MODULE 5

Module – 5 Digital Electronics Fundamentals & Communication Systems

SYLLABUS

1. ***Number systems***

a. ***Difference between analog and digital signals,***

b. ***Number System-Binary, Hexadecimal,***

c. ***Conversion-Decimal to binary, Hexadecimal to decimal and vice-versa,***

2. ***Boolean algebra,***

a. *Basic and Universal Gates,*

3. ***Sequential and combinatorial circuits***

a. *Half and Full adder, Multiplexer,*

b. *Decoder,*

c. *SR and JK flip-flops,*

d. *Shift register,*

e. *3 bit Ripple Counter (10.1-10.7 of Text 1).*

4. ***Communication system,***

a. *Basic Communication system,*

b. *Principle of operations of Mobile phone (18.2 and 18.18 of Text 1).*

TEXTBOOKS

1. ***"Basic Electronics", D.P. Kothari, I. J. Nagrath, MHE (India) Private Limited, 2014.***

2. ***"Electronic Devices", Thomas L Floyd, Pearson Education, 9th edition 2012***

NOTES

Introduction to Number systems

- Digital Electronics is a branch of electronics that deals with zeros and ones.
- A digital circuit responds to only two states of input namely a HIGH and a LOW. A HIGH state corresponds to logic 1 and a LOW state corresponds to logic 0.
- Logic gates are the basic building blocks of digital circuits and are designed to obey a special algebra called Boolean algebra.
- A digital computer is basically a digital system which comprises of digital circuits.
- Digital signals or numbers, are processed by means of digital systems using the concept of binary numbers and Boolean algebra.
- In a digital computer, the numbers are coded in the form of binary (ON/OFF) electrical pulses and processed by means of logic gates and memory cells.
- Binary signals are used extensively in communication, control and instrumentation systems as well in computers.
- Binary signals have the great advantage of being far less susceptible to disturbance (noise) compared to analog signals.

Analog signal

- An analog signal is a continuous wave that changes over a time period.
- An analog signal is represented by a sine wave as shown in fig.5.1.
- An analog signal is described by the amplitude, period or frequency, and phase.
- An analog signal transmits data in the form of a wave.

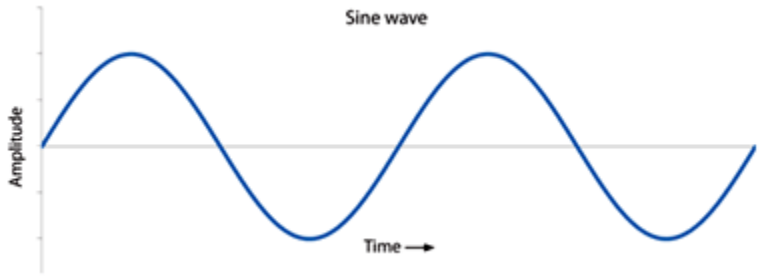


Fig. 5.1: Sinusoidal signal

Digital signal

- A digital signal is a discrete wave that carries information in binary form.
- A digital signal is represented by square waves as shown in fig.5.2.
- A digital signal carries data in the binary form i.e. 0 and 1.

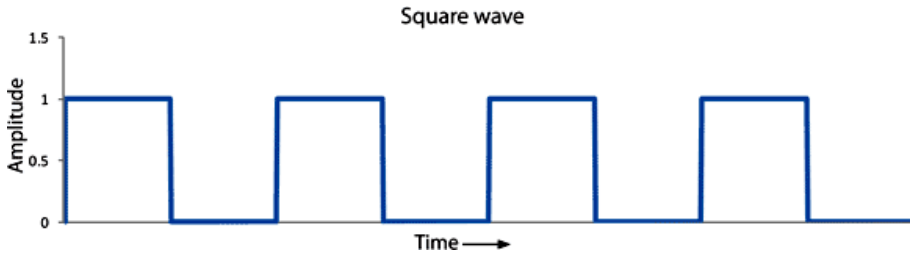


Fig.5.2: Square wave

Advantages of Digital Systems:

i. Easy to design
ii. Flexible in design
iii. Provides higher accuracy
iv. Less affected by noise
v. Error correction and detection is possible
vi. Cost effective

Number System

There are four number systems of arithmetic that are used in the digital systems

1. Decimal
2. Binary
3. Octal
4. Hexadecimal

Polynomial Notation (Series Representation)

Any number system can be represented by the following polynomial.

$$\mathbf{a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + \cdots + a_1r^1 + a_0r^0 + a_{-1}r^{-1} + \cdots + a_{-m}r^{-m}}$$

r = radix or base

m = number of fractional digits to the right of the radix point

n = number of integer digits to the left of the radix point

a_{n-1} = most significant digit

a_{-m} = least significant digit

Decimal number system

- The decimal number system is composed of 10 digits. The 10 digits of decimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- The decimal number system is also called as base 10 system because it has 10 possible digits.

Binary number system

- The binary number system is composed of 2 digits. The 2 digits of binary number system are 0 and 1.
- The binary number system is also called as base 2 system because it has 2 possible digits.

Octal Number System

- The octal number system is composed of 8 digits. The 8 digits of octal number system are 0, 1, 2, 3, 4, 5, 6, 7
- The octal number system is also called as base 8 system because it has 8 possible digits.

Hexadecimal number system

- The hexadecimal number system is composed of 16 digits. The 16 digits of binary number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- The hexadecimal number system is also called as base 16 system because it has 16 possible digits.

Conversion of number systems

- Converting from one number system to another is called conversion of number system or code conversion, like converting from binary to decimal or converting from hexadecimal to decimal etc.
- The possible conversions of number system are shown fig.3.3.

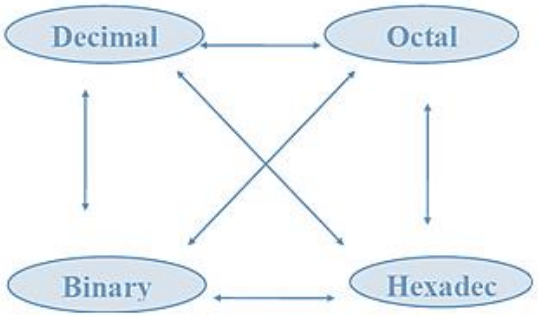


Fig.3.3: Possible number conversions

Conversion from Decimal to Binary number system

Conversion of decimal number to binary number system is carried two in steps

i. Conversion of integer part is done by successive division
ii. Conversion of fractional part is done by successive multiplication

Conversion of integer part

- Divide the integer part of decimal number by 2 until quotient is one.
- Remainders of each division form the digits of binary number
- Remainders are taken in the reverse order (bottom to top) to form a binary number

Conversion of fractional part

- Multiply the fractional part of decimal number by 2
- The integer part of the product forms the digits of binary number
- The fractional part is again multiplied by 2 and this process is repeated until fractional part becomes zero or sufficient digits are obtained.
- The integer part are read downwards (top to bottom) to form a binary number

Convert $(109)_{10}$ to Binary

		Remainder
2	109	
2	54	1
2	27	0
2	13	1
2	6	1
2	3	0
	1	1

Therefore $(109)_{10} = (1101101)_2$

Convert $(341)_{10}$ to Binary

		Remainder
2	341	
2	170	1
2	85	0
2	42	1
2	21	0
2	10	1
2	5	0
2	2	1
	1	0

Therefore (341)10 = (101010101)2

Convert $(5.25)_{10}$ to Binary

		Remainder
2	5	
2	2	1
	1	0

Integer

0.25 * 2 = 0.5 0

0. 5 * 2 = 1.0 1

Therefore $(5.25)_{10} = (101.01)_2$

Convert $(6.8125)_{10}$ to Binary

		Remainder
2	6	
2	3	0
	1	1

Integer

0.8125 * 2 = 1.625 1

0.625 * 2 = 1.25 1

0.25 * 2 = 0.5 0

0.5 * 2 = 1.0 1

Therefore $(6.8125)_{10} = (110.1101)_2$

Conversion from Binary to Decimal number system

- Decimal equivalent of a given binary number is obtained by multiplying each digit of the binary number with the radix to the power of position value and add each multiplication result

Convert $(1101101)_2$ to Decimal

$(1101101)_2 = 1*2^6 + 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

$(1101101)_2 = 64 + 32 + 0 + 8 + 4 + 0 + 1$

$(1101101)_2 = (109)_{10}$

Convert $(101010101)_2$ to Decimal

$(101010101)_2 = 1*2^8 + 0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

$(101010101)_2 = 256 + 0 + 64 + 0 + 16 + 0 + 4 + 0 + 1$

$(101010101)_2 = (341)_{10}$

Convert $(101.01)_2$ to Decimal

$(101.01)_2 = 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2}$

$(101.01)_2 = 4 + 0 + 1 + 0 + 0.25$

$(101.01)_2 = (5.25)_{10}$

Convert $(110.1101)_2$ to Decimal

$(110.1101)_2 = 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$

$(110.1101)_2 = 4 + 2 + 0 + 0.5 + 0.25 + 0 + 0.0625$

$(110.1101)_2 = (6.8125)_{10}$

Conversion from Decimal to hexadecimal number system

Conversion of decimal number to hexadecimal number system is carried two in steps

i. Conversion of integer part is accomplished by successive division
ii. Conversion of fractional part is accomplished by successive multiplication

Conversion of integer part

- Divide the integer part of decimal number by 16 until quotient is one.
- The remainders of each division form the digits of hexadecimal number
- If the remainder is from 10 to 15 then represent it as A to F
- The remainders are taken in the reverse order (bottom to top) to form a hexadecimal number

Conversion of fractional part

- Multiply the fractional part of decimal number by 16
- The integer part of the product form the digits of hexadecimal number
- The fractional part is again multiplied by 16 and this process is repeated until fractional part becomes zero or sufficient digits are obtained.

- If the integer is from 10 to 15 then represent it as A to F
- The integer part are read downwards (top to bottom) to form a hexadecimal number

Convert $(57345)_{10}$ to Hexadecimal

16	57345	Remainder
16	3584	1
16	224	0
	14 = E	0

Therefore $(57345)_{10} = (E001)_{16}$

Convert $(972.625)_{10}$ to hexadecimal

16	972	Remainder
16	60	12 = C
	3	12 = C

Integer

0.625* 16 = 10.0 10 = A

Therefore $(972.625)_{10} = (3CC.A)_{16}$

Conversion from Hexadecimal to Decimal number system

- Decimal equivalent of a given hexadecimal number is obtained by multiplying each digit of the hexadecimal number with the radix to the power of position value and add each multiplication result
- Before multiplying hexadecimal numbers A to F it should be converted into equivalent decimal.

Convert $(F8E.28)_{16}$ to decimal

$(F8E.28)_{16} = F*16^2 + 8*16^1 + E*16^0 + 2*16^{-1} + 8*16^{-2}$

$(F8E.28)_{16} = 15*16^2 + 8*16^1 + 14*16^0 + 2*16^{-1} + 8*16^{-2}$

$(F8E.28)_{16} = 3840 + 128 + 14 + 0.25 + 0.125$

Therefore $(F8E.28)_{16} = (3982.15625)_{10}$

Convert $(1C00)_{16}$ to decimal

$(1C00)_{16} = 1*16^3 + C*16^2 + 0*16^1 + 0*16^0$

$(1C00)_{16} = 1*16^3 + 12*16^2 + 0*16^1 + 0*16^0$

$(1C00)_{16} = 4096 + 3072$

Therefore $(1C00)_{16} = (7168)_{10}$

Conversion from Hexadecimal to Binary number system

- Each digit of hexadecimal number is individually converted to its binary equivalent
- Leading and trailing zeros can be removed.

Convert $(20E.CA)_{16}$ to binary

$(20E.CA)_{16} = (0010\ 0000\ 1110.1100\ 1010)_2$

Therefore $(20E.CA)_{16} = (001000001110.11001010)_2$

Convert $(7B3.C2)_{16}$ to binary

$(7B3.C2)_{16} = (0111\ 1011\ 0011.1100\ 0010)_2$

Therefore $(7B3.C2)_{16} = (11110110011.1100001)_2$

Conversion from Binary to Hexadecimal number system

- Make group of 4-bits starting from LSB for integer part by inserting 0's at the MSB end.
- Make group of 4-bits starting from MSB for fractional part by inserting 0's at the LSB end.
- Write equivalent hexadecimal number for each group of 4-bits.

Convert $(10100110111110)_2$ to hexadecimal

$(10100110111110)_2 = (0010\ 1001\ 1011\ 1110)_{16}$

Therefore $(10100110111110)_2 = (29BE)_{16}$

Convert $(1101101110.1001101)_2$ to hexadecimal

$(1101101110.1001101)_2 = (\mathbf{00}11\ 0110\ 1110.1001\ 101\mathbf{0})_{16}$

Therefore $(1101101110.1001101)_2 = (36E.9A)_{16}$

Conversion from Octal to Binary number system

- Each digit of Octal number is individually converted to its binary equivalent
- Leading and trailing zeros can be removed.

Convert $(642.71)_8$ to binary

$(642.71)_8 = (110\ 100\ 010.111\ 001)_2$

Therefore $(642.71)_8 = (110100010.111001)_2$

Convert $(255.34)_8$ to binary

$(255.34)_8 = (010\ 101\ 101.011\ 100)_2$

Therefore $(255.34)_8 = (10101101.0111)_2$

Conversion from Binary to Octal number system

- Group 3-bits starting from LSB for integer part by inserting 0's at the MSB end.
- Group 3-bits starting from MSB for fractional part by inserting 0's at the LSB end.
- Write equivalent octal number for each group of 3-bits.

Convert $(1110.01101)_2$ to octal

$(1110.01101)_2 = (\overline{001}\ 110\overline{.011}\ \ 010\ )_8$

Therefore $(1110.01101)_2 = (16.32)_8$

Convert $(0.101010110)_2$ to octal

$(0.101010110)_2 = (0.\overline{101}\ 010\ \overline{110})_8$

Therefore $(0.101010110)_2 = (0.526)_8$

Numeral systems conversion table

Decimal Base-10	Binary Base-2	Octal Base-8	Hexadecimal Base-16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

Binary addition and subtraction

The addition and subtraction of binary numbers are simpler than decimal addition and subtraction, because here only two digits 0 and 1 are involved.

Rules of binary addition are

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$

Addition of 1 with 1 gives 0 as sum and 1 as carry.

Add $(101011)_2$ and $(11001)_2$

$$\begin{array}{cccccccc} & 1 & 0 & 1 & 0 & 1 & 1 \\ & & 1 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Complement of Binary Numbers

Complement numbers is used to perform subtraction of two numbers. Binary numbers can be represented either in l's and 2's complement form

- l's complement of a binary number is obtained by changing all 1's to 0's and 0's to 1's.

- The 2's complement of a binary number is obtained by adding 1 to 1's complement.

2's Complement = 1's Complement + 1

Find l's complement of $(101010)_2$

l's complement of $(101010l)_2$ is $(0l0l0l0)_2$

Find 2's complement of $(101010l)_2$

2's complement of $(101010l)_2$ is $(0l0l0l0)_2 + 1 = (0l0l0l1)_2$

Procedure for 1's complement & 2's complement subtraction of X-Y

1's complement Method

1. Take 1's complement of Y
2. Result = X+1's complement of Y
3. If carry is generated then

   result is +ve and in the true form. Add carry to the result to get the final result

   ELSE

   result is negative and in the 1's complement form

2's complement Method

1. Take 2's complement of Y
2. Result = X+2's complement of Y
3. If carry is generated then

   result is positive and in the true form. In this case, carry is ignored

   ELSE

   result is negative and it's in the 2's complement form.

Subtract $(28)_{10} - (15)_{10}$ using 1's complement method

GIVEN : $(28)_{10} = (11100)_2$ | $(15)_{10} = (01111)_2$

STEP1 **1's complement of $(15)_{10} = (10000)_2$**

STEP 2 **Add 28 to 1's complement of 15**

$$\begin{array}{cccccc} & 1 & 1 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$

STEP 3 **Add carry to the LSB**

0	0	1	0	0
				1
0	1	1	0	1

Therefore $(11100)_2$ - $(01111)_2$ = $(01101)_2$

Subtract $(15)_{10}$ – $(28)_{10}$ using 1's complement method

$(15)_{10} = (01111)_2$

$(28)_{10} = (11100)_2$

1's complement of $(28)_{10} = (00011)_2$

Add 15 to 1's complement of 28

0	1	1	1	1
0	0	0	1	1
1	0	0	1	0

Since no carry, the result is negative and it is in 1's complement form.

Therefore 1's complement of result is = $(01101)_2$

Subtract $(28)_{10}$ – $(15)_{10}$ using 2's complement method

$(28)_{10} = (11100)_2$

$(15)_{10} = (01111)_2$

1's complement of $(15)_{10} = (10000)_2$

2's complement of $(15)_{10}$

1	0	0	0	0
				1
1	0	0	0	1

Add 28 to 2's complement of 15

	1	1	1	0	0
	1	0	0	0	1
1	0	1	1	0	1

Ignore the carry

Therefore $(11100)_2$ - $(01111)_2$ = $(01101)_2$

Subtract $(15)_{10} - (28)_{10}$ using 2's complement method

$(15)_{10} = (01111)_2$

$(28)_{10} = (11100)_2$

1's complement of $(28)_{10} = (00011)_2$

2's complement of $(28)_{10}$

$$\begin{array}{r} 0\ 0\ 0\ 1\ 1 \\ \quad\quad\quad\quad 1 \\ \hline 0\ 0\ 1\ 0\ 0 \end{array}$$

Add 15 to 2's complement of 28

$$\begin{array}{r} 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 1\ 0\ 0 \\ \hline 1\ 0\ 0\ 1\ 1 \end{array}$$

Since no carry, the result is negative and it is in 2's complement form.

Therefore 2's complement of result is

$$\begin{array}{r} 0\ 1\ 1\ 0\ 0 \\ \quad\quad\quad\quad 1 \\ \hline 0\ 1\ 1\ 0\ 1 \end{array}$$

Subtract $(10.0101)_2 - (101.111)_2$ using 1's complement method and 2's complement method

1's complement method

$(10.0101)_2 - (101.111)_2$

Make the length of two binary numbers same

$(010.0101)_2 - (101.1110)_2$

1's complement of $(101.1110)_2 = (010.0001)_2$

Add $(010.0101)_2$ to 1's complement of $(101.1110)_2$

$$\begin{array}{r} 0\ 1\ 0\ .\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ .\ 0\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ .\ 0\ 1\ 1\ 0 \end{array}$$

Since no carry, the result is negative and it is in 1's complement form.

Therefore 1's complement of result is = $(011.1001)_2$

2's complement method

$(10.0101)_2$ - $(101.111)_2$

Make the length of two binary numbers same

$(010.0101)_2$ - $(101.1110)_2$

1's complement of $(101.1110)_2$ = $(010.0001)_2$

2's complement of $(101.1110)_2$ =

0	1	0	.	0	0	0	1
							1
0	1	0	.	0	0	1	0

Add $(010.0101)_2$ to 2's complement of $(101.1110)_2$

0	1	0	.	0	0	0	1
0	1	0	.	0	1	0	1
1	0	0	.	0	1	1	1

Since no carry, the result is negative and it is in 2's complement form.

Therefore 2's complement of result is

0	1	1	.	1	0	0	0
							1
0	1	1	.	1	0	0	1

Digital Circuits

Digital circuits are classified into: Combinational circuits and Sequential circuits.

In Combinational circuits output depends on the present input only.

In Sequential circuits output is produced on the basis of both present and previous inputs.

Logic Gates

- Logic gates are the basic building blocks of any digital system.
- It is an electronic circuit having one or more than one input and only one output.
- The relationship between the input and the output is based on certain **logic**. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.
- The operation of a logic gate can be easily understood with a help of a truth table.
- A truth table is a table that shows all the input output possibilities of a logic gate.

Types of Logic Gates

The various types of logic gates are

1. NOT Gate
2. AND Gate
3. OR Gate
4. NAND Gate
5. NOR Gate
6. OR Gate (Exclusive-OR)
7. XNOR Gate (Exclusive NOR)

BASIC GATES

Basic logic operation are NOT, AND & OR. Hence these gates are called as basic gates.

UNIVERSAL GATES

All the logic functions can be realized either using NAND gate or NOR gate. Hence these gates are called as Universal gates

NOT Gate

In NOT operation, the output is the complement of input.

Symbol

Truth table

A	Y
0	0
1	0

Output expression of NOT gate is given by $\mathbf{Y = \overline{A}}$

AND Gate

In AND operation, the output is 1 only if both inputs A and B are 1; otherwise it is zero.

Symbol

Truth table

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

The output expression of AND gate is given by

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

OR Gate

In OR operation, the output is 0 only if both inputs A and B are 0; otherwise, it is one.

Symbol 

Truth table

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

The output expression is given by **Y = A+B**

NAND Gate

In NAND operation, the output is 0 only if both inputs A and B are 1; otherwise, it is one.

Symbol 

Truth table

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

The output expression is given by $\mathbf{Y = \overline{A \cdot B}}$

NOR Gate

In NOR operation, the output is 1 only if both inputs A and B are 0; otherwise it is zero.

Symbol

Truth table

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

The output expression is given by

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

XOR Gate

In XOR operation, the output is 1 only if both inputs A and B are not equal; otherwise, it is zero.

Symbol 

Truth table

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

The output expression is given by

$$\mathbf{Y = A \oplus B}$$

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

XNOR Gate (Exclusive NOR)

In XNOR operation, the output is 1 only if both inputs A and B are equal; otherwise, it is zero.

Symbol 

Truth table

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

The output expression is given by

$$\mathbf{Y = \overline{A \oplus B}}$$

$$\mathbf{Y = \overline{A}\,\overline{B} + AB}$$

Boolean Algebra

Boolean Theorems

Name	AND form	OR form
Identity law	$1 \cdot A = A$	$0 + A = A$
Null law	$0 \cdot A = 0$	$1 + A = 1$
Idempotent law	$A \cdot A = A$	$A + A = A$
Inverse law	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$A \cdot B = B \cdot A$	$A + B = B + A$
Associative law	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B) \cdot (A + C)$	$A(B + C) = AB + AC$
Absorption law	$A \cdot (A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

De-Morgan's Theorem

1. The complement of product is equal to the sum of complements.

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

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

2. The complement of sum is equal to the product of the complements.

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

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

Universal gates

Realization of logic gates using NAND gates

1. **NOT gate**

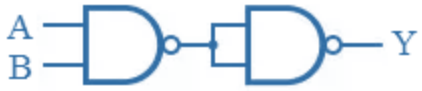
$$\mathbf{Y = \overline{A}}$$

2. **AND Gate**

$$\mathbf{Y = AB}$$

Compliment twice on both sides

$$\mathbf{Y = \overline{\overline{AB}}}$$



3. **OR Gate**

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

Compliment on both sides

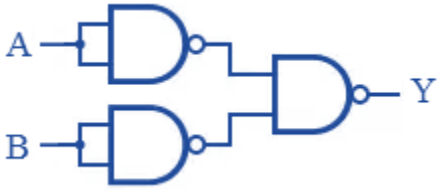
$$\mathbf{\overline{Y} = \overline{A + B}}$$

Apply Demorgan's theorem

$$\mathbf{\overline{Y} = \overline{A}.\overline{B}}$$

Compliment on both sides

$$\mathbf{Y = \overline{\overline{A}.\overline{B}}}$$



4. **NOR Gate**

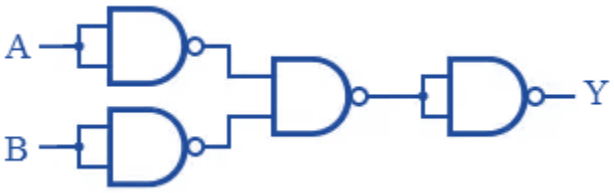
$$\mathbf{Y = \overline{A + B}}$$

Apply Demorgan's theorem

$$\mathbf{Y = \overline{A}.\overline{B}}$$

Compliment on both sides twice

$$\mathbf{Y = \overline{\overline{A}.\overline{B}}}$$



5. **XOR Gate**

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

Compliment on both sides

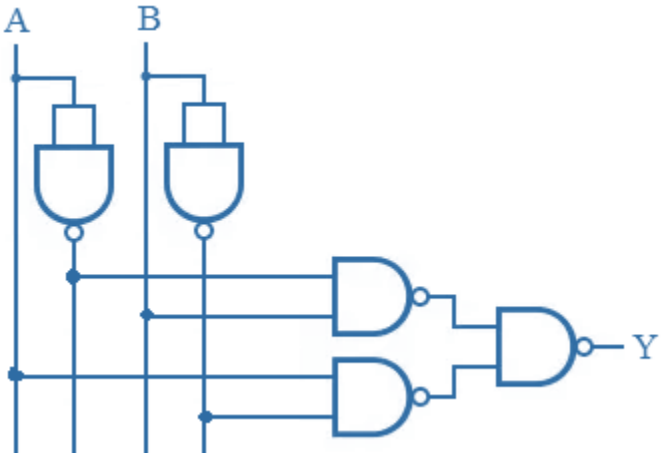
$$\mathbf{\overline{Y} = \overline{\overline{A}B + A\overline{B}}}$$

Apply Demorgan's theorem

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

Compliment on both sides

$$\mathbf{Y} = \overline{(\overline{\mathbf{A}}\mathbf{B})(\mathbf{A}\overline{\mathbf{B}})}$$



6. XNOR Gate

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

Compliment on both sides

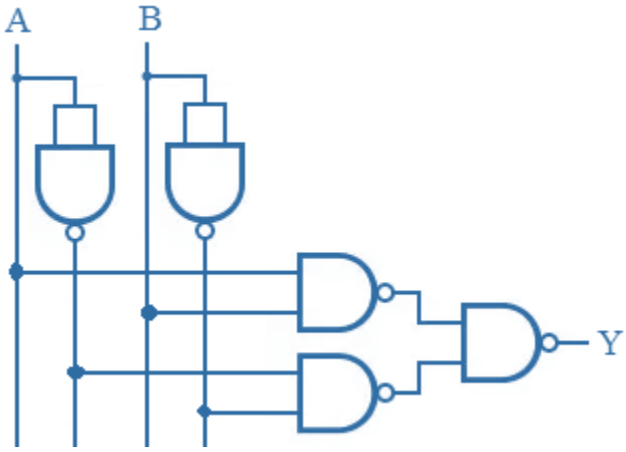
$$\overline{\mathbf{Y}} = \overline{\mathbf{A}.\mathbf{B} + \overline{\mathbf{A}}.\overline{\mathbf{B}}}$$

Apply Demorgan's theorem

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

Compliment on both sides

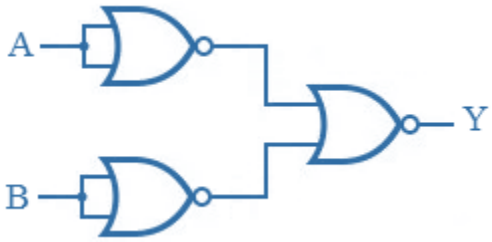
$$\mathbf{Y} = \overline{(\overline{\mathbf{A}\mathbf{B}}).(\overline{\overline{\mathbf{A}}\overline{\mathbf{B}}})}$$



Realization of logic gates using NOR gates

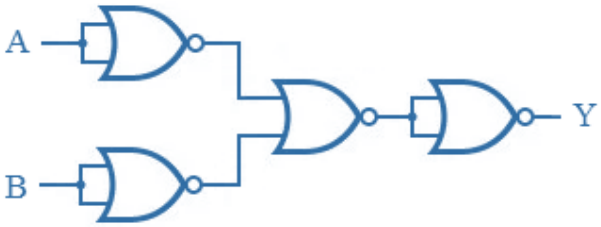
1. **NOT gate**

2. **AND Gate**

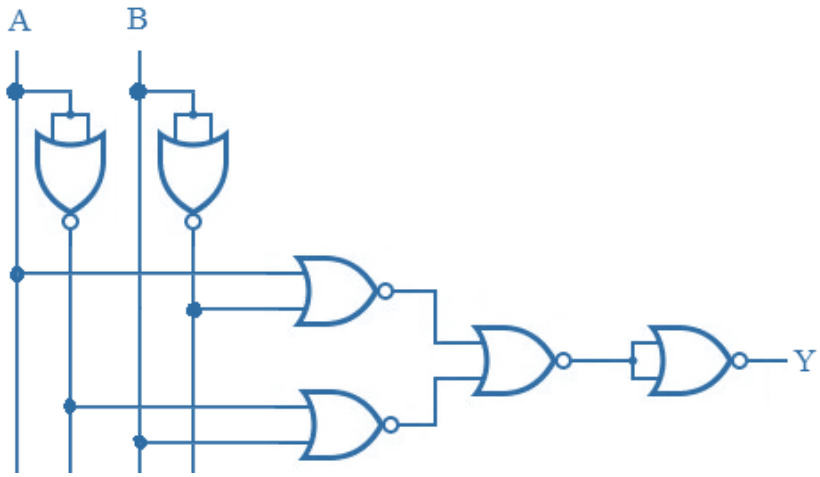


3. **OR Gate**

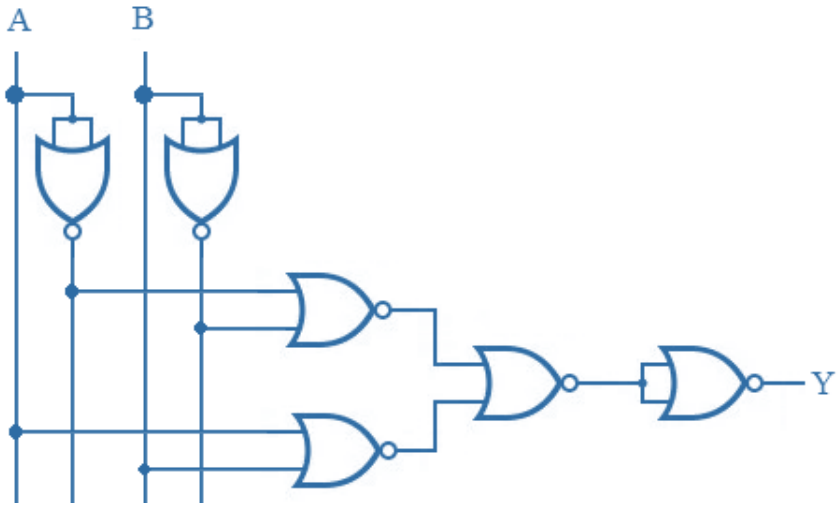
4. **NAND Gate**



5. **XOR Gate**



6. **XNOR Gate**



Problems

Simplify the following Boolean expression and realize using basic gates

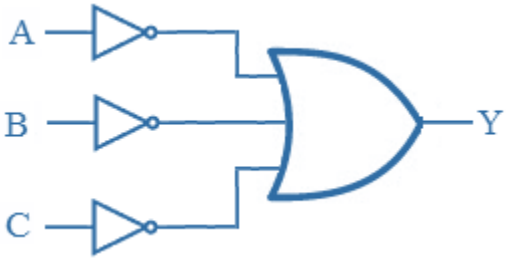
1. $Y = A\overline{BC} + \overline{ABC} + \overline{A}B\overline{C} + \overline{AB}C$

$= A(\overline{B} + \overline{C}) + \overline{A} + \overline{B} + \overline{C} + \overline{A}B\overline{C} + (\overline{A} + \overline{B})C$ [DeMorgan's theorem]

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

$= \overline{B}(A + 1 + C) + \overline{C}(A + 1 + \overline{A}B) + \overline{A}(1 + C)$ [1 + A = 1]

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



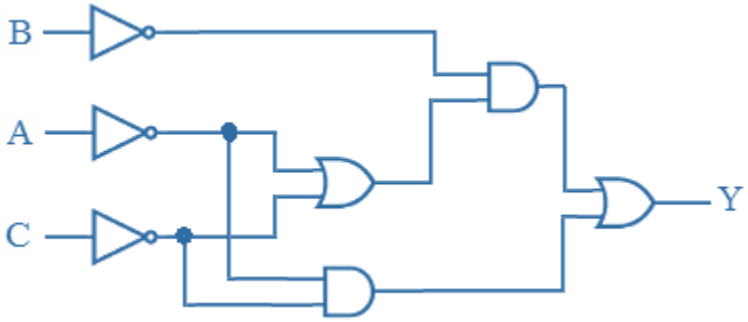
2. $Y = A\overline{BC} + \overline{ABC} + \overline{AB} + \overline{AC}$

$= \overline{BC}(A + \overline{A}) + \overline{AB} + \overline{AC}$

$= \overline{BC}(A + \overline{A}) + \overline{AB} + \overline{AC}$

$= \overline{BC} + \overline{AB} + \overline{AC}$

$= \overline{B}(\overline{C} + \overline{A}) + \overline{AC}$



3. $Y = (A + \overline{B} + C)(\overline{A} + B + \overline{C})(A + \overline{B})$

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

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

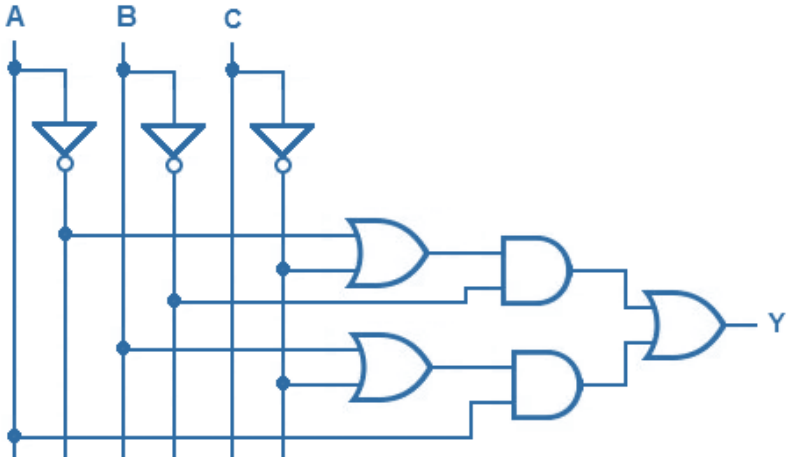
$= \overline{A}\,\overline{B}A + \overline{A}CA + ABA + CBA + A\overline{C}A + \overline{B}\overline{C}A + \overline{A}\,\overline{B}\overline{B} + \overline{A}C\overline{B} + AB\overline{B} + CB\overline{B} + A\overline{C}\overline{B} + \overline{B}C\overline{B}$

$= AB + ABC + A\overline{C} + A\overline{B}\overline{C} + \overline{A}\,\overline{B} + \overline{A}\overline{B}C + A\overline{B}\overline{C} + \overline{B}\overline{C}$

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

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

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



Realize the following expression using NAND gates only

1. $Y = (A + B)(C + D)(A + \overline{B})$

Complement on both sides

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

Complement on both sides once again

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

Let $X = A + B$

Complement on both sides

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

Apply De Morgans theorem

$$\overline{X} = \overline{A}\,\overline{B}$$

Complement on both sides once again

$$X = \overline{\overline{A}\,\overline{B}}$$

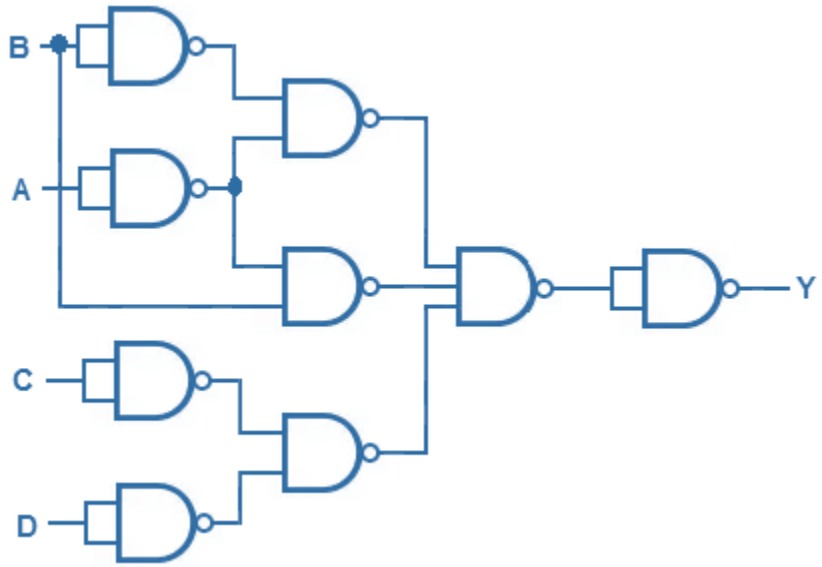
Similarly we can write

$$C + D = \overline{\overline{C}\,\overline{D}}$$

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

Therefore

$$Y = \overline{\overline{(\overline{A}\,\overline{B})(\overline{C}\,\overline{D})(\overline{A}\,\overline{B})}}$$



Realize the following expression using NOR gates only

1. $Y = (A + B)(C + D)(A + \overline{B})$

   Complement on both sides

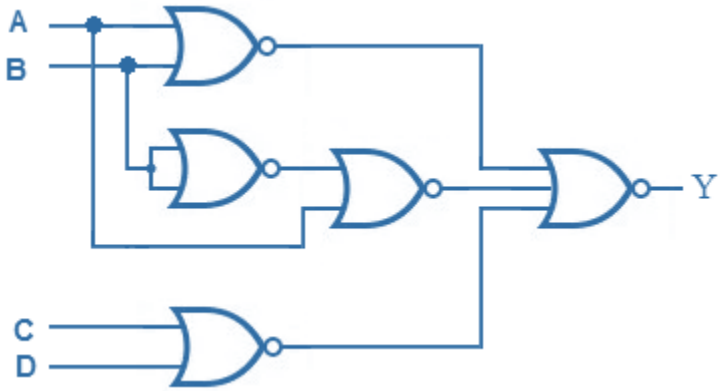
   $$\overline{Y} = \overline{(A+B)(C+D)(A+\overline{B})}$$

   Apply De Morgans theorem

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

Complement on both sides once again

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



Combinatorial circuits

Half Adder

A half adder is a combinational circuit which adds two one-bit binary numbers and produces two binary outputs.

The logic circuit, truth table and logic symbol for a half adder is shown below.

Truth table

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

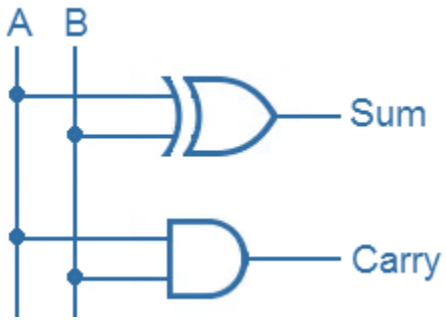
From the truth table

$$Sum = A \oplus B$$

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

$$Carry = AB$$

Realization of half adder using logic gates



Limitations (disadvantages) of half-adder

- In multi-digit addition we have to add two bits along with the carry of previous digit addition. Such addition requires addition of 3 bits. This is not possible in half-adders.

Full Adder

A full adder is a combinational circuit which adds three one-bit binary numbers and produces two binary outputs.

The logic circuit, truth table and logic symbol for a full adder is shown below.

Truth table

Inputs			Outputs	
A	**B**	**C_{in}**	**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

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

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

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

$$\text{Sum} = C_{in}(\overline{A \oplus B}) + \bar{C}_{in}(A \oplus B)$$

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

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

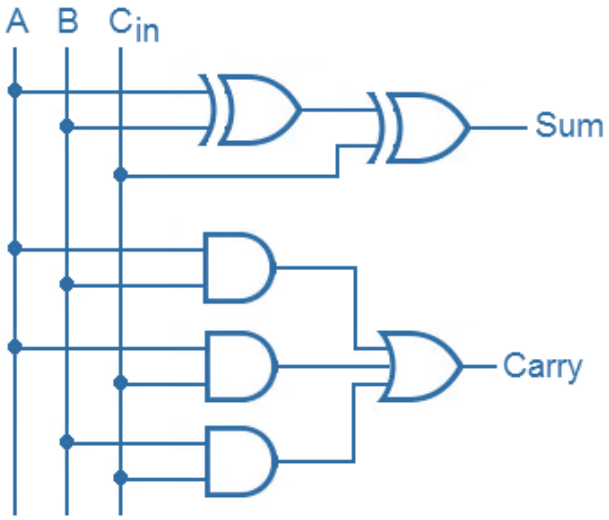
$$\text{Carry} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} + ABC_{in} + ABC_{in}$$

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

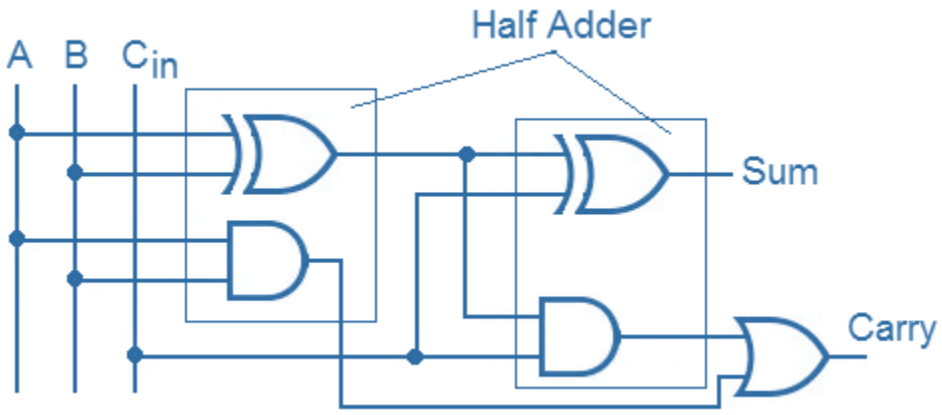
$$\text{Carry} = BC_{in}(\bar{A} + A) + AC_{in}(\bar{B} + B) + AB(\bar{C}_{in} + C_{in})$$

$$\text{Carry} = AB + BC_{in} + AC_{in}$$

Realization of full adder using logic gates

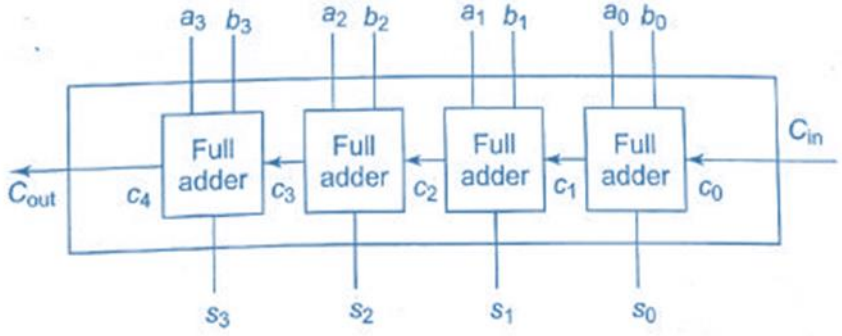


- Realization of full adder using two half adder and OR gate



Parallel Adder (4 bit)

An n-bit adder can be designed by connecting the carry out and carry in lines of n full adders. Figure shows a 4-bit adder. This design is called a ripple-carry adder. Similarly, 4-bit adders can be connected to form a 16-bit adder.



Multiplexer

Multiplexer (MUX) is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer (MUX) are also known as data selectors because they can "select" each input line.

4:1 MUX

- A 4-to-1 multiplexer consists four data input lines as I_0 to I_3, two select lines as S_0 and S_1 and a single output line Y as shown in the fig.5.3. The select lines S_1 and S_2 select one of

the four input lines to connect the output line.

- The truth table of a 4-to-1 multiplexer has four combinations 00, 01, 10 and 11 on the select lines S_1 and S_0.
- The select line selects one of the inputs I_0, I_2, I_1 and I_3 to the output i.e. when $S_1 = 0$ and $S_0 = 0$, the output Y is I_0, similarly Y is I_1 if the select inputs $S_1 = 0$ and $S_0 = 1$ and so on.

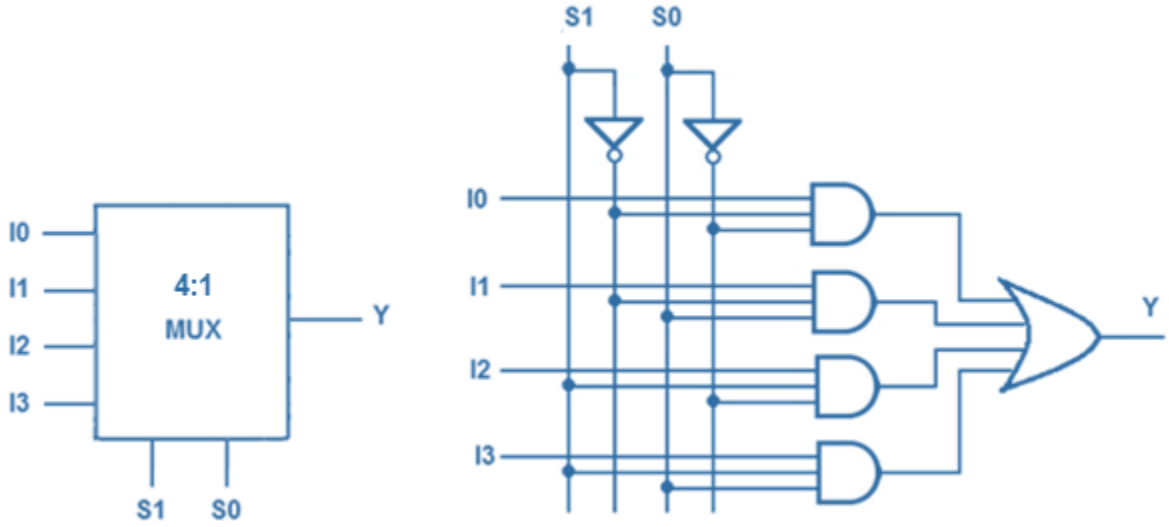


Fig.5.3: 4:1 MUX

Truth table

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

Therefore, the output expression is given by

$$Y = \bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3$$

4-to-1 multiplexer can be implemented by using basic logic gates as shown in the fig.5.3.

8:1 MUX

A 8-to-1 multiplexer consists four data input lines as I_0 to I_7, three select lines as S_2, S_1 and S_0 and a single output line Y as shown in the fig.5.4. The select lines S_0, S_1 and S_2 select one of the eight input lines to connect the output line.

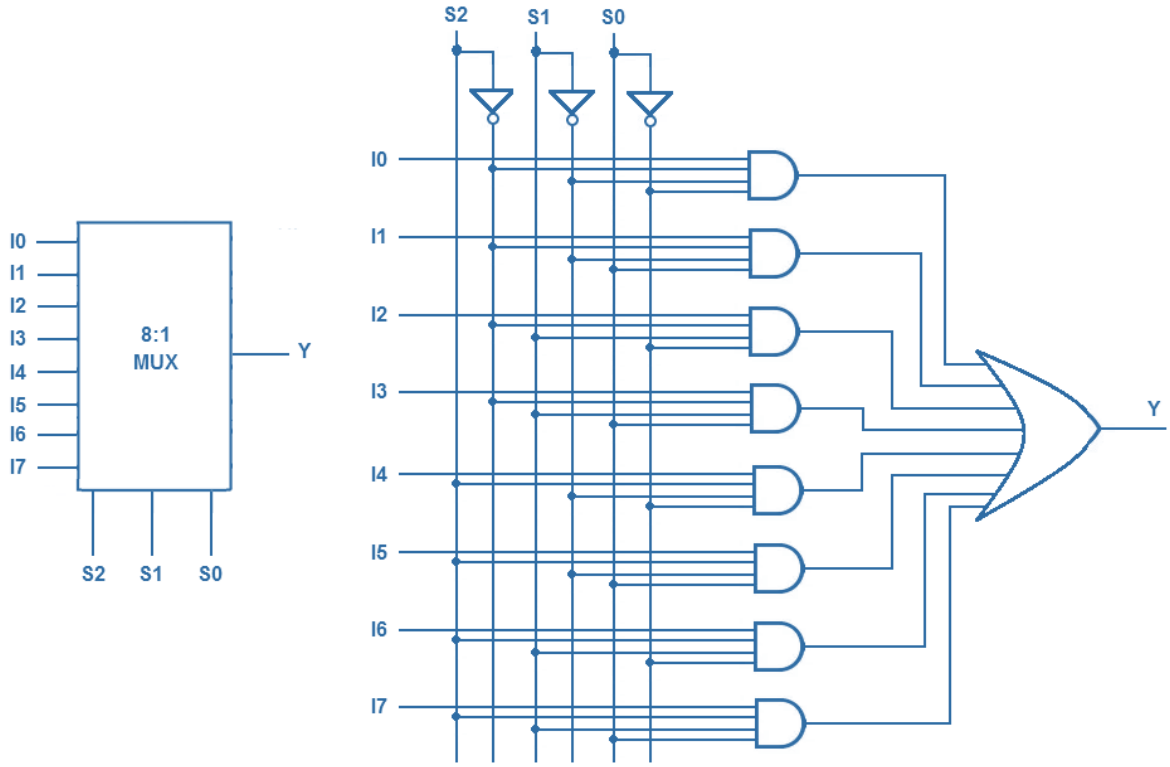


Fig.5.4: 8:1 MUX

Truth table

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

- The truth table of a 8-to-1 multiplexer has eight combinations 000, 001, 010, 011, 100, 101, 110 and 111 on the select lines S_2, S_1 and S_0
- The select line selects one of the inputs to the output i.e. when $S_2 = 0$, $S_1 = 0$ and $S_0 = 0$, the output Y is I_0, similarly Y is I_1 if the select inputs $S_2 = 0$, $S_1 = 0$ and $S_0 = 1$ and so on.
- Therefore the output expression is given by

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0I_0 + \bar{S}_2\bar{S}_1S_0I_1 + \bar{S}_2S_1\bar{S}_0I_2 + \bar{S}_2S_1S_0I_3 + S_2\bar{S}_1\bar{S}_0I_4 +$$

$$S_2\bar{S}_1S_0I_5 + S_2S_1\bar{S}_0I_6 + S_2S_1S_0I_7$$

- 8-to-1 multiplexer can be implemented by using basic logic gates as shown in the fig.5.4.

Decoders

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

- The objective of the decoder is to decode an n bit binary number, producing a signal on one of the 2^n output lines.

2-to-4 decoder

- 2 to 4 Decoder has two inputs and four outputs as shown in the fig.5.5.

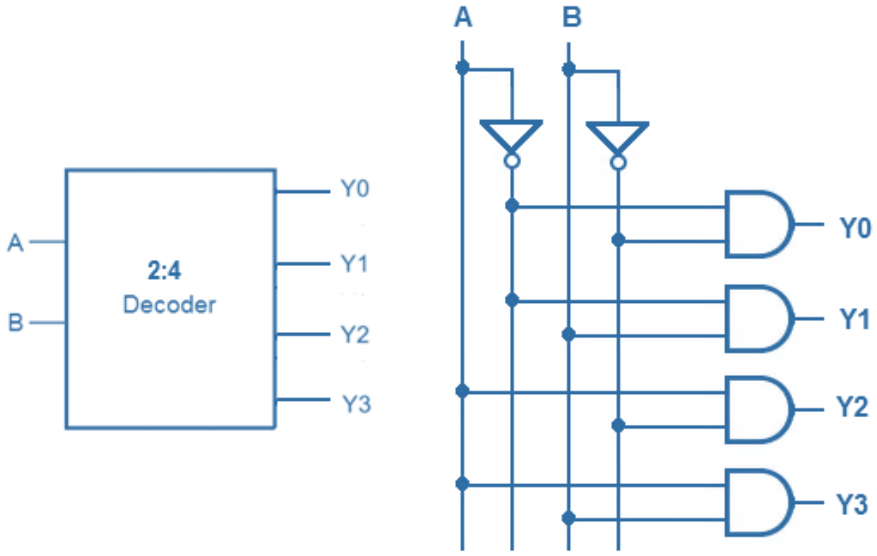


Fig.5.5: 2-to-4 decoder

Truth table

Inputs		Outputs			
A	**B**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- From the truth table the expression can be written as

$$Y_0 = \overline{A}\overline{B}$$

$$Y_1 = \overline{A}B$$

$$Y_2 = A\overline{B}$$

$$Y_3 = AB$$

- 2-to-4 decoder can be implemented by using basic logic gates as shown in the fig.5.5.

3-to-8 decoder

- 3 to 8 Decoder has three inputs and eight outputs as shown in the fig.5.6.

Truth table

Inputs			Outputs							
A	**B**	**C**	**Y_7**	**Y_6**	**Y_5**	**Y_4**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

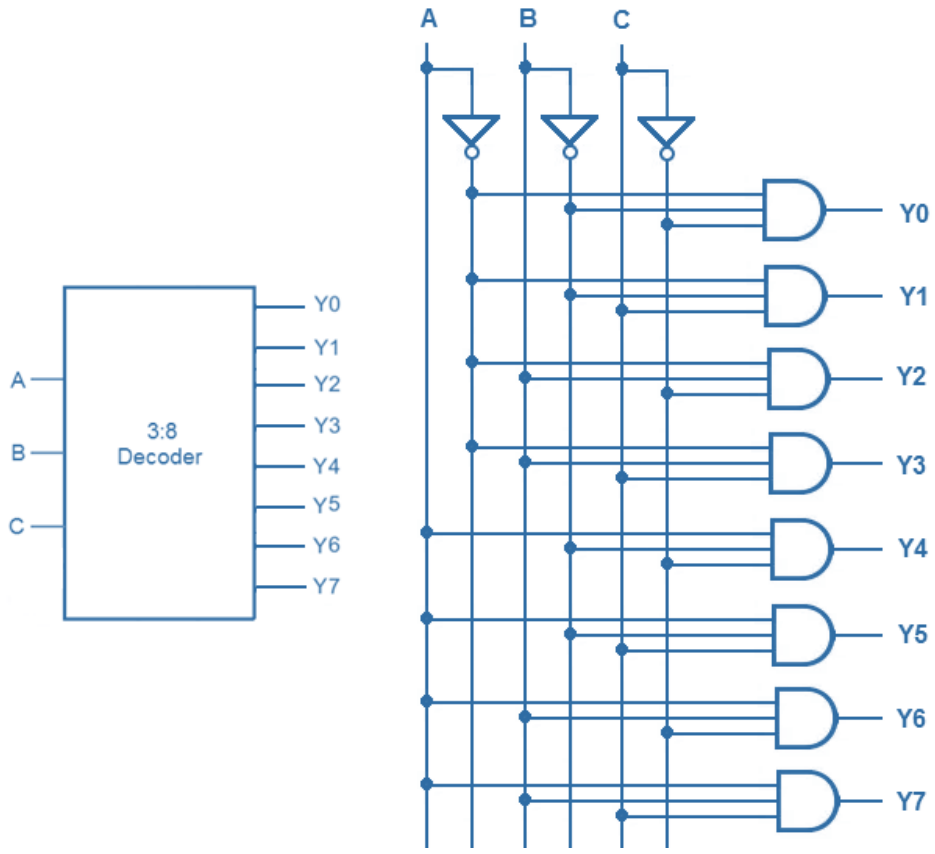


Fig.5.6: 3-to-8 decoder

- From the truth table the expression can be written as

$$Y_0 = \bar{A}\bar{B}\bar{C}$$

$$Y_1 = \bar{A}\bar{B}C$$

$$Y_2 = \bar{A}B\bar{C}$$

$$Y_3 = \bar{A}BC$$

$$Y_4 = A\bar{B}\bar{C}$$

$$Y_5 = A\bar{B}C$$

$$Y_6 = AB\bar{C}$$

$$Y_7 = ABC$$

- 3-to-8 decoder can be implemented by using basic logic gates as shown in the fig.5.6.

Flip-flop

- A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs.
- Due to the storage ability flip-flops are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

SR Flip-flop

- The SR flip flop circuit has three inputs: Clock (Clk), Set (S), Reset (R) and two outputs Q and $\overline{Q}$.
- A clock signal is fed as input to both the NAND gates of the SR flip-flop as shown in fig.5.7.
- When the clock is LOW, the output of NAND gate 1 and 2 are 1 and the latch is disabled and output does not change.
- When the clock is HIGH, the output of NAND gate 1 and 2 are $\overline{S}$ and $\overline{R}$, thus the latch operates normally.

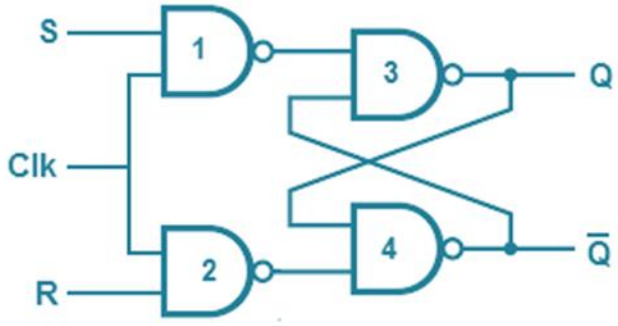


Fig.5.7: *SR flip-flop*

Truth Table

Clk	S	R	Q	$\overline{Q}$
0	X	X	Q	$\overline{Q}$
1	0	0	Q	$\overline{Q}$
1	0	1	0	1
1	1	0	1	0
1	1	1	Indeterminate	

When $S = 0$ and $R = 0$

Since both the inputs $S = 0$ & $R = 0$, Output cannot be determined. Let us assume the output Q = 1 ($\overline{Q} = 0$), then the inputs to the gate 3 are $\overline{S} = 1$ and $\overline{Q} = 0$, ∴ Q = 1. Now the inputs to the gate 4 are $\overline{R} = 1$ and $Q = 1$, ∴ $\overline{Q} = 0$.

Thus **there is no change in the output $Q = Q$ and $\overline{Q} = \overline{Q}$**

When $S = 0$ and $R = 1$

$R = 1$, will make $\overline{Q} = 1$. Since $\overline{Q} = 1$ it will results in $Q = 0$. Thus **$Q = 0$ and $\overline{Q} = 1$**

When $S = 1$ and $R = 0$

$S = 1$, will make $Q = 1$. Since $Q = 1$ it will results in $\overline{Q} = 0$. Thus **$Q = 1$ and $\overline{Q} = 0$**

When $S = 1$ and $R = 1$

$S = 1$ and $R = 1$, will make both Q and $\overline{Q}$ equal 1 ($\overline{Q} = Q = 1$) which is not possible.

∴ condition **$S = 1$ and $R = 1$** is not acceptable and **doesn't exist for RS flip-flop.**

JK Flip-flop

- The uncertainty in the state of an SR flip-flop when S = R = 1 can be eliminated by converting it into a JK flip - flop.
- The data inputs are J and K which are ANDed with $\overline{Q}$ and Q respectively to obtain S and R inputs as shown in fig.5.8. Thus S = J$\overline{Q}$ and R = KQ.

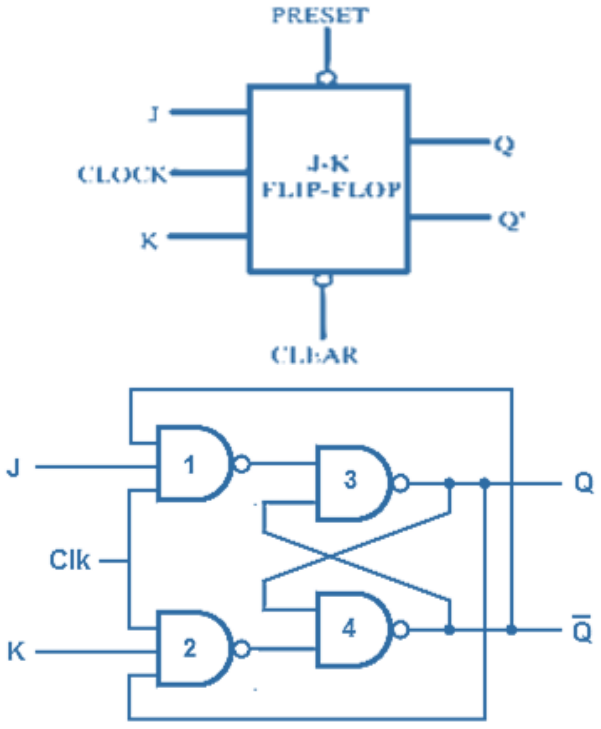


Fig.5.8: *JK flip-flop*

Truth Table

Clk	J	K	Q	$\overline{Q}$	Remarks
0	X	X	Q	$\overline{Q}$	**No change**
1	0	0	Q	$\overline{Q}$	**No change**
1	0	1	0	1	**Reset**
1	1	0	1	0	**Set**
1	1	1	$\overline{Q}$	Q	**Toggle**

When J = 0 and K = 0

- These J and K inputs disable the NAND gates 1 and 2, therefore clock pulse have no effect on the flip flop. In other words, Q returns it last value.

When J = 0 and K = 1

- The NAND gate 1 is disabled and NAND gate 2 is enabled if Q is 1, therefore, flip flop will be reset (Q = 0, $\overline{Q}$ = 1).

When J = 1 and K = 0

- The NAND gate 2 is disabled and the NAND gate 1 is enabled if $\overline{Q}$ is at 1, therefore, flip flop will be set (Q = 1, $\overline{Q}$ = 0).

When J = 1 and K = 1

- If Q = 0 the NAND gate 2 is disabled and NAND gate 1 is enabled. This will set the flip flop and hence Q will be 1. On the other hand if Q = 1, the NAND gate 2 is enabled and flip flop will be reset and hence Q will be 0. In other words, when J and K are both high, the clock pulses cause the JK flip flop to toggle.

***Race-Around condition*:**

Consider that J-K flip-flop is in RESET state i.e. Q = 0. Now that we apply J=K=1, output becomes complement to previous value i.e. now Q = 1. After the time interval equal to propagation delay of gates, the output is again complemented and becomes Q = 0. This continues as long as CLOCK = 1 and J=K=1. This phenomenon is termed as output races from 0 to 1 to 0 to 1 and so on . . .

This is called race-around condition.

Although the output of J-K flip-flop for fourth row is not uncertain, it is not useful either. The figure shows how output races between 0 and 1 as long as CLOCK is HIGH. When CLOCK goes LOW, whatever is the output is retained.

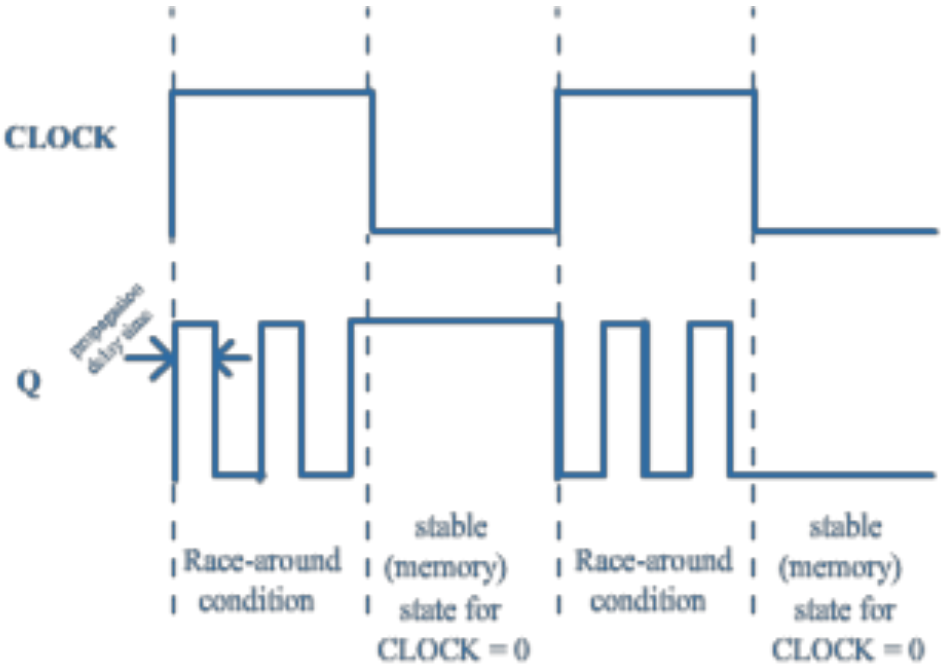


Fig.5.9 (a): JK flip-flop schematic output with race around condition

JK Master Slave Flipflop

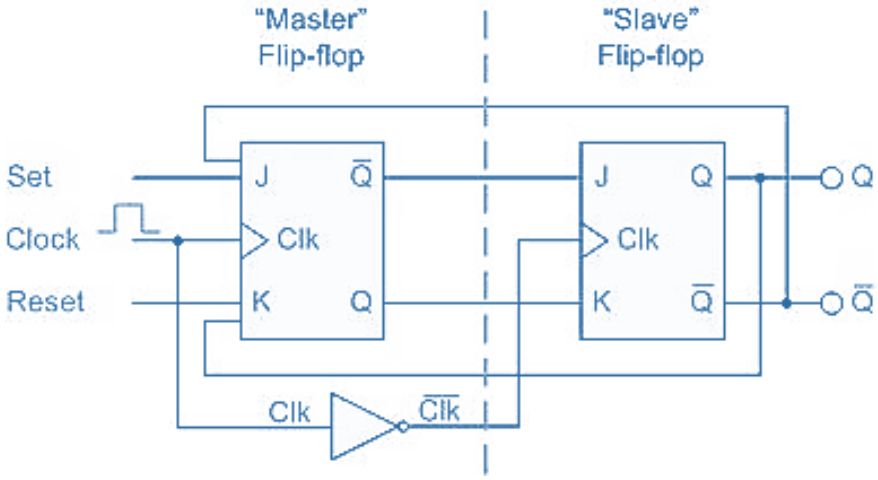


Fig.5.9(b): Master Slave JK flip-flop schematic

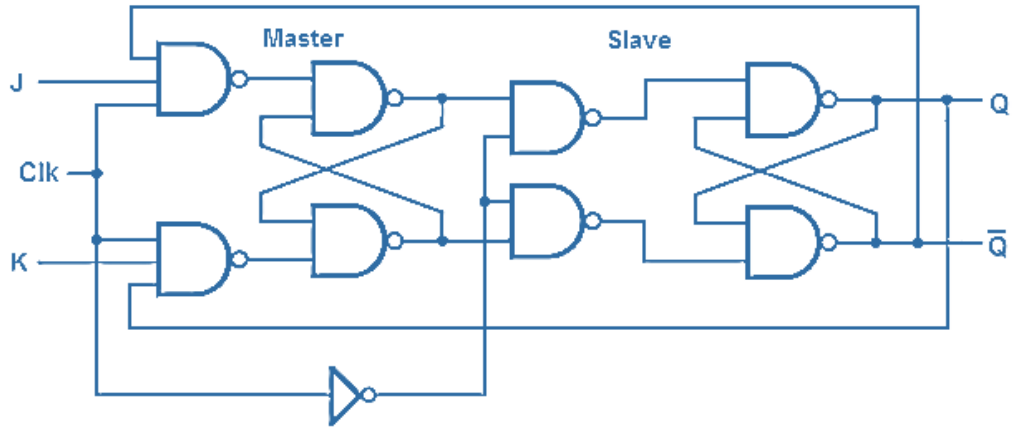


Fig.5.9c: ***Master Slave JK flip-flop***

- For J-K flip-flop, when clk = 1, and J = K = 1, for a long period of time, then Q output will toggle as long as Clk is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop. This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic "1" only for a very short time.
- The master-slave flip-flop eliminates **Race Around Condition** by using two JK flip-flops connected together in a series configuration as shown in fig.5.9. One flip-flop acts as the "Master" circuit, which triggers on the leading edge of the clock pulse while the other acts as the "Slave" circuit, which triggers on the falling edge of the clock pulse.
- In addition to these two flip-flops, the circuit also includes an **inverter**. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop. In other words, if clock = 0 for a master flip-flop, then clock = 1 for a slave flip-flop and if clock = 1 for master flip flop then it becomes 0 for slave flip flop.
- When the clock pulse goes to 1, the slave is isolated; J and K inputs may affect the state of the system. The slave flip-flop is isolated until the clock goes to 0. When the clock goes back to 0, information is passed from the master flip-flop to the slave and output is obtained.
- Firstly the master flip flop is positive level triggered and the slave flip flop is negative level triggered, so the master responds before the slave.
- If J = 0 and K = 1, the high $\overline{Q}$ output of the master goes to the K input of the slave and the clock forces the slave to reset, thus the slave copies the master.
- If J = 1 and K = 0, the high Q output of the master goes to the J input of the slave and the

Negative transition of the clock sets the slave, copying the master.

- If J = 1 and K = 1, it toggles on the positive transition of the clock and thus the slave toggles on the negative transition of the clock.
- If J = 0 and K = 0, the flip flop is disabled and Q remains unchanged.

Shift Register

Shift Register is a group of flip flops used to store multiple bits of data. The bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses.

- An n-bit shift register can be formed by connecting n flip-flops where each flip flop stores a single bit of data.
- Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:
  i. Serial-in to Parallel-out (SIPO): The register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
  ii. Serial-in to Serial-out (SISO): The data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
  iii. Parallel-in to Serial-out (PISO): The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
  iv. Parallel-in to Parallel-out (PIPO): The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.
- Fig.5.10 shows 4-bit shift register using JK flipflop in which input data is connected to J input and its compliment to K input of the left most flip flop of flip flop chain and all other flip flops are connected in serially.

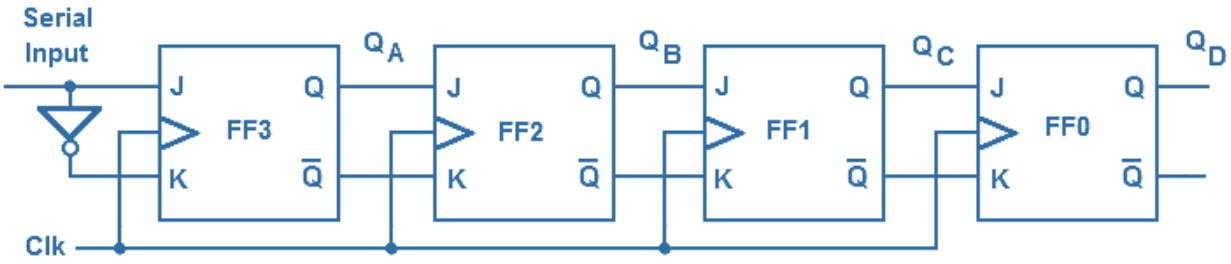


Fig.5.10: 4-bit Shift register

Clock pulse	Q_A	Q_B	Q_C	Q_D	Serial Output at Q_D
Initial Value	0	0	0	0	0
1	**1**	0	0	0	0
2	**1**	1	0	0	0
3	**0**	1	1	0	0
4	**1**	0	1	1	**1**
5	-	1	0	1	**1**
6	-	-	1	0	**0**
7	-	-	-	1	**1**

- Consider the serial input 1011 for the shift register. The input is applied to shift register by taking LSB first and lastly the MSB.
- On the first clock pulse LSB (logic 1) is applied as serial input to the serial input pin of FF3 therefore Q_A will be set (logic 1) and all the other outputs still remains at logic 0.
- On the second clock pulse the next bit (logic 1) is applied as serial input to the serial input pin of FF3 therefore Q_A will be set (logic 1). The previous output of Q_A is connected to FF2 therefore Q_B will be set (logic 1) and all the other outputs still remains at logic 0.
- On the third clock pulse the next bit (logic 0) is applied as serial input to the serial input pin of FF3 therefore Q_A will be reset (logic 0). The previous output of Q_A is connected to FF2 therefore Q_B will be set (logic 1). The previous output of Q_B is connected to FF1 therefore Q_C will be set (logic 1) and all the other outputs still remains at logic 0.

On the fourth clock pulse the next bit (logic 1) is applied as serial input to the serial input pin of FF3 therefore Q_A will be set (logic 1). The previous output of Q_A is connected to FF2 therefore Q_B will be reset (logic 0). The previous output of Q_B is connected to FF1 therefore Q_C will be set (logic 1) and the previous output of Q_C is connected to FF0 therefore Q_D will be set (logic 1).

Counters

Counter circuits count the applied clock pulses. These are usually designed using flip-flops.

- Based on the way clock is applied for their functioning counters are classified as **Synchronous and Asynchronous counters**.
- In **synchronous counter all the flip-flops are working in sync** with clock pulse as well as each other. Here clock pulse is applied to every flip flop.
- In asynchronous counter clock pulse is applied only to the initial flip flop whose value would be considered as LSB. Instead of the clock pulse, the output of first flip-flop acts as a clock pulse to the next flip flop, whose output is used as a clock to the next in line flip-flop and so on.

➢ Asynchronous counter is a ripple counter as the clock pulse ripples through the circuit. An n-MOD ripple counter contains n number of flip-flops and the circuit can count up to 2^n values before it resets itself to the initial value.

3-bit Binary Ripple Counter using JK Flip Flop

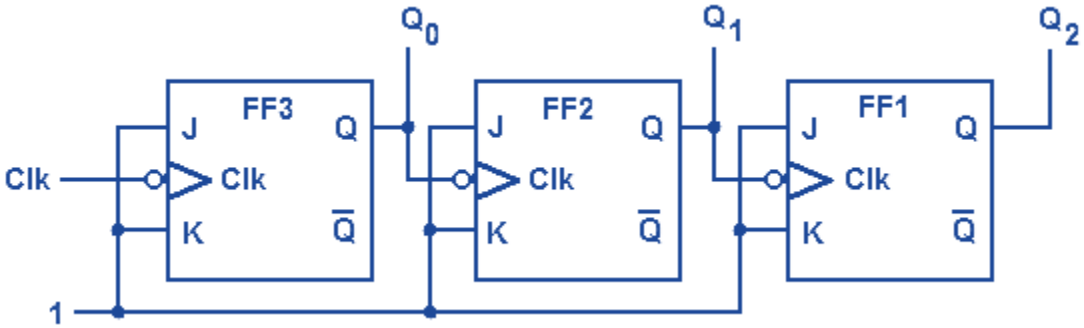


Fig.5.11: 3-bit binary ripple counter

Clock pulse	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1
9	0	0	0

- The **binary ripple counter** is as shown in the fig.5.11 uses three **JK flip flops** FF3, FF2 and FF1.
- JK inputs of flip flops are tied to logic 1. The bubble at the clock input indicates a negative triggered clock pulse.
- From the figure, it can be observed that the output Q_0 of the first flip flop is applied as a clock pulse to the second flip flop and output Q_1 of the second flip flop is applied as a clock pulse to the third flip flop.
- Here the output Q_0 is the LSB and the output Q_2 is the MSB bit.

The 3-bit ripple counter can count up to $2^3 = 8$ values.i.e., 000, 001, 010, 011, 100, 101, 110, 111.

Basic Communication System

- Communication is the process of transforming and transferring information from **one point to other**.
- All electronic communication systems consist of three basic components: a transmitter, a communication channel (medium), and a receiver.
- Messages are converted to electrical signals and sent over electrical or fiber-optic cable or free space to a receiver.
- Attenuation (weakening) and noise can interfere with transmission and reception.
- Communication systems is of two types
    i. **Wired communication**: through a wire. Ex: Cable TV, telephone etc
    ii. **Wireless communication**: is without using a wire. Ex: Radio, TV etc

Elements of Communication Systems

Fig.5.12 shows the block diagram of a communication system.

- The message produced by the source is not an electrical signal. Hence the input transducer is required for converting the message to electrical signal called message signal.
- At the destination another transducer converts the electrical signal into the appropriate message signal.

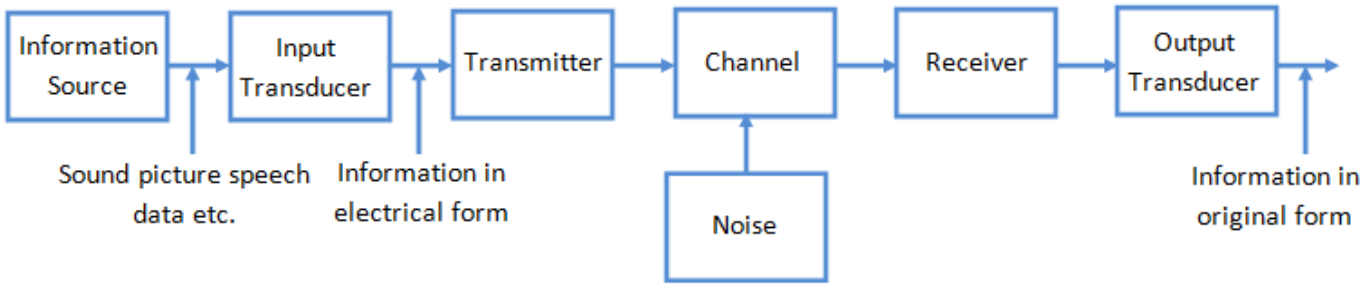


Fig.5.12: Block diagram of a communication system

Transmitter

- The transmitter couples the message signal into channel by converting it into suitable form for transmission.
- The operations performed by transmitter are amplification, filtering and modulation.

Channel and noise

- The channel is the medium through which the message travels from transmitter to receiver. Channel may be wired or wireless.
- During the process of transmission and reception the signal gets degraded due to superposition of noise into it.
- A noise is a random undesirable electric energy that enters communication system via the medium and interferes with the transmitted message.

Receiver

- Receiver is used to extract the input signal from the received signal through the channel which is degraded.
- The process of extraction is called as demodulation.

Principle of Operation of Mobile Phone

A cellular/mobile system provides standard telephone operation by full-duplex two way radio at remote locations.

- It provides a wireless connection to the public switched telephone network (PSTN) from any user location within the radio range of the system.

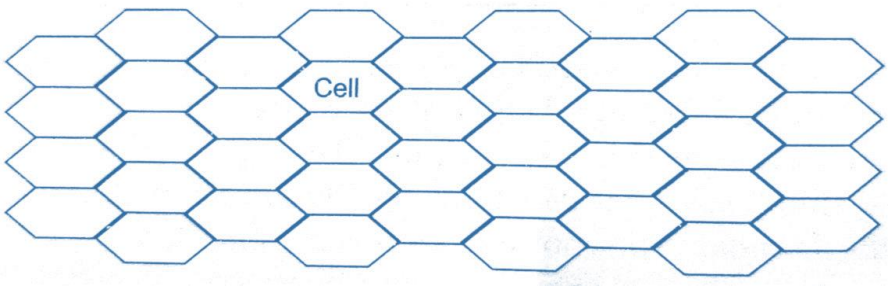


Fig.5.13: Cell area

- Rather than serving a given geographical area within a single transmitter and receiver, the cellular system divides the service area into many small areas known as cells as shown in fig.5.13.
- The typical cell covers only several square kilometers and contains its own receiver and low power transmitter.
- The cell area is shown in fig.5.13 is an ideal hexagon. However in reality they will have circular or other geometric shapes. These areas may overlap and cells may be of different sizes.
- A basic cellular system consists of mobile stations, base stations and mobile switching center (MSC). The MSC is also known as mobile telephone switching office (MSTO).
- The MSTO controls the cells and provides the interface between each cell and the main telephone office. Each mobile communicates via radio with one of the base stations and may be handed off to any other base station throughout the duration of the cell.
- Each mobile station consists of a transceiver, an antenna and control circuit.
- The base station consists of several transmitters and receivers which simultaneously handle full duplex communication and generally have towers which support several transmitting and receiving antennas. The base station serves as bridge between all mobile users in the cell and connects the simultaneous mobile calls via telephone lines or microwave link to the MSC.
- The MSC co-ordinates the activities of all the base stations and connects the entire cellular system to the public switching telephone center (PSTN). Most cellular systems also provide a service known as roaming.
- A simple block diagram representing working of mobile networks through GSM is shown in fig.5.14.

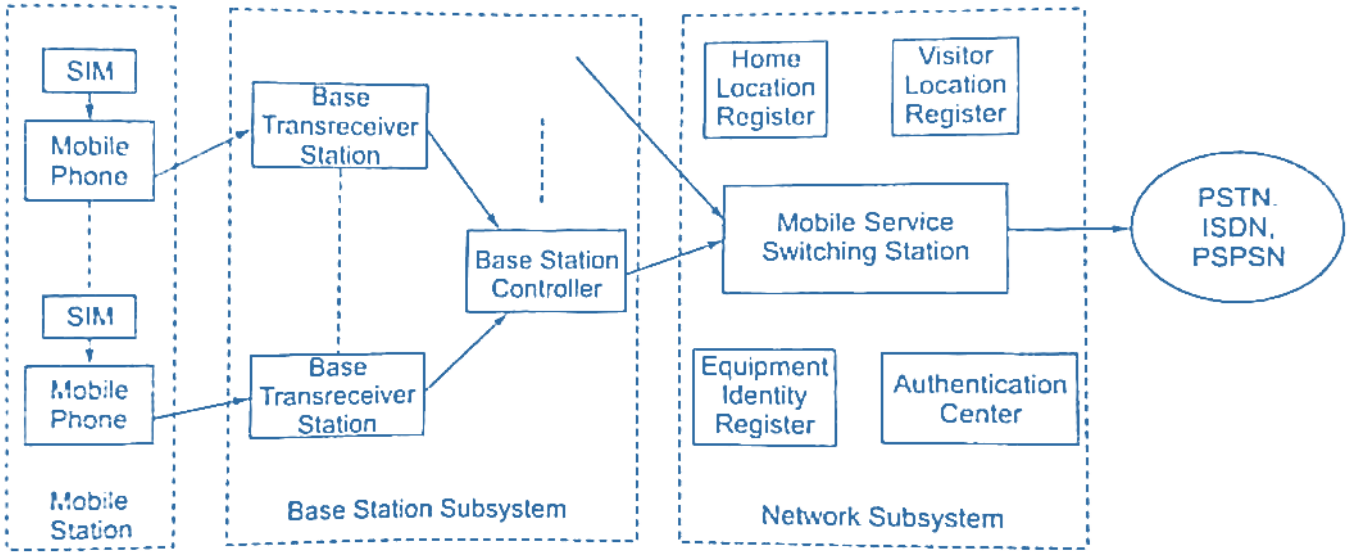


Fig.5.14: Block diagram of GSM system

- The cellular system operates in the 800-900 MHz range. The newer digital cellular system has even greater capacity. Some of these systems operate in 1.7-1.8GHz bands.

Cellular Telephone unit

- Fig.15 shows the block diagram of a cellular mobile radio unit. The unit consists of five major parts:
    i. Transmitter
    ii. Receiver
    iii. Synthesizer
    iv. Logic unit
    v. Control unit

Transmitter

- It is the low power FM transmitter operating in a frequency range of 825 to 845MHz. There is 66630 KHz transmit channel. Transmitter produces a deviation of ±12kHz.
- The modulated output is translated up to final transmitter frequency with the help of mixer, whose second input also comes from frequency synthesizer.
- The unique feature of high power translator is that output is controllable by the cell site & MTSO (mobile telephone switching office).

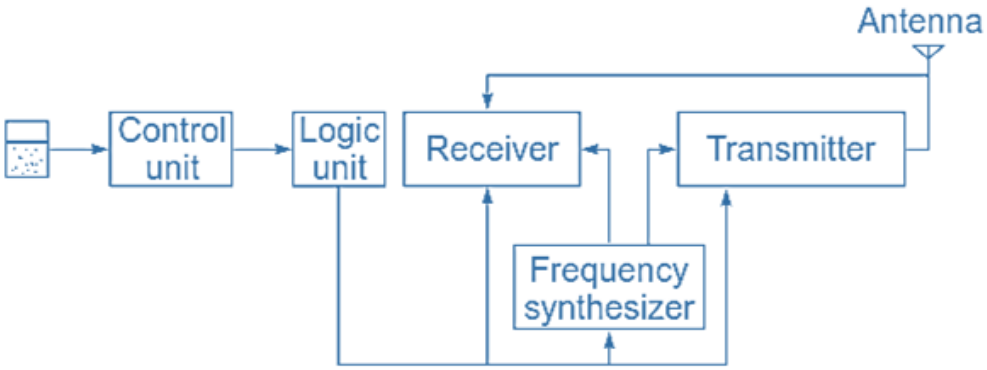


Fig.15: Block diagram of a cellular mobile radio unit

Receiver

- The cellular receiver consists of RF amplifier, FM demodulator and filters.
- An RF amplifier boosts the level of received cell site signal. Received signal is monitored by MTSO.
- If the signal is weak in the present cell then mobile unit is shifted other site where the signal is strong.

Frequency synthesizer

- Frequency synthesizer is used to generate various signals required for transmitter and receiver.
- When a mobile unit initiates a call, MTSO identifies the user and assigns a frequency channel which is not used by any other mobile in the cell. MTSO sends a unique code for setting the channel frequencies.

Logic unit

- Logic unit is micro-processor-controlled master control circuit for cellular radio. it basically controls the complete operation of MTSO and mobile unit.

Control unit

- The control unit is a set of speakers, microphone with touch tone dialing facility and it stores the memory like numbers and dialing features.

Question Bank

1. Convert
    a. $(294.6875)_{10} = (\ )_8$
    b. $(356.15)_8 = (\ )_2 = (\ )_{10}$
    c. $(526.44)_8 = (\ )_2 = (\ )_{10}$
    d. $(48350)_{10} = (\ )_{16} = (\ )_2$
    e. $(342.56)_{10} = (\ )_2 = (\ )_{16}$
    f. $(BCDE)_{16} = (\ )_2 = (\ )_8$
    g. $(35.45)_{10} = (\ )_2$
    h. $(475.25)_8 = (\ )_2$
    i. $(3FD)_{16} = (\ )_2$
    j. $(172.625)_{10} = (\ )_2$
    k. $(ABCD.72)_{16} = (\ )_8$
    l. $(10111101.0101)_2 = (\ )_{10}$
    m. $(A6B.F5)_{16} = (\ )_2$
    n. $(110.111)_2 = (\ )_{10}$
2. Subtract the following using 2'scomplement method
    a. $(101011)_2$ from $(111001)_2$
    b. $(111001)_2$ from $(101011)_2$
    c. $(1010100)_2$ from $(1000100)_2$
    d. $(111)_2$ from $(10100)_2$
    e. $(79)_{10}$ from $(56)_{10}$
    f. $(18)_{10}$ from $(23)_{10}$
    g. $(1101)_2$ from $(11010)_2$
    h. $(10011)_2$ from $(10010)_2$
3. Prove the following Boolean identity using truth table
    a. $A + AB = A$
    b. $A + \bar{A}B = A + B$
4. Simplify the following Boolean expression and realize using basic gates
    a. $Y = A(\overline{ABC} + A\bar{B}C)$
    b. $Y = ABC + A\bar{B}C + AB\bar{C} + \bar{A}BC$
    c. $Y = (\overline{A + B})(\bar{A} + \bar{C})(\bar{B} + C)$

5. Simplify $Y = AB + ABC + \overline{A}B + A\overline{B}C$ and construct logic circuit
6. Show that
    a. $\overline{\overline{AB} + \overline{A} + AB} = 0$
    b. $AB + A(B + C) + B(B + C) = B + AC$
    c. $A\overline{B}C + B + B\overline{D} + AB\overline{D} + \overline{A}C = B + C$
7. Simplify and realize the following expression using only NAND and NOR
    a. $Y = (A + \overline{B})(B + C)(\overline{C} + \overline{B})$
    b. $Y = AB + AC + BD + CD$
8. State and prove De-Morgan's theorem
9. Explain the construction of an OR gate and AND gate using diodes
10. Realize basic gates using NAND gate and NOR gate
11. Design half adder and implement it using logic gates
12. Design full adder and implement it using logic gates
13. Design full adder circuit and implement it using two half adders.
14. What is a multiplexer? Explain the working of 4:1 multiplexer.
15. What is a decoder? Explain the working of 2:4 decoder
16. With the help of a logic diagram and truth table, explain the working of a clocked SR flip-flop.
17. With a neat circuit diagram and truth table, explain the working of a JK flipflop.
18. What is a flip flop? Explain the Master Slave JK flip flop operation.
19. What is a shift register? Explain the working of a 4-bit SISO shift register.
20. What is a counter? With a neat timing and block diagram, explain three bit asynchronous counter operation.
21. With a neat diagram, explain the working of a communication system.
22. With a neat block diagram explain the operating principle of the GSM system