APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

LOGIC SYSTEM DESIGN (CST-203)

MODULE 3
NOTES

PREPARED BY : JIBIN EP, ASSISTANT PROFESSOR, EKC TC MANJERI

INDEX

SL NO	TOPIC	SLIDE LINK	VIDEO LINK
1	Introduction To Combinational Circuits	INTRODUCTION TO COMBINATIONAL CIRCUITS	https://youtu.be/gvTM4oGD_XU
2	Design Of Adders (Half Adder And Full Adder)	DESIGN OF ADDERS (HALF ADDER AND FULL ADDER)	https://youtu.be/gvTM4oGD_XU
3	Design Of Subtractors (Half Subtractor And Full Subtractor)	DESIGN OF SUBTRACTORS (HALF SUBTRACTOR AND FULL SUBTRACTOR)	https://youtu.be/FOGczanvVEk
4	Multiplexer And Demultiplexer	MULTIPLEXER AND DEMULTIPLEXER	https://youtu.be/J8-fKhmLWuQ https://youtu.be/Tt_cZOn2YBA
5	Multiplexer And Demultiplexer (Numerical Problesm)	MULTIPLEXER AND DEMULTIPLEXER (NUMERICAL PROBLEMS)	https://youtu.be/2p9Ex6YPBc4
6	Decoder And Encoder	DECODER AND ENCODER	https://youtu.be/Zxd-imLGIFc
7	Code Converters	CODE CONVERTERS	https://youtu.be/cjAMfV-HubQ
8	Magnitude Comparator	MAGNITUDE COMPARATORS	https://youtu.be/7xHM5HeGCHo
9	Parity Generator And Checker	PARITY GENERATOR AND CHECKER	https://youtu.be/7xHM5HeGCHo
10	Binary Parallel Adder, Carry Look Ahead Adder & Bcd Adder	BINARY PARALLEL ADDER CARRY LOOK AHEAD ADDER BCD ADDER	https://youtu.be/8G29l4BFg_o https://youtu.be/hfbvmGZ79SA

Module III

Combinational Logic Circuits

Design Procedure & Implementation of combinational logic circuits- Binary adders and subtractors, Binary Parallel adder, Carry look ahead adder, BCD adder, Code converter, Magnitude comparator, Decoder, Demultiplexer, Encoder, Multiplexer, Parity generator/ Checker.

MODULE 3
(TOPIC-1)

INTRODUCTION TO COMBINATIONAL CIRCUITS

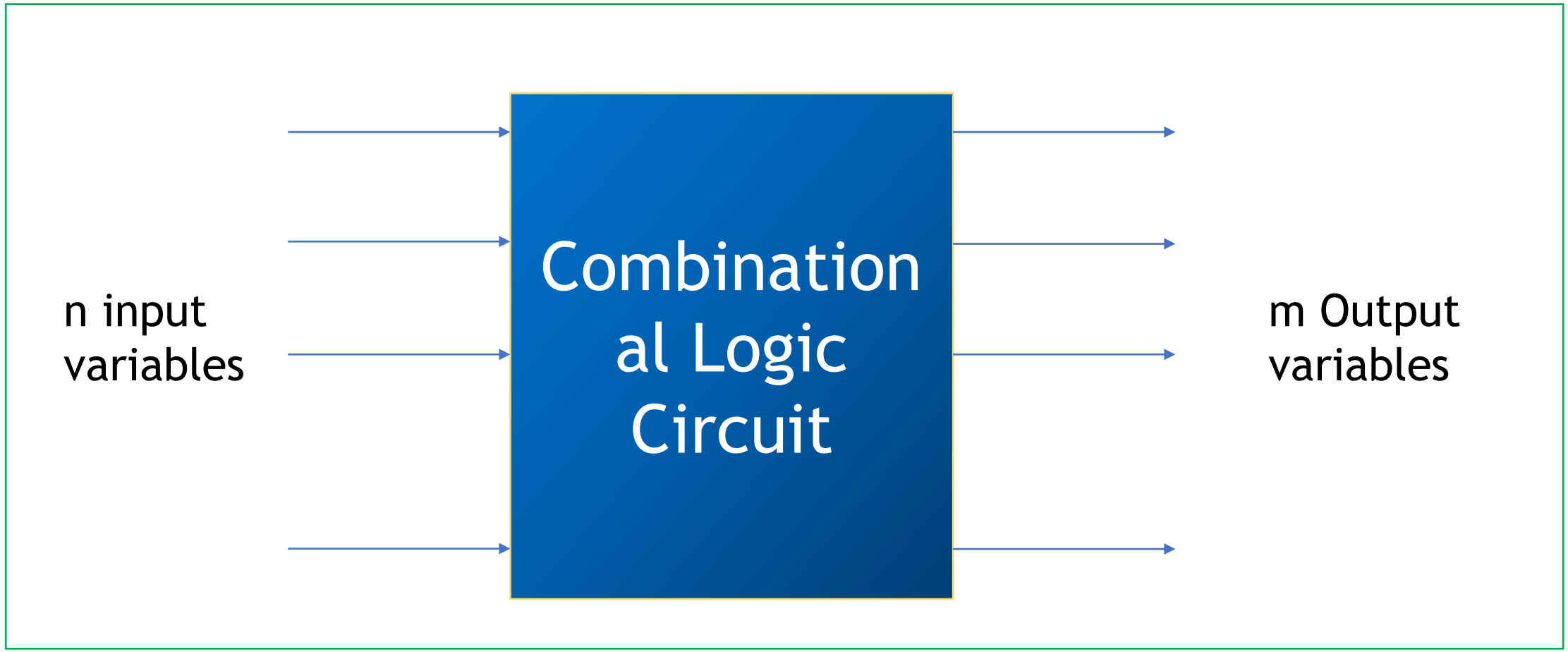
https://youtu.be/gvTM4oGD_X

YouTube

Combinational Logic Circuits

- When Logic Gates are connected together to produce a specified output for certain specified combination of input variables, with no storage involved , the resulting circuit is called combinational logic circuit
- In Combinational Logic Circuits, the output variables are all time depends on combination of input variables
- A Combinational Circuit consist of Input variables, Logic Gates & Output Variables.
- The Logic Gates accepts signal from the input variable and generate output signals

Block Diagram of Combinational Logic Circuit



TYPES OF COMBINATIONAL CIRCUITS

- ADDER
  - Half Adder
  - Full Adder
- SUBTRACTOR
  - Half Subtractor
  - Full Subtractor
- MULIPLEXER MUX
  - 2 :1
  - 4 : 1
  - 8:1
- DEMULTIPLEXER DEMUX
  - 1:2
  - 1:4
  - 1:8
- DECODER
- ENCODER

Design Procedure

1. Problem Definition
2. Determination of input and output variables
3. Assigning letter symbols to input and output variables
4. Derivation of Truth Table
5. Obtain Simplified Boolean Expression
6. Obtain the Logic Diagram

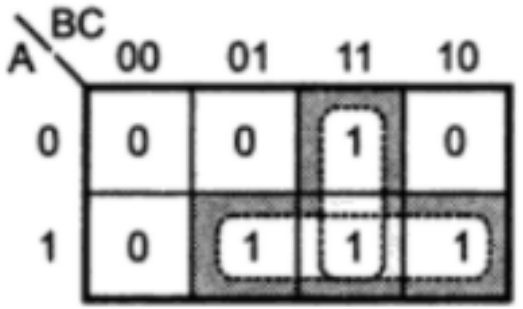
Example 1 :

Design a combinational Logic Circuit with three input variables that will produce a logic 1 output when more than one input variable are logic 1

TRUTH TABLE

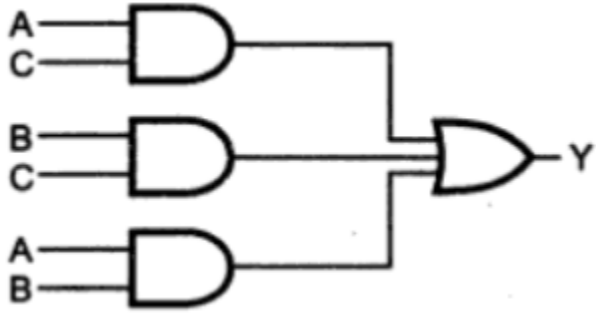
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

BOOLEAN EXPRESSION



$$Y=AC + BC + AB$$

LOGIC DIAGRAM



Example 2 :

A Majority gate is digital circuit whose output is equal to 0 if majority of inputs are 1's . The output is 1 otherwise. Using TT Find Boolean Function and Logic Diagram

TRUTH TABLE

BOOLEAN EXPRESSION

LOGIC DIAGRAM

MODULE 3
(TOPIC-2)

DESIGN OF ADDERS
(HALF ADDER AND FULL ADDER)

https://youtu.be/gvTM4oGD_XU

Design of Adders

0+0 =0
0+1 =1
1+0=1
1+1= 10

(1+1= Sum 0, Carry 1)

- The Simple addition consist of four possible elementary operations
- The First 3 Operations Produces Sum whose length is one digit
- Last Operation Performed sum in two digits. Higher Significant result is Called Carry & Lower Significant bit is Called Sum

HALF ADDER

- Half adder is a combinational logic circuit with two inputs and two outputs.
- The half adder circuit is designed to add two single bit binary number A and B.
- It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

HALF ADDER

0+0 =0,0+1 =1
1+0=1,1+1= 0

TRUTH TABLE

Inputs		Outputs	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

BOOLEAN EXPRESSION

For Carry

A \ B	0	1
0	0	0
1	0	(1)

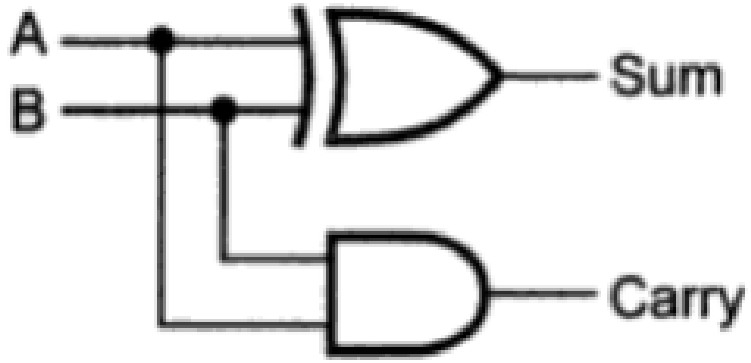
Carry = AB

For Sum

A \ B	0	1
0	0	(1)
1	(1)	0

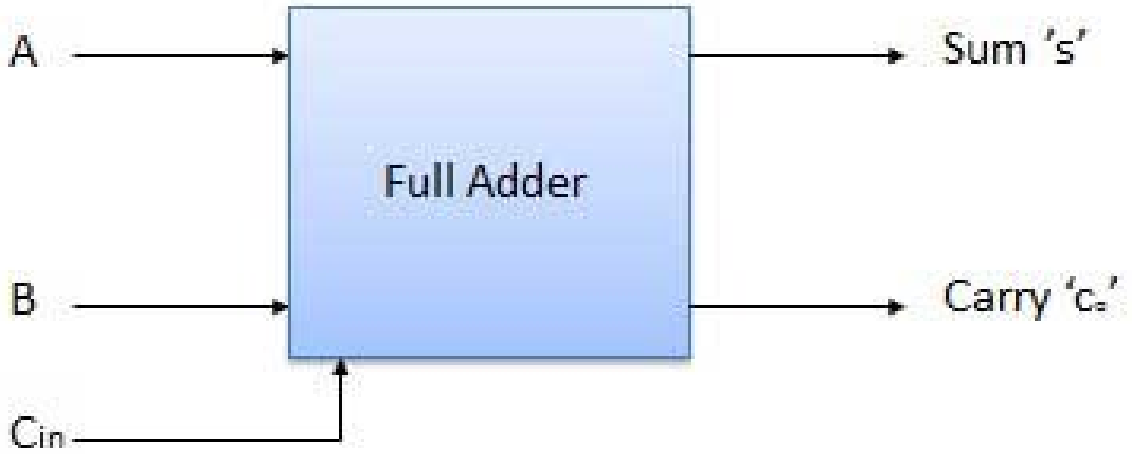
Sum = $A\bar{B} + \bar{A}B$
= $A \oplus B$

LOGIC DIAGRAM



FULL ADDER

- Full adder is developed to overcome the drawback of Half Adder circuit.
- It can add two one-bit numbers A and B, and C.
- The full adder is a three input and two output combinational circuit.



FULL ADDER

0+0 =0,0+1 =1
1+0=1,1+1= 0

TRUTH TABLE

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BOOLEAN EXPRESSION

For Carry (C_{out})

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_{out} = AB + A C_{in} + B C_{in}$

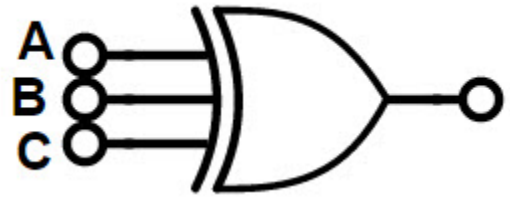
For Sum

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$Sum = \bar{A}\,\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\,\bar{C}_{in} + ABC_{in}$

Reference : 3 Input XOR Gate

Inputs			outputs
W	X	Y	Q = A⊕B⊕C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

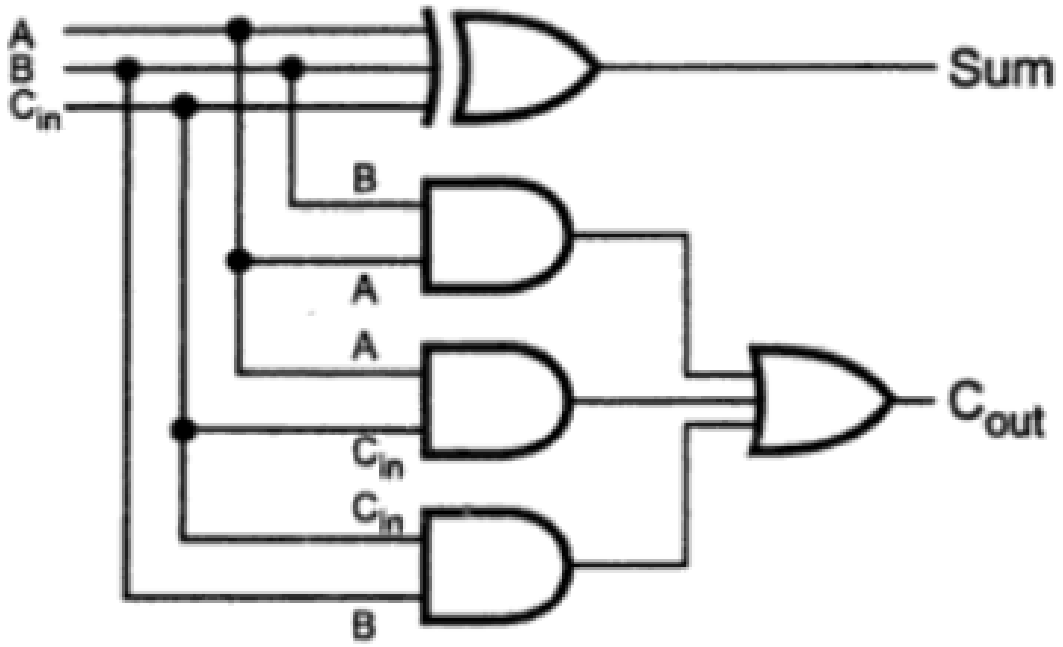


Sum= $A \oplus B \oplus C\_in$

FULL ADDER

0+0 =0,0+1 =1
1+0=1,1+1= 0

LOGIC DIAGRAM



Sum= $A \oplus B \oplus C\_in$

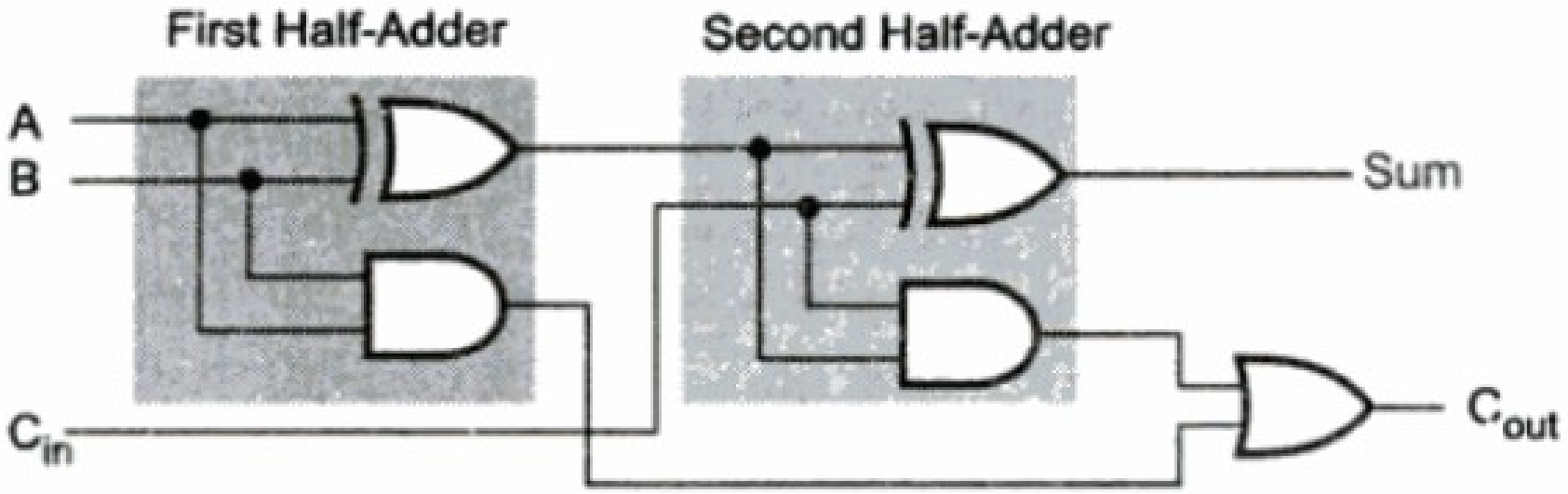
Carry= $AB + AC_{in} + BC_{in}$

FULL ADDER

0+0 =0,0+1 =1
1+0=1,1+1= 0

LOGIC DIAGRAM

Implementation of Full Adder Circuit using two Half Adders



MODULE 3
(TOPIC-3)

DESIGN OF SUBTRACTORS (HALF SUBTRACTOR AND FULL SUBTRACTOR)

https://youtu.be/FOGczanvVEk

Design of Subtractors

0-0 =0

0-1 =1 With Borrow 1

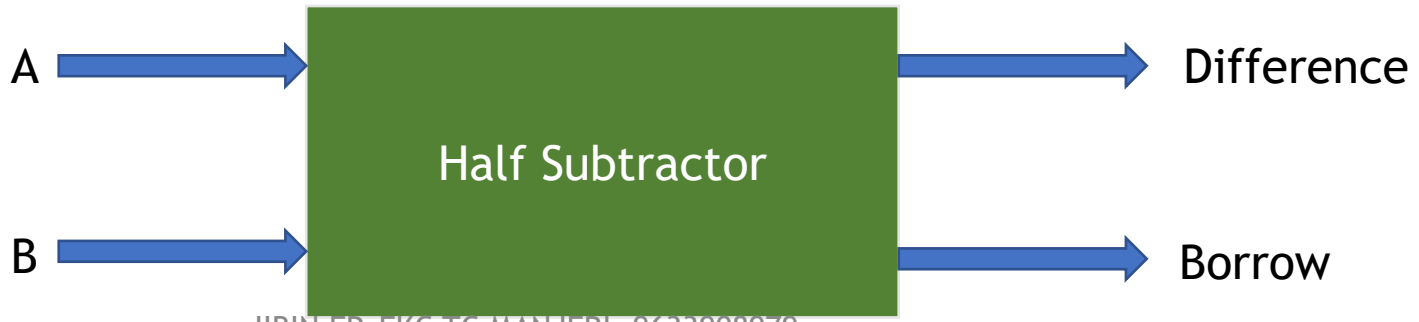
1-0=1

1-1= 0

- The Simple Subtraction consist of four possible elementary operations
- The First , 3rd and 4th Operations Produces Difference whose length is one digit. In these Operations, Each Subtrahend bit is subtracted from Minuend bit
- 2nd Operation the minuend bit is smaller than the subtrahend bit, Hence 1 is borrowed

Half Subtractor

- Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow).
- It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed.
- In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit



HALF SUBTRACTO

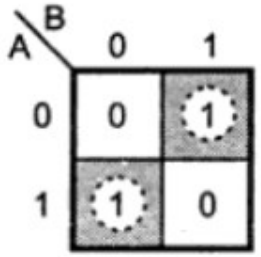
0-0 =0,0-1 =1 WITH BORROW 1
1-0=1,1-1= 0

TRUTH TABLE

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

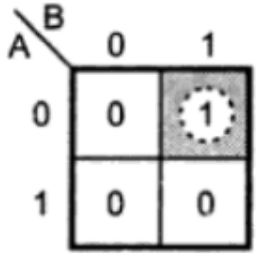
BOOLEAN EXPRESSION

For Difference



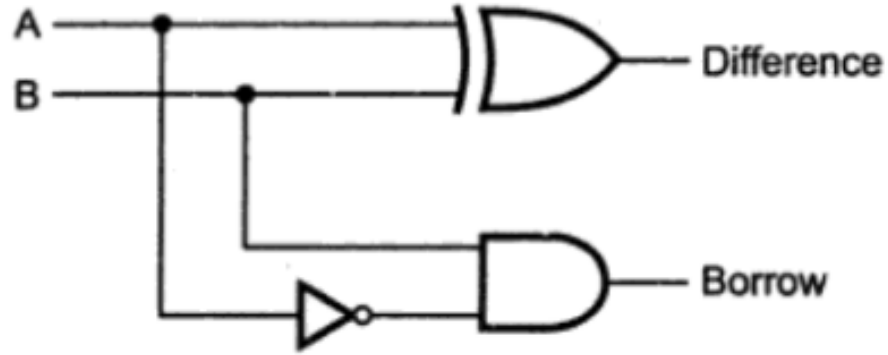
Difference = $A\bar{B} + \bar{A}B$
= $A \oplus B$

For Borrow



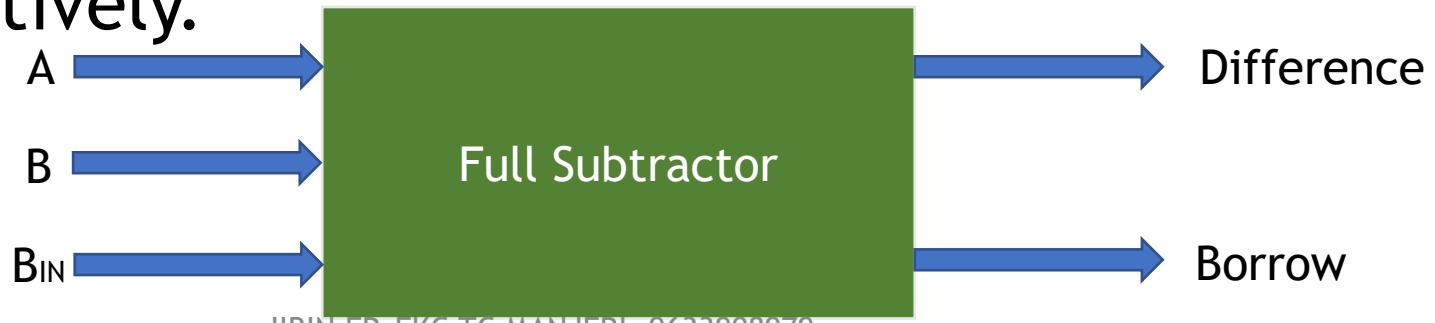
Borrow = $\bar{A}B$

LOGIC DIAGRAM



Full Subtractor

- The disadvantage of a half subtractor is overcome by full subtractor.
- The full subtractor is a combinational circuit with three inputs A,B,B_{in} and two output D and B_{OUT}.
- The three inputs A, B and Bin, denote the Minuend, Subtrahend, and Previous Borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.



FULL SUBTRACTO

0-0 =0,0-1 =1 WITH BORROW 1
1-0=1,1-1= 0

TRUTH TABLE

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

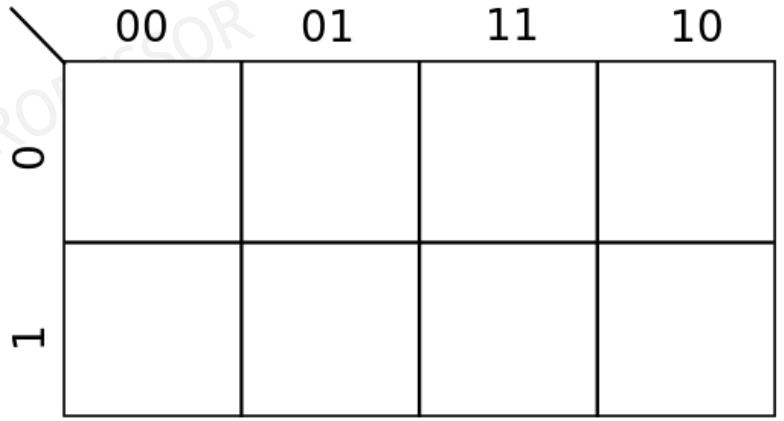
FULL SUBTRACTO

0-0 =0,0-1 =1 WITH BORROW 1
1-0=1,1-1= 0

TRUTH TABLE

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSION



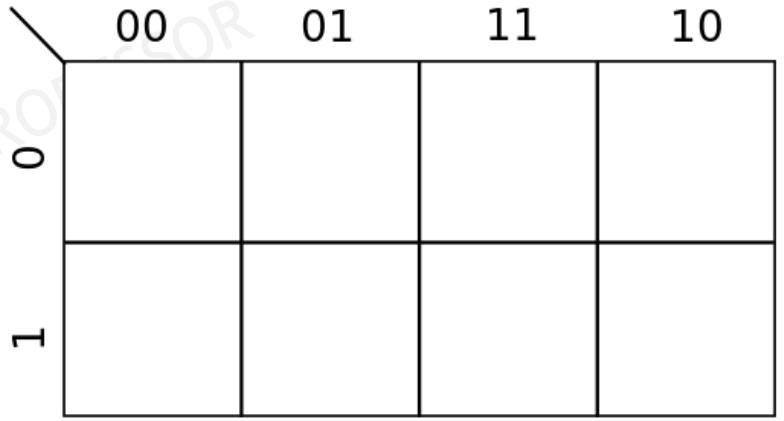
FULL SUBTRACTO

0-0 =0,0-1 =1 WITH BORROW 1
1-0=1,1-1= 0

TRUTH TABLE

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSION



FULL SUBTRACTO

0-0 =0,0-1 =1 WITH BORROW 1
1-0=1,1-1= 0

TRUTH TABLE

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSION

For D

A \ BB_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$D = \bar{A}\bar{B}B_{in}+\bar{A}B\bar{B}_{in}+A\bar{B}\bar{B}_{in}+ABB_{in}$$

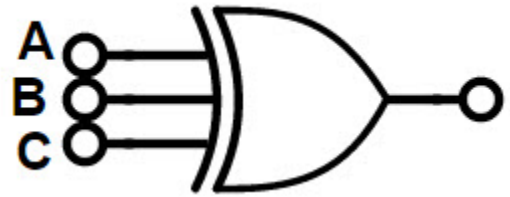
For B_{out}

A \ BB_{in}	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$$B_{out} = \bar{A}B_{in}+\bar{A}B+BB_{in}$$

Reference : 3 Input XOR Gate

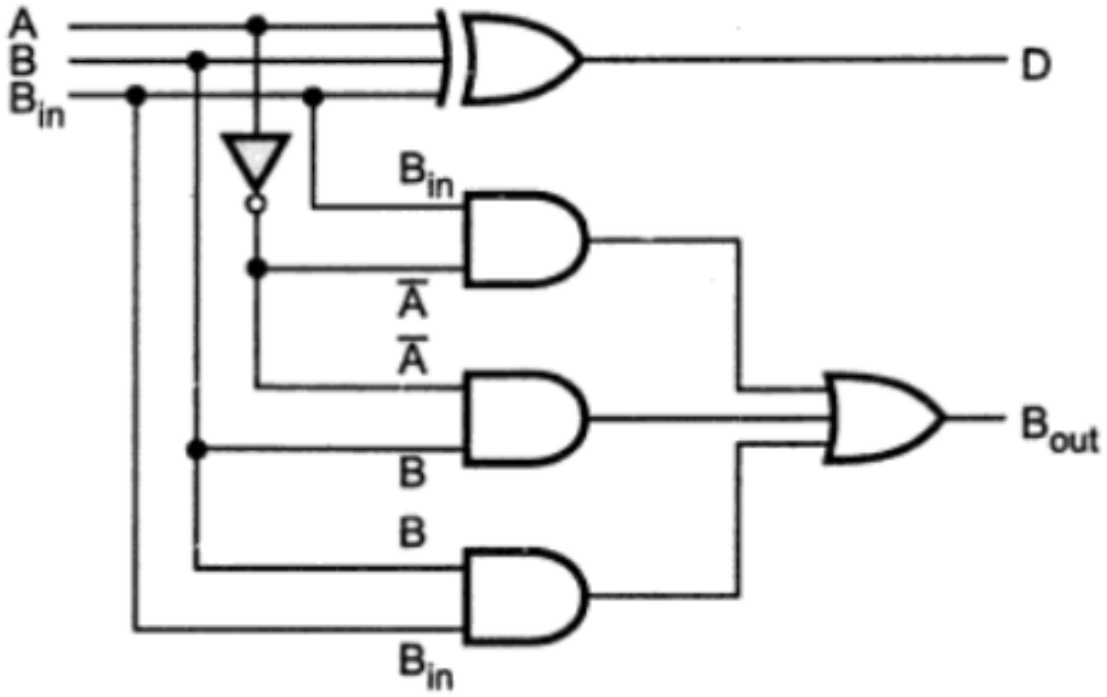
Inputs			outputs
W	X	Y	Q = A⊕B⊕C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Sum= $A \oplus B \oplus C\_in$

FULL SUBTRACTOR

LOGIC DIAGRAM



Difference

$$B_{in} \oplus (A \oplus B)$$

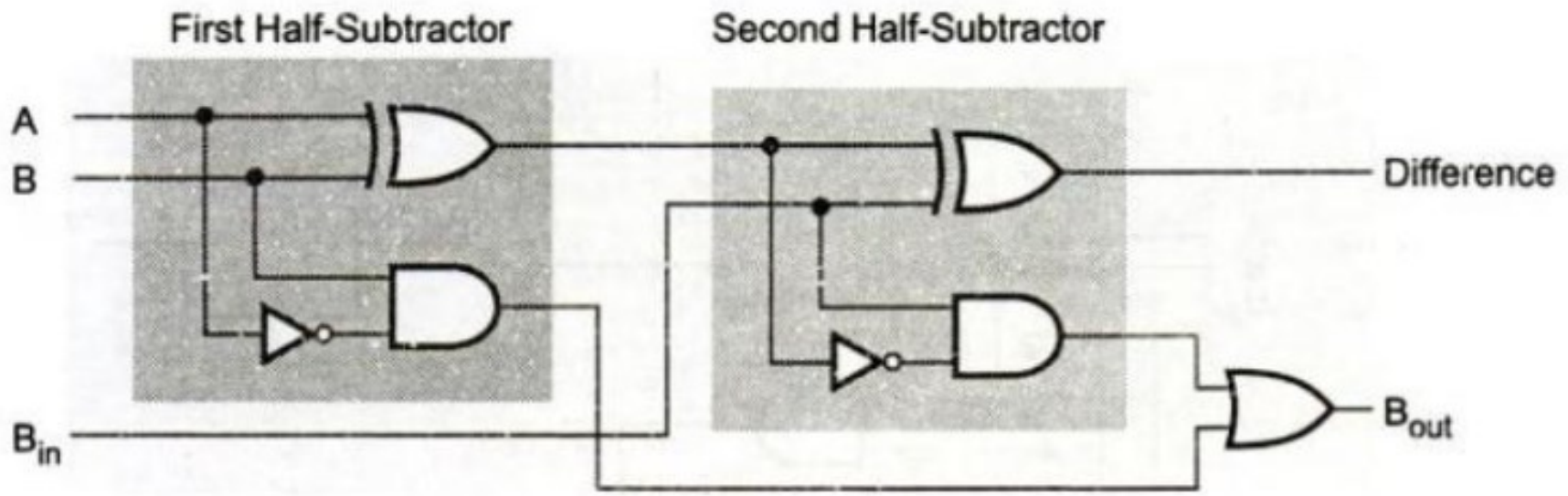
Borrow

$$B_{out} = \overline{A}B_{in} + \overline{A}B + BB_{in}$$

FULL SUBTRACTOR

LOGIC DIAGRAM

Implementation of Full Subtractor Circuit using two Half Subtr



MODULE 3
(TOPIC-4)

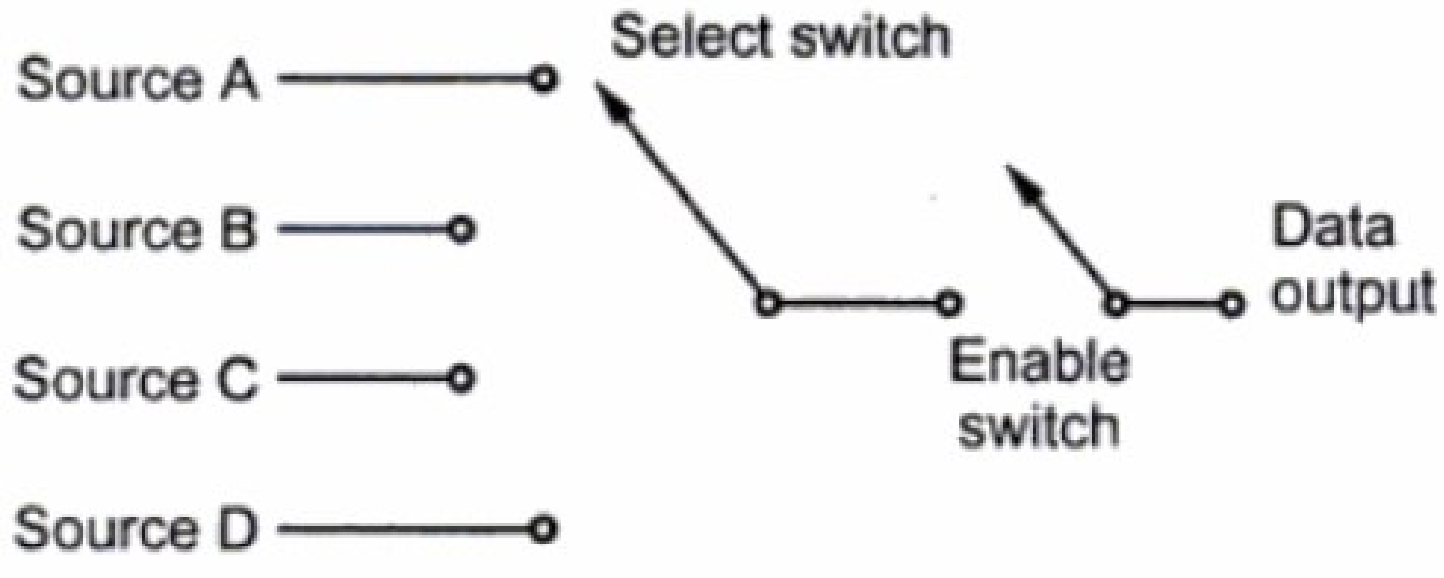
MULTIPLEXER AND DEMULTIPLEXER

https://youtu.be/J8-fKhmLWuQ
https://youtu.be/Tt_cZOn2YBA

MULTIPLEXER (MUX)

- Multiplexer is a digital circuit that has multiple inputs and a single output.
- It is also known as Data Selector
- The Section of particular input line is controlled by set of Selection lines
- Normlly there are 2^n Input lines and n Selection Lines Whose bit combination determines which input is selected
- Therefore Multiplexer is 'Many to One' and it provide digital equivalent of an analog selector switch

MULTIPLEXER (MUX)



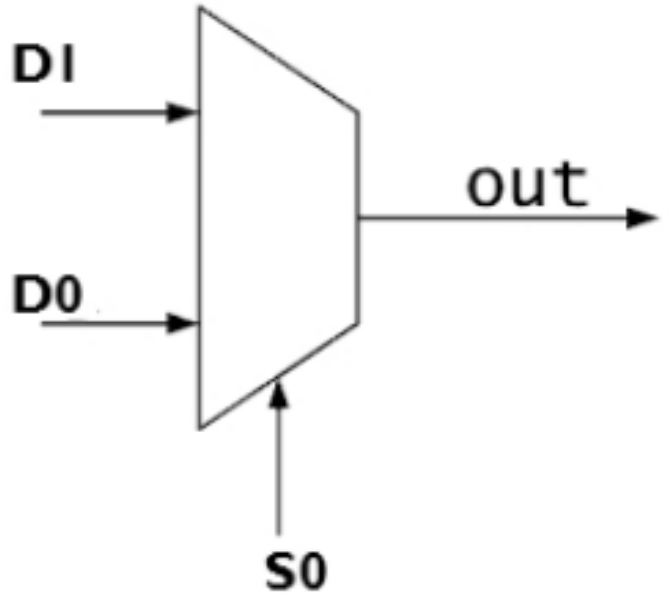
TYPES OF MULIPLEXE

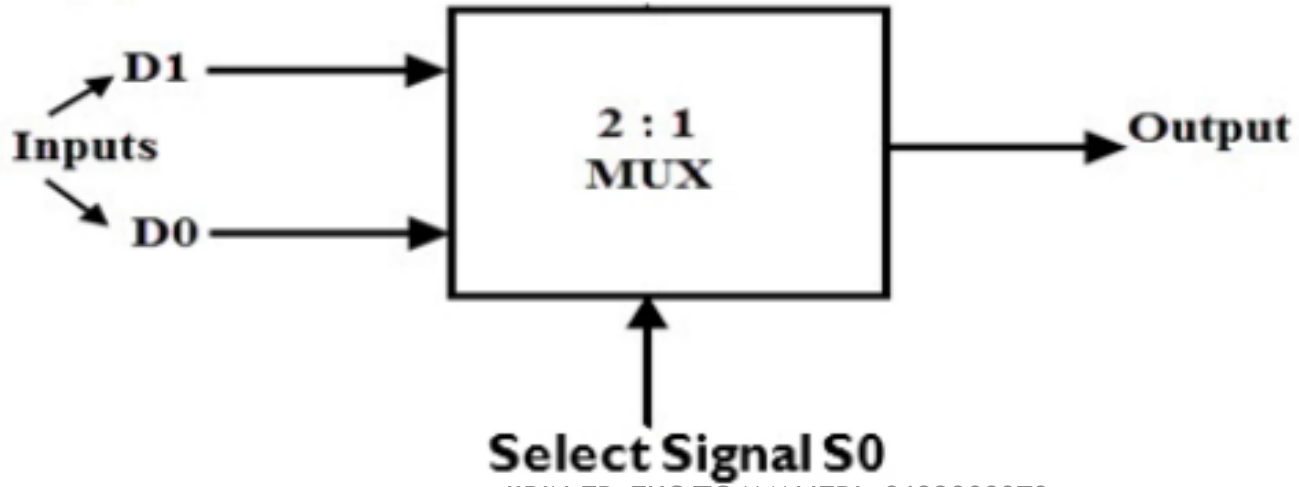
n Selection Lines
2^n Input Lines
1 Output Line

- 2 :1 Mux
- 4 :1 Mux
- 8 : 1 Mux

2:1 MULTIPLEXER

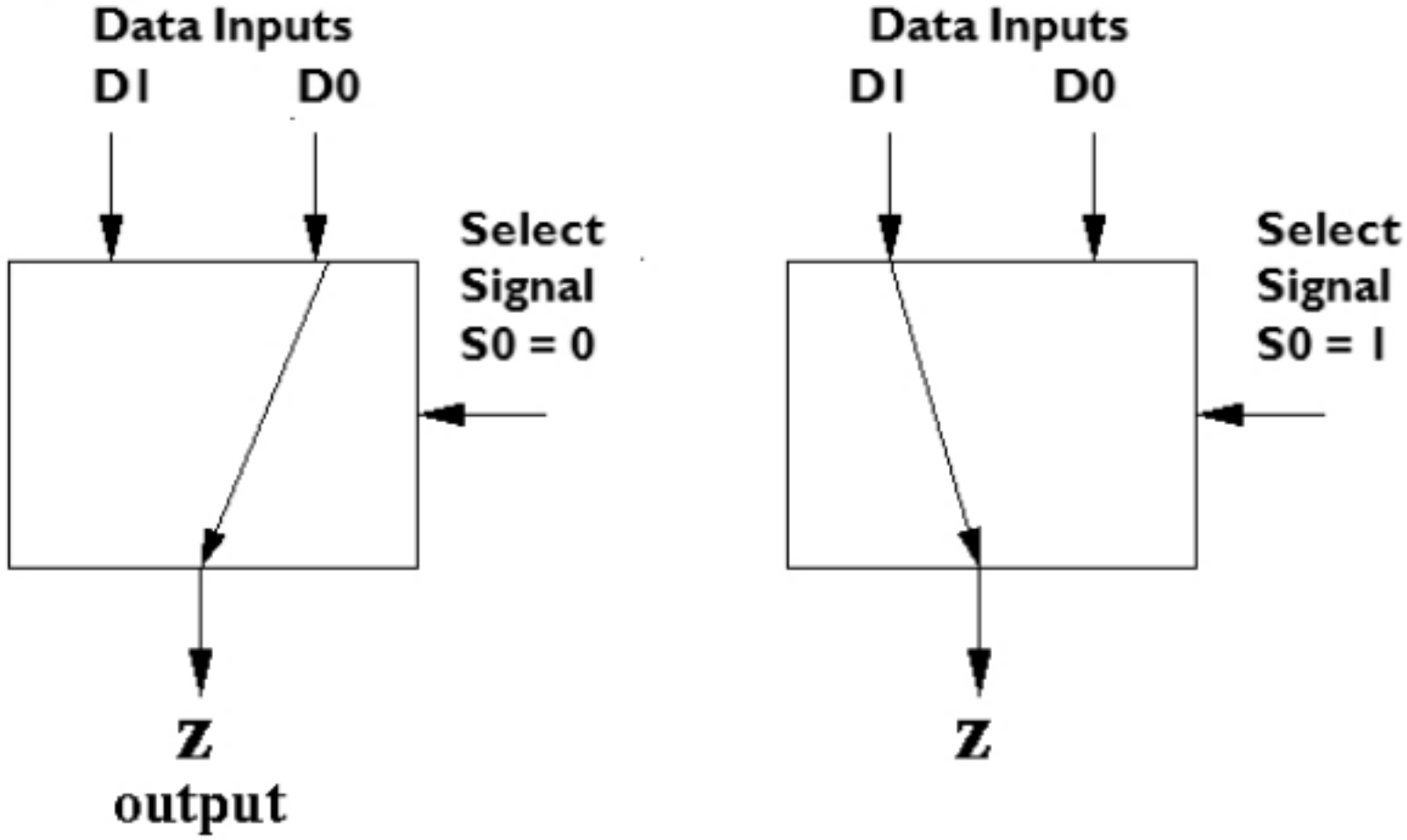
- 2:1 Multiplexer has
  - two data inputs, D0 and D1
  - one select line S0.
  - One output Z.





OPERATION OF 2:1 MUX

- If Select Line S0=0, then output = D0 and if Select Line S0=1, then output = D1



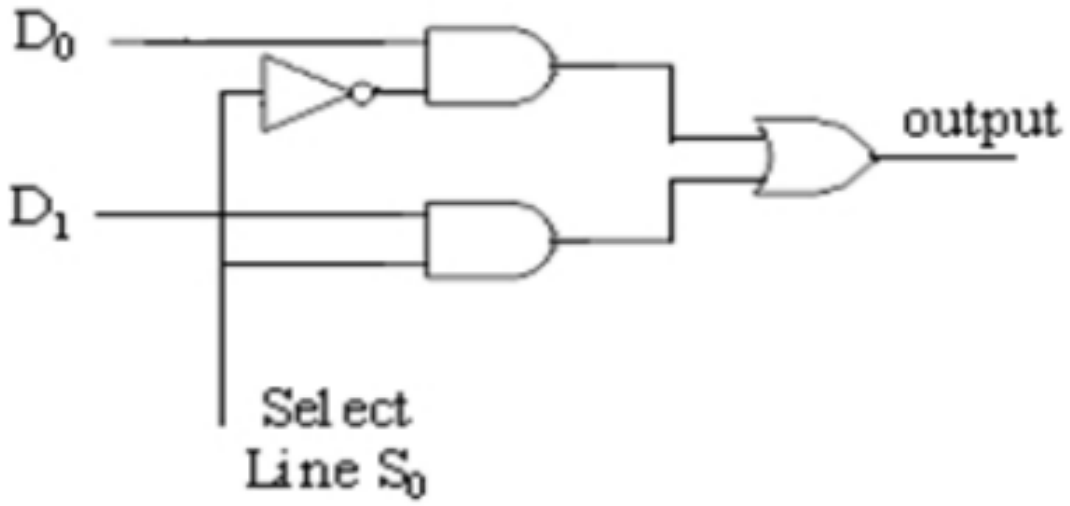
2:1 MUX

S0	Output Z
0	D0
I	D I

$$Z = D0.\overline{S0} + D1.S0$$

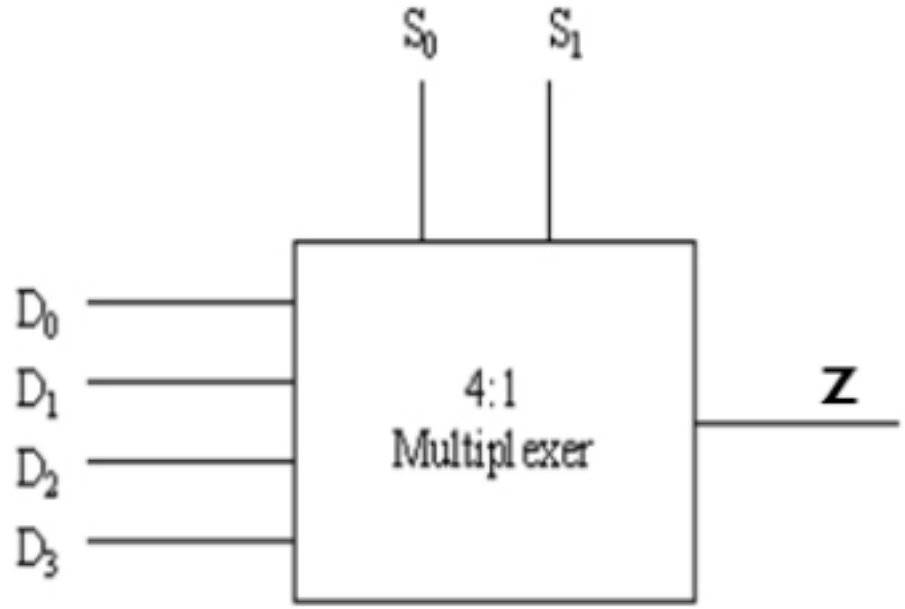
- 2:1 MUX requires
  - 2 AND Gates
  - 1 NOT Gate
  - 1 OR Gate

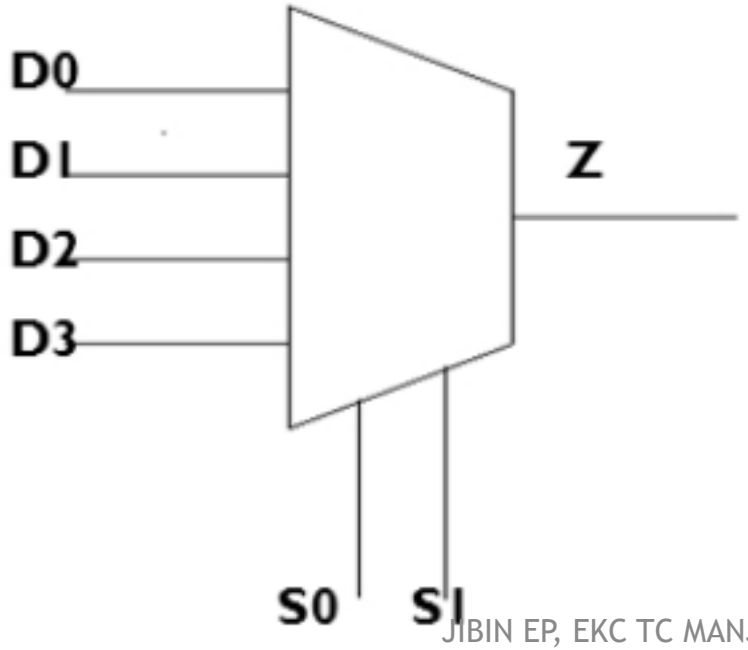
Circuit



4:1 MULTIPLEXER

- 4:1 Multiplexer has
  - four data inputs, D0, D1, D2 and D3
  - Two select lines S0 and S1.
  - One output Z.





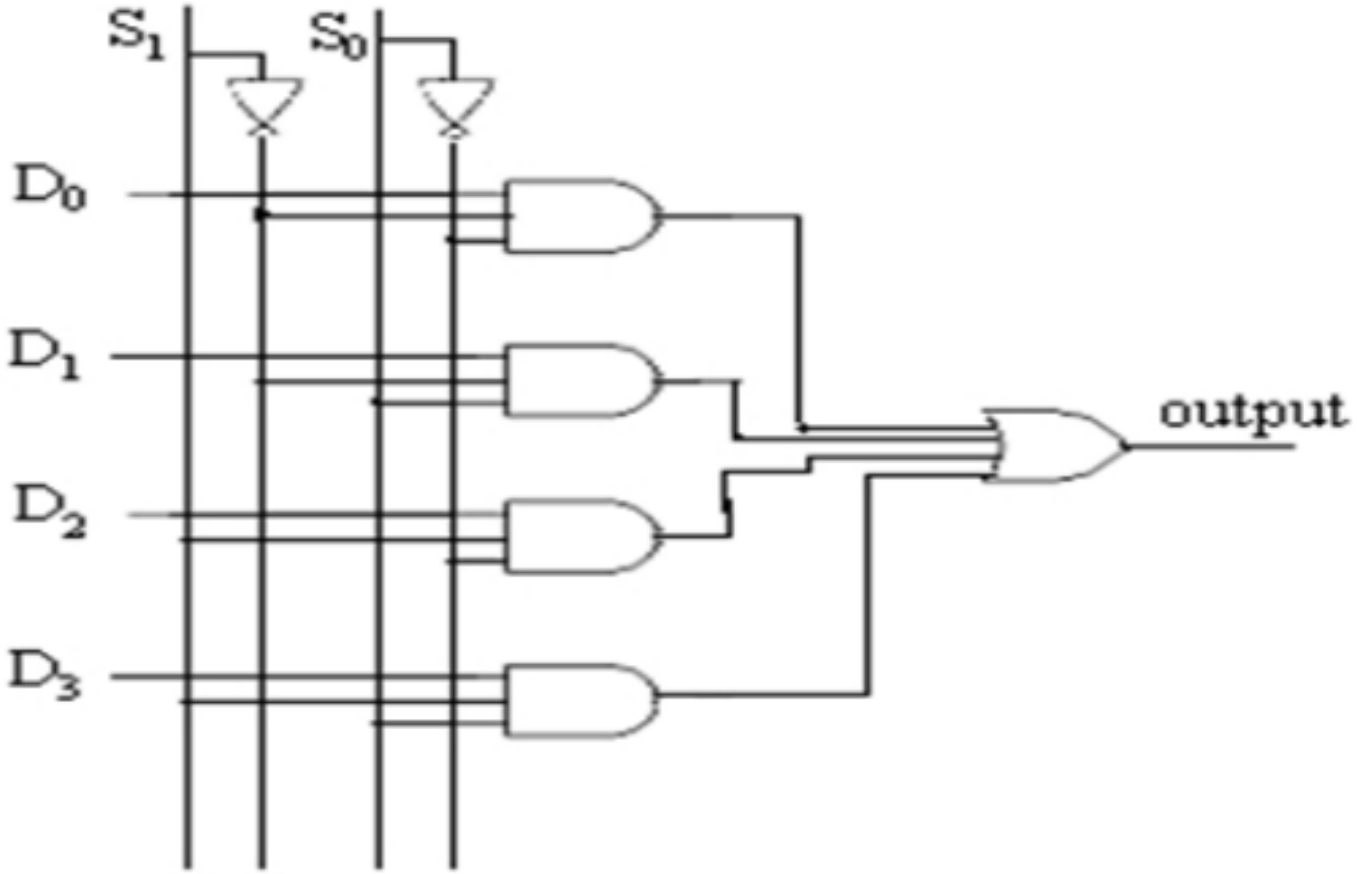
TRUTH TABLE OF 4:1 MUX

S1	S0	Output Z
0	0	D0
0	1	D1
1	0	D2
1	1	D3

$$Z = D0.\overline{S1}.\overline{S0} + D1.\overline{S1}.S0 + D2.S1.\overline{S0} + D3.S1.S0$$

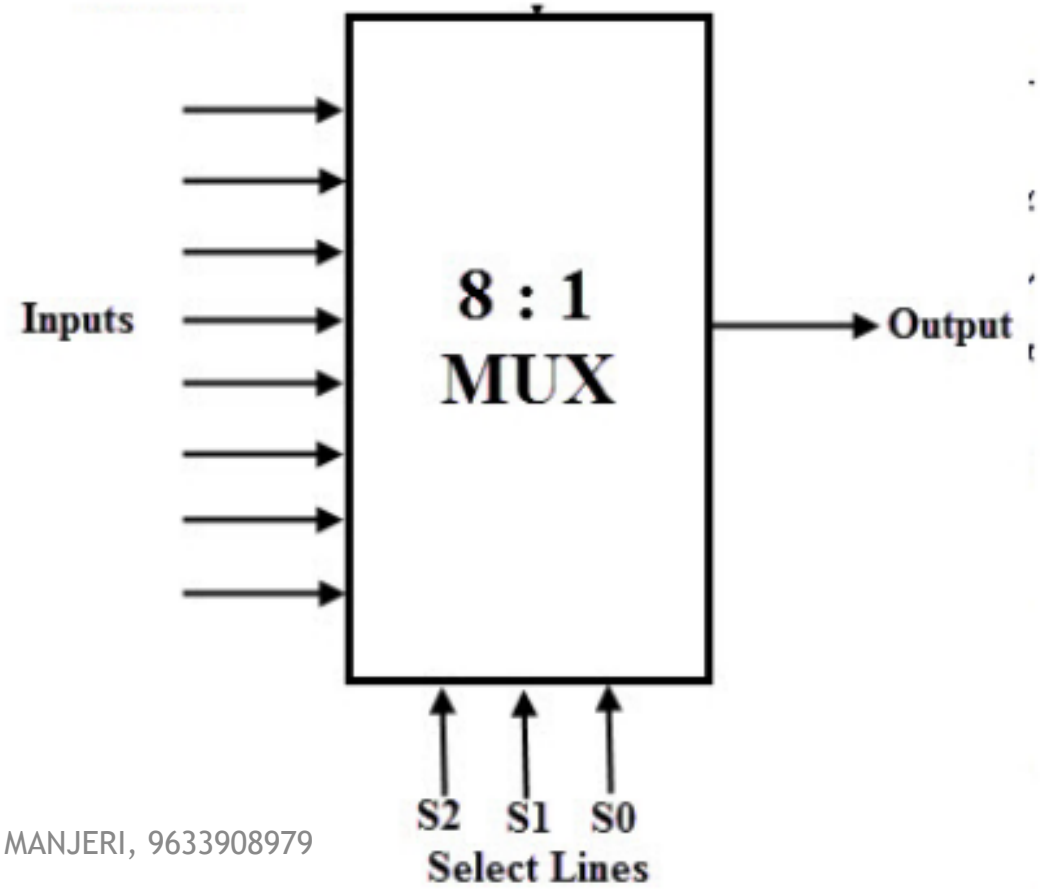
CIRCUIT OF 4:1 MUX

$$Z = D0.\overline{S1}.\overline{S0} + D1.\overline{S1}.S0 + D2.S1.\overline{S0} + D3.S1.S0$$



8:1 MULTIPLEXER

- 4:1 Multiplexer has
  - Eight data inputs, D0, D1, D2, D3, D4, D5, D6 and D7
  - Three select lines S0, S1 and S2.
  - One output Z.

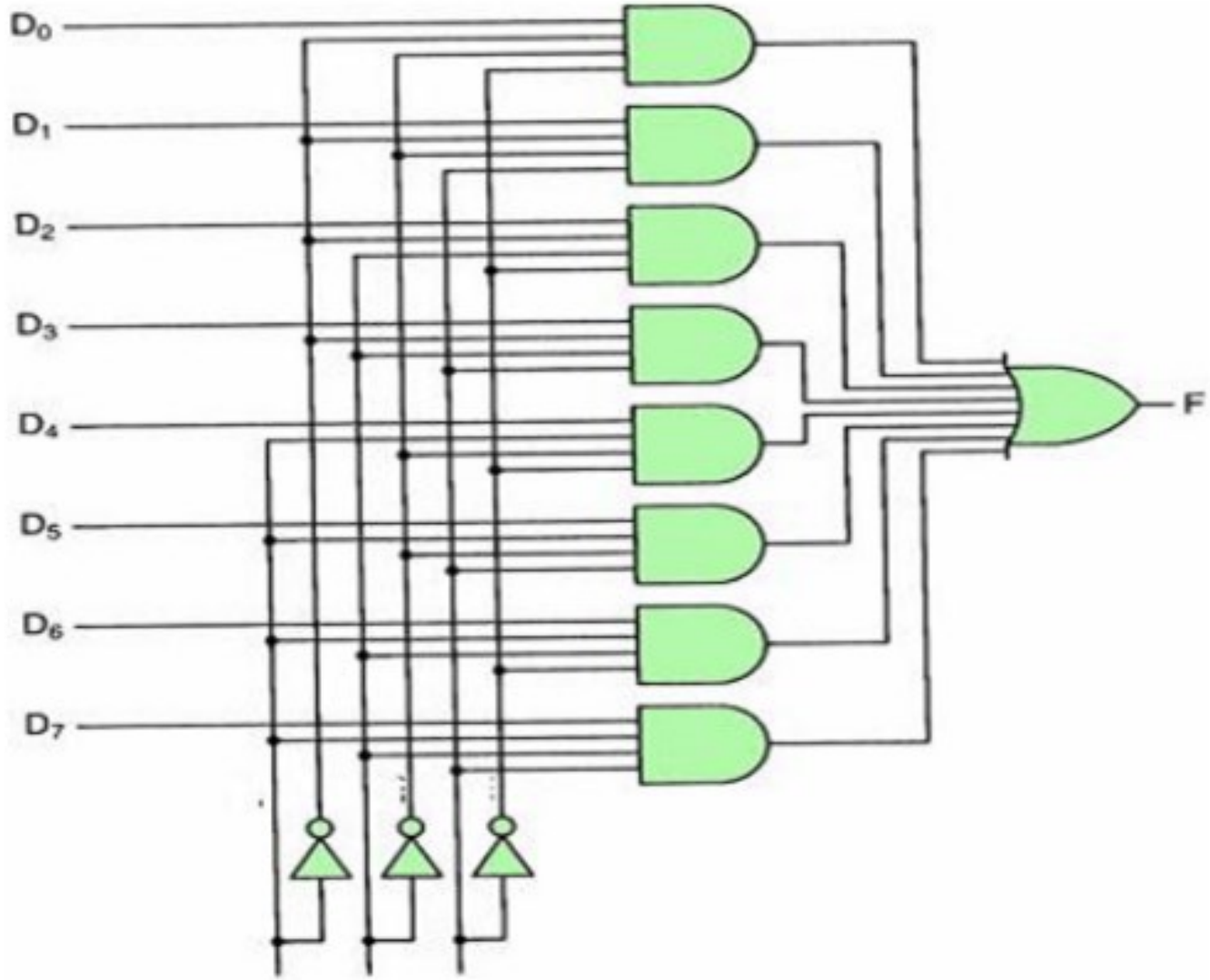


TRUTH TABLE OF 8:1 MUX

S2	S1	S0	Output Z
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

$$Z = S2'.S1'.S0'.D0 + S2'.S1'.S0.D1 + S2'.S1.S0'.D2 + S2'.S1.S0.D3 + S2.S1'.S0'.D4 + S2.S1'.S0.D5 + S2.S1.S0'.D6 + S2.S1.S0.D7$$

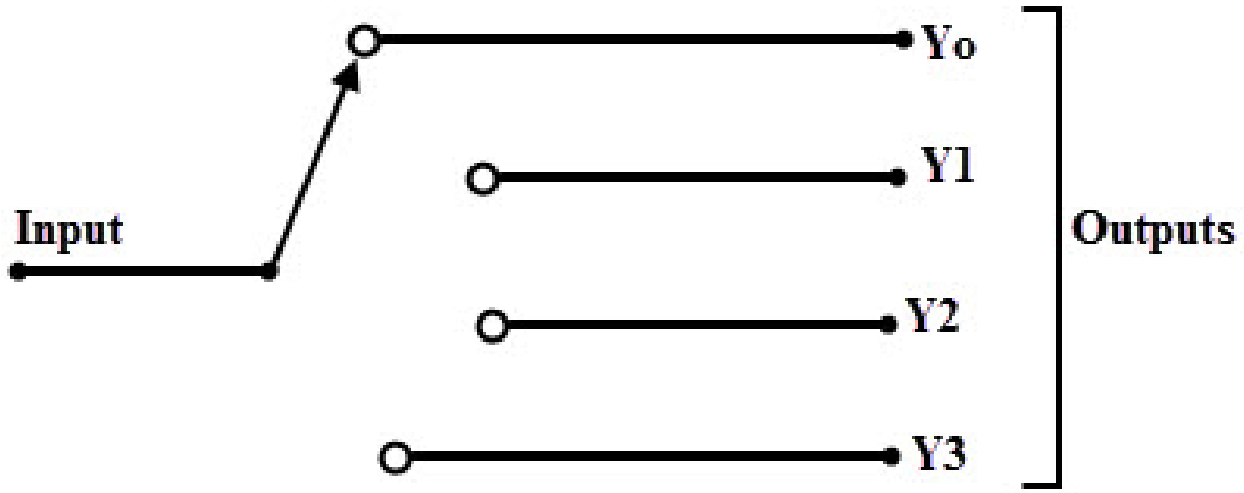
CIRCUIT OF 8:1 MUX



DEMULTIPLEXER (DEMUX)

- Demultiplexer is a digital circuit that has multiple Outputs and a single Input.
- It is also known as Data Distributer
- The Section of particular Output line is controlled by set of Selection lines
- Normlly there are 2^n Output lines and n Selection Lines Whose bit combination determines which input is selected
- Therefore demultiplexer is 'One to Many'

DEMULTIPLEXER (DEMUX)



TYPES OF DEMULIPLE

n Selection Lines
2^n Output Lines
1 Input Line

- 1 :2 Mux
- 1 :4 Mux
- 1 : 8 Mux

1x2 DEMUX

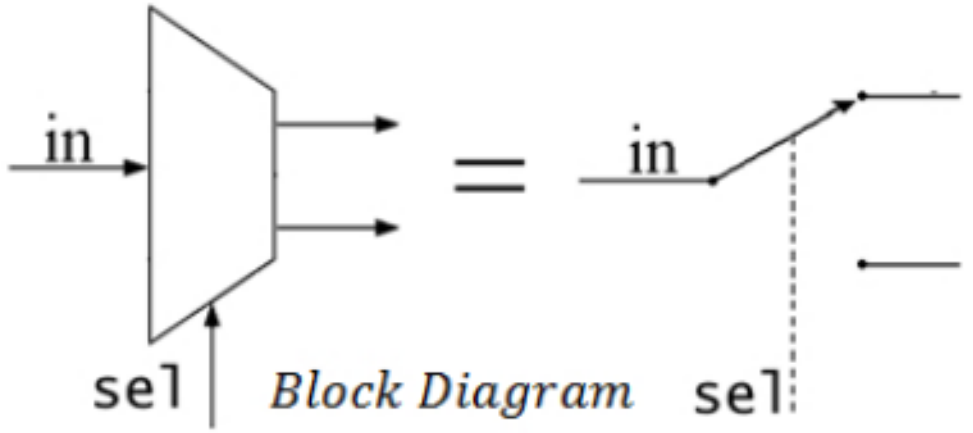
It has **one select line** and **two outputs,** Y_0 and Y_1. The input connected is F.

TRUTH TABLE:

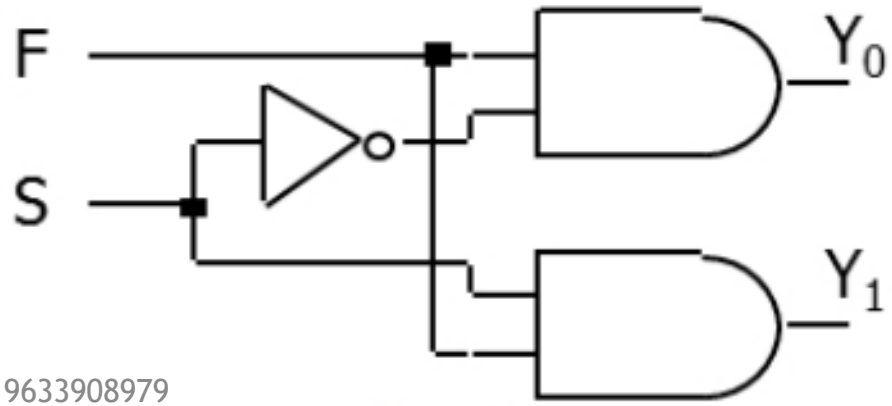
S_0	Y_0	Y_1
0	F	0
1	0	F

$$Y_o = \overline{S_o}.F + S_o.0 = \overline{S_o}.F$$

$$Y_1 = \overline{S_o}.0 + S_o.F = S_o.F$$



Block Diagram



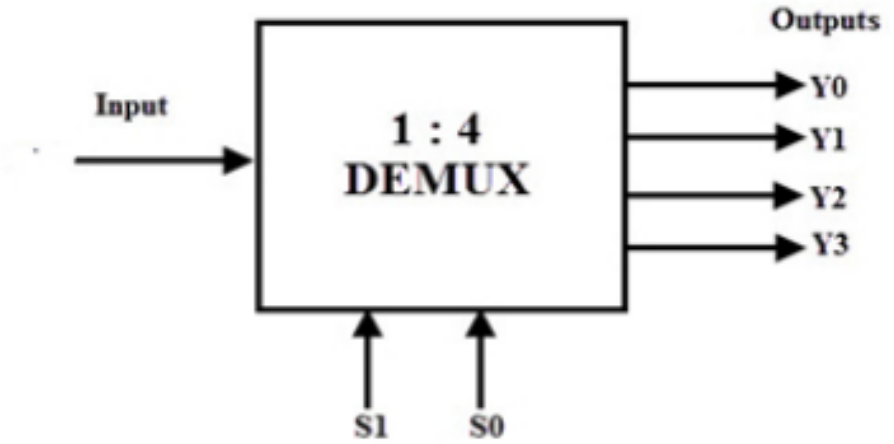
Circuit

1x4 DEMUX

It has **two select lines** and **four outputs,** Y_0 , Y_1, Y_2 and Y_3. The input connected is F.

TRUTH TABLE:

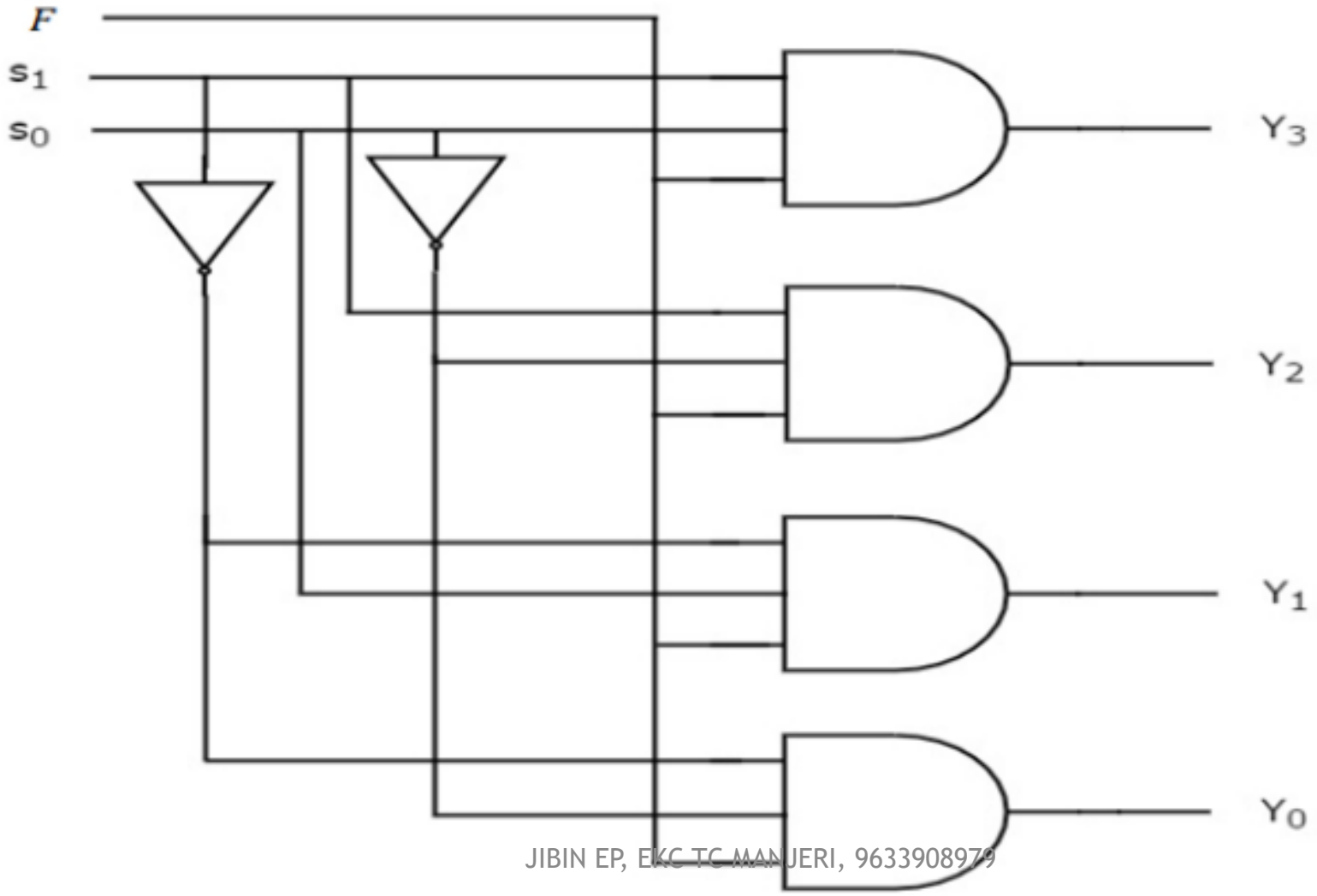
S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	F	0	0	0
0	1	0	F	0	0
1	0	0	0	F	0
1	1	0	0	0	F

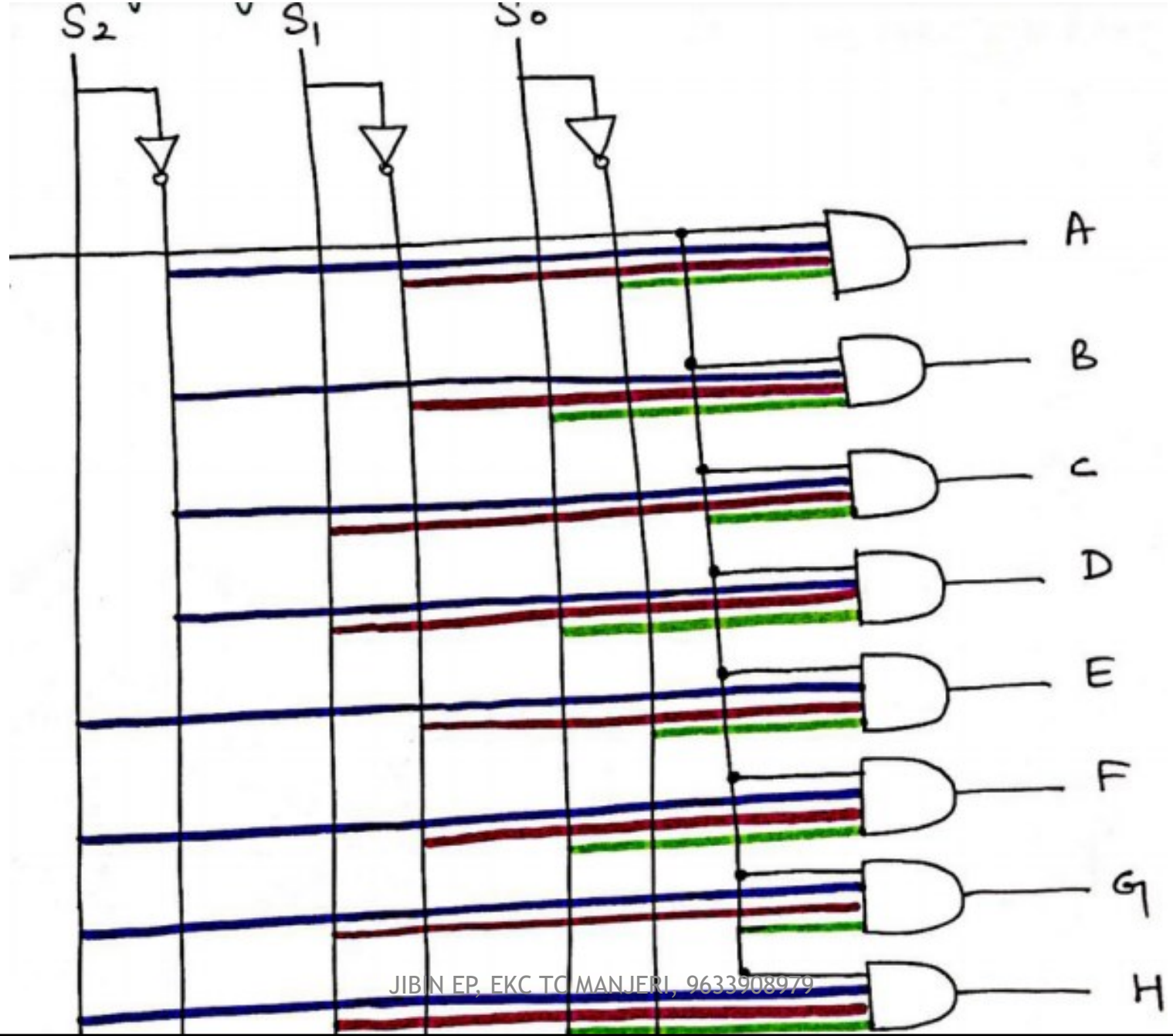


$$Y_o = \overline{S_1}.\overline{S_0}.F \qquad Y_1 = \overline{S_1}.S_0.F$$

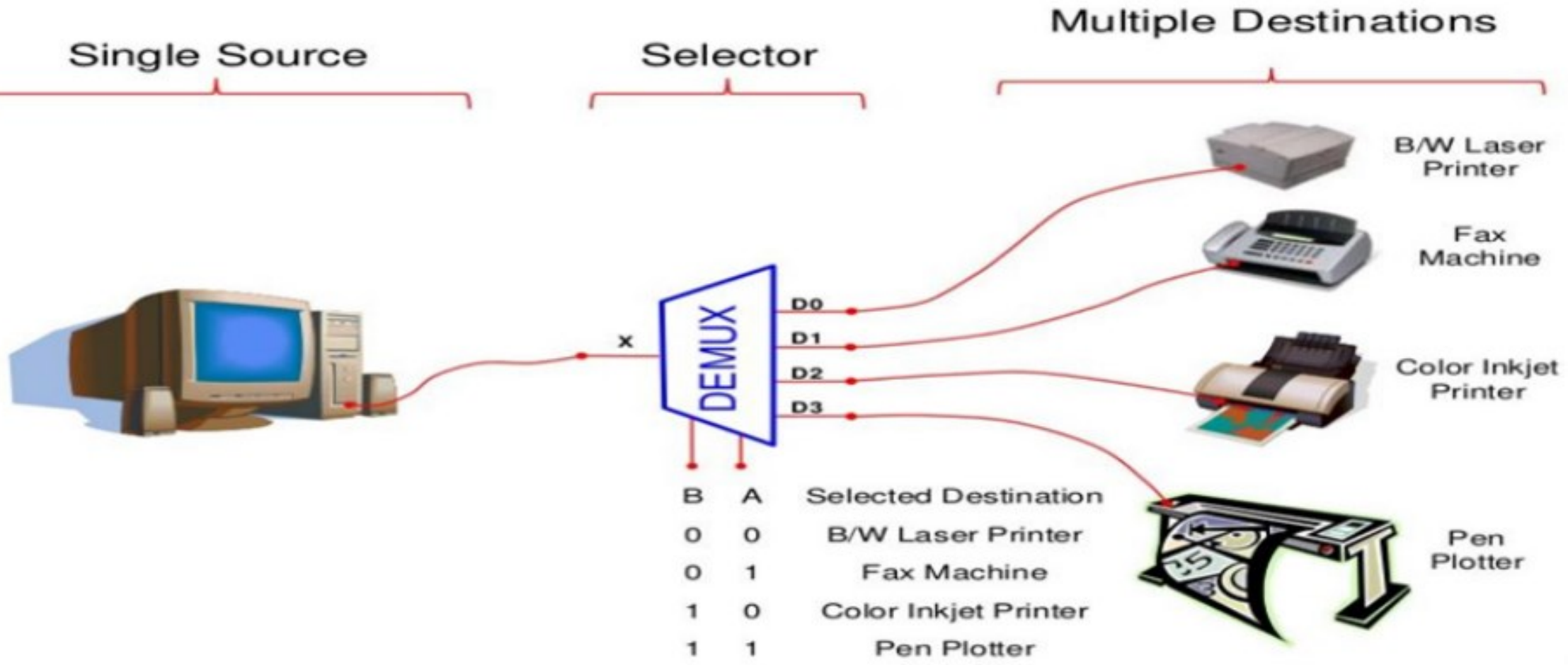
$$Y_2 = S_1.\overline{S_0}.F \qquad Y_3 = S_1.S_0.F$$

1x4 DEMUX CIRCUIT




S2
S1
S0
A
B
C
D
E
F
G
H

APPLICATION OF DEMUX



B	A	Selected Destination
0	0	B/W Laser Printer
0	1	Fax Machine
1	0	Color Inkjet Printer
1	1	Pen Plotter

MODULE 3
(TOPIC-5)

MULTIPLEXER AND DEMULTIPLEXER (NUMERICAL PROBLEMS)

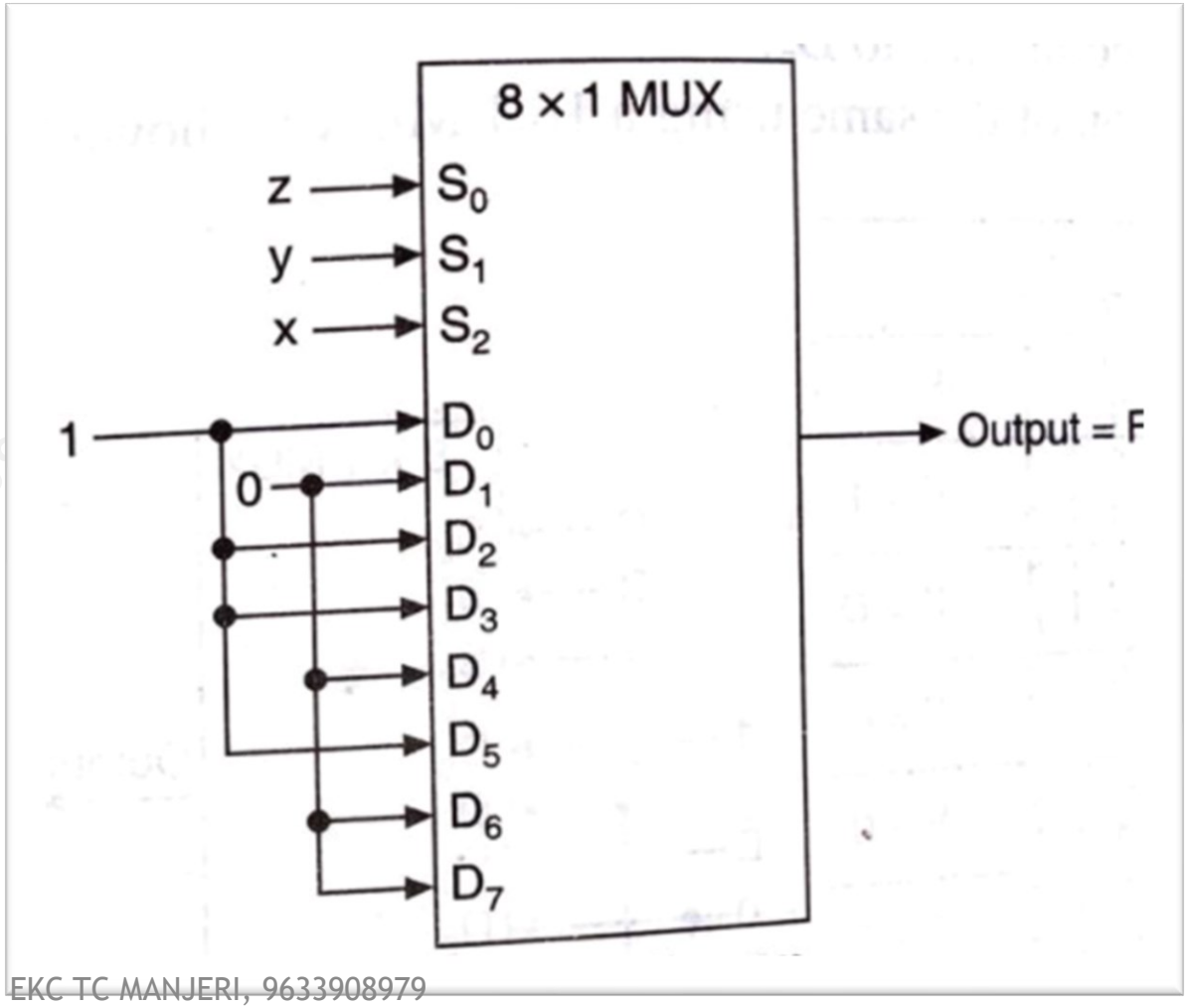
https://youtu.be/2p9Ex6YPBc4

Problem 1:

Implement the following function using 8-to-1 Mux

$f(x,y,z) = \sum m(0,2,3,5)$

S2	S1	S0	F
x	y	z	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Problem 2:

Use a 4:1 Mux to implement the logic function

$f(A,B,C) = \sum m(1,2,4,7)$

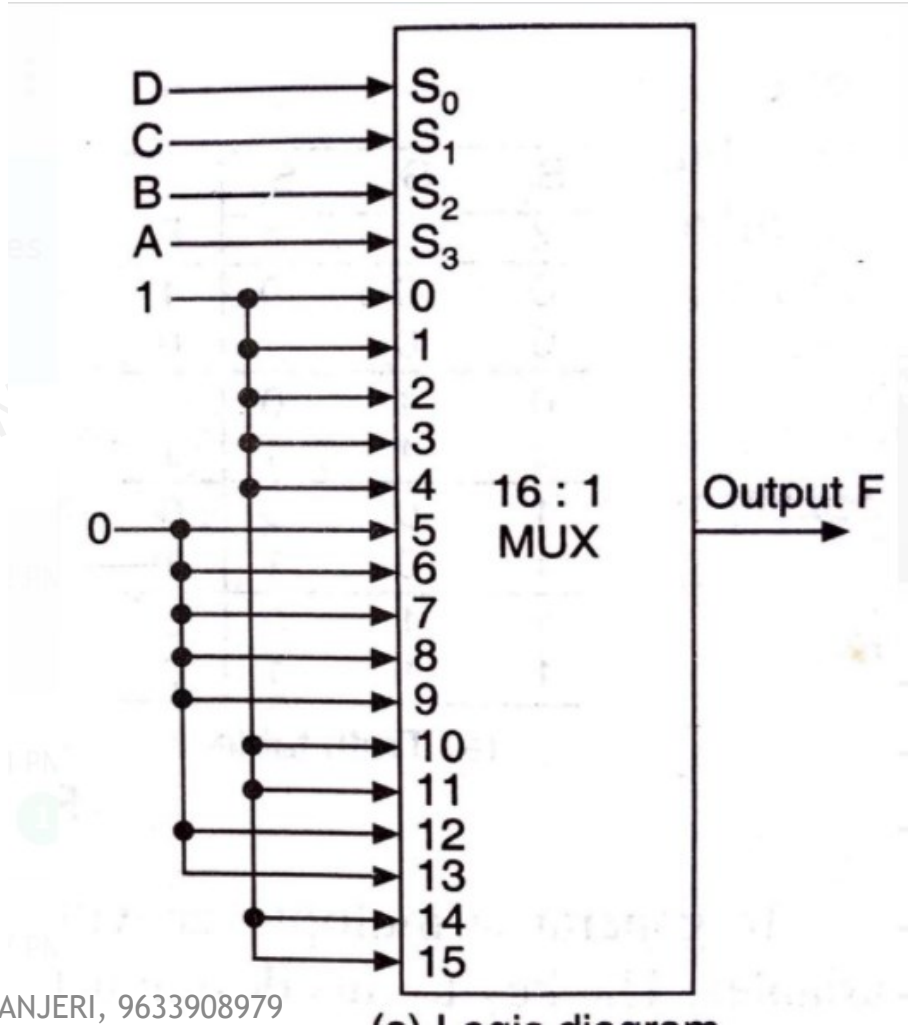
Minterm	S2	S1	C	F	
	A	B			
0	0	0	0	0	F=C
1	0	0	1	1	
2	0	1	0	1	F=C'
3	0	1	1	0	
4	1	0	0	1	F=C'
5	1	0	1	0	
6	1	1	0	0	F=C
7	1	1	1	1	

Problem 3a:

Use a multiplexer having Four data select the input to implement the logic for the function given below. Also realize the same using 16:1 Mux and 8:1 Mux

$f = \sum m(0,1,2,3,4,10,11,14,15)$

Minter	S3 A	S2 B	S1 C	S0 D	F
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

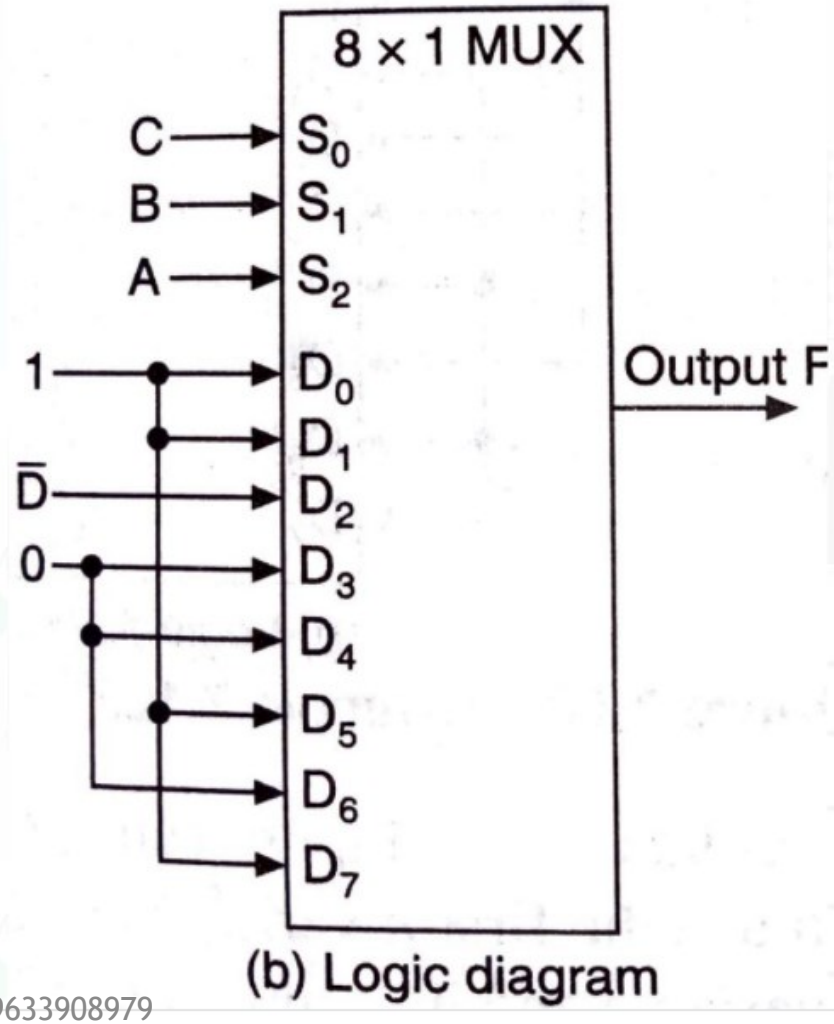


Problem 3b:

Use a multiplexer having three data select the input to implement the logic for the function given below. Also realize the same using 8:1 Mux

$f = \sum m(0,1,2,3,4,10,11,14,15)$

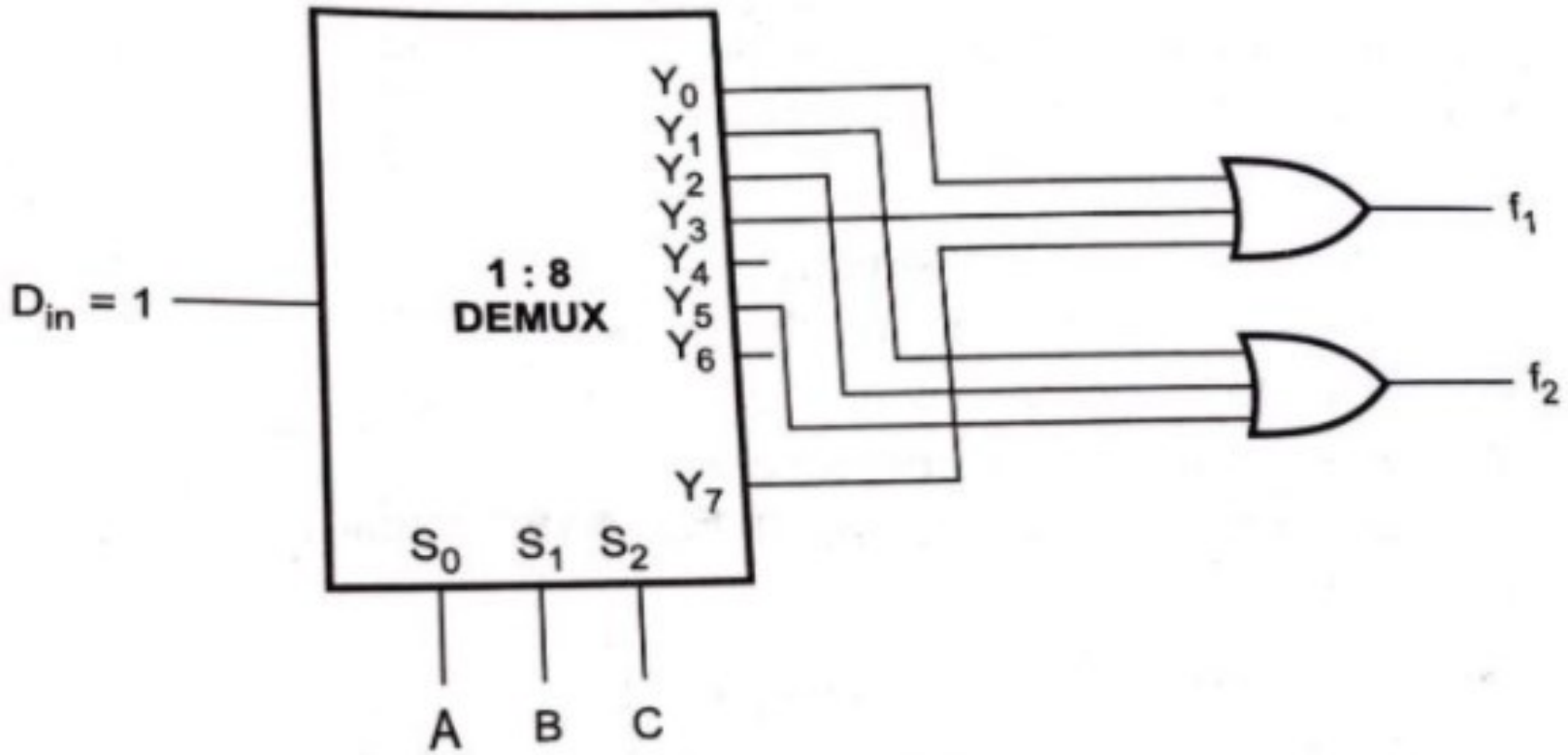
Minterm	S3	S2	S1	D	F	
	x	x	y			
0	0	0	0	0	1	
1	0	0	0	1	1	F=1
2	0	0	1	0	1	
3	0	0	1	1	1	F=1
4	0	1	0	0	1	
5	0	1	0	1	0	F=D'
6	0	1	1	0	0	
7	0	1	1	1	0	F=0
8	1	0	0	0	0	
9	1	0	0	1	0	F=0
10	1	0	1	0	1	
11	1	0	1	1	1	F=1
12	1	1	0	0	0	
13	1	1	0	1	0	F=0
14	1	1	1	0	1	
15	1	1	1	1	1	F=1



(b) Logic diagram

Problem 4

Use a demultiplexer Implement the functions f1= $\sum m(0,3,7)$ f2= $\sum m(1,2,5)$



Problem 5 (Home Work)

Use Multiplexer implement the logic functions

F=$A \oplus B \oplus C$

Inputs			outputs
W	X	Y	Q = A⊕B⊕C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Problem 6 (Home Work)

Use a demultiplexer Implement the functions f1= $\sum m(1,5,7)$ f2= $\sum m(3,6,7)$

Problem 7 (Home Work)

Implement the functions $f(w,x,y,z)=\sum m(1,4,6,7,8,9,10,11,15)$ Using 8 to 1 Mux

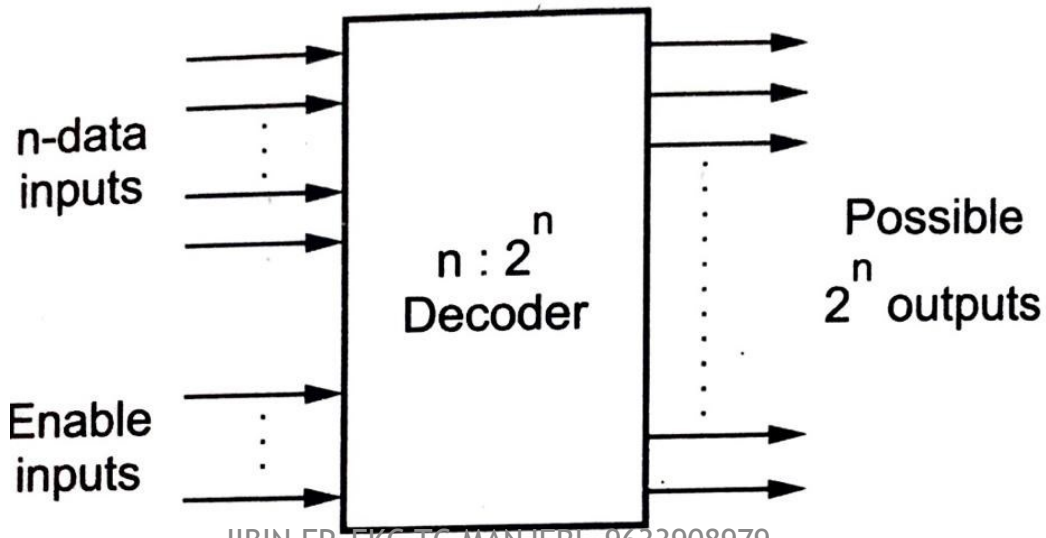
MODULE 3
(TOPIC-6)

DECODER AND ENCODER

https://youtu.be/Zxd-imLGIFc

DECODER

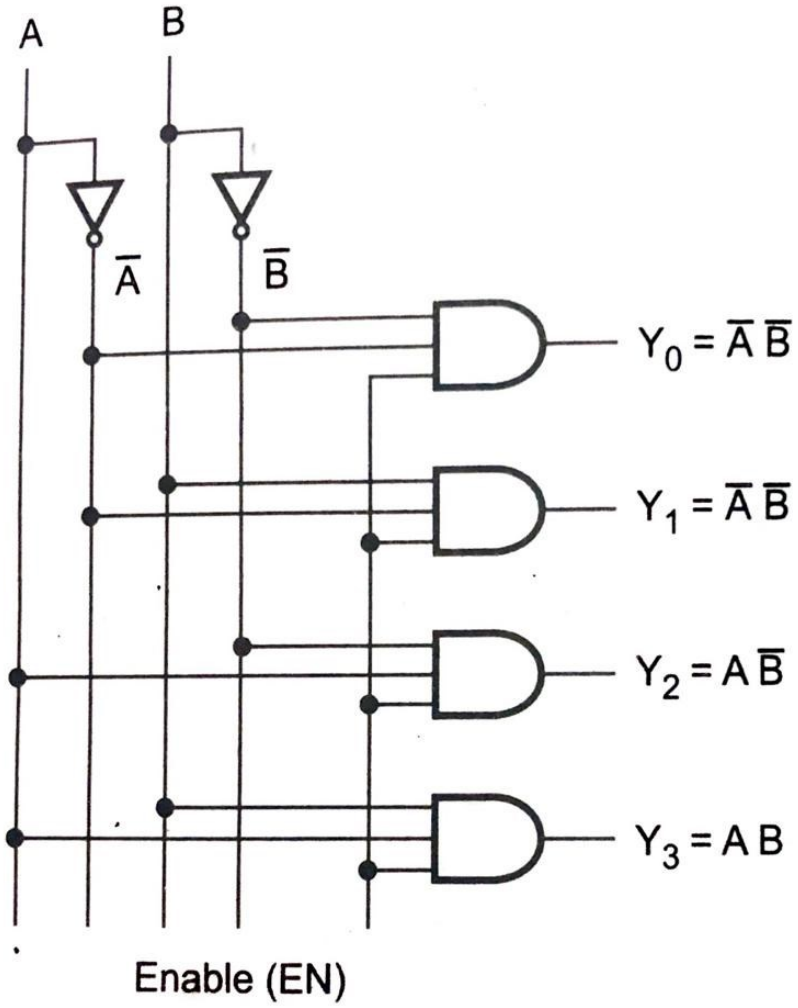
- Decoder is a Multiple Input , Multiple Output logic circuit which converts coded inputs to coded outputs, where the input and output are different
- Decoder consist of n input and 2^n Maximum Possible Outputs



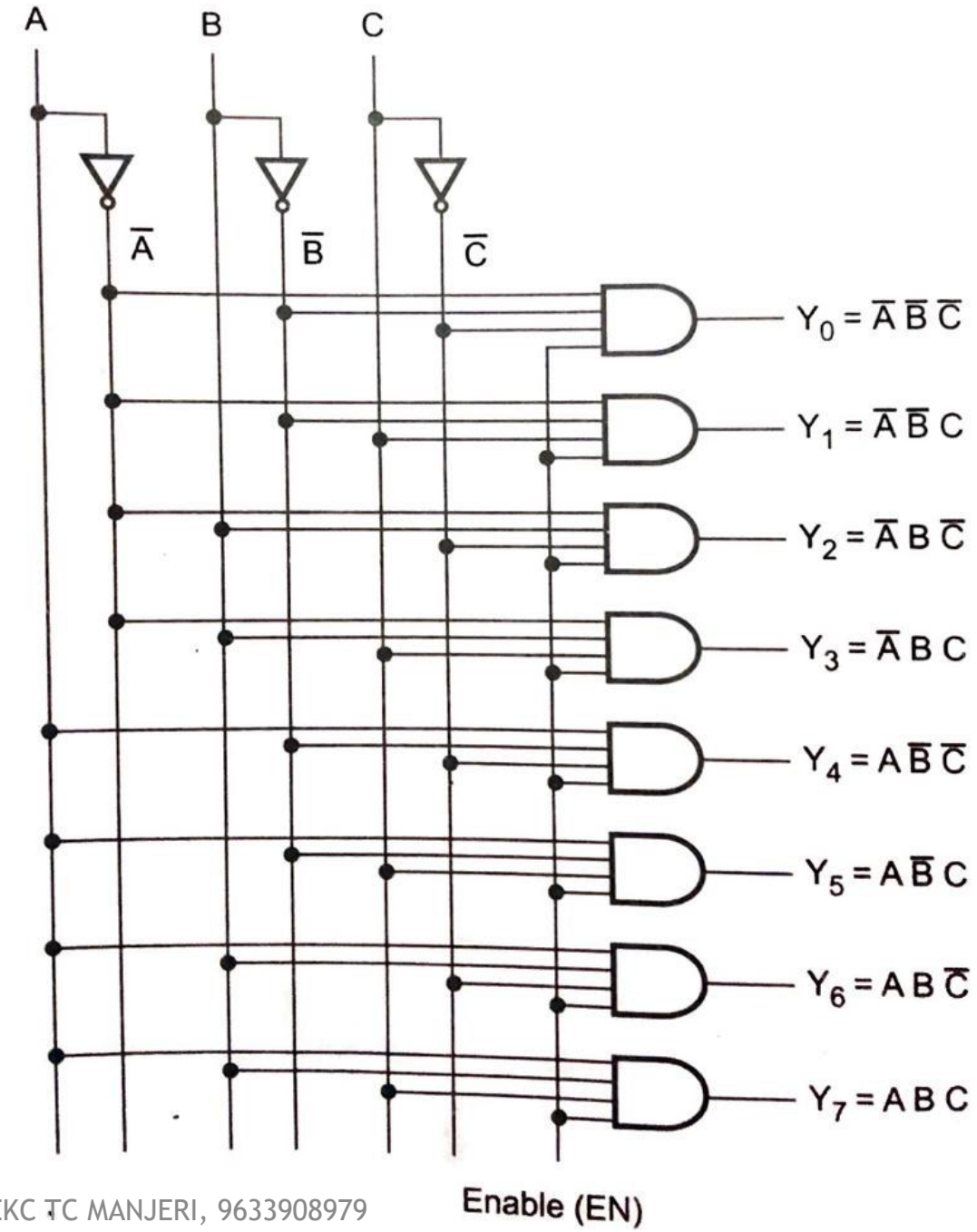
Examples

Binary Decoder (2 to 4 Decoder)

Inputs			Outputs			
ENABLE	A	B	Y3	Y2	Y1	Y0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



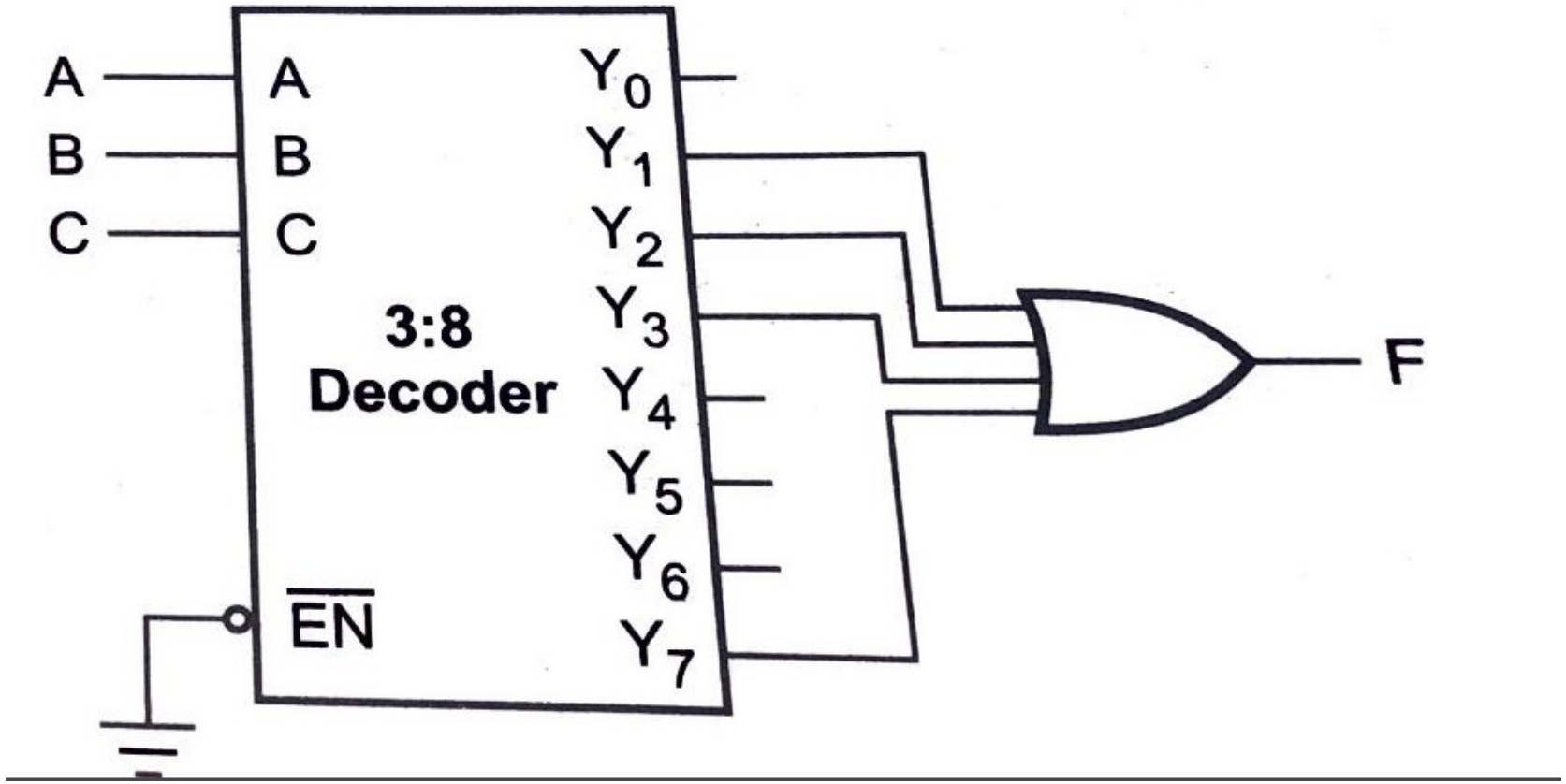
3 to 8 Decoder



3 to 8 Decoder

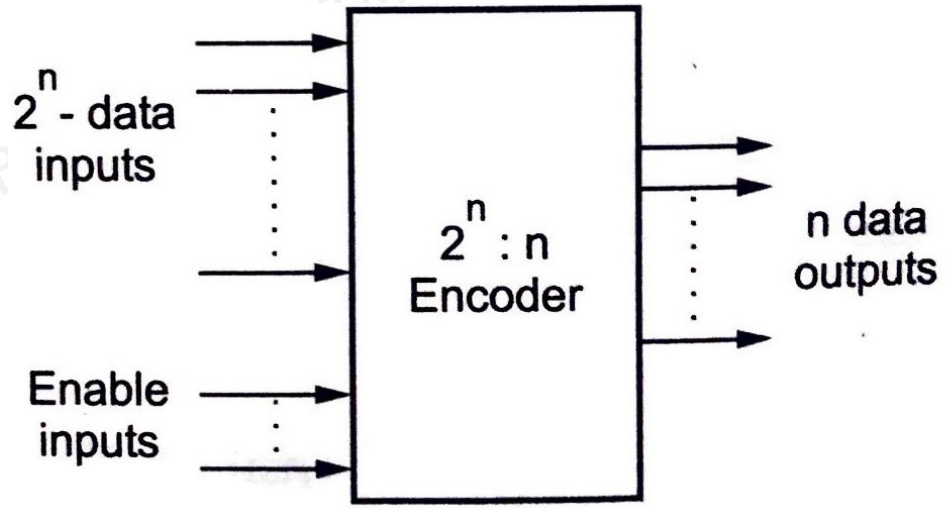
Inputs								Outputs			
ENABLE	A	B	C	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Implement the Boolean function $f = \sum m(1,2,3,7)$

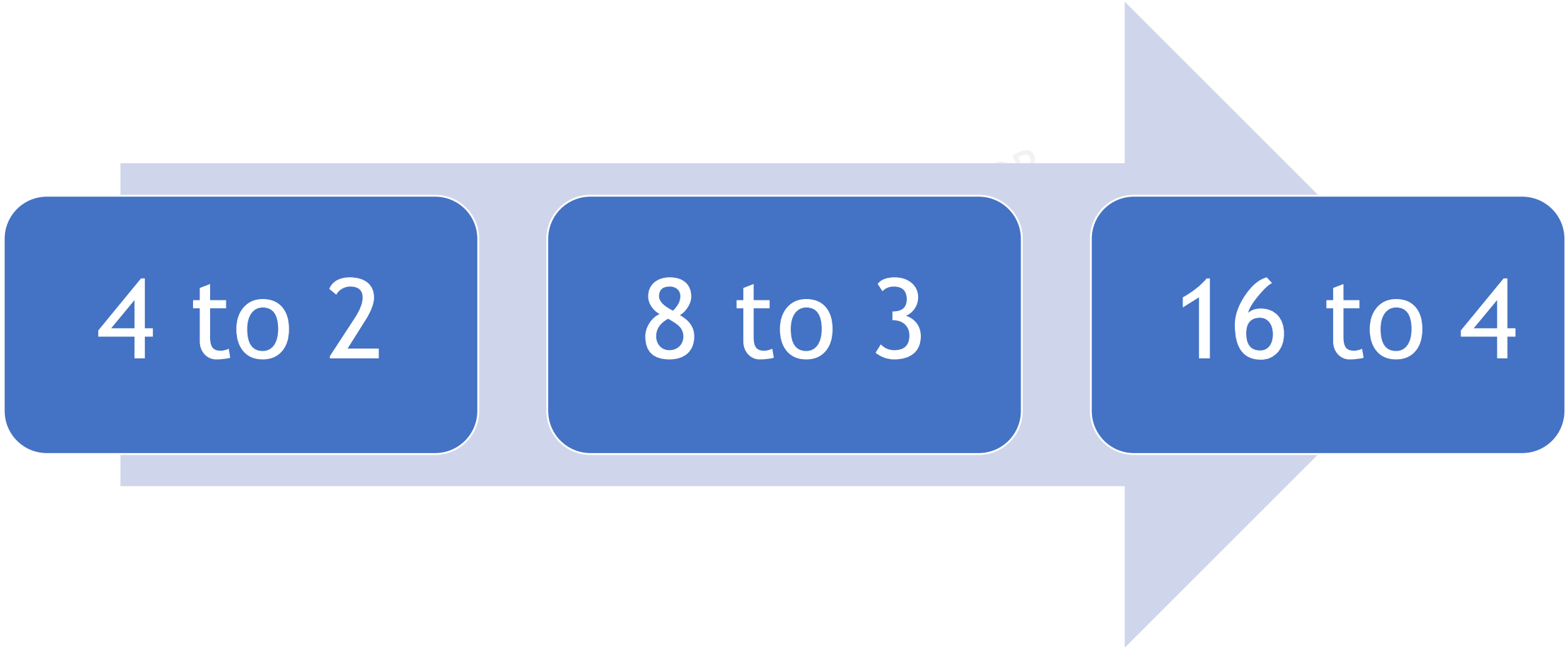


ENCODER

- Encoder is a digital circuit which perform inverse operation of Decoder
- Decoder is a Multiple Input , Multiple Output logic circuit which converts coded inputs to coded outputs, where the input and output are different
- Decoder consist of n Outputs and 2^n Maximum Possible Inputs



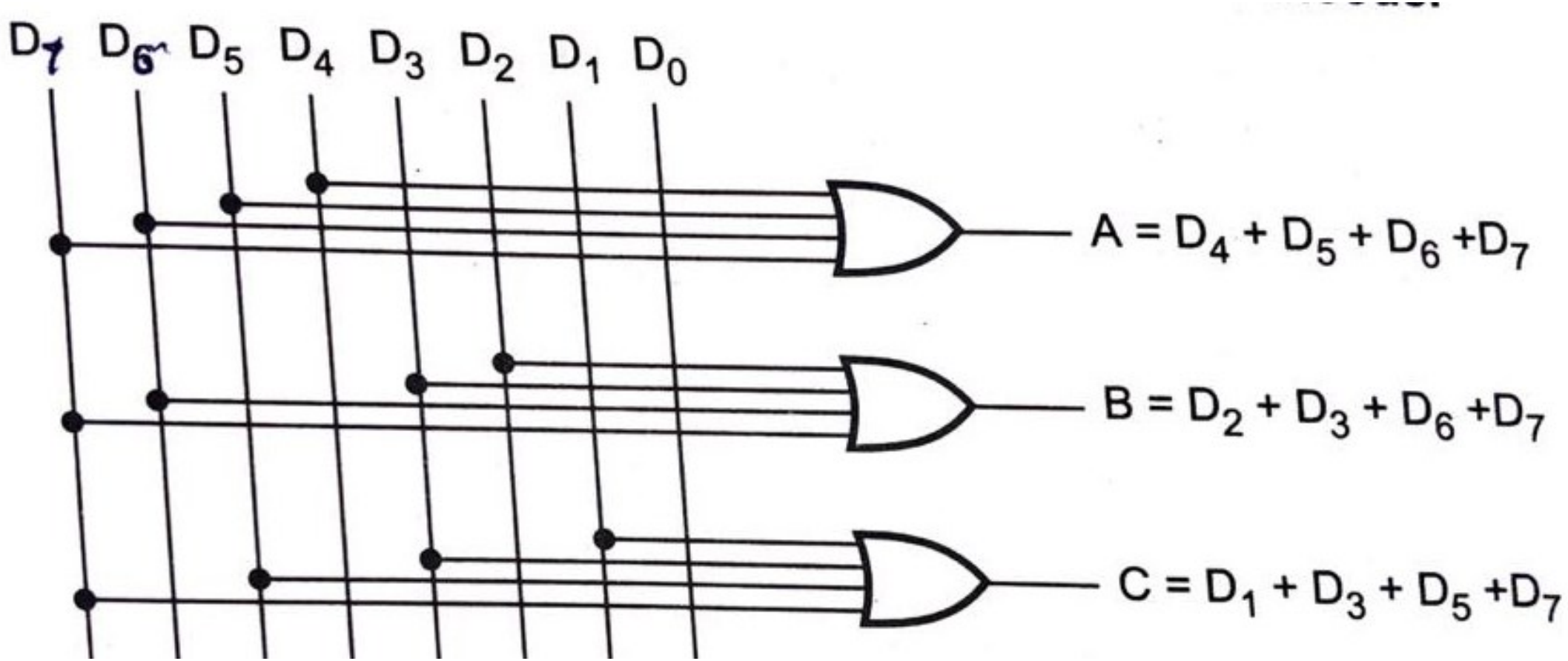
Examples



8 TO 3 ENCODER (OCTAL TO BINARY)

Inputs								Outputs		
D0	D1	D2	D3	D4	D5	D6	D7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

8 TO 3 ENCODER (OCTAL TO BINARY)



PRIORITY ENCODER

- A Priority encoder is one of the types of encoders in which an ordering is imposed to the inputs that means compared with the standard encoder, it includes the priority function.
- However, this priority is based on the relative magnitudes of the inputs. Hence, the input with larger magnitude is the one that is encoded first.
- Priority encoders can select the inputs with highest priority in many practical applications. This process of selection is called arbitration.
- One of the most common examples of arbitration is that , there are numerous input devices in computer system and several of which devices attempt to supply the data to the computer simultaneously. In those cases, a priority encoder enables the input device having the highest priority among those devices trying to access the computer at the same time.

PRIORITY ENCODER

This Priority encoder consists of 4 inputs and three outputs.
Although an encoder has 2^n inputs and n outputs, it has a third output 'V' which is a valid bit indicator and is set to one when one or more inputs are active or equal to 1.

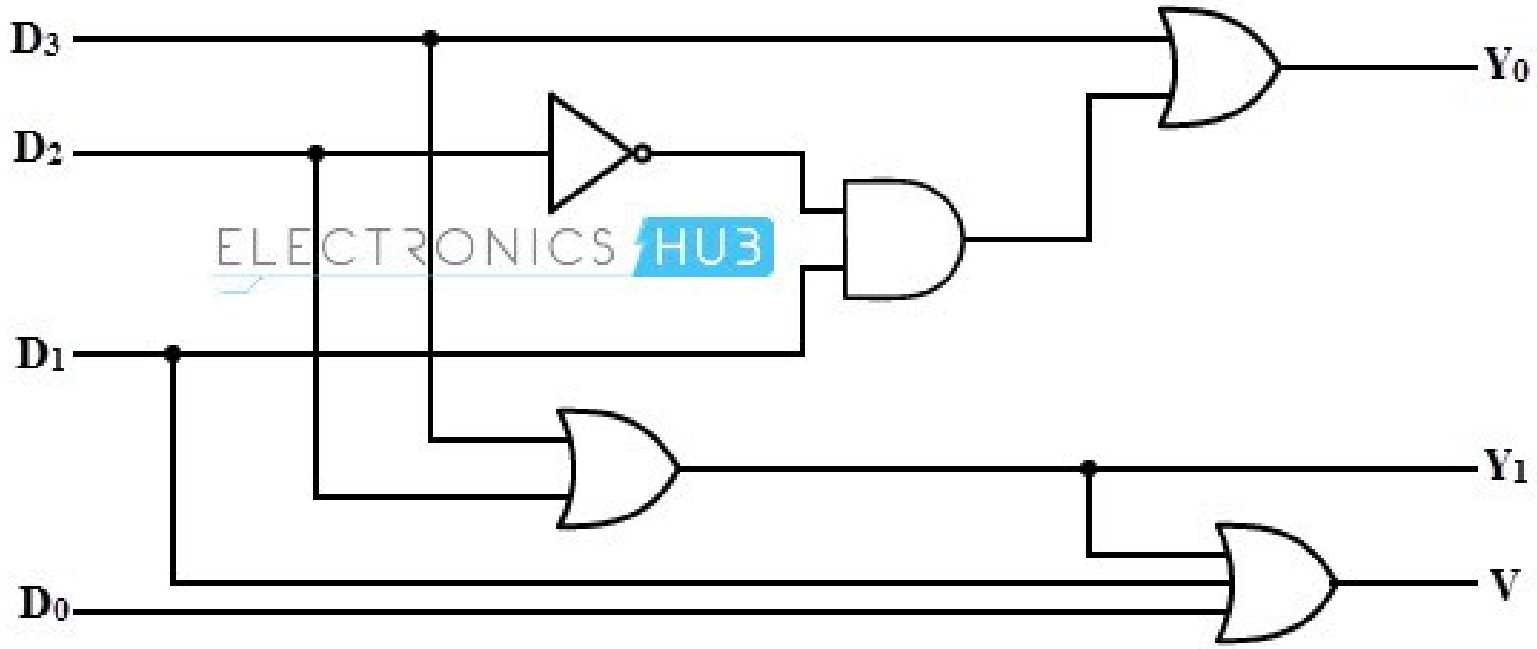
Inputs				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	×	×	0
1	0	0	0	0	0	1
×	1	0	0	0	1	1
×	×	1	0	1	0	1
×	×	×	1	1	1	1

$$Y1 = D3 + \overline{D3}\,D2 = D3 + D2$$

$$Y0 = D3 + \overline{D3}\,\overline{D2}\,D1 = D3 + \overline{D2}\,D1$$

$$V = D3 + D2 + D1 + D0$$

PRIORITY ENCODER



MODULE 3
(TOPIC-7)

CODE CONVERTERS

https://youtu.be/cjAMfV-HubQ

CODE CONVERTERS

- There is a wide variety of binary codes used in digital system
- The code converter convert one code to another
- **Steps**

1. Truth Table showing relationship between input and Output
2. For Each output code determine the simplified boolean expression using KMAP
3. Realize the code converter using Logic Gates

BINARY TO BCD CONVE

Binary code				BCD code				
D	C	B	A	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0	1

For B_0

DC \ BA	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

$B_0 = A$

For B_1

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	0	0

$B_1 = DC\overline{B} + \overline{D}B$

For B_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

$B_2 = \overline{D}C + CB$

For B_3

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

$B_3 = D\overline{C}\,\overline{B}$

For B_4

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$B_4 = DC + DB$

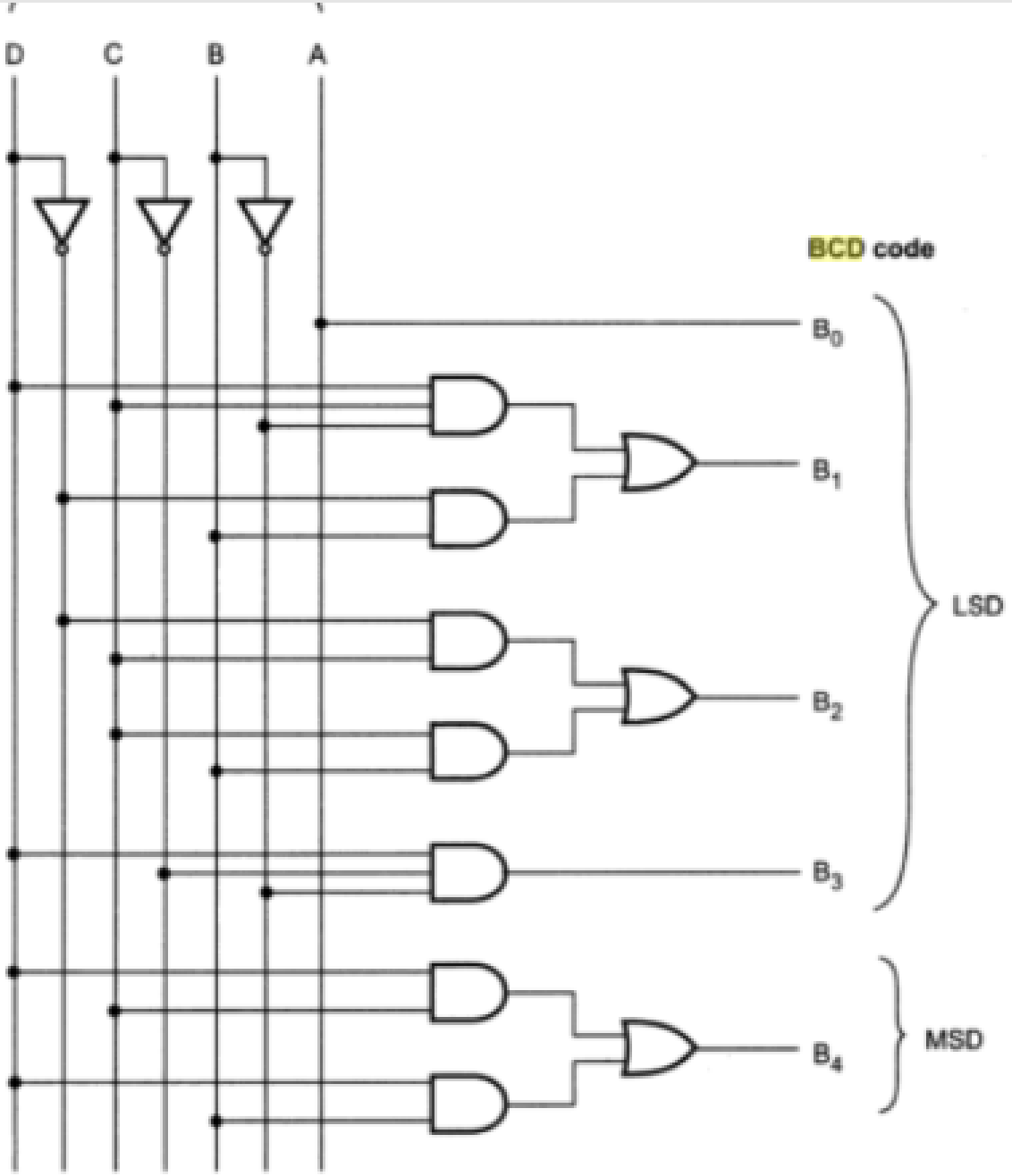


Fig. 3.43 Binary to BCD converter

BCD TO EXCESS 3

Decimal	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

For E_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$E_3 = B_3 + B_2(B_0 + B_1)$

For E_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$E_2 = B_2\overline{B}_1\overline{B}_0 + \overline{B}_2(B_0 + B_1)$

For E_1

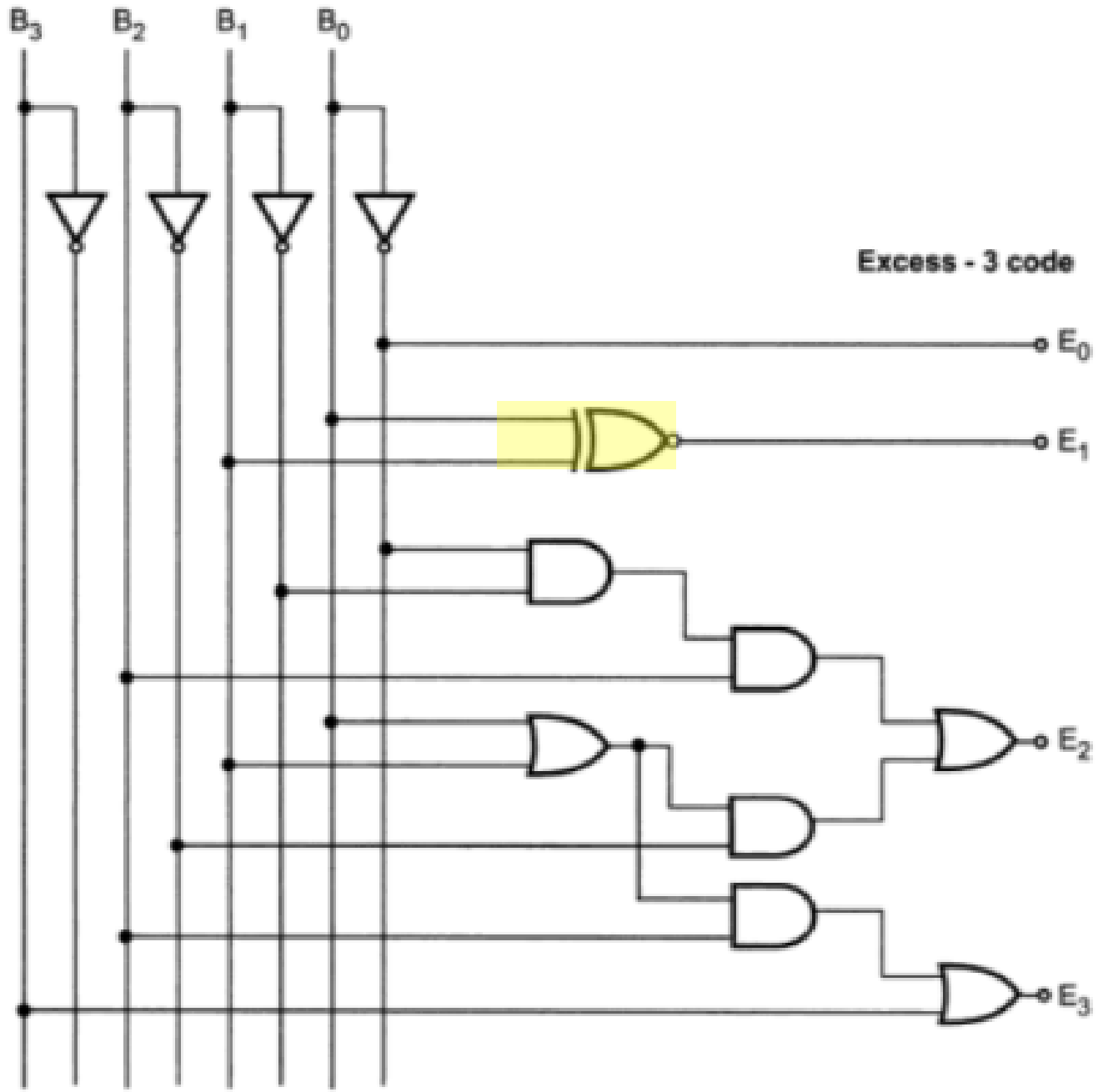
B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$E_1 = \overline{B}_1\overline{B}_0 + B_1B_0$
$= B_1 \odot B_0$

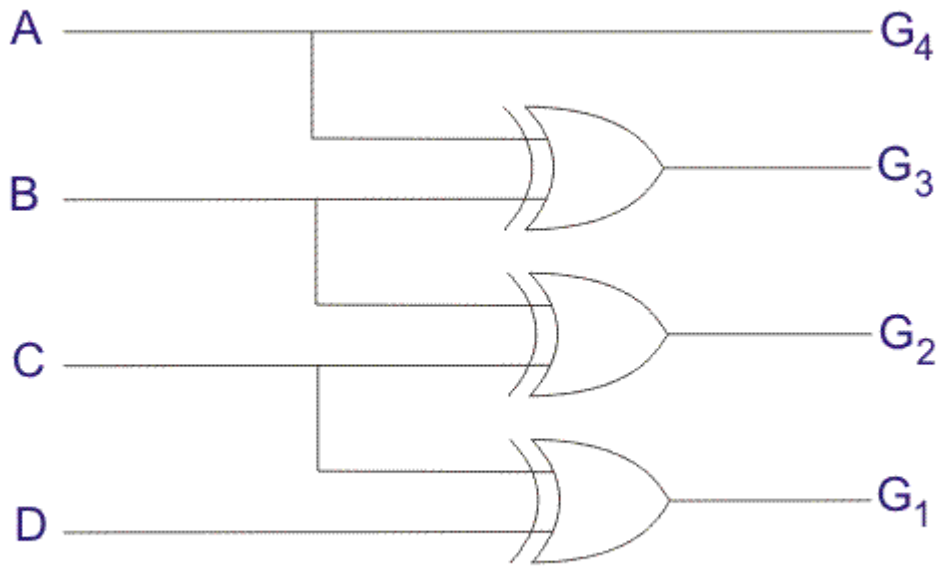
For E_0

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_0 = \overline{B}_0$


B3
B2
B1
B0
Excess - 3 code
E0
E1
E2
E3

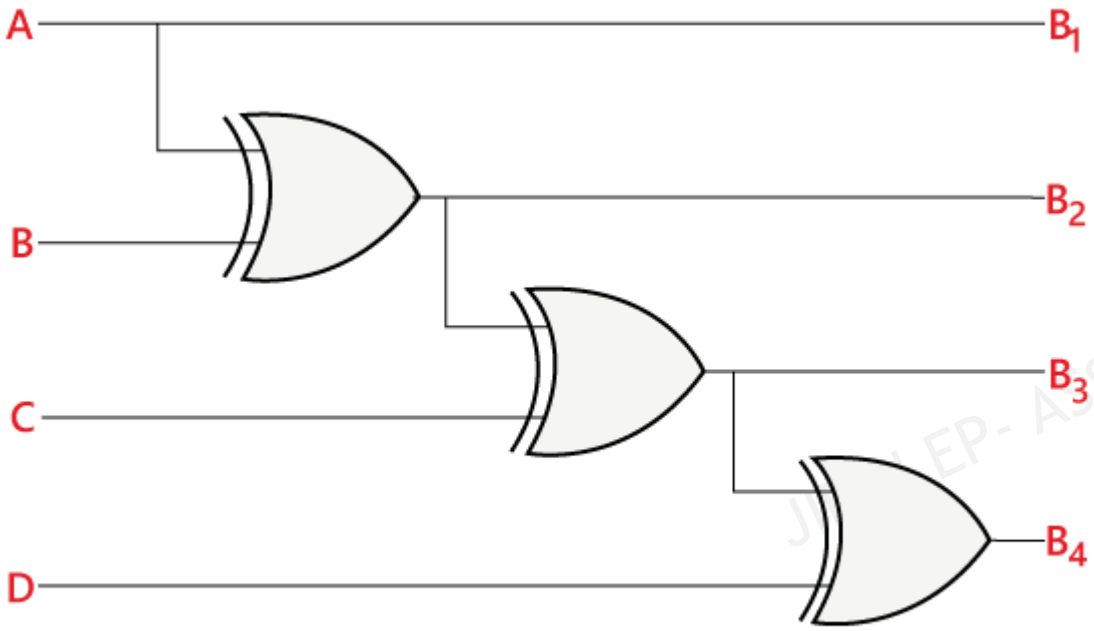
BINARY TO GRAY CODE



Logic Circuit for Binary to Gray Code Converter

Decimal	Binary code				Gray code			
	D	C	B	A	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

GRAY TO BINARY CODE



Logic Circuit for Gray to Binary Code Converter

Gray code				Binary code			
G_3	G_2	G_1	G_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

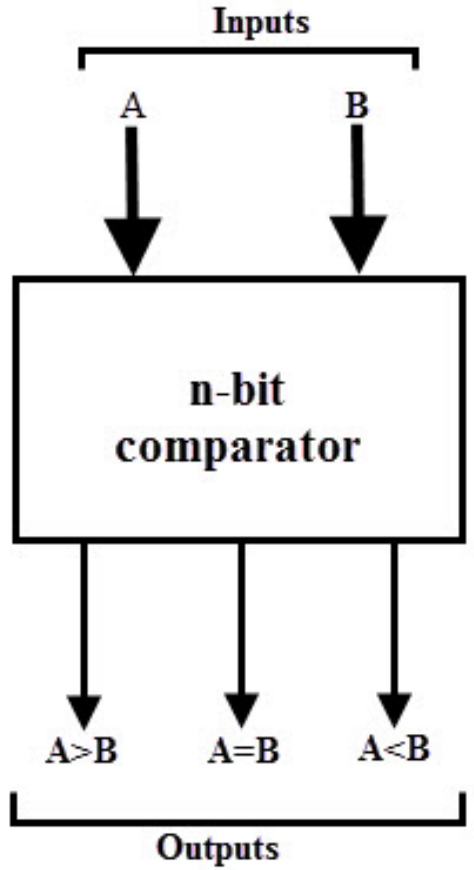
MODULE 3
(TOPIC-8)

MAGNITUDE COMPARATORS

https://youtu.be/7xHM5HeGCHo

MAGNITUDE COMPARATOR

- A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers (consider A and B) and determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.
- Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B. The below figure shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.

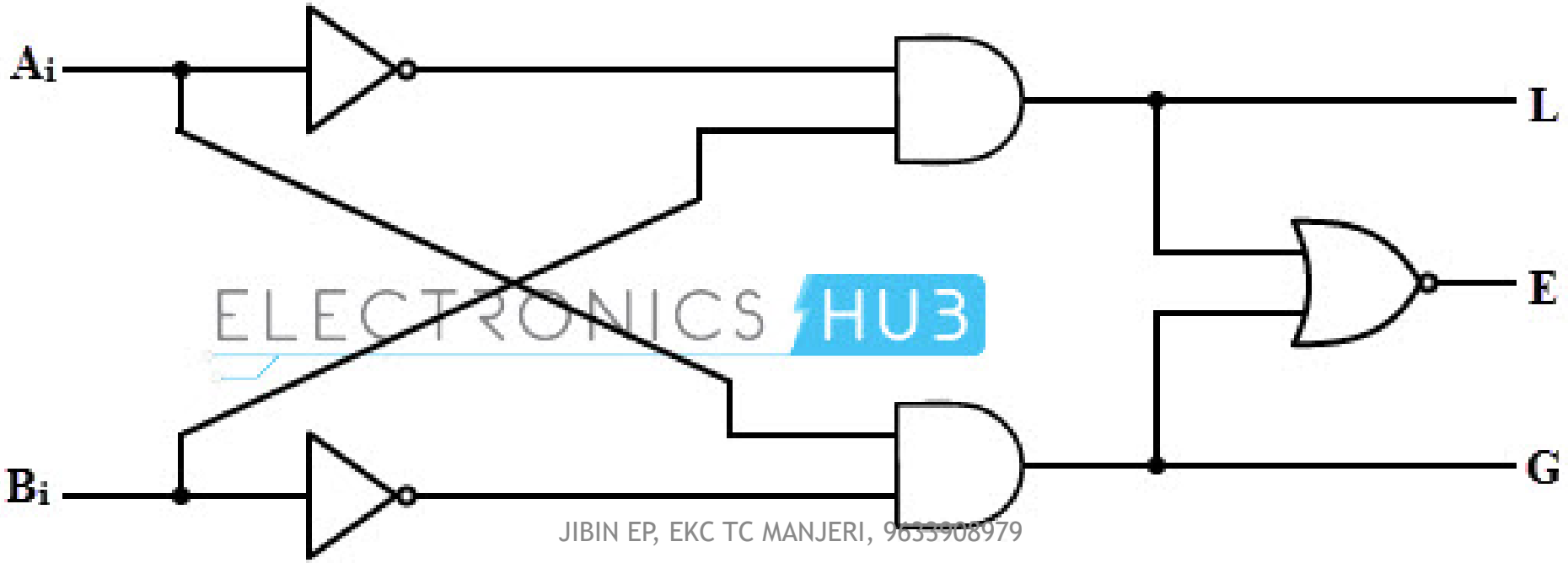


Single Bit Magnitude Comparator

- comparator used to compare two bits, i.e., two numbers each of single bit is called a single bit comparator. It consists of two inputs for allowing two single bit numbers and three outputs to generate less than, equal and greater than comparison outputs.
- The figure below shows the block diagram of a single bit magnitude comparator. This comparator compares the two bits and produces one of the 3 outputs as L (A<B), E (A=B) and G (A>B).

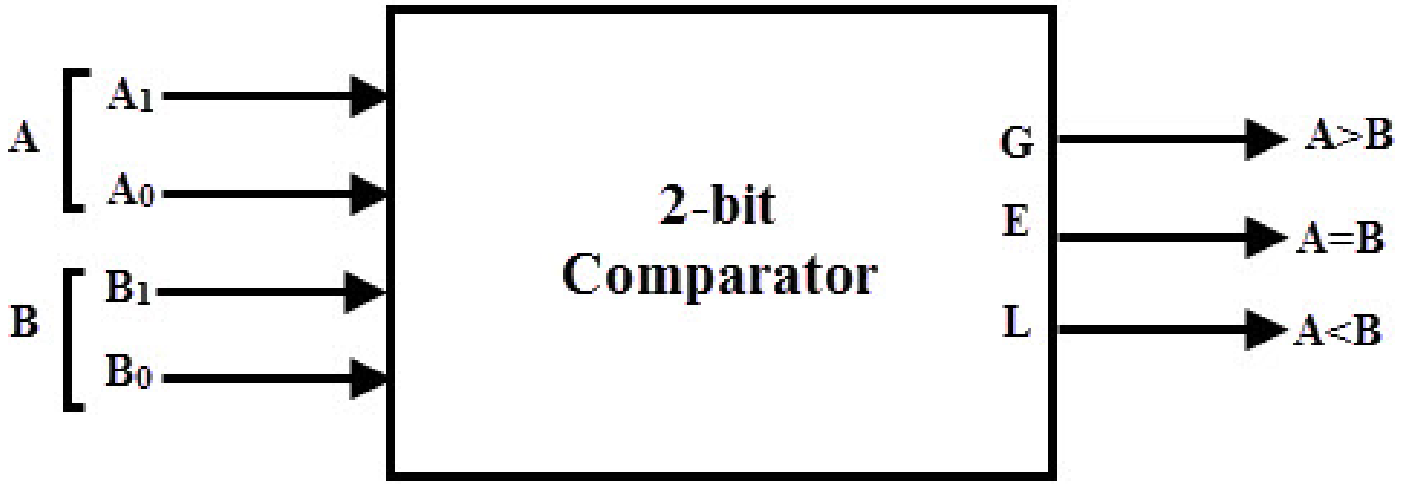
Truth Table

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



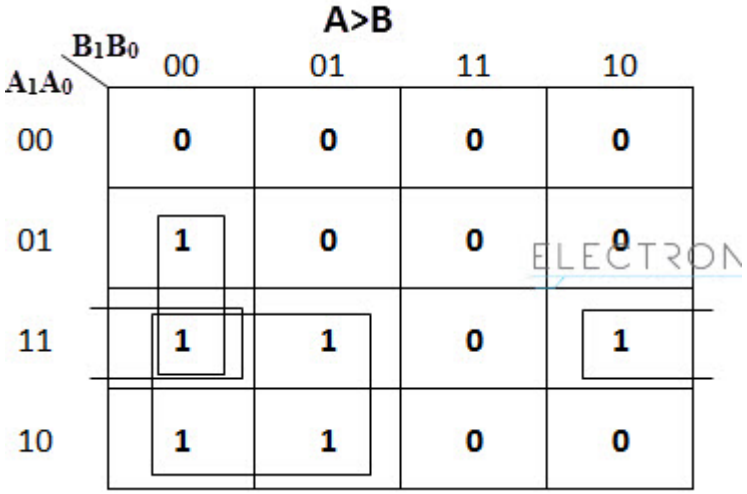
Two Bit Magnitude Comparator

- A 2-bit comparator compares two binary numbers, each of two bits and produces their relation such as one number is equal or greater than or less than the other. The figure below shows the block diagram of a two-bit comparator which has four inputs and three outputs.
- The first number A is designated as A = A1A0 and the second number is designated as B = B1B0. This comparator produces three outputs as = 1 if A<B).

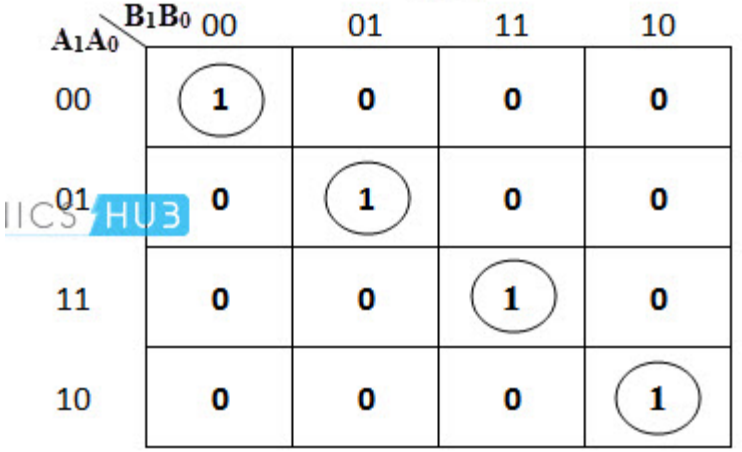


Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

A>B



A=B



$$A>B:\ G = A0\,\overline{B1}\ \ \overline{B0} + A1\,\overline{B1} + A1\ \ A0\,\overline{B0}$$

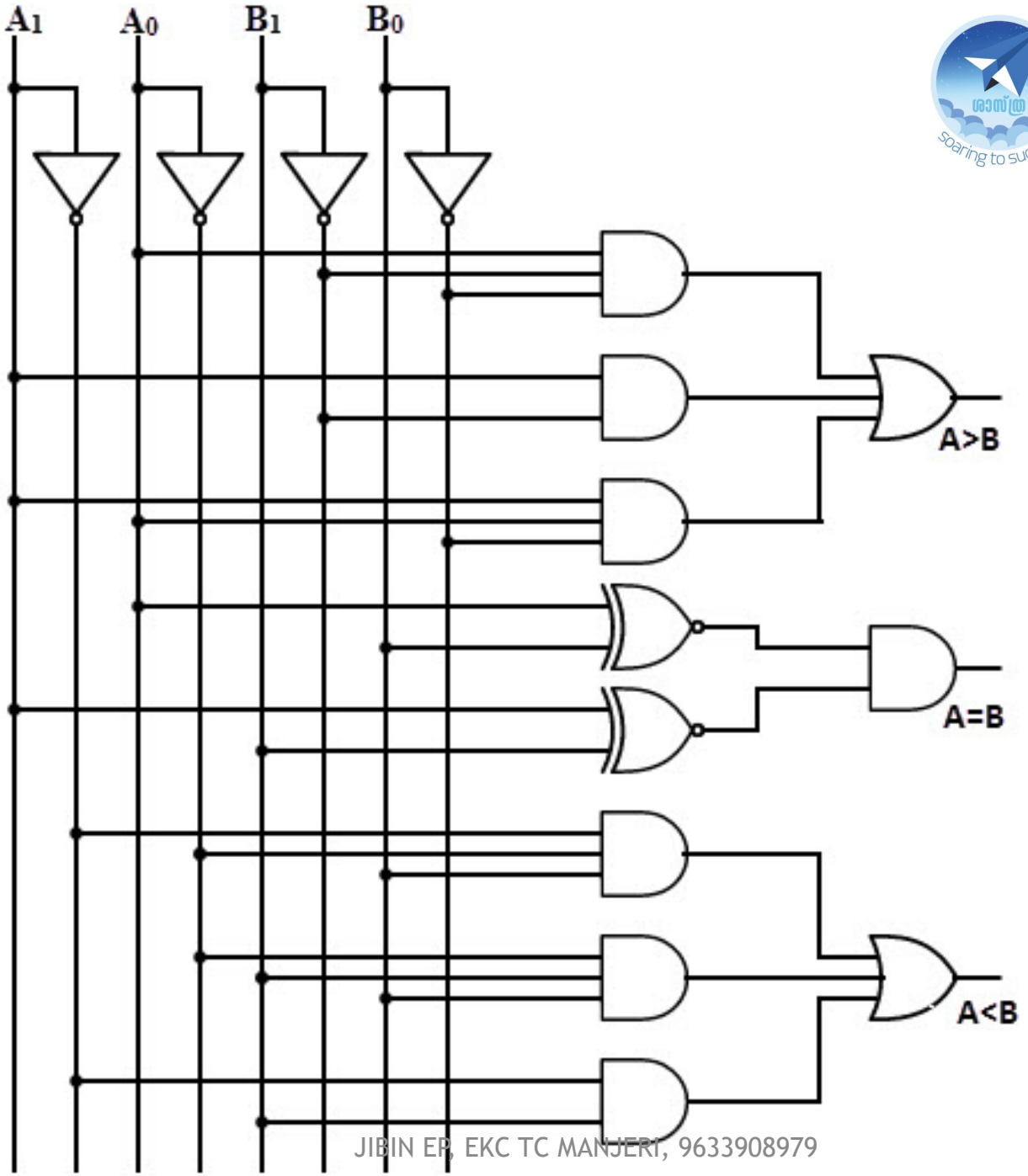
$$A = B:\ E = \overline{A1}\ \ \overline{A0}\,\overline{B1}\ \ \overline{B0} + \overline{A1}\,A0\ \ \overline{B1}\,B0 + A1\ \ A0\,B1\,B0 + A1\,\overline{A0}\,B1\,\overline{B0}$$

$$= \overline{A1}\ \ \overline{B1}\,(\overline{A0}\ \ \overline{B0} + A0\,B0) + A1\,B1\,(A0\,B0 + \overline{A0}\ \ \overline{B0}\ )$$

$$= (A0\,B0 + \overline{A0}\ \ \overline{B0})\,(A1\,B1 + \ \overline{A1}\ \ \overline{B1})$$

$$= (A0\ \text{Ex-NOR}\ B0)\ (A1\ \text{Ex-NOR}\ B1)$$

$$A < B:\ L = \overline{A1}\ \ B1 + \overline{A0}\,B1\,B0 + \overline{A1}\ \ \overline{A0}\,B0$$


A1
A0
B1
B0
A>B
A=B
A<B

Application of Comparators

- These are used in the address decoding circuitry in computers and microprocessor based devices to select a specific input/output device for the storage of data.
- These are used in control applications in which the binary numbers representing physical variables such as temperature, position, etc. are compared with a reference value. Then the outputs from the comparator are used to drive the actuators so as to make the physical variables closest to the set or reference value.
- Process controllers
- Servo-motor control

MODULE 3
(TOPIC-9)

PARITY GENERATOR AND CHECKER

https://youtu.be/7xHM5HeGCHo

What is Parity ?

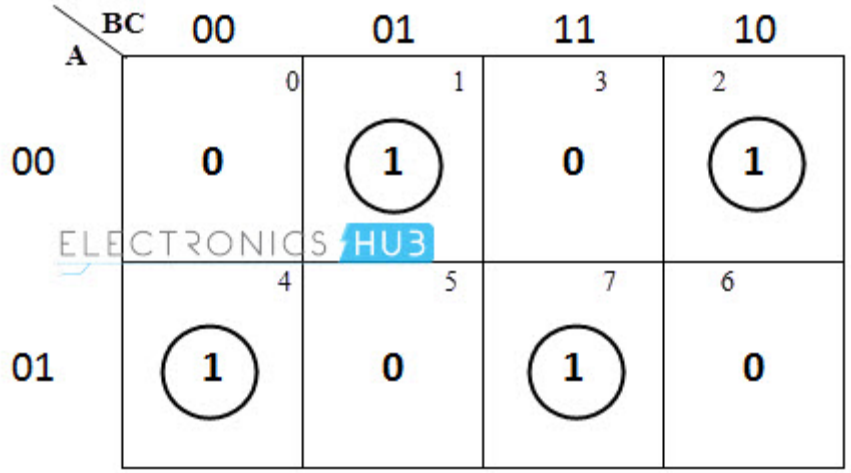
- The parity generating technique is one of the most widely used error detection techniques for the data transmission. In digital systems, when binary data is transmitted and processed , data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.
- Hence, **parity bit** is added to the word containing data in order to make number of 1s either even or odd.Thus it is used to detect errors , during the transmission of binary data .The message containing the data bits along with parity bit is transmitted from transmitter node to receiver node.
- At the receiving end, the number of 1s in the message is counted and if it doesn't match with the transmitted one, then it means there is an error in the data.

EVEN PARITY GENERATOR

- Let us assume that a 3-bit message is to be transmitted with an even parity bit. Let the three inputs A, B and C are applied to the circuits and output bit is the parity bit P. The total number of 1s must be even, to generate the even parity bit P.
- The figure below shows the truth table of even parity generator in which 1 is placed as parity bit in order to make all 1s as even when the number of 1s in the truth table is odd.

EVEN PARITY GENERATOR

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

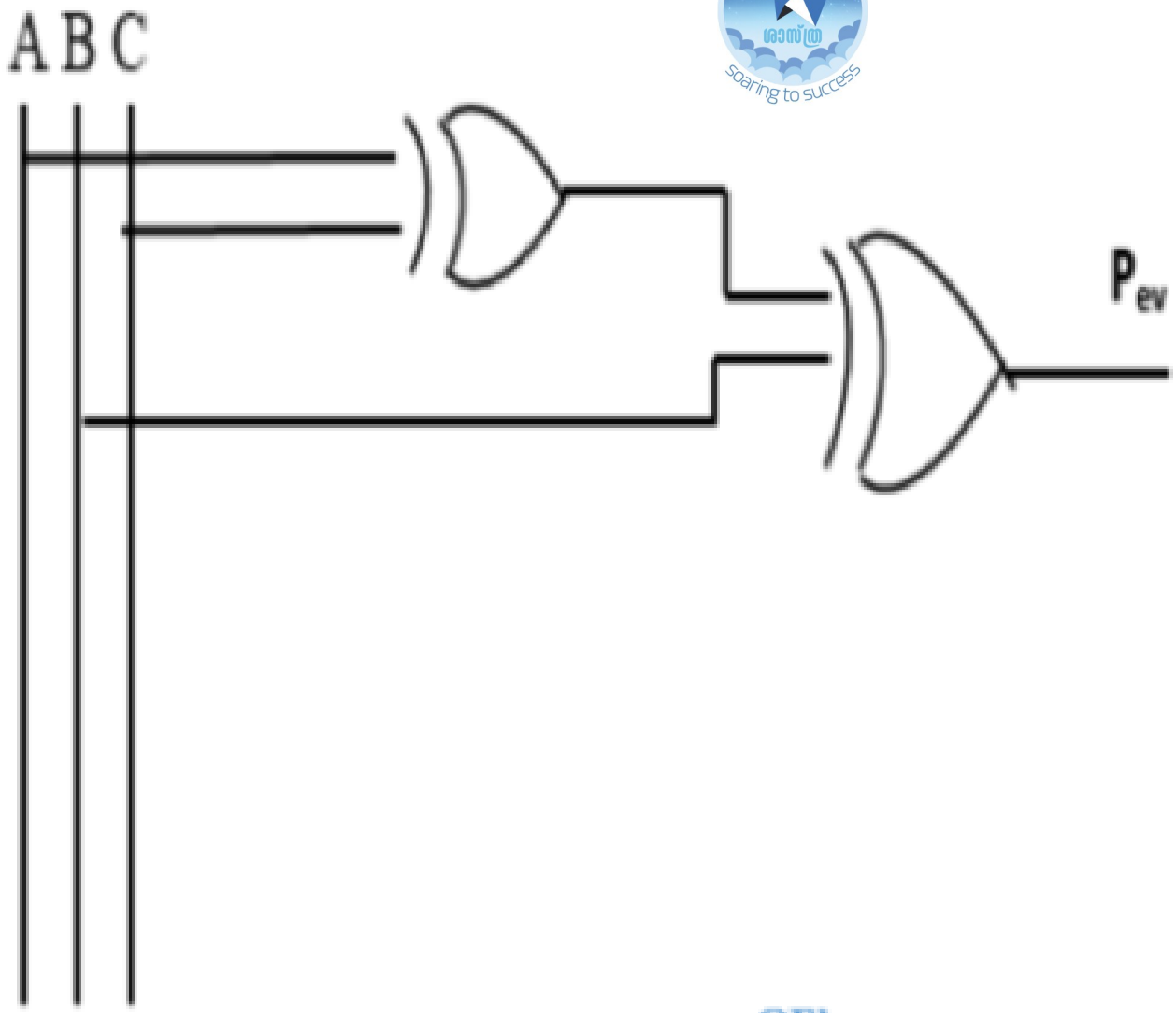


$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

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

$$= \bar{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

$$P = A \oplus B \oplus C$$

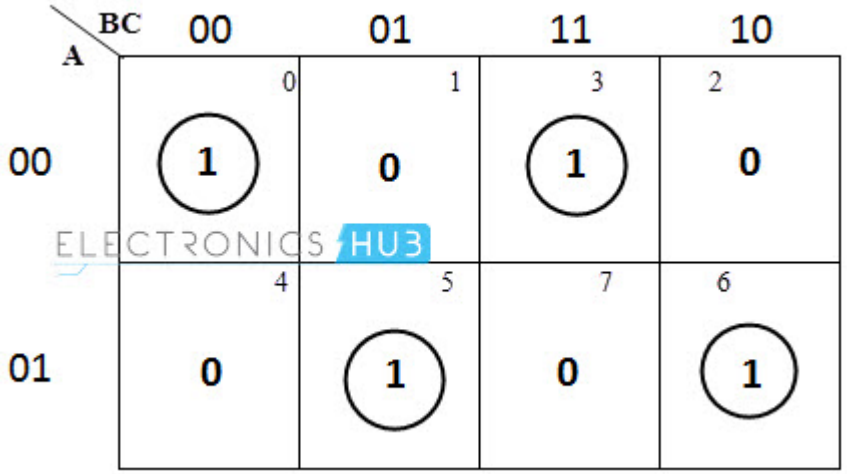


Odd Parity Generator

- Let us consider that the 3-bit data is to be transmitted with an odd parity bit. The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity bit.
- In the given truth table below, 1 is placed in the parity bit in order to make the total number of bits odd when the total number of 1s in the truth table is even.

Odd Parity Generator

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



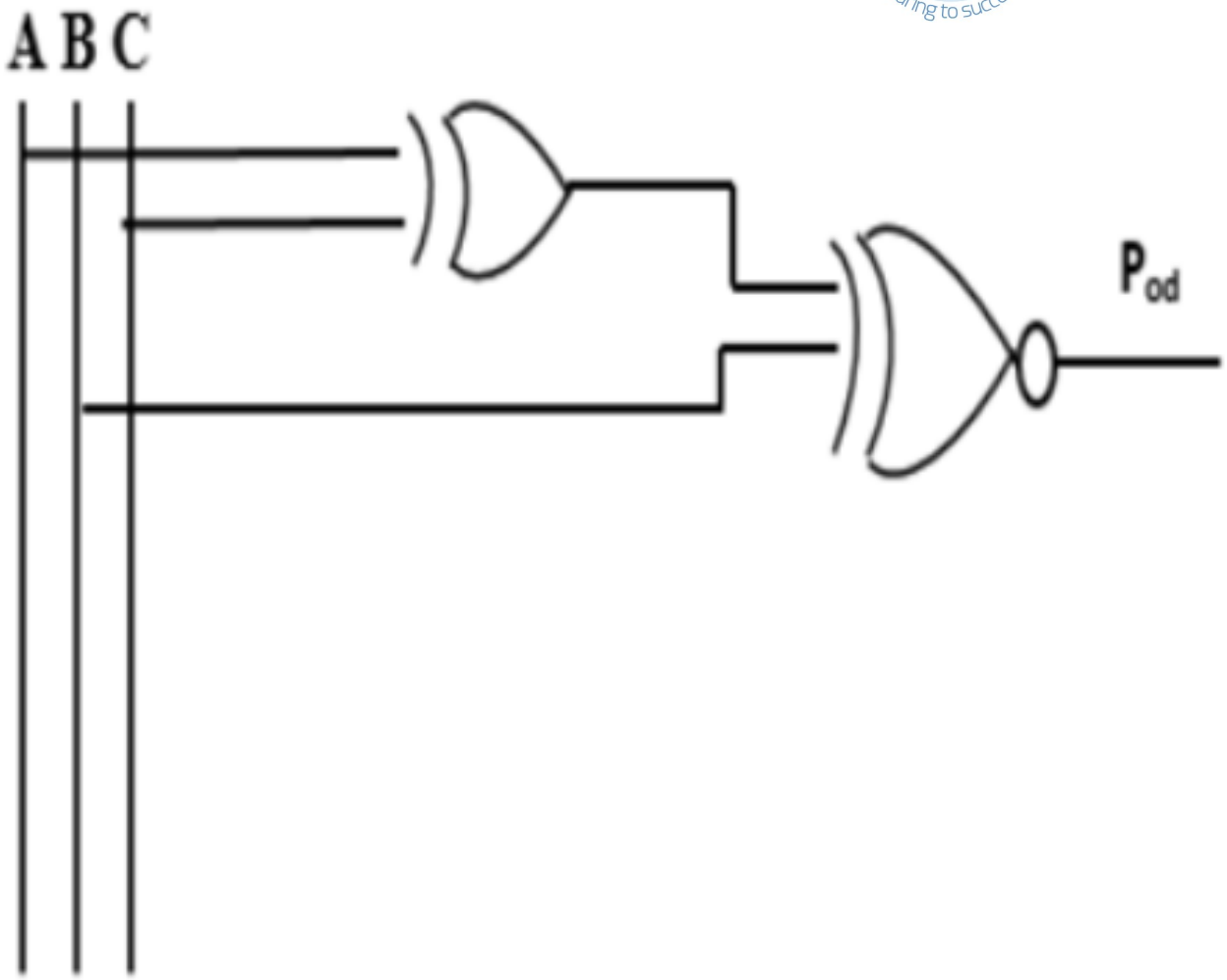
$$P_{od} = (\bar{A}\,\bar{B}\,\bar{C} + A\,\bar{B}\,C) + (\bar{A}\,B\,C + A\,B\,\bar{C})$$

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

$$= \bar{C}\,(A \oplus B) + C(A \oplus B)$$

$$= \bar{C}\,\bar{X} + C\,X = C \odot X = C \odot \overline{A \oplus B}$$

Odd Parity Generator



PARITY CHECKER

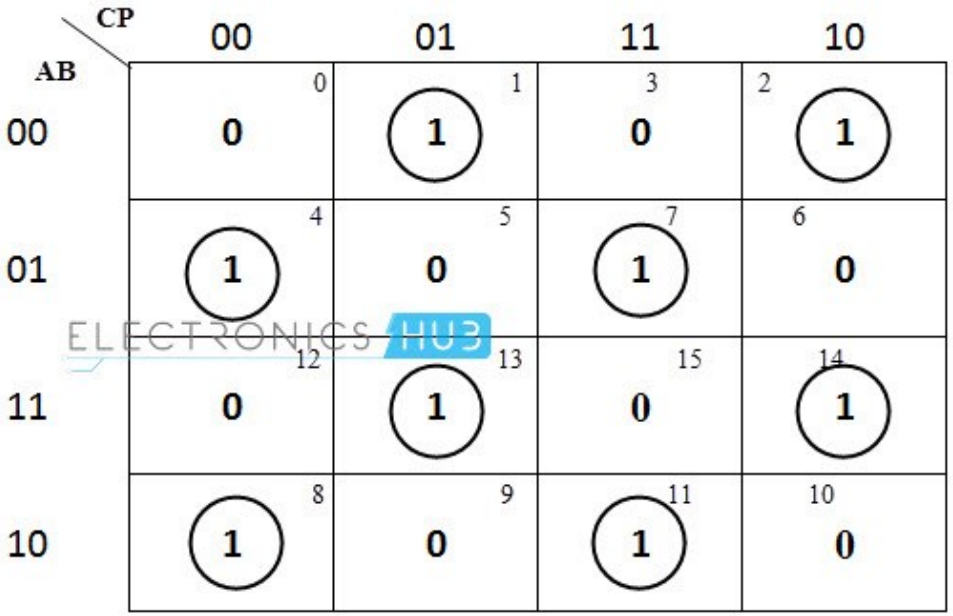
PARITY CHECK

- It is a logic circuit that checks for possible errors in the transmission. This circuit can be an even parity checker or odd parity checker depending on the type of parity generated at the transmission end. When this circuit is used as even parity checker, the number of input bits must always be even.
- When a parity error occurs, the ‘sum even’ output goes low and ‘sum odd’ output goes high. If this logic circuit is used as an odd parity checker, the number of input bits should be odd, but if an error occurs the ‘sum odd’ output goes low and ‘sum even’ output goes high

EVEN PARITY CHECK

- Consider that three input message along with even parity bit is generated at the transmitting end. These 4 bits are applied as input to the parity checker circuit which checks the possibility of error on the data. Since the data is transmitted with even parity, four bits received at circuit must have an even number of 1s.
- If any error occurs, the received message consists of odd number of 1s. The output of the parity checker is denoted by PEC (parity error check).
- The below table shows the truth table for the even parity checker in which PEC = 1 if the error occurs, i.e., the four bits received have odd number of 1s and PEC = 0 if no error occurs, i.e., if the 4-bit message has even number of 1s.

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



$$PEC = \bar{A}\,\bar{B}\,(\bar{C}\,D + C\,\bar{D}) + \bar{A}\,B\,(\bar{C}\,\bar{D} + C\,D) + A\,B\,(\bar{C}\,D + C\,\bar{D}) + A\,\bar{B}\,(\bar{C}\,\bar{D} + C\,D)$$

$$= \bar{A}\,\bar{B}\,(C \oplus D) + \bar{A}\,B\,(\overline{C \oplus D}) + A\,B\,(C \oplus D) + A\,\bar{B}\,(\overline{C \oplus D})$$

$$= (\bar{A}\,\bar{B} + A\,B)\,(C \oplus D) + (\bar{A}\,B + A\,\bar{B})\,(\overline{C \oplus D})$$

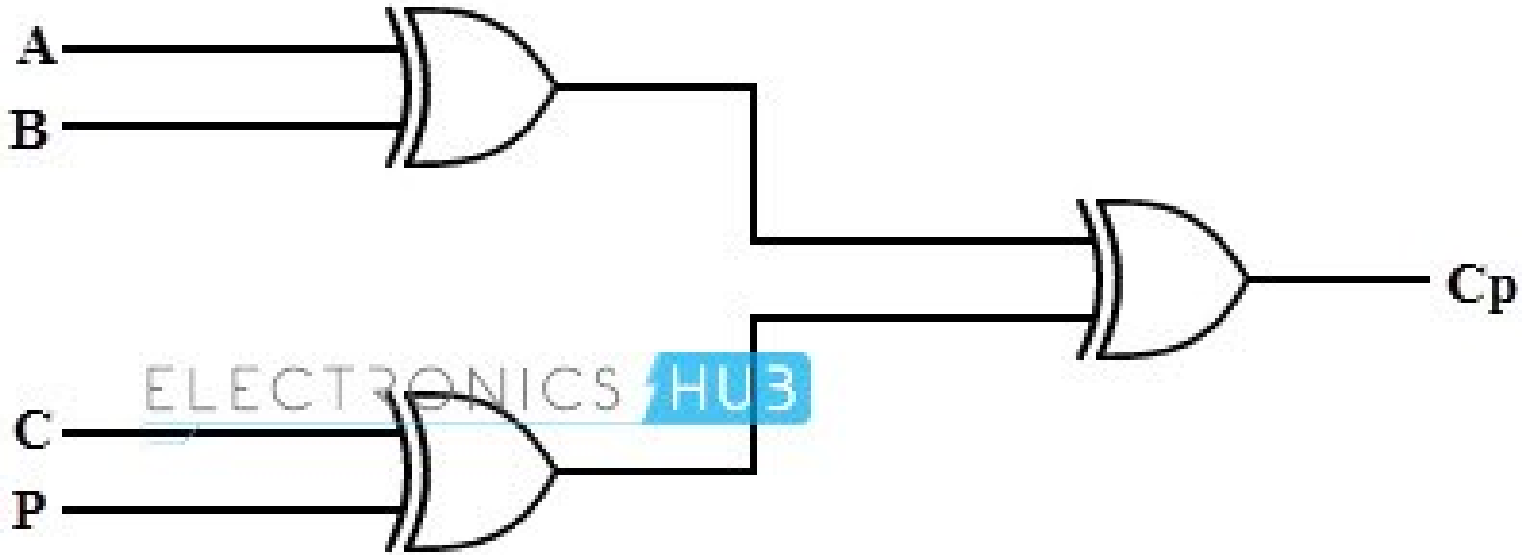
$$= (A \oplus B) \oplus (C \oplus D)$$

$$PEC = \bar{A}\ \bar{B}\ (\bar{C}\ D + C\ \bar{D}) + \bar{A}\ B\ (\bar{C}\ \bar{D} + C\ D) + A\ B\ (\bar{C}\ D + C\ \bar{D}) + A\ \bar{B}\ (\bar{C}\ \bar{D} + C\ D)$$

$$= \bar{A}\ \bar{B}\ (C \oplus D) + \bar{A}\ B\ (\overline{C \oplus D}) + A\ B\ (C \oplus D) + A\ \bar{B}\ (\overline{C \oplus D})$$

$$= (\bar{A}\ \bar{B} + A\ B)\ (C \oplus D) + (\bar{A}\ B + A\ \bar{B})\ (\overline{C \oplus D})$$

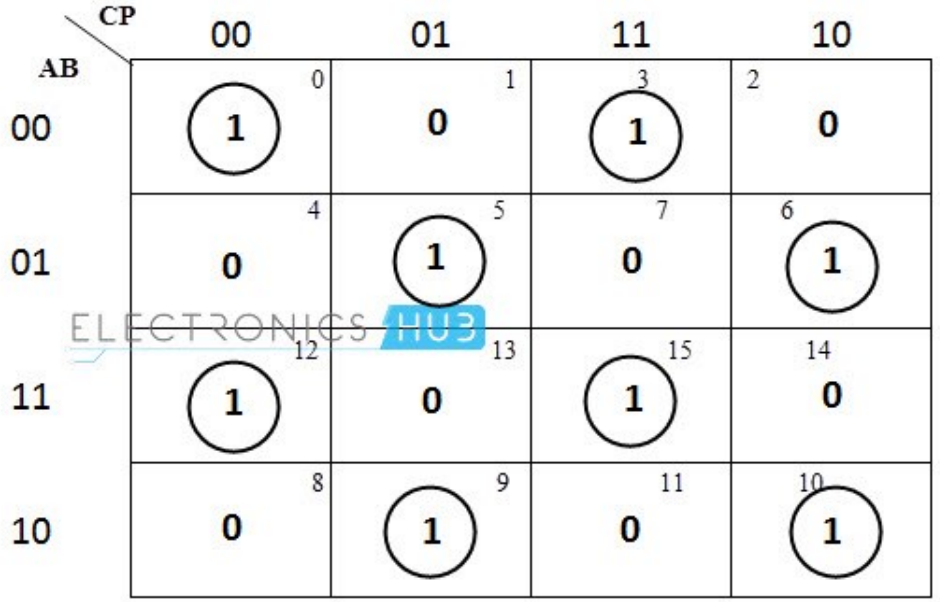
$$= (A \oplus B) \oplus (C \oplus D)$$



ODD PARITY CHECKER

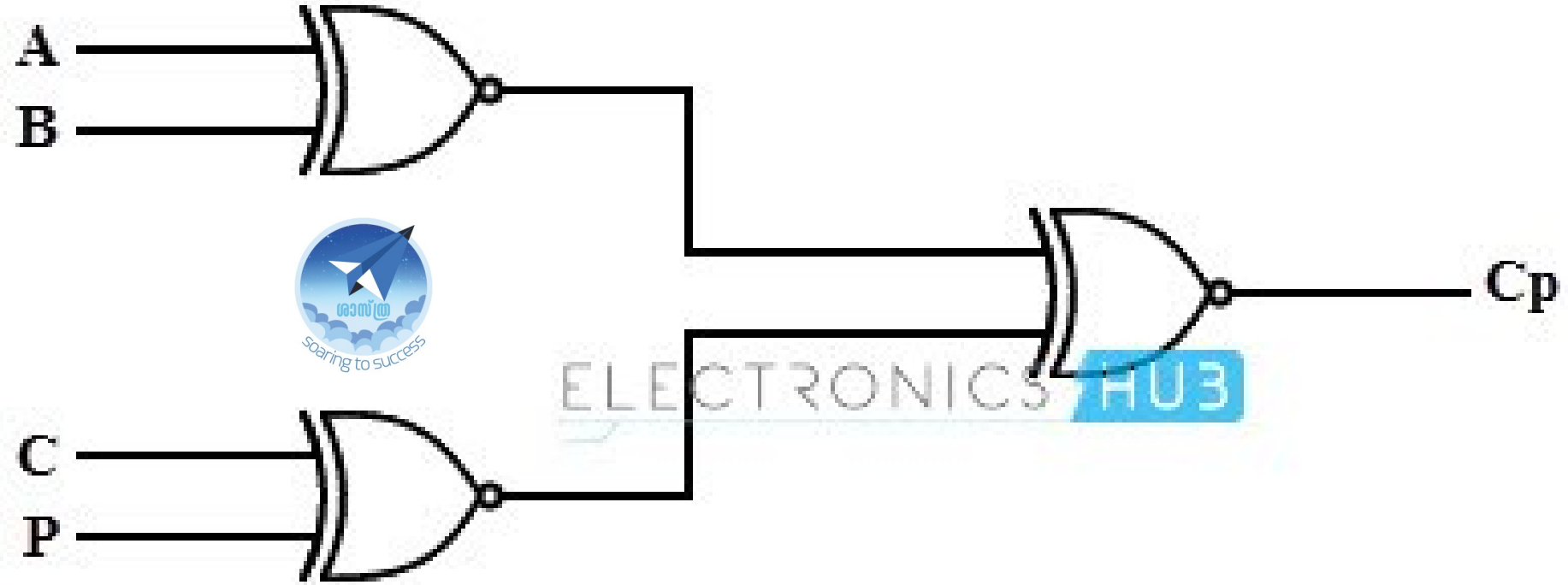
- Consider that a three bit message along with odd parity bit is transmitted at the transmitting end. Odd parity checker circuit receives these 4 bits and checks whether any error are present in the data.
- If the total number of 1s in the data is odd, then it indicates no error, whereas if the total number of 1s is even then it indicates the error since the data is transmitted with odd parity at transmitting end.
- The below figure shows the truth table for odd parity generator where **PEC =1** if the 4-bit message received consists of **even number of 1s** (hence the error occurred) and **PEC= 0** if the message contains **odd number of 1s** (that means no error).

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)



MODULE 3
(TOPIC-10)

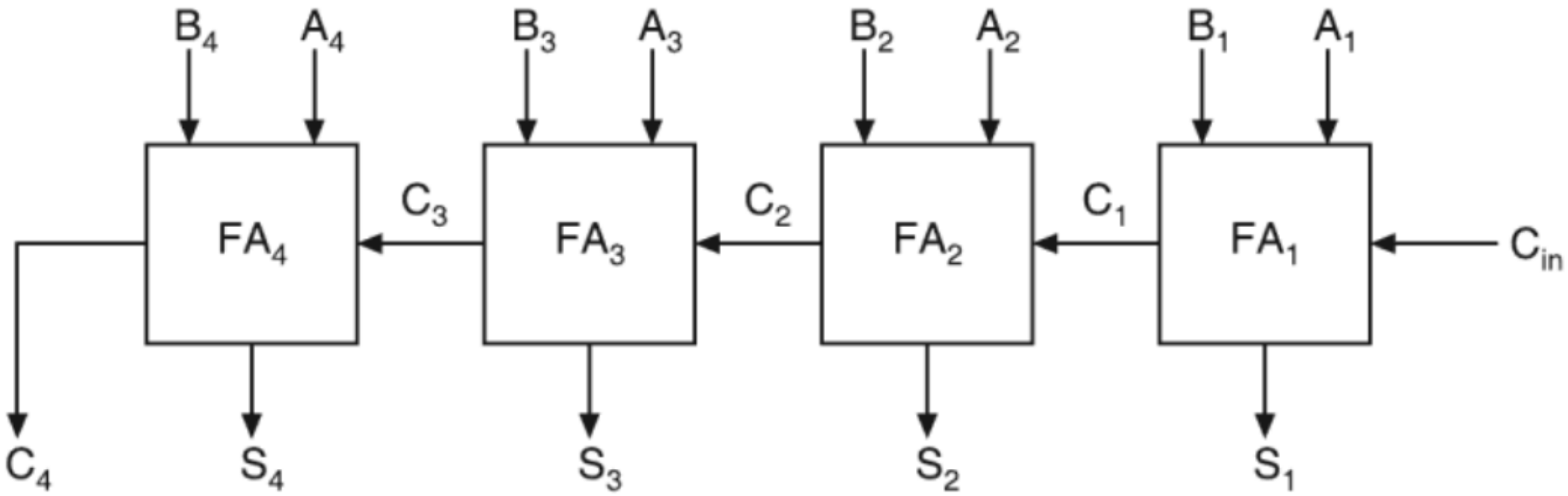
BINARY PARALLEL ADDER
CARRY LOOK AHEAD ADDER
BCD ADDER

https://youtu.be/8G29l4BFg_o
https://youtu.be/hfbvmGZ79SA

PARALLEL ADDER

- A single full adder performs the addition of two one bit numbers and an input carry.
- But a **Parallel Adder** is a digital circuit capable of finding the arithmetic **sum** of two binary numbers that is **greater than one bit** in length by operating on corresponding pairs of bits in parallel.
- It consists of **full adders connected in a chain** where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain.
- **A n bit parallel adder requires n full adders to perform the operation**. So for the two-bit number, two adders are needed while for four bit number, four adders are needed and so on. Parallel adders normally incorporate carry lookahead logic to ensure that carry propagation between subsequent stages of addition does not limit addition speed.

PARALLEL ADDER



CARRY LOOK AHEAD ADDER

- A carry-Lookahead adder is a fast parallel adder as it reduces the propagation delay by more complex hardware, hence it is costlier. In this design, the carry logic over fixed groups of bits of the adder is reduced to two-level logic, which is nothing but a transformation of the ripple carry design.
- This method makes use of logic gates so as to look at the lower order bits of the augend and addend to see whether a higher order carry is to be generated or not.
- **Carry-Lookahead Adder Ics** A typical carry-Lookahead generator IC is 74182 which accept four pairs of active low carry propagate (as P0, P1, P2 and P3) and carry generate (Go, G1, G2 and G3) signals and an active high input (Cn).

CARRY LOOK AHEAD ADDER

Consider the full adder circuit shown above with corresponding truth table. If we define two variables as carry generate Gi and carry propagate Pi then,

$$P_i = A_i \oplus B_i$$

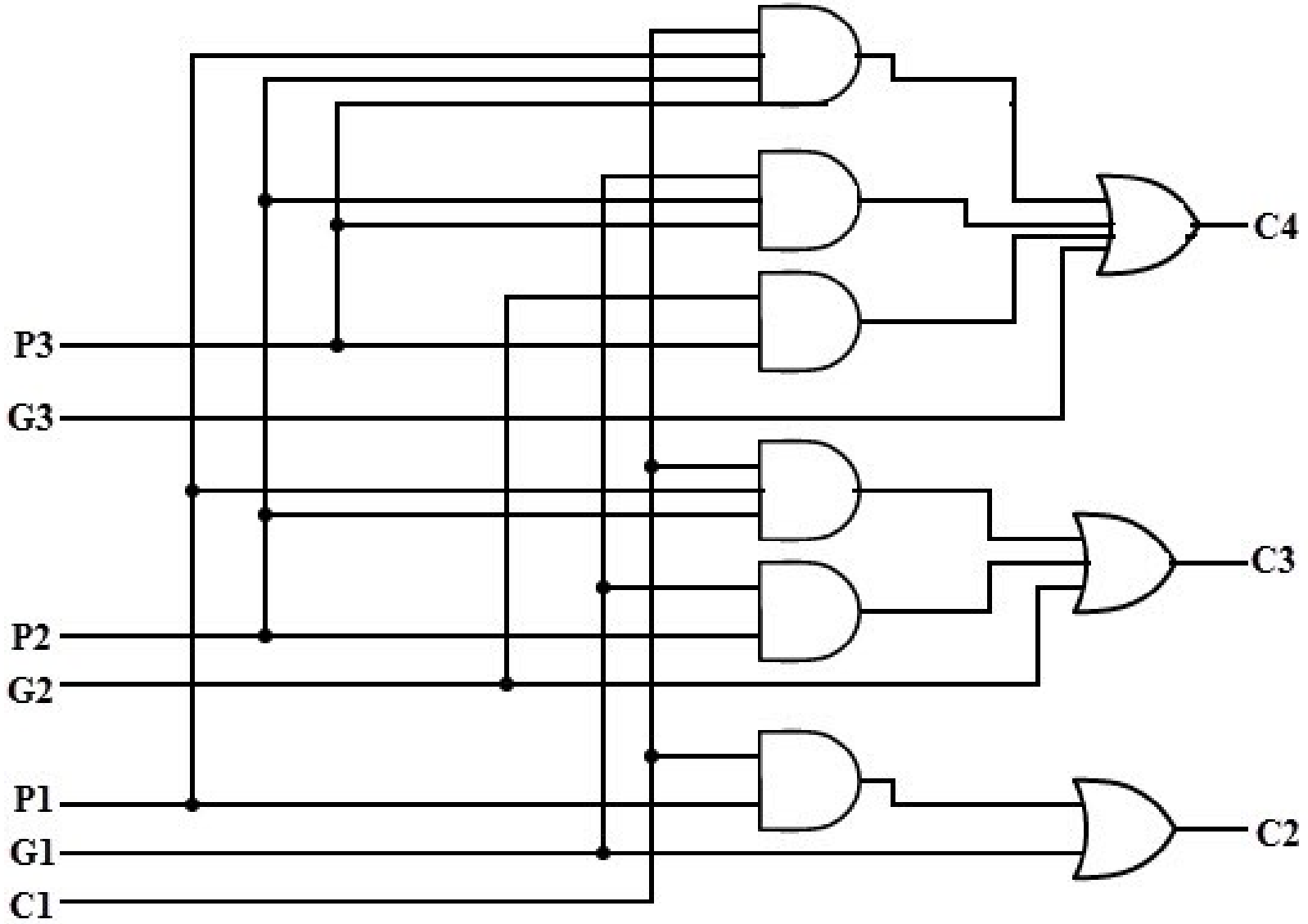
$$Gi = Ai\ Bi$$

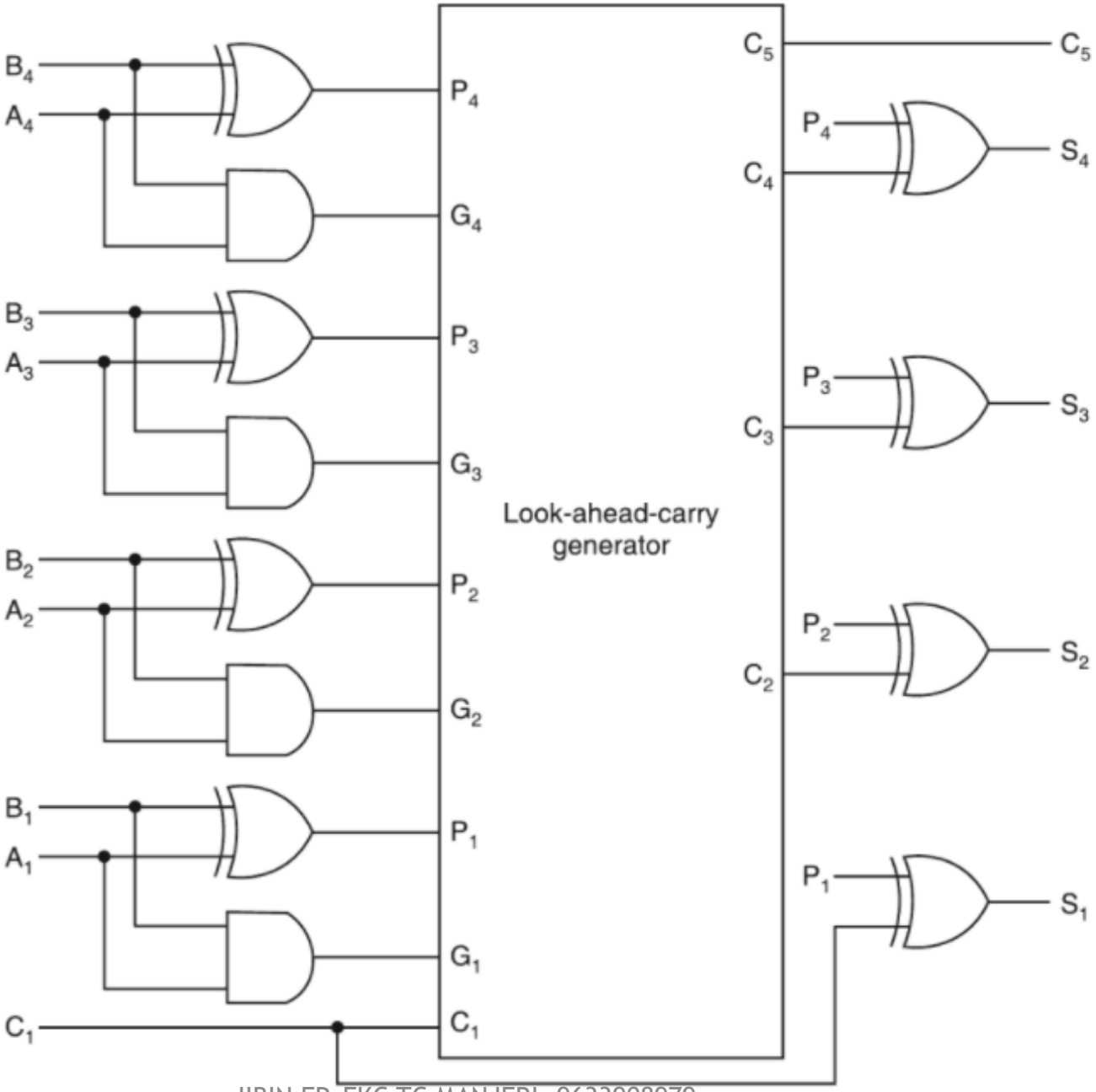
The sum output and carry output can be expressed as

$$Si = Pi \oplus Ci$$

$$C\ i+1 = Gi + Pi\ Ci$$

A	B	Ci	C i +1	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	


P3
G3
P2
G2
P1
G1
C1
C4
C3
C2


B4
A4
B3
A3
B2
A2
B1
A1
C1
P4
G4
P3
G3
P2
G2
P1
G1
C1
Look-ahead-carry generator
C5
C4
C3
C2
P4
P3
P2
P1
C5
S4
S3
S2
S1

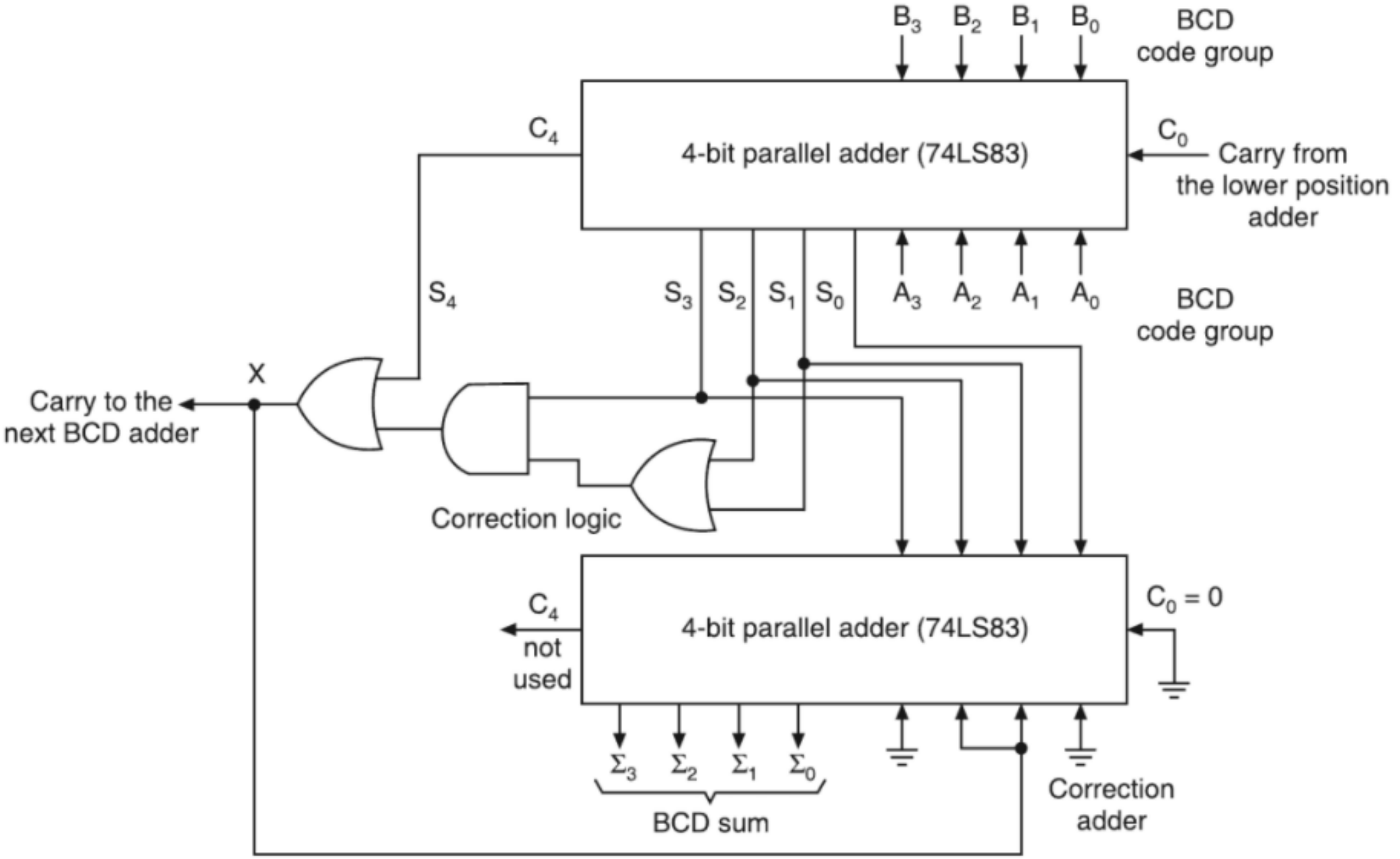
BCD ADDER

- BCD adder must be able to operate in accordance with above steps
- **1) Add two bit BCD Code Groups , using straight binary addition**
- **2) Determine if the sum of addition is greater than 1001 (Decimal 9); If it is , add 0110 (Decimal 6) to this sum and generate a carry to the next decimal position**
- The first requirement is easily met by using a 4 bit binary parallel adder
- The sum output S4 S3 S2 S1 S0 can range anywhere from 00000 to 10010

The Circuitry for a BCD adder must include the logic needed to detect whenever the sum is greater than 01001, so that correction can be added in

The circuit consists of three basic parts. The two BCD code groups $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are added together in the upper 4-bit adder, to produce the sum $S_4S_3S_2S_1S_0$. The logic gates shown implement the expression for X. The lower 4-bit adder will add the correction 0110 to the sum bits, only when X = 1, producing the final BCD sum output represented by $\Sigma_3\Sigma_2\Sigma_1\Sigma_0$. The X is also the carry-out that is produced when the sum is greater than 01001. Of course, when X = 0, there is no carry and no addition of 0110. In such cases, $\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = S_3S_2S_1S_0$.

Two or more BCD adders can be connected in cascade when two or more digit decimal numbers are to be added. The carry-out of the first BCD adder is connected as the carry-in of the second BCD adder, the carry-out of the second BCD adder is connected as the carry-in of the third BCD adder and so on.


B3 B2 B1 B0
BCD code group
C4
4-bit parallel adder (74LS83)
C0
Carry from the lower position adder
S4
S3 S2 S1 S0
A3 A2 A1 A0
BCD code group
X
Carry to the next BCD adder
Correction logic
C4
not used
4-bit parallel adder (74LS83)
C0 = 0
Σ3 Σ2 Σ1 Σ0
BCD sum
Correction adder

PREPARED BY

SHASTRA TECHNICAL INSTITUTE

KTU | PSC | UPSC

ശാസ്ത്ര soaring to success

Jibin EP
Assistant Professor
Department of Electronics and Communication
Eranad Knowledge City Technical Campus Manjeı
Contact :9633908979, 70121716607
Email : epjibin@gmail.com, jibin@ekc.edu.in

YouTube

https://www.youtube.com/ShastraTechnicalInstitute