CHAPTER 1

Introduction

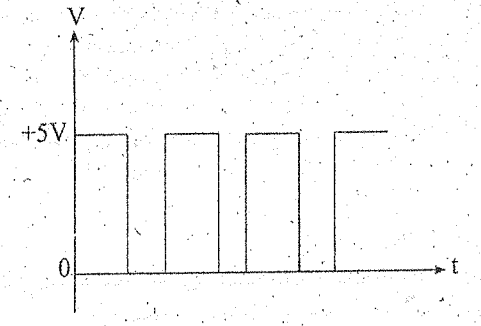
1.1 Definition for Digital Signal:- [2069 Ashad]

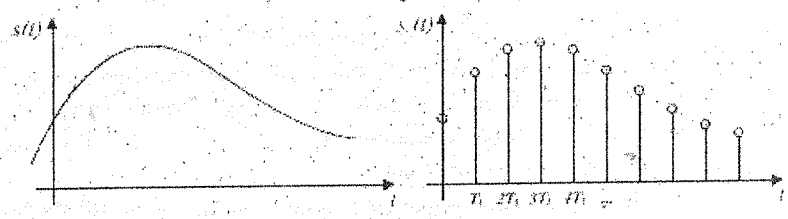
Digital signals are discrete time signals generated by digital modulation. It is a signal where amplitude can have only given discrete values between defined limit. Simply, it a signal that change amplitude is discrete steps.

Digital signals are obtained when discrete time signals are quantized and then coded. Generally digital signals are less subjected to deterioration during transmission. Digitals signals are denoted by square wave. The output of digital computer is an example of digital signal.

1.2 Digital Wave form :-

The waveform of a digital signal is knows as digital waveform. A typical digital waveform is shown in the figure below,





It is clear from above waveform that digital signal changes amplitude in discrete step

The digital waveform shown above has two precise voltage level +5V and 0v dc.

1.3 Digital Logic :-

Digital logic is the representation of signals and sequences of digital circuit through numbers. It is the basis for digital computing and provides a fundamental understanding on how circuits and hardware communicate within a computer. Digital logic is typically embedded into most electronic devices including calculators, computers, video games and watches.

Digital logic involves the study of digital electronics in logical way. It is of two types positive logic and negative logic. We shall deal about positive and negative logic in chater-2.

1.4 Moving and Storing Digital Information :-

In digital electronics it is necessary to move and store digital information. For the purpose of storing digital information, a digital memory element is used. A digital memory element is a device or a circuit that will maintain a desired logic level at it output till it is changed by changing the input condition. The simplest memory element is switch. The simplest electronic circuit used as a memory element is called flip-flop. The flip-flop can be used to store a logic level (high or low) and it will retain a stored logic level indefinitely provided the dc supply voltage is maintained. A group of flip-flop can be converted together to store more than a single logic level.

A group of flip-flop used to store a binary number is called a register.

A register is also capable of shifting (moving) the binary information stored in it. Such register are called shift register. A register capable of shifting in one direction only is a unidirectional shift register. One that can shift in both directions is a bi-directional shift register.

Besides switch, flip-flop and registers for storing and moving digital information other devices for same purpose are magnetic and optical devices such as floppy disk, hard disk drive, magnetic tape drive, optical disk drive etc.

1.5 Digital Operation:-

The operations performed in digital electronics are called digital operations. Some common digital operations are counting, arithmetic operation and logic operations.

The counting operation is performed by "**Counters**". The arithmetic operation is performed by Arithmetic and logic unit and are addition, subtraction, multiplication and division and are accomplished with other digital circuits. The logic operations too are preformed by ALU and they include inversion (NOT), AND and OR.

Selecting a single output of multiple inputs (MULTIPLEXING) or giving out many outputs with single inputs (DEMULTIPLEXING) are can also be treated as digital operation. Similarly the process of encoding and decoding are also digital operations. These operations are preformed by different data processing circuits like Multiplexes, de-multiplexes, encoder, decoder etc.

1.6 Digital computer:-

A digital computer is a machine that stores data in a numerical format and performs operations on that data using mathematical manipulation. This type of computer typically includes some sort of device to store information, some method for input and output of data and components that allows mathematical operations to be performed on stored data. Digital computers are almost always electronic but do not necessarily need to be so.

In short, digital computer is a computer that performs calculations and logic operations with quantities represented as digit usually is binary number system.

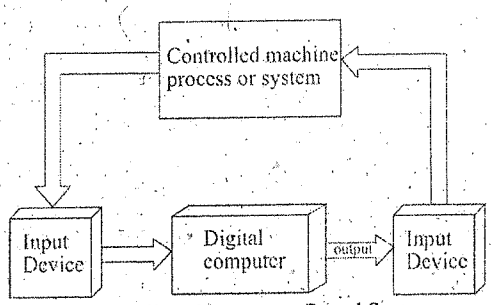


Fig:- A digital computer Based System.

1.7 Digital Integrated Circuits :-

A integrated circuits or IC is a silicon semiconductor crystal, called a chip, containing the electronic components (resistor, transistors, diodes, capacitors etc) for constructing logic gates. The various gates are inter-connected inside the chip to form the required circuit.

Digital integrated circuits are those circuits, which perform logic functions with the help of binary numbers 0 and 1; such as logic gates, flipflop, counters, shift registers etc. Digital IC are mostly popular in realization of electronic systems in the areas of instrumentation, communication, controls and computers. Digital integrated circuits operates with binary signals and are invariably constructed with ICs.

We shall in deal with Digital Integrated Circuits in Chapter-9 in further detail.

1.8 Digital IC Signal levels:-

The voltages that are used to define the two digital logic levels, H=1=T and L=0=F is as shown below,

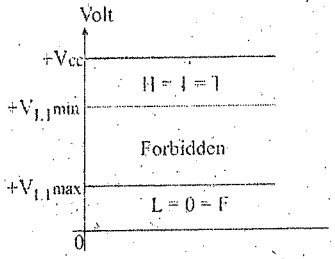


Fig:- Logic Level Profile

IC signal levels are of two types – TTL logic and CMOS logic level.

(1) **TTL Logic levels :-** The input and output profile of a TTL logic level is as shown,

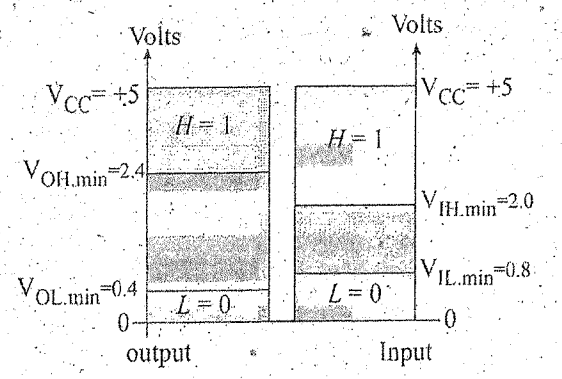


Fig:- TTL logic level profile

(2) **CMOS logic levels:-** The Input and output profile of a CMUS logic level is as shown,

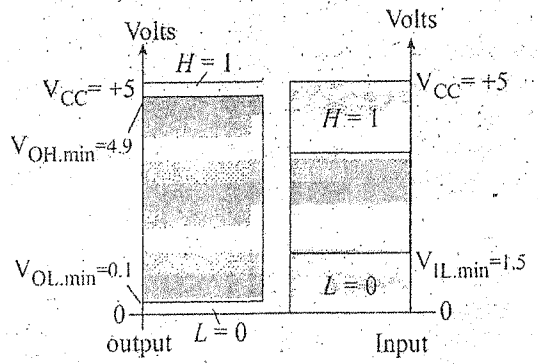


Fig:- CMOS logic level profile

1.9 Clock Wave form:-

A symmetrical Waveform or signal which is frequently used as a basis for timing all operations in a digital system is called clock waveform. A clock waveform is as shown,

Fig:- Clock Waveforms

The electronic circuit used to generate square wave is referred to as the system clock.

1.10 Coding:-

A code is a symbolic representation of discrete information which may be present in the form of numbers, letters or any other physical quantity. When Numbers letters or words are represented by a special group of symbols, it is called encoding or simply coding.

Binary codes can be classified as numeric codes and alphanumeric codes. Alphanumeric codes represent the alphabet and decimal numbers as a sequence of 0s and 1s. ASCII code is an example of alphanumeric code. Numeric codes represent decimal digits and are called Binary coded Decimal (BCD) codes.

BCD codes are further classified as weighted and non-weighted codes.

Weighted Code : Weighted codes obey position weighting principle. In weighted codes each position of the number represent a specific weight and for each group of four bits, the sum of the weights of those positions where the binary digit is 1 is equal to the decimal digit which the group represents.

The codes 8421, 2421, 5211, 5421 are some of the weighted codes out of which 8-4-2-1 is commonly used.

Non-Weighted Code: In a weighted code, no definite weights are assigned to the four digit positions such as in excess – 3 code and Gray code. There code does not obey the position weighting principle.

1.10.1 ASCII Code:-

ASCII stands for American Standard Code For Information Inter-charge. The ASCII code is an alphanumeric code widely used for transferring data into and out of a computer. This is a 7-bit code to represent alphabet, letters, numbers and other symbols. It corresponds to $2^7 = 128$ characters.

1.10.2 BCD [2066 Bhadra] :-

BCD stands for Binary Coded Decimal. The BCD code means that each decimal digit is represented by a binary code of four bits. Clearly only the 4-bit binary numbers from 0000 through 1001 are used. The BCD code does not use the numbers 1010, 1011, 1100, 1101, and 1111. In other words, only 10 of the sixteen (2^4) possible 4-bit binary code groups are used. Any decimal number can be expressed in BCD code by replacing each decimal digit by the appropriate 4-bit combination. The case of conversion between BCD code numbers and familiar decimal numbers is the main advantage of this code.

Disadvantage:- BCD requires more bits than straight binary to represent decimal numbers of more than one digit. This is because BCD does not are all possible 4-bit groups and is therefore inefficient.

Table:- 1.1

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

1.10.3 Excess-3 code:- The excess-3 code is an important 4-bit code sometimes used with binary coded decimal (BCD) numbers. To convert any decimal number into its excess-3 form, add 3 to each decimal digit and then convert the sum to a BCD number.

For example:- To convert 12 to excess-3 number, first add 3 to each decimal digit:

```
  1     2
+ 3   + 3
  4     5
```

Second covert the sum to BCD form

$4 \rightarrow 0100$

$5 \rightarrow 0101$

Here 0100 0101 is the excess 3-code for decimal digit 12.

Take another example, convert 29 too an excess 3 number.

```
   2      9
 + 3    + 3
 ---    ---
   5     12
   ↓      ↓
 0100   1100
```

∴ Excess-3 or Xs-3 code for 29 is 0100 1100.

Table given shows excess-3 code. In each case, the excess 3 code number is 3 greater than the BCD equivalent. Such coding helps in BCD arithmetic as 9's complement of any excess-3 coded number can be obtained simply by complementing each bit.

For example, take a decimal digit 2. It's complement is 9-2 = 7. Excess-3 code of 2 is 0101. complementation each bit we get 1010 and its decimal equivalent is 7.

To covert BCD to excess-3 we need an adder and for reverse we need subtractor.

Table: 1.2- Excess-3 code.

Decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

1.10.4 Gray Code [2068 Baishak] [2067 Ashad]:-

This code belongs to a class of codes called minimum change code is which only one bit in the code group changes when going from one step to the next. This is an un-weighted code which means that there are no specific weights assigned to the bit positions. Because of this, the Gray code is not suited for arithmetic operations but finds applications is input/output devices and some types of analog to digital converters-(ADCs)

Advantage:-

In Gray code, if we go from one decimal number to next, only one bit of the Gray code changes. Because of this feature an amount of switching is minimized and the reliability of the switching systems is improved.

Binary to Gray Conversion

Table 1.3 Gray code

Decimal	Binary Code	Gray code	Decimal	Binary Code	Gray Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

1.10.4.1 Binary-To-Gray conversion

For conversion of binary number into a Gray code number, following rules are applied.

1. The most significant digit (left-most digit) in the Gray code is the same as the corresponding digit in the binary number.
2. Going from left to right, and each adjacent pair of binary digits to get the next Gray code digit. Discard carries if any.

Example 1.10.4: obtain the following conversion: $(111011)_{binary}$ to Gray code.

Solution: Step 1 : The left-most Gray digit is the same as the left-most binary digit

1	1	1	0	1	1	Binary code
1						Gray code

Step 2 : Add the left-most binary digit to the adjacent one

1 + 1 1 0 1 1 Binary code

↓

1 0 Gray code

Step 3 : Add the next adjacent binary number

1 1 + 1 0 1 1 Binary code

↓

1 0 0 Gray code

Step 4 : Add the next adjacent binary number

1 1 1 + 0 1 1 Binary code

↓

1 0 0 0 Gray code

Step 5 : Add the next adjacent binary number

1 1 1 0 + 1 1 Binary code

↓

1 0 0 1 0 Gray code

Step 6 : Add the next adjacent binary number

1 1 1 0 1 + 1 Binary code

↓

1 0 0 1 1 0 Gray code

The complete conversion is thus $(100110)_{Gray}$.

1.10.4.2 Gray-To-Binary conversion

For conversion of Gray code number into a binary code number, the applicable rules are as follows:

1. The most significant digit (left-most digit) in the binary code is the same as the corresponding digit in the Gray code.
2. Add each binary digit generated to Gray digit in the next adjacent position. Discard carries if any.

Example 1.10.4.2 Convert the Gray code number 11011 into the binary number.

Solution: Step 1 : The left-most digit are the same.

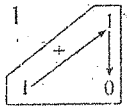
1 1 0 1 1 Gray code

↓

1 Binary code

Step 2 : Add the last binary digit just generated to the Gray digit in the next position. Discard carry.

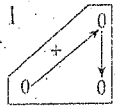
1 0 1 1 Gray code



Binary code

Step 3 : Add the last binary digit just generated to the Gray digit.

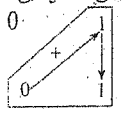
1 1 0 1 1 Gray code



1 Binary code

Step 4 : Add the last binary digit just generated to the Gray digit.

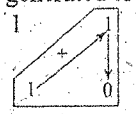
1 1 0 1 Gray code



1 0 Binary code

Step 5 : Add the last binary digit just generated to the Gray digit.

1 1 0 1 1 Gray code



1 0 0 Binary code

Thus conversion is complete and final result is $(10010)_2$.

CHAPTER 2

Digital Logic

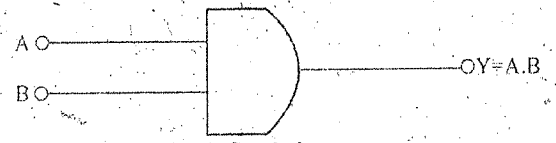
Introduction:-

A digital circuit having one or more input signals but only one output signal is called a logic gate or simply a gate. Logic gates are the basic building blocks of any digital systems.

The table which show different combination of input along with their possible outputs is called truth table.

2.1 The Basic Gates-NOT, OR, AND:-

AND gate:- The AND gate has two or more inputs and a single output. The AND gate provides high output only when all inputs are high. The AND gate performs logical multiplication, more commonly known as AND function.



Logic Symbol

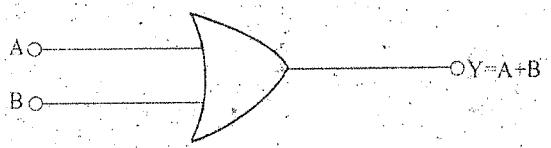
Truth table

Inputs		Output
A	B	Y=A·B
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Equation ,

$$Y = A \cdot B$$

OR gate:- The OR gate has two or more input, and a single output. The OR gate provide high output if any or all inputs are high. The AND gate performs logical addition, more commonly known as OR function.



Logic symbol

Truth table

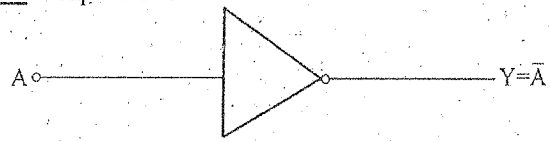
Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Equation,

$$Y = A + B$$

NOT gate:- The NOT gate has only one input, and one output. The NOT gate provide high output if input is low. It provides low output if input is high. The NOT gate performs a basic logical function called inversion or complementation.

The NOT gate is also called inverter because output state is always **opposite** to input state.



Logic symbol

Truth table

Input	Output
A	Y
0	1
1	0

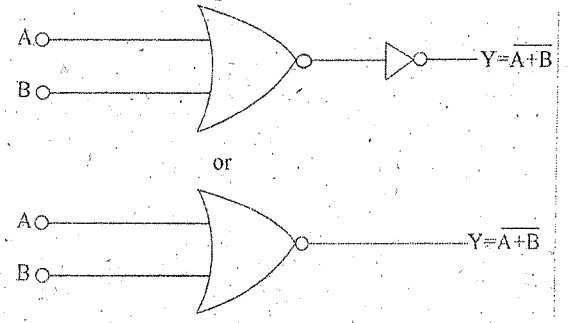
Boolean Equation ,

$$Y = \overline{A} \text{ or } A'$$

2.2 Universal Logic Gate-NOR, NAND:- [2068 Chaitra, Shrawan]

NOR Gate :- It is the combination of NOT and OR gate in such a way that output of OR gate is connected to the input of NOT gate.

The output of the gate is high only when input are low. Hence its outputs are reverse of OR gate.



Logic symbol

Truth table

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

Boolean Equation ,

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

NAND Gate:- It is the combination of NOT and AND gate in such a way that output of NAND gate is connected to the input of NOT gate. The output of this gate is low only when all inputs are high. Hence out this gate is revere of AND gate.

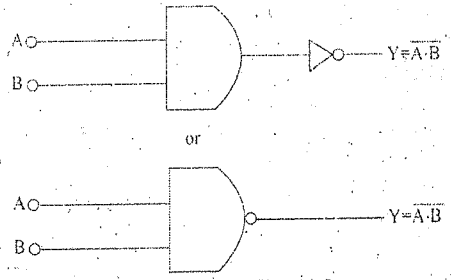


Fig:- Logic Symbol

Truth table

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

Boolean Equation ,

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

Bubbled Gate:- A Bubbled gate is one whose input are inverted . It is a negated gate, AND gate with inverted inputs is as shown in figure below,

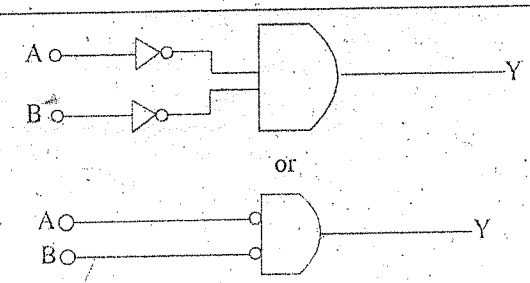


Fig:- AND gate with Inverted Input signals (Bubbled AND gate)

The bubbles on the inputs acts as a remainder of the inversion or complementation that occur before AND operation of inputs. Such gates are called bubbled AND gate.

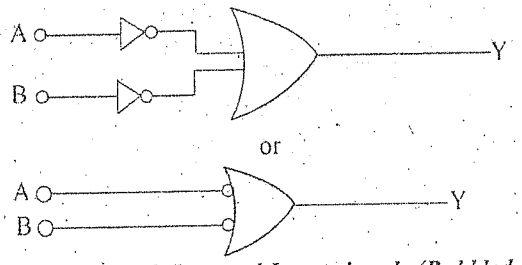


Fig:- OR gate with Inverted Input signals (Bubbled OR gate)

Some other gates:-

Exclusive – OR gate (Ex-OR gate):- The Ex-OR gate or X-OR gate has two or more inputs and only one output. The output will be high if the inputs are of opposite logic. It means that when the two inputs are different (one is high, and another low), the output is high. When two inputs are same (either low or high), the output is low.

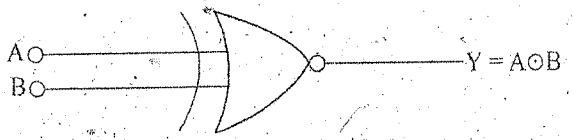


Fig:- Logic Symbol

Truth table

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

Boolean Equation ,

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

Exclusive – NOR gate (Ex-NOR gate):- The Ex-NOR gate or X-NOR gate has two inputs and only one output. The output will be high if the inputs are of same logic. It means when inputs are same (either low or high), the output is high, when the inputs are different (one is low and other is high), the output is low.

Hence X-NOR gate operates just opposite to the EX-OR gate.

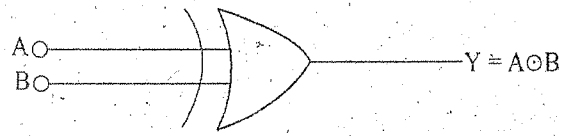


Fig:- Logic Symbol

Truth table

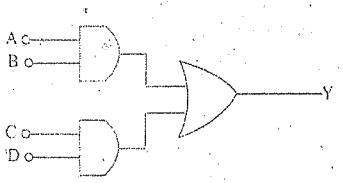
Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Equation ,

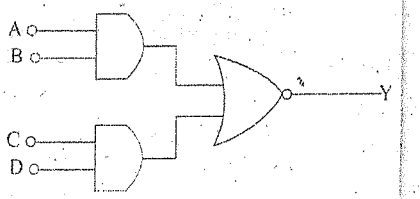
$$Y\overline{A} = A \odot B = AB + \overline{A}\,\overline{B}$$

2.3 AND – OR – INVERT gates

The basic gates AND-OR and INVERT can be interconnected with each other to build and desired logic circuit.

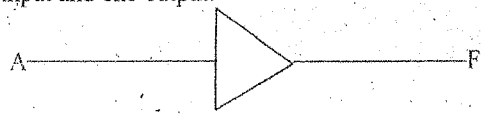


AND-OR circuit



AND-OR-INVERT gate

Buffer gate:- A buffer gate produces the output same as input. It has only one input and one output.



Logic Symbol

Truth table

A	Y
0	1
1	1

2.4 Positive and Negative logic [2069 Ashad]

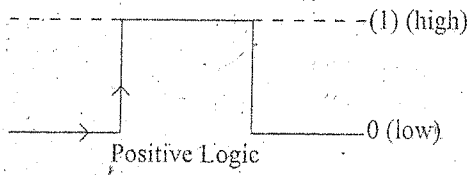
In digital electronics we use two level of voltages ⇒ 0V and +SV dc. Generally we use binary 0 for low voltage (0V) and binary 1 for high voltage (+SV dc). This system of digital logic is called positive logic.

But the system of digital logic where binary 0 represents high voltage and binary 1 represent low voltage is called negative logic.

The following discussion introduces concepts of both types of logic:-

The truth table of OR gate as discussed previously is,

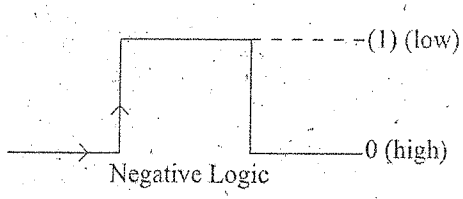
A	B	Y
0	0	0
0	1	1
0	0	1
1	1	1



The truth table shown above is of OR gate in positive logic system where 0 represents low and 1 represents high.

Let us now draw the same truth table in negative logic system where 0 represents high and 1 represents low as shown,

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



Note that this second truth table is of AND gate we have learnt so far. Thus we can conclude that an OR gate is positive logic system becomes an AND gate is negative logic system.

i.e. positive OR ↔ negative AND

Similarly we can show,

positive AND ↔ negative OR
positive NOR ↔ negative NAND
positive NAND ↔ negative NOR

2.5 Introduction to HDL: [2068 shrawan, Baishak]

In electronics, a hardware description language (HDL) is a specialized computer language used to program the structure, design and operation of electronic circuits, and most commonly, digital logic circuits.

A hardware description language enables a precise, formal description of an electronic circuit that allows for the automated analysis, simulation and simulated testing of an electronic circuit.

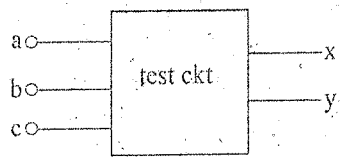
A HDL looks much like a computer programming language such as C; it is a textual description consisting of expression, statements and control structures. One important different between most programming language and HDLs is that HDLs explicitly include the notion of time.

HDLs forms integral part of electronic design automation (EDA) system, especially for complex circuits, such as microprocessors.

Currently there are two widely used HDLs – Verilog and VHDL (Very High Speed Integrated Circuit Hardware Decription Language). Verilog is considered simpler of the two and is more popular.

Verilog HDL or VHDL:-

Describing input/output :- In any digital circuits, we find there are a set of input and set of outputs. Design any circuit that has three inputs a,b,c and two outputs x,y as shown, the corresponding Verilog code can be written as,




```
module test ckt (x,y,a,b,c);  // module name with port list
input a,b,c; // defines input ports
output x,y; // defines output ports.
// module body begins next describing logic relations.
,
,
,
// module body ends
end module.
```

Writing Module Body:- There are three different models of writing module body in Verilog HDL (1st Model)
We start with structural model by example of two input OR gate.

```
Module or_gate (A, B, Y);
Input A, B;          // defines two input port
Output Y;            // defines two output port
Or g1 (Y, A, B);     // Gate declaration with pre-defined keyword
                     or representing logic OR, g1 is optional user
                     defined gate identifier.
end module.
```

2nd Model

Consider a circuit as show,

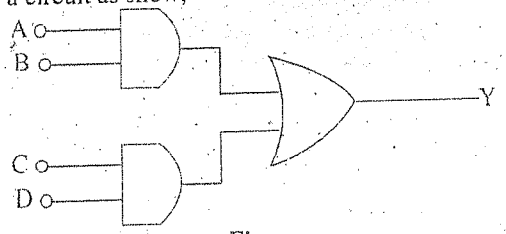
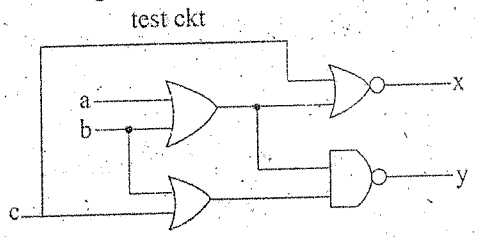


Fig: a

The logic circuit shown above has 4 inputs and 1 output. The inputs are fed two 2-input AND gate. AND gate outputs are fed to a 2-input OR gate to generate final output. The Verilog code for this circuit is as given below.
Note that, we define two intermediate variables and – op1 &, and-op2 representing two AND gate outputs through keyword wire. Wire represents a physical wire in a circuit.

```
module fig a (A, B, C, D, Y);
input A, B, C, D
wire and –op1, and op-2; // internal connections
and g1 (and_op1, A, B); // g1 represents upper AND gate
and g2 (and_op2, C, D); // g2 represents lower AND gate
or g3 (Y, and_op1, and_op2); // g3 represents OR gate
end module.
```

Consider another logic circuit as shown;



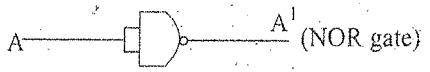
The Verilog code for above circuit can be written as,

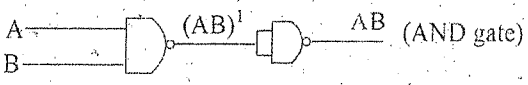
```
module test ckt (a,b,c,x,y)
input a,b,c;
output x,y;
wire or-op1, or-op2; // internal connections
or g1 (or-op1, a, b); // g1 represents upper OR gate)
or g2 (or-op2, b, c); // g2 represents lower OR gate)
nor g3 (x,c, or-op1); // g3 represents the NOR gate
nand g4 (y, or_op1, or_op2); // g1 represents that NAND gate
end module
```

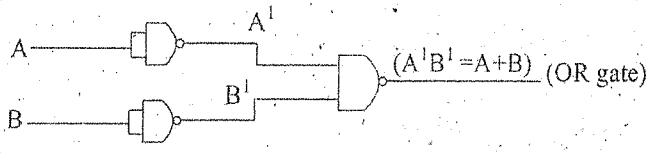
Old Question/Solution

1. *Construct the basic gate using only universal gate. [2068 Baishak] [2068 Shrawan] [2068 Chaitra] [2069 Ashad]*

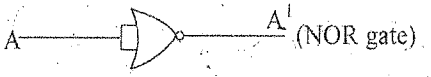
Ans : a) Realization of different basic gates by using universal (NAND) gate:

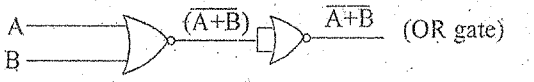


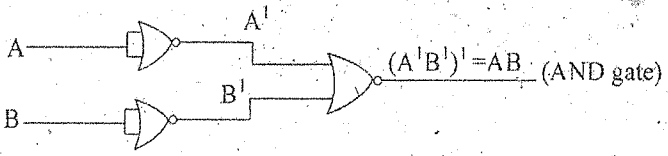




b) Realization of different basic gates by using universal (NOR) gate:







2. *List out the name of universal gates and why they are called universal gate? Realize Ex-OR gate using only NAND gates.*
[2068 Chaitra]

And : We know the Boolean expression for the Ex-OR gate is
$Y = A \oplus B = AB' + A'B$
Where A and B are two input part and Y is output

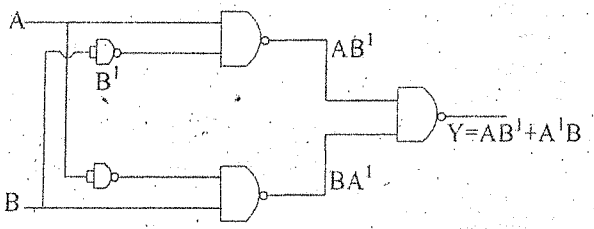


Fig:-Realization of Ex-OR gate using NAND gates only

3. *When FF_H is ANDed with CO_H what will be the resulting numbers?*
[2065 Shrawan]

Ans : Truth table for AND gate

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

We know,
$FF_H = (1111 \quad 1111)_L$
$Co_H = (1100 \quad 0000)_L$
When ANDed = 1100 0000
$\therefore$ Ans = Co_H

4. *Draw the standard symbol and logic circuit of two-input exclusive NOR gate. Write in Boolean equation and truth table. A NAND gate has seven inputs. How many numbers of input combinations you need to complete the truth table of such gate? If on inverters is connected to its output then what logic function it will per form?*
[2064 Jestha]

Ans :

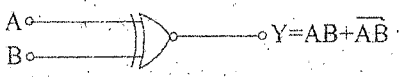


Fig:- Standard symbol of 2-input exclusive NOR gate

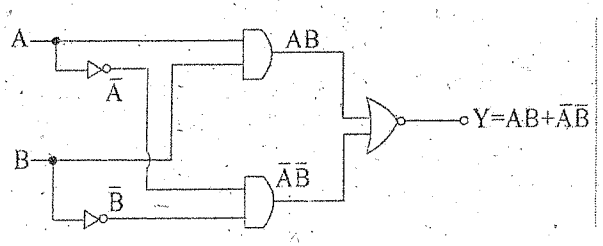


Fig:- Logic Circuit of 2 input exclusive NOR Gate

Boolean expression : $Y = AB + \overline{A}\ \overline{B}$
Truth table:

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

For 2nd part,
If n = 7
Number of combination, N $= 2^n$
$= 2^7$
$= 128$

∴ If 7 inputs are provided, 128 combinations are required to complete the truth table

If on inverter is connected at the output of NAND gate it will perform as an AND gate i.e. output is obtained as ANDing of those inputs.

5. *Realize Ex-OR gate using only NAND gate.* *[2068 Chaitra]*

Ans:

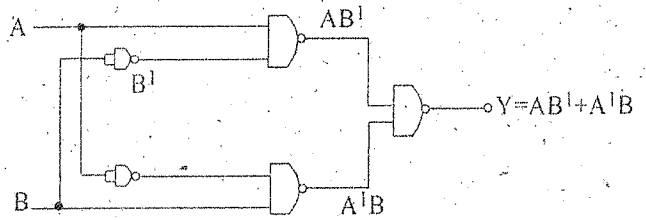
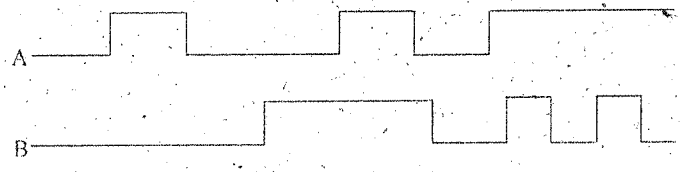
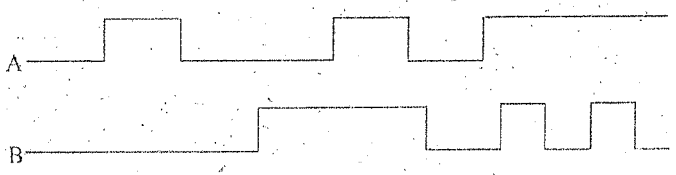


Fig: Realization of Ex-OR gate using NAND gates

6. *If the following two waveforms are applied to XNOR gate, what will be the output waveform?* *[2066 Bhadra]*



Sol^{n}:



The truth table for X-NOR gate is,

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

CHAPTER 3

Combination Logic Circuits

Introduction:-

A combinational logic circuit is a type of logic circuit which is implemented by Boolean circuits, where the output is a pure function of present input only. This is in contract to sequential logic circuit where output depends not only on present input but also on the history of input.

Combinational logic is used in computer circuits to perform Boolean Algebra on input signals and on stored data.

3.1 Boolean laws and Theorems:-

Basic rules that are useful in manipulation and simplification of Boolean algebra expression are.

(1) OR Rules

$A + 0 = A$ $\quad A + A = A$

$A + 1 = 1$ $\quad A + \bar{A} = 1$

(2) AND Rules

$A \cdot 0 = 0$ $\quad A \cdot A = A$

$A \cdot 1 = A$ $\quad A \cdot \bar{A} = 0$

(3) Complementation Rule

$\bar{\bar{A}} = A$

(4) Absorptive Rules

$A + AB = A$

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

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

Proof :- $A + AB = A$

L.H.S $= A + AB$

$= A(1 + B)$

$= A \cdot 1$ $\quad [\because (1 + B) = 1]$

$= A$

Proof :- $(A + \bar{A}B) = A + B$

L.H.S $= A + \bar{A}B$

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

$= AA + AB + \overline{A}B \quad [\because A \cdot A = A]$

$= AA + AB + A\overline{A} + \overline{A}B \quad [\because A\overline{A} = 0]$

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

$= 1 \cdot (A + B)$

$= A + B$

Proof :- $(A + B)(A + C) = A + BC$

L.H.S $= (A + B)(A + C)$

$= AA + AC + AB + BC$

$= A + AC + AB + BC \quad [\because A \cdot A = A]$

$= A(1 + C) + AB + BC$

$= A \cdot 1 + AB + BC \quad [\because 1 + C = 1]$

$= A(1 + B) + bc$

$= A \cdot 1 + BC \quad [\because 1 + B = 1]$

$= A + BC$

(5) Commutative Rule

$A + B = B + A$

$A \cdot B = B \cdot A$

(6) Distributive Rule

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

(7) Associative Rule

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

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

De-Morgan's Theorem :- **[2067 Ashad] [2066 Bhadra]**

De-Morgan's theorem can be stated in two statements as follows.

(i) The complement of sum of two or more variables is equal to the product of the complements of individual variables.

i.e. If A & B are two variables then,

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

For 3-variables,

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

(ii) The complement of product of two or more variables is equal to the sum of complements of variables

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

For 3-variables,

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

Verification of De-Morgan's First Theorem,

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

Verification of De-Morgan's Second Theorem,

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

These theorems are illustrated by gate equivalences as shown,

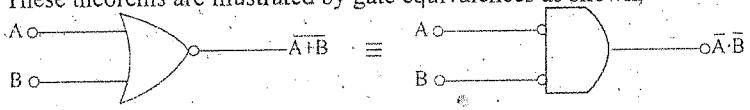


Fig: Illustration of De-Morgan's First Theorem

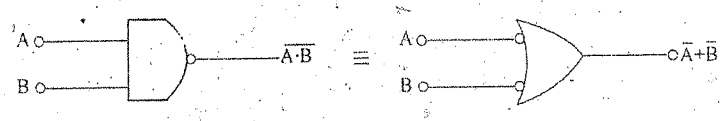


Fig: Illustration of De-Morgan's Second Theorem

Note :- De-Morgan's theorem is extremely useful in simplification of Boolean expression in which sum or product of variables is inverted.

Duality Theorem : It states that,

"Each Boolean expression has its dual which is as true as original expression".

The dual of a given Boolean expression can be obtained by following the procedure given below:

(1) Change each OR (+) sign to an AND (·) sign

(2) Change each AND (·) sign to an OR (+) sign

(3) Complement any 0 or 1 appearing in the expression.

Some of the Boolean relations and their duals are given below,

Relation	Dual Relation
$A + 0 = A$	$A \cdot 1 = A$
$A + 1 = 1$	$A \cdot 0 = 0$
$A + A = A$	$A \cdot A = A$

$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
$A + B = B + A$	$A \cdot B = B \cdot A$
$A + (B+C) = (A+B)+C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
$A + (B+C) = AB + AC$	$A + (B \cdot C) = (A+B) \cdot (A+C)$
$\overline{A \cdot B} = \overline{A}\ \overline{B}$	$\overline{AB} = \overline{A} + \overline{B}$
$A + AB = A$	$A(A+B) = A$
$A + \overline{A}B = A+B$	$A \cdot (\overline{A} + B) = A \cdot B$

Minterms and Maxterm

Boolean function : A Boolean function is described by an algebraic expression consists of binary variables, the constant 0 and 1 and the logic operation systems for a given value of binary variables, the function can be equal to either 1 or 0. E.g. $F_1 = \overline{X}\,\overline{Y}\,Z + X\overline{Y}\,\overline{Z} + XYZ$ Each of binary variable may appear either in its normal form (X) or in its complement form ($\overline{X}$). Since each variable may appears in either form $\overline{X}\,\overline{Y}, \overline{X}\,Y, X\overline{Y}, XY$. Each of the these four AND term is called a minterm or a standard product. Each minterm is obtained from an AND term of n variables with each variable being primed if the corresponding bit of the binary number is zero (0) and unprimed if a 1 which is shown below and each represented by m_j where j denote equivalent binary number.

In a similar fashion forming an OR term, with each variable being primed corresponding and unprimed provide 2^n possible combinations called max terms or standard sum represented by M_j. Minterm and Maxterm for 3 binary variables can be shown as in given table.

			Min term		Max term	
X	Y	Z	Term	Designation	Term	Designation
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	$X+Y+Z$	M_0
0	0	1	$\overline{X}\,\overline{Y}\,Z$	m_1	$X+Y+\overline{Z}$	M_1
0	1	0	$\overline{X}\,Y\overline{Z}$	m_2	$X+\overline{Y}+Z$	M_2
0	1	1	$\overline{X}\,YZ$	m_3	$X+\overline{Y}+\overline{Z}$	M_3
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	$\overline{X}+Y+Z$	M_4
1	0	1	$X\overline{Y}\,Z$	m_5	$\overline{X}+Y+\overline{Z}$	M_5
1	1	0	$XY\overline{Z}$	m_6	$\overline{X}+\overline{Y}+Z$	M_6
1	1	1	XYZ	m_7	$\overline{X}+\overline{Y}+\overline{Z}$	M_7

Question: Write the function output interms of Minterms and Maxterms.

X	Y	Z	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

For minterm,

$F_1 = \overline{X}\,\overline{Y}\,Z + X\overline{Y}\,\overline{Z} + XY\overline{Z}$
$= m_1 + m_4 + m_7$
$= \sum (m_1, m_4, m_7)$

$F_2 = \overline{X}\,YZ + X\overline{Y}\,Z + XY\overline{Z} + XYZ$
$= m_3 + m_5 + m_6 + m_7$
$= \sum (m_3, m_5, m_6, m_7)$

For maxterm

$F_1 = (X+Y+Z)\,(X+\overline{Y}+Z)\,(X+\overline{Y}+\overline{Z})\,(\overline{X}+Y+\overline{Z})\,(\overline{X}+\overline{Y}+Z)$
$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

The Standard SOP Form

In SOP standard form, every variable in the domain must appear in each term. You can expand a nonstandard term to standard form by multiplying the term by a term consisting of the sum of the missing variable and its complement.

Example: Convert $X = \overline{A}\,\overline{B} + A\,B\,C$ to standard form.

Solution : The first term does not include the variable C. Therefore, multiply it by the $(C + \overline{C})$, which is equal to 1:

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

The Standard POS Form

In POS standard form, every variable in the domain must appear in each sum term of the expression. You can expand an nonstandard POS expression to standard form by adding the product of the missing variable and its complement and applying rule 12, which states that

$A + BC = (A + B)\,(A + C)$

Example: Convert $X=(\overline{A}+\overline{B})(A+B+\overline{C})$ to standard form.

Solution: The first sum term does not include the variable C. Therefore, add $C\cdot\overline{C}$, which is equal 0, and expand the result by rule 12.

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

Example: Convert $X=A(B+C)(A+\overline{B}+C)$ into standard POS form.

$$=(A+B\overline{B})(B+C)(A+\overline{B}+C)$$
$$=(A+B)(A+\overline{B})(B+C)(A+\overline{B}+C)$$
$$=(A+B+C\overline{C})(A+\overline{B}+C\overline{C})(B+C+A\overline{A})(A+\overline{B}+C)$$
$$=\prod(M_0, M_2, M_3, M_5, M_6)$$

F_1 $=(X+Y+Z)(X+Y+\overline{Z})(X+\overline{Y}+Z)(X+\overline{Y}+\overline{Z})$
$=M_0, M_1, M_2, M_4$
$=\prod(M_0, M_1, M_2, M_4)$

Sum-of-Products(SOP) and Product-of-Sums(POS)Forms:

Boolean expressions can be written in the sum-of-products form (SOP) or in the product-of-sums form (POS). There forms can simplify the implementation of combinational logic, particularly with PLDs. In both forms, an over bar cannot extend over more than one variable.

An expression is in SOP form when two or more product terms are summed as in the following examples;

$\overline{A}\,\overline{B}\,\overline{C}+AB$ $\qquad$ $AB\overline{C}+\overline{C}\,\overline{D}+\overline{E}$

An expression is in POS form when two or more sum terms are multiplied as in the following examples :

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

Example: Convert each of the following Boolean expressions to SOP form:

(a) $AB+B(CD+EF)$ $\qquad$ (b) $(A+B)(B+C+D)$ $\qquad$ (c) $((A+B)+C)$

Solution:

(a) $AB+B(CD+EF)=AB+BCD+BEF$

(b) $(A+B)(B+C+D)=AB+AC+AD+BB+BC+BD$

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

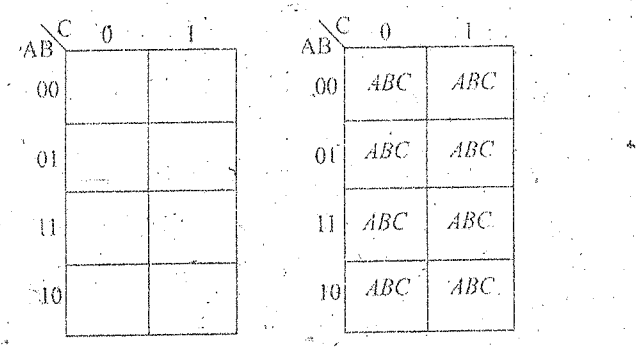
The Karnaugh Map

It is similar to a truth because it represents all possible values of input variables and the resulting output for each value. Instead of being organized into columns and rows like true table, the K, map is an array of cells in which each cell represents a binary value of the inputs variables. The cells arranged in a way so that simplification of a given expression is simple a matter of properly grouping the cells.

The number of cells in K, maps, as well as the number of rows in truth tables, is equal to the total number of possible combinations. For 3 variables, the number of cells is $2^3 = 8$. For 1 variables, the number of cells is $2^4 = 16$. The K-maps can be used for expressions up to 5 variables.

3-Variable K- Map:

The 3-variable K. map is an array of 8 cells, as shown in figure.



3-Variable K-Map

Cells are usually labeled using 0's and 1's to represent the variable and its complement. The numbers are entered in gray code. To force adjacent cells to be different by only one variable 1's are read as the true variable and 0's are read as the complemented variable. The value 0's given cell is the binary values of A and B combined with the value of C, for each corresponding row and column.

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

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

Conversion between Canonical forms:

The conversion of minterm to Maxterm (SOP to POS):

1. Choose the output for 0's
2. Write this in Min terms forms.
3. Apply Demorgan's theorem to obtain in Max term.

From the above example;

$F_1 = \overline{X}\,\overline{Y}\,Z + X\overline{Y}\,\overline{Z} + XYZ$ (min term form by choosing for 1's)

$\overline{F_1} = \overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}\,YZ + \overline{X}\,YZ + X\overline{Y}\,Z + XY\overline{Z}$

Taking complement on both sides.

$\overline{F_2} = \overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}\,YZ + \overline{X}\,YZ + X\overline{Y}\,Z + XY\overline{Z}$

Appling Demorgan's theorem

$F_1 = \overline{X}\,\overline{Y}\,\overline{Z} \cdot \overline{X}\,YZ \cdot \overline{X}\,YZ \cdot X\overline{Y}\,Z \cdot XY\overline{Z}$

$F_1 = (X+Y+Z)\,(X+\overline{Y}+Z)\,(X+\overline{Y}+\overline{Z})\,(\overline{X}+Y+\overline{Z})\,(\overline{X}+\overline{Y}+Z)$

$= M_0 . M_2 . M_3 . M_5 . M_6$

$= \prod(M_0, M_2, M_3, M_5, M_6)$

Mapping a standard SOP Expression

For an SOP expression in standard form, a 1 is placed on the K, map for each product term the expression. Each 1 is placed in a cell corresponding to the value of a product term.

The figure below, illustrates mapping the expression : $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\overline{B}\,\overline{C} + A\overline{B}\,C$ into the 3-variable K, map

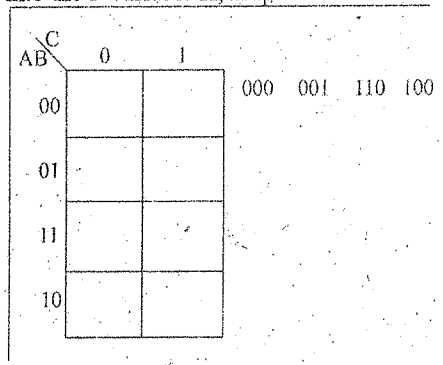


Fig:- Illustration on Mapping of Standard SOP expression

Mapping a non-standard SOP Expression

A Boolean expression must first be in standard form before you use the K. map SOP expression is not in standard form, then it must be converted to standard form.

Example: Map the following SOP expression on a K. map: $\overline{A} + A\overline{B} + AB\overline{C}$

Solution: First expand the terms numerically as follows:

$\overline{A}$ +	$A\overline{B}$ +	$AB\overline{C}$
000	100	110
001	101	
011		

Variable K. Map

The 4-variable K. map is an array of 16 cells, as shown below:

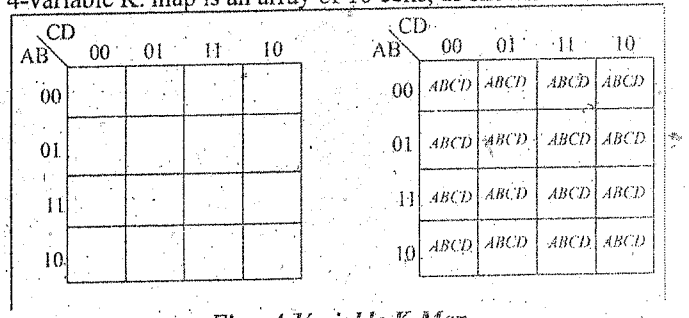


Fig:- 4-Variable K-Map

The cells in K. map are arranged so that there is only a single-variable change between adjacent cells. Physically, each cell is adjacent to the cells that are immediately next to it on any of its 4 sides. The cells in the top row are adjacent to the corresponding cells in the bottom row, and the cells in the outer left column are adjacent to the corresponding cells in the outer right column, as shown below:

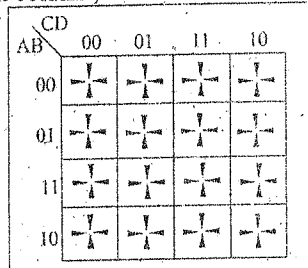
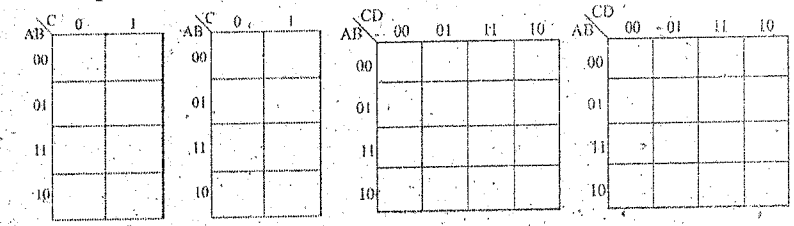


Fig:- Technique of arrangement of adjacent of cells

4. **Each 1 on the map must be included in at least one group.**

Example Group the 1's in each of the following K. maps



Solution:

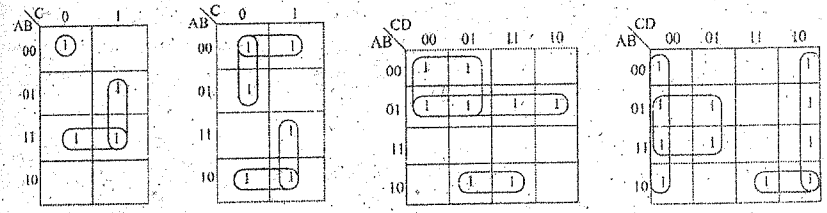


Fig:- Illustration on grouping of SOP expression

Determining the minimum SOP Expression from the Map

The following rules are applied to find minimum product terms and the minimum SOP expression.

1. Each group creates one product term composed of all variables that occur in only one form (either un-complemented or complemented) within the group.
2. Variables that occur both un-complemented and complemented within the group are eliminated (these are called contradictory variables).

Thus the expression in standard SOP form will be as follows:

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

Now you can map each of the product terms by placing a 1 in the appropriate cell of the 3-variable K. map as shown below:

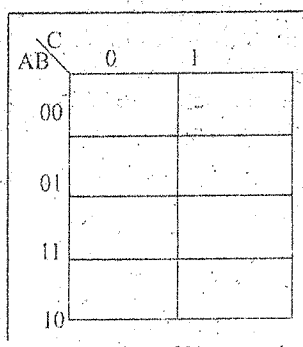


Fig:- Illustration on mapping of Non-standard SOP expression

K-Map SOP Minimization

K. map is used for simplifying Boolean expressions to their minimum form. A minimized SOP expression contains the fewest possible terms with the fewest possible variables per term. Generally, a minimum SOP expression can be implemented with fewer logic gates than the standard expression.

After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map.

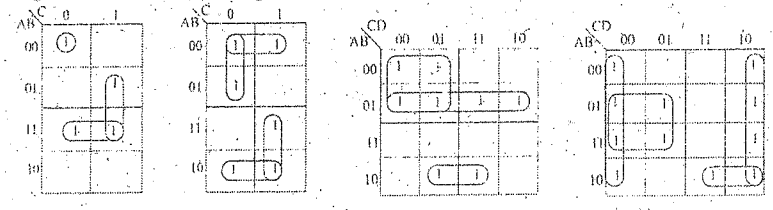
You can group 1s on the K. map by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and minimize the number of groups according to the following rules:

1. A group must contain either 1,2,4,8 or 16 cells, which are all power of 2. In case of a 3-variable map. $2^3 = 8$ cells, is the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.

3. Always include the largest possible number of 1s in a group (in accordance to rule 1).
   - The group of four 1s produces a 1-variable term, B'. This is determined by observing that within the group, B' is the only variable that does not change from cell to cell.
   - The group of two is produces a 2-variable term, A'C. This is determined by observing that within the group, A' and C do not change from one cell to the next.
   - The resulting minimum SOP expression is : $\overline{B} + \overline{A}\,C$

Example:

Determine the product terms for each of the Karnaugh maps in Fig, and write the resulting minimum SOP expression.



The resulting minimum product term for each group is shown in figure. The minimum SOP expression for each of the Karnaugh maps in the figure are:

(a) $AB + BC + \overline{A}\,\overline{B}\,\overline{C}$ (b) $\overline{B} + \overline{A}\,\overline{C} + AC$

(c) $\overline{A}\,B + \overline{A}\,\overline{C} + A\overline{B}\,D$ (e) $\overline{D} + A\overline{B}\,C + B\overline{C}$

Don't Care Condition:

Sometimes a situations arises in which some input variable combinations are not allowed. For example, recall that in the BCD code there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111. Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output. That is, for these "don't care" terms either a 1 or a 0 may be assigned to the output: it really does not matter since they will never occur. The "don't care" terms can be used to advantage on the Karnaugh map. Fig. shows that for each "don't care" an x is placed in the cell. When grouping the 1 s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be. The truth table in Fig. (a) describes a logic function that has a 1 output only when the BCD code for 7,8 or 9 is present on the inputs. If the "don't cares" are used as 1s, the resulting expression for the function is $A + BCD$, as indicated in part (b). If the "don't cares" are not

When all the minimum product terms are derived from the K. map, they are summed to form the minimum SOP expression

Notes: For a 3-variable map:

- A 1-cell group yields a 3-variable product term
- A 2-cell group yields a 2-variable product term
- A 4-cell group yields a 1-variable term
- An 8-cell group yields a value of 1 expression

Example: Use the K, map to minimize the following standard SOP expression:

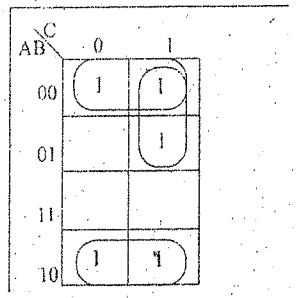
$A\overline{B}\,C + \overline{A}\,BC + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\overline{B}\,\overline{C}$

Solution:

The binary values of the expression are:

101 + 011 + 001 + 000 + 100

Map the standard SOP expression and group the cells as follows:



Notice that:

- The "wrap around" 4-cell group includes the top row and the bottom row of 1s.
- The remaining 1 is absorbed in an overlapping group of two cells.

Which the expression is 1. Usually, when working with POS expressions, the 1s are left off. The following steps and the illustration in Figure show the mapping process.

Step 1: Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.

Step 2: As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.

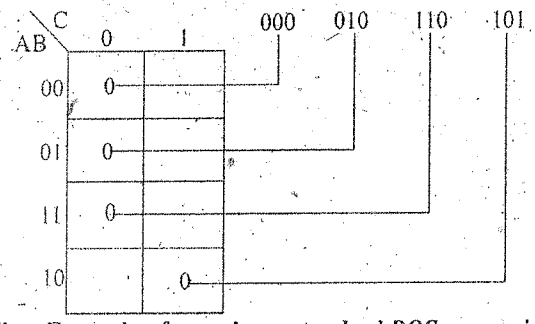


Fig:- Example of mapping a standard POS expression

Example:

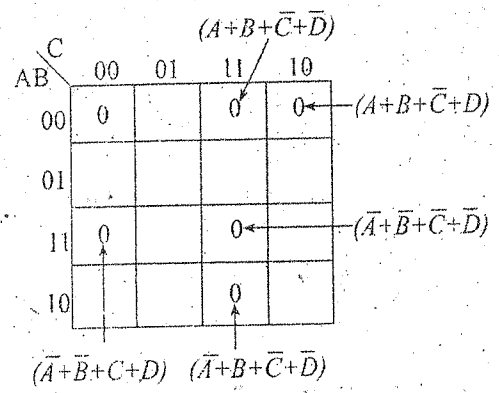
Map the following standard POS expression on a Karnaugh map:

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

Solution:

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

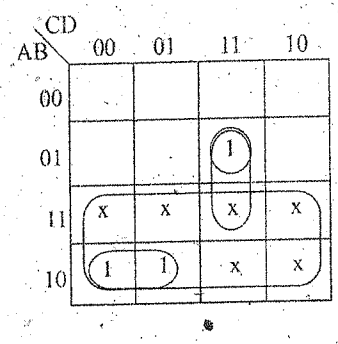
1100 1011 0010 1111 0011



used as 1s, the resulting expression is $ABC + ABCD$: so you can see the advantage of using "don't case" terms to get the simplest expression.

Inputs A B C D	Output Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

Truth Table



Without don't care : $\overline{ABC} + \overline{A}\,BCD$

With don't care : $A + BCD$

Fig:- Implementation of Don't Care condition on SOP expression

Karnaugh Map POS Minmization:

In this section, we will focus on POS expressions. The approaches are much the same except that with POS expressions. 0s representing the standard sum terms are placed on the Karnaugh map instead of 1s. For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression. Each 0 is placed in a cell corresponding to the value of a sum term. For examples, for the sum term A + B + C, a 0 goes in the 0 1 0 cell on a 3-variable map. When a POS expression is completely mapped, there will be a number of 0s on the Karnaugh map equal to the number of sum terms in the standard POS expression. The cells that do not have a 0 are the cells form.

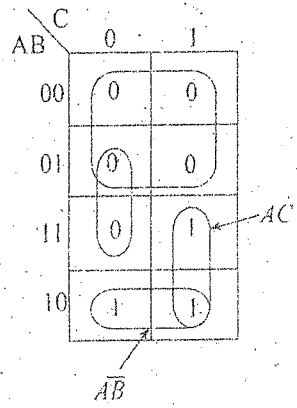
Karnaugh Map Simplification of POS Expressions

The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms. The rules for grouping the 0s are the same as those for grouping the 1s that you learned before.

Example:

Use a Karnaugh map to minimize the following standard POS expression: Also write SOP expression:

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



Hence, Minimize expression in POS form is $A(\overline{B}+C)$ also minimize in SOP form is $AC + A\overline{B}$

Example: Use a Karnaugh map to minimize the following POS expression:

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

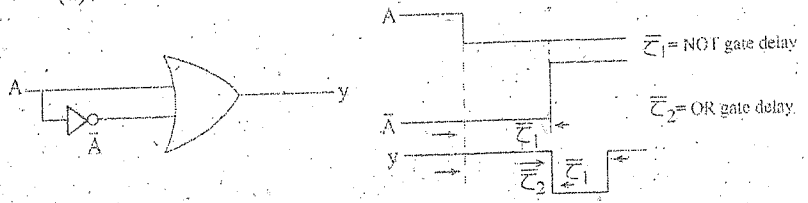
Hazards and Hazard covers

A Hazard is an unwanted switching transient (spike or glitch) that occurs due to finite propagation delay of logic circuits.

The additional gates in the logic circuits to prevent hazard is known as hazard cover.

Types of Hazard

(1) **Static 1-Hazard :**

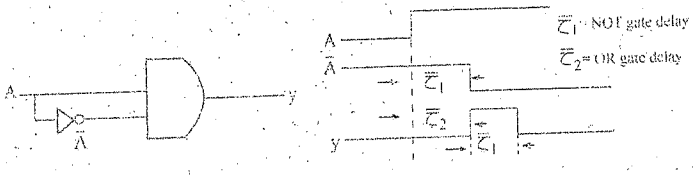


Static-1 Hazard

This type of hazard occurs when Y= A + A' type of situation appears for certain combination of other inputs and A make transition 1→ 0. An A+A' condition should always generate 1 at the output, i.e. static-1. But NOT gate output takes a finite time to become 1 following 1→ 0 transition of A.

(2) **Static 0-Hazard:**

This type of hazard occurs when Y=A, A' kind of situation occurs in a logic circuit for certain combination of other input and A makes a transition 0→1. An AA' condition should always generate 0 at the output i.e. static – 0. But the NOT gate output takes a finite time to become 0 following 0 → 1 transition of A.



(3) **Dynamic Hazard:**

Dynamic hazard occurs when circuit output makes multiple transition before it settles to a final value while the logic equation ask for only one transition. An output transition designed 1→ 0 may give 1 → 0→ 1→ 0 when such hazard occurs and a 0 → 1 can behave like 0 → 1 →0 → 1. The output of logic equation in dynamic hazard degenerates into Y=A+A'A or Y=(A+A'). A kind of relations for certain combinations of other input variables.

Old Question/Solution

1. ***Simplify the following using K-map $\sum m$ (3, 4, 6, 8, 10, 15) + d (0, 2, 7, 14)*** ***[2069 Ashad]***

Ans: Given Function, $\sum m$ (3, 4, 6, 8, 10, 15) + d (0, 2, 7, 14)

Now using K-Map

AB \ CD	00	01	11	10
00	×		1	×
01	1		×	1
11			1	×
10	1			1

Now expressing in the sum of product (SOP) expression F(A, B, C, D) = $A^1D^1 + A^1C + BC + B^1D^1$

2. ***Simplify the function using K-map $F = \sum(0, 1, 4, 8, 10, 11, 12)$ and $D = \sum(2, 3, 6, 9, 15)$. Also convert the result into standard minterm.*** ***[2068 Chaitra]***

Ans : Given function

F(A, B, C, D) = $\sum$ (0, 1, 4, 8, 10, 11, 12)

And D (A, B, C, D) = $\sum$(2, 3, 6, 9, 15)

Now using K-map;

AB \ CD	00	01	11	10
00	1	1	×	×
01	1			1
11	1		×	×
10	1	×	1	1

Expressing in the form of standard minterms i.e sum of product expression

F(A, B, C, D) = $C^1D^1 + B^1$

3. ***Simplify $F = \sum(0, 2, 4, 6, 8, 9, 13, 14)$ by using K-map. Implement the given circuit using decoder (use only the block diagram of decoder). Define pairs, Quads and octels in K-map. [2068 Shrawan]***

Ans : Given Function

F(A, B, C, D) = $\sum$ (0, 2, 4, 6, 8, 9, 13, 14)

Using K-map

AB \ CD	00	01	11	10
00	1			1
01	1			1
11		1		1
10	1	1		

Simplifying using SOP expression

F(A, B, C, D) = $A^1D^1 + B^1C^1D^1 + AC^1D + BCD^1$

Now designing the given functional system by using decoder

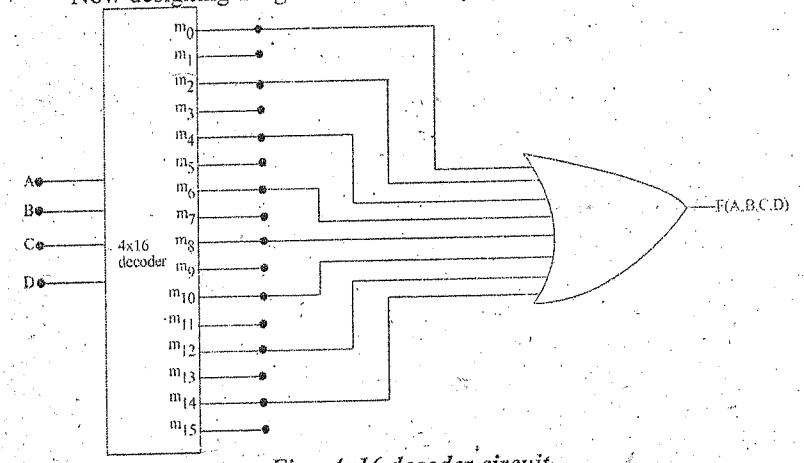


Fig:- 4x16 decoder circuit

For 2nd part of Question:-

Pairs :- A Set of 2 elements in k-map is called pairs.

Quads :- A set of 4 elements in k-map is called quads.

Octels :- A set of 8 elements in k-map is called octels.

4. ***Simplify $\pi(0, 4, 5, 8, 9, 11, 15)$ using K-map and write its standard SOP expression.*** ***[2068 Baishak]***

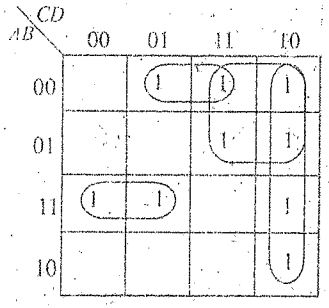
Ans : Given Function

F(A, B, C, D) = π(0, 4, 5, 8, 9, 11, 15)

Now using K-map

AB \ CD	00	01	11	10
00	0			
01	0	0		
11			0	
10	0	0	0	

Now representing the standard SOP expression



∴ SOP expression

$F(A, B, C, D) = ABC^1 + A^1B^1D + A^1C + CD^1$

5. *Obtain the product of sum expression for the function $F = \sum (0, 2, 3, 7)$ and draw a circuit using any universal gate of your choice to realize this function* *[2066 Bhadra]*

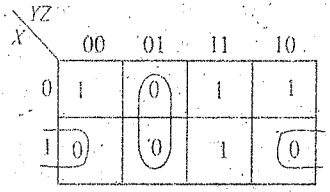
Ans : Let we choose

Input terminals: A, B and C

Output terminal Y

Given function $F = \sum (0, 2, 3, 7)$

Now using K-map



Now expressing the given function on POS expression

$F^1 = y^1z^1 + xz^1$

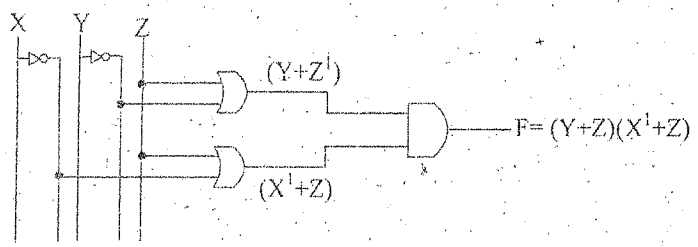
∴ $F = (F^1)^1$

$F = [y^1z + xz^1]$

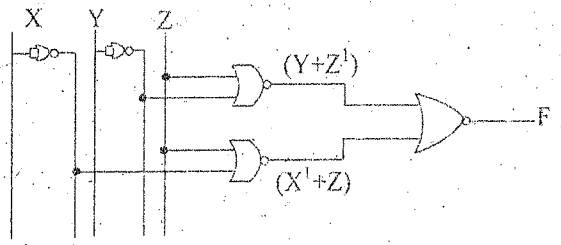
$F = (y^1z)^1 \cdot 1(xz^1)^1$ [∵ using De Morgon's theorom]

$F = (y + z^1) \cdot (x^1 + z)$ [∵ using De Morgon's theorom]

First we design the circuit using basic gates,



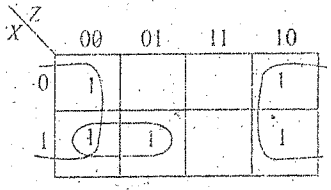
Realizing the given representation by using universal (NOR) gate only.



6. *Obtain the SOP expression for $f(x, y, z) = \sum (0, 2, 4, 5, 6)$* *[2065 Shrawan]*

Ans : Given function $f(x, y, z) = \sum (0, 2, 4, 5, 6)$

Using K-map expression



Now, sum of product (SOP) expression

$F = z^1 + xy^1$

Expressing into tabular form, we get,

x	y	z	Output
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

7. *Simplify the following Boolean expression.* *[2064 Falgun]*

Ans : a) $B + AB^1 + AB^1C + A^1B + AB^1CD^1 + A^1BC^1DE^1$

$= B + A^1B + AB^1 + AB^1C + AB^1CD^1 + A^1BC^1DE^1$

$= B(1 + A^1) + AB^1(1 + C) + AB^1CD^1 + A^1BC^1DE^1$

$= B + AB^1 + AB^1CD^1 + A^1BC^1DE^1$

$= B + AB^1(1 + CD^1) + A^1BC^1DE^1$

$= B + AB^1 + A^1BC^1DE^1$

$= B(1 + A^1C^1DE^1) + AB$

$= B + AB^1$

$= A + B$

b) $[\{(A+B)^1+A\}^1+\{(A+B)^1+B\}^1]^1$

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

$= \{(A+B)^1+A\}.\{(A+B)^1+B\}$

$= (A+B)^1+AB$

$= A^1B^1+AB = A \odot B$

8. ***Prone the following Boolean expression $AB+AB^1C = A^1BC = AB + AC + BC$. And simplify $\sum(1, 5, 7, 9, 10, 11, 14)$ by using K-map and write its standard product of sum (SOP) expression. [2064, Jestha]***

Ans : Here,

$LHS = AB + AB^1C + A^1BC$

$= A(B + B^1C) + A^1BC$

$= A(B + C) + A^1BC$ [$\because A + A^1B = A+B$]

$= AB + AC + A^1BC$

$= AB(A + A^1B)C$

$= AB(A + B)C$

$= AB + AC + BC = RHS$

Given function

$F(A, B, C, D] = \sum(1, 5, 7, 9, 10, 11, 14)$

Using K-map

AB \ CD	00	01	11	10
00		1		
01		1	1	
11				1
10	1	1		1

For POS expression;

AB \ CD	00	01	11	10
00	0		0	0
01	0			0
11	0	0	0	
10				

$\therefore F^1(A, B, C, D) = A^1D^1 + A^1B^1C + ABC^1 + ABD$

Now, $F(A, B, C, D) = [F^1(A, B, C, D)]^1$

$= [A^1D + A^1B^1C + ABC^1 + ABD]^1$

$= (A+D).(A+B+C^1).(A^1+B^1+C).(A^1+B^1+D^1)$

Which is required POS expression

9. ***Prove the following Boolean expression***

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

Simplify $\sum(1, 2, 3, 8, 9, 10, 11, 14)$ and $d(0, 4, 12)$ using K-Map and write its standard POS expression. [2067 Ashad]

Sol^n: $LHS = AB + A\bar{B}C + \bar{A}BC$

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

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

$= AB + AC + \bar{A}BC$

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

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

$= AB + AC + BC = RHS$ proved

For the second part of Question:-

Given $\sum(1, 2, 3, 8, 9, 10, 11, 14)$ and $d(0, 4, 12)$

Representing in K-map,

AB \ CD	00	01	11	10
00	×	1	1	1
01	×			
11	×			1
10	1	1	1	1

$\therefore$ sum of product for $F(A, B, C, D) = A\bar{D} + \bar{B}$

For standard POS

AB \ CD	00	01	11	10
00	×			
01	×	0	0	0
11	×	0	0	
10				

$\therefore F(A, B, C, D) = [F^1(A, B, C, D)]^1$

$= [A^1B + BD]^1$

$= (A^1B).(BD)^1$ [using De-Morgan's Theoram]

$= (A^1+B^1).(B^1 + D^1)$

CHAPTER 4

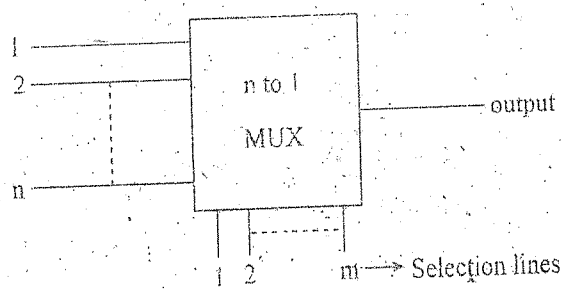
Data Processing Circuits

4.1 Multiplexer:-

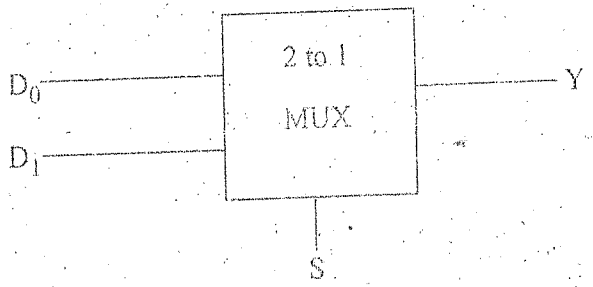
The term multiplex means many to one. Hence multiplexing is the process of transmitting a large number of information over a single line.

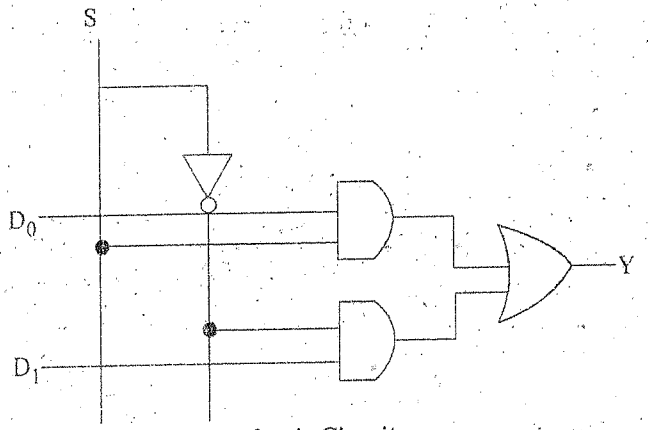
Thus a multiplexer is a logic circuit that accepts data from several data input lines and allows one of them at a time to get through the output. The selection of particular line is controlled by a set of selection lines. In general there are 2^n input lines and n selection line whose bit combination determine the input to be selected.

A multiplexer is also called data selector since it selects one of many inputs and drives the information to output.



2×1 MUX : A 2×1 or 2:1 MUX has two inputs only one output and one selection lines. The block diagram for 2×1 MUX is as shown,

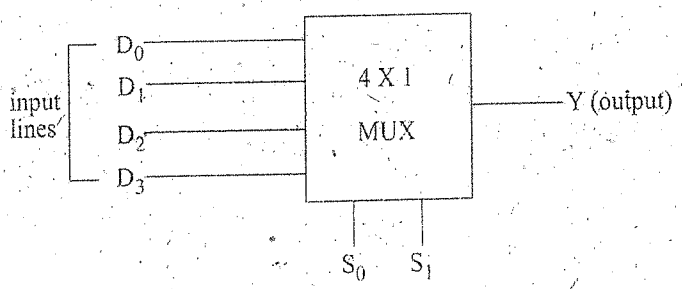




Logic Circuit

When S = 0, Y = D_0
When S = 1, Y = D_1

4×1 MUX : A 4×1 MUX has four data input lines ($D_0 - D_3$), Single output line (Y) and two selection lines (S_0 and S_1). The block diagram of 4×1 MUX is as shown,



Truth table

Input		Output
S_0	S_1	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

The data output is equal to D_0 only if $S_1 = 0, S_0 = 0 : Y = D_0 \overline{S_0}\ \overline{S_1}$

The data output is equal to D_1 only if $S_1 = 0, S_0 = : Y = D_1 S_0 \overline{S_1}$

The data output is equal to D_2 only if $S_1 = , S_0 = 0 : Y = D_2 \overline{S_0} S_1$

The data output is equal to D_3 only if $S_1 = 0, S_0 = 1 : Y = D_3 S_0 S_1$

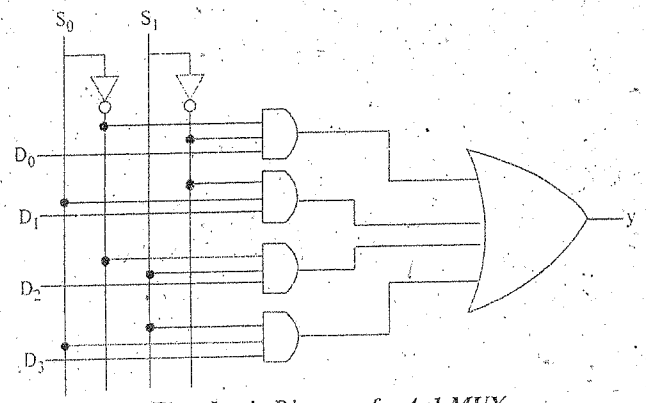


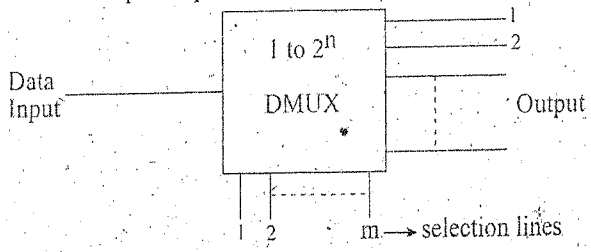
Fig:- Logic Diagram for 4x1 MUX

4.2 Demultiplexers:-

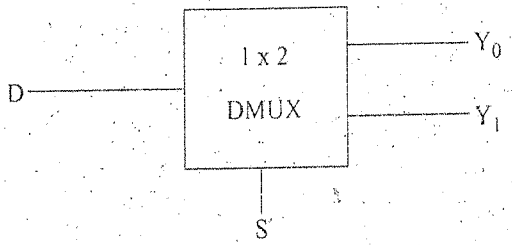
The term "Demultiplex" means one to many. Hence demultiplexing is the process of taking information from one input and transmitting the same over one of several outputs.

Thus demultiplexer is a logic circuit that receives information on a single input and transmits the same information over one of the 2^n possible output lines. The selection of specific output line is controlled by the bit values of n-selection line.

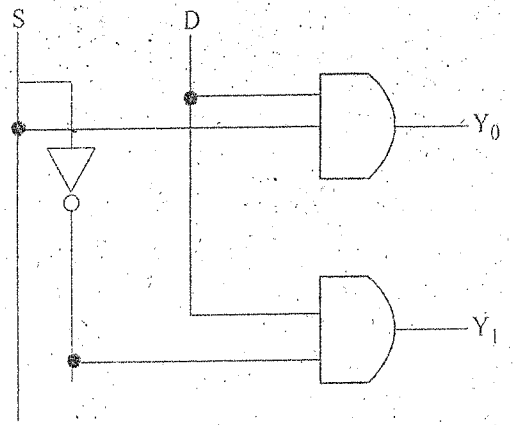
Hence demultiplexer performs the reverse operation of multiplexer



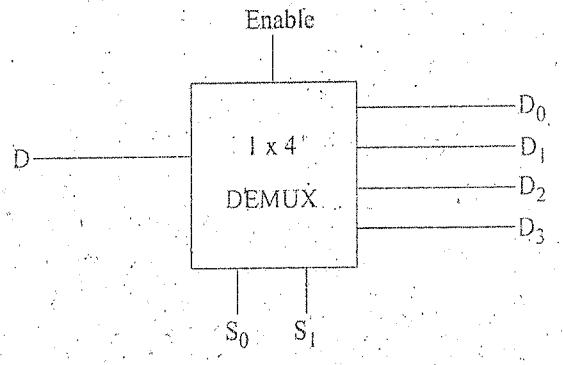
1×2 DMUX : A 1:2 DMUX has one input and two output signals. The block diagram for 1×2 DMUX is,



When S = 0, Y_0 = D
When S = 1, Y_1 = D



1×4 DMUX : A 1×4 DE MUX has one input line and 4 output lines with two selection lines. The block diagram for 1×4 DE MUX is as shown,



Truth table

Data selection			Outputs			
D	S_1	S_0	D_3	D_2	D_1	D_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

When S_0 = 0 & S_1 = 0 The data input is connected to D_0
When S_0 = 0 & S_0 = 1 The data input is connected to D_1
When S_0 = 1 & S_0 = 0 The data input is connected to D_2
When S_0 = 1 & S_1 = 1 The data input is connected to D_3

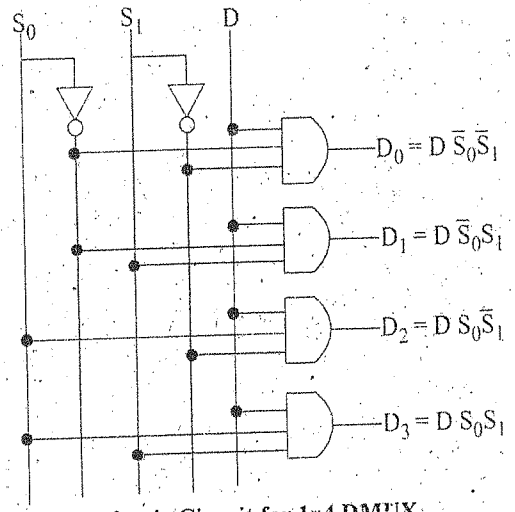
The expression for output can be written as,

$D_0 = D\,\overline{S_0}\,\overline{S_1}$

$D_1 = D\,\overline{S_0}\,S_1$

$D_2 = D\,S_0\,\overline{S_1}$

$D_3 = D\,S_0\,S_1$

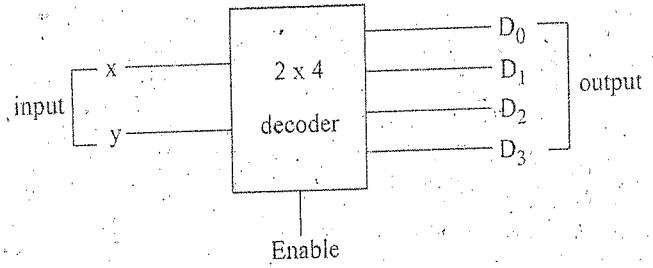


Logic Circuit for 1x4 DMUX

4.3 Decoder:-

A decoder is a combinational circuit that converts the binar information from n input lines to a maximum of 2^n unique output lines. Th basic function of decoder is to detect the presence of a specified combinatio of bits on its input and to indicate the presence of that code by a specifie output level. A decoder is similar to de-max but without any data input

2×4 Decoder or 2 to 4 line Decoder : A 2×4 decoder has two inpu and 4 outputs lines. Figure below shown 2×4 decoder



Block Diagram of 2x4 decoder

Truth table

Data selection		Outputs			
X	Y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

The logic function can be written as,

$D_0 = \overline{X}\,\overline{Y}$

$D_1 = \overline{X}\,Y$

$D_2 = X\,\overline{Y}$

$D_3 = XY$

The logic Circuit will be as shown,

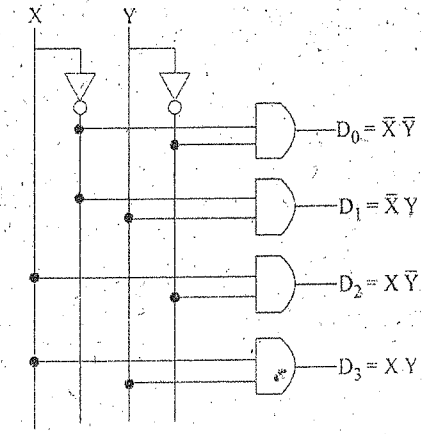
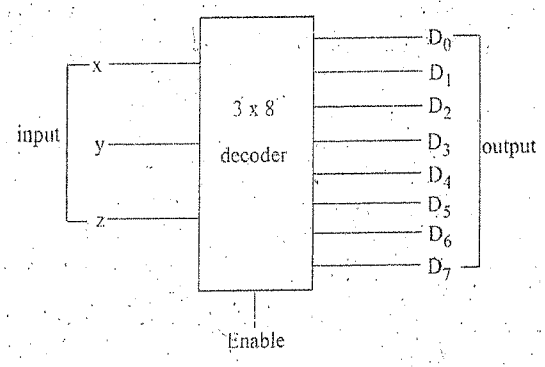


Fig: Logic Diagram 0f 2x4 decoder

3×8 Decoder: A 3×8 decoder has three inputs and 8 outputs. The block diagram, truth table and logic diagram of 3×8 decoder is shown.



Truth table:-

Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

The logic function can be written as.

$D_0 = \overline{X}\ \overline{Y}\ \overline{Z}$ $\quad D_4 = X\overline{Y}\ \overline{Z}$

$D_1 = \overline{X}\ \overline{Y}\ Z$ $\quad D_5 = X\overline{Y}\ Z$

$D_2 = \overline{X}\ Y\overline{Z}$ $\quad D_6 = X\,Y\,\overline{Z}$

$D_3 = X\,Y\,Z$ $\quad D_7 = X\,Y\,Z$

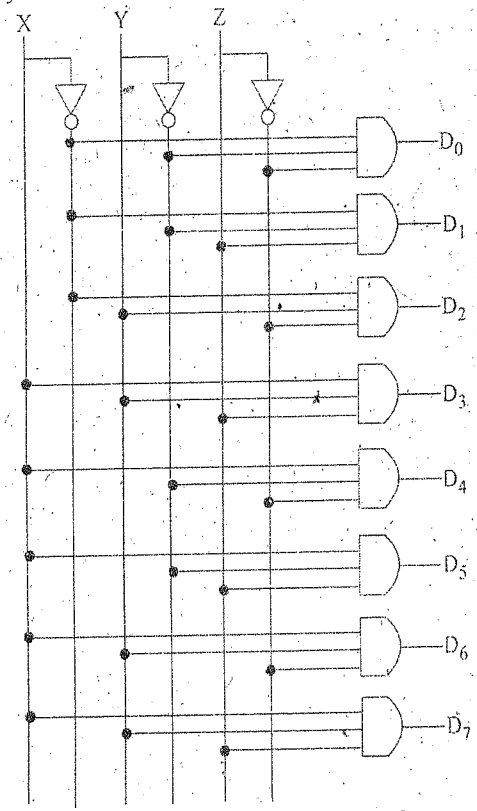


Fig: Logic Diagram of 3x8 decoder

4.4 BCD to decimal decoders:- [2069 Ashad]

The BCD to decimal decoder converts each BCD code into one of ten possible decimal digits indications. It is referred as 4 line to 10 the or 1 of 10 decoders.

Truth table

Inputs				Outputs									
A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

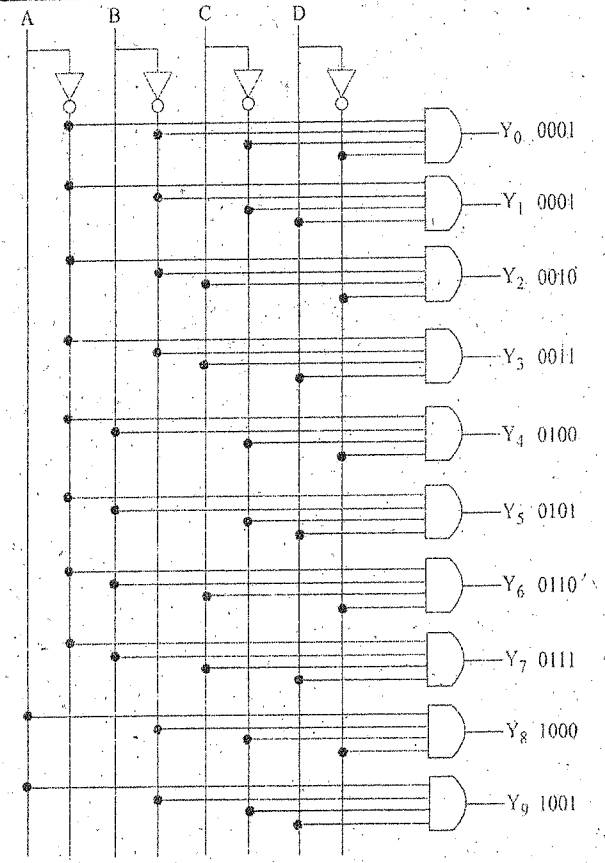


Fig:- 1 of 10 decoder

4.5 Encoder:-

An encoder is a logic circuit that performs the inverse operation of a decoder. An encoder have 2^n input lines and n output lines. The output line generates the binary code corresponding to the input value.

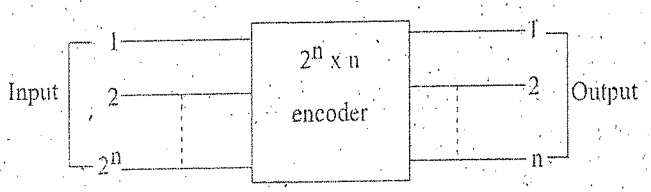


Fig: Block diagram of an encoder

Octal to Binary Encoder (8 to 3 line encoder) :-

A 8 to 3 line encoder has eight inputs and 3 outputs. The encoder can be implemented with OR gate where inputs are determined directly from the truth table. Output Z is equal to 1 when the input octal digit is 1 or 3 or 5 or 7, output Y is equal to 1 when the input octal digits is 2 or 3, 6, 7 and output X is 1 for digits 4 or 5 or 6 or 7.

These conditions can be expressed by the following Boolean function,

$$X = D_4 + D_5 + D_6 + D_7$$
$$Y = D_2 + D_3 + D_6 + D_7$$
$$X = D_1 + D_3 + D_5 + D_7$$

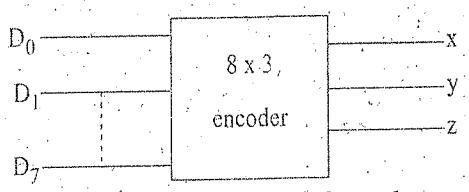


Fig: Block diagram of a 8x3 encoder

Truth table

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

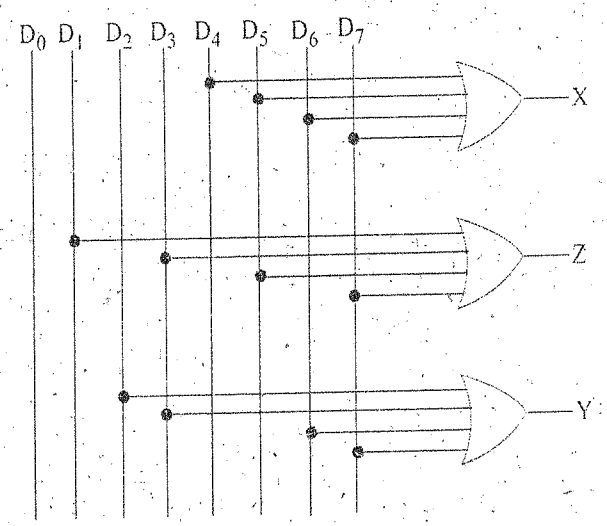


Fig:- Logic diagram for 8 to 3 line encoder

4.6 Exclusive OR gates:-

This section has already been discussed is chapter-2.

4.7 Parity Generates and Checkers:-

Even parity means an n-bit input has an even number of 1s. for instance 110011 has even parity because it contains four 1s.

Odd parity means an n-bit input has an odd number of 1s. For instance 110001 has odd parity because it contains three 1s.

Here are two more example,

1111 0000 1111 0011 → even parity
1111 0000 1111 0111 → odd parity

Parity Checker :- Exclusive OR gate are ideal for checking the parity of a binary number because they produce an output 1 when input has an odd number of 1s. Therefore, an even-parity input to an exclusive OR gate produces a low output, while an odd-parity input produces high output.

Figure below shows 16- input exclusive OR gate. A 16 bit number drives the input. The exclusive OR gate produces an output 1 because the input has odd parity. If the 16 bit input changes to another value, the output becomes 0 for even parity number and 1 for odd-parity numbers.

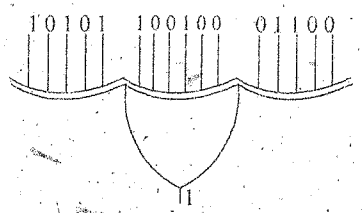


Fig: EX-OR gate with 16 input

Parity Generator :- Consider a circuit as shown with 8-bit binary numbers input $X_7\ X_6\ X_5\ X_4 \qquad X_3\ X_2\ X_1\ X_0$

Suppose this number equals 0100 0001. Then the number has even parity, which means the exclusive OR gate produces an output of 0. Because of the inverter

$X_8 = 1$

And the final 9-bit output is 0 0110 0001. It has odd parity.

Suppose this number is changed to 0110 0001. Now it has odd parity, which means the exclusive OR gate produces an output of 1. But the inverter produces 0, so that final 9-bit output is 0 0110 0001. It has odd parity again.

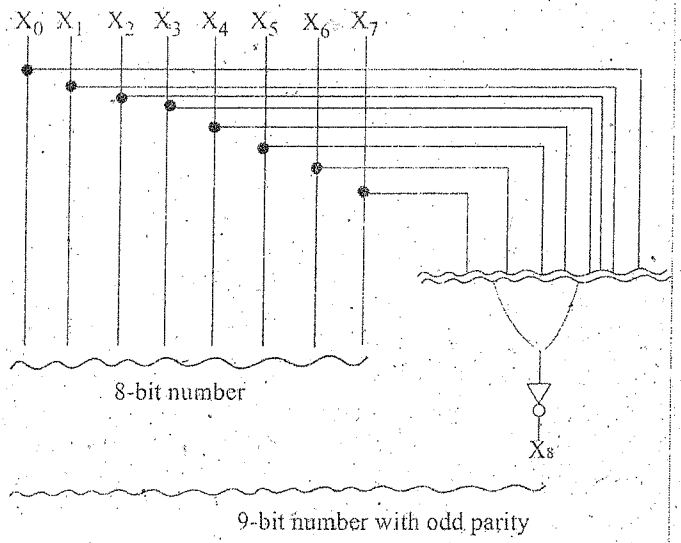


Fig:-Odd-parity generator

Note:- The circuit shown above is odd-parity generator because it always produces 9-bit output number with odd-parity.

Application:- Because of transients, noise and other disturbances, 1-bit error sometimes occur when binary data is transmitted over telephone lines or other communication paths. One way to check for errors is to use an odd-parity generator at the transmitting end and an odd-parity checker at the receiving end. If no 1-bit error occurs in transmission the received data will have odd parity. But if one of the transmitted bits is changed by noise or any other disturbance, the received data will have even parity.

4.8 Magnitude Comparator:-

A magnitude comparator is a circuit that compares two n-bit binary numbers; say X and Y and actives one of these three output X = Y, X >Y and X<Y.

Single bit magnitude comparator

Truth table

a	b	a = b	a>b	ac
0	0	0	0	0
0	1	0	0	1
1	0	1	1	0
1	1	0	0	0

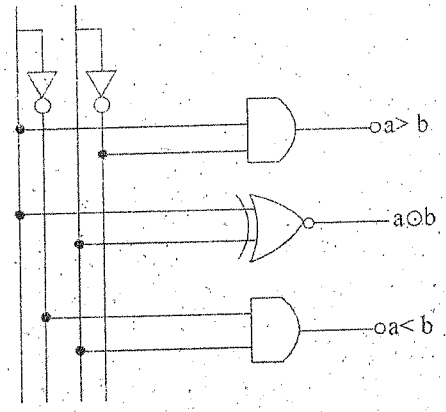


Fig:-1-bit magnitude comparator

For a = b

$Y(a=b) = \bar{a}\,\bar{b} + ab = a \odot b$

For a>b

$Y(a>b) = a\,\bar{b}$

For a<b

$Y(a<b) = \bar{a}\,b$

4.9 Read Only Memory:-

A Read Only Memory (ROM) is an IC that can store thousands of binary numbers representing computer instructions and other fixed data. Some of the smaller ROMs are used to implement truth tables.

A ROM is a device that include both the decoder and the OR gates within a single IC package. The connections between outputs of decoder and inputs of OR gates can be specified for each particular configuration by programming the ROM. The ROM is often used to implement a complex combinational circuit in one IC package and two eliminate all interconnecting wires.

Application:-

1) ROMs are used to implement complex combinational circuits directly from their truth tables.
2) They are used for converting from one binary code to another.
3) They are use in the design of control units of digital systems.

Types of ROM

1) PROM
2) EPROM
3) EEPROM

PROMs are user – Programme able and ideal for small production runs.

EPROMs are not only user-programmable, but they are also erasable and reprogrammable during the design and development cycle.

4.10 Programmable Array Logic:-

Programmable Array Logic (PAL) is a programmable array of logic gates on a single chip. PALs are another design solution; similar to SOP solution, POS solution and multiplexer logic.

PALs has programmable AND array and a fixed OR array. The PAL has the advantage of having up to 16 inputs in commercially available devices.

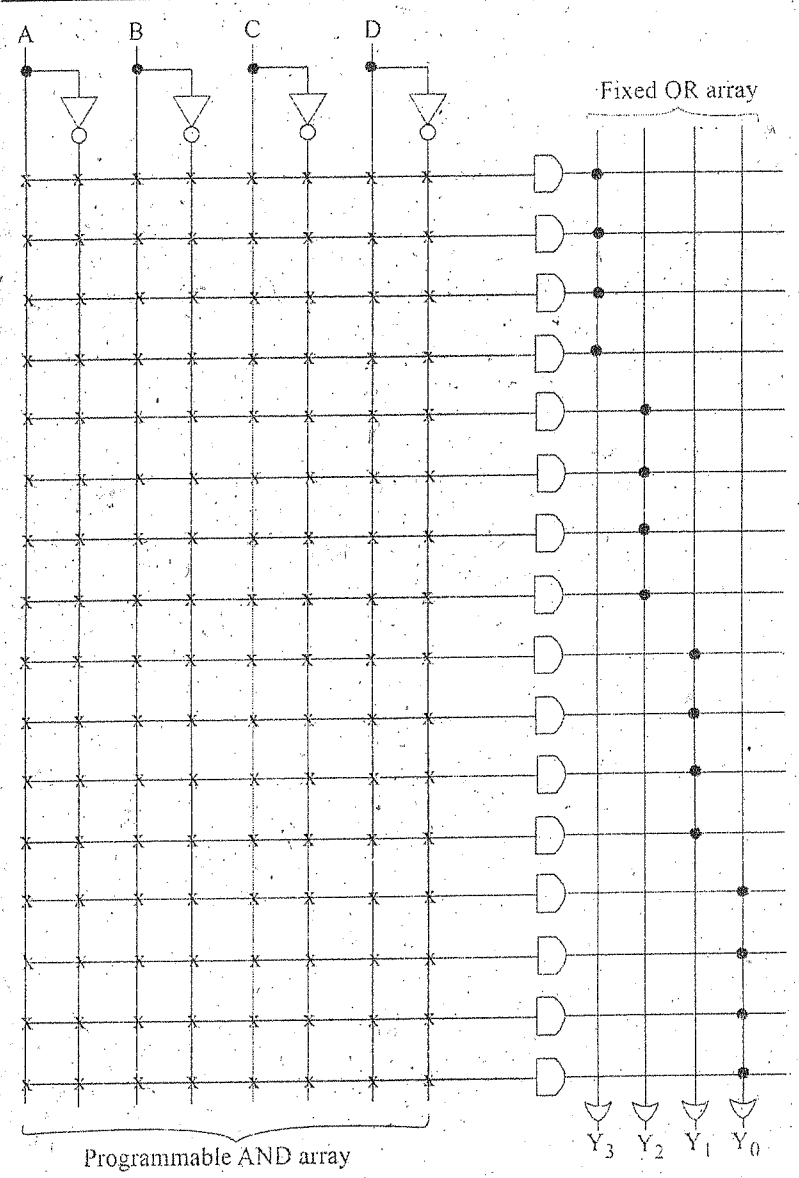


Fig:-Structure of PAL

4.11 Programmable Logic Array:-

The Programmable Logic Array is much more versatile device than PAL since both its AND its AND-gate array and its OR-gate array are fusible- linked and programmable.

A PLA having 3 input variables (A B C) and 3 output variables (XYZ) is as shown in figure. Eight AND gates are required to decode the 8 possible input states. In this case, there are three OR gates that can be used to generate logic functions at the output.

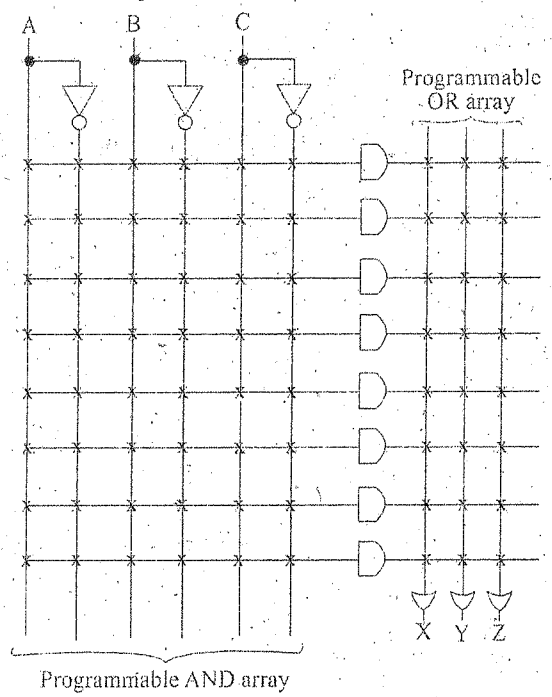


Fig:- Structure of PAL

4.12 Trouble shooting with logic probe:-

Figure below shows a logic probe which is a trouble shooting tool helpful in diagnosing faulty circuits. When we touch the probe tip to the output node as shown, the device lights up for a high state and goes dark for a low state. For instance if either A or B or both are low, then Y is high and the probe lights up. On the other hand if A and B are both high, Y is low and probe is dark.

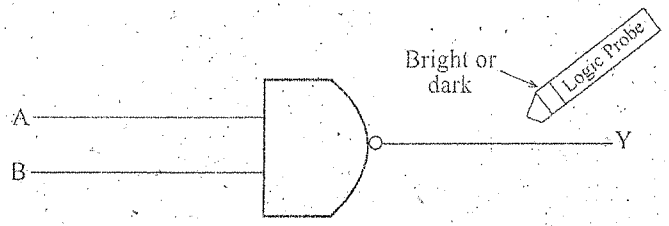


Fig:-Using a logic Probe

Old Question/Solution

1. ***Design a 32 to 1 Multiplexer using 16 to 1 and 2 to 1 multiplexers.*** *[2068 Chaitra]*

Ans:

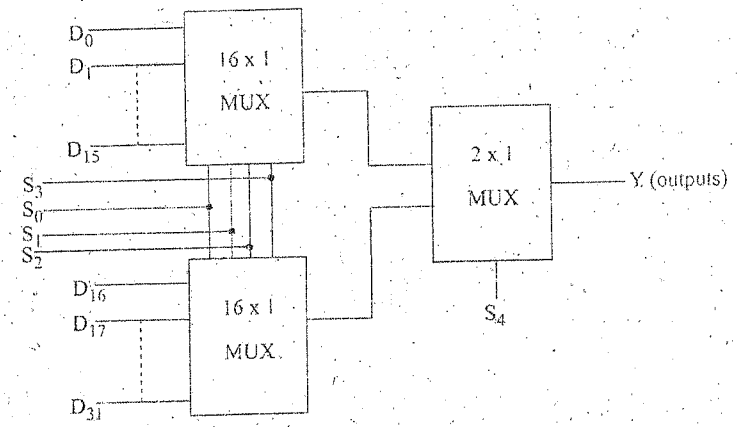


Fig:- 32x1 MUX using 16x1 & 2x1 MUX

2. ***Design a 3-bit even parity generator & parity checker.*** *[2068 Shrawan]*

Ans: **Parity Generator:-** To design 3-bit even parity generator lets take a 3 bit binary number as X_2 X_1 $X_0$7

Let this numbers equals 100.

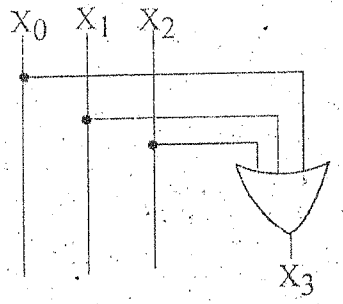
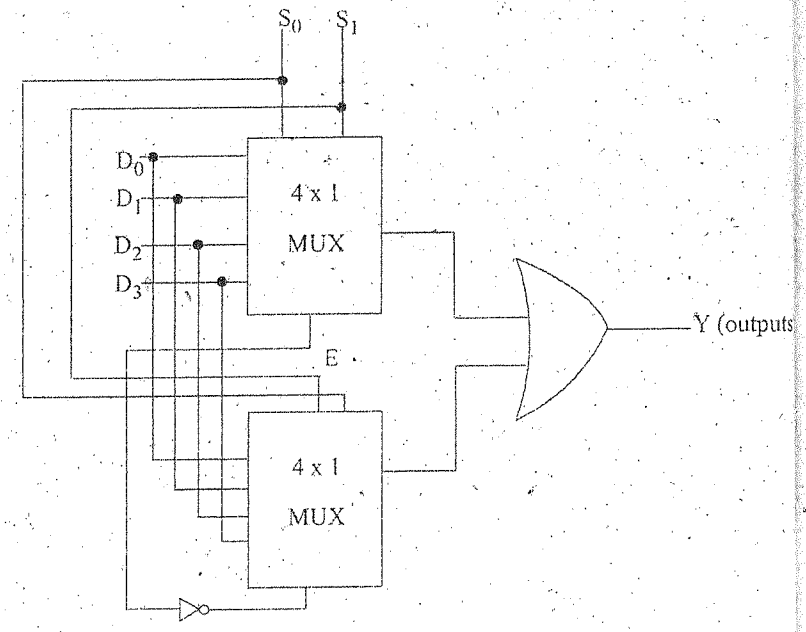


Fig:- 3-bit even parity generator

As the number has odd parity, the exclusive OR-gate produces an output of 1. The final 4-bit output is 1001. It has even parity.

Parity Checker:- See Theory Part.

3. *Design 8 × 1 MUX using two 4 × 1 MUX.* *[2066 Bhadra*



4. *Determine the parity bit of 100101 following odd parity and writ down 1 byte long representation for this number including the pari bits. Why are parity bits important in digital system.*

[2065 Shrawan]

Ans: Given bit is : 100101

Since it has odd number of 1s, the given bit has odd parity.

∴ odd parity bit : 0

So, 8 bit Representation : 01001010

For Next part refer theory portion

5. *Write down the truth table for an odd parity generator for any 2-bi word. Design the circuit using any universal gate of your choice*

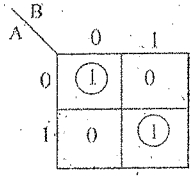
[2065 Shrawan]

Ans: Here,

Truth table for odd parity

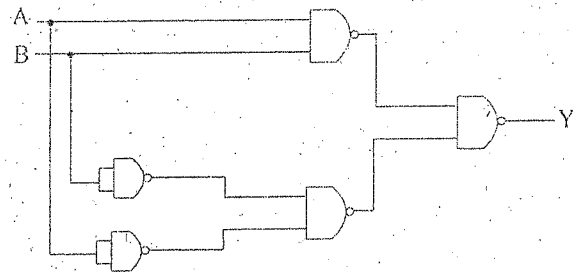
Message		Odd Parity
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

, Representing the output using K-map,



$\therefore y = \overline{A}\,\overline{B} + AB = A \odot B$

Implementing the output using NAND gate as shown,



6. *What is priority encoder ? Design octal to binary priority encoder.*

Ans:- A Priority encoder is an encoder that includes the priority function. The operation of priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. In priority encoder when multiple inputs are high, priority is given to that input having higher subscript. For instance in 4×2 encoder, if all the inputs are high, the input D_3 will have higher priority & output will be 11.

Octal to binary priority encoder:-

Inputs								Output		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	×	0	0	1
0	0	0	0	0	1	×	×	0	1	0
0	0	0	0	1	×	×	×	0	1	1
0	0	0	1	×	×	×	×	1	0	0
0	0	1	×	×	×	×	×	1	0	1
0	1	×	×	×	×	×	×	1	1	0
1	×	×	×	×	×	×	×	1	1	1

7. ***Design 2 bit magnitude comparator.***

Ans:- Consider a 2-bit number as $A = A_1A_0$

$B = B_1B_0$

For A = B,

$A_1 = B_1$, $A_0 = B_0$

$\Rightarrow \quad (A_1 \odot B_1)\,(A_0 \odot B_0)$

For A = B,

Case a: $A_1 > B_1$ [assume A_1=1 & B_1=0]

$\rightarrow (A_1B_1^{1})$

Case b: $A_1 = B_1$ but $A_0 > B_0$ [assume A_0=1 & B_1=0]

$\rightarrow (A_1 \odot B_1)\,(A_0 \bar{B}_0)$

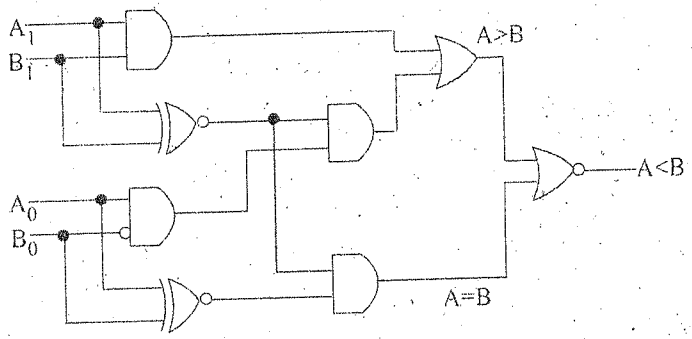


Fig:- 2 bit magnitude comparator

8. ***Design a combinational logic circuit that performs multiplication between two 4-bit numbers using binary parallel adder and other gates.***

Ans:- Consider two 4-bit numbers $A_0\,A_1\,A_2\,A_3$ & $B_0\,B_1\,B_2\,B_3$

			B_3	B_2	B_1	B_0
			$\times A_3$	A_2	A_1	A_0
			$A_0 B_3$	$A_0 B_2$	$A_0 B_1$	$A_0 B_0$
		$A_1 B_3$	$A_1 B_2$	$A_1 B_1$	$A_1 B_0$	$\times$
	$A_2 B_3$	$A_2 B_2$	$A_2 B_1$	$A_2 B_0$	$\times$	$\times$
$+ A_3 B_3$	$A_3 B_2$	$A_3 B_1$	$A_3 B_0$	$\times$	$\times$	$\times$
C_6	C_5	C_4	C_3	C_2	C_1	C_0

The logic circuit will be as shown,

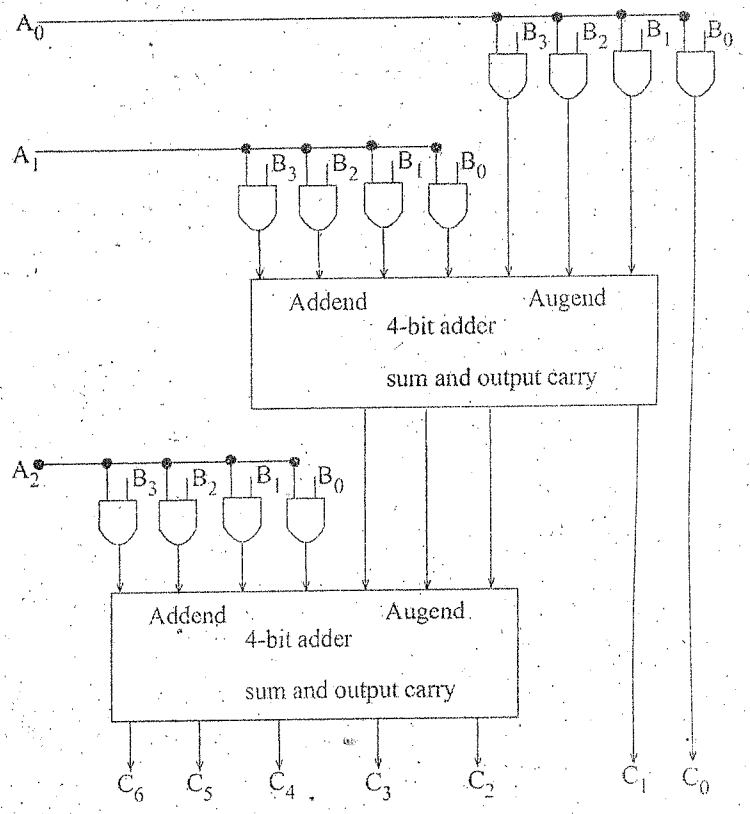


Fig:- 4-bit binary multiplier

CHAPTER 5

Arithmetic Circuits

Introduction :-

The logic circuit constructed by combining logic gates and flip-flops to perform the arithmetic operation such as binary addition, subtraction multiplication and division are called arithmetic circuit. There circuits are used in the design of arithmetic logic unit (ALU).

5.1 Binary Addition:-

Computer circuits don't process decimal number; they process binary numbers. Binary addition is the key to binary subtraction, multiplication and division. So, let's begin with four must basic cases of binary addition;

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Some example

Add the following 4-bit, 8-bit and 16-bit binary numbers.

1111	0101	0111	0000	1111	1010	1100
+ 1101	+ 0011	0101	+ 0011	1000	0111	1111
$(11100)_2$	$(1000\ 1100)_2$		0100	1000	0010	1011

5.2 Binary Subtraction:-

Let's begin with four basic cases of binary subtraction,

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 1 = 0$$
$$10 - 1 = 1$$

For larger binary numbers, subtract column by column as done with decimal numbers. This means that we sometimes have to borrow from the next higher column. Here is an example:

$$\begin{array}{r} 1101 \\ -\ 1010 \\ \hline ? \end{array}$$

Subtract the LSB to get

$$\begin{array}{r} 1101 \\ -\ 1010 \\ \hline 1 \end{array}$$

To subtract the bits of second column, borrow from the next-higher column obtain

$$\begin{array}{rr} \text{Burrow} \rightarrow & 1 \\ & 1001 \\ & 1010 \\ \hline & 1 \end{array}$$

In the second column from the right, subtract as follows : 10-1=1, to get

$$\begin{array}{rr} \text{Burrow} \rightarrow & 1 \\ & 1001 \\ - & 1010 \\ \hline & 11 \end{array}$$

Now subtract remaining column:

$$\begin{array}{rr} \text{Burrow} \rightarrow & 1 \\ & 1001 \\ - & 1010 \\ \hline & 0011 \end{array}$$

5.3 Unsigned Binary Numbers:-

Data of the foregoing type is called unsigned binary because all of the bits in a binary numbers are used to represent the magnitude of corresponding decimal numbers. Hence unsigned binary numbers are those numbers which represents only magnitude i.e. no sign (+ or -) is associated with them.

The unsigned 8-bit numbers are from 0000 0000 to 1111 1111, equivalent to decimal 0 to 255 and hexadecimal (00 H to FFH).

The unsigned 16-bit numbers are from decimal 0 to 65, 535 and hexadecimal 0000H to FFFF H

Overflow occurs when a sum exceeds the range of number system. For eg: In 8-bit arithmetic, addition of two unsigned numbers where sum is greater than 255 causes an overflow.

For eg

1111	0101
+ 1101	+ 1010
11100	1111

The above condition is called overflow.

5.4 Signed Magnitude Numbers:-

The foregoing number contains a sign bit followed by magnitude bits. Numbers in this form are called sign magnitude numbers.

Sign magnitude numbers use the MSB as a sign bit, with O for '+' sign and 1 for '-' sign. The rest of the bits are for the magnitude of the numbers. For this reason, 8-bit numbers cover the decimal range of -127 to + 127 while 16-bit numbers cover –32767 to +32767.

Here are some example of converting sign-magnitude numbers.

+ 7	→	0000	0111		
- 16	→	1001	0000		
+ 25	→	0000	0000	0001	1001
+ 128	→	1000	0000	1000	0000

In simple understanding – 01001 is encoded as 101001 and 1 stands for –ve.

Note:- Sign-Magnitude Numbers are often used in analog to digital conversion (ADC). Their main advantage is their simplicity. However, sign-magnitude numbers have limited use because they require complicated arithmetic circuits.

5.5 2's complement Representation:-

2's complement representation is the most widespread code for positive and negative number. Positive numbers are coded as sign-magnitude numbers and negative numbers are coded as 2's complements. The key feature of this number system is that taking the 2's complement of a number is equivalent to changing its sign. This characteristic allows us to subtract numbers by adding the 2's complement of the subtrahend. The advantage is simpler arithmetic hardware.

1's Complement :- The 1's complement of a binary number is the number that results when we complement each bit. Fig (1) shows how to produce the 1's complement with logic circuits.

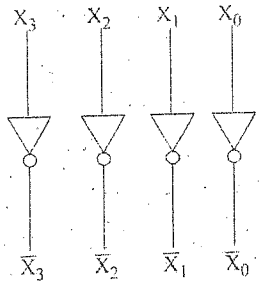


Fig:- Inverters produces the 1's complement

For instance if input is, $X_3\ X_2\ X_1\ X_0 = 1000$ then is complement is $\overline{X}_3\ \overline{X}_2\ \overline{X}_1\ \overline{X}_0 = 0111$

2's Complement :- The 2's complement is the binary number that results when we add 1 to the 1's complement

i.e, 2's complement = 1's complement +1

For instance, to find 2's complement of 1011, we proceed like this,

1011→ 0100 (1's complement)

0100 + 1 = 0101 (2's complement)

5.6 2's Complement Arithmetic:-

2.1 Subtract $(1000011)_2 - (1010101)_2$ using 1's complement

Here,

1's complement of $(1010101)_2$

$= 0101010$

Now,

$$\begin{array}{r} 1010100 \\ +\ 0111101 \\ \hline (10010001)_2 \end{array}$$

Since there is end carry, the answer will be 1's complement of $(110110)_2$ i.e. $(-0010010)_2$

2.2 Subtract $(1010100)_2 - (1000010)_2$ using 1's complement

Here,

1's complement of (1000010)

$= (0111101)_2$

Now,

$$\begin{array}{r} 1010100 \\ +\ 0111101 \\ \hline 10010001 \end{array}$$

Since there is end carry we discard it and add 1 on above result i.e,

$$\begin{array}{r} 0010001 \\ +\ 1 \\ \hline (0010010)_2 \end{array}$$

2.3 Use 2's complement method perform $(1000011)_2 - (1010101)_2$.

Here, first we find is complement of (1010101) and add 1 on the result that comes, i.e.

1's complement of 1010101

$= 0101010$

Now,

$$\begin{array}{r} 1010100 \\ +\quad 1 \\ \hline 0101011 \end{array}$$

So,

$$\begin{array}{r} 1000011 \\ +\ 0101011 \\ \hline 1101110 \end{array}$$

Since there is no end carry, the answer will be 2's complement of (1101110) i.e,

1's complement of 1101110

$= 0010001$

&

$$\begin{array}{r} 0010001 \\ +\quad 1 \\ \hline (-0010010)_2 \end{array}$$

2.4 Subtract $(1010100)_2 - (1000010)_2$ using 2's complement

Here, 1's complement of 1000010

$= 0111101$

Now,

$$\begin{array}{r} 0111101 \\ +\quad 1 \\ \hline 0111110 \end{array}$$

Next,

$$\begin{array}{r} 1010100 \\ +\ 0111110 \\ \hline 10010010 \end{array}$$

Since there is end carry we discard it. Hence the answer will be $(10010010)_2$

2.5 Subtract $(16)_{10}$ from $(26)_{10}$ using 2's complement binary method. **[2064 Falgun]**

First we convert $(16)_{10}$ and $(26)_{10}$ into binary form as shown,

2	26	$\rightarrow 0$
2	13	$\rightarrow 1$
2	6	$\rightarrow 0$
2	3	$\rightarrow 1$
	1	$\rightarrow 1$

&

2	16	$\rightarrow 0$
2	8	$\rightarrow 0$
2	4	$\rightarrow 0$
2	2	$\rightarrow 0$
	1	$\rightarrow 1$

$\therefore (26)_{10} = (11010)_2$

& $(16)_{10} = (10000)_2$

Now, we find 2's complement of $(16)_{10}$ i.e $(10000)_2$.

For this first finding 1's complement of $(10000)_2$

$= (01111)_2$

Next,

$$\begin{array}{r} 11111 \\ +\ 1 \\ \hline 10000 \end{array}$$

Finally,

$$\begin{array}{r} 11010 \\ +10000 \\ \hline 101010 \end{array}$$

Since there is end carry 1 we discard it. Hence the answer will be, $(01010)_2$

$= (10)_{10}$

$\therefore (26)_{10} - (16)_{10} = (10)_{10}$

2.6 Convert decimal number 73 to binary, 8 hexadecimal and use 2's complement method to subtract 5-16. **[2064 Jestha]**

Solution

Convert 73 to binary

2	73	$\rightarrow 1$
2	36	$\rightarrow 0$
2	18	$\rightarrow 0$
2	9	$\rightarrow 1$
2	4	$\rightarrow 0$
2	2	$\rightarrow 0$
	1	$\rightarrow 1$

$\therefore (73)_{10} = (1001001)_2$

Converting $(73)_{10}$ to hexadecimal,

Here,

$(73)_{10} = (1001001)_2$

Now making group of four bits as shown,

(0100) (1001)

$\therefore (73)_{10} = (49)_{16}$

For the second Part :-

First we convert 5 and 16 into binary as shown,

2	5	→ 1	2	16	→ 0
2	2	→ 0	2	8	→ 0
2	1	→ 1	2	4	→ 0
			2	2	→ 0
				1	→ 1

$\therefore (5)_{10} = (101)_2$ & $(16)_{10} = (10000)_2$

Next,

1's of $(16)_{10} = (01111)_2$

2's complement $(16)_{10} = 01111$

$+ 1$

$(10000)_2$

Finally,

$$\begin{array}{r} 10000 \\ +\ 00101 \\ \hline (10101)_2 \end{array}$$

Since there is end carry, answer will be 2's complement of, $(10101)_2$

For this,

1's complement of 10101

$= 01010$

So,

2's complement of 10101 = 01010

$+ 1$

$(01011)_2$

$\therefore (5)_{10} - (16)_{10} = (-01011)_2$

5.7 Arithmetic Building Blocks:-

The basic circuits used as arithmetic building blocks are half –adder, full-adder and the controller Inverter.

Half Adder :- A logic circuit used for the addition of two 1-bit numbers is referred to as an half adder.

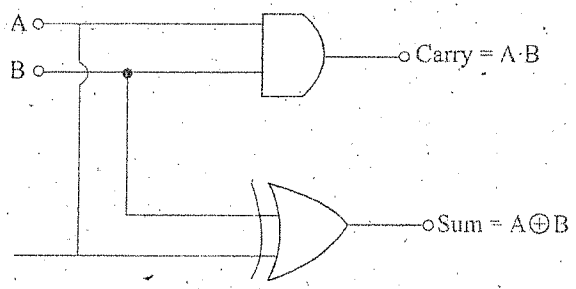


Fig:- Logic Circuit

Fig:- Logic Symbol

Figure above shows the circuit of half adder. It consists of an Ex-OR gate and an OR gate. The output of Ex-OR gate is called SUM while the output if AND gate is called CARRY. Hence half adder has two outputs.

As the AND gate produces high output only when both inputs are high and Ex-OR gate produces high output if either input (not both) is high, the truth table of Half Adder is developed by writing the truth table output of AND gate is CARRY column and the output truth table of Ex-OR gate is SUM column.

Truth Table

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table for a half adder, the logic equations for CARRY and SUM can be written as,

CARRY, $C = A \cdot B$

SUM, $S = A\overline{B} + \overline{A}B = A \oplus B$

This circuit is called half adder because it cannot accept a CARRY – IN from previous additions. This is the reason that half-adder circuit can be used for binary addition of lower must bit only.

Full adder :- A logic circuit used for the addition of three bits – two bits to be added and a carry bit from lower bit order is called full adder. [2068 chaitra]

It also has two outputs SUM and CARRY. A simple circuit for full adder is as shown below although other designs are also possible. It uses 3AND gates, two Ex-OR gates and one OR gate. The final CARRY is given by OR gate while the final sum is given by Ex-OR gate.

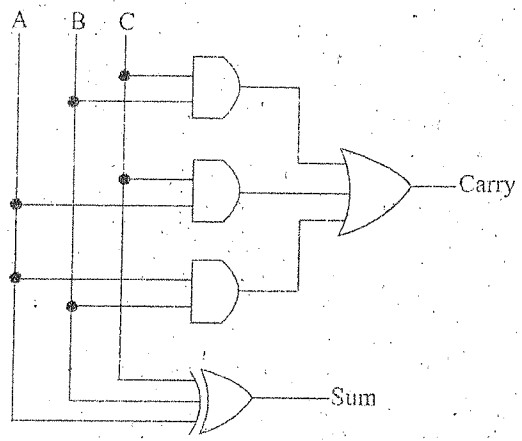


Fig:- Full adder

Truth Table

Input			Output	
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

From this truth table we get K-Map as shown,

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

CARRY = AB+BC+AC

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

SUM = A⊕B⊕C

Controlled inverter:- Figure below shown a circuit for controlled inverter. When INVERT in low, it transmits the 8-bit input to the output. When INVERT is high, it transmit the 1's complement. For instance, if the input number is,

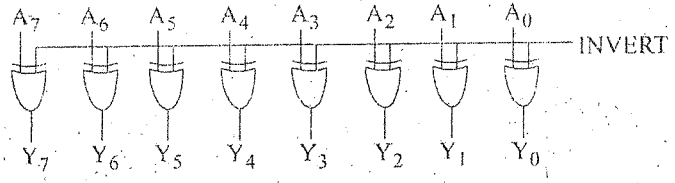
$A_7\ A_6\ A_5 \ldots\ldots\ldots\ldots A_0 = 0110\ 1110$

a low INVERT produces,

$Y_7\ Y_6\ Y_5 \ldots\ldots\ldots\ldots Y_0 = 0110\ 1110$

a high INVERT produces,

$Y_7\ Y_6\ Y_5 \ldots\ldots\ldots\ldots Y_0 = 1001\ 0001$

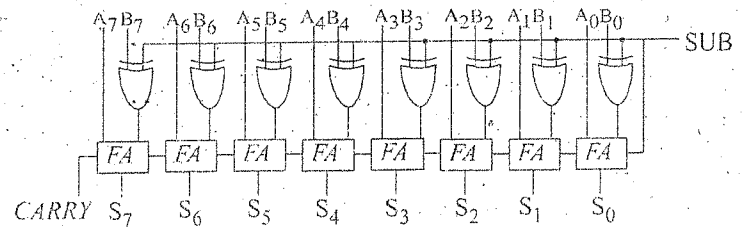


5.8 The Adder – Subtractor :- [2066 Bhadra]

A logic circuit used for both addition and subtraction of binary numbers is called The Adder-Subtractor. This is done barically with the help of full adder.

The full adder shown below can be used to add or subtract binary numbers. The circuit is laid out from right to left, similar to the way we add binary numbers. Hence the least significant column is on right and most significant column is on left. The boxes labeled FA are full adders.

The CARRY OUT from each full adder is the CARRY IN to the next higher full adder. The numbers being processed are $A_7 \ldots\ldots A_0$ and $B_7 \ldots\ldots B_0$ and the output is $S_7 \ldots S_0$.



Addition: The addition operation appears as shown,

$$\begin{array}{rr} A_7\ A_6\ A_5\ A_4 & A_3\ A_2\ A_1\ A_0 \\ +\ B_7\ B_6\ B_5\ B_4 & B_3\ B_2\ B_1\ B_0 \\ \hline S_7\ S_6\ S_5\ S_4 & S_3\ S_2\ S_1\ S_0 \end{array}$$

During addition operation, the SUB signal is kept in low state. Therefore the binary numbers $B_7 \ldots\ldots B_0$ passes through the controlled inverter without no change. The full adders than produce the correct output sum. They do this by adding the bits in each columns, passing carries to the next higher column and so on.

Subtraction: The subtraction operation appears as shown,

$$\begin{array}{rr} A_7\,A_6\,A_5\,A_4 & A_3\,A_2\,A_1\,A_0 \\ +\ B_7\,B_6\,B_5\,B_4 & B_3\,B_2\,B_1\,B_0 \\ \hline S_7\,S_6\,S_5\,S_4 & S_3\,S_2\,S_1\,S_0 \end{array}$$

During subtraction operation, the SUB signal is put to high state. Therefore the controlled inverter produce the 1's complement of B_7 ..... B_0. Furthermore because SUB is the CARRYIN to the first full adder, the circuit processes the data like this,

$$\begin{array}{rr} A_7\,A_6\,A_5\,A_4 & A_3\,A_2\,A_1\,A_0 \\ +\ \overline{B}_7\,\overline{B}_6\,\overline{B}_5\,\overline{B}_4 & \overline{B}_3\,\overline{B}_2\,\overline{B}_1\,\overline{B}_0 \\ \hline S_7\,S_6\,S_5\,S_4 & S_3\,S_2\,S_1\,S_0 \end{array}$$

When A_7.....A_0 = 0, the circuit produces the 2's complement of B_7 .....B_0 because 1 is being added to the 1's complement B_7.....B_0. When A_7.....A_0 doesnot equal zero, the effect is equivalent to adding A_7.....A_0 and the 2's complement of B_7.....B_0.

5.9 Fast Adder :-

A fast adder is a type of adder used in digital logic. It is commonly called carry-look ahead adder or parallel adder. A fast adder improves speed by reducing the amount of time required to determine carry bits. It calculate one or more carry bits before the sum, which reduces the wait time to calculate the result of larger value bits. Thus it bring parallelism is addition process.

Parallel adder (fast adder) generates the carry in advance. This is achieved by following method.

The general equation of carry generation for fast adder can be written as, (as written for full adder).

$$C_i = A_iB_i + (A_i + B_i)\,C_{i-1}$$

This can be written as,

$$C_i = G_i + P_i\,C_{i-1}$$

Where, $G_i = A_iB_i$

$P_i = A_i + B_i$

Gi stands for generation of carry and P_i stands for propagation of carry is particular stage depending on input to that stage.

Starting from LSB, designated for suffix 0,

$$C_0 = G_0 + P_0\,C_{-1}$$

$$C_1 = G_1 + P_i\,C_0 = G_1 + P_1\,(G_0 + P_0\,C_{-1})$$
$$= G_1 + P_1G_0 + P_1\,P_0\,C_{-1}$$

Similarly,

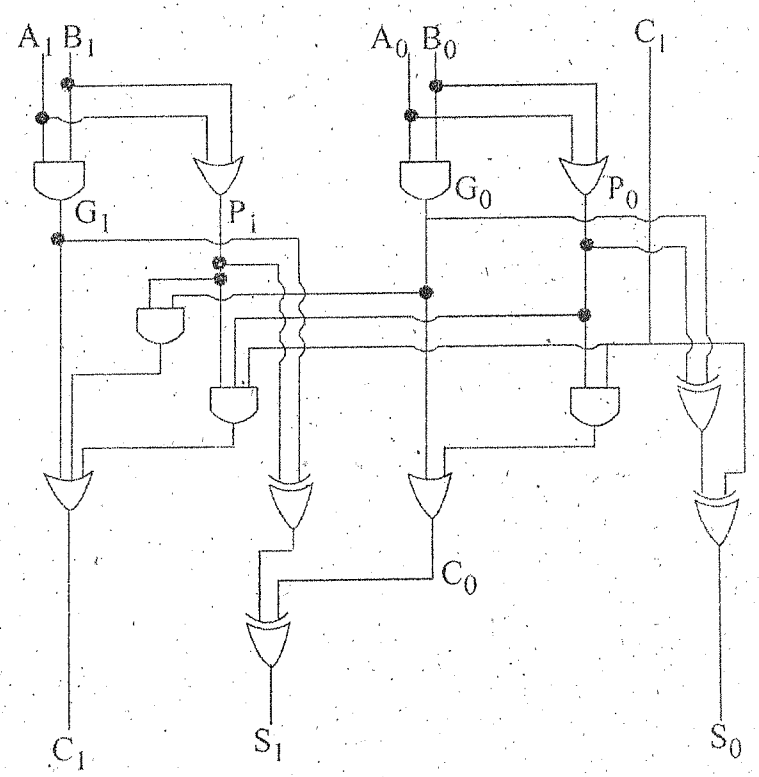
$$C_2 = G_2 + P_2\,C_1 = G_2 + P_2\,(G_1 + P_1\,G_0 + P_1P_0\,C_{-1})$$
$$= G_2 + P_2\,G_1 + P_2\,P_1\,G_0 + P_2\,P_1P_0\,C_{-1}$$

& so on.

These equation can be realized in hardware using multi-input AND and OR gate in two levels. Now, for each carry whether C_0 or C_2 we require only two gate delays once the G_i and P_i are available. As they are already available after one gate delay thus parallel adder generates carry within 1+2 = 3 gate delays.

After the carry is available at any stage there are two more gate delays from Ex-OR gate to generate the sum as we can write

$$S_i = G_i \oplus P_i \oplus C_{i-1}$$



Note :- Parallel Adder increase the hardware complexity for large n.

5.10 Arithmetic Logic Unit :-

In digital electronics, an Arithmetic Logic Unit (ALU) is a digital circuit that performs integer arithmetic and logic operations with appropriate input selection. It represents the fundamental building block of central processing unit of a computer. Modern CPUs contain very powerful and complex ALUs.

Arithmetic operations are addition, subtraction, multiplication and division. Logical operations are comparison of values such as NOT, AND and OR.

5.11 Binary Multiplication and Division :-

Multiplication is done with addition instruction. Let's begin with four most basic cases of binary multiplication,

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

some examples

(1) Multiply $(1011)_2$ by $(111)_2$

```
  1011          1011
× 111         × 111
------        ------
  1011          1011
+1 011×       1011×
------       +1011××
100001       ------
             (100001)2
```

(1) Multiply $(11100)_2$ by $(101)_2$

```
  11100
× 101
------
  11100
 00000×
+11100××
--------
(10001100)2
```

Divide is done with subtraction instruction

$0 \div 1 = 0$
$1 \div 1 = 1$

Some examples

(1) Divide (110111) by $(101)_2$

```
101) 110111 ( 101
   - 101↓ | |
     0011 | |
   -  000↓ |
      111  |
    - 101↓ |
       101 |
     -  101
        ×
```

∴ Quotient = $(101)_2$
Remainder = ×

(2) Divide $(110011)_2$ by $(101)_2$

```
101) 110011 ( 1010
   - 101↓ | | |
     010  | | |
   -  000↓ | |
      101  | |
    - 101↓ | |
       001 | |
     -  000 |
        1
```

∴ Quotient = $(1010)_2$
Remainder = 1

Old Question/Solution

1. *Implement 1:4 DEMUX using VHDL.* *[2069 Ashad]*

Ans:
```
Module 1 to 4DEMUX (A,S1, S1,D0, D1, D2, D3)
input A, S0, S1
output D0, D1, D2, D3;
assign D0 = A & ≈ S0 & ≈ S1;
assign D1 = A & ≈ S0 & S1;
assign D2 = A & ≈ S0 & ≈ S1;
assign D3 = A & ≈ S0 & S1;
end module
```

2. ***Design full adder using HDL.*** ***[2069 Ashad, 2068 Shrawan]***

Ans:
```
module full adder (A, B, C, Sm, Cr);
input A, B, C
output Sm, Cr;
assign Sm = A^B^C;
assign Cr = (A&B) | (B&C) | (C&A);
end module
```

3. ***Design 2 to 4 line decoder using HDL.*** ***[2068 Baisakh]***

Ans:
```
module 2 to 4 decoder (A, B, D0, D1, D2, D3);
input A, B;
output D0, D1, D2, D3;
assign D0 = ≈A & ≈B;
assign D1 = ≈A & B;
assign D2 = A & ≈B;
assign D3 = A & B;
end module
```

4. ***Convert the following numbers from the given base to based indicated.*** ***[2067 Ashad]***

Sol^n: a) Octal 623.77 to decimal, binary and hexadecimal

Octal to decimal

$(623.77)_8 = 6 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 + 7 \times 8^{-1} + 7 \times 8^{-2}$

$= 403 + 0.98437$

$= (403.98437)_{10}$

Octal to binary

Here we have,

$(623.77)_8 = (403.98437)_{10}$

Now,

2	403	→ 1
2	201	→ 1
2	100	→ 0
2	50	→ 0
2	25	→ 1
2	12	→ 0
2	6	→ 0
2	3	→ 1
	1	→ 1

Next,

$(0.98437)_{10} \Rightarrow 0.98437 \times 2 = 1.9874 \rightarrow 1$

$0.9874 \times 2 = 1.39748 \rightarrow 1$

$0.39748 \times 2 = 1.87498 \rightarrow 1$

$0.87498 \times 2 = 1.74992 \rightarrow 1$

$0.75 \times 2 = 1.499 = 1.5 \rightarrow 1$

$0.5 \times 2 = 1$

$\therefore (0.98437)_{10} = (111111)_2$

$\therefore (403)_{10} = (110010011)_2$

Here,

$(623.77)_8 = (110010011.111111)_2$

Octal to Hexadecimal

Here,

$(623.77)_8 = (110010011.111111)_2$

Making group of 4,

$= (0001\ 1001\ 0011.\ 1111\ 1100)_2$

$= (193.FC)_{16}$

$\therefore (623.77)_8 = (193.FC)_{16}$

b) Hexadecimal 2AC5.D to decimal, Octal & binary

Sol^n : Hexadecimal to decimal,

$(2AC5.D)_{16} = 2 \times 16^3 + 10 \times 16^2 + 12 \times 16^1 + 5 \times 16^0 + 13 \times 16^{-1}$

$= (8192 + 2560 + 192 + 5 + 0.8125)_{10}$

$= (10949.8125)_{10}$

Hexadecimal to Octal

Here we have,

$(623.77)_8 = (10949.8125)_{10}$

Now,

2	10949	→ 5
2	1368	→ 0
2	171	→ 3
2	21	→ 5
	2	→ 2

Next,

$0.8125 \times 8 = 6.5 \rightarrow 6$

$0.5 \times 8 = 4$

$= (64)_8$

$\therefore (2AC5.0)_{16} = (25305.64)_8$

Hexadecimal to binary,

$(2AC5.0)_{16} = (0010\ 1010\ 1100\ 0101.\ 1101)_2$

$= (10101011000101.1101)_2$

5. ***Perform the subtraction with the following decimal & binary using 9's and 1's complement respectively.*** ***[2067 Ashad]***

(i) 3570 – 2100 (Using 9's complement)

(ii) 10010 – 10011 (Using 1's complement)

Ans: (i) 3570 – 2100

Here,

9's complement of 2100 = 9999 – 2100 = 7899

$\therefore$ 3570

+7899

11469

Since the end carry 1 is present, the final answer will be,

$$\begin{array}{r} 1469 \\ +\ \ 1 \\ \hline 1470 \end{array}$$

$\therefore 3570 - 2100 = 1470$

(ii) 10010 – 10011

Here, 1's complement of 10011 = 01100

$$\begin{array}{r} \therefore\ \ 1001\ 0 \\ +0110\ 0 \\ \hline 1111\ 0 \end{array}$$

Since there is no end carry the answer will be, 1's complement of 1111 0 i.e. 00001

$\therefore$ 10010 – 10011 = 00001

6. ***Subtract 8 from 6 using binary representation using 1's complement numbers whenever necessary verify your answer by comparing it to decimal subtraction.*** ***[2066 Bhadra]***

Sol^{n}: First we convert 8 & 6 into binary

$8 \rightarrow 1000$

$6 \rightarrow 0110$

Now,

1's complement of 8. i.e 1000 is 0111

To find's complement of 8 we add 1 to 1's complement as,

$$\begin{array}{r} 0111 \\ +\ \ 1 \\ \hline 1000 \end{array}$$

Next,

$$\begin{array}{r} 0110 \\ +\ 1000 \\ \hline 1110 \end{array}$$

Since there is no end carry; the answer will be,

- (2's complement of 1110)

1's complement of $1110 \rightarrow 0001$

2's complement of $1110 \rightarrow 0001$

$$\begin{array}{r} +\ \ 1 \\ \hline 0010 \end{array}$$

$\therefore 6 - 8 = -(0010)$

For Decimal subtraction,

$$\begin{array}{r} 6 \\ -\ 8 \\ \hline -\ 2 \end{array}$$

Hence verify

7. ***Perform the following conversions as indicated.*** ***[2065 Shrawan]***

Soln: **(a) $(65.31)_8 = (?)_{16}$**

Here,

$$(65.31)_8 = (110\ 101 \cdot 011\ 001)$$
$$= (0011\ 0101 \cdot 0110\ 0100)_2$$
$$= (35.64)_{16}$$

(b) $(25.25)_{10} = (?)_2$

Here,

2	25	$\rightarrow 1$
2	12	$\rightarrow 0$
2	6	$\rightarrow 0$
2	3	$\rightarrow 1$
	1	$\rightarrow 1$

Next,

$0.25 \times 2 = 0.5 \rightarrow 0$

$0.5 \times 2 = 1 \rightarrow 1$

$\therefore (25.25)_{10} = (11001.01)_2$

(c) $(56.8)_{16} = (?)_{10}$

Here,

$$(56.8)_{16} = 5 \times 16^1 + 6 \times 16^0 + 8 \times 16^{-1}$$
$$= 80 + 6 + 0.5$$
$$= 86.5$$

$\therefore (56.8)_{16} = (86.5)_{16}$

(d) $(1100110.1011)_2 = (?)_8$

Here,

$$(1100110.1011)_2 = (001\ 100\ 110\ .\ 101\ 100)_8$$
$$= (146.58)_8$$

(e) $(101.101)_2 = (?)_{10}$

Here,

$$(101.101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-2} + 1 \times 0$$
$$= 4 + 1 + 0.5 + 0.125$$
$$= (5.625)_{10}$$

$\therefore (101.101)_2 = (5.625)_{10}$

8. *Implement outputs of full adder circuit by using 8 × 1 MUX.* *[2064 Falgun]*

Ans: The truth table for full adder is

A	B	C	S	C
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,

Sum [S (x, y, z)] $= \sum (1, 2, 4, 7)$

Carry [C (x, y, z)] $= \sum (3, 5, 6, 7)$

Implementing the circuit using 8 × 1 MUX

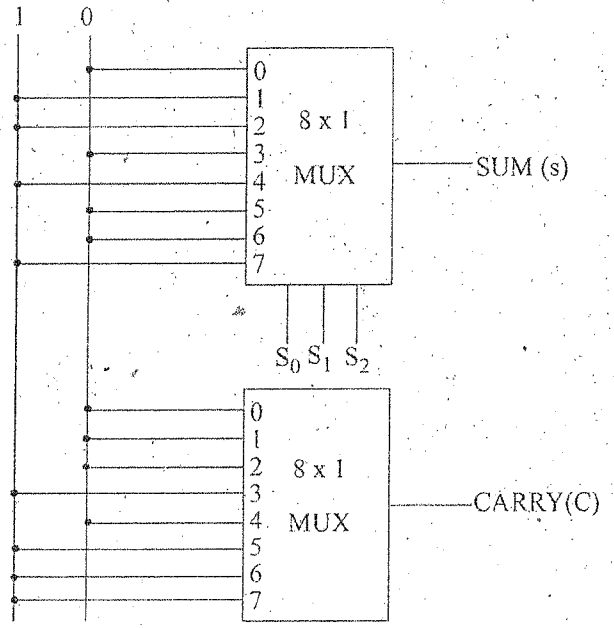
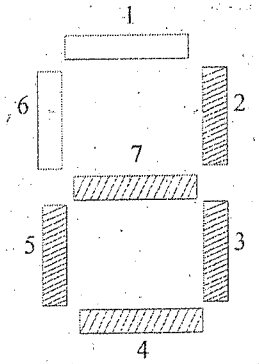


Fig:- Full adder Circuit Using 8:1 MUX

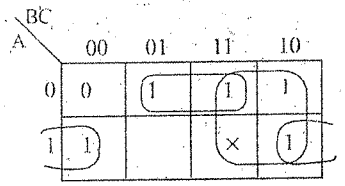
9. *Find a simple logic circuit for 'd' segment of seven-segment display decoder and apply NAND gates only in the circuit.* *[2064 Falgun]*

Sol^n: To display the letter 'd' the seven segment display must glow the section 2, 3, 4, 5, 7 and sections 1 & 6 must be off as shown,



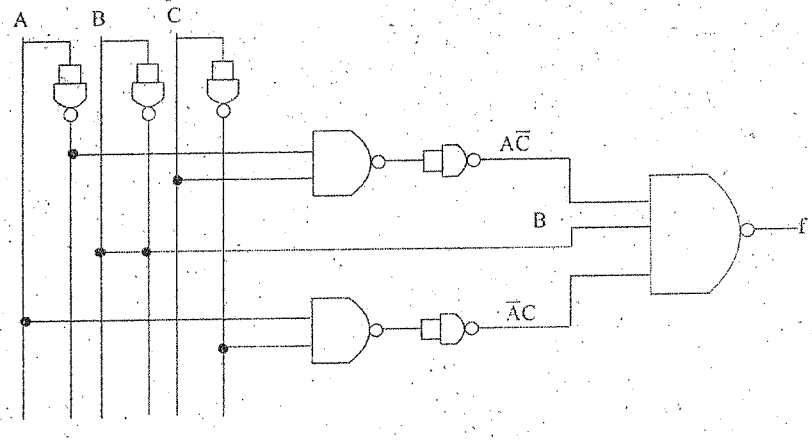
So, $F = \sum (2, 3, 4, 5, 7)$

Now simplifying the function using K-map,



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

Now implementing the circuit using NAND gates only,



CHAPTER 6
Flip Flops

Introduction to flip flop :-

A flip flop is a binary storage device capable of storing one bit of information. In a stable state, the output of flip flop is either 0 or 1. Hence a flip flop is a bistable electronic circuit also known as basic digital memory circuit having two stable states i.e. its output is either low or high.

The two output of flip flop are denoted by Q and $\bar{Q}$ and there results are complementary of each other.

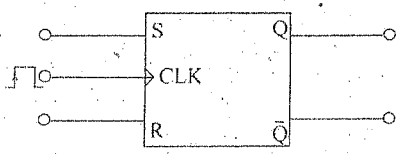
Edge Trigger Flipflop :-

(1) **SR Flipflop:-** **[2069 Ashad, 2070 Ashad]**

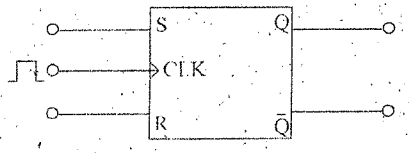
SR flipflop has two inputs namely SET(S) and RESET(R) and two output Q and $\bar{Q}$. The two input S and R are synchronous input because data on there inputs are transferred to be flipflop's output only on triggering the edge of clock pulse.

Working:-

- When S is high, R is low, the output Q goes high on triggering edge of clock pulse and flipflop is set.
- When S is low, R is high, the output Q goes low on triggering edge of clock pulse and flipflop is reset.
- When both inputs S and R are low , the output remain same i.e. flipflop remains in its present state
- When both inputs S and R are high , an invalid condition exists.



Block Diagram of positive edge trigerred SR flipflop (or Logic symbol)



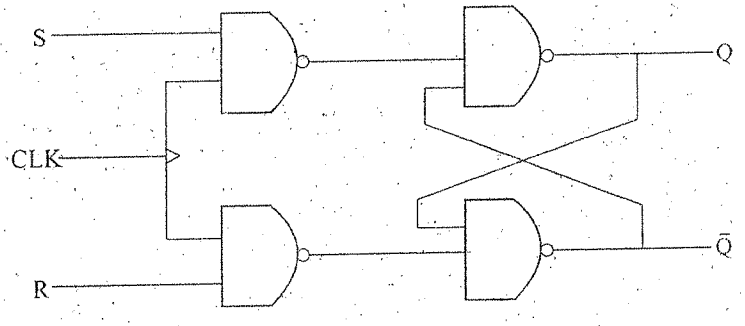
Block Diagram of positive edge trigerred SR flipflop (or Logic symbol)

Truth Table

Input			Outputs		Comments
S	R	CLK	Q	$\bar{Q}$	
1	0	↑	1	0	Set
0	1	↑	0	1	Reset
0	0	×	Q_0	$\bar{Q}_0$	No change
1	1	↑	?	?	Invalid/forbidden

↑ denotes the clock transition from low to high

Logic Circuit:-



Characteristics table :-

Q_n	S	R	Q_{n+1}	Remarks
0	0	0	0	No change
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	×	Invalid
1	0	0	1	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	×	Invalid

Representing in K-Map as shown,

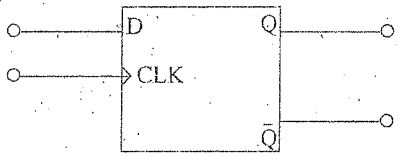
Q_n \ SK	00	01	11	10
0	0	0	×	1
1	1	0	×	1

$$\therefore Q_{n+1} = S + Q_n\bar{R}$$

This is the characteristics equation of SR flipflop.

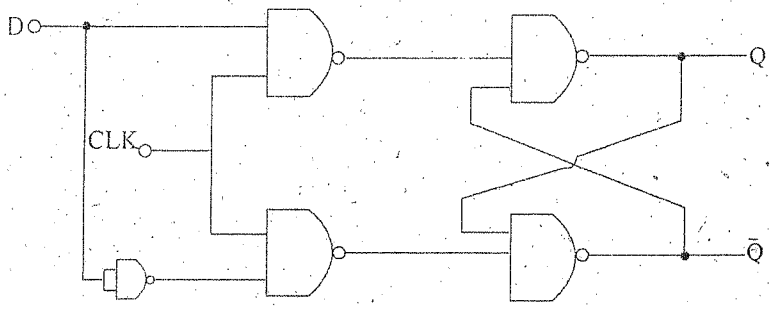
(2) D- Flipflop (Data or Delay Flipflop):- [2069 Chaitra]

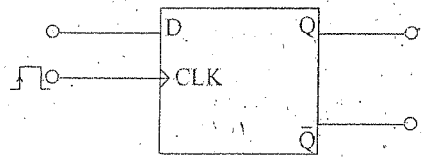
D-flipflop has only one input and two output. It is nothing but RS flipflop with as inverter in the R-input. The added inverter reduces the number of inputs from two to one. Sometime this type of flipflop is also called gated D-catch.



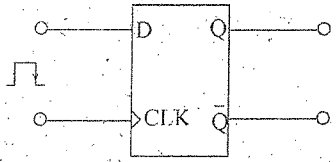
Logic Symbol

Logic Circuit Using NAND gate:-





Block Diagram of positive trigerred D-flipflop



Block Diagram of Negative edge trigerred D-flipflop

Truth table:

Inputs		Outputs		Comments
D	CLK	Q	$\bar{Q}$	
1	↑	1	0	Set
0	↑	0	1	Reset

Characteristics table :-

Q_n	D	Q_{n+1}	Remarks
0	0	0	Reset
0	1	1	Set
1	0	0	Reset
1	1	1	Set

Q_n \ D	0	1
0	0	1
1	0	1

$\therefore Q_{n+1} = D$

This is the characteristics equation of D-flipflop.

Since the transfer of data from the input to output is delayed due to the presence of NOT gate, the flipflop is called Delay flipflop.

Advantages of D-flipflop over RS flipflop:- [2068 Chaitra Regular]

In a clocked R-S flip-flop two input signals are required to drive the flip-flop which is a disadvantage with many digital circuits. In some events, both input signals become high which is again an undeniable condition. So these draw-barks of clocked R-S flip-flop are overcome in D-flip-flop. D-flip-flop has only one input to drive the flip-flop.

(3) J-K Flipflop:-

[2068 Baishak, 2067 Ashad, 2066 Bhadra, 2064 Jestha]

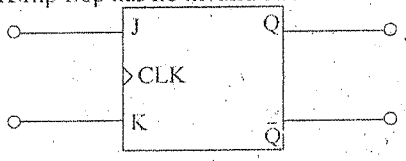
J-K flipflop is the refinement of RS flipflop. The invalid state of RS flipflop is overcome in J-K flipflop by defining it as Toggle state.

The J and K input of JK flip-flop are synchronous input because data on there inputs are transferred to be flipflop output only on trigerring the edge of clock pulse.

Working

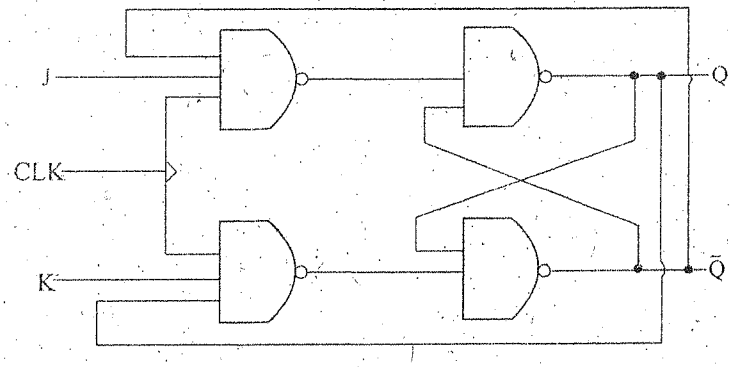
- When J is high, K is low, the output Q goes high on trigerring the edge of clock pulse and flip-flop is set.
- When J is low, K is high, the output Q goes low on trigerring the edge of clock pulse and flip-flop is reset.
- When both J and K are low, the output Q doesnot changes from its prior state.
- When both J and K are high, a toggle mode condition appears.

Hence the working of JK flip-flop is identical to that of SR flip-flop in RESET, SET and no change conditions of operations. The difference is that the J-K flip flop has no invalid state as does the RS flipflop.



Logic Symbol

Logic Circuit Using NAND gates:-



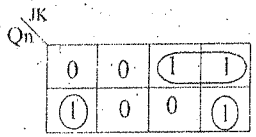
Truth table:

Inputs			Outputs		Comments
J	K	CLK	Q	$\bar{Q}$	
1	0	↑	1	0	Set
0	1	↑	0	1	Reset
0	0	↑	Q_0	$\bar{Q}_0$	No change
1	1	↑	$\bar{Q}_0$	θ_0	Toggles

Characteristics table :-

Q_n	J	K	Q_{n+1}	Remarks
0	0	0	0	No change
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	1	Toggle
1	0	0	1	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	0	Toggle

Now,

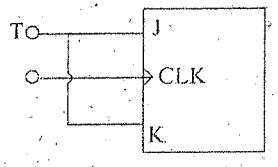


$$\therefore Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$$

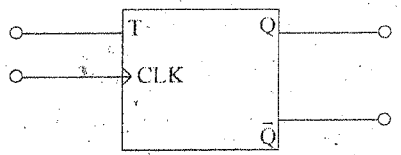
This is the characteristics equation of JK flipflop.

(4) T Flip-flop:-

T flip-flop is basically a J-K flip flop. In this circuit, input terminals J and K are connected with each other and this input is named as T. In other words we can say that, the J and K input are shorted.

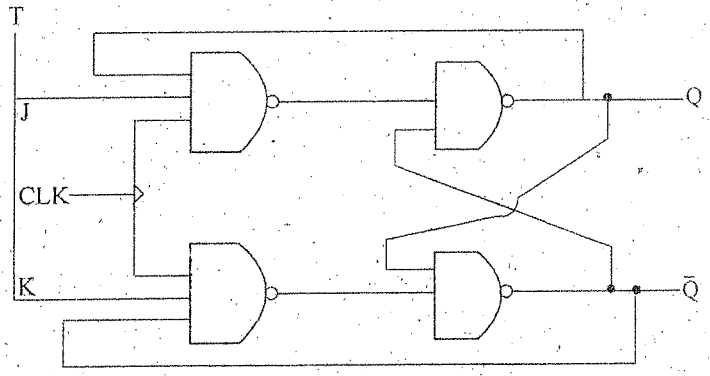


Circuit Diagram



Logic Symbol

Logic Diagram:-



Working:-

- When T is low then initial state of output of flip flop remains the same, after the arrival of clock pulse.
- When T is high then output of flip flop toggles after arrival of every new clock pulse.

Truth table:-

Input	Output
0	Q_0
1	$\bar{Q}_0$

From truth table it is clear that if T = 1 if acts as a toggle switch. For every clock pulse, the output changes.

Note:- An R-S flip flop cannot be converted into a T-type because R=S=1 is not permitted.

In essence the T-flip flop can be treated as frequency divider or more generally as a device which takes the input frequency at the clock terminal and divides it by 2.

Characteristics table :-

Q_n	D	Q_{n+1}	Remarks
0	0	0	No change
0	1	1	Toggles
1	0	1	No change
1	1	0	Toggles

Q_n \ T	0	1
0	0	1
1	1	0

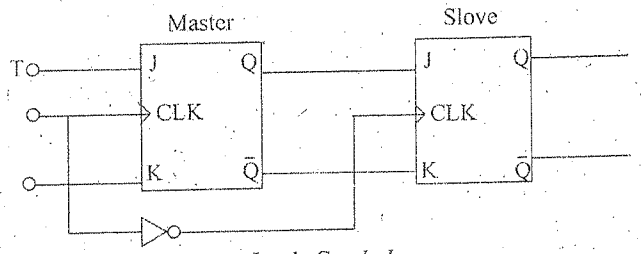
$$\therefore Q_{n+1} = \overline{Q_n}\,T + Q_n\overline{T}$$
$$= Q_n \oplus T$$

This is the characteristics equation.

(5) Master Slave J-K flip flop :- **[2069 Falgun]**

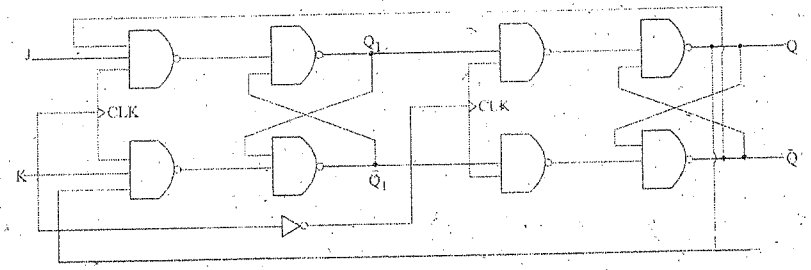
A master slave flip flop is constructed from two separate flip flop, one circuit serves as master and other as a slave and overall combination is referred to as master slave flip flop.

The master is positive edge trigerred and slave is negative edge trigerred. The master section is basically a gated latch and slave is also the same except that it is clocked on inverted clock pulse and is controlled by output of master section rather than by external JK input.



Logic Symbol

Logic Circuit:-



Working:-

- When J = 1, K = 0; master sets on positive clock edge and $Q_1 = 1$ of master flip flop drives the input of J of slave on negative clock edge causing Q = 1.
- When J = 1, K=0; master resets on (positive) clock edge and $Q_1 = 1$ of master flip flop drives the input of J of slave on negative clock edge causing Q = 1.
- When J = 1, K = 0; it toggles clock edge and the slave then toggles on negative clock edge.

Excitation table of Different Flip flop:-

The table which determines the input of flip-flop based on present output (Q_n) and next output (Q_{n+1}) is called excitation table, table-excitable plays a vital role in analysis of problem related to sequential circuits.

The table below shows the excitation table of different flip-flop:-

Q_n	Q_{n+1}	S	K	J	K	D	T
0	0	0	×	0	×	0	0
0	1	1	0	1	×	1	1
1	0	0	1	×	1	0	1
1	1	×	0	×	0	1	0

Note:- To remember the excitation table, remember

S = ZOZD R = DZOZ
J = ZODD K= DDOZ

Where Z = zero
O = one
D = Don't care

Various Representation of Flip flop:-

(1) Characteristics table and characteristics equation :-

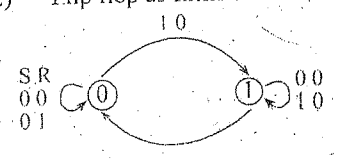
A characteristics table defines the logical properties of a flip-flop by describing its operation in tabular form characteristics table defines the next state as a function of the inputs and present state.

The logical properties of a flip-flop, as described in the characteristics table, can be expressed algebraically with a characteristics equation.

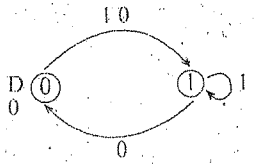
The Characteristics eq^n of various flip-flop are:

SR : $Q_{n+1} = S + Q_n\overline{R}$
D : $Q_{n+1} = D$
JK : $Q_{n+1} = \overline{Q_n}\,J + Q_n\overline{K}$
T : $Q_{n+1} = \overline{Q_n}\,T + Q_n\overline{T}$

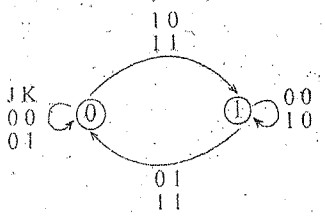
(2) Flip flop as finite state machine:-



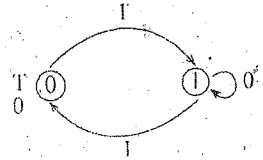
(a) SR flip-flop



(b) D flip-flop



(c) JK flip-flop



(d) T flip-flop

Fig:- State Transition Diagram

Conversion from One type to other type of Flip-Flop:-

(1) Convert SR flip flop to J K flip flop

[2069 Ashad, Bask 2064 Falgun]

Step 1: First draw the excitation table of SR flip flop as shown,

Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

Step 2: Then draw the characteristics and excitation table for JK flip flop as shown,

Q_t	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Characteristics Table for JK flip flop

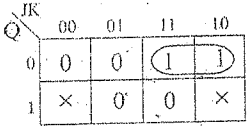
Q_t	Q_{t+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Excitation table for JK flip flop

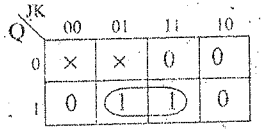
Step 3: Now draw the synthesis table as shown with the aid of step 1 and step 2.

Q_n	J	K	S	R	Q_{n+1}
0	0	0	0	×	0
0	0	1	0	×	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	×	0	1
1	0	1	0	1	0
1	1	0	×	0	1
1	1	1	0	1	0

Step 4: Write SR inputs as a function of JK inputs and θ_t using K-Map as shown,



$S = \bar{Q}_t J$



$R = Q_t K$

Step 5: Draw the equivalent circuit using SR flip flop and JK input as shown,

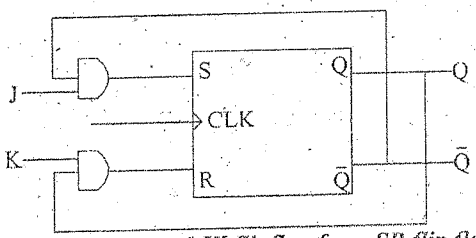


Fig:- Realization of JK flipflop from SR flip flop

(2) Convert D flip flop to J K Flip flop

→ Excitation table of D-flip flop is,

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

→ **Excitation table of JK flip flop is,**

Q_t	Q_{t+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

→ **Characteristics table of JK flip flop is,**

Q_t	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Draw the synthesis table an shown,

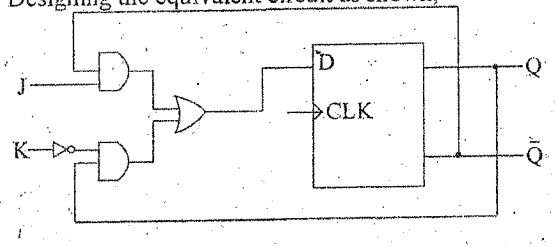
Q_t	J	K	Q_{t+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

Writing D input as a function of JK input using K-map,

Q_n \ JK	00	01	11	10
0	0	1	1	1
1	0	0	1	1

$$D = J\overline{Q}_t + \overline{K}Q_t$$

Designing the equivalent circuit as shown,



(3) **Convert SR to D Flip flop**

→ **Drawing the synthesis table as shown,**

Q_t	D	Q_{t+1}	S	R
0	0	0	0	×
0	1	1	1	0
1	0	0	0	1
1	1	1	×	0

Writing D input as a function of JK input using K-map,

S: Q_n \ D	0	1
0	0	1
1	0	×

$S = D$

R: Q_n \ D	0	1
0	×	0
1	1	0

$R = \overline{D}$

Designing the equivalent circuit as shown,

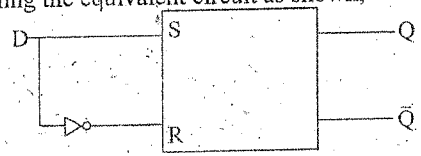


Fig:- D-flip flop using SR flipflop

Switch Contact Bound Circuits:- **[2068 Shrawan]**

Contact Bounce is a common problem associated with mechanical switching devices and relays. Any mechanical switching device consists of a moving contact arm restrained by some sort of spring system. As a result, when the arm is moved from one stable position to other, the arm bounces much as a hard ball bounces when dropped on a hard surface.

When the contacts strikes together their momentum and elasticity acts together to cause bounces. The result is rapidly pulsed electric volt instead of a clean transition from OV to full voltage.

Consider a contact bounce circuit as shown,

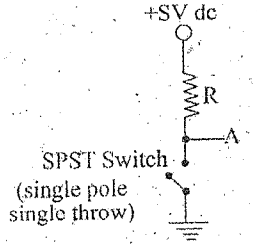


Fig:- Contact Bounce Circuit

When the switch is open the voltage at point A is + 5V dc. When the switch is cleared the voltage at A must be OV ideally as shown in fig. below,

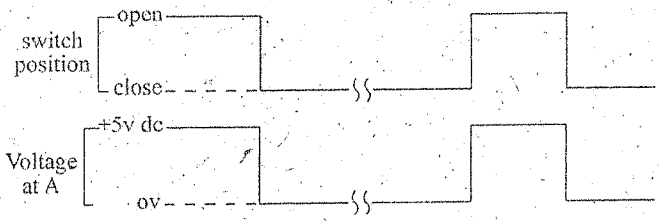


Fig:- Ideal voltage A

But is actual practice, the voltage waveform at A will appear as shown below due to the phenomenon known as contact bounce.

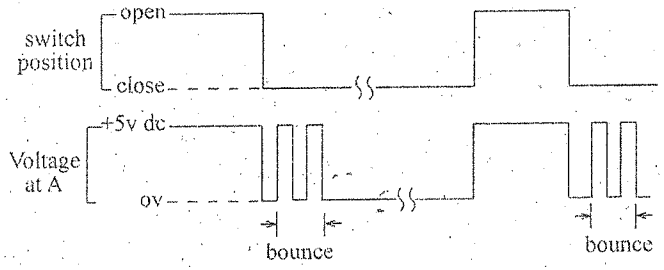


Fig:- Voltage at A showing contact bounce

Elimination of Contact Bounce:- **[2068 Shrawan]**

→ The problem of contact bounce can be eliminated with the help of SR latch as debounce circuit.

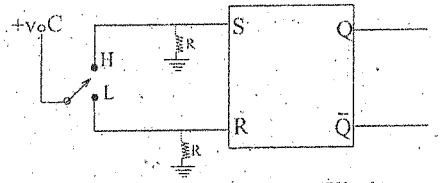


Fig:- Switch Contact Bounce Eliminator

When the switch is moved to position H, R = 0 & S = 1, then flip flop sets with Q = 1

When the switch bounces losing contact then R = S = 0 therefore the flip flop remains sets with Q = 1

Analysis of Sequential Circuit:-

A logic circuit where output at any instant of time depend not only on present inputs but also on part outputs are called sequential circuits.

A sequential circuit contains flip flops as memory elements and also contains logic gates as combinational circuits. Analysis of circuit helps to explain its performance.

Consider the sequential circuit as shown in figure below. The output X is generated from flip flop outputs as shown,

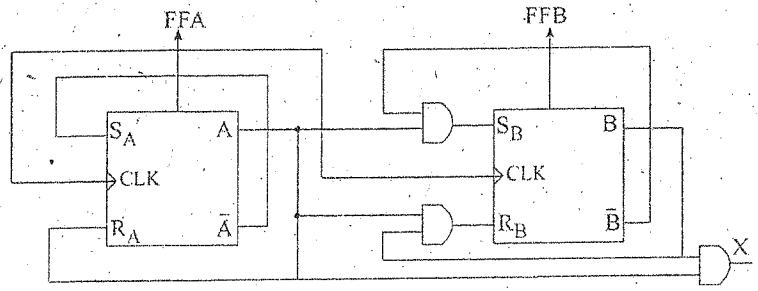


Fig:- A Sequential logic Circuit for Analysis

From the Circuit, flip flop input relations;

$S_A = \bar{A}_n$, $R_A = A_n$ for FFA

$S_B = A_n\bar{B}_n$, $R_B = A_nB_n$ for FFB

Using the characteristics equations for SR flipflop to represent next output i.e. $Q_{n+1} = S + Q_n\bar{R}$ we can write,

For FFA,

$$A_{n+1} = S_A + A_n\,\bar{R}_A$$
$$= \bar{A}_n + \bar{A}_n A_n \qquad [\because S_A = \bar{A}_n \;\&\; R_A = A_n]$$
$$= \bar{A}_n$$

For FFB,

$$B_{n+1} = S_B + \bar{R}_B B_n$$
$$= A_n\bar{B}_n + \bar{B}_n\,\overline{A_nB_n}$$
$$= A_n\bar{B}_n + (\bar{A}_n + \bar{B}_n)\,B_n \text{ [using De-Margan's Theorem]}$$
$$= A_n\bar{B}_n + \bar{A}_n B_n + \bar{B}_n B_n$$
$$= A_n\bar{B}_n + \bar{A}_n B_n + 0$$
$$= A_n\bar{B}_n + \bar{A}_n B_n$$
$$= A_n \oplus B_n$$

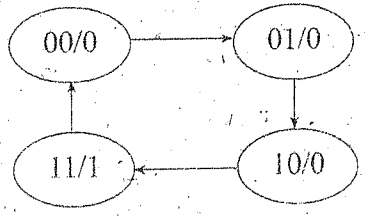
Writing the equation for output,

$X = A_nB_n$

Writing state Analysis table,

Present State		Present Input				Next State		Present
B_n	A_n	$S_B = A_n\overline{B_n}$	$R_B = A_nB_n$	$S_A = \overline{A_n}$	$R_A = A_n$	$B_{n+1} = \overline{A_n} \oplus \overline{B_n}$	$A_{n+1} = \overline{A_n}$	Output X
0	0	0	0	1	0	0	1	0
0	1	1	0	0	1	1	0	0
1	0	0	0	1	0	1	1	0
1	1	0	1	0	1	0	0	1
0	0	0	0	1	0	0	1	0

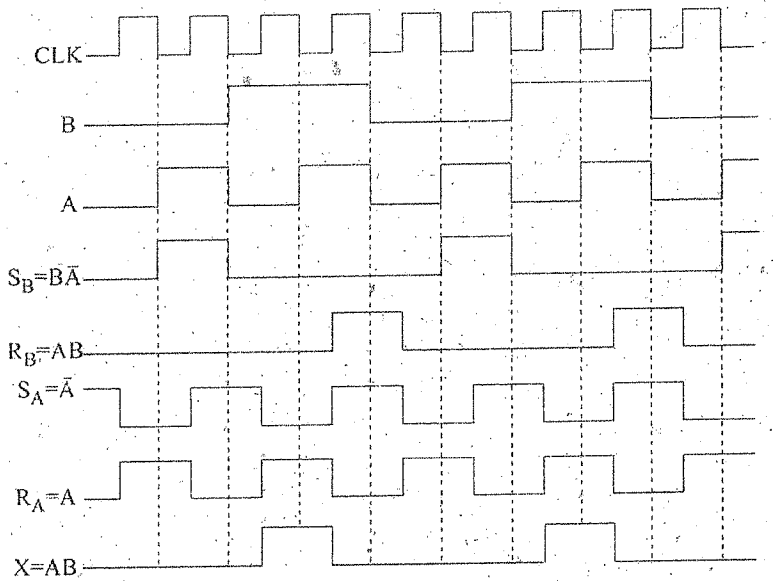
State Diagram:-



The value within the circle follows the system: B_nA_n/Y_n

Present state ← → Output

Hence the sequential circuit is Analyzed using flip flop



Application of Flip flop:

→ For data storage or as memory element.
→ In various types of registers.
→ In counters/times.
→ For frequency division (T-flip flop)
→ As delay elements
→ Other computer applications

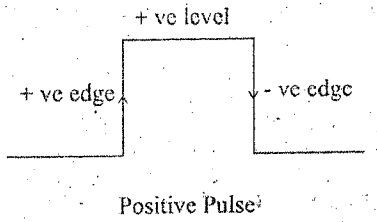
Old Question Solutions

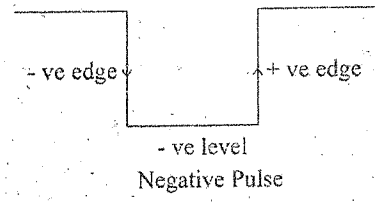
1) Differentiate between edge trigerred and pulse trigerred flip-flops [3]. [2070 chaitra] [2064 Falgun]

Ans:

→ Edge trigerred flip-flop :- It is a flip flop that change its state either at positive edge (rising edge) or at negative edge (falling edge) of a clock pulse and is sensitive to its input only at then transmission of clock e.g, S-R, D, J-K etc.

→ Pulse trigerred flipflop :- it is a flip flop that changes its state either at positive pulse or at negative pulse of the applied clock pulse.





CHAPTER 7

Registers

Introduction to Register:-

A register is a group of flip-flop, each one of which is capable of storing one bit of information. An n-bit register consists of a group of flip-flop capable of storing n bits of binary information. In addition to the flip-flop, a register may have combinational gates that perform certain data processing tasks.

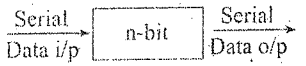
Shift Registers:-

A register capable if shifting the binary information held in each cell to its neighboring cell in a selected direction is called a shift register. The logical configuration of a shift register consists of a chain of flip-flop is cascade, with the output of one flip-flop connected to the input if next flip-flop. All flip-flop receive common clock pulse.

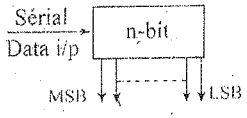
- A register capable of shifting in one directing only is a unidirectional shift register.
- One that can shift in both directions is a bi-directional shift register.
- If the registers has both shifts and parallel load capabilities, it is referred to as universal shift register.

7.1 Types of shift Registers:-

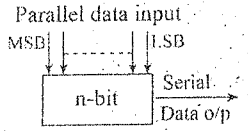
1) Serial in-Serial out (SISO)
2) Serial in-Parallel out (SIPO)
3) Parallel in-Serial out (PISO)
4) Parallel in-Parallel out (PIPO)



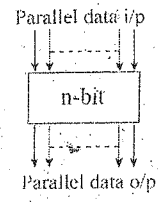
SISO



SIPO



PISO



PIPO

7.2 Serial in-Serial out (SISO):- [2069 Ashad, 2068 Chairta, 2068 Shrawan]

The SISO register accepts data serially i.e. one bit at a time on a single line. It produces the stored information on its output also in serial form.

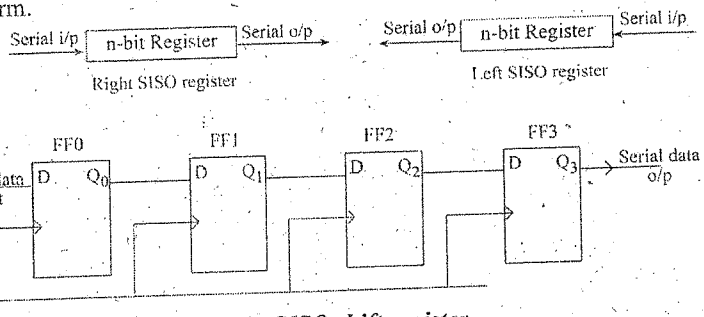


Fig:-Right SISO shift register

Figure above illustrates the entry of four bits into the register beginning with right most bit. The register is initially clear. Let 1011 is stored from right most bit.

Working:-

The first right most bit is 1. It is put into data input line making D=1 for FFO. When the first clock pulse is applied, FFO is set storing 1.

Next the second bit which is also 1 is put into data input line making D= 1 for FFO & D=1 for FF1 because the D input of FF1 is connected to θ_0 output when the second clock pulse is applied, the 1 on data input line is shifted to FFO and 1 stored on FFO is shifted to FF1.

The third bit 0 is now put into data input line and the clock pulse is applied. The 0 on data input line is shifted to FFO, 1 on FFO is shifted to FF1 & 1 on FF1 is shifted to FF2.

Finally the fourth bit 1 is put into data line. After applying clock pulse FFO stores 1, FF1 stores 0, FF2 stores 1 & FF3 stores 1.

Hence all the bits are stored in the register.

Clock	Register Content				
Initially	Q_0	Q_1	Q_2	Q_3	
CLK 1	0	0	0	0	
CLK 2	1	0	0	0	
CLK 3	1	1	0	0	
CLK 4	0	1	1	0	
CLK 5	1	0	1	1	→ All data stored after 4th CLK pulse
CLK 6	0	0	1	0	
CLK 7	0	0	0	1	
CLK 8	0	0	0	0	→ Register clear after 8th CLK pulse

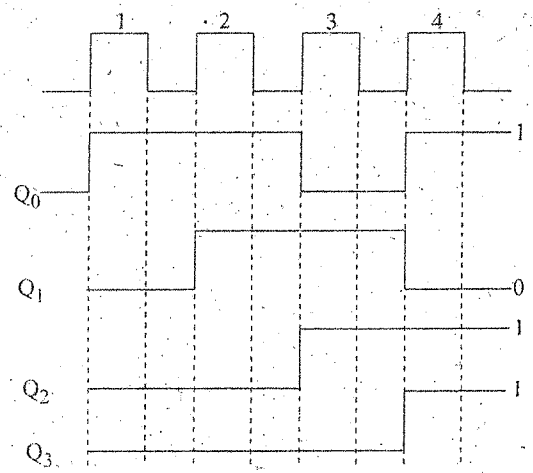


Fig:- Timing diagram for Storing 1011; SISO register

7.3 Serial in-Parallel out (SIPO):- [2067 Ashad, 2064 Jestha]

The Serial in-parallel out shift register accepts data serially i.e one bit at a time on a single line but produces the stored information on its output parallely.

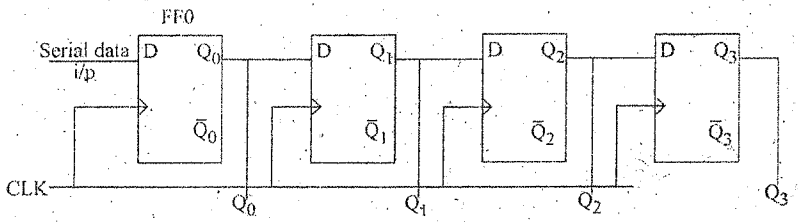


Fig:-Logic Diagram for SIPO shift register

Figure above illustrates the input of four bits on SIPO shift register beginning with Right most bit. Let 1010 is to be stored in the register. The register is initially clear.

Clock	Register Content				
Initially	Q_0	Q_1	Q_2	Q_3	
CLK 1	0	0	0	0	
CLK 2	1	0	0	0	
CLK 3	0	1	0	0	
CLK 4	1	0	1	0	Data is completely stored

Here the output is available at a time parallel.

Timing Diagram:-

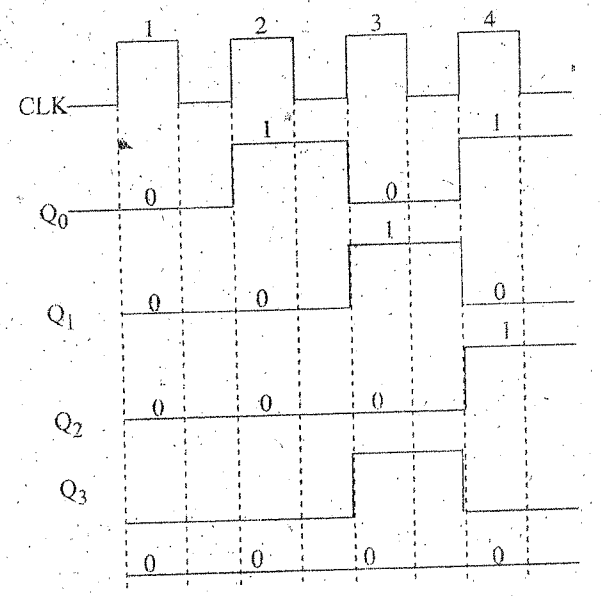


Fig:-Timing Diagram for Storing 1010 in SIPO

7.4 Parallel in-Serial out (PISO):- [2068 Baishak, 2065 Sharwan, 2063 Baishak, 2070 Ashad]

The Parallel in-Serial out register accepts data simultaneously into their respective stages on parallel line rather than on a bit by bit basis but it produces the stored inform action on its output in serial form.

Figure below illustrate a 4-bit parallel in serial out shift register and a typical logic symbol. There are four data input lines, D_0, D_1, D_2, and D_3 and a SHIFT/$\overline{\text{LOAD}}$ input, which allows four bits of data to load in parallel into register. When SHIFT/$\overline{\text{LOAD}}$ is LOW, gates G_1 through G_3 are enabled, allowing each data bit to be applied to the D input of its respective flip-flop. When clock pulse is applied, the flip-flop with D = 1 will set and those with D = 0 will reset, thereby storing all four bits simultaneously.

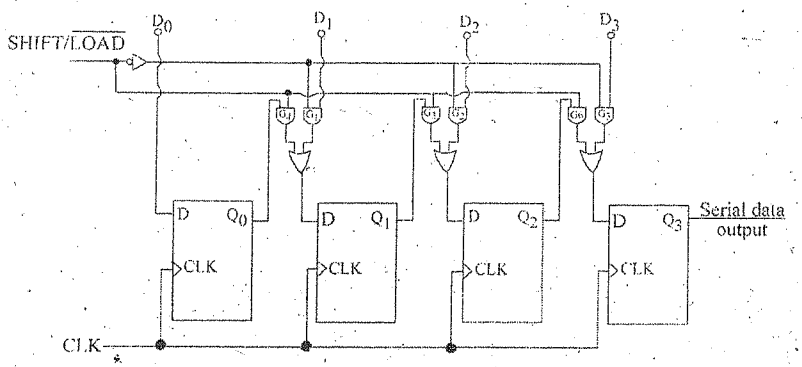


Fig:-Logic Diagram

When SHIFT/$\overline{\text{LOAD}}$ is HIGH, gates G_1 through G_3 are disabled and gates G_3 through G_6 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operating or the parallel data entry operation, depending on which AND gates are enabled by the level on the SHIFT/$\overline{\text{LOAD}}$ input.

7.5 Parallel in-Parallel out (PIPO):-

The Parallel in-Parallel out shift register accepts data simultaneously into their respective stages on parallel line rather than on a bit by bit basis. The output is also obtained simultaneously.

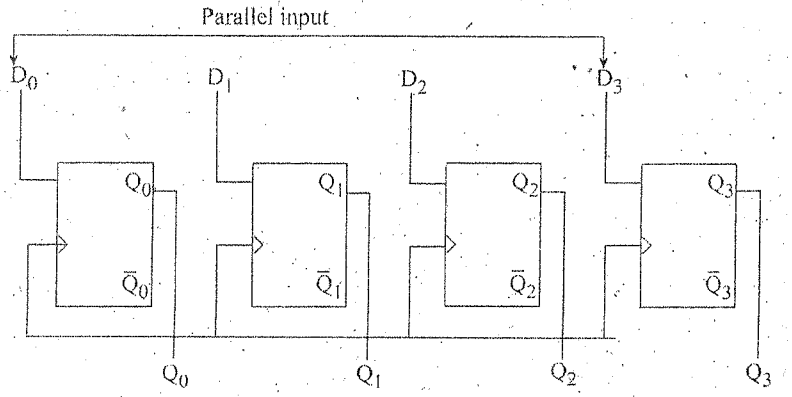
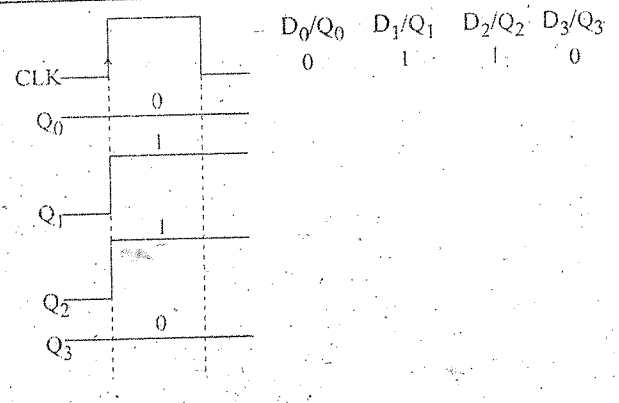


Fig:- Logic Diagram for 4-bit PIPO shift register



Timing Diagram for storing 0110 in PIOP shift register

7.6 Applications of Shift Registers:-

1) Shift Register can be used to store & shift binary data entered to it.
2) It is used in digital systems like multiplies, divider, microprocessor etc.
3) It can be used to change data from a serial format into a parallel format & vice-versa.
4) Registers are used for counter design (As Ring or Johnson Counter)

CHAPTER 8
Counters

Introduction To Counter:- [2068 Baishak]

A counter is a sequential circuit consisting a set of flip-flops in predetermined sequence to count the sequence of input pulses presented to it in digital form. It is probably one of the most useful and versatile subsystems in digital system.

Counters are classified into 2 broad categories on the basis they are clocked –

1) Synchronous Counter
2) Asynchronous Counter

8.1 Asynchronous Counter :-

In asynchronous counter flip-flop are connected in such a way that the first flip-flop clocked by external clock pulse and then each successive flip-flop is clocked by the output of the proceeding flip-flop. Hence all flip-flops do not change states simultaneously. They are serial counter also known as Ripple counter.

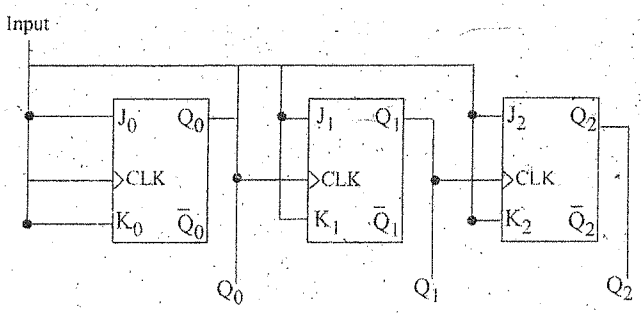


Fig:- 3-bit Asynchronous Counter

8.2 Synchronous Counter :- [2069 Ashad]

In Synchronous counter, all the flip-flop are clocked by a common clock pulse. Hence all flip-flops changes state simultaneously. They are parallel counter and fast in operation compared to Asynchronous counter.

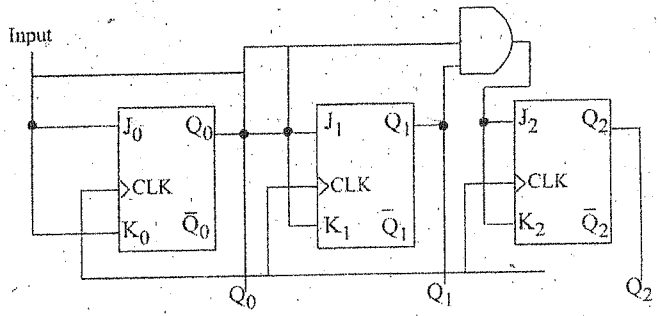


Fig:- 3-bit Synchronous Counter

Advantage of Synchronous Counter Over Asynchronous:- [2066 Bhadra]

1) In synchronous Counter all flip-flop are clocked simultaneously. by a common there is a limit to highest operating frequency. This drawback has been overcome in synchronous counter by trigerring every flip-flop simultaneously and hence there is no propagation delay.
2) Synchronous Counter is free from glitches.
3) Synchronous Counter are faster in operation than asynchronous counter

3-Bit Ripple Up Counter (Asynchronous Counter):- (MOD-8)

An n-bit counter in general counts from 0 to $2^n - 1$ in decimal. Hence a 3-bit counter counts from 0 to 7 i.e. 000 to 111. The maximum number of states for any bit counter – 2^n. So for this care it in $2^3 = 8$.

A 3-bit Ripple Up counter is as shown in figure. A ripple counter consists of J K or T flip-flop. The ripple counter shown below is constructed from JK flip-flop. Here the J & K inputs are tied together to common input +Vcc such that at each negative transition of its clock input, the flip-flop toggles its states.

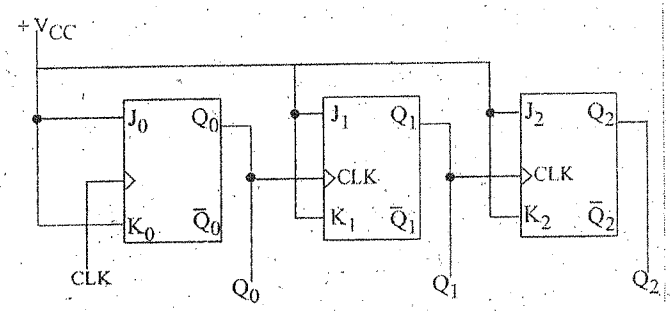
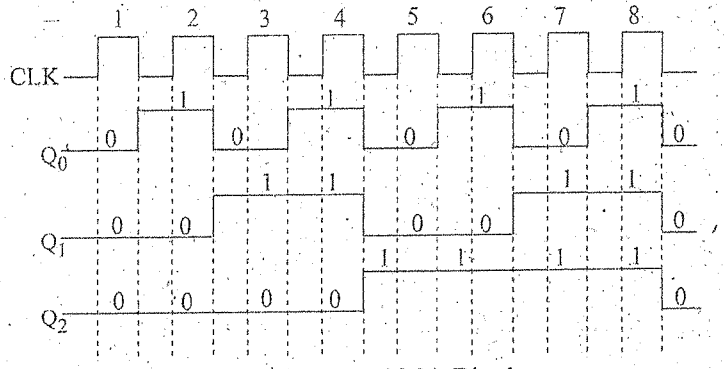


Fig:- 3-bit Ripple Up Counter

Truth-Table

No. of states	Q_0	Q_1	Q_2	Count
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7



Timing Diagram of 3-bit Ripple

3-bit Ripple Down Counter (MOD-8):- [2065 Shrwan]

Figure below shows the 3-bit Ripple Down Counter constructed from JK flip-flop. Here only first flip-flop is given clock pulse from external source. The remaining flip-flop gets clocked from the output of preceding flip-flop.

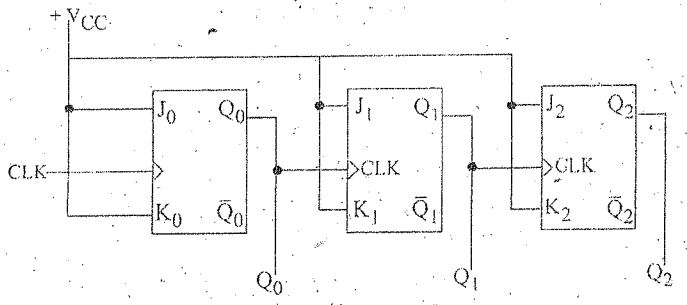
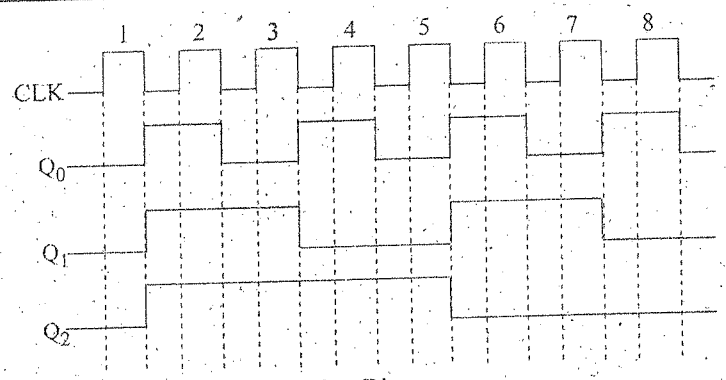


Fig:-3-bit Ripple Down Counter

Truth-Table

No. of states	Q_0	Q_1	Q_2	Count
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0



Timing Diagram

3-bit Ripple Up & Down counter:-

It is the combination of ripple up & ripple-down counter. It is also known as multimode counter because of its ability to count upper old and down old.

In up counter each flip-flop is trigerred by the normal output of proceeding flip-flop. In down counter each flip-flop is trigerred by the output of proceeding flip-flop.

In both the counter, first flip is trigerred by external clock pulse.

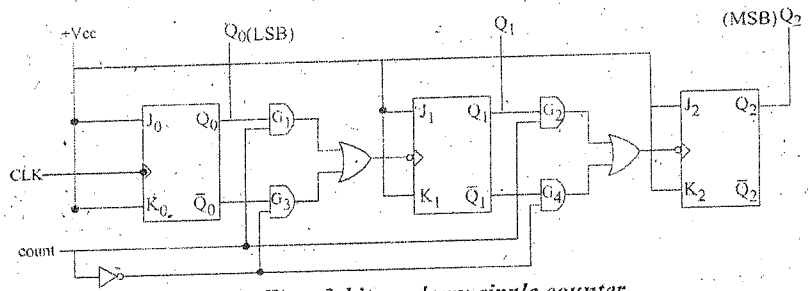


Fig:- 3-bit up-down ripple counter

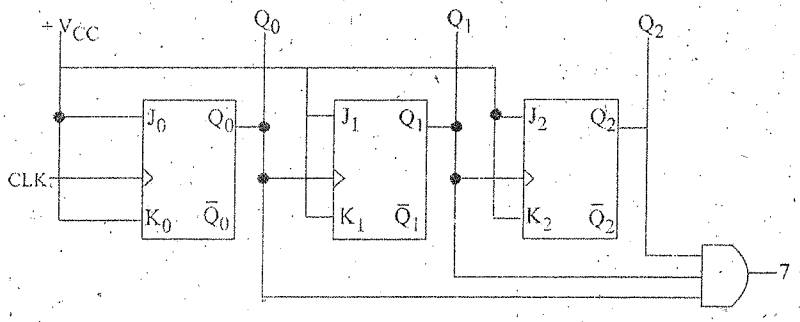
The working of Ripple Up-Down counter is controlled by "count"
When count = 1, gates G_1 & G_2 are enabled & it performs up count.
When count = 0, gates G_3 & G_4 are enabled & it performs down count.
For Up-Count the truth table & timing diagram will be same as that of 3-bit Ripple up counter.
For Down-Count the truth table & timing diagram will be same as that of 3-bit Ripple down counter.
Here, the gate will be high only when $Q_0 = Q_1 = Q_2 = 1$ i.e. state 7

8.3 Decoding Gates:- [2068 Chaitra]

A decoding gate is a logic gate where output is high (or low) only during one of the unique states of a counter.

A decoding gates can be connected to the outputs of a counter in such a way that output of the gate will be high or low only when the counter contents are equal to a given state.

For eg:- A decoding gate of state 7 for 3-bit Ripple counter is as shown,



3-bit Synchronous Up counter (MOD-8 Synchronous Up counter)
Initially $Q_0 = Q_1 = Q_2 = 0$ i.e. state 111, when the first clock pulse is applied, Q_0 toggles to 1. J_1 is still 0 for first clock pulse hence $Q_1 = 0$. The low Q_1 is fed to one leg of AND gate which makes $J_2 = Q$ resulting $Q_2 = 0$.
Here the gate will be high only when $Q_0 = Q_1 = Q_2 = 1$

When second clock pulse is applied, Q_0 toggles to 0 and J_1 is no more equal to 0 i.e. $J_1 = 1$. So Q_1 toggles to 1 and Q_2 remains 0.

When third clock pulse is applied, Q_0 toggles to 1, $J_1 = 0$ so $Q_1 = 0$ and $Q_2 = 0$.
When fourth clock pulse is applied, Q_0 toggles to 1, $J_1 = 1$ so Q_1 toggles to 0 then $J_2 = Q_0 Q_1$
$= 1.1$
$= 1$ and so on.

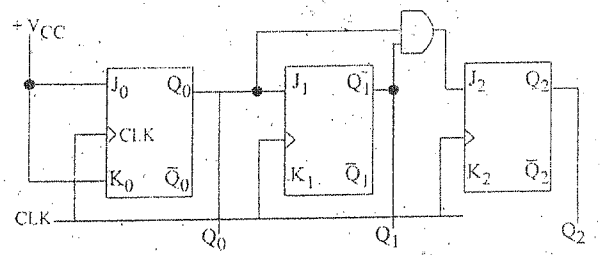


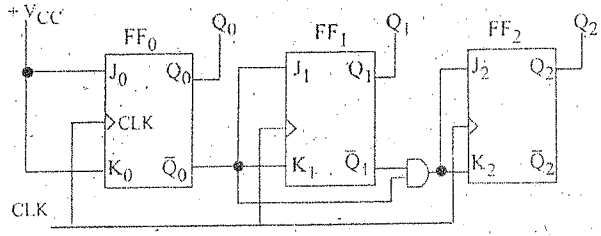
Fig:- MOD-8 binary or synchronous up counters

Truth-Table

No. of states	Q_2	Q_1	Q_0	Count
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	0	7

Now you can draw the timing diagram on your own.

3-bit Synchronous Down counter (MOD-8 Synchronous down counter):-
Figure below shows the 3-bit synchronous down counter constructed from JK flip-flop.



No. of states	Q_2	Q_1	Q_0	Count
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0

3-bit Synchronous UP/DOWN Counter:-

It is the combination of Synchronous up & synchronous down counter. It is also known as multimode counter because of its ability to count upward & downward.

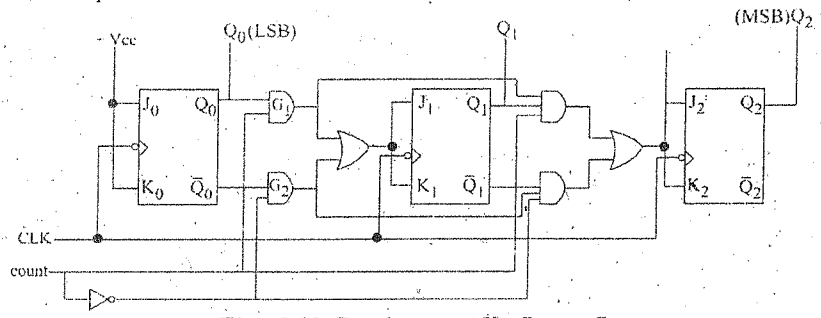


Fig:- 3-bit Synchronous Up-Down Counter

The working of this counter is also controlled by "COUNT:

When count = 1, AND gate 1 is "ON" no output of Q_0 is transferred to J_1 and K_1 so it starts counting up.

When count = 0, AND gate 2 is "ON" no output of $\bar{Q}_0$ is transferred to J_1 and K_1 so it starts counting down.

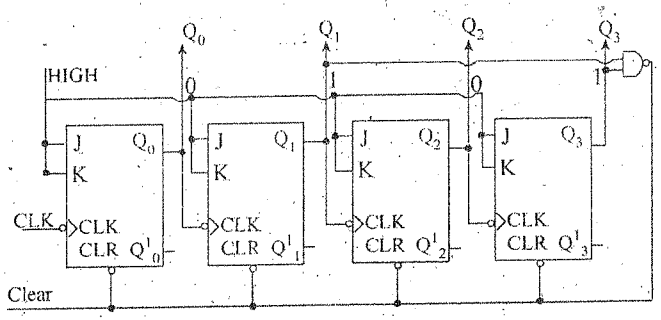
8.4 Decode Counters:- (Decimal, MOD-10 or BCD counter)

[2069 Ashad]

A decimal/decode counter is a circuit of flip-flop in cascade, which counts from 0 to 9 in decimal(0000 to 1001). It means that there is a sequence of 10 distinct counts in increasing order.

Three FFs used in cascade, progress through 8 distinct states (000 – 111) while four FFs in cascade progress through 16 distinct states (0000 – 1111). Hence to get a count of 10, a minimum of 4FFs are required.

Pluses	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	0
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0



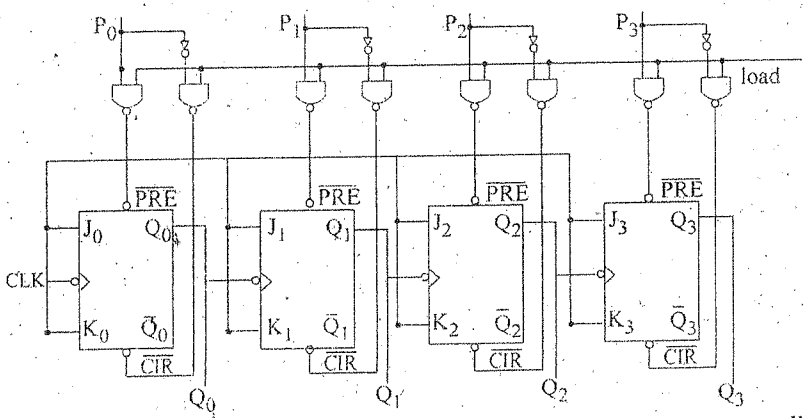
When the counter counts Q_3 Q_2 Q_1 Q_0 = 1001 then the output of NAND gate is low so it clean the flip-flop & all FF resets to zero.

Note:- Decade counters find widespread use in applications where pulse or events are to be counted and the results displayed on some type of decimal numerals read out. A decade counter also often used for dividing a pulse frequency exactly by 10.

8.5 Presettable Counters:-

Pre-settable Counters are those counters, which counts from any desired number rather than from zero. Pre-settable counters may be synchronous or asynchronous.

Figure below shown 4-bit pre-settable counters constructed from JK flip-flop. It has a special to LOAD terminal and counting begins with P_3, P_2, P_1, P_0 which may be any number between 0000 – 1111.



When LOAD is low, all NAND gates have high outputs so all PRESET and CLEAR are inactive. It counts in normal way.

When LOAD is high, the inputs P_3, P_2, P_1 , P_0 and their complement $\overline{P_3}$, $\overline{P_2}$, $\overline{P_1}$, $\overline{P_0}$ passes through NAND gates. This action presets the counters to P_3, P_2 , P_1 , P_0 states in the initial.

When LOAD reverts to Low again, then the counting starts and successive clock pulse produces the remaining states till to maximum.

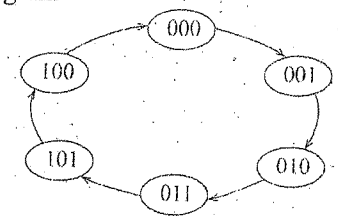
8.6 Counter Design as Synthesis Problem:-

Steps to follow for Designing Counter :-

1) Draw the state diagram from the question given.
2) Draw the state table and find the number of FF required Note: (state table consists of complete information about present state, next state and outputs of a sequential circuit. The number of FF required is equal to the no of bits of counter)
3) By selecting a suitable flip-flop for the design determine the FF excitation table based on transition of present state (Q_n) to next state (Q_{n+1}).
4) Use K-Map to simplify the input function of FF selected.
5) Connect the circuit using FF and other gates according to the minimized expression.

Example 1 : Design a MOD-6 synchronous counter and draw its timing diagram **[2068 Baishak]**

Step 1: State Diagram



Step 2 and Step 3:-

Lets choose a JK flip-flop. The excitation table of JK flip-flop is,

Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

State Table:-

Present State (Q_n)			Next State (Q_{n+1})			FF Excitation					
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	1	0	1	×	0	0	×	1	×
1	0	1	0	0	0	× 1		0	×	×	1

Note :- See $Q_{2n} \rightarrow Q_{2n+1}$ transition to write J_A & K_A excitation table.
$Q_{1n} \rightarrow Q_{1n+1}$ transition to write J_B & K_B excitation table.
$Q_{0n} \rightarrow Q_{0n+1}$ transition to write J_C & K_C excitation table.

Stet 4:- Draw the K-map to simply the input function of FF.

For J_A ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	0	1	0
1	×	×	×	×

$\therefore J_A = Q_{1n} Q_{0n}$

For K_A ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	×	×
1	0	1	×	×

$\therefore K_A = Q_{0n}$

For J_B ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	1	×	×
1	0	0	×	×

$\therefore J_A = \overline{Q}_{2n} Q_{0n}$

For K_B ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	1	0
1	×	×	×	×

$\therefore K_B = Q_{0n}$

For J_C ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	1	×	×	1
1	1	×	×	×

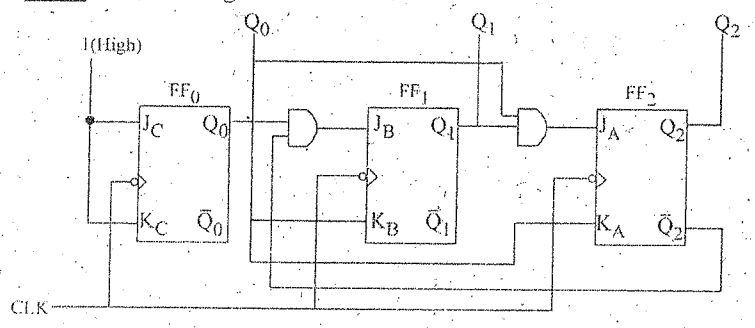
$\therefore J_C = 1$

For K_C ,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	1	1	×
1	×	1	×	×

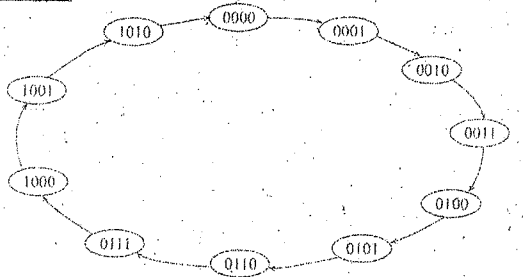
$\therefore K_C = 1$

Step 5:- Circuit Diagram.



Example 2 : Design a MOD-11 binary counter by showing its state, circuit diagram and output waveforms.

State Diagram:-



For Convenience lets choose T-flip-flop. The excitation table for T-flip-flop is,

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State Table:-

Present State (Q_n)				Next State (Q_{n+1})				FF Exeitation			
Q_{3n}	Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	0	0	0	1	0	1	0

Simplifying the input function of FF using K-Map as shown,

For T_A ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	×	×	×	×
10	0	0	×	1

$T_A = Q_{2n} Q_{1n} Q_{0n} + Q_{3n} Q_{1n}$

For T_B ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	×	×	×	×
10	0	0	×	0

$T_B = Q_{1n} Q_{0n}$

For T_C ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	×	×	×	×
10	0	1	×	1

$T_C = Q_{0n} + Q_{3n} Q_{1n}$

For T_D ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	×	×	×	×
10	1	1	×	0

$T_D = \bar{Q}_{1n} + \bar{Q}_{3n}$

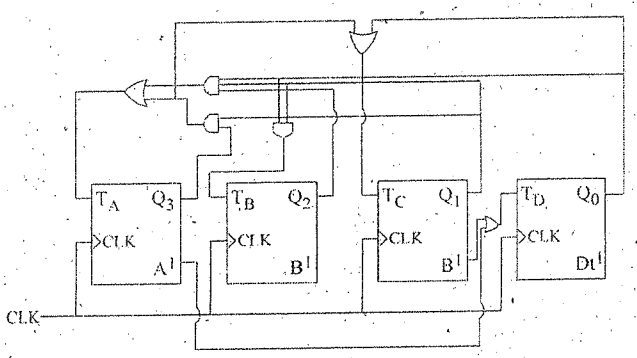
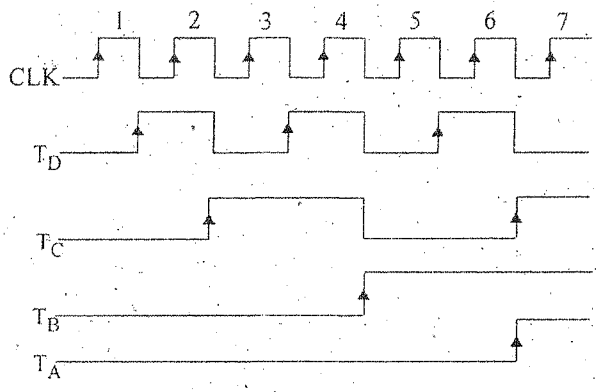


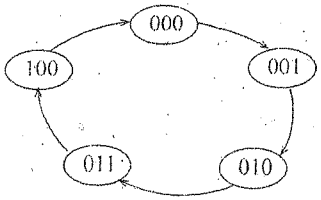
Fig: MOD 11 Counter



Timing Diagram

Example 3 : Design MOD-5 synchronous counter. The flip-flip being used in the design are your choice. Draw a logic circuit in output timing diagram for seven clock cycles. **[2068 Chaitra, 2066 Bhadra, 2064 Falgun]**

State Diagram:-



For Convenience we use J-K flip-flop
The excitation table of J K flip-flop is,

Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

State Table:-

Present State (Q_n)			Next State (Q_{n+1})			FF Excitation					
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	0	0	0	×	0	0	×	0	×

Simplifying the input function RF of using K-Map as shown,

For J_A,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	0	1	0
1	×	×	×	×

$\therefore J_A = Q_{1n}\,Q_{0n}$

For K_A,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	×	×
1	1	×	×	×

$\therefore K_A = 1$

For J_B,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	1	×	×
1	0	×	×	×

$\therefore J_B = \overline{Q}_{0n}$

For K_B,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	0	1
1	×	×	×	×

$\therefore K_B = \overline{Q}_{0n}$

For J_C,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	1	×	×	1
1	0	×	×	×

$\therefore J_C = X_1$

For K_C,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	1	1	×
1	×	×	×	×

$\therefore K_C = 1$

This logic diagram will be as shown,

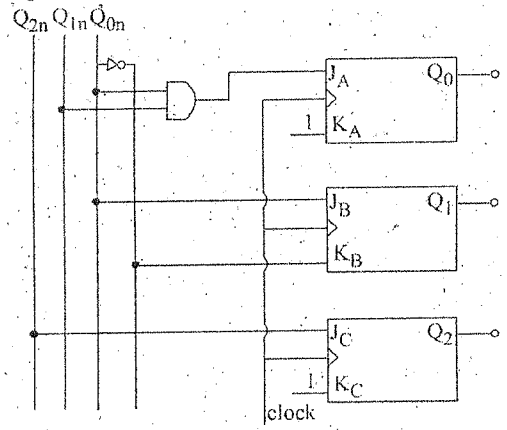
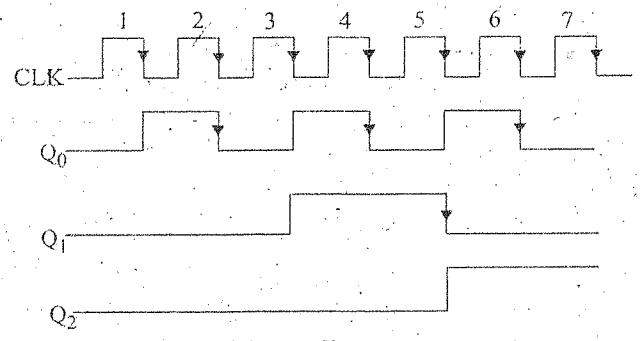


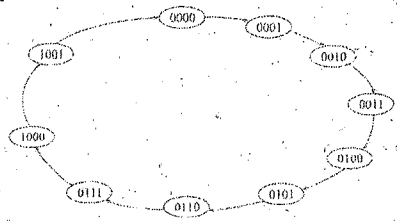
Fig:-MOD.5 Synchronous counter



Timing diagram

Example 4 : Design MOD-10 synchronous counter showing its state circuit diagram & output waveform. **[2067Ashad, 2068 Chaitra, 2068 Ashad]**

State Diagram:-



For Convenience we use T flip-flop
The excitation table of T flip-flop is,

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Present State (Q_n)				Next State (Q_{n+1})				FF Excitation			
Q_{3n}	Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

Simplifying the input function of FF using K-Map as shown,

For T_A ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	×	×	×	×
10	0	1	×	×

$T_A = Q_{3n}Q_{0n} + Q_{2n}Q_{1n}Q_{0n}$

For T_B ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	×	×	×	×
10	0	0	×	×

$T_B = Q_{1n}Q_{0n}$

For T_C ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	×	×	×	×
10	0	0	×	×

$T_C = \bar{Q}_A Q_D$

For T_D ,

$Q_{3n}Q_{2n}$ \ $Q_{1n}Q_{0n}$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	×	×	×	×
10	1	1	×	×

$T_D = 1$

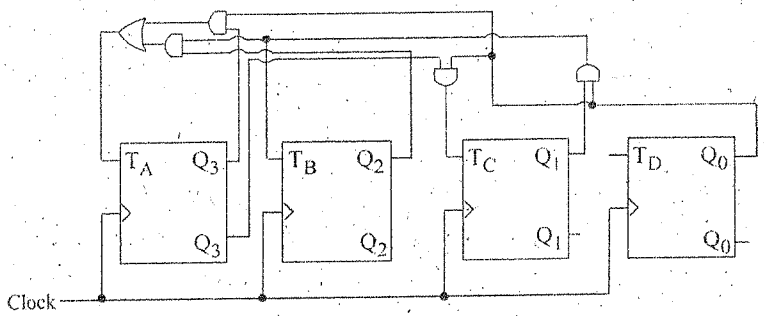
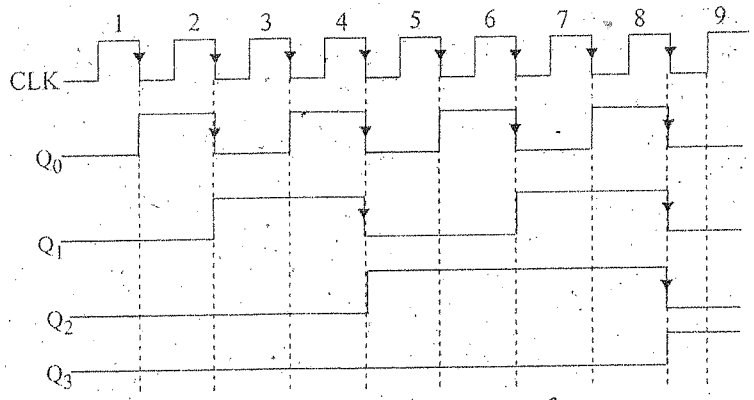


Fig:- MOD-10 synchronous counter



Timing diagram/output waveform

8.7 A digital clock:-

A digital clock is a type of clock that displays the time digitally i.e. in number or other symbols as opposed to an analog clock, where the time is indicated by the positions of rotating hands.

Digital clock typically uses the 50 or 60Hz oscillation of ac power or a 32,768Hz crystal oscillator as in quartz clock to keep time. To represent time, most digital clock uses a seven-segment LED, VFD or LCD display for each of four digits.

Suppose that we want to construct an ordinary clock which will display hours, minutes and seconds. The power supply for this system is the usual 60-Hz, 120Vac commercial power. In order to obtain pulses occurring at a rate of one each second, it is necessary to divide the 50 Hz power source by 60. If the resulting 1 Hz waveform is again divided by 60, a one-per-minute a one-per-hour waveform. This is the basic idea to be used in forming digital clock.

A clock diagram showing the functions to be performed is given in fig(1). The first divide by – 60 counter simply divides the 60Hz power signal down to a 1-Hz square wave. The second divide-by-60 counter changes state once each second and has 60 discrete states. It can therefore be decoded to provide signals to display seconds. This counter is then referred to as seconds counter.

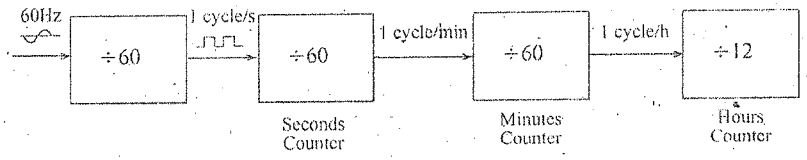
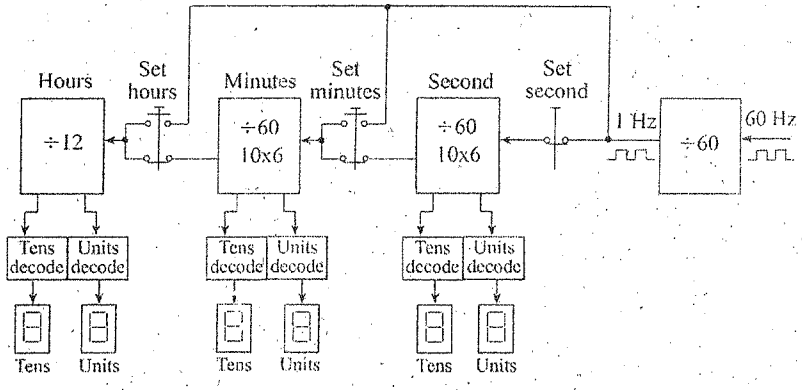


Fig :- Block Diagram of a Digital clock

The third divide-by-60 counter changes state once each minute and has 60 discrete states. It can thus decoded to provide the necessary signals to display minutes. This counter is then the minutes counter.

The last counter changes state one each 60 minutes (once each hour). Thus if it is divide-by-12 counter, it will have 12 states that can be decoded to provide signals to display correct hour. This counter is then hour counter.

By means of large-scale integration (LSI), it is possible to construct a digital clock entirely on one semiconductor chip.



Digital Clock

CHAPTER 9

Sequential Machines

Introduction:-

Sequential circuit is a type of logic circuit where output depends not only on the present value of its input signals but on the past history of its inputs. This is in contract to combinational logic, where output is a function of only present input. That is, sequential logic has state (memory) while combinational logic does not. In other words sequential logic is a combinational logic with memory.

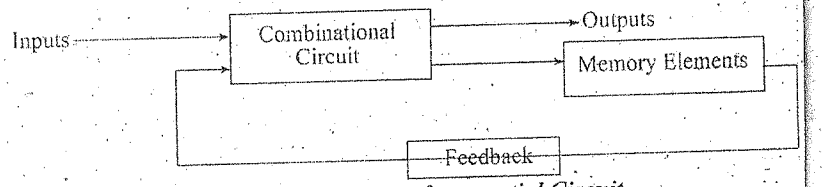


Fig:-Block Diagram of sequential Circuit

Types of Sequential Circuit:-

1) **Synchronous Sequential Circuit** :- It is a sequential circuit where the state changes only at discrete instant of times in response to clock signal.
2) **Asynchronous Sequential Circuit** :- It is a sequential circuit where the state changes at any time in response to changing inputs.

Part A: Design of Synchronous Sequential Circuit

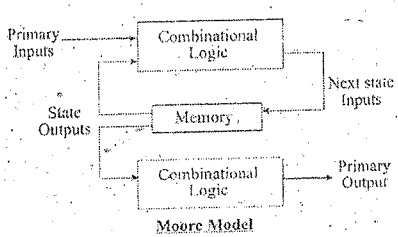
Design problem normally starts with a word description of input output relation and ends with a circuit diagram having sequential and combinational logic elements. The word description is first concerted to a state transition diagram followed by preparation of state synthesis table. For flip-flop based implementation, excitation tables are used to generate design equations through Karnaugh Map. The final circuit is designed from there design equations.

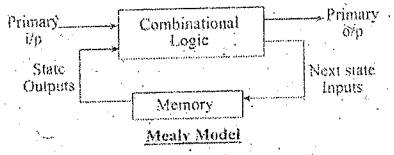
There are 2 distinct ways by which a synchronous sequential logic circuit can be designed- **Moore Model** and **Mealy Model**.

In Moore model the output depends only on present state and not on input. In Mealy model the output is derived from present state as well as input.

Comparison of Mealy Model & Moore Model:-

The option to include input in output generation logic gives certain advantage to Mealy model. Usually it requires less number if states and thereby less hardware to solve any problem. Also, the output is generated one clock cycle earlier. However the disadvantage of such circuit is that the input transient, glitches etc (if any) are directly conveyed to output. Also if we want output transitions to be synchronized while input can change any time Mealy Model is not preferred. In Moore model output remains stable over entire clock period and changes only where there occurs a state change at clock trigger based on input available at that time.





Steps to be followed for as designing:-

1) Choose the model to be implemented (Moore or Mealy)
2) Draw the state transition diagram from word description given
3) Draw the next state table from the given information.
4) The number of states may be reduced by state reduction method if necessary.
5) Assign the state with binary value.
6) Choose the type if FF to be used, determine number of FF to be used based on states & derive the excitation table based on present state (θ_n) to next state (θ_{n+1}) transition and output table (synthesis table).
7) Use K-Map to simplify each input of FF & obtain design equation
8) Draw the logic diagram from the designed equation.

Example 1: A synchronous machine has one bit serial input 'x'. the output 'z' of the machine is to be set high when the input contains the message '100'. Draw the state diagram, derive the transition table (state table), excitation table and design the circuit diagram. [2066 Bhadra]

Using Moore Model

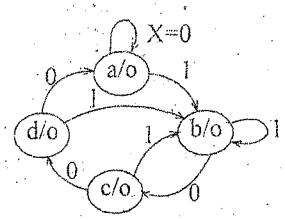


Fig:- State transition diagram

- Circuit is initialized with state 'a'
- If x = 1, then it moves to state 'b' otherwise remains in same state
- If x = 0, then it remains in same state 'a'.
- At state 'b' if x = 1 it remains in same state 'b' but if x = 0 it moves to state 'c'
- At state 'c' if x = 0 then i/p stream is 100 and circuit moves state 'd' with output = 1 at state 'd' but if x = 1 it moves to state 'B' because state 'b' has detected '1'.
- At state 'd', if x = 1 it goes to 1 state 'b' but if x= 0 it moves to initial state 'a' signifying not a single bit has been detected.

Next state table : (Transition table)

Previous	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	b	0	0
c	d	b	0	0
d	a	b	1	1

Assigning pure binary,
Let a = 00, b = 01, c = 10, d = 11

No. of FF required = 2 because only four states are used with two.
Let us select a JK flip-flop.

Synthesis table:-

Present State (θ_n)		Next State				Output y		Excitation table							
A_n	B_n	x = 0		x = 1		X	x = 1	x = 0				x = 1			
A_nB_n	A_nB_n	B_{n+1}	A_{n+1}	B_{n+1}	A_{n+1}			J_B	K_B	J_A	K_A	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	1	0	×	0	×	0	×	1	×
0	1	1	0	0	1	0	0	1	×	×	1	0	×	×	0
1	0	1	1	0	1	0	0	×	0	1	×	×	1	1	×
1	1	0	0	0	1	1	1	×	1	×	1	×	1	×	0

Representing in K-Map for each input of FF

For J_B,

B_n \ A_nx	00	01	11	10
0	0	0	0	1
1	×	×	×	×

$\therefore J_B = A_A\overline{x}$

For K_B,

B_n \ A_nx	00	01	11	10
0	×	×	×	×
1	0	1	1	1

$\therefore K_A = X + A_n$

For J_A,

B_n \ A_nx	00	01	11	10
0	0	1	×	×
1	0	1	×	×

$\therefore J_A = B_n + x$

For K_A,

B_n \ A_nx	00	01	11	10
0	×	×	0	1
1	×	×	0	1

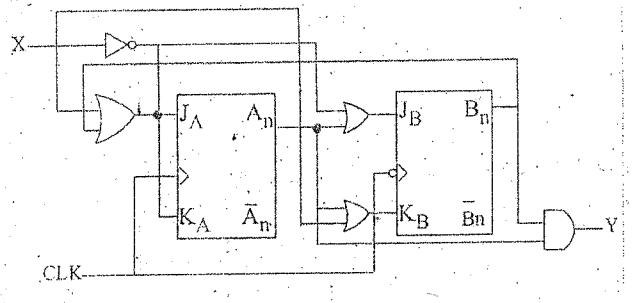
$\therefore K_B = \overline{x}$

For output Y,

B_n \ A_nx	00	01	11	10
0	1	×	×	1
1	0	×	×	×

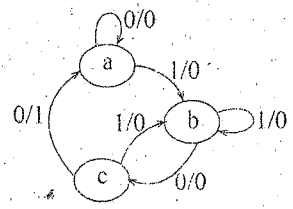
$\therefore X = A_nB_n$

The logic diagram will be as shown,
Step 8: Logic diagram,



Logic diagram for '100' sequences detector

Using Mealy Model:-



State transition Diagram

Assigning Pure binary,
Let a = 00, b = 01, c = 10
No. of FF required = 2. Let us select a JK flip-flop.

Synthesis Table

Present State (Q_n)		Present i/p	Next state		O/p	FF excitation			
B_n	A_n	x	B_{n+1}	A_{n+1}	y	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	×
0	1	0	1	0	0	1	×	×	1
0	1	1	0	1	0	0	×	×	0
1	0	0	0	0	0	×	1	0	×
1	0	1	0	1	1	×	1	1	×

For simplifying the inputs of FF using K-Map,

For J_B,

B_n \ A_nx	00	01	11	10
0	0	0	1	
1	×	×	×	×

$\therefore J_B = A_A\bar{x}$

For K_B,

B_n \ A_nx	00	01	11	10
0	×	×	×	×
1	1	1	×	×

$\therefore K_B = 1$

For J_A,

B_n \ A_nx	00	01	11	10
0	0	1	×	×
1	0	1	×	×

$\therefore J_A = X$

For K_A,

B_n \ A_nx	00	01	11	10
0	×	×	0	1
1	×	×	×	×

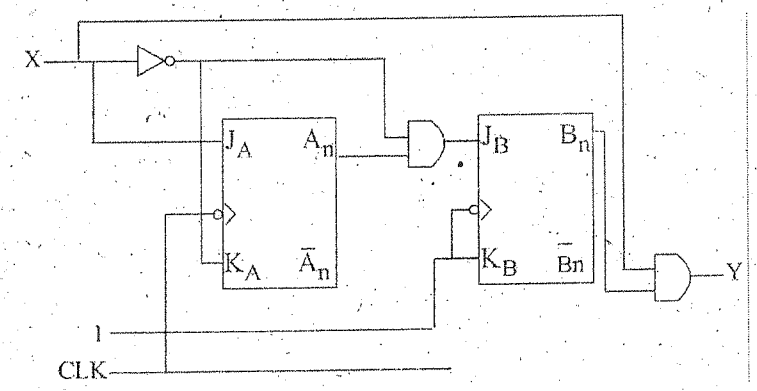
$\therefore K_A = \bar{x}$

For output Y,

	00	01	11	10
0	0	0	0	0
1	0	1	×	×

$\therefore Y = B_n x$

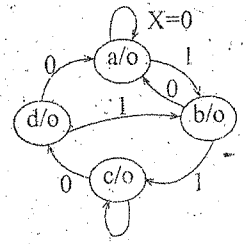
The logic diagram will be as shown,



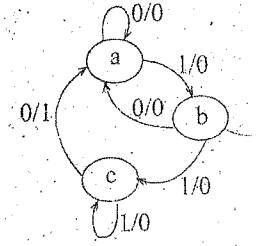
Logic diagram for '100' sequences detector

Example 2: Design a sequential detector that receives binary data stream at its input 'x' and signals when a combination '011' arrives at the i/p by making its output 'y' high which otherwise remains low. Consider data is coming from left i.e. the first bit to be identified is 1, second also 1 and third is 0 from the input sequence.

Solution:



State transition diagram for 011 sequence detector Using Moore Model

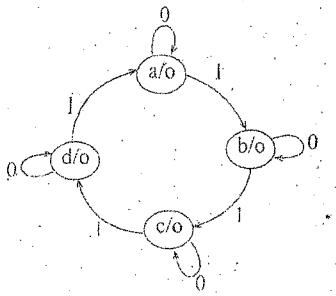


State transition diagram for 011 sequence detector Using Mealy Model

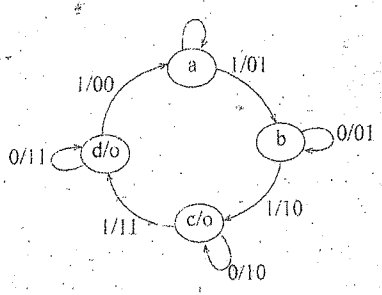
Now you can proceed on your own same as before.

Example 3: Consider a machine that has a single input 'x' and the clock and two outputs A and B. On consecutive rising edge if the clock, the code on A and B changes from 00 to 01 to 10 to 11 and repeats itself when 'x' is ENABLE. If any time 'x' is disabled, this machine is supposed to hold its present state. Design the sequential machine.

Solution:



State transition diagram for Moore Model



State transition diagram for Mealy Model

Using Mealy Model

State table or Transition Table :-

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0 0	0 1
b	b	c	0 1	1 0
c	c	a	1 0	1 1
d	d	d	1 1	0 0

Assigning the pure binary,
Let a = 00, b = 01, c = 10, d = 11

No. of FF required = 2. Let us select a JK flip-flop.

Synthesis table:-

Present State		Present i/p	Next state		Output		FF excitation			
Q_{1n}	Q_{0n}	x	Q_{1n+1}	Q_{0n+1}	A	B	S_1	R_1	S_0	R_0
0	0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	1	0	×	1	0
0	1	0	0	1	0	1	0	×	×	0
0	1	1	1	0	1	0	1	0	0	1
1	0	0	1	0	1	0	×	0	0	×
1	0	1	1	1	1	1	×	0	1	0
1	1	0	1	1	1	1	×	0	×	0
1	1	1	0	0	0	0	0	1	0	1

For S_1 ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	0	1	0
1	×	×	0	×

$\therefore S_1 = Q_{1n} Q_{0n} X$

For R_1 ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	×	0	×
1	0	0	1	0

$\therefore R_1 = Q_{1n} Q_{0n} X$

For S_0 ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	1	0	×
1	0	1	0	×

$\therefore S_0 = \overline{Q}_{1n} X$

For R_0 ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	0	1	0
1	×	0	1	0

$\therefore R_0 = Q_{0n} X$

For Output (A) ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	1	×	×	1
1	0	×	×	×

$\therefore A = Q_{1n}Q_{0n} + Q_{1n}\overline{X} + \overline{Q}_{1n}Q_{0n} X$

For Output (B) ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	1	1	×
1	×	×	×	×

$\therefore B = Q_{0n} X + Q_{0n} X$
$= (Q_{0n} \oplus X)$

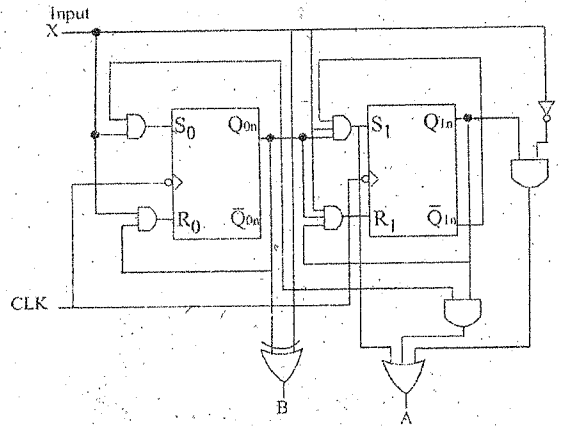
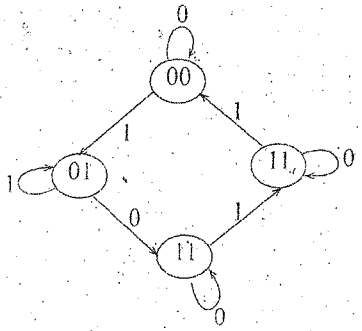


Fig: Logic Circuit

Example 4: Design synchronous sequential circuit for the given state diagram using JK flip-flop. [2069 Ashad]



State table:-

Present State		Present i/p	Next state		FF excitation			
A_n	B_n	x	A_{n+1}	B_{n+1}	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	1	0	1	×	×	1
0	1	1	0	1	0	×	×	0
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	1	×	0
1	1	1	0	0	×	1	×	1

Simplifying the inputs of FF using K-Map,

For J_A,

A_n \ B_nx	00	01	11	10
0	0	0	0	1
1	×	×	×	×

$\therefore J_A = B_n\overline{x}$

For K_A,

A_n \ B_nx	00	01	11	10
0	×	×	×	×
1	0	0	1	0

$\therefore K_A = B_nx$

For J_A,

A_n \ B_nx	00	01	11	10
0	0	1	×	×
1	×	1	×	×

$\therefore J_B = x$

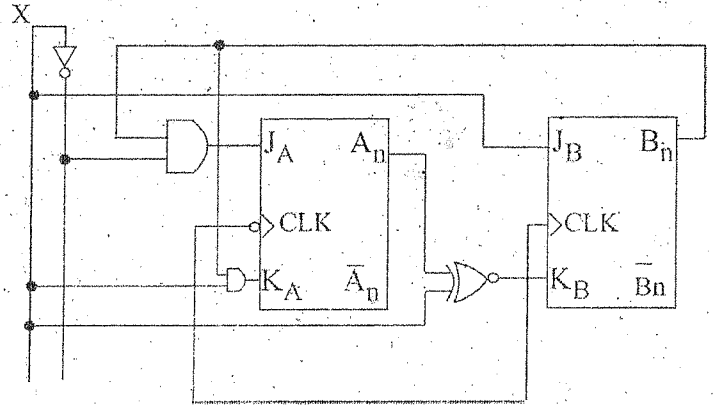
For K_A,

A_n \ B_nx	00	01	11	10
0	×	×	×	1
1	×	×	1	0

$\therefore K_B = A_n{}' + \overline{A}_n{}'\overline{X}'$

$= A_n X$

The logic diagram will be as shown,

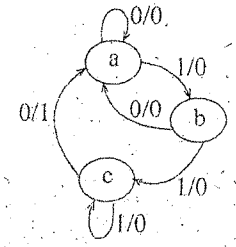


Example 5: Design a synchronous state machine with the following specifications. [2068 Baishak]

a) No. of input 1
b) No. of output 2
c) The output of the machine is set to be high when the data in the input 110 in sequence from MSB (use SR flip-flop)

Solution:

State Diagram



Transition table:-

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	a	c	0	0
c	a	c	1	0

Assigning the pure binary,
Let a – 00, b=01, c=10.
No. of FF required = 2. By question we are using SR flip-flop.

Synthesis table:-

Present State		Present i/p	Next state		Output	FF excitation			
Q_{1n}	Q_{0n}	x	Q_{1n+1}	Q_{0n+1}	Z	S_A	R_A	S_B	R_B
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	0
0	1	0	0	0	0	0	×	0	1
0	1	1	1	0	0	1	0	0	1
1	0	0	0	0	1	0	1	0	×
1	0	1	1	0	0	×	0	0	×

Simplifying the inputs of FF using K-Map,

For S_A ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	0	1	0
1	0	0	×	0

$\therefore S_A = Q_{0n}X$

For R_A ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	×	0	×
1	1	0	×	×

$\therefore R_A = X$

For S_B ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	1	0	0
1	0	0	×	×

$\therefore S_B = \overline{Q}_{1n}\overline{Q}_{1n}X$

For R_B ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	0	1	1
1	×	×	×	×

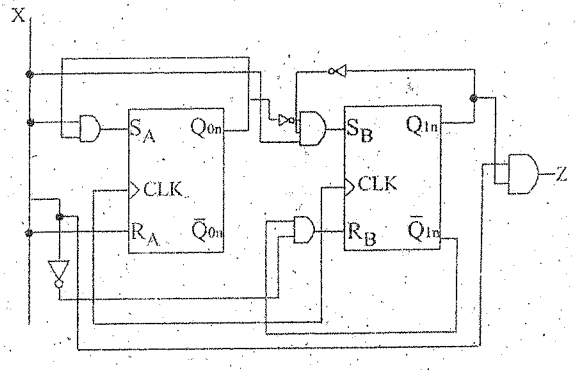
$\therefore R_B = Q_{0n}$

For Output Z ,

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	0	0	0
1	1	0	×	×

$\therefore Z = Q_{1n}\overline{X}$

The logic diagram will be as shown,



Part B:- Asynchronous Sequential Circuit

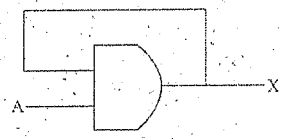
Asynchronous sequential Circuit, also called Event Driven Circuit doesnot have any clock to trigger change of state. State changes are trigerred by change in input signal. In clock driven circuit all the memory elements change their state together. Inspite if all advantages it offers, there are certain limitations with such circuit. The most important being the speed of operation. If the input changes is a manner that warrants change in state, it cannot do that immediately and wait till the next clock trigger comes.

Asynchronous Sequential Circuit is a solution to this however, design of such circuit is very complex & has several constraints to be taken care of, which is not required for synchronous circuit.

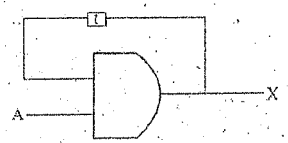
Analysis of Asynchronous Sequential Circuit

In Synchronous System we use clock driven flip-flop which we cannot work here. This is done through feedback similar to basic latch portion of a flip-flop. Before we discuss that let us see how a two input AND gate and two input NAND gate behave with output feedback to one of the input. We shall use K-Map for the analysis.

AND gate:-



(a) AND gate



(b) AND gate with Propagation delay

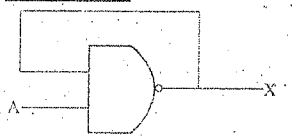
X \ A	0	1
0	0	0
1	0	1

X = x.A

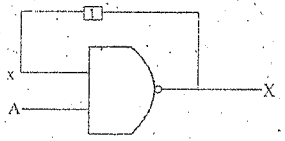
(c) Truth table

- The two input AND gate with output feedback is as one input is shown in figure (a).
- The circuit can be redrawn as shown in fig. (b) that includes the effect of propagation delay of the gate (τ), the finite time after which a gate reacts to its input.
- Fig.(c) shows the truth table of given circuit and encircled states indicate stable condition of the circuit.
- For example, if A=0 and by any reason previous output (that is currently feedback) x=0 then x = n.A = 0.0 = 0.
- After time t = τ , x takes the values of X, i.e. 0 and because of that output x does not change.
- Hence x = 0, A = 0 represents a stable states and is encircled.
- In the similar manner, x = 0, A = 1 position and x = 1, A = 1 position are also stable as is each care X = x and no change in output is necessary.
- But at x = 1, A = 0, X becomes 0 and after time τ, x becomes 0 i.e. we move by one position in K-map to x = 0, A = 0 position.
- We make an important observation from this discussion which is universally true for asynchronous sequential circuit. For any state if x = X then the circuit is stable & if x ≠X it is unstable.

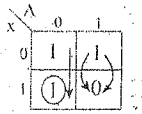
NAND gate:-



(a) NAND gate



(b) NAND gate with Propagation delay

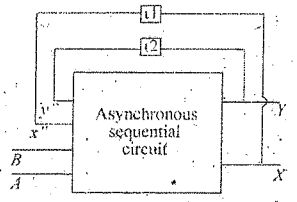


(c) Truth table

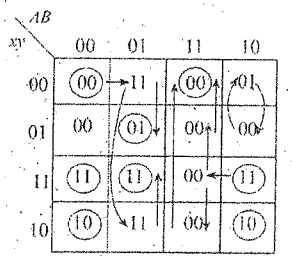
We extend the above observation to feedback NAND circuit as shown in above figure. Fig. (a) and arrive at the truth table in fig (c). It is interesting to note that for A = 1, there is no stable state and x = X' for both x = 0 and x = 1. Thus there is oscillation between x = 0, A = 1 and x = 1, A = 1 state.

Problems with Asynchronous Sequential Circuit [2068 Shrawan]

Asynchronous Circuit responds to all the transient values and problems like oscillation, critical race, hazards can cause major problem unless they are addressed at design stage. To explain these problems we take help of Truth Table shown in where the circuit has two external inputs *A, B* and two outputs *X, Y*. Both the outputs are fed back to the input side in the form of x and y but with different propagation delays. Thus x, y cannot change simultaneously but with time delays τ1 and τ2 respectively and we can write x = X (t - τ) and y = Y (t = τ).



(a) Block Diagram



(b) Truth Table

Oscillation:-

Consider, the stable state xyAB = 0000, where x = X and y = Y. If input AB change from 00 to 10, the circuit goes to xyAB = 0010 state and then output XY = 01. This is a transient state because xy ≠XY. After time τ2, y take the value of Y = 1 and the circuit goes to xyAB = 0110 where output XY = 00. This again is a transient state and after another propagation delay of τ2, the circuit goes to xyAB=0010. Thus the circuit oscillates between state 0010 and 0110 and the output Y oscillates between 0 and 1 with a time gap τ2. In asynchronous sequential circuits for any given input, transitions between two unstable states like these are to be avoided to remove oscillation.

Critical Race

Race condition is another major problem in asynchronous sequential circuit. This occurs when an input change tries to modify more than one output. In the truth table of consider the stable state xyAB=0000. Now, if AB changes to 01 the circuit moves to xyAB=0001 where XY=11, Now depending which of τ1 and τ2 is lower, xy moves from 00 to either 01 or 10. If τ1 is lower, x changes earlier and the circuit goes to xyAB=1001 which is a unstable state with output XY=11 and xy ≠XY. The circuit next moves to state xyAB=1101 which is a stable state and final output XY=11. If τ2 is lower, y changes earlier and the circuit goes to xyAB=0101, a stable state and the final output is 01. Thus, depending on propagation delays is feedback path, the circuit settles at two different states generating two different set of outputs. Such a situation is called critical race condition and is to be avoided in asynchronous sequential circuit.

Hazards

Static and dynamic hazards causes malfunctioning of asynchronous sequential circuit. Situations like Y = A +A' or Y = AA' are to be avoided for any input output combination with the help of hazard covers in the table. In circuit with feedback even when these hazards are adequately covered there can be another problem called essential hazard. This output when change in input does not reach one part of the circuit while from other part one output fed back to the input side becomes available. Essential hazard is avoided by adding delay, may be in the form of additional gates that does not change the logic level, in the feedback path.

CHAPTER 10

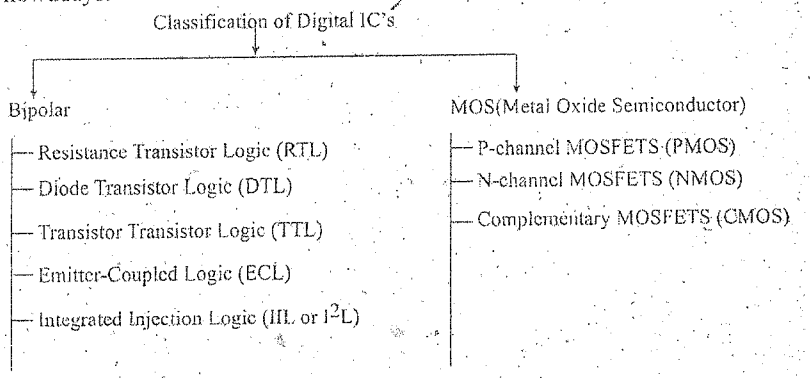
Digital Integrate Circuit

Introduction:-

A digital integrated circuits or IC is a silicon semi-conductor crystal, called a chip, containing the electronic components (resistors, diodes, transistors, capacitor etc) for constructing logic gates. The various gates are interconnected inside the chip to form the required circuit.

Digital integrated circuits operate with binary signals and are invariably constructed with ICs. In other words, they perform logic functions with the help of binary numbers 0 & 1; such as logic gates, flip-flop, counters, shift register etc. Digital ICs are most popular in realization of electronic systems in the areas of instrumentation, communication, controls & computers.

In the view of low cost & excellent performance of digital ICs in silicon monolithic form, ICs based on monolithic technology are used nowadays.



Characteristic of Digital ICs

Various characteristics of Digital ICs are as follows:-

1) **Operating Speed :-** Speed of logic gate depends upon the time that elapses between the application of a signal to an input terminal and the resulting change in logical state at output terminal. It takes into consideration the transition time and propagation delays. Higher operating speed is usually the main requirement of digital ICs.

2) **Fan-In:-** The fan-in of a logic gate is defined as the number of inputs that it can handle properly.

3) **Fan-out:-** The fan-out is defined as the maximum number of standard logic inputs that an output can drive reliably. For example, a logic gate that is specified to have a fan-out of 8 can drive. 8 standard logic inputs. If this number exceeds, the output logic level voltage cannot be guaranteed.

[Note:- Fan-out is sometimes called the loading factor.]

4) **Power dissipation:-** This is the amount of power dissipated in an IC. It is determined by current Icc that it draws from the Vcc supply and equals Vcc Icc where Icc is average value of Icc (0) and Icc (1).
This power is specified is mW-Lower power dissipation is desirable feature for any IC.

5) **Power Supply Requirement:-** Every IC requires a certain amount of electrical power to operate. The power is supplied by one or more power supply voltage connected to the power pin (or pins) on the chip. Low power consumption is desirable feature in any digital IC.

6) **Noise immunity:-** Stray electric & magnetic fields can induce voltages on the connecting between logic circuits. There unwanted spurious signals are known as noise & can sometimes lead to false trigerring of logic levels in the circuit.
The noise immunity of a logic circuit refers to the circuit ability to tolerate noise voltages on its input. A quantitative measure of noise immunity is called noise margin.
Higher the noise margin, better the logic circuit.

7) **Operating Temperature Range:-** Digital ICs should be capable of operating for temperature ranging from 0° to 70^{0} C for consumers & from -55^{0} C to $+125^{0}$ C for military applications.

10.1 Switching Circuits:-

The semiconductor device used in digital integrated circuits such as diodes, bipolar junction transistors (BJTs), metal-oxide-semiconductor field effect transistors (MOSFETs) can be used as switching circuits.

(1) The Semiconductor Diode as Switch:-

The symbol for a semiconductor diode is as shown in fig 10.1 (a). The diode behaves like a one-way switch. That is, it will allow an electric current in one direction but not the other. Let's consider conventional current flow rather than electron flow.

Fig. 10.1(b) shown the direction of current through a diode-this is forward direction. When conducting current, a silicon diode will have a nominal voltage of 0.7 V across its terminal. In this condition diode is said to be forward-biased.

It is not possible to pass current through the diode in other direction the reverse direction. When reverse biased, the diode will act as an open switch as illustrated in fig. 10.1(c).

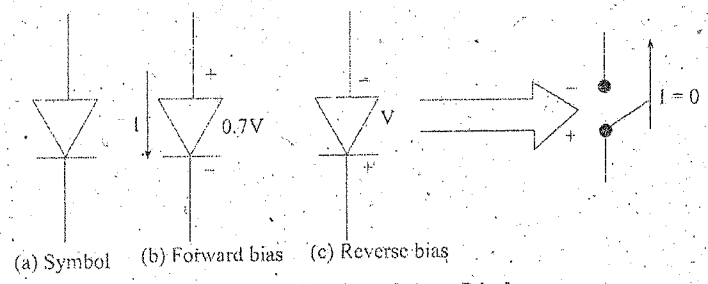


Fig:-10.1 Semiconductor Diode

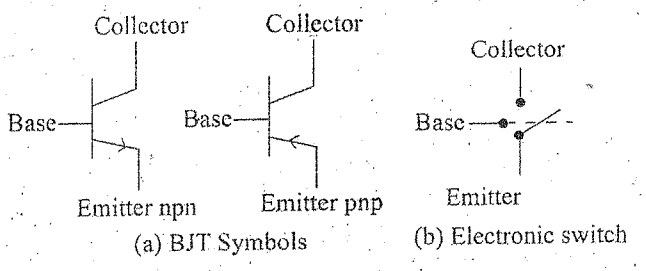
(2) **BJTs as Switch :-** **[2068 Shrwan]**

The bipolar junction transistor (BJT) is available in two polarities (npn & pnp) as shown by symbols in Fig. 10.2 (a). The BJT terminals are named collector, emitter and base. In Fig. 10.2(b), the BJT behaves as an electronic switch. The switch is activated by applying a voltage between base and emitter. It's working is explained in two points,

(i) The voltage between base and emitter is zero. The switch is open, and no current is allowed between collector and emitter. The transistor is off.

(ii) A voltage is applied between base and emitter. The switch is closed and a current is allowed between collector and emitter. The transistor is on. The voltage between emitter & collector (across a closed switch) is zero.

The current through the npn transistor must be from collector to emitter as shown in fig. 10.2 (c). For pnp, current must be from emitter to collector.



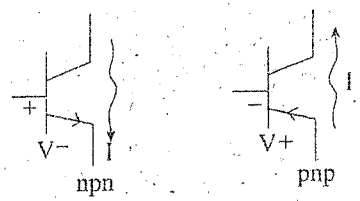


Fig:- 10.2

(3) **MOSFETs as Switch:-**

MOSFETs are available in two polarities (n-channel & p-channel) as shown by the symbols in fig. 10.3 (a). The MOSFET operates in two modes – depletion and enhancement. The MOSFET terminals are named gate, source, drain & body as indicated. The MOSFET also behaves as the electronic switch in fig. 10.3 (b). The switch is in fig. 10.3 (b). The switch is activated by applying a voltage between gate and source. It's working is explained in two points.

(i) The voltage between gate and source is zero. The switch is open, and no current is allowed between source and drain. The transistor is off.

(ii) The voltage is applied between gate & source. The switch is closed, and a current is allowed between source and drain. The transistor is on. The voltage between source and drain (across a closed switch) is zero.

The current through n-channel must be from drain to source as shown in fig. 10.3(c). For p-channel, it must be from source to drain.

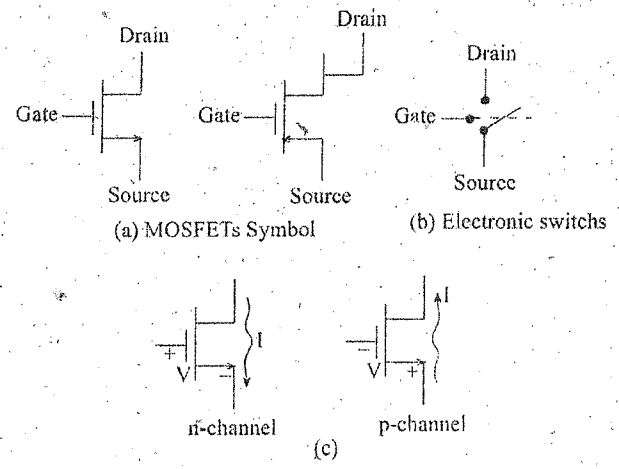


Fig:- 10.3

(4) **Complementary Metal-Oxide Semiconductor (CMOS) as Switch:-**

ICs constructed entirely with n-channel MOSFETs are called NMOS ICs. ICs constructed entirely with p-channel MOSFETs are called PMOS ICS. Since n-channel & p-channel MOSFETs are considered complementary devices, there ICs are referred to as CMOS ICs.

A CMOS inverter is as shown in fig. 10.4. It's working is explained in two points.

(i) $V_1 = 0$Vdc. Q_n is off & Q_p is on. $V_2 = +5$Vdc

(ii) $V_1 = +5$Vdc. Q_n is on & Q_p is off. $V_2 = 0$Vdc

In steady state (while not switching), one of the transistor is always off. As a result, I=0mA. When switching between states, both transistors are on for a very short time because of rise or fall time of V_1. This is the only time the current I is non-zero. This is the reason CMOS is used in applications where dc power supply currents must be held minimum watches, pocket calculator etc.

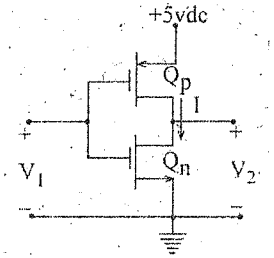


Fig.:- 10.4 CMOS Inverter

10.2 TTL NAND gate:- [2069 Ashad] [2068 Shrawan]

The basic TTL logic circuit is the NAND gate. The circuit diagram is shown in Fig.

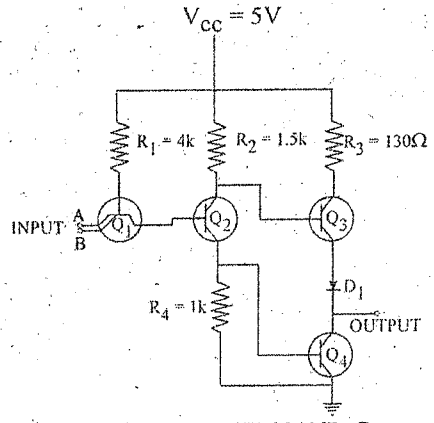


Fig:- Basic TTL NAND Gate

It can be seen that transistor Q_1 has two emitters so it has two emitter-base junctions which can be used to turn transistor Q_1 on. Up to eight emitters can be used in a multiple-emitter input transistor for an eight-input NAND gate. Other special feature of this circuit is totem-pole arrangement made by transistors Q_3 and Q_4 on the output side of the circuit.

Now let us have a look on the circuit operation. In the circuit diagram resistance R_1 and transistor Q_1 act like a 2-input AND gate. Transistor Q_2 acts like an inverter, which inverts the output of transistor Q_1 and hence the circuit operates like a NAND gate. Transistors Q_3 and Q_4 form a totem-pole connections at the output stage, either Q_3 or Q_4 is on at a time. When Q_3 is on, the output is high; when Q_4 is on, the output is low.

Now consider a case when both inputs A and B, are high. Emitter diodes of Q_1 stop conducting and collector diode gets forward biased. With this base of transistor Q_2 becomes high and it starts conducting. So a potential drop occurs across resistance, R_4, and base of transistor Q_4 becomes high. So transistor Q_4 becomes saturated and gives low output.

Consider the other case also, when both inputs or either is low, then base of Q_1 becomes approximately at 0.7 V which reduces the base voltage of Q_2 at almost zero volt, therefore, transistor Q_2 is cut off. Almost no current flows through resistances R_2 and R_4 and, therefore, base voltage of Q_4 becomes zero and Q_4 is cut off. Base voltage of Q_3 becomes high and since Q_3 acts as an emitter follower, it produces high output.

The function of diode D_1 is to prevent even the slight conduction of Q_3, when the output is low. Voltage drop across the diode D_1 keeps the emitter diode of Q_3 reverse biased and so only Q_4 conducts during the low output condition.

Totem-pole transistors are used to get low output impedance. During high output state Q_3 conducts and provides output impedance of 70Ω. During low output state, Q_4 gets saturated and provides output impedance of approximately 12Ω. So in both states, output impedance is low and it ensures that output voltage changes its value from one state to another quickly because stray capacitance is rapidly charged or discharged through low output impedance.

Characteristics of 7400 TTL:- **[2068 Baishak]**

(1) **Propagation Delay Time:-** The propagation delay time is the time it takes for the output of a gate to change after inputs have changed. The propagation delay time of a TTL gate is approximately 10 nanoseconds.

(2) **Power dissipation:-** A standard TTL gate has a power dissipation of about 10 milliwatts(mw).It may vary from this value because of signal levels, tolerance etc, but on average it is 10mw per gate.

(3) **Temperature Range:-** 54 series operates properly in ambient temperature ranging from -55^0 c to $+125^0$c while 74 series operates only is range from 0^0C to 70^0C.

(4) **High Voltage Range:-** Normal supply voltage (Vcc) for both 54 & 74 series is 5V. 54 series operates reliably over the range of 4.5 to 5.5 V but 74 series operates reliably over the range of 4.75 to 5.25V. [2068 Chaitra]

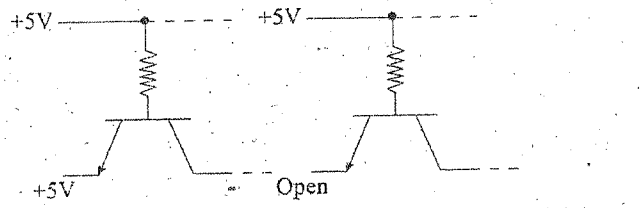
(5) **Device Number:-** Different TTL devices can be designed by varying the number of multiple emitter inputs & totem pole outputs.

10.3 TTL Parameter:-

The various TTL parameters are,

(1) **Floating Inputs:-** When a TTL input is high (ideally +5V), the emitter current is approximately zero (Fig 10.4 (a)). Where a TTL input is floating (unconnected fig 10.4 (b)), no emitter current is possible because of the open circuit. Therefore a floating TTL input is equivalent to a high output.

There is a disadvantage of floating inputs. When we leave an input open, it acts as a small antenna. Therefore it will pick up stray electromagnetic noise voltage.



(a) High input (b) Open is equivalent to high input

Fig:-10.4

(2) **Sourcing & Sinking:-** When a standard TTL output is low (Fig 10.5 (a)), an emitter current of approximately 1.6 mA exits in direction shown. The conventional flow is from the emitter of Q_1 to the collector of Q_4, Because it is saturated, Q_4 acts as a current sink.

When a standard TTL output is high (Fig 10.5 (b)), a reserve emitter current of 40μA exist in direction as shown. Conventional current flows out of Q_3 to emitter of Q_1. In this case Q_3 is acting as a source.

$\therefore I_{IL(max)} = -1.6$ mA

& $I_{IH(max)} = 40$ μA

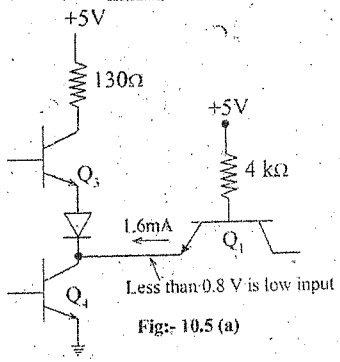


Fig:- 10.5 (a)

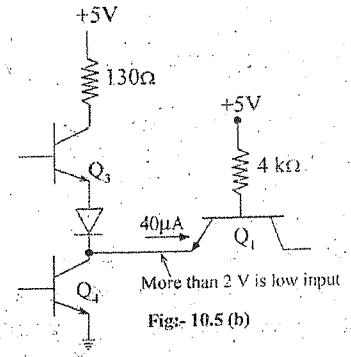


Fig:- 10.5 (b)

(3) **Noise Immunity:-** In a worst case, there is a difference of 0.4V between the driver output voltage & required load input voltage. For instance, the worst-case low values are,

$V_{OL,\,max} = 0.4$ V driver output
$V_{IL,\,max} = 0.8$ V load input

Similarly, the worst-cast high values are,

$V_{OH,\,max} = 2.4$ V driver output
$V_{IH,\,max} = 2$V load input

In either case, the difference is 0.4V. This difference is called noise immunity.

(4) **Standard loading:-** A TTL device can source current or sink current. Data sheets of standard TTL devices indicate that any 7400 series device can sink up to 16mA, designated

$$I_{OL,\,max} = 16\text{mA}$$

And source up to 400μA, designated

$$I_{OH,\,max} = -400\mu\text{A}$$

But the worst case TTL currents are,

$$I_{IH,\,max} = -1.6\text{mA}$$
$$I_{IH,\,max} = 40\mu\text{A}$$

Since the maximum output currents are 10 times larger than the input currents, we can connect up in 10 TTL emitters to any TTL output.

(5) **Worst case Input & Output Voltage:-** The worst case input voltages for TTL gate is,

$$V_{IL(max)} = 0.8\text{ V}, \quad V_{IL(min)} = 2\text{V}$$

The worst case output voltages are,

$$V_{OL(max)} = 0.4\text{ V}, \quad V_{OL(min)} = 2.4\text{V}$$

10.4 TTL Overview :-

This overview will give us an idea of variety of gate, & circuits found is TTL family.

(1) **NAND gate:-** The NAND gate is the backbone of 7400 series. The basic TTL logic circuit is the NAND gate. The circuit diagram and working principle of TTL NAND gate has already been discussed.

(2) **NOR Gate:-** **[2068 Baishak]**

Fig. shown a 2-input NOR gate. Here Q_5 and Q_6 have been added to basic NAND –gate design. Since Q_2 and Q_6 are in

parallel, we get the OR function, which is followed by inversion to get the NOR functions.

When A and B are both low, the bases of Q_1 and Q_5 are pulled low; this cuts off Q_2 and Q_6. Then Q_1 acts as an emitter-follower, and we get a high output.

If A or B is high. Q_1 and Q_5 are cut off, foreing Q_2 or Q_6 to turn on. When this happens, Q_4 saturates and pulls the output down to a low voltage.

With more transistors, a manufacturer can produce 3-and 4-input NOR gates. (Note: A TTL 8-input NOR gate is not available)

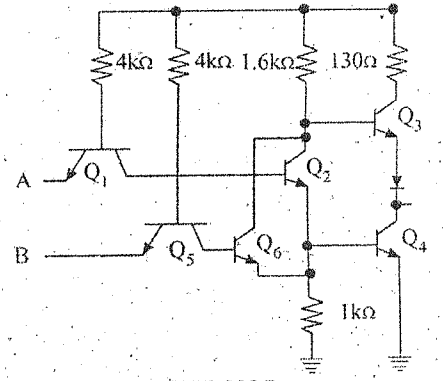


Fig: TTL NOR gate

(3) **TTL inverter (NOT gate):-** Here the combination of Q_3 and Q_4 transistors forms the output circuit often reffered to as totem-pole arrangements.

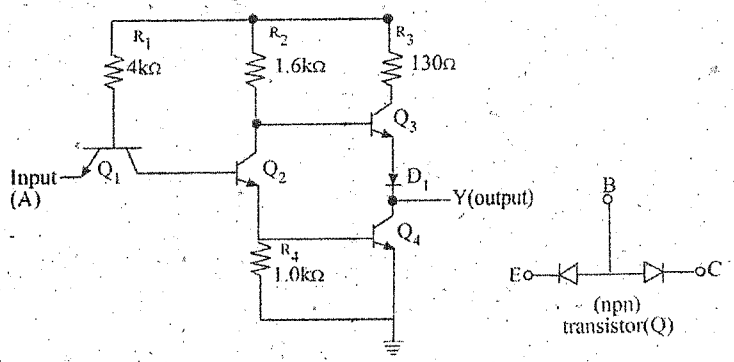


Fig:-TTL Inverter

(1) When the input is HIGH, the loase-emither junction of Q_1 is Reverred Bised and the base collector is forward biased.

- This condition permits current through R1 and the base collector function of Q_1 into the base of Q_2, thus driving Q_2 into saturation. (Q_2
- As a result, Q_4 is turned 'ON' by Q_2, and its collector voltage, which is output is near ground potential.
- At the same time, the collector of Q_2 is at LOW (ground) which keep Q_2 'OFF'.

(2) When the input is 'low', the base-emitter junction is forward biased and the base-collector junction is revered biased.

- There is current through R_1 and the base-emitter junction of Q_1 to the 'Low' input. A low provides path to ground for current.
- There is no current on the base of Q_2 so 'OFF'.
- Thus collector of Q_2 is HIGH, causing transistor Q_2 turned 'ON', meanwhile transistor Q_4 is turned off.
- Hence the output is 'HIGH'.
- Diode 'D_1' provides on additional $V_{BE}(0\neq v)$ equivalent drop in series with base-emitter of Q_6 to ensure it is turned OFF when Q_2 is 'ON'.

(4) **AND and OR gate:-** To produce the AND function another inverting stage is inverted in the basic NAND-gate design. The extra inversion converts the NAND gate to an AND gate.

Similarly another inverting stage can be inserted in the NOR gate. This converts NOR gate to an OR gate.

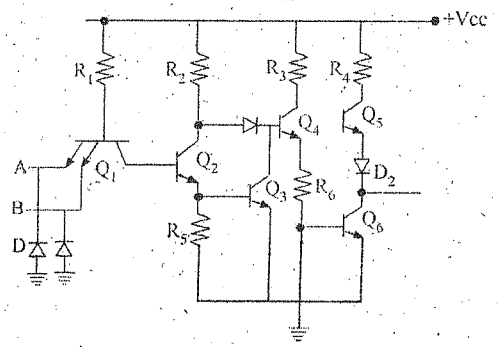


Fig:-TTL-AND gate

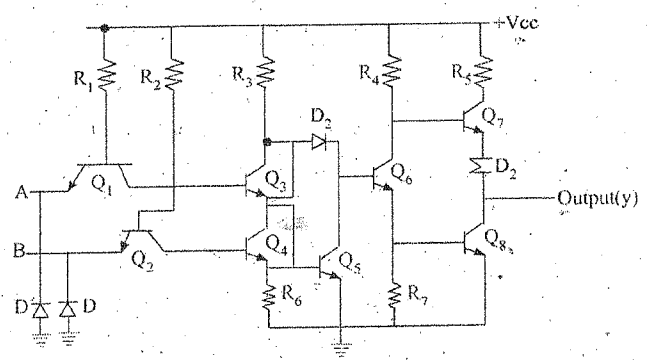
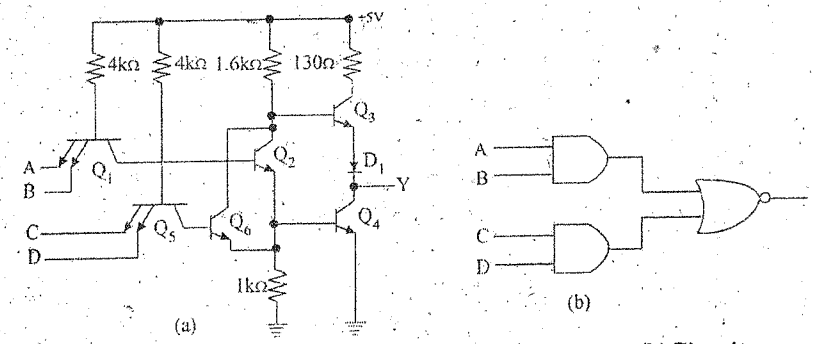


Fig:-TTL-OR gate

(5) **AND-OR-INVERT gates:-** Here Q_1, Q_2, Q_3 & Q_4 form the basic 2-input NAND gate to AND-OR-INVERT.

Both Q_1 and Q_5 acts as 2-input AND gates. Q_2 & Q_6 produces O King and inversion. Because of this circuit is logically equivalent to Fig. (b)



(a) AND-OR-INVERT schematic diagram, (b)Circuit

10.5 Open Collector Gates:-

Instead of a totem-pole output, some TTL devices have an open-collector output. Fig. below shows a 2-input NAND gate with an open-collector output. As the collector of Q_4 is open, a gate like this will not work properly until we connect an external pull up resistor.

Hence for this type of gate, an external pull up resistor is required for proper operation.

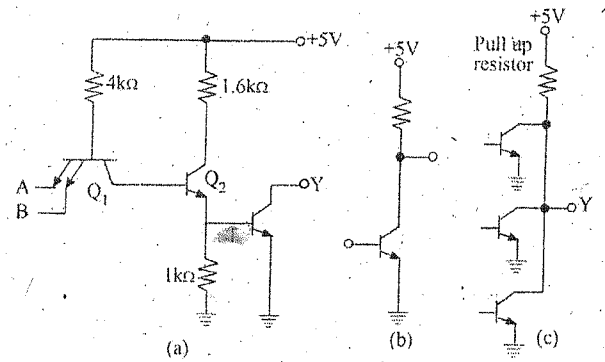
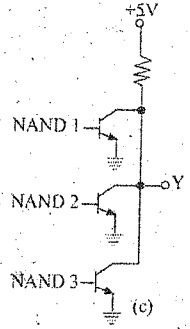


Fig: open-collector TTL: (a) Circuit, (b) Pull-up resistor (c) open-collector outputs connected to a pull-up resistor.

Advantages:

It has the advantages of combining three devices without using a final OR gate (or AND gate). This is done through direct connecting of the '3' outputs to the lower end of common pull-up resistor.



Disadvantage:-

The big disadvantage of open-collector gates is their slow switching speed. Because the pull-up resistance is a few kiloohms, which results in a relatively long time constant when it is multiplied by the stray output capacitance ($\tau = RC$).

10.6 Three-State TTL Devices:-

Three state or Tristate logic refers to the condition that the output exists in three states i.e. low, high & high impedance states. This logic has another input called control input. It utilizes the advantages of high speed of operation of the totem-pole configuration and wire ANDing of the open-collector configuration.

Tristate Logic: In normal logic circuits there are two states of output, either low or high. But in complex digital systems like microcomputers and microprocessors, a number of gate outputs may be required to be connected to a common line which is referred to as a

bus which, in turn, may be required to drive a number of gate inputs. When a number of gate output are connected to the bus, some difficulties are encountered. These are:

1) Totem-pole outputs cannot be connected together due to very large current drain from the supply and consequent heating of ICs which may get damaged.
2) Open collector outputs can be connected together with a common collector resistor connected externally. This causes the problems of loading and operation speed.

To overcome these difficulties, special circuits have been developed in which there is one more state of output, referred to as the third state or high impedance (Hi-Z) state, in addition to low and high states. Such circuits are called Tristate or three state logic.

The tristate circuit utilizes the high speed operation of the totem-pole arrangement while permitting outputs to be wired-ANDed (connected together). The Hi-Z state is a condition in which both transistors in the totem-pole arrangement are turned off so that the output terminal is a high impedance to ground and to V_{CC}. In other words, the output is an open or floating terminal that is neither a low nor a high. In practice the output terminal is not an exact open circuit, but has a high resistance of several MΩ or more relative to ground and V_{CC}.

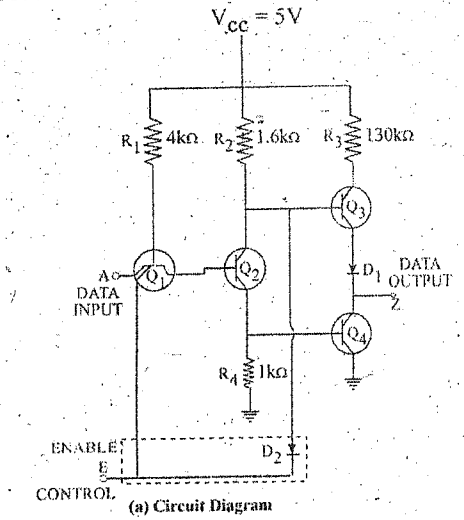


Fig:- TTL Inverter

The tristate operation is achieved by modifying the basic totem-pole circuit. Fig shows the circuits for a tristate INVERTER where the portion enclosed in a dotted rectangle has been added to a basic circuit. The circuit has two inputs : A is the normal logic input and E is an ENABLE input capable of producing the Hi-Z state.

When E = 0, the circuit goes into its High impedance state regardless of the state of logic input A. The low at E forward-baises the emitter-base junction of transistor Q_1 and shunts the resistor R_1 current away from transistor Q_2 so that Q_2 so that Q_2 turns off, which turns transistor Q_4 off. The low at E also forward biases diode D_2 to shunt current away from the base of transistor Q_3, so that Q_3 also turns off. With both totem-pole transistors in the off-state, the output terminal is essentially an open circuit. This is illustrated in Table (truth table for TTL inverter).

10.7. External Drive for TTL loads:-

To drive a TTL load with an external source, we need to satisfy the TTL input requirements for voltage & current which is, input voltage must be between 0 to 0.8V with a current of approximately 1.6mA in low state. In high state, the voltage must be between 2 to 5V with a current of approximately 40μA.

When there conditions are fulfilled we use a following technique to drive a TTL load.

(1) Switch Drive:- It is preferred method for driving a TTL input from a switch. When the switch is open the input is pulled up to +5V. in the worst case, only 40μA of input current exists. Therefore the voltage appearing at the input is slightly less than supply voltage because of the voltage drop across the pull up resistor.

$$V_I = 5V - (40\mu A)(1\ KA) = 4.96V$$

This is well above the minimum requirement of 2V.

When the switch is closed, the input is pulled down to ground. In the worst case, the input current is 1.6mA. this sink current creates no problem because it flows through the closed switch to ground.

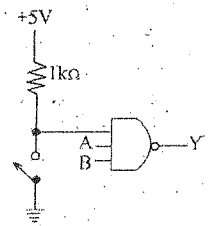


Fig: Switch Drive for TTL input

(2) Size of Pull Up Resistance:- A pull up resistance of 1KA is nominal . the current drain with a closed switch is

$$I = \frac{5V}{1kA} = 5mA$$

The smaller the pull-up resistance, the larger the current drain. But too much current drain becomes a problem for power supply. So we have to use a resistance that is large enough to keep current drain to tolerable level.

But again, too large pull-up resistance between 1-10kA are typical. They result is current drain & time constants that are acceptable in most applications.

(3) Transistor drive:- A transistor switch can be used to control the state of TTL input.

When V_1 is low, the transistor is off & is equivalent to open switch. Then the TTL input is pulled to +5V through a resistance of 1KΩ.

When V_1 is high, the transistor is on 8 is equivalent to closed switch. in this case, it easily sinks 1.6mA of input current.

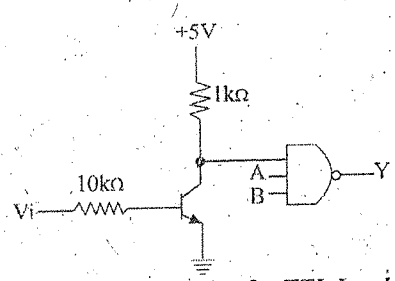


Fig: Transistor drive for TTL Load

(4) Operational Amplifier Drive:- The operational amplifier output can be used to drive TTL load.

The output of 0A ideally swings from +15 to −15V. The positive swing closes the transistor switch, producing a TTL input of approximately 0V.

The negative swing drives the transistor into cut off, producing the TTL input of +5V.

The diode in the base circuit protects the base against excessive reverse voltage.

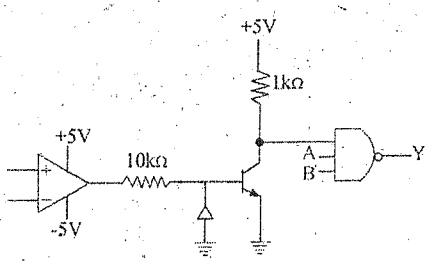


Fig:- Up-amp & transistor drive for TTL input

10.8 TTL Drive External loads:-

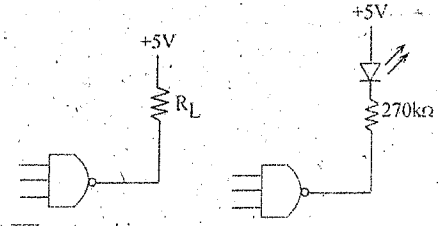
Since TTL can sink up to 16mA, we can use a TTL driver to control an external load such as a relay, LED etc.

When the TTL output is high, there is no load current. But when the TTL output is low, the lower end of R_1 is ideally grounded. This sets up a load current of approximately,

$$I_L = \frac{5V}{R_L}$$

Since standard TTL can sink a maximum of 16mA, the load resistance is limited to a minimum of,

$$R_L = \frac{5V}{16mA} = 312\Omega$$



10.9 74100 CMOS Circuits:-

Complementary metal oxide semiconductor devices are chips in which both P-channel and N-channel enhancement MOSFETs are connected in a push-pull arrangement. Main advantages of CMOS over TTL devices are given below.

1. These are simple and cheaper in fabrication.
2. These are small in size and consume very little power.
3. Fabrications of MOS ICs is not so complex as TTL ICs.

Normally MOS ICs do not use IC resistance elements,
So they occupy much less space on a chip. That is why MOS ICs are used in LSI and VLSI (very large scale integration).

Main disadvantage of MOS ICs is that it has low operating speed in comparison to TTL ICs.

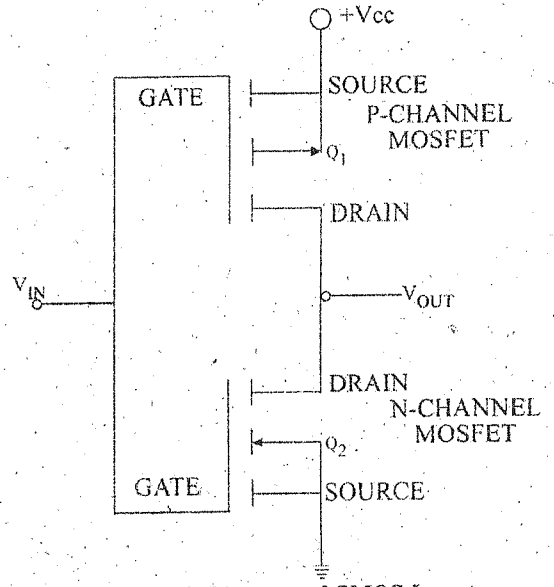


Fig:- Circuit Diagram of CMOS Inverter

1) **CMOS Inverter:-** The circuit diagram of the basic CMOS inverter is shown in Fig. In this circuit, CMOS inverter has two MOSFETs connected in series so that source of P-channel device Q_1 is connected to a +ve voltage supply + V_{DD} and the source of N-channel device Q_2 is connected to the ground. Gates of both the devices are connected as common input. Drain terminals of both the devices are connected together as a common output.

+ V_{DD} volt represents logic 1 and 0 volt represents logic 0. Now consider the first case when input is kept at low level i.e., at 0 volt, then gate of MOSFET Q_1 is at –ve potential relative to the source. So Q_1 will be on with its resistance R_{on} = 1 kΩ, while gate of Q_2 will be at zero potential relative to its source. So Q_2 will be off with its resistance $R_{off} = 10^{10}\Omega$. Both of these resistances act like a potential divider and output of this will be approximately + V_{DD} volts.

In other case when input is kept at high level i.e. at + V_{DD} volts, then gate of MOSFET Q_1 is at zero potential relative to its source so Q_1 will be off with its resistance $R_{off} = 10^{10}\ \Omega$ while gate of Q_2 will be at +ve potential relative to its source so Q_2 will be on with its resistance $R_{on} = 1k\Omega$. In this case output will be approximately 0 volt. Thus circuit shown in Fig. acts as an inverter.

2) **CMOS NOR Gate:-** Circuit diagram of two-input NOR gate is shown in Fig. In this circuit, one P-channel MOSFET is added in series in series and one N-channel MOSFET is added in parallel further to the basic inverter.

In first case when both inputs are at low level, then both P-channel MOSFETs Q_1 and Q_2 will be on and both N-channel MOSFETs Q_3 and Q_4 will be off which will give output of high level.

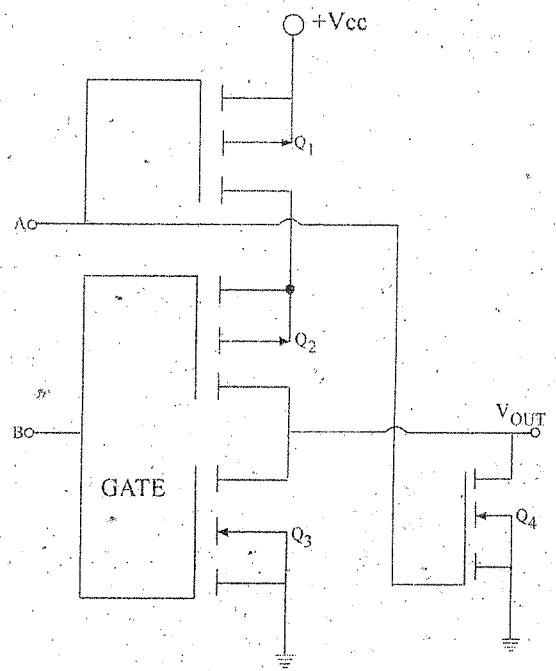


Fig:- Circuit Diagram of NOR Gate

In other case, when either or both the inputs are at high level, then P-channel MOSFETs and N-channel MOSFETs corresponding to high level input will be off and on respectively, thus giving output of low level.

Thus circuit shown in fig. acts as a NOR gate.

3) **CMOS NAND Gate:-** Circuit diagram of two input NAND gate is shown in Fig. In this circuit one P-channel MOSFET is added in parallel and one N-channel MOSFET is added in series to the basic inverter.

In the first case, when both inputs are at high level then both P-channel MOSFETs will be off and both N-channel MOSFETs will be on and give low-level output.

In other case when either or both of the inputs are at low level the P-channel MOSFET and N-channel MOSFET corresponding to low level input will be on and off respectively giving output of high level.

Thus circuit shown in fig. acts as a NAND gate.

CMOS AND and CMOS OR gates can be formed by combining NAND and NOR gates with inverters.

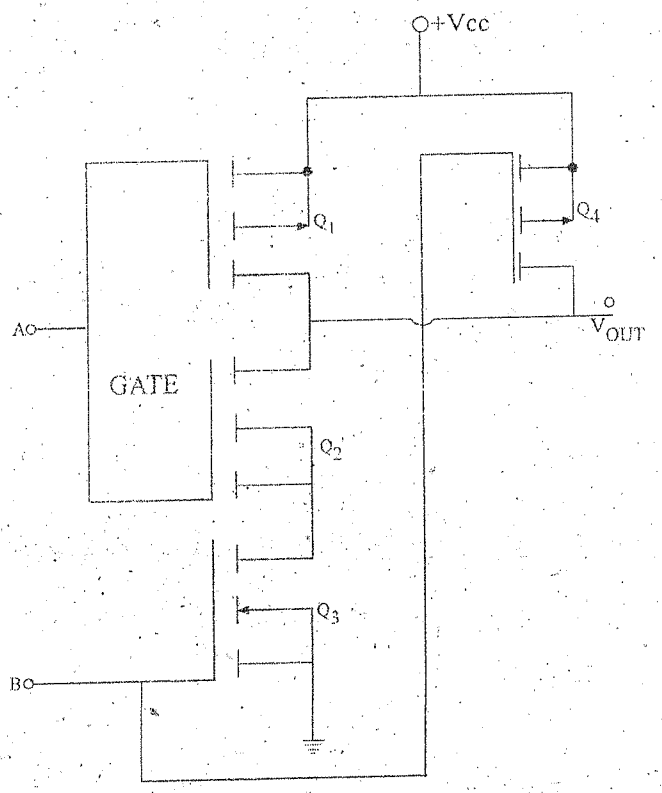


Fig:- Circuit Diagram of NAND Gate

10.10 CMOS Characteristics:- [2069 Ashad]

Same of the operating and performance characteristics of CMOS series are.

1) **Supply voltage:-** The 4000 series and 74C series operate with 3 to 15V power supply and 74HC, 74HCT series operated with 2 to 6V power supply.
2) **Temperature Range:-** 74c series operates properly in ambient temperature ranging from -40^0C to $+85^0$C and this is adequate for most commercial application. But 54C series operates in temperature ranging from 55^0C to $+125^0$C.
3) **Power Dissipation:-** when a CMOS logic circuit is in a static stable, power dissipation in this series is extremely low since in any of the output state supply. So power dissipation of CMOS series is only 2.5nW per gate. This is the reason why CMOS is used so widely.
4) **Propagation delay time:-** A standard CMOS gate has a propagation delay time of approximately 25 to 100 ns depending on the operating voltage & other factors.
5) **Sinking & sourcing current:-** The worst case input currents for CMOS devices are

$$I_{IL(max)} = -1\mu A \text{ \& } I_{IH(max)} = 1\mu A$$

The worst case output currents for CMOS devices are,

$$I_{OL(max)} = 10\mu A \text{ \& } I_{OH(max)} = -10\mu A$$

6) **Noise Margin :-** In CMOS series, the noise margin is typically about 4.5% of the supply voltage V_{DD}.

10.11 TTL-to-CMOS Interface:- [2068 Chaitra]

Interface refers to the way a driving device is connected to a loading device.

For TTL to-CMOS interface, TTL is a driving device and CMOS is a loading device.

We need to pull up TTL high with pull up resistor at CMOS input

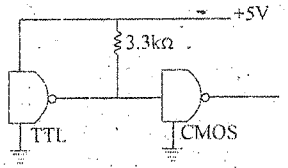


Fig:- TTL driver & CMOS load

10.12 CMOS to TTL interface:-

Here, CMOS is a driving device and TTL is a loading device.

We need a buffer to handle extra sinking current when CMOS output goes low as shown,

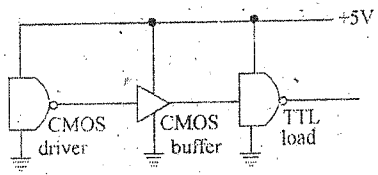


Fig:-CMOS –To-TTL interface

Comparison of TTL & CMOS logic families.

Parameter	Logic Family	
	TTL	CMOS
Propagation delay inns	9	<50
Power dissipation per gate in mw	10	0.01
Noise margin in volt	0.4	5
Fan-In	8	10
Fan-Out	10	50
Cost	low	low

Difference between TTL & CMOS:-

	TTL		CMOS
1	TTL circuit utilizes Bipolar Junction Transistors (BJTs)	1	CMOS circuit utilizes field effect transistors (FFTs)
2	Less density of logic function can be fabricated in a single chip.	2	More density of logic function is fabricated in a single chip compared to TTL.
3	TTL chips are less susceptible to static discharged compared to CMOS.	3	More susceptible to static discharge compared to TTL
4	Consumer more power compared to CMOS	4	Consumer less power compared to TTL

CHAPTER 11

Application

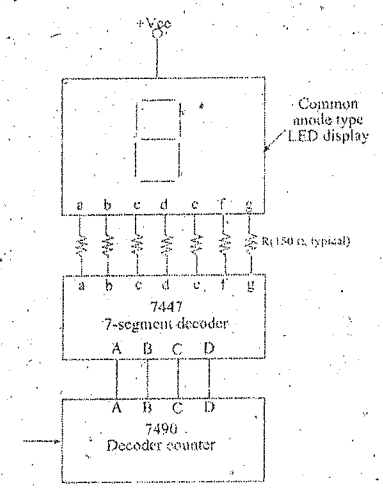
11.1 Multiplexing Display :-

The decimal outputs of digital instruments such as digital voltmeters (DVMs) and frequency counters are often displayed using seven-segment indicators. Such indicators are constructed by using a fluorescent bar, a liquid crystal bar, or a LED bar for each segment. LEDs do generally require more power than either of the other two types, and multiplexing is a technique used to reduce indicator power requirements.

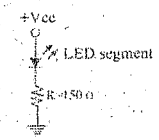
The circuit in Fig. a is a common-anode LED-type seven-segment indicator used to display a single decimal digit. The 7447 BCD to seven-segment decoder is used to drive the indicator, and the four inputs to the 7447 are the four-flip-flop outputs of the 7490 decade counter. 7447 has active low outputs, so the equivalent circuit of an illuminated segment appears as in Fig.b. A 1-Hz square wave applied at the clock input of the 7490 will cause the counter to count upward, advancing one count each second, and the equivalent decimal number will appear on the display.

A similar single decimal digit display using a common-cathode-type LED indicator is shown in Fig. c. The seven-segment decoder used here is the 7448; its outputs are active high, and they are intended to drive buffer amplifiers since their output current capabilities are too small to drive LEDs directly. The equivalent circuit for an illuminated segment is shown in Fig.1.2b.

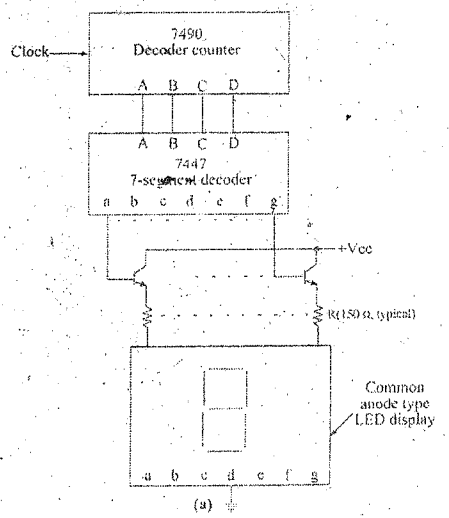
Basically, multiplexing is accomplished by applying current to each display digit in short, repeated pulses rather than continuously.



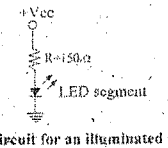
(a) Single decimal-digit display



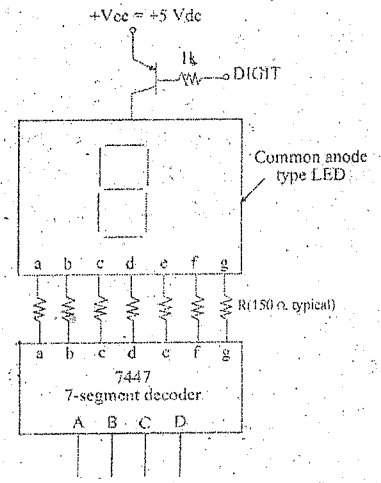
(b) Equivalent circuit for an illuminated segment



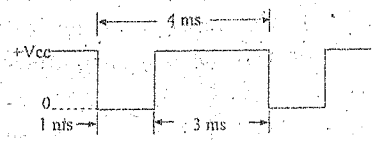
(a)



(b) Equivalent circuit for an illuminated segment



(a) Multiplexed display



(b) DIGIT waveform

11.2 Frequency Counter:- [2069 Ashad] [2068 Chaitra]

A frequency counter is a digital instrument that can be used to measure the frequency of any periodic waveform. The fundamental concepts involved are illustrated in the block diagram. A GATE ENABLE signal that has a known period t is generated with a clock oscillator and a divider circuit and is applied to one leg of an AND gate. The unknown signal is applied to the other leg to the AND gate and acts as the clock for the counter. The counter will advance one count for each transition of the unknown signal, and at the end of the known time period, the contents of the counter will equal the number of periods of the unknown signal that have occurred during t. In other words, the counter contents will be proportional to the frequency of the unknown signal. For instance, suppose that the gate signal is exactly 1 s and the unknown input signal is a 750-Hz square wave. At the end of 1 s, the counter will have counted up to 750, which is exactly the frequency of the input signal.

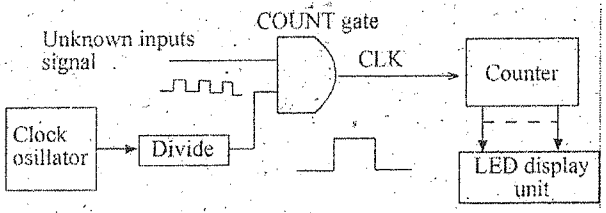


Fig: Basic Frequency Counter

11.3 Time Measurement:-

With only slight modifications, the frequency counter can be converted into an instrument for measuring time. The logic block diagram in illustrates the fundamental ideas used to construct an instrument that can be used to measure the period of any periodic waveform. The unknown voltage is passed through a conditioning amplifier to produce a periodic waveform that is compatible with TTL circuits and is then applied to a JK flip-flop. The output of this flip-flop is used as the ENABLE-gate signal, since it is high for a time t that is exactly equal to the time period of the unknown input voltage. The oscillator and divider provide a series of pulses that are passed through the count gate and serve as the clock for the counter. The contents of the counter and display unit will then be proportional to the time period of the unknown input signal.

For instance, if the unknown input signal is a 5-kHz sine wave and the clock pulses from the divider are 0.1-μs in width and are spaced every 1.0 μs, the counter and display will read 200. Clearly this means 200 μs, since 200 of these 0.1-μs pulses will pass through the COUNT gate during the 200 μs that ENABLE-gate signal is high. Naturally the counter and the display have an accuracy of plus or minus one count.

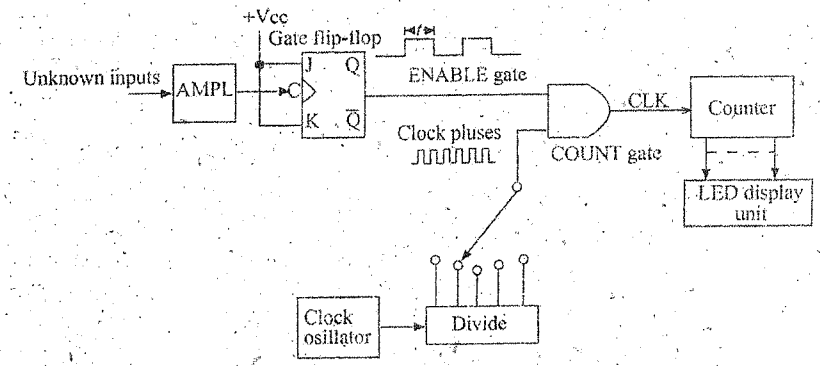


Fig: Instrument to measure time period

Addition Questions

Q.1. What are Johnson and Ring Counter? Explain in detail.

Ans: **The Johnson Counter:-**

In a Johnson Counter the complement of the output of last flip-flop is connected back to the D-input of first flip-flop. This feedback arrangement produces a characteristics sequence of states. The 4-bit sequence has a total of eight states or bit pattern.

In general, a Johnson counter will produce a modules of 2n, where n is the number of stages in the counter. The implementation of 4-bit Johnson counter is as shown,

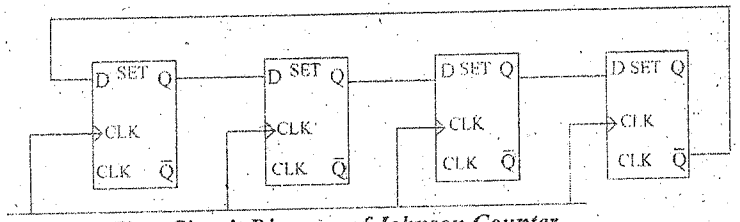


Fig:- Circuit Diagram of Johnson Counter

The Ring Counter:-

The Ring Counter utilizes one flip-flop for each state in its sequence. It has the advantage that decoding gates are not required. A logic diagram for a 4-bit ring counter is as shown,

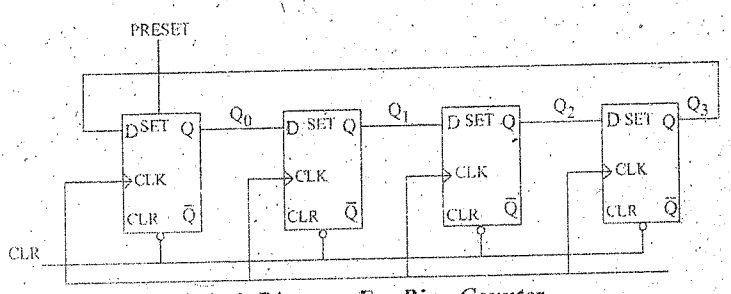
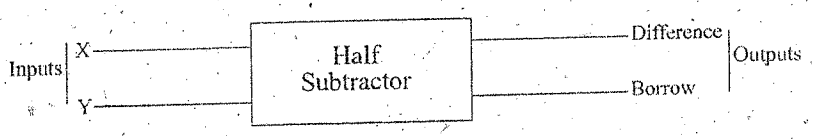


Fig:- Circuit Diagram For Ring Counter

Initially a 1 is present in first flip-flop and the rest of flip-flop are cleared. The inter stage connections are same as those for Johnson counter, except that θ rather than $\bar{\theta}$ is feedback from the last stage. The four outputs of counter indicate directly the decimal count of the clock pulse from zero to three.

Q.2. What are Half Subtractor and Full Subtractor? Explain using their block diagram and truth table?

Ans: **Half Subtractor:-** It is a combinational logic circuit that perform the subtracting of two bits so it has two binary inputs and two binary outputs called DIFFERENCE and BORROW. It consists of Ex-OR gate, NOT gate and AND gate. The output of Ex-OR gate is called DIFFERENCE while the output of AND gate is called BORROW.



Block Diagram of Half Subtractor

Truth Table:

Inputs		Output	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

From the truth table the logic function can be written as,

$$D = X\bar{Y} + \bar{X}Y = X \oplus Y$$

$$B = \bar{X}Y$$

The logic Circuit will be as shown,

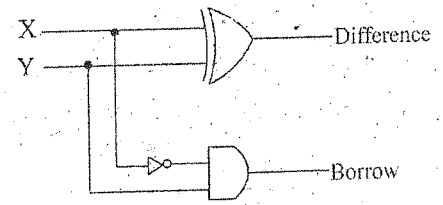


Fig:- Logic Circuit for Half Suntractor

Full Subtractor:- The Half subtractor can handle only 2 bits at a time and can be used for least significant column of a subtraction problem. Hence a full subtractor is a combinational logic circuit that subtracts 3-bits –2 bits minued and subtracted and another Borrow from previous stage. It produces two outputs called DIFFERENCE and BORROW.

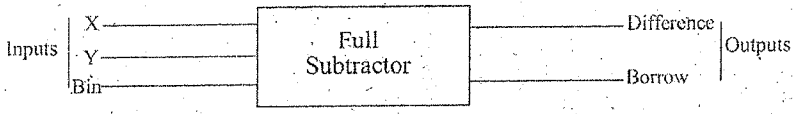


Fig:- Block Diagram of Full Subtractor

Truth Table:

Inputs			Outputs	
X	Y	Bin	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From truth table,

$$B = \bar{X}\,\bar{Y}\,B_{in} + \bar{X}\,Y\,\bar{B}_{in} + \bar{X}\,YB_{in} + XYB_{in}$$

$$= B_{in}(\bar{X}\,\bar{Y} + YY) + \bar{X}\,Y(B_{in} + \bar{B}_{in})$$

$$= B_{in}(\overline{X \oplus Y}) + \bar{B}_{in}(\bar{X}\,Y + X\bar{Y})$$

and, $$D = \bar{X}\,\bar{Y}\,B_{in} + \bar{X}\,Y\,\bar{B}_{pn} + X\bar{Y}\,B_{in} + X\bar{Y}\,\bar{B}_{in}$$

$$= B_{in}(\bar{X}\,\bar{Y} + YY) + \bar{B}_{in}(\bar{X}\,Y + X\bar{Y})$$

$$= B_{in}(\overline{X \oplus Y}) \oplus \bar{B}_{in}(X \oplus Y)$$

$$= X \oplus Y \oplus B_{in}$$

The logic circuit will be as shown,

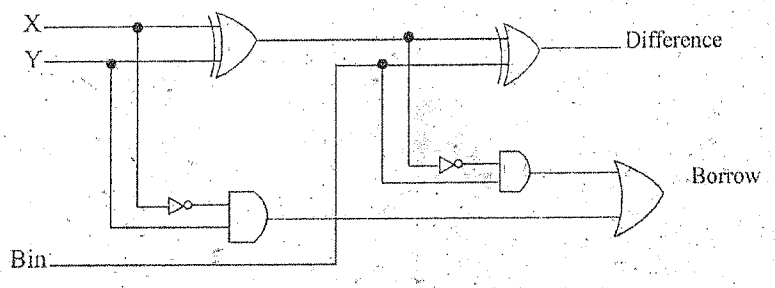


Fig:- Logic Diagram for full-Subtractor

Q.3. Design a full subtractor using two half subtractor.

Sol^n :

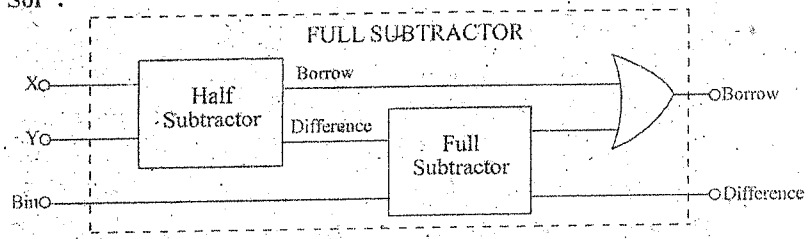


Fig: Full Subtractor Using Half subtractor

Q.4. Design a full adder using two half adder.

Sol^n :

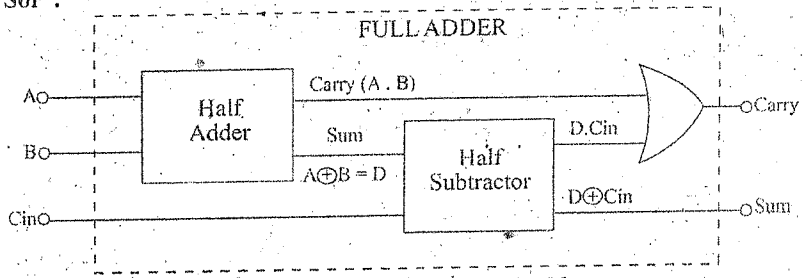
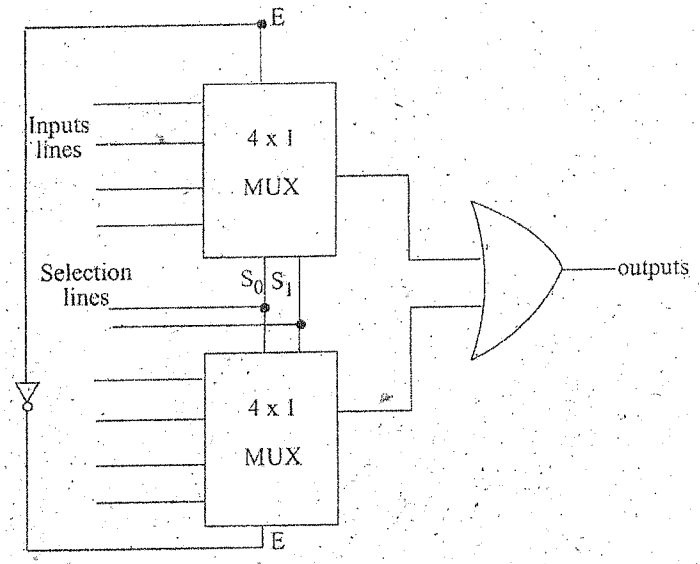


Fig: Full Adder Using Half Adder

Q. 5 Design 8 × 1 MUX using two 4×1 MUX.



Q.6. What do you mean by DC logic system and pulse logic system?

Ans: DC logic (or, level logic) system:- In a dc logic or level logic system, a bit is represented by one of the two voltage levels high or low. If the higher of two voltages represents a 1 and the lower voltage represents 0; the system is called positive logic. On the other hand if the lower voltage represents. 1 and a higher voltage represents 0; the system if called negative logic.

Pulse logic System:- In this logic system, a bit is recognized by the presence or absence of pulse. It is also known as dynamic logic system. This can also be dynamic positive or dynamic negative i.e. state 1 may signify presence of positive pulse or negative pulse. In both cases the absence of pulse signifies 0 level.

Q.7. Realize (a) NOR gate using NAND gate
(b) NAND gate using NOR gate

Sol^n:

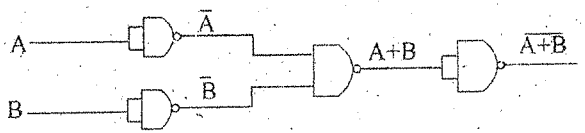


Fig:- Realization of NOR gate using NAND gate

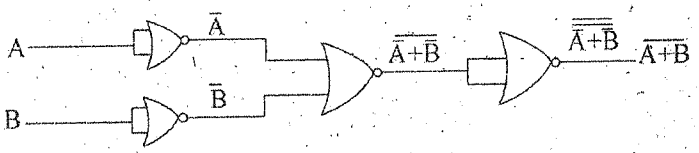
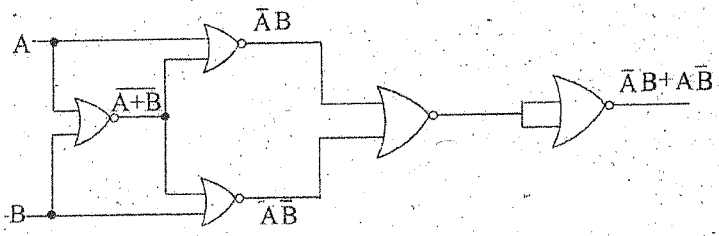


Fig:- Realization or NAND gate using NOR gate

Q.8. Implement XOR gate using NOR gates only.

Sol^n:



Q.9. Realize Ex-OR gate using minimum number of gates.

Sol^n:

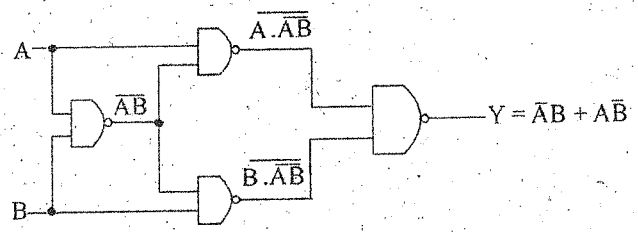


Fig:- Realization of Ex-OR gate with minimum no. of gates

Q.10. Implement (i) AND gate with OR and NOT gates
(ii) OR gate with AND gate and NOT gates

Sol^n:

(i) Let $Z = A \cdot B$

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

Again taking complement,

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

(ii) Let $Z = A + B$

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

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

Again taking complement

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

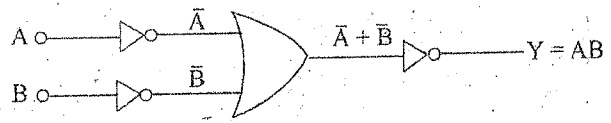


Fig:- AND gate using OR & NOT gate

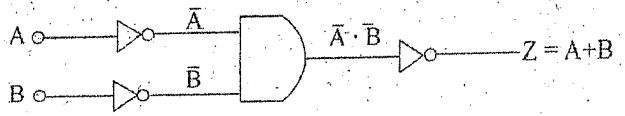


Fig: OR gate using AND & NOT gate

Q.11. Realize X-NOR gate Using NAND gates only

Sol^n : Let

$$Z = \overline{A \oplus B}$$

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

Truth table for X-NOR gate is,

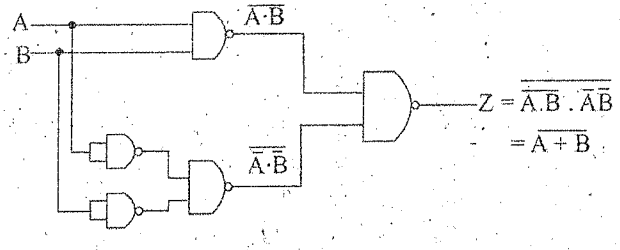
A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

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

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

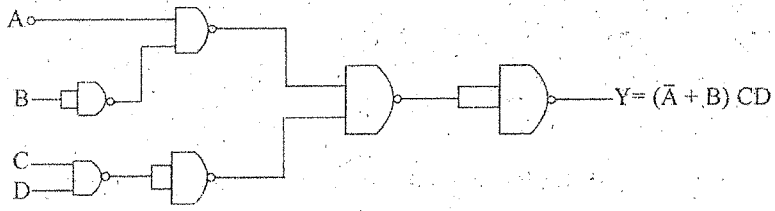
$$= A \cdot B + \bar{A} \cdot \bar{B}$$

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

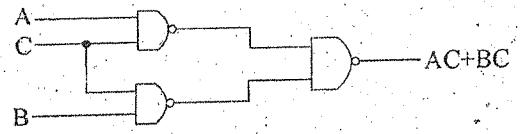


Q.12. Develop a circuit (i) Y = ($\bar{A}$ + B) CD using NAND gates only.
(ii) Y = (A + B) C using NAND gates only.

Sol^n : (i) Circuit Y = ($\bar{A}$ + B) CD is as shown,



(ii) Circuit for Y = (A + B)C



Q.13. Implement the equation $F = \bar{x}\,\bar{y} + xy + \bar{y}\,z$ using
(i) NAND gates only
(ii) NOR gates only
(iii) AND and NOT gates
(iv) OR and NOT gates

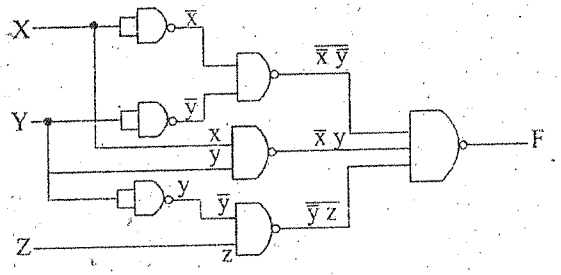


Fig: Implementation Using NAND gates only

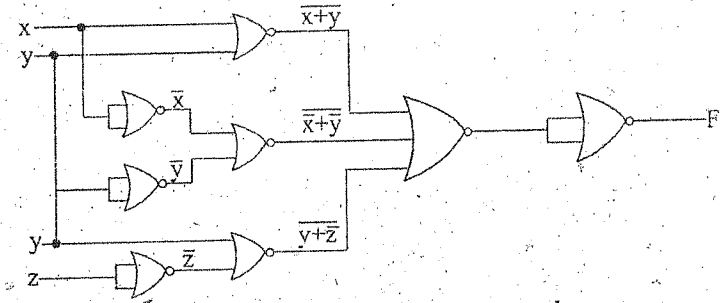


Fig: Implementation Using NOR gates only

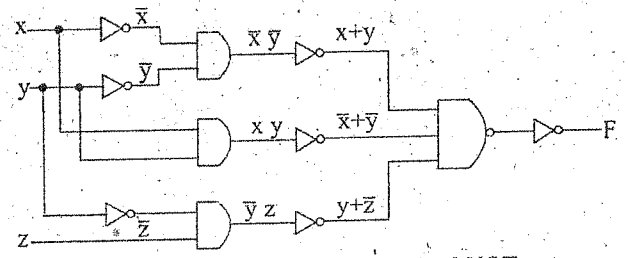


Fig: Implementation Using AND and NOT gates

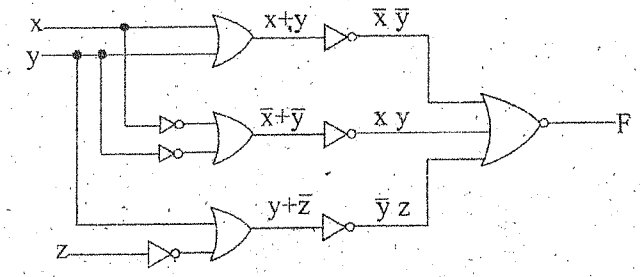
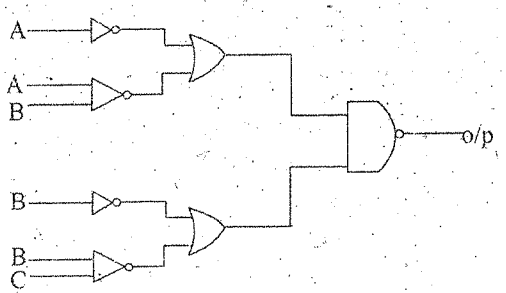


Fig: Implementation by using OR and NOT gates

Q.14. Draw the simplest possible logic diagram that implements the output of the logic diagram shown below,



Sol^n : The output of a logic diagram shown above is,

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

Apply De- Morgan's theorem,

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

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

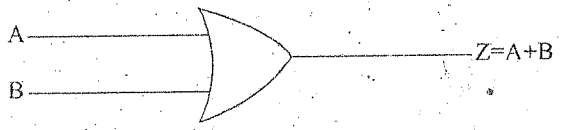
$$= A^2 + AB + B^2 + BC$$

$$= A + AB + B + BC$$

$$= A(1 + B) + B(1 + C)$$

$$= A + B$$

Hence the simplest possible logic diagram is as shown,



Q.15. Realize the expression using Ex-OR gate and basis gates if required.

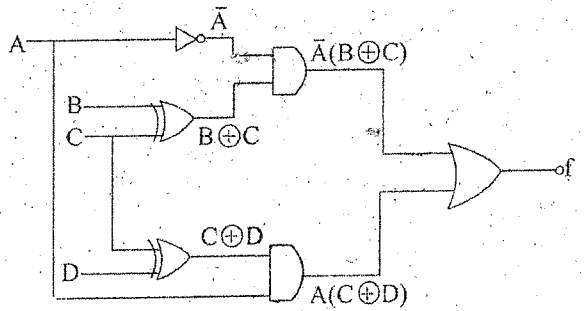
$$f = \bar{A}\,B\,\bar{C} + \bar{A}\,\bar{B}\,C + A\bar{C}\,D + AC\bar{D}$$

Sol^n: $f = \bar{A}\,B\,\bar{C} + \bar{A}\,\bar{B}\,C + A\bar{C}\,D + AC\bar{D}$

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

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

Realizing above circuit using logic gates,



Q.16. Express the Boolean Function $F = AB + \bar{A}\,C$ is POS & SOP form.

Ans: Here,

$$F = AB + \bar{A}\,C$$

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

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

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

$$= m_7 + m_6 + m_3 + m_1$$

Thus SOP form of Boolean function given is $ABC + AB\bar{C} + \bar{A}\,BC + \bar{A}\,\bar{B}\,C$

In SOP form, the minterms 0, 2, 4, 5 are missing. So in POS form, the maxterms 0, 2, 4, 5 will be present.

Thus POS form is $\pi\,(0, 2, 4, 5)$

Writing in POS form,

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

Q.17. Convert the given expression into comornical SOP form $f = A + AB + ABC$.

Sol^n : $f = A + AB + ABC$

$= A+AB + (1 + C)$

$= A + AB$

$= A (1 + B)$ $[\because 1+C = 1]$

$= A$ $[\because 1+B = 1]$.

SOP form may be had as,

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

Q.18. Express the Boolean function $F = xy + z$ in product of maxterms.

Soln: Here,

$F = xy + z$

$= xy (z + \bar{z}) + z (xy + x\bar{y} + \bar{x} y + \bar{x} \bar{y})$

$= xyz + xy\bar{z} + xyz + x\bar{y} z + \bar{x} yz + \bar{x} \bar{y} z$

$= m_7 + m_6 + m_5 + m_3 + m_1$

$\therefore \bar{F} = m_0 + m_2 + m_4$

By De-Morgan's theorem,

$F = \bar{\bar{F}} = m_0 + m_2 + m_4$

$= \pi m (0, 2, 4)$

$= (x+y+z) (x+\bar{y} +z) (\bar{x} +y+z)$

Q.19 Convert the given expression into canonical POS form

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

Sol^n: $f = (A+B) (B+C) (C+A)$

$= (AB+AC+B^2+BC) (C + A)$

$= ABC + A^2B + AC^2 + A^2C + B^2C + B^2A + BC^2 + BCA$

$= ABC + AB + AC + AC + BC + AB + BC + ABC$

$= ABC + AB + BC + AC$

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

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

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

$\therefore f = m_7 + m_6 + m_5 + m_3$

Now,

$f = m_7 + m_6 + m_5 + m_3$

By De-Morgan's theorem,

$f = \overline{m_0 + m_1 + m_2 + m_4} = \bar{m}_0 \ \bar{m}_1 \ \bar{m}_2 \ \bar{m}_4$

$= \pi m(0, 1 2, 4)$

Writing in POS form,

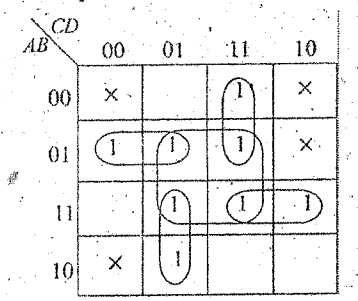
$f = (A + B + C) . (A + B + \bar{C}) . (A + \bar{B} + C) . (\bar{A} + B + C)$

Q.20. Minimize the given Boolean function using K-map $f (A, B, C, D) = \sum_m (3, 4, 5, 7, 9, 13, 14, 15) + d (0, 2, 8)$. Implement the minimized function using NAND gates only.

Soln: Given,

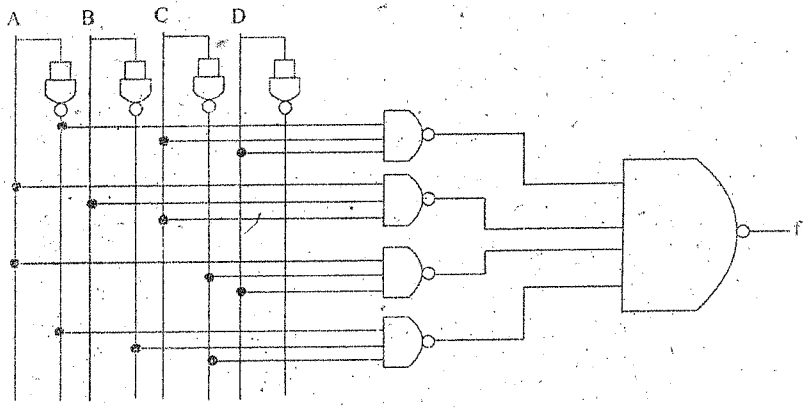
$f (A, B, C, D) = \sum_m (3, 4, 5, 7, 9, 13, 14, 15) + d (0, 2, 8)$

Representing in K-map



$\therefore f (A, B, C, D) = \bar{A} CD + ABC + A\bar{C} D + \bar{A} B\bar{C}$

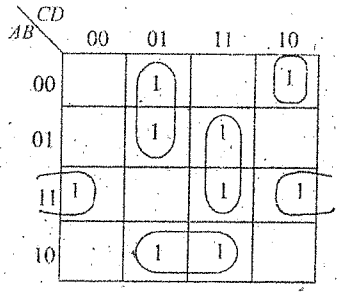
minimized function implementation using NAND gate is as shown,



Q.21 Minimize & realize $f = \sum (1, 2, 5, 7, 9, 11, 12, 14, 15)$ using one type of universal gates only.

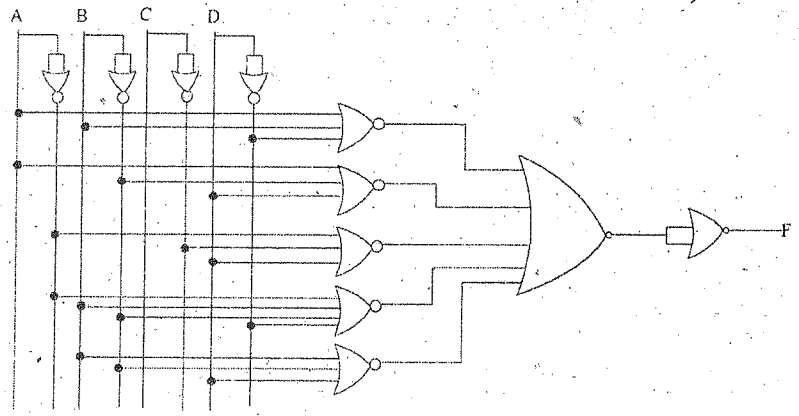
Sol^n: $f = \sum (1, 2, 5, 7, 9, 11, 12, 14, 15)$

Representing in K-map



$\therefore f = AB\bar{D} + A\bar{B}D + \bar{A}\bar{C}D + \bar{A}\bar{B}C\bar{D} + BCD$

Simplified function implementation using NOR gates only in as shown,



Q.22. An equality detector gives the output y = 1 if both inputs A & B are either 1 or 0.

(i) Construct the truth table

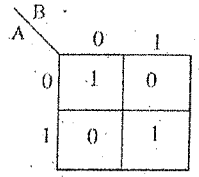
(ii) Write the Boolean expression of y

(iii) Implement the circuit using NAND gates only

Sol^n: (i) The truth table is given as,

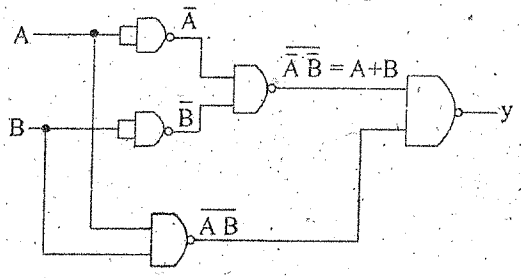
A	B	Y
0	0	1
1	1	1
0	1	0
1	0	0

(ii) Constructing K-map we have,



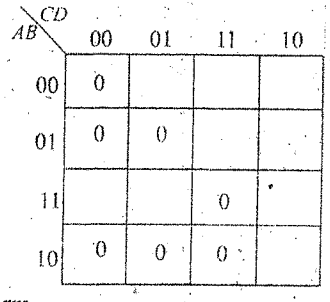
$\therefore Y = \bar{A}\bar{B} + AB$

(iii) Circuit Implementation with NAND gates,

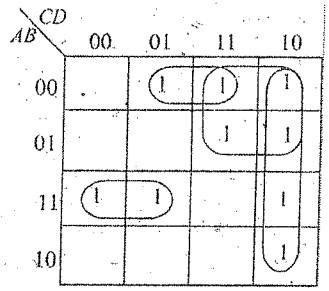


Q.23. Simplify π (0, 4, 5, 8, 9, 11, 15) using K-map & write its SOP expression.

Sol^n:



Now for SOP form,



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

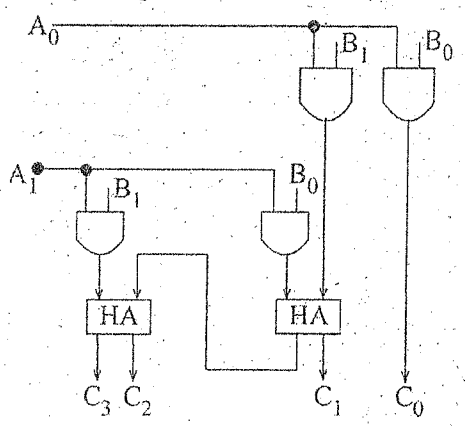
Q.24. Design a combinational logic circuit that performs multiplication between two 2-bit numbers.

Soln: Consider two bit numbers A_1A_0 and $_1B_0$

Then,

$$\begin{array}{cccc} & & B_1 & B_0 \\ & & \times A_1 & A_0 \\ \hline & & A_1B_1 & A_0B_0 \\ & A_1B_1 & A_1B_0 & \times \\ \hline C_3 & C_2 & C_1 & C_0 \end{array}$$

The logic diagram will be as shown,



Q.25. Prove that : $AB + \bar{A}C = (A + C)(\bar{A} + B)$

Soln: Here,

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

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

$$= 0 + AB + \bar{A}C + BC$$

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

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

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

$$= AB + \bar{A}C = \text{LHS proved}$$

Q.26. Simplify :- (i) $(AB + \bar{A}C + BC)(A + \bar{B} + A\bar{B})$

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

(iii) $(AB + C + D)(\bar{C} + D)(\bar{C} + D + E)$

Sol^n: (i) $(AB + \bar{A}C + BC)(A + \bar{B} + A\bar{B})$

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

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

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

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

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

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

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

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

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

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

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

(iii) $(AB + C + D)(\bar{C} + D)(\bar{C} + D + E)$

$$= AB\bar{C} + \bar{C}D + ABD + D(1 + C)(\bar{C} + D + E) \quad [\because D.D = D,\ C.\bar{C} = 0 \& 1 + C = C]$$

$$= [AB\bar{C} + \bar{C}D + D + ABD](\bar{C} + D + E)$$

$$= [AB\bar{C} + D(1 + \bar{C} + AB](\bar{C} + D + E)$$

$$= (AB\bar{C} + D)(\bar{C} + D + E)$$

$$= AB\bar{C} + D\bar{C} + ABD\bar{C} + D + AB\bar{C}E + DE$$

$$= AB\bar{C} + D(1 + \bar{C} + ABD + E) + AB\bar{C}E$$

$$= AB\bar{C}(1 + E)D)$$

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

Q.27 Prove that :- $\overline{\overline{AB} + \bar{A} + AB} = 0$

Soln: LHS $= \overline{\overline{AB} + \bar{A} + AB}$

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

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

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

$= AB \cdot \overline{AB}$

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

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

$= 0$ proved