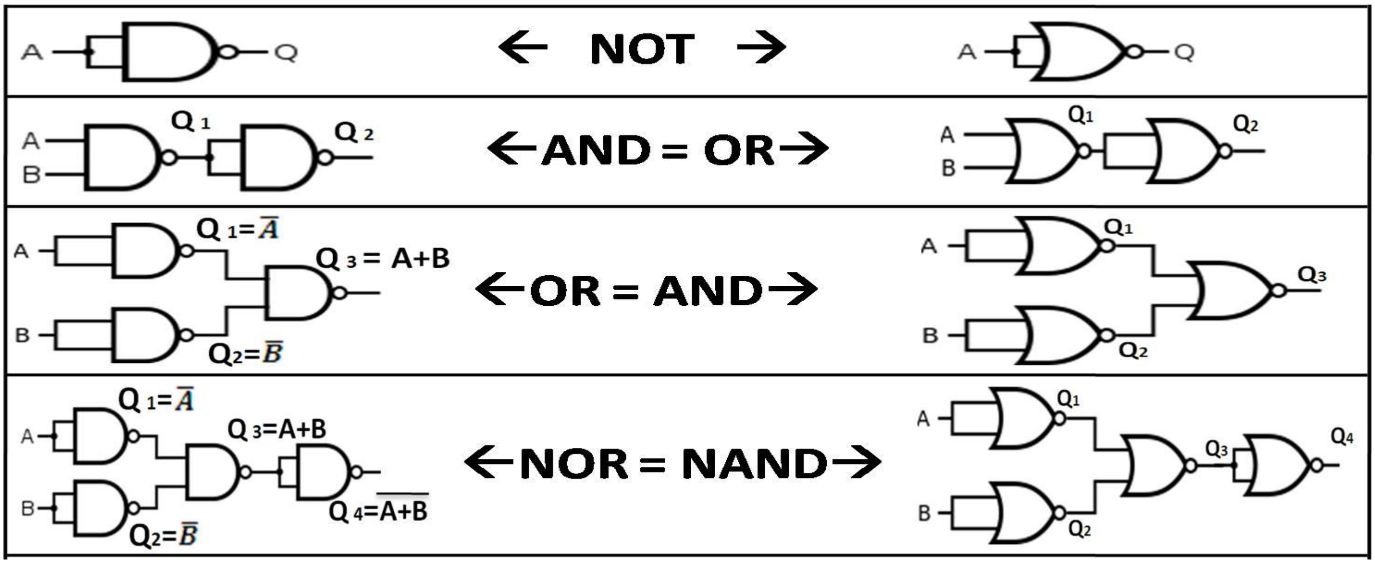
A NOTE ON

# DIGITAL ELECTRONICS

# [Theory-3] [1ST UNIT]

# BASICS OF DIGITAL ELECTRONICS

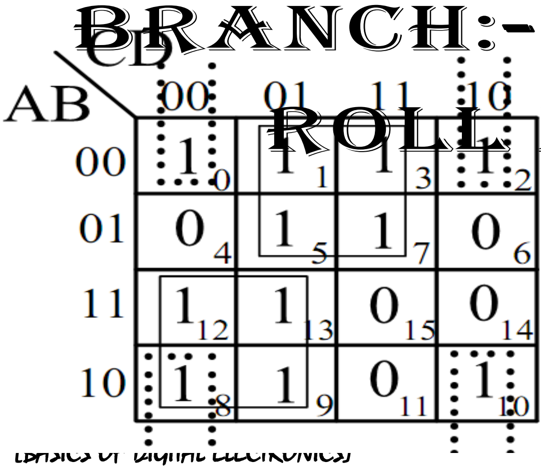
| Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 0 | 0 | 8 | 1000 | 10 | 8 | 16 | 10000 | 20 | 10 | 24 | 11000 | 30 | 18 |
| 1 | 0001 | 1 | 1 | 9 | 1001 | 11 | 9 | 17 | 10001 | 21 | 11 | 25 | 11001 | 31 | 19 |
| 2 | 0010 | 2 | 2 | 10 | 1010 | 12 | A | 18 | 10010 | 22 | 12 | 26 | 11010 | 32 | 1A |
| 3 | 0011 | 3 | 3 | 11 | 1011 | 13 | B | 19 | 10011 | 23 | 13 | 27 | 11011 | 33 | 1B |
| 4 | 0100 | 4 | 4 | 12 | 1100 | 14 | C | 20 | 10100 | 24 | 14 | 28 | 11100 | 34 | 1C |
| 5 | 0101 | 5 | 5 | 13 | 1101 | 15 | D | 21 | 10101 | 25 | 15 | 29 | 11101 | 35 | 1D |
| 6 | 0110 | 6 | 6 | 14 | 1110 | 16 | E | 22 | 10110 | 26 | 16 | 30 | 11110 | 36 | 1E |
| 7 | 0111 | 7 | 7 | 15 | 1111 | 17 | F | 23 | 10111 | 27 | 17 | 31 | 11111 | 37 | 1F |



NAME: - ______________________________

BRANCH:- 3RD SEM E.T.C.

ROLL NO:- ____________________



| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 (0) | 1 (1) | 1 (3) | 1 (2) |
| 01 | 0 (4) | 1 (5) | 1 (7) | 0 (6) |
| 11 | 1 (12) | 1 (13) | 0 (15) | 0 (14) |
| 10 | 1 (8) | 1 (9) | 0 (11) | 1 (10) |

Prepared By:-

Er. Paramananda Gouda

(Dept. of ETC, UCP Engg School)

# [UNIT - 1]

## BASICS OF DIGITAL ELECTRONICS

❖ **INTRODUCTION :-**

- The term digital refers to a process that is achieved by using discrete unit.
- The study of number systems is important from the viewpoint of understanding how data are represented before they can be processed by any digital system. It is one of the most basic topics in digital electronics.
- In number system there are different types of symbols and each symbol has an **absolute** value and also has **place** value.

1. UNIT-1: BASICS OF DIGITAL ELECTRONICS
   - 1.1. Binary, Octal. Hexadecimal Numbering Systems. Conversion from one system to another number system
   - 1.2. Arithmetic Operation-Addition, Subtraction, Multiplication, Division, 1's & 2's Complement of Binary numbers & Subtraction using Complements method
   - 1.3. Digital Code & its application & distinguish between weighted & non-weight Code, Binary codes, excess-3 and Gray Codes
   - 1.4. Logic Gates: AND, OR, NOT, NAND, NOR, Exclusive-OR, Exclusive-NOR-Symbol, Function, Expression, Truth table & Timing diagram
   - 1.5. Universal Gates & its Realization
   - 1.6. Boolean algebra, Boolean expressions. Demorgan's Theorems.
   - 1.7. Represent Logic Expression: SOP & POS forms
   - 1.8. Karnaugh map (3 & 4 Variables) & Minimization of logical expressions, don't care conditions

❖ **RADIX or BASE : -**

- The radix or base of a number system is defined as the number of different digits which can occur in each position in the number system.

❖ **RADIX POINT :-**

- The generalized form of a decimal point is known as radix point.
- In any positional number system the radix point divides the integer and fractional part.

$$N_r = [\text{Integer Part}\ \mathbf{.}\ \text{Fractional Part}]$$

↑
Radix point

❖ **NUMBER SYSTEM:-**

- In general a number in a system having base or radix ' r ' can be written as

$$a_n\ \ a_{n-1}\ \ a_{n-2}\ \ldots\ldots\ldots\ldots\ldots\ldots\ a_1\ \ a_0\ \ .\ \ a_{-1}\ \ a_{-2}\ \ldots\ldots\ldots\ldots\ldots\ldots a_{-m}$$

- This will be interpreted as

$$Y = a_n \times r^n + a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots\ldots\ldots + a_0 \times r^0 + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \ldots\ldots\ldots + a_{-m} \times r^{-m}$$

Where **Y** = Value of the entire number $a_n$ = the value of the nth digit r = radix

❖ **TYPES OF NUMBER SYSTEM:-**

- There are generally four types of number systems. They are
  1. Decimal number system
  2. Binary number system
  3. Octal number system
  4. Hexadecimal number system

❖ **DECIMAL NUMBER SYSTEM:-**

- The decimal number system contain ten unique symbols 0,1,2,3,4,5,6,7,8 and 9.
- In decimal system 10 symbols are involved, so base or radix is 10.
- It is a positional weighted number system.
- All higher numbers after '9' are represented in terms of these ten digits only.
- The value attached to the symbol depends on its location with respect to the decimal point.
- In general, $\mathbf{d_n\ \ d_{n-1}\ \ d_{n-2}\ \ldots\ldots\ldots\ldots\ d_1\ \ d_0\ \ .\ \ d_{-1}\ \ d_{-2}\ \ldots\ldots\ldots\ldots\ldots\ d_{-m}}$ is given by

$$(d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + \ldots + (d_1 \times 10^1) + (d_0 \times 10^0) + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + \ldots + (d_{-m} \times 10^{-m})$$

♣ **For Example:-** 9256.26 = 9000 + 200 + 50 + 6 + 0.2 + 0.06

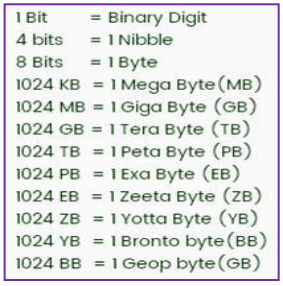
$$= 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 \times 1 + 2 \times (1/10) + 6 \times (1/100)$$

$$= 9 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 6 \times 10^{-2}$$

❖ **NOTE: -**

- ✓ The Base or Radix is equal to the no of digits. The Largest digit is one less than that of its base.
- ✓ Each digit is multiplied by the base or radix raised to an appropriate power depending upon the digit position to get its place value.
- ✓ To distinguish a number system from another number system, the radix of the system is written as a subscript to the numbers. $(9861)_{10}$ represents that this number is in Decimal Number System.

## ❖ BINARY NUMBER SYSTEM:-



| | |
|---|---|
| 1 Bit | = Binary Digit |
| 4 bits | = 1 Nibble |
| 8 Bits | = 1 Byte |
| 1024 KB | = 1 Mega Byte (MB) |
| 1024 MB | = 1 Giga Byte (GB) |
| 1024 GB | = 1 Tera Byte (TB) |
| 1024 TB | = 1 Peta Byte (PB) |
| 1024 PB | = 1 Exa Byte (EB) |
| 1024 EB | = 1 Zeeta Byte (ZB) |
| 1024 ZB | = 1 Yotta Byte (YB) |
| 1024 YB | = 1 Bronto byte (BB) |
| 1024 BB | = 1 Geop byte (GB) |

- The binary number system is a positional weighted system.
- The base or radix of this number system is 2.
- It has two independent symbols that are 0 and 1.
- A binary digit is called a bit. The radix point of a binary number system which separates integer and fraction part called *Binary Point*.
- In general, $d_n \; d_{n-1} \; d_{n-2}$ …………… $d_0 \; d_0 \; . \; d_{-1} \; d_{-2}$ …………$d_{-k}$ is given by

$$(d_n \times 2^n) + (d_{n-1} \times 2^{n-1}) + (d_{n-2} \times 2^{n-2}) + \ldots + (d_0 \times 2^0) + (d_{-1} \times 2^{-1}) + (d_{-2} \times 2^{-2}) + \ldots + (d_{-k} \times 2^{-k})$$

## ❖ OCTAL NUMBER SYSTEM:-

- It is also a positional weighted system. Its base or radix is 8.
- So its 8 independent symbols are 0,1,2,3,4,5,6 and 7.
- Any number in this system can be written as a group of these eight symbols with subscripts of 8.
- Its base 8 = $2^3$, every 3- bit group of binary can be represented by an octal digit.
- In early computer system octal number system was used but now a days it has been replaced by Hexadecimal number system.

## ❖ HEXADECIMAL NUMBER SYSTEM:-

- The hexadecimal number system is a positional weighted system.
- The base or radix of this number system is 16 such as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.
- The base 16 = $2^4$, every 4 – bit group of binary can be represented by a hexadecimal digit.
- As both numbers & letters are used, so this number system is also called as *alphanumeric number system*

## ❖ CONVERSION FROM ONE NUMBER SYSTEM TO ANOTHER SYSTEM :-

## 1. DECIMAL NUMBER SYSTEM TO OTHER NUMBER SYSTEM:- [D→B, D→O, D→H]

### (A.) Decimal to Binary Conversion:-

- The **integer** number is converted to the desired base using successive division by the base or radix.
- Divide **decimal** number successively by 2, read integer part remainder upwards to get equivalent binary.
- Multiply the fraction part by 2. Keep the integer in the product as it is and multiply the new fraction in the product by 2. The process is continued and the integers are read in the products from top to bottom.

♣ **For Example: (i) Convert $(13)_{10}$ into binary.**

**Solution:**

```
2 I 13
2 I 6    — 1
2 I 3    — 0
2 I 1    — 1
   0     — 1
```

➔ So, the Result of $(13)_{10}$ is **$(1101)_2$**

**[Also,** $(46)_{10}$→$(101110)_2$ ; $(115)_{10}$→$(1110011)_2$ ; $(25)_{10}$→$(11001)_2$ ; $(52)_{10}$→ $(110100)_2$ ; $(27)_{10}$→$(11011)_2$ **]**

♣ **(ii) Convert $(105.15)_{10}$ into Binary.**

**Solution:**

```
Integer part            Fraction part
2 I 105                 0.15 x 2 = 0.30
2 I  52   —  1          0.30 x 2 = 0.60
2 I  26   —  0          0.60 x 2 = 1.20
2 I 13    —  0          0.20 x 2 = 0.40
2 I 6     —  1          0.40 x 2 = 0.80
2 I 3     —  0          0.80 x 2 = 1.60
2 I 1     —  1
   0      —  1
```

➔ So, the Result of $(105.15)_{10}$ is **$(1101001.001001)_2$**

[**Similarly**, $(0.375)_{10}$→$(0.011)_2$ ; $(0.2)_{10}$→$(0.0011)_2$ ; $(0.75)_{10}$→$(0.110)_2$ ; $(111.625)_{10}$→$(1101111.101)_2$ ]

### (B.) Decimal to Octal Conversion:-

- To convert decimal integer number to octal, successively divide the given number by 8 till quotient is 0.
- To convert the given decimal fractions to octal successively multiply the decimal fraction and the subsequent decimal fractions by 8 till the product is 0 or till the required accuracy is obtained.

♣ **For Example: (i) Convert (378.93)10 into Octal.**

**Solution:**

| **Integer part** | **Fraction part** |
|---|---|
| 8 I 378 | 0.93 x 8 = 7.44 → 7 |
| 8 I 47 — 2 | 0.44 x 8 = 3.52→ 3 |
| 8 I 5 — 7 | 0.52 x 8 = 4.16→ 4 |
| 0 — 5 | 0.16 x 8 =1.28→ 1➔ So Result is **$(572.7341)_8$** |

**[Similarly,** $(86)_{10}$→$(126)_8$ **;** $(543)_{10}$→$(1037)_8$ **;** $(2143.53)_{10}$→$(4137.4172)_8$ **;** $(1062.3)_{10}$ →$(2046.2314)_8$ **]**

**(C.) Decimal to Hexadecimal Conversion: -**

➢ The decimal to hexadecimal conversion is same as octal.

♣ **For Example: (i) Convert $(2598.675)_{10}$ into Hexadecimal.**

**Solution:**

| **Integer part** | **Fraction part** |
|---|---|
| 16 I 2598 | 0.675 x 16 = 10.8 →A |
| 16 l 162 — 6 | 0.800 x 16 = 12.8 →C |
| 16 l 10 — 2 | 0.800 x 16 = 12.8 →C |
| 0 — 10→A | 0.800 x 16 = 12.8 →C ➔ Result is **$(A26.ACCC)_{16}$** |

**[Also,** $(3127)_{10}$→$(C37)_{16}$ **;** $(15514)_{10}$→$(3C9A)_{16}$ **;** $(82.25)_{10}$→$(52.4)_{16}$ **;** $(1195.27)_{10}$ →$(4AB.451EB8)_{16}$ **]**

| DEC→ | BIN | OCT | HEX | DEC→ | BIN | OCT | HEX |
|---|---|---|---|---|---|---|---|
| **19.0625** | 10011.0001 | 23.04 | 13.1 | **40.375** | 1001000.0110 | 50.3 | 28.6 |
| **24.125** | 11000.0010 | 30.1 | 18.2 | **49.4375** | 110001.0111 | 61.34 | 31.7 |
| **28.1875** | 11100.0011 | 34.14 | 1C.3 | **54.5** | 110110.1000 | 66.4 | 36.8 |
| **30.25** | 11110.0100 | 36.2 | 1E.4 | **57.5625** | 111001.1001 | 71.44 | 39.9 |
| **38.3125** | 100110.0101 | 46.24 | 26.5 | **62.625** | 111110.1010 | 76.5 | 3E.A |
| **91.123** | 1011011.00011 | 133.053 | 5B.1F7 | **97.884** | 1100001.111 | 141.704 | 61.E24 |
| **68.6875** | 1000100.1011 | 104.54 | 44.B | **80.875** | 1010000.1110 | 120.7 | 50.E |
| **73.75** | 1001001.1100 | 111.6 | 49.C | **88.9375** | 1011000.1111 | 130.74 | 58.F |
| **77.8125** | 1001101.1101 | 115.64 | 4D.D | **105.781** | 1101001.1100 | 151.617 | 69.C7E |

## 2. BINARY NUMBER SYSTEM TO OTHER NUMBER SYSTEM:- [B→D, B→O, B→H]

**(A.) Binary to Decimal Conversion:-**

➢ In this method, each binary digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number.

♣ **For Example: - (i) Convert $(10101)_2$ to Decimal.**

**Solution :** Binary number 10101 = $(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

= 16 + 0+ 4+ 0+ 1 = $(21)_{10}$ ➔ So the Result of $(10101)_2$ is **$(21)_{10}$**

♣ **(ii) Convert (111.101)2 to Decimal.**

**Solution:** $(111.101)_2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

= 4+ 2+ 1 + 0.5 + 0 + 0.125 = $(7.625)_{10}$ ➔ So Result of $(111.101)_2$ is **$(7.625)_{10}$**

**[Also,** $(1011.111)_2$→$(11.875)_{10}$ **;** $(101101.01)_2$→$(45.25)_{10}$ **;** $(1101.11)_2$→$(13.75)_{10}$ **;** $(100.001)_2$→$(4.125)_{10}$ **;]**

**(B.) Binary to Octal Conversion:-**

➢ For conversion binary to octal the binary numbers are divided into groups of 3 bits each, starting at the binary point and proceeding towards left and right.

♣ **For Example: (i) Convert $(101111010110.110110011)_2$ into Octal.**

**Solution :**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Group of 3 bits are | 101 | 111 | 010 | 110 | **.** | 110 | 110 | 011 |
| Convert each group into octal = | 5 | 7 | 2 | 6 | **.** | 6 | 6 | 3 |

➔ So, The result of **$(101111010110.110110011)_2$** is **$(5726.663)_8$**

♣ **(ii) Convert $(10101111001.0111)_2$ into Octal.**

**Solution :** Binary number 10 101 111 001 **.** 011 1

Group of 3 bits are =010 101 111 001 **.** 011 100

Convert each group into octal = 2 5 7 1 **.** 3 4 → The result is **$(2571.34)_8$**

| BINARY | OCTAL | BINARY | OCTAL | BINARY | OCTAL | BINARY | OCTAL |
|---|---|---|---|---|---|---|---|
| **000** | **0** | **010** | **2** | **100** | **4** | **110** | **6** |
| **001** | **1** | **011** | **3** | **101** | **5** | **111** | **7** |

**[Also,** $(1011.101)_2 \rightarrow (13.5)_8$; $(10101.10)_2 \rightarrow (25.4)_8$ ; $(101101.01)_2 \rightarrow (55.2)_8$ ; $(1101000.11)_2 \rightarrow (150.6)_8$ **]**

### (C.) Binary to Hexadecimal Conversion:-

- ➢ For conversion binary to hexadecimal number the binary numbers starting from the binary point, groups are made of 4 bits each, on either side of the binary point.

♣ **For Example: - (i) Convert $(1011011011)_2$ into hexadecimal.**

**Solution:** Given Binary number 10 1101 1011

Group of 4 bits are 0010 1101 1011

Convert each group into hex = 2 D B → So The result is **$(2DB)_{16}$**

♣ **(ii) Convert (01011111011.011111)2 into hexadecimal.**

**Solution:** Given Binary number 010 1111 1011 **.** 0111 11

Group of 3 bits are = 0010 1111 1011 **.** 0111 1100

Convert each group into octal = 2 F B **.** 7 C → The result is **$(2FB.7C)_{16}$**

**[Also,** $(10110.001)_2 \rightarrow (16.2)_{16}$; $(111011.10)_2 \rightarrow (3B.8)_{16}$ ; $(1100111.01)_2 \rightarrow (67.4)_{16}$ ; $(101010.11)_2 \rightarrow (2A.C)_{16}$ **]**

| **BIN →** | **DEC** | **OCT** | **HEX** | **BIN →** | **DEC** | **OCT** | **HEX** |
|---|---|---|---|---|---|---|---|
| **10111011.0101** | 187.3125 | 273.24 | BB.5 | **10101010.1010** | 170.626 | 252.5 | AA.A |
| **10111011.0101** | 179.7656 | 263.62 | 5B.C8 | **1100011000.110** | 792.75 | 1430.6 | 318.C |
| **10110011.11001** | 204.50 | 314.40 | CC.8 | **10101101.11111** | 173.9678 | 255.76 | AD.F8 |
| **11001100.1000** | 141.8125 | 115.64 | 4D.D | **110101101.00101** | 173.1562 | 255.12 | AD.B8 |
| **10001001.011** | 137.1875 | 111.14 | 89.3 | **1011101.10111** | 93.71875 | 135.56 | 5D.B8 |
| **1110011.1110** | 115.875 | 163.70 | 73.E | **1000100111.110** | 551.75 | 1047.6 | 227.C |
| **100011.0110** | 35.375 | 43.3 | 23.6 | **11111100.0011** | 252.1875 | 374.14 | FC.3 |
| **1001001.0011** | 73.1875 | 111.14 | 49.3 | **11000111.001101** | 199.20312 | 307.15 | C7.34 |
| **11001100.1100** | 204.75 | 314.6 | CC.C | **111001110011.1111** | 3699,9375 | 3163.74 | E73.F |
| **1110111.1110** | 199.875 | 167.7 | 77.E | **1100001100.00111** | 780.21875 | 1414.16 | 30C.38 |

## 3. OCTAL NUMBER SYSTEM TO OTHER NUMBER SYSTEM: – [O→B, O→D, O→H]

### (A.) Octal to Binary Conversion: -

- ➢ To convert a given a octal number to binary, replace each octal digit by its 3- bit binary equivalent.

♣ **For Example: - Convert $(367.52)_8$ into binary.**

**Solution:** Given Octal number is 3 6 7 **.** 5 2

Convert each group octal = 011 110 111 **.** 101 010

→ Result of $(367.52)_8$ is **$(011110111.101010)_2$**

### (B.) Octal to Decimal Conversion: -

- ➢ For conversion octal to decimal number, multiply each digit in the octal number by the weight of its position and add all the product terms

♣ **For Example: - Convert $(4057.06)_8$ to decimal**

**Solution:** $(4057.06)_8 = 4x8^3 + 0x8^2 + 5x8^1 + 7x8^0 + 0x8^{-1} + 6x8^{-2} = 2048 + 0 + 40 + 7 + 0 + 0.0937$

$= (2095.\ 0937)_{10}$ → So the Result is **$(2095.0937)_{10}$**

**[Also,** $(1035.205)_8 \rightarrow (1035.26)_{10}$ ; $(264.34)_8 \rightarrow (180.437)_{10}$ ; $(213.45)_8 \rightarrow (139.5781)_{10}$ ; $(42.24)_8 \rightarrow (34.875)_{10}$ **]**

### (C.) Octal to Hexadecimal Conversion:-

- For conversion of octal to Hexadecimal, first convert the given octal number to binary and then binary number to hexadecimal.

♣ **For Example: - Convert $(756.603)_8$ to Hexadecimal.**

**Solution:-** Given octal no. 7 5 6 . 6 0 3
Convert each octal digit to binary = 111 101 110 . 110 000 011
Group of 4bits are = 0001 1110 1110 . 1100 0001 1000
Convert 4 bits group to hex. = 1 E E . C 1 8 ➔ **$(1EE.C18)_{16}$**

**[Also,** $(55)_8$→$(2D)_{16}$;$(32.15)_8$→$(1A.3A)_{16}$;$(106.63)_8$→$(42.CC)_{16}$;$(1062.57)_8$→$(232.BC)_{16}$ ; $(642)_8$ →$(1A2)_{16}$**]**

| OCT→ | BIN | HEX | DEC | OCT→ | BIN | HEX | DEC |
|---|---|---|---|---|---|---|---|
| **3.33** | 011.011011 | 3.421888 | 36C | **36.26** | 011110.010110 | 30.34375 | 1E. 58 |
| **6.14** | 110.001100 | 6.1875 | 6.30 | **46.46** | 100110.100110 | 38.59375 | 26.98 |
| **7.14** | 111.001100 | 7.1875 | 7.3 | **73.11** | 111011.001001 | 59.14063 | 3B24 |
| **9.43** | 1001.100011 | 9.546875 | 9.8C | **213.17** | 010001011.001111 | 139.2344 | 8B. 3C |
| **12.7** | 001010.111 | 10.875 | A. E | **234** | 010011100 | 156 | 9C |
| **13.66** | 001011.110110 | 11.84375 | B. D8 | **416.14** | 100001110.001100 | 270.1875 | 10E.C |
| **14.17** | 001100.001111 | 12.23438 | C. 3C | **435.1** | 100011101.001 | 285.125 | 11D. 2 |
| **21.46** | 010001.100110 | 17.59375 | 11.98 | **464.2** | 100110100.010 | 308.25 | 134.4 |
| **24.31** | 010100.011001 | 20.39063 | 14.64 | **732.33** | 111011010.011011 | 474.4219 | 1DA. 6C |

## 4. HEXADECIMAL NUMBER SYSTEM: - [H→B, H→D, H→O]

### (A.) Hexadecimal to Binary Conversion:-

- For conversion of hexadecimal to binary, replace hexadecimal digit by its 4 bit binary group.

♣ **For Example: Convert $(3A9E.B0D)_{16}$ into Binary.**

**Solution:** Given Hexadecimal number is 3 A 9 E . B 0 D
Convert each hexadecimal Digit to 4 bit binary = 0011 1010 1001 1110 . 1011 0000 1101
➔ Result of $(3A9E.B0D)_8$ is **$(0011101010011110.101100001101)_2$**

### (B.) Hexadecimal to Decimal Conversion: -

- For conversion of hexadecimal to decimal, multiply each digit in the hexadecimal number by its position weight and add all those product terms.

♣ **For example: - Convert $(A0F9.0EB)_{16}$ to Decimal**

**Solution:** $(A0F9.0EB)_{16} = (10x16^3)+(0x16^2)+(15x16^1)+(9x16^0)+(0x16^{-1})+(14x16^{-2})+(11x16^{-3})$
$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026 =$ **$(41209.0572)_{10}$**

### (C.) Hexadecimal to Octal Conversion: -

- For conversion of hexadecimal to octal, first convert the given hexadecimal number to binary and then binary number to octal.

♣ **For Example: - Convert $(B9F.AE)_{16}$ to Octal.**

**Solution :-** Given hexadecimal no. is B 9 F . A E
Convert each hexadecimal digit to binary = 1011 1001 1111 . 1010 1110
Group of 3 bits are = 101 110 011 111 . 101 011 100
Convert 3 bits group to octal. = 5 6 3 7 . 5 3 4 ➔ **$(5637.534)_8$**

| Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 0 | 0 | 8 | 1000 | 10 | 8 | 16 | 10000 | 20 | 10 | 24 | 11000 | 30 | 18 |
| 1 | 0001 | 1 | 1 | 9 | 1001 | 11 | 9 | 17 | 10001 | 21 | 11 | 25 | 11001 | 31 | 19 |
| 2 | 0010 | 2 | 2 | 10 | 1010 | 12 | A | 18 | 10010 | 22 | 12 | 26 | 11010 | 32 | 1A |
| 3 | 0011 | 3 | 3 | 11 | 1011 | 13 | B | 19 | 10011 | 23 | 13 | 27 | 11011 | 33 | 1B |
| 4 | 0100 | 4 | 4 | 12 | 1100 | 14 | C | 20 | 10100 | 24 | 14 | 28 | 11100 | 34 | 1C |
| 5 | 0101 | 5 | 5 | 13 | 1101 | 15 | D | 21 | 10101 | 25 | 15 | 29 | 11101 | 35 | 1D |
| 6 | 0110 | 6 | 6 | 14 | 1110 | 16 | E | 22 | 10110 | 26 | 16 | 30 | 11110 | 36 | 1E |
| 7 | 0111 | 7 | 7 | 15 | 1111 | 17 | F | 23 | 10111 | 27 | 17 | 31 | 11111 | 37 | 1F |

| HEX | BIN | OCT | DEC | HEX | BIN | OCT | DEC |
|---|---|---|---|---|---|---|---|
| $(2B)_{16}$ | 101011 | $(53)_8$ | $(43)_{10}$ | $(4FC.2A)_{16}$ | 1001111101.0010101 | 2374.124 | 1276.16406 |
| $(98.64)_{16}$ | 10011000.011001 | $(230.31)_8$ | 152.3906 | (5DC.2B8) | 10111011100.001010111 | $(2734.127)_8$ | 1500.16992 |
| $(123.26)_{16}$ | 100100011.0010011 | $(443.114)_8$ | 291.148437 | $(35.F4)_{16}$ | 110101.111101 | $(65.75)_8$ | 53.953125 |
| $(AB.88)_{16}$ | 10101011.10001 | $(253.42)_8$ | 171.53125 | $(25.BAD)_{16}$ | 100101.101110101101 | $(45.5655)_8$ | 37.729736 |
| (21F.56C) | 1000011111.0101011011 | 1037.2554 | 543.338867 | $(80.0D5)_{16}$ | 10000000.000011010101) | 200.0325 | 128.052001 |
| $(593.ABC)_{16}$ | 10110010011.1010101111 | 2623.5274 | 1427.67089 | $(3C.58)_{16}$ | 111100.01011000 | $(74.26)_8$ | (60.34375) |
| $(1012.1F)_{16}$ | 1000000010010.00011111 | 10022.076 | 4114.12109 | $(E8C.5)_{16}$ | 111010001100.0101 | $(7214.24)_8$ | 3724.3125 |
| $(AB.BA)_{16}$ | 10101011.1011101 | $(253.564)_8$ | 171.72656 | $(250.9C)_{16}$ | 1001010000.100111 | $(1120.47)_8$ | 592.609375 |
| $(211)_{16}$ | 10000.10001 | $(1021)_8$ | $(529)_{10}$ | $(999.DC)_{16}$ | 100110011001.110111 | $(4631.67)_8$ | 2457.85937 |
| $(A0B.C0)_{16}$ | 1001000001011.11 | $(5013.6)_8$ | $(2571.75)_{10}$ | $(9FED.4)_{16}$ | 1001111111101101.01 | 117755.2 | (40941.25) |
|  |  |  |  |  |  |  |  |

## –: ARITHEMATIC OPERATION ON BINARY NUMBER SYSTEM:–

### 1. BINARY ADDITION:-

➢ The binary addition rules are as follows

**(i)** 0 + 0 = 0 ; **(ii)** 0 + 1 = 1 ; **(iii)** 1 + 0 = 1 ; **(iv)** 1 + 1 = 10 , i.e. 0 with a carry of 1

♣ **For Example: - (i) Add $(100101)_2$ and $(1101111)_2$.**

**Solution: -**

```
     1 0 0 1 0 1
+  1 1 0 1 1 1 1
 1 0 0 1 0 1 0 0
```

→ Result is $\mathbf{(10010100)_2}$

♣ **(ii) Add (1001)2 and (1110)2.**

**Solution: -**

```
     1 0 0 1
+    1 1 1 0
   1 0 1 1 1
```

→ Result is $\mathbf{(10111)_2}$

**{Also,** 11111+1011=**101010;** 111+1011=**10010;** 1101.01+101.11=**10011.00;** 1101.11 + 1.01 =**1111.00;** 111+1011=**10010;** 1101.101+10001.01 =**11110.111;** 1011+0101=**10000;** 11011+10101+10111=**1000111}**

### 2. BINARY SUBTRACTION:-

➢ The binary subtraction rules are as follows

**(i)** 0 - 0 = 0 ; **(ii)** 1 - 1 = 0 ; **(iii)** 1 - 0 = 1 ; **(iv)** 0 - 1 = 1 , with a borrow of **1**

♣ **For Example: - (i) Subtract $(111.111)_2$ from $(1010.01)_2$.**

**Solution :-**

```
   1 0 1 0. 0 1 0
-    1 1 1 . 1 1 1
   0 0 1 0 . 0 1 1
```

→ Result is $\mathbf{(0010.011)_2}$

**(ii)**

```
   1 1 0 1
-  1 0 1 1
   0 0 1 0
```

→ Result is $\mathbf{(11)_2}$

**{Also,** 10000-1=**1111;** 1101-1010=**11;** 100011-1111=**10100;** 11100-11011=**1;** 1110001-111001=**111000}**

### 3. BINARY MULTIPLICATION: -

➢ The binary multiplication rules are as follows

**(i)** 0 x 0 = 0 ; **(ii)** 1 x 1 = 1 ; **(iii)** 1 x 0 = 0 ; **(iv)** 0 x 1 = 0

♣ **For Example: - Multiply (1101)2 by (110)2.**

**Solution :-**

```
        1 1 0 1
x        1 1 0
        0 0 0 0
      1 1 0 1
+   1 1 0 1
    1 0 0 1 1 1 0
```

→ Result of $\mathbf{(1101)_2}$ **x** $\mathbf{(110)_2 = (1001110)_2}$

**{Also,** 1001 x 110=**110110 ;** 10111 x 101 = **1110011 ;** 101 x 1110 = **1000110 ;** 110.01 x 1.10 =**1001.0110;** 111 x 101 =**100011;** 1111 x 1101 = **11000011;** 11011 x 11101 = **1100001111;** 11.001 x 10.01 =**111.00001 }**

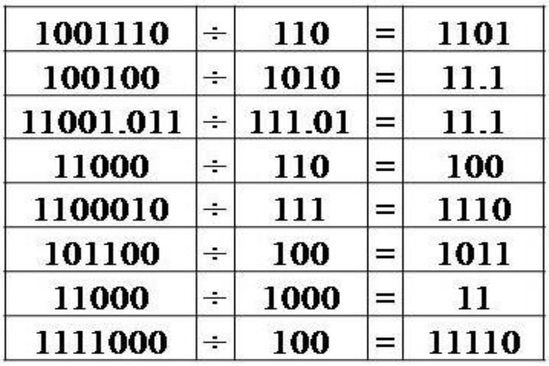
### 4. BINARY DIVISION:-

- The binary division is very simple and similar to decimal number system. The division by '0' is meaningless. So we have only 2 rules **(i)** 0 ÷ 1 = 0 **(ii)** 1 ÷ 1 = 1

♣ **For Example: - Divide $(10110)_2$ by $(110)_2$.**

**Solution: -**

```
110) 101101 (111.1
   -  110
       1010
        110
        1001
         110
         110
         110
         000 ➔ Result is (111.1)2
```



| | | | | |
|---|---|---|---|---|
| 1001110 | ÷ | 110 | = | 1101 |
| 100100 | ÷ | 1010 | = | 11.1 |
| 11001.011 | ÷ | 111.01 | = | 11.1 |
| 11000 | ÷ | 110 | = | 100 |
| 1100010 | ÷ | 111 | = | 1110 |
| 101100 | ÷ | 100 | = | 1011 |
| 11000 | ÷ | 1000 | = | 11 |
| 1111000 | ÷ | 100 | = | 11110 |

### ❖ SIGNED MAGNITUDE REPRESENTATION: -

- It is the method of representing a signed number. It use an extra bit at the left most end.
- This additional bit is usually known as sign bit and placed at most significant end to represent the sign.
- A " **0** " is used to represent **" + "** Sign where as " **1** " is used to represent **" - "** Sign.

♣ **For Example: -** +7 → 0, 111 **;** - 6→ 1, 110 **;** + 12→ 0,1100 **;** - 12 → 1,1100 **;** - 17 → 1, 10001

### ❖ 1'S COMPLEMENT REPRESENTATION: -

- The 1's complement of a binary number is obtained by changing each 0 → 1 and each 1 → 0.
- This completed value represents the negative of the original number.

♣ **For Example: - Find $(1100)_2$ 1's complement.**

```
Solution: - Given         1    1    0    0
1's complement is         0    0    1    1    ➔ Result is (0011)2
```

**{Similarly,** 101010→**010101;** 1001001→**0110110;** 01110→**10001;** 11110111→**1000;** 101101→10010**}**

### ❖ ADDITION BY USING 1'S COMPLEMENT METHOD: -

To get the Accurate Result, Find Number of Digits (**n**) by using the formula **$2^{n-1}$ -1 > Max ($D_1$, $D_2$, R)**

♣ **Example: -** Add +4 and +9 by 1's Complement Method

```
Solution: -   1's complement of +4 →    0100
              1's complement of +9 →+   1001
                                        1101 [Both are +ve numbers. So add them as in binary numbers]
```

♣ **Example: -** Add +3 and -8 by 1's Complement Method

```
Solution: -   1's complement of +3 →    0011  [As +ve numbers. So, Find only its binary equivalent]
              1's complement of - 8 →+  0111  [As -ve numbers. So, 1's complement of 8 (1000) = 0111]
                                        1010  [As result is -ve no. to check the result Find 1's of 1010
    i.e. 0101 (=5), So the sum of +3 and -8 is -5]
```

### ❖ SUBTRACTION BY USING 1'S COMPLEMENT METHOD: -

♣ **Example: -** Subtract **7** from 21 by 1's Complement Method

**Solution: -** 21 – 7 = 21 + (-7) → Find 1's complement of 7 to get -7. Then add with 21 as binary addition.

```
      1's complement of 21 →    1 0 1 0 1  [As +ve numbers. So, Find only its binary equivalent]
      1's complement of - 7 →+  1 1 0 0 0  [As -ve numbers. So, 1's complement of 7 (00111) = 11000]
                               1 0 1 1 0 1
                              +          1   [As carry Obtained add this carry to the result]
                                 0 1 1 1 0  [As result is +ve no. to the result (1110)2 = (14)10]
```

♣ **Subtract $(10000)_2$ from $(11010)_2$ using 1's complement.**

```
Solution:-  1 1 0 1 0                1 1 0 1 0                       =   26
          - 1 0 0 0 0     =>      +  0 1 1 1 1  (1's complement)     = - 16
                    Carry   →     1  0 1 0 0 1                         + 10
                                  +          1
                                     0 1 0 1 0  = +10   ➔   Result is +10
```

✍ *If carry (end around) is obtained during addition then add this carry to LSB to get the correct result.*

## ❖ 2'S COMPLEMENT REPRESENTATION: -

➢ The 2's complement of a binary number is a binary number which is obtained by adding 1 to the 1's complement of a number i.e. **2's complement = 1's complement + 1**

♣ **For Example: - Find $(1010)_2$ 2's complement.**

**Solution: -** Given 1010 ➔ 1's complement is 0101 + 1 = **0110** ➔ 2's complement of $(1010)_2$ is **$(0110)_2$**

## ❖ DIFFERENT METHODS TO FIND 2'S COMPLEMENT OF A NUMBER: -

♣ **Example: -** Find 2's complement of $(-18)_{10}$

### METHOD-1

**Solution: -** Binary of $(18)_{10}$ is = 00010010

1's Complement of this = 11101101

Add +1 to this + 1

2's Complement of 00010010 → 11101110

### METHOD-2

**Solution: -** Binary representation of $(18)_{10}$ is = 00010010

Stating from LSB move towards left and coping upto first 1 = 01

Then Complement remaining bits to find 2's complement no = 11101110

2's Complement of $(00010010)_2$ → $(11101110)_2$

### METHOD-3

**Solution: -** The word length of the number is 8 → 1 0 0 0 0 0 0 0 0

Subtracting (18=00010010) from this - 0 0 0 1 0 0 1 0

2's Complement of $(00010010)_2$ → $(1\ 1\ 1\ 0\ 1\ 1\ 1\ 0)_2$

♣ [ **Similarly,** 2's complement of $(-12)_{10}$ = $(00001100)_2$ →$(11110100)_2$ **;** 2's complement of $(-101)_{10}$ = $(01100101)_2$ →$(10011011)_2$ **;** 2's complement of $(-128)_{10}$ = $(10000000)_2$ →$(10000000)_2$ ]

## ❖ SUBSTRACTION USING 2'S COMPLEMENT METHOD: -

➢ In 2's complement subtraction, add the 2's complement of subtrahend to the minuend. If there is a carry out, ignore it. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 2's complement form. Then take its 2's complement to get the magnitude in binary.

♣ **For Example: - Subtract $(1010100)_2$ from $(1010100)_2$ using 2's complement.**

**Solution: -** 1 0 1 0 1 0 0 1 0 1 0 1 0 0 = 84

- 1 0 1 0 1 0 0 => + 0 1 0 1 1 0 0 (2's complement) = - 84

**1** 0 0 0 0 0 0 0 (Ignore the carry) 0

= 0 (Result = 0) ➔ So the result is +0000000 = **0**

♣ **Example: -** Subtract **18** from **17** by 2's Complement Method

**Solution: -** 17 – 18 **=** 17 + (-18) → Find 1's complement of 18 to get -18. Then add with 17 like binary

2's complement of 17 → 1 0 0 0 1 [As +ve numbers. So, Find only its binary equivalent]

2's complement of - 18 →+ 0 1 1 1 0 [As -ve numbers. So, 2's complement of 18 (10010) = 01110]

**1 1 1 1 1** [To check result find 2's of $(11111)_2$ = $(00001)_2$. So Result→ -1]

♣ **Example: -** Add **- 15** and **-20** by 2's Complement Method

**Solution: -** 2's complement of -15 → 1 1 1 1 0 0 0 1

2's complement of -20 → 1 1 1 0 1 1 0 0

$\cancel{1}$ **1 1 0 1 1 1 0 1** [By Neglating Carry and As it is a negative number, So to check the result find the 2's complement of 11011101 is 00100011 ( =35) So the result is ➔ -35 ]

| DEC | SM | 1'S | 2'S | DEC | SM | 1'S | 2'S | DEC | SM | 1'S | 2'S | DEC | SM | 1'S | 2'S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0,0000 | 0000 | 0000 | **8** | 0,1000 | 1000 | 1000 | **- 1** | 1,0001 | 1110 | 0010 | **- 9** | 1,1001 | 0110 | 0111 |
| **1** | 0,0001 | 0001 | 0001 | **9** | 0,1001 | 1001 | 1001 | **- 2** | 1,0010 | 1101 | 0011 | **- 10** | 1,1010 | 0101 | 0110 |
| **2** | 0,0010 | 0010 | 0010 | **10** | 0,1010 | 1010 | 1010 | **- 3** | 1,0011 | 1100 | 0100 | **- 11** | 1,1011 | 0100 | 0101 |
| **3** | 0,0011 | 0011 | 0011 | **11** | 0,1011 | 1011 | 1011 | **- 4** | 1,0100 | 1011 | 0101 | **- 12** | 1,1100 | 0011 | 0100 |
| **4** | 0,0100 | 0100 | 0100 | **12** | 0,1100 | 1100 | 1100 | **- 5** | 1,0101 | 1010 | 1011 | **- 13** | 1,1101 | 0010 | 0011 |
| **5** | 0,0101 | 0101 | 0101 | **13** | 0,1101 | 1101 | 1101 | **- 6** | 1,0110 | 1001 | 1010 | **- 14** | 1,1110 | 0001 | 0010 |
| **6** | 0,0110 | 0110 | 0110 | **14** | 0,1110 | 1110 | 1110 | **- 7** | 1,0111 | 1000 | 1001 | **- 15** | 1,1111 | 0000 | 0001 |
| **7** | 0,0111 | 0111 | 0111 | **15** | 0,1111 | 1111 | 1111 | **- 8** | 1,1000 | 0111 | 1000 | | | | |

## ARITHMETIC OPERATIONS ON BINARY NUMBER SYSTEM

| SN | DATA - 1 | DATA - 2 | SUM | DIFFERENCE | PRODUCT | DIVISION ( A ÷ B) | |
|---|---|---|---|---|---|---|---|
| | (A) | (B) | (A + B) | (A – B) | (A x B) | Result | Reminder |
| 1. | 1100101 | 1011 | 1110000 | 1011010 | 10001010111 | 1001 | 1 |
| 2. | 10011001101 | 10101 | 10011100010 | 10010111000 | 110010011010001 | 111010 | 1011 |
| 3. | 101110.01 | 101.1 | 11111110.011 | 101000.11 | 11111110.011 | 1000 | 100.1 |
| 4. | 1100011.001 | 11.11 | 1100110.111 | 1011111.011 | 101110011.10111 | 11010 | 10.11 |
| 5. | 1111.1111 | 111.11 | 10111.1011 | 1000.0011 | 1111011.100001 | 10 | 111 |

## ARITHMETIC OPERATIONS ON OCTAL NUMBER SYSTEM

| SN | DATA - 1 | DATA - 2 | SUM | DIFFERENCE | PRODUCT | DIVISION ( A ÷ B) | |
|---|---|---|---|---|---|---|---|
| | (A) | (B) | (A + B) | (A – B) | (A x B) | Result | Reminder |
| 1. | 345742 | 256 | 346220 | 345464 | 116111634 | 1244 | 152 |
| 2. | 23410154 | 1605 | 23411761 | 23406347 | 42257746034 | 13656 | 206 |
| 3. | 7534.21 | 2.56 | 7536.77 | 7531.43 | 24702.6616 | 2646 | 0.75 |
| 4. | 62100.45 | 34.21 | 62134.66 | 62044.24 | 2610661.2565 | 1613 | 33.52 |
| 5. | 7354.215 | 11.754 | 7366.171 | 7342.241 | 112245.414374 | 577 | 5.171 |

## ARITHMETIC OPERATION ON HEXADECIMAL NUMBER SYSTEM

| SN | DATA - 1 | DATA - 2 | SUM | DIFFERENCE | PRODUCT | DIVISION ( A ÷ B) | |
|---|---|---|---|---|---|---|---|
| | (A) | (B) | (A + B) | (A – B) | (A x B) | Result | Reminder |
| 1. | 2F3A29 | B4F | 2F4578 | 2F2EDA | 21612B5A7 | 42D | 146 |
| 2. | 3BAD45 | 19A | 3BAEDF | 3BABAB | 5F938082 | 2542 | 191 |
| 3. | 6436C.2 | 1B3.9 | 6451F.B | 641B8.9 | AA816A7.32 | 3AE | B0.4 |
| 4. | 68BD.BD | 2.AD | 68C0.6A | 68BB.1 | 11843.B2B9 | 2724 | 2.69 |
| 5. | CD5.48F | AB.6 | D80.A8F | C29.E8F | 89747.B3AA | 13.2B | 5.FD |

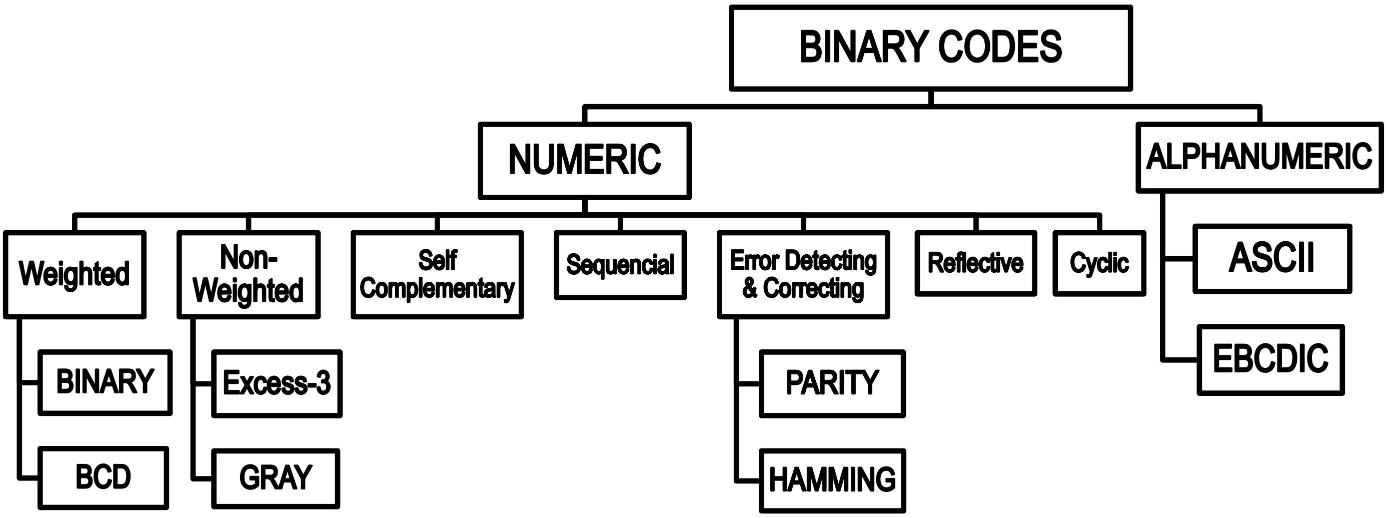
## ❖ DIGITAL CODES: -

- Codes are the representation of information in particular format.
- The information may include numbers, alphabets and symbols which can man and machine recognize.
- Date is transmitted in terms of code-words over long distance it is. During the transmission process, error may introduce. To detect and correct the errors, special codes are used in digital communication.
- This requires the conversion of the incoming data into binary format before it can be processed.
- There is various possible ways of doing this and this process is called encoding.
- To achieve the reverse of it, we use decoders.

### ♣ APPLICATION : -

- To represent numeric or alpha-numeric or special characters in only binary digits i.e. 0 or 1.
- To check whether a character transmitted in the coded form is correctly received if not then to correct it. i. e. for detecting and correcting of errors. Different codes are used to store and transmit data efficiently.

### ❖ CLASSIFICATION OF BINARY CODES: -



### ❖ WEIGHTED AND NON-WEIGHTED CODES: -

- In weighted codes, for each position or bit, there is specific weight attached.
- For example, in Binary Number, each bit is assigned particular weight $2^n$ where 'n' is the bit number for n = 0, 1, 2, 3 and 4 the weights are 1, 2, 4, 8, and 16 respectively. **Example**: - BCD, Binary etc.
- Non-weighted codes are codes which are not assigned with any weight to each digit position, i.e., each digit position within the number are not assigned fixed value. **Example: -** Excess-3 code & Gray codes.

## ❖ BINARY CODED DECIMAL (BCD):-

- BCD is a weighted code. In weighted codes, each successive digit from right to left represents weights equal to some specified value and to get the equivalent decimal number add the products of the weights by the corresponding binary digit.
- 8421 is the most common because 8421 BCD is the most natural amongst the other possible codes.
- In this code each decimal digit is expressed by its 4 bit binary equivalent.

♣ **For example: - $(567)_{10}$ is encoded in various 4 bit codes →** 0101 0110 0111 (In 8421 BCD Code)

| Decimal | Binary | BCD | Decimal | Binary | BCD | Decimal | Binary | BCD |
|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 0000 | 10 | 1010 | 0001 0000 | 45 | 101101 | 0100 0101 |
| 2 | 0010 | 0010 | 12 | 1100 | 0001 0010 | 52 | 110100 | 0101 0010 |
| 4 | 0100 | 0100 | 15 | 1111 | 0001 0101 | 59 | 111011 | 0101 1001 |
| 5 | 0101 | 0101 | 23 | 10111 | 0010 0011 | 98 | 1100010 | 1001 1000 |
| 7 | 0111 | 0111 | 27 | 11011 | 0010 0111 | 102 | 1100110 | 0001 0000 0010 |
| 9 | 1001 | 1001 | 30 | 11110 | 0011 0000 | 105 | 1101001 | 0001 0000 0101 |

- **Examples: -** Convert the BCD code $(100101010111)_{BCD}$ to its decimal equivalent.
  **Solution: -** 1001 0101 0111 → $(9\ 5\ 7)_{10}$
- **Examples: -** Convert the decimal number $(215.43)_{10}$ to it equivalent BCD Code.
  **Solution: -** 215.43→ $(0010\ 0001\ 0101.\ 0100\ 0011)_{BCD}$
- **Examples: -** Convert the binary number $(1101.10)_2$ to it equivalent BCD Code.
  **Solution: - (**$1101\ .10)_2 = (13.5)_{10} = (0001\ 0011.0101)_{BCD}$

## BCD ADDITION: -

- Addition of BCD (8421) is performed by adding two digits of binary, starting from least significant digit.
- In case if the result is an illegal code (Greater than 9) or if there is a carry out of one then add 0110 (6) to that part(4-bit group) only and add the resulting carry to the next most significant.
- **For Example: - Add 647 with 782 using BCD Addition.**

| | | | |
|---|---|---|---|
| **Solution: -** 6 4 7 | | 0110 0100 0111 | (647 in BCD) |
| + 7 8 2 | ➔ (+) | 0111 1000 0010 | (782 in BCD) |
| 1 4 2 9 | | 1101 1100 1001 | (1st & 2nd Terms are Illegal codes) |
| | | + 0110 +0110 | (Add 0110 to Illegal Parts only) |
| | | + 1 | (As Carry obtained, so add this to next MSB) |
| | | 1 0100 0010 1001 | |
| | | 1 4 2 9 | (Corrected Sum) ➔ Result is $(\mathbf{1429})_{10}$ |

- **For Example: - Add 679.6 with 536.8 using BCD Addition.**

| | | | |
|---|---|---|---|
| **Solution: -** 6 7 9 . 6 | | 0110 0111 1001 . 0110 | (679.6 in BCD) |
| + 5 3 6 . 8 | ➔ + | 0101 0011 0110 . 1000 | (536.8 in BCD) |
| 1 2 1 6 . 4 | | 1011 1010 1111 . 1110 | (All are illegal codes) |
| | | + 0110 +0110 +0110 +0110 | (Add 0110 to each) |
| | | 0001 0010 0001 0110 . 0100 | |
| | | 1 2 1 6 . 4 | (Corrected Sum) ➔ Result is $(\mathbf{1216.4})_{10}$ |

## BCD SUBTRACTION: -

- The BCD subtraction is performed by subtracting the digits of each 4 – bit group of the subtrahend from corresponding 4 – bit group of the minuend in the binary starting from the LSD.
- If there is no borrow from the next higher group then no correction is required.
- If there is a borrow from the next group, then $6_{10}$ (0110) is subtracted from difference term of this group.
- **For Example: - Subtract 147.8 from 206.7 using 8421 BCD code.**

| | | | |
|---|---|---|---|
| **Solution: -** 2 0 6 . 7 | | 0010 0000 0110 . 0111 | (206.7 in BCD) |
| (-) 1 4 7 . 8 | ➔ (-) | 0001 0100 0111 . 1000 | (147.8 in BCD) |
| 5 8 . 9 | | 0000 1011 1110 . 1111 | (Borrows are present) |
| | | (-)0110 (-)0110 (-)0110 | |
| | | 0101 1000 . 1001 | |
| | | 5 8 . 9 | [Corrected Difference is ➔ $(\mathbf{58.9})_{10}$ ] |

## EXCESS THREE (XS-3) CODE: -

- The Excess-3 code, also called XS-3, is a non- weighted BCD code. It is a sequential code.
- This can be derived by adding **03** (0011) to each decimal digit before converting into equivalent binary.
- It is also a self complementing code. Because In this code, the 1's complement of the excess-3 code is excess-3 code for the 9's complement of the corresponding decimal number.
- For Ex**:** The excess-3 code for 2 is 0101, the 1's complement of 0101 is 1010. It is the excess-3 code of 7 and 7 is the 9's complement of 2.
- **Examples: - Convert decimal number 129 to its equivalent excess-3 number.**
  **Solution: -** Add (+3) to each of the decimal digits → 4 5 12 ➔ $(0100\ 0101\ 1100)_{XS\text{-}3}$
- **Examples: - Convert XS-3 code 10011010 to its equivalent decimal number.**
  **Solution: -** Given XS-3 is 10011001. Subtract (3 or 0011) from each of the digits → 0110 0111 ➔$(67)_{10}$

| Decimal | BCD | After Adding 3 to each decimal digits | XS-3 | Decimal | BCD | After Adding 3 to each decimal digits | XS-3 |
|---|---|---|---|---|---|---|---|
| 0 | 0000 | 3 | 0011 | 12 | 0001 0010 | 45 | 0100 0101 |
| 5 | 0101 | 8 | 1000 | 23 | 0010 0011 | 56 | 0101 0110 |
| 7 | 0111 | 10 | 1010 | 45 | 0100 0101 | 78 | 0111 1000 |
| 9 | 1001 | 12 | 1100 | 63 | 0111 0011 | 96 | 1001 0110 |

## ❖ XS-3 ADDITION: -

- ➢ In XS-3 addition, add the XS-3 numbers by adding the 4 bit groups in each column starting from the LSD. If there is no carry out from the addition of any of the 4 bit groups, subtract 0011 from the sum term of those groups. If there is a carry out, add 0011 to the sum term of those groups

♣ **For example: - Add 37 and 28 using XS-3 code.**

```
Solution: -   3  7              0110  1010     (37 <6 10> in XS-3)
            + 2  8        ➔   + 0101  1011     (28 < 5 11> in XS-3)
              6  5              1100  0101     (Carry is generated Add this propagated carry to MSB)
                              - 0011 +0011     (Add 0110 to correct 0101 & Subtract 0011 to 1100)
                                1001  1000     (Corrected sum in XS-3 = 65₁₀)  {9-3 ; 8-3}
```

## ❖ XS-3 SUBTRACTION:-

- ➢ To subtract in XS-3 number by subtracting each 4-bit group of the subtrahend from corresponding 4-bit group of the minuend starting from the LSB.
- ➢ If there is no borrow from the next 4-bit group, add 0011 to the difference term of such groups.
- ➢ If there is a borrow, subtract 0011 from the difference term.

♣ **For Example: - Subtract 175 from 267 using XS-3 code**.)

```
Solution: -   2  6  7          0101   1001   1010  (267<5 9 10> in XS-3)
          (-) 1  7  5    ➔   - 0100   1010   1000  (175<4 10 8> in XS-3)
              0  9  2          0000   1111   0010    (Correct 0010 and 0000 both by adding 0011
                             + 0011 - 0011 +0011        and Correct 1111 by subtracting 0011)
                               0011   1100   0101 (Corrected difference in XS-3 = 92₁₀) [3-3; 12-3; 5-3]
```

## ❖ ALPHANUMERIC CODES: -

- ➢ Alphanumeric codes are used to encode the characters of alphabet in addition to the decimal digits.
- ➢ They are used primarily for transmitting data between computers and its I/O devices such as printer, keyboards and video display terminals.
- ➢ Most popular modern alphanumeric codes are the ASCII Code and the EBCDIC Code.

## ❖ ASCII CODE: -

- ➢ The American Standard Code for Information Interchange (ASCII) pronounced as 'ASKEE' is widely used alphanumeric code. This is basically a 7 bit code.
- ➢ The number of different bit patterns that can be created with 7 bits is $2^7 = 128$.
- ➢ The ASCII can be used to encode both the uppercase and lowercase characters of the alphabet (52 symbols) and some special symbols in addition to the 10 decimal digits.

## ❖ EBCDIC CODE:-

- ➢ The Extended Binary Coded Decimal Interchange Code (EBCDIC) pronounced as 'eb-si-dik' is an 8 bit alphanumeric code. Since $2^8 = 256$ bit patterns can be formed with 8 bits.
- ➢ This code is used to encode all symbols and characters found in ASCII along with some other special characters. In fact, many of the bit patterns in EBCDIC code are not assigned. It is used by most large computers to communicate in alphanumeric data. The table shown below shows the EBCDIC code.

## ❖ GRAY CODE: -

- ➢ The gray code is a non-weighted code. It is not a BCD code. It is cyclic code because successive words in this differ in one bit position only i.e. it is a unit distance code.
- ➢ Gray code is used in instrumentation & data acquisition systems where linear or angular displacement is measured. They are also used in shaft encoders, I/O devices, A/D converters & other peripheral devices.

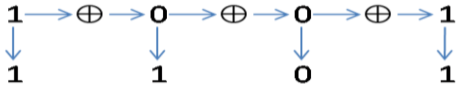
## ❖ BINARY- TO – GRAY CONVERSION: -

- ➢ If an n-bit binary number is represented by $B_n$ $B_{n-1}$ ….. $B_1$ and its gray code equivalent by $G_n$ $G_{n-1}$ … $G_1$, Where $B_n$ and $G_n$ are the MSBs , then gray code bits are obtained from the binary code as follows : -

  $G_n = B_n$ ; $G_{n-1} = B_n \oplus B_{n-1}$; . . . $G_1 = B_2 \oplus B_1$, Where symbol "⊕ " stands for Exclusive-OR (X-OR)

♣ **For Example: - Convert the binary 1001 to the Gray code.**

**Solution: -** Binary →

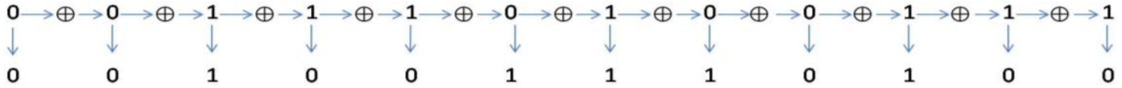


Gray → ➔ The gray code is **(1101)** G

♣ **Example: - Convert the (3A7) 16 into the Gray code.**

**Solution: -** Now find the binary of $(3A7)_{16}$ → $(001110100111)_2$ ➔ (001001110100) G

Binary →

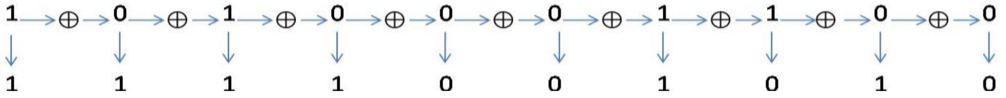


Gray →

♣ **Example: - Convert the (652) 10 into the Gray code.**

**Solution: -** Now find the binary of $(652)_{10}$ → $(1010001100)_2$ ➔ (1111001010) G

Binary →



Gray →

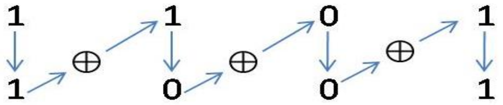
## ❖ GRAY- TO - BINARY CONVERSION: -

- ➢ If an n-bit gray number is represented by $G_n$ $G_{n-1}$ ----- $G_1$ and its binary equivalent by $B_n$ $B_{n-1}$ - - - - $B_1$, then binary bits are obtained from Gray bits as follows :

  $B_n = G_n$ ; $B_{n-1} = B_n \oplus G_{n-1}$ . . . . . $B_1 = B_2 \oplus G_1$

♣ **For Example: - Convert the Gray code 1101 to the binary.**

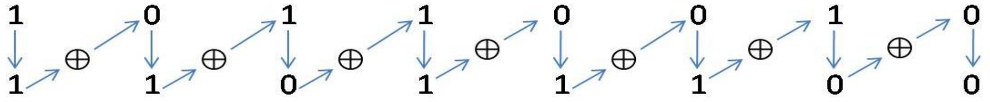
**Solution: -** Gray →



Binary → ➔ Result is **(1001)B**

♣ **Example: - Convert Gray code 10110010 into equivalent Binary, Octal, Hexadecimal and Decimal.**

**Solution: -** Gray →



Binary →

➔ Thus the equivalent results are $(11011100)_2$, $(334)_8$, $(DC)_{16}$, $(220)_{10}$.

## ❖ ERROR DETECTION & CORRECTION CODE: -

- ➢ When information in digital form is transmitted in long distance, errors may get introduced & 1 becomes 0 OR 0 becomes 1 and a wrong information may received at destination. Special codes are used to detect and correct such errors. **Parity** and **Hamming** Codes are used for error detection and correction process.

### ♣ ERROR DETECTION CODE: -

- ➢ The problem of wrong information received at destination due to introduction of error in transmission of digital formed data over long distance is overcome by using error-detecting codes.
- ➢ The simple error detection codes are (i) Parity Codes and (ii) Block Parity codes.

### PARITY CODES: -

- ➢ Usually ASCII code is used for sending digital data over telephone lines.
- ➢ The 1-bit or more than 1-bit errors may occur in transmitted data. To detect these errors, a parity bit is usually transmitted along with the data bits. At the receiving end, parity will be checked.
- ➢ The format of parity code word is $PM_{N-1}$ $M_{N-2}$ $M_{N-3}$… $M_3M_2M_1M_0$ where P is a parity bit and $M_{N-1}$ to $M_0$ are N message bits. There are two types of parity: →

  (i) even parity and (ii) odd parity.
- ➢ For an even parity code, the total number of 1s in the parity code word is even.

| 4-BIT MESSAGE | | | | Even Parity Code Word | | | | | Odd Parity Code Word | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M3 | M2 | M1 | M0 | P | M3 | M2 | M1 | M0 | P | M3 | M2 | M1 | M0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- For an odd parity code, the total number of 1s in the parity code word is odd.
- A 4-bit message with even parity & odd parity is in Table.
- The single parity bit code can detect single bit errors.
- If the error is more than 1-bit, it cannot be detected.
- For example, assume the even parity code word sent by the transmitter is 10111 and the code word received by the receiver is 10011, the parity of received code word is odd, it shows that a 1-bit error is introduced in channel.
- But if the receiving code word is 10001, then the parity of received code word is even, and it shows that there is no error introduced over the channel, actually two bits are changed over the path.
- The 1-bit parity code word can detect 1-bit errors, but it cannot detect the location of the error or correct the error.
- The circuit that generates the parity bit at the transmitting end is called a parity generator and the circuit that checks the parity at the receiving end is called a **parity generator** and the circuit that checks the parity at the receiving end is called a **parity checker**

## LOGIC GATES

### ❖ LOGIC GATES: -

- Logic gates are the fundamental building blocks of digital systems.
- The name Logic is derived from the ability of such a device to make decisions.
- It produces one output level when some combinations of input levels are present and a different output level when other combinations of input levels are present.
- There are three Basic types of gates such as NOT Gate, AND Gate & OR Gate.
- Computers are able to perform very complex logic operations by interconnecting these basic gates.
- The interconnection of gates to perform a variety of logical operations is called logic design.
- Logic gates are electronic circuits because they are made up of a number of electronic devices and components. Each gate is dedicated to specific logic operation.
- Inputs and outputs of logic gates can occur only in two levels.
- These two levels are termed HIGH and LOW **or** TRUE and FALSE **or** ON and OFF **or** simply 1 and 0.
- The table which lists all the possible combinations of input variables and the corresponding outputs is called a **Truth Table**. It shows how the logic circuits output responds to various combinations of inputs.

### ❖ DIFFERENT LEVEL OF LOGIC: -

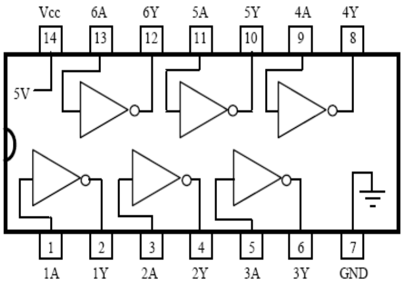
- A logic in which the voltage levels represents logic 1 & logic 0. Level logic may be +ve or -ve logic.
- **Positive Logic: -** A positive logic system is the one in which the higher of the two voltage levels represents the logic 1 and the lower of the two voltages level represents the logic 0.
- **Negative Logic: -** A negative logic system is the one in which the lower of the two voltage levels represents the logic 1 and the higher of the two voltages level represents the logic 0.

### ❖ ANALOG SIGNALS Vs DIGITAL SIGNALS: -

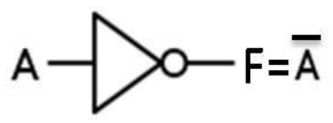
- A Signal which can assumes any value in a given range is known as **Analog Signal.**
- A Sinusoidal signal and amplitude modulated signal are examples of Analog Signals.
- A signal which can assume only two possible values is known as a **Digital Signal.**
- Voltage levels of 0V or 1V and presence or absence of pulses are examples of Digital Signals.
- Analog signal is *continuous* and can assume any value in a given range;
- Where as a digital signal can have only two *discrete* values.

## *DIFFERENT TYPES OF LOGIC GATES: - [Fundamental Gates]*

### ♣ NOT GATE (INVERTER): -



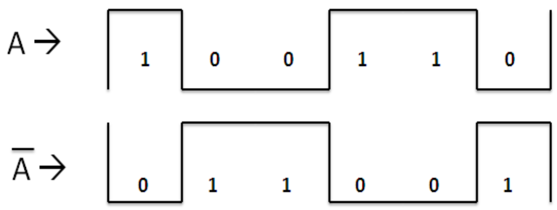
- A NOT gate, also called an inverter, has only one input and one output.
- It is a device whose output is always complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state when its inputs is in logic 1 state.
- The IC **7404** contains **Six** numbers of NOT Gate.
- The logic symbol and Truth Table of NOT Gate is shown in figures.



**[**Logic Symbol of NOT Gate**]**

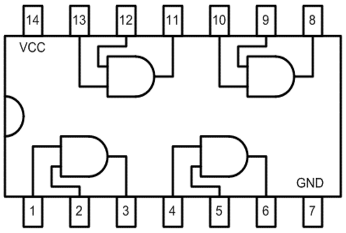
| INPUT (A) | OUTPUT ($\bar{A}$) |
|---|---|
| 0 | **1** |
| 1 | 0 |

[Truth Table of NOT Gate]

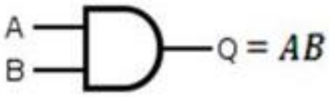


[Timing Diagram NOT Gate]

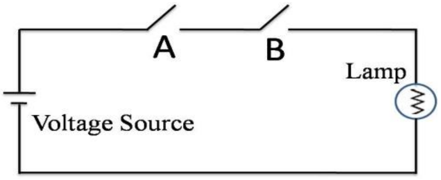
## ♣ AND GATE:-



- An AND gate has two or more inputs but only one output.
- The output is logic 1 state only when each one of its inputs is at logic 1 state.
- The output is logic 0 state even if one of its inputs is at logic 0 state.
- An **AND Gate** is a logic circuit whose output is HIGH when all inputs are HIGH. The IC **7408** contains **four** numbers of **two I/P** AND Gates.
- Similarly IC **7411** contains **three** numbers of **three I/P** AND Gates and IC **7421** contains **two** numbers of **four I/P** AND Gates.



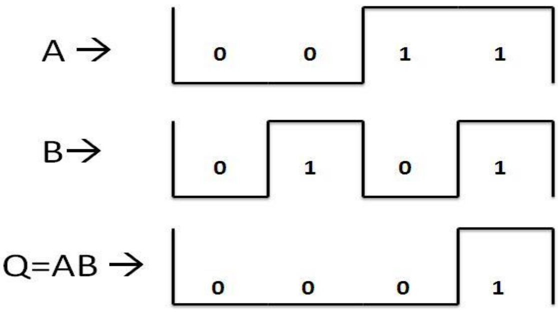
**[**Logic Symbol of AND Gate**]**



[AND function by Switch]

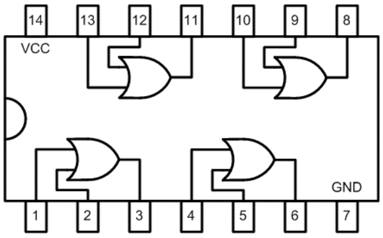
| IN PUTS | | OUT PUT |
|---|---|---|
| **A** | **B** | **Q = A . B** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**[**Truth Table of AND Gate**]**

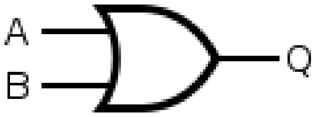


[Timing Diagram AND Gate]

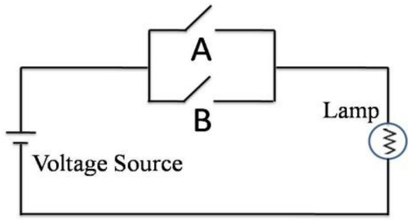
## ♣ OR GATE: -



- An OR gate may have two or more inputs but only one output.
- The output is logic 1 state, even if one of its input is in logic 1 state.
- The output is logic 0, only when each one of its inputs is in logic 0 state.
- An **OR Gate** is a logic circuit whose output is HIGH when at least one input is HIGH. The IC **7432** contains **four** numbers of **two I/P** OR Gates.



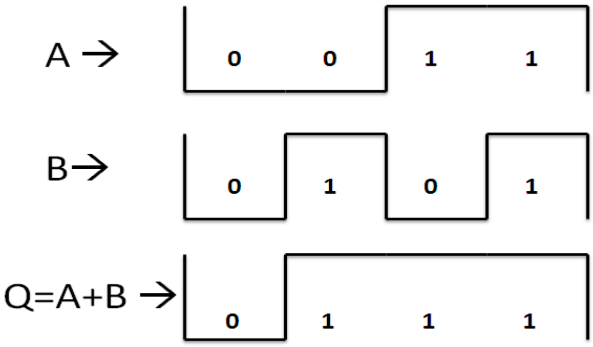
**[**Logic Symbol of OR Gate**]**



**[**OR function by Switch**]**

| IN PUT | | OUT PUT |
|---|---|---|
| **A** | **B** | **Q =A + B** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

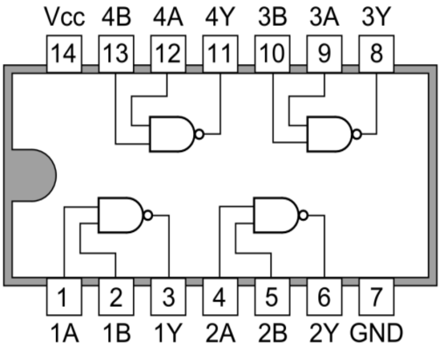
**[**Truth Table of OR Gate**]**



**[**Timing Diagram OR Gate**]**

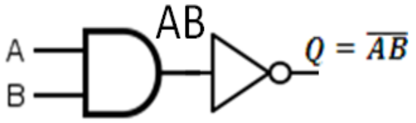
## *Universal Gates: -*

### ♣ NAND GATE: -

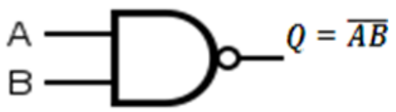


- NAND (Means NOT AND) gate is the combination of an AND gate and a NOT gate i.e. the output of AND Gate is feed to NOT Gate.
- The expression for the output of NAND Gate can be written as $\overline{AB}$.
- The output is logic 0 when each of the input is logic 1 and for any other combination of inputs, the output is logic 1. An **NAND Gate** is a logic circuit whose output is LOW when all the inputs are HIGH.
- The IC **7400** contains **four** numbers of **two I/P** NAND Gates.

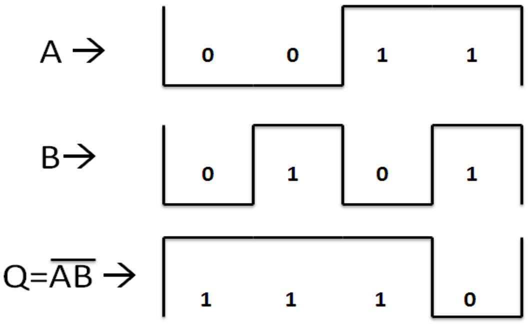
- IC 7410 contains 3 numbers of 3 I/P NAND Gate & IC 7420 contains 2 numbers of 4 I/P NAND Gates.



[AND Gate Followed by Not Gate]



**[Logic Symbol of NAND Gate]**



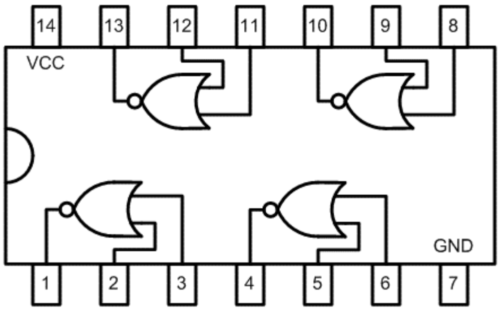
[Truth Table of NAND Gate]

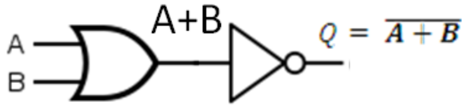
| IN PUT | | OUT PUT |
|---|---|---|
| A | B | $Q = \overline{A.B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

[Timing Diagram NAND Gate]

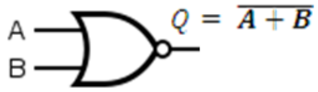
♣ **NOR GATE:-**

- NOR (Means NOT OR) gate is the combination of an OR gate and a NOT gate i.e. the output of OR Gate is feed to NOT Gate.
- The output expression of NOR Gate is written as $Q = \overline{A + B}$.
- The output is logic 1, only when all of its input is logic 0 and for other combination of inputs, the output is a logic 0 level.
- An **NOR Gate** is a logic circuit whose output is LOW when any one input is HIGH. The IC **7402** contains **four** numbers of **two I/P** NOR Gates. Similarly IC 7427 contains three numbers of three I/P NOR Gates and IC 7425 contains two numbers of four I/P NOR gate.





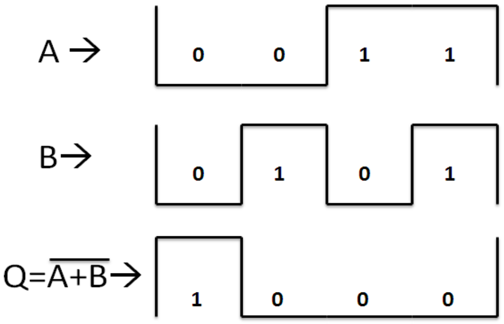
[OR Gate Followed by Not Gate]



**[Logic Symbol of NOR Gate]**

| IN PUT | | OUT PUT |
|---|---|---|
| A | B | Q=A+B |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

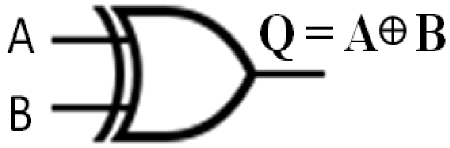
[Truth Table of NOR Gate]



[ Timing Diagram NOR Gate]

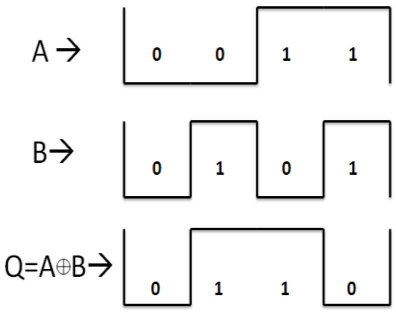
*EXCLUSIVE GATES: -* **Ex-OR (X-OR) GATE:-**

- An X-OR gate is a two input, one output logic circuit.
- The output is 1 when one and only one of its two inputs is 1.
- When both inputs are 0 or both inputs are 1, then output is 0.
- The IC **7486** contains **four** numbers of **two I/P** Ex-OR Gates.

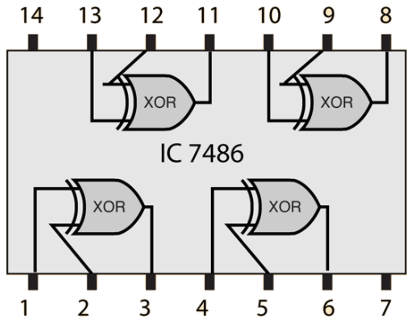


| INPUT | | OUTPUT |
|---|---|---|
| A | B | $Q = A\oplus B$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

[Truth Table of EX-OR Gate]



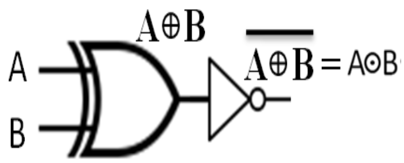
[Timing Diagram Ex-OR Gate]

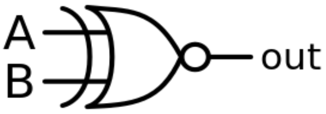


[IC Diagram Ex-OR Gate]

♣ **Ex-NOR (X-NOR) GATE:-**

- An X-NOR gate is the combination of an X-OR gate and a NOT gate.
- An X-NOR gate is a two input, one output logic circuit. The output is logic 1 only when both the inputs are 0 or when both the inputs are 1.
- The output is logic 0 when one of the inputs is logic 0 and other is 1.
- The IC **74266** contains **four** numbers of **two I/P** Ex-NOR Gates.

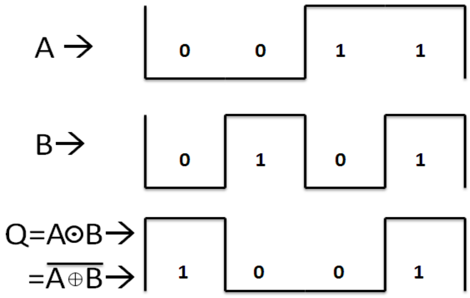




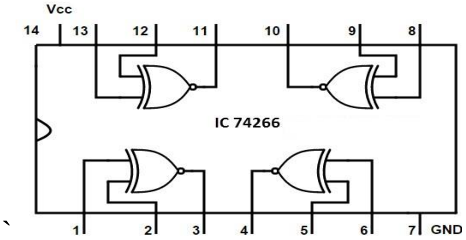
[Logic Symbol of EX-NOR Gate]

| INPUT | | OUTPUT |
|---|---|---|
| **A** | **B** | Q=A⊙B |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

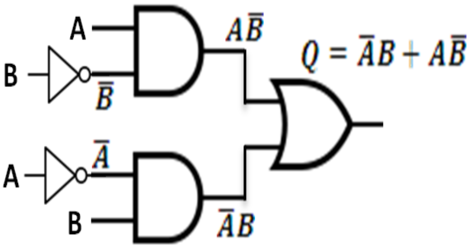
[Truth Table of EX-NOR Gate]



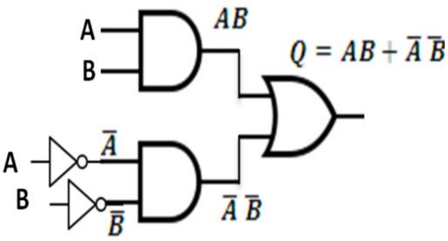
[Timing Diagram EX-NOR Gate]



**[**IC Diagram of X-NOR Gates**]**



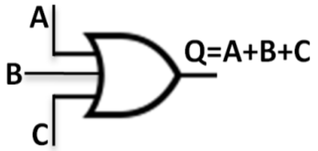
[EX-OR Gate by Basic Gates]


A
B
AB
$Q = AB + \overline{A}\,\overline{B}$
A
$\overline{A}$
B
$\overline{B}$
$\overline{A}\,\overline{B}$

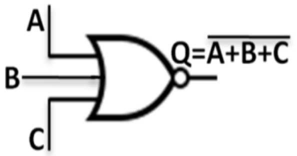
[EX-NOR Gate by Basic Gates]

- ♣ The expression for the output of EX-OR Gate can be written as $\boldsymbol{Q = A \oplus B = \overline{A}B + A\overline{B}}$.
- ♣ The expression for the output of EX-NOR Gate can be written as $\boldsymbol{Q = A \odot B = AB + \overline{A}\,\overline{B}}$.
- ♣ The X-NOR of two variable is the complement of the X-OR gates i.e. $\boldsymbol{A \odot B = \overline{A \oplus B}}$ .
- ♣ But X-NOR of three variables is not the complement of it's X-OR gate i.e. $\boldsymbol{A \odot B \odot C \neq \overline{A \oplus B \oplus C}}$ .

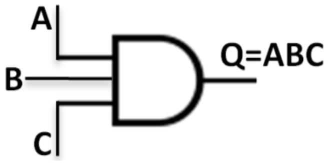
## THREE INPUT GATES <SYMBOLS> : -



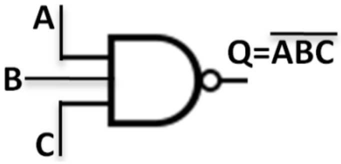
(OR Gate)



(NOR Gate)



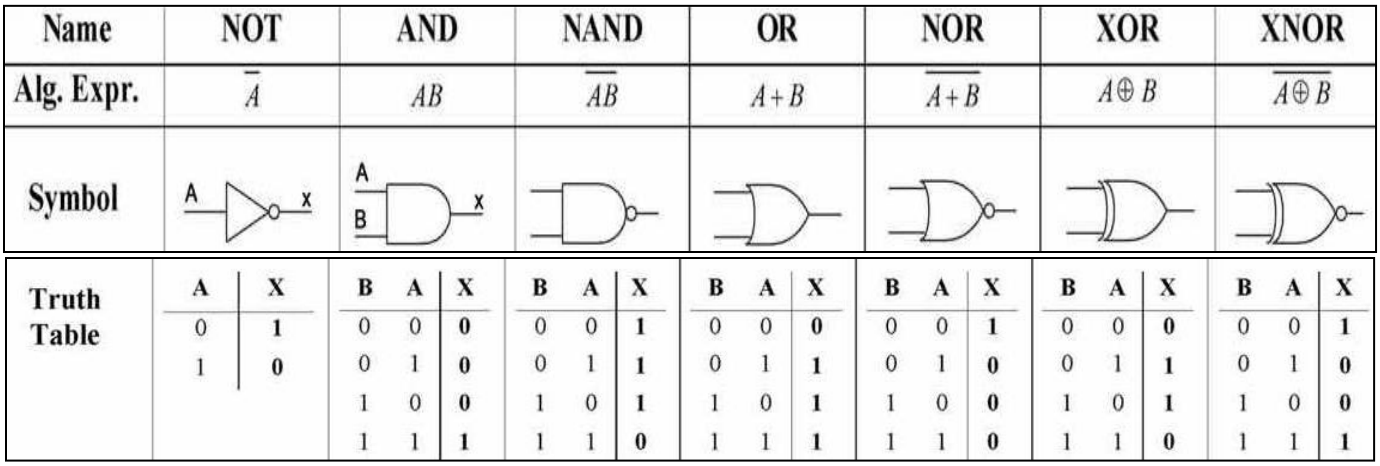
(AND Gate)



(NAND Gate)

## THREE INPUT GATES <TRUTH TABLES> : - {OR, AND, NOR, NAND}

| A | B | C | A+B+C | A | B | C | ABC | A | B | C | $\overline{A+B+C}$ | A | B | C | $\overline{ABC}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |



| Name | NOT | AND | NAND | OR | NOR | XOR | XNOR |
|---|---|---|---|---|---|---|---|
| Alg. Expr. | $\overline{A}$ | $AB$ | $\overline{AB}$ | $A+B$ | $\overline{A+B}$ | $A \oplus B$ | $\overline{A \oplus B}$ |
| Symbol | A x | A<br>B x | | | | | |
| Truth Table | A X<br>0 1<br>1 0 | B A X<br>0 0 0<br>0 1 0<br>1 0 0<br>1 1 1 | B A X<br>0 0 1<br>0 1 1<br>1 0 1<br>1 1 0 | B A X<br>0 0 0<br>0 1 1<br>1 0 1<br>1 1 1 | B A X<br>0 0 1<br>0 1 0<br>1 0 0<br>1 1 0 | B A X<br>0 0 0<br>0 1 1<br>1 0 1<br>1 1 0 | B A X<br>0 0 1<br>0 1 0<br>1 0 0<br>1 1 1 |

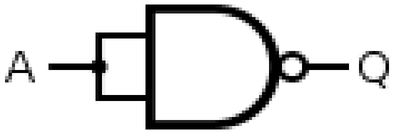
## ❖ UNIVERSAL GATES: -

- There are three basic gates AND, OR and NOT.
- There are two universal gates NAND and NOR, each of which can realize logic circuits single handedly.
- The NAND and NOR gates are called **universal** building blocks.
- As by using these Gates we can construct other type of logic Gates.
- Both NAND and NOR gates can perform all logic functions i.e. AND, OR, NOT, EX-OR and EX-NOR.

## NAND GATE AS A UNIVERSAL GATES: –

### a) NAND Gate as Inverter (NOT Gate) : -

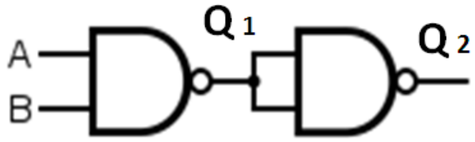
- When two inputs of NAND gate are joined together, so that it has one input and one output, resulting circuit act as a NOT Gate.
- Here Input is = **A** and Output is = $\bar{A}$



**[NOT** Gate by Using NAND Gates only]

### b) NAND Gate as an AND Gate : -

- Here, we use two NAND Gates in a manner as shown in figure.
- The output of first NAND Gate is given to the second NAND gate acting as inverter. The resulting circuit act as a AND Gate.
- Here Input are = **A & B** and Output is = **AB.**



**[AND** Gate using NAND Gates only]

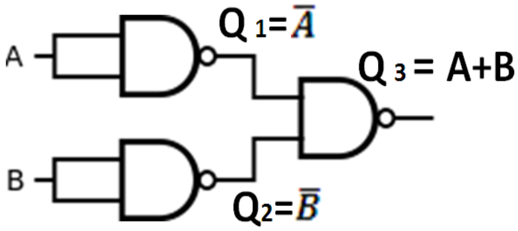
AND Gate by NAND Gate

| A | B | Q 1 | Q 2 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

[Truth Table for NAND Gate as an **AND** Gate]

### c) NAND Gate as OR Gate : -

- For this purpose we use three gates in a manner as shown in figure.
- The first two gates are operated as NOT Gates and their outputs are fed to third NAND Gate.
- The resulting Circuit act as an OR gate.
- Here Input are = **A & B** and Output is = **A+B.**



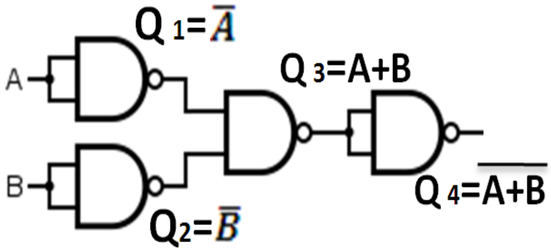
**[OR** Gate using NAND Gates only]

OR Gate by NAND Gate

| A | B | Q 1 | Q 2 | Q 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

[Truth Table for NAND Gate as an **OR** Gate]

### d) NAND Gate as NOR Gate : -



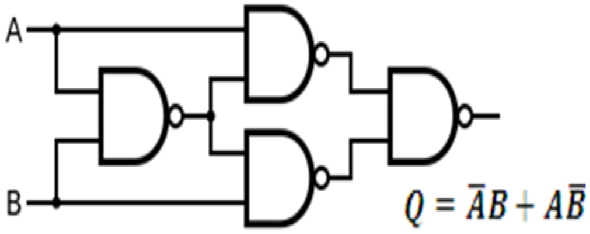
**[NOR** Gate using NAND Gates only]

NOR Gate by NAND Gate

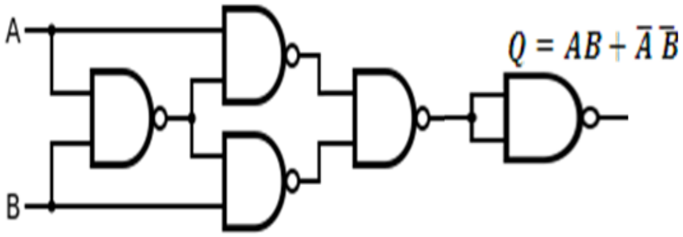
| A | B | Q 1 | Q 2 | Q 3 | Q 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 |

[Truth Table for NAND Gate as an **NOR** Gate]

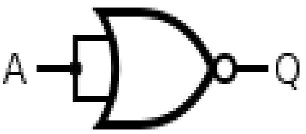
### e) NAND Gate as EX-OR Gate : -



### NAND Gate as EX-NOR Gate : -



## NOR GATE AS A UNIVERSAL GATES: -



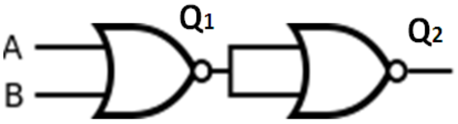
**a) NOR Gate as Inverter (NOT Gate) : -**

- When two inputs of NOR gate are joined together, so that it has one input and one output, resulting circuit act as a NOT Gate.
- Here Input is = **A** and Output is = $\overline{A}$

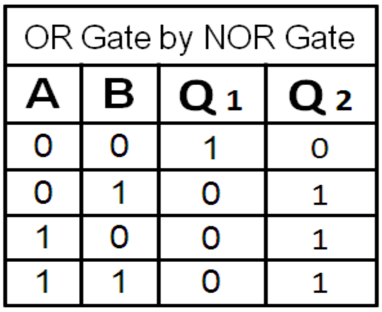
[**NOT** Gate by using NOR Gates only]

**b) NOR Gate as an OR Gate : -**

- Here, we use two NOR Gates in a manner as shown in figure.
- The output of first NOR Gate is given to the second NOR gate acting as inverter.
- The resulting circuit act as a OR Gate.
- Here Input are = **A & B** and Output is = **A+B.**



[**OR** Gate by using NOR Gates only] [Truth Table for NOR as **OR** Gate→]

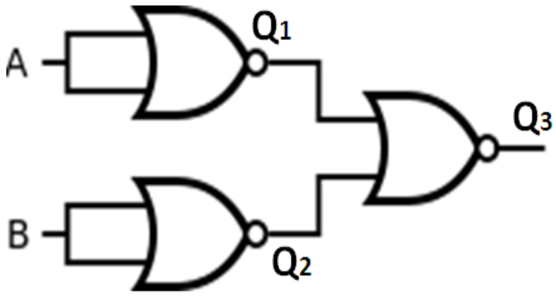


OR Gate by NOR Gate

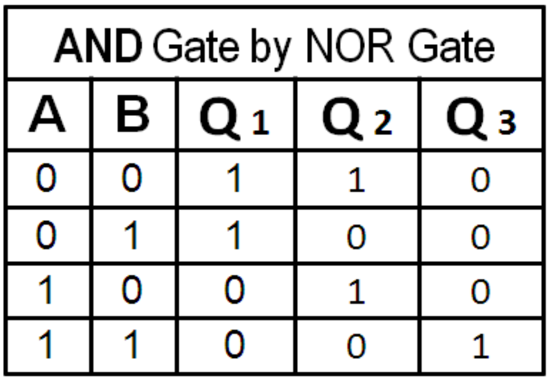
| A | B | Q 1 | Q 2 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

**c) NOR Gate as AND Gate : -**

- For this purpose we use three gates in a manner as shown in figure.
- The first two gates are operated as NOT Gates and their outputs are fed to third NOR Gate.
- The resulting Circuit act as an AND gate.
- Here Input are = **A & B** and Output is = **AB.**



[**AND** Gate by using NOR Gates only]

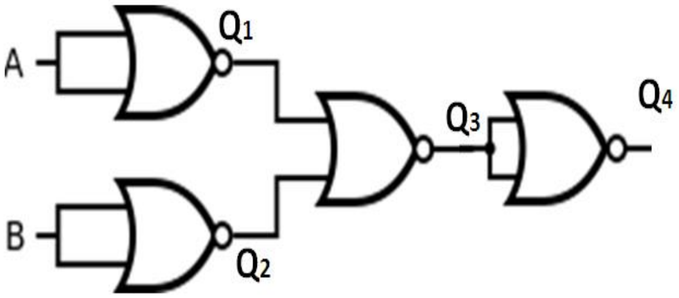


AND Gate by NOR Gate

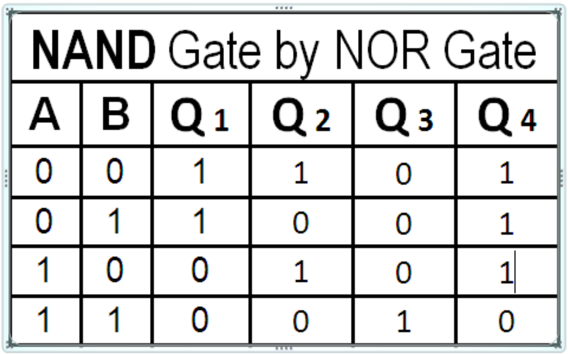
| A | B | Q 1 | Q 2 | Q 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

[Truth Table for NOR Gate as an **AND** Gate]

**d) NOR Gate as NAND Gate : -**



[**NAND** Gate by using NOR Gates only]

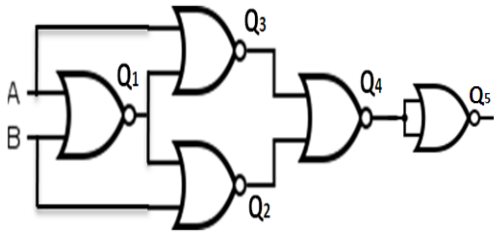


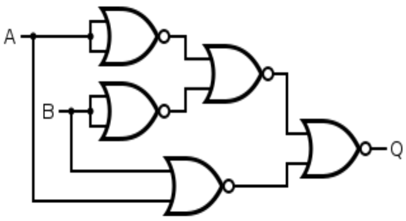
NAND Gate by NOR Gate

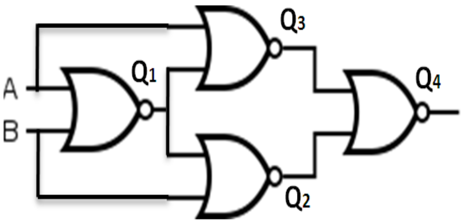
| A | B | Q 1 | Q 2 | Q 3 | Q 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

[Truth Table for NOR Gate as an **NAND** Gate]

**e) NOR Gate as EX-OR Gate : -** **NOR Gate as EX-NOR Gate : -**







[First Two Figures are for NOR Gate as EX-OR Gate and third Figure is for NOR Gate as EX-NOR Gate]

----ૐ---- ALL THE BEST ---🕮----ૐ----🕮--- ALL THE BEST ---ૐ----

# BOOLEAN ALGEBRA

♣ **Boolean Variable**: -

- A Variable having only two possible values, Such as 1/0, HIGH/LOW, ON/OFF or TRUE/FALSE.

♣ **Boolean Algebra : -**

- A System of algebra that operates on Boolean variables. The binary nature of Boolean algebra makes it useful for analysis, simplification and design of logic circuits.

♣ **Boolean Expression: -**

- An algebraic expression made up of Boolean variables and operators such as AND, OR and NOT.
- Boolean Expression is also called as **Boolean Function** or **Logic Function.**

## ❖ *INTRODUCTION : -*

- Switching circuits are also called logic circuits, gates circuits and digital circuits.
- Switching algebra is also called Boolean algebra. Boolean algebra is a system of mathematical logic.
- It is an algebraic system consisting of the set of elements (0, 1), two binary operators called OR and AND and unary operator called NOT.
- It is the basic mathematical tool in the analysis and synthesis of switching circuits. It is a way to express logic functions algebraically. Any complex logic can be expressed by a Boolean function.
- The Boolean algebra is governed by certain well developed rules and laws.
- Boolean algebra is different from both ordinary algebra and binary number system.
- In Boolean algebra **A + A = A** & **A . A = A,** because th variable has only a logical value either 1 or 0.
- In Boolean algebra 1 + 1 = 1 where as in Binary algebra 1 + 1 = 10 and in Ordinary algebra 1 + 1 = 2.
- There is nothing like Subtraction or Division. Also no negative or fractional value in Boolean algebra.
- In Boolean algebra, the multiplication and addition of the variables and functions are also only logical.
- Logical multiplication is same as the AND operation and logical addition is same as the OR operation.
- Here only two constants 0 and 1 found where as it can any numbers of constant in other algebra system.

## ❖ *AXIOMS: - [Accepts without Proof ]*

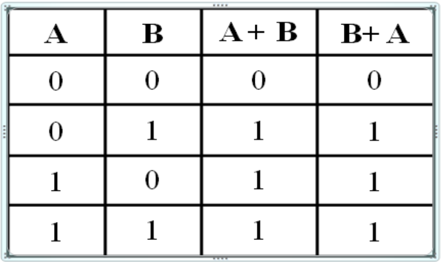
| NOT Operations | AND Operations | OR Operations |
|---|---|---|
| **Axiom 1** : $\overline{1} = 0$ | **Axiom 3**: 0. 0 = 0 | **Axiom 7**: 0 + 0 = 0 |
| | **Axiom 4**: 0. 1 = 0 | **Axiom 8**: 0 + 1 = 1 |
| **Axiom 2** : $\overline{0} = 1$ | **Axiom 5**: 1. 0 = 0 | **Axiom 9**: 1 + 0 = 1 |
| | **Axiom 6**: 1. 1 = 1 | **Axiom 10**: 1 + 1 = 1 |

## ❖ *LAWS OF BOOLEAN ALGEBRA: -*

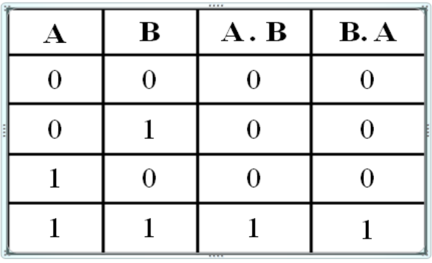
| Complementation LAWS | OR LAWS | AND LAWS |
|---|---|---|
| **Law 1:** $\overline{0} = 1$ | **Law 1:** A + 0 = A (Null law) | **Law 1:** A . 0 = 0 (Null law) |
| **Law 2:** $\overline{1} = 0$ | **Law 2:** A + 1 = 1 (Identity law) | **Law 2:** A . 1 = A (Identity law) |
| **Law 3:** If A = 0, then $\overline{A} = 1$ | **Law 3:** A + A = A | **Law 3:** A . A = A |
| **Law 4:** If A = 1, then $\overline{A} = 0$ | **Law 4:** $A + \overline{A} = 1$ | **Law 4:** $A . \overline{A} = 0$ |
| **Law 5:** $\overline{\overline{A}} = A$ | | |

♣ **COMMUTATIVE LAWS: -** Commutative laws allow change in position of AND or OR variables.

- There are two commutative laws. **Law 1:** A + B = B + A **&** **Law 2:** A . B = B . A
- **Proof by Truth Table : -**



| A | B | A + B | B+ A |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |



| A | B | A . B | B. A |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |

- This law can be extended to any number of variables. For example A.B. C = B. C. A = C. A. B = B. A. C

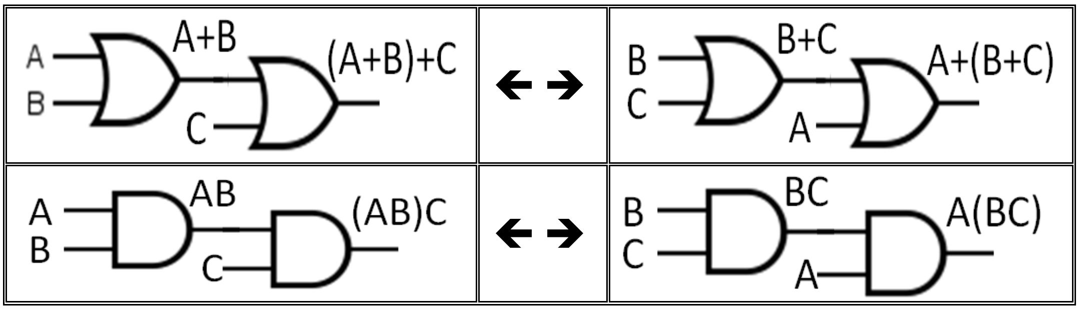
♣ **ASSOCIATIVE LAWS: -** The associative laws allow grouping of variables.

- There are two associative laws. **Law 1:** (A + B) + C = A + (B + C) **&** **Law 2:** (A . B) C = A (B . C)
- **Proof by Truth Table : -**

| A | B | C | A + B | (A + B) + C | B + C | A + (B + C) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

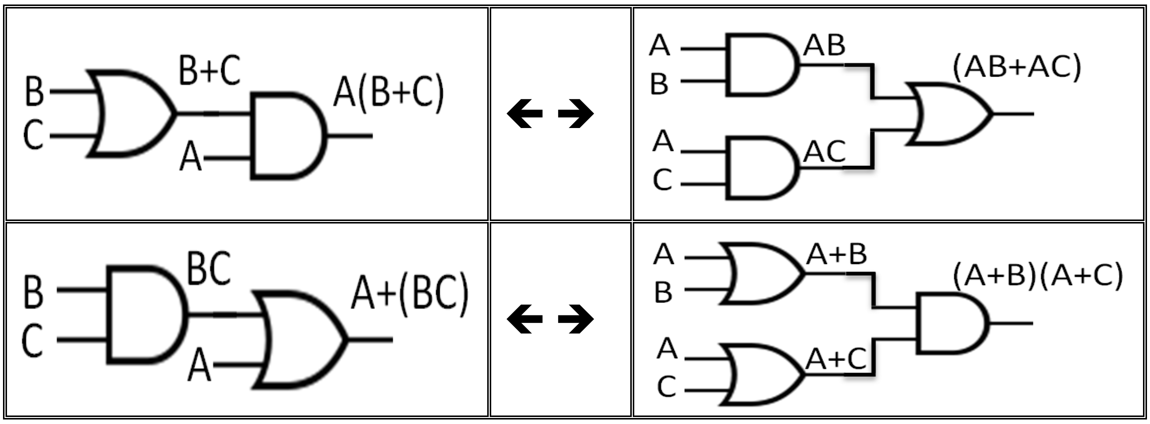
| A | B | C | AB | (AB) C | B.C | A (B.C) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- This law can be extended to any number of variables. For example: - A (BCD) = (ABC) D = (AB) (CD)



♣ **DISTRIBUTIVE LAWS: -**The distributive laws allow factoring or multiplying out of expressions.

There are two distributive laws. **Law 1:** A (B + C) = AB + AC **Law 2:** A + BC = (A + B) (A + C)



**Proof of 2nd Law: RHS** = (A + B) (A + C) = AA + AC + BA + BC = A + AC + AB + BC

= A (1+ C + B) + BC = A. 1 + BC = A + BC = **LHS**

- This law also given by : - **(I)** ABC (D + E) = ABCD + ABCE **(II)** AB (CD + EF) = ABCD + ABEF
- **Proof by Truth Table (Distributive Laws) : -**

| A | B | C | B+C | A(B+C) | AB | AC | AB+AC |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

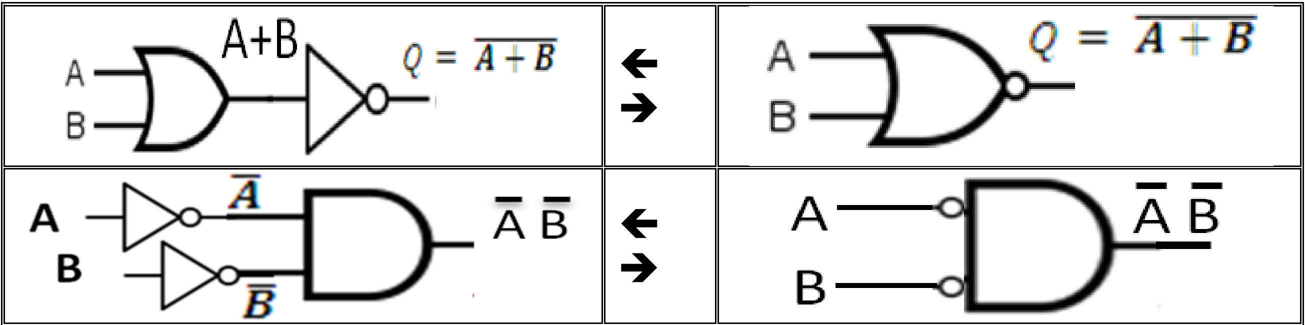
| A | B | C | BC | A+BC | A+B | A+C | (A+B)(A+C) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## ♣ DE MORGAN'S THEOREM:-

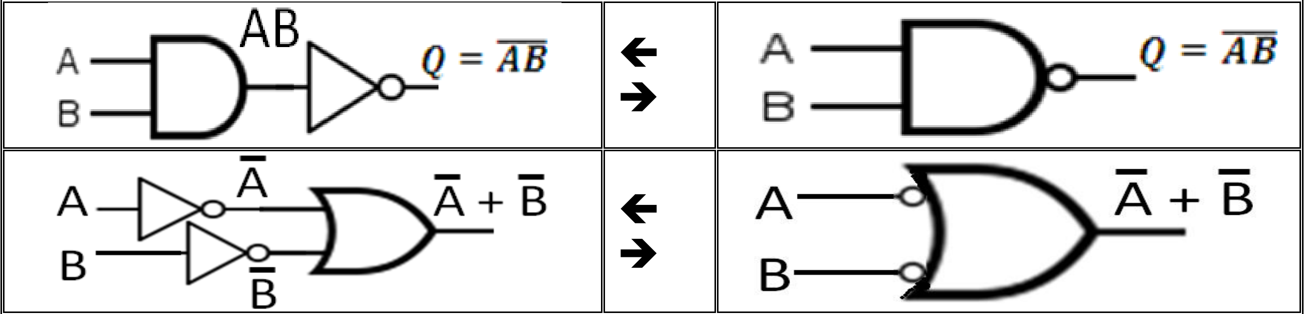
- De Morgan's theorem represents two of the most powerful laws in Boolean algebra.
- 1st **Law** States that the complement of a Sum of variables is equal to the Product of their Individual complements. That is → **Law 1:** $\overline{A+B} = \overline{A} \cdot \overline{B}$
- 2nd **Law** States that the complement of a Product of variables is equal to the Sum of their Individual complements. That is → **Law 2:** $\overline{A\,.\,B} = \overline{A} + \overline{B}$
- **Proof by Truth Table (Distributive Laws): -**

| A | B | A + B | $\overline{A+B}$ | $\overline{A}$ | $\overline{B}$ | $\overline{A}\cdot\overline{B}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | **1** | 1 | 1 | **1** |
| 0 | 1 | 1 | **0** | 1 | 0 | **0** |
| 1 | 0 | 1 | **0** | 0 | 1 | **0** |
| 1 | 1 | 1 | **0** | 0 | 0 | **0** |

| A | B | A . B | $\overline{A\,.\,B}$ | $\overline{A}$ | $\overline{B}$ | $\overline{A}+\overline{B}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | **1** | 1 | 1 | **1** |
| 0 | 1 | 0 | **1** | 1 | 0 | **1** |
| 1 | 0 | 0 | **1** | 0 | 1 | **1** |
| 1 | 1 | 1 | **0** | 0 | 0 | **0** |



**NOR Gate is Equivalent to Bubbled AND Gate**



**NAND Gate is Equivalent to Bubbled OR Gate**

♣ **Demorgan's law** can be extended any number of variables or combinations of variables. Such as: -

**(1)** $\overline{ABCD\ldots\ldots\ldots} = \overline{A} + \overline{B} + \overline{C} + \overline{D} + \ldots\ldots\ldots$ **(2)** $\overline{A+B+C+D\ldots\ldots\ldots} = \overline{A}\,.\,\overline{B}\,.\,\overline{C}\,.\,\overline{D}\,.\,\ldots\ldots\ldots$

**(3)** $\overline{(AB)(CD)(EFG)} = \overline{AB} + \overline{CD} + \overline{EFG}$ **(4)** $\overline{(A+B)(C+D)(E+F+G)} = \overline{A}\;\overline{B} + \overline{C}\,\overline{D} + \overline{E}\,\overline{F}\,\overline{G}$

## ♣ DUALITY:-

- The implication of the duality concept is that once a theorem or statement is proved, the dual also thus stand proved. This is called the principle of duality.
- In this method we can produce a dual identity by changing all ' + ' sign to ' * ' ; all ' * ' sign to ' + ' **and** complementing all 0s and 1s. The variables are *not complemented* in this process.

  **i. e.** $[f(A, B, C, \ldots\ldots, 0, 1, +, \cdot)]_d = f(A, B, C, \ldots., 1, 0, \cdot, +)$
- Relations between complement and dual

$f_c(A, B, C, \ldots.) = \overline{f(A, B, C, \ldots\ldots)} = f_d(A, B, C, \ldots)$ ←→ $f_d(A, B, C, \ldots..) = \overline{f(\overline{A}, \overline{B}, \overline{C}, \ldots\ldots)} = f_c(\overline{A}, \overline{B}, \overline{C}, \ldots.)$

- The first relation states that the complement of a function f (A, B, C, …) can be obtained by complementing all the variables in the dual function $f_d(A, B, C, \ldots..)$.
- The 2nd relation states that the dual can be obtained by complementing all the literals in $\overline{f(\overline{A}, \overline{B}, \overline{C}, \ldots\ldots)}$

## ❖ *EXAMPLES: -*

**Q1. De Morgan: -** $[\overline{\mathbf{AB + (A + B)}}]$

**Solution: -** $[\overline{AB + (A + B)}] = \overline{AB} \cdot \overline{(A + B)} = (\overline{A} + \overline{B})\ (\overline{A} \cdot \overline{B}) = \overline{A}\ .\ \overline{A} \cdot \overline{B} + \overline{B}.\overline{A} \cdot \overline{B} = \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{B} = \mathbf{\overline{A} \cdot \overline{B}}$

**Q2. Apply De Morgan's theorem to the given expression: -** $\overline{\mathbf{(A + \overline{B})(C + \overline{D})}}$

**Solution: -** $\overline{(A + \overline{B})(C + \overline{D})} = \overline{(A + \overline{B})} + \overline{(C + \overline{D})} = (\overline{A} \cdot \overline{\overline{B}}) + (\overline{C} \cdot \overline{\overline{D}}) = \overline{A}B + \overline{C}D$

**Q3. Reduce the expression, f =** $\overline{[\mathbf{\overline{AB} + \overline{A} + AB}]}$

**Solution: -** $\overline{[\overline{AB} + \overline{A} + AB]} = \overline{\overline{AB}} \cdot \overline{\overline{A}} \cdot \overline{AB} = AB \cdot A \cdot \overline{AB} = AB \cdot \overline{AB} = 0$

**Q4. Reduce the expression, f = A [B +** $\mathbf{\overline{C}\ \overline{(AB + A\overline{C})}}$**]**

**Solution: -** $A\ [B + \overline{C}\ \overline{(AB + A\overline{C})}] = A\ [B + \overline{C}\ (\overline{AB} \cdot \overline{A\overline{C}})] = A\ [B + \overline{C}\ (\overline{A} + \overline{B})\ (\overline{A} + C)]$

$= A\ [B + \overline{C}\ (\overline{A}\,\overline{A} + \overline{A}C + \overline{B}\,\overline{A} + \overline{B}C)] = A\ [B + \overline{C}\,\overline{A} + \overline{C}\,\overline{A}C + \overline{B}\,\overline{A}\overline{C} + \overline{C}\,\overline{B}C$ (As, $\overline{C}\,\overline{A}C = 0$, $\overline{C}\,\overline{B}C = 0$)

$= A\ [B + \overline{A}\,\overline{C} + \overline{B}\,\overline{A}\,\overline{C}] = AB + A\overline{A}\,\overline{C} + A\overline{B}\,\overline{A}\,\overline{C} = AB$ (As, $A\overline{A}\overline{C} = 0$, $A\overline{B}\,\overline{A}\,\overline{C} = 0$)

**Q5. Reduce the expression, f = A + B [AC + (B +** $\mathbf{\overline{C}}$**) D]**

**Solution: -** $A + B\ [AC + (B + \overline{C})\ D] = A + B\ (AC + BD + \overline{C}D) = A + ABC + BD + B\overline{C}D$

$= A(1 + BC) + BD\ (1 + \overline{C}) = A \cdot 1 + BD \cdot 1 = A + BD$

**Q6. Reduce the expression, f =** $\mathbf{\overline{(A + \overline{BC})}\ (A\overline{B} + ABC)}$

**Solution: -** $\overline{(A + \overline{BC})}\ (A\overline{B} + ABC) = (\overline{A} \cdot \overline{\overline{BC}})\ (A\overline{B} + ABC) = \overline{A}BC \cdot A\overline{B} + \overline{A}BC \cdot ABC = 0 + 0 = 0$

**Q7. Reduce the expression, f = (B + BC) (B +** $\mathbf{\overline{B}C}$**) (B + D)**

**Solution: -** $(B + BC)\ (B + \overline{B}C)\ (B + D) = (BB + B\overline{B}C + B \cdot BC + BC \cdot \overline{B}C)\ (B + D)$ [As, $B\overline{B}C = 0$, $BC \cdot \overline{B}C = 0$]

$= (B + BC)\ (B + D) = B(1 + C)\ (B + D) = B \cdot 1 + B + D = B(1 + D) = B \cdot 1 = B$

**Q8. Show that** $\mathbf{AB + A\overline{B}C + B\overline{C} = AC + B\overline{C}}$

**Solution: -**L.H.S: - $AB + A\overline{B}C + B\overline{C} = A(B + \overline{B}C) + B\overline{C} = A(B + \overline{B})\ (B + C) + B\overline{C} = AB + AC + B\overline{C}$

$= AB(C + \overline{C}) + AC + B\overline{C} = ABC + AB\overline{C} + AC + B\overline{C} = AC(1 + B) + B\overline{C}(1 + A) = AC + B\overline{C} =$ **R.H.S**

**Q9. Show that** $\mathbf{A\overline{B}C + B + B\overline{D} + AB\overline{D} + \overline{A}C = B + C}$

**Solution: -**L.H.S: - $A\overline{B}C + B + B\overline{D} + AB\overline{D} + \overline{A}C = A\overline{B}C + \overline{A}C + B + B\overline{D} + AB\overline{D}$

$= C(A\overline{B} + \overline{A}) + B(1 + \overline{D} + A\overline{D}) = C(\overline{A} + A)(\overline{A} + \overline{B}) + B = C(\overline{A} + \overline{B}) + B = C\overline{A} + C\overline{B} + B$ (As, $\overline{A} + A = 1$)

$= (B + C)\ (B + \overline{B}) + C\overline{A} = (B + C) + C\overline{A} = B + C(1 + \overline{A}) = B + C \cdot 1 = B + C =$ **R.H.S** (Proved)

## **SIMPLIFY FOLLOWING BOOLEAN EXPRESSION TO A MINIMUM NUMBER OF LITERALS : -**

- $\mathbf{\overline{X}\,\overline{Y} + XY + \overline{X}Y} = \overline{X}\,\overline{Y} + Y(X + \overline{X}) = \overline{X}\,\overline{Y} + Y = (\overline{X} + Y)\ (\overline{Y} + Y) = \overline{X} + Y$
- $\mathbf{(X + Y)\ (X + \overline{Y})} = XX + X\overline{Y} + XY + Y\overline{Y} = X + X\overline{Y} + XY + 0 = X\ (1 + \overline{Y} + Y) = X\ .\ 1 = X$
- $\mathbf{\overline{A}\,\overline{C} + ABC + A\overline{C}} = \overline{C}\ (\overline{A} + A) + ABC = (\overline{C} + ABC) = (\overline{C} + C)\ (\overline{C} + AB) = \overline{C} + AB$
- $\mathbf{\overline{A}B(\overline{D} + \overline{C}D) + B(A + \overline{A}CD)} = \overline{A}B\overline{D} + \overline{A}B\overline{C}D + AB + \overline{A}BCD = \overline{A}BD + \overline{A}B\overline{D} + AB$

  $= \overline{A}B\ (D + \overline{D}) + AB = \overline{A}B + AB = B\ (\overline{A} + A) = B$
- $\mathbf{(\overline{A} + C)\ (\overline{A} + \overline{C})\ (\overline{A} + B + \overline{C}\ D)} = (\overline{A}\ .\ \overline{A} + \overline{A}\ .\ \overline{C} + \overline{A}\ .\ C + \overline{C}\ .\ C)\ (\overline{A} + B + \overline{C}D)$

  $= \overline{A}\ (1 + \overline{C} + C)\ (\overline{A} + B + \overline{C}D) = \overline{A} + \overline{A}B + \overline{A}\overline{C}D = \overline{A}\ (1 + B + \overline{C}D) = \overline{A}$
- $\mathbf{AB + A\ (B + C) + \overline{B}\ (B + D)} = AB + AB + AC + \overline{B}B + \overline{B}D = AB + AC + \overline{B}D = A\ (B + C) + \overline{B}D$
- $\mathbf{A + B + \overline{A}\,\overline{B}C} = (A + \overline{A})\ (A + \overline{B}C) + B = A + \overline{B}C + B = A + (B + \overline{B})\ (B + C) = A + B + C$
- $\mathbf{ABEF + AB\ \overline{EF} + \overline{AB}\ EF} = AB\ (EF + \overline{EF}) + \overline{AB}\ EF = AB + \overline{AB}\ EF = (AB + \overline{AB})\ (AB + EF)$
- $\mathbf{ABC\overline{D} + A + AB\overline{D} + \overline{D}(\overline{A}\,\overline{B}\,\overline{C})} = ABC\overline{D} + A + AB\overline{D} + \overline{D}\,\overline{A}\,\overline{B}\,\overline{C} = A\ (BC\overline{D} + 1 + B\overline{D}) + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

  $= A + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = (A + \overline{A})\ (\overline{A} + \overline{B}\,\overline{C}\,\overline{D}) = A + \overline{B}\,\overline{C}\,\overline{D}$
- $\mathbf{X\ [Y + Z\ \overline{(XY + XZ)}]} = X\ [Y + Z\ (\overline{XY}\ .\ \overline{XZ})] = XY + XZ\ .\ \overline{XY}\ .\ \overline{XZ} = XY + 0 = XY$
- $\mathbf{\overline{X}\,\overline{Z} + \overline{Y}\,\overline{Z} + Y\overline{Z} + XYZ} = \overline{X}\,\overline{Z} + \overline{Z}\ (\overline{Y} + Y) + XYZ = \overline{X}\,\overline{Z} + \overline{Z} + XYZ = \overline{Z}\ (\overline{X} + 1) + XYZ$

  $= \overline{Z} + XYZ = (\overline{Z} + XY)\ (\overline{Z} + Z) = \overline{Z} + XY$

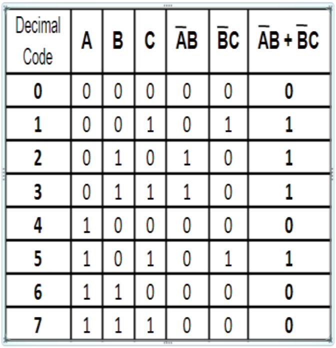
## ❖ *BOOLEAN EXPRESSIONS & THEIR REPRESENTATIONS : -*

- There are different ways of representing a given function in following ways:-

♣ **Sum - Of - Products (SOP) Form: -** This form is also called the Disjunctive Normal Form (DNF). For Example: - $f\ (A, B, C) = \overline{A}B + \overline{B}C$

♣ **Product- Of – Sums (POS) Form: -** This form is also called the Conjunctive Normal Form (CNF). For Example: - $f\ (A, B, C) = (\overline{A} + \overline{B}) + (B + C)$

♣ **Truth Table Form : -** [For example $f\ (A, B, C) = \overline{A}B + \overline{B}C$]



| Decimal Code | A | B | C | $\overline{A}B$ | $\overline{B}C$ | $\overline{A}B + \overline{B}C$ |
|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | **0** |
| **1** | 0 | 0 | 1 | 0 | 1 | **1** |
| **2** | 0 | 1 | 0 | 1 | 0 | **1** |
| **3** | 0 | 1 | 1 | 1 | 0 | **1** |
| **4** | 1 | 0 | 0 | 0 | 0 | **0** |
| **5** | 1 | 0 | 1 | 0 | 1 | **1** |
| **6** | 1 | 1 | 0 | 0 | 0 | **0** |
| **7** | 1 | 1 | 1 | 0 | 0 | **0** |

## ♣ STANDARD SUM - OF - PRODUCTS (SOP) FORM: -

- It is also called the *Disjunctive Canonical Form* (DCF) or *Expanded Sum of Products* or *Canonical Sum of Products* Form.
- In this form, the function is the sum of a number of products terms where each product term contains all the variables of the function either in complemented or uncomplimentary form.
- This can also be derived from the truth table by finding the sum of all the terms that corresponds to those combinations for which the function 'f ' assumes the value 1.
- It can also be obtained from the SOP form algebraically as shown below;

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

- A product term which contains all the variables of the functions either in complemented or un complement form is called a **MINTERM**.
- The Sum of the minterms whose value is equal to 1 is the standard sum of product form of the function.
- The minterm is denoted as $m_0$, $m_1$, $m_2$ …. Where the suffixes are the decimal codes of the combinations.
- An 'n' variable function can have $2^n$ numbers of minterms. So, for a 3-variable function the minterms are: - $\mathbf{m_0} = \bar{A}\bar{B}\bar{C}$, $\mathbf{m_1} = \bar{A}\bar{B}C$, $\mathbf{m_2} = \bar{A}B\bar{C}$, $\mathbf{m_3} = \bar{A}BC$, $\mathbf{m_4} = A\overline{BC}$, $\mathbf{m_5} = A\bar{B}C$, $\mathbf{m_6} = AB\bar{C}$, $\mathbf{m_7} = ABC$.
- Another way of representing the function in canonical SOP form is by showing the sum of minterms for which the function equals to 1. Thus, $\mathbf{f\,(A,B,C) = m_1 + m_2 + m_3 + m_5}$
- Another way of representing the function is by listing the decimal codes of the minterms for which f = 1. Thus, **f (A, B, C) =∑m (1, 2, 3, 5).** Where, **∑m** represents the sum of all the minterms whose decimal codes are given in the parenthesis.

## ♣ STANDARD PRODUCT- OF – SUMS (POS) FORM: -

- This form is also called as Conjunctive Canonical Form (CCF) or Expanded Product-Of–Sums Form or Canonical Product-Of-Sums Form.
- It is derived by considering the combinations for which f = 0, each term is a sum of all the variables.
- A variable is written in uncomplimentary form if it has a value of '0' in the combination and appears in complemented form if it has a value of "1" in the combination.
- For example the sum corresponding to 4th row in the above table (3 = 011) is $(A + \bar{B} + \bar{C})$.
- It can also be obtained from the POS [Normal form] form algebraically as shown below: -
- Thus the function $f(A, B, C) = (\bar{A} + \bar{B})\,(A + B)$ is given by the product of sum can converted to standard or canonical form as follows, $f\,(A, B, C) = (\bar{A} + \bar{B} + C \cdot \bar{C}) + (\,A + B + C \cdot \bar{C})$

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

- A sum term which contains all the variables in either complemented or uncomplimentary form is called a **MAXTERM.**
- A Maxterm assumes the value " 0 " only for one combination of the variables. For other it will " 1 ".
- The product of Maxterm corresponding to the rows for which, f = 0 if the Standard Product- Of - Sums form of the function.
- Maxterm are represented as $M_0$, $M_1$, $M_2$….where the suffixes are the decimal codes of the combinations.
- An 'n' variable function can have $2^n$ numbers of Maxterms. So, for a 3-variable function the Maxterms are: - $\mathbf{M_0} = A + B + C$, $\mathbf{M_1} = A + B + \bar{C}$, $\mathbf{M_2} = A + \bar{B} + C$, $\mathbf{M_3} = A + \bar{B} + \bar{C}$,
  $\mathbf{M_4} = \bar{A} + B + C$, $\mathbf{M_5} = \bar{A} + B + \bar{C}$, $\mathbf{M_6} = \bar{A} + \bar{B} + C$, $\mathbf{M_7} = \bar{A} + \bar{B} + \bar{C}$.
- Thus CCF of a function 'f' is written as, $\mathbf{f\,(A, B, C) = M_0 \cdot M_4 \cdot M_6 \cdot M_7}$ or **f (A, B, C) = ΠM (0, 4, 6, 7)**. Where, Π represents product of all the Maxterm whose decimal codes are given within the parenthesis.

## ♣ CONVERSION BETWEEN CANONICAL FORM: -

- The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.

  For Example: - f (A, B, C) = ∑m (0, 2, 4, 6, 7); So, $\overline{f\,(A, B, C)}$ = ∑m (1, 3, 5) = $m_1 + m_3 + m_5$
- Now if we take the complement of " $\bar{f}$ " by DE Morgan's theorem, we obtain " f " in a different form,

  $f = \overline{m_1 + m_3 + m_5} = \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_5} = \overline{M_1} \cdot \overline{M_3} \cdot \overline{M_5} = \pi M\,(1, 3, 5)$
- In fact $\overline{m_j} = M_j$. Thus, the **Maxterm** with subscript **j** is a complement of the **minterm** with the same subscript **j** and vice versa.
- To convert one canonical form to another, inter change the symbol ∑ **& Π** and list out those numbers missing from the original form.

## ❖ *KARNAUGH MAP OR K- MAP:-*

- The K- map is a chart or a graph, composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.
- Like a truth table K-map shows the relation between Inputs and desired outputs.
- Since a K-map is a graphical representation of Boolean expression.
- A two variable K-map will have $2^2$ = 4 cells, similarly a three variable K-map will have $2^3$ cells and a four variable K-map will have $2^4$ = 16 number of cells.

| A \ B | 0 | 1 |
|---|---|---|
| 0 | $0_{0}$ | $1_{1}$ |
| 1 | $2_{2}$ | $3_{3}$ |

- A K-map can used for solving problems involving any number of variables but it becomes complicated when we will go for five or more variables.
- Usually these limited to 6 variables.
- Any Boolean expression can expression can be expressed in a standard or canonical form or Expended product (AND) of sum (OR) i.e. in the POS form.
- Any given function which is not in the standard form, can always be converted to standard form by un reducing i.e. expanding the function.
- A standard SOP form can always be converted to a standard POS, by treating the missing minterm of the SOP form as the Maxterm & the POS form.

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

- Similarly a standard POS form can always be converted to a standard SOP form, by treating the missing Maxterm of the POS form as the minterm of the corresponding SOP form.
- The possible minterm & Maxterm of a **2-Variable** function f (A,B) are,

$m_0 = \overline{A}\overline{B}$, $m_1 = \overline{A}B$, $m_2 = A\overline{B}$, $m_3 = AB$;

$M_0 = (A + B)$, $M_1 = A + \overline{B}$, $M_2 = \overline{A} + B$, $M_3 = \overline{A} + \overline{B}$.

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

- The possible minterms of a **3-Variable** function f (A,B,C) are :-

$m_0 = \overline{A}\overline{B}\overline{C}$, $m_1 = \overline{A}\overline{B}C$, $m_2 = \overline{A}B\overline{C}$, $m_3 = \overline{A}BC$,

$m_4 = A\overline{B}\overline{C}$, $m_5 = A\overline{B}C$, $m_6 = AB\overline{C}$, $m_7 = ABC$;

- The possible Maxterms of a **3-Variable** function f (A,B,C) are :-

$M_0 = A+B+C$, $M_1 = A+ B+\overline{C}$, $M_2 = A+\overline{B}+C$, $M_3 = A + \overline{B} + \overline{C}$,

$M_4 = \overline{A}+B+C$, $M_5 = \overline{A}+B+\overline{C}$, $M_6 = \overline{A} + \overline{B} +C$, $M_7 = \overline{A} + \overline{B} + \overline{C}$

- The possible minterm & Maxterm of a **4-Variable** function f (A, B, C, D) are shown in figure: -

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 4 | 12 | 8 |
| 01 | 1 | 5 | 13 | 9 |
| 11 | 3 | 7 | 15 | 11 |
| 10 | 2 | 6 | 14 | 10 |

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0<br>$A+B+C+D$<br>$(M_0)$ | 1<br>$A+B+C+\overline{D}$<br>$(M_1)$ | 3<br>$A+B+\overline{C}+\overline{D}$<br>$(M_3)$ | 2<br>$A+B+\overline{C}+D$<br>$(M_2)$ |
| 01 | 4<br>$A+\overline{B}+C+D$<br>$(M_4)$ | 5<br>$A+\overline{B}+C+\overline{D}$<br>$(M_5)$ | 7<br>$A+\overline{B}+\overline{C}+\overline{D}$<br>$(M_7)$ | 6<br>$A+\overline{B}+\overline{C}+D$<br>$(M_6)$ |
| 11 | 12<br>$\overline{A}+\overline{B}+C+D$<br>$(M_{12})$ | 13<br>$\overline{A}+\overline{B}+C+\overline{D}$<br>$(M_{13})$ | 15<br>$\overline{A}+\overline{B}+\overline{C}+\overline{D}$<br>$(M_{15})$ | 14<br>$\overline{A}+\overline{B}+\overline{C}+D$<br>$(M_{14})$ |
| 10 | 8<br>$\overline{A}+B+C+D$<br>$(M_8)$ | 9<br>$\overline{A}+B+C+\overline{D}$<br>$(M_9)$ | 11<br>$\overline{A}+B+\overline{C}+\overline{D}$<br>$(M_{11})$ | 10<br>$\overline{A}+B+\overline{C}+D$<br>$(M_{10})$ |

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0<br>$\overline{A}\overline{B}\overline{C}\overline{D}$<br>$(m_0)$ | 1<br>$\overline{A}\overline{B}\overline{C}D$<br>$(m_1)$ | 3<br>$\overline{A}\overline{B}CD$<br>$(m_3)$ | 2<br>$\overline{A}\overline{B}C\overline{D}$<br>$(m_2)$ |
| 01 | 4<br>$\overline{A}B\overline{C}\overline{D}$<br>$(m_4)$ | 5<br>$\overline{A}B\overline{C}D$<br>$(m_5)$ | 7<br>$\overline{A}BCD$<br>$(m_7)$ | 6<br>$\overline{A}BC\overline{D}$<br>$(m_6)$ |
| 11 | 12<br>$AB\overline{C}\overline{D}$<br>$(m_{12})$ | 13<br>$AB\overline{C}D$<br>$(m_{13})$ | 15<br>$ABCD$<br>$(m_{15})$ | 14<br>$ABC\overline{D}$<br>$(m_{14})$ |
| 10 | 8<br>$A\overline{B}\overline{C}\overline{D}$<br>$(m_8)$ | 9<br>$A\overline{B}\overline{C}D$<br>$(m_9)$ | 11<br>$A\overline{B}CD$<br>$(m_{11})$ | 10<br>$A\overline{B}C\overline{D}$<br>$(m_{10})$ |

## ❖ *TWO VARIABLE K- MAP:-*

- A two variable K-map has $2^2$ = 4 squares.
- Each square on the K-map represents a unique minterm.
- A 1 placed in any square indicates that corresponding minterm is included in output expression and 0 entries indicates that does not appear in the expression for output.
- A two variable expression can have $2^2$ = 4 possible combinations of the input variables A and B.

| A \ B | 0 | 1 |
|---|---|---|
| 0 | $\overline{A}\overline{B}_{0}$ | $\overline{A}B_{1}$ |
| 1 | $A\overline{B}_{2}$ | $AB_{3}$ |

- **Q1. Map the expression ∑m (0, 2, 3) in a 2-variable K-map.**
- **Q2. Map the expression f = $\overline{A}$B + A$\overline{B}$**
- **Solution: -** The given expression in minterm is, F = $m_1 + m_2$ = ∑m (1, 2). So the K-map for this expression will be, →

| A \ B | 0 | 1 |
|---|---|---|
| 0 | $0_{0}$ | $1_{1}$ |
| 1 | $1_{2}$ | $0_{3}$ |

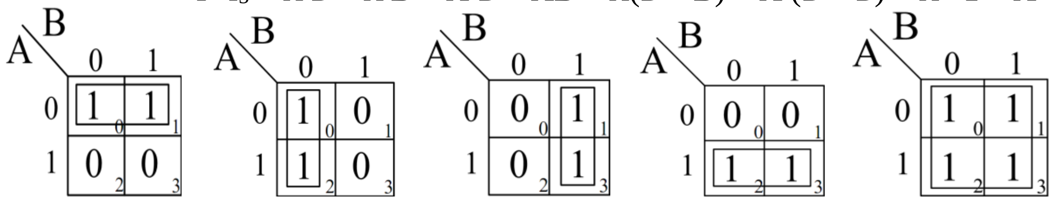
| A \ B | 0 | 1 |
|---|---|---|
| 0 | $1_{0}$ | $0_{1}$ |
| 1 | $1_{2}$ | $1_{3}$ |

## ❖ MINIMIZATION OF SOP EXPRESSION IN 2-VARIABLE K-MAP : -

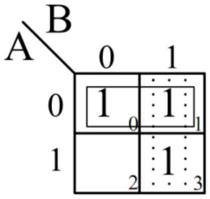
- To minimize a Boolean expression given in the SOP from by using the K-map, we have to look for adjacent squares having …….., i.e. minterm adjacent to each other are combined to form larger squares to eliminate some variables.

- For example in 2-variable K-map $m_0$ ($\bar{A}\bar{B}$) & $m_1$ ($\bar{A}B$) differ only in variable B ($\bar{A}$ is common to both of them). So they may be combined to form a 2-square to eliminate the variable B. **Similarly**, minterms $m_0$ ($\bar{A}\bar{B}$) & $m_2$ ($A\bar{B}$) ; $m_1$ ($\bar{A}B$) & $m_3$ (AB) ; $m_2$ ($A\bar{B}$) & $m_3$ (AB) are adjacent to each other.
- However, minterms $m_0$ ($\bar{A}\bar{B}$) & $m_3$ (AB) ; $m_1$ ($\bar{A}B$) & $m_2$ ($A\bar{B}$) are not adjacent to each other, because they differ in more than one variables.
- A minterm can be combined with any number of minterm adjacent to it to form larger squares.
- This eliminates one variable which is not common to both.
- For example, **$m_0$ & $m_1$** can be combined to yield. ➔ **$f_1$** $= m_0 + m_1 = \bar{A}\,\bar{B} + \bar{A}\,B = \bar{A}(\bar{B} + B) = \bar{A} - 1 = \bar{A}$ ,
- **$m_0$ & $m_2$** can be combined to yield. ➔ $f_2 = m_0 + m_2 = \bar{A}\,\bar{B} + A\bar{B} = \bar{B}(\bar{A} + A) = \bar{B}$
- **$m_1$ & $m_3$** can be combined to yield. ➔ $f_3 = m_1 + m_3 = \bar{A}\,B + AB = B(\bar{A} + A) = B$
- **$m_2$ & $m_3$** can be combined to yield. ➔ $f_4 = m_2 + m_3 = A\,\bar{B} + AB = A(\bar{B} + B) = A$
- **$m_0$, $m_1$, $m_2$, $m_3$** can be combined to form,

➔ $f_5 = \bar{A}\,\bar{B} + \bar{A}\,B + A\,\bar{B} + AB = \bar{A}(\bar{B} + B) + A\,(\bar{B} + \bar{B}) = \bar{A}\cdot 1 + A\cdot 1 = \bar{A} = A = 1$

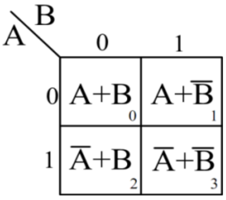


Ans➔



- **Q1. Reduce the expression f = $\bar{A}\bar{B}$ + $\bar{A}$B + AB using mapping.**
- The expression in minterm is $f = m_0 + m_1 + m_3 = \sum m\ (0, 1, 3)$ So mapping will be

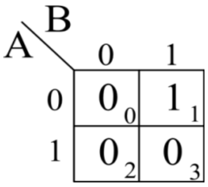
  In one 2-square combination it is $\bar{A}$ and other is B ➔ **f = $\bar{A}$ + B**



❖ **<u>MAPPING OF POS EXPRESSION</u>: -**

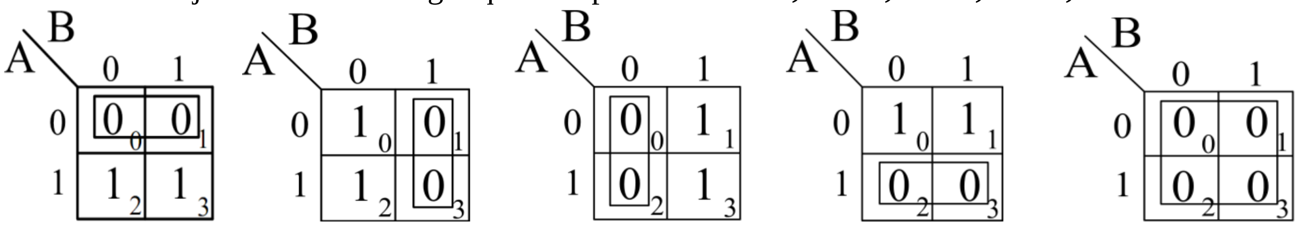
- The sum term in the standard POS expression is called a **Maxterm**.
- For mapping a POS expression on to K-map, 0s are placed in the squares corresponding to the Maxterms which are present in the expression and 1s are placed for the Maxterms which are not present in the expression.
- **Q1. Plot the expression f = (A + B) ($\bar{A}$ + B) ($\bar{A}$ + $\bar{B}$) on the K-map.**

The expression in Maxterm, f = ΠM (0, 2, 3) so it can map as

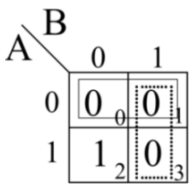


❖ **<u>MINIMIZATION OF SOP EXPRESSION</u>: -**

- To obtain the minimal expression in POS form, map the given POS expression on to the K-map and combine the adjacent 0s into a large square as possible. **$f_1$ = A, $f_2$ = $\bar{B}$, $f_3$ = B, $f_4$ = $\bar{A}$, $f_5$ = 0**



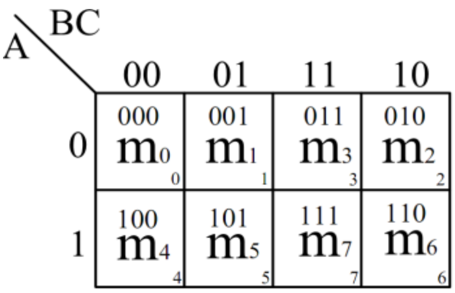
- **Q1. Reduce the expression f = (A + B) (A + $\bar{B}$) ($\bar{A}$ + $\bar{B}$) using mapping.**
- The given expression in Maxterm is f = Π M (0, 1, 3). So its mapping is shown as fig.
- So the corresponding function will be A$\bar{B}$.
- This realization is same as that of SOP expression i.e. $\sum m_2 = A\bar{B}$).

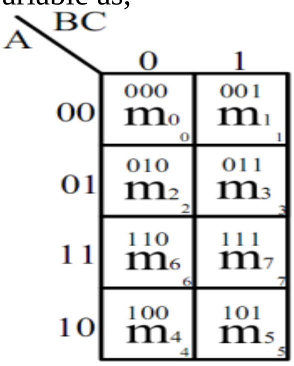


❖ ***<u>THREE VARIABLE K- MAP:-</u>***

- A 3-Variable (A, B, C) expression can have $2^3$ = 8 possible combinations of I/P variable as,

| Binary | Minterms | Expression | Maxterms | Expression |
|---|---|---|---|---|
| 000 | $m_0$ | $\bar{A}\,\bar{B}\,\bar{C}$ | $M_0$ | $A + B + C$ |
| 001 | $m_1$ | $\bar{A}\,\bar{B}\,C$ | $M_1$ | $A + B + \bar{C}$ |
| 010 | $m_2$ | $\bar{A}\,B\,\bar{C}$ | $M_2$ | $A + \bar{B} + C$ |
| 011 | $m_3$ | $\bar{A}\,B\,C$ | $M_3$ | $A + \bar{B} + \bar{C}$ |
| 100 | $m_4$ | $A\,\bar{B}\,\bar{C}$ | $M_4$ | $\bar{A} + B + C$ |
| 101 | $m_5$ | $A\,\bar{B}\,C$ | $M_5$ | $\bar{A} + B + \bar{C}$ |
| 110 | $m_6$ | $A\,B\,\bar{C}$ | $M_6$ | $\bar{A} + \bar{B} + C$ |
| 111 | $m_7$ | $A\,B\,C$ | $M_7$ | $\bar{A} + \bar{B} + \bar{C}$ |





- **Q1. Map the expression $f = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,B\,\overline{C} + A\,B\,C$**
- **Solution: -**From the expression minterm is, f = 001 + 101 + 010 + 110 + 111= $m_1 + m_5 + m_2 + m_6 + m_7$. So the given expression will be f = ∑ m (1, 2, 5, 6, 7).
- So the corresponding K-map will be →

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $0_0$ | $1_1$ | $0_3$ | $1_2$ |
| 1 | $0_4$ | $1_5$ | $1_7$ | $1_6$ |

- **Q2. Map the expression $f = (A+B+C)\,(\overline{A}+B+\overline{C})\,(\overline{A}+\overline{B}+\overline{C})\,(A+\overline{B}+\overline{C})\,(\overline{A}+\overline{B}+C)$**
- **Solution: -**The given expression in Maxterm are: -000= $M_0$, 101= $M_5$, 111= $M_7$, 011 = $M_3$, 110 = $M_6$ ➔ f = ΠM (0, 3, 5, 6, 7). So its mapping will be →

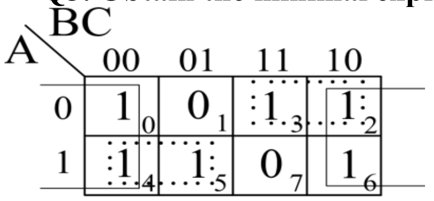
| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $0_0$ | $1_1$ | $0_3$ | $1_2$ |
| 1 | $1_4$ | $0_5$ | $0_7$ | $0_6$ |

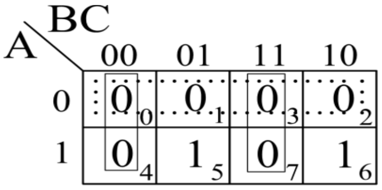
## ❖ *MINIMIZATION OF SOP AND POS EXPRESSIONS:-*

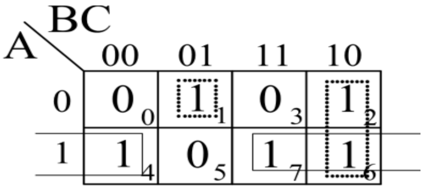
♣ **Generalized procedure to simplify the Boolean expression by K-map**

- Plot the K-map and place 1s (0s) corresponding to the minterms (Maxterms) of the SOP (POS) expression. Check the K-map for 1s (0s) which are not adjacent to any other 1(0).
- They are isolated minterms. They can't be combined even into 2-Square.
- Check for those 1s (0s) which are adjacent to only one other 1(0) and make them pairs (2-Square).
- Check for quads (4-squares) and octets (8-Squares) of adjacent 1s (0s) even if they contain some 1s (0s) which have already been combined.
- They must geometrically form a square or a rectangle. They can't be diagonal conform.
- Check for any 1s (0s) that have not been combined yet and combine them into bigger square if possible.
- Form the minimal expression by summing (Multiplying) the product (sum) terms of all the groups.

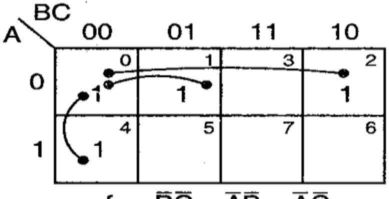
**Q1. Reduce the expression f = ∑m (0, 2, 3, 4, 5, 6) using 3-variable K=map.**
**Q2. Reduce the expression f = ΠM (0, 1, 2, 3, 4, 7) by K=map.**
**Q3. Obtain the minimal expression for f = ∑m (1, 2, 4, 6, 7)**



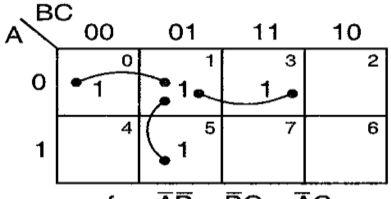




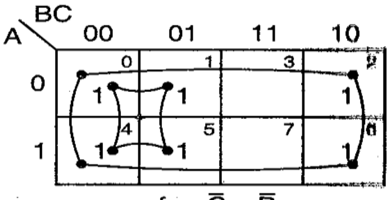
- **Solution**: - **(1)** $f_{min} = \overline{C} + A\,\overline{B} + \overline{A}\,B$ **(2)** $f_{max} = (B + C)\,(\overline{B} + \overline{C})\,(\overline{A})$ **(3)** $f_{min} = \overline{A}\,\overline{B}\,C + A\,\overline{C} + AB + B\,\overline{C}$
- These possible combinations are also for POS but 1s are replaced by 0s.



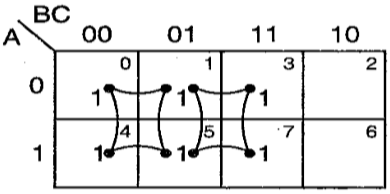
$f_1 = \overline{B}\,\overline{C} + \overline{A}\,\overline{B} + \overline{A}\,\overline{C}$



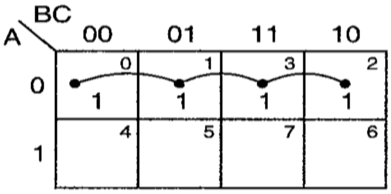
$f_2 = \overline{A}\,\overline{B} + \overline{B}C + \overline{A}C$



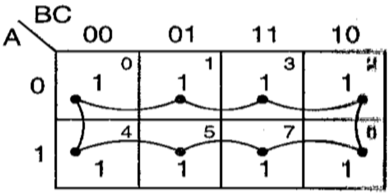
$f_3 = \overline{C} + \overline{B}$



$f_4 = \overline{B} + C$



$f_5 = \overline{A}$

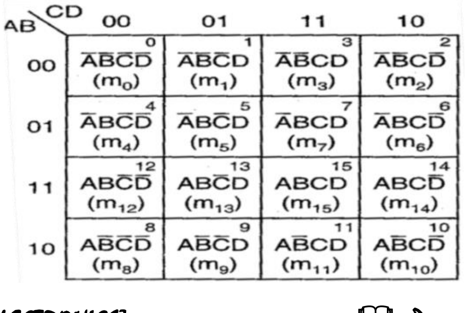


$f_6 = 1$

## ❖ *FOUR VARIABLE K-MAP : -*

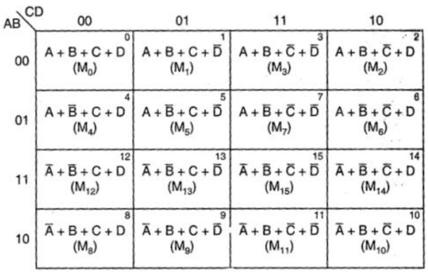
- A four variable (A, B, C, D) expression can have $2^4$ = 16 possible combinations of input variables.
- A four variable K-map has $2^4$ = 16 squares or cells and each square on the map represents either a minterm or a Maxterm as shown in the figure below.

**[SOP FORM]** →



| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0<br>$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ ($m_0$) | 1<br>$\overline{A}\,\overline{B}\,\overline{C}D$ ($m_1$) | 3<br>$\overline{A}\,\overline{B}CD$ ($m_3$) | 2<br>$\overline{A}\,\overline{B}C\overline{D}$ ($m_2$) |
| 01 | 4<br>$\overline{A}B\overline{C}\,\overline{D}$ ($m_4$) | 5<br>$\overline{A}B\overline{C}D$ ($m_5$) | 7<br>$\overline{A}BCD$ ($m_7$) | 6<br>$\overline{A}BC\overline{D}$ ($m_6$) |
| 11 | 12<br>$AB\overline{C}\,\overline{D}$ ($m_{12}$) | 13<br>$AB\overline{C}D$ ($m_{13}$) | 15<br>$ABCD$ ($m_{15}$) | 14<br>$ABC\overline{D}$ ($m_{14}$) |
| 10 | 8<br>$A\overline{B}\,\overline{C}\,\overline{D}$ ($m_8$) | 9<br>$A\overline{B}\,\overline{C}D$ ($m_9$) | 11<br>$A\overline{B}CD$ ($m_{11}$) | 10<br>$A\overline{B}C\overline{D}$ ($m_{10}$) |

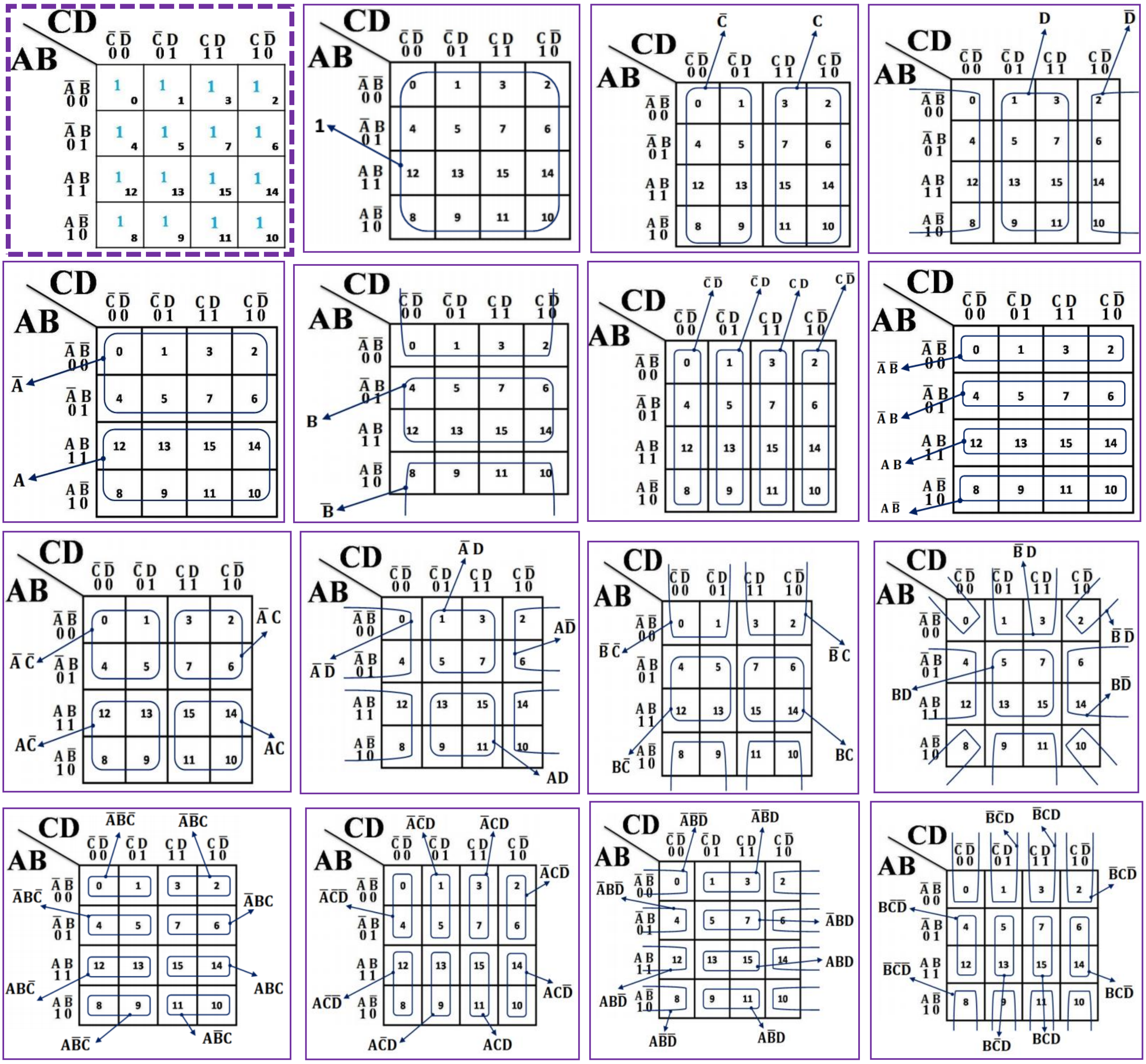
**[POS FORM]**➔



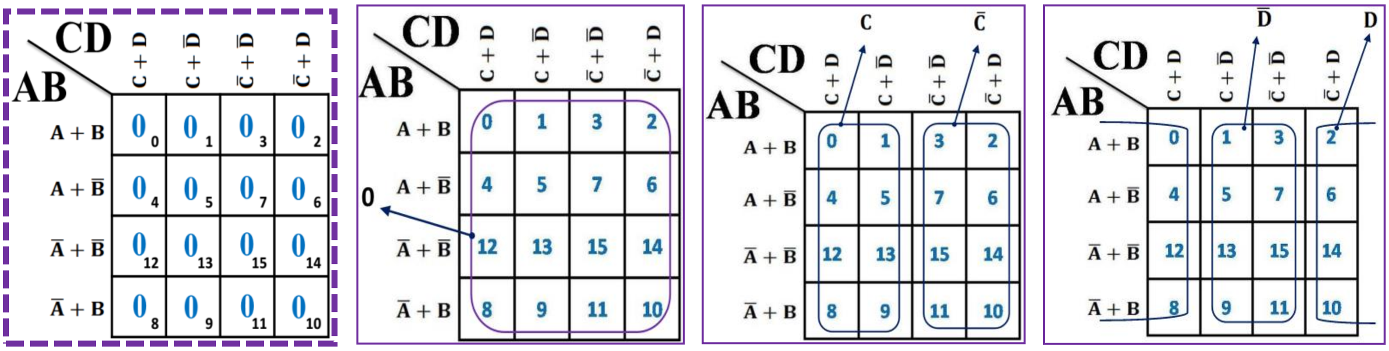
| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0<br>$A+B+C+D$ ($M_0$) | 1<br>$A+B+C+\overline{D}$ ($M_1$) | 3<br>$A+B+\overline{C}+\overline{D}$ ($M_3$) | 2<br>$A+B+\overline{C}+D$ ($M_2$) |
| 01 | 4<br>$A+\overline{B}+C+D$ ($M_4$) | 5<br>$A+\overline{B}+C+\overline{D}$ ($M_5$) | 7<br>$A+\overline{B}+\overline{C}+\overline{D}$ ($M_7$) | 6<br>$A+\overline{B}+\overline{C}+D$ ($M_6$) |
| 11 | 12<br>$\overline{A}+\overline{B}+C+D$ ($M_{12}$) | 13<br>$\overline{A}+\overline{B}+C+\overline{D}$ ($M_{13}$) | 15<br>$\overline{A}+\overline{B}+\overline{C}+\overline{D}$ ($M_{15}$) | 14<br>$\overline{A}+\overline{B}+\overline{C}+D$ ($M_{14}$) |
| 10 | 8<br>$\overline{A}+B+C+D$ ($M_8$) | 9<br>$\overline{A}+B+C+\overline{D}$ ($M_9$) | 11<br>$\overline{A}+B+\overline{C}+\overline{D}$ ($M_{11}$) | 10<br>$\overline{A}+B+\overline{C}+D$ ($M_{10}$) |

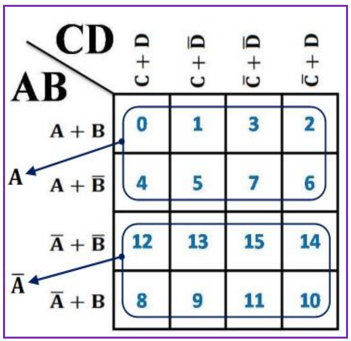
- The binary number designations of the rows and columns are in the gray code.
- The binary numbers along the top of the map indicate the conditions of C and D along any column and binary numbers along left side indicate the conditions of A and B along any row.
- The numbers in top right corners of the squares indicate the minterm or Maxterm designations as usual.
- Mapping and minimization procedures are same as that of three – variable K-map.

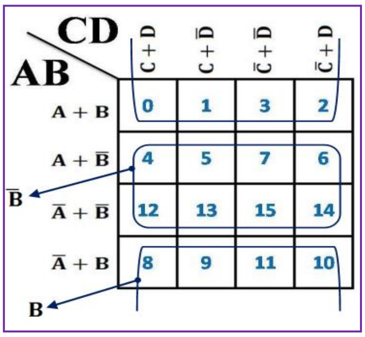
## ❖ DIFFERENT CASES OF 4 VARIABLE K–MAP [SOP]

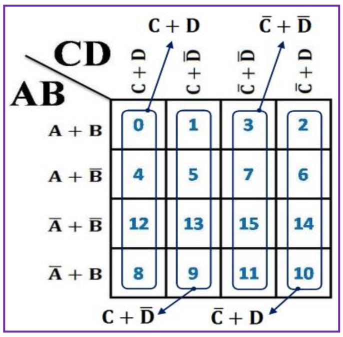


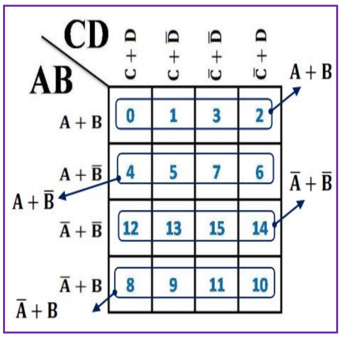
## ❖ DIFFERENT CASES OF 4 VARIABLE K–MAP [POS]

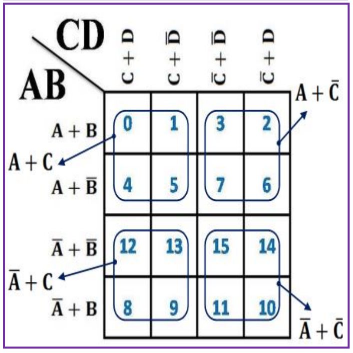


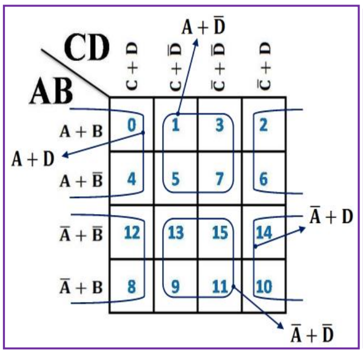


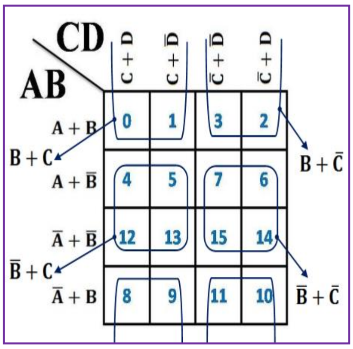


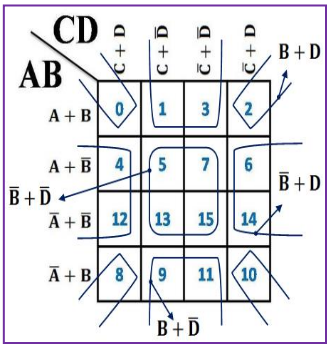


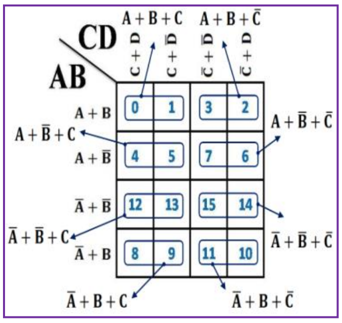


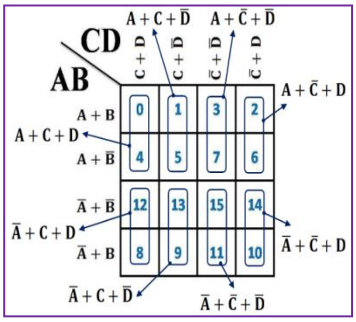


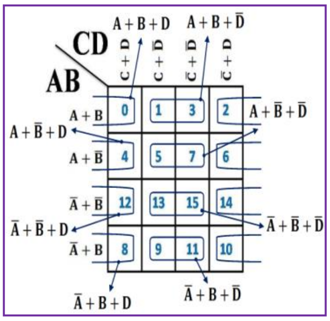


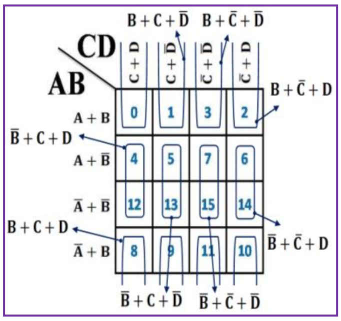




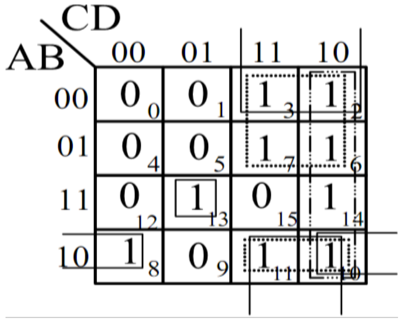




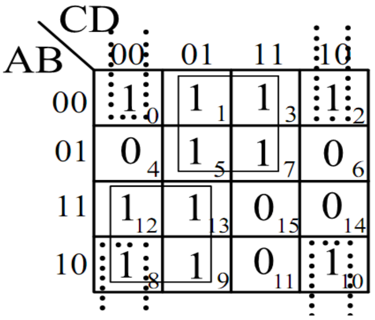




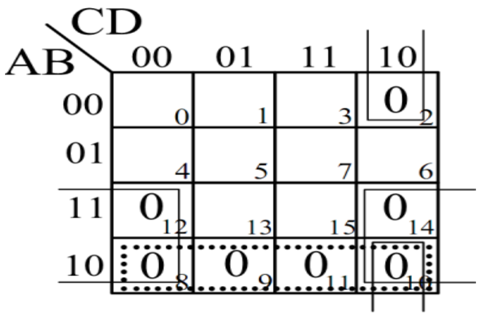
- **Q1. Reduce the expression f = ∑ m (2, 3, 6, 7, 8, 10, 11, 13, 14) by 4-variable K-map method.**
- **Q2. Reduce using mapping the expression f (A, B, C, D) = ∑ m (0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13).**
- **Q3. Reduce using mapping the expression f (A, B, C, D) = ∏ M (2, 8, 9, 10, 11, 12, 14).**



| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $0_{0}$ | $0_{1}$ | $1_{3}$ | $1_{2}$ |
| 01 | $0_{4}$ | $0_{5}$ | $1_{7}$ | $1_{6}$ |
| 11 | $0_{12}$ | $1_{13}$ | $0_{15}$ | $1_{14}$ |
| 10 | $1_{8}$ | $0_{9}$ | $1_{11}$ | $1_{10}$ |



| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $1_{0}$ | $1_{1}$ | $1_{3}$ | $1_{2}$ |
| 01 | $0_{4}$ | $1_{5}$ | $1_{7}$ | $0_{6}$ |
| 11 | $1_{12}$ | $1_{13}$ | $0_{15}$ | $0_{14}$ |
| 10 | $1_{8}$ | $1_{9}$ | $0_{11}$ | $1_{10}$ |

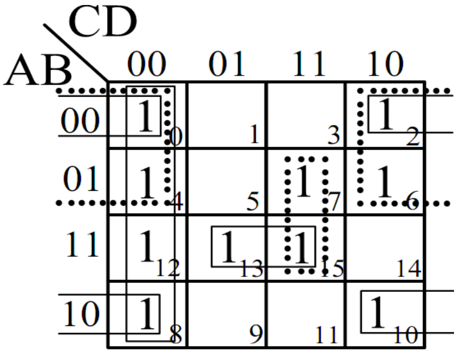


| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $_{0}$ | $_{1}$ | $_{3}$ | $0_{2}$ |
| 01 | $_{4}$ | $_{5}$ | $_{7}$ | $_{6}$ |
| 11 | $0_{12}$ | $_{13}$ | $_{15}$ | $0_{14}$ |
| 10 | $0_{8}$ | $0_{9}$ | $0_{11}$ | $0_{10}$ |

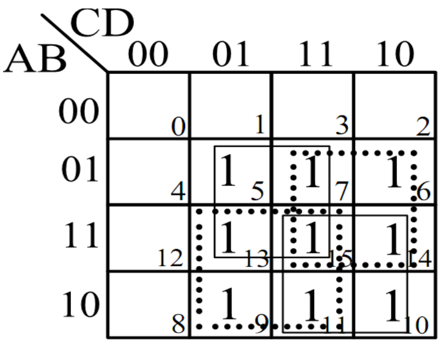
➢ **Solution**: - **(1) f$_{\min}$** = $A\,B\,\bar{C}\,D + A\,\bar{B}\,\bar{D} + \bar{B}\,C + \bar{A}\,C + C\,\bar{D}$ **(2) f$_{\min}$** = $\bar{B}\,\bar{D} + A\,\bar{C} + \bar{A}\,D$

**(3) f$_{\max}$** = $(\bar{A} + B)\,(\bar{A} + D)\,(B + \bar{C} + D)$

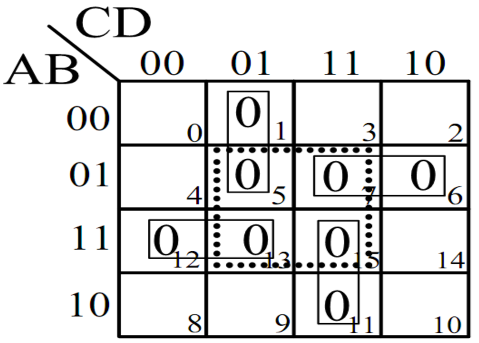
- **Q4. Reduce using mapping the expression f (A, B, C, D) = ∑m (0, 2, 4, 6, 7, 8, 10, 12, 13, 15).**
- **Q5. Reduce the expression using K-map f (A, B, C, D) = ∑m (5, 6, 7, 9, 10, 11, 13, 14, 15).**
- **Q6. Reduce the expression by K-map f (A, B, C, D) = ∏ M (1, 5, 6, 7, 11, 12, 13, 15).**



| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $1_{0}$ | $_{1}$ | $_{3}$ | $1_{2}$ |
| 01 | $1_{4}$ | $_{5}$ | $1_{7}$ | $1_{6}$ |
| 11 | $1_{12}$ | $1_{13}$ | $1_{15}$ | $_{14}$ |
| 10 | $1_{8}$ | $_{9}$ | $_{11}$ | $1_{10}$ |



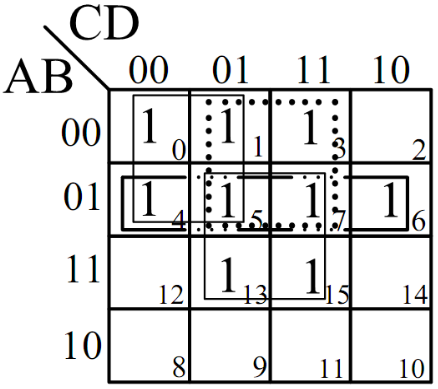
| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $_{0}$ | $_{1}$ | $_{3}$ | $_{2}$ |
| 01 | $_{4}$ | $1_{5}$ | $1_{7}$ | $1_{6}$ |
| 11 | $_{12}$ | $1_{13}$ | $1_{15}$ | $1_{14}$ |
| 10 | $_{8}$ | $1_{9}$ | $1_{11}$ | $1_{10}$ |

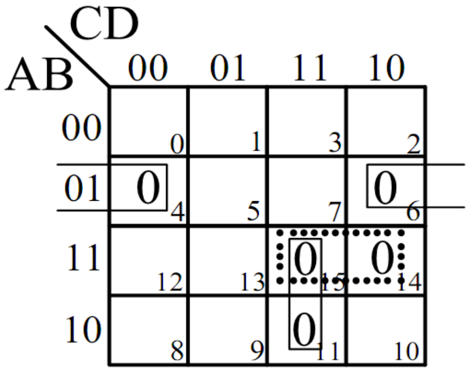


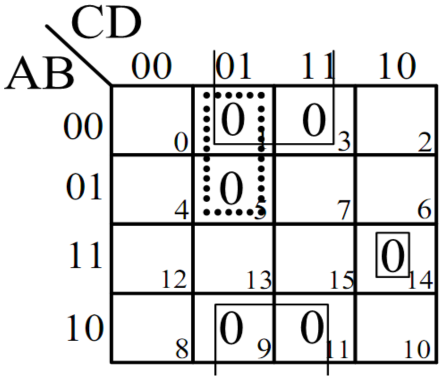
| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $_{0}$ | $0_{1}$ | $_{3}$ | $_{2}$ |
| 01 | $_{4}$ | $0_{5}$ | $0_{7}$ | $0_{6}$ |
| 11 | $0_{12}$ | $0_{13}$ | $0_{15}$ | $_{14}$ |
| 10 | $_{8}$ | $_{9}$ | $0_{11}$ | $_{10}$ |

- **Solution**: - **(4) f$_{min}$** = $\bar{B}\,\bar{D} + \bar{C}\,\bar{D} + \bar{A}D + ABD + BCD$ **(5) f$_{min}$** = $BD + BC + AD + AC$
  **(6) f$_{max}$** = $(\bar{A} + \bar{B} + C)\,(A + C + \bar{D})\,(A + \bar{B} + \bar{C})\,(\bar{A} + \bar{C} + \bar{D})$

- **Q7. Reduce the given expression in the SOP is f (A, B, C, D) = ∑ m (0, 1, 3, 4, 5, 6, 7, 13, 15).**
- **Q8. Reduce the given expression in the POS is f (A, B, C, D) = ∏ M (4, 6, 11, 14, 15).**
- **Q9. Reduce the given expression in the POS is f (A, B, C, D) = ∏ M (1, 3, 5, 9, 11, 14).**

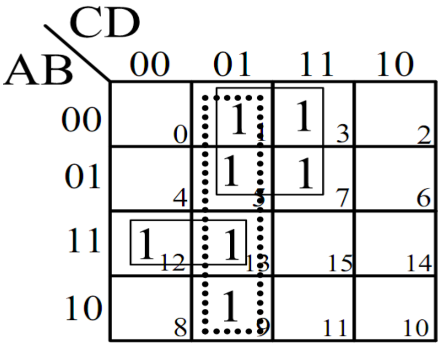


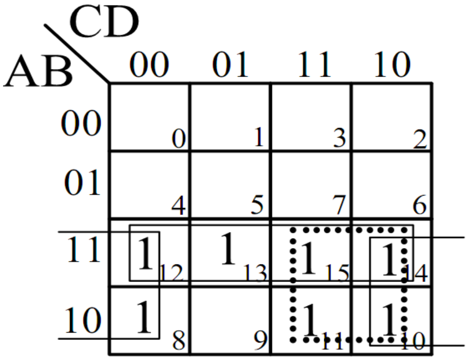


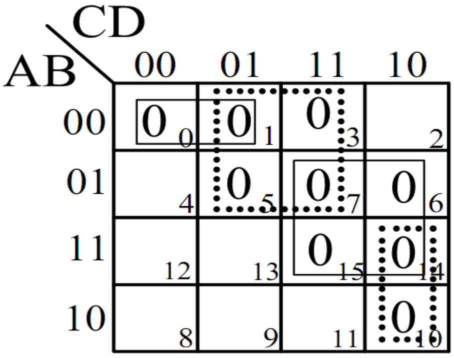


**Solution**: - **(7) f$_{min}$** = $\bar{A}\,\bar{C} + \bar{A}D + \bar{A}B + BD$ **(8) f$_{max}$** = $(A + \bar{B} + D)\,(\bar{A} + \bar{C} + \bar{D})\,(\bar{A} + \bar{B} + \bar{C})$
**(9) f$_{max}$** = $(B + \bar{D})\,(A + C + \bar{D})\,(\bar{A} + \bar{B} + \bar{C} + D)$

- **Q10. Reduce the given expression in the SOP is f (A, B, C, D) = ∑ m (1, 3, 5, 7, 9, 12, 13).**
- **Q11. Reduce the given expression in the SOP is f (A, B, C, D) = ∑ m (8, 10, 11, 12, 13, 14, 15).**
- **Q12. Reduce the given expression in the POS is f (A, B, C, D) = ∏ M (0, 1, 3, 5, 6, 7, 10, 14, 15).**





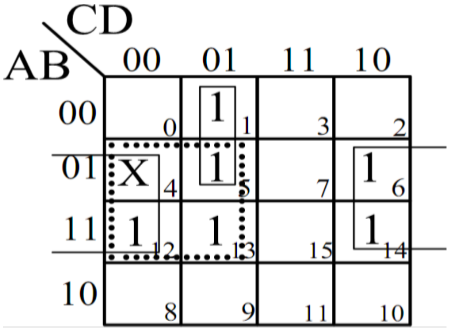


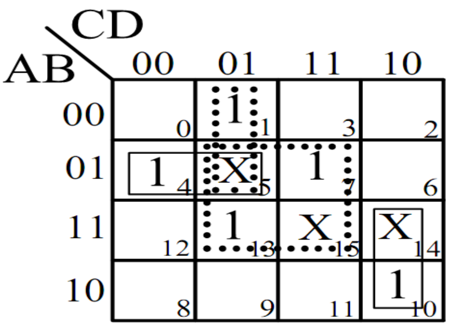
**Sol**:- **(10) f$_{min}$** = $\bar{A}D + \bar{C}D + AB\bar{C}$ **(11) f$_{min}$** = $AB + AC + A\bar{D}$ **(12) f$_{max}$** = $(\bar{A} + \bar{C} + D)(\bar{B} + \bar{D})(A + \bar{D})\,(A+B+C)$

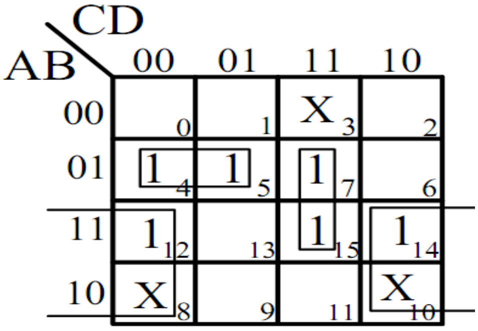
## ❖ *DON'T CARE COMBINATIONS*: -

- So far, the expression considered have been completely specified for every combination of the input variables, i.e. each minterm (Maxterm) has been specified as 1 or 0.
- If it often occurs that for certain input combinations, the value of the output is unspecified either because the input combinations are invalid or no value.
- The combinations for which the values of the expressions are not specified are called **Don't Care / Optional Combinations**. The don't care term are denoted by **d, x or φ.**
- During the process of design using an SOP map, each don't care is treated as 1 if it helpful in map reduction; otherwise it is treated as 0 or left alone.
- During the process of design using an POS map, each don't care is treated as 0 if it is helpful in map reduction; otherwise it is treated as 1 and left alone.
- A standard SOP expression with don't cares can be converted into standard POS form by keeping all don't cares as it is & the missing minterms of the SOP form are written as the Maxterms for POS form.
- Similarly, to convert a standard POS expression with don't cares can be converted into standard SOP form by keeping the don't cares as they are, and the missing Maxterms of the POS form are written as the minterms of the SOP form.

- **Q1. Reduce the given expression, f = ∑m (1, 5, 6, 12, 13, 14) + ∑ d (2, 4) using K- map.**
- **Q2. Minimize the expression using K-map, f (A, B, C, D) = ∑m (1, 4, 7, 10, 13) + ∑d (5, 14, 15).**
- **Q3. Minimize the expression, f (A, B, C, D) =∑m (4, 5, 7, 12, 14, 15) + ∑d (3, 8, 10) using K-map.**

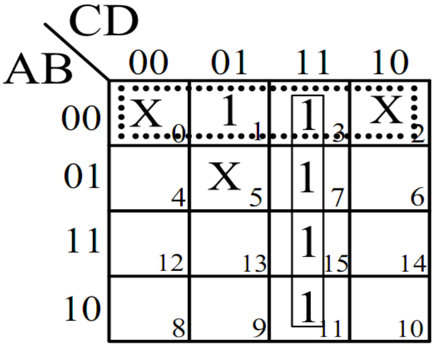


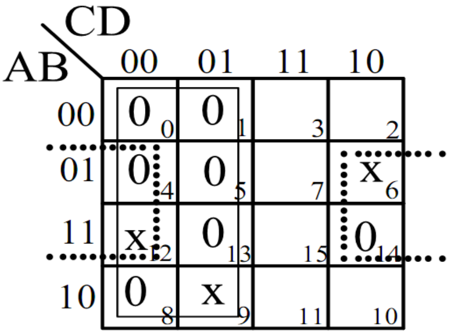


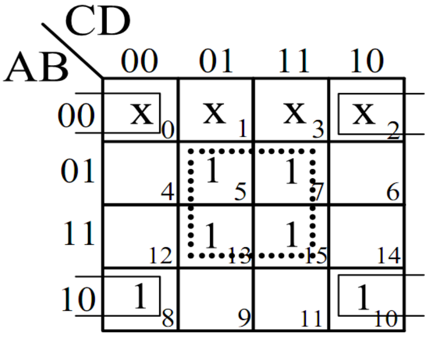


**Solution**: - **(1) f$_{min}$** = $B\bar{D} + B\bar{C} + \bar{A}\,\bar{C}\,D$ **(2) f$_{min}$** = $BD + \bar{A}B\bar{C} + \bar{A}\bar{C}D + AC\bar{D}$ **(3) f$_{min}$** = $A\bar{D} + \bar{A}B\bar{C} + BCD$

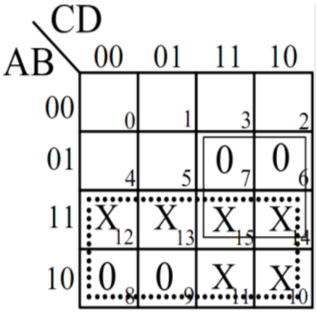
- **Q4. Minimize the expression using K-map: - f (A, B, C, D) = ∑m (1, 3, 7, 11, 15) + ∑d (0, 2, 5)**
- **Q5. Minimize the expression: - f (A, B, C, D) = ∏M (0, 1, 4, 5, 8, 13, 14) . ∏d (6, 9,12) using K-map**
- **Q6. Minimize the expression using K-map: - f (A, B, C, D) = ∑m (5, 7, 8, 10, 13, 15) + ∑d (0, 1, 2, 3)**







**Solution**: - **(4) f$_{min}$** = $\bar{A}\,\bar{B} + CD$ **(5) f$_{max}$** = $C\,(\,\bar{B} + D)$ **(6) f$_{min}$** = $(BD + \bar{B}\,\bar{D})$

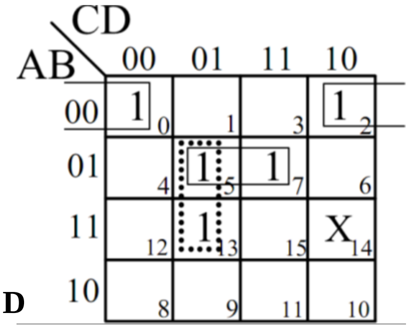


- **Q7. Reduce expression in POS form: -**
  **f (A, B, C, D) = ∑m (0, 1, 2, 3, 4, 5) + ∑d (10, 11, 12, 13, 14, 15)**

- **Solution: -** The given expression is in SOP form. They can be written in POS form by treating the missing minterm of the SOP form as the Maxterms of the POS form. So the expression in POS form,
- f (A, B, C, D)=ΠM (6, 7, 8, 9). Πd (10, 11, 12, 13, 14, 15) ➔**f = $\bar{A}$ ($\bar{B}$ + $\bar{C}$)**

- **Q8. Obtain the simplified expression using K-map.**

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



- **Solution: -** Given, f = $\bar{A}\,\bar{B}\,\bar{D}$ + $\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$+ $\bar{A}$ B D + A B $\bar{C}$ D
  = 00X0 + 0000 + 01X1 + 1101

So the minterms are, 0000, 0010, 0000, 0101, 0111, 1101

➔ Hence f = ∑m (0, 2, 5, 7, 13)

- So, mapping into k-map & reduced values is➔**f = $\bar{A}$ $\bar{B}$ $\bar{D}$ + $\bar{A}$ BD + B$\bar{C}$D**

- **Q9. Obtain the simplified expression using K-map.**

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

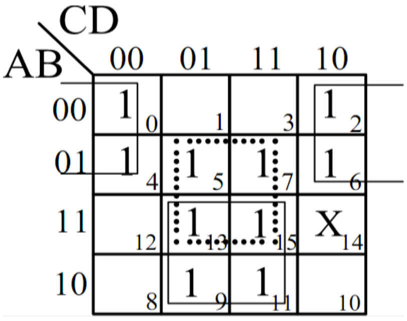
**Solution: -** From given expression, **f = A B D + $\bar{A}$ $\bar{C}$ $\bar{D}$ + $\bar{A}$ B + $\bar{A}$ C $\bar{D}$ + $\bar{A}$ $\bar{B}$ D**

f = 11X1 + 0X00 + 01XX + 0X10 + 10X1

So the minterm are given by,

1101, 1111, 0000, 0100, 0100, 0101, 0110, 0111, 0010, 0110, 1001, 1011 So, f = ∑m (13, 15, 0, 4, 4, 5, 6, 7, 2, 6, 9, 11)

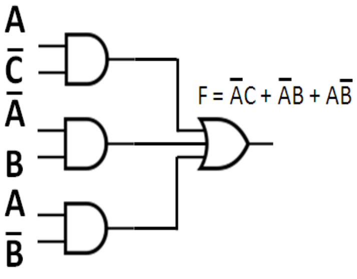
➔ After rearranging, we get **f** = ∑m (0, 2, 4, 5, 6, 7, 9, 11, 13, 15)



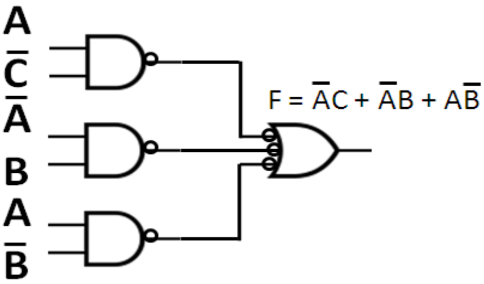
- So, mapping into k-map & reduced values is ➔ **f = $\bar{A}$ $\bar{D}$ + BD + AD**

## *REALIZATION OF SIMPLIFIED LOGIC EXPRESSIONS USING GATES* : -

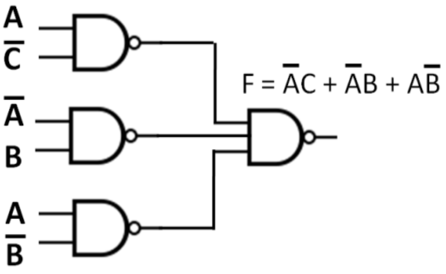
- **f = $\bar{A}C + \bar{A}B + A\bar{B}$. Realize by using NAND Gates Only.**



{By Fundamental Gates}

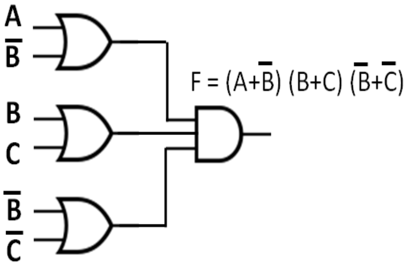


{By Bubbled Gates <Bubbled OR ≡ NAND>}

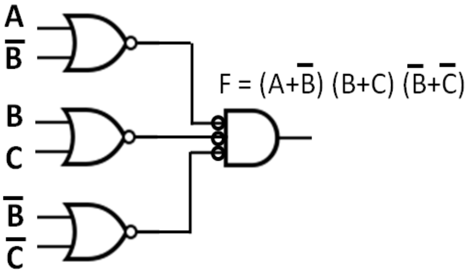


{By NAND Gates Only}

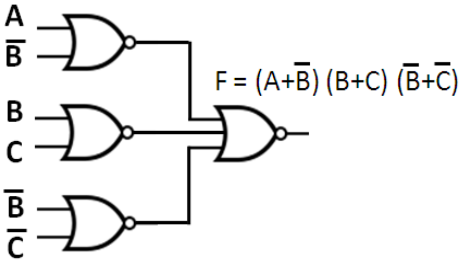
- **f = (A + $\bar{B}$) (B + C) ($\bar{B}$ + $\bar{C}$). Realize by using NOR Gates Only.**



{By Fundamental Gates}

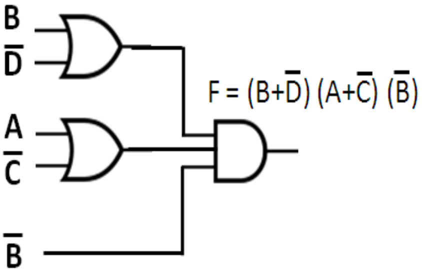


{By Bubbled Gates <Bubbled AND ≡ NOR>}

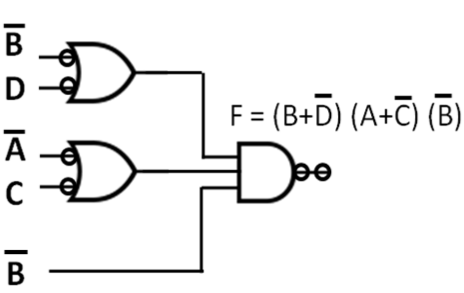


{By NOR Gates Only}

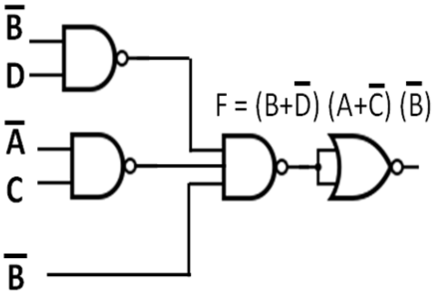
- **f = (B + $\bar{D}$) (A + $\bar{C}$ ) ($\bar{B}$) Realize by using NAND Gates Only.**



{By Fundamental Gates}

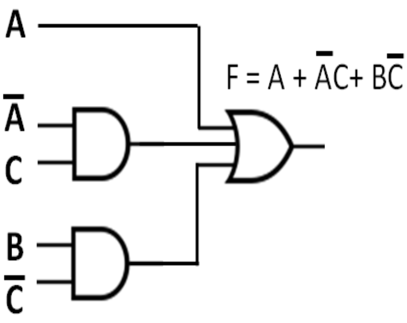


{By Bubbled Gates <Bubbled OR ≡ NAND>}

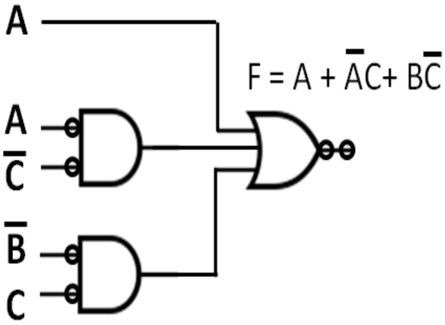


{By NAND Gates Only}

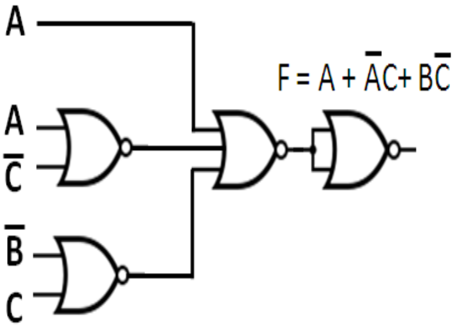
- **f = A + $\bar{A}C + B\bar{C}$. Realize by using NOR Gates Only.**



{By Fundamental Gates}



{By Bubbled Gates <Bubbled AND ≡ NOR>}



{By NOR Gates Only}

## SOME MORE QUESTIONS FOR PRACTICE: -

**1.** Convert the following to minterm,

**(A)** $A + \overline{B}\,\overline{C}$ **(B)** $\overline{A} + B + CA$ **(C)** $ABC + AB + DC + \overline{D}$

**2.** Convert the following to Maxterm,

**(A)** $A (B + \overline{C})$ **(B)** $(A + \overline{B}) (\overline{A} + D)$ **(C)** $(A + B + \overline{D}) (\overline{A} + C + D) (\overline{A} + \overline{D})$ **(d)** $A (\overline{A} + B) \overline{C}$

**3.** Reduce the following by K-map, **[A]** $AB + A\overline{B}C + \overline{A}B\overline{C} + B\overline{C}$ **[B]** $AB\overline{C}+AB+B\overline{C}+D\overline{B}$ **[C]** $AB + A\overline{C}+ AD + A\overline{B}C + ABC$

**4.** Reduce the following by K-map, **[A]** $(A+B)(A+\overline{B}+C)(A+\overline{C})$ **[B]** $A(B+\overline{C})(A+\overline{B})(B+C+\overline{D})$ **[C]** $(\overline{A}+B)(A+B+\overline{D})(B+\overline{C})(B+C+D)$

**5.** Obtain minimal POS expression for ∏M(0, 1, 2, 4, 5, 6, 9, 11, 12, 13, 14, 15) & implement in NOR Gate

**6.** Reduce ∏M (1, 2, 3, 5, 6, 7, 8, 9, 12, 13) and implement it in universal Logic

**7.** Reduce the following four Variable expressions by K-map and implement them in universal Logic: -

**(a)** F (a, b, c, d) =Σ m (0, 1, 2, 3, 8, 9, 10, 11, 13, 15)
**(b)** F (a, b, c, d) = Σ m (0, 2, 4, 6, 7, 8, 10, 12, 13 15)
**(c)** F (a, b, c, d) = ∑ m (3,4,6,7,11,12,13,14)
**(d)** F (a, b, c, d) = ∑ m(1,5,7,8,9,10,11,14,15)
**(e)** F (a, b, c, d) = Σ m (2, 3, 5, 7, 9, 11, 12, 13, 14, 15)
**(f)** F (a, b, c, d) = Σ m (4, 5, 6, 12, 14, 15) + *Σ*d (3, 8, 10)
**(g)** F (a, b, c, d) = ∑m (4, 7, 12, 15) + *Σ*d (0, 3, 8, 11)
**(h)** F (a,b,c,d) = ∑m (4,7,12,15) + *Σ*d (0,1,2,3,8,9,10,11)
**(i)** F (a, b, c, d) = Σ m (0, 2, 3, 4, 7, 9, 15) + Σd (6, 8, 11)
**(j)** F (a,b,c,d) = Σ m (0,2,4,6,10,12,15) + Σd (l,3,7,9,11)
**(k)** f (w,x,y,z)=Σ m(0, 1,. 3, 7, 8, 12) + Σd (5, 10, 13, 14)
**(l)** F(a,b,c,d) = Σ m (9,10,12) + Σd (3,5,6,7,11,13,14,15)
**(m)**F (a, b, c, d) = ∑m (1, 3, 7, 11, 15) + ∑d (0, 2, 5)
**(n)** F (p,q,r,s) = ∑m(0,1,2,7,9,12,13) + ∑d (3, 5, 8, 10)
**(o)** F (p, q, r, s) = ∑ (0, 2, 4, 6, 8, 12, 15) + Σ d (1, 3, 5, 7)
**(p)** F (a, b, c, d) = Σ m (1, 5, 6, 12, 13, 14). Σ d (2, 4)
**(q)** F (p,q,r,s) = ∑(5,6,7,8,9) + *Σ* d (10, 11, 12, 13, 14, 15)
**(r)** Y (a, b, c, d) = ∑(1,5, 10, 11, 12, 13, 15) + *Σ*d(14, 9)
**(s)** F(a,b,c,d) =∑ m(4,7,12,15) + *Σ*d(0, 1, 2, 3, 8, 9,10,11)
**(t)** F (w, x, y, z) = ∑ m(0, 1, 2, 5,6,8) + d (3, 4, 7, 14)
**(u)** F (a, b, c, d) = ∑ m(5,7,8,10,13,15) + ∑ d(0,1,2,3)
**(v)** F (a, b, c, d) = ∏M (3,6,8,11,13,14) . ∏d (1,5,7,10)
**(w)**F (a, b, c, d) = ∏M (1, 4, 5, 11, 12, 14) . ∏d (6, 7, 15)
**(x)** F(a,b,c,d) = ΠM (0,2,4,5,6,8,9,10,12,13,14).d(0,2,5)
**(y)** F(a,b,c,d) =∏M (6,7,8,9). ∏d (10, 11, 12, 13, 14, 15)
**(z)** F (p, q, r, s) = IIM (0, 1, 4, 5, 8, 13, 14,). IId (6, 9, 12)

**------৩----☯----ॐ--- ALL THE BEST ---ഌ---🕮----ॐ----🕮---ଓ--- ALL THE BEST ---ॐ----☯----ഴ------**