# DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

## DIGITAL SIGNAL PROCESSING LAB MANUAL

**IVYEAR I SEMESTER (ECE)**


GURUNANAK ENGINEERING COLLEGE
Ibrahimpatnam,R.R.District,Hyderabad-506501,A.P

# DIGITAL SIGNAL PROCESSING LAB (IV-I SEM)
## INDEX

# CHAPTER-I 1. INTRODUCTION TO DSP PROCESSORS

A signal can be defined as a function that conveys information, generally about the state or behavior of a physical system. There are two basic types of signals viz Analog (continuous time signals which are defined along a continuum of times) and Digital (discrete-time).

Remarkably, under reasonable constraints, a continuous time signal can be adequately represented by samples, obtaining discrete time signals. Thus digital signal processing is an ideal choice for anyone who needs the performance advantage of digital manipulation along with today's analog reality.

Hence a processor which is designed to perform the special operations(digital manipulations) on the digital signal within very less time can be called as a Digital signal processor. The difference between a DSP processor, conventional microprocessor and a microcontroller are listed below.

**Microprocessor** or General Purpose Processor such as Intel xx86 or Motorola 680xx family
Contains - only CPU
-No RAM
-No ROM
-No I/O ports
-No Timer

**Microcontroller** such as 8051 family
Contains - CPU
- RAM
- ROM
-I/O ports
- Timer &
- Interrupt circuitry
Some Micro Controllers also contain A/D, D/A and Flash Memory

**DSP Processors** such as Texas instruments and Analog Devices

Contains - CPU
- RAM
-ROM
- I/O ports
- Timer

Optimized for – fast arithmetic
- Extended precision
- Dual operand fetch
- Zero overhead loop
- Circular buffering

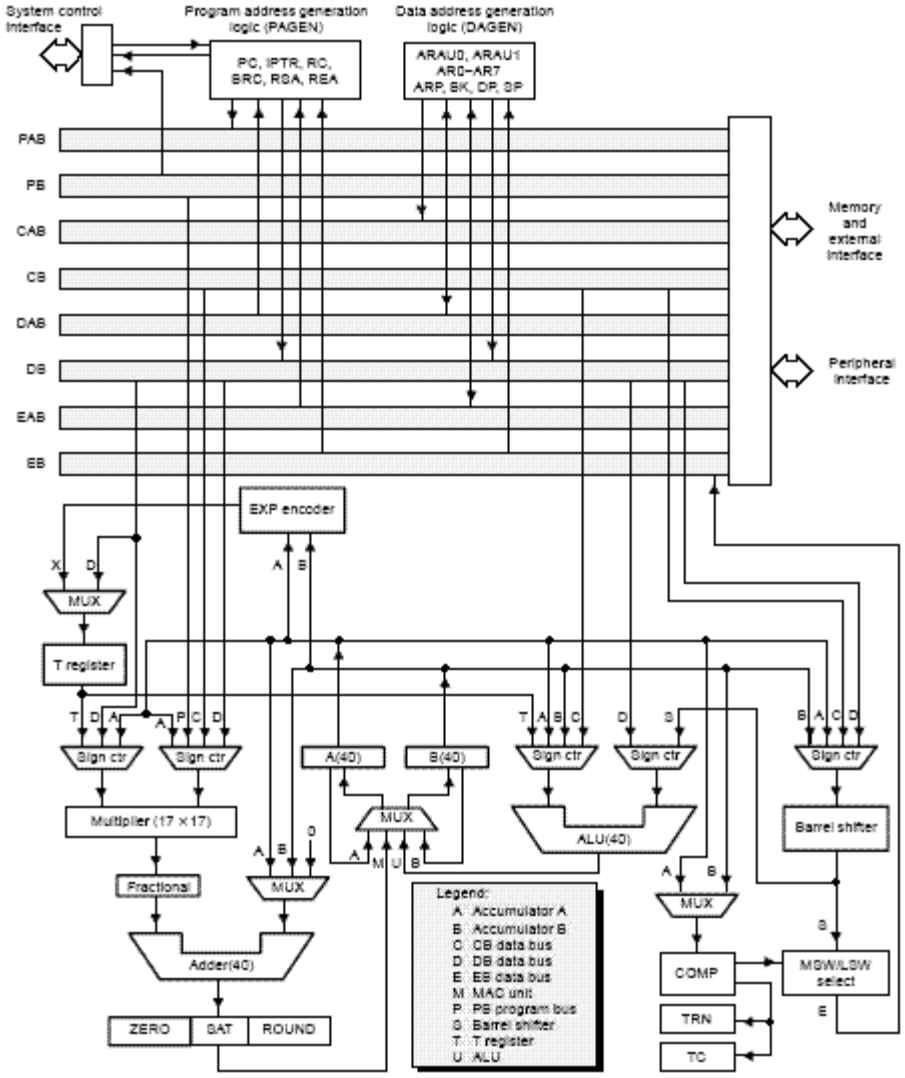
The basic features of a DSP Processor are

| **Feature** | **Use** |
|---|---|
| Fast-Multiply accumulate | Most DSP algorithms, including filtering, transforms, etc. are multiplication- intensive |
| Multiple – access memory architecture | Many data-intensive DSP operations require reading a program instruction and multiple data items during each instruction cycle for best performance |
| Specialized addressing modes | Efficient handling of data arrays and first-in, first-out buffers in memory |
| Specialized program control | Efficient control of loops for many iterative DSP algorithms. Fast interrupt handling for frequent I/O operations. |
| On-chip peripherals and I/O interfaces | On-chip peripherals like A/D converters allow for small low cost system designs. Similarly I/O interfaces tailored for common peripherals allow clean interfaces to off-chip I/O devices. |

## 2. ARCHITECTURE OF 6713 DSP PROCESSOR

This chapter provides an overview of the architectural structure of the TMS320C67xx DSP, which comprises the central processing unit (CPU), memory, and on-chip peripherals. The C67xE DSPs use an advanced modified Harvard architecture that maximizes processing power with eight buses. Separate program and data spaces allow simultaneous access to program instructions and data, providing a high degree of parallelism. For example, three reads and one write can be performed in a single cycle. Instructions with parallel store and application-specific instructions fully utilize this architecture. In addition, data can be transferred between data and program spaces. Such Parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. Also, the C67xx DSP includes the control mechanisms to manage interrupts, repeated operations, and function calling.

Fig 2 – 1 BLOCK DIAGRAM OF TMS 320VC 6713




## Bus Structure

The C67xx DSP architecture is built around eight major 16-bit buses (four program/data buses and four address buses):
_ The program bus (PB) carries the instruction code and immediate operands from program memory.
_ Three data buses (CB, DB, and EB) interconnect to various elements, such as the CPU, data address generation logic, program address generation logic, on-chip peripherals, and data memory.
_ The CB and DB carry the operands that are read from data memory.
_ The EB carries the data to be written to memory.
_ Four address buses (PAB, CAB, DAB, and EAB) carry the addresses needed for instruction execution.

The C67xx DSP can generate up to two data-memory addresses per cycle using the two auxiliary register arithmetic units (ARAU0 and ARAU1). The PB can carry data operands stored in program space (for instance, a coefficient table) to the multiplier and adder for multiply/accumulate operations or to a destination in data space for data move instructions (MVPD and READA). This capability, in conjunction with the feature of dual-operand read, supports the execution of single-cycle, 3-operand instructions such as the FIRS instruction. The C67xx DSP also has an on-chip bidirectional bus for accessing on-chip peripherals. This bus is connected to DB and EB through the bus exchanger in the CPU interface. Accesses that use this bus can require two or more cycles for reads and writes, depending on the peripheral's structure.

## Central Processing Unit (CPU)

The CPU is common to all C67xE devices. The C67x CPU contains:

_ 40-bit arithmetic logic unit (ALU)
_ Two 40-bit accumulators
_ Barrel shifter
_ 17 × 17-bit multiplier
_ 40-bit adder
_ Compare, select, and store unit (CSSU)
_ Data address generation unit
_ Program address generation unit

### Arithmetic Logic Unit (ALU)

The C67x DSP performs 2s-complement arithmetic with a 40-bit arithmetic logic unit (ALU) and two 40-bit accumulators (accumulators A and B). The ALU can also perform Boolean operations. The ALU uses these inputs:

_ 16-bit immediate value
_ 16-bit word from data memory
_ 16-bit value in the temporary register, T
_ Two 16-bit words from data memory

_ 32-bit word from data memory
_ 40-bit word from either accumulator

The ALU can also function as two 16-bit ALUs and perform two 16-bit operations simultaneously.

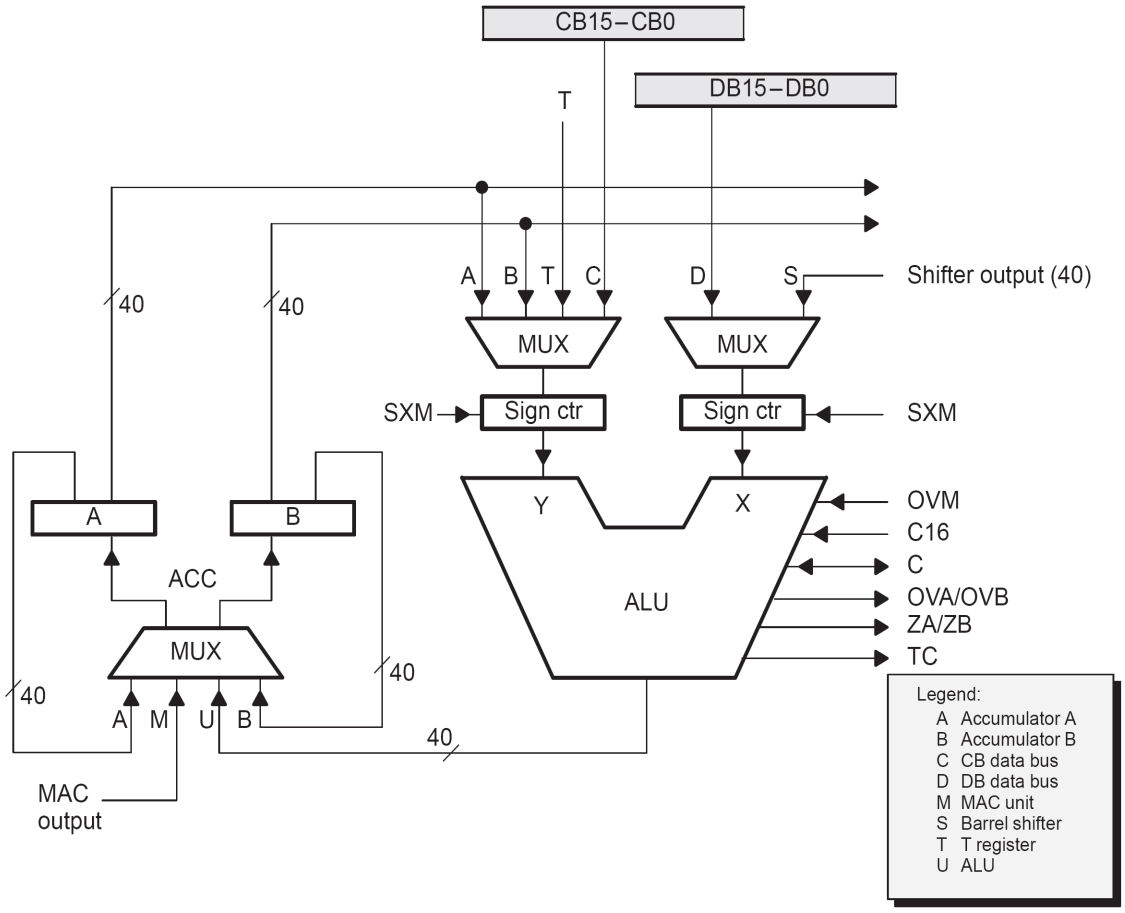



Fig 2 – 2 ALU UNIT

**Accumulators**

Accumulators A and B store the output from the ALU or the multiplier/adder block. They can also provide a second input to the ALU; accumulator A can be an input to the multiplier/adder. Each accumulator is divided into three parts:
_ Guard bits (bits 39–32)
_ High-order word (bits 31–16)
_ Low-order word (bits 15–0)
Instructions are provided for storing the guard bits, for storing the high- and the low-order accumulator words in data memory, and for transferring 32-bit accumulator words in or out of data memory. Also, either of the accumulators can be used as temporary storage for the other.

**Barrel Shifter**

The C67x DSP barrel shifter has a 40-bit input connected to the accumulators or to data memory (using CB or DB), and a 40-bit output connected to the ALU or to data memory (using EB). The barrel shifter can produce a left shift of 0 to 31 bits and a right shift of 0 to 16 bits on the input data. The shift requirements are defined in the shift count field of the instruction, the shift count field (ASM) of status register ST1, or in temporary register T (when it is designated as a shift count register).The barrel shifter and the exponent encoder normalize the values in an accumulator in a single cycle. The LSBs of the output are filled with 0s, and the MSBs can be either zero filled or sign extended, depending on the state of the sign-extension mode bit (SXM) in ST1. Additional shift capabilities enable the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.

**Multiplier/Adder Unit**

The multiplier/adder unit performs 17 _ 17-bit 2s-complement multiplication with a 40-bit addition in a single instruction cycle. The multiplier/adder block consists of several elements: a multiplier, an adder, signed/unsigned input control logic, fractional control logic, a zero detector, a rounder (2s complement), overflow/saturation logic, and a 16-bit temporary storage register (T). The multiplier has two inputs: one input is selected from T, a data-memory operand, or accumulator A; the other is selected from program memory, data memory, accumulator A, or an immediate value. The fast, on-chip multiplier allows the C54x DSP to perform operations efficiently such as convolution, correlation, and filtering. In addition, the multiplier and ALU together execute multiply/accumulate (MAC) computations and ALU operations in parallel in a single instruction cycle. This function is used in determining the Euclidian distance and in implementing symmetrical and LMS filters, which are required for complex DSP algorithms. See section 4.5, Multiplier/Adder Unit, on page 4-19, for more details about the multiplier/adder unit.

Fig 2 – 3 MULTIPLIER/ADDER UNIT

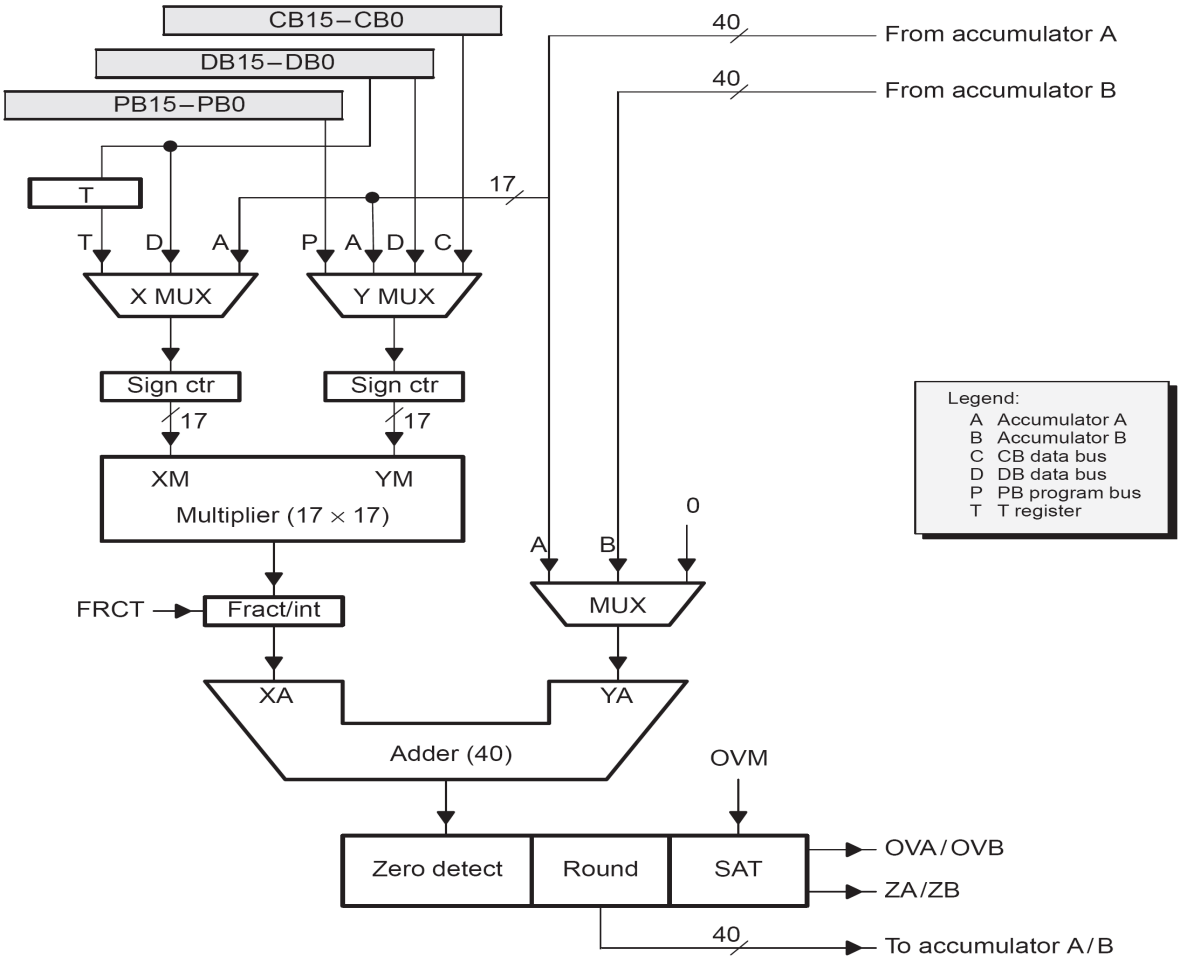



Fig 2 – 3 MULTIPLIER/ADDER UNIT

These are the some of the important parts of the processor and you are instructed to go through the detailed architecture once which helps you in developing the optimized code for the required application.

# DSP PROGRAMS IN C & MATLAB

## 1. Linear Convolution

**AIM:**

To verify Linear Convolution.

**EQUIPMENTS:**

| | |
|---|---|
| Operating System | – Windows XP |
| Constructor | - Simulator |
| Software | - CCStudio 3 & MATLAB 7.5 |

**THEORY:**

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two *numbers* and produces a third *numbe*r, while convolution takes two *signals* and produces a third *signa*l. Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal.

$$y(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k) \quad 0 < n < N-1 \qquad (1)$$

In this equation, x1(k), x2(n-k) and y(n) represent the input to and output from the system at time n. Here we could see that one of the input is shifted in time by a value every time it is multiplied with the other input signal. Linear Convolution is quite often used as a method of implementing filters of various types.

**PROGRAM:**

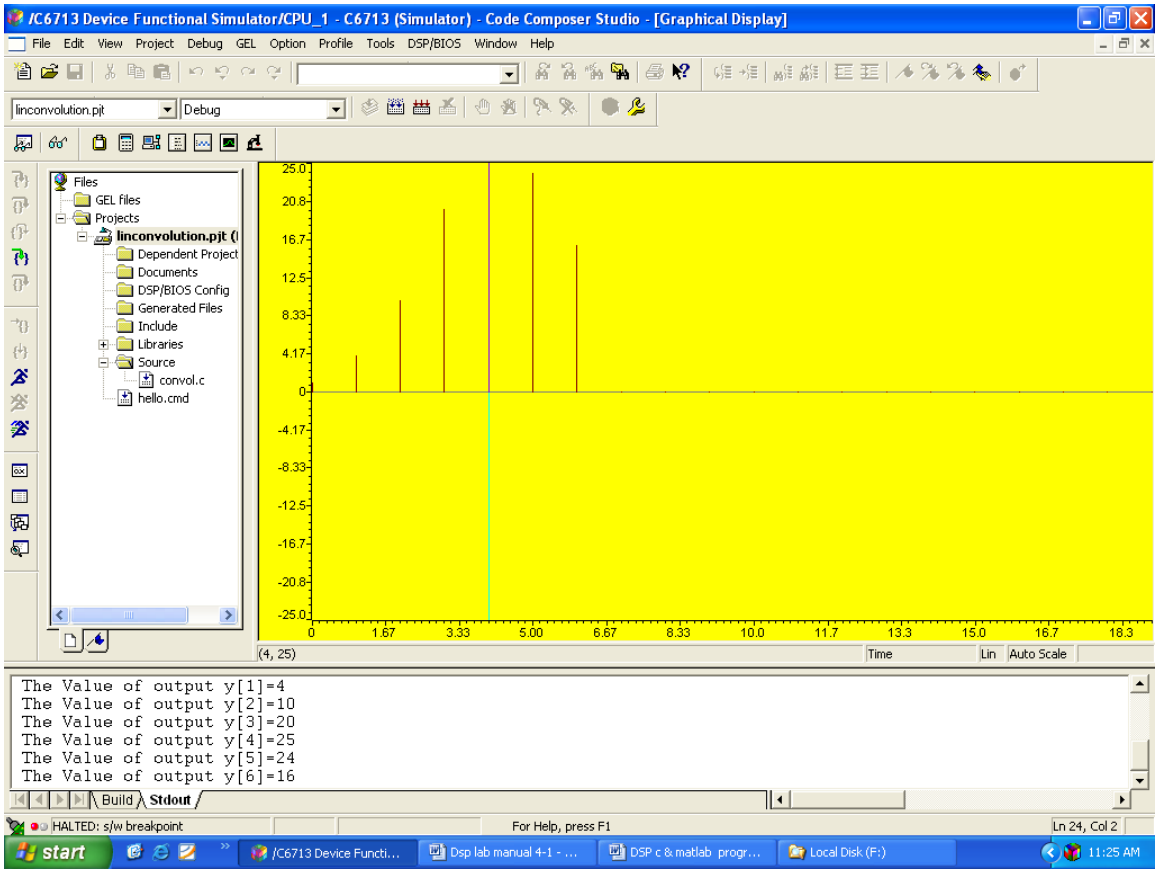
// Linear convolution program in c language using CCStudio

```
#include<stdio.h>
int x[15],h[15],y[15];
main()
{
int i,j,m,n;

printf("\n enter value for m");
scanf("%d",&m);
printf("\n enter value for n");
scanf("%d",&n);

printf("Enter values for i/p x(n):\n");
for(i=0;i<m;i++)
scanf("%d",&x[i]);
printf("Enter Values for i/p h(n) \n");
for(i=0;i<n; i++)
scanf("%d",&h[i]);
```

```
// padding of zeros
for(i=m;i<=m+n-1;i++)
x[i]=0;
for(i=n;i<=m+n-1;i++)
h[i]=0;
/*  convolution operation */
for(i=0;i<m+n-1;i++)
{
y[i]=0;
for(j=0;j<=i;j++)
{
y[i]=y[i]+(x[j]*h[i-j]);
}
}
//displaying the o/p
for(i=0;i<m+n-1;i++)
printf("\n The Value of output y[%d]=%d",i,y[i]);
}
```

**Result:**

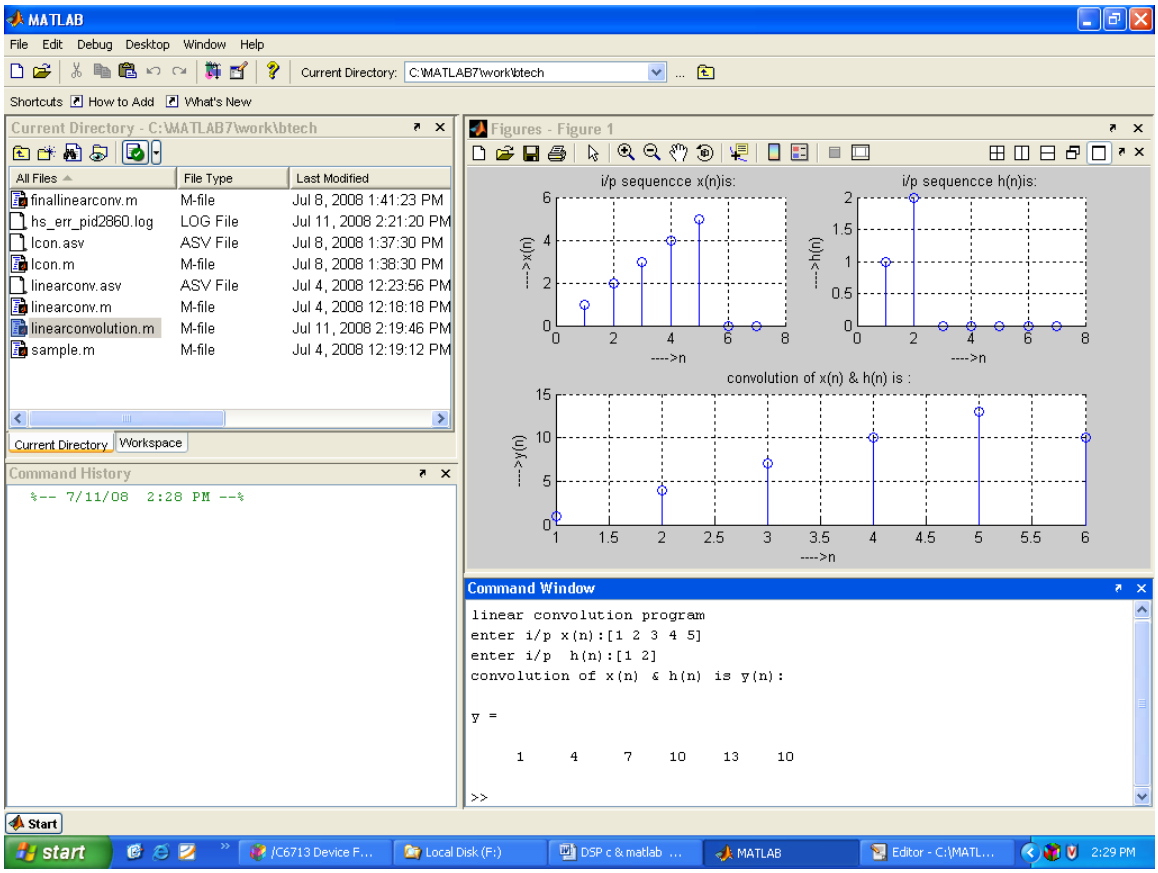
```
 enter value for m4

 enter value for n4
Enter values for i/p
1 2 3 4
Enter Values for n
1 2 3 4

 The Value of output y[0]=1
 The Value of output y[1]=4
 The Value of output y[2]=10
 The Value of output y[3]=20
 The Value of output y[4]=25
 The Value of output y[5]=24
 The Value of output y[6]=16
```


/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio - [Graphical Display]
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
linconvolution.pjt
Debug
Files
GEL files
Projects
linconvolution.pjt (
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
convol.c
hello.cmd
25.0
20.8
16.7
12.5
8.33
4.17
0
-4.17
-8.33
-12.5
-16.7
-20.8
-25.0
0
1.67
3.33
5.00
6.67
8.33
10.0
11.7
13.3
15.0
16.7
18.3
(4, 25)
Time
Lin
Auto Scale
The Value of output y[1]=4
The Value of output y[2]=10
The Value of output y[3]=20
The Value of output y[4]=25
The Value of output y[5]=24
The Value of output y[6]=16
Build
Stdout
HALTED: s/w breakpoint
For Help, press F1
Ln 24, Col 2
start
/C6713 Device Functi...
Dsp lab manual 4-1 - ...
DSP c & matlab progr...
Local Disk (F:)
11:25 AM

% MATLAB program for linear convolution

```
%linear convolution program

clc;
clear all;
close all;
disp('linear convolution program');

x=input('enter i/p x(n):');
m=length(x);
h=input('enter i/p  h(n):');
n=length(h);

x=[x,zeros(1,n)];
subplot(2,2,1), stem(x);
title('i/p sequence x(n)is:');
xlabel('---->n');
ylabel('---->x(n)');grid;

h=[h,zeros(1,m)];
subplot(2,2,2), stem(h);
title('i/p sequence h(n)is:');
xlabel('---->n');
ylabel('---->h(n)');grid;

disp('convolution of x(n) & h(n) is y(n):');

y=zeros(1,m+n-1);

for i=1:m+n-1
   y(i)=0;
   for j=1:m+n-1
      if(j<i+1)
      y(i)=y(i)+x(j)*h(i-j+1);
      end
   end
end
y
subplot(2,2,[3,4]),stem(y);
title('convolution of x(n) & h(n) is :');
xlabel('---->n');
ylabel('---->y(n)');grid;
```

## Result :


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\btech
Shortcuts How to Add What's New
Current Directory - C:\MATLAB7\work\btech
All Files File Type Last Modified
finallinearconv.m M-file Jul 8, 2008 1:41:23 PM
hs_err_pid2860.log LOG File Jul 11, 2008 2:21:20 PM
Icon.asv ASV File Jul 8, 2008 1:37:30 PM
Icon.m M-file Jul 8, 2008 1:38:30 PM
linearconv.asv ASV File Jul 4, 2008 12:23:56 PM
linearconv.m M-file Jul 4, 2008 12:18:18 PM
linearconvolution.m M-file Jul 11, 2008 2:19:46 PM
sample.m M-file Jul 4, 2008 12:19:12 PM
Current Directory
Workspace
Command History
%-- 7/11/08 2:28 PM --%
Figures - Figure 1
i/p sequencce x(n)is:
i/p sequencce h(n)is:
convolution of x(n) & h(n) is :
----->x(n)
----->h(n)
----->y(n)
----->n
Command Window
linear convolution program
enter i/p x(n):[1 2 3 4 5]
enter i/p h(n):[1 2]
convolution of x(n) & h(n) is y(n):
y =
1 4 7 10 13 10
>>
Start
start
/C6713 Device F...
Local Disk (F:)
DSP c & matlab ...
MATLAB
Editor - C:\MATL...
2:29 PM

# 2. **Circular Convolution**

## AIM

To verify Circular Convolution.

## EQUIPMENTS:

Operating System – Windows XP
Constructor - Simulator
Software - CCStudio 3 & MATLAB 7.5

## THEORY

Circular convolution is another way of finding the convolution sum of two input signals. It resembles the linear convolution, except that the sample values of one of the input signals is folded and right shifted before the convolution sum is found. Also note that circular convolution could also be found by taking the DFT of the two input signals and finding the product of the two frequency domain signals. The Inverse DFT of the product would give the output of the signal in the time domain which is the circular convolution output. The two input signals could have been of varying sample lengths. But we take the DFT of higher point, which ever signals levels to. For eg. If one of the signal is of length 256 and the other spans 51 samples, then we could only take 256 point DFT. So the output of IDFT would be containing 256 samples instead of 306 samples, which follows N1+N2 – 1 where N1 & N2 are the lengths 256 and 51 respectively of the two inputs. Thus the output which should have been 306 samples long is fitted into 256 samples. The 256 points end up being a distorted version of the correct signal. This process is called circular convolution.

## PROGRAM:

/* program to implement circular convolution */

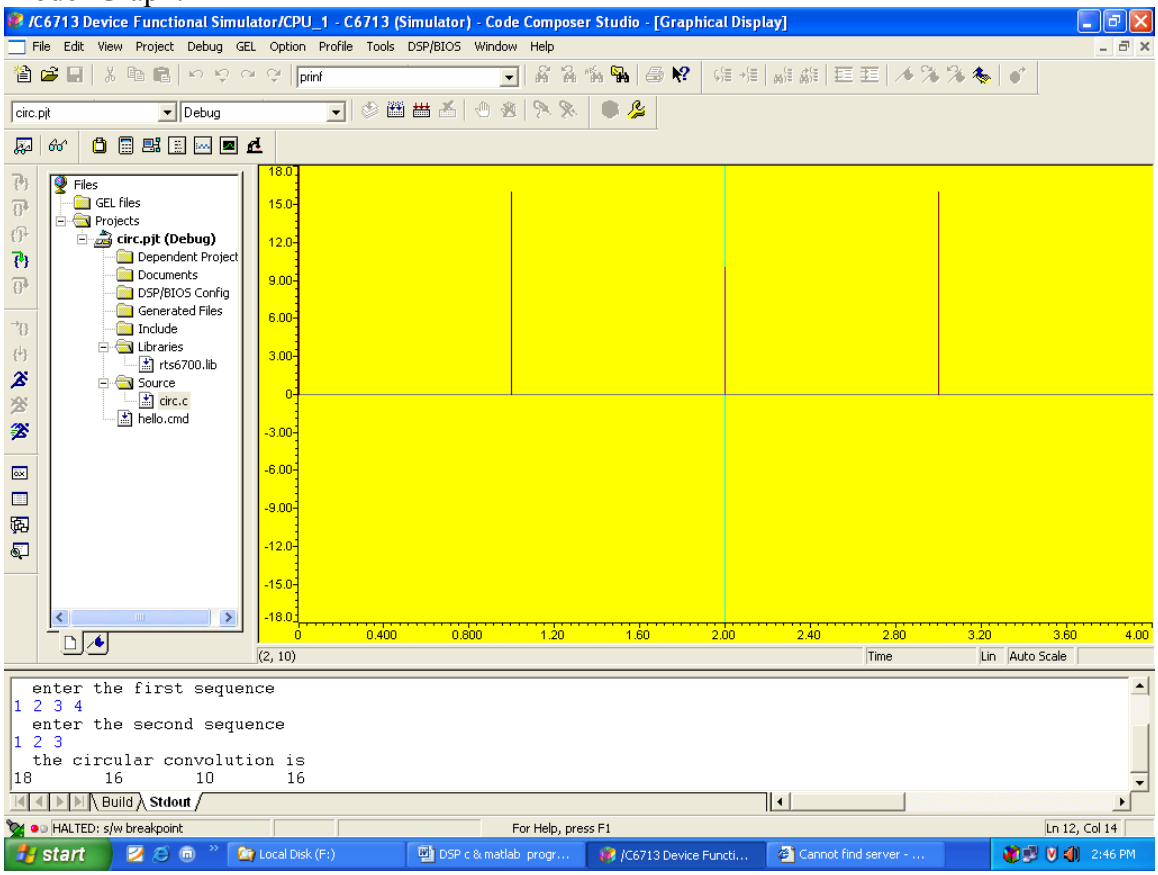
```
#include<stdio.h>
 int m,n,x[30],h[30],y[30],i,j, k,x2[30],a[30];
void main()
{
  printf("  Enter the length of the first sequence\n");
  scanf("%d",&m);
  printf("  Enter the length of the second sequence\n");
  scanf("%d",&n);
  printf("  Enter the first sequence\n");
  for(i=0;i<m;i++)
  scanf("%d",&x[i]);
  printf("  Enter the second sequence\n");
  for(j=0;j<n;j++)
  scanf("%d",&h[j]);
```

```
if(m-n!=0)                          /*If length of both sequences are not equal*/
  {
            if(m>n)                 /* Pad the smaller sequence with zero*/
            {
             for(i=n;i<m;i++)
             h[i]=0;
             n=m;
            }
            for(i=m;i<n;i++)
            x[i]=0;
            m=n;
  }
     y[0]=0;
     a[0]=h[0];
    for(j=1;j<n;j++)                                  /*folding h(n) to h(-n)*/
    a[j]=h[n-j];

 /*Circular convolution*/
   for(i=0;i<n;i++)
     y[0]+=x[i]*a[i];
   for(k=1;k<n;k++)
   {
     y[k]=0;
       /*circular shift*/
     for(j=1;j<n;j++)
            x2[j]=a[j-1];

     x2[0]=a[n-1];
     for(i=0;i<n;i++)
     {
              a[i]=x2[i];
              y[k]+=x[i]*x2[i];
     }
   }

/*displaying the result*/
    printf("  The circular convolution is\n");
     for(i=0;i<n;i++)
    printf("%d \t",y[i]);
 }
```

OUTPUT:-

```
 Enter the length of the first sequence
4
 Enter the length of the second sequence
3
 Enter the first sequence
1 2 3 4
 Enter the second sequence
1 2 3
 The circular convolution is
18      16      10      16
```

Model Graph:-

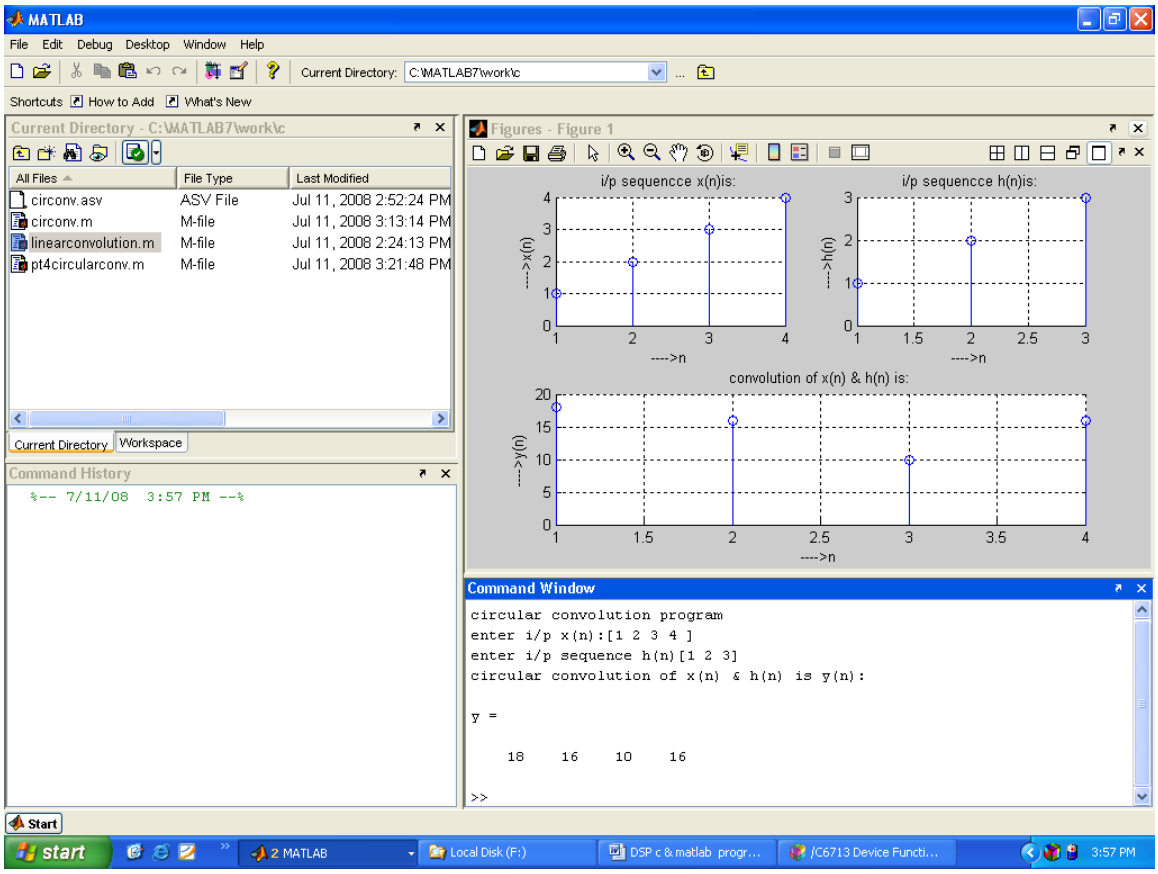


## %circular convolution program

```
clc;
clear all;
close all;
disp('circular convolution program');

x=input('enter i/p x(n):');
m=length(x);
h=input('enter i/p sequence h(n)');
n=length(h);

subplot(2,2,1), stem(x);
title('i/p sequencce x(n)is:');
xlabel('---->n');
ylabel('---->x(n)');grid;

subplot(2,2,2), stem(h);
title('i/p sequencce h(n)is:');
xlabel('---->n');
ylabel('---->h(n)');grid;

disp('circular convolution of x(n) & h(n) is y(n):');
if(m-n~=0)
   if(m>n)
      h=[h,zeros(1,m-n)];
      n=m;
   end
      x=[x,zeros(1,n-m)];
      m=n;
end

y=zeros(1,n);
y(1)=0;
a(1)=h(1);

for j=2:n
   a(j)=h(n-j+2);
end

%ciruclar conv
for i=1:n
   y(1)=y(1)+x(i)*a(i);
   end

for k=2:n
   y(k)=0;
   % circular shift
   for j=2:n
      x2(j)=a(j-1);
```

```
    end
    x2(1)=a(n);

    for i=1:n
        if(i<n+1)
            a(i)=x2(i);
        y(k)=y(k)+x(i)*a(i);
        end
    end
end
y
subplot(2,2,[3,4]),stem(y);
title('convolution of x(n) & h(n) is:');
xlabel('---->n');
ylabel('---->y(n)');grid;
```

**Result :**



# 3. **FIR filters**

**AIM**

To verify FIR filters.

**EQUIPMENTS:**

| | |
|---|---|
| Operating System | – Windows XP |
| Constructor | - Simulator |
| Software | - CCStudio 3 & MATLAB 7.5 |

**THEORY:**

A Finite Impulse Response (FIR) filter is a discrete linear time-invariant system whose output is based on the weighted summation of a finite number of past inputs. An FIR transversal filter structure can be obtained directly from the equation for discrete-time convolution.

$$y(n) = \sum_{k=0}^{N-1} x(k)h(n-k) \quad 0 < n < N-1 \qquad \textbf{(1)}$$

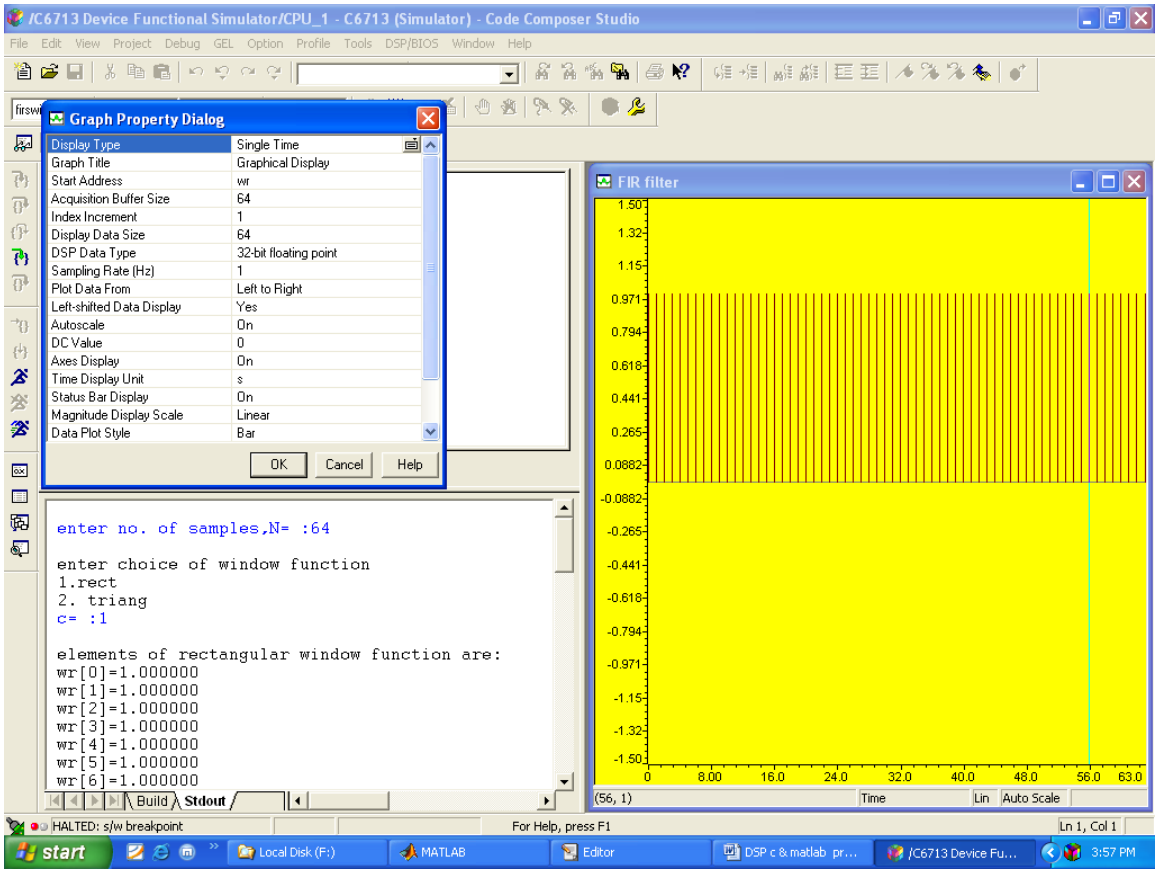
In this equation, x(k) and y(n) represent the input to and output from the filter at time n. h(n-k) is the transversal filter coefficients at time n. These coefficients are generated by using FDS (Filter Design Software or Digital filter design package).

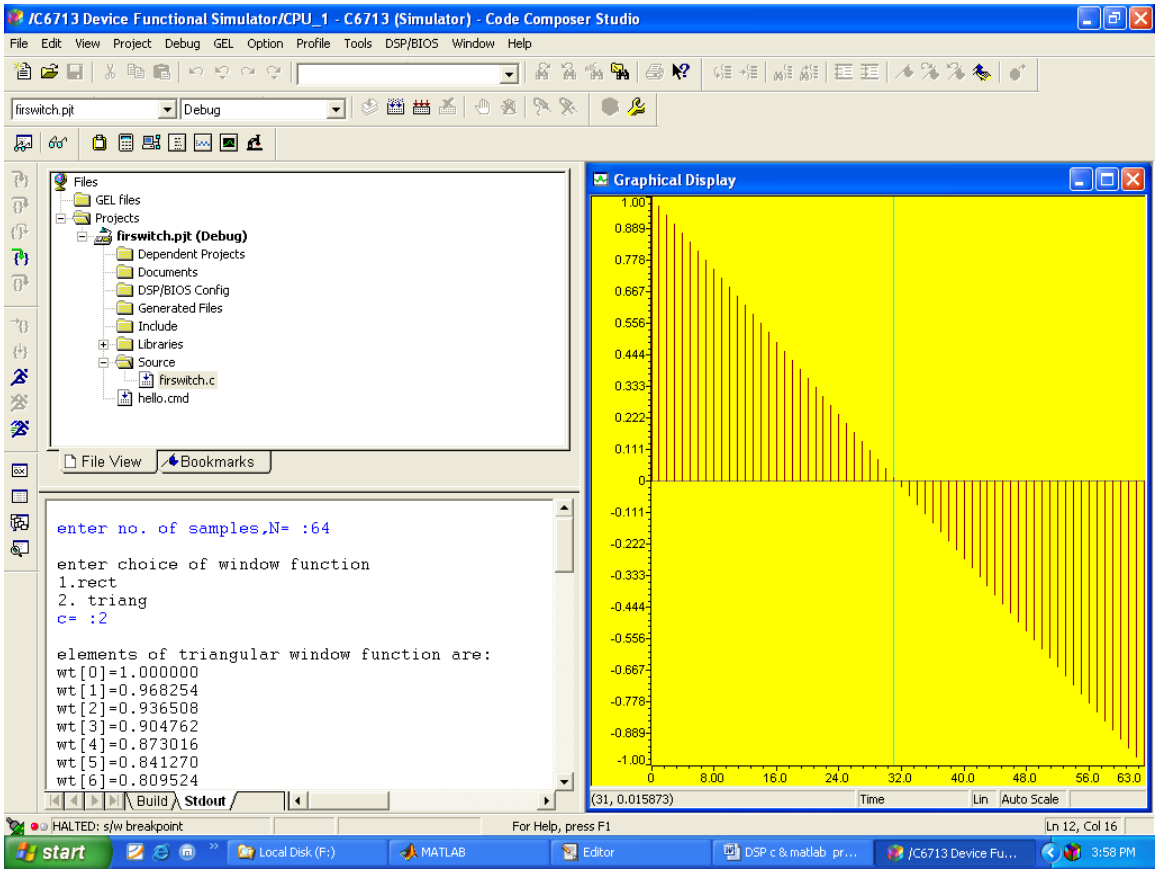
FIR – filter is a finite impulse response filter. Order of the filter should be specified. Infinite response is truncated to get finite impulse response. placing a window of finite length does this. Types of windows available are Rectangular, Barlett, Hamming, Hanning, Blackmann window etc. This FIR filter is an all zero filter.

**PROGRAM:**

```
#include<stdio.h>
#include<math.h>
#define pi 3.1415
int n,N,c;
float wr[64],wt[64];
void main()
{
printf("\n enter no. of samples,N= :");
scanf("%d",&N);

printf("\n enter choice of window function\n 1.rect  \n 2. triang \n c= :");
scanf("%d",&c);
printf("\n elements of window function are:");
switch(c)
{
case 1:
for(n=0;n<=N-1;n++)
{
wr[n]=1;
printf(" \n wr[%d]=%f",n,wr[n]);
}
break;
case 2:
for(n=0;n<=N-1;n++)
{
wt[n]=1-(2*(float)n/(N-1));
printf("\n wt[%d]=%f",n,wt[n]);
}
break;

}
}
```

**RESULT:**


/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Graph Property Dialog
Display Type Single Time
Graph Title Graphical Display
Start Address wr
Acquisition Buffer Size 64
Index Increment 1
Display Data Size 64
DSP Data Type 32-bit floating point
Sampling Rate (Hz) 1
Plot Data From Left to Right
Left-shifted Data Display Yes
Autoscale On
DC Value 0
Axes Display On
Time Display Unit s
Status Bar Display On
Magnitude Display Scale Linear
Data Plot Style Bar
OK Cancel Help
FIR filter
1.50
1.32
1.15
0.971
0.794
0.618
0.441
0.265
0.0882
-0.0882
-0.265
-0.441
-0.618
-0.794
-0.971
-1.15
-1.32
-1.50
0 8.00 16.0 24.0 32.0 40.0 48.0 56.0 63.0
enter no. of samples,N= :64
enter choice of window function
1.rect
2. triang
c= :1
elements of rectangular window function are:
wr[0]=1.000000
wr[1]=1.000000
wr[2]=1.000000
wr[3]=1.000000
wr[4]=1.000000
wr[5]=1.000000
wr[6]=1.000000
Build Stdout
(56, 1) Time Lin Auto Scale
HALTED: s/w breakpoint
For Help, press F1
Ln 1, Col 1
start
Local Disk (F:)
MATLAB
Editor
DSP c & matlab pr...
/C6713 Device Fu...
3:57 PM

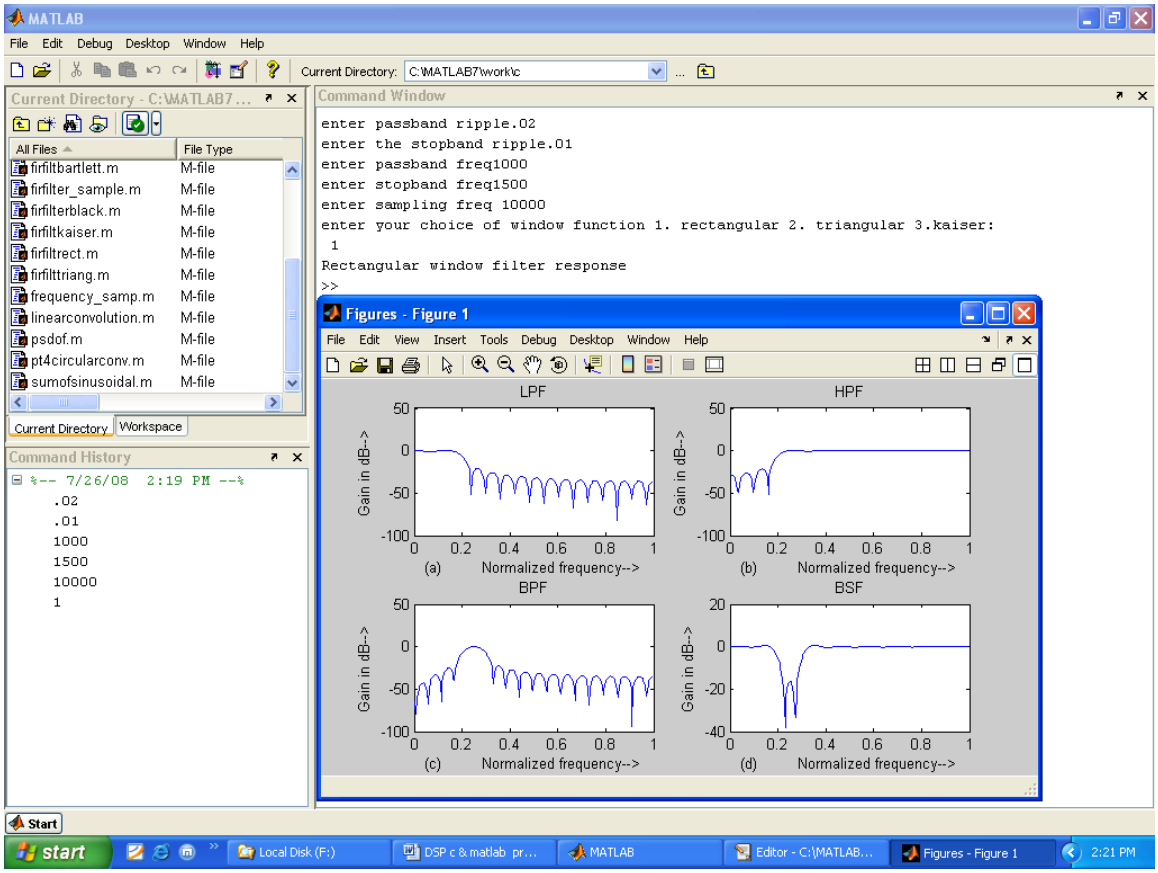

/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
firswitch.pjt
Debug
Files
GEL files
Projects
firswitch.pjt (Debug)
Dependent Projects
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
firswitch.c
hello.cmd
File View
Bookmarks
enter no. of samples,N= :64
enter choice of window function
1.rect
2. triang
c= :2
elements of triangular window function are:
wt[0]=1.000000
wt[1]=0.968254
wt[2]=0.936508
wt[3]=0.904762
wt[4]=0.873016
wt[5]=0.841270
wt[6]=0.809524
Build
Stdout
Graphical Display
1.00
0.889
0.778
0.667
0.556
0.444
0.333
0.222
0.111
0
-0.111
-0.222
-0.333
-0.444
-0.556
-0.667
-0.778
-0.889
-1.00
0
8.00
16.0
24.0
32.0
40.0
48.0
56.0
63.0
(31, 0.015873)
Time
Lin
Auto Scale
HALTED: s/w breakpoint
For Help, press F1
Ln 12, Col 16
start
Local Disk (F:)
MATLAB
Editor
DSP c & matlab pr...
/C6713 Device Fu...
3:58 PM

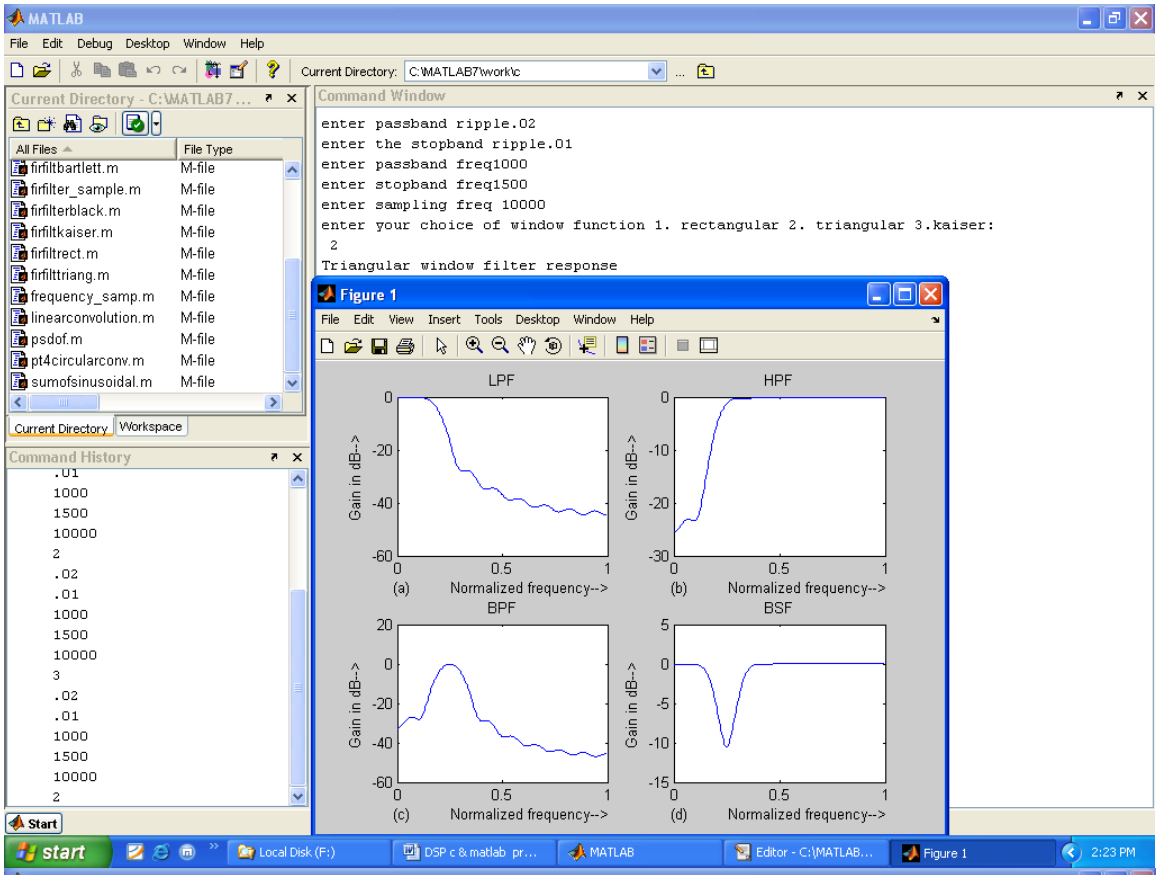
**PROGRAM:**

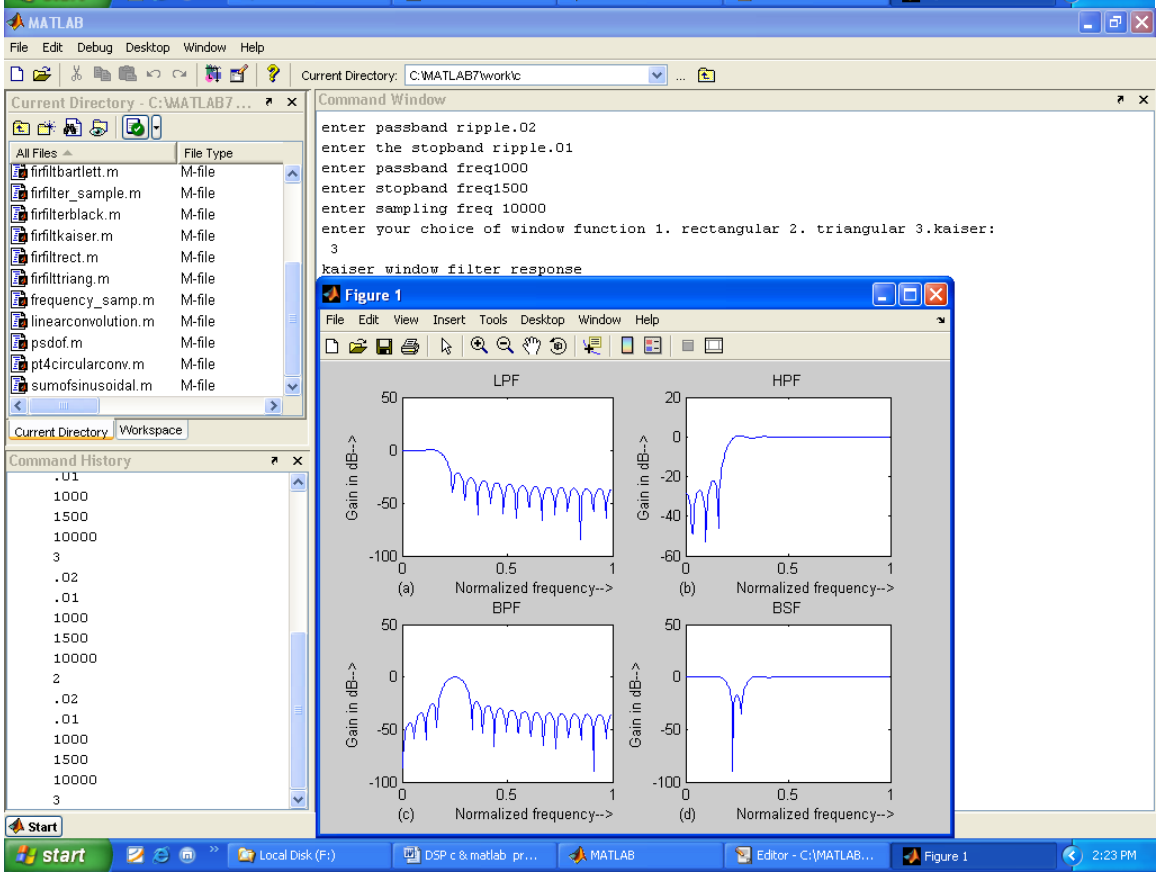
```
%fir filt design  window techniques
clc;
clear all;
close all;

rp=input('enter passband ripple');
rs=input('enter the stopband ripple');
fp=input('enter passband freq');
fs=input('enter stopband freq');
f=input('enter sampling freq ');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2)~=0)
    n1=n;
    n=n-1;
end
c=input('enter your choice of window function 1. rectangular 2. triangular 3.kaiser: \n ');
if(c==1)
    y=rectwin(n1);
    disp('Rectangular window filter response');
end
if (c==2)
    y=triang(n1);
   disp('Triangular window filter response');
end
 if(c==3)
     y=kaiser(n1);
   disp('kaiser window filter response');
 end

%LPF
b=fir1(n,wp,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);plot(o/pi,m);
title('LPF');
ylabel('Gain in dB-->');
xlabel('(a)        Normalized frequency-->');
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b)        Normalized frequency-->');
```

```
%BPF
wn=[wp ws];
b=fir1(n,wn,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);plot(o/pi,m);
title('BPF');
ylabel('Gain in dB-->');
xlabel('(c)        Normalized frequency-->');
%BSF
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);plot(o/pi,m);
title('BSF');
ylabel('Gain in dB-->');
xlabel('(d)        Normalized frequency-->');
```

**RESULT:**


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
firfiltbartlett.m M-file
firfilter_sample.m M-file
firfilterblack.m M-file
firfiltkaiser.m M-file
firfiltrect.m M-file
firfilttriang.m M-file
frequency_samp.m M-file
linearconvolution.m M-file
psdof.m M-file
pt4circularconv.m M-file
sumofsinusoidal.m M-file
Current Directory
Workspace
Command History
%-- 7/26/08 2:19 PM --%
.02
.01
1000
1500
10000
1
Command Window
enter passband ripple.02
enter the stopband ripple.01
enter passband freq1000
enter stopband freq1500
enter sampling freq 10000
enter your choice of window function 1. rectangular 2. triangular 3.kaiser:
1
Rectangular window filter response
>>
Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
LPF
HPF
BPF
BSF
Gain in dB-->
Normalized frequency-->
(a)
(b)
(c)
(d)
Start
start
Local Disk (F:)
DSP c & matlab pr...
MATLAB
Editor - C:\MATLAB...
Figures - Figure 1
2:21 PM


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
firfiltbartlett.m M-file
firfilter_sample.m M-file
firfilterblack.m M-file
firfiltkaiser.m M-file
firfiltrect.m M-file
firfilttriang.m M-file
frequency_samp.m M-file
linearconvolution.m M-file
psdof.m M-file
pt4circularconv.m M-file
sumofsinusoidal.m M-file
Current Directory
Workspace
Command History
.01
1000
1500
10000
2
.02
.01
1000
1500
10000
3
.02
.01
1000
1500
10000
2
Command Window
enter passband ripple.02
enter the stopband ripple.01
enter passband freq1000
enter stopband freq1500
enter sampling freq 10000
enter your choice of window function 1. rectangular 2. triangular 3.kaiser:
2
Triangular window filter response
Figure 1
File Edit View Insert Tools Desktop Window Help
LPF
HPF
BPF
BSF
Gain in dB-->
Normalized frequency-->
(a)
(b)
(c)
(d)
Start
start
Local Disk (F:)
DSP c & matlab pr...
MATLAB
Editor - C:\MATLAB...
Figure 1
2:23 PM


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
firfiltbartlett.m M-file
firfilter_sample.m M-file
firfilterblack.m M-file
firfiltkaiser.m M-file
firfiltrect.m M-file
firfilttriang.m M-file
frequency_samp.m M-file
linearconvolution.m M-file
psdof.m M-file
pt4circularconv.m M-file
sumofsinusoidal.m M-file
Current Directory
Workspace
Command History
.01
1000
1500
10000
3
.02
.01
1000
1500
10000
2
.02
.01
1000
1500
10000
3
Command Window
enter passband ripple.02
enter the stopband ripple.01
enter passband freq1000
enter stopband freq1500
enter sampling freq 10000
enter your choice of window function 1. rectangular 2. triangular 3.kaiser:
3
kaiser window filter response
Figure 1
File Edit View Insert Tools Desktop Window Help
LPF
HPF
BPF
BSF
Gain in dB-->
Normalized frequency-->
(a)
(b)
(c)
(d)
Start
start
Local Disk (F:)
DSP c & matlab pr...
MATLAB
Editor - C:\MATLAB...
Figure 1
2:23 PM

# 4. **IIR filters**

**AIM**

To design and implement IIR (LPF/HPF)filters.

**EQUIPMENTS:**

Operating System – Windows XP
Constructor - Simulator
Software - CCStudio 3 & MATLAB 7.5

**THEORY:**

The IIR filter can realize both the poles and zeroes of a system because it has a rational transfer function, described by polynomials in z in both the numerator and the denominator:

$$H(z)\frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=1}^{N} a_k Z^{-k}} \qquad (2)$$

The difference equation for such a system is described by the following:

$$y(n)=\sum_{k=0}^{M} b_k x(n-k) \; + \; \sum_{k=1}^{N} a_k y(n-k) \qquad (3)$$

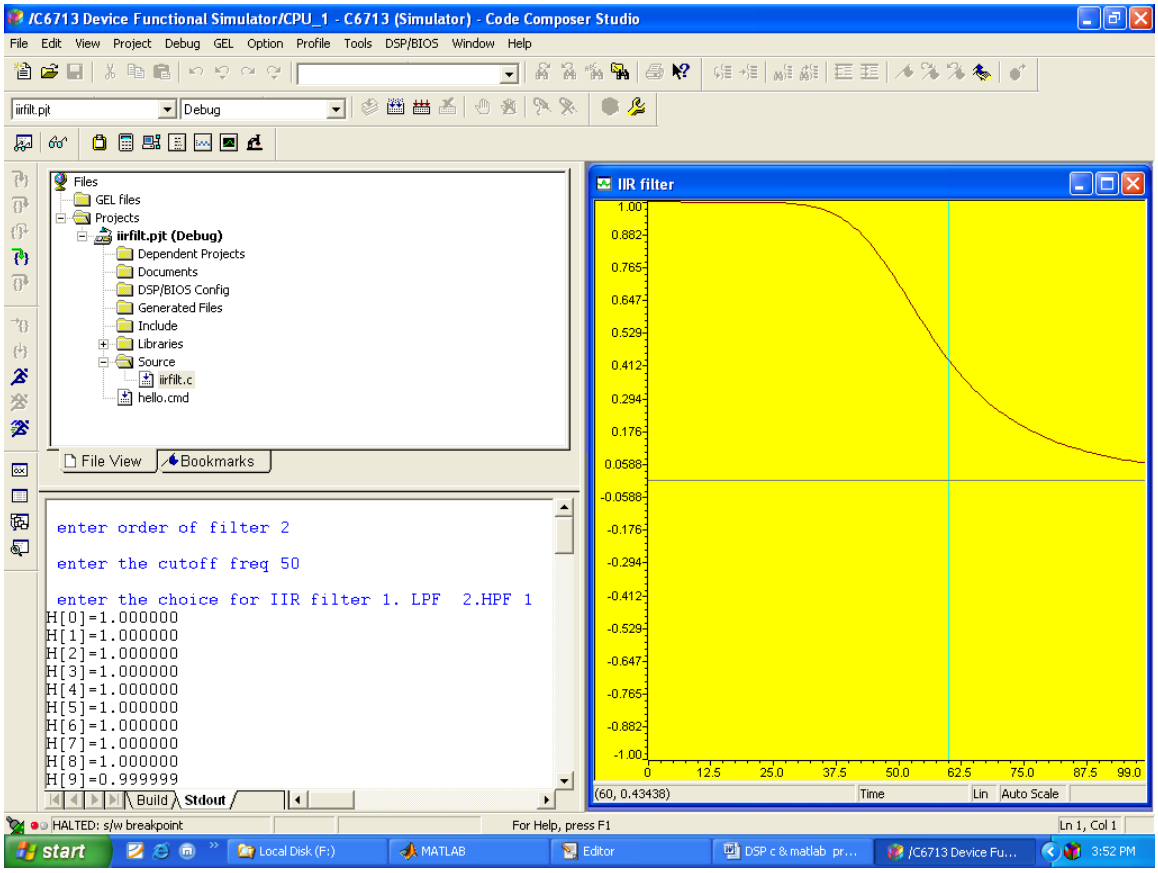
M and N are order of the two polynomials
$b_k$ and $a_k$ are the filter coefficients. These filter coefficients are generated using FDS (Filter Design software or Digital Filter design package).

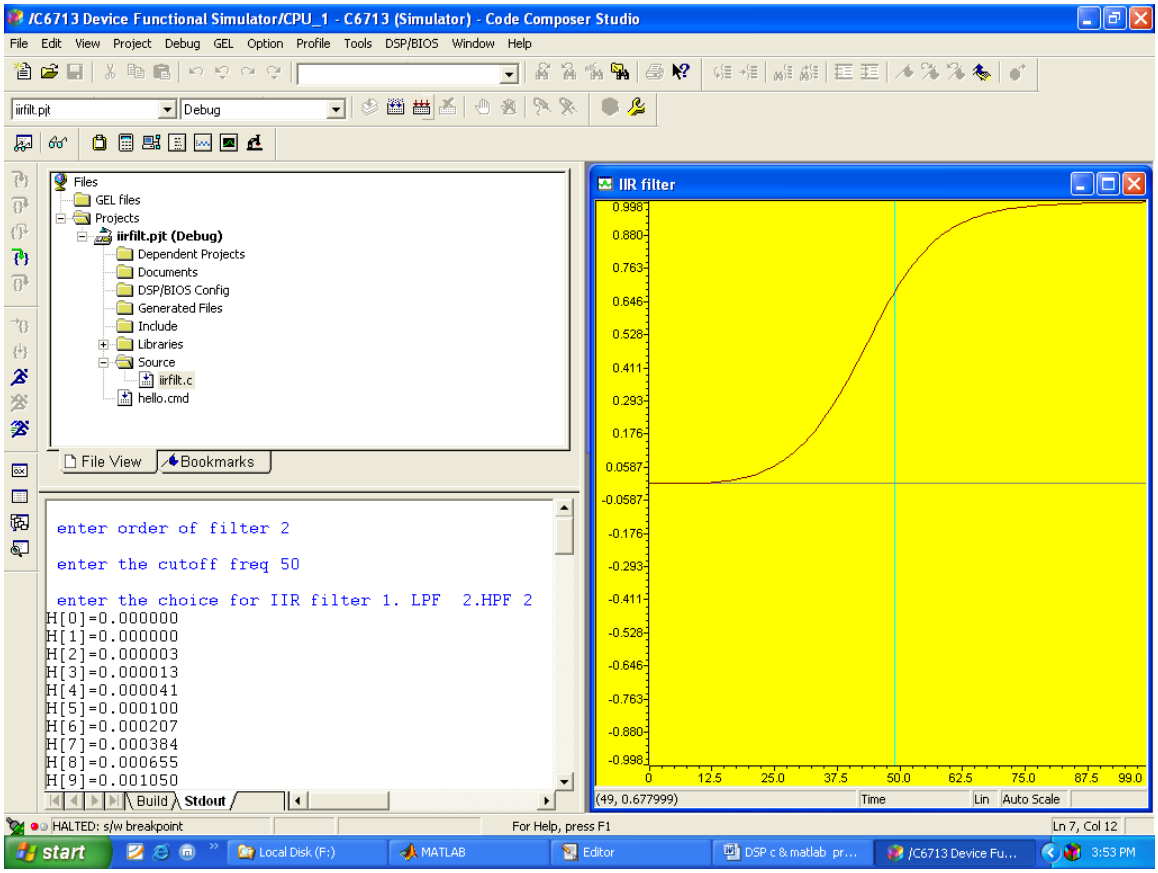
IIR filters can be expanded as infinite impulse response filters. In designing IIR filters, cutoff frequencies of the filters should be mentioned. The order of the filter can be estimated using butter worth polynomial. That's why the filters are named as butter worth filters. Filter coefficients can be found and the response can be plotted.

**PROGRAM**:

```
//iirfilters
#include<stdio.h>
#include<math.h>
int i,w,wc,c,N;
float H[100];
float mul(float, int);
void main()
{
printf("\n enter order of filter ");
scanf("%d",&N);
printf("\n enter the cutoff freq ");
scanf("%d",&wc);
printf("\n enter the choice for IIR filter 1. LPF  2.HPF ");
scanf("%d",&c);
switch(c)
{
case 1:
for(w=0;w<100;w++)
{
H[w]=1/sqrt(1+mul((w/(float)wc),2*N));
printf("H[%d]=%f\n",w,H[w]);
}
break;
    case 2:
 for(w=0;w<=100;w++)
{
H[w]=1/sqrt(1+mul((float)wc/w,2*N));
printf("H[%d]=%f\n",w,H[w]);
}
break;
}
}
float mul(float a,int x)
{
for(i=0;i<x-1;i++)
a*=a;
return(a);
}
```

**RESULT :**

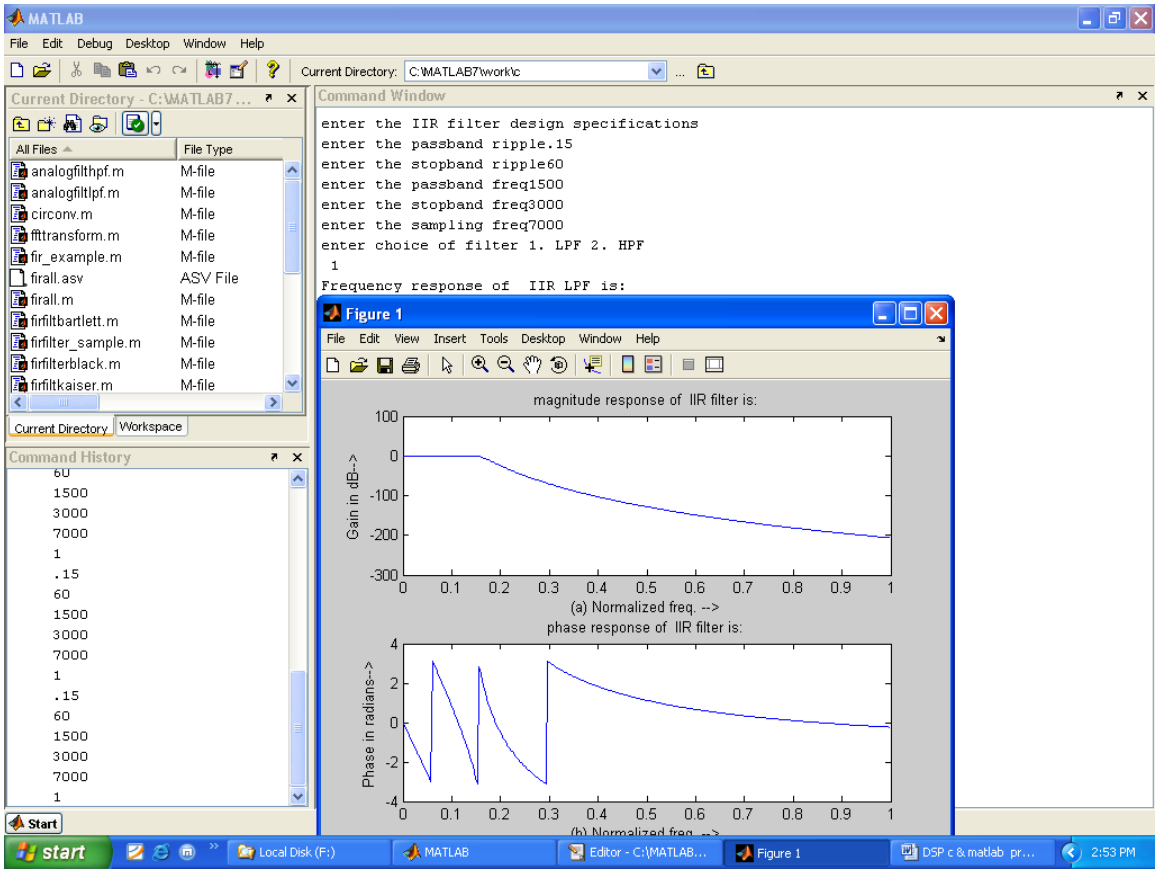

/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
iirfilt.pjt
Debug
Files
GEL files
Projects
iirfilt.pjt (Debug)
Dependent Projects
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
iirfilt.c
hello.cmd
File View
Bookmarks
enter order of filter 2
enter the cutoff freq 50
enter the choice for IIR filter 1. LPF 2.HPF 1
H[0]=1.000000
H[1]=1.000000
H[2]=1.000000
H[3]=1.000000
H[4]=1.000000
H[5]=1.000000
H[6]=1.000000
H[7]=1.000000
H[8]=1.000000
H[9]=0.999999
Build
Stdout
IIR filter
1.00
0.882
0.765
0.647
0.529
0.412
0.294
0.176
0.0588
-0.0588
-0.176
-0.294
-0.412
-0.529
-0.647
-0.765
-0.882
-1.00
0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
99.0
(60, 0.43438)
Time
Lin
Auto Scale
HALTED: s/w breakpoint
For Help, press F1
Ln 1, Col 1
start
Local Disk (F:)
MATLAB
Editor
DSP c & matlab pr...
/C6713 Device Fu...
3:52 PM

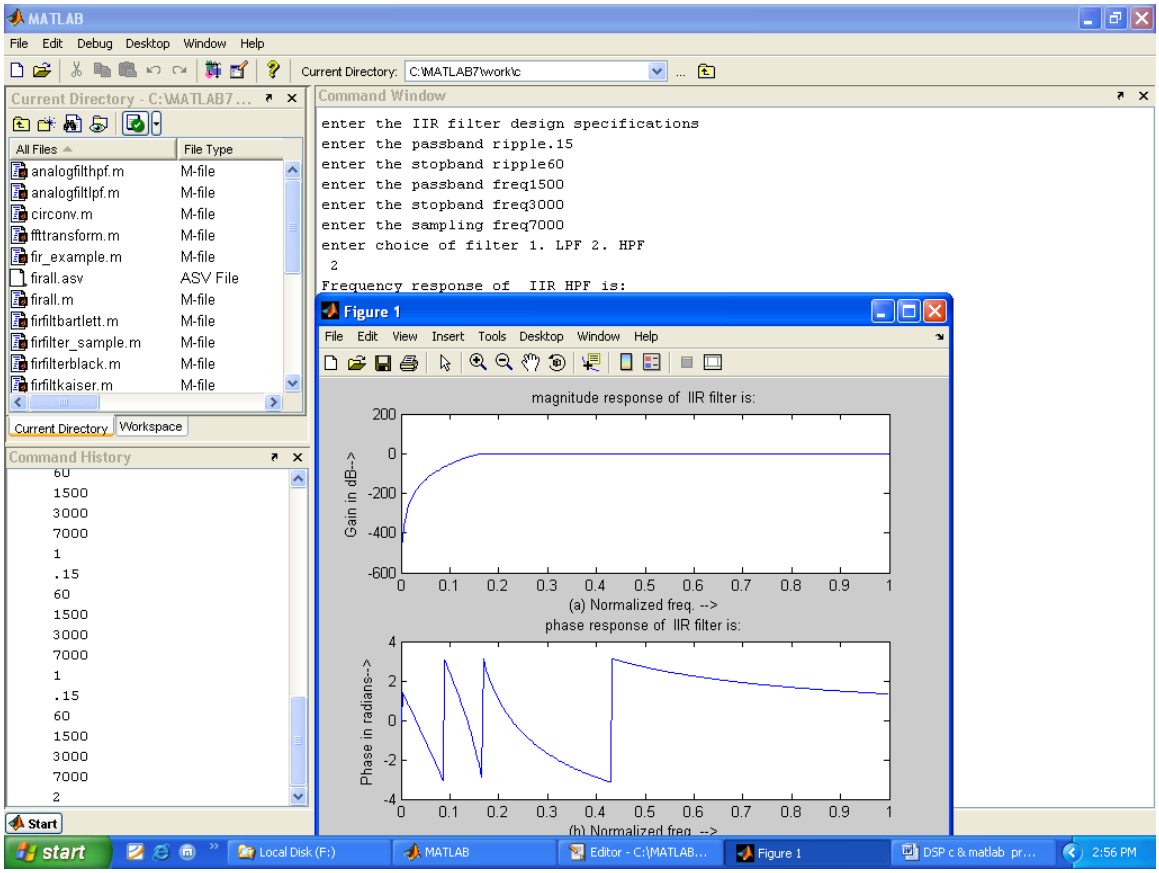

/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
iirfilt.pjt
Debug
Files
GEL files
Projects
iirfilt.pjt (Debug)
Dependent Projects
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
iirfilt.c
hello.cmd
File View
Bookmarks
enter order of filter 2
enter the cutoff freq 50
enter the choice for IIR filter 1. LPF 2.HPF 2
H[0]=0.000000
H[1]=0.000000
H[2]=0.000003
H[3]=0.000013
H[4]=0.000041
H[5]=0.000100
H[6]=0.000207
H[7]=0.000384
H[8]=0.000655
H[9]=0.001050
Build
Stdout
IIR filter
0.998
0.880
0.763
0.646
0.528
0.411
0.293
0.176
0.0587
-0.0587
-0.176
-0.293
-0.411
-0.528
-0.646
-0.763
-0.880
-0.998
0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
99.0
(49, 0.677999)
Time
Lin
Auto Scale
HALTED: s/w breakpoint
For Help, press F1
Ln 7, Col 12
start
Local Disk (F:)
MATLAB
Editor
DSP c & matlab pr...
/C6713 Device Fu...
3:53 PM

**PROGRAM :**

```
%  IIR filters LPF & HPF
clc;
clear all;
close all;

disp('enter the IIR filter design specifications');
rp=input('enter the passband ripple');
rs=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
fs=input('enter the sampling freq');
w1=2*wp/fs;w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs,'s');
c=input('enter choice of filter 1. LPF 2. HPF \n ');
if(c==1)
   disp('Frequency response of  IIR LPF is:');
[b,a]=butter(n,wn,'low','s');
end
if(c==2)
   disp('Frequency response of  IIR HPF is:');
[b,a]=butter(n,wn,'high','s');
end
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure,subplot(2,1,1);plot(om/pi,m);
title('magnitude response of  IIR filter is:');
xlabel('(a) Normalized freq. -->');
ylabel('Gain in dB-->');
subplot(2,1,2);plot(om/pi,an);
title('phase response of  IIR filter is:');
xlabel('(b) Normalized freq. -->');
ylabel('Phase in radians-->');
```

**RESULT :**


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
analogfilthpf.m M-file
analogfiltlpf.m M-file
circonv.m M-file
ffttransform.m M-file
fir_example.m M-file
firall.asv ASV File
firall.m M-file
firfiltbartlett.m M-file
firfilter_sample.m M-file
firfilterblack.m M-file
firfiltkaiser.m M-file
Current Directory
Workspace
Command History
60
1500
3000
7000
1
.15
60
1500
3000
7000
1
.15
60
1500
3000
7000
1
Command Window
enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq1500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
1
Frequency response of IIR LPF is:
Figure 1
File Edit View Insert Tools Desktop Window Help
magnitude response of IIR filter is:
Gain in dB-->
100
0
-100
-200
-300
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->
4
2
0
-2
-4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Start
start
Local Disk (F:)
MATLAB
Editor - C:\MATLAB...
Figure 1
DSP c & matlab pr...
2:53 PM


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
analogfilthpf.m M-file
analogfiltlpf.m M-file
circonv.m M-file
ffttransform.m M-file
fir_example.m M-file
firall.asv ASV File
firall.m M-file
firfiltbartlett.m M-file
firfilter_sample.m M-file
firfilterblack.m M-file
firfiltkaiser.m M-file
Current Directory
Workspace
Command History
60
1500
3000
7000
1
.15
60
1500
3000
7000
1
.15
60
1500
3000
7000
2
Command Window
enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq1500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
2
Frequency response of IIR HPF is:
Figure 1
File Edit View Insert Tools Desktop Window Help
magnitude response of IIR filter is:
Gain in dB-->
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->
(b) Normalized freq. -->
Start
start
Local Disk (F:)
MATLAB
Editor - C:\MATLAB...
Figure 1
DSP c & matlab pr...
2:56 PM

# 5. Fast Fourier Transform

**AIM:**

To verify Fast Fourier Transform.

**EQUIPMENTS:**

Operating System – Windows XP
Constructor - Simulator
Software - CCStudio 3 & MATLAB 7.5

**THEORY:**

The Fast Fourier Transform is useful to map the time-domain sequence into a continuous function of a frequency variable. The FFT of a sequence {x(n)} of length N is given by a complex-valued sequence X(k).

$$X(k) = \sum_{k=0}^{M} x(n) \quad e^{-j2\pi\frac{nk}{n}} ; 0 < k < N-1$$

The above equation is the mathematical representation of the DFT. As the number of computations involved in transforming a N point time domain signal into its corresponding frequency domain signal was found to be $N^2$ complex multiplications, an alternative algorithm involving lesser number of computations is opted.

When the sequence x(n) is divided into 2 sequences and the DFT performed separately, the resulting number of computations would be $N^2/2$

(i.e.)

$$x(k) = \sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N^2}{2-1}} \quad x(2n+1) \quad W_N^{(2n+1)k} \quad (6)$$

Consider x(2n) be the even sample sequences and x(2n+1) be the odd sample sequence derived form x(n).

$$\sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk}$$

would result in (7)

$(N/2)^2$ multiplication's $\sum_{n=0}^{\frac{N^2}{2}-1} x(2n+1) \quad W_N^{(2n+1)k}$ (8)

an other $(N/2)^2$ multiplication's finally resulting in $(N/2)^2 + (N/2)^2$

$= \frac{N^2}{4} + \frac{N^2}{4} = \frac{N^2}{2}$ *Computations*

Further solving Eg. (2)

$$x(k) = \sum_{n=0}^{\frac{N^2}{2}-1} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) \quad W_N^{(2nk)} \, W_N^k \quad (9)$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) \quad W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) \quad W_N^{(2nk)} \quad (10)$$

Dividing the sequence x(2n) into further 2 odd and even sequences would reduce the computations.

$W_N \rightarrow$ is the twiddle factor

$$= e^{\frac{-j2\pi}{n}}$$

$$W_N^{nk} = e^{\left(\frac{-j2\pi}{n}\right)nk}$$

$$W_N^{\left(K+\frac{N}{2}\right)} = W_N \quad W_N^{\left(K+\frac{N}{2}\right)} \quad (11)$$

$$= e^{\frac{-j2\pi}{n}k} \quad e^{\frac{-j2\pi}{n}\frac{n}{2}}$$

$$= W_N^k \quad e^{\frac{-j2\pi}{n}k}$$

$$= W_N^k \quad (\cos\pi - j\sin\pi)$$

$$= W_N^{\left(K+\frac{N}{2}\right)} = W_N^k(-1)$$

$$= W_N^{\left(K+\frac{N}{2}\right)} = W_N^k \qquad (12)$$

Employing this equation, we deduce

$$x(k) = \sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N}{2-1}} x(2n+1) \quad W_N^{(2nk)} \quad (13)$$

$$x(k+\frac{N}{2}) = \sum_{n=0}^{\frac{N}{2-1}} x(2n) \quad W_N^{2nk} - W\sum_{N}^{K} x(2n+1)^{\frac{N}{2-1}} \quad W_N^{(2nk)} \quad (14)$$

The time burden created by this large number of computations limits the usefulness of DFT in many applications. Tremendous efforts devoted to develop more efficient ways of computing DFT resulted in the above explained Fast Fourier Transform algorithm. This mathematical shortcut reduces the number of calculations the DFT requires drastically. The above mentioned radix-2 decimation in time FFT is employed for domain transformation.

Dividing the DFT into smaller DFTs is the basis of the FFT. A radix-2 FFT divides the DFT into two smaller DFTs, each of which is divided into smaller DFTs and so on, resulting in a combination of two-point DFTs. The Decimation -In-Time (DIT) FFT divides the input (time) sequence into two groups, one of even samples and the other of odd samples. N/2 point DFT are performed on the these sub-sequences and their outputs are combined to form the N point DFT.

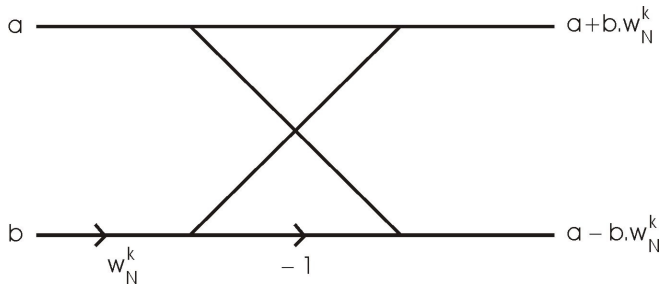
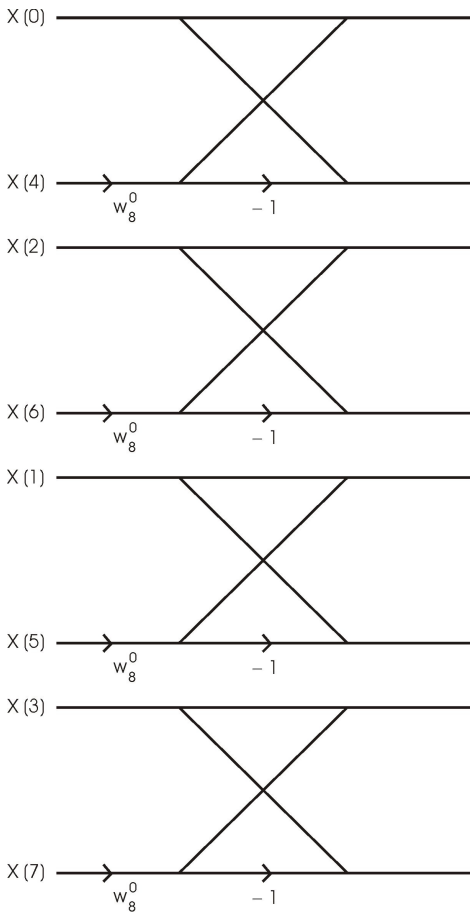
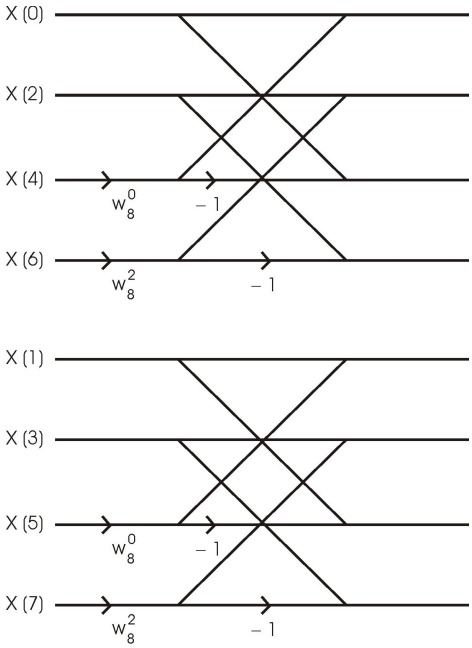



**FIG. 3A.1**

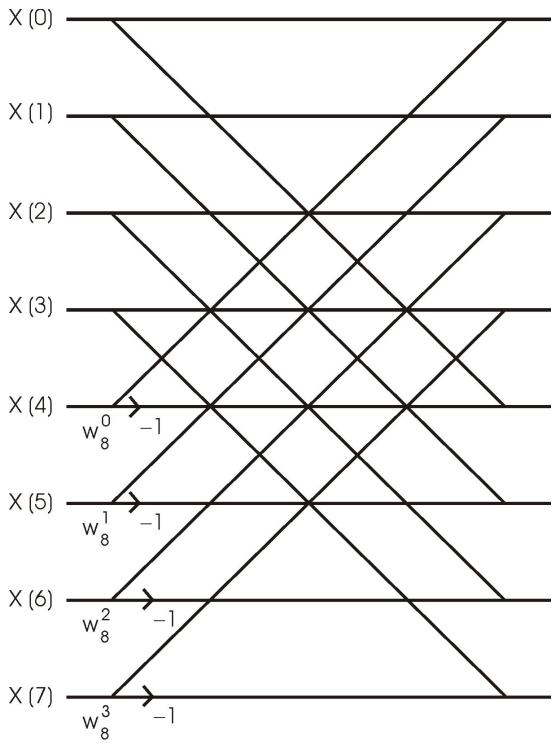
The above shown mathematical representation forms the basis of N point FFT and is called the **Butterfly Structure**.




**STAGE – I**




**STAGE - II**



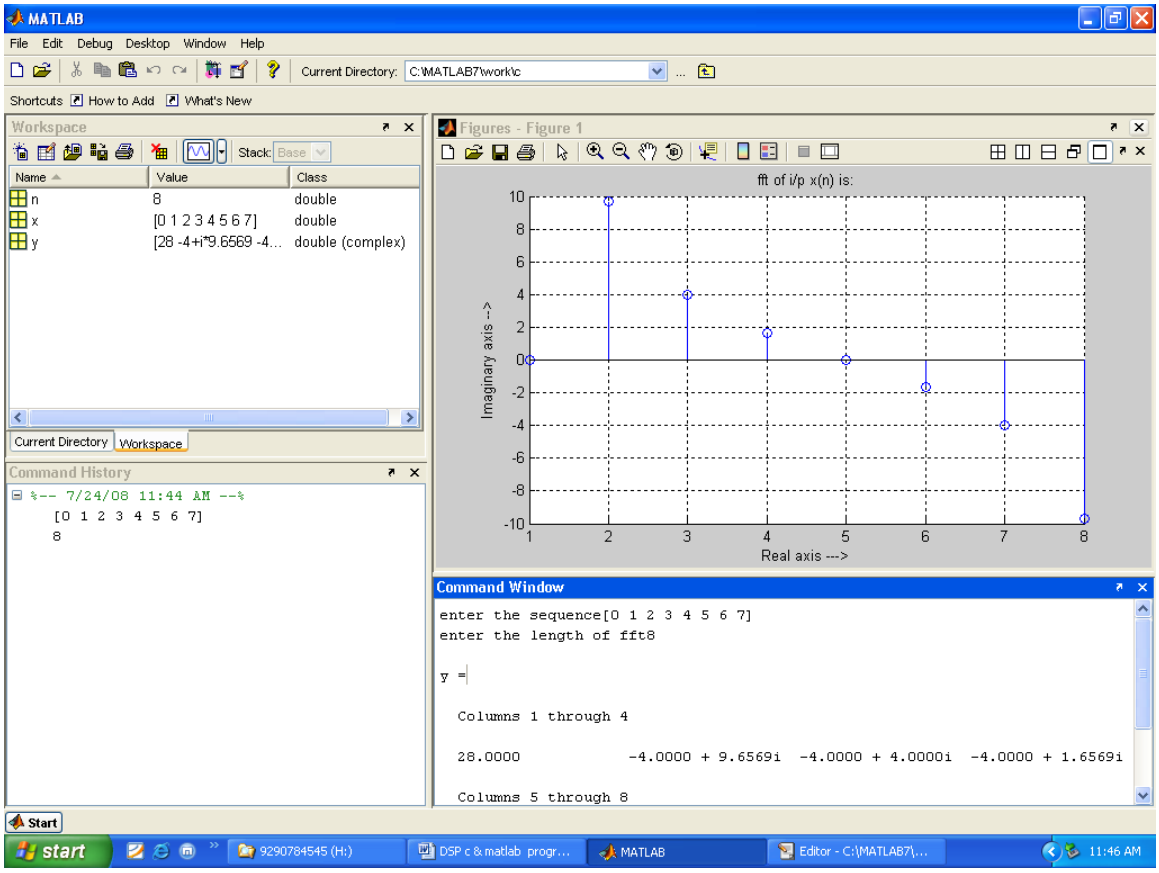

**STAGE – III**

**FIG. 3A.2 – 8 POINT DIT**

**PROGRAM:**

```
%fast fourier transform

clc;
clear all;
close all;
tic;
x=input('enter the sequence');
n=input('enter the length of fft');
%compute fft
disp('fourier transformed signal');
X=fft(x,n)
subplot(1,2,1);stem(x);
title('i/p signal');
xlabel('n --->');
ylabel('x(n) -->');grid;
subplot(1,2,2);stem(X);
title('fft of i/p x(n) is:');
xlabel('Real axis --->');
ylabel('Imaginary axis -->');grid;
```

**RESULT:**


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
n 8 double
x [0 1 2 3 4 5 6 7] double
y [28 -4+i*9.6569 -4... double (complex)
Current Directory Workspace
Command History
%-- 7/24/08 11:44 AM --%
[0 1 2 3 4 5 6 7]
8
Figures - Figure 1
fft of i/p x(n) is:
Imaginary axis -->
Real axis --->
Command Window
enter the sequence[0 1 2 3 4 5 6 7]
enter the length of fft8
y =
Columns 1 through 4
28.0000 -4.0000 + 9.6569i -4.0000 + 4.0000i -4.0000 + 1.6569i
Columns 5 through 8
Start
start
9290784545 (H:)
DSP c & matlab progr...
MATLAB
Editor - C:\MATLAB7\...
11:46 AM

# 6. Power Spectral Density

**AIM:**

To verify Power Spectral Density

**EQUIPMENTS:**

| | |
|---|---|
| Operating System | – Windows XP |
| Constructor | - Simulator |
| Software | - CCStudio 3 & MATLAB 7.5 |

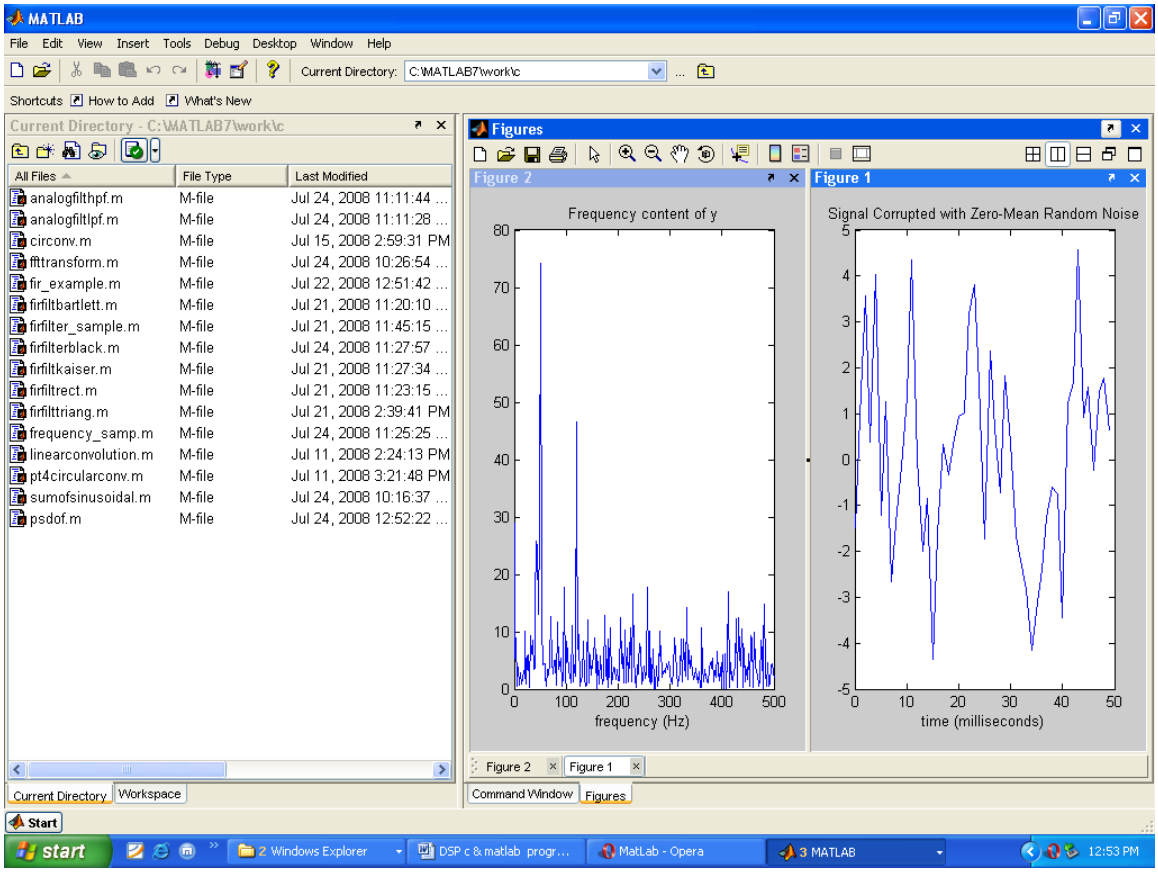
**THEORY:**

The power spectral density(P.S.D) is a measurement of the energy at various frequencies.

**PROGRAM:**

```
%Power spectral density
t = 0:0.001:0.6;
x = sin(2*pi*50*t)+sin(2*pi*120*t);
y = x + 2*randn(size(t));
figure,plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)');
Y = fft(y,512);
%The power spectral density, a measurement of the energy at various frequencies, is:
Pyy = Y.* conj(Y) / 512;
f = 1000*(0:256)/512;
figure,plot(f,Pyy(1:257))
title('Frequency content of y');
xlabel('frequency (Hz)');
```

**RESULT:**


MATLAB
File Edit View Insert Tools Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Shortcuts How to Add What's New
Current Directory - C:\MATLAB7\work\c
All Files File Type Last Modified
analogfilthpf.m M-file Jul 24, 2008 11:11:44 ...
analogfiltlpf.m M-file Jul 24, 2008 11:11:28 ...
circonv.m M-file Jul 15, 2008 2:59:31 PM
ffttransform.m M-file Jul 24, 2008 10:26:54 ...
fir_example.m M-file Jul 22, 2008 12:51:42 ...
firfiltbartlett.m M-file Jul 21, 2008 11:20:10 ...
firfilter_sample.m M-file Jul 21, 2008 11:45:15 ...
firfilterblack.m M-file Jul 24, 2008 11:27:57 ...
firfiltkaiser.m M-file Jul 21, 2008 11:27:34 ...
firfiltrect.m M-file Jul 21, 2008 11:23:15 ...
firfilttriang.m M-file Jul 21, 2008 2:39:41 PM
frequency_samp.m M-file Jul 24, 2008 11:25:25 ...
linearconvolution.m M-file Jul 11, 2008 2:24:13 PM
pt4circularconv.m M-file Jul 11, 2008 3:21:48 PM
sumofsinusoidal.m M-file Jul 24, 2008 10:16:37 ...
psdof.m M-file Jul 24, 2008 12:52:22 ...
Current Directory Workspace
Figures
Figure 2
Frequency content of y
frequency (Hz)
Figure 1
Signal Corrupted with Zero-Mean Random Noise
time (milliseconds)
Figure 2 Figure 1
Command Window Figures
Start
start
2 Windows Explorer
DSP c & matlab progr...
MatLab - Opera
3 MATLAB
12:53 PM

# 7. Sum of Sinusoidal Signals

**AIM:**

To verify Sum of Sinusoidal Signals using MATLAB

**EQUIPMENTS:**

Operating System – Windows XP
Constructor - Simulator
Software - CCStudio 3 & MATLAB 7.5

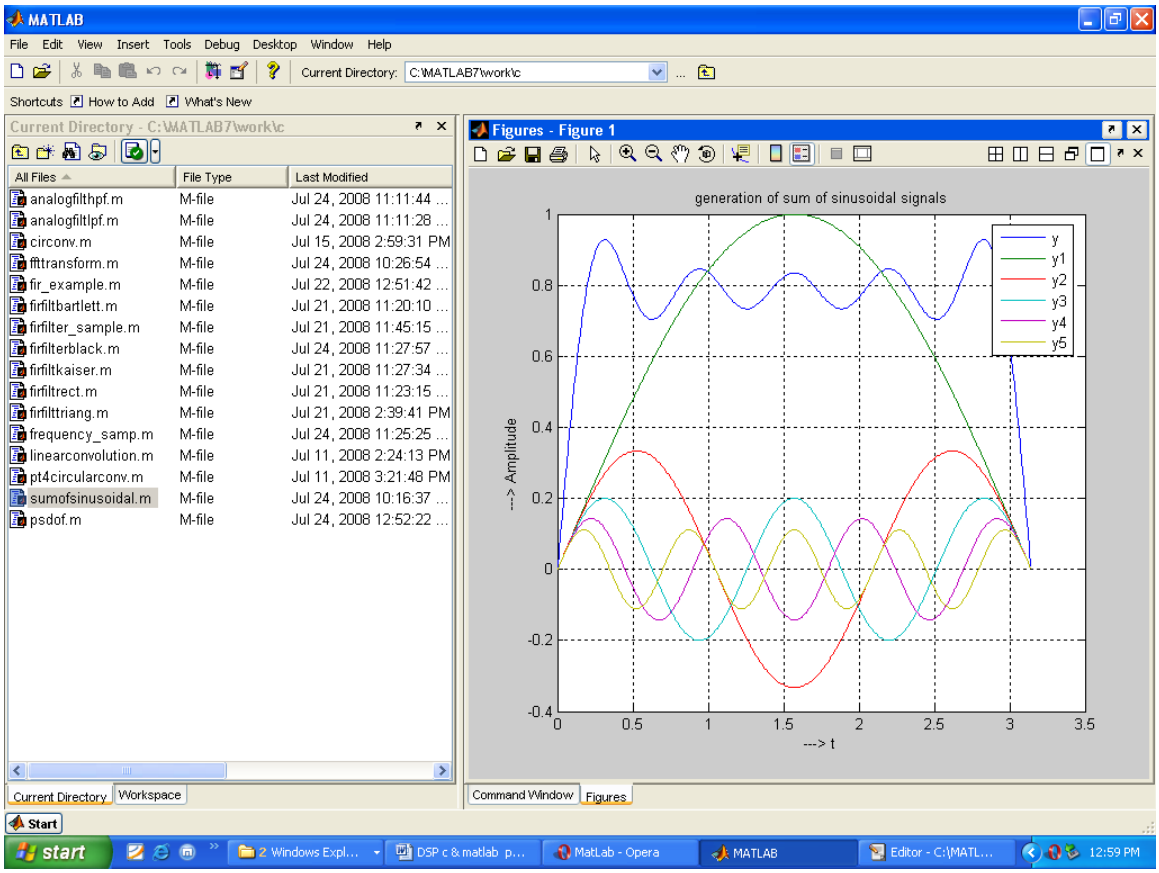
**THEORY:**

To generate fourier series of a signal by observing sum of sinusoidal signals & observing gibbs phenomenon effect.

**PROGRAM:**

```
% sum of sinusoidal signals
clc;
clear all;
close all;
tic;
%giving linear spaces
t=0:.01:pi;
% t=linspace(0,pi,20);
%generation of sine signals

y1=sin(t);
y2=sin(3*t)/3;
y3=sin(5*t)/5;
y4=sin(7*t)/7;
y5=sin(9*t)/9;

y = sin(t) + sin(3*t)/3 + sin(5*t)/5 + sin(7*t)/7 + sin(9*t)/9;

plot(t,y,t,y1,t,y2,t,y3,t,y4,t,y5);
legend('y','y1','y2','y3','y4','y5');
title('generation of sum of sinusoidal signals');grid;
ylabel('---> Amplitude');
xlabel('---> t');
toc;
```

**RESULT:**


MATLAB
File Edit View Insert Tools Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Shortcuts How to Add What's New
Current Directory - C:\MATLAB7\work\c
All Files File Type Last Modified
analogfilthpf.m M-file Jul 24, 2008 11:11:44 ...
analogfiltlpf.m M-file Jul 24, 2008 11:11:28 ...
circonv.m M-file Jul 15, 2008 2:59:31 PM
ffttransform.m M-file Jul 24, 2008 10:26:54 ...
fir_example.m M-file Jul 22, 2008 12:51:42 ...
firfiltbartlett.m M-file Jul 21, 2008 11:20:10 ...
firfilter_sample.m M-file Jul 21, 2008 11:45:15 ...
firfilterblack.m M-file Jul 24, 2008 11:27:57 ...
firfiltkaiser.m M-file Jul 21, 2008 11:27:34 ...
firfiltrect.m M-file Jul 21, 2008 11:23:15 ...
firfilttriang.m M-file Jul 21, 2008 2:39:41 PM
frequency_samp.m M-file Jul 24, 2008 11:25:25 ...
linearconvolution.m M-file Jul 11, 2008 2:24:13 PM
pt4circularconv.m M-file Jul 11, 2008 3:21:48 PM
sumofsinusoidal.m M-file Jul 24, 2008 10:16:37 ...
psdof.m M-file Jul 24, 2008 12:52:22 ...
Current Directory Workspace
Figures - Figure 1
generation of sum of sinusoidal signals
y
y1
y2
y3
y4
y5
---> Amplitude
---> t
Command Window Figures
Start
start
2 Windows Expl...
DSP c & matlab p...
MatLab - Opera
MATLAB
Editor - C:\MATL...
12:59 PM

# 8. LPF & HPF

**AIM:**

To verify response of analog LPF & HPF using MATLAB

**EQUIPMENTS:**

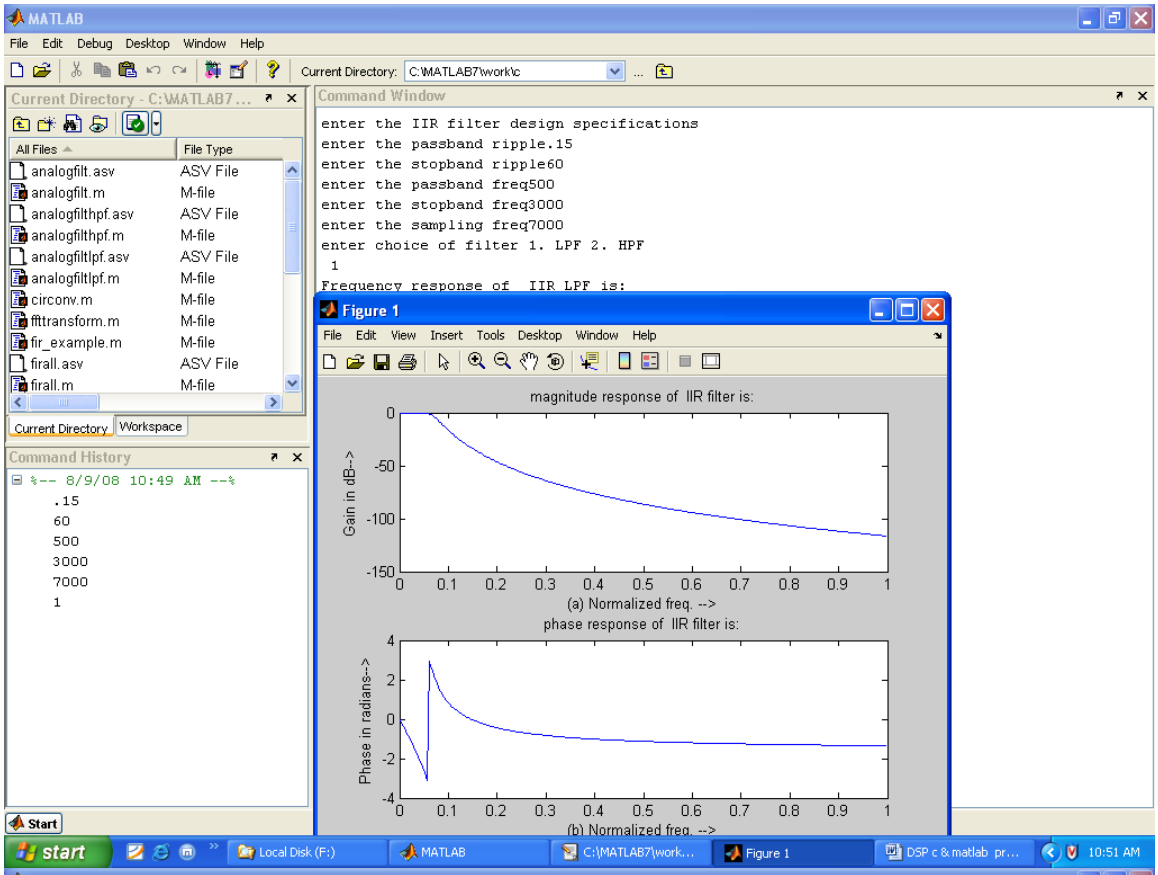
| | |
|---|---|
| Operating System | – Windows XP |
| Constructor | - Simulator |
| Software | - CCStudio 3 & MATLAB 7.5 |

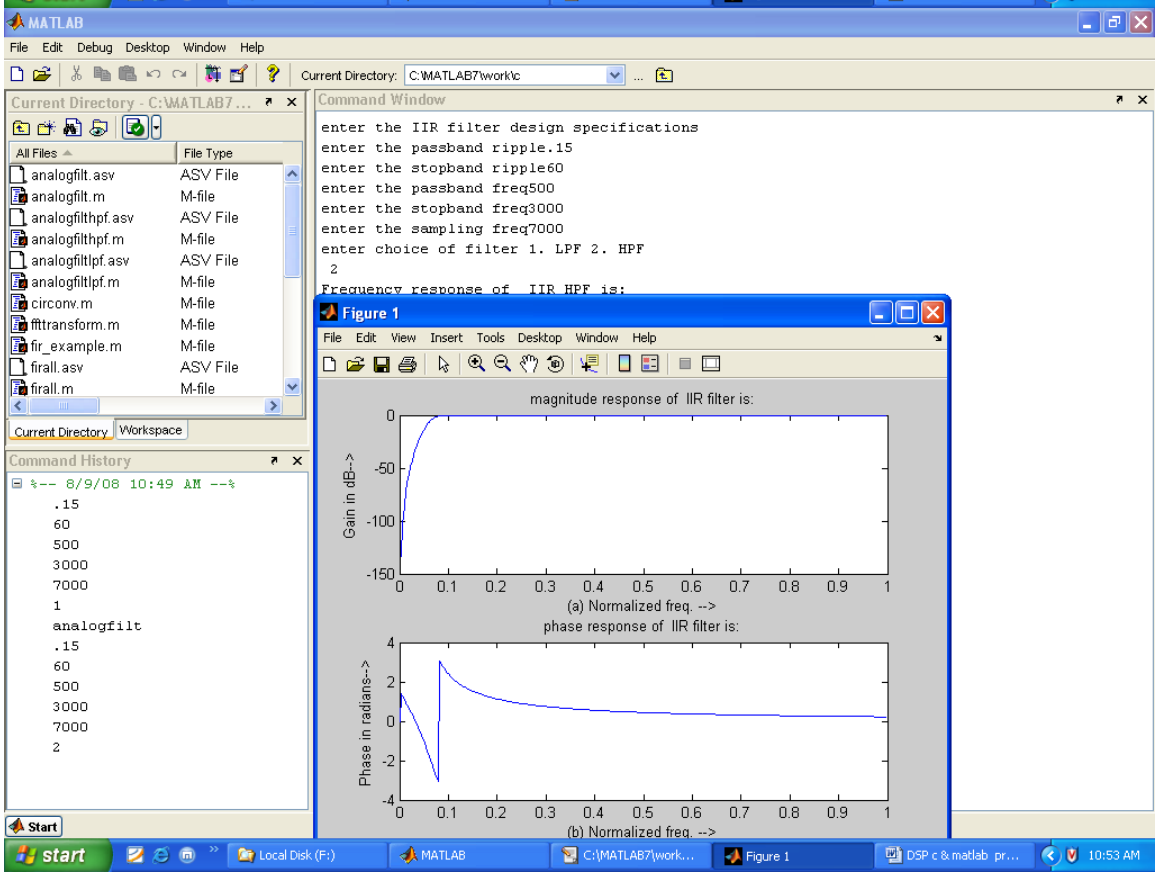
**THEORY:**

Analog Low pass filter & High pass filter are obtained by using butterworth or chebyshev filter with coefficients are given. The frequency – magnitude plot gives the frequency response of the filter.

**PROGRAM:**

```
%  IIR filters LPF & HPF
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
rp=input('enter the passband ripple');
rs=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
fs=input('enter the sampling freq');
w1=2*wp/fs;w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs,'s');
c=input('enter choice of filter 1. LPF 2. HPF \n ');
if(c==1)
   disp('Frequency response of  IIR LPF is:');
[b,a]=butter(n,wn,'low','s');
end
if(c==2)
   disp('Frequency response of  IIR HPF is:');
[b,a]=butter(n,wn,'high','s');
end
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure,subplot(2,1,1);plot(om/pi,m);
title('magnitude response of  IIR filter is:');
xlabel('(a) Normalized freq. -->');
ylabel('Gain in dB-->');
subplot(2,1,2);plot(om/pi,an);
title('phase response of  IIR filter is:');
xlabel('(b) Normalized freq. -->');
ylabel('Phase in radians-->');
```


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
analogfilt.asv ASV File
analogfilt.m M-file
analogfilthpf.asv ASV File
analogfilthpf.m M-file
analogfiltlpf.asv ASV File
analogfiltlpf.m M-file
circonv.m M-file
ffttransform.m M-file
fir_example.m M-file
firall.asv ASV File
firall.m M-file
Current Directory
Workspace
Command History
%-- 8/9/08 10:49 AM --%
.15
60
500
3000
7000
1
Command Window
enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
1
Frequency response of IIR LPF is:
Figure 1
File Edit View Insert Tools Desktop Window Help
magnitude response of IIR filter is:
Gain in dB-->
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->
(b) Normalized freq. -->
Start
start
Local Disk (F:)
MATLAB
C:\MATLAB7\work...
Figure 1
DSP c & matlab pr...
10:51 AM


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work\c
Current Directory - C:\MATLAB7...
All Files
File Type
analogfilt.asv ASV File
analogfilt.m M-file
analogfilthpf.asv ASV File
analogfilthpf.m M-file
analogfiltlpf.asv ASV File
analogfiltlpf.m M-file
circonv.m M-file
ffttransform.m M-file
fir_example.m M-file
firall.asv ASV File
firall.m M-file
Current Directory
Workspace
Command History
%-- 8/9/08 10:49 AM --%
.15
60
500
3000
7000
1
analogfilt
.15
60
500
3000
7000
2
Command Window
enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
2
Frequency response of IIR HPF is:
Figure 1
File Edit View Insert Tools Desktop Window Help
magnitude response of IIR filter is:
Gain in dB-->
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->
(b) Normalized freq. -->
Start
start
Local Disk (F:)
MATLAB
C:\MATLAB7\work...
Figure 1
DSP c & matlab pr...
10:53 AM