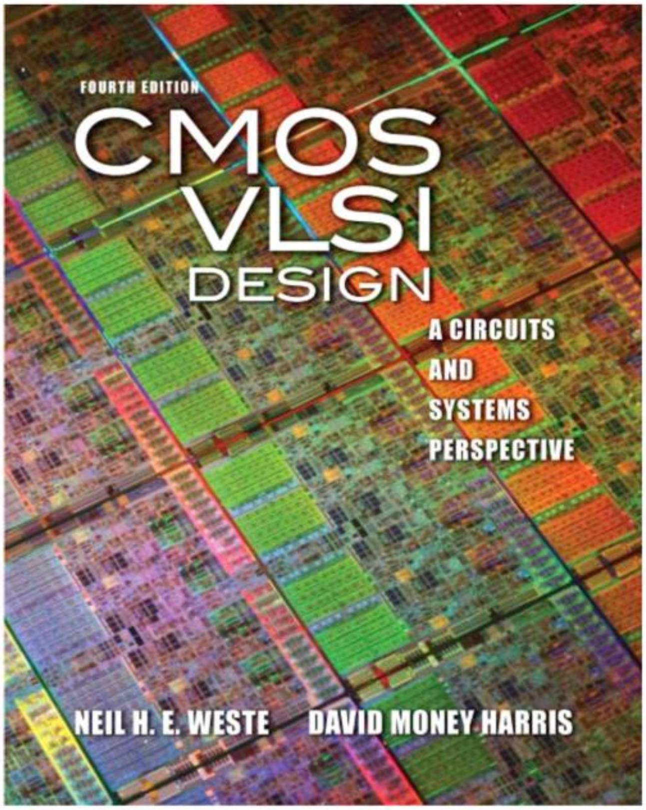
Printout

Monday, October 21, 2019 9:24 PM



Lecture12: Datapath Functional Units

Outline

- Comparators
- Shifters
- Multi-input Adders
- Multipliers

14: Datapath Functional Units CMOS VLSI Design 4th Ed. 2

Page 3

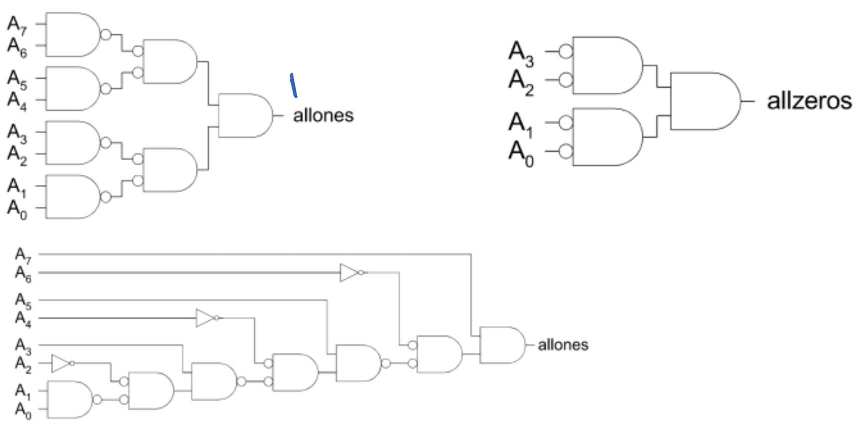
Monday, October 21, 2019 9:24 PM

Comparators

- 0's detector: A = 00…000
- 1's detector: A = 11…111
- Equality comparator: A = B
- Magnitude comparator: A < B

1's & 0's Detectors

- 1's detector: N-input AND gate
- 0's detector: NOTs + 1's detector (N-input NOR)



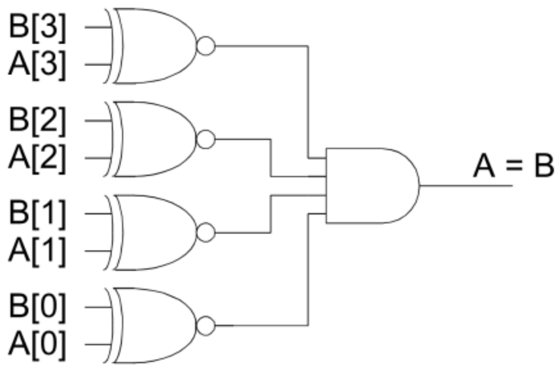
0000
0010

Page 5

Monday, October 21, 2019 9:24 PM

Equality Comparator

- Check if each bit is equal (XNOR, aka equality gate)
- 1's detect on bitwise equality

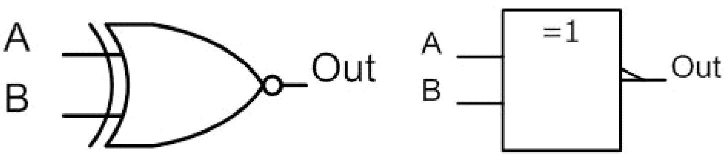


XNOR

Monday, October 21, 2019 9:25 PM

XNOR Gate

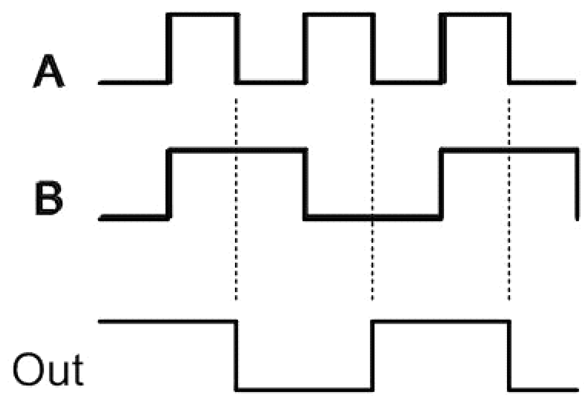
- Symbols



- Truth Table

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

- Timing Diagram



- Logic Expression:

  $Out = \overline{A}\,\overline{B} + AB;\ \overline{A \oplus B}$

Page 6

Monday, October 21, 2019 9:24 PM

Magnitude Comparator

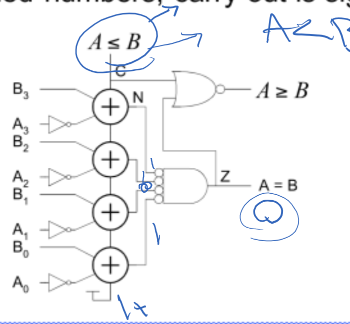
- Compute B – A and look at sign
- B – A = B + ~A + 1
- For unsigned numbers, carry out is sign bit

$B + \bar{A} + 1$

12 - 8

8+2=

A < B



8 4 2 1

A = 1101 = 8+4+1 = 13

B = 1111 = 8+4+2+1 = 15

$\bar{A}$ = 0010

+B = 1111

1

10010

Shifters

- Logical Shift:
  - Shifts number left or right and fills with 0's
    - 1011 LSR 1 = 0101 1011 LSL1 = 0110
- Arithmetic Shift:
  - Shifts number left or right. Rt shift sign extends
    - 1011 ASR1 = 1101 1011 ASL1 = 0110
- Rotate:

0111 ASR1 = 0011

  - Shifts number left or right and fills with lost bits
    - 1011 ROR1 = 1101 1011 ROL1 = 0111

ROR1 1011 = 1101 ⇒ 1110 = 0111

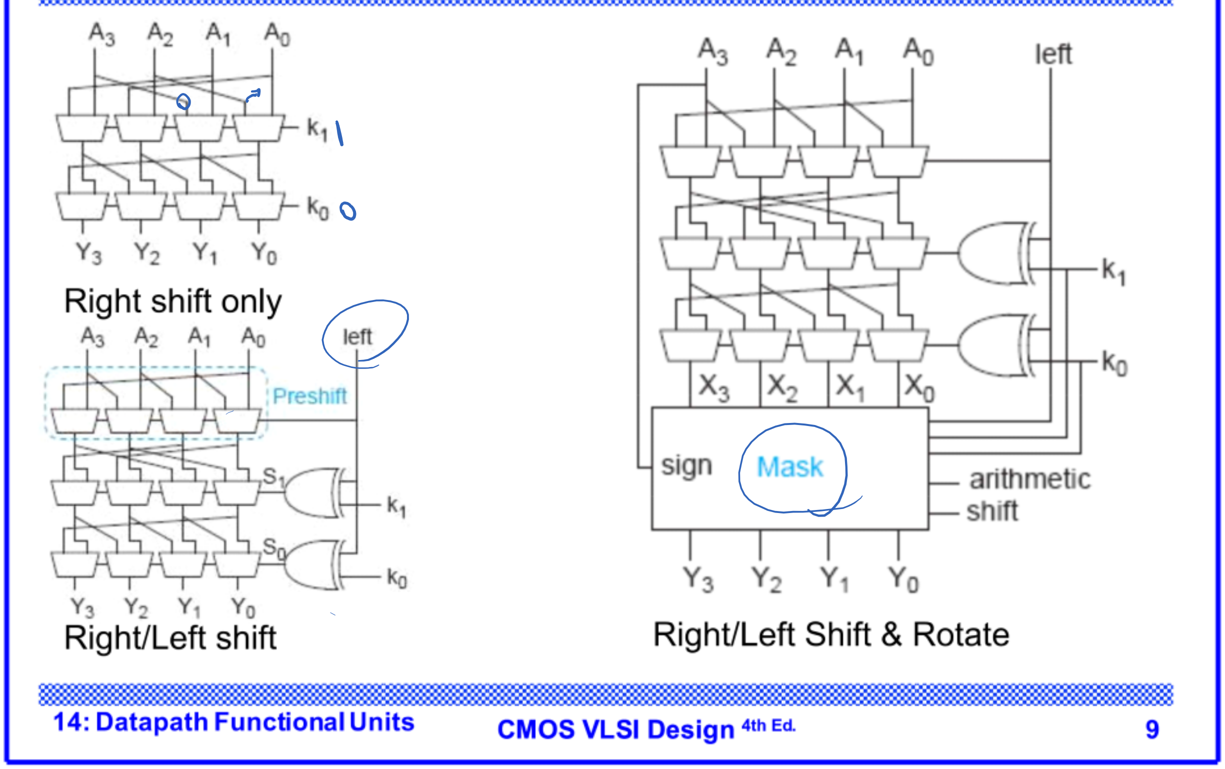
ROL1 1011 = 0111

Barrel Shifter

- Barrel shifters perform right rotations using wrap-around wires.
- Left rotations are right rotations by $N - k = \overline{k} + 1$ bits.
- Shifts are rotations with the end bits masked off.

Monday, October 21, 2019 9:24 PM

Logarithmic Barrel Shifter

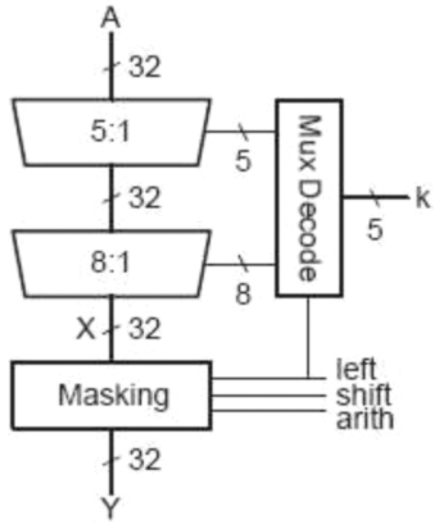


Page 10

Monday, October 21, 2019 9:24 PM

32-bit Logarithmic Barrel

- Datapath never wider than 32 bits
- First stage preshifts by 1 to handle left shifts

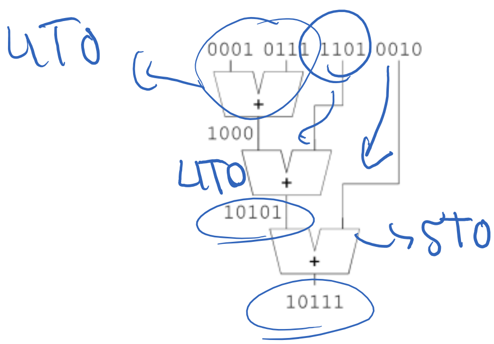


Page 11

Monday, October 21, 2019 9:24 PM

Multi-input Adders

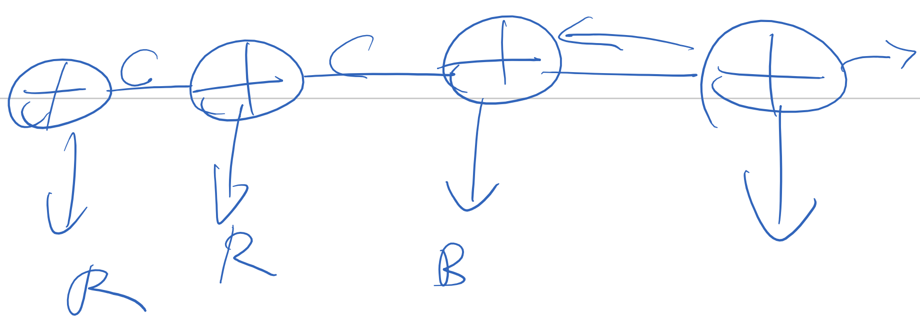
- Suppose we want to add k N-bit words
  - Ex: 0001 + 0111 + 1101 + 0010 = 10111
- Straightforward solution: k-1 N-input CPAs
  - Large and slow



111
0001
0111
1000
→ TO

= 4+4+5 = 13To

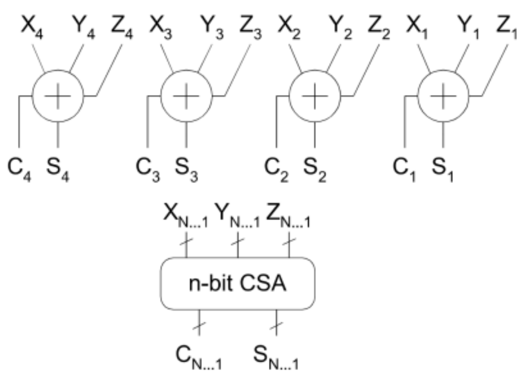
14: Datapath Functional Units CMOS VLSI Design 4th Ed. 11



Carry Save Addition

- A full adder sums 3 inputs and produces 2 outputs
  - Carry output has twice *weight* of sum output
- N full adders in parallel are called *carry save adder*
  - Produce N sums and N carry outs

To



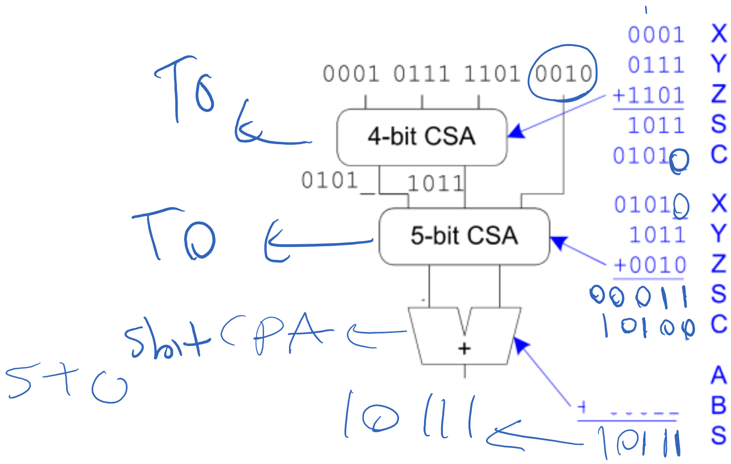
1111
1001
1
0110 S
1001 C

Page 13

Monday, October 21, 2019 9:24 PM

CSA Application

- Use k-2 stages of CSAs
  - Keep result in carry-save redundant form
- Final CPA computes actual result

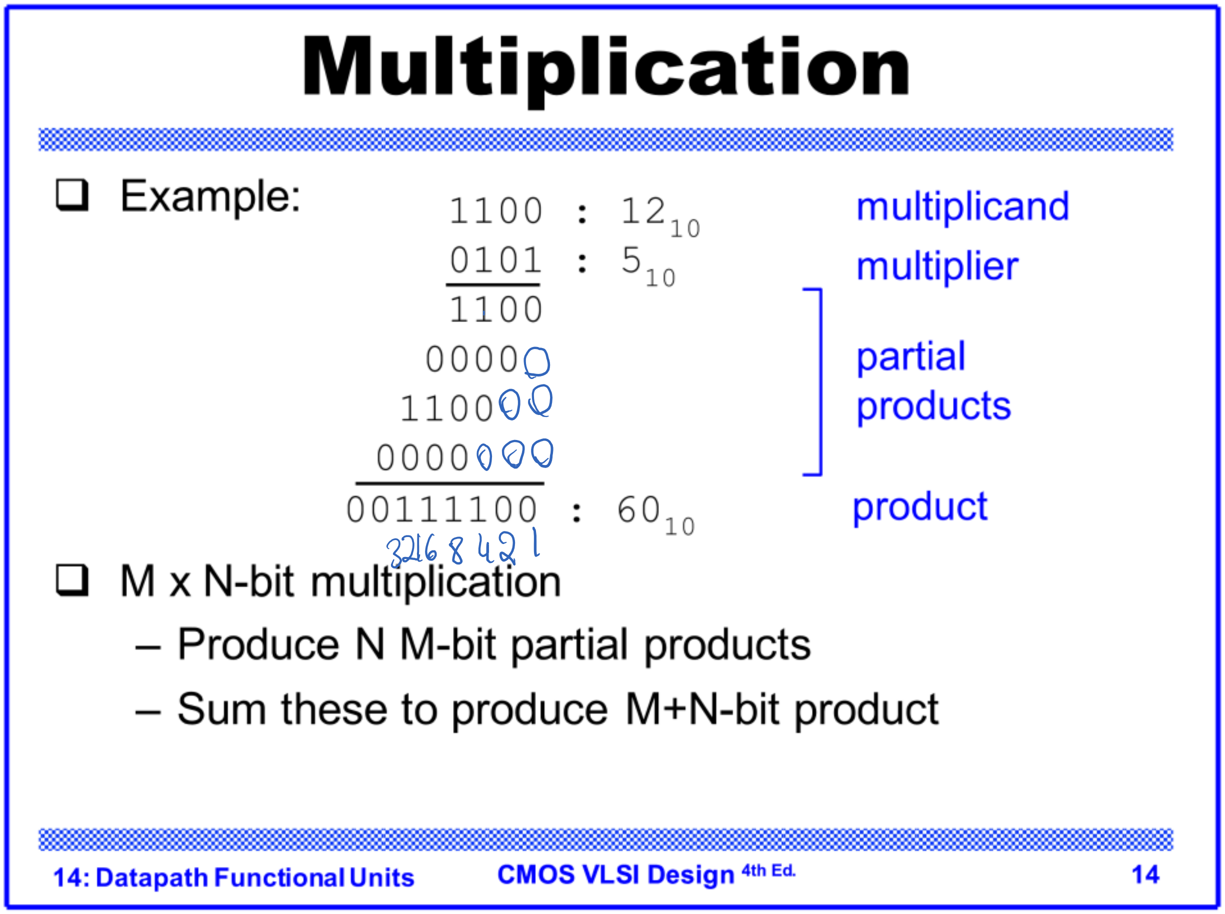


$T_0 + T_0 + 5T_0 = 7T_0$

14: Datapath Functional Units CMOS VLSI Design 4th Ed. 13

Page 14

Monday, October 21, 2019 9:24 PM



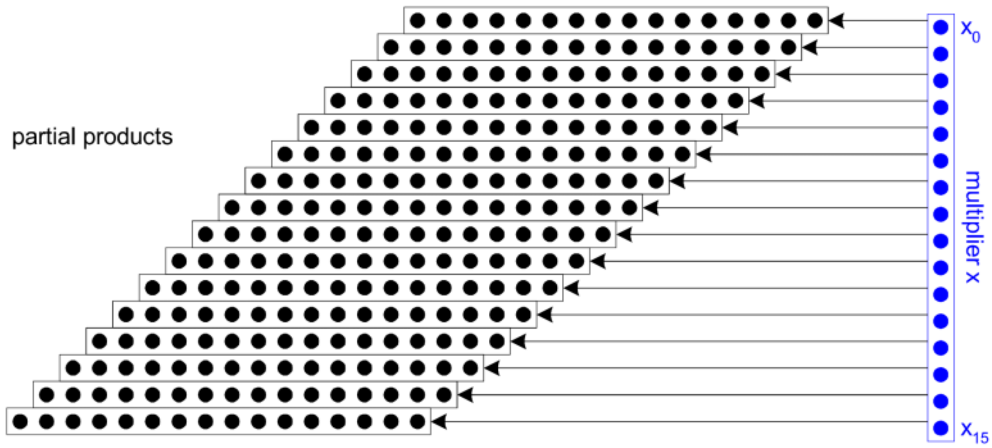
General Form

- Multiplicand: $Y = (y_{M-1}, y_{M-2}, \ldots, y_1, y_0)$
- Multiplier: $X = (x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$
- Product: $P = \left(\sum_{j=0}^{M-1} y_j 2^j\right)\left(\sum_{i=0}^{N-1} x_i 2^i\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$

						y_5	y_4	y_3	y_2	y_1	y_0	multiplicand
						x_5	x_4	x_3	x_2	x_1	x_0	multiplier
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	
					x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0		
				x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0			partial products
			x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0				
		x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0					
	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0						
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	product

Dot Diagram

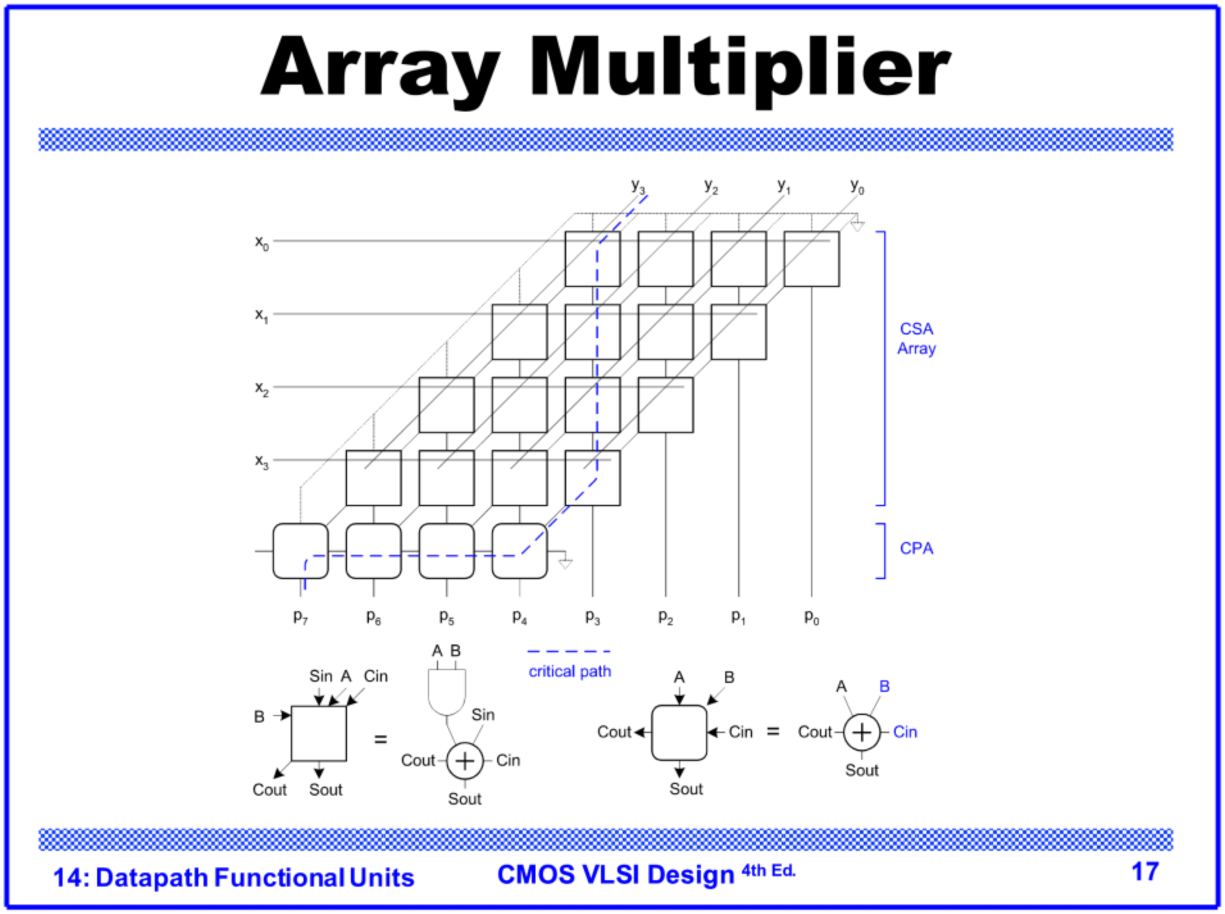
- Each dot represents a bit



14: Datapath Functional Units CMOS VLSI Design 4th Ed. 16

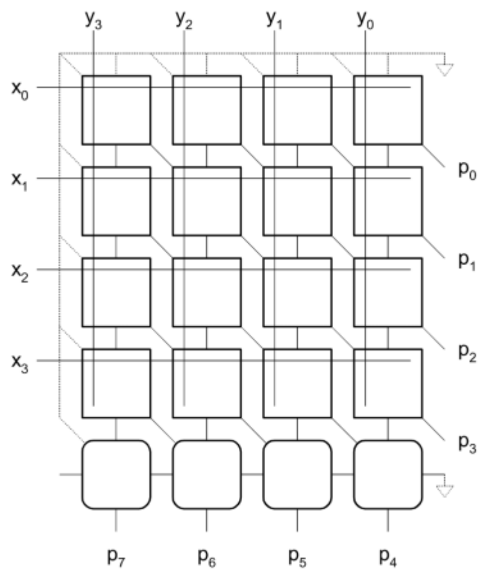
Page 17

Monday, October 21, 2019 9:24 PM



Rectangular Array

- Squash array to fit rectangular floorplan



Advanced Multiplication

- Booth Encoding
- Signed vs. unsigned inputs
- Higher radix Booth encoding
- Array vs. tree CSA networks