# Chapter 7

## Virtex Arithmetic Structures

# Overview

- Addition and subtraction
- Multiplication
- DSP48
- DSP48E

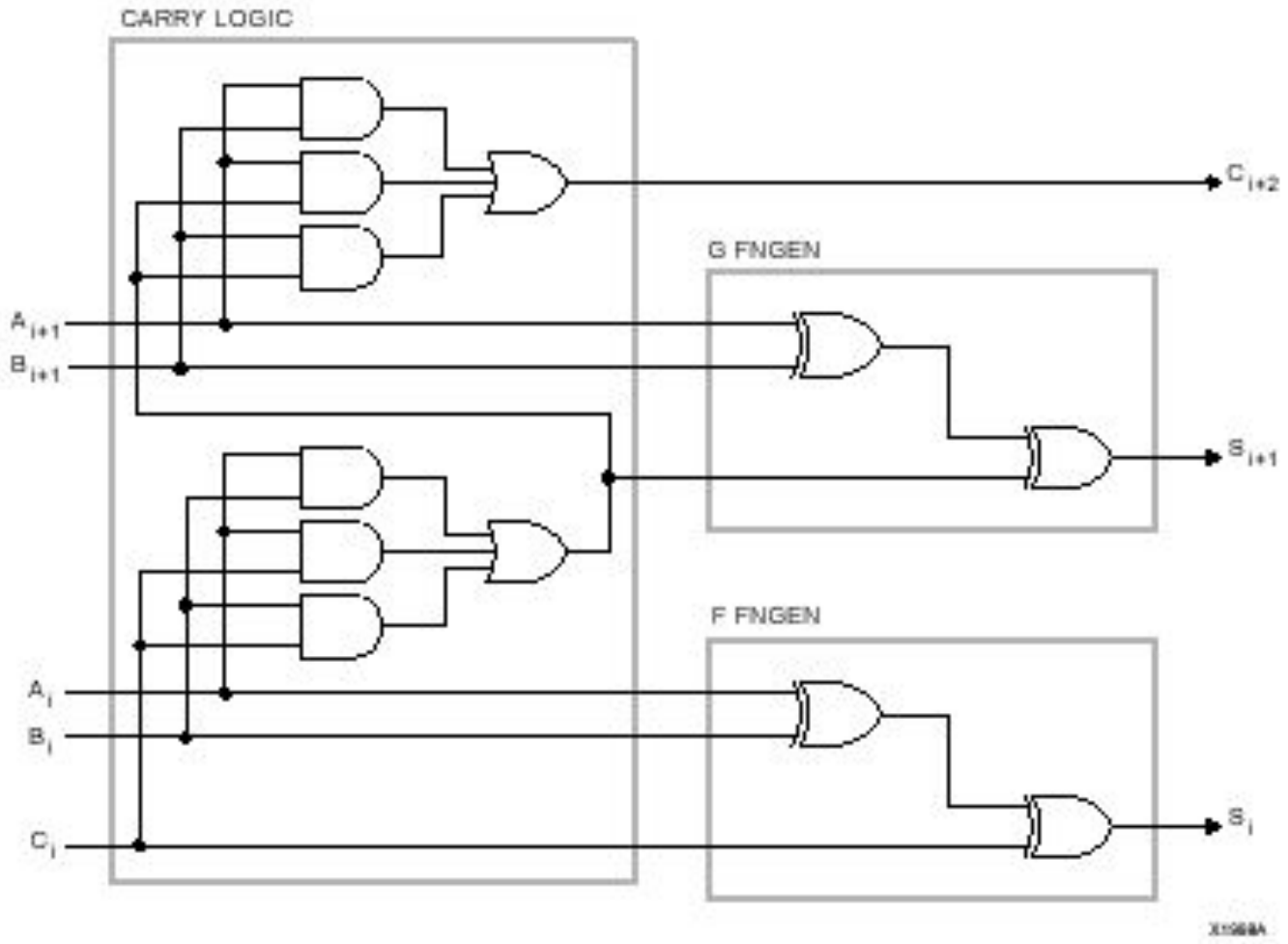
# Basic Adder Equations

$$S_i = A_i \text{ XOR } B_i \text{ XOR } C_i$$

$$C_{i+1} = A_i B_i + (A_i \text{ XOR } B_i) C_i$$

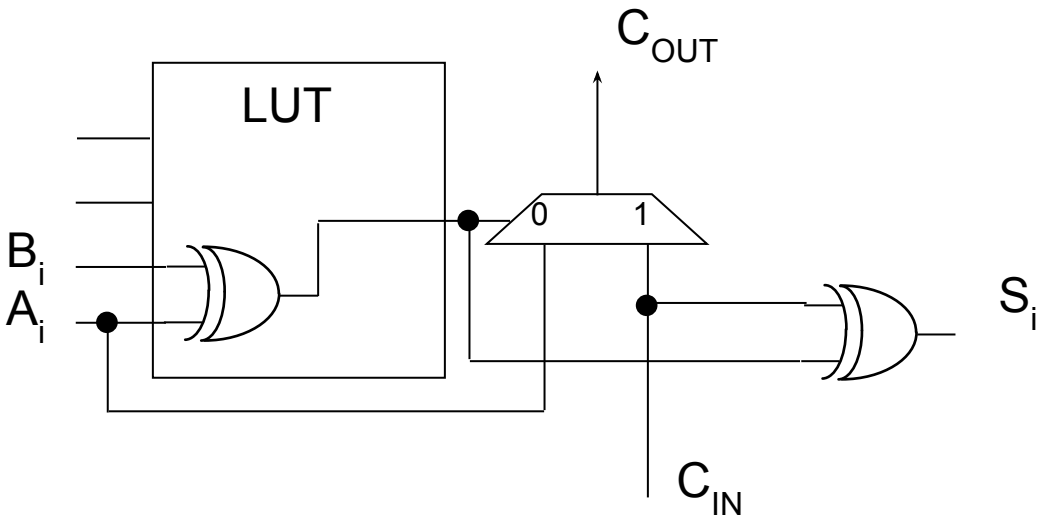
Note: not unique expression for $C_{i+1}$

# Basic Ripple Adder

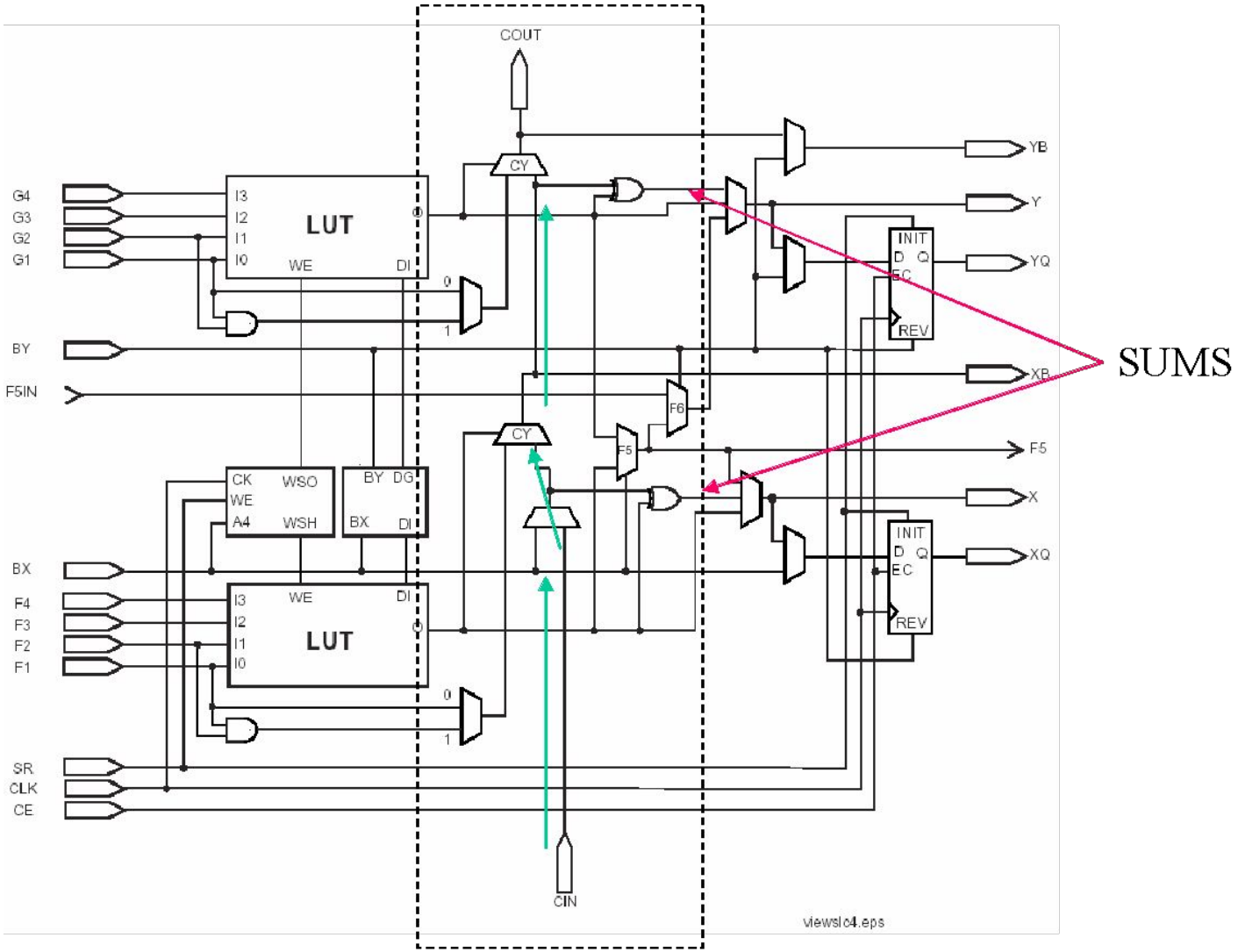


# Fast Carry Chains

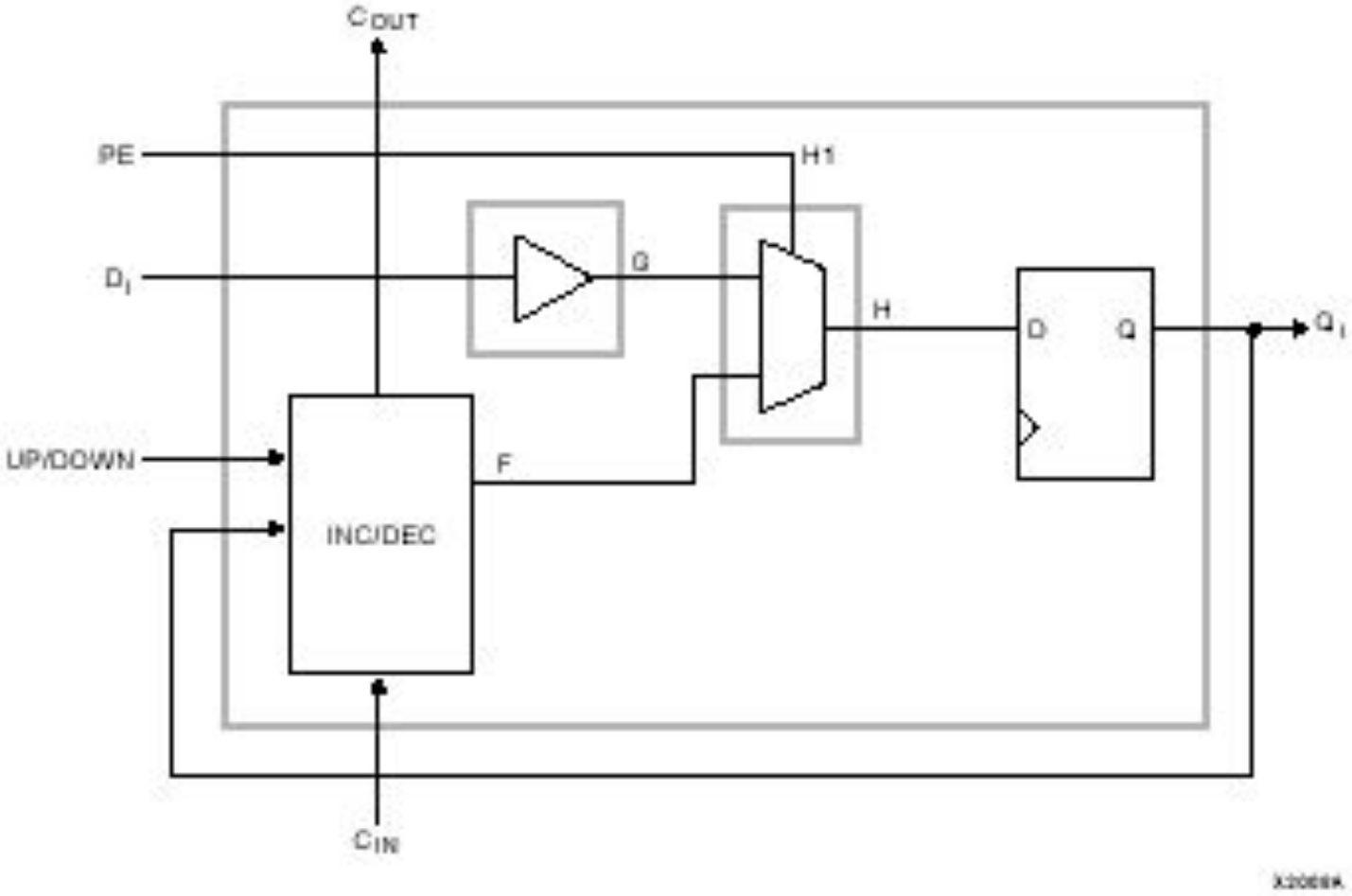
| A | B | C | $C_{OUT}$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $A = B, C_{OUT} = A$ |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | $A \neq B, C_{OUT} = C_{IN}$ |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | $A = B, C_{OUT} = A$ |
| 1 | 1 | 1 | 1 | |



# Tracing Carry Chain



# Counters Built from Adders



# Multiplication Review

```
                                        A3A2A1A0
                                        X     B1B0
                ------------------------------------------------
                          A3B0   A2 B0   A1B0  A0B0
               A3B1       A2B1   A1 B1  A0 B1
               ------------------------------------------------
P5       P4         P3         P2         P1      P0
```

Just like in the third grade . . .

# Fabric Based Multiplier

(comment: should be $C_{OUT}$)

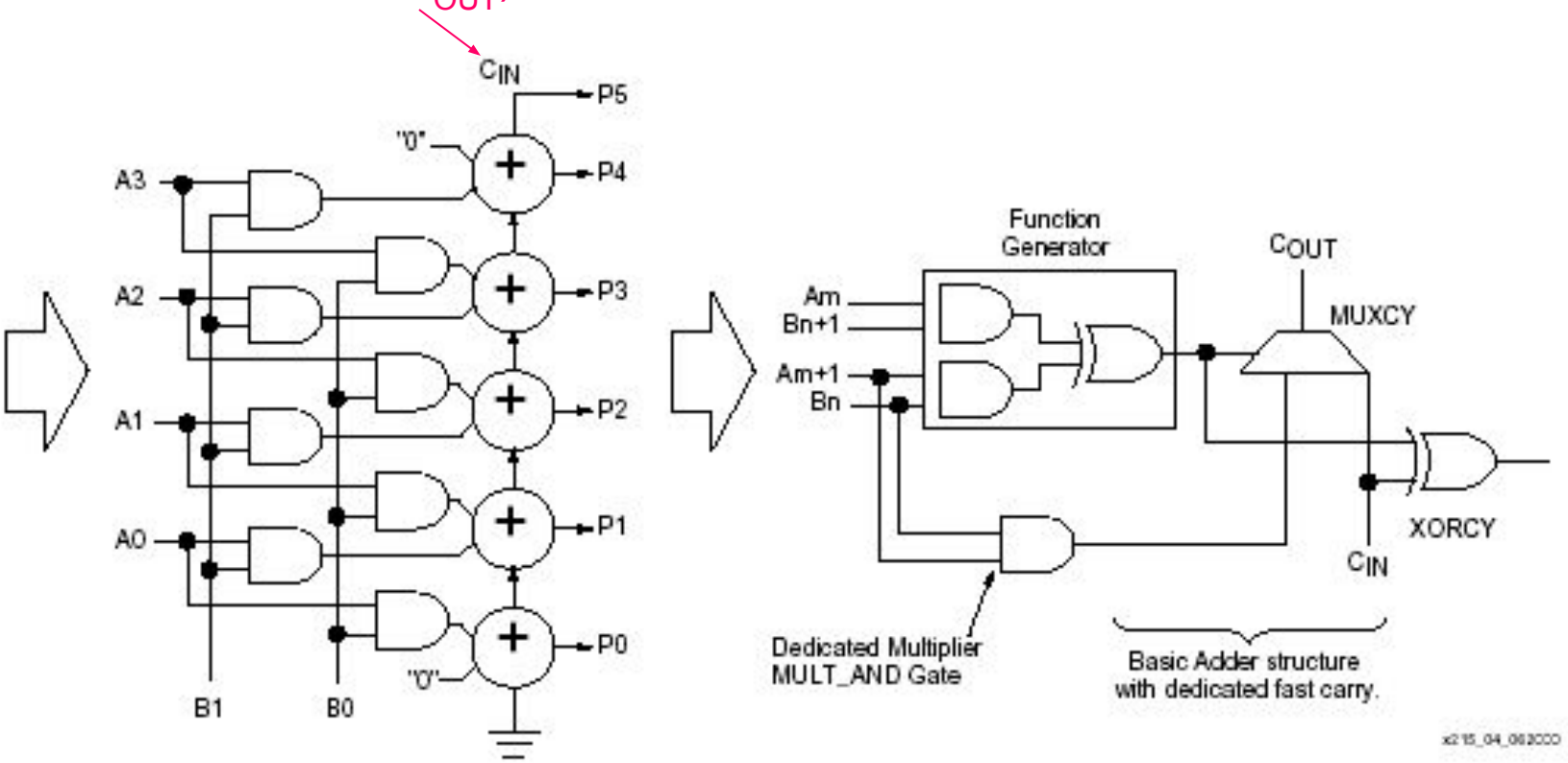
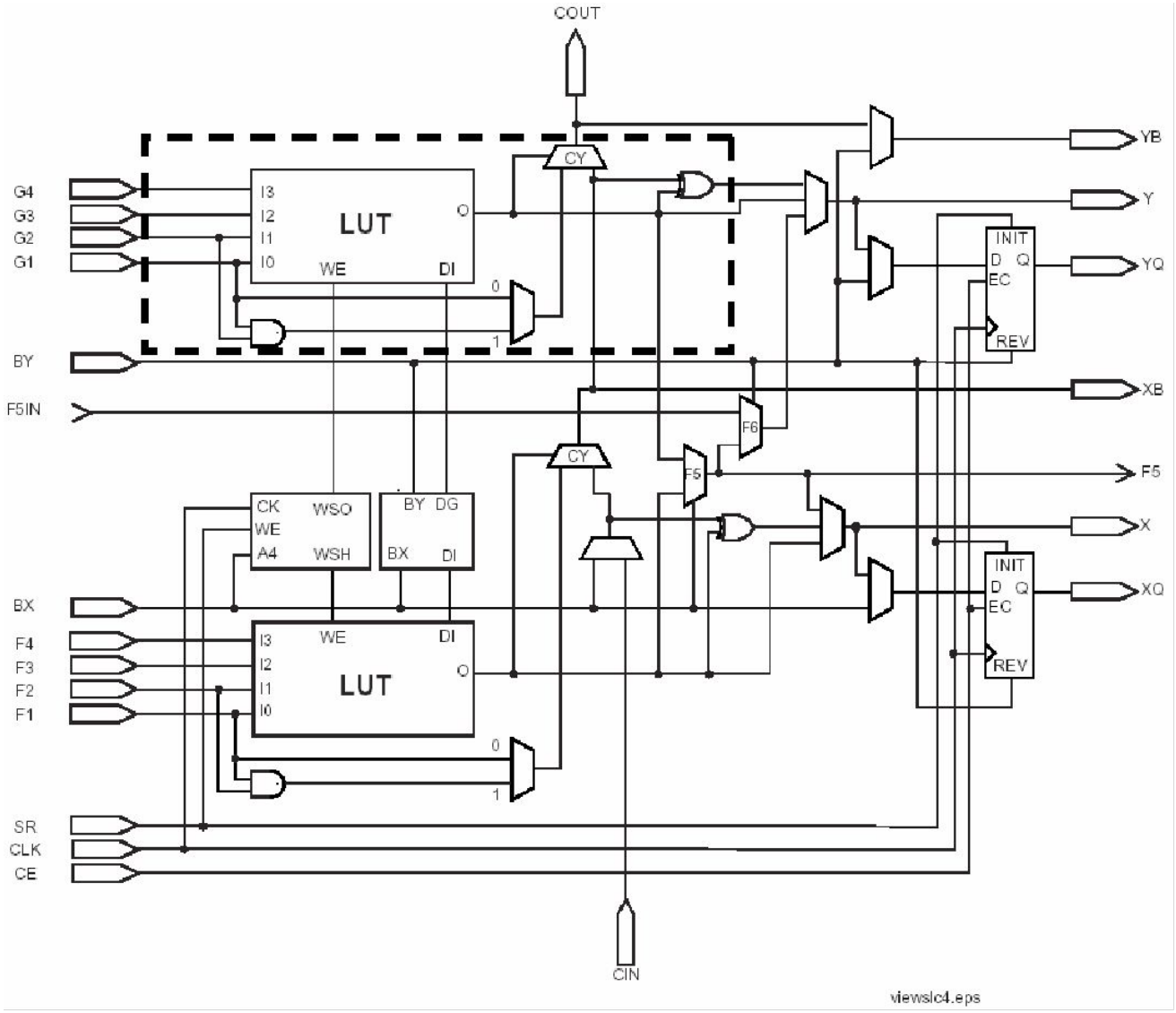



*Figure 5:* **N x 2 Full Multiplier Implementation**

# Mult/AND Identification

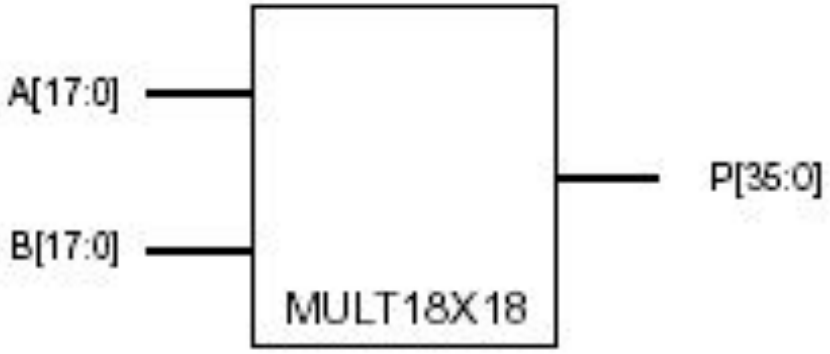


# Block Multipliers

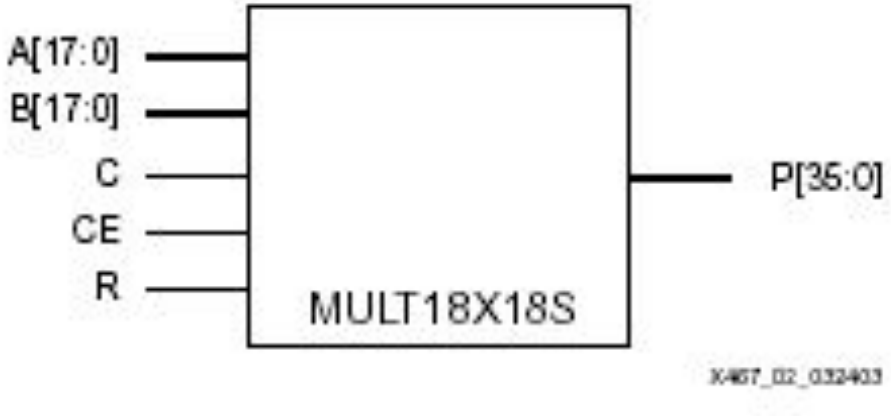
- Xilinx Spartan 3/E and Virtex II = 18 bit
- Applications needing fewer bits can tie off higher order bits appropriately
- Applications needing more bits must either:
  - Cascade block multipliers
  - Combine blocks with fabric multipliers
- Other uses for block multipliers

# Virtex II/Spartan 3/E Block Multipliers

| Primitive | A Width | B Width | P Width | Signed/Unsigned | Output |
|---|---|---|---|---|---|
| MULT18X18 | 18 | 18 | 36 | Signed (Two's Complement) | Combinatorial |
| MULT18X18S | 18 | 18 | 36 | Signed (Two's Complement) | Registered |




Combinational 2's Complement



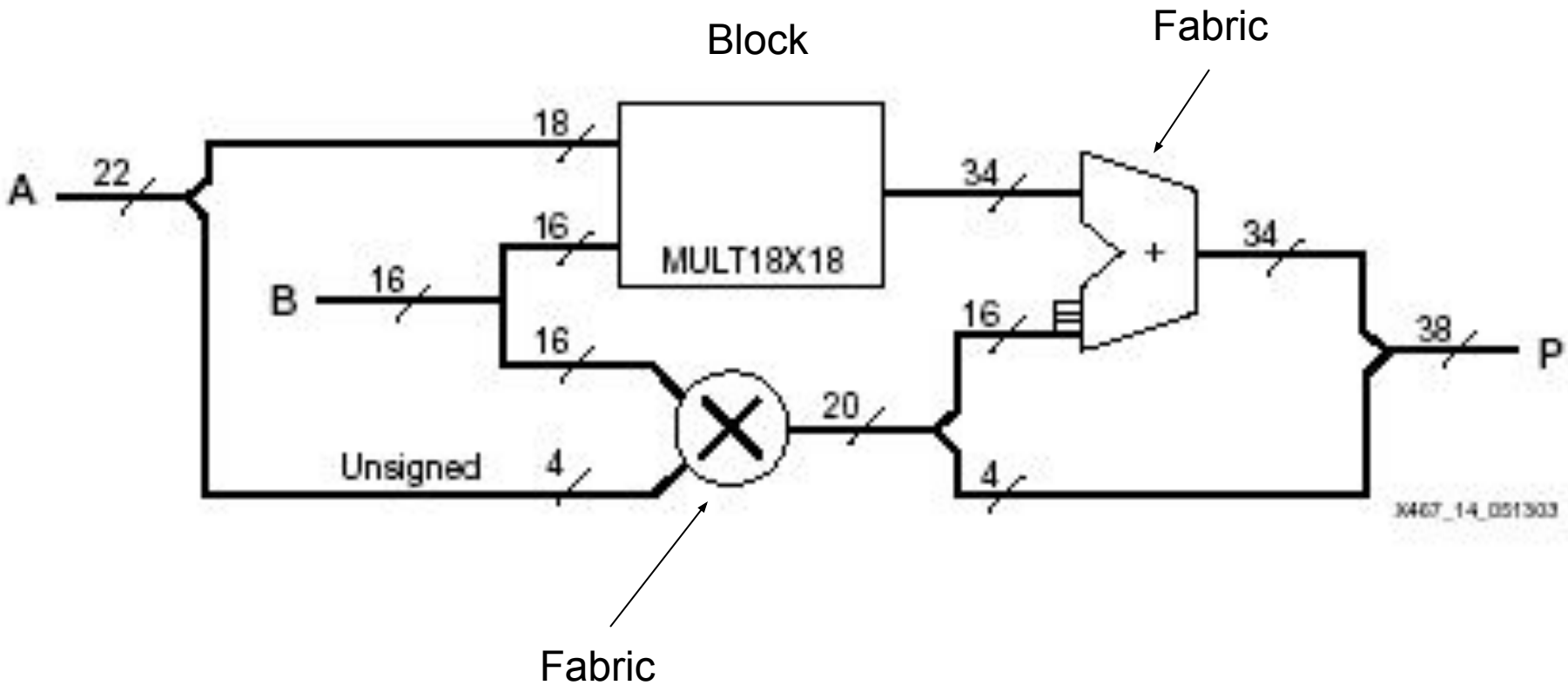

Sequential 2's Complement

# Spartan 3 Multiplier Mix

| Device | Multiplier Columns | Multipliers |
|---|---|---|
| XC3S50 | 1 | 4 |
| XC3S200 | 2 | 12 |
| XC3S400 | 2 | 16 |
| XC3S1000 | 2 | 24 |
| XC3S1500 | 2 | 32 |
| XC3S2000 | 2 | 40 |
| XC3S4000 | 4 | 96 |
| XC3S5000 | 4 | 104 |

With this many multipliers, better be something we can do with them if we Don't need this many. Also, must be other sizes we can create – somehow. See **XAPP 467**

# 22 X 16 Bit Multiplier



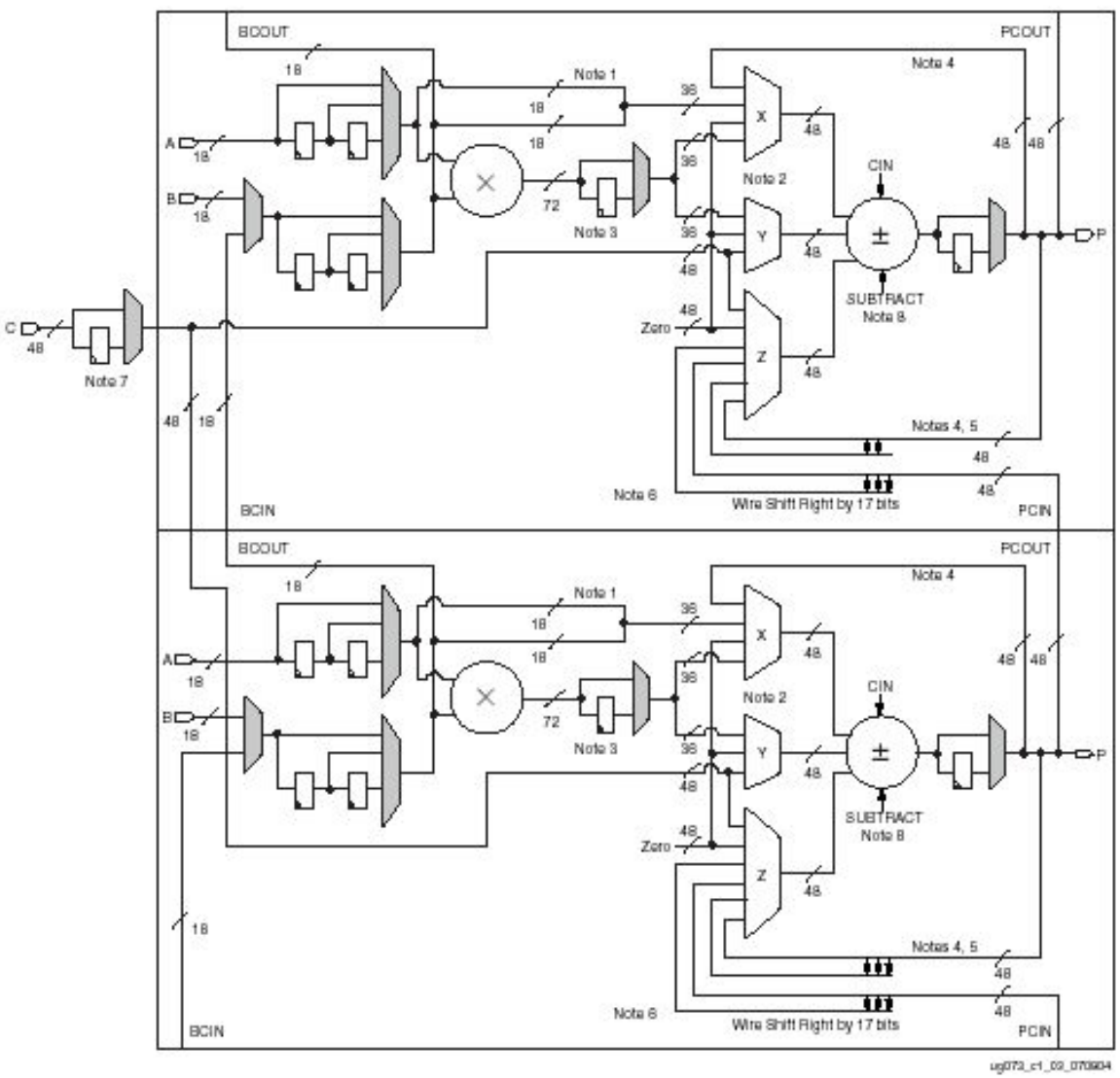
# Other Uses

- Multiply several small numbers with one block multiplier
- Division by repeated multiplication
- Barrel shift using multiplier
- Take two’s complement of a number
- Get the magnitude of a two’s complement number
- There are others

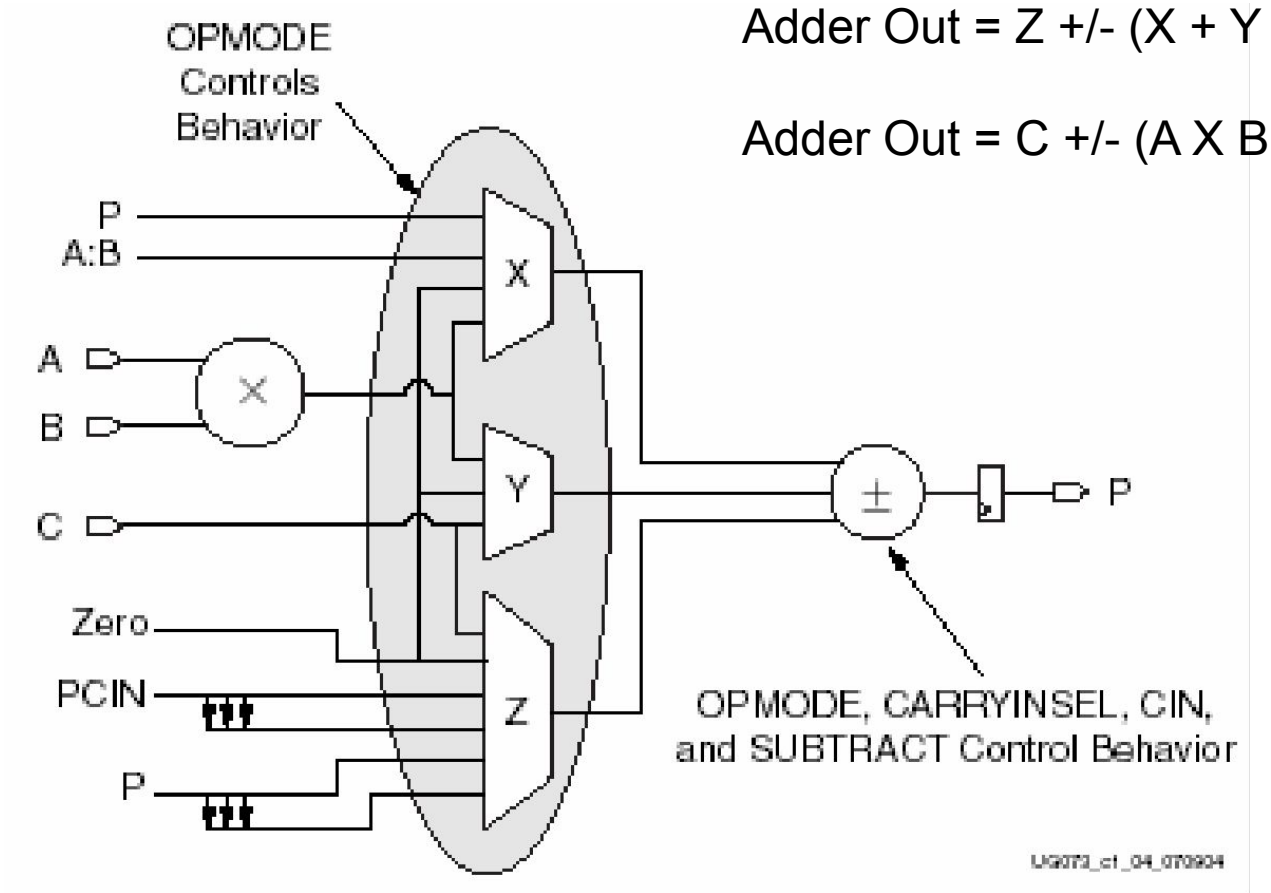
# DSP 48

- Introduced for Virtex 4
- Designed to efficiently cascade
- Xilinx teamed with outside arithmetic specialists to get best speed/area blocks
- Supports multiple rounding formats
- Not present in all V4 families

# Two Slice DSP 48



# Simplified DSP 48

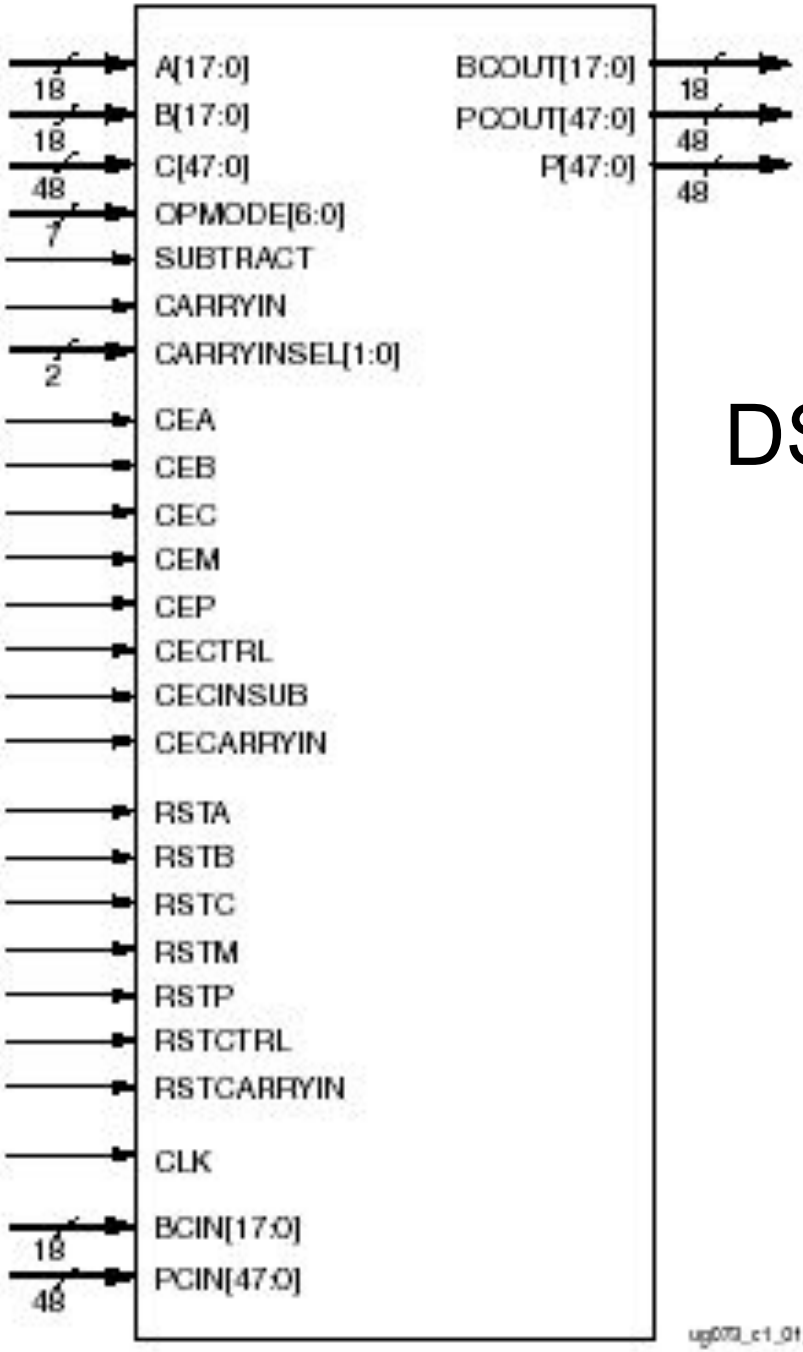



Adder Out = Z +/- (X + Y + CIN)

Adder Out = C +/- (A X B + CIN)

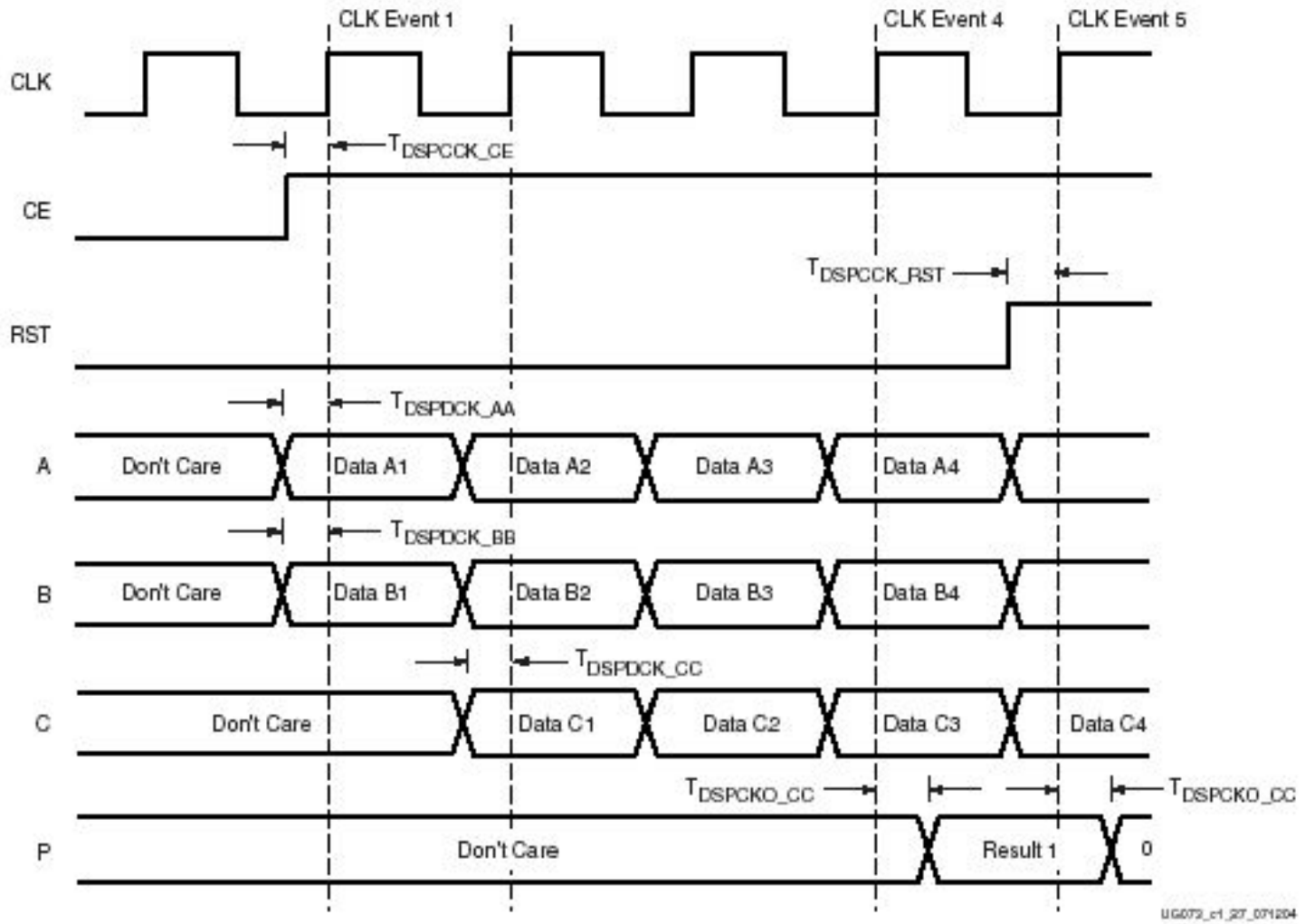
# The DSP 48 Instruction Set

| Hex OPMODE | Binary OPMODE | XYZ Multiplexer Outputs and Adder/Subtracter Output | | | |
|---|---|---|---|---|---|
| [6:0] | Z Y X | Z | Y | X | Adder/Subtracter Output |
| 0x00 | 000 00 00 | 0 | 0 | 0 | ±CIN |
| 0x02 | 000 00 10 | 0 | 0 | P | ±(P + CIN) |
| 0x03 | 000 00 11 | 0 | 0 | A:B | ±(A:B + CIN) |
| 0x05 | 000 01 01 | 0 | Note 1 | | ±(A × B + CIN) |
| 0x0c | 000 11 00 | 0 | C | 0 | ±(C + CIN) |
| 0x0e | 000 11 10 | 0 | C | P | ±(C + P + CIN) |
| 0x0f | 000 11 11 | 0 | C | A:B | ±(A:B + C + CIN) |
| 0x10 | 001 00 00 | PCIN | 0 | 0 | PCIN ± CIN |
| 0x12 | 001 00 10 | PCIN | 0 | P | PCIN ± (P + CIN) |
| 0x13 | 001 00 11 | PCIN | 0 | A:B | PCIN ± (A:B + CIN) |
| 0x15 | 001 01 01 | PCIN | Note 1 | | PCIN ± (A × B + CIN) |
| 0x1c | 001 11 00 | PCIN | C | 0 | PCIN ± (C + CIN) |
| 0x1e | 001 11 10 | PCIN | C | P | PCIN ± (C + P + CIN) |
| 0x1f | 001 11 11 | PCIN | C | A:B | PCIN ± (A:B + C + CIN) |
| 0x20 | 010 00 00 | P | 0 | 0 | P ± CIN |
| 0x22 | 010 00 10 | P | 0 | P | P ± (P + CIN) |
| 0x23 | 010 00 11 | P | 0 | A:B | P ± (A:B + CIN) |
| 0x25 | 010 01 01 | P | Note 1 | | P ± (A × B + CIN) |
| 0x2c | 010 11 00 | P | C | 0 | P ± (C + CIN) |
| 0x2e | 010 11 10 | P | C | P | P ± (C + P + CIN) |
| 0x2f | 010 11 11 | P | C | A:B | P ± (A:B + C + CIN) |
| 0x30 | 011 00 00 | C | 0 | 0 | C ± CIN |
| 0x32 | 011 00 10 | C | 0 | P | C ± (P + CIN) |
| 0x33 | 011 00 11 | C | 0 | A:B | C ± (A:B + CIN) |
| 0x35 | 011 01 01 | C | Note 1 | | C ± (A × B + CIN) |
| 0x3c | 011 11 00 | C | C | 0 | C ± (C + CIN) |
| 0x3e | 011 11 10 | C | C | P | C ± (C + P + CIN) |
| 0x3f | 011 11 11 | C | C | A:B | C ± (A:B + C + CIN) |
| 0x50 | 101 00 00 | Shift (PCIN) | 0 | 0 | Shift(PCIN) ± CIN |
| 0x52 | 101 00 10 | Shift (PCIN) | 0 | P | Shift(PCIN) ± (P + CIN) |
| 0x53 | 101 00 11 | Shift (PCIN) | 0 | A:B | Shift(PCIN) ± (A:B + CIN) |
| 0x55 | 101 01 01 | Shift (PCIN) | Note 1 | | Shift(PCIN) ± (A × B + CIN) |
| 0x5c | 101 11 00 | Shift (PCIN) | C | 0 | Shift(PCIN) ± (C + CIN) |
| 0x5e | 101 11 10 | Shift (PCIN) | C | P | Shift(PCIN) ± (C + P + CIN) |
| 0x5f | 101 11 11 | Shift (PCIN) | C | A:B | Shift(PCIN) ± (A:B + C + CIN) |
| 0x60 | 110 00 00 | Shift (P) | 0 | 0 | Shift(P) ± CIN |
| 0x62 | 110 00 10 | Shift (P) | 0 | P | Shift(P) ± (P + CIN) |
| 0x63 | 110 00 11 | Shift (P) | 0 | A:B | Shift(P) ± (A:B + CIN) |
| 0x65 | 110 01 01 | Shift (P) | Note 1 | | Shift(P) ± (A × B + CIN) |
| 0x6c | 110 11 00 | Shift (P) | C | 0 | Shift(P) ± (C + CIN) |
| 0x6e | 110 11 10 | Shift (P) | C | P | Shift(P) ± (C + P + CIN) |
| 0x6f | 110 11 11 | Shift (P) | C | A:B | Shift(P) ± (A:B + C + CIN) |

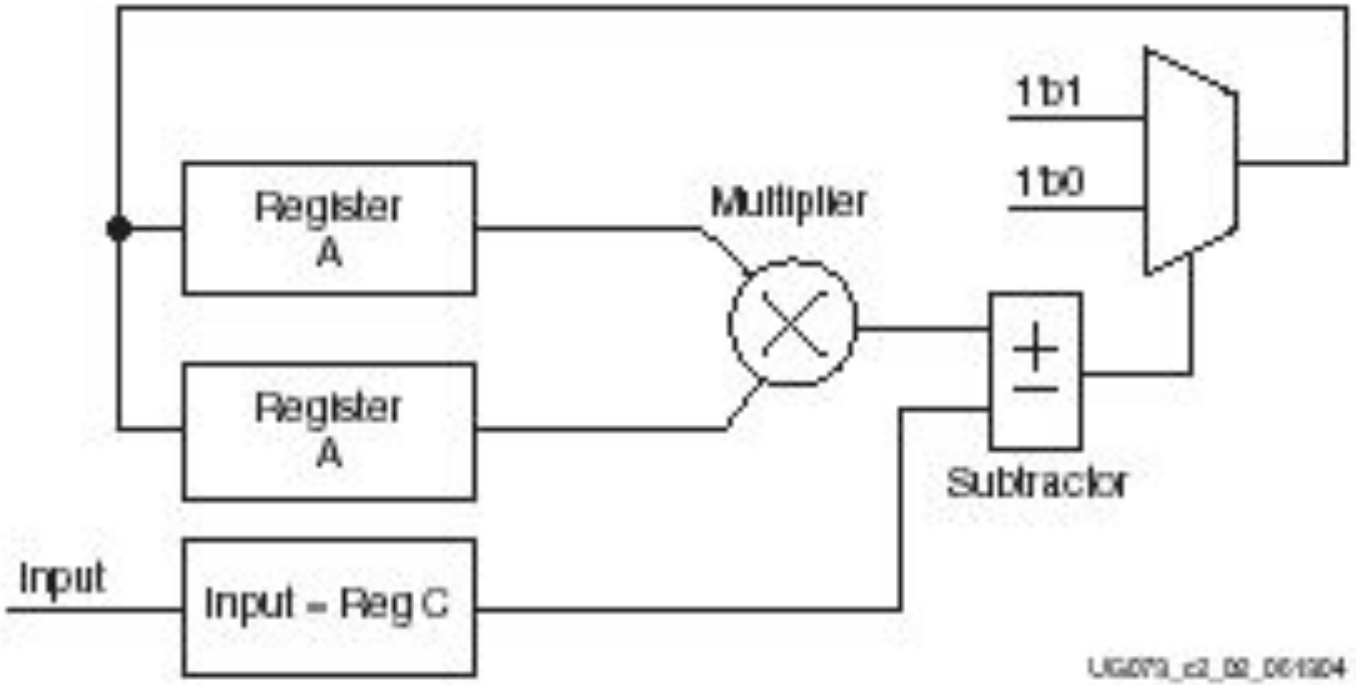



# DSP 48 Primitive Symbol

# DSP 48 Timing



# Doing Other Things - SQRT

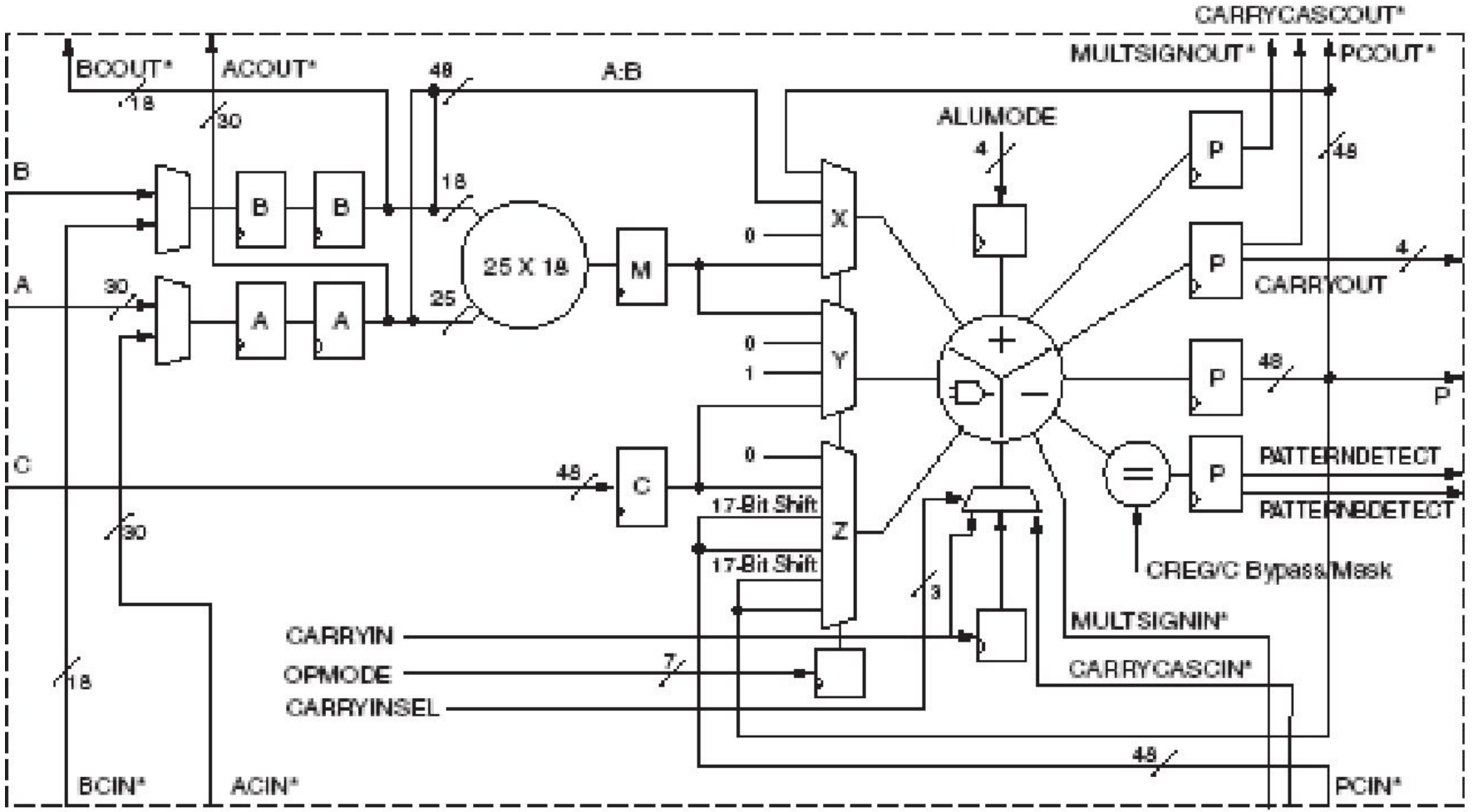


# Important to Do Other Things with DSP 48

- Divider
- Large multiplexer
- Barrel shifters
- Two’s Complement converters
- Large counters
- Etc.

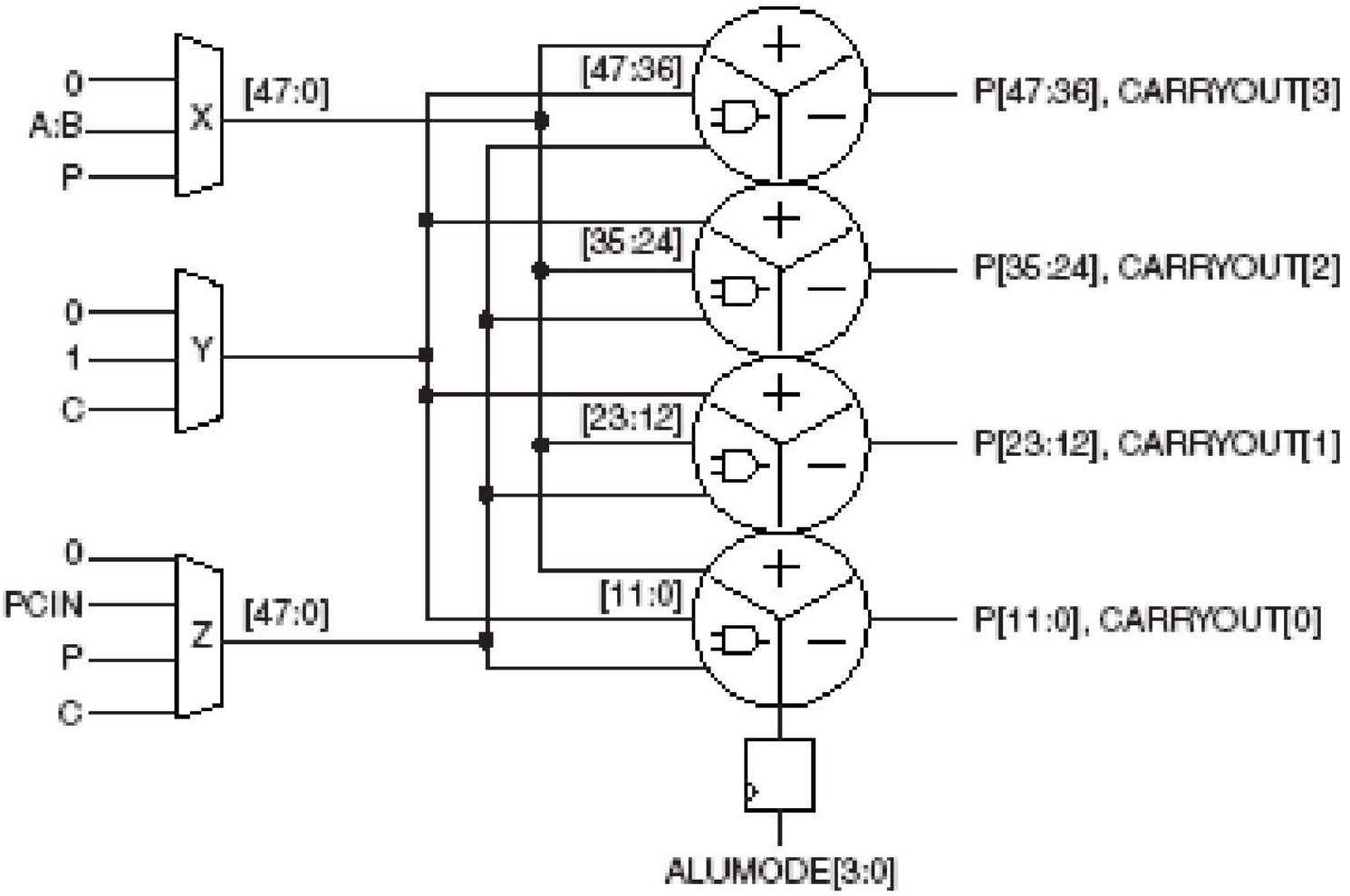
DSP 48 Handbook available on Xilinx website

# Virtex 5 DSP48E




*These signals are dedicated routing paths internal to the DSP48E column. They are not accessible via fabric routing resources.

# Simpler DSP48E



# Other Useful Functions

- Building up “word wise” logic
  - AND,OR
- Same list as for the DSP48

# Support

- Xilinx offers an elaborate (read:$) set of tools that interface to the MatLab toolchain
- Focused on DSP algorithm development
- Permits “hardware in the loop” simulation
- Also working on higher level compile tools as DSP developers tend to be more mathematicians and less “hardware designers” (read: C, C++ oriented)