Unit III

COMBINATIONAL SYSTEMS

Course Articulation Matrix

18ECC103J - DIGITAL ELECTRONIC PRINCIPLES

Course Learning Outcomes (CLOs)	Program Outcomes (POs) Graduate Attributes												PSO		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
Simplify Boolean expressions; carry out arithmetic operations with binary numbers; apply parity method for error detection and correction.	H	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Explain the operational characteristics / properties of digital ICs; implement gates as well as other types of IC devices using two major IC technologies, TTL and CMOS.	H	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Identify eight basic types of fixed-function combinational logic functions and demonstrate how the devices / circuits can be used in building complete digital systems such as computers.		M	H	-	H	-	-	-	-	-	-	-	-	-	-
Analyze and design Mealy and Moore models of sequential circuits using several types of flip-flops.		M	H	-	H	-	-	-	-	-	-	-	-	-	-
Implement multiple output combinational logic circuits using PLDs; Explain the operation of a CPLD and FPGA.	-	M	H	-	L	-	-	-	-	-	-	-	-	-	-
Solve specific design problem, which after completion will be verified using modern engineering tools such as PSPICE / Logisim	-	M	H	-	H	-	-	-	H	-	-	-	M	-	L

Program Outcomes (PO)

PO1- Engineering knowledge
Apply the knowledge of Mathematics, Science, Engineering fundamentals, and an Engineering specialization to the solution of complex Engineering problems.

PO2- Problem analysis
Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3- Design/development of solutions
Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4-Conduct investigations of complex problems
Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5-Modern tool usage
Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6-The engineer and society
Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7-Environment and sustainability
Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8-Ethics
Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9-Individual and team work
Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10- Communication
Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11-Project management and finance
Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12- Life-long learning
Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Index

Unit-III: Combinational Systems

CLO3: *Identify eight basic types of fixed-function combinational logic functions and demonstrate how the devices / circuits can be used in building complete digital systems such as computers.*

Learning Unit/ Module	Session	Description of Topic (Theory)	No. of Contact hours	CO	PO	BL	Reference
				3	3,5	1,2,3	[1] chapter 4
	1.	Binary arithmetic units, Adder	1				
	1.	Design of Half adder, Design of Full adder	1	3	3,5	1,2,3	[1] chapter 4
	1.	Subtractor , Design subtractor using logic gates	1	3	3,5	1,2,3	[1] chapter 4
Unit-III: Combinational Systems	1.	n-bit parallel adder & subtractor, look ahead carry generator	1	3	3,5	1,2,3	[1] chapter 4
	1.	Decoder, Encoder	1	3	3,5	1,2,3	[1] chapter 4
	1.	Multiplexer, Demultiplexer, Code converters, Magnitude comparators	2	3	3,5	1,2,3	[1] chapter 4
	1.	Parity generators (Odd parity), Parity generators (Even parity)	1	3	3,5	1,2,3	[1] chapter 4
	1.	Implementation of combinational logic by standard IC's.	1	3	3,5	1,2,3	[1] chapter 4

Reference: Morris Mano M, Michael D. Ciletti, Digital Design with an Introduction to the Verilog HDL, 5th ed., Pearson Education, 2014 .

Combinational Logic

- Logic circuits for digital systems may be combinational or sequential.
- A combinational circuit consists of input variables, logic gates, and output variables.
- Hence, a combinational circuit can be described by:

- A truth table that lists the output values for each combination of the input variables, or m Boolean functions, one for each output variable.

Combinational vs. Sequential Circuits

- Combinational circuits are memory-less. Thus, the output value depends ONLY on the current input values.
- Sequential circuits consist of combinational logic as well as memory elements (used to store certain circuit states). Outputs depend on BOTH current input values and previous input values (kept in the storage elements).

Design Procedure

- Given a problem statement:
- Determine the number of inputs and outputs
- Derive the truth table
- Simplify the Boolean expression for each output
- Produce the required circuit

Binary Adders

Half Adder: is a combinational circuit that performs the addition of two bits, this circuit needs two binary inputs and two binary outputs.

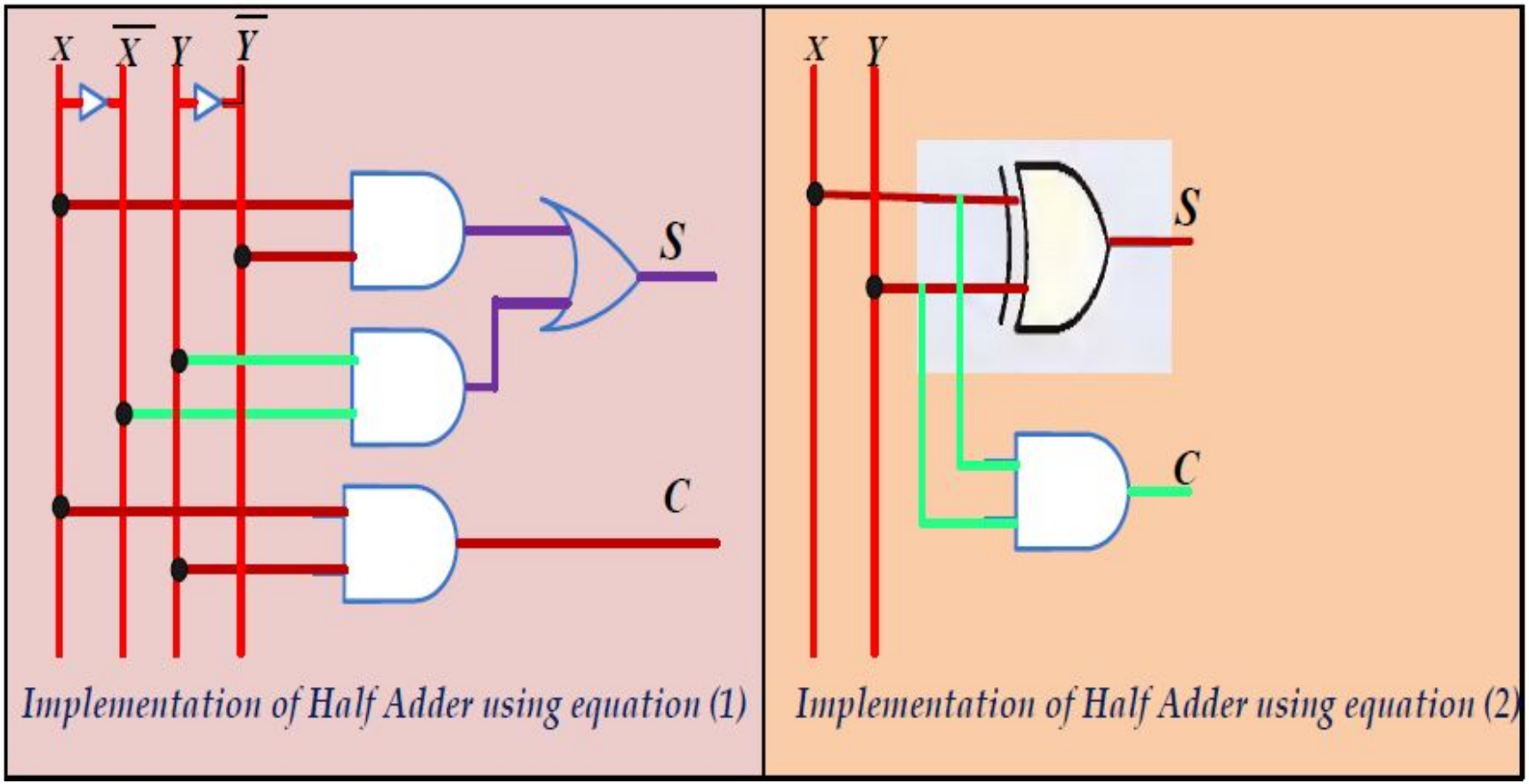
Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0
Truth table			

The simplified Boolean function from the truth table:

$$\begin{cases} S = \bar{X}Y + X\bar{Y} \\ C = XY \end{cases} \quad 1$$ (Using sum of product form)

Where S is the sum and C is the carry.

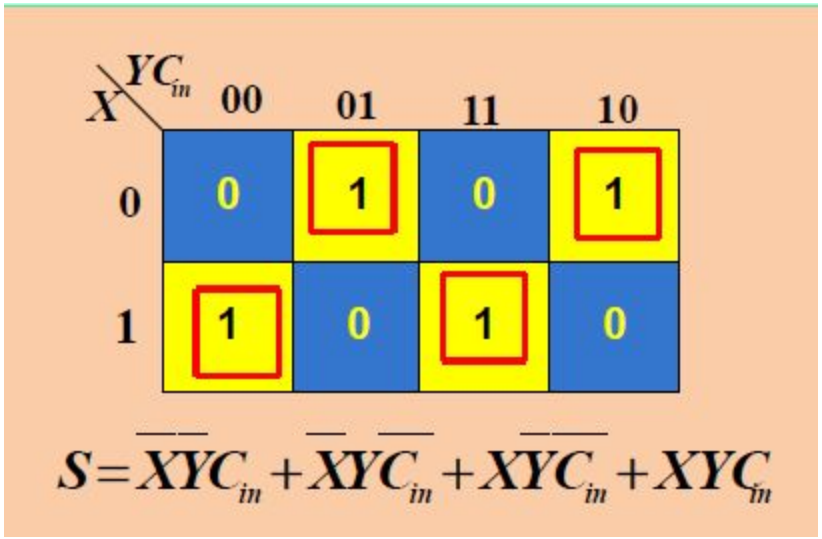
$$\begin{cases} S = X \oplus Y \\ C = XY \end{cases} \quad 2$$ (Using **XOR** and **AND** Gates)

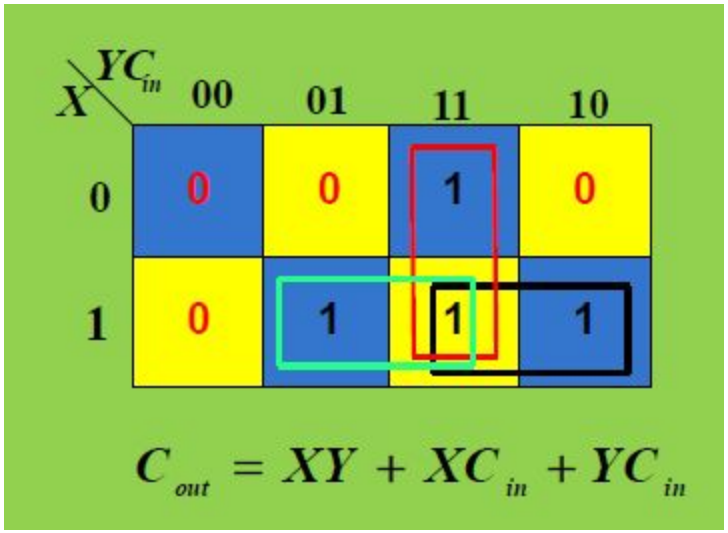

X
$\overline{X}$
Y
$\overline{Y}$
S
C
Implementation of Half Adder using equation (1)
X
Y
S
C
Implementation of Half Adder using equation (2)

Full Adder

Inputs			Outputs	
X	Y	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

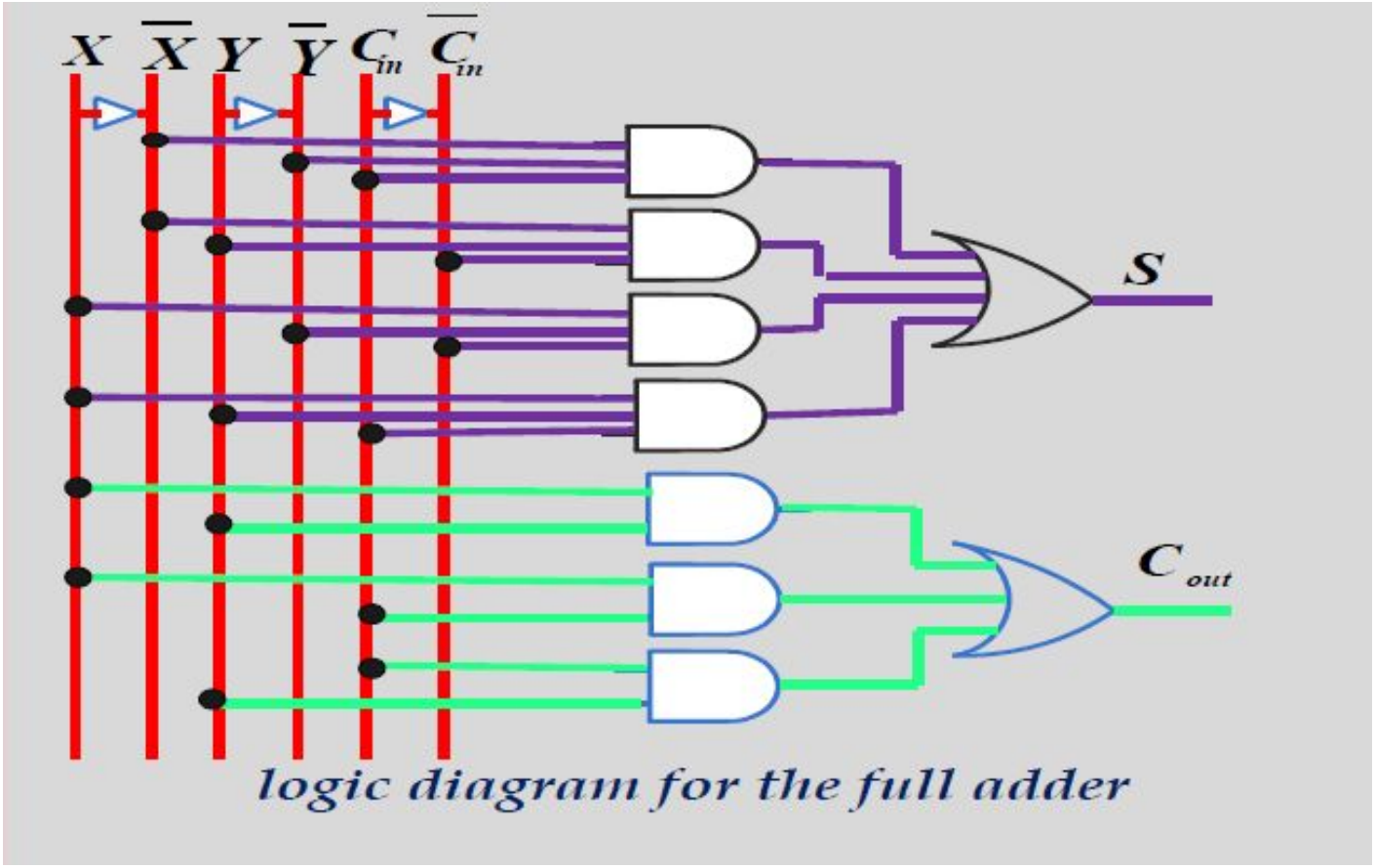
Truth table for the full adder





$$\begin{cases} S = \bar{X}\bar{Y}C_{in} + \bar{X}Y\bar{C}_{in} + X\bar{Y}\bar{C}_{in} + XYC_{in} \\ C_{out} = XY + XC_{in} + YC_{in} \end{cases}$$

FA- using basic gates



logic diagram for the full adder

FA using two HA and OR gate

$$\left\{\begin{matrix} S = C_{in} \oplus (X \oplus Y) \\ C_{out=}\, C_{in} \cdot (X \oplus Y) + XY \end{matrix}\right\}$$

Proof:

The sum:

$$S = \overline{X}\,\overline{Y}C_{in} + \overline{X}Y\overline{C_{in}} + X\overline{Y}\,\overline{C_{in}} + XYC_{in}$$

$$= \overline{C_{in}}(\overline{X}Y + X\overline{Y}) + C_{in}(\overline{X}\,\overline{Y} + XY)$$

$$= \overline{C_{in}}(\overline{X}Y + X\overline{Y}) + C_{in}\overline{(\overline{X}Y + X\overline{Y})}$$

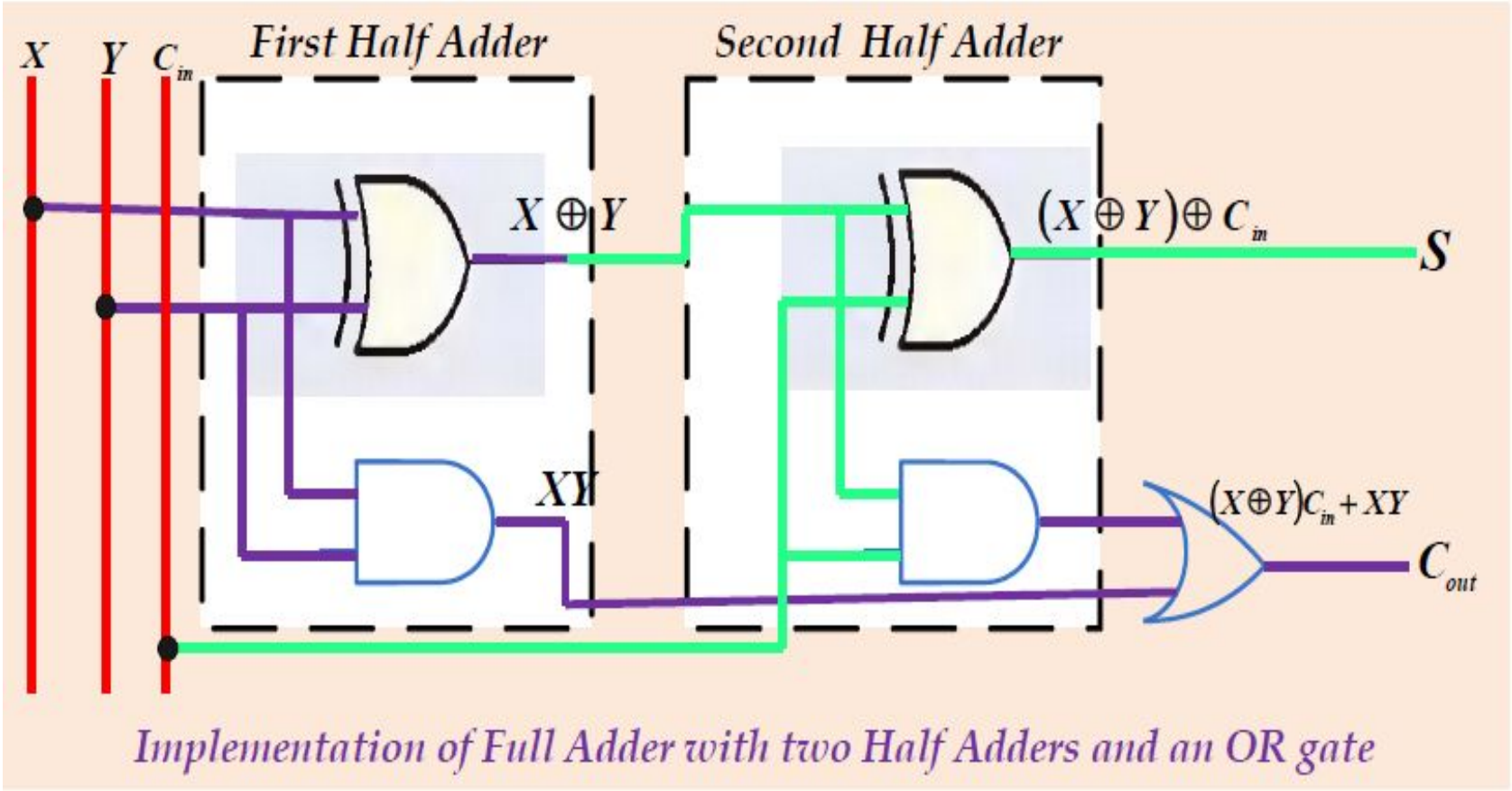
$$S = C_{in} \oplus (X \oplus Y)$$

The carry output:

$$C_{out} = \overline{X}YC_{in} + X\overline{Y}C_{in} + XYC_{in} + XY\,\overline{C_{in}}$$

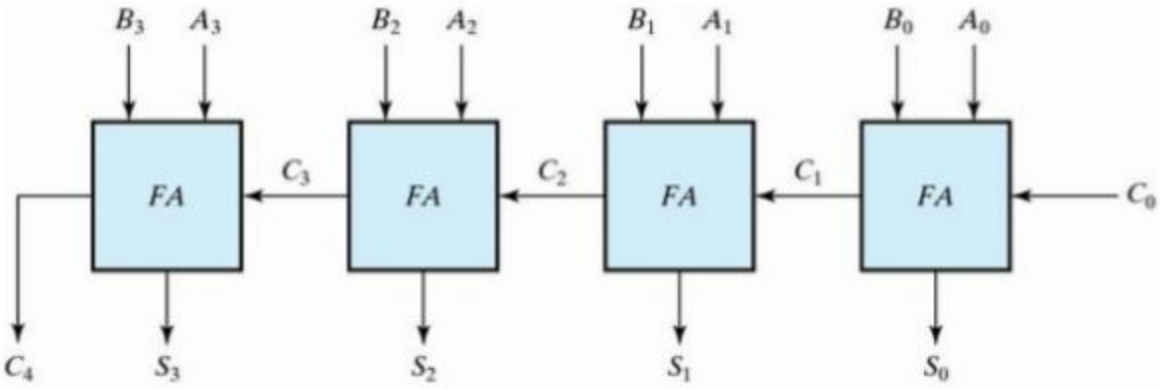
$$= C_{in}(\overline{X}Y + X\overline{Y}) + XY(C_{in} + \overline{C_{in}})$$

$$C_{out=}\, C_{in} \cdot (X \oplus Y) + XY$$



Implementation of Full Adder with two Half Adders and an OR gate

4-bit Full adder



Example:

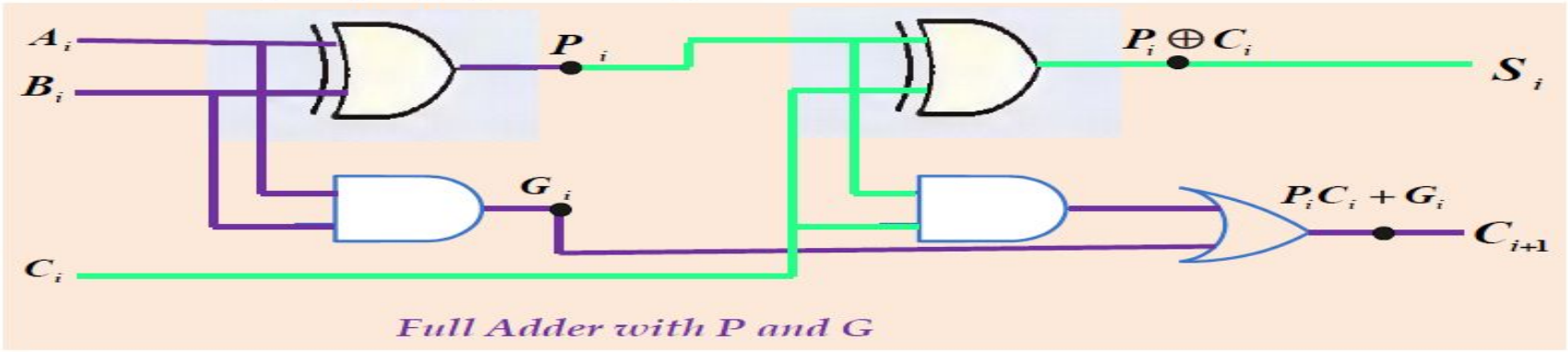
$A + B$ $(A = 1011)$ and $(B = 0011)$

Subscript i	3	2	1	0	
Input Carry	0	1	1	0	C_i
A	1	0	1	1	A_i
+					
B	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output Carry	0	0	1	1	C_{i+1}

$C_0 = 0$

Carry Propagation

- The addition of $A + B$ binary numbers in ***parallel*** implies that all the bits of A and B are available for computation at the same time.
- As in any combinational circuit, the signal must ***propagate*** through the gates before the correct output sum is available.
- The output will not be correct unless the signals are given enough time to propagate through the gates connected form the input to the output.
- The longest ***propagation delay time*** in an adder is the time it takes the carry to propagate through the full adders.



Full Adder with P and G

- The signal form the carry input C_i to the output carry C_{i+1} propagates through an ***AND*** gate and an ***OR*** gate, which equals ***2 gate levels***.
  - If there are ***4*** full adders in the binary adder, the output carry C_4 would have ***2×4=8 gate levels***, form C_0 to C_4

- The ***carry propagation time*** is an important attribute of the adder because it limits the speed with which two numbers are added.
- To reduce the carry propagation delay time:
  1) Employ faster gates with reduced delays.
  2) Employ the principle of Carry Lookahead Logic.

Proof: ***(using carry lookahead logic)***

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

The output sum and carry are:

$$S_i = P_i \oplus C_i$$

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

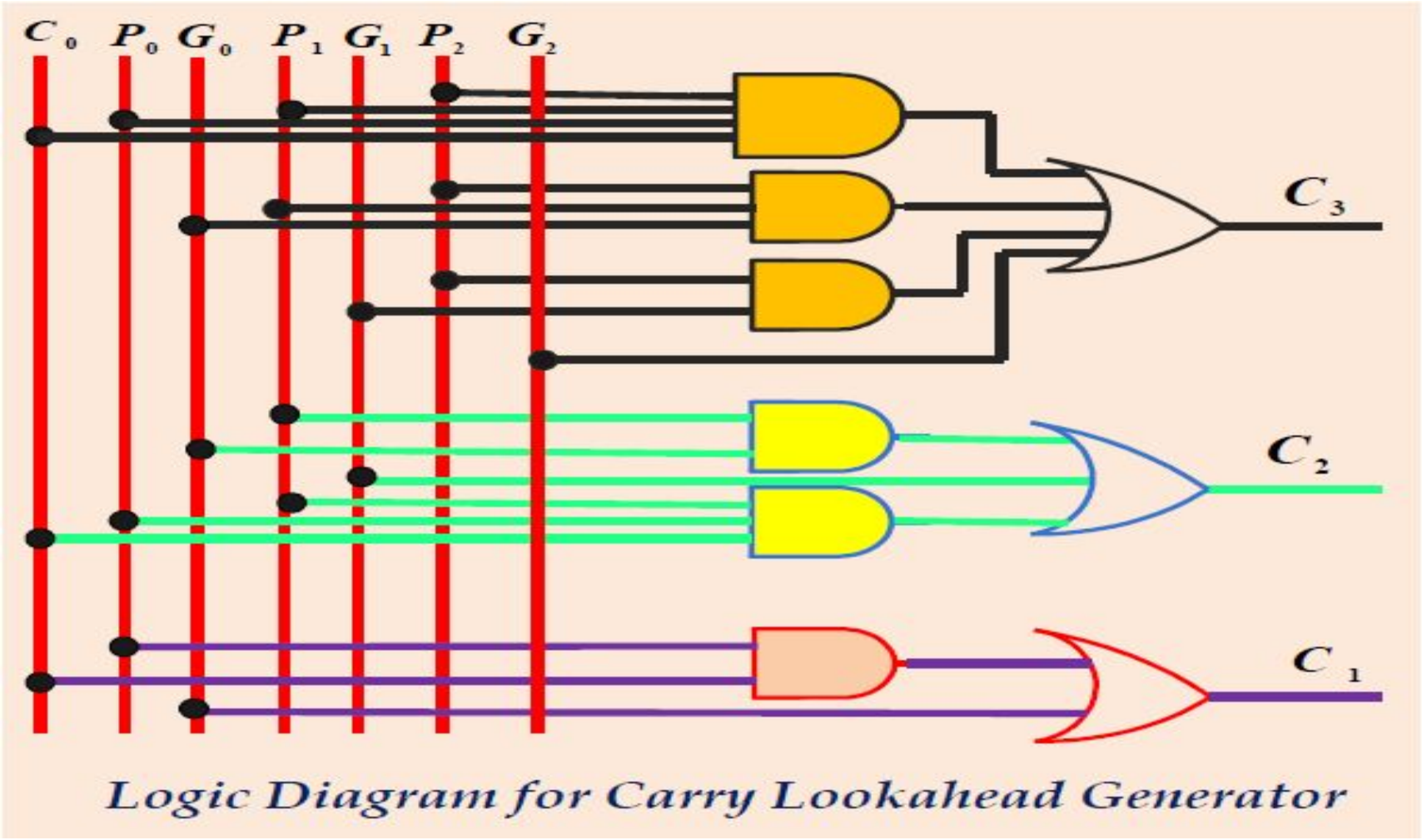
- ✓ G_i-called a ***carry generate***, and it produces a carry of 1 when both A_i and B_i are 1.
- ✓ P_i-called a ***carry propagate***, it determines whether a carry into stage i will propagate into stage $i+1$.
- ✓ The ***Boolean function*** for the carry outputs of each stage and substitute the value of each C_i form the previous equations:

$$\left\{\begin{array}{l} C_0 = \textit{input carry} \\ C_1 = G_0 + P_0 C_0 \\ C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) \\ \quad = G_1 + P_1 G_0 + P_1 P_0 C_0 \\ C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_0) \\ \quad = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \end{array}\right\}$$

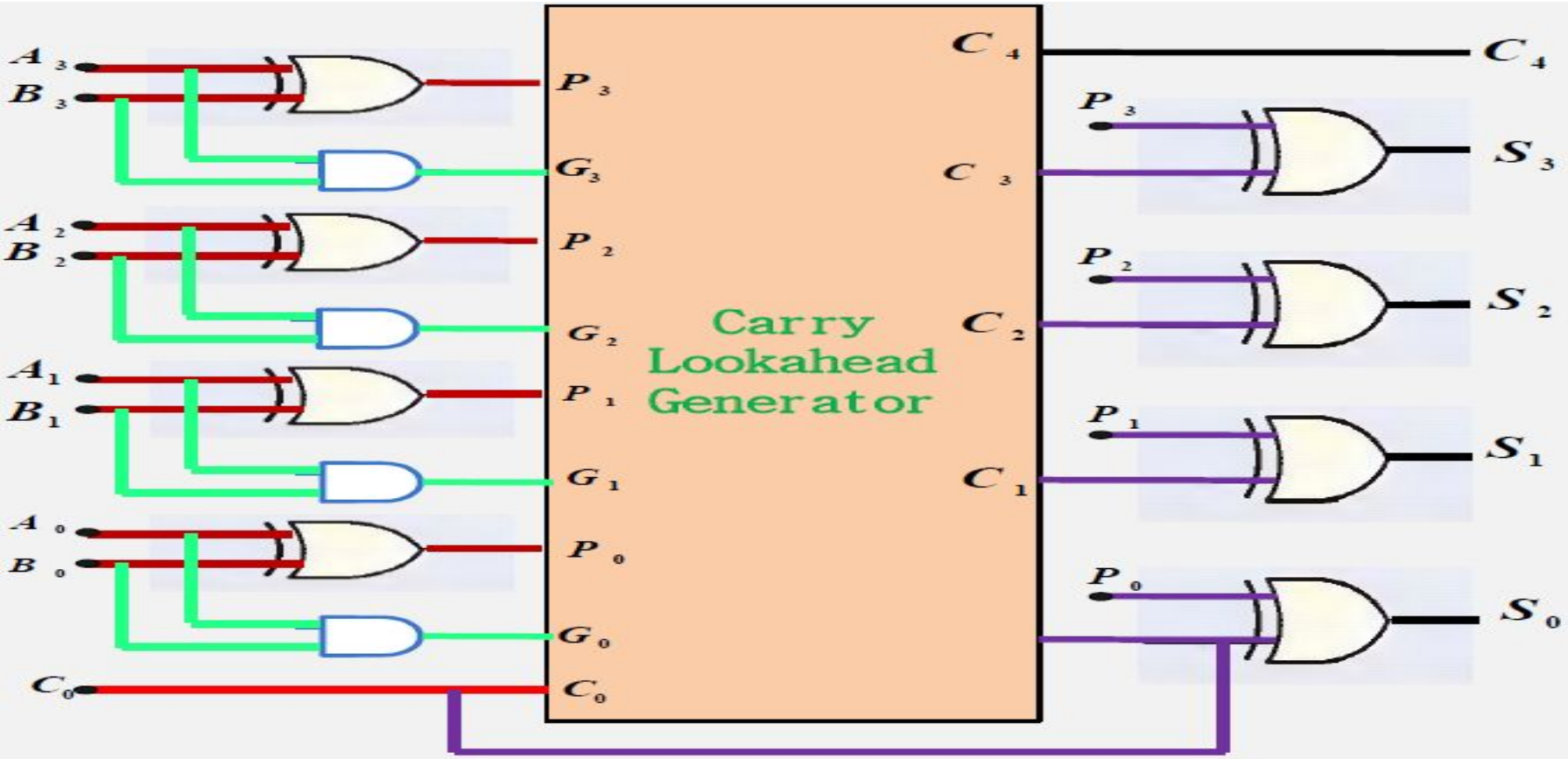
- The three Boolean functions C_1, C_2 and C_3 are implemented in the ***carry lookahead generator.***

> ***The two level-circuit for the output carry C_4 is not shown, it can be easily derived by the equation.***

- C_3 does not have to wait for C_2 and C_1 to propagate, in fact C_3 is propagated at the same time as C_1 and C_2.



Logic Diagram for Carry Lookahead Generator



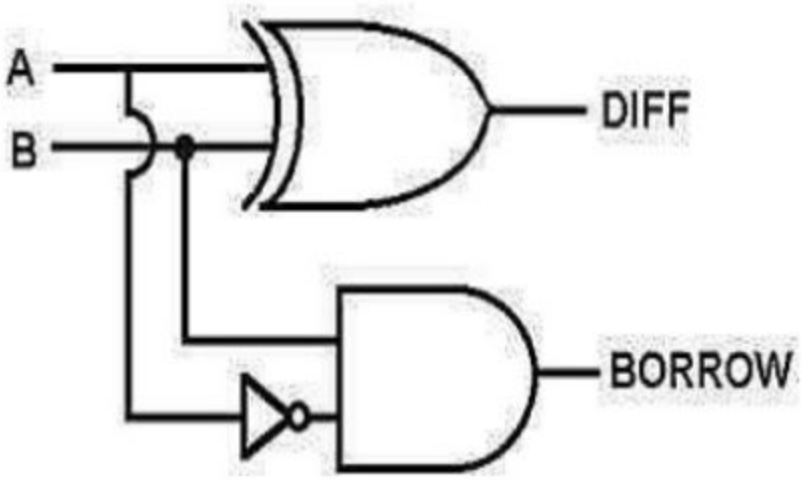
four-bit adder with a carry lookahead scheme

Subtractor is an electronic logic circuit for calculating the difference between two binary numbers which provides the difference and borrow as output.

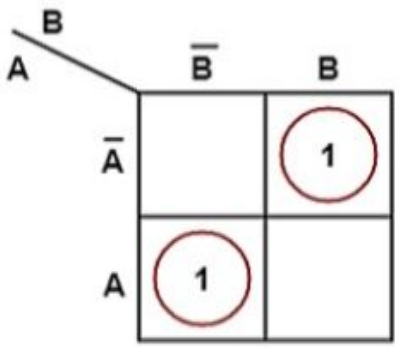
Half Subtractor is used for subtracting one single bit binary number from another single bit binary number.

It has two inputs; Minuend (A) and Subtrahend (B) and two outputs; Difference (D) and Borrow (B_{out}).

Input		Output	
A	B	Difference (D)	Borrow (B_{out})
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

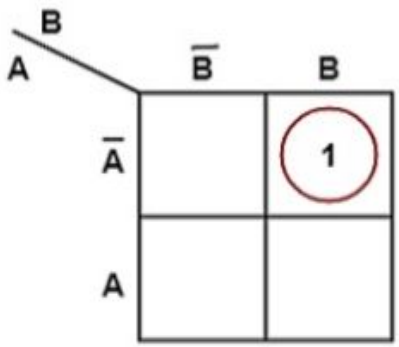


For D:



$D = A \oplus B$

For b:



$b = \bar{A}\,B$

Borrow = $\bar{A}.B$

Difference = $A \oplus B$

Full subtractor

A logic Circuit Which is used for subtracting three single bit binary numbers is known as Full Subtractor.

- It has three inputs;

1. Minuend (A),
2. Subtrahend(B)
3. Subtrahend(C)

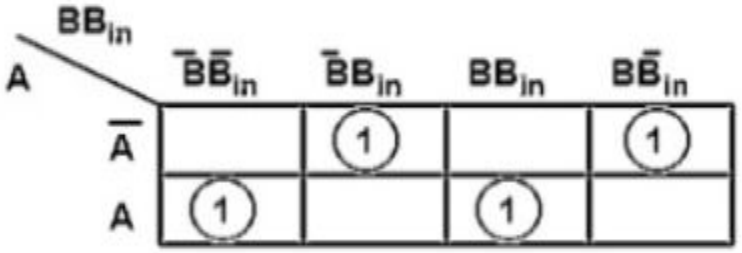
- two outputs;

1.Difference (D)

2.Borrow (B_{out}).

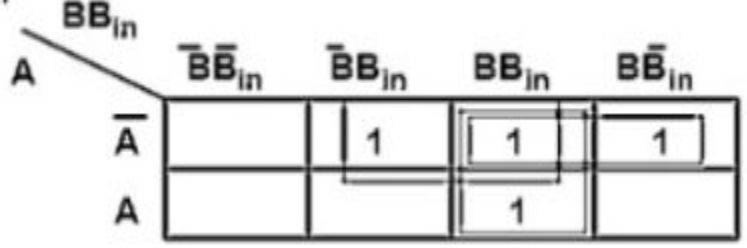
Input			Output	
A	B	C	D	$B_{(out)}$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

For D:



$$D = A \oplus B \oplus B_{in}$$

For B_{in}:



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

From the Truth Table The Difference and Borrow will written as,

Difference=A'B'C+A'BC'+AB'C'+ABC

Difference=A $\oplus$B$\oplus$C

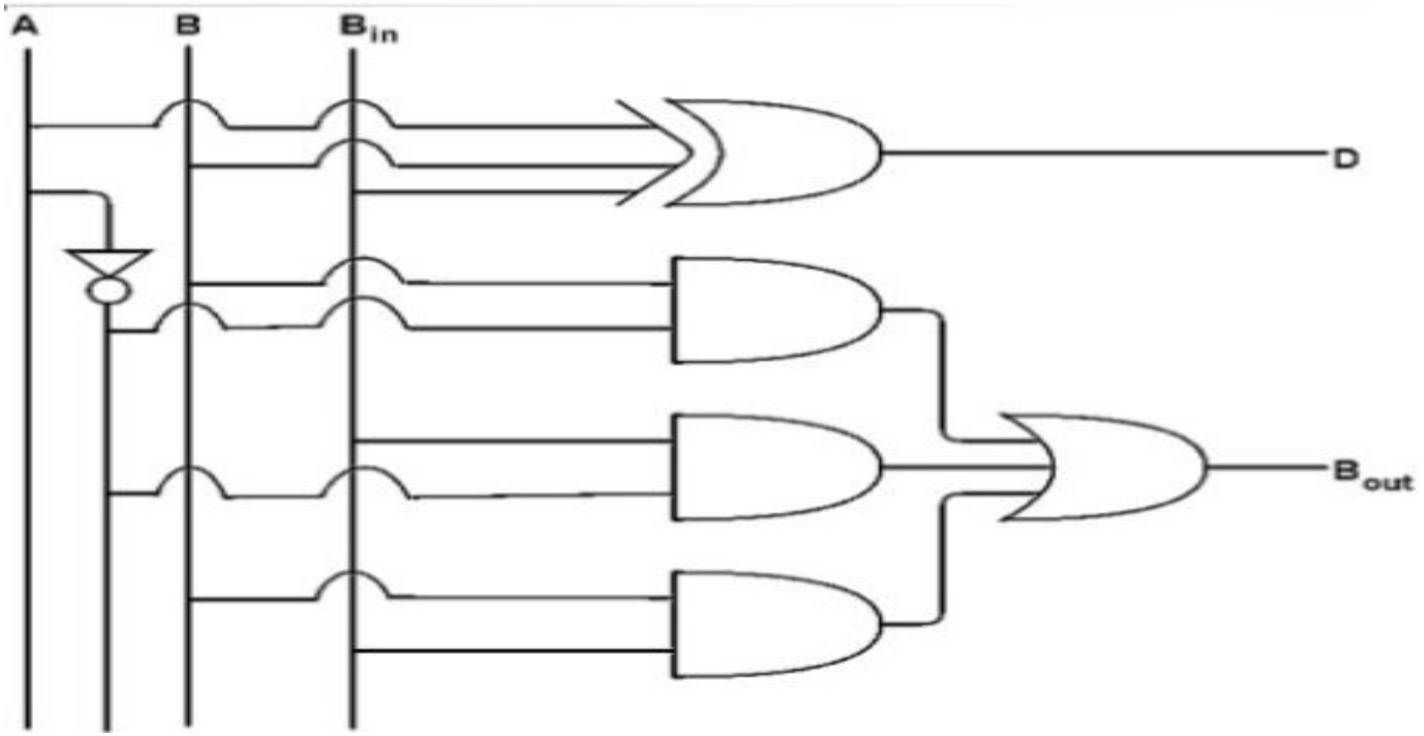
Borrow=A'B'C+A'BC'+A'BC+ABC

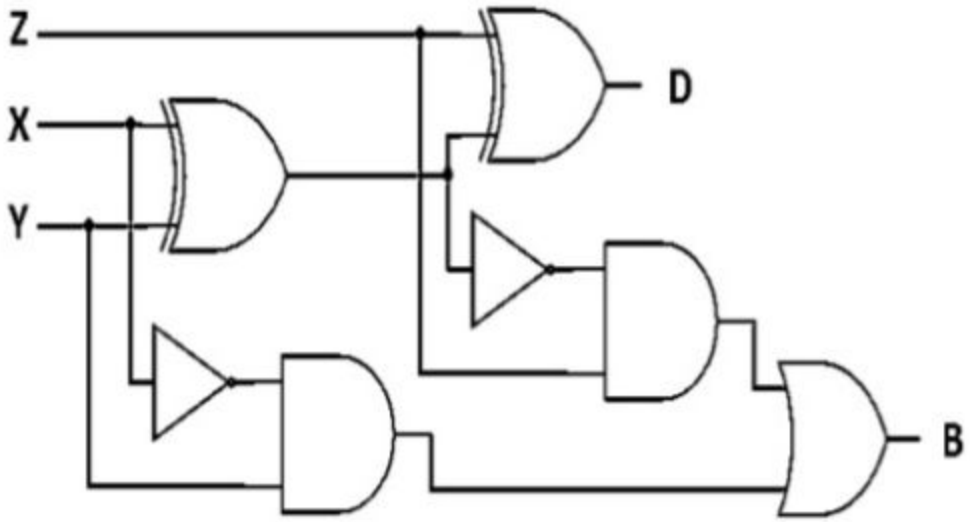
=A'B'C+A'BC'+A'BC+A'BC+A'BC+ABC

=A'C(B'+B)+A'B(C'+C)+BC(A'+A)

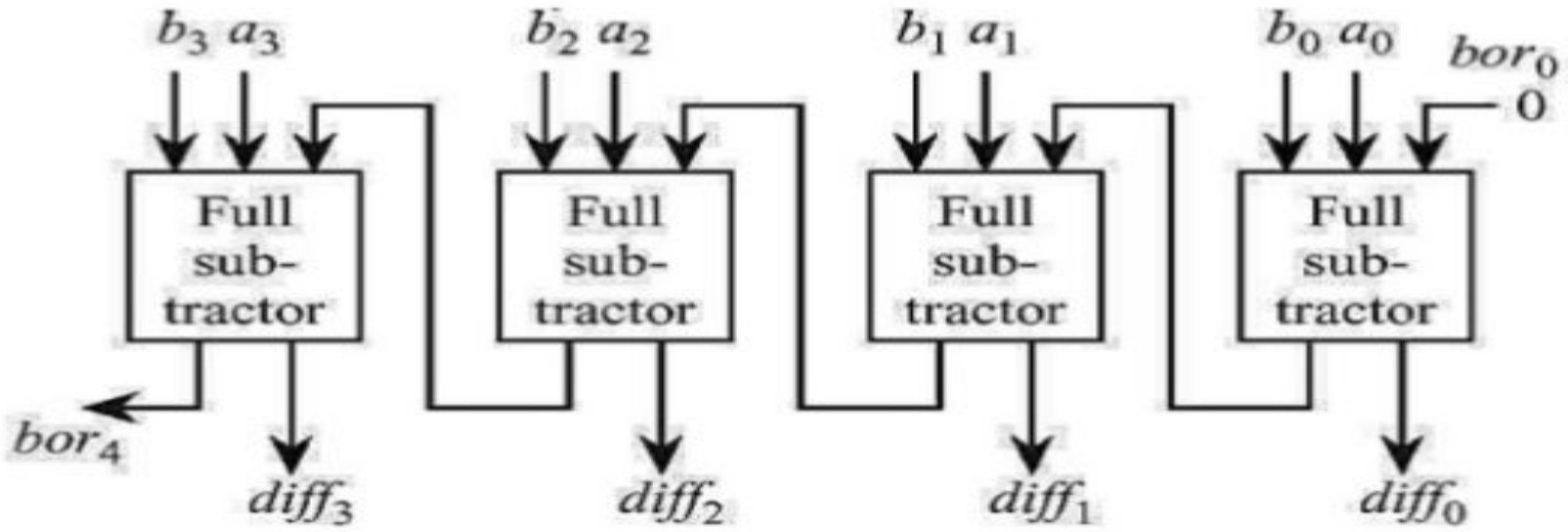
Borrow=A'C+A'B+BC

B(out) = BC + (B $\oplus$ C) A

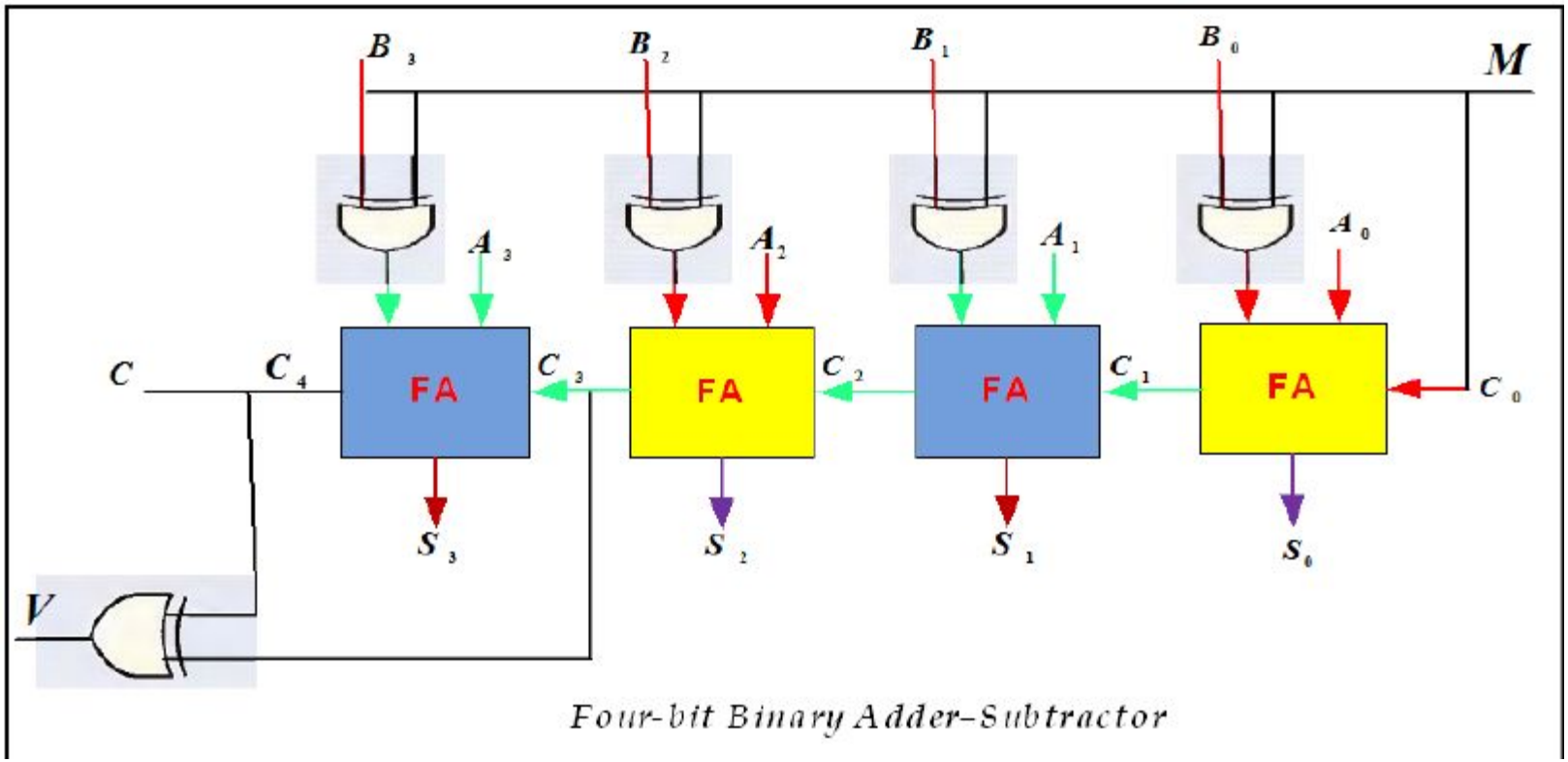

A
B
B_{in}
D
B_{out}


Z
X
Y
D
B

4-bit Full subtractor



Binary Adder-Subtractor



Four-bit Binary Adder-Subtractor

The addition and subtraction operations can be combined into one circuit with one common binary adder by including an ***exclusive-OR*** gate with each full-adder.

The mode input M controls the operation as the following:

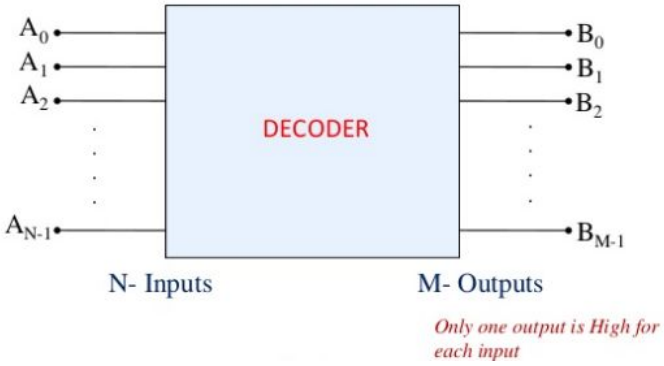
- ($M = 0$ →adder.
- $M = 1$ → subtractor.

- Each *XOR* gate receives M signal and B
  - When $M = 0$ then $B \oplus 0 = B$ and the carry = 0, then the circuit performs the operation $A + B$.
  - When $M = 1$ then $B \oplus 1 = \bar{B}$ and the carry = 1, then the circuit performs the operation $A - B$.
- The *exclusive-OR* with output V is for detecting an overflow.

Decoder &Encoder

DECODER

- A decoder is a combinational circuit.
- A decoder accepts a set of inputs that represents a binary number and activates only that output corresponding to the input number. All other outputs remain inactive.
- Fig. 1 shows the block diagram of decoder with 'N' inputs and 'M' outputs.
- There are 2^N possible input combinations, for each of these input combination only one output will be HIGH (active) all other outputs are LOW
- Some decoder have one or more ENABLE (E) inputs that are used to control the operation of decoder.



2 to 4 Line Decoder:

- Block diagram of 2 to 4 decoder is shown in fig. 2
- A and B are the inputs. (No. of inputs =2)
- No. of possible input combinations: $2^2=4$
- No. of Outputs : $2^2=4$, they are indicated by D_0, D_1, D_2 and D_3
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.

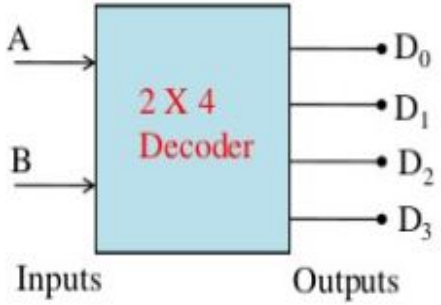


Fig. 2

TRUTH TABLE

INPUTS		OUTPUTS			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

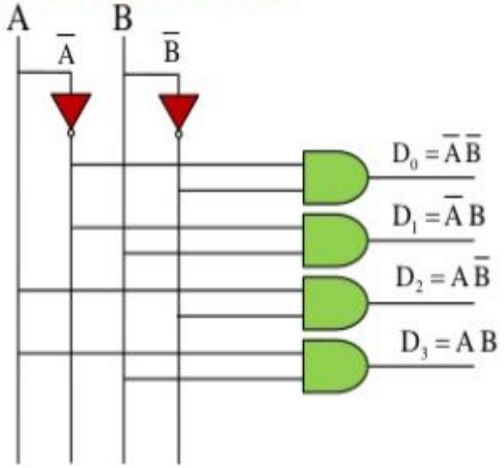
BOOLEAN EXPRESSION:

From Truth Table

$$D_0 = \bar{A}\,\bar{B} \qquad D_1 = \bar{A}\,B$$

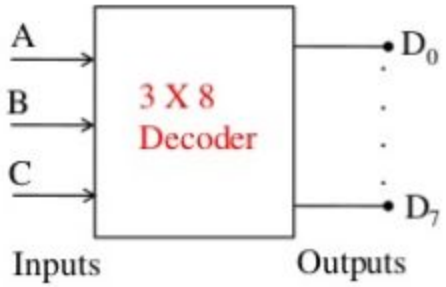
$$D_2 = A\,\bar{B} \qquad D_3 = AB$$

LOGIC DIAGRAM:



3 to 8 Line Decoder:

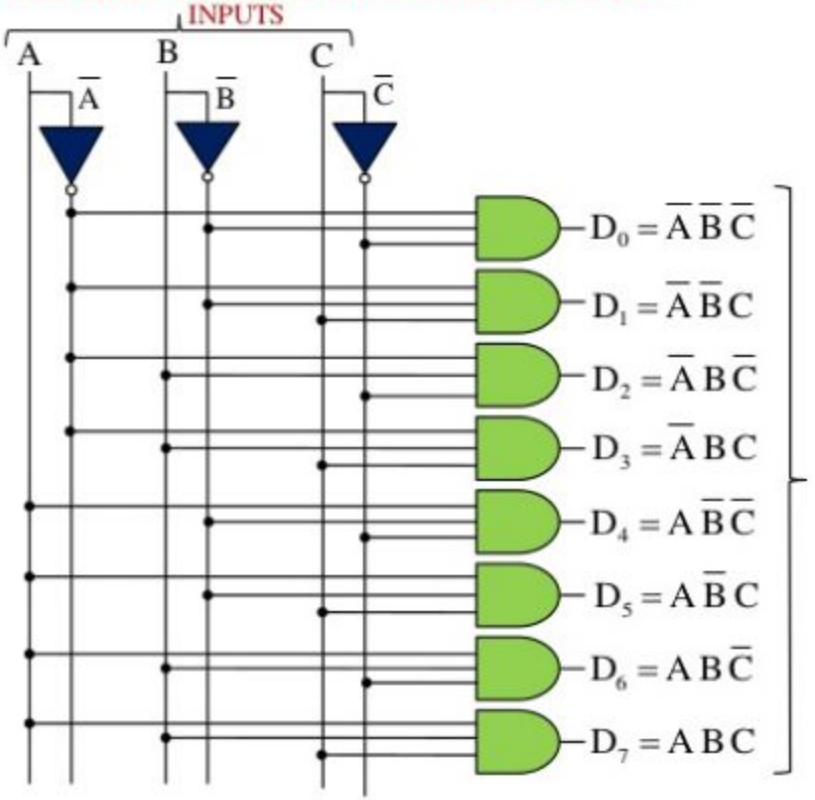
- Block diagram of 3 to 8 decoder is shown in fig. 4
- A , B and C are the inputs. (No. of inputs =3)
- No. of possible input combinations: $2^3=8$
- No. of Outputs : $2^3=8$, they are indicated by D_0 to D_7
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.



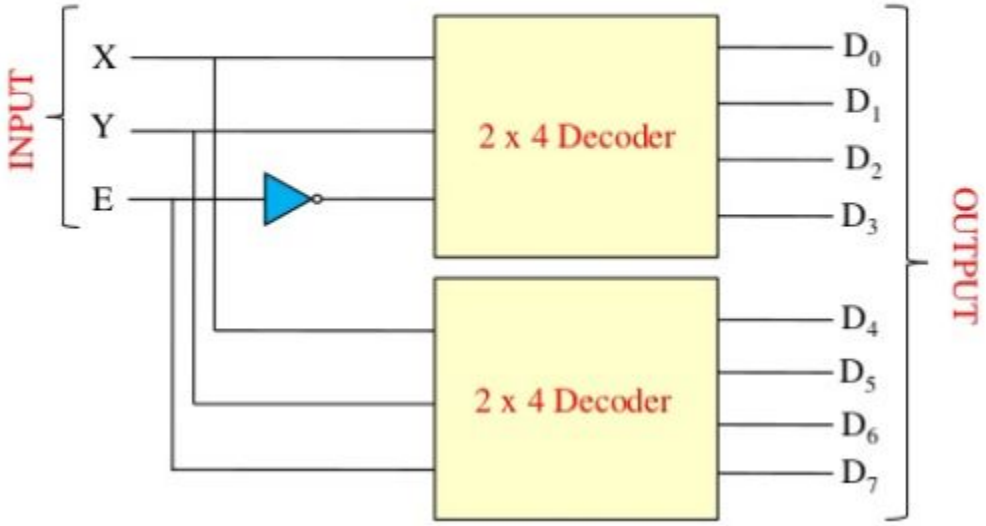
TRUTH TABLE FOR 3 X 8 DECODER:

INPUTS			OUTPUTS								
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7	
0	0	0	1	0	0	0	0	0	0	0	$D_0=\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1=\overline{A}\,\overline{B}C$
0	1	0	0	0	1	0	0	0	0	0	$D_2=\overline{A}B\overline{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3=\overline{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$D_4=A\overline{B}\,\overline{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5=A\overline{B}C$
1	1	0	0	0	0	0	0	0	1	0	$D_6=AB\overline{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7=ABC$

LOGIC DIAGRAM OF 3 X 8 DECODER:



3 x 8 Decoder From 2 x 4 Decoder:



4 to 16 decoder using 3 to 8 Decoder

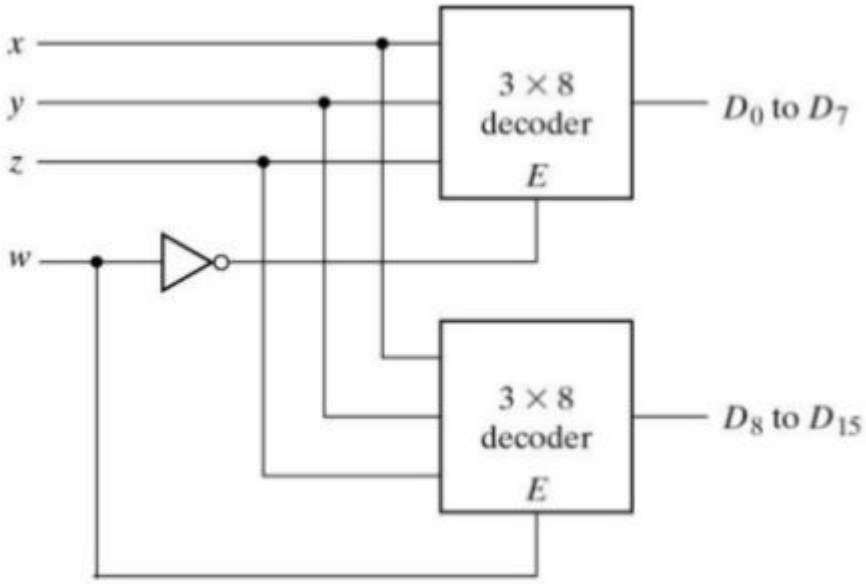


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

EXAMPLE: Implement the following Boolean function using suitable Decoder.

$f_1(x,y,z)=\sum m(1,5,7)$

$f_2(x,y,z)=\sum m(0,3)$

$f_3(x,y,z)=\sum m(2,4,5)$

Solution:

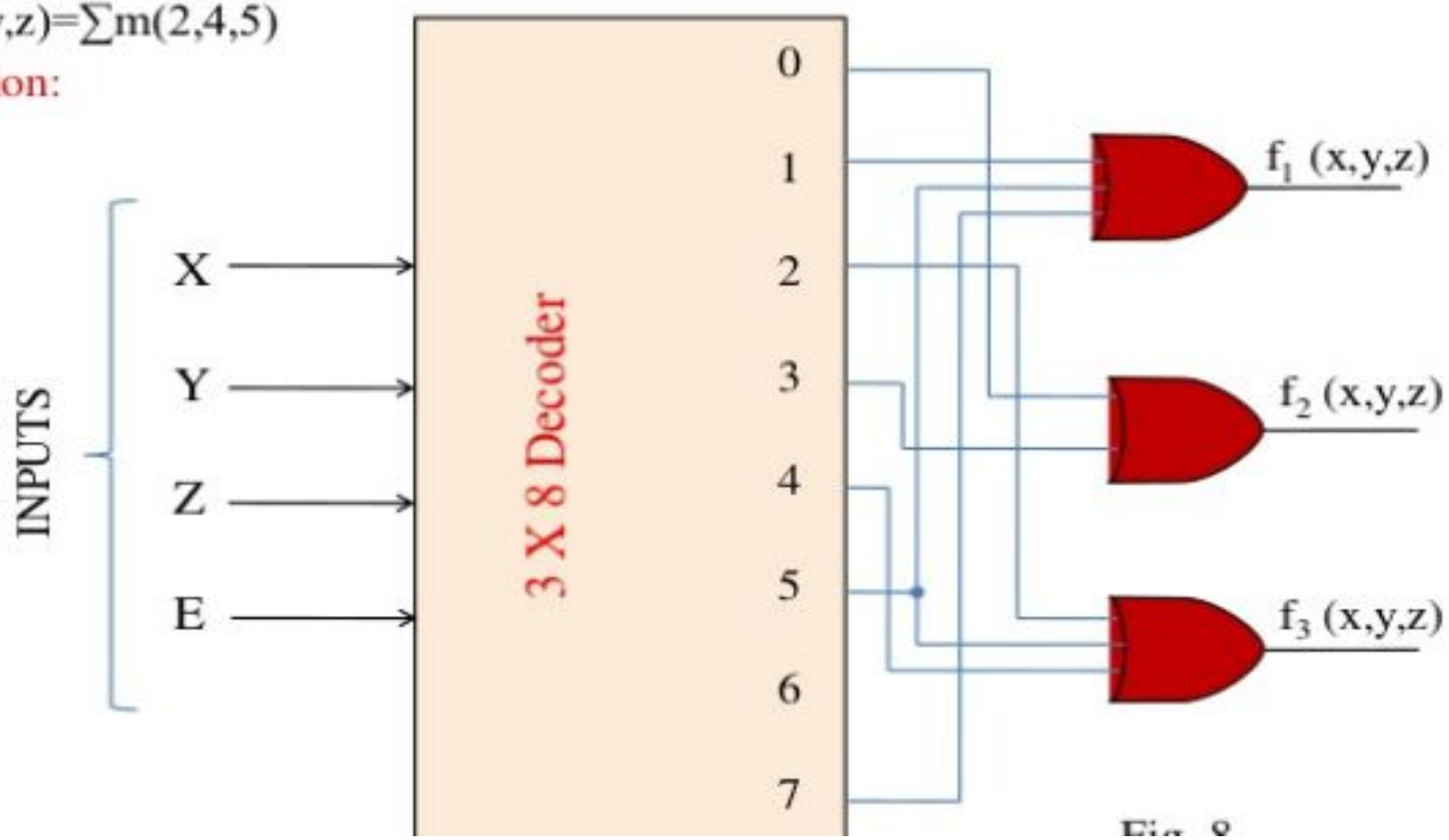


Fig. 8

Example: Implement the following multiple output function using a suitable Decoder.

$f_1(A, B, C) = \sum m(0,4,7) + d(2,3)$

$f_2(A, B, C) = \sum m(1,5,6)$

$f_3(A, B, C) = \sum m(0,2,4,6)$

Solution: f1 consists of don't care conditions. So we consider them to be logic 1.

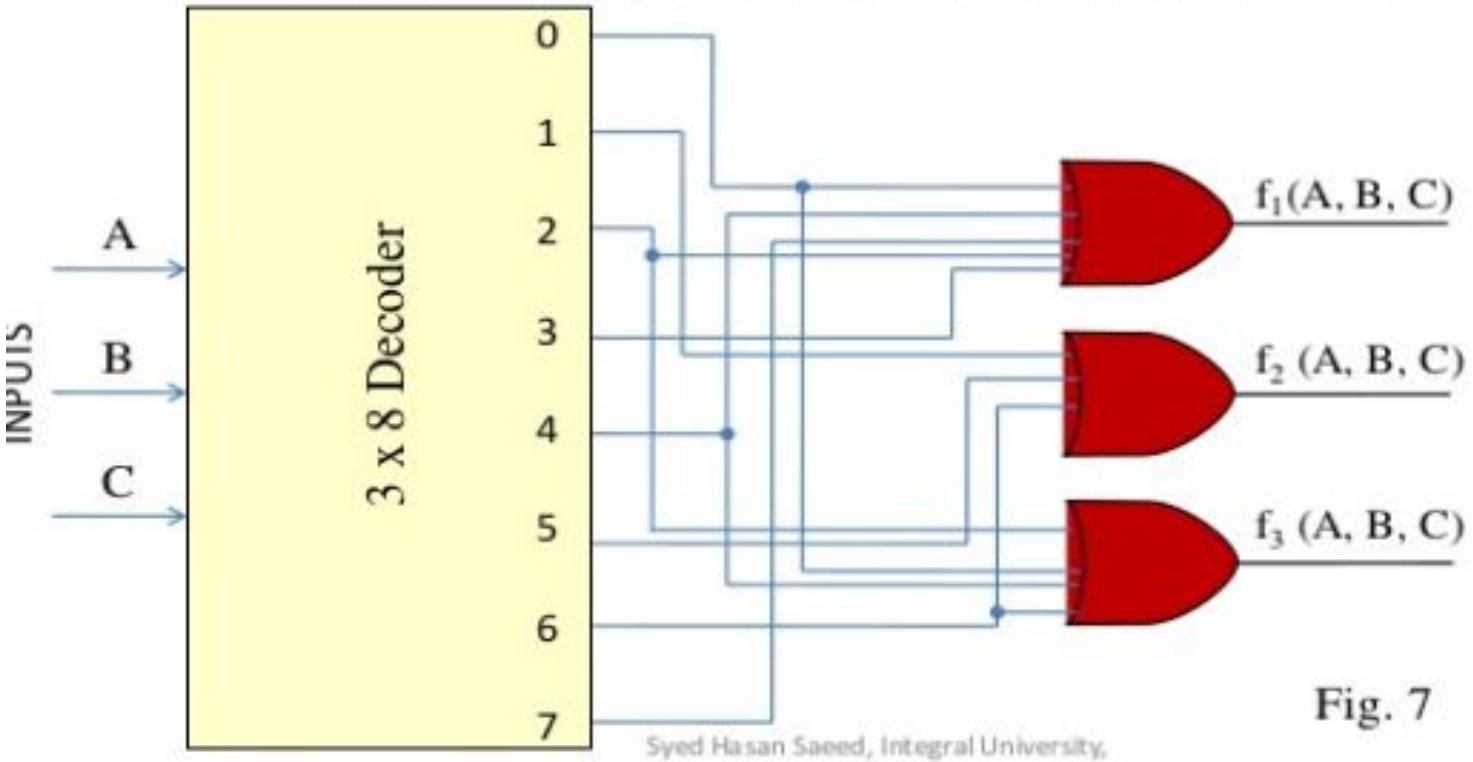
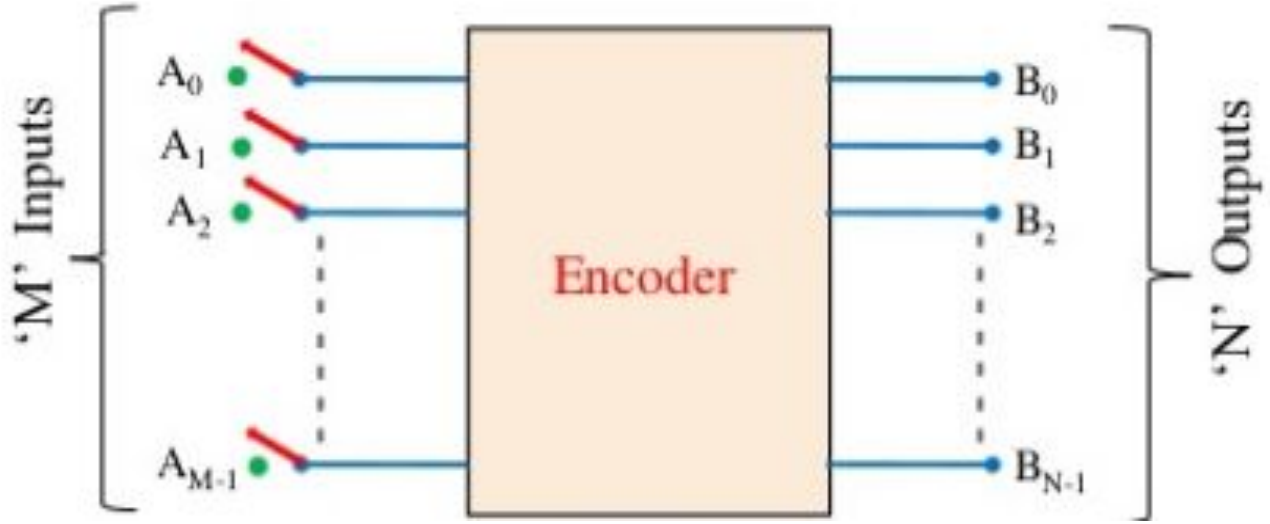


Fig. 7

ENCODER

- An Encoder is a combinational logic circuit.
- It performs the inverse operation of Decoder.
- The opposite process of decoding is known as Encoding.
- An Encoder converts an active input signal into a coded output signal.
- Block diagram of Encoder is shown in Fig.10. It has 'M' inputs and 'N' outputs.
- An Encoder has 'M' input lines, only one of which is activated at a given time, and produces an N-bit output code, depending on which input is activated.

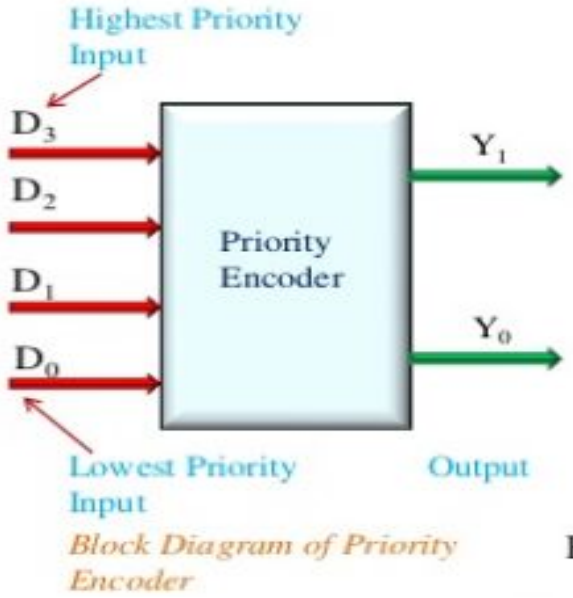


- Encoders are used to translate the rotary or linear motion into a digital signal.
- The difference between Decoder and Encoder is that Decoder has Binary Code as an input while Encoder has Binary Code as an output.
- Encoder is an Electronics device that converts the analog signal to digital signal such as BCD Code.
- Types of Encoders

i. Priority Encoder
ii. Decimal to BCD Encoder
iii. Octal to Binary Encoder
iv. Hexadecimal to Binary Encoder

PRIORITY ENCODER:

- As the name indicates, the priority is given to inputs line.
- If two or more input lines are high at the same time i.e 1 at the same time, then the input line with high priority shall be considered.
- Block diagram and Truth table of Priority Encoder are shown in fig.15



Block Diagram of Priority Encoder

Fig.15

TRUTH TABLE:

INPUTS				OUTPUTS		V
D_3	D_2	D_1	D_0	Y_1	Y_0	
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

- There are four inputs D_0, D_1,D_2, D_3 and two outputs Y_1 and Y_2.
- D_3 has highest priority and D_0 is at lowest priority.
- If D_3=1 irrespective of other inputs then output Y_1Y_0=11.
- D_3 is at highest priority so other inputs are considered as don't care.

K-map for Y_1 and Y_0

D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$Y_1 = D_2 + D_3$

D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$Y_0 = D_3 + \overline{D_2}\, D_1$

LOGIC DIAGRAM OF PRIORITY ENCODER:

$$Y_1 = D_2 + D_3$$

$$Y_0 = D_3 + \overline{D_2}\, D_1$$

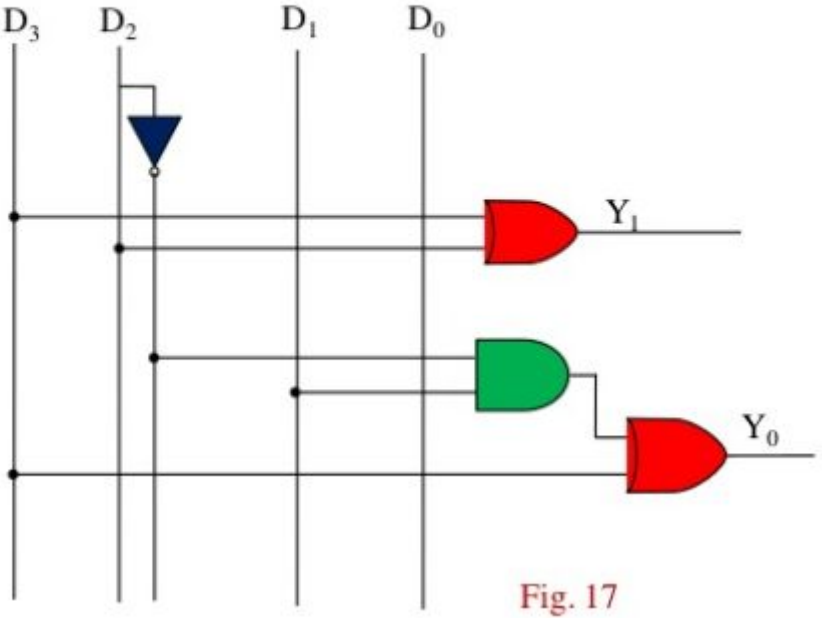


Fig. 17

DECIMAL TO BCD ENCODER:

- It has ten inputs corresponding to ten decimal digits (from 0 to 9) and four outputs (A,B,C,D) representing the BCD.
- The block diagram is shown in fig.18 and Truth table in fig.19

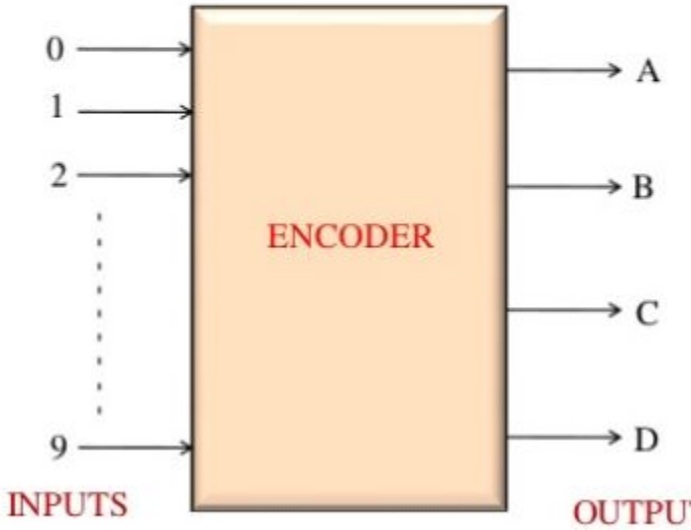


Fig. 18

Truth table:

INPUTS										BCD OUTPUTS			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

- From Truth Table it is clear that the output A is HIGH when input is 8 OR 9 is HIGH

  Therefore A=8+9
- The output B is HIGH when 4 OR 5 OR 6 OR 7 is HIGH

  Therefore B=4+5+6+7
- The output C is HIGH when 2 OR 3 OR 6 OR 7 is HIGH

  Therefore C=2+3+6+7
- Similarly D=1+3+5+7+9

  Logic Diagram is shown in fig.20

DECIMAL TO BCD ENCODER

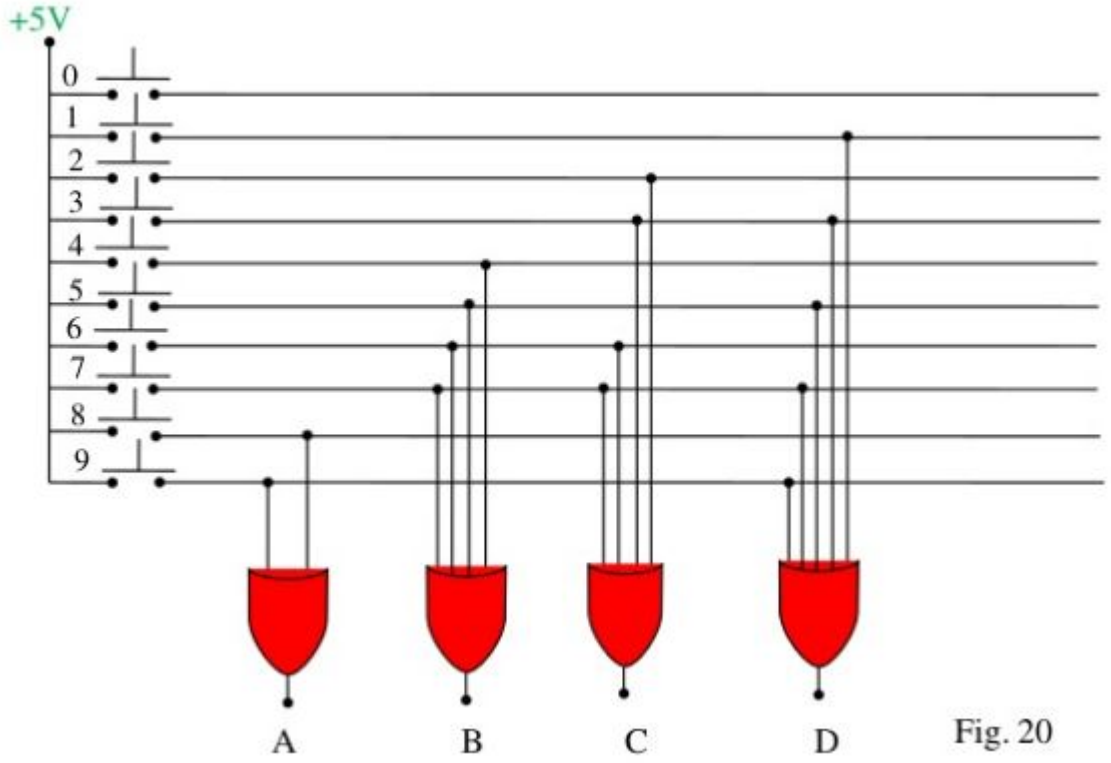


Fig. 20

OCTAL TO BINARY ENCODER:

- Block Diagram of Octal to Binary Encoder is shown in Fig. 21
- It has eight inputs and three outputs.
- Only one input has one value at any given time.
- Each input corresponds to each octal digit and output generates corresponding Binary Code.

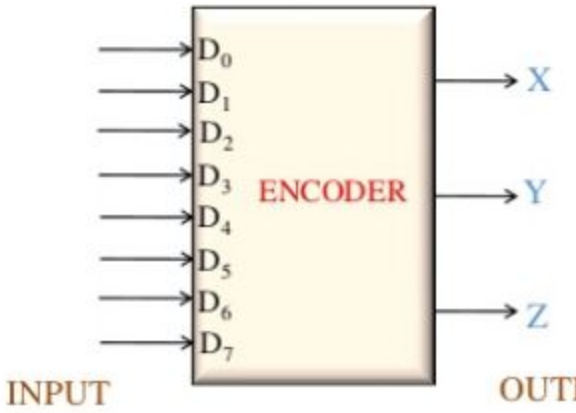


Fig. 21

INPUT								OUTPUT		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
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

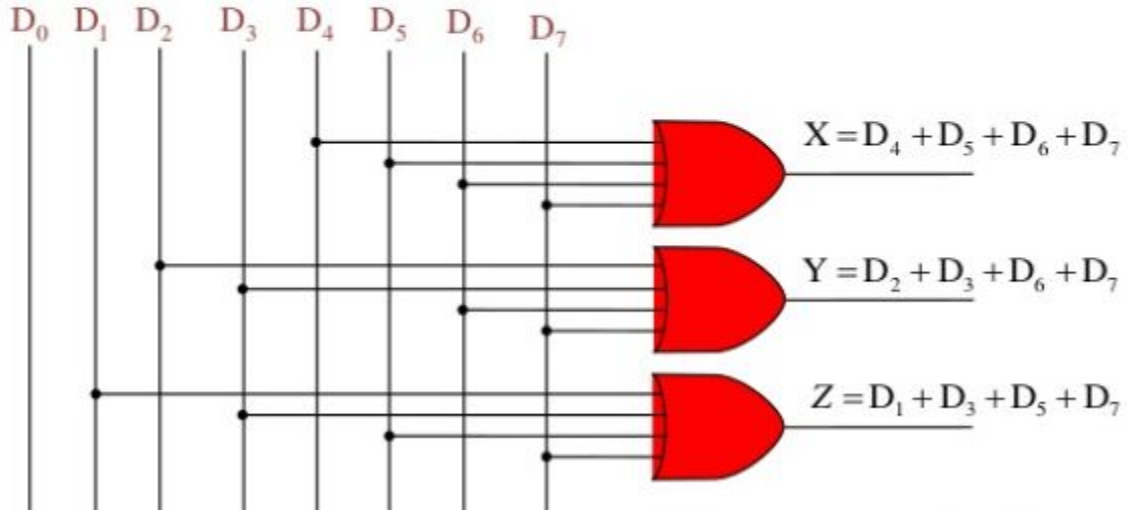
From Truth table:

$$X = D_4 + D_5 + D_6 + D_7$$
$$Y = D_2 + D_3 + D_6 + D_7$$
$$Z = D_1 + D_3 + D_5 + D_7$$

- It is assume that only one input is HIGH at any given time. If two outputs are HIGH then undefined output will produced. For example D_3 and D_6 are HIGH, then output of Encoder will be 111. This output neither equivalent code corresponding to D_3 nor to D_6.
- To overcome this problem, priorities should be assigned to each input.
- Form the truth table it is clear that the output X becomes 1 if any of the digit D_4 or D_5 or D_6 or D_7 is 1.
- D_0 is considered as don't care because it is not shown in expression.
- If inputs are zero then output will be zero. Similarly if D_0 is one, the output will be zero.

$$X = D_4 + D_5 + D_6 + D_7$$
$$Y = D_2 + D_3 + D_6 + D_7$$
$$Z = D_1 + D_3 + D_5 + D_7$$

LOGIC DIAGRAM:



Multiplexer (data Selectors)

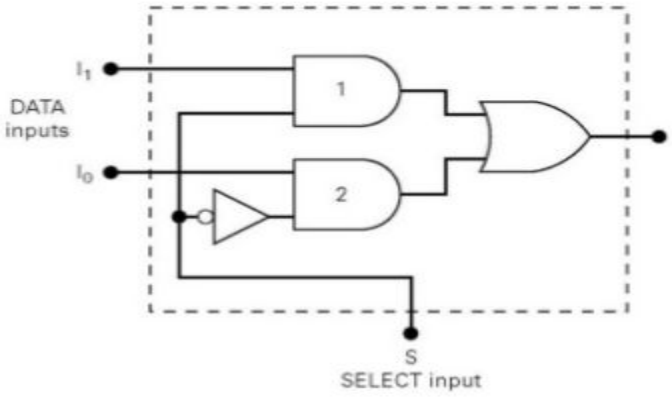
- Definition : A multiplexers (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.

- Several data input lines
- Some select line (less than the no. of input lines)
- Single output line

- If there are n data input lines and m select lines, then

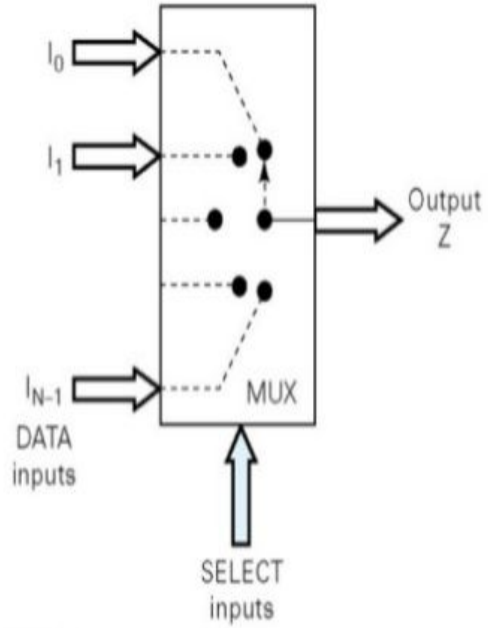
$$2^m = n$$

2 : 1 Multiplexer

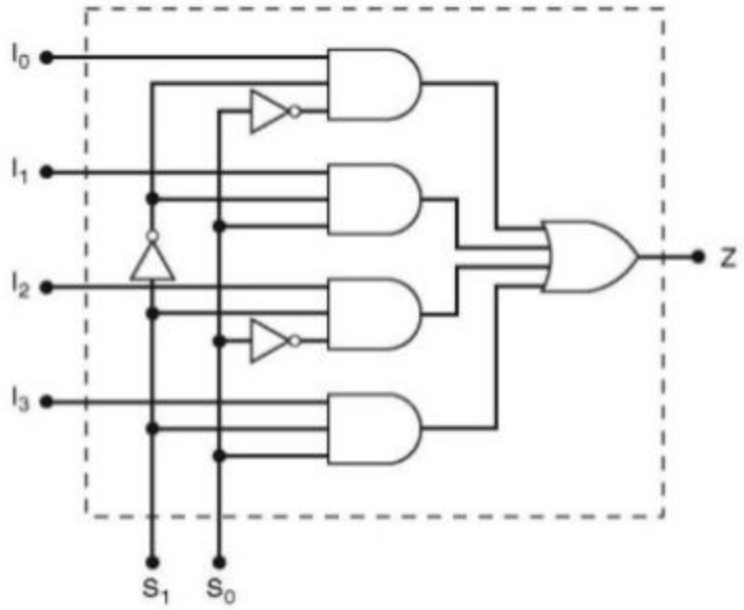


S	Z
0	I_0
1	I_1

Functional Diagram Of a Multiplexer



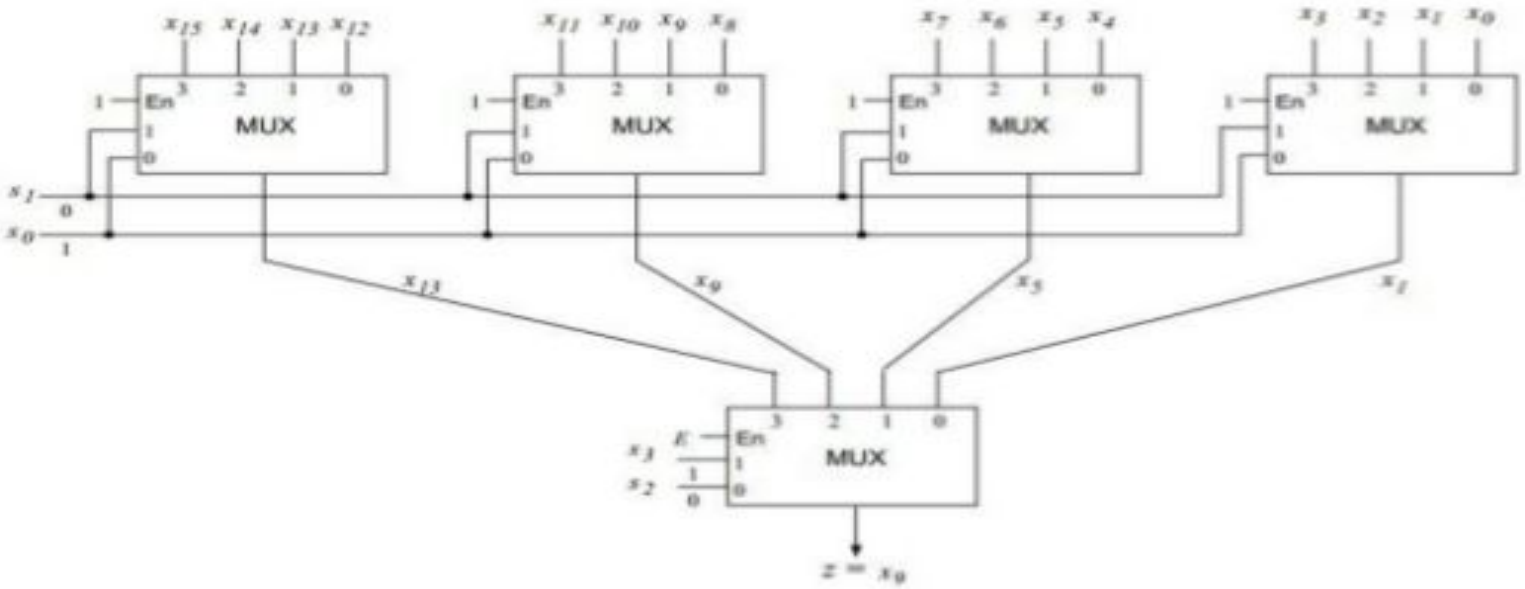
4 : 1 Multiplexer



S_0	S_1	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

16:1 Mux using 4:1 Mux

- The Multiplexers with more number of inputs can be obtained by cascading two or more multiplexers with less number of inputs.
- Below is a design of 16:1 MUX using 4 4:1 MUXs :-



Demultiplexer (Data Distributor)

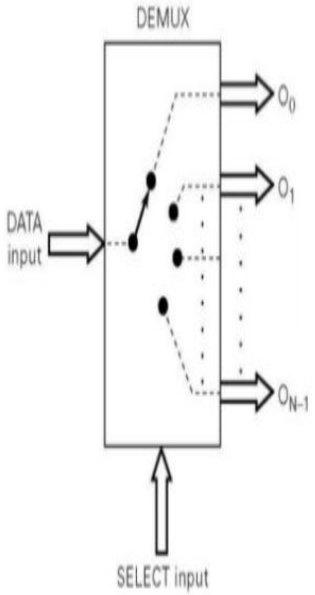
- Definition : A DEMULTIPLEXER (DEMUX) basically reverses the multiplexing function. It takes data from one line and distributes them to a given number of output lines. For this reason, the demultiplexers is also known as a data distributor.

- Single data input lines
- Some select line (less than the no. of output lines)
- Several output line

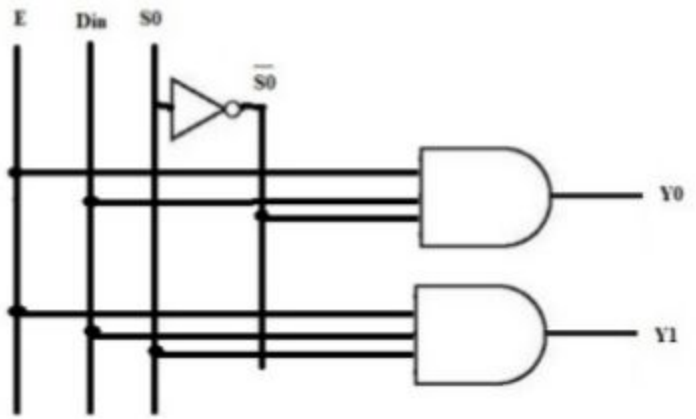
- If there are n data output lines and m select lines, then

$$2^m = n$$

Functional Diagram Of a Demultiplexer

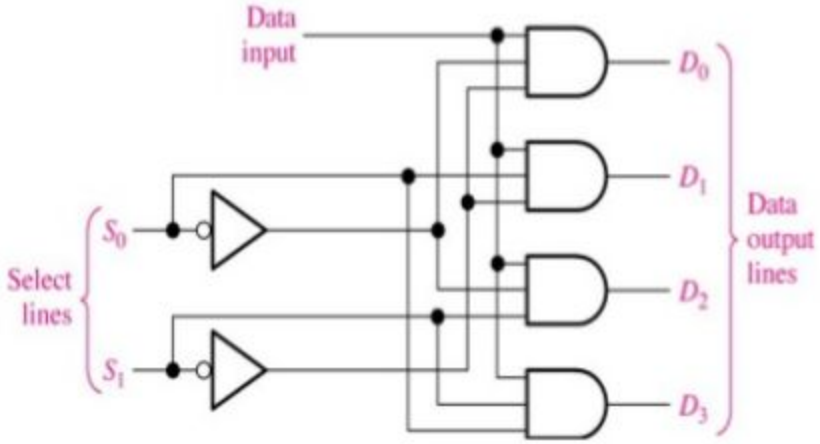


1 : 2 Demultiplexer



S_0	Y_0	Y_1
0	D	0
1	0	D

1 : 4 Demultiplexer



S_0	S_1	D_0	D_1	D_2	D_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

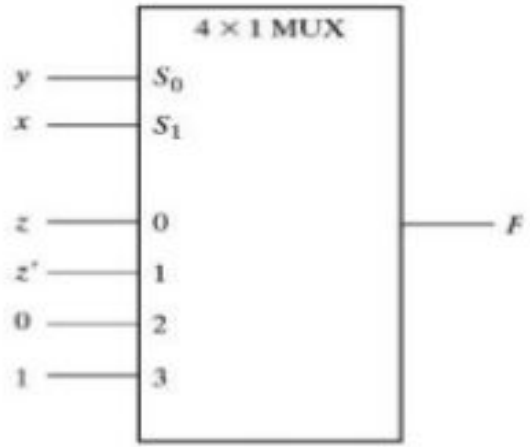
Boolean function implementation using Mux

- A more efficient method for implementing a Boolean function of n variables with a **multiplexer**.

$$F(x, y, z) = \Sigma(1,2,6,7)$$

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



(b) Multiplexer implementation

Fig. 4-27 Implementing a Boolean Function with a Multiplexer

$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

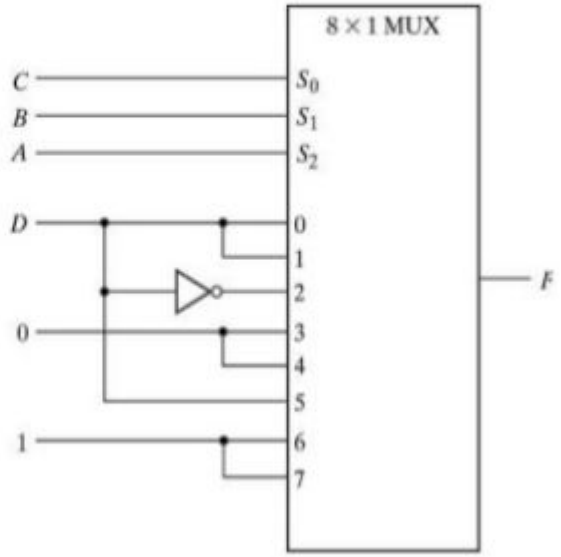
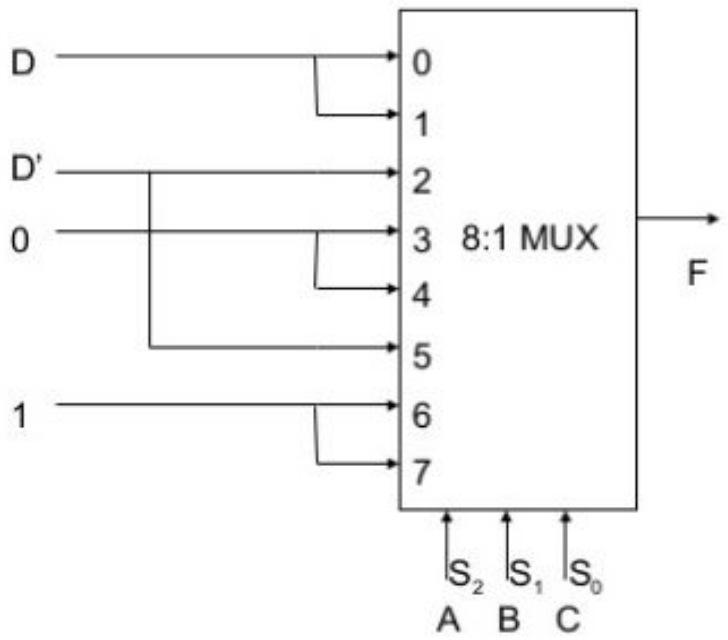


Fig. 4-28 Implementing a 4-Input Function with a Multiplexer

f(a, b, c) = F=A'B'C'D + A'B'CD + A'BC'D' + AB'CD + ABC'D' + ABC'D + ABCD' +ABCD

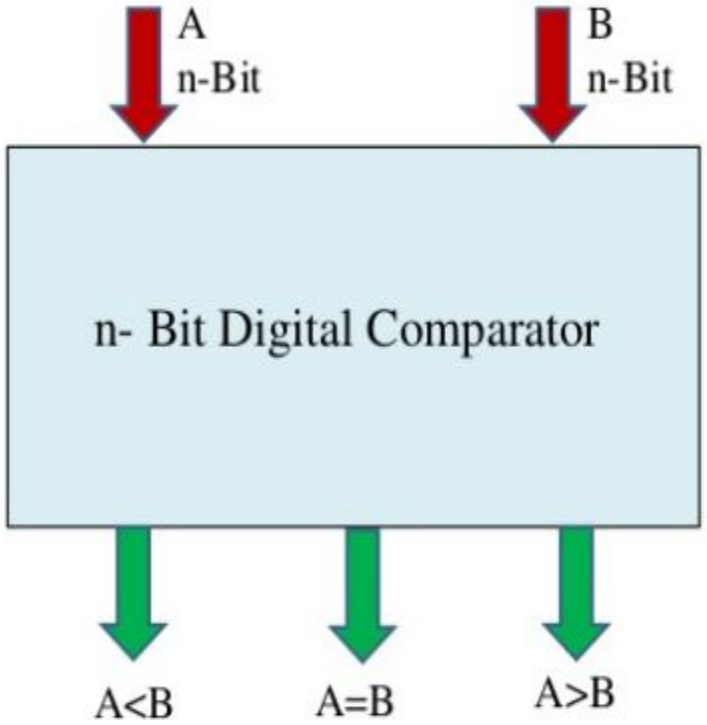
A	B	C	D	O	F
0	0	0	0	0	D
0	0	0	1	1	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	1	D'
0	1	0	1	0	
0	1	1	0	0	0
0	1	1	1	0	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	0	D'
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	1	1
1	1	1	1	1	



Magnitude comparator

- It is a combinational logic circuit.
- Digital Comparator is used to compare the value of two binary digits.
- There are two types of digital comparator (i) Identity Comparator (ii) Magnitude Comparator.
- IDENTITY COMPARATOR: This comparator has only one output terminal for when A=B, either A=B=1 (High) or A=B=0 (Low)
- MAGNITUDE COMPARATOR: This Comparator has three output terminals namely A>B, A=B, A<B. Depending on the result of comparison, one of these output will be high (1)
- Block Diagram of Magnitude Comparator is shown in Fig. 1

BLOCK DIAGRAM OF MAGNITUDE COMPARATOR



1- Bit Magnitude Comparator:

- This magnitude comparator has two inputs A and B and three outputs A<B, A=B and A>B.
- This magnitude comparator compares the two numbers of single bits.
- Truth Table of 1-Bit Comparator

INPUTS		OUTPUTS		
A	B	Y_1 (A<B)	Y_2 (A=B)	Y_3 (A>B)
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-Maps For All Three Outputs :

A \ B	$\overline{B}$ 0	B 1
$\overline{A}$ 0	0	(1)
A 1	0	0

K-Map for Y_1 : A<B

$Y_1 = \overline{A}B$

A \ B	$\overline{B}$ 0	B 1
$\overline{A}$ 0	(1)	0
A 1	0	(1)

K-Map for Y_2 : A=B

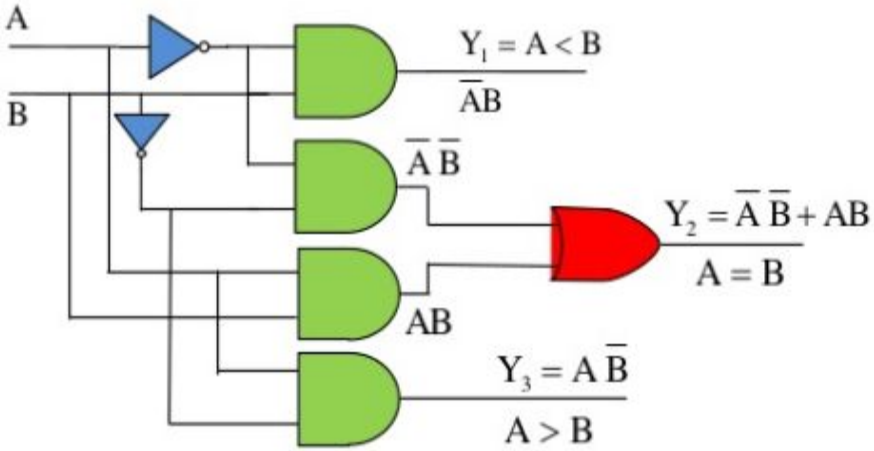
$Y_2 = \overline{A}\,\overline{B} + AB$

A \ B	$\overline{B}$ 0	B 1
$\overline{A}$ 0	0	0
A 1	(1)	0

K-Map for Y_2 : A>B

$Y_3 = A\overline{B}$

Realization of One Bit Comparator

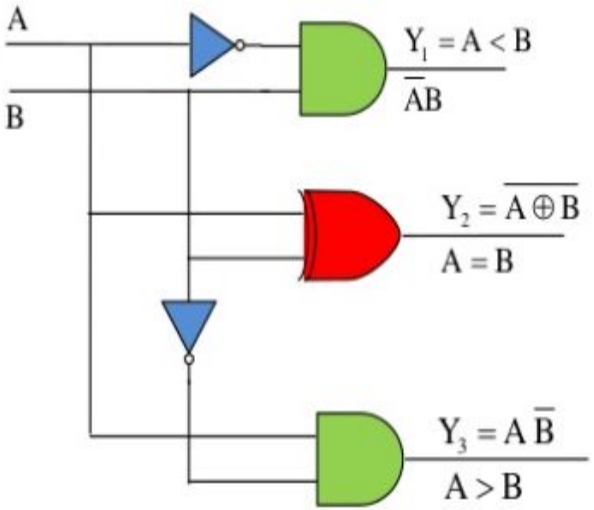


$Y_1 = \overline{A}B$

$Y_2 = \overline{\overline{A}B} + AB$

$Y_3 = A\overline{B}$

Realization of by Using AND , EX-NOR gates



2-bit magnitude comparator

- A comparator which is used to compare two binary numbers each of two bits is called a 2-bit magnitude comparator.
- Fig. 2 shows the block diagram of 2-Bit magnitude comparator.
- It has four inputs and three outputs.
- Inputs are A_0 ,A_1,B_0 and B_1 and Outputs are Y_1, Y_2 and Y_3

GREATER THAN (A>B)

A_1	A_0	B_1	B_0
1	0	0	1
1	1	1	0
0	1	0	0

1. If $A_1 = 1$ and $B_1 = 0$ then A>B
2. If A_1 and B_1 are same, i.e $A_1=B_1=1$ or $A_1=B_1=0$ and $A_0=1$, $B_0=0$ then A>B

LESS THAN (A<B)

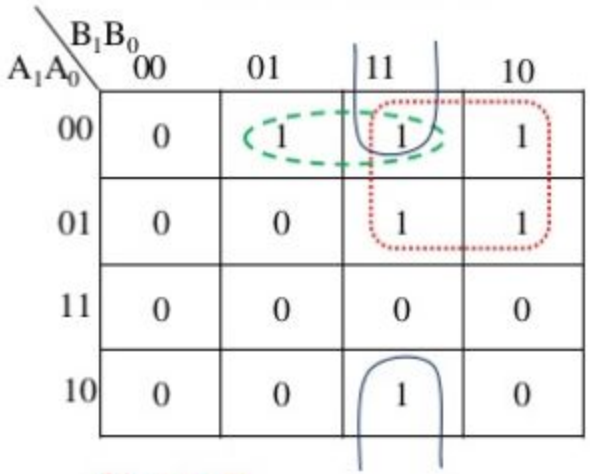
Similarly,

1. If $A_1 = B_1=1$ and $A_0 = 0$, $B_0=1$, then A<B
2. If $A_1 = B_1 = 0$ and $A_0 = 0$, $B_0=1$ then A<B

TRUTH TABLE

INPUT				OUTPUT		
A_1	A_0	B_1	B_0	Y_1=A<B	Y_2=(A=B)	Y_3=A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

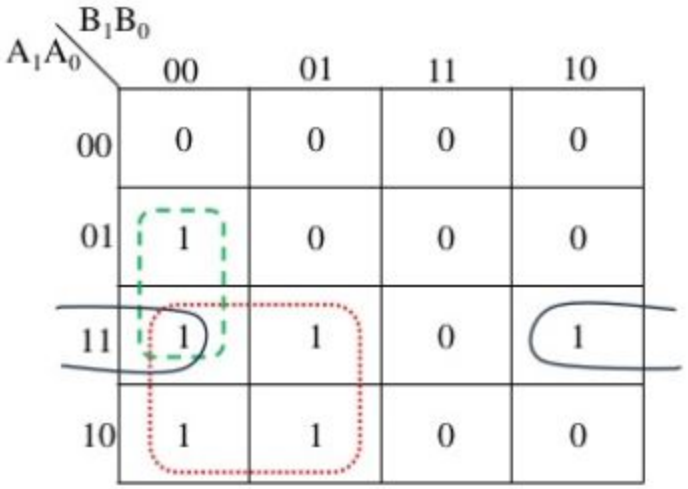
K-Map for A<B:



For A<B

$$Y_1 = \overline{A_1}\,\overline{A_0}\,B_0 + \overline{A_1}\,B_1 + \overline{A_0}\,B_1\,B_0$$

K-Map For A>B



$$Y_3 = A_0\,\overline{B_1}\,\overline{B_0} + A_1\,\overline{B_1} + A_1A_0\,\overline{B_0}$$

K-Map for A=B:

A_1A_0 \ B_1B_0	00	01	11	10
00	(1)	0	0	0
01	0	(1)	0	0
11	0	0	(1)	0
10	0	0	0	(1)

For A=B

$$Y_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + A_1A_0B_1B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

For A=B From K-Map

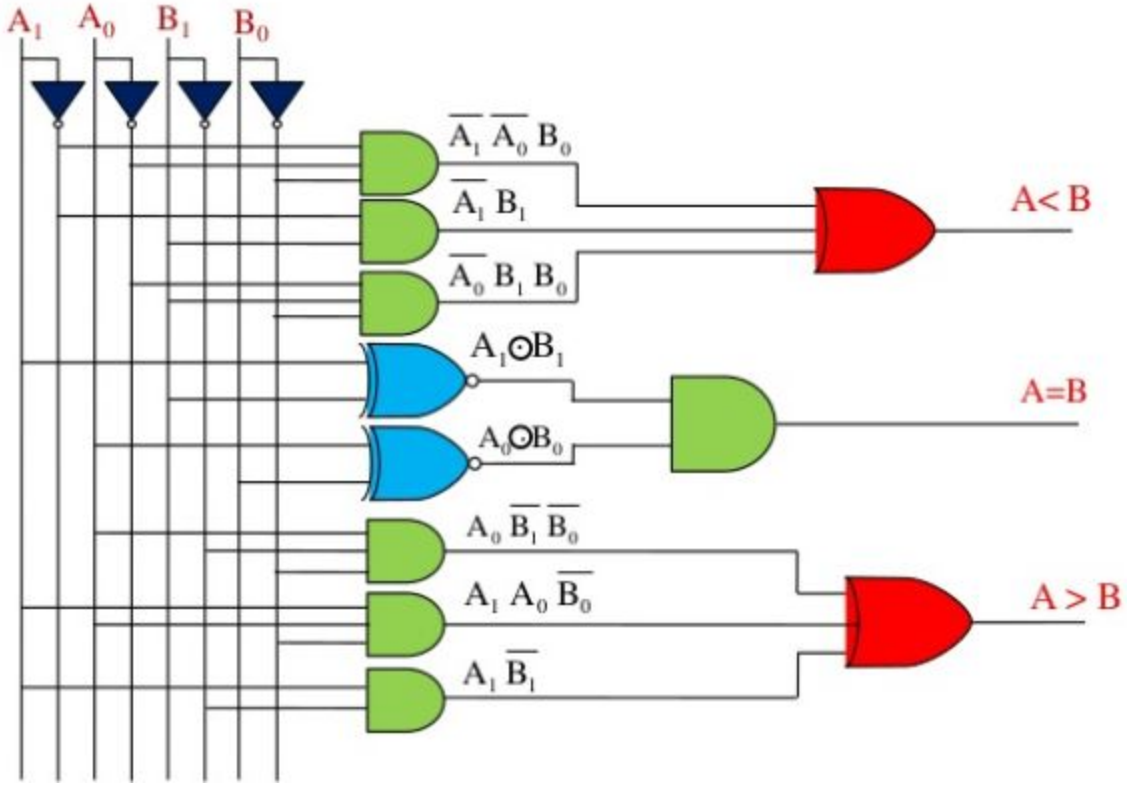
$$Y_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + A_1A_0B_1B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

$$Y_2 = \overline{A_0}\,\overline{B_0}(\overline{A_1}\,\overline{B_1} + A_1B_1) + A_0B_0(\overline{A_1}\,\overline{B_1} + A_1B_1)$$

$$Y_2 = (\overline{A_1}\,\overline{B_1} + A_1B_1)\;(\overline{A_0}\,\overline{B_0} + A_0B_0)$$

$$Y_2 = (A_1 \odot B_1)\,(A_0 \odot B_0)$$

LOGIC DIAGRAM OF 2-BIT COMPARATOR:



Parity Generator

What is Parity Generator?

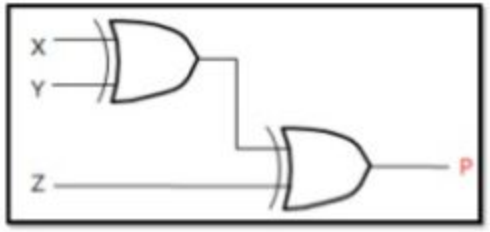
- A **Parity Generator** is a Combinational Logic Circuit that Generates the Parity bit in the Transmitter.
- A Parity bit is used for the Purpose of Detecting Errors during Transmissions of binary Information.
- It is an Extra bit Included with a binary Message to Make the Number of 1's either Odd or Even.

Two Types of Parity

- In **Even** Parity, the added Parity bit will Make the Total Number of 1's an Even Amount.
- In **Odd** Parity, the added Parity bit will Make the Total Number of 1's an Odd Amount.

Parity generator truth table and logic diagram

3-bit Message			Odd Parity Bit	Even Parity Bit
X	Y	Z		
0	0	0	1	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	0	1



Boolean Expression

Even Pair

$$P=\bar{X}\bar{Y}Z+\bar{X}Y\bar{Z}+X\bar{Y}\bar{Z}+XYZ$$
$$=\bar{X}(\bar{Y}Z+Y\bar{Z})+X(\bar{Y}\bar{Z}+YZ)$$
$$=\bar{X}(Y\oplus Z)+X(\overline{Y\oplus Z})$$
$$=X\oplus(Y\oplus Z)$$

Odd Pair

$$P=\bar{X}\bar{Y}\bar{Z}+\bar{X}YZ+X\bar{Y}Z+XY\bar{Z}$$
$$=\bar{X}(\bar{Y}\bar{Z}+YZ)+X(\bar{Y}Z+Y\bar{Z})$$
$$=\bar{X}(\overline{Y\oplus Z})+X(Y\oplus Z)$$
$$=\bar{X}\oplus(Y\oplus Z)$$

K-Map Simplification

X \ YZ	00	01	11	10
0	0	1	0	1
1	1	0	1	0

X \ YZ	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Code converter

1.Design of 4-bit binary to gray code converter

4-BIT BINARY				4-BIT GRAY			
B4	B3	B2	B1	G4	G3	G2	G1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-MAP

FOR G4 (G4=B4)

B2B1 / B4B3	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

FOR G3 (G3= $\overline{B4}.B3+B3.\overline{B4}$ B4⊕B3)

B2B1 / B4B3	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

K-MAP

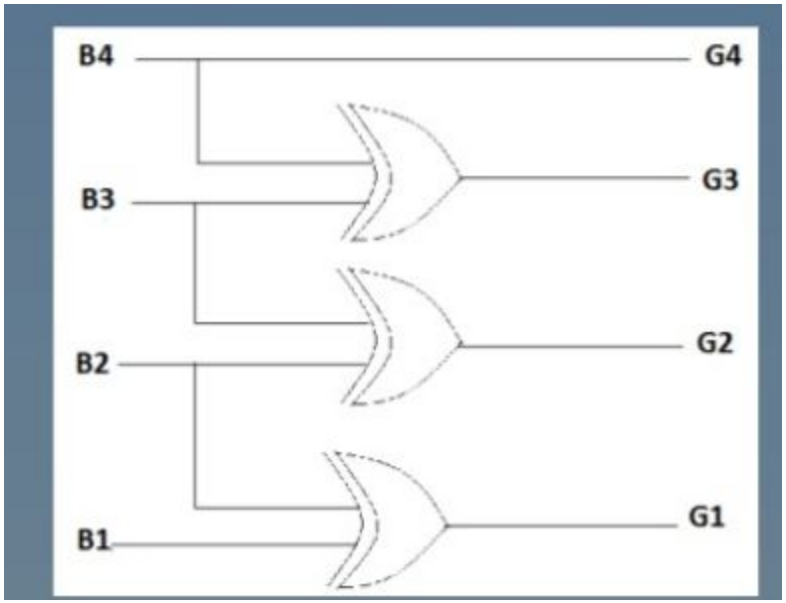
FOR G2 (G2=$\overline{B2}.B3+\overline{B3}.B2$ B2 ⊕B3)

B2B1 / B4B3	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

FOR G1 (G1= $\overline{B2}.B3+\overline{B3}.B2$ B2 ⊕B1)

B2B1 / B4B3	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Circuit diagram



BCD to Excess-3 code converter

4-BIT BCD				4-BIT EXCESS-3			
B4	B3	B2	B1	Y4	Y3	Y2	Y1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

K-MAP

FOR Y4= B4+B3.B1+B3.B2

B2B1 / B4B3	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

FOR Y3=$\overline{B3}$.B1+$\overline{B3}$.B2+B3.$\overline{B2}$.$\overline{B1}$

B2B1 / B4B3	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

K-MAP

FOR Y2= $\overline{B2}$.B1+B2.$\overline{B1}$

B2B1 / B4B3	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

FOR Y1=$\overline{B2}$.$\overline{B1}$+B2.$\overline{B1}$
=$\overline{B1}$

B2B1 / B4B3	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

Circuit Diagram