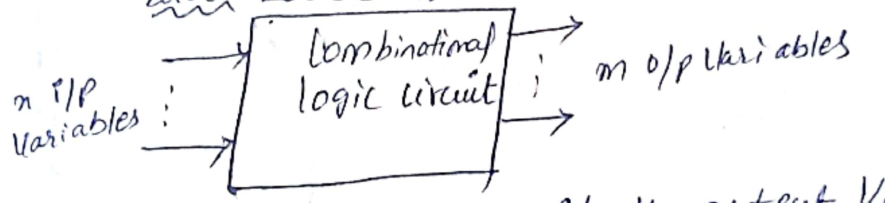
UNIT-3

✱ COMBINATIONAL LOGIC ✱

→ When logic gates are connected together to produce a specified combinations of input variables with no storage involved, the resulting circuit is called combinational logic circuit.

→ Combinational circuit may have 'n' input variables and 'm' output variables.

Block diagram of a combinational circuit



→ In combinational logic circuit, the output variables are at all times dependent on the combinations of input variables.

→ Circuit :- It is closed path in the combination of active & passive element

→ logic circuit :- It is closed path in the combinations of logic gates

→ Combinational circuits are used in many Applications

(1) Arithmetic and Boolean operations eg : Arithmetic.
(2) Digital signal processing eg Binary multipliers
(3) Computers eg to perform operations on data.
(4) Digital calculators
(5) Robotics and digital communications.

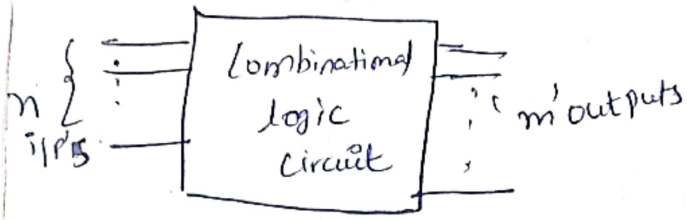
Logic circuit

Combinational

→ It has 'n' number of inputs and 'm' number of outputs

→ Output depends on present input

→ Time delay is less



→ No feedback

→ eg:

Examples

1. Adders
2. Subtractors
3. Decoders
4. Encoders
5. Multiplexers
6. Demultiplexers
7. Multipliers
8. Comparators.

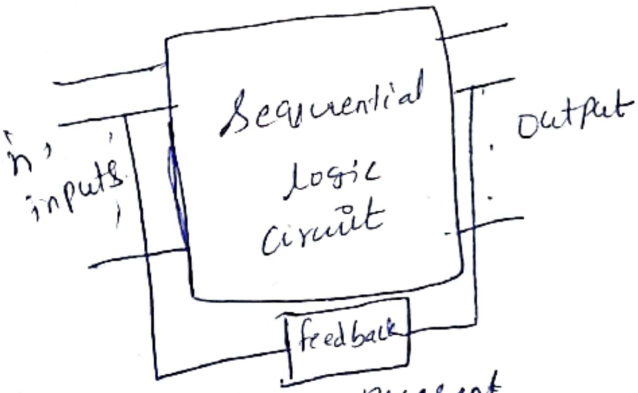
→ circuit is simple to construct.

→ Circuit complexity is less.

Sequential

→ It has 'n' number of inputs but output depends on present input and past output.

→ Time delay is more



→ Feedback present

→ Complex to design and construct

Examples :-

1. Registers
2. Counters
3. Circuits like serial binary adders

→ Circuit complexity is more.

Analysis procedure:-

Steps of Analysis of Combinational Circuit:

(1) First make sure that the given circuit is combinational circuit and not the sequential circuit. The combinational circuit has logic gates with no feedback path (or) memory elements

(2) Label the all gate outputs that are function of input variables with arbitary symbols, and determine the boolean functions for each gate output

(3) Label the gates that are a function of input variables and previously labelled gates and determine the boolean function for them.

(4) Repeat the step 3, until the boolean function for outputs of the circuit are obtained.

(5) finally, substituting previously defined boolean functions, obtain the output boolean functions interms of output variables.

~~Analysis procedure~~ and Design procedure : (3)

(A)

Designing steps for a combinational logic circuits

(1) observe / Analyse the required input and output Variables

(2) Determine the required input and output Variables

(3) Assign letters (or) symbols to the input Variables

(4) Make the table that defines required relationship

(5) Determine the simplified boolean expression using k-map

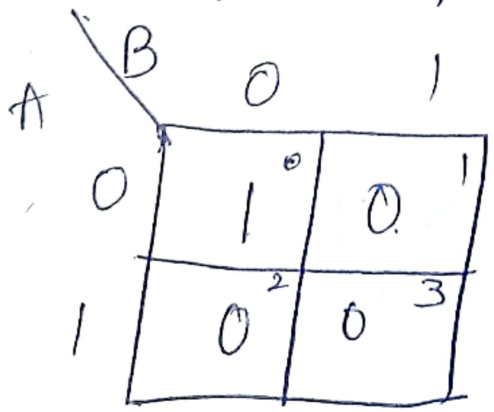
(6) Draw the logic diagram

Example :- Design a combinational circuit with two inputs, which produce output is logic '0' when any one input is one.

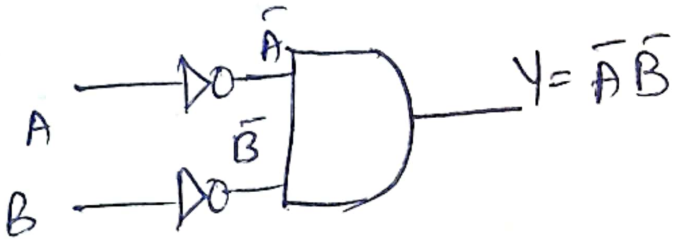
Two inputs ⇒ A, B output ⇒ Y.

$2^n = 2^2 = 4$ combinations

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



$Y = \bar{A}\bar{B}$



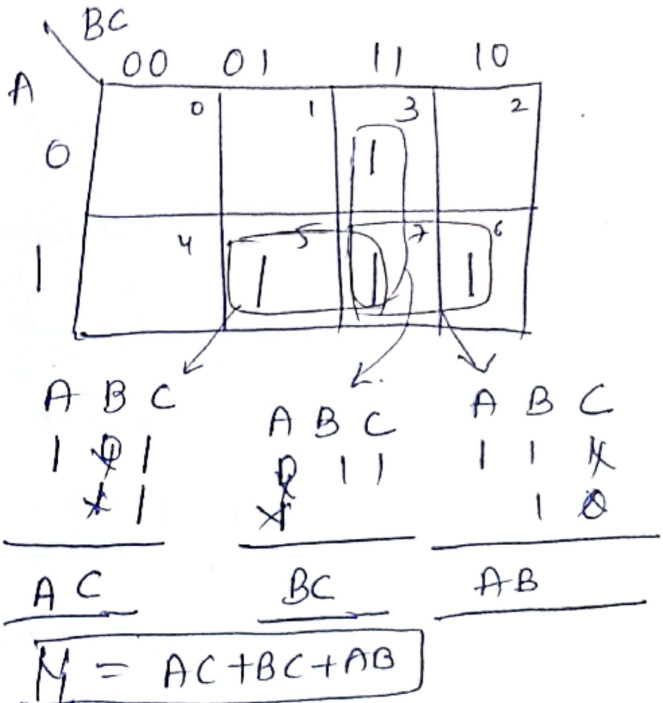
Example: Design a 3-bit combinational circuit which produces a logic output when more than one input variables are at logic 1.

Sol

Step 1: Derive the truth table for the given statement

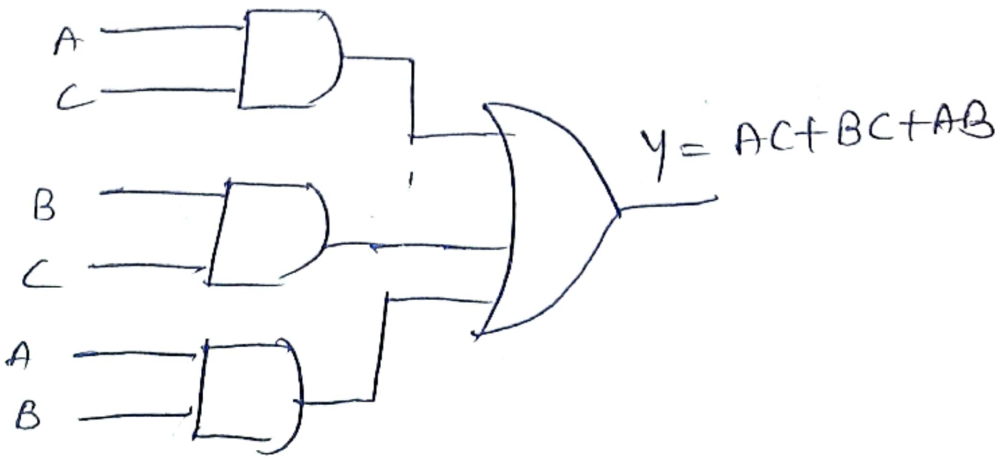
Step 2: Obtain simplified boolean expression.

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



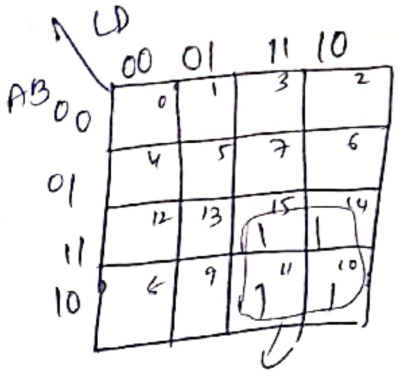
$Y = AC + BC + AB$

Step 3: Draw logic diagram



Example :- Design a combinational logic circuit with 4 inputs A, B, C, D. The output Y goes high if and only if A and C inputs are high. Draw the truth table. minimize the Boolean function using K-map. Draw the circuit diagram

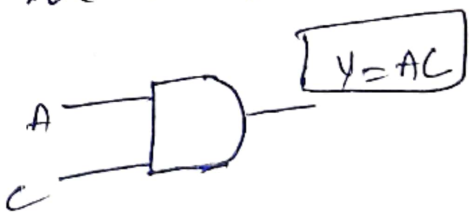
A	B	C	D	Y
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	0	
1	0	0	0	
1	0	0	1	
1	0	1	0	1
1	0	1	1	1
1	0	0	0	
1	1	0	1	
1	1	1	0	1
1	1	1	1	1



A B CD
1 X 1 X
1 0 1 0

AC

logic diagram



* Adders and Subtractors *

* Auithmetic Circuits *

* **Adders :-** The Basic operation in digital computer is Binary addition. The circuit which perform addition of binary bits are called as "Adder".

* The most basic operation, is the addition of two binary digits, the simple addition consists of four possible elementary operations namely

$$0+0 = 0$$

$$0+1 = 1$$

$$1+0 = 0$$

$$1+1 = (10)_2$$

* The first three operations procedure a sum whose length is one digit, but when the last operation is performed sum is two digits. The higher significant bit of this result is called 'carry', the lower significant bit is called 'sum'

→ the logic circuit which performs this operation is called "Half Adder"

*. Half-adder *

* The logic circuit which performs the addition of two bits is called a "Half adder"

* The half adder operation needs two binary inputs, "Augend" and "addend" bits, and the binary outputs are "sum" and "carry".

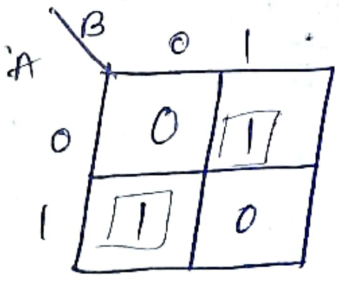
*

Input		output	
A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table for Half adder

* K-map Simplification for Sum and carry :

For Sum



A	B
1	0

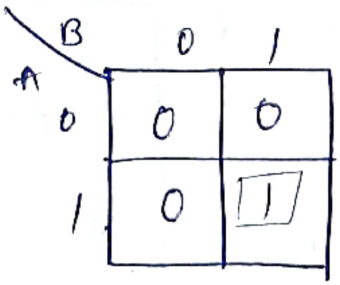
$A\bar{B}$

A	B
0	1

$\bar{A}B$

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

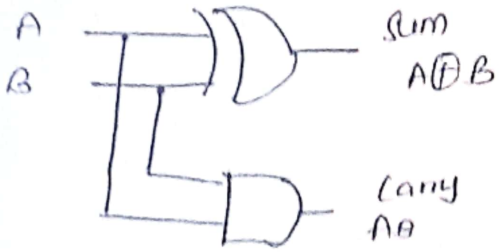
For carry



A	B
1	1

AB

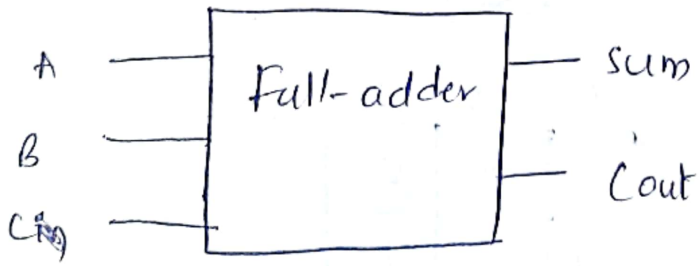
$Carry = AB$



logic diagram of Half adder

* full adder *

* A full adder is a combinational circuit that forms the arithmetic sum of three input bits.
* It consists of three inputs and two outputs.
* Two of the input variables denoted by A and B. Represent the two significant bits to be added. The third input C_{in} represents the carry form the previous lower significant position

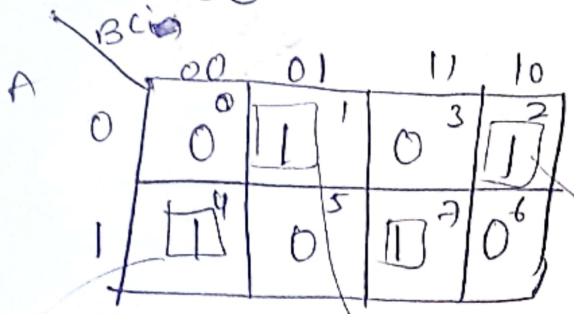


Block diagram of Full adder

Truth table for full adder

Input			output	
A	B	C_{in}	Sum	Carry
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

for sum



for carry

Sum (K-map)

A B C
1 0 0
A B̄ C̄

A B C
0 0 1
Ā B̄ C

A B C
1 1 1
A B C

A B C
0 1 0
Ā B C̄

Carry (K-map)

A B C
1 ⊗ 1
⊗ 1
AC

A B C
1 1 ⊗
1 ⊗
Ā B

A B C
⊗ 1 1
⊗
BC

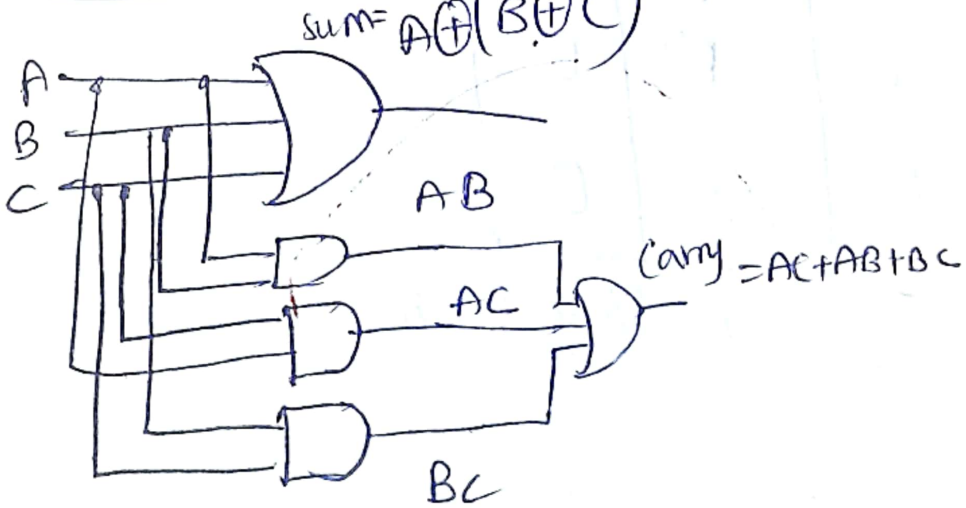
$F = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

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

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

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

$F = AC + AB + BC$



* Subtractor *

→ The subtraction consists of four possible elementary operation namely

$0-0=0$
$0-1=1$ with '1' borrow
$1-0=1$
$1-1=0$

→ In all operations, each subtrahend bit is subtracted from minuend bit

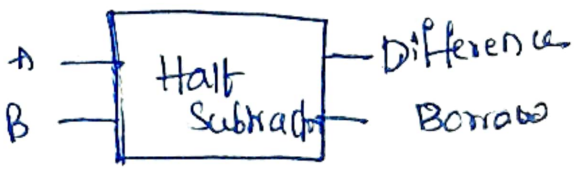
→ In case of second operation the minuend bit is smaller than the subtrahend bit, hence '1' is borrowed.

* Half Subtractor *

→ A Half subtractor is a combinational circuit that subtracts two bits and gives two outputs one is difference another one is Borrow

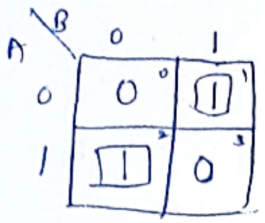
Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth table of Half subtractor



K-map Simplification

for difference



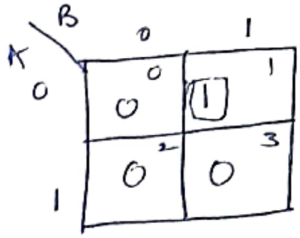
AB = 10 → $A\bar{B}$

AB = 01 → $\bar{A}B$

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

Difference = $A \oplus B$

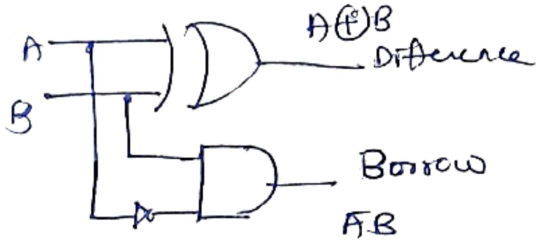
for Borrow



A B
0 1

$\bar{A}B$

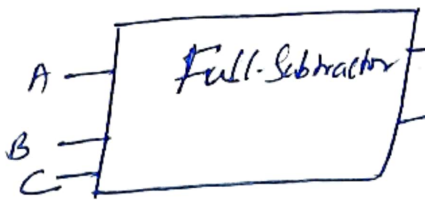
Borrow = $\bar{A}B$



Logic circuit for Half subtractor

* Full Subtractor *

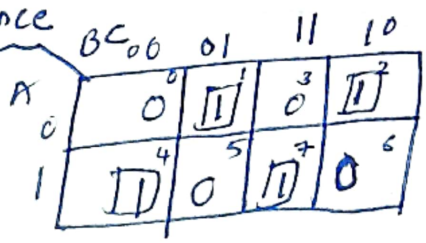
→ Full Subtractor is combinational logic circuit which is used to subtract three bits and produce two outputs difference and Borrow.



Inputs			outputs.	
A	B	C	Difference	Borrow
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

K-map simplification

For difference



Difference

A	B	C
1	0	0

$A\bar{B}\bar{C}$

A	B	C
0	0	1

$\bar{A}\bar{B}C$

A	B	C
1	1	1

ABC

A	B	C
0	1	0

$\bar{A}B\bar{C}$

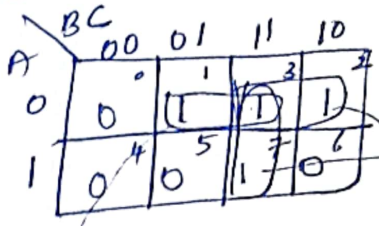
$Diff = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

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

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

$\boxed{Diff = A \oplus (B \oplus C)}$

Borrow



Borrow ⇒

A	B	C
0	✗	1

$\bar{A}BC$

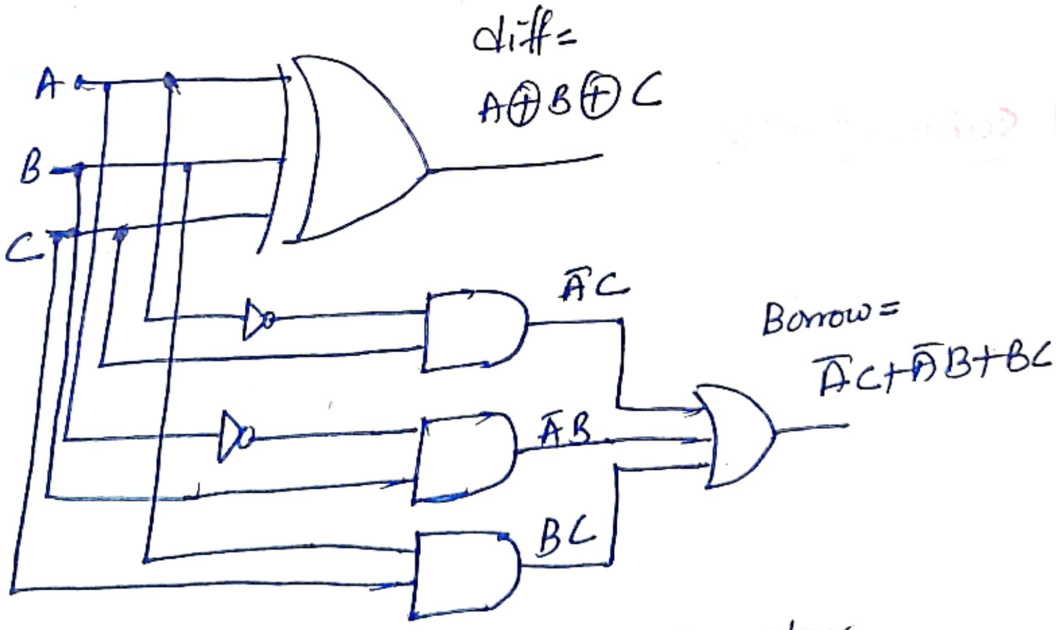
A	B	C
0	1	✗
1	✗	

$\bar{A}B$

A	B	C
✗	1	1
✗		

BC

$\boxed{Borrow = \bar{A}C + \bar{A}B + BC}$



logic circuit Full Subtractor

* Design of full adder using two half adders *

(or)

FULL

* Design of full adder using two half adders *

(or)

* full adder using two half adders *

Boolean expression for sum and carry in a full adder are

$$Sum = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$

$$Carry = AC + AB + BC$$

a full adder can also be implemented with two half adders and one OR gate, by deriving the sum carry expression as

$$Sum = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$

$$= \bar{C}[A\bar{B} + \bar{A}B] + C[AB + \bar{A}\bar{B}]$$

$$= \bar{C}[A \oplus B] + C[\overline{A \oplus B}]$$

$$= \bar{C}[A \oplus B] + C[\overline{A \oplus B}]$$

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

$$Carry = AC + AB + BC$$

Since AB' term is available in half adder as carry include missing literals in remaing two terms.

$$= AB + AC(B + \bar{B}) + BC(A + \bar{A})$$

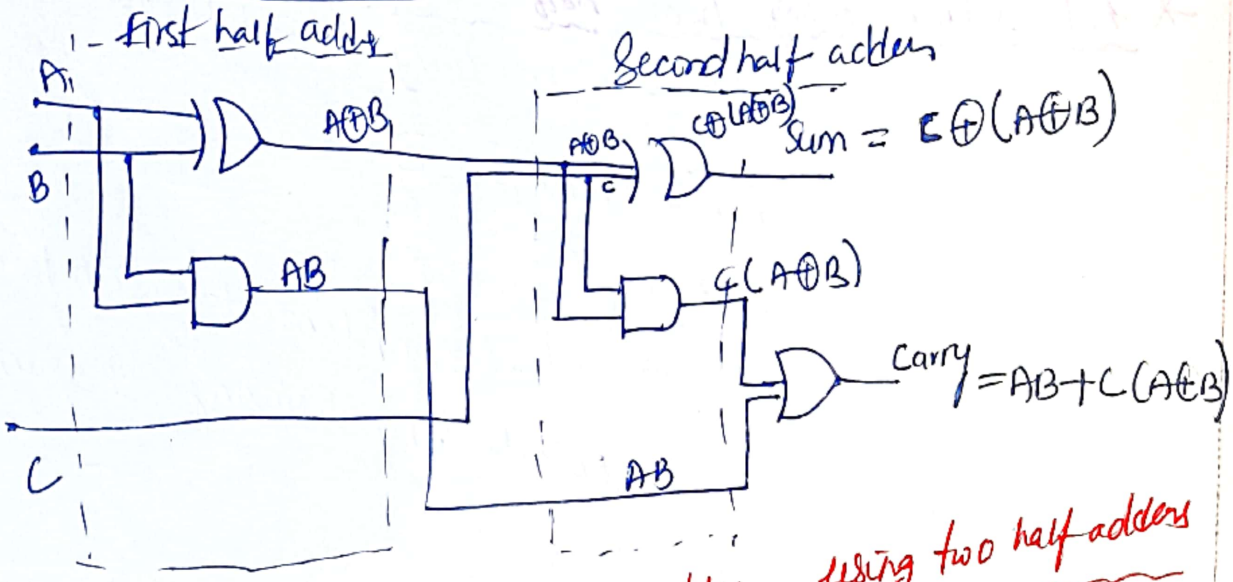
$$= AB + ABC + A\bar{B}C + ABC + \bar{A}BC$$

$$= AB[1 + C + C] + A\bar{B}C + \bar{A}BC$$

$$= AB + A\bar{B}C + \bar{A}BC$$

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

$\boxed{\text{Carry} = AB + C(A \oplus B)}$



Implementation of full adder using two half adders

* Full subtractor using two half subtractors *

→ A full subtractor can also implemented with two half subtractors and one OR gate.

The boolean functions for difference & borrow are.

$$D = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$

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

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

$$\boxed{D = C \oplus (A \oplus B)}$$

$$\text{Borrow} = \bar{A}C + \bar{A}B + BC$$

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

$$= \bar{A}BC + \bar{A}\bar{B}C + \bar{A}B + ABC + \bar{A}BC$$

$= \bar{A}B(C+1+C) + \bar{A}\bar{B}C + ABC$ $(\because 1+C+C = 1)$

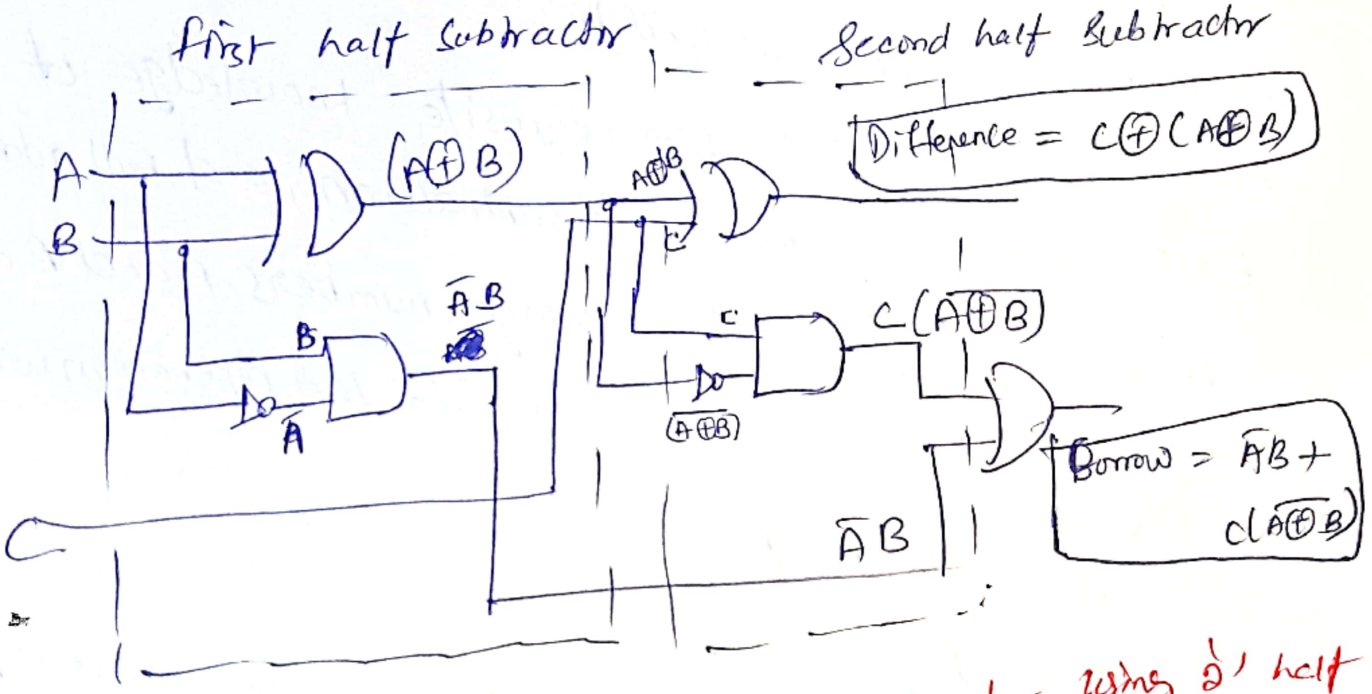
$= \bar{A}B + \bar{A}\bar{B}C + ABC$ $(\because 1+C = 1)$

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

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

Carry ⇒

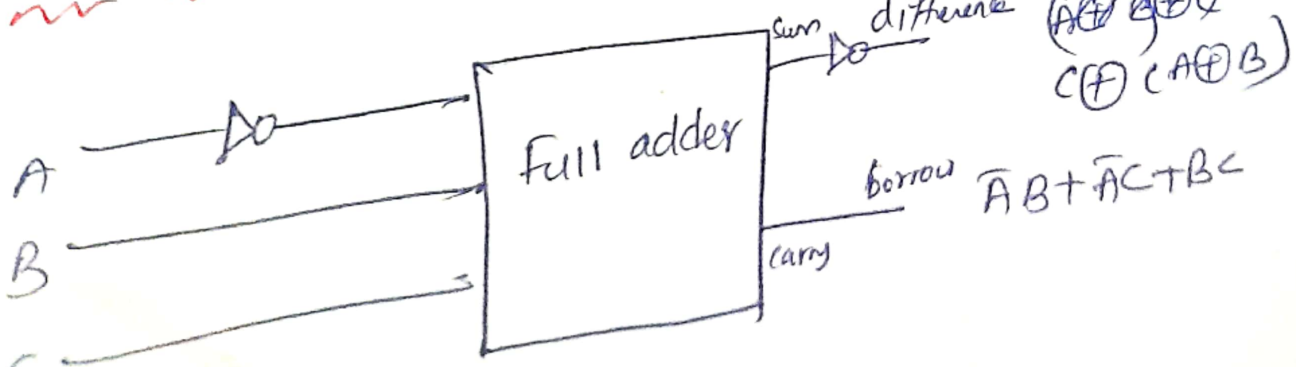
Borrow = $\bar{A}B + C(\overline{A \oplus B})$



Implementation of a full Subtractor using 2 half Subtractors.

V.Imp

Design full subtractor by using full Adder



Decimal adder (or) Binary adder and Subtractor Circuit

→ In a Digital circuits, A Binary Adder-Subtractor circuit can do both the addition & subtraction of binary numbers in one circuit itself.

→ The operation is performed depending on the binary value the control signal holds. It is one of the components of the ALU (Arithmetic logic unit).

→ This circuit requires pre requisite knowledge of EX-OR gate, binary addition & subtraction, and full adder.

→ Let's consider two 4 bit binary numbers A and B as inputs to the Digital circuit for the operation with digits.

$A \rightarrow A_1\ A_2\ A_3\ A_4$

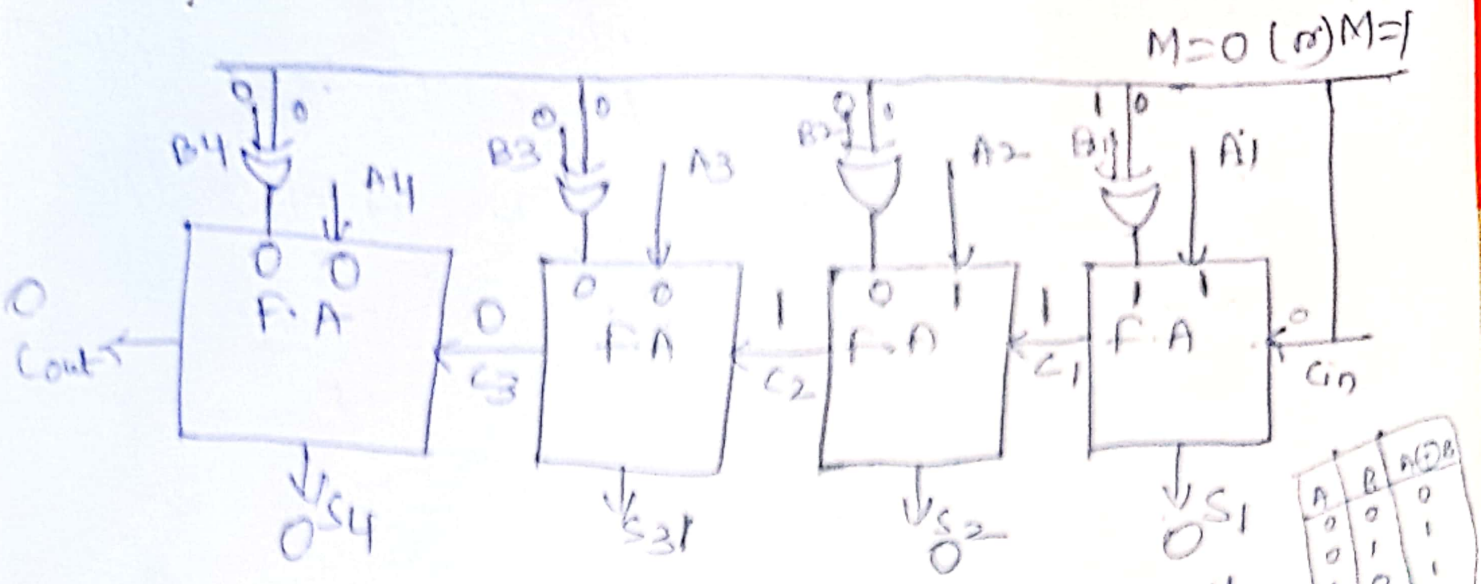
$B \rightarrow B_1\ B_2\ B_3\ B_4$

→ The circuit consists of 4 full adders since we are performing operations on 4-bit numbers. There is a control line M that holds a binary value of either 0 (or) 1 which determines that the operation is carried out is addition (or) subtraction.

→ The main advantage of binary adder & subtractor is (1) low Design complexity

(2) High-speed operations

(3) Versatility → It allows the same hardware circuit to add as to subtract.



(i) If M=0 ⇒ It will acts as Adder circuit

(ii) If M=1 ⇒ It will acts as Subtractor circuit.

For example $A = \overset{A_4\ A_3\ A_2\ A_1}{0\ 0\ 1\ 1}$ $B = \overset{B_4\ B_3\ B_2\ B_1}{0\ 0\ 0\ 1}$

$$A+B = \begin{array}{r} 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1 \\ \hline 0\ 1\ 0\ 0 \end{array}$$

$$A-B = \begin{array}{r} 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1 \\ \hline 2 \end{array} \quad \begin{array}{l} \rightarrow 3 \\ \rightarrow 1 \end{array}$$

→ When M=0, for 1st EX-OR gate one input 0, and B_1 is 1 so $1 \oplus 0 = 1$, then output is 1 and $A_1 = 1$ so $C_{in} = 0$, then $\boxed{S_1 = 0}$; $\boxed{C_1 = 0}$

→ M=0, for 2nd EX-OR gate one input 0 & $B_2 = 0$, then $0 \oplus 0 = 0$; the O/P is 0 & $A_2 = 1$ & $C_1 = 1$, the 101 $\boxed{S_2 = 0}$ $\boxed{C_2 = 1}$

→ M=0; for 3rd EX-OR gate one input 0 & $B_3 = 0$ then $0 \oplus 0 = 0$; then O/P is 0 & $A_3 = 0$; next 001 → $\boxed{S_3 = 1}$, $\boxed{C_3 = 0}$

→ M=0; for 4th EX-OR gate one input is 0, $B_4 = 0$, then $0 \oplus 0 = 0$, the output 0 & $A_4 = 0$; next 000 then $\boxed{S_4 = 0}$

S_4	S_3	S_2	S_1
0	1	0	0

⇒ 4

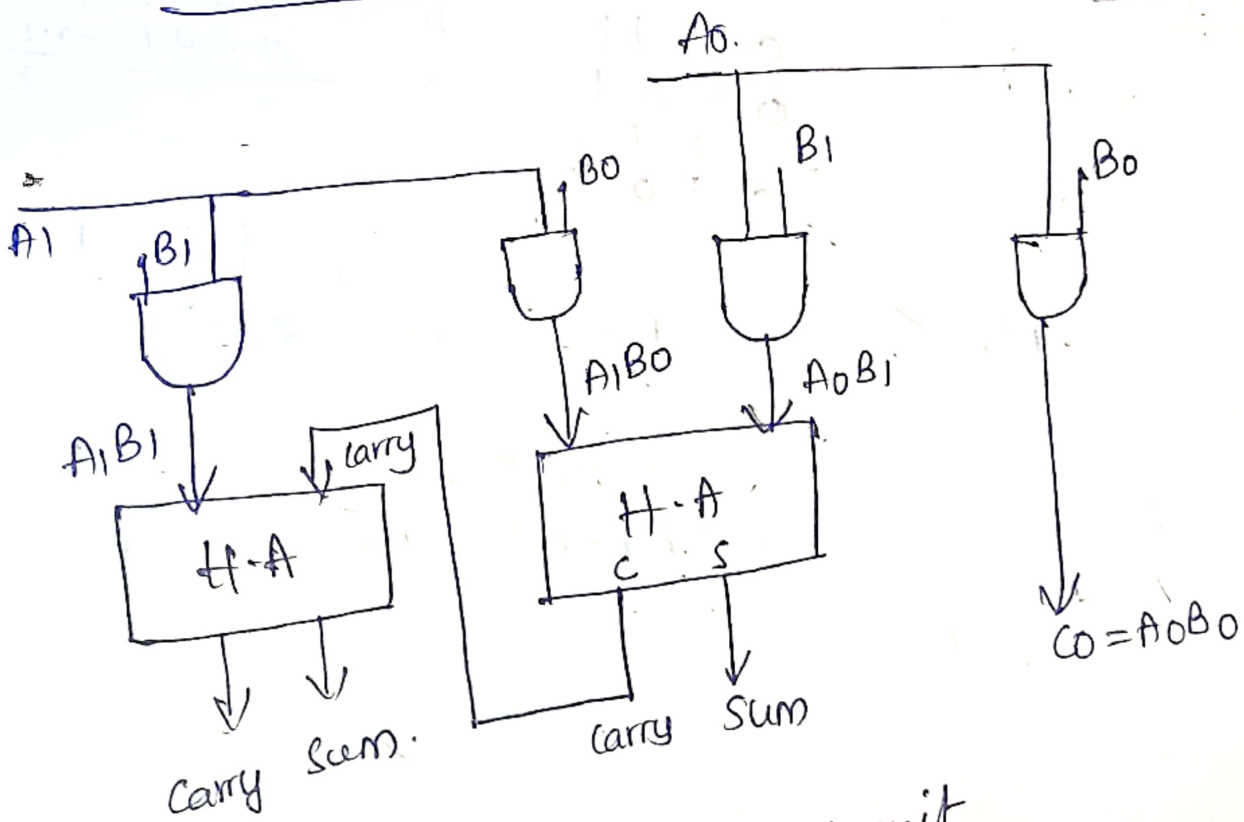
* Binary multiplier *

→ Binary multiplier is a combinational logic circuit is used in digital systems to perform the multiplication of two binary numbers

→ Binary multipliers are used in Digital signal processing and algorithms.

$$\begin{array}{cc} A_1 & A_0 \\ B_1 & B_0 \\ \hline A_1B_0 & A_0B_0 \\ A_1B_1 & A_0B_1 \quad X \\ \hline A_1B_1 & A_1B_0 + A_0B_1 \quad A_0B_0 \end{array}$$

$$\begin{array}{r} 10 \rightarrow 2 \\ \times\ 11 \rightarrow 3 \\ \hline 10 \\ \hline 10X \\ \hline 110 \rightarrow 6 \end{array}$$



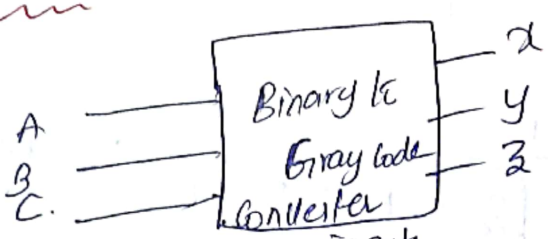
Binary multiplier circuit

Code Converters :-

→ Code Converter is a logic circuit whose inputs are bit patterns representing numbers in one code and whose outputs are the corresponding representations in a different code.

→ Code converters are the digital circuits (or) algorithms that are designed to translate data representation from one format to the other format.

Example :- Design a 3-bit binary to 3-bit Gray code.



A B C — Inputs

x y z. — outputs

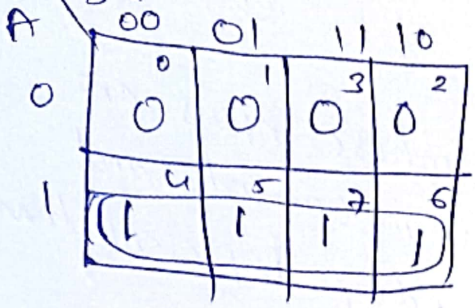
	3-Bit Binary Inputs			Gray code		
	A (4)	B (2)	C (1)	x	y	z
0 →	0	0	0	0	0	0
1 →	0	0	1	0	0	1
2 →	0	1	0	0	1	1
3 →	0	1	1	0	1	0
4 →	1	0	0	1	1	0
5 →	1	0	1	1	1	1
6 →	1	1	0	1	0	1
7 →	1	1	1	1	0	0

A B	A ⊕ B
0 0	0
0 1	1
1 0	1
1 1	0

0 ⊕ 0 ⊕ 1 → 0 0 1

0 ⊕ 1 ⊕ 0 → 0 1 1

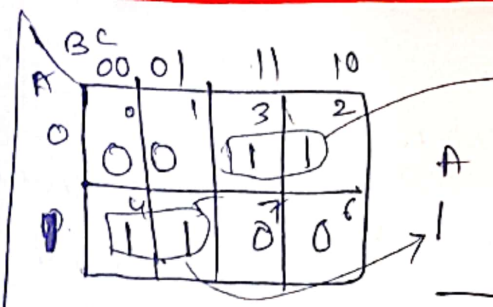
0 ⊕ 1 ⊕ 1 → 0 1 0



A B C
1 ⊘ ⊘
⊘ ✗
✗ ⊘

$x = A$

$\boxed{x = A}$ ✓



A B C
1 0 ⊘
0 ✗

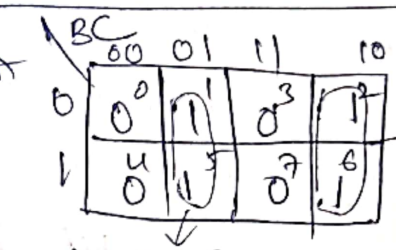
$A\bar{B}$

A B C
⊘ 1 ✗
1 ⊘

$\bar{A}B$

$y = A\bar{B} + \bar{A}B$

$\boxed{y = A \oplus B}$ ✓



A B C
⊘ 1 0
✗

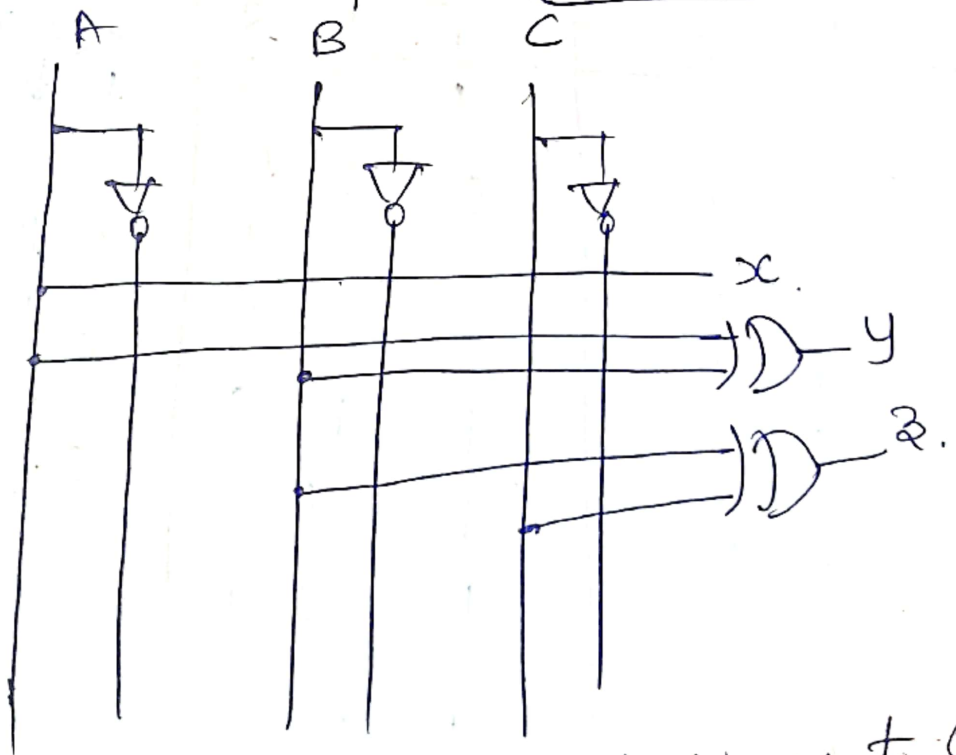
$B\bar{C}$

A B C
⊘ 0 1
✗

$\bar{B}C$

$z = \bar{B}C + B\bar{C}$

$\boxed{z = B \oplus C}$

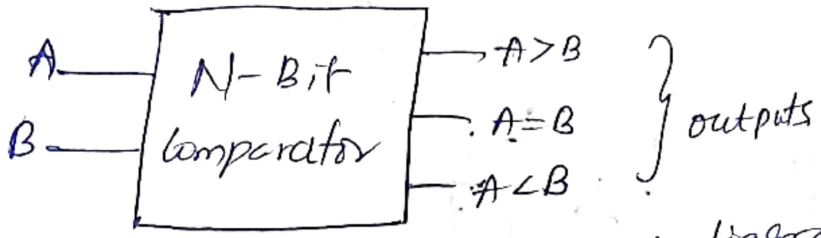


logic circuit for 3 Bit binary to Graycode

Comparator * (Magnitude comparator) *

→ In digital electronics a comparator is a device that compares two binary number to determine if one is greater than (or) less than (or) equal to the other

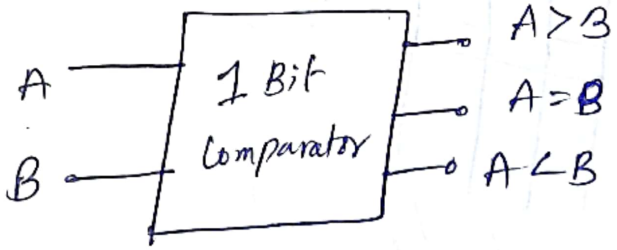
→ Comparator is a combinational logic circuit which is used to compare two bits and produces the three types of outputs $A<B$, $A=B$, $A>B$.



Magnitude Comparator Block diagram

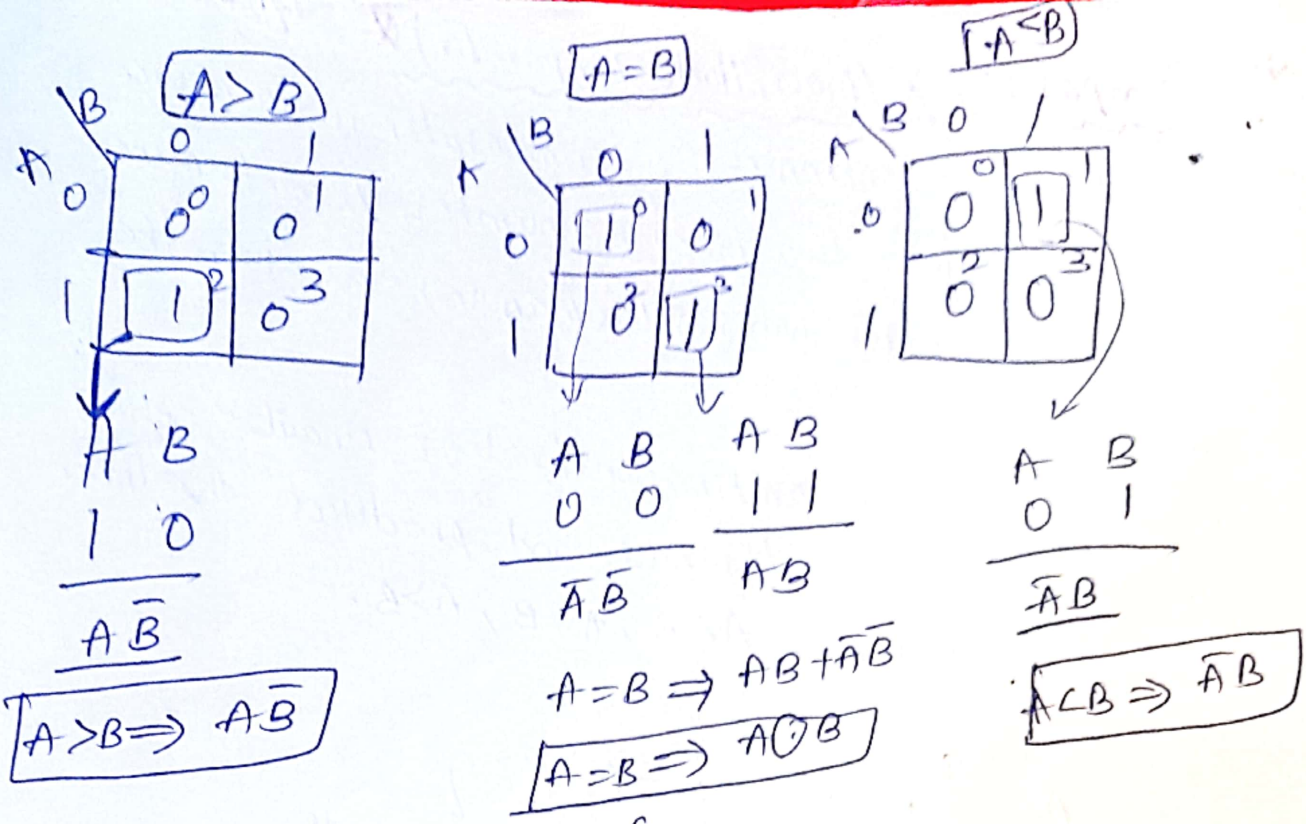
$N = 1, 2, 4 \ldots$

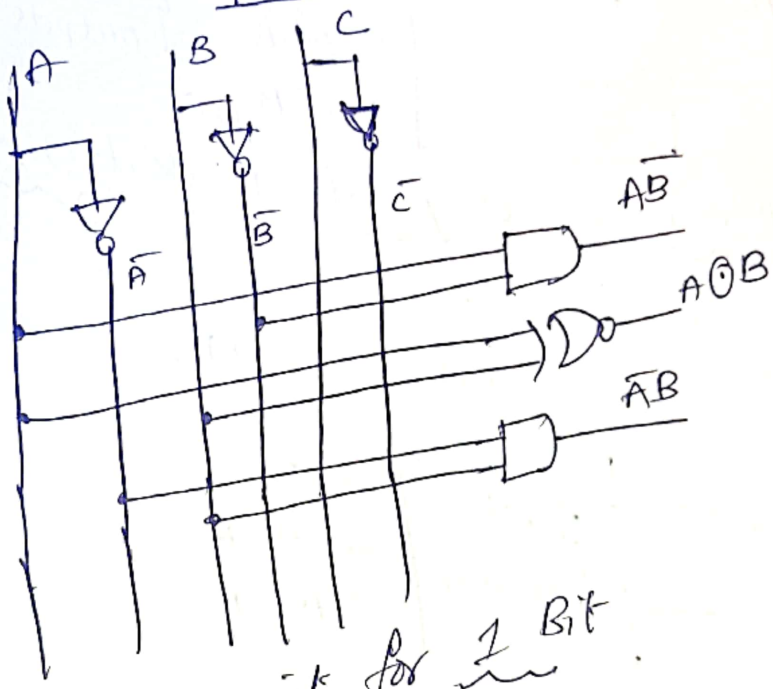
If $N=1 \Rightarrow$ 1 Bit comparator



A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

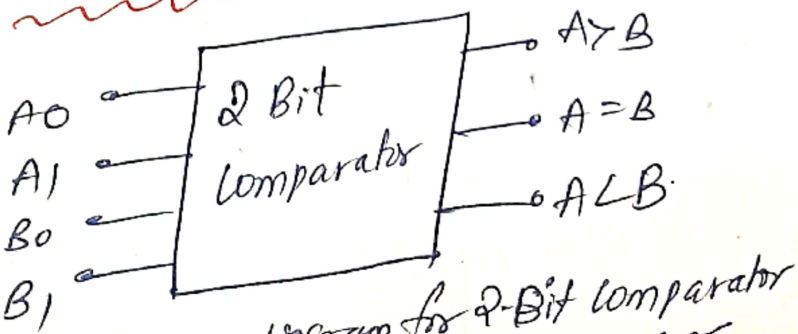
→ Applications of comparator are used in CPU & Micro - Controllers, In Biometric verification and to compare the temperature (or) position to a reference value





Logic circuit for 1 Bit Comparator circuit

* 2 Bit Magnitude comparator *



Block diagram for 2-Bit comparator

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	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	0	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

$A_1 A_0$ $B_1 B_0$

0 0 | 0 1

$0 < 1$

0 0 | 1 1

$0 < 3$

1 1 | 1 0

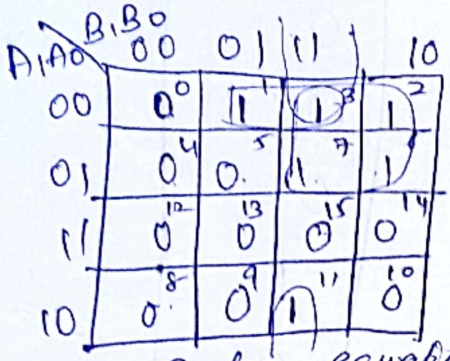
$1 < 2$

$A_1 A_0$ $B_1 B_0$

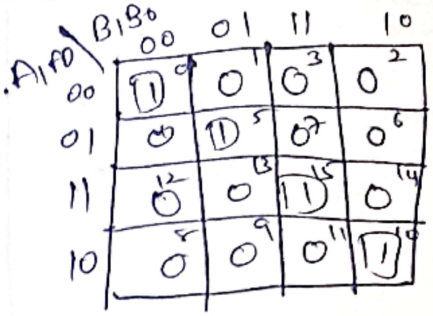
1 1 | 1 0

$3 > 2$

K-map for $A<B$



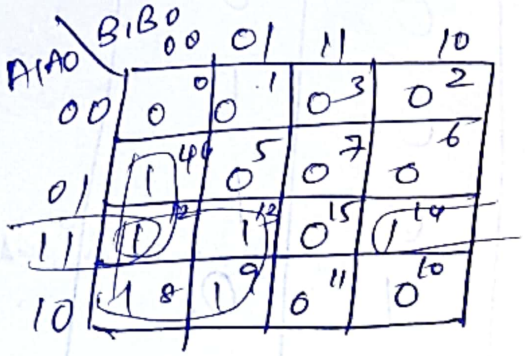
K-map for $A=B$



Write Boolean equation & simplify.

$$L = \bar{A}_1 B + (A_1 \odot B_1)\bar{A}_0 B_0$$

$$E = (A_1 \odot B_1)(A_0 \odot B_0)$$



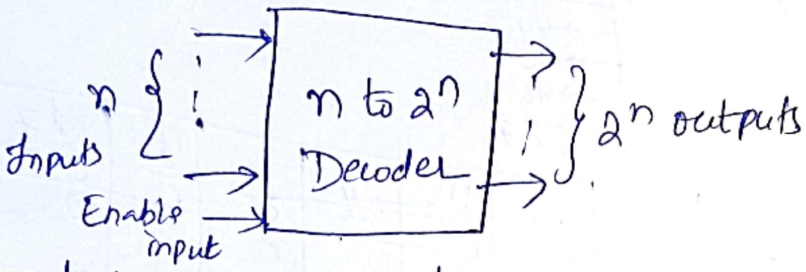
$$G = A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0$$

Decoders

→ Decoder is a combinational circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs. They are used in a wide variety of applications, including instruction decoding, data multiplexing and data demultiplexing, seven-segment displays, and as address decoders for memory and port-mapped I/O.

Decoder :- The combinational circuit that change the Binary information into 2^n output lines is known as Decoders. The binary information is passed in the form of n input lines. The o/p lines define the 2^n-bit code for the binary information. In simple words the Decoder performs the reverse operation of the Encoder. At a time, only one input line is activated for simplicity. The produced 2^n-Bit output code is equivalent to the Binary information.

Block diagram



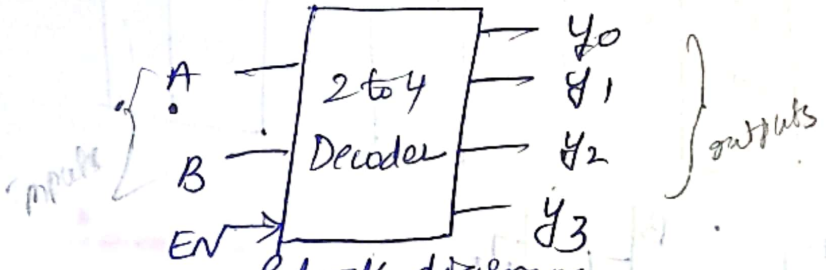
Example:

If $n=1$; $2^1=2 \Rightarrow$ 1 to 2 Decoder

If $n=2$; $2^2=4 \Rightarrow$ 2 to 4 Decoder

If $n=3$; $2^3=8 \Rightarrow$ 3 to 8 Decoder

If $n=4$; $2^4=16 \Rightarrow$ 4 to 16 Decoder



Block diagram

A	B	y_0	y_1	y_2	y_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

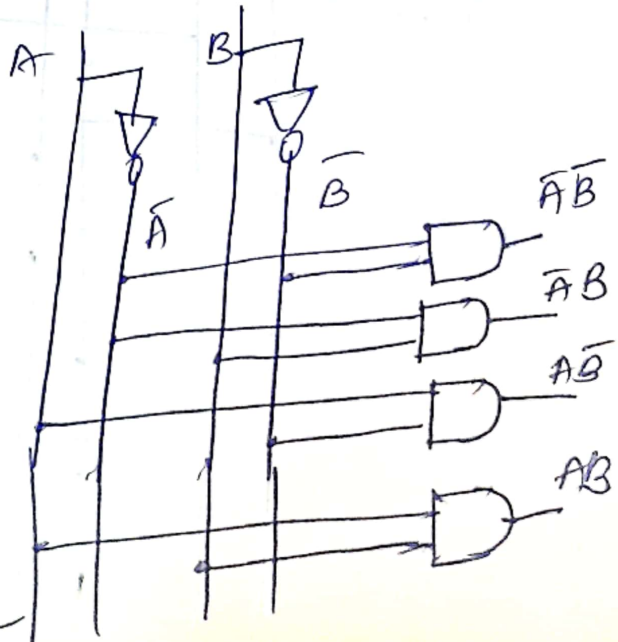
EN	A	B	y_0	y_1	y_2	y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$y_0 = \bar{A}\bar{B}$

$y_1 = \bar{A}B$

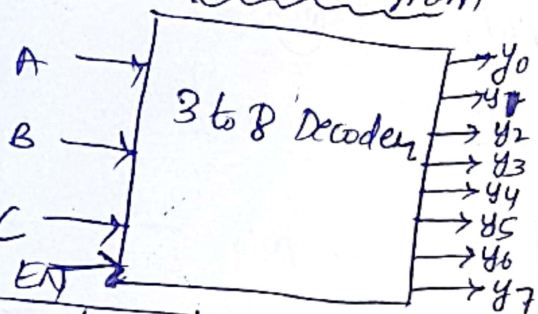
$y_2 = A\bar{B}$

$y_3 = AB$



2 to 4 line decoder

Block diagram



EN	A	B	C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
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	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

$y_0 = \bar{A}\bar{B}\bar{C}$

$y_1 = \bar{A}\bar{B}C$

$y_2 = \bar{A}B\bar{C}$

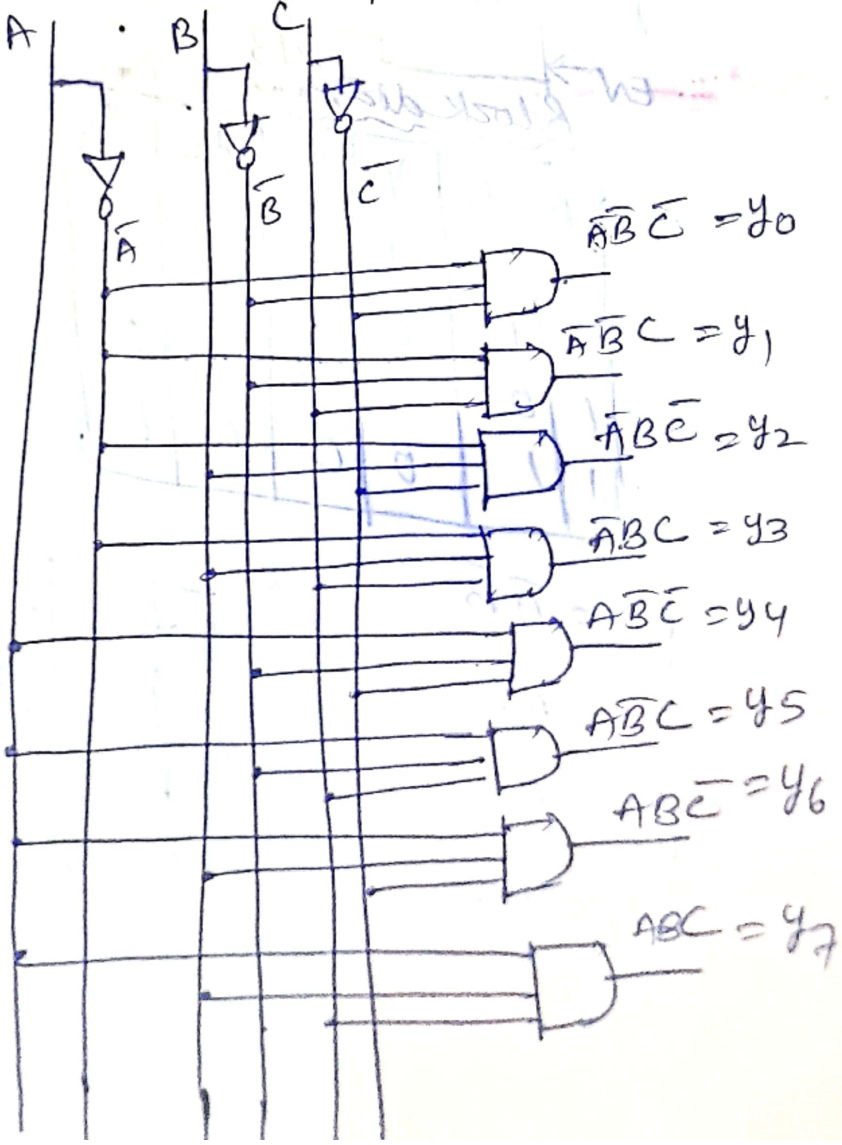
$y_3 = \bar{A}BC$

$y_4 = A\bar{B}\bar{C}$

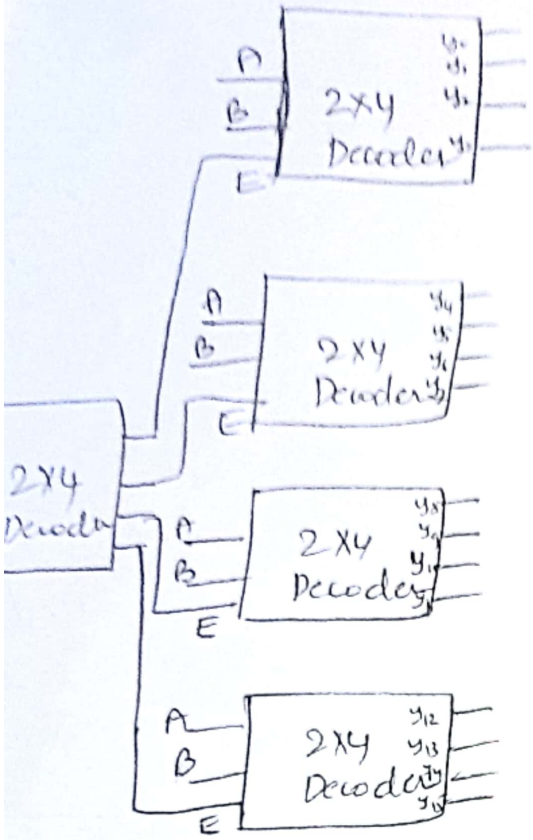
$y_5 = A\bar{B}C$

$y_6 = AB\bar{C}$

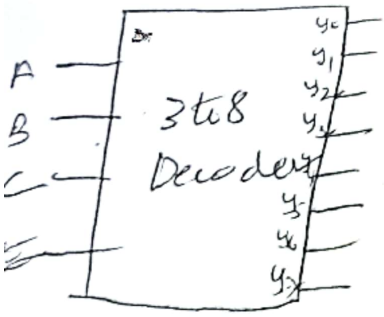
$y_7 = ABC$

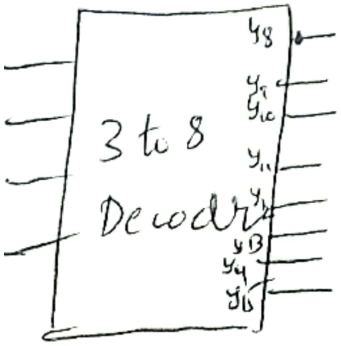


* Design 4x16 Decoder by using 2 to 4 Decoder



Design a 4x16 Decoder by using a 3 to 8 Decoder.





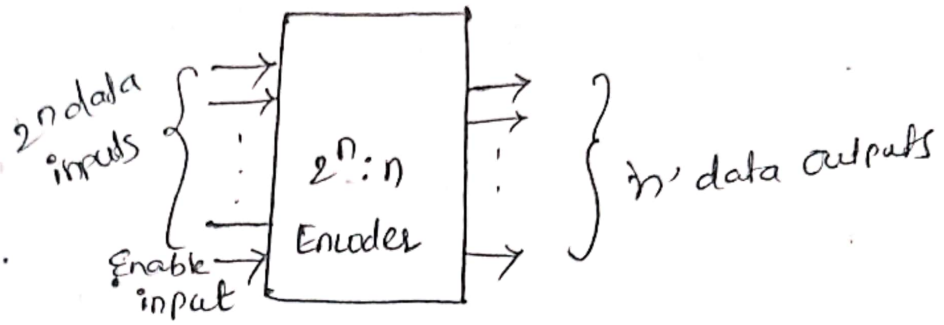
E	A	B	C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	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	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

*Encoders:-

→ An Encoder is a digital circuit that performs the inverse operation of a decoder.

→ Encoder generates the binary code corresponding to the input value.

→ Encoder has '2^n' input lines and 'n' output lines



Block diagram of Encoder

→ *Types of Encoders:-

1. Priority Encoder
2. Decimal to BCD Encoder
3. octal to binary Encoder

→ Priority Encoder:-

A Priority encoder is a digital logic circuit that provides a binary output code corresponding to the highest-priority active input. Unlike a standard Encoder, which can produce an incorrect (or) ambiguous output if multiple inputs are active, a priority Encoder resolves this conflict by assigning a priority level to each input.

→ A 4 to 2 Priority Encoder has four inputs y_3, y_2, y_1, y_0 and two outputs A_1 & A_0, Here the input, y_3 has the highest priority, whereas the input, y_0 has the lowest priority. In this case, even if more than one input is '1' at the same time, the output will be the binary code corresponding to the input, which is having higher priority. The truth table of 4 to 2 priority Encoder is shown below.

Inputs				outputs		o/p valid (or) not
y_3	y_2	y_1	y_0	A_1	A_0	V.
0	0	0	0	0	0	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

→ We consider one more output, 'V' in order to know whether the code available at outputs is valid (or) not

→ If at least one input of the encoder is '1', then the code available at outputs is valid one.

→ In this case, the output, V will be equal to 1. (A)

→ If all the inputs of Encoder are '0', then the code available at outputs is not a valid one. In this case, the output, V will be equal to 0.

→ The simplified Boolean functions are.

$$A_1 = y_3 + y_2 A_1 = y_3 + y_2$$

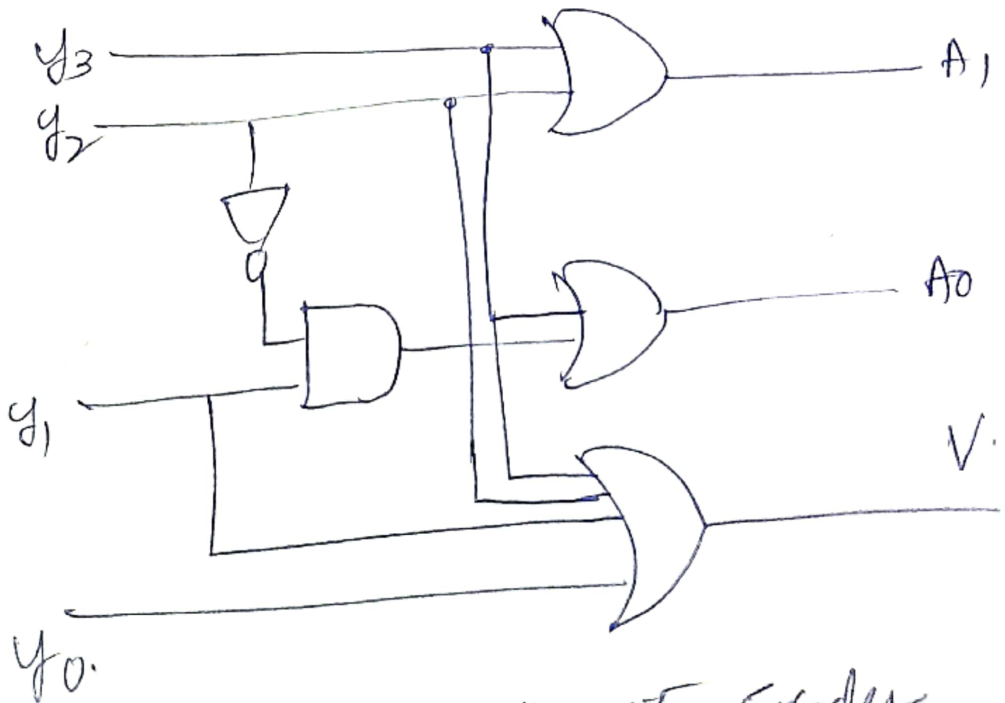
$$A_0 = y_3 + y_2 y_1 A_0 = y_3 + y_2 y_1$$

Similarly we will get the Boolean function of o/p, V as

$$V = y_3 + y_2 + y_1 + y_0 V$$

$$= y_3 + y_2 + y_1 + y_0$$

→ We can implement the above boolean functions using logic gates. The circuit diagram of 4 to 2 priority Encoder is shown in the following figure.



4 to 2 Priority Encoder

The circuit diagram contains two- 2- input OR gates One 4-input OR gate, one 2-input AND gate & an inverter. Here AND gate & inverter combination are used for producing a valid code at the outputs, even when multiple inputs are equal to '1' at the same time. Hence, this circuit encodes the four inputs with two bits based on the priority assigned to each input.

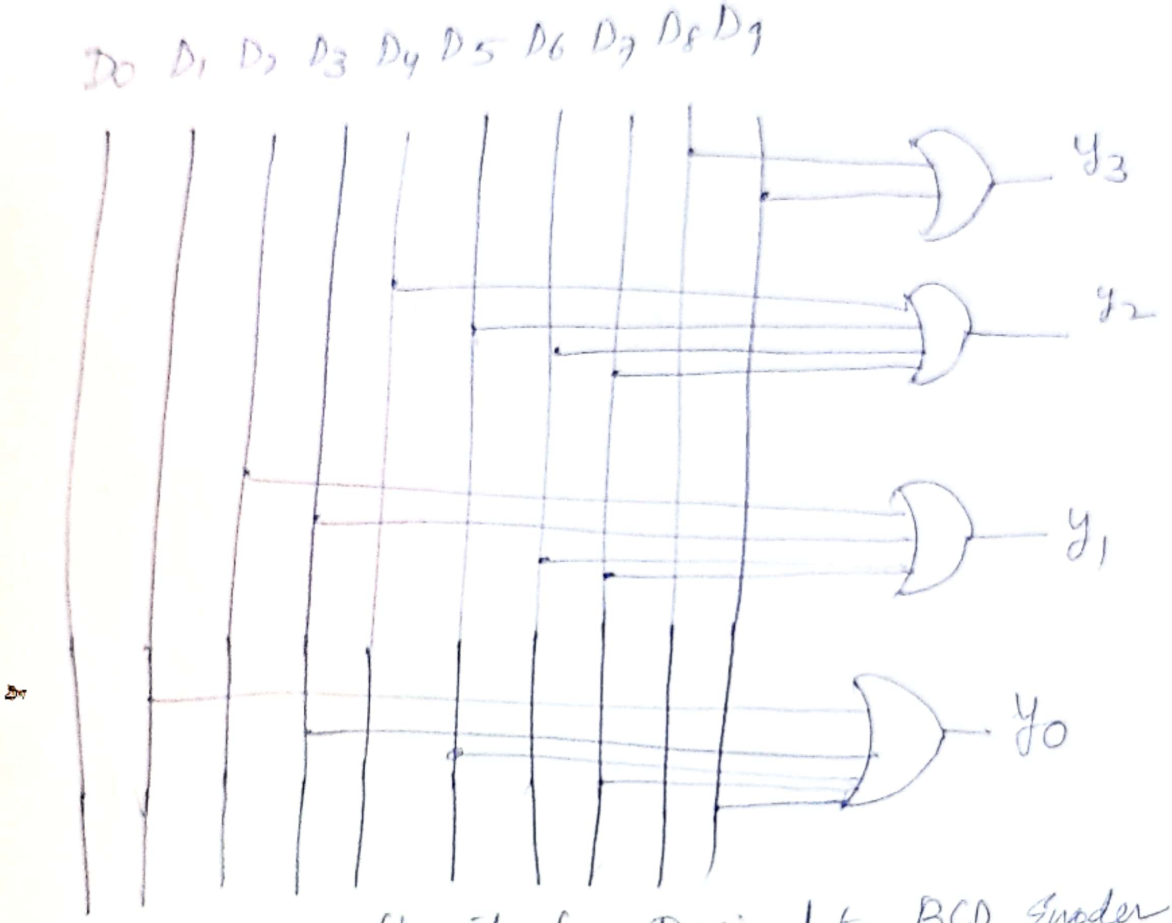
(2) Decimal to BCD Encoder:-

A decimal to BCD encoder has 10 input lines D0 to D9 and 4 output lines y_0 to y_3. The truth table for a decimal to BCD encoder.

Input										output			
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

→ Boolean functions are formed by ORing all the input lines for which output is 1. For instance y_0 is 1 if D_1, D_3, D_5, D_7 & D_9 input lines

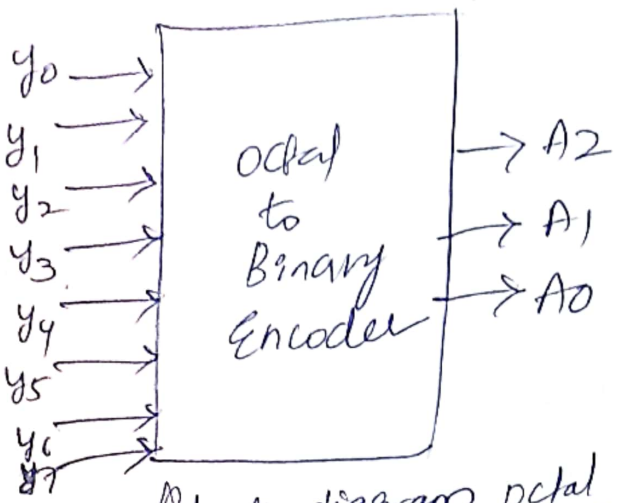
$$y_0 = D_1 + D_3 + D_5 + D_7 + D_9$$
$$y_1 = D_2 + D_3 + D_6 + D_7$$
$$y_2 = D_4 + D_5 + D_6 + D_7$$
$$y_3 = D_8 + D_9$$



Circuit for Decimal to BCD Encoder:-

(3) Octal to Binary Encoder:-

→ Octal to binary Encoder has eight inputs, y_7 to y_0 & three outputs A_2, A_1, A_0. Octal to binary Encoder is nothing but 8 to 3 Encoder. The block diagram of octal to binary Encoder is



Block diagram Octal to Binary Encoder

→ At any time, only one of these eight inputs can be '1' in order to get the respective binary code.

y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$A_2 = y_7 + y_6 + y_5 + y_4$$

$$A_1 = y_7 + y_6 + y_3 + y_2$$

$$A_0 = y_7 + y_5 + y_3 + y_1$$

* Multiplexer *

→ 'Multiplexer is a special type of a combinational circuit

→ In digital systems, many times it is necessary to select single data line from several data i/p lines, and the data from the selected data line should be available on the output the digital ckt which does this task is a "multiplexer"

→ The selection of a particular input line is controlled by a set of selection lines.

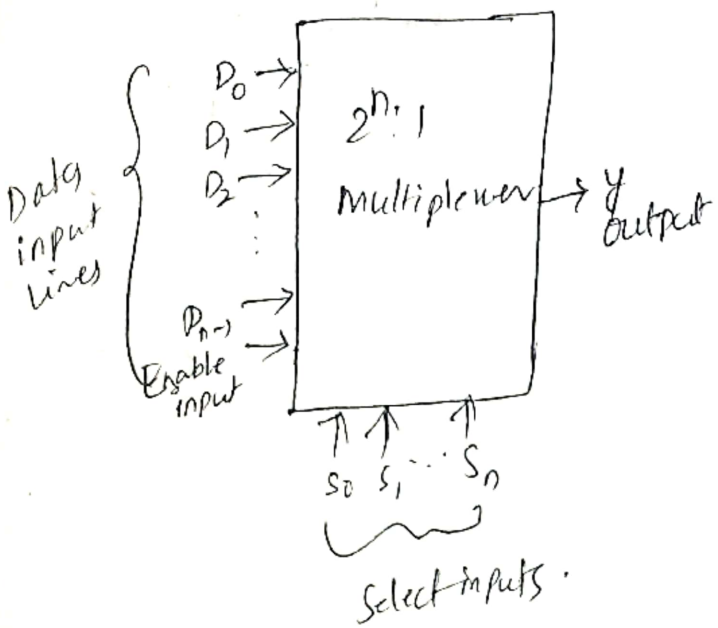
→ "multiplexer" is a combinational logic circuit, it has '2^n' inputs and 'n' selection lines and 1 output.

→ The selection of a particular i/p line is controlled by a set of selection lines

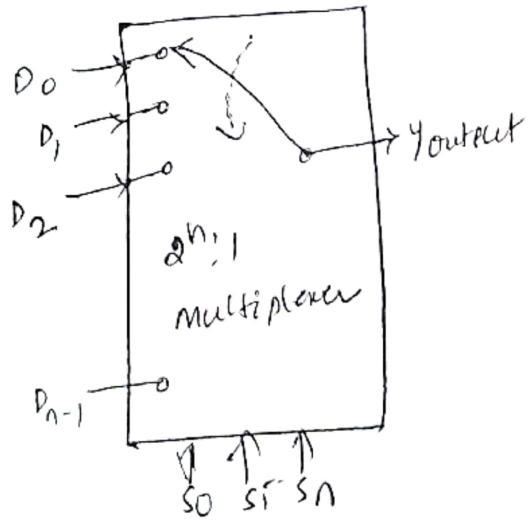
→ Normally, there are 2^n input lines and 'n' selection lines. whose bit combinations determine which i/p is selected. Therefore, multiplexer is 'many in to one' & it provides the digital equivalent of an analog selector switch.

→ multiplexer is also known as "Data Selector"

→ multiplexer is also known as MUX

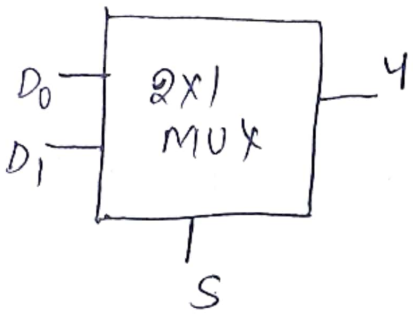


(a) Block diagram of $2^n:1$ multiplexer



(b) Equivalent ckt

① 2×1 MUX :-

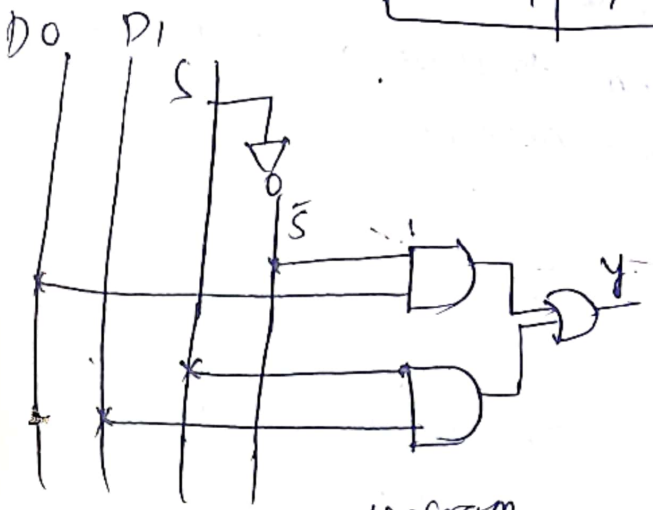


Truth table

D_1	D_0	S	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

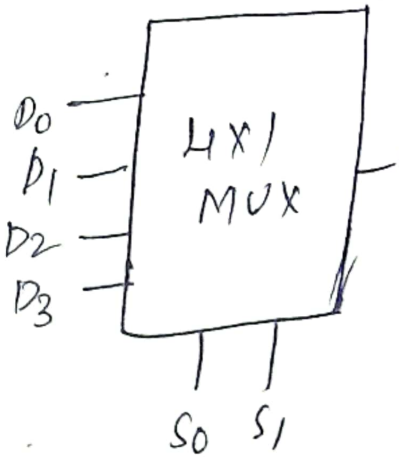
S	y
0	D_0
1	D_1

$y = \bar{S}D_0 + SD_1$



Circuit diagram

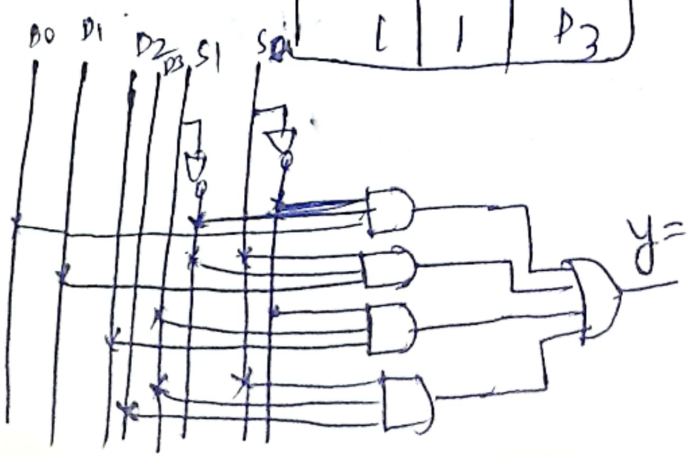
② 4×1 MUX :-



Truth table

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

$y = \bar{S_1}\bar{S_0}D_0 + \bar{S_1}S_0D_1 + S_1\bar{S_0}D_2 + S_1S_0D_3$



Example (1) 2:1 MUX
(2) 4:1 MUX
(3) 8:1 MUX
(4) 16:1 MUX

→ The generalized size of multiplexer is $2^n:1$

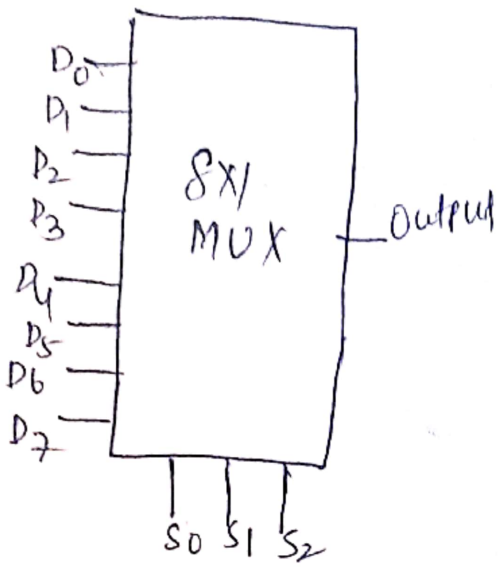
→ The MUX are used in control system for signal routing & data consolidation.

* Applications :-

(1) computer memory
(2) Telecommunication Networks (Broad casting & N/w systems to share a single communication channel (or) medium among multiple)
(3) Digital applications
(4) Consumer applications
(5) control systems
(6) used to reduce complexity in wireless systems
(7) ~~Satellite~~ Satellite communications
(8) Space communications
(9) High-speed optical circuits
(10) Analog to Digital conversions, memory addressing

→ multiplexer is a digital switch, it allows digital information from several sources to be routed on to a single o/p line.

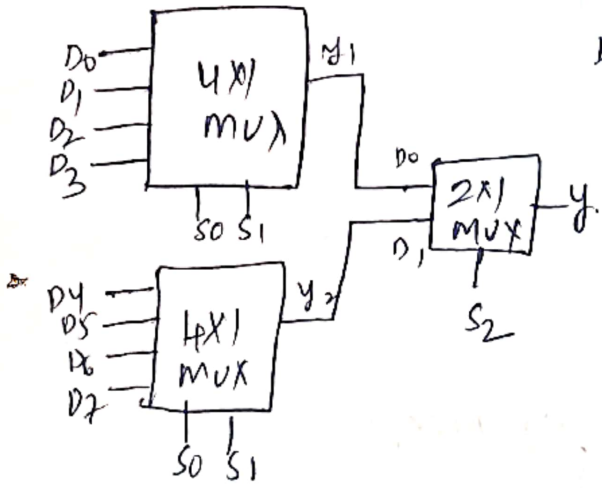
(3) 8x1 MUX :



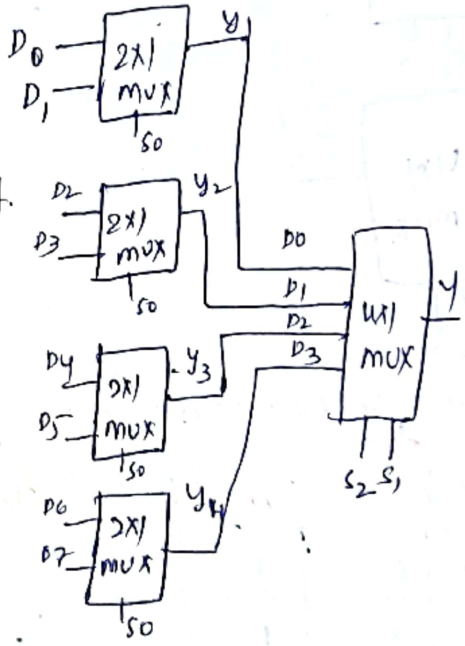
Truth table

S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

8x1 mux using 4x1 mux

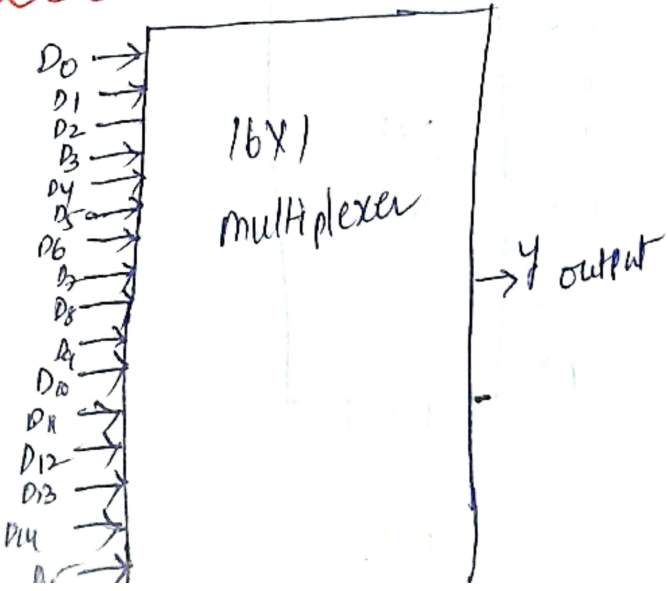


8x1 mux using 2x1 multiplexer



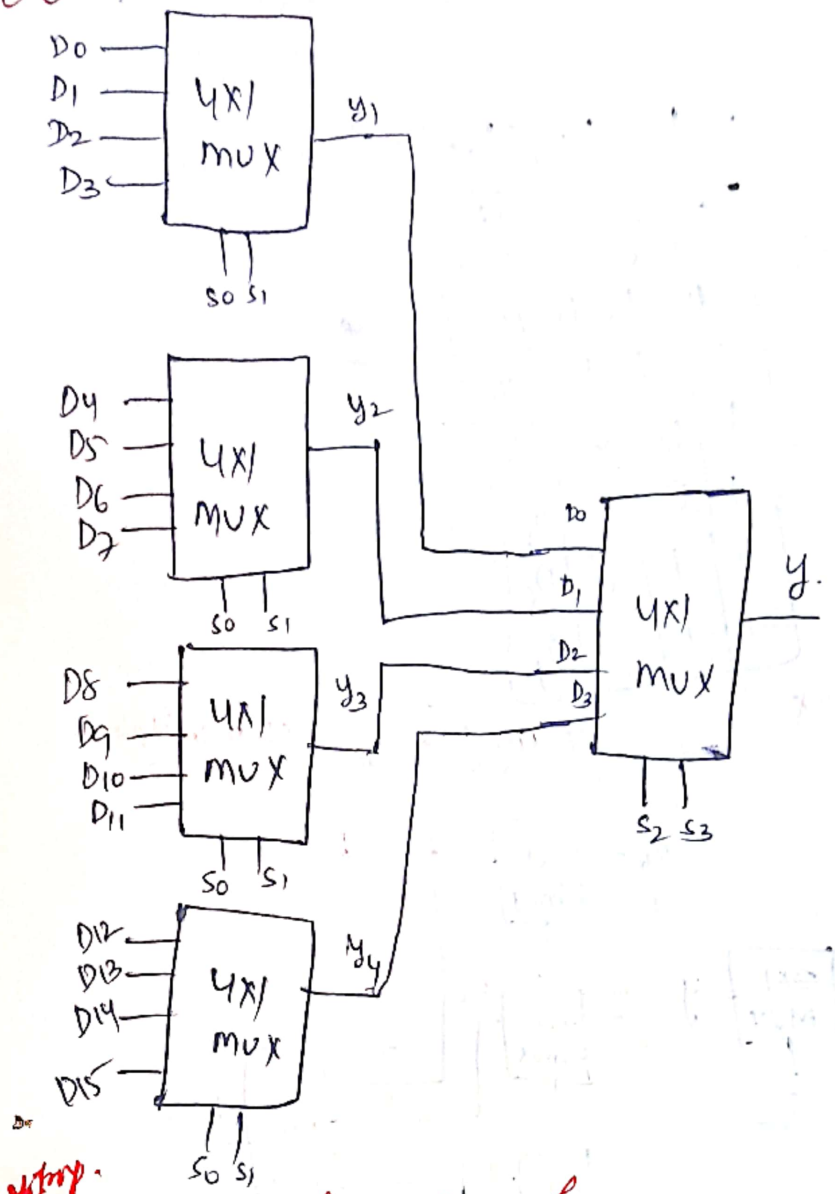
wrong

* 16x1 multiplexer using ~~using~~ *



imp.

16X1 MUX using 4X1 MUX

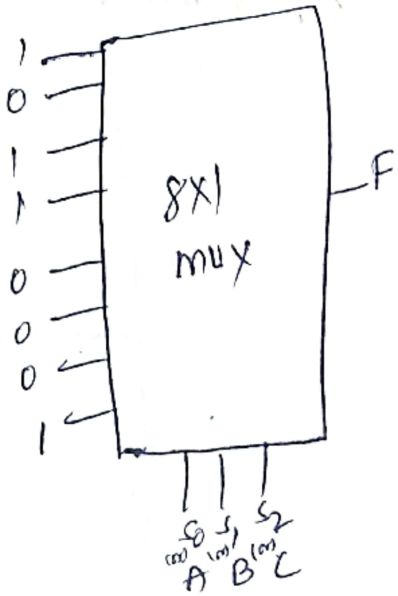


imp.

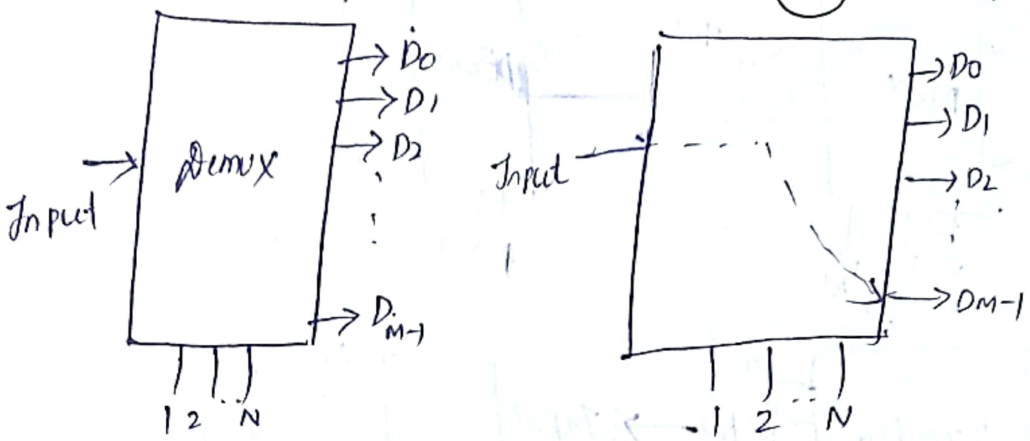
Implementation of Boolean function using MUX :-

$F(A,B,C) = \sum m(0,2,3,7)$

	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1



Demultiplexers

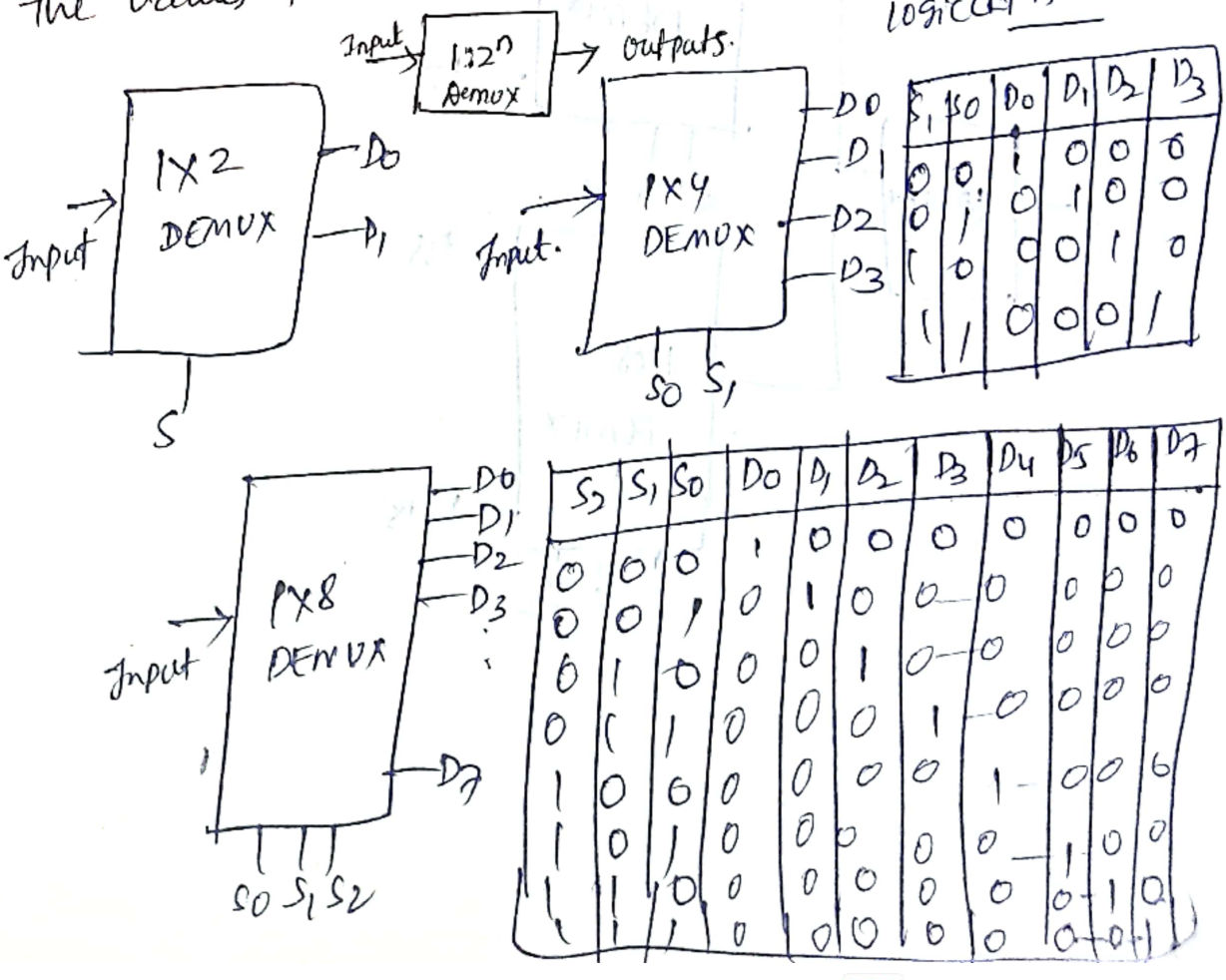


one to many device

Demultiplexers:

→ A Demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines

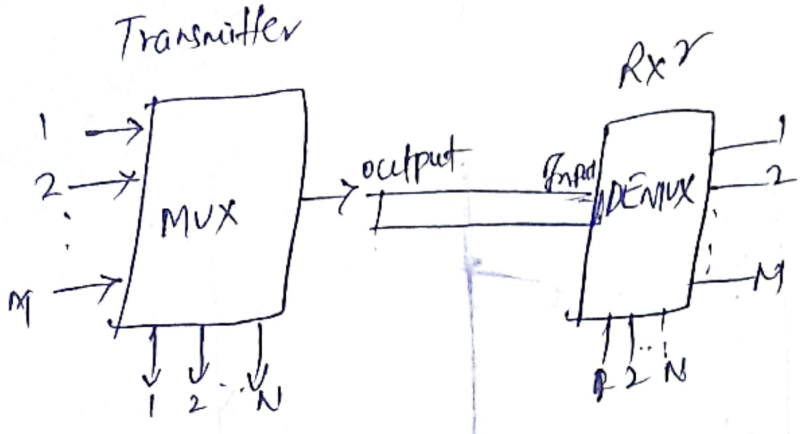
→ The selection of specific output line is controlled by the values of 'n' select lines

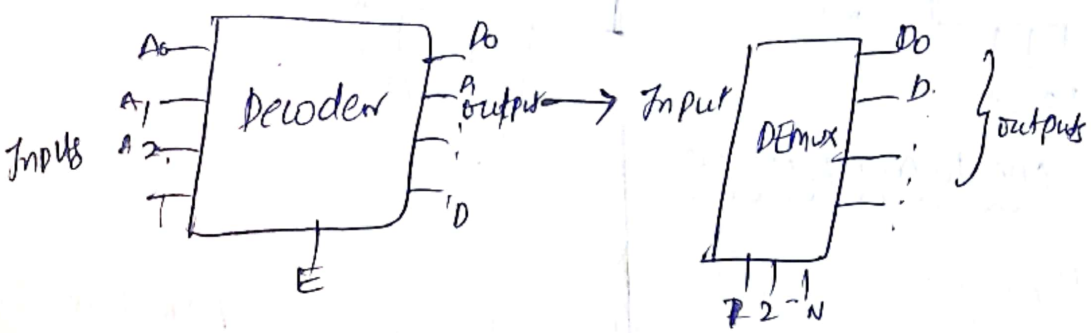


using logic ckt AND gate

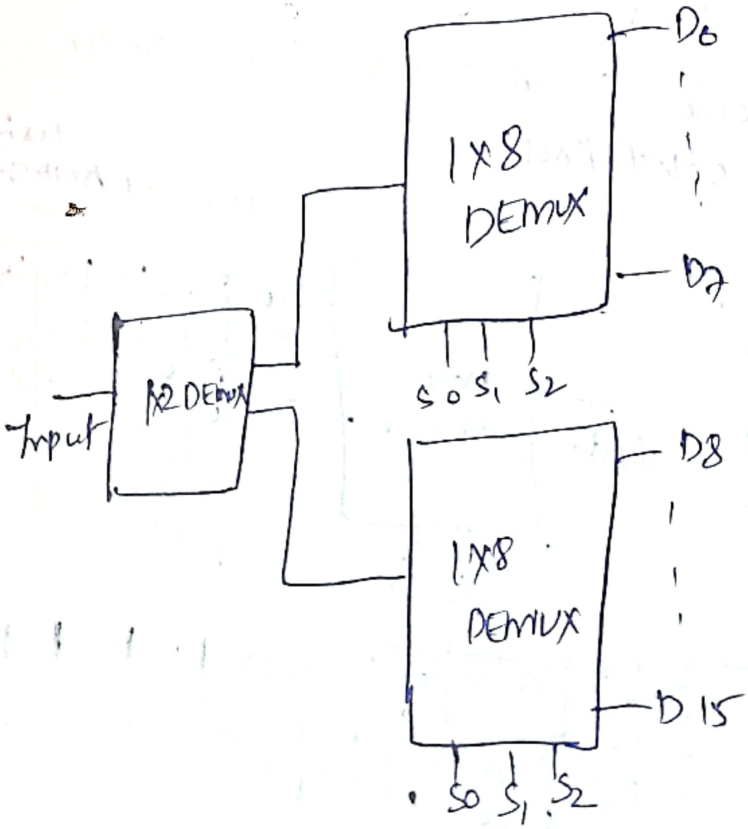
S_1	S_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

S_2	S_1	S_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

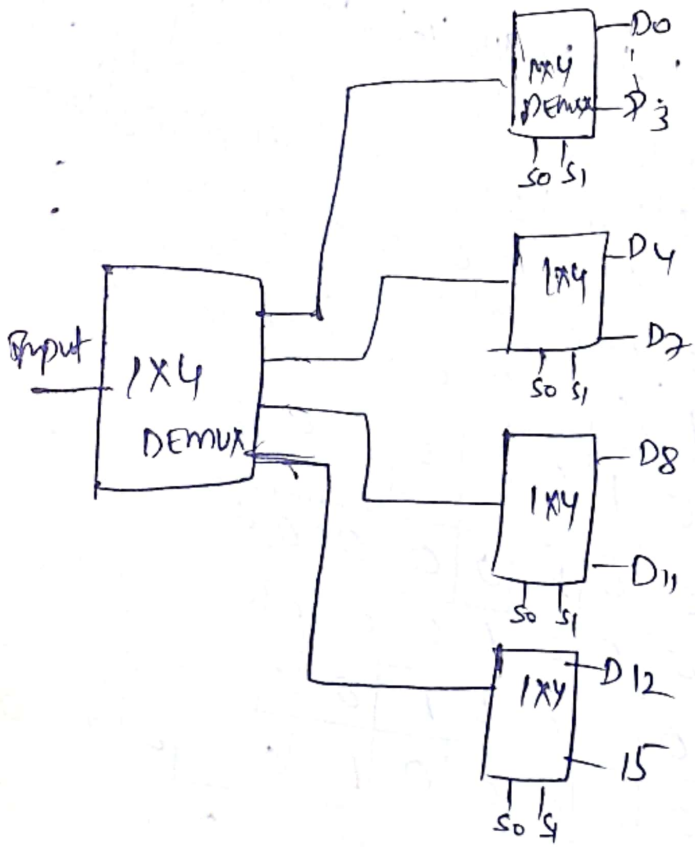




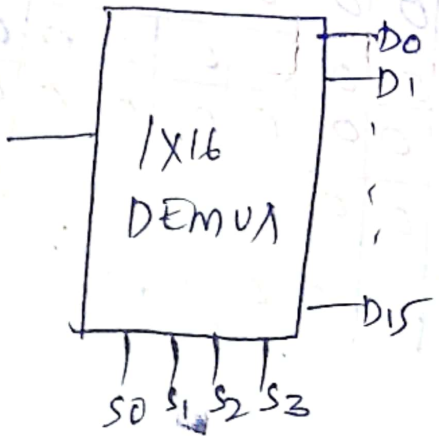
1X16 DEMUX using 1X8 DEMUX



1X16 DEMUX using 1X4 DEMUX



1X16 DEMUX



logicckt Implementation

=> In 1X16 DEMUX Input is one and the selection lines are four ie S_0, S_1, S_2, S_3 and the output lines are $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, D_{13}, D_{14}, D_{15}$ ...

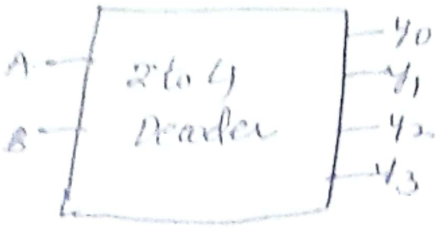
=> We can Implement the 1X16 DEMUX using 1X4 DEMUX

=> The truth Table of 1X16 DEMUX we can observe clearly.

1X16 Demux Truth table

S_3	S_2	S_1	S_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_8
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
10	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
11	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
12	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Design a 2 to 4 Decoder using NAND gates.



Actual 2 to 4 Decoder outputs are

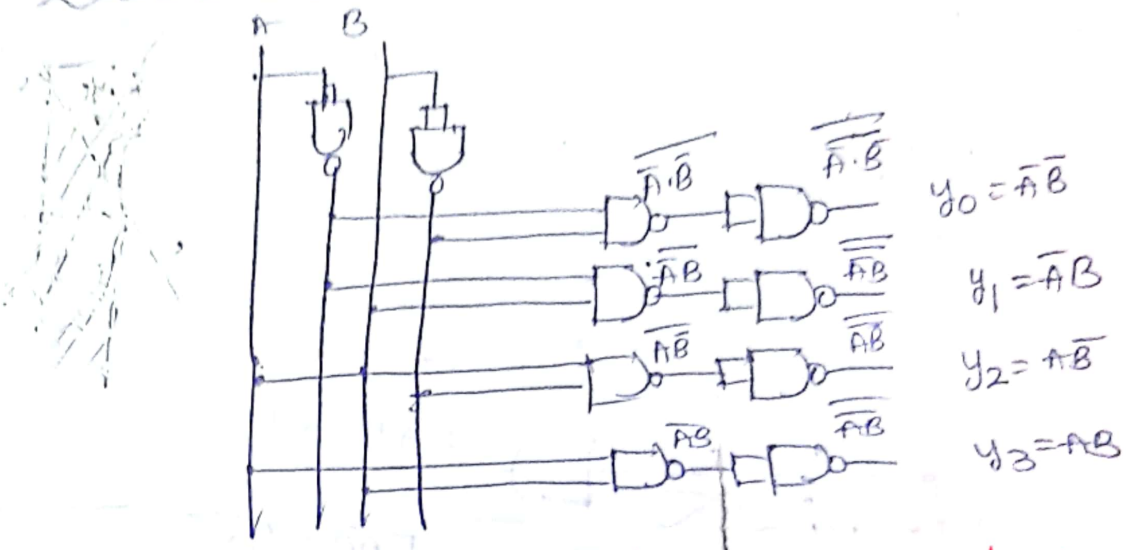
$y_0 = \bar{A}\bar{B}$

$y_1 = \bar{A}B$

$y_2 = A\bar{B}$

$y_3 = AB$

By using NAND gates

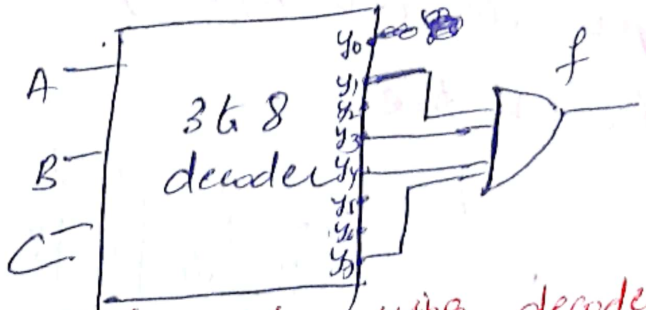


Imp.

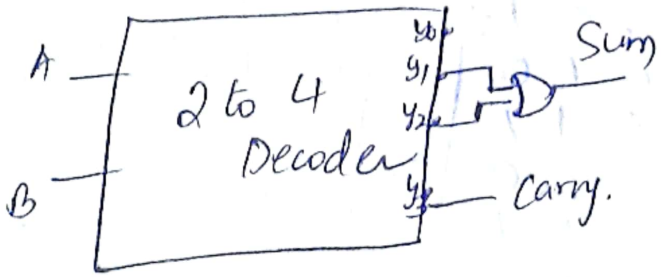
* Implement the given boolean function using decoder

$f = sm(1,3,4,7)$

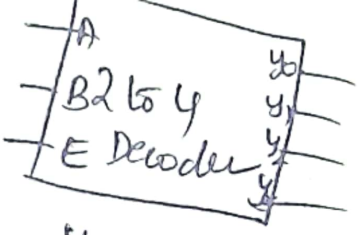
By using 3-to-8 decoder we can implement the given Boolean function



* Design Half adder using decoder $S = \Sigma m(1,2)$; $C = \Sigma m(3)$



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



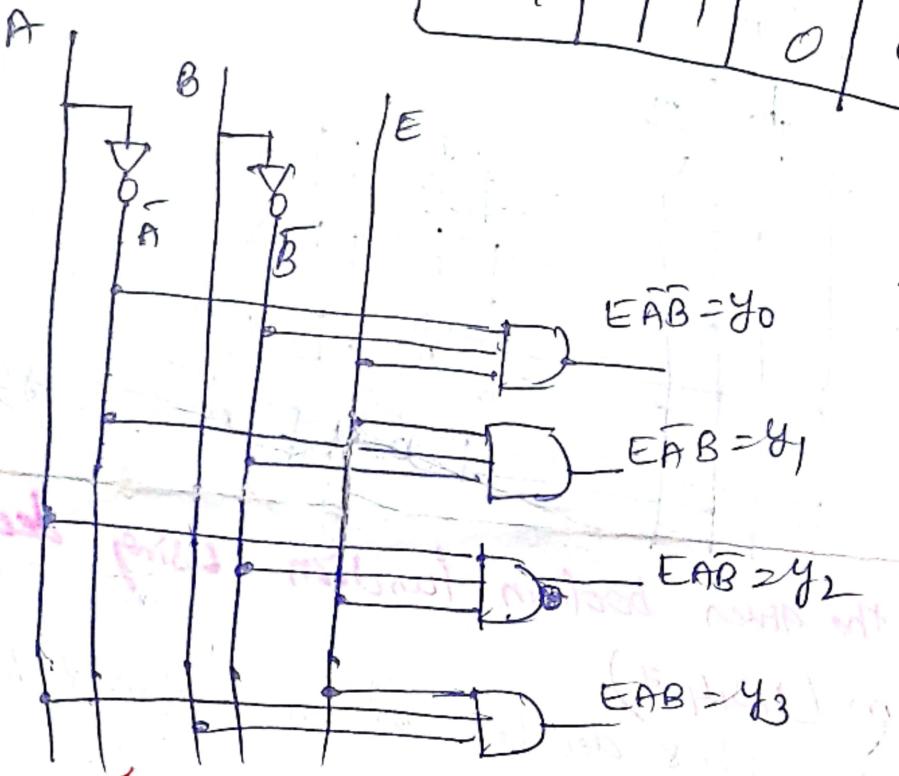
$y_0 = E\bar{A}\bar{B}$

$y_1 = E\bar{A}B$

$y_2 = EA\bar{B}$

$y_3 = EAB$

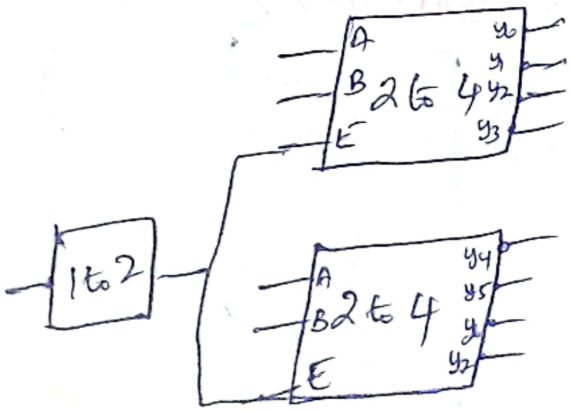
A	B	E	y_0	y_1	y_2	y_3
X	X	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1



* Decoder Tree *.

→ Designing of larger decoder by using smaller decoder is known as decoder tree

→ (1) Design 3 to 8 decoder by using 2 to 4 decoder



A B C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1

HDL for combinational circuits :-

HDL is Hardware description languages describe the architecture and behaviour of discrete and integrated electronic systems. Modern HDL's & their associated simulators are very powerful tools for integrated circuit designers

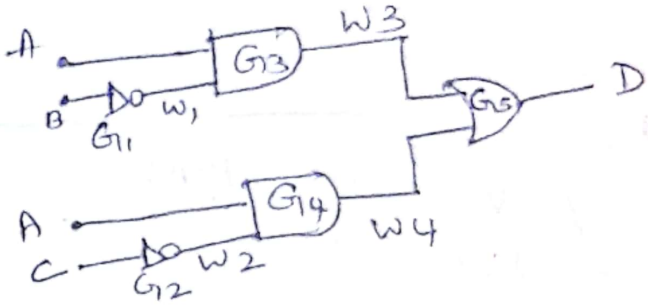
→ The main reason HDL in modern design methodology are

- Design functionality can be verified easily in the design process. Design simulations at higher level, before implementation at the gate level, this allows you to evaluate architectural and design decisions

// Verilog HDL program

```
module sample (D, A, B, C)
 output D;
 input A, B, C;
 wire w1, w2, w3, w4;

 not G1 (w1, B);
 not G2 (w2, C);
 and G3 (w3, A, w1);
 and G4 (w4, A, w2);
 or G5 (D, w3, w4);
```



Programs:-

① Half adder

```
module half_adder (
 input A, B,
 output sum, carry
);
```

assign {cout, sum} = A+B;

endmodule

(2) Decoder

```
module decoder_2to4(
input A,B,
output y0, y1, y2, y3
);
assign y0 = ~A & ~B;
assign y1 = ~A & B;
assign y2 = A & ~B;
assign y3 = A & B;
endmodule
```

(3) multiplexer

```
module MUX_4to1(
input A,B,C,D,sel,
output y
);
assign y = (sel==0)?A:
(sel==1)?B:(sel==2)?C:D;
endmodule.
```

(4) Encoder

```
module encoder_4to2(
input A,B,C,D,
output y1, y0
);
assign {y1, y0} = (A)?2'b01:
(B)?2'b10:(C)?2'b11:2'b00;
endmodule.
```

The module encoder_4to2 defines the inputs (A,B,C,D) and outputs (y0, y1)

The assign statement uses a conditional expression to determine the output values based on the input that is '1' if A is '1', the o/p is 2'b01. If B is '1', the output is 2'b10; If C is '1', the output is is 2'b11 If none of A, B or C is '1', the output is 2'b00.