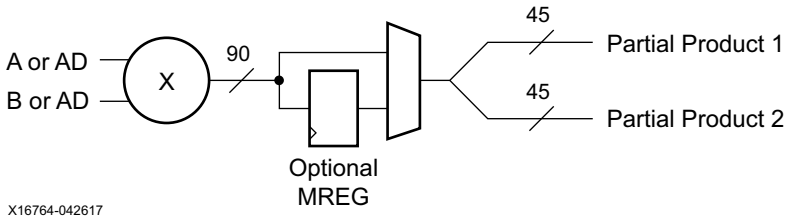


Figure 2-13: **Two's Complement Multiplier Followed by Optional MREG**

Adder/Subtracter or Logic Unit

The adder/subtracter or logic unit output is a function of control and data inputs (see Figure 2-14). The data inputs to the adder/subtracter are selected by the OPMODE and the CARRYINSEL signals. The ALUMODE signals choose the function implemented in the adder/subtracter. Thus, the OPMODE, ALUMODE, and CARRYINSEL signals together determine the functionality of the embedded adder/subtracter/logic unit. When using the logic unit, the multiplier must not be used. The values of OPMODEREG and CARRYINSELREG must be identical.

As with the input multiplexers, the OPMODE bits specify a portion of this function. The symbol ± in the table means either add or subtract and is specified by the state of the ALUMODE control signal. The symbol ":" in the table means concatenation. The outputs of the X and Y multiplexer and CIN are always added together. Refer to ALUMODE Inputs, page 30.

Two-Input Logic Unit or Three-Input XOR Special Case

The capability to perform an addition, subtraction, and simple logic functions in the DSP48E2 slice exists through use of a second-stage, four-input adder.

Table 2-10 lists the logic functions that can be implemented in the second stage of the four input adder/subtracter/logic unit. The table also lists the settings of the control signals, namely OPMODE and ALUMODE.

Setting OPMODE[3:2] to `00` selects the default `0` value at the Y multiplexer output. OPMODE[3:2] set to `10` selects all `1`s at the Y multiplexer output. OPMODE[1:0] selects the output of the X multiplexer, OPMODE[6:4] selects the output of the Z multiplexer. For two-input logic operations, OPMODE[8:7] must be set to `00` for the default all `0`s value at the W multiplexer output.

An XOR3 can be built by setting the OPMODE[3:2] to `11`, selecting the C input at the Y multiplexer output. The XOR3 is only valid for ALUMODE[3:0] = `0100`, as shown in Table 2-10.

Table 2-10: **OPMODE and ALUMODE Control Bits Select Logic Unit Outputs**

Logic Unit Mode	OPMODE[3:2]		ALUMODE[3:0]			
	3	2	3	2	1	0
X XOR Z	0	0	0	1	0	0
X XNOR Z	0	0	0	1	0	1
X XNOR Z	0	0	0	1	1	0
X XOR Z	0	0	0	1	1	1
X AND Z	0	0	1	1	0	0
X AND (NOT Z)	0	0	1	1	0	1
X NAND Z	0	0	1	1	1	0
(NOT X) OR Z	0	0	1	1	1	1
X XNOR Z	1	0	0	1	0	0
X XOR Z	1	0	0	1	0	1
X XOR Z	1	0	0	1	1	0
X XNOR Z	1	0	0	1	1	1
X OR Z	1	0	1	1	0	0

Table 2-10: **OPMODE and ALUMODE Control Bits Select Logic Unit Outputs *(Cont'd)***

Logic Unit Mode	OPMODE[3:2]		ALUMODE[3:0]			
	3	2	3	2	1	0
X OR (NOT Z)	1	0	1	1	0	1
X NOR Z	1	0	1	1	1	0
(NOT X) AND Z	1	0	1	1	1	1
X XOR Y XOR Z(1)	1	1	0	1	0	0

Notes:
1. Valid when Y multiplexer selects C input.

Single Instruction, Multiple Data (SIMD) Mode

The 48-bit adder/subtracter/accumulator can be split into smaller data segments where the internal carry propagation between segments is blocked to ensure independent operation for all segments. The adder/subtracter/accumulator can be split into four 12-bit adder/subtracter/accumulators or two 24-bit adder/subtracter/accumulators with carry out signal per segment. The SIMD mode segmentation is a static configuration as opposed to dynamic OPMODE type control (see Figure 2-14).

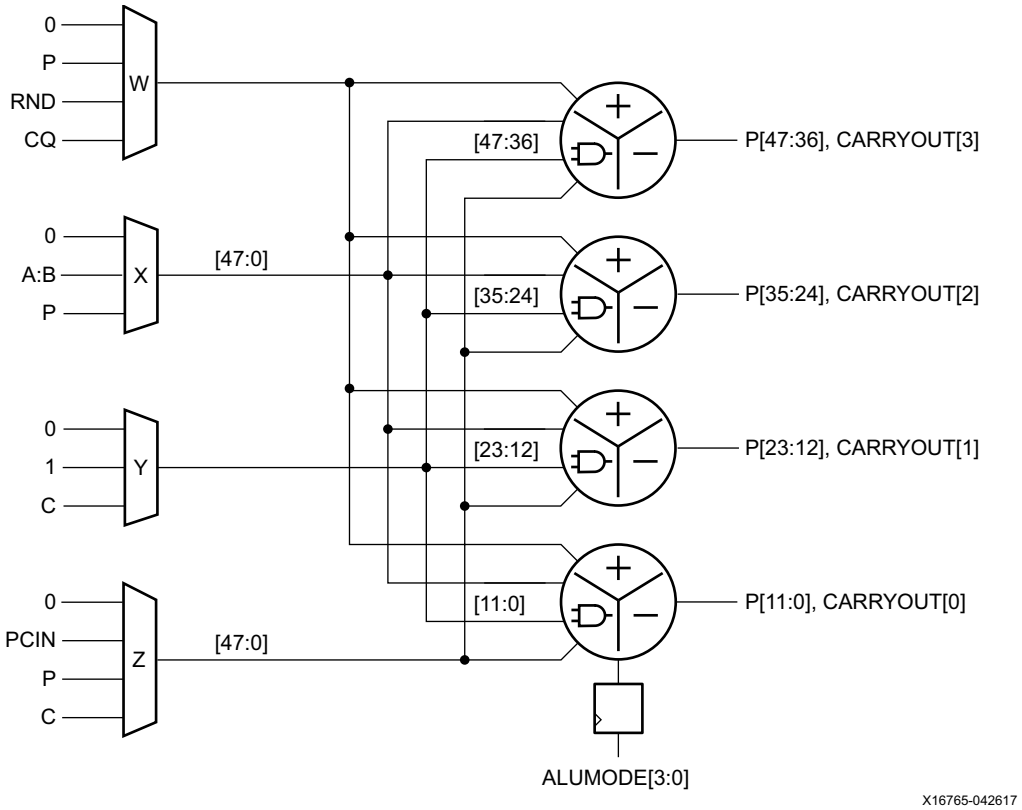


Figure 2-14: **Four 12-Bit SIMD Adder Configuration**

- Four segments of dual or ternary or quad adders with 12-bit inputs, a 12-bit output, and a carry output for each segment
- Function controlled dynamically by ALUMODE[3:0], and operand source by OPMODE[8:0]
- All four adder/subtracter/accumulators perform same function
- Two segments of dual or ternary or quad adders with 24-bit inputs, a 24-bit output, and a carry output for each segment is also available (not pictured).

The SIMD feature, shown in Figure 2-14, allows the 48-bit logic unit to be split into multiple smaller logic units. Each smaller logic unit performs the same function. This function can also be changed dynamically through the ALUMODE[3:0] and OPMODE control inputs.

Pattern Detect Logic

The pattern detector is connected to the output of the add/subtract/logic unit in the DSP48E2 slice (see Figure 2-12).

The pattern detector is best described as an equality check on the output of the adder/subtracter/logic unit that produces its result on the same cycle as the P output. There is no extra latency between the pattern detect output and the P output of the DSP48E2 slice. The use of the pattern detector leads to a moderate speed reduction due to the extra logic on the pattern detect path (see Figure 2-15).

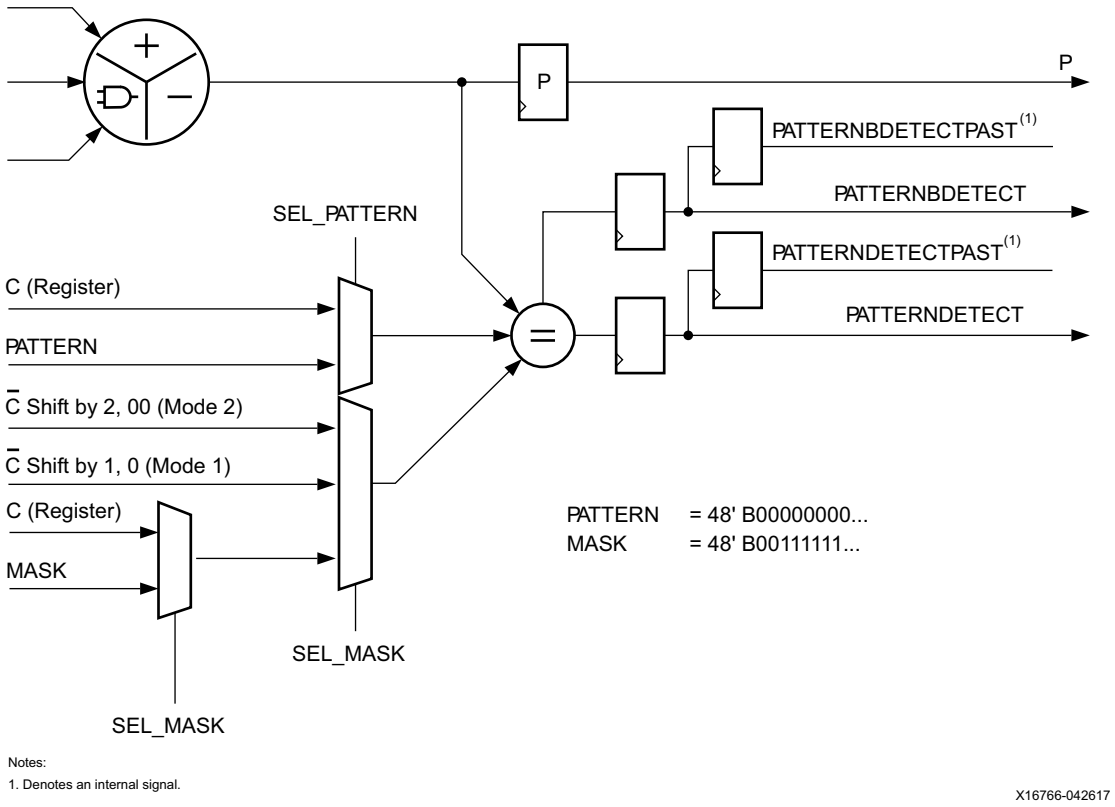


Figure 2-15: **Pattern Detector Logic**

Some of the applications that can be implemented using the pattern detector are:

- Pattern detect with optional mask
- Dynamic C input pattern match with A x B
- Overflow/underflow/saturation past P[46]
- A:B == C and dynamic pattern match, e.g., A:B OR C == 0, A:B AND C == 1
- A:B {function} C == 0
- 48-bit counter auto reset (terminal count detection) with option for CEP priority
- Detecting mid points for rounding operations

If the pattern detector is not being employed, it can be used for other creative design implementations. These include:

- Duplicating a pin (e.g., the sign bit) to reduce fanout and thus increase speed.
- Implementing a built-in inverter on one bit (e.g., the sign bit) without having to route out to the CLBs.
- Checking for sticky bits in floating point, handling special cases, or monitoring the DSP48E2 slice outputs.

- Raising a flag if a certain condition is met or if a certain condition is no longer met.

A mask field can also be used to mask out certain bit locations in the pattern detector. The pattern field and the mask field can each come from a distinct 48-bit memory cell field or from the (registered) C input.

Overflow and Underflow Logic

The discussion of overflow and underflow below applies to sequential accumulators (MACC or Adder-Accumulator) implemented in a single DSP48E2 slice. The accumulator should have at least one guard bit. When the pattern detector is set to detect a pattern equal to `00000...0` with a mask of `0011111 ...1` (default settings), the DSP48E2 slice flags overflow beyond `00111 ... 1` or underflow beyond `11000... 0`. The USE_PATTERN_DETECT attribute is set to PATDET to enable the use of the pattern detect logic. This overflow/underflow implementation uses a redundant sign bit and reduces the output bit width to 47 bits.

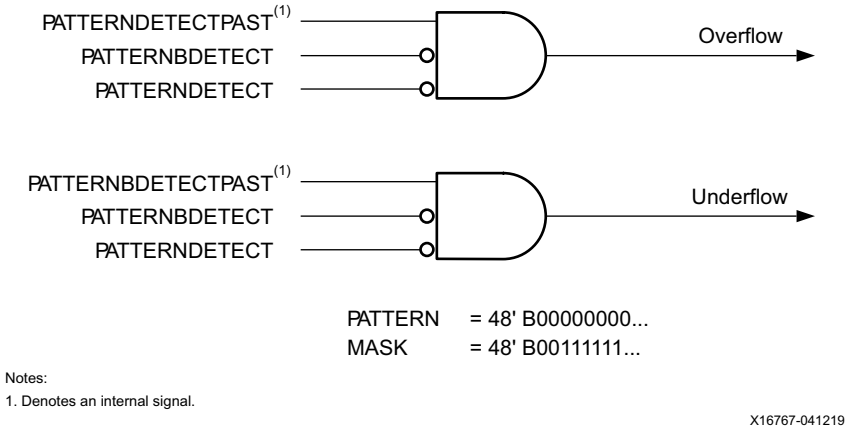


Figure 2-16: **Overflow/Underflow Logic in Pattern Detect**

By setting the mask to other values like `0000111 ...1`, the bit value P[N] at which overflow is detected can be changed. This logic supports saturation to a positive number of 2^N - 1 and a negative number of 2^N in two's complement where N is the number of 1s in the mask field.

To check overflow/underflow condition for N = 2, the following example is used:

- Mask is set to `0...11`.
- The (N) LSB bits are not considered for the comparison.
- For N = 2, the legal values (patterns) are 2^2-1 to -2^2 or 3 to –4.

See Figure 2-17 and Figure 2-18 for overflow and underflow examples, respectively.

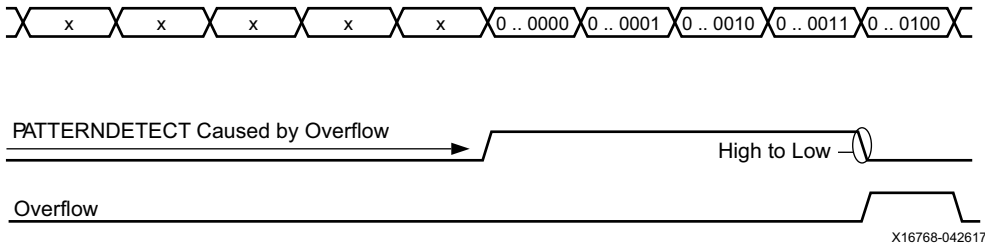


Figure 2-17: **Overflow Condition in the Pattern Detector**

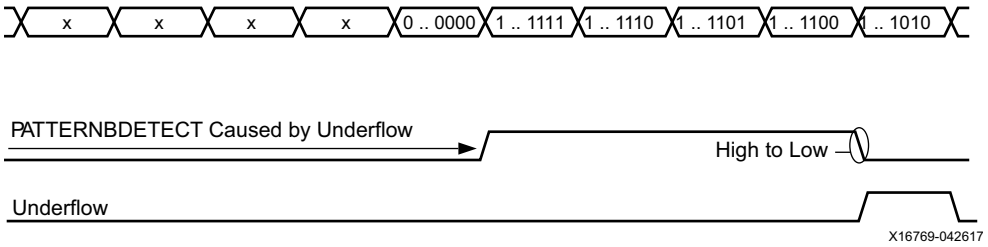


Figure 2-18: **Underflow Condition in the Pattern Detector**

- PATTERNDETECT is 1 if P == pattern or mask
- PATTERNBDETECT is a 1 if P == patternb or mask

Overflow is caused by addition when the value at the output of the adder/subtracter/logic unit goes over 3. Adding 1 to the final value of `0..0011` gives `0..0100` as the result, which causes the PATTERNDETECT output to go to `0`. When the PATTERNDETECT output goes from `1` to `0`, an overflow is flagged.

Underflow is caused by subtraction when the value goes below –4. Subtracting 1 from `1..1100` yields `1..1010` (–5), which causes the PATTERNBDETECT output to go to 0. When the PATTERNBDETECT output goes from `1` to `0`, an underflow is flagged.

Overflow and underflow are relative to the previous value (positive or negative, respectively). Overflow can result from subtraction when a value outside the valid range is subtracted from a positive value. Similarly, underflow can result from addition when a value outside the valid range is added to a negative value.

Wide XOR

A new feature in the DSP48E2 slice is the ability to perform a 96-bit wide XOR function. The XOR uses the X, Y, and Z multiplexers as inputs. The W multiplexer selects all 0s at its output. The ALU logic is used for the first stage of the wide XOR by using the proper OPMODE and ALUMODE signals as shown in Table 2-10, to implement either X XOR Z or X XOR Y XOR Z. The signals then branch out to an XOR logic tree with dedicated outputs. Multiplexers allow selection as eight 12-bit wide XOR, four 24-bit wide XOR, two 48-bit wide XOR, or one 96-bit wide XOR. See Figure 2-19. In Figure 2-19 the S[47:0} internal bus is not the P[47:0] output, it is one of the 4:2 compressor busses.

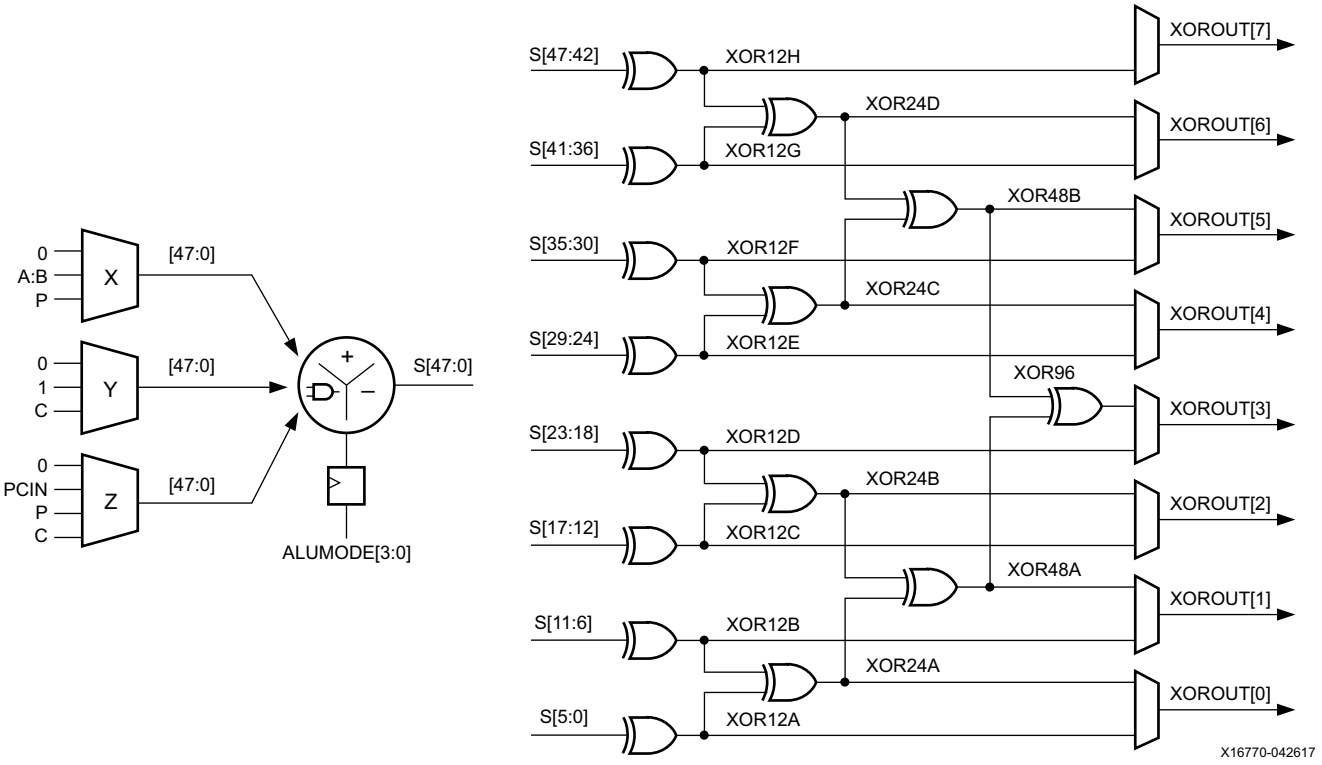


Figure 2-19: **Wide XOR Function in ALU**

The XORSIMD attribute is used to select the width of the XOR function as either 96-bits or 12-/24-/48- bits, as shown in Table 2-11.

Table 2-11: **XOR9_XOR SIMD Mode Bits**

XORSIMD Attribute	XOR Width	XOR INPUT Bits (A:B^C)	Corresponding XOROUT
XOR12	12-bits	S[5:0]	XOROUT[0]
		S[11:6]	XOROUT[1]
		S[17:12]	XOROUT[2]
		S[23:18]	XOROUT[3]
		S[29:24]	XOROUT[4]
		S[35:30]	XOROUT[5]
		S[41:36]	XOROUT[6]
		S[47:42]	XOROUT[7]

Table 2-11: **XOR9_XOR SIMD Mode Bits *(Cont'd)***

XORSIMD Attribute	XOR Width	XOR INPUT Bits (A:B^C)	Corresponding XOROUT
XOR24_48_96	24-bit	S[11:0]	XOROUT[0]
		S[23:12]	XOROUT[2]
		S[35:24]	XOROUT[4]
		S[47:36]	XOROUT[6]
	48-bit	S[23:0]	XOROUT[1]
		S[47:24]	XOROUT[5]
	96-bit	S[47:0]	XOROUT[3]

The dedicated XOR logic enables performance improvements when implementing forward error correction and cyclic redundancy checking algorithms. There is also a USE_WIDEXOR attribute to enable a power saving mode if the wide XOR function is not desired (see Table 3-3, page 53.

The first level XOR can be either XOR2 or XOR3. In both cases, ALUMODE[3:0] = `0100` for the XOR function in the ALU. When the Y multiplexer selects `0`, an XOR2 is created. When the Y multiplexer selects the C register, an XOR3 is created, supporting up to 48 XOR3 in the ALU. The third input can come from the P output or the PCIN cascade, which provides XOR-accumulate and cascade capability for even wider XOR functions.

DSP48E2 Operation Modes

Table 2-12 provides a summary of the key operation modes available in a single DSP48E2 slice, showing the largest functions available and the key resources used. Table 2-13 through Table 2-17 show similar operation modes extended to two, three, four, six, and eight slices, cascaded.

Table 2-12: **DSP48E2 Operation Modes: One Slice**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
27x18 + C MULT/MACC	17/26-bit	N/A	Used	Yes
27x18 Sequential Complex MACC	Optional	N/A	Optional	Yes
27x19 or 28x18	N/A	N/A	Used	Limited
PreAdder Squared	17-bit	N/A	Optional	Optional
SIMD Add/Sub/Counter/ACC	N/A	N/A	Used	No
48-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes
48-Bit 2:1 Bus Mux	N/A	N/A	Used	N/A
XOR96/48/24/12	N/A	N/A	Used	N/A

Table 2-12: **DSP48E2 Operation Modes: One Slice *(Cont'd)***

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
AND96/NOR96	N/A	N/A	Used	N/A
48 2-input Logic Operations	N/A	N/A	Used	N/A

Table 2-13: **DSP48E2 Operation Modes: Two Slices**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
27x18 + C MACC96	26-bit	P Used	Used	Yes
35x27 + C	26-bit	Yes	Used	Yes
35x28 or 36x27	N/A	Yes	Used	Limited
44x18 + C	17-bit	Yes	Used	Yes
44x19 or 45x18	N/A	Yes	Used	No
27x18 + C Systolic MultAdd 2-tap Filter	17/26-bit	Yes	Used	Yes
Sum of 2 PreAdder Squared	17-bit	P Used	Optional	Optional
96-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes
18-bit Barrel Shifter	N/A	Yes	N/A	N/A
34-bit Bus Shifter	N/A	Yes	N/A	N/A
48-Bit 4:1 Bus Mux	N/A	P Used	Used	N/A
XOR192/96/48/24	N/A	P Used	Used	N/A
AND144/NOR144	N/A	P Used	Used	N/A
48 3-input Logic Operations (48 XOR4)	N/A	P Used	Used	N/A

Table 2-14: **DSP48E2 Operation Modes: Three Slices**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
18x18 Complex MULT/MACC	18-bit	A Used	Used	Yes
26x17 Complex MULT/MACC	26-bit	A/B Used	Used	Yes
52x27 + C	26-bit	Yes	Used	Yes
52x28 or 53x27	N/A	Yes	Used	Limited
61x18 + C	17-bit	Yes	Used	Yes
61x19 or 62x18	N/A	Yes	Used	Limited
27x18 + C Systolic MultAdd 3-tap Filter	17/26-bit	Yes	Used	Yes
Sum of 3 PreAdder Squared	17-bit	P Used	Optional	Optional
144-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes
48-Bit 6:1 Bus Mux	N/A	P Used	Used	N/A
XOR288/144/72/36	N/A	P Used	Used	N/A

Table 2-14: **DSP48E2 Operation Modes: Three Slices *(Cont'd)***

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
AND192/NOR192	N/A	P Used	Used	N/A
48 4-input Logic Operations (48 XOR6)	N/A	P Used	Used	N/A

Table 2-15: **DSP48E2 Operation Modes: Four Slices**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
27x19 Complex MULT	N/A	P Used	Used	Yes
27x18 + C Complex MULT/MACC	17-bit	P Used	Used	Yes
44x35 + C	N/A	B/P Used	Used	Yes
44x36 or 45x35	N/A	B/P Used	Used	Yes
69x27 + C	26-bit	Yes	Used	Yes
69x28 or 70x27	N/A	Yes	Used	Limited
78x18 + C	17-bit	Yes	Used	Yes
78x19 or 79x18	N/A	Yes	Used	Limited
27x18 + C Systolic MultAdd 4-tap Filter	17/26 bit	Yes	Used	Yes
Sum of 4 PreAdder Squared	17-bit	P Used	Optional	Optional
192-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes
48-Bit 8:1 Bus Mux	N/A	P Used	Used	N/A
XOR384/192/96/48	N/A	P Used	Used	N/A
AND240/NOR240	N/A	P Used	Used	N/A
48 5-input Logic Operations (48 XOR8)	N/A	P Used	Used	N/A

Table 2-16: **DSP48E2 Operation Modes: Six Slices**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
27x18 + C Complex MACC96	17-bit	P Used	Used	Yes
61x35 + C	N/A	B/P Used	Used	Yes
61x36 or 62x35	N/A	B/P Used	Used	Yes
103x27 + C	26-bit	Yes	Used	Yes
103x28 or 104x27	N/A	Yes	Used	Limited
112x18 + C	17-bit	Yes	Used	Yes
112x19 or 113x18	N/A	Yes	Used	Limited
53x53 Unsigned	N/A	Yes	Used	No
27x18 + C Systolic MultAdd 6-tap Filter	17/26 bit	Yes	Used	Yes
Sum of 6 PreAdder Squared	17-bit	P Used	Optional	Optional
288-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes

Table 2-16: **DSP48E2 Operation Modes: Six Slices *(Cont'd)***

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
48-Bit 12:1 Bus Mux	N/A	P Used	Used	N/A
XOR576/288/144/72	N/A	P Used	Used	N/A
AND336/NOR336	N/A	P Used	Used	N/A
48 7-input Logic Operations (48 XOR12)	N/A	P Used	Used	N/A

Table 2-17: **DSP48E2 Operation Modes: Eight Slices**

Operation Mode	PreAdder	A/B/P Cascade	48-bit C Port	RND Support
35x27 + C Complex MULT	26-bit	B/P Used	Used	Yes
78x35 + C	N/A	B/P Used	Used	Yes
78x36 or 79x35	N/A	B/P Used	Used	Limited
69x44 + C	N/A	B/P Used	Used	Yes
69x45 or 70x44	N/A	B/P Used	Used	Limited
27x18 + C Systolic MultAdd 8-tap Filter	17/26-bit	Yes	Used	Yes
Sum of 8 PreAdder Squared	17-bit	P Used	Optional	Optional
384-bit Add/Sub/Counter/ACC	N/A	N/A	Used	Yes
48-Bit 16:1 Bus Mux	N/A	P Used	Used	N/A
XOR768/384/192/96	N/A	P Used	Used	N/A
AND432/NOR432	N/A	P Used	Used	N/A
48 9-input Logic Operations (48 XOR16)	N/A	P Used	Used	N/A

DSP48E2 Design Entry

Overview

Xilinx offers integrated DSP design flows tailored for the unique needs of hardware, algorithm, and traditional processor-based DSP designers, supporting all mainstream DSP design entry methods to ensure productivity.

Vivado™ Design Suite System Generator for DSP enables high-level model-based designs to be created using MathWorks MATLAB and Simulink and provides automatic fixed or floating-point hardware generation, co-simulation, and system integration into RTL or embedded systems. See *Vivado Design Suite Reference Guide: Model-Based DSP Design Using System Generator* (UG958) [Ref 4].

Vivado High-Level Synthesis [Ref 5] accelerates design implementation by enabling C, C++, and System C specifications to be directly targeted into programmable logic without the need to manually create RTL.

Vivado Design Suite includes an extensive library of device-optimized DSP IP that can be used with RTL or with System Generator or Vivado HLS to quickly assemble DSP designs that deliver high quality of results without requiring extensive FPGA design experience. DSP algorithms implemented in RTL can be verified from within DSP specific simulation environments such as MATLAB/Simulink or C/C++.

The DSP48E2 slices are inferred automatically from HDL code for most DSP functions and many arithmetic functions when using synthesis tools (check the documentation for your synthesis tools for details). Instantiation of the DSP48E2 primitive can be used to directly access specific features and provide more advanced user control.

Table 3-1: **Design Entry Methods**

Method	Support
Instantiation	Yes
Inference	Recommended
Vivado IP Catalog	Yes
Macros	Yes

DSP48E2 Slice Primitive

Figure 3-1 shows the DSP48E2 primitive. It also shows the input and output pins of the DSP48E2 slice along with the bit widths of each port. The pin definitions are in Table 3-2. See the *UltraScale Architecture Libraries Guide* (UG974) [Ref 6] and the Vivado Language Templates for instantiation examples.

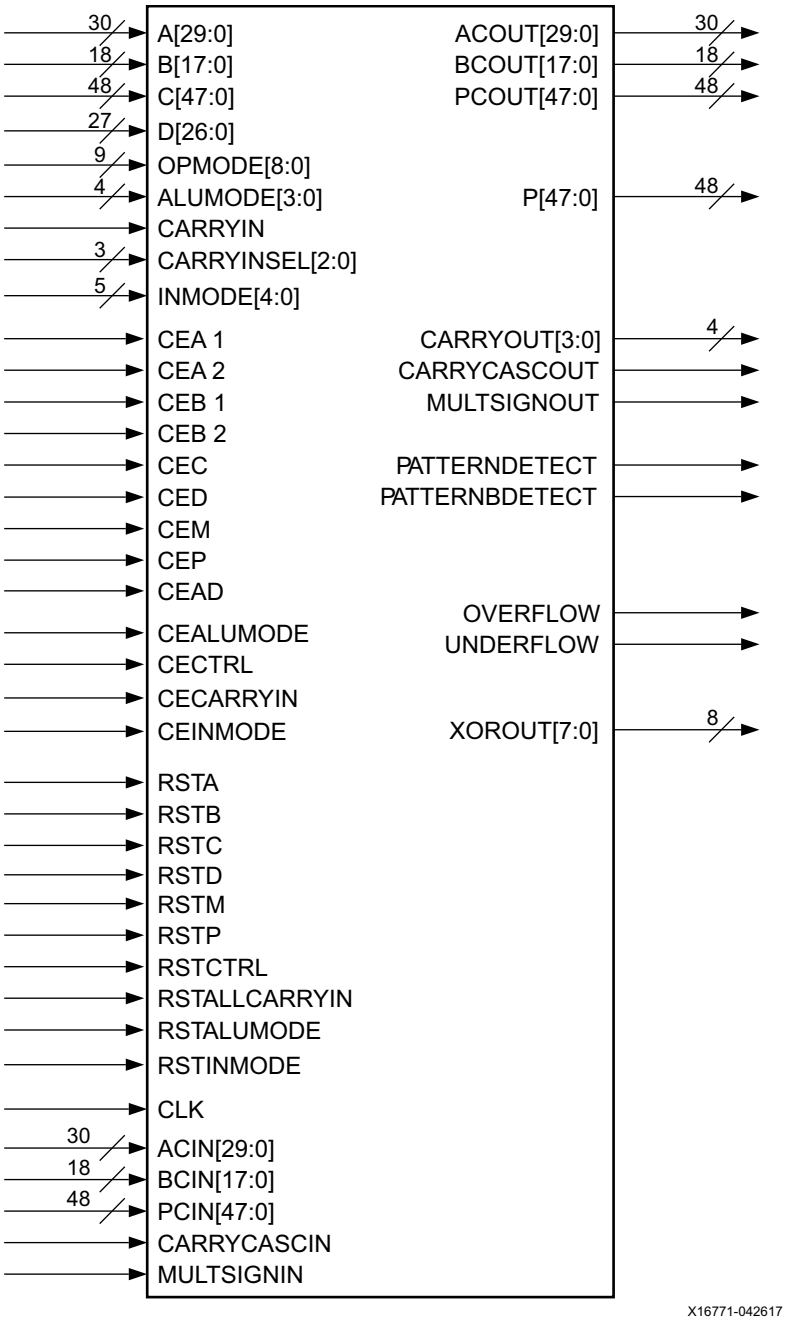


Figure 3-1: **DSP48E2 Slice Primitive**

Table 3-2: **DSP48E2 Pin Descriptions**

Name	Direction	Bit Width	Description
A[(1)]	In	30	A[26:0] is the A input of the multiplier or the pre-adder. A[29:0] are the most significant bits (MSBs) of the A:B concatenated input to the second-stage adder/subtracter or logic function.
ACIN[(2)]	In	30	Cascaded data input from ACOUT of previous DSP48E2 slice (muxed with A).
ACOUT[(2)]	Out	30	Cascaded data output to ACIN of next DSP48E2 slice.
ALUMODE	In	4	Controls the selection of the logic function in the DSP48E2 slice.
B[(1)]	In	18	The B input of the multiplier. B[17:0] are the least significant bits (LSBs) of the A:B concatenated input to the second-stage adder/subtracter or logic function.
BCIN[(2)]	In	18	Cascaded data input from BCOUT of previous DSP48E2 slice (muxed with B).
BCOUT[(2)]	Out	18	Cascaded data output to BCIN of next DSP48E2 slice.
C[(1)]	In	48	Data input to the second-stage adder/subtracter, pattern detector, or logic function.
CARRYCASCIN[(2)]	In	1	Cascaded carry input from CARRYCASCOUT of previous DSP48E2 slice.
CARRYCASCOUT[(2)]	Out	1	Cascaded carry output to CARRYCASCIN of next DSP48E2 slice. This signal is internally fed back into the CARRYINSEL multiplexer input of the same DSP48E2 slice.
CARRYIN	In	1	Carry input from the logic.
CARRYINSEL	In	3	Selects the carry source.
CARRYOUT	Out	4	4-bit carry output from each 12-bit field of the accumulate/adder/logic unit. Normal 48-bit operation uses only CARRYOUT3. SIMD operation can use four carry out bits (CARRYOUT[3:0]).
CEA1	In	1	Clock enable for the first A (input) register. A1 is only used if AREG = 2 or INMODE[0] = 1.
CEA2	In	1	Clock enable for the second A (input) register. A2 is only used if AREG = 1 or 2 and INMODE[0] = 0.
CEAD	In	1	Clock enable for the pre-adder output AD pipeline register.
CEALUMODE	In	1	Clock enable for ALUMODE (control inputs) registers.
CEB1	In	1	Clock enable for the first B (input) register. B1 is only used if BREG = 2 or INMODE[4] = 1.
CEB2	In	1	Clock enable for the second B (input) register. B2 is only used if BREG = 1 or 2 and INMODE[4] = 0.
CEC	In	1	Clock enable for the C (input) register.
CECARRYIN	In	1	Clock enable for the CARRYIN (input from the logic) register.

Table 3-2: **DSP48E2 Pin Descriptions *(Cont'd)***

Name	Direction	Bit Width	Description
CECTRL	In	1	Clock enable for the OPMODE and CARRYINSEL (control inputs) registers.
CED	In	1	Clock enable for the D (input) register.
CEINMODE	In	1	Clock enable for the INMODE control input registers.
CEM	In	1	Clock enable for the post-multiply M (pipeline) register and the internal multiply round CARRYIN register.
CEP	In	1	Clock enable for the P (output) register.
CLK	In	1	The DSP48E2 input clock, common to all internal registers and flip-flops.
D(1)	In	27	27-bit data input to the pre-adder or alternative input to the multiplier. The pre-adder implements D ± A as determined by the INMODE3 signal.
INMODE	In	5	These five control bits select the functionality of the pre-adder, the A, B, and D inputs, and the input registers. These bits should be tied to GND if unused.
MULTSIGNIN(2)	In	1	Sign of the multiplied result from the previous DSP48E2 slice for MACC extension.
MULTSIGNOUT(2)	Out	1	Sign of the multiplied result cascaded to the next DSP48E2 slice for MACC extension.
OPMODE	In	9	Controls the input to the W, X, Y, and Z multiplexers in the DSP48E2 slice.
OVERFLOW	Out	1	Overflow indicator when used with the appropriate setting of the pattern detector.
P	Out	48	Data output from second stage adder/subtracter or logic function.
PATTERNBDETECT	Out	1	Match indicator between P[47:0] and the complement of the pattern.
PATTERNDETECT	Out	1	Match indicator between P[47:0] and the pattern.
PCIN(2)	In	48	Cascaded data input from PCOUT of previous DSP48E2 slice to adder.
PCOUT(2)	Out	48	Cascaded data output to PCIN of next DSP48E2 slice.
RSTA	In	1	Reset for both A (input) registers.
RSTALLCARRYIN	In	1	Reset for the Carry (internal path) and the CARRYIN register.
RSTALUMODE	In	1	Reset for ALUMODE (control inputs) registers.
RSTB	In	1	Reset for both B (input) registers.
RSTC	In	1	Reset for the C (input) register.
RSTCTRL	In	1	Reset for OPMODE and CARRYINSEL (control inputs) registers.

Table 3-2: **DSP48E2 Pin Descriptions *(Cont'd)***

Name	Direction	Bit Width	Description
RSTD	In	1	Reset for the D (input) register and for the pre-adder (output) AD pipeline register.
RSTINMODE	In	1	Reset for the INMODE (control input) registers.
RSTM	In	1	Reset for the M (pipeline) register.
RSTP	In	1	Reset for the P (output) register.
UNDERFLOW	Out	1	Underflow indicator when used with the appropriate setting of the pattern detector.
XOROUT	Out	8	Wide XOR outputs, based on XORSIMD attribute. See Figure 2-19.

Notes:

1. When these data pins are not used and to reduce leakage power dissipation, the data pin input signals must be tied High, the input register must be selected, and the CE and RST input control signals must be tied Low. An example of unused C input recommended settings would be setting C[47:0] = all ones, CREG = 1, CEC = 0, and RSTC = 0.
2. These signals are dedicated routing paths internal to the DSP48E2 column. They are not accessible via general routing resources.
3. All signals are active High.

DSP48E2 Slice Attributes and Registers

The synthesis attributes for the DSP48E2 slice are described in this section. The attributes call out pipeline registers in the control and datapaths. The value of the attribute sets the number of pipeline registers. See Table 3-3 for the attribute descriptions and Table 3-4 for the internal register descriptions.

Table 3-3: **Attribute Setting Description**

Attribute Name	Settings (Default)	Attribute Description
Register Control Attributes		
ACASCREG	0, 1, 2 (1)	In conjunction with AREG, selects the number of A input registers on the A cascade path, ACOUT. This attribute must be equal to or one less than the AREG value: AREG = 0: ACASCREG must be 0 AREG = 1: ACASCREG must be 1 AREG = 2: ACASCREG can be 1 or 2
ADREG	0, 1 (1)	Selects the number of AD pipeline registers.
ALUMODEREG	0, 1 (1)	Selects the number of ALUMODE input registers.

Table 3-3: **Attribute Setting Description *(Cont'd)***

Attribute Name	Settings (Default)	Attribute Description
AREG	0, 1, 2 (1)	Selects the number of A input registers to the X multiplexer to the ALU. When 1 is selected, the A2 register is used. Used in conjunction with INMODE[0] for the multiplier and ACASCREG for the cascade path.
BCASCREG	0, 1, 2 (1)	In conjunction with BREG, selects the number of B input registers on the B cascade path, BCOUT. This attribute must be equal to or one less than the BREG value: BREG = 0: BCASCREG must be 0 BREG = 1: BCASCREG must be 1 BREG = 2: BCASCREG can be 1 or 2
BREG	0, 1, 2 (1)	Selects the number of B input registers to the X multiplexer to the ALU. When 1 is selected, the B2 register is used. Used in conjunction with INMODE[4] for the multiplier and BCASCREG for the cascade path.
CARRYINREG	0, 1 (1)	Selects the number of CARRYIN input registers.
CARRYINSELREG	0, 1 (1)	Selects the number of CARRYINSEL input registers.
CREG	0, 1 (1)	Selects the number of C input registers.
DREG	0, 1 (1)	Selects the number of D input registers.
INMODEREG	0, 1 (1)	Selects the number of INMODE input registers.
MREG	0, 1 (1)	Selects the number of M pipeline registers.
OPMODEREG	0, 1 (1)	Selects the number of OPMODE input registers.
PREG	0, 1 (1)	Selects the number of P output registers (also used by CARRYOUT/ PATTERNDETECT/ PATTERNBDETECT / OVERFLOW / UNDERFLOW / XOROUT / CARRYCASCOUT/ MULTSIGNOUT, and PCOUT.).
Feature Control Attributes		
A_INPUT	DIRECT, CASCADE (DIRECT)	Selects the A input between parallel input (DIRECT) or the cascaded input from the previous slice (CASCADE).
B_INPUT	DIRECT, CASCADE (DIRECT)	Selects the B input between parallel input (DIRECT) or the cascaded input from the previous slice (CASCADE).
PREADDINSEL	A, B (A)	Selects the input to be added with D in the preadder.
AMULTSEL	A, AD (A)	Selects the input to the 27-bit A input of the multiplier. In the 7 series primitive DSP48E1 the attribute is called USE_DPORT, but has been renamed due to new pre-adder flexibility enhancements (default AMULTSEL = A is equivalent to USE_DPORT=FALSE).
BMULTSEL	B, AD (B)	Selects the input to the 18-bit B input of the multiplier.

Table 3-3: **Attribute Setting Description *(Cont'd)***

Attribute Name	Settings (Default)	Attribute Description
USE_MULT	NONE, MULTIPLY, DYNAMIC (MULTIPLY)	Selects usage of the multiplier. Set to NONE to save power when using only the Adder/Logic Unit. The DYNAMIC setting indicates that the user is switching between A*B and A:B operations on the fly and therefore needs to get the worst-case timing of the two paths.
RND	48-bit field (00...00)	This 48-bit value is used as the Rounding Constant into the WMUX.
USE_SIMD	ONE48, TWO24, FOUR12 (ONE48)	Selects the mode of operation for the adder/subtracter. The attribute setting can be one 48-bit adder mode (ONE48), two 24-bit adder mode (TWO24), or four 12-bit adder mode (FOUR12). Selecting ONE48 mode is compatible with Virtex-6 devices DSP48 operation and is not actually a true SIMD mode. Typical Multiply-Add operations are supported when the mode is set to ONE48. When either TWO24 or FOUR12 mode is selected, the multiplier must not be used, and USE_MULT must be set to NONE.
USE_WIDEXOR	TRUE, FALSE (FALSE)	Determines whether the Wide XOR is used or not used.
XORSIMD	XOR12, XOR24_48_96 (XOR24_48_96)	Selects the mode of operation for the Wide XOR. The attribute setting can be one 96-bit, two 48-bit and four 24-bit XOR mode (XOR24_48_96), or eight 12-bit XOR mode (XOR12).
Pattern Detector Attributes		
AUTORESET_PATDET	NO_RESET, RESET_MATCH, RESET_NOT_MATCH (NO_RESET)	Automatically resets the P Register (accumulated value or counter value) on the next clock cycle, if a pattern detect event has occurred on this clock cycle. The RESET_MATCH and RESET_NOT_MATCH settings distinguish between whether the DSP48E2 slice should cause an auto reset of the P Register on the next cycle: • if the pattern is matched or • whenever the pattern is not matched on the current cycle but was matched on the previous clock cycle
AUTORESET_PRIORITY	RESET, CEP (RESET)	When using the AUTORESET_PATDET feature, if the attribute is set to CEP, the P Register will only reset pending the value of the clock enable. Otherwise, the autoreset will have precedence.
MASK	48-bit field (0011...11)	This 48-bit value is used to mask out certain bits during a pattern detection. When a MASK bit is set to 1, the corresponding pattern bit is ignored. When a MASK bit is set to 0, the pattern bit is compared.
PATTERN	48-bit field (00...00)	This 48-bit value is used in the pattern detector.

Table 3-3: **Attribute Setting Description *(Cont'd)***

Attribute Name	Settings (Default)	Attribute Description
SEL_MASK	MASK, C, ROUNDING_MODE1, ROUNDING_MODE2 (MASK)	Selects the mask to be used for the pattern detector. The C and MASK settings are for standard uses of the pattern detector (counter, overflow detection, etc.). ROUNDING_MODE1 (C-bar left shifted by 1) and ROUNDING_MODE2 (C-bar left shifted by 2) select special masks based off of the optionally registered C input. These rounding modes can be used to implement convergent rounding in the DSP48E2 slice using the pattern detector.
SEL_PATTERN	PATTERN, C (PATTERN)	Selects the input source for the pattern field. The input source can either be a 48-bit dynamic "C" input or a 48-bit static attribute field.
USE_PATTERN_DETECT	NO_PATDET, PATDET (NO_PATDET)	Selects whether the pattern detector and related features are used (PATDET) or not used (NO_PATDET). This attribute is used for speed specification and Simulation Model purposes only.
Optional Inversion Attributes		
IS_ALUMODE_INVERTED	4-bit binary (4'b0000)	Indicates if the ALUMODE[3:0] is optionally inverted within the DSP slice. The default 4'b0000 indicates that all bits of the ALUMODE bus are not inverted. Each attribute bit controls its respective bit of the ALUMODE bus.
IS_CARRYIN_INVERTED	1-bit binary (1'b0)	Indicates if the CARRYIN is optionally inverted within the DSP slice. The default 1'b0 indicates that the CARRYIN is not inverted.
IS_CLK_INVERTED	1-bit binary (1'b0)	Indicates if the CLK is optionally inverted within the DSP slice. The default 1'b0 indicates that the CLK is not inverted.
IS_INMODE_INVERTED	5-bit binary (5'b00000)	Indicates if the INMODE[4:0] is optionally inverted within the DSP slice. The default 5'b00000 indicates that all the bits of the INMODE bus are not inverted. Each Attribute bit controls its respective bit of the INMODE bus.
IS_OPMODE_INVERTED	9-bit binary (9'b000000000)	Indicates if the OPMODE[8:0] is optionally inverted within the DSP slice. The default 9'b000000000 indicates that all the bits of the OPMODE bus are not inverted. Each attribute bit controls its respective bit of the OPMODE bus.
IS_RSTA_INVERTED	1-bit binary (1'b0)	Indicates if the RSTA is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTA is not inverted.
IS_RSTALLCARRYIN_INVERTED	1-bit binary (1'b0)	Indicates if the RSTALLCARRYIN is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTALLCARRYIN is not inverted.

Table 3-3: **Attribute Setting Description *(Cont'd)***

Attribute Name	Settings (Default)	Attribute Description
IS_RSTALUMODE_INVERTED	1-bit binary (1'b0)	Indicates if the RSTALUMODE is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTALUMODE is not inverted.
IS_RSTB_INVERTED	1-bit binary (1'b0)	Indicates if the RSTB is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTB is not inverted.
IS_RSTC_INVERTED	1-bit binary (1'b0)	Indicates if the RSTC is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTC is not inverted.
IS_RSTCTRL_INVERTED	1-bit binary (1'b0)	Indicates if the RSTCTRL is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTCTRL is not inverted.
IS_RSTD_INVERTED	1-bit binary (1'b0)	Indicates if the RSTD is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTD is not inverted.
IS_RSTINMODE_INVERTED	1-bit binary (1'b0)	Indicates if the RSTINMODE is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTINMODE is not inverted.
IS_RSTM_INVERTED	1-bit binary (1'b0)	Indicates if the RSTM is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTM is not inverted.
IS_RSTP_INVERTED	1-bit binary (1'b0)	Indicates if the RSTP is optionally inverted within the DSP slice. The default 1'b0 indicates that the RSTP is not inverted.

Table 3-4: **Internal Register Descriptions**

Register	Description and Associated Attribute
2-Deep A Registers	These two optional registers for the A input are selected by AREG, enabled by CEA1 and CEA2, respectively, and reset synchronously by RSTA.
2-Deep B Registers	These two optional registers for the B input are selected by BREG, enabled by CEB1 and CEB2, respectively, and reset synchronously by RSTB.
AD Register	This register for the optional pre-adder result is selected by ADREG, enabled by CEAD, and reset synchronously by RSTD.
ALUMODE Register	This optional pipeline register for the ALUMODE control signal is selected by ALUMODEREG, enabled by CEALUMODE, and reset synchronously by RSTALUMODE.
C Register	This optional register for the C input is selected by CREG, enabled by CEC, and reset synchronously by RSTC.
CARRYIN Register	This optional pipeline register for the CARRYIN control signal is selected by CARRYINREG, enabled by CECARRYIN, and reset synchronously by RSTALLCARRYIN.
CARRYINSEL Register	This optional pipeline register for the CARRYINSEL control signal is selected by CARRYINSELREG, enabled by CECTRL, and reset synchronously by RSTCTRL.

Table 3-4: **Internal Register Descriptions *(Cont'd)***

Register	Description and Associated Attribute
D Register	This register for the optional D pre-adder input is selected by DREG, enabled by CED, and reset synchronously by RSTD.
INMODE Register	This 5-bit register selects the pre-adder and its mode, and the sign and the source of the A and B registers feeding the multiplier. This register is selected by INMODEREG, enabled by CEINMODE, and reset synchronously by RSTINMODE.
Internal Multiplier Carry Register	This optional pipeline register for the Internal Carry signal (used for Multiply Symmetric Rounding Only) is enabled by CEM and reset synchronously by RSTM.
M Register	This optional pipeline register for the output of the multiplier consists of two 45-bit partial products. These two partial products are input into the X and Y multiplexers and finally into the adder/subtracter to create the product output. The M register is selected by MREG, enabled by CEM, and reset synchronously by RSTM.
OPMODE Register	This optional pipeline register for the OPMODE control signal is selected by OPMODEREG, enabled by CECTRL, and reset synchronously by RSTCTRL.
Output Registers	These optional registers for the P, OVERFLOW, UNDERFLOW, PATTERNDETECT, PATTERNBDETECT, CARRYOUT, and XOROUT outputs are selected by PREG, enabled by CEP, and reset synchronously by RSTP. The same registers also clock out PCOUT, CARRYCASCOUT, and MULTSIGNOUT, which are the cascade outputs to the next DSP48E2 slice.

DSP48E2 Usage Guidelines

Overview

This chapter describes some design features and techniques to use to achieve higher performance, lower power, and lower resources in a particular design.

This chapter contains the following sections:

Designing for Performance

To achieve maximum performance when using the DSP48E2 slice, the design needs to be fully pipelined. For multiplier-based designs, the DSP48E2 slice requires a three-stage pipeline. For non-multiplier-based designs, a two-stage pipeline should be used. See the UltraScale device data sheets [Ref 2] for performance details.

IMPORTANT: *If latency is important in the design and only one or two registers can be used within the DSP48E2 slice, always use the M register.*

Designing for Power

The USE_MULT attribute selects usage of the multiplier. This attribute should be set to NONE to save power when using only the Adder/Logic Unit. Functions implemented in the DSP48E2 slice use less power than those implemented in fabric. Using the cascade paths within the DSP48E2 slice instead of fabric routing is another way to reduce power. A multiplier with the M register turned on uses less power than one where the M register is not used. For operands less than 27 x 18, fabric power can be reduced by placing operands into the MSBs and zero padding unused LSBs. If one of the multiplier input operands is a constant, then assign this to the B input to reduce Booth encoding logic power dissipation.

Adder Tree vs. Adder Cascade

Adder Tree

In typical direct form FIR filters, an input stream of samples is presented to one input of the multipliers in the DSP48E2 slices. The coefficients supply the other input to the multipliers. An adder tree is used to combine the outputs from many multipliers as shown in Figure 4-1.

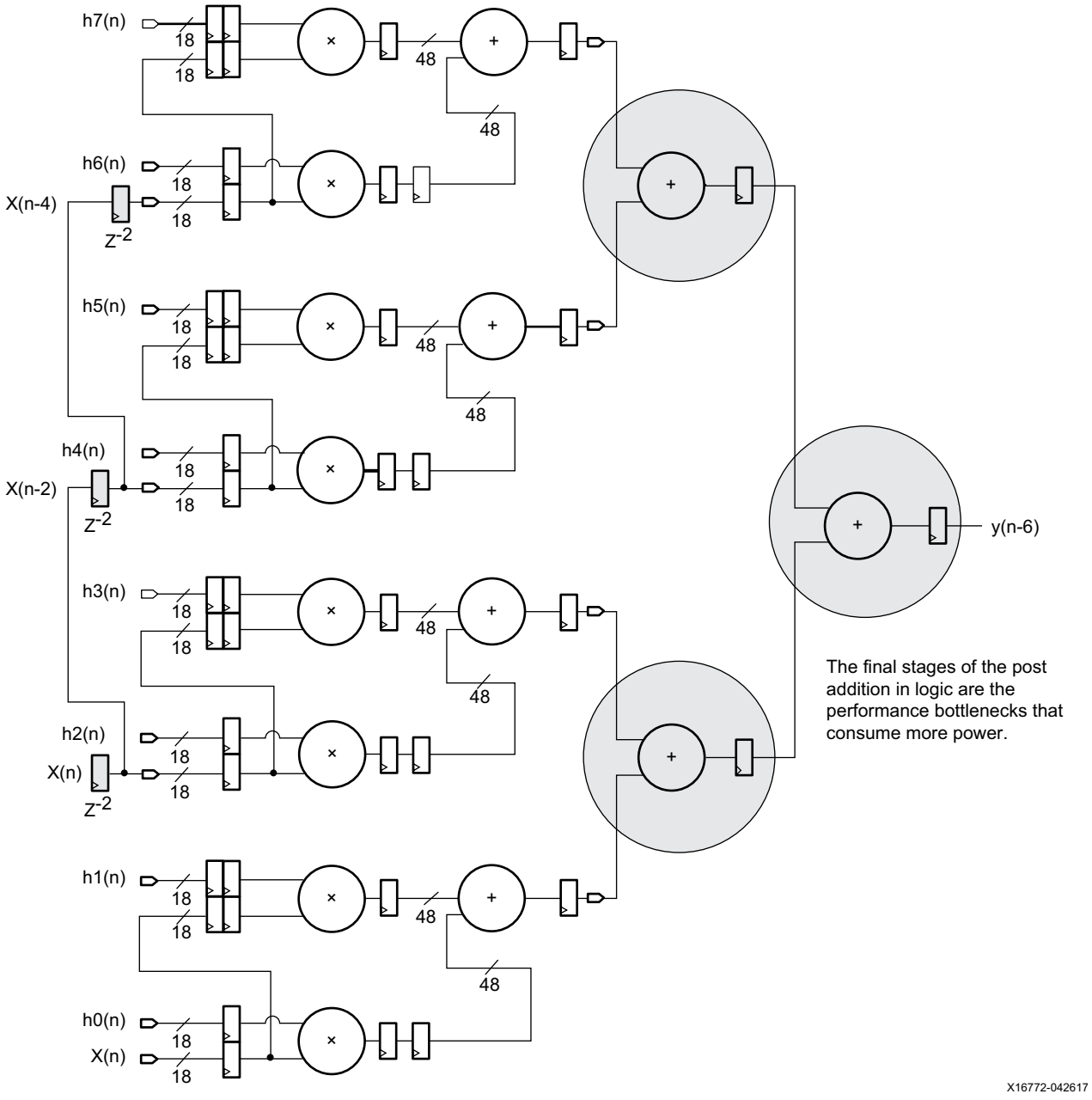


Figure 4-1: **Traditional FIR Filter Adder Tree**

In the traditional approach, the fabric adders are usually the performance bottleneck. The number of adders needed and the associated routing depends on the size of the filter. The depth of the adder tree scales as the $\log_2$ of the number of taps in the filter. Using the adder tree structure shown in Figure 4-1 could also increase the cost, logic resources, and power.

The UltraScale™ architecture CLB allows the use of both the 6LUT and the carry chain in the same slice to build an efficient ternary adder. The 6LUT in the CLB functions as a dual 5LUT. The 5LUT is used as a 3:2 compressor to add three input values to produce two output values. The 3:2 compressor is shown in Figure 4-2.

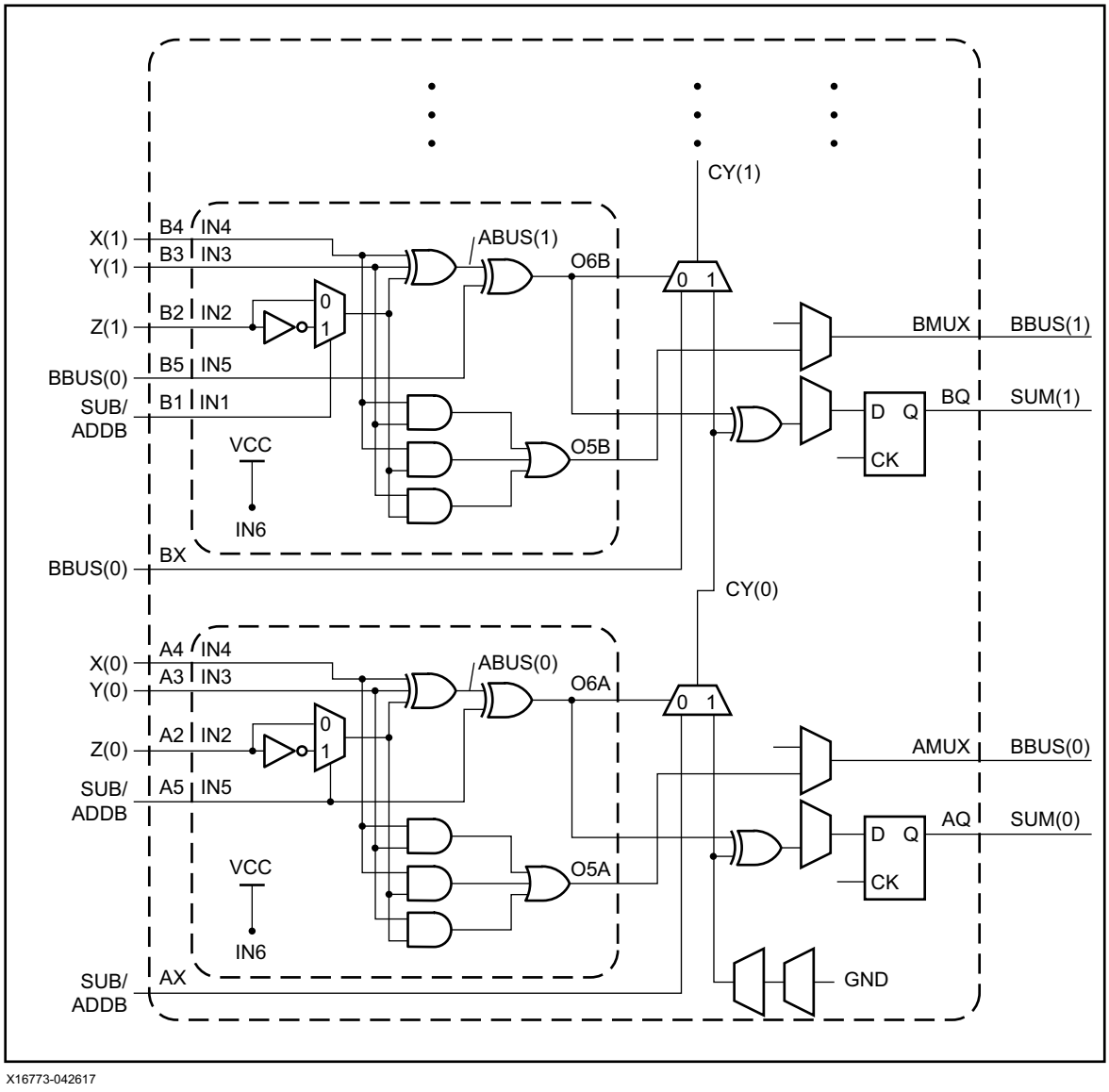


Figure 4-2: **Ternary Add/Sub with 3:2 Compressor**

The dual 5LUT configured as a 3:2 compressor in combination with the 2-input carry cascade adder adds three N-bit numbers to produce one N+2 bit output, as shown in Figure 4-3, by vertically stacking the required number of slices.

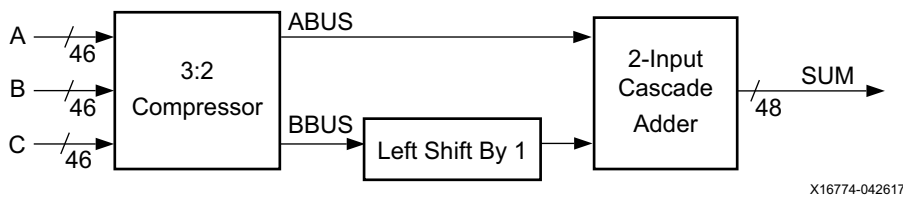


Figure 4-3: **Three-Input Adder**

The 3:1 adder shown in Figure 4-3 is used as a building block for larger adder trees. Depending on the number of inputs to be added, a 5:3 or a 6:3 compressor is also built in CLB logic using multiple 5LUTs or 6LUTs. The serial combination of 6:3 compressor, along with two DSP48E2 slices, adds six operands together to produce one output, as shown in Figure 4-4. The LSB bits of the first DSP48E2 slice that are left open due to left shift of the Y and Z buses should be tied to zero. The last DSP48E2 slice uses 2-deep A:B input registers to align (pipeline matching) the X bus to the output of the first DSP48E2 slice. Multiple levels of 6:3 compressors can be used to expand the number of input buses.

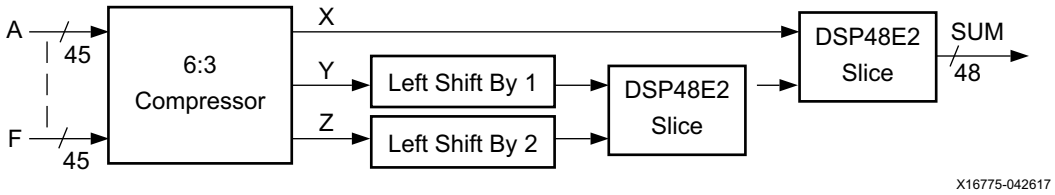


Figure 4-4: **Six-Input Adder**

The logic equations for the X, Y, and Z buses in Figure 4-4 are listed here:

$$X(n) = A(n)\ XOR\ B(n)\ XOR\ C(n)\ XOR\ D(n)\ XOR\ E(n)\ XOR\ F(n) \quad \text{Equation 4-1}$$

$$Y(n) = A(n)B(n)\ XOR\ A(n)C(n)\ XOR\ A(n)D(n)\ XOR\ A(n)E(n)\ XOR\ A(n)F(n)\ XOR\ B(n)C(n)\ XOR\ B(n)D(n)\ XOR\ B(n)E(n)\ XOR\ B(n)F(n)\ XOR\ C(n)D(n)\ XOR\ C(n)E(n)\ XOR\ C(n)F(n)\ XOR\ D(n)E(n)\ XOR\ D(n)F(n)\ XOR\ E(n)F(n) \quad \text{Equation 4-2}$$

$$Z(n) = A(n)B(n)C(n)D(n)\ OR\ A(n)B(n)C(n)E(n)\ OR\ A(n)B(n)C(n)F(n)\ OR\ A(n)B(n)D(n)E(n)\ OR\ A(n)B(n)D(n)F(n)\ OR\ A(n)B(n)E(n)F(n)\ OR\ A(n)C(n)D(n)E(n)\ OR\ A(n)C(n)D(n)F(n)\ OR\ A(n)C(n)E(n)F(n)\ OR\ A(n)D(n)E(n)F(n)\ OR\ B(n)C(n)D(n)E(n)\ OR\ B(n)C(n)D(n)F(n)\ OR\ B(n)C(n)E(n)F(n)\ OR\ B(n)D(n)E(n)F(n)\ OR\ C(n)D(n)E(n)F(n) \quad \text{Equation 4-3}$$

The compressor elements and cascade adder can be arranged like a tree in order to build larger adders. The last add stage should be implemented in the DSP48E2 slice. Pipeline registers should be added as needed to meet timing requirements of the design. These adders can have higher area and/or power than the adder cascade.

Adder Cascade

The adder cascade implementation accomplishes the post addition process with minimal silicon resources by using the cascade path within the DSP48E2 slice. This involves computing the additive result incrementally, utilizing a cascaded approach as illustrated in Figure 4-5.

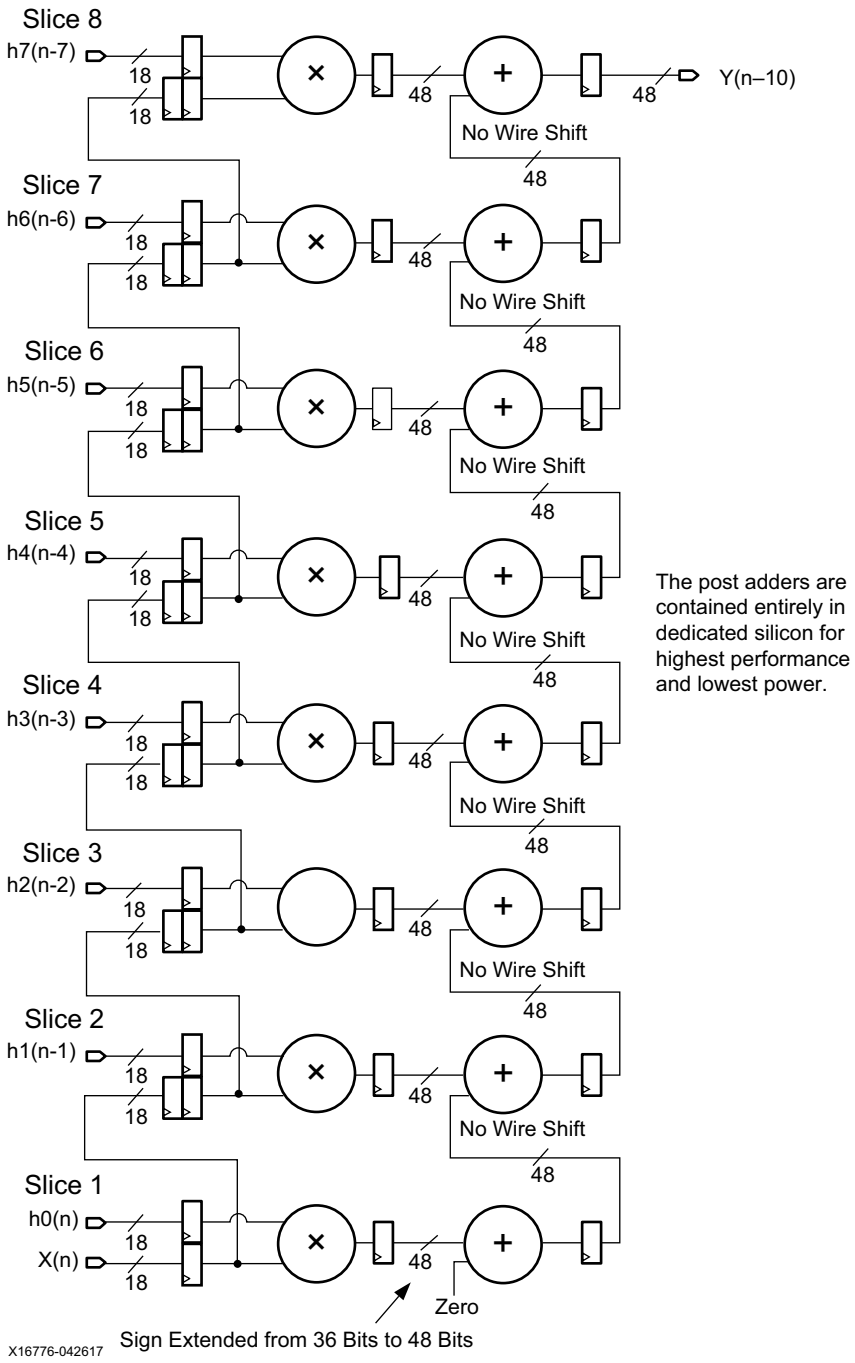


Figure 4-5: **Adder Cascade**

It is important to balance the delay of the input sample and the coefficients in the cascaded adder to achieve the correct results. The coefficients are staggered in time (wave coefficients).

Connecting DSP48E2 Slices across Columns

Using the cascade paths to implement adders significantly improves power consumption and speed. The maximum number of cascades in a path is limited only by the total number of DSP48E2 slices in one column in the chip. See Device Resources, page 10.

IMPORTANT: *The height of the DSP column can differ between devices and should be considered while porting designs.*

Spanning columns is possible by taking the bus output from the top of one DSP column and adding CLB slice pipeline registers to route this bus to the C input of the bottom DSP48E2 slice of the adjacent DSP column. Alignment of input operands is also necessary to span multiple DSP columns.

Time Multiplexing the DSP48E2 Slice

The high-speed math elements in the DSP48E2 slice enable you to use time multiplexing in DSP designs. Time multiplexing is the process of implementing more than one function within a single DSP48E2 slice at different instances of time. Time multiplexing can be done easily for designs with low sample rates. The calculation to determine the number of functions (N) that can be implemented in one single DSP48E2 slice is shown in Equation 4-4:

$$N * \text{channel frequency} \leq \text{maximum frequency of the DSP48E2 slice} \qquad \text{Equation 4-4}$$

These time-multiplexed DSP designs have optional pipelining that permits aggregate multichannel sample rates of up to 500 million samples per second. Implementing a time-multiplexed design using the DSP48E2 slice results in reduced resource utilization and reduced power.

The DSP48E2 slice contains the basic elements of classic FIR filters: a multiplier followed by an adder, delay or pipeline registers, and the ability to cascade an input stream (B bus) and an output stream (P bus) without exiting to a general CLB slice.

Multichannel filtering can be viewed as time-multiplexed, single-channel filters. In a typical multichannel filtering scenario, multiple input channels are filtered using a separate digital filter for each channel. Due to the high performance of the DSP48E2 slice, a single digital filter can be used to filter all eight input channels by clocking the single filter with an 8x clock. This implementation uses 1/8th of the total resource as compared to implementing each channel separately.

Miscellaneous Notes and Suggestions

- Implement small multiplies (e.g., 4 x 4 multiplies) and small bit-width adders and counters using the CLB logic LUTs and carry chain. If the design has a large number of small add operations and/or counters, take advantage of the SIMD mode and implement the operation in the DSP48E2 slice. Factor of 2x area and power savings occur, when compared to using interconnect logic, whenever input registers are also folded into the DSP48E2 slice for SIMD mode functions.
- Always sign extend the input operands when implementing smaller bit width functions. For lower fabric power, push operands into MSBs and ground (GND) LSBs.
- While cascading different DSP48E2 slices, match the pipestages of the different signal paths.
- Implement a count-up-by-one counter within the DSP48E2 slice using the CARRYIN input. A count-by-N or variable-bit counter can use the C or A:B inputs.
- DSP48E2 counters can be used to implement control logic that runs at maximum speed.
- Use SRL16s/SRL32s in the CLB and block RAM to store filter coefficients or act as a register file or memory elements in conjunction with the DSP48E2 slice. The bit pitch of the input bits is designed to pitch match the CLB and block RAM.
- The block RAM can also be used as a fast, finite state machine to drive the control logic for the DSP design.
- The DSP48E2 slice can also be used in conjunction with a processor, e.g., MicroBlaze™ or PicoBlaze™ processors, for hardware acceleration of processor functions.
- Use a pipeline register at the output of an SRL16 or block RAM before connecting it to the input of the DSP48E2 slice. This ensures the best performance of input operands feeding the DSP48E2 slice.
- The register at the output of the SRL16 in the slice has a reset pin and a clock-enable pin. To reset the SRL16, a zero is input into the SRL16 for 16 clock cycles while holding the reset of the output register High. This capability is particularly useful in implementing filters where the SRL16s are used to store the data inputs.

Pre-Adder Block Applications

DSP48E2 slice users can benefit from the pre-adder in several applications ranging from wireless applications (for example, in algorithms as in the Long-Term Evolution specification), in generic filtering (FIR and IIR), in video processing (for example, alpha blending), and many others. The most common use for the pre-adder is to pre-add corresponding values of a symmetric FIR filter tap delay line.

Figure 4-6 illustrates how the pre-adder (shaded in gray) can be used in an 8-tap even symmetric systolic FIR design.

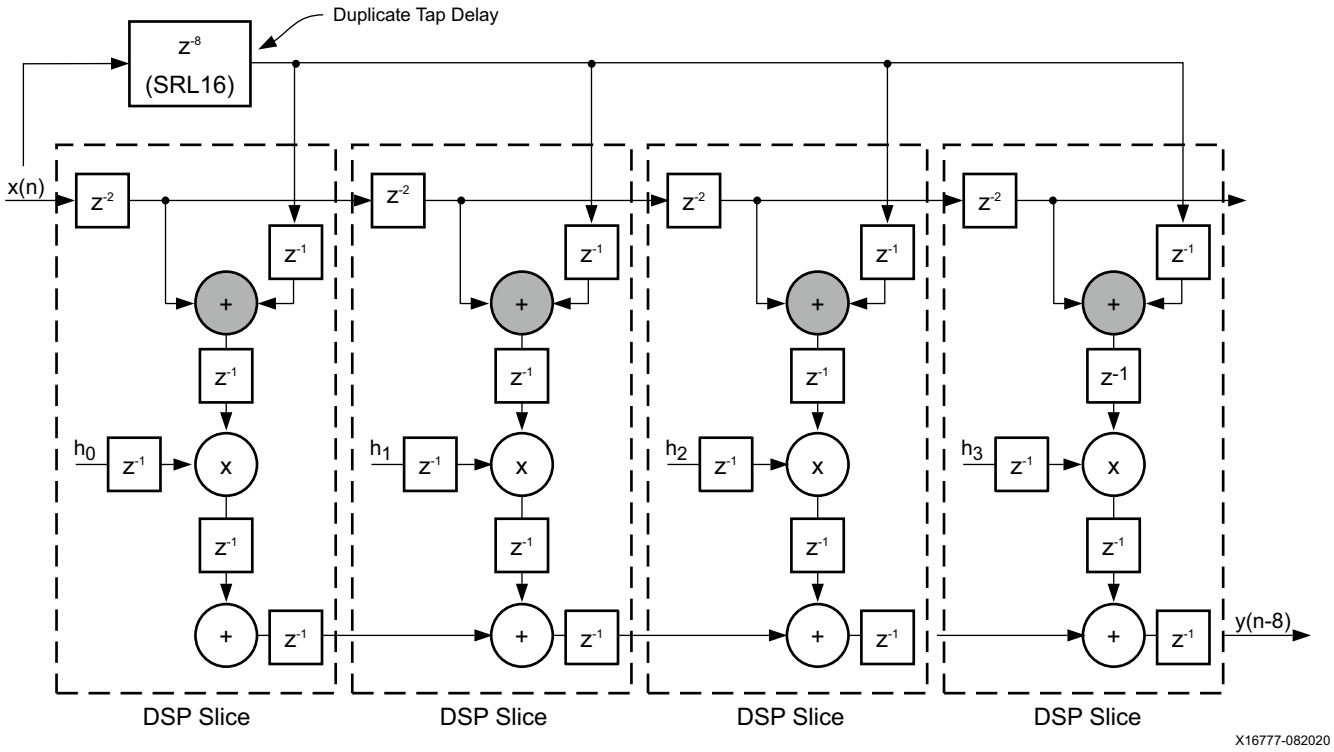


Figure 4-6: **8-Tap Even Symmetric Systolic FIR**

Memory-Mapped I/O Register Application

To use DSP48E2 slices as memory-mapped I/O registers, you must broadcast the write data bus feeding to all the DSP48E2 slices to be used in this fashion. To have random read access, a wide multiplexer is needed. Additional DSP48E2 slices can be configured as a wide bus multiplexer to help reduce routing congestion. An address decoder should be implemented in fabric logic to control individual PREG CEs to load the appropriate DSP output register from the write data bus.

XILINX

Chapter 5

Cascading: CARRYOUT, CARRYCASCOUT, and MULTSIGNOUT

Overview

This chapter describes the dedicated cascade features and clarifies key details of cascade signals.

CARRYOUT/CARRYCASCOUT

The DSP48E2 slice and the fabric carry chain use a different implementation style for subtract functions. The carry chain implementation in the CLB slices requires the CLB carry input pin to be connected to a constant `1` during subtract operations. The standard subtract operation with ALUMODE = `0011` in the DSP48E2 slice does not require the CARRYIN pin to be set to `1`.

In two's complement notation, negative values are obtained by performing a bitwise inversion and adding one, e.g., –B = ((not B) + 1). The CLB implements A–B as (A + (not B) + 1), as shown in Figure 5-1. An alternate implementation of a two-input subtracter is [not (B + (not A))], as shown in Figure 5-2. The alternate implementation allows the carry inputs to the adder/subtracter to still be available for normal use.

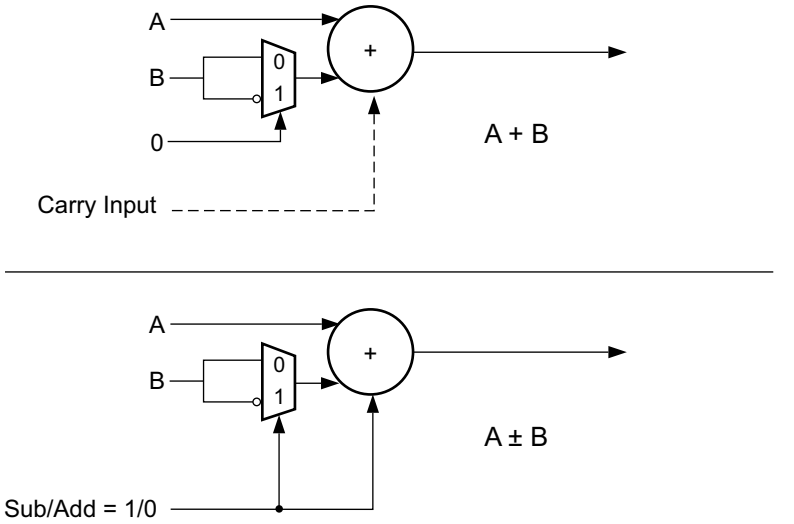


Figure 5-1: **CLB-Based Adder/Subtracter**

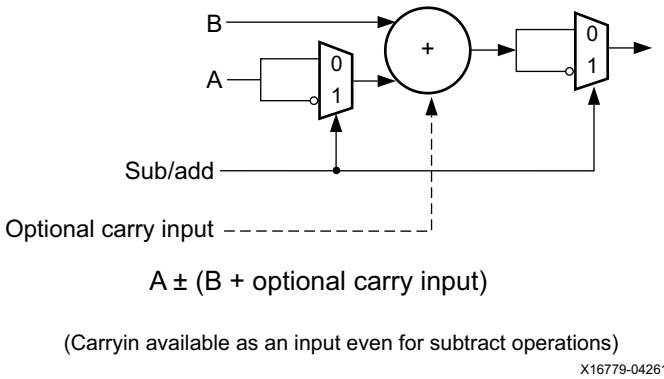


Figure 5-2: **Adder/Subtracter with Optional Carry Inputs**

The DSP48E2 slice uses the second implementation extended to 4-input adders with a CARRYIN input as shown in Figure 5-3, with the adder using ALUMODE = 0000. This allows DSP48E2 SIMD operations to perform subtract operations without a CARRYIN for each smaller add/subtract unit.

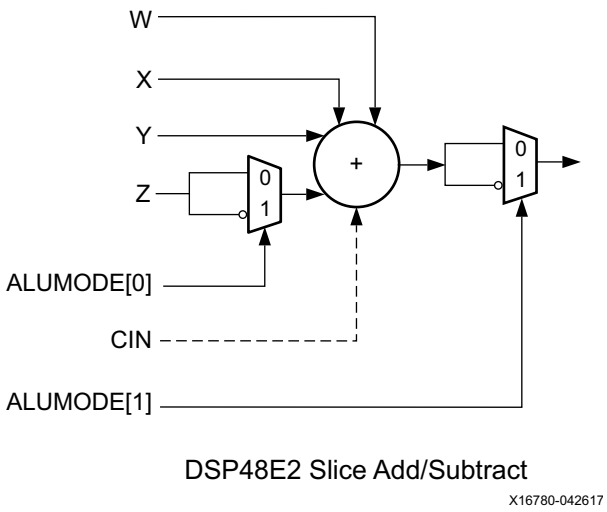


Figure 5-3: **DSP48E2 Slice Four-Input Adder**

ALUMODE supports additional subtract operations:

ALUMODE = `0001` implements (–Z + (W + X + Y + CIN) – `1`).

- For most uses, CIN is set to `1` to cancel out the `-1`.

ALUMODE = `0010` actually implements –(Z + W + X + Y + CIN) – `1`.

- This inversion of the P output obtained by using ALUMODE `0010` can be cascaded to another DSP slice to implement a two's complement subtract.

The result of the subtract operation is controlled by ALUMODE[1:0]. ALUMODE[0] inverts the Z input to the DSP adder. ALUMODE[1] inverts the output of the DSP adder. When ALUMODE[1:0] = `2'b11`, the CARRYOUT bits are inverted. Consider treating W, X, Y, and CIN as a single input (N). See Table 5-1 for simplified ALUMODE operations where P is the output of the DSP, and CDSP is the CARRYOUT output of the DSP as defined by an internally calculated carry C.

Table 5-1: **DSP ALUMODE CARRYOUT**

ALUMODE[3:0]	DSP Operation	CLB Extension (inputs A, B, CI)
`0000`	P = Z + N, CDSP = C	A:N, B:Z, CI:CDSP
`0001`	P = not(Z) + N ≤ DSP internal calculation = –Z + N – 1, CDSP = C	A:N, B:!Z, CI:CDSP
`0010`	P = not(Z + N) ≤ DSP internal calculation = –Z – N – 1, CDSP = C (Output only inverted)	A:N, B:Z, CI:CDSP All sum outputs must be inverted, all carries must be chained non-inverting

Table 5-1: **DSP ALUMODE CARRYOUT *(Cont'd)***

ALUMODE[3:0]	DSP Operation	CLB Extension (inputs A, B, CI)
0011	P = not(not(Z) + N) ≤ DSP internal calculation = –(–Z – 1 + N) – 1 = Z – N, CDSP = !C (Output and carryout inverted, carry cascaded to next DSP in true polarity relative to internal calculation)	A:Z, B:!N, CI:CDSP

For add operations, CARRYOUT[3] and CARRYCASCOUT are identical. CARRYOUT[3] matches the convention for CLB subtractions. The matched convention allows a CLB add-sub function to use the CARRYOUT[3] pin directly to extend two-input DSP48E2 slice subtract operations in the CLB. The CARRYCASCOUT is ideal for cascading up vertically to another DSP slice.

These CARRYOUT[3] and CARRYCASCOUT signals enable the construction of higher precision add/sub functions using multiple DSP48E2 slices or using a hybrid approach with both a DSP48E2 slice and a CLB adder/subtracter.

IMPORTANT: *CARRYOUT[3] and CARRYCASCOUT are not valid for three-input and four-input add/sub functions.*

MULTSIGNOUT and CARRYCASCOUT

RECOMMENDED: *CARRYOUT[3] should not be used for multiply operations because the first stage multiplier generates two partial products, which are added together in the second stage of the DSP48E2 slice.*

All DSP four-input add operations (including Multiply-Add and Multiply Accumulate) produce two CARRYOUT bits for retaining full precision. This is shown in Figure 5-4.

MULTSIGNOUT and CARRYCASCOUT serve as the two carry bits for MACC_EXTEND operations. If MULTSIGNOUT is the multiplier sign bit and CARRYCASCOUT is the cascaded carryout bit, the result is the software/Unisim model abstraction, shown in Figure 5-4.

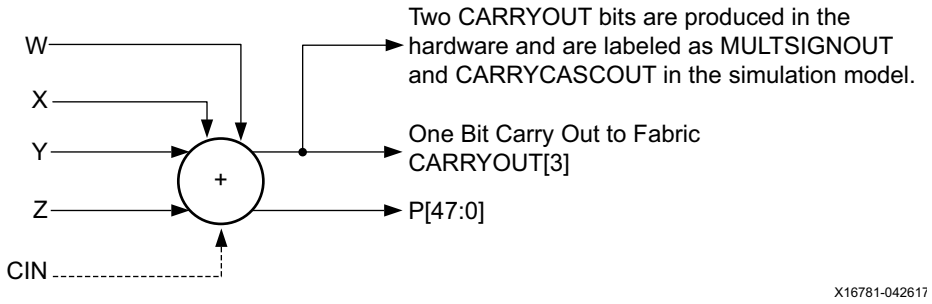


Figure 5-4: **DSP48E2 Four-Input Adder**

MULTSIGNOUT and CARRYCASCOUT in the simulation model do not match the hardware, but the output P bits do match for supported functions, such as MACC_EXTEND. For example, routing CARRYCASCOUT to the logic by using all zeros in the upper DSP slice does not match the CARRYOUT[3] of the lower DSP slice. Similarly, MULTSIGNOUT routed out to CLB logic is not actually the sign of the multiply result.

These MULTSIGNOUT and CARRYCASCOUT signals enable the construction of higher precision multiply-accumulate (MACC_EXTEND) functions, such as a 27x18 multiply accumulated for up to 96 bits of accumulator precision and running at the maximum DSP48E2 slice frequency.

It is also necessary to set the OPMODEREG and CARRYINSELREG to 1 when building large accumulators such as the 96-bit Multiply Accumulate. This prevents the simulation model from propagating unknowns to the upper DSP48E2 slice when a reset occurs.

Summary

Adder/Subtracter-only Operation

CARRYOUT[3]: Hardware and software match.

CARRYCASCOUT: Hardware and software match when ALUMODE = `0000`, `0001`, and `0010`, and inverted when ALUMODE = `0011`. The mismatch happens because the DSP48E2 slice performs the subtract operation using a different algorithm from the CLB logic; thus, the DSP48E2 slice requires an inverted CARRYOUT from the CLB logic.

MULTSIGNOUT is invalid in adder-only operation.

MACC Operation

CARRYOUT[3] is invalid in the MACC operation.

CARRYCASCOUT and MULTSIGNOUT: Hardware and software do not match due to modeling difference. The software simulation model is an abstraction of the hardware model. The software views of CARRYCASCOUT and MULTSIGNOUT enable higher-precision MACC functions to be built in the Unisim model. They are not logically equivalent to hardware CARRYCASCOUT and MULTSIGNOUT. Only the hardware and software results (P output) are logically equivalent. The internal signals (CARRYCASCOUT and MULTSIGNOUT) are not. See Figure 5-5.

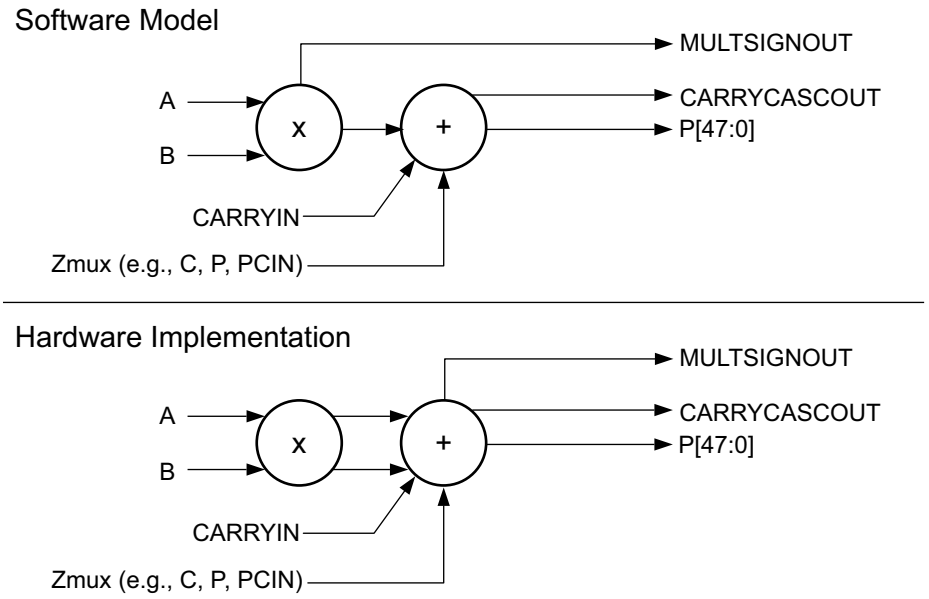


Figure 5-5: **MACC Software and Hardware Models**

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

1. *UltraScale Architecture Migration Methodology Guide* (UG1026)
2. UltraScale and UltraScale+ device data sheets:
   - *UltraScale Architecture and Product Overview* (DS890)
   - *Zynq UltraScale+ MPSoC Overview* (DS891)
   - *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
   - *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)
   - *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
   - *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)
3. UltraScale and UltraScale+ device packaging and pinout user guides:
   - *Kintex UltraScale and Virtex UltraScale FPGAs Packaging and Pinouts Product Specification User Guide* (UG575)
   - *Zynq UltraScale+ MPSoC Packaging and Pinout User Guide* (UG1075)
4. *Vivado Design Suite Reference Guide: Model-Based DSP Design Using System Generator* (UG958)
5. Vivado High-Level Synthesis
6. *UltraScale Architecture Libraries Guide* (UG974)
7. *Virtex-5 FPGA XtremeDSP Design Considerations User Guide* (UG193)
8. *XtremeDSP for Virtex-4 FPGAs User Guide* (UG073)
9. *7 Series DSP48E1 Slice User Guide* (UG479)
10. *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085)
11. UltraScale Architecture
12. DSP Solution
13. Vivado Video Tutorials
14. Xilinx DSP Training

Please Read: Important Legal Notices

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.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.