(BCS-II)

Syllabus :-

UNIT-I

Binary Codes - Weighted and Non-Weighted - Binary Arithmetic Conversion Algorithms - Error Detecting and Error Correcting Codes - Canonical and Standard Boolean Expressions - Truth Tables.

UNIT-II

K-Map Reduction - Don't Care Condition - Adders/Subtractors - Carry Look-Ahead Adder - Code Conversions Algorithm - Design of Code Converters - Equivalence functions. Binary/Decimal Parallel Adder/Subtractor for Signed Numbers - Magnitude Comparator - Decoders, Encoders - Multiplexers/Demultiplexers - Boolean function Implementation using Multiplexers.

UNIT-III

Sequential Logic - Basic Latch - Flip-flops (SR, D, JK, T and Master-Slave) - Triggering of flip-flops - Counters - Design Procedure - Ripple Counters - BCD and Binary - Synchronous Counters.

UNIT-IV

Registers - Shift Registers - Registers with Parallel Load - Memory Unit - Example of RAM, ROM, PROM, EPROM - Reduction of State and flow Tables - Race-free State Assignment - Hazards.

8620 Convert this decimal no. into BCD and Excess-3 code.

8 6 2 0

1000 0110 0010 0000

BCD - 1000 | 0110 | 0010 | 0000

1000 0110 0010 0000
0011 0011 0011 0011

XS-3 = 1011 1001 0101 0011

XS-3 = 1011 | 1001 | 0101 | 0011

Digital Systems and Binary Numbers

⇒A sequence of instruction is called Program.
⇒Any set that is restricted to a finite number of elements contains discrete information. exp— 10 decimal digits, the 26 letters of the alphabet, the 52 playing cards, and the 64 squares of a chessboard.

Number System:—

1. Binary Number System – Base 2 (1, 0)
2. Octal Number System – Base 8 (0, 1, 2, 3, 4, 5, 6, 7)
3. Decimal Number System – Base 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
4. Hexadecimal Number System – Base 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

Conversion:—

1. Binary to Octal—

$(10110)_2 = (?)_8$

1 0 1 1 0
16 8 4 2 1

$= 16 + 4 + 2$
$= 22$

$(10110)_2 = (26)_8$

5 5
101 101

$(101101)_2 = (55)_8$

2. Octal to Binary—

~~$(1010111100)_2$~~

$(1274)_8 = (?)_2$

1 2 7 4
001 010 111 100

$(1274)_8 = (001010111100)_2$

Octal Digit Value	Binary equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

3. Binary to Decimal—

$(101110)_2 = (?)_{10}$

1 0 1 1 1 0
32 16 8 4 2 1

$= 32 + 8 + 4 + 2$
$= 46$

$(101110)_2 = (46)_{10}$

4. Decimal to Binary

$(46)_{10} = (?)_2$

$(46)_{10} = (101110)_2$

2	46
2	23 0
2	11 1
2	5 1
2	2 1
2	1 0
	0 1

5. Binary to Hexadecimal—

$(100|0100|111)_2 = (?)_{16}$

0|100|0100|111

4 5 7

$(100|0100|111)_2 = (457)_{16}$

6. Hexa to Binary—

$(B2C)_{16} = (?)$

B 2 C

1011 0010 1100

$(B2C)_{16} = (101100101100)_2$

Hexa digit value	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

7. Octal to Decimal—

$(345)_8 = (?)_{10}$

3 4 5

64 8 1

$= 3\times64 + 4\times8 + 5\times1$

$= 229$

$(345)_8 = (229)_{10}$

8. Decimal to Octal—

$(229)_{10} = (?)_8$

8	229
8	28 5
8	3 4
	3

$(229)_{10} = (345)_8$

9. Octal to Hexa:—

$(345)_8 = (?)_{16}$

3 4 5

011 100 101

011|100|101

E 5

E5

$(345)_8 = (E5)_{16}$

10. Hexa to Octal—

$(A2D)_{16} = (?)_8$

A 2 D

1010 0010 1101

|0|000|0|101|

5 0 5 5

$(A2D)_{16} = (5055)_8$

11. Hexa to Decimal:—

$(A2DE)_{16} = (?)_{10}$

A 2 D E

4096 256 16 1

$= 10\times4096 + 2\times256 + 13\times16 + 14\times1$

$= 41694$

$(A2DE)_{16} = (41694)_{10}$

12. Decimal to Hexa:—

$(41694)_{10} = (?)_{16}$

16	41694
16	2605 14
16	162 13
16	10 2
	10

$(41694)_{10} = (A2DE)_{16}$

Binary Code:- A binary code represents text, computer processor instruction, or any other data using a two-symbol system. The two symbol system used is often "0" and "1" from the binary number system. The binary code assigns a pattern of binary digits, also known as bits, to each character, instruction etc. for exp- a binary string of eight bits can represent any of 256 possible values and can, therefore, represent a wide variety of different ~~patterns~~ items.

Range of an n-bit binary number:-

1-bit → 0, 1 0 to 1

2-bit → 0 → 00
1 → 01
2 → 10
3 → 11

3-bit → 0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110
7 → 111

for n bit Range is $\boxed{0 \text{ to } (2^n - 1)}$

for 2 bit $0 \text{ to } 2^2 - 1$

$\boxed{0 \text{ to } 3}$

for 3 bit $0 \text{ to } 2^3 - 1$

0 to 7

Classification of Binary Codes:-

- Weighted Codes
  - Positive Weighted Code
    - 8421
    - 2421
    - 3321
    - 4221
  - Negative Weighted Code
    - 84-2-1
    - 74-2-1

Decimal	8421 (Binary)
0	0000
1	0001
7	0111
9	1001

Decimal	2421
0	0000
1	0001
7	0111
9	1111

Decimal	3321
0	0000
1	0001
7	1101
9	1111

Decimal	4221
0	0000
1	0001
7	1101
9	1111

Decimal	84-2-1
1	0111
7	1001
9	1111

Non Weighted Code

Excess 3 Code OR XS-3 Code | Gray Code

Decimal	Binary (8421)	Excess 3 Code (8421) (8421 + 0011)
1	0001	0100 (4)
5	0101	1000 (8)
8	1000	1011 (11)

Gray Code—

I Method— Binary to Gray Conversion

1011 — Binary

MSB

Gray Code ⇒ 1110 — Sum of III + IV bit

Sum of II + III bit

Sum of I st + II Bit

In this process we don't ~~de~~calculate carry.

II Method —

1011 — Binary

1011

+ 101(1) — Omit

1110

Gray Code — 1110

In this method we don't de calculate carry.

B Gray to Binary Conversion —

1110 — Gray Code

1110

1011 — Binary Code

Discard Carry.

Subtraction of binary numbers using 2's Complement:-

#1 X = 1010100 , Y = 1000011 X-Y = ?

Y 2's Complement + X

Y 1's = 0111100

+1

Y 2's = 0111101

X = 84
Y = 67
X-Y = 17

~~X-Y = 0111101 + 1010100~~
~~= 1100~~

X-Y = 0111101 (7 bit) + 1010100 (+7 bit)

1 0010001 (8 bit) — Overflow

Then carry discard

X-Y = 0010001

#1 Y = 1000011 X = 1010100 Y-X = ?

X 1's = 0101011

+1

X 2's = 0101100

X = 84
Y = 67
Y-X = -17

Y-X = 1000011 (7bit) + 0101100 (+7bit)

R = 1101111 (7bit)

Carry not means not desired Result so calculate 2's Complement of R.

R 2's = 0010000

+1

-0010001

Y-X = -0010001

Addition Rules:-

0+0=0
0+1=1
1+0=1
1+1=0 | Carry
1+1+1=1 | Carry

Rules for Binary Subtraction:-

0-0=0
1-0=1
0-1=1 | Borrow.
1-1=0

Binary Multiplication Rule:-

0X0=0
0X1=0
1X0=0
1X1=1

Binary Division:-

0÷0=0
0÷1=0
1÷0=1
1÷1=1

```
      100110
110 ) 11100110
      110
      ----
       1001
        110
        ---
        111
        110
        ---
          10
```

Signed Magnitude Representation :-

+5 → 0101

−5 → 1101

8 bit representation of +12 —

+12 = 00001100

Signed Magnitude Rep. = 00001100

Signed 1's Complement Rep. = 00001100

Signed 2's Complement Rep. = 00001100

Positive No का तीनों same होगा।

8 bit representation of -12 :-

-12 = 00001100

Signed Magnitude Re = 10001100

1's of (12) = 11110011

Signed 1's Complement Repr

Signed Binary Numbers :-

Positive integers can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values. In ordinary arithmetic, a negative number is indicated by a minus sign and a positive no. by a plus sign. Because of hardware limitations, computers must represent everything with binary digits. It is customary to represent a sign with a bit placed in the leftmost position of the number. The convention is make the sign bit 0 for positive and 1 for negative.

It is important to realize that both signed and unsigned binary numbers consist of a string of bits when represented in a computer. If the binary number is signed, then the leftmost bits represents the sign and the ~~leftmost bit is~~

rest of bits represents the number. If the binary number is assumed to be unsigned, then the left-most bit is the most significant bit of the number.

Exp:- The string of bits 01001 can be considered as 9 (unsigned binary) or as +9 (signed binary) because the leftmost bit is 0. The string of bits 11001 represent the binary equivalent of 25 when considered as an unsigned number and the binary equivalent of -9 when considered as signed number. This is because the 1 that is in the leftmost position designates a negative and the other four bits represent binary 9.

Complements:- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation. Simplifying operations leads to simpler, less expensive circuits to implement the operation. There are two types of complements for each base-r system: the radix complement and the diminished radix complement. The first is referred to as the r's complement and the second as (r-1)'s complement. When the value of the base r is substituted in the name, the two types are referred to as the 2's complement and 1's complement for binary numbers and 10's complement and 9's complement for decimal numbers.

Signed Magnitude Representation—

+5 = 0101

-5 = (1)101

Signed bit or MSB (Most significant Bit)

8 - bit Representation of +12! —

+12 = 00001100

Signed Magnitude Representation = 00001100

Signed 1's Complement Representation = 00001100

Signed 2's Complement Representation = 00001100

Note ⇒ +ve No. का तीनो Representation same होगा।

8-bit Representation of -12! —

+12 = 00001100

Signed Magnitude Representation = 10001100

1's of (+12) = 11110011

Signed 1's Complement Representation = 11110011

2's of (+12) = 11110011

+1

11110100

Signed 2's Complement Representation = 11110100

Range of Signed Magnitude and Signed 1's Complement Representation (where n stands for no. of bit)

$$\boxed{-(2^{n-1}-1) \text{ to } (2^{n-1}-1)} \text{ or } \boxed{(-2^{n-1}+1) \text{ to } (2^{n-1}-1)}$$

Range of Signed 2's Complement Representation — $\boxed{-2^{n-1} \text{ to } (2^{n-1}-1)}$

111110 एक signed 2's Complement Representation है तो इसका decimal No. ज्ञात करो।

111110

$= -1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= -32 + 16 + 8 + 4 + 2$

$= -2$ Ans

Weighted Codes:-

Weighted binary codes are those binary codes which obey the positional weight principal. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.

Decimal → 4

Positional Weights → 8+4+2+1

Code → 0010

Non-Weighted Codes:-

In this type of binary codes, the positional weights are not assigned. Example of non-weighted codes are Excess-3 code and Gray Code.

Excess-3 Code:- The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding 0011 or 3 to each code word in 8421. The Excess-3 codes are obtained as follows:-

Decimal Number → 8421 BCD $\xrightarrow[0011]{\text{ADD}}$ Excess-3

Decimal	BCD 8421	Excess-3 BCD + 0011
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Gray Code:— It is the non weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that only one bit change each time the decimal number is incremented as shown in figure. The gray code is called unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

Decimal	BCD	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

It is also known as reflected Code.

Decimal	Binary (3 bits)	Unsigned	signed	1's complement	2's Complement
0	000	0	+0	0	0
1	001	1	+1	1	1
2	010	2	+2	2	2
3	011	3	+3	3	3
4	100	4	-0	-3	-4
5	101	5	-1	-2	-3
6	110	6	-2	-1	-2
7	111	7	-3	-0	-1

In unsigned representation first bit should not be treated as sign bit and the remaining three representation (signed, 1's complement & 2's complement) first bit treated as sign bit.

$n = 3$ bits ; $2^3 = 8$

	from	to	
unsigned (8 value)	0	7	0 to $2^3 - 1$
Signed (7 value)	-3	+3	$-(2^{3-1}-1)$ to $+(2^{3-1}-1)$
1's (7 value)	-3	+3	$-(2^{3-1}-1)$ to $+(2^{3-1}-1)$
2's (8 value)	-4	+3	$-(2^{3-1})$ to $(2^{3-1}-1)$

There are better

System में हम 2's Complement का use करते हैं signed & 1's Complement की अपेक्षा क्योंकि signed & 1's complement में -0 & +0 दो value प्राप्त होती है जबकि Mathematics में 0 मतलब 0 होता है 2's Complement में केवल 0 मानते हैं

Unsigned ⇒ 0 to $2^n - 1$

Signed ⇒ $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

1's ⇒ $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

2's ⇒ $-(2^{n-1})$ to $+(2^{n-1}-1)$

If you consider utilization of bit then unsigned and 2'sC is always better as compared to signed magnitude as well as 1's complement. $2^3 = 8$ (After 2 Page)

Convert decimal 41 to binary.

2	41	
2	20	1
2	10	0
2	5	0
2	2	1
2	1	0
	1	

$(41)_{10} = (101001)_2$

Conversion from decimal integers to any base-r system is similar to this example, except that division is done by r instead of 2.

Convert $(0.6875)_{10}$ to binary.

$0.6875 \times 2 = 1.3750$: 1
$0.3750 \times 2 = 0.7500$: 0
$0.7500 \times 2 = 1.5000$: 1
$0.5000 \times 2 = 1.0000$: 1

$(0.6875)_{10} = (.1011)_2$

To convert a decimal fraction to a number expressed in base-r, a similar procedure is used. However, multiplication is by r instead of 2, and the coefficients found from the integers may range in value from 0 to r-1 instead of 0 and 1.

convert decimal $(41.6875)_{10}$ to binary.

$(41.6875)_{10} = (101001.1011)_2$

Complement:-

There are two methods for find out complement-

① Diminished Radix Complement. (r's-1)
② Radix Complement (r's).

Complements are often used in digital computers to simplify the subtraction operation.

① Diminished Radix Complement (r's-1) :—

$(547600)_{10}$

$r = 0$

$\boxed{r's-1 = (r^n - 1) - N}$

n = no. of total Digit
n = 6
N = 547600
r = 10

$(10-1) = (10^6 - 1) - 547600$

~~9's~~ $= 1000000 - 547601$

9's Complement $= 999999 - 547600$

$\boxed{9's \text{ Comp.} = 452399}$ Ans

This complement is also known as 1's Complement and 9's Complement.

~~Let r = 2~~

$\Rightarrow (10101101)_2$

$r = 2, n = 8, N = 10101101$

$(2's-1) = (2^8 - 1) - 10101101$

$= (255)_{10} - 10101101$

$= 11111111 - 10101101$

$= 01010010$

	r's Comp	(r-1)'s Comp
r=10	10's Comp	9's Comp
r=2	2's	1's
r=8	8's	7's
r=16	16's	F's

② Radix Complement (r's) :— $\boxed{r's = [(r^n - 1) - N] + 1} = \boxed{r's = r^n - N}$

$(1234)_{10}$ $r = 10, n = 4, N = 1234$

10's $= [(10^4 - 1) - 1234] + 1$

$= (9999 - 1234) + 1$

$= 8765 + 1$

$\boxed{10's = 8766}$

This complement is also known as 2's Complement and 10's complement.

7's Comp. of Octal No. 5674.

N = 5674, n = 4, r = 8

$= (8^4 - 1) - 5674$

$(4096-1)_{10} \rightarrow (7777)_8$

$= 7777 - 5674$

$= 7777 - 5674$

7's = 2103 Ans

Simply we can do this

7's = 7777 - 5674

7's = 2103 Ans

8's Comp of Octal No. 5674.

7's Comp = 2103

8's Comp = 7's Comp + 1

= 2103 + 1

8's Com = 2104 Ans

1's Comp of 1101

1's = 1111 - 1101

1's = 0010 Ans

76.34 in 10's Comp.

n = 2 (76), r = 10, N = 76.34

$= 10^2 - 76.34$

= 23.66

#

Subtraction with r's Complement:-

unsigned data :- $M_1 = 1025$ (Minued) $N_1 = 50$ (Subtrahend) $D_1 = M - N$

Singed data :- $M_2 = -370$ $N_2 = 4312$ $D_2 = M - N$

Step 1:- Equale the length

$M_1 = 1025$ $N_1 = 50 \rightarrow 0050$

$M_2 = -370 \rightarrow -0370$ $N_2 = 4312$

Step 2:- Represent Negative Operands by taking r's complement of -ve operands

$M_2 = -370$

r's of M_2 (r = 10)

$= 9630$

$N_2 = 4312$

$$\begin{array}{r} 9999 \\ -0370 \\ \hline 9629 \\ +1 \\ \hline 9630 \end{array}$$

Step 3 → Complement the subtrahend

$N_1 = 50$

r's of $N_1 = 9950$

$9999 - 0050 = 9949 + 1 = 9950$

r = 10

$N_2 = 4312$

r = 10 r's of $N_2 = 5688$

$9999 - 4312 = 5687 + 1 = 5688$

Step 4:- Addition and Carry:-

$M_1 = 1025$ $N_1 = 9950$

$D_1 = 1025 + 9950$

$= 10975$

↑ Carry

$M_2 = 9630$ $N_2 = 5688$

$D_2 = 15318$

↑ Carry

with the help of 3 bit 8 values ($2^3 = 8$) can be represented however because of repetation of -0 signed Magnitude and 1's complement will represent one value less than the possible no. of 8.

0 has got multiple representation in SNM and 1'sC

0 has got its unique representation in 2's C, and because of repetation you've can represent only 7 values in SNM and 1's C.

Signed Magnitude

Binary Subtraction using 1's Complement :-

Step1 ⇒ Convert number to be subtracted to it's 1's complement form.

Step2 ⇒ Perform the addition.

Step3 ⇒ If the final carry is 1, then add it to the result obtained in step 2. If final carry is 0, result obtained in step2 is negative and its in the 1's complement form.

$A - B = A + (-B)$

A - B = A + B's 1's Complemt

⇒ If carry is 1 then
$A + (-B) = S$
$+1$
Result Ans

This process is called End-around carry

⇒ If carry is 0
$A + (-B) = S$ Ans

Ex:- Perform $(1100)_2 - (0101)_2$

A = 1100 ; B = 0101

1's of B = 1010

A - B = A + (1's of B)

= 1100 + 1010

= 10110 ... 7 ✗

0110
+1
0111

A - B = 0111 (7 → 0111)

A = 1100 (12)
B = 0101 (5)
12 - 5 = 7

Ex:- Perform $(0101)_2 - (1100)_2$

A = 0101 B = 1100

(A - B)

1's of B = 0011

A - B = A + (1's of B)

= 0101 (5)
+ 0011 (12)
1000 ← -ve

A=5
B=12
5+12=7

∴ Invert this result because it is in 1's Com then,

A - B = -0111 → (-7)

Decimal Subtraction Using 10's Complement:—

Case 1:- Carry is produced—

Ex:- $(2058)_{10} - (1729)_{10} = ?$

$X = 2058$

$Y = 1729$

$X - Y = X + Y$ का 10's

9's of Y = 9999 − 1729 = 8270

10's of Y = 8270 + 1 = 8271

10's of Y = 8271

X − Y = 2058 + 8271 = 10329

↓ discard the carry then

X − Y = 0329

Case 2:- No Carry—

Ex:- $(7398 - 9207)_{10} = ?$

$X = 7398$

$Y = 9207$

X − Y = X + (10's of Y)

9's of Y = 9999 − 9207 = 0792

10's of Y = 0792 + 1 = 0793

X − Y = 7398 + 0793 = 8191

In this step we don't have any carry so calculate 10's complement of the sum.

10's of 8191 = 9999 − 8191 = 1808 — 9's

10's of 8191 = −1809

Binary Codes:- An n bit binary code is a group of n bits that assumes up to 2^n distinct combinations of 1's and 0's with each combination representing one element of the set that is being coded. A set of four elements can be coded with two bits, with each element assigned one of the following bit combinations: 00, 01, 10, 11. A set of eight elements requires a three-bit code and a set of 16 elements requires a four-bit code. The bit combination of an n-bit code is determined from the count in binary from 0 to $2^n - 1$. Each element must be assigned a unique binary bit combination, and no two elements can have the same value: otherwise, the code assignment will be ambiguous.

BCD:- A number with K decimal digits will require 4K bits in BCD. Decimal 396 is represented in BCD with 12 bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit. A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9. A BCD number greater than 10 looks different from its equivalent binary number even though both contains 0's and 1's. Moreover, the binary combinations 1010 through 1111 are not used and have no meaning in BCD. Consider decimal 185 and its corresponding value in BCD and binary:

$$(165)_{10} = (0001\ 0110\ 0101)_{BCD} = (10100101)_2$$

The BCD value has 12 bit to encodes the characters of the decimal value, but the equivalent binary number needs only 8 bit. It is obvious that the representation of a BCD number needs more bits than its equivalent binary value.

It is important to realize that BCD numbers are decimal numbers and not binary numbers, although they use bits in their representation. The only difference between a decimal number and BCD is that decimals are written with the symbols 0, 1, 2 ... 9 and BCD numbers use the binary code 0000, 0001, 0010 ... 1001. The decimal value is exactly the same.

ASCII Character Code!- It includes 128 characters.

ASCII - American Standard Code for Information Interchange

Uppercase letters (A through Z) - 26
Lowercase letters (a through z) - 26
Numbers (0-9) - 10
Special characters - 66

Error Detection Code!- (Only for single bit error)

Odd Parity		Even Parity	
Message	Parity	Message	Parity
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1

In single bit error only one bit of data unit is changed from 0 to 1 or 1 to 0.

Parity :- It is an extra bit that is sent with the message signal. It detects single bit Error.

Boolean Algebra and Logic Gates

~~Boolean Al~~

Algebric Structure :- A set associated with binary operation.

Closure property not hold for natural No. It holds only for Integers.

$(N,+)$ ✗ $(N,-)$ ✗ $(I,-)$ ✓ , $(I,+)$ ✓

Associativity :- A binary operator $*$ on a set S is said to be associative when $\boxed{a*(b*c)=(a*b)*c}$ for all $a,b,c \in S$

$\Rightarrow (a+b)+c = a+(b+c)$

$\Rightarrow (a*b)\times c = a\times(b\times c)$

$\Rightarrow a-(b-c) \neq (a-b)-c$

$\hookrightarrow a-b+c \neq a-b-c$

Commutative Property :-

$\boxed{a*b=b*a}$

$\Rightarrow a+b=b+a$

$\Rightarrow a\times b=b\times a$

$\Rightarrow a-b \neq b-a$

Existance of Identity :-

$\boxed{e*x=x*e=x}$

$\Rightarrow 0+x=x+0=x$

$\Rightarrow 1\times x=x\times 1=x$

Distributive Law :-

$\boxed{a*(b*c)=(a*b)\cdot(a*c)}$

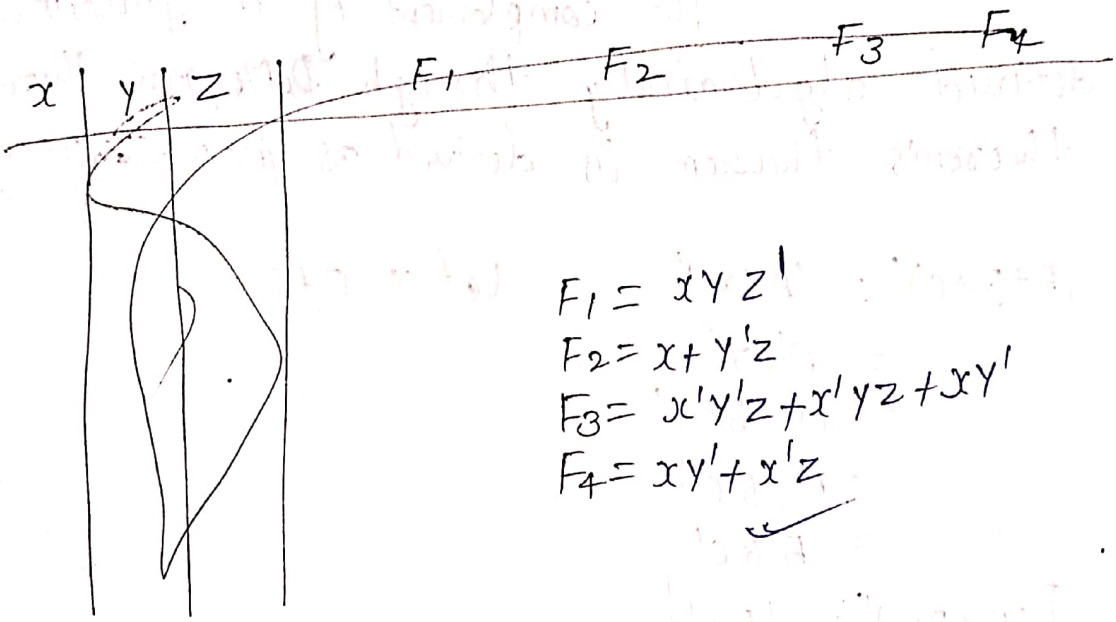
$\Rightarrow a\times(b+c)=(a\times b)+(a\times c)$

$\Rightarrow a+(b\times c)\neq(a+b)\times(a+c)$

(S, +, *), field

An algebric structure, which the properties of closor accosiativity, commutative, identity, inverse and distributive hold is called a field.

Boolean function:- Boolean function is a function of boolean variable.



$F_1 = xyz'$

$F_2 = x + y'z$

$F_3 = x'y'z + x'yz + xy'$

$F_4 = xy' + x'z$

x	y	z	F1	F2	F3	F4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

Complement of a function:-

The complement of a function F is F' and is obtained from the interchange of 0's to 1's and 1's to 0's.

Canonical and Standard form for representing function:

Maxterms and Minterms

The complement of a function may be derived algebraically through DeMorgan's theorem. DeMorgan's theorem's theorem is derived as follows:-

$$(A+B+C)' = (A+x)' \qquad \text{Let } x = B+C$$
$$= A'x'$$
$$= A'(B+C)'$$
$$= A'(B'C')$$
$$= A'B'C'$$

$$\boxed{(A+B+C)' = A'B'C'}$$

Demorgan's theorems for any number of variables resemble the two-variable case in form and can be derived by successive substitutions similar to the method used in the preceding derivation. These theorems can be generalized as follows:-

$$\boxed{(A+B+C+D+\cdots+F)' = A'B'C'D'\cdots F'}$$

$$\boxed{(ABCD\cdots F)' = A'+B'+C'+\cdots+F'}$$

Boolean Algebra:- Boolean algebra is a mathematical system with a set of elements, a set of operations and a number of unproved postulates.

Rules for Boolean Algebra :-

1. $x + 0 = x$
2. $x + x' = 1$
3. $x + x = x$
4. $x + 1 = 1$
5. $(x')' = x$
6. $x + y = y + x$
7. $x + (y + z) = (x + y) + z$
8. $x(y + z) = xy + xz$
9. $(x + y)' = x'y'$
10. $x + xy = x$
11. $x \cdot 1 = x$
12. $x \cdot x' = 0$
13. $x \cdot x = x$
14. $x \cdot 0 = 0$
15. $xy = yx$
16. $x(yz) = (xy)z$
17. $x + yz = (x + y)(x + z)$
18. $(xy)' = x' + y'$
19. $x(x + y) = x$

$x + x = x$

Proof

$(x + x) \cdot 1 = (x + x) \cdot (x + \bar{x})$

$= x \cdot x + x\bar{x} + x \cdot x + x\bar{x}$

$= x + x$

$= x$

$x + xy = x$

$x + xy = x(1 + y)$

$= x(y + 1)$

$= x \cdot 1$

$= x$

$\Rightarrow x(x + y)$

$= x \cdot x + xy$

$= x + xy$

$= x \cdot 1 + xy$

$= x \cdot (y + y') + xy$

$= xy + xy' + xy$

$= xy + xy'$

$= x(y + y') = x \cdot 1 = x$

Canonical And Standard form for Representing function

Minterms and Maxterms :- A binary variable may appear either in its normal form (x) or in its complement form (x'). Now consider two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations :- x'y', x'y, xy' and xy. Each of these four AND terms is called a minterm or a standard Product. In a similar manner, n variables can be combined to form 2^n minterms. A symbol for each minterm is also shown in the table and is of the form m_j, where the subscript j denotes the decimal equivalent

of the binary number of the minterm designated.

In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms or standard sums. For n variables total no. of maxterms will be 2^n similarly. Each maxterm is obtained from an OR term of the n variables with each variable being unprimed if the corresponding bit is 0 and primed if a 1.

Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1.

			Minterms (SOP)		Maxterms (POS)	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

0 — x'
1 — x
minterm

Maxterm
0 — x
1 — x'

xy { $x'y'$, $x'y$, xy', xy } — minterm

$2^{2-\text{No. of variable}}$

$x+y$ { $x+y$, $x+y'$, $x'+y$, $x'+y'$ } — Maxterm

$2^{2-\text{No. of Variable}}$

In Case of n variable total minterm $= 2^n$

In case of n variable total Maxterm $= 2^n$

functions of Three Variables :—

x	y	z	function f_1	function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all these terms.

exp:- function f_1 is determined by expressing the combinations 001, 100 and 111 as $x'y'z$, $xy'z'$ and xyz respectively. Since each ~~of the result~~ one of these minterms result in $f_1 = 1$.

Sum of product form (Sum of minterm) :—

$$f_1 = x'y'z + xy'z' + xyz$$

$$\boxed{f_1 = m_1 + m_4 + m_7}$$

$$= \Sigma(1, 4, 7)$$

~~$= \Sigma(m_1, m_4, m_7)$~~

$$f_2 = x'yz + xy'z + xyz' + xyz$$

$$\boxed{f_2 = m_3 + m_5 + m_6 + m_7}$$

$$\boxed{f_2 = \Sigma(3, 5, 6, 7)}$$

Product of Sum form:-

$f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$

$f_1 = M_0 M_2 M_3 M_5 M_6$

$f_1 = \pi(0,2,3,5,6)$

$f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$

$f_2 = M_0 M_1 M_2 M_4$

$f_2 = \pi(0,1,2,4)$

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in a canonical form.

Sum of Minterms (Sum-of-Product Terms):-

Express the Boolean function $F = A + B'C$ as a sum of minterms. The function has three variables: A, B and C.

$F = A + B'C$

$= A \cdot 1 + 1 \cdot B'C$

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

$= AB + AB' \neq AB'C + A'B'C$

$= AB(C+C') + AB'(C+C') + AB'C + A'B'C$

$= ABC + ABC' + \underline{AB'C} + AB'C' + \underline{AB'C} + A'B'C$

If any term appears twice remove one of those,

$= ABC + ABC' + AB'C + AB'C' + A'B'C$

$= m_7 + m_6 + m_5 + m_4 + m_1$

$F = m_1 + m_4 + m_5 + m_6 + m_7$

$F = m_1 + m_4 + m_5 + m_6 + m_7$

$F = \Sigma(1,4,5,6,7)$

Product of Maxterms (Product-of-Sum Terms):-

Express the Boolean function $F = xy + x'z$ as a product of maxterms.

$$F = xy + x'z$$
$$= A + x'z$$
$$= (A + x')(A + z)$$
$$= (xy + x')(xy + z)$$
$$= (x + x')(y + x')(x + z)(y + z)$$
$$= (y + x')(x + z)(y + z) \quad \text{— (1)}$$

$x + z$
$= x + z + 0$
$= x + z + yy'$
$= A + yy'$
$= (A + y)(A + y')$
$= (x + z + y)(x + z + y')$
$= (x + y + z)(x + y' + z)$

$y + z$
$= y + z + 0$
$= y + z + xx'$
$= A + xx'$
$= (A + x)(A + x')$
$= (y + z + x)(y + z + x')$
$= (x + y + z)(x' + y + z)$

$x' + y$
$= x' + y + 0$
$= x' + y + zz'$
$= A + zz'$
$= (A + z)(A + z')$
$= (x' + y + z)(x' + y + z')$

Putting these value in (1),

$$\Rightarrow \frac{(x' + y + z)(x' + y + z')(x + y + z)}{(x + y' + z)(x + y + z)(x' + y + z)}$$

$$= (x' + y + z)(x' + y + z')(x + y + z)(x + y' + z)(x' + y + z)$$
$$= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z')$$

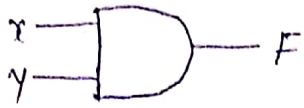
$$\boxed{F = M_0 M_2 M_4 M_5}$$

$$\boxed{F = \Pi(0, 2, 4, 5)}$$

Digital Logic Gates:-

Name	Graphic symbol	Algebraic fun^n	Truth Table

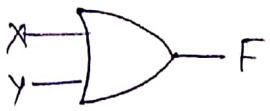
AND



$F = xy$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

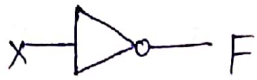
OR



$F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

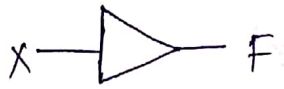
Inverter



$F = x'$

x	F
0	1
1	0

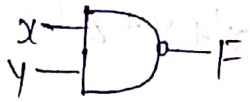
Buffer



$F = x$

x	F
0	0
1	1

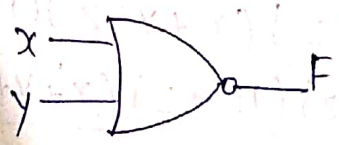
NAND



$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

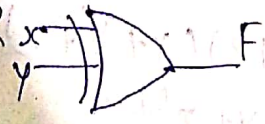
NOR



$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive-OR (XOR)



$F = xy' + x'y$
$= x \oplus y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR or equivalence

$F = xy + x'y'$
$= (x \oplus y)'$

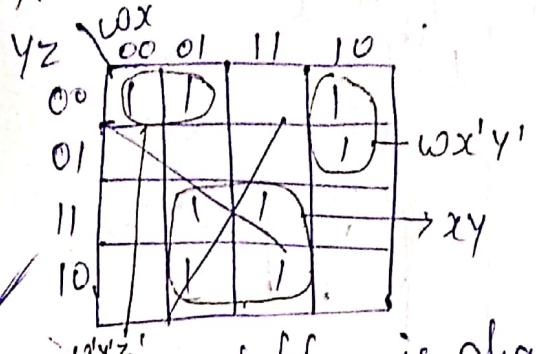
x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Gate Level Minimization

Example of K-Map:-

minimize-

$f(w,x,y,z) = \Sigma(0,4,6,7,8,9,15)$

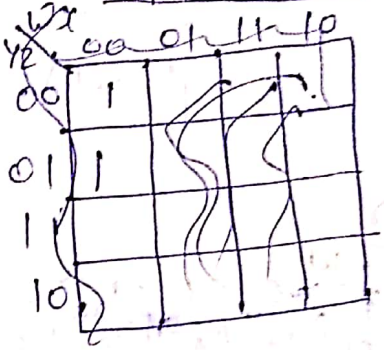


0 ⇒ w x y z
0 0 0 0

4 ⇒ w x y z
0 1 0 0

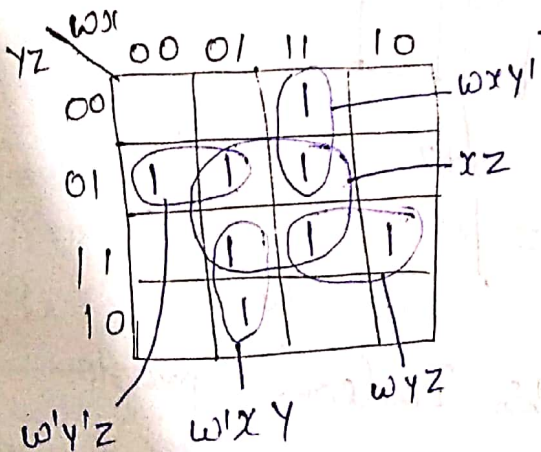
	w x y z
0	0 0 0 0
4	0 1 0 0
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
15	1 1 1 1

If any variable is change we skip that variable and if any variable not changes we just write them.



$$= w'y'z' + xy + wx'y'$$

$f(w,x,y,z) = \Sigma(1,5,6,7,11,12,13,15)$



	w x y z
1	0 0 0 1
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
15	1 1 1 1

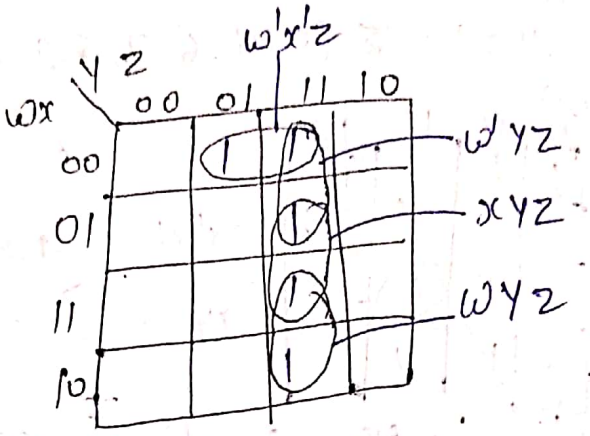
$$= w'y'z + wxy + wyz + wxy' + xz$$

The kmap is a graphical representation that provides a systematic method for simplifying the Boolean Expression.

Simplify the boolean function-

$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$

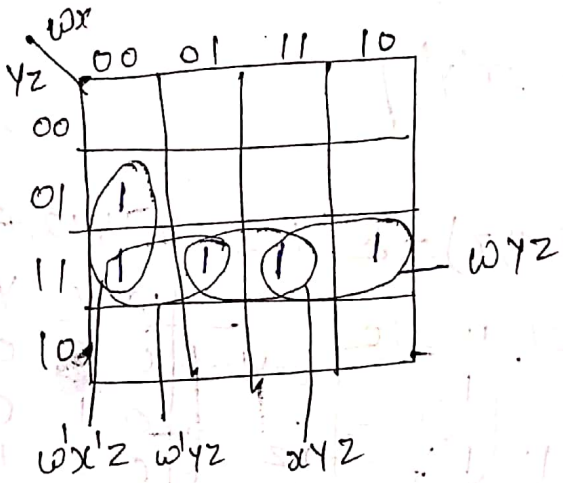
wx \ yz	00	01	11	10
00	m_0	m_1 1	m_3 1	m_2
01	m_4	m_5	m_7 1	m_6
11	m_{12}	m_{13}	m_{15} 1	m_{14}
10	m_8	m_9	m_{11} 1	m_{10}



$\therefore$ $F(sop) = w'x'z + w'yz + xyz + wyz$

OR

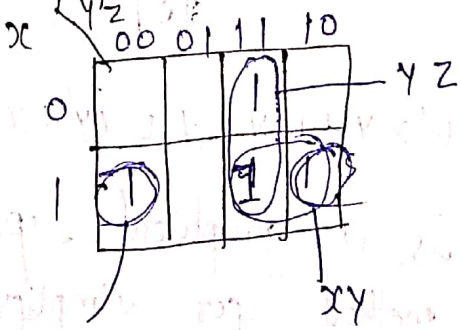
yz \ wx	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}



$F(sop) = w'x'z + w'yz + xyz + wyz$

$F(x, y, z) = \Sigma(3, 4, 6, 7)$ Simplify this Boolean Expression.

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

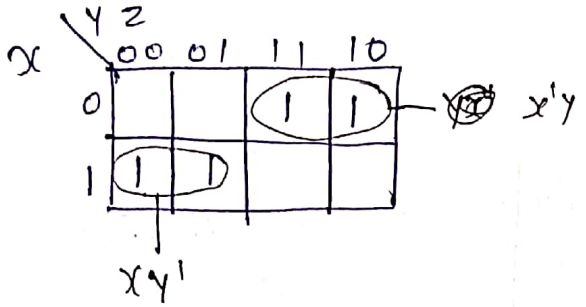


yz + xz

Simplify the boolean function $F(x,y,z) = \Sigma(2,3,4,5)$

First, 0 1 is marked in each minterm that represents the function

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6



$$F_{(SOP)} = x'y + xy'$$

~~Karnaugh map or K-map:-~~

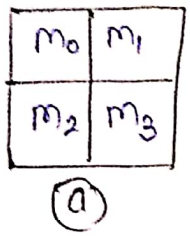
Map Method:- The map method presented here provides a simple, straightforward procedure for minimizing Boolean function. This method may be regarded as a pictorial form of a truth table. The map method is also known as the Karnaugh map or K-map.

A K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized. Since any Boolean function ~~is recognized graphically in the map from the~~ can be expressed as a sum of minterms, it follows that a Boolean function is recognized graphically in the map ~~form~~ from the area enclosed by those squares whose minterms are included in the function. In fact the map presents a visual diagram

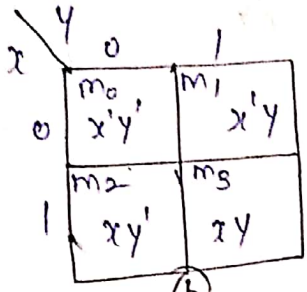
of all passible ways a function may be expressed in standard form, by recognizing various patterns.

The simplified expressions produced by the map are always in one of the two standard forms: sum of products or product of sums.

Two-Variable map :-

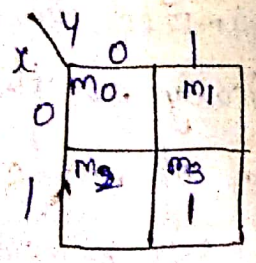


(a)

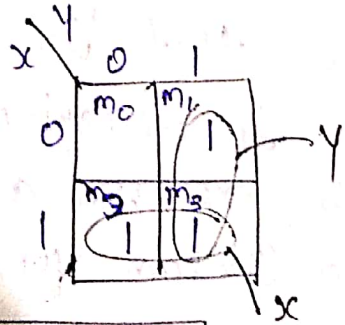


(b)

The two variable map is shown in the figure. There are four minterms for two variables, Hence the map consists of four squares, one for each minterm. The map is redrawn in (b) to shown the relationship between the squares and the two variables x and y. The 0 and 1 marked in each row and column designate the value of variables. The values x appears primed in row 0 and unprimed in row 1. Similarly, y appears primed in column 0 and unprimed in column 1.



$F = xy$



$F = x + y$

Note:- There will be 2^n minterms for n variable.

Three-variable Map!—

A three variable map is shown in the figure. There are eight minterms for three binary variables; therefore, the map consists of eight squares. The minterms are arranged, not in a binary sequence, but in a sequence similar to Gray code. The characteristics of this sequence is that only one bit changes in value from one adjacent column to the next. The map drawn in part (b) is marked with numbers in each row and each column to show the relationship between the squares and the three variables.

For exp!— The square assigned to m_5 corresponds to row 1 and column 01. When these two numbers are concatenated, they give the binary number 101, whose decimal equivalent is 5.

To understand the usefulness of the map in simplifying Boolean functions, we must recognize the basic property possessed by adjacent squares: ~~in the maps differ~~ by only one variable. Any two adjacent squares in the map differ by only one variable, which is primed in one square and unprimed in the other.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	

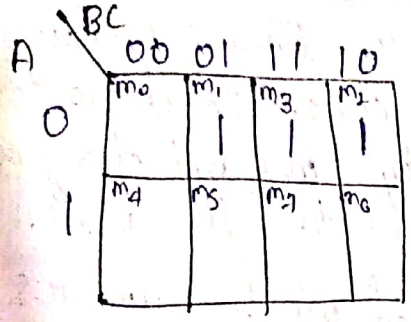
(b)

⇒ Let the Boolean function $F = A'C + A'B + AB'C + BC$

(a) Express this function as a sum of minterms.

(b) Find the minimal sum-of-products expression.

$F = A'C + A'B + AB'C + BC$



Four Variable Maps:—

The map for Boolean function of four binary variables is shown in the figure, are listed the 16 minterms and the squares assigned to each. In (b) the map is redrawn to show the relationship between the squares and the four variables. The rows and columns are numbered in a Gray code sequence, with only one digit changing value between two adjacent rows or columns. The minterm corresponding to each square can be obtained from the concatenation of the row number with the column number.

for example:- The number of the third row (11) and the second column (01), when concatenated, give the binary number 1101, the binary

equivalent of decimal 13. Thus, the square of in the third row and second column represents min term m_{13}.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

wx \ yz	00	01	11	10
00	m_0 $w'x'y'z'$	m_1 $w'x'y'z$	m_3 $w'x'yz$	m_2 $w'x'yz'$
01	m_4 $w'xy'z'$	m_5 $w'xy'z$	m_7 $w'xyz$	m_6 $w'xyz'$
11	m_{12} $wxy'z'$	m_{13} $wxy'z$	m_{15} $wxyz$	m_{14} $wxyz'$
10	m_8 $wx'y'z'$	m_9 $wx'y'z$	m_{11} $wx'yz$	m_{10} $wx'yz'$

The following terms can be defined with respect to K-map simplification:-

1) Pair:- Group Group of two adjacent minterms (m_4, m_5). A pair eliminates one variable in output expression.

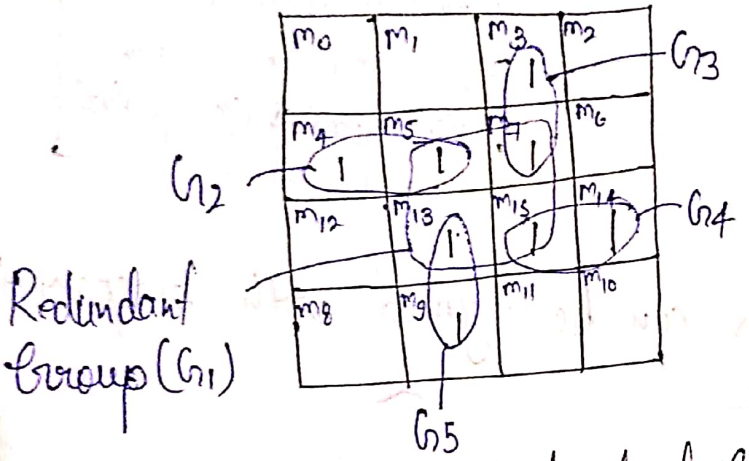
2. Quad:- Group of four adjacent minterms ($m_0 m_2 m_4 m_6$). A quad eliminates two variables in output expressions.

3. Octet:- Group of eight adjacent minterms ($m_0 m_1 m_4 m_5 m_8 m_9 m_{12} m_{13}$). An octet eliminates three variables in output expression.

4. Redundant group:- A redundant group is a group in which all the elements in this group are covered by some other groups.

Consider the example shown in figure. Here $m_5 m_7 m_{13} m_{15}$ form a quad (G1). However

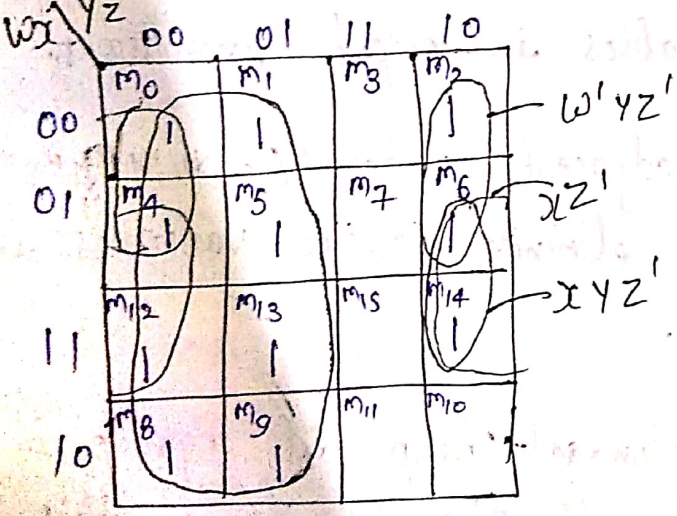
all elements of this group are covered by the adjacent groups. Minterm m_5 is covered by pair G_2, similarly minterm m_7 is covered by G_3, and so on. Hence, G_1 becomes a redundant group and final output expression contains only four product terms instead of five terms.



Map for Redundant Group

Simplify the Boolean function –

$F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$

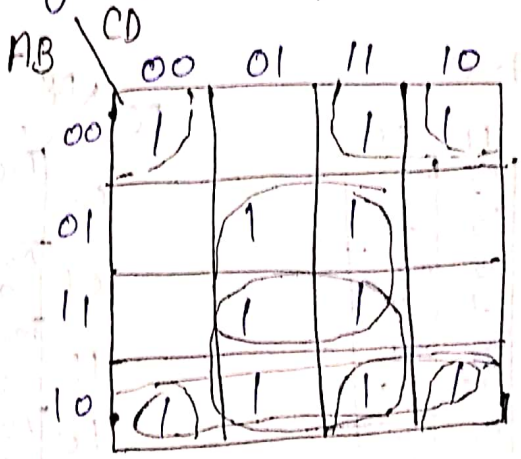


Prime Implicants:-

A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map. If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be essential prime implicant.

The prime implicants of a function can be obtained from the map by combining all possible maximum numbers of squares. This means that a single 1 on a map represents a prime implicant if it is not adjacent of any other 1's. Two adjacent 1's form a prime implicant, provided that they are not within a group of four adjacent squares.

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}



~~Ex~~ $F(A,B,C,D) = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$

Essential Prime Implicant = $BD + B'D'$

Prime Implicant = CD, $B'C$, AD, AB'

$F = BD + B'D' + CD + AD$

$F = BD + B'D' + CD + AB'$

$F = BD + B'D' + B'C + AB$

$F = BD + B'D' + B'C + AB'$

five-variable map:-

Maps more than four variables are not as simple to use as maps for four or fewer variables. A five-variable map needs 32 ($2^5 = 32$) squares and a six-variable map needs 64 squares.

The five-variable map is shown in figure. It consists of two four-variable maps with variable A, B, C, D and E. Variable distinguishes between the two maps as indicated at the top of diagram. The left-hand four variable map represent the 16 squares in which A=0, and the other four-variable map represents the squares in which A=1. Minterms 0 through 15 belong with A=0 and minterms 16 through 31 with A=1.

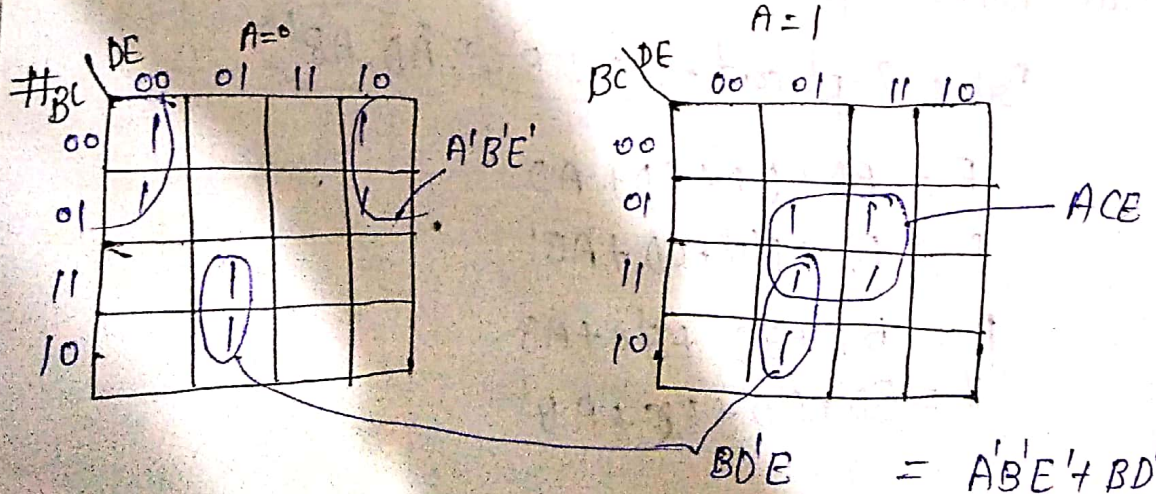
A=0

BC \ DE	00	01	11	10
00	m_0 0	m_1 1	m_3 3	m_2 2
01	m_4 4	m_5 5	m_7 7	m_6 6
11	m_{12} 12	m_{13} 13	m_{15} 15	m_{14} 14
10	m_8 8	m_9 9	m_{11} 11	m_{10} 10

A=1

BC \ DE	00	01	11	10
00	m_{16} 16	m_{17} 17	m_{19} 19	m_{18} 18
01	m_{20} 20	m_{21} 21	m_{23} 23	m_{22} 22
11	m_{28} 28	m_{29} 29	m_{31} 31	m_{30} 30
10	m_{24} 24	m_{25} 25	m_{27} 27	m_{26} 26

five-variable map



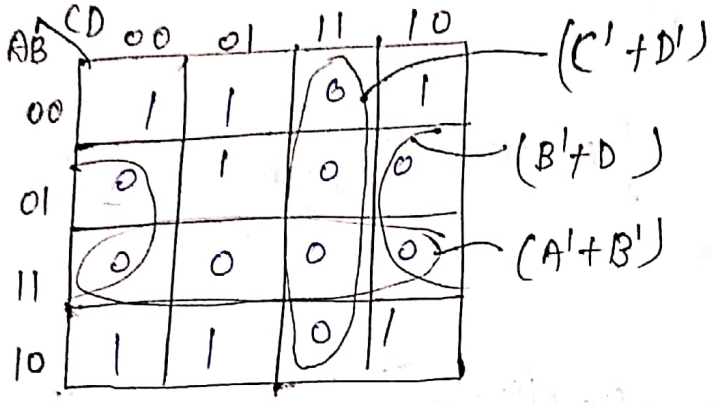
$= A'B'E' + BD'E + ACE$

Product of Sum Simplification:-

If ~~Sum~~ Product of Sum form is given $F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$ the find out Sum of product form.

$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$

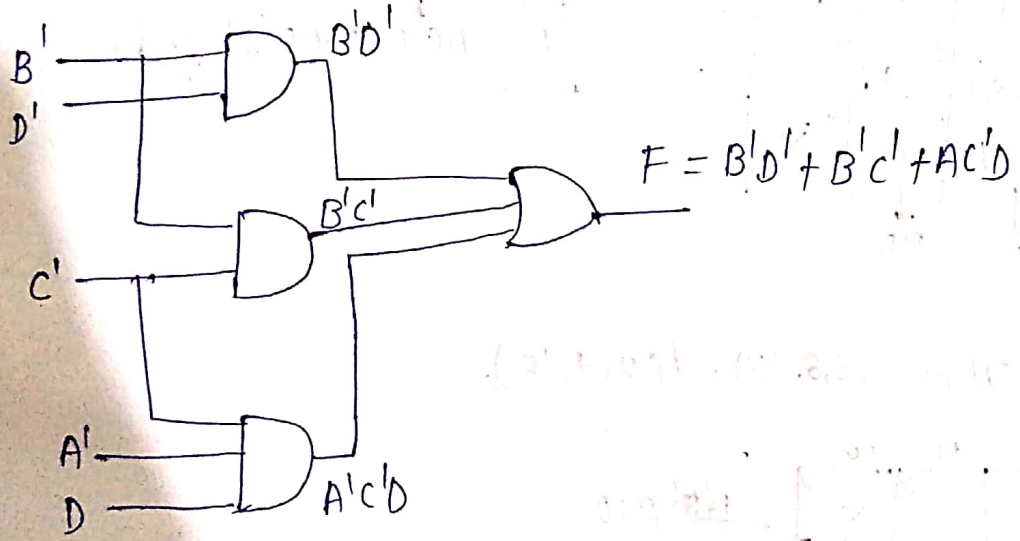
$F(A,B,C,D) = \pi(3,4,6,7,11,12,13,14,15)$



$F_{(POS)} = (C'+D')(B'+D)(A'+B')$

Gate Implementation:-

$F = B'D' + B'C' + A'C'D$



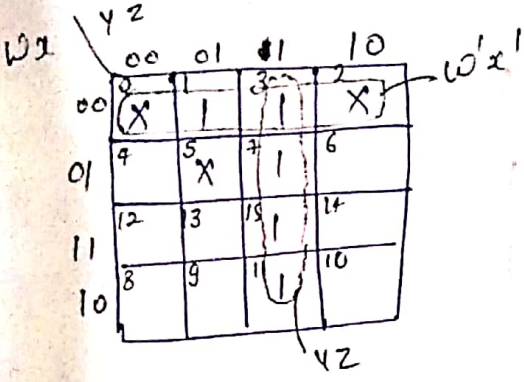
Don't-care Conditions:- Σ-SOP Π-POS

Simplify the Boolean function:—

$F(w,x,y,z) = \Sigma(1,3,7,11,15)$

which has the don't-care conditions

$d(w,x,y,z) = \Sigma(0,2,5)$

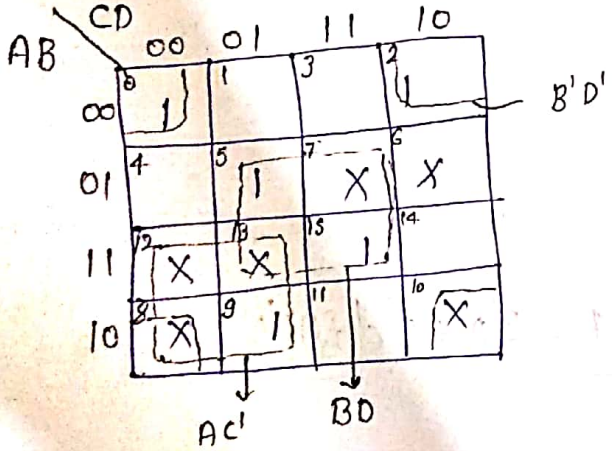


$F = YZ + w'x'$ Ans

Simplify the Boolean function:—

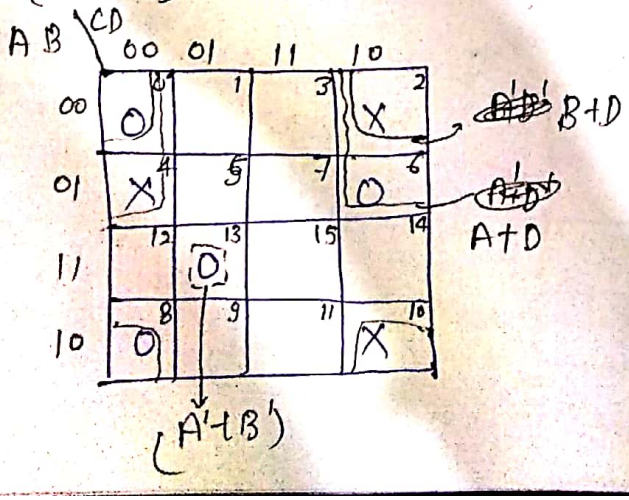
$F(A,B,C,D) = \Sigma m(0,2,5,9,15) + d(6,7,8,10,12,13)$

~~which has~~



$F = AC' + BD + B'D'$ Ans

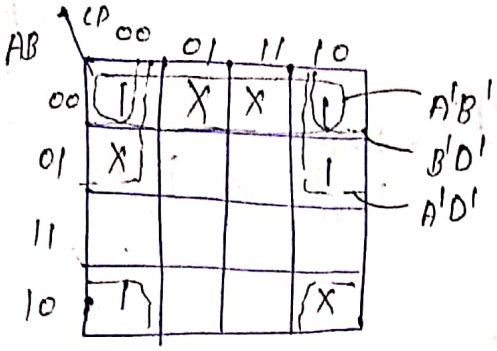
$F(A,B,C,D) = \Pi m(0,6,8,13) \cdot d(2,4,10)$.



~~$F = (A+D)(B+D)$~~

$F = (A+D)(B+D)(A'+B')$ Ans

$F(A,B,C,D) = \Sigma(0,2,6,8) + d(1,3,4,10)$



$F = A'B' + B'D' + A'D'$ Ans

The truth table represents the ---- function—

ⓐ x ✓
ⓑ x+y
ⓒ x⊕y
ⓓ y

x	y	f(x,y)
0	0	0
0	1	0
1	0	1
1	1	1

The simplified SOP of the boolean expression- $(P+Q'+R')(P+Q'+R)(P+Q+R')$ is

ⓐ $P'+Q+R'$
ⓑ $P+Q'R'$ ✓
ⓒ $P'Q+R$
ⓓ $PQ+R$

The given expression is in POS form

$(P+Q'+R')(P+Q'+R)(P+Q+R')$

↓ ↓ ↓

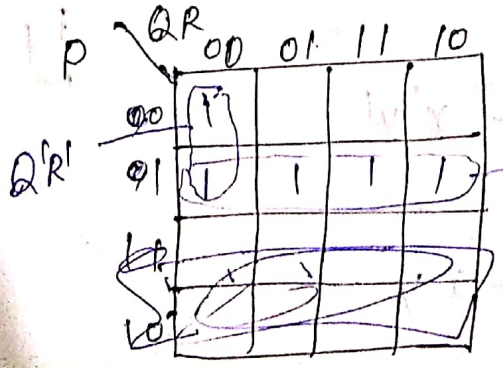
011 010 001

3 2 1

$\Pi(3,2,1)$

$\Sigma(0,4,5,6,7)$

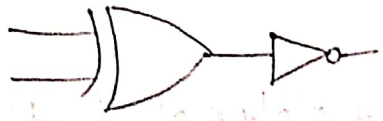
Total 8 element



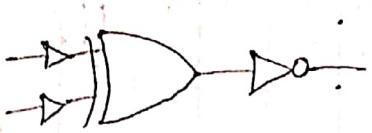
$F = P + Q'R'$ Ans

Which one of the following circles is not equivalent

(a)

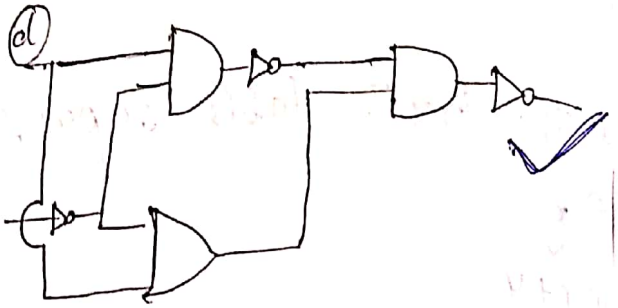


(b)

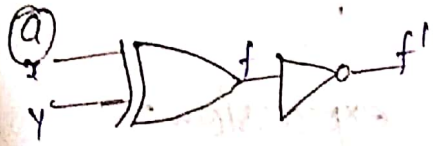


(c)

(d)



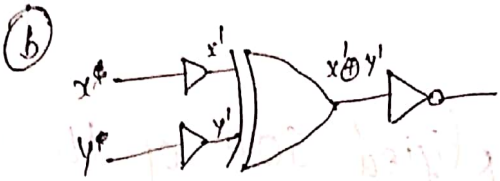
(a)



x	y	f	f'	
0	0	0	1	$x'y'$
0	1	1	0	
1	0	1	0	
1	1	0	1	xy

$f' = xy + x'y'$

(b)



$x' \oplus y'$

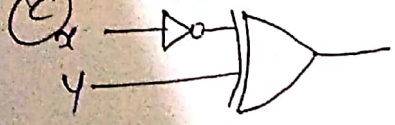
$= x'(y')' + (x')'y'$

$F = x'y + xy'$

x	y	x'	y'	EXOR $(x'y + xy')$	XNOR	
0	0	1	1	0	1	$x'y'$
0	1	1	0	1	0	
1	0	0	1	1	0	
1	1	0	0	0	1	xy

$F = xy + x'y'$

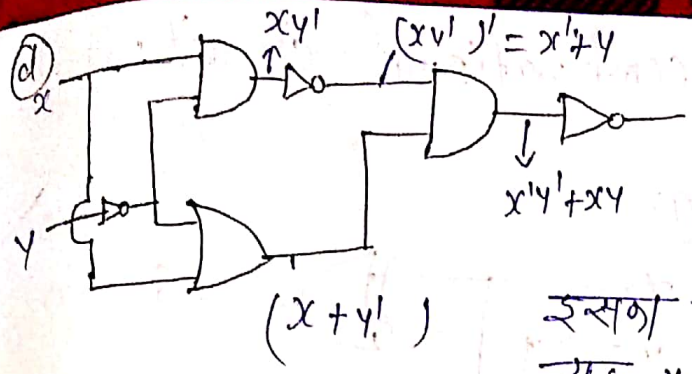
(c)



x	y	x'	XOR	
0	0	1	1	$x'y'$
0	1	1	0	
1	0	0	0	
1	1	0	1	xy

$F = xy + x'y'$

(d)



$(x'+y)(x+y')$
$= 0 + x'y' + xy + 0$
$= x'y' + xy$

यह XNOR का Output है जब इसका पुनः NOT निकाला जायेगा तो यह XOR Gate बन जायेगा अतः यह अन्य सभी से Different है।

The minterm expansion of $f(P,Q,R) = PQ + QR + PR's$

(a) $m_2 + m_4 + m_6 + m_7$ ✓

(b) $m_0 + m_1 + m_6 + m_7$

(c) $m_0 + m_1 + m_6 + m_7$

(d) $m_2 + m_3 + m_4 + m_5$

$PQ + QR' + PR'$
$= PQ(R+R') + QR'(P+P') + PR'(Q+Q')$
$= PQR + PQR' + PQR' + P'QR' + PQR' + PQ'R'$
$= PQR + PQR' + P'QR' + PQ'R'$
$= m_7 + m_6 + m_2 + m_4$
$= m_2 + m_4 + m_6 + m_7$ Ans

Logic Circuits

Logic Circuits

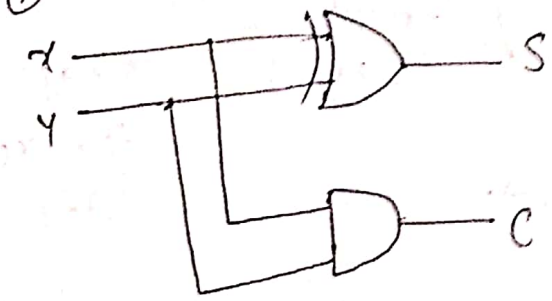
Combinational Ckt.	Sequential Ckt.
Present output depends on present input only.	Present output depends on previous output as well as recent input.

$A - B = A + (-B)$
$A \times B = A + A + A + \cdots$ B times
$\frac{A}{B} = A - B$

Half Adder:- It is a combinational circuit.

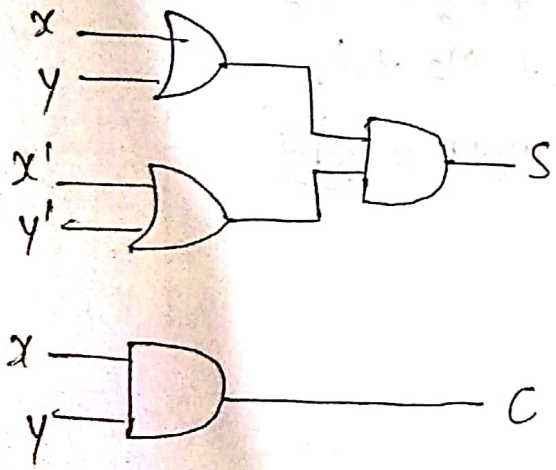
Truth Table

x	y	C	S
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

① 

② $C = xy$

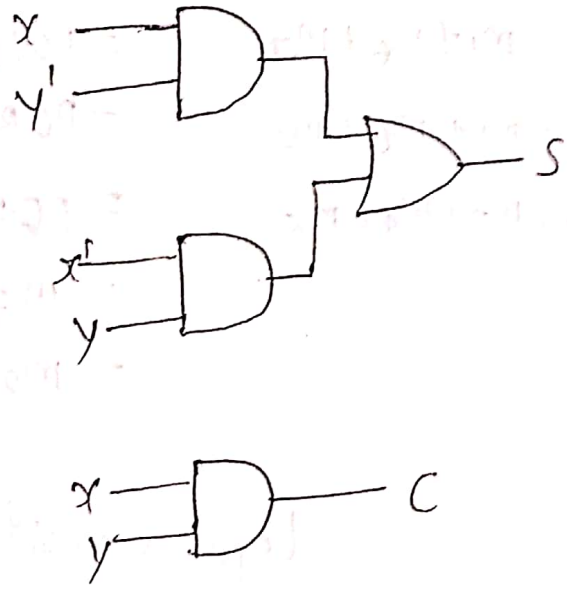
$S = x'y + xy'$



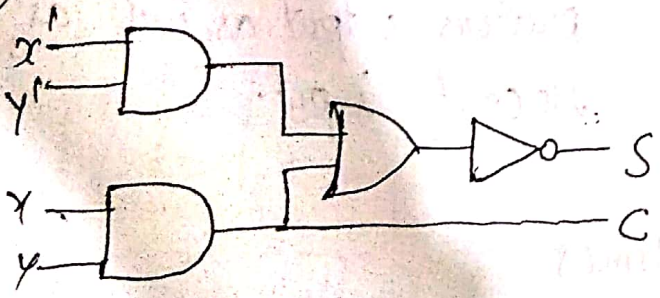
$S = (x+y)(x'+y')$

$= \underline{x}x' + xy' + yx' + \underline{y}y'$

$S = xy' + x'y$

③ 

$S = xy' + x'y$

④ 

$S = [(x'y' + xy)]'$

$= (x'y')' \cdot (xy)'$

$= [(x')' + (y')'] \cdot [x' + y']$

$= (x+y)(x'+y')$

$= xx' + xy' + x'y + yy'$

$S = xy' + x'y$

Full Adder :-

Truth Table

x	y	z	C	S	
0	0	0	0	0	m_0
0	0	1	0	1	m_1
0	1	0	0	1	m_2
0	1	1	1	0	m_3
1	0	0	0	1	m_4
1	0	1	1	0	m_5
1	1	0	1	0	m_6
1	1	1	1	1	m_7

$C = \Sigma(m_3, m_5, m_6, m_7)$

$C = x'yz + xy'z + xyz' + xyz$

$S = x'y'z + x'yz' + xy'z' + xyz$

$S = \Sigma(m_1, m_2, m_4, m_7)$

Carry

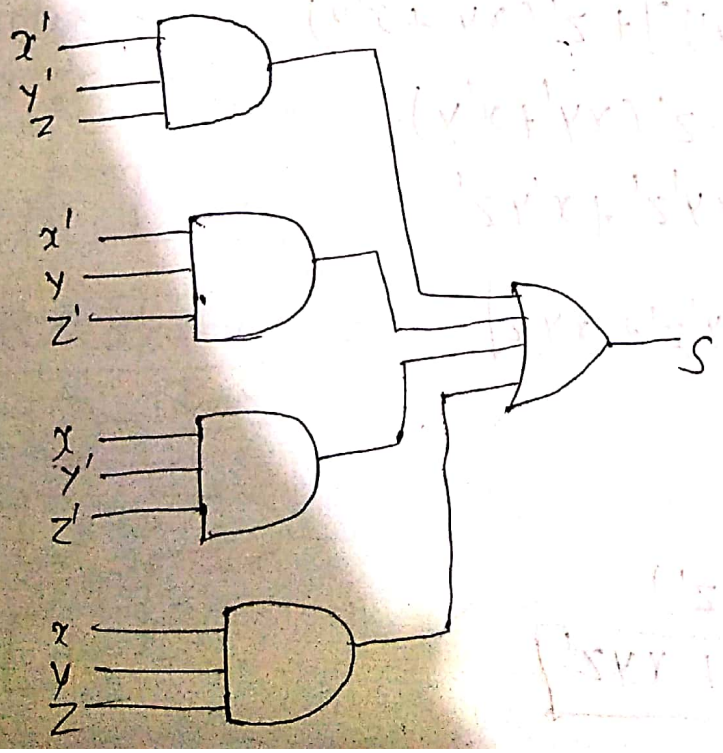
x \ yz	00	01	11	10
0			1	
1		1	1	1

$C = xy + xz + yz$

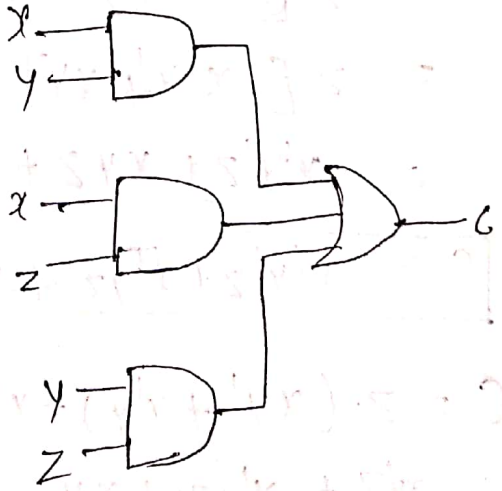
Sum

x \ yz	00	01	11	10
0		1		1
1	1		1	

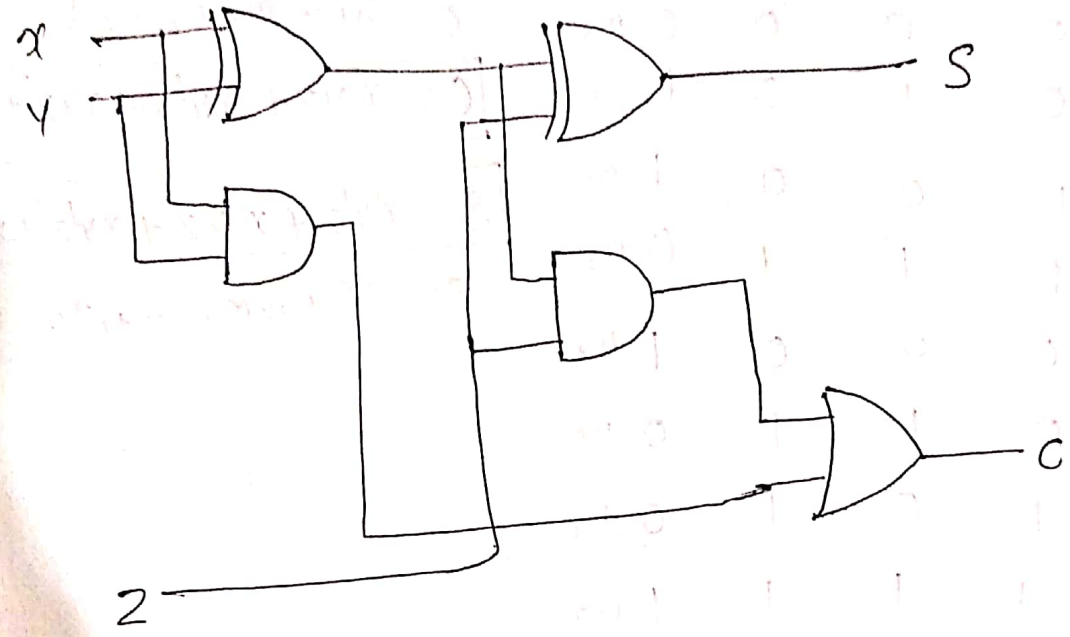
Sum



Carry



Implementing full Adder using half Adder:-



$S = z \oplus (x \oplus y) = z \oplus (xy' + x'y)$

A B

$A \oplus B$

$AB' + A'B$

$= z[(xy' + x'y)]' + z'(xy' + x'y)$

$= z[(xy')' \cdot (x'y)'] + z'(xy' + x'y)$

$= z[(x' + y)(x + y')] + z'(xy' + x'y)$

$= z[x'y' + xy] + z'(xy' + x'y)$

$= x'y'z + xyz + xy'z' + x'yz'$

$$\boxed{S = x'y'z + x'yz' + xy'z' + xyz}$$

$C = z(xy' + x'y) + xy$

$= xy'z + x'yz + xy$

$= xy'z + x'yz + xy(z + z')$

$$\boxed{C = xy'z + x'yz + xyz + xyz'}$$

Subtractor:-

① Half Subtractor:-

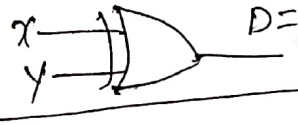
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$B = x'y$

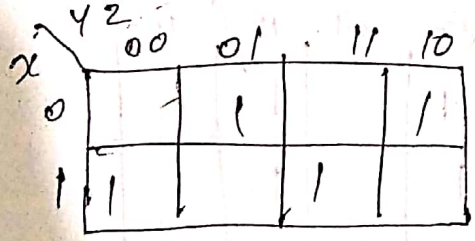
$D = x'y + xy'$

Ckt:-

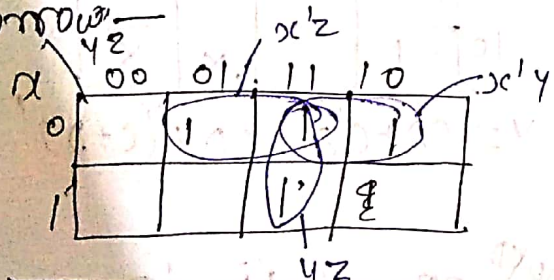
Borrow:-



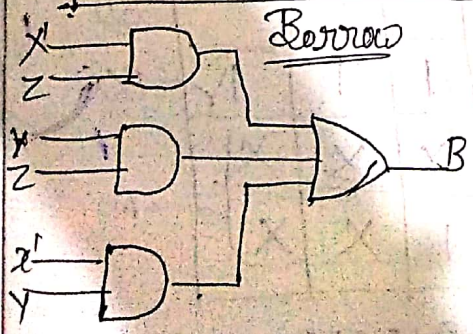
Difference:-



Borrow:-



$B = x'z + yz + x'y$



Full Subtractor:-

$x - y - z = x - (y + z)$

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

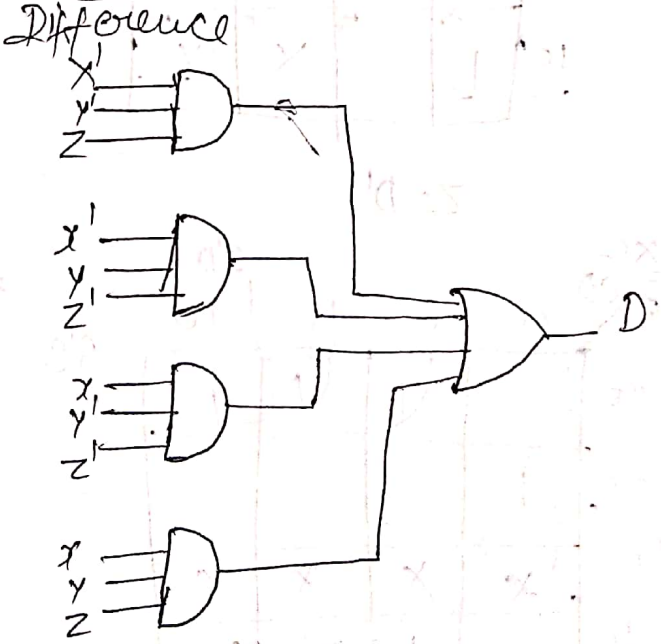
$B = x'y'z + x'yz' + x'yz + xyz$

$B = m_1 + m_2 + m_3 + m_7$

$D = m_1 m_2 + m_4 + m_7$

$D = x'y'z + x'yz' + xy'z' + xyz$

Difference

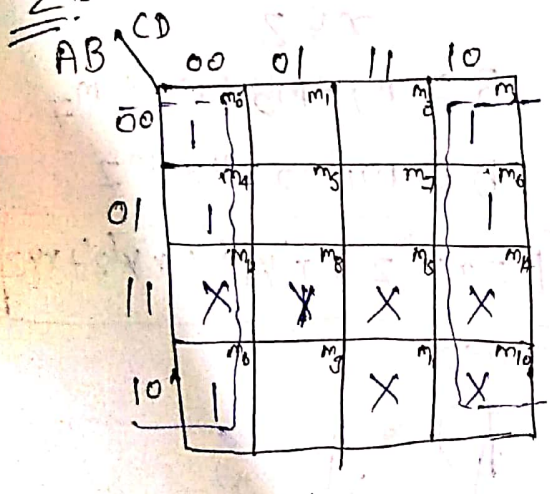


Code Conversion

Input BCD				Output Excess-3 Code				
A	B	C	D	W	x	y	Z	
0	0	0	0	0	0	1	1	m_0
0	0	0	1	0	0	0	0	m_1
0	0	1	0	0	0	0	1	m_2
0	0	1	1	0	0	1	0	m_3
0	1	0	0	0	1	0	0	m_4
0	1	0	1	0	1	0	1	m_5
0	1	1	0	0	1	1	0	m_6
0	1	1	1	0	1	1	1	m_7
1	0	0	0	1	0	0	0	m_8
1	0	0	1	1	0	0	1	m_9

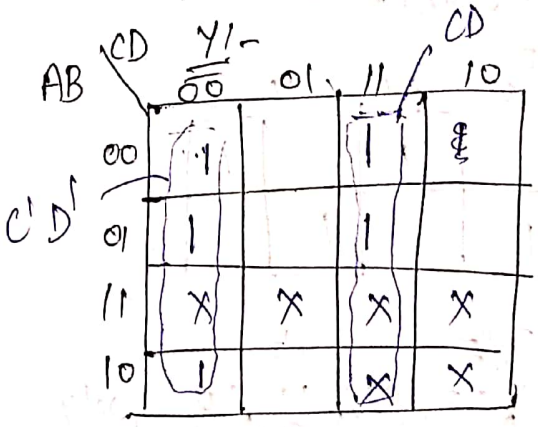
Z =

Z =



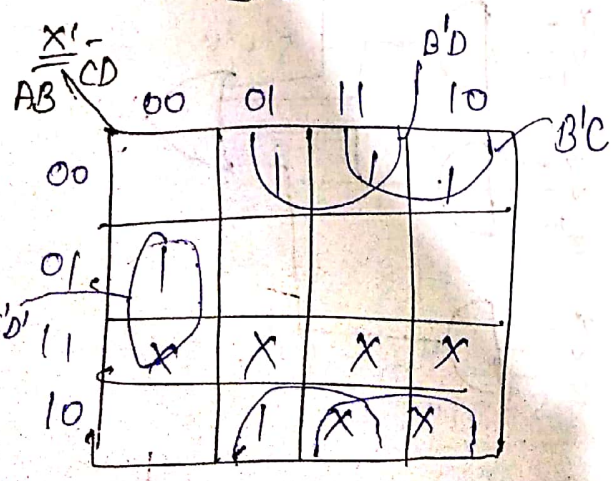
Z = D'

Y =



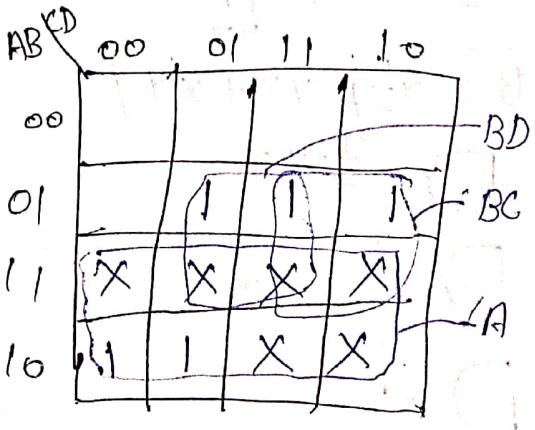
Y = CD + C'D' = CD + (C+D)'

X =



X = B'C + B'D + BC'D'

W =



W = A + BC + BD

$z = D'$

$Y = CD + C'D' = CD + (C+D)'$

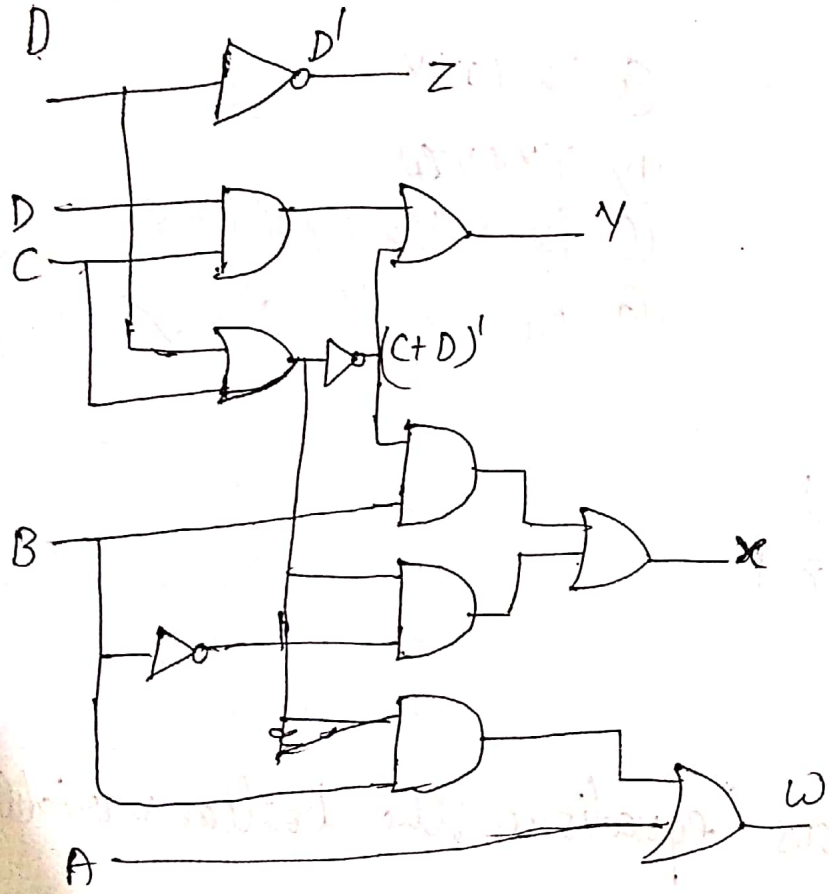
$x = B'C + B'D + BC'D'$

$= B'(C+D) + BC'D'$

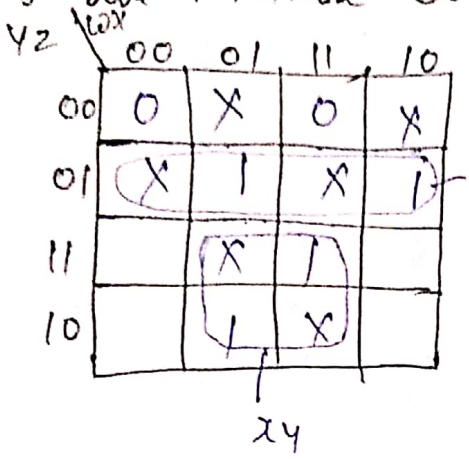
$= B'(C+D) + B(C+D)'$

$W = A + BC + BD$

$= A + B(C+D)$

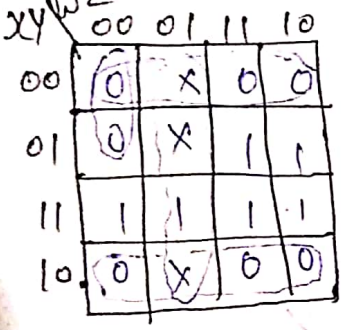


Q:- Given the following K-map which one of the following represents the minimal SOP of the map



(a) $xy+y'z$ ✓
(b) $wx'y'+xy+xz$
(c) $w'x+y'z+xy$
(d) $xz+y$

Q⇒ Which function does not implement the K-map given below -



(a) $(w+x)y$
(b) $xy+yz\bar{w}$
(c) $(w+x)(\bar{w}+y)(\bar{x}+y)$
(d) None of the above ✓

(c) $(w+x)(\bar{w}+y)(\bar{x}+y)$
$=(w+x)(y+\bar{w})(y+\bar{x})$
$=(w+x)(y+\bar{w}\bar{x})$

Q⇒ The simultaneous equations the boolean variables x, y, z, and w:-

$x+y+z=1$
$xy=0$
$xz+w=1$
$xy+\bar{z}\bar{w}=0$

having the following solution for x, y, z and w respectively:-

(a) 0100 (b) 1101 (c) 1011 ✓ (d) 1000

$\Sigma w = 0$

$z \neq 1$, $w \neq 1$

$z \neq 0$ $\quad w \neq 0$

Self dual function:- A boolean function is said to be self dual if and only if its dual is equivalent to the given function

$$f(x,y,z) = xy + yz + zx$$

$$f_d(x,y,z) = (x+y)(y+z)(z+x)$$

$xy + yz + zx$

$= xy(z+z') + yz(x+x') + zx(y+y')$

$= xyz + xyz' + xyz + x'yz + xyz + xy'z$

$= m_3 + m_5 + m_6 + m_7$

$= \Sigma(3,5,6,7)$

$(x+y)(y+z)(z+x)$

$= (x+y+zz')(y+z+xx') + (x+yy'+z)$

$= (x+y+z)(x+y+z')(x+y+z)(x'+y+z)(x+y+z)(x+y'+z)$

$= (x+y+z)(x'+y+z)(x+y'+z)(x+y+z')$

$= M_0 M_4 M_2 M_1$

$= M_0 M_1 M_2 M_4$

$= \Pi(0,1,2,4)$

$= \Sigma(3,5,6,7)$

Hence above function is self dual function

A boolean function is self dual if — 2^{n-1} (No. of variables)

(i) It is neutral (No. of minterms = No. of maxterms)

(ii) The function does not contain two mutually exclusive terms.

(a) $m_7 \rightarrow xyz$ $\quad xy'z - m_5$ $\quad$ (b) ये एक साथ नहीं रह सकते हैं
$m_0 - x'y'z'$ $\quad x'yz' - m_2$

इसमें से कोई एक लेना है।

$$\begin{cases} m_0 \neq m_7 \\ m_1 \neq m_6 \\ m_2 \neq m_5 \\ m_3 \neq m_4 \end{cases}$$

#Q. The 3 variable function $f = \Sigma(0,1,2,4)$ its self dual.

$n = 3$

$2^{n-1} = 2^2 = 4$

This function is self dual function.

How many self dual function are possible for n boolean variables :—

n variable के case में total possible boolean function $= 2^{2^n}$

x	y	z
0	0	0
0	0	1
0	1	0
.	.	.
.	.	.
1	1	1

$(m_0\ m_7)$ $\quad m_0\ m_1\ m_2\ m_3$
$(m_1\ m_6)$ $\quad m_0\ m_6\ m_2\ m_3$
$(m_2\ m_5)$ $\quad m_7\ m_1\ m_5\ m_4$
$(m_3\ m_4)$

$= 2^4$

No. of self dual function $= 2^{2^{n-1}}$

Q→ Design a logic ckt. to convert a single digit BCD number to number modulo 6 as booleans (Do not detect the illegal inputs)

(a) Write the truth table for all bits labeled the input bits $I_1, I_2 \ldots$ with I_1 as the least significant bit, Label the output bits ~~for each output using~~ $R_1, R_2 \ldots$ with R_1 the least significant bit.

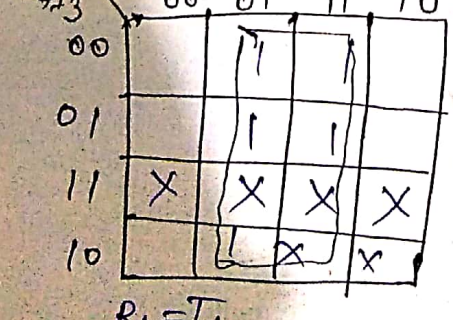
(b) Draw one circuit for each output using altogether two input

modulo 6 ⇒ 0–5

0/6 = 0
1/6 = 1
2/6 = 2
3/6 = 3
4/6 = 4
5/6 = 5
6/6 = 0
7/6 = 1
8/6 = 2
9/6 = 3

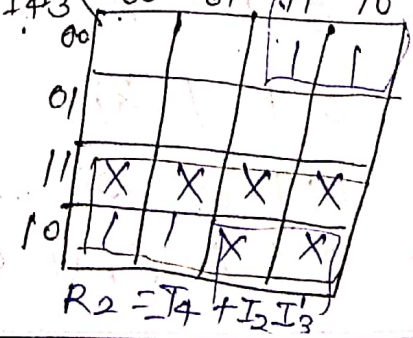
I_4	I_3	I_2	I_1	R_3	R_2	R_1	
0	0	0	0	0	0	0	m_0
0	0	0	1	0	0	1	m_1
0	0	1	0	0	1	0	m_2
0	0	1	1	0	1	1	m_3
0	1	0	0	1	0	0	m_4
0	1	0	1	1	0	1	m_5
0	1	1	0	0	0	0	m_6
0	1	1	1	0	0	1	m_7
1	0	0	0	0	1	0	m_8
1	0	0	1	0	1	1	m_9
1	0	1	0				
1	0	1	1				

R_1 :–



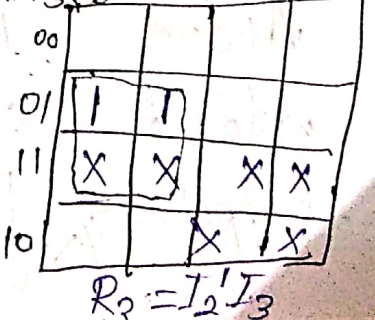
$R_1 = I_1$

R_2 :–

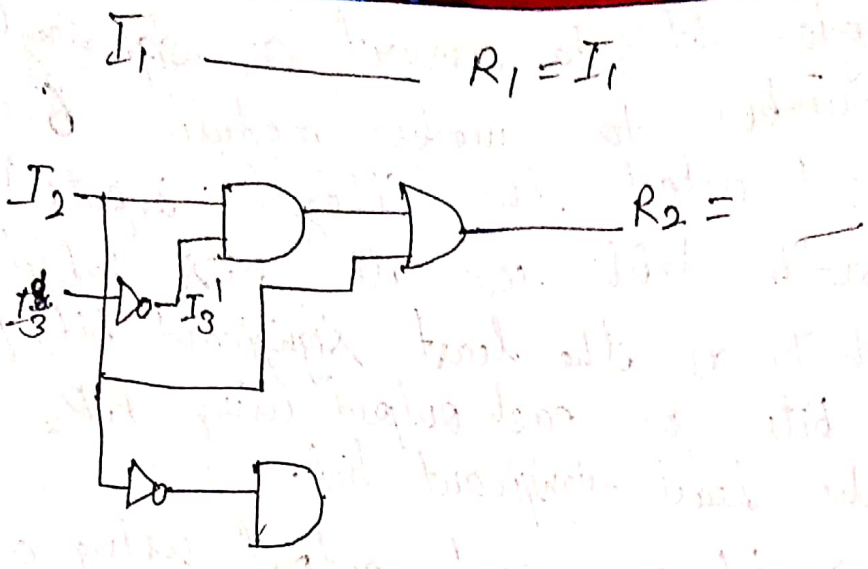


$R_2 = I_4 + I_2 I_3'$

R_3 :–

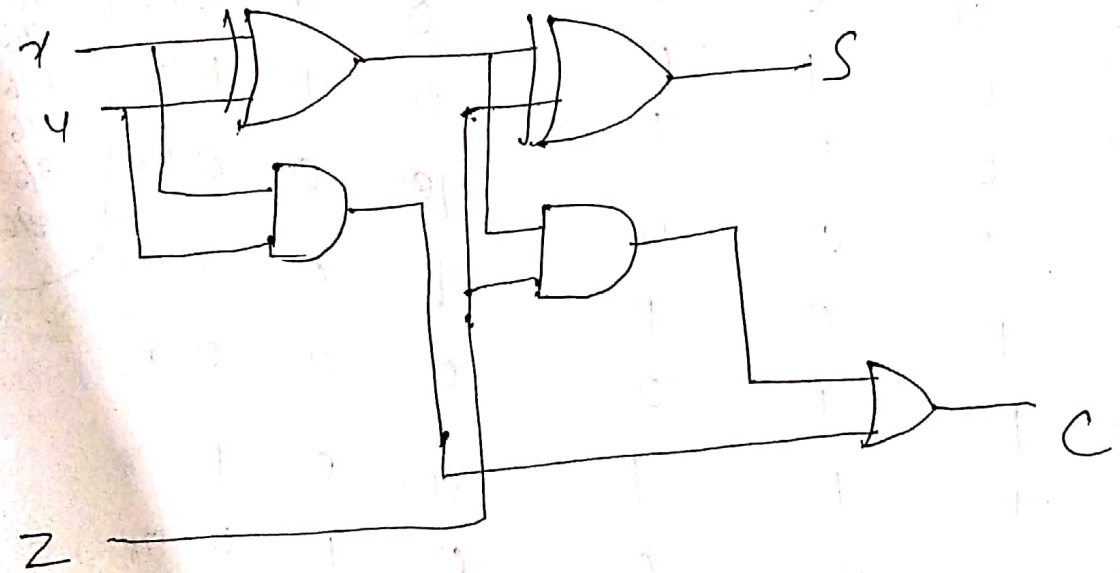


$R_3 = I_2' I_3$

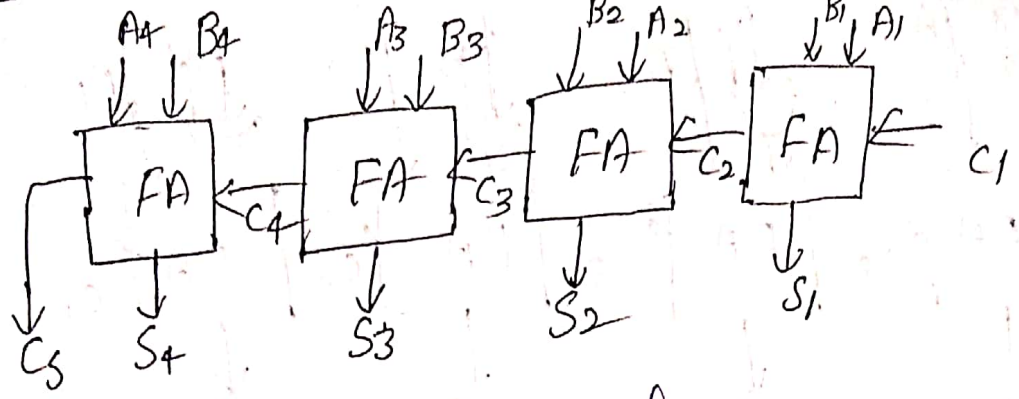


Binary Adder & Subtractor:-

Full Adder using two half Adder and OR Gate



Subscript	4	3	2	1		Full Adder
Input carry	0	1	1	0	C_i	z
Augend	1	0	1	1	A_i	x
Addend	0	0	1	1	B_i	y
Output carry	1	1	1	0	S_i	S
	0	0	1	1	C_{i+1}	C

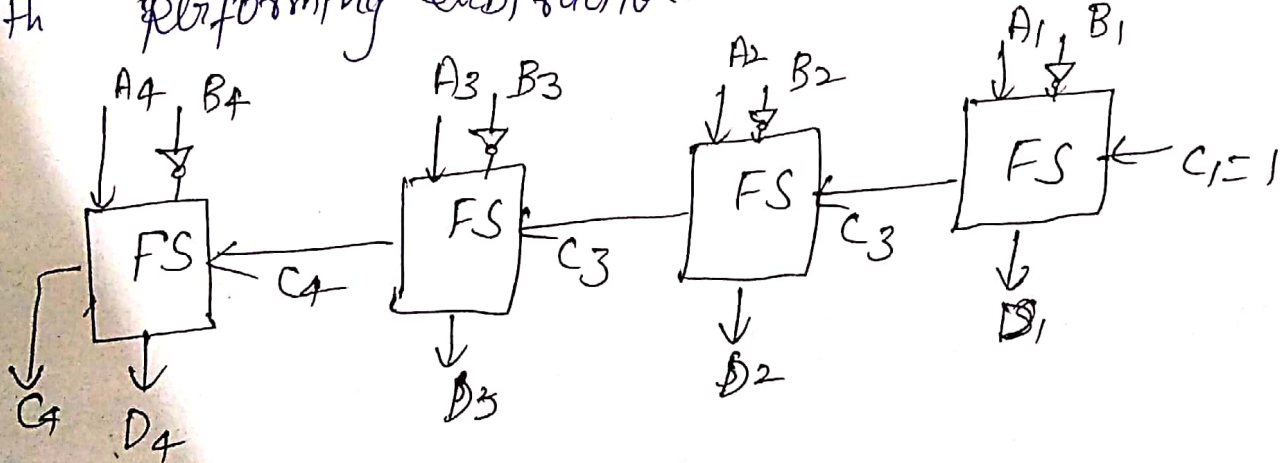


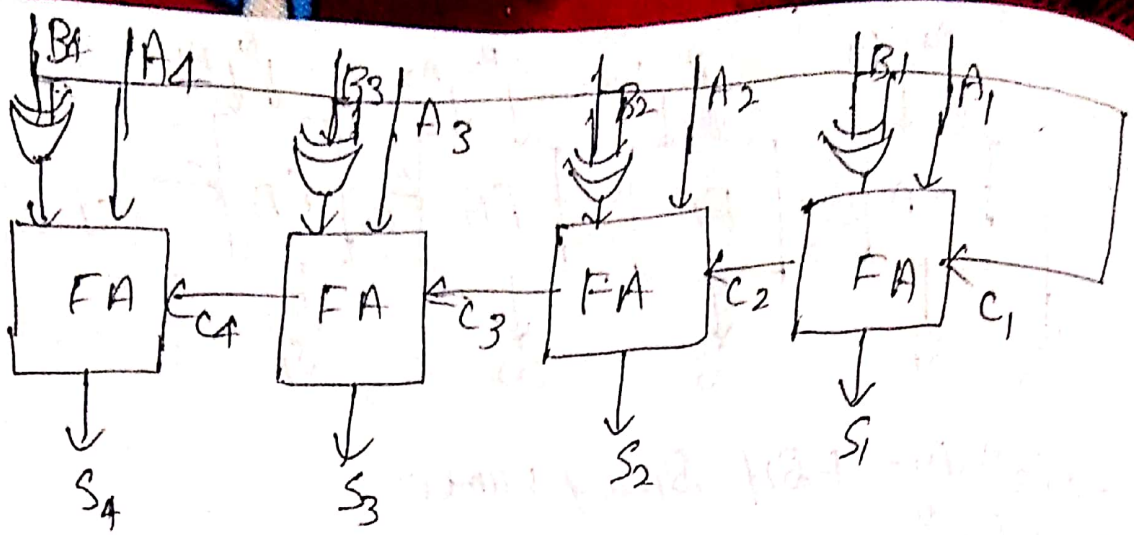
$A_4A_3A_2A_1$ – 4 Bit Binary number
+ $B_4B_3B_2B_1$

Subtractor :-

A – B
A + (2's Complement of B)
A + (1's comp. + 1)

The ckt. for subtractor A-B consists of a parallel adder with inverters placed between each data input B. The input carry C_1 must be equal to 1 with performing subtraction.





$B_4\ B_3\ B_2\ B_1$
$\oplus\ \oplus\ \oplus\ \oplus$
$0\ 0\ 0\ 0$
$B_4\ B_3\ B_2\ B_1$

$x \oplus 0 = x$
$1 \oplus 0 = 1$
$0 \oplus 0 = 0$

$B_4\ B_3\ B_2\ B_1$
$\oplus\ \oplus\ \oplus\ \oplus$
$1\ 1\ 1\ 1$
$B_4'\ B_3'\ B_2'\ B_1'$

$x \oplus 1 = x'$
$0 \oplus 1 = 1$
$1 \oplus 1 = 0$

$M = 1$ then Adder

$M = 0$

functionally Complete Operations :—

A set of operation is said to be functionally complete or universal if and only if every switching function can be expressed by means of operations in it.

The set {AND, OR, NOT} is clearly functionally complete
The set {AND, NOT} is also functionally complete.
The set {OR, NOT} is also functionally complete.

A set is said to be functionally complete if we can derive a set which is already functionally complete.

{AND, NOT} is also functionally Complete.

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

{OR, NOT} is also functionally complete.

$A\cdot B = \overline{(\overline{A}+\overline{B})} = \overline{\overline{A}}\cdot\overline{\overline{B}} = AB$

Show that $f(A,B,C) = A' + BC'$ is functionally complete FC.

$f(A,A,A) = A' + AA' = A'$
$f(B,B,B) = B' + BB' = B'$
$f(C,C,C) = C' + CC' = C'$

$f\{f(A,A,A), B, f(B,B,B)\} = (A')' + B(B')'$
$= A + B\cdot B$
$= A + B$ Hence it is F.C.

Show that $\{1, A\bar{B}\}$ is functionally complete.

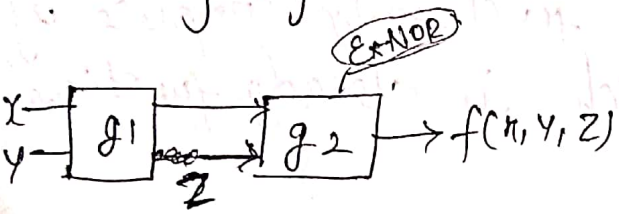
$f(A,B) = A\bar{B}$

$f(1,A) = 1 \cdot \bar{A} = \bar{A}$

$f(A, f(1,B)) = A(\bar{B}) = A\bar{B}$

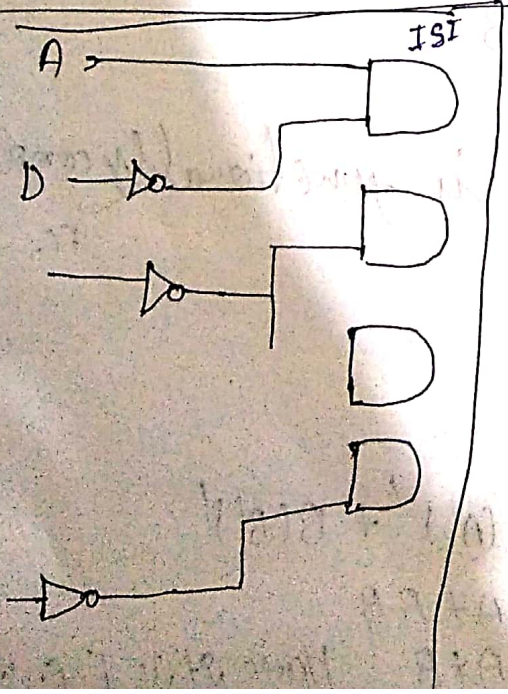
Ex-NOR
0 0 → 1
0 1 → 0
1 0 → 0
1 1 → 1

In the combinational ckt shown below g_1 is an unknown logic gate and $g_2(A,B) = AB + \bar{A}\bar{B}$, A partial truth table of the boolean function realized by the circuit is given below complete the table with justification.

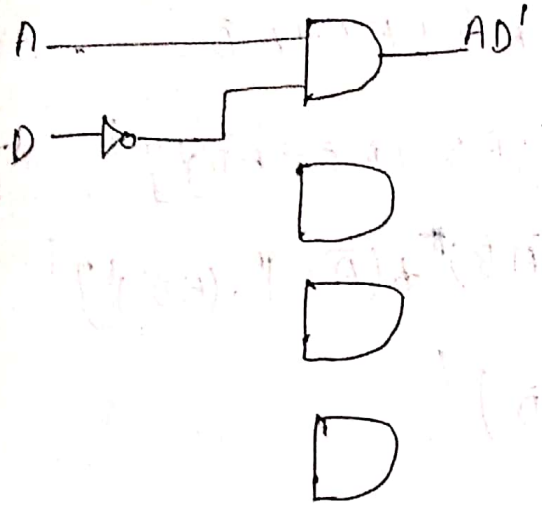


X	Y	Z	f
0	0	1	1
0	0	1	1	0
1	1	0	0	0
1	0	ϕ	1	0

x	Y	z	f
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	1
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	0



Q⇒ Assuming that all gates cost Rs 5 per input find the minimum cost realization F using only Inverters, AND/OR gates.



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

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

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

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

$$= m_{14} + m_{12} + m_{10} + m_8$$

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

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

$$= m_5 + m_4 + m_1 + m_0$$

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

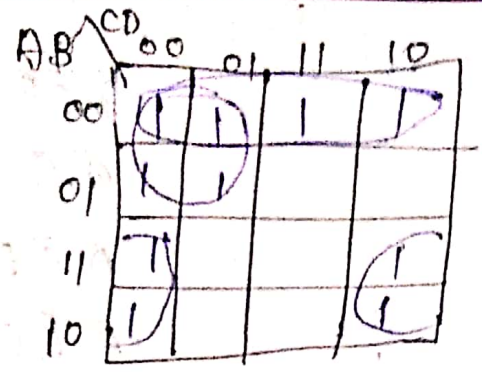
$$= m_4 + m_0$$

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

$$= m_3 + m_2$$

$$F = m_{14} + m_{12} + m_{10} + m_8 + m_5 + m_4 + m_1 + m_0 + m_4 + m_0 + m_3 + m_2$$

$$= m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_8 + m_{10} + m_{12} + m_{14}$$



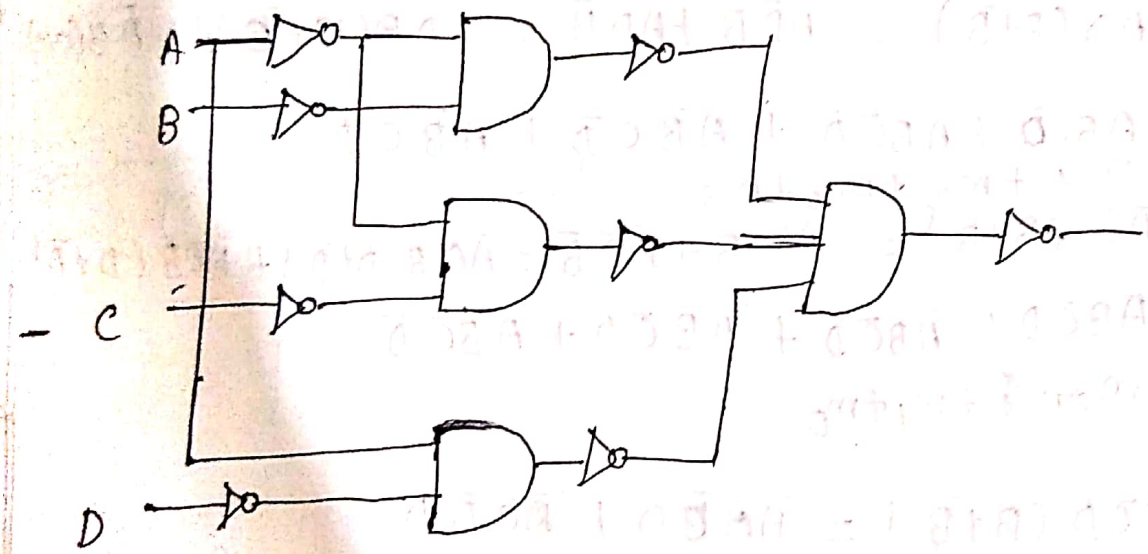
$f = \bar{A}\bar{B} + \bar{A}C + AD$

$f = [(\bar{A}\bar{B} + \bar{A}C + AD')']'$

$= [(\bar{A}\bar{B})' \cdot (\bar{A}C)' \cdot (AD')']'$

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

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



Consider a four input hypothetical logic gate graph the following K-map.

AB \ CD	00	01	11	10
00	1			
01		1		
11			1	
10				1

(i) Prove or disprove that G can be treated as a universal logic gate. A large no. of candidates they are a variable to your

(ii) Implement A+B using minimum no. of gates.

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

$g(A,A,A,A) = \bar{A}\bar{A}\bar{A}\bar{A} + \bar{A}A\bar{A}A + AAAA + A\bar{A}A\bar{A}$

$= \bar{A} + 0 + A + 0$

$= \bar{A} + A$

$= 1$ Not passible

$g(A,A,0,0) = \bar{A}\bar{A}\cdot 1\cdot 1 + \bar{A}A\cdot 1\cdot 0 + AA\cdot 0\cdot 0 + A\bar{A}\cdot 0\cdot 1$

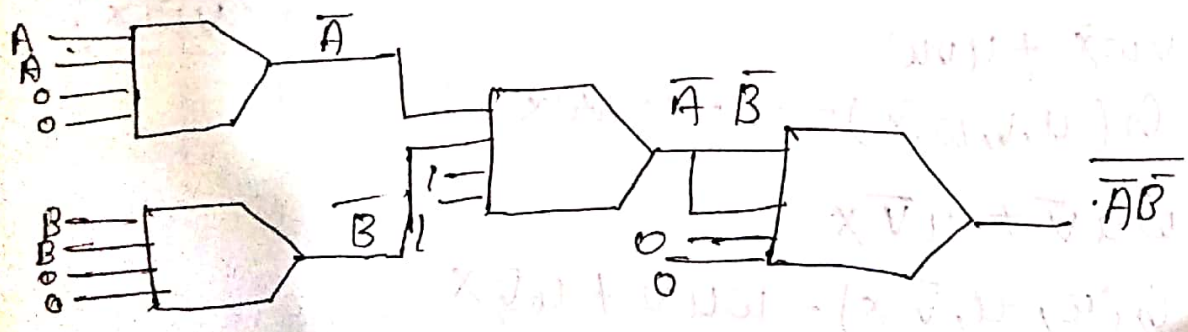
$= \bar{A}$

$g(A,B,1,1) = \bar{A}\bar{B}\cdot 0\cdot 0 + \bar{A}B\cdot 0\cdot 1 + AB\cdot 1\cdot 1 + A\bar{B}\cdot 1\cdot 0$

$= AB$

It is universal gate.

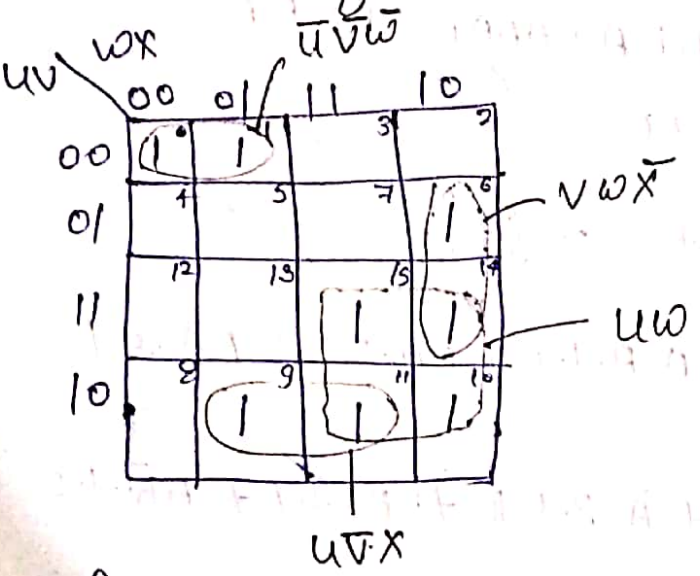
(ii) $\overline{(\bar{A}\,\bar{B})} = \bar{\bar{A}} + \bar{\bar{B}} = A + B$



A certain four-input gate G realizes the switching $G(a,b,c,d) = abc + bcd$.
Assuming that the input variable are a variable in both complemented and uncomplemented form

(i) Show a realization of the function
$f(u,v,w,x) = \Sigma(0,1,6,9,10,11,14,15)$.
With only three G gates and one OR gate

(ii) Can all switching function be realized with a G, OR & logic set.



$f_{(u,v,w,x)} = \bar{u}\bar{v}\bar{w} + vw\bar{x} + u\bar{v}x + uw$

$= \bar{u}\bar{v}\bar{w} + vw\bar{x} + u\bar{v}x + uw(v+\bar{v})$

$= \bar{u}\bar{v}\bar{w} + vw\bar{x} + u\bar{v}x + uvw + u\bar{v}w$

$vw\bar{x} + uvw$ ($\bar{u}w\bar{x}$)

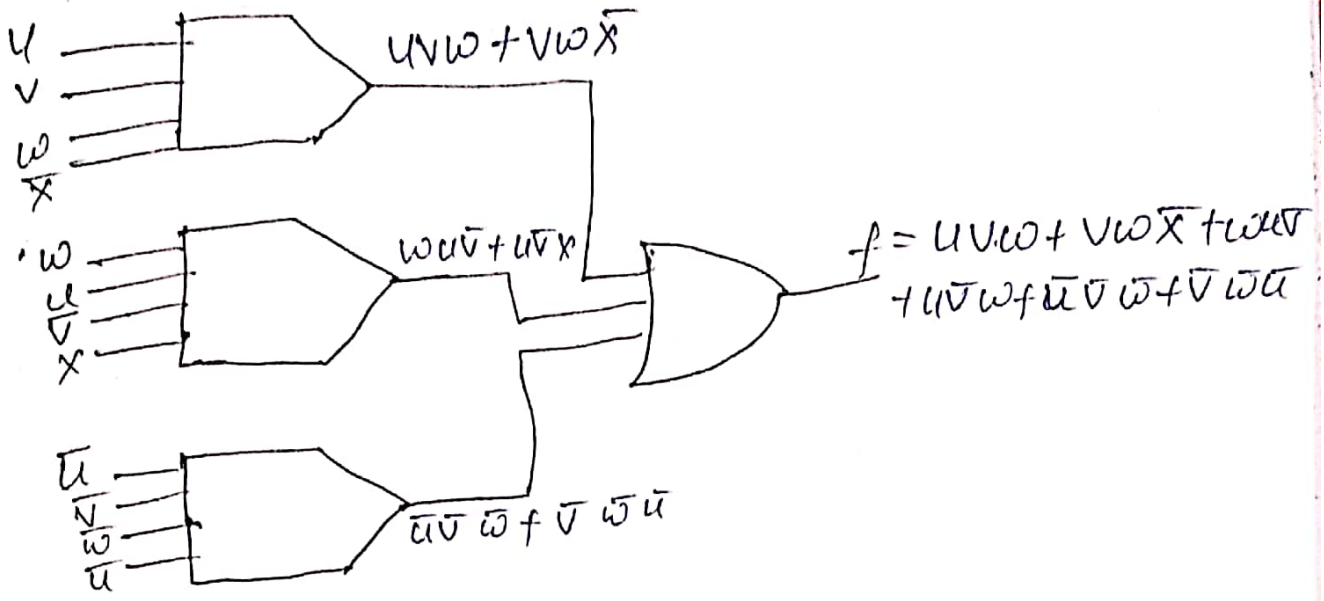
$G(u,v,w,\bar{x}) = uvw + vw\bar{x}$

$w\,u\bar{v} + u\bar{v}x$

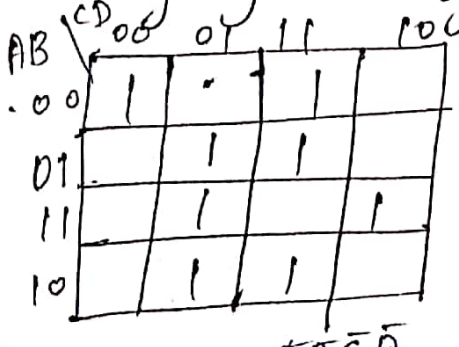
$G(w,u,\bar{v},x) = wu\bar{v} + u\bar{v}x$

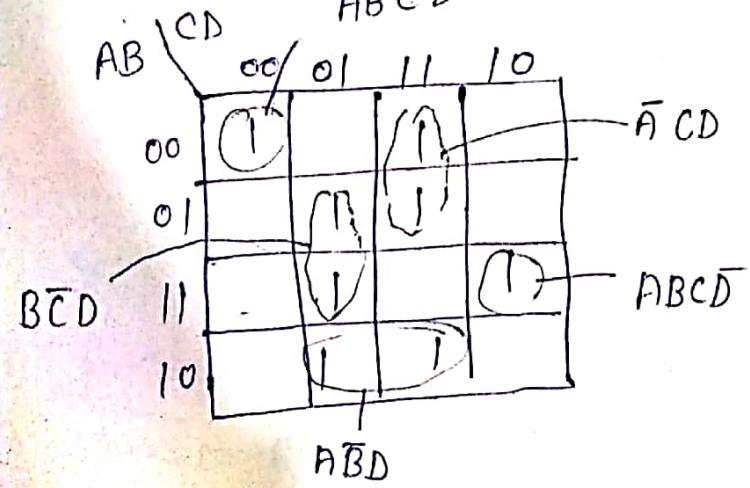
$u\bar{v}\bar{w} + \bar{u}\bar{v}\bar{w}$ ← add.

$G(\bar{u},\bar{v},\bar{w},u) = \bar{u}\bar{v}\bar{w} + \bar{v}\bar{w}u$



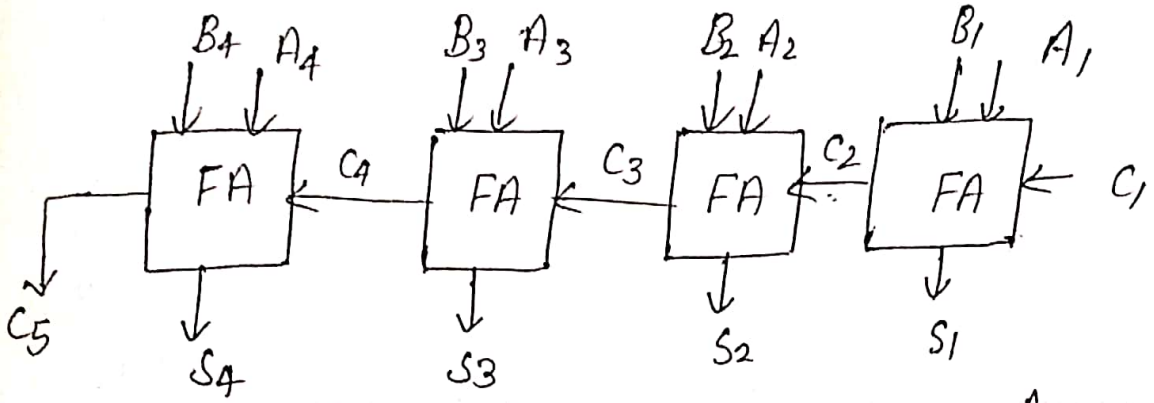
Given a library of 2 input AND, NOT and 2 input XOR gates, synthesize the function f(A,B,C,D) as shown in the Karnaugh map using minimum no. of gates of the library.



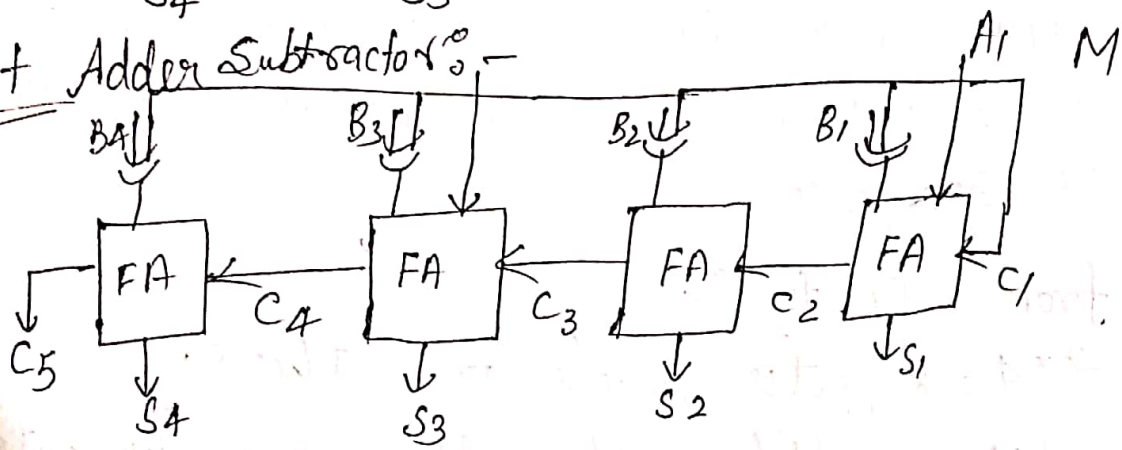


$$f = \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}D + A\bar{B}D + \bar{A}CD + ABC\bar{D}$$

Parallel Adder :-



4-bit Adder Subtractor :-



1 – Subtractor
0 – Adder

In any combinational circuit the signal must propagate through the gates before the correct output is available at the terminals.

Total propagate time is equal to the progation delay of a typical gate times the number of gate levels in the circuits. The longest propagation delay time in a parallel adder is the time it takes the carry to propagate through the parrell adder.

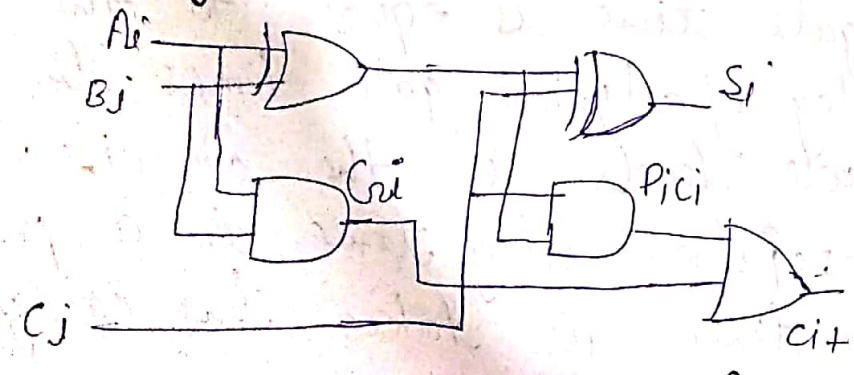
The signal from the input carry c_i to total output carry c_{i+1} propagates through an AND gate and an OR ~~OR~~ gate.

from C_1 to C_5

$2 \times 4 = 8$ gate levels are there.

for n bit parallel adder there are $2n$ gate levels for the carry to propagate through.

⇒ There are several techniques for reducing the carry propagation time in a parallel adder. The most widely used technique types the principl of carry look ~~ab~~ ahead.



$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

$S_i = P_i \oplus C_i$

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

G_i is called carry generate

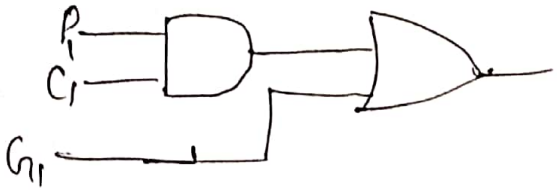
G_i is called carry generate and it produces an output carry when both A_i and B_i are 1, regardless the input carry C_i.

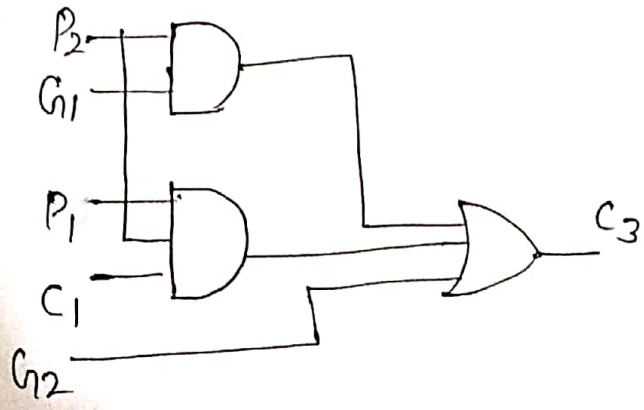
P_i is called a carry progate because it is the term associated with the propagation of carry from c_i to ~~c_i~~ c_{i+1}.

$$C_2 = G_1 + P_1 C_1$$

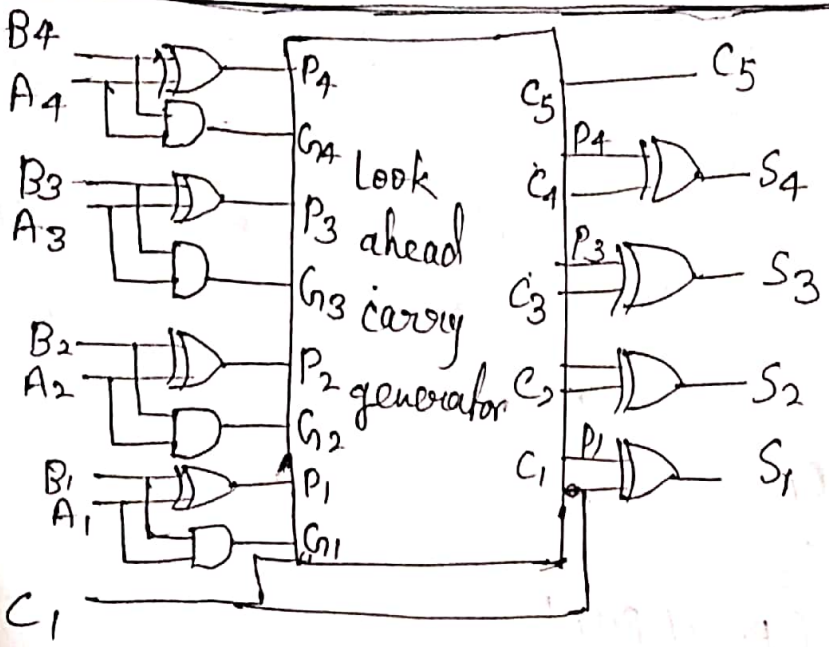
$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 (G_1 + P_1 C_1)$$
$$= G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$



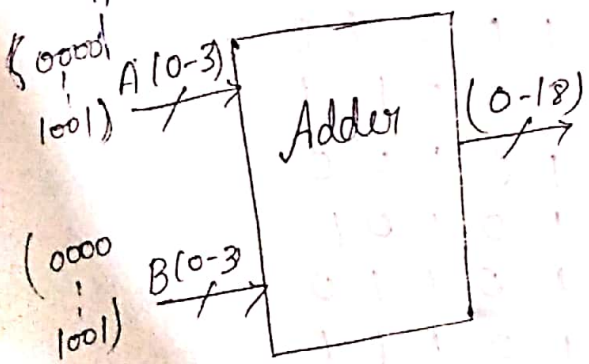


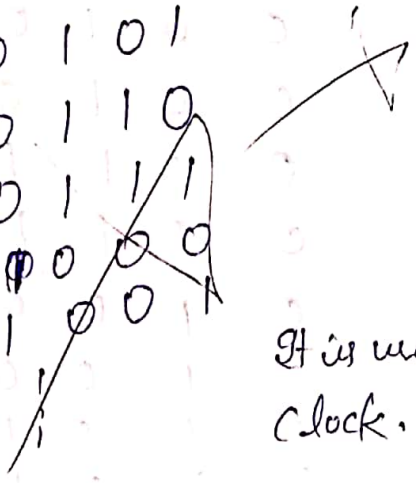
4-bit full adder with look ahead carry:-



BCD Adder:-

Decimal	Binary Sum					BCD Sum				
	C	S_3	S_2	S_1	S_0	C	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10										
11										

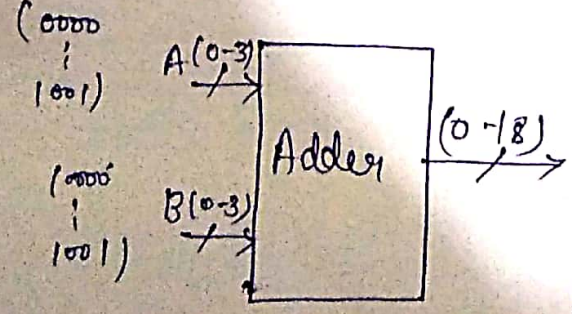




It is used in digital clock.

BCD Adder:- It is used in digital clock.

Decimal	Binary Sum					BCD Sum				
	C*	S_3^*	S_2^*	S_1^*	S_0^*	C	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1



(0-9) Binary sum is same as BCD sum.

(10-18) the Binary sum is not same as BCD sum.

(15) → 1111 Binary of 15

BCD of 15 → 0001 0101

How we convert ~~the BCD Sum into~~ Binary Sum to the BCD Sum!—

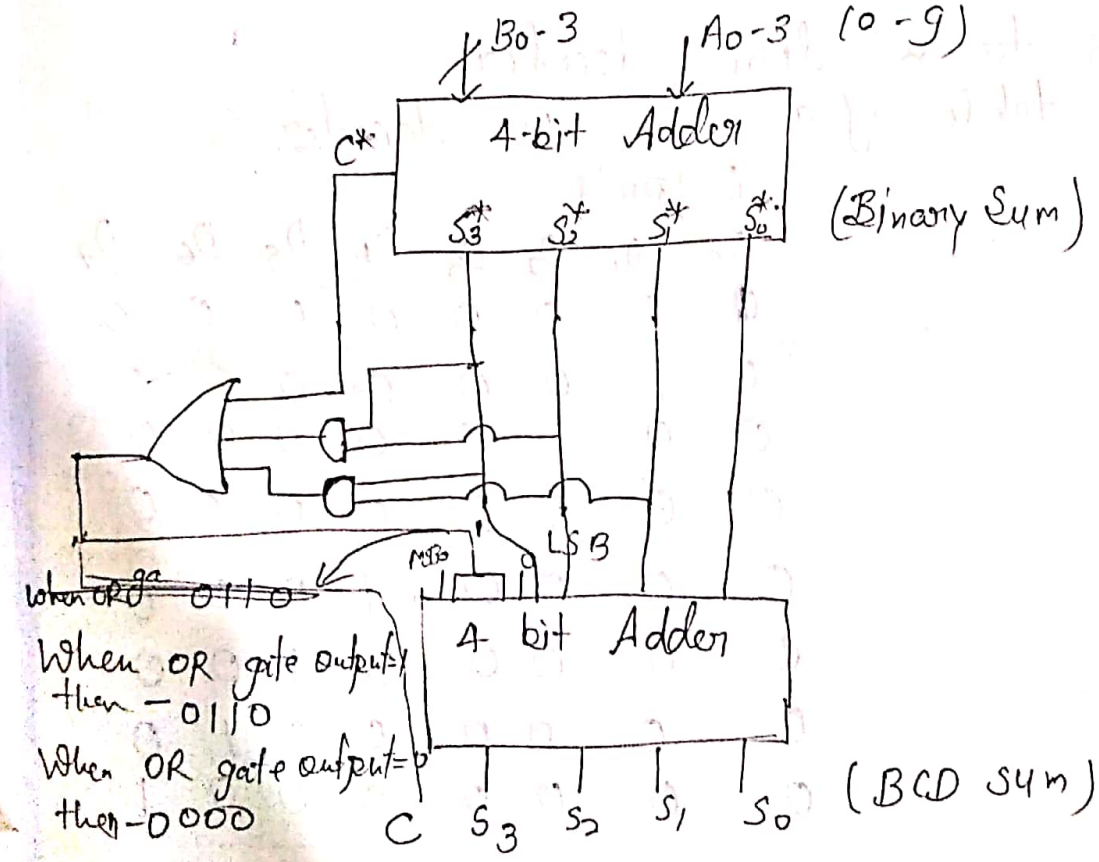
Whenever the number is greater than 9 we have to add 6 (0110) and you will add this the output will be in BCD.

"0110" when :— i) $C^* = 1$ (16 - 19)

ii) $S_3^* \cdot (S_2^* + S_1^*)$ (12 · 15)

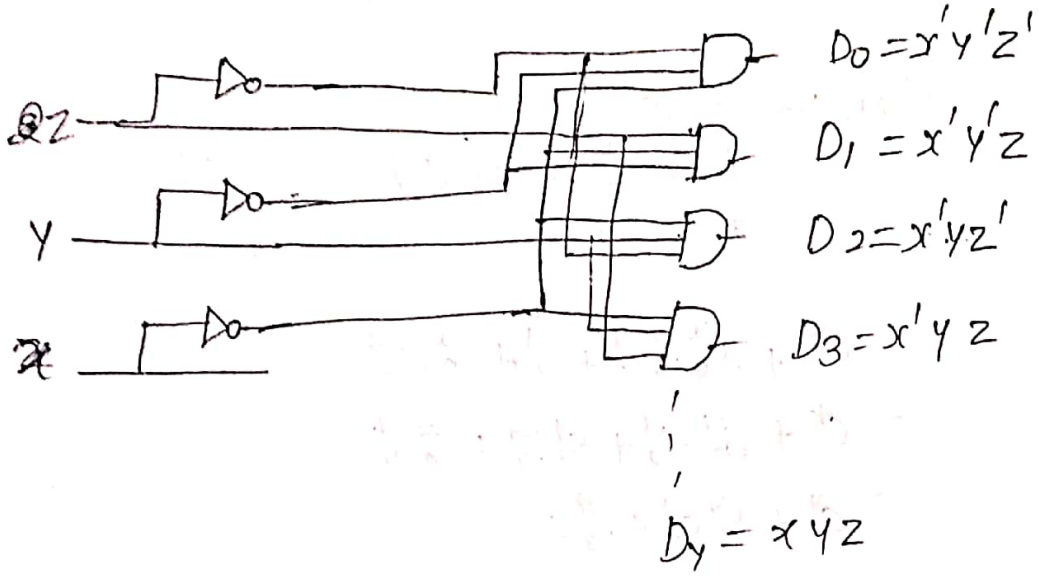
iii) $S_3^* \cdot S_1^*$ (10, 11)

$$= C^* + S_3^* \cdot (S_2^* + S_1^*) + S_3^* \cdot S_1^*$$
$$= C^* + S_3^* \cdot S_2^* + S_3^* S_1^* + S_3^* S_1^*$$
$$= C^* + S_3^* S_2^* + S_3^* S_1^*$$



Decoder:-

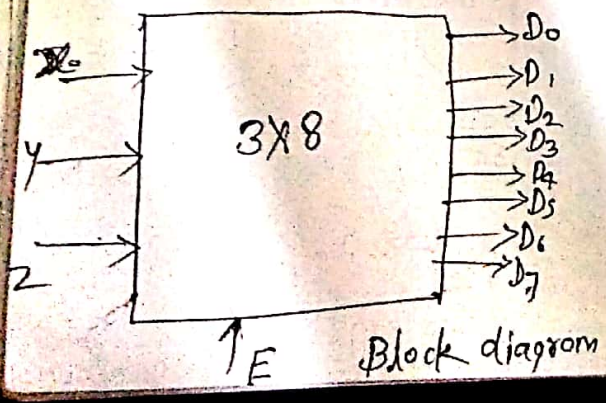
A decoder is a combinational circuit that converts Gray information from n input lines to a maximum of 2^n unique output lines.



A 3 to 8 line decoder

Truth table of a 3-8 line decoder

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



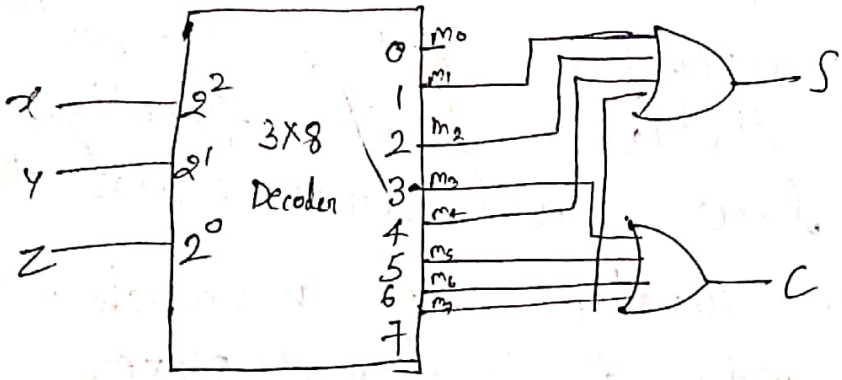
Block diagram

Implementation of a full adder with the help of a decoder and two OR gates:-

full adder में 3 Input and 2 Output होते हैं।

$S(x,y,z) = \Sigma(1,2,4,7)$
$C(x,y,z) = \Sigma(3,5,6,7)$

We make full adder with the decoder because it reduces the ~~size~~ no. of gates and also size of ckt.



Truth Table

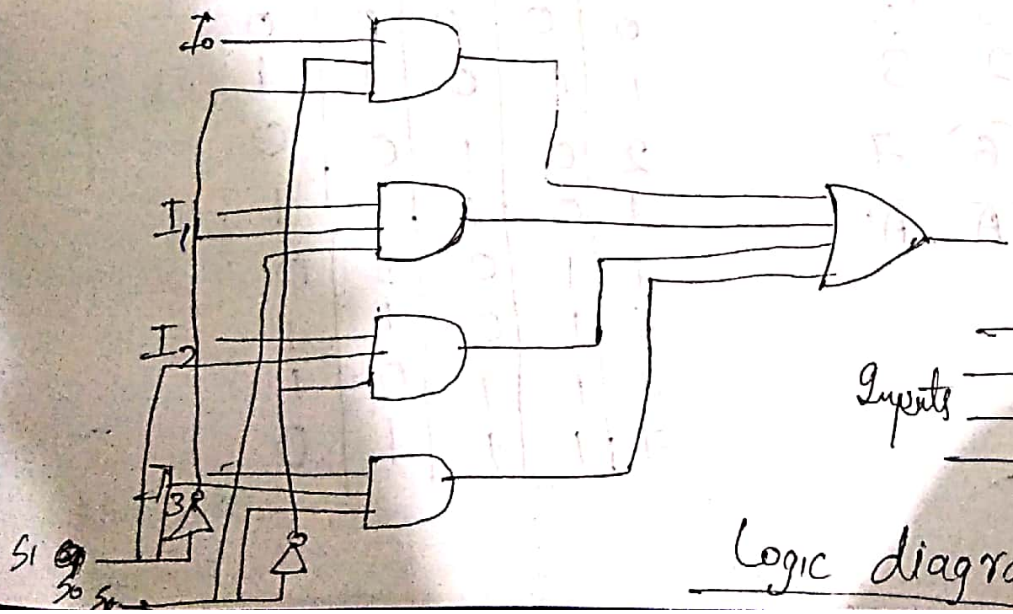
x	y	z	S	C	
0	0	0	0	0	m_0
0	0	1	1	0	m_1
0	1	0	1	0	m_2
0	1	1	0	1	m_3
1	0	0	1	0	m_4
1	0	1	0	1	m_5
1	1	0	0	1	m_6
1	1	1	1	1	m_7

$S(x,y,z) = \Sigma(m_1, m_2, m_4, m_7)$
$C(x,y,z) = \Sigma(m_3, m_5, m_6, m_7)$

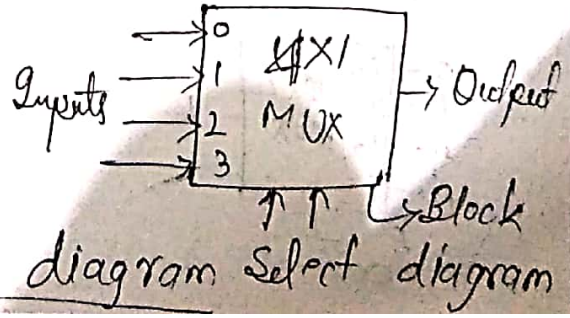
Multiplexer:-

Multiplexing means transmitting a large no. of information units over a smaller no. of channels of lines. A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

S_1	S_0	Y ← func. table
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Logic diagram



Select diagram

Multiplexer में 2^n input के लिए n output होते हैं।

A multiplexer is essentially a decoder with the OR gate already available. The minterms out of the decoder to be chosen can be connected with the input lines. The minterms table included with the function being implemented are chosen by making their corresponding input lines equal to 1. The minterms not included in the function disabled by making their input lines equal to 0.

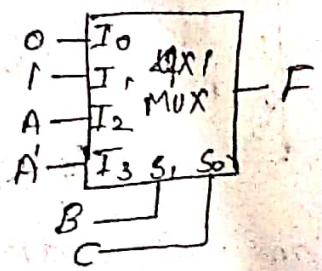
So any boolean function of n variables can be implemented with 2^n to 1 multiplexer.

If we have a function of n+1 boolean variables we take n of those variables and connect them to the selection lines of the multiplexer. The remaining single variable of the function is used for the inputs of the multiplexer.

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

	I_0	I_1	I_2	I_3
A'	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	A'

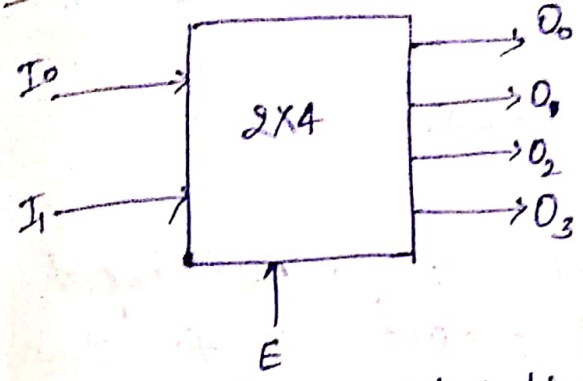
minterms	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



Decoder:-

n = 2 input then Output = 2^n

2 × 4 या 1 Decoder है।

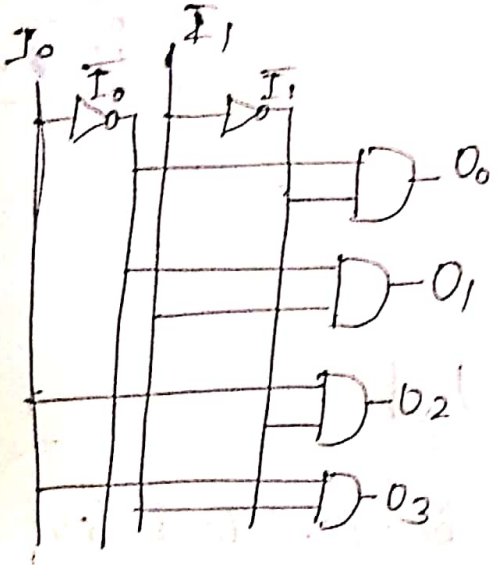


Input		Output			
I_0	I_1	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

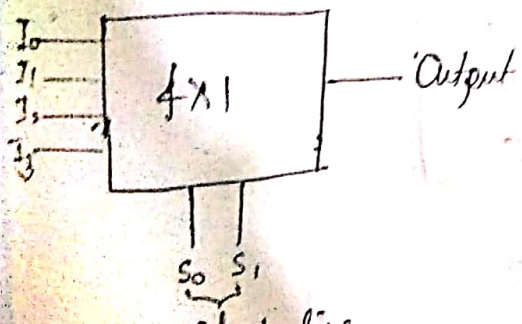
(Enable line or select line)

Block Diagram of Decoder

→ It is a switch of decoder by which decoder is on or off.



Multiplexer or MUX:-



How we can find no. of select line

$$n = 2^s$$

where n = Total no. of input
s = Total no. of select line

When input n = 4, select line then

$4 = 2^s$

$2^2 = 2^s$

$s = 2$

When input line n = 8

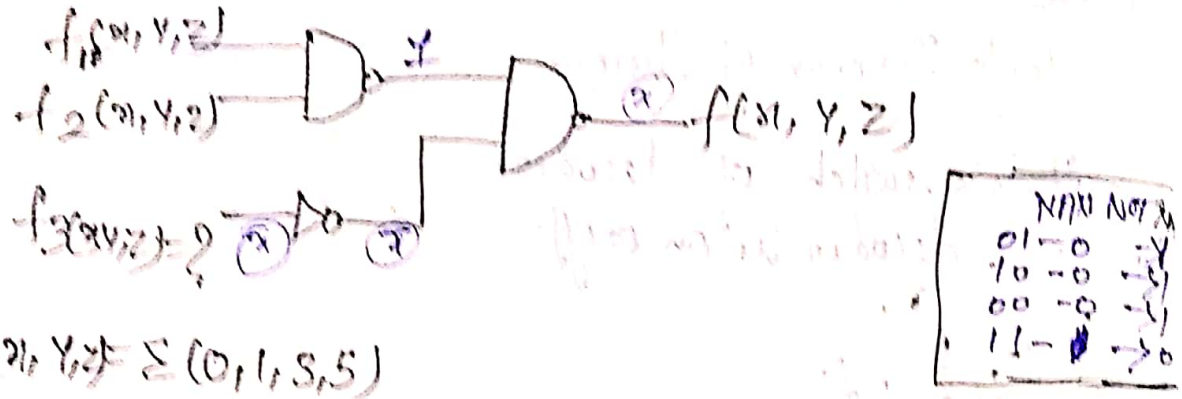
$2^3 = 2^s$ $\quad s = 3$

There will be 3 select line

Select line defines which input will be your output.

Multiplexer is also known as MUX.

Q→ Consider the following logic ckt whose input are functions f_1, f_2, f_3 and output is f.



$f_1(x,Y,Z) = \Sigma(0,1,5,5)$

$f_2(x,Y,Z) = \Sigma(6,7)$

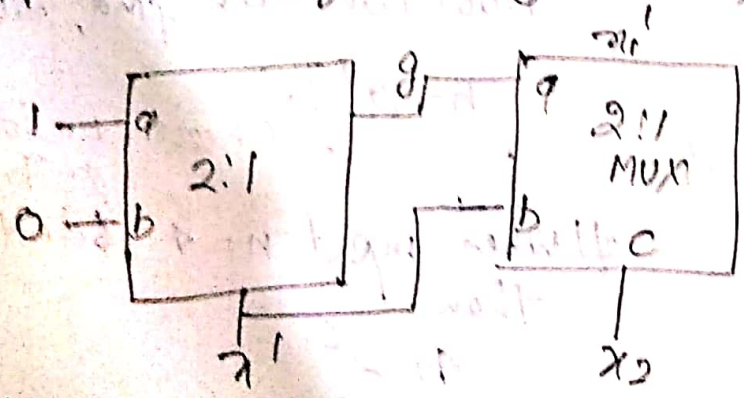
$f(x,Y,Z) = \Sigma(1,4,5)$

Hence $f = f_3$

$\therefore f_3(x,Y,Z) = \Sigma(1,4,5)$ Ans

MUX

Q→ Consider the circuit shown below. The output of a 2:1 MUX is given by the function $ac' + bc$



$Y = I_0\bar{S} + I_1 S$

when $S=0$ then $Y = I_0$

if $S_1 = 1$ then $Y = I_1$

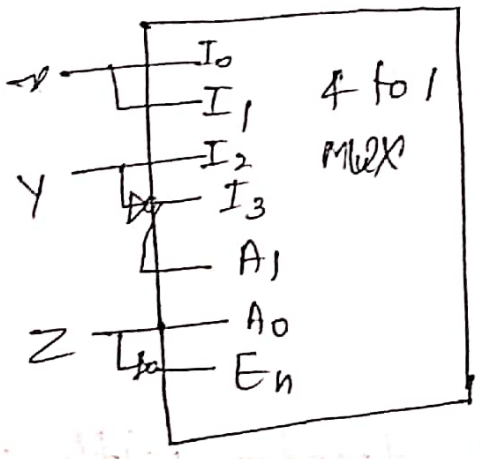
$g = ac' + bc$

$= 1\bar{x}_1 + 0x_1$

$= \bar{x}_1$

$x_1' x_2' + x_1 x_2$

Q=> Consider the following multiplexer where I_0, I_1, I_2, I_3 are four data input lines selected by two ~~ddo~~ address line combinations $A_1A_0 = 00, 01, 10, 11$ respectively and f is the output of the multiplexor. En is the ~~ena~~ enable input.



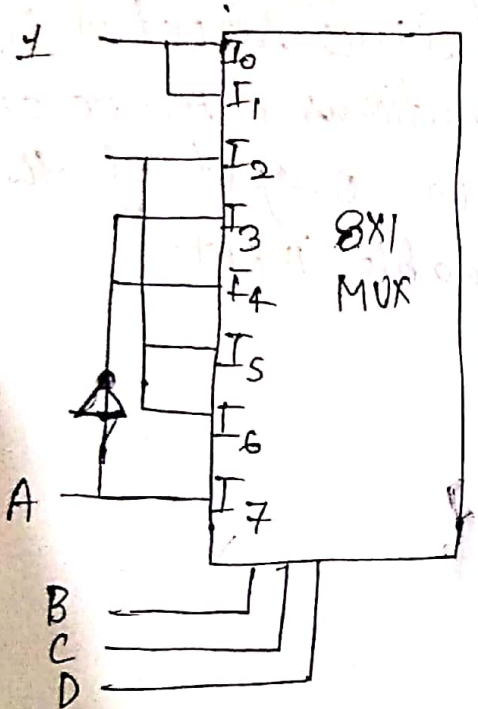
x	Y	Z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

A_1	A_0	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

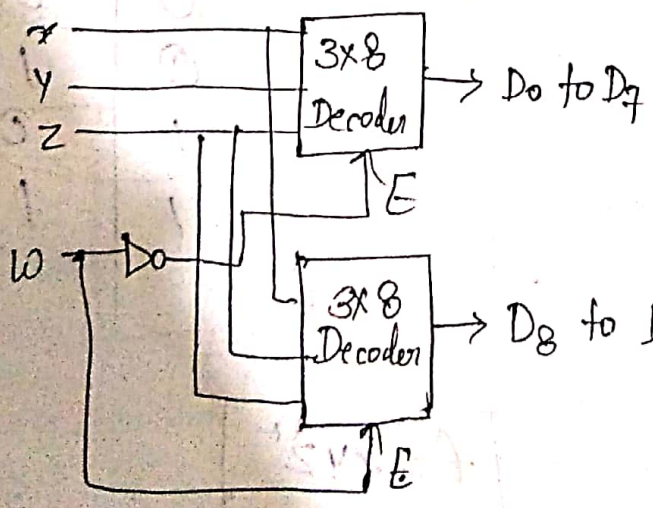
$f = xyz'$

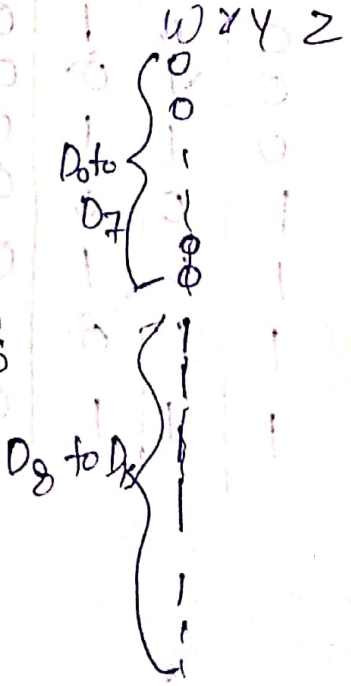
Q→ $F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	0	A



Constructing a 4X16 decoder with two 3X8 decoders.



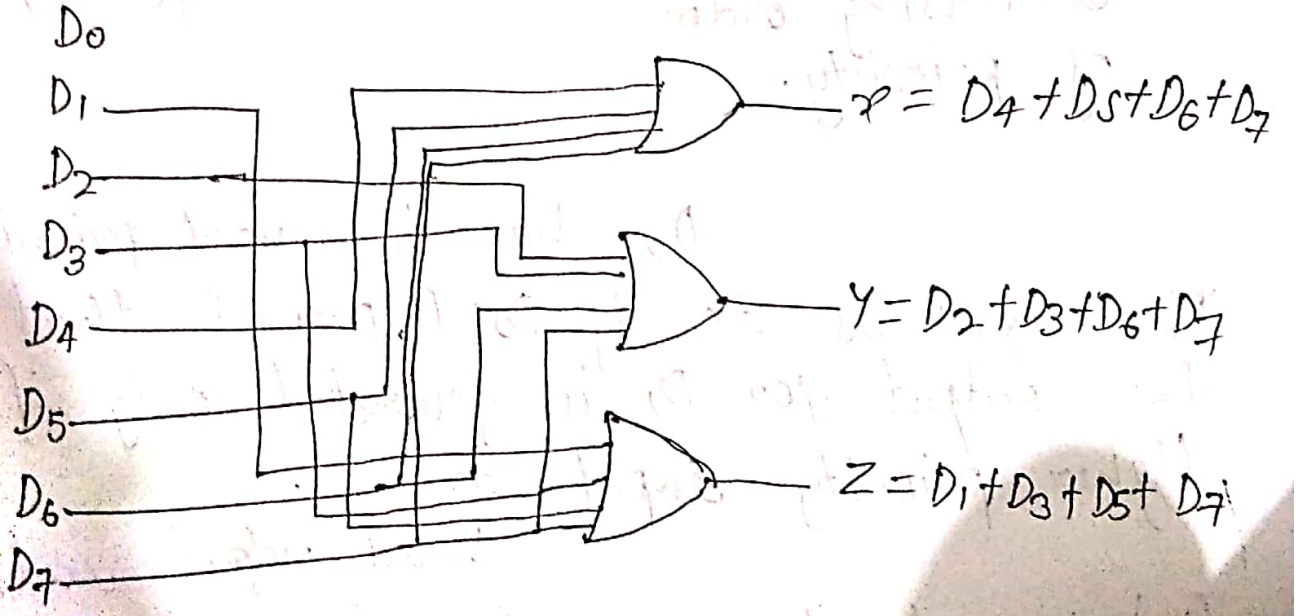


Encoders:-

An encoder is a digital circuit that performs the inverse operation of a decoder. An Encoder has 2^n (or fewer) input lines and n output lines.

Truth table for octal to Binary Encoder:-

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1



Priority Encoder:-

A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

Truth table of a priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	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

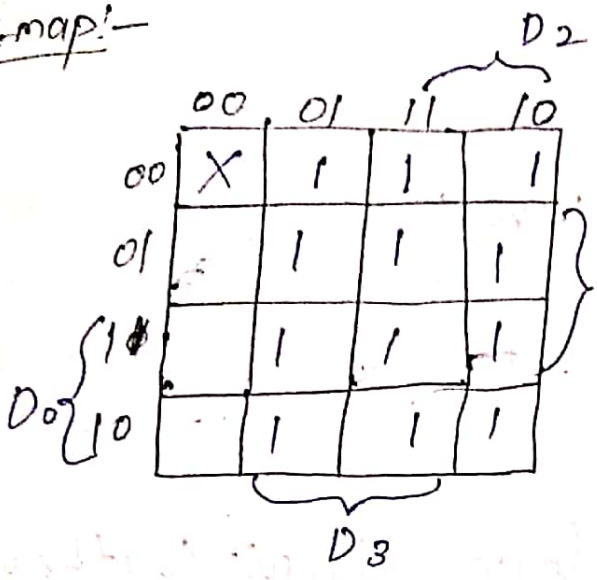
Input D_3 has the highest priority so regardless of themselves of other inputs When this input is 1, the output for xy is 11.

D_0 D_1 D_2 D_3 → Increasing order of priority.

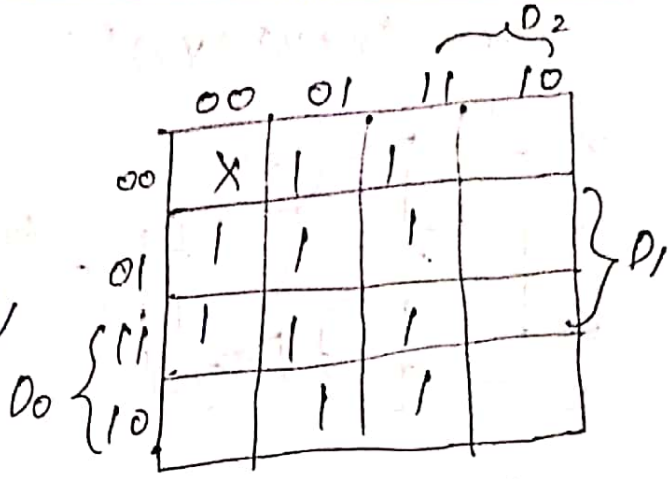
D_2 has the next priority level The output is 10 if $D_2=1$, proved that $D_3=0$. The output for D_1 is generated only if other higher priority input are 0.

Valid output indicator (V) is set to 1 only when one or more inputs are equal to 1.

K-map:-



$x = D_2 + D_3$



$Y = D_3 + D_1 D_2'$

Q⇒ Express the function $f(x,y,z) = xy' + yz'$ with only one complement operation and one or more AND/OR operations. Draw the logic circuit implementing the expression obtained a single NOT gate and one or more AND/OR gates.

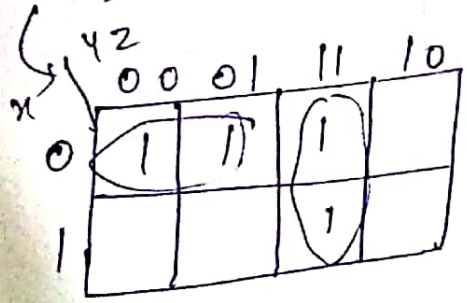
$f(x,y,z) = xy' + yz'$

$= xy'z' + xy'z + xyz' + x'yz'$

$= m_4 + m_5 + m_6 + m_2$

$f(x,y,z) = m_2 + m_4 + m_5 + m_6$

$f'(x,y,z) = m_0 + m_1 + m_3 + m_7$

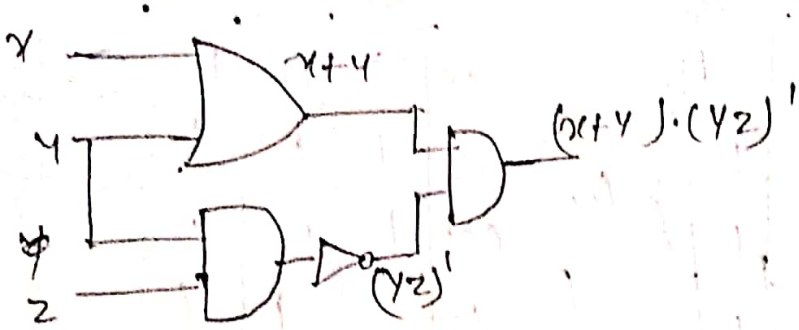


$f'_{(x,y,z)} = x'y' + yz$

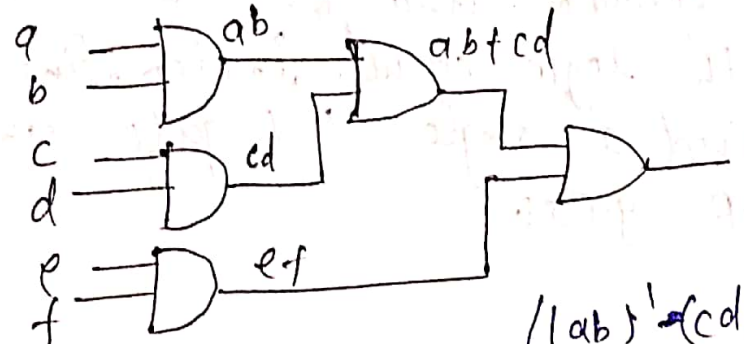
$f = (f')' = (x'y' + yz)'$

$= (x'y')' \cdot (yz)' = (x+y)(y'z)'$

$f = (x+y)(yz)'$



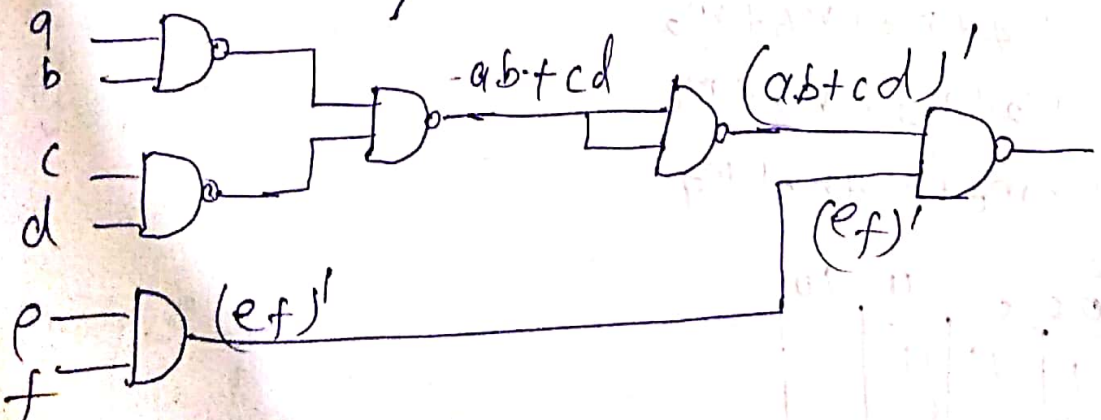
Transform the following logic circuit (without expression its switching function) into an equivalent logic circuit that employs only 6 NAND gates each with 2 inputs



$ab+cd+ef$

$((ab)'(cd)')' = (ab')' + (cd')'$

$= ab + cd$



$= ((ab+cd)'(ef)')'$

$= ((ab+cd)')' + ((ef)')'$

$= ab + cd + ef$

Let $f=(w'+y)(x'+y)(w+x'+z)(w'+z)(x'+z)$

(i) Express f in the minimal sum of products form.

(ii) If the output line is stuck at 0, for how many input combinations were the value of f be in correct.

$f=(w'+y)(x'+y)(w+x'+z)(w'+z)(x'+z)$.

$=(w'+xx'+y+zz')(ww'+x'+y+zz')(w+x'+yy'+z)$
$(w'+xx'+yy'+z)(xw'+x'+yy'+z)$

$f=(w'+x+y+z)(w'+x+y+z')(w'+x'+y+z)(w'+x'+y+z')$
$(w+x'+y+z)(w+x'+y+z')(w'+x'+y+z)(w'+x'+y+z)$
$(w+x'+y+z)(w+x'+y'+z)(w'+x+y+z)(w'+x'+y+z)$
$(w'+x'+y'+z)(w'+x'+y'+z)(w+x'+y+z)(w'+x'+y+z)$
$(w+x'+y'+z)(w'+x'+y'+z)$

$f=M_8M_9M_{12}M_{13}M_4M_5M_{12}M_{13}M_4M_6M_8M_{10}M_{12}$
$M_{14}M_4M_6M_{12}M_{14}$

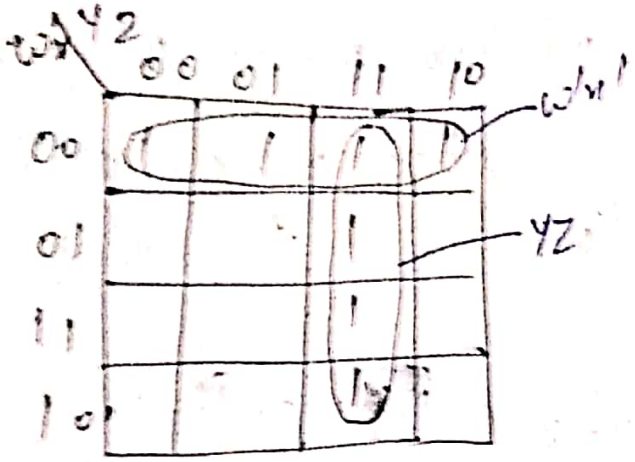
$f=M_4M_5M_6M_8M_9M_{10}M_{12}M_{13}M_{14}$

$f=\pi(4,5,6,8,9,10,12,13,14)$

$f=\Sigma(0,1,2,3,7,11,15)$

(ii) $f=\Sigma(0,1,2,3,7,11,15)$

① $f = \Sigma(0,1,2,5,7,11,15)$



$f = w'x' + yz$ Ans

Express the boolean function

$f = AB + BC + CA$

by using a 4:1 multiplexer.

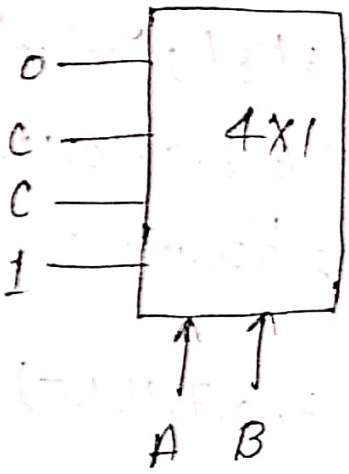
$f = AB + BC + CA$

$f = ABC + ABC' + ABC + A'BC + ABC + AB'C$

$f = ABC + ABC + AB'C + ABC'$

$f = m_3 + m_5 + m_6 + m_7$

A	B	C	f	
0	0	0	0	0
0	0	1	0	
0	1	0	0	C
0	1	1	1	
1	0	0	0	C
1	0	1	1	
1	1	0	1	1
1	1	1	1	

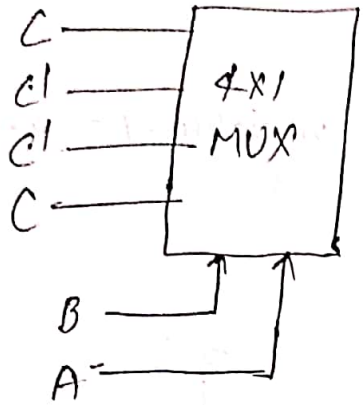


No. of variables = 3

No of selector lines for MUX = No. of variables - 1 = 3 - 1 = 2

Multiplexer Used: 4 to 1

What is the output of the following multiplexor circuit:-



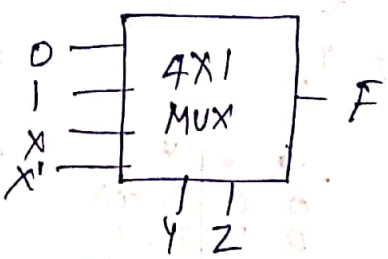
A	B	C	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$f = A'B'C + A'BC' + AB'C' + ABC$$
$$= A'BC' + AB'C' + A'B'C + ABC$$
$$= A'BC' + AB'C' + (AB + A'B')C$$
$$= (A'B + AB')C' + (AB + A'B')C$$
$$= \underbrace{(A \oplus B)}_{X} C' + \underbrace{(A \oplus B)'}_{X'} C$$
$$= XC' + X'C$$
$$= X \oplus C$$
$$= A \oplus B \oplus C$$ Ans

Implementing Digital function by using a Multiplexer

Truth table for $F(x,y,z) = \Sigma(1,3,5,6)$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



-1 = 3-1 = 2

Implementing Digital functions by using a Multiplexer

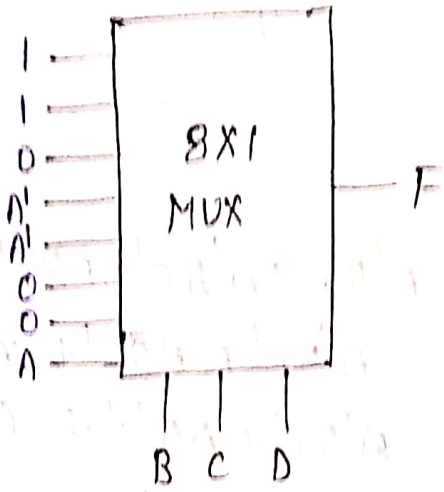
$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

No. of Variables = 4

No. of selector lines for MUX = No. of variables − 1 = 4 − 1 = 3

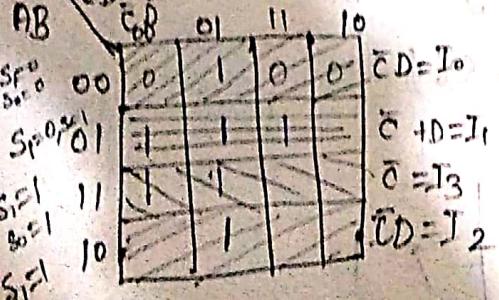
Multiplexer used : 8 to 1

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Implementing of Boolean function using Multiplexers

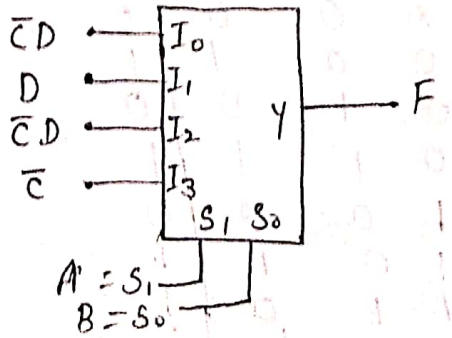
Implement :- $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$ using 4×1 MUX



No. of variable = 4 No. of select line $2^n = 4$

~~No. of selector lines for mux = 4 − 1~~ $n = 2$

S_1	S_0	Y
0	0	$I_0 = \bar{C}\bar{D}$
0	1	$I_1 = D$
1	0	$I_2 = \bar{C}D$
1	1	$I_3 = \bar{C}$



Implement $\Sigma m(0,1,2,3,4,10,11,14,15)$ using 8:1 MUX.

S_2 A	S_1 B	S_0 C	D	F	
0	0	0	0	1	F=1
0	0	0	1	1	
0	0	1	0	1	F=1
0	0	1	1	1	
0	1	0	0	1	F=D'
0	1	0	1	0	
0	1	1	0	0	F=0
0	1	1	1	0	
1	0	0	0	0	F=0
1	0	0	1	0	
1	0	1	0	1	F=1
1	0	1	1	1	
1	1	0	0	0	F=0
1	1	0	1	0	
1	1	1	0	1	F=1
1	1	1	1	1	

No of select lines $2^n = 8$

$n = 3$

$S_2 = A$, $S_1 = B$, $S_0 = C$

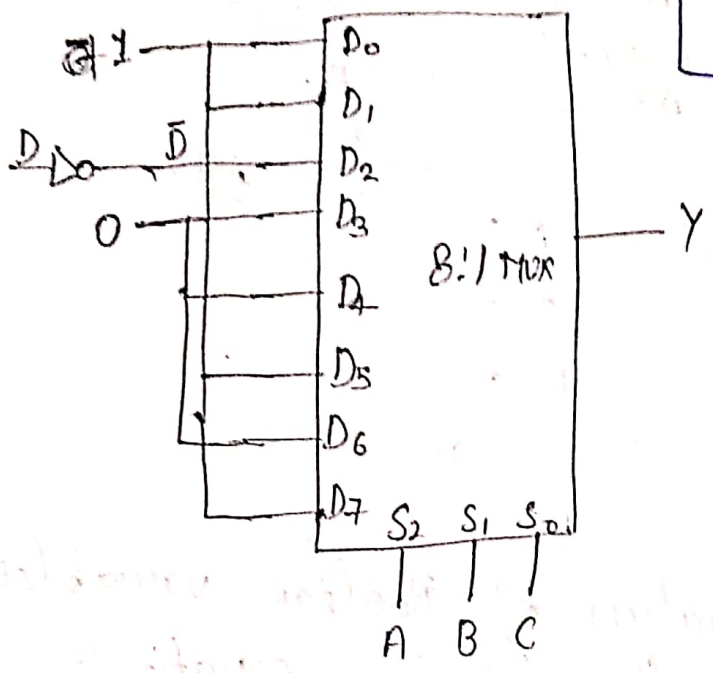
$n = 8$

$m = \log_2 n$ (no. of select lines)

$m = \log_2 8$

$m = \log_2 2^3$

$m = 3$



1 mux

n=4

n=2

16/09/2019

find : values of Boolean variables A,B,C which satisfy the following equation:-

$A+B=1$
$AC=BC$
$A+C=1$
$AB=0$

$\Rightarrow C=0$
$\Rightarrow A=1$
$\Rightarrow B=0$ Ans

check by 7 combination also but it is very complex but we can do it

A+B	0	1
0	0	1
1	1	0

AB	0	1
0	0	0
1	0	0

find values of Boolean variables A, B, C, D which satisfy the following equation:-

$A'+AB=0$
$AB=AC$
$AB+AC'+CD+C'D$

$A=1, B=0, C=0$

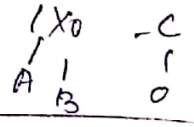
$0+1+0\times1 = 1\times1$

$1=1$

$D=1$

$\boxed{A=1,\ B=0,\ C=0,\ D=1}$ Ans

$A=1, B=0$



$A'+AB = (A'+A)(A'+B) = 1\cdot(A'+B) = 0$

$A'+B=0$

$A'=0 \Rightarrow A=1$

$B=0$

from ②, $AB=AC$

$A=1$ then $B=C$

$\Rightarrow C=0$

Put $A=1, B=0, C=0$ in ③ eqn

$D=1$

Boolean Expression $A \oplus B \oplus A$ is equivalent to

(A) $AB + A'B'$
(B) $A'B + AB'$
(C) B
(D) A

$\oplus$ Competetive & Associative follo

$A \oplus B \oplus A$
$= (A \oplus A) \oplus B$
$= 0 \oplus B$
$= B$ Ans

$F(P,Q) = ((1 \oplus P) \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus (Q \oplus 0))$

(i) $P+Q$
(ii) $(P+Q)'$
(iii) $P \oplus Q$
(iv) $(P+Q)'$

$= ((\bar{1}P + \bar{P}\cdot 1) \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus (Q\bar{0} + 0\bar{Q}))$

$= (\bar{P} \oplus P \oplus Q) \oplus (P \oplus Q \oplus Q)$

$= (\bar{P}P + \bar{P}\bar{P}) \oplus Q \oplus (P \oplus 0)$

$= (P\bar{P} + \bar{P}\bar{P}) \oplus Q \oplus P$ $\quad 1 \oplus (P \oplus Q) \oplus 0$

$= (P + \bar{P}) \oplus Q \oplus P$ $\quad (P+Q)' \oplus 0$

$= 1 \oplus Q \oplus P$ $\quad = (P+Q)'$ Ans

$= 1 \oplus \bar{1}Q + \bar{Q}\cdot 1 \oplus P$

$= \bar{Q} \oplus P$

OR $1 \oplus (P \oplus P) \oplus Q \oplus P \oplus Q \oplus Q \oplus 0$
$1 \oplus 0 \oplus Q \oplus P \oplus 0 \oplus 0$
$1 \oplus Q \oplus P \oplus 0$
$1 \oplus (Q \oplus P)$
$= (Q \oplus P)'$

Let ⊕ and ⊙ denote the exclusive OR and exclusive NOR operation, respectively. Which one of the following is not correct.

$(P \oplus Q)' = P \odot Q$

$P \oplus P = 0$

$P \oplus P' = 1$

$P \odot P = 1$

$P \odot P' = 0$

(a) $(P \oplus Q)' = P \odot Q$

(b) $P' \oplus Q = P \odot Q$

(c) $P' \oplus Q' = P \oplus Q$

(d) $P \oplus P' \oplus Q = P \odot P' \odot Q'$ ✓

(a) $(P \oplus Q)' = (\bar{P}Q + P\bar{Q})' = \overline{\bar{P}Q} \cdot \overline{P\bar{Q}} = (\bar{\bar{P}} + \bar{Q})(\bar{P} + \bar{\bar{Q}})$

$= (P + \bar{Q})(\bar{P} + Q)$

$= P\bar{P} + PQ + \bar{Q}\bar{P} + Q\bar{Q} = PQ + \bar{P}\bar{Q} = P \odot Q$

(b) $P' \oplus Q = \bar{P}Q + \bar{P}\bar{Q} = PQ + \bar{P}\bar{Q} = P \odot Q$

(c) $\bar{P} \oplus \bar{Q} = \bar{P}\bar{Q} + \bar{P}\bar{\bar{Q}} = P\bar{Q} + \bar{P}\bar{Q} = P \oplus Q$

$P \odot Q' = (Q)' = Q$

(d) $P \oplus P' \odot Q = 1 \odot Q = \bar{Q}$

$P \odot P' \odot Q' = (\bar{P}\bar{P} + P\bar{P}) \odot Q'$

$= (\bar{P}P + P\bar{P}) \odot Q'$

$= P\bar{P} \odot Q'$

$= 0 \odot Q'$

$= \bar{0}\bar{\bar{Q}} + 0 \cdot \bar{Q}$

$= Q$

#9 w, x, y and z are boolean variables then which one of the following is incorrect—

(a) $wx + w(x+y) + x(x+y) = x+wy$

(b) $(wx'(y+z')')' + w'x = w' + x + y'z$

(c) $(wx'(y+xz') + w'x')y = xy'$

(d) $(w+y)(wxy+wyz) = wxy + wyz$

$x + wx + wy = x$

(a) $wx + w(x+y) + x(x+y)$

$= wx + wx + wy + x + xy$

$= wx + wy + x + xy$

$= x(w+1) + wy + xy$

$= x + wy + xy$

$= x(1+y) + wy$

$= x + wy$

(b) $(wx'(y+z')')' + w'x$

$= \overline{wx'} + (\overline{y+z}) + w'x$

$= (\bar{w} + \bar{x}) + (\bar{y} \cdot \bar{z}) + w'x$

$= (\bar{w} + x) + (\bar{y}z) + w'x$

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

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

$= \bar{w}(1+x) + x + \bar{y}z$

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

(d) $(w+y)(wxy + wyz)$

$= wwxy + wwyz + wxyy + ywyz$

$= wxy + wyz + wxy + wyz$

$= wxy + wyz$

(c) $(wx'y + w'x')y$

$= wx'yy + w'x'y$

$= wx'y + w'x'y$

$= x'y(w + w')$

$= x'y \cdot 1$

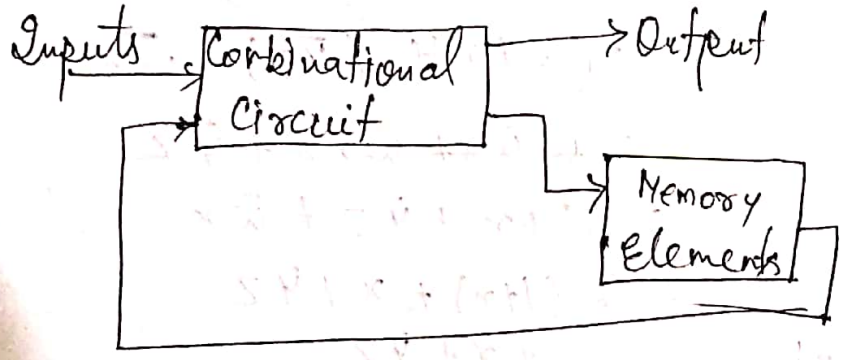
$= x'y$

Digital Circuits

- Combinational
- Sequential
  - Synchronous
  - Asynchronous

In combinational circuit present output depends on present inputs only.

In sequential ckt present outputs depends on previous output as well as present inputs.



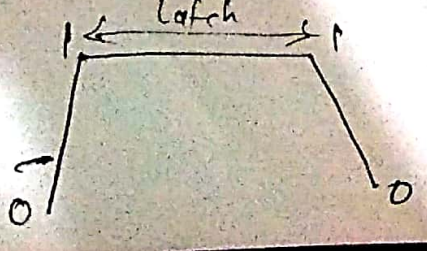
Block diagram of Sequential Circuit

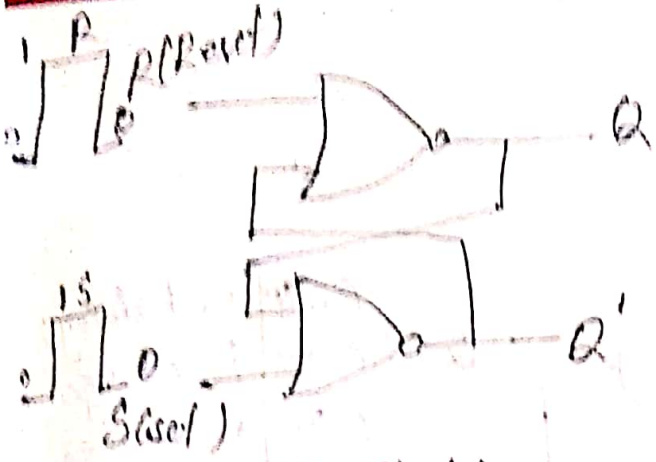
flip-flop :-

A flip-flop circuit can maintain a binary state ~~independently~~ until directed by an input signal to change states.

Latch – It is a storage element that operate with signal levels.

flip-flop – It is controlled by a clock transition.





S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

This is output

SR Latch

RSR flip-flop using NOR Gate

RS - NOR से

SP - NAND से

S=1 Then Set

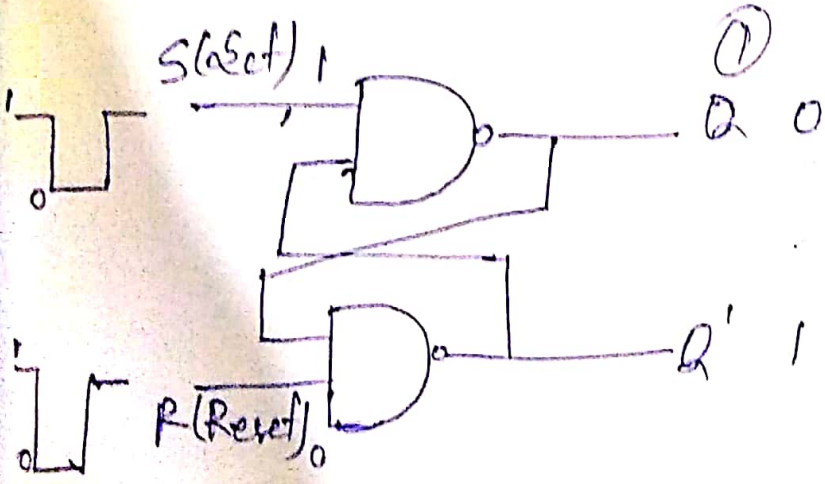
R=1 Then Reset

R	S	Q	Q'
0	0	0	1
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1
1	1	0	0

This is Invalid Input.

forbidden Condition.

SR flip-flop using NAND Gate :-

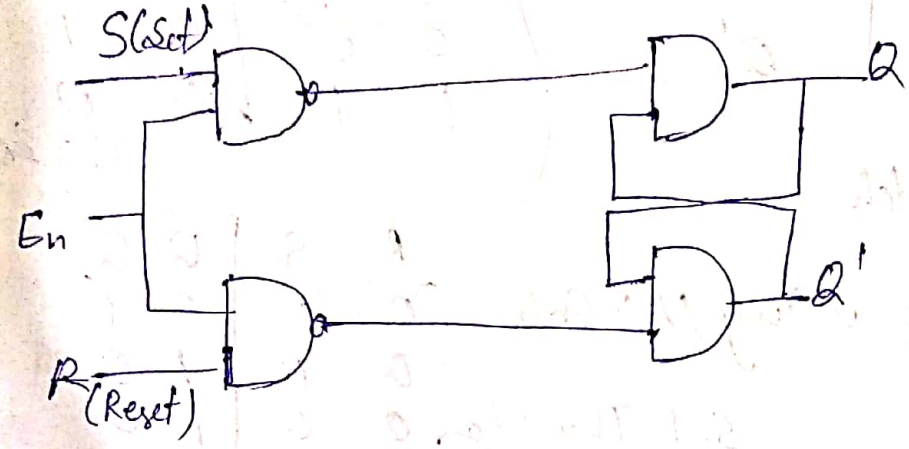


Quit state

S	R	Q	Q'
1	1	0	1
1	0	0	1
#	1	0	1
0	1	1	0
1	1	1	0
#	0	1	

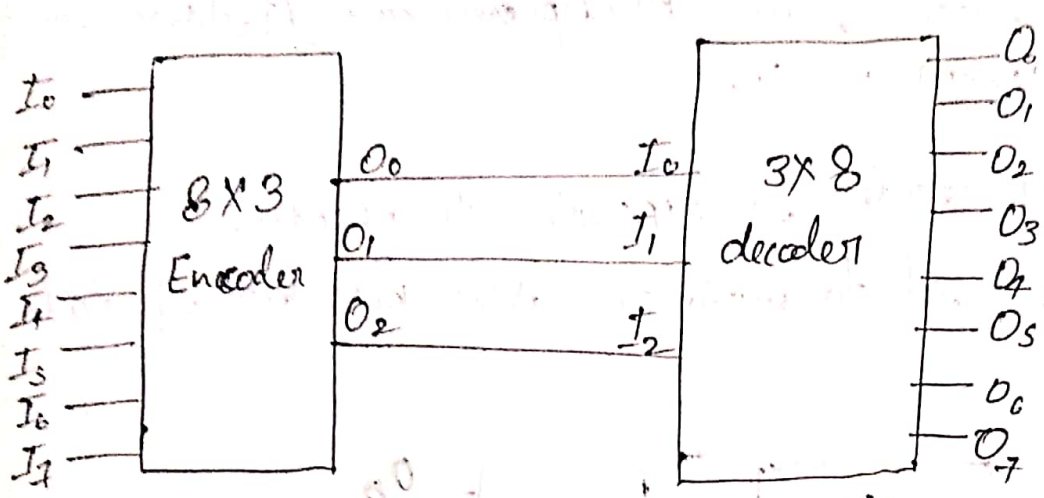
S = 0 Then Set

R = 0 Then Reset



En SR	Next state logic
0 XX	No change
1 00	No change
1 01	Q=0 Reset state
1 10	Q=1 Set state
1 11	Indeterminate

Using Encoder and decoder we reduce the number of data lines since cost also be reduces.



$n = 8$

$n = 2^3$

$\boxed{m = 3}$

$m = 2^3$

$\boxed{m = 8}$

Let I_5 is high = 101 then $O_1 = 1$, $O_2 = 0$, $O_3 = 1$ and again this information will be decode and O_5 will be high.

Type of Encoders :-

There are 4 type of Encoders :-

1. Priority Encoders
   Decimal to BCD Encoder
   Octal to binary Encoder
   Hexadecimal to Binary encoder.

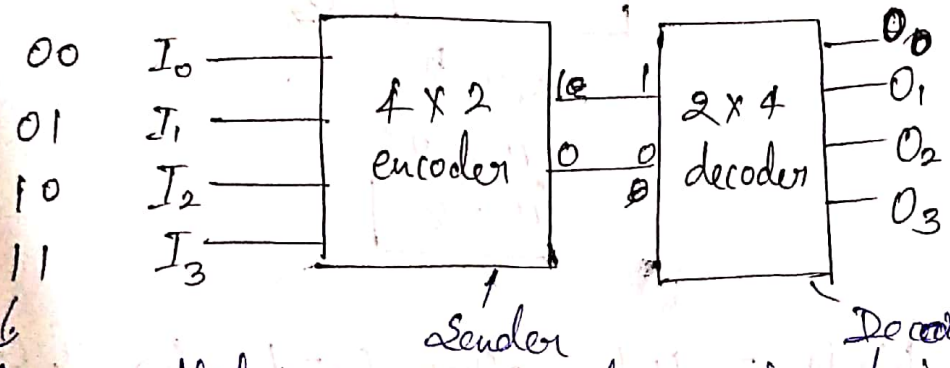
Introduction to Encoders & Decoders

⇒ They are combinational circuits.

⇒ ~~And~~ Encoder, decoder, and multiplexer are Medium scale Integrated (MSI) circuit.

⇒ Function of decoder is opposite to encoder.

⇒ Encoder is used to minimize the number of data lines.



We see that in every case only one input is high and remaining is low then we use only two data line one for high and other for low.

In Encoder ~~Eg~~ no. of input (n) = 4

$n = 4$

$n = 2^m$

Relation b/w input & output. ⇒ $\boxed{n = 2^m}$

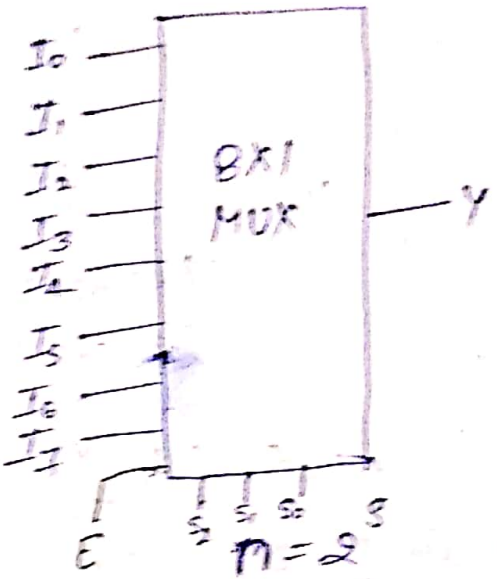
⇒ In case of decoder ~~no of~~ Relation b/w no. of input and no. of output:-

$$\boxed{m = 2^n}$$

Where n = no. of input lin
m = no. of output lin

8x1 MUX:-

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



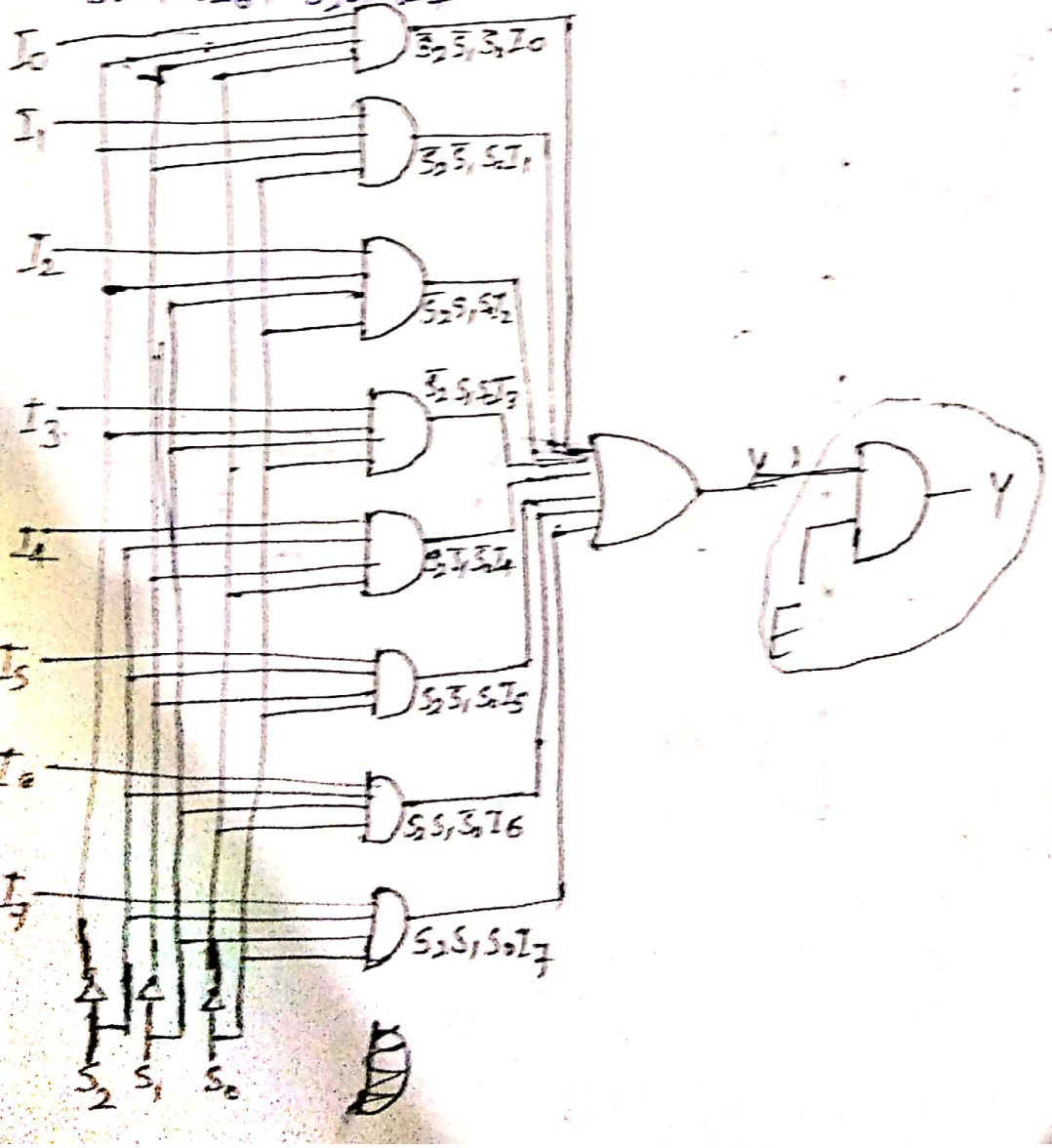
$n = 2^s$

$8 = 2^s$

$2^3 = 2^s$

$\boxed{s = 3}$

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



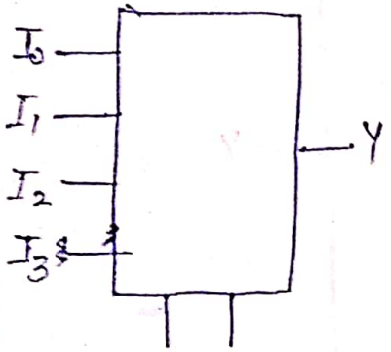
4×1 Multiplexer:-

No. of select lines

$n = 2^S$

$4 = 2^S$

$\boxed{S = 2}$



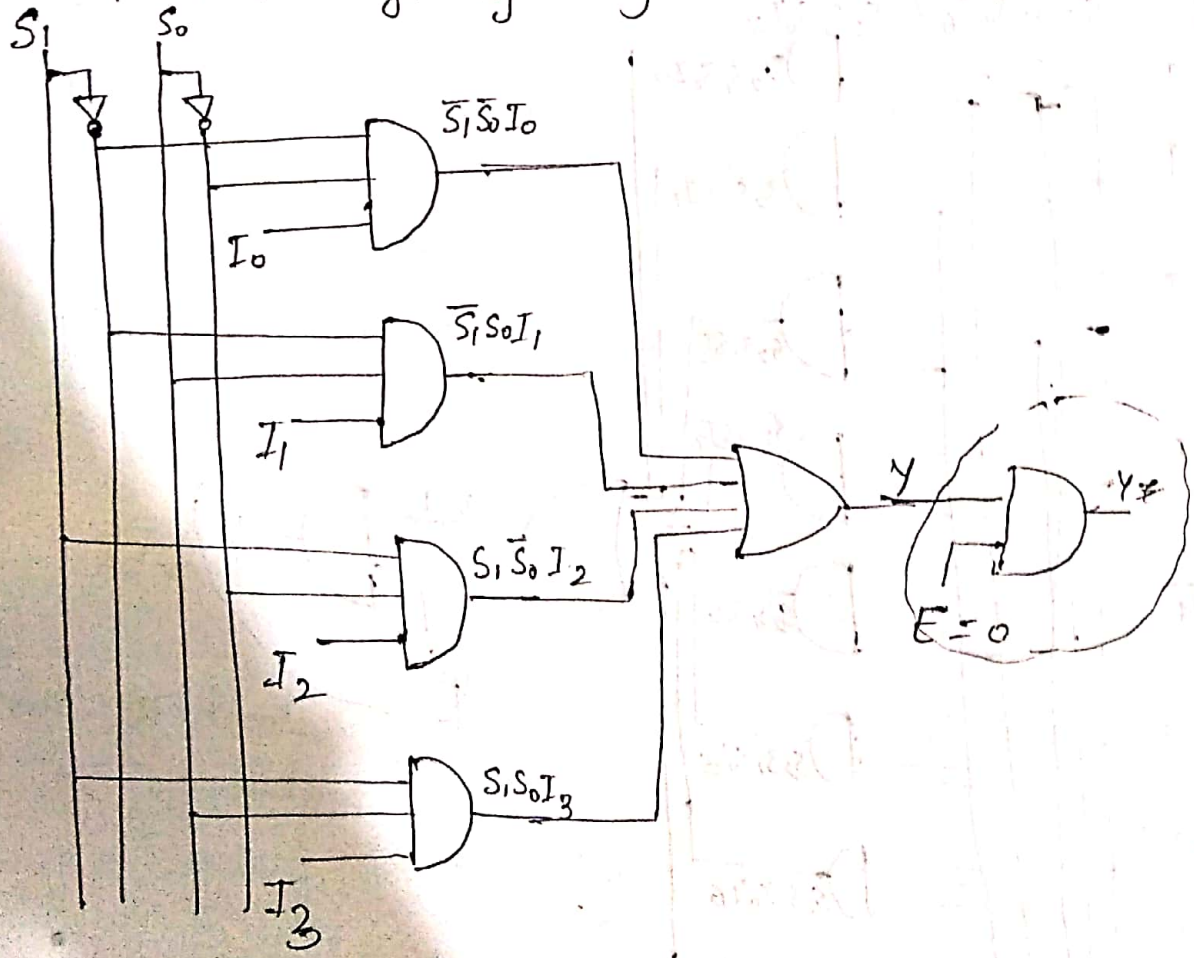
Hence in the case of 4×1 Multiplexer there will be 2 select lines.

S_1 S_0 — Select lines

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

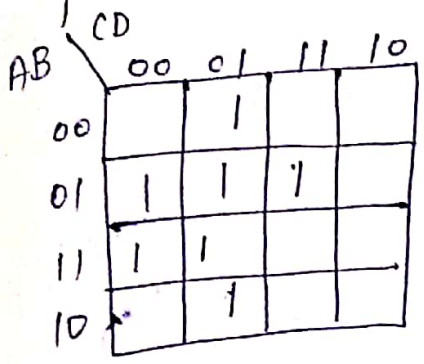
$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$

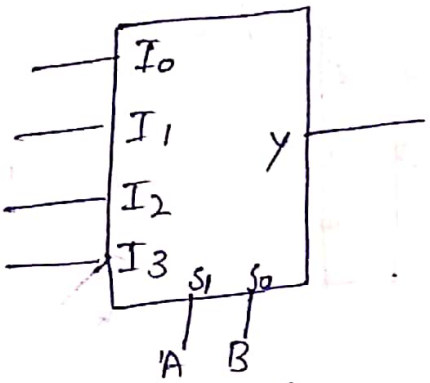
Implement this logic by using the gate:-



Implementing of Boolean function using Multiplexers

Implement:- $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$ using 4X1 MUX.





$S_0 = B$

$S_1 = A$

S_1	S_0	Y
0	0	
0	1	
1	0	
1	1	

Implementing 8X1 MUX using 4X1 MUX. (Special case)

using enable we can implement

$n_1 = 8$

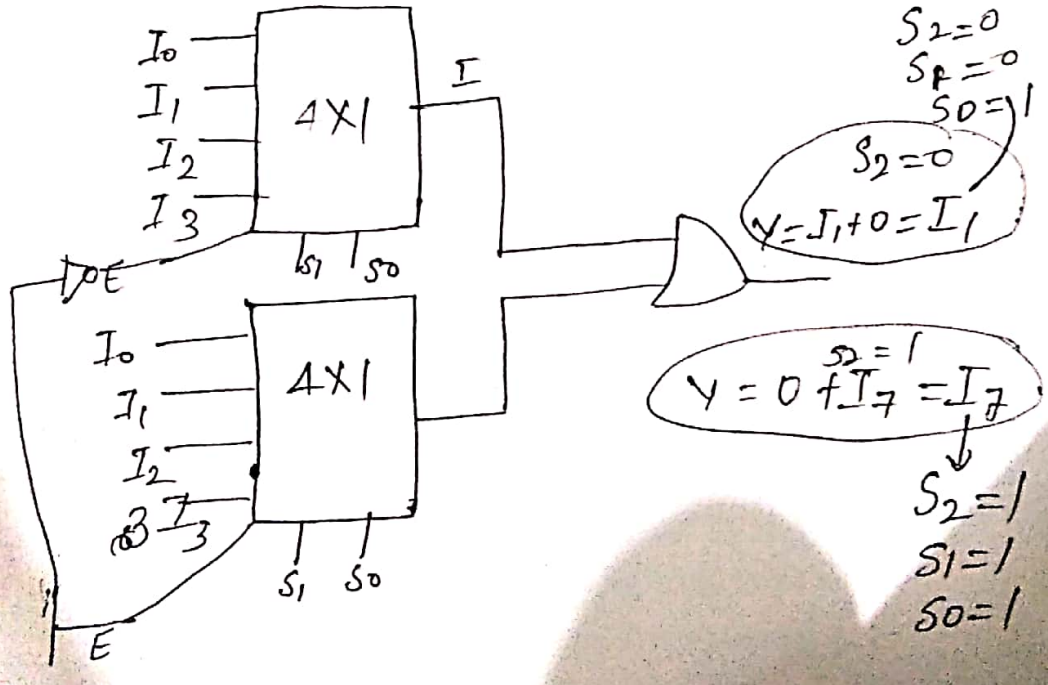
$n_2 = 4$

$\frac{8}{4} = 2$

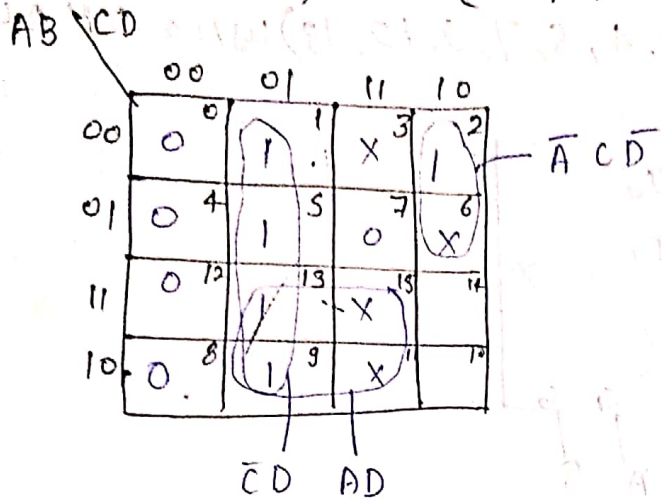
$\frac{2}{4} = 0.5$

We use 2.5 4X1 MUX to implement 8X1 MUX

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



$F(ABCD) = \Sigma m(1,2,5,9,13) + \Sigma d(3,6,11,15)$



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

Complements:-

Find the 2's Complement of the given number:-

(a) 11011

$N = 11011$

$n = 5$

$r = 2$

2's Comp. $= 2^5 - 11011$

$= (32)_{10} - 11011$

$= 100000 - 11011$

$= 00100$ Ans

(b) 1101.01

$n = 4$

$r = 2$

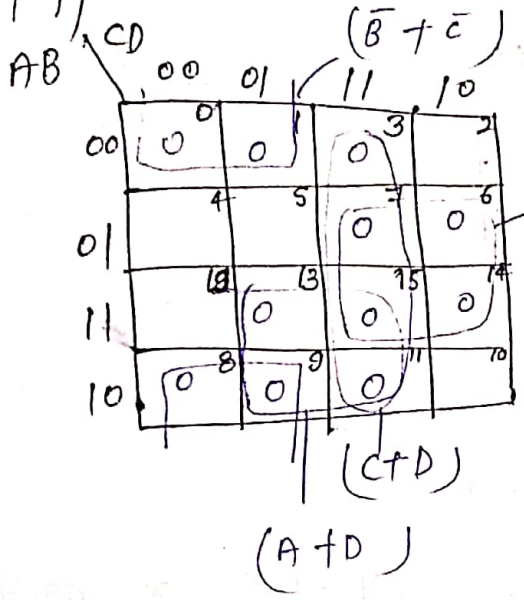
2's comp. $= 2^4 - 1101.01$

$= (16)_2 - 1101.01$

$= 10000 - 1101.01$

$= 0010.11$ Ans

Q⇒ Simplify $F(AB'CD) = \pi M(0,1,3,6,7,8,9,11,13,14,15)$

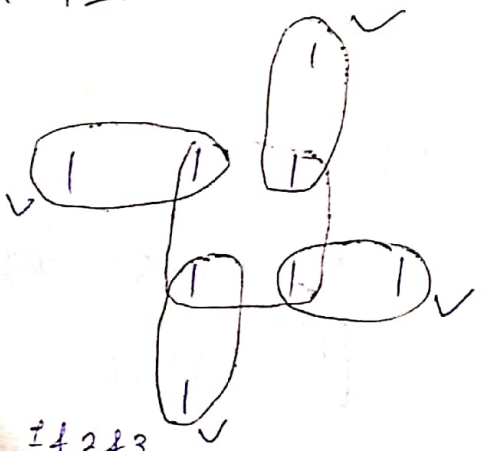


$F = (\bar{B}+\bar{C}) \cdot (B+C) \cdot (C+D) \cdot (A+D)$

Prime Implicant and Essential Prime Implicant:-

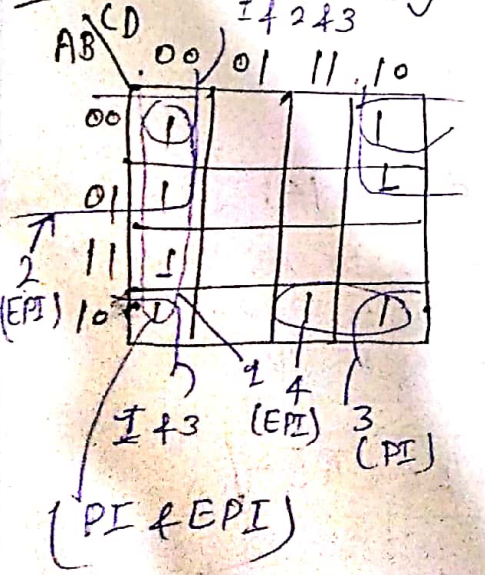
Prime Implicant में बड़ा से बड़ा Pair बनाते हैं

Essential Prime ~~Implicat~~Implicant (EPI) :- At least one minterm which is not covered by other PI.



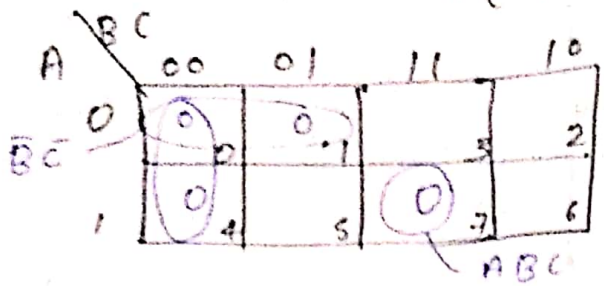
5 PI
4 EPI

Exp:-



4 PI
3 EPI

POS form, $F = \Pi(0, 1, 4, 7)$



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

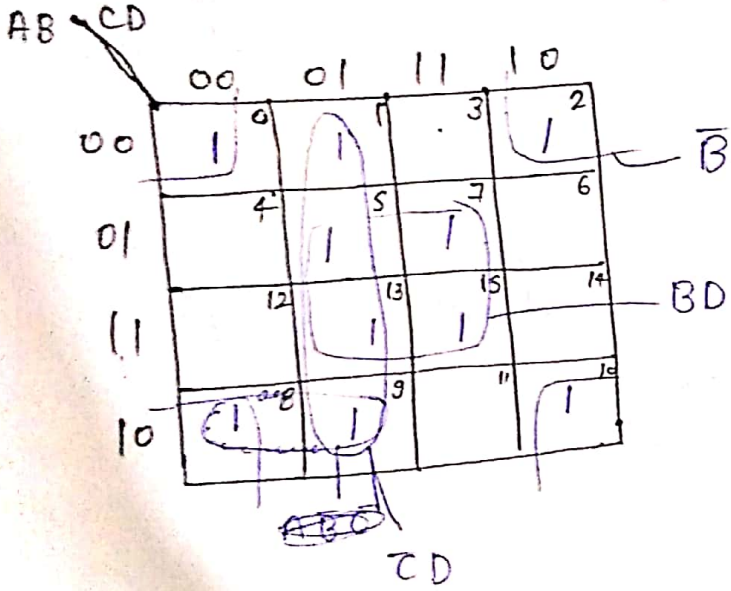
$F = \overline{\bar{B}\bar{C} + \bar{A}\bar{B} + ABC}$

$F = (\overline{\bar{B}\bar{C}}) \cdot (\overline{\bar{A}\bar{B}}) \cdot (\overline{ABC})$

$\boxed{F = (B+C)(A+\bar{B})(\bar{A}+\bar{B}+\bar{C})}$ — we can write this directly.

Q→ Obtain Reduced Expression for

$F(A,B,C,D) = \Sigma m(0,1,2,5,7,8,9,10,13,15)$

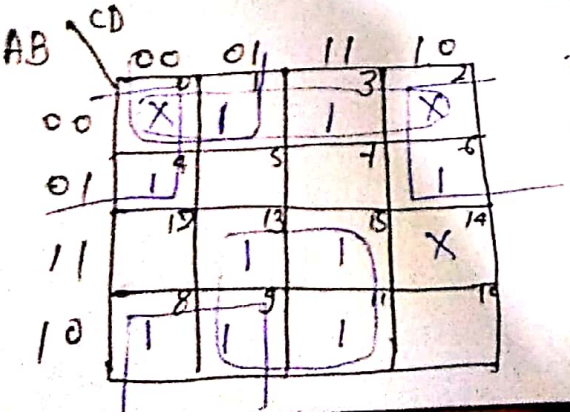


$F = BD + \bar{B}\bar{D} + \cancel{AB\bar{C}}\ \bar{C}D$

$\boxed{F = BD + \bar{B}\bar{D} + \bar{C}D}$

Q→ Minimize with the help of K-map :—

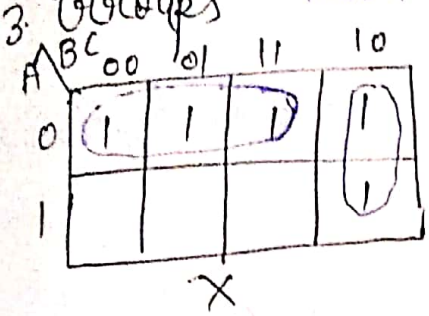
$F(A,B,C,D) = \Sigma m(1,3,4,6,8,9,11,13,15) + \Sigma d(0,2,14)$



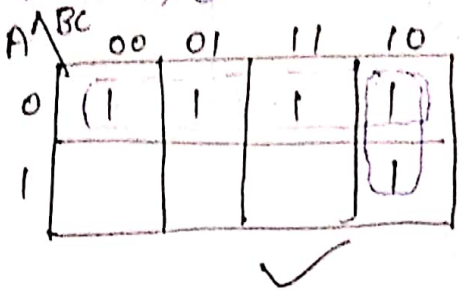
Don't care condition में हम यह नहीं करते हैं कि वहाँ 0 लगा या 1 इसलिए यें 0 और 1 दोनों के साथ Pairing कर सकते हैं।

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

3. Groups must contain 1, 2, 4, 8 or 2^n cells.

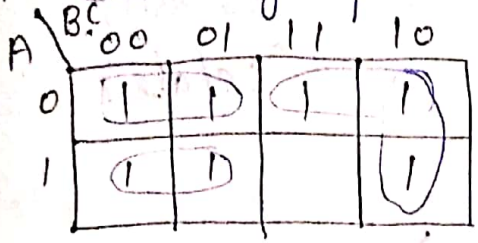


X

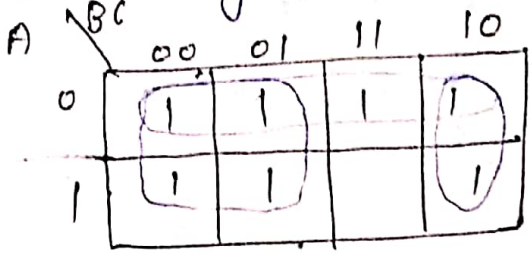


✓

4. Each group should be as large as possible.



X

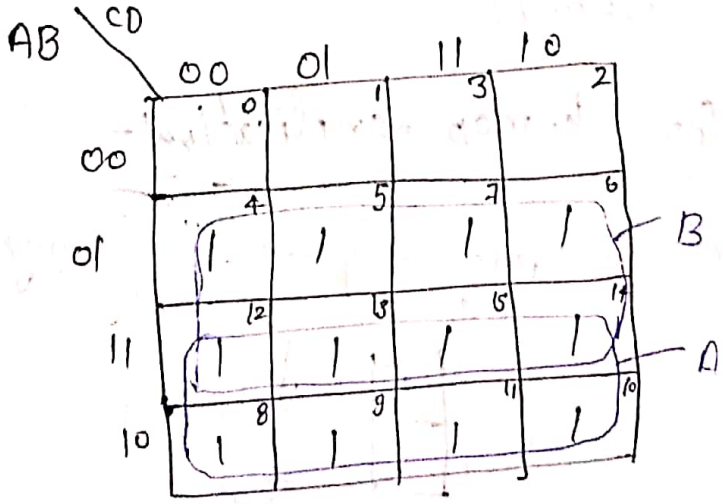


✓

(This simplification does not give minimum literals)

Q. Simplify the following expression using K-map:-

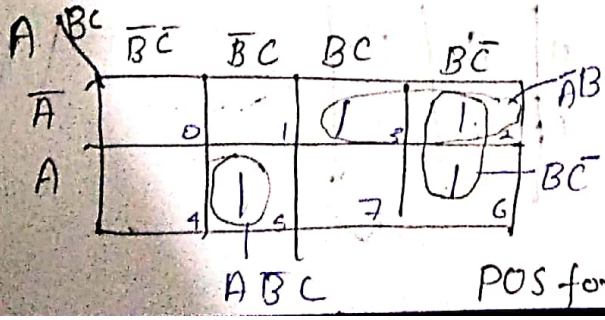
$F(ABCD) = \Sigma m(4,5,6,7,8,9,10,11,12,13,14,15)$



$F = A + B$

Q⇒ Simplify $F(ABC) = \bar{A}BC + B\bar{C} + AB\bar{C} + A\bar{B}C$ using K-map in SOP form and POS form.

$B\bar{C}$ के लिए दोनों columns consider करेंगे।



$F = \Sigma m(2,3,5,6)$

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

This is in SOP form.

POS form ⇒ $F = \Pi(0,1,4,7)$

Grouping of cells for simplification:-

⇒ Adjacent cells which have 1's can be grouped together in 2's power.

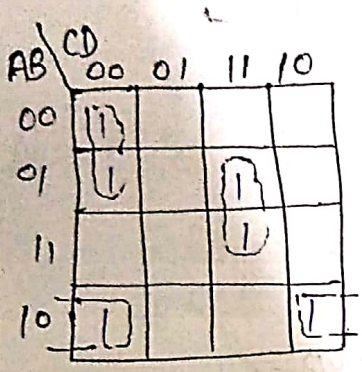
i.e $2^0 = 1$

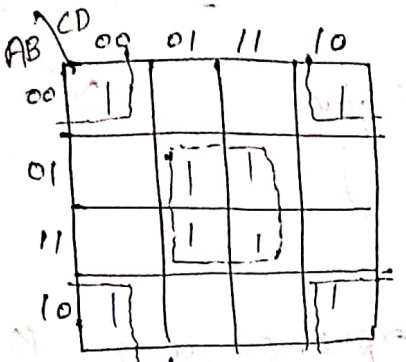
$2^1 = 2$ adjacent cells can be grouped (Pair)

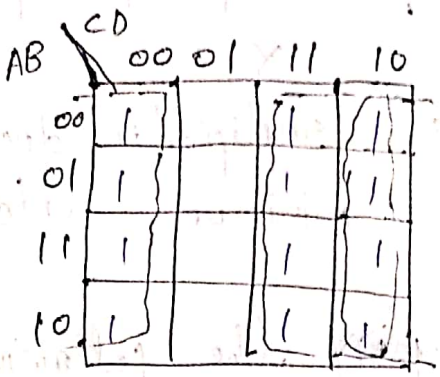
$2^2 = 4$ " " " " " (Quads)

$2^3 = 8$ " " " " " (Octals)

$2^4 = 16$ " " " " "







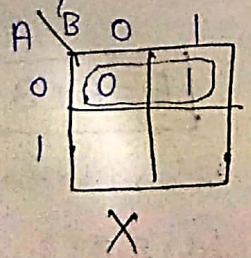
$F = \bar{A}C\bar{D} + BCD + A\bar{B}\bar{D}$

$F = BD + \bar{B}\bar{D}$

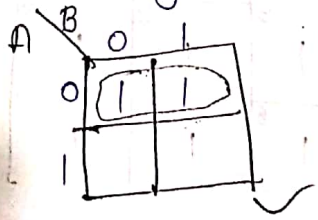
$F = \bar{D} + C$

Rules followed for k-map simplification:-

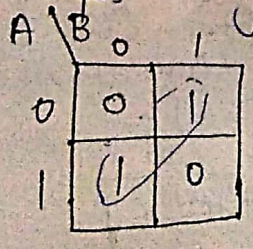
1. Groups do not include only cell containing a zero.



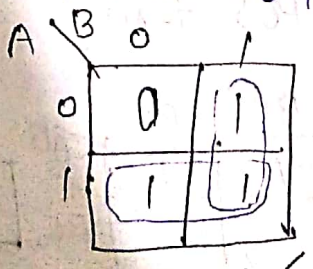
X



2. Groups may be horizontal or vertical, but not diagonal



X



✓