UNIT-2

Combinational Circuits

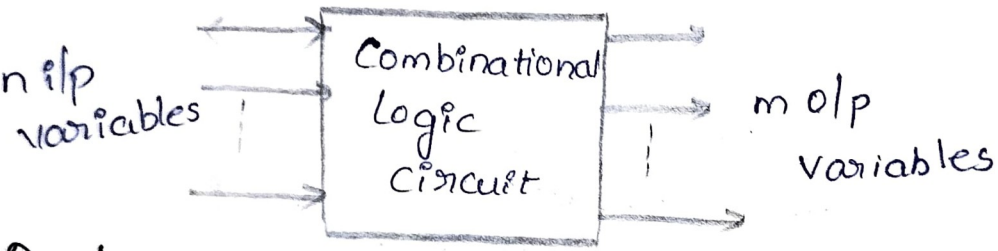
Combinational circuits :-

The o/p of a combinational circuits depends o present inputs only.

A combinational circuit consists of input variables, logic gates and output variables.

When logic gates are connected together T produce a specified output for certain Specified combination of input variables w no storage involved The resulting circuit is called combinational logic circuits.

In combinational logic the output variables a at all times dependent on the combination of input variables.



Analysis procedure:-

The analysis procedure of combinational circu involves two major steps they are

1) To determine the boolean function
2) To construct the truth table.

To determine the boolean function:-

The sequence of steps involved in obtaining Th boolean function is

Step-1:- Initially analyze the circuits based o the logic gates. If the circuit consist of logic gates with absence of feedback pat then the circuit is combinational circuit otherwise the circuit is a sequential circui

Step-2 :- Based on the input variables label the respective gate outputs with symbols and determine their boolean function.

Step-3 :- Step-2 is repeated until the boolean function for all the outputs are obtained

Step-4 :- finally determine the output boolean function in terms of input variables by substituting all the previously obtained boolean function.

To construct the truth table :-

The various steps involved in the construction of Truth table.

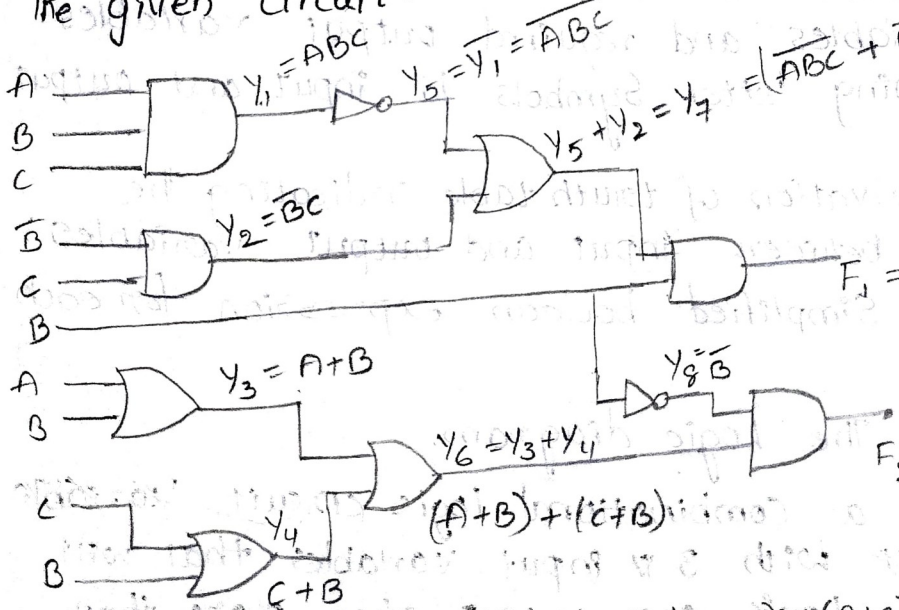
Step-1 :- List the binary numbers based on the no. of input variables

Step-2 :- indicate the symbols labelled at the respective gates in the truth table.

Step-3 :- construct the truth table for the outputs of those gates that are the function of input variables.

Step-4 :- finally step-3 is repeated until the truth table for all the output is determined.

1. Obtain the boolean function for outputs of the given circuit.



$F_1 = B(\overline{ABC} + \overline{B}C)$

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

$= \overline{A}B + B\overline{C}$

$F_2 = \overline{B}((A+B) + (B+C))$

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

$= A\overline{B} + C\overline{B}$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	F_1	F_2
0	0	0	0	0	0	0	1	0	1	1	0	0
0	0	1	0	1	0	1	1	1	1	1	0	1
0	1	0	0	0	1	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	1	1	0	1	0
1	0	0	0	0	1	0	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	1	1	0	1
1	1	0	0	0	1	1	1	1	1	0	1	0
1	1	1	1	0	1	1	0	1	0	0	0	0

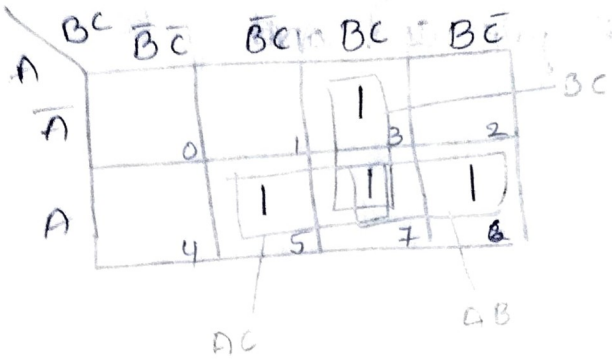
Design procedure:-

The design of combinational circuit starts from the outline of the problem statement and ends in a logic circuit diagram or set of boolean functions from which the logic diagram can be easily obtain. The design procedure of the combinational circuit, involved following steps

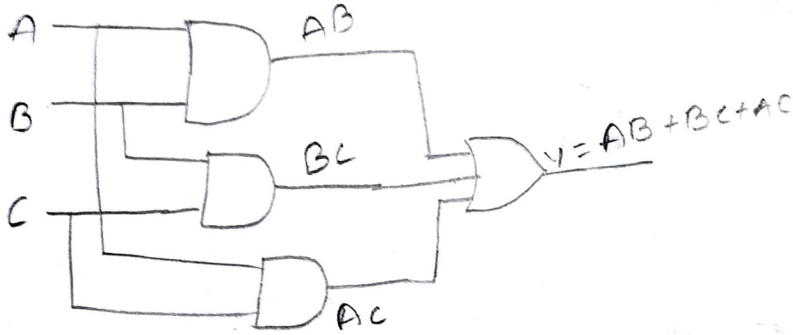
* The problem definition.
* The determination of number of available input variables and required output variables.
* Assigning letter symbols to input and output variables.
* The derivation of truth table indicating the relation between input and output variables.
* Obtain simplified boolean expression for each output.
* obtain the Logic diagram.

1. Design a combinational logic circuit ~~that~~ with 3 input variables that will produce a Logic one output when more than one input variables are logic one.

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



$Y = AC + AB + BC$



Adders :-

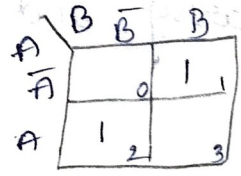
Digital computers perform various of information processing tasks one is arithmatic operation and the most basic arithmatic operation is the addition of two binary digits.

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

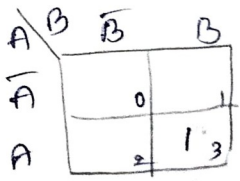
Half Adder :-

It is a combinational circuit with two binary inputs (augend and addend) and two binary outputs (Sum and carry). It adds the two inputs A and B and produces the Sum (S) and the carry (C) bits.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

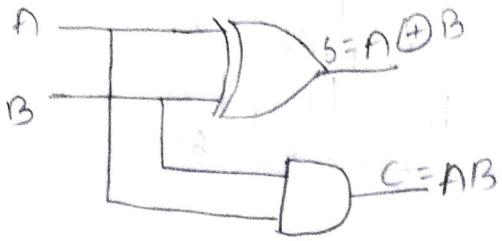


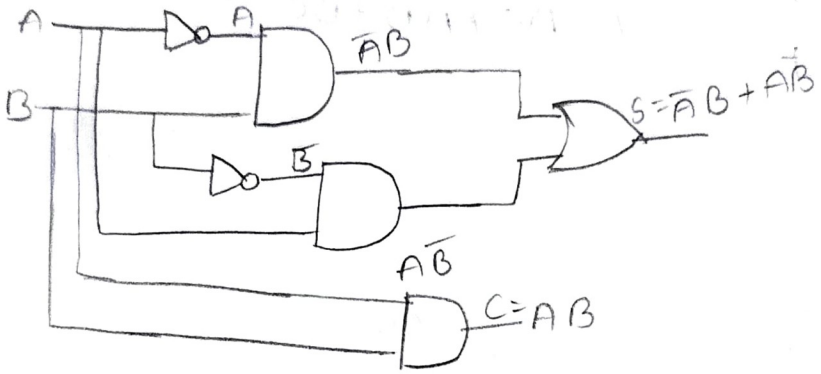
$S = \bar{A}B + A\bar{B}$

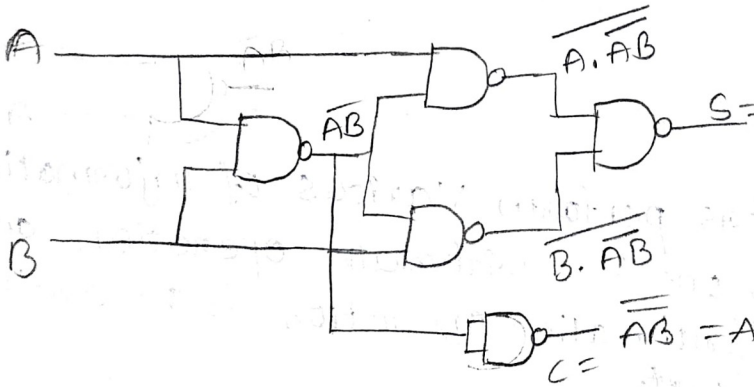


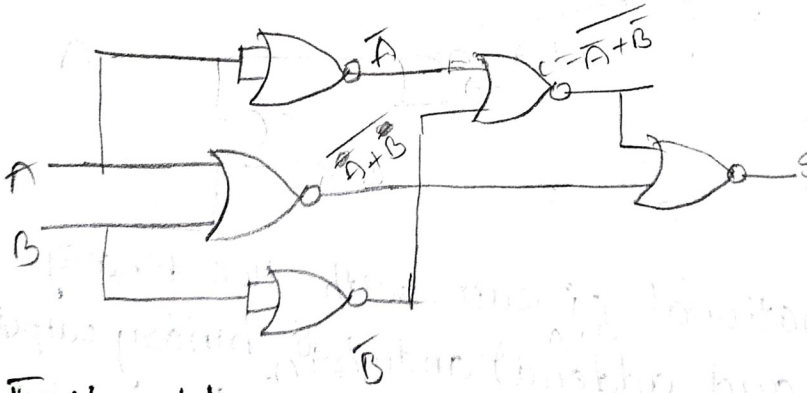
$C = AB$

A half adder can be realized by using one X-OR gate and one AND gate







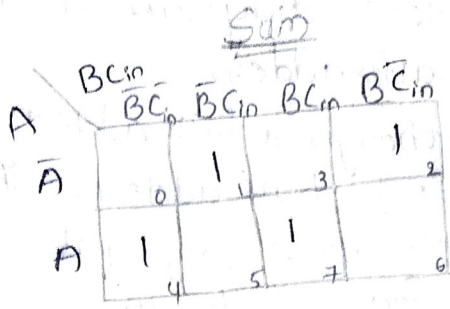


Full Adder:-

It is a combinational circuit that adds two bits and a carry and outputs are Sum and Carry.

A	B	C_{in}	Sum	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

Sum



Sum :-

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

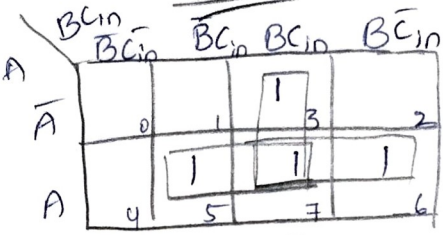
$$= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(\bar{A}B + A\bar{B})$$

$$= C_{in}(A \odot B) + \bar{C}_{in}(A \oplus B)$$

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

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

Carry

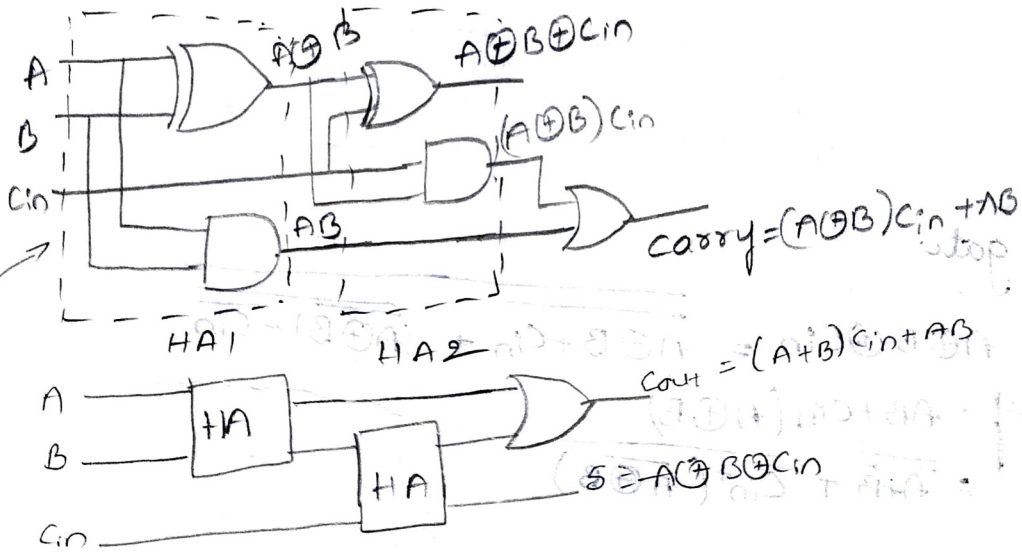


$$Carry = BC_{in} + AC_{in} + AB$$

$$carry = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$

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

$$= C_{in}(A \oplus B) + AB$$



Logic diagram of a full adder using two half adders.

The block diagram of a full adder using two half adders

Even though a full adder can be constructed using two half adders the disadvantage is that the bits must propagate through several gates in accession. which makes the total propagation delay greater than that of the full adder circuit using AOI logic.

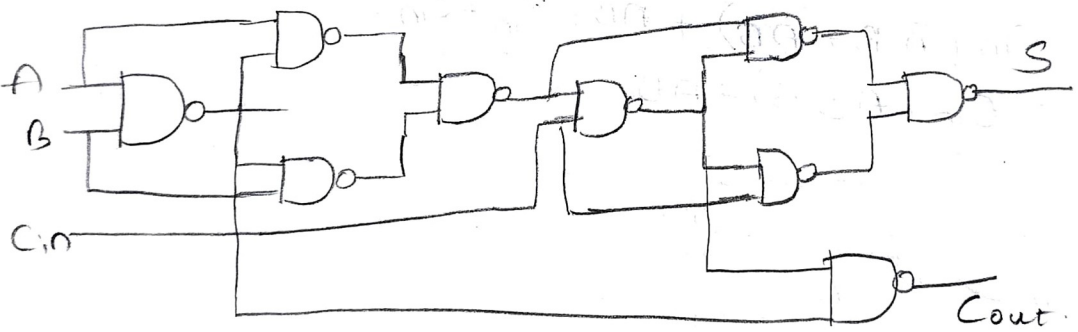
The full adder neither can also be realized using universal gates.

Using NAND logic

$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

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

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

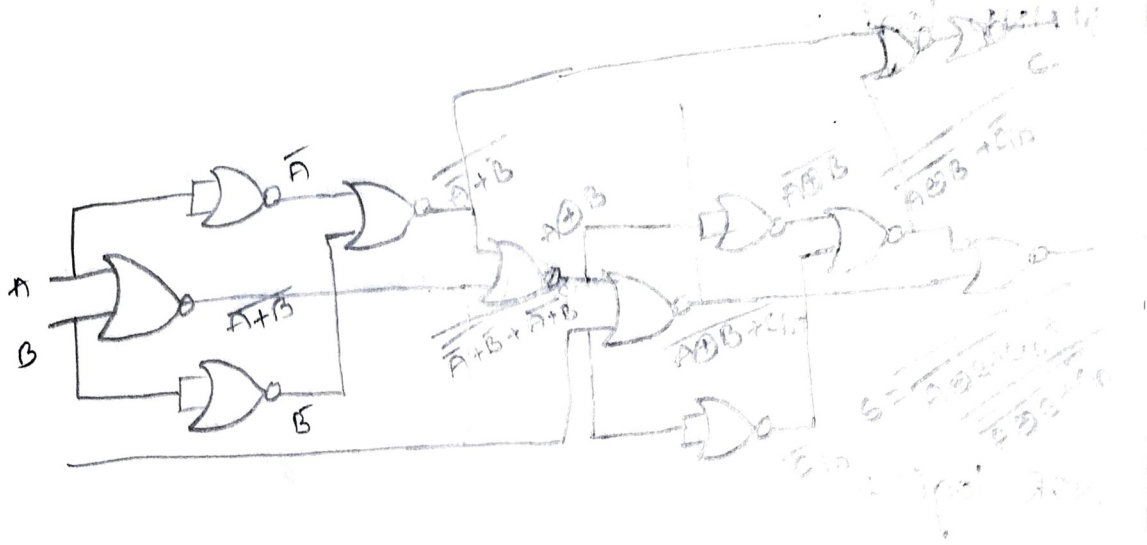


NOR gate.

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

$$\text{Carry} = AB + C_{in}(A \oplus B)$$

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

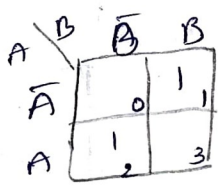


Subtractor:-

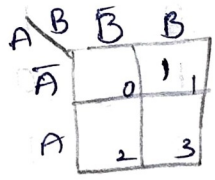
Half Subtractor:-

It is a combinational circuit that subtract one bit from the other and produces the difference it has also has an output to specify if a one has been borrowed. It has two inputs A and B and two outputs d and b. 'd' indicates the difference and b is borrow.

A	B	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

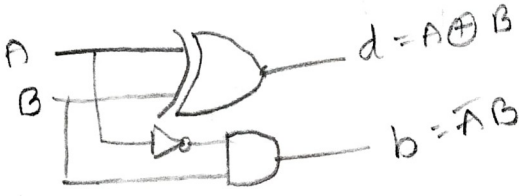


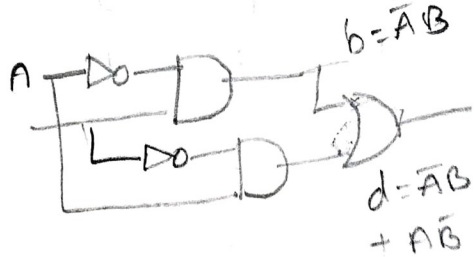
$d = \bar{A}B + A\bar{B}$



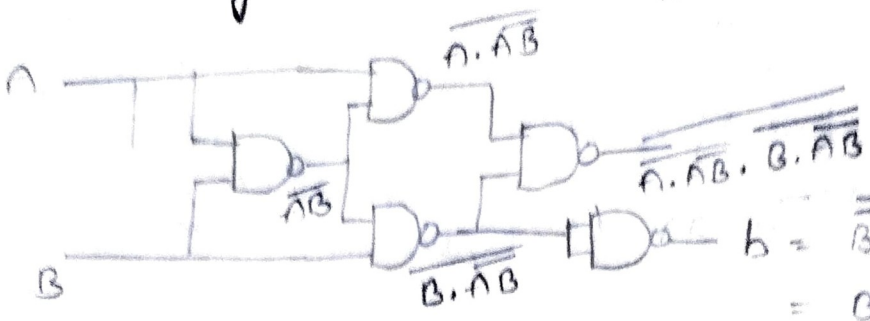
$b = \bar{A}B$

$d = \bar{A}B + A\bar{B} = A \oplus B$, $b = \bar{A}B$

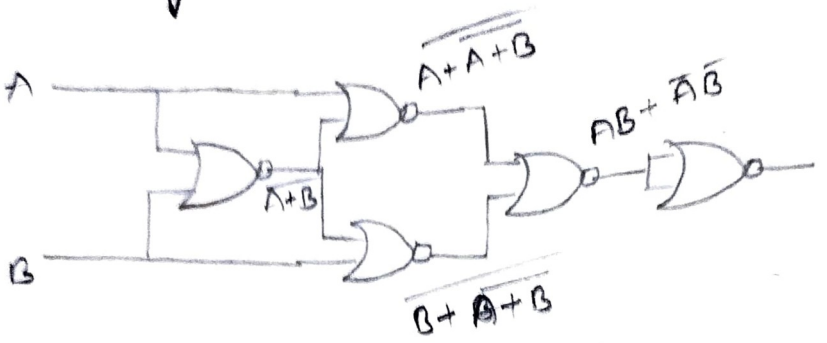




NAND logic :-



NOR logic :-

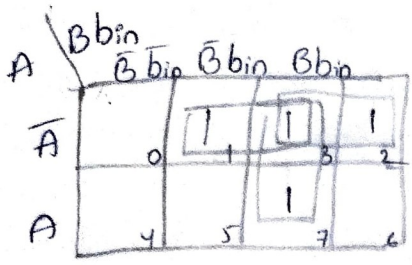


Full Subtractor :-

It is a combinational circuit used to perform Subtractor among 3 bits

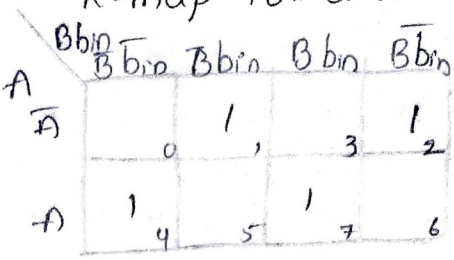
A	B	bin	Borrow	diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map for borrow :-



$\bar{A} b_{in} + \bar{A}B + B b_{in}$

K-map for diff :-



$\bar{A}B b_{in} + \bar{A}B\bar{b}_{in} + A\bar{B}\bar{b}_{in} + AB b_{in}$

$= b_{in}(\bar{A}\bar{B} + AB) + \bar{b}_{in}(\bar{A}B + A\bar{B})$

$= b_{in}(A \odot B) + \bar{b}_{in}(A \oplus B)$

$= A \oplus B \oplus b_{in}$

Based on truth tables :-

(Borrow)

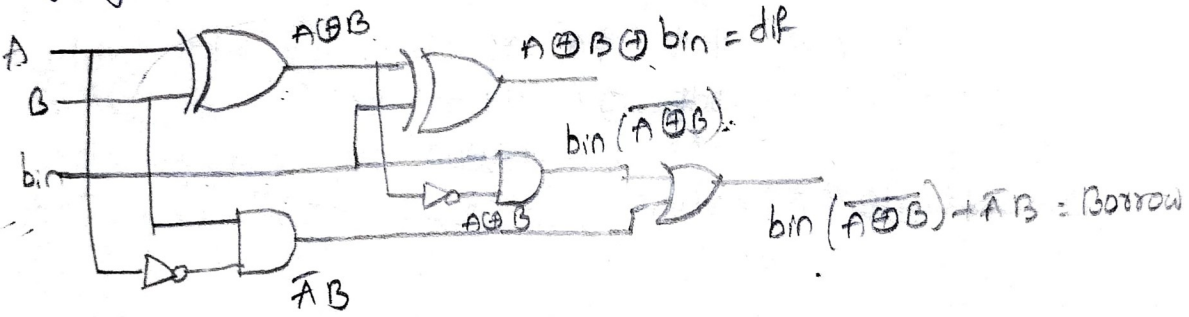
$= \bar{A}\bar{B} b_{in} + \bar{A}B\bar{b}_{in} + \bar{A}B b_{in} + AB b_{in}$

$= b_{in}(\bar{A}\bar{B} + AB) + \bar{A}B(\bar{b}_{in} + b_{in})$

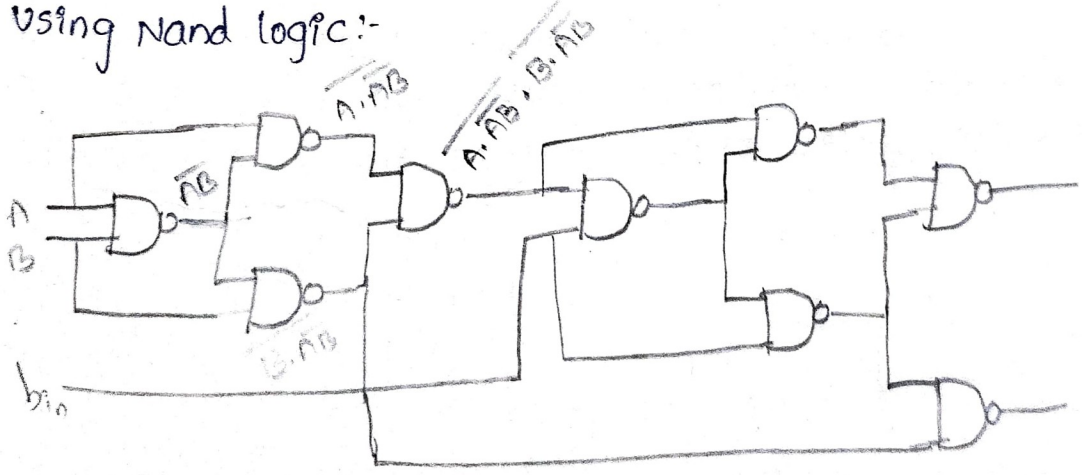
$= b_{in}(\overline{A \oplus B}) + \bar{A}B,$

Diff :- $\bar{A}\bar{B} b_{in} + \bar{A}B\bar{b}_{in} + AB\bar{b}_{in} + AB b_{in}$

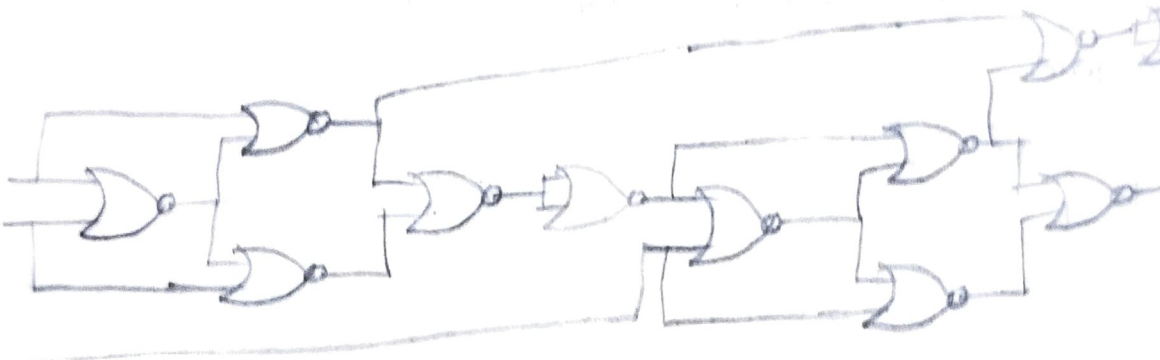
Logic gate :-



Using Nand logic :-

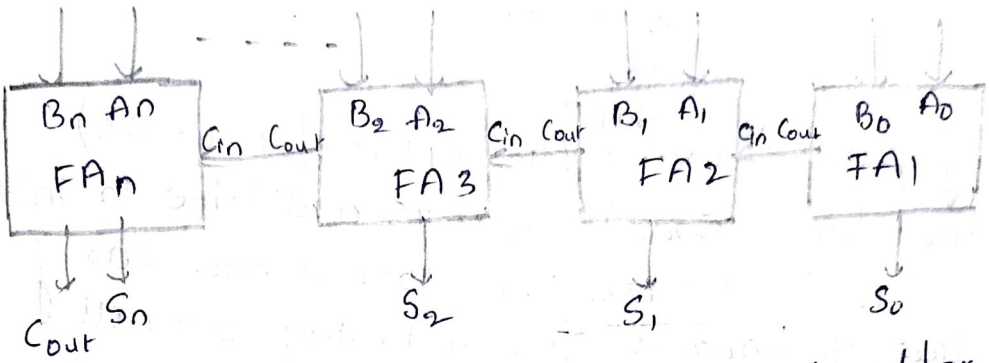


NOR



Binary adder :- / parallel adder

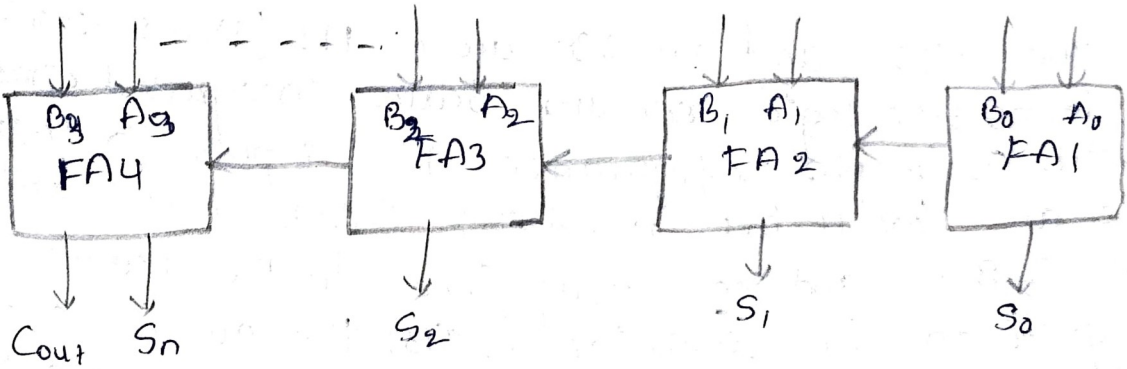
A single full adder is capable of adding two one bit number and an input carry in order to add binary numbers with more than one bit additional full adders must be employed. A n-bit parallel adder can be constructed using number of full adder circuit connected in parallel

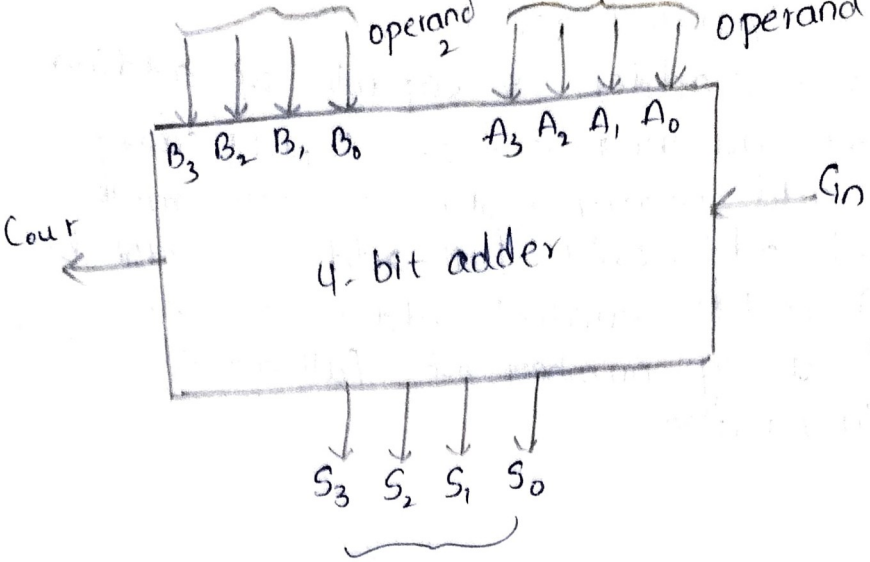


Block diagram of n-bit parallel adder.

It shows the block diagram of n-bit parallel adder using number of full adder circuits connected in cascade i.e the carry output of each adder is connected to the carry input o the next higher order adder. It should be noted that either a half adder can be used f the least significant position or the carry input of a full adder is made zero because there i no carry into the least significant bit position

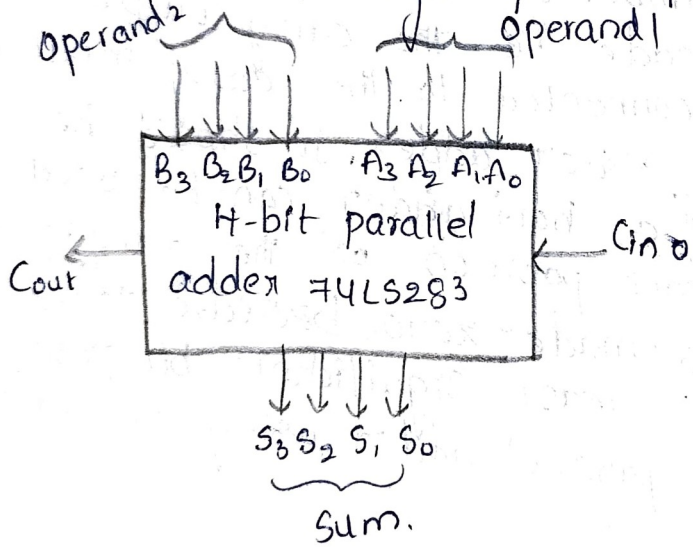
1. Design a 4-bit parallel adder using full adder





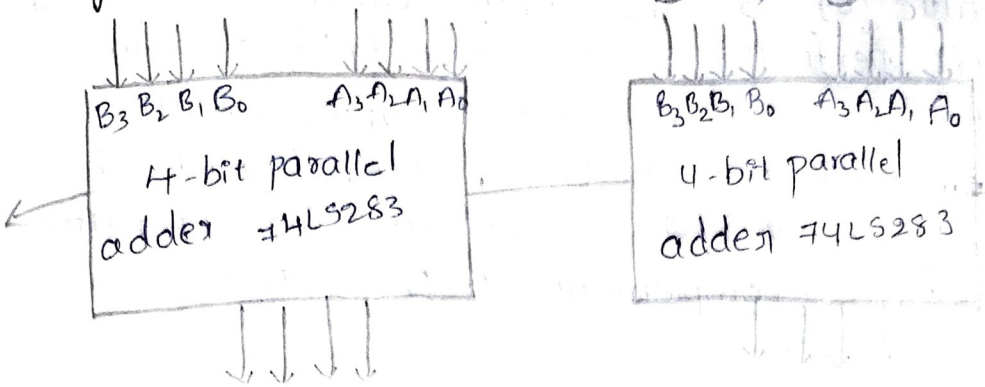
Binary parallel adder (IC 74LS83/74LS283)

Many high speed adders available in integrated circuit form utilize the look ahead carry or a similar technique for reducing overall propagation delays. It contains 4 interconnected full adders and the look ahead carry circuit the needed for high speed operation.



The 7483 and 74283 are a TTL (transistor to transistor logic) medium scale integrated circuit with same pin configuration the inputs to this IC are two 4-bit numbers A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0 and the carry C_{in} 0 into the least significant bit position (LSB) the outputs are the S_0, S_1, S_2, S_3 and C_{out} output of the most significant bit position (MSB)

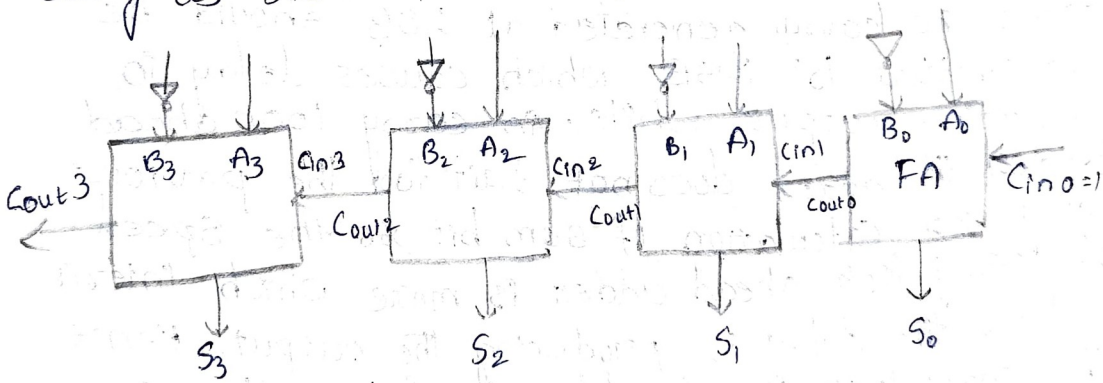
Design an 8-bit adder using two 74283



Binary Subtractor (parallel subtractor):-

The Subtraction of binary numbers can be done most conviently by means of compliments, The Subtraction of A-B can be done by taking the 2's complement of B and adding it to A.

The 2's complement can be obtained by adding the taking the 1's complement and adding 1 to the Least significant pair of bits the 1's complement can be implemented with inverters and a '1' can be added to the sum through the input carry as shown in the figure.

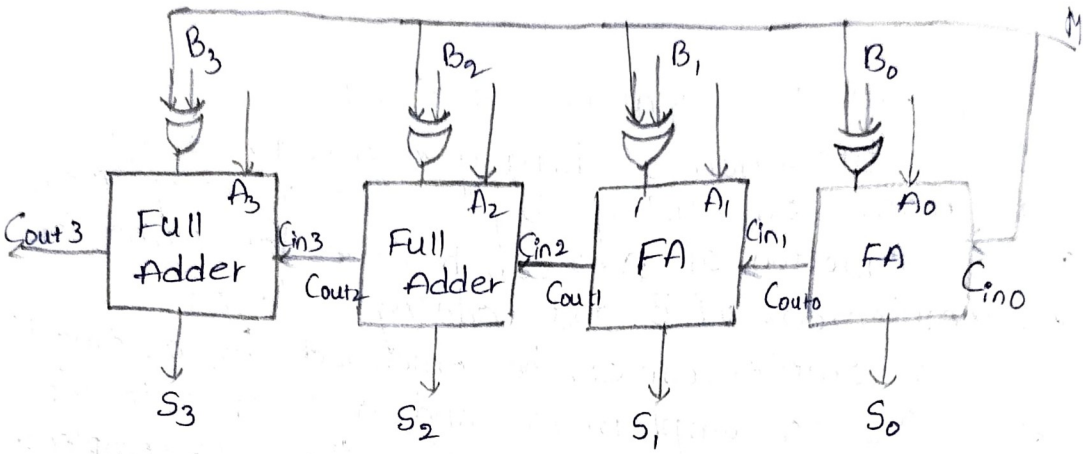


Parallel adder/subtractor:-

The addition and subtraction operation can be combine into one circuit with one common binary adder this is done by including an X-OR gate with each full adder.

The mode input 'M' controls the operation of the circuit when M=0 the circuit is an adder and when M=1 the circuit becomes a subtractor. Each X-OR gate receives input M and one of the input of B. when M = 0

We have $B \oplus 0 = B$ the full adder receives the value of B. The input carry is 0 and the circuit performs A+B. when M=1 we have $B \oplus 1 = \bar{B}$ and $C_{in0} = 1$ The B inputs are all complemented and a '1' is added through the input carry. The circuit performs the operation A plus the 2's complement of B i.e A-B



Carry look ahead adder:-

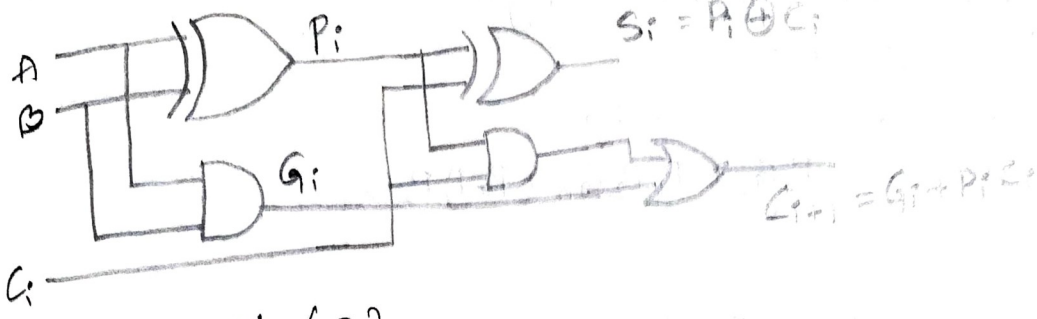
In ripple carry adder as the number of bits increases the speed of the adder is decreased since the carry generated at LSB should be transmitted to MSB which causes delay in producing output while in carry look ahead adder the adder does not wait for the previous carry for calculation of sum bit so the speed of carry look ahead adder is more which intern reduces the delay in producing the output. Hence the carry look ahead adder is faster than a ripple carry adder.

Operation of carry look ahead adder:-

The circuit used to speedup the addition process by reducing the time during intermediate stage is referred as carry look ahead adder The carry look ahead adder employes two functions namely 1) carry propagate.

2) carry generate

In order to define these functions the circuit of full adder is considered as shown in figure.



carry propagate (P_i) :-

The function of P_i is to determine whether a carry into stage 'i' propagates into stage i+1. From figure the expression of P_i can be written as $A_i \oplus B_i = P_i$

Carry generate (G_i) :-

The function of G_i is to generate a carry when inputs A_i, B_i are 1 from figure The expression of G_i can be written as $G_i = A_i . B_i$

The carry out C_{i+1} can be expressed as $C_{i+1} = G_i + P_i C_i$ and output sum S_i if $S_i = P_i \oplus C_i$

for different stages the boolean functions for carry outputs are

Stage (i)	Carry out C_{i+1}
$i=0$	$C_1 = G_0 + P_0 C_0$
$i=1$	$C_2 = G_1 + P_1 C_1$
	$= G_1 + P_1 [G_0 + P_0 C_0]$
	$= G_1 + P_1 G_0 + P_0 P_1 C_0$
$i=2$	$C_3 = G_2 + P_2 C_2$
	$= G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_0)$
	$= G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_0$

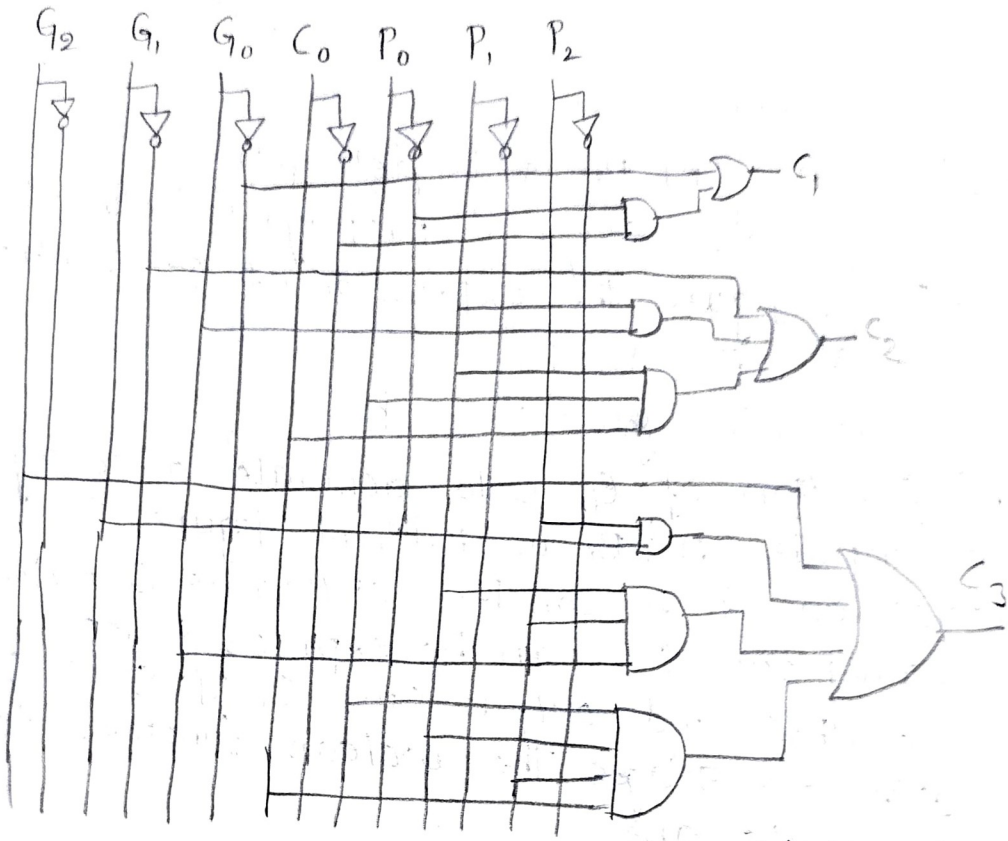
It is observe that the C_3 does not contain C_2 and C_1 terms i.e the propagation time of C_3 collaborates with the propagation time of C_2 and C_1 as a result the processing time is reduced thereby increasing the speed of operation.

The logic diagram of carry look ahead adder can be implemented by using AND and OR gates as shown in figure.

$C_1 = G_0 + P_0 C_0$

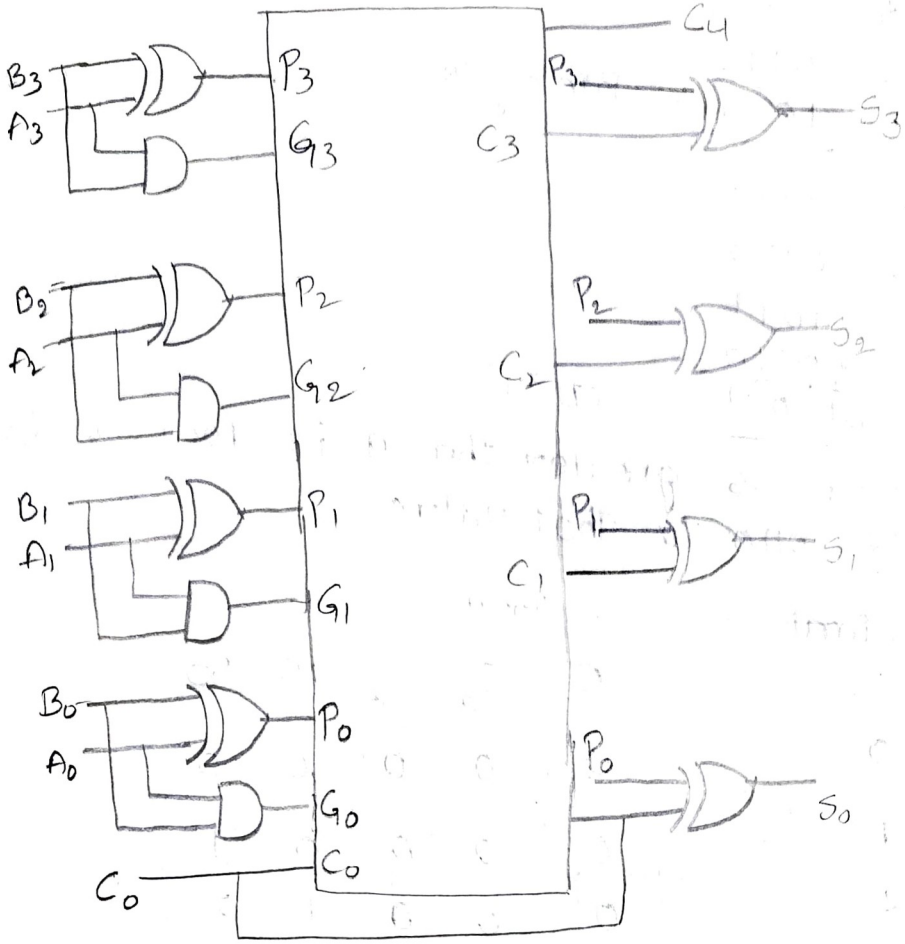
$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_0$



4-bit binary carry look ahead adder :-

The 4-bit carry look ahead adder employees X-OR and AND gates. Two X-OR gates and an AND gate are required to produce a single output for instance if A_i and B_i are given as inputs to first X-OR gate and AND gate. The output will be P_i and G_i the carries are generated using carry look ahead adder and then fed as a input to the second X-OR gate. This gate produces the corresponding outputs Sum.



BCD Adder :-

The digital system handles the decimal numbers in the form of binary coded decimal numbers. A BCD adder is a circuit that adds two BCD digits and produces a sum digit is also in BCD. BCD numbers use 10-digits (0-9) which are represented in the binary form to (0000 to 1001). Adding 4-bit BCD code.

In 4-bit we have a maximum value upto 9 We can add maximum value upto 9+9+1. Here '1' is the input carry $19 \Rightarrow 9+9+1$

Case i :- Total is less than or equal to 9

Eg :- $1 \rightarrow 0001$

$5 \rightarrow 0101$

$0110 \rightarrow 6$

upto 9 both BCD and binary are going to be same.

Case ii :- Total is greater than 9

```
  5 - 0101
(+) 6 - 0110
-----------
 11   1011   → Binary '11'
```

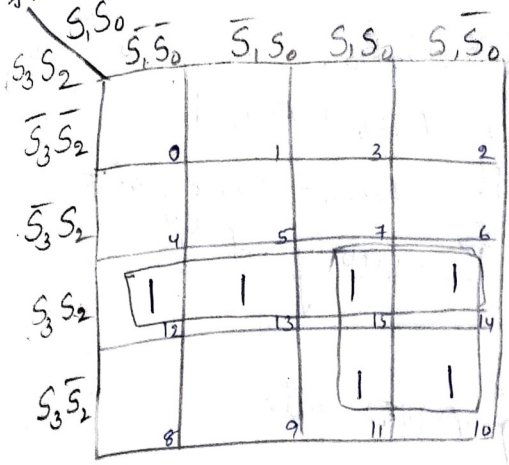
```
  5 - 0101
(+) 6 - 0110
-----------
      1011
    + 0110
-----------
  00010001   → BCD '11'
```

If value is greater than 9 i.e 1001 we have to add 0110 to that value.

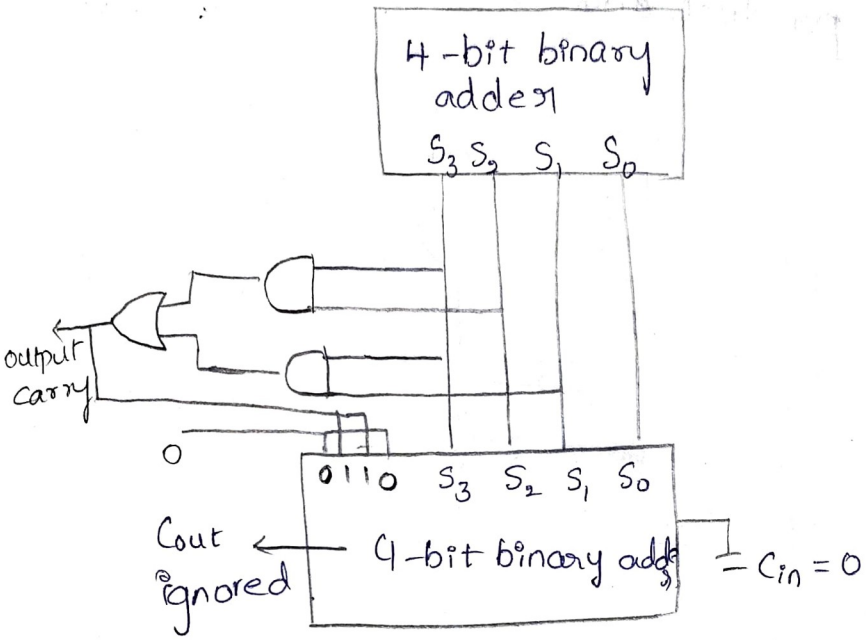
Decimal	BCD				
	C	S_3	S_2	S_1	S_0
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1

From truth table if C=0 comes under case i so no change is required. If C=1 it comes under case (ii) We have to add 0110 to the output. To find the relation we have to

find boolean function for 'c'



$y = S_3S_2 + S_3S_1$



Binary Multiplier:-

Binary multiplier is used Multiply two binary numbers. It is built using binary adders the two numbers are known as multiplicand and multiplier and the result is known as product

A → Multiplicand

B → Multiplier

A x B → product.

Eg:-

$$\begin{array}{r} 0\ 1\ 1 \\ 1\ 0\ 0 \\ \hline 0\ 0\ 0 \\ 0\ 1\ 1\ * \leftarrow \text{Left shift} \\ 0\ 1\ 1\ *\ * \leftarrow \text{lef shift} \\ \hline 1\ 0\ 0\ 1\ 0 \end{array}$$

product

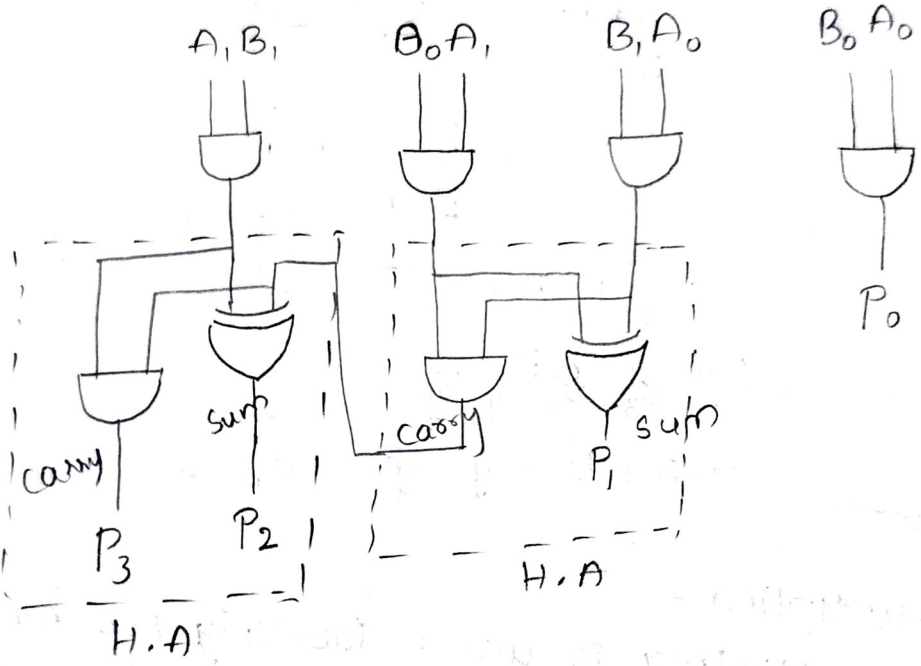
Multiplicand $A = A_1 A_0$

Multiplier $B = B_1 B_0$

$$P_0 = A_0 B_0$$
$$P_1 = A_1 B_0 + A_0 B_1$$
$$P_2 = A_1 B_1 + \text{carry of } P_1$$
$$P_3 = \text{carry of } P_2$$

$$\begin{array}{cccc} & & A_1 & A_0 \\ & & B_1 & B_0 \\ \hline & & A_1B_0 & A_0B_0 \\ + & A_1B_1 & A_0B_1 & \times \\ \hline P_3 & P_2 & P_1 & P_0 \end{array}$$

* partial product - using AND gate
* partial product, add - full adder / Half adder

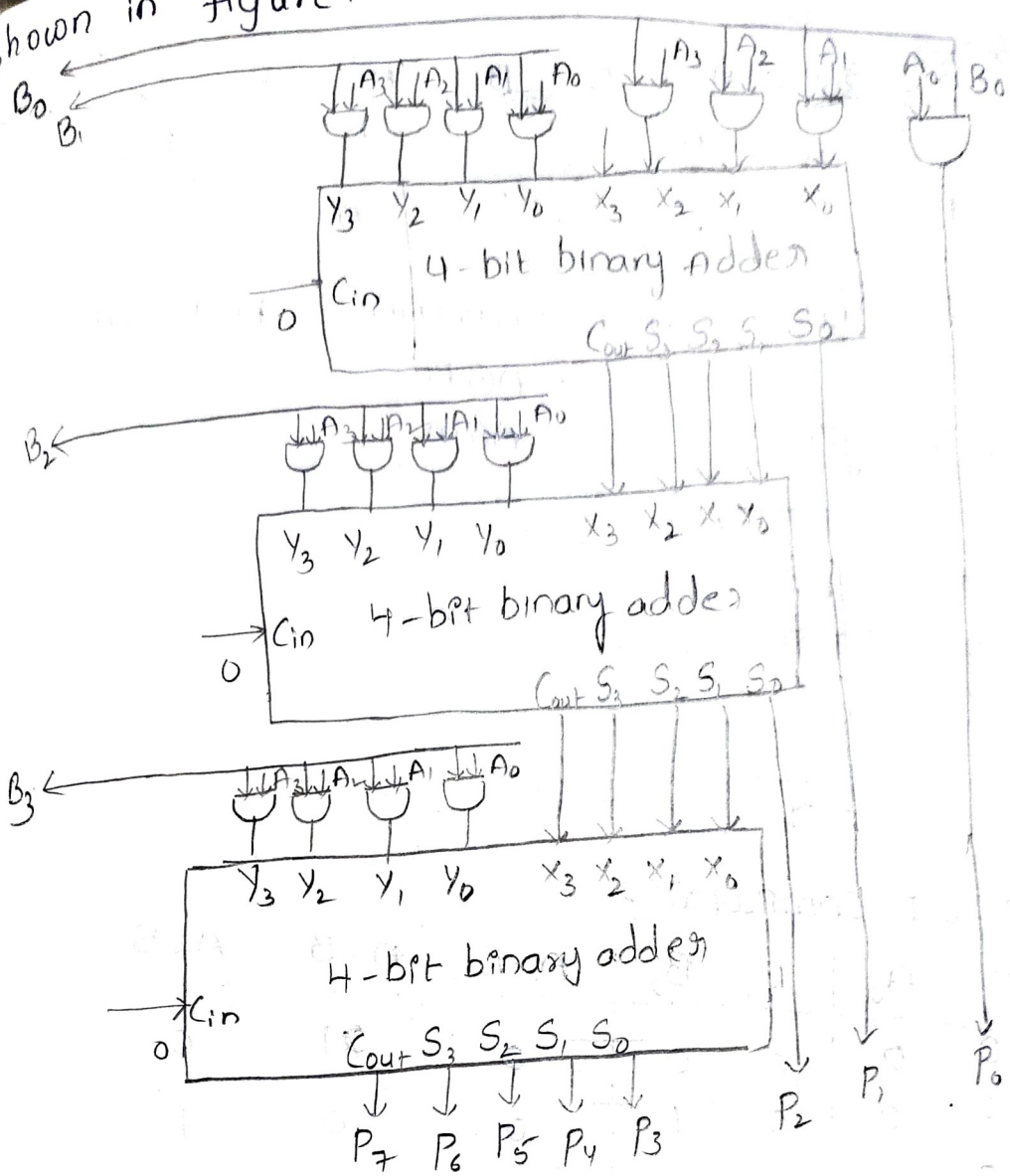


4-bit binary multiplier :-

$$\begin{array}{cccccccc} & & & A_3 & A_2 & A_1 & A_0 & \\ & & & B_3 & B_2 & B_1 & B_0 & \\ \hline & & & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 & \\ & & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 & + & \\ & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 & + & + & \\ A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 & + & + & + & \\ \hline P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0 \end{array}$$

In order to implement 4x4 binary mult we require 3 binary adders $4 \times 4 = 16$ AND The logic diagram of 4x4 binary multiplier.

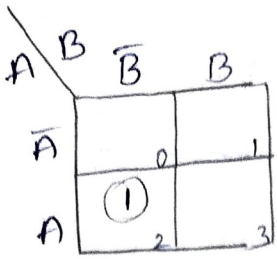
shown in figure.



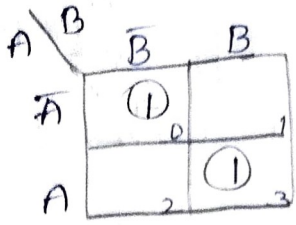
Magnitude comparator:-

A combinational circuit which compares magnitude of two binary number is called a comparator it has 2 inputs and 3 outputs If A and B are two inputs then there outputs are $A > B$, $A = B$, $A < B$. The truth table of 1 bit comparator as shown in below table

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



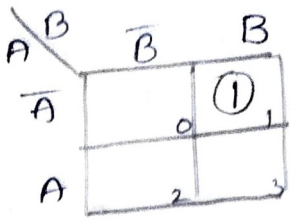
$A > B = A\bar{B}$



$A = B \Rightarrow \bar{A}\bar{B} + AB$

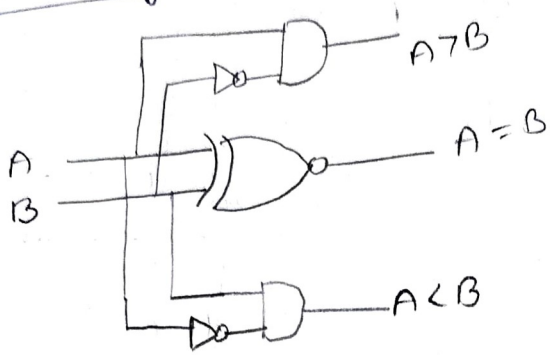
$= A \odot B$

$= \overline{A \oplus B}$



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

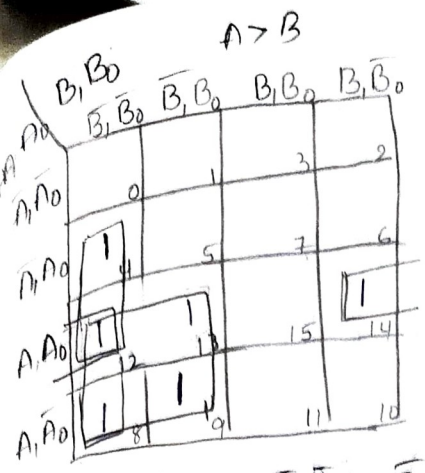
Logic Diagram:-



Two-bit comparator.

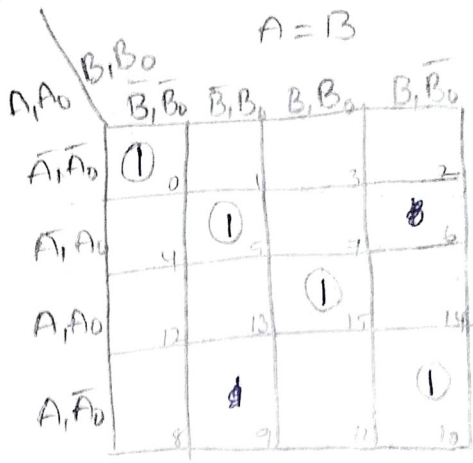
A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

A > B



$A > B = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + \bar{B}_0A_1A_0$

A = B



$A = B \Rightarrow$

$\bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0$

$+ A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$= \bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0)$

$+ A_1B_1(A_0B_0 + \bar{A}_0\bar{B}_0)$

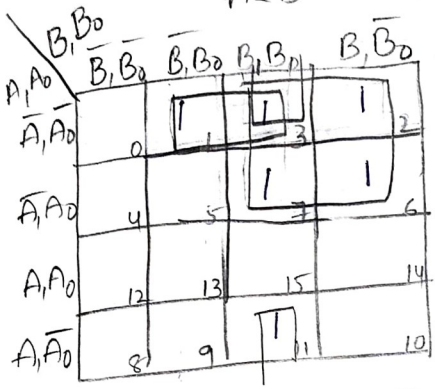
$= \bar{A}_1\bar{B}_1(A_0 \odot B_0)$

$+ A_1B_1(A_0 \odot B_0)$

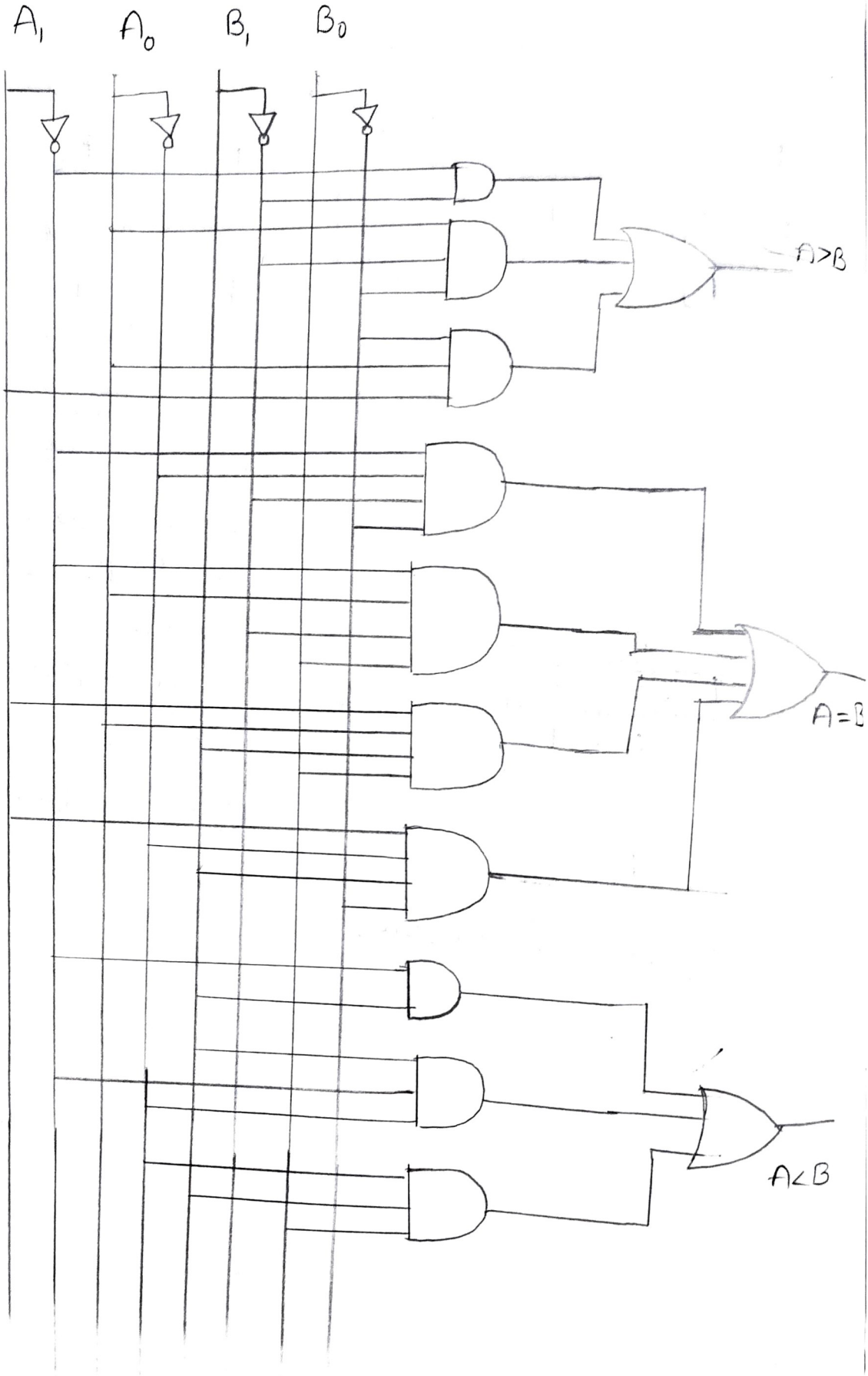
$= A_0 \odot B_0(\bar{A}_1\bar{B}_1 + A_1B_1)$

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

A < B



$A < B \Rightarrow \bar{A}_1B_1 + B_0\bar{A}_1\bar{A}_0 + \bar{A}_0B_1B_0$


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

4-bit comparator:-

A_3B_3	A_2B_2	A_1B_1	A_0B_0	$A>B$	$A=B$	$A<B$
$A_3>B_3$				1	0	0
$A_3=B_3$	$A_2>B_2$			1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_2$		1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_2$	$A_0>B_0$	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_2$	$A_0=B_0$	0	1	0
$A_3<B_3$				0	0	1
$A_3=B_3$	$A_2<B_2$			0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$		0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	0	0	1

⇒ It can be used to compare two four bit words.

⇒ The two 4-bits

It compare each of these bits in one number with bits in that of other number and produces one of the following outputs as $A=B$, $A<B$ and $A>B$. The output logic statements of this comparator are

1) If $A_3=1$ and $B_3=0$ then $A>B$
2) If A_3 and B_3 are equal and $A_2=1$, $B_2=0$ then $A>B$.
3) If A_3 and B_3 are equal and A_2 and B_2 are equal and $A_1=1$, $B_1=0$ then $A>B$.
4) If $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ and $A_0=1$, $B_0=0$ then $A>B$.

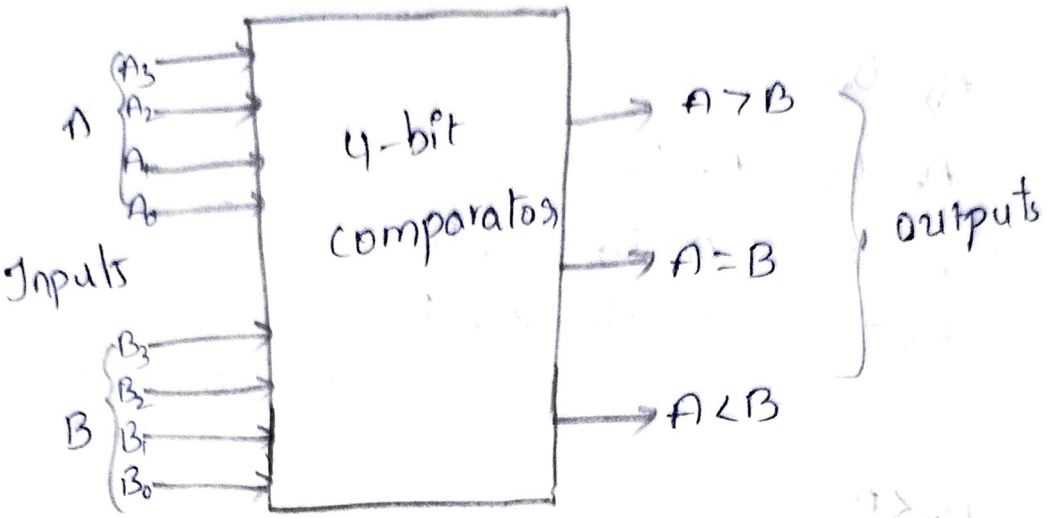
from the above statements the output logical expression can be written as

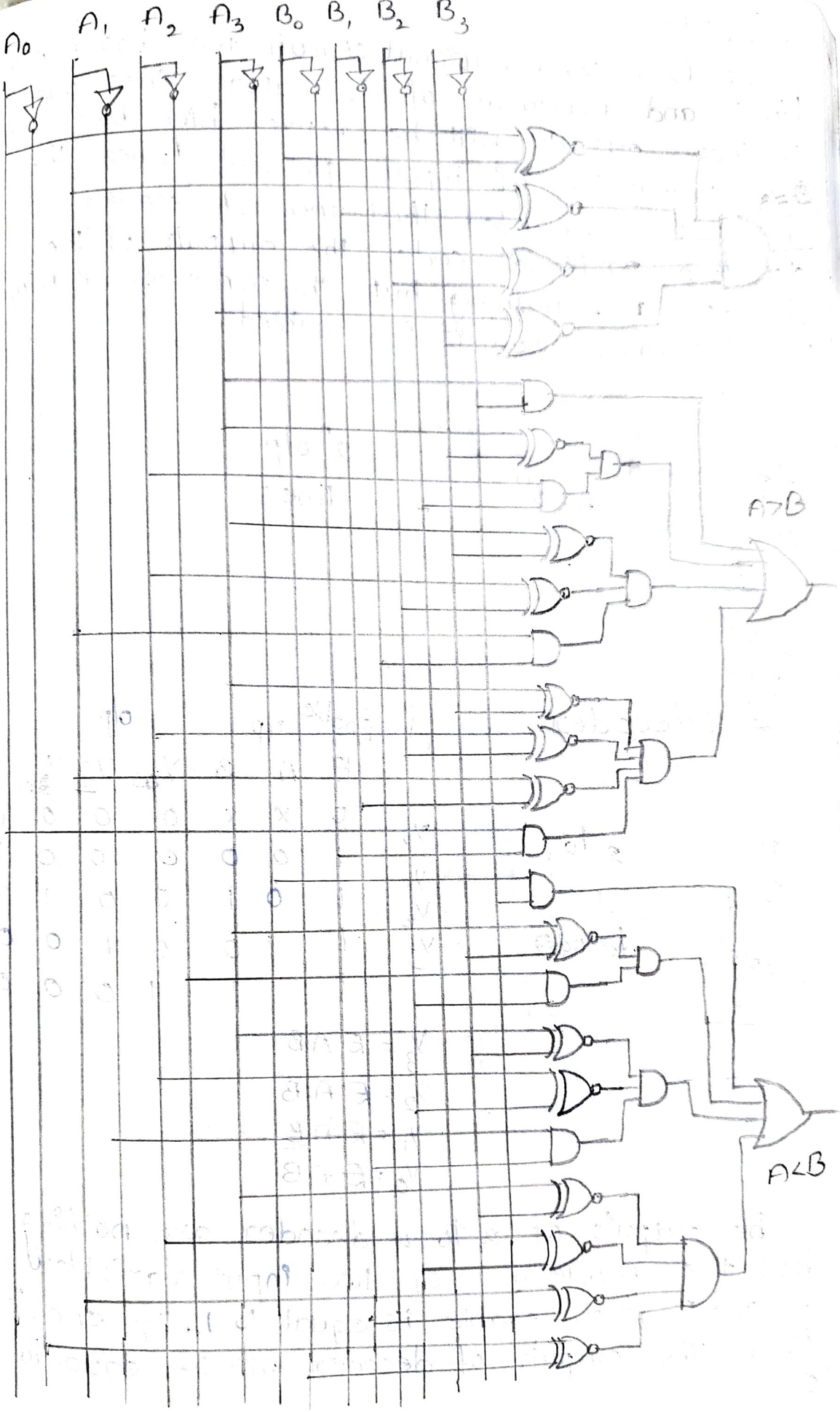
Equal to $= (A_0 \odot B_0)(A_1 \odot B_1)(A_2 \odot B_2)(A_3 \odot B_3)$

greater $= A_3\overline{B_3} + (A_3 \odot B_3)A_2\overline{B_2} + (A_3 \odot B_3)(A_2 \odot B_2)A_1\overline{B_1}$

$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0\overline{B_0}$

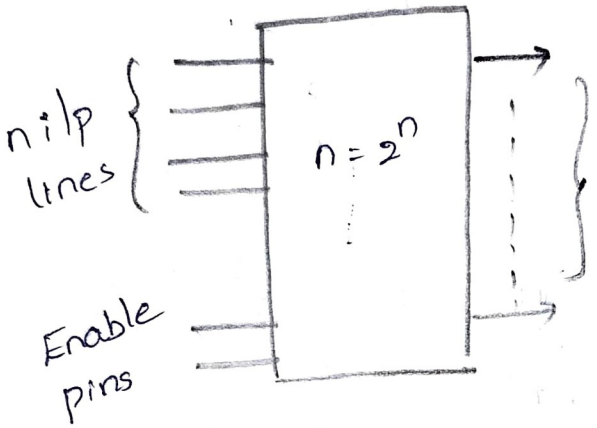
Less than = $\bar{A}_3 B_3 + (A_3 \odot B_3)\bar{A}_2 B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1 B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0 B_0$




A0
A1
A2
A3
B0
B1
B2
B3
A=B
A>B
A<B

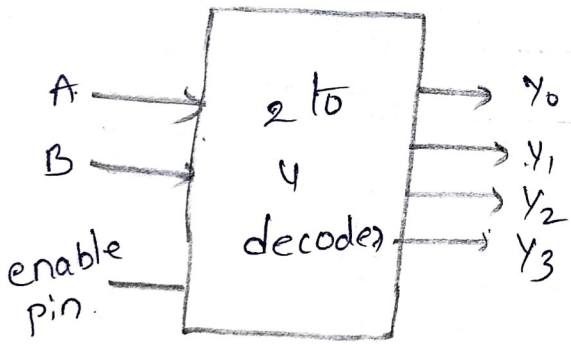
Decoder:-

It is a combinational circuit that has n input lines and maximum of 2^n output lines. one of these outputs will be active high based on the combination of inputs presents. when the decoder is enabled that means decoder detects a particular code the outputs of the decoder are nothing but the minterms of n input variable lines. when it is enabled.



active low - 0
active high - 1

2-4 decoder



Enable	i/p		o/p			
E	A	B	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

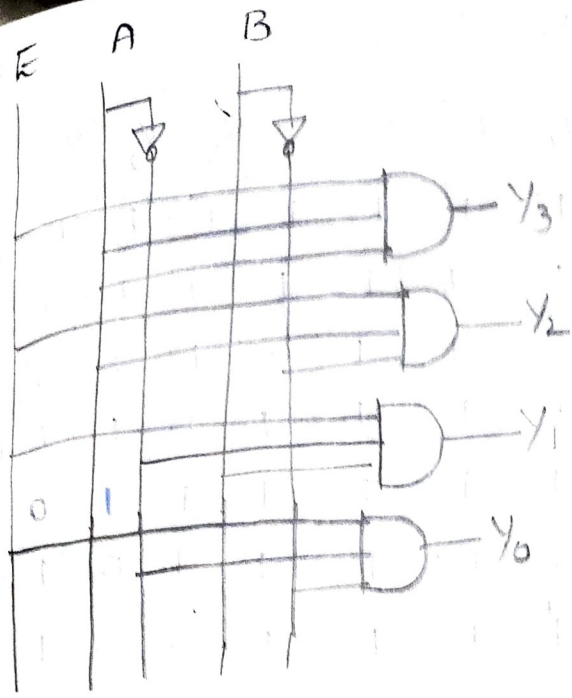
$Y_3 = E.AB$

$Y_2 = E.A.\overline{B}$

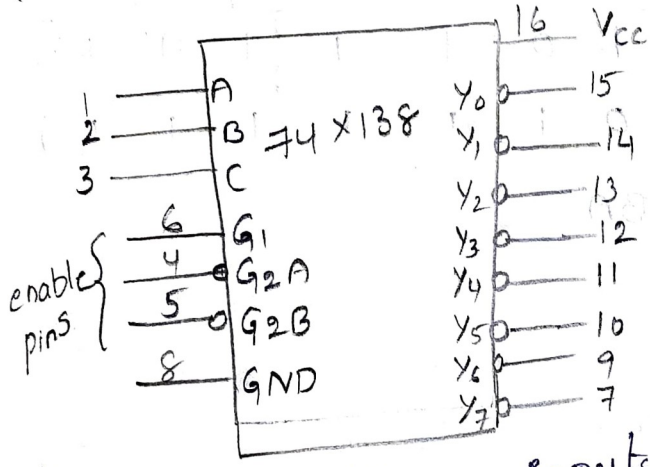
$Y_1 = E.\overline{A}B$

$Y_0 = E\overline{A}B$

The outputs of 2 to 4 decoder are nothing but the minterms of two input variables A and B when enable is equal to 1. If enable is 0 the outputs of decoder will be equal to '0'.



(74x138) 3 to 8 decoder



It consist 3 binary inputs A, B, C and when enable provides 8 individual active low outputs Y_0 - Y_7 the device has 3 enable input pins two active low G_2A, G_2B that mean when ~~when~~ low signals are applied to those pins they will be active and one active high G_1

G_{2B} G_{2A} G_1 C B

G_{2B}	G_{2A}	G_1	C	B	A	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y
1	x	x	x	x	x	1	1	1	1	1	1	1	
x	1	x	x	x	x	1	1	1	1	1	1	1	
x	x	0	x	x	x	1	1	1	1	1	1	1	
0	0	1	0	0	0	1	1	1	1	1	1	1	
0	0	1	0	0	1	1	1	1	1	1	1	0	
0	0	1	0	1	0	1	1	1	1	1	0	1	
0	0	1	0	1	1	1	1	1	1	0	1	1	
0	0	1	1	0	0	1	1	1	0	1	1	1	
0	0	1	1	0	1	1	1	0	1	1	1	1	
0	0	1	1	1	0	1	0	1	1	1	1	1	
0	0	1	1	1	1	0	1	1	1	1	1	1	

$Y_7 = G_{2B}G_{2A}G_1CBA$

$Y_6 = \overline{G}_{2B}\overline{G}_{2A}G_1\overline{A}BC$

$Y_5 = \overline{G}_{2B}\overline{G}_{2A}G_1A\overline{B}C$

$Y_4 = \overline{G}_{2B}\overline{G}_{2A}G_1\overline{A}\overline{B}C$

$Y_3 = \overline{G}_{2B}\overline{G}_{2A}G_1AB\overline{C}$

$Y_2 = \overline{G}_{2B}\overline{G}_{2A}G_1\overline{A}B\overline{C}$

$Y_1 = \overline{G}_{2B}\overline{G}_{2A}G_1A\overline{B}\overline{C}$

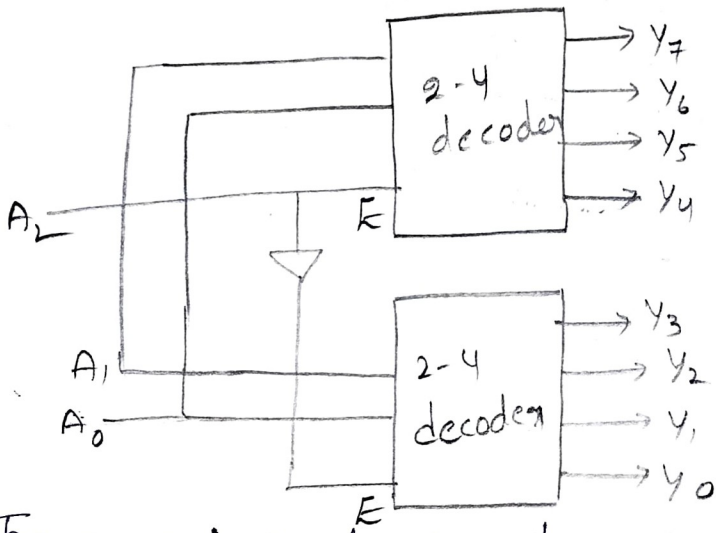
$Y_0 = \overline{G}_{2B}\overline{G}_{2A}G_1\overline{A}\overline{B}\overline{C}$


G_{2A}
G_{2B}
G_1
A
B
C

1. Design a 3-8 decoder using 2-4 decoders

In these 3-8 decoder using 2-4 decoders we know that 2-4 decoder has 2 inputs A_1 and A_0 and 4 outputs Y_3 to Y_0 where as 3-8 decoder as 3 inputs and A_2, A_1, A_0 and 8 outputs Y_7 to Y_0 We can find the no. of lower order decoders required for implementing higher order decoder using the following formula. Required number of lower ordered decoders $= \frac{m_2}{m_1}$

Where m_1 is the no. of outputs of lower order decoder, m_2 is the no. of outputs of higher order decoder. Here $m_1 = 4$ and $m_2 = 8 \Rightarrow \frac{8}{4} = 2$



The parallel inputs A_1 and A_0 are applied to each 2-4 decoder. The input of A_2 is connected to Enable E of lower 2-4 decoder in orides to get the outputs of Y_3 to Y_0. These are the lower 4 minterms the input A_2 is directly connected to enable of upper 2-4 decoder in order to get the outputs of Y_7 to Y_4. These are the higher 4 minterms for active high output:-

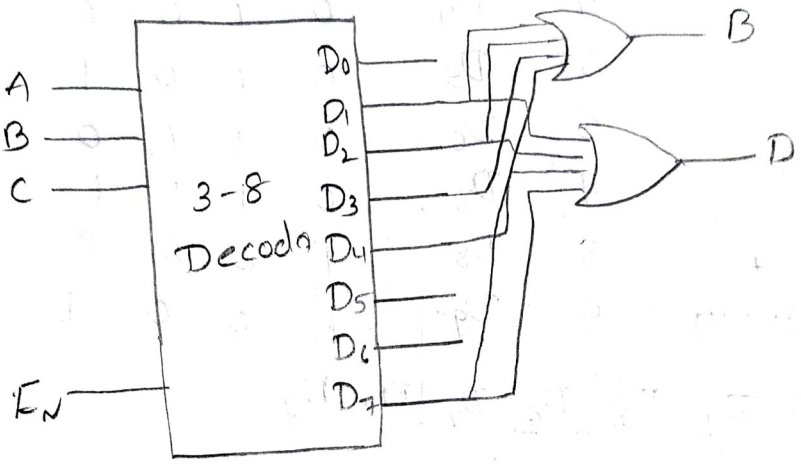
In SOP implementation in decoder we are using OR gate. POS implementation in decoder we are using NOR gate.

$F = \Sigma m (1, 2, 3, 7)$

$F = \Pi M (1, 3, 5, 7)$

for active low output :-
In SOP implementation in decoder we are using NAND gate
$F = \Sigma m(1, 2, 5, 7)$
In POS implementation in decoder we are using NAND gate $F = \pi M(1, 3, 5, 7)$

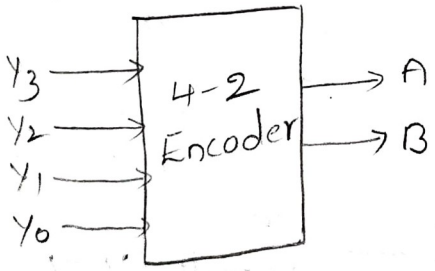
Eg :- Realize a full subtractor using a 3-8 decoder
$D(x, y, B_{in}) = \Sigma(1, 2, 4, 7)$
$B_{out}(x, y, B_{in}) = \Sigma(1, 2, 3, 7)$



Encoder :-

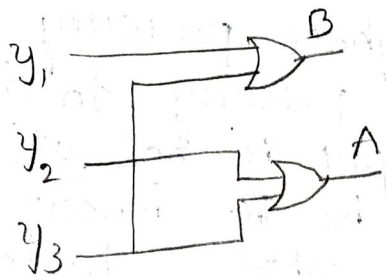
Encoder is a combinational circuit that performs the reverse operation of decoder. It has maximum of 2^n input lines and n output lines.

4-2 Encoder :-



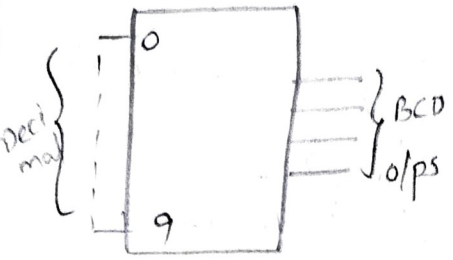
Y_3	Y_2	Y_1	Y_0	A	B
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$A = Y_2 + Y_3$, $B = Y_1 + Y_3$



Decimal to BCD encoder :-

This type of encoder has 10 inputs one for each decimal digit and 4 outputs corresponding to the BCD code. as shown in figure. This is a basic 10-H encoder.



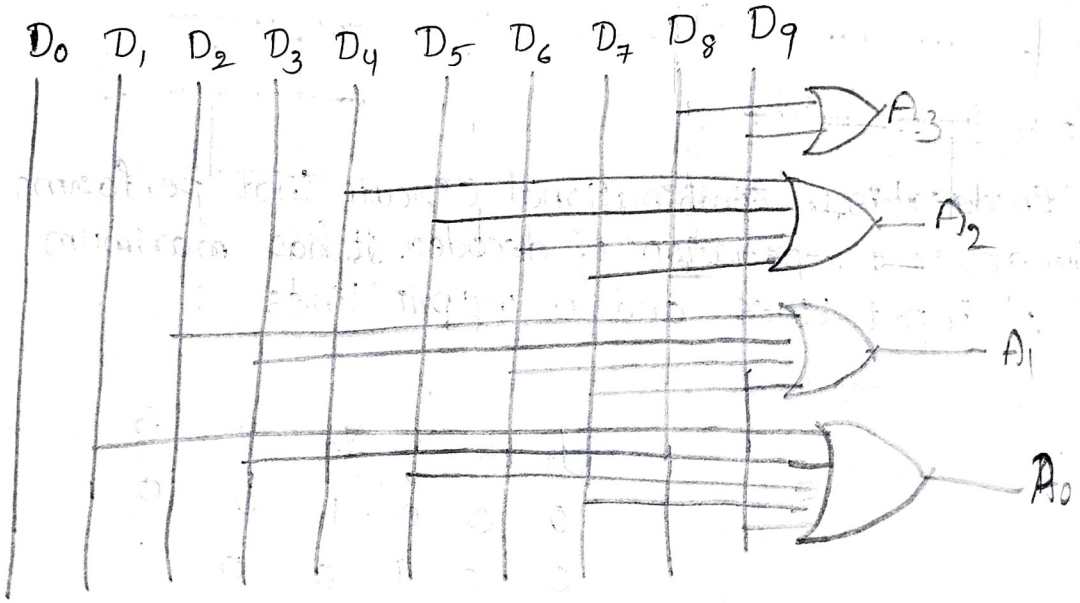
		A_3	A_2	A_1	A_0
0	D_0	0	0	0	0
1	D_1	0	0	0	1
2	D_2	0	0	1	0
3	D_3	0	0	1	1
4	D_4	0	1	0	0
5	D_5	0	1	0	1
6	D_6	0	1	1	0
7	D_7	0	1	1	1
8	D_8	1	0	0	0
9	D_9	1	0	0	1

$A_3 = D_8 + D_9$

$A_2 = D_4 + D_5 + D_6 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$

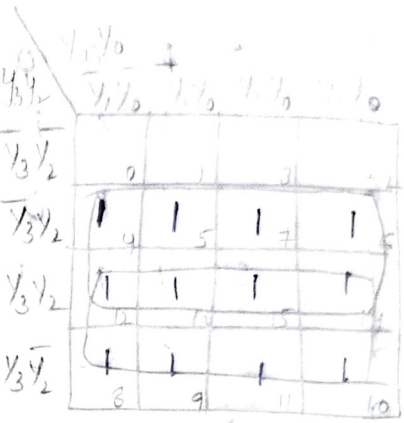


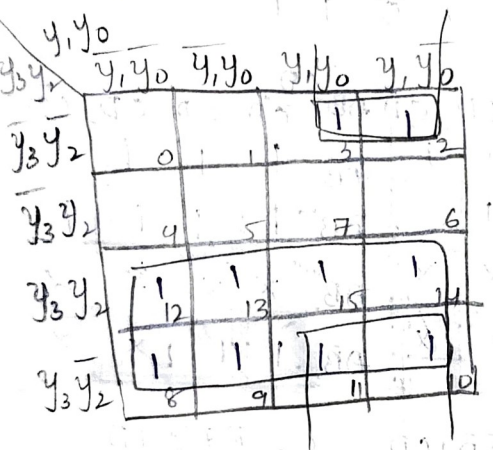
Priority Encoder

A 4×2 priority encoder has 4 inputs Y_3, Y_2, Y_1 and Y_0 and 2 outputs A_1 and A_0. Here the input Y_3 has the highest priority whereas the input Y_0 has the lowest priority. In these case if more than one input is one at the same time the output will be the binary code corresponding to the input which is having

higher priority. We consider one more output V, in order to know whether the code available outputs is valid or not. If atleast one input of the encoder is one then the code available at outputs is a valid one. In these case the output 'V' will be equal to 1. If all the inputs of encoder are zero then the code available at outputs is not a valid one. In this case the output 'V' will be equal to '0'

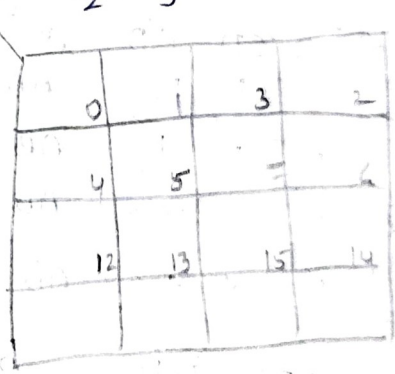
Y_3	Y_2	Y_1	Y_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1



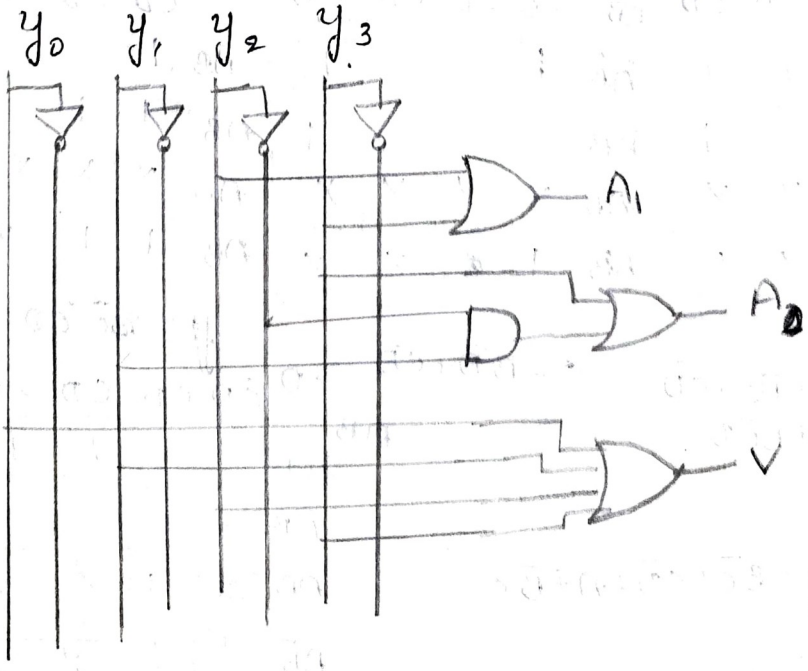


$A_0 = y_3 + \bar{y}_2 y_1$

$y_2 + y_3 = A_1$

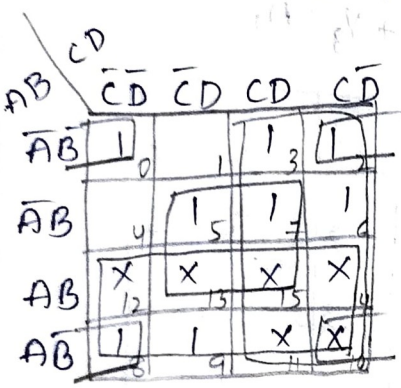


$V = y_0 + y_1 + y_2 + y_3$

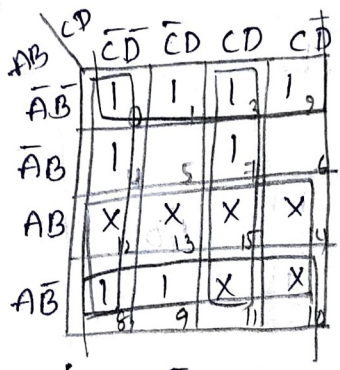


BCD to 7 segment display decoder:

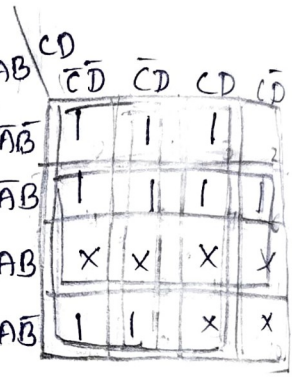
digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1



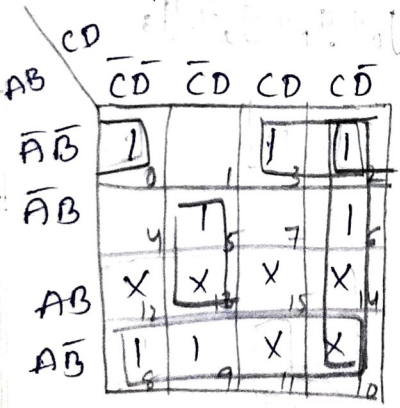
$a = A + C + BD + \bar{B}\bar{D}$



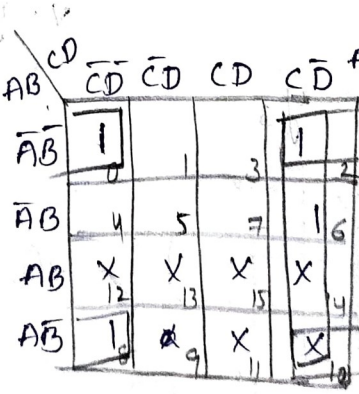
$b = A\bar{B} + \bar{C}\bar{D} + CD$



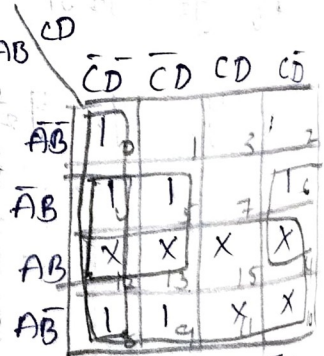
$C = AB + \bar{C} + D$



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

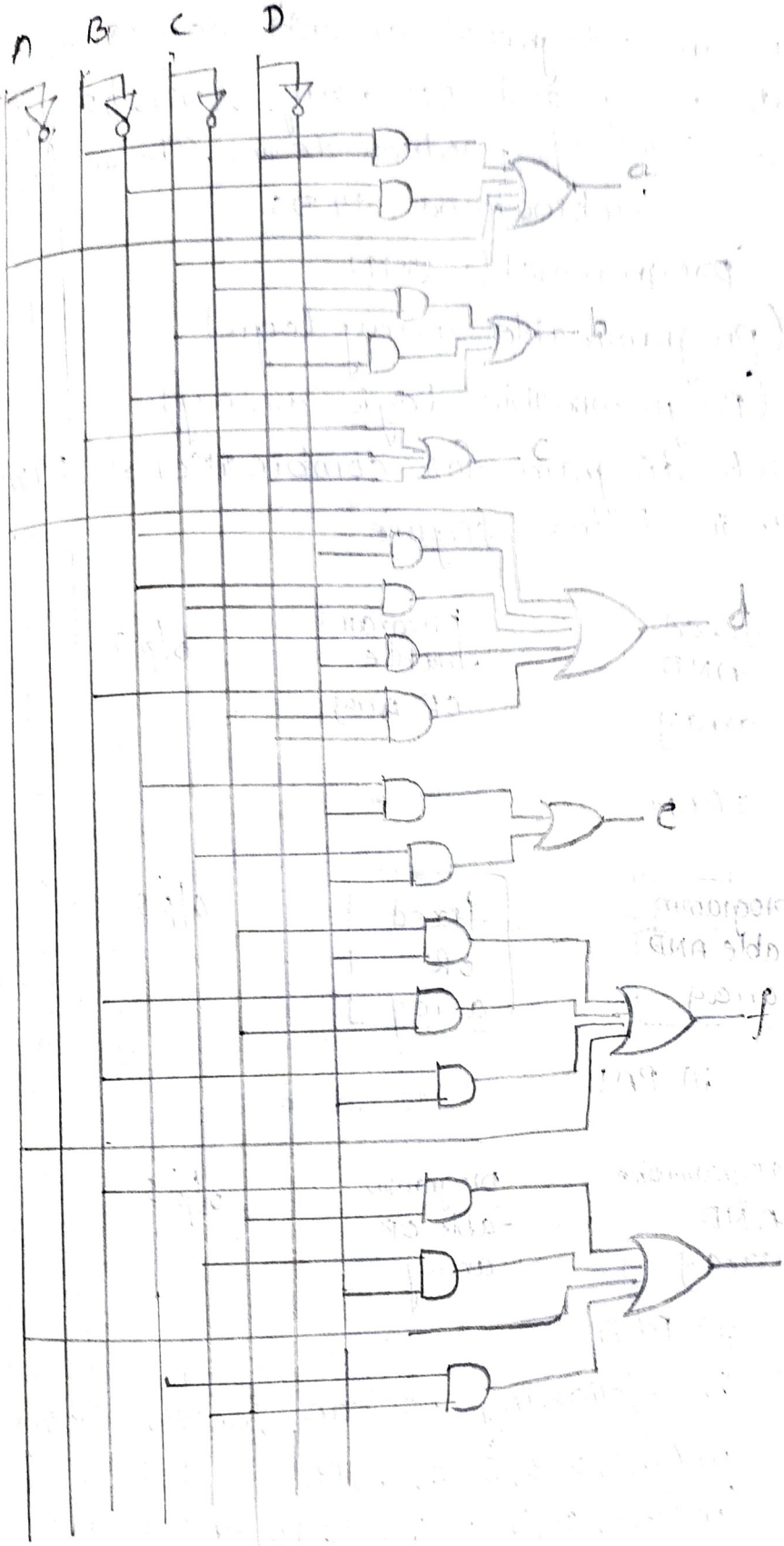


$e = \bar{B}\bar{D} + C\bar{D}$



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

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

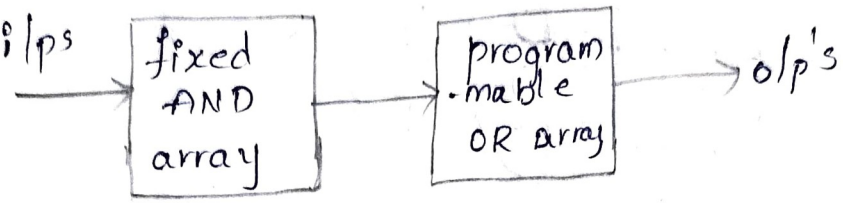

A
B
C
D
a
b
c
d
e
f

Combinational PLDs

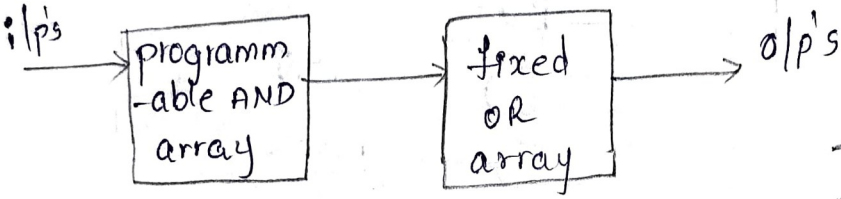
It is an integrated circuits constructed with AND array and OR array. AND-OR array gives the sum of product terms. There are 3 types of combinational PLDs

1) PROM :- (programmable ROM)
2) PAL :- (programmable array logic)
3) PLA :- (programmable logic array)

The block diagram for combinational PLDs as shown in below figure.

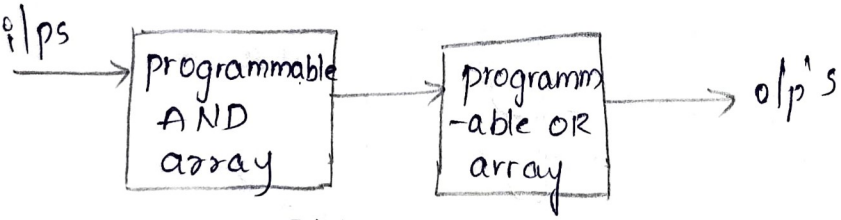


i) PROM



ii) PAL

fixed - •
programm able - x

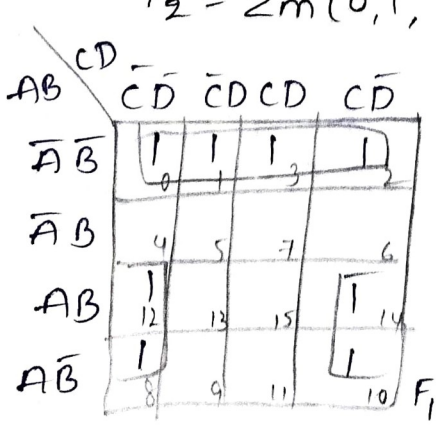


iii) PLA

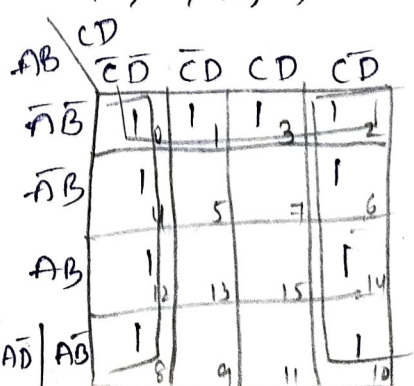
1. Implement the following boolean function using PAL

$F_1 = \Sigma m(0,1,2,3,8,10,12,14)$

$F_2 = \Sigma m(0,1,2,3,4,6,8,10,12,14)$



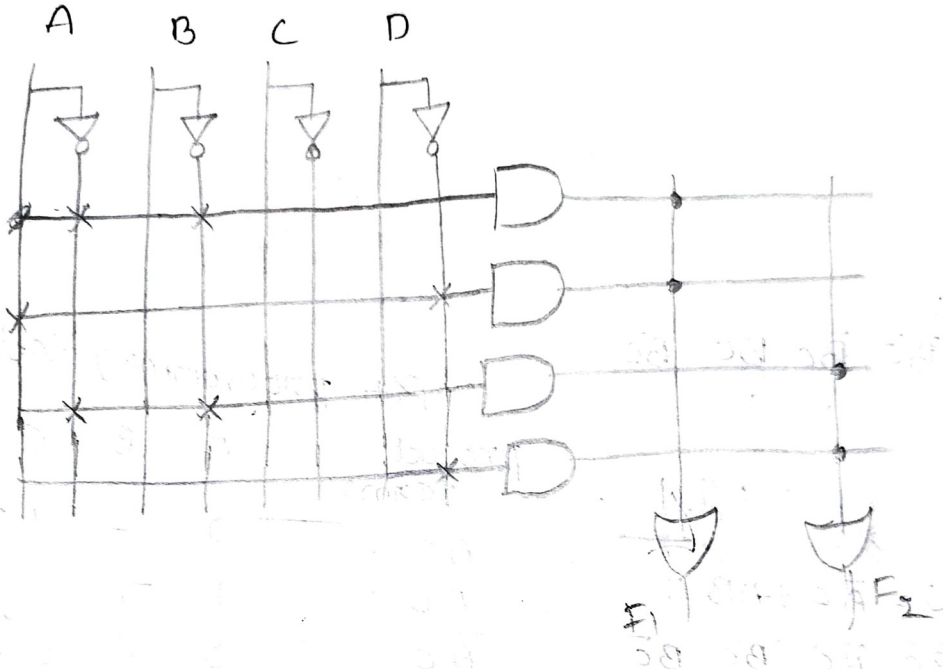
$F_1 = \bar{A}\bar{B} + A\bar{D}$



$f_2 = \bar{A}\bar{B} + \bar{D}$

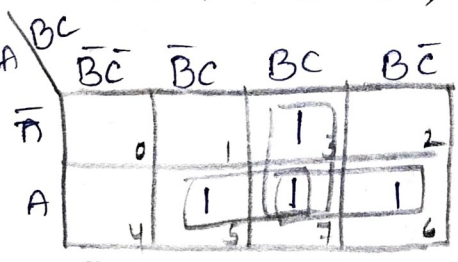
PAL programming table

product term	AND inputs A	B	C	D	
$\bar{A}\bar{B}$	0	0	-	-	F_1
$A\bar{D}$	1	-	-	0	F_1
$\bar{A}\bar{B}$	0	0	-	-	F_2
$\bar{D}$	-	-	-	0	F_2

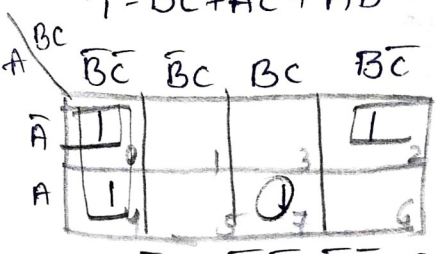


1) Design and implement the following boolean function using PLA $F_1(A,B,C) = \Sigma m(3,5,6,7)$
$f_2(A,B,C) = \Sigma m(0,2,4,7)$

2) Realize the boolean function $w(a,b,c) = \Sigma m(1,3,6,7)$
$x(a,b,c) = \Sigma m(0,2,4,5)$ using PROM



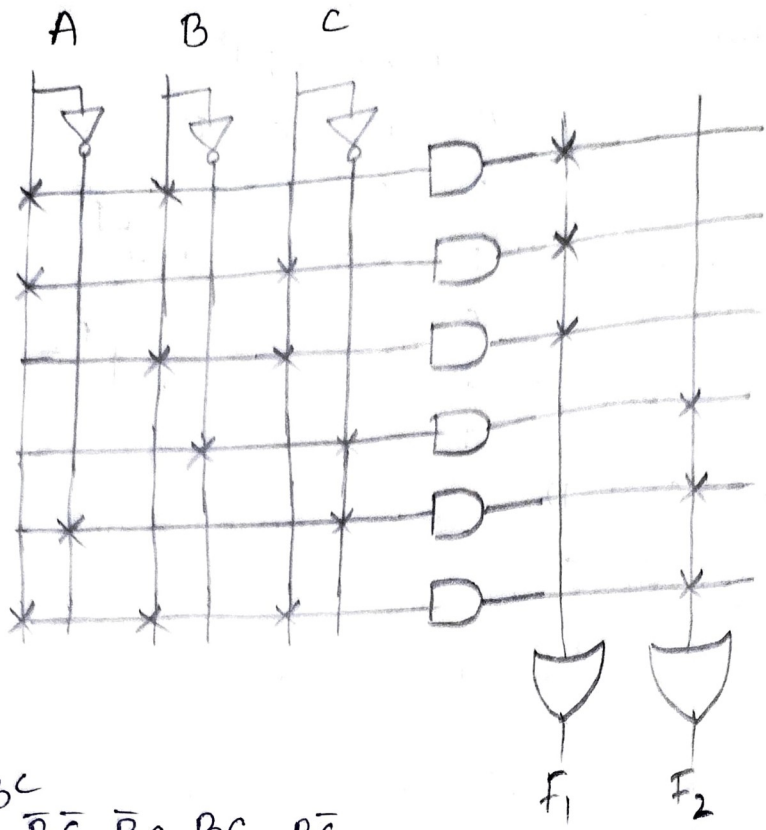
$F_1 = BC + AC + AB$



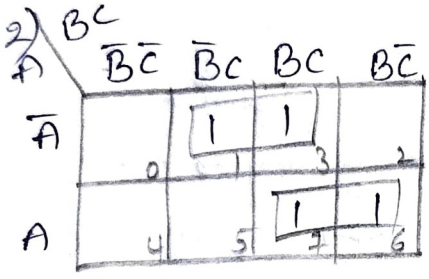
$F_2 = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$

PLA programming table

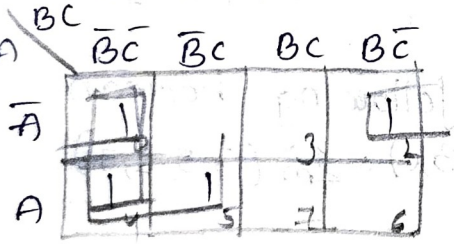
product term	AND inputs A	B	C	
BC	-	1	1	F_1
AC	1	-	1	F_1
AB	1	1	-	F_1
$\bar{B}\bar{C}$	-	0	0	F_2
$\bar{A}\bar{C}$	0	-	0	F_2
ABC	1	1	1	F_2



2)



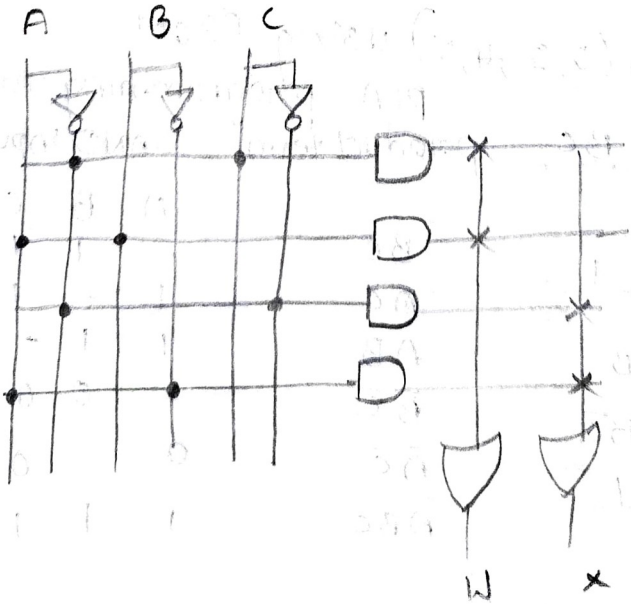
$W = \bar{A}C + A\bar{B}$



$X = \bar{B}\bar{C} + A\bar{B} + \bar{A}\bar{C}$

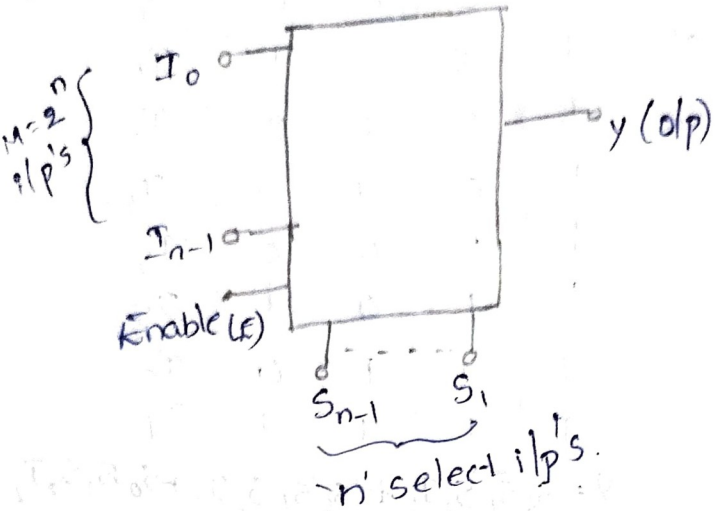
PLA programming table

product terms	A	B	C	
$\bar{A}C$	0	-	1	W
$A\bar{B}$	1	1	-	W
$\bar{A}\bar{C}$	0	-	0	X
$A\bar{B}$	1	0	-	X

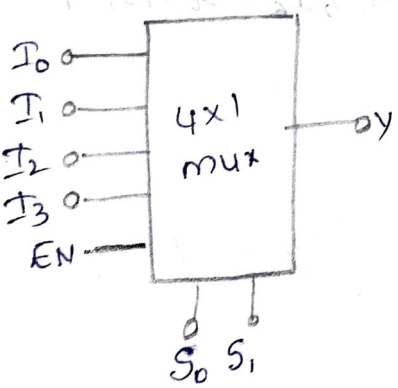


Multiplexers:- (or data selector)

It is a combinational circuit having multiple inputs and a single output. The transfer of an input to the output is controlled by select lines. It is also called as data selector.

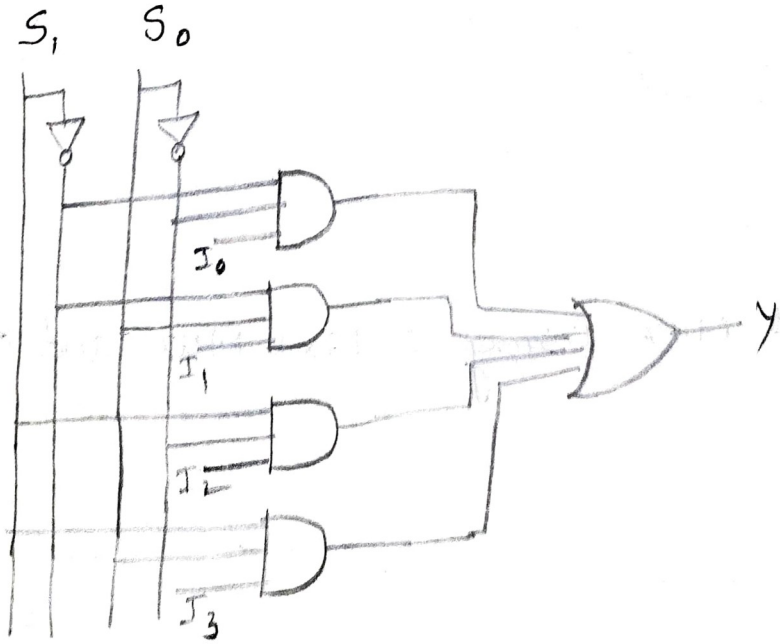


4x1 MUX: (2) → 2 selectors
$2^2 = 4$

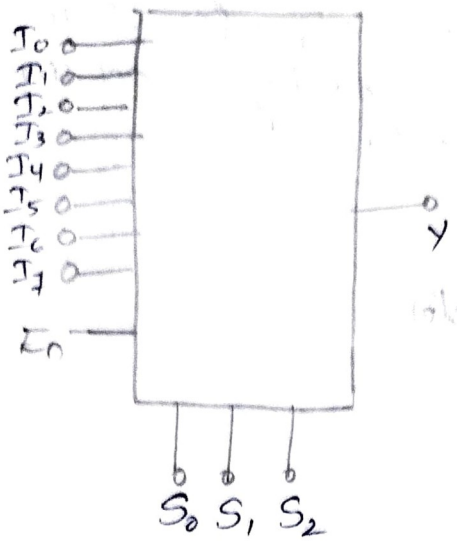


EN	Select lines		output
	S_1	S_0	y
0	x	x	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

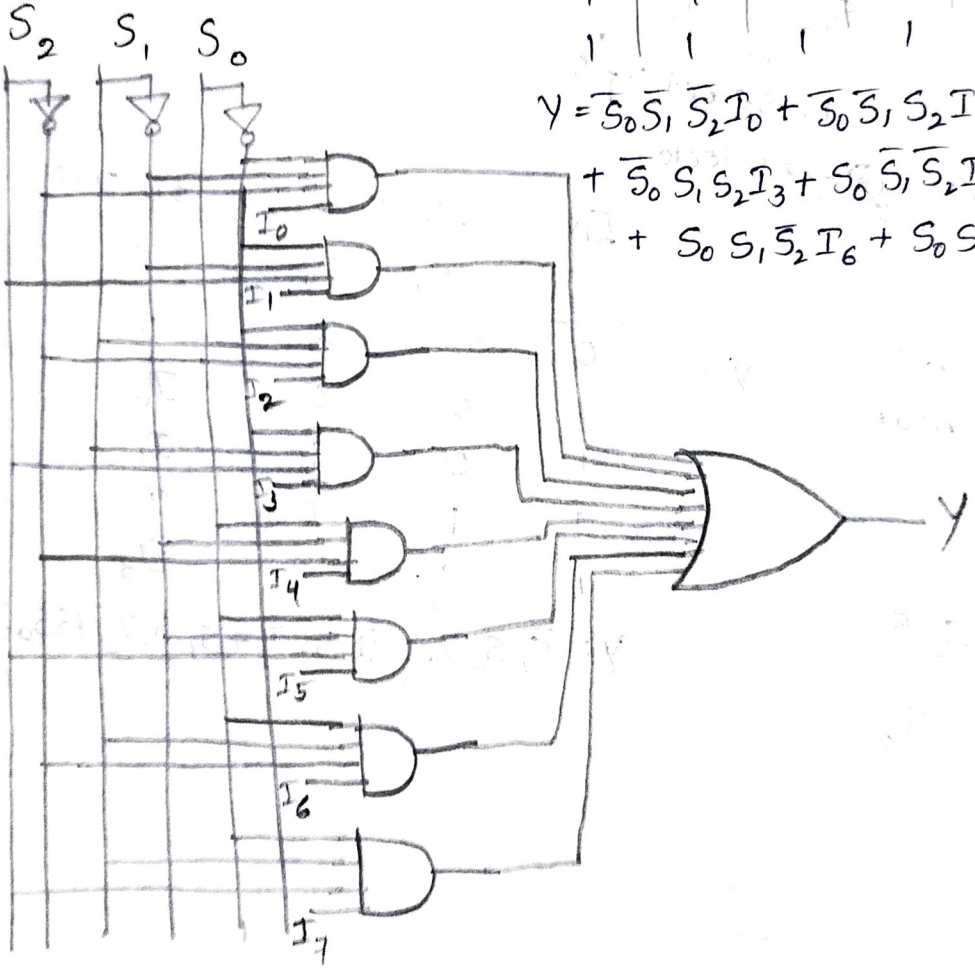
$$y = \overline{S_1}\,\overline{S_0}\,I_0 + \overline{S_1}\,S_0\,I_1 + S_1\,\overline{S_0}\,I_2 + S_1 S_0\,I_3$$



8 × 1 Mux $2^3 = 8 \Rightarrow 3$ selectors

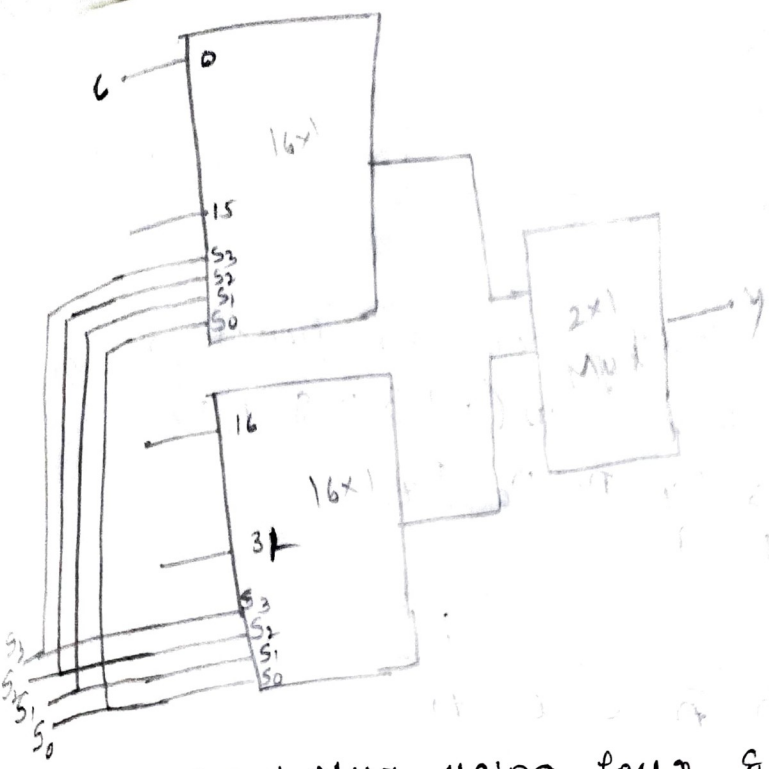


EN	select lines			output
	S_0	S_1	S_2	Y
0	x	x	x	0
1	0	0	0	I_0
1	0	0	1	I_1
1	0	1	0	I_2
1	0	1	1	I_3
1	1	0	0	I_4
1	1	0	1	I_5
1	1	1	0	I_6
1	1	1	1	I_7

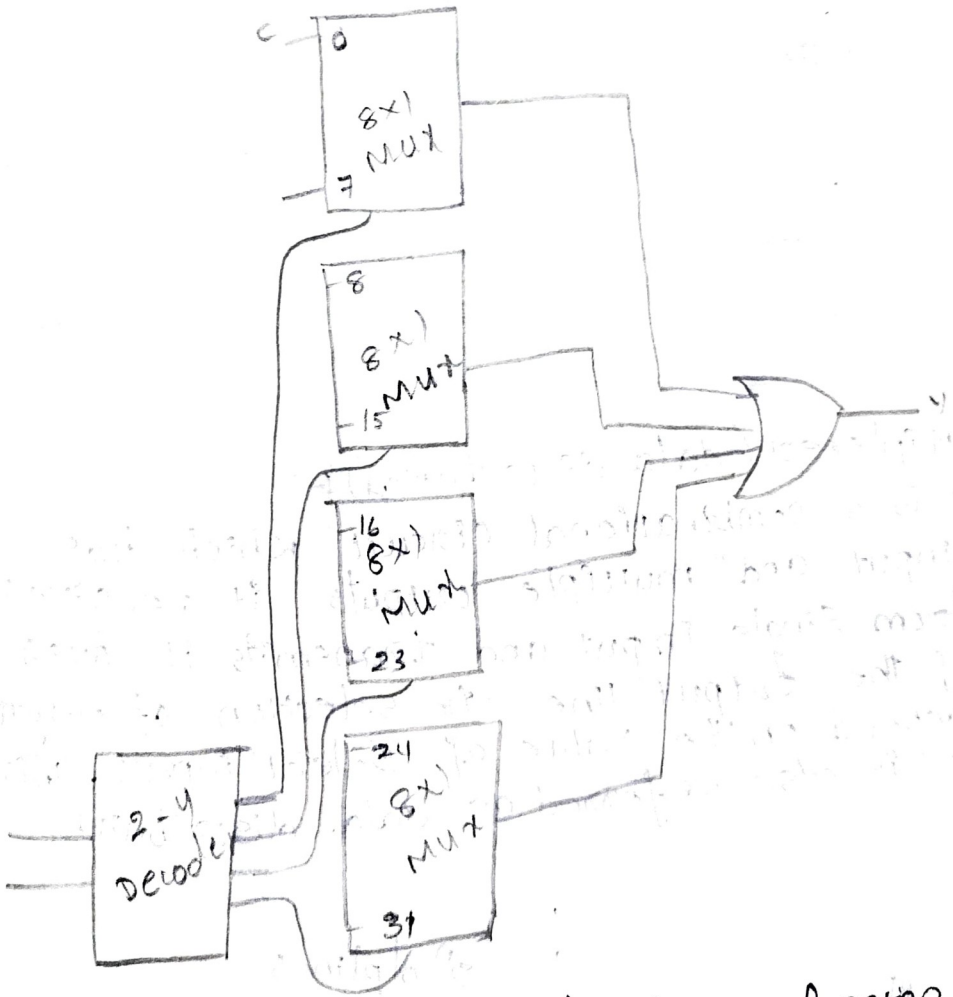


$$Y = \bar{S}_0 \bar{S}_1 \bar{S}_2 I_0 + \bar{S}_0 \bar{S}_1 S_2 I_1 + \bar{S}_0 S_1 \bar{S}_2 I_2 + \bar{S}_0 S_1 S_2 I_3 + S_0 \bar{S}_1 \bar{S}_2 I_4 + S_0 \bar{S}_1 S_2 I_5 + S_0 S_1 \bar{S}_2 I_6 + S_0 S_1 S_2 I_7$$

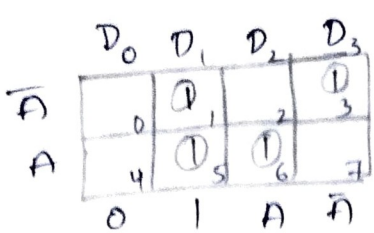
Design 32×1 Mux using 2 16×1 Mux and one 2×1 mux

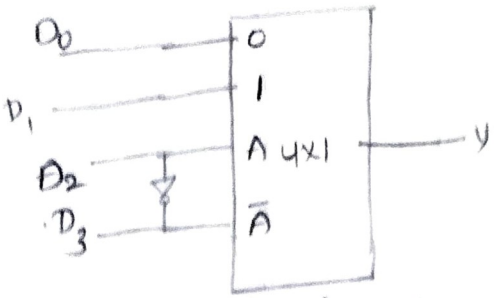


Design 32x1 Mux using four 8x1 Mux and one 2-4 Decoder

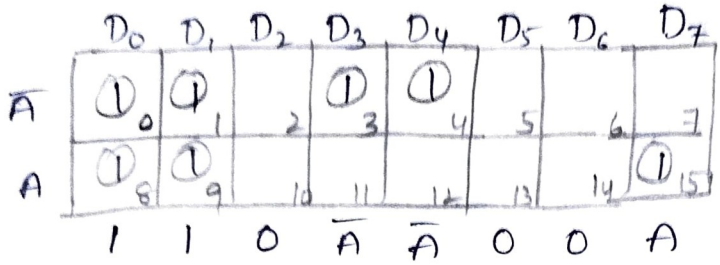


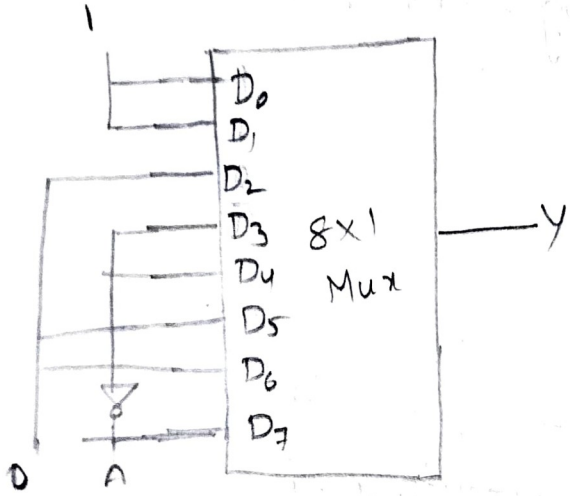
Implement the following boolean function using 4x1 mux $F(A,B,C) = \Sigma m(1,3,5,6)$





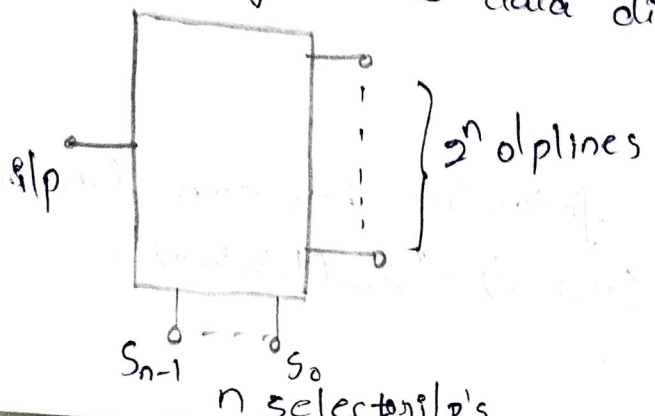
Implement the following boolean function using 8x1 Mux $F(P,Q,R,S) = \Sigma m(0,1,3,4,8,9,15)$





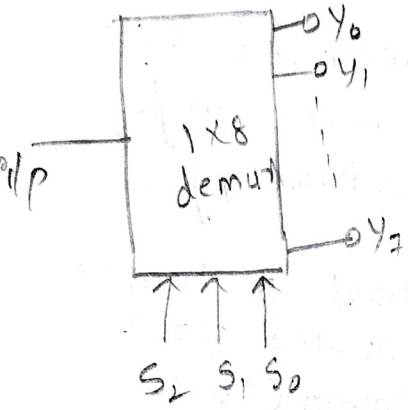
demultiplexer (data distributor) :-

It is a combinational circuit which has single input and multiple outputs it receives data from single input and transmits it over one of the output line the selection of output line depend on the value of select inputs. This circuit is also referred as data distributor.



* Demultiplexers are used in data transmission
* They are used in the implementation of Boolean functions.
* They are used in designing of combinational logic circuits.
* These are also preferred in enable signal generation.

1x8 demux:-



S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	D
0	0	1	0	0	0	0	0	0	D	0
0	1	0	0	0	0	0	0	D	0	0
0	1	1	0	0	0	0	D	0	0	0
1	0	0	0	0	0	D	0	0	0	0
1	0	1	0	0	D	0	0	0	0	0
1	1	0	0	D	0	0	0	0	0	0
1	1	1	D	0	0	0	0	0	0	0

$y_0 = \bar{S_2}\bar{S_1}\bar{S_0}$
$y_1 = \bar{S_2}\bar{S_1}S_0$
$y_2 = \bar{S_2}S_1\bar{S_0}$
$y_3 = \bar{S_2}S_1S_0$
$y_4 = S_2\bar{S_1}\bar{S_0}$
$y_5 = S_2\bar{S_1}S_0$
$y_6 = S_2S_1\bar{S_0}$
$y_7 = S_2S_1S_0$