

# SKP Engineering College

Tiruvannamalai – 606611

A Course Material

on

Digital Electronics



By

**Dr.N.Nandhagopal**

**Associate Professor**

**Electronics and Communication Engineering Department**

### Quality Certificate

This is to Certify that the Electronic Study Material

Subject Code: EC6302

Subject Name: DIGITAL ELECTRONICS

Year/Sem:II/III

Being prepared by me and it meets the knowledge requirement of the University curriculum.

Signature of the Author

Name: Dr.N.Nandhagopal

Designation: Associate Professor

This is to certify that the course material being prepared by Dr.N.Nandhagopal is of the adequate quality. He has referred more than five books and one among them is from abroad author.

Signature of HD

Name:Mr.R.Saravanakumar

Seal:

Signature of the Principal

Name: Dr.V.Subramania Bharathi

Seal:

## EC6302 DIGITAL ELECTRONICS

L T P C 3 0 0 3

## OBJECTIVES:

- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
- To introduce the concept of memories and programmable logic devices.
- To illustrate the concept of synchronous and asynchronous sequential circuits

## UNIT I MINIMIZATION TECHNIQUES AND LOGIC GATES

9

Minimization Techniques: Boolean postulates and laws – De-Morgan’s Theorem - Principle of Duality - Boolean expression - Minimization of Boolean expressions — Minterm – Maxterm - Sum of Products (SOP) – Product of Sums (POS) – Karnaugh map Minimization – Don’t care conditions – Quine - Mc Cluskey method of minimization.

Logic Gates: AND, OR, NOT, NAND, NOR, Exclusive-OR and Exclusive-NOR Implementations of Logic Functions using gates, NAND-NOR implementations – Multi level gate implementations- Multi output gate implementations. TTL and CMOS Logic and their characteristics – Tristate gates

## UNIT II COMBINATIONAL CIRCUITS

9

Design procedure – Half adder – Full Adder – Half subtractor – Full subtractor – Parallel binary adder, parallel binary Subtractor – Fast Adder - Carry Look Ahead adder – Serial Adder/Subtractor - BCD adder – Binary Multiplier – Binary Divider - Multiplexer/ Demultiplexer – decoder - encoder – parity checker – parity generators – code converters - Magnitude Comparator.

## UNIT III SEQUENTIAL CIRCUITS

9

Latches, Flip-flops - SR, JK, D, T, and Master-Slave – Characteristic table and equation – Application table – Edge triggering – Level Triggering – Realization of one flip flop using other flip flops – serial adder/subtractor- Asynchronous Ripple or serial counter – Asynchronous Up/Down counter - Synchronous counters – Synchronous Up/Down counters – Programmable counters – Design of Synchronous counters: state diagram- State table – State minimization –State assignment - Excitation table and maps-Circuit implementation - Modulo-n counter, Registers – shift registers - Universal shift registers – Shift register counters – Ring counter – Shift counters - Sequence generators.

UNIT IV MEMORY DEVICES

9

Classification of memories – ROM - ROM organization - PROM – EPROM – EEPROM – EAPROM, RAM – RAM organization – Write operation – Read operation – Memory cycle - Timing wave forms – Memory decoding – memory expansion – Static RAM Cell- Bipolar RAM cell – MOSFET RAM cell – Dynamic RAM cell –Programmable Logic Devices – Programmable Logic Array (PLA) - Programmable Array Logic (PAL) – Field Programmable Gate Arrays (FPGA) - Implementation of combinational logic circuits using ROM, PLA, PAL

UNIT V SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS 9

Synchronous Sequential Circuits: General Model – Classification – Design – Use of Algorithmic State Machine – Analysis of Synchronous Sequential Circuits Asynchronous Sequential Circuits: Design of fundamental mode and pulse mode circuits – Incompletely specified State Machines – Problems in Asynchronous Circuits – Design of Hazard Free Switching circuits. Design of Combinational and Sequential circuits using VERILOG.

TOTAL: 45

PERIODS OUTCOMES: Students will be able to:

- Analyze different methods used for simplification of Boolean expressions.
- Design and implement Combinational circuits.
- Design and implement synchronous and asynchronous sequential circuits.
- Write simple HDL codes for the circuits.

TEXT BOOK:

1. M. Morris Mano, "Digital Design", 4th Edition, Prentice Hall of India Pvt. Ltd., 2008 / Pearson Education (Singapore) Pvt. Ltd., New Delhi, 2003.

REFERENCES:

1. John F.Wakerly, "Digital Design", Fourth Edition, Pearson/PHI, 2008
2. John.M Yarbrough, "Digital Logic Applications and Design", Thomson Learning, 2006.
3. Charles H.Roth. "Fundamentals of Logic Design", 6th Edition, Thomson Learning, 2013.
4. Donald P.Leach and Albert Paul Malvino, "Digital Principles and Applications", 6th Edition, TMH, 2006.
5. Thomas L. Floyd, "Digital Fundamentals", 10th Edition, Pearson Education Inc, 2011
6. Donald D.Givone, "Digital Principles and Design", TMH, 2003.

### CONTENTS

<b>S.No</b>	<b>Particulars</b>	<b>Page</b>
1	Unit – I	6
2	Unit – II	36
3	Unit – III	60
4	Unit – IV	78
5	Unit – V	98

**Unit- I****Minimization Techniques and Logic Gates****Part-A****1. State the associative property of Boolean algebra[CO1-L1]**

The associative property of Boolean algebra states that the OR ing of several variables results in the same regardless of the grouping of the variables. The associative property is stated as follows:

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

**2. State the commutative property of Boolean algebra[CO1-L1].**

The commutative property states that the order in which the variables are OR ed makes no difference.

The commutative property is:

$$A + B = B + A$$

**3. State the distributive property of Boolean algebra[CO1-L1].**

The distributive property states that ANDing several variables and OR ing the result with a single variable is equivalent to OR ing the single variable with each of the several variables and then ANDing the sums. The distributive property is:

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

**4. Simplify the following using De Morgan's theorem[CO1-L1].**

$$[((AB)'C)'' D]' = ((AB)'C)'' + D' [(AB)' = A' + B']$$

$$= (AB)' C + D'$$

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

**5. State De Morgan's theorem[CO1-L1].**

De Morgan suggested two theorems that form important part of Boolean algebra.

They are,

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

$$(AB)' = A' + B'$$

- 2) The complement of a sum term is equal to the product of the complements.

$$(A + B)' = A'B'$$

**6. Reduce A.A'C[CO1-H1].**

$$\begin{aligned} A.A'C &= 0.C \quad [A.A' = 1] \\ &= 0 \end{aligned}$$

**7. Reduce A (A + B) [CO1-H1].**

$$\begin{aligned} A(A + B) &= AA + AB \\ &= A(1 + B) \quad [1 + B = 1] \\ &= A. \end{aligned}$$

**8. Reduce A'B'C' + A'BC' + A'BC[CO1-H1].**

$$\begin{aligned} A'B'C' + A'BC' + A'BC &= A'C'(B' + B) + A'BC \\ &= A'C' + A'BC \quad [A + A' = 1] \\ &= A'(C' + BC) \\ &= A'(C' + B) \quad [A + A'B = A + B] \end{aligned}$$

**9. Reduce AB + (AC)' + AB'C (AB + C) [CO1-H1].**

$$\begin{aligned} AB + (AC)' + AB'C(AB + C) &= AB + (AC)' + AAB'BC + AB'CC \\ &= AB + (AC)' + AB'CC \quad [A.A' = 0] \\ &= AB + (AC)' + AB'C \quad [A.A = 1] \\ &= AB + A' + C' = AB'C \quad [(AB)' = A' + B'] \end{aligned}$$

$$\begin{aligned}
&= A' + B + C' + AB'C [A + AB' = A + B] \\
&= A' + B'C + B + C' [A + A'B = A + B] \\
&= A' + B + C' + B'C \\
&= A' + B + C' + B' \\
&= A' + C' + 1 \\
&= 1 [A + 1 = 1]
\end{aligned}$$

**10. Simplify the following expression  $Y = (A + B)(A + C')(B' + C')$  [CO1-H1].**

$$\begin{aligned}
Y &= (A + B)(A + C')(B' + C') \\
&= (AA' + AC + A'B + BC)(B' + C') [A.A' = 0] \\
&= (AC + A'B + BC)(B' + C') \\
&= AB'C + ACC' + A'BB' + A'BC' + BB'C + BCC' \\
&= AB'C + A'BC'
\end{aligned}$$

**11. Show that  $(X + Y' + XY)(X + Y')(X'Y) = 0$  [CO1-H1].**

$$\begin{aligned}
&(X + Y' + XY)(X + Y')(X'Y) \\
&= (X + Y' + X)(X + Y')(X' + Y) [A + A'B = A + B] \\
&= (X + Y')(X + Y')(X'Y) [A + A = 1] \\
&= (X + Y')(X'Y) [A.A = 1] \\
&= X.X' + Y'.X'.Y \\
&= 0 [A.A' = 0]
\end{aligned}$$

**12. Prove that  $ABC + ABC' + AB'C + A'BC = AB + AC + BC$  [CO1-H1].**

$$\begin{aligned}
&ABC + ABC' + AB'C + A'BC \\
&= AB(C + C') + AB'C + A'BC \\
&= AB + AB'C + A'BC
\end{aligned}$$



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

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

$$=AB + AC + A'BC$$

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

$$=AB + BC + AC$$

$$=AB + AC + BC$$

**13. Convert the given expression in canonical SOP form  $Y = AC + AB + BC$  [CO1-H1].**

$$Y = AC + AB + BC$$

$$=AC(B + B') + AB(C + C') + (A + A')BC$$

$$=ABC + ABC' + AB'C + AB'C' + ABC + ABC' + ABC$$

$$=ABC + ABC' + AB'C + AB'C' [A + A = 1]$$

**14. Define duality property[CO1-H1].**

Duality property states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

**15. Find the complement of the functions  $F1 = x'yz' + x'y'z$  and  $F2 = x(y'z' + yz)$  [CO1-L1].**

By applying De-Morgan's theorem.

$$\begin{aligned} F1' &= (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' \\ &= (x + y' + z)(x + y + z') \end{aligned}$$

$$\begin{aligned} F2' &= [x(y'z' + yz)]' = x' + (y'z' + yz)' \\ &= x' + (y'z')'(yz)' \end{aligned}$$

$$= x' + (y + z) (y' + z')$$

**16. Define Canonical form[CO1-L1].**

Boolean functions expressed as a sum of minterms ( $\sum m$ ) or product of maxterms ( $\prod M$ ) are said to be in canonical form.

**17. Define Minterm and Maxterm[CO1-L1].**

Each individual term in standard SOP form is called a Minterm

$$\text{Example: } f(A, B, C) = AB'C + ABC + A'BC'$$

Each individual term in standard POS form is called a Maxterm

$$\text{Example: } f(A, B, C) = (A+B+C) (A+B'+C)$$

**18. What are the methods adopted to reduce Boolean function? [CO1-L1].**

- i) Karnaugh map
- ii) Tabular method or Quine Mc-Cluskey method
- iii) Variable entered map technique.

**19. State the limitations (or) disadvantages of karnaugh map. [CO1-L1].**

- i) Generally it is limited to six variable map (i.e.) more than six variable involving expressions are not reduced.
- ii) The map method is restricted in its capability since they are useful for simplifying only Boolean expression represented in standard form.

**20. What is a karnaugh map? [CO1-L1].**

A karnaugh map or k map is a pictorial form of truth table, in which the map diagram is made up of squares, with each squares representing one minterm of the function.

**21. Find the minterms of the logical expression  $Y = A'B'C' + A'B'C + A'BC + ABC'$  [CO1-H1].**

$$Y = A'B'C' + A'B'C + A'BC + ABC'$$

$$=m_0 + m_1 + m_3 + m_6$$

$$=\sum m (0, 1, 3, 6)$$

**22. Write the maxterms corresponding to the logical expression [CO1-L1].**

$$Y = (A + B + C') (A + B' + C') (A' + B' + C)$$

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

$$=M_1.M_3.M_6$$

$$= \prod M (1, 3, 6)$$

**23. What are called don't care conditions? [CO1-L1].**

In some logic circuits certain input conditions never occur, therefore the corresponding output never appears. In such cases the output level is not defined, it can be either high or low. These output levels are indicated by 'X' or 'd' in the truth tables and are called don't care conditions or incompletely specified functions.

**24. What is a prime implicant? [CO1-L1].**

A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map.

**25. What is an essential implicant? [CO1-L1].**

If a min term is covered by only one prime implicant, the prime implicant is said to be essential

**26. What is a Logic gate? [CO1-L1].**

Logic gates are the basic elements that make up a digital system. The electronic gate is a circuit that is able to operate on a number of binary inputs in order to perform a particular logical function.

**27. Which gates are called as the universal gates? What are its advantages?(Nov/Dec 2009) [CO1-L1].**

The NAND and NOR gates are called as the universal gates. These gates are used to perform any type of logic application.

**28. Classify the logic family by operation? [CO1-L1].**

The Bipolar logic family is classified into Saturated logic and Unsaturated logic.

- ✓ The RTL, DTL, TTL, I<sup>2</sup>L, HTL logic comes under the saturated logic family.
- ✓ The Schottky TTL, and ECL logic comes under the unsaturated logic family.

**29. Mention the classification of saturated bipolar logic families. [CO1-L1].**

The bipolar logic family is classified as follows:

RTL- Resistor Transistor Logic

DTL- Diode Transistor logic

IIL- Integrated Injection Logic

TTL- Transistor Transistor Logic

ECL- Emitter Coupled Logic

**30. Define Fan-out? [CO1-L1].**

Fan out specifies the number of standard loads that the output of the gate can drive without impairment of its normal operation.

**31. Define power dissipation? [CO1-L1].**

Power dissipation is measure of power consumed by the gate when fully driven by all its inputs.

**32. What is propagation delay? [CO1-L1].**

Propagation delay is the average transition delay time for the signal to propagate from input to output when the signals change in value. It is expressed in ns.

**33. Define noise margin? [CO1-L1].**

It is the maximum noise voltage added to an input signal of a digital circuit that does not cause an undesirable change in the circuit output. It is expressed in volts.

**34. What are the types of TTL logic? [CO1-L1].**

1. Open collector output
2. Totem-Pole Output
3. Tri-state output.

**35. What is depletion mode operation MOS? [CO1-L1].**

If the channel is initially doped lightly with p-type impurity a conducting channel exists at zero gate voltage and the device is said to operate in depletion mode.

**36. What is enhancement mode operation of MOS?**

If the region beneath the gate is left initially uncharged the gate field must induce a channel before current can flow. Thus the gate voltage enhances the channel current and such a device is said to operate in the enhancement mode.

**37. Mention the characteristics of MOS transistor? [CO1-L1].**

1. The n- channel MOS conducts when its gate- to- source voltage is positive.
2. The p- channel MOS conducts when its gate- to- source voltage is negative
3. Either type of device is turned off if its gate- to- source voltage is zero.

**38. How schottky transistors are formed and state its use? [CO1-L1].**

A schottky diode is formed by the combination of metal and semiconductor. The presence of schottky diode between the base and the collector prevents the transistor from going into saturation. The resulting transistor is called as schottky transistor. The use of schottky transistor in TTL decreases the propagation delay without a sacrifice of power dissipation.

**39. State advantages and disadvantages of TTL[CO1-L1].**

Advantages:

Easily compatible with other ICs

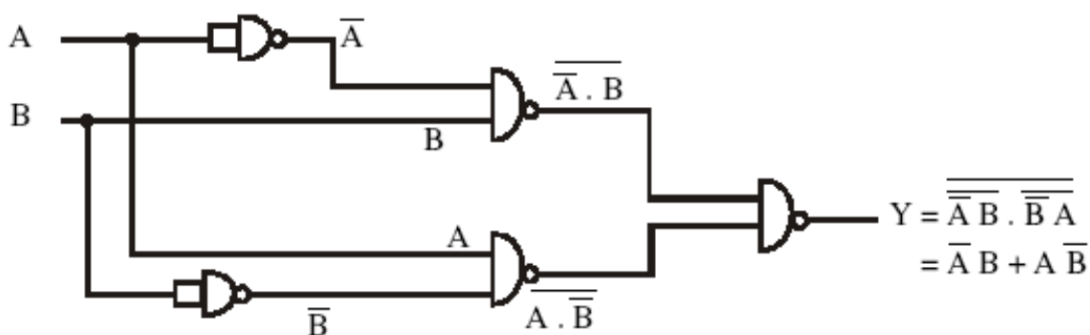
Low output impedance

Disadvantages:

Wired output capability is possible only with tristate and open collector types Special circuits in Circuit layout and system design are required.

**40. When does the noise margin allow digital circuits to function properly? [CO1-L1].**

When noise voltages are within the limits of V<sub>NA</sub> (High State Noise Margin) and V<sub>NK</sub> for a particular logic family.

**41. Implement the Boolean Expression for EX – OR gate using NAND Gates.[CO1-L1].****42. Define Binary Logic[CO1-L1].**

Binary logic consists of binary variables and logical operations. The variables are

designated by the alphabets such as A, B, C, x, y, z, etc., with each variable having only two distinct values: 1 and 0. There are three basic logic operations: AND, OR, and NOT.

**43. What is Totem output? [CO1-L1].**

In two input NAND gate the transistors  $Q_3$  &  $Q_4$  form a totem pole output. The active pull up formed by  $Q_3$  &  $Q_4$  has specific advantage. Totem pole transistors are used because they produce a low output impedance.

**44. State the different classification of binary codes? [CO1-L1].**

1. Weighted codes
2. Non - weighted codes
3. Reflective codes
4. Sequential codes
5. Alphanumeric codes
6. Error Detecting and correcting codes.

**Part-B**

**1. Explain SOP AND POS FORMS. [CO1-L2].**

**Sum of Products (SOP):**

The logical sum of two or more logical product terms is called a Sum of Products expression. It is basically an OR operation of AND operated variables.

Example:  $AB + ABC + CDE$ .

**Product of Sum (POS):**

A product of sums expression is a logical product of two or more logical sum terms. It is basically an AND operation of OR operated variables.

Example :  $(A+B) * (A + B + C) * (C +D)$ .

The term "standard" here means that the expression consists exclusively of **minterms** (in the case of Standard SOP) or **maxterms** (in the case of Standard POS).

### Converting Boolean Expressions into SOP/POS Form:

The process of converting any Boolean expression into either POS or SOP form (canonical or otherwise) is very straightforward.

To get the expression in SOP form, you simply distribute all AND operations over any OR operations and continue doing this as long as possible. When finished, you will have an expression in SOP form. If you want it in canonical form, then you simply expand each term as necessary.

To get the expression in POS form, you simply distribute all OR operations over any AND operations and continue doing this as long as possible. When finished, you will have an expression in POS form. If you want it in canonical form, then you simply expand each term as necessary.

### 2. What are Minterms and Maxterms? [CO1-L1].

Each individual term in standard SOP form is called minterm. Minterms are standard product.(AND terms)

Each individual term in standard POS form is called maxterm. Maxterms are standard sum.(OR terms)

Variable			Minterm		Maxterm	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	$m_0$	$x+y+z$	$M_0$
0	0	1	$x'y'z$	$m_1$	$x+y+z'$	$M_1$
0	1	0	$x'yz'$	$m_2$	$x+y'+z$	$M_2$
0	1	1	$x'yz$	$m_3$	$x+y'+z'$	$M_3$
1	0	0	$xy'z'$	$m_4$	$x'+y+z$	$M_4$
1	0	1	$xy'z$	$m_5$	$x'+y+z'$	$M_5$
1	1	0	$xyz'$	$m_6$	$x'+y'+z$	$M_6$
1	1	1	$xyz$	$m_7$	$x'+y'+z'$	$M_7$



### 3. Write short notes on various K-MAP representations [CO1-L1].

A Karnaugh map provides a pictorial method of grouping together expressions with common factors and therefore eliminating unwanted variables. The Karnaugh map can also be described as a special arrangement of a truth table.

The diagram below illustrates the correspondence between the Karnaugh map and the truth table for the general case of a two variable problem.

A	B	F
0	0	a
0	1	b
1	0	c
1	1	d

Truth Table.

		A	
		0	1
B	0	a	b
	1	c	d

F.

The values inside the squares are copied from the output column of the truth table, therefore there is one square in the map for every row in the truth table. Around the edge of the Karnaugh map are the values of the two input variable. A is along the top and B is down the left hand side. The diagram below explains this:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

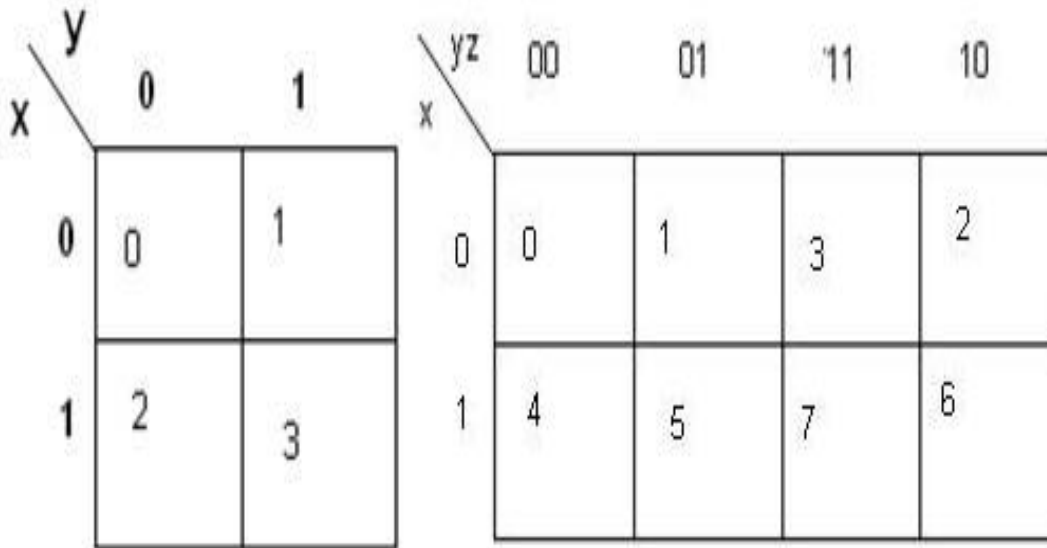
Truth Table.

		A	
		0	1
B	0	0	1
	1	1	1

F.

The values around the edge of the map can be thought of as coordinates. So as an example, the square on the top right hand corner of the map in the above diagram has coordinates A=1 and B=0. This square corresponds to the row in the truth table where A=1 and B=0 and F=1. Note that the value in the F column represents a particular function to which the Karnaugh map corresponds.

If we have  $n$  variables then there would be  $2^n$  min terms and hence  $2^n$  number of blocks in the K-map and also min terms are arranged in their gray code order (not in binary order). 2-variable map and 3-variable map are shown below:



Also there are minterms mentioned in the squares of the map

Similarly we can have a K-map for 4 variables, 5 variables, 6 variables etc but we generally use only till 4 variables maps as higher maps are difficult to use.

Now to simplify any expression we first convert the expression into its canonical form and then mark a 1 in the corresponding column of min terms present in the expression and then we'll combine 1's and make groups of 2 or 4 or 8 or 16 terms and then write the shortened expression.

Any two adjacent squares in the K map differ only by 1 bit change. So the thing where the term corresponding to one square differs from the term corresponding to the adjacent square is that one literal is in compliment form and same literal in other is non-compliment. So sum of those min terms would lead to elimination of one literal and simplified result would be a single AND term.

Also note that squares on one edge of the k-map are adjacent to the opposite edge of the K-map. Hence m0 is adjacent to m2, m4 is adjacent to m6 but m0 is not adjacent to m6. Similarly we can check the adjacent cells in 4-variable map.

#### 4. Explain Don't care conditions[CO1-L2].

When ever there are don't cares present in the K-map then we have the option of including don't care in the group to maximize the size of group and hence we get more minimized form but this is not compulsory to include each and every don't care. So we can use don't care to our advantage otherwise skip them.

e.g. Minimize the K-map given below:

	yz	00	01	11	10
x	0	1	X	0	0
	1	1	X	0	X

If we ignore don't care then we make groups of 1 then we get the equations as

	yz	00	01	11	10
x	0	1	X	0	0
	1	1	X	0	X

The equation we get is

$$F = y'z'$$

But if we use the don't cares then we make groups as

	yz	00	01	11	10
x	0	1	X	0	0
	1	1	X	0	X

The equation we get is

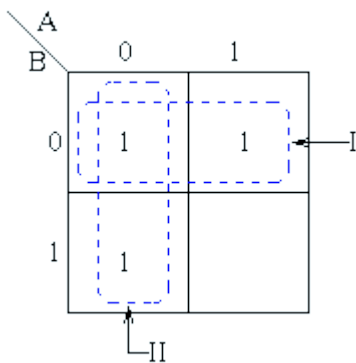
$$F = Y'$$

And we see that we have used 2 don't care and we make a group of 4 rather than 2 while we skip the 3<sup>rd</sup> don't care as we do not have to use each don't care. Hence we see that using 2 out of 3 don't care we have got more minimized equation.

### 5. Simplify the given functions using K- Map Minimization method.

(1) Simplify  $Z = f(A,B) = \bar{A}\bar{B} + A\bar{B} + \bar{A}B$  [CO1-H1].

Consider the expression  $Z = f(A,B) = \bar{A}\bar{B} + A\bar{B} + \bar{A}B$  plotted on the Karnaugh map:



Pairs of 1's are *grouped* as shown above, and the simplified answer is obtained by using the following steps:

Note that two groups can be formed for the example given above, bearing in mind that the largest rectangular clusters that can be made consist of two 1s. Notice that a 1 can belong to more than one group.

The first group labelled I, consists of two 1s which correspond to  $A = 0, B = 0$  and  $A = 1, B = 0$ . Put in another way, all squares in this example that correspond to the area of the map where  $B = 0$  contains 1s, independent of the value of A. So when  $B = 0$  the output is 1. The expression of the output will contain the term  $\bar{B}$ .

For group labelled II corresponds to the area of the map where  $A = 0$ . The group can therefore be defined as  $\bar{A}$ . This implies that when  $A = 0$  the output is 1. The output is therefore 1 whenever  $B=0$  and  $A=0$ . Hence the simplified answer is  $Z = \bar{A} + \bar{B}$ .

6.. Minimize the following problems using the Karnaugh maps method[CO1-H1].

$$Z = f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}B + AB\bar{C} + AC$$

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

$$1. Z = f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}B + AB\bar{C} + AC$$

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

By using the rules of simplification and ringing of adjacent cells in order to make as many variables redundant, the minimised result obtained is  $B + AC + \bar{A}\bar{C}$

$$2. Z = f(A,B,C) = \bar{A}B + B\bar{C} + BC + A\bar{B}\bar{C}$$

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

By using the rules of simplification and ringing of adjacent cells in order to make as many variables redundant, the minimised result obtained is  $B + A\bar{C}$

**7. Explain redundancy in K map. [CO1-L2].**

Redundant groups

A groups of 1s or 0s whose all members are overlapped by other groups is called redundant group. We don't consider this group while writing the simplified equations from the K-map.

zw \ xy	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	1	0	0	0
10	0	0	0	0

In the above K-map the group which is represented by the oval is a redundant group and hence while writing the equations we ignore it or we don't make this kind of group and the K-map representation becomes as given next:

The equation we get is

$$F = yz'w' + x'z'w \text{ (ignoring the redundant group)}$$

If we consider this group then equation would be  $F = yz'w' + x'z'w + x'yz'$

And this is a not the simplified expression and hence WRONG.

K-map without the redundant group is:

zw	00	01	11	10
xy	00	1	0	0
01	1	1	0	0
11	1	0	0	0
10	0	0	0	0

8. Simplify the function using Karnaugh map [CO1-H1].

(i)  $F(ABCD) = \sum (0, 1, 2, 4, 5, 7, 11, 15)$

(ii)  $F(WXYZ) = \sum (2, 3, 10, 11, 12, 13, 14, 15)$

(i)  $F(A,B,C,D) = \sum (0,1,2,4,5,7,11,15)$

1	1		1
1	1	1	
		1	
		1	

$$= \overline{A}\overline{B}\overline{D} + \overline{A}\overline{C} + BCD + ACD$$

(ii)  $F(W,X,Y,Z) = \sum (2,3,10,11,12,13,14,15)$

		1	1

1	1	1	1
		1	1

$$= x'y=wx$$

9.(i) Express the function  $f(x,y,z) = XY + X\bar{Z}$  as a product of sum terms form.

(ii) Express the following function as the minimal sum of products, using a K-map

$$F(a,b,c,d) = \sum m(0,2,4,5,6,8,10,15) + \sum \Phi(7,13,14) \text{ [CO1-H1].}$$

Solution:

$$\begin{aligned}
 \text{(i) } f(x,y,z) &= XY + X\bar{Z} \\
 &= X(Y + \bar{Z}) \\
 &= (X + Y \cdot \bar{Y} + Z \cdot \bar{Z})(X \cdot \bar{X} + Y + \bar{Z}) \\
 &= (X + Y + Z)(X + Y + \bar{Z})(X + \bar{Y} + Z)(X + \bar{Y} + \bar{Z})(X + Y + \bar{Z}) \\
 &= (\bar{X} + Y + \bar{Z}) \\
 &= (X + Y + Z)(X + Y + \bar{Z})(X + \bar{Y} + Z)(X + \bar{Y} + \bar{Z})(\bar{X} + Y + \bar{Z}) \\
 &= \quad M_0 \quad \quad M_1 \quad \quad M_2 \quad \quad M_3 \quad \quad M_5 \\
 &= \pi M (0,1,2,3,5)
 \end{aligned}$$

$$\text{(ii) } F(a,b,c,d) = \sum m(0,2,4,5,6,8,10,15) + \sum \Phi(7,13,14)$$

1			1
1	1	x	1
	x	1	x
1			1

$$= \bar{b}\bar{d} + \bar{a}b + bcd$$



10. Simplify the Boolean function using K-map and tabular methods. Compare the methods.  $F(A, B, C, D) = \sum m(4,5,6,7,8)$   $d(A, B, C, D) = \sum m(11,12,13,14,15)$  Implement using only NAND gates[CO1-H1].

**Solution:**

1	1	1	1
x	x	x	X
1		x	

$$= B + \overline{ACD}$$

**Tabulation method:**

(i) Arranging the minterms according to the number of 1's

Minterm	Binary Representation	Minterm	Binary Representation
$m_4$	0100	$m_4$	0100
$m_5$	0101	$m_8$	1000
$m_6$	0110	$m_5$	0101
$m_7$	0111	$m_6$	0110
$m_8$	1000	$m_{12}$	1100
$m_{11}$	1011	$m_7$	0111
$m_{12}$	1100	$m_{11}$	1011
$m_{13}$	1101	$m_{13}$	1101
$m_{14}$	1110	$m_{14}$	1110
$m_{15}$	1111	$m_{15}$	1111

Elimination of literals:

Minterm	Binary Representation	Minterm	Binary Representation
4,5	010-	4,5,6,7	01--
4,6	01-0	4,6,5,7	01--
4,12	-100	4,12,5,13	-10-
8,12	1-00	5,7,13,15	-1-1
5,7	01-1	5,13,7,15	-1-1
5,13	-101	6,7,14,15	-11-
6,7	011-	6,14,7,15	-11-

6,14	-110		
7,15	-111		
11,15	1-11		
13,15	11-1		
14,15	111-		

Further elimination of literals:

Minterm	Binary Representation
4,5,6,7	01--
5,7,13,15	-1-1
4,5,12,13,6,7,14,15	-1--
8,12	1-00
11,15	1-11

Selecting the minimum number of prime implicants as

Prime Implicants	m <sub>4</sub>	m <sub>5</sub>	m <sub>6</sub>	m <sub>7</sub>	m <sub>8</sub>	m <sub>11</sub>	m <sub>12</sub>	m <sub>13</sub>	m <sub>14</sub>	m <sub>15</sub>
4,5,6,7	*	*	*	*						
5,7,13,15		*		*				*		*
4,5,12,13,6,7,14,15	*	*	*	*			*	*	*	*
8,12					*		*			
11,15						*				*

$$= B + \overline{ACD}$$

11. State and prove demorgan's theorem and expand the function[CO1-H1].

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

**Solution:**

$$\begin{aligned} \overline{((A+B)C + \overline{CD})} &= \overline{(A+B)C} \cdot \overline{\overline{CD}} \\ &= \overline{((A+B) + \overline{C})} \cdot (C + \overline{D}) \\ &= \overline{(\overline{A} \cdot \overline{B}) + \overline{C}} \cdot (C + \overline{D}) \\ &= \overline{(\overline{AC} + \overline{BC})} (C + \overline{D}) \\ &= \overline{\overline{ACC} + \overline{BCC} + \overline{ACD} + \overline{BCD}} \\ &= \overline{\overline{ACD} + \overline{BCD}} \end{aligned}$$

12. Explain the characteristics of CMOS logic[CO1-L2].

- Dissipates low power: The power dissipation is dependent on the power supply voltage, frequency, output load, and input rise time. At 1 MHz and 50 pF load, the power dissipation is typically 10 nW per gate.
- Short propagation delays: Depending on the power supply, the propagation delays are usually around 25 nS to 50 nS.
- Rise and fall times are controlled: The rise and falls are usually ramps instead of step functions, and they are 20 - 40% longer than the propagation delays.
- Noise immunity approaches 50% or 45% of the full logic swing.
- Levels of the logic signal will be essentially equal to the power supplied since the input impedance is so high.
- Voltage levels range from 0 to VDD where VDD is the supply voltage. A low level is anywhere between 0 and 1/3 VDD while a high level is between 2/3 VDD and VDD.

**Characteristics of TTL logic:**

- Power dissipation is usually 10 mW per gate.
- Propagation delays are 10 nS when driving a 15 pF/400 ohm load.
- Voltage levels range from 0 to Vcc where Vcc is typically 4.75V - 5.25V. Voltage range 0V - 0.8V creates logic level 0. Voltage range 2V - Vcc creates logic level 1.

**CMOS compared to TTL:**

- CMOS components are typically more expensive than TTL equivalents. However, CMOS technology is usually less expensive on a system level due to CMOS chips being smaller and requiring less regulation.
- CMOS circuits do not draw as much power as TTL circuits while at rest. However, CMOS power consumption increases faster with higher clock speeds than TTL does. Lower current draw requires less power supply distribution, therefore causing a simpler and cheaper design.
- Due to longer rise and fall times, the transmission of digital signals becomes simpler and less expensive with CMOS chips.
- CMOS components are more susceptible to damage from electrostatic discharge than TTL components.

**13. Explain the characteristics of digital logic families[CO1-L2].**

**Fan-in**

The fan-in of a gate is the number of inputs connected to the gate without any degradation in the voltage levels. For example, an eight-input gate requires one Unit Load(UL) per input. Its fan-in is 8. This parameter determines the functional capabilities of a logic circuit.

**Fan-out :**

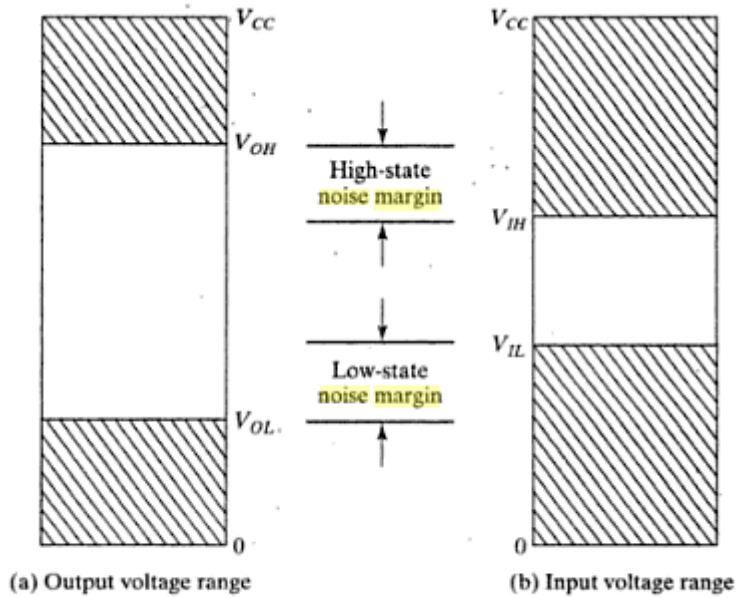
Fan-out is the maximum number of similar logic gates that a gate can drive without any degradation in voltage levels. The fan-out of a gate specifies the number of standard loads that can be connected to the output of the gate without degrading its normal operation. A standard load is usually defined as the amount of current needed by an input of another gate in the same logic family. Fan-out is calculated from the amount of current available in the output of a gate and the amount of current needed in each input of a gate.

**Totem output**

In TTL gate, the transistor Q4 'sits' upon Q3 . Such a configuration is called totem pole and it has low output impedance. Hence output voltage can change quickly from one state to the other due to rapidly charging of stray capacitance. So propagation delay is low in totem-output.

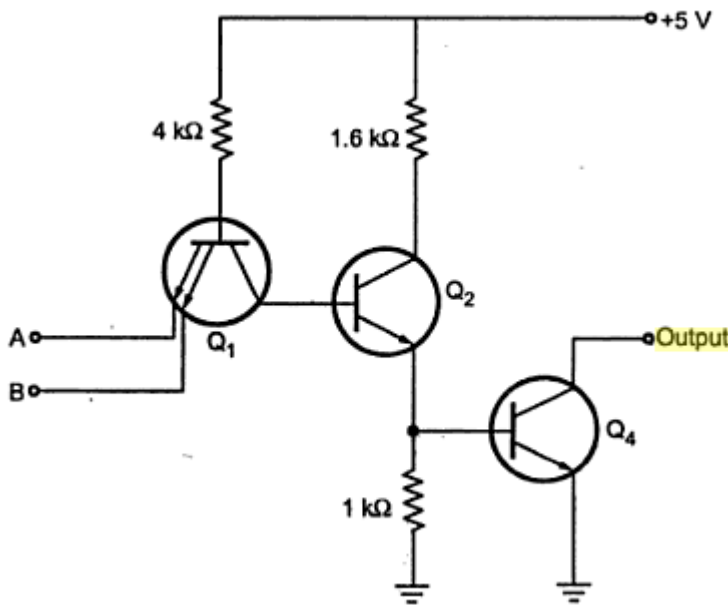
**Noise Margin:**

Noise is a term used to denote an undesirable signal that is superimposed upon the normal operating signal. The noise margin is the maximum noise voltage added to an input signal of a digital circuit that does not cause an undesirable change in the circuit's output. Noise margin is expressed in volts and represents the maximum noise signal that can be tolerated by the gate.



**FIGURE**  
Signals for evaluating noise margin

14.Explain the operation of a two input NAND gate with Open collector output TTL NAND[CO1-L2].



**Fig. Open Collector 2 input TTL NAND gate**

**15. Write short notes on Digital Logic States[CO1-L1].**

The Digital Logic Gate is the basic building block from which all digital electronic circuits and microprocessor based systems are constructed from. Basic digital logic gates perform logical operations of AND, OR and NOT on binary numbers.

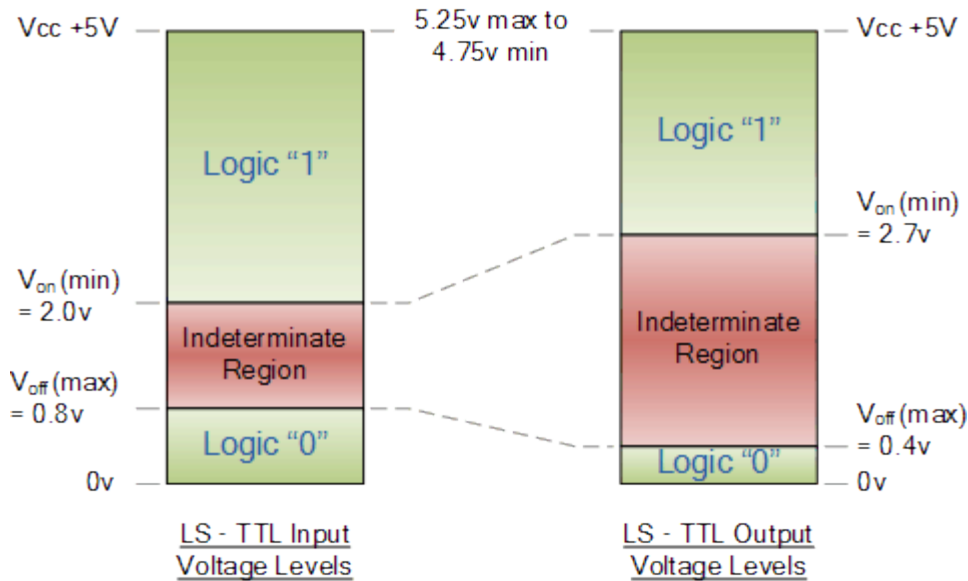
In digital logic design only two voltage levels or states are allowed and these states are generally referred to as Logic "1" and Logic "0", High and Low, or True and False. These two states are represented in [Boolean Algebra](#) and standard truth tables by the binary digits of "1" and "0" respectively. A good example of a digital signal is a simple light as it is either "ON" or "OFF" but not both at the same time.

Most *digital logic gates* and logic systems use "Positive logic", in which a logic level "0" or "LOW" is represented by a zero voltage, 0v or ground and a logic level "1" or "HIGH" is represented by a higher voltage such as +5 volts, with the switching from one voltage level to the other, from either a logic level "0" to a "1" or a "1" to a "0" being made as quickly as possible to prevent any faulty operation of the logic circuit.

There also exists a complementary "Negative Logic" system in which the values and the rules of a logic "0" and a logic "1" are reversed but in this tutorial section about digital logic gates we shall only refer to the positive logic convention as it is the most commonly used.

**16. Explain TTL logic circuit and its Input & Output Voltage Levels. [CO1-L1Nov 2014]**

In standard TTL (transistor-transistor logic) IC's there is a pre-defined voltage range for the input and output voltage levels which define exactly what is a logic "1" level and what is a logic "0" level and these are shown below.



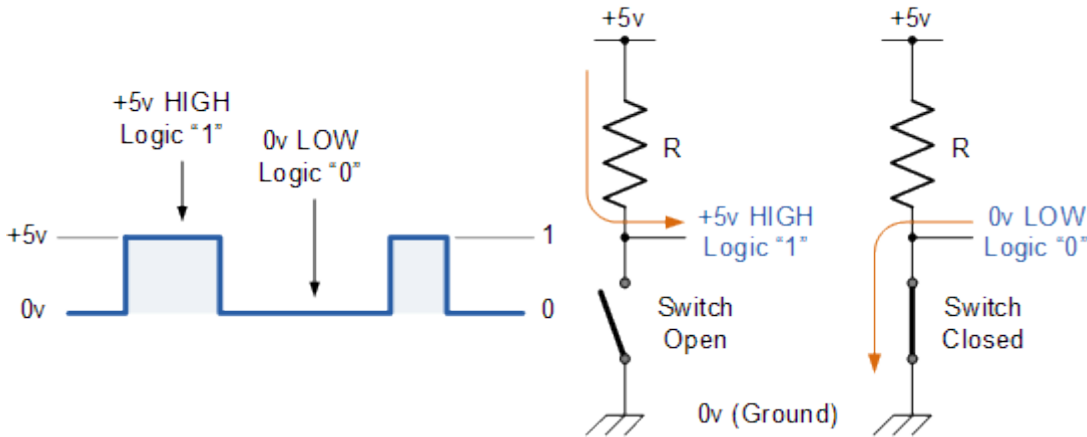
There are a large variety of logic gate types in both the bipolar 7400 and the CMOS 4000 families of digital logic gates such as 74Lxx, 74LSxx, 74ALSxx, 74HCxx, 74HCTxx, 74ACTxx etc, with each one having its own distinct advantages and disadvantages compared to the other. The exact switching voltage required to produce either a logic "0" or a logic "1" depends upon the specific logic group or family.

However, when using a standard +5 volt supply any TTL voltage input between 2.0v and 5v is considered to be a logic "1" or "HIGH" while any voltage input below 0.8v is recognised as a logic "0" or "LOW". The voltage region in between these two voltage levels either as an input or as an output is called the *Indeterminate Region* and operating within this region may cause the logic gate to produce a false output.

The CMOS 4000 logic family uses different levels of voltages compared to the TTL types as they are designed using field effect transistors, or FET's. In CMOS technology a logic "1" level operates between 3.0 and 18 volts and a logic "0" level is below 1.5 volts.

Then from the above observations, we can define the ideal **Digital Logic Gate** as one that has a "LOW" level logic "0" of 0 volts (ground) and a "HIGH" level logic "1" of +5 volts and this can be demonstrated as:

Ideal Digital Logic Voltage Levels



Where the opening or closing of the switch produces either a logic level "1" or a logic level "0" with the resistor R being known as a "pull-up" resistor.

**17. With circuit schematic and explain the operation and characteristics of a ECL gate. [CO1-L1].**

**Emitter Coupled Logic** or **ECL** is another type of digital logic gate that uses bipolar transistor logic where the transistors are not operated in the saturation region, as they are with the standard TTL digital logic gate. Instead the input and output circuits are push-pull connected transistors with the supply voltage negative with respect to ground. This has the effect of increasing the speed of operation of the ECL gates up to the Gigahertz range compared with the standard TTL types, but noise has a greater effect in ECL logic, because the unsaturated transistors operate within their active region and amplify as well as switch signals.

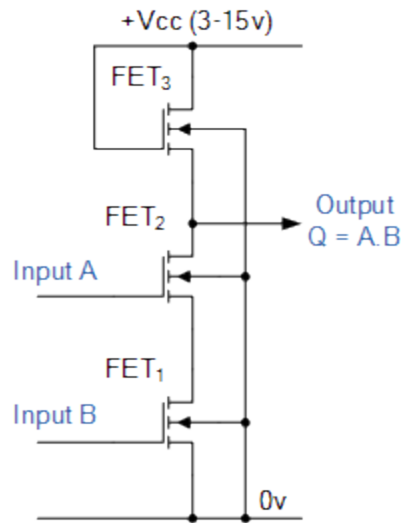
**18. Draw the CMOS logic circuit for NOR gate and explain its operation. [CO1-L1-Nov 2015]**

One of the main disadvantages of the TTL logic series is that the gates are based on bipolar transistor logic technology and as transistors are current operated devices, they consume large amounts of power from a fixed +5 volt power supply. Also, TTL bipolar transistor gates have a limited operating speed when switching from an "OFF" state to an "ON" state and vice-versa called the "gate" or "propagation delay". To overcome these limitations



complementary MOS called "CMOS" logic gates using "Field Effect Transistors" or FET's were developed.

As these gates use both P-channel and N-channel MOSFET's as their input device, at quiescent conditions with no switching, the power consumption of CMOS gates is almost zero, (1 to 2uA) making them ideal for use in low-power battery circuits and with switching speeds upwards of 100MHz for use in high frequency timing and computer circuits.



### 2-input NAND gate

This CMOS gate example contains 3 N-channel MOSFET's, one for each input FET<sub>1</sub> and FET<sub>2</sub> and one for the output FET<sub>3</sub>. When both the inputs A and B are at logic level "0", FET<sub>1</sub> and FET<sub>2</sub> are both switched "OFF" giving an output logic "1" from the source of FET<sub>3</sub>.

When one or both of the inputs are at logic level "1" current flows through the corresponding FET giving an output state at Q equivalent to logic "0", thus producing a NAND gate function.

Improvements in the circuit design with regards to switching speed, low power consumption and improved propagation delays has resulted in the standard CMOS 4000 "CD" family of logic IC's being developed that complement the TTL range.

As with the standard TTL digital logic gates, all the major digital logic gates and devices are available in the CMOS package such as the CD4011, a Quad 2-input NAND gate, or the CD4001, a Quad 2-input NOR gate along with all their sub-families.

Like TTL logic, complementary MOS (CMOS) circuits take advantage of the fact that both N-channel and P-channel devices can be fabricated together on the same substrate material to form various logic functions. One of the main disadvantage with the CMOS range of IC's compared to their equivalent TTL types is that they are easily damaged by static electricity so extra care must be taken when handling these devices. Also unlike TTL logic gates that operate on single +5V voltages for both their input and output levels, CMOS digital logic gates operate on a single supply voltage of between +3 and +18 volts.

In the next tutorial about **Digital Logic Gates**, we will look at the digital Logic [AND Gate](#) function as used in both TTL and CMOS logic circuits as well as its Boolean Algebra definition and truth tables.

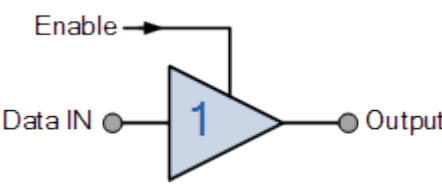
---

### 19. Explain "Tri-state Buffer" logic[CO1-L2].

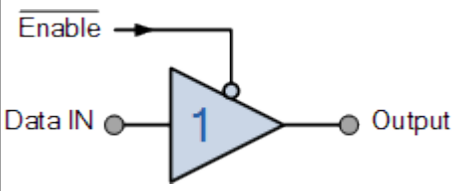
A Tri-state Buffer can be thought of as an input controlled switch which has an output that can be electronically turned "ON" or "OFF" by means of an external "Control" or "Enable" signal input. This control signal can be either a logic "0" or a logic "1" type signal resulting in the Tri-state Buffer being in one state allowing its output to operate normally giving either a logic "0" or logic "1" output.

But when activated in the other state it disables or turns "OFF" its output producing an open circuit condition that is neither "High" or "low", but instead gives an output state of very high impedance, **high-Z**, or more commonly Hi-Z. Then this type of device has two logic state inputs, "0" or a "1" but can produce three different output states, "0", "1" or "Hi-Z" which is why it is called a "3-state" device. There are two different types of Tri-state Buffer, one whose output is controlled by an "**Active-HIGH**" control signal and the other which is controlled by an "**Active-LOW**" control signal, as shown below.

Active "HIGH" Tri-state Buffer

Symbol	Truth Table		
 <p>Tri-state Buffer</p>	Enable	A	Q
	1	0	0
	1	1	1
	0	0	Hi-Z
	0	1	Hi-Z
Read as Output = Input if Enable is equal to "1"			

An **Active-high** Tri-state Buffer is activated when a logic level "1" is applied to its "enable" control line and the data passes through from its input to its output. When the enable control line is at logic level "0", the buffer output is disabled and a high impedance condition, Hi-Z is present on the output. Active "LOW" Tri-state Buffer is shown below.

Symbol	Truth Table		
 <p>Tri-state Buffer</p>	Enable	A	Q
	0	0	0
	0	1	1
	1	0	Hi-Z
	1	1	Hi-Z
Read as Output = Input if Enable is <b>NOT</b> equal to "1"			

An **Active-low** Tri-state Buffer is the opposite to the above, and is activated when a logic level "0" is applied to its "enable" control line. The data passes through from its input to its output. When the enable control line is at logic level "1", the buffer output is disabled and a high impedance condition, Hi-Z is present on the output.

## Unit- II

### Combinational Circuits

#### Part-A

#### **1. How logic circuits of a digital system are classified? [CO2-L1].**

Logic circuits of a digital system are classified into two types as combinational and sequential.

#### **2. Define combinational logic[CO2-H3].**

When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called combinational logic.

#### **3. Write down the design procedure for combinational circuits[CO2-L1].**

- The problem definition
- Determine the number of available input variables & required O/P variables.
- Assigning letter symbols to I/O variables
- Obtain simplified Boolean expression for each O/P.
- Obtain the logic diagram.

#### **4. Define Half adder and full adder[CO2-L1].**

The logic circuit that performs the addition of two bits is a half adder. The circuit that performs the addition of three bits is a full adder.

#### **5. Define Decoder? [CO2-L1].**

A decoder is a multiple - input multiple output logic circuit that converts coded inputs into coded outputs where the input and output codes are different.

#### **6. What is binary decoder? [CO2-L1].**

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

**7. Define Encoder? [CO2-L1].**

An encoder has  $2^n$  input lines and  $n$  output lines. In encoder the output lines generate the binary code corresponding to the input value.

**8. What is priority Encoder? [CO2-L1].**

A priority encoder is an encoder circuit that includes the priority function. In priority encoder, if 2 or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

**9. Define multiplexer and draw the block diagram. [CO2-L1].**

Multiplexer is a digital switch. It allows digital information from several sources to be routed onto a single output line.

**10. What do you mean by comparator? [CO2-L1].**

A comparator is a special combinational circuit designed primarily to compare the relative magnitude of two binary numbers.

**11. What is a demux? [CO2-L1].**

A decoder with an enable input is referred to as a demultiplexer.

**12. What is priority encoder? [CO2-L1].**

A priority encoder is an encoder circuit that includes the priority function. The operation of the 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.

**13. How can a decoder be converted into a demultiplexer? [CO2-L1].**

A decoder can be converted into a demultiplexer by including enable input.

**14. What are tri-state gates? [CO2-L1].**

A three-state gate is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and logic 0 and the third state is a high impedance state.

**15. Distinguish between decoder and demultiplexer. [CO2-L1].**

Decoder	Demultiplexer
A decoder is a combinational circuit that converts binary information from $n$ input lines to a maximum of $2^n$ unique output lines. It is sometimes called as $n$ -to- $m$ -line decoder, where $m \leq 2^n$ .	A demultiplexer is a circuit that receives information from a single line and transmits this information on one $2^n$ possible output lines.

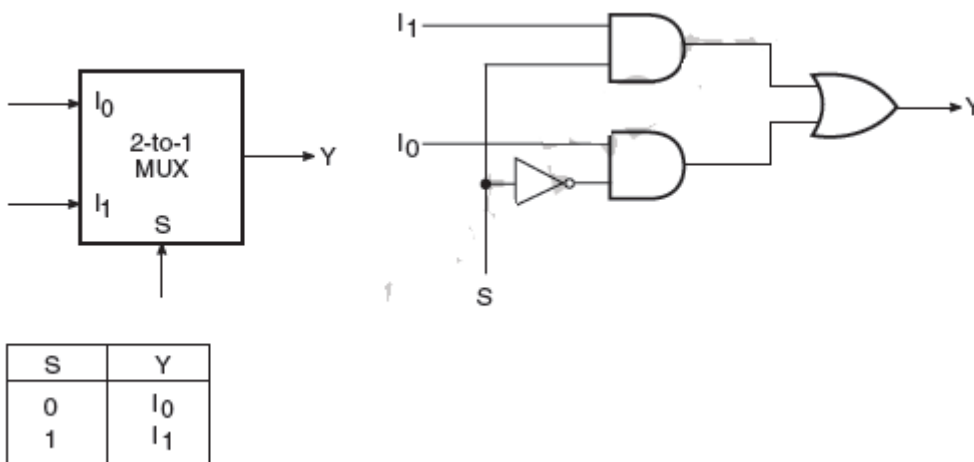
**16. How can a multiplexer be used to convert 8-bit parallel data into serial form? [CO2-L1].**

- A 8-to-1-line multiplexer is used to convert 8-bit parallel data into serial form.
- Each of the eight inputs,  $I_0$  through  $I_7$ , is applied to one input of an AND gate.
- Selection lines  $S_0$ ,  $S_1$ ,  $S_2$ , and  $S_3$  are decoded to select a particular AND gate.
- The outputs of the AND gates are applied to a single OR gate that provides the 1-line (serial form) output.
- Consider the case when the selection lines  $S_2S_1S_0 = 011$ .
- The AND gate associated with input  $I_3$  has three of its inputs equal to 1 and the third input connected to  $I_3$ .
- The other seven AND gates have at least one input equal to 0, which makes their outputs equal to 0.
- The OR output is now equal to the value of  $I_3$ , providing a path from the selected input to the output. Thus it converts 8-bit parallel data into a serial form.

$S_2$	$S_1$	$S_0$	Y
0	0	0	$I_0$
0	0	1	$I_1$
0	1	0	$I_2$
0	1	1	$I_3$

17. Draw a 2 to 1 multiplexer circuit. [CO2-L1].

2 to 1 line Multiplexer



18. What is a combinational circuit? Give an example[CO2-L1].

Combinational logic circuits are circuits in which the output at any time depends upon the combination of the input signals present at that instant only and does not depend upon the past conditions

Example:

- Decoder
- Multiplexer

- Adder
- Subtractor

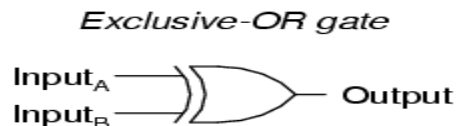
**19. What is meant by multilevel gate network? [CO2-L1].**

The multilevel gate networks are 2 level, 3level, 4level gate network, etc. The maximum number of gates cascaded in series between a network input and the output is referred to as the number of levels of gates

**20. What is look-ahead carry addition?(or) Suggest a solution to overcome the limitation on speed of an adder[CO2-L1]..**

The speed with which an addition is performed is limited by the time required for the carries to propagate or ripple through all of the adder. One method of speeding up the process is by eliminating the ripple carry delay and this is called look-ahead carry addition. This method is based on two functions of the full adder, called the carry generate and carry propagate function.

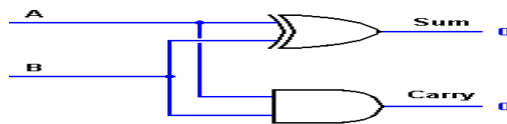
**21. Draw the logic symbol and construct the truth table for two input EXOR gate[CO2-L1]..**



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

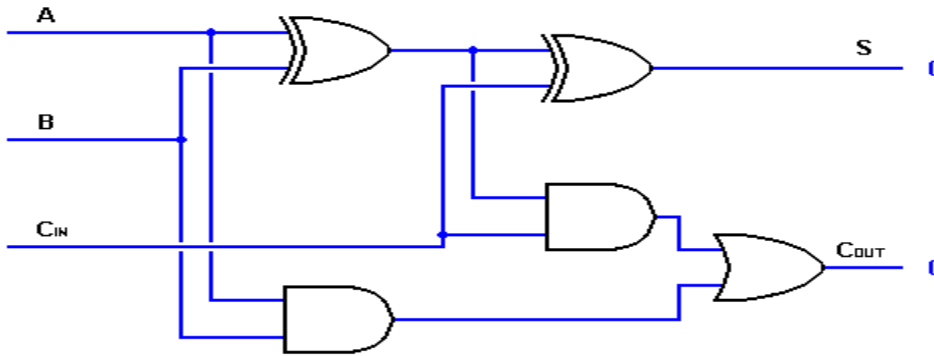
**22. Draw the circuit of half adder and half subtractor[CO2-L1].**

**Half Adder:**

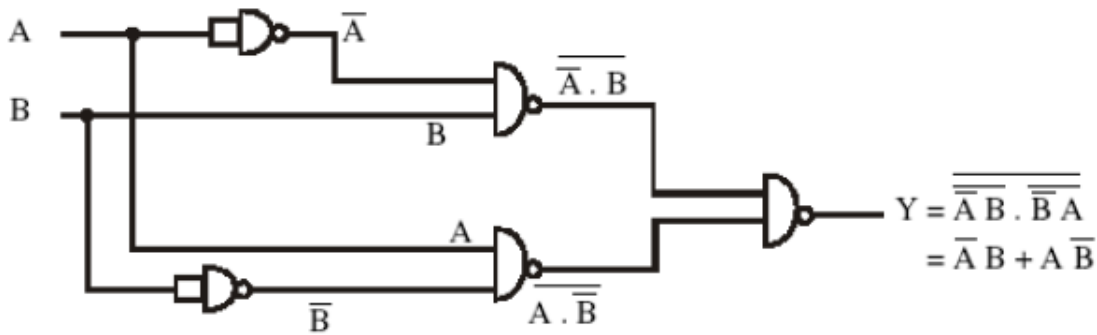




**Full adder:**



**23. Design an EXOR gates only using Nand gates[CO2-L1].**

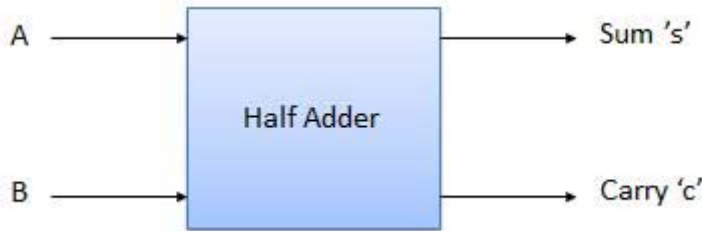


**Part-B**

**1. Explain with a neat sketch Half Adder[CO2-L1].**

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two single bit numbers. This circuit has two outputs carry and sum.

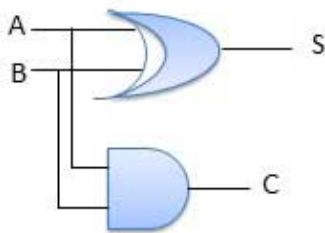
Block diagram



Truth Table

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit Diagram



## 2. Design a full adder using two half adders and an OR gate[CO2-L1].

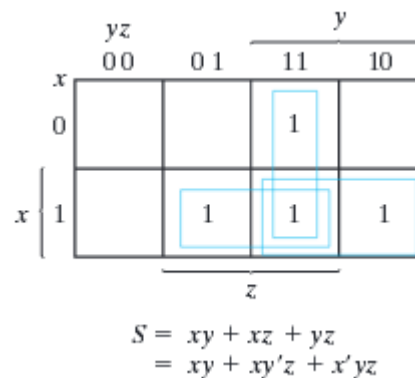
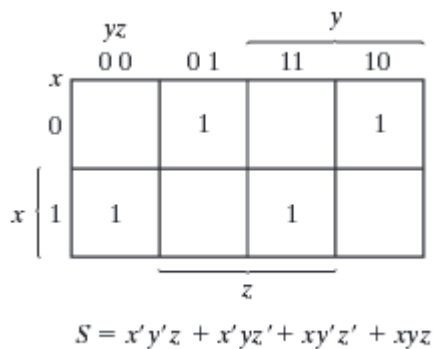
A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by  $x$  and  $y$ , represent the two significant bits to be added. The third input  $z$ , represents the carry from the previous lower significant position.

**Step 1 :** Determine the number of inputs and outputs.

**Step 2 :** Draw the truth table for a full adder with three inputs  $x, y$  and  $z$  and two outputs  $S$  and  $C_{out}$ .

Inputs			Outputs	
x	y	z	S <sub>um</sub>	C <sub>out</sub>
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

**Step 3 :** Obtain the K map Simplification for Sum and Carry Out



**Step 4 :** Implementation

A full adder can be implemented with two half adders and one OR gate as shown in the Fig. The S output from the second half-adder is the exclusive-OR of Z and the output of the first half adder, giving

$$\begin{aligned}
 S &= z \oplus (x \oplus y) \\
 &= z'(xy' + x'y) + z(xy' + x'y)' \\
 &= z'(xy' + x'y) + z(xy + x'y') \\
 &= xy'z' + x'yz' + xyz + x'y'z
 \end{aligned}$$

and the carry output is

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

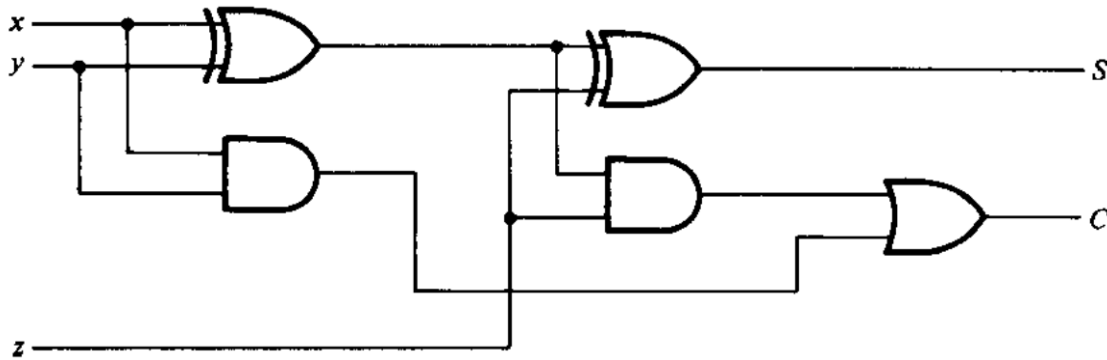
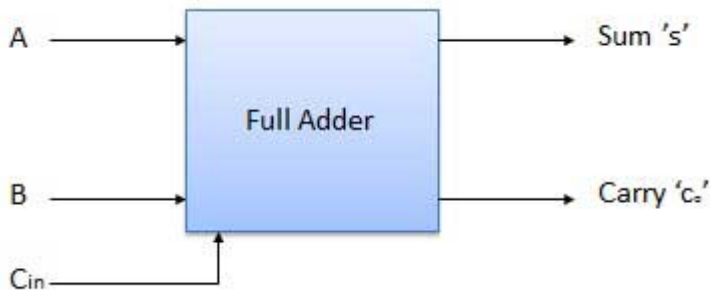


Fig. Implementation of a full adder with two Half adders and an OR gate

**3. Explain Full Adder with neat sketch[CO2-L2].**

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

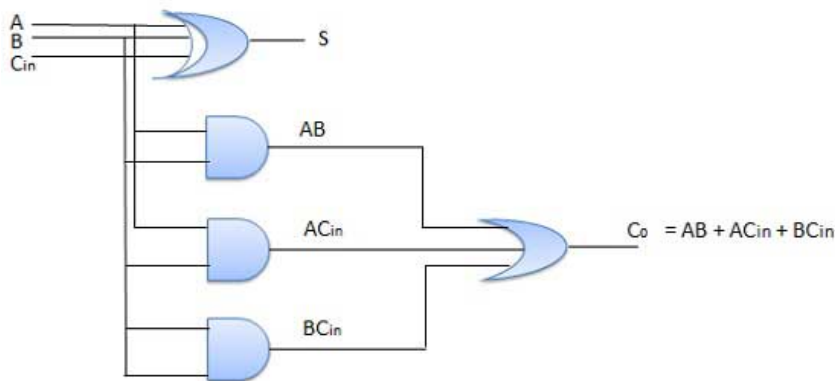
Block diagram



Truth Table

Inputs			Output	
A	B	C <sub>in</sub>	S	Co
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

Circuit Diagram



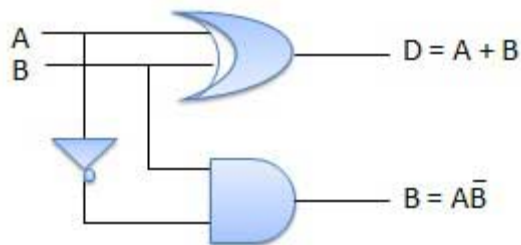
**4. Design a Half Subtractor and explain its operation [CO2-H3].**

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

Truth Table

Inputs		Output	
A	B	(A - B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit Diagram



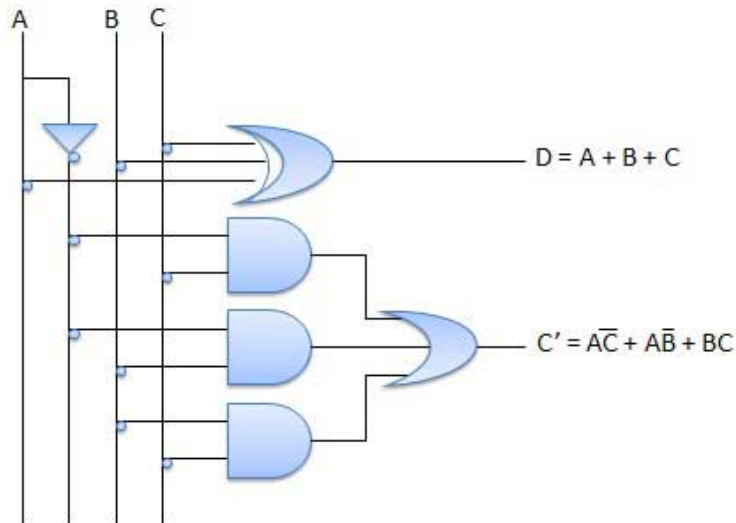
### 5. Design a Full Subtractor circuit and explain its operation[CO2-H3].

The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

Truth Table

Inputs			Output	
A	B	C	(A-B-C)	C'
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

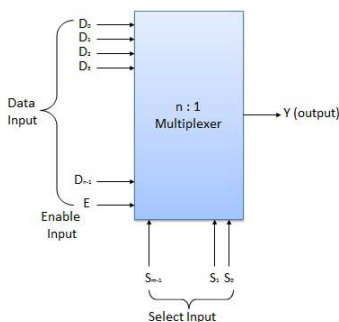
Circuit Diagram



### 6. Explain the operation of Multiplexer circuit[CO2-L2].

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with  $2^m = n$ . It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

Block diagram

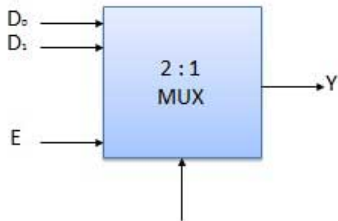


Multiplexers come in multiple variations

- 2 : 1 multiplexer

- 4 : 1 multiplexer
- 16 : 1 multiplexer
- 32 : 1 multiplexer

Block Diagram



Truth Table

Enable	Select	Output
E	S	Y
0	x	0
1	0	D <sub>0</sub>
1	1	D <sub>1</sub>

x = Don't care

**7. Implement the given Boolean function using 8: 1 multiplexer[CO2-L1].**

$$F(A, B, C) = \sum(1, 3, 5, 6)$$

Step 1 : The given function can be implemented using a 4 x1 multiplexer

B and C inputs are connected to the Selection lines S<sub>1</sub> and S<sub>0</sub> respectively.

Figures (a) shows the multiplexer implementation

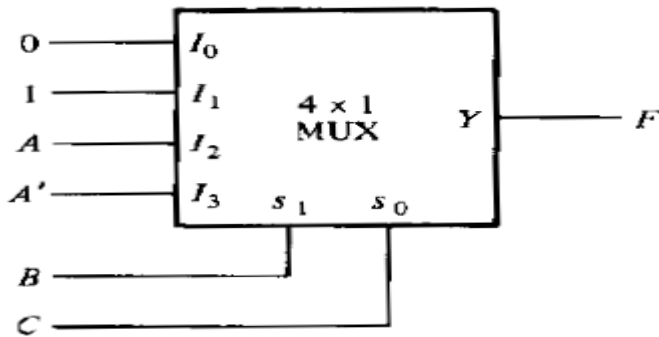


Fig (a) Multiplexer implementation



Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Fig (b) Truth table

### 8. Explain Demultiplexer[CO2-L2].

A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

	$I_0$	$I_1$	$I_2$	$I_3$
$A'$	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	$A'$

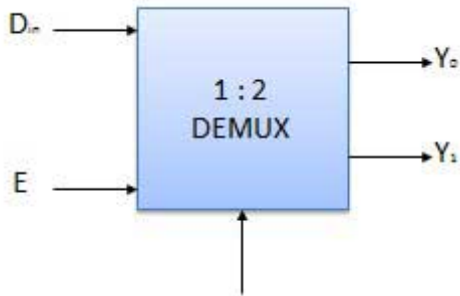
(c) Implementation table

Demultiplexers comes in multiple variations.

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 16 demultiplexer

- 1 : 32 demultiplexer

Block diagram



Truth Table

Enable	Select	Output	
E	S	Y <sub>0</sub>	Y <sub>1</sub>
0	x	0	0
1	0	0	D <sub>in</sub>
1	1	D <sub>in</sub>	0

x = Don't care

### 9. Implement the full subtractor using demultiplexer[CO2-H1].

Step 1 : Write the truth table of full subtractor.

Step 2 : Represent output of full subtractor in minterm form.

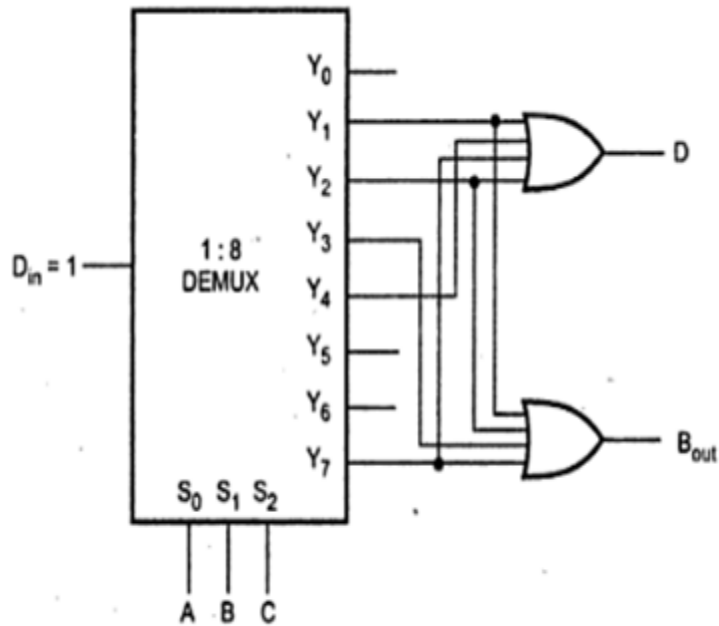
For full subtractor, difference D function can be written as  $D = f(A,B,C) = \sum m = (1,2,4,7)$  and the Borrow out can be written as  $B_{out} = f(A,B,C) = \sum m = (1,2,3,7)$

Step 3 : Logically OR the outputs corresponding to minterms.

With D<sub>0</sub> input 1, demultiplexer gives minterms at the output so by logically ORing the required minterms we can implement Boolean functions for full subtractor.

**Truth table of Full Subtractor :**

Inputs			Outputs	
A	B	C	D	B <sub>out</sub>
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
	1	1	1	1



**Fig. Full subtractor using 1 : 8 Demux**

**10. Explain Decoders and its types. [CO2-L2].**

A decoder is a combinational circuit. It has n input and to a maximum m = 2<sup>n</sup> outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

Block diagram



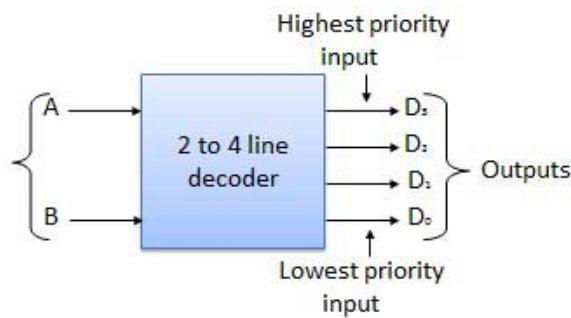
Examples of Decoders are following.

- Code converters
- BCD to seven segment decoders
- Nixie tube decoders
- Relay actuator

### 2 to 4 Line Decoder

The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where  $D_0$  through  $D_3$  are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for only a specific combination of inputs.

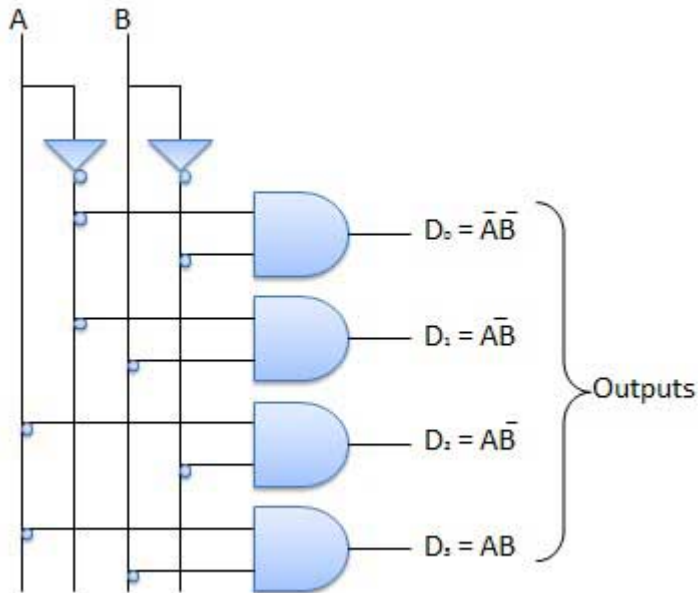
#### Block diagram



#### Truth Table

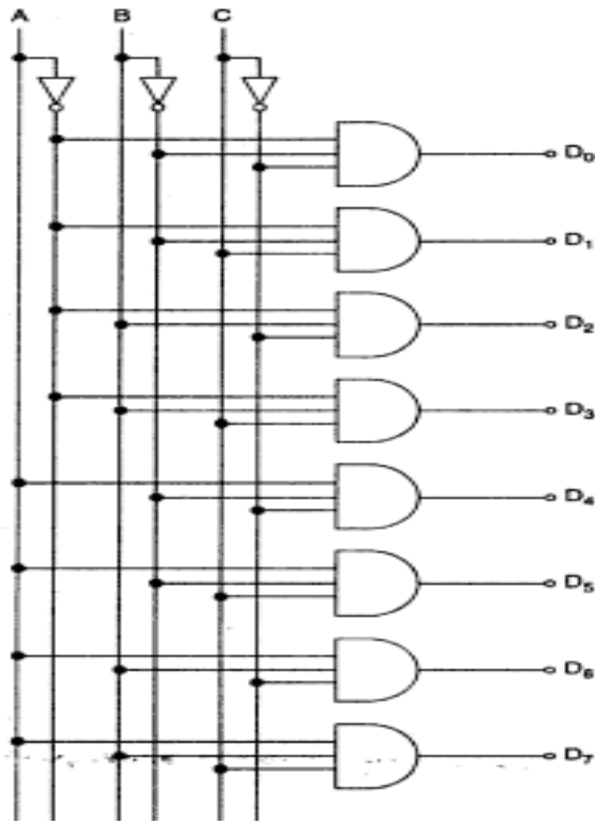
Inputs		Output			
A	B	$D_0$	$D_1$	$D_2$	$D_3$
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

## Logic Circuit

**11. Design a 3:8 decoder using basic gates. [CO2-H3].**

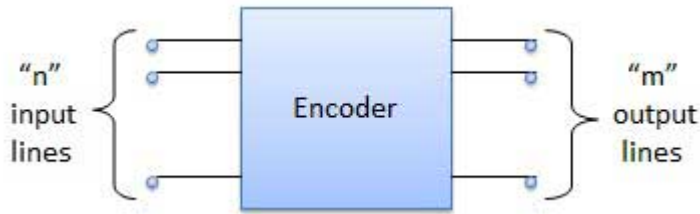
A 3 to 8 decoder has three inputs(A,B,C) and eight outputs ( $D_0$  to  $D_7$ ). Based on the three inputs one of the eight outputs is selected. The Truth table for 3 to 8 Decoder is given below :

Inputs			Outputs							
A	B	C	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$	$D_7$
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

**Logic diagram of 3 to 8 Decoder:****12. Explain Encoders. [CO2-L2].**

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has  $n$  number of input lines and  $m$  number of output lines. An encoder produces an  $m$  bit binary code corresponding to the digital input number. The encoder accepts an  $n$  input digital word and converts it into an  $m$  bit another digital word.

Block diagram



Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

**13. What is a code converter? State its types. [CO2-L1].**

- A code converter circuit will convert coded information in one form to a different coding form.

Types of code converters ;

- Binary to Gray code converter
- Gray to Binary code converter
- BCD to Excess 3 code converter
- Excess 3 to BCD converter

**14. Design a binary to gray code converter. [CO2-H3].**

Step 1 : Form a truth table relating Binary code as input and Gray code as output.

Decimal	Binary Code				Gray Code			
	D	C	B	A	G <sub>3</sub>	G <sub>2</sub>	G <sub>1</sub>	G <sub>0</sub>
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0

4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Step 2 : Perform K map simplification for each gray code output.

**For  $G_0$**

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G_0 = \overline{B}A + B\overline{A}$$

$$= B \oplus A$$

**For  $G_1$**

DC \ BA	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$G_1 = C\overline{B} + \overline{C}B$$

$$= C \oplus B$$

**For  $G_2$**

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$G_2 = \overline{D}C + D\overline{C}$$

$$= D \oplus C$$

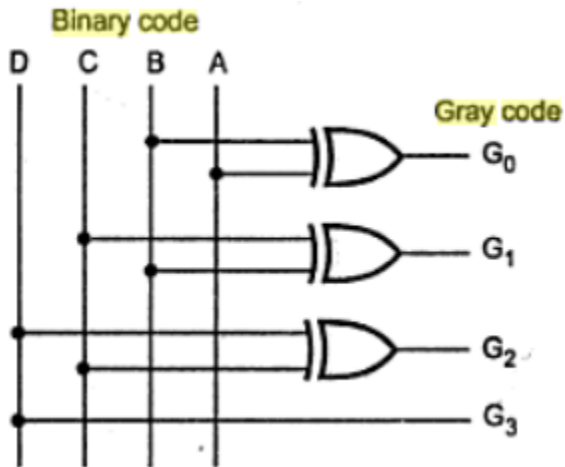
**For  $G_3$**

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$G_3 = D$$



Step 3 : Realization of code converter using XOR gate.

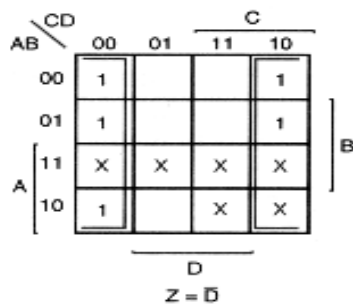
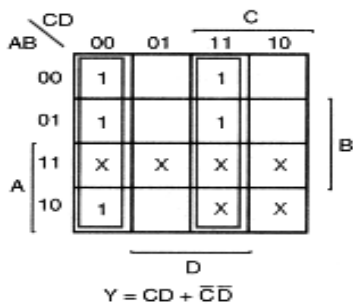
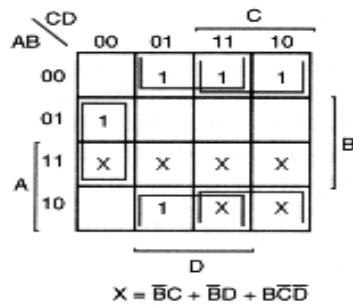
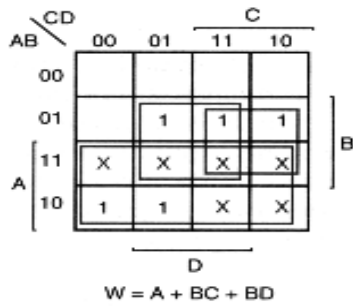


15. Design a four bit BCD to excess-3 code converter. Draw the logic diagram. [CO2-H3].

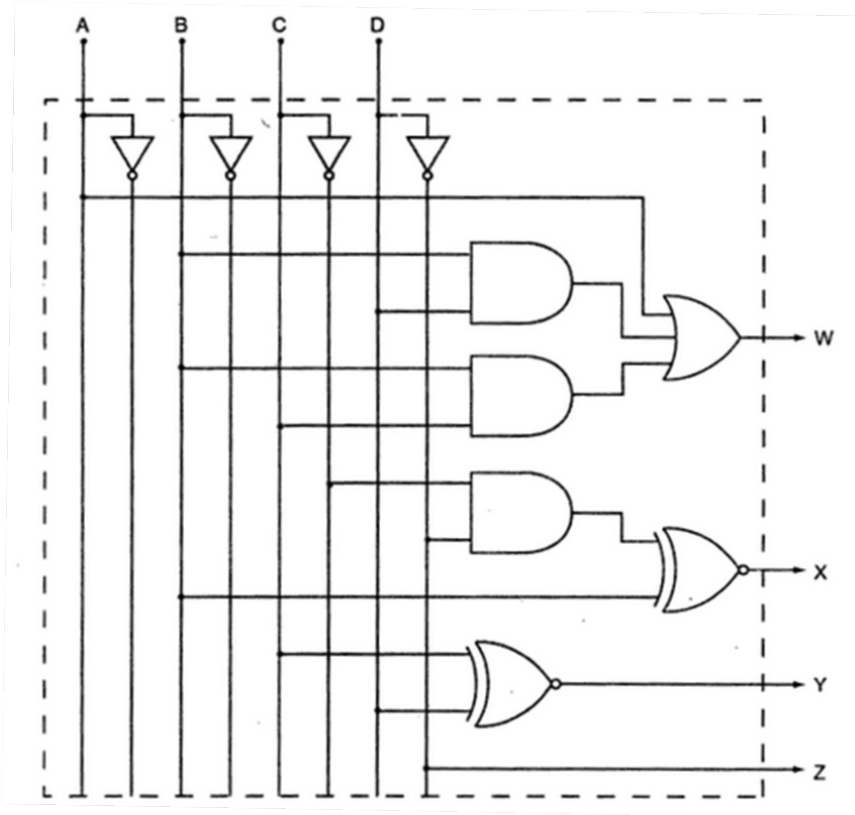
Step 1 : Form a truth table relating BCD code and Excess-3 code.

Decimal number	Inputs				Outputs			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	d	d	d	d
11	1	0	1	1	d	d	d	d
12	1	1	0	0	d	d	d	d
13	1	1	0	1	d	d	d	d
14	1	1	1	0	d	d	d	d
15	1	1	1	1	d	d	d	d

Step 2 : K map simplification for each Excess-3 code output



Step 3 : Realization of Code converter



## Unit -III

### Sequential Circuits

#### Part-A

#### **1. What is the classification of sequential circuits? [CO3-L1].**

The sequential circuits are classified on the basis of timing of their signals into two types.

They are,

- 1) Synchronous sequential circuit.
- 2) Asynchronous sequential circuit.

#### **2. Define Flip flop. ? [CO3-L1].**

The basic unit for storage is flip flop. A flip-flop maintains its output state either at 1 or 0 until directed by an input signal to change its state.

#### **3. What are the different types of flip-flop? ? [CO3-L1].**

There are various types of flip flops. Some of them are mentioned below they are,

- RS flip-flop
- SR flip-flop
- D flip-flop
- JK flip-flop
- T flip-flop

#### **4. Define race around condition. [CO3-L1].**

In JK flip-flop output is fed back to the input. Therefore change in the output results change in the input. Due to this in the positive half of the clock pulse if both J and K are high then output toggles continuously. This condition is called 'race around condition'.

#### **5. What is edge-triggered flip-flop? [CO3-L1].**

The problem of race around condition can be solved by edge triggering flip flop. The term edge triggering means that the flip-flop changes state either at the positive edge or negative edge of the clock pulse and it is sensitive to its inputs only at this transition of the clock.

#### **6. What is a master-slave flip-flop? [CO3-L1]**

A master-slave flip-flop consists of two flip-flops where one circuit serves as a master and the other as a slave.

#### **7. Define rise time & fall time . [CO3-L1]**

The time required to change the voltage level from 10% to 90% is known as rise time ( $t_r$ ).

The time required to change the voltage level from 90% to 10% is known as fall time ( $t_f$ ).

#### **8. Define skew and clock skew. [CO3-L1]**

The phase shift between the rectangular clock waveforms is referred to as skew and the time delay between the two clock pulses is called clock skew.

#### **9. Define setup time. [CO3-L1]**

The setup time is the minimum time required to maintain a constant voltage level at the excitation inputs of the flip-flop device prior to the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip flop. It is denoted as  $t_{\text{setup}}$ .

#### **10. Define hold time. [CO3-L1]**

The hold time is the minimum time for which the voltage levels at the excitation inputs must remain constant after the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip flop. It is denoted as  $t_{\text{hold}}$ .

#### **11. Define propagation delay. [CO3-L1]**

A propagation delay is the time required to change the output after the application of the input. It is denoted as  $t_{\text{pd}}$ .

#### **12. Define registers. [CO3-L1]**

A register is a group of flip-flops flip-flop can store one bit information. So an n-bit register has a group of n flip-flops and is capable of storing any binary information/number containing n-bits.

**13. Define shift registers.**

The binary information in a register can be moved from stage to stage within the register or into or out of the register upon application of clock pulses. This type of bit movement or shifting is essential for certain arithmetic and logic operations used in microprocessors. This gives rise to group of registers called shift registers.

**14. Define sequential circuit?**

In sequential circuits the output variables dependent not only on the present input variables but they also depend up on the past history of these input variables.

**15. Give the comparison between combinational circuits and sequential circuits.**

Combinational circuits	Sequential circuits
Memory unit is not required	Memory unity is required
Parallel adder is a combinational circuit	Serial adder is a sequential circuit

**16. What do you mean by present state?**

The information stored in the memory elements at any given time defines the present state of the sequential circuit.

**17. What do you mean by next state?**

The present state and the external inputs determine the outputs and the next state of the sequential circuit.

**18. State the types of sequential circuits?**

1. Synchronous sequential circuits
2. Asynchronous sequential circuits

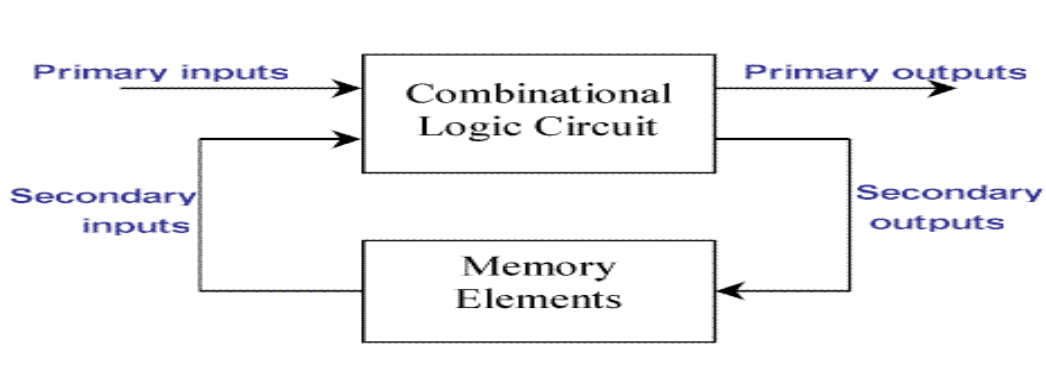
**19. Define synchronous sequential circuit**

In synchronous sequential circuits, signals can affect the memory elements only at discrete instant of time.

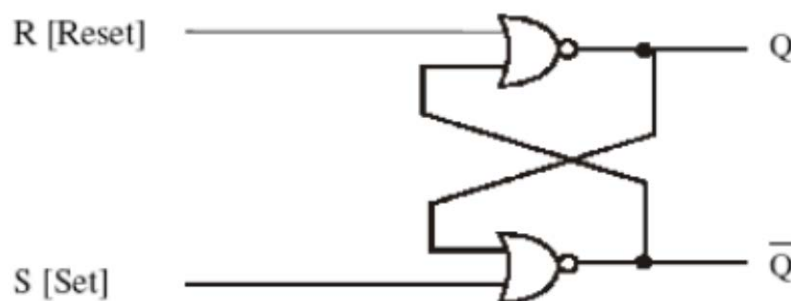
**20. Define Asynchronous sequential circuit. [CO3-L1]**

In asynchronous sequential circuits change in input signals can affect memory element at any instant of time.

**21. Draw the Block diagram of sequential circuit. [CO3-L1]**



**22. Draw the logic diagram for SR latch using two NOR gates. [CO3-L1]**



**23. What are the types of shift register?(or) Classify the registers with respect to serial in parallel input output? [CO3-L1]**

1. Serial in serial out shift register

2. Serial in parallel out shift register
3. Parallel in serial out shift register
4. Parallel in parallel out shift register
5. Bidirectional shift register shift register

**24. State the types of counter? [CO3-L1]**

1. Synchronous counter
2. Asynchronous Counter

**25. The  $t_{pd}$  for each flip-flop is 50 ns. Determine the maximum operating frequency for MOD -32 ripple counter [CO3-L1]**

$$f_{\max(\text{ripple})} = 5 \times 50 \text{ ns} = 4 \text{ MHz}$$

**26. What is the function of Universal shift Register? [CO3-L1]**

The function of shift register is to shift the data in both directions and also to accept the data in parallel.

**27. How many Flip flops are required to design mod-10 counter? [CO3-L1]**

The number of flip flops required to design mod 10 counter is 4.

$$(\text{Since } 2^4=16)$$

**28. Give the excitation for JK flipflop. [CO3-L1]**

$Q$	$Q_{next}$	$J$	$K$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

**29. What is Priority Encoder? [CO3-L1]**

The priority encoder is an encoder that includes priority function. In the priority encoder if two or more inputs are equal to 1 at the same time, the inputs having the highest priority will take precedence.



**30. Write down the characteristic equation of JK Flipflop. [CO3-L1]**

The characteristic equation of JK Flipflop is

$$Q_{n+1} = J + KQ$$

**31. What is meant by programmable counter? Mention its application. [CO3-L1]**

A counter that divides the input frequency by a number which can be programmed, is called programmable counter.

**Applications:**

1. Frequency division.
2. Digital Clock.
3. Stop Watch.
4. Programmable Logic Counters.

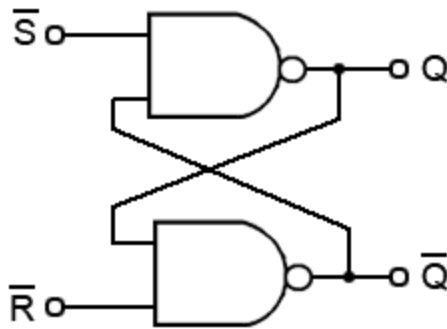
**32. Write down the difference between Moore and Mealy model. [CO3-L1]**

S.No	Moore Model	Mealy Model
1	Its output is a function of present state only.	Its output is a function of present state as well as present input.
2	Input changes do not affect the output.	Input changes may affect the output of the circuit.
3	Moore model requires more number of states for implementing the same function.	It requires less number of states for implementing the same function.

**Part-B****1. Explain the circuit of a SR flip-flop and explain its operation. [CO3-L2]**

The SR (Set-Reset) flip-flop is one of the simplest sequential circuits and consists of two gates connected as shown in Fig. 5.2.1. Notice that the output of each gate is connected to one of the inputs of the other gate, giving a form of positive feedback or 'cross-coupling'.

The circuit has two active low inputs marked S and R, 'NOT' being indicated by the bar above the letter, as well as two outputs, Q and  $\bar{Q}$ . Table shows what happens to the Q and  $\bar{Q}$  outputs when a logic 0 is applied to either the S or R inputs.



	$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	Comments
1.	0	1	1	0	Q is set to 1 by 0 on $\bar{S}$
2.	1	1	1	0	No change, (1 on Q is remembered)
3.	1	0	0	1	Q is reset to 0 by 0 on $\bar{R}$
4.	1	1	0	1	No change, (0 on Q is remembered)
5.	0	0	1	1	Both inputs at 0 – both outputs are at 1 (Non-allowed state)
6.	1	1	?	?	Inputs change from 0,0 to 1,1 together - outputs will be INDETERMINATE

**The SR Flip-flop Truth Table**

- Q output is set to logic 1 by applying logic 0 to the S input.

2. Returning the S input to logic 1 has no effect. The 0 pulse (high-low-high) has been 'remembered' by the Q.
3. Q is reset to 0 by logic 0 applied to the R input.
4. As R returns to logic 1 the 0 on Q is 'remembered' by Q.

From the diagram it is evident that the flip flop has mainly four states. They are

$S=1, R=0 \rightarrow Q=1, Q'=0$

This state is also called the SET state.

$S=0, R=1 \rightarrow Q=0, Q'=1$

This state is known as the RESET state.

In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the value of S.

$S=0, R=0 \rightarrow Q \text{ \& } Q' = \text{Remember}$

If both the values of S and R are switched to 0, then the circuit remembers the value of S and R in their previous state.

$S=1, R=1 \rightarrow Q=0, Q'=0$  [Invalid]

This is an invalid state because the values of both Q and Q' are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

### 1. Explain the operation of a master slave JK flip flop. [CO3-L1]

- The SR flip-flop can be modified to a JK flip-flop to eliminate the undesirable condition that leads to undefined outputs and indeterminate behavior.
- A Master-Slave JK Flip-Flop is shown in the Figure .
- Here, the output gets complemented when both J and K inputs are high.

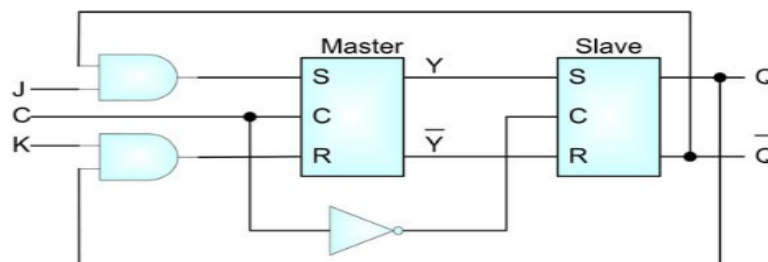
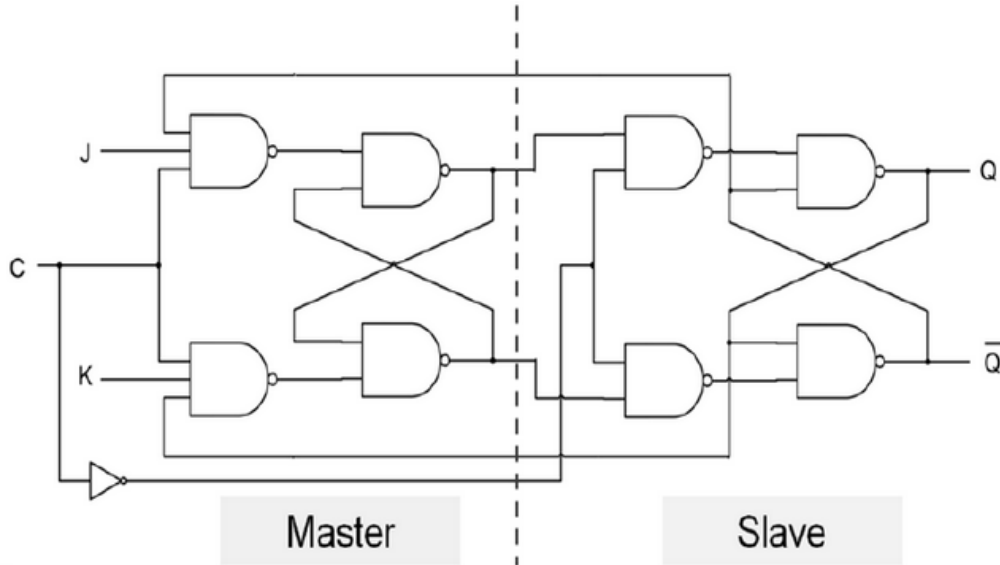


Figure : JK Master-Slave Flip-Flop



**2. Realize T flip-flop using JK flip-flop. [CO3-H3]**

Realization of JK flip-flop using T flip-flop is illustrated below.

J and K are the actual inputs of the flip flop and T is taken as the external input for conversion. Four combinations are produced with T and Qp. J and K are expressed in terms of T and Qp. The conversion table, K-maps, and the logic diagram are given below.

J-K Flip Flop to T Flip Flop

T Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps

Qp	0	1
T=0	0	X
T=1	1	X

Qp	0	1
T=0	X	0
T=1	X	1

Logic Diagram

[www.CircuitsToday.com](http://www.CircuitsToday.com)

**3. Explain the various types of triggering with suitable diagrams. Compare their merits and demerits.(8) (Nov 2014) [CO3-L1]**

**Level Trigger:**

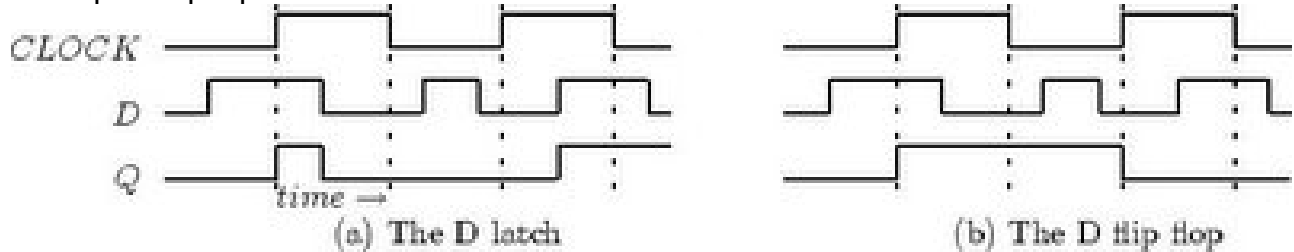
- 1) The input signal is sampled when the clock signal is either HIGH or LOW.
- 2) It is sensitive to Glitches.

Example: Latch.

### Edge Trigger:

- 1) The input signal is sampled at the RISING EDGE or FALLING EDGE of the clock signal.
- 2) It is not-sensitive to Glitches.

Example: Flipflop.



### Asynchronous Type

1. Design an asynchronous Module-8 Down counter using JK flipflops.

### Synchronous Type

4. Design synchronous sequential circuit that goes through the count sequence 1,3,4,5 repeatedly. Use T flip-flops for your design.

### Flip-flops – SR,JK, T and D

5. Explain the circuit of a SR flip-flop and explain its operation.
6. Explain the operation of a master slave JK flip flop.
7. Realize T flip-flop using JK flip-flop.
8. Explain the operation of a JK master slave flip flop.

### Level triggering and edge triggering

9. Explain the various types of triggering with suitable diagrams. Compare their merits and demerits.

### Shift Registers

10. Design a 3-bit bidirectional shift register.

### 11. Counter - Sequence Detector

12. Design a MOD-5 synchronous counter using JK flip-flops.
13. Design a sequence detector to detect the sequence 101 using JK flipflop.

14. Design a synchronous decade counter using T flip flop and construct the timing diagram.
15. Design a mealy model of sequence detector to detect the pattern 1001.
16. Design a MOD-5 counter using T flip flops

#### **Design of synchronous sequential circuits**

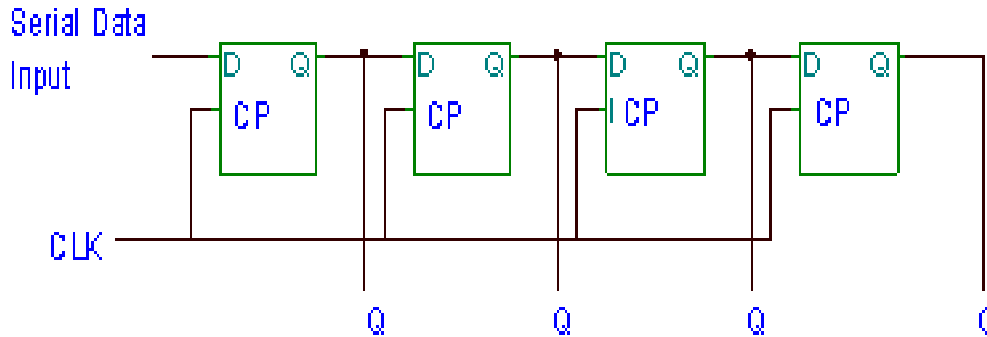
17. A sequential circuit with two D flip flops A and B, input X and output Y is specified by the following next state and output equations  
 $A(t+1) = AX + BX$  ;  $B(t+1) = A'X$  ;  $Y=(A+B)X'$   
Draw the logic diagram, derive state table and state diagram.

#### **Mealy Model**

18. Design a serial adder using Mealy state model.
19. **State Diagram-State Reduction-State Assignment.**
21. Explain the state minimization using partitioning procedure with a suitable diagram.

#### **5. Draw a 4 bit serial in parallel out (SIPO) and explain its operation. [CO3-L1]**

- The second type of register is one in which data is shifted in serially, but shifted out in parallel.
- In order to shift the data out in parallel it is simply necessary to have all the data available as outputs at the same time.
- This is done by connecting the output of each flip flop to an output pin.

**A Serial In Parallel Out shift register:**

- Input data is applied one bit at a time to the D input of the first flip flop in a chain and read out from the Q outputs in parallel after a data word is all shifted in.

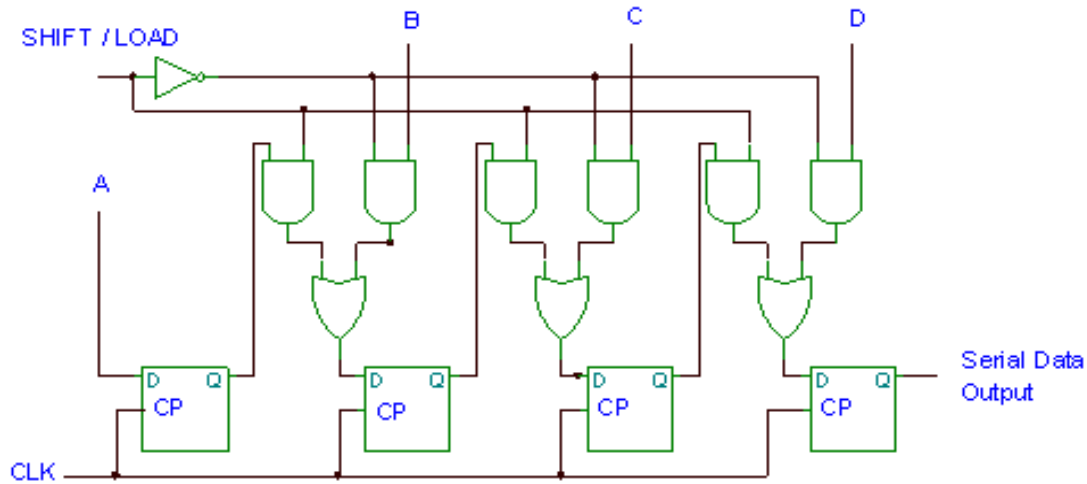
**6. Draw a 4 bit parallel in serial out shift register and explain briefly. Or Explain the parallel load serial out data transfer operation in a 4-bit shift register. [CO3-L1].**

**Parallel In Serial Out:**

- In a parallel in serial out shift registers, the bits are entered simultaneously into their respective flip flops.

**Circuit diagram:**

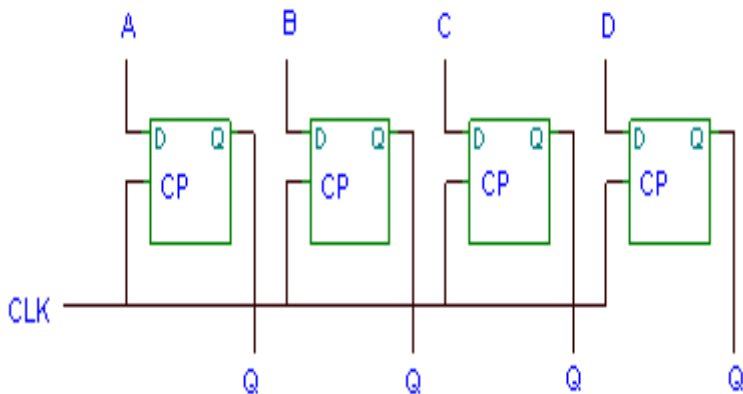
- It has four input data lines A, B, C and D and a shift / LOAD' input that allows four bits data to be entered into the shift register in a parallel fashion.
- When shift/LOAD' is low, gates G1, G2 and G3 are enabled, allowing each data bit applied to the D input of the respective flip flops.
- When a clock pulse is obtained, the flip flops store all bits simultaneously.
- When shift/LOAD' is high, gates G4, G5 and G6 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data entry operation, depending on which AND gates are enabled by the shift/LAOD' control signal.



**7. Explain in detail PIPO (Parallel In Parallel Out) shift register. [CO3-L2].**

- In PIPO bits are enabled simultaneously into their respective flip flops and the Q output are read out in parallel.
- This type of register is simply used to store data, and is sometimes called a data register or data latch.

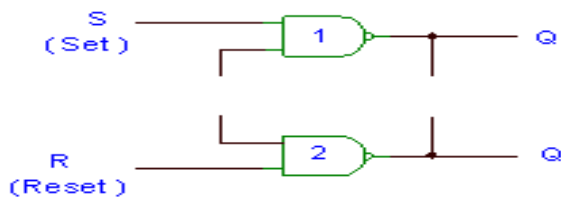
**Logic Diagram:**





**8. Realize a SR flip-flop using NAND gates and explain its operation. [CO3-L1]**

- The inputs signal for NAND circuit requires the complement of values used for NOR latch. Therefore it is referred to as S'-R' latch.
- When the input of NAND gate is S=1 and R=1 the output doesn't change. It remains in the previous state (No Change).
- When the input of NAND gate is S=0 and R=0 the output of NAND gates becomes HIGH which is defined.

**Logic Diagram:****Case (i): S = R = 0**

- The logical operation of NAND gate is such that when any one of the input is low(0), the output will become high. Therefore when the input S=0 R=0 is given, the output of both gates becomes high which is an undefined state.

**Case (ii) S=0 and R=1**

- When the input S=0 and R=1 is given, the output of NAND gate 1, i.e. is Q becomes high, which is fed back to the input of NAND gate 2 (with R=1) the output of Q' becomes 0. The circuit is said to be in the set state (Q = 1 and Q' = 0).

**Case (iii) S=1 and R= 0**

- When the input S=1 and R=0 is given, the output of NAND gate 2, i.e. is Q' becomes high, which is fed back to the input of NAND gate 1 (with R=0) the output

NAND gate becomes 0. The circuit is said to be in the reset state ( $Q=0$  and  $Q' = 1$ ).

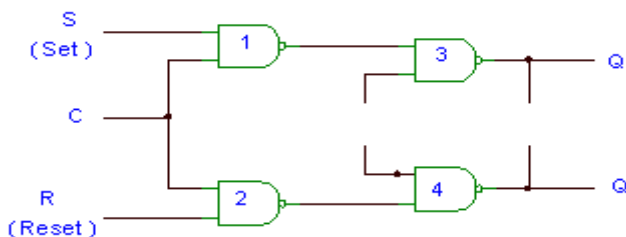
#### Case (iv) $S=1$ and $R=1$ (No change)

- When  $S=R=1$ , the next state doesn't change, it remains in the previous state.
- When the input,  $S = 1$  and  $R = 1$  is applied, assume  $Q = 0$  and  $Q'=1$ , the output of NAND 2 becomes high since its input,  $R=1$  and  $Q=0$ . The output of NAND gate 1 becomes 1, since its input  $S=1$  and  $Q=0$ . Thus the circuit remains in the reset state.
- Similarly when the input  $S=1$  and  $R=1$  is applied with  $Q=1$  and  $Q'=0$ (set state), output of NAND 1 becomes high, since its input are  $S = 1$  and  $Q' = 0$ . The output of NAND gate 2 becomes 0, since its input  $R=1$  and  $Q=1$ . Thus the circuit remains in the set state.

#### SR latch with Control Input:

- SR latch with control input determines when the state of the latch can be changed.
- It consists of the basic SR latch and two additional NAND gates.
- The control input C acts as an enable signal for the other two inputs.
- The output of the NAND gates stays at the logic 1 level as long as the control input remains at 0. This is the quiescent condition for the SR latch.

#### Circuit diagram:



- When the control input goes to 1, information from the S or R input is allowed to effect the SR latch.

C	S	R	NextState of Q
0	X	x	No change –Quiescent condition
1	0	0	No change
1	0	1	Q = 0, ResetState
1	1	0	Q = 1, Set state
1	1	1	Indeterminate State

- To change to the reset state, the inputs must be S=0, R=1 and C=1. In either case, when C returns to 0, the circuit remains in it's current state. Control input disables the circuit by applying 0 to C, so that the state of the output doesn't change regardless of the values of S and R.
- When C=1 and both the S and R inputs are equal to 0, the state of the circuit doesn't change.
- An indeterminate state occurs when all the three inputs are equal to 1. This condition places 0's on both inputs of the basic SR latch, which places it in the undefined state.

#### **Advantages:**

- It is an important circuit because other latches and flip-flops are constructed from it.

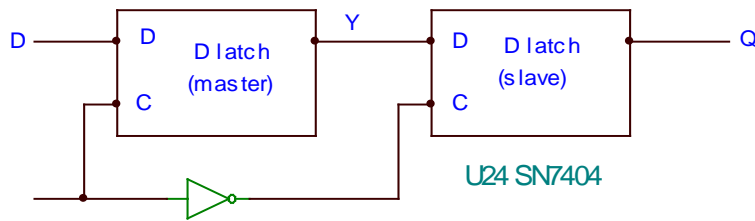
#### **Disadvantages:**

- In the indeterminate state, when the control input goes back to 0, one cannot conclusively determine the next state as it depends on whether the S or R input goes to zero first. This makes it difficult to manage and it is seldom used in practical.

### **9.Explain the working of Master-slave flip-flop. [CO3-L2]**

#### **Master Slave D Flip-Flop:**

- It consists of two D latches and an inverter.
- The first latch is called the master and the second the slave.

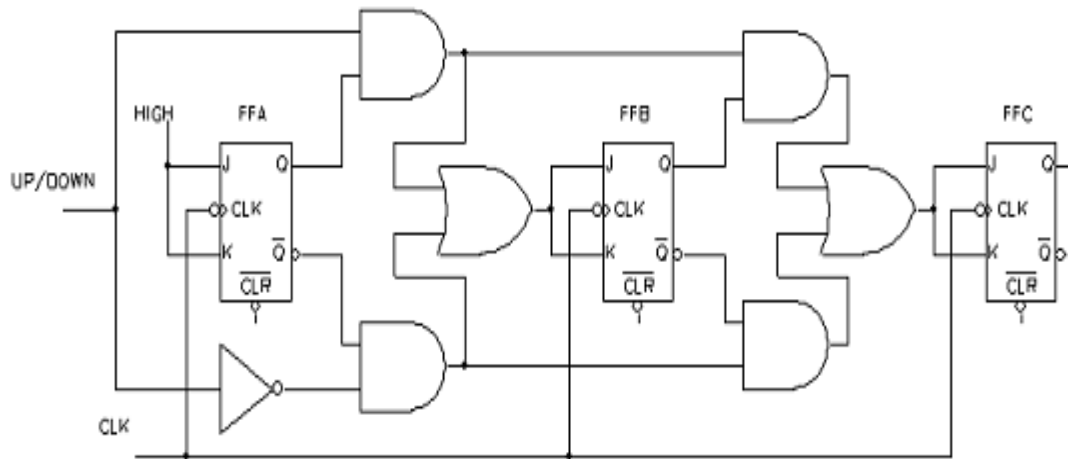


- The circuit samples the D input and changes its output Q only at the negative edge of the controlling clock (designated as CLK).
- When the clock is 0, the output of the inverter is 1.
- The slave latch is enabled and its output Q is equal to the master output Y.
- The master latch is disabled because  $CLK = 0$ . When the input changes to the logic 1 level, the data from the external d input is transferred to the master.
- The Slave input is disabled as long as the clock remains in the 1 level because its C input is equal to 0.
- Any change in the input changes the master output at Y but cannot effect the slave output.
- When the pulse returns to 0, the master is disabled and is isolated from the D input.
- At the same time, the slave is enabled and the value of Y is transferred to the output of the flip flop at Q. Thus the output of the flip-flop can change only during the transition of the clock from 1 to 0.
- Master Slave flip flop dictates that the output may change only during the negative edge of the clock.
- It is also possible to design the circuit so that the flip flop output changes on the positive edge of the clock.
- This happens in a flip flop that has an additional inverter between the CLK terminal and the junction between the other inverter and input C of the master latch.

**10. Draw and explain the working of 3 bit up/down synchronous counter. [CO3-L2]**

The synchronous Up/down counter is designed by using inverted flip flop outputs to drive the following inputs.

To form a parallel up/down is used to control whether the normal flip flop output or the inverted flip flop outputs are fed to the following inputs.



A logic 1 on the Up/down enables AND gates 1 and 2 and disables AND gates 3 and 4 .

This allows  $Q_A$  and  $Q_B$  outputs through the J and K inputs of next flip flops so that the counter will count up.

A logic 0 on the Up/down disables AND gates 1 and 2 and enables AND gates 3 and 4 . This allows  $Q_A$  and  $Q_B$  outputs through the J and K inputs of next flip flops so that the counter will count down.

**Unit – IV**  
**Memory Devices**

**Part-A**

**1. What is RAM? [CO4-L1]**

A memory Unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the device. The time it takes to transfer information to or from any desired random location is always same. Hence, the name Random Access Memory abbreviated RAM

**2. What are the different types of programming the PLA? [CO4-L1]**

1. Mask programmable PLA
2. Field programmable PLA

**3. List basic types of programmable logic devices? [CO4-L1]**

1. Read only memory
2. Programmable logic Array
3. Programmable Array Logic

**4. Explain ROM?(or) Define the term ROM. [CO4-L1]**

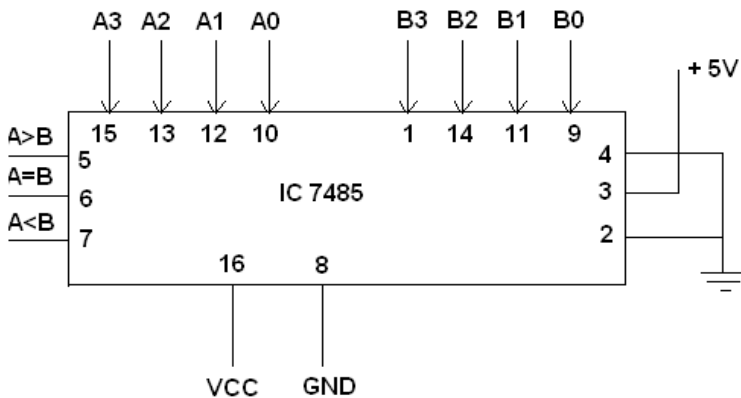
A read only memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package. It consists of  $n$  input lines and  $m$  output lines. Each bit combination of the input variables is called an address. Each bit combination that comes out of the output lines is called a word. The number of distinct addresses possible with  $n$  input variables is  $2^n$ .

**5. How does ROM retain information? [CO4-L1]**

ROM retains information since it is non – volatile memory.

**6. Define address and word? [CO4-L1]**

In a ROM, each bit combination of the input variable is called on address. Each bit combination that comes out of the output lines is called a word.

**7. Using a single IC 7485, draw the logic diagram of a 4 bit comparator? [CO4-L1]****8. What is programmable logic array? How it differs from ROM? [CO4-L1]**

In some cases the number of don't care conditions is excessive, it is more economical to use a second type of LSI component called a PLA. A PLA is similar to a ROM in concept; however it does not provide full decoding of the variables and does not generate all the minterms as in the ROM.

**9. State the types of ROM. [CO4-L1]**

- Masked ROM.
- Programmable Read only Memory
- Erasable Programmable Read only memory.
- Electrically Erasable Programmable Read only Memory

**10. Explain PROM. [CO4-L1]**

**PROM (Programmable Read Only Memory):**

It allows user to store data or program. PROMs use the fuses with material like nichrome and polycrystalline. The user can blow these fuses by passing around 20 to 50 mA of current for the period 5 to 20 $\mu$ s. The blowing of fuses is called programming of ROM. The PROMs are one time programmable. Once programmed, the information is stored permanent. (or) PROM is Programmable Read Only Memory. It consists of a set of fixed AND gates connected to a decoder and a programmable OR array.

#### **11. Explain EPROM. [CO4-L1]**

##### **EPROM: (Erasable Programmable Read Only Memory)**

EPROM use MOS circuitry. They store 1's and 0's as a packet of charge in a buried layer of the IC chip. We can erase the stored data in the EPROMs by exposing the chip to ultraviolet light via its quartz window for 15 to 20 minutes. It is not possible to erase selective information. The chip can be reprogrammed.

#### **12. Explain EEPROM.(or) How is individual location in EEPROM programmed or erased? [CO4-L1]**

##### **EEPROM : (Electrically Erasable Programmable Read Only Memory)**

EEPROM also use MOS circuitry. Data is stored as charge or no charge on an insulated layer or an insulated floating gate in the device. EEPROM allows selective erasing at the register level rather than erasing all the information since the information can be changed by using electrical signals.

#### **13. What is programmable logic array? How it differs from ROM? [CO4-L1]**

In some cases the number of don't care conditions is excessive, it is more economical to use a second type of LSI component called a PLA. A PLA is similar to a ROM in concept; however it does not provide full decoding of the variables and does not generate all the minterms as in the ROM.



**14. List the major differences between PLA and PAL? (OR) Is the PAL same as the PLA? [CO4-L1]**

Sl.no	PLA	PAL
1	Both AND and OR arrays are programmable and Complex	AND arrays are programmable OR arrays are fixed
2.	Costlier than PAL	Cheaper and Simpler.

**15. Define PLD? [CO4-L1]**

Programmable Logic Devices consist of a large array of AND gates and OR gates that can be programmed to achieve specific logic functions.

**16. Give the classification of PLD's ? [CO4-L1]**

PLDs are classified as

PROM (Programmable Read Only Memory),

Programmable Logic Array (PLA),

Programmable Array Logic (PAL),

Generic Array Logic (GAL).

**17. Define PLA.(or) What is PLA? [CO4-L1]**

PLA is Programmable Logic Array (PLA). The PLA is a PLD that consists of a programmable AND array and a programmable OR array.

**18. What is mask – programmable PLA? [CO4-L1]**

With a mask programmable PLA, the user must submit a PLA program table to the manufacturer.

**19. What is field programmable logic array? [CO4-L1]**

The second type of PLA is called a field programmable logic array. The user by means of certain recommended procedures can program the FPLA.

**20. Define PAL?(or) Write notes on PAL. [CO4-L1]**

PAL is Programmable Array Logic. PAL consists of a programmable AND array and a fixed OR array with output logic.

**21. Why was PAL developed? [CO4-L1]**

It is a PLD that was developed to overcome certain disadvantages of PLA, such as longer delays due to additional fusible links that result from using two programmable arrays and more circuit complexity.

**22. What does PAL 10L8 specify? [CO4-L1]**

PAL - Programmable Logic Array

10 - Ten inputs

L - Active LOW Output

8 - Eight Outputs

**23. Why the input variables to a PAL are buffered? [CO4-L1]**

The input variables to a PAL are buffered to prevent loading by the large number of AND gate inputs to which available or its complement can be connected.

**24. Give the comparison between PROM and PLA. [CO4-L1]**

Sl.no	PROM	PLA
1	And array is fixed and OR array is programmable.	Both AND and OR arrays are Programmable.
2.	Cheaper and simple to use.	Costliest and complex than PROMS.

**25. Comparison between RAM and ROM[CO4-L1]**

RAM	ROM
RAM's have both read and write capability	ROM's have only read operation
RAM's are volatile memories They lose stored data when the power is turned off	ROM's are non-volatile memories They retain stored data even if power is turned off
RAM's are available in both bipolar and MOS technologies	ROM's are available in both bipolar and MOS technologies

**26. Compare SRAM & DRAM . [CO4-L1]**

Static RAM	Dynamic RAM
SRAM consists of flip-flops. Each flip-flop stores one bit.	DRAM stores the data as charge on the capacitor. It consists of MOSFET and the capacitor for each cell
SRAM contains less memory cells per unit area	DRAM contains more memory cells per unit area
costly	Less costly
It's access time is less, hence faster memories	It's access time is greater than SRAM
Refreshing circuitry is not required	Refreshing circuitry is required

**27. What are the advantages of RAM. [CO4-L1]**

- Fast operating speed ( $< 150$  nS)
- Low power dissipation ( $< 1$  mW)
- Compatibility
- Non-destructive readout

**28. Write a note on advantages of ROM[CO4-L1]**

- Ease and speed of design
- Faster than MSI devices PLD and FPGA

- The program that generates the ROM contents can easily be structured to handle unusual or undefined cases
  - A ROM's function is easily modified just by changing the stored pattern, unusually without changing any external connections
  - More economical

**29. what are the disadvantages of ROM[CO4-L1]**

- for functions more than 20 inputs, a ROM based circuit is impractical because of the limit on ROM sizes that are available
- For simple to moderately complex functions, ROM based circuit may costly, consume more power, run slower.

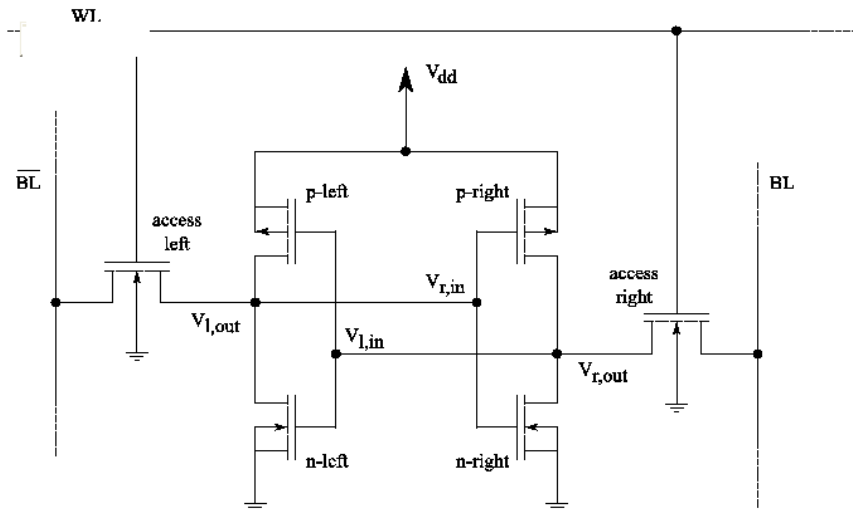
**31. How is memory size is specified? [CO4-L1]**

Memory size is specified by the total number of words in a memory array which is in the form of ( $2^n \times m$ ).

Where,  $2^n$  - total number of words

m- Number of bits in a word / number of output data lines.

**32. Draw the circuit of MOSFET RAM cell.(Nov/Dec 2010,May/June 2006,Nov/Dec 2011)  
[CO4-L1]**



### 33. What are the advantages of static RAM compared to dynamic RAM? [CO4-L1]

- Static RAM contains less memory cells than dynamic RAM.
- It has less access time.
- Refreshing circuit is not required.

### 34. What is meant by Memory expansion? Mention its limit. [CO4-L1]

Memory expansion is the process of expanding the memory size.

It is achieved by 2 ways. (i) Expanding Word size (ii) Expanding Memory Capacity

Limitations:

The memory can be expanded as  $2^n$  where n is the address lines.

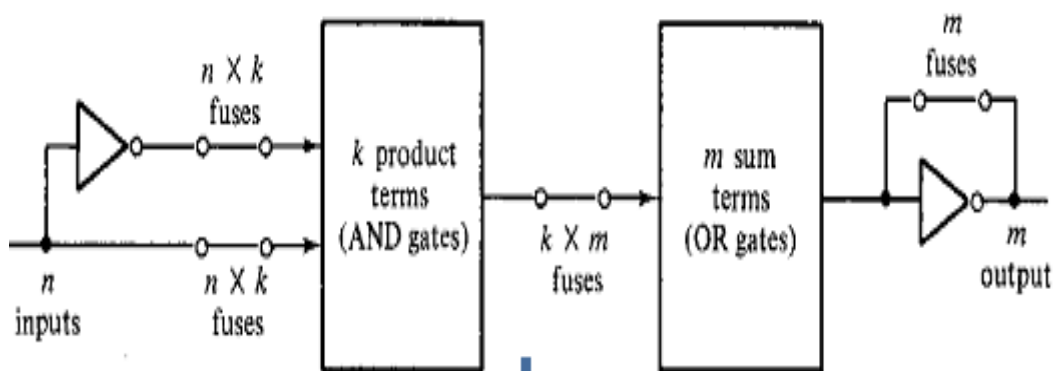
## Part-B

### 1.Explain in detail about PLA. [CO4-L2]

#### Programmable Logic Array:

- A **programmable logic array (PLA)** is a kind of programmable logic device used to implement combinational logic circuits

- The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. A combinational circuit Combinational logic may occasionally have don't-care conditions.
- When implemented with a Read-only memory, a don't care condition becomes an address input that will never occur. The words at the don't care address need not be programmed and may be left in their original state (all 0's or all 1's). The result is not all the bit patterns available in Read-only memory are used which may be considered a waste of available component.
- Logically, a PLA is a circuit that allows implementing Boolean functions in sum-of-product form. The typical implementation consists of input buffers for all inputs, the programmable AND-matrix followed by the programmable OR-matrix, and output buffers.
- The size of PLA is specified by the number of inputs and the number of outputs.
- A typical PLA has 16 inputs 48 product terms and 8 outputs.
- A with ROM PLA may be mask programmable or field programmable.

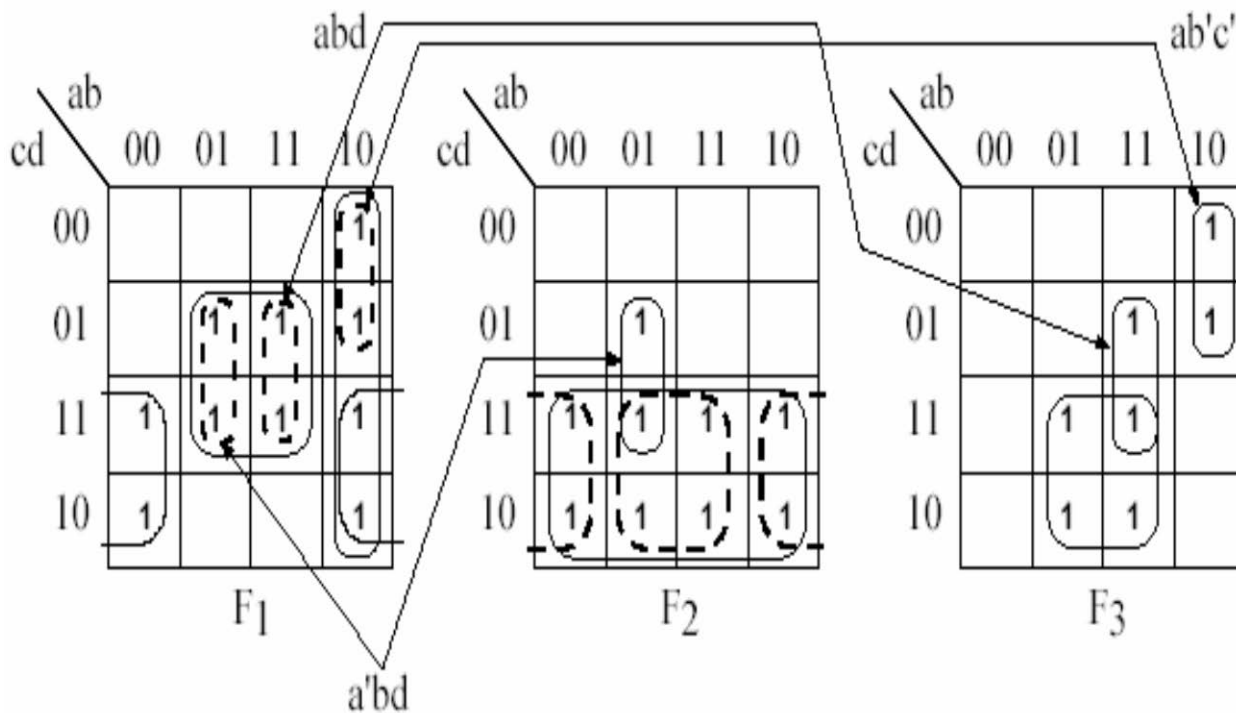


**2. Realization of a given function using min number of rows in the PLA [CO4-L2]**

$F1 = \sum m(2,3,5,7,8,9,10,11,13,15) \dots (1)$

$F2 = \sum m(2,3,5,6,7,10,11,14,15) \dots (2)$

$F3 = \sum m(6,7,8,9,13,14,15) \dots (3)$



**Fig – Multiple Output Karnaugh Map**

$F1 = BD+B'C+AB' \dots (4)$

$F2 = C+A'BD \dots (5)$

$F3 = BC+AB'C'+ABD \dots (6)$

Equations 1,2 and 3 can be reduced to equations 4, 5 and 6 respectively using Karnaugh map as shown in Fig.

If we implement these reduced equations 4,5 and 6 in a PLA then a total of 8 different product terms (including C) are required. So instead of minimizing each function separately, we have to minimize the total number of rows in the PLA table. When we are trying to design



a logic using PLA, the number of terms in each equation is not important since the size of the PLA does not depend on the number of terms. The term  $AB'C'$  is already present in function F3. So we can use it in F1 instead of  $AB'$  by writing  $AB'$  as  $AB'(C+C')$ . F1 can be now written as

$$\begin{aligned} F1 &= BD + B'C + AB'(C+C') \\ &= BD + B'C + AB'C + A'B'C' \\ &= BD + B'C(1+A) + A'B'C' \\ &= BD + B'C + A'B'C' \end{aligned}$$

This simplification of F1 eliminates the need to use a separate row for the original term  $AB'$  present in F1 of equation (4).

Since the terms  $A'BD$  and  $ABD$  are needed in F2 and F3 respectively, we can replace the term  $BD$  in F1 with  $A'BD + ABD$ . This eliminates the need for a row to implement the term  $BD$  in PLA. Similarly, since  $B'C$  and  $BC$  are used in F1 and F3 respectively, we can replace  $C$  in F3 with  $B'C + BC$ . Now the equations for F1, F2 and F3 with the above said changes can be written as :

$$\begin{aligned} F1 &= BD(A+A') + B'C + AB'(C+C') \\ &= ABD + A'BD + B'C + AB'C' \dots(7) \end{aligned}$$

$$\begin{aligned} F2 &= C(B+B') + A'BD \\ &= BC + B'C + A'BD \dots(8) \end{aligned}$$

$$F3 = BC + AB'C' + ABD \dots(9)$$

The current equations for F1, F2 and F3 as shown in equations 7,8 and 9 respectively have only 5 different product terms. So the PLA table can now be written with only 5 rows. This is a significant improvement over the equations 4, 5 and 6, which resulted in 8 product terms. The reduced PLA table corresponding to equations 7, 8 and 9 is as shown in the figure 13.

a	b	c	d	F1	F2	F3
0	1	-	1	1	1	0
1	1	-	1	1	0	1
1	0	0	-	1	0	1
-	0	1	-	1	1	0
-	1	1	-	0	1	1

**Fig – Reduced PLA table**

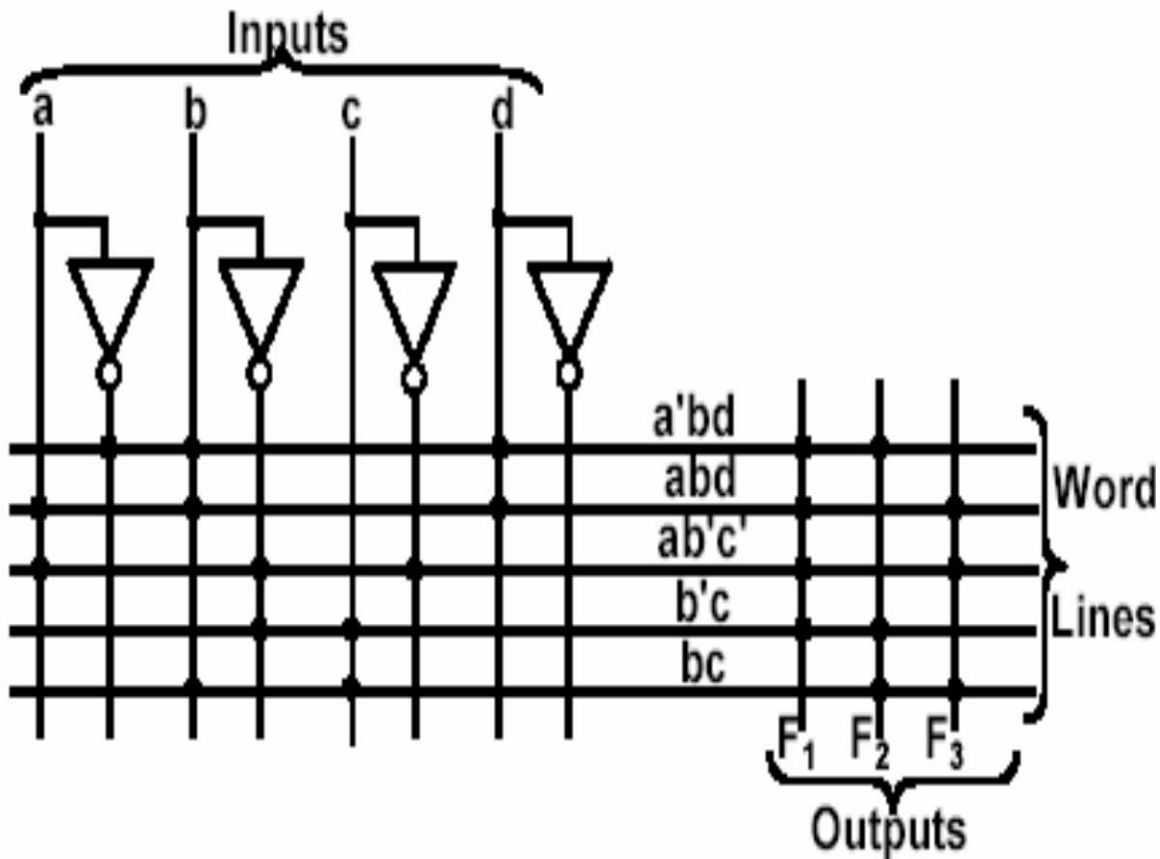
PLA table is significantly different from that of ROM truth table. In a truth table, each row represents a minterm ; therefore, one row will be exactly selected by each combination of input values. The 0s and 1s of the output portion of the selected row determine the corresponding output values.

On the other hand, each row in a PLA table represents a general product term. Therefore, 0, 1 or more rows may be selected by each combination of input values. To determine the value of F for a given input combination, the values of F in the selected rows of the PLA table must be ORed together.

For example, if **abcd=0001** is given as input , no rows are selected as this combination does not exist in the PLA table; and all the F outputs are 0. If **abcd = 1001**, only the third row is selected resulting in  $F_1F_2F_3 = 101$ . If **abcd = 0111**, the first and the fifth rows are selected.

Therefore, to get the values for F1, F2 and F3 is got by ORing the respective values of F1, F2 and F3 in the corresponding rows resulting in  $F_1 = 1 + 0 = 1$ ,  $F_2 = 1+1 = 1$  and  $F_3 = 0 + 1 = 1$ .

## PLA Realization of Equations



The PLA structure, which has four inputs, five product terms and three outputs as shown in equation 7,8 and 9. A dot at the intersection of word line and an input or output line indicates the presence of a switching element in the array.

**3.Design using PAL the following Boolean function. (Apr /May 2013) [CO4-H3]**

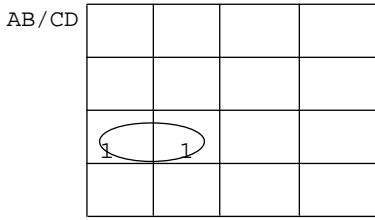
$$W(A,B,C,D)= \Sigma(2,12,13)$$

$$X(A,B,C,D)= \Sigma(7,8,9,10,11,12,13,14,15)$$

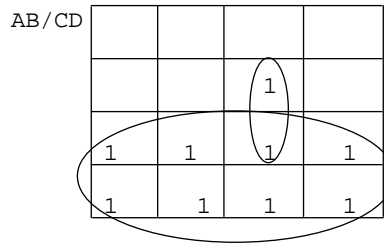
$$Y(A,B,C,D)= \Sigma(0,2,3,4,5,6,7,8,10,11,15)$$

$$Z(A,B,C,D)= \Sigma(1,2,8,12,13)$$

K MAP FOR W



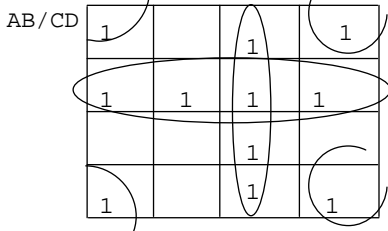
K MAP FOR X



$X = A + ACD$

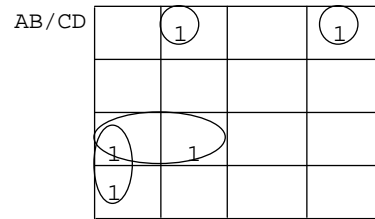
$W = A'B'CD' + ABC'$

K MAP FOR Y



$Y = CD + AB' + B'D'$

K MAP FOR Z



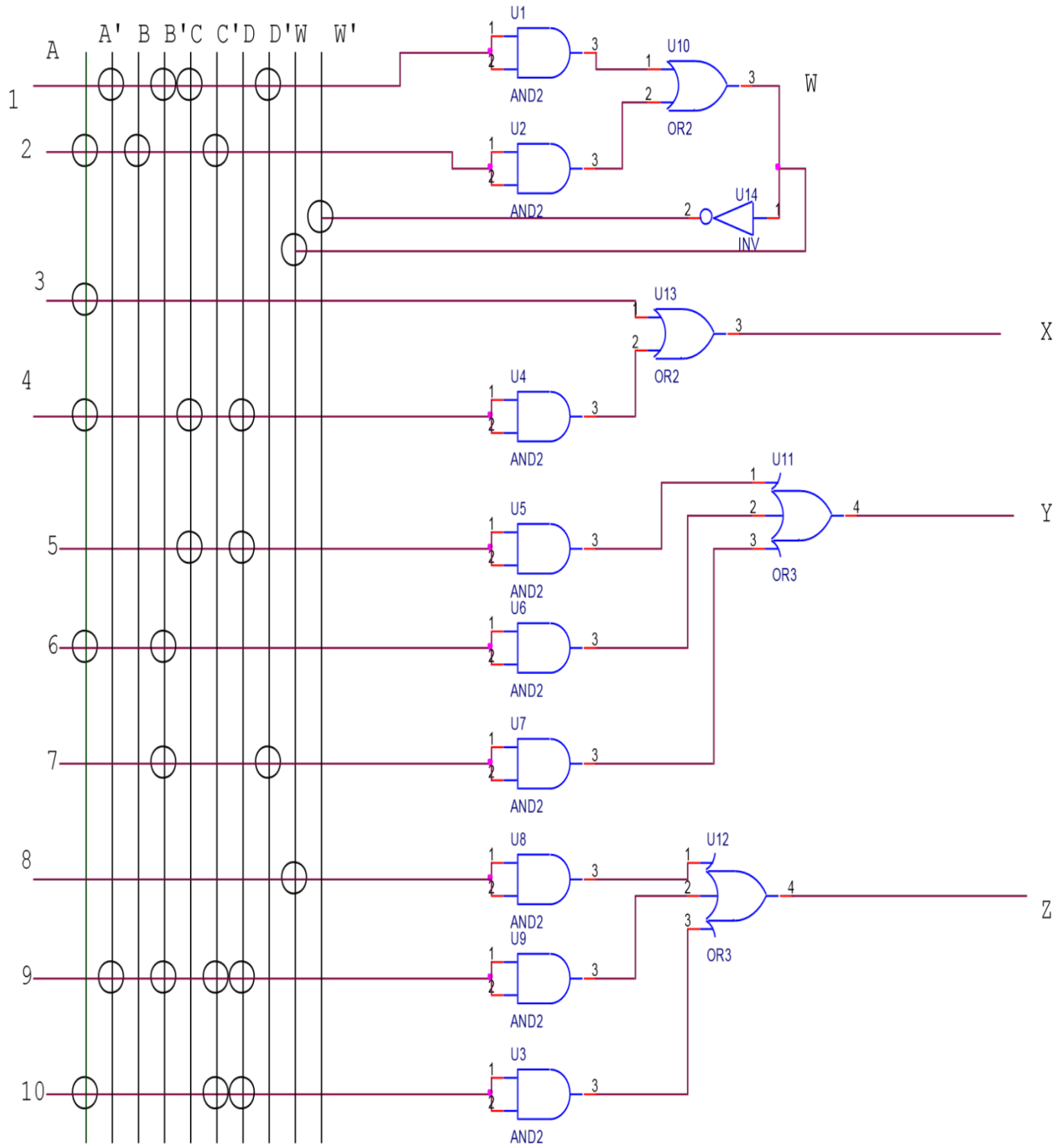
$Z = A'B'C'D + A'B'CD' + ABC' + AC'D'$

$Z = W + A'B'C'D + AC'D'$

**PROGRAMMING TABLE**

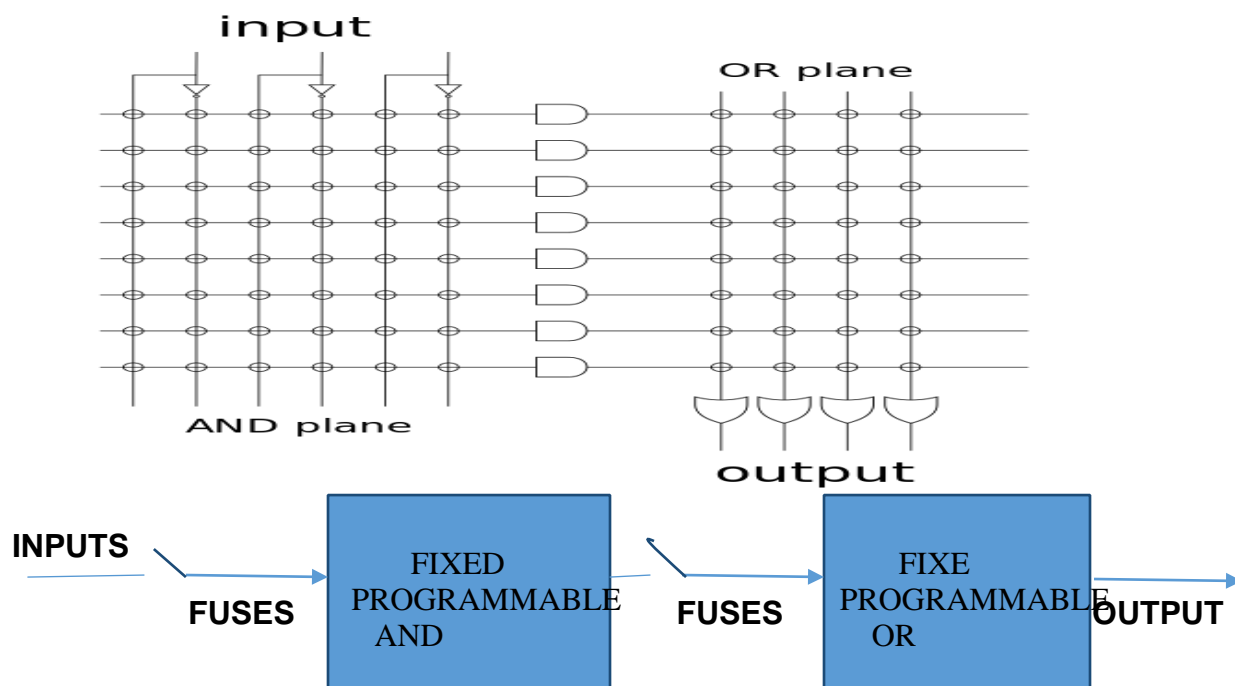
		A	B	C	D	W	OUTPUT
A'B'CD'	1	0	0	1	0	-	W
ABC'	2	1	1	0	-	-	
A	3	1	-	-	-	-	X
ACD	4	1	-	1	1	-	
CD	5	-	-	1	1	-	Y
AB'	6	1	0	-	-	-	
B'D'	7	-	0	-	0	-	
W	8	-	-	-	-	1	Z
A'B'C'D	9	0	0	0	1	-	

<b>AC'D</b>	<b>10</b>	<b>1</b>	<b>-</b>	<b>0</b>	<b>1</b>	<b>-</b>	
-------------	-----------	----------	----------	----------	----------	----------	--



**4. Draw the basic block diagram of PLA device and explain each block. List out its applications. Implement a combinational circuit using PLA by taking a suitable Boolean function. [CO4-L1-Nov/Dec 2011]**

A programmable logic array (PLA) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.



**EXAMPLE**

$$F1(A, B, C) = \Sigma(0, 1, 2, 4)$$

$$F2(A, B, C) = \Sigma(0, 5, 6, 7)$$

		<i>BC</i>		<i>B</i>	
		00	01	11	10
<i>A</i>	0	1	1	0	1
<i>A</i>	1	1	0	0	0
		—————			
		<i>C</i>			

$$F_1 = A'B' + A'C' + B'C'$$

$$F_1 = (AB + AC + BC)'$$

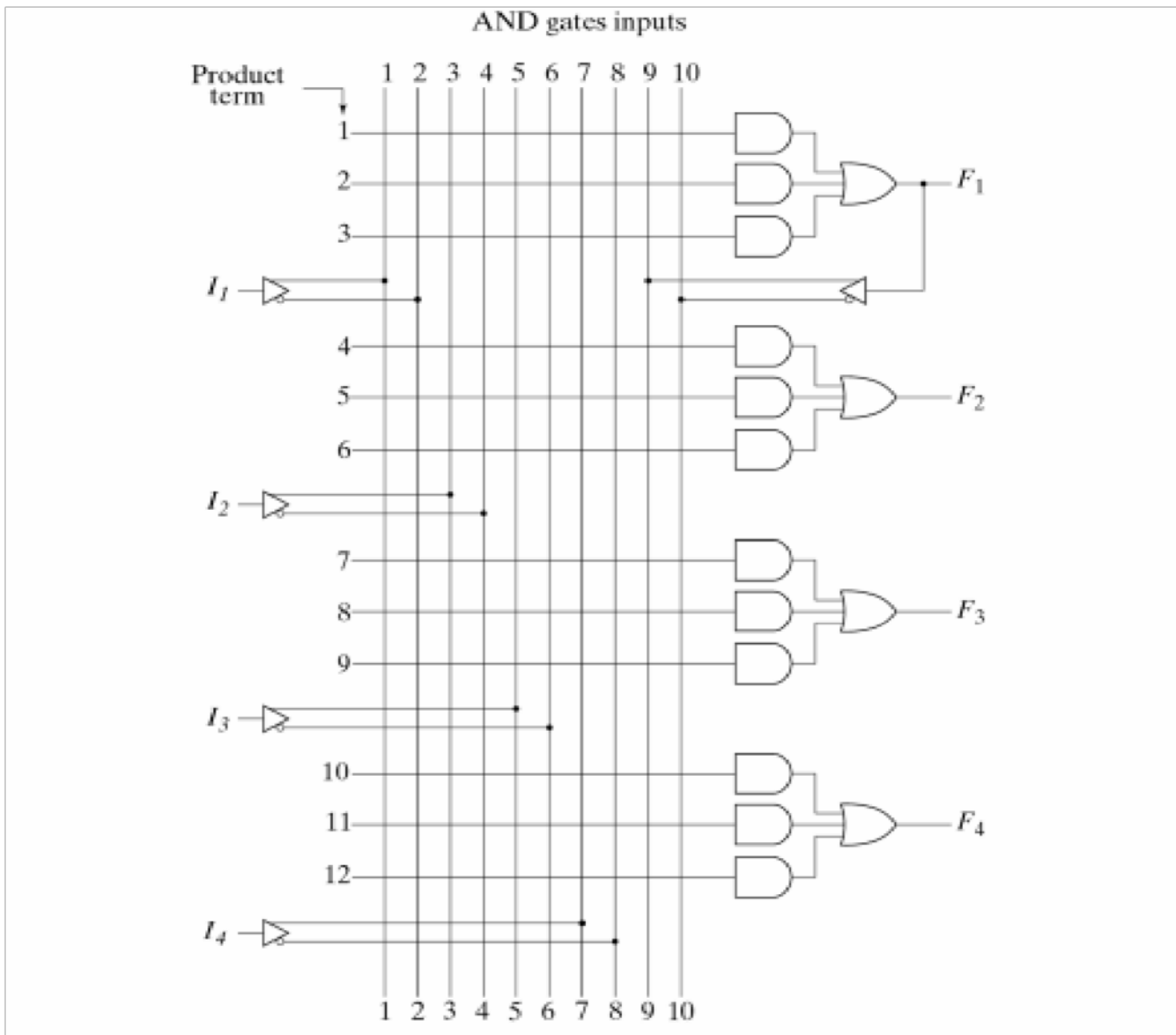
		<i>BC</i>		<i>B</i>	
		00	01	11	10
<i>A</i>	0	1	0	0	0
<i>A</i>	1	0	1	1	1
		—————			
		<i>C</i>			

$$F_2 = AB + AC + A'B'C'$$

$$F_2 = (A'C + A'B + AB'C)'$$

PLA programming table

	Product term	Inputs					Outputs	
		A B C			(C)	(T)		
		A	B	C	<i>F</i> <sub>1</sub>	<i>F</i> <sub>2</sub>		
<i>AB</i>	1	1	1	-	1	1		
<i>AC</i>	2	1	-	1	1	1		
<i>BC</i>	3	-	1	1	1	-		
<i>A'B'C'</i>	4	0	0	0	-	1		



5. Implement the following Boolean functions with a PLA [CO4-L2]

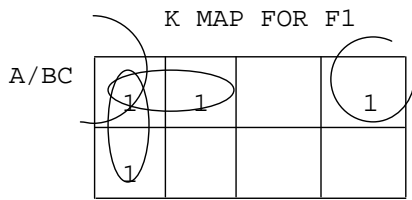
$$F_1(A, B, C) = \Sigma(0, 1, 2, 4)$$

$$F_2(A, B, C) = \Sigma(0, 5, 6, 7)$$

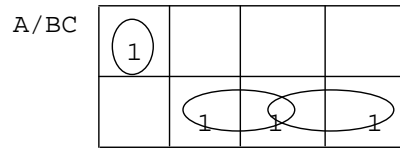
$$F_3(A, B, C) = \Sigma(0, 3, 5, 7)$$



K MAP FOR F2

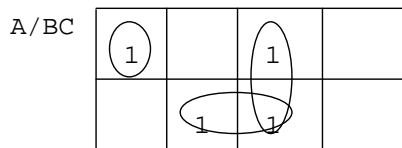


$F1 = B'C' + A'C' + A'B'$

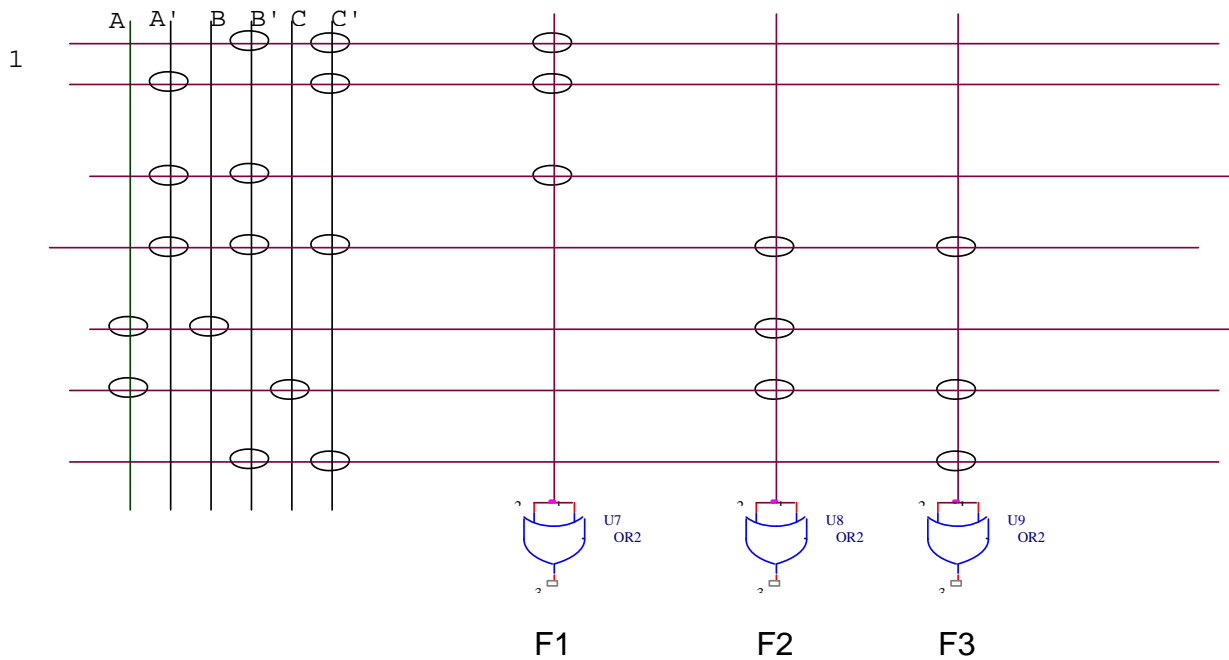


$F2 = A'B'C' + AC + AB$

K MAP FOR F3



$F3 = A'B'C' + AC + BC$



## Unit – V

### Synchronous And Asynchronous Sequential Circuits

#### Part-A

##### **1. What are races? [CO5-L1]**

When 2 or more binary state variables change their value in response to a change in an input variable, race condition occurs in an asynchronous sequential circuit. In case of unequal delays, a race condition may cause the state variables to change in an unpredictable manner.

##### **2. Define non critical race. [CO5-L1]**

If the final stable state that the circuit reaches does not depend on the order in which the state variable changes, the race condition is not harmful and it is called a non critical race.

##### **3. Define critical race. [CO5-L1]**

If the final stable state depends on the order in which the state variable changes, the race condition is harmful and it is called a critical race.

##### **4. What is a cycle?(or)When does a cycle occur? [CO5-L1]**

A cycle occurs when an asynchronous circuit makes a transition through a series of unstable states. If a cycle does not contain a stable state, the circuit will go from one unstable to stable to another, until the inputs are changed.

##### **5. Write a short note on fundamental mode asynchronous circuit. [CO5-L1]**

Fundamental mode circuit assumes that. The input variables change only when the circuit is stable. Only one input variable can change at a given time and inputs are levels and not pulses.

**6. Write a short note on pulse mode circuit. [CO5-L1]**

Pulse mode circuit assumes that the input variables are pulses instead of level. The width of the pulses is long enough for the circuit to respond to the input and the pulse width must not be so long that it is still present after the new state is reached.

**7. Define secondary variables[CO5-L1]**

The delay elements provide a short term memory for the sequential circuit. The present state and next state variables in asynchronous sequential circuits are called secondary variables.

**8. Define flow table in asynchronous sequential circuit. [CO5-L1]**

In asynchronous sequential circuit state table is known as flow table because of the behaviour of the asynchronous sequential circuit. The stage changes occur in independent of a clock, based on the logic propagation delay, and cause the states to flow from one to another.

**9. What is fundamental mode. [CO5-L1]**

A transition from one stable state to another occurs only in response to a change in the input state. After a change in one input has occurred, no other change in any input occurs until the circuit enters a stable state. Such a mode of operation is referred to as a fundamental mode.

**10. Write short note on shared row state assignment. [CO5-L1]**

Races can be avoided by making a proper binary assignment to the state variables. Here, the state variables are assigned with binary numbers in such a way that only one state variable can change at any one state variable can change at any one time when a state transition occurs. To accomplish this, it is necessary that states between which transitions occur be given adjacent assignments. Two binary are said to be adjacent if they differ in only one variable.

**11. Write short note on one hot state assignment. [CO5-L1]**

The one hot state assignment is another method for finding a race free state assignment. In this method, only one variable is active or hot for each row in the original flow table, ie, it requires one state variable for each row of the flow table. Additional row are introduced to provide single variable changes between internal state transitions.

**12. Define hazard. [CO5-L1]**

Hazard is an unwanted transient i.e, spike or glitch that occurs due to unequal path or unequal propagation delays.

**13. Define Static hazard[CO5-L1]**

Static hazard is a condition which results in a single momentary incorrect output due to change in a single input variable when the output is expected to remain in the same state.

**14. Define Static-0 hazard[CO5-L1]**

When the output is to remain at the value 0 and a momentary 1 output is possible during the transition between the two input states, then the hazard is called a static-0 hazard

**15. Define Static-1 hazard[CO5-L1]**

If the two input states both produce a 1 output in the steady state and a momentary 0 output is possible during the transition between the two input states, then the hazard is called a Static-1 hazard.

**16. Define Dynamic hazard. [CO5-L1]**

A transient change occurring 3 or more times at an output terminal of a logic network when the output is supposed to change only once during a transition between two input states differing in the value of one variable.

**17. How to eliminate a Dynamic hazard[CO5-L1]**

Dynamic hazard can be eliminated by covering every pair of 1 cells and every pair of 0 cells in the K-map by at least one sub cube.

**18. Define Essential hazard[CO5-L1]**

Essential hazard is a type of hazard that exists only in asynchronous sequential circuits with two or more feedbacks. An essential hazard is caused by unequal delays along two or more paths that originate from the same input. An excessive delay through an inverter circuit in comparison to the delay associated with the feedback path may cause essential hazard.

**19. How to eliminate Essential hazard[CO5-L1]**

Essential hazard can be eliminated by adding redundant gates as in static hazards. They can be eliminated by adjusting the amount of delay in the affected path. For this, each feedback loop must be designed with extra care to ensure that the delay in the feedback path is long enough compared to the delay of other signals that originate from the input terminals.

**20. Define a primitive flow table. [CO5-L1]**

- ◆ In the design of asynchronous sequential circuits, it is easy to name the states by letter symbols. Such a table is called a flow table.
- ◆ If the flow table has only one stable state for each row in the table which is defined as primitive flow table.

**21. What are the significance of state assignment? [CO5-L1]**

In synchronous circuits-state assignments are made with the objective of circuit Reduction.

Asynchronous circuits-its objective is to avoid critical races

**22. When do race condition occur? [CO5-L1]**

Two or more binary state variables change their value in response to the change in i/p Variable races occur in asynchronous sequential circuits.

**23. State the difference between the synchronous and asynchronous sequential circuits. [CO5-L1]**

S. No	Synchronous sequential circuits	Asynchronous sequential circuits
1.	Memory elements are clocked Flip flops.	Memory elements are either unclocked Flip flops or time delay elements.
2	The operating speed of clock depends on time delay involved. Therefore synchronous circuits can operate slower than asynchronous circuits.	Because of absence of clock, it can operate faster than synchronous circuits.
3	The change in input signals can affect memory elements upon activation of clock signal.	The change in input signals can affect memory elements at any instant of time.
4	Easier to design.	Difficult to design.

**24. What is a pulse mode asynchronous sequential circuit? [CO5-L1]**

- ✓ The inputs are pulses rather than levels.
- ✓ It's operation is similar to clocked synchronous networks, except that state changes are triggered by the input pulses rather than by the clock.
- ✓ Separation between two input pulses must be sufficient so that the network has time to respond to one pulse before the next one occurs.

**25. What are the different techniques used in state assignment? [CO5-L1]**

shared row state assignment

one hot state assignment

**26. What are the steps for the design of asynchronous sequential circuit? [CO5-L1]**

- Construction of primitive flow table

- Reduction of flow table
- state assignment is made
- realization of primitive flow table

### 27. Define compatibility [CO5-L1]

States  $S_i$  and  $S_j$  said to be compatible states, if and only if for every input sequence that affects the two states, the same output sequence, occurs whenever both outputs are specified and regardless of whether  $S_i$  or  $S_j$  is the initial state.

### 28. Define merger graph. [CO5-L1]

The merger graph is defined as follows. It contains the same number of vertices as the state table contains states. A line drawn between the two state vertices indicates each compatible state pair. If two states are incompatible no connecting line is drawn.

## Part- B

1. Design a sequential circuit whose state tables are specified, using D flip-flops. [CO5-H3]

State table of a sequential circuit.

Present State	Next State		Output	
	Q0	Q1	x = 0	x = 1
00	00	01	0	0
01	00	10	0	0
10	11	10	0	0
11	00	01	0	1

Excitation table for a D flip-flop.

Output Transitions		Flip-flop inputs
Q	Q(n)	D
0	0	0
0	1	1
1	0	0
1	1	1

Next step is to derive the excitation table for the design circuit, which is. The output of the circuit is labelled Z.

Present State	Next State	Input	Flip-flop Inputs		Output
			D0	D1	
Q0 Q1	Q0 Q1	x			Z
0 0	0 0	0	0	0	0
0 0	0 1	1	0	1	0
0 1	0 0	0	0	0	0
0 1	1 0	1	1	0	0
1 0	1 1	0	1	1	0
1 0	1 0	1	1	0	0
1 1	0 0	0	0	0	0
1 1	0 1	1	0	1	1

**Excitation table**

Now plot the flip-flop inputs and output functions on the Karnaugh map to derive the Boolean expressions.



## Karnaugh maps

	$x$	0	1
00		0	0
01		0	1
11		0	0
10		1	1
	$Q_0Q_1$		

D0 map

	$x$	0	1
00		0	1
01		0	0
11		0	1
10		1	0
	$Q_0Q_1$		

D1 map

	$x$	0	1
00		0	0
01		0	0
11		0	1
10		0	0
	$Q_0Q_1$		

Z map

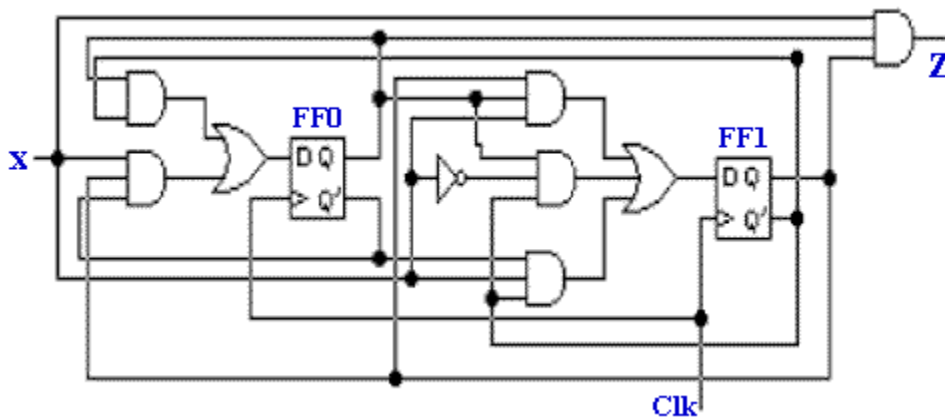
The simplified Boolean expressions are:

$$D0 = Q_0'Q_1' + Q_0'Q_1x$$

$$D1 = Q_0'Q_1'x + Q_0'Q_1x + Q_0Q_1'x'$$

$$Z = Q_0'Q_1'x$$

## Logic Diagram:



## 2. Give a brief note on Synchronous and Asynchronous circuits[CO5-L2]

Sequential circuits are divided into two main types: **synchronous** and **asynchronous**. Their classification depends on the timing of their signals.

*Synchronous* sequential circuits change their states and output values at discrete instants of time, which are specified by the rising and falling edge of a free-running **clock signal**. The clock signal is generally some form of square wave as shown in Figure below.

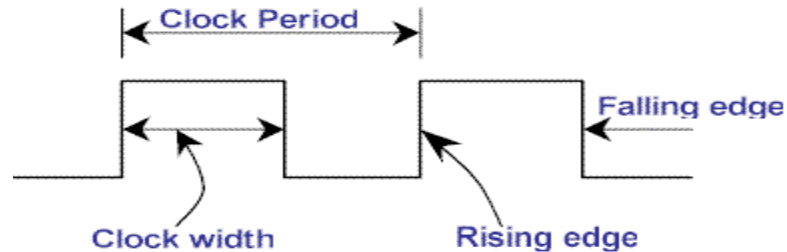


Figure. Clock Signal

From the diagram you can see that the **clock period** is the time between successive transitions in the same direction, that is, between two rising or two falling edges. State transitions in synchronous sequential circuits are made to take place at times when the clock is making a transition from 0 to 1 (rising edge) or from 1 to 0 (falling edge). Between successive clock pulses there is no change in the information stored in memory.

The reciprocal of the clock period is referred to as the **clock frequency**. The **clock width** is defined as the time during which the value of the clock signal is equal to 1. The ratio of the clock width and clock period is referred to as the duty cycle. A clock signal is said to be **active high** if the state changes occur at the clock's rising edge or during the clock width. Otherwise, the clock is said to be **active low**. Synchronous sequential circuits are also known as **clocked sequential circuits**.

The memory elements used in synchronous sequential circuits are usually flip-flops. These circuits are binary cells capable of storing one bit of information. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it. Binary information can enter a flip-flop in a variety of ways, a fact which give rise to the different types of flip-flops. For information on the different types of basic flip-flop circuits and their logical properties, see the previous tutorial on flip-flops.

In **asynchronous** sequential circuits, the transition from one state to another is initiated by the change in the primary inputs; there is no external synchronisation. The memory commonly used in asynchronous sequential circuits are time-delayed devices, usually

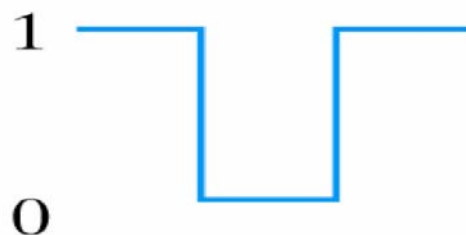
implemented by feedback among logic gates. Thus, asynchronous sequential circuits may be regarded as combinational circuits with feedback. Because of the feedback among logic gates, asynchronous sequential circuits may, at times, become unstable due to transient conditions. The instability problem imposes many difficulties on the designer. Hence, they are not as commonly used as synchronous systems.

**3.Explain the various types of hazards in sequential circuit design and the methods in eliminate them. Give suitable examples. [CO5-L1-Nov/Dec 2014 ]**

In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly.

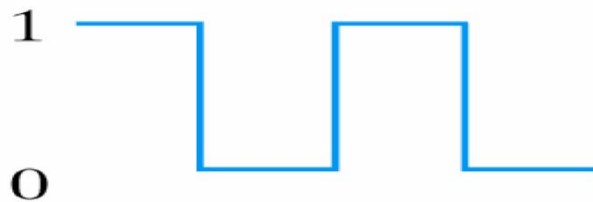
The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**. Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. However, despite the logic arriving at the correct output, it is imperative that hazards be eliminated as they can have an effect on other connected systems. A **static hazard** is the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. There are two types of static hazards:

**Static-1 Hazard:** the output is currently 1 and after the inputs change, the output momentarily changes to 0 before settling on 1



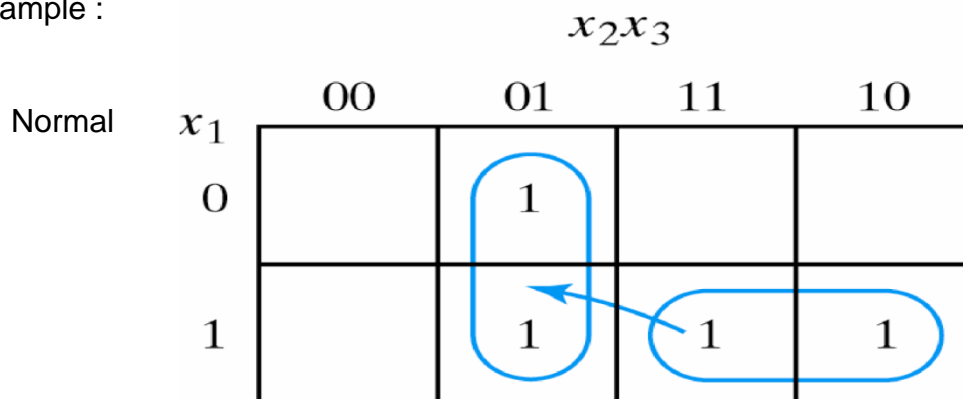
**Static-0 Hazard:** the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0

A **dynamic hazard** is the possibility of an output changing more than once as a result of a single input change. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output. e.g. A logic circuit is meant to change output state from 1 to 0, but instead changes from 1 to 0 then 1 and finally rests at the correct value 0. This is a dynamic hazard.

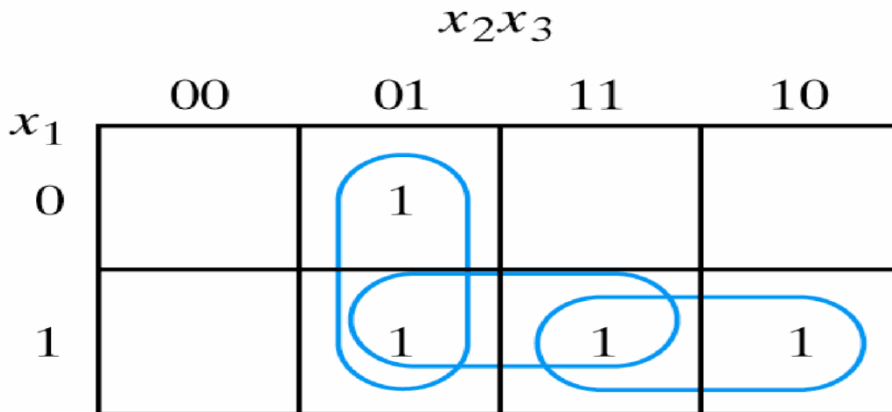


To eliminate static hazard an additional minterm can be added which maintains the output as 1 when hazard occurs.

Example :



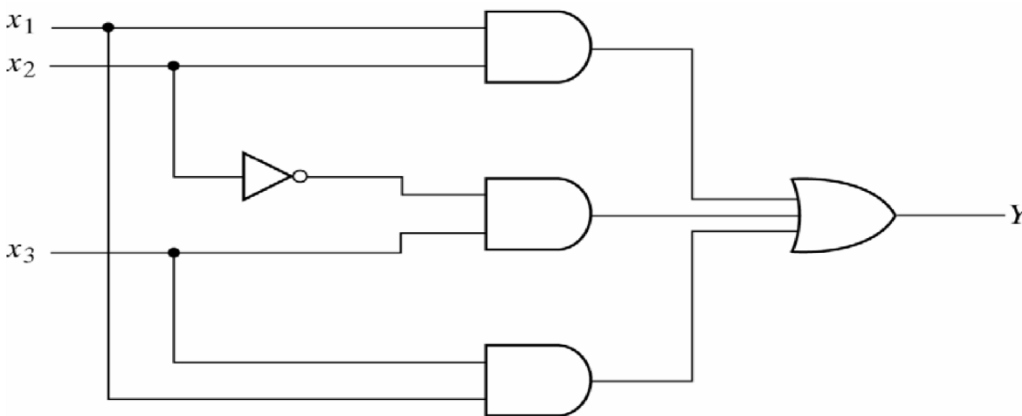
K-Map answer is  $Y = x_1x_2 + x_2'x_3$



Hazard free map is

$$Y = x_1x_2 + x_2'x_3 + x_1x_3$$

Hazard Free Logic diagram is

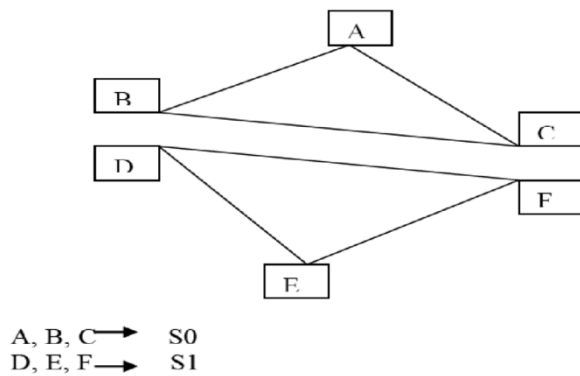


**4.Design an asynchronous sequential circuit (with detailed steps involved) that has two inputs X2 and X1 and one output Z. When X1=0, The circuit is required to give an output Z=1 when X1=1 and X2 = 1 and X1=1 being first. [CO5-H3-Nov/Dec 2015 ]**

**PRIMITIVE FLOW TABLE:**

Present state	Next state, output Z For XY inputs			
	00	01	11	10
A	A	B,-	-,-	C,-
B	A,-	B	D,-	-,-
C	A,-	-,-	D,-	C
D	-,-	B,-	D	E,1
E	F,-	-,-	D,-	E
F	F	B,-	-,-	E,-

Merger graph for problem:



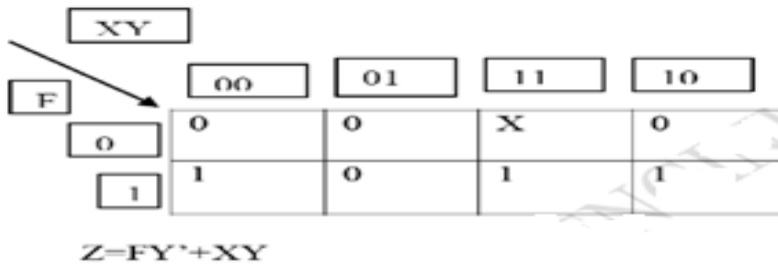
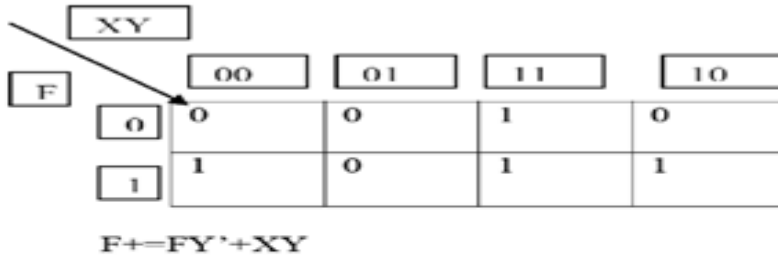
REDUCED FLOW TABLE:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
S0	(S0)	(S0)	S1,-	(S0)
S1	(S1)	S0,-	(S1)	(S1)

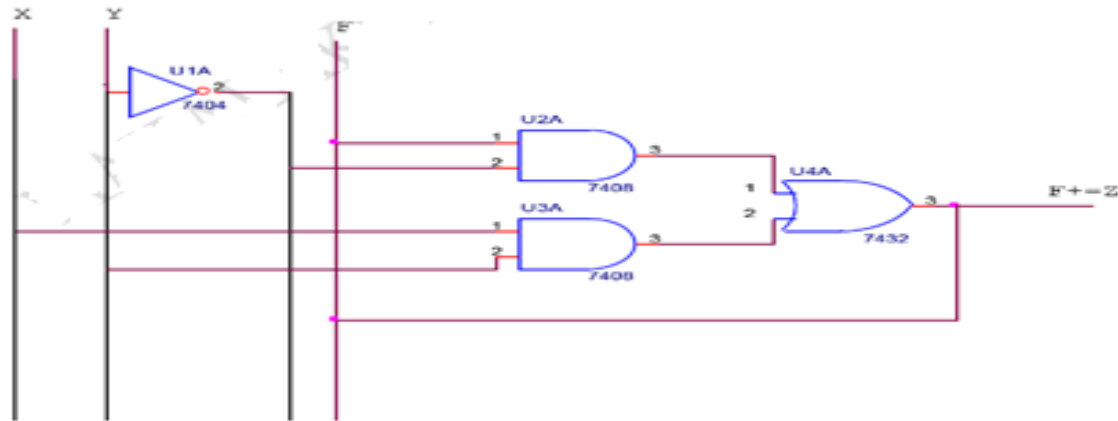
**Transition table:**

Present state	Next state, output Z For XY inputs			
	00	01	11	10
0	(0)	(0)	1,-	(0)
1	(1)	0,-	(1)	(1)

**K-map simplification:**



**Logic diagram:**



5. Describe the various methods of race free state assignment with examples. [CO5-L2-Nov/Dec 2014 ]

**RACE-FREE STATE ASSIGNMENT:**

- Choose a proper binary state assignment to prevent critical races.
- Only one variable can change at any given time when a state transition occurs.
- States between which transitions occur will be given adjacent assignments



- Two binary values are said to be adjacent if they differ in only one variable
- To ensure that a transition table has no critical races, every possible state transition should be checked

State assignments can be demonstrated by means of four row flow table example , multiple row flow table example, one hot state assignment technique

**Multiple row assignment method**

In this method ,each row in the original flow table is replaced by two or more equivalent rows. Thus there will be two or more assignments of state variables to each state. A

		$y_2y_3$		
$y_1$	00	01	11	10
0	$a_1$	$b_1$	$c_1$	$d_1$
1	$c_2$	$d_2$	$a_2$	$b_2$

possible multiple row assignment is shown. Here there are two state variables for each state, and each is a logical compliment of the other. For instance state a is represented by a1 and a2 where a1 has an assignment of 000 while a2 is assigned 111.Also a1 is adjacent to b1 ,d1 and c2, while a2 is adjacent to b2,d2 and c1

**Expanded flow table**

		$x_1x_2$		
	00	01	11	10
$a_1$	$c_2$	$a_1$	$b_1$	$a_1$
$a_2$	$c_1$	$a_2$	$b_2$	$a_2$
$b_1$	$b_1$	$c_1$	$b_1$	$a_1$
$b_2$	$b_2$	$c_2$	$b_2$	$a_2$
$c_1$	$c_1$	$c_1$	$d_1$	$d_1$
$c_2$	$c_2$	$c_2$	$d_2$	$d_2$
$d_1$	$b_2$	$a_1$	$d_1$	$d_1$
$d_2$	$b_1$	$a_2$	$d_2$	$d_2$

Here each row is replaced by two rows. For instance , row a is replaced by rows a1 and a2. In the original table ,state a makes transition to states b and c. In the expanded table a1 makes transition to b1 and c2 and a2 makes transition to b2 and c1. for state a1 with input 01 the sequence will be c2 ,d2,b1,c1,d1 01 : 00 ,10, 00,01,11

**6.Explain the steps for the design of asynchronous sequential circuits. ([CO5-L2-Nov/Dec 2013]**

**Design steps:**

1. Construction of a primitive flow table from the statement. And intermediate step may include the development of a state diagram
2. Primitive flow table is reduced by eliminating redundant states by using state reduction techniques.
3. state assignment is made
4. The primitive flow table is realized using appropriate logic elements.

**Design problems:**

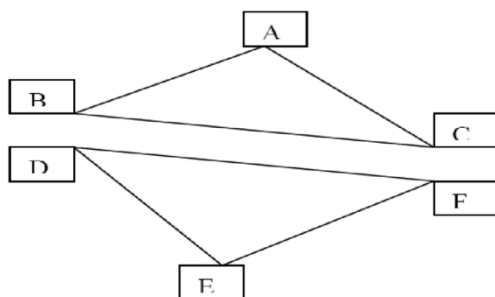
**Example:**

Design a asynchronous sequential circuit with two inputs X and Y and with one output Z. Whenever Y is one, input X is transferred to Z. When Y is zero, the output does not change for any change in X.

**PRIMITIVE FLOW TABLE:**

Present state	Next state, output Z For XY inputs			
	00	01	11	10
A	(A)	B,-	-,-	C,-
B	A,-	(B)	D,-	-,-
C	A,-	-,-	D,-	(C)
D	-,-	B,-	(D)	E,1
E	F,-	-,-	D,-	(E)
F	(F)	B,-	-,-	E,-

Merger graph for problem:



A, B, C → S0  
 D, E, F → S1

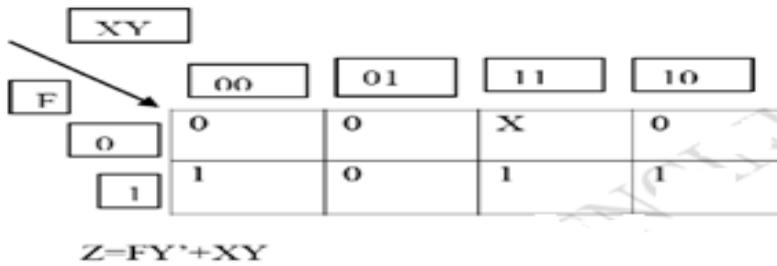
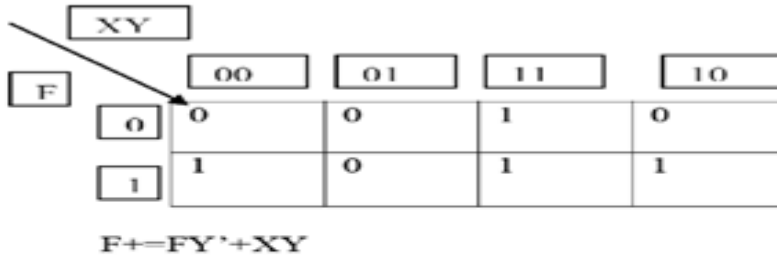
**REDUCED FLOW TABLE:**

Present state	Next state, output Z For XY inputs			
	00	01	11	10
S0	S0	S0	S1,-	S0
S1	S1	S0,-	S1	S1

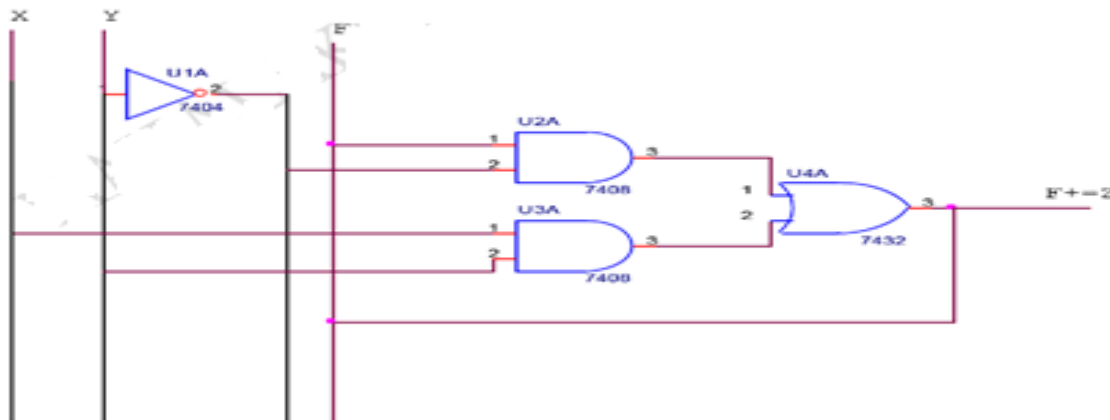
**Transition table:**

Present state	Next state, output Z For XY inputs			
	00	01	11	10
0	0	0	1,-	0
1	1	0,-	1	1

**K-map simplification:**



**Logic diagram:**



7. What are called as essential hazards? How does the hazard occur in sequential circuits? How can the same be eliminated using SR latches? Give an example. [CO5-L1-Apr/ May 2010]

Essential hazard is caused by unequal delays along two or more paths that originate from the same input. An excessive delay through an inverter circuit in comparison to the delay associated with the feedback path may cause such a hazard.

**How does the hazard occur in sequential circuits?**

In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**.

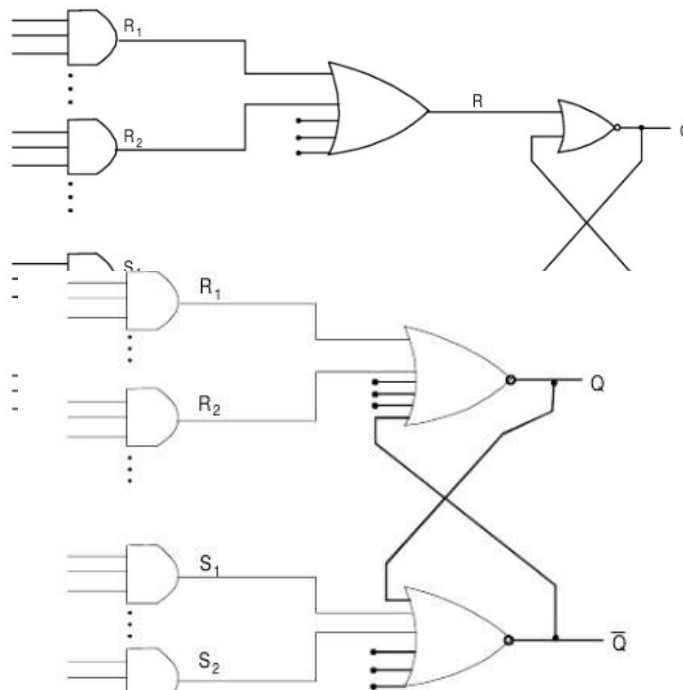
### Elimination using SR latch

A typical network structure with the S-R flip-flop driven by 2-level AND-OR networks constructed from cross-coupled NOR gates is shown in the diagram. The equivalent network structure is provided with multiple input NOR gates.

The two structure are equivalent since in both cases

$$Q = (\bar{Q} + R_1 + R_2 + \dots)'$$

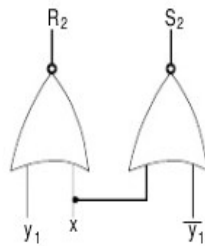
$$\bar{Q} = (Q + S_1 + S_2 + \dots)'$$



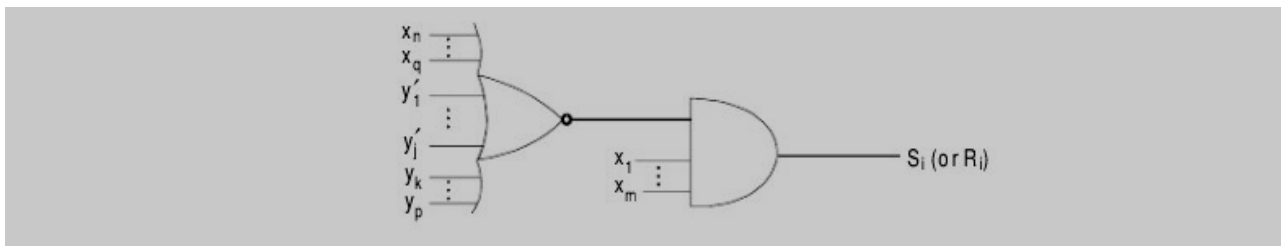
Even if an asynchronous network is realized using S-R flip-flops and S and R networks are free of 0-hazards, essential hazards may still be present. Such essential hazards may

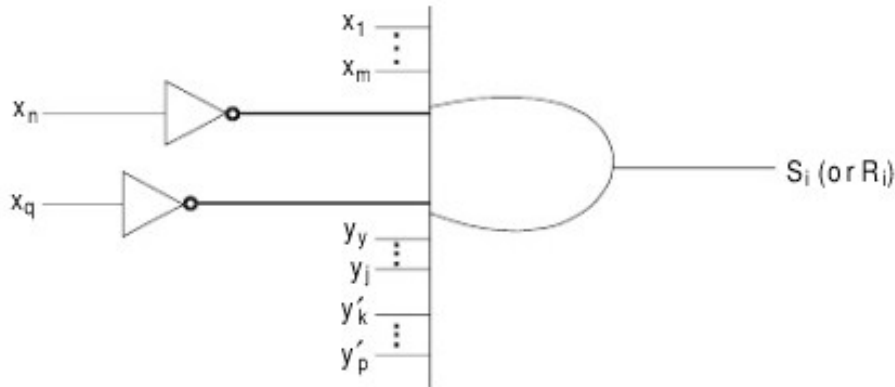
be eliminated as discussed previously by adding delays in the feedback paths for the state variables. Therefore, in an asynchronous network with S-R flip-flops, we can eliminate the essential hazards by arranging the gate structure so that the effect of any input change will propagate to all flip-flop inputs before any state variable changes can propagate back to the flip-flop inputs. For example, the essential hazard of Fig. 8.24 can be eliminated by replacing the R2 and S2 networks

Assuming that wiring delays are negligible that the gate delay is concentrated at the gate output any change in  $x$  will propagate to R2 and S2 before flip-flop 1 output  $y_1$  can change state and this change in  $y_1$  can propagate to R2 and S2



This eliminates the essential hazard.  $x$ 's are external inputs to the circuit, and the  $y$ 's are feedback from flip-flop outputs. If there are essential hazards in the flow table, then the circuit could malfunction due to the inverter delays. By replacing the AND gate with the NOR-AND network the inverters on the  $x$  variables are eliminated. Therefore by replacing all of the AND gate with the NOR-AND combinations as indicated, all of the estimate hazards will be eliminated.





### 8.. Write a VHDL code for 4-bit Universal shift registers. [CO5-L1]

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity reg is
port(din:in STD_LOGIC_VECTOR(3 downto 0);
clk,rst: in std_logic;
S:in STD_LOGIC_vector(1 downto 0);
dout:inout std_logic_vector(3 downto 0));
end reg;

```

```

architecture Behavioral of reg is
signal msbin,lsbin:STD_LOGIC;
begin
process(clk,rst)
begin
if(rst='1') then
dout <= "0000";
elsif(clk'event and clk='1') then
msbin <= din(3);
lsbin <= din(0);
case S is
when "00" => dout <= dout;--hold
when "01" => dout <= msbin & dout(3 downto 1);-- right shift
when "10" => dout <= dout(2 downto 0) & lsbin;-- left shift
when "11" => dout <= din;-- parallel

```



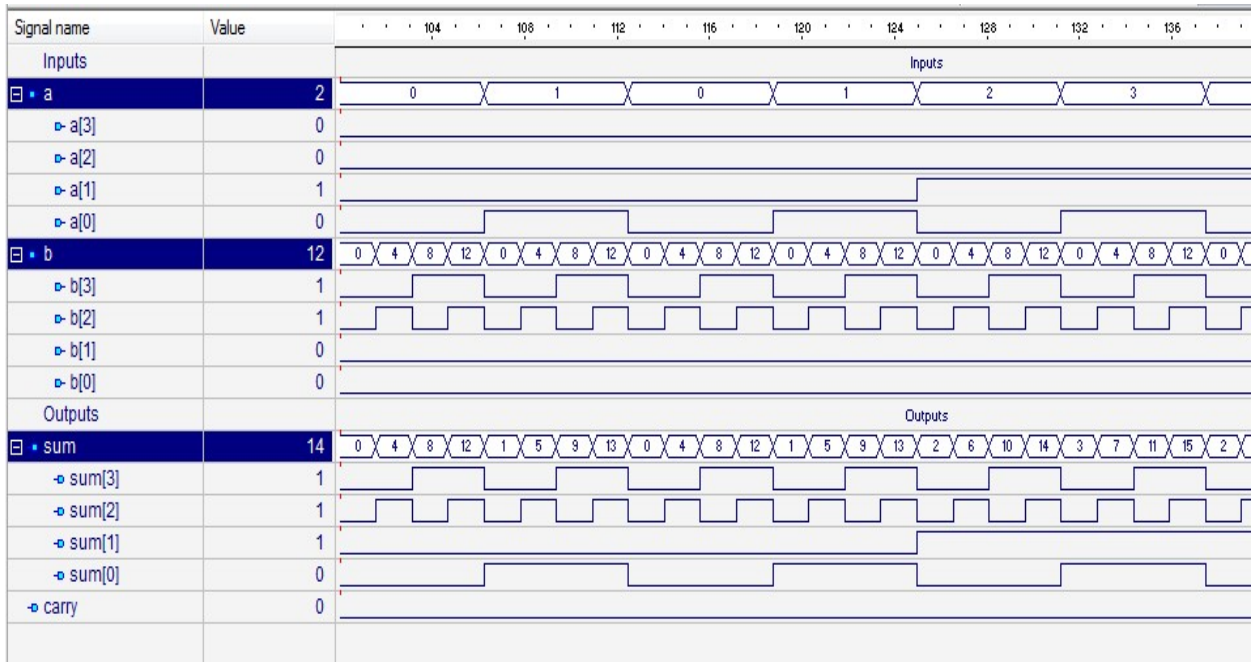
```

when others => dout <= "XXXX";
end case;
end if;
end process;
end Behavioral;

```

9. Explain the concept of behavioral modeling and structural modeling in VHDL. Take the examples of Full adder design for both and write the coding.

Full adder design and code in behaviour modelling [CO5-L2]



```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity full_adder is
    port(
        a : in STD_LOGIC;
        b : in STD_LOGIC;
        cin : in STD_LOGIC ;
        sum : out STD_LOGIC;
        cout: out STD_LOGIC
    );
end full_adder;
--}} End of automatically maintained section
architecture full_adder_beh of full_adder is
begin
    process(a,b,cin)
    begin
        if(a='0')then
            elsif(b='0' and cin='0')then
                sum<='0';cout<='0';
            elsif((b='0' and c='1') or (b='1' and c='0'))then
                sum<='1';cout<='0';
            elsif(b='1' and c='1')then
                sum<='0';cout<='1';
            end if;

            if(a='1')then
            elsif(b='0' and cin='0')then
                sum<='1';cout<='0';
            elsif((b='0' and c='1') or (b='1' and c='0'))then

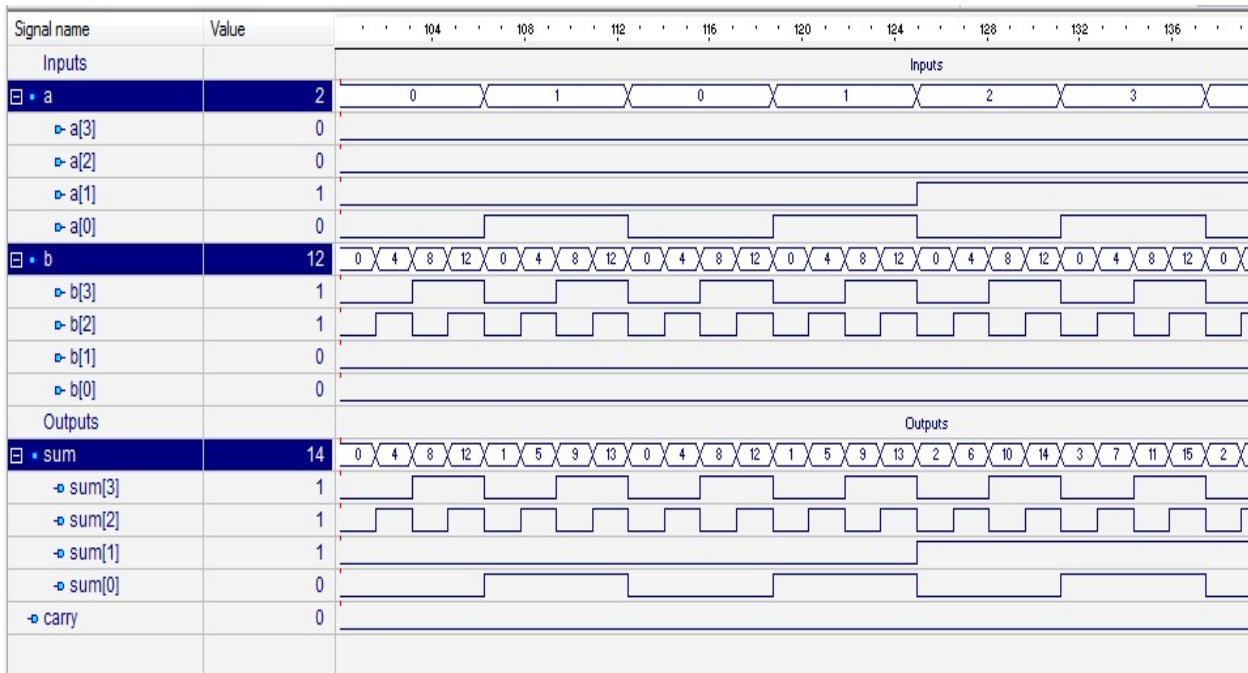
```

```

    sum<='0';cout<='1';
elseif(b='1' and c<='1')then
    sum<='0';cout<='1';
end if;
end process;
end full_adder_beh;

```

### Full adder design and code in structural modeling



```

libraryIEEE;
useIEEE.STD_LOGIC_1164.ALL;
useIEEE.STD_LOGIC_ARITH.ALL;
useIEEE.STD_LOGIC_UNSIGNED.ALL;
entityfull_adder is
    Port (a,b,cin:in STD_LOGIC;
          sum,cout : out STD_LOGIC);
end full_adder;
architecture fa_str of full_adder is
    component xor2

```

```

        port(d1,d2:in std_logic;
            dz:out std_logic);
end component;
component or2
    port(n1,n2:std_logic;
        z:out std_logic);
end component;
component and2
    port(a1,a2:in std_logic;
        u:out std_logic);
end component;
signal s1,s2,s3,s4,s5:std_logic;
begin
    x1:xor2 port map(a,b,s1);
    x2:xor2 port map(s1,cin,sum);
    r1:and2 port map(a,b,s2);
    r2:and2 port map(b,cin,s3);
    r3:and2 port map(a,cin,s4);
    o1:or2 port map(s2,s3,s5);
    o2:or2 port map(s4,s5,cout);
end fa_str;

```

**10. Write a VHDL code to realize a 3 bit Gray code counter using case statement.**

**[CO5-L1-Apr/May 2015 ]**

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Gray_to_Binary is
    port(
        din : in STD_LOGIC_VECTOR(3 downto 0);
        dout : out STD_LOGIC_VECTOR(3 downto 0)
    );

```

```

end Gray_to_Binary;
architecture Gray_to_Binary_arc of Gray_to_Binary is
begin
    btog : process (din) is
    begin
        case din is
            when "0000" => dout <= "0000";
            when "0001" => dout <= "0001";
            when "0010" => dout <= "0011";
            when "0011" => dout <= "0010";
            when "0100" => dout <= "0111";
            when "0101" => dout <= "0110";
            when "0110" => dout <= "0100";
            when "0111" => dout <= "0101";
            when "1000" => dout <= "1111";
            when "1001" => dout <= "1110";
            when "1010" => dout <= "1100";
            when "1011" => dout <= "1101";
            when "1100" => dout <= "1000";
            when "1101" => dout <= "1001";
            when "1110" => dout <= "1011";
            when others => dout <= "1010";
        end case;
    end process btog;
end Gray_to_Binary_arc;

```

**11. Write a VHDL program for 1 to 4 Demux using data flow modeling. [CO5-L1-Nov/Dec 2015 ]**

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

```

```

entity demultiplexer1_4 is
  port(
    din : in STD_LOGIC;
    sel : in STD_LOGIC_VECTOR(1 downto 0);
    dout : out STD_LOGIC_VECTOR(3 downto 0)
  );
end demultiplexer1_4;
architecture demultiplexer1_4_arc of demultiplexer1_4 is
begin
  with sel select
    dout <= (din & "000") when "00",
            ('0' & din & "00") when "01",
            ("00" & din & '0') when "10",
            ("000" & din) when others;
end demultiplexer1_4_arc;

```

**12. Write a VHDL program for Full adder using data structural modeling. [ CO5-L1-  
Nov/Dec 2015 ]**

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity full_adder is
  Port ( a,b,cin : in STD_LOGIC;
         sum,cout : out STD_LOGIC);
end full_adder;

architecture fa_str of full_adder is
  component xor2

```

```
        port(d1,d2:in std_logic;
            dz:out std_logic);
end component;
component or2
    port(n1,n2:std_logic;
        z:out std_logic);
end component;
component and2
    port(a1,a2:in std_logic;
        u:out std_logic);
end component;
signal s1,s2,s3,s4,s5:std_logic;
begin
    x1:xor2 port map(a,b,s1);
    x2:xor2 port map(s1,cin,sum);
    r1:and2 port map(a,b,s2);
    r2:and2 port map(b,cin,s3);
    r3:and2 port map(a,cin,s4);
    o1:or2 port map(s2,s3,s5);
    o2:or2 port map(s4,s5,cout);
end fa_str;
```