UNIT-III

COMBINATIONAL LOGIC CIRCUITS DESIGN

Combinational circuit:-

Logic circuits for digital systems may be combinational or sequential

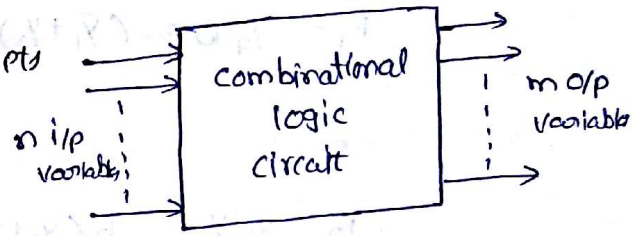
When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called "combinational logic circuit".

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

→ It performs an operation that can be specified logically by a set of Boolean functions.

→ It consists of input variables, logic gates and output variables. The logic gate accept signals from the inputs and generate signals to the outputs

Fig: shows the combination circuit accepts n-i/p variables and generates o/p variables depending on the logical combination of gates.



→ If 'n' i/p binary variables come from an external source and the 'm' o/p variables go to an external destination.

→ For 'n' i/p variables, there are 2^n possible o/p values for each i/p combinations. for each possible i/p combination, there is one possible o/p value. Thus a combinational circuit can be specified with a truth table that lists the o/p values for each combination of i/p variables.

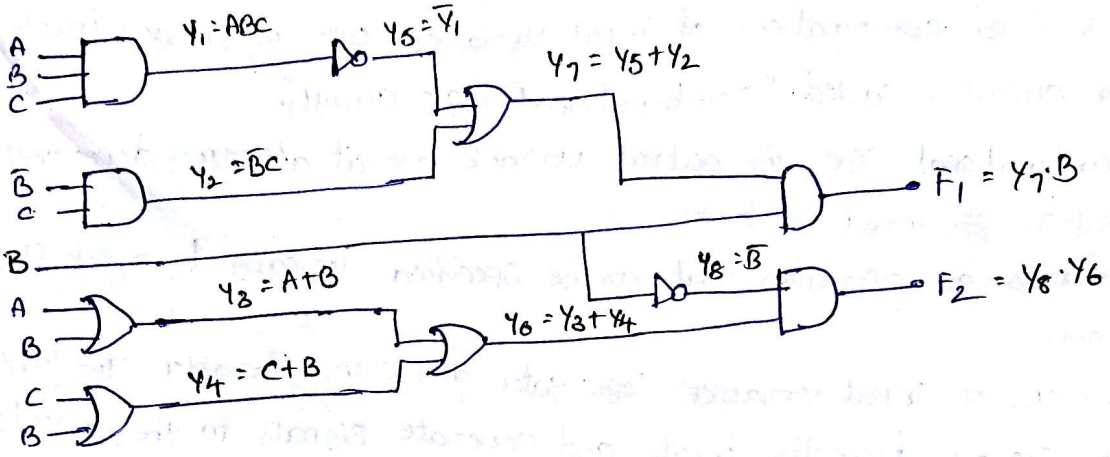
Analysis Procedure:-

In this from the given circuit diagram we have to obtain a set of Boolean functions for outputs of circuit, a truth table or a possible explanation of the circuit operation.

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 all gate o/p's that are a function of i/p variables with arbitrary symbol's, and determine the Boolean functions for each gate output
3. Label the gates that are a function of i/p variables and previously labeled gates and determine the Boolean function for them.

4. Repeat the step 3 until the Boolean function for o/p's of the circuits are obtained
5. Finally, substituting previously defined Boolean functions obtain the output Boolean functions in terms of i/p variables.

Eg: obtain the Boolean functions for outputs of the given circuit.



$$F_1 = Y_7 B = (\bar{Y}_1 + Y_2) B = (\overline{ABC} + \bar{B}C)B$$

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

$$F_2 = Y_8 Y_6 = \bar{B}(Y_3 + Y_4) = \bar{B}[(A+B) + (C+B)]$$

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

Adders:—

→ Digital computers perform various arithmetic operation. The most basic arithmetic operation is the addition of two binary digits.

The simple addition consists of '4' possible combinations:

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

The first '3' operations produce 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 a "Carry". and lower significant bit is called "Sum". The logic circuit which performs this operation is called a "half-adder". the circuit which performs addition of three bits (two significant bits and a previous carry) is called "full-adder"

G.NAVEEN
EEE DEPT

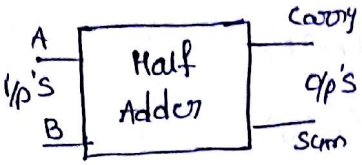
Half Adder:-

The operation needs two binary i/p's & two binary o/p's (sum & carry).

Let A, B are i/p's & carry & sum are o/p's.

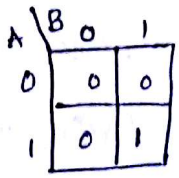
Inputs		outputs	
A	B	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table.



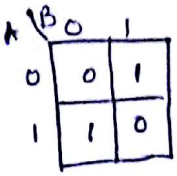
Block Diagram.

K-map for carry:-



carry = AB

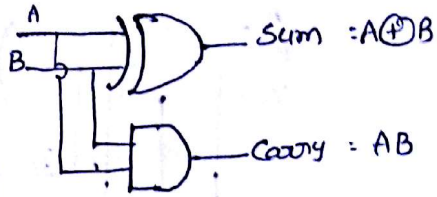
Kmap for sum:



sum: $A\bar{B}+\bar{A}B$

$= A \oplus B$

Logic diagram:-



Limitations:-

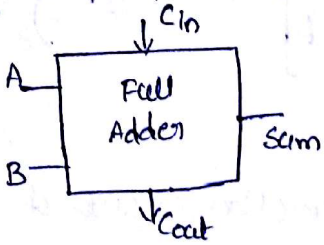
Addition of '3' bits is not possible in half-adder. Hence half-adders are not used in practice.

Full-adder:-

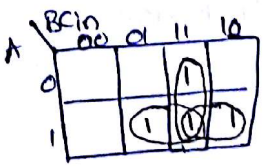
Full adder is a combinational circuit that performs the sum of '3' bits. It consists of 3 i/p's and 2 o/p's. Two of the i/p variables denoted by A & B, represents two significant bits to be added. The 3rd i/p Cin represents the carry from the previous lower significant position (LSB). The two o/p's are: carry and sum.

→ The sum is equal to '1' when only one i/p is 'equal to '1' or when all three i/p's are equal to '1'. The carry is equal to '1' if two or three i/p's are equal to '1'.

Inputs			outputs	
A	B	Cin	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

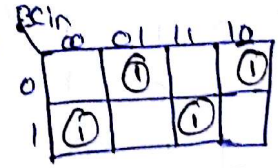


Kmap for carry:-



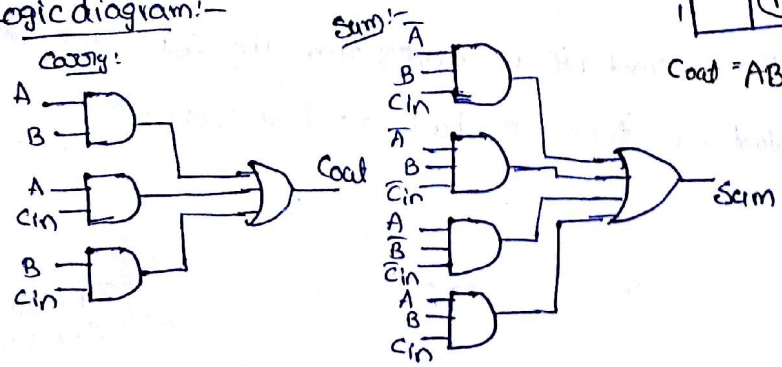
Cout = $AB+AC_{in}+BC_{in}$

K-map for sum



Sum: $\bar{A}BC_{in}+\bar{A}B\bar{C}_{in}+A\bar{B}\bar{C}_{in}+ABC_{in}$

Logic diagram:-



The Boolean function for Sum can be further simplified as.

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

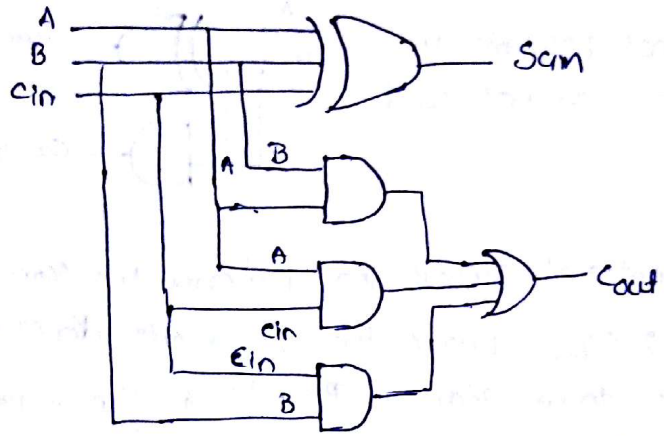
$$= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(\bar{A}B + A\bar{B})$$

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

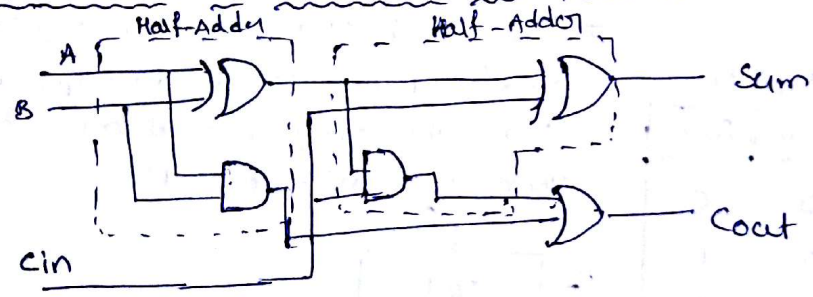
$$= C_{in}(\overline{A \oplus B}) + \bar{C}_{in}(A \oplus B)$$

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

Implementation of Full-Adder:-



Implementation of Full-Adder with two Half-Adder and OR & Ex-OR gate:-



B Subtractors:-

The subtraction consists of four possible elementary operations.

0 - 0 = 0

0 - 1 = 1 with '1' borrow

1 - 0 = 1

1 - 1 = 0

In 2nd operation the minuend bit is smaller than the subtrahend bit, hence '1' is borrowed. Just as those are half and full subtractors.

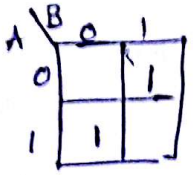
G. NAVEEN
EEE DEPT

Half-Subtractor:-

It performs subtraction b/w two bits and produces their difference. It also have an output to specify if a 1 has been borrowed. Let us designate minuend bit as A and the subtrahend bit as B. The result of operation A-B for all possible values of A & B.

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

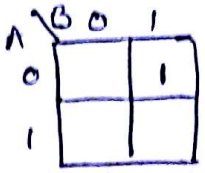
K-map for Difference:-



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

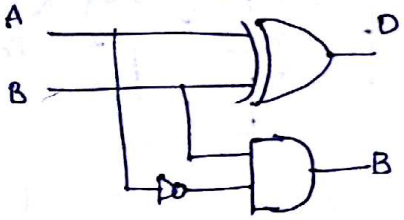
$= A \oplus B$

K-map for Borrow:-



$B = \bar{A}B$

Logic diagram:-



Limitations:-

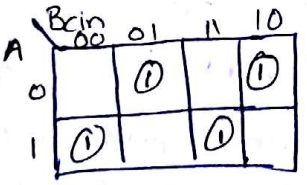
In multidigit subtraction, we have to perform subtraction b/w two bits along with the borrow of the previous digit subtraction. effectively such subtraction requires subtraction of '3' bits. This is not possible with half-subtractor.

Full-Subtractor:-

It performs a subtraction b/w two bits, taking into account borrow of the lower significant stage. This circuit has 3 i/p's & 2 o/p's. The 3 i/p's are A, B & Cin, denotes minuend, subtrahend & previous borrow resp. The two o/p's D & Bout represents the difference & o/p borrows. resp.

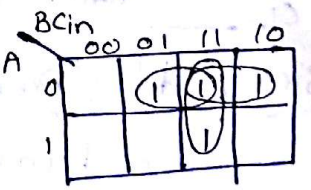
Inputs			Outputs	
A	B	Cin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map simplification for D:-

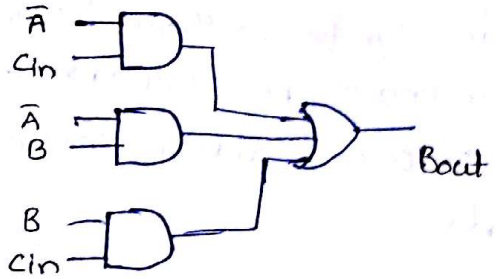


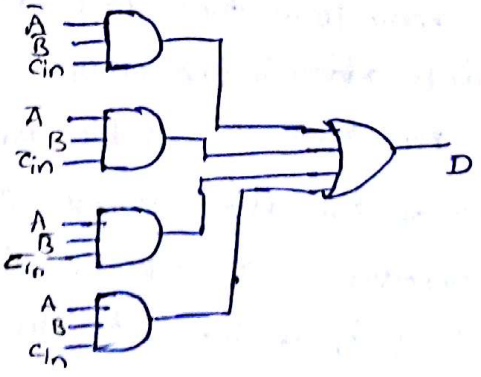
$D = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

For Bout:-



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





The Boolean function for D (difference) can be simplified as

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

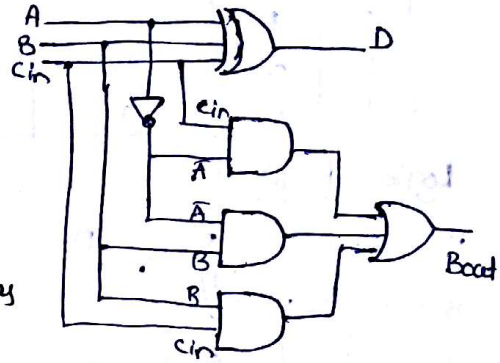
$$= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(\bar{A}B + A\bar{B})$$

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

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

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

With this simplified function, CKT for full-subtractor can be implemented as



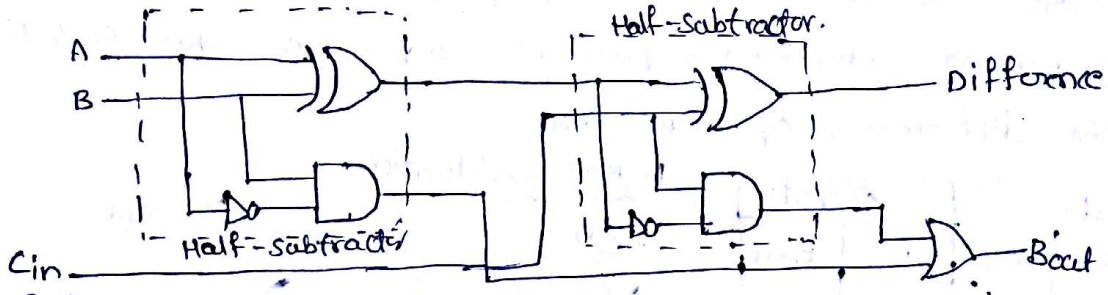
A full subtractor can also be implemented with two half-subtractor's and one OR gate as shown in fig. The difference o/p from the second half-subtractor is the exclusive-OR of Cin and the o/p of the first half-subtractor, which is same as difference o/p of full-subtractor.

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

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

by solving we get $= \bar{A}B + \bar{A}C_{in} + BC_{in}$

Implementation of a full-subtractor with two half-subtractor's & an OR gate:-

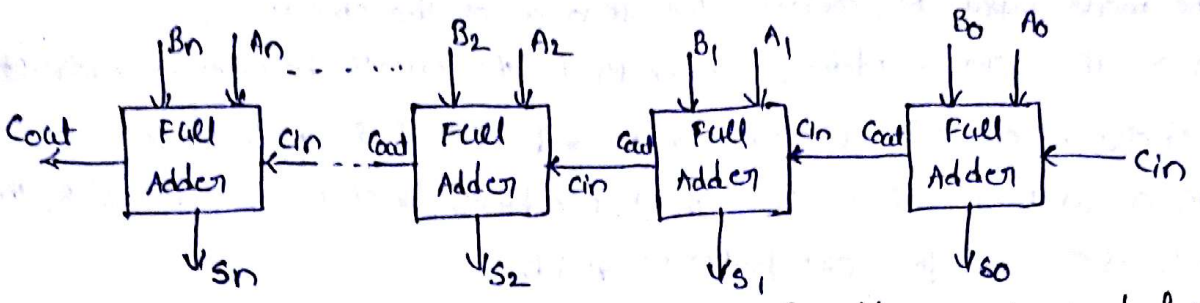


Binary Adder/Parallel Adder:- (or) 4-bit parallel Adder:-

A binary parallel adder is a digital circuit that adds two binary numbers in parallel form and produces the arithmetic sum of those numbers in parallel form. It consists of full adders connected in a chain, with the output carry from each full-adder connected to the input carry of the next full adder in the chain.

Fig. shows the n-bit parallel adder can be constructed using number of full adder circuits connected in parallel. here full adder circuits are connected in cascade. ie the carry o/p of each adder is connected to the carry i/p of the next higher-order adder.

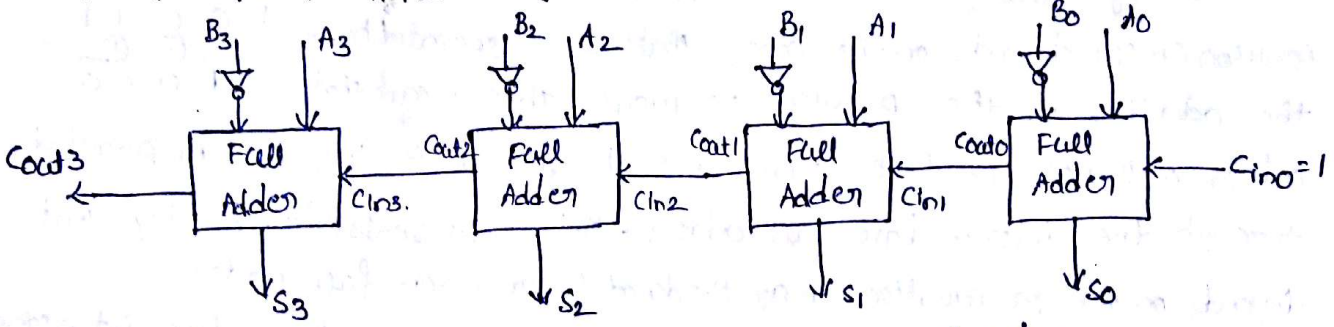
G.NAVEEN
EEE DEPT



It should be noted that either a half-adder can be used for the least significant position or the carry input of a full-adder is made '0' because there is no carry into the least significant bit position.

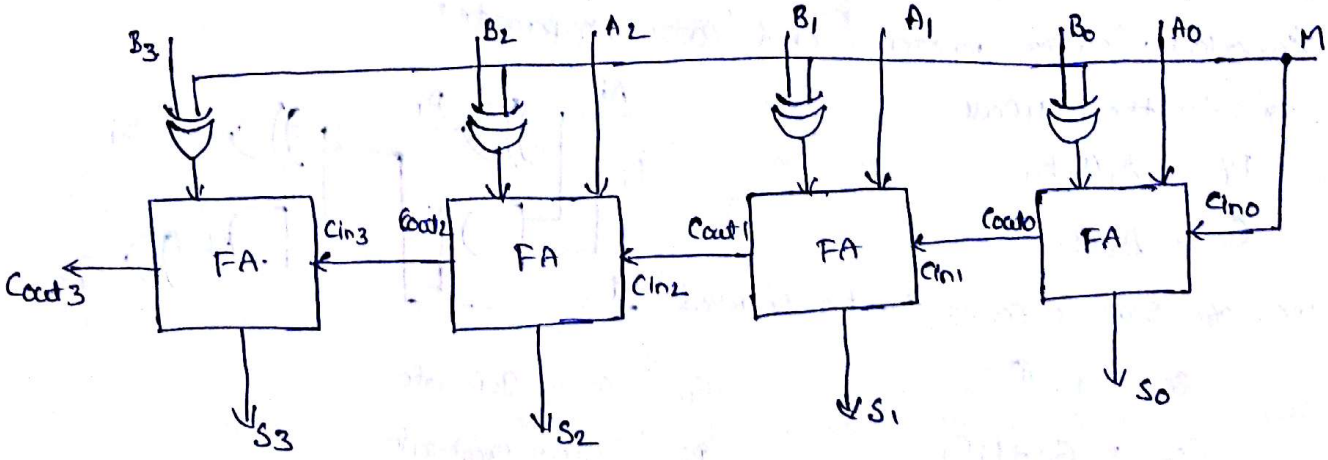
Binary Subtractor / Parallel Subtractor / 4-Bit Parallel Subtractor:-

The subtraction of binary numbers can be done by means of complements. The subtraction 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 taking the 1's complement and adding one 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 fig.



Parallel Adder/Subtractor or Adder-Subtractor Circuit:-

The addition and subtraction operations can be combined into one circuit with one common binary adder. this is done by including an exclusive-OR gate with each full adder as shown in fig.



4-bit adder-subtractor.

The mode input 'M' control's the operation of the circuit. When M=0, the CKT is adder, when M=1, the circuit becomes a subtractor Each exclusive-OR gate receives i/p M and one of the i/p's of B. when M=0, we have $B \oplus 0 = B$, the full adder's receive the values of B, the i/p carry is 0, and the CKT performs A+B.

when M=1, we have $B \oplus 1 = \bar{B}$ and $C_{in} = 1$. the B i/p's are all complemented and a 1 is added through the i/p carry. the circuit performs the operation A plus the 2's complement of B, i.e A-B.

Look-a-Head Adder Circuit for Carry Look Ahead Adder:—

In full Adder the carry output of each full-adder stage is connected to the carry input of the next higher-order stage. ∴ the sum and carry o/p's of any stage cannot be produced until the input carry occurs. this leads to a time delay in the addition process. this delay is known as "carry propagation delay".

Eg:
$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

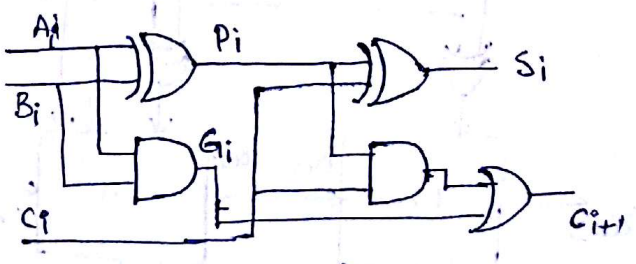
In Eg: that the sum bit generated in the last position (MSB) depends on the carry that was generated by the addition in the previous positions. This means that, adder will not produce correct result until LSB carry has propagated through the intermediate full-adders. this represents a "time delay" that depends on the propagation delay produced in on each full-adder.

one method of speeding up this process by eliminating inter stage carry delay is called "Look-A-head-carry addition". This method utilizes logic gates to look at the lower-order bits of the augend and addend to see if a higher-order carry is to be generated. It uses two functions: "carry generate" and "carry propagate"

consider the circuit.

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$



The o/p sum & carry can be expressed

$S_i = P_i \oplus C_i$

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

G_i = carry generate

P_i = carry Propagate.

G.NAVEEN
EEE DEPT

Now the Boolean function for the carry o/p of each stage can be written as.

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 =$$

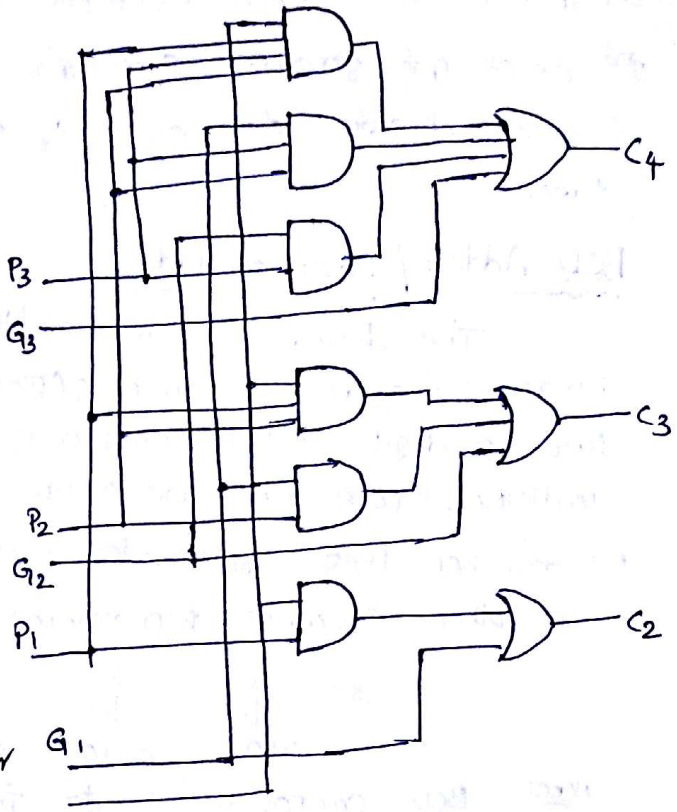
$$= G_2 + P_2 (G_1 + P_1 C_1)$$

$$= G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3$$

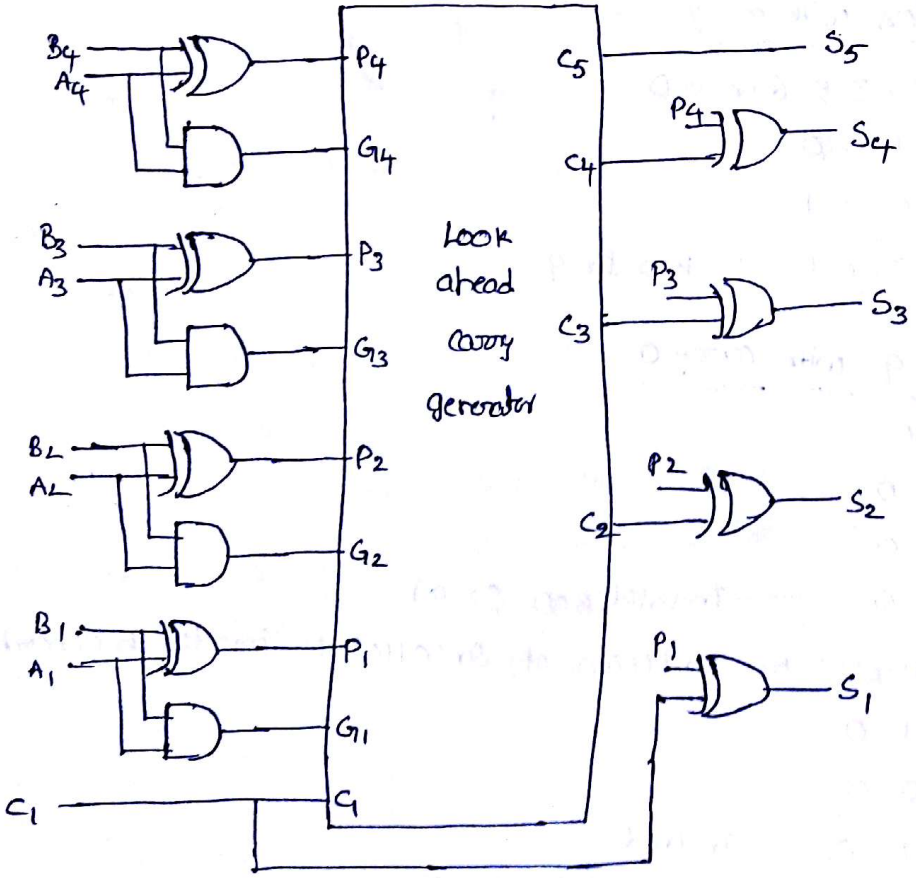
$$= G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 C_1)$$

$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$



from the above expression it can be seen that C_4 does not have to wait for C_3 & C_2 to propagate; in fact C_4 is propagated at the same time as C_2 & C_3

Using a look-ahead carry generator we can easily construct a 4-bit parallel adder with a look-ahead carry scheme as shown in fig



from fig each sum output requires two ex clusive-OR gates. the output of first E-OR gate generates P_i and the AND gate generates G_i.

The carries are generated using look-ahead carry generator and applied as i/p's to the second EX-OR gate. other i/p's to EX-OR gate is P_i. Thus 2nd EX-OR gate generates sum o/p's. Each o/p is generated after a delay of two levels of gate. Thus o/p's S_2 through S_4 have equal propagation delay times.

BCD Adder / Decimal Adder:-

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

When we write BCD number 526, is

5	2	6
↓	↓	↓
0101	0010	0110

Note: BCD cannot be greater than 9.

BCD addition of two numbers can be done by considering 3 cases.

Sum Equals 9 (or) less with carry 0:-

Addition of 3 & 6 in BCD

6	0 1 1 0
+3	0 0 1 1
9	1 0 0 1 ← BCD for 9

Sum greater than 9 with carry 0:-

6 + 8 in BCD

6	0 1 1 0
+ 8	1 0 0 0
14	1 1 1 0 ← Invalid BCD (> 9)

So this can be corrected by addition of 6 (0110) to invalid BCD number

6	0 1 1 0
8	1 0 0 0
14	1 1 1 0 ← Invalid
	+ 0 1 1 0 ← Add 6 for correction
0001	0 1 0 0
1	4 ← BCD for 14

G. NAVEEN
EEE DEPT

Sum equals 9 or less with carry 1:-

addition of 8 & 9 in BCD

```
  8          1 0 0 0
 +9          1 0 0 1
 --          -------
 17     0 0 0 1 0 0 0 1   ← Incorrect BCD
```

To correct Incorrect BCD add '6' to BCD.

```
  8          1 0 0 0
 +9          1 0 0 1
 --          -------
 17     0 0 0 1 0 0 0 1   ← Incorrect
      + 0 0 0 0 0 1 1 0   ← Add 6 for correction
        -----------------
        0 0 0 1 0 1 1 1   ← BCD for 17.
```

BCD Addition Procedure:-

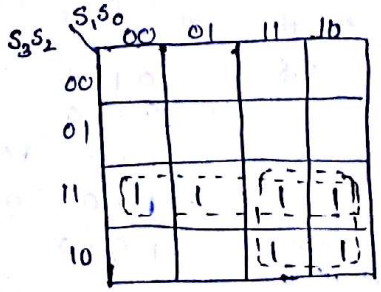
1. Add two BCD numbers using ordinary binary addition.
2. If 4-bit sum is equal to or less than 9, no correction is needed. the sum is in proper BCD form
3. If 4-bit sum is greater than 9 or if a carry is generated from the 4-bit sum, the sum is invalid
4. To correct the invalid sum, add 0110_2 to the 4-bit sum. If carry results from this addition, add it to the next higher-order BCD digit.

Thus to implement BCD adder we require 4 bit binary adder for initial addition.

- Logic circuit to detect sum greater than 9 and
- One more 4-bit adder to add 0110 in the sum if sum is greater than 9 or carry is '1'.

The circuit for a BCD adder must include the logic needed to detect when even the sum is greater than 9 so that the correction can be added in. These cases where the sum is greater than 1001 are listed in Table.

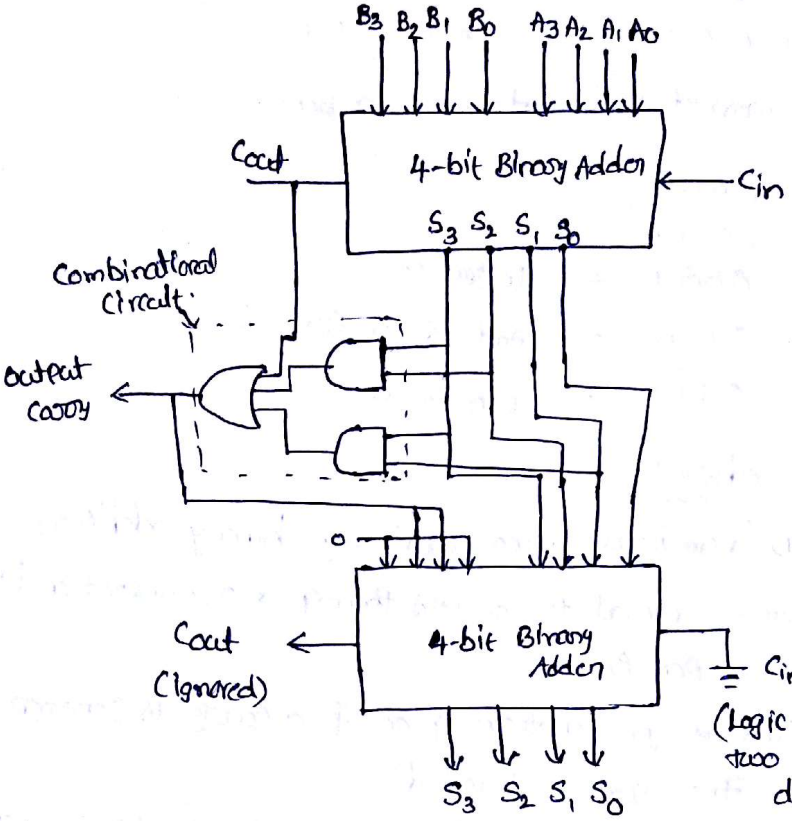
S_3	S_2	S_1	S_0	Y carried
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



$Y = S_3S_2 + S_3S_1$

Y=1 indicates sum is greater than 9. We can put one more term, C_{out} in the above expression to check whether carry is one. If any one condition is satisfied we add 6 (0110) in the sum.

Block diagram of BCD Adder:—



(Logic dia of a BCD adder using two 4-bit adders & correction-detector circuit).

The two BCD numbers, together with i/p carry, are first added in the top-4-bit binary adder to produce a binary sum. When the o/p carry is equal to zero (sum ≤ 9 & C_{out} = 0) nothing is added to the binary sum.

When it is equal to one (sum > 9, or C_{out} = 1), binary 0110 is added to the binary sum through the bottom 4-bit binary adder. The output carry generated from the bottom binary adder can be ignored. Since it supplies information already available at the output carry terminal.

Excess-3 (XS-3) Adder:— (or) Excess-3-Adder Circuit:—

To perform Excess-3 additions we have to

1. Add two XS-3 code's
2. If carry = 1 add 0011 (3) to the sum of those two code groups
   If carry = 0 subtract 0011 (3), i.e add (1101) 13 to the sum of those two code group's

Eg: Add 9 & 5

```
  1100   - 9 in XS-3
+ 1000   5 in XS-3
------
 10100   (carry)
+ 0011   add -3.
------
 0100 0111
```

(b) 4 & 3

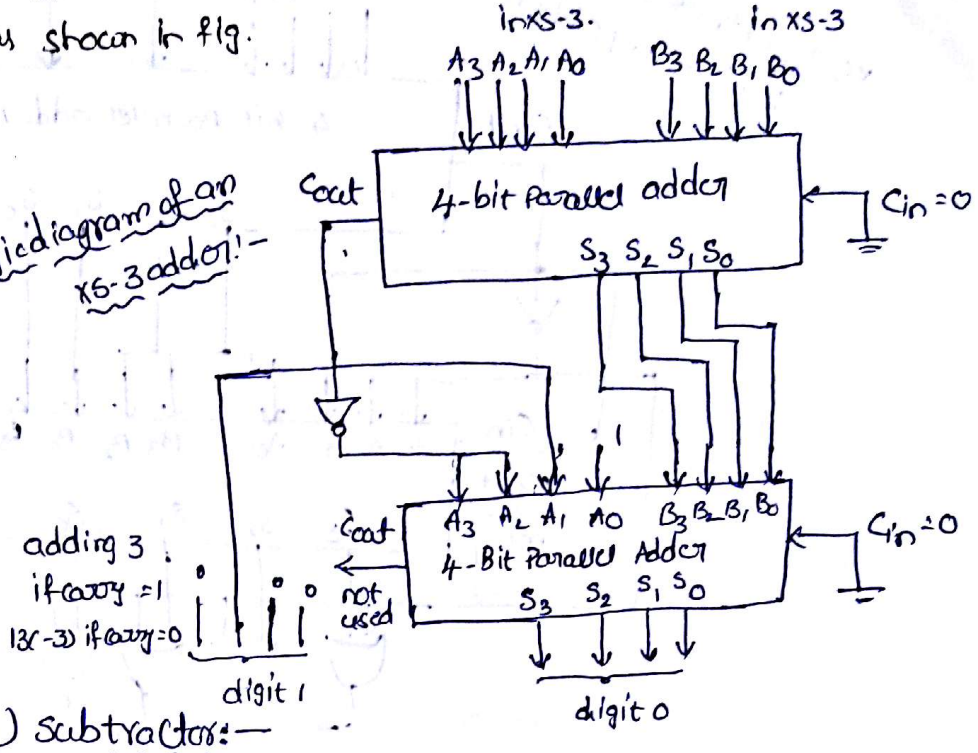
```
  0111
+ 0110
------
 1101   no carry
+ 1101   add (13)
------
 1 1010
```

Ignore carry ←

Implementation of XS-3 adder using 4-bit binary adders is shown in fig. The inputs $A_3A_2A_1A_0$ & $B_3B_2B_1B_0$ in XS-3 are added using the 4-bit parallel adder. If carry is a 1, then 0011 (3) is added to the sum bits $S_3S_2S_1S_0$ of the upper adder in the lower 4-bit parallel adder.

If carry '0', then 1101 (13) is added to the sum bits. The correct sum in XS-3 is obtained as shown in fig.

Logic diagram of an XS-3 adder:-



EXCESS-3 (XS-3) Subtractor:-

To perform Excess-3 subtraction

1. Complement the subtrahend
2. Add the complemented subtrahend to the minuend
3. If carry = 1, result is positive. Add 3 and end around carry to the result.
   If carry = 0, result is negative. subtract 3. i.e add 13 and take 1's complement of the result.

Eg:- ① 9 − 4

```
  1 1 0 0
  1 0 0 0   complement of 4 in XS-3
(1)0 1 0 0
+ 0 0 1 1   Add (3)
-----------
  0 1 1 1
        1   end around carry
-----------
  1 0 0 0   5 in XS-3
```

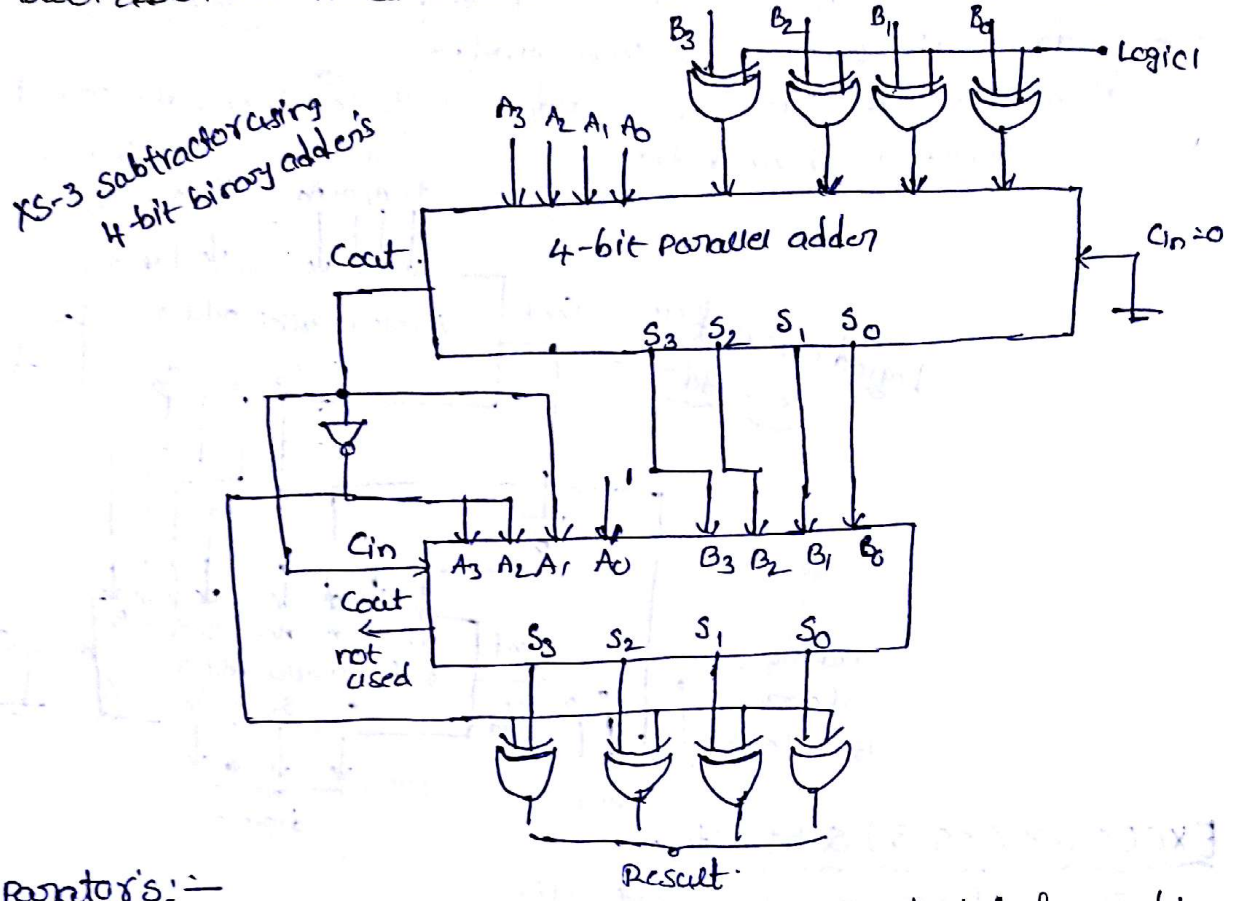
② 4 − 9.

```
  0 1 1 1
  0 0 1 1   Complement of 9 in XS-3
-----------
  1 0 1 0
+ 1 1 0 1   adding '13'
-----------
(1)0 1 1 1
-----------
  1 0 0 0   → complement the result
```

→ −5 in XS-3

Implementation of XS-3 subtractor using two 4-bit parallel adders as shown in fig. the minuend and the 1's complement of the subtrahend in XS-3 are added in the upper 4-bit parallel adder. If carry-out from the upper adder is '0', then 1101 is added to the sum bits of the upper adder in the lower adder and

the sum bits of the lower adder are complemented to get the result. If the carry-out from the upper adder is a '1', then 3 = 0011 is added to sum bits of the lower adder and the sum bits of the lower adder give the result.



Comparators:-

It is a logic circuit used to "compare the magnitudes" of two binary numbers. Depending on the design it may either simply provide an output that is active (High) when the two numbers are equal, or additionally provide outputs that signify which of the numbers is greater when equality does not hold.

The X-NOR gate is a basic comparator, because its o/p is a '1' only if its two input bits are equal; i.e the o/p is a '1' if and only if the i/p bits coincide.

Eg:- Two 4-bit binary numbers, A_3, A_2, A_1, A_0 & B_3, B_2, B_1, B_0 are equal if $A_3=B_3$, i.e $A_2=B_2$, $A_1=B_1$ & $A_0=B_0$. Thus, equality holds when A_3 coincides with B_3, A_2 with B_2, A_1 with B_1, and A_0 coincides with B_0.

$$\therefore \text{EQUALITY} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

The block diagram of a 1-bit comparator which can be used as a module for comparison of larger numbers.

G.NAVEEN
EEE DEPT

1-Bit Magnitude Comparator:-

The logic for a 1-bit magnitude comparator

Let 1-bit numbers be $A = A_0$ & $B = B_0$.

→ If $A_0 = 1$, & $B_0 = 0$ then $A > B$

$\therefore$ $A > B$; $G_1 = A_0\overline{B_0}$

→ If $A_0 = 0$ & $B_0 = 1$, then $A < B$

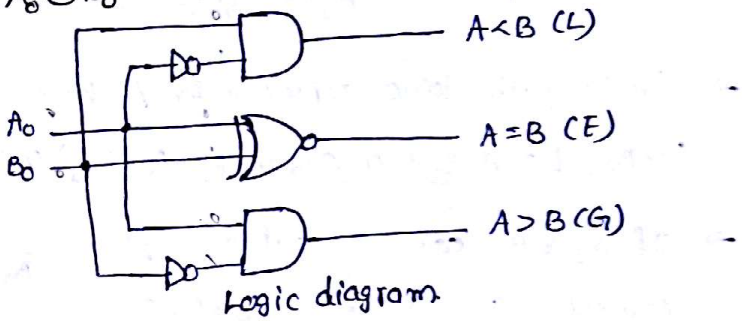
$\therefore$ $A < B$; $L = \overline{A_0}B_0$

→ If A_0 & B_0 coincide; ie $A_0 = B_0 = 0$ (or) if $A_0 = B_0 = 1$ then $A = B$

$\therefore A = B$; $E = A_0 \odot B_0$

Truth table & logic diagram

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



Logic diagram

2-Bit Magnitude comparator:-

The logic for a 2-bit magnitude comparator:

Let the two 2-bit numbers be $A = A_1A_0$ and $B = B_1B_0$

→ 1. If $A_1 = 1$. & $B_1 = 0$ then $A > B$ (or)

2 If A_1 & B_1 coincide & $A_0 = 1$ & $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is

$A > B$; $G_1 = A_1\overline{B_1} + (A_1 \odot B_1)A_0\overline{B_0}$

→ 1. If $A_1 = 0$ & $B_1 = 1$, then $A < B$ (or)

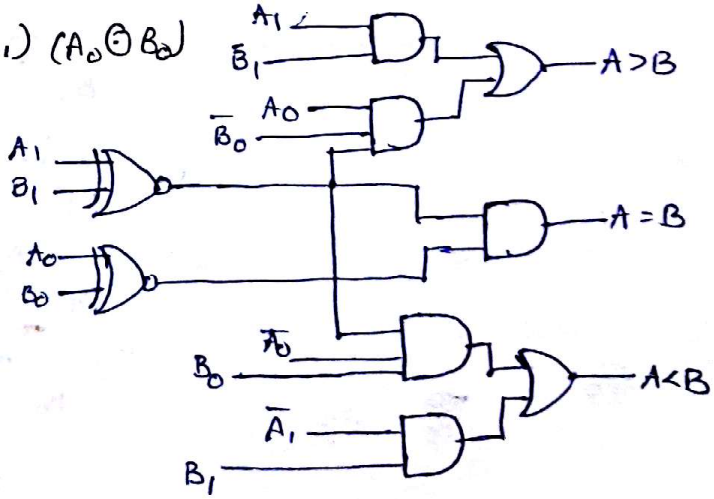
2. If A_1 & B_1 coincide & $A_0 = 0$, & $B_0 = 1$, then $A < B$. So the expression for $A < B$ is

$A < B$; $L = \overline{A_1}B_1 + (A_1 \odot B_1)\overline{A_0}B_0$

→ If A_1 & B_1 coincide & if A_0 & B_0 coincide then $A = B$. So the expression for $A = B$ is

$A = B$; $E = (A_1 \odot B_1)(A_0 \odot B_0)$

Logic diagram:-



4-Bit Magnitude Comparator :-

The logic for a 4-bit magnitude comparator :

Let the two 4-bit numbers be $A = A_3 A_2 A_1 A_0$ & $B = B_3 B_2 B_1 B_0$

1. If $A_3 = 1$ and $B_3 = 0$, then $A > B$ or
2. If A_3 and B_3 coincide, & if $A_2 = 1$ and $B_2 = 0$ then $A > B$ or
3. If A_3 and B_3 " " A_2 & B_2 coincide, & if $A_1 = 1$, & $B_1 = 0$, then $A > B$ or
4. If A_3 & B_3 " " if A_2 & B_2 " " if A_1 & B_1 coincide & if $A_0 = 1$, & $B_0 = 0$ then $A > B$

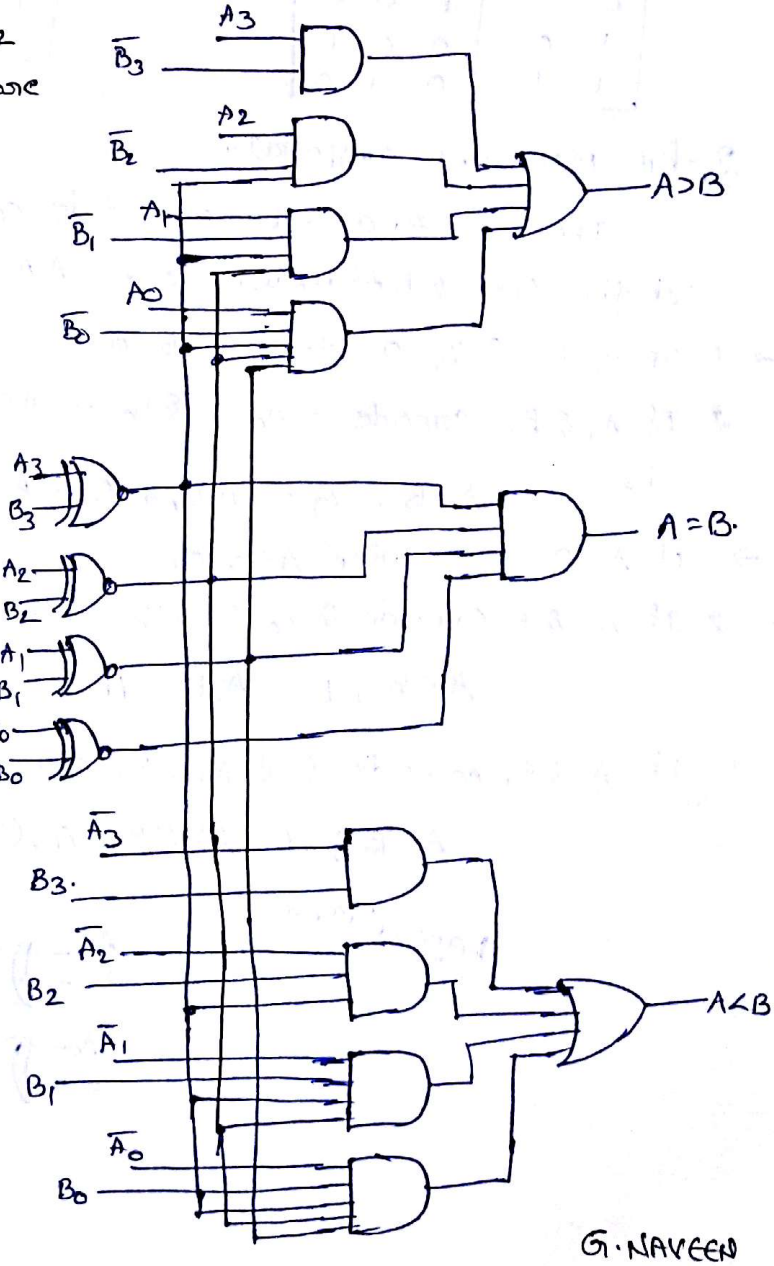
$\therefore (A>B);\ G_1 = A_3\bar{B}_3 + (A_3 \odot B_3) A_2\bar{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2) A_1\bar{B}_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) A_0\bar{B}_0$

→ Similarly the logic expression for $A < B$ is

$(A<B);\ L = \bar{A}_3 B_3 + (A_3 \odot B_3)\bar{A}_2 B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1 B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0 B_0$

→ If A_3 & B_3 coincide, & if A_2 & B_2 coincide & if A_1 & B_1, & A_0 & B_0 are coincide then $A = B$.

$(A=B);\ E = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$



G. NAVEEN
EEE Dept

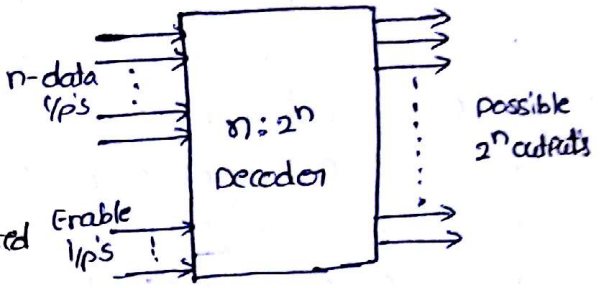
Decoder:—

Decoder is a multiple-input, multiple output logic circuit which converts coded inputs into coded o/p's, where the i/p & o/p codes are different.

→ i/p code generally has fewer bits than the o/p code. Each input code word produces a different o/p word i.e there is one to one mapping from i/p code words into o/p code words.

→ the encoded information is presented as n i/p producing 2^n possible outputs. the 2^n o/p values are from '0' through 2^n-1

→ some times an n-bit binary code is truncated to represent fewer o/p values than 2^n.



Binary Decoder:—

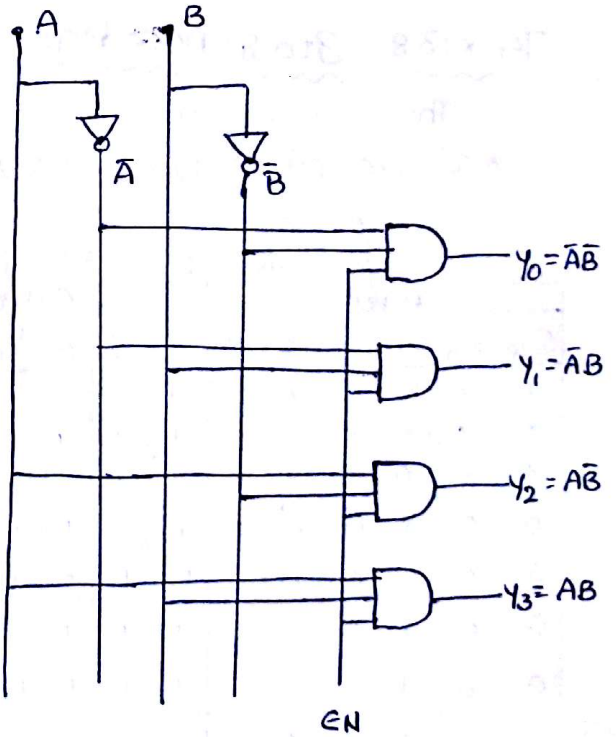
A Decoder which has an n-bit binary input code and a one activated output out of 2^n o/p code is called "Binary Decoder".

A Binary decoder is used when it is necessary to activate exactly one of 2^n o/p's based on an n-bit i/p value.

Fig shows 2 to 4 decoder. Here 2 i/p's are decoded into four o/p's each o/p representing one of the minterms of the 2 i/p variables.

INPUTS			OUTPUTS			
EN	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

From Truth table when EN=1, then only the output Y_0 to Y_3 is active for given input.



2 to 4 Decoder.

Eg:- Draw the circuit for 3 to 8 decoder using logic gates & Truth Table.

Sol: Here, 3 i/p's are decoded into 8 outputs. each output represent one of the minterms of the 3-i/p variables.

Truth Table:-

	Inputs			outputs							
EN	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Enable i/p is provided to activate decoded output based on data inputs A, B and C.

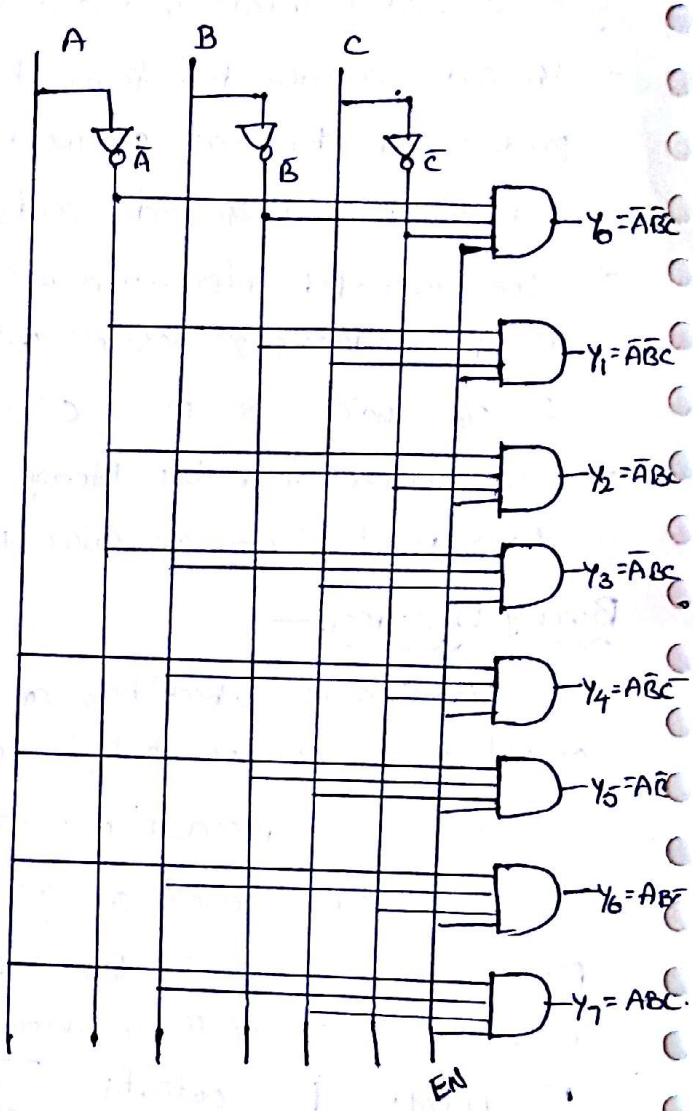


Fig: 3-8 line decoder:-

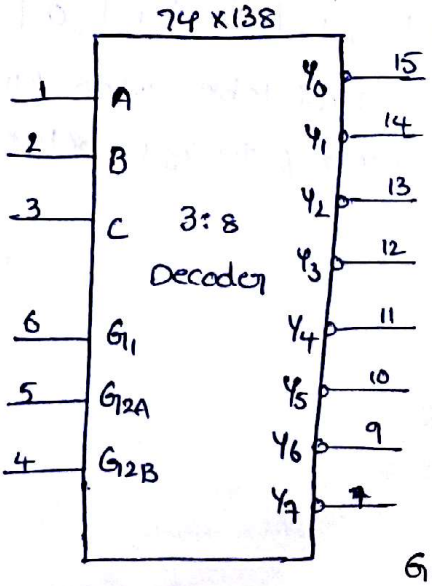
74x138 3 to 8 Decoder:-

→ The 74x138 is a commercially available, 3 to 8 decoder

→ It accepts 3 binary i/p's (A, B, C) and when enabled provides eight individual active low o/p's (Y_0-Y_7)

→ The device has 3 Enable inputs: 2 active low (G_{2A}, G_{2B}) and one active high G_1.

Inputs						Outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\bar{Y}_7$	$\bar{Y}_6$	$\bar{Y}_5$	$\bar{Y}_4$	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

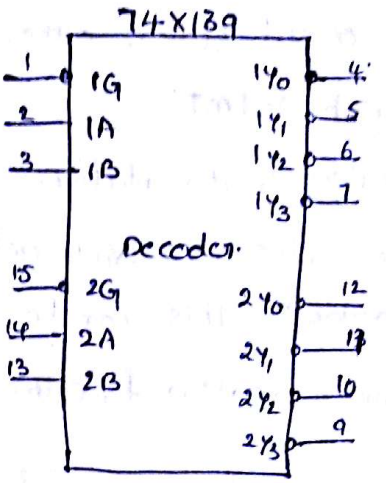


G. NAVEEN
EEE Dept

74X139 Dual 2 to 4 Decoder:—

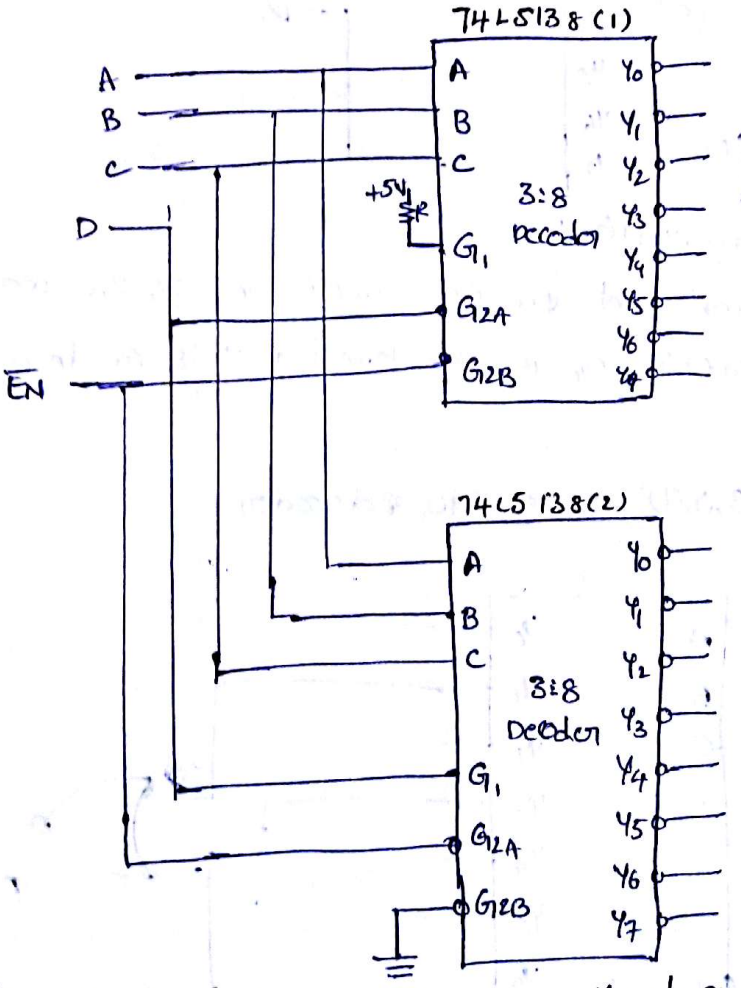
It consists of two independent and identical 2 to 4 decoders the enable i/p's and outputs are active low.

Inputs			outputs			
$\bar{G}$	B	A	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1



Cascading Binary Decoders:— (4X16 Decoder)

Binary decoder CKT's can be connected together to form a bigger decoder circuit. Fig shows 4X16 decoder using two 3X8 decoders



one I/p line D, is used to enable/disable the decoders.
when D=0 the top decoder is enabled the other is disabled. Thus bottom decoder o/p's are all 1's. and the top 8 o/p generate minterms 0000 to 0111.
when D=1 the enable conditions are reversed and thus bottom decoder o/p's generate minterms 1000 to 1111, while the o/p's of the top decoder are all 1's.

Realization of Multiple output Function using Binary Decoder:-

The combination of decoder and external logic gates can be used to implement single or multiple output functions. The decoder have one of the two output states: either active low or active High.

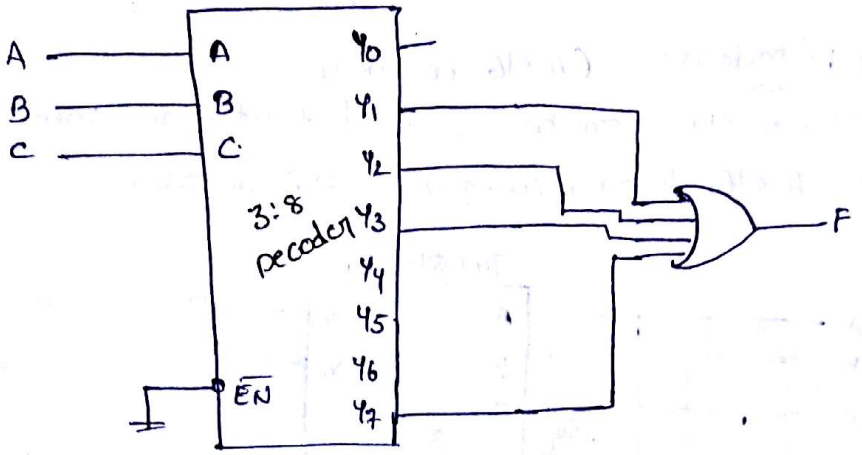
Active High Output:-

Sop Function Implementation:-

when decoder o/p is active High, it generates minterms (Product terms) for i/p variables. This can be implemented by ORing the selected decoder o/p's.

eg:- Implementation function $f = \Sigma M(1,2,3,7)$ using 3 to 8 Decoder.

Sol:-

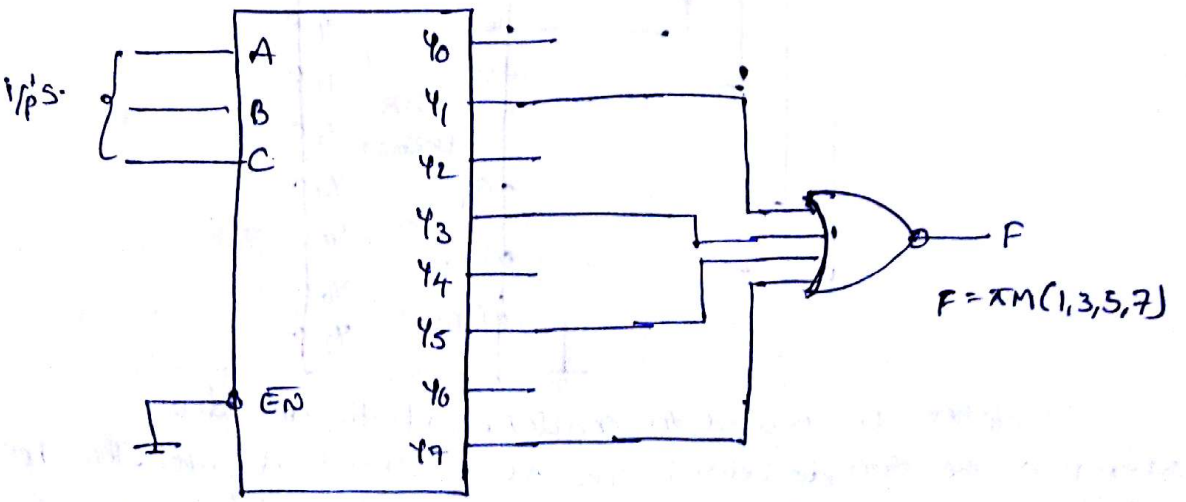


Pos Function Implementation:-

We can implement the pos function in similar manner as for Sop function except function o/p is complemented. This can be achieved by connecting NOR gate.

eg:- $f = \pi M(1,3,5,7)$ using 3 to 8 decoder

Sol:-



G.NAVEEN
EEE Dept

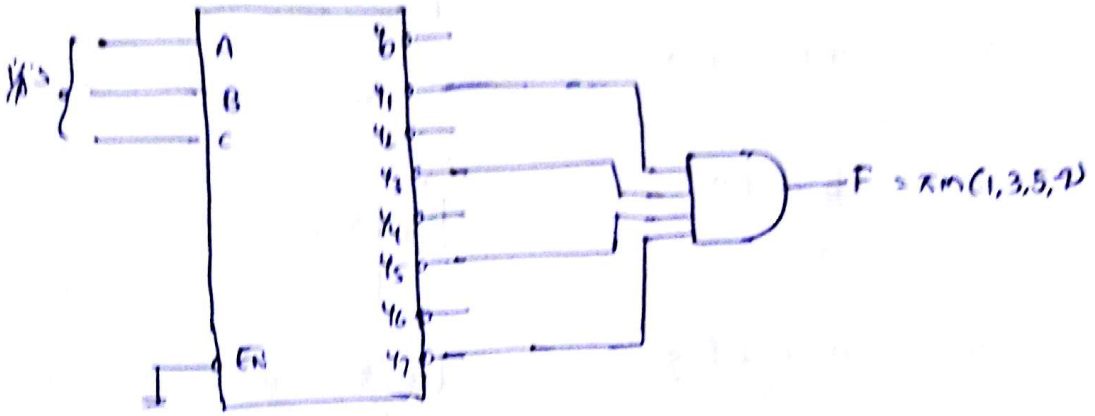
Active low output:-

POS function Implementation:-

When decoder o/p is active low, the o/p is in complemented form i.e it generates maxterms (sum terms) for i/p variables. This can be achieved by ANDing the selected decoder outputs

Eg:- $f = \pi M(1,3,5,7)$ using 3:8 decoder

Sol:

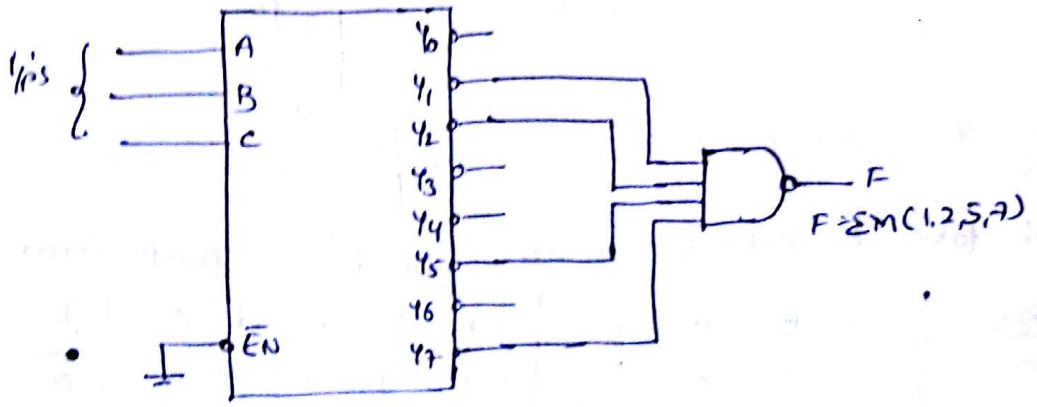


SOP Function Implementation:-

When decoder o/p is active low, we can implement the SOP function in similar manner as for POS function except function o/p is complemented. This can be achieved by connecting NAND gate instead of AND gate.

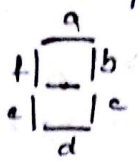
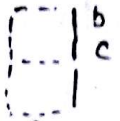
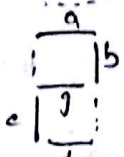
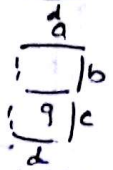
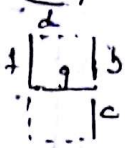
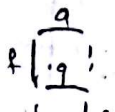
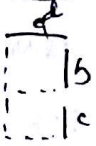
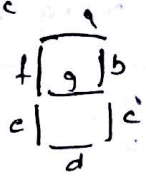
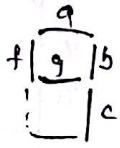
Eg:- $f = \Sigma m(1,2,5,7)$ using 3:8 decoder

Sol:



BCD to 7 segment display Decoder:-

In practical applications, 7 segment display are used to give a visual identification of the o/p sates. The special BCD to 7 segment decoder/driver ICs are used to convert the BCD signal into a form suitable for driving these displays.

Digit	Segments activated	Display
0	a, b, c, d, e, f	
1	b, c	
2	a, b, d, e, g	
3.	a, b, c, d, g	
4	b, c, f g	
5	a, c, d, f, g	
6	a, c, d, e, f, g	
7	a, b, c	
8	a, b, c, d, e, f, g	
9.	a, b, c, d, f, g	

Truth table for BCD-to-common-cathode 7-segment 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

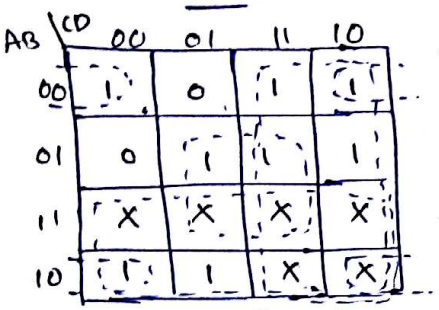
G. NAVEEN
EEE DEPT

Truth table for BCD to common - anode, 7-segment decoder :-

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

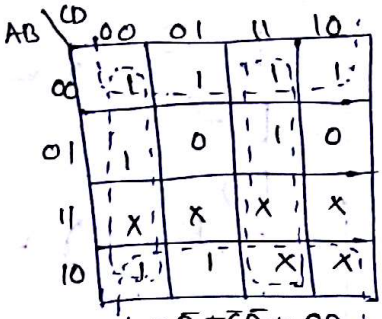
k-map Simplification for cathode :-

For a



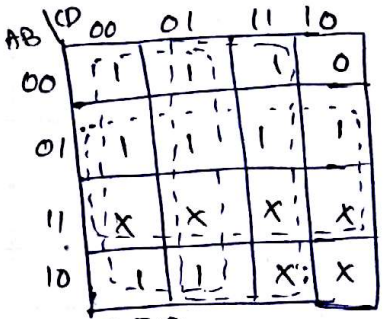
$a = A + C + BD + \bar{B}\bar{D}$

For b



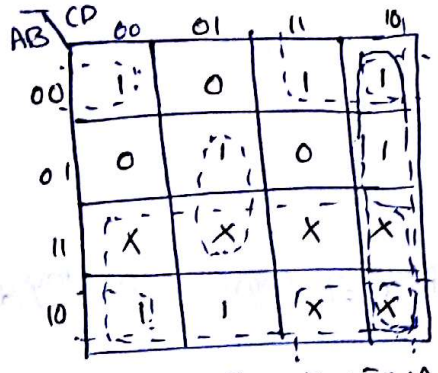
$b = \bar{B} + \bar{C}\bar{D} + CD$

For c



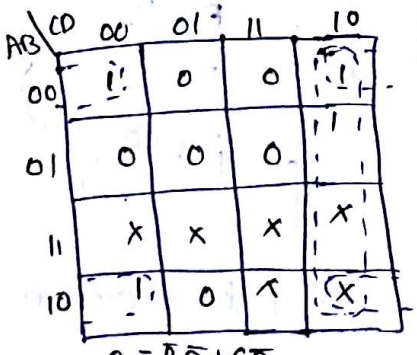
$c = B + \bar{C} + D$

For d



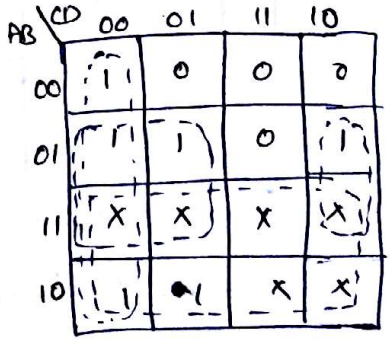
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$

For e



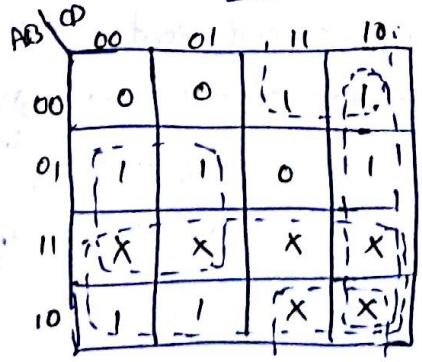
$e = \bar{B}\bar{D} + C\bar{D}$

For f

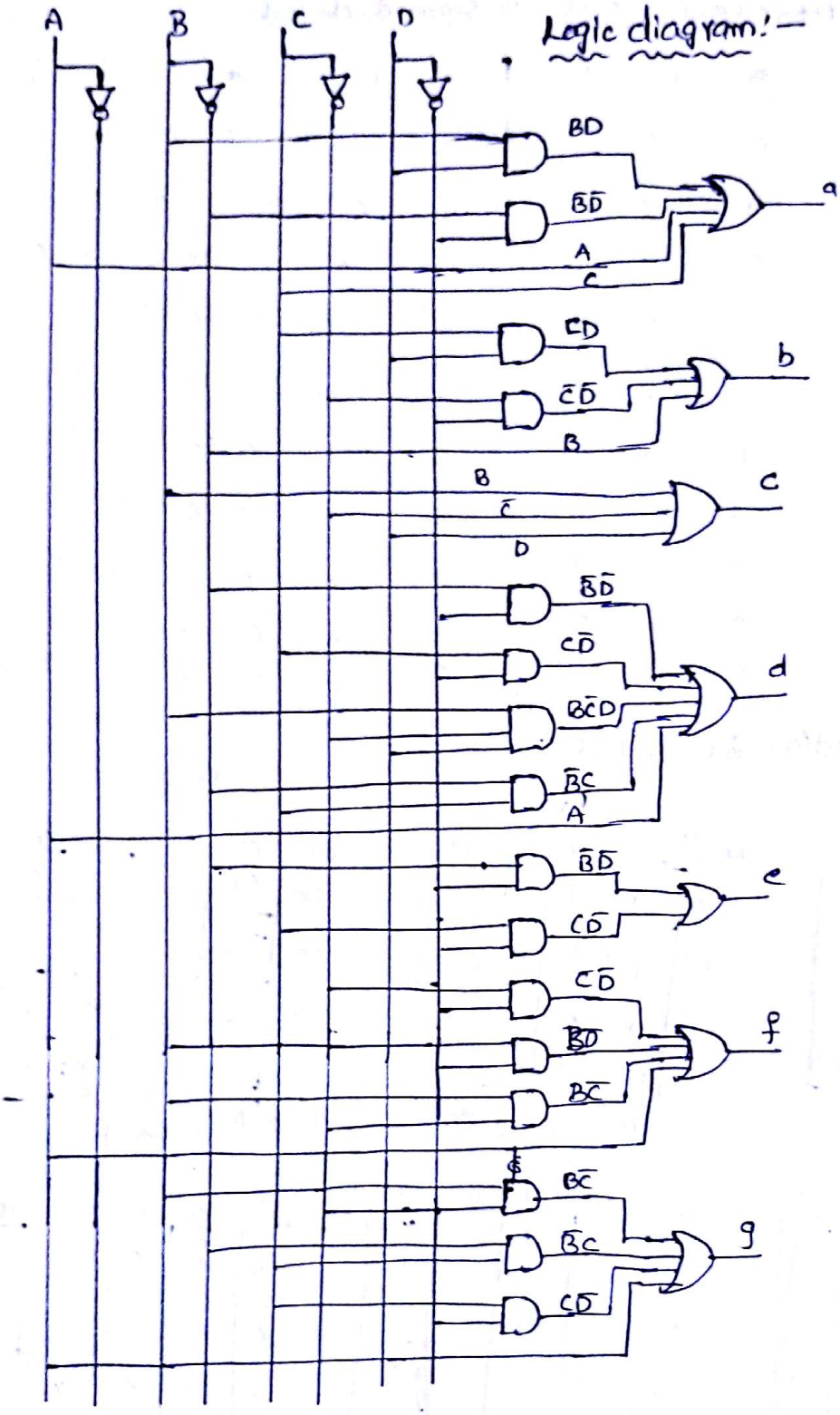


$f = A + C\bar{D} + B\bar{C} + \bar{B}\bar{D}$

For g



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

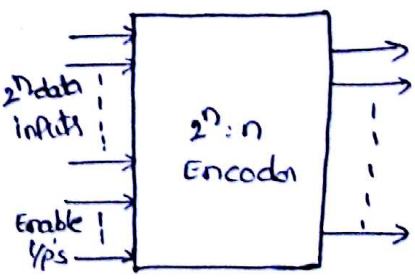


Combinational circuit for common cathode 7-segment display/driver

Encoders:-

An Encoder is a digital circuit that performs the inverse operation of a Decoder. An encoder has 2^n i/p lines and 'n' o/p lines.

In Encoder the o/p lines generate the binary code corresponding to the i/p value.

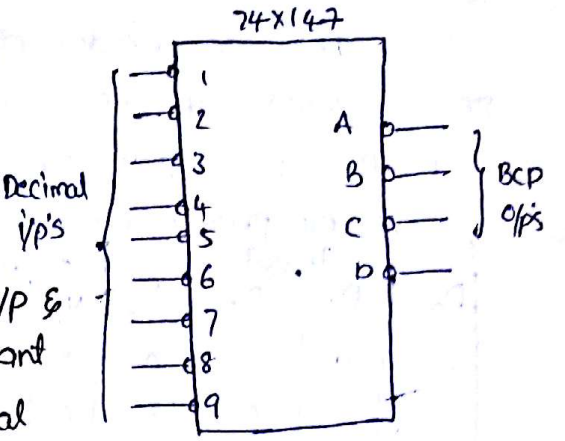


G. NAVEEN
EEE Dept

Decimal to BCD Encoder:— (Priority Encoder)

Usually it has ten input lines and four output lines. the decoded decimal data acts as an input for encoder and encoded BCD output is available on the '4' o/p lines.

→ It has 9 i/p lines and 4 o/p lines. Both i/p & o/p lines are asserted active low. It is important to note that there is no i/p line for "decimal zero". when this condition occurs, all o/p lines are '1'.



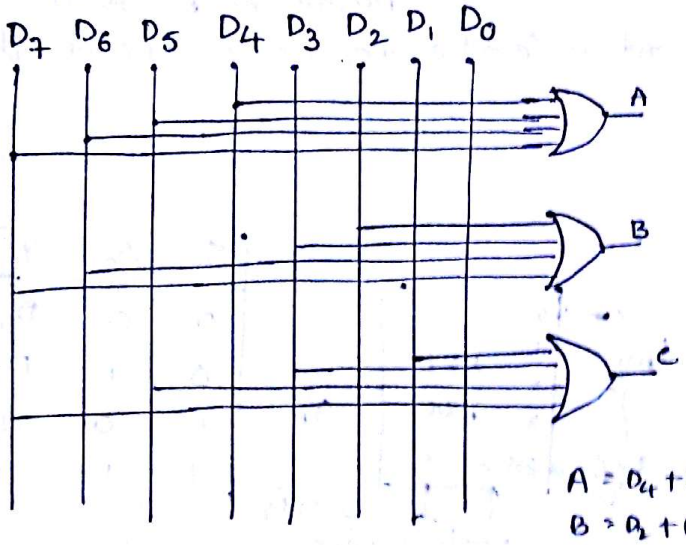
Truth table:-

Decimal	Inputs									Outputs			
	1	2	3	4	5	6	7	8	9	D	C	B	A
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	0
2	x	0	1	1	1	1	1	1	1	1	1	0	1
3	x	x	0	1	1	1	1	1	1	1	1	0	0
4	x	x	x	0	1	1	1	1	1	1	0	1	1
5	x	x	x	x	0	1	1	1	1	1	0	1	0
6	x	x	x	x	x	0	1	1	1	1	0	0	1
7	x	x	x	x	x	x	0	1	1	1	0	0	0
8	x	x	x	x	x	x	x	0	1	0	1	1	1
9	x	x	x	x	x	x	x	x	0	0	1	1	0

x - Indicates Don't care condition

Octal to Binary Encoder:

It has '8' i/p's, one for each octal digit & 3 o/p that generate the corresponding code.



			Inputs					Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

Priority Encoder:-

It is an encoder circuit that includes the priority function. In priority encoder, if two or more inputs are equal to '1' at the same time, the i/p having the highest priority.

4-bit priority encoder:-

Inputs				outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Table shows D_3 i/p with highest priority and D_0 i/p with lowest priority.

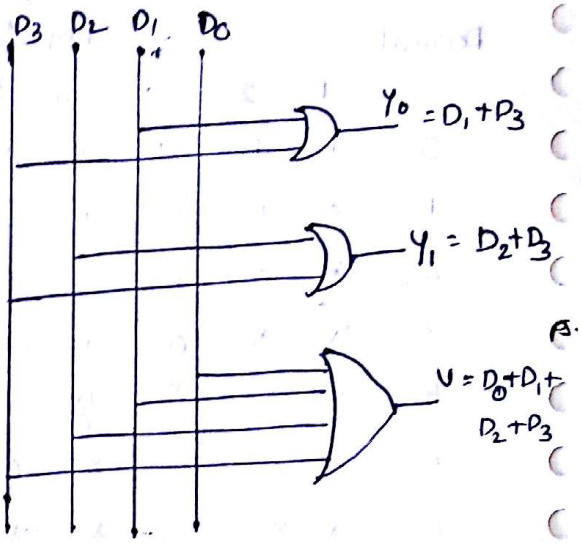
When D_3 i/p is high. output is 11.

The o/p V (valid o/p) indicates, one (or) more of the i/p's are equal to 1.

If all i/p's are '0', V=0.

K-map Simplification:-

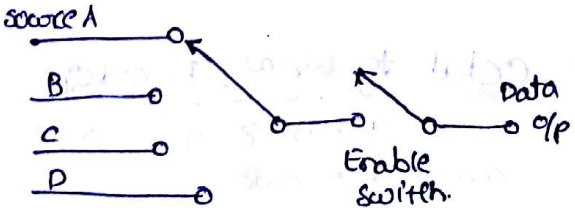
Logic



Multiplexer's:-

Multiplexer is a digital switch. It allows digital information from several sources to be routed onto a single o/p line. as shown in fig.

The multiplexer is consists of several data i/p lines and a single o/p line. the selection of a particular i/p line is controlled by a set of selection lines.

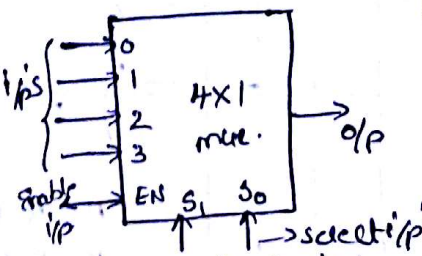


Analog selector switch.

→ Normally, These are 2^n i/p lines and n selection lines whose bits combinations determine which i/p is selected.

4 to 1 Line multiplexer:-

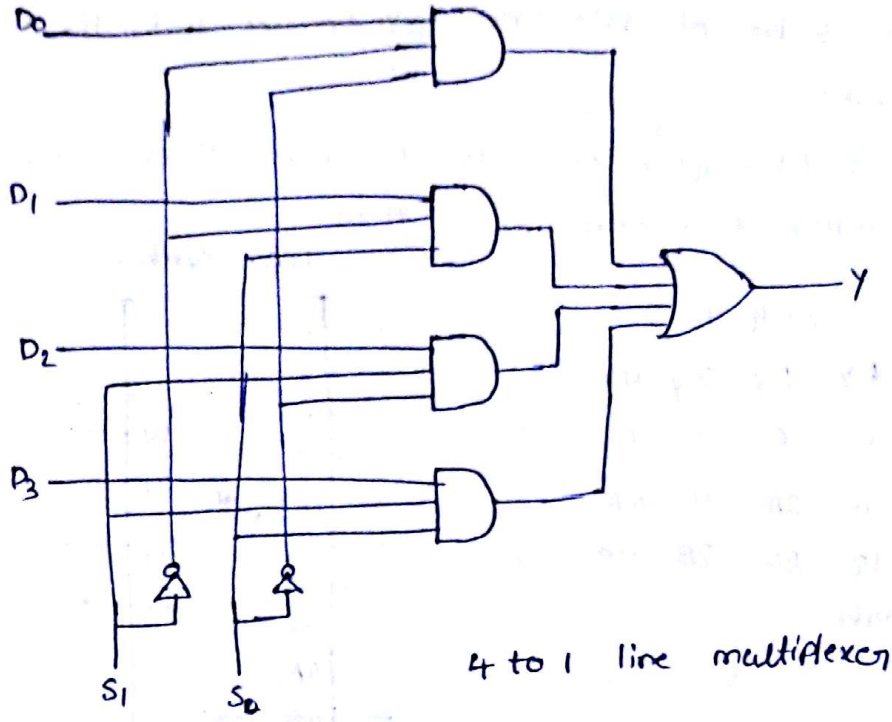
It consists four lines, D_0 to D_3, is applied to one i/p of an AND gate. selection lines are decoded to select a particular AND gate.



Logic Symbol

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

G. NAVEEN
EEE DEPT



4 to 1 line multiplexer.

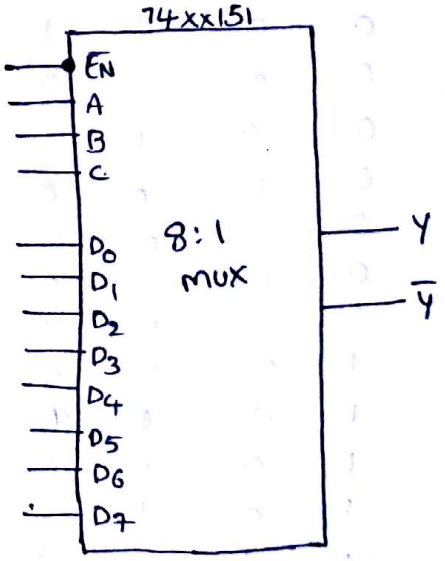
8 to 1 Multiplexer:-

It has 8 inputs, It provides two o/p's. one is active high, the other is active Low. There are 3 select inputs C, B & A which select one of the 8 i/p's.

In this o/p is not specified in is and o's. Because multiplexer is a data switch. It does not generate any data of its own. but it simply passes external i/p data from the selected i/p to the o/p. ∴ o/p column represent data by D_n & $\overline{D_n}$

Input				Outputs	
Select			Enable		
C	B	A	$\overline{EN}$	Y	$\overline{Y}$
X	X	X	1	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	1	0	D_1	$\overline{D_1}$
0	1	0	0	D_2	$\overline{D_2}$
0	1	1	0	D_3	$\overline{D_3}$
1	0	0	0	D_4	$\overline{D_4}$
1	0	1	0	D_5	$\overline{D_5}$
1	1	0	0	D_6	$\overline{D_6}$
1	1	1	0	D_7	$\overline{D_7}$

Truth Table.



Logic symbol

Quad 2-Input Multiplexer:-

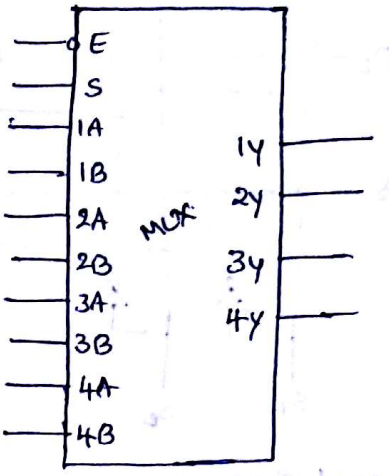
It selects 4 bits of data from two sources under the control of a common select input (S).

The Enable I/p ($\bar{E}$) is active LOW. when $\bar{E}$ is high. all of the O/p's (Y) are forced low regardless of all other i/p conditions.

Inputs		Outputs			
$\bar{E}$	S	1Y	2Y	3Y	4Y
1	X	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

Truth Table.

Logic symbol

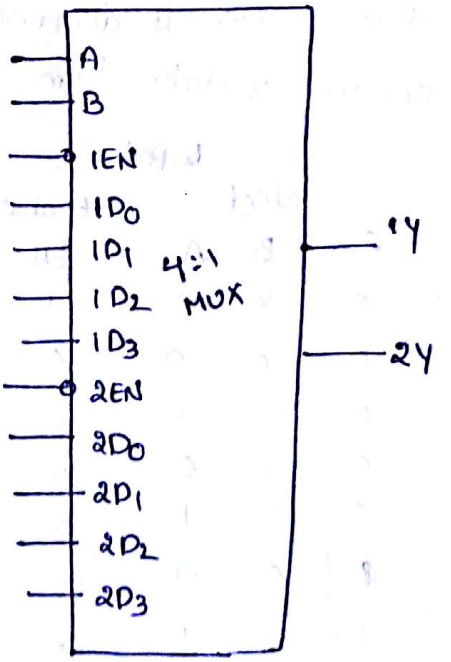


Dual 4 to 1 Multiplexer:-

It contains two identical and independent 4 to 1 multiplexers. Each multiplexer has separate enable inputs.

	Inputs			Outputs	
1EN	2EN	B	A	1Y	2Y
0	0	0	0	$1D_0$	$2D_0$
0	0	0	1	$1D_1$	$2D_1$
0	0	1	0	$1D_2$	$2D_2$
0	0	1	1	$1D_3$	$2D_3$
0	1	0	0	$1D_0$	0
0	1	0	1	$1D_1$	0
0	1	1	0	$1D_2$	0
0	1	1	1	$1D_3$	0
1	0	0	0	0	$2D_0$
1	0	0	1	0	$2D_1$
1	0	1	0	0	$2D_2$
1	0	1	1	0	$2D_3$
1	1	X	X	0	0

Truth Table.

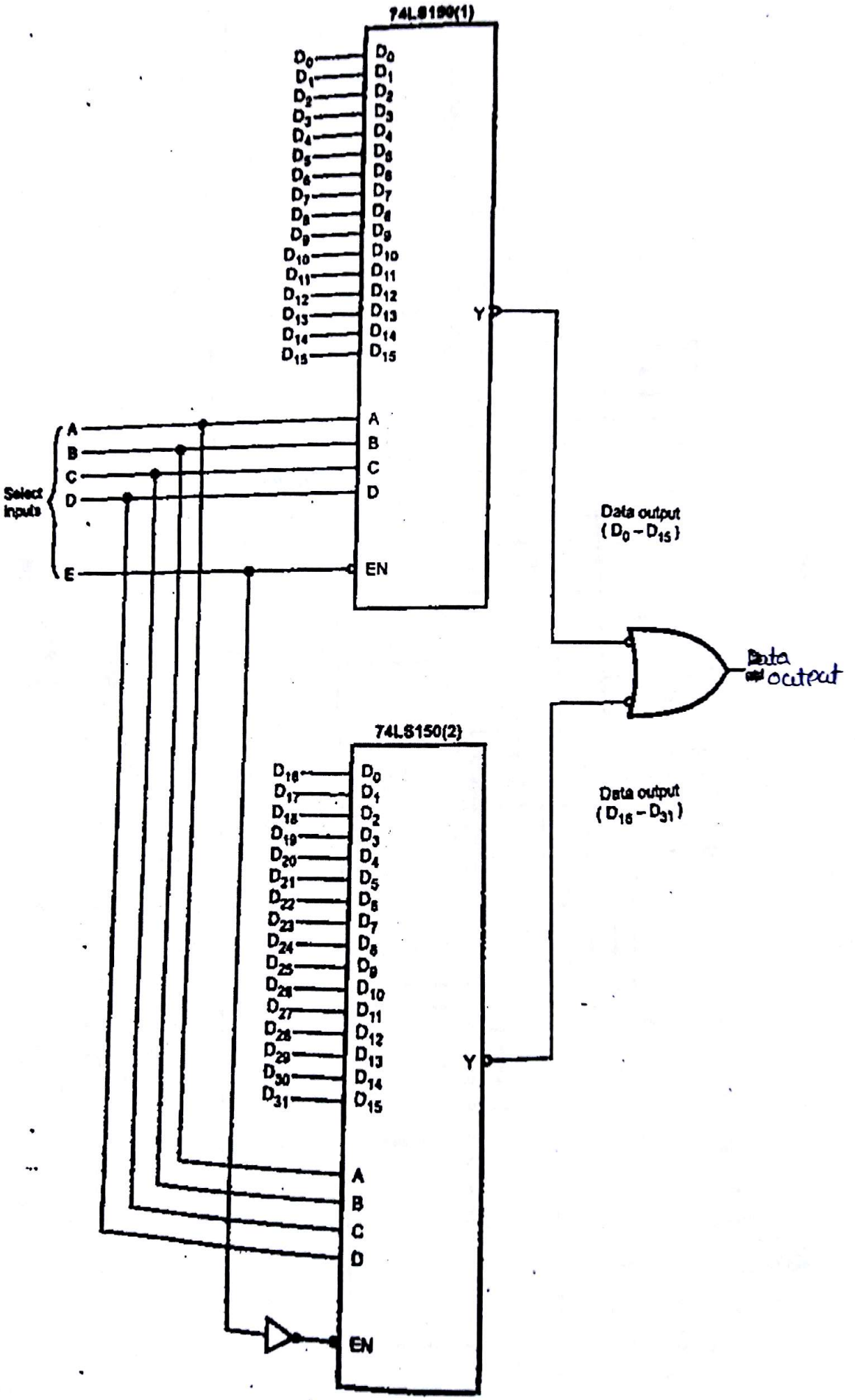


G.NAVEEN
EEE DEPT

Higher order multiplexing :- (or Expanding Multiplexers)

Eg:- Design 32 to 1 multiplexer.

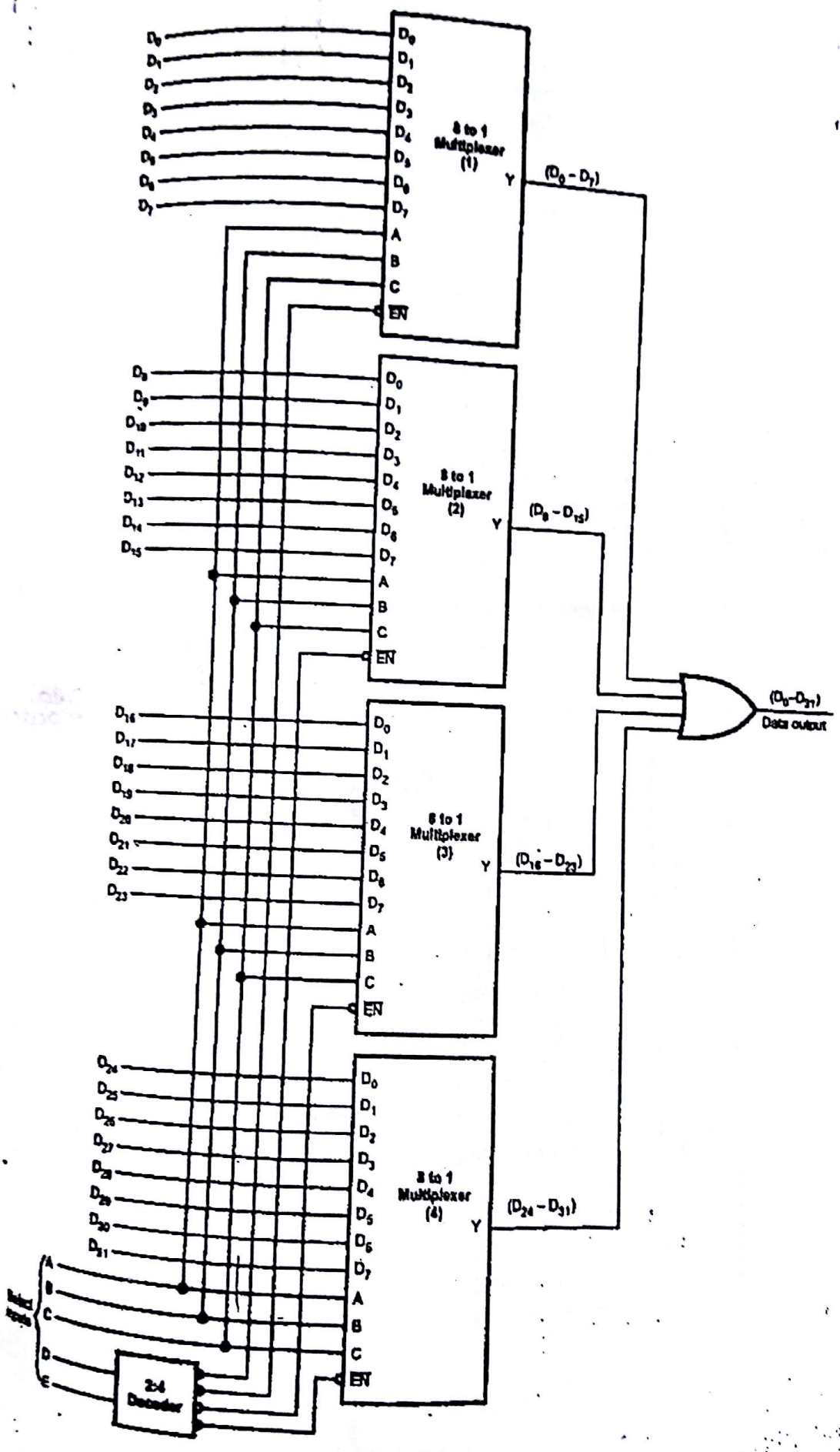
Sol:-



32 to 1 multiplexer

Q: Design 32 to 1 multiplexer using '4' 8 to 1 multiplexers & 2 to 4 decoder.

Sol:



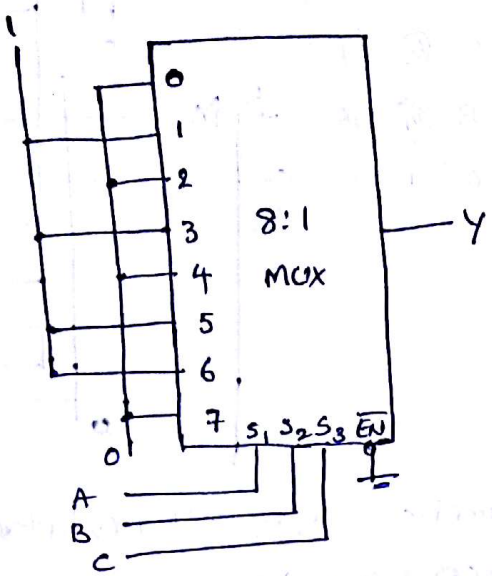
G. NAVEEN
EEE Dept

Implementation of Combinational Logic using MUX:-

Eg:- Implement the following Boolean function using 8:1 multiplexer

$F(A,B,C) = \Sigma m(1,3,5,6)$

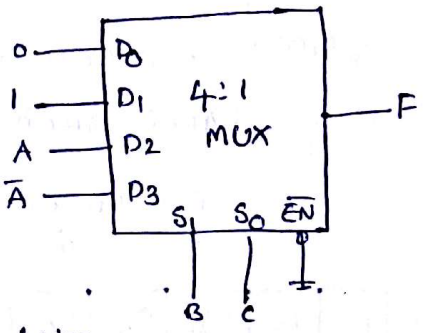
Sol:- A, B, C are applied to the select lines.



Eg: $F(A,B,C) = \Sigma m(1,3,5,6)$ using 4:1 multiplexer

Sol:-

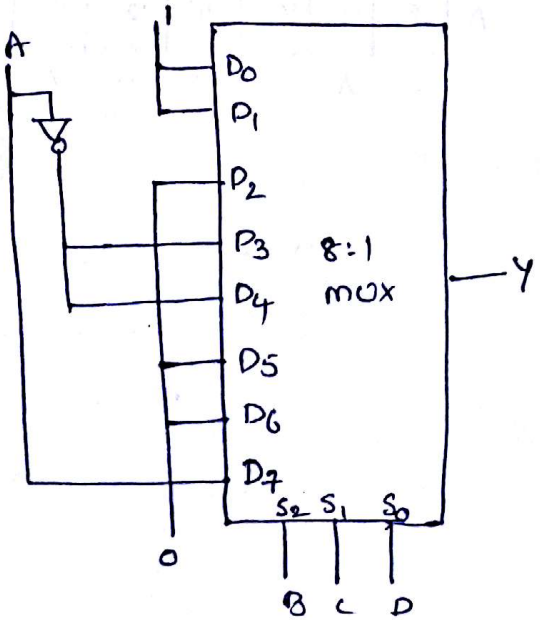
	D_0	D_1	D_2	D_3	
$\bar{A}$	0	(1)	2	(3)	row 1
A	4	(5)	(6)	7	row 2
	0	1	A	$\bar{A}$	



Eg:- $F(A,B,C,D) = \Sigma m(0,1,3,4,8,9,15)$ using 8:1 mux

Sol:-

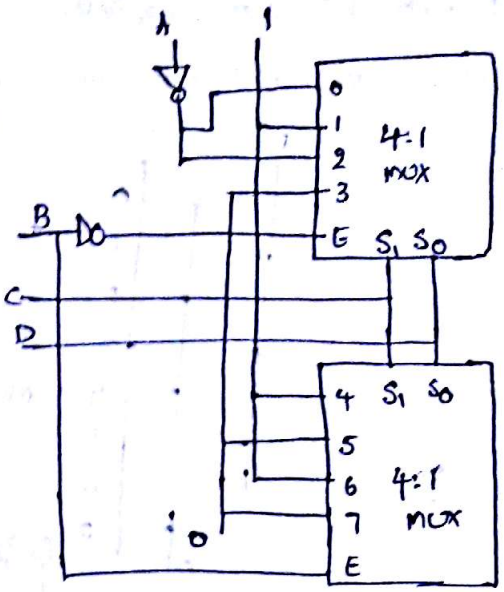
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A



Eg:- $F(A,B,C,D) = \Sigma m(0,1,2,4,6,9,12,14)$ using 4:1 MUX.

Sol:- The function has 4 variables. To implement the function we require 8:1 mux. i.e two 4:1 mux

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	3	(4)	5	(6)	7
A	8	(9)	10	11	(12)	13	(14)	15
	$\bar{A}$	1	$\bar{A}$	0	1	0	1	0



Eg: Implement the Boolean function using 8:1 multiplexer.

$F(A,B,C,D) = \bar{A}B\bar{D} + ACD + \bar{B}CD + \bar{A}\bar{C}D$.

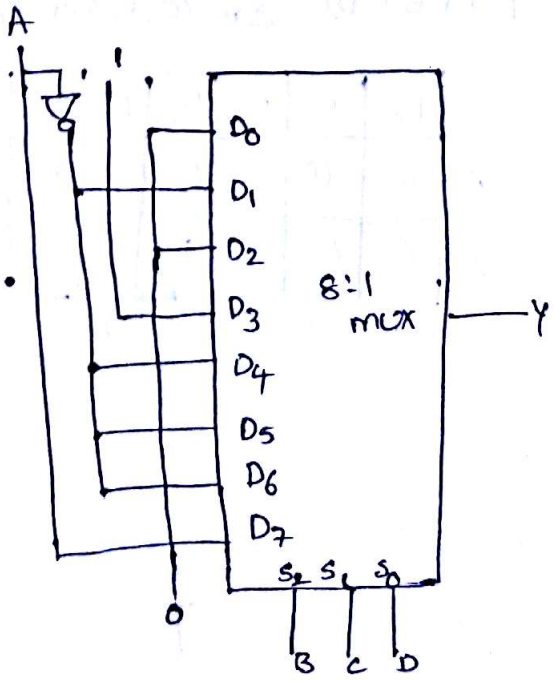
Sol: Convert given function in standard SOP form.

$$F(A,B,C,D) = \bar{A}B\bar{D}(C+\bar{C}) + ACD(B+\bar{B}) + \bar{B}CD(\bar{A}+A) + \bar{A}\bar{C}D(B+\bar{B})$$

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

$$= \underset{6}{\bar{A}BC\bar{D}} + \underset{4}{\bar{A}B\bar{C}\bar{D}} + \underset{15}{ABCD} + \underset{11}{A\bar{B}CD} + \underset{3}{\bar{A}\bar{B}CD} + \underset{5}{\bar{A}B\bar{C}D} + \underset{1}{\bar{A}\bar{B}\bar{C}D}$$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	2	(3)	(4)	(5)	(6)	7
A	8	9	10	(11)	12	13	14	(15)
	0	$\bar{A}$	0	1	$\bar{A}$	$\bar{A}$	$\bar{A}$	A

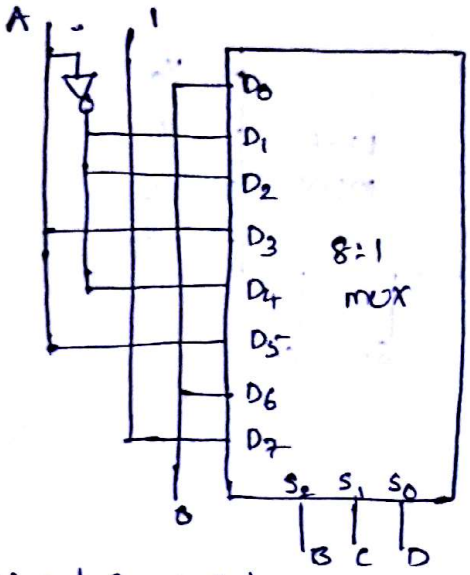


G. NAVEEN
EEE Dept

Eg:- $F(A,B,C,D) = \pi M(0,3,5,8,9,10,12,14)$ using 8:1 mux

Sol:- Given function is in maxterms. We have to circle the maxterms which are not included in the function.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	(2)	3	(4)	5	6	(7)
A	8	9	10	(11)	12	(13)	14	(15)
	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$	A	0	1

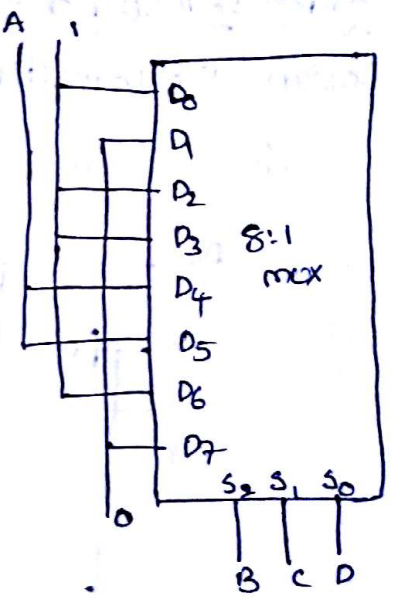


Eg:- $F(A,B,C,D) = \Sigma m(0,2,6,10,11,12,13) + d(3,8,14)$ using 8:1 mux

Sol:-

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	1	(2)	(3)	4	5	(6)	7
A	(8)	9	(10)	(11)	(12)	(13)	(14)	15
	1	0	1	1	A	A	1	0

By taking don't care conditions 8 & 14 as 1's we eliminated $\bar{A}$ term.



Demultiplexers:-

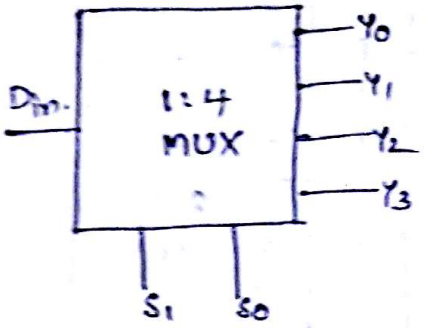
Demultiplexer is a circuit that receives information on a single line and transmits the information on one of 2^n possible output lines.

The selection of specific output line is controlled by the values of n selection lines. The single i/p variable Din has a path to all four outputs, but the i/p information is directed to only one of the o/p lines.

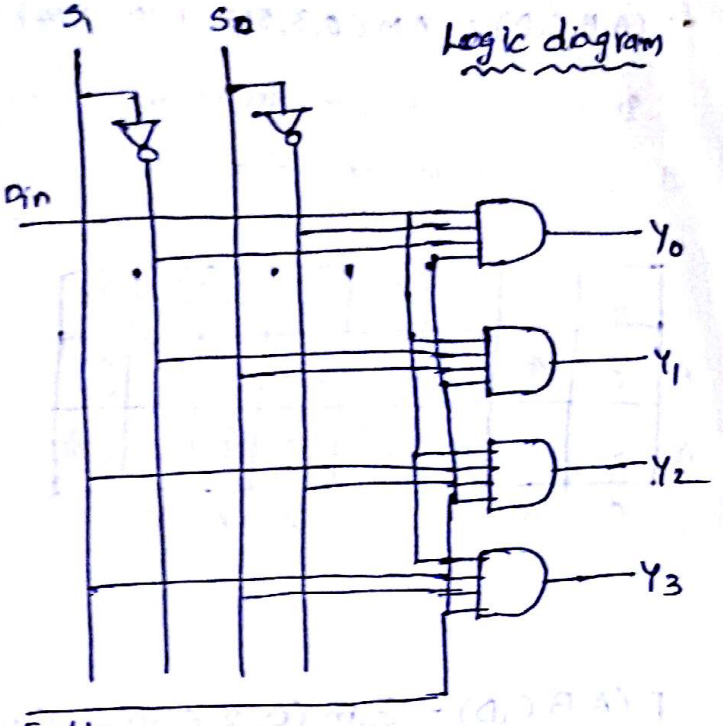
Function table for 1:4 Demux

Enable	S_1	S_0	Din	Y_0	Y_1	Y_2	Y_3
0	X	X	X	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

Logic Symbol



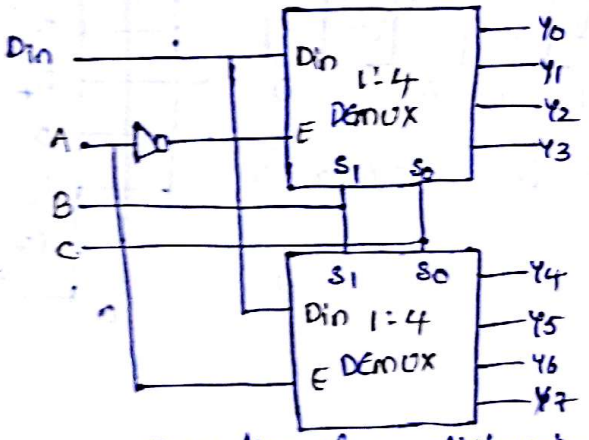
Logic diagram



Higher order Demultiplexer:-

Eg:- Design 1:8 Demultiplexer using two 1:4 demultiplexers

Sol:- cascading of de-multiplexers is similar to the cascading of decoder.

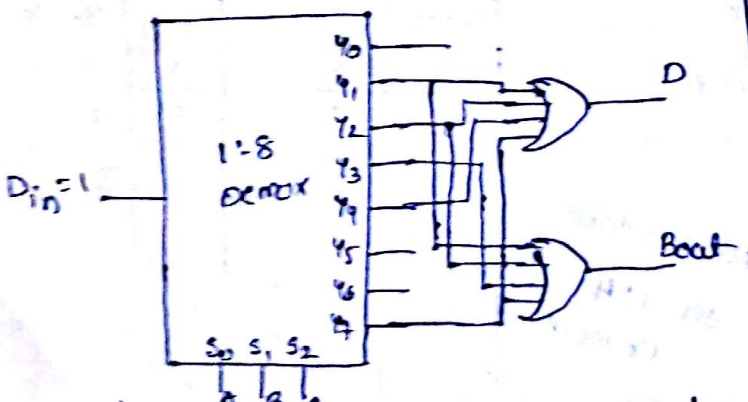


Cascading of Demultiplexers.

Eg: Implement full subtractor using demultiplexer.

Sol:- Difference (D) = $f(A,B,C) = \Sigma m(1,2,4,7)$

Bout = $F(A,B,C) = \Sigma m(1,2,3,7)$



A	B	Bin	D	Bout
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

Truth table for full subtractor.

Din = 1, it gives minterms at the output

G.NAVEEN
EEE Dept