MODULE 3

Digital Electronics: Introduction, Switching and Logic Levels, Digital Waveform. Number Systems: Decimal Number System, Binary Number System, Converting Decimal to Binary, Hexadecimal Number System: Converting Binary to Hexadecimal, Hexadecimal to Binary, Converting Hexadecimal to Decimal, Converting Decimal to Hexadecimal, Octal Numbers: Binary to Octal Conversion. Complement of Binary Numbers. Boolean Algebra Theorems, De Morgan's theorem. Digital Circuits: Logic gates, NOT Gate, AND Gate, OR Gate, XOR Gate, NAND Gate, NOR Gate, X-NOR Gate. Algebraic Simplification, NAND and NOR Implementation: NAND Implementation, NOR Implementation. Half adder, Full adder.

Digital Electronics:

Introduction:

Analog System:

It is a system in which signals have infinite number of values. The information varies continuously with time. Analog system processes time varying signals that can take on any value across a continuous range of voltage, current or any physical parameter.

Digital System:

It is a system in which signals have finite number of discrete values to represent information. Examples are electric impulses, decimal digits, arithmetic operations, etc can be processed by the digital system

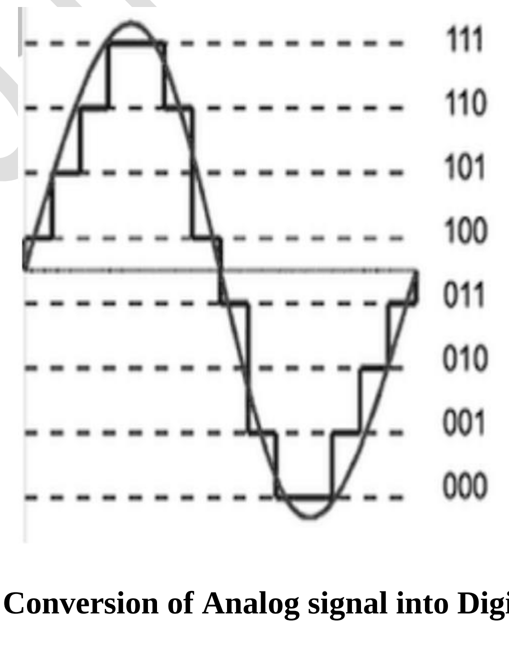


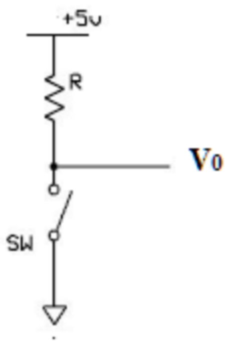
Fig. 3.1 Conversion of Analog signal into Digital signal

The main steps involved in converting the analog signal into digital signal are sampling, Quantizing and encoding. The figure 3.1 shows the three processes. The sampling is done with the help of sample and hold circuit. And Encoder converts voltage level into digital data.

Binary Signal: In Digital systems, the most common digital signal; is the signal that has one of the two possible values, like ON or OFF (often represented as binary 1 or 0). These signals are called as binary signal.
Binary signals have the great advantage of being far less susceptible to noise compared to analog signals.
Binary signal are used extensively in communication control and instrumentation as well in computers. The numbers are coded in the form of binary (ON/OFF) electrical pulses and processed by means of logic gates and memory cells. Four major technologies used here

TTL-Transistor Transistor Logic
ECL-Emitter Coupled Logic
NMOS N-type Metal Oxide Semiconductor CMOS-
Complementary Metal Oxide Semiconductor

Switching and Logic Levels



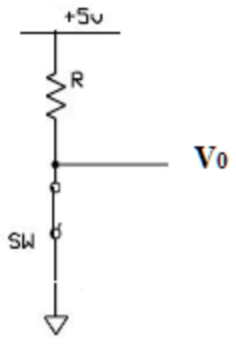


Fig. 3.2 a) Switch open **Fig. 3.2 (b) switch close**

Consider a simple circuit shown in figure 3.2. The switch SW connected in series with a resistor 'R' to a DC supply V_{CC} of 5V. The output voltage is measured at the junction of resistor and the switch. Such circuit is called a swicthing circuit.
When Switch is OPEN, The current flowing through it is zero.
The output is equal to V_{CC} - IR=V_0
When I=0. V_0=V_{CC}
The output voltage is equal to supply voltage V_{CC} is considered as logic 1(High).

When the Switch is CLOSED. The supply is connected to ground directly through a switch SW, hence the output voltage becomes zero. V_O=0. Logic 0 (LOW). Truth table is shown in below in table3.1.

Table 3.1 Truth table of NOT gate

Switch status	**Output Voltage**	**Logic status**
CLOSE	0V	Low
OPEN	V_{cc}	High

Switching circuit:
It is defined as a circuit whose output voltage has only specific voltage level namely High and Low. The two voltage levels used to represent logic state are called logic levels. The digital electronics works on data which is in one of the two states.

- Logic 1 means +5 Volts
- Logic 0 means 0 Volts
- True(1) and False(0) used in Boolean Algebra.

Concept of Positive and Negative Logic

If the higher of two voltages represents a 1 and lower voltage represents a 0, the system is called a Positive Logic system. On the other hand, if the lower voltage represents a 1 and higher voltage represents a 0, we have negative logic system. Positive logic is more Common.

Digital Waveform

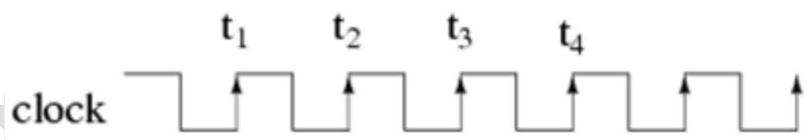


Fig 3.3 Clock signal

Clock signal is a digital waveform showing the logic ‘0’ and Logic ‘1’ levels are represented in figure 3.3. It’s a square waveform. The period (T_P) is measured from one edge of the clock to the next similar edge of clock.

The digital waveforms logic levels are ideal . The values are
where 5V=high= logic 1
where 0V=low= logic 0
the change over from High (1) to Low (0) and vice versa is in zero time. But in practice, the voltage at different nodes in the digital circuit may differ slightly due to internal resistances, parasitic effects and loading effects.

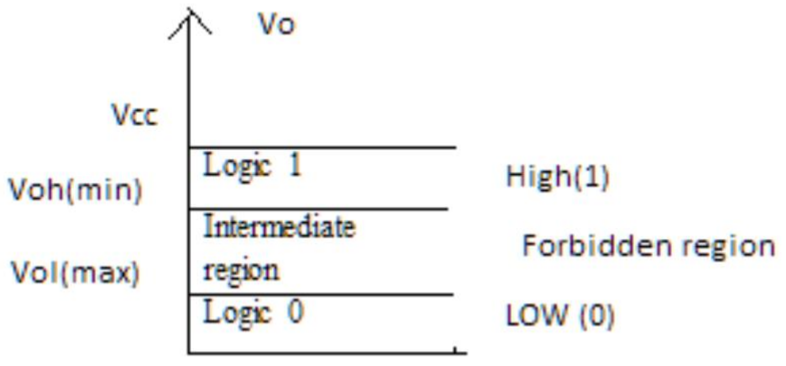


Fig. 3.4 Logic Levels

Therefore, to define a logic level, a range of voltage is assigned instead of a particular voltage level. As shown in figure 3.4 any voltage level between Vcc and Voh(min) is considered as high output i.e. binary 1. Similarly any voltage between Vol(max) and ground is considered as low voltage i.e., binary 0.

Introduction to Number systems

Number system is the basis for counting various items. The representation of number in a specific way is called a number system. **Data** are a collection of facts and figures which act as a raw material for information. The processed data which help to make decisions or further manipulations are called **information**. The total number of symbols used in a particular number system is called base or radix and each symbol is called a **digit**.

There are Four number system in Digital electronics are

1. Decimal Number system
2. Octal
3. Binary
4. Hexa-decimal

1). Decimal Number System: The decimal number system has base 10 and the digits are 0, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9.

2). Binary Number System: To binary number system have base 2 and the binary digits 0 and 1. **BIT** is a contraction of the world's **BI**nary digi**T.**

Applications: used in Electronic and electrical components of a computer. Computer uses data, these are bi-stable in nature. The binary digit 0 and 1 are most suitable and are conveniently used to express the two possible status.

3). Octal number systems: These are also used in digital computers. Octal number system has a base 8 and the symbols used are 0, 1, 2, 3, 4, 5, 6, 7.

4). Hexadecimal number systems: Hexadecimal number system has a base 16 and the symbols used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

1). Decimal Number System

Decimal number system can be expressed in terms of units, tens, hundreds, thousands and som on. In this system, the symbols used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and the base is 10. Thus the number

$d_{n-1}\ d_{n-2}\ldots.d_1\ d_0.\ d_{-1}\ d_{-2}\ldots\ldots$ means

$d_{n-1} * 10^{n-1} + d_{n-2} * 10^{n-2} + \ldots\ldots + d_1 *10^1 + d_0 *10^0$ and this is an n-digit number.

If the number be extended to the right of the decimal point, then the powers of the base will be negative starting from -1.

$d_1 *10^{-1} + d_0 *10^{-2}$

For example:

1) the number has the magnitude

$1527=1\times10^3+5\times10^2+2\times10^1+7\times10^0$

2) the number has the magnitude

$27.56=2\times10+7\times10^0+5\times10^{-1}+6\times10^{-2}$

3) 526.44 has magnitude

Given Number	5	2	6	.	4	4
Position	Digit-2	Digit-1	Digit-0		Digit(-1)	Digit(-2)
Weightage	10^2	10^1	10^0		10^{-1}	10^{-2}

2). Binary Number System

These are used in the design of digital computers. Binary number system uses two symbols 0 and 1 and its radix is 2. The symbols 0 and 1 are generally called **BITS** which is a contraction of the two words Binary digits.

An n-bit binary number of the form $A_{n-1}\ A_{n-2}\ \ldots..\ A_1\ A_0$ where each A_i ($i = 0, 1, \ldots. n - 1$) is either 0 or 1 has the magnitude.

$A_{n-1}\ 2^{n-1} + A_{n-2}\ 2^{n-2} + \ldots\ldots + A_1\ 2^1 + A_0 2^0$.

For fractional binary numbers, the base has negative integral powers starting with -1 for the bit position just after the binary point. The bit at the extreme left of a binary number has the highest positional value and is usually called the **Most Significant Bit** or **MSB**. Similarly, the bit occupying the extreme right position of a given binary number has the least positional value and is referred to as the **Least Significant Bit** or **LSB**.

In binary number system a few examples on binary numbers and their decimal equivalents are given below:

$(100101)_2=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$
$= 32+0+0+4+0+1$
$= (37)_{10}$
It can also be written as below in the table

Given Number	1	0	0	1	0	1	Decimal Number
Position	2^5	2^4	2^3	2^2	2^1	2^0	
Weightage	32	16	8	4	2	1	
Value	32	0	0	4	0	1	37

Binary point
101.101_2
$= 1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$
$= 4+0+1+.5+0+.125$
$= (5.625)_{10}$

The results can be written as below

Given Number	1	0	1	.	1	0	1	Decimal Number
Position	2^2	2^1	2^0		2^{-1}	2^{-2}	2-3	
Weightage	4	2	1		0.5	0.25	0.125	
Value	4	0	1		0.5	0	0.125	5.625

3). Octal Number System

Eight numbers are used namely 0, 1, 2, 3, 4, 5, 6, 7 are used to represent octal numbers. Octal number system has a base or radix 8.

Convert $\mathbf{273_8} = (?)_{10}$
$= 2\times8^2+7\times8^1+3\times8^0$
$= 128+56+3$
$= (187)_{10}$
Convert the decimal numbers to their octal equivalents:

The octal number representation

The number 526.44 in octal representations

Given Number	5	2	6	.	4	4
Position	Digit 2	Digit 1	Digit 0		Digit(-1)	Digit(-2)
Weightage	8^2	8^1	8^0		8^{-1}	8^{-2}

4). Hexadecimal Number System

- The hexadecimal number system has a radix or base 16.
- It requires 16 symbols to represent a number in this system.
- The symbols are 0 to 9, A, B, C, D, E, F where the symbols A, B, C, D, E, F represent the decimal numbers 10, 11, 12, 13, 14, 15 respectively.

The number 526.44 in Hexadecimal representations

Given Number	5	2	6	.	4	4
Position	Digit 2	Digit 1	Digit 0		Digit(-1)	Digit(-2)
Weightage	16^2	16^1	16^0		16^{-1}	16^{-2}

1. Conversion between Decimal, Octal, Hexadecimal and Binary Numbers

Case1) Converting Any Number system into Decimal

Procedure:

Step1: Identify the weightage of each digit in the number system.

Step2: Multiply each digit by the weightage of the digit.

Step3: add all the products.

1) Converting Binary to Decimal Conversion

1) Convert the given binary number (11010) into decimal

Given Number	1	1	0	**1**	0
Position	Bit 4	Bit3	Bit2	**Bit1**	Bit0
Weightage	2^4	2^3	2^2	2^1	2^0

$= 1x2^4+1x2^3+0x2^2+1x2^1+0x2^0$

$=16+8+0+2+0$

$=(26)_{10}$

2) Convert binary number 1011.110 to decimal

Given Number	1	0	1	1	.	1	1	0
Position	Bit 3	Bit2	Bit1	Bit0		Bit3	Bit2	Bit1
Weightage	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
value	1x8	0x4	1x2	1x1		1x0.5	1x0.25	0x0.125

$=8+2+1+0.5+0.25$

$=(11.75)_{10}$

2). Convert Octal to Decimal

1. Convert given octal number 212 into decimal

Given NO.	2	1	2
Position	Digit 2	Digit 1	Digit 0
Weightage	8^2	8^1	8^0

$= 2x\ 8^2+1x8^1+2x8^0$

$=128+8+2 = (138)_{10}$

2. Convert $(235.67)_8$ into

Given NO.	2	3	5	.	6	7
Position	Digit 2	Digit 1	Digit 0		Digit(-1)	Digit(-2)
Weightage	8^2	8^1	8^0		8_{-1}	8_{-2}

$= 2x\ 8^2+3x8^1+5x8^0+6x\ 8^{-1}+7x8^{-2}$

$=128+24+5+0.75+0.109375 = (157.8594)_{10}$

3). Conversion of Hexadecimal Number into Decimal

1. Convert $A6A_{16}$ to its decimal equivalent.
   Solution:
   $(A6A)_{16}$
   $= 10\times16^2+6\times16^1+10\times16^0$
   $= 2560 + 96 + 10$
   $= 2666_{10}$
   Therefore, $A6A_{16} = 2666_{10}$
2. Convert $(235.A0)_{16}$ into decimal

Given NO.	2	8	5	.	A	9
Position	Digit 2	Digit 1	Digit 0		Digit(-1)	Digit(-2)
Weightage	16^2	16^1	16^0		16^{-1}	16^{-2}

$= 2\times16^2+8\times16^1+5\times16^0+A\times16^{-1}+9\times16^{-2}$
$= 512+128+5+0.625+0.03516$
$= 645.66016_{10}$
Therefore, $(235.A0)_{16}$

Case2). Conversion of Decimal Numbers to Any other Number system.

Integer Part Conversion: Successive division is used.

- Perform repeated division of the given decimal value by the base of number system into which the conversion is required.
- After each division, the remainder is taken as one digit/ bit in destination number system.
- For the next division, take the quotient of the previous division as the dividend.

For the fractional part conversion: Successive multiplication is used.

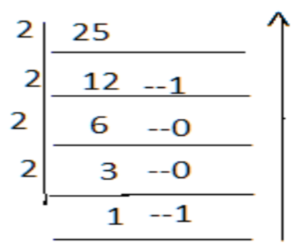
- Perform the repeated multiplication of the fractional part of the decimal number with the base of the number system into which the conversion required.
- After each conversion, the integer part is taken as one digit/bit in the destination number system.
- For the next multiplication, take any previous fractional part
- Repeat above steps till fractional part of the multiplication becomes ‘0’ (or until desired value we get).

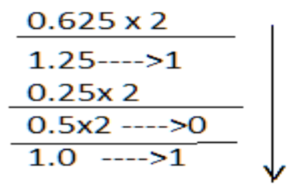
Note: Approximate the result to first four places.

4). Convert Decimal Number into Binary Number

1) **Convert $(25.625)_{10}$ =( )$_2$**

Integer Part: fractional Part:



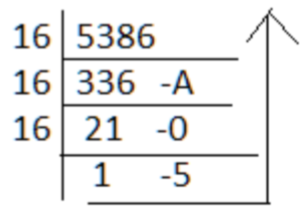


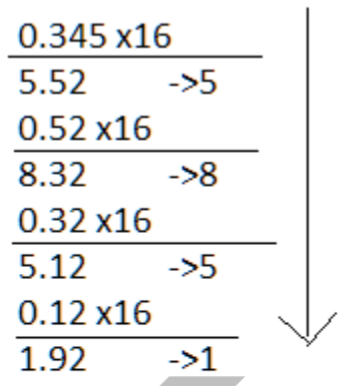
Therefore $(25.625)_{10} = (11001.101)_2$

2) Convert $(264.325)_{10}$= ( )$_2$

6). Convert Decimal number into hexadecimal number

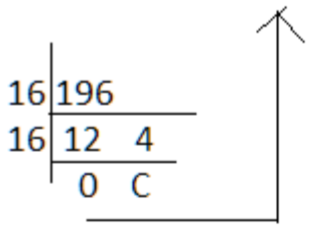
1).Convert $(5386.345)_{10} = (?)_{16}$

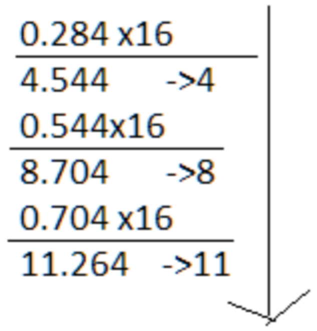




hence $(5386.345)_{10} = (150A.5851)_{16}$

2).Convert $(196.284)_{10} = (?)_{16}$





The answer of decimal number $(196.284)_{10} = (C4.48B)_{16}$

Case3). Conversion of Binary Numbers to Octal or Hexadecimal Numbers

7). Conversion of binary numbers to octal Number system

Conversion Table

Decimal	Binary	Octal	3-bit data	hexadecimal	4-bit data
0	0	0	000	0	0000
1	1	1	001	1	0001
2	10	2	010	2	0010
3	11	3	011	3	0011
4	100	4	100	4	0100

5	101	5	101	5	0101
6	110	6	110	6	0110
7	111	7	111	7	0111
8	1000	10	-	8	1000
9	1001	11	-	9	1001
10	1010	12	-	A	1010
11	1011	13	-	B	1011
12	1100	14	-	C	1100
13	1101	15	-	D	1101
14	1110	16	-	E	1110
15	1111	17	-	F	1111

1. Convert the following to binary numbers into octal numbers

1). 1110101111_2
Solution:
001 $\underline{110}$ 101 $\underline{111}$
= 001 110 101 111
= 1657_8
Hence the required octal equivalent is $(1657)_8$.

2). 101101.01111_2
Solution:
101 $\underline{101}$.011 $\underline{110}_2$
= 55.36_8
Hence the required octal equivalent is $(55.36)_8$.

2. Convert the following to Octal Numbers to their binary equivalents:

(a) 163_8
Solution:
163_8
= 001 110 011
= 1110011_2
Hence the required binary number is $(1110011)_2$

(b) 34.75_8
Solution:
34.75_8
= 011 100 . 111 101

$= 11100.111101_2$

Hence the required binary number is $(11100.111101)_2$.

8). Convert Binary to Hexadecimal

1. Convert the following binary number into Hexadecimal numbers:

(a) 111101100_2

Solution:

$\underline{0001}$ 1110 $\underline{1100}$ =

0001 1110 1100

$= 1EC_{16}$

Therefore, $11110\ 1101_2 = 1EC_{16}$

(b) 11111.01011_2

Solution:

$1\underline{1111}.01011_2$

= 0001 1111 . 0101 1000

$= 1F.58_{16}$

Therefore, $11111.01011_2 = 1F.58_{16}$

9). Convert hexadecimal to binary equivalents

1. Convert the following Hexadecimal to binary equivalents:

(a) $AB948_{16}$

Solution:

$AB948_{16}$

= 1010 1011 1001 0100 1000

$= 10101011100101001000_2$

Hence the required binary equivalent is 10101011100101001000.

(b) $BA2.23C_{16}$

Solution:

$BA2.23E_{16}$

$= 1011\ 1010\ 0010\ .\ 0010\ 0011\ 1110_2$

= 101110100010.00100011111

Hence the required binary equivalent is 101110100010 . 00100011111

10). Convert Octal Number into Hexadecimal Number

Procedure:

Step1: Convert Octal number into 3-bit binary number.

Step2: Then make a group of 4 bit number

Step3: Each 4-bit number is replaced with hexadecimal equivalent

1. **Convert 1273_8 to hexadecimal Solution:**

   1273_8
   = 001 010 111 011 =0010 1011 1011 =$2BB_{16}$
   Hence 1273_8 = $2BB_{16}$
2. **Convert $(57.34)_8$ to hexadecimal**
   = 101 111 . 011 100
   = 0010 1111. 0111 0000
   = $2F.70_{16}$

11). Convert Hexadecimal into Number Octal Number

Procedure:
Step1: Write the 4-bit binary number for the Hexadecimal number
Step2: Form a Group of 3 bit number
Step2. Then write the equivalent Octal number

1. **Convert $A273_{16}$ to Octal Solution:**

   $A273_{16}$
   = 1010 0010 0111 0011
   = 001 010 001 001 110 011 =121163_8
   Hence $A273_{16}$ = 121163_8
2. **Convert $(ABCD.EF)_{16}$ to octal**
   = 1010 1011 1100 1101 . 1110 1111
   = 001 010 101 111 001 101 . 111 011 110
   = 125715.736_8
   Hence $ABCD.EF_{16}$ = 125715.736_8

Problem on Conversions

1a) Convert:

i).$(526.44)_8$=$()_2$=$()_{10}$

ii).$(48350)_{10}$=$()_{16}$=$()_8$ (4 Marks)

----

i) $(526.44)_8$ = $(?)_2$=$(?)_{10}$

$(526.44)_8$ = $(101,010,110.100, 100)_2$

Given NO.	5	2	6	4	4
Position	Digit-2	Digit-1	Digit-0	Digit(-1)	Digit(-2)
Weightage	8^2	8^1	8^0	8^{-1}	8^{-2}

$= 5 \times 8^2 + 2 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1} + 4 \times 8^{-2}$

= 320+16+6+0.5+0.0625

= $(342.5625)_{10}$

ii) $(48350)_{10}$= $(BCDE)_{16}$= $(1011,1100,1101,1110)_2$

(001, 011, 110, 011, 011, 110) $_2$= $(136336)_8$

---

2. Convert $(1101101)_2$= (?) $_{10}$ and $(69)_{10}$= (?)$_2$ (4 Marks)

Converting binary number into decimal is shown below.

$(1101101)_2$= ( ) $_{10}$

Given No.	1	1	0	1	1	0	1
Position	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Weightage	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$=1*2^6+1*2^5+0*2^4+1*2^3+1*2^2+0*2^1+1*2^0$

$=64+32+0+8+4+0+1=(109)_{10}$

Converting decimal to binary is shown below.

$(69)_{10}$ = ( )$_2$

```
2|69
2  34 -1
2  17 -0
2  8 -1
2  4 -0
2  2 -0
   1 -0          Read from bottom to top
```

$(69)_{10}$ = $(1000101)_2$

---

3. Covert $(1010111011110101)_2$= ( ) $_{16}$ and $(FA876)_{16}$= ( )$_2$ (4 Marks)

$(1010111011110101)_2$= ( ) $_{16}$

Form the groups of 4 bits from Right to the left, in case of in sufficient bits to form group add zeros

Grouping

$(1010, 1110, 1111, 0101) = (AEF5)_{16}$

Converting hexadecimal number into binary number $(FA876)_{16} = (\ )_2$

Write 4-bit equivalent binary number for each given hexadecimal digit

$(FA876)_{16} = (1111, 1010, 1000, 0111, 0110)_2 = (11111010100001110110)_2$

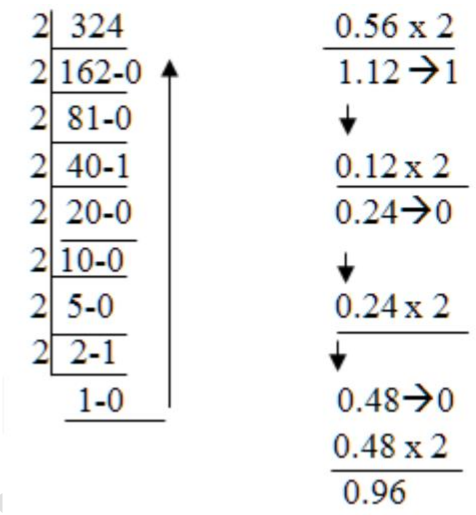
----

4). Convert: **(4 Marks)**

i)$(324.56)_{10}=(?)_2=(?)_8$ **ii)$(BCDE)_{16}=(?)_2=(?)_8$**

----

i).$(324.56)_{10}=(101000100.1000)_2 =(101,000,100.100,000)=(504.40)_8$



ii) $(BCDE)_{16} =(1011\ 1100\ 1101\ 1110)_2$ = 001,011,110,011,011,110=$(136336)_8$

----

5) a). $(1101101)_2 = (\ )_{10}$ b). $(96)_{10}=(\ )_2$ c) $(FA876)_{16}= (\ )_8$ d) d) $(237)_8= (\ )_{16}$

a). $(1101101)_2 = (\ )_{10}$ and

Given No.	1	1	0	1	1	0	1
Position	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Weightage	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$=1*2^6+1*2^5+0*2^4+1*2^3+1*2^2+0*2^1+1*2^0$

$=64+32+0+8+4+0+1= (109)_{10}$

b). $(96)_{10}=(\ )_2$

Converting decimal to binary is shown below.

$(96)_{10} = (\ )_2$

```
2 | 96
2 | 48 -0
2 | 24 -0
2 | 12 -0
2 | 6 -0
2 | 3 -0
    1 -1        Read from bottom to top
```

$(96)_{10} = (1100000)_2$

c) $(FA876)_{16} = (\ )_8$

$= (1111, 1010, 1000, 0111, 0110)_2$

$= (111, 110,101,000,011,101,110)_2$

$= (7650356)_8$

d) $(237)_8 = (\ )_{16}$

$= (010, 011, 111)_2$

$= (0000, 1001, 1111)_2$

$= (09F)_{16}$

Binary Addition

Half addition

The simple addition consists of four possible elementary operations namely as shown in table3.2.

Table 3.2: Addition table

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

In above cases the first combination A=0 and B=1 produces outputs sum and carry as zero. The next two cases produces output sum as one and carry as zero. The last case when A=1 and B=1 produces output sum as zero and carry as one.

Full addition

The operation which performs addition of three bits (two significant bits) and one carry (previous) is called a full addition. Truth table for the full addition is shown below in table 3.3.

Table 3.3: Full Addition table

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

Complement of Binary Numbers

1's Complement of Binary Number

In digital Computer arithmetic process is done not only with positive numbers; we also do it with negative number.

Processor manages the signed numbers and unsigned numbers. 1's Complement and 2's Complement are useful in this case.

1's Complement of binary number is an inversion of individual bit's i.e 1 to 0, 0 to 1. 1's complement of $(1010110)_2$ is

$=(0101001)_2$

2's Complement of Binary Number

The 2's complement of binary number is addition of 1 to the 1's complement of that binary number.

Eg. 2's complement of $(101011011)_2$

```
1's complement 010100100
                       1
----------------------------------
          (010100101)2
```

It also holds the binary decimal number.
Find the 2's complement of $(0.1110100)_2$
```
First 1's complement 0.0001011
                             1
------------------------------------------
```

(0.0001100)2

2’s Complement of Floating Point number
Find the 2’s Complement of (010111.1100)

```
Step1: 1’s complement 101000.0011
                                1
---------------------------------------------
                      101000.0100
```

Subtraction by 1’s Complement

In subtraction by 1’s complement we subtract two binary numbers using carried by 1’s complement.

The steps to be followed in subtraction by 1’s complement are:

Compute (m-n)2

i) Write down 1’s complement of the subtrahend (n).
ii) To add this with the minuend (m).
iii) If the result of addition generates a carry, then result is positive. and an 1 is added in the last bit (LSB).
iv) If carry is not generated, then the 1’s complement of the result of addition is obtained to get the final result and it is negative.

Evaluate:

(i) 110101 –

100101 Solution:

1’s complement of 10011 is 011010. Hence

```
                  Minuend -            110101
   1’s complement of subtrahend -      011010
                                     --------
                   Carry over -       1001111
                                            1
                                     --------

                                       010000
```

The required difference is 10000

(ii) 101011 – 111001

Solution:
m=101011   n=111001

1’s complement of n=111001 is 000110. Hence

```
                 Minuend (m) -            101011
       1’s complement of n -              000110
                                          ------
                                          110001
```

Hence the difference is – (0 0 1 1 1 0)2

(iii) 1011.001 –
110.10 Solution:
m= 1011.001 n=110.10

1’s complement of n= 0110.100 is 1001.011 Hence

```
                          Minuend (m) -            1011.001
     1’s complement of subtrahend (n) -            1001.011
                                                   --------
                     Carry generated -  1          0100.100
                                                          1
                                                   --------
                                                   0100.101
```

Hence the required difference is (0100.101)2

(iv) 10110.01 – 11010.10

Solution: m=10110.01
n= 11010.10

1’s complement of n= 11010.10 is 00101.01

```
          Minuend (m) =            10110.01
   1’s complement of n         0 0 1 0 1 . 0 1
          11011.10
```

Hence the required difference is – 00100.01 i.e. – 100.01

Subtraction by 2’s Complement

Subtraction by 2’s complement method we can easily subtract two binary numbers.

The operation is carried out by means of the following steps:

(i) At first, 2’s complement of the subtrahend **(n)** is found.
(ii) Then it is added to the minuend **(m)**.
(iii) If carry is generated, and it is dropped and the result is positive.
(iv) If there is no carry generated from addition, then the two’s complement of the sum will be the result and it is negative.

The following examples on subtraction by 2's complement will make the procedure clear:
Evaluate:
(i) 110110 -
10110 Solution:

The numbers of bits in the subtrahend is 5 while that of minuend is 6. We make the number of bits in the subtrahend equal to that of minuend by taking a `0' in the sixth place of the subtrahend.

Now, 2's complement of 010110 is (101101 + 1) i.e.101010. Adding this with the minuend.

```
            1 1   0110     Minuend
            1 0   1010     2's complement of subtrahend
  Carry     1 1    00000     Result of addition
```

After dropping the last carry bit we get the result of subtraction as 100000.

(ii) 10110 – 11010

Solution: m=10110, n=11010

2's complement of n= 11010 is (00101 + 1) i.e. 00110. Hence

```
                   Minuend (m) -          10110
  2's complement of subtrahend (n)-        00110
              Result of addition -        11100
```

As there is no carry over, the result of subtraction is negative and is obtained by writing the 2's complement of 11100 i.e.(00011 + 1) or 00100.

Hence the difference is – 00100.

(iii) 1010.11 – 1001.01

Solution: m=1010.11 n= 1001.01

2's complement of 1001.01 is 0110.11. Hence

```
                   Minuend (m) -          1010.11
2's complement of subtrahend (n) -        0110.11
                   Carry over          1  0001.10
```

After dropping the carry we get the result of subtraction as 1.10.

(iv) 10100.01 – 11011.10 Solution:

m=10100.01 n=11011.10

2's complement of 11011.10 is 00100.10. Hence

```
                    Minuend (m) -              10100.01
  2's complement of subtrahend (n) -           01100.10
                                            ____________
               Result of addition -            11000.11
```

As there is no carry over the result of subtraction is negative and is obtained by writing the 2's complement of 11000.11.

Hence the required result is – 00111.01.

Exam question and solutions

---

1).Subtract the following using 2's complement method:

i).$(101011)_2$ from $(111001)_2$

ii).$(111001)_2$ from $(101011)_2$ **(4 Marks)**

---

i). $(101011)_2$ from $(111001)_2$ = $(111001)_2$ – $(101011)_2$ m= $(111001)_2$ n= $(101011)_2$

```
1's comp n=            010100
+1 to LSB =                 1
--------------------------------------
2's comp of n =        010101
+m            =        111001
------------------------------------------
              =      1 001110
```

$\rightarrow$ carry is generated, neglect the carry & the result is positive.

Result $\rightarrow$ $(001110)_2$

ii). $(111001)_2$ from $(101011)_2$ = $(101011)_2$ – $(111001)_2$ m = $(101011)_2$ n= $(111001)_2$

```
1's comp of n=  000110
+1 to LSB=           1
--------------------------------------
2's comp of n = 000111
+m            = 101011
--------------------------------------
                = 110010 (R )
```

$\rightarrow$ no carry is generated, take 2's complement of R and result is negative

R $\rightarrow$ 110010

1's comp of R 001101

+1 to LSB $\rightarrow$ +1

2's comp of R $\rightarrow$ (001110)

Result = - $(001110)_2$

----

2). Perform the subtraction

$(11010)_2$ – $(10000)_2$ using 1's complement.

$(1000100)_2$ - $(1010100)_2$ using 2's complement. **(4 Marks)**

6b). $(11010)_2$ – $(10000)_2$

m= $(11010)_2$, n= $(10000)_2$

1's complement of n $\rightarrow$ 01111

+ m $\rightarrow$ 11010

1 01001(R) $\rightarrow$ carry is generated, add the carry to R, and the result is positive R $\rightarrow$

01001

+1

01010

Result= + $(1010)_2$

ii)$(1000100)_2$ - $(1010100)_2$

m= $(1000100)_2$, n= $(1010100)_2$

1's comp of n $\rightarrow$ 0101011

+1 to LSB $\rightarrow$ +1

2's comp of n $\rightarrow$ 0101100

+m $\rightarrow$ 1000100

1110000(R) $\rightarrow$ carry is not generated, take 1's complement of R and the result is positive.

R $\rightarrow$ 1110000

1's comp of R $\rightarrow$ 0001111

+1 to LSB $\rightarrow$ +1

2's comp of R $\rightarrow$ 10000

Result = - $(10000)_2$

Boolean Algebra Theorems:

In 1854 the person by name George Boole developed a systematic treatment of logic for algebraic sum.

Variables: These are the symbols used to represent Boolean function.
Constant: Logic 1 and 0 are called constants.
Literal: It is a variable or complement of a variable used in Boolean functions

1. **Law of Complementation:** The term complement means to invert, to change 1's to 0's and 0's to 1's. The following are the laws of complementation.

2. **OR laws:**
   - The OR operation of a Boolean variable with '0' or with itself doesn't change the value of the variable.

     A+0=A
     A+A=A
   - The OR operation with 1 or complement of the variable will make the variable 1

     A+1=1
     A+ =1

3. **AND laws:**
   - The AND operation of the Boolean variable with '0' or complement of the variable itself makes its value as '0'.

     A .0 =0
     A. =0
   - The AND operation of the Boolean variable with '1' or itself doesn't change the value of variable.

     A .A=A
     A.1=A

4. **Commutative law:** Changing the position of the variables in AND & OR operation doesn't change the value of the Boolean expression. Proof using method of induction or truth table method is shown below in table3.2.

i.e. **A.B=B.A**
A+B=B+A

Proof:

Table 3.2:Truth table Commutative law

A	B	A.B	B.A	A+B	B+A
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

5. **Associative law:** Changing the grouping of the variables on AND & OR operation doesn't change the value of the Boolean expression. Proof using method of induction or truth table method is shown below in table3.3.

i.e. **A.(B.C)=(A.B).C**
A+(B+C)=(A+B)+C

Proof:

Table 3.3:Truth table Associative law

A	B	C	B.C	A.(B.C)	A.B	(A.B).C	B+C	A+(B+C)	A+B	(A+B)+C
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	1	1	1	1
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1	1
1	1	0	0	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

6. **Distributive law**
Proof using method of induction or truth table method is shown below in table3.4.

A.(B+C) = (A.B) + (AC)

Proof:

Table 3.4:Truth table Distributive law

A	B	C	B+C	A.(B+C)	A.B	A.C	A.B+A.C
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+BC)= (A+B).(A+C)

Proof:

RHS = (A+B) (A+C)

= AA+AB+AC+BC

= A+AB+AC+BC $\rightarrow$ w.k t A.A=A

= A(1+B+C)+BC

= A.1+BC $\rightarrow$ w.k.t 1+B+C = 1

=A+BC=LHS

Proof:

A.(B+C)=AB+AC

RHS= (AB+AC)

A(B+C)=LHS

7. **Absorption law:**

A+AB=A

Proof:

LHS =A+AB

=A(1+B) $\rightarrow$ w.k.t. 1+B=1,

= A.1 =A = RHS

A.(A+B)=A

Proof:

LHS =A.(A+B)

= A.A +A.B $\rightarrow$ w.k.t. A.A=A

=A + A.B

=A(1 + B) $\rightarrow$ w.k.t. 1+B=1,

= A.1 =A = RHS

8. **Idempotent law**

A.A=A

A+A=A

9. **De Morgan's theorem**

De- Morgan's theorem of two variables

It states that any logical binary expressions remains unchanged if we

1. Change all variables to their complements
2. Change all AND operations to ORs
3. Change all OR operations to ANDs
4. Take the complement of the entire expressions

Truth table for De-Morgan's theorem of two variables is shown in table3.5 below

Theorem 1: The complement of AND is equal to the OR of the complements

i.e.

Theorem 2: The complement of OR is equal to the AND of the complements

i.e.

Proof:

Table 3.5. Truth table for De-Morgan's theorem for two variables

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	**B**			**A.B**	**A+B**				
0	0	1	1	0	0	1	1	1	1
0	1	1	0	0	1	1	0	0	1
1	0	0	1	0	1	1	0	0	1
1	1	0	0	1	1	0	0	0	0

By observing coloum 7 and 10 theorem1 is verified.

i.e.

By observing coloum 8 and 9 theorem2 is verified

i.e.

DeMorgan's theorem of Three Variables

Theorem1: The complement of AND is equal to the OR of the complements

i.e.

Theorem2: The complement of OR is equal to the AND of the complements

i.e.

Truth table for De-Morgan's theorem of three variables is shown in table3.6 below

Proof:

Table 3.6. Truth table of DeMorgan's theorem of three variables

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
A	**B**	**C**				**A.B.C**	**A+B+C**				
0	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
1	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
1	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
1	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
1	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**

By observing coloum 9 and 12 theorem1 is verified.

i.e.

By observing coloum 10 and 11 theorem2 is verified i.e.

--

State and Prove De-Morgan's theorem for four variables

--

It states that any logical binary expressions remains unchanged if we

5. Change all variables to their complements
6. Change all AND operations to ORs
7. Change all OR operations to ANDs
8. Take the complement of the entire expressions

Theorem1: The Products of complement is equal to the sum of the complements

i.e. +

Theorem2: The sum of complement of is equal to the products of the complements

i.e.

The truth table of DeMorgan's theorem of four variables is shown in table 3.7

<u>Proof:</u>

Table 3.7. DeMorgan's theorem of four variables

1	**2**	**3**	**4**	**5**	**6**	**7**	8	**9**	**10**	**11**	**12**	**13**	**14**
A	**B**	**C**	**D**					**A.B.C.D**	**A+B+C+D**				+
0	0	0	0	1	1	1	1	0	0	1	1	1	1
0	0	0	1	1	1	1	0	0	1	1	0	0	1
0	0	1	0	1	1	0	1	0	1	1	0	0	1
0	0	1	1	1	1	0	0	0	1	1	0	0	1
0	1	0	0	1	0	1	1	0	1	1	0	0	1
0	1	0	1	1	0	1	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1	0	1	1	0	0	1
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	1	1	1	0	1	1	0	0	1
1	0	0	1	0	1	1	0	0	1	1	0	0	1
1	0	1	0	0	1	0	1	0	1	1	0	0	1
1	0	1	1	0	1	0	0	0	1	1	0	0	1
1	1	0	0	0	0	1	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0	1	1	0	0	1
1	1	1	0	0	0	0	1	0	1	1	0	0	1
1	1	1	1	0	0	0	0	1	1	0	0	0	0

By observing column 11 and 14 it is proved that

+

By observing column 12 and 13 it is proved that

i.e.

Identities of Boolean Algebra:

Boolean laws are shown in table3.8

Table 3.8. Table for Boolean Laws

Sl. No.	Name	AND form	OR form
1.	Identity law	1.A=A	0+A=A
2.	Null law	0.A=0	1+A=1
3.	Idempotent law	A.A=A	A+A=A
4.	Inverse law	A.	A+ =1
5.	Commutative law	A.B = B.A	A+B = B+A
6.	Associative law	A.(B.C) = (A.B).C	A+(B+C)=(A+B)+C
7.	Distributive law	A+ (B.C) = (A+B). (A+C)	A.(B+C)=A.B + A.C
8.	Absorption law	A.(A+B)=A	A+A.B=A
9.	De-Morgan's law		

Logic Gates:

In Digital electronics we have seven logic gates. they are classified into three groups namely i)Basic gates
ii) Universal gates
iii)Derived gates

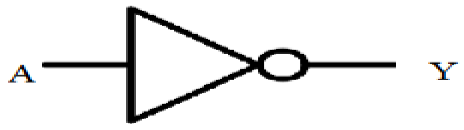
Basic gates

The three basic gates used in digital electronics are namely
1)NOT gate
2)AND gate
3)OR gate

1) Not Gate

Logic NOT gates provide the complement of their input signal and are so called because when their input signal is "HIGH" their output state will *NOT* be "HIGH" and output will be complement of the input hence they are also called as inverters.

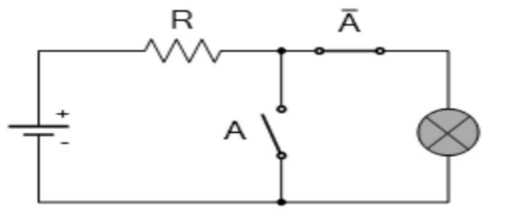
Symbol **Truth Table**



A	Y
0	1
1	0

The symbol and truth table of NOT gate is as shown above.
The expression of NOT gate is

For the “LOW” input signal their output state will **NOT** be “LOW”. The “bubble” (o) present at the end of the NOT gate symbol above denotes a signal inversion (complementation) of the output signal.



Switch A - Open = "0", Lamp - ON = "1"
Switch A - Closed = "1", Lamp - OFF = "0"

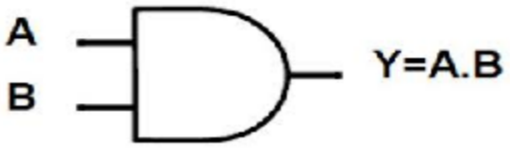
Fig 3.11. Switching Circuit of NOT gate

When switch A is open (0) then Lamp is ON (1), on other hand hand When switch A is closed(1) then Lamp is OFF (0). The switching circuit of NOT gate isshown in figure 3.11.

2).AND Gate

Working: The output of the AND gate will be HIGH if and only if both the input are HIGH. When anyone of inputs is LOW the output will be LOW.

Symbol **Truth Table**



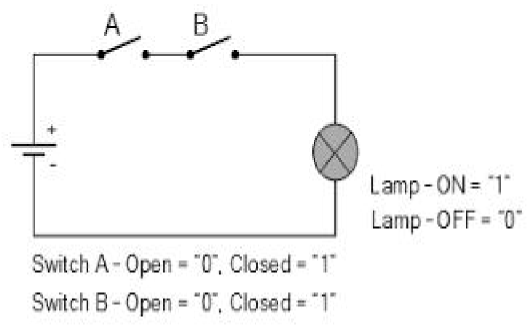
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

The symbol and truth table of AND gate is as shown above.
The expression of AND gate is

Equivalent circuit representation

The AND gate can be represented by circuit contain two switches as shown in figure 3.12.

A and B are two swittches connected in series with a lamp and battery as shown in figure. Each switch can be regarded as one input of logic gate. The status of lamp is the output. When lamp glows the ouput is considered to be HIGH. The lamp glows only when the both switches are closed.


A
B
+
-
Lamp - ON = "1"
Lamp - OFF = "0"
Switch A - Open = "0", Closed = "1"
Switch B - Open = "0", Closed = "1"

A	B	Lamp
Open	Open	OFF (LOW)
Open	Closed	OFF (LOW)
Closed	Open	OFF (LOW)
Closed	Closed	ON (LOW)

Fig. 3.12. Switching Circuit of AND gate and truth table

Diode eqivalent circuit of the AND gate is shown in figure 3.13 below. It uses two diodes D1, D2. Equivalent output for all the four combination of the input is shown in figure 3.14.

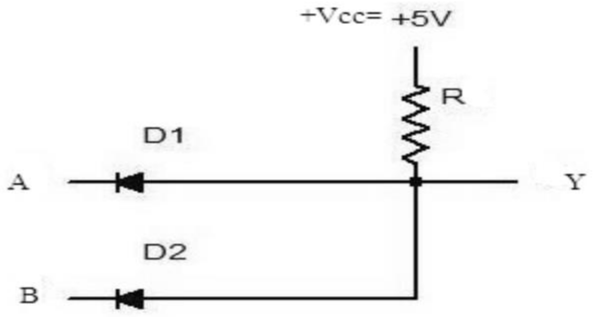


Fig. 3.13. Diode equivalent circuit of AND gate

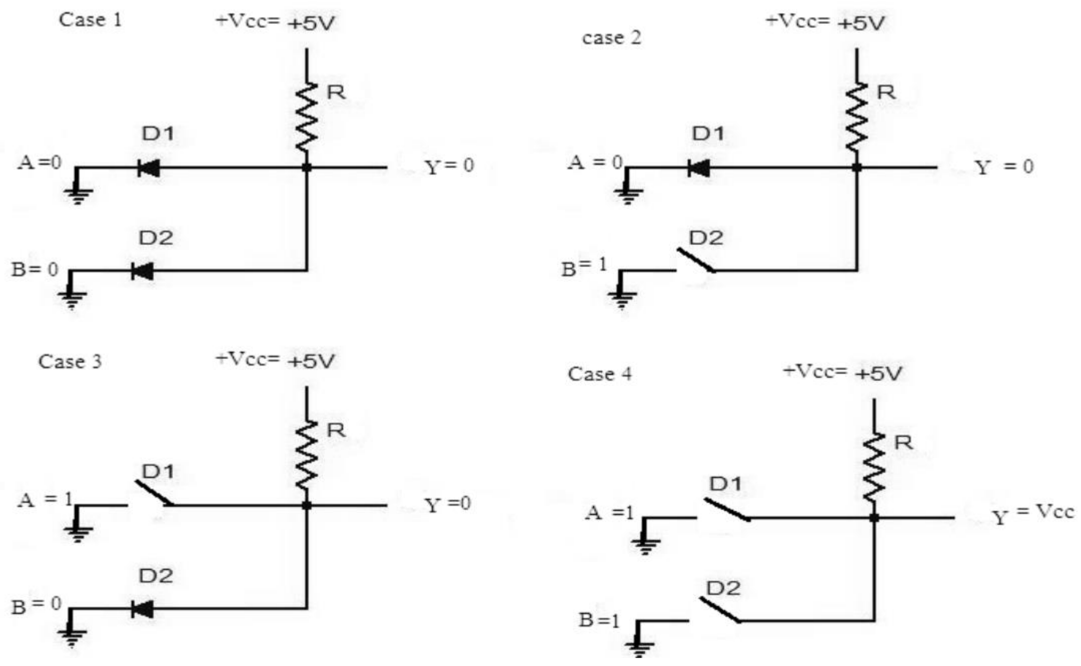


Fig. 3.14. Diode equivalent circuit combination for AND gate

When $V_A>V_K$ (Anode voltage is greater than Cathode voltage) the diode is forward bised and start conducting replace diode by short circuite.

When $V_A<V_K$ (Cathode voltage is greater than Anode voltage) the diode is reverse replace diode by open circuite.

Case 1. When A=0, B=0 : $V_A>V_K$ condition satisfied for the both diodes and output Y =0.

Case 2. When A=0, B=1: $V_A>V_K$ condition satisfied for the diode D1 output Y=0.

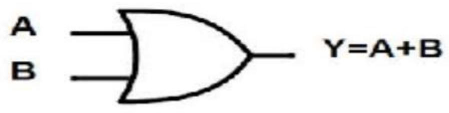
Case 3. When A=1, B=0 : $V_A>V_K$ condition satisfied for the diode D2 output Y =0.

Case 4. When A=1, B=1: D1 and D2 both are reverse biased output Y =1

OR Gate

Working: The output of the OR gate will be LOW if and only if both the input are LOW. When anyone of inputs is HIGH the output will be HIGH.

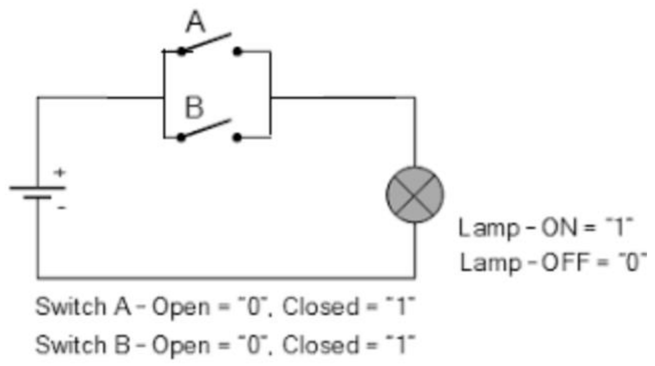
Symbol **Truth Table**



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

The symbol and truth table of OR gate is as shown above.
The expression of OR gate is

Equivalent circuit representation


A
B
+
-
Lamp - ON = "1"
Lamp - OFF = "0"
Switch A - Open = "0", Closed = "1"
Switch B - Open = "0", Closed = "1"

A	B	Lamp
Open	Open	OFF (LOW)
Open	Closed	ON (HIGH)
Closed	Open	ON (HIGH)
Closed	Closed	ON (HIGH)

Fig. 3.15. Switching Circuit of OR gate and truth table

the OR gate can be represented by circuit containing two switches in parallel combination as shown in figure 3.15.

A and B are two swittches connected in parallel with a lamp and battery as shown in figure. Each switch can be regarded as one input of logic gate. The status of lamp is the output. When lamp glows the ouput is considered to be HIGH. The lamp glows when any one of the switch is closed and also when both the switches closed.

Diode eqivalent circuit of the OR gate is shown in figure 3.15 . It uses two diodes D1, D2. Equivalent output for all the four combination of the input is shown in figure 3.16.

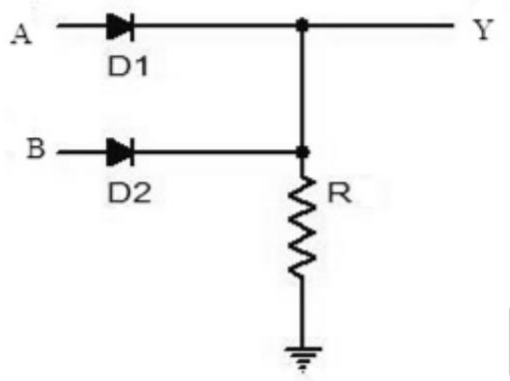


Fig. 3.16.Diode equivalent circuit of AND gate

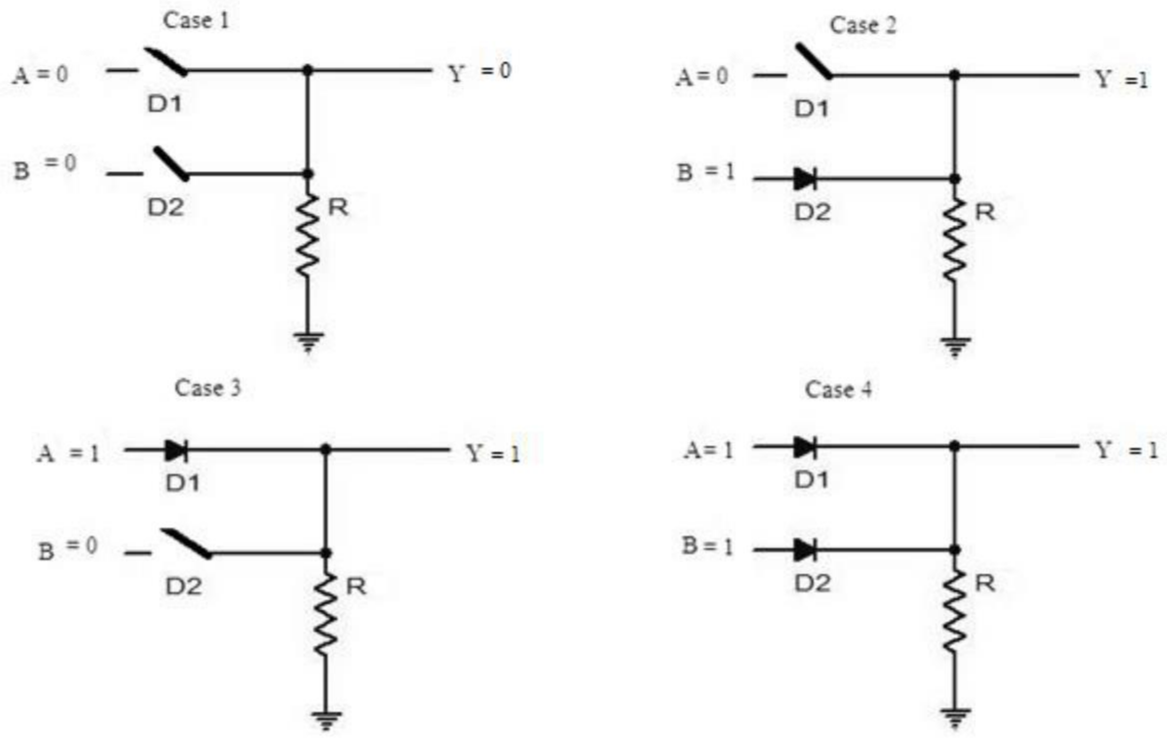


Fig.3.17. Diode equivalent circuit combination for AND gate

When $V_A > V_K$ (Anode voltage is greater than Cathode voltage) the diode is forward bised and start conducting replace diode by short circuite.

When $V_A < V_K$ (Cathode voltage is greater than Anode voltage) the diode is reverse replace diode by open circuite.

Case 1. When A=0, B=0: D1 and D2 both are reverse biased output Y=0.

Case 2. When A=0, B=1: Diode D2 forward biased output Y =1

Case 3. When A=1, B=0: Diode D1 forward biased output Y =1

Case 4. When A=1, B=1: D1 and D2 both are forward biased output Y =1

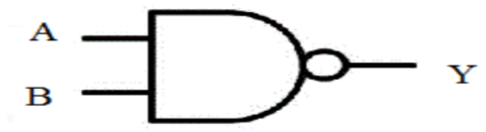
Universal gates:

There are two universal gates namely NAND and NOR gate. Any Boolean expression can be realized by using only NAND gate and by using only NOR gate. Hence they are called universal gates.

Write the symbol, truth table and final expression for NAND.

NAND Gate: The output of the NAND gate will be LOW when both the inputs are HIGH. otherwise the output of the NAND will be HIGH.

Symbol **Truth table**



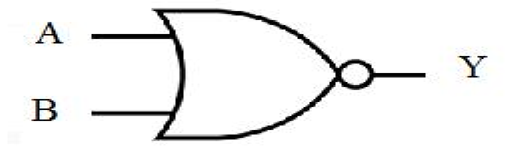
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

The symbol and truth table of NAND gate is as shown above.

The expression of NAND gate is

NOR Gate: The output of the NOR gate will be High when all the inputs are LOW. otherwise the output will be LOW.

Symbol **Truth Table**



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

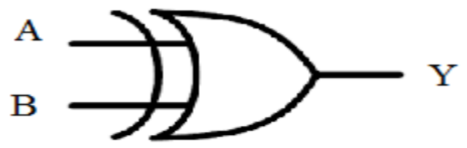
The symbol and truth table of NOR gate is as shown above.
The expression of NOR gate is

Derived gates

There are two derived gates namely XOR and XNOR. These gates can be derived using basic gates, hence the name derived gate.

XOR Gate: The output of XOR gate is HIGH when the inputs are different. when the inputs are same the output is LOW.

Symbol **Truth Table**

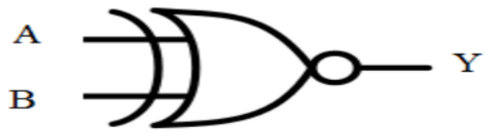


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

The truth table and symbol of Ex-OR gate is shown above. The expression foe Ex-OR gate is given by
The realization of XOR gate using basic gates, NAND gates and NOR gates is explained in next section.

XNOR Gate: The output of XNOR gate is HIGH when the inputs are same. when the inputs are different the output is LOW.

Symbol **Truth Table**

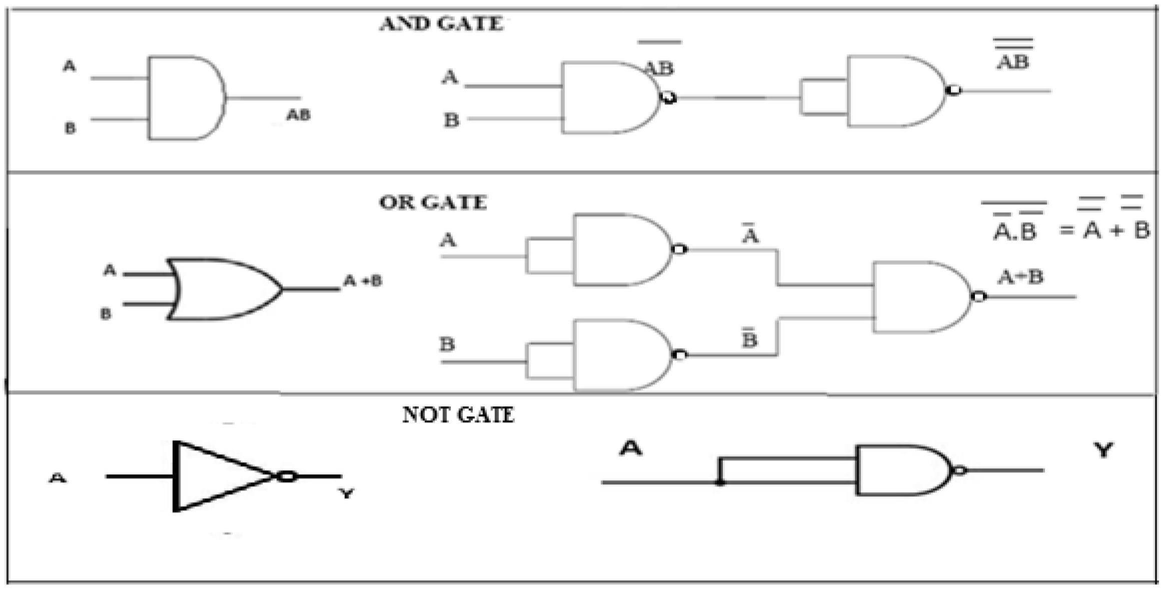


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

The Symbol and truth table of Ex-NOR gate is shown above.
The expression foe Ex-NOR gate is given by
The realization of XOR gate using basic gates, NAND gates and NOR gates is explained in next section.

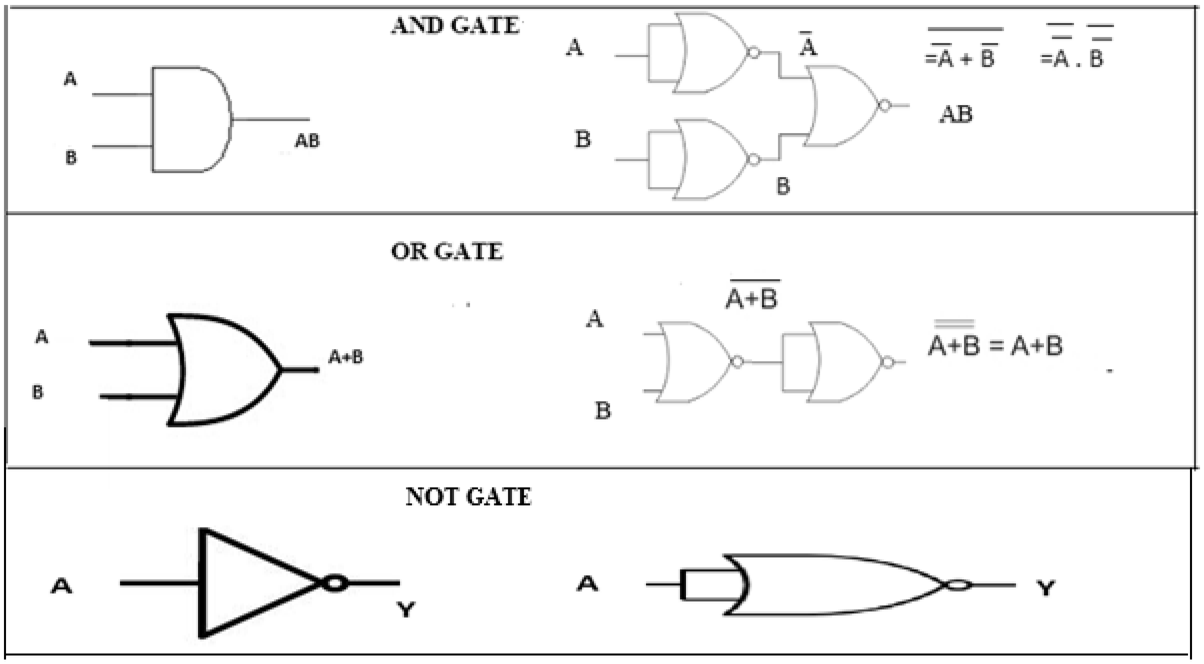
Algebraic Simplification, NAND and NOR Implementation:

1). NAND Implementation:



AND GATE
A
B
AB
A
B
$\overline{AB}$
$\overline{\overline{AB}}$
OR GATE
A
B
A+B
A
B
$\bar{A}$
$\bar{B}$
$\overline{\bar{A}.\bar{B}} = \bar{\bar{A}} + \bar{\bar{B}}$
A+B
NOT GATE
A
Y
A
Y

Fig.3.18. Realize AND, OR & NOT gates using NAND gates 2). NOR Implementation:



AND GATE
A
B
AB
A
B
$\bar{A}$
B
$=\overline{\bar{A} + \bar{B}}$
$=\bar{\bar{A}} . \bar{\bar{B}}$
AB
OR GATE
A
B
A+B
A
B
$\overline{A+B}$
$\overline{\overline{A+B}} = A+B$
NOT GATE
A
Y
A
Y

Fig. 3.19. Realize AND, OR & NOT gates using NOR gate

The figure 3.18 and 3.19 shows the NAND and NOR implementation of the basic gates

----

1). Realize XOR gate using

1. basic gate
2. using only NAND gate
3. using only NOR gate

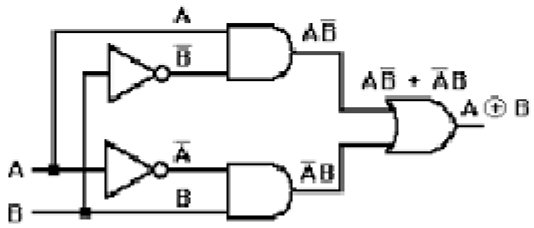
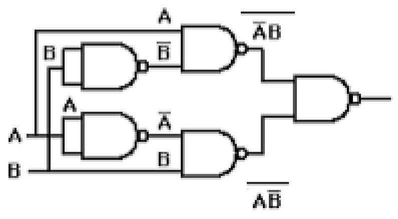


Fig. 3.20 Basic gates

To Implement using basic gates we need two NOT gate, two AND gate and one OR gate. the Circuit is shown above



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

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

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

$$= A \oplus B$$

Fig. 3.21 NAND gates

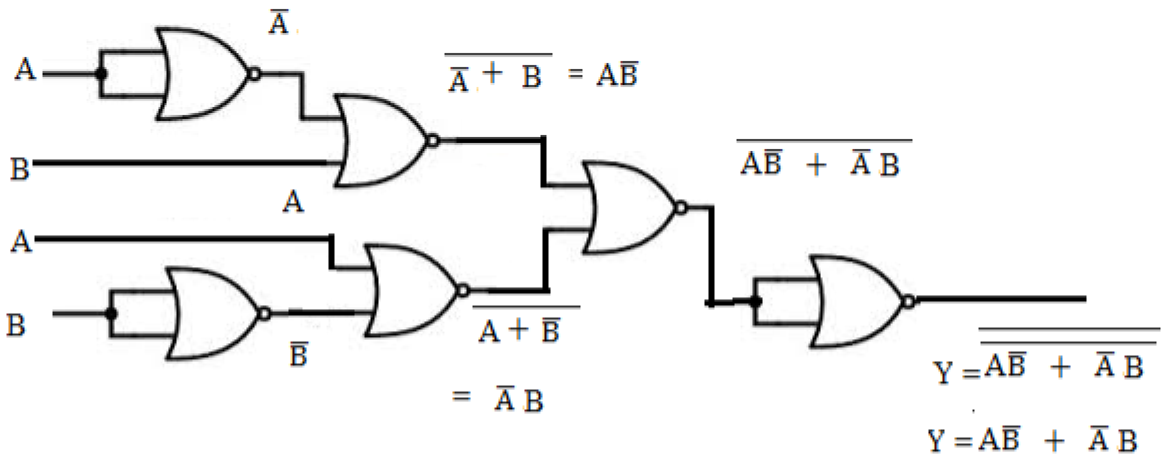


Fig. 3.22 NOR gates

The implementation using basic gates, NAND gate and NOR gates are shown in figure 3.20, 3,21 and 3.22.

2). Realize XNOR gate using

1. basic gate
2. using only NAND gate
3. using only NOR gate

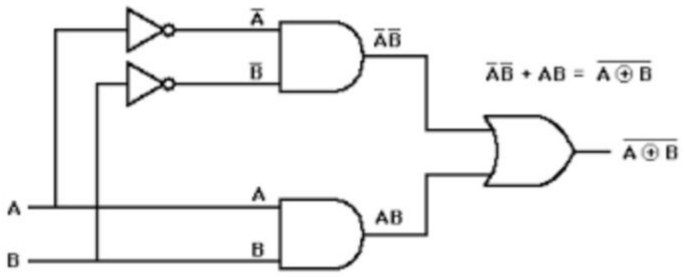


Fig. 3.23 Basic gates

To Implement using basic gates we need two NOT gate, two AND gate and one OR gate. the Circuit is shown above

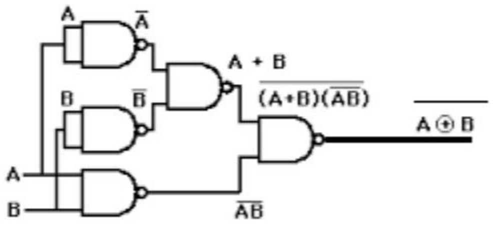
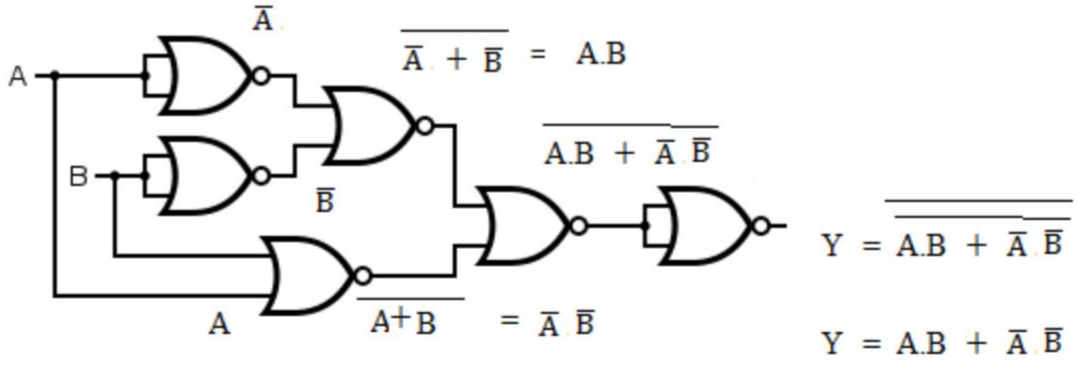


Fig. 3.24 NAND gates



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

Fig. 3.25 NOR gates

The implementation using basic gates, NAND gate and NOR gates are shown in figure 3.23, 3,24 and 3.25.

Simplification of Boolean algebra Theorem

1). Factorize the following Boolean equations

(4 Marks)

= we know that (A+ =1

=B.1 B.1=B

=B

= (B.C + B.B=B , C.C=C

= (B.C + + CA) (We Know That

----

2). **Simplify the following expression and realize using basic gates:**

(4 Marks)

=

=

=

=

= (1+C) +

= +

After simplification realizing using basic gates is shown in figure 3.26.

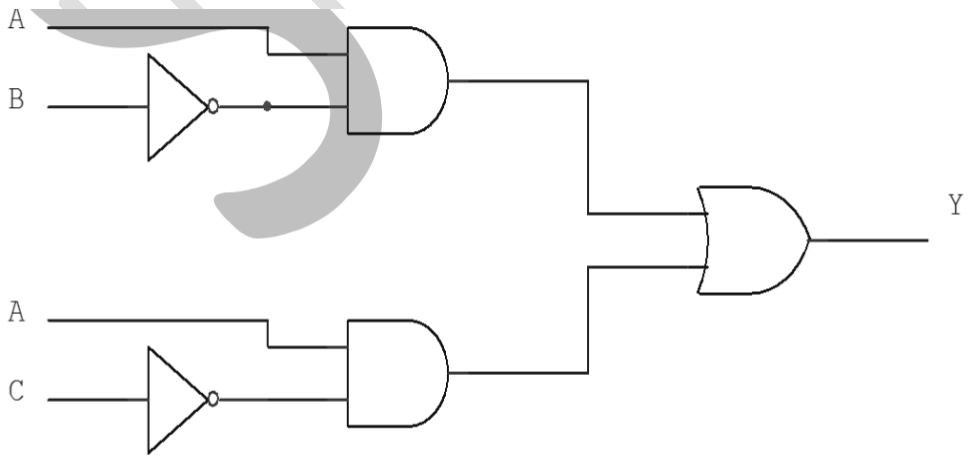


Fig.3.26 Realization using basic gates

----

3). Simplify and realize the following expressions using only NAND and NOR Gates. (DEC-2015)

i)

ii) Y= AB + AC +BD +CD

----

i)

= (

= + , here =0, =0,

= +

=

=

After simplification we need two AND gates,(in that one three input and one two input gate) one OR gate and two NOT gates. Realization using basic gates is shown in figure 3.27a. Afterword the realization of the basic gates can be implemented using only NAND and NOR gate as shown in figure 3.27b and figure 3.27c respectively.

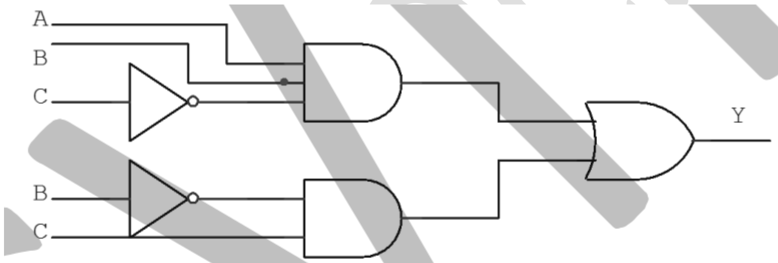


Fig. 3.27a. Realization using basic gates gates

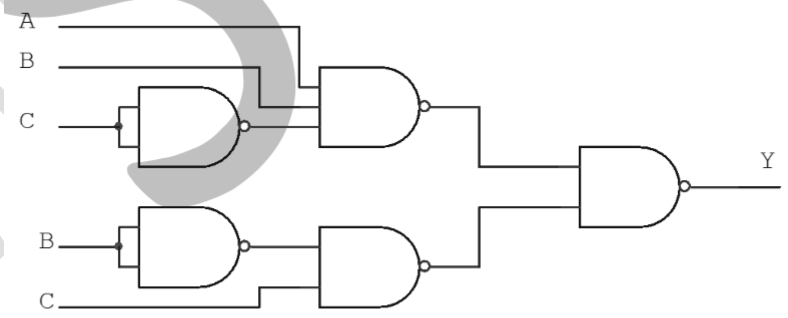


Fig. 3.27b. Realization using NAND gates

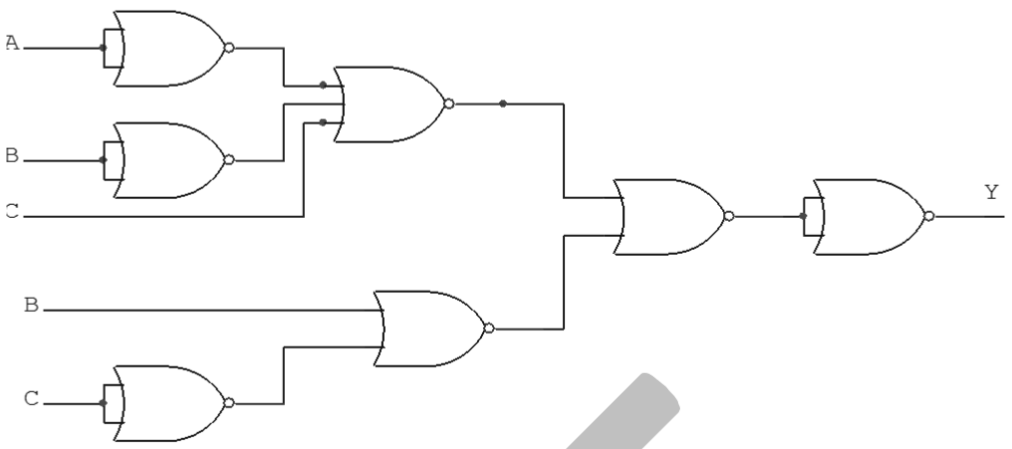


Fig.3.27c. Realization using NOR gates

ii) Y= AB + AC +BD +CD

=A(B+C) + D(B+C)

=(B+C)(A+D)

After simplification, the Realization of the simplified expression using basic gates is shown in figure 3.28a. the realizaton of the expression using NAND and NOR gates is shown in figure 3.28b and figure 3.28c respectively.

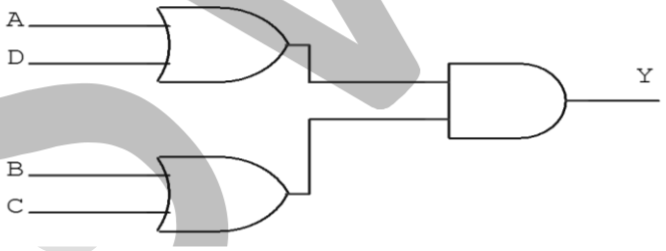


Fig.3.28a. Realization using basic gates

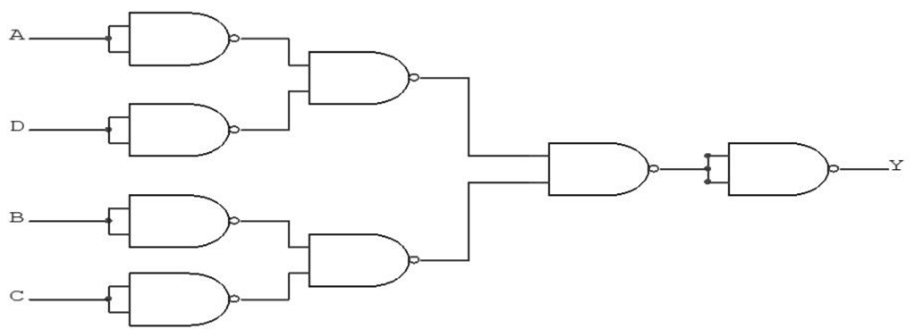


Fig. 3.28b. Realization using NAND gates

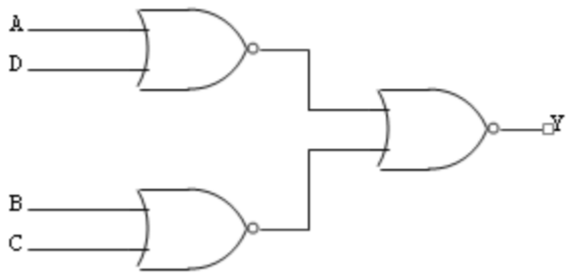


Fig.3.28c. Realization using NOR gates

---

4). Simplification of the following expression **(June-2016)**

=

=

=

=

= (1+C) +

= +

After simplification realizing using basic gates is shown in figure 3.29

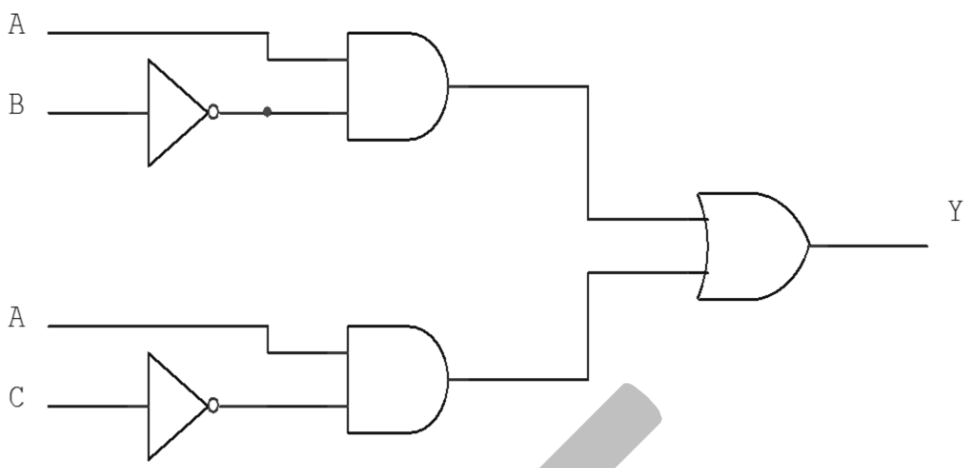


Fig.3.29 Realization using basic gates

Adders:

These are logical circuits used to add binary numbers and used in digital computers

Half adder

Realize half adder **(6 Marks)**

Half adder is a logic circuit that performs addition of two binary bits and provides the result of two binary bits.

- One bit of result indicates the "sum" and the other represents "carry";
- The block diagram of half adder is as shown in figure 3.30.

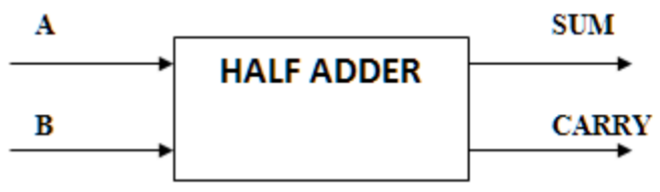


Fig.3.30. Logic symbol of Half adder circuit

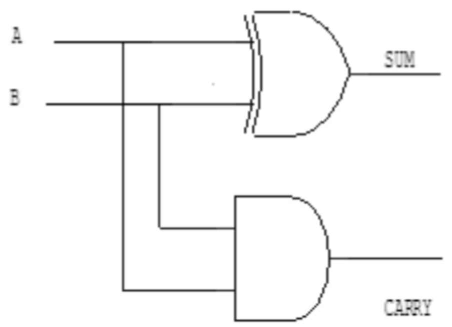
- The truth table of half adder is given below table 3.9

Table 3.9. truth table of half adder

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- By observing the truth table we write that
  CARRY=1 for A=1 & B=1, therefore CARRY=A.B and gate logic.
- While SUM=1 for A=0,B=1 & A=1,B=0, therefore SUM=
- Realization of sum and carry of half adder circuit using basic gates is shown in figure 3.30b. and Nand gate realization in figure 3.30c.

- For the above sum expression, two NAND inverters and three-two input NAND gates necessary to implement the sum circuit and for carry we require one NAND inverters and one-two input NAND gates as shown in figure 3.30b.



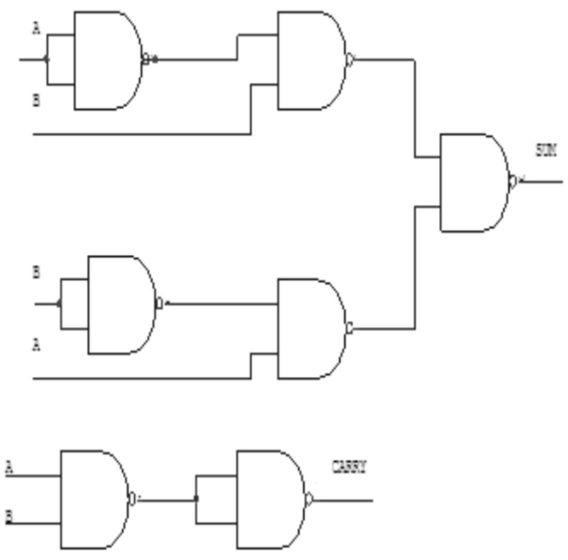


Fig. 3.30 b. Half adder circuit using basic gates Fig. 3.30c. Nand gate realization of half adder

- **Half adder using only NAND gate realization**
  Apply DeMorgan's theorem two times and simplify to realise using NAND gates

=

Y =

Half adder using only NOR gate realization

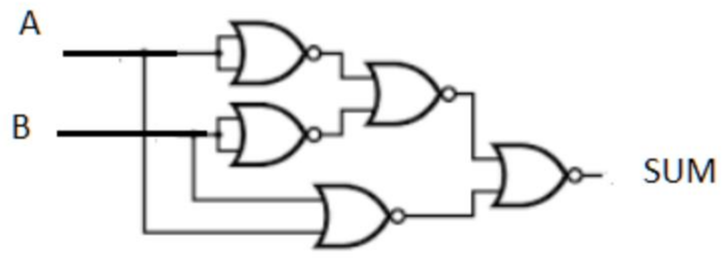
=

Y=

Y=

Y=

Apply DeMorgan's theorem two times and simplify to realise using NOR gates



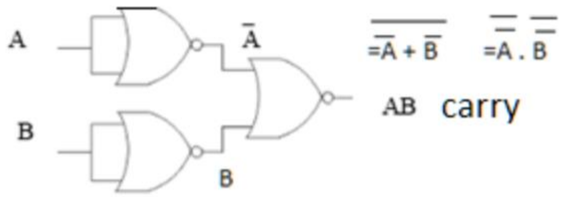

A
B
Ā
B
=$\overline{\overline{A}+\overline{B}}$
=$\overline{\overline{A}}.\overline{\overline{B}}$
AB carry

Fig. 3.30d. NOR Implementation of half adder

For the above sum expression, two NOR inverters and three-two input NOR gates necessary to implement the sum circuit and for carry we require two NOR inverters and one-two input NOR gates as shown in figure 3.30d.

Full adder

Write a note on full adder **(6 Marks)**

- The full adder is a logic circuit that performs the addition of three binary bits to provide two binary output bits of results.
- One bit of result is sum and other is carry.
- Block diagram of full adder is given below in figure 3.31a.

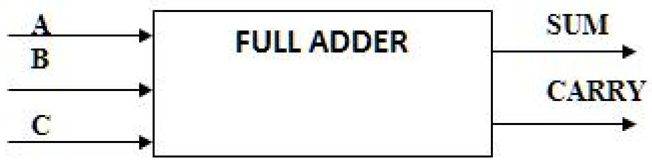


Fig. 3.31a Logic symbol of Full adder

- The truth table of full adder is given below in table 3.10

Table3.10. Truth table of Full adder

A	B	C	SUM	CARRY
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

- From the truth table
  By observing the sum column we can write Product of sum terms as below.

SUM=
  = +
  = (B XOR C)+A(B XNOR C)
  = (A XOR B XOR C)

By observing the sum column we can write Product of sum terms as below.
CARRY= (
  = (
  = BC (
  = AB+BC+AC

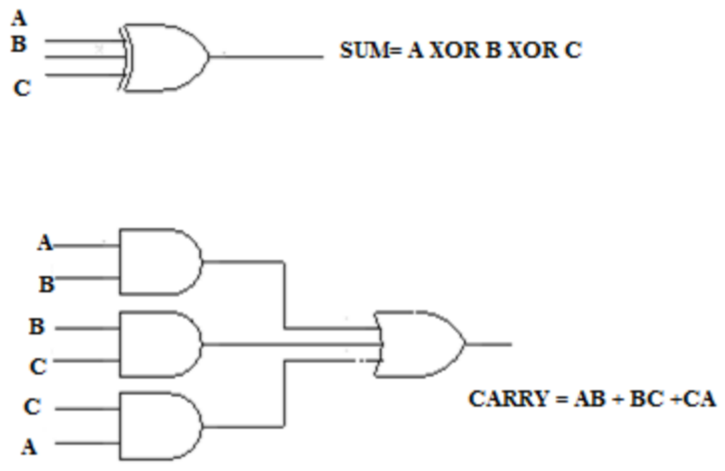


Fig. 3.31b. Logic Diagram of Full adder using gates

Realize the circuit for sum and carry of full adder using basic gates The basic gate implementation is shown in figure 3.31c

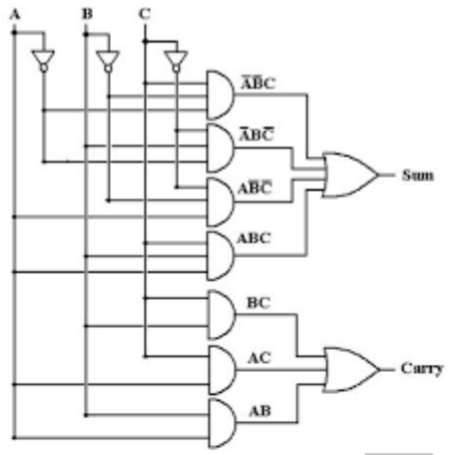


Fig.3.31c. Full Adder using basic gates

Full adder using half adder

Realize Full adder using half adder. Write the truth table, necessary circuits.
The full adder can be realized using two half adder and one OR gates. The circuit used for the realization is shown in figure 3.32. The truth table of Full adder is in Table 3.11.

Table 3.11. The truth table for the full adder

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

The full adder can be obtained by combining two full adders. The half adders are cascaded as shown in figure 3.32. The carry output of the two half adder are ORed to get the final carry.

SUM=S1$\oplus$ C

SUM=A$\oplus$B$\oplus$C

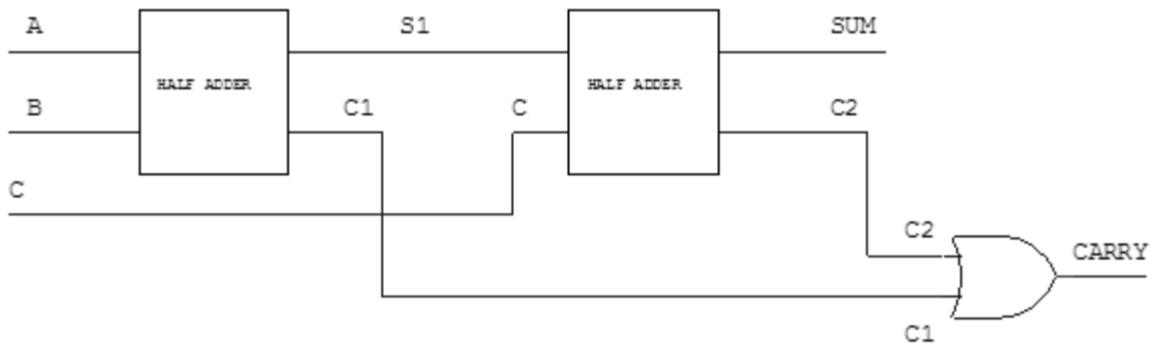


Fig.3.32. Full adder using two half adder

C1=AB

C2=C.( )

CARRY= C1+C2

CARRY== AB+S1.C

CARRY=AB+ C. ( )

CARRY= + C. ( ) Because (C+)=1

CARRY=

Hence the sum and carry expressions are same we can conclude that Full adder can be realized using two half adder and one OR gate. figure 5c and 5d respectively.

And CARRY is given by CARRY= C1 + C2