Module 6:- Data path Design

Prepared By YP Sir (VESIT)

Module 6:- Data Path Design

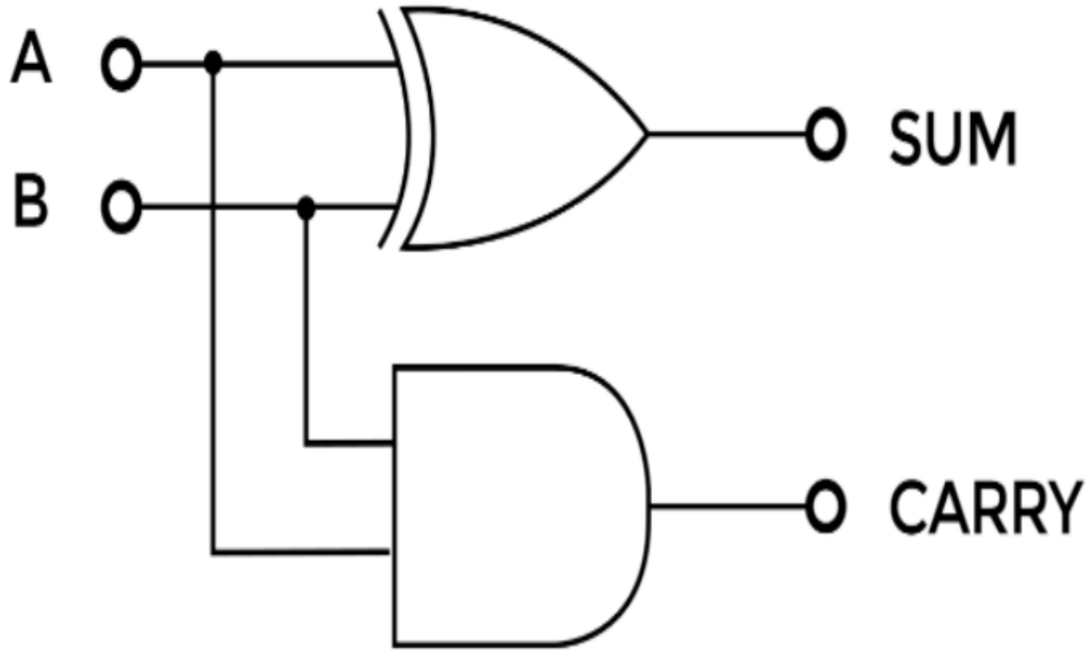
6.1 Adder: CLA adder, MODL, Manchester carry chain High-speed adders: carry skip, carry select and carry save.

6.2 Multipliers and shifter: Array multiplier and barrel shifter.

Half Adder

- A half adder is a digital logic circuit that performs binary addition of two single-bit binary numbers.
- It has two inputs, A and B, and two outputs, SUM and CARRY.

$$\text{Sum} = \bar{A}B + A\bar{B}$$
$$\text{Carry} = AB$$



Logic diagram

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table

→ for CMOS Implementation ft should be in Complement form

$\therefore S = A\bar{B} + \bar{A}B$

$\bar{\bar{S}} = \overline{\overline{A\bar{B} + \bar{A}B}}$

$= \overline{(\overline{A\bar{B}}) \cdot (\overline{\bar{A}B})}$

$= \overline{(\bar{A}+B) \cdot (A+\bar{B})}$

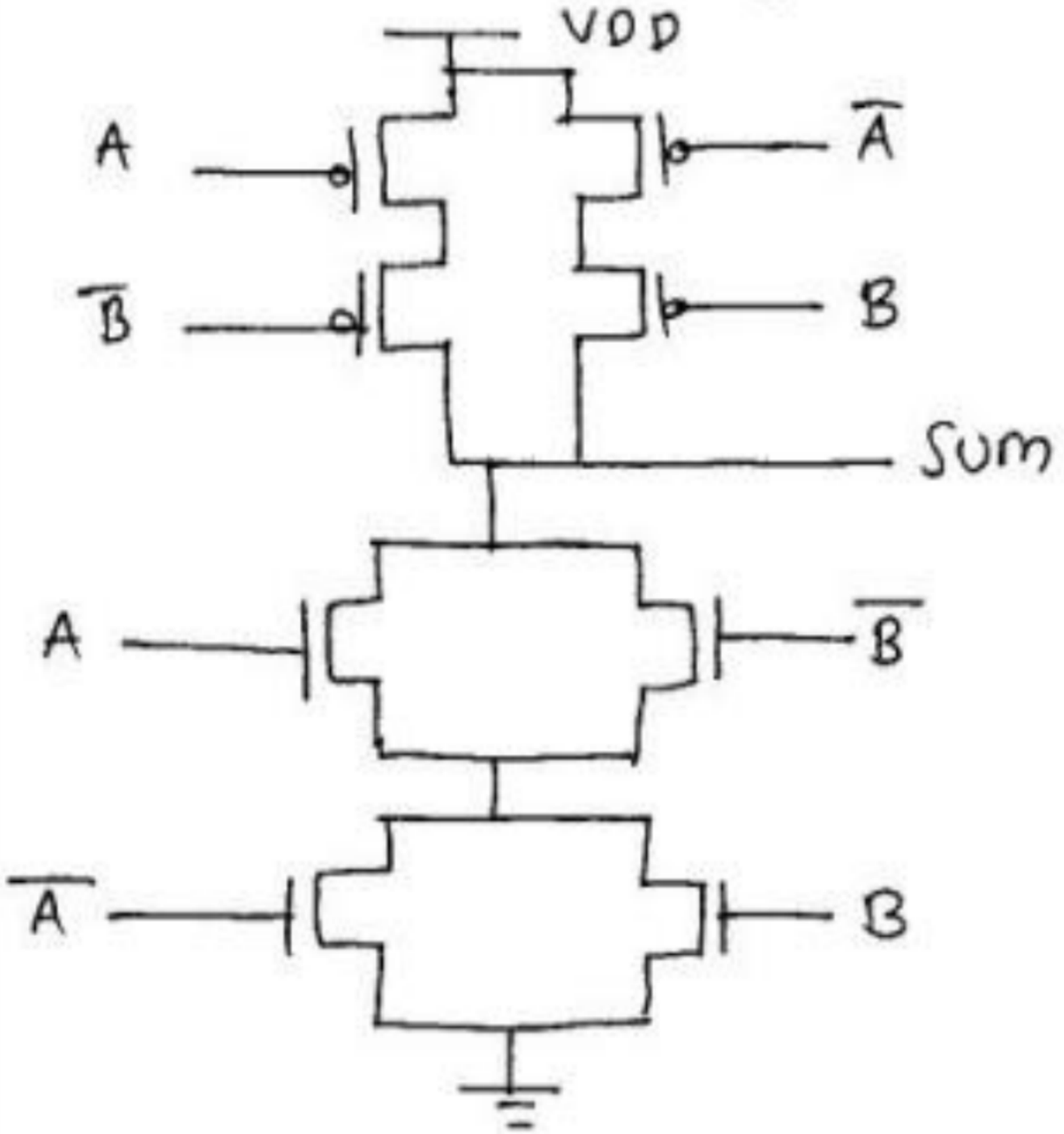
$C = AB$

$\therefore \bar{\bar{C}} = \overline{\overline{AB}}$

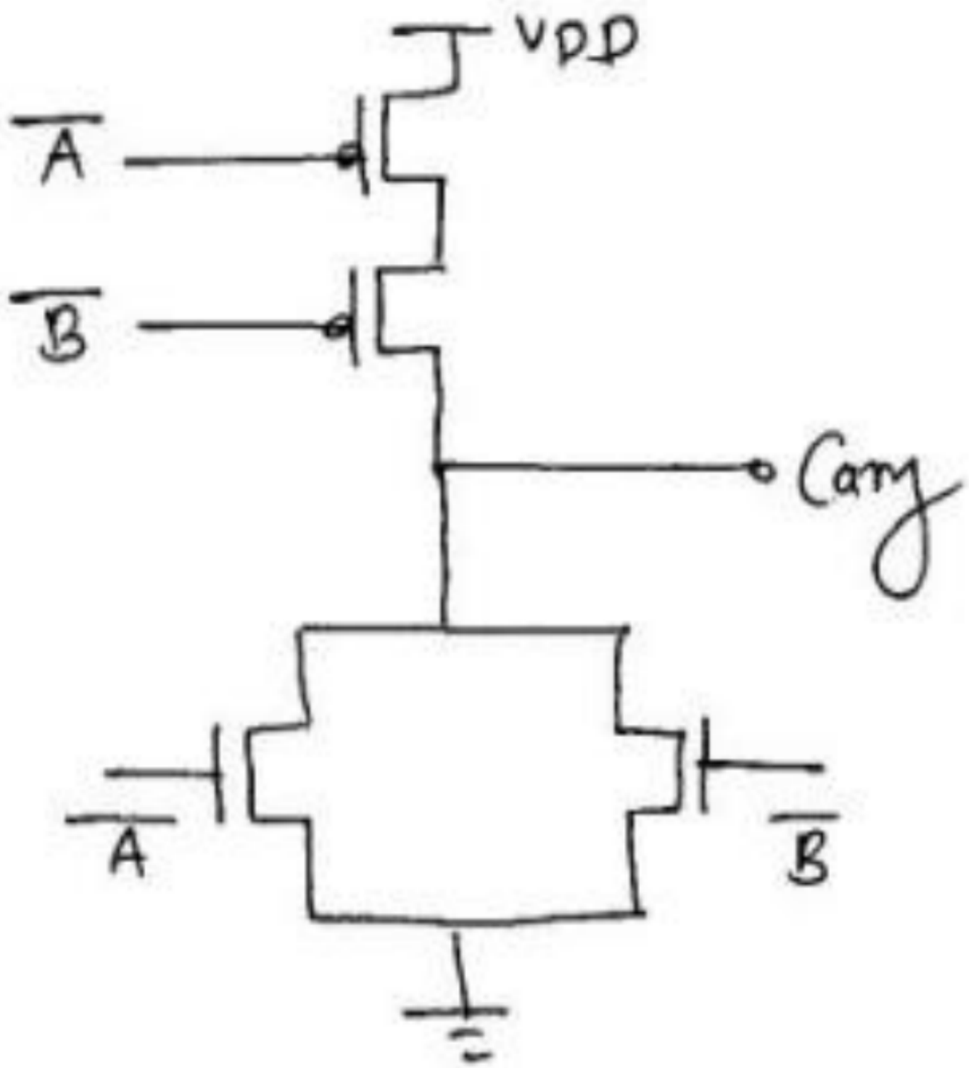
$= (\bar{A}+\bar{B})$

Static CMOS Implementation of Half adder

for Sum $= \overline{(A+\bar{B})\cdot(\bar{A}+B)}$

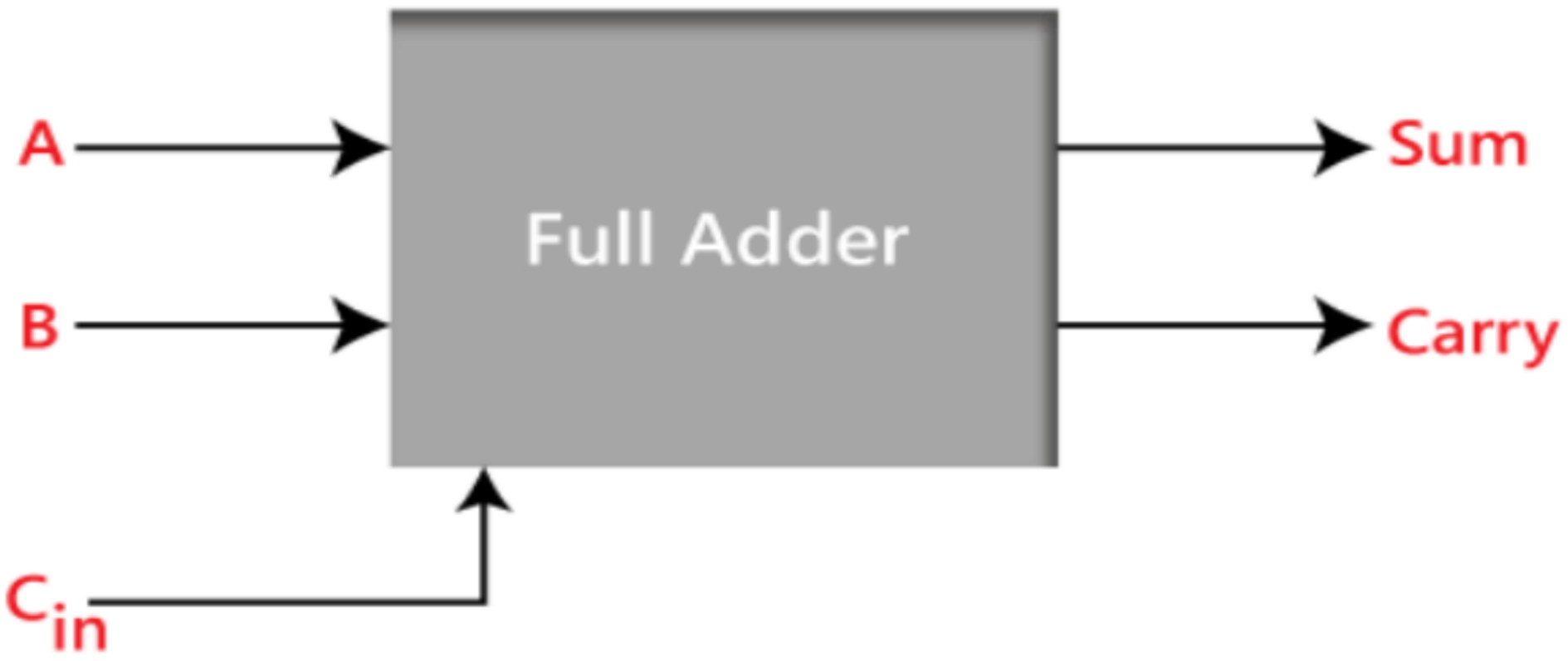


for Carry $= \overline{\bar{A}+\bar{B}}$



Full Adder

- A full adder is a digital logic circuit that performs binary addition of three inputs and produces two outputs.
- It has three inputs as A,B & Cin and two outputs, SUM and CARRY

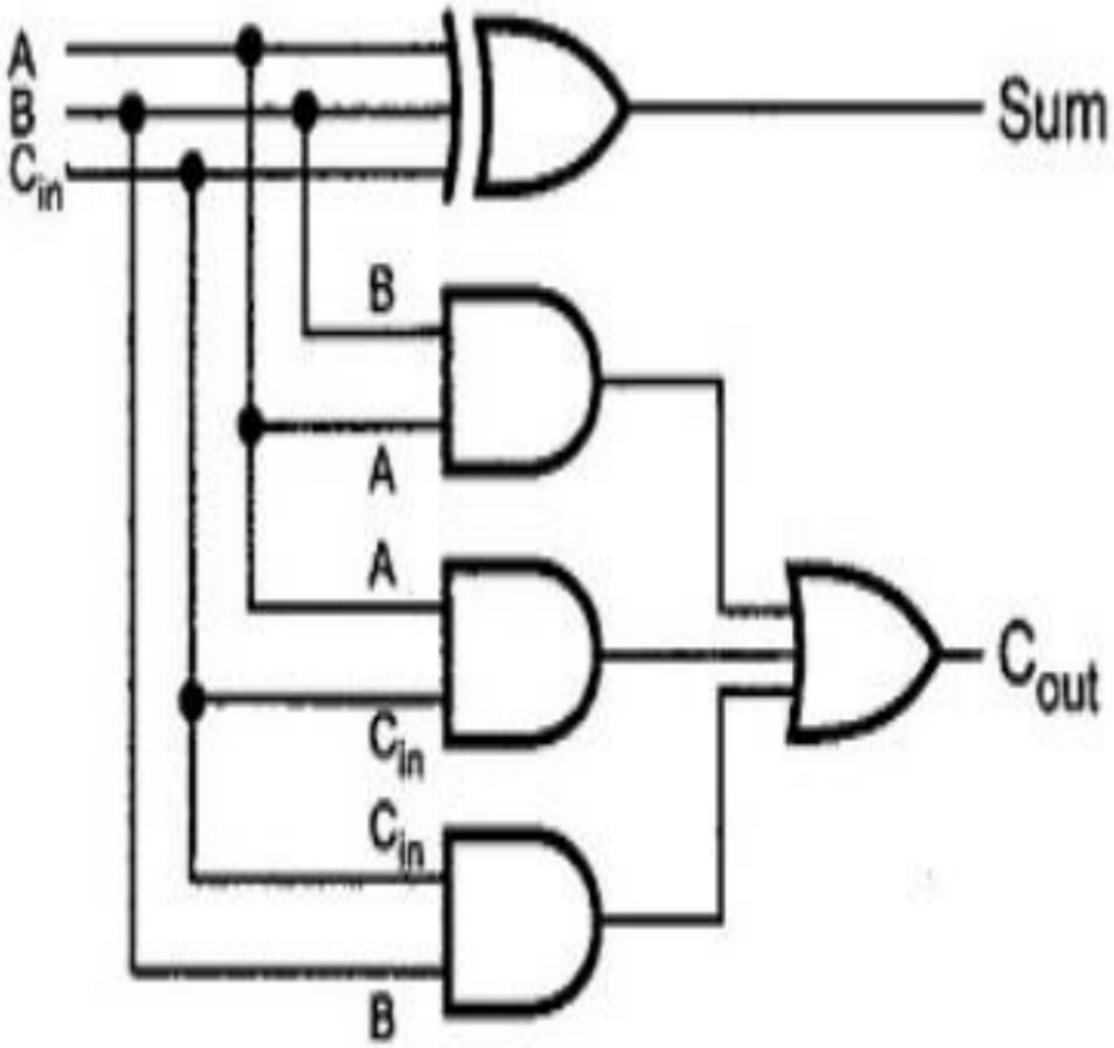


Block diagram

$$\text{Sum} = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + A\,C_{in} + BC_{in}$$

Logic expression



Logic diagram

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth table

→ CMOS Implementation of <u>full-Adder ckt</u>.

$$S = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C_{in}} + A\overline{B}\,\overline{C_{in}} + ABC_{in}$$

Take Double Inversion on b.S.

$$\overline{\overline{S}} = \overline{\overline{(\overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C_{in}} + A\overline{B}\,\overline{C_{in}} + ABC_{in})}}$$

$$= \overline{(\overline{\overline{A}\,\overline{B}C_{in}}) \cdot (\overline{\overline{A}\,B\overline{C_{in}}}) \cdot (\overline{A\overline{B}\,\overline{C_{in}}}) \cdot (\overline{ABC_{in}})}$$

$$S = \overline{(A+B+\overline{C_{in}}) \cdot (A+\overline{B}+C_{in})(\overline{A}+B+\overline{C_{in}}) \cdot (\overline{A}+\overline{B}+\overline{C_{in}})}$$

<u>Smillarly</u>,

$$C = AB + BC_{in} + AC_{in}$$

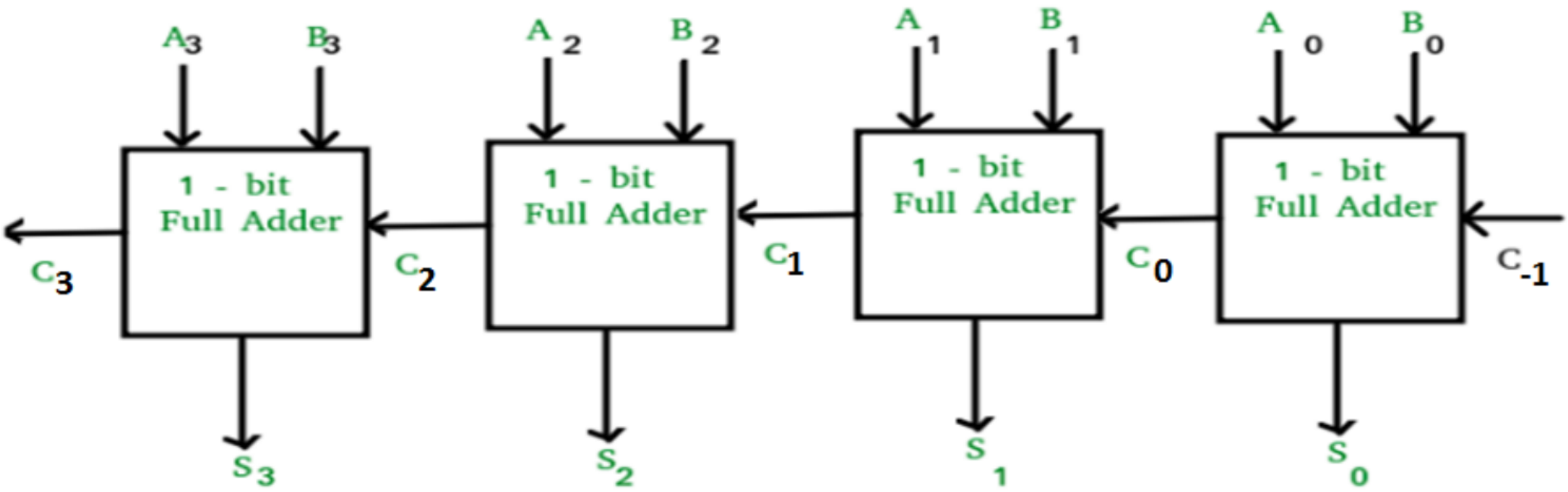
$$\overline{\overline{C}} = \overline{\overline{AB + BC_{in} + AC_{in}}}$$

$$= \overline{(\overline{AB}) \cdot (\overline{BC_{in}}) \cdot (\overline{AC_{in}})}$$

$$C = \overline{(\overline{A}+\overline{B}) \cdot (\overline{B}+\overline{C_{in}}) \cdot (\overline{A}+\overline{C_{in}})}$$

Ripple Carry Adder

- An n-bit binary adder can be constructed by connecting n-full adders in series such that carry o/p of the first full adder is connected to carry i/p of the next full adder.
- The carry i/p has ripple through all stages before the final sum & carry is produced. **Hence called as Ripple carry adder.**
- The delay of the circuit depends on the propagation delay of each gate & is more if carry is generated by each stage of full adder circuit.
- Following fig shows block diagram of 4 bit Ripple carry adder circuit,

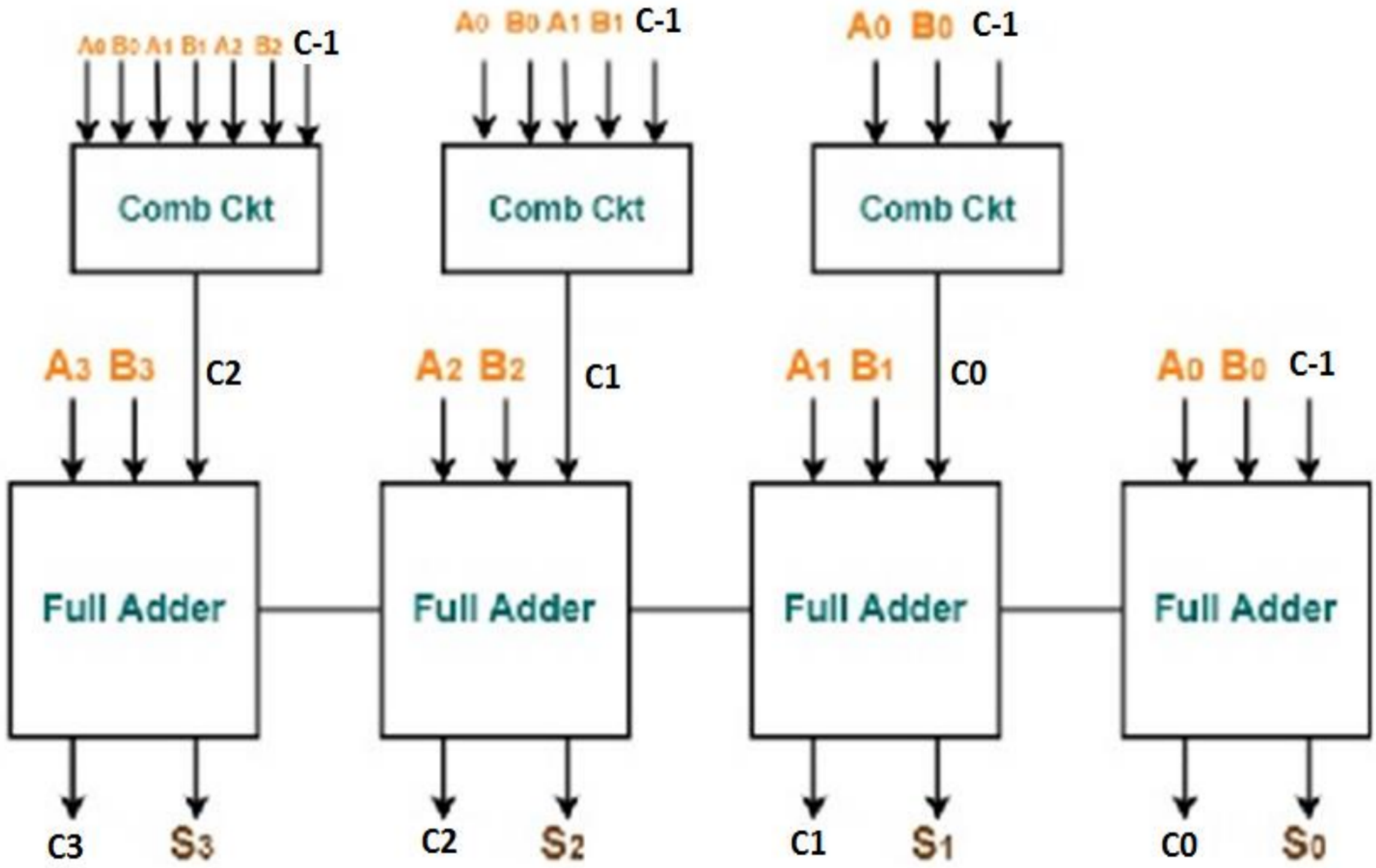


- As shown in fig, 4 full adders are cascaded. Thus to produce final sum and carry o/p , we have to consider the carry generated by individual full adder circuit & propagate to the last stage.
- Thus, propagation delay of the circuit is very large.
- Hence, to overcome this problem and speed up the speed of operation Carry Look Ahead (CLA) generator circuit is used

Carry look Ahead Adder

- In this adder, the carry input at any stage of the adder is independent of the carry bits generated at the independent stages.
- Here the output of any stage is dependent only on the bits which are added in the previous stages and the carry input provided at the beginning stage.
- Hence, the circuit at any stage does not have to wait for the generation of carry-bit from the previous stage and carry bit can be evaluated at any instant of time.

4 bit CLA


A0 B0 A1 B1 A2 B2 C-1
Comb Ckt
A3 B3
C2
A2 B2
C1
A1 B1
C0
A0 B0
C-1
Full Adder
Full Adder
Full Adder
Full Adder
C3
S3
C2
S2
C1
S1
C0
S0

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Carry Generate (Gi) = A.B

Carry Propgate (Pi) = [A⊕B].Ci

Output carry is given by,

Ci = Gi +Pi Ci-1

Therefore, the carry bits C0, C1, C2, and C3 can be calculated as

$$C_i = G_i + P_i C_{i-1} \text{ ---------- (A)}$$

Keep i=0 in equation (A), we get,

$$C0 = G0 + P0\ C-1 \text{ ---------- (1)}$$

Keep i=1 in equation (A), we get,

$$C1 = G1 + P1\ C0$$

Put value of C0 from equation (1) in above equation

We get,

$$C1 = G1 + P1\ (G0 + P0\ C-1)$$

$$C1 = G1 + P1G0 + P1P0C-1 \text{ ---------- (2)}$$

Keep i=2 in equation (A), we get,

$C2 = G2 + P2\ C1$

Put value of C1 from equation (2) in above equation

We get,

$C2 = G2 + P2\ (G1 + P1G0 + P1P0\ C\text{-}1)$

$C2 = G2 + P2G1 + P2P1G0 + P2P1P0C\text{-}1$ ---------- (3)

Keep i=3 in equation (A), we get,

$C3 = G3 + P3\ C2$

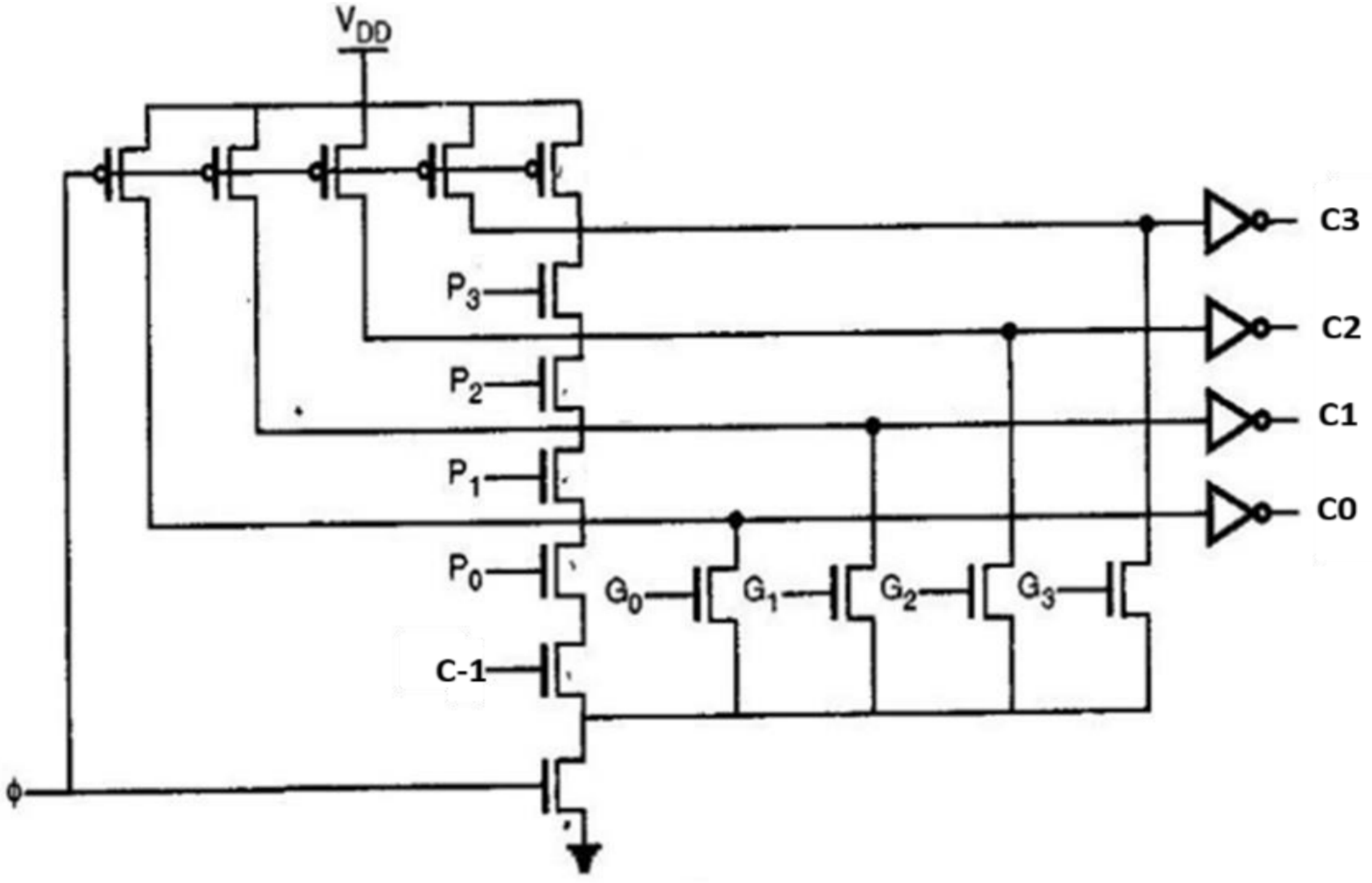
Put value of C2 from equation (3) in above equation

We get,

$C3 = G3 + P3\ (G2 + P2G1 + P2P1G0 + P2P1P0C\text{-}1\ )$

$C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C\text{-}1$ ----(4)

Multiple O/P Domino Logic (MODL) CLA CMOS schematic



Manchester Carry Adder

- CLA carry generator circuit requires more hardware. Hence to reduce the hardware Manchester carry generator is being used.
- It is an alternate structure for carry evaluation using CLA bits.
- It is based on building a switch-logic network for the basic equation,
- $C_{i+1} = G_i + P_i C_i$
- It define carry in terms of control signal (Gi,Pi,Ki) such that only one control signal is active at a time.

Truth table for Full adder

Ai	Bi	Ci	Ci+1	Pi	Gi	Ki
0	0	0	0	0	0	1
0	0	1	0	0	0	1
0	1	0	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	1	1	1	0	0
1	1	0	1	0	1	0
1	1	1	1	0	1	0

Where , Generate Gi = Ai.Bi

Propagate Pi = $\overline{Ai \oplus Bi}$

Carry Kill (Ki) = $\overline{Ai + Bi}$ = $\overline{Ai} \cdot \overline{Bi}$

Ci+1 is always Zero when Ki=1 , hence called as Carry Kill

- $C_{i+1} = G_i + P_i C_i$

Manchester Carry Generation Concept (Switch logic)

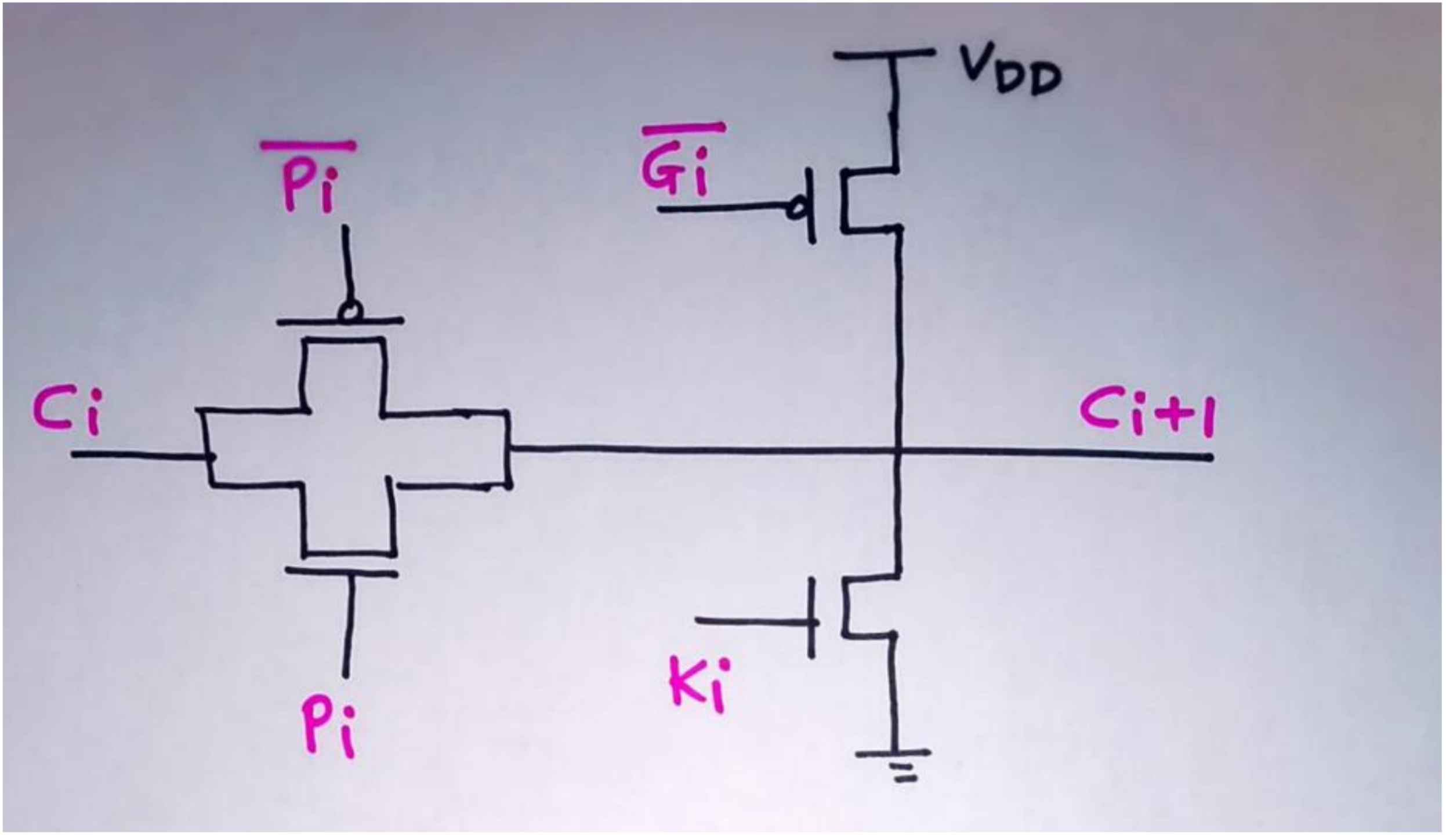
Ai	Bi	Ci	Ci+1	Pi	Gi	Ki
0	0	0	0	0	0	1
0	0	1	0	0	0	1
0	1	0	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	1	1	1	0	0
1	1	0	1	0	1	0
1	1	1	1	0	1	0

If Ai = Bi = 1 then Gi=1 and Ci+1 = 1

If Ai ǂ Bi then Pi=1 and Ci+1 = Ci

If Ai = Bi = 0 then Ki=1 and Ci+1 = 0

Manchester carry generation using Static CMOS



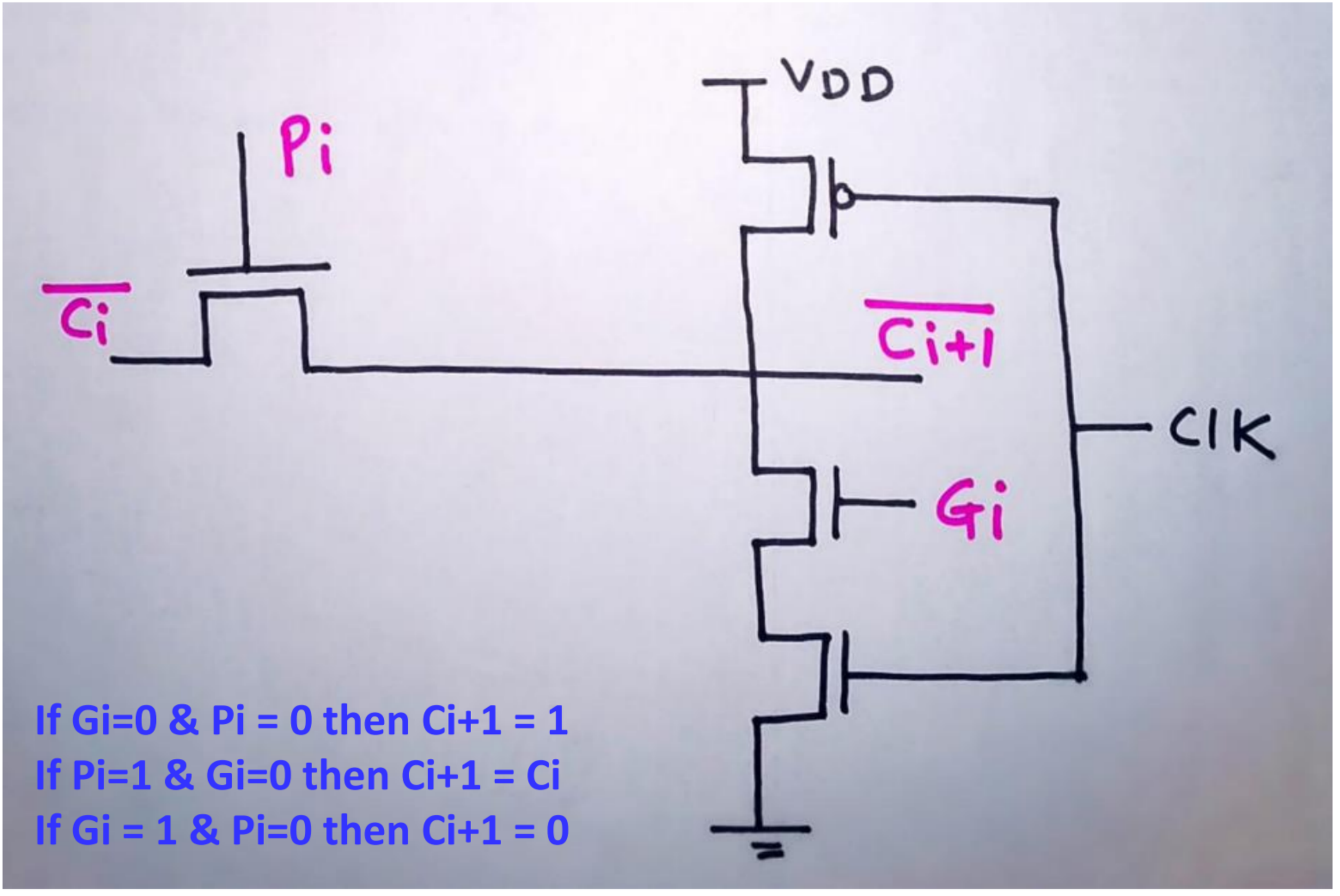
Manchester carry generation using Dynamic CMOS

Ai	Bi	Ci	$\overline{Ci}$	Ci+1	$\overline{Ci+1}$	Pi	Gi
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0
0	1	0	1	0	1	1	0
0	1	1	0	1	0	1	0
1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	0	1

If Gi=0 & Pi = 0 then $\overline{Ci+1} = 1$

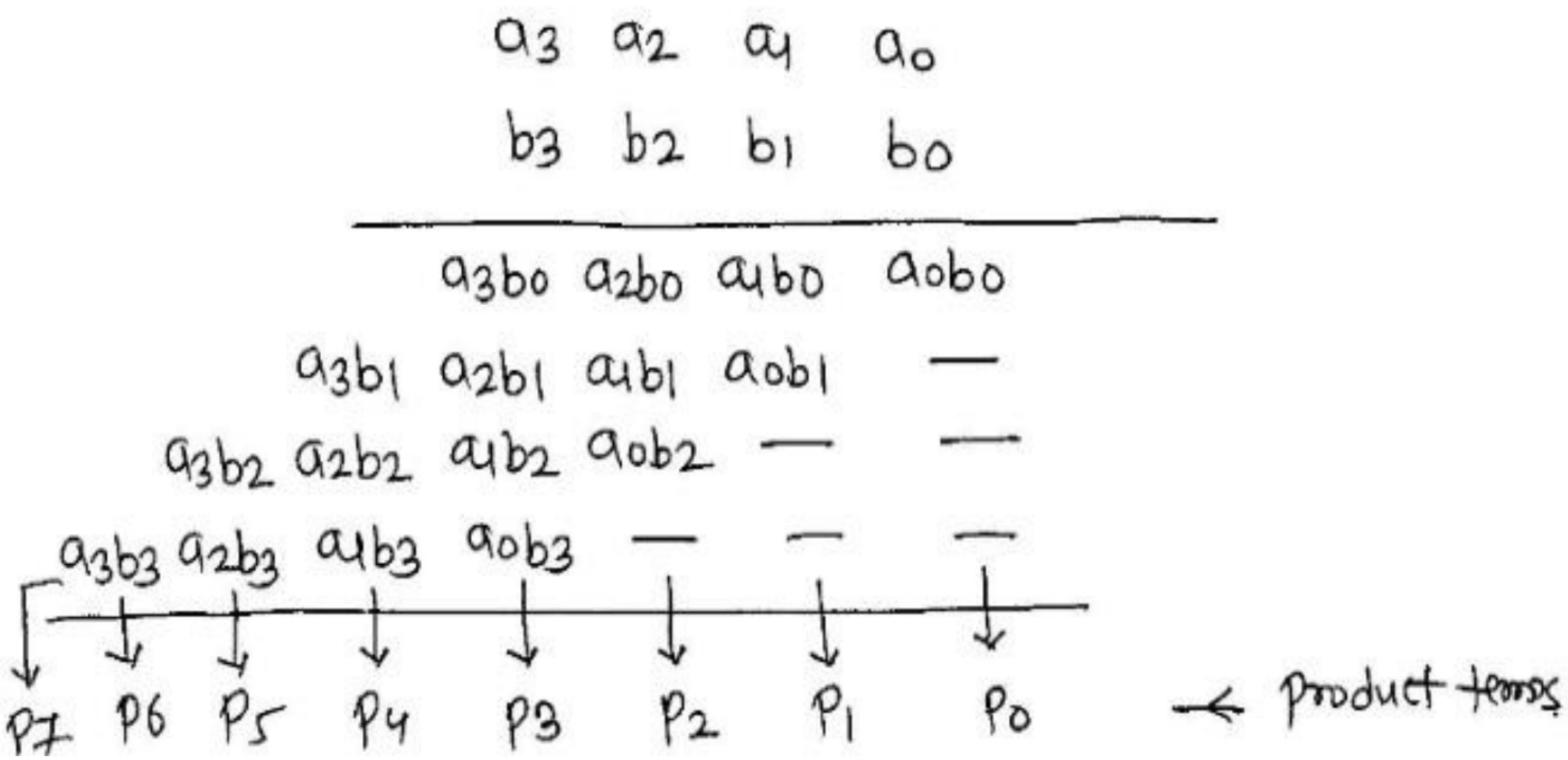
If Pi=1 & Gi=0 then $\overline{Ci+1} = \overline{Ci}$

If Gi = 1 & Pi=0 then $\overline{Ci+1} = 0$


VDD
Pi
Ci
Ci+1
CIK
Gi
If Gi=0 & Pi = 0 then Ci+1 = 1
If Pi=1 & Gi=0 then Ci+1 = Ci
If Gi = 1 & Pi=0 then Ci+1 = 0

Array Multiplier

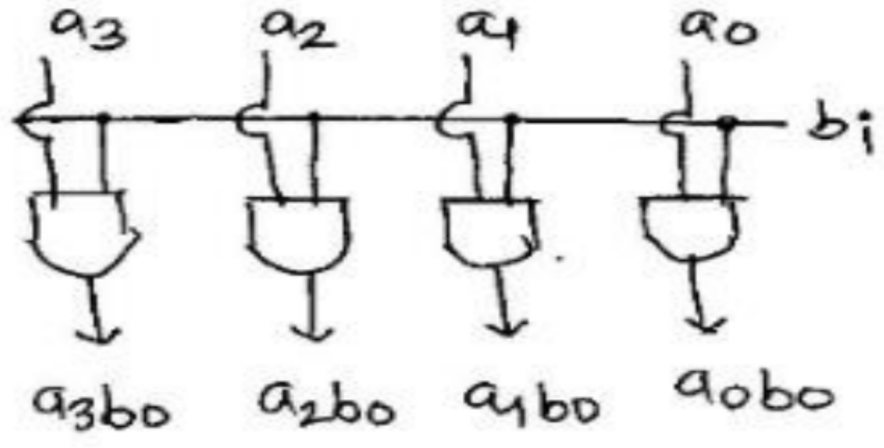
- Multiplication is basic operation in most signal processing algorithm.
- The multiplication process is shown below,



- To map the concept of multiplier , the hardware must perform the following three functions,
- Partial product generation
- Partial product accumulation
- Final addition

Partial Product Generation

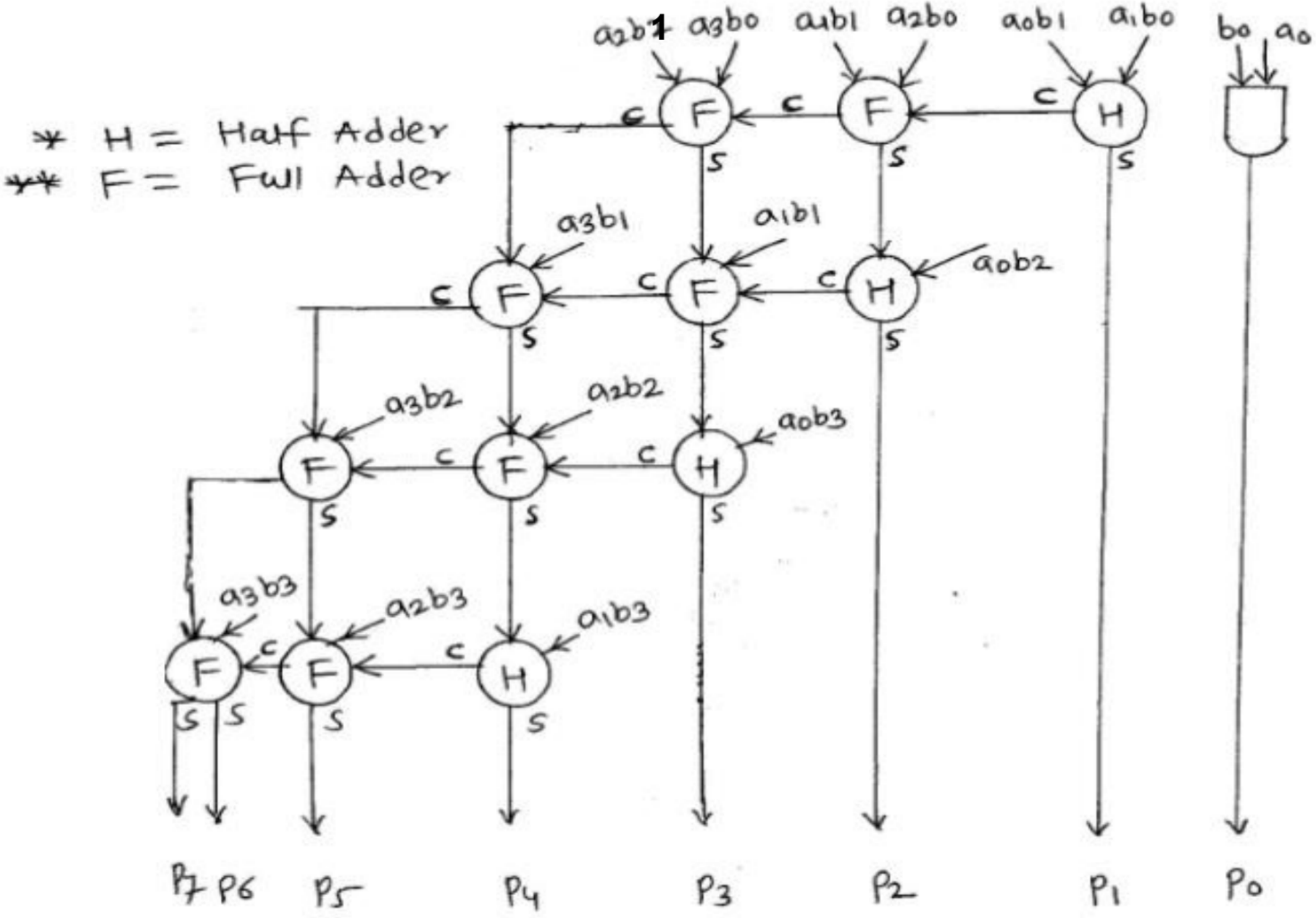
- This partial products are generated for each multiplier bit when multiplied with the multiplicand.
- Thus, it implies that the row of partial product array is eighter row of all zeros if the multiplier bit is zero or is same as that of multiplicand if the multiplier bit is one.



Partial Product Accumulation

- The partial product generated during the multiplication has to be added to get the final result.
- Accumulation is basically Multi operand addition.
- ie. Accumulation of partial products are achieved by using array to address.

Structure of 4 X 4 Array Multiplier



Barrel Shifter

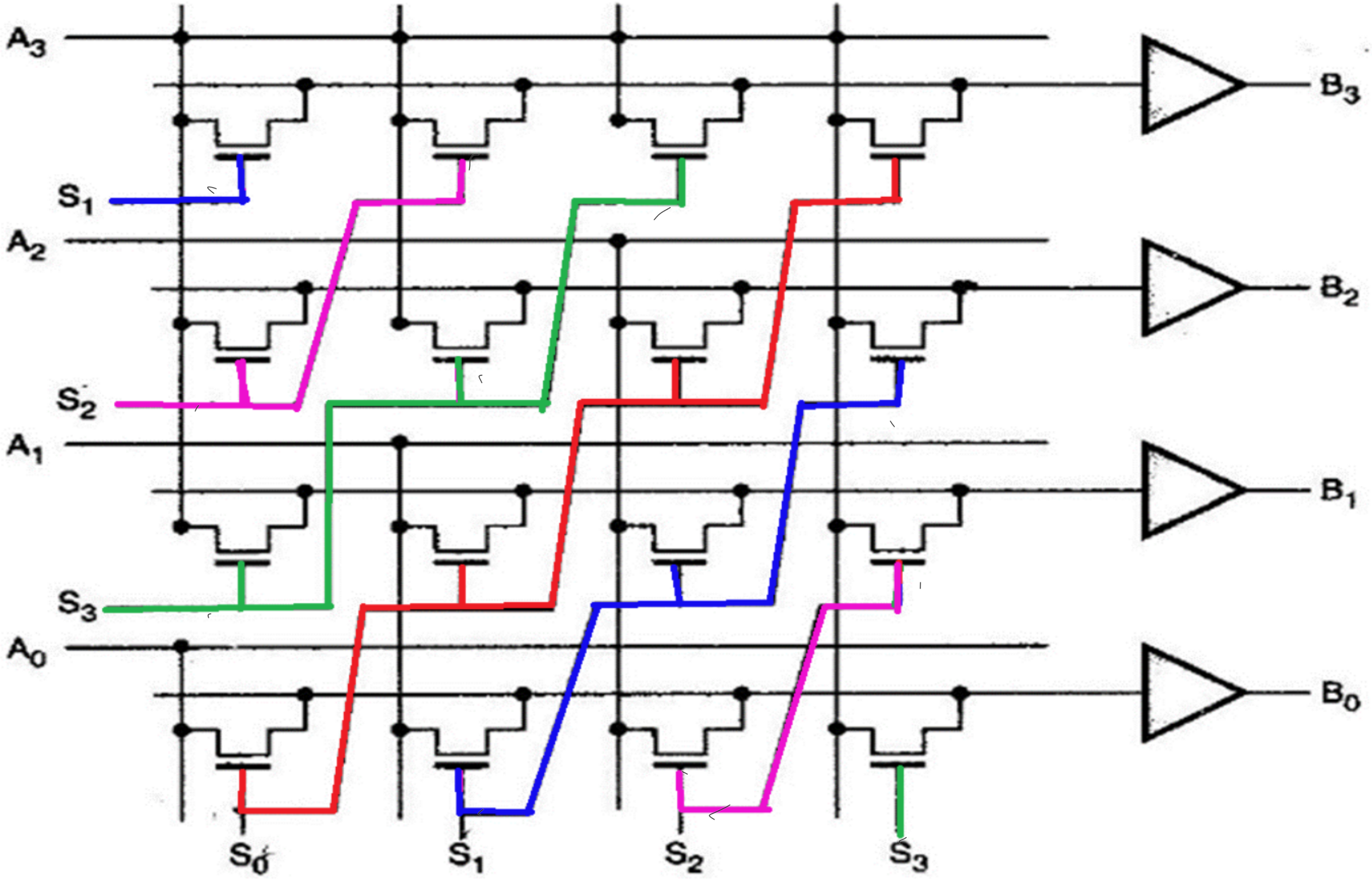
- The disadvantage of latch based shift register is that it can shift only one bit per clock cycle. Thus, if we want to shift 8 bits of data then operation takes 8 clock pulses.
- Thus, the performance of this shift register can be improve with the use of Barrel shifter.

- **Barrel shifter is a combinational logic circuit which perform n-bit shifts in single clock cycle depending on control signal.**

n-bit shifts → single clock cycle depending on control signal

4 X 4 Barrel Shifter using Pass transistor

- The number of Rows equals the word length of the data and number of Columns equals the maximum shift length.



- **Following table shows the shifting operation to the right side depending on the control signal.**

S0 = 1

Control	B3	B2	B1	B0	Operation
S0 = 1	A3	A2	A1	A0	No shift
S1 = 1	A3	A3	A2	A1	Shift by 1 bit
S2 = 1	A3	A3	A3	A2	Shift by 2 bits
S3 = 1	A3	A3	A3	A3	Shift by 3 bits

Note :- In answer you can explain how the shifting takes place in barrel shifter by giving reference of above table

High-speed adders

- **High-speed adder is a type of adder circuit that is optimized to perform addition operations faster than conventional adders *(like ripple-carry adders).***

High-Speed Adders

skip select save

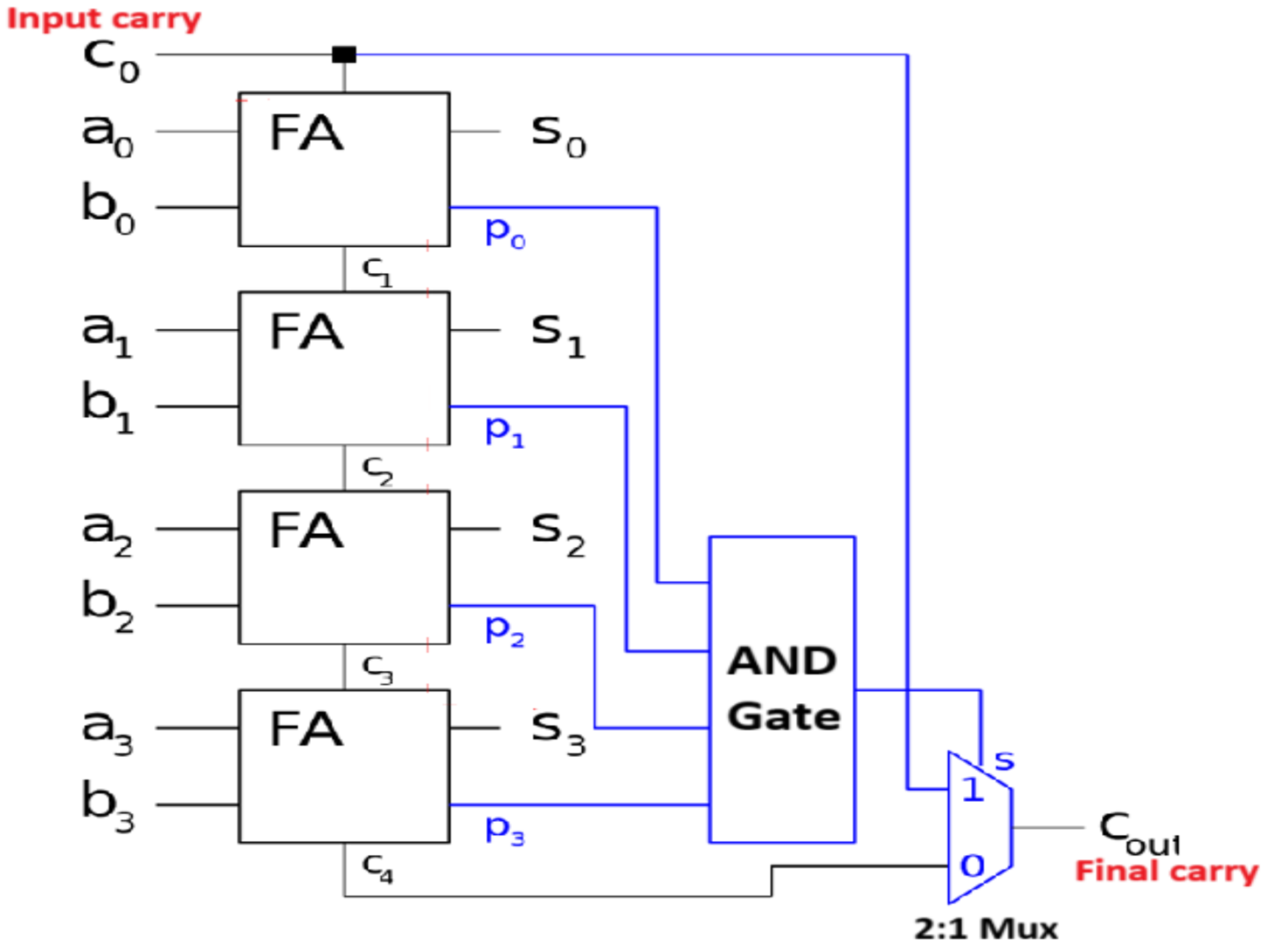
Carry Skip adder	Carry Select adder	Carry Save adder
Divides the adder into blocks and uses skip logic to bypass carry propagation across groups.	**Pre-computes Sum & carry both for each block with Cin = 0 as well as Cin=1.**	**Used for multioperands *(more than 2)* addition. Instead of immediately propagating carries, it stores them and resolves later.**

Carry Skip adder / Carry Bypass adder

- It divides the adder into blocks and uses skip logic to bypass carry propagation across groups.

- Sum & Carry operations are performed separately.

- For each input pair A(i) , B(i) , the propagate condition P(i) = A(i) XOR B(i) are determined & when all propagate conditions are true ie. [P(0) P(1) P(2) P(3)= 1] then the input carry (C0) determines the final carry (Cout).


Input carry
c_0
a_0
b_0
FA
s_0
p_0
c_1
a_1
b_1
FA
s_1
p_1
c_2
a_2
b_2
FA
s_2
p_2
c_3
a_3
b_3
FA
s_3
p_3
c_4
AND Gate
S
1
0
C_{out}
Final carry
2:1 Mux

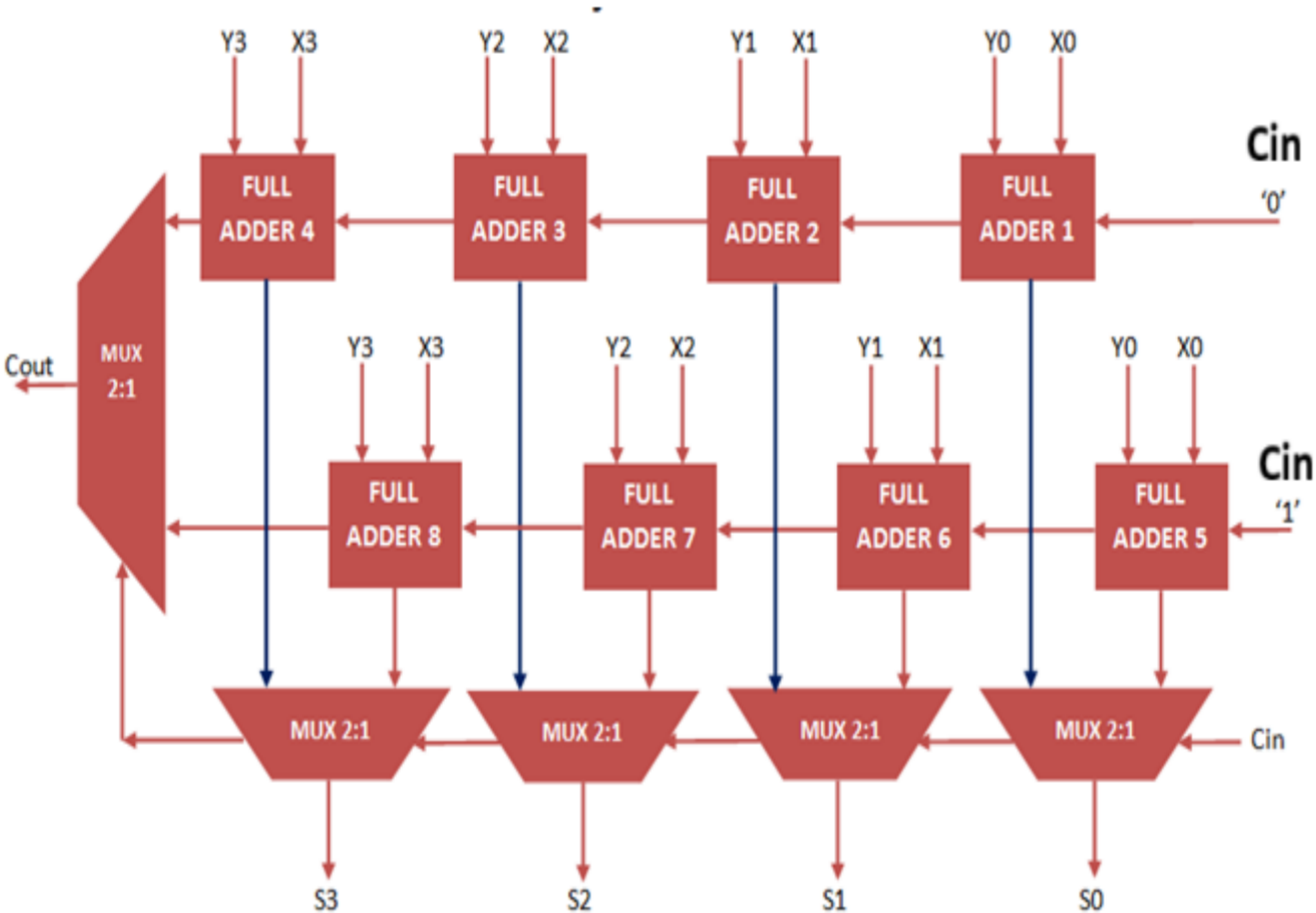
Operation: 1010 & 0101 1101 & 0101

Note :- You can explain the working of circuit by giving any one of above example

Carry Select adder

- It pre-computes the sum & carry for each block with Cin=0 as well as Cin = 1.
- It consist of two adders and Mux circuits.
- After the two results are calculated, the correct sum as well as the correct carry out is then selected with the Mux once the correct carry in is known.

Operation: 0111 & 1000



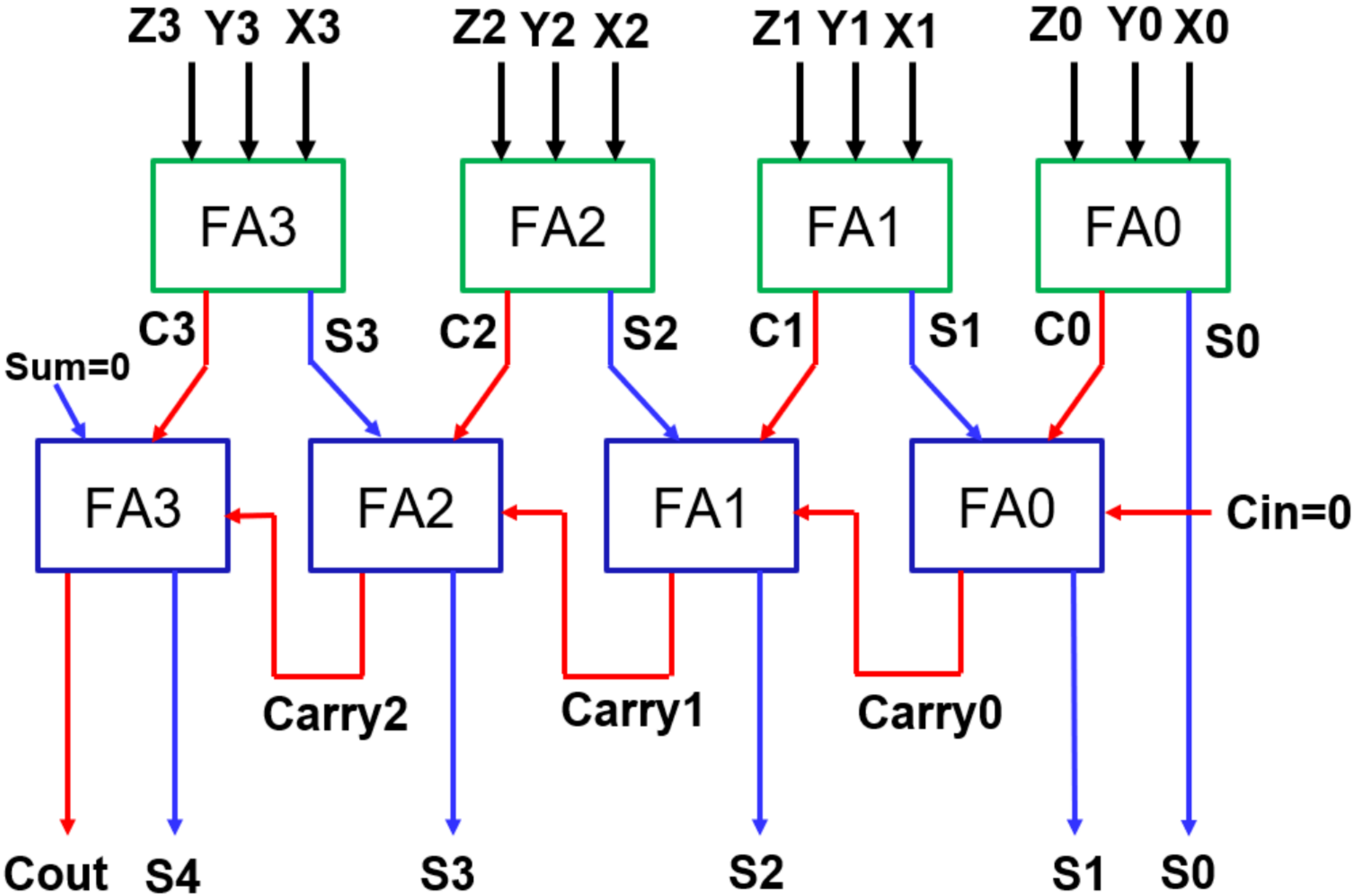
Carry Save adder

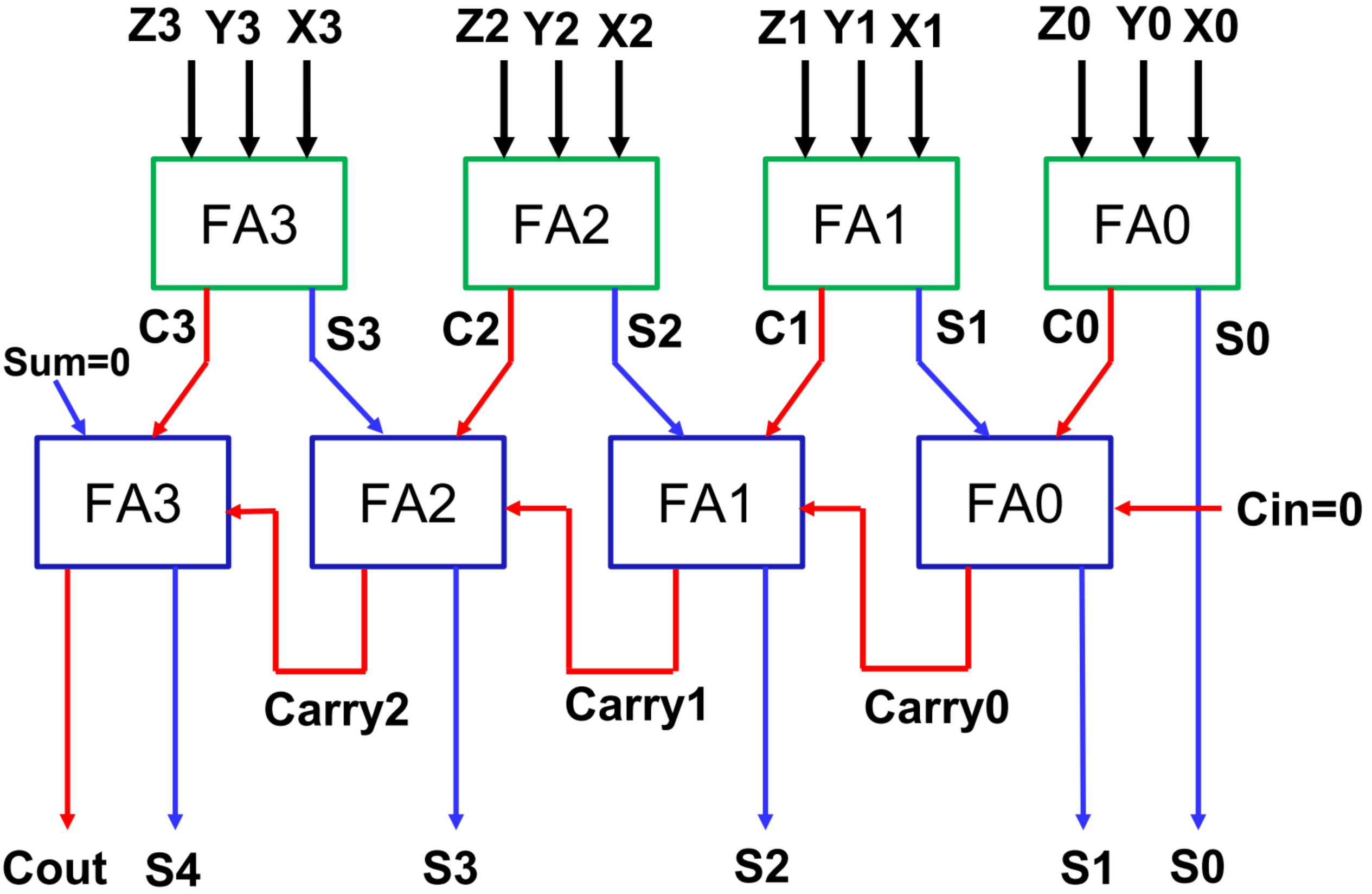
- It is used for multioperands *(more than 2 bit)* addition.
- Instead of immediately propagating carries, it stores them and resolves later.

Operation: 15 + 10 + 12

(15) = 1111
(10) = 1010
(12) = 1100

Note :- You can explain the working of circuit by giving above example


Z3 Y3 X3
Z2 Y2 X2
Z1 Y1 X1
Z0 Y0 X0
FA3
FA2
FA1
FA0
C3
S3
C2
S2
C1
S1
C0
S0
Sum=0
FA3
FA2
FA1
FA0
Cin=0
Carry2
Carry1
Carry0
Cout
S4
S3
S2
S1
S0


Z3 Y3 X3
Z2 Y2 X2
Z1 Y1 X1
Z0 Y0 X0
FA3
FA2
FA1
FA0
C3
S3
C2
S2
C1
S1
C0
S0
Sum=0
FA3
FA2
FA1
FA0
Cin=0
Carry2
Carry1
Carry0
Cout
S4
S3
S2
S1
S0


Thank
you!