Unit - V

Digital Electronics

There are 4 different types of number systems that are used in digital electronics. They are:

(i) Decimal Number System.
(ii) Binary Number System.
(iii) Octal Number System.
(iv) Hexadecimal Number system.

(i) Decimal Number System :- (Base radix 10)

The decimal system contains 10 separate symbols, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ex :- $\overset{3\ 2\ 1\ 0}{3609} = 3000 + 600 + 00 + 9$

$= 3 \times 10^3 + 6 \times 10^2 + 0 \times 10^1 + 9 \times 10^0$

(ii) Binary Number System :- (Base radix 2)

The binary number system uses only 2 symbols (0, 1).

In Binary, $\overset{2\ 1\ 0}{1\ 1\ 1} = 1(2)^2 + 1(2)^1 + 1(2)^0$

$= 1(4) + 1(2) + 1(1)$

$= 4 + 2 + 1 = 7$

Converting Binary to Decimal :-

Ex :- $\overset{4\ 3\ 2\ 1\ 0}{1\ 0\ 1\ 1\ 1} = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 16 + 4 + 2 + 1 = 23$

$(10111)_2 = (23)_{10}$

Ex :- $\overset{2\ 1\ 0}{1\ 0\ 1} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 4 + 1 = 5$

$(101)_2 = (5)_{10}$

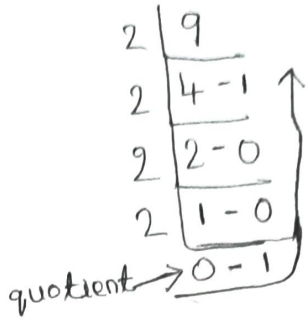
Procedure :-

For converting binary number to decimal number, express the binary number as a sum of powers of '2' & then find the total sum.

Converting Decimal to Binary :-

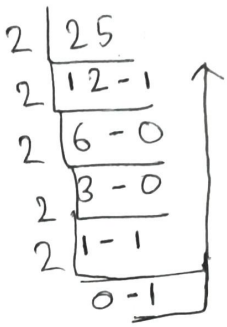
Ex :- $(9)_{10} = (\ )_2$

$= (1001)_2$



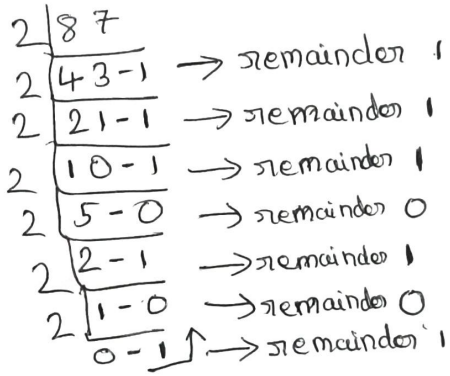
Ex :- $(25)_{10} = (\ )_2$

$= (11001)_2$



Ex :- $(87)_{10} = (\ )_2$

$= (1010111)_2$



Procedure :-

→ For converting decimal number to binary number, progressively divide the decimal number by '2', until the quotient becomes '0', writing down the remainder after each division.

→ The remainders taken in reverse order form the binary number

(iii) **Octal Number System :-** (Base '8')

Octal Number system uses first eight decimal digits :

0, 1, 2, 3, 4, 5, 6, 7.

Fraction

Ex:- $(13.85)_{10} = (\quad)_2$

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

$(0.85)_{10} = (0.110110)_2$

$\therefore (13.85)_{10} = (1101.110110)_2$

2	13
2	6 - 1
2	3 - 0
2	1 - 1
	0 - 1

$0.85 \times 2 = 1.7$, carry '1'

$0.7 \times 2 = 1.4$, carry '1'

$0.4 \times 2 = 0.8$, carry '0'

$0.8 \times 2 = 1.6$, carry '1'

$0.6 \times 2 = 1.2$, carry '1'

$0.2 \times 2 = 0.4$, carry '0'

Fraction

Ex:- $(11.011)_2 = (\quad)_{10}$ (1 0 | -1 -2 -3)

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

$= 2 + 1 + 0 + \frac{1}{4} + \frac{1}{8}$

$= 3\ \frac{3}{8}$

$\therefore (11.011)_2 = \left(3\ \frac{3}{8}\right)_{10}$

iii) Octal Number System:- (Base '8')

Octal Number system uses first eight decimal digits:

0, 1, 2, 3, 4, 5, 6, 7.

→Octal to Decimal Conversion:- $(325)_8 = (\quad)_{10}$ (2 1 0)

$= 3 \times 8^2 + 2 \times 8^1 + 5 \times 8^0$

$= 3 \times 64 + 2 \times 8 + 5 \times 1$

$= 192 + 16 + 5 = 213$

$(325)_8 = (213)_{10}$

Procedure:-

To convert from octal to decimal, multiply each octal digit by its weight (power) and add the resulting products.

→ Decimal to octal conversion:-

Procedure:-

To convert from decimal to octal, we divide by '8', writing down the remainders after each division. The remainders in reverse order form the octal number.

Ex:- $(175)_{10} = (\quad)_8$

$(175)_{10} = (257)_8$

8 | 175
8 | 21 - remainder 7
8 | 2 - remainder 5
0 - remainder 2

Fraction

Ex:- $(0.23)_{10} = (\quad)_8$

$(0.23)_{10} = (0.165)_8$

$0.23 \times 8 = 1.84$ carry 1
$0.84 \times 8 = 6.72$ carry 6
$0.72 \times 8 = 5.76$ carry 5

→ Octal (8) to Binary (2) Conversion:-

Procedure:-

As '8' is the third power of '2', we can convert from octal to binary as follows:

Change each octal digit to its binary equivalent.

Ex:- $(345)_8 = (\quad)_2$

3 → 011
4 → 100
5 → 101

$(345)_8 = (011\ 100\ 101)_2$

Fraction

Ex:- $(253.641)_8 = (\quad)_2$

2	5	3	.	6	4	1
↓	↓	↓	↓	↓	↓	↓
010	101	011	.	110	100	001

$(253.641)_8 = (010\ 101\ 011 . 110\ 100\ 001)_2$

→ Binary (2) to Octal (8) Conversion:-

Procedure:-

→ Group the bits in to threes, starting at the binary point, then convert each group of three to its octal equivalent. (0's are added at each end, if necessary)

Ex:- $(11011)_2 = (\quad)_8$

11011 → [0]11 011 ('0' is added at the left)

[0]11	011
↓	↓
3	3

$(11011)_2 = (33)_8$

Fraction

Ex:- $(1100.01101)_2 = (\quad)_8$

1100.01101 → [00]1 100 . 011 01[0]

[00]1	100	.	011	01[0]
↓	↓	↓	↓	↓
1	4	.	3	2

$(1100.01101)_2 = (14.32)_8$

(iv) Hexadecimal numbers:- (Base 16)

Hexadecimal number system uses 16 digits.

0 to 9, A, B, C, D, E, F

A → 10 D → 13

B → 11 E → 14

C → 12 F → 15

→ The advantage of the hexadecimal system, is its usefulness in converting directly from a 4-bit binary number.

→ Hexadecimal to Decimal conversion:-

Procedure:- To convert from hexadecimal to decimal, multiply each hexadecimal digit by its weight (power) & add the resulting products.

Fraction 4 3 2 1 0 -1 -2 -3

Ex:- $(E\ 8\ B\ A\ 6 \cdot 3\ 9\ C)_{16} = (\quad)_{10}$

$= E(16)^4 + 8(16)^3 + 2(16)^2 + A(16)^1 + 6(16)^0 + 3(16)^{-1} + 9(16)^{-2} + C(16)^{-3}$

Now write the decimal equivalent to hexadecimal numbers

$= 14(16)^4 + 8(16)^3 + 2(16)^2 + 10(16)^1 + 6(16)^0 + 3(16)^{-1} + 9(16)^{-2} + 12(16)^{-3}$

$= 9,53,254 \cdot 2256$

$(E8BA6 \cdot 396)_{16} = (9,53,254 \cdot 2256)_{10}$

→ Decimal to Hexadecimal Conversion:-

Procedure:- To convert decimal to hexadecimal, divide the decimal number successively by 16, writing down the remainders and read up from the bottom to obtain the hexadecimal number.

Ex:- Convert $(2479)_{10}$ into hexadecimal.

$(2479)_{10} = (\quad)_{16}$

16 | 2479		
16 | 154	→ remainder 15	→ F
16 | 9	→ remainder 10	→ A
0	→ remainder 9	→ 9

$(2479)_{10} = (9AF)_{16}$

→ Hexadecimal (16) to Binary (2) Conversion:-

Procedure:- To convert a hexadecimal number to a binary number, convert each hexadecimal digit to its 4-bit equivalent using the binary code.

Ex:- $(4A8C)_{16} = (\quad)_2$

4	A	8	C
↓	↓	↓	↓
0100	1010	1000	1100

$(4A8C)_{16} = (0100\ 1010\ 1000\ 1100)_2$

Ex:- $(FACE)_{16} = (\quad)_2$

F	A	C	E
↓	↓	↓	↓
1111	1010	1100	1110

$(FACE)_{16} = (1111\ 1010\ 1100\ 1110)_2$

Binary to Hexadecimal conversion :-

→ Procedure :- Binary numbers can be easily converted to hexadecimal by grouping in groups of four starting at the binary point. To convert in the opposite direction, from binary to hexadecimal, again use the code.

Ex :- Convert $(11011110101110)_2 = (\quad)_{16}$

First form a group in fours from binary point =

11, 0111, 1010, 1110

Group in fours 0011 0111 1010 1110

↓ ↓ ↓ ↓

3 7 A E

$\therefore (11\ 0111\ 1010\ 1110)_2 = (37AE)_{16}$

Ex :- $(1110\ 1000\ 1101\ 0110)_2 = (\quad)_{16}$

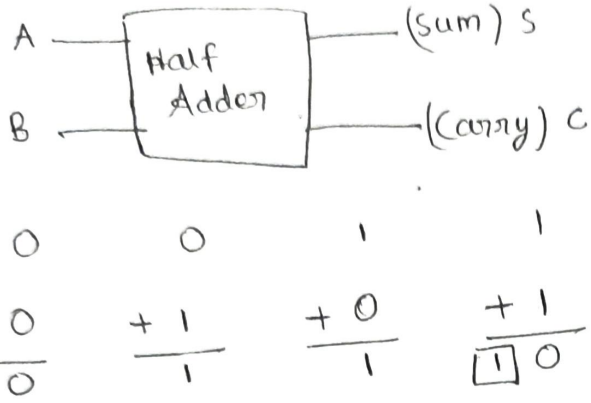
1110 1000 1101 0110

↓ ↓ ↓ ↓

E 8 D 6

$(1110\ 1000\ 1101\ 0110)_2 = (E8D6)_{16}$

* Half Adder



Truth Table

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ +\,0 & +\,1 & +\,0 & +\,1 \\ \hline 0 & 1 & 1 & \boxed{1}\,0 \end{array}$$

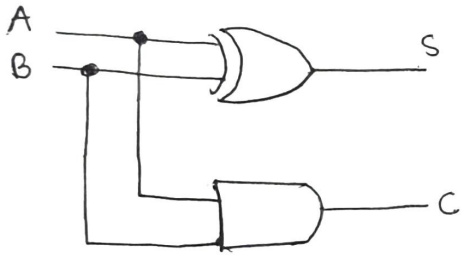
Boolean expression

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

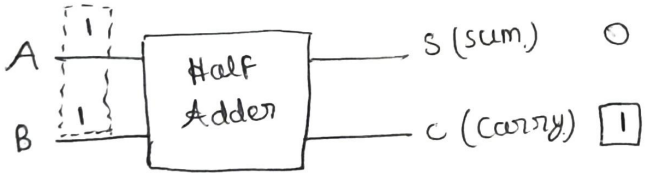
$$C = A \cdot B$$

Implementation of above expressions by using logic gates.

Logic circuit

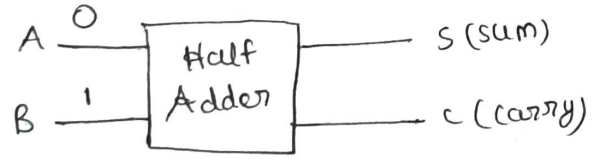


Ex:- 1st addition



$$\begin{array}{cc} \boxed{1} & \\ 0 & 1 \\ +\;1 & 1 \\ \hline 0 & \end{array}$$

2nd addition



After 1st addition of '2' bits, carry is generated. This carry can not be considered in the 2nd addition. In the 2nd addition, only next '2' bits will be added.

So the disadvantage in Half adder is, we can not add the previous carry.

* Full Adder

Full adder is able to add 2 bits along with previous carry.

Truth Table

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2nd row

$0 \rightarrow A$

$+\ 0 \rightarrow B$

0

$+\ 1 \rightarrow C_{in}$

$1 \rightarrow$ sum

$C_{out} = 0$

Last row

$1 \rightarrow A$

$+\ 1 \rightarrow B$

Carry → 1 0

$+\ 1 \rightarrow C_{in}$

1 $1 \rightarrow$ sum

C_{out}

Boolean expression

sum (s) = 1 for 4 combinations

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

$= \bar{A}(\underbrace{\bar{B}C_{in} + B\bar{C}_{in}}_{(1)}) + A(\underbrace{\bar{B}\bar{C}_{in} + BC_{in}}_{(2)})$

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

$\overline{A \oplus B} = \bar{A}\bar{B} + AB$ → (2)

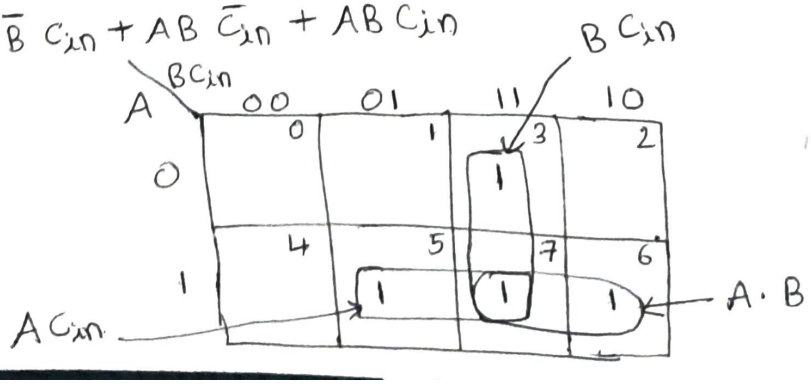
$= \bar{A}\underbrace{(B \oplus C_{in})}_{(1)} + A\underbrace{(\overline{B \oplus C_{in}})}_{(2)}$

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

1st method

$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$

K-map method



$\therefore C_{out} = AB + BC_{in} + A \cdot C_{in}$

2nd method

$C_{out} = \bar{A}\,B\,C_{in} + A\,\bar{B}\,C_{in} + AB\,\bar{C}_{in} + ABC_{in}$

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

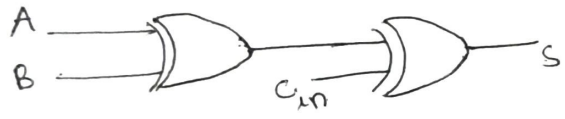
($C_{in} \cdot \bar{C}_{in} = 0$, $C_{in} + \bar{C}_{in} = 1$)

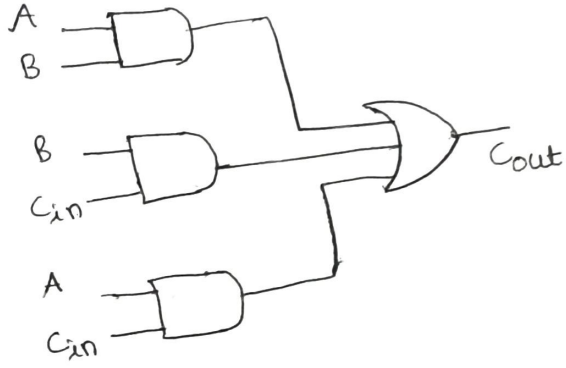
$= C_{in}(A \oplus B) + AB$

$\therefore S = A \oplus B \oplus C_{in}$

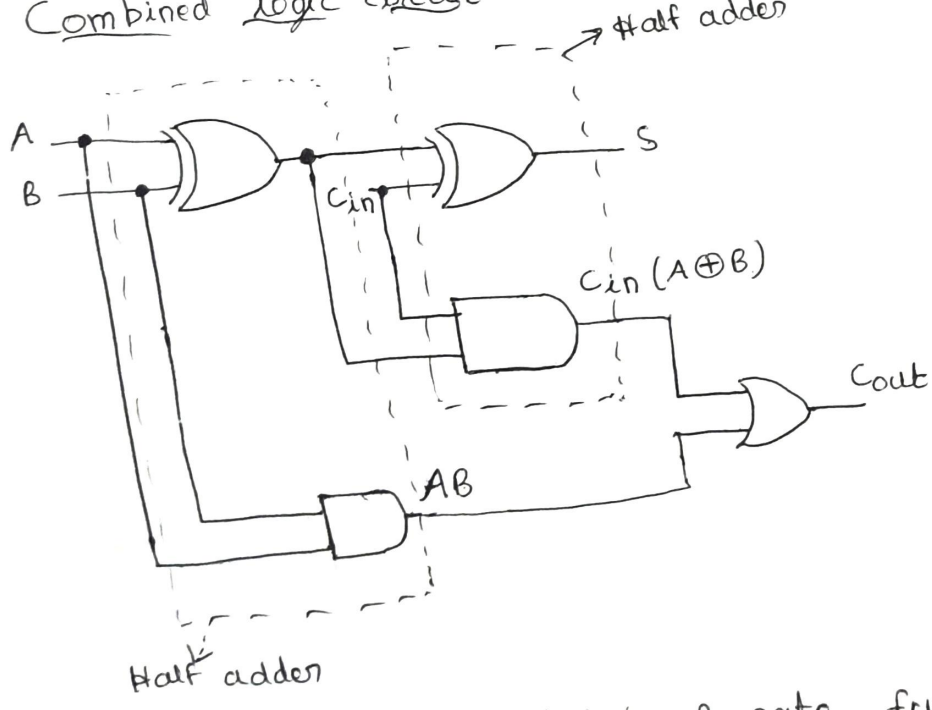
$C_{out} = AB + BC_{in} + AC_{in} = AB + C_{in}(A \oplus B)$

Logic circuit





Combined logic circuit



i.e using 2 Half adders and 1 OR gate, full adder can be generated.

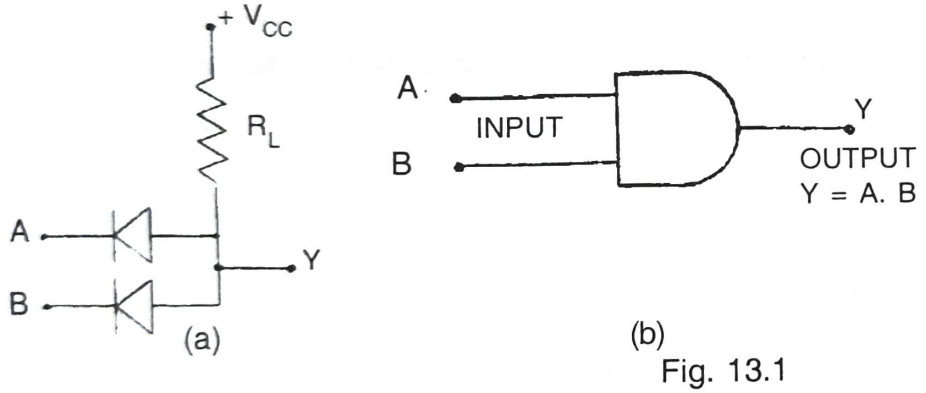
13.10 LOGIC GATES:

A logic gate is a circuit used to realize a basic logic function. The logic gate is the basic building block in digital systems. Logic gates operate with binary numbers. Gates are therefore referred to as binary logic gates. All voltages used with logic gates will be either "HIGH" or "LOW". In the binary system, a HIGH voltage will mean a binary number 1 and a LOW voltage will mean a binary number 0. The Logic gates are electronic circuits. These circuits will respond only to HIGH voltages (called 1s) or LOW (ground) voltages (called 0s).

All digital systems are constructed by using only three basic logic gates. These basic gates are called the AND gate, the OR gate, and the NOT gate, combinations of logic gates serves to implement complex boolean equations. A gate is a logic circuit with one or more inputs but only one output. By connecting these three basic gates in different ways, we can build circuits that perform arithmetic and other functions associated with the human brain. Because they simulate mental processes, gates are often called as "logic gates" or "logic circuits". Hence a logic gate is an electronic circuit which makes logic decision.

13.11 THE AND GATE :

The output of an AND gate is high only when all the inputs are high. The AND gate is called as "all or nothing" gate. Fig. 13.1 (a) shows a simplified circuit for the AND gate. The standard logic symbol for the AND gate is shown in fig. 13.1(b). This symbol shows the inputs as A and B. The output is shown as Y. This is the symbol for a 2 input AND gate. The truth table for the 2 input AND gate is shown in Table shown in fig. 13.1 (C)



Truth Table

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(c)

Fig. 13.1

The inputs are shown as binary digits (bits). Note that only when both input A and input B are 1 (high) then only the output will be 1 (high). Binary 0 is defined as a LOW or ground voltage and Binary 1 is defined as HIGH voltage. High voltage will mean about +5 volts.

The AND function is denoted by the symbol of a dot(•). Boolean expression for AND gate is

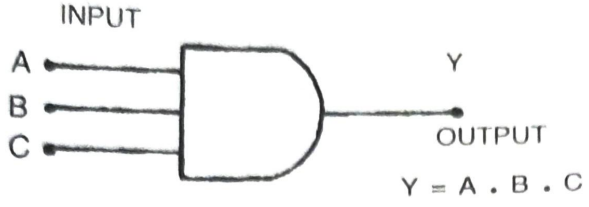
$$Y = A \bullet B$$

The Boolean expression is read as the output Y equals to A AND (•means AND) B. The dot (•) means the logic function AND in Boolean algebra, not multiply as in regular algebra. Sometimes the dot (•) is left out of the Boolean expression. The Boolean expression for the 2-input AND gate is then

$$Y = AB$$

It is read as the output Y equals to A AND B.

The AND gates may have any number of input. Fig. 13.2(a) shows a 3 input AND gate. The inputs are A, B and C. When all inputs are low the output Y is low. Even if one input is low then the output will be low. The only way to get a high output is to make all the input as high. The logic symbol and truth table for 3 input AND gate is shown in Fig. 13.2 (a) and (b).



(a)

Truth Table

Input			Output
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b)

Fig. 13.2

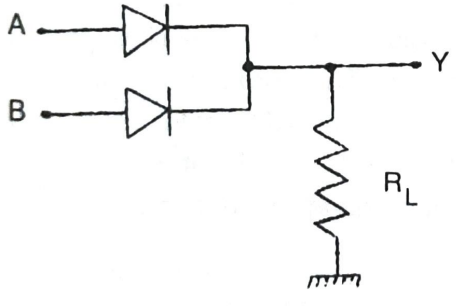
The Boolean expression for this output Y = A.B.C. or Y = ABC

Hence the AND gate has a high output when all the inputs are high. In other words, the AND gate is an all-or-nothing gate; a high output occurs only when all inputs are high.

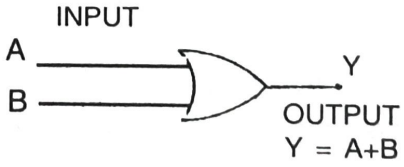
13.12 THE OR GATE :

The output of an OR gate is high if any or all of the inputs are high. The OR gate is called as "any or all" gate. Fig. 13.3 (a) shows the simplified circuit for the OR gate. The standard logic symbol for the OR gate is shown in fig. 13.3(b). This symbol shows the input as A and B. The output is shown as Y. This is the symbol for a 2 input OR gate. The truth table for a 2 input OR gate is shown in Table shown in fig. 13.3 (c).

The OR gate implements the logic of high output when one or more inputs are high. That is, the output will be low only when all the inputs are low. In boolean algebra, the OR function is denoted by symbol "+". So the Boolean expression for OR gate is, Y = A+B.



(a)



(b)

Truth Table

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(c)

Fig. 13.3

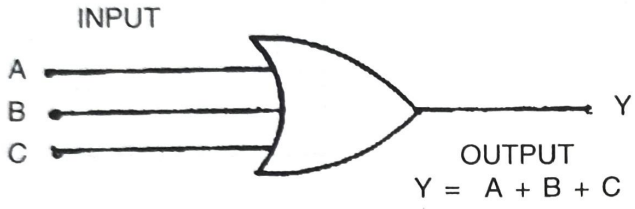
Note that the plus (+) symbol means OR in Boolean algebra. The expression (Y=A+B) is read as the output Y equals to A OR (+ means OR) B. The plus sign does not mean to add as it does in regular algebra.

By careful examination of the truth table given in fig.13.3(c), the OR gate has a high output when either A or B or both are high. In otherwords the OR gate is an any-or-all gate, an output occurs high when any or all of the inputs are high.

TRUTH TABLE :

A truth table may be defined as a table which gives the output state for all possible input combinations.

The OR gates may have any number of inputs. Fig. 13.4 (a) shows a 3 input OR gate. The inputs are A, B and C. When all inputs are low the output Y is low. Even if one input is high, then the output will be high. The logic symbol and truth table for 3 input OR gate is shown in fig. 13.4 (a) and (b).



(a)

Fig. 13.4

Truth Table

Input A	Input B	Input C	Output Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b)

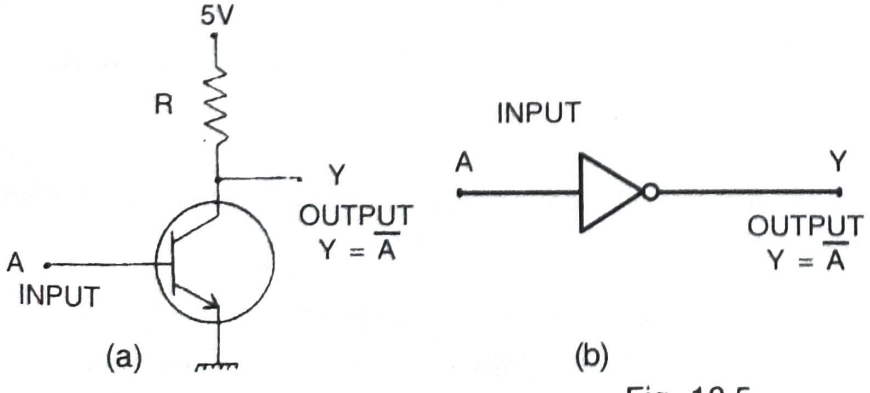
Fig. 13.4

The Boolean expression for this output Y = A+B+C.

Hence the OR gate has a high output when any or all the inputs are high. In otherwords, the OR gate is any or all gate, a high output occurs when any or all inputs are high.

13.13 THE INVERTER GATE OR NOT GATE :

A NOT gate is also called as an Inverter gate. A NOT gate, or Inverter, is an unusual gate. The NOT gate has only one input and one output. It is called as NOT gate because its output is not the same as its input. It is also called as an inverter because it inverts the input signal. All it does is to invert (or complement) the input as seen from its Truth table Fig. 13.5 (c). Fig. 13.5. (a) shows the simplified circuit for the NOT gate. The standard logic symbol for the NOT gate is shown in Fig. 13.5 (b). This symbol shows one input and one output. The output is the opposite of input. The input is always changed to its opposite. If the input is 0, the NOT gate will give its complement or opposite which is 1. If the input to the NOT gate is a 1, the circuit will invert or complement it to give a 0. This inverting is also called as complementing or negating.



Truth Table

Input A	Output Y
0	1
1	0

(c)

Fig. 13.5

The inverter cannot be designed using passive devices alone, such as resistors and diodes. The circle in the symbol Fig. 13.5 (b) actually represents the inversion and the triangle an amplifier.

The Boolean expression for inverter is $y = \overline{A}$. $\overline{A}$ reads as A equals to output not A. The bar over the A means to complement A. Complement of 1 is 0 and complement of 0 is 1.

i.e. $\overline{0} = 1$ (or) $\overline{1} = 0$.

In case double complementation gives the original value i.e. $\overline{\overline{0}} = \overline{1} = 0$

(or) $\overline{\overline{1}} = \overline{0} = 1$

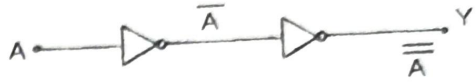
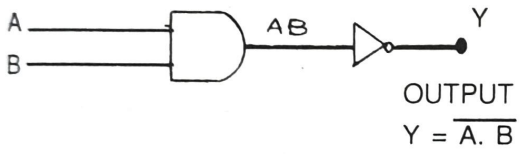


Fig. 13.6

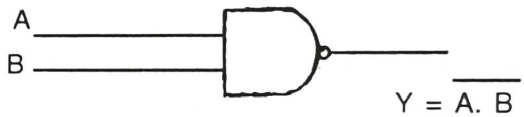
Fig. 13.6 shows the double complementation which gives the original value as output.

13.14 THE NAND GATE :

This gate is a combination of AND and NOT gates. It is in fact a NOT-AND gate. It can be obtained by connecting a NOT gate in the output of an AND gate as shown in Fig. 13.7 (a). The standard logic symbol for the NAND gate is shown in fig. 13.7 (b). The Truth Table for the NAND gate is shown in fig. 13.7(c).



(a)



(b)

Fig 13.7

Truth Table

Input		Output
A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

(c)

The Boolean expression for the entire circuit is $Y = \overline{A.B}$. It is said that this is a NOT-AND or NAND gate. Note that the NAND gate symbol is an AND gate symbol with a small bubble at the output. The bubble is sometimes called as an invert bubble. The truth table describes the exact operation of a logic gate. The truth table for the NAND gate is shown in fig. 13.7(c). The words NOT-AND are contracted to NAND. Whenever we see this NAND symbol, remember that the output is NOT the AND of the inputs. With a NAND gate, all inputs must be high to get a low output. If any or both input is low, the output is high. In otherwords, it gives an output 1 if either A or B or both are 0.

The diode transistor equivalent of a NAND gate is shown in fig. 13.8.

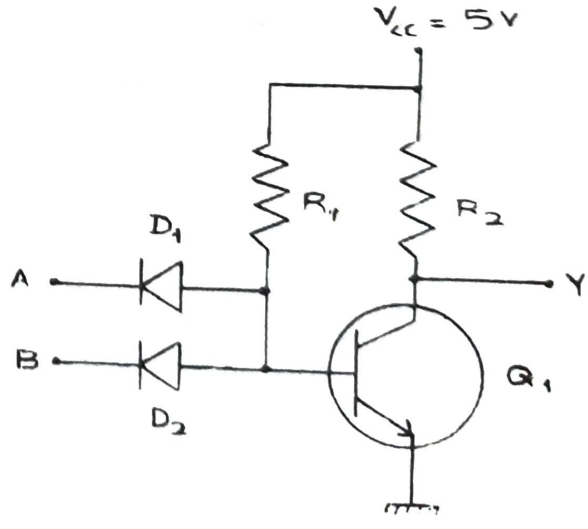
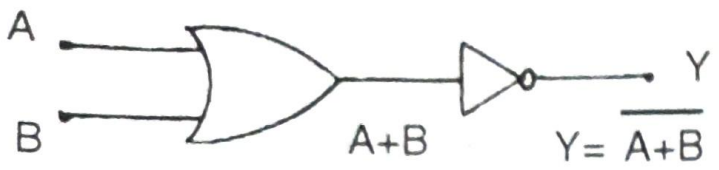


Fig 13.8

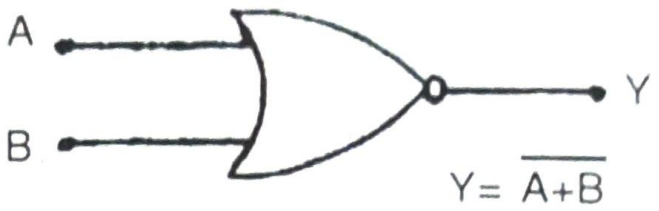
the three basic logic gates such as [illegible]. Hence the NAND gate is called as universal gate or universal building blocks.

13.15 THE NOR GATE :

This gate is a combination of OR and NOT gates. It is in fact a NOT-OR gate. It can be obtained by connecting a NOT gate in the output of an OR gate as shown in fig. 13.10 (a). The standard logic symbol for the NOR gate is shown in fig. 13.10 (b). The truth table for the NOR gate is shown in fig. 13.10 (c).



(a)



(b)

Truth Table

Input A	Input B	Output Y
0	0	1
0	1	0
1	0	0
1	1	0

(c)

Fig. 13.10

The Boolean expression for the entire circuit is $Y = \overline{A+B}$. It is said that this is a NOT-OR or NOR gate. Note that the NOR gate symbol is a OR gate symbol with a small bubble at the output. The bubble is some times called as an invert bubble. The Truth table describes the exact operation of a logic gate. The truth table for the NOR gate is shown in Fig. 13.10 (c). The words NOT-OR are contracted to NOR. Whenever we see this NOR symbol remember that the output is NOT the OR of the inputs. With a NOR gate, all the inputs must be low to get a high output. If any or both input is high the output is low. In other words, it gives an output of 0 if either A or B or both are 1. A NOR gate will have an output of 1 only when all its inputs are 0. Obviously, if any output is 1, the output will be 0. The transistor equivalent, of the NOR gate is shown in fig. 13.11.

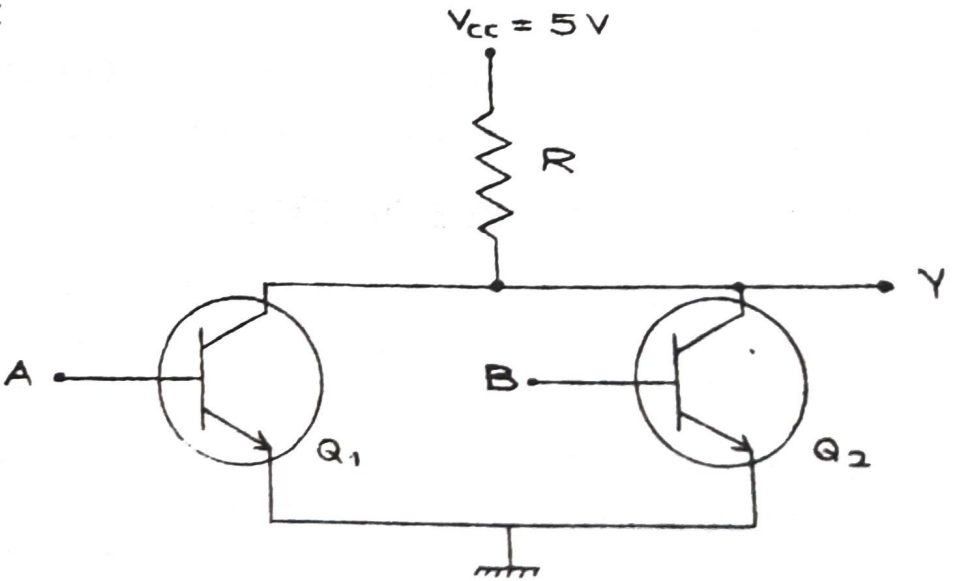


Fig. 13.11

NOR GATE IS A UNIVERSAL GATE :

It is interested to note that a NOR gate also can be used to realize the basic logic functions of OR gate, AND gate and NOT (Inverter) gate. Hence this NOR gate can also be called as universal gate. Since like NAND gate, the NOR gate also can perform all the three fundamental logic gate's function, the NOR gate is also called as the universal gate.

USE OF NOR GATE AS NOT

13.16 THE EXCLUSIVE OR GATE :

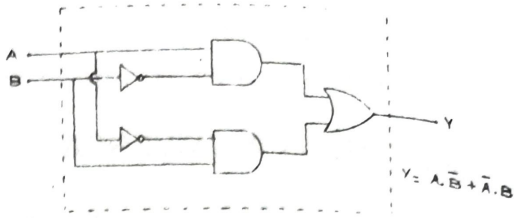
The output of this gate is high if only one of the two inputs is high. The standard symbol of Exclusive OR gate is given in fig. 13.13 (a) and its truth table is given in fig. 13.13 (b). Fig. 13.13.(c) shows the equivalent logic circuit that performs the exclusive OR function. The exclusive - OR gate is referred to as the "any but not all" gate. The exclusive - OR term is often shortened to read as XOR or some times as EXOR.

(a)

Truth Table

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(b)



(c)

Fig 13.13

The exclusive OR gate is also called as XOR gate. In boolean algebra, we denote XOR operation by ⊕ symbol. Thus for XOR gate $Y = A \oplus B$

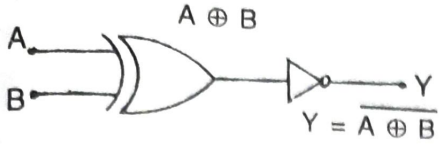
It means $Y = A \cdot \overline{B} + \overline{A} \cdot B$.

The XOR gate is enabled only when an odd number of 1s appear at the inputs. Lines 2 and 3 of the truth table have odd numbers of 1s, and therefore the output is enabled with a 1. Lines 1 and 4 of the truth table contain even numbers of 0 s and 1s, and therefore the XOR gate is disabled and a 0 appears at the output.

In this XOR gate, the output is 1 if it's either input is 1, and output is 0 when the inputs are the same. The circuit is also called an inequality comparator or detector because it produces an output only when the two inputs are different.

13.17 THE EXCLUSIVE NOR GATE :

It is known as a NOT-XOR gate, i.e. $\overline{\text{XOR}}$ gate. The output of an $\overline{\text{XOR}}$ gate is shown inverted in Fig. 13.14 (a). The output of the inverter on the right side is called as the exclusive-NOR (XNOR) function. The XOR gate produces the expression A⊕B. When this is inverted, it forms the Boolean expression for the XNOR gate, $Y = \overline{A \oplus B}$. The standard logic symbol for XNOR gate is shown in fig. 13.14 (b). Note that the symbol is an XOR symbol with an invert bubble attached to the output. The truth table for this X NOR gate is shown in fig. 13.14 (c).



(a)

(b)

Truth Table

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(c)

Fig. 13.14

Note that all outputs of the XNOR gate are the complements of the XOR gate outputs. While the XOR gate is an odd number of 1s detector, the XNOR gate detects even numbers of 1s. The XNOR gate will produce a 1 output when an even number of inputs appear in the input. In otherwords, for getting an output its both inputs should be at the same logic level of either 0 or 1. Obviously, it produces no output if its two inputs are at the opposite logic level.

13.18 DE-MORGAN'S THEOREMS :

First Theorem states that the complement of a sum equals the products of complements

i.e. $\overline{A + B} = \overline{A} \cdot \overline{B}$

Second theorem states that the complement of a product equals the sum of complements.

i.e. $\overline{A \cdot B} = \overline{A} + \overline{B}$

In fact these theorems allows transformation from a sum of product form to a product of sum form.

13.19 BOOLEAN ALGEBRA AND LAWS:

INTRODUCTION :

Boolean Algebra (named after its Pioneer George Boole 1815 - 1864) is the algebra of logic presently applied to the operation of Computer devices. The rules of this algebra are based on human reasoning. It originated from the study of how we reason, what lines of reason are valid and what constitutes proof etc.

Starting with his investigation of the laws of thought, Boole developed in 1854 a mathematical system of logic in which he expressed truth functions as symbols and then manipulated these symbols to arrive at a conclusion. His new system was not the ordinary numerical algebra which we know from our high school days but a totally new system called logic algebra or Boolean Algebra. For example in Boolean algebra $A + A = A$ and not 2A as is the case in ordinary algebra.

As we know, all thinking and logic is concerned with finding answers to binary or two-valued questions like: is it good or bad, right or wrong, true or false, day or night,etc. This binary nature of logic is exactly like the binary working of relay and switching circuits where relay is either energised or not, light is ON or OFF or pulse is present or not. Because of its very logical nature, Boolean algebra is ideal for the design and analysis of logic circuits used in computers. Moreover it provides an economical and straight forward way of describing computer circuiting and complicated switching circuits. As compared to other mathematical tools of analysis and design, Boolean algebra has the advantages of simplicity, speed and accuracy.