Unit-I

Combinational Circuits

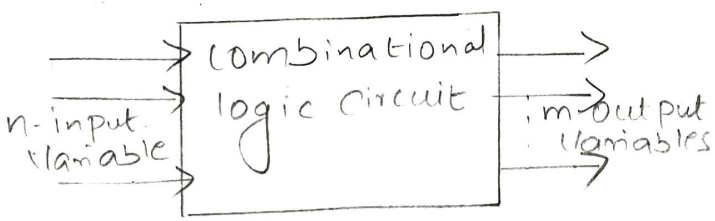
Combinational Circuits - Karnaugh map (K-map) - Design & Analysis Procedure - Binary Adder - Decimal Adder - Magnitude Comparator - Encoder - Decoder - Multiplexers - De Multiplexers

Combinational Circuits

A combinational circuit consists of input variables, logic gates & output variables

The combinational circuit accepts n-input binary variables & generates output variables depending on the logical combination of gates

(or) The present o/p is depend only on present i/p.



Block diagram of a combinational circuit.

K-map (Karnaugh map)

Gate level minimization is the design task of finding an optimal gate-level implementation of the Boolean functions describing a digital circuit.

→ It is used to minimize the Boolean functions without using laws & theorems.

→ The Karnaugh map (K-map) is a diagram made up of squares. Each square represents one minterm of the function that is to be minimized. 2^n → no. of variables
↳ no. of cells

→ Group of two cells - Pair
→ Group of four cells - quad
→ Group of Eight cells - Octet

2 variable K-map (contains $2^2 = 4$ Squares)
3-variable K-map (contains $2^3 = 8$ Squares)
4-variable K-map (contains $2^4 = 16$ Squares)
5-variable K-map (contains $2^5 = 32$ Squares)

Rules for grouping

1. Groups may not contain Zero.
2. We can group 1, 2, 4, 8 (or) 2^n cells.
3. Each group should be large as possible
4. Cells containing 1 must be grouped.
5. Groups may overlap
6. Opposite grouping & Corner grouping is allowed.

7. There Should be a least groups as possible.

8. Diagonal grouping & odd number of grouping is not allowed.

Two-Variable K-map

In 2-variable K-map, there are 4 Squares (2^2), one for each minterms.

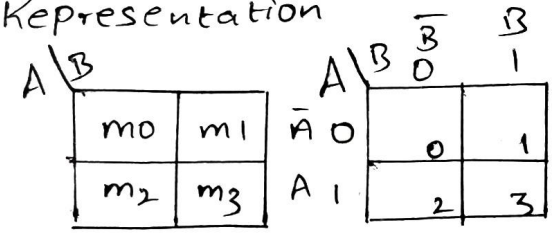
Here,

→ One Square represents a minterm, giving a term with 2 literals.

→ Two adjacent Square, represents a term with 1 literal.

→ Four adjacent Squares represent a function that is always equal to 1.

Two-Variable K-map Representation



Problems

1. Map the Expression $\bar{A}\bar{B} + \bar{A}B + AB$ using K-map & reduce the function

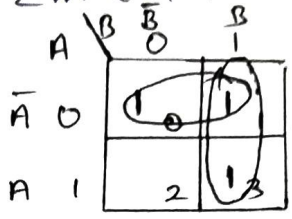
Solution

This Expression can be expressed as Sum of minterms as follows.

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

$= 00 + 01 + 11$

$F = \Sigma m(0,1,3)$ → Sum of minterms

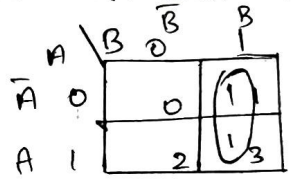


$F = \bar{A} + B$ Ans

2) Simplify the boolean function $f = \Sigma m(1,3)$ using K-map.

Solution

Given: $\Sigma m(1,3)$ → Sum of minterms



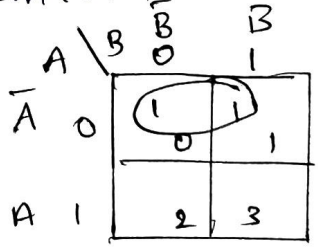
$F = B$ Ans

3) Using K-map obtain minimal Sop for the given truth table.

A	B	F
0	0	1
0	1	1
1	0	0
1	1	0

Solution

From the truth table, the minterms are identified as, $\Sigma m(0,1)$



$F = \bar{A}$ Ans

Three-Variable K-map

In a 3 Variable K-map, there are 8 Squares (2^3), one for each minterm

Here

-> One Square represents a minterm with three literals

-> Two adjacent Squares represent a term with two literals.

-> Four adjacent Squares represent a term with one literal

-> Eight adjacent Squares Produce a function that is always equal to 1

Three-Variable K-map Representation

A \ BC	00	01	11	10
0	m_0 0	m_1 1	m_3 3	m_2 2
1	m_4 4	m_5 5	m_7 7	m_6 6

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	$\bar{A}\bar{B}\bar{C}$ 0	$\bar{A}\bar{B}C$ 1	$\bar{A}BC$ 3	$\bar{A}B\bar{C}$ 2
A 1	$A\bar{B}\bar{C}$ 4	$A\bar{B}C$ 5	ABC 7	$AB\bar{C}$ 6

Problems

(1) Simplify $F = \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$ using K-map

Solution

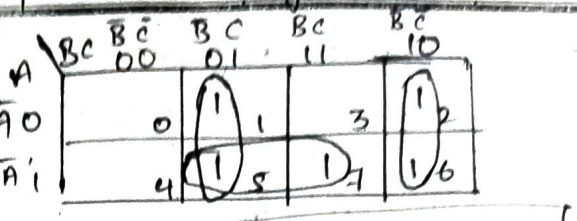
$F = \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$

$= 001 + 101 + 010 + 110 + 111$

$= \sum m(1, 5, 2, 6, 7)$

Re-arrange

$\sum m(1, 2, 5, 6, 7)$ -> Sum of minterms

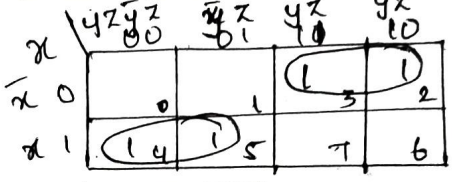


Simplified $F = \bar{B}C + B\bar{C} + AC$

(2) Simplify the Boolean function $F(x, y, z) = \sum(2, 3, 4, 5)$ using K-map

Solution

Given the minterms $\sum(2, 3, 4, 5)$



$F = x\bar{y} + \bar{x}y$

(3) Simplify the Boolean function

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

i) Express F as Sum of minterms

ii) Find minimal Sop Expression

Solution

i) Sum of minterms of F

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

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

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

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

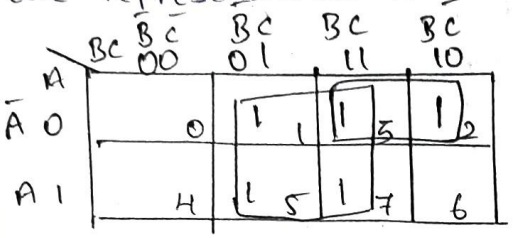
$= 011 + 001 + 010 + 101 + 111$

$= m_3 + m_1 + m_2 + m_5 + m_7$.

$F = \sum(1, 2, 3, 5, 7)$ -> Sum of minterms.

ii) Finding Minimal Sop Expression

* Using 3-Variable K-map The minterms are represented as '1'



$G_1(1,3,5,7) = C$, $G_2(2,3) = \bar{A}B$

Minimal Sop $= C + \bar{A}B$

Four-Variable K-map:

In a four-variable K-map, there are 16 Squares (2^4), one for each minterms

Here,

→ One Square represents one minterm giving a term with four literals

→ Two adjacent Squares represent a term with three literals

→ Four adjacent Squares represent a term with two literals

→ Eight adjacent Squares represent a term with one literal.

→ Sixteen adjacent Squares, represent a function that is always 1

Four Variable K-map Representation

m0	m1	m3	m2
m4	m5	m7	m6
m12	m13	m15	m14
m8	m9	m11	m10

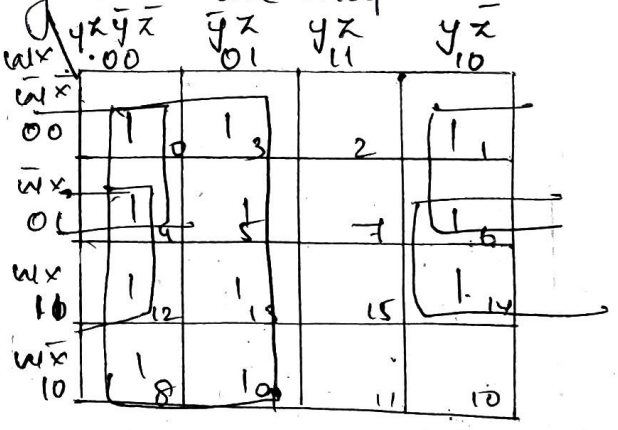
CD \ AB	C̄D̄ 00	C̄D 01	CD 11	CD̄ 10
ĀB̄ 00	ĀB̄C̄D̄ 0	ĀB̄C̄D 1	ĀB̄CD 3	ĀB̄CD̄ 2
ĀB 01	ĀBC̄D̄ 4	ĀBC̄D 5	ĀBCD 7	ĀBCD̄ 6
AB 11	ABC̄D̄ 12	ABC̄D 13	ABCD 15	ABCD̄ 14
AB̄ 10	AB̄C̄D̄ 8	AB̄C̄D 9	AB̄CD 11	AB̄CD̄ 10

Problems

① Simplify the Boolean function $F(w,x,y,z) = \Sigma\{0,1,2,4,5,6,8,9,12,13,14\}$ using K-map

Solution

Since the function has four variables, a four-variable map must be used. The minterm listed in the Sum are marked by 1's in the map



Group1 $(0,1,4,5,12,13,8,9) = \bar{y}$

Group2 $(0,4,2,6) = \bar{w}\bar{z}$

Group3 $(4,12,6,14) = x\bar{z}$

∴ Simplified function

$F = \bar{y} + \bar{w}\bar{z} + x\bar{z}$

② Simplify the function

$F = \bar{A}\bar{B}\bar{C} + \bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}$

using K-map

Solution

The given function F should

be expressed as Sum of min-terms.

$F = \bar{A}\bar{B}\bar{C} + \bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}$

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

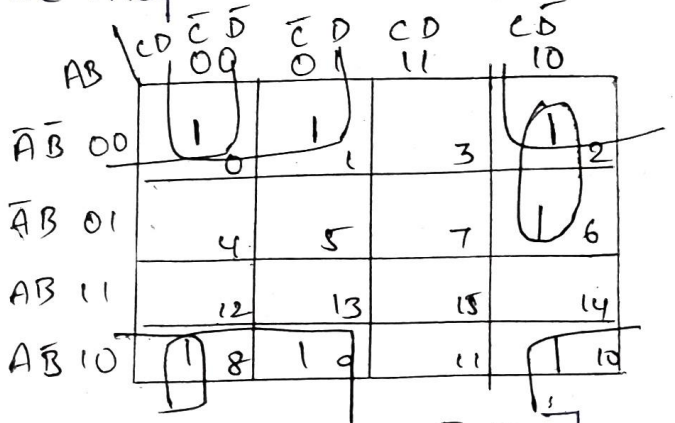
$= \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D}$,

$= 0001 + 0000 + 1010 + 0010 + 0110 + 1001 + 1000$

$= m_1 + m_0 + m_{10} + m_2 + m_6 + m_9 + m_8$

$F = \Sigma(0,1,2,6,8,9,10)$

The minterm listed in the Sum are marked by 1's in the map:



Group1 $(0,1,8,9) = \bar{B}\bar{C}$
Group2 $(2,6) = \bar{A}C\bar{D}$
Group3 $(0,2,8,10) = \bar{B}\bar{D}$

∴ Simplified function

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

(3) Plot the logical Expression $ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB$ on a 4-Variable map & reduce it.

Solution

Find the Sum of minterms for the logical expression

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

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

$= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + ABC + ABC(D+\bar{D})$

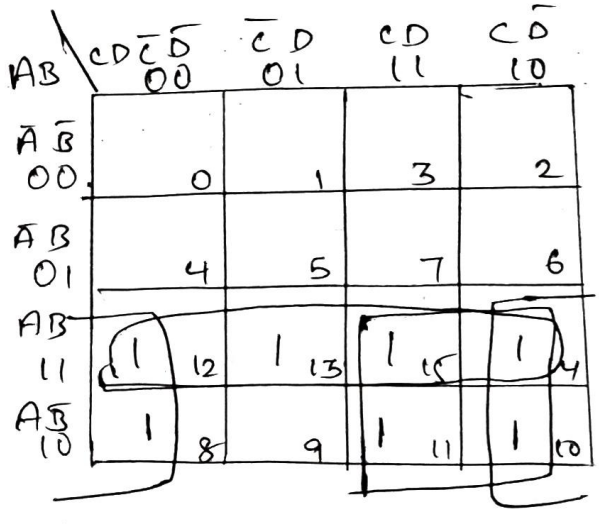
$= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD + ABC\bar{D} + AB\bar{C}D + ABC\bar{D}$

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

$= 1111 + 1001 + 1011 + 1010 + 1110 + 1101 + 1100$

$= m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12}$.

Sum of minterms $= \Sigma(8,10,11,12,13,14,15)$



Group1 $(12,13,14,15) = AB$
Group2 $(10,11,14,15) = AC$
Group3 $(8,10,12,14) = A\bar{D}$

∴ The reduced expression

$= AB + AC + A\bar{D}$

Five-Variable K-map

→ Five-Variable K-map contains Contains (2^5) 32 Squares, each Square contain One term

→ A five Variable K-map Consists of '2' four-Variable maps with Variables A, B, C, DE

→ The left hand 4-Variable map represents the 16 Squares containing minterms 0 to 15, in which A = 0

→ The right-hand 4-Variable map represents the 16 Squares containing minterms 16 to 32, in which A = 1

→ Each Square in the A = 0 map is adjacent to the Corresponding Square in the A = 1 map.

Five Variable map

→ One Square represents a minterm, giving a term with 5 literals

→ Two adjacent Squares represent a term with 4 literals

→ Four adjacent Squares represent a term with 3 literals

→ Eight adjacent represent a term with 2 literals

→ Sixteen adjacent Squares represent a term with 1 literal

→ 32 adjacent Squares represent a term which is always 1

BC \ DE	$\bar{D}\bar{E}$ 00	$\bar{D}E$ 01	DE 11	$D\bar{E}$ 10
$\bar{B}\bar{C}$ 00	0	1	3	2
$\bar{B}C$ 01	4	5	7	6
BC 11	12	13	15	14
$B\bar{C}$ 10	8	9	11	10

A = 0

BC \ DE	$\bar{D}\bar{E}$ 00	$\bar{D}E$ 01	DE 11	$D\bar{E}$ 10
$\bar{B}\bar{C}$ 00	16	17	19	18
$\bar{B}C$ 01	20	21	23	22
BC 11	28	29	31	30
$B\bar{C}$ 10	24	25	27	26

A = 1

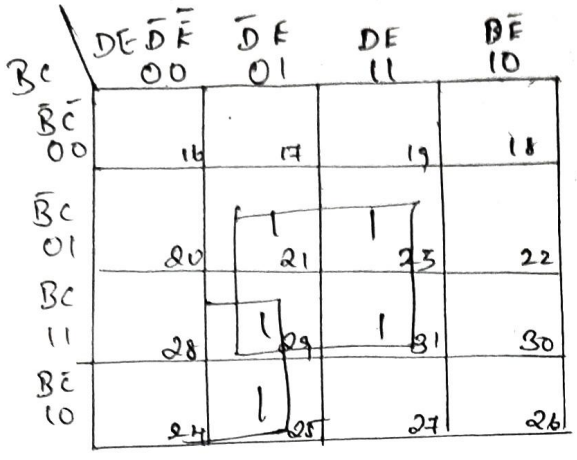
(1) Simplify the Boolean Function $F(A,B,C,D,E) = \Sigma(0,2,4,6,9,13,21,23,25,29,31)$

Solution

A = 0

BC \ DE	$\bar{D}\bar{E}$ 00	$\bar{D}E$ 01	DE 11	$D\bar{E}$ 10
$\bar{B}\bar{C}$ 00	1 0	1	3	1 2
$\bar{B}C$ 01	1 4	5	7	1 6
BC 11	12	1 13	15	14
$B\bar{C}$ 10	8	1 9	11	10

A = 0



Group 1 (21,23,29,31) = ACE
Group 2 (9,13,25,29) = B D̄ Ē
Group 3 (0,2,4,6) = Ā B̄ Ē

Simplified function

F = ACE + B D̄ Ē + Ā B̄ Ē

Prime Implicants

→ A Prime implicant is a Product term the can be obtained from the map by combining all possible maximum no. of adjacent Squares

→ The Prime implicant is called as essential Prime implicant if it is the "only Prime implicant" that covers the minterm.

Problems based on SOP & POS

1. SOP (Sum of Product)
2. POS (Product of Sum)

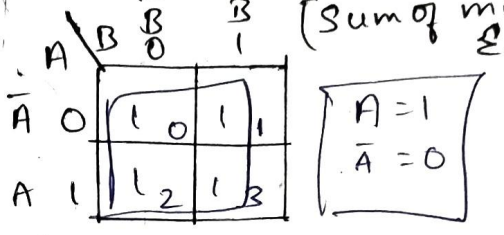
SOP (2-variable)

① using K-map obtain the minimal SOP for $\Sigma m(0,1,2,3)$

Solution Given SOP $\Sigma m(0,1,2,3)$

The given minterms are represented as '1' in the K-map

(Sum of minterm Σm)



minimal SOP = 1

These minterms can be represented as

Ā B̄ + Ā B + A B̄ + A B

= Ā (B̄ + B) + A (B̄ + B)

⇒ Ā + A = 1

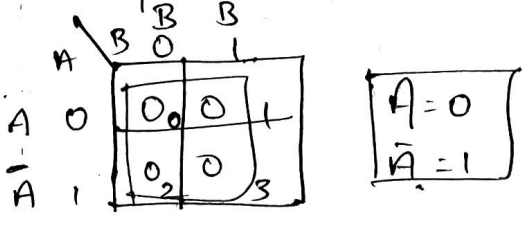
POS (2-variable)

① using K-map obtain the minimal POS for $\Pi m(0,1,2,3)$

Solution (Product of maxterms Πm)

Given POS $\Pi m(0,1,2,3)$.

The given maxterms are represented as '0' in the K-map.



minimal POS = 1

These maxterms can be represented as

A B + A B̄ + Ā B + Ā B̄

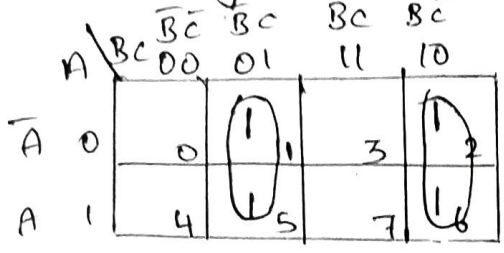
= A (B + B̄) + Ā (B + B̄) ⇒ A + Ā = 1

3-Variable (Sop & Pos)

(1) Obtain a) minimal Sop
b) minimal Pos for $F = \sum m(1,2,5,6)$ using k-map

Solution

a) finding minimal Sop:



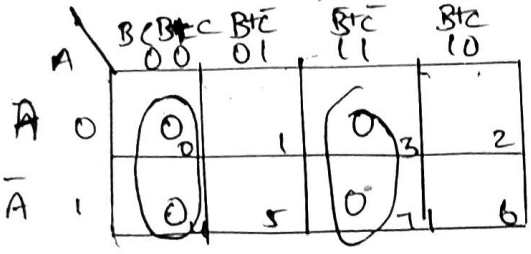
$[G_1(1,5) = \bar{B}C,\ G_2(2,6) = B\bar{C}]$

minimal Sop $= \bar{B}C + B\bar{C}$

b) Finding minimal Pos

Given minterms $= \sum m(1,2,5,6)$

The maxterms are identified as $\prod m(0,3,4,7)$



$G_1(0,4) = B + C,\ G_2(3,7) = \bar{B} + \bar{C}$

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

4-Variable (Sop & Pos).

(1) SOP

Minimize the following Boolean expression using K-map $y(A,B,C,D) = AB\bar{C} + BCD + B\bar{C}D$

Solution

The given boolean expression should be written as Sum of minterms

$y = AB\bar{C} + BCD + B\bar{C}D$

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

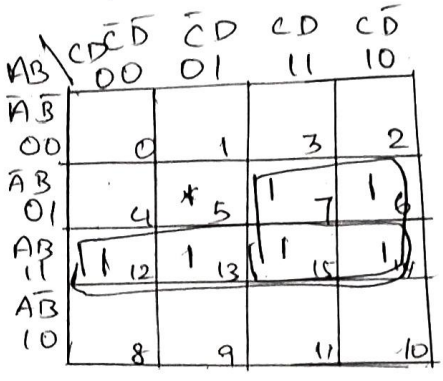
$= AB\bar{C}D + AB\bar{C}\bar{D} + ABCD + \bar{A}BCD + ABC\bar{D} + \bar{A}BC\bar{D}$

$= 1101 + 1100 + 1111 + 0111 + 1110 + 0110$

$= m_{13} + m_{12} + m_{15} + m_7 + m_{14} + m_6$

$y = \sum(6,7,12,13,14,15)$

Now, these minterms are represented as 1's on the K-map.



$[\text{Group1}(6,7,14,15) = BC]$
$[\text{Group2}(12,13,14,15) = A\bar{B}]$

Simplified function $y = A\bar{B} + BC$

Pos

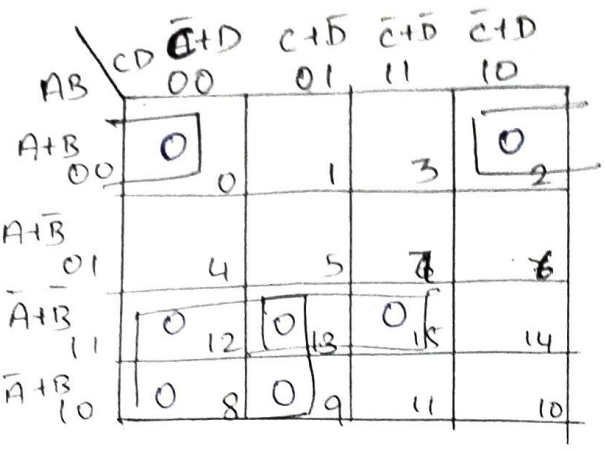
(2) Reduce the following expression using K-map technique.

$y = \prod m(0,2,8,9,12,13,15)$

Solution

Given the maxterms.

The maxterms are represented as '0' on the K-map.



Group 1 (8,9,12,13) = $\bar{A}+C$
Group 2 (13,15) = $\bar{A}+\bar{B}+\bar{D}$
Group 3 (0,2) = $A+B+D$

∴ The Simplified function

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

Analysis & Design Procedure for Combinational Circuit:

* Observe the Problem definition
* Determine the required input & output variables.
* Assign letters or Symbols to the input variables.
* Make truth table that defines required relationship
* Determine the Simplified Boolean Expression using K-map
* Draw logic diagram

Example (Problem)

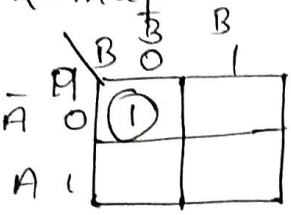
① Design a combinational circuit with two inputs which produce output as logic 1 when any one input is one.

input A B
Output y.

Truth table

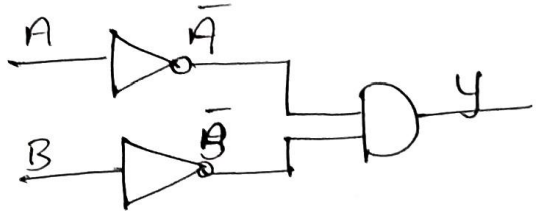
A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

K-map



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

Logic Diagram



Problem 2

① A majority gate is a digital circuit whose o/p is equal to 1 if the majority of i/ps are 1's otherwise the o/p will be 0. Using a by a 3-input majority gate. Simplify the function & implement with gates.

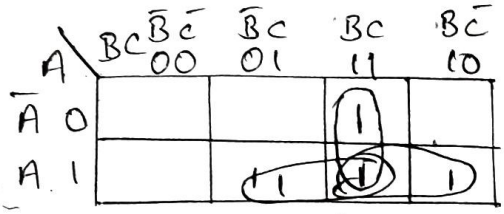
Solution

input A B C
Output y

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

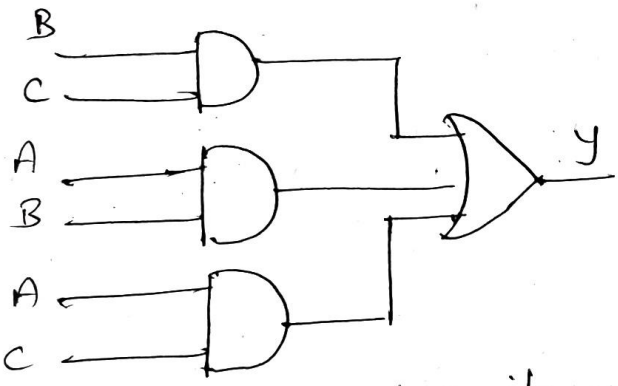
← Truth table

K-map Simplification

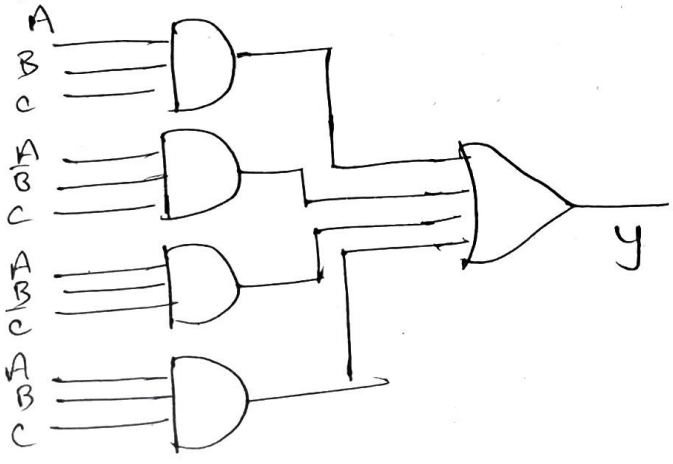


$Y = BC + AB + AC$

Implementation after Simplification



Implementation with 3-i/p gate



Design of Adder

Binary adder
→ Half & full adder
→ Parallel & carry adder

Digital computers perform various arithmetic operations. The most basic operation is the addition of two binary digits. The four possible elementary operations are,

$0+0=0$ $\quad 1+0=1$

$0+1=1$ $\quad 1+1=10_2$

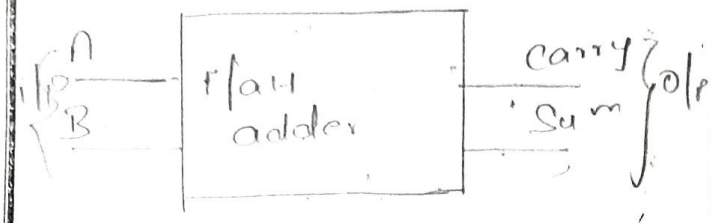
The higher significant bit of this result is called a carry. The logic circuit which performs this operation is called half-adder. The circuit which performs addition of three bits (2 significant bit & a previous carry) is a full adder.

The half adder operation needs two binary i/ps augend & addened bits & two binary o/p: Sum & carry

Truth table

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

Block Schematic of half adder



K-map Simplification for Carry & Sum

For Carry

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

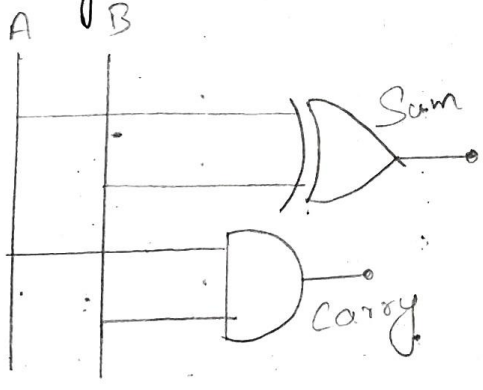
For Sum

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

Carry = AB

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

Logic Diagram



Limitations

→ In multidigit addition we have to add two bits along with the carry of Previous digit addition

→ Effectively Such addition requires addition of 3 bits.

→ This is not possible with half-adder

→ Hence half adders are not used in Practice

Full adder

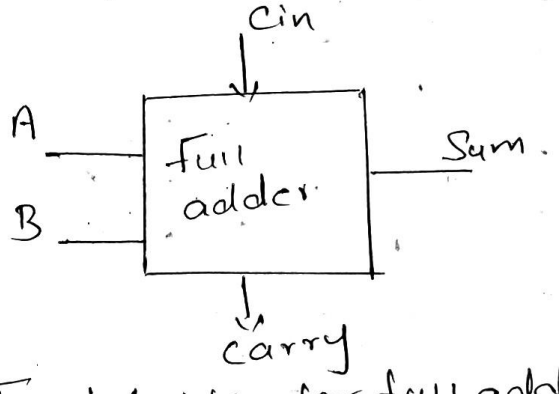
- A full adder is a combinational Circuit that forms the arithemetic Sum of three i/p bits.

→ It consists of three i/ps & two o/ps.

→ Two of the i/p variables, denoted by A & B, represent the two Significant bits to be added

→ The third i/p cin, represents the Carry from the Previous lower Significant Position

Block Schematic of full adder



Truth table for full adder.

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0 → 0
0	0	1	1	0 → 1
0	1	0	1	0 → 2
0	1	1	0	1 → 3
1	0	0	1	0 → 4
1	0	1	0	1 → 5
1	1	0	0	1 → 6
1	1	1	1	1 → 7

-For Carry

A \ BCin	$\bar{B}\bar{C}in$ 00	$\bar{B}C in$ 01	$BC in$ 11	$B\bar{C}in$ 10
$\bar{A}$ 0	0	0	1	0
A 1	0	1	1	1

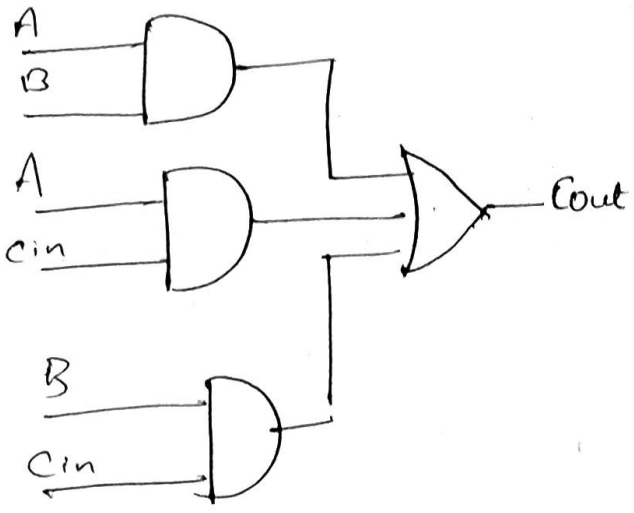
$Cout = AB + ACin + BCin$

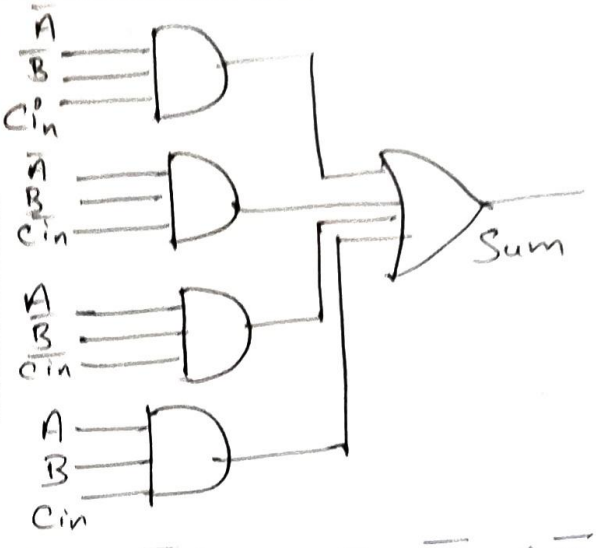
-For Sum

A \ BCin	$\bar{B}\bar{C}in$ 00	$\bar{B}Cin$ 01	$BCin$ 11	$B\bar{C}in$ 10
$\bar{A}$ 0	0	1	0	1
A 1	1	0	1	0

$Sum = \bar{A}\bar{B}Cin + \bar{A}B\bar{C}in + A\bar{B}\bar{C}in + ABCin$ (or) $S = A \oplus B \oplus Cin$

Logic Diagram





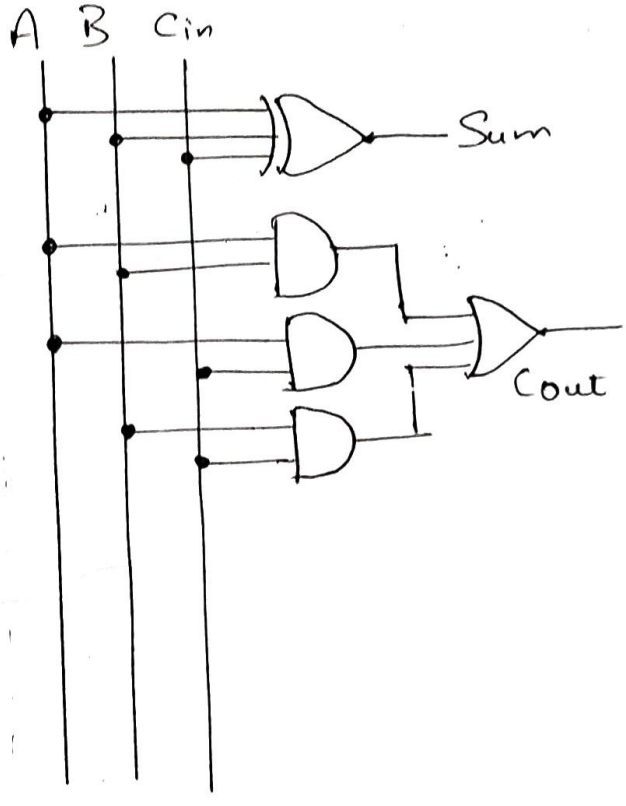
$Sum = \bar{A}\bar{B}Cin + \bar{A}B\bar{C}in + A\bar{B}\bar{C}in + ABCin$

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

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

$= \bar{A}(y \oplus z) + A(\bar{y \oplus z})$

$S = A \oplus B \oplus Cin$



Full adder using two half adders

→ A full adder can also be implemented with two half adders and one R-gate

→ The Sum output from the second half adder is the exclusive-OR of cin & the output of first half-adder giving

$$C_{out} = AB + Ac_{in} + Bc_{in}$$

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

$$= AB + ABc_{in} + A\bar{B}c_{in} + ABc_{in} + \bar{A}Bc_{in}$$

$$= AB + ABc_{in} + A\bar{B}c_{in} + \bar{A}Bc_{in}$$

$$= AB(1+c_{in}) + A\bar{B}c_{in} + \bar{A}Bc_{in}$$

$$= AB + A\bar{B}c_{in} + \bar{A}Bc_{in}$$ Formula

$$= AB + c_{in}(A\bar{B} + \bar{A}B) \quad [A\bar{B} + \bar{A}B = A \oplus B]$$

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

[Important

Binary adder

→ Half adder

→ Full adder

→ Parallel adder / Ripple carry

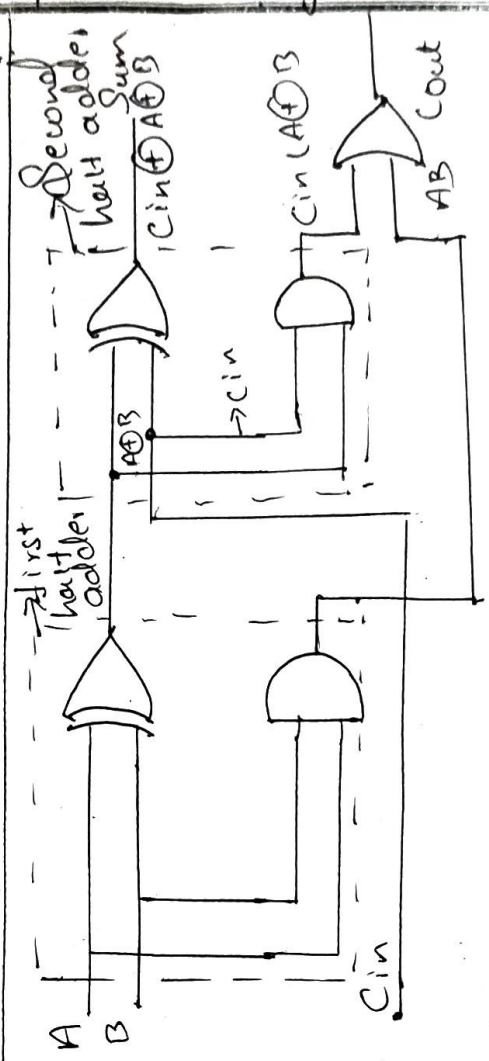
→ 100% carry ahead adder

Binary Subractor

→ Half Subractor

→ Full Subractor

→ Parallel Subractor



Parallel adder / Ripple Carry adder

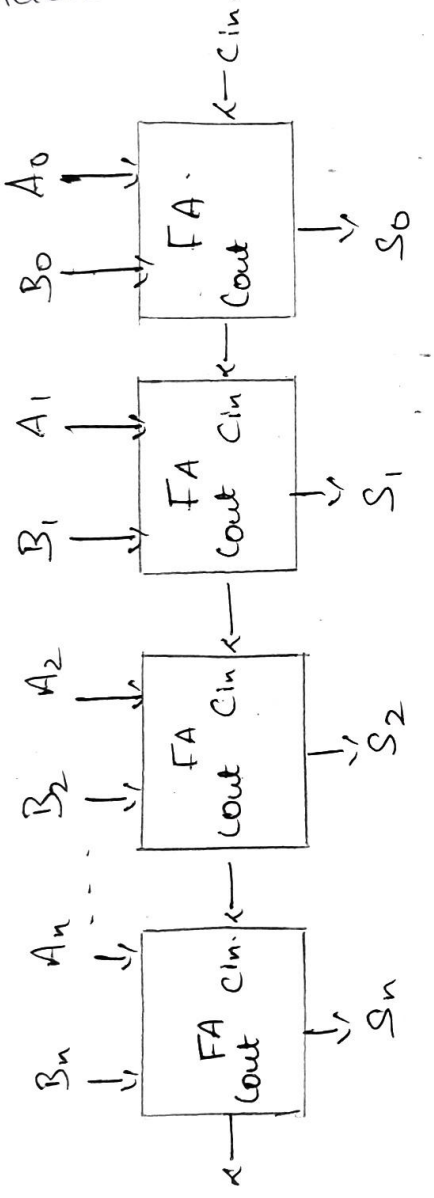
In Serial adder, a full adder is capable of adding two one-bit number & an input carry. If the number of bit increases with bit by bit, addition is the time taking Process. In order to increase the Speed of Operation we Prefer Parallel adder

Construction & Operation of n-bit Parallel Adder

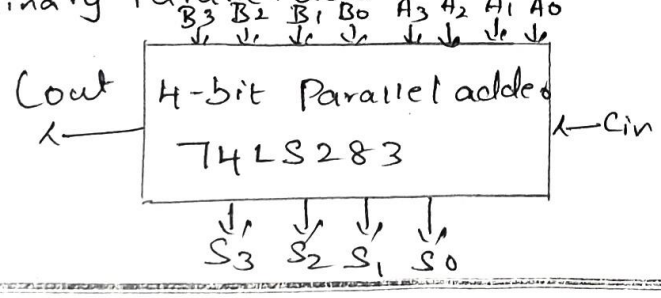
→ If the no. of bit increases in the binary number we can employ additional full adder.

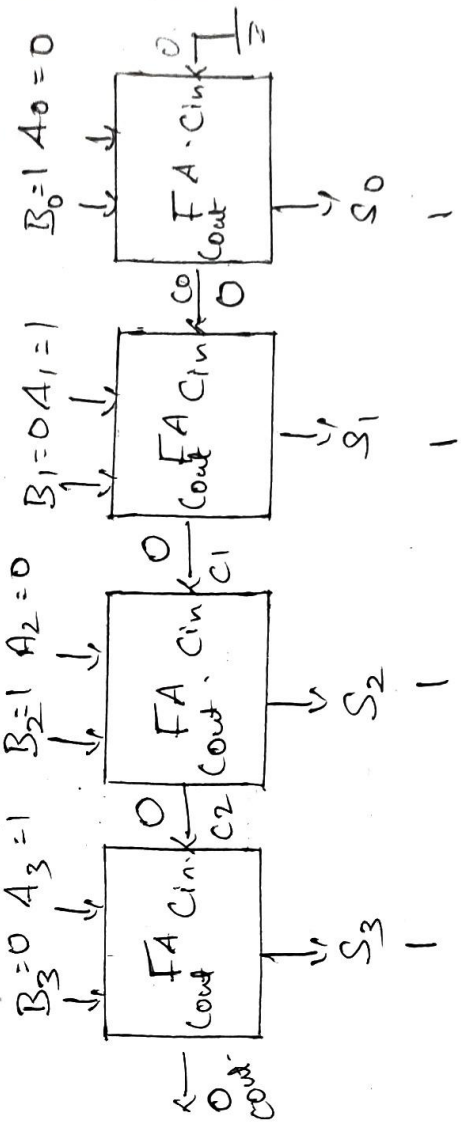
→ A n-bit parallel adder can be constructed by using 'n' no. of full adders connected in parallel.

→ The carry o/p of each adder is connected to the carry i/p of the next higher order adder, hence the name "Ripple carry Adder"



Binary Parallel adder IC 74LS283





Sum = A ⊕ B ⊕ C [inequality Detector]

Carry = AB + BC + CA

MSB LSB

Ex: 0 A3 1 A2 0 A1 1 A0

0 B3 1 B2 1 B1 0 B0

1 1 1 1

Look Ahead Carry

→ In Parallel adder, the Carry o/p of each full adder Stage is connected to the carry i/p of the next higher Order Stage.

→ Therefore, the Sum & carry

Outputs of any stage cannot be produced until the i/p carry occurs. This leads to a time delay in the addition process. This delay is known as "Carry Propagation delay"

Eg:
```
  0101
  0011
 ------
  1000
```

Addition of LSB produces carry into 2nd & 2nd to 3rd. From this the sum bit generated in the last position (MSB), depends on the carry that was generated by the addition of the previous position

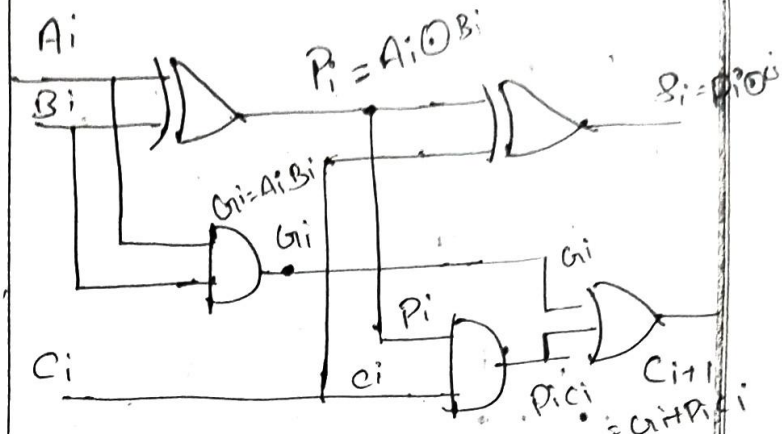
→ This means adder will not produce correct result until LSB carry has propagate through intermediate full adders.

→ So because of this time delay we can't add more no. of bits.

→ One method of speeding up this process by eliminating inter stage carry delay is called look ahead carry addition.

Implement full adder using 2 half adder

The output sum & carry can be expressed as



2n gate level → Carry Propogate

n → no of bits

For, 4-bit adder

Carry Propogate = 2 × 4 = 8 gate levels

P_i = Carry Propogate,

G_i = Carry generator

$P_i = A_i + B_i$ $\quad S_i = P_i \oplus C_i$

$G_i = A_i \cdot B_i$ $\quad C_{i+1} = G_i + P_i C_i$

C_0 = input carry

$i=0 : C_{0+1} = C_1 = G_0 + P_0 C_0$

$i=1, C_2 = G_1 + P_1 C_1 \Rightarrow G_1 + P_1 (G_0 + P_0 C_0)$

$= G_1 + P_1 G_0 + P_1 P_0 C_0$

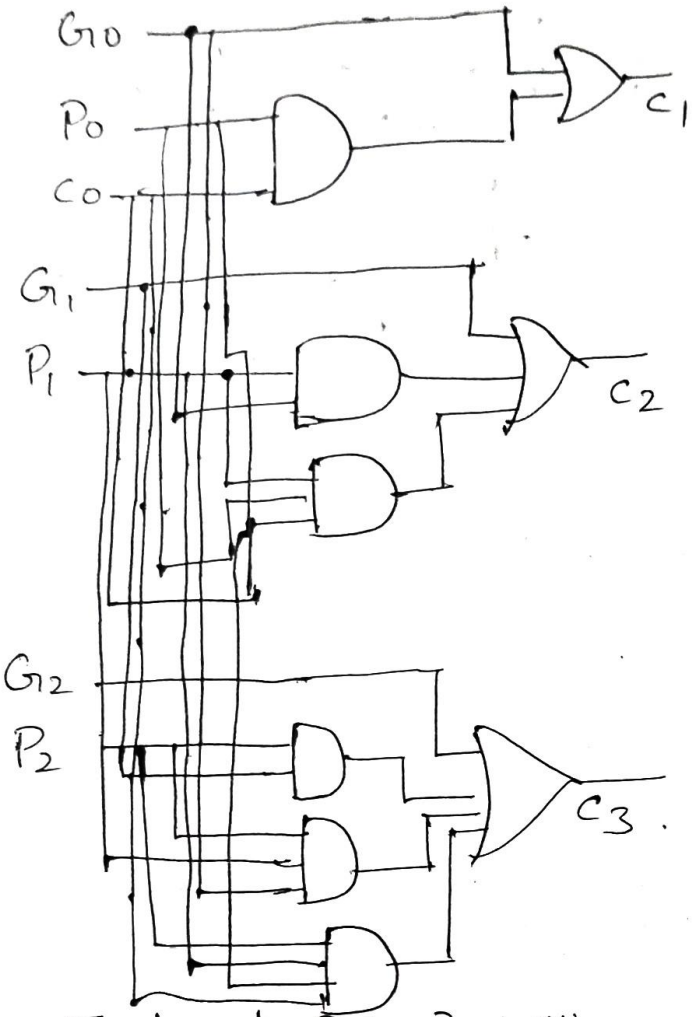
$i=2, C_3 = G_2 + P_2 C_2 \Rightarrow G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$

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

$i=3, C_4 = G_3 + P_3 C_3$

$= G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$



For clear dig: Refer Pg.No.44

-> Using a look ahead carry generator we can easily construct a 4-bit Parallel adder with a look ahead carry Scheme

-> In this each Sum o/p requires 2 EXOR gates.

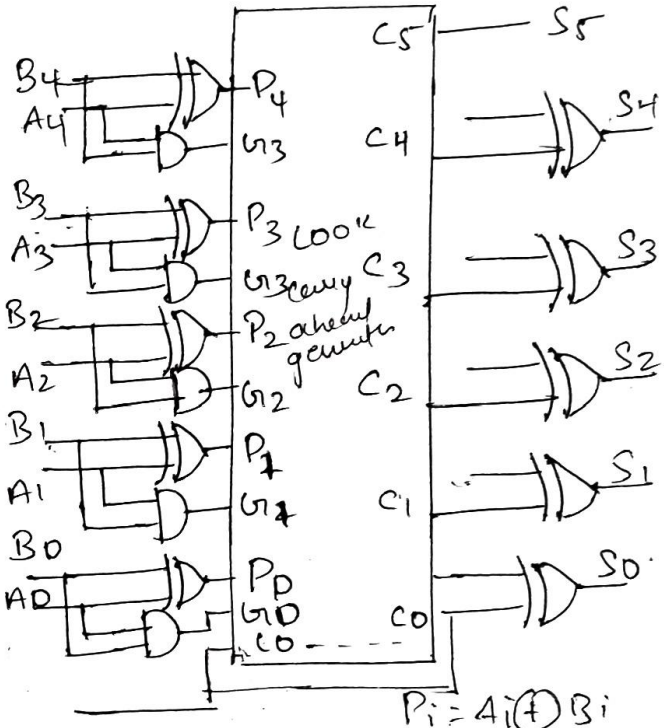
-> The o/p of the 1st XOR gate generates P_i & the AND gate generates G_i

-> The Carries are generated using look ahead carry generator & applied as i/p's to the Second EXOR gate & the i/p to the Ex-OR gate is P_i

-> Thus Second Ex-OR gates generates Sum o/p's

-> Each o/p generated after a delay of 2 levels of gates

-> Thus o/p's S_2 through S_4 have Equal Propagation delay time.



$P_i = A_i \oplus B_i$

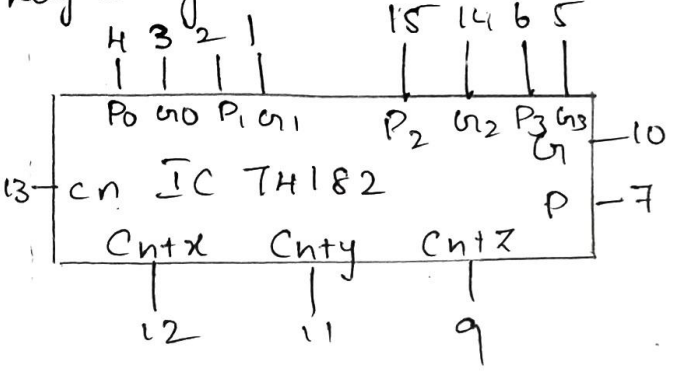
$G_i = A_i B_i$

$S_i = P_i \oplus C_i$

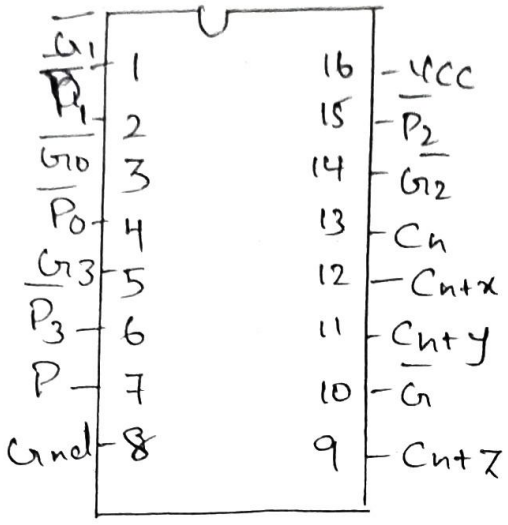
$C_{i+1} = G_i + P_i C_i$

IC 74182 - look ahead carry generator

logic Symbol



Pin Diagram



$(\overline{P_0}, \overline{P_1}, \overline{P_2}, \overline{P_3})$ → four Pairs of active low Propagate

$(\overline{G_0}, \overline{G_1}, \overline{G_2}, \overline{G_3})$ → Carry generate Signals, C_n → active high carry i/p

C_{n+x}, C_{n+y}, C_{n+z} → Anticipated active high Carries

$\overline{P}$ - active low Carry Propagate

$\overline{G}$ - Carry generate o/ps

Binary Subtracter

→ Half Subtracter

→ full Subtracter

→ Parallel Subtracter

Half Subtracter

→ Subtracthend bit is Subtracted from the minuend bit.

→ Minuend bit is Smaller than the Subtrahend bit, hence 1 is borrowed.

0 - 0 = 0

0 - 1 = 1 (with 1 borrow)

1 - 0 = 1

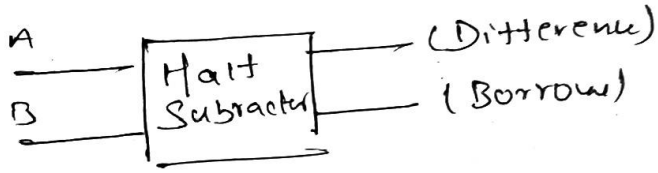
1 - 1 = 0

Half Subtracter

→ A half Subtracter is a combinational circuit ~~that~~ used to get the difference between two Single bit numbers

→ It also has an o/p to Specify if a 1 has been borrowed.

→ Let 'A' be minuend bit & 'B' be Subtractend bit.

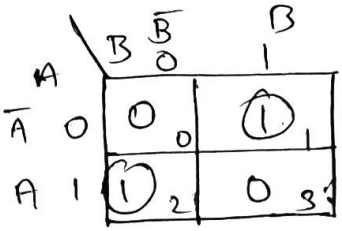


Truth table

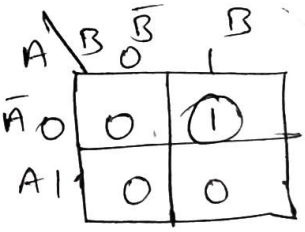
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-map

For Difference



For Borrow

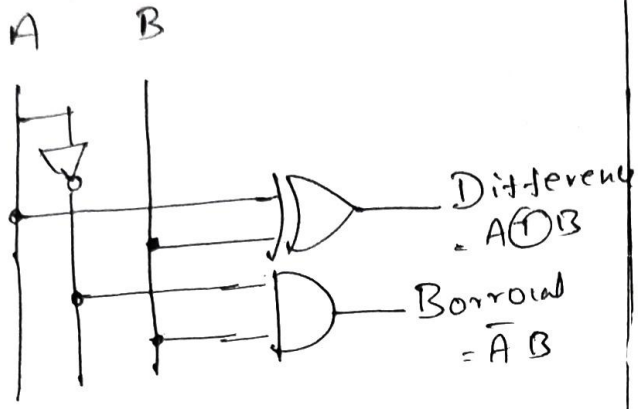


Difference $= A\overline{B} + \overline{A}B$

$= A \oplus B$

Borrow $= \overline{A}B$

Logic diagram



Limitations

-> In multidigit Subtraction, we have to Subtract two bits along with the borrow of the previous digit Subtraction

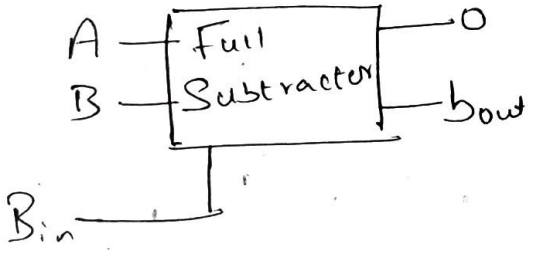
-> Effectively Such Subtraction requires Subtraction of three bits

-> This is not possible with half-Subtracter.

Full Subtracter

Full Subtractor is combinational Circuit used to perform Subtraction among 3-bits.

~~Truth~~ table.



It has 3 i/p & 2 o/ps

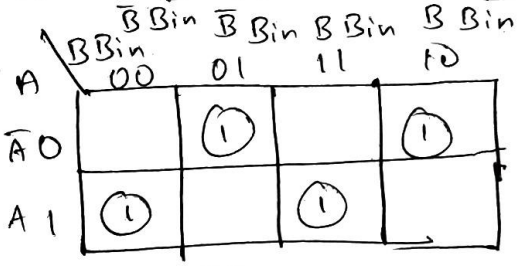
i/p -> A, B & Bin

o/p -> D & Bout

Truth Table

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

K-map for Difference



$D = A\bar{B}\bar{B}in + \bar{A}\bar{B}Bin + ABBin + \bar{A}B\bar{B}in$

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

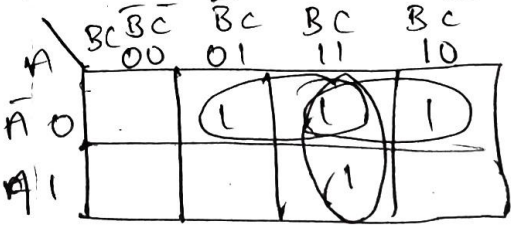
$= Bin(A \odot B) + \bar{B}in(A \oplus B)$

$D = Bin \oplus A \oplus B$

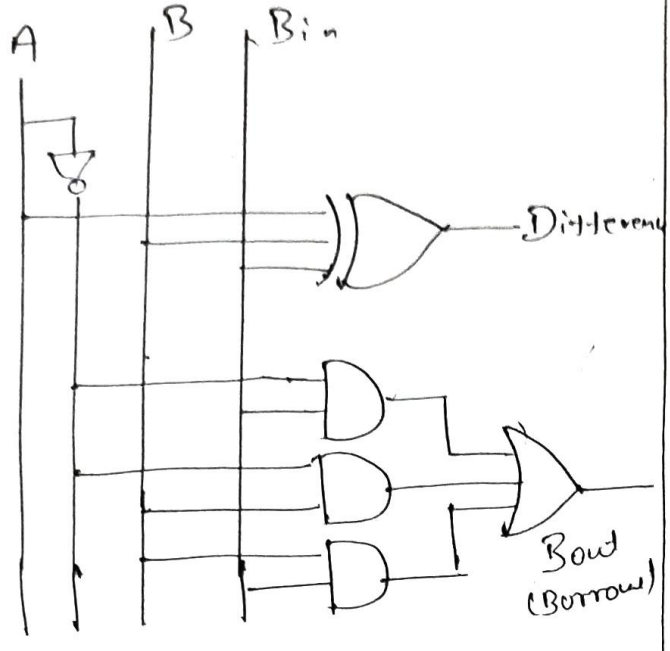
Re-arrange

$A \oplus B \oplus Bin = D$

K-map for Bout



$Bout = \bar{A}Bin + \bar{A}B + BBin$



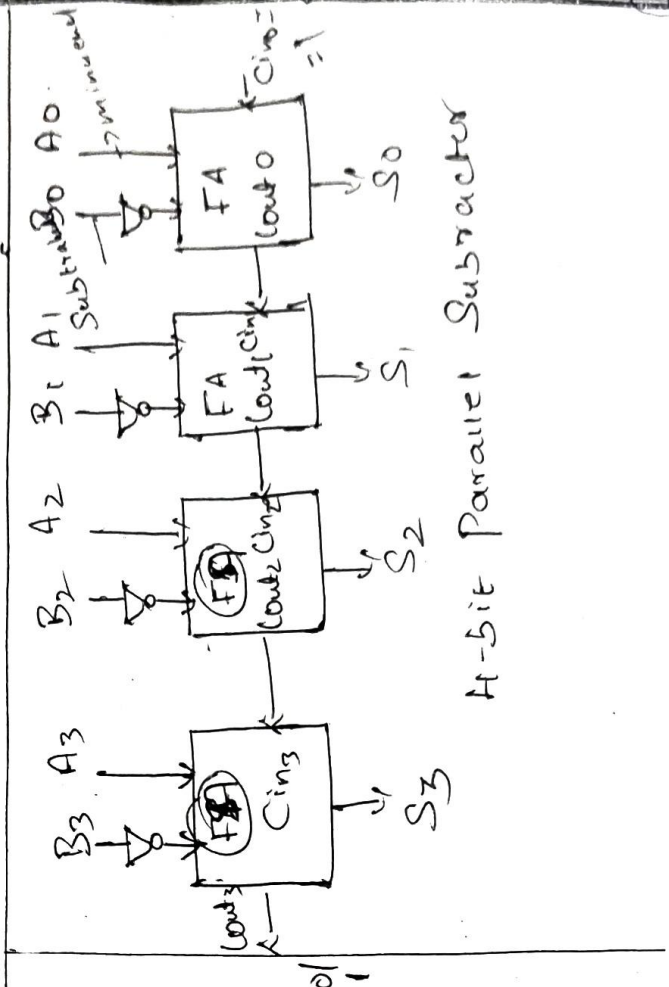
Parallel Subtractor

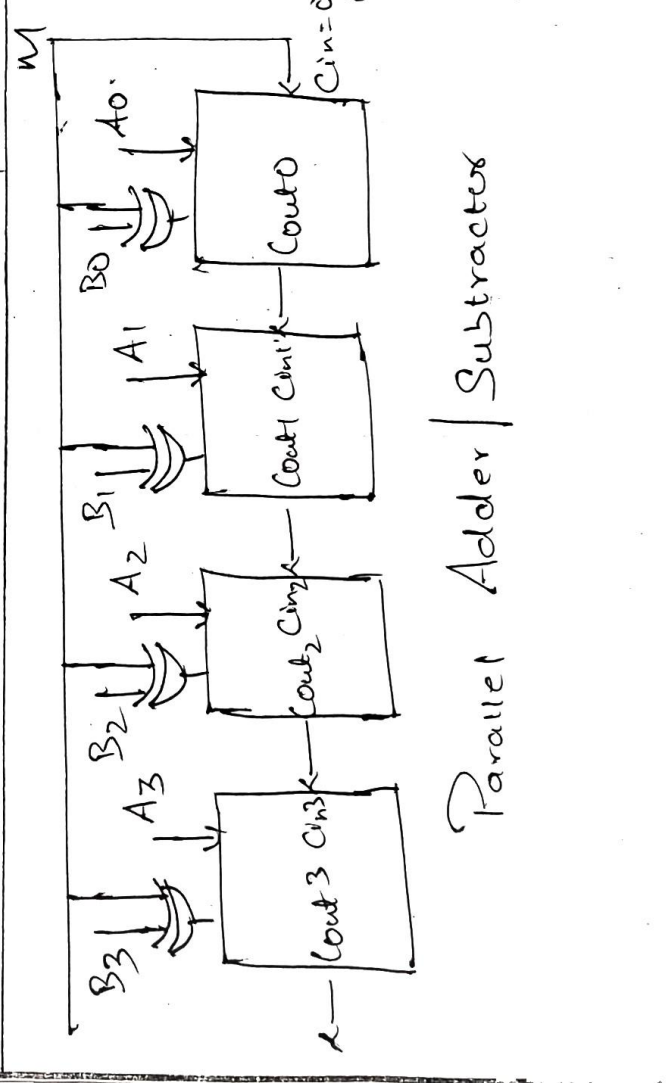
→ The Subtraction of binary numbers can be done most conveniently by means of complements

→ Subtraction of A-B can be done by taking 2's complement of B & adding it to 'A'

→ The 2's complement can be obtained by taking the 1's complement & adding 1 to the LS pair of bits.

→ The 1's complement can be implemented with inverters & one can be added to the Sum through i/p carry.





Construction

→ The addition & Subtraction operations can be combined into one circuit with one common binary adder. This is done by including an Ex-OR gate with each full adder. It has mode selection switch 'M' to control the operation.

Operation

The Mode 'M' i/p controls the operation of the ckt.

→ M=0 → Ckt adds as adder
M=1 → Ckt becomes Subtractor

→ Each Ex-OR gate receives i/p M & one of the i/ps of B.

→ When M=0, we have $B \oplus 0 = B$. The full adders receives the values of B, the i/p carry is 0 & Circuit Performs A+B Operation

→ When M=1, we have $B \oplus 1 = \bar{B}$ & Cin 0 = 1. The B i/ps are all implemented & a 1 is added through the i/p carry. The circuit performs the operation A plus the 2's complement of B, (ie) A-B

[Note:
M = 0, Sum = A+B+Cin
= A+B (Add)

M = 1
= A + (1's complement of B) + Cin
= A + 1's comp B + 1
= A + 2's comp B
= A - B (Subtract)

Whenever,
→ X-OR gate with 0 gives same i/p
→ X-OR gate with 1 gives complement of o/p]

Decimal Adder / BCD adder

→ The digital systems handle the decimal number in the form of binary coded decimal numbers (BCD).

→ A BCD adder is a circuit that adds 2 BCD digits & Produces a Sum digit also in BCD.

→ BCD numbers uses 10 digits, 0 to 9 which are represented in binary form 0000 to 1001, (ie) each BCD digit is represented as a 4-bit binary number.

BCD addition Procedure:-

1) Add 2 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 on if a Carry is generated form a bit Sum, the Sum is invalid.

4) To correct the invalid Sum, add $(0110)_2$ to the 4-bit Sum, if a Carry results from this addition, add it to the next higher order BCD digit.

To implement BCD adder we require

→ 4-bit adder for inital addition

→ logic Circuit to detect Sum greater than 9.

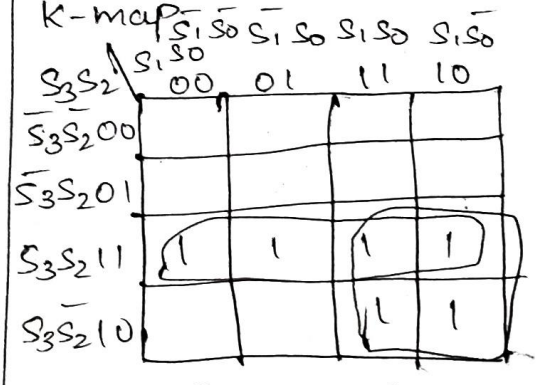
→ one more 4 bit adder to add $(0110)_2$ in the Sum if the Sum is greater than 9 or Carry is 1.

The logic Circuit to detect Sum greater than 9 can be determined by Simplifying the boolean expression of given truth table.

Truth table

Inputs				Output
S_3	S_2	S_1	S_0	y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
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

K-map

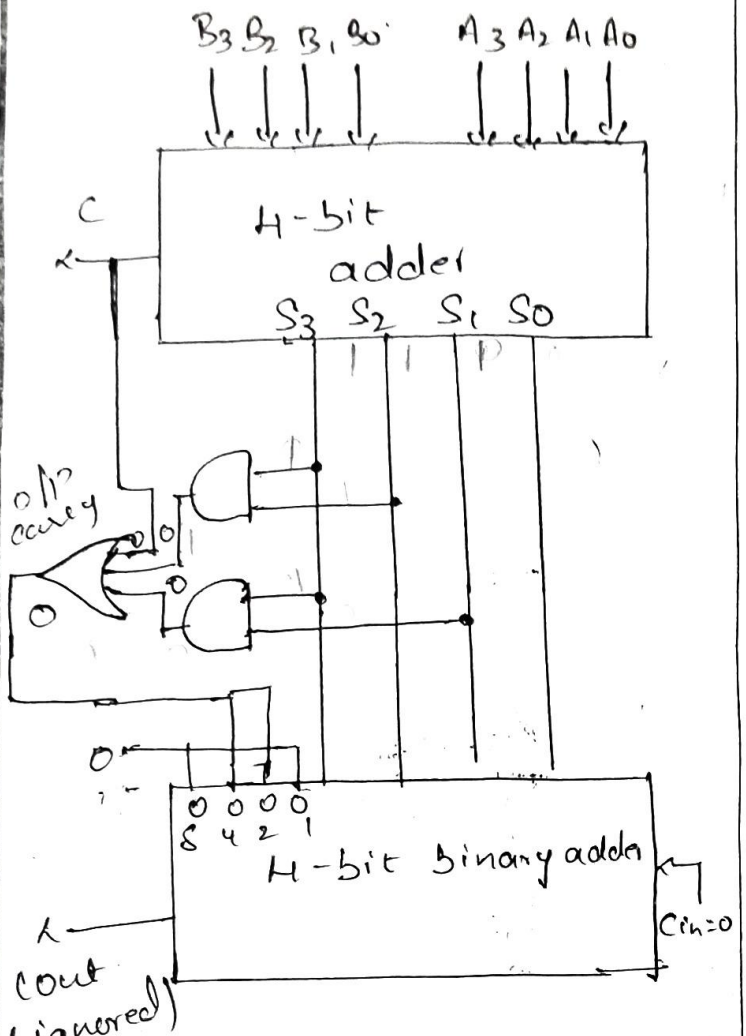


$$y = S_3S_2 + S_3S_1$$

y=1 → indicates Sum is greater than 9. We can add one more term Cout in the above Expression

$$y = S_3S_2 + S_3S_1 + Cout$$

to check whether the carry is 1 If any one condition is Satisfied, we add 6 $(0110)_2$ in the Sum.



Block diagram of BCD Adder

→ The 2 BCD no's together with i/p carry are first added in the top 4 bit binary adder to produce binary sum.

→ when the o/p carry is equal to zero (ie, when Sum≤9 (cout=0) nothing is added to binary sum. when it is equal to one (ie, when Sum>9 on cout=1) binary 0110 is added to the binary sum through the bottom 4-bit binary adder.

→ The o/p carry generated from the bottom binary adder can be ignored, since it supplies information already available at the output carry terminal

Ex:

(i) A : 0011 03
B : 1000 08
S 1011 11 1011>1001

(ii) If Sum>1001 ; x=1

(iii) 1011 - 11
0110 +6 => 0001 0001
10001

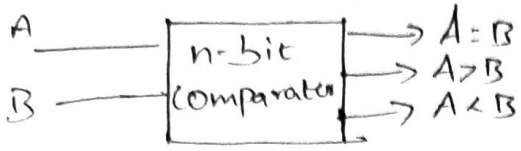
Valid BCD o/p

Applications of BCD Adder

→ It is widely used in the computers & calculators
→ used in Arithmetic logic unit
→ it accepts the binary coded form of decimal numbers.

Magnitude Comparator

It is a combinational circuit, designed to compare the relative magnitude of two binary number (A & B) & generates one of the following o/p.



Types

2-bit magnitude comparator

4-bit magnitude comparator

No of enteries in truth table $= 2^{2^n}$

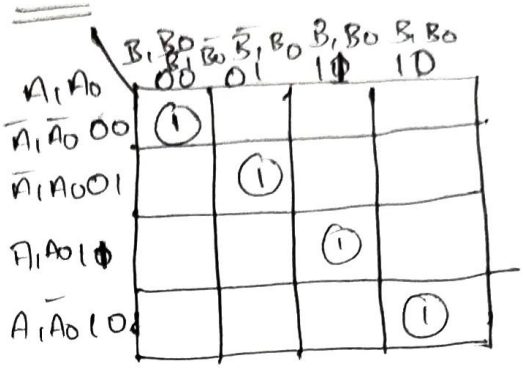
$n=2 = 2^{2^2} = 2^4 = 16$

$n=4 = 2^{2^4} = 2^{16} = 65,536$

Truth table

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

A=B



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

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

$= \bar{A}_0\bar{B}_0[\bar{A}_1\bar{B}_1 + A_1B_1] + A_0B_0[\bar{A}_1\bar{B}_1 + A_1B_1] + \quad \overline{A_1\bar{B}_1 + A_1B_1} = A \odot B$ EX-NOR

$= [\bar{A}_1\bar{B}_1 + A_1B_1][\bar{A}_0\bar{B}_0 + A_0B_0]$

This is represented in the form of x-NOR gate.

x_1 or x_0

$A=B = x_1 \cdot x_0$

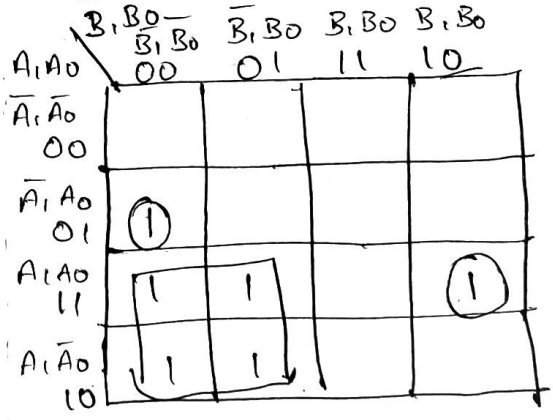
$x_1 = \bar{A}_1\bar{B}_1 + A_1B_1$ $\quad A = A_1A_0 \; 11$

$x_0 = \bar{A}_0\bar{B}_0 + A_0B_0$ $\quad B = B_1B_0 \; 11$

$A = B \cdot 1 \cdot 1 = 1$

whenever $A_1 \& B_1$, $A_0 \& B_0$ have to be equal, then only A=B.

A>B



$A>B = A_1\bar{B}_1 + \bar{A}_1A_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0B_1\bar{B}_0$

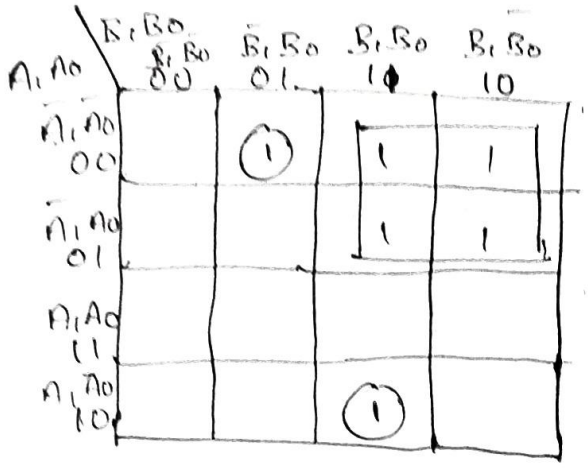
$A_1\bar{B}_1 + A_0B_0[\bar{A}_1\bar{B}_0 + A_1B_1]$

$A_1\bar{B}_1 + A_0\bar{B}_0 \cdot x_1$

$A_1\ A_0$ Here, $A_1\ A_0$ Level < B
$B_1\ B_0$ $B_1\ B_0$

In this equation itself we are getting A is greater & B is lesser

$A < B$



$(A<B) = \bar{A}_1\bar{A}_0\bar{B}_1B_0 + A_1\bar{A}_0B_1B_0 + \bar{A}_1B_1$

$= \bar{A}_1B_1 + \bar{A}_1\bar{A}_0\bar{B}_1B_0 + A_1\bar{A}_0B_1B_0$

$= \bar{A}_1B_1 + \bar{A}_0B_0[\bar{A}_1\bar{B}_1 + A_1B_1]$

$= \bar{A}_1B_1 + \bar{A}_0B_0 \cdot x_1$

$A_1\ A_0 \Rightarrow \bar{A}_1\ \bar{A}_0$ Here A > B
$B_1\ B_0$ $B_1\ B_0$

→ In this Expression itself we are Obtaining A is lesser & B is greater

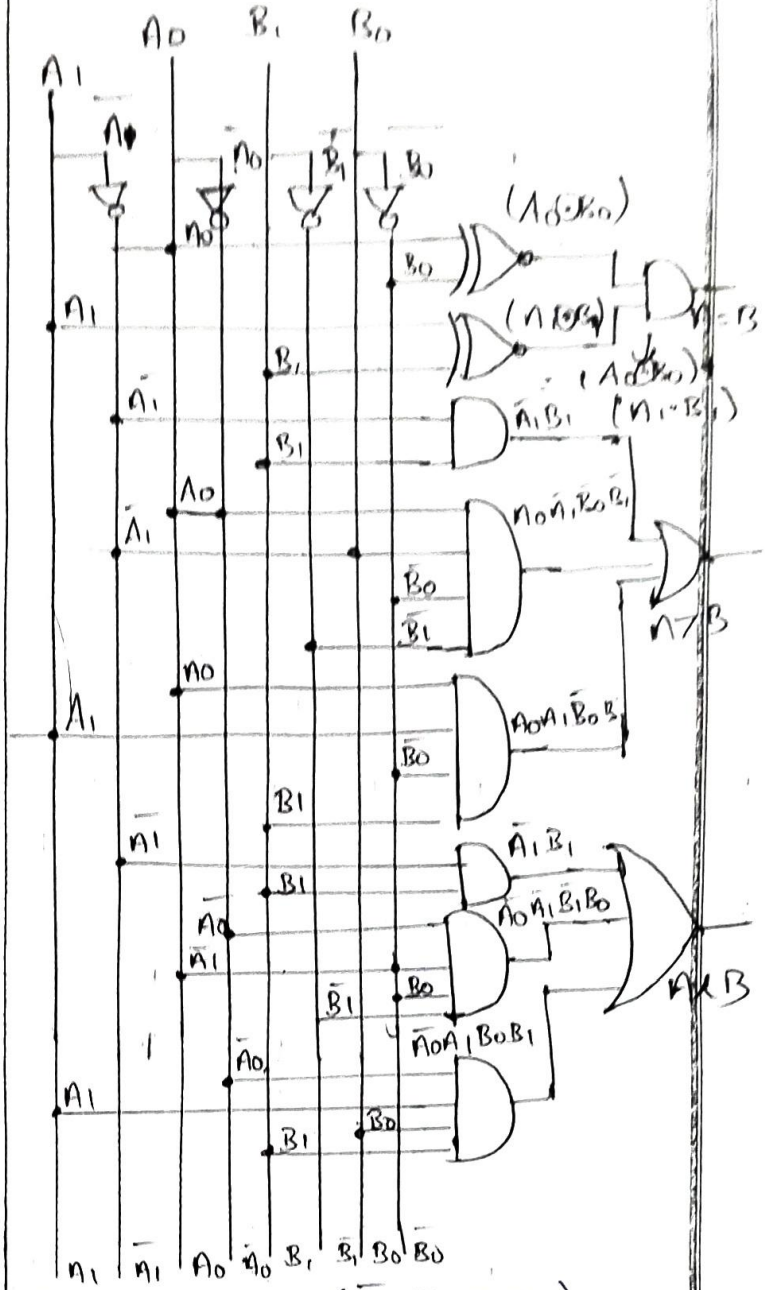
The obtained Equation is,

((r) Expression

$(A=B) = x_1 \cdot x_0$ (or) $[\bar{A}_1\bar{B}_1 + A_1B_1][\bar{A}_0\bar{B}_0 + A_0B_0]$ / $(A\odot B)(A\odot B)$

$(A>B) = A_1\bar{B}_1 + A_0B_0x_1$

$(A<B) = \bar{A}_1B_1 + \bar{A}_0B_0x_1$



$A=B$ $x_1 \cdot x_0$ (or) $(\bar{A}_1\bar{B}_1 + A_1B_1)$ $(\bar{A}_0\bar{B}_0 + A_0B_0)$ (or) $(A_0\odot B_0)(A_1\odot B_1)$

$A>B = A_1\bar{B}_1 + A_0\bar{B}_0x_1$

$= A_1\bar{B}_1 + A_0\bar{B}_0(\bar{A}_1\bar{B}_1 + A_1B_1)$

$= A_1\bar{B}_1 + A_0\bar{B}_0\bar{A}_1\bar{B}_1 + A_0\bar{B}_0A_1B_1$

$A>B = A_1\bar{B}_1 + A_0\bar{A}_1\bar{B}_0\bar{B}_1 + A_0A_1\bar{B}_0B_1$

$A<B = \bar{A}_1B_1 + \bar{A}_0B_0x_1$

$= \bar{A}_1B_1 + \bar{A}_0B_0x_1$

$= \bar{A}_1B_1 + \bar{A}_0B_0(\bar{A}_1\bar{B}_1 + A_1B_1)$

$= \bar{A}_1B_1 + \bar{A}_0B_0\bar{A}_1\bar{B}_1 + \bar{A}_0B_0A_1B_1$

$A<B = \bar{A}_1B_1 + \bar{A}_0B_0\bar{A}_1\bar{B}_1 + \bar{A}_0B_0A_1B_1$

$A<B = \bar{A}_1B_1 + \bar{A}_0A_1B_0\bar{B}_1 + \bar{A}_0A_1B_0B_1$

4-bit comparator

To find A=B

A => A_3 A_2 A_1 A_0

B => B_3 B_2 B_1 B_0

x_3 x_2 x_1 x_0

$x_3 = A_3B_3 + \bar{A}_3\bar{B}_3 = (A_3 \odot B_3)$

$x_2 = A_2B_2 + \bar{A}_2\bar{B}_2 = (A_2 \odot B_2)$

$x_1 = A_1B_1 + \bar{A}_1\bar{B}_1 = (A_1 \odot B_1)$

$x_0 = A_0B_0 + \bar{A}_0\bar{B}_0 = (A_0 \odot B_0)$

For $f(A=B) = x_3 \cdot x_2 \cdot x_1 \cdot x_0$

To find (A>B) A is greater than B

$(A>B) = A_3\bar{B}_3 + A_2\bar{B}_2x_3 + A_1\bar{B}_1x_3x_2 + A_0\bar{B}_0x_3x_2x_1$

To find A is lesser than B.

$(A<B) = \bar{A}_3B_3 + A_2B_2x_3 + A_1B_1x_3x_2 + \bar{A}_0B_0x_3x_2x_1$

A>B

i) If $A_3 > B_3$ ($A_3=1, B_3=0$) then A > B. So it is represented as $A_3\bar{B}_3$

ii) If $A_3 = B_3$ & $A_2 > B_2$ ($A_2=1, B_2=0$) then A > B. It is represented as $A_2\bar{B}_2x_3$.

iii) If $A_3 = B_3$ & $A_2 = B_2$ & $A_1 > B_1$, ($A_1=1, B_1=0$), then A > B. It is represented as $A_1\bar{B}_1x_3x_2$

iv) If $A_3 = B_3$ & $A_2 = B_2$ & $A_1 = B_1$ & $A_0 > B_0$ ($A_0=1, B_0=0$), then A > B. It is represented as $A_1\bar{B}_1x_3x_2x_1$

$(A>B) = A_3\bar{B}_3 + A_2\bar{B}_2x_3 + A_1\bar{B}_1x_3x_2 + A_0\bar{B}_0x_3x_2x_1$

(A<B)

i) If $A_3 < B_3$ ($A_3=0; B_3=1$); then A<B. It is represented as $\bar{A}_3B_3$ $A_2 < B_2$

ii) If $A_3 = B_3$ $\therefore$ ($A_2=0; B_2=1$) then A<B. It is represented as $\bar{A}_2B_2x_3$

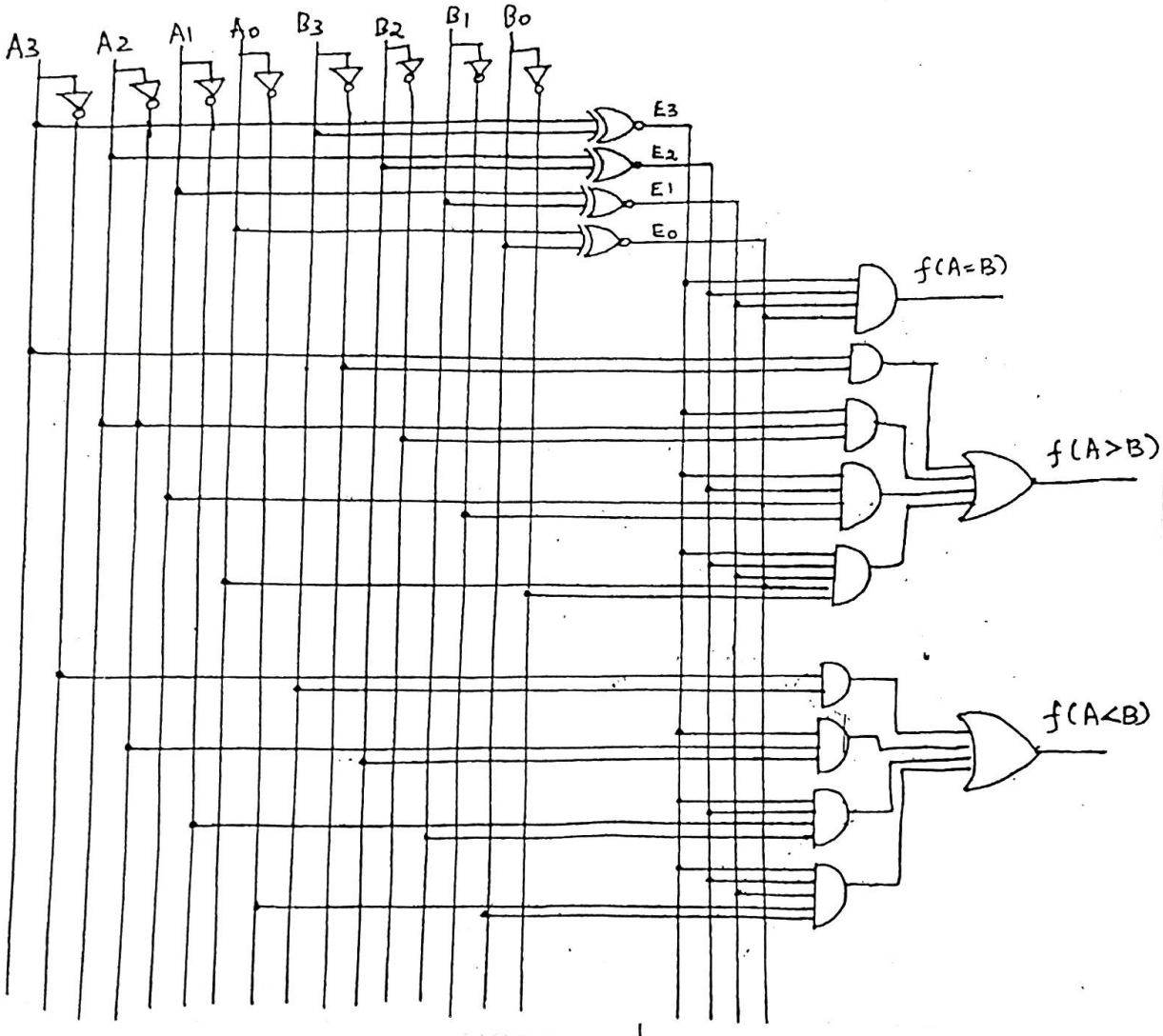
iii) If $A_3 = B_3$ & $A_2 = B_2$ & $A_1 < B_1$ ($A_1=0, B_1=1$) then A<B. It is represented as $\bar{A}_1B_1x_3x_2$

iv) If $A_3 = B_3$ & $A_2 = B_2$ & $A_1 = B_1$ & $A_0 < B_0$ ($A_0=0, B_0=1$) then A<B. It is represented as $\bar{A}_0B_0x_3x_2x_1$

Now the expression for A<B

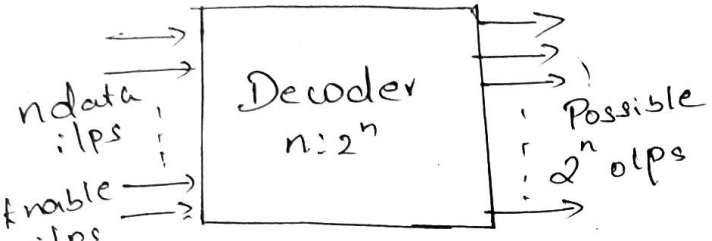
$(A<B) = \bar{A}_3B_3 + \bar{A}_2B_2x_3 + \bar{A}_1B_1x_3x_2 + \bar{A}_0B_0x_3x_2x_1$

Logic diagram for 4-bit Magnitude Comparator:



Decoder

Decoder is a multi-i/p multi-o/p logic circuit which decodes n no.of. i/ps to 2^n Possible o/ps.



↳ Enable → E=0 Decoder is disabled
→ E=1 Decoder is enabled

Binary decoder / 2 to 4 decoder

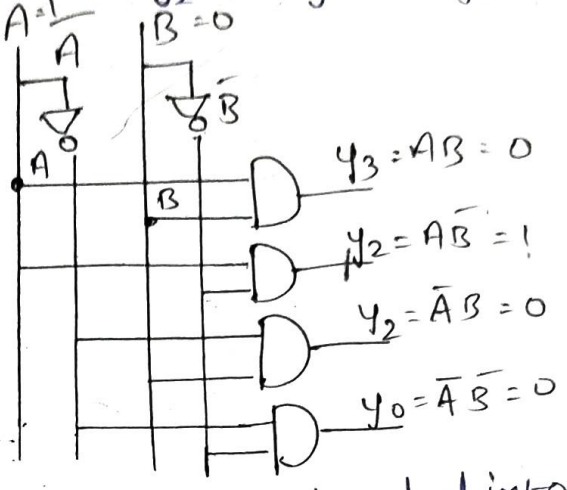
A decoder which has an n-bit binary i/p code & one activated o/p, Out of 2^n o/p code is called binary decoder.

A binary decoder is used when it is necessary to activate exactly one of the 2^n o/p based on an n-bit input value.

Truth table:

i/p E	A	B	o/p 2^3 y_3	2^2 y_2	2^1 y_1	2^0 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

$y_3 = AB$, $y_2 = A\bar{B}$, $y_1 = \bar{A}B$, $y_0 = \bar{A}\bar{B}$



Two i/ps are decoded into 4 o/ps, each o/p representing one of the minterms of 2 i/p variable.

The two inverters provide complement of the i/ps & each one of 4 AND gates generates one of the minterms.

Applications

→ Code converters
→ Implementation of combinational circuits
→ Address decoding
→ BCD to 7-Segment decoder

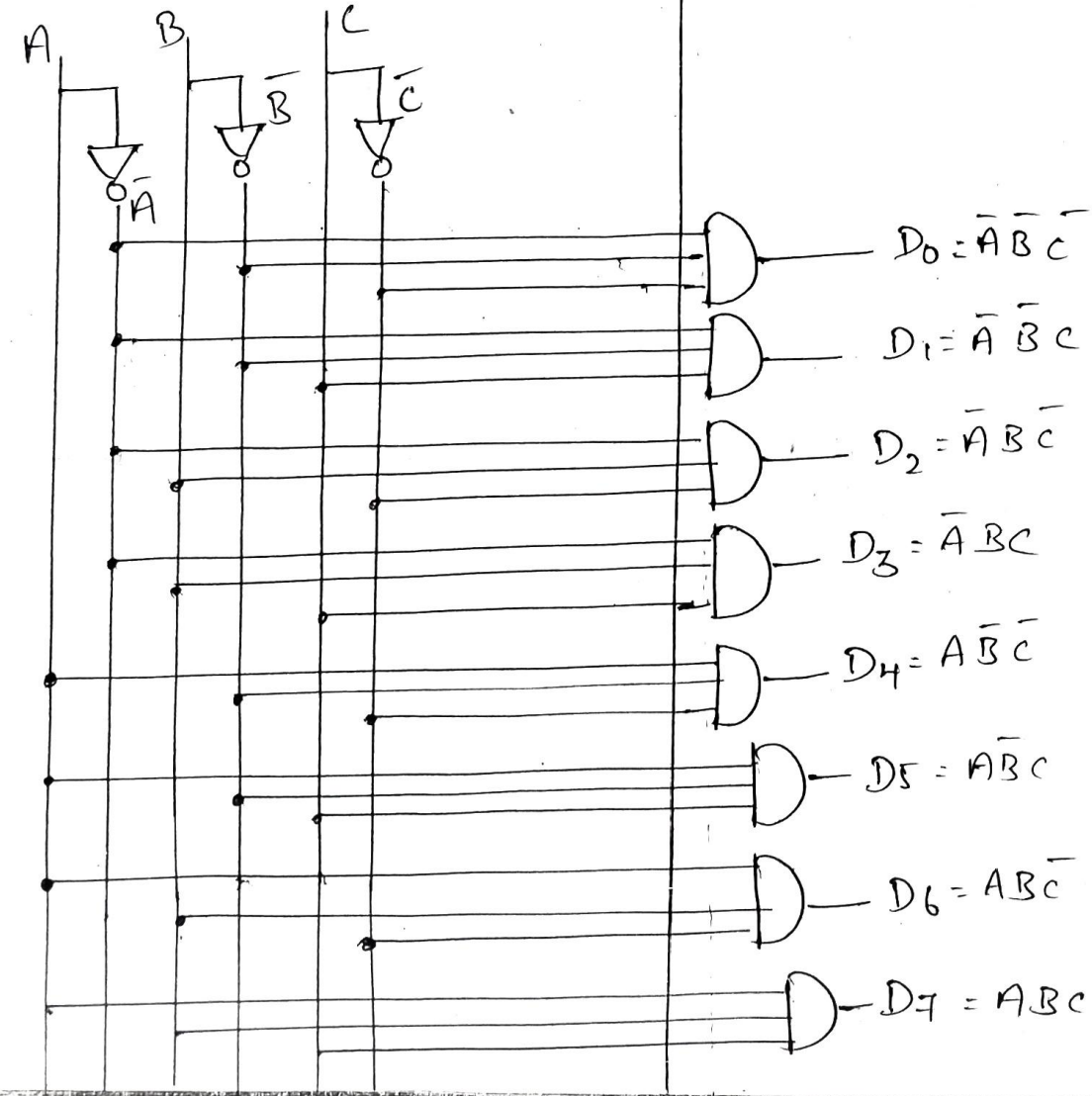
3 to 8 decoder

→ 3 i/p (A, B, C)
→ 8 o/p (D_0 to D_7)
→ Only one of eight o/ps (D_0 to D_7) is selected based on 3 Select i/p.

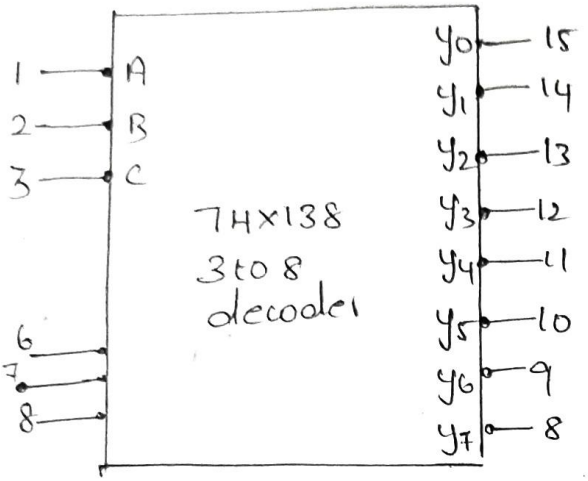
E	A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{A}\bar{B}\bar{C}$ $D_1 = \bar{A}\bar{B}C$ $D_2 = \bar{A}B\bar{C}$ $D_3 = \bar{A}BC$

$D_4 = A\bar{B}\bar{C}$ $D_5 = A\bar{B}C$ $D_6 = AB\bar{C}$ $D_7 = ABC$



IC 74x138 3to8 decoder



74x138, it accepts 3 binary i/ps (ABC) & when enabled Provides eight individual active low o/ps (y_0-y_7).

The device has three Enable i/ps

two active low ($\overline{G_{2A}}$, $\overline{G_{2B}}$)

one active high (G_1)

Decoder may have one of the two o/p states

(i) active low

(ii) active high

FOR active high output

active high o/p + OR gate in the o/p

= minterms of SOP

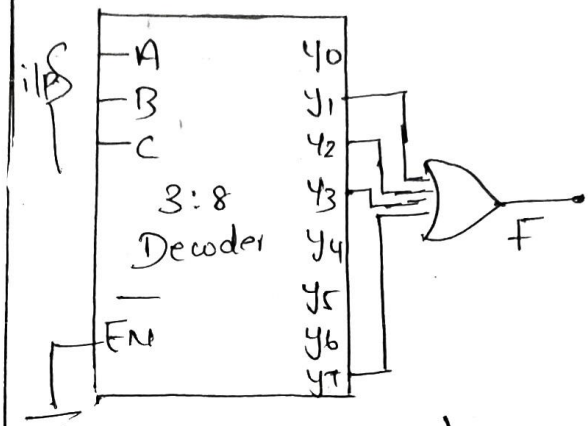
active high o/p + NOR gate in the o/p

= maxterms of POS

Ex: $\sum m(1,2,3,7) = F$ using 3x8 decoder.

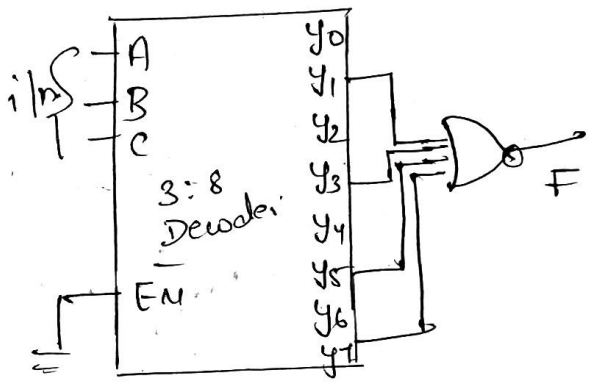
$F = \sum m(1,2,3,7)$

↓ minterm



$\therefore F = \overline{\sum m(1,3,5,7)}$

↓ maxterm

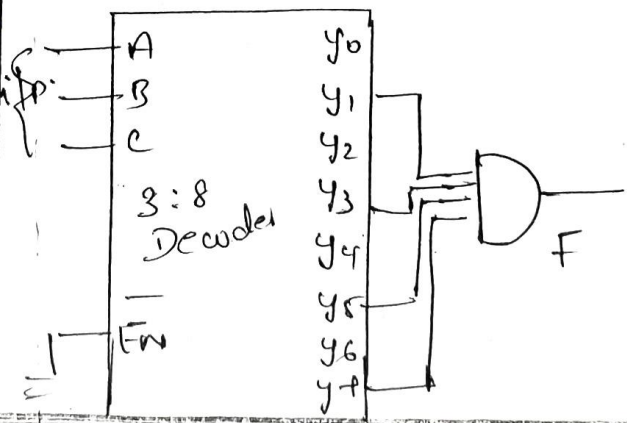


For active low o/p

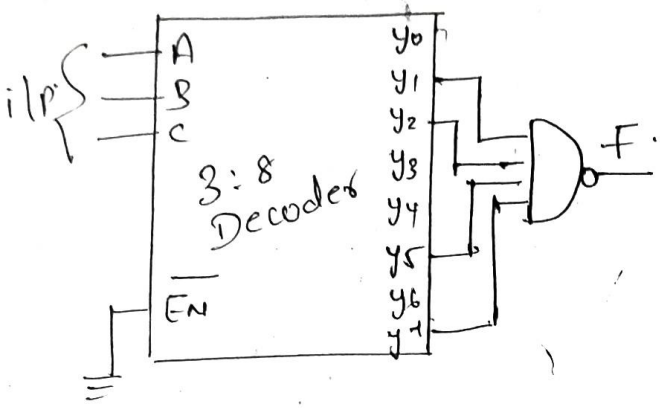
active low o/p + and gate at o/p = maxterms of POS

active low o/p + NAND gate at o/p = Minterms of SOP.

$F = \sum m(1,3,5,7)$



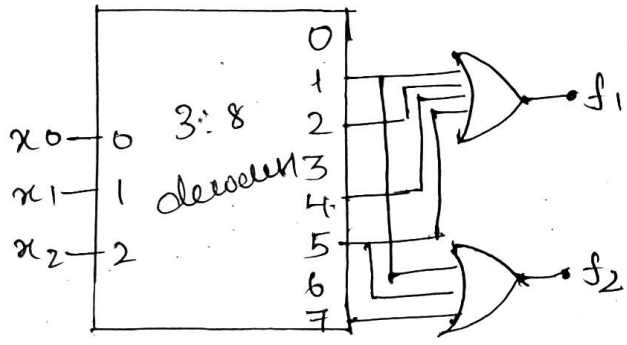
$F = \Sigma m(1,2,5,7)$



Implementation of logic Boolean expression using decoder.

① Implement the function $f_1(x_2,x_1,x_0) = \Sigma m(1,2,4,5)$ & $f_2 = \Sigma m(1,5,7)$ using 3 to 8 line decoder.

Soln:-



② Implement $f_1(x_2,x_1,x_0) = \Sigma m(0,1,3,4,5,6)$ & $f_2 = \Sigma m(1,2,3,4,6)$

Soln

$f_1 = \Sigma m(0,1,3,4,5,6)$

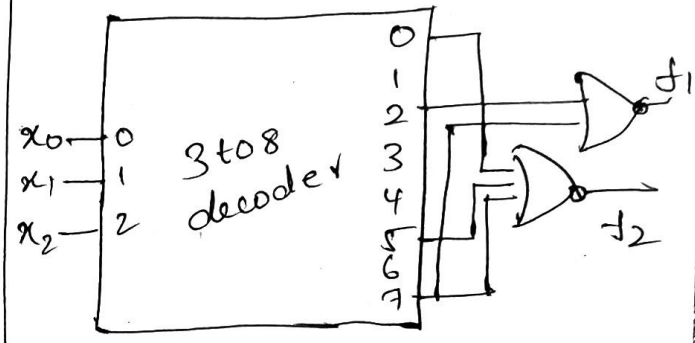
$\overline{f_1} = \Sigma m(2,7)$

$f_1 = \overline{\Sigma m(2,7)}$

$f_2 = \Sigma m(1,2,3,4,6)$

$\overline{f_2} = \Sigma m(0,5,7)$

$f_2 = \overline{\Sigma m(0,5,7)}$

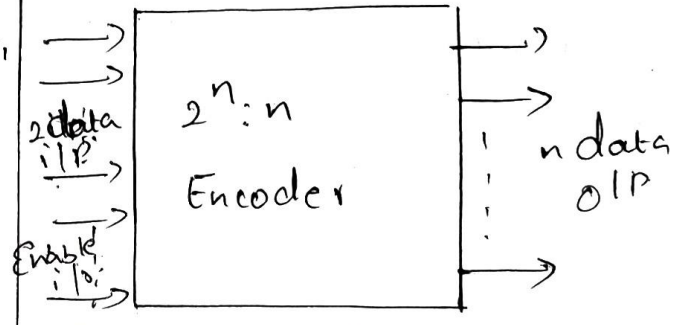


Encoder

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

→ An Encoder has 2^n (or fewer) i/p lines & n o/p lines

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



Types of Encoder

i) Priority Encoder

ii) Decimal to BCD

iii) Octal to binary

iv) Hexadecimal to Binary

i) Priority Encoder

A Priority Encoder is an Encoder circuit that includes the Priority function. In Priority Encoder, if 2 or more i/p are equal to 1 at the same time, the i/p having the highest Priority will take the precedence

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

D_3 is assigned with highest Priority

D_0 is the lowest priority

$D_3 > D_2 > D_1 > D_0$

[Highest Priority] [lowest Priority]

V → Valid output indicator

* If $V=0$, it indicates that no Priority is assigned to any of the inputs

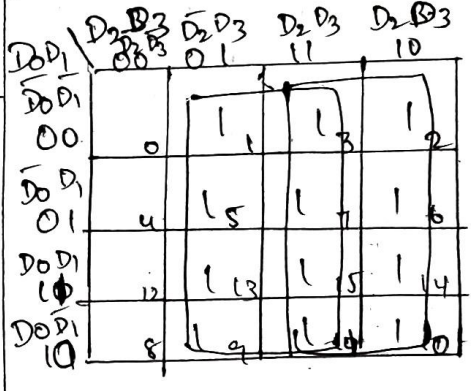
→ if $V=1$, it indicates that Priority is assigned to one of the i/ps.

→ when $D_3 = 1$, output is always $y_1 y_0$ 11

→ when $\{D_2 = 1, D_3 = 0\}$, Output is always $y_1 y_0$ 10

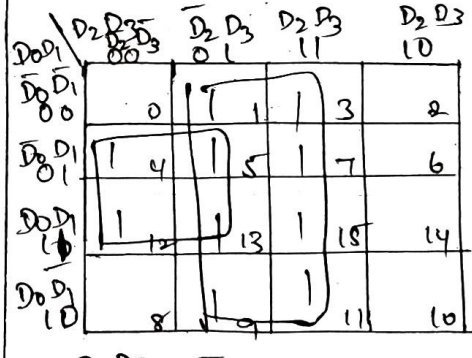
→ when $\{D_3 = 0, D_2 = 0, D_1 = 1\}$, the output is always $y_1 y_0$ 01

K-map for y_1.

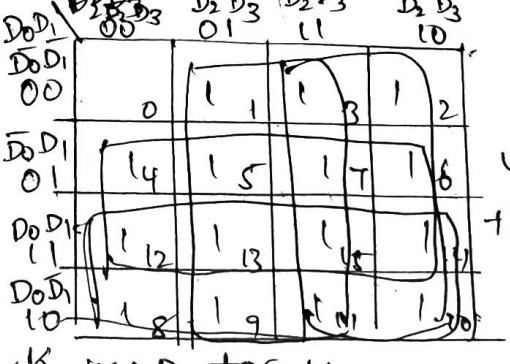


$y_1 = D_2 + D_3$

K-map for y_0

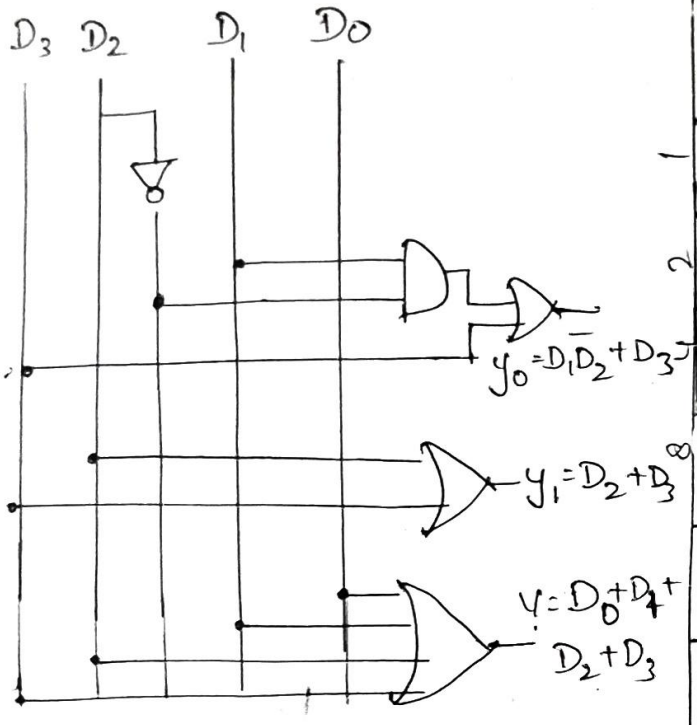


$y_0 = D_3 + \bar{D}_1 D_2$



$V = D_0 + D_1 + D_2 + D_3$

K-map for V.



Application of Priority Encoder:

→ It can generate the o/p code based on the highest Prioritized o/p.

→ Robotic Vechile

→ Industries

→ Monitoring Systems etc

ii) Decimal to Binary / BCD Encoder

Decimal to BCD Encoder has got => 10 i/ps & 4 o/ps.

[10 to 4 Encoder

i/ps → $D_9, D_8, D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0$

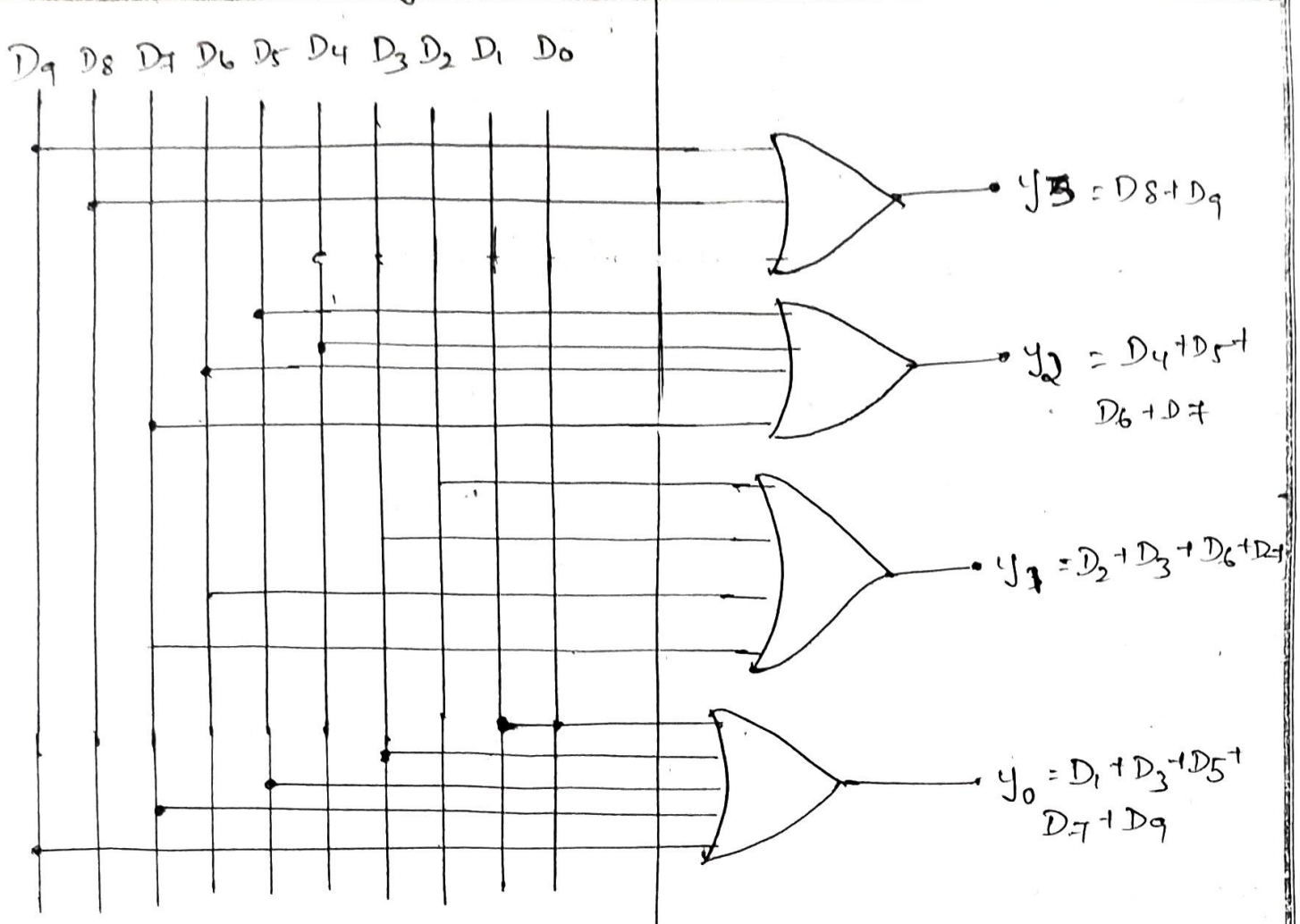
o/p → y_3, y_2, y_1, y_0

i/p	y_3	y_2	y_1	y_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	1

$y_3 = D_8 + D_9$, $y_2 = D_4 + D_5 + D_6 + D_7$

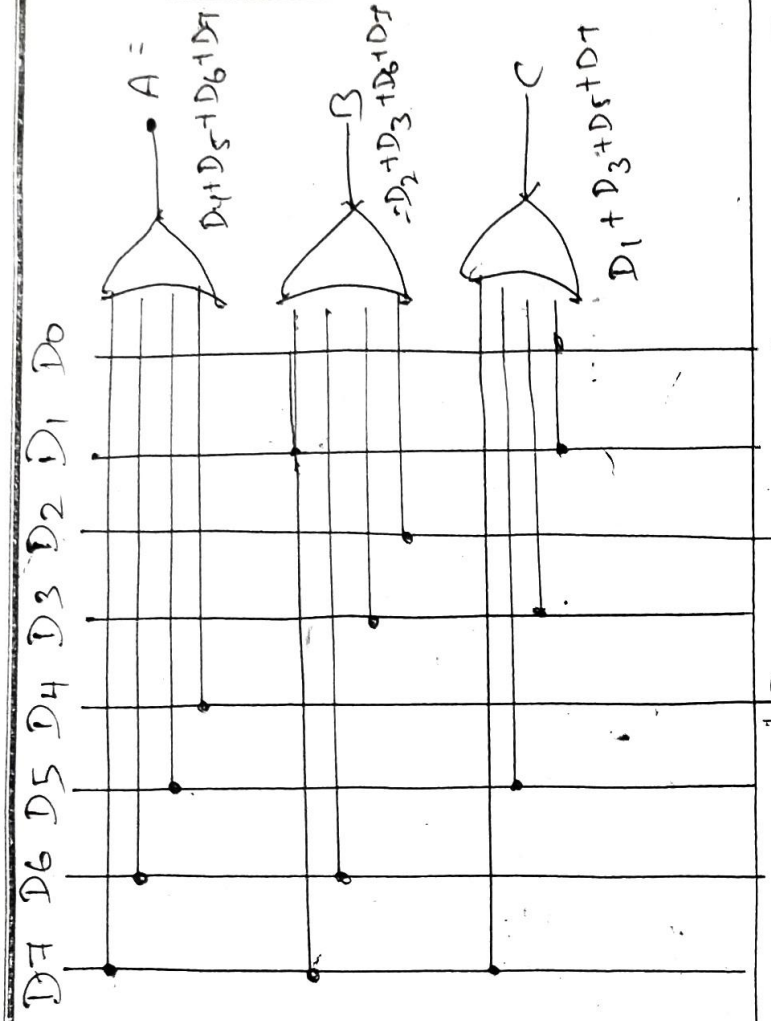
$y_1 = D_2 + D_3 + D_6 + D_7$

$y_0 = D_1 + D_3 + D_5 + D_7 + D_9$



iii) Octal to Binary Encoder

Input								Output			
								4	2	1	
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	D_0
0	1	0	0	0	0	0	0	0	0	1	D_1
0	0	1	0	0	0	0	0	0	1	0	D_2
0	0	0	1	0	0	0	0	0	1	1	D_3
0	0	0	0	1	0	0	0	1	0	0	D_4
0	0	0	0	0	1	0	0	1	0	1	D_5
0	0	0	0	0	0	1	0	1	1	0	D_6
0	0	0	0	0	0	0	1	1	1	1	D_7



Octal to binary Encoder has Eight i/ps & three o/ps. It is assumed that only one i/p has value 1 at any time.

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

Hexadecimal to Binary Encoder

Hexadecimal to Binary [16 to 2] Encoder

$n = 16 = 2^m \Rightarrow m = \log_2 16$

$= \log_2 2^4 = 4 \rightarrow$ o/p

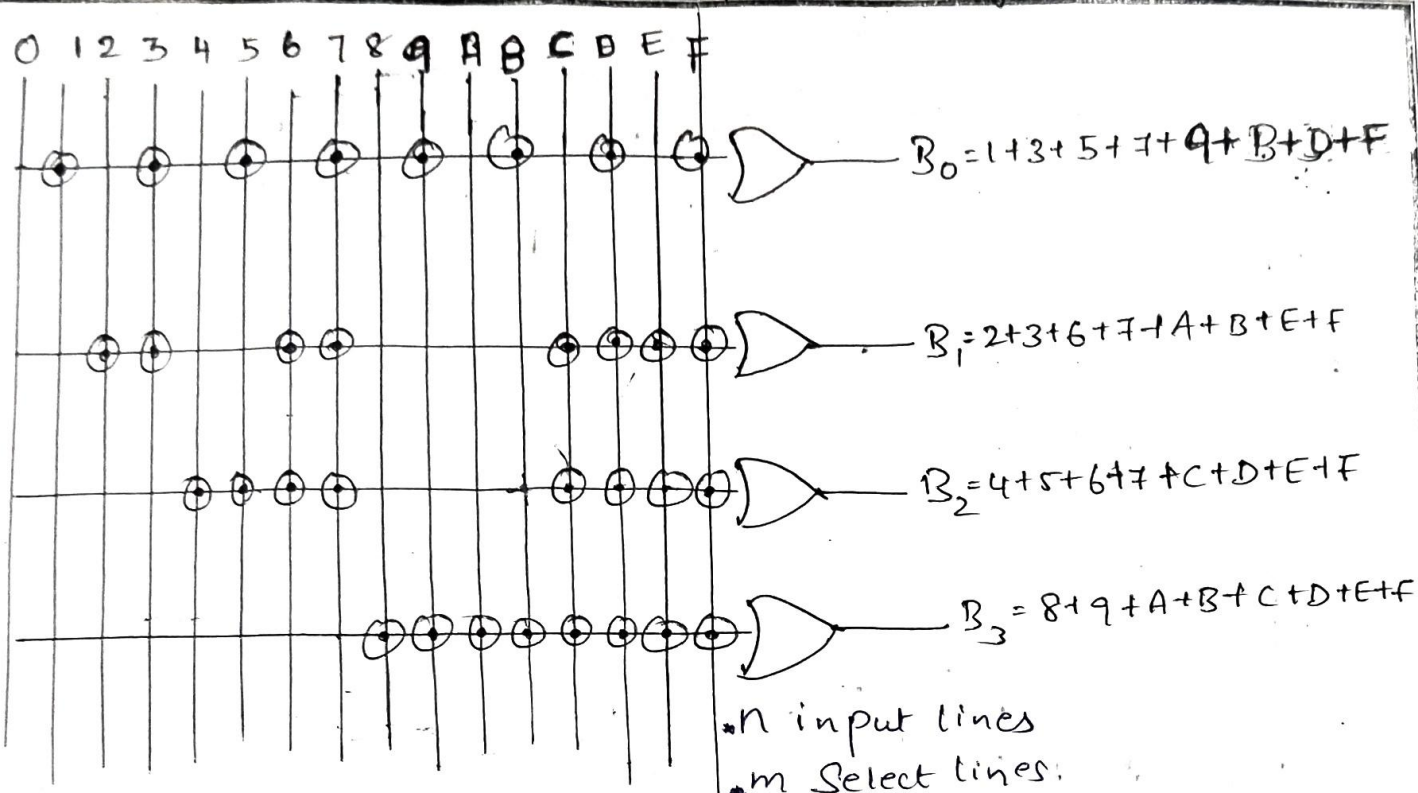
i/p Hexa-decimal	B_3	B_2	B_1	B_0	o/p
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
A (10)	1	0	1	0	A
B (11)	1	0	1	1	B
C (12)	1	1	0	0	C
D (13)	1	1	0	1	D
E (14)	1	1	1	0	E
F (15)	1	1	1	1	F

$B_0 = 1 + 3 + 5 + 7 + 9 + B + D + F$

$B_1 = 2 + 3 + 6 + 7 + A + B + E + F$

$B_2 = 4 + 5 + 6 + 7 + C + D + E + F$

$B_3 = 8 + 9 + A + B + C + D + E + F$



- n input lines
- m Select lines.
- 1 olp line
- $n = 2^m$
- To Select 1 out of 4 ilp lines, 2 select lines are required.

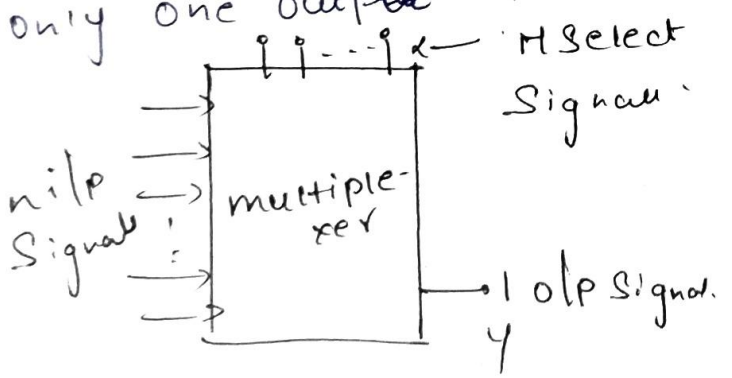
Multiplexer

Multiplexer is a combinational logic circuit used to Select Only one ilp among several inputs based on selection lines.

→ This can act as a digital switch.

→ This is also called as data Selector

→ For a mux there can be 2^n inputs n Selection lines & only one output



2x1 multiplexer

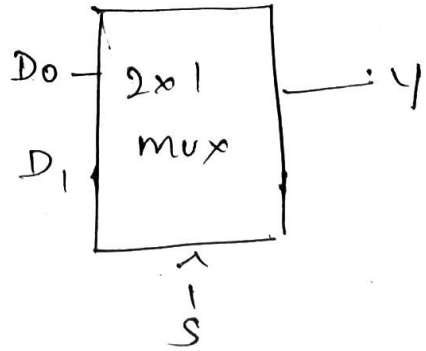
ilp is 2 $n=2, m=1$

2 ilp lines : D_0, D_1

1 Select lines = S_0

1 olp line: Y

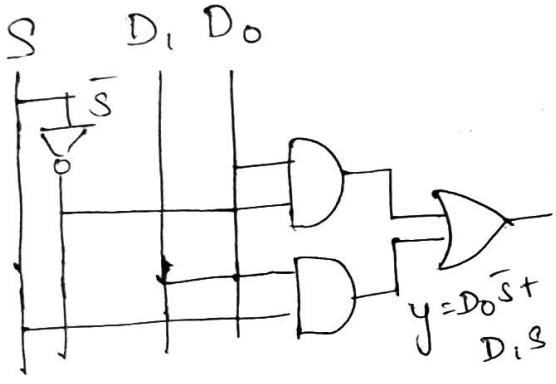
logic Symbol



Truth table

I/P (Data select)	O/P
S	y
0	D_0
1	D_1

$y = D_0 \bar{S} + D_1 S$



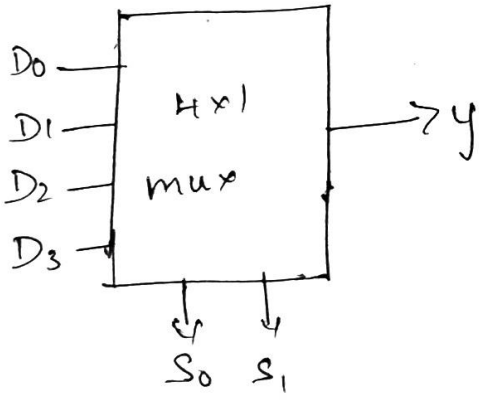
4×1 multiplexer

4 i/p Signals - D_0, D_1, D_2, D_3

2 select lines - S_0, S_1

1 output lines - y

Logic Symbol



Truth table

Data select i/p		output
S_1	S_0	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$y = D_0$,
if $S_1 = 0$ & $S_0 = 0$

$y = D_0 \bar{S_0} \bar{S_1}$

$y = D_1$,
if $S_1 = 0$; $S_0 = 1$

$y = D_1 S_0 \bar{S_1}$

$y = D_2$,
if $S_1 = 1, S_0 = 0$

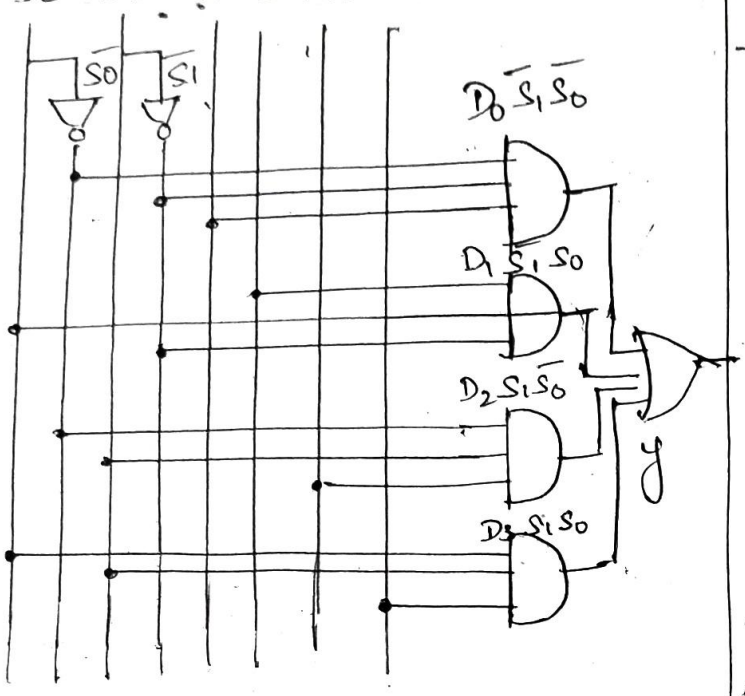
$y = D_2 \bar{S_0} S_1$

$y = D_3$,
if $S_1 = 1, S_0 = \phi$

$y = D_3 S_0 S_\phi$

$y = D_0 \bar{S_0} \bar{S_1} + D_1 S_0 \bar{S_1} + D_2 \bar{S_0} S_1$
$+ D_3 S_0 S_1$

Logic Diagram



Truth table:

inputs				outputs	
$\bar{E}$	S_2	S_1	S_0	y	$\bar{y}$
1	x	x	x	0	0
0	0	0	0	D_0	$\bar{D}_0$
0	0	0	1	D_1	$\bar{D}_1$
0	0	1	0	D_2	$\bar{D}_2$
0	0	1	1	D_3	$\bar{D}_3$
0	1	0	0	D_4	$\bar{D}_4$
0	1	0	1	D_5	$\bar{D}_5$
0	1	1	0	D_6	$\bar{D}_6$
0	1	1	1	D_7	$\bar{D}_7$

$$y = [D_0 \bar{S}_2 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_2 \bar{S}_1 S_0 + D_2 \bar{S}_2 S_1 \bar{S}_0 + D_3 \bar{S}_2 S_1 S_0 + D_4 S_2 \bar{S}_1 \bar{S}_0 + D_5 S_2 \bar{S}_1 S_0 + D_6 S_2 S_1 \bar{S}_0 + D_7 S_2 S_1 S_0] \bar{E}$$

8:1 multiplexer

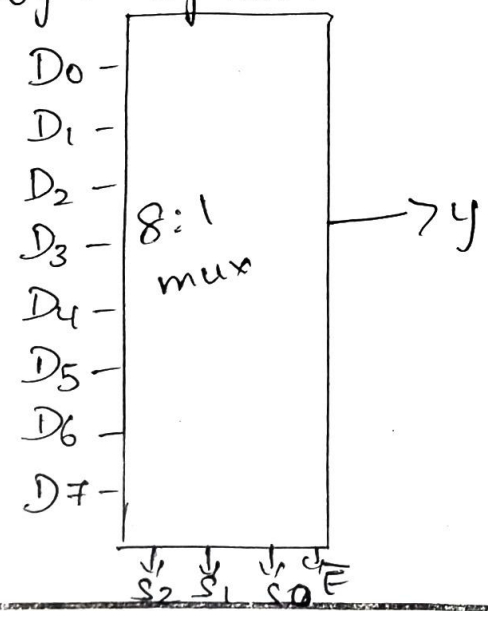
8 data i/ps (D_0-D_7) lines

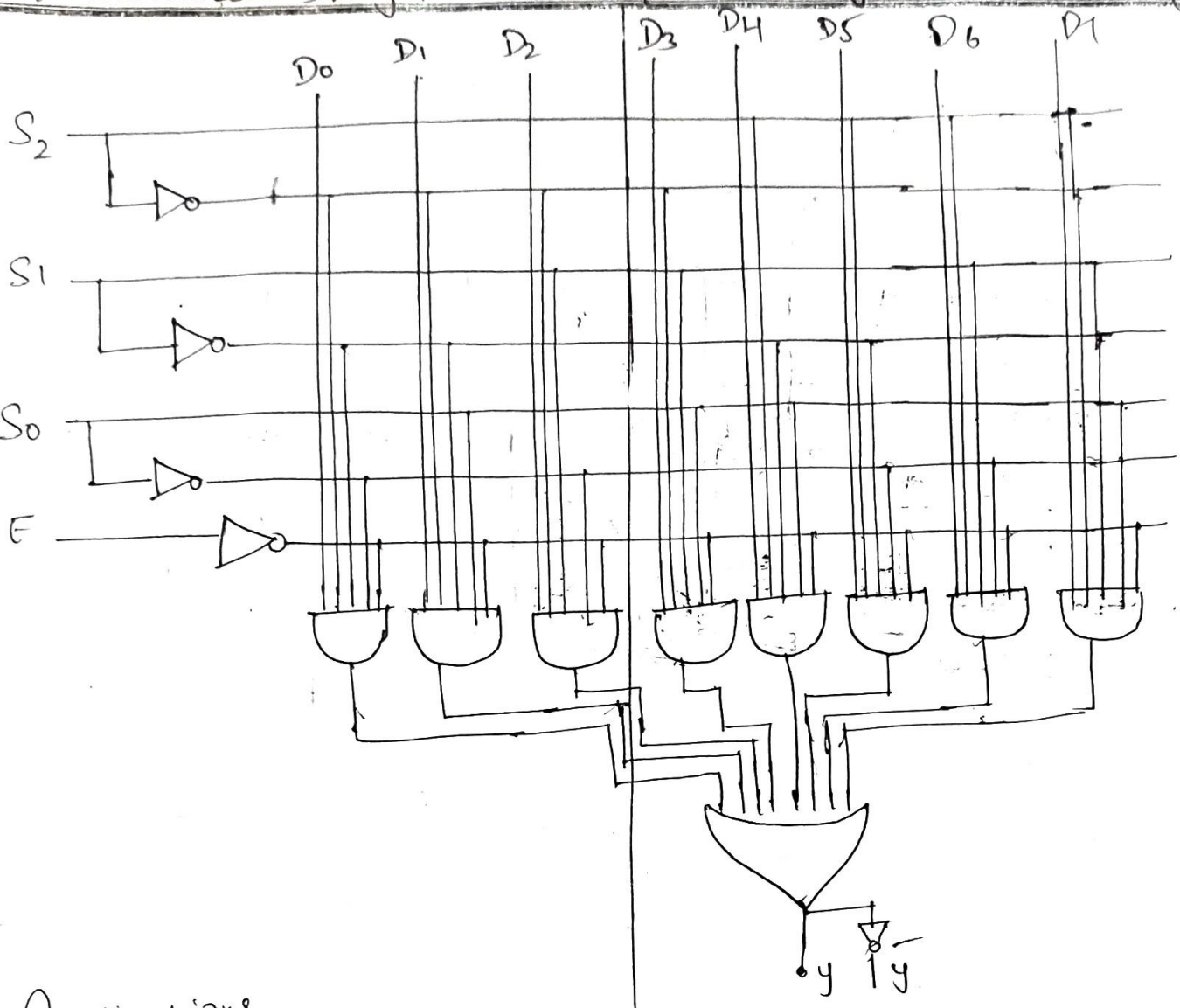
3 Select i/ps (S_2-S_0) lines, $2^3=8$

1 o/p line (y)

Enable i/p ($\bar{E}$), when $\bar{E}=0$, the Select i/p's $S_2 S_1 S_0$ will Select one of the data i/p to pass through o/p y. when $\bar{E}=1$, mux is disabled.

Logic Symbol





Applications

→ Data Selection
→ Data routing
→ Operation Sequencing
→ Parallel to Serial Conversion
→ Waveform generation.

1) Implement the following Boolean expression using Suitable multiplexer.

$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

Soln:

Since the given function is a four variable function, we need a multiplexer with 8 i/p lines & 3 Select lines

Apply variable B, C, D to the Select lines.

Implementation table.

	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

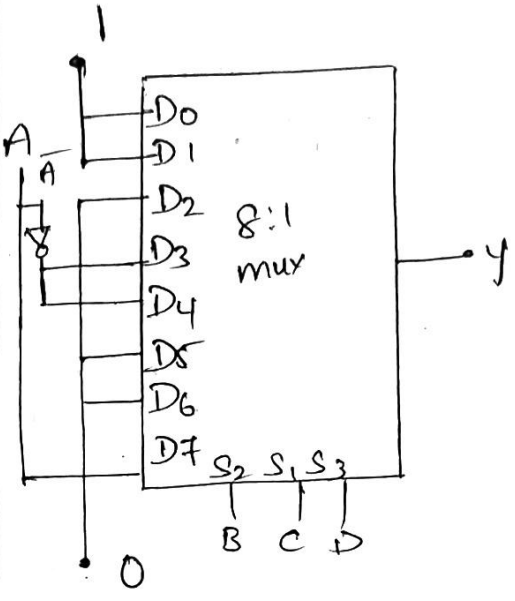
→ Circle the minterms of the function

→ Both terms i.e minterms in a column are not circled, apply '0' the corresponding i/p

→ If both minterms are circled, apply 1 to the corresponding i/p.

→ If top minterm alone is circled, apply $\bar{A}$ & if bottom minterm alone is circled apply A

According to implementation table, draw 8x1 mux apply logic 0 & logic 1 to the i/p.



2) Implement the following Boolean function using 4:1 mux

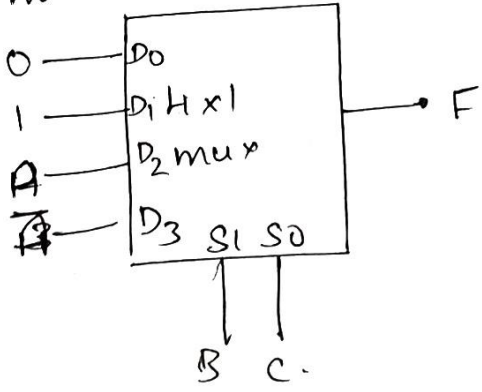
$F(A,B,C) = \Sigma m(1,3,5,6)$

Soln

Implementation table.

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

mux implementation



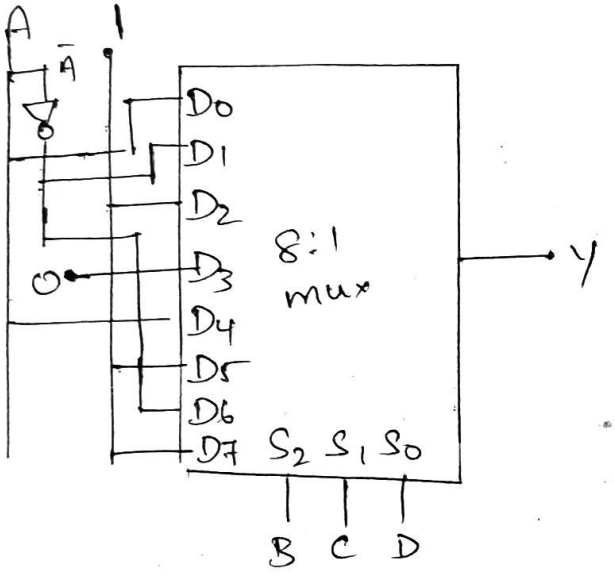
3) Implement $F = \Sigma(1,2,5,6,7,8,10,12,13,15)$ using 74151 mux

Sol

Implementation table

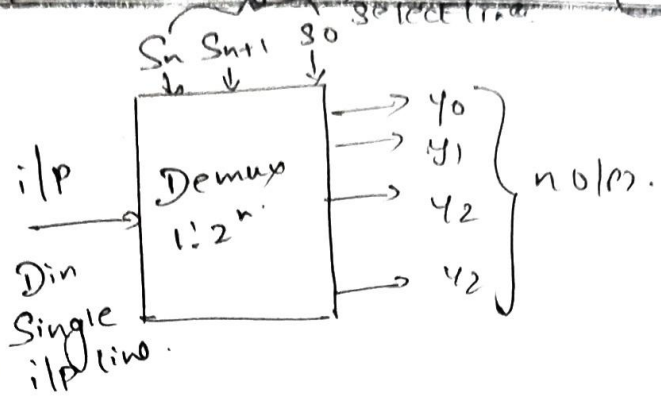
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	(2)	3	4	(5)	(6)	(7)
A	(8)	9	(10)	11	(12)	(13)	14	(15)
	A	$\bar{A}$	1	0	A	1	$\bar{A}$	1

Multiplexer implementation



Demultiplexers

A demultiplexer is a combinational logic circuit that receives information on a single input lines & transmits the same information on one of 2^n possible output lines



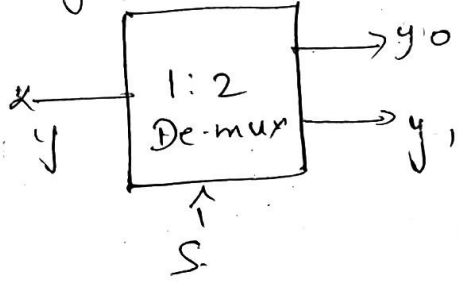
1 to 2 Demultiplexer

input — 1
Output — y_0, y_1
Select lines — n (S)

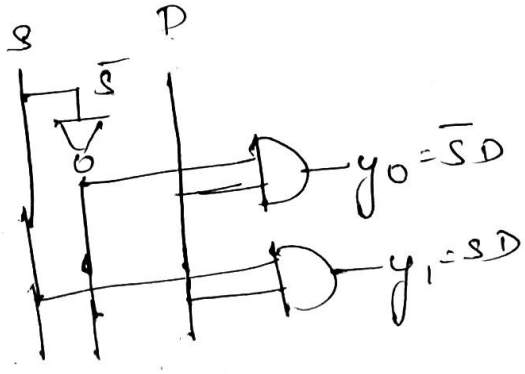
Truth table

Data i/p	Select i/p	o/p	
D	S	y_1	y_0
D	0	0	D
D	1	D	D

logic Diagram



$y_0 = \bar{S}D$, $y_1 = SD$



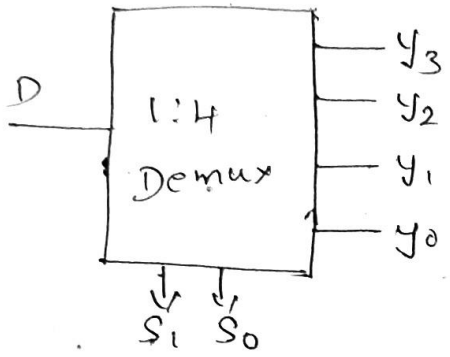
1:4 Demultiplexer

1 Input - Single (D)

Output = 4 (y_0, y_1, y_2, y_3)

Select lines - 2 (S_0, S_1)

logic diagram



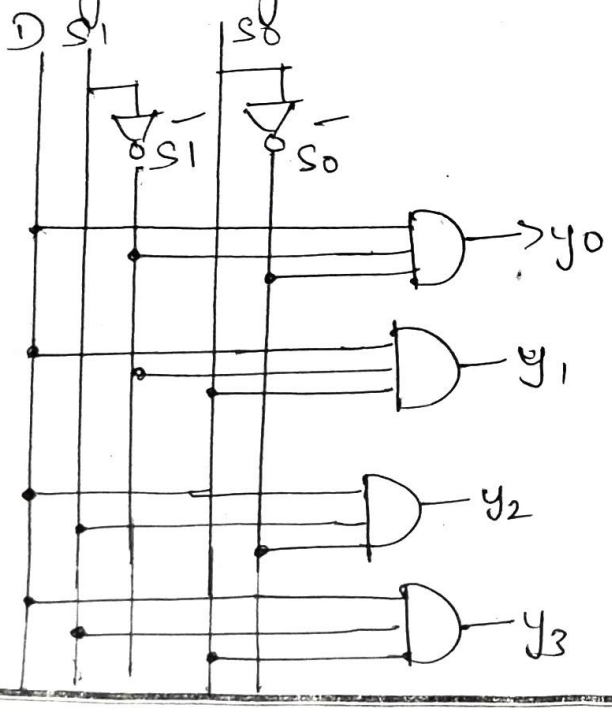
Truth table

Data i/p	Select i/p		Outputs			
D	S_1	S_0	y_3	y_2	y_1	y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

$y_0 = D\bar{S_1}\bar{S_0}$, $y_1 = D\bar{S_1}S_0$, $y_2 = DS_1\bar{S_0}$

$y_3 = DS_1S_0$

Logic Diagram



1:8 Demultiplexer

Consider a 1 to 8 demultiplexer which has a Single i/p (D) & Eight O/p (D_0 to D_7) & three Select lines & input (S_0, S_1 & S_2)

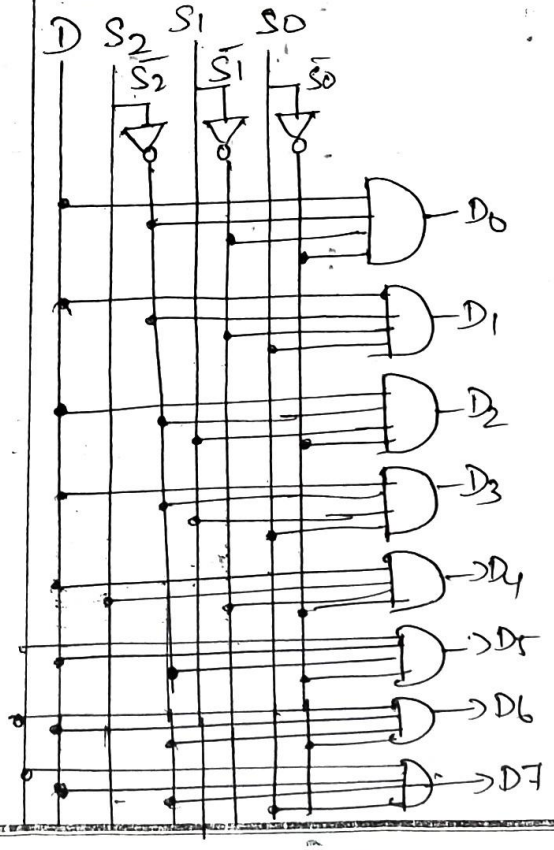
Inputs				Outputs							
S_2	S_1	S_0	D	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	D	D	0	0	0	0	0	0	0
0	0	1	D	0	D	0	0	0	0	0	0
0	1	0	D	0	0	D	0	0	0	0	0
0	1	1	D	0	0	0	D	0	0	0	0
1	0	0	D	0	0	0	0	D	0	0	0
1	0	1	D	0	0	0	0	0	D	0	0
1	1	0	D	0	0	0	0	0	0	D	0
1	1	1	D	0	0	0	0	0	0	0	D

$D_0 = D\bar{S_2}\bar{S_1}\bar{S_0}$, $D_1 = D\bar{S_2}\bar{S_1}S_0$

$D_2 = D\bar{S_2}S_1\bar{S_0}$, $D_3 = D\bar{S_2}S_1S_0$

$D_4 = DS_2\bar{S_1}\bar{S_0}$, $D_5 = DS_2\bar{S_1}S_0$

$D_6 = DS_2S_1\bar{S_0}$, $D_7 = S_2S_1S_0$



Applications of Demultiplexer:

→ used as decoder, data distributor

→ used in time division multiplexing at the receiving end as a data separator.

→ used to implement Boolean Expressions

74154 – 1:16 Demultiplexer

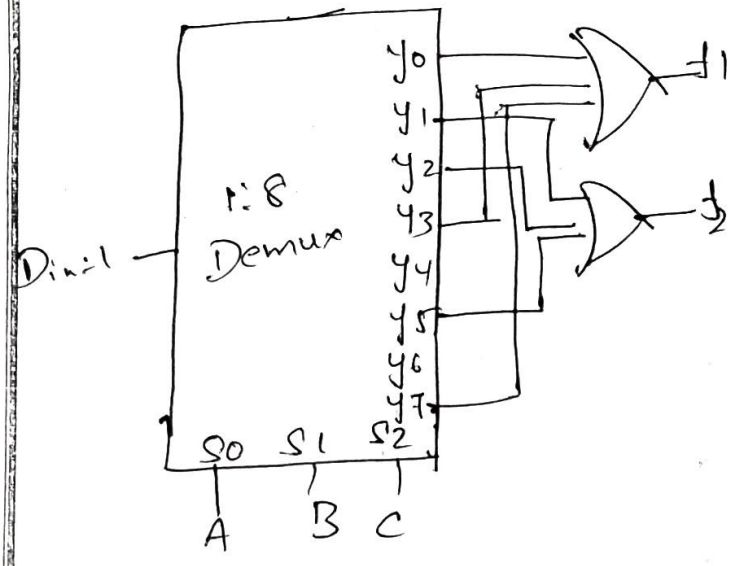
74155 – Dual 1:4 Demultiplexer

Demultiplexer Problems

1. Implement the following functions using demultiplexer

Sol $f_1(A,B,C) = \Sigma m(0,3,7)$

$f_2(A,B,C) = \Sigma m(1,2,5)$



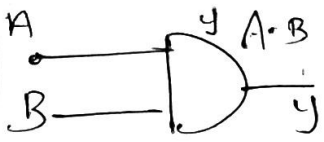
Logical gates

AND, OR, NOT – Basic gates

NAND, NOR – Universal gate

Ex-OR, Ex-NOR – Exclusive / Special gates

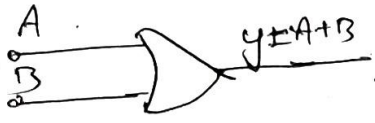
AND (IC 7408)



Truth table

A	B	y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

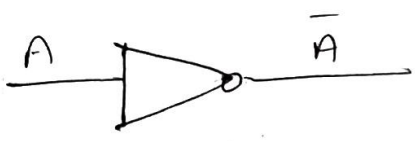
OR (IC 7432)



Truth table

A	B	y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

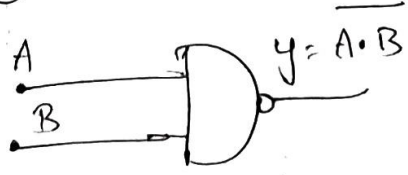
(3) NOT (IC 7404)



Truth table

A	$y = \bar{A}$
0	1
1	0

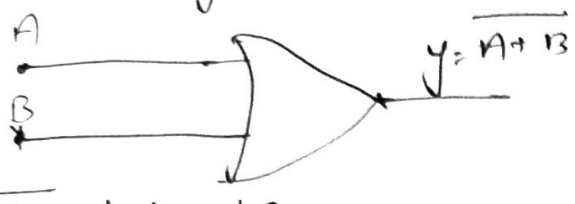
(4) NAND Gate (IC 7400)



Truth Table

A	B	$y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

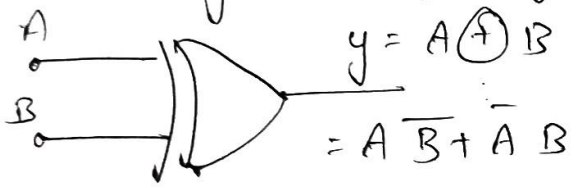
5) NOR gate (IC 7402)



Truth table

A	B	$y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

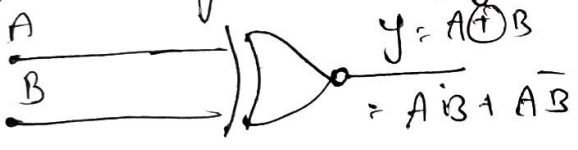
6) Ex-OR gate (Inequality Detector)



Truth table -

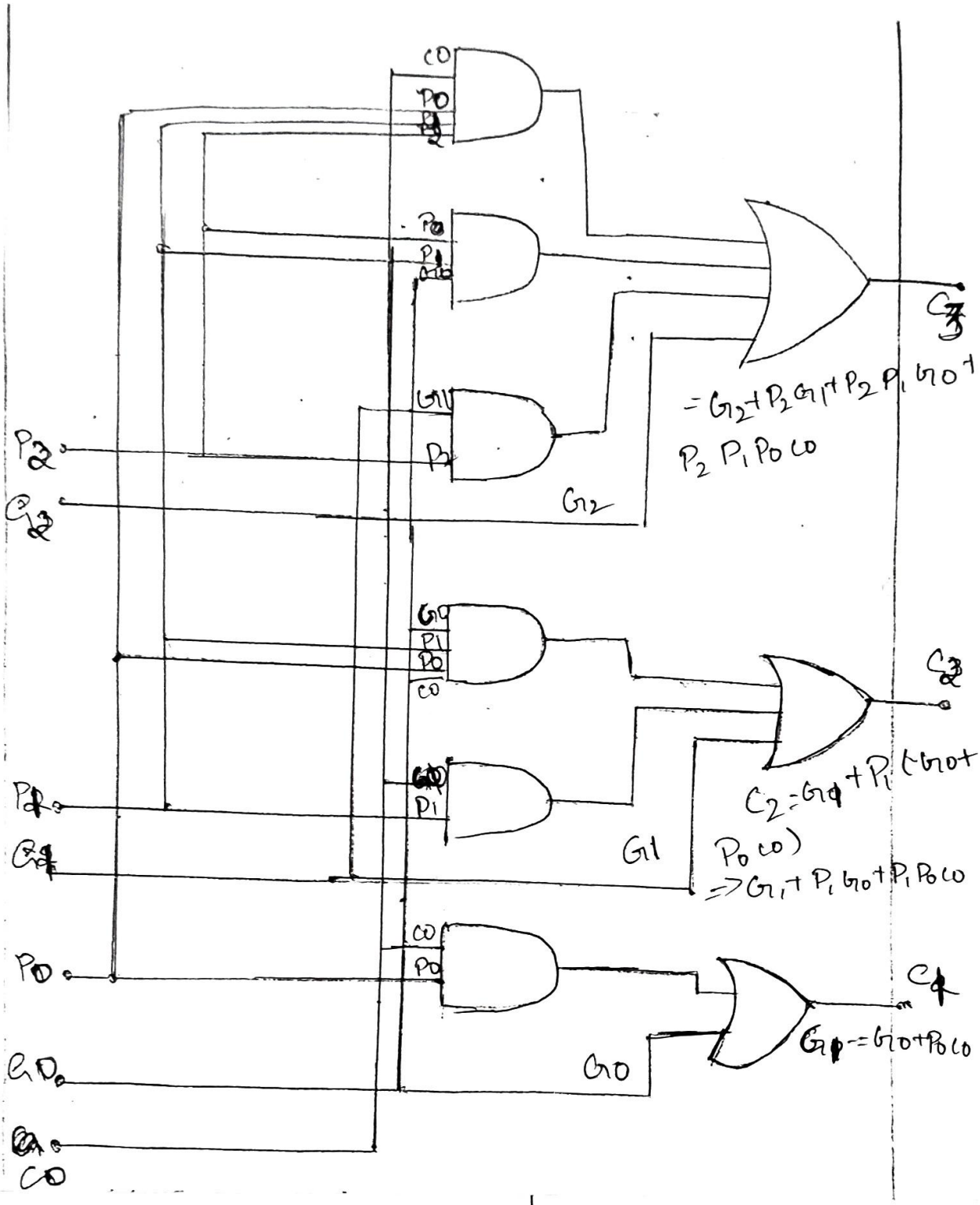
A	B	$y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

7) Ex-NOR gate [Equality Detector]



Truth table -

A	B	$y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1



look carry ahead adder.