Digital Techniques

SEMESTER I

MASTER OF COMPUTER APPLICATION /

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

BLOCK - 2

KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Expert

Prof. Anjana Kakati Mahanta, Gauhati University
Prof. (Retd.) Prannari Talukdar, Gauhati University
Prof. Sikhar Kumar Sarma, Cotton University
Dr. Jyotiprokash Goswami, Assam Engineering College

Course Coordinator

Dr. Sanjib Kr. Kalita, Associate Professor, Dept. of Computer Science, KKHSOU
Dr. Tapashi Kashyap Das, Assistant Professor, Dept. of Computer Science, KKHSOU
Sruti Sruba Bharali, Assistant Professor, Dept. of Computer Science, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
9,10,11,12,13,14	Dr. Chakradhar Das, Lakhimpur Polytechnic
15	Dr. Sangeeta Kakoty, KKHSOU

Editorial Team

Content	:	Dr. Sanjib Kr. Kalita, KKHSOU
Language	:	Prof.(Retd) Robin Goswami, Cotton College
Structure, Format & Graphics	:	Dr. Sanjib Kr. Kalita, KKHSOU

May 2018
ISBN NO: 978-93-87940-21-5

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 License (international): http://creativecommons.org/licenses/by-nc-sa/4.0/

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University

Headquarters : Patgaon, Rani Gate, Guwahati - 781017
Housefed Complex, Dispur, Guwahati-781006; Web: www.kkhsou.in

The University acknowledges with thanks the financial support provided by the **Distance Education Bureau**, **UGC** for the preparation of this study material.

BLOCK INTRODUCTION

This course contains fifteen units in two blocks. **Block II** contains 7 units. These units discuss the basic concept of digital circuit. Unit 9 deals with the basic concept of reduction technique like Sum of Product(SOP), Product of Sum(POS) etc. Unit 10 introduces combinational circuit like multiplexer, de-multiplexer etc.The basis of binary arithmetic circuits like like half adder, full adder are discussed in Unit 11. Introduction to sequential circuit and various flip flops are discussed in unit 12. Unit 13 discusses the concept of registers like serial and parallel register. Concept of counters and various types of counters like synchronous and asynchronous counters and their limitations are discussed in unit 14. Unit 15 discusses the details of memory organization used in digital system.

MASTER OF COMPUTER APPLICATIONS /

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Digital Techniques

DETAILED SYLLABUS

BLOCK-2

Semester I

Page No:

Unit 9: Reduction Techniques [10 marks] **5-23**
Minterm, Maxterms, Boolean Expression, SOP, POS, Conversion of SOP to POS and vice-versa, Simplification of Boolean expression by Algebraic method and Karnaugh Map, Don't care condition

Unit 10: Combinational Circuits [8 marks] **24-40**
Multiplexer, Magnitude Comparator, Demultiplexer, Encoder, Decoder

Unit 11: Binary Arithmetic Circuits [8 marks] **41-50**
Binary Half adder, Full adder, Binary Half subtractor, Full subtractor

Unit 12: Sequential Circuits [10 marks] **51-62**
Sequential circuits, Flip-Flop and its different types: RS, D JK, Master-Slave JK Flip Flops

Unit 13: Registers [8 marks] **63-72**
Shift register, SISO, PISO, SIPO, PIPO

Unit 14: Counters [8 marks] **73-80**
Synchronous and Asynchronous Counter, Limitations of asynchronous counter

Unit 15: Memory Organization [5 marks] **81-108**
Memory types, Random access memory, Access time, RAM package, Static RSM, CMOS memory cell, Dynamic RAM, Read Only Memory, Memory expansion

*****

UNIT 9 : REDUCTION TECHNIQUES

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Minterm and Maxterms

9.4 Simplification of Boolean Function

- 9.4.1 Algebric Manipulation
- 9.4.2 Map Method
- 9.4.3 Don't Care Condition

9.5 Let Us Sum Up

9.6 Further Reading

9.7 Answers to Check Your Progress

9.8 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define and describe minterms and maxterms
- describe the algebric manipulation method
- describe the map method
- describe dont care conditions.

9.2 INTRODUCTION

In the previous units, we have learnt important concepts like number systems, boolean algebra and logic gates. In this unit, you will be introduced to reduction techniques. Minterns and maxterms are introduced in this unit. Simplification of Boolean function using methods like algebric manipulation and map method are covered in this unit.

9.3 MINTERMS AND MAXTERMS

A Boolean function can be expressed in three ways, viz, i) in canonical form, ii) in standard form and in iii) non standard form. To understand these

forms we must have the idea on Minterms and Maxterms.

Minterms and Maxterms : Let us consider two binary variables x and y, which may appear in their normal form or its complement form. Now combining these two variables with an AND operation, we get four possible combinations: $x'y'$, $x'y$, xy', xy. Each of these AND terms represents one of the distinct areas in the Venn diagram of the fig 9.1, and is called a ***minterm*** or a ***standard product***.

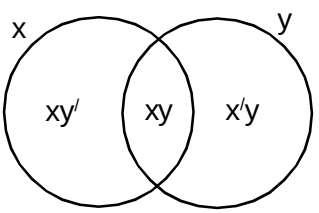


Figure 9.1 : Venn Diagram for two variables

In this way, n variables can be combined to form 2^n minterms. Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1. Each minterm is designated as m_j, where j denotes the decimal equivalent of the binary number of the minterm designated.

Table 9.1 : Minterms and Maxterms for three binary variables

			Minterms		Maxterms	
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	$x' y z'$	m_2	$x + y' + z$	M_2
0	1	1	$x' y z$	m_3	$x + y' + z'$	M_3
1	0	0	$x y' z'$	m_4	$x' + y + z$	M_4
1	0	1	$x y' z$	m_5	$x' + y + z'$	M_5
1	1	0	$x y z'$	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Similarly, with these n variables an OR term can be formed with each variables being primed or unprimed. Thus, we may have 2^n possible combinations and these are called ***Maxterms*** or ***standard sums***. In Table

9.1, the eight maxterms for three variables x, y and z, together with their symbolic designation, are listed. Any 2^n maxterms for n variables may be determined similarly. Each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1. Each maxterm is the complement of its corresponding minterm and vice versa.

If the truth table for a Boolean function is given we can express the Boolean function algebraically by forming a minterm for each combination of the variables which produces a 1 in the function, and then taking the OR of all those terms.

In the Table 9.2 two functions f_1 and f_2 are listed.

Table 9.2 : Functions of three variables

a	b	c	function f_1	function f_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	1

From the table, we see that, function f_1 is 1 for four combinations of a, b, c i.e. $a'b'c'$, $a'b\,c$, $a\,b'c$ and $a\,b\,c'$. Thus, for each one of these minterms the function f_1=1, and we can express the function f_1 as:

$f_1 = a'b'c' + a'\,b\,c + a\,b'c + a\,b\,c'$

$= mo + m3 + m5 + m6$

$= \sum (0, 3, 5, 6)$

(The symbol '$\sum$', is used to mean sum of minterms. Similarly, we can easily verify that –

$f_2 = a'\,b\,c' + a\,b'\,c' + a\,b'\,c + a\,b\,c$

$= m_2 + m_4 + m_5 + m_7$

$= \sum (2,4, 5, 7)$

From here we observe an important property of Boolean algebra. Any Boolean function can be expressed as a sum of minterms (sum means Or ing of terms)

We can obtain the complement of f_1 by forming a minterm for each combination of a, b, c, that produce a 0 in the function and then ORing those terms.

Thus $f1' = a' b' c + a' b c' + a b' c' + a b c$

Now taking the complement of $f1'$, we obtain the function f_1:

$(f_1') = f_1 = (a' b' c + a' b c' + a b' c' + a b c)'$

$= (a + b + c') (a + b' + c) (a' + b + c) (a' + b' + c')$

$= M_1 M_2 M_4 M_7$

$= \pi (1, 2, 4, 7)$ (π is used to mean product of maxterms)

This demonstrates a second important property of Boolean algebra: Any Boolean function can be expressed as product of maxterms (product means ANDing of terms).

Now, we are in a position to define the canonical form of a Boolean function. Boolean functions expressed as a sum of minterms or product of maxterms (i.e. each term of the function must have all the literals) are said to be in canonical form.

Sum of Minterms : It is sometimes convenient to express the Boolean function in its sum-of-minterms form. It can be done by first expanding the expression into a sum of AND terms. Then each term is inspected to see if it contains all the variables or literals. If one or more variable is missing in a term, it is ANDed with an expression $(a + a')$, where a is one of the missing variables. Let us take an example to clarify this procedure.

Note : $a + a' = 1$

Example 9.1: Express the Boolean function $xy + x z' + y'$ in a sum of minterms. Here we see that the function has 3 variables and there are 3 AND terms. Two terms missing one variable and one term missing two variables.

Solution : In term xy, z is missing, therefore: $xy (z + z')$

$= x y z + x y z'$

Similarly in $x z'$, y is missing therefore $x z' (y + y') = x y z' + x y' z'$

In y' two variables x & z are missing, therefore:

$y' (z + z') = y' z + y' z'$, still missing x

$\therefore y' z (x + x') + y' z' (x + x')$

$= x y' z + x' y' z + x y' z' + x' y' z'$

Now, combining all the terms

$F = xy + xz' + y'$

$= x y z + x y z' + x y' z + x y' z' + x' y' z' + x' y' z + x y' z' + x' y' z'$

Eliminating duplicate terms: (x + x = x)

$F = x y z + x y z' + x y' z' + x y' z + x' y' z + x' y' z'$

Rearranging the minterms in ascending order

$F = x' y' z' + x' y z' + x y' z' + x y' z + x y z' + x y z$

$= \sum (0, 2, 4, 5, 6, 7)$

Products of Maxterms : A Boolean function can also be expressed as a product of maxterms. For this, the function is first brought into a form of OR terms. It can be done by using the distributive law x + yz = (x + y) (x + z). Then for any missing variable x in each OR term, the term is ORed with xx'. The process will be clear from the following example.

Example 9.2: Express the Boolean function $f = x y' + x z'$ in a product of maxterm form.

First we convert the function into OR terms using the distributive law.

$f = xy' + x z'$ (Let $x y' = x$, $y = x$ & , $z = z'$)

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

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

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

Now ORing each OR term with $x x'$ (where x is the missing variable)

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

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

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

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

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

Combining all the terms and removing those that appear more than once we finally obtain:

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

$= M_0 M_1 M_2 M_3 M_7$

$= \pi (0, 1, 2, 3, 7)$ (The symbol π denotes the ANDing of maxterms)

Conversion between canonical forms : Boolean functions represented in one canonical form can easily be convered to other canonical forms. It will be clear from the following example.

Example 9.3: Convert the function $F (x, y, z) = \sum (2, 4, 5, 7)$ into product of maxterm form.

Solution : We have $F (x, y, z) = \sum (2, 4, 5, 7) = m_2+m_4+m_5+m_7$

Taking complement $F'(x, y, z) = \sum (0, 1, 3, 6) = m_0+m_1+m_3+m_6$

Now, if we take the complement of F' by De Morgan's theorem,

We obtain $F = (m_0+m_1 + m_3+m_6)'$

$= m_0' m_1' m_3' m_6'$

$= M_0 M_1 M_3 M_6$

$= \pi (0, 1, 3, 6)$

Note : From Table 9.1, it is clear that relation $m_j'=M_j$ is true i.e. a maxterm is a complement of the corresponding minterm and vice versa.

The simplest way to convert from one canonical form to other is: Interchange the symbols $\sum$ and π and list those numbers missing from the original form.

Standard form : In this form, the terms that form the Boolean function may contain one, two or any number of literals. Two types of standard forms are: ***the sum of product*** and ***product of sums***.

The sums of products is a Boolean expression containing, AND terms, called product terms, of one or more literals each.

eg: $F_1 = a + b c' + a b' c$

A product of sums is a Boolean expression containing OR terms, called sum terms. Here also, each term may have any number of literals.

eg: $F_2 = y (x + y') (x' + z) (x + y' + z')$

A Boolean function may also be expressed in a non standard form.

eg: $F_3 = (x' y + x' z) (x' y + x'y')$

This function is neither in sum of product nor in product of sums. Using distributive law it can be converted into a standard form.

9.4 SIMPLIFICATION OF BOOLEAN FUNCTION

We have seen in previous sections that representation of a Boolean function is not unique. The same function can be represented by different numbers of literals and terms. These Boolean functions have to be finally implemented by digital logic gates. The complexity of the digital logic gates is directly related to the complexity of the algebric expression from which the function is implemented. So, it is an important task to simplify the Boolean functions so that they can be implemented easily in a cost effective manner. In this section, we will discuss two methods of simplifying a Boolean function, viz, ***algebric manipulation method*** and the ***map method***.

9.4.1 Algebric Manipulation

On implementing a Boolean function with logic gates, each literal in the function designates an input to a gate, and each term is implemented with a gate. The minimization of the number of literals and the number of terms results in a circuit with less equipment, but it is not always possible to minimize both simultaneously. We will now consider literal minimization. Literal minimization can be done by algebric manipulation. There are no specific rules to follow which will ensure the final answer. The only method available is a cut-and-try procedure using the postulates, the basic theorems and any other manipulation method. To illustrate this procedure let us consider the following examples:

Simplify the following Boolean functions to a minimum number of literals.

1) $x\,y + x\,y'$
   $= x\,(y + y')$
   $= x.\,1$
   $= x$

2) $x\,y\,z + x'\,y + x\,y\,z'$
   $= x\,y\,(z + z') + x'\,y$
   $= x\,y + x'\,y$
   $= y\,(x + x')$
   $= y.$

3) $xy + x'\,z + yz$
   $= xy + x'\,z + y\,z\,(x + x')$
   $= x\,y + x'\,z + x\,y\,z + x'\,y\,z$
   $= x\,y\,(1 + z) + x'\,z\,(1 + y)$

$= x\,y + x'\,z$

In these examples, literals are minimized in 1 and 2, and in 3 a term is minimized.

The algebric manipulation procedure of minimizing a Boolean function is awkward because it lacks specific rules to predict each succeeding step in the process.

9.4.2 Map Method

The map method provides a simple straight forward method for minimizing Boolean functions. It was first proposed by Veitch and slightly modified by Karnaugh and known as the **Veitch diagram** or the "**Karnaugh Map**". This method may be regarded as the pictorial form of a truth table.

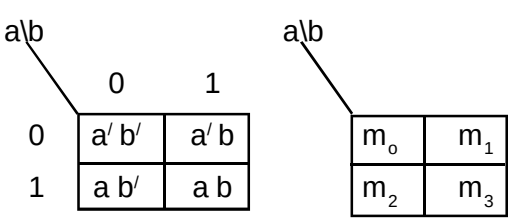
The map is a diagram made up of squares. Each square represents one minterm. As we have already seen that any Boolean function can be represented as a sum of minterms so a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function.

The number of squares in a K-map depends upon the number of variables of the Boolean expression. For n variables, we have 2^n minterms (possible combinations of n variables) and hence 2^n squares in the map. However, the map method is convenient only for small number of variables, i.e. upto six variables. In this section, we will consider up to four variables map.

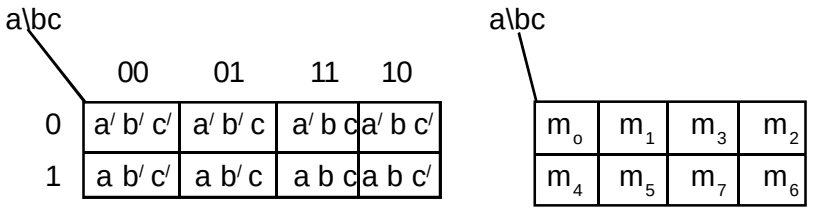
Two, Three and Four variables maps :

K-maps for two, three and four variables map are shown in Fig.9.2 a, b & c.

(a) Two variable K map



(b) Three variables K map



(c) Four variables K map

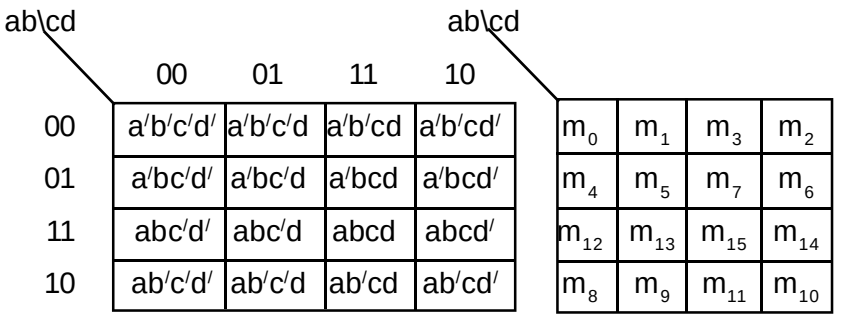
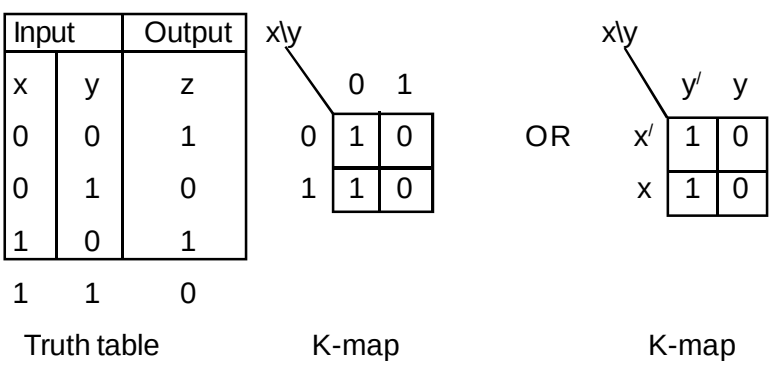


Figure 9.2 : K-map

There are four minterms for two variables; hence the map consists of four squares, one for each minterm. Similarly, for three and four variables there are eight and sixteen minterms respectively and hence the maps consist of eight and sixteen squares.

Representation of Truth Table on K-maps :

We have already known that truth table defines the relationship between output and input.



Truth table K-map K-map

Here, we see that we put 1's in the squares whose minterms are 0 and 2. This is because, from the truth table we see that the output is 1, only for the input combinations x=0, y=0 and x=1 and y=0. For others the

output is 0, and we put 0's in the map accordingly. Similarly, we can represent a truth table having n variables with a K-map of 2^n squares.

Plotting a Boolean expression on the Karnaugh Map : As we have already seen that each square of the K map represents one minterm, so the expression to be plotted must be in sum of product form. The following example clarifies the point.

Example 9.4: Plot the Boolean function $f=yz + xy'z' + xyz'$ on a K-map.

Solution: The term y z is missing one variable, so

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

Now, $F = x'\,y\,z + x\,y\,z + x\,y'\,z' + x\,y\,z'$

$$= m_3 + m_7 + m_4 + m_6$$

$$= m_3 + m_4 + m_6 + m_7$$

Since the expression has three variables, we need a K-map of $2^3 = 8$ squares.

x\yz

	00	01	11	10
0	0	0	1	0
1	1	0	1	1

We plot a 1 in the squares of minterms 3, 4, 6 and 7, and a zero in the other squares.

Simplification of Boolean Function by K-map : We have already seen how a Boolean function can be plotted on the K-map. Now, we will see the process of simplification of Boolean function using K-map. If you observe carefully the K-maps of two, three and four variables you will see that the neighbouring squares are adjacent to each other. Two squares are said to be adjacent if they differ in only one variable. Consider the following three variables k-map.

a\bc	00	01	11	10
0	$a'b'c'$	$a'b'c$	$a'bc$	$a'bc'$
1	$ab'c'$	$ab'c$	abc	abc'

OR

a\bc	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

In this map, m_0 ($a'b'c'$) is adjacent to m_1 ($a'b'c$) because only one variable i.e. "c" appears in m_o in its complement form and m_1, appears in normal form, other variables are same. Similarly, m_o is adjacent with m_4 and m_2; m_1 is adjacent with m_3, m_0 and m_5; m_4 is adjacent with m_o, m_5, and m_6; and so on. When we are going to simplify a Boolean function using K-map, we will consider this adjacency of squares and try to combine as much as possible adjacent squares. We may combine the number of adjacent squares as two, four, eight and sixteen in a four variables map.

By combining two adjacent squares of a four variable map we get a term of three variables or literals.

– combining four adjacent squares we get a term of two literals.

– combining eight adjacent squares we get a term of one literal

– combining sixteen adjacent squares we get a 1 i.e. the function in equal to 1. (i.e. the sum of all minterms is equal to 1).

Example 9.5: Simplify the Boolean function:

Solution: $F = x' y' z' + x' y' z + x y z' + x y z$

	00	01	11	10
0	1	1		
1			1	1

Note : Here, adjacent squares are marked with ovals.

After plotting the function, adjacent squares are combined with each other.

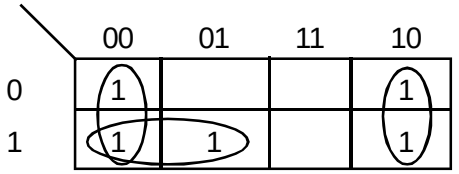
Combining the two squares on top we get the term $x' y'$ (i.e. here variable z is changing, so omit it).

Similarly, combining the two squares on bottom, we get xy.

So, the given function is simplified to $F = x' y' + xy$.

Example 9.6: f (a, b, c) = ∑ (0, 2, 4, 5, 6)

Solution: a\bc



Observing the K-map we see that minterm m_0 is adjacent with m_4 and collectively they are adjacent to m_2 and m_6. This type of adjacency of squares may be determined easily as follows.

Consider the table as an open book, fold it in middle, the squares overlapped with each others are the adjacent squares.

Now, combining m_0, m_2, m_4 and m_6 we get c' (i.e. other variables/ literals are changing, only c' is common to all these squares).

Again, combining m_4 and m_5 we get a b'.

We may combine one square more than once, if necessary.

So, the simplified function is f (a, b, c) = c' + a b'.

Example 9.7: Simplify the Boolean function:

F = $A' B' C'$ + $B' C D'$ + $A' B C D'$ + $A B' C'$

Solution : Here, we need a four variable map. The first, second and fourth terms are missing one variable each. So ANDing with (x + x') (where x is the missing variable) we get the function:

F = $A' B' C' D$ + $A' B' C' D'$ + $A' B' C D'$ + $A B' C D'$ + $A' B C D'$
+ $A B' C' D'$ + $A B' C' D$

Now, plotting the function to the K-map, we get

AB\CD

	00	01	11	10
00	m_0 1	m_1 1	m_3	m_2 1
01	m_4	m_5	m_7	m_6 1
11	m_{12}	m_{13}	m_{15}	m_{11}
10	m_8 1	m_9 1	m_{11}	m_{10} 1

Combining mean terms i.e. squares in the four corners, i.e. m_0, m_2, m_8 and m_{10}, we get $B' D'$.

Combining m_0, m_1, with m_8, m_9 we get $B' C'$. Now, only one square, m_6 remains without combining. We can combine it with m_2 ($\therefore$ they are adjacent) and it gives the term $A' B C'$. Note that if we do not combine m_6, we have to write the term as $A'BCD'$, i.e. with four literals.

So, the simplified form of the function is $F = B' D' + B' C' + A' C D'$

Product of Sum Simplification : In all the previous examples, the minimized Boolean functions derived from the K-maps were expressed in the sum of products (SOP) form. The product of Sums (POS) form can be obtained with a minor modification.

The procedure for obtaining a minimized function in product of sums follows from the basic properties of Boolean functions. The minterms included in the function are represented by placing 1's in the corresponding squares on the map. The minterms not included in the function denote the complement of the function. If we mark these empty squares by 0's and combine them into valid adjacent squares, we obtain a simplified expression of the complement of the function, i.e. of F'. If we take the complement of F, we get back the function F. The function so obtained is automatically in the product of sums form, because of the De Morgan's theorem. The following example clarifies the procedure.

Example 9.8: Simplify the Boolean function

F (A,B,C,D) = $\sum$ (1, 5, 8, 9, 10) in (a) sum of product and (b) product of sums.

Solution : Sum of products simplification of left as an exercise, and we will show only POS simplification.

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

The squares marked with 0's represents the minterms not included in F i.e., they denote the complement of F. Combining the 0's as shown in the map, we obtain the simplified complemented function: $F^{/} = AB + CD + A^{/} D^{/}$

Applying De Morgan's theorem, we obtain:

$F^{/} = AB + CD + A^{/} D^{/}$

$(F^{/})^{/} = F = (AB + CD + A^{/} D^{/})^{/}$

$= (AB)^{/} (CD)^{/} (A^{/} D^{/})^{/}$

$= (A^{/} + B^{/}) (C^{/} + D^{/}) (A+D)$

which is the simplified function in product of sums.

$F = A'B'C + AB'C' + AB'D'$

9.4.3 Don't Care Condition

The function is said to be completely specified when for every possible combination of input variables, we know an output value. Accordingly, these output values (i.e. 1's and 0's) are plotted in the map. The combinations are usually obtained from a truth table that lists the conditions under which the function is a 1. The function is assumed equal to 0 under all other conditions. But this assumption is not always true. There are some applications where certain combinations of input variables never occur. For example, we use 4 bit code to represent the decimal numbers, (0 to 9). Thus, we use the codes from 0000 to 1001, but other 6 codes from 1010 to 1111 are never used. These combinations are called don't care conditions. These don't care conditions can be used on a map to provide further simplification of the function.

It is to be kept in mind that a don't care condition cannot be marked with a 1 on the map because it would mean that the function always produces a 1 for such input combinations. Similarly, for the same reason, we cannot mark these by a 0. To distinguish from 1's and 0's don't care conditions are marked by an X. The X's may be assumed to be either 0 or 1, in choosing adjacent squares to simplify the function in the map depending on which give the simplest expression. An X, which does not contribute to cover a larger area i.e. more minterms/squares, need not be used.

Example 9.9: Simplify the Boolean function

F (A,B,C,D) = ∑ (1,3,7,11,15) with the don't care conditions:

d (A,B,C,D) = ∑ (0, 2, 5) in (a) SOP and (b) POS.

Solution :

Step 1 : Construct a four variables K-map

Step 2 : Squares that represent the minterms of the function are marked with 1s.

Step 3 : The squares for don't care conditions are marked with Xs.

Step 4 : The 1s and Xs are combined in any convenient manner so as to enclose the maximum number of adjacent squares.

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

(a) SOP

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

(b) POS

Step 5 : Combining the squares with 1s and Xs on top in fig. (a), we get the term: $A^{/} B^{/}$

Combining the four 1s we get the term: CD

F (A, B, C, D) = $A^{/} B^{/}$ + CD is the solution in SOP.

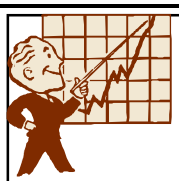
Step 6 : The square marked with 0's represent the complement of the function. Combining 0's and Xs in fig (b) we get:

$F^{/} = D^{/} + AC^{/}$

$(F^{/})^{/} = F = (D^{/} + AC^{/})^{/}$

$= D (A^{/} + C)$ - (applying De Morgan's theorem)

This is the simplified POS form of the function.



CHECK YOUR PROGRESS

Q.1. What is a minterm?

Q.2. What is a maxterm?

Q.3. What is a Karnaugh map?

Q.4. What do you understand by sum of products and product of sums?

Q.5. How are the Don't care conditions useful?



9.5 LET US SUM UP

- The three basic logic functions and operations are AND (logical multiplication), OR (logical addition), and the NOT (logical complementation) operation.
- Boolean addition is same as logical OR and Boolean multiplication is same as logical AND operation.
- We can simplify a Boolean function using algebric manipulation method and map method. There is no straight forward procedure in the algebric manipulation method.
- The K-map method is simple systematic and straight forward.
- Any Boolean function can be expressed as a sum of minterms (SOP) and as a product of maxterms (POS). Minterms are AND terms and maxterms are OR terms.
- The variables in a product term (minterm) and in a sum term (maxterm) may appear either in a complemented form or a normal form.
- A function is said to be completely specified, when for every possible combination of input variables the output is defined, otherwise the function is incompletely specified.
- In an incompletely specified function those combination of input variables for which output values are not known i.e. which minterms

or maxterms are not used as parts of the output function are called Don't care terms.

- Don't care terms may be considered as 1s or as 0s, in simplifying a Boolean function.



9.6 FURTHER READING

- Mano, M.M. (1982). Computer System Architecture, Englewood Cliffs, N J, 53-54.
- Mano, M.M. (2017). Digital logic and Computer Design, Pearson Education India
- Ram, B. (2000). Computer Fundamentals Architecture and Organization, New Age International.



9.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : A minterm is an AND term also called product term Minterms are obtained by combining the variables, which may appear in their normal form or complement form, with an AND operation. For example, for the two variables "a" and "b", we have four minterms, viz, $a'b'$ (m_0), a' b (m_1), a b' (m_2) and a b (m_3).

Ans. to Q. No. 2 : A maxterm, on the other hand an OR term, also called sum term. Maxterms are obtained by combining the variables with an OR operation. As for example, for two variables "a" and "b", we have four maxterms and they are: a+b (M_0), a+b' (M_1), a'+b (M_2), and a'+b' (M_3).

Note : In case of maxterm, observe that – when a=0, b=0 the maxterm is: a + b i.e. a and b appears in normal form.

Ans. to Q. No. 3 : The Karnaugh map is a pictorial or graphical method used to simplify a Boolean function in an orderly and

systematic manner. It is convenient to use 2 to maximum 6 variable map, but complexity in handling the map increases as the number of variables increase.

Ans. to Q. No. 4 : We have already seen that minterms are product terms. One of the important properties of Boolean algebra is that: any Boolean function can be expressed as a sum of minterms/product terms (SOP). Those minterms, which are present in a function are combined with OR operation and this is called sum of product representation of the function. Example: F (x, y z) = $x^{/}$ y z + x $y^{/}$ z + x y z.

On the other hand, product of sum (POS) form of a Boolean function is obtained by combining the maxterms present in the function with AND operations, Example: F (x, y, z) = (x + $y^{/}$ + z) ($x^{/}$ + y + z) ($x^{/}$ + $y^{/}$ + z).

Note : Since each maxterm is the complement of its corresponding minterms and vice versa, the SOP form of Boolean function is also the complement of its POS form and vice versa.

Ans. to Q. No. 5 : Don't care conditions are plotted in the K-map using the symbol "X". If we can get a larger area by considering these Xs, (i.e. more squares can be combined) then we get a more simplified form of the function. Thus, don't care condition helps in further simplification of a function.



9.8 MODEL QUESTIONS

Q.1. How are the terms "maxterm" and "minterms" related to each other?

Q.2. What is the canonical form of a Boolean function?

Q.3. What is the standard form of a Boolean function?

Q.3. How can you convert a function from one canonical form to other?

Q.5. The minterms of a 4 variable map are marked as: m_0, m_1, m_3, m_2 (in the first raw), not as m_0, m_1, m_2, m_3. Why?

Q.6. Simplify the following Boolean functions in SOP form.

a) F (a, b, c) = (2, 3, 6, 7)

b) F (w, x, y, z) = (7, 13, 14, 15)

c) F (a, b, c, d) = (2, 3, 12, 13, 14, 15)

d) $x\ y\ z' + x\ y\ z + x'\ z'$

e) $a\ b'\ c + a\ b\ c' + a'\ b\ c + a\ b\ c$

Q.7. Consider the following truth table.

a	b	c	F_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

a) Express the function F_1 in product of sum form.

b) Obtain the simplified function in sum of products.

c) Obtain the simplified function in product of sums.

Q.8. Obtain the simplified expression in POS.

a) F (a, b, c) = π (0, 1, 4, 5)

b) F (A, B, C, D) = π (0, 1, 2, 3, 4, 10, 11)

Q.9. Simplify the Boolean function F using the Don't care condition d in

(1) SOP and (ii) POS.

(a) $F = w'\ x'\ z' + w'\ y\ z + w'\ x\ y$

$d = w'\ x\ y'\ z + w\ y\ z + w\ x'\ z'$

(b) $F = A'\ B'\ C + A'\ B'\ C' + A'\ B\ C\ D + B'\ C\ D' + A\ B'\ D'$

$d = A'\ B\ C'\ D + A'\ B\ C\ D' + A\ C\ D$

****

UNIT 10 : COMBINATIONAL CIRCUITS

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.3 Combinational Circuit

10.4 Multiplexer

- 10.4.1 2-to-1 Multiplexer
- 10.4.2 4-to-1 Multiplexer
- 10.4.3 8-to-1 Multiplexer
- 10.4.4 16-to-1 Multiplexer
- 10.4.5 Multiplexer Application

10.5 De Multiplexer

- 10.5.1 1 to 16 De Multiplexer

10.6 Encoder

- 10.6.1 Octal to Binary Encoder
- 10.6.2 Decimal to BCD Encoder

10.7 Decoder

10.7. 13-to-8 Decoder

10.8 Magnitude Comparator

10.9 Let Us Sum Up

10.10 Further Reading

10.11 Answers to Check Your Progress

10.12 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define and describe multiplexer and de-multiplexer
- describe different types of multiplexer and de-multiploxer
- identify encoder and decoder
- describe magnitude comparator

10.2 INTRODUCTION

In the previous unit, we have learnt about the reduction techniques. Maxterms and minterns, along with methods to simplify Boolean functions are discussed in the previous unit. Map method and algebraic manipulation method here discussed in detail. In this unit, you will be introduced to the concept of combinational circuit. Multiplexer, De-multiplexer, encoder and decoder are also introduced in this unit. The digital circuits in which the status of the output at any instant of time solely depend on the status of the input at that instant are called combinational circuits. Different combinational circuits are discussed in detail in this unit. In the next unit, binary arithmetic circuits will be explored in detail.

10.3 COMBINATIONAL CIRCUIT

A combinational circuit is a collection of some interconnected set of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs.

A combinational circuit consists of input variables, logic gates, and output variables. The logic gates accept binary signals from the inputs and generate some binary signals to the outputs. Here, binary signals mean only two possible values, one representing logic 1 and the other logic 0. A combinational circult can be represented as a network of gates and, therefore, can be expressed by a truth table or boolean expression. A block diagram of combinational circuit is shown in Fig. 10.1.

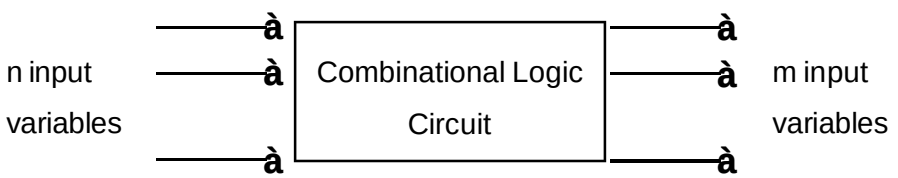


Figure 10.1 Block Diagram of Combinational Circuit

Here, n input binary variables come from an external source and the m output variables go to an external destination. For n input variables, there are 2n possible combinations of binary input values. For each possible input combination, there is only one posible output combination. A combinational circuit can be described by m boolean functions, one for

each output variable. Each output function is expressed in terms of the n input variables.

The basic design issue related to combinational circuits is: the minimization of number of gates. The normal circuit constraints for combinational circuit design level.

- the depth of the circuit should not exceed a specific level.
- number of input to a gate and to how many gates its output can be fed.

10.4 MULTIPLEXER

In the simplest term multiplexer is a combinational circuit which perform sharing. A digital data multiplexer is a logic circuit that has many inputs and only one output. It can select any one of its many inputs by applying a control signal and steer the selected input to the output. A multiplexer is also called a data selector. Generalized block diagram of a multiplexer is shown in Figure 10.2

It has 2^n inputs where, n is the numbers of control lines and only one output. A multiplexer is also called a many – to – one data selector.

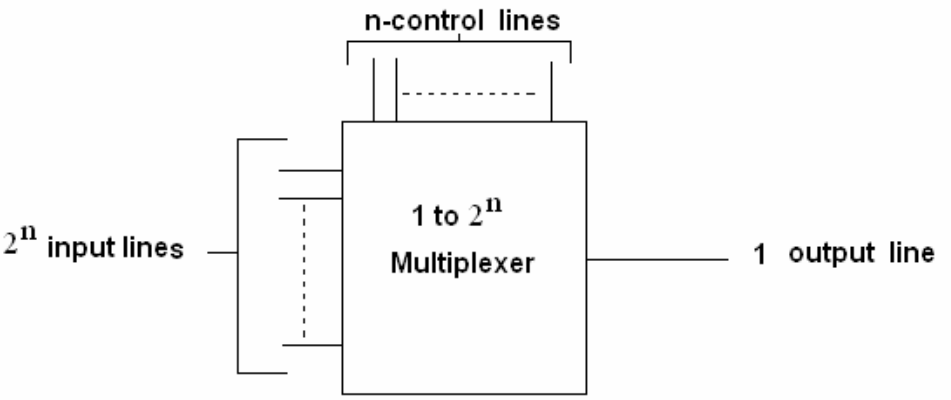


Figure 10.2 : Block Diagram of Multiplexer

10.4.1 2-to-1 Multiplexer

A 2-to-1 multiplexer has 2 inputs and a single control line as shown in Figure 10.3. Here, data inputs are denoted by I_0 and I_1. The control signal A specifies which input is routed to the output. When A = 0, then I_0 is routed to the output and when A = 1 then I_1 is routed to the output.

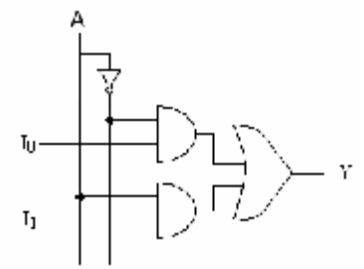


Figure 10.3 : 2-to-1 Multiplexer

10.4.2 4-to-1 Multiplexer

To construct a 4-to-1 multiplexer we use two control signals which can control four inputs. The two bit binary number at the control A and B specifies which of the four data inputs is to go to the output. The four data inputs are termed as I_0, I_1, I_2 and I_3. If AB = 00, then I_0 is allowed to go to the output. If AB = 01, the I_1 is allowed to go to the output. In this manner, AB = 10 will take I_2 and AB = 11 will take I_3 to the output. The circuit for the 4-to-1 multiplexer is shown in Figure 10.4.

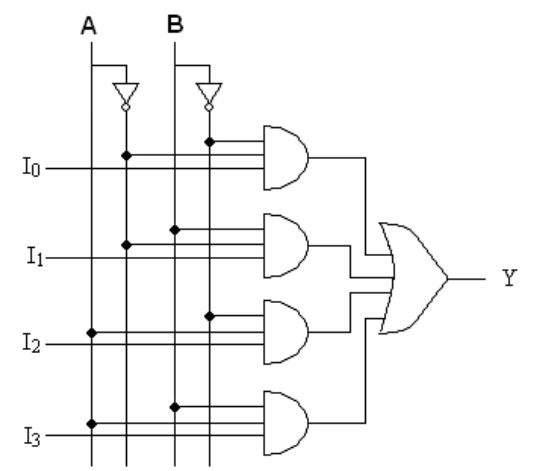


Figure 10.4 : 4-to-1 Multiplexer

10.4.3 8-to-1 Multiplexer

Figure 10.5 shows a 8-to-1 multiplexer, where there are 8 inputs, 3 control lines and 1 output. The eight inputs are labeled as $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$ and the control lines are as A, B, C. Here, which input is steered to the output depends on the value of ABC. For example, if ABC

= 000 then the upper AND gate is enabled and all other AND gates are disabled. As a result, the input I_0 alone is steered to the output. Similarly if ABC = 110, then the AND gate connected to the data line I_6 is enabled while all the other AND gates are disabled. Therefore, the input I_6 appears at the output. Hence when ABC = 110 , the output is Y = I_6

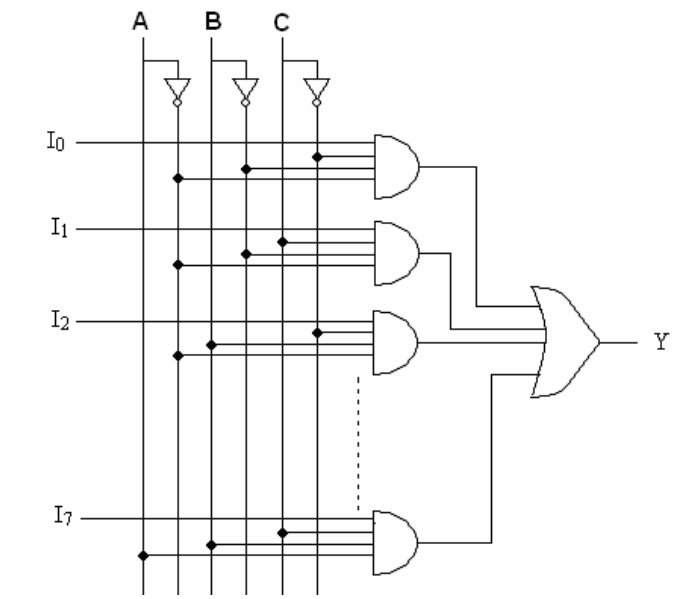
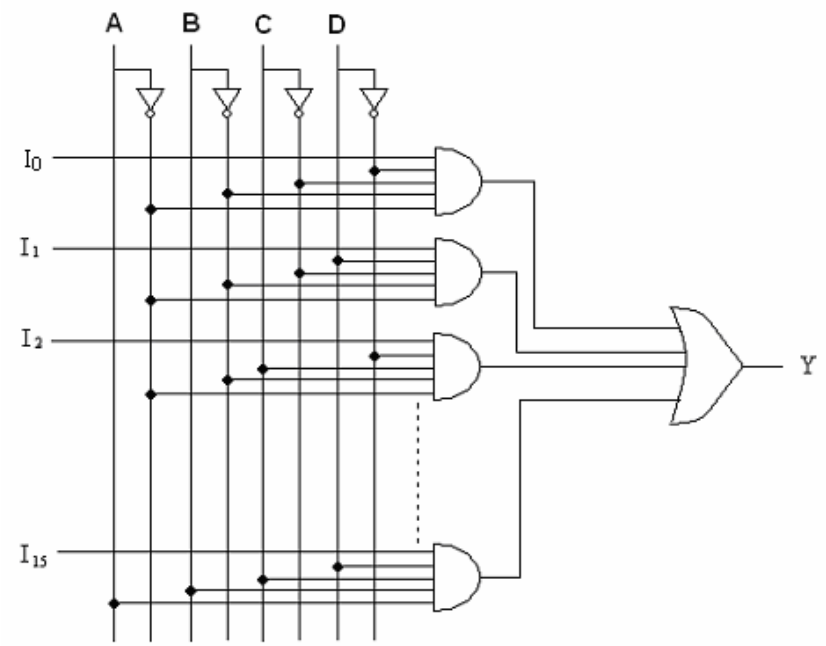


Figure 10.5 : 8-to-1 Multiplexer

10.4.4 16-to-1 Multiplexer

Fig 10.6 shows a 16 – to – 1 multiplexer. In this circuit, there are 16 data input lines, 4 control lines and 1 output. The input lines are denoted by $I_0, I_1, I_2, I_3 \ldots\ldots\ldots\ldots I_{15}$ and the control lines are by ABCD. The output is denoted by Y.

Out of 16 input lines, only one is transmitted to the output depending upon the value of ABCD. If ABCD = 0000 then I_0 is steered to the output since the upper AND gate is enabled alone and all others are disabled. Similarly, if ABCD = 0010 then I_2 appears at the output. If ABCD = 1111 then the last AND gate is only enabled and therefore I_{15} appears at the output.



10.4.5 Multiplexer Application

I Function Generator

A multiplexer can perform any logic function which otherwise needs logic gates to implement. It can be connected in a manner so it duplicates the logic of any truth table. In such application, the multiplexer is viewed as a function generator. One advantage of this use of a multiplexer is that a single IC can perform a function that might need many ICs. Another advantage is that the implemented function can be easily changed if required due to changing necessity.

In the design of a function generator using a multiplexer first the truth table is to construct that is corresponding to the logic expression to be implemented. The next step is to connect logic 1 to each input of the multiplexer corresponding to each combination of input variables for which there is 1 in the output of the truth table. Then, to connect logic 0 to all remaining inputs of the multiplexer. The variables of the truth table are used as control signals. The application can be best understood by one example. Here, we implement the logic function

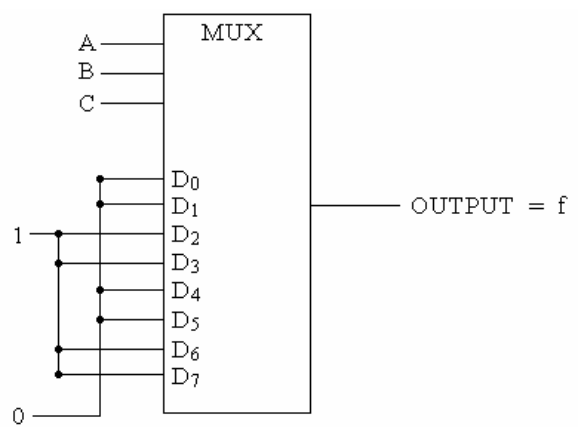
$$f = AB + \overline{A}BC + B\overline{C}$$

The truth table for the function is :

Truth Table

A	B	C	AB	$\bar{A}BC$	$B\bar{C}$	$f = AB + \bar{A}BC + B\bar{C}$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	1
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	1	1
1	1	1	1	0	0	1

Since the expression has three variables we use these three variables as three control or selector signals. As such the multiplexer must have $2^3 = 8$ data inputs line. The Boolean expression has output 1 for ABC = 010, 011, 110 and 111. We therefore connect logical 1 to data inputs 2, 3, 6 and 7 of the multiplexer and logical 0 to all the remaining inputs. Figure 10.7 shows the circuit of the function generator for the chosen Boolean expression.



Figurer 10.7 : Multiplexer as a function generator

Multiplexing Seven-Segment Display

Seven-segment displays are often used as optical output display in many applications. These displays consume considerable power. To minimize power consumption in small portable device like pocket calculator,

where several such displays are to be illuminated simultaneously, multiplexers are used. In this technique, all the displays are not illuminated simultaneously, instead they are illuminated one after another in a round robin fashion. If all the displays are illuminated about 30 times per second, the human eye will not be able to detect any flicker, rather the displays appear as illuminated steadily. The power consumption is greatly reduced in this technique.

10.5 DE-MULTIPLEXER

A demultiplexer steers a single input to one of many outputs. It is opposite in function to the multiplexer. De-multiplexer means One-to-Many. It can be visualized as a distributor, since it distributes the same data to different output terminals. As such it has 1 input and many outputs. With the application of appropriate control signal, the common input data can be steered to one of the output lines. A good example of demultiplexer is the distributor of an automobile ignition system where electric pulse is distributed among different fuel ignition plugs. Fig 10.8 shows a generalized block diagram of a de-multiplexer. To have 2^n output lines, there must be n control lines in a de-multiplexer.

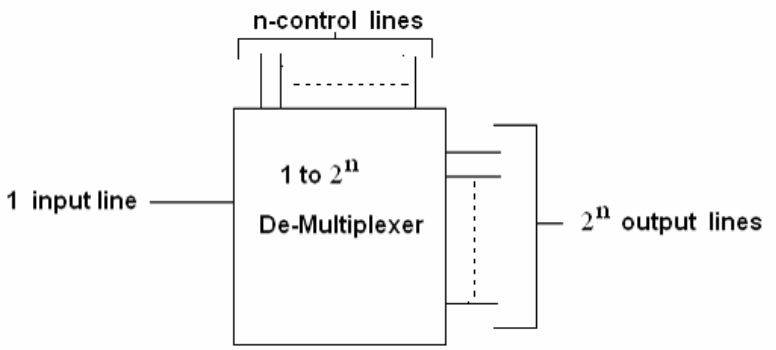


Figure 10.8 : Block diagram of a De-Multiplexer

10.5.1 1-to-16 De-Multiplexer

In Figure 10.9 we have shown a 1-to- 16 de-multiplexer.

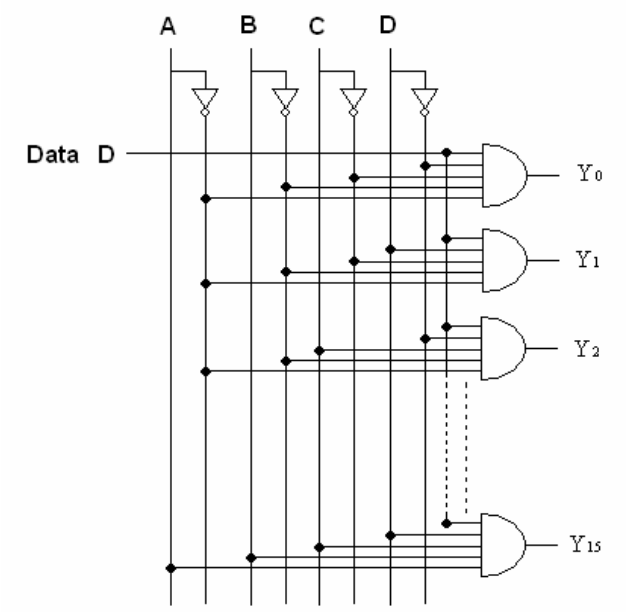


Figure 10.9 : 1-to-16 De-Multiplexer

Here the data input line is denoted by D. This input line is connected to all the AND gates through which output appears. Depending upon the control signal, only one AND gate becomes enabled and the data input D appears through that AND gate. So, when ABCD = 0000, the upper AND gate is enabled and data input D appears at Y_0 as output.

When ABCD = 1111, the bottom AND gate becomes enabled and D appears at Y_{15} as output. For other combination, D appears at other output terminal.



CHECK YOUR PROGRESS

Q.1. Multiplexer means :

(a) multiple to many (b) one to many

(c) many to one (d) one to one

Q.2. A 16-to-1 multiplexer has

(a) 1 control lines (b) 2 control lines

(c) 3 control lines (d) 4 control lines

Q.3 De-multiplexer means

(a) deduct multiple bits (b) one-to-many

(c) multiple-to-multiple (d) one-to-one

Q.4. To implement the logic function $F = AC + BC + \overline{A}\overline{B}\overline{C}$ we need a

(a) 2-to-1 multiplexer (b) 4-to-1 multiplexer

(c) 8-to-1 multiplexer (d) 16-to-1 multiplexer

10.6 ENCODER

An encoder is a combinational circuit which has 2n or less input lines and n output lines. An example of an encoder is the octal to binary encoder which has eight inputs, one for each of the eight digit and three outputs that generates the corresponding binary number. A generalized view of an encoder is shown in Figure 10.10.

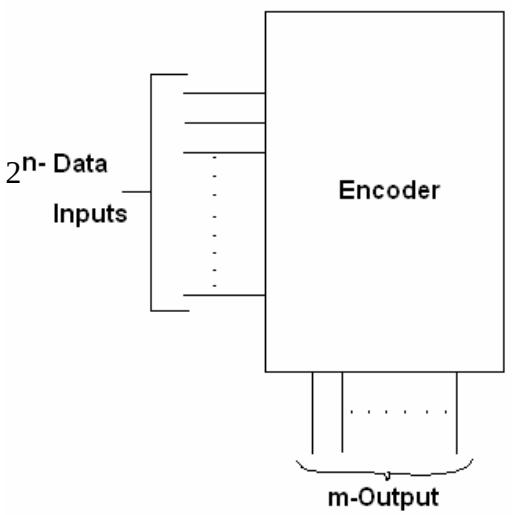


Figure 10.10 : Block Diagram of an Encoder

In the figure we can see that there are 2^n input lines and n – numbers of output lines. Out of inputs, only one input line is active at a time. Encoder generates a coded output which is unique for each of the active input.

10.6.1 Octal-To-Binary Encoder

An octal-to-binary encoder is a device which takes octal number as input and generates the equivalent binary number. Figure 10.11 shows a circuit of an octal-to-binary encoder. It is assumed here that any one input keys is pressed at a time. The keys are connected to a +5 Volt. When the input key is pressed, 5 volt signal is passed to the horizontal lines to which three OR gates are connected selectively.

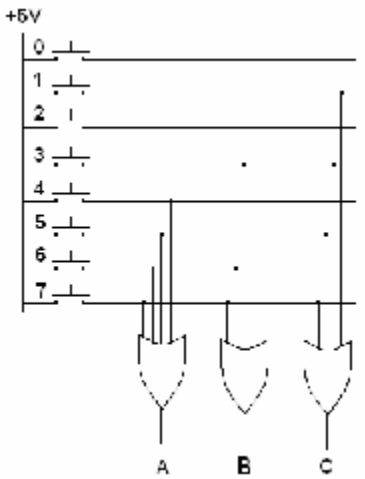


Figure 10.11 : Octal-to-Binary encoder

10.6.2 Decimal-To-BCD Encoder

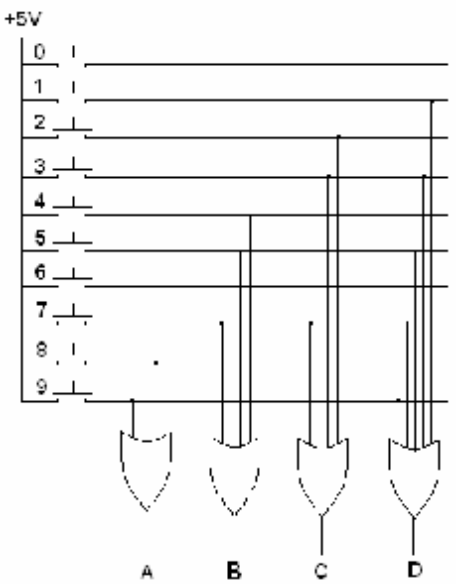


Figure 10.12 : Decimal-to-BCD Encoder

A decimal to BCD encoder is shown in Fig 10.12. This circuit generates BCD output when any one of the push button switches is pressed. As for example, if button 6 is pressed, the B and C OR gates have high ABCD = 0110. If button 8 is pressed, the OR gate A receives a high input and therefore the output becomes ABCD = 1000.

10.7 DECODER

A decoder is a digital circuit which has n-input lines and 2^n or less output lines. A decoder and a de-multiplexer share similarity. In a de-multiplexer, there is a single input line connected to every output 'AND' gate whereas in a decoder the input line is absent. Various kinds of decoder are constructed to decode signals given in different forms. The input to a decimal decoder is a binary code and the output is the decimal digit the code represents. Some decoders take 3 bits binary inputs and give octal outputs.

In practical applications, decoders are used to select one of many devices in a computer system. For example, a computer system has many peripheral devices like hard disk drive, DVD drive, printer etc for each of which there exist an unique address. Generally all the devices are connected to the computer system through a decoder. The computer generates the address of a particular device when it wants to access it in the form of binary code. The decoder receives that address as input and activates the lines connecting the particular device to make it enable.

10.7.1 3-to-8 Decoder

A 3 – to – 8 decoder is shown in Fig 10.13. It's truth table is shown below :

Truth table for a 3-to-8 decoder

Inputs			Outputs							
A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_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

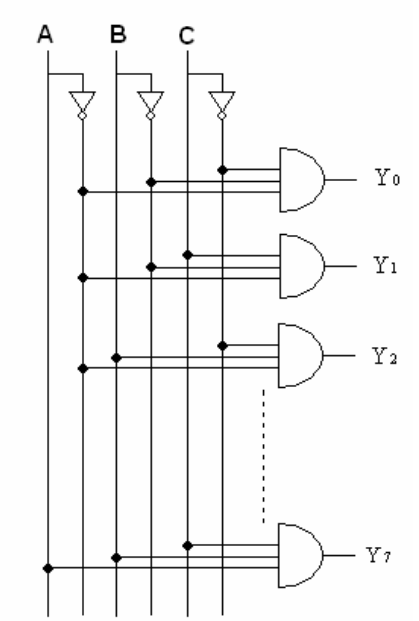


Figure 10.13 : 3- to -8 Decoder

10.8 MAGNITUDE COMPARATOR

A magnitude comparator circuit can compare two binary numbers to determine whether one is greater than the other or if they are eqqual. Such a magnitude comparator has three output lines for A > B, A = B, A < B where A and B are two n-bits binary numbers. Every bit of one number is compared with the corresponding bit of the other number by ExOR gate.

A 4-bit magnitude comparator, SN 7485 is available in chip form the block diagram of which is shown in Fig 10.14. It compares two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$. Three output terminals are available for A < B, A = B and A > B.

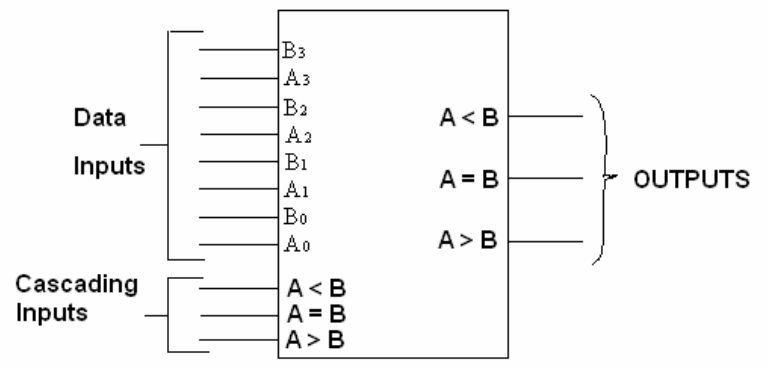


Figure 10.14 : Two 4-bit words magnitude comparator SN 7485

FUNCTION TABLE SN 7485

Comparing Inputs				Cascading Inputs			Outputs		
A_3B_3	A_2B_2	A_1B_1	A_0B_0	A > B	A < B	A = B	A > B	A < B	A = B
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	H	L	L	H

To compare any number having more than 4-bits, two or more such chips can be cascaded. The A > B, A < B and A = B outputs of a stage that handles less significant bits are connected to the corresponding cascading inputs of the next stage that handles the more significant bits.



CHECK YOUR PROGRESS

Q.5. An encoder :

(a) converts a digital input to another form of digital output.

(b) converts analog input to digital output

(c) selects one out of many inputs (d) none of these.

Q.6. Decoder has n inputs line and

(a) n output lines (b) 2^n output lines.

(c) n^2 output lines (d) no output lines.

Q.7. Magnitude comparator

(a) compares two multi bit binary number

(b) magnify any digital signal

(c) compress binary numbers.

(d) check error in a binary number.



10.9 LET US SUM UP

- A combinational circuit is some combinations of logic gates as per specific relationship between inputs and outputs.
- A multiplexer is a combinational circuit which selects one of many inputs.
- Demultiplexer is opposite to a multiplexer.
- An encoder generates a binary code for 2^n input variables.
- A decoder decodes an information receives from n input lines and transmits the decoded information to maximum outputs.
- A magnitude comparator circuit can compare two binary numbers to determine which one is greater than the other or their equality.



10.10 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



10.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (c)

Ans. to Q. No. 2 : (d)

Ans. to Q. No. 3 : (b)

Ans. to Q. No. 4 : (c)

Ans. to Q. No. 5 : (a)

Ans. to Q. No. 6 : (b)

Ans. to Q. No. 7 : (a)



10.12 MODEL QUESTIONS

Q.1. Define combinational circuit with example.

Q.2. Using truth table and logic diagram explain the working of a full-adder circuit.

Q.3. Using truth table and logic diagram explain the working of a full-subtractor circuit.

Q.4. Draw and explain the operation of a 4-bit parallel adder.

Q.5. Explain how a serial adder works using block diagram.

Q.6. Define multiplexer? Using diagram explain the working of a 8-to-1 multiplexer.

Q.7. Use an 8-to-1 multiplexer to generate the logic function

Q.8. Explain the principle of an encoder. Draw a decimal-to-BCD encoder.

Q.9. What is a decoder? Draw and explain the working of a 3-to-8 decoder.

Q.10. What is magnitude comparator? Draw block diagram and the function table of the magnitude comparator SN 7485.

****

UNIT 11 : BINARY ARITHMETIC CIRCUITS

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Half-Adder

11.4 Full-Adder

11.5 Parallel Adder

11.6 Serial Adder

11.7 Subtractor Circuit

- 11.7.1 Half-Subtractor
- 11.7.2 Full-Subtractor

11.8 Let Us Sum Up

11.9 Further Reading

11.10 Answer to Check Your Progress

11.11 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit you will be able to :

- describe working principle of half-adder
- describe working principle of full-adder
- describe working of parallel adder
- describe working of serial adder
- describe working principle of half-subtractor
- describe working principle of full-subtractor

11.2 INTRODUCTION

In the previous unit, you have already been accustomed with different logic gates and their associated boolean expressions. Understanding of these would help you to follow the subject matter of this unit.

The advancement in technology has made it possible to fabricate

cheap **SSI (Small Scale Integration)** circuits, **MSI (Medium Scale Integration)** functional devices and **LSI (Large Scale Integration)** systems. Example of these are single chip pocket calculator, single chip computer memories and single chip CPU etc. Many of these products now become building blocks to design other higher level digital devices.

Adder circuits are used to add binary bits. If we require to find sum of two 1-bit we use a circuit called *half-adder*. Sometimes it becomes necessary to add three 1-bit binary numbers. The circuit for this is called *full-adder*.

11.3 HALF-ADDER

In the truth table, the *sum(S)* and *carry(C)* bits are shown as result which are obtained when two 1-bit number *X* and *Y* are added. The truth table contains all the possible combination of values of these two numbers. From the table two individual circuits can be constructed to obtain *sum* and *carry* in response to every combination of the two input numbers. Combining both a single circuit is constructed to treat it as a single combinational circuit to give us two outputs viz *sum* and *carry.* The circuit is generally termed as *half-adder.*

Hence we can say that a half-adder is a circuit that can add two binary bits. Its outputs are SUM and CARRY.

Truth Table for a Half-Adder

X	Y	CARRY (C)	SUM (S)
0	0	0	0
0	1	0	1 ($\bar{X}Y$)
1	0	0	1 ($X\bar{Y}$)
1	1	1 (XY)	0

The minterms for Sum and CARRY are shown in the bracket. The Sum-Of-Product equation for SUM is :

$$S = \bar{X}Y + X\bar{Y} \quad \ldots\ldots\ldots\ldots\ldots (1)$$

Similarly the SOP equation for the CARRY is :

$$C = XY \quad \ldots\ldots\ldots\ldots\ldots (2)$$

Combining the logic circuits for equation (1) & (2) we get the circuit for Half-Adder as :

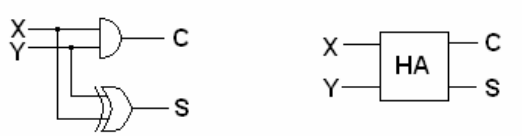


Figure 11.1 : Half-Adder Circuit and Symbol

11.4 FULL-ADDER

To add two 2-bit binary numbers, we first add the two least significant bits of each number and take the carry (if any) that generate into the addition of the two most significant bits. Hence the second addition involves addition of three single bits. That necessitates another type of addition circuit which we will discuss here. This new type of adder circuit is known as *full-adder*.

Full-Adder is a logic circuit to add three binary bits. Its outputs are SUM and CARRY. In the following truth table X, Y, Z are inputs and C and S are CARRY & SUM.

Truth Table for Full-Adder

X	Y	Z	CARRY (C)	SUM (S)
0	0	0	0	0
0	0	1	0	1 ($\bar{X}\bar{Y}Z$)
0	1	0	0	1 ($\bar{X}$**Y**$\bar{Z}$)
0	1	1	1 (($\bar{X}$**Y**Z	0
1	0	0	0	1 ($X\bar{Y}\bar{Z}$)
1	0	1	1 ($X\bar{Y}Z$)	0
1	1	0	1 ($XY\bar{Z}$)	0
1	1	1	1 (XYZ)	1 (XYZ)

The minterms are written in the brackets for each '1' output in the truth table. From these the SOP equation for full summation can be written as :

$S = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$

$= \bar{X}(\bar{Y}Z+Y\bar{Z}) + X(\bar{Y}\bar{Z}+YZ)$

$= \bar{X}S + XS$..... (3)

Here S is SUM of Half-Adder.

Again SOP equation for Full – Adder CARRY is :

$C = \bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ$

$= \bar{X}YZ + XYZ + X\bar{Y}Z + XY\bar{Z}$

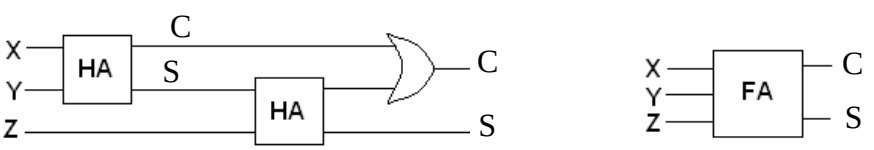
$= (\bar{X} + X)\, YZ + X\,(\bar{Y}Z + Y\bar{Z})$

$= YZ + XS$

$= C + XS$ ..... (4)

Here also, C means CARRY of half-adder and S means SUM of half-adder.

Now using two half-adder circuits and one OR gate we can implement equation (3) and (4) to obtain a full-adder circuit as follows.



Figurer 11.2 : Full-Adder Circuit and Symbol

11.5 BINARY PARALLEL ADDER

When two multi-bit binary numbers are to be added under the situation that their bits are available in parallel form, then a parallel adder is used. It can add each corresponding bit of the numbers and simultaneously produce the sum bits as parallel output.

I 4-BIT BINARY PARALLEL ADDER

Block diagram of a 4-bit binary parallel adder capable of adding two 4-bit numbers is shown in Fig 11.3. The numbers to be added are designated as $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ and the corresponding sum bits are $S_3S_2S_1S_0$. To build the parallel adder, one full-adder is required for each pair of corresponding bits in the numbers to be added, except for the pair of least significant bits, for which a half-adder is sufficient. But to facilitate cascading the pair of least significant bits is also added by a full-adder. The carry out from each stage is taken as the carry into the next more significant stage. In figure 11.3 we use four full-adder to construct the 4-bit binary

parallel adder where the carry in to the full-adder for the least significant bits is made logical 0.

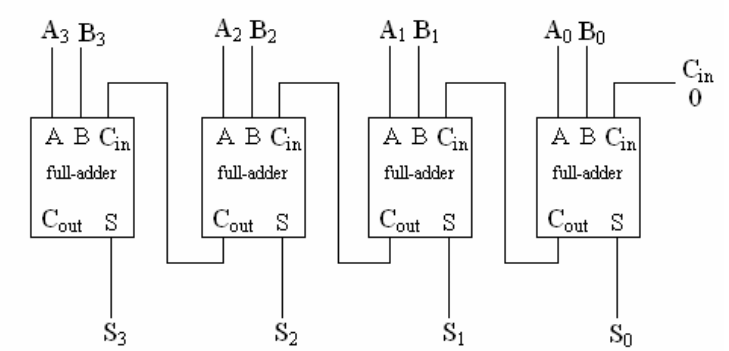


Figure 11.3 : A 4-bit binary parallel adder

11.6 SERIAL ADDER

A serial adder is used to add serial binary numbers. To implement a serial adder, one full-adder is required. In figure 11.4, a serial adder is shown. The bits to be added come serially one after another from both the numbers in synchronism, first A_0 and B_0, then after one clock pulse A_1 and B_1 and so forth. The carry bit generated (C_{out}) in the process is saved for one clock pulse by a flip-flop and it is added to the next higher order pair of input bits as C_{in}. It can be seen from the figure that the output sum bits are shifted into the output register as the input bits are shifted out of the input register. In practice, to minimize the circuit, the output bits are shifted in to one of the input registers behind the data being shifted out. The register which contains one of the binary numbers to be added before the commencement of addition and contains the sum after the addition process is completed, is called an *accumulator*.

It can be easily understood that the serial adder is slower than a parallel adder, as they require one clock pulse to add one pair of bits. A serial adder requires much lesser hardware than a parallel adder and hence it finds application in such devices where space is of much concern than speed. One such application is pocket calculator.

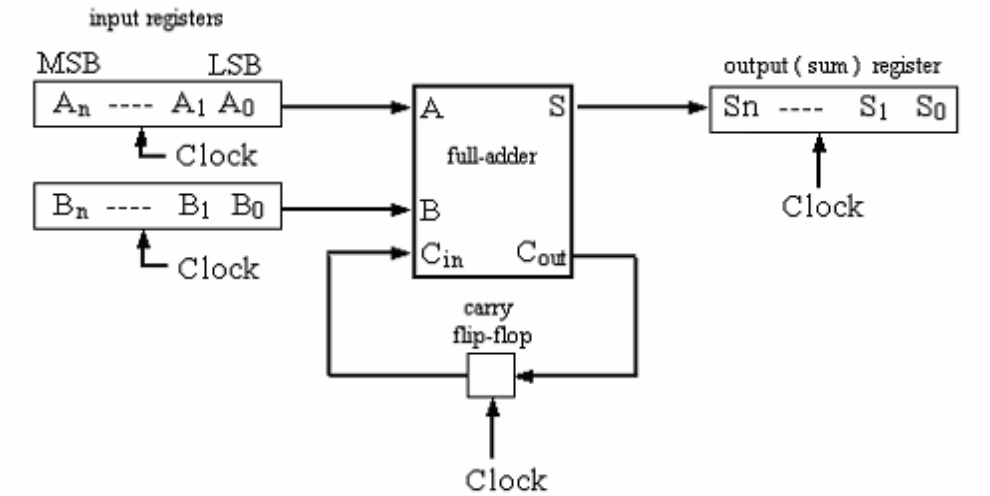


Figure 11.4 : A serial adder

11.7 SUBTRACTOR CIRCUIT

In arithmetic expression subtraction is also a frequently occurring operation like addition. Hence it is necessary to design subtractor circuits involving two and three binary bits. Just like adder circuits, we have two subtractor circuits viz *half-subtractor* and *full-subtractor*.

11.7.1 Half-Subtractor

A half-subtractor subtracts one bit from another bit. It has two outputs viz DIFFERENCE (D) and BORROW (B).

Truth Table for Half-Subtractor

X	Y	BORROW (B)	DIFFERENCE (D)
0	0	0	0
0	1	1 ($\bar{X}$Y)	1($\bar{X}$Y)
1	0	0	1 (X$\bar{Y}$)
1	1	0	0

The mean terms are written within parenthesis for output 1 in each column. The SOP equations are :

$D = \bar{X}Y + X\bar{Y}$

$= S$ ..... (5)

$B = \bar{X}Y$ ..... (6)

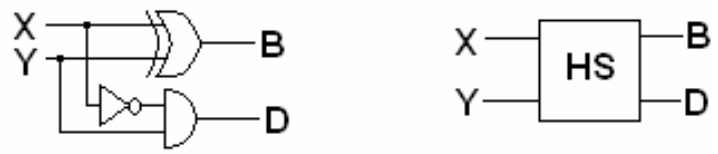


Fig. 11.5 : The half-subtractor circuit and symbol

11.7.2 Full-Subtractor

A full-subtractor circuit finds the difference and borrow for the subtraction operation involving three binary bits.

Truth Table of Full-Subtractor

X	Y	Z	BORROW (B′)	DIFFERENCE(D′)
0	0	0	0	0
0	0	1	1 ($\overline{X}\overline{Y}Z$)	1 ($\overline{Y}\overline{X}Z$)
0	1	0	1 ($\overline{X}Y\overline{Z}$)	1 ($\overline{X}Y\overline{Z}$)
0	1	1	1 ($\overline{X}YZ$)	0
1	0	0	0	1 ($X\overline{Y}\overline{Z}$)
1	0	1	0	0
1	1	0	0	0
1	1	1	1 (XYZ)	1 (XYZ)

The SOP equation for the DIFFERENCE is :

$D' = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$

$= \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ + \overline{X}\overline{Y}Z$

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

$= D\overline{Z} + DZ$ ..... (7)

And SOP equation for BORROW is :

$B/ = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + \overline{X}YZ + XYZ$

$= \overline{X}\overline{Y}Z + XYZ + \overline{X}Y\overline{Z} + \overline{X}YZ$

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

$= \overline{D}Z + \overline{X}Y$ ..... (8)

In equation (7) and (8), D stands for DIFFERENCE of half-subtractor. Now from the equations (7) and (8) we can construct a full-subtractor using two half-subtractor and an OR gate.

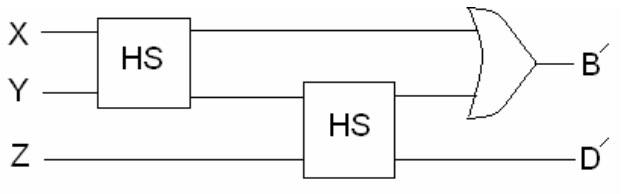
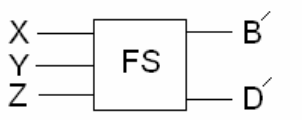


Fig. 11.6 : Full-Subtractor Circuit

The symbol of full-subtractor is :





CHECK YOUR PROGRESS

Q.1. A half-adder can add :

(a) Two binary number of 4 bit each

(b) Two binary bit

(c) Add half of a binary number

(d) None of these

Q.2. A full-adder is a logic circuit that has two output namely:

(a) product & sum

(b) sum & borrow

(c) sum & carry

(d) carry & borrow

Q.3. A parallel adder can be implemented using

(a) Two or more half-adder

(b) Two or more full-adder

(c) One full-adder and remaining half-adder

(d) none of these.

Q.4. A serial adder use

(a) One full-adder

(b) One half-adder

(c) One full-adder with a flip-flop

(d) One full-adder, one flip-flop, three shift registers

Q.5. A half-subtractor can perform

(a) subtraction of two binary bits

(b) product of two binary bits

(c) complement of half binary bits

(d) none of these

Q.6. A full-subtractor has the ability to do

(a) subtraction of two binary numbers

(b) subtraction of three binary bits

(c) product of three binary bits

(d) division three binary bits



11.8 LET US SUM UP

- Adder circuits are used to add binary bits.
- Subtractor circuits are used to subtract bits.
- Adder and subtractor circuits can perform binary addition and subtraction.
- There are different types of adder and subtractor like 2 bit adder/ subtractor, 3 bit adder/subtractor etc.
- Serial adder is used to add serial binary numbers.



11.9 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar, P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.

11.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (b)

Ans. to Q. No. 2 : (c)

Ans. to Q. No. 3 : (b)

Ans. to Q. No. 4 : (d)

Ans. to Q. No. 5 : (a)

Ans. to Q. No. 6 : (b)

11.11 MODEL QUESTIONS

Q.1. Use truth table and logic diagram to explain the working of a half adder and full-adder circuit.

Q.2. Use truth table and logic diagram to explain the working of a half and full-subtractor circuit.

Q.3. Draw and explain the operation of a 4-bit parallel adder.

Q.4. Explain how a serial adder works using block diagram.

Q.5. Give the logic diagram of half adder and half subtractor.

Q.6. Give the truth table for full subtractor.

****

UNIT 12 : SEQUENTIAL CIRCUITS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Sequential Circuits

12.4 Flip-Flop

- 12.4.1 RS Flip-Flop
- 12.4.2 D Flip-Flop
- 12.4.3 JK Flip-Flop
- 12.4.4 MS Flip-Flop

12.5 Let Us Sum Up

12.6 Further Reading

12.7 Answers to Check Your Progress

12.8 Model Questions

12.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define flip-flops
- describe the working principle of flip-flops
- describe different flip-flops like RS, D, JK and Master Slave flip-flop.

12.2 INTRODUCTION

In the previous units, we have learnt about different combinational circuits. We have learnt about multiplexer, de-multiplexer, encoder and decoders. We have also been introduced to binary arithmetic circuits like half adder, full adder, half subtractor and full subtractor.

In this unit, you will be introduced to sequential circuits and flip flops. Different flip flops like RS flip flop, D flip flop, JK flip flop, D flip flop and MS flip flop are covered in this unit. In this unit we will discuss several sequential circurits in detail.

12.3 SEQUENTIAL CIRCUITS

If the output of a circuit depends on its present inputs and immediate past output, then the circuit is called sequential circuit. To build a sequential circuit, we need memory circuits along with some other logic gates. Flip-Flops are used as memory circuit, the application of which we would see in counter, register etc.

We already know from unit 10, that a combinational circuit has no memory element. A combinational circuit has some basic differences with a sequential circuit. The most important difference is that sequential circuits has memory elements and the present value of outputs is determined by the present inputs and its present state. Some differences are listed below:

Combinational Circuit	Sequential Circuit
• It contains no memory element	• It contains memory elements
• The present value of its outputs is determined solely by the present values of its inputs	• The present value of its outputs is determined by the present values of its inputs and its present state
• Its behavior is described by the set of output functions	• Its behavior is described by the set of next-state functions and the set of output functions.

12.4 FLIP-FLOPS

A digital circuit that can produce two states of output, either high or low is called a multi-vibrator. Multivibrators are further divided into three types, viz. monostable, bi-stable and astable.

A Flip-Flop is a bi-stable multivibrator and therefore it has two stable states of output-either high or low. It is a device that can be triggered to high state or triggered to low state. Depending on its inputs and previous output, its new output is either high (or 1) or low (or 0). The applied inputs must be kept in the input terminals of the flip-flop so that the inputs can affect the output. Once the output is fixed, the inputs can be removed and then also

the already fixed output will be retained by the flip-flop. Hence a flip-flop can be used as a basic circuit of memory for storing one bit of data. To store multiple bits we can use multiple flip-flops. Such set of flip-flops are called register.

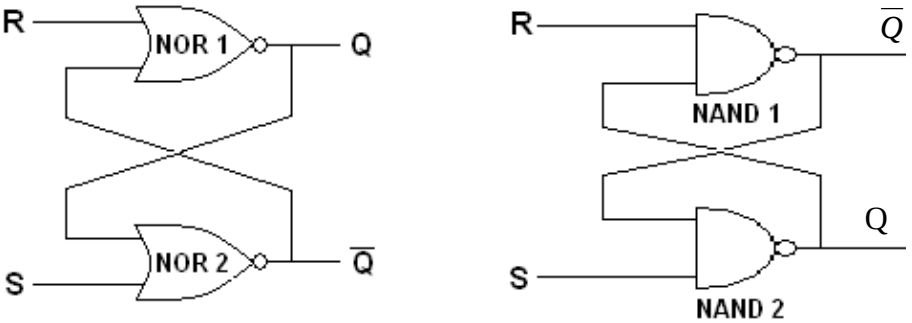
Flip-flop also serves as a fundamental component of another important digital circuit called counter. This device can keep track of events that occur in a digital circuit such as the number of pulses that a circuit can generate in a given time period. It can also divide frequency of a clock pulse which can lead to a digital clock circuit.

Flip-flops are constructed in a number of ways, each type having specific characteristics necessary for a particular application. Here we discuss the following types of flip-flops :

- RS Flip-flop
- D Flip-Flop
- JK Flip-Flop
- MS Flip-Flop

12.4.1 RS Flip-Flop

RS Flip-Flop is also called Set-Reset Flip-Flop. An RS flip-flop can be built using many different circuits. Here, we have shown two RS flip-flop circuits using NOR and NAND gate also called the basic flip flop. To study its working principle, we can take any one of them though their working principle differs slightly. Let us consider the RS flip-flop using NOR gate. Note that the NOR gates are cross-coupled, that is one of the inputs of each gate is the output of the other gate.



(a) **(b)**

Figure 12.1 : Basic circuit of RS Flip-Flop using NOR and NAND gate

RS flip-flop has two inputs—Set(S) and Reset(R). It has two outputs, Q and $\overline{Q}$. It should be noted $\overline{Q}$ is always the complement of Q. Though both outputs are not used in all the applications, the availability of both makes the flip-flop more versatile. Various combinations of inputs and their corresponding outputs are listed in the truth table below :

R	S	Q	Action
0	0	last value	No change
0	1	1	set
1	0	0	Reset
1	1	?	Forbidden

Truth Table of RS Flip-Flop

To understand the operation of the flip-flop let us assume that the flip-flop is initially *SET*, that means Q = 1 and $\overline{Q}$ = 0. Now the first input condition in the table i.e. R = 0, S = 0, will not change the state of the flip-flop. We can verify that by observing the fact that the input to the NOR gate 2 are 0 and 1, which keeps $\overline{Q}$ at 0. Since $\overline{Q}$ is 0, the input to the NOR gate 1 are both 0, making Q = 1. So we can say R = S = 0 has no effect on its output, the flip-flop retains its previous state. Hence Q remains unchanged.

The second input condition R = 0, S = 1 makes no change. The output of gate 2 remains low when both inputs are made high. Thus a 1 at the S input and 0 at R input will SET the flip-flop and Q will be equal to 1.

The third input condition R = 1, S = 0 will reset the flip-flop. The

input to gate 1 causes its output Q to change from 1 to 0. That, 0 is coupled to gate 2, which now has two low inputs. Therefore its output changes from 0 to 1, that 1 is coupled to gate 1 and therefore Q is held at 0. Hence, when R = 1, S = 0, then Q = 0, R = 1. Thus the flip-flop is RESET.

The last input condition in the table R = 1, S = 1 is forbidden since it forces both the NOR gates to the low state, means both Q = 0, and R = 0 at the same time, which violets the basic definition of flip-flop that requires Q to be the complement of $\overline{Q}$. Hence this input condition is forbidden and its output is unpredictable.

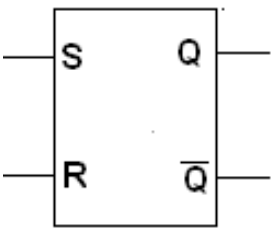
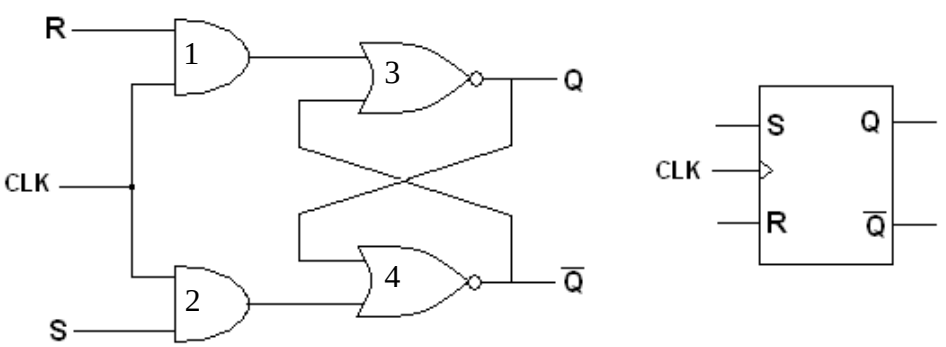


Figure 12.2 : Symbol of RS Flip-Flop

CLOCK INPUT : For synchronization of operation of multiple flip-flop, an additional signal is added to all types of flip-flop which is called clock signal, generally abbreviated as CLK. Addition of CLK signal ensures that, what ever may be the input to the flip-flop, it effects the output only when CLK signal is given. Figure 12.3 shows a clocked RS flip-flop.



(a) Clocked RS Flip-Flop **(b)Symbol of Clocked RS FF**

Figure 12.3 : RS flip-flop

SET-UP TIME : When the clock is low, the changes in R and S inputs does not effect the state of a flip-flop since the output of gate 1 and gate 2 (i.e. R and S input to the basic flip-flop) are zero or low. To effect the output, the R and S input must remain constant for a minimum time before a clock

pulse is applied. This minimum time is called *set-up time*. Its value is given in manufacturers' specifications.

12.4.2 D FLIP-FLOP

D flip-flop is a modification of RS flip-flop. In RS flip-flop when both the inputs are high i.e. R = 1, S = 1, the output becomes unpredictable and this input combination is termed as forbidden. To avoid this situation, the RS flip-flop is modified so that both the inputs can not be same at a time. The modified flip-flop is called D flip-flop. Figure 12.4 shows a clocked D flip-flop.

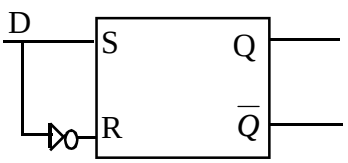


Figure 12.4 : (a) D flip-flop

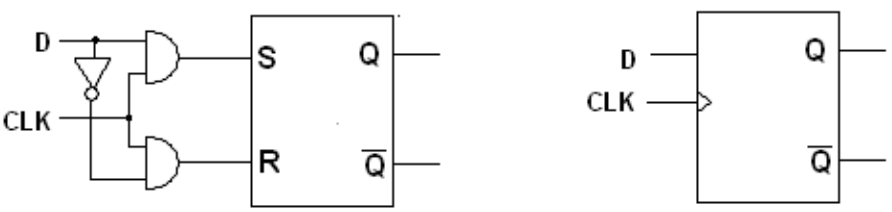


Figure 12.4 : (b) Clocked D Flip-Flop (c) Symbol of D Flip-Flop

In D flip-flop, both inputs of RS flip-flip are combined together to make it one by a NOT gate so that inputs can not be same at a time. Hence in D flip-flop there is only one input. The truth table is shown below:

CLK	D	Q
0	X	Last state
1	0	0
1	1	1

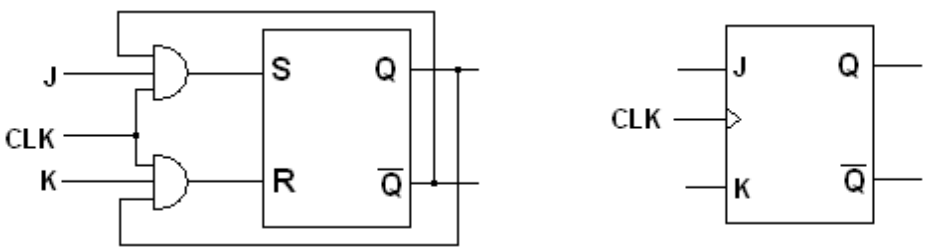
Truth Table of D F lip-Flop

In a clocked D flip-flop the value of D cannot reach the output Q when the clock pulse is low. During a low clock, both AND gates are disabled, therefore, D can change value without affecting the value of Q. On the

other hand, when the clock is high, both AND gates are enabled. In this situation, Q is forced to be equal to the value of D. In another way we can say that in the D flip-flop above, Q follows the value of D while the clock is high. This kind of D flip-flop is often called a D latch.

12.4.3 JK FLIP-FLOP

In RS flip-flop, the input R = S = 1 is called forbidden as it causes an unpredictable output. In JK flip-flop this condition is used by changing the RS flip-flop in some way. In JK flip-flop both input can be high simultaneously . Figure 12.5 shows one way to build a JK flip-flop.



(a) (b)

Figure 12.5 : (a) JK Flip-Flop, (b) Symbol of JK Flip-Flop

CLK	J	K	Q
X	0	0	Last state
↑	0	1	0
↑	1	0	1
↑	1	1	Toggle

Truth table for JK Flip-Flop

The inputs J and K are called control inputs because their combinations decide what will be the output of JK flip-flop when a +ve clock pulse arrives. When J and K are both low, both the AND gates are disabled. Therefore the CLK pulse has no effect. The first input combination of the truth table shows this and under this case the output Q retains its last state.

When J is low, K is high, the upper AND gate is disabled while the lower AND gate is enabled. Hence, the flip-flop cannot be set ; instead it is reset, i.e. Q = 0. This is shown by the second entry in the truth table.

When J is high, K is low, the upper AND gate is enabled while the lower one is disabled. So the flip-flop is set, there by making Q = 1.

When J and K are both high, then the flip-flop is set or reset depending on the previous value of Q. This is implemented by the feedback connections from $\overline{Q}$ to the upper AND gate and Q to the lower AND gate, as shown in Figure 12.5(a). If Q is high previously, the lower AND gate sends a RESET trigger to the flip-flop on the next clock pulse. Then Q becomes equal to 0. On the other hand if Q is low previously, the upper AND gate sends a SET trigger on the flip-flop making Q = 1.

So, when J = K = 1, Q changes its value from 0 to 1 or 1 to 0 on the positive clock pulse. This changing of Q to $\overline{Q}$ is called toggle. Toggle means to change to the opposite state.

Any flip-flop may be driven by +ve as well as –ve clock. As such JK flip-flop can also be driven by positive clock as well as negative clock. Figure 12.6 shows symbol of positive clocked and negative clocked JK flip-flop.

Racing: Racing means multiple transition of states of a flip flop for the same input. It results because of the feedback connections in the JK flip flops. It can be avoided by making the flip flop edge triggered sensitive or using MS flip flop.

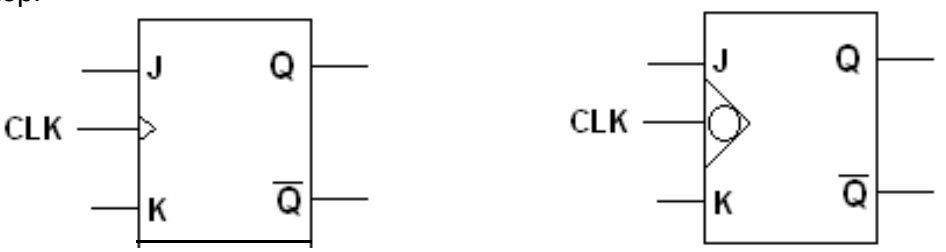
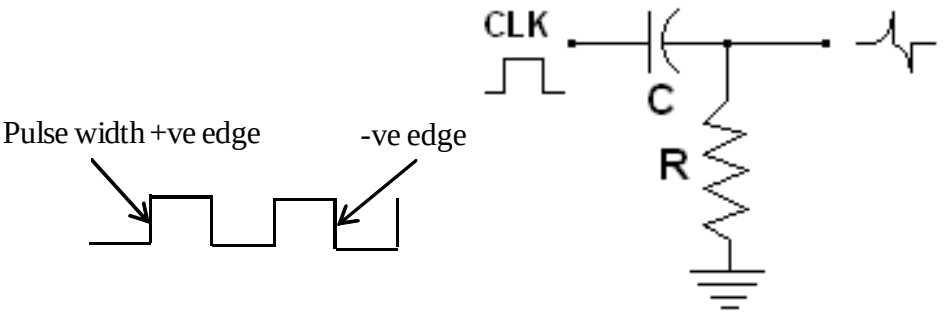


Figure 12.6 : Positive clocked JK Flip-Flop Negative clocked JK Flip-Flop

RC Differentiator Circuit : The clock pulse applied to a flip-flop is a square wave signal. Clock pulse is used to achieve synchronization of different flip-flops operation. The time span of the square wave is a longer time in comparison to the response time of flip-flops. Therefore, the time span of the clock pulse is reduced by converting its shape to a narrow spike, so that the flip-flops can change their states at an instant rather than transiting their states one after another in a lethargic manner. It makes the synchronization more precise. The square wave pulse is modified to a narrow spike by using a RC differentiator circuit as shown in Figure 12.7

Propagation : Propagation delay is the time required to convert the inputs of a circuit to outputs



(a) Clock pulse (b) RC Differentiator Circuit

Figure 12.7 : Clock pulse and Circuit

The upper tip of the differentiated pulse is called positive edge and the lower tip is called is negative edge. When a flip-flop is triggered by this type of narrow spike, it is called edge triggered flip-flop. If the flip-flop is driven by +ve edge, it is called +ve edge triggered flip-flop. If it is driven by negative edge, it is called negative edge triggered flip-flop.

Racing : In a flip-flop if the output toggles more than ones without changing input during a clock edge than it is called racing. All flip-flop have a propagation delay, means the output changes its state after a certain time period from applying the input and the clock pulse. So, when a flip-flop is edge triggered, then due to propagation delay the output cannot affect the input again, because by that time the edge of clock pulse has already passed away. If the propagation delay of a flip-flop is 20 ns and the width of the spike is less than 20 ns, then the returning Q and $\overline{Q}$ arrive too late to cause false triggering.

12.4.4 Master Slave Flip-Flop

MS flip-flop is another way to avoid racing. Figure 12.8 shows one way to build MS flip-flop using two JK flip-flop to one of which clock pulse is applied directly and to the other applied through an inverter. The first JK flip-flop is called master and the later is called slave. The master responds to its J and K inputs at the positive pulse and the slave responds to the negative clock pulse. When J=1, K=0, the master sets on the positive clock pulse. The high Q output of the master drive the J input of the slave.

So, at the negative clock pulse, the slave also sets, copying the action of the master.

When J=0, K=1, master resets at +ve clock pulse and the slave resets at the –ve clock pulse.

When J=K=1 master toggles at +ve clock pulse, and the slave toggles at the –ve clock pulse.

Hence whatever master does, the slave copies it.

MS flip-flop is a very popular flip-flop in industry due to its inherent resistance to racing. Hence it is extensively used to build counter.

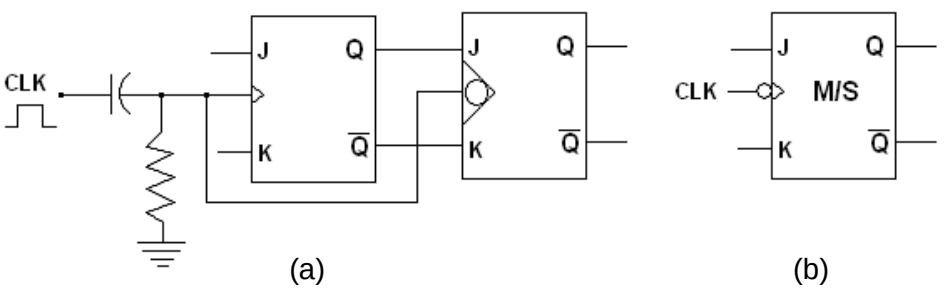


Figure 12.8 (a) Edge triggered JK MS Flip-Flop,
(b) Symbol of JK MS Flip-Flop



CHECK YOUR PROGRESS

Q.1. A flip-flop is basically a
(a) mono-stable multi-vibrator
(b) astable multi-vibrator
(c) bi-stable multi-vibrator (d) none of these

Q.2. JK flip-flop has the specialty in
(a) fast response time (b) toggle property
(c) spike shaped clock input (d) preset input

Q.3. In MS flip-flop the master changes state
(a) after the slave (b) with the slave at the same time
(c) before the slave (d) never

Q.4. In edge triggering the clock pulse used is
(a) Square wave (b) Triangular wave
(c) Narrow spike (d) None



12.5 LET US SUM UP

- Digital circuits are of two categories - combinational and sequential
- A sequential circuit's output depends on past output and present inputs.
- A flip-flop is basically a single cell of memory which can store either 1 or 0.
- Sequential circuits use flip-flop as their building block.
- There are many types of flip-flop viz RS, D, JK, MS flip-flop.



12.6 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



12.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (c)

Ans. to Q. No. 2 : (b)

Ans. to Q. No. 3 : (c)

Ans. to Q. No. 4 : (c)



12.8 MODEL QUESTIONS

Q.1. Distinguish between combinational circuit and sequential circuit.

Q.2. Define flip-flop. Draw an RS flip-flop using NOR gate and explain its operating principle with truth table.

Q.3. What modification is done in RS flip-flop to construct D flip-flop? What is achieved after the modification ?

Q.4. Explain the operation of MS flip-flop with its symbol and truth table.

Q.5. What is the advantage of JK flip-flop over an RS flip-flop ? Write the working principle of a JK flip-flop.

Q.6. Why a square wave clock pulse is converted to a narrow spike ? How it is done ?

Q.7. What is the drawback of JK flip-flop? How it is removed?

****

UNIT 13 : REGISTERS

UNIT STRUCTURE

13.1 Learning Objectives
13.2 Introduction
13.3 Registers
13.4 Serial In—Serial Out(SISO) Register
13.5 Serial In –Parallel Out (SIPO) Register
13.6 Parallel In –Serial Out(PISO) Register
13.7 Parallel In –Parallel Out(PIPO) Register
13.8 Let Us Sum Up
13.9 Further Reading
13.10 Answers to Check Your Progress
13.11 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define registers
- describe the working principle of registers
- describe different registers like Serial In - Serial Out, Serial In - Parallel Out, Parallel In - Serial Out, Parallel In - Parallel Out.

13.2 INTRODUCTION

In the previous units, we have learnt about sequential and combinational circuits. In this unit, we will learn about registers. Different types of registers like SISO, SIPO, PISO and PIPO are covered in detail in this unit.

13.3 REGISTERS

A register is a set of flip-flops used to store binary data. The data to be stored is entered or shifted into the register and also taken out or shifted out as per necessity. Hence registers with the facility to shift their contents

right or left or in serial or parallel are known as shift registers.

Registers are used to store data temporarily. Registers can be used to perform some important arithmetic operations like complementation, multiplication, division etc. It can be connected to form counters, to convert serial data to parallel and parallel to serial data.

Types of registers: According to shifting of binary number different types of registers are:

- Serial In—Serial Out(SISO)
- Serial In –Parallel Out (SIPO)
- Parallel In –Serial Out(PISO)
- Parallel In –Parallel Out(PIPO)

13.3 SERIAL IN – SERIAL OUT (SISO) REGISTER

Figure 13.1 shows a typical 4 bit Serial In – Serial Out register using flip-flops. Here the content of the register is named as QRST. Let us consider that all flip-flops are initially reset. Hence at the beginning QRST = 0000. Let us consider a binary number 1011 we want to store in the SISO register.

At time A : A 1 is applied at the J input and 0 at the K input of the first flip-flop. At the negative edge of the CLK pulse, this 1 is shifted into Q. The O of Q is shifted into R, O of R is shifted into S and O of S is shifted into T. The output of the register just after time A is QRST = 1000.

(During the input/output operation of the register, whatever we give to the J input, we must give complement of it to the K input.)

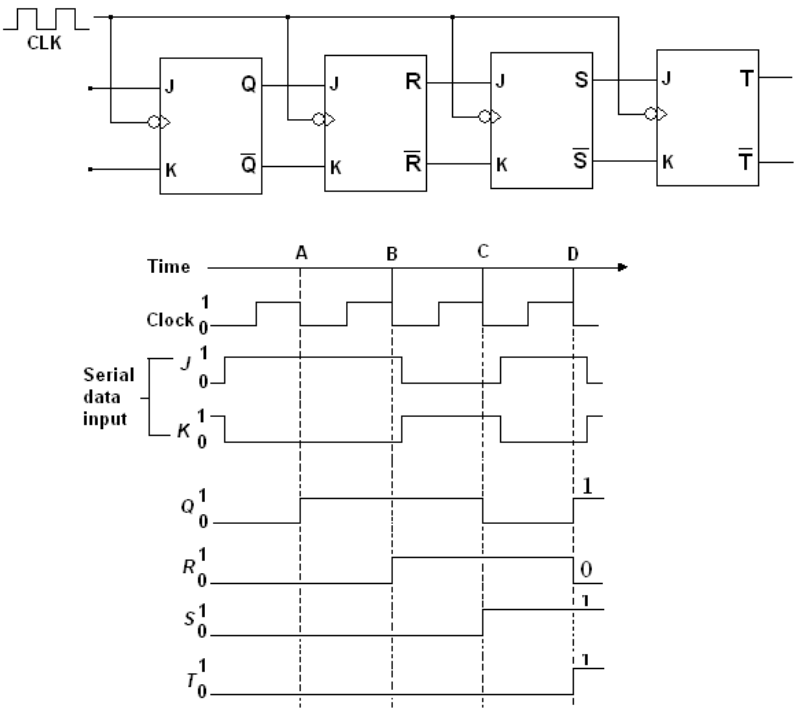


Figure 13.1 : 4 Bit Serial In – Serial Out Shift Register

At time B : Another 1 and 0 is applied in the J & K input of the first flip-flop. So at the negative CLK edge, the 1 is shifted to Q. The 1 of Q is shifted in R,0 of R is shifted in S, 0 of S is shifted into T. So, at the end of time B the output of all the flip-flops are QRST=1100.

At time C : A 0(zero) is applied in the J input and 1 at the K input of the 1st first flip-flop. At the negative CLK edge, the O shifts to Q. The 1 of Q shifts into R,1 of R shifts into S, O of S shifts into T. Hence the output becomes QRST=0110.

At time D : 1 is applied to J input and 0 at the K input of the first flip-flop. So after the negative clock pulse the 1 shifts into Q. The previous O of Q shifts into R, the 1 of R shifts into S, the 1 of S shifts into T. Hence at the end of time D, the registers contains QRST=1011.

In the above steps, using 4 CLK pulses, we have shifted a 4 bit binary number 1011 in the register in a serial fashion.

To take out this binary number serially, we need another 4 CLK pulses and four O inputs into J pin and four 1 into K pin of the first flip-flop. The binary number leave the register serially through the T pin of the last flip-flop.

13.5 SERIAL IN—PARALLEL OUT (SIPO) REGISTER

In this type of shifts register, data is entered serially into the register and once data entry is completed it can be taken out parallely. To take the data parallely, it is simply required, to connect the output of each flip-flop to an output pin. All other constructional features are same as in Serial In—Serial Out (SISO) register.

The shifting of data into SIPO is same as SISO registers. In the SIPO of Figure 13.2 , a binary number say, 1011 would be shifted just like the manner as described in the previous section. It would take 4 CLK pulses to complete the shifting. As soon as shifting is completed, the stored binary number becomes available in the output pins QRST. SIPO register is useful to convert serial data into parallel data.

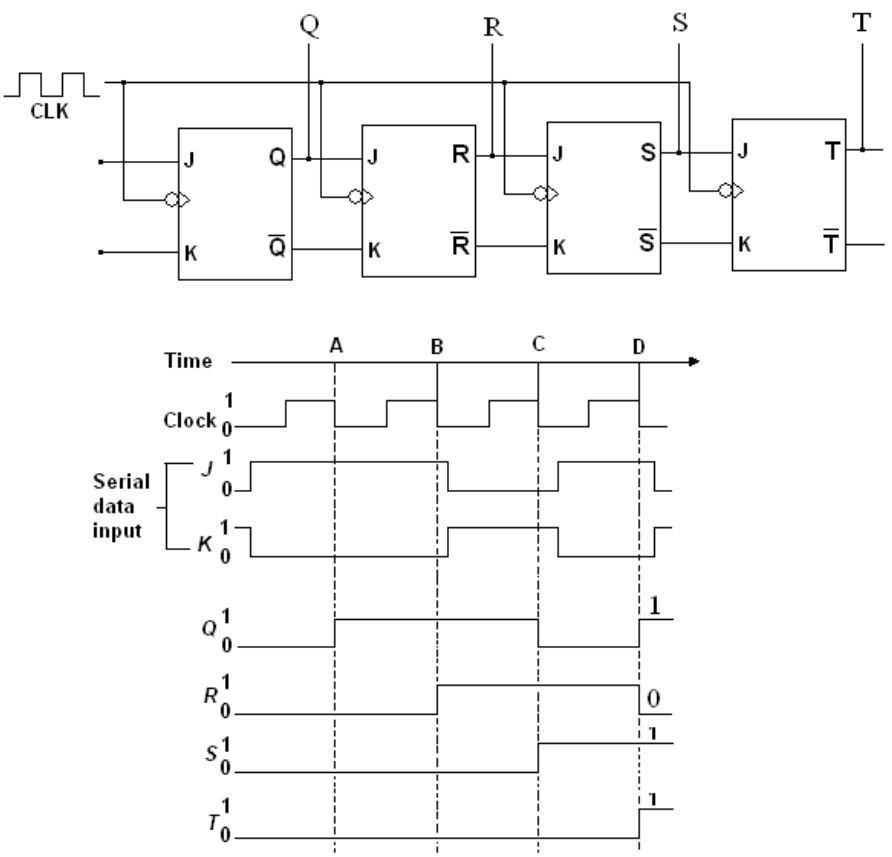


Figure 13.2 : 4 Bit Serial In – Parallel Out Shift Register

13.6 PARALLEL IN-SERIAL OUT (PISO) REGISTER

PISO registers take data parallely and shift data serially. Commercially available TTL IC for PISO is 54/74166. To understand the functional block diagram of 54/74166, we should first understand the Figure 13.3.

Figure 13.3 (a) is a clocked RS flip-flop, which is converted to D flip-flop by a NOT gate. The output of the flip-flop is 1 if Data IN ($\overline{X}$) is 0. Next we add a NOR gate as in Figure 13.3(b). Here, if X_2 is at ground level, X_1 will be inverted by the NOR gate. As for example, if =1, then output of the NOR gate will be =0, thereby a 1 will be clocked into the flip-flop.

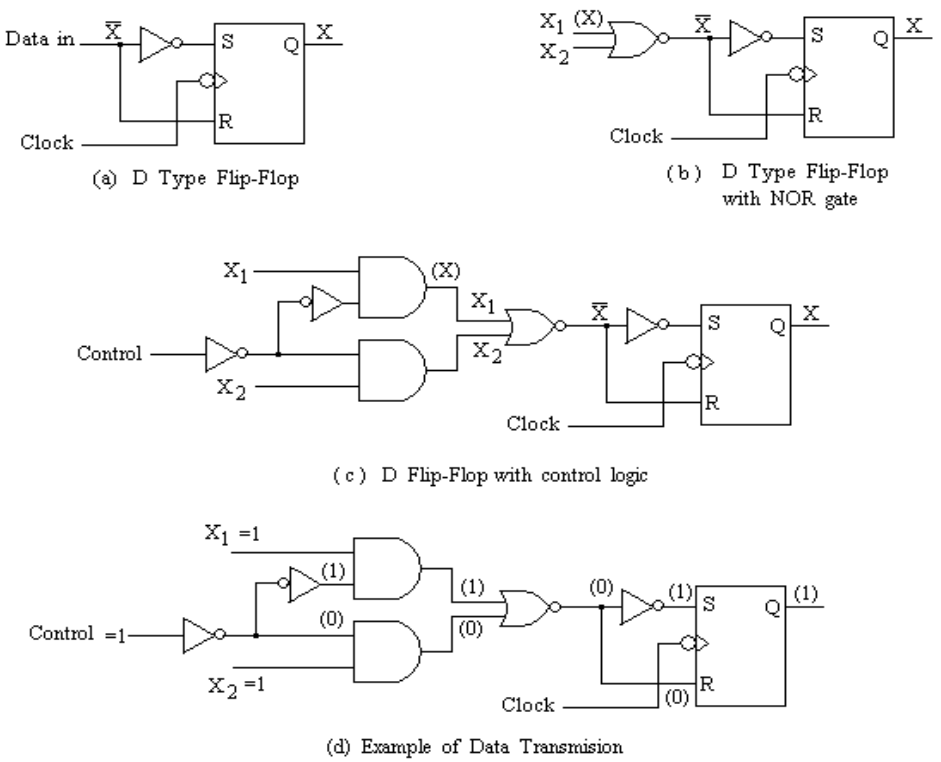
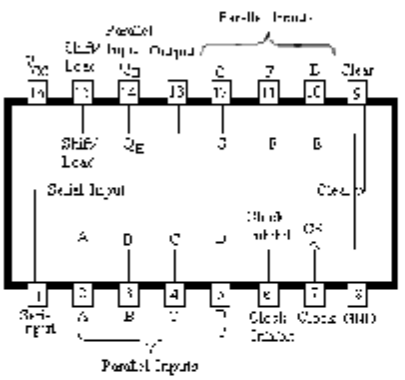


Figure 13.3 : Building Block of Parallel In- Serial Out Register

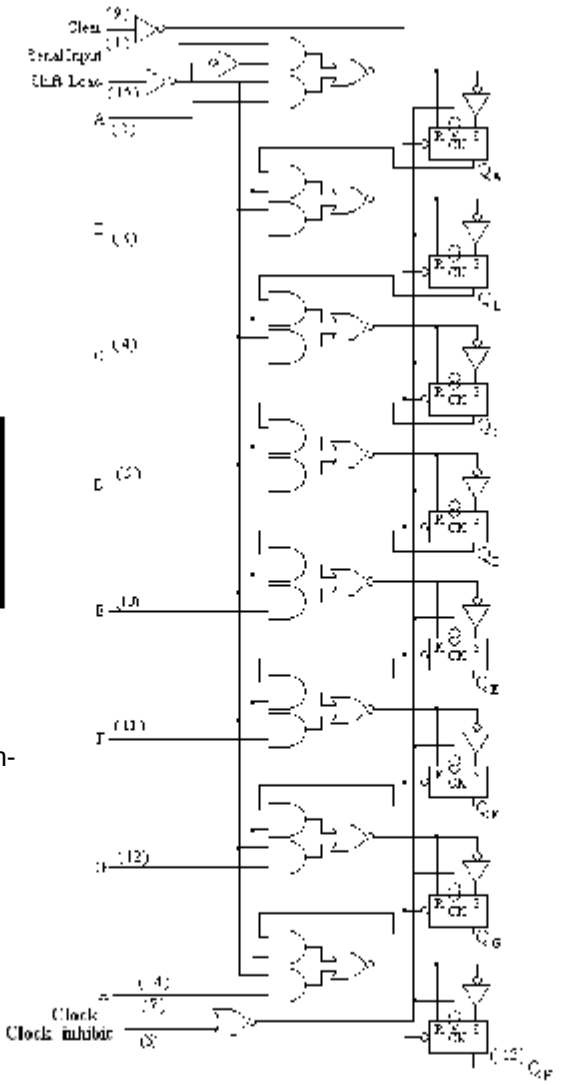
Now in Figure 13.3(c), two AND gates and two NOT gates are added. These will allow the selection of data . If the control line is high, the upper AND gate is enabled and the lower AND gate is disabled. Thus the data will enter at the upper leg of the NOR gate and at the same time the lower leg of the NOR gate is kept at ground. Opposite to this, if the control

is low, the upper AND gate is disabled and the lower AND gate is enabled.

If we study the Figure 13.4 of PISO, we see that circuit of Figure 13.3 (c) is repeated 8 times to form the 54/74166 shift register. These 8 circuits are connected in such a style that it allows two operations: (1) The parallel data entry and (2) shifting of data serially through the flip-flop from Q_A toward Q_B .



(a) Pin Out Diagram of Serial In-Parallel Out Register



(b) Logic Diagram of Serial In-Parallel Out Register

Figure 13.4 : Circuit of 54/74166

In Figure 13.4, the X_2 input of Figure 13.3(c) is taken out from each flip-flop to form 8 inputs named as ABCDEFGH to enter 8 bit data. parallely to the register. The control is named here as SHIFT/LOAD which is kept low to load 8 bit data into the flip-flops with a single clock pulse parallely. If the SHIFT/LOAD is kept high it will enable the upper AND gate for each flip-flop. If any input is given to this upper AND gate then a clock pulse will shift a data bit from one flip-flop to the next flip-flop. This means data will be shifted serially.

13.7 PARALLEL IN –PARALLEL OUT (PIPO) REGISTER

The register of Figure 13.4 can be converted to PIPO register simply by adding an output line from each flip-flop.

The 54/74198 is an 8 bits such PIPO and 54/7495A is a 4 bit PIPO register. Here, the basic circuit is same as Figure 13.3(c). The parallel data outputs are simply taken out from the Q sides of each flip-flop. In Figure 13.5 the internal structure of 54/7495A is shown.

When the MODE CONTROL line is high, the data bits ABCD will be loaded into the register parallely at the negative clock pulse. At the same time the output is available at $Q_A Q_B Q_C Q_D$. When the MODE CONTROL is low, then the left AND gate of the NOR gate is enabled. Under this situation, data can be entered to the register serially through SERIAL INPUT. In each negative transition, a data bit shifted serially from Q_A to Q_B, from Q_B to Q_C and so on. This operation is called right-shift operation.

With a little modification of the connection, the same circuit can be used for shift-left operation. To operate in shift-left mode, the input data is to be entered through D input pin. It is also necessary to connect Q_D to C, Q_C to B, Q_B to A. MODE CONTROL line is high to enter data through the D input pin and each stored data bits of flip-flops will be shifted to left flip-flop on each negative clock transition. This is serial data and left shift operation.

Two clock inputs—clock1 and clock2 is used here to perform shift right and shift left operation..

Hence 54/7495A can be used as Parallel In –Parallel Out shift register as well as shift right and shift left register.

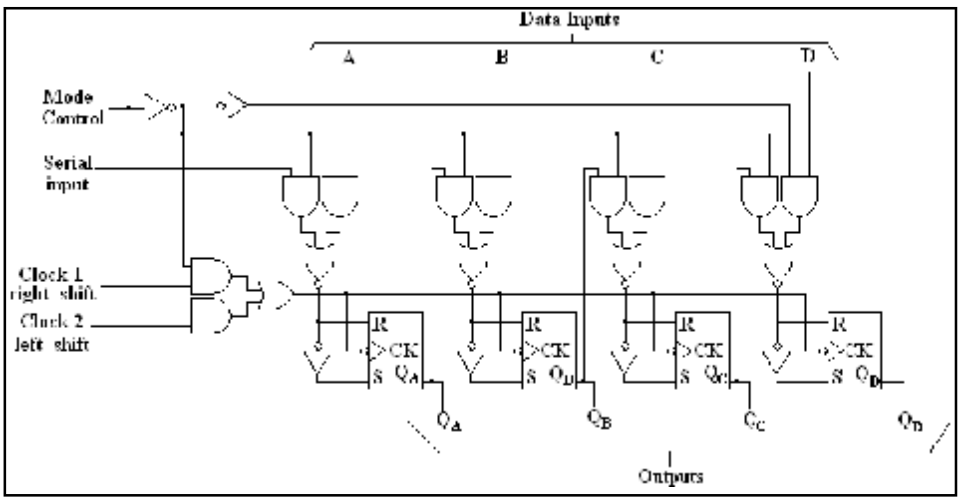


Figure 13.5 : Parallel In- Parallel Out Shift Register



CHECK YOUR PROGRESS

Q.1. Shift registers are.....

(a) basically a sequential circuit (b) a combinational circuit

(c) permanent memory (d) none of these

Q.2. In SIPO

(a) data enters parallely and leaves serially

(b) data enters serially and leaves serially

(c) data enters serially and leaves parallely

(d) data enters parallely and leaves parallely

13.8 LET US SUM UP

- A register is a set of flip flop used to store binary data.
- Shift registers are sequential circuit which are used to store binary bits.

- They are of four different types - Serial In- Serial Out, Serial In-Parallel Out, Parallel In- Parallel Out and Parallel In-Serial Out register.
- PISO registers take data parallely and shift data serially.
- In SIPO registers, data is entered serially and taken out parallely.



13.9 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar, P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



13.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (a)

Ans. to Q. No. 2 : (c)



13.11 MODEL QUESTIONS

Q 1. What is register? Explain different types of register.

Q.2. Draw logic diagram with output wave form of a 4-bit Serial In-Parallel Out shift register for an input of 1101. Explain its operation.

Q.3. What are Shift registers?

Q.4. Describe the working of SISO register.

Q.5. Compare SISO register with PISO registers.

Q.6. Compare PIPO register with PISO registers.

Q.7. Compare SISO register with SIPO registers.

****

UNIT 14 : COUNTERS

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Counters

14.4 Asynchronous Counter

14.5 Synchronous Counter

14.6 Let Us Sum Up

14.7 Further Reading

14.8 Answers to Check Your Progress

14.9 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define and describe the working of counters
- describe the different types of counters
- describe asynchronous counter
- describe synchronous counter.

14.2 INTRODUCTION

In the previous units we have learnt about different sequential and combinational circuits. We have also learnt about the different types of registers like SIPO, PIPO, SISO and PISO.

In this unit, you will be introduced to the concept of counters. Different types of counters like asynchronous and synchronous counters are discussed in this unit.

14.3 COUNTERS

A counter is a sequential circuit that goes through a prescribed sequence of states when input pulses, also called count pulses are applied. These pulses may be clocked pulses or they may originate from an external source. They may also occur at prescribed intervals of time or at random.

A counter may follow the sequence of binary count or any other sequence of states. Counters are used for counting the number of occurences of an event and are useful for generating timing sequences to control operations in a digital system.

A counter can also be used as a frequency divider circuit. In this application, a counter can reduce the frequency of a clock pulse.

There are basically two types of counter—Synchronous counter and asynchronous counter.

Counter are constructed by using flip-flops and other logic gates. If the flip-flops are connected serially then output of one flip-flop is applied as input to the next flip-flop. Therefore this type of counter have cumulative settings time due to propagation delay. Counter of this type is called serial or asynchronous counter. These counters have speed limitation.

Speed can be increased by using parallel or synchronous counter. Here, flip-flops are triggered by a clock at a time and thus setting time is equal to the propagation delay of a single flip-flop. But this type of synchronous counters require more hardware and hence these are costly.

Combination of serial and parallel counter is also done to get an optimize solution of speed and hardware cost. If each clock pulse advances the count of the counter by one, it is called **up counter**. If the count of the counter goes down at each clock pulse, it is called **down counter**.

Before operation, sometimes it is required to reset all the flip-flops to zero. It is called “Clear”. Sometimes, it is required to set the flip-flops before hand. It is called preset. To do this, two extra inputs are there in every flip-flop called CLR and PR.

14.4 ASYNCHRONOUS COUNTER

When the output of a flip-flop is used as the clock input for the next flip-flop it is called asynchronous counter.

Asynchronous counters are also called ripple counter because flip-flop transitions ripple through from one flip-flop to the next in sequence until all flip-flop reach a new state.

A binary ripple counter can be constructed by using clocked JK flip-flop. Figure 14.1(a) shows three MS JK flip-flops connected in series. The clock drives flip-flop A. The output of A drives B and the output of B drives C. J and K inputs of all the flip-flops are connected to positive to make them equal to 1. Under this condition each flip-flop will change state (toggle) with a negative transition at its clock point.

In the counter shown in Fig 14.1 (a), the flip-flop A changes its state at the negative edges of the clock pulses. Its output is applied to the B flip-flop as its clock input.

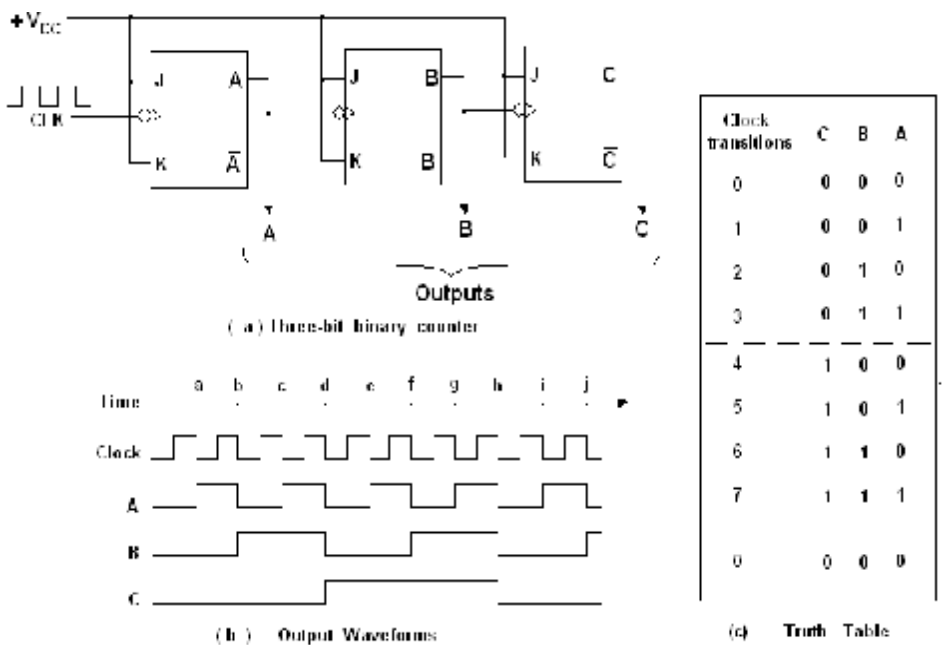


Figure 14.1 : Three Bit Binary Counter

The output of B flip-flop toggles at the negative edges of the output of A flip-flop. Similarly, output of B flip-flop is used as clock input to the C flip-flop and therefore C toggles at the negative edges of the output of B flip-flop. We can see that triggering pulses move through the flip-flops like a ripple in water.

The wave form of the ripple counter is shown in Figure 14.1 (b). It shows the action of the counter as the clock runs. To understand the wave form, let us assume that the counter is cleared before the operation. The A output is assumed the list significant bit(LSB) and C is the most-

significant-bit (MSB). Hence at very beginning the contents of the counter is CBA=000.

Flip-flop A changes its state to 1 after the negative clock pulse transition. Thus at point 'a' on the time line, A goes high. At point 'b' it goes low, at 'c' it goes back to high and so on.

Now output of A acts as clock input of B. So, each time the output of A goes low, flip-flop B will toggles. Thus at point 'b' on the time line, B goes high, at point 'd' it goes low, and toggles back high again at point 'f' and so on.

Since B acts as the clock input for C, each time the output of B goes low, the C flip-flop toggles. Thus C goes high at point 'd' on the time line, it goes back to low again at point 'h'.

We can see that the output wave form of A has half the frequency then the clock input wave. B has half the frequency than that of A and C has half the frequency than that of B.

We can further see that since the counter has 3 flip-flops cascaded together, it progresses through 000—— 001——010—011—100—101—110—111 as its CBA output. After CBA= 111, it starts the cycle again from CBA=000. One cycle from 000—111 takes 8 clock pulses, as it is evident from the wave form as well as from the truth table.

MOD of a counter : The number of input clock pulses that causes a counter to reset to its initial state is called the *modulus* of a counter. Therefore, the modulus of a counter means the total number of distinct states including zero that a counter can store.

For example, the three stage counter shown in figure 5.9 has modulus 8 which can be seen from the wave diagram and the truth table. The modulus is called MOD in short.

When the modulus of a counter is equal to 2^n, where n is the number of stages or flip-flops used, then the counter is called a natural counter. Hence, a three stage counter has MOD = 2^3 = 8, four stage counter has MOD = 2^4 =16 etc and therefore these are natural counters.

A counter can be forcefully reset to its initial state before it completes its natural count by applying appropriate feedback. These type of counters have modulus lesser than its natural count. As such we can have MOD 6, MOD 7, MOD 10 counters etc.

14.5 SYNCHRONOUS COUNTER

An asynchronous counter or ripple counter has limitation in its operating frequency. Each flip-flop has a delay time which is additive in asynchronous counter.

In synchronous counter the delay of asynchronous counter is overcome by the use of simultaneous applications of clock pulse to all the flip-flops. Hence in synchronous counter, the common clock pulse triggers all the flip-flops simultaneously and therefore the individual delay of flip-flop does not add together. This feature increase the speed of synchronous counter. The clock pulse applied can be counted by the output of the counter.

To build a synchronous counter, flip-flops and some additional logic gates are required. Figure 14.2 shows a three stage synchronous or parallel binary counter along with its output wave forms and truth table. Here, the J and K inputs of the first flip-flop is kept high and therefore the flip-flops toggle at the negative clock transition at its clock input. From figure we can see that the output of A is ANDed with count enable input to drive the 2^{nd} flip-flop and the outputs of A, B are ANDed with CLK to drive the third flip-flop. This logic configuration is often referred to as “steering logic” since the inputs J and K are steered to each individual flip-flop.

In the figure, the clock pulse is directly applied to all the flip-flops. J and K are both high for the 1st flip-flop, so the first flip-flop toggles state at the negative transition of the input clock pulses. This can be seen at points a,b,c,d,e,f,g,h,i on the time line.

The AND gate X is enabled when A is high, and it allows the inputs J and K=1 high to reach the 2^{nd} flip-flop. So the 2^{nd} flip-flop toggles with every other negative clock transition at points b, d, f and h on the time line.

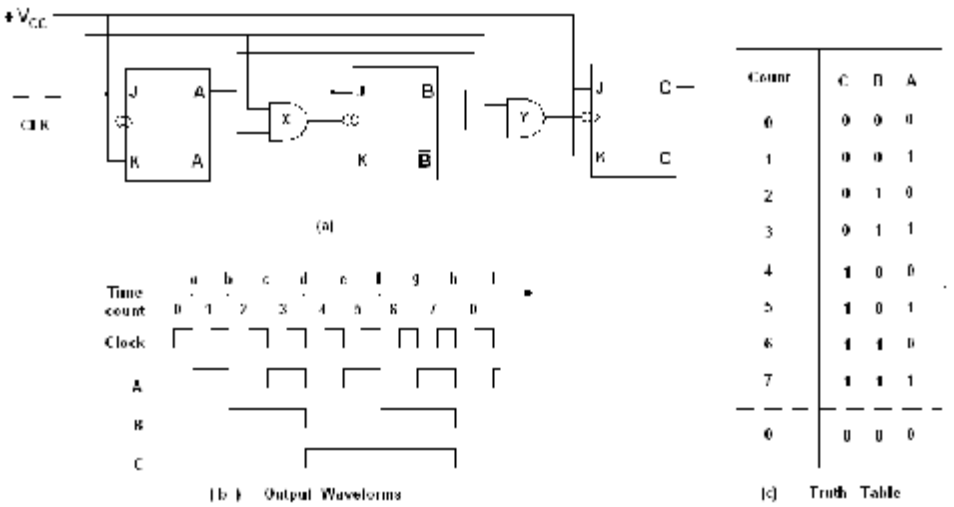


Figure 14.2 : 3 bit Parallel Binary Counter

The AND gate Y is enabled only when both A and B are high and it transmits the clock pulses to the clock input of the 3rd flip-flop. The 3rd flip-flop toggles state with every fourth negative clock transition at d and h on the time line.

The wave form and the truth table shows that the synchronous counter progresses upward in a natural binary sequence from 000 to 111. The total count from 000 to 111 is 8 and hence this counter can also be called MOD-8 counter, in count up mode.



CHECK YOUR PROGRESS

Q.1. State *true or false*

(a) Counters are non sequential digital circuits.

(b) Asynchronous counters are fast in operation than synchronous counters.

(c) The total number natural progression of a counter is called MODULUS.

(d) Counters can be used to build digital clock.

Q.2. Define Counters.



14.6 LET US SUM UP

- Modules of a counter is the number of clock pulses that cause a counter to reset its initial state.
- A counter is a sequential circuit that can count square waves given as clock input.
- There are two types of counters- asynchronous and synchronous counter.
- Asynchronous counters are also called ripple counters.
- Synchronous counters overcome the delay of asynchronous counters.



14.7 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



14.8 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (a) False, (b) False, (c) True, (d) True

Ans. to Q. No. 2 : A counter is a sequential circuit that goes through a perscribed sequence of states when input pulses are applied.



14.9 MODEL QUESTIONS

Q.1. What are the differences between asynchronous and synchronous counter? Draw a MODE-8 counter and explain its working principle.

Q.2. What do you mean by modulus of a counter? What is a forced counter?

Q.3. Describe asynchronous counters in detail.

Q.4. Describe synchronous counters in detail.

Q.5. What are ripple counters?

****

UNIT 15 : MEMORY ORGANISATION

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Random Access Memory

15.4 Types of RAM

- 15.4.1 Static RAM
- 15.4.2 DRAM

15.5 Organization of RAM

- 15.5.1 2D Organization
- 15.5.2 3D Organization

15.6 ROM

15.7 Types of ROM

15.8 Organization of Simple ROM Cell

15.9 Let Us Sum Up

15.10 Further Reading

15.11 Answer To Check Your Progress

15.12 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit you will be able to

- define random access memory
- define and describe the structure of RAM and ROM
- define their types and organization
- distinguish between 2D and 3D organization of RAM

15.2 INTRODUCTION

In all the other units of this block we discussed various circuits and different types of logic gates. In this unit we are going to discuss the internal organization of storage device of computer system where these circuits

are associated to transfer data and information time to time. This is the memory unit, which is a collection of storage cells and circuits. A computer system uses varieties of storage devices for its different operations and accordingly the main memory is divided into two main categories: primary and secondary memory. Here we will discuss mainly about the primary memory which is accessed directly by the processor. It is mainly based on integrated circuits.

We will also discuss various types of primary memory, their classification and use with their organizational structures.

15.3 RANDOM ACCESS MEMORY (RAM)

Each storage element of memory is directly or randomly accessible and can be examined and modified without affecting other cells and hence it is called random access memory. It can be both written into and read from. RAM gives a temporary place in computer to process electronic data. So RAM chip can store information only as long as computer has electronic power. It means that the data will be lost from RAM as and when the computer is shut off. So RAM is volatile in nature. The following figure 15.1 represents the block diagram of typical RAM.

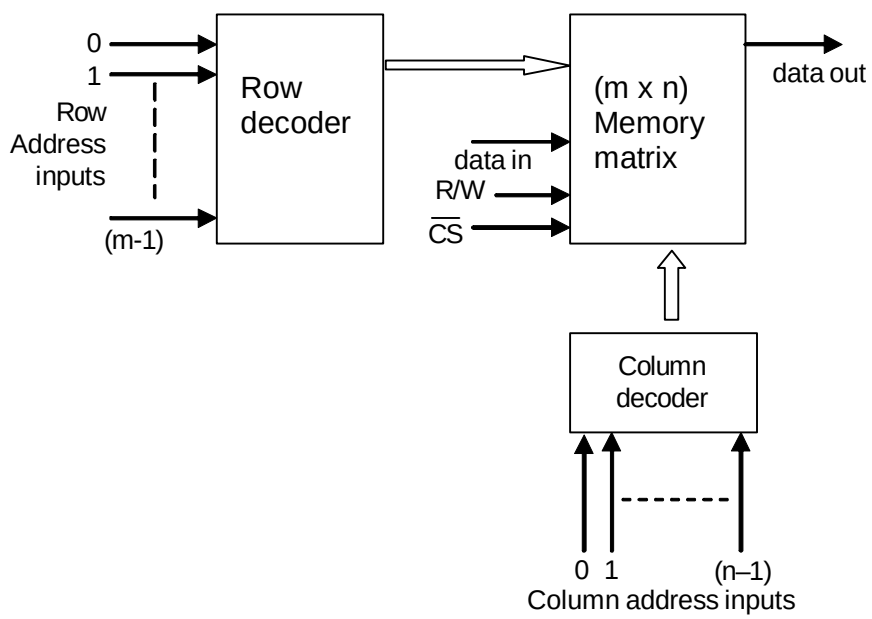


Figure 15.1 : Block diagram of a RAM chip

The mode of access of a memory system is determined by the type of components used. In a RAM, the word locations may be thought of as being separated in space, with each word occupying one particular location. The *access time* of a memory is the time required to select a word and either read or write it. In RAM, the access time is always the same regardless of the particular location of the word.

A semiconductor RAM is arranged in the form of a matrix of "m" words and each word is "n" bits wide. The intersections of rows and columns of such matrix are called *memory elements* or *cells*. Each cell has unique address representation. Figure 15.2 shows the matrix representation of a RAM.

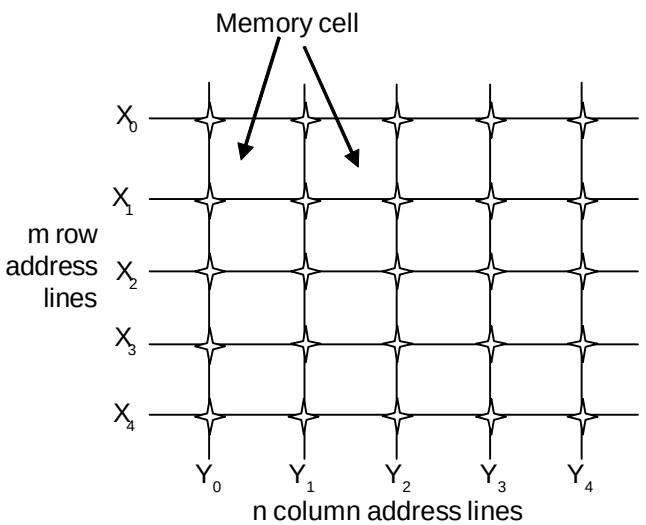


Figure 15.2 : RAM memory matrix representation

The address and data stored in a specific location of RAM are binary in form. Data are typically organized into words and each word has n bits. For k bit address there will be 2^k memory locations.

k-bit address à 2^k memory locations

The sii

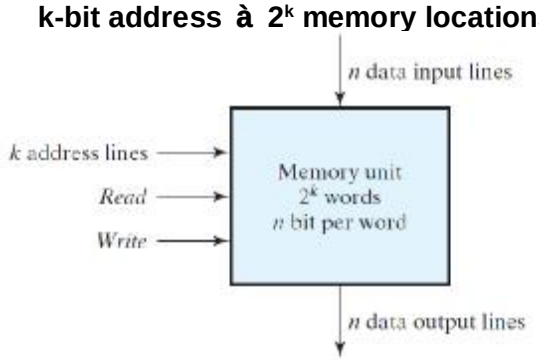


Figure 15.3 : Simplified RAM chip

15.4 TYPES OF RAM

Here, in this section we are going to discuss two important types of RAM: Static RAM and Dynamic RAM. They are categorized according to their operating mode.

15.4.1 Static RAM

Static RAM is also called SRAM. It is one type of semiconductor memory that consist of internal flip-flops which store binary information. The name static indicates that it does not need to be periodically refreshed. It is also volatile in nature as data is eventually lost when the memory is not powered.

Static RAM cell and its structure : Basically each bit in an SRAM is stored on four transistors that form two cross-coupled inverters. This storage cell has two stable states: 0 and 1. Two additional access transistors are used to control the read and write access operations. So, all total six transistors are required for a single SRAM memory cell. The write and read operations are performed by executing a sequence of actions that are controlled by the outside circuit.

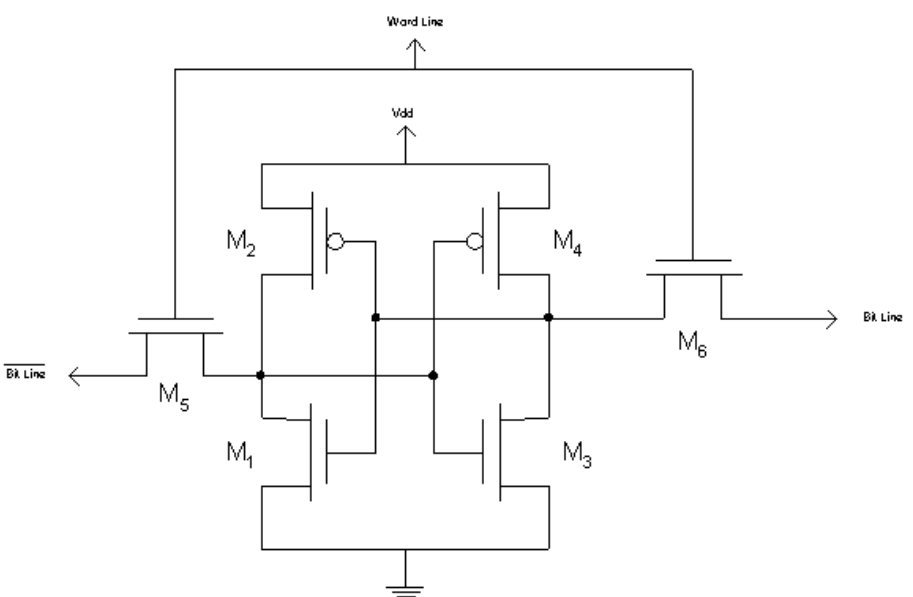


Figure 15.4 : 6-transistor SRAM Memory Cell

Cell is accessed through the word line. It controls two access transistors M_5 and M_6, to see whether the cell is connected to the bit lines. They are used to transfer data for both read and write operations. But it is not strictly necessary to have two bit lines.

A read operation is initiated by pre-charging both bit lines to logic 1. During read access inverters of SRAM are perform high and low operation. This improves SRAM bandwidth as compared to DRAMs. A write operation is performed by first charging the bit lines with values that are desired to be stored in the memory cell. Setting the word line high performs the actual write operation and the new data is latched into the circuit.

The size of an SRAM with *m* address lines and *n* data lines is 2^m words, or $2^m \times n$ bits.

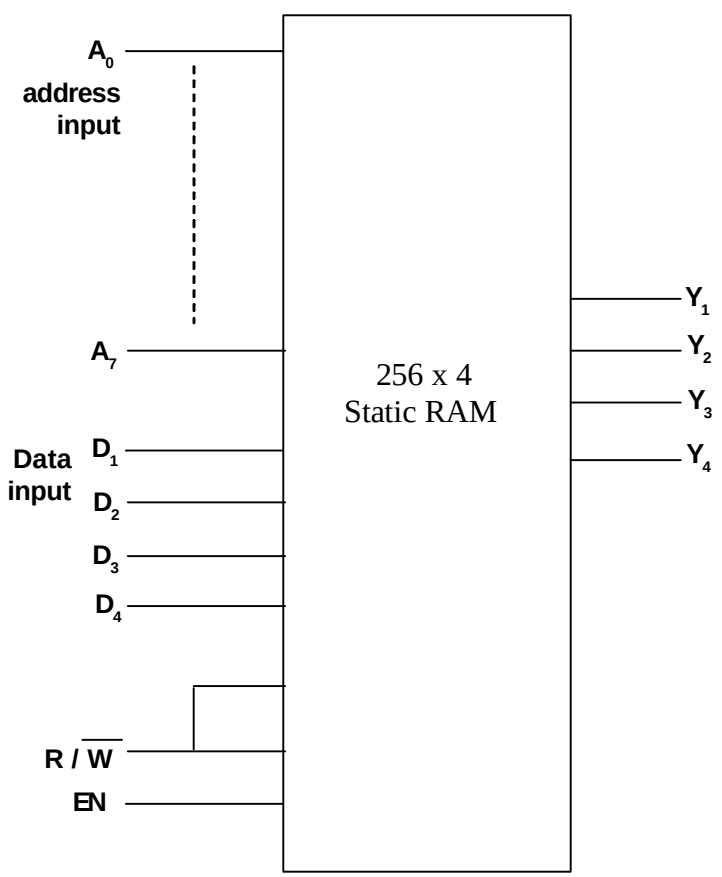


Figure 15.5 : Logic diagram for 256 x 4 static RAM

SRAM operation : An SRAM cell has three different states and they perform differently as follows:

a) Standby states : If the word line is not asserted, the access transistors M_5 and M_6 disconnect the cell from the bit lines. The two cross coupled inverters formed by $M_1 – M_4$ can continue to support each other as long as they are connected to the supply.

b) Reading states : First of all let us assume that content of memory is initially 1 and stored at Q. As we know that the read cycle starts with the pre charging bit line values as 1, so it assert the word line which enables both access transistors. In the next step, values of Q transferred to bit lines by leaving BL at its pre charges values and it discharge BL through M_1 to M_5 with the logic 0. On the BL side, the transistors M_4 and M_6 pull the bit line towards VDD with logic 1. The same will happen as opposite if the memory value is 0.

c) Writing states : The write cycle begins by applying the values to the bit lines. If we wish to write 0, we would have to apply 0 to the bit lines. A 1 is written by applying the values of the bit lines. Word Line is then asserted and the value that is to be stored is latched in.

Characteristics of SRAM :

a) More expensive, but faster and significantly less power is used than DRAM.

b) Easier to control and generally more truly random access.

c) Cannot be used in high-capacity, low-cost applications, such as the main memory in PCs.

d) SRAM are used in many industrial and scientific subsystems. It is embedded in practically all modern appliances, toys etc.

Types of SRAM :

a) Non-volatile SRAM : This is a special type of SRAM which can retain their data even when the power supply is lost. They are used in a wide range of situations where the preservation of data is critical like networking, aerospace and medical.

b) Asynchronous SRAM : This is fast access memory and is used in main memory. They are available from 4 Kb to 32 Mb.

c) By transistor type : As according to the transistor type we can divide SRAM again in two categories like:

- **Bipolar junction transistor** – This is used in TTL and ECL. They are very fast but consume a lot of power.
- **MOSFET** – This is used in CMOS. They have low power and very common today.

d) By Function: By function, SRAM can be divided into two categories :

- **Asynchronous**–These are independent of clock frequency. Here, data in and data out are controlled by address transition.
- **Synchronous –** Here, all timings are initiated by the clock edges. Address, data in and other control signals are associated with the clock signals.

e) By Feature : By feature we can divide SRAM into few separate categories like Zero Bus Turnaround (ZBT), syncBurst SRAM, DDR SRAM etc.

Usage of SRAM : Other parts of the computer, such as cache memories and data buffers in hard disks, normally use static RAM (SRAM).

15.4.2 Dynamic RAM

Dynamic RAM (DRAM) is the most common memory for personal computers and workstations. The main memory in PCs is DRAM. They store each bit of data in a separate capacitor within an integrated circuit. The capacitor can be either charged or discharged. These two states are represented by two values of a bit 0 and 1. Since capacitor leak charge, the information eventually fades unless the capacitor charge is refreshed periodically. Because of this refresh requirement, it is a dynamic memory.

The main advantage of DRAM is its structural simplicity. In DRAM only one transistor and a capacitor are required per bit, compared

to six transistors in SRAM. They are extremely small. So, hundreds of billions can fit on a single memory chip. This allows DRAM to reach very high densities. It is volatile in nature, since it loses its data quickly when power is removed.

The first DRAM with multiplexed row and column address lines was the Mostek MK4096 (4096x1) designed by Robert Proebsting and introduced in 1973. DRAM is usually arranged in a square array of one capacitor and transistor per data bit storage cell. In the structure, there are two level decoding. They are:

- Row access – goes into decoder to select row
- Column access – controls the MUX.

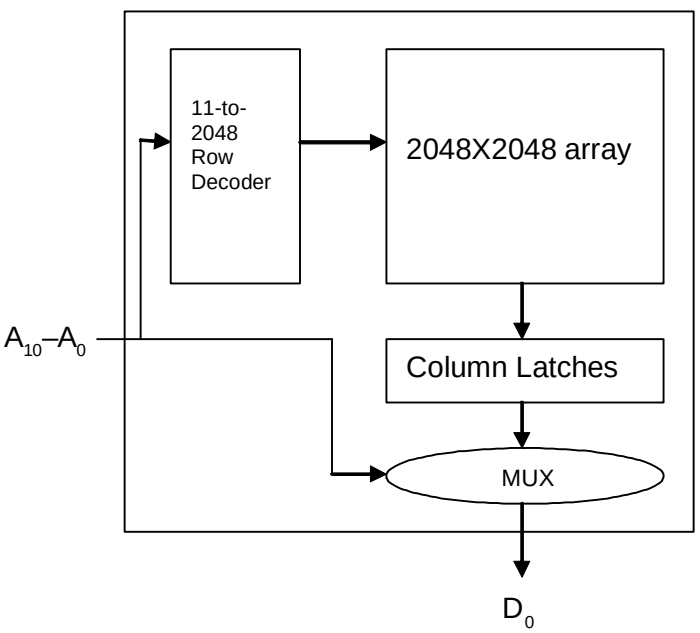


Figure 15.6 : Basic DRAM structure

DRAM memory cell : A typical DRAM cell consists of a single MOSFET and a capacitor. An array of such cells is used in a DRAM cell. Here, the transistor acts as a switch. For READ and WRITE operations also it uses MOSFETs.

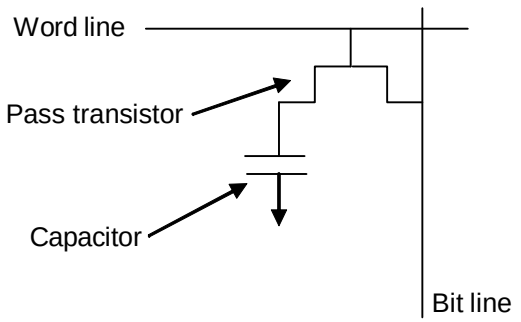


Figure 15.7 : A dynamic MOS RAM cell

The steps of reading and writing in a memory cell are :

Writing to DRAM cell

Steps

– Assert word line (turn the pass transistor “on”)

– Write value to bit line

- If the value is 1, the capacitor will be charged
- If the value is 0, the capacitor will be discharged

Reading DRAM cell

Steps

– Bit line charged 1/2 between low and high voltage

– Assert word line

– Value in capacitor is “read out” onto the bit line

– Bit line swings slightly to low or high

– Sense amp detects swing and indicates a 0 or 1

– B/C charge used in detection, rewrite after read

Features of DRAM

– Very dense

– Slow compared to SRAM (5-10X slower)

Usage : For economic reasons, the large (main) memories found in personal computers, workstations, and non-handheld game-consoles (such as PlayStation and Xbox) normally consist of dynamic RAM.

Comparisons between static and dynamic RAM

a) No refreshing in static RAM as it uses bistable latching circuitry to store each bit, but in DRAM, it is refreshed periodically.

b) 6 to 8 MOS transistors are required to form one memory cell in SRAM, whereas in DRAM only 3 to 4 transistors are used to form one memory cell.

c) Information stored in SRAM as voltage level in a flip flop, whereas in DRAM, information is stored as a charge in the gate to substrate capacitance.

d) SRAM is more reliable, but four times expensive than DRAM

e) An SRAM cell takes a lot more space on a chip than a DRAM.

f) SRAM is used to create the CPU's speed sensitive cache, while DRAM forms the larger system RAM space.

The difference we can get from their functional behavior is shown below:

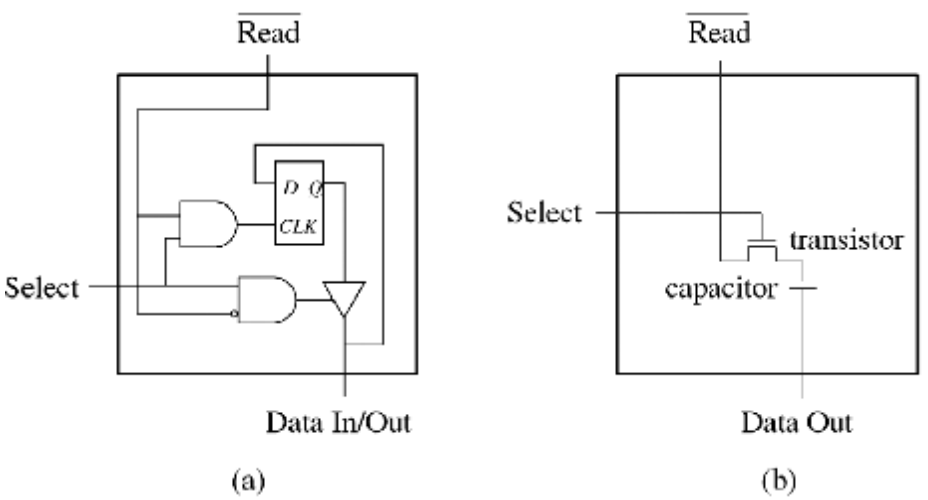


Figure 15.8: (a) Static RAM cell and (b) dynamic RAM cell



CHECK YOUR PROGRESS

Q.1. Give classification of RAMs.

.....

.....

Q.2. Differentiate between SRAM and DRAM.

.....

.....

Q.3. Fill in the blanks:

a) RAM is _____ memory device.

b) In personal computer _____ memory is used.

c) In DRAM, if the value is 1, the capacitor is _____.

d) A typical DRAM cell consists of a single _____ and a capacitor.

e) In SRAM _____ transistors are used.

f) _____ are more expensive

g) For n bit address there will be _____ memory locations.

h) The most common and low powered RAM is _____ used in CMOS.

i) _____ is used to create the CPU's speed sensitive cache.

j) The first DRAM with multiplexed row and column address lines was introduced in _____.

15.5 ORGANIZATION OF RAM

A RAM memory cell is organized in a number of different ways. A (m x n) memory has m-number of words, each word having n number of bits. In RAM, each memory location has a unique address. Depending on access mechanism, RAM is organized into different dimensions most commonly like 2D, 3D, 2.5D etc.

15.5.1 2D Organization

In this organization the memory on a chip is considered to be a list of words in which any word can be accessed randomly e.g., the memory of PCs used to have 16 bit words and normally in a chip it have 64KB memory = 32K words. The cells are organized as a 2-dimensional array with rows and columns. Each row refers to a word line. For n-bits per word, n-number of cells is interconnected

to a word line. Each column represents a bit line. The simplified representation of this organization is:

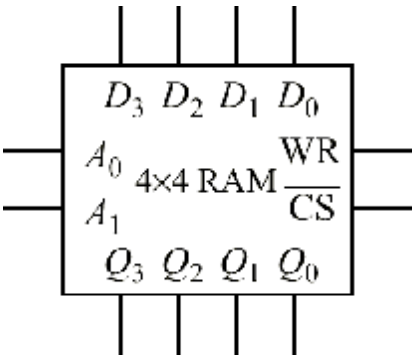


Figure 15.9 : Simplified 2D organization with 4X4 RAM chip

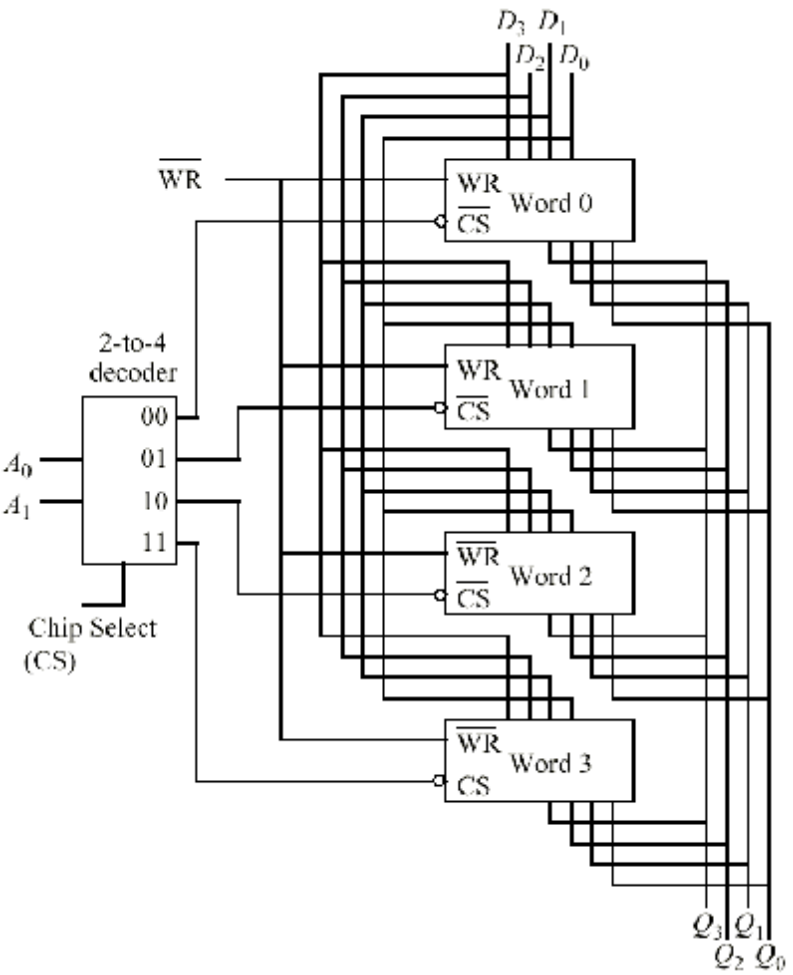


Figure 15.10 : A 4-word memory with 4-bit per word 2D organization

In this array organization bit line is connected to the input and sense (or output) terminal of each cell in its respective column. Each decoded line of decoder drives a word line. A complete word

can be input or output from the memory simultaneously. In write operation, the address decoder selects the required word and the bit lines are activated for a value 0 or 1 according to the data line values. Thus, it completes the write operation. On read operation, the value of each bit line is passed through a sense amplifier and passed on the data lines. Thus it completes the read operation. The word line identifies the word that has been selected for reading or writing. Usually ROMs and read mostly memories use 2D chip organization.

Example 15.1 : Design a 4-word by 8-bit RAM chip by connecting two 4 X 4 RAM chips.

Solution : The 4 x 8 RAM chip by using two 4 x 4 RAM chip is as follows :

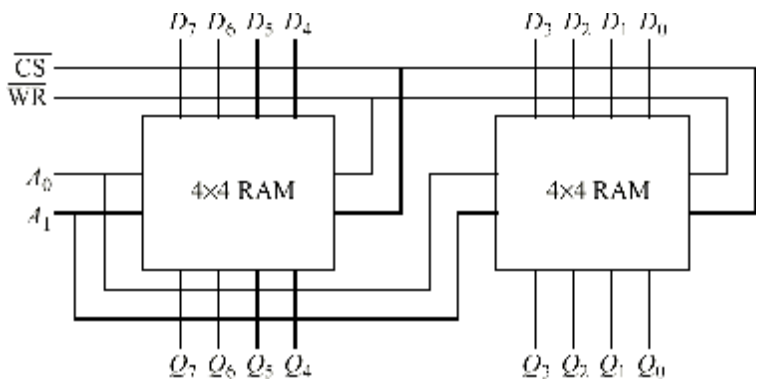


Figure 15.11 : A 4 X 8 RAM chip

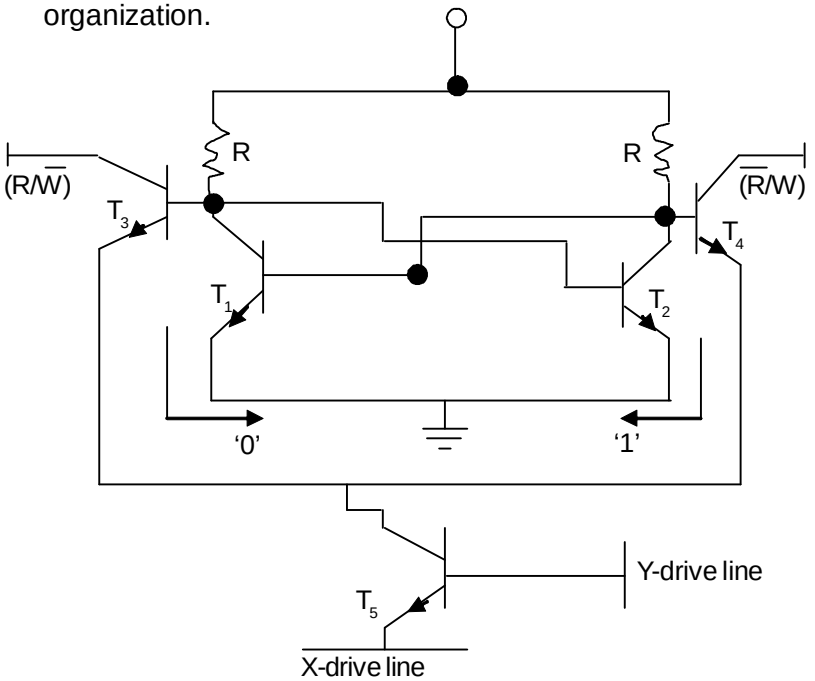
15.5.2 3 D Organization

In this organization of memory, each cell needs three terminals:

- X drive lines for rows
- Y drive lines for columns
- A bit/sense pair

In comparison to 2D organization, 3D is more economical. The content of memory address register (MAR) has two parts having number of bits. Suppose they are x and y. For a n-bit word there are n-bit planes. For each bit plane, a sense amplifier is attached. Any cell in a bit plane is selected through X and Y drive lines. This

organization makes the circuit more complicated and also gives a low switching speed. The figure 15.12 is the 3 terminal 3D organization.



Figgure 15.12 : 3 terminal semiconductor memory cell



CHECK YOUR PROGRESS

Q.4. Write TRUE or FALSE

a) In a RAM each memory location has a unique address.

b) In 2D organization of RAM, word cannot be accessed randomly.

c) In 2D organization, each column represents a bit line.

d) 2D organization is more economical than 3D organization of RAM.

e) Each row refers to a word line in 2D organization.

15.6 ROM

Read only memory (ROM) is a class of storage device used in computer as well as in other electronic devices. Data stored in ROM cannot be modified, or can be modified only slowly or with difficulty. So, it is used mainly in distributed system. Its internal organization is similar to SRAM. It is used in computer systems for initialization because it does not lose storage value when power is removed. In other sense, we can say that ROM is used to *boot* the operating system of the computer. It is programmed for particular purpose during the manufacturing time and user cannot alter its function.

ROM is a combinational logic circuit. Multiple single-bit functions embedded in a single ROM. It uses decoders to minimize the number of address lines. ROMs are effective at implementing truth tables. Any logic function can be implemented using ROMs. So, ROM can be used to implement complex combinational circuits within one IC package or as permanent storage package for binary information given by the designer.

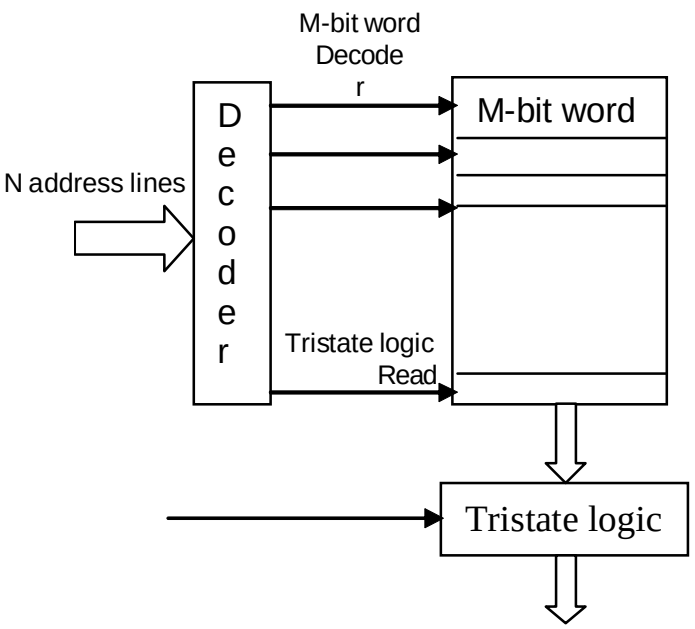


Figure 15.13 : Block diagram of ROM

ROM: A semiconductor memory where can store information permanently.

Here, N is the address lines and M is the output lines. Each bit combination of the address variable is called an address and each bit combination that comes out from the output lines is a data word. An output

word can be selected by a unique address. Since, there are N input bits, ROM will form 2^N words by M outputs. So, word available on the output lines at any instant depends on the address value applied to the input lines. For example, a ROM with 8 words by 5 bits is:

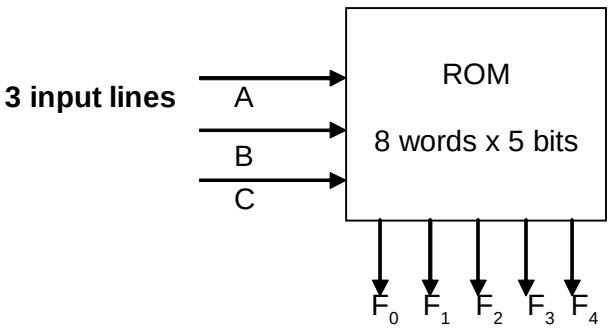


Figure 15.14 : 8 words X 5 bit ROM cell

Since, here there are 3 input lines, the words formed by this ROM are $2^3 = 8$ words.

ROM access time : ROM access time is the time from the application of a valid address code on the address inputs until the appearance of valid output data. It can also be measured from the activation of ***chip select*** to the occurrence of valid output data when a valid address is already on the inputs. Following figure 15.15 shows the ROM access time.

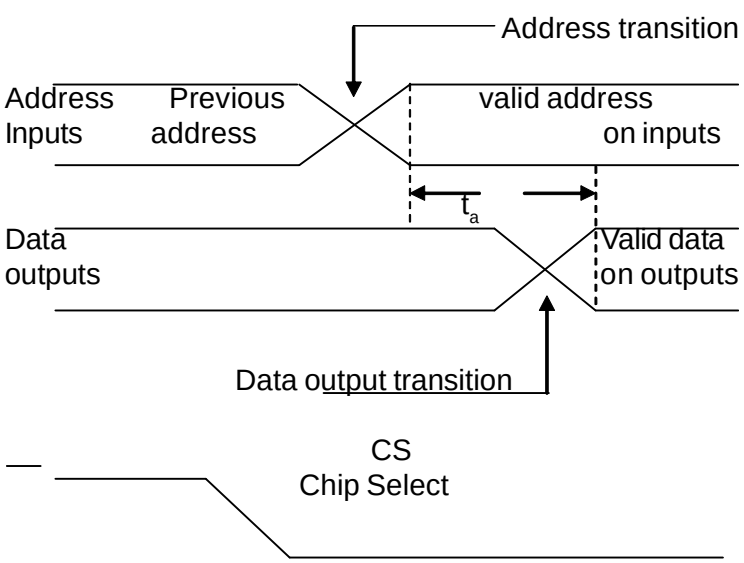


Figure 15.15 : ROM Access Time

Example 15.2 : Draw a 8 X 4 ROM

Solution : Since has to draw 8-word ROM, and 2^3=8, so there will be 3 input lines.

Say, N = 3 input lines,

M = 4 output lines

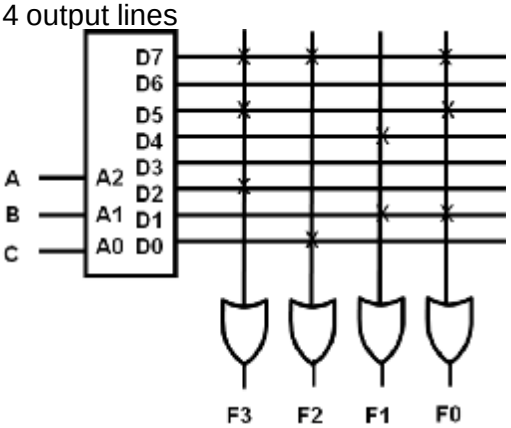


Figure 15.16 : A 8 X 4 ROM

In the above ROM chip, the fixed "AND" array is a "decoder" with 3-inputs and 8-outputs that implement minterms. The programmable "OR" array uses a single line to represent all inputs to an OR gate. An "X" in the array corresponds to attaching the minterm to the OR. In read mode for input (A_2,A_1,A_0) = 011, the output is (F_3,F_2,F_1,F_0) = 0011.

15.7 TYPES OF ROM

ROMs are classified according to how information are written or programmed into the memory storage locations. The classification of ROM is given is figure 15.17:

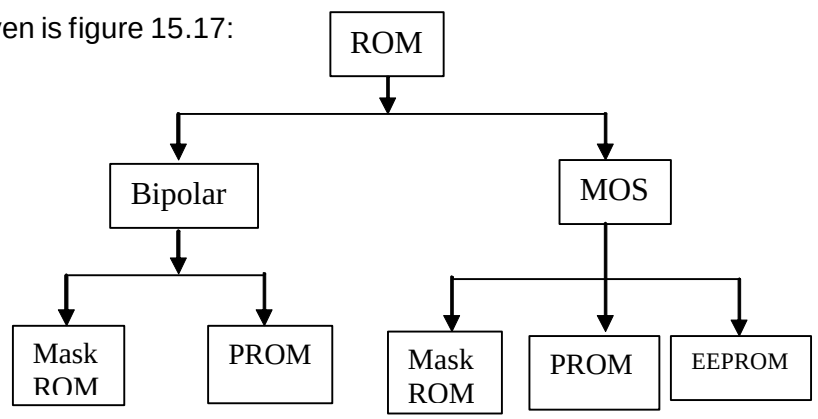


Figure 15.17 : The ROM family

Bipolar ROM uses bipolar transistors within an integrated circuit. They can be programmed. On the other hand, MOS ROM is constructed using MOSFETs. In a MOS cell, the silicon gate is completely insulated from the n-type substrate. The MOS can also be programmed. ROMs are programmed into two different ways: first is called *mask programming* and is done by manufacturer during the last fabrication process of the unit; the second type is *programmable read only memory* (PROM).

Mask ROM : It is programmed at the time of manufacturing according to customer's requirements. Once the memory is programmed, it cannot be changed. Most IC ROMs utilize the presence or absence of a transistor connection at ROW or COLUMN junction to represent 1 or 0. The 1s and 0s are obtained by providing a mask in the last fabrication step. A photographic negative called a mask is used to control the electrical interconnections on the chip.

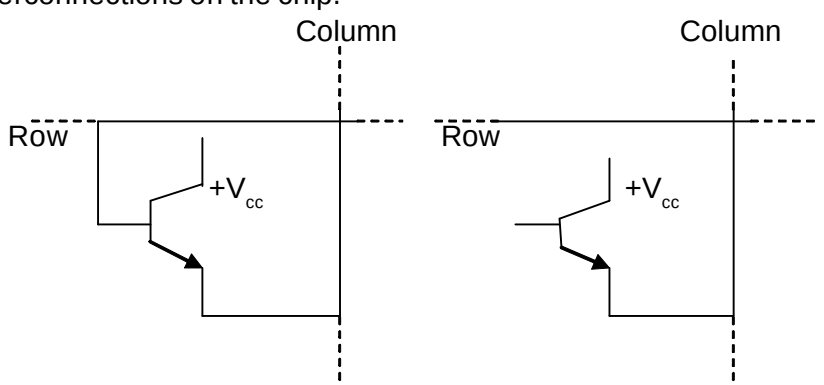


Figure 15.18 : Bipolar ROM cell

In bipolar ROM cell, when row line is taken high, all transistors with base connection to the Row line turn ON and connect the HIGH to the associated column lines. So, the presence of a connection from a Row line to the base of a transistor represents a '1' at that location. When there are no base connections at row or column junctions, the column lines remain LOW and are represented as '0'.

In MOS ROM cell the presence or absence of a gate of MOSFET connection at junction stores 1 or 0 permanently. Manufacturer makes the corresponding musk for the path of bipolar or for MOS ROM cell to produce the

1s or 0s according to specified customer's truth table. This process is called ***custom or mask programming***. Mask programmed ROM are economical only if large quantities of the same ROM configuration are to be manufactured.

The figure 15.19 illustrates the function of MOS ROM cell.

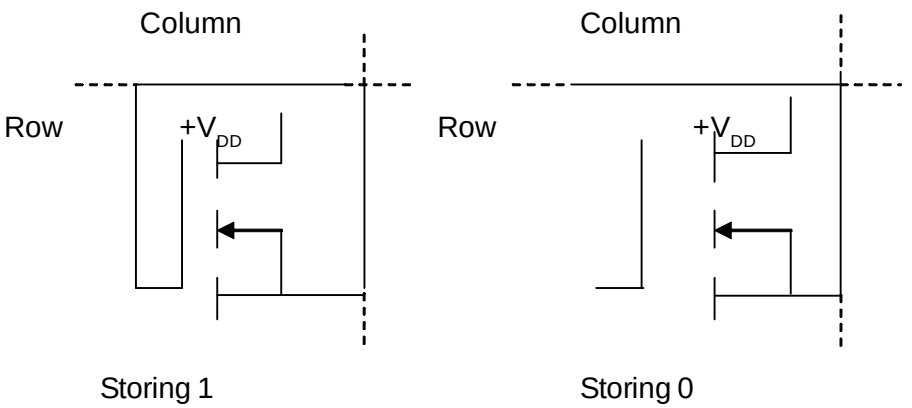


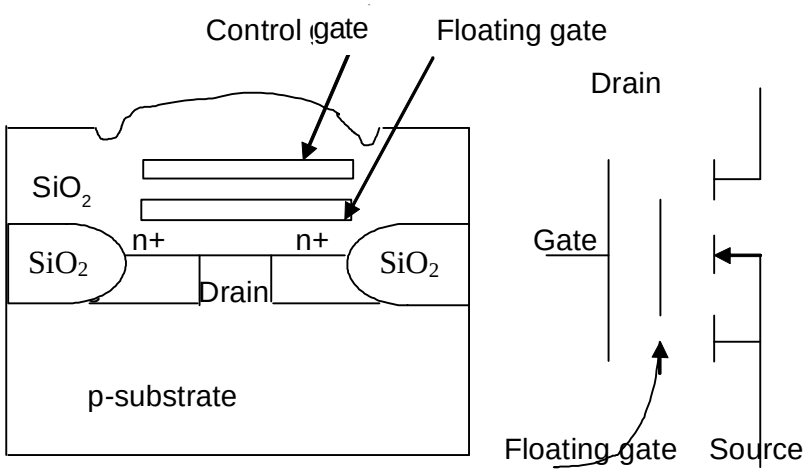
Figure 15.19 : MOS ROM cell

A major disadvantage of this type is the fact that it cannot be reprogrammed in the event of the design change or cannot do any modification of the stored program. Vendors are trying to overcome this problem by developing several program on this connection.

Programmable Read-only Memory (PROM) : In order to provide flexibility in some ROM applications, programmable ROM has been introduced. Blank PROM chips can be brought inexpensively and coded by anyone with a special tool called a **programmer**. But it cannot be reprogrammed. It is manufactured as a generalized integrated circuit with all matrix intersections and every intersection has a **fuse** connecting them. The higher voltage breaks the connection between the column and row by **burning** out the fuse and can thereby program the PROM according to the required truth table. This process is known as **burning the PROM**. Once a PROM is programmed, it cannot be changed and therefore it has to be done carefully and correctly in the first time itself. Hence, the fusing process is irreversible. PROMs are widely used in the control of electrical equipment such as washing machines and electric oven.

PROMs are available in both bipolar and MOS technologies. They have 4-bit or 8-bit output word formats with capacities ranging in excess of 2,50,000 bits.

Erasable Programmable ROM (EPROM) : A PROM device which can be erased and reprogrammed is called EPROM device. EPROM chips can be rewritten many times. Erasing an EPROM requires a special tool that emits a certain frequency of ultraviolet (UV) light. Once again, we have a grid of columns and rows. In an EPROM, the cell at each intersection has two transistors. One of the transistors is known as the **floating gate** and the other as the **control gate**. To erase it, you must supply a level of energy strong enough to break through the negative electrons blocking the floating gate. In a standard EPROM, this is best accomplished with UV light at a frequency of 253.7. The EPROM must be very close to the eraser's light source, within an inch or two, to work properly.



(a) Structure (b) Symbol

Figure 15.20 : EPROM cell

Here an additional floating gate is formed within the silicon dioxide (SiO_2) layer. The floating gate is left encountered while the normal control gate is connected to the row decoder output of EPROM. The data bits are represented by the presence or absence of a stored charge. The initial values of un-programmed EPROM cells may be all 0s or all 1s.

Disadvantages :

a) Changes in the selected memory locations cannot be made in the reprogramming. The entire memory should be erased before reprogramming.

b) The process of reprogramming cannot take place with the IC in the circuit. The EPROM IC must be removed from the circuit

and the stored program can be erased by exposing the memory cells to ultraviolet light through a 'window' on the IC package. This process takes about half an hour.

Electrically Erasable PROM (EEPROM) : It is also known as Electrically Alterable PROM (EAPROM). It removes the biggest drawbacks of EPROMs. The entire chip does not have to be completely erased to change a specific portion of it. It can be erased and programmed by the application of controlled electric pulses to the IC in the circuit, and thereby changes can be made in the selected memory locations without disturbing the correct data in other memory locations. Changing the contents does not require additional dedicated equipment. They are non-volatile like EPROM but do not require ultraviolet light to erase.

EEPROM is a rugged, low power semiconductor device and it occupies less space. It has the advantage of program flexibility, small size and semiconductor memory ruggedness. With EEPROM, the programs can be altered remotely, possibly by telephone. EEPROMs are changed 1 byte at a time.

Flash memory : These memories are basically EEPROMs except that erasure occurs at the block level in order to speed up the write process. They work much faster than traditional EEPROMs because it writes data in chunks, usually 512 bytes in size, instead of 1 byte at a time. One of the most common uses of flash memory is for the **basic input/output system** of computer. Advantages of flash memory are that they are non-volatile in nature and densities are higher than both SRAM and DRAM and the disadvantages are that they are slow and data cell must be erased before writing data to it.

15.8 ORGANIZATION OF SIMPLE ROM CELL

A ROM is sometimes specified by the total number of bits it contains, which is 2^n x m. We already specified that a ROM can be organized by its address and data bits. Figure 15.21 shown a 16 x 8 bit simplified ROM array.

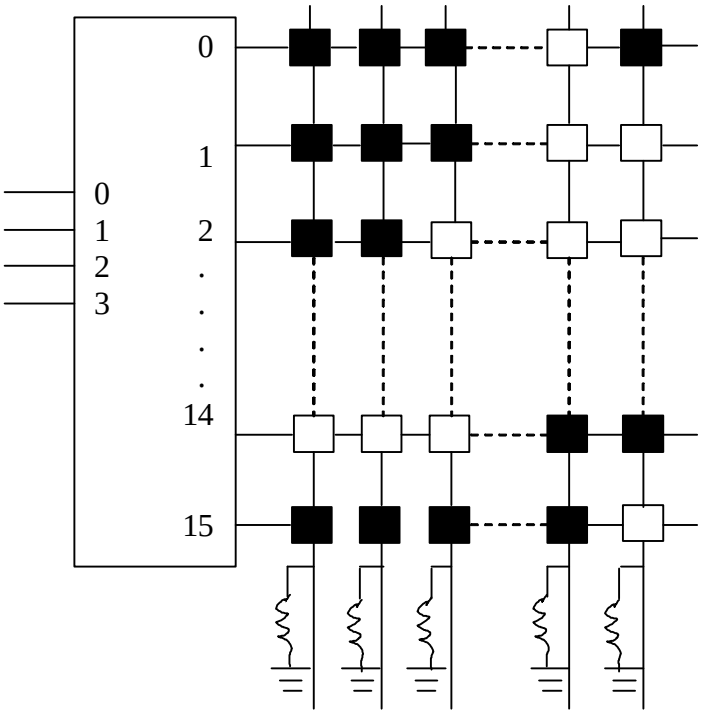
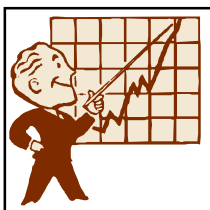


Figure 15.21 : A 16 x 8-bit ROM array

Here ROM is organized into 16 addresses, each of which stores 8 data bits. The total capacity of this ROM is 128 bits. The dark squares represents that ROM stored 1s by base connected transistor or gate-connected MOSFET and light squares represent 0s. When a 4 bit binary address is applied to the address inputs, the corresponding row line becomes HIGH. This HIGH is connected to the column line through transistors at each junction where 1 is stored. The column line stays LOW at each cell where 0 is stored because of the terminating resistor. The column lines form the data output. Thus, the eight data bits stored in the selected ROW appear on the output lines.



CHECK YOUR PROGRESS

Q.5. Fill in the blanks

a) ROM is used to _____ the operating system of the computer.

b) ROM is used mainly to _____.

c) The internal organization of ROM is similar to _____.

d) ROM uses _____ to minimize the number of address lines.

e) _____ is done by manufacturer during the last fabrication process of the unit.

f) Mask is used to control the _____ on the ROM chip.

g) In Mask ROM cell, if column lines are LOW, it represents as _____.

h) _____ are available in both bipolar and MOS technologies.

i) In an EPROM, the cell at each intersection has _____transistors.

j) In EPROM, the program can remove for reprogramming is done by _____.

Q.6. Write TRUE or FALSE

a) Two transistors used in EPROM are floating gate and control gate.

b) ROM is a volatile memory.

c) ROM access time is measured from the chip select activation to the occurrence of valid output data.

d) Flash memory is used to speed up the read process of ROM chip.

e) EEPROM is a low power semiconductor device and it occupies less memory space.



15.9 LET US SUM UP

- Data used in a program as well as instructions for executing program are stored in memory device. So, digital systems require memory facilities for temporary as well as for permanent storage of data to

perform their functions.

- Based on the method of access, memory devices can be classified into different categories.
- In RAM any location can be accessed in a random manner and thus has equal access time for all memory locations.
- RAM chip can store information only as long as computer has electronic power.
- A semiconductor RAM is arranged in the form of a matrix of “m” words and each word is “n” bits wide and makes *cells*. Each cell has unique address representation.
- The address and data stored in a specific location of RAM are binary in form.
- An SRAM cell has three different states and they perform differently.
- DRAM use capacitance of a transistor as the storage device.
- Depending on access mechanism, RAM is organized into different dimensions most commonly like 2D, 3D, 2.5D etc.
- In comparison to 2D organization, 3D is more economical.
- In order to provide flexibility in some ROM applications, different programmable ROMs are introduced as PROM, EPROM and EEPROM.
- PROMs are widely used in the control of electrical equipment such as washing machine and electric oven.
- DRAM stores its binary information in the form of electric charges on capacitors.
- The bipolar ROMs can be subdivided into Mask ROMs and PROMs.
- Flash memories are basically EEPROMs except that erasure occurs at the block level in order to speed up the write process.



15.10 FURTHER READING

- Talukdar, P (2010), Digital Techniques N.L. Publication.
- Mano, M. M. (2002), EBSCO Publishing, Inc.

- Ram. B. (2002) Computer Fundamentals, Architecture and Organisation New Age International.

15.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : a

Ans. to Q. No. 2 : a

Ans. to Q. No. 3 : a) Volatile, b) DRAM, c) charged, d) MOSFET, e) 6 (six), f) SRAM, g) 2^n, h) MOSFET , i) SRAM, j) 1973

Ans. to Q. No. 4 : a) T, b) F, c) T, d) F, e) T

Ans. to Q. No. 5 : a) Boot, b) distributed system, c) SRAM, d) Decoders, e) mask programming, f) electrical interconnections, g) 0 (zero), h)PROMs, i) two, j) UV light.

Ans. to Q. No. 6 : a) T, b) F, c) T, d) F, e) T

15.12 MODEL QUESTIONS

Q.1. What is RAM? Give an example.

Q.2. What is access time? How to find ROM access time in a computer?

Q.3. Explain ROM family. How it is classified?

Q.4. Discuss the applications of ROM.

Q.5. Give two examples of non-volatile memory devices.

Q.6. Give the classification of semiconductor RAMs.

Q.7. What is the purpose of employing address multiplexing in a DRAM?

Q.8. Differentiate SRAM with DRAM.

Q.9. Describe the ROM internal structure.

Q.10. Draw an internal logic of 32 x 8 word ROM memory circuit by using OR output gates.

Q.11. Discuss the 3D structure of RAM with their internal organizational structure.

Q.12. List the pins required by a (64 x 4) bit RAM memory chip.

Q.13. Draw the block diagram of a typical (2048 x 16) bits ROM and describe its working principles.

Q.14. What is flash memory? Where it is used?

Q.15. A memory chip is organized as (1024 x 4) bits RAM. Find the number of such chips required to obtain:

a) (2048 x 8) RAM

b) 4k bytes of RAM

****