ACS College of Engineering

Approved by AICTE New Delhi, Affiliated to VTU, Belagavi

(A Unit of RajaRajeswari Group of Institutions)

DEPARTMENT OF AEROSPACE ENGINEERING

18ASL76

LAB MANUAL

SPACE SIMULATION LABORATORY

Outcomes:

1. Understand the basics of stability analysis.
2. Acquire the knowledge on Hoffmann transfer and orbit maneuvering.
3. Get the ideas about the orbital perturbations.
4. To provide background and fundamentals of MATLAB tools
5. To understand the concept and importance of Fourier and Z-Transforms

LIST OF EXPERIMENTS

1. Plot root locus with variables in transfer function through MATLAB.
2. Plot root locus for a dynamic system through MATLAB.
3. Draw Bode plot for a transfer function in MATLAB and explain Gain margin and Phase margin.
4. Simulate a servo mechanism motion with feedback in the following domains
    a. Time Domain
    b. Laplace Domain
5. Simulate a space shuttle landing with parachute deployed.
6. Simulate Hohmann transfer orbit.
7. Perform a planetary orbit simulation.
8. Simulate the Position of a moving object using GNSS simulator/Given position vectors.
9. Model a satellite motion and determine time period for its orbital motion.
10. Perform trajectory simulation of a small atmospheric re-entry module.
11. Perform and validate with an experimental setup for a simple feedback servo experiment.
12. Perform 3-DOF Gyroscope experiment for System Identification.
13. Perform 2- DOF Rotor System experiment for Coupled Dynamic Analysis
14. Model and simulate a simple Magnetic Levitation system and validate with the experimental setup.

CONTENTS

INTRODUCTION TO MATLAB:

- The name MATLAB stands for **MAT**rix **LA**boratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.
- MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.
- MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.
- It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.
- ***Starting MATLAB:*** you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

  •The Command Window

  •The Command History

  •The Workspace

  • The Current Directory

  • The Help Browser

  • The Start button

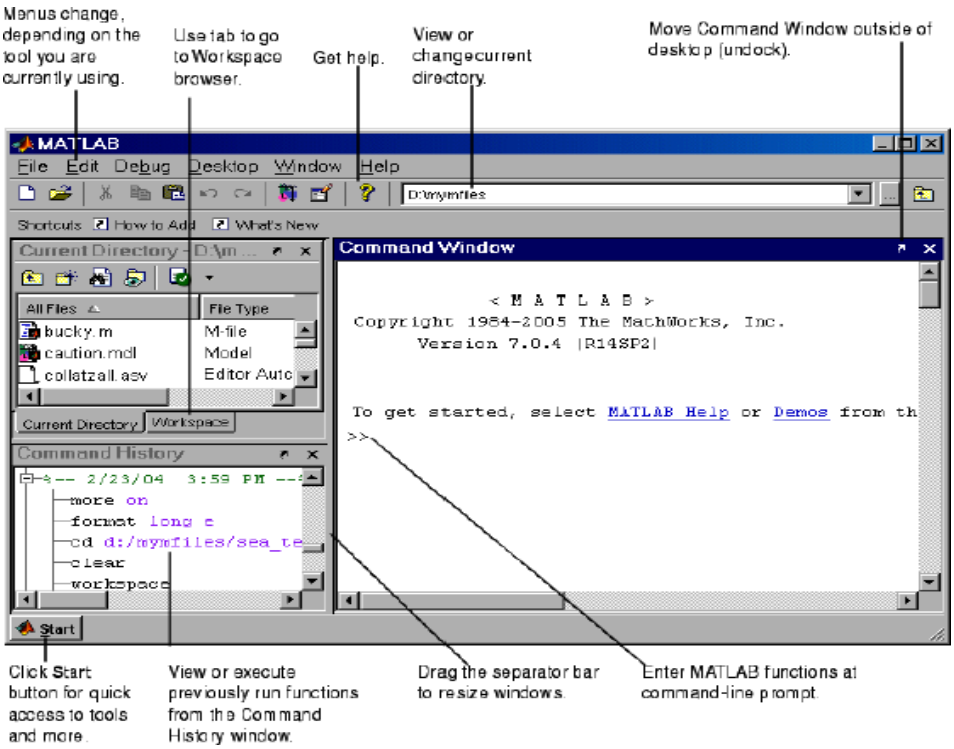


Fig:1 The graphical interface to the MATLAB workspace

When MATLAB is started for the first time, the screen looks like the one that shownin the Figure 1.1. This illustration also shows the default configuration of the MATLABdesktop. You can customize the arrangement of tools and documents to suit your needs.Now, we are interested in doing some simple calculations. We will assume that you have sufficient understanding of your computer under which MATLAB is being run.You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt: >> for full version EDU> for educational version

- ***Quitting MATLAB:*** To end your MATLAB session, type quit in the Command Window, or select File → Exit MATLAB in the desktop main menu.

- ***Creating MATLAB variables:*** MATLAB variables are created with an assignment statement. The syntax of variable assignment is

  variable name = a value (or an expression)

  For example,

  >> x = expression

Where expression is a combination of numerical values, mathematical operators, variables, and function calls. On other words, expression can involve:

1. manual entry
2. built-in functions
3. user-defined functions

- ***Overwriting variable***: Once a variable has been created, it can be reassigned. In addition, if you do not wish tosee the intermediate results, you can suppress the numerical output by putting a semicolon(;) at the end of the line. Then the sequence of commands looks like this:

```
>> t = 5;
>> t = t+1
t =    6
```

- ***Error messages:*** If we enter an expression incorrectly, MATLAB will return an error message. For example, in the following, we left out the multiplication sign, *, in the following expression

```
>> x = 10;
>> 5x
??? 5x
     |
Error: Unexpected MATLAB expression.
```

- ***Making corrections:*** To make corrections, we can, of course retype the expressions. But if the expression is lengthy, we make more mistakes by typing a second time. A previously typed command can be recalled with the up-arrow key". When the command is displayed at the command prompt, it can be modified if needed and executed.
- ***Controlling the appearance of floating point number***: MATLAB by default displays only 4 decimals in the result of the calculations, for example 163:6667, as shown in above examples. However, MATLAB does numerical calculation in double precision, which is 15 digits. The command format controls how the results of computations are displayed. Here are some examples of the di®erent formats together withthe resulting outputs.

```
>> format short

>> x=-163.6667
```

If we want to see all 15 digits, we use the command format long

```
>> format long

>> x= -1.636666666666667e+002
```

To return to the standard format, enter format short, or simply format.There are several other formats. For more details, see the MATLAB documentation,or type help format.Note - Up to now, we have let MATLAB repeat everything that we enter at theprompt (>>). Sometimes this is not quite useful, in particular when the output is pages enlength. To prevent MATLAB from echoing what we type, simply enter a semicolon (;) atthe end of the command. For example,

```
>> x=-163.6667;
```

and then ask about the value of x by typing,

```
>> x

x =-163.6667
```

- ***Managing the workspace:*** The contents of the workspace persist between the executions of separate commands. Therefore, it is possible for the results of one problem to have an e®ect on the next one. To avoidthis possibility, it is a good idea to issue a clear command at the start of each new independent calculation.

```
>> clear
```

The command clear or clear all removes all variables from the workspace. Thisfrees up system memory. In order to display a list of the variables currently in the memory,type

```
>> who
```

while, whos will give more details which include size, space allocation, and class of thevariables.

- ***Keeping track of your work session:*** It is possible to keep track of everything done during a MATLAB session with the diary command.

```
>> diary
```

or give a name to a created file,

```
>> diary FileName
```

where FileName could be any arbitrary name you choose.The function diary is useful if you want to save a complete MATLAB session. Theysave all input and output as they appear in the MATLAB window. When you want to stopthe recording, enter diary off. If you want to start recording again, enter diary on. The file that

is created is a simple text ¯le. It can be opened by an editor or a word processingprogram and edited to remove extraneous material, or to add your comments. You canuse the function type to view the diary ¯le or you can edit in a text editor or print. Thiscommand is useful, for example in the process of preparing a homework or lab submission.

- ***Entering multiple statements per line:***It is possible to enter multiple statements per line. *Use* commas (,) or semicolons (;) toenter more than one statement at once. Commas (,) allow multiple statements per linewithout suppressing output.

```
>> a=7; b=cos(a), c=cosh(a)
b =0.6570
c =548.3170
```

- ***Miscellaneous commands***Here are few additional useful commands:
  1. To clear the Command Window, type clc
  2. To abort a MATLAB computation, type ctrl-c
  3. To continue a line, type . . .
- ***Getting help:*** To view the online documentation, select MATLAB Helpfrom Help menu of MATLAB Helpdirectly in the Command Window. The preferred method is to use the *Help Browser*. TheHelp Browser can be started by selecting the ? icon from the desktop toolbar. On the otherhand, information about any command is available by typing

```
>> help Command
```

Another way to get help is to use the lookfor command. The lookfor command di®ersfrom the help command. The help command searches for an exact function name match,while the lookfor command searches the quick summary information in each function fora match. For example, suppose that we were looking for a function to take *the inverse ofa matrix*. Since MATLAB does not have a function named inverse, the command helpinverse will produce nothing. On the other hand, the command lookfor inverse willproduce detailed information, which includes the function of interest, inv.

```
>> lookfor inverse
```

Note - At this particular time of our study, it is important to emphasize one main point.Because MATLAB is a huge program; it is impossible to cover *all the details* of each functionone by one. However, we will give you information how to get help. Here are some examples:

- Use on-line help to request info on a specific function

```
>> help sqrt
```

- In the current version (MATLAB version 7), the doc function opens the on-line versionof the help manual. This is very helpful for more complex commands.

```
>> doc plot
```

- Use lookfor to find functions by keywords. The general form is

```
>> lookfor FunctionName
```

 Programming in MATLAB: So far in these lab sessions, all the commands were executed in the Command Window.The problem is that the commands entered in the Command Window cannot be savedand executed again for several times. Therefore, a different way of executing repeatedlycommands with MATLAB is:

1. to create a file with a list of commands,
2. save the file, and
3. run the file.

If needed, corrections or changes can be made to the commands in the file. The files thatare used for this purpose are called script files or scripts for short.

This section covers the following topics:

 M-File Scripts: A *script file* is an external file that contains a sequence of MATLAB statements. Scriptfiles have a filename extension .m and are often called M-files. M-files can be *scripts* thatsimply execute a series of MATLAB statements, or they can be *functions* that can acceptarguments and can produce one or more outputs.

Example
Consider the system of equations:

$$x + 2y + 3z = 1$$
$$3x + 3y + 4z = 1$$
$$2x + 3y + 3z = 2$$

Find the solution x to the system of equations.

Solution:
Use the MATLAB *editor* to create a file: File New M-file.
Enter the following statements in the file:

```
A = [1 2 3; 3 3 4; 2 3 3];
b = [1; 1; 2];
x = A\b
```

Save the file, for example, example1.m.
Run the file, in the command line, by typing:

```
>> example1
x =
   -0.5000
    1.5000
   -0.5000
```

When execution completes, the variables (A, b, and x) remain in the workspace. To see a listing of them, enter whos at the command prompt.

Note: The MATLAB editor is both a text editor specialized for creating M-files and agraphical MATLAB debugger. The MATLAB editor has numerous menus for tasks such as*saving*, *viewing*, and *debugging*. Because it performs some simple checks and also uses color to differentiate between various elements of codes, this text editor is recommended as thetool of choice for writing and editing M-files.

There is another way to open the editor:

```
>> edit
```

or

```
>> edit filename.m
```

to open filename.m.

- M-File Functions: As mentioned earlier, functions are programs (or *routines*) that accept *input* arguments andreturn *output* arguments. Each M-file function (or *function* or *M-file* for short) has its *own*area of workspace, separated from the MATLAB base workspace.
- Anatomy of a M-File function

This simple function shows the basic parts of an M-file.

```
function f = factorial(n)                    (1)
% FACTORIAL(N) returns the factorial of N.   (2)
% Compute a factorial value                  (3)
f = prod(1:n);                               (4)
```

The first line of a function M-file starts with the keyword function. It gives the function*name* and order of *arguments*. In the case of function factorial, there are up to one outputargument and one input argument. Table given below summarizes the M-file function.

As an example, for $n = 5$, the result is,

```
>> f = factorial(5)
 f =120
```

Table: Anatomy of a M-File function

Part No	M-File Element	Description
1	Function Definition Line	Define the function name, and the definition number and order of input and line output arguments
2	H1 Line	A one line summary description of the program, displayed when you request Help
3	Help Text	Help text A more detailed description of the program
4	Function body	Function body Program code that performs the actual computations

Both functions and scripts can have all of these parts, except for the function definition line which applies to function only. In addition, it is important to note that function name must begin with a letter, andmust be no longer than the maximum of 63 characters. Furthermore, the name of thetextfile that you save will consist of the function name with the extension .m. Thus, theabove example file would be factorial.m.

Table: Difference between scripts and function

SCRIPTS	FUNCTION
• Do not accept input arguments. • Store variables in a workspace that is shared with other scripts. • Are useful for automating a series of commands	• Can accept input arguments and return output arguments. • Store variables in workspace that is internal to the function • Are useful for extending the MATLAB a series of commands language for your application

INTRODUCTION TO VHDL:

VHDL is an acronym for VHSIC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuit).It is a Hardware Description Language that can be used to model a digital system at many levels of abstraction,ranging from algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of simple gate to a complex digital electronic system or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description.

The VHDL language can be regarded as an integrated amalgamation of following languages.

Sequential language +

Concurrent language +

Net-list language +

Timing specifications +

Waveform generation language =>VHDL.

The language not only defines the syntax but also defines very clear simulation semantics for each language construct. Therefore models written in this language can be verified using a VHDL simulation.

CAPABILITIES:

The following are the major capabilities that the language provides along with the features that differentiate it from other hardware description languages.

- The language can be used as an exchange medium between chip vendors and CAD tool users. Different chip vendors can provide VHDL descriptions of their components to system designers. CAD tool users can use it to capture the behavior of the design at a high level of abstraction of functional simulation.
- The language can also be used as a communication medium between different CAD and CAE tools. For example, a schematic capture PROGRAM may be used to generate a VHDL description for the design which can be used as an input to a simulation PROGRAM.
- The language supports hierarchy, that is, a digital system can be modeled as a set of interconnected subcomponents.
- The language supports flexible design methodologies: top-down, bottom-up or mixed.
- It supports both synchronous and asynchronous timing models.

- Various digital modeling techniques, such as finite state machine descriptions, algorithmic descriptions and Boolean equations can be modeled using the language.
- The language supports three basic different description styles: structural, dataflow and behavioral. A design may also be expressed in any combination of these three descriptive styles.
- The language is not technology-specific, but is capable of supporting technology specific features. It can also support various hardware technologies.

BASIC TERMINOLOGY:

A hardware abstraction of a digital system is called an entity. An entity X when used in another entity Y becomes a component for the entity Y. therefore the component is also an entity, depending on the level at which you are trying to model.

To describe an entity, VHDL provides five different types of primary constructs called design units. They are:

- Entity declaration.
- Architecture body.
- Configuration declaration.
- Package declaration.
- Package body.

- **ENTITY DECLARATION:**

The entity declaration specifies the name of the entity being modeled and lists the set of interface ports. Ports are signals through which the entity communicates with the other models in its external environment.

- **ARCHITECTURE BODY:**

The internal details of an entity are specified by an architecture body using any of the following modeling styles:

- As a set of interconnected components (to represent structure).
- As a set of concurrent assignment statements (to represent dataflow).
- As a set of sequential assignment statements (to represent behavior).
- **CONFIGURATION DECLARATION:**

This is used to select one of the many possibly architecture bodies that an entity may have, and to bind components , used to represent structure in that architecture body, to entities represented by an entity-architecture pair or by a configuration which reside in a design library.

- **PACKAGE DECLARATION:**

This is used to store a set of common declarations, such as components, types, procedures and functions. These declarations can then be imported into other design units using a „use" clause.

- **PACKAGE BODY:**

This is used to store the definitions of functions and procedures that were declared in the corresponding package declaration, and also complete constant declarations for any deferred constants that appear in the package in the package declaration.

STRUCTURAL MODELING:

In the structural style of modeling, an entity is described as a set of interconnected components. Example: Half adder. The entity declaration for half adder specifies the interface ports for this architecture body. The architecture body is composed of two parts: the declarative part (before the keyword **begin**) and the statement part(after the keyword **begin**). Two component declarations are present in the declarative part of the architecture body. These declarations specify the interface of components that are used in the architecture body. The declared components are instantiated in the statement part of the architecture body using component labels for these component instantiation statements. The signals in the port map of a component instantiated and the port signals in the component declaration are associated by position (called positional association). However the structural representation for the Half adder does not say anything about its functionality. Separate entity models would be described for the components XOR2 and AND2, each having its own entity declaration and architecture body.

A component instantiated statement is a concurrent statement. Therefore, the order of these statements is not important. The structural style of modeling describes only an interconnection of components, without implying any behavior of the components themselves nor the entity that they collectively represent.

DATAFLOW MODELING:

In this modeling style, the flow of data through the entity is expressed primarily using concurrent signal assignment statements. The structure entity of the entity is not explicitly specified

in this modeling style, but it can be implicitly deduced. In a signal assignment statement, the symbol <= implies an assignment of a value to a signal. The value of the expression on the right-hand-side of the statement is computed and is assigned to the signal on the left-hand-side, called the target signal. A concurrent signal assignment statement is executed only when any signal used in the expression on the right-hand-side has an event on it, that is, the value for the signal changes.

BEHAVIORAL MODELING:

The behavioral modeling specifies the behavior of an entity as a set of statements that are executed sequentially in the specified order. This set of sequential statements, which are specified inside a process statement, do not explicitly specify the structure of the entity but merely its functionality. A process statement is a concurrent statement that can appear within an architecture body. A process statement also has a declarative part (before the keyword **begin**) and a statement part (between the keywords **begin** and **end process**). The statements appearing within the statement part are sequential statements and are executed sequentially. The list of signals specified within the parenthesis after the keyword **process** constitutes a sensitivity list, and the process statement is invoked whenever there is an event on any signal in this list.

A variable is assigned using the assignment operator: = compound symbol; contrast this with a signal that is assigned a value using the assignment operator <= compound symbol. Signal assignment statements appearing within a process are called sequential signal assignment statements. Sequential signal statements, including variable assignment statements, are executed sequentially independent of whether an event occurs on any signals in its right-hand-side expression; contrastthis with the execution of concurrent signal assignment statements in the dataflow modeling style.

EXPERIMENT NO-1

AIM: -
To write a MATLAB program to perform some basic operation on matrices such as addition, subtraction, multiplication.

SOFTWARE REQURIED:-
1.MATLAB R2010a.
2.Windows XP SP2.

THEORY:-
MATLAB, which stands for MATrixLABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation. As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

Built in Functions:
1. Scalar Functions:
Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector). They are summarized below.
1. sin - trigonometric sine
2. cos - trigonometric cosine
3. tan - trigonometric tangent
4. asin - trigonometric inverse sine (arcsine)
5. acos - trigonometric inverse cosine (arccosine)
6. atan - trigonometric inverse tangent (arctangent)
7. exp - exponential
8. log - natural logarithm
9. abs - absolute value
10. sqrt - square root
11. rem - remainder
12. round - round towards nearest integer
13. floor - round towards negative infinity
14. ceil - round towards positive infinity

2. Vector Functions:
Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these functions are given below.
1. max largest component : get the row in which the maximum element lies
2. min smallest component
3. lengthlength of a vector
4. sortsort in ascending order
5. sumsum of elements
6. prod product of elements
7. medianmedian value
8. meanmean value std standard deviation

3. Matrix Functions:
Much of MATLAB‟s power comes from its matrix functions. These can be further separated into two sub-categories.
The first one consists of convenient matrix building functions, some of which are given below.
1. eye - identity matrix
2. zeros - matrix of zeros
3. ones - matrix of ones
4. diag - extract diagonal of a matrix or create diagonal matrices
5. triu - upper triangular part of a matrix
6. tril - lower triangular part of a matrix
7. rand - randomly generated matrix

commands in the second sub-category of matrix functions are
1. sizesize of a matrix
2. det determinant of a square matrix
3. inv inverse of a matrix
4. rankrank of a matrix
5. rref reduced row echelon form
6. eig eigenvalues and eigenvectors
7. poly characteristic polynomial

PROCEDURE:-
□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

PROGRAM:-

```
clc;
close all;
clear all;
a=[1 2 -9 ; 2 -1 2; 3 -4 3];
b=[1 2 3; 4 5 6; 7 8 9];
disp('The matrix a= ');
a
disp('The matrix b= ');
b
% to find sum of a and b
c=a+b;
disp('The sum of a and b is ');
c
% to find difference of a and b
d=a-b;
disp('The difference of a and b is ');
d
%to find multiplication of a and b
e=a*b;
disp('The product of a and b is ');
e
```

OUTPUT:-

```
The matrix a=
a =
1 2 -9
2 -1 2
3 -4 3
The matrix b=
b =
1 2 3
4 5 6
7 8 9
The sum of a and b is
c =
2 4 -6
6 4 8
10 4 12
The difference of a and b is
d =
0 0 -12
-2 -6 -4
-4 -12 -6
The product of a and b is
e =
-54 -60 -66
12 15 18
8 10 12
```

RESULT:-
Finding addition, subtraction, multiplication using MATLAB was Successfully completed.

POSSIBLE VIVA QUESTIONS:-
1.Expand MATLAB?And importance of MATLAB?
2.What is clear all and close all will do?
3.What is disp() and input()?
4.What is the syntax to find the eigen values and eigenvectors of the matrix?
5. What is the syntax to find the rank of the matrix?

EXERCISE:
1. Enter the matrix
M = [1,-2,8,0] and N = [1 5 6 8; 2 5 6 9]
Perform addition on M and N and see how matlab reacts.
2. Find the transpose of null matrix using matlab
3. write a MATLAB program to perform the division operation on the following matrix
A = [24,-30, 64,-81], b= [6,5,8,9] and verify the result.
4. Write a matlab program to perfom addition operation using 2x3 matrix. Assume any numbers
5. Enter the matrix
A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7], B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9],
C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4]
Find [(A+B)+C]T
6. Enter the matrix
A = [1 6 9 8 5; 9 3 5 8 4; 5 6 3 5 7], B = [6 5 9 3 5; 6 5 4 8 5; 6 3 5 7 9],
C = [2 5 9 3 4; 5 6 3 7 8; 9 8 6 5 4]
Find $[(A-B)+C]^{-1}$
7. Write a matlab program to perfom addition operation using 3x2 matrix. Assume any numbers
8 write a MATLAB program to perform the division operation on the following matrix A = [25,-35, 121,-21], b= [5,5,11,3] and perform the transpose function on the answer
9.Find the addition of null matrix and unity matrix of order 3x3.
10. Enter the Matrix the following Matrices and multiply M and N using M*N. Observe the output in the command window.

$$M = \begin{matrix} -1 & 2 & 4 \\ 2 & -1 & -1 \\ 4 & 2 & 0 \end{matrix} \quad N = \begin{matrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{matrix}$$

Experiment No-2

AIM:-To write a "MATLAB" Program to generate various signals and sequences,such as unit impulse, unit step, unit ramp, sinusoidal,square,sawtooth,triangular,sinc signals.

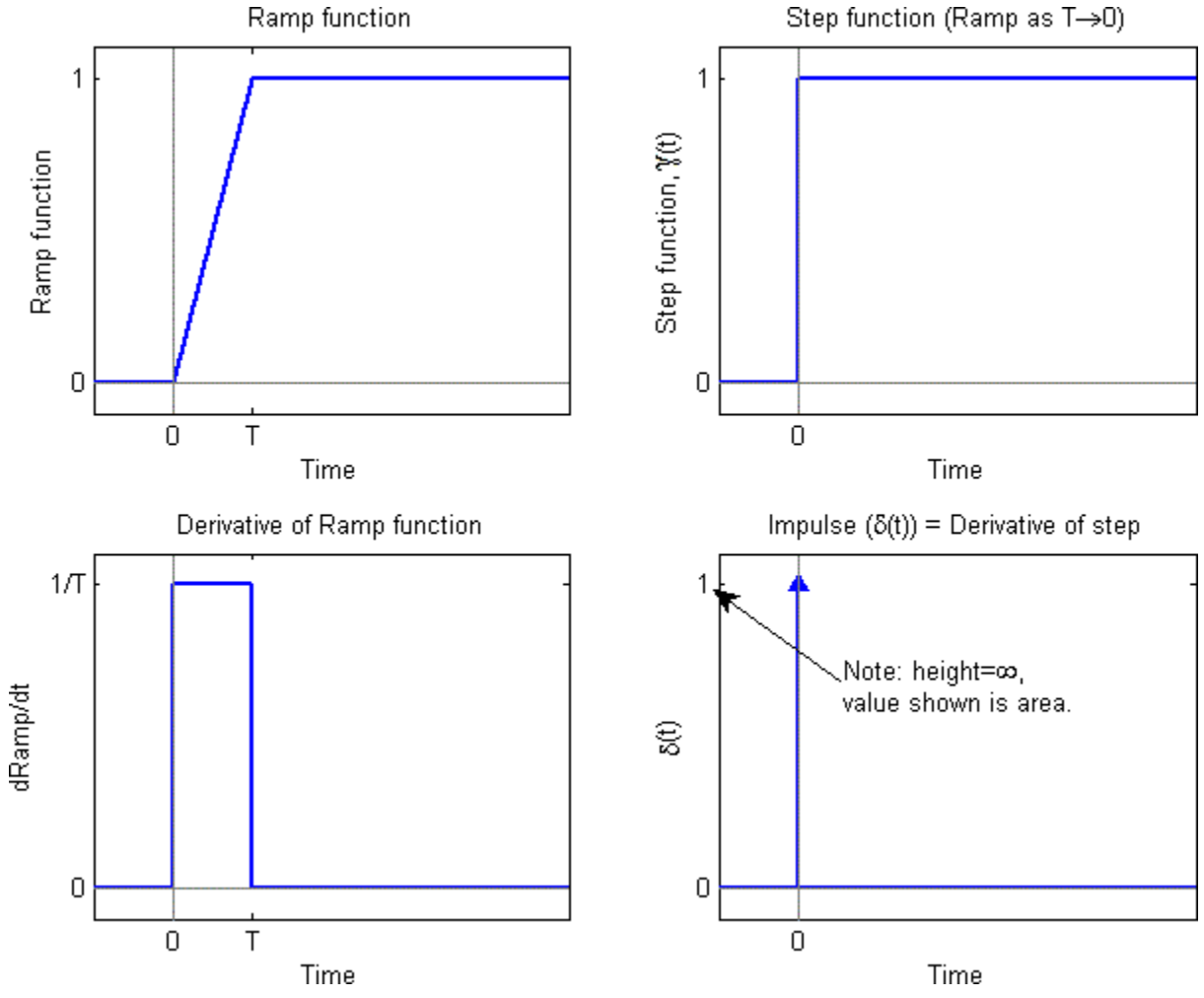
SOFTWARE REQURIED:-

1.MATLAB R2010a.
2.Windows XP SP2.

THEORY:-

UNIT IMPULSE FUNCTION:

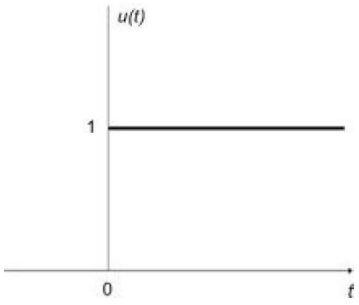
One of the more useful functions in the study of linear systems is the "unit impulse function."An ideal impulse function is a function that is zero everywhere but at the origin, where it is infinitely high. However, the *area* of the impulse is finite. This is, at first hard to visualize butwe can do so by using the graphs shown below.



UNIT STEP FUNCTION

The unit step function and the impulse function are considered to be fundamental functions in engineering, and it is strongly recommended that the reader becomes very familiar with both of these functions.The unit step function, also known as the Heaviside function, is defined as such:

$$u(t) = \begin{cases} 0, & \text{if } t < 0 \\ 1, & \text{if } t > 0 \\ \frac{1}{2}, & \text{if } t = 0 \end{cases}$$



Sinc Function

There is a particular form that appears so frequently in communications engineering, that wegive it its own name. This function is called the "Sinc function and discussed below

The Sinc function is defined in the following manner:

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \text{ if } x \neq 0$$

And Sinc(0)=1

The value of sinc(x) is defined as 1 at x = 0, since

$$\lim_{x \to 0} \operatorname{sinc}(x) = 1$$

***Rect* Function**

The Rect Function is a function which produces a rectangularcentered at t = 0. The Rect function pulse also has a height of 1. The Sinc function and therectangular function form a Fourier transform pair.A *Rect* function can be written in the form:

$$\operatorname{rect}\left(\frac{t - X}{Y}\right)$$

where the pulse is centered at X and has widthY. We can define the impulse function above interms of the rectangle function by centering the pulse at zero (X = 0), setting it's height to 1/Aand setting the pulse width to A, which approaches zero:

$$\delta(t) = \lim_{A \to 0} \frac{1}{A} \operatorname{rect}\left(\frac{t - 0}{A}\right)$$

We can also construct a *Rect* function out of a pair of unit step functions

$$\operatorname{rect}\left(\frac{t - X}{Y}\right) = u(t - X + Y/2) - u(t - X - Y/2)$$

Here,both unit step functions are set a distance of Y/2 away from the center point of (t - X).

SAW TOOTH:-

The sawtooth wave (or saw wave) is a kind of non-sinusoidal waveform. It is named a sawtooth based on its resemblance to the teeth on the blade of a saw. The convention is that a sawtooth wave ramps upward and then sharply drops. However, there are also sawtooth waves in which the wave ramps downward and then sharply rises. The latter type of sawtooth wave is called a 'reverse sawtooth wave' or 'inverse sawtooth wave'. As audio signals, the two orientations of sawtooth wave sound identical. The piecewise linear function based on the floor function of time t, is an example of a sawtooth wave with period 1.

$$x(t) = 2\left(\frac{t}{a} - \text{floor}\left(\frac{t}{a} + \frac{1}{2}\right)\right)$$

TRIANGLE WAVE

A triangle wave is a non-sinusoidal waveform named for its triangular shape.Abandlimited triangle wave pictured in the time domain (top) and frequency domain (bottom). The fundamental is at 220 Hz (A2).Like a square wave, the triangle wave contains only odd harmonics. However, the higher harmonics roll off much faster than in a square wave (proportional to the inverse square of the harmonic number as opposed to just the inverse).It is possible to approximate a triangle wave with additive synthesis by adding odd harmonics of the fundamental, multiplying every (4n..1)th harmonic by 1 (or changing its phase by), and rolling off the harmonics by the inverse square of their relative frequency to the fundamental. This infinite Fourier series converges to the triangle wave:

$$\begin{aligned} x_{\mathbf{triangle}}(t) &= \frac{8}{\pi^2}\sum_{k=0}^{\infty}(-1)^k\frac{\sin\left((2k+1)\omega t\right)}{(2k+1)^2} \\ &= \frac{8}{\pi^2}\left(\sin(\omega t) - \frac{1}{9}\sin(3\omega t) + \frac{1}{25}\sin(5\omega t) - \cdots\right) \end{aligned}$$

where ω is the angular frequency.

Sinusoidal Signal Generation

The sine wave or sinusoid is a mathematical function that describes a smooth repetitiveoscillation. It occurs often in pure mathematics, as well as physics, signal processing, electrical engineering and many other fields. Its most basic form as a function of time (t)is:where:

• A, the amplitude, is the peak deviation of the function from its center position.

• the angular frequency, specifies how many oscillations occur in a unit time interval, in radians per second

• the phase, specifies where in its cycle the oscillation begins at t = 0.

A sampled sinusoid may be written as:

$$x(n) = Asin(2\pi\frac{f}{f_s}n + \theta)$$

where f is the signal frequency, fs is the sampling frequency, θ is the phase and A is theamplitude of the signal.

PROCEDURE:-

□ Open MATLAB

□ Open new M-file

□ Type the program

□ Save in current directory

□ Compile and Run the program
□ For the output see command window\ Figure window

PROGRAM:-

%unit impulse function%

```
clc;
clearall;
closeall;
t=-10:1:10;
x=(t==0);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit impulse function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit impulse discreat function');
```

%unit step function%

```
clc;
clearall;
closeall;
N=100;
t=1:100;
x=ones(1,N);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit step function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit step discreat function');
```

%unit ramp function%

```
clc;
clearall;
closeall;
t=0:20;
x=t;
```

```
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('unit ramp function');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('unit ramp discreat function');
```

%sinusoidal function%

```
clc;
clearall;
closeall;
t=0:0.01:2;
x=sin(2*pi*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sinusoidal signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sinusoidal sequence');
```

%square function%

```
clc;
clearall;
closeall;
t=0:0.01:2;
x=square(2*pi*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('square signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('square sequence');
```

%sawtooth function%

```
clc;
clearall;
```

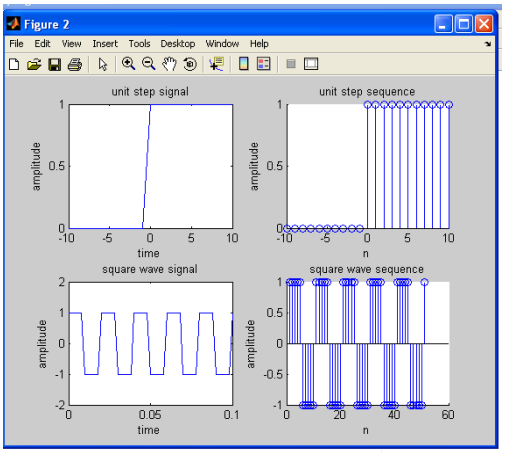
```
closeall;
t=0:0.01:2;
x=sawtooth(2*pi*5*t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sawtooth signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sawtooth sequence');
```

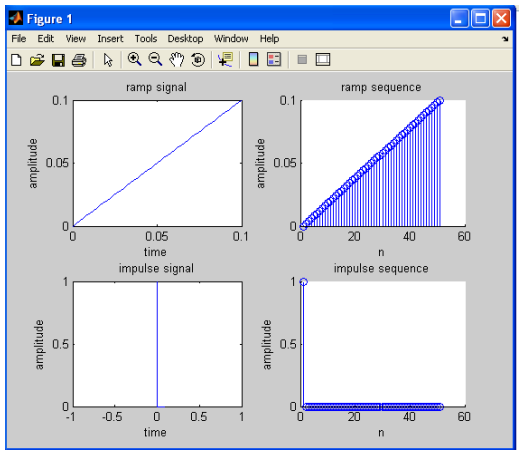
%trianguler function%

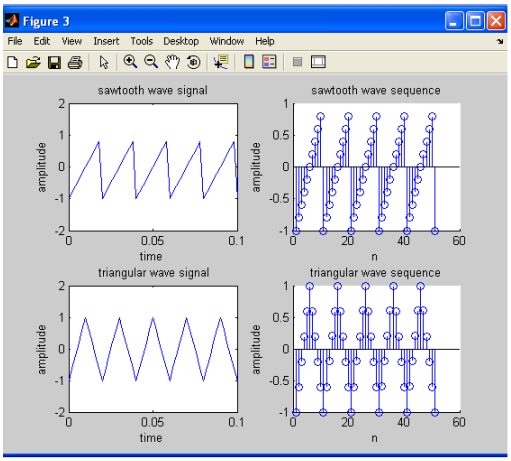
```
clc;
clearall;
closeall;
t=0:0.01:2;
x=sawtooth(2*pi*5*t,0.5);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('trianguler signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('trianguler sequence');
```

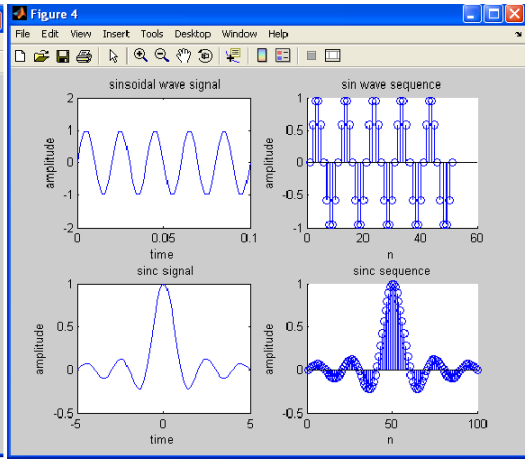
%sinc function%

```
clc;
clearall;
closeall;
t=linspace(-5,5);
x=sinc(t);
subplot(2,1,1);
plot(t,x,'g');
xlabel('time');
ylabel('amplitude');
title('sinc signal');
subplot(2,1,2);
stem(t,x,'r');
xlabel('time');
ylabel('amplitude');
title('sinc sequence');
```







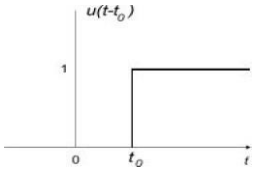


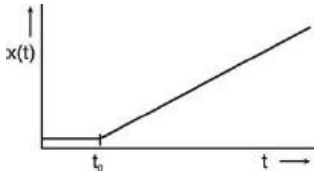
POSSIBLE VIVA QUESTIONS:-

1. Define Signal?
2. Define determistic and Random Signal?
3. Define Delta Function?
4. What is Signal Modeling?
5. Define Periodic and a periodic Signal?

EXCERSISE

1. Write a matlab program to generate a sine wave with amplitude = 3, frequency 20Hz.
2. Write a matlab program to generate a cos wave with amplitude = 3, frequency 20Hz.
3. Write a matlab program to generate a triangular wave with amplitude = 8, frequency 10Hz.
4. Write a matlab program to generate a square wave with amplitude = 2, frequency 10kHz.
5. Write a matlab program to get the output shown below where $t_0 = 2$





5. Write a program to get the result in signalr(t) = u(t) – 2*u(t+1)

EXPERIMENT No-3

AIM:-

To performs operations on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and average power.

SOFTWARE REQURIED:-

1.MATLAB R2010a.
2.Windows XP SP2.

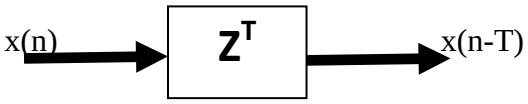
THEORY:-

Basic Operation on Signals:

Time shifting: y(t)=x(t-T)The effect that a time shift has on the appearance of a signal If T is a positive number, the time shifted signal, x(t -T) gets shifted to the right, otherwise it gets shifted left.

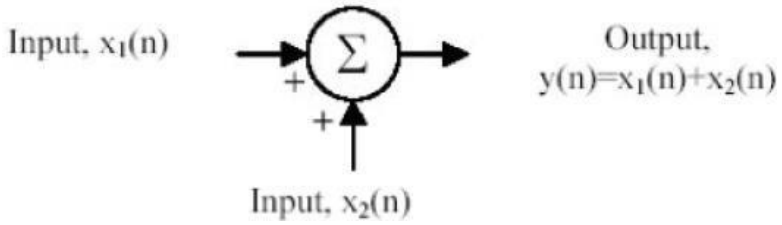
Signal Shifting and Delay:

Shifting : y(n)={x(n-k)} ; m=n-k; y=x;
Time reversal: Y(t)=y(-t) Time reversal flips the signal about t = 0

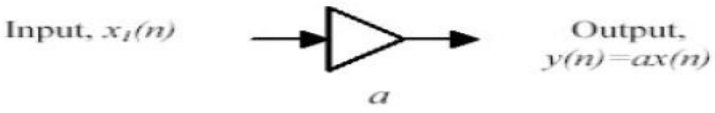


Signal Addition and Subtraction:

Addition: any two signals can be added to form a third signal,z (t) = x (t) + y (t)

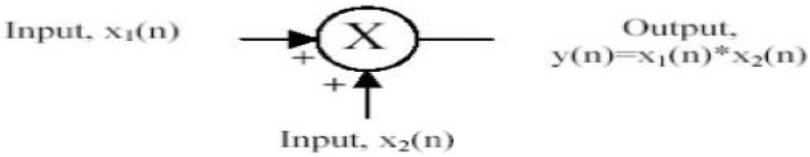


Signal Amplification/Attuation :



Multiplication/Divition :

of two signals, their product is also a signal.
z (t) = x (t) y (t)



folding:

y(n)={x(-n)} ; y=fliplr(x); n=-fliplr(n);

PROCEDURE:-

□Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

PROGRAM:-

```
clear   all;
close   all;
t=0:.01:1;
% generating two input signals
x1=sin(2*pi*4*t);
x2=sin(2*pi*8*t);
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('signal1:sine wave of frequency 4Hz');
subplot(2,2,2);
plot(t,x2);
xlabel('time');
subplot(4,1,3);
ylabel('amplitude');
title('signal2:sine wave of frequency 8Hz');
% addition of signals
y1=x1+x2;
subplot(2,2,3);
plot(t,y1);
xlabel('time');
ylabel('amplitude');
title('resultant signal:signal1+signal2');
% multiplication of signals
y2=x1.*x2;
subplot(2,2,4);
plot(t,y2);
xlabel('time');
ylabel('amplitude');
title('resultant signal:dot product of signal1 and signal2');
% scaling of a signal1
A=10;
y3=A*x1;
figure;
```

```
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
subplot(2,2,2);
plot(t,y3);
xlabel('time');
ylabel('amplitude');
title('amplified input signal1 ');
% folding of a signal1
l1=length(x1);
nx=0:l1-1;
subplot(2,2,3);
plot(nx,x1);
xlabel('nx');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
y4=fliplr(x1);
nf=-fliplr(nx);
subplot(2,2,4);
plot(nf,y4);
xlabel('nf');
ylabel('amplitude');
title('folded signal');
%shifting of a signal
figure;
t1=0:.01:pi;
x3=8*sin(2*pi*2*t1);
subplot(3,1,1);
plot(t1,x3);
xlabel('time t1');
ylabel('amplitude');
title('sine wave of frequency 2Hz');
subplot(3,1,2);
plot(t1+10,x3);
xlabel('t1+10');
ylabel('amplitude');
title('right shifted signal');
subplot(3,1,3);
plot(t1-10,x3);
xlabel('t1-10');
ylabel('amplitude');
title('left shifted signal');
%operations on sequences
n1=1:1:9;
s1=[1 2 3 0 5 8 0 2 4];
figure;
subplot(2,2,1);
stem(n1,s1);
```

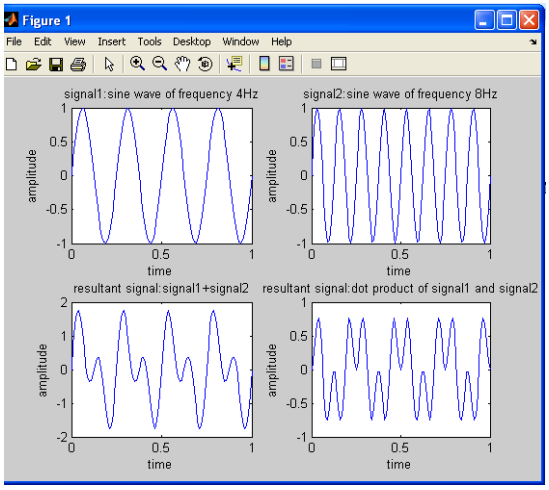
```
xlabel('n1');
ylabel('amplitude');
title('input sequence1');

n2=-2:1:6;
s2=[1 1 2 4 6 0 5 3 6];
subplot(2,2,2);
stem(n2,s2);
xlabel('n2');
ylabel('amplitude');
title('input sequence2');
% addition of sequences
s3=s1+s2;
subplot(2,2,3);
stem(n1,s3);
xlabel('n1');
ylabel('amplitude');
title('sum of two sequences');
% multiplication of sequences
s4=s1.*s2;
subplot(2,2,4);
stem(n1,s4);
xlabel('n1');
ylabel('amplitude');
title('product of two sequences');
% scaling of a sequence
figure;
subplot(2,2,1);
stem(n1,s1);
xlabel('n1');
ylabel('amplitude');
title('input sequence1');
s5=4*s1;
subplot(2,2,2);
stem(n1,s5);
xlabel('n1');
ylabel('amplitude');
title('scaled sequence1');

subplot(2,2,3);
stem(n1-2,s1);
xlabel('n1');
ylabel('amplitude');
title('left shifted sequence1');
subplot(2,2,4);
stem(n1+2,s1);
xlabel('n1');
ylabel('amplitude');
title('right shifted sequence1');
% folding of a sequence
```

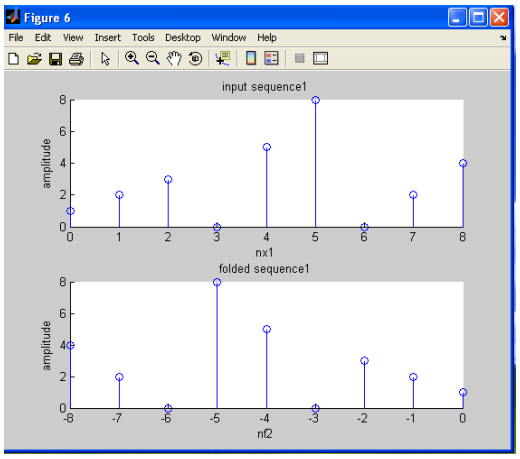
```
l2=length(s1);
nx1=0:l2-1;
figure;
subplot(2,1,1);
stem(nx1,s1);
xlabel('nx1');
ylabel('amplitude');
title('input sequence1');
s6=fliplr(s1);
nf2=-fliplr(nx1);
subplot(2,1,2);
stem(nf2,s6);
xlabel('nf2');
ylabel('amplitude');
title('folded sequence1');
% program for energy of a sequence

e1=sum(abs(z1).^2);
e1
% program for energy of a signal
t=0:pi:10*pi;
z2=cos(2*pi*50*t).^2;
e2=sum(abs(z2).^2);
e2
% program for power of a saequence
p1=  (sum(abs(z1).^2))/length(z1);
p1
% program for power of a signal
p2=(sum(abs(z2).^2))/length(z2);
p2
```

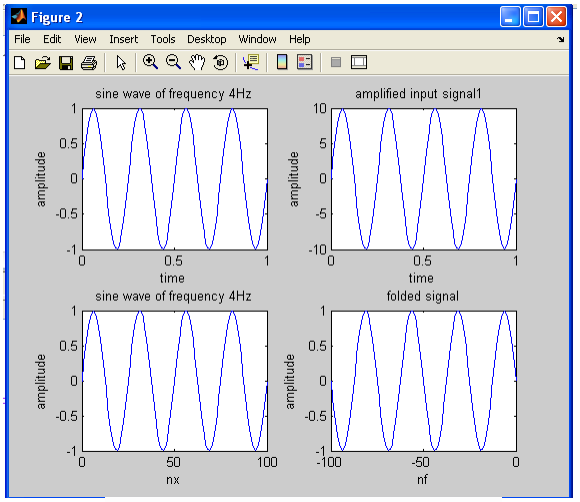
OUTPUT:

```
enter the input sequence [1 3 5 6]
e1 =
71
e2 =
4.0388
p1 =
17.7500
p2 =
0.3672
```

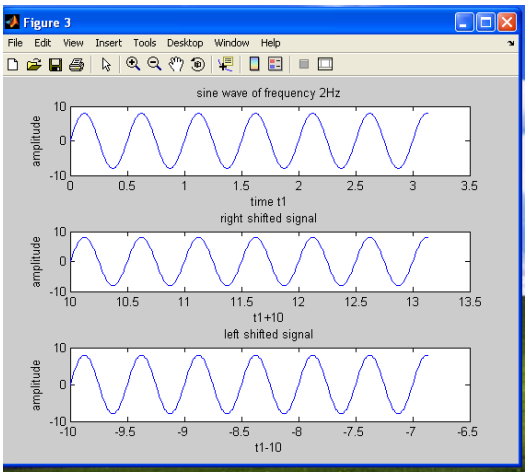
Result: Various operations on signals and sequences are performed.



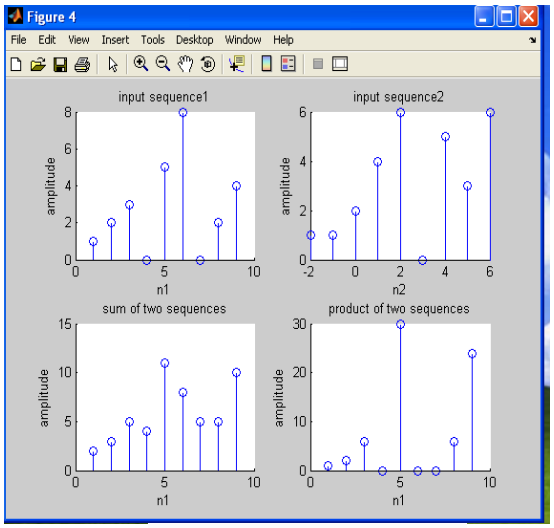




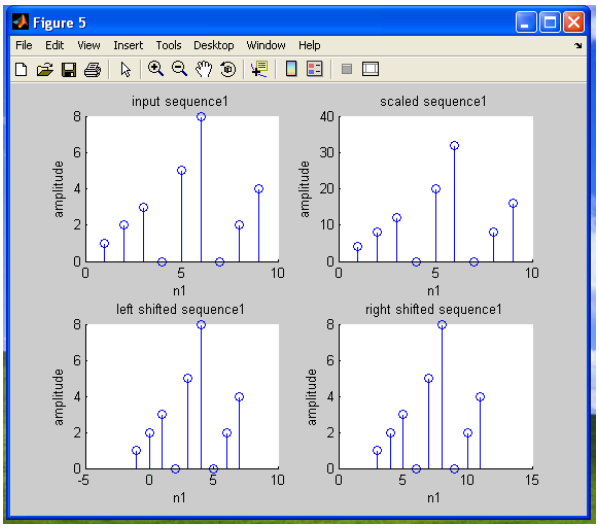
Amplitude scaling for signals



Time scaling for signals



Time shifting of a signal



Time folding of a signal

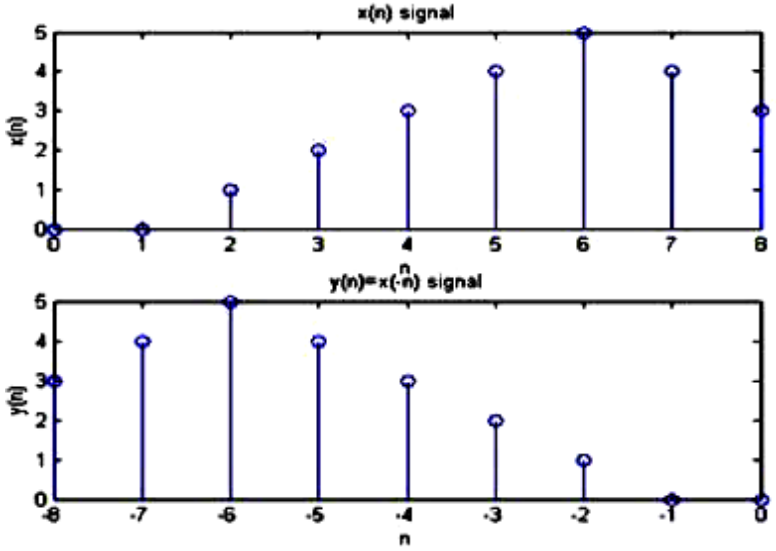
VIVA QUESTIONS:-

1. Define Symetric and Anti-Symmetric Signals?

2. Define Continuous and Discrete Time Signals?
3. What are the Different types of representation of discrete time signals?
4. What are the Different types of Operation performed on signals?
5. What is System?

EXCERSISE:

1. Write a MATLAB program to generate amplitude scaling of a sequence.
2. Write a MATLAB program to subtract to sinusoidal signals.
3. Write a MATLAB program to subtract and multiply to sinusoidal signals.
4. Write a MATLAB program to right shift the signal to 5 times of the original signal.
5. Write a MATLAB program to left shift the signal to 8 times of the original signal.
6. Write a MATLAB program to add to different signals with 2 <t<5
7. Write a MATLAB program to shift a positive time line signal to negative timeline signal.
8. Write a MATLAB program to get the following output.



EXPERIMENTY No-4

AIM: Finding even and odd part of the signal and sequence and also findreal and imaginary parts of signal.

Software Required:
Matlab software 7.0 and above.

Theory:
EVEN AND ODD PART OF A SIGNAL:
Any signal x(t) can be expressed as sum of even and odd components I e
$X(t)=x_e(t)+x_o(t)$

$$x_e(t)=\frac{1}{2}\{x(t)+x(-t)\}, \quad ; \quad x_o(t)=\frac{1}{2}\{x(t)-x(-t)\}$$

$$x(t)=x_e(t)+x_o(t)$$

$$=\frac{1}{2}\{x(t)+x(-t)\}+\frac{1}{2}\{x(t)-x(-t)\}$$

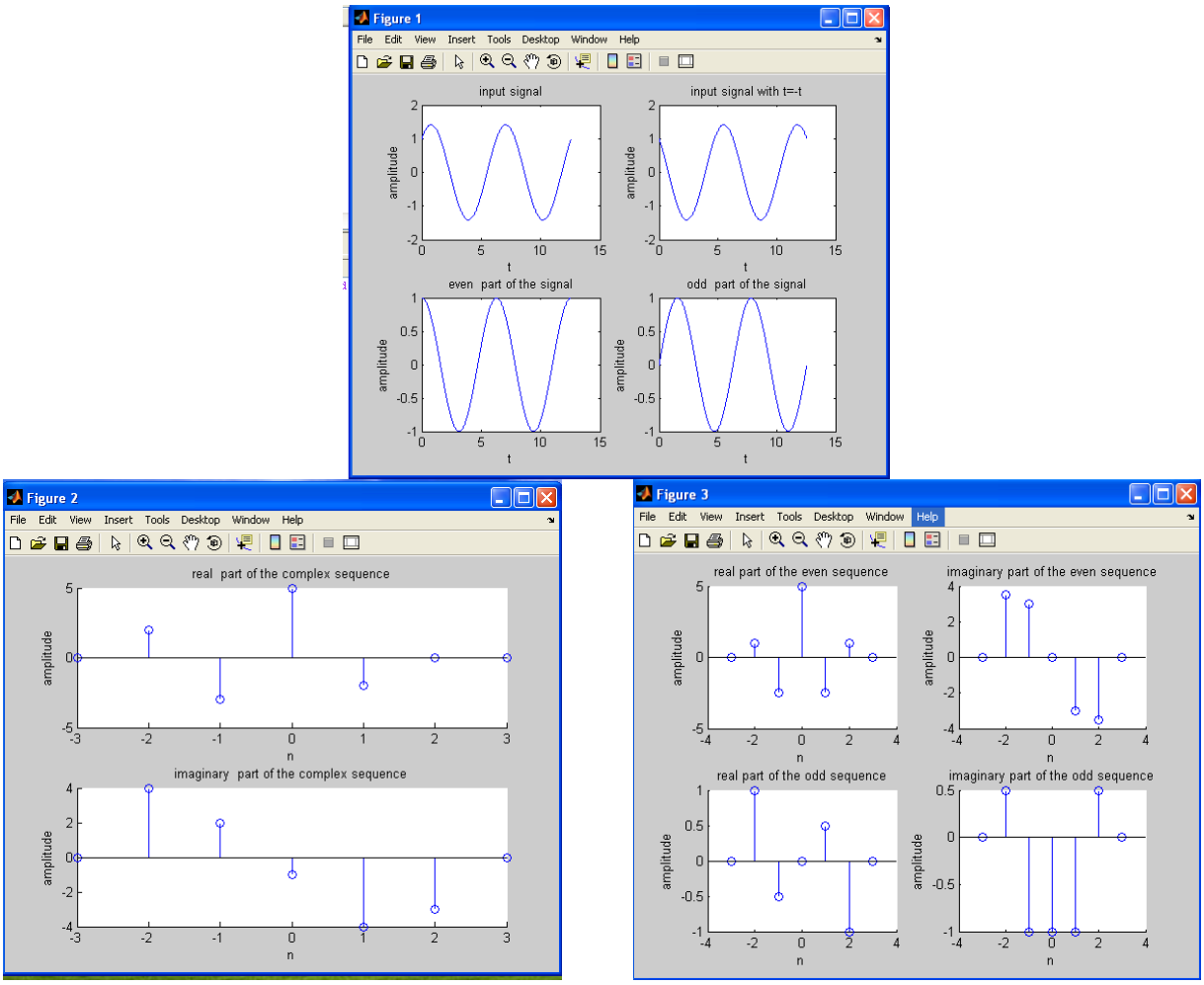
Program:

```
Clc;
close all;
clear all;
% Even and odd parts of a signal
t=0:.005:4*pi;
x=sin(t)+cos(t); % x(t)=sint(t)+cos(t)
subplot(2,2,1)
plot(t,x)
xlabel('t');
ylabel('amplitude')
title('input signal')
y=sin(-t)+cos(-t) % y=x(-t)
subplot(2,2,2)
plot(t,y)
xlabel('t');
ylabel('amplitude')
title('input signal with t=-t')
z=x+y
subplot(2,2,3)
plot(t,z/2)
xlabel('t');
ylabel('amplitude')
title('even part of the signal')%assigning a name to the plot
p=x-y
subplot(2,2,4)
plot(t,p/2)
xlabel('t');
```

```
ylabel('amplitude');
title('odd part of the signal');
% Even and odd parts of a sequence
z=[0,2+j*4,-3+j*2,5-j*1,-2-j*4,-j*3,0];
n=-3:3
% plotting real and imginary parts of the sequence
figure;
subplot( 2,1,1);
stem(n,real(z));
xlabel('n');
ylabel('amplitude');
title('real part of the complex sequence');
subplot( 2,1,2);
stem(n,imag(z));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the complex sequence');
zc=conj(z);
zc_folded= fliplr(zc);
zc_even=.5*(z+zc_folded);
zc_odd=.5*(z-zc_folded);
% plotting even and odd parts of the sequence
figure;
subplot( 2,2,1);
stem(n,real(zc_even));
xlabel('n');
ylabel('amplitude');
title('real part of the even sequence');
subplot( 2,2,2);
stem(n,imag(zc_even));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the even sequence');
subplot( 2,2,3);
stem(n,real(zc_odd));
xlabel('n');
ylabel('amplitude');
title('real part of the odd sequence');
subplot( 2,2,4);
stem(n,imag(zc_odd));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the odd sequence');
```

RESULT: Even and odd part of the signal and sequence is computed.

OUTPUT:



VIVA QUESTIONS:-

1. What is the formula to find odd part of signal?
2. What is Even Signal?
3. What is Odd Signal?
4. What is the formula to find even part of signal?
5. What is the difference b/w stem&plot?

EXERCISE

1. Write a MATLAB program to find even part of a signal by considering 10 input samples.
2. Write a MATLAB program to find odd part of a signal by considering atleast 7 samples.
3. Write a MATLAB program to add even an odd part of a signal and see how matlab reacts for the above program.
4. Write a matlab program to get the out put as [-5, 3, 0, 8] as imaginary values and [2 4 6 8 0] as real values.
5. Write a MATLAB program to subtract even an odd part of a signal and see how matlab reacts for the above program.

EXPERIMENTY No-5

AIM: -
To find the output with linear convolution operation Using MATLAB Software.

SOFTWARE REQURIED:-
1.MATLAB7.2(2006b) / MATLAB 8.6(2015b)/MATLAB 7.6 2008a(Trial version)/MATLAB 7.9(2009b)(Trial Version)/MATLAB 7.10(2010a) Trial version.

2.Windows XP SP2.

THEORY:-
Linear Convolution involves the following operations.
1. Folding
2. Multiplication
3. Addition
4. Shifting
These operations can be represented by a Mathematical Expression as follows:

$$y[n] = \sum_{k=-\infty} x[k]h[n-k]$$

PROCEDURE:-
□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

%**program for convolution of two sequences**

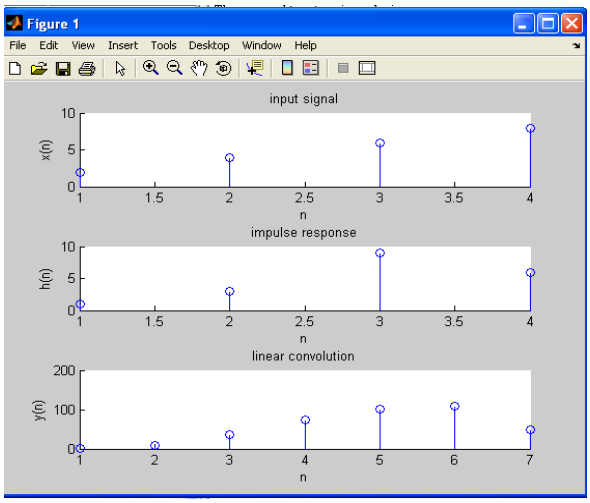
```
clc;
close all;
clear all;
%program for convolution of two sequences
x=input('enter input sequence');
h=input('enter impulse response');
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input signal')
subplot(3,1,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse response')
subplot(3,1,3);
stem(y);
```

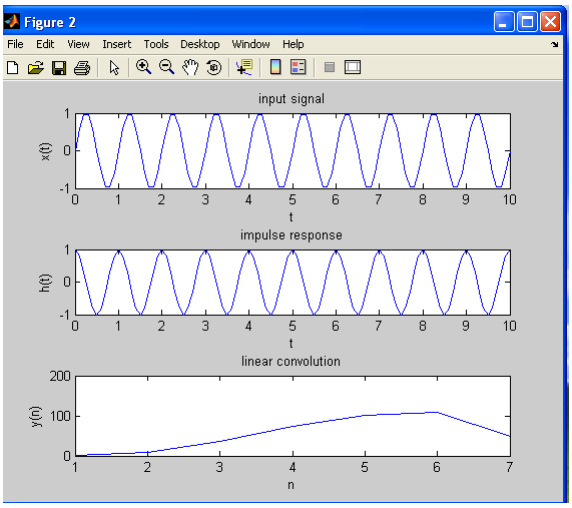
```
xlabel('n');
ylabel('y(n)');
title('linear convolution')
disp('The resultant signal is');
disp(y)
```

%program for signal convolution

```
t=0:0.1:10;
x1=sin(2*pi*t);
h1=cos(2*pi*t);
y1=conv(x1,h1);
figure;
subplot(3,1,1);
plot(t,x1);
xlabel('t');
ylabel('x(t)');
title('input signal')
subplot(3,1,2);
plot(t,h1);
xlabel('t');
ylabel('h(t)');
title('impulse response')
subplot(3,1,3);
plot(y1);
xlabel('n');
ylabel('y(n)');
title('linear convolution');
```

OUTPUT:-





RESULT: convolution between signals and sequences is computed.

Output:

```
enter input sequence[2 4 6 8]
enter impulse response[1 3 9 6]
The resultant signal is
2 10 36 74 102 108 48
```

VIVA QUESTIONS:-

1. Define Convolution?
2. Define Properties of Convolution?
3. What is the Difference Between Convolution& Correlation?
4. What are Dirchlet Condition?
5. What is Half Wave Symmetry?

EXERCISE:

1. Write the MATLAB program to perform convolution between the following sequences X(n)=[1 -1 4], h(n) = [-1 2 -3 1].
2. Write a mat lab program to perform the convolution between sinusoidal and ramp function and see how mat lab reacts to it.
3. Write a MATLAB program to perform convolution between square and step signal and see how mat lab reacts to it.
4. Write a MATLAB program to perform convolution between sinusoidal and ramp signal and see how mat lab reacts to it.
5. Write a MATLAB program to perform the convolution between X (n) = [1 2 3 5] and y (n) = [- 1 – 2] and see how matlab reacts to it.
6. Write a MATLAB program to perform the convolution between X (n) = [1 -3 5] and y (n) = [1 2 3 4] and see how matlab reacts to it.

EXPERIMENT NO-6

AIM: -
To compute auto correlation and cross correlation between signals and Sequences.

Software Required:
MATLAB software 7.0 and above

Theory:
Correlations of sequences:
It is a measure of the degree to which two sequences are similar. Giventwo real-valued sequences $x(n)$ and $y(n)$ of finite energy,
These operations can be represented by a Mathematical Expression asfollows:

Cross correlation

$$r_{x,y}(l) = \sum_{n=-\infty}^{+\infty} x(n)y(n-l)$$

The index l is called the shift or lag parameter

Autocorrelation

$$r_{x,x}(l) = \sum_{n=-\infty}^{+\infty} x(n)x(n-l)$$

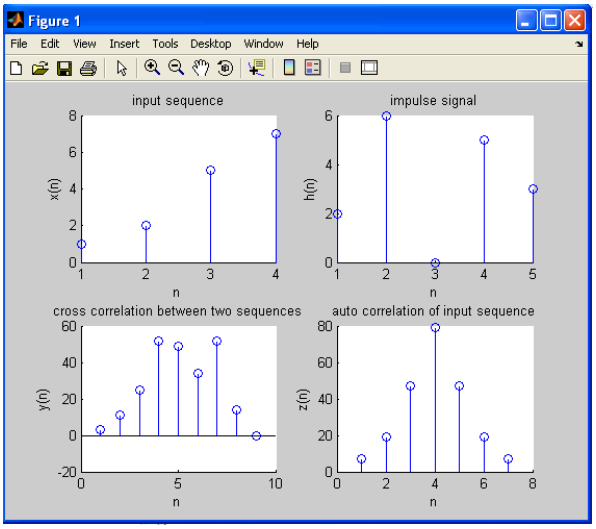
Program:

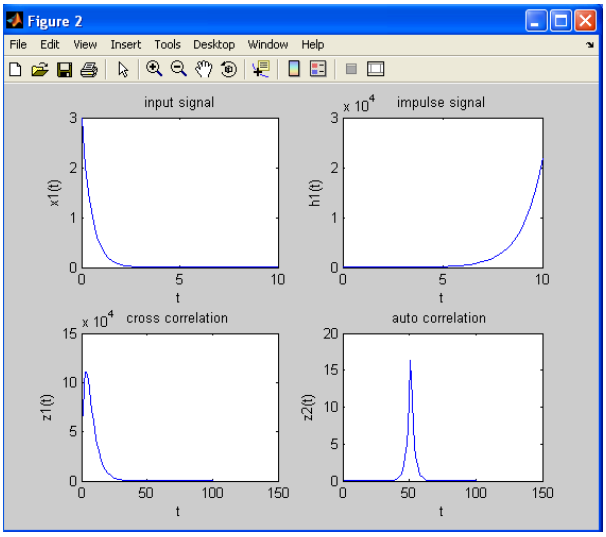
```
clc;
close all;
clear all;
% two input sequences
x=input('enter input sequence');
h=input('enter the impulse suquence');
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');
subplot(2,2,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse signal');
% cross correlation between two sequences
y=xcorr(x,h);
subplot(2,2,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title(' cross correlation between two sequences ');
```

```
% auto correlation of input sequence
z=xcorr(x,x);
subplot(2,2,4);
stem(z);
xlabel('n');
ylabel('z(n)');
title('auto correlation of input sequence');

% cross correlation between two signals
% generating two input signals
t=0:0.2:10;
x1=3*exp(-2*t);
h1=exp(t);
figure;
subplot(2,2,1);
plot(t,x1);
xlabel('t');
ylabel('x1(t)');
title('input signal');
subplot(2,2,2);
plot(t,h1);
xlabel('t');
ylabel('h1(t)');
title('impulse signal');
% cross correlation
subplot(2,2,3);
z1=xcorr(x1,h1);
plot(z1);
xlabel('t');
ylabel('z1(t)');
title('cross correlation ');
% auto correlation
subplot(2,2,4);
z2=xcorr(x1,x1);
plot(z2);
xlabel('t');
ylabel('z2(t)');
title('auto correlation ');
```

Result: Auto correlation and Cross correlation between signals andsequences is computed.

Output: enter input sequence [1 2 5 7]
Enter the impulse sequence [2 6 0 5 3]





VIVA QUESTIONS:-

1. Define Correlation?
2. Define Auto-Correlation?
3. Define Cross-Correlation?
4. What is the importance of correlation?
5. What is the difference b/w correlation and convolution?

EXERCISE

1. Write a MATLAB program to compute auto correlation between signals and Sequences. x=cos(2*pi*10*t),y=cos(2*pi*15*t).
2. Write a MATLAB program to compute cross correlation between signals and Sequences. x=cos(2*pi*7*t),y=cos(2*pi*14*t).
3. Write a MATLAB program to compute the cross correleation between signals and Sequences. x=cos(2*pi*10*t),y=cos(2*pi*15*t) by increasing the amplitude of the signal by 3 times and verify how matlab reacts to it.
4. Write a MATLAB program to compute the auto correleation between signals and Sequences. x=cos(2*pi*15*t),y=cos(2*pi*10*t) by increasing the amplitude of the signal by 2 times and verify how matlab reacts to it.
5. Write a MATLAB program to compute auto correlation between x =sin(2*pi*5*t) , y = sin(2*pi*10*t). and see how matlab reacts to it.
6. Write a MATLAB program to compute cross correlation between x =sin(2*pi*5*t) , y =cos(2*pi*10*t). and see how matlab reacts to it.

EXPERIMENT No-7(a)

AIM: Verify the Linearity of a given Discrete System.

Software Required:
Mat lab software 7.0 and above

Theory:
LINEARITY PROPERTY:
Any system is said to be linear if it satisfies the superposition principal.superposition principal state that Response to a weighted sum of input signalequal to the corresponding weighted sum of the outputs of the system toeach of the individual input signals.

$$T[a_1 x_1(n) + a_2 x_2(n)] = a_1 T[x_1(n)] + a_2 T[x_2(n)]$$

X(n)----------input signal
Y(n) --------- output signal

Y(n)=T[x(n)]

Y1(n)=T[X1(n)] : Y2(n)=T[X2(n)]

x3=[a X1(n)] +b [X2(n)]

Y3(n)= T [x3(n)]

= T [a X1(n)] +b [X2(n)] = a Y1(n)+ b [Y

Let a [Y1(n)]+ b [X2(n)] =Z 3(n)

Program:
```
clc;
clear all;
close all;
% entering two input sequences and impulse sequence
x1 = input (' type the samples of x1 ');
x2 = input (' type the samples of x2 ');
if(length(x1)~=length(x2))
disp('error: Lengths of x1 and x2 are different');
return;
end;
h = input (' type the samples of h ');
% length of output sequence
N = length(x1) + length(h) -1;
disp('length of the output signal will be ');
disp(N);
% entering scaling factors
a1 = input (' The scale factor a1 is ');
a2 = input (' The scale factor a2 is ');
x = a1 * x1 + a2 * x2;
% response of x and x1
yo1 = conv(x,h);
y1 = conv(x1,h);
% scaled response of x1
y1s = a1 * y1;
% response of x2
y2 = conv(x2,h);
```

```
% scaled response of x2
y2s = a2 * y2;
yo2 = y1s + y2s;
disp ('Input signal x1 is '); disp(x1);
disp ('Input signal x2 is '); disp(x2);
disp ('Output Sequence yo1 is '); disp(yo1);
disp ('Output Sequence yo2 is '); disp(yo2);
/if ( yo1 == yo2 )
disp(' yo1 = yo2. Hence the LTI system is LINEAR ')
end;
```

Result: The Linearity of a given Discrete System is verified.

Output:

```
Type the samples of x1 [1 5 6 7]
Type the samples of x2 [2 3 4 8]
Type the samples of h [2 6 5 4]
Length of the output signal will be
7
The scale factor a1 is 2
The scale factor a2 is 3
Input signal x1 is
1 5 6 7
Input signal x2 is
2 3 4 8
Output Sequence yo1 is
16 86 202 347 424 286 152
Output Sequence yo2 is
16 86 202 347 424 286 152
yo1 = yo2. Hence the LTI system is LINEAR
```

EXPERIMENT No-7(b)

AIM: Verify the Time Invariance of a given Discrete System.

Software Required:
Mat lab software 7.0 and above

Theory:
TIME INVARIENT SYSTEMS(TI):
A system is called time invariant if its input – output characteristics do notchange with time
X(t)---- input : Y(t) ---output
X(t-T) -----delay input by T seconds : Y(t-T) ------ Delayed output by Tseconds

Program:

```
clc;
clear all;
close all;
% entering two input sequences
x = input( ' Type the samples of signal x(n) ' );
h = input( ' Type the samples of signal h(n) ' );
% original response
y = conv(x,h);
disp( ' Enter a POSITIVE number for delay ' );
d = input( ' Desired delay of the signal is ' );
% delayed input
xd = [zeros(1,d), x];
nxd = 0 : length(xd)-1;
%delayed output
yd = conv(xd,h);
nyd = 0:length(yd)-1;
disp(' Original Input Signal x(n) is ');
disp(x);
disp(' Delayed Input Signal xd(n) is ');
disp(xd);
disp(' Original Output Signal y(n) is ');
disp(y);
disp(' Delayed Output Signal yd(n) is ');
disp(yd);
xp = [x , zeros(1,d)];
subplot(2,1,1);
stem(nxd,xp);
grid;
xlabel( ' Time Index n ' );
ylabel( ' x(n) ' );
title( ' Original Input Signal x(n) ' );
subplot(2,1,2);
stem(nxd,xd);
grid;
xlabel( ' Time Index n ' );
ylabel( ' xd(n) ' );
title( ' Delayed Input Signal xd(n) ' );
yp = [y zeros(1,d)];
```

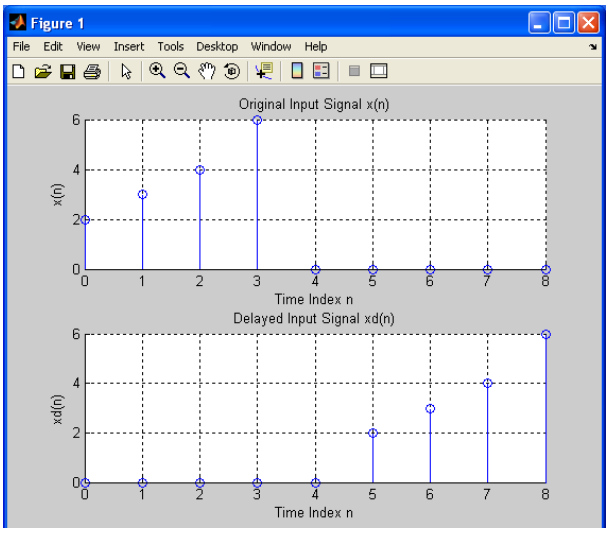
```
figure;
subplot(2,1,1);
stem(nyd,yp);
grid;
xlabel( ' Time Index n ' );
ylabel( ' y(n) ' );
title( ' Original Output Signal y(n) ' );
subplot(2,1,2);
stem(nyd,yd);
grid;
xlabel( ' Time Index n ' );
ylabel( ' yd(n) ' );
title( ' Delayed Output Signal yd(n) ' );
```

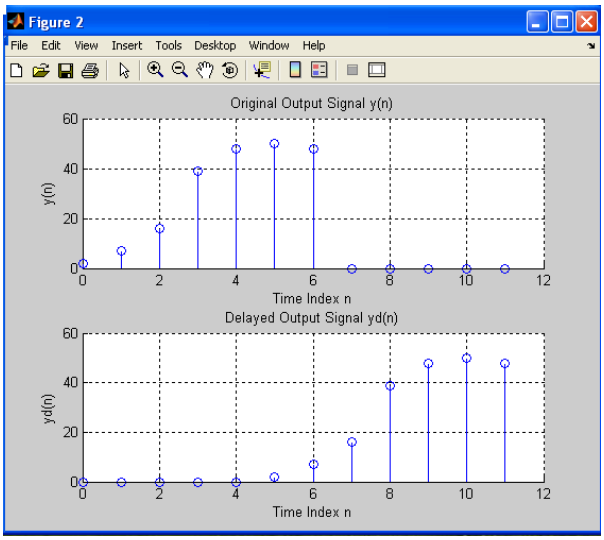
Result:

The Time Invariance of a given Discrete System is verified.

Output:

```
Type the samples of signal x(n) [2 3 4 6]
Type the samples of signal h(n) [1 2 3 8]
Enter a POSITIVE number for delay
Desired delay of the signal is 5
Original Input Signal x(n) is
2 3 4 6
Delayed Input Signal xd(n) is
0 0 0 0 0 2 3 4 6
Original Output Signal y(n) is
2 7 16 39 48 50 48
Delayed Output Signal yd(n) is
0 0 0 0 0 2 7 16 39 48 50 48
```





EXSERCISE

1. Write a MATLAB program to verify the linearity prpoerty of the following sequencyx1= sin(2*pi*1*n); x2= sin(2*pi*2*n), and chech whether it satisfies the linearity property or not.

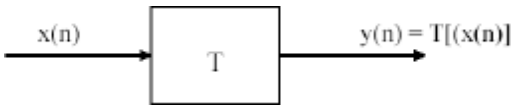
2. Write a MATLAB program to verify the linearity prpoerty of the following sequency x1= sin(2*pi*1*n); x2= sin(2*pi*2*n), and chech whether it satisfies the linearity property or not
3. Write a MATLAB program to verify the linearity prpoerty of the following sequency x1= sin(2*pi*0.1*n); cos(2*pi*0.3*n), and chech whether it satisfies the linearity property or not
4. Write a MATLAB program to verify the time invariance prpoerty of the following sequency x1= sin(2*pi*1*n); x2= sin(2*pi*2*n), and chech whether it satisfies the time invariance property or not.
5. Write a MATLAB program to verify the time invariance prpoerty of the following sequency x1= sin(2*pi*1*n); x2= sin(2*pi*2*n), and chech whether it satisfies the time invariance property or not
6. Write a MATLAB program to verify the time invariance prpoerty of the following sequency x1= sin(2*pi*0.1*n); cos(2*pi*0.3*n), and chech whether it satisfies the time invariance property or not

EXPERIMENT NO-8

AIM: Compute the Unit sample, unit step and sinusoidal response of thegiven LTI system and verifying its stability

Software Required:
Mat lab software 7.0 and above

Theory:
A discrete time system performs an operation on an input signal based onpredefined criteria to produce a modified output signal. The input signal x(n)is the system excitation, and y(n) is the system response. The transform\ operation is shown as,



If the input to the system is unit impulse i.e. x(n) = δ(n) then the output ofthe system is known as impulse response denoted by h(n) where,h(n) = T[δ(n)]we know that any arbitrary sequence x(n) can be represented as a weightedsum of discrete impulses. Now the system response is given by,

$$y(n) = T[x(n)] = T\left[\sum_{k=-\infty}^{\infty} x(k)\,\delta(n-k)\right]$$

For linear system (1) reduces to

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\,T[\delta(n-k)]$$

given difference equation y(n)-y(n-1)+.9y(n-2)=x(n);

$$H(Z) = \frac{\sum_{k=0}^{M} b_k X(n-k)}{\sum_{k=1}^{N} a_k X(n-k)}$$

$$H(z) = \frac{b_0+b_1Z^{-1}+b_2Z^{-2}+\ldots\ldots\ldots\ldots b_{N-1}Z^{(N-1)}+b_NZ^{-N}}{1+a_1Z^{-1}+a_2Z^{-2}+\ldots\ldots\ldots\ldots a_{N-1}Z^{(N-1)}+a_NZ^{-N}}$$

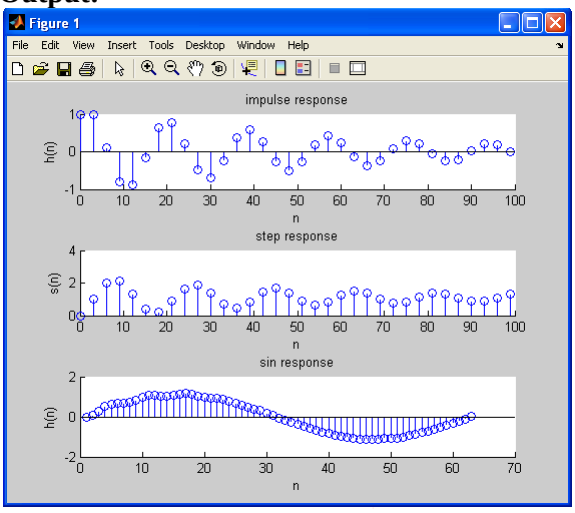
Program:

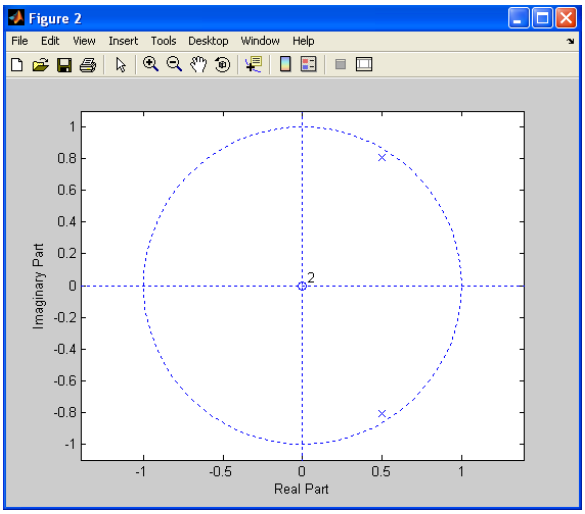
```
%given difference equation y(n)-y(n-1)+.9y(n-2)=x(n);
b=[1];
a=[1,-1,.9];
n =0:3:100;
%generating impulse signal
x1=(n==0);
%impulse response
```

```
h1=filter(b,a,x1);
subplot(3,1,1);
stem(n,h1);
xlabel('n');
ylabel('h(n)');
title('impulse response');
%generating step signal
x2=(n>0);
% step response
s=filter(b,a,x2);
subplot(3,1,2);
stem(n,s);
xlabel('n');
ylabel('s(n)')
title('step response');

%generating sinusoidal signal
t=0:0.1:2*pi;
x3=sin(t);
% sinusoidal response
h2=filter(b,a,x3);
subplot(3,1,3);
stem(h2);
xlabel('n');
ylabel('h(n)');
title('sin response');
% verifing stability
figure;
zplane(b,a);
```

Result: The Unit sample, unit step and sinusoidal response of the givenLTI system is computed and its stability verified.Hence all the poles lie inside the unit circle, so system is stable.

Output:





VIVA QUESTIONS:-

1.What operations can be performed on signals and sequence?
2.Define causality and stability?
3.Define scaling property and give its importance?
4. Define shifting property and give its importance?
5.Define folding property and give its importance?

EXERCISE PROGRAM:-

1.Write a MATLAB program for generating u(n)-u(n-1).
2. Write a MATLAB program for generating delayed unit step response
3. Write a MATLAB program for generating delayed impulse response
4. Write a MATLAB program for generating u(n)+u(n-1) and verify how matlab reacts to it.

EXPERIMENT NO-09

AIM: -
To obtain Fourier Transform and Inverse Fourier Transform of a given signal / sequence and to plot its Magnitude and Phase Spectra.

SOFTWARE REQURIED:-
1.MATLAB R2010a.
2.Windows XP SP2.

THEORY:-
Fourier Transform :
The Fourier transform as follows. Suppose that f is a function which is zerooutside of some interval $[-L/2, L/2]$. Then for any $T \geq L$ we may expand f in\ a Fourier series on the interval $[-T/2, T/2]$, where the "amount" of the wave$e^{2\pi inx/T}$ in the Fourier series of f is given by definition Fourier Transform of signal f(x) is defined as

$$F(\omega) = \int_{-\infty}^{\infty} f(t).\, e^{-j\omega t}\, .\, dt$$

Inverse Fourier Transform of signal F(w) is defined as

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) e^{j\omega t}\; d\omega$$

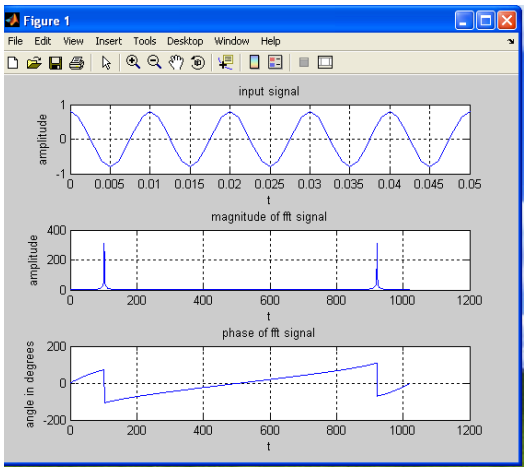
Program:

```
clc;
clear all;
close all;
fs=1000;
N=1024; % length of fft sequence
t=[0:N-1]*(1/fs);
% input signal
x=0.8*cos(2*pi*100*t);
subplot(3,1,1);
plot(t,x);
axis([0 0.05 -1 1]);
grid;
xlabel('t');
ylabel('amplitude');
title('input signal');
% magnitude spectrum
x1=fft(x);
k=0:N-1;
Xmag=abs(x1);
subplot(3,1,2);
plot(k,Xmag);
grid;
```

```
xlabel('t');
ylabel('amplitude');
title('magnitude of fft signal')
%phase spectrum
Xphase=angle(x1)*(180/pi);
subplot(3,1,3);
plot(k,Xphase);
grid;
xlabel('t');
ylabel('angle in degrees');
title('phase of fft signal');
```

Result: Magnitude and phase spectrum of FFT of a given signal isplotted.

Output:



VIVA QUESTIONS:-

1. Define Fourier Series?
2. What are the properties of Continuous-Time Fourier Series?
3. What is the Sufficient condition for the existence of F.T?
4. Define the F.T of a signal?
5. What is the difference b/w F.T&F.S?

EXERCISE PROGRAMS

1. Write a MATLAB program to find the correlation using FFT.

.

EXPERIMENT-10

AIM: Write the program for locating poles and zeros and plotting pole-zero maps in s-plane and z-plane for the given transfer function.

Software Required:
Matlab software 7.0 and above.

Theory:

Z-transforms

The Z-transform, like many other integral transforms, can be defined aseither a *one-sided* or *two-sided* transform.Bilateral Z-transform. The *bilateral* or *two-sided* Z-transform of a discrete-time signal *x[n]* is thefunction *X(z)* defined as

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Unilateral Z-transform

Alternatively, in cases where *x*[*n*] is define

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n}$$

In signal processing, this definition is used when the signal is causal.

$$\text{where } z = r.e^{j\omega}$$

$$X(z) = \frac{P(z)}{Q(z)}$$

The roots of the equation P(z) = 0 correspond to the 'zeros' of X(z)
The roots of the equation Q(z) = 0 correspond to the 'poles' of X(z)

Program:

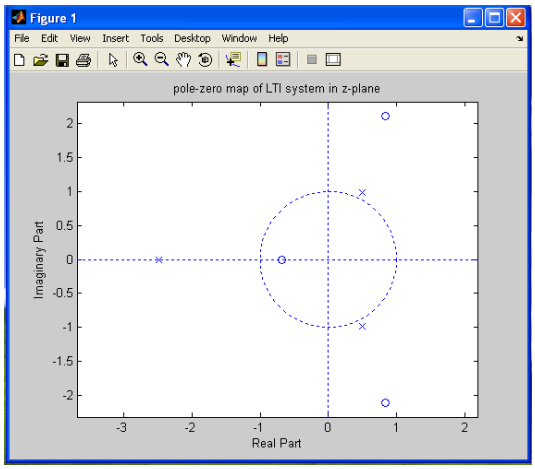
```
clc;
clear all;
close all;
%enter the numerator and denominatorcoefficients in square brackets
num=input('enter the numerator coefficients');
den=input('enter the denominatorcoefficients');
%find the transfer function using built-in function 'filt'
H=filt(num,den)
%find locations of zeros
z=zero(H);
disp('zeros are at ');
disp(z);
%find residues,pole locations and gain constant of H(z)
[r p k]=residuez(num,den);
disp('poles are at ');
```

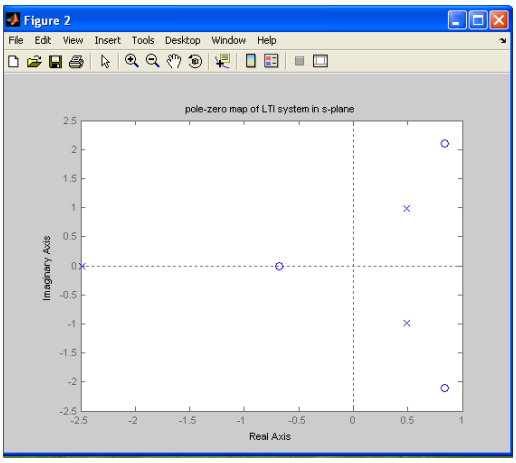
```
disp(p);
%plot the pole zero map in z-plane
zplane(num,den);
title('pole-zero map of LTI system in z-plane');
% ploe-zero plot in s-plane
H1=tf(num,den) % find transfer function H(s)
[p1,z1]=pzmap(H1); % find the locations of poles and zeros
disp('poles ar at ');disp(p1);
disp('zeros ar at ');disp(z1);
figure;
%plot the pole-zero map in s-plane
pzmap(H1);
title('pole-zero map of LTI system in s-plane');
```

Result: Pole-zero maps are plotted in s-plane and z-plane for the given transfer function.

Output:

```
enter the numerator coefficients[1 -1 4 3.5]
enter the denominatorcoefficients[2 3 -2.5 6]
Transfer function:
1 - z^-1 + 4 z^-2 + 3.5 z^-3
-----------------------------
2 + 3 z^-1 - 2.5 z^-2 + 6 z^-3
zeros are at
0.8402 + 2.1065i
0.8402 - 2.1065i
-0.6805
poles are at
-2.4874
0.4937 + 0.9810i
0.4937 - 0.9810i
Transfer function:
s^3 - s^2 + 4 s + 3.5
---------------------
2 s^3 + 3 s^2 - 2.5 s + 6
polesar at
-2.4874
0.4937 + 0.9810i
0.4937 - 0.9810i
zerosar  at
0.8402 + 2.1065i
0.8402 - 2.1065i
-0.6805
```





VIVA QUESTIONS:-

1.Study the details of ztrans() and iztrans() functions?
2.What are poles and zeros?
3.How you specify the stability based on poles and zeros?
4.Define S-plane and Z-plane?
5.What is the difference b/w S-plane and Z-plane?

EXERCISE

1.Write a MATLAB program to find the impulse response of the following difference equation 3y(n)-5y(n-1)+4y(n-2)=x(n)-2x(n-1).

EXPERIMENT No-11

AIM: Verify the sampling theorem.

Software Required:

Matlab software 7.0 and above.

Theory:

Sampling Theorem:

A bandlimited signal can be reconstructed exactly if it is sampled at a rateat least twice the maximum frequency component in it." Figure 1 shows asignal g(t) that is bandlimited.

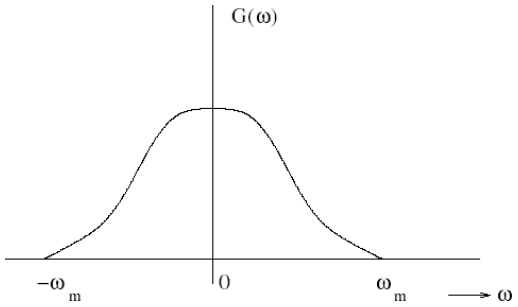


Figure 1: Spectrum of bandlimited signal g(t)

The maximum frequency component of g(t) is fm. To recover the signal g(t)exactly from its samples it has to be sampled at a rate fs ≥ 2fm.The minimum required sampling rate fs = 2fm is called ' Nyquist rate

Proof: Let g(t) be a band-limited signal whose bandwidth is fm(wm = 2πfm).

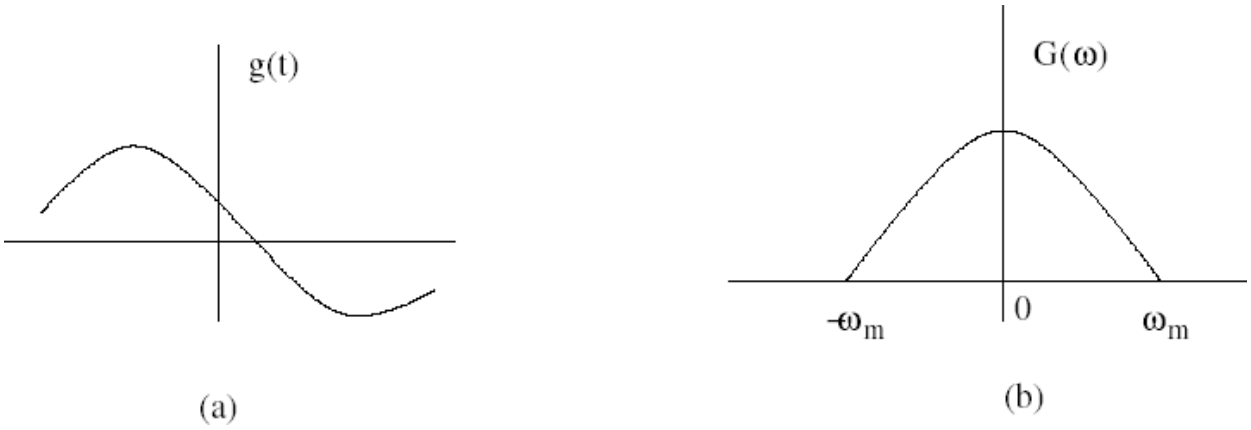


Figure 2: (a) Original signal g(t) (b) Spectrum G(w)

δ (t) is the sampling signal with fs = 1/T > 2fm.

Let $g_s(t)$ be the sampled signal. Its Fourier Transform Gs(w) is given by

$$\begin{aligned}
\mathcal{F}(g_s(t)) &= \mathcal{F}\left[g(t)\delta_T(t)\right] \\
&= \mathcal{F}\left[g(t)\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right] \\
&= \frac{1}{2\pi}\left[G(\omega) * \omega_0 \sum_{n=-\infty}^{+\infty}\delta(\omega - n\omega_0)\right] \\
G_s(\omega) &= \frac{1}{T}\sum_{n=-\infty}^{+\infty} G(\omega) * \delta(\omega - n\omega_0) \\
G_s(\omega) &= \mathcal{F}\left[g(t) + 2g(t)\cos(\omega_0 t) + 2g(t)\cos(2\omega_0 t) + \cdots\right] \\
G_s(\omega) &= \frac{1}{T}\sum_{n=-\infty}^{+\infty} G(\omega - n\omega_0)
\end{aligned}$$

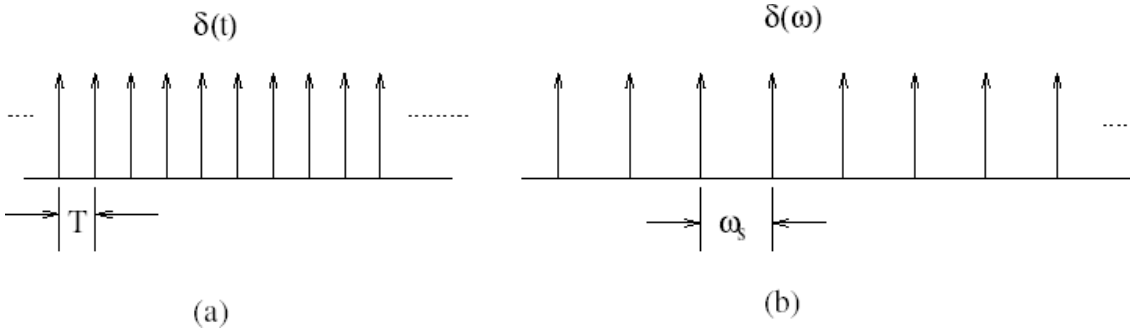


Figure 3: (a) sampling signal δ (t)) (b) Spectrum δ (w)

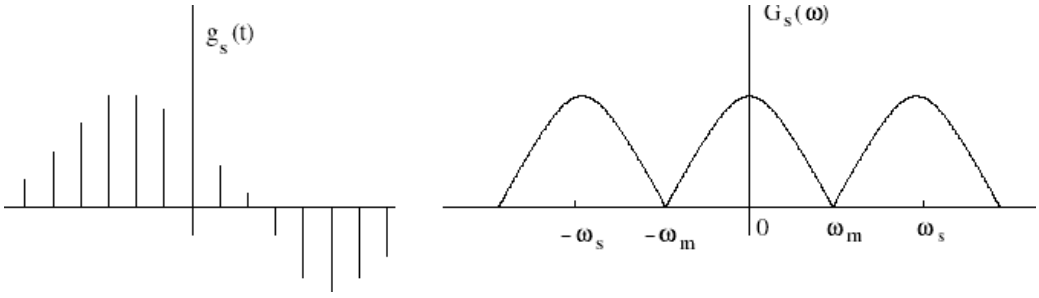


Figure 4: (a) sampled signal gs(t) (b) Spectrum Gs(w)

If $\omega_s = 2\omega_m$, i.e., $T = 1/2f_m$. Therefore, $G_s(\omega)$ is given by

$$G_s(\omega) = \frac{1}{T} \sum_{n=-\infty}^{+\infty} G(\omega - n\omega_m)$$

To recover the original signal G(w):

1. **Filter with a Gate function, $H_{2wm}(w)$ of width $2w_m$ Scale it by T.**

$$G(\omega) = TG_s(\omega)H_{2\omega_m}(\omega).$$

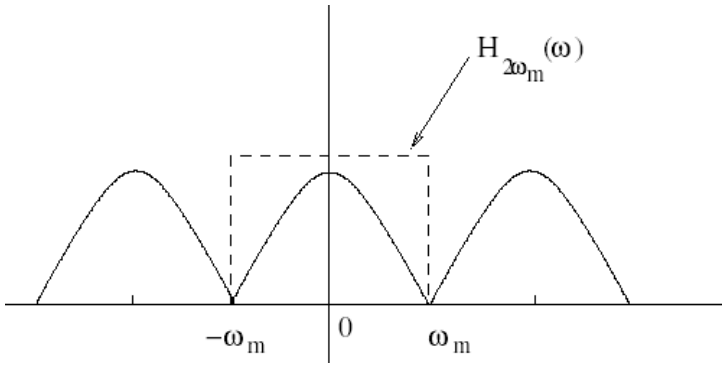


Figure 5: Recovery of signal by filtering with a filter of width $2\omega_m$

Aliasing $\omega_s < 2\omega_m$

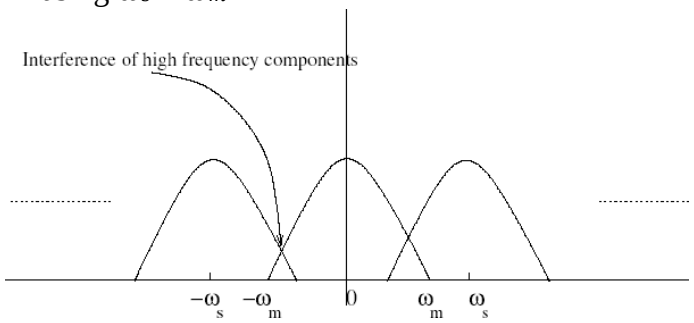


Figure 6: Aliasing due to inadequate sampling

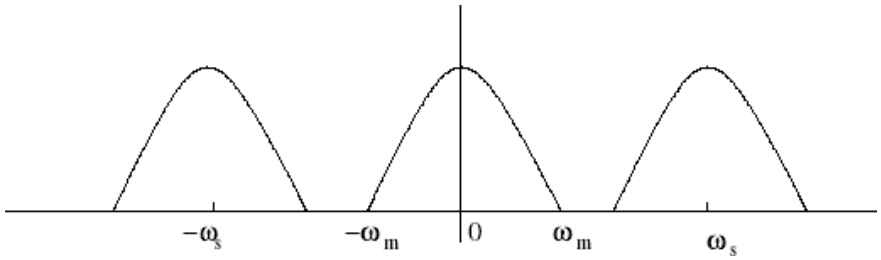


Figure 7: Oversampled signal-avoids aliasing

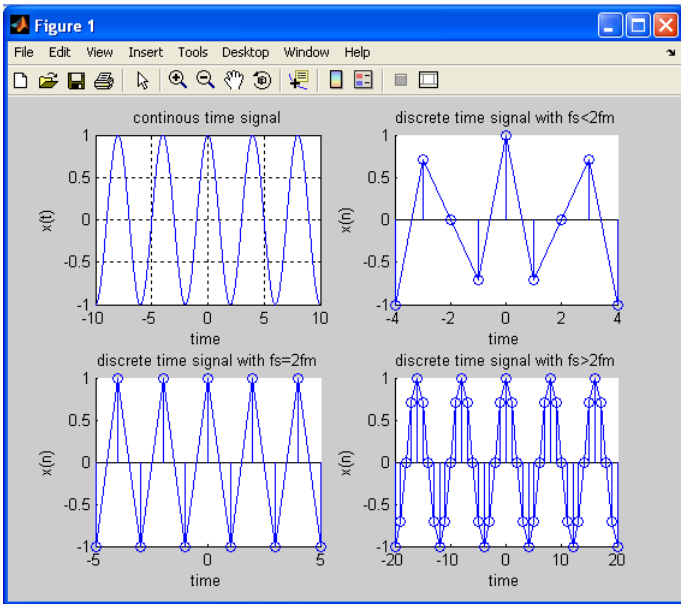
Program:

```
clc;
clear all;
close all;
t=-10:.01:10;
T=4;
fm=1/T;
x=cos(2*pi*fm*t);
subplot(2,2,1);
plot(t,x);
xlabel('time');
ylabel('x(t)');
title('continous time signal');
grid;
n1=-4:1:4;
fs1=1.6*fm;
fs2=2*fm;
fs3=8*fm;
x1=cos(2*pi*fm/fs1*n1);
subplot(2,2,2);
stem(n1,x1);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs<2fm');
hold on;
subplot(2,2,2);
plot(n1,x1);
grid;
n2=-5:1:5;
x2=cos(2*pi*fm/fs2*n2);
subplot(2,2,3);
stem(n2,x2);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs=2fm');
hold on;
subplot(2,2,3);
plot(n2,x2)
grid;
n3=-20:1:20;
x3=cos(2*pi*fm/fs3*n3);
subplot(2,2,4);
stem(n3,x3);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs>2fm')
```

```
hold on;
subplot(2,2,4);
plot(n3,x3)
grid;
```

Result: Sampling theorem is verified.

OUTPUT:



VIVA QUESTIONS:-

1. State Paeseval"s energy theorem for a periodic signal?
2. Define sampling Theorem?
3. What is Aliasing Effect?
4. what is Under sampling?
5. What is Over sampling?

EXERCISE PROGRAM:-

1. Write a MATLAB program to find the effect of up sampling in frequency domain.

EXPERIMENT-1

AIM : To develop a VHDL Code for Logic Gates-AND, OR, NOT, NAND, NOR, XOR, XNOR and to verify its functionality.

APPARATUS:Model Sim 5.7

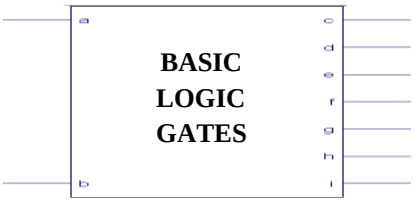
.**VHDL CODE:**

```
library ieee;
use ieee.std_logic_1164.all;
entity logicgates is
port(a,b: in std_logic;c,d,e,f,g,h,i: out std_logic);
end logicgates;
architecture dataflow of logicgates is
begin
c<= a and b;
d<= a or b;
e<= not b;
f<= a xor b;
g<= a nand b;
h<= not(a xor b);
i<= a nor b;
end dataflow;
```

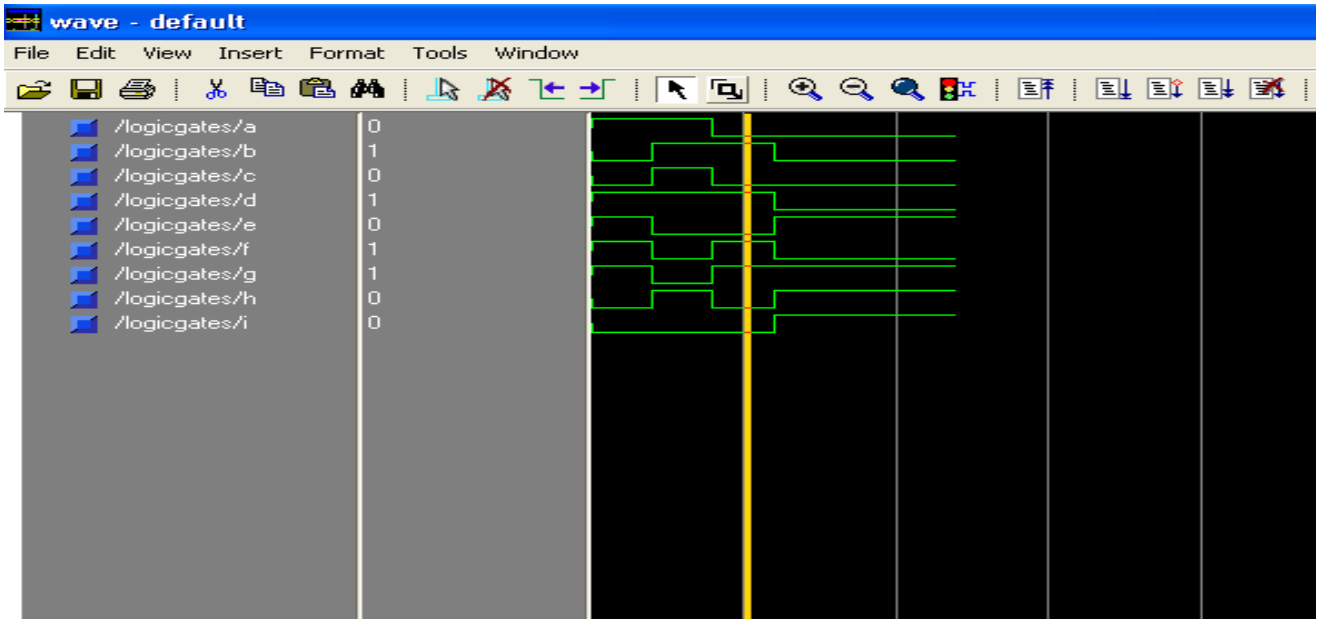
TRUTH TABLE:-

INPUTS		OUTPUTS						
A	**b**	**AND c**	**OR d**	**NOT e**	**XOR f**	**NAND g**	**XNOR h**	**NOR i**
0	0	0	0	1	0	1	1	1
0	1	0	1	0	1	1	0	0
1	0	0	1	1	1	1	0	0
1	1	1	1	0	0	0	1	0

RTL SCHEMATIC:



WAVEFORMS:



RESULT: - Hence all the logic gates are simulated in VHDL using dataflow modeling and their functionality is verified.

VIVA QUESTIONS:

1. What is VHDL?
2. What is the need for VHDL?
3. What is meant by simulation?
4. What is meant by synthesis?
5. Who initialized the VHDL and in which year?

EXPERIMENT-2(a)

AIM : (a) To write a Program in VHDL for simulating the half adder and to verify its functionality**.**

APPARATUS: Model Sim 5.7

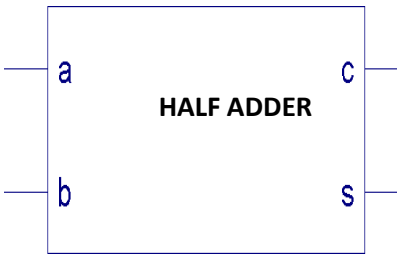
VHDL CODE:

```
library ieee;
use ieee.std_logic_1164.all;
entity halfadder is
port(a,b: in std_logic; s,c: out std_logic);
end halfadder;
architecture dataflow of halfadder is
begin
s<= a xor b;
c<= a and b;
end dataflow;
```

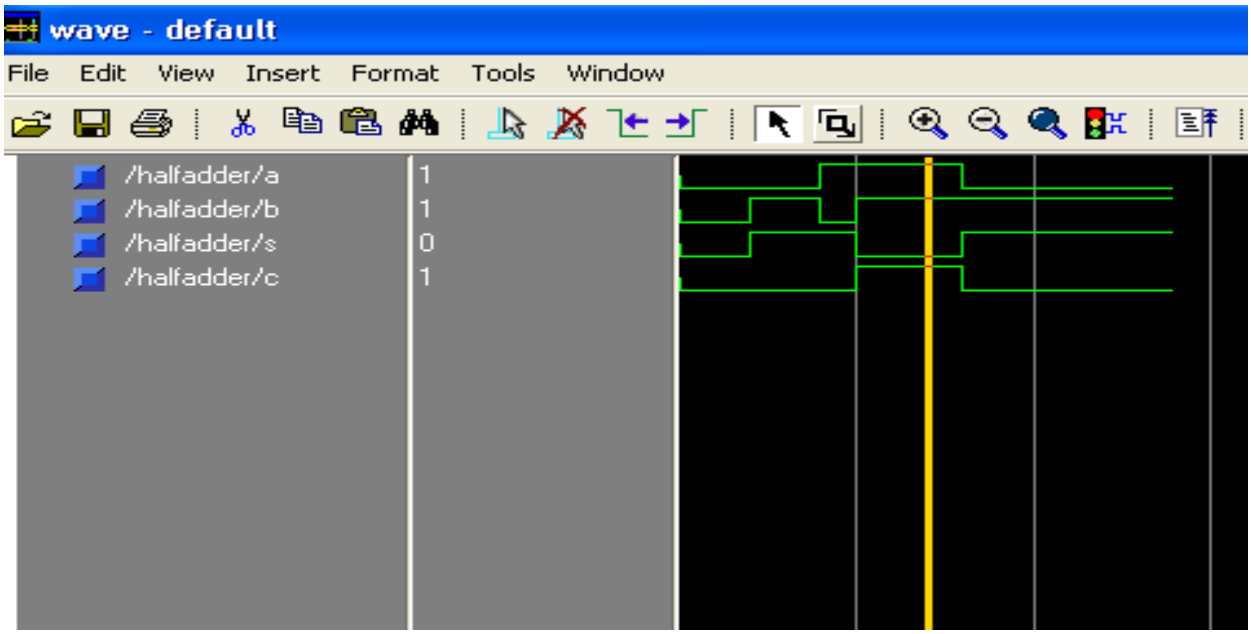
TRUTH TABLE:-

INPUTS		**OUTPUTS**	
a	**b**	**Sum s**	**Carry c**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

RTL SCHEMATIC:



WAVEFORMS:



RESULT:- Hence the half adder is simulated in VHDL using data flow modeling and its functionality is verified .

EXPERIMENT-2(b)

AIM: - To write a PROGRAM in VHDL for simulating the full adder and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:

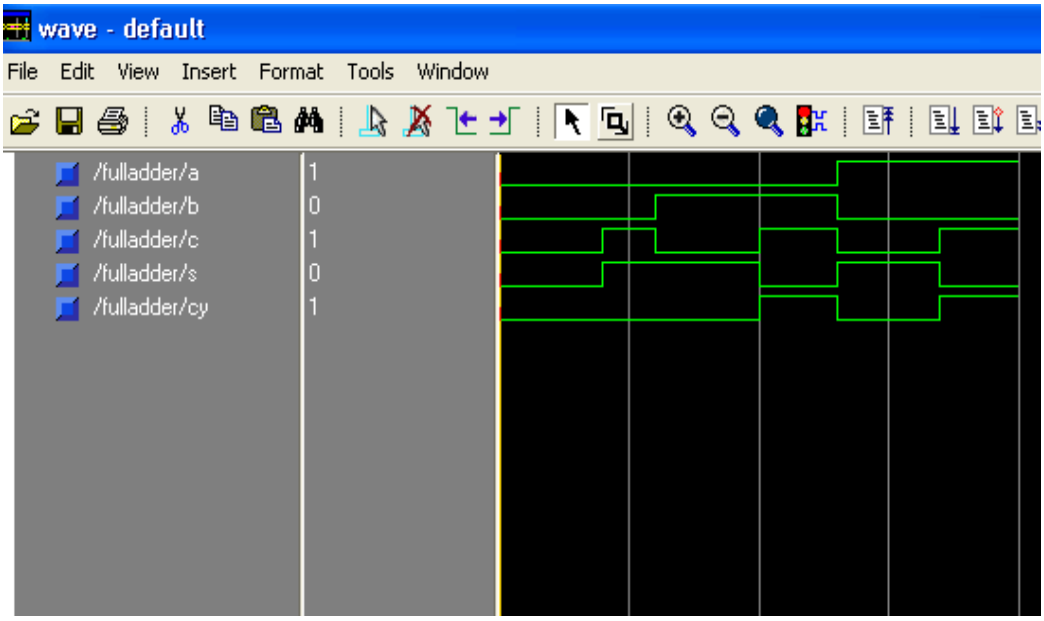
```
library ieee;
use ieee.std_logic_1164.all;
entity fulladder is
port(a,b,c: in std_logic;s,cy: out std_logic);
 end fulladder;
architecture dataflow of fulladder is
begin
s<= (a xor b)xor c;
cy<= (a and b) or (b and c) or (c and a);
end dataflow;
```

TRUTH TABLE:-

INPUTS			**OUTPUTS**	
a	**b**	**c**	**s**	**cy**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RTL SCHEMATIC:

WAVEFORMS:



RESULT: - Hence the full adder is simulated in VHDL and its functionality is verified.

VIVA QUESTIONS:

1. What is the assignment operator for i) signal ii) variable?
2. What is the difference between signal and variable?
3. Is process used for combinational or sequential logic?
4. What is the difference between function and procedure?
5. Define i) entity ii) architecture.

EXPERIMENT-3(a)

AIM : (a) To write a Program in VHDL for simulating the half substractor and to verify its functionality**.**

APPARATUS: Model Sim 5.7

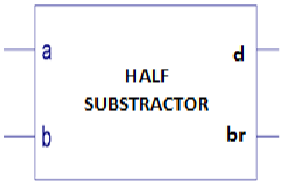
VHDL CODE:

```
library ieee;
use ieee.std_logic_1164.all;
entity halfadder is
port(a,b: in std_logic; d,br: out std_logic);
end halfadder;
architecture dataflow of halfadder is
begin
d<= a xor b;
br<=nor(a) and b;
end dataflow;
```

TRUTH TABLE:-

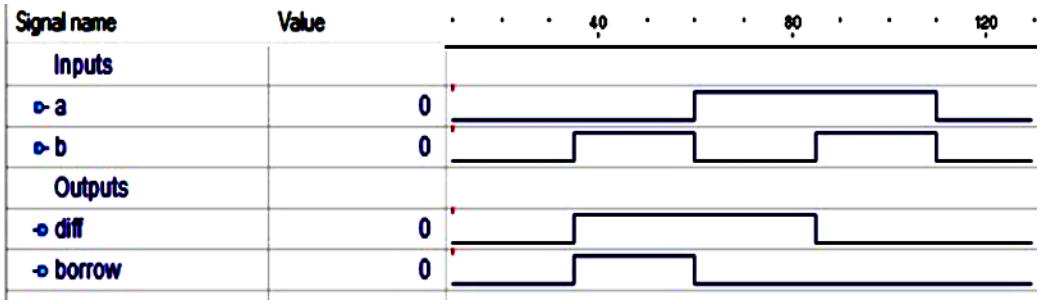
INPUTS		OUTPUTS	
a	**b**	**difference (d)**	**borrow (br)**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

RTL SCHEMATIC:



WAVEFORMS:

RESULT:- Hence the half substractor is simulated in VHDL using data flow modeling and its functionality is verified .



EXPERIMENT-3(b)

AIM: - To write a PROGRAM in VHDL for simulating the full substractor and to verify its functionality.

APPARATUS: Model Sim 5.7

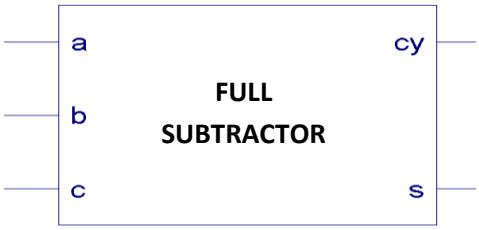
VHDL CODE:

```
library ieee;
use ieee.std_logic_1164.all;
entity fulladder is
port(a,b,c: in std_logic;d,br: out std_logic);
 end fulladder;
architecture dataflow of fulladder is
begin
d<= (a xor b)xor c;
br<= (not(a) and c) or (not(a) and b) or (b and c);
end dataflow;
```

TRUTH TABLE:-

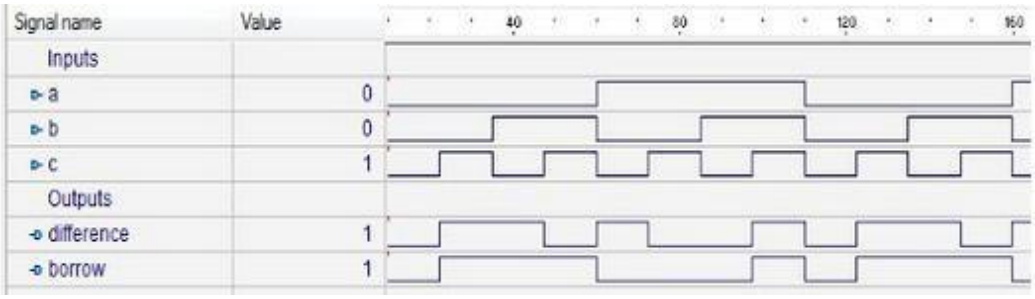
INPUTS			**OUTPUTS**	
a	**b**	**c**	**d**	**br**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

RTL SCHEMATIC:



WAVEFORMS:

RESULT: - Hence the full subtractor is simulated in VHDL and its functionality is verified.



EXERIMENT-4(a)

AIM:- To write a code in VHDL for simulating the 4x1 multiplexer and to observe the waveforms.

APPARATUS: Model Sim 5.7

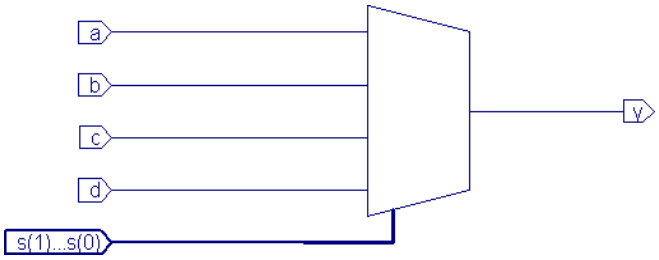
VHDL CODE:-

```
library ieee;
use ieee.std_logic_1164.all;
entity mux41 is
port(a,b,c,d:in std_logic;s:in std_logic_vector(1 downto 0);y:out std_logic);
end mux41;
architecture beh of mux41 is
begin
process(a,b,c,d,s)
begin
case s is
when  "00"=>y<=a;
   when "01"=>y<=b;
   when "10"=>y<=c;
   when "11"=>y<=d;
   when others=>y<='U';
end case;
end process;
end beh;
```

TRUTH TABLE:-

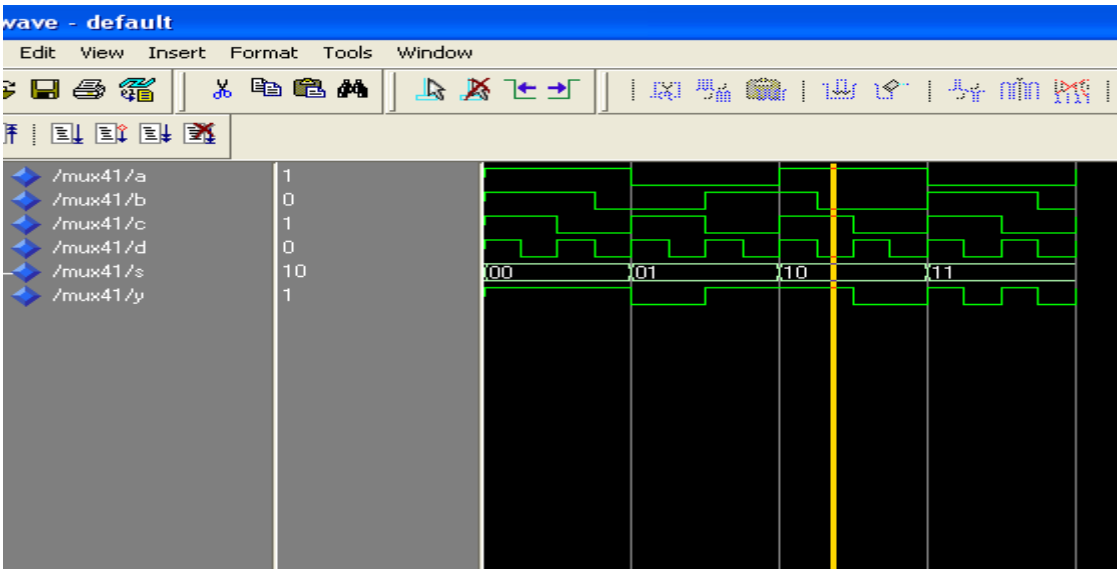
SELECT DATA INPUTS		**OUTPUTS**
S1	**S0**	**Y**
0	0	D0
0	1	D1
1	0	D2
1	1	D3

RTL SCHEMATIC:



MUX 4 x 1

WAVEFORMS:



RESULT:- Hence the 4x1 multiplexer is simulated in VHDL and its functionality is verified.

EXERIMENT-4(b)

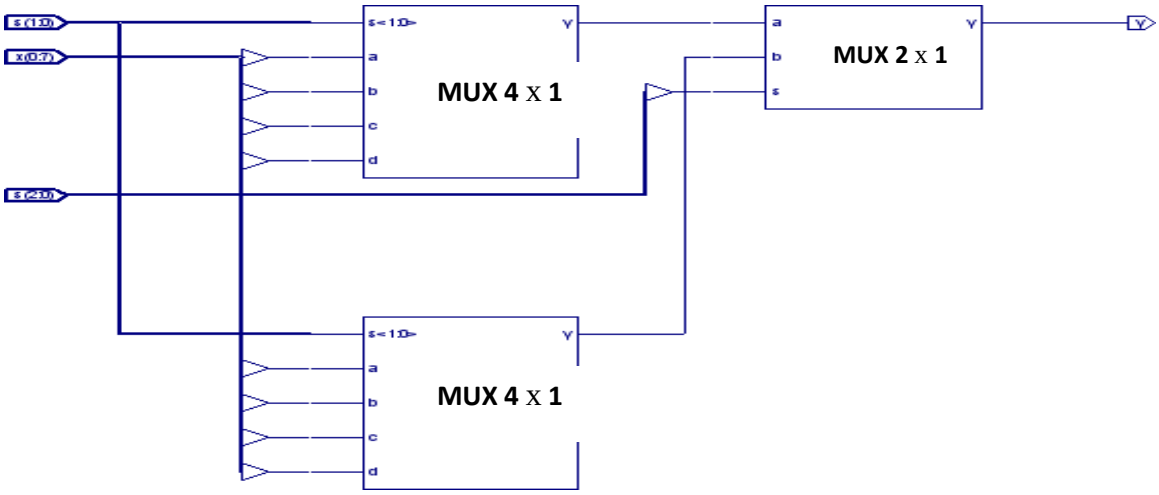
(iii) AIM:- To write a code in VHDL for simulating the 8x1 multiplexer and to verify its functionality.

APPARATUS: Model Sim 5.7

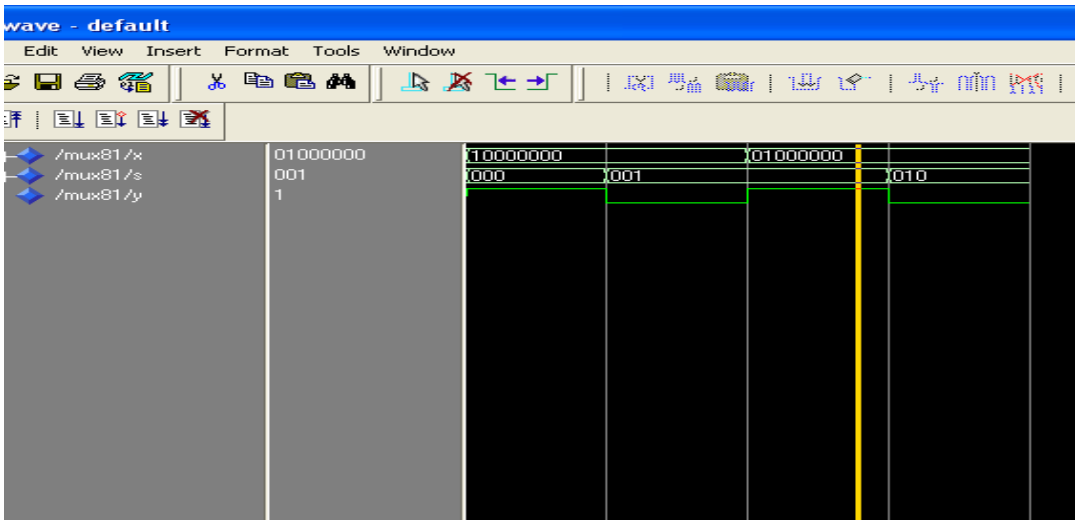
VHDL CODE:-

```
library ieee;
use ieee.std_logic_1164.all;
entity mux81 is
port(x:in std_logic_vector(0 to 7);s:in std_logic_vector(2 downto 0);y:out std_logic);
end mux81;
architecture structure of mux81 is
component mux41
port(a,b,c,d:in std_logic;s: in std_logic_vector(1 downto 0);y: out std_logic);
end component;
component mux21
port(a,b,s: in std_logic;y: out std_logic);
end component;
signal p1,p2: std_logic;
begin
X1: mux41 port map(x(0),x(1),x(2),x(3),s(1 downto 0),p1);
X2: mux41 port map(x(4),x(5),x(6),x(7),s(1 downto 0),p2);
X3: mux21 port map(p1,p2,s(2),y);
end structure;
```

RTL SCHEMATIC:



WAVEFORMS:



RESULT:- Hence the 8x1 multiplexer is simulated in VHDL using structural modeling and its functionality is verified.

EXERIMENT-5(a)

AIM:- To write a code in VHDL for simulating the 1x4 demultiplexer and to verify its functionality.

APPARATUS: Model Sim 5.7

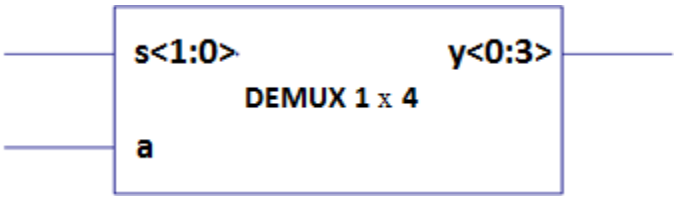
VHDL CODE:-

```
Library ieee;
use ieee.std_logic_1164.all;
entity dmux14 is
port(a: in std_logic;s: in std_logic_vector(1 downto 0);y: out std_logic_vector(0 downto 3));
end dmux14;
architecture dmux of dmux14 is
begin
process(a,s)
begin
y<="0000";
        case s is
when "00"=>y(0)<=a;
when "01"=>y(1)<=a;
   when "10"=>y(2)<=a;
   when "11"=>y(3)<=a;
   when others=>y<="UUUU";
   end case;
end process;
end dmux;
```

TRUTH TABLE:-

DATA INPUT	SELECT INPUTS		OUTPUTS			
	S1	S0	Y0	Y1	Y2	Y3
a	0	0	a	0	0	0
a	0	1	0	a	0	0
a	1	0	0	0	a	0
a	1	1	0	0	0	a

RTL SCHEMATIC:



WAVEFORMS:

RESULT:- Hence the 1x8 demultiplexer is simulated in VHDL using behavioral modeling and its functionality is verified

EXPERIMENT-5(b)

AIM:- To write a code in VHDL for simulating the 1x8 demultiplexer and to verify its functionality.

APPARATUS: Model Sim 5.7

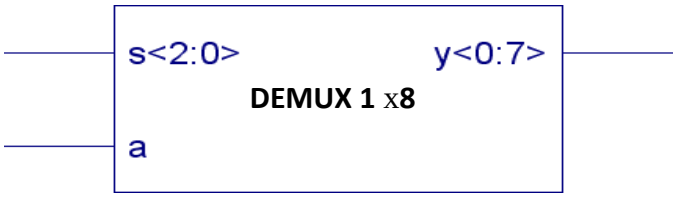
VHDL CODE:-

```
library ieee;
use ieee.std_logic_1164.all;
entity dmux81 is
port(a: in std_logic;s: in std_logic_vector(2 downto 0);y: out std_logic_vector(0 downto 7));
end dmux18;
architecture dmux of dmux18 is
begin
process(a,s)
begin
y<="00000000";
        case s is
when "000"=>y(0)<=a;
when "001"=>y(1)<=a;
   when "010"=>y(2)<=a;
   when "011"=>y(3)<=a;
when "100"=>y(4)<=a;
   when "101"=>y(5)<=a;
   when "110"=>y(6)<=a;
   when "111"=>y(7)<=a;
   when others=>y<="UUUUUUUU";
   end case;
end process;
end dmux;
```

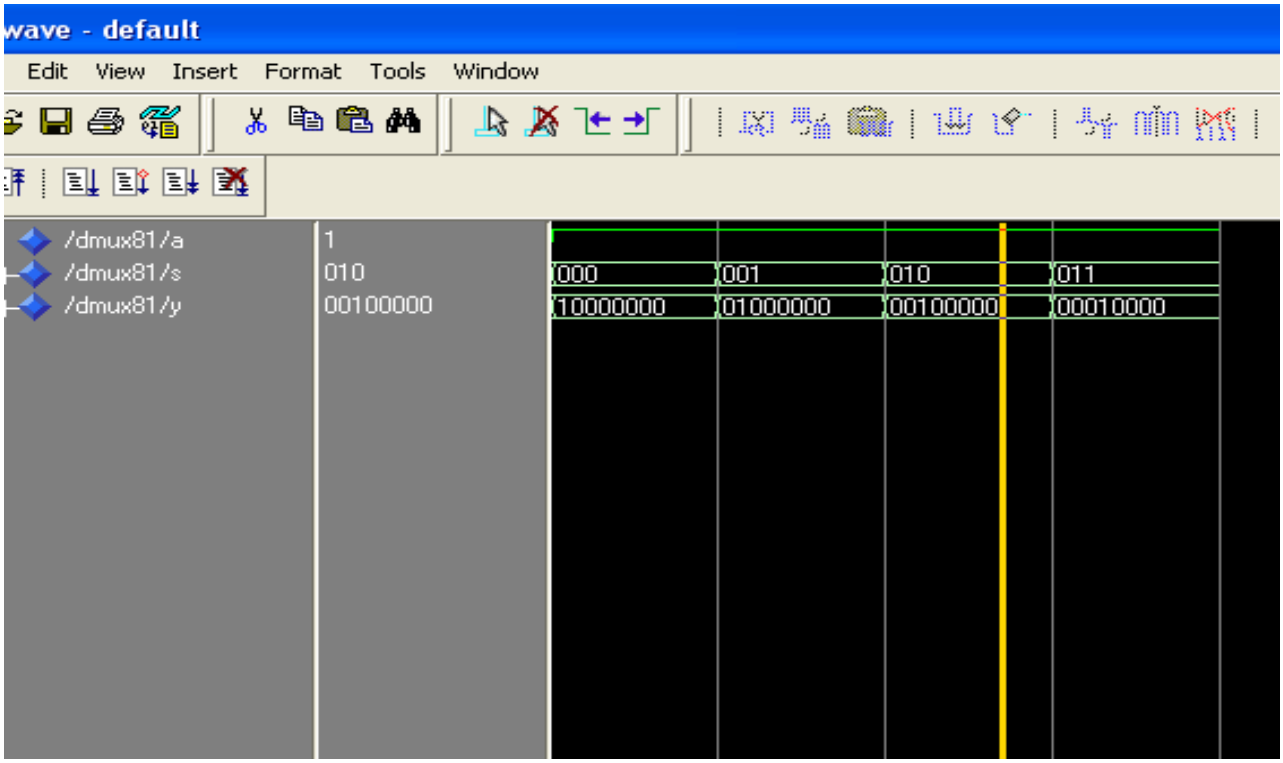
TRUTH TABLE:-

DATA INPUT	SELECT INPUTS			OUTPUTS							
	S2	S1	S0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
a	0	0	0	a	0	0	0	0	0	0	0
a	0	0	1	0	a	0	0	0	0	0	0
a	0	1	0	0	0	a	0	0	0	0	0
a	0	1	1	0	0	0	a	0	0	0	0
a	1	0	0	0	0	0	0	a	0	0	0
a	1	0	1	0	0	0	0	0	a	0	0
a	1	1	0	0	0	0	0	0	0	a	0
a	1	1	1	0	0	0	0	0	0	0	a

RTL SCHEMATIC:



WAVEFORMS:



EXPERIMENT- 6(a)

AIM: - To write a code in VHDL for simulating the 8:3 Priority Encoder and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:

```
 library ieee;
use ieee.std_logic_1164.all;
entity p_encoder_8_to_3 is
 port (a : in STD_LOGIC_VECTOR (7downto 0); d : out STD_LOGIC_VECTOR (2downto 0));
endp_encoder_8_to_3;
architecture behavioral of p_encoder_8_to_3 is
 begin
process (a)
begin
casea is
 when “00000001”=> d<=“000”;
when “0000001X”=> d<=“001”;
when “000001XX”=> d<=“010”;
when “00001XXX”=> d<=“011”;
when “0001XXXX”=> d<=“100”;
when “001XXXXX”=>d<=“101”;
when “01XXXXXX”=> d<=“110”;
when “1XXXXXXX”=> d<= “111”;
when others=>d<=“XXX”;
end case;
end process;
end behavioral;
```

TRUITH TABBLE

Inputs								outputs		
A7	A6	A5	A4	A3	A2	A1	A0	D2	D1	D0
0	0	0	0	0	0	0	0	X	X	X
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

EXPERIMENT-6(b):

AIM: - To write a code in VHDL for simulating the 3:8 decoder and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:

```
library ieee;
 use ieee.std_logic_1164.all;
entity decoder_3_to_8is
 port (a : in std_logic_vector (2 downto 0); d : out std_logic_vector (7 downto 0));
end decoder_3_to_8;
architecture behavioral of decoder_3_to_8 is
 begin
 process (a)
begin
case a is
 when “000”=> d<=“00000001”;
when “001”=> d<=“00000010”;
when “010”=> d<=“00000100”;
when “011”=> d<=“00001000”;
when “100”=> d<=“00010000”;
when “101”=> d<=“00100000”;
when “110”=> d<=“01000000”;
when others=>d<=“10000000”;
end case;
end process;
 end behavioral;
```

TRUTH TABLE:

Inputs			outputs							
A	**B**	**C**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

EXPERIMENT-7(a)

AIM: - To write a code in VHDL for simulating the SR flip-flop and to verify its functionality.

APPARATUS: Model Sim 5.7

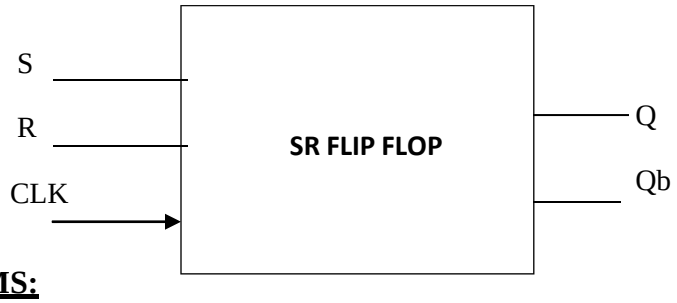
VHDL CODE:

```
library ieee;
use ieee.std_logic_1164.all;
entity SR is
port(S,R,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end SR;
architecture ff of SR is
begin
process(S,R,clk)
variable t,tb: std_logic;
begin
t:=Q;
tb:=Qb;
if (clk='0'and clk'event) then
if(S='0'and R='0') then t:=t;tb:=tb;
elsif(S='0'and R='1') then t:='0';tb:='1';
elsif(S='1'and R='0') then t:='1';tb:='0';
elsif(S='1'and R='1') then t:='U';tb:='U';
end if;
Q<=t;
Qb<=tb;
end if;
end process;
end ff;
```

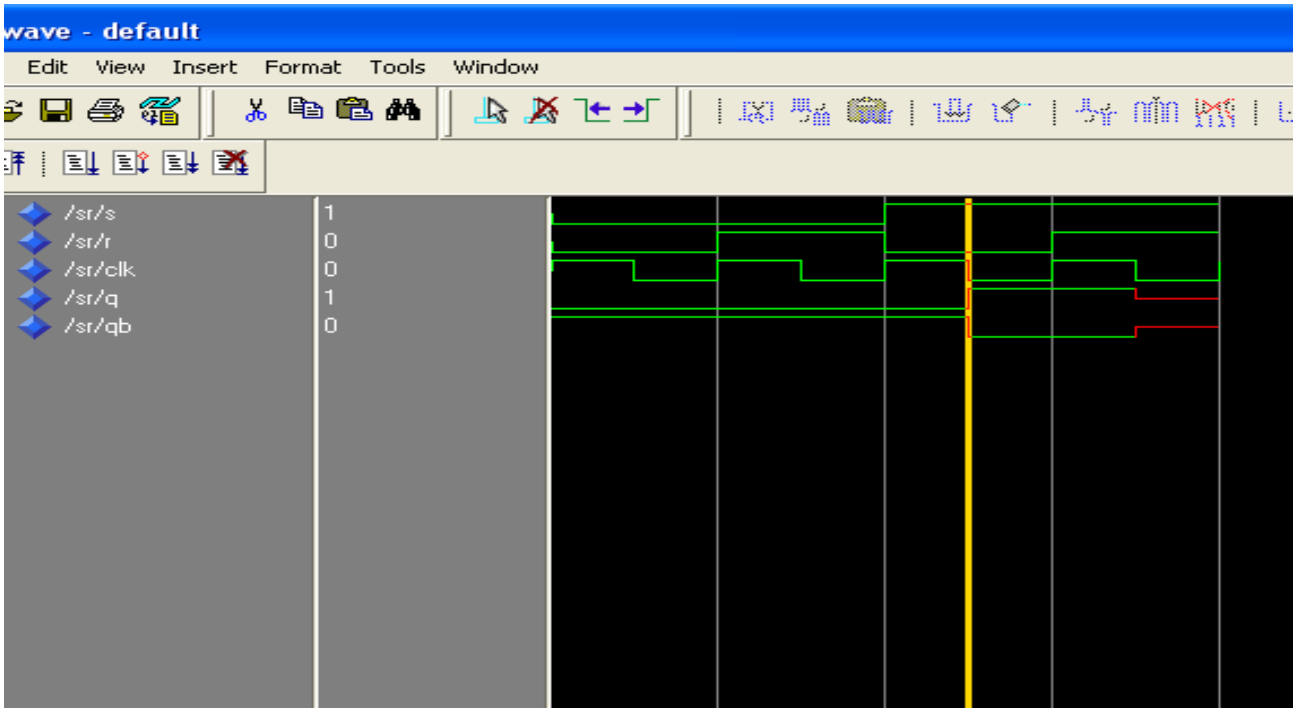
TRUTH TABLE:-

INPUTS		OUTPUTS	
S	**R**	**Q**	**Qb**
0	0	Q	Qb
0	1	0	1
1	0	1	0
1	1	X	X

RTL SCHEMATIC:



WAVEFORMS:



RESULT:-Hence the SR flip-flop is simulated in VHDL and its functionality is verified.

EXPERIMENT-7(b)

AIM:- (b)To write a code in VHDL for simulating the D flip-flop and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:-

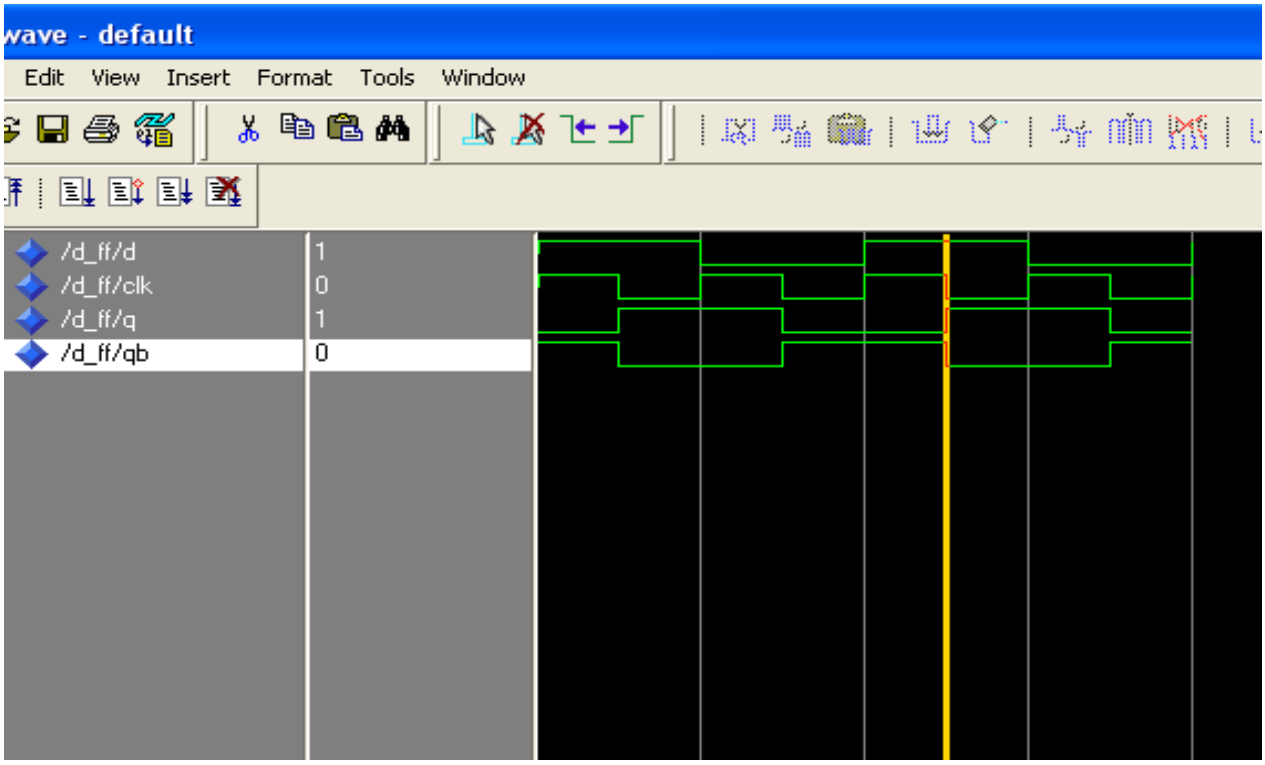
```
library ieee;
use ieee.std_logic_1164.all;
entity d_ff is
port(d,clk:in std_logic; Q:inout std_logic:='0';Qb:inout std_logic:='1');
end d_ff;
architecture behaviour of d_ff is
begin
process(d,clk)
begin
if (clk='0' and clk'event)then
q<=d;
qb<=not(d);
end if;
end process;
end behaviour;
```

TRUTH TABLE:-

INPUTS	**OUTPUTS**	
D	**Q**	**Qb**
0	0	1
1	1	0

RTL SCHEMATIC:

WAVEFORMS:



RESULT:- Hence the D flip-flop is simulated in VHDL and its functionality is verified.

EXPERIMENT-7(c)

AIM:- (b)To write a code in VHDL for simulating the JK flip-flop and to verify its functionality.

APPARATUS: Model Sim 5.7

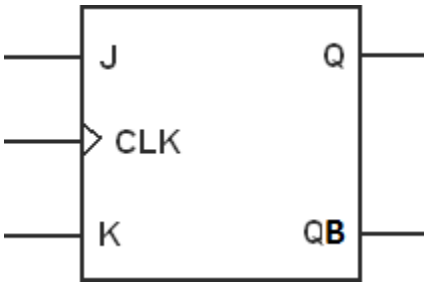
VHDL CODE:

```
library ieee;
use ieee.std_logic_1164.all;
entity JK_FF is
PORT( J,K,CLOCK: in std_logic; Q, QB: out std_logic);
end JK_FF;
Architecture behavioral of JK_FF is
begin
process(CLOCK)
variable TMP: std_logic;
begin
if(CLOCK='1' and CLOCK'EVENT) then
if(J='0' and K='0')then
TMP:=TMP;
elsif(J='1' and K='1')then
TMP:= not TMP;
elsif(J='0' and K='1')then
TMP:='0';
else
TMP:='1';
end  if;
end if;
Q<=TMP;
Q <=not TMP;
end PROCESS;
end behavioral;
```

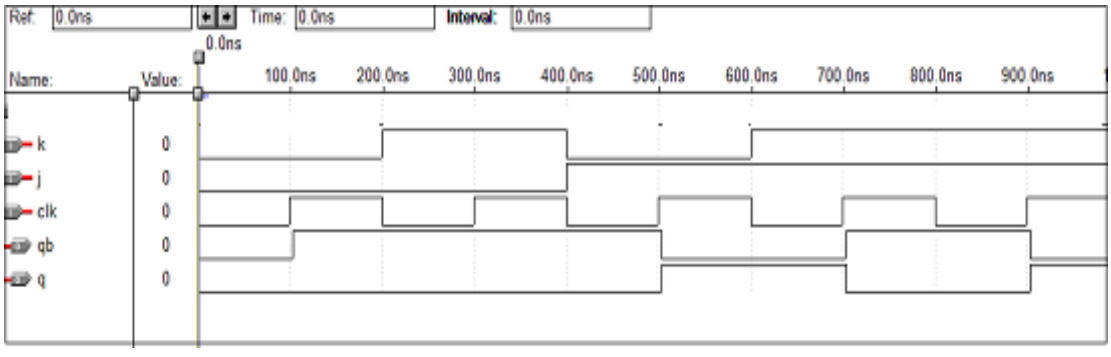
TRUTH TABLE:

Q	J	K	Q(T+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

RTL SCHEMATIC:



WAVEFORM:



VIVA QUESTIONS:

1. What are the various types of operators supported by VHDL?

2. What are the different concurrent assignment statements?
3. What are the different sequential assignment statements?
4. What is the purpose of PROCESS statement?
5. Give the general form of CASE statement.

ANSWERS:

1. Boolean(AND, OR,NAND, NOR,XOR, XNOR), arithmetic(*,/.MOD,REM,-,&), and relational(=,/<,<=,>,>=)

2. Simple signal assignment, selected signal assignment, conditional signal assignment, and generate statements.

3. IF statement, CASE statement, and two types of Loop statement(FOR-LOOP and WHILE-LOOP)

4. To separate the sequential statements from concurrent statements, PROCESS statement is used. The PROCESS statement appears inside an architecture body, and it encloses other statements within it. The IF, CASE, and LOOP statements can appear only inside a process.

.

EXPERIMENT-8(a)

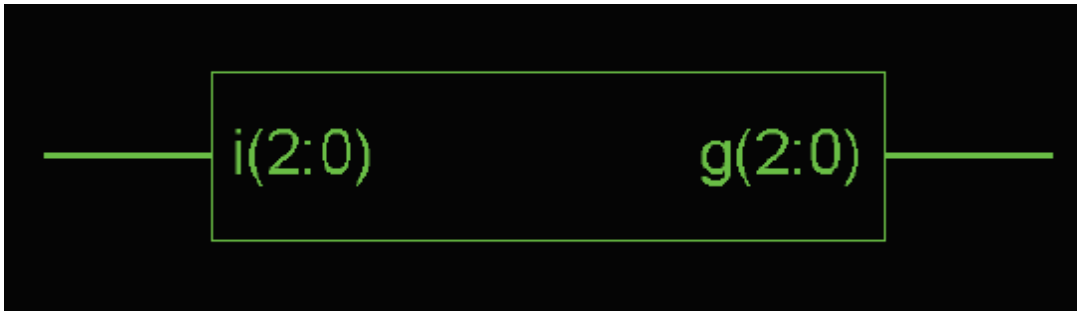
AIM:- To Design a BCD to GRAY converter using VHDL

APPARATUS: Model Sim 5.7

VHDL CODE:-

```
library ieee;
use ieee.std_logic_1164.all;
        entity bg is
        Port ( i : in STD_LOGIC_VECTOR (3 downto 0);
        g : out STD_LOGIC_VECTOR (3 downto 0));
        end bg;
        architecture Behavioral of bg is
        begin
        process(i)
        begin
        case i is
        when "0000" => g <= "000";
        when "0001" => g <= "001";
        when "0010" => g <= "011";
        when "0011" => g <= "010";
        when "0100" => g <= "110";
        when "0101" => g <= "111";
        when "0110" => g <= "101";
        when "1000" => g <= "000";
        when "1001" => g <= "001";

        when others => g <= "100";
        end case;
        end process;
end Behavioral;
```

RTL SCHEMATIC:



TRUTH TABLE:

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

EXPERIMENT-9(a)

AIM:- To write a code in VHDL for simulating the Serial In Serial Out(SISO) and Serial In Parallel Out(SIPO) shift registers using single entity and multiple architectures and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:-COMPONENT :

```
library ieee;
use ieee.std_logic_1164.all;
entity D is
port(D,clk: in std_logic;Q:inout std_logic:='0');
end D;
architecture behaviour of D is
begin
process(D,clk)
begin
if (clk='0' and clk'event)then
Q<=D;
end if;
end process;
end behaviour;
```

SISO :

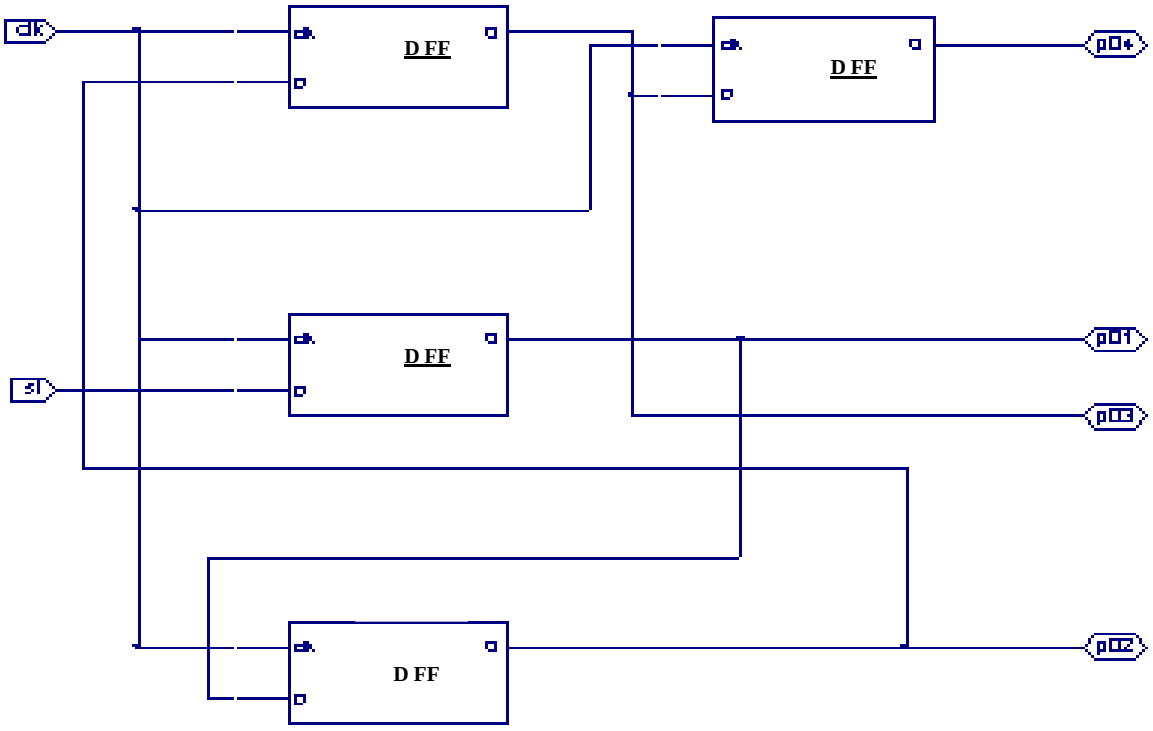
```
library ieee;
use ieee.std_logic_1164.all;
entity siso_sipo is
port(si,clk: in std_logic;s0,p01,p02,p03:inout std_logic);
end siso_sipo;
architecture siso_d of siso_sipo is
component D
port(D,clk: in std_logic;Q:inout std_logic:='0');
end component;
begin
```

```
D1: D port map(si,clk,p01);
D2: D port map(p01,clk,p02);
D3: D port map(p02,clk,p03);
D4: D port map(p03,clk,s0);
end siso_d;
```

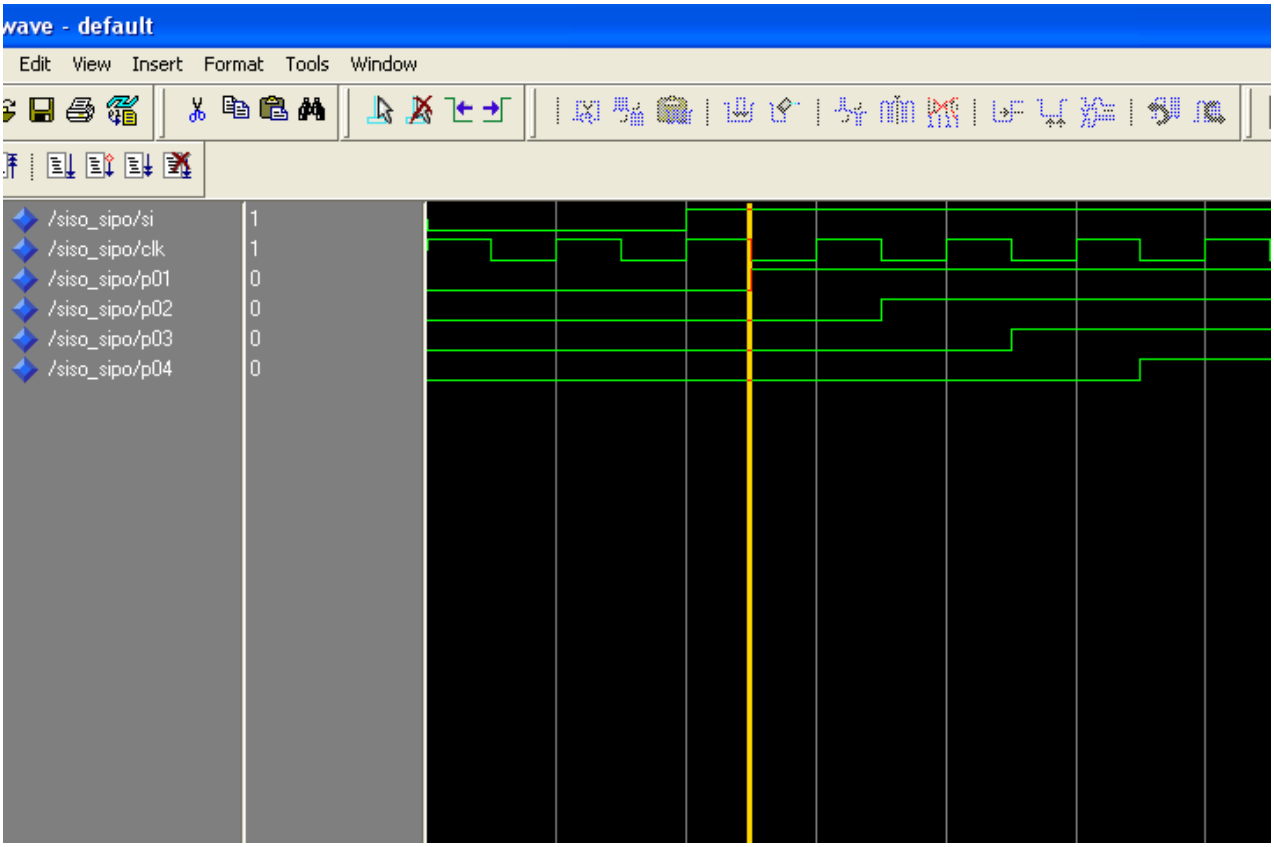
SIPO:

```
architecture sipo_d of siso_sipo is
component D
port(D,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end component;
begin
D1: D port map(si,clk,p01);
D2: D port map(p01,clk,p02);
D3: D port map(p02,clk,p03);
D4: D port map(p03,clk,p04);
end sipo_d;
```

RTL SCHEMATIC:



WAVEFORMS:



RESULT:- Hence the Serial In Serial Out(SISO) and Serial In Parallel Out(SIPO) shift registers using single entity and multiple architectures is simulated in VHDL and its functionality is verified.

EXPERIMENT-9(b)

AIM:- To write a code in VHDL for simulating the Parallel In Serial Out shift register(PISO) and to verify its functionality.

APPARATUS: Model Sim 5.7

VHDL CODE:-

COMPONENT D:-

```
library ieee;
use ieee.std_logic_1164.all;
entity D is
port(D,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end D;
architecture behaviour of D is
begin
process(D,clk)
begin
if (clk='0' and clk'event)then
Q<=D;
Qb<=not(D);
end if;
end process;
end behaviour;
```

COMPONENT OR2:-

```
library ieee;
use ieee.std_logic_1164.all;
entity or2 is
port(a,b: in std_logic;c: out std_logic);
end or2;
architecture dataflow of or2 is
begin
c<= a or b;
end dataflow;
```

COMPONENT AND2:-

```
library ieee;
use ieee.std_logic_1164.all;
entity and2 is
port(a,b: in std_logic;c: out std_logic);
end and2;
architecture dataflow of and2 is
begin
c<= a and b;
end dataflow;
```

COMPONENT NOT1:-

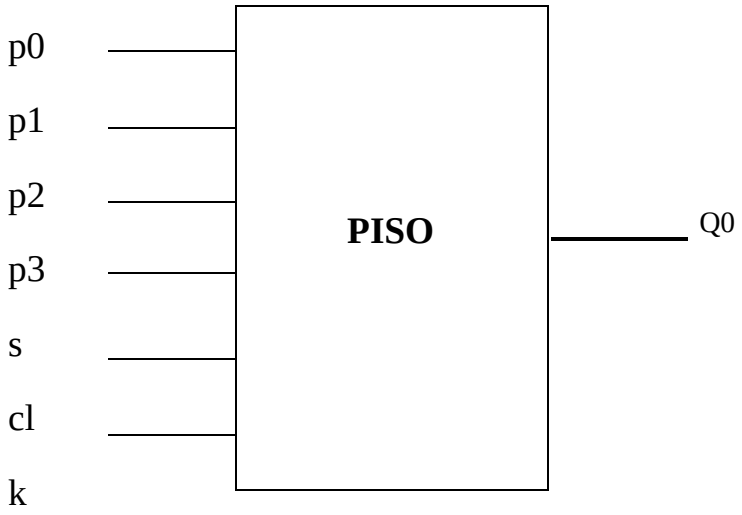
```
library ieee;
use ieee.std_logic_1164.all;
entity not1 is
port(a: in std_logic;c: out std_logic);
end not1;
architecture dataflow of not1 is
begin
c<= not(a);
end dataflow;
```

TOP MODULE:-

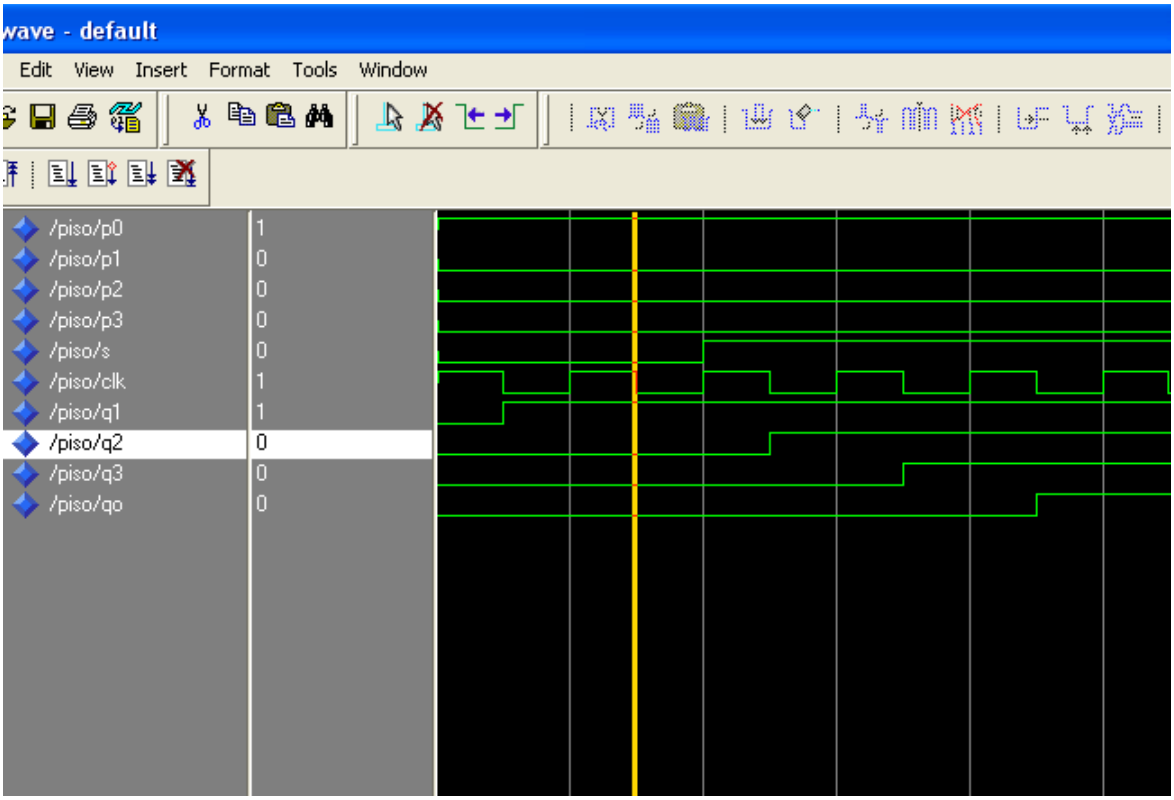
```
library ieee;
use ieee.std_logic_1164.all;
entity piso is
port(p0,p1,p2,p3,s,clk: in std_logic;Qo: inout std_logic);
end piso;
architecture piso of piso is
component D
port(D,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end component;
component and2
```

```
port(a,b: in std_logic;c: out std_logic);
end component;
component or2
port(a,b: in std_logic;c: out std_logic);
end component;
component not1
port(a: in std_logic;c: out std_logic);
end component;
signal s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,q1,q2,q3: std_logic;
begin
n1: not1 port map(s,s1);
D1: D port map(p0,clk,q1,open);
a1: and2 port map(s,q1,s2);
a2: and2 port map(s1,p1,s3);
O1: or2 port map(s2,s3,s4);
D2: D port map(s4,clk,q2,open);
a3: and2 port map(s,q2,s5);
a4: and2 port map(s1,p2,s6);
O2: or2 port map(s5,s6,s7);
D3: D port map(s7,clk,q3,open);
a5: and2 port map(s,q3,s8);
a6: and2 port map(s1,p3,s9);
O3: or2 port map(s8,s9,s10);
D4: D port map(s10,clk,Qo,open);
end piso;
```

RTL SCHEMATIC:



WAVEFORMS
:



RESULT:- Hence the Parallel In Serial Out shift register(PISO) is simulated in VHDL and its functionality is verified.

EXPERIMENT-10

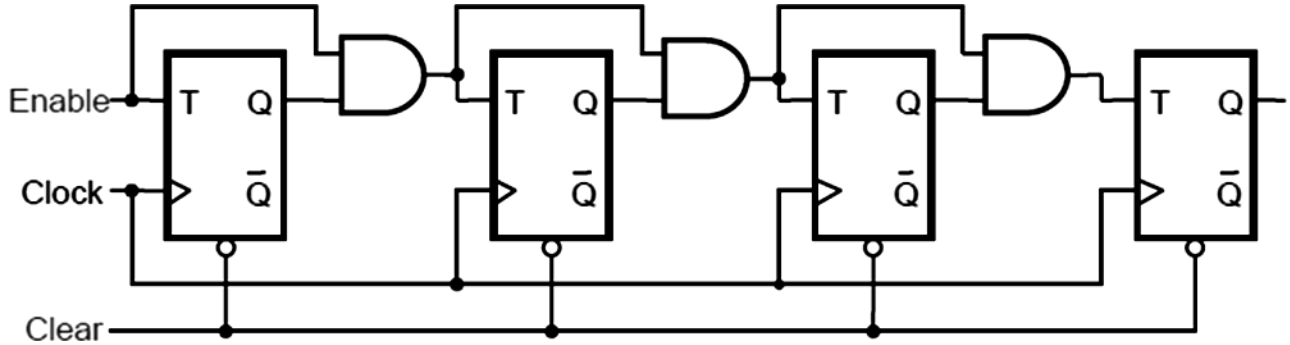
AIM:- To write a code in VHDL for simulating 4-bit Up-counter and to verify its functionality.

APPARATUS: Model Sim 5.7

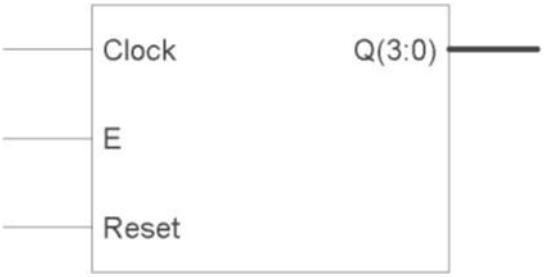
VHDL CODE:

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
entity counter is
port (clock, reset, e: in std_logic ;
q: out std_logic_vector (3 downto 0) ) ;
end counter ;
architecture behavior of counter is
signal count: std_logic_vector (3 downto 0);
begin
process (clock, reset)
begin
if reset = „0“ then
count <= „0000“;
elsif (clock“event and clock = „1“) then
if e=„1“ then
count <= count +1;
else
count <= count;
endif;
endif;
end process ;
q <= count;
end behavior ;
```

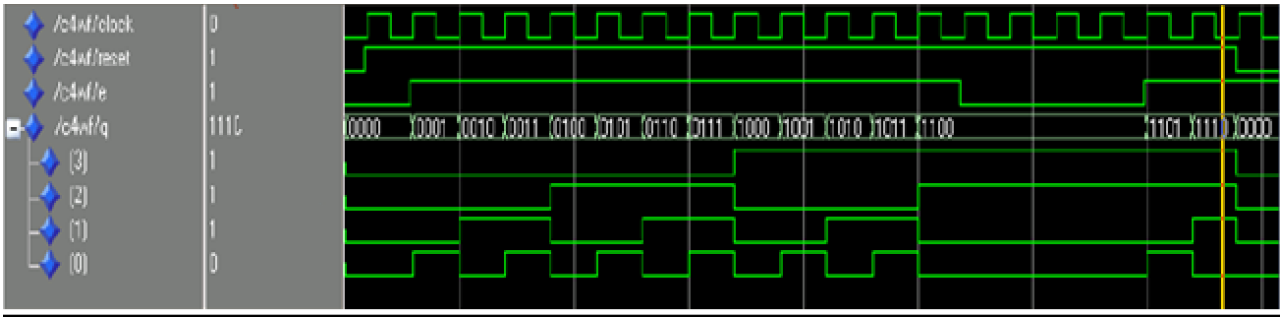
BLOCK DIAGRAM:



RTL SCHEMATIC:



WAVEFORMS:



MATLAB WORKBOOK

PERFORM A PLANETARY ORBIT SIMULATION.

```
clear;
 clc;
 at=149.5963995;
 bt=149.57552243;
 x0t=0;
 y0t=0;
 t=-4*pi:0.01:4*pi;
 xt=x0t+at*cos(t);
 yt=y0t+bt*sin(t);
 n=length(xt)
 for i=1:1:n
 xt2=xt(i);
 yt2=yt(i);
 al=38.1575;
 bl=38.0744275597;
 x0l=xt2-0.025165;
 y0l=yt2;
 t=-111*pi:0.01:111*pi;
 xl=x0l+al*cos(t);
 yl=y0l+bl*sin(t);
 m=length(xl);
 xs=2.4991665;
 ys=0;
 xm=xl(m-14*i);
 ym=yl(m-14*i);
 pts=(yt(i)-ys)/(xt(i)-xs);
 plt=(ym-yt(i))/(xm-xt(i));
 plot(xl,yl,'m')
 hold on
 plot(xl(m-14*i),yl(m- 14*i),'ok','MarkerSize',5,'MarkerFaceColor','w')
 hold on
 plot(xt,yt,'k')
 hold on
 plot(xt(i),yt(i),'ob','MarkerSize',10,'MarkerFaceColor','c')
 hold on
 plot(xs,ys,'or','MarkerSize',20,'MarkerFaceColor','y')
 porcentaje=(pts-plt)/((pts+plt)/2);
 per=abs(porcentaje);
 if per<0.1
 line([xt(i) xs],[yt(i) ys])
 line([xm xt(i)],[ym yt(i)])
 hold off
 vector(i)=i;
 end
 hold off
 axis([-200 200 -200 200])
 pause(.0000000001)
 end
```

ATMOSPHERIC DESCENT

```
function Y = atmosphere(h, vel, CL)
%(c) 2005 Ashish Tewari
R = 287; %sea-level gas constant for air (J/kg.K)
go = 9.806; %sea level acceleration due to gravity (m/s^2)
Na = 6.0220978e23; %Avogadro's number
sigma = 3.65e-10; %collision diameter (m) for air
S = 110.4; %Sutherland's temperature (K)
Mo = 28.964; %sea level molecular weight (g/mole)
To = 288.15; %sea level temperature (K)
Po = 1.01325e5; %sea level pressure (N/m^2)
re = 6378.14e3; %earth's mean radius (m)
Beta = 1.458e-6; %Sutherland's constant (kg/m.s.K^0.5)
gamma = 1.405; %sea level specific-heat ratio
B = 2/re; layers = 21; Z = 1e3*[0.00; 11.0191; 20.0631; 32.1619;
47.3501; 51.4125;
71.8020; 86.00; 100.00; 110.00; 120.00; 150.00; 160.00; 170.00; 190.00;
230.00; 300.00; 400.00; 500.00; 600.00; 700.00; 2000.00];
T = [To; 216.65; 216.65; 228.65; 270.65; 270.65; 214.65; 186.946;
210.65; 260.65; 360.65; 960.65; 1110.60; 1210.65; 1350.65; 1550.65;
1830.65; 2160.65; 2420.65; 2590.65; 2700.00; 2700.0];
M = [Mo; 28.964; 28.964; 28.964; 28.964; 28.964; 28.962; 28.962;
28.880;
28.560; 28.070; 26.920; 26.660; 26.500; 25.850; 24.690;
22.660; 19.940; 17.940; 16.840; 16.170; 16.17];
LR = [-6.5e-3; 0; 1e-3; 2.8e-3; 0; -2.8e-3; -2e-3;
1.693e-3; 5.00e-3; 1e-2; 2e-2; 1.5e-2; 1e-2; 7e-3; 5e-3; 4e-3;
3.3e-3; 2.6e-3; 1.7e-3; 1.1e-3; 0];
rho0 = Po/(R*To); P(1) = Po; T(1) = To; rho(1) = rho0;
for i =1:layers
if ~(LR(i) == 0)
C1 = 1 + B*( T(i)/LR(i) - Z(i) );
C2 = C1*go/(R*LR(i));
C3 = T(i+1)/T(i);
C4 = C3^(-C2);
C5 = exp( go*B*(Z(i+1)-Z(i))/(R*LR(i)) );
P(i + 1) = P(i)*C4*C5;
C7 = C2 + 1;
rho(i + 1) = rho(i)*C5*C3^(-C7);
else
C8 = -go*(Z(i+1)-Z(i))*(1 - B*(Z(i + 1) + Z(i))/2)/(R*T(i));
P(i+1) = P(i)*exp(C8); rho(i+1) = rho(i)*exp(C8);
end
end
for i = 1:21
if h < Z(i+1)
if ~(LR(i)== 0)
C1 = 1 + B*( T(i)/LR(i) - Z(i) );
TM = T(i) + LR(i)*(h - Z(i));
C2 = C1*go/(R*LR(i));
C3 = TM/T(i);
C4 = C3^(-C2);
C5 = exp( B*go*(h - Z(i))/(R*LR(i)) );
PR = P(i)*C4*C5; %Static Pressure (N/m^2)
```

```
C7 = C2 + 1;
rhoE = C5*rho(i)*C3^(-C7); %Density (kg/m^3)
else
TM = T(i);
C8 = -go*(h - Z(i))*(1 - (h + Z(i))*B/2)/(R*T(i));
PR = P(i)*exp(C8); %Static Pressure (N/m^2)
rhoE = rho(i)*exp(C8); %Density (kg/m^3)
end
MOL = M(i) + ( M(i+1)-M(i) )*( h - Z(i) )/( Z(i+1) - Z(i) );
TM = MOL*TM/Mo; %Kinetic Temperature

asound = sqrt(gamma*R*TM); % Speed of Sound (m/s)
MU = Beta*TM^1.5/(TM + S); % Dynamic Viscosity Coeff. (N.s/m^2)
KT = 2.64638e-3*TM^1.5/(TM + 245.4*10^(-12/TM));
Vm = sqrt(8*R*TM/pi); m = MOL*1e-3/Na; n = rhoE/m;
F = sqrt(2)*pi*n*sigma^2*Vm;
L = Vm/F; % Mean free-path (m)
Mach = vel/asound; % Mach Number
T0 = TM*(1 + (gamma - 1)*Mach^2/2);
MU0 = Beta*T0^1.5/(T0 + S);
RE0 = rhoE*vel*CL/MU0;
RE = rhoE*vel*CL/MU; % Reynold's Number
Kn = L/CL; % Knudsen Number
Kno = 1.25*sqrt(gamma)*Mach/RE0;
%flow regime parameter
if Kn >= 10
d = 1; % free-molecule flow
elseif Kn <= 0.01
d = 2; % continuum flow
else
d = 3; % transition flow
end
Y = [TM; rhoE; Mach; Kn; asound; d; PR; MU; RE; KT];
return;
end
end
0
```

HOHMANN TRANSFER

```
% This is for Hohmann TRANSFER
% Stephen Walker 2009 (stephen.walker@student.uts.edu.au)
% Thrust force is disabled and the hohmann transfers have been implemented by
directly altering the
% magnitude of the velocity vector.

clear
count_increment = 10;
%this is the 'delta t'
%This has been set not just to set accuracy, but to control the rate of the
%animated display


scenario_duration = 100000
counter = scenario_duration / count_increment;

sub_sat_theta = -80;
%the start point is it at this location for simplicity

launch_lattitude = 0 ;
launch_longitude = 0;

M = 6.67e-11;
H= 5.975e24;
G = H  ; %
radius = 6371000;%metres
%count_increment = engine(4,1);
geosync_radius = 20000000; %42164142.15

% this sets up intial velocity conditions
siderial_day  = 0.9972695664 *24 * 60 * 60;




%Initial Positions and velocities are set
pos.mag(1) =  6371000 + 2000000;
pos.theta(1) = launch_longitude * pi/180;
pos.phi(1) = launch_lattitude * pi /180;

actual_pos_theta(1) = pos.theta(1);
%these initial values are to ensure a circular starting orbit.
%since the start point is 0,0, lat/long, the phi value exactly translates
%to inclination. This has been zeroed to simplify transfer, but can be
%increased to incline the orbit before hohmann transfer.

%the ideal velocity / altitude is used to set initial velocity, to
%elimenate any eccentricity in the orbit .

vel.mag(1)= sqrt(G*M/pos.mag(1));
vel.theta(1)= pi/2;
vel.phi(1) = 0.15;
% a small inclination of 0.15 radians is added to the orbit, to make the mercator
plot more
```

```
% interesting

% the engine_calc function is not used, and as the velocity is directly
% altered instantly without altering mass, these are redundant
 delta_mass = 212.78;
 thrust = 88143;

craft.mass(1) = 35000;
craft.thrust(1) = 0;

% vectors are converted between spherical and cartesian
[vel.x(1),vel.y(1),vel.z(1)] = sph2cart(vel.theta(1),vel.phi(1),vel.mag(1));
[pos.x(1),pos.y(1),pos.z(1)] = sph2cart(pos.theta(1),pos.phi(1),pos.mag(1));


%initial forces defined.
%  force.theta = 0;
% force.phi = 0;
% force.mag =0;


delta_v_periapsis = 0;
delta_v_apoapsis = 0;
flag(1) = 'b';

THETA = linspace(0,2*pi);
XXX= radius * cos(THETA);
YYY= radius * sin(THETA);

for t = 2:counter;


    if round((pos.theta(t-1))* 180 /pi) == (sub_sat_theta + 180)&& flag(t-1)
=='y';
       geosync_radius = geosync_radius +  (1000000 * rand(1));
    end
     if round((pos.theta(t-1))* 180 /pi) == (sub_sat_theta + 180)&& flag(t-1)
=='g';
       flag(t-1) = 'b';
    end
flag(t) = flag(t-1);
 % this will begin the hohmann transfer, when the spacecraft is on the
 % opposite side of the planet. Hohmann transfer equations are intended for
 % engines with a high SI, so the delta_v will be imparted quickly. As a
 %  simplification, the delta_v is direcly applied instantly to the velocity
 % vector, without modeling the delta_v over time,as would be the case with
 % a real booster.

 %the hohmann transfer burn(velocity change) is initiated, when spacecraft
 %is on the opposite side of the earth to the desired sub sat point.
 if round((pos.theta(t-1))* 180 /pi) ==  (sub_sat_theta + 180) && flag(t) == 'b';
 pos_mag = pos.mag(t-1) ;
%delta_v_periapsis & delta_v_apoapsis calculated from current pos.mag value and
target geosync_radius
delta_v_periapsis = sqrt(G*M/pos_mag)*(sqrt(2*geosync_radius /(pos_mag +
geosync_radius)) -1);

vel.mag = vel.mag + delta_v_periapsis;
```

```
delta_v_apoapsis = sqrt(G*M/geosync_radius)*(1 - sqrt (2* pos_mag /(pos_mag +
geosync_radius)));

flag(t) ='r';
 periapsis_burn = t;

%the flags are to allow only one periapsis burn/delta v

 end

%If perriapsis burn has occured(flag=) AND the spacecraft is close to
%geosync altitude AND the velocity vector has changed direction (turning
%back to earth) the apoapsis burn(velocity change) is impemented.
     if vel.theta - pos.theta(t-1) > pi/2 && pos.mag(t-1) > (geosync_radius -
500000) && flag(t) == 'r';
       current_vel = vel.mag;
       delta_vee = delta_v_apoapsis;
         vel.mag = current_vel + delta_vee; % should be 3066.262945 - its a bit
high


  flag(t) ='g';
  apoapsis_burn = t;
     end

     %craft.mass reserved for expansion of code
 craft.mass(t) = craft.mass(1);




%The only force on the vehicle is the engines for Hohmann Transfer. As a
%simplificiation, these arent modelled, and the dlta_velocity is applied
%instantly.


%
%[force.x,force.y,force.z] = sph2cart(force.theta,force.phi,force.mag);
%gravity vector at t is calculated
grav.theta = pi + pos.theta(t-1);
grav.phi =  -1 * pos.phi(t-1);
grav.mag = craft.mass(t) * G * M /(pos.mag(t-1)^2);

[grav.x,grav.y,grav.z] = sph2cart(grav.theta,grav.phi,grav.mag);
%net force is calculated. This is redundant, as the only force in this
%model is gravity - engine thrust is directly added to velocity as
%delta_velocity
% net.x = force.x + grav.x;
% net.y = force.y + grav.y;
% net.z = force.z + grav.z;

net.x =  grav.x;
net.y =  grav.y;
net.z =  grav.z;



[vel.x,vel.y,vel.z] = sph2cart(vel.theta,vel.phi,vel.mag);
%Net Force used to update velocity vector
```

```
vel.x = vel.x + net.x /craft.mass(t) * count_increment;
vel.y = vel.y + net.y /craft.mass(t) * count_increment;
vel.z = vel.z + net.z /craft.mass(t) * count_increment;

%velocity vector updates position vector
pos.x(t) = pos.x(t-1) + vel.x * count_increment;
pos.y(t) = pos.y(t-1) + vel.y * count_increment;
pos.z(t) = pos.z(t-1) + vel.z * count_increment;

%before the end of the loop, all the current vectors at t are converted
%back to spherical, for the t+1 iteration
%[net.theta(t),net.phi(t),net.mag(t)] = cart2sph(pos.x(t),net.y(t),net.z(t));
[pos.theta(t),pos.phi(t),pos.mag(t)] = cart2sph(pos.x(t),pos.y(t),pos.z(t));
[vel.theta,vel.phi,vel.mag] = cart2sph(vel.x,vel.y,vel.z);



 % The output for the mercator map is shifted per time (t) to make an
 % allowance for the rotation of the earth
     actual_pos_theta(t) = pos.theta(t) + (t * (2*pi /siderial_day));

if actual_pos_theta(t) > (2* pi) ;
    actual_pos_theta(t)= actual_pos_theta(t) - (2 * pi);
end

% Altitude watch - if the spacecraft altitude drops below the earths radius
% the simulation aborts
 if pos.mag(t-1) < 6317000;

   %as the code takes time to run, these flags indicate progress
    break
else
end
 if rem(t,1000) == 0
     disp( (t/counter)* 100)
 end


 % the animated plot is done. This is not the most efficient way, as it is
 % drawing the whole plot every iteration
   figure(79)
 clf
 hold on
% t
  scatter (pos.x(t),pos.y(t),5,flag(t));
   plot (pos.x,pos.y);
  plot(XXX,YYY,'g--');
%axis equal

axis ([ -43000000, 43000000,-43000000,43000000])
axis equal
  title ('equatorial view - (looking down from above north pole)')


  % a ground track is drawn. The variable vis is set to 1 presently, but
  % will be used to determine satellite coverage area on the earth (in the next
release).
vis = 1
 if flag(t) ~= flag(t-1)
```

```
    figure(88)
 clf
 m = imshow('lo_res_earth.jpg');
  hold on
%
 scatter ((actual_pos_theta*163)+512 ,(pos.phi*-163)+256,vis,'red');
%
 title ('mercator map - ground track')
 end

end
```