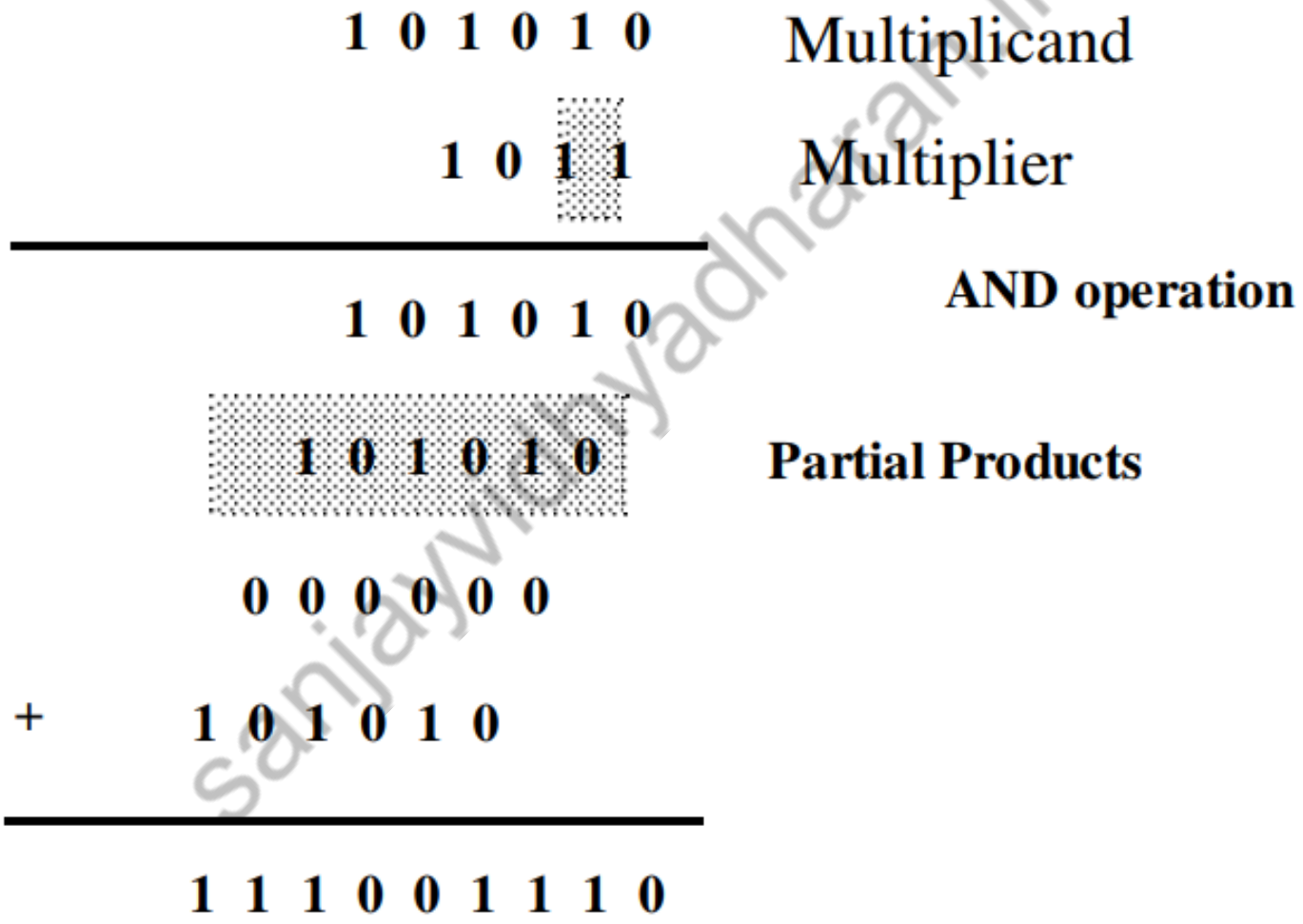
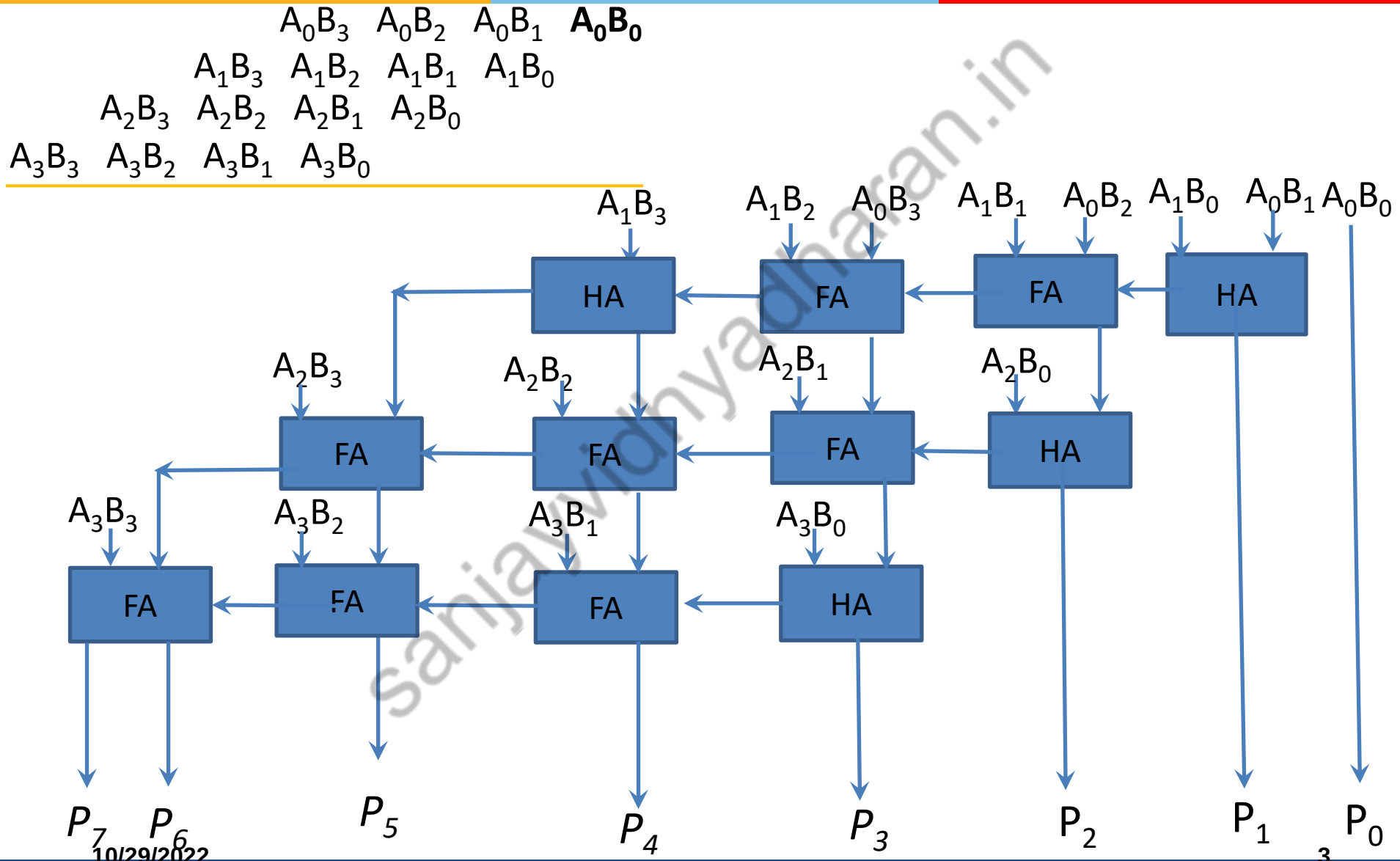
Advanced VLSI Design: 2022-23
Lecture 12
Arithmetic Circuits: Part-2

By Dr. Sanjay Vidhyadharan

The Binary Multiplication

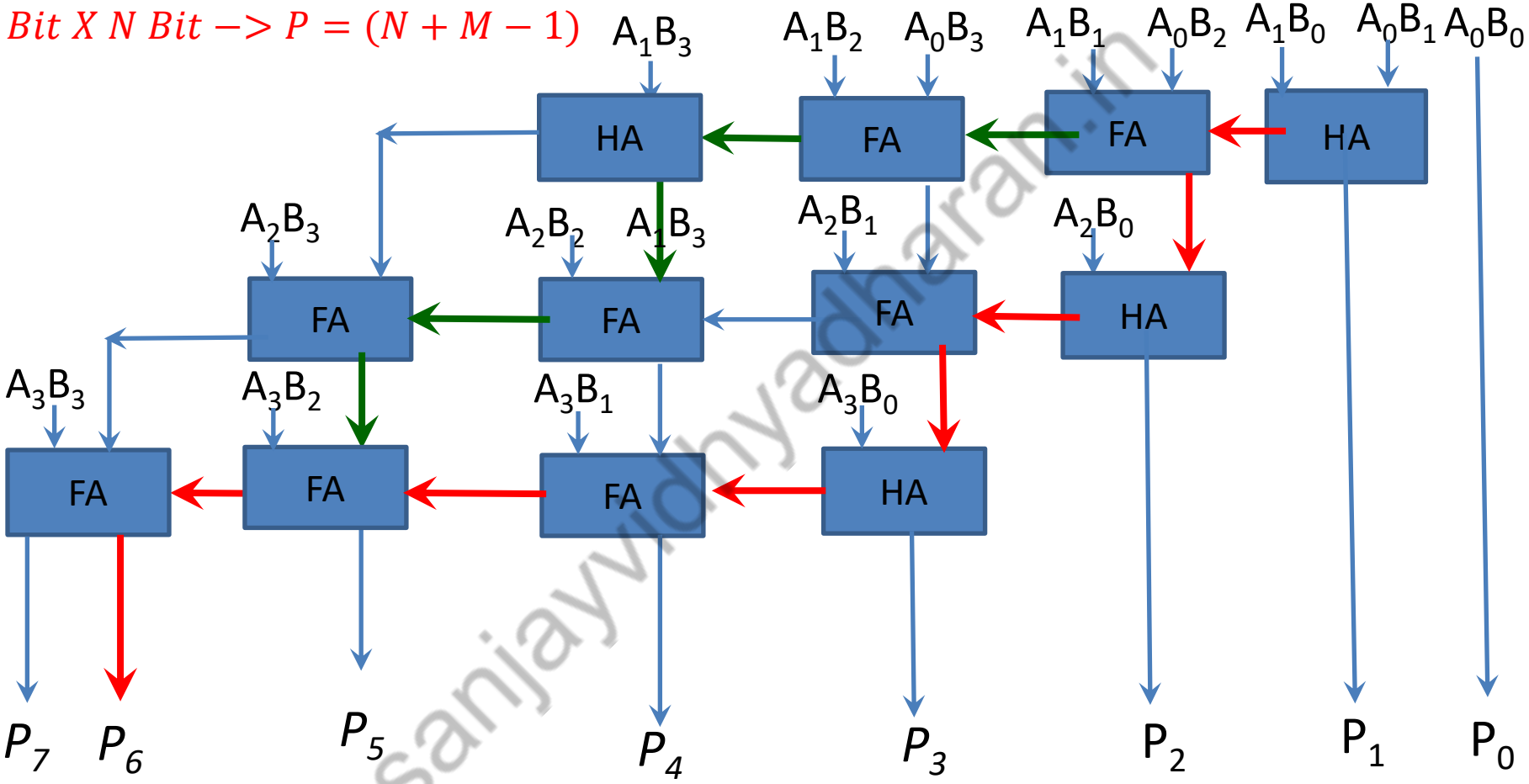


The Array Multiplier



The Array Multiplier

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

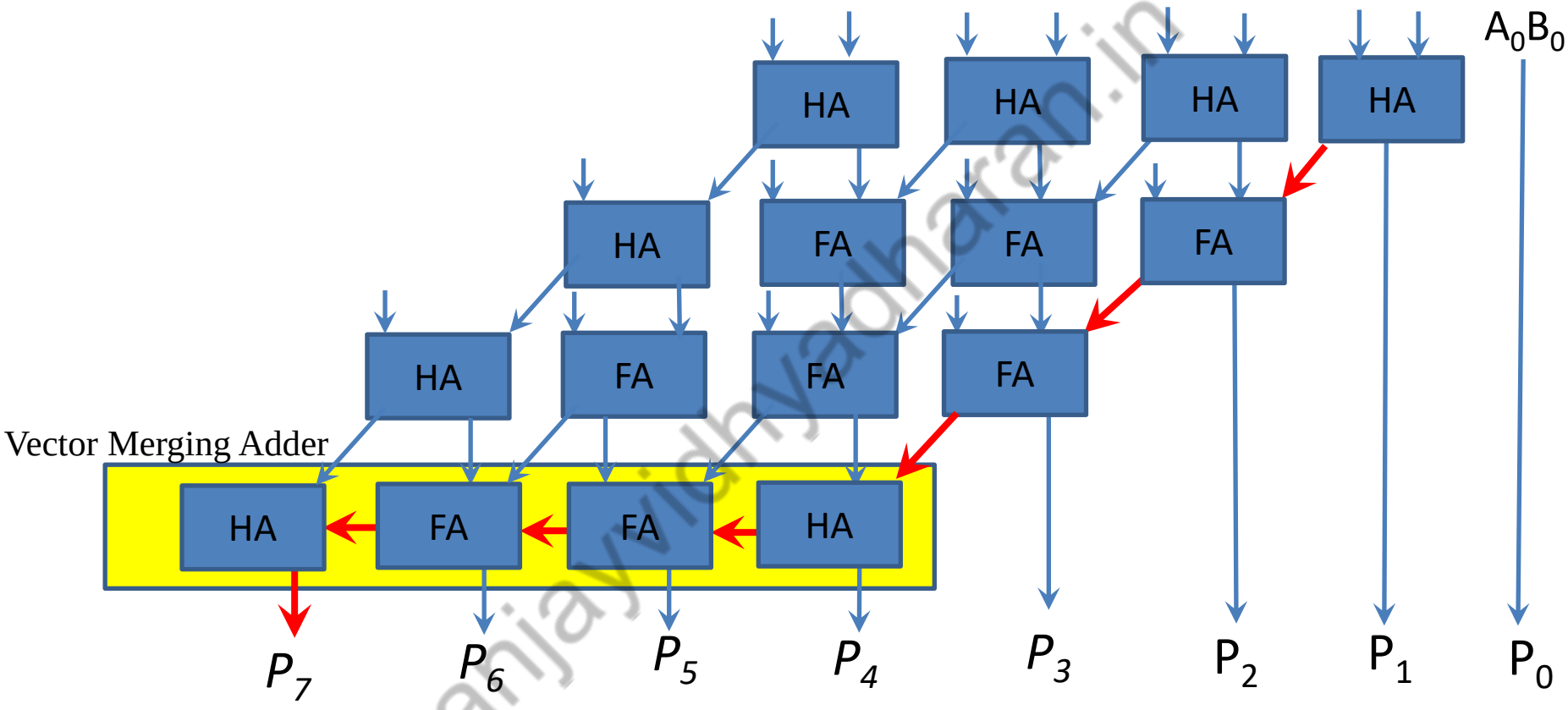


Require $(N-1)$ *M – Bit Adders*

Require M X N AND Gates

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

The Carry-Save Multiplier

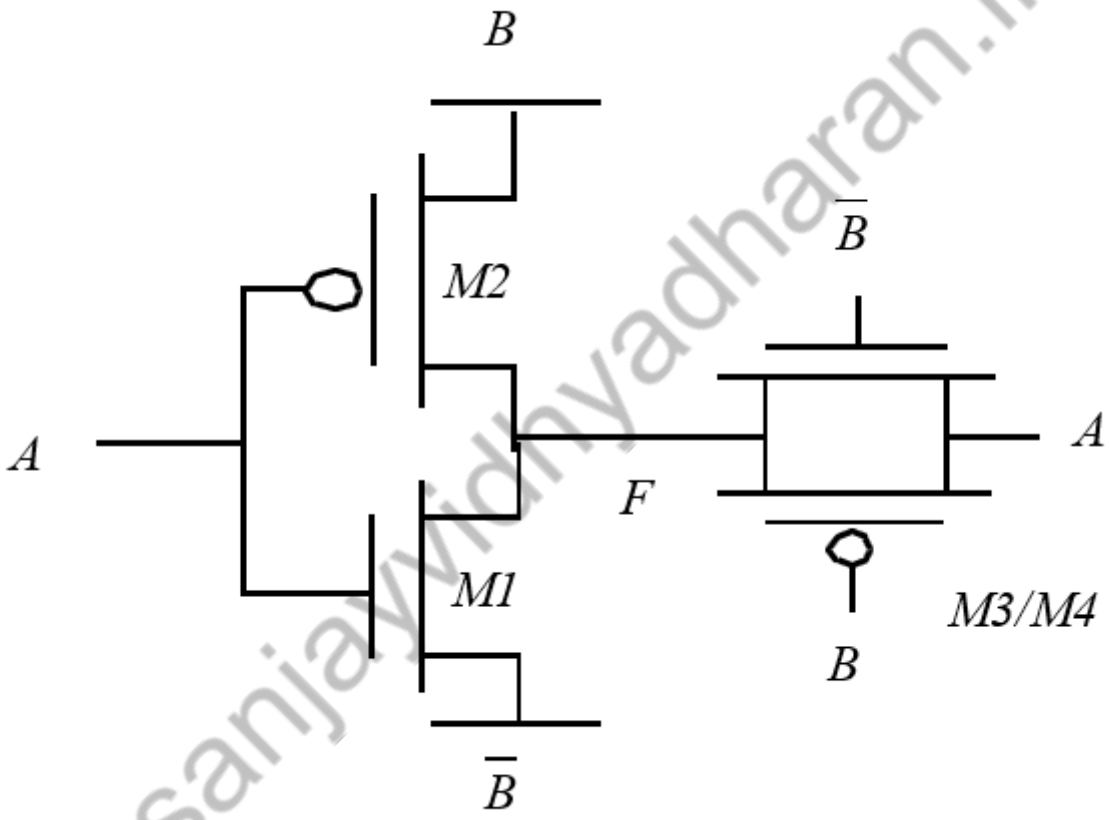


Require N M − Bit Adders

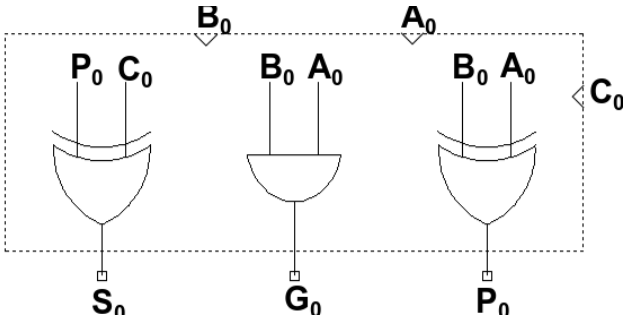
Require M X N AND Gates

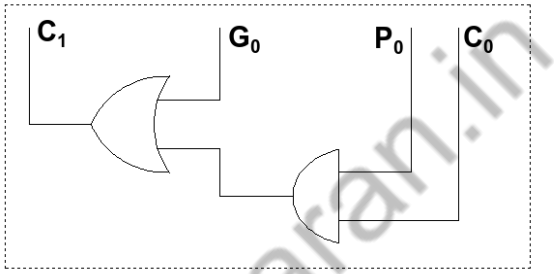
$$t_{mul} = t_{and} + (N-1)t_{carry} + t_{merge}$$

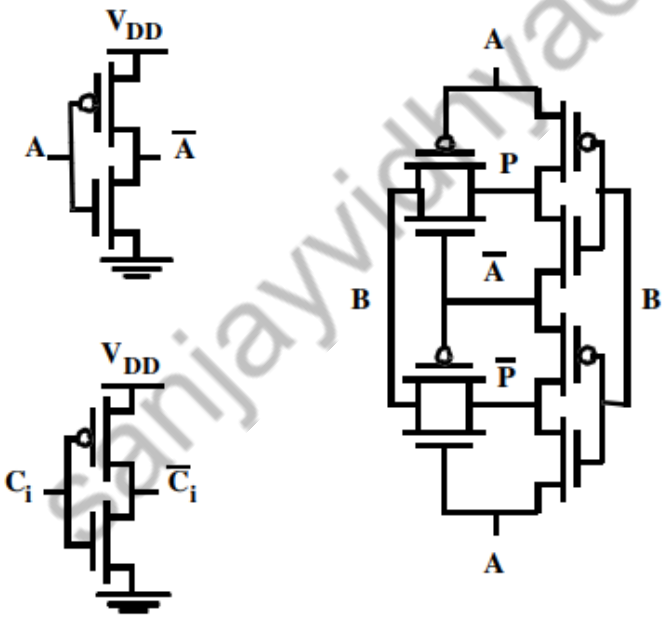
Transmission Gate XOR

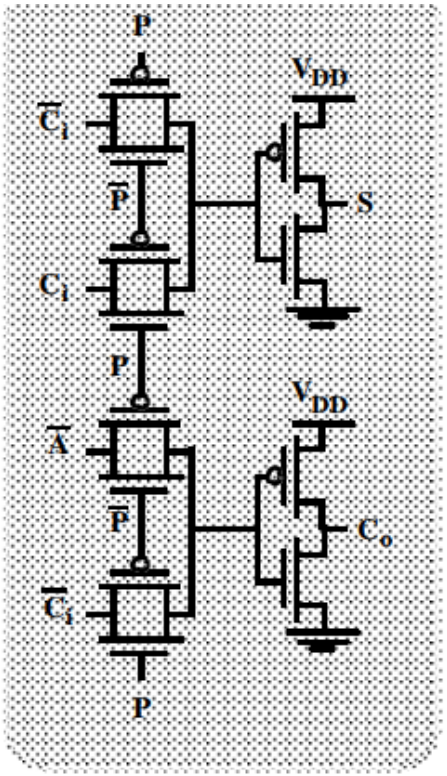


Adder Cells in Array Multiplier



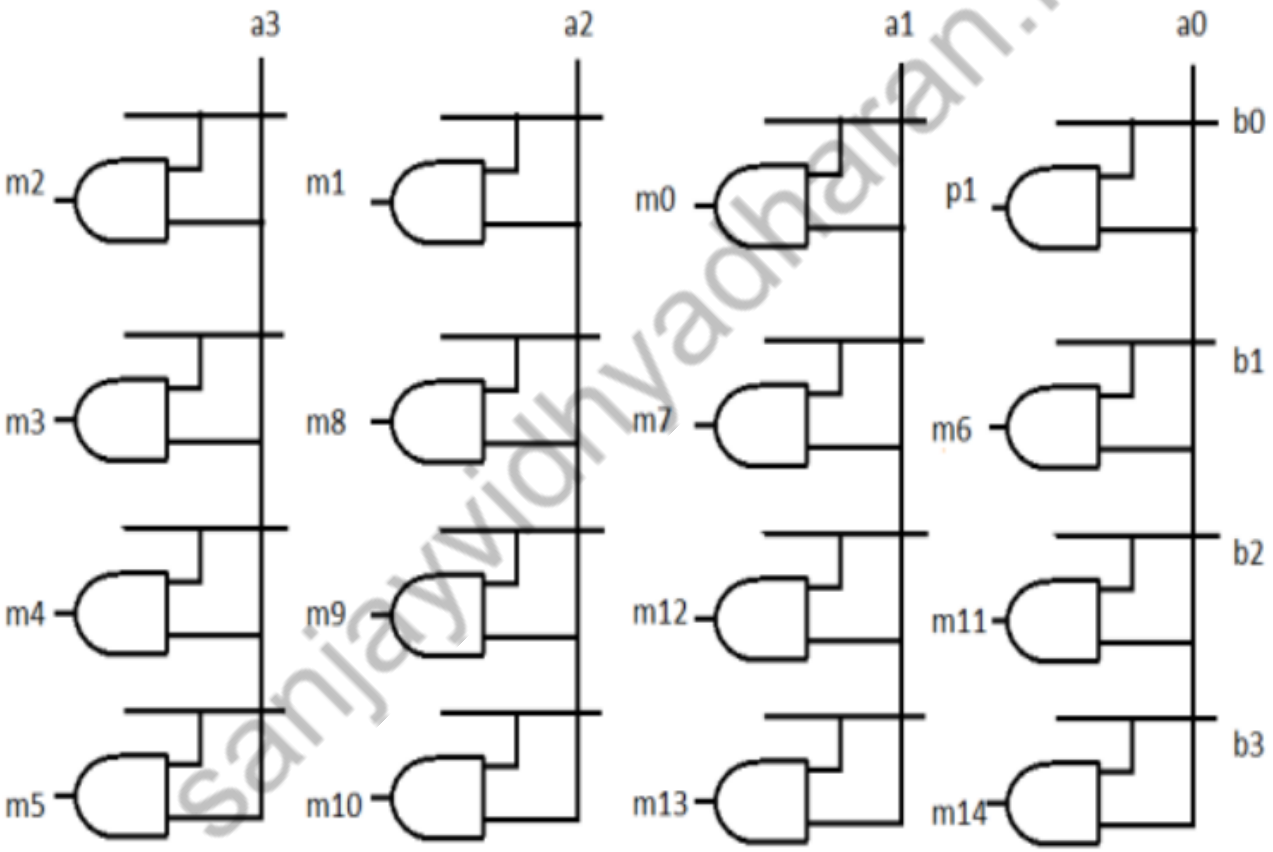






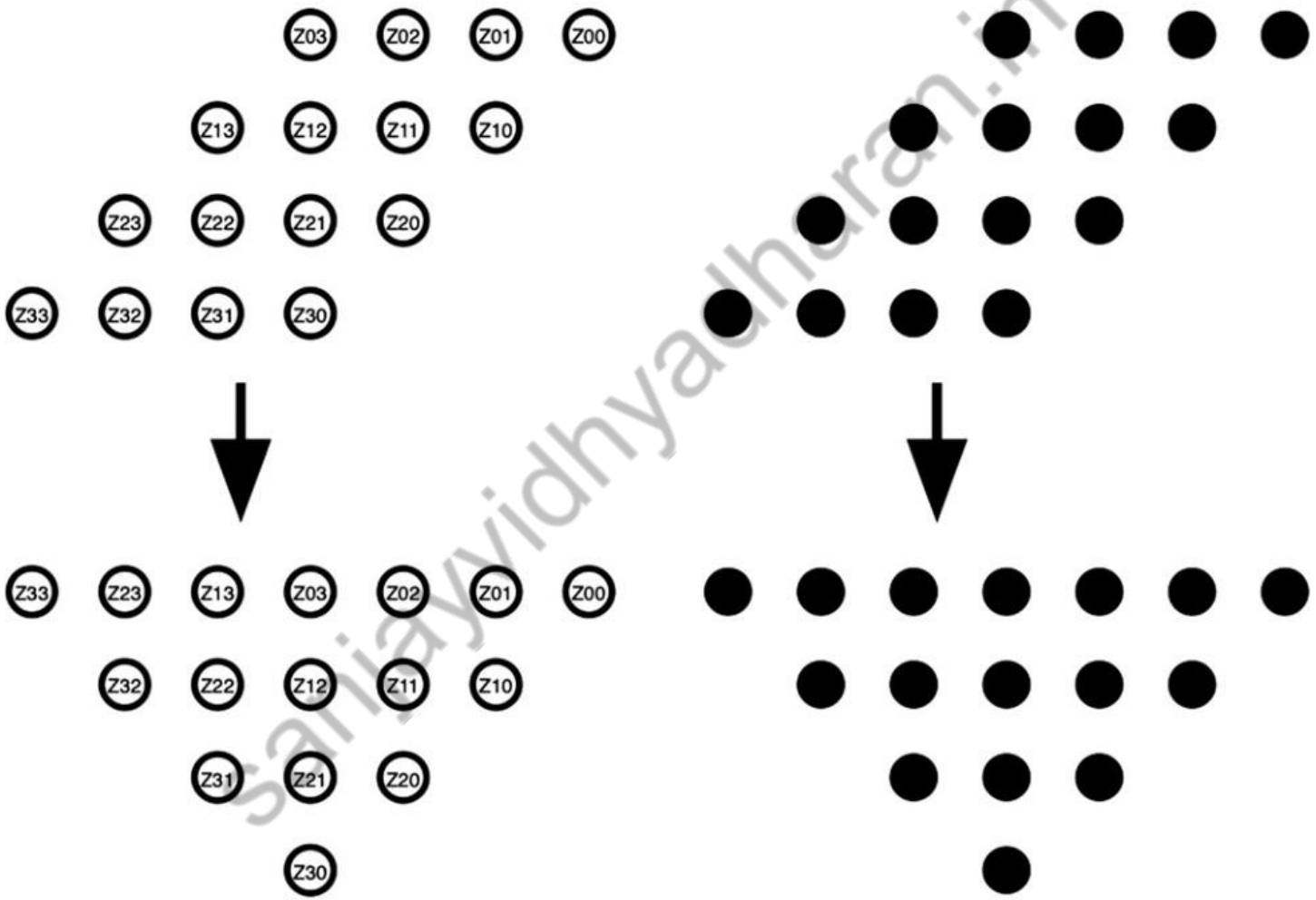
Identical Delays for Carry and Sum

Wallace Tree Multiplication



Product terms generated by a collection of AND gates

Wallace Tree Multiplication



Wallace Tree Multiplication

Step -1

			a3	a2	a1	a0
			b3	b2	b1	b0
p33	p23	p13	p03	p02	p01	p00
	p32	p22	p12	p11	p10	
		p31	p21	p20		
			p30	p29		

1. Use FA and HA to reduce column height
2. To use FA/HA previous column should have 3 or more partial products and to use HA previous column should have 2 or more partial products
3. Stop if two bits are available in the column an also in preceding column.

Step -2

			a3	a2	a1	a0
			b3	b2	b1	b0
p33	p23	p13	p03	p02	p01	p00
	p32	p22	p12	p11	p10	
		p31	p21	p20		
			p30			

			a3	a2	a1	a0
			b3	b2	b1	b0
p33	p23	S-3	S-2	S-1	p01	p00
	p32	C-2	p30	p20	p10	
	C-3		C-1			

FA-2 & HA-1

Wallace tree Multiplication

Step -2

			a3 b3	a2 b2	a1 b1	a0 b0
p33	p23	p13	p03	p02	p01	p00
	p32	p22	p12	p11	p10	
		p31	p21	p20		
			p30			

			a3 b3	a2 b2	a1 b1	a0 b0
p33	p23	S-3	S-2	S-1	p01	p00
	p32	C-2	p30	p20	p10	
	C-3		C-1			

FA-2 & HA-1

Step -3

				a3 b3	a2 b2	a1 b1	a0 b0
p33	p23	S-3	S-2	S-1	p01	p00	
	p32	C-2	p30	p20	p10		
	C-3		C-1				

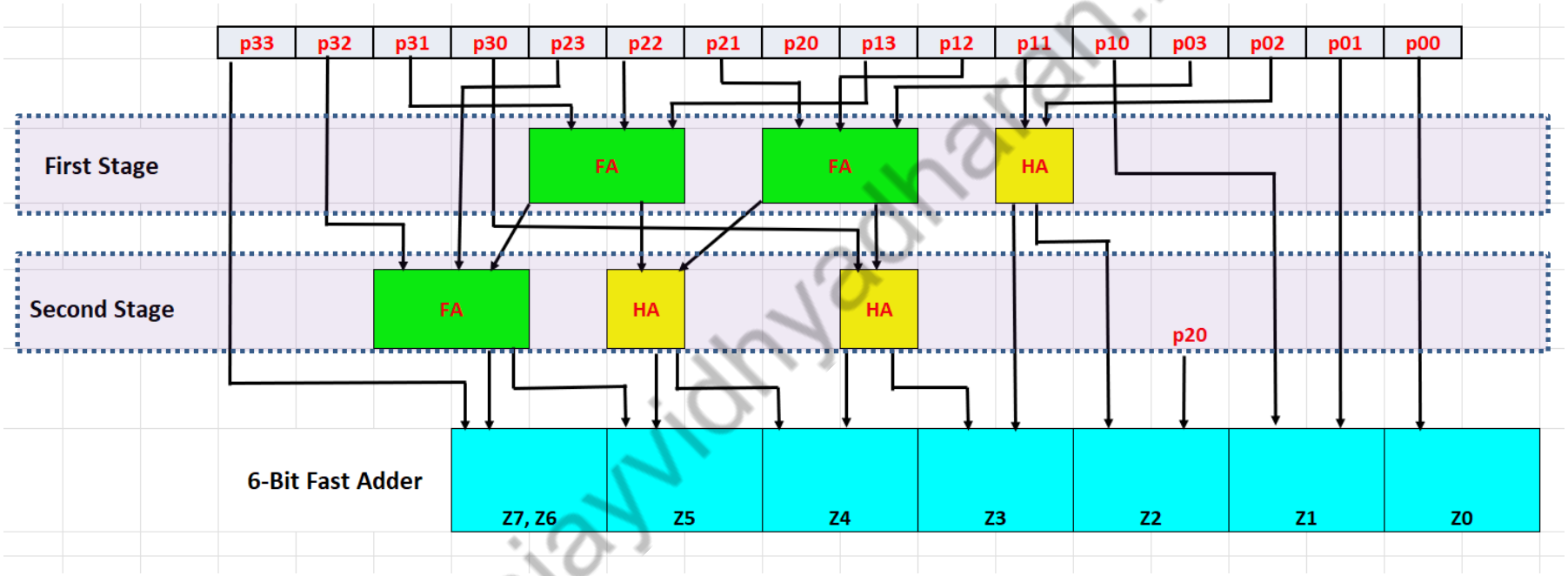
			a3 b3	a2 b2	a1 b1	a0 b0
p33	S-6	S-5	S-4	S-1	p01	p00
C-6	C-5	C-4	C-1	p20	p10	

FA-1 & HA-2

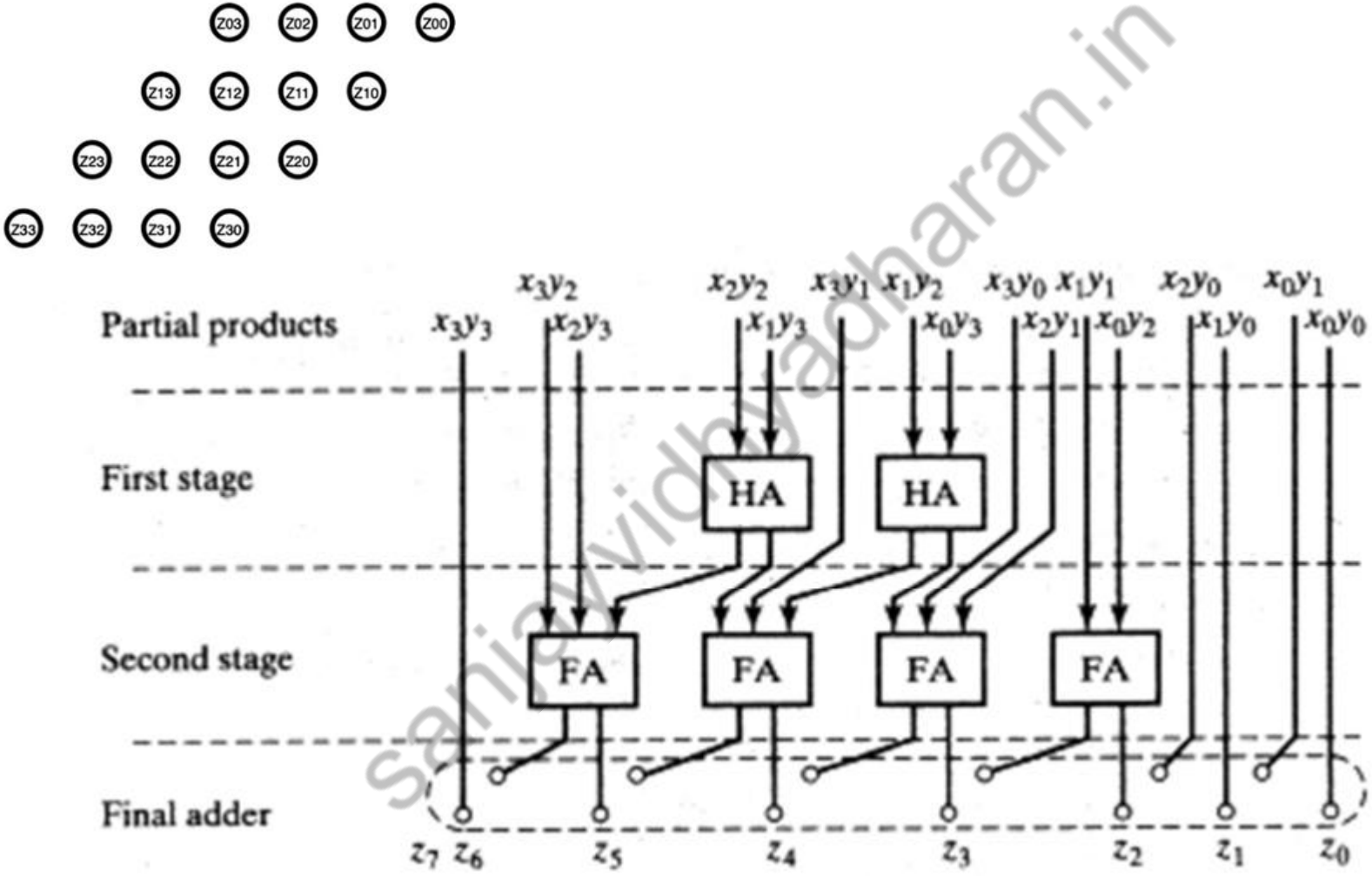
Step -4 Six- Bit Fast Adder

Handbook of Digital CMOS Technology, Circuits, and Systems
Karim Abbas

Wallace Tree Multiplication

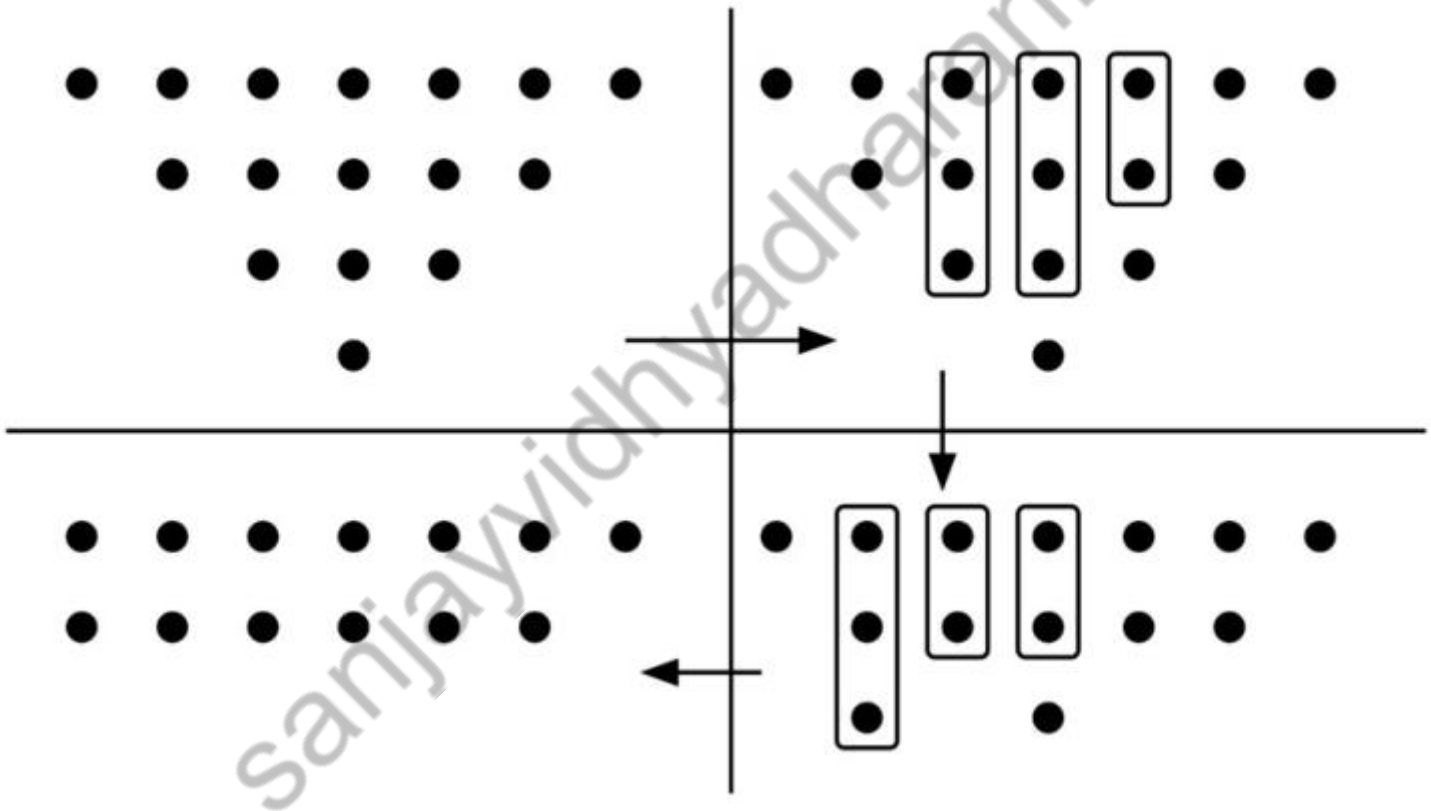


Wallace Tree Multiplication



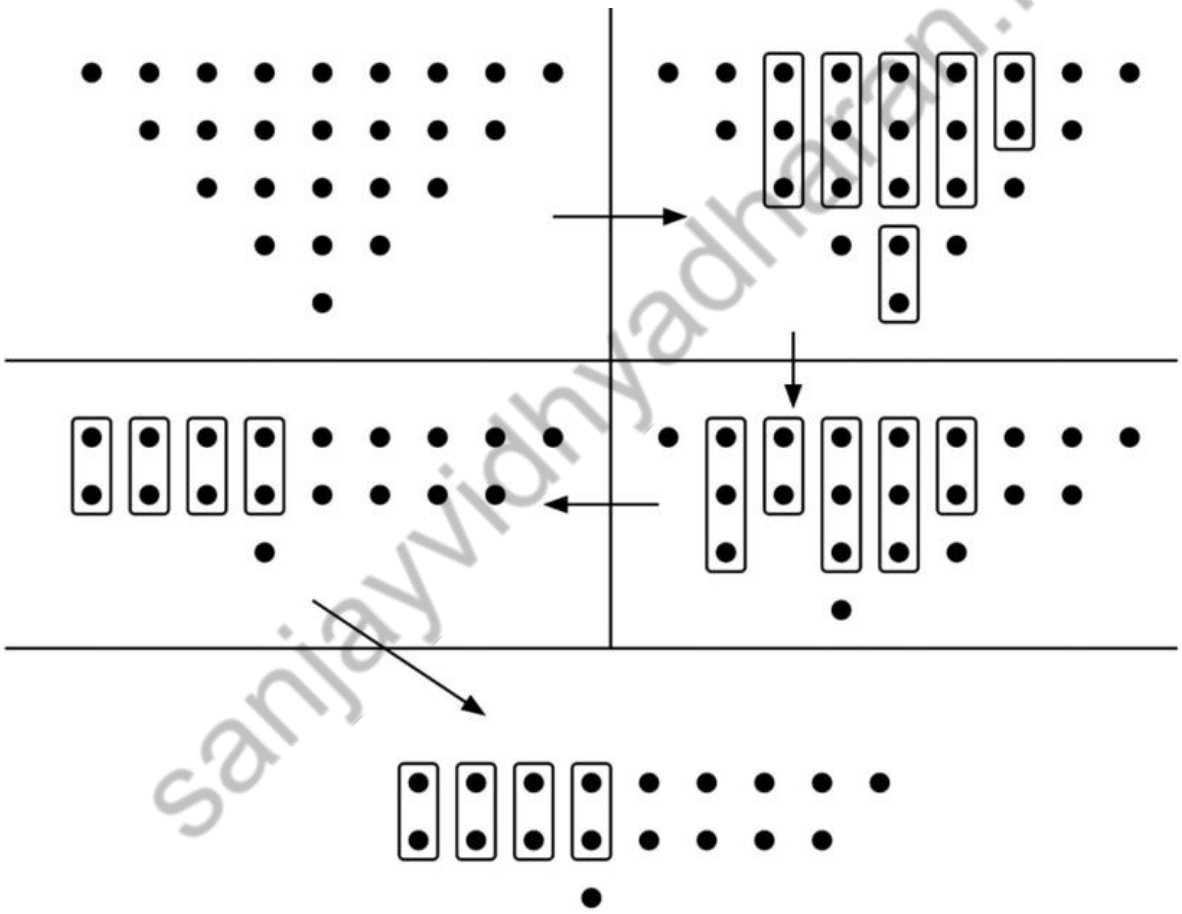
Wallace Tree Multiplication

Wallace tree reduction for 4 X 4 multiplier



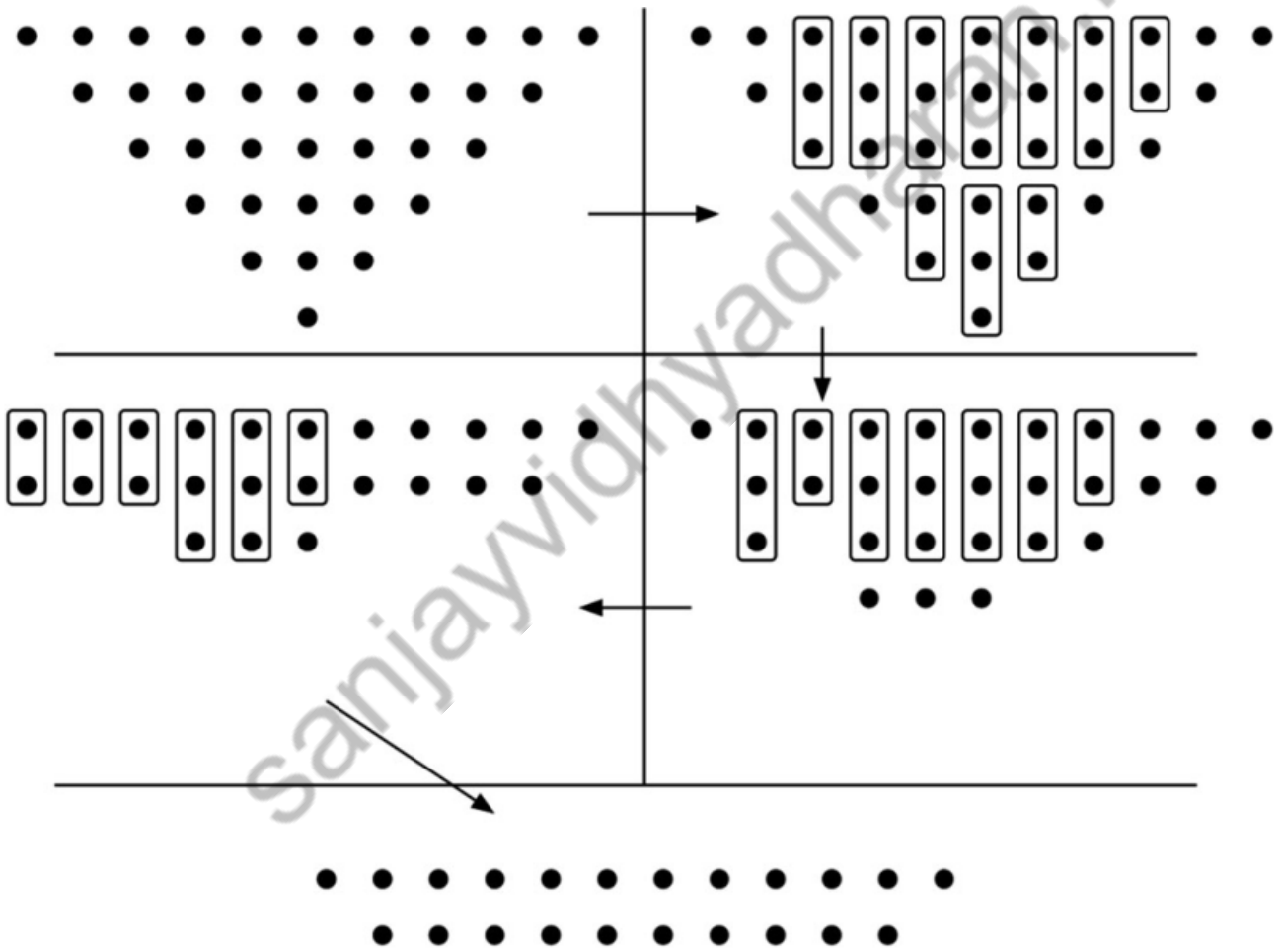
Wallace Tree Multiplication

Wallace tree reduction for 5 X 5 multiplier



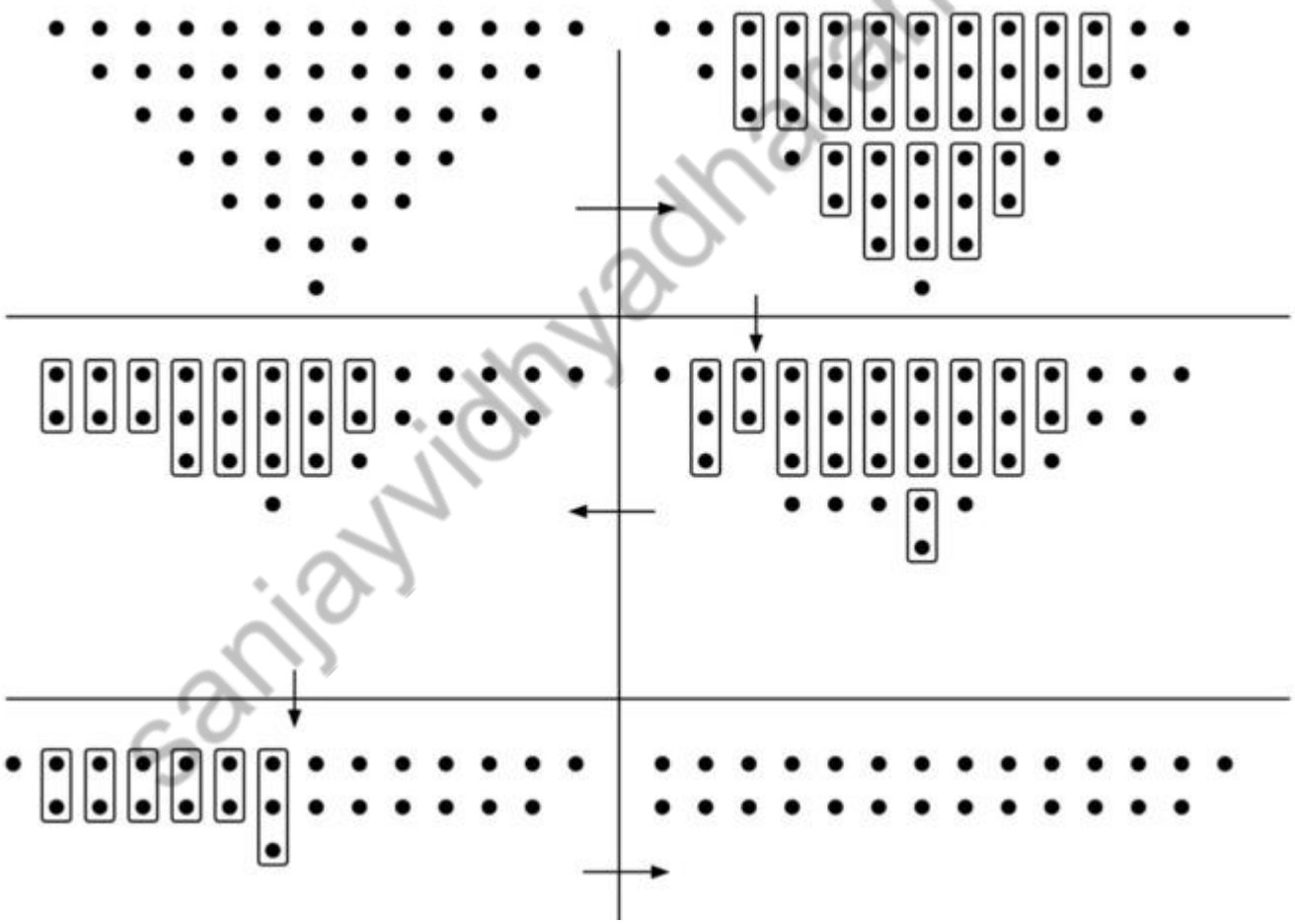
Wallace Tree Multiplication

Wallace tree reduction for 6 X 6 multiplier



Wallace Tree Multiplication

Wallace tree reduction for 7 X 6 multiplier



DADDA Multiplication

For an N X M multiplier, we choose the d which is less than the smaller of N and M, thus for a 3 X 8 multiplier the chosen d would be 2 because 2 < min(3, 8).
For a 4 X 4 multiplier d would be 3.
For an 8 X 8 multiplier d would be 6.

1. If the position has bits less than d, move on

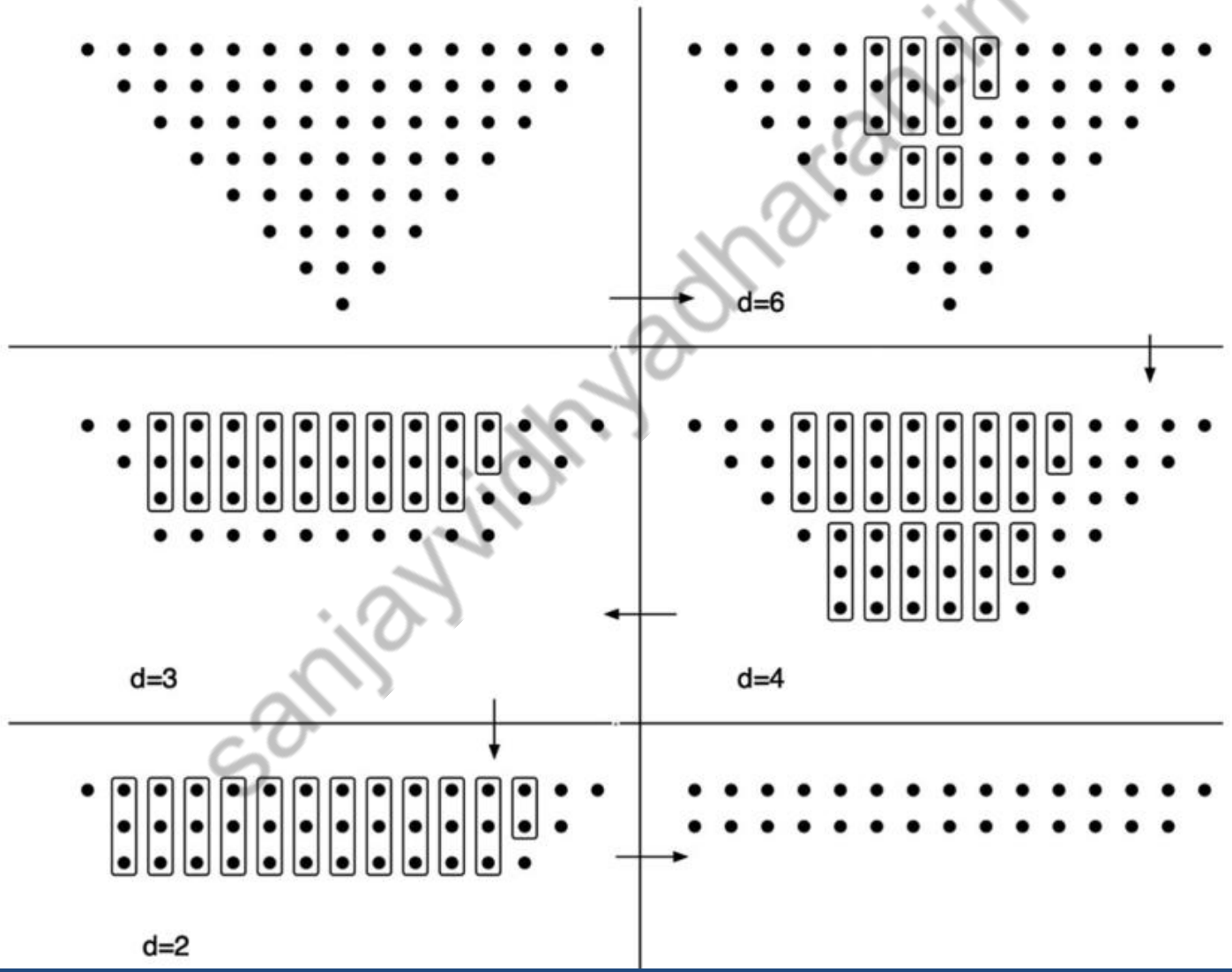
2. If the position has bits equal to d + 1, use an HA to compress two bits, pushing one result to the next position. The current position would be reduced to d, allowing us to move to the next bit position

3. If the position has any higher number of bits, use an FA to compress, pushing bits and repeating again from the first step. In other words, this step would keep us looping on the same position until the height in the position
reduces to d.

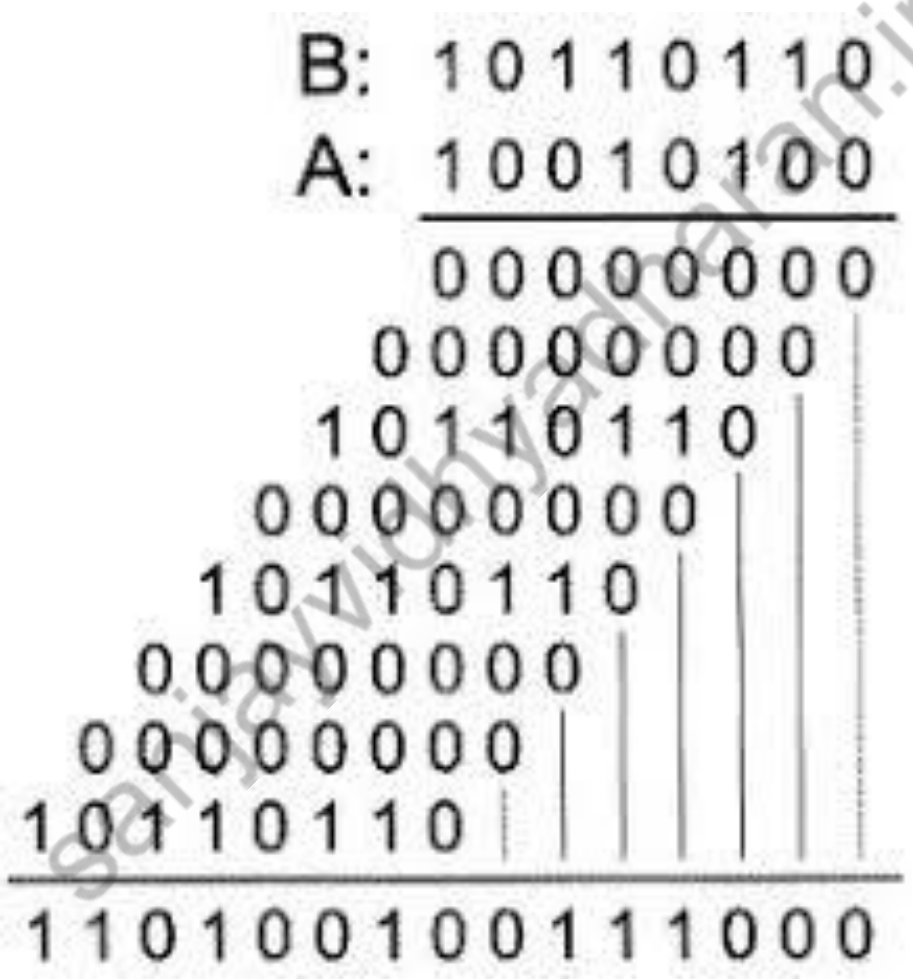
Specifically the sequence is 2, 3, 4, 6, 9, 13, 19, 28, etc.

DADDA Multiplication

8 X 8 DADDA multiplier

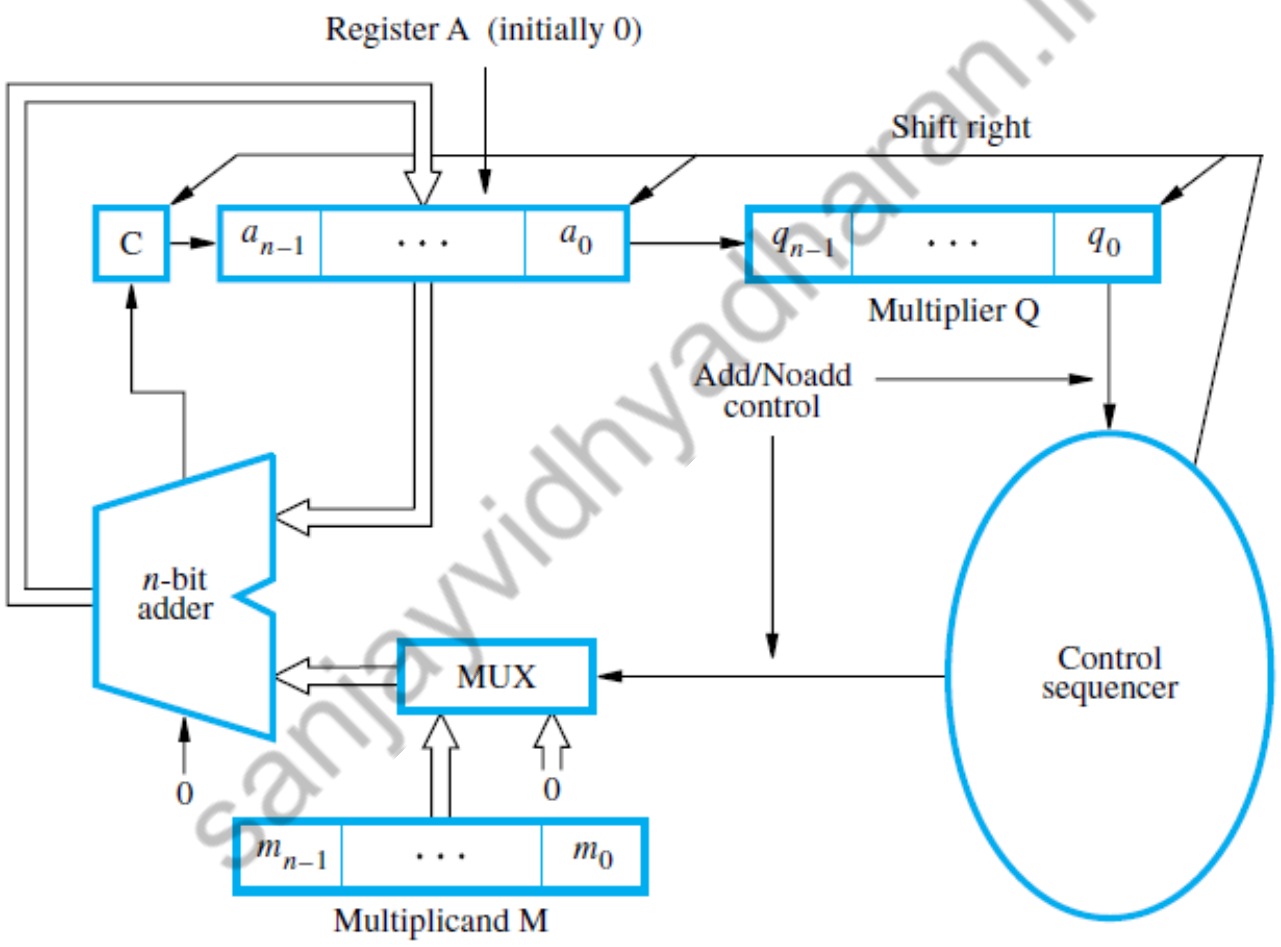


Sequential Multiplier



Sequential Multiplier

Booth Algorithm



(a) Register configuration

Booth Algorithm

Example:

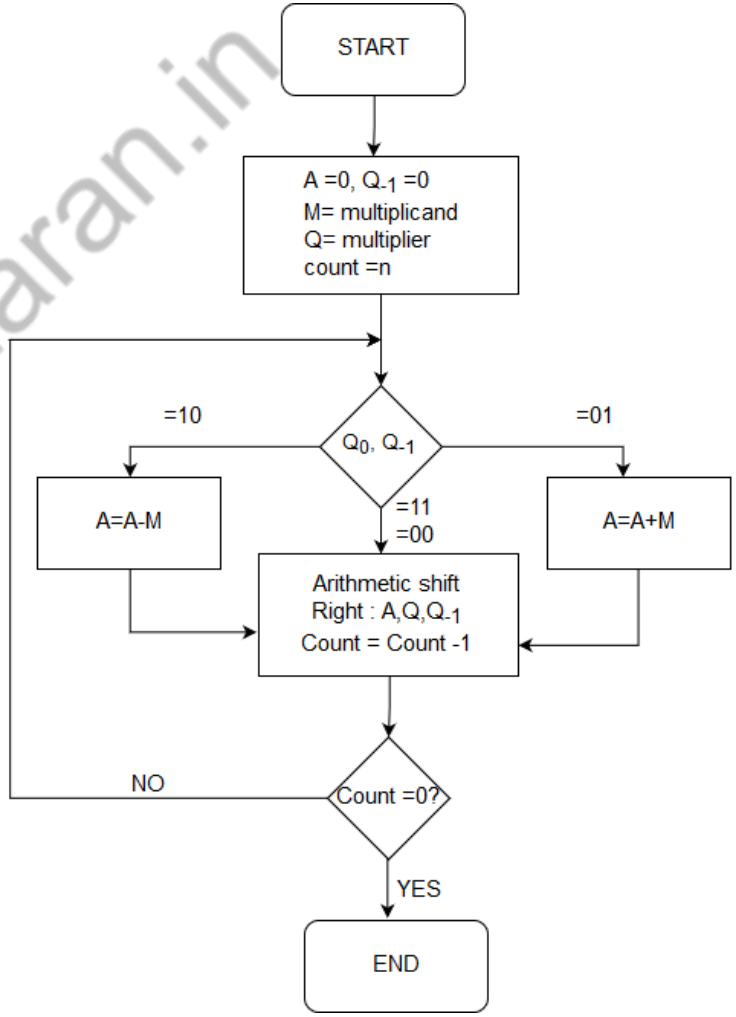
3x5				
	M	A	Q	
Initial Condition	0011	00000	0101	
Clk ↑ (Load)	0011	00011	0101	1
Clk ↓ (Shift Right)	0011	00001	1010	
Clk ↑ (Load)	0011	00001	1010	2
Clk ↓ (Shift Right)	0011	00000	1101	
Clk ↑ (Load)	0011	00011	1101	3
Clk ↓ (Shift Right)	0011	00001	1110	
Clk ↑ (Load)	0011	00001	1110	4
Clk ↓ (Shift Right)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

Booth Algorithm

4x-5

		M	A	Q	Q-1	
	Initial Condition	1011	0000	0100	0	
1	Clk ↑(Load)	1011	0000	0100	0	None
	Clk ↓(Shift Right)	1011	0000	0010	0	
2	Clk ↑(Load)	1011	0000	0010	0	None
	Clk ↓(Shift Right)	1011	0000	0001	0	
3	Clk ↑(Load)	1011	0101	0001	0	A-M
	Clk ↓(Shift Right)	1011	0010	1000	1	
4	Clk ↑(Load)	1011	1101	1000	1	A+M
	Clk ↓(Shift Right)	1011	1110	1100	0	

0101	Unsigned 5	00010100	20
1010	1s Complement of 5	00010011	1s Complement of 20
1011	2s Complement of 5	11101100	2s Complement of 20

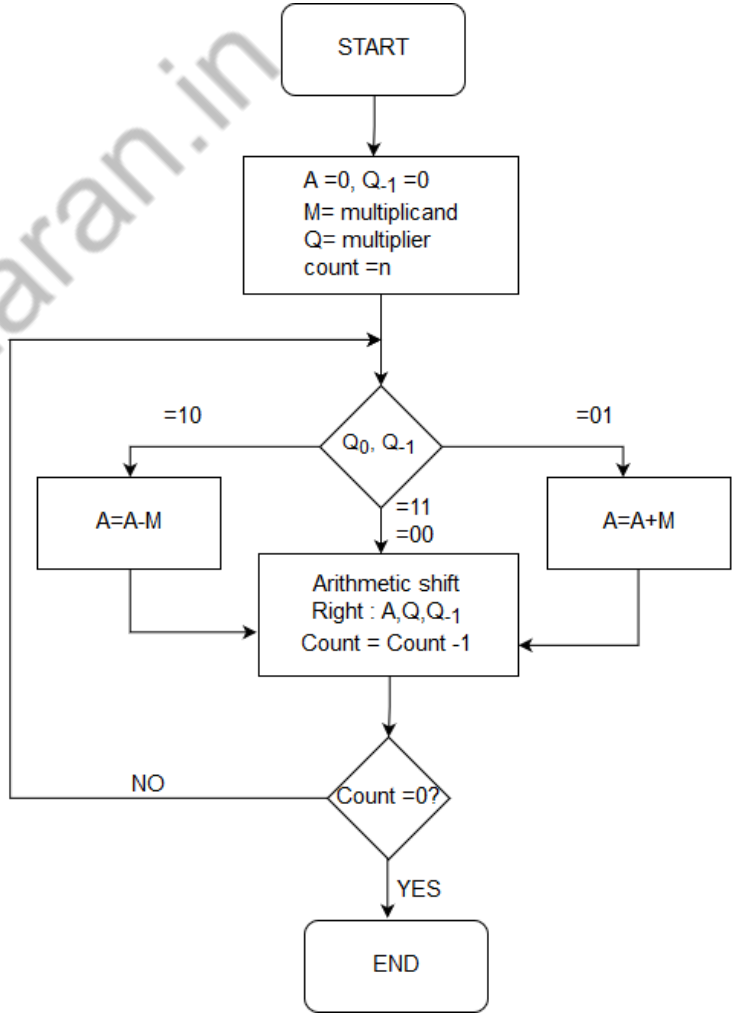


Booth Algorithm

4x-5

		M	A	Q	Q-1	
	Initial Condition	0100	0000	1011	0	
1	Clk ↑(Load)	0100	1100	1011	0	A-M
	Clk ↓(Shift Right)	0100	1110	0101	1	
2	Clk ↑(Load)	0100	1110	0101	1	None
	Clk ↓(Shift Right)	0100	1111	0010	1	
3	Clk ↑(Load)	0100	0011	0010	1	A+M
	Clk ↓(Shift Right)	0100	0001	1001	0	
4	Clk ↑(Load)	0100	1101	1001	0	A-M
	Clk ↓(Shift Right)	0100	1110	1100	0	

0100	4	00010100	20
1011	1s Complement of 4	00010011	1s Complement of 20
1100	2s Complement of 4	11101100	2s Complement of 20



Thank you