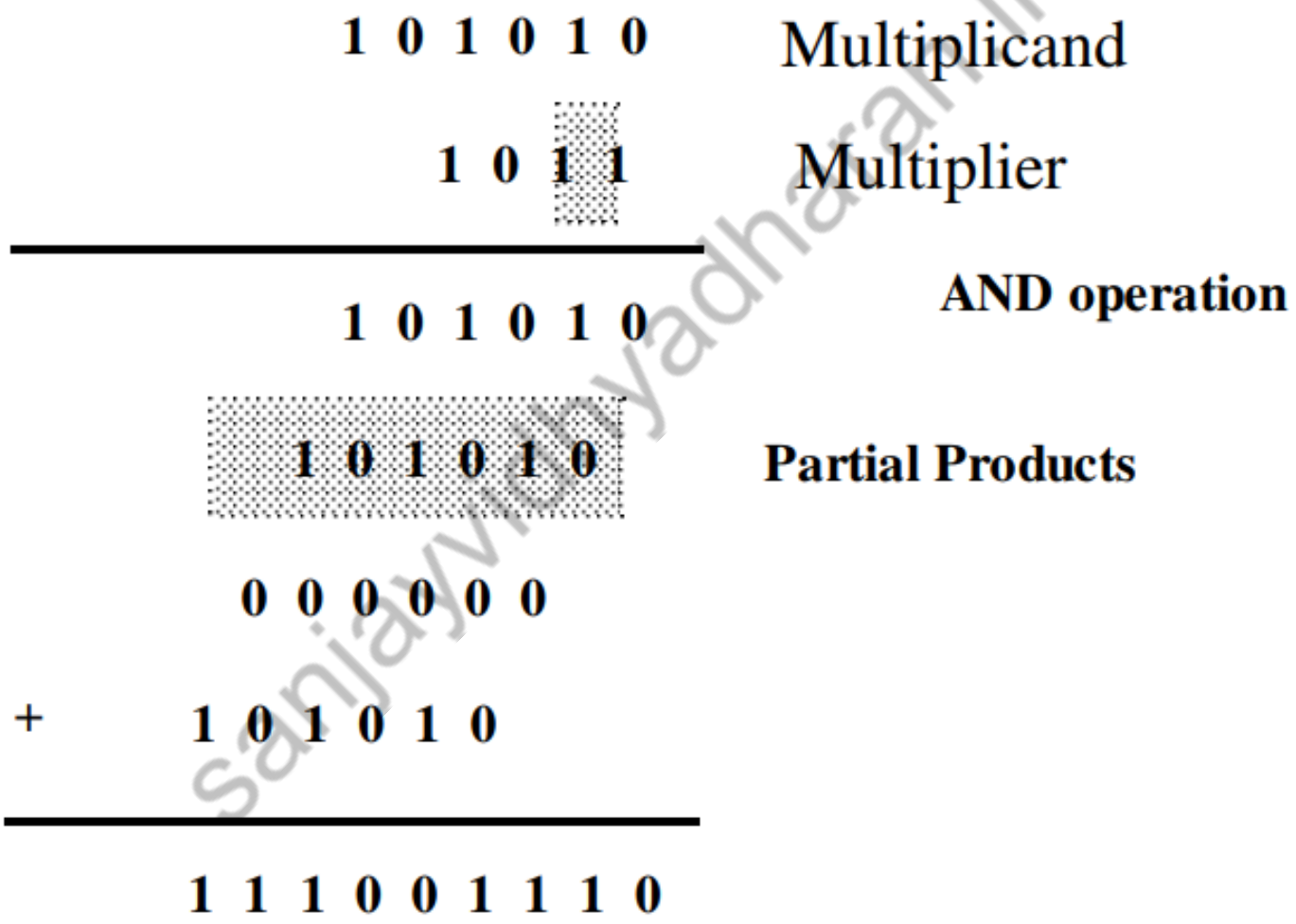
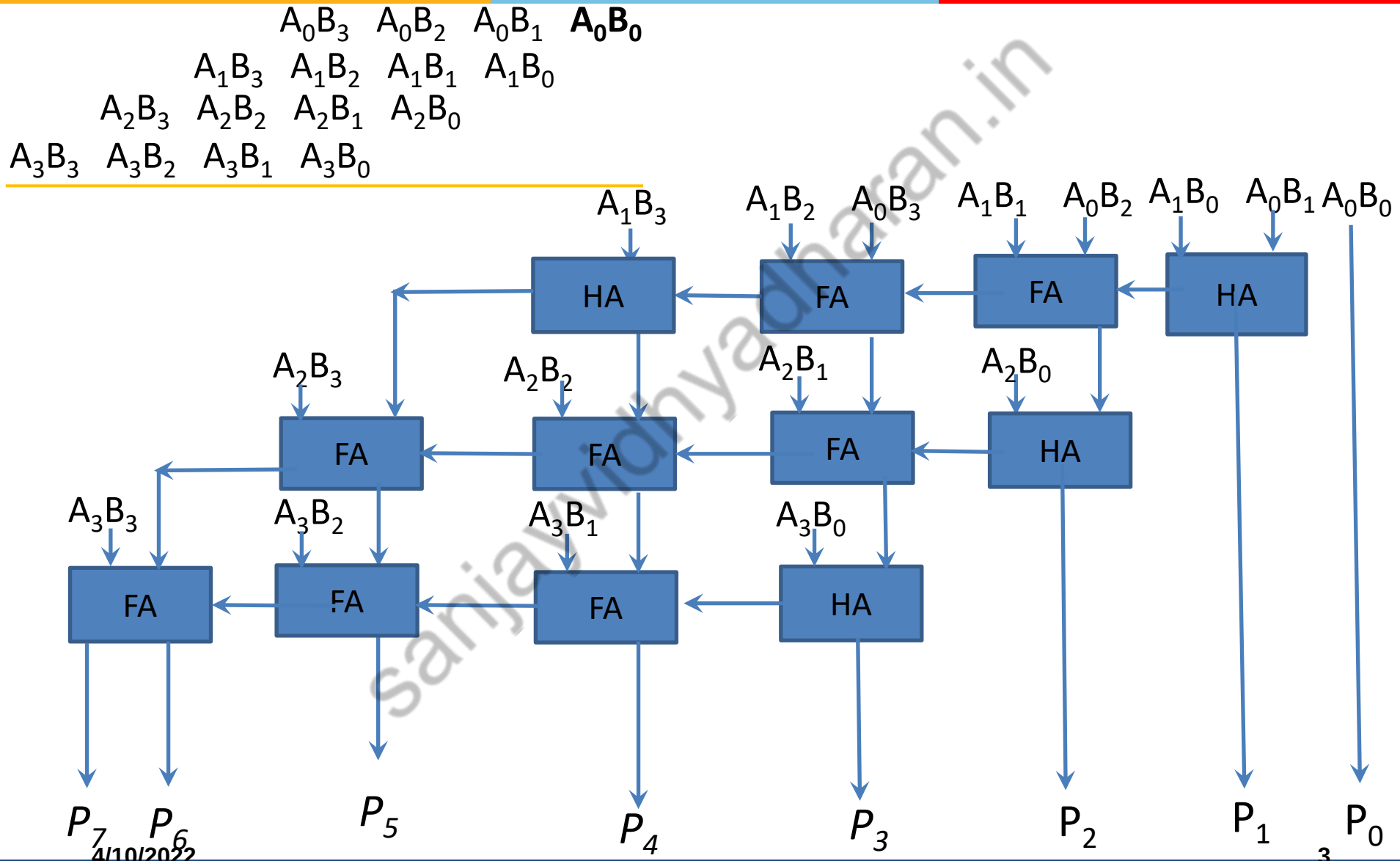
Advanced VLSI Design: 2021-22
Lecture 10
Arithmetic Circuits: Part-2

By Dr. Sanjay Vidhyadharan

The Binary Multiplication

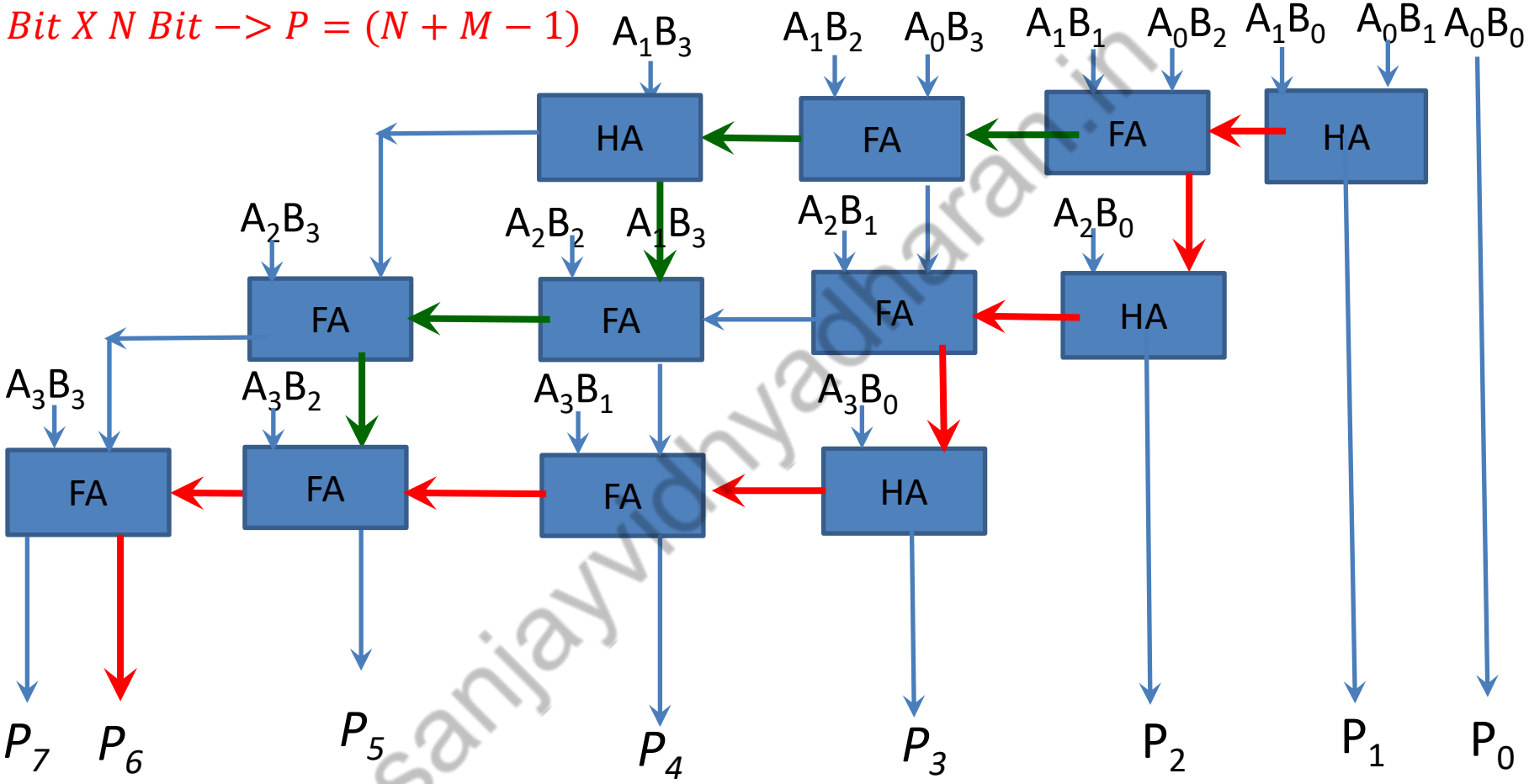


The Array Multiplier



The Array Multiplier

$M\ Bit\ X\ N\ Bit \rightarrow P = (N + M - 1)$

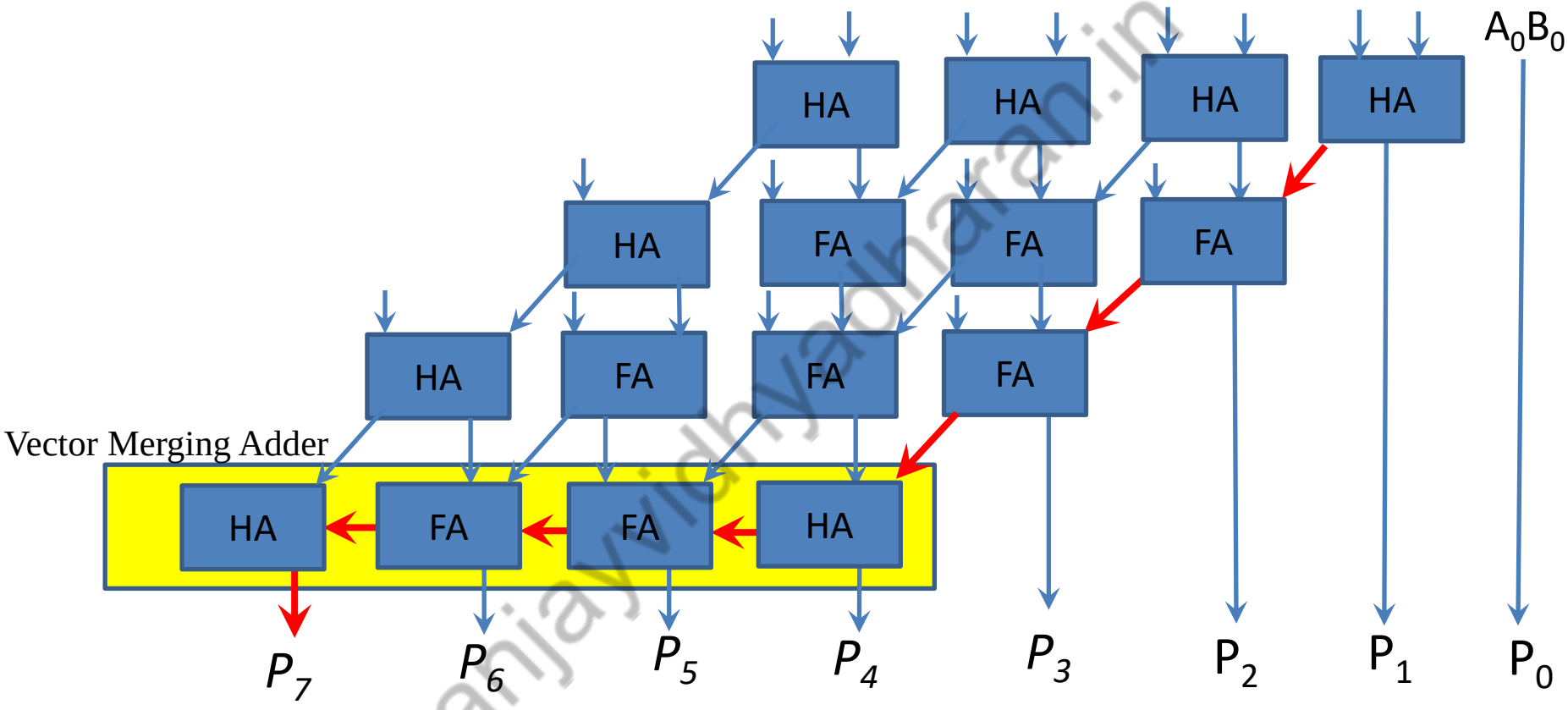


Require $(N-1)$ *M – Bit Adders*

Require M X N AND Gates

$$t_{mul} = [(M-1) + (N-2)]t_{carry} + (N-1)t_{sum} + t_{and}$$

The Carry-Save Multiplier

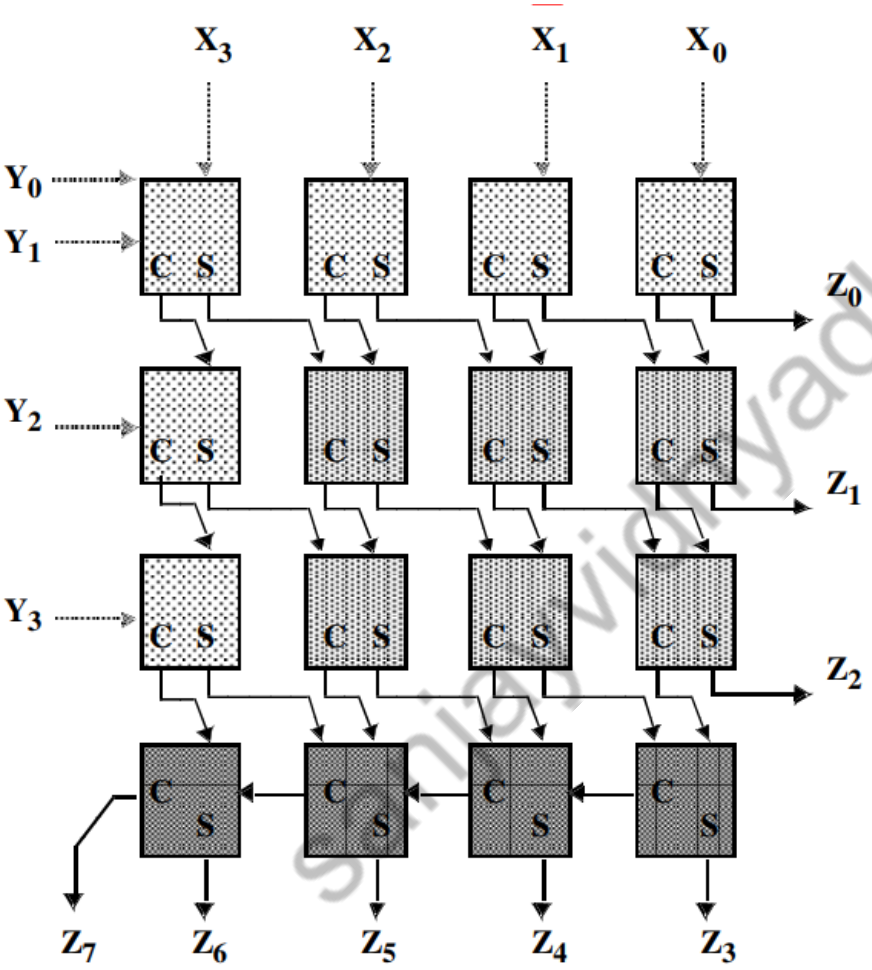


Require N M − Bit Adders

Require M X N AND Gates

$$t_{mul} = t_{and} + (N-1)t_{carry} + t_{merge}$$

Multiplier Floorplan



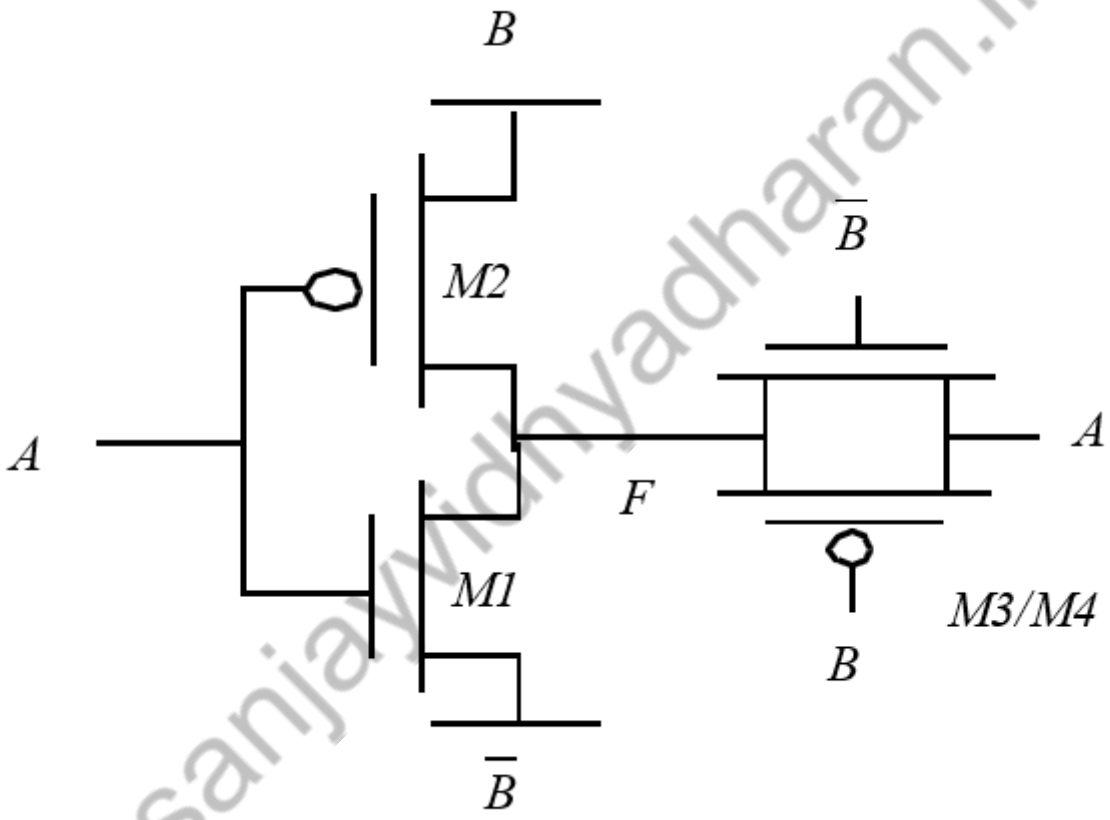
HA Multiplier Cell

FA Multiplier Cell

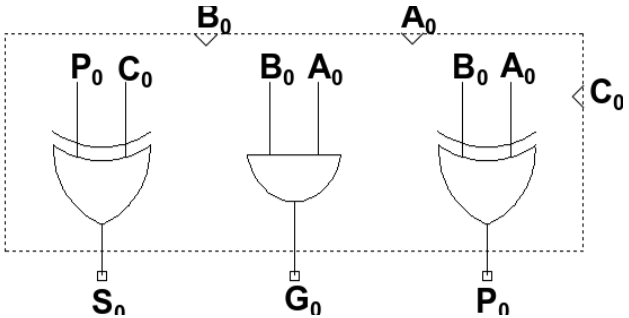
Vector Merging Cell

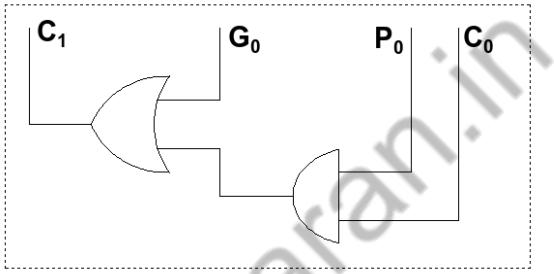
X and Y signals are broadcasted through the complete array.

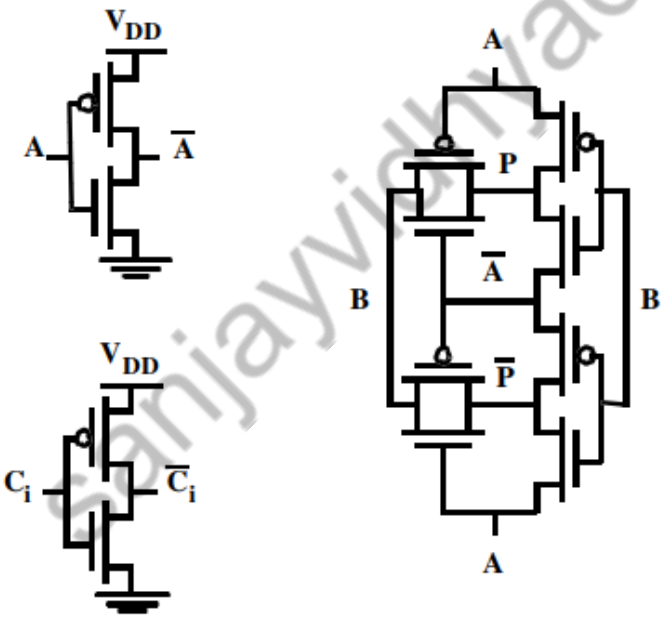
Transmission Gate XOR

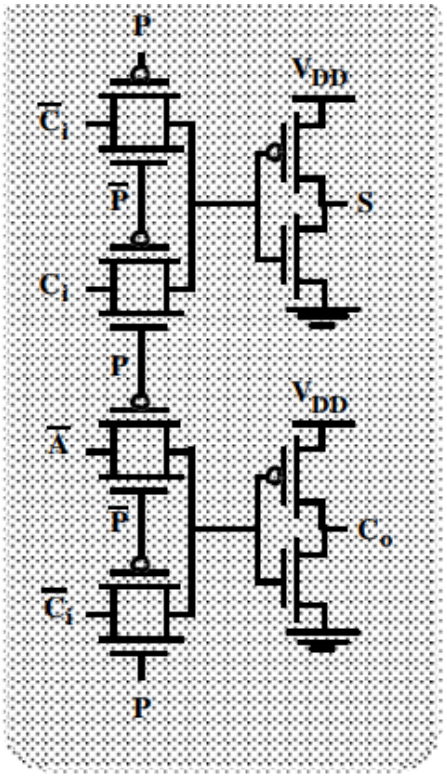


Adder Cells in Array Multiplier



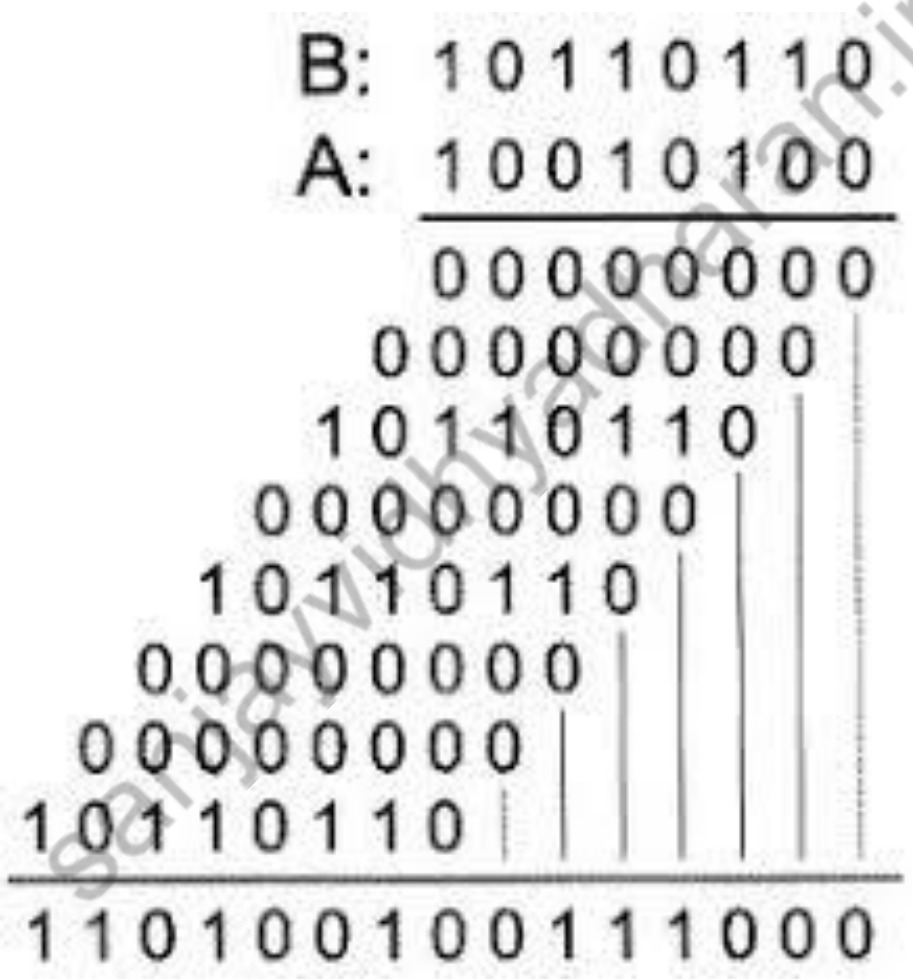




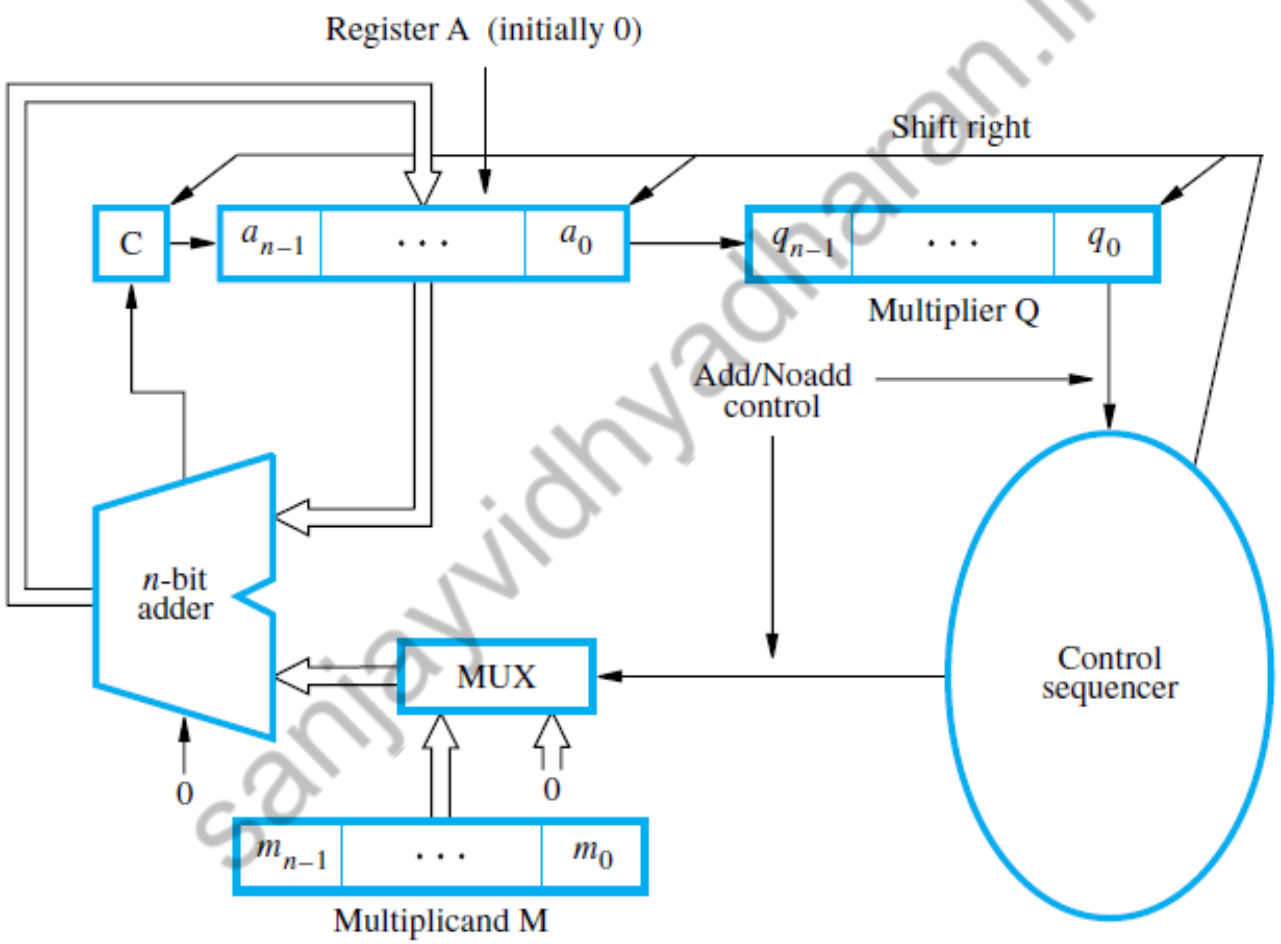


Identical Delays for Carry and Sum

Sequential Multiplier



Booth Algorithm



(a) Register configuration

Booth Algorithm

Example:

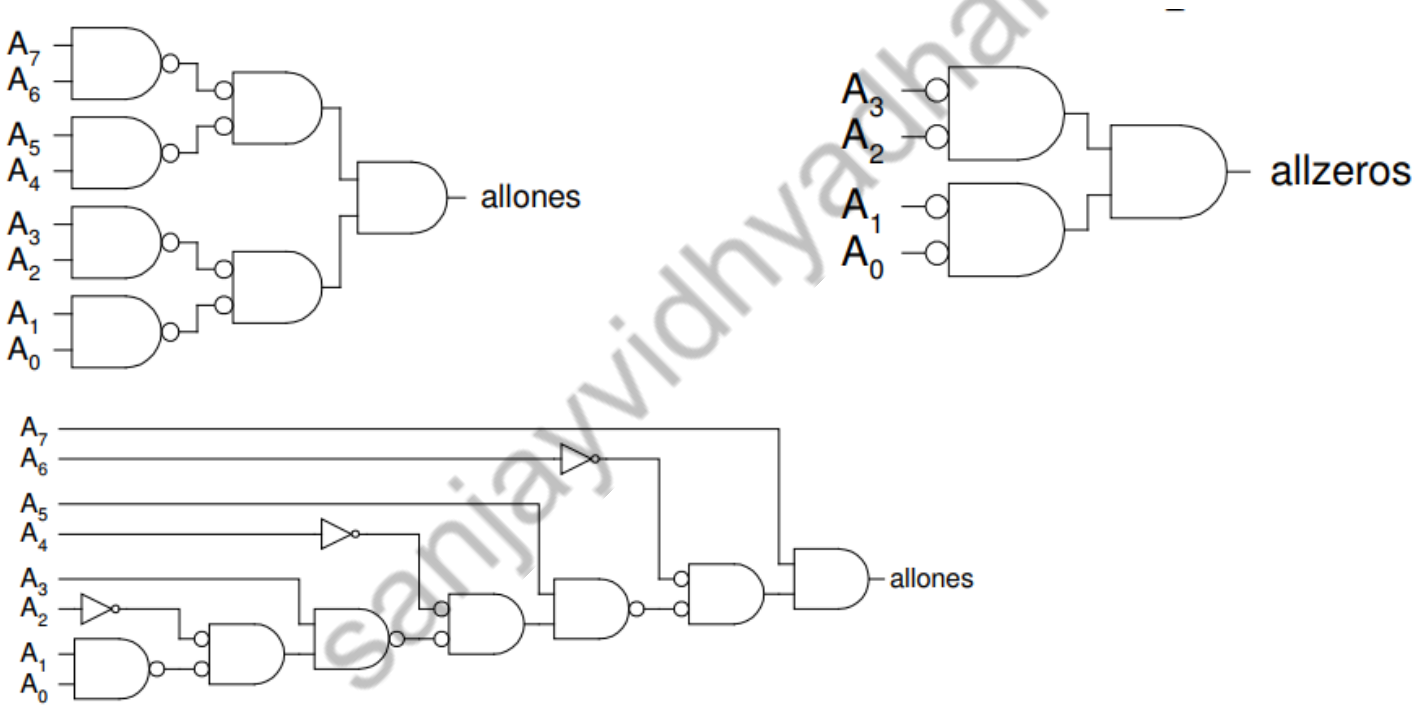
3x5				
	M	A	Q	
Initial Condition	0011	00000	0101	
Clk ↑ (Load)	0011	00011	0101	1
Clk ↓ (Shift Right)	0011	00001	1010	
Clk ↑ (Load)	0011	00001	1010	2
Clk ↓ (Shift Right)	0011	00000	1101	
Clk ↑ (Load)	0011	00011	1101	3
Clk ↓ (Shift Right)	0011	00001	1110	
Clk ↑ (Load)	0011	00001	1110	4
Clk ↓ (Shift Right)	0011	00000	1111	

Comparators

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

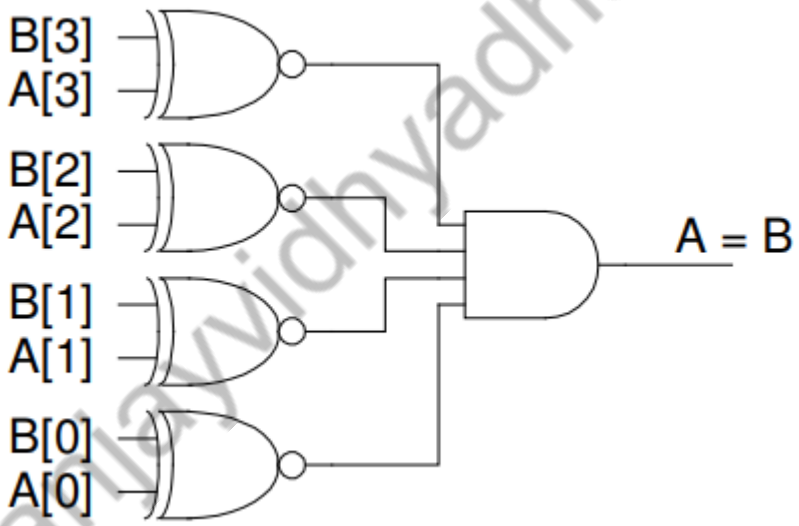
1's & 0's Detectors

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



Equality Comparator

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



Magnitude Comparator

1	0	1
1	1	0

Magnitude Comparator

1	0	1
1	1	0

Magnitude Comparator

1-Bit Comparator

A_0

B_0

Let G_0 indicate greater, L_0 indicate Lesser and E_0 indicate equal

A_0	B_0	G_0	L_0	E_0
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$G_0 = A_0B_0'$

$L_0 = A_0'B_0$

$E_0 = (A_0 \oplus B_0)' = (A_0 \odot B_0) = (G_0 + L_0)'$

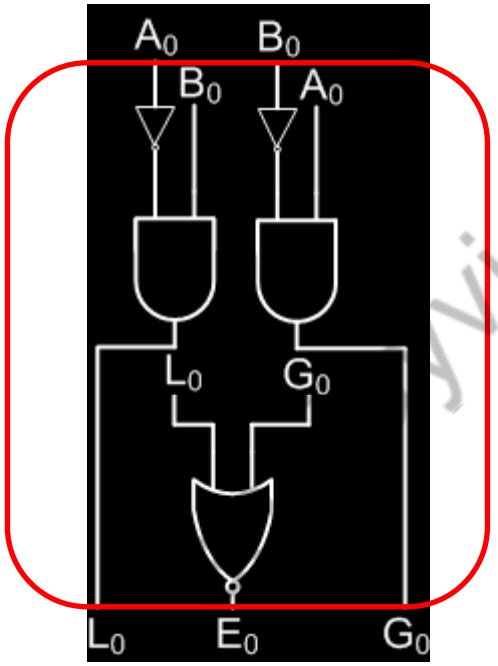
Magnitude Comparator

1-Bit Comparator

A_0

B

0



$G_0 = A_0B_0'$

$L_0 = A_0'B_0$

$E_0 = (G_0 + L_0)' = (A_0 \oplus B_0)'$

B_0 A_0

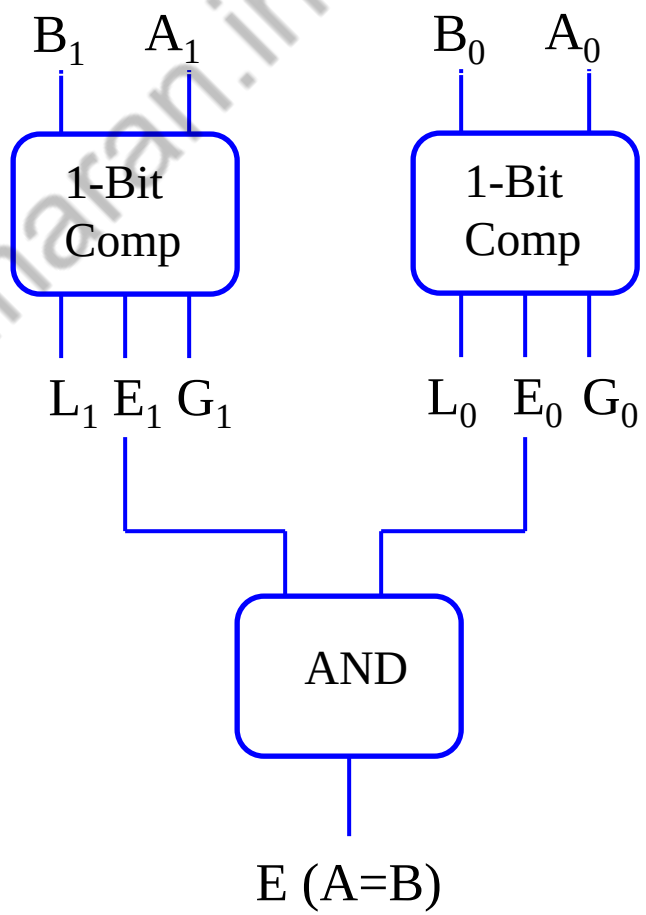
1-Bit Comp

L_0 E_0 G_0

Magnitude Comparator

2-Bit Comparator

A	A_1A_0
B	B_1 B_0



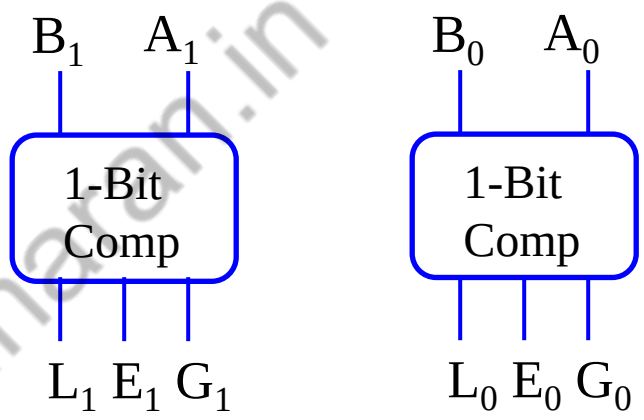
A = B When $A_0 = B_0$ and $A_1 = B_1$

$E = E_0 . E_1$

Magnitude Comparator

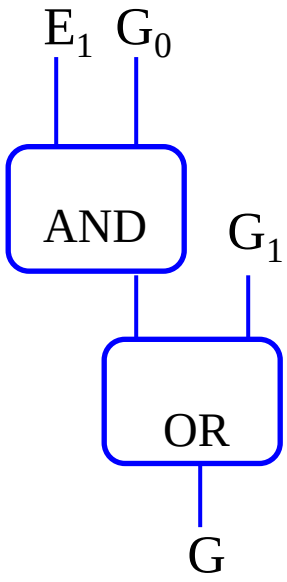
2-Bit Comparator

A	A_1A_0
B	B_1 B_0



A > B When $A_1 > B_1$ OR
When $A_1 = B_1$ and $A_0 > B_0$

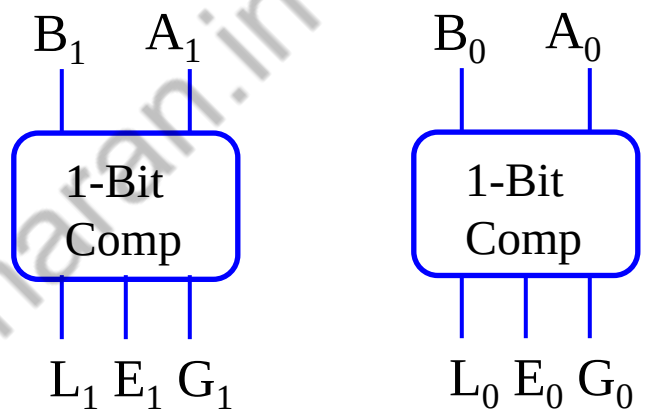
$G = G_1 + E_1 . G_0$



Magnitude Comparator

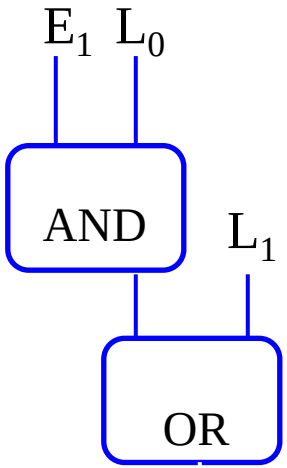
2-Bit Comparator

A	A_1A_0
B	B_1 B_0



A < B When $A_1 < B_1$ OR
When $A_1 = B_1$ and $A_0 < B_0$

$$L = L_1 + E_1 . L_0$$



Magnitude Comparator

1-Bit Comparator

$G = G_0 \quad L = L_0 \quad E = E_0$

2-Bit Comparator

$G = G_1 + E_1 . G_0 \qquad L = L_1 + E_1 . L_0 \qquad E = E_1 . E_0$

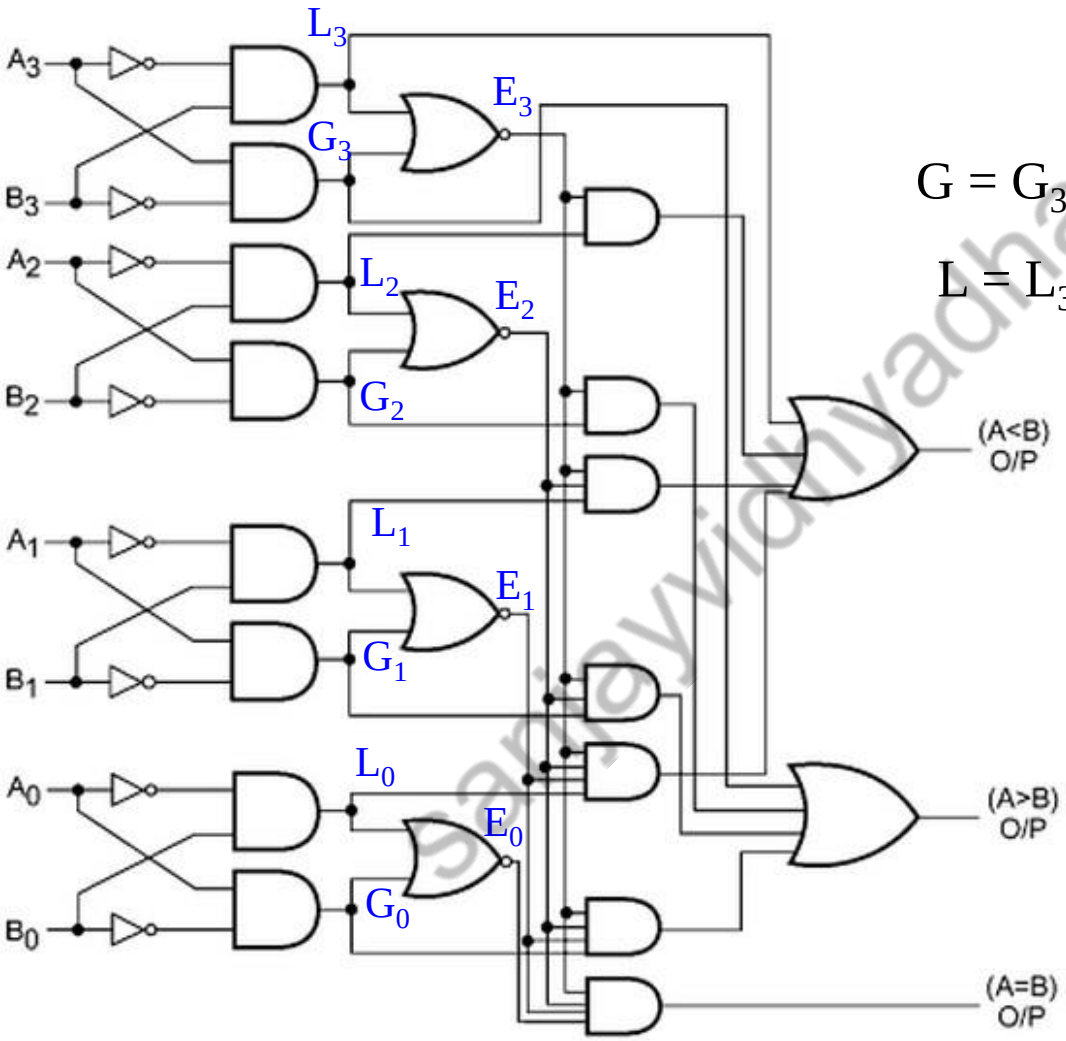
3-Bit Comparator

$G = G_2 + E_2 . G_1 + E_2 . E_1 . G_0$

$L = L_2 + E_2 . L_1 + E_2 . E_1 . L_0$

$E = E_2 . E_1 . E_0$

Magnitude Comparator



4-Bit Comparator

$$E = E_2 . E_1 . E_0$$

$$G = G_3 + E_3.G_2 + E_3.E_2. G_1 + E_3.E_2. E_1. G_0$$

$$L = L_3 + E_3.L_2 + E_3.E_2. L_1 + E_3.E_2. E_1. L_0$$

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

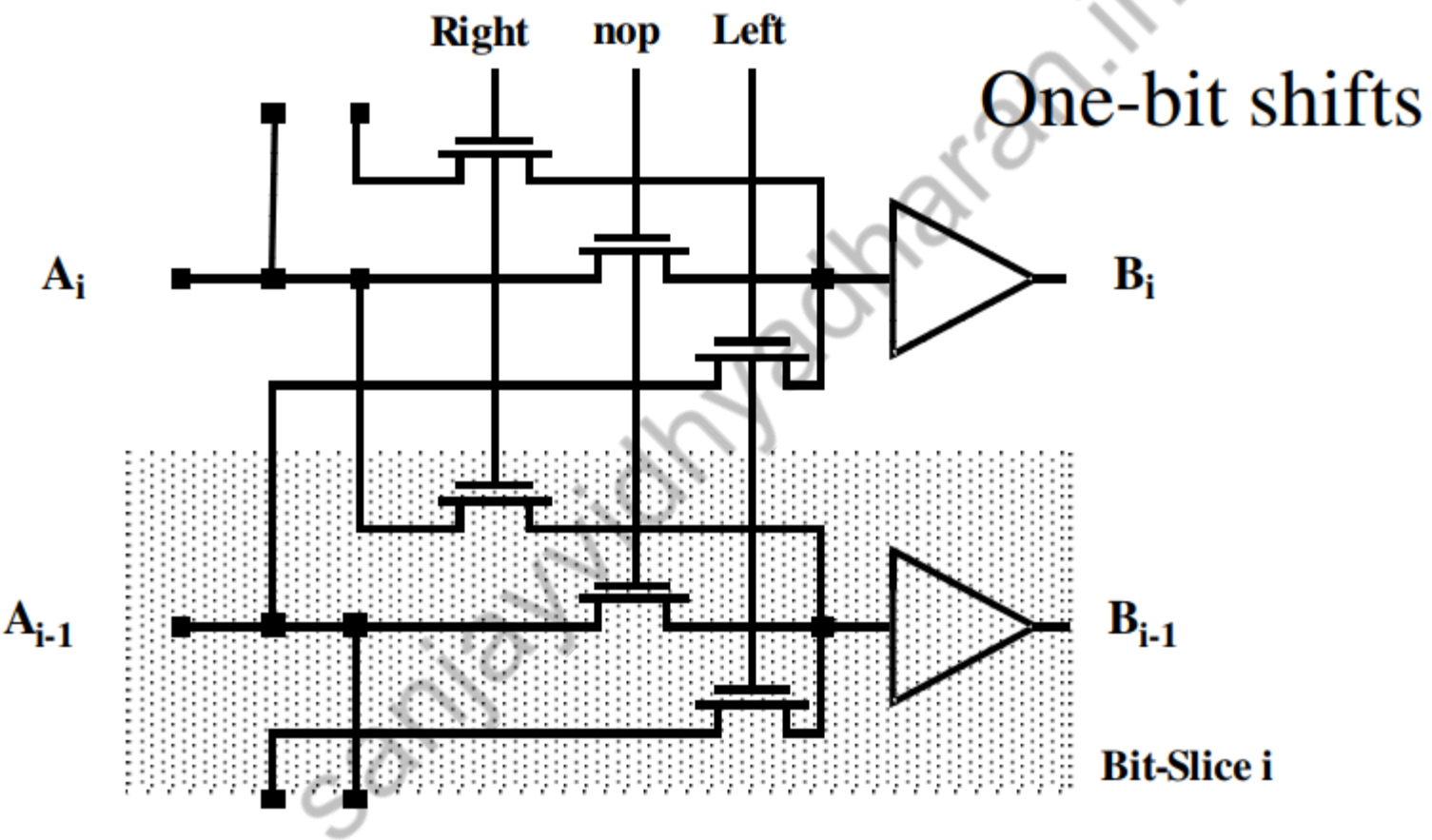
• 1011 ASR1 = 1101 1011 ASL1 = 0110

• Rotate:

– Shifts number left or right and fills with lost bits

• 1011 ROR1 = 1101 1011 ROL1 = 0111

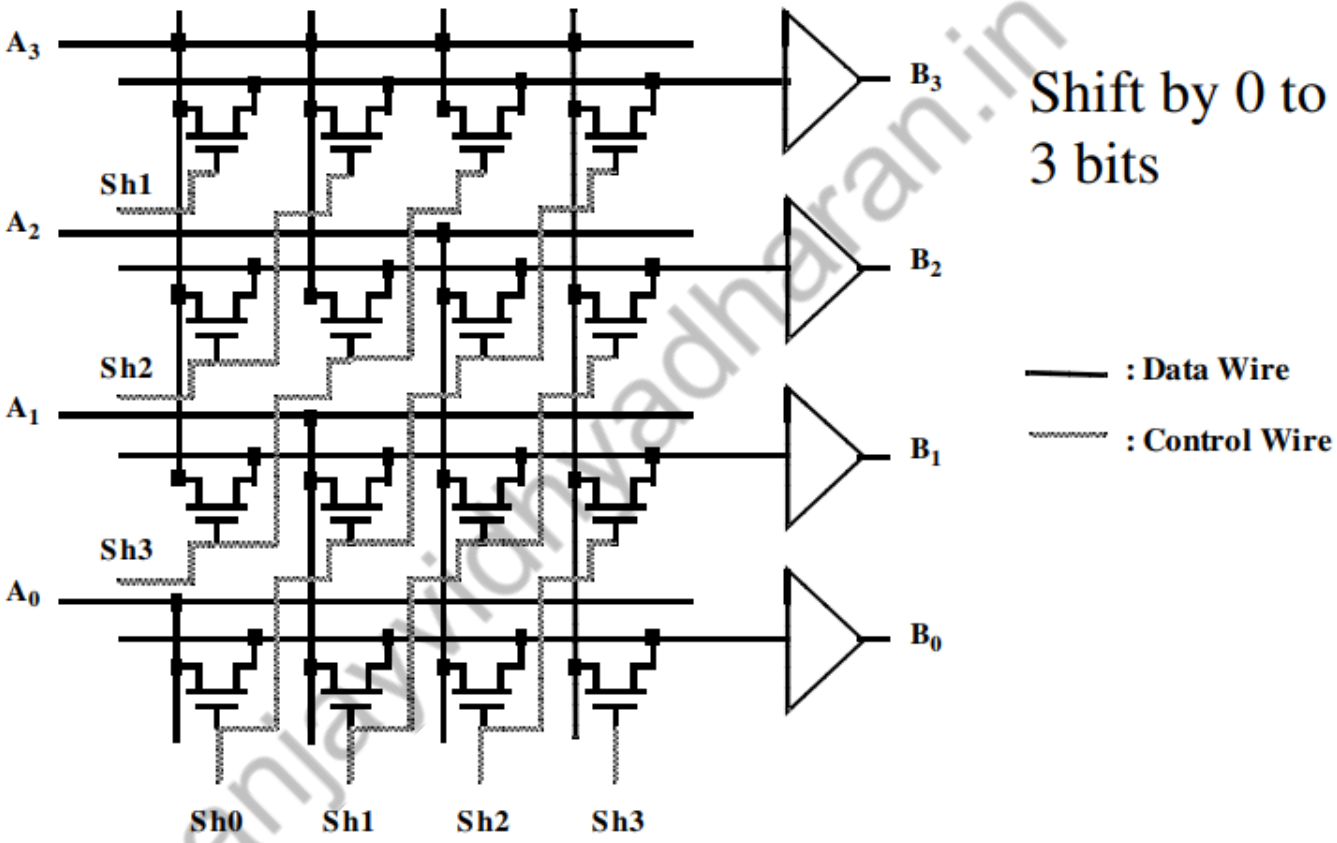
The Binary Shifter



Multi-bit Shifters

- Cascade one-bit shifters?
- Complex, unwieldy, slow for larger number of shifts
- Two other types of shifters
  - Barrel shifter
  - Logarithmic shifter

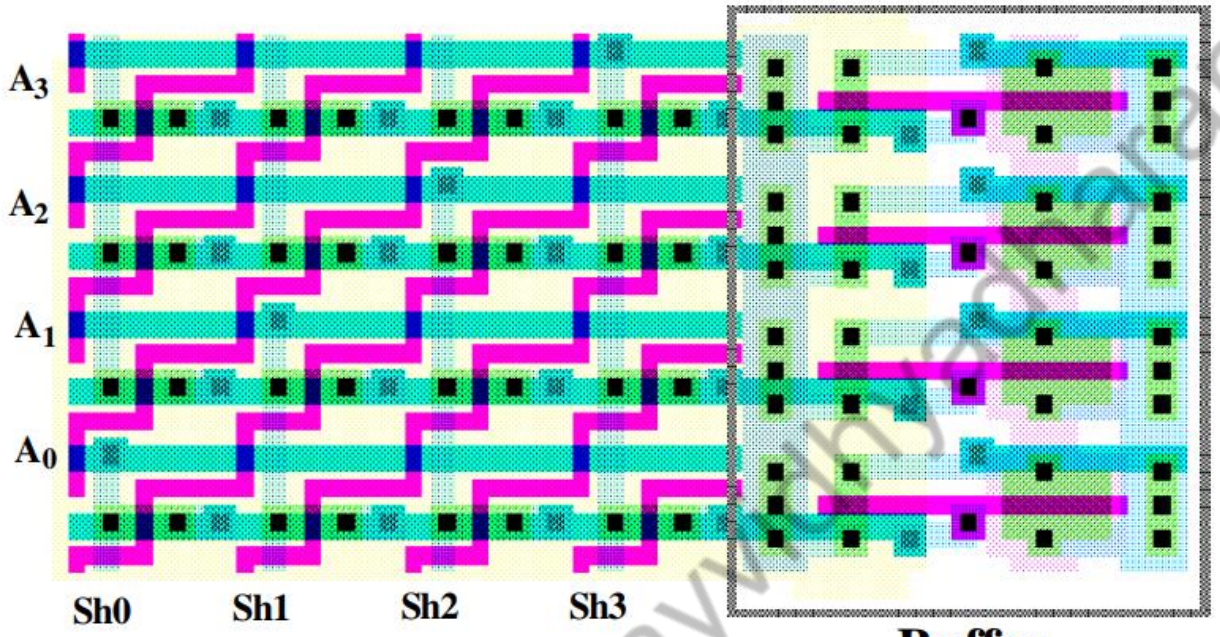
The Barrel Shifter

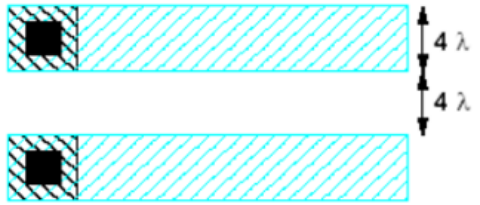


Area dominated by wiring

Propagation delay is theoretically constant

The Barrel Shifter





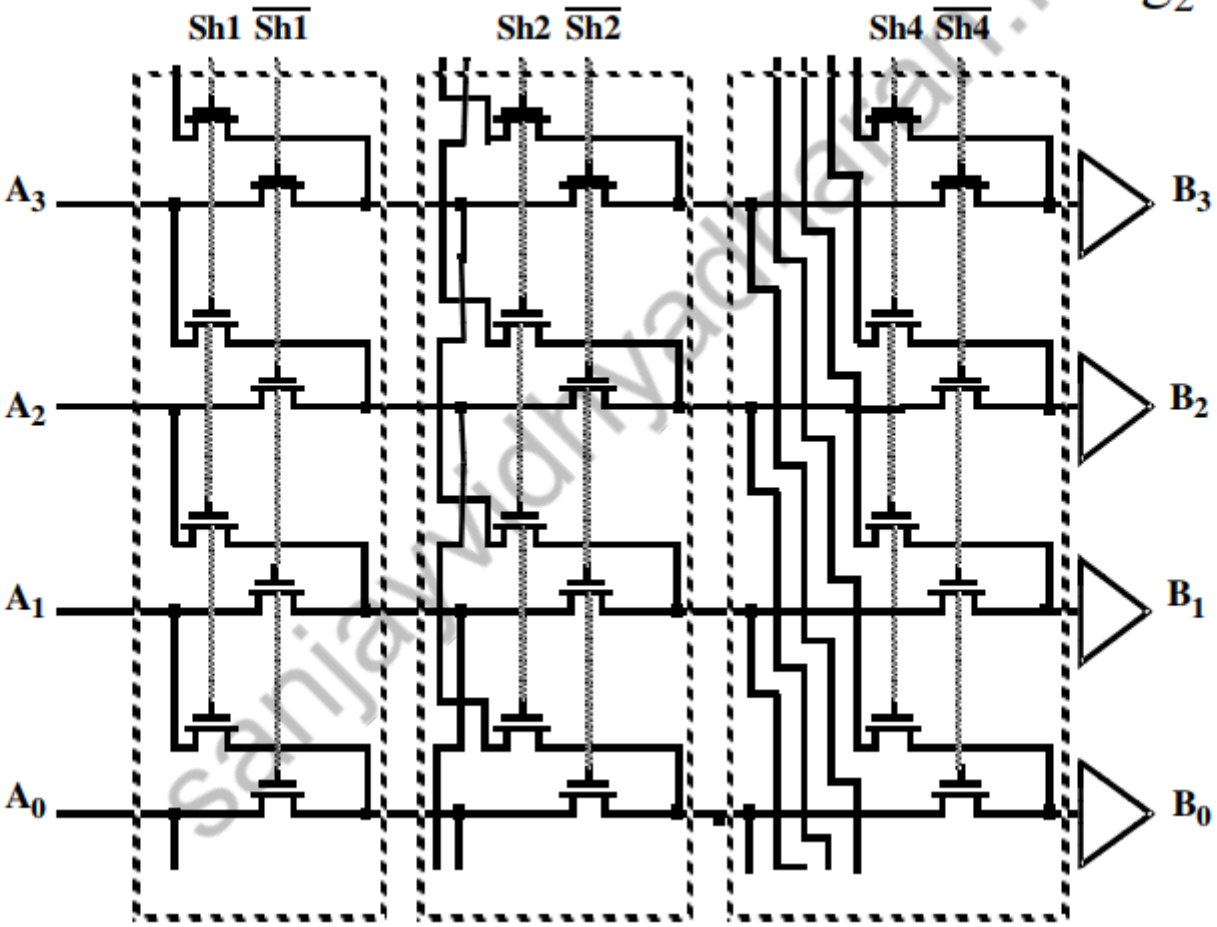
$Width_{barrel} \sim 2\ p_m\ M$

p_m = metal/poly pitch

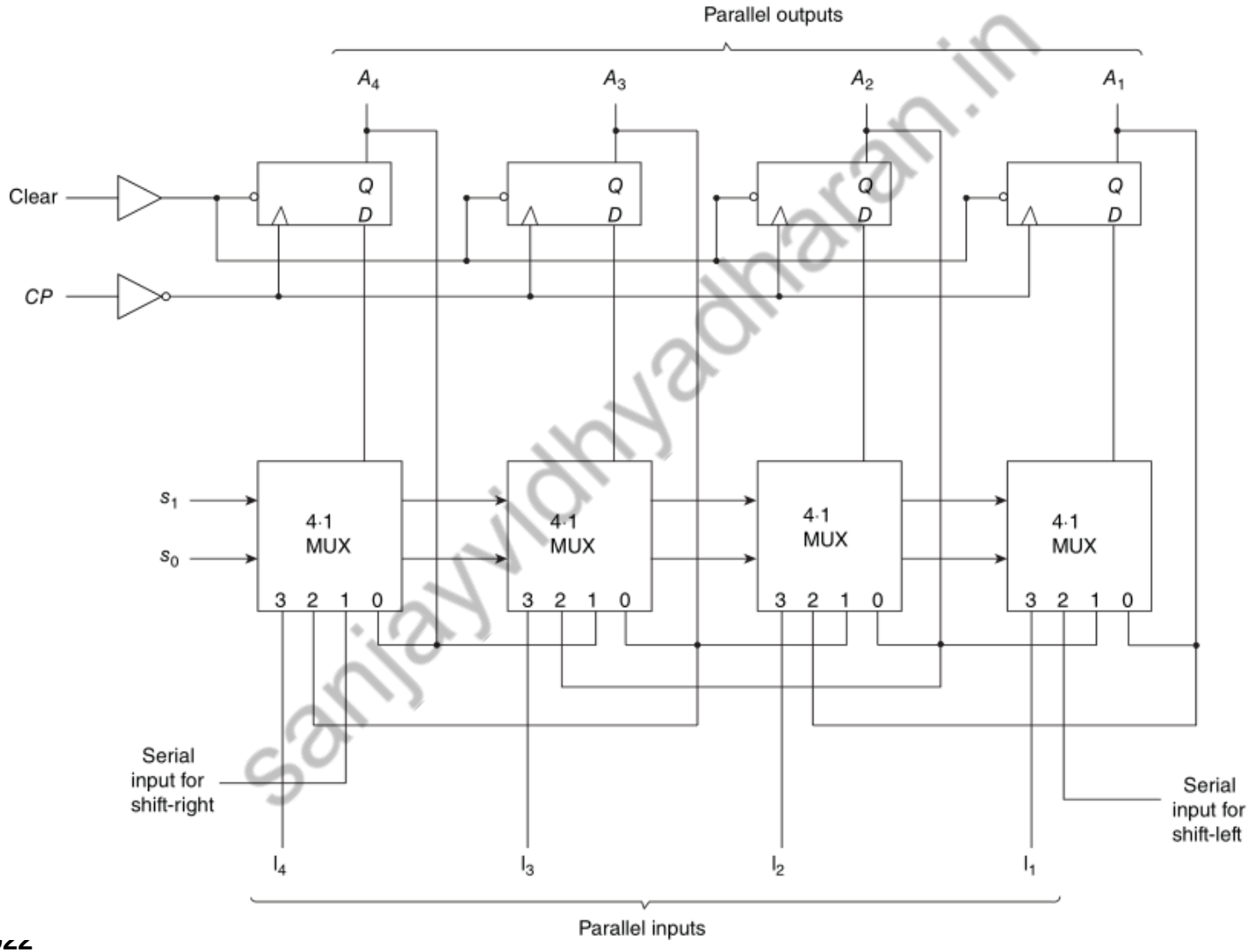
M = no. of shifts

Logarithmic Shifter

- Shifter with maximum shift width M consists of $\log_2 M$ stages



Shifter Using Mux



Thank you