Unit - I

Introduction to Digital Electronics

CO1: Perform basic binary arithmetic & simplify logic expressions.

NUMBER SYSTEM

Introduction:

The number system in which an ordered set of ten symbols - 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9, known as digits - are used to specify any number. This number system is popularly known as the decimal number system. The radix or base of this no. system is 10 (number of distinct digits). Any no. is a collection of these digits.

For example, 1982.365 signifies a number with an integer part equal to 1982 & a fractional part equal to 0.365 seperated from the integer part with a radix point (.) also known as decimal point.

Some of the other commonly used number systems are: binary, octal & hexadecimal number systems. These no. systems are widely used in digital systems like microprocessors, logic circuits, computers etc.

* Number Systems:

A number system defines an ordered set of symbols known as digits with rules defined for performing arithmetic operations like addition, multiplication etc.

A collection of these digits makes a number which in general has two parts - integer & fractional, set apart by a radix point (.) that is

$$(N)_b = \underbrace{d_{n-1}\, d_{n-2} \cdots d_i \cdots d_1 d_0}_{\text{Integer portion}} \cdot \underbrace{d_{-1}\, d_{-2} \cdots d_{-f} \cdots d_{-m}}_{\text{fractional portion}}$$

Radix point

Where N → a number
b → radix or base of the number system
n → no. of digits in integer portion
m → no. of digits in fractional portion
dn-1 → most significant digit (msd)
d-m → least significant digit (lsd)

The digit in a number are placed side by side & each position in the number is assigned a weight or index.

* Common characteristics of all numbering systems:

1) Base or Radix:
The no. of values that a digit (a character) can assume is equal to the base of the system.
It is also called as the radix of the system. It is denoted by 'r'.
eg: for a decimal system, the base is '10' hence every digit can assume 10 values (0, 1, 2, ... 9).

2) The largest value of a digit is always one less than the base:
eg: The largest digit in a decimal system is 9 (one less than the base 10)

3) Each digit position (place) represents a different multiple of base:
i.e. The numbers have positional importance.

$$\times B^4 \; \times B^3 \; \times B^2 \; \times B^1 \; \times B^0 \cdot \times B^{-1} \; \times B^{-2} \; \times B^{-3} \; \times B^{-4}$$

↳ Radix point

where B → Base or Radix

eg: Consider the decimal no. $(349.25)_{10}$

$N = 3\times10^{2} + 4\times10^{1} + 9\times10^{0} \cdot 2\times10^{-1} + 5\times10^{-2}$

4) Column nos.:

The column no. is the no. assigned to the digit placed in relation with the radix pt.

Column nos. to the left of the radix pt. start with 0 & go up (0, 1, 2, ···).

Column nos. to the right of the radix pt. start from -1 & become more & more negative (-1, -2, -3, ····).

Various number systems:

Various no. systems and their bases are as shown below.

1) Binary no. system (Base - 2)
2) Octal no. system (Base - 8)
3) Decimal no. system (Base - 10)
4) Hexadecimal no. system (Base - 16)

* Decimal Number System:

- It uses the base of 10
- The largest value of a digit is 9.
- Each place (column no.) represents a different multiple of 10. These multiples are also called as weighted values.

Most Significant Digit (msd):

The leftmost digit having the highest ~~wate~~ weight is called as the most significant digit of a number.

Least significant digits (lsd):

The rightmost digit having the lowest weight is called as the least significant digit of a number.

* Binary Number system:

The number system with base (or radix) two is known as binary number system.

Only two symbols are used to represent numbers in this system & these are 0 & 1. These are known as bits. This system has the minimum base

It is a positional system i.e. every position is assigned a specific weight.

eg:

$$N = \times 2^{2}\ \times 2^{1}\ \times 2^{0} \cdot \times 2^{-1}\ \times 2^{-2}\ \times 2^{-3}$$

Binary point.

Similar to decimal no. system, the left most bit is known as the most significant bit (MSB) & the right most bit is known as Least significant bit (LSB).

Any no. of 0's can be added to the left of the no. without changing the value of the number.

Bit

The smallest 'unit' of data is defined as a single bit

Size : 1

eg : 0 or 1

The Nibble:

A group of four bit is known as nibble. Size of the nibble is 4 bits.

eg: 0101

The Byte:

A group of 8-bits is known as a byte. The size of the byte is of 8 bits.

eg: 0000 0101

The word:

A group of 16-bits is known as a word. The size of the word is of 16 bits.

eg: 0000 0101 0000 0101

The Double word:

A double word means two words. Therefore a double word quantity is 32 bits. Naturally, this double word can be divided into a higher order word & a lower order word, four bytes, or eight nibbles.

* **Binary numbers from $(0)_{10}$ to $(15)_{10}$:**

Decimal no.	Binary no.	Decimal no.	Binary no.
1 0	0000	8 8	1000
2 1	0001	9 9	1001
3 2	0010	10 10	1010
4 3	0011	11 11	1011
5 4	0100	12 12	1100
6 5	0101	13 13	1101
7 6	0110	14 14	1110
8 7	0111	15 15	1111

★ Conversion from Binary to Decimal :

Any binary number can be converted into its equivalent decimal number using the weights assigned to each bit position.

eg: ① Find the decimal equivalent of the binary no. $(11111)_2$.

Soln: $(11111)_2 = 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 16 + 8 + 4 + 2 + 1$

$= (31)_{10}$

2) Convert the binary number $(1011.01)_2$ into its decimal equivalents.

Soln: $(1011.01)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$

$= 8 + 0 + 2 + 1 + 0 + \frac{1}{2^2}$

$= 8 + 2 + 1 + \frac{1}{4}$

$= (11.25)_{10}$

—x—

Examples :

HW : a) .110101
b) 101101
c) 11111111
d) 11001101
e) 01010011
f) 01011
g) 0011001
h) 101101.10101
i) 1100.1011
j) 1001.0101
k) 0.10101
l) 11111.01
m) 110011.0001
n) 11101.1100

Determine the decimal nos. represented by the given binary nos.

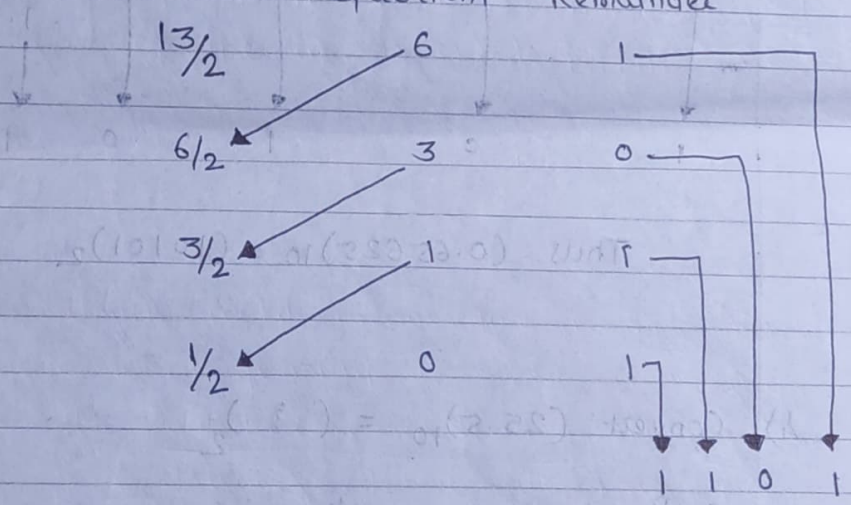
Decimal to Binary conversion:

Any decimal number can be converted into its equivalent binary no. For integers the conversion is obtained by continuous division by 2 (base of binary) & keeping track of the remainders, while for fractional parts the conversion is affected by continuous multiplication by 2 & keeping track of the integers generated. The conversion process is illustrated by the following examples.

Examples:

1) Convert $(13)_{10}$ to an equivalent base-2 no.

∴ Soln:



Thus $(13)_{10} = (1101)_2$

---

2) Convert $(174)_{10} = (\ldots)_2$

Soln:

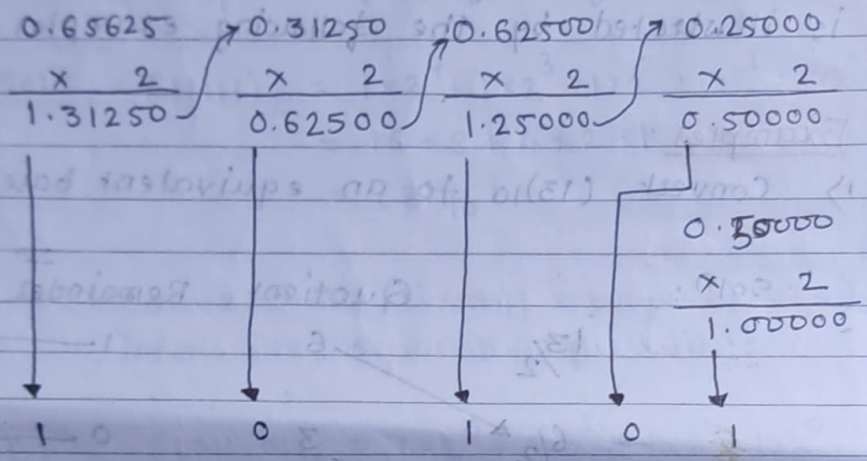
	Quotient	Remainder
174/2	87	0
87/2	43	1
43/2	21	1
21/2	10	1
10/2	5	0
5/2	2	1
2/2	1	0

	Quotient	Remainder
1/2	0	1

$\therefore (174)_{10} = (10101110)_2$

3) Convert $(0.65625)_{10} = (\quad)_2$

Soln:



Thus $(0.65625)_{10} = (10101)_2$

4) Convert $(25.5)_{10} = (\quad)_2$

Soln: a) Integer part

	Quotient	Remainder
25/2	12	1
12/2	6	0
6/2	3	0
3/2	1	1
1/2	0	1

1 1 0 0 1

Thus $(25)_{10} = (11001)_2$

b) Fractional part

0.5
x 2
1.0

$\therefore (0.5)_{10} = (0.1)_2$

Therefore $(25.5)_{10} = (11001.1)_2$

HW

a) $(250)_{10}$
b) $(555)_{10}$
c) $(63)_{10}$
d) $(96)_{10}$
e) $(10.625)_{10}$
f) $(0.6875)_{10}$
g) $(3000.45)_{10}$
h) $(0.42)_{10}$

Signed Binary Numbers:

- **Sign - Magnitude Representation:**

In the decimal number system a plus (+) sign is used to denote a positive number & a minus (-) sign for denoting a -ve number. The plus sign is usually dropped, & the absence of any sign means that the number has +ve value. This representation of numbers is known as signed number.

As is well known, digital circuits can understand only two symbols, 0 & 1, therefore we must use the same symbols to indicate the sign of the number also. Normally an additional bit is used as the sign bit & it is placed as the most significant bit. A '0' is used to represent a positive number & a '1' to represent a negative number.

For example, an 8-bit signed number 01000100 represents a positive number & its value (magnitude) is $(1000100)_2 = (68)_{10}$. The left most '0' (MSB) indicates that the no. is +ve. On the other hand, in the signed binary form, 11000100 represents a negative number with magnitude $(1000100)_2 = (68)$. The left most 1 (MSB) indicates that the no. is -ve. This kind of representation for signed numbers is known as sign-magnitude representation.

Example:

Find the decimal equivalent of the following binary numbers assuming sign magnitude representation of the binary numbers.

a) 101100 b) 001000 c) 0111 d) 1111

Soln:

a) Sign bit is 1, which means no. is -ve

Magnitude = 01100 = $(12)_{10}$

$\therefore (101100) = (-12)_{10}$

b) sign bit is 0, which means no. is +ve

Magnitude = 01000 = 8

$\therefore (001000) = (+8)_{10}$

c) $(0111) = (+7)_{10}$

d) $(1111) = (-7)_{10}$

★ One's (1's) Complement Representation:

In a binary number, if each 1 is replaced by 0 and each 0 by 1, the resulting number is known as the one's complement of the first number.

In fact, both the numbers are complement of each other. If one of the number is +ve then the other no. will be -ve with the same magnitude & vice-versa.

For example, $(0101)_2$ represents $(+5)_{10}$ whereas $(1010)_2$ represents $(-5)_{10}$ in this representation.

Examples:

1) Find the one's complement of the following numbers (binary).

a) 0100111001

Soln:

Given number → 0100111001

One's Complement → 1011000110

b) 11011010

Soln:

Given number → 11011010

One's comple. → 00100101

—×—

2) Represent the following numbers in one's complement form.

a) +7 and -7

⇒ $(+7)_{10} = (0111)_2$

$(-7)_{10} = (1000)_2$

b) +8 and -8

⇒ $(+8)_{10} = (01000)_2$

$(-8)_{10} = (10111)_2$

c) +15 and -15

⇒ $(+15)_{10} = (01111)_2$

$(-15)_{10} = (10000)_2$

—×—

* Two's (2's) Complement Representation

If 1 is added to 1's complement of a binary number, the resulting number is known as the two's complement of the binary number.

For example, 2's complement of 0101 is 1011. Since 0101 represents $(+5)_{10}$, therefore, 1011 represents $(-5)_{10}$ in 2's complement representation.

Examples:

Find 2's complement of the numbers:

1) 01001110 2) 00110101

Soln:

1) Given number → 01001110
   1's complement → 10110001
   Add 1 → 1
   10110010

∴ 2's complement of 01001110 = 10110010

2)
   Given number → 00110101
   1's complement → 10110001

∵ Add 1 to 1's complement

```
  10110001
+         1
----------
  10110010
```

∴ 2's complement of 00110101 = 10110010

—×—

* Binary Arithmetic :

• Binary Addition :

The rules of binary addition are given below:

	A	B	Sum	carry
1)	0	0	0	0
2)	0	1	1	0
3)	1	0	1	0
4)	1	1	0	1
5)	1	1+1	1	1

• Examples :

a) Add the binary numbers.

① 1011 and 1100 ② 0101 and 1111

Soln

$$\begin{array}{r} 1011 \\ +\ 1100 \\ \hline 10111 \end{array}$$

↑ carry

$$\begin{array}{r} 0101 \\ +\ 1111 \\ \hline 10100 \end{array}$$

↑ carry

b) Add the following binary numbers

$A = (10111)_2$ & $B = (11001)_2$

Soln:

$$\begin{array}{r} 10111 \\ +\ 11001 \\ \hline (1)(1)(1)(1)\ 0 \\ 110000 \end{array}$$

$\therefore (10111)_2 + (11001)_2 = (110000)_2$

c) $(12)_{10} + (8)_{10} = (\ldots)_2$

Soln:

	Quotient	Remainder
12/2	6	0
6/2	3	0
3/2	1	1
1/2	0	1

1 1 0 0

$\therefore (12)_{10} = (1100)_2$

	Quotient	Remainder
8/2	4	0
4/2	2	0
2/2	1	0
1/2	0	1

1 0 0 0

$\therefore (8)_{10} = (1000)_2$

∴ Add both the binary numbers

$$\begin{array}{r} 1100 \\ 1000 \\ \hline 10100 \end{array}$$

Thus, $(12)_{10} + (8)_{10} = (1100)_2 + (1000)_2 = (10100)_2$

HW

1) $(101101)_2 + (1111111)_2$

2) 011 & 101

3) $(173)_{10} + (741)_{10}$

- **Binary Subtraction:**

The rules of binary subtraction are given below.

	A	B	Difference	Borrow
1)	0	0	0	0
2)	0	1	1	1
3)	1	0	1	0
4)	1	1	0	0

Examples:

a) Perform the following subtraction.

$(1011)_2 - (0110)_2$

Soln:

$$\begin{array}{r} {\scriptstyle (1)} \\ 1011 \\ -\ 0110 \\ \hline 0101 \end{array}$$

b) $A = (11011)_2$ & $B = (10110)_2$

Soln:

$$\begin{array}{r} 11011 \\ -\ 10110 \\ {\scriptstyle (1)} \\ \hline 00101 \end{array}$$

$\therefore (11011)_2 - (10110)_2 = (00101)_2$

HW

1) $(64)_{10} - (32)_{10}$
2) $(93.5)_{10} - (42.75)_{10}$
3) $(38)_{10} - (29)_{10}$
4) $(9)_{10} - (4)_{10}$

• Binary multiplication

Binary multiplication is similar to decimal multiplication. In binary, each partial product is either 0 (multiplication by 0) or exactly same as the multiplicand (multiplication by 1).

Examples

1) Multiply 1001 by 1101

Soln.

```
    1001
  x 1101
  ------
    1001
   0000x
  1001xx
 1001xxx
 -------
 1110101
```

2) Multiply $(9)_{10}$ by $(8)_{10}$

Soln.

$\because A = (9)_{10} = (1001)_2$

$B = (8)_{10} = (1000)_2$

$\therefore$

```
    1001
  x 1000
  ------
    0000
   0000x
  0000xx
 1001xxx
 -------
 1001000
```

$\therefore (1001)_2 \times (1000)_2 = (1001000)_2$

Binary Division:

Binary division is obtd. using the same procedure as decimal devision.

Examples:

1) Divide 1110101 by 1001

Soln:

```
          1101          ← Quotient
Divisor → 1001 | 1110101  ← Dividend
        - 1001
        -------
          01011
           1001
          ------
          001001
            1001
           ------
            0000
```

$\therefore (1110101)_2 \div (1001)_2 = (1101)_2$

---

2) Divide 110 by 10

Soln:

```
        11
10 | 110
     10
    ---
     010
      10
     ---
      00
```

$\therefore (110)_2 \div (10)_2 = (11)_2$

$\therefore (6)_{10} \div (2)_{10} = (3)_{10}$

---

3) $(110110)_2 \div (101)_2$

Soln:

```
        1010
101 | 110110
       101
      -----
       00111
         101
        ----
         0100
```

$\therefore$ Quotient $= (1010)_2 = (10)_{10}$

Remainder $= (0100)_2 = (4)_{10}$

$\therefore (110110)_2 \div (101)_2 = (1100.1)_2$

HW

1) $(25)_{10} \div (5)_{10}$
2) $(15)_{10} \div (5)_{10}$
3) $(75)_{10} \div (15)_{10}$

* Binary subtraction using 1's & 2's complement:

• Binary subtraction using 1's complement:

Binary subtraction can be performed by adding the 1's complement of B with A. If final carry is 1, then add it to the result of addition obtd. to get the final result. Note that if the final carry is 1 then the subtraction is +ve & in its true form.

If the final carry produced is 0, then the result obtd is -ve & in the 1's complement form. So convert it into the true form.

Examples:

1) Perform $(9)_{10} - (4)_{10}$ using 1's complement method.

Soln: Convert $(4)_{10}$ into 1's complement.

$\therefore (4)_{10} = (0100)_2$

& 1's complement of $(0100)_2 = (1011)_2$

$\therefore (9)_{10} \Rightarrow 1001$

1's complement of $(4)_{10} \Rightarrow +1011$

10100

$\therefore$ Add carry into the final result

$$\begin{array}{r} 0100 \\ +\quad\quad 1 \\ \hline 0101 \end{array}$$

$\rightarrow$ Answer is +ve

2) Subtract $(9)_{10}$ from $(4)_{10}$ using 1's complement method

Soln: Given $A = (4)_{10} = (0100)_2$

$B = (9)_{10} = (1001)_2$

∴ 1's complement of $(9)_{10} = (0110)_2$

∴ Add $(4)_{10}$ & 1's complement of $(9)_{10}$

$$\begin{array}{r} 0100 \\ +\ 0110 \\ \hline 01010 \end{array}$$

↑ Final carry

∴ Take 1's complement of the result

1's complement of $1010 = 0101$

But $(0101)_2 = (5)_{10}$

∴ $(4)_{10} - (9)_{10} = (-5)_{10}$

- HW

1) Perform the following using 1's complement method.

a) $(-4)_{10} - (8)_{10}$

b) $(5)_{10} - (5)_{10}$

c) $(1101)_2 - (1001)_2$

d) $(1011)_2 - (0110)_2$

e) $(85)_{10} - (35)_{10}$

f) $(1011001)_2 - (1101010)_2$

Binary subtraction using 2's complement method:

Binary subtraction can be performed by adding 2's complement B with A. If a final carry is generated, discard the carry & the remaining answer is +ve. If the final carry is 0, the answer is -ve & is in 2's complement form.

Examples

1) perform binary subtraction using 2's complement form.

a) $(7)_{10} - (5)_{10}$

Soln:

$$\begin{array}{r} 7 \\ -5 \\ \hline +2 \end{array} \Rightarrow \begin{array}{r} 0111 \\ 1011 \\ \hline 10010 \end{array} \quad \begin{array}{l} \\ \text{2's complement of (5)} \\ + \end{array}$$

↑ Discard final carry

∴ The answer is 0010 equivalent to $(+2)_{10}$

b) $(5)_{10} - (7)_{10}$

$$\begin{array}{r} 5 \\ -7 \\ \hline -2 \end{array} \Rightarrow \begin{array}{r} 0101 \\ 1001 \\ \hline 1110 \end{array} \quad \begin{array}{l} \\ \rightarrow \text{2's complement of (7)} \\ \end{array}$$

∴ Final carry is 0, therefore the answer is -ve & in its two complement form.

∴ 2's complement of 1110 = 0010

∴ The answer is $(-2)_{10}$

—×—

HW

2) Perform the following operations using 2's complement method.

a) 48 - 23
b) 23 - 48
c) 48 - (-23)
d) -48 - 23
e) 9 - 5
f) -4 - (-6)
g) (11010) - (10000)

★ Octal Number System:

The number system with base (or radix) eight is known as the octal number system. In this system eight symbols, 0, 1, 2, 3, 4, 5, 6 and 7 are used to represent numbers. Similar to decimal & binary number system, it is also a positional system & has two parts: integer & fractional, set apart by a radix (octal) point.

eg: $(6327.4051) = 6\times8^3 + 3\times8^2 + 2\times8^1 + 7\times8^0 + .4\times8^{-1} + 0\times8^{-2} + 5\times8^{-3} + 1\times8^{-4}$

$= 3072 + 192 + 16 + 7 + \frac{4}{8} + 0 + \frac{5}{512} + \frac{1}{4096}$

$= (3287.5100098)_{10}$

Thus, $(6327.4051)_8 = (3287.5100098)_{10}$

Using the above procedure, an octal number can be converted into an equivalent decimal no. or base-8 no. can be converted into into an base-10 no.

Example:

a) Convert $(247)_{10}$ into octal

Soln:

	Quotient	Remainder
247/8	30	7
30/8	3	6
3/8	0	3

3 6 7

Thus $(247)_{10} = (367)_8$

—×—

b) Convert $(0.6875)_{10} = (\ ?\ )_8$

Soln:

$$0.6875 \times 8 = 5.5000 \rightarrow 5$$

$$0.5000 \times 8 = 4.0000 \rightarrow 4$$

Thus $(0.6875)_{10} = (0.54)_8$

—×—

c) Convert (3000.45) into octal

Soln:

Integer part:

	Quotient	Remainder
3000/8	375	0
375/8	46	7
46/8	5	6
5/8	0	5

5 6 7 0

$\therefore (3000)_{10} = (5670)_8$

Fractional part:

0.45	0.60	0.80	0.40
× 8	× 8	× 8	× 8
3.60	4.80	6.40	3.20
↓	↓	↓	↓
3	4	6	3

$\therefore (0.45)_{10} = (0.3463)_8$

Combine both the results

$\therefore (3000.45) = (5670.3463)_8$

★ Octal to Binary Conversion:

Octal numbers can be converted into equivalent binary numbers by replacing each octal digit by its 3-bit equivalent binary

eg: Convert $(736)_8$ into binary

$(736)_8 =$ 7 3 6

↓ ↓ ↓

111 011 110

$\therefore (736)_8 = (111011110)_2$

Practice problems:

1) $(0.514)_8 = (\quad)_2$
2) $(116)_8 = (\quad)_2$
3) $(555)_8 = (\quad)_2$
4) $(496)_8 = (\quad)_2$
5) $(1525)_8 = (\quad)_2$

* **Binary to octal conversion:**

Binary numbers can be converted into equivalent octal numbers by making groups of three bits starting from LSB & moving towards MSB for integer part of the no. & then replacing each group of three bits by its octal representation. For fractional part, the groupings of three bits are made starting from the binary point.

Examples:

1) Convert $(1001110)_2$ to its octal equivalent

Soln:

$$(1001110)_2 = \underbrace{001}_{1}\ \underbrace{001}_{1}\ \underbrace{110}_{6}$$

$$= (116)_8$$

2) $(0.10100110)_2 = 0.\underbrace{101}_{5}\underbrace{001}_{1}\underbrace{100}_{4}$

$$= (0.514)_8$$

Practice problems:

1) $(1011.110)_2 = (\quad)_8$
2) $(11010010)_2 = (\quad)_8$
3) $(11111110)_2 = (\quad)_8$
4) $(0.11111000)_2 = (\quad)_8$
5) $(11100100.11)_2 = (\quad)_8$

—— × ——

★ Hexadecimal Number System:

The base for hexa decimal number system is 16 which requires 16 distinct symbols to represent the numbers. These are numerals 0 through 9 & alphabets A to F. Since numeric digits & aphabets both are used to represent the digits in the hexadecimal number system, therefore, this is an alphanumeric number system.

★ Hexadecimal to Decimal Conversion:

Hexa decimal number can be converted to their equivalent decimal numbers.

eg: ↑Find Decimal equivalent of hexadecimal no. $(3A)_{16}$

$$\therefore (3A)_{16} = 3\times16^{1} + 10\times16^{0}$$
$$= 48 + 10$$
$$= (58)_{10}$$

$$(0.2F)_{16} = 2\times16^{-1} + 15\times16^{-2}$$
$$= \frac{2}{16} + \frac{15}{16^2}$$
$$= (0.1836)_{10}$$

$$\therefore (3A.2F)_{16} = (58.1836)_{10}$$

HW

1) $(5C7)_{16} = (\quad)_{10}$
2) $(F)_{16} = (\quad)_{10}$
3) $(0.12E)_{16} = (\quad)_{10}$

★ Decimal to Hexa-decimal conversion:

For conversion from decimal to hexa-decimal, the procedure used in binary as well as octal systems is applicable, using 16 as the dividing (for integer part) and multiplying (for fractional part) factor.

Examples:

1) Convert the following decimal numbers into hexadecimal nos.

a) $(95.5)_{10}$

Soln:

Integer part:

	Q	R
95/16	5	15
5/16	0	5

5 15

$\therefore$ Thus, $(95)_{10} = (5F)_{16}$

Fractional part:

$$\begin{array}{r} 0.5 \\ \times\ 16 \\ \hline 8.0 \end{array}$$

↓

8

Thus $(0.5)_{10} = (0.8)_{16}$

Therefore, $(95.5)_{10} = (5F.8)_{16}$

—x—

Practice problems:

1) $(1708)_{10} \rightarrow (\ )_{16}$
2) $(15)_{10} \rightarrow (\ )_{16}$
3) $(965)_{10} \rightarrow (\ )_{16}$
4) $(93.50)_{10} \rightarrow (\ )_{16}$
5) $(675.625)_{10} \rightarrow (\ )_{16}$

★ **Binary to Hexadecimal conversion:**

Binary numbers can be converted into their equivalent hexadecimal numbers by making groups of four bits starting from LSB & moving towards MSB for integer part & then replacing each group of four bits by its hexadecimal representation.

For the fractional part the above procedure is repeated starting from the bit next to the binary point & moving towards the right.

Example:

Convert the following binary numbers to their equivalent hex numbers.

a) 1010011010111

Soln: $(1010011010111)_2 = \underbrace{0010}_{2}\ \underbrace{1001}_{9}\ \underbrace{1010}_{A}\ \underbrace{1111}_{F}$

$\therefore (1010011010111)_2 = (29AF)_{16}$

—x—

b) $(0.0001111010110 1)_2 = 0.\underbrace{0001}_{1}\ \underbrace{1110}_{E}\ \underbrace{1011}_{B}\ \underbrace{0100}_{4}$

$\therefore (0.0001111010110 1)_2 = (1EB4)_{16}$

Practice problems

1) $(10010010)_2 = (\quad)_{16}$
2) $(11101101)_2 = (\quad)_{16}$
3) $(11101.0001)_2 = (\quad)_{16}$
4) $(0001.01101)_2 = (\quad)_{16}$

* Hexadecimal to Binary conversion:

Hexadecimal numbers can be converted into their equivalent binary numbers by replacing each hex digit by its equivalent 4-bit binary no.

Examples

1) Convert $(2F9A)_{16}$ into equivalent binary number.

$\therefore (2F9A)_{16} =$ 2 F 9 A
↓ ↓ ↓ ↓
0010 1111 1001 1010

$\therefore (2F9A)_{16} = (0010\,1111\,1001\,1010)_2$

Practice problems:

1) $(5C7)_{16} = (\quad)_2$
2) $(4A97)_{16} = (\quad)_2$
3) $(99A)_{16} = (\quad)_2$
4) $(3000.9F)_{16} = (\quad)_2$
5) $(4F9)_{16} = (\quad)_2$

— x —

★ Conversion from Hex to Octal:

Hexadecimal numbers can be converted to equivalent octal numbers by converting the hex number to equivalent binary & then to octal.

Examples

Convert the following hex numbers to octal no.

a) A72E b) 0.BF85

Soln:

a) $(A72E)_{16}$ = A 7 2 E

= 1010 0111 0010 1110

$= (1010011100101110)_2$

$\therefore (1010011100101110)_2 = 001\ 010\ 011\ 100\ 101\ 110$

(1 2 3 4 5 6)

$\therefore (A72E)_{16} = (123456)_8$

b) $(0.BF85)_{16} = (0.1011\ 1111\ 1000\ 0101)_2$

$= 0.101\ 111\ 111\ 000\ 010\ 100$

(5 7 7 0 2 4)

$= (0.577024)_8$

★ Conversion from Octal to Hex:

Octal numbers can be converted to equiv. hex numbers by converting the octal no. to equiv. binary & then to hex.

eg: Convert $(247.36)_8$ to equiv hex number.

Soln: $(247.36)_8 = (010\ 100\ 111 \cdot 011\ 110)$

$= (01010\ 0111 \cdot 0111\ 1000)_2$

$= (A7.78)_{16}$

CODES

When numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or coord is being encoded. And the groups of symbols is called as the code.

The digital data is represented, stored & transmitted as group of binary bits. This group is also called as binary code.

Codes are also used for error detection & error correction in digital systems.

Some of the commonly used codes are:

* Binary coded Decimal (BCD) code:

BCD is the short form of "Binary Coded Decimal". In this code each decimal digit is represented by a 4-bit binary no.

Thus BCD is a way to express each of the decimal digit with a binary code.

The positional weights associated to the binary bits in BCD code are 8-4-2-1 with 1 corresponding to LSB & 8 corresponding to MSB. These weights are actually 2^3, 2^2, 2^1 & 2^0.

* Conversion from decimal to BCD:

The decimal digits 0 to 9 are converted into a BCD, exactly in the same way as binary. For example, decimal 4 corresponds to the ~~decimal numbers~~ 0100 BCD which is same as binary. Similarly the BCD numbers corresponding to the decimal numbers 0 to 9 are exactly same as the corresponding binary nos.

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

* Conversion of big decimal numbers to BCD:

Each individual decimal digit of a decimal number is represented by this four bit code (Binary) individually. For example,

In BCD, $(23)_{10} = (0010\ 0011)_{BCD}$

& In Binary, $(23)_{10} = (10111)_2$

★ Excess-3 code:

Excess-3 is also called as XS-3 code. It is a nonweighted code used to express decimal nos.

Excess-3 code words are derived from the 8421 BCD code words by adding $(0011)_2$ or $(3)_{10}$ to each code word in 8421.

Examples:

1) Obtain XS-3 code for $(428)_{10}$

Sol^n:

Given number	4	2	8
	0100	0010	1000
+3 →	0011	0011	0011
	0111	0101	1011

∴ XS-3 equivalent = 0111 0101 1011

2) XS-3 of $(5)_{10}$

∴ $(5)_{10}$ = 0101 + 0011 = 1000

—×—

Excess-3 codes of decimal nos.

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:

Gray code is another non-weighted code. It is not an arithmetic code. It has a very special feature that only one bit will change, each time the decimal no. is increamented. Therefore it is called as unit distance code.

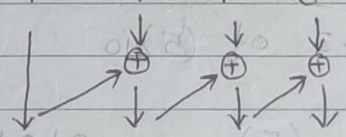
★ Gray to Binary conversion:

For conversion some steps are given below

1) The MSB of gray & binary are same. So write it directly
2) Add binary MSB to the next bit of gray code. Record the result & ignore the carry
3) Continue this process until the LSB is reached.

eg: Convert gray 1110 to binary

Soln: Gray code: 1 1 1 0



Binary: 1 0 1 1

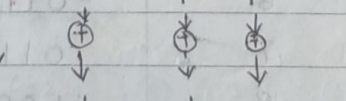
$\therefore (1110)_{gray} = (1011)_2$

★ Binary to Gray conversion:

1) Record the MSB as it is
2) Add this bit to the next position, recording the sum & neglecting any carry.
3) Record successive sums until completed.

eg: convert 1011 binary to gray.

Binary : 1 0 1 1



Gray 1 1 1 0

$\therefore (1011)_2 = (1110)_{gray}$.

LOGIC GATES

In the decimal system, the arithmetic operations such as addition, subtraction, multiplications, division, square, square root are used to solve arithmetic operations (equations).

In the middle of 19th century, an English mathematician George Boole developed rules for manipulations of binary variables, known as Boolean Algebra & it is applied in the design & analysis of digital systems.

★ Boolean Algebra:

Boolean Algebra is used to analyze & simplify the digital (logic) CKTs. There are some rules to be followed while using a Boolean algebra, they are,

1) Variables used can have only two values. Binary 1 for HIGH and binary 0 for LOW.
2) Complement of a variable is represented by a overbar (-). Thus complement of a variable B is represented as $\bar{B}$. Thus if $B=0$ then $\bar{B}=1$ and if $B=1$ ($\bar{B}=0$)
3) ORing of the variables is represented by a plus (+) sign between them such as $(A+B+C)$.
4) Logical ANDing of the two or more variables is represented by writting a dot betⁿ them such as $(A \cdot B \cdot C)$

★ Boolean theorems:

The boolean algebraic theorems are given below. From these theorems, we observe that the even numbered theorems can be obtd. from their preceding odd numbered theorems by interchanging + and · sign,

& interchanging 0 & 1, Theorems which are relat-ed in this way are called duals

Thm No.	Theorem	Thm No.	Theorem
1	$A+0=A$	2	$A \cdot 1 = A$
3	$A+1=1$	4	$A \cdot 0 = 0$
5	$A+A=A$	6	$A \cdot A = A$
7	$A+\bar{A}=1$	8	$A \cdot \bar{A} = 0$
9	$A(B+C)=AB+AC$	10	$A+BC=(A+B)(A+C)$
11	$A+AB=A$	12	$A(A+B)=A$
13	$A+\bar{A}B=(A+B)$	14	$A(\bar{A}+B)=AB$
15	$AB+A\bar{B}=A$	16	$(A+B)(A+\bar{B})=A$
17	$AB+\bar{A}C=(A+C)(\bar{A}+B)$	18	$(A+B)(\bar{A}+C)=AC+\bar{A}B$
19	$AB+\bar{A}C+BC=AB+\bar{A}C$	20	$(A+B)(\bar{A}+C)(B+C)=(A+B)(\bar{A}+C)$
21	$\overline{(A \cdot B \cdot C \cdots)}=\bar{A}+\bar{B}+\bar{C}+\cdots$	22	$\overline{A+B+C+\cdots}=\bar{A} \cdot \bar{B} \cdot \bar{C} \cdots$

★ **Boolean Laws:**

Some boolean laws are,

1) Commutative law
2) Associative law
3) Distributive law
4) AND law
5) OR law
6) INVERSION LAW.

1) **Commutative Law:**

Any binary operation which satisfies the following expression is refereed to as commutative operation.

a) $A \cdot B = B \cdot A$

b) $A+B = B+A$

2) Associative law:

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

a) $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

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

Similarly we can verify the other statement i.e.

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

3) Distributive law:

The distributive law states that,

$$A \cdot (B + C) = AB + AC$$

4) AND laws:

Thes laws use the AND operation therefore they are called as "AND" laws. The AND laws are as follows:

a) $A \cdot 0 = 0$

b) $A \cdot 1 = A$

c) $A \cdot A = A$

d) $A \cdot \bar{A} = 0$

5) OR laws:

These laws use the OR operation. Hence they are called as OR laws. The OR laws are as follows:

a) $A + 0 = A$

b) $A + 1 = 1$

c) $A + A = A$

d) $A + \bar{A} = 1$

6) INVERSION LAW:

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself i.e.

$$\overline{\overline{A}} = A$$

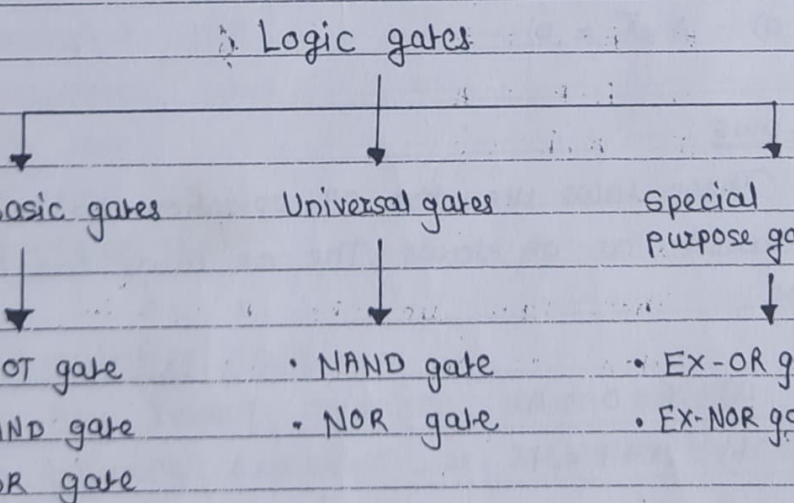
★ Logic gates:

"A logic gate is an electronic ckt having one or more than one i/p's & only one output."

The relationship betⁿ the input & the output is based on a "certain logic". Based on this logic, the gates are named as NOT gate, AND gate, OR, NAND, NOR etc.

★ Classification of Logic gates:

Logic gates are classified into three categories:



Definitions:

1) **Truth table:**

The operation of a logic gate can be best understood with the help of a table called "Truth table".

The truth table consist of all possible combinations of the i/p's & the corresponding state of output of a logic gate.

2) **Boolean Expression:**

The relation between the i/p's & the o/p's of a gate can be expressed mathematically by means of the Boolean expression.

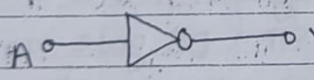
★ NOT gate or Inverter:

Fig. shows a NOT gate, which is also known as an inverter. It has one input (A) and one o/p (Y). It logic eq^n is written as

$$Y = \text{NOT } A$$

$$\boxed{Y = \overline{A}}$$

Symbol:



Truth table

Input	Output
A	$Y = \overline{A}$
0	1
1	0

The NOT operation is also referred to as an inversion or complementation. The bubble (○) in the symbol of a NOT gate indicates inversion.

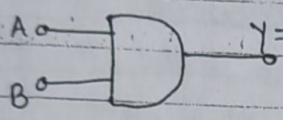
☆ AND gate:

Fig below shows the logical symbol & truth table of two i/p AND gate. A & B are the i/p's while Y is the o/p. AND gate can have two or more i/p's but only one o/p.

The AND operation is defined as: the o/p is 1 if and only if all the i/p's are 1.

The boolean exp^{z} for an AND gate is

$$Y = A \cdot B$$



a) Symbol

Input's		Output
A	B	Y=A·B
0	0	0
0	1	0
1	0	0
1	1	1

b) Truth table

☆ OR gate

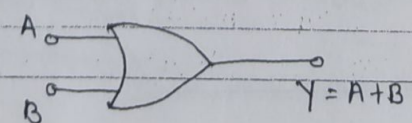
Fig. below shows the logical symbol and truth table of two i/p OR gate. A & B are the i/p's while Y is the o/p. OR gate can have two or more i/p's but only one o/p.

An OR gate performs the logical addn on its i/p therefore its o/p will be high (1) if any one or both the i/p's are high (1).

The boolean exp^{z} for an OR gate is

$$Y = A + B$$

Symbol :



Truth table:

Inputs		output
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

★ NAND gate:

The NOT-AND is known as the NAND operation. It can be implemented with the combination of an AND gate & a NOT gate.

Thus a NAND gate is equivalent to an AND gate followed by an inverter as shown in fig (b)

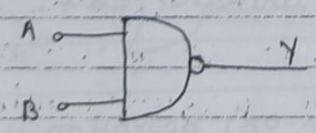
The symbol of a two i/p NAND gate is shown in fig (a) where a bubble (o) on the o/p side represents inversion.

The truth table of a two i/p NAND gate is shown in fig (c) which shows that the o/p is low (0) if and only if both the i/p's are high (1)

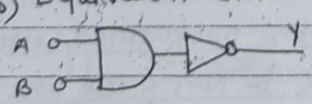
The boolean exp^{2} for the NAND gate is

$$Y = \overline{A \cdot B}$$

a) Symbol



b) Equivalent ckt.



c Truth table

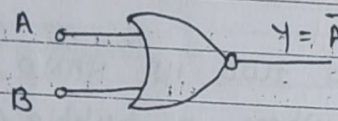
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

A NAND gate is called as universal gate because we can construct NOT, AND & OR gate using only NAND gate.

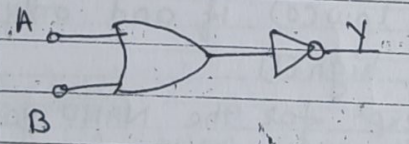
★ NOR gate:

The NOT-OR operation is known as the NOR operation. It is implemented with the combination of OR gate & a NOT gate. Thus a NOR gate is equivalent to an OR gate followed by an inverter as shown in fig (b). The symbol of a NOR gate is shown in fig (a) & truth table in fig (c) which shows that the o/p of a NOR gate is high (1) if & only if all its i/p's are low (0).

a) Logical symbol



b) Equivalent ckt



c) Truth table

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

The boolean exp^2 is

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

The NOR gate is called as universal gate.

★ EX-OR gate:

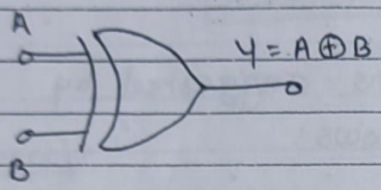
The EXCLUSIVE-OR (EX-OR) operation is widely used in digital ckts. It is not a basic operation & can be performed using the basic gates - AND, OR & NOT gate or universal gates NAND or NOR. Because of its importance a standard symbol shown

in fig is used for this operation.

Fig (a) shows the two i/p & one o/p EX-OR gate & fig (b) shows its truth table which shows when both the i/p's are identical (A=B) then the o/p is low.

The boolean exp^{n} is $\boxed{Y = A \oplus B = \bar{A}B + A\bar{B}}$

a) Symbol



b) Truth table

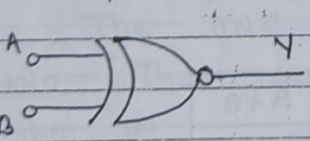
A	B	Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

★ EX-NOR gate:

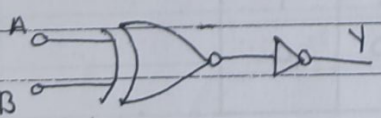
The word EX-NOR is a short form of EXCLUSIVE-NOR. EX-NOR is equivalent to an EX-OR gate followed by a NOT gate is shown in fig (b). Symbol & truth table of EX-NOR gate is shown in fig (a) & (c) resp.

It shows that the o/p's of an EX-NOR gate is high (1) if both the i/p's are identical.

a) Symbol



b) Equivalent ckt



c) Truth table

A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

The boolean exp^{n} is:

$$Y = \overline{A \oplus B} \text{ or } \overline{A \odot B}$$

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

$$\boxed{Y = \bar{A}\bar{B} + AB}$$

★ Demorgan's theorems:

The two theorems suggested by Demorgan are as follows:

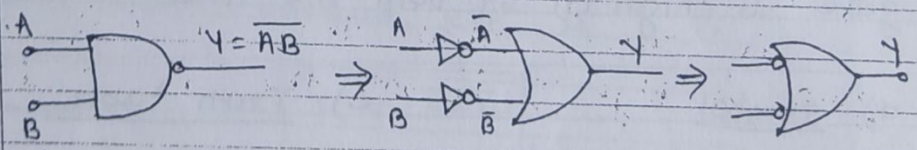
Theorem 1:

Statement: "It states that the complement of product is equal to sum of the complements"

$$\therefore \boxed{\overline{AB} = \bar{A} + \bar{B}}$$

NAND = Bubbled OR

Proof:



A	B	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

The OR gate is called as the "bubbled OR" Thus we can state Demorgan's thm as

NAND = Bubbed OR

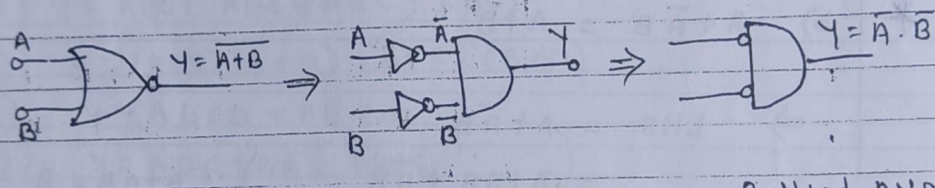
Theorem 2

statement :- " It states that the Complement of the sum is equal to the product of the complements."

$$\therefore \overline{A+B} = \overline{A} \cdot \overline{B}$$

NOR = Bubbled AND

Proof:



NOR

Bubbled AND

Truth table:

A	B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

The AND gate is called as the bubbled AND. Thus we can state De-morgan's second thm as,

NOR = Bubbled AND

* Examples on reducing the boolean exp^{n}:

1) Prove the following boolean exp^{n}

a) $A + AB = A$

$\Rightarrow$ LHS $= A + AB$

$= A(1 + B)$ $\quad \{ \because 1 + B = 1$

$= A \cdot 1$ $\quad A \cdot 1 = 1 \}$

$= A$

$\therefore \boxed{A + AB = A}$

* b) $A + \bar{A}B = A + B$

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

$= A + AB + \bar{A}B$ $\quad \{ \because A + AB = A$

$= A + B(A + \bar{A})$ $\quad B + \bar{B} = 1 \}$

$= A + B$

$\therefore \boxed{A + \bar{A}B = A + B}$

c) $(A + \bar{B} + AB)(A + \bar{B})(\bar{A}B) = 0$

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

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

$= (A + \bar{B})(\bar{A}B)$ $\quad \because A \cdot A = A$

$= A \cdot \bar{A}B + B \cdot \bar{B}\bar{A}$ $\quad \because (A + \bar{B})(A + \bar{B}) = (A + \bar{B})$

$= 0 + 0$ $\quad \because A \cdot \bar{A} = 0$

$= 0$ $\quad B \cdot \bar{B} = 0 \}$

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

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

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

$= A + B$

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

Sol ⇒ $Y = (B+BC)(B+\bar{B}C)(B+D)$

$= B(B+C)(B+D)$ — $\because B+BC = B$, $B+\bar{B}C = B+C$

$= (BB+BC)(B+D)$ — $BB = B$

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

$= B(B+D)$

$= BB+BD$

$= B+BD$ — $B+BD = B$

$= B$

HW (8)

Simplify the following exp^{n}s.

1) $Y = ABC(AB\bar{C}+A\bar{B}C+\bar{A}BC)$
2) $Y = (A+C)(A+D)(B+C)(B+D)$
3) $Y = ABCD + A\bar{B}CD$
4) $Y = \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$
5) Show that $A\bar{B}C + B + B\bar{D} + \bar{A}C = B+C$
6) Prove $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B} = \bar{A}C + A\bar{B}$
7) $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} = \bar{A}\bar{B}$
8) $Y = A(\bar{A}+C)(\bar{A}B+\bar{C})$

★ Construction of a logic ckt from a boolean exp^{n}:

Examples

1) Draw the logic ckt using the basic gates to obtain the following o/p.

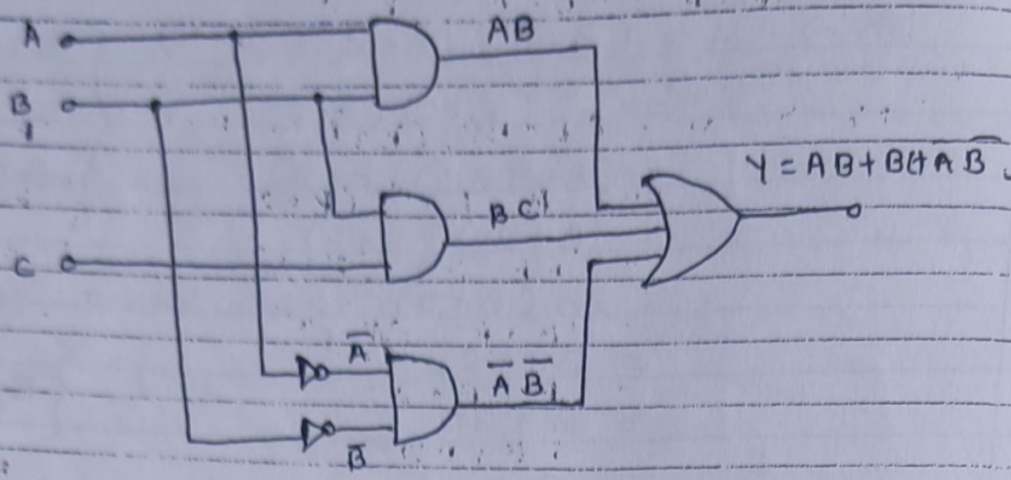
$Y = AB + BC + \bar{A}\bar{B}$

⇒ This ckt has three i/p's A, B, C & one o/p.

∴ AB → AND gate, $\bar{B}$ → NOT gate

BC → AND gate $\bar{A}\bar{B}$ → AND gate.

$\bar{A}$ → NOT gate

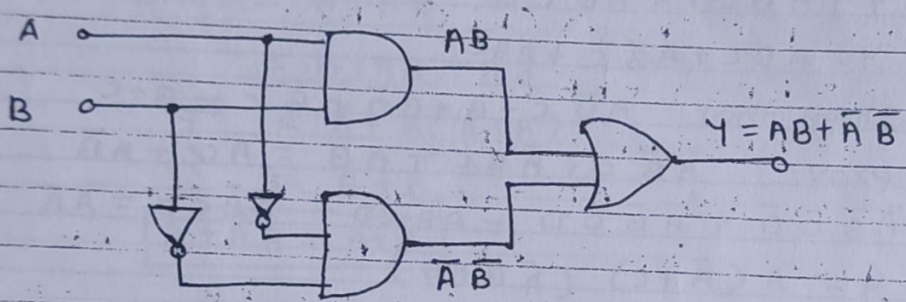


2) $Y = AB + \bar{A}\bar{B} + \bar{A}\bar{B}$

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

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

$\left\{ \because \bar{A}\bar{B} + \bar{A}\bar{B} = \bar{A}\bar{B} \right.$



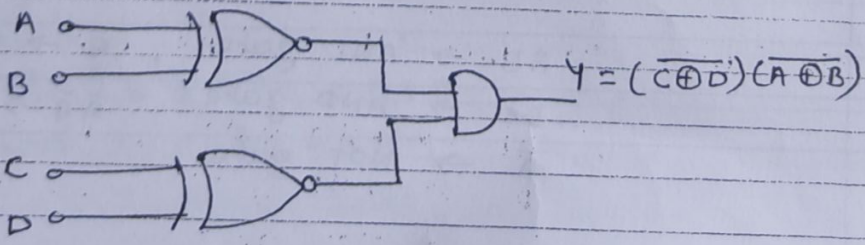
3) Reduce the following exp^{n} & implement using logic gates only.

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

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

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

$= (\overline{C \oplus D})(\overline{A \oplus B}) \quad = (C \odot D)(A \odot B)$



HW ⑤

1) Draw the logic ckt for the following exp^2.

a) $Y = \bar{A}\bar{B}C + AC$

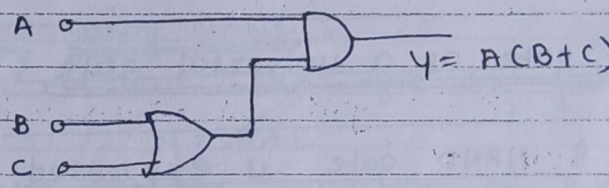
b) $Y = \bar{A}\bar{B}\bar{C} + A\bar{C}$

c) $Y = \bar{A}B + A\bar{B}C + A\bar{C}$

2) Sketch the given exp^2. Use one AND gate & one OR gate only.

$\Rightarrow \quad Y = AB + AC$

$= A(B + C)$



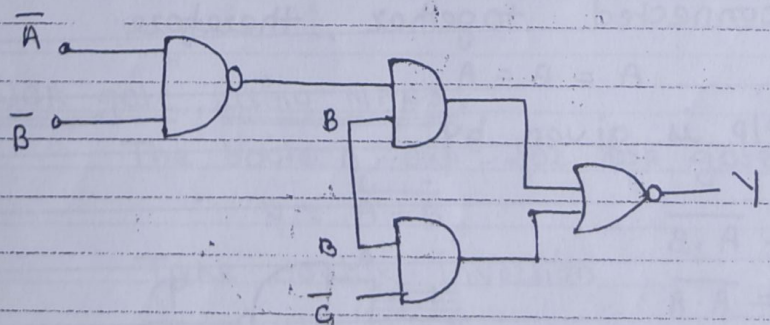
3) Draw logic diagram, use one AND gate & one OR gate.

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

4) Draw the logic dig & also write its truth table for the given exp^2.

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

5) For the given sketch, derive the boolean exp^2 for Y



* Universal gates :

The NAND & NOR gates are called as "Universal gates" because it is possible to implement any boolean exp^{z} with the help of only NAND or only NOR gates

Hence a user can build any combinational ckt with the help of only NAND gates or only NOR gates. This is a great advantage because a user will have to make a stock of only NAND or NOR ICs.

* NAND gate as a universal gate :

A NAND gate is expressed mathematically as

$$Y = \overline{A \cdot B}$$

Hence we have to bring the given boolean expression into this form.

1) NOT gate using NAND gate :

The boolean exp^{z} for an NOT gate is

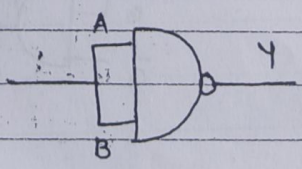
$$Y = \overline{A}$$

As both the i/p's of NAND gate are connected together, therefore

$$A = B = A$$

$\therefore$ o/p is given by

$$Y = \overline{A \cdot B}$$
$$= \overline{A \cdot A}$$
$$= \overline{A}$$



2) AND gate using NAND:

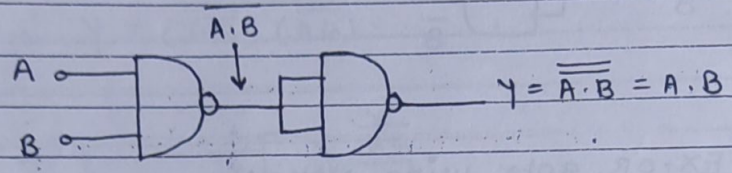
The boolean exp^2 for an AND gate is

$$Y = A \cdot B$$

Tak double inversion of R.H.S.

$$Y = \overline{\overline{A \cdot B}}$$

$$\therefore Y = A \cdot B = \overline{\overline{A \cdot B}}$$



3) OR gate using NAND:

The boolean exp^2 for OR gate is

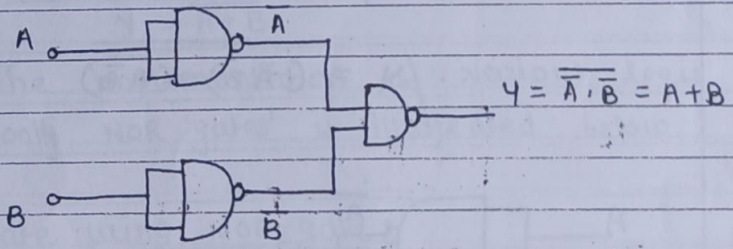
$$Y = A + B$$

Take double inversion of R.H.S.

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

Apply De-morgan's thm. ($\overline{A+B} = \overline{A} \cdot \overline{B}$) 2nd

$$Y = \overline{\overline{A} \cdot \overline{B}}$$



4) NOR gate using NAND:

The boolean exp^2 for NOR gate is,

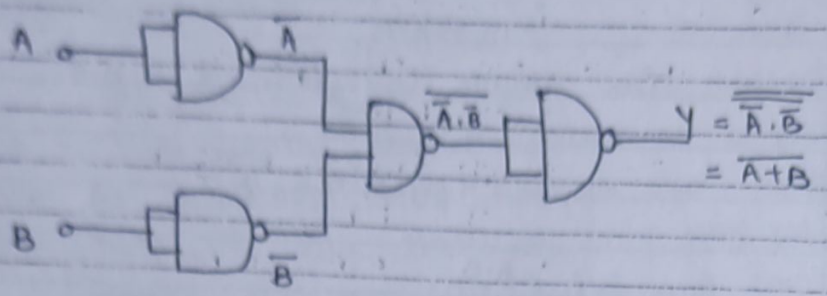
$$Y = \overline{A + B}$$

Take double inversion

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

Apply De-morgan's thm,

$$Y = \overline{\overline{A} \cdot \overline{B}}$$



5) EX-OR gate using NAND:

The boolean exp^{n} for EX-OR gate is

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

Take double inversion of RHS

$$\therefore Y = \overline{\overline{\bar{A}B + A\bar{B}}}$$

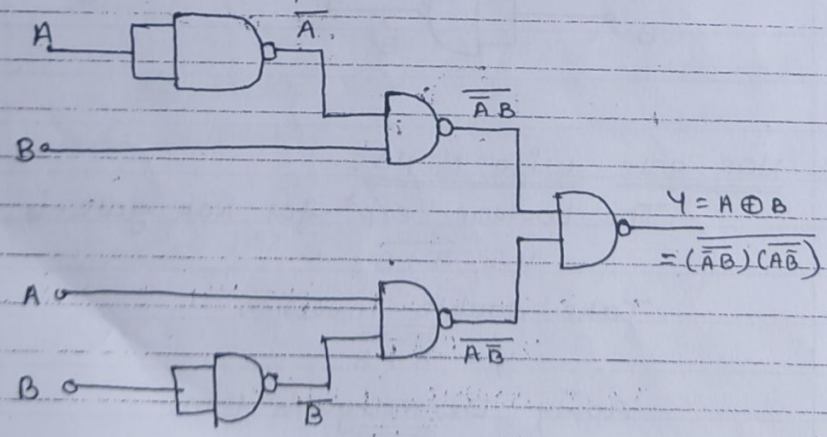
Let $X = \bar{A}B$ & $Z = A\bar{B}$

$$\therefore Y = \overline{\overline{X+Z}}$$

Apply De-morgan's thm.

$$Y = \overline{\bar{X}.\bar{Z}}$$

$$\therefore Y = \overline{(\overline{\bar{A}B}).(\overline{A\bar{B}})}$$



6) EX-NOR using NAND:

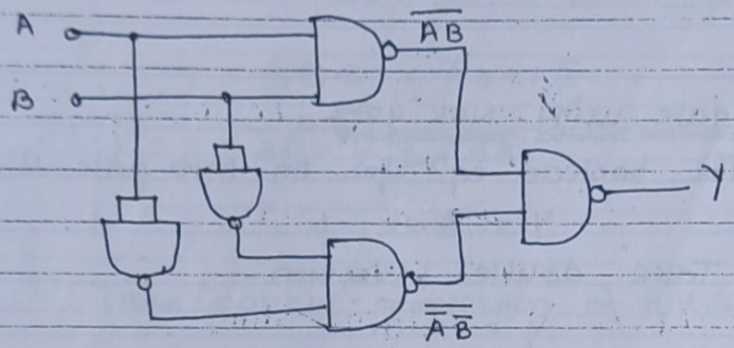
The boolean exp² for EX-NOR is

$$Y = A \odot B = \bar{A}\bar{B} + AB = X + Z$$

Take double inversion of RHS

$$Y = \overline{\overline{X+Z}} = \overline{\overline{X}\cdot\overline{Z}}$$

$$\therefore Y = \overline{(\overline{\bar{A}\bar{B}})\,(\overline{AB})}$$



★ NOR gate as a universal gate:

The Boolean exp² of the given logic ckt must be first converted into the NOR format which is

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

The implementation of various logic functions using only NOR gates is illustrated below.

1) NOT gate using NOR gate:

As both the i/p's of the NOR gate connected together, we can write that

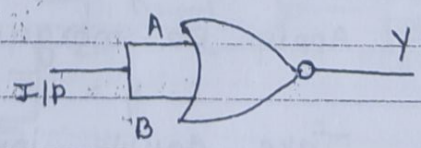
$$A = B = A$$

∴ o/p of NOR gate is given by

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

But $A+A = A$

$$\therefore \boxed{Y = \bar{A}}$$



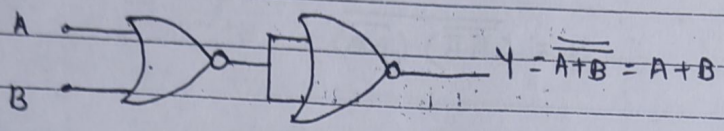
2) OR gate using NOR gate :

The Boolean exp^2 for OR gate is

$$Y = A + B$$

Take double inversion

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



3) AND gate using NOR gate :

The boolean eq^n for an AND gate is

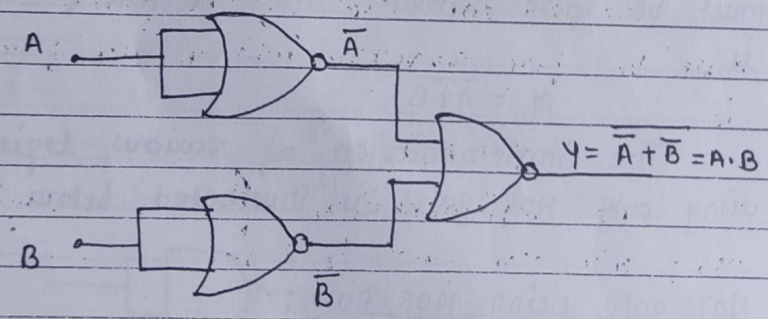
$$Y = A \cdot B$$

Take double inversion

$$Y = \overline{\overline{A \cdot B}}$$

Apply De-morgan's thm 1st

$$\boxed{Y = \overline{\overline{A} + \overline{B}}}$$



4) NAND gate using NOR gate :

The boolean exp^2 for NAND gate is,

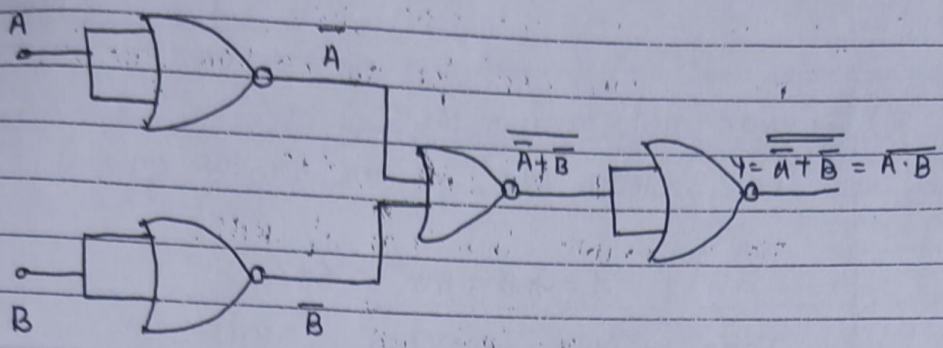
$$Y = \overline{A \cdot B}$$

Apply De-morgan's thm,

$$\therefore Y = \overline{A} + \overline{B}$$

Take double inversion

$$\boxed{Y = \overline{\overline{\overline{A} + \overline{B}}}}$$



5) **EX-OR gate using NOR gate:**

The boolean exp^2 for a EX-OR gate is

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

Let $X = \bar{A}B$ & $Z = A\bar{B}$

$$\therefore Y = X + Z$$

Take double inversion of R.H.S.

$$Y = \overline{\overline{X+Z}}$$

$$= \overline{\bar{X} \cdot \bar{Z}} \quad \leftarrow \text{Demorgan's thm.}$$

$$= \overline{(\bar{A}B)(A\bar{B})}$$

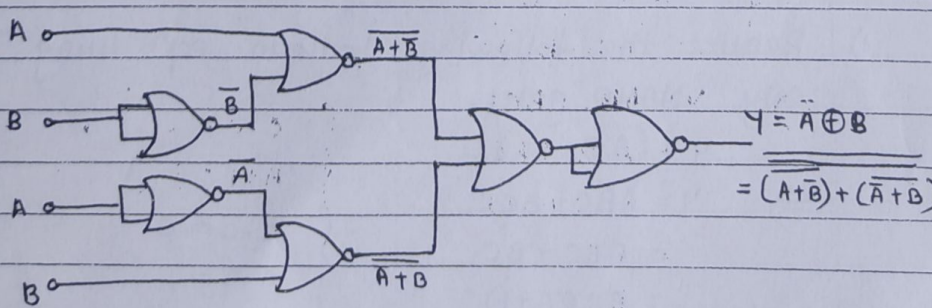
But $\overline{\bar{A}B} = (\bar{\bar{A}}+\bar{B}) = (A+\bar{B})$
& $\overline{A\bar{B}} = (\bar{A}+\bar{\bar{B}}) = (\bar{A}+B)$ } Demorgan's thm

$$\therefore Y = \overline{(A+\bar{B})(\bar{A}+B)}$$

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

Take double inversion

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



6) Ex-NOR gate using NOR gate:

The boolean exp^2 for an Ex-NOR gate is,

$$Y = \bar{A}B + A\bar{B} = X + Z$$

Take double inversion of RHS.

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

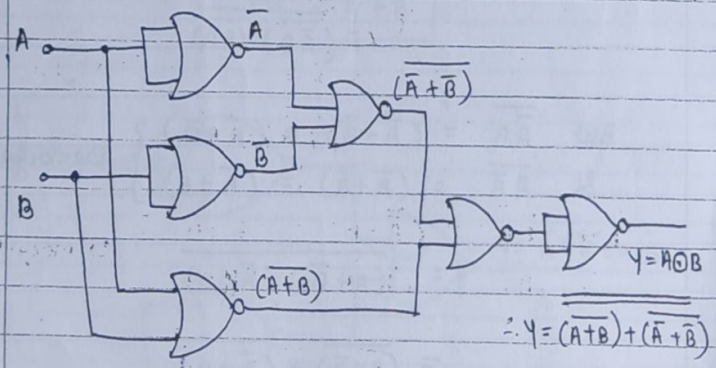
$$= \overline{(\overline{\bar{A}B}) \cdot (\overline{A\bar{B}})}$$

$$\therefore Y = \overline{(A+B)(\bar{A}+\bar{B})}$$

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

Take double inversion of RHS.

$$\therefore Y = \overline{\overline{(A+B)} + \overline{(\bar{A}+\bar{B})}}$$



★ Examples:

1) Realize the following boolean eqn using only NAND gates.

$$Y = (AB + BC)C$$

$$\Rightarrow Y = ABC + BCC$$

$$= ABC + BC \qquad \left\{ \begin{array}{l} C \cdot C = C \\ A + 1 = 1 \end{array} \right.$$

$$= BC(A+1)$$

$$= BC$$

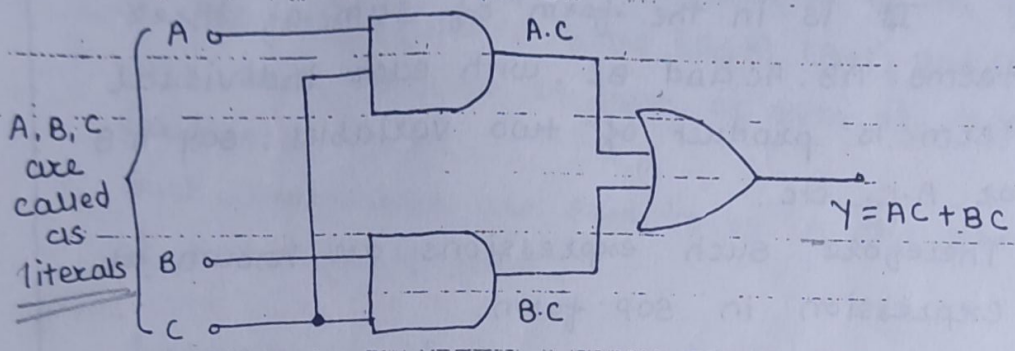
$$\therefore Y = \overline{\overline{BC}}$$

* SOP and POS Representations for Logic Expt:

Assume that the logic expt given to us is

$$Y = AC + BC$$

Then it can be realized using basic gates as shown below



where Y ⇒ Result or output

A, B, C ⇒ Literals

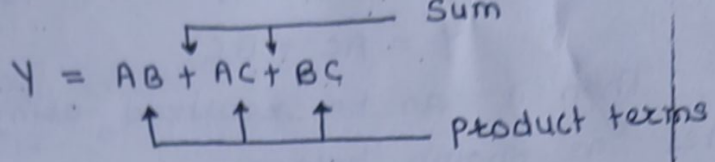
(Each literal act as an input)

Any logic expression can be expressed in the following two standard forms.

1) Sum-of-products form (SOP)
2) Product-of-sums form (POS)

These two forms are suitable for reducing the given logic expression to its simplest form.

1) Sum-of-products (SOP) form:



It is in the form of sum of three terms AB, AC and BC with each individual term is product of two variables. say A·B or A·C etc.

- Therefore such expressions are known as expression in SOP form.
- The sums and products in the SOP form are not the actual additions or multiplications. Infact they are the OR and AND functions.
- A, B and C are the literals or the i/p's of the combinational ckts.

- Examples:

$$Y = ABC + BCD + AB\bar{D}$$

$$A = XY + \bar{X}Y + X\bar{Y}$$

$$Y = \bar{P}\bar{Q} + PQR + PQ\bar{R}$$

Thus in each product term there can be one or more than one literals ANDed together. The literals can be in their complemented or uncomplemented form.

2) Product of sums (POS) form:

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

Products

Sum terms

The above logic expression is in the form of product of three terms (A+B), (B+C) and (A+C), with each term is in form of sum of two variables.

Such expressions are said to be in the product of sums (SOP) form.

Examples:

$$Y = (A + \bar{B} + \bar{C}) \cdot (A+B) \cdot (\bar{A} + C)$$
$$A = (\bar{X} + Y) \cdot (X + Y + Z)$$
$$Y = (P+R) \cdot (P+\bar{Q}) \cdot (\bar{P}+R)$$

The literals are ORed together to form the sum terms and the sum terms are ANDed to get the expression in the POS form.

* Standard or Canonical SOP and POS forms:

A logic expression is said to be in the standard (or canonical) SOP or POS form if each product term (for SOP), and sum term

(for pos) Consists of all the literals in their complemented or uncomplemented form.

Standard sop form:

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

Each product term consists of all the literals in the complemented or uncomplemented form.

Standard pos form:

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

Each sum term consists of all the literals in the complemented or uncomplemented form.

The examples of standard & non standard sop and pos exp^{s} are given below.

1) $Y = AB + AB\bar{C} + \bar{A}BC$ → Non-standard sop
2) $Y = AB + A\bar{B} + \bar{A}\bar{B}$ → Standard sop
3) $Y = (\bar{A}+B) \cdot (A+B) \cdot (A+\bar{B})$ → Standard pos
4) $Y = (\bar{A}+B) \cdot (A+B+C)$ → Non-standard pos

* Conversion from SOP to Standard SOP form:

The procedure to be followed for converting a general SOP expression into a standard SOP expression is as follows:

1) For each term find the missing literal.
2) The AND term with the term formed by oring the missing literal and its complement.
3) Simplify the expression to get the standard SOP.

Example:

① Convert the expression $Y = AB + A\bar{C} + BC$ into the standard SOP form.

Soln ⇒ Given exp^{n}: $Y = AB + A\bar{C} + BC$

Step 1 → Find the missing literal for each term.

$$Y = \underbrace{AB} + \underbrace{A\bar{C}} + \underbrace{BC}$$

- AB → Missing literal C
- $A\bar{C}$ → Missing literal B
- BC → Missing literal A

Step 2:

AND each term with (Missing literal + Its Complement).

$$Y = \underbrace{AB}\cdot\underbrace{(C+\bar{C})} + A\bar{C}\,\underbrace{(B+\bar{B})} + BC\,\underbrace{(A+\bar{A})}$$

- AB → Original product term
- $(C+\bar{C})$, $(B+\bar{B})$, $(A+\bar{A})$ → Term formed by oring of missing literal & its complement

Step 3:

Simplify the exp^n to get the standard SOP:

$$Y = AB(C+\bar{Z}) + A\bar{Z}(B+\bar{B}) + \bar{B}C(A+\bar{A})$$
$$= AB\bar{C} + AB\bar{Z} + A\bar{Z}B + A\bar{Z}\bar{B} + \bar{B}CA + \bar{A}\bar{B}C$$
$$= (ABC + AB\bar{C}) + (AB\bar{Z} + A\bar{B}\bar{Z}) + A\bar{B}\bar{Z} + \bar{A}\bar{B}C$$

But, $A + A = A$ $\therefore$ $ABC + ABC = ABC$

& $AB\bar{Z} + AB\bar{Z} = AB\bar{Z}$

$\therefore$ $\boxed{Y = ABC + AB\bar{Z} + A\bar{B}\bar{Z} + \bar{A}\bar{B}C}$ ← Standard SOP form

(Each term consist all the literals)

★ Conversion from POS to standard POS form:

Steps to be followed:

1) For each term find the missing literal
2) Then OR each term with the term formed by ANDing the missing literal in that term with its' complement.
3) Simplify the expression to get the standard POS.

Example:

① Convert the expression $Y = (A+B)(A+C)(B+\bar{Z})$ into standard POS form.

Soln ⇒ **Step 1**:

Find the missing literal for each term

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

- $(B+\bar{C})$ → Missing literal A
- $(A+B)$ → Missing literal C
- $(A+C)$ → Missing literal B

Step 2:

OR each term with (Missing literal · Its Complement)

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

Missing literal AND with its complement

This term is ~~formed~~ ORed with the term formed by ANDing the missing literal with its complement.

Step 3:

Simplify the expression to get standard pos:

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

BW- $A+BC = (A+B)(A+C)$

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

BW- $A \cdot A = A$

$$\therefore (A+B+C)(A+B+C) = (A+B+C)$$

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

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

↓ standard pos form.

(Each term consist all the literals)

Examples:

Convert the following expressions into their standard SOP or POS forms:

① $Y = AB + \bar{A}C + BC$

② $Y = (A+B)(\bar{B}+C)$

③ $Y = A + B\bar{C} + ABC$

① Given: $Y = AB + \bar{A}C + BC$

Soln:

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

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

$\because A+A=A$

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

$$\therefore Y = \boxed{ABC + AB\bar{C} + \bar{A}BC + \bar{A}C\bar{B} + BC\bar{A}}$$ ← standard SOP form

② Given: $Y = (A+B)(\bar{B}+C)$

Soln

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

But $A+BC = (A+B)(A+C)$

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

↓

Standard POS form

③ Given: $Y = A + BC + ABC$

Soln:

$$Y = A(B+\bar{B})(C+\bar{C}) + BC(A+\bar{A}) + ABC$$
$$= (AB + A\bar{B})(C+\bar{C}) + BCA + BC\bar{A} + ABC$$
$$= ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC + ABC$$
$$= (ABC + ABC + ABC) + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$$
$$\boxed{Y = ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC}$$

Standard sop form.

★ Concepts of Minterm and Maxterm:

Minterm:

Each indivisual term in the standard sop form is called as minterm.

Maxterm:

Each indivisual term in the standard pos form is called as maxterm.

Standard sop $Y = \underbrace{ABC} + \underbrace{A\bar{B}\bar{C}} + \underbrace{\bar{A}B\bar{C}}$

Each indivisual term is called minterm.

Standard pos $Y = \underbrace{(A+B)}\ \underbrace{(A+\bar{B})}$

Each indivisual term is called Maxterm

- Following table gives the minterms & maxterms for a three variable / literal logic function.
- Let Y be the o/p & A, B, C be the variables
- In general for 'n' no. of variables the no. of minterms or maxterms will be 2^n
- For three variables the no. of minterms & maxterms are

$$2^3 = 8$$

- Each minterm is represented by 'm$_i$' where $i = 0, 1, \cdots, 2^n - 1$
- & each maxterm is represented by 'M$_i$'

Variables			Minterms	Maxterms
A	B	C	m_i	M_i
0	0	0	$\bar{A}\bar{B}\bar{C} = m_0$	$A+B+C = M_0$
0	0	1	$\bar{A}\bar{B}C = m_1$	$A+B+\bar{C} = M_1$
0	1	0	$\bar{A}B\bar{C} = m_2$	$A+\bar{B}+C = M_2$
0	1	1	$\bar{A}BC = m_3$	$A+\bar{B}+\bar{C} = M_3$
1	0	0	$A\bar{B}\bar{C} = m_4$	$\bar{A}+B+C = M_4$
1	0	1	$A\bar{B}C = m_5$	$\bar{A}+B+\bar{C} = M_5$
1	1	0	$AB\bar{C} = m_6$	$\bar{A}+\bar{B}+C = M_6$
1	1	1	$ABC = m_7$	$\bar{A}+\bar{B}+\bar{C} = M_7$

★ How are the Minterm & Maxterms obtd:

⇒ Minterm for a particular combination of ABC:

★ Let ABC = 011

★ Assume that A', B, C are i/p to an AND gate. We want the o/p of the AND gate to be 1. For that all its i/p's should be 1. So complement the i/p which is 0 i.e. A in this case.

∴ Minterm = $\bar{A}BC$ corresponding to ABC = 011

⇒ Maxterm for a particular combination of ABC:

- Let ABC = 011.
- Assume that A, B, C are the i/p's to an OR gate
- We want the o/p of the OR gate to be 0. So all the i/p's to the OR gate should be 0s
- So invert the i/p's which are 1s i.e. B & C in this case

∴ Maxterm = $(A + \bar{B} + \bar{C})$ corresponding to ABC = 011

* Truth table of two variable (Minterms & Maxterms):

Variables / literals		Minterms	Maxterms
A	B	m_i	M_i
0	0	$\bar{A}\bar{B} = m_0$	$A + B = M_0$
0	1	$\bar{A}B = m_1$	$A + \bar{B} = M_1$
1	0	$A\bar{B} = m_2$	$\bar{A} + B = M_2$
1	1	$AB = m_3$	$\bar{A} + \bar{B} = M_3$

* Representation of logical exp^z using minterms & maxterms:

We can represent the logical exp^z using minterms & maxterms as follows:

1) $Y = \underbrace{ABC} + \underbrace{\bar{A}BC} + \underbrace{A\bar{B}\bar{C}}$ ← Given logic exp^z

$m_7 \quad m_3 \quad m_4$ ← corresponding minterms

$\therefore Y = m_7 + m_3 + m_4$

$\therefore Y = \Sigma m\,(3, 4, 7)$ ← other way of representation

where Σ denotes sum of products.

2) $Y = \underbrace{(A + \bar{B} + C)}\ \underbrace{(A + B + C)}\ \underbrace{(\bar{A} + \bar{B} + C)}$ ← Given exp^z

$M_2 \quad M_0 \quad M_6$ ← corresponding Maxterms

$\therefore Y = M_2 M_0 M_6$

$\therefore Y = \Pi M(0, 2, 6)$ ← other way of representation

where Π denotes product of sums.

* Write standard SOP Expⁿ for a given truth table:

Procedure:

1) Consider only those combinations of inputs which correspond to $Y = 1$
2) Write down a product term for each such combination.
3) OR all these product terms to get the std. SOP

* Examples:

1) From the truth table obtain the logical expⁿ in the standard SOP form.

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

Solⁿ: Step 1:

Consider only those combinations of inputs which

correspond to Y = 1

	A	B	Y	
1	0	0	0	← Y = 0; so do not consider this entry
2	0	1	1	← Y = 1, so these entries should be considered
3	1	0	1	←
4	1	1	0	← Y = 0, so don't consider

Step 2 :

For the second & third entries, write the product term:

	A	B	Y	
1	0	0	0	
2	0	1	1	→ $Y_1 = \bar{A}B$
3	1	0	1	→ $Y_2 = A\bar{B}$ } Boolean exp^s in the product form
4	1	1	0	

Step 3 :

OR (Add) all the product terms to write the final exp^s in the std. Sop form.

$\therefore Y = Y_1 + Y_2$

$$\boxed{Y = \bar{A}B + B\bar{A}}$$ ← Std. Sop form

It can also be represented as follows:

$Y = \bar{A}B + A\bar{B} = m_1 + m_2$

01 10

m_2

where m_1 → minterm corresponding to $\bar{A}B = 01$

m_2 → minterm corresponding to $A\bar{B} = 10$

The exp^n can also expressed as.

$$Y = \Sigma m(1,2)$$

2) For the truth table given below, write the logic exp^n in the std. sop form.

Soln:

A	B	C	Y	
0	0	0	0	
0	0	1	1	← $\bar{A}\bar{B}C$ (m_1)
0	1	0	0	
0	1	1	0	
1	0	0	1	← $A\bar{B}\bar{C}$ (m_4)
1	0	1	0	
1	1	0	0	
1	1	1	1	← ABC (m_7)

A B C
0 0 0 0
0 0 1 1
1 0 0
1 1 1 7

OR all the product terms

$$\therefore \boxed{Y = \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC}$$ ← Std sop form

This exp^n can also be written as:

$$Y = m_1 + m_4 + m_7$$
$$= \Sigma m(1,4,7)$$

★ Write std. POS exp^{n} for a given truth table:

Procedure:

1) Consider only those combinations of i/p's which produce a low o/p ($Y = 0$)
2) Write the maxterms only for such combinations
3) AND these maxterms to obtain the exp^{n} in std. pos form.

★ Example:

① Write the logic exp^{n} in std pos form for the truth table given below.

Soln: step 1:

	A	B	C	Y	
0	0	0	0	0	← $A+B+C$ (M_0)
1	0	0	1	1	←
2	0	1	0	1	←
3	0	1	1	0	← $A+\bar{B}+\bar{C}$ (M_3)
4	1	0	0	1	
5	1	0	1	0	← $\bar{A}+B+\bar{C}$ (M_5)
6	1	1	0	0	← $\bar{A}+\bar{B}+C$ (M_6)
7	1	1	1	1	

Write maxterms for the combinations of i/p which produce $Y=0$

Step 2:

AND (take product of) all the maxterms to get std. pos form:

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

↑ std pos form

This exp^{n} can also be written as

$$Y = M_0 M_3 M_5 M_6$$

OR $$Y = \pi M(0,3,5,6)$$

★ Simplification Technique for Boolean Exp^{n}:

Simplification techniques of boolean exp^{n}s are given below:

1) Algebraic simplification
2) Simplification using Karnaugh maps (K-maps)

★ Algebraic simplification:

Standard procedure for algebraic simplification:

1) Bring the given exp^{n} into the SOP form by using the boolean laws & Demorgan's thms.

2) Simplify this SOP exp^{n} by checking the product terms for common factors.

Examples:

1) Simplify the exp^{n} given below:

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

Soln:

Bring the given exp^{n} in sop form

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

$$Y = AB + A\bar{A} + AB + \bar{A}B + BB$$

But, $AB + AB = AB$

$A\bar{A} = 0$

$BB = B$

$\therefore Y = AB + \bar{A}B + B$

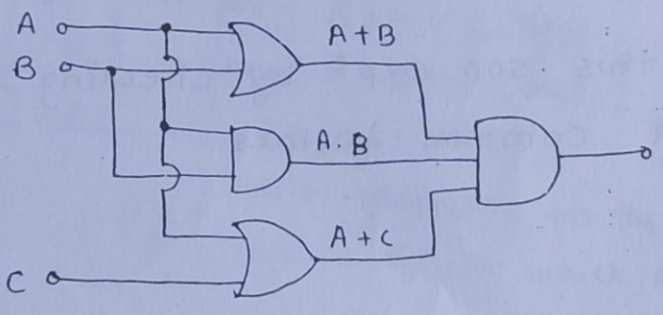
$= \bar{A}B + B(1+A)$

$= \bar{A}B + B$ $\quad \{ \because 1+A = 1 \; ; \; \bar{A}+1 = 1 \}$

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

$= B$

2) For the given logic ckt write the boolean exp^{n} & simplify it.



Soln:

The o/p exp^{n} for the given logic ckt is

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

Bring this exp^{n} in SOP form

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

But $AA = A$

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

$$= AAB + AA\bar{B}C + BAB + BA\bar{B}C$$

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

But $AB + AB = AB$ & $A\bar{B}C + ABC = AC$

$$= AB + AC$$

$$= AB(1+C)$$

$$\therefore Y = AB \quad \{\because 1+C = 1$$

3) Simplify the following three variable boolean exp^{n}

$$Y = \Sigma m(2, 4, 6)$$

Soln: The given exp^{n} in SOP form is as follows:

$$Y = m_2 + m_4 + m_6$$

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

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

$$= \bar{A}B\bar{C} + A\bar{C} \quad \{\because \bar{B}+B = 1$$

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

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

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

Disadvantages of Algebraic method of simplification:

① In the algebraic method of simplification we need to write lengthy eq^n^s.. find the common terms.. manipulate the exp^t^ etc. so it is a time consuming work.

② Some time we are not sure whether the further simplification is really possible or not.

★ K-map Simplification:

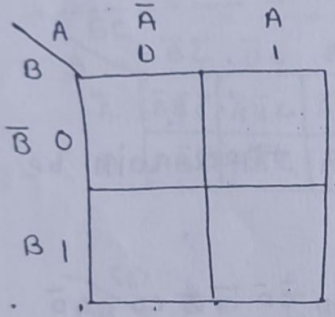
- K-map (short form of Karnaugh map) is a graphical method of simplifying a boolean eq^n^
- It contains boxes.
- The information contained in a truth table or available in the SOP or POS form is represented on a K-map.
- The K-map method is a systemetic approach to simplification of boolean exp^t^.
- K-maps can be written for 2, 3, 4.. upto 6 variables. Beyond the K-map technique becomes very cumberseme.

(K-map ideally suitable for deigning the combinationcuitan logic ckt uing either a sop method or a pos method.)

K-map structure:

- The K-map comprises a boxes for every line in the truth table.
- The 0's & 1's written on top or sides of the boxes represent the values of corresponding variables.
- In truth tables, the values of i/p's follow a standard sequence (00, 01, 10, 11). But in K-maps the i/p values are ordered in a different sequence (00, 01, 11, 10). This is as per the gray code not binary code so that as we move along a row or column only one variable will change its value.

2- Variable K-map:

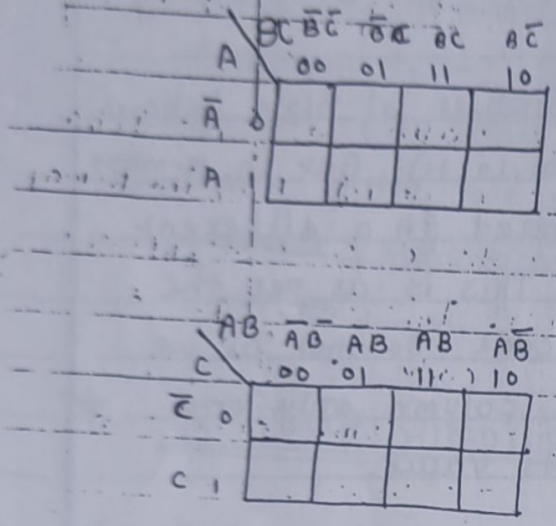


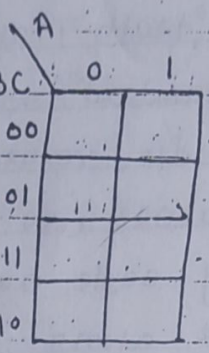
For a 2-variable K-map there will be 4 boxes

- A & B are the i/p's or variables
- 0 & 1 are the value of A or B
- Inside 4 boxes we have enter the values of Y i.e. output.

3-variable K-map:

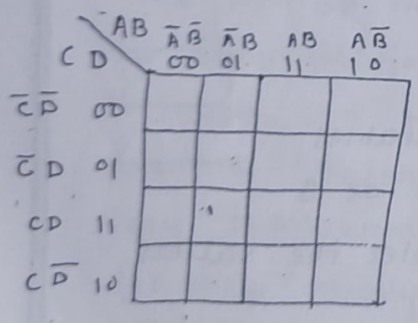
1) For a 3-variable K-map there will be 8 boxes.
2) 0 & 1 are the values of corresponding variable.



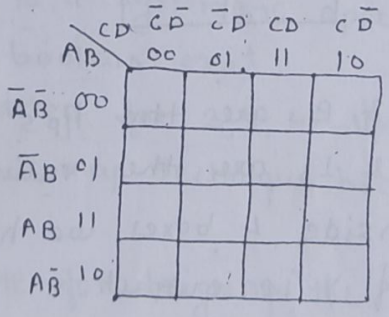


4-variable K-map:

- For a 4-variable K-map there will be 16 boxes.



OR



* K-map boxes & associated product terms:

- Each row & column of a K-map is labelled with a variable, or a group of variables or their complements.
- In fig (a), shaded box corresponds to the first row which is labelled by $\bar{A}$ & first column labelled by $\bar{B}$. Hence the product term written inside this box is $\bar{A}\bar{B}$.

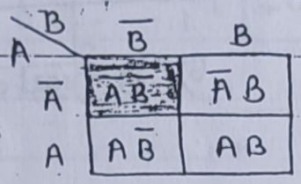


fig (a):
2-variable K-map

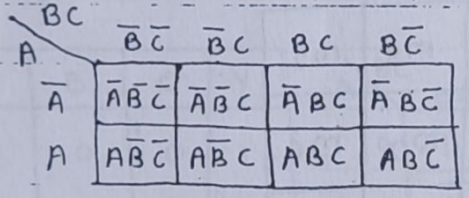


fig (b):
3-variable K-map

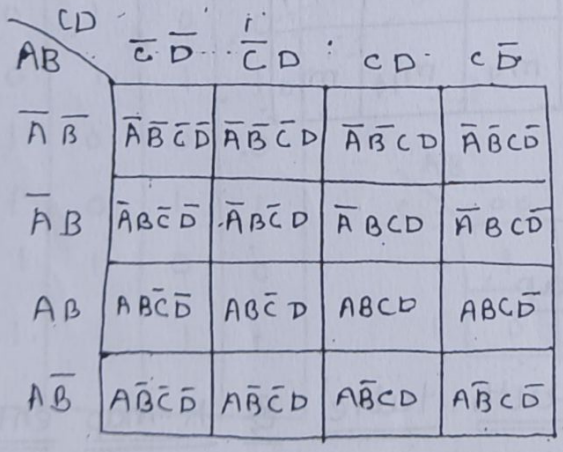


fig (c):
4-variable K-map

* Alternative way to label the K-map:

Instead of writing the actual product term, the corresponding shorthand minterm notations $m_0, m_1, m_2 \ldots$ are entered.

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

a) Two-variable K-map

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

b) 3-variable K-map

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}

* Plotting the K-map:

Relation bet^n a truth table & K-map entries:

fig (a) shows the representation of a

Truth table using a 2-variable K-map Inside the boxes of the K-map we have to enter the values of o/p Y corresponding to various combinations of A & B.

2-variable K-map:

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

A \ B	0	1
0	0	0
1	0	1

OR

B \ A	0	1
0	0	0
1	0	1

3-variable K-map:

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

A \ BC	00	01	11	10
0	1	0	1	0
1	0	1	1	0

OR

C \ AB	00	01	11	10
0	1	0	0	0
1	0	1	1	1

A / BC	0	1
00	1	0
01	0	1
11	1	1
10	0	0

4-Variable K-map:

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

AB / CD	00	01	11	10
00	1 $_0$	0 $_1$	1 $_3$	0 $_2$
01	0 $_4$	0 $_5$	1 $_7$	0 $_6$
11	0 $_{12}$	0 $_{13}$	1 $_{15}$	0 $_{14}$
10	1 $_8$	0 $_9$	1 $_{11}$	0 $_{10}$

OR

CD / AB	00	01	11	10
00	1 $_0$	0 $_4$	0 $_{12}$	1 $_8$
01	0 $_1$	0 $_5$	0 $_{13}$	0 $_9$
11	1 $_3$	1 $_7$	1 $_{15}$	1 $_{11}$
10	0 $_2$	0 $_6$	0 $_{14}$	0 $_{10}$

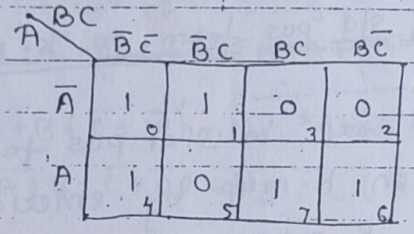
✱ Representation of std. SOP form on K-map:

1) The logical expression in standard SOP form can be represented on a K-map corresponding to each minterm present in the eqⁿ. by simply entering 1's in the cells (boxes) of the K-map corresponding to each minterm present in the eqⁿ.
2) The remaining cells (boxes) are filled with zeros.

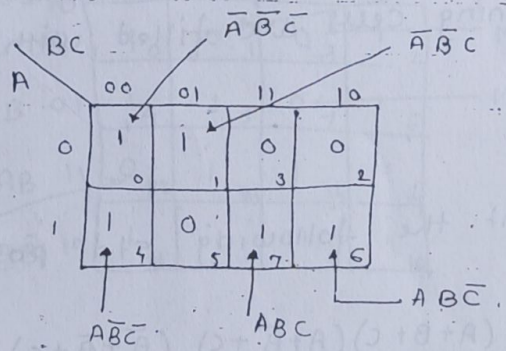
Examples:

1) Represent the eqⁿ given below on K-map.

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

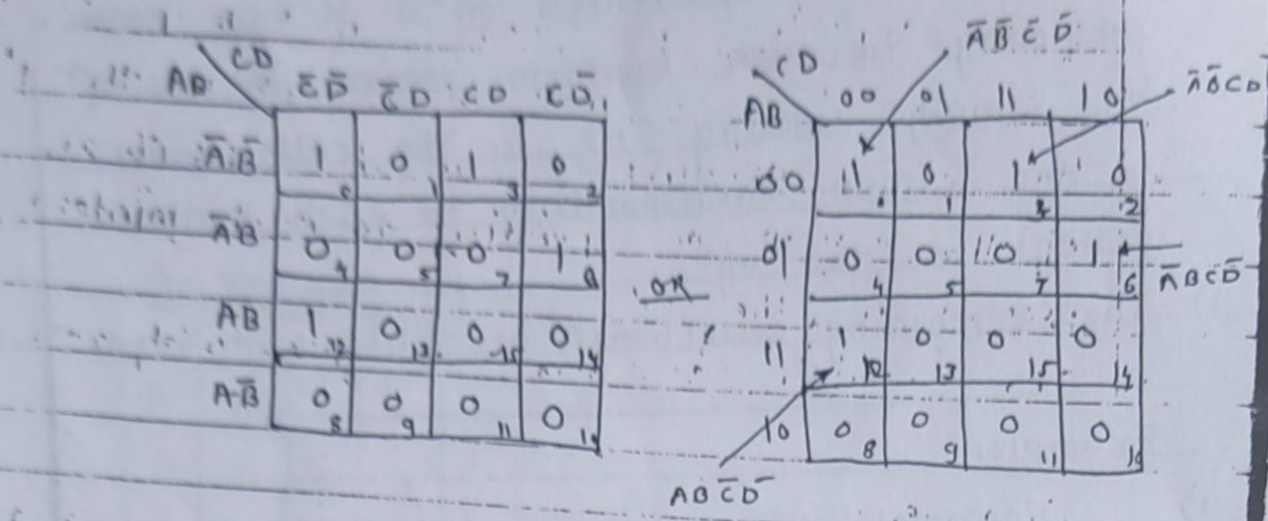


OR



2) Plot the following boolean exp² on K-map

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



★ Representation of std pos form on K-map:

1) The logical exp² in std pos form can be represented on K-map by entering 0's in the cells of K-map corresponding to each maxterm present in the given eq².
2) The remaining cells are filled with 1's.

★ Examples:

1) Represent the following std. pos exp² on K-map.

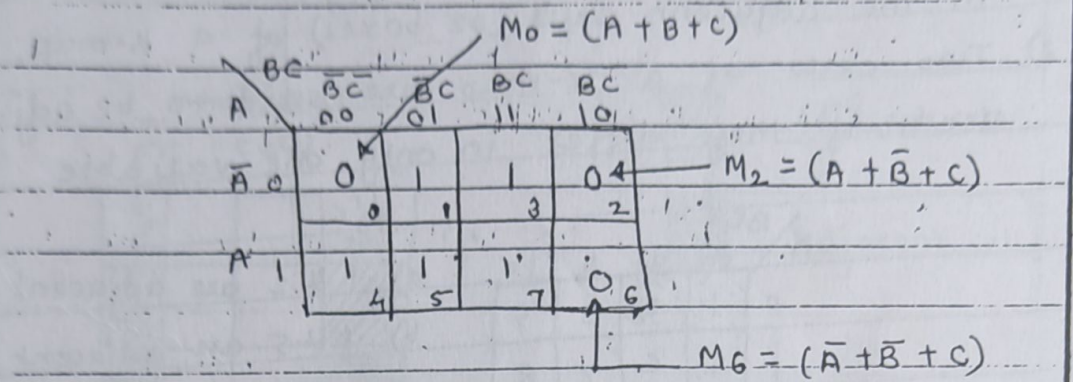
$$Y = (A+B+C)(A+\bar{B}+C)(\bar{A}+\bar{B}+C)$$

The given exp^{n} has three maxterms as follows:

$A+B+C = M_0$

$A+\bar{B}+C = M_2$

$\bar{A}+\bar{B}+C = M_6$

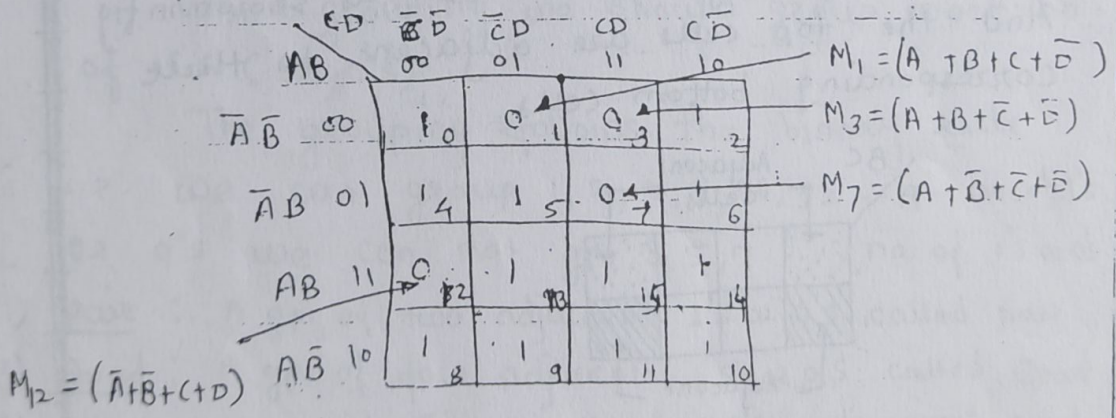


2) Represent the following std. pos eq^{n} on K-map.

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

$\Rightarrow (A+B+C+\bar{D}) = M_1 \qquad (A+\bar{B}+\bar{C}+\bar{D}) = M_7$

$(A+B+\bar{C}+\bar{D}) = M_3 \qquad (\bar{A}+\bar{B}+C+D) = M_{12}$



* Simplification of boolean exp^{t}s using K-map:

1) Simplification of boolean exp^{r}s using K-map is based on combining or grouping the terms in the adjacent cells (or boxes) of a K-map.
2) Two cells of a K-map are said to be adjacent if they differ in only one variable

A \ BC	00	01	11	10
0	1	2	3	4
1	5	6	7	8

a) 1 & 2 are adjacent
b) 1 & 5 are adjacent
c) Bw. 1 & 6 or 2 & 5 are not adjacent.

3) Note the cells one left or right side or at the bottom & top cells are adjacent cells. But the cells connected digonally are not the adjacent ones.
4) The left most cells are adjacent to their corresponding right most cell as shown in fig
5) And the top cells are adjacent to their corresponding bottom cells.

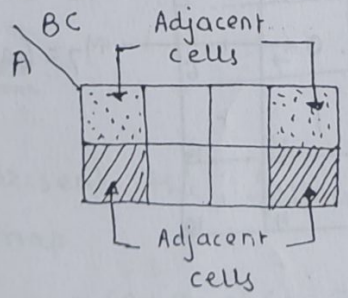
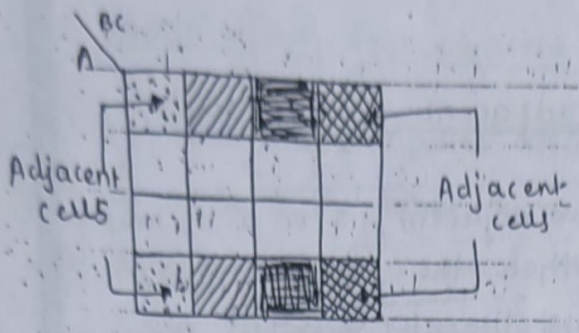
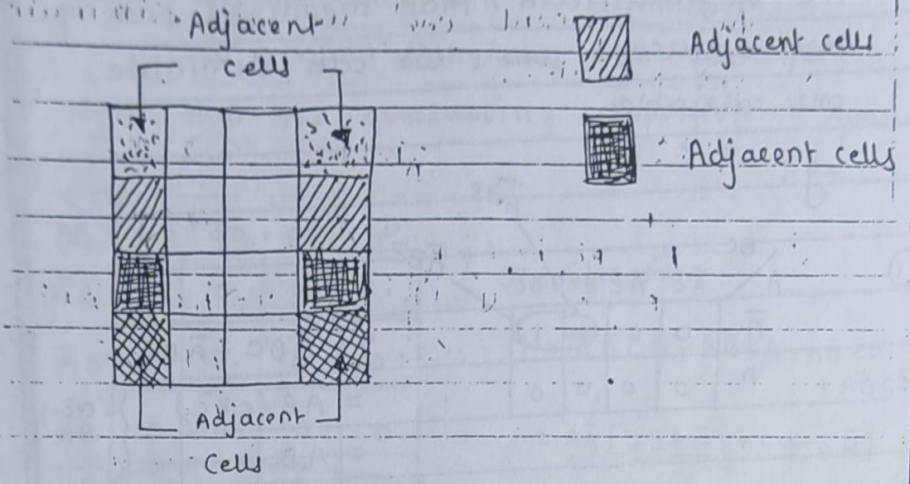


fig ④ Left most & right most cells are adjacent.





✱ Way of grouping :

While grouping we should group most no. of 1's (or 0's)

The grouping follows the binary rule

i.e. we can group 1, 2, 4, 8, 16, 32 ---- no. of 1's or 0's we can not gp 3, 5, 7 ---- no. of 1's or 0's

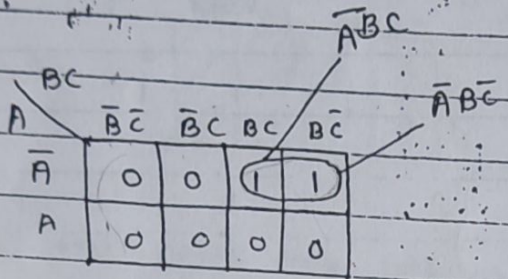
1) Pair : A gp of two adjacent 1's or 0's called pair
2) Quad : A gp of four adjacent 1's or 0's called Quad
3) Octet : A gp of Eight adjacent 1's or 0's called octet

* Grouping of two adjacent one's (pairs):

If we gp. two adjacent 1's on a K-map to form a pair, than the resulting term will have one less literal (variable) than the original term. That means by pairing two adjacent one's we can eliminate one variable.

eg:

① eg



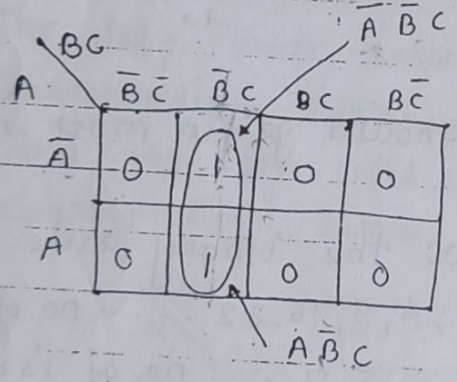
$\therefore Y = \bar{A}B\bar{C} + \bar{A}BC$

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

$= \bar{A}.B$

(C is eliminated)

②



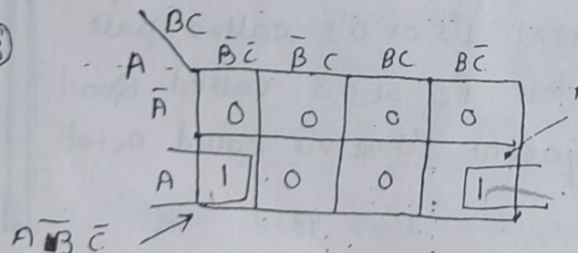
$\therefore Y = \bar{A}\bar{B}C + A\bar{B}C$

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

$= \bar{B}C$

(A is eliminated)

③



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

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

$= A\bar{C}$

* Grouping of four adjacent one's (Quad):

1) In Quad two variables associated with the minterms are same & the other two are not the same.
2) After forming a quad, the simplification takes place in such a way that the two variables which are not same will be eliminated.

Thus a quad eliminates two variables.

3) Note that the overlapping is possible in quad.

eg: ①

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	0	0
AB	0	0	0	0
$A\bar{B}$	1	1	1	1

$A\bar{B}$

Simplification:

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

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

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

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

$= A\bar{B}$

The two variables which are same in all minterms are A & $\bar{B}$ while C & $\bar{D}$ are not same.

∴ C & D will be eliminated

∴ $Y = A\bar{B}$

(2)

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	0	0
$A\bar{B}$	0	1	0	0

1) $\bar{C}$ & D are same
2) A & B are not same
∴ A & B will be eliminated
∴ $Y = \bar{C}D$

Simplification:

$$Y = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + A\bar{C}D + A\bar{B}\bar{C}D$$
$$= \bar{C}D[\bar{A}\bar{B} + \bar{A}B + AB + A\bar{B}]$$
$$= \bar{C}D[\bar{A}(\bar{B}+B) + A(B+\bar{B})]$$
$$= \bar{C}D[\bar{A} + A]$$
$$= \bar{C}D$$

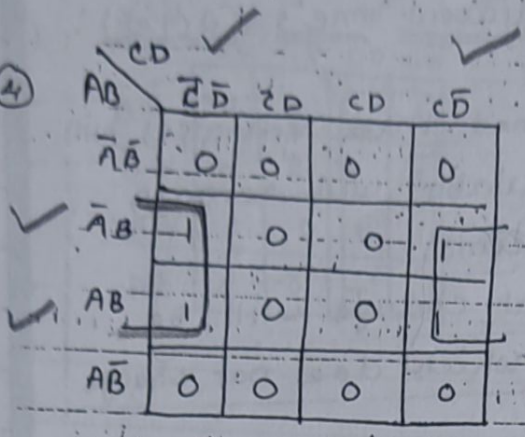
(3)

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	1	1	0	0
AB	1	1	0	0
$A\bar{B}$	0	0	0	0

$B\bar{C}$

$$Y = \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}B\bar{C}D + AB\bar{C}D$$
$$= B\bar{C}\bar{D}(\bar{A}+A) + B\bar{C}D(\bar{A}+A)$$
$$= B\bar{C}\bar{D} + B\bar{C}D$$
$$= B\bar{C}(\bar{D}+D)$$
$$= B\bar{C}$$ (A & D are eliminated.

④



$B\bar{D}$

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

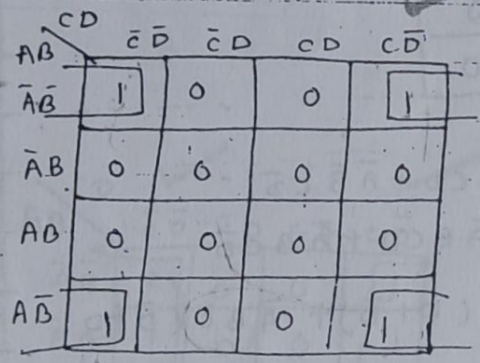
$= B\bar{C}\bar{D}(\bar{A}+A) + BC\bar{D}(\bar{A}+A)$

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

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

$= B\bar{D}$ (A & C are eliminated)

⑤



$\bar{B}\bar{D}$

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

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

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

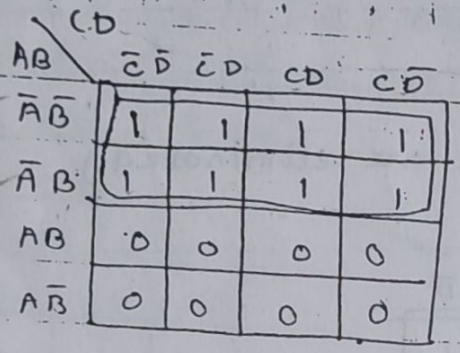
$= \bar{B}\bar{D}(\bar{C}+C)$

$= \bar{B}\bar{D}$ (A & C are eliminated)

* Grouping of Eight adjacent one's (octet):

1) When an octet is formed three variables will change & only one variable will remain same in all the minterms.
2) The three variables that change will be eliminated & variable which does not change will appear as o/p.
3) Thus octet eliminates three variables.

① eg:



$Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} +$
$+\bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D}$
$= \bar{A}\bar{B}\bar{C}(\bar{D}+D) + \bar{A}\bar{B}C(D+\bar{D}) + \bar{A}B\bar{C}(\bar{D}+D$
$+ \bar{A}BC(D+\bar{D})$
$= \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$
$= \bar{A}\bar{B}(\bar{C}+C) + \bar{A}B(\bar{C}+C)$
$= \bar{A}\bar{B} + \bar{A}B.$
$= \bar{A}(\bar{B}+B)$
$= \bar{A}$ (B, C & C are eliminated)

(2)

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	1
$\bar{A}B$	0	0	1	1
AB	0	0	1	1
$A\bar{B}$	0	0	1	1

$Y = C$

(A, B & D are eliminated)

(3)

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	0	0	0	0
AB	0	0	0	0
$A\bar{B}$	1	1	1	1

$Y = \bar{B}$

(A, C & D are eliminated)

(4)

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	1	0	0	1
AB	1	0	0	1
$A\bar{B}$	1	0	0	1

$Y = \bar{D}$

(A, B & C are eliminated)

* Minimization of SOP Expressions:

Procedure:

1) Prepare the K-map & place 1's according to the given truth table or logical exp^{n}. Fill the remaining cells by 0's.
2) Locate the isolated 1's i.e. the 1's which can not be combined with any other 1. Encircle such 1's.
3) Identify the 1's which can be combined to form a pair in only one way & encircle them.
4) Identify the 1's which can form a quad in only one way & encircle them.
5) Identify the 1's which can form a octet in only one way & encircle them.
6) After identifying the pairs, quad & octet check if any 1 is yet to be encircle. If yes then encircle them with each other 1 or with the already encircled 1's (by overlapping).

Note:

1) No. of groups should be minimum.
2) Any 1 can be included any no. of times without affected the exp^{n}.

Examples:

1) A logical exp^{n} in the std. sop form is as follows:

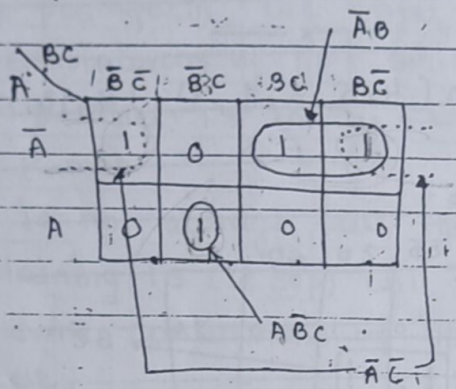
$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$

Minimize it using the K-map.

⇒

$\bar{A}\bar{B}\bar{C} = m_0$; $\bar{A}BC = m_3$

$\bar{A}B\bar{C} = m_2$; $A\bar{B}C = m_5$



$\bar{A}\bar{B}\bar{C} = 000 = m_0$

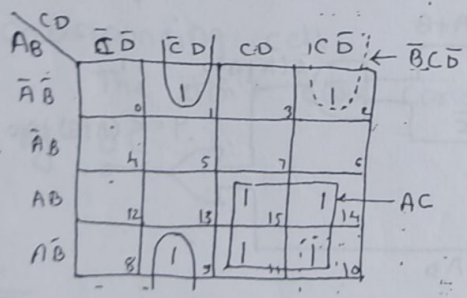
$\therefore Y = \bar{A}B + \bar{A}\bar{C} + A\bar{B}C$

---

② Minimize the following exp^{n} using K-map & realize using the basic gates.

$Y = \Sigma m(1, 2, 9, 10, 11, 14, 15)$

⇒ $Y = m_1 + m_2 + m_9 + m_{10} + m_{11} + m_{14} + m_{15}$



$Y = \bar{B}\bar{C}D + \bar{B}C\bar{D} + AC$

$= \bar{B}(\underbrace{\bar{C}D + C\bar{D}}_{\text{EX-OR}}) + AC$

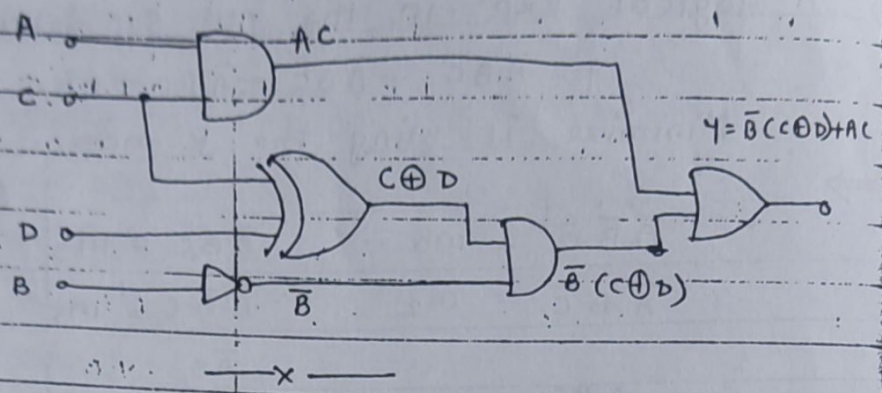
$= \bar{B}(C \oplus D) + AC$

$\bar{A}\bar{B}\bar{C} = 000 = 0 = m_0$

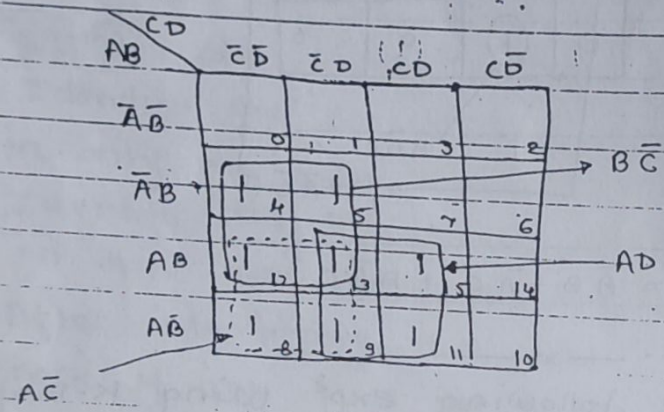
$\bar{A}B\bar{C} = 010 = 2 = m_2$

$\bar{A}BC = 011 = 3 = m_3$

$A\bar{B}C = 101 = 5 = m_5$

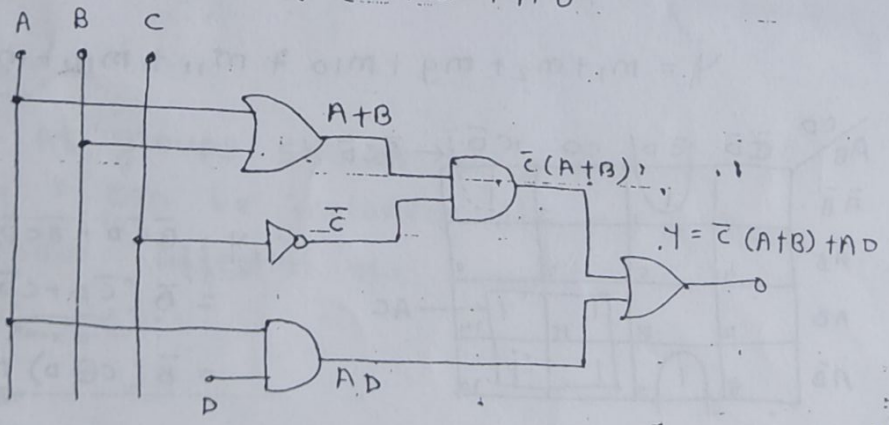


③ $Y = \sum m(4,5,8,9,11,12,13,15)$



$\therefore Y = AD + B\bar{C} + A\bar{C}$

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



$Y = AC + \bar{B}.(C \oplus D)$

AND NOT EX-OR

AND

OR

Don't care conditions

① For sop form, we enter 1's corresponding to the combinations of i/p variables which produce a high o/p. And we enter 0's in the remaining cells of the k-map.

② For the pos form, we enter 0's corresponding to the combinations of i/p's which produce a low o/p & enter 1's in the remaining cells of the k-map.

③ But it is not always true that the cells not containing 1's (in sop) will contain 0's, because some combinations of i/p variable do not occur.

④ Also for some fn's the o/p corresponding to certain combinations of i/p variables do not matter

⑤ In such situation we have a freedom to assume 0 or 1 as o/p for each of these combinations. These conditions are known as "Don't care conditions" & in the k-map it is represented as × (cross) mark in the corresponding cell.

The don't care conditions are indicated by d( ).

Examples:

① Simplify the expt given below using K-map.

$$Y = \Sigma m(1,3,7,11,15) + d(0,2,5)$$

Soln.

$$Y = m_1 + m_3 + m_7 + m_{11} + m_{15} + d(0,2,5)$$

Regular minterms so enter 1's

Don't care conditions so enter X mark

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	X	1	1	X
$\bar{A}B$			1	
AB			1	
$A\bar{B}$				

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

CD

$\therefore Y = CD + \bar{A}\bar{B}$

★ K-map representation of POS form:

A \ B	$\bar{B}$ 0	B 1
A 0	$A+\bar{B}$	$A+B$
$\bar{A}$ 1	$\bar{A}+\bar{B}$	$\bar{A}+B$

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

AB \ CD	00	01	11	10
00	$A+\bar{B}+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+D$	$A+B+\bar{C}+D$	$A+B+\bar{C}+\bar{D}$
01	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+D$	$\bar{A}+B+\bar{C}+D$	$\bar{A}+B+\bar{C}+\bar{D}$
11	$\bar{A}+\bar{B}+C+\bar{D}$	$\bar{A}+\bar{B}+C+D$	$\bar{A}+B+C+D$	$\bar{A}+B+C+\bar{D}$
10	$A+\bar{B}+C+\bar{D}$	$A+\bar{B}+C+D$	$A+B+C+D$	$A+B+C+\bar{D}$

* Simplification of pos form using K-map

Procedure:

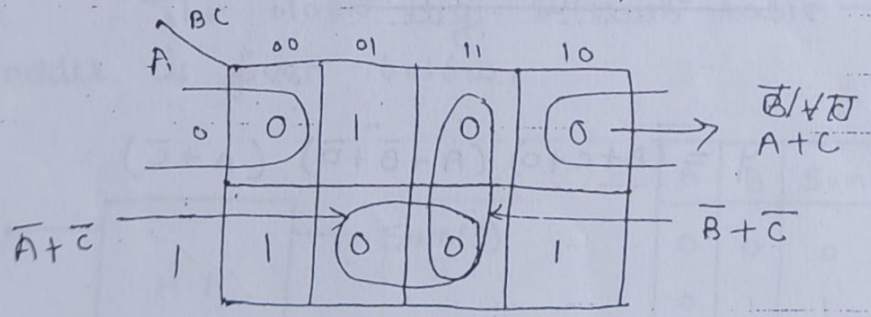
1) The given pos exp^{n} ~~in the pos form~~ consist of maxterms.
2) Corresponding to every maxterm enter a '0' in the K-map.
3) Enter 1's in the remaining cells of K-map.
4) Encircle/group 0's instead of 1's for carrying out the simplification.
5) Rules of simplification are exactly same those for the sop form.

Examples:

① Minimize the following std. pos exp^{n} using K-map.

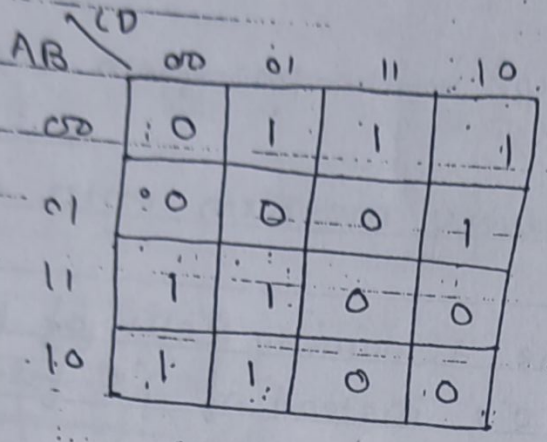
$\therefore Y = \Pi M (0,2,3,5,7)$

$\Rightarrow \therefore Y = M_0 M_2 M_3 M_5 M_7$

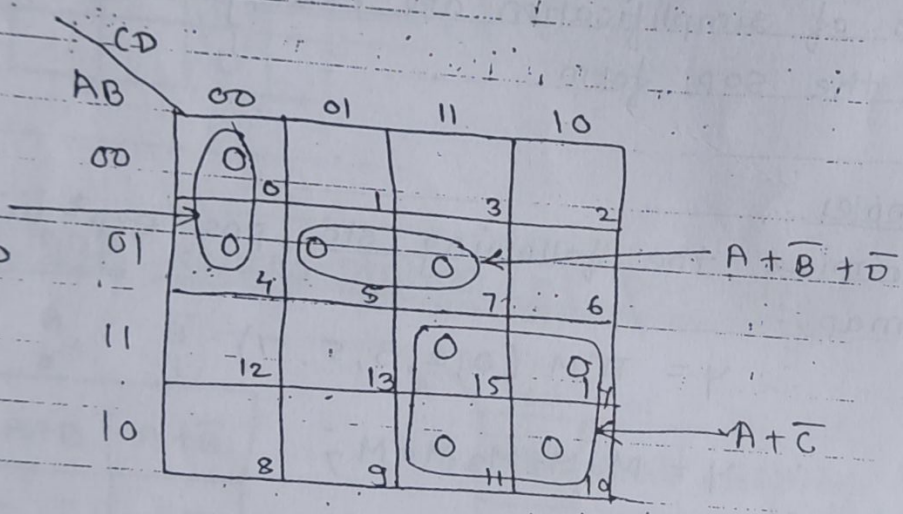


$\therefore Y = (A+C)(\bar{A}+\bar{C})(\bar{B}+\bar{C})$

2) For the K-map shown in fig write the exp^{n} for o/p in the pos form.



⇒



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