THE NEOTIA UNIVERSITY

LAB MANUAL

DIGITAL SIGNAL PROCESSING

PC-RE/P/403

CONTENT

SL NO.	NAME OF THE EXPERIMENT	EXPERIMENT NUMBER
1	Simulation of sampled Sinusoidal signal, various sequences and different arithmetic operations.	**PC-RE/P/403/01**
2	Simulation of convolution of two sequences using graphical method and using commands, verification of the properties of convolution.	**PC-RE/P/403/02**
3	Simulation of z transform of various sequences - verification of the properties of z transform.	**PC-RE/P/403/03**
4	Simulation of Twiddle factor-verification of the properties.	**PC-RE/P/403/04**
5	Simulation of DFT and IDFT using matrix multiplication and also using commands	**PC-RE/P/403/05**
6	Simulation of circular convolution of two sequences using graphical methods and using commands, differentiation between linear and circular convolutions.	**PC-RE/P/403/06**
7	Verifications of the different algorithms associated with filtering of long data sequences and Overlap –add and Overlap-save methods.	**PC-RE/P/403/07**
8	Butterworth filter design with different set of parameters.	**PC-RE/P/403/08**
9	FIR filter design using rectangular, Hamming and Blackman windows.	**PC-RE/P/403/09**
10	Writing & execution of small programs related to arithmetic operations and convolution using	**PC-RE/P/403/10**

	Assembly Language of TMS320C5416/6713 Processor, study of MAC *instruction*.	
11	Writing small programs in VHDL/Verilog and downloading onto Xilinx FPGA.	PC-RE/P/403/11
12	Mapping of some DSP algorithms onto FPGA.	PC-RE/P/403/12

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403/01

2.0 NAME OF JOB/EXPERIMENT: Simulation of sampled Sinusoidal signal, various sequences and different arithmetic operations.

3.0 OBJECTIVE: Simulation of sampled Sinusoidal signal, various sequences and different arithmetic operations.

4.0 PRINCIPLE:

Representation of a sequence is given by,

x(n)= {……..0,0,1,4,1,0,0,1,3,2……} [infinite duration sequence]

x(n)= {0,0,1,4,1,0,0,1,3,2} [finite duration sequence]

In the sequence time origin (n=0) is indicated by the symbol ‘ ’.

Time shifting:
The shift operation takes in the input sequence and it shifts the values by an integer increment of the independent variable. The shifting may delay or advance the sequence in time. Mathematically this can be represented as,

$y(n)=x(n-k)$

Where y(n) is the output sequence , x(n) is input sequence and k is integer which gives the shift in x(n).

Time reversal:
The time reversal of the sequence x(n) can be obtained by folding the sequence about n=0.It is denoted as x(-n).

$y(n)=x(-n)$

Scalar multiplication:

Here we need a scalar multiplier ‘a’. The sequence x(n) is multiplied by the scalar multiplier.

$y(n)=ax(n)$

$x_1(n)=\{1,2,3,4\}$
$x_2(n)=\{4,3,2,1\}$

$y(n)=x_1(n)+x_2(n)=\{5,5,5,5\}$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

```
%Generation of sampled sinusoidal signal

m=0:1:80;
x=cos(m*0.1*pi);
z=sin(m*0.1*pi);
y=x+z;
figure(5);
subplot(3,1,1),stem(m,x),
title('Generation of sampled cosine signal');
xlabel('n'),ylabel('amplitude');
subplot(3,1,2),stem(m,z),
title('Generation of sampled sine signal');
xlabel('n'),ylabel('amplitude');
subplot(3,1,3),stem(m,y);
title('Generation of sampled sinusoidal signal by adding sine & cosine signals');
xlabel('n'),ylabel('amplitude');

%Various sequences and different arithmetic operations

%Generation of sequence
n=-3:1:10;
a=[0 0 0 1 2 3 2 1 0 0 0 0 0 0];


figure(2);
subplot(3,1,1),stem(n,a),title(' Sequence'),axis([-4 10 0 4]);
xlabel('n'),ylabel('amplitude');

%Time shifting
subplot(3,1,2),stem(n+2,a),axis([-4 10 0 4]);
title('Sequence after twice right shift'),xlabel('n'),ylabel('amplitude');
subplot(3,1,3),stem(n-2,a),axis([-4 10 0 4]);
title('Sequence after twice left shift'),xlabel('n'),ylabel('amplitude');

%Time reversal
figure(3);
subplot(2,1,1),stem(n,a),axis([-10 10 0 4]);
title('Sequence'),xlabel('n'),ylabel('amplitude');
subplot(2,1,2),stem(-n,a),axis([-10 10 0 4]);
title('Time reversal'),xlabel('n'),ylabel('amplitude');
```

```
%Scalar multiplication
figure(4);
subplot(2,1,1),stem(n,a),axis([-4 10 0 7]);
title('Sequence'),xlabel('n'),ylabel('amplitude');
subplot(2,1,2),stem(n,a*2),axis([-4 10 0 7]);
title('Scalar multiplication (scalar multiplier =2)'),xlabel('n'),ylabel('amplitude');

%Multiplication and addition of two sequences
n=0:1:9;
b=[-1 2 -1 1 0 0 1 2 1 1];%one sequence
c=[1 2 3 2 0 0 1 2 -1 0];%another sequence
d=b.*c %multiplication
e=b+c %addition
```

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-1

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE: _____

_____ _____

Marks Obtained

Signature of the
Sessional in – charge

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403/02

2.0 NAME OF JOB/EXPERIMENT:

Simulation of convolution of two sequences using graphical method and using commands, verification of the properties of convolution.

3.0 OBJECTIVE:

To simulate the convolution of two sequences using graphical method and using commands-verify of the properties of convolution.

4.0 PRINCIPLE:

The convolution of sequences x(n) and h(n) is given by,

$$y(n) = \sum_{k=-\alpha}^{+\alpha} x(k)h(n-k)$$

Properties of convolution:

Commutative law:

$$x(n) * h(n) = h(n) * x(n)$$

Associative law:

$$[x(n) * h_1(n)] * h_2(n) = x(n) * [h_1(n) * h_2(n)]$$

Distributive law:

$$x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n)$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

```
%Convolution of two sequences
x=[1 0 -1 1 2 0]%1st sequence
y=[1 3 2 1]%2nd sequence
z=conv(x,y)%linear convolution of x & y
d=deconv(z,y)
figure(1);
n=0:1:5;
subplot(2,2,1),stem(n,x),axis([0  5  0  4]),
title('1st Sequence,x(n)'), xlabel('n'), ylabel('amplitude');
n=0:1:3;
subplot(2,2,2),stem(n,y),axis([0 5 0 4]),
title('2nd Sequence, y(n)'), xlabel('n'), ylabel('amplitude');
n=0:1:8;
subplot(2,2,3),stem(n,z),axis([0 11 -1 7]);
title(' linear convolution of 1st & 2nd Sequence,c(n)'),xlabel('n'),ylabel('amplitude');
n=0:1:5;
subplot(2,2,4),stem(n,d),axis([0 5 0 4]);
title('Deconvolution of y(n) & c(n)'),xlabel('n'),ylabel('amplitude');

%verification of the properties of convolution

%commutative law: [x(n)*y(n)=y(n)*x(n)]
x
y
a=conv(x,y)%L.H.S.
b=conv(y,x)%R.H.S.

%associative law: [x(n)*y(n)]*h(n)=x(n)*[y(n)*h(n)]
x
y

h=[1 1 1 1]
c=conv(x,y);
e=conv(y,h);
F=conv(c,h)%L.H.S.
G=conv(x,e)%R.H.S.


%distributive law: x(n)*[y(n)+h(n)]=x(n)*y(n)+x(n)*h(n)
x
y
h
k=y+h;
L=conv(x,k)%L.H.S.
o=conv(x,y);
p=conv(x,h);
R=o+p%R.H.S.
```

9.0 SAFETY:

NOT APPLICABLE

10.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE

OBJECTIVE: _____

_____ _____

Marks Obtained Signature of the Sessional in – charge

WORK INSTRUCTION

3.0 JOB/EXPERIMENT NO.: PC-RE/P/403/03

4.0 NAME OF JOB/EXPERIMENT:

Simulation of z transform of various sequences -verification of the properties of z transform.

5.0 OBJECTIVE:

To simulate of z transform of various sequences -verify of the properties of z transform

4.0 PRINCIPLE:

The z transform of discrete time signal sequences x(n) is given by,

$$X(Z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \quad \text{Where z is a complex variable.}$$

Properties of convolution:

Shifting property:

$$Z\{x(n-m)\} = z^{-m}X(z)$$

Linearity property:

$$Z[ax_1(n)+bx_2(n)] = aX_1(z)+bX_2(z)$$

Multiplication property:

$$Z\{a^n x(n)\} = X(a^{-1}z)$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

```
%program for computing z-transform and verify its properties
a=[0 0 1 0 3 -1 2 0 0 0]%a finite sequence
d=size(a);
%ztransfrm of a causal sequence
```

```
y1=ztran(a)

%verification of shifting property[Z{a(n+m)}=z^m*X(z)]
syms z p q;
m=2;%right shift 2
n=1:1:d(2)-m;
b(n)=a(n+m);
y2=ztran(b)%z transform of a(n+m)
y3=y1.*z^2

%verification of linearity property[Z{a*x1(n)+b*x2(n)}=a*X1(z)+b*X2(n)]
c=[0 0 1 1 2 1 2 0 1 0];%taking another sequence
y4=ztran(c);
e=(p*a)+(q*c);
y5=ztran(e)%computing LHS
y6=p*y1+q*y4%coputing RHS

%verification of mutiplication property[Z{a^n*x(n)}=X(z/a)]
b=2;%scaling factor
for n=1:1:d(2);
f(n)=a(n).*(b^(n-1));
y7=ztran(f);%computing LHS
end
y7
y8=rep(a,b)%%computing RHS


function ans=ztran(a)
        m=size(a);
        syms z;
        ans=0;
        for n=1:1:m(2);
                b(n)=a(n)*z^(-n+1);
                ans=ans+b(n);
        end
end

function ans=rep(a,x)
        m=size(a);
        syms z;
        ans=0;
        for n=1:1:m(2);
                b(n)=a(n)*(z/x)^(-n+1);
                ans=ans+b(n);
                end
end
```

7.0 SAFETY:

7.1 All the programs should be checked before run.

7.2 Shut down the computer after experiment.

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME ROLL NO.

1. _____ _____

2. _____ _____

3. _____ _____

4. _____ _____

5. _____ _____

TITLE _____

OBJECTIVE: _____

_____ _____

Marks Obtained Signature of the Sessional in - charge

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403//04

2.0 NAME OF JOB/EXPERIMENT:
Simulation of Twiddle factor-verification of the properties.

3.0 OBJECTIVE:
To simulate of Twiddle factor and to verify of the properties.

4.0 PRINCIPLE:

The N point discrete Fourier transform of a sequence x (n) is given below,

$$X(k)=\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi nk/N} \qquad 0 \le k \le N-1$$

Now exp(-j2π/N) in the above equation is called Twiddle factor(W).

$$W=\exp(-j2\pi/N)$$

Properties of Twiddle Factor:

Symmetry property:

Symmetry property of Twiddle factor is given below,

$$W^{(k+N/2)} = -W^{(k)}$$

Periodicity property:

Periodicity property of Twiddle factor is given below,

$$W^{(k+N)} = W^{(k)}$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

```
%Simulation of twiddle factor and verification its property
N=8;
W=exp(-1*i*2*pi/N) %generate twiddle factor

%Verification of symmetry property[W^(k+N/2)=-W^k]
k=2;
a=W^(k+N/2)%computing LHS
b=-W^k%coputing RHS

%Verification of periodicity property[W^(k+N)=W^k]
c=W^(k+N)%computing LHS
d=W^k%computing RHS
```

7.0 SAFETY:

7.1 All the programs should be checked before run.

7.2 Shut down the computer after experiment

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE: _____

_____
Marks Obtained

_____
Signature of the
Sessional in - charge

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403//05

2.0 NAME OF JOB/EXPERIMENT:

Simulation of DFT and IDFT using matrix multiplication and also using commands

3.0 **OBJECTIVE:**

To Simulate the DFT and IDFT of a given sequence.

4.0 PRINCIPLE:

The N point DFT of a given discrete sequence x(n) is given by,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

The N point IDFT of a given sequence X(k) is given by ,

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{j2\pi kn/N}$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

```
%Simulation of  N point DFT & IDFT

%simulation of N point DFT
x1=[1 1 1];%input sequence
N=60;
D=dft(x1,N);%calculating N point DFT
n=1:1;N;
A=abs(D);%amplitude of X(k)
B=angle(D);%phase of X(k)
k=1:1:N;
subplot(2,2,1),stem(k-1,A),title('Amplitude plot of X(k) of x(n) when
N=60'),xlabel('k'),ylabel('amplitude');
subplot(2,2,3 ),stem(k-1,B),title('Phase plot of X(k) of x(n)when
N=60'),xlabel('k'),ylabel('phase');

%comparing DFT for different value of N
N=10;
```

```
D=dft(x1,N);
n=1:1;N;
A=abs(D);
B=angle(D);
figure(1)
k=1:1:N;
subplot(2,2,2),stem(k-1,A);
title('Amplitude plot of X(k) of x(n)when N=10'),xlabel('k'),ylabel('phase');
subplot(2,2,4),stem(k-1,B);
title('Phase plot of X(k) of x(n)when  N=10'),xlabel('k'),ylabel('phase');

%simulation of N point IDFT
X=[1 0 1 0]
N=4;
I=idft(X,N)



function s=dft(x1,N)
        d=size(x1);
        L=d(2);
        x=[x1 zeros(1,N-L)];
        W=exp((-1)*i*2*pi/N);
        for n=1:1:N;
                sum=0;
                for m=1:1:N;
                        y(m)=x(m)*W^((m-1)*(n-1));
                        sum=sum+y(m);
                end
               s(n)=sum;
        end
end



function s=idft(x1,N)
        d=size(x1);
        L=d(2);
        x=[x1 zeros(1,N-L)];
        W=exp(i*2*pi/N);
        for n=1:1:N;
                sum=0;
                for m=1:1:N;
                        y(m)=x(m)*W^((m-1)*(n-1));
                        sum=sum+y(m);
                end
                s(n)=sum/N;
        end
end
```

11.0 SAFETY:

NOT APPLICABLE

12.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME ROLL NO.

1. _____ _____

2. _____ _____

3. _____ _____

4. _____ _____

5. _____ _____

TITLE

OBJECTIVE: _____

_____ _____

Marks Obtained Signature of the Sessional in - charge

WORK INSTRUCTION

4.0 JOB/EXPERIMENT NO.: EC PC-RE/P/403/692/06

5.0 NAME OF JOB/EXPERIMENT:

Simulation of circular convolution of two sequences using graphical methods and using commands, differentiation between linear and circular convolutions.

3.0 OBJECTIVE:

To simulate circular convolution of two sequences using graphical methods and using commands, differentiation between linear and circular convolutions.

4.0 PRINCIPLE:

The circular convolution of two discrete sequences $x_1(n)$ and $x_2(n)$ is given by,

$$x_3(m) = \sum_{n=0}^{N-1} x_1(n) x_2((m-n))_N \quad \text{m=0,1,2.....N-1}$$

Graphical method:

a) Graph N samples of $x_1(n)$ as equally spaced points around an circle in anticlockwise direction.
b) Start at same point as $x_1(n)$, graph N samples of $x_2(n)$ as equally spaced points around an inner circle in clockwise direction.
c) Multiply corresponding samples on the two circles and sum the products to produce the output.
d) Rotate the inner circle by one sample at a time in counter clockwise and go to step 3 to obtain the next value of output.

Matrix method:

If the Circular convolution of $x_1(n)$ & $x_1(n)$ is $x_3(n)$,then $x_3(n)$ can be found from yhe following equation,

$$\begin{bmatrix} x_2(0) & x_2(N-1) & x_2(N-2) & . & . & x_2(1) \\ x_2(1) & x_2(0) & x_2(N-1) & . & . & x_2(1) \\ x_2(2) & x_2(1) & x_2(0) & . & . & x_2(2) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_2(N-1) & x_2(N-2) & x_2(N-3) & . & . & x_2(0) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ . \\ . \\ x_1(N-1) \end{bmatrix} = \begin{bmatrix} x_3(0) \\ x_3(1) \\ x_3(2) \\ . \\ . \\ x_3(N-1) \end{bmatrix}$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Matlab program:

%Simulation of circular convolution and differentiation between linear and circular convolution

%Simulation of circular convolution between x(n) and h(n)

```
x=[1 1 1 1 1];
h=[1 1 1];
n=5;
n1=length(x);
n2=length(h);
x=[x zeros(1,n-n1)];
h=[h zeros(1,n-n2)];
C=crconv(x,h,n)
L=conv(x,h)
m=1:1:n;
subplot(2,2,1),stem(m-1,x),axis([0 n 0 4]),title('x(n)'),xlabel('n'),ylabel
 ('amplitude');

subplot(2,2,2),stem(m-1,h),axis([0 n 0 4]),title('h(n)'),xlabel('n'),ylabel
('amplitude') ;

subplot(2,2,3),stem(m-1,C),axis([0 n 0 4]),title('Circular convolution of x(n) &
h(n)'),xlabel('n'),ylabel('amplitude');

m=1:1:n1+n2+1;

subplot(2,2,4),stem(m-1,L),axis([0 8 0 4]),title('Linear convolution of x(n) &
h(n)'),xlabel('n'),ylabel('amplitude');


        function Y=crconv(x,h,n)
        q=length(h);%length of h(n)
        %formation of 1st matrix
        for i=1:1:n;
          p=0;
          for j=1:1:n;
            if j<=i;
              for k=1:1:i;
                X(i,k)=x(i-k+1);
              end
            else
              X(i,j)=x(n-p);
              p=p+1;
```

```
        end
      end
    end
    %formation of 2nd matrix
    H=[h zeros(1,n-q)];
    y=X*H.';
    %get final result
    Y=y.';
```

7.0 SAFETY:

NOT APPLICABLE

8.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE

OBJECTIVE: _____

Marks Obtained

Signature of the
Sessional in - charge

WORK INSTRUCTION

6.0 JOB/EXPERIMENT NO.: PC-RE/P/403/07

2.0 NAME OF JOB/EXPERIMENT: Verifications of the different algorithms associated with filtering of long data sequences and Overlap –add and Overlap-save methods.

3.0 OBJECTIVE: To verify the different algorithms associated with filtering of long data sequences and Overlap –add and Overlap-save methods.

4.0 PRINCIPLE:

In real-time signal monitoring applications the input signal x(n) is often very long. To find out response of such a long signal at tine is not possible through linear filtering via the DFT due to large memory requirements. This problem can be overcome by the following strategy:

a. Segment the input signal into fixed-size blocks prior to processing.
b. Compute DFT-based linear filtering of each block separately via the FFT.
c. Fit the output blocks together in such a way that the overall output is equivalent to the linear filtering of x(n) directly.

There are two approaches available in real-time linear filtering of long inputs:

I) Overlap-Add Method
II) Overlap-Save Method

The ***Overlap-Add Method*** deals with the following signal processing principles:

- The linear convolution of a discrete-time signal of length L and a discrete-time signal of length M produces a discrete-time convolved result of length L+M-1.
- **Addititvity:**

$$(x_1(n)+x_2(n))^* h(n) = x_1(n) *h(n)+x_2(n)*h(n)$$

The ***Overlap-Save Method*** deals with the following signal processing principles:

- The N = (L+M-1)-circular convolution of a discrete-time signal of length N and a discrete-time signal of length M using an N-DFT and N-IDFT.
- Time-Domain Aliasing:

$$x_c(n) = \sum_{l=-\infty}^{\infty} x_L(n - lN) \; ; n = 0,1,2, \dots \dots N-1.$$

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

6.0 PROCEDURE:

Algorithm:

Overlap-Add Method

1. Break the input signal x(n) into non-overlapping blocks $x_m(n)$ of length L.
2. Zero pad h(n) to be of length N = L + M -1.
3. Take N-DFT of h(n) to give H(k), k 0,1,2,…… N-1.
4. For each block m:
   - 4.1 Zero pad $x_m(n)$ to be of length N = L + M -1.
   - 4.2 Take N-DFT of $x_m(n)$ to give $X_m(k)$, k = 0,1,2,…… N-1.
   - 4.3 Multiply: $Y_m(k) = Xm(k)\ H(k)$, k = 0,1,2,…… N-1.
   - 4.4 Take N-IDFT of $Y_m(k)$ to give ym(n), n = 0,1,2,…… N-1.
5. Form y(n) by overlapping the last M -1 samples of $y_m(n)$ with the first M -1 samples of $y_{m+1}(n)$ and adding the result.

Overlap-Save Method

1. Insert M-1 zeros at the beginning of the input sequence x(n).
2. Break the padded input signal into overlapping blocks $x_m(n)$ of length N = L + M- 1 where the overlap length is M-1.
3. Zero pad h(n) to be of length N = L + M- 1.
4. Take N-DFT of h(n) to give H(k), k = 0,1,2,…… N-1.
5. For each block m:
   - 5.1 Take N-DFT of $x_m(n)$ to give $X_m(k)$, k = 0,1,2,…… N-1.
   - 5.2 Multiply: $Y_m(k) = Xm(k)\ H(k)$, k = 0,1,2……N -1.
   - 5.3 Take N-IDFT of $Y_m(k)$ to give $y_m(n)$, n = 0,1,2……N -1.
   - 5.4 Discard the first M -1 points of each output block $y_m(n)$.
6. Form y(n) by appending the remaining (i.e., last) L samples of each block $y_m(n)$.

13.0 SAFETY:

NOT APPLICABLE

14.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.
9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.
9.3 Write your final report as per the Work Instruction

ANNEXURE-1

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME ROLL NO.

1. _____ _____

2. _____ _____

3. _____ _____

4. _____ _____

5. _____ _____

TITLE _____

OBJECTIVE:

_____ _____

Marks Obtained Signature of the sessional in – charge

WORK INSTRUCTION

7.0 JOB/EXPERIMENT NO.: PC-RE/P/403//08

6.0 NAME OF JOB/EXPERIMENT:

Butterworth filter design with different set of parameters.

5.0 OBJECTIVE:

To design Butterworth filter with different set of parameters.

6.0 PRINCIPLE:

IIR filters are digital filters with infinite impulse response. Unlike FIR filters, they have the feedback (a recursive part of a filter) and are known as recursive digital filters therefore. For this reason IIR filters have much better frequency response than FIR filters of the same order. Unlike FIR filters, their phase characteristic is not linear which can cause a problem to the systems which need phase linearity. For this reason, it is not preferable to use IIR filters in digital signal processing when the phase is of the essence. Otherwise, when the linear phase characteristic is not important, the use of IIR filters is an excellent solution.

A digital filter, $H(e^{j\omega})$, with infinite impulse response (IIR), can be designed by first transforming it into a prototype analog filter $H_c(e^{j\omega})$, and then design this analog filter using a standard procedure. Once the analog filter is properly designed, it is then mapped back to the discrete-time domain to obtain a digital filter that meets the specifications.

There are two main techniques used to design IIR filters:

a) The Impulse Invariant method, and

b) The Bilinear transformation method.

Low-pass Butterworth analog filters are filters whose frequency response is a monotonous descending function. They are also known as "maximally flat magnitude" filters at the frequency of $\Omega = 0$, as first 2N-1 derivatives of the transfer function when $\Omega = 0$ are equal to zero.

Butterworth filter is characterized by 3dB attenuation at the frequency of $\Omega=1$, no matter the filter order is. Figure 8.1 illustrates frequency responses for a few various parameters N (filter order).

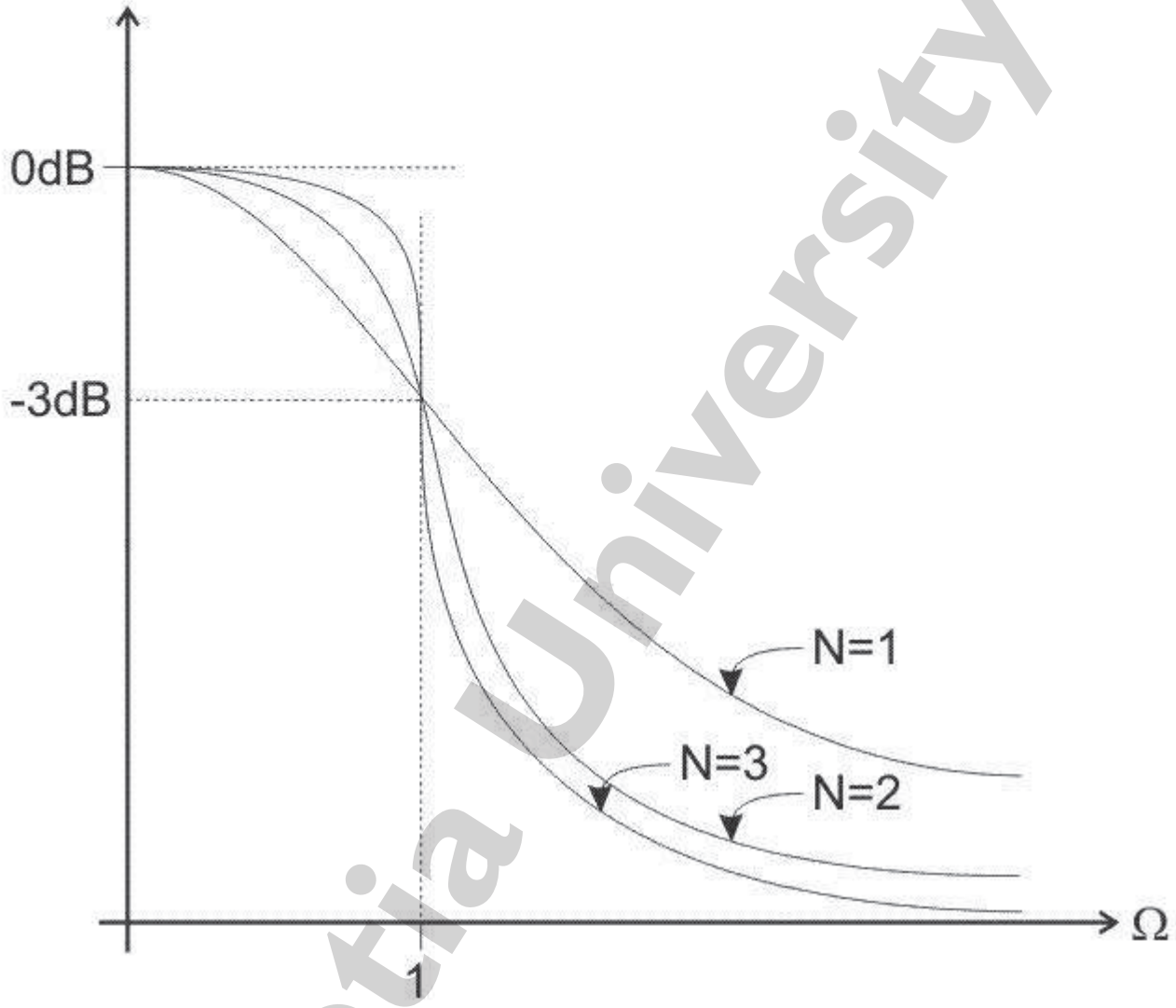


Figure 8.1. Frequency response of Butterworth filter

Butterworth filter is defined via expression:

$$\left|H_a(j\Omega)\right|^2 = H_a(j\Omega)\cdot H_a(-j\Omega) = \frac{1}{1+\Omega^{2N}}$$

where:

Ω is the frequency; and
N is the filter order.

5.0 APPARATUS REQUIRED:

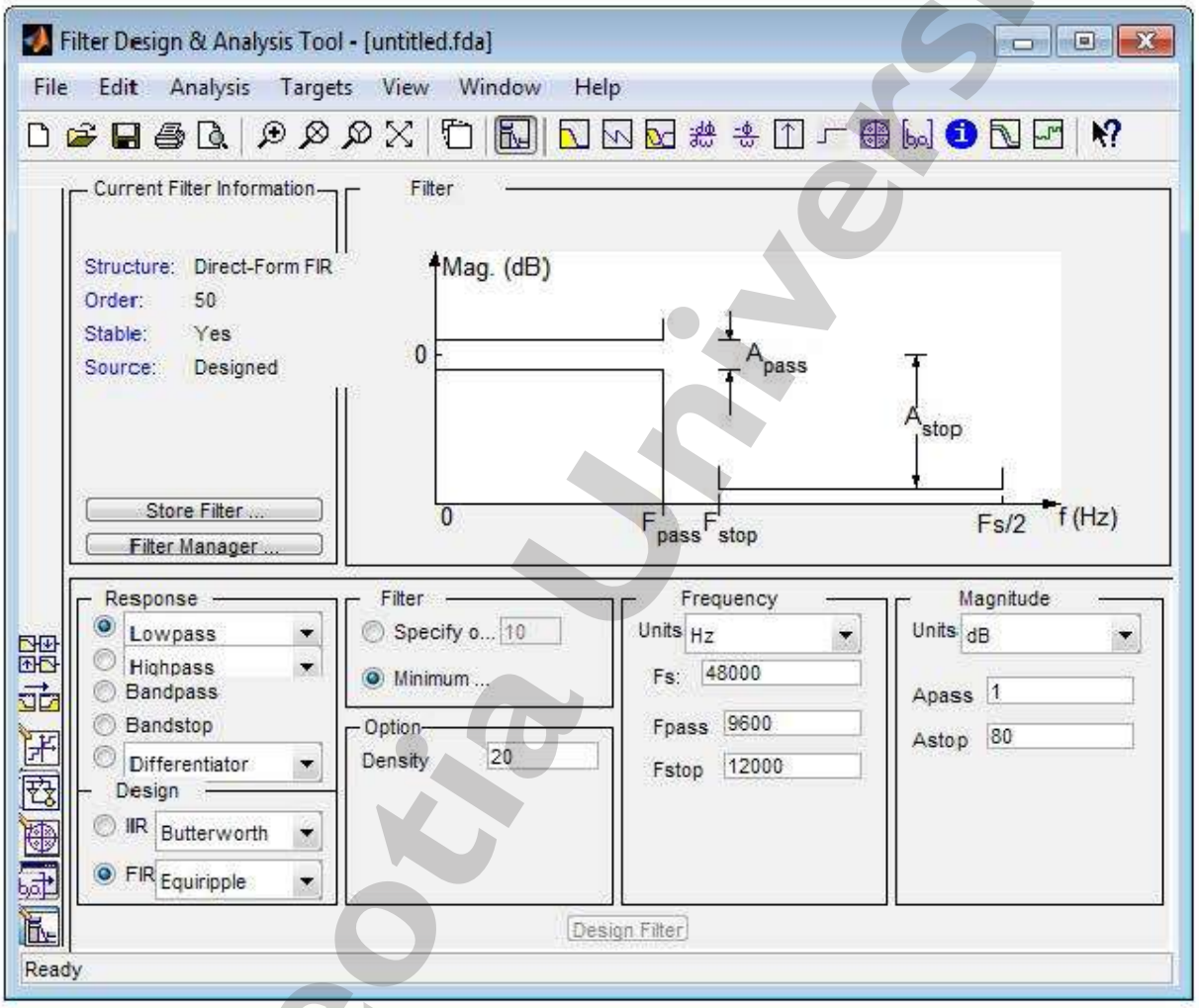
SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

9.0 PROCEDURE:

To open the ***Filter Design & Analysis tool***: go to matlab command Window and execute the following command

>>***fdatool***

The following window will appear



After getting the ***Filter Design & Analysis tool***(above window), select the following Parameters

Response type : Lowpass / Highpass

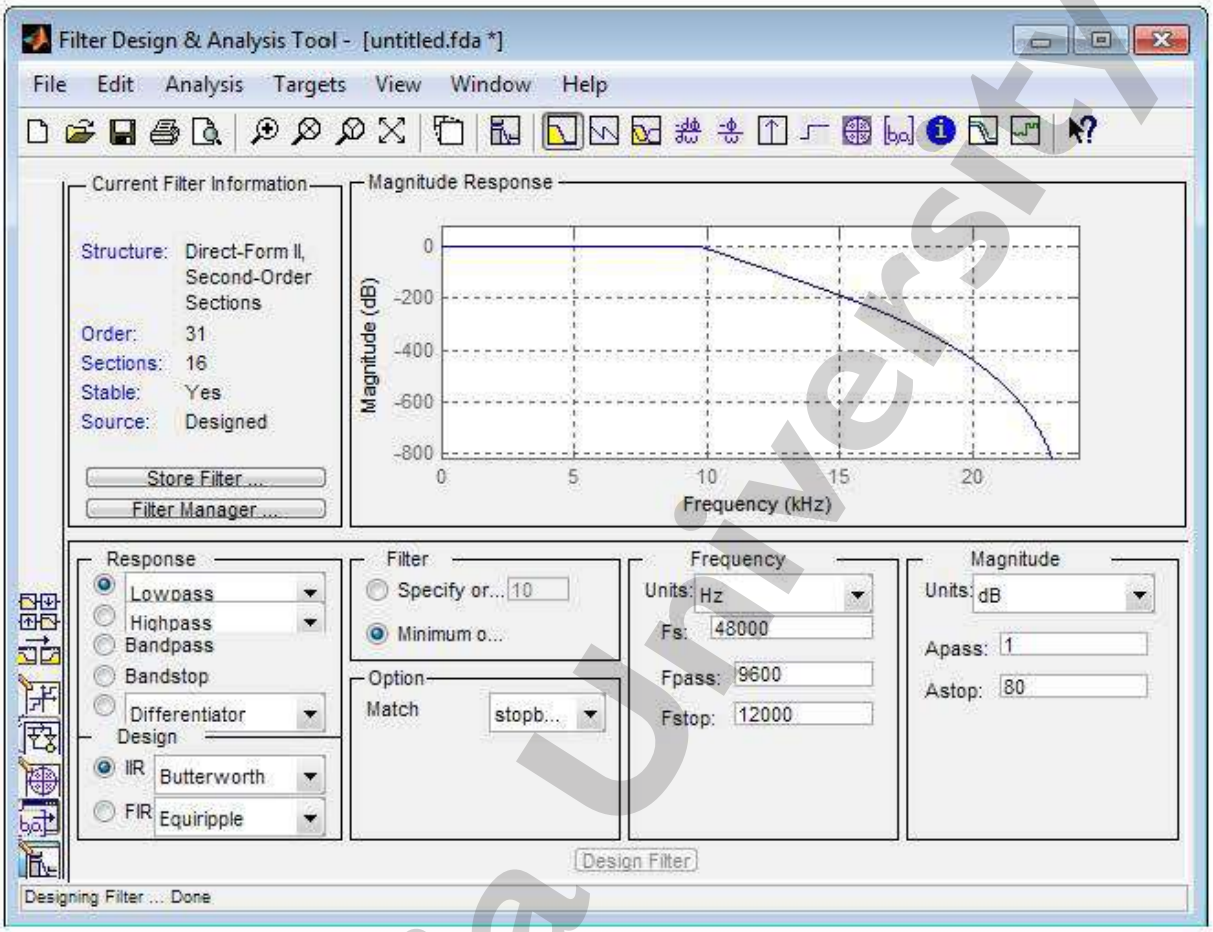
Design Method : Select IIR:Butterworth

Filter Order : Minimum order

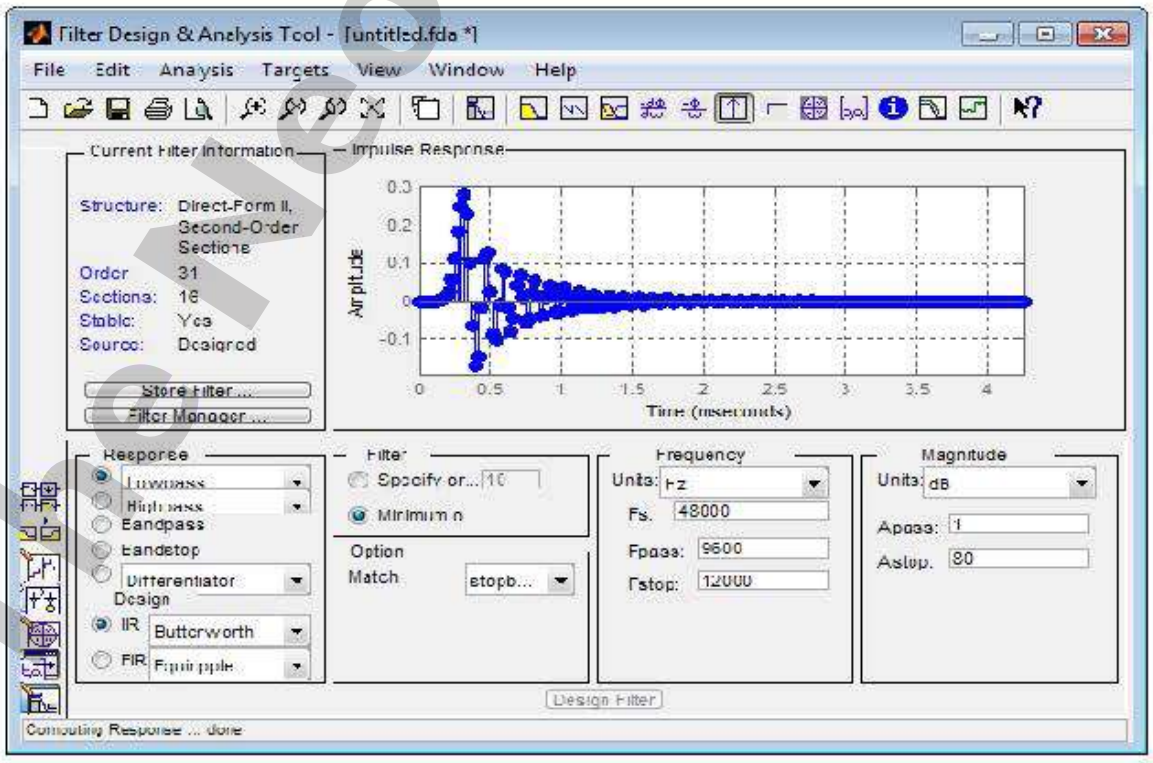
Frequency Specification : Select units and mention sampling frequency(Fs), passband frequency(**Fpass**) and stopband frequencie(**Fstop**) according to the requirement.

Magnitude Specification : Select units and mention **Apass** and **Astop**

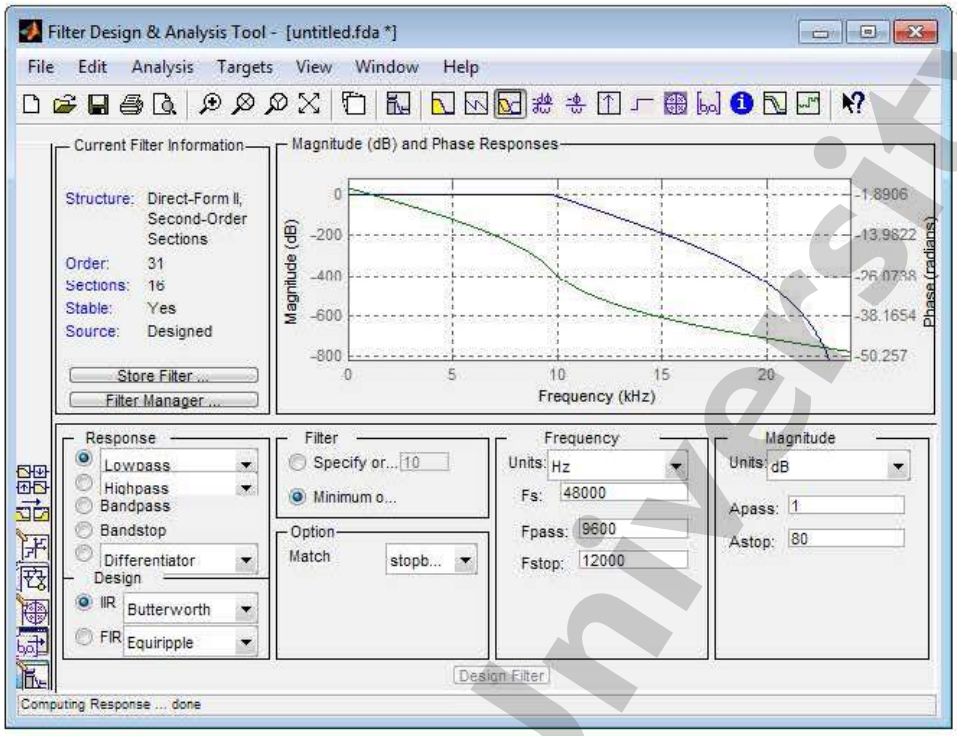
Click on the '***Design Filter***' button to see the magnitude response of the desired filter



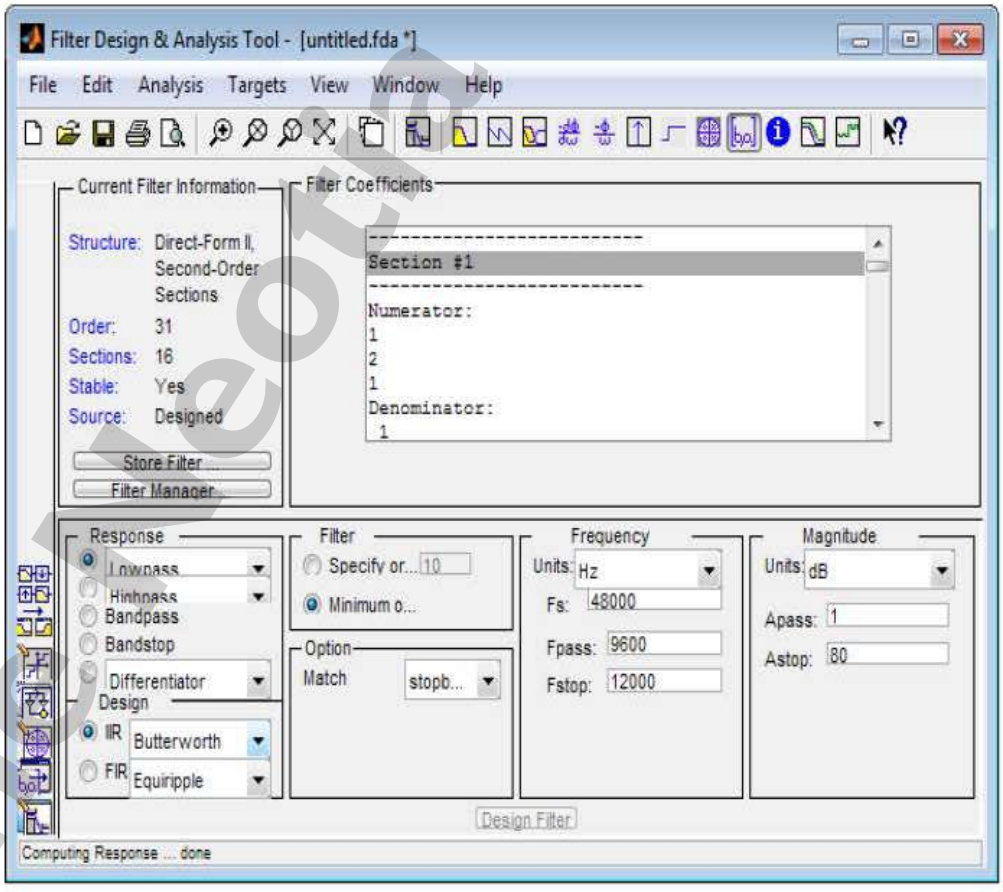
To see impulse response : ***Analysis → Impulse Response***



To see magnitude and phase responses : ***Analysis → Magnitude and Phase responses***



To see the filter coefficients : *Analysis → Filter Coefficients*



10.0 SAFETY:

NOT APPLICABLE

11.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – I.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE:

_____ _____

Marks Obtained Signature of the Sessional in - charge

WORK INSTRUCTION

8.0 JOB/EXPERIMENT NO.: PC-RE/P/403/09

7.0 NAME OF JOB/EXPERIMENT:

FIR filter design using rectangular, Hamming and Blackman windows.

7.0 OBJECTIVE:

To design FIR filter using rectangular, Hamming and Blackman windows.

8.0 PRINCIPLE:

FIR filters are digital filters with finite impulse response. They are also known as **non-recursive digital filters** as they do not have the feedback (a recursive part of a filter), even though recursive algorithms can be used for FIR filter realization. FIR filters can be designed using different methods, but most of them are based on ideal filter approximation. The objective is not to achieve ideal characteristics, as it is impossible anyway, but to achieve sufficiently good characteristics of a filter. The transfer function of FIR filter approaches the ideal as the filter order increases, thus increasing the complexity and amount of time needed for processing input samples of a signal being filtered.

An FIR filter can be described by the difference equation

$$y(n) = \sum_{l=0}^{M} b_i x(n-l)$$

and by the transfer function

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \sum_{l=0}^{M} b_i e^{-j\omega l}$$

We address the problem of designing an FIR filter that meets specifications of limited deviation from the ideal response in specified frequency bands. The window design method does not produce filters that are optimal (in the sense of meeting the design specifications in the most computationally-efficient fashion), but the method is easy to understand and does produce filters that are reasonably good. Of all the hand-design methods, the window method is the most popular and effective.

In brief, in the window method we develop a causal linear-phase FIR filter by multiplying an ideal filter that has an infinite-duration impulse response (IIR) by a finite-duration window function:

$$h[n] = h_d[n].W[n]$$

where *h[n]* is the practical FIR filter, $h_d[n]$ is the ideal IIR prototype filter, and $W[n]$ is the finite-duration window function. Some definition of the important windows are given below.

Rectangular window:

$$W[n] = \begin{cases} 1\ ;\ 0 \le n \le M \\ 0;\ otherwise \end{cases}$$

Hamming window:

$$W[n] = \begin{cases} 0.54 - 0.46cos\left(\frac{2\pi n}{M}\right)\ ;\ 0 \le n \le M \\ 0;\ otherwise \end{cases}$$

Blackmam window:

$$W[n] = \begin{cases} 0.42 - 0.5cos\left(\frac{2\pi n}{M}\right) + 0.08cos(4\pi n/M)\ ;\ 0 \le n \le M \\ 0;\ otherwise \end{cases}$$

5.0 APPARATUS REQUIRED:

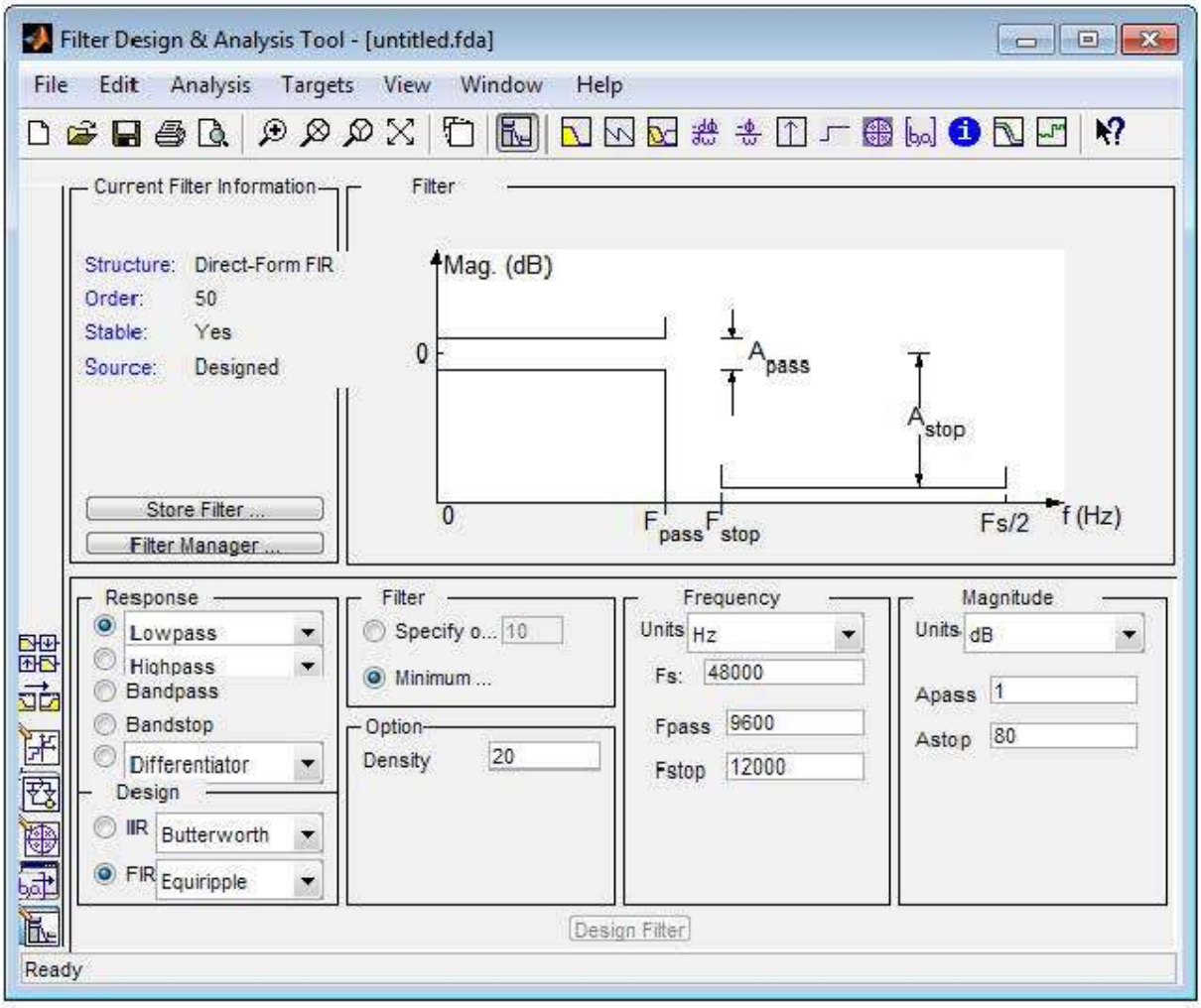
SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	Matlab		

12.0 PROCEDURE:

To open the ***Filter Design & Analysis tool***: go to matlab command Window and execute the following command

>>*fdatool*

The following window will appear



After getting the ***Filter Design & Analysis tool***(above window), select the following Parameters

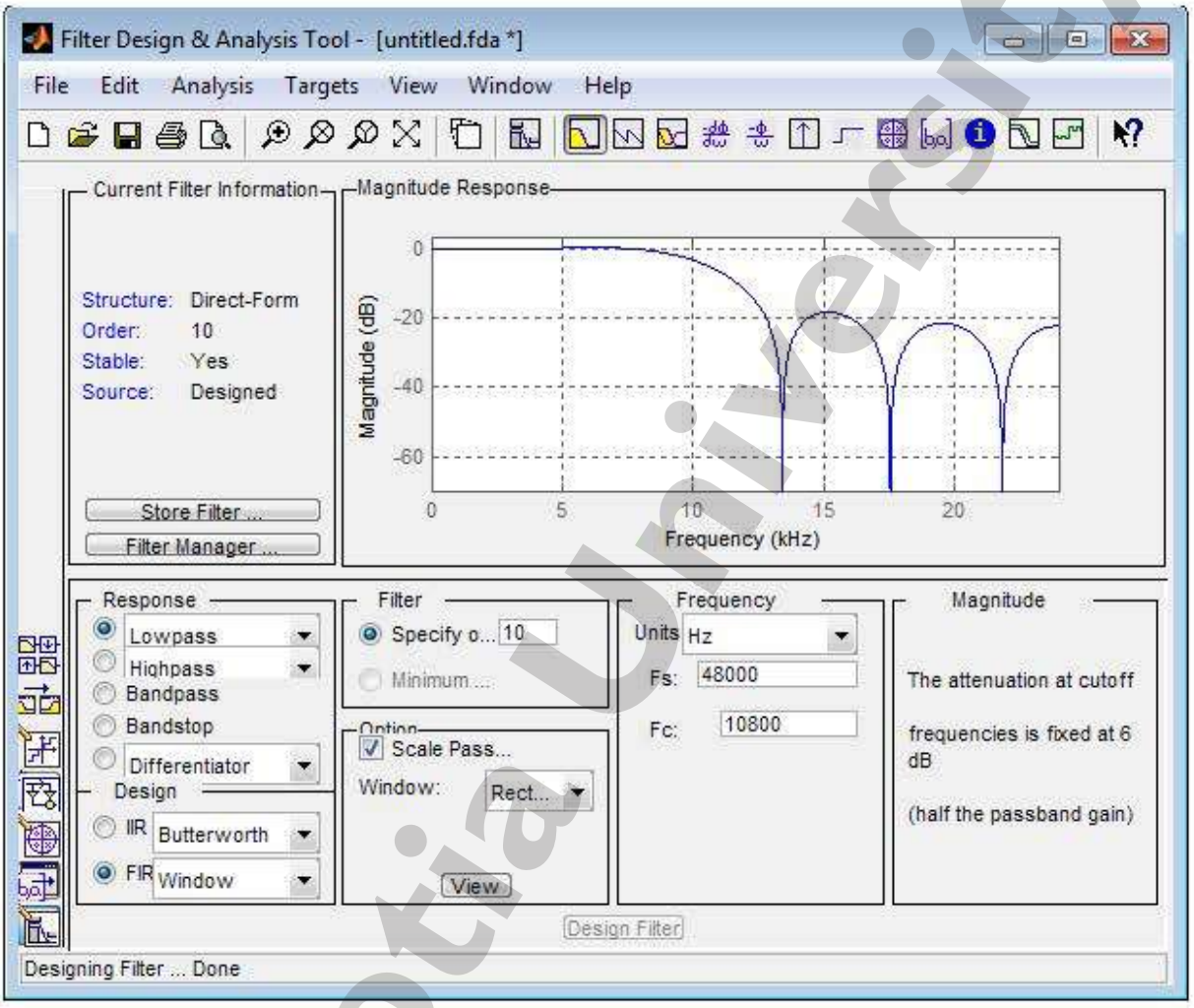
Response type : Lowpass / Highpass

Design Method : Select FIR:Window
Select required window type from ***Window options.***

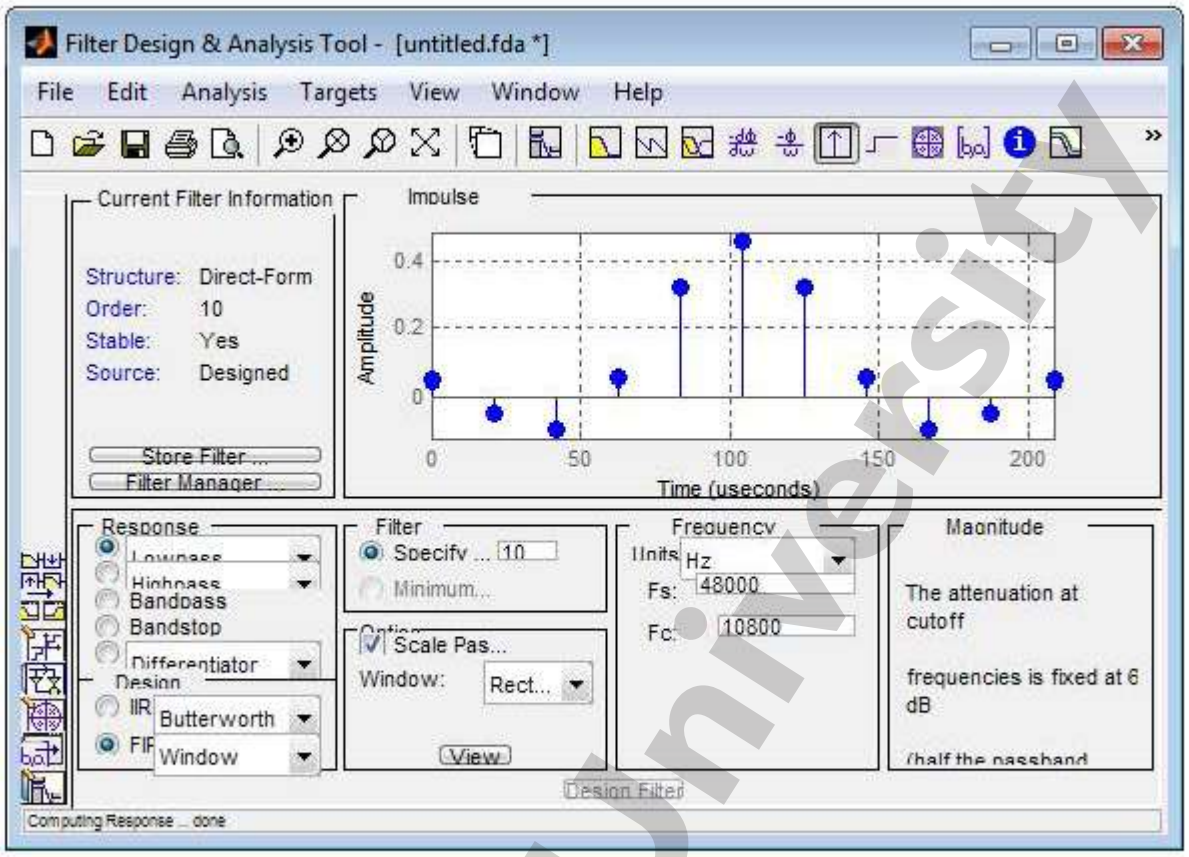
Filter Order : Minimum order

Frequency Specification : Select units and mention sampling frequency(Fs) and cutoff frequency(**Fc**).

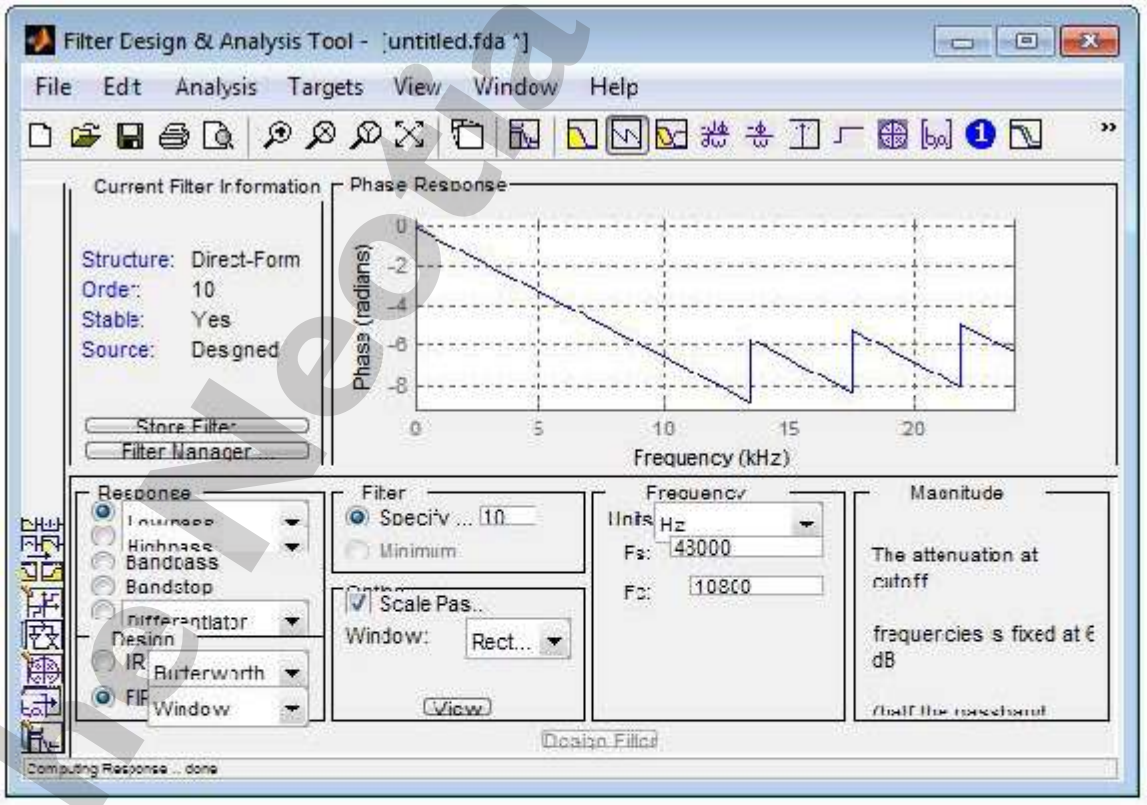
Click on the '***Design Filter***' button to see the magnitude response of the desired filter



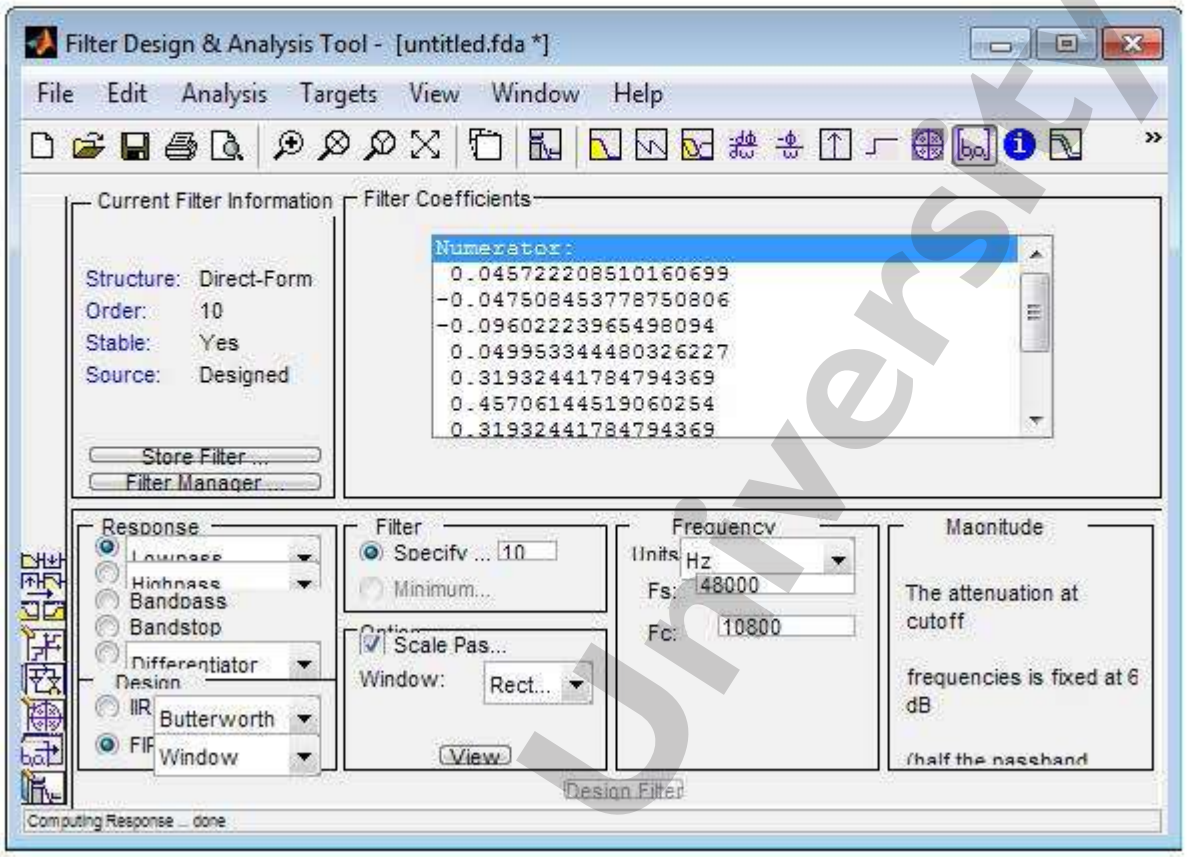
To see impulse response : ***Analysis → Impulse Response***



To phase responses : ***Analysis → Phase responses***



To see the filter coefficients : ***Analysis → Filter Coefficients***



13.0 SAFETY:

NOT APPLICABLE

14.0 DISPOSAL:

NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – I.

9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE:

_____ _____

Marks Obtained Signature of the Sessional in - charge

WORK INSTRUCTION

9.0 JOB/EXPERIMENT NO.: PC-RE/P/403/10

8.0 NAME OF JOB/EXPERIMENT:

Writing & execution of small programs related to arithmetic operations and convolution using Assembly Language of TMS320C5416/6713 Processor, study of MAC *instruction*.

9.0 OBJECTIVE:

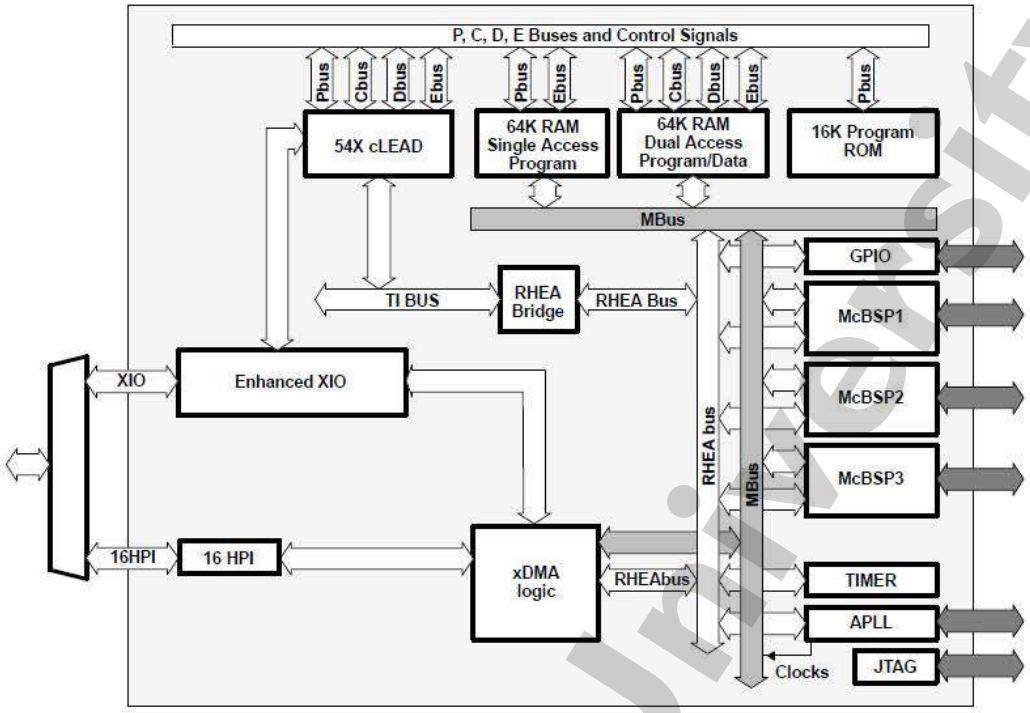
To write & execution of small programs related to arithmetic operations and convolution using Assembly Language of TMS320C5416/6713 Processor, study of MAC instruction.

10.0 PRINCIPLE:

The TMS320VC5416 fixed-point, digital signal processor (DSP) (hereafter referred to as the device unless otherwise specified) is based on an advanced modified Harvard architecture that has one program memory bus and three data memory buses. This processor provides an arithmetic logic unit (ALU) with a high degree of parallelism, application-specific hardware logic, on-chip memory, and additional on- chip peripherals. The basis of the operational flexibility and speed of this DSP is a highly specialized instruction set.

Separate program and data spaces allow simultaneous access to program instructions and data, providing a high degree of parallelism. Two read operations and one write operation can be performed in a single cycle. Instructions with parallel store and application-specific instructions can 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. The device also includes the control mechanisms to manage interrupts, repeated operations, and function calls.

Functional Overview



Software Development Tools Overview:

The following figure illustrates the C54x software development flow. The shaded portion of the figure highlights the most common path of software development; the other portions are optional.

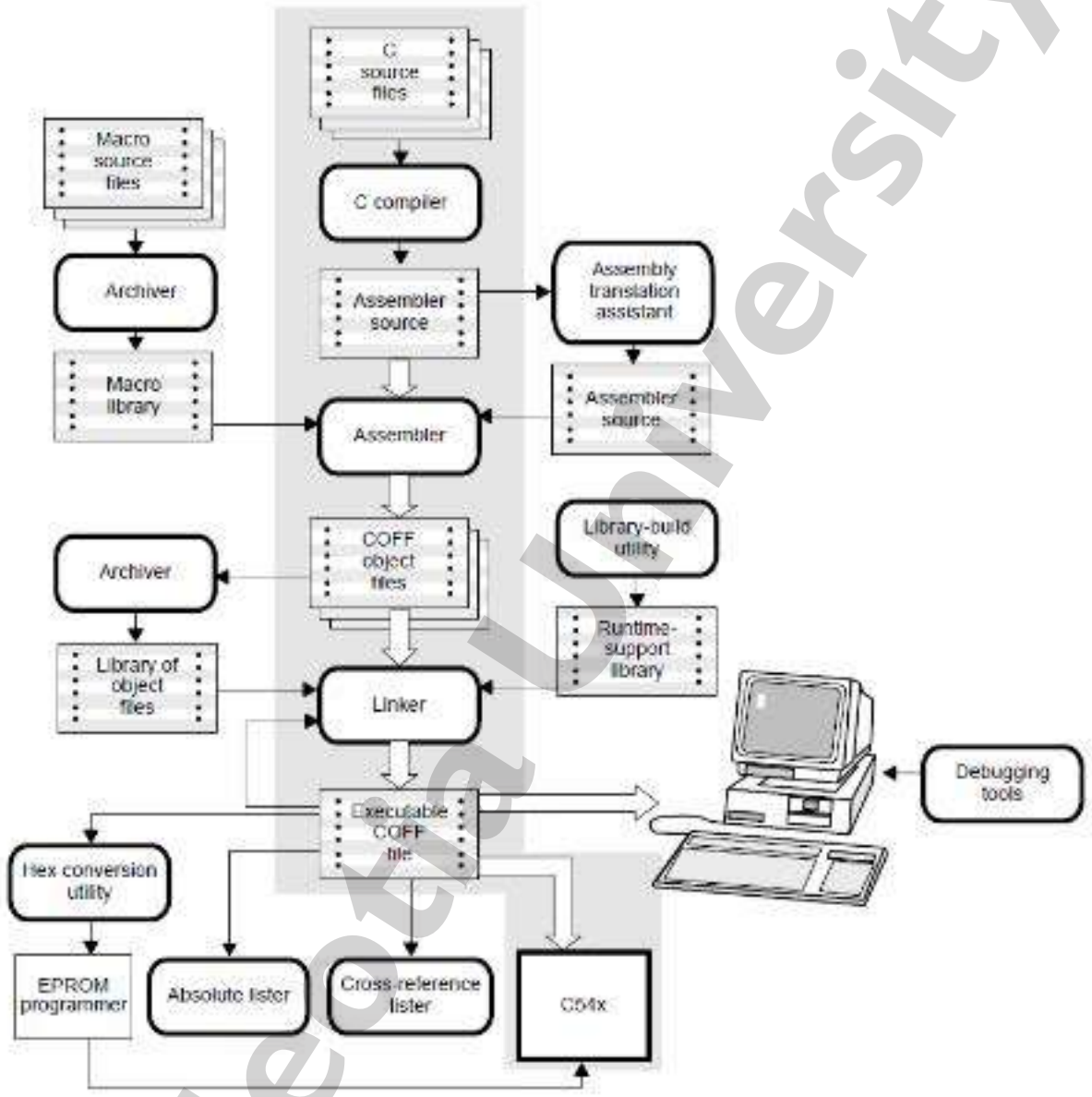


Fig:-TMS320C54xSoftware Development Flow

The following list describes the tools that are shown in the figure:

The **C/C++ compiler** translates C/C++ source code into C54x assembly language source code.

The **assembler** translates assembly language source files into machine language COFF object files. The TMS320C54x tools include two assemblers. The mnemonic assembler accepts C54x and C54xmnemonic assembly source files. The algebraic assembler accepts C54xalgebraic assembly source files. Source files can contain instructions, assembler directives, and macro directives.

The **linker** combines relocatable COFF object files (created by the assembler) into a single executable COFF object module. Linker directives allows to combine object file sections, bind sections or symbols to addresses or within memory ranges, and define or redefine global symbols.

The archiver collects a group of files into a single archive file.

The library-build utility builds your own customized C/C++ runtime-support library. Standard runtime-support library functions are provided a ssource code in rts.src and as object code in rts.lib.

The TMS320C54x Code Composer Studio debugger accepts COFF files as input, but most EPROM programmers do not.

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME	RANGE
01	Computer		
02	TMS320C5416/6713 Processor	Texas Instruments	
03	Code Composer Studio IDE	Texas Instruments	

15.0 PROCEDURE:

To open the CCS window: Double click on CCS icon in the desktop

Create a new project: ***Project→New***

Type the project name(**project_name**) & store in myprojects folder.

To open an existing project: ***Project →Open***

Files to be added to the project: ***Project→Add*** files to project

a) C:\ti\c5400\cgtools\lib\rts.lib

<Select type of the file as : library(.lib)>

b) C:\ti\tutorial\dsk5416\datedisplay\mainapplication

<Select type of the file as : linkar(.cmd)>

To write program: ***File→New→Source*** file

To save: ***File→save***

To compile : ***Project→Compile File***

To build the project : ***Project→Build***

<Obtained Build successful >

To execute: ***File→Load program***

< Select project_name.out file>

To Run : ***Debug→Run***

Observe the output on the stdout window or waveform using graph or CRO.

To view waveforms: ***View→Graph→time/frequency***

Changes in the graph property dialog box to be made

a) Display type→Dual(to observe 2 waveforms)
b) Title → User defined.
c) Start address-upper display→x
<User defined variable array names used in C program.
Note: x,y should be made global to be used by graph>
d) Start address-lower display→y
e) Acuisition Buffer size →60
<depends on the array size>
f) Display Buffer size→60
<depends on the requirments>
g) DSP data type- 32 bit floating point
h) Auto scale → ON
<if off choose max y value=1>

16.0 SAFETY:
NOT APPLICABLE

17.0 DISPOSAL:
NOT APPLICABLE

9.0 REPORT WRITING:

9.1 Attach the rough note with your final report.
9.2 The 1st Page of the report shall be as per the format shown in Annexure – I.
9.3 Write your final report as per the Work Instruction

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE: _____

_____ _____

Marks Obtained

Signature of the
Sessional in - charge

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403/11

2.0 NAME OF JOB/EXPERIMENT: Writing small programs in VHDL/Verilog and downloading onto Xilinx FPGA.

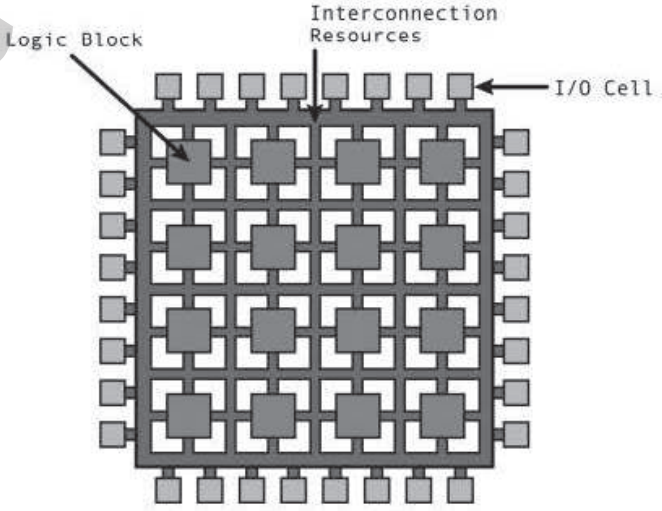
3.0 OBJECTIVE: To write a program in HDL and thereby download the bit stream file on to Xilinx FPGA.

4.0 PRINCIPLE:

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast IOs and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. Planning enables resources allocation within FPGA to meet these time constraints. FPGAs can be used to implement any logical function that an ASIC can perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non-recurring engineering costs relative to an ASIC design offer advantages for many applications.

FPGAs consist of three major resources:

- Configurable Logic blocks (CLB)
- Routing blocks (Programmable Interconnect)
- I/O blocks



FPGA Architecture

Configurable Logic Blocks (CLBs): These blocks contain the logic for the FPGA. CLBs contain RAM for creating arbitrary combinatorial logic functions, also known as lookup tables (LUTs). It also contains flip-flops for clocked storage elements, along with multiplexers in order to route the logic within the block and to and from external resources. The multiplexers also allow polarity selection and reset and clear input selection.

Programmable Interconnect (Routing Blocks): In the above figure a hierarchy of interconnect resources can be seen. There are lines that can be used to connect the CLBs present on the chip without inducing much delay. These lines can also be used as buses within the chip. There are also short lines that are used to connect individual CLBs that are located physically close to each other. Transistors are used to turn on or off connections between different lines. There are also several programmable switch matrices in the FPGA to connect these long and short lines together in specific, flexible combinations. Three-state buffers are used to connect many CLBs to a long line, creating a bus. Special long lines, called global clock lines, are specially designed for low impedance and thus fast propagation times.

These are connected to the clock buffers and to each clocked element in each CLB. This is how the clocks are distributed throughout the FPGA, ensuring minimal skew between clock signals arriving at different flip-flops within the chip. In an ASIC, the majority of the delay comes from the logic in the design, because logic is connected with metal lines that exhibit little delay. In an FGPA, however, most of the delay in the chip comes from the interconnect, because the interconnect – like the logic – is fixed on the chip. In order to connect one CLB to another CLB in a different part of the chip often requires a connection through many transistors and switch matrices, each of which introduces extra delay.

Configurable I/O Blocks: A Configurable input/output (I/O) Block is used to bring signals onto the chip and send them back off again. It consists of an input buffer and an output buffer with three-state and open collector output controls. Typically there are pull up resistors on the outputs and sometimes pull down resistors that can be used to terminate signals and buses without requiring discrete resistors external to the chip.

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME
01	Computer	
02	Xilinx ISE	Xilinx
03	Xilinx FPGA Board (Spartan-3)	Xilinx

6.0 PROCEDURE:

6.1 To start ISE, double-click the desktop icon or start ISE from the Start menu by selecting: **Start → All Programs → Xilinx ISE 10.1→ Project**.

6.2 To create a new project:

Select File > New Project... The New Project Wizard appears. Type tutorial in the Project Name field. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically. Verify that HDL is selected from the Top-Level Source Type list. Click Next to move to the device properties page.

Fill in the properties in the table as shown below:

- Product Category: All
- Family: Spartan3
- Device: XC3S200
- Package: FT256
- Speed Grade: -4
- Top-Level Source Type: HDL
- Synthesis Tool: XST (VHDL/Verilog)
- Simulator: ISE Simulator (VHDL/Verilog)
- Preferred Language: Verilog (or VHDL)

Verify that Enable Enhanced Design Summary is selected.

Leave the default values in the remaining.

6.3 To create a VHDL/Verilog source file for the project:

- Click the New Source button in the New Project Wizard.
- Select VHDL/Verilog Module as the source type.
- Type in the file name.
- Verify that the Add to project checkbox is selected.
- Click Next.
- Declare the ports for the program design by filling in the port.
- Click Next, then Finish in the New Source Wizard - Summary dialog box to complete.

The source file containing the entity/architecture pair displays in the Workspace.

6.4 Final Editing of the Verilog/VHDL Source:

- Write down the code in Verilog/VHDL in the workspace.
- Save the file by selecting File → Save.

6.5 Checking the syntax of the new VHDL/Verilog module:

- When the source files are complete, check the syntax of the design to find errors and typos.
- Verify that Implementation is selected from the drop-down list in the Sources window.
- Select the source file in the Sources window to display the related processes in the Processes window.
- Click the "+" next to the Synthesize-XST process to expand the process group.
- Double-click the Check Syntax process.
- Close the HDL file.

6.6 Design Simulation:

- Create a test bench waveform containing input stimulus you can use to verify the functionality of the source module. The test bench waveform is a graphical view of a test bench. Create the test bench waveform as follows:
- Select the counter HDL file in the Sources window.
- Create a new test bench source by selecting Project → New Source.
- In the New Source Wizard, select Test Bench Waveform as the source type, and type <source file>_tbw in the File Name field. Click Next.
- The Associated Source page shows that you are associating the test bench waveform with the source file counter. Click Next.
- The Summary page shows that the source will be added to the project, and it displays the source directory, type, and name. Click Finish.
- You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens.
- Click Finish to complete the timing initialization.
- The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the design.
- Save the waveform.
- In the Sources window, select the Behavioral Simulation view to see that the test bench waveform file is automatically added to your project.
- Close the test bench waveform.

6.7 Simulating Design Functionality:

Verify that the design functions as you expect by performing behavior simulation as follows:

- Verify that Behavioral Simulation and <source_file>_tbw are selected in the Sources window.
- In the Processes tab, click the "+" to expand the Xilinx ISE Simulator process and double-click the Simulate Behavioral Model process.
- The ISE Simulator opens and runs the simulation to the end of the test bench. To view your simulation results, select the Simulation tab and zoom in on the transitions.

6.8 Create Timing Constraints:

To constrain the design do the following:

- Select Implementation from the drop-down list in the Sources window. Select the HDL source file.
- Click the "+" sign next to the User Constraints processes group, and double-click the Create Timing Constraints process.

- ISE runs the Synthesis and Translate steps and automatically creates a User Constraints File (UCF). You can also customize the UCF.

6.9 Implement Design:

Select the source file in the Sources window.

- Open the Design Summary by double-clicking the View Design Summary process in the Processes tab.
- Double-click the Implement Design process in the Processes tab.
- Notice that after Implementation is complete, the Implementation processes have a green check mark next to them indicating that they completed successfully without errors or warnings.
- Locate the Performance Summary table near the bottom of the Design Summary. Close the Design Summary.

6.10 Assigning Pin Locations:

To constrain the design ports to package pins, do the following:

- Verify that source file is selected in the Sources window.
- Double-click the Floor plan Area/IO/Logic - Post Synthesis process found in the User Constraints process group. The Xilinx Pin out and Area Constraints Editor (PACE) opens.
- Select the Package View tab.
- In the Design Object List window, enter a pin location for each pin in the Loc column, Notice that the assigned pin locations are shown in blue in the package view.
- Select File → Save.
- Close PACE.

6.11 Reimplement Design and Verify Pin Locations:

First, review the Pin out Report from the previous implementation by doing the following:

- Open the Design Summary by double-clicking the View Design Summary process in the Processes window.
- Select the Pin out Report and select the Signal Name column header to sort the signal names. Notice the Pin Numbers assigned to the design ports in the absence of location constraints.
- Reimplement the design by double-clicking the Implement Design process.
- Select the Pin out Report again and select the Signal Name column header to sort the signal names.
- Verify that signals are now being routed to the correct package pins. Close the Design Summary.

6.12 Download Design to the Spartan™-3 Demo Board:

This is the last step in the design verification process.

- Connect the 5V DC power cable to the power input on the demo board (J4).
- Connect the download cable between the PC and demo board (J7).
- Select Implementation from the drop-down list in the Sources window.
- Select the source file in the Sources window.
- In the Process window, double-click the Configure Target Device process.
- The Xilinx Web Talk Dialog box may open during this process. Click Decline.
- iMPACT opens and the Configure Devices dialog box is displayed.
- In the Welcome dialog box, select Configure devices using Boundary-Scan (JTAG).
- Verify that "Automatically connect to a cable and identify Boundary-Scan chain" is selected.
- Click Finish.
- If you get a message saying that there are two devices found, click OK to continue.
- The devices connected to the JTAG chain on the board will be detected and displayed

in the iMPACT window.

- The Assign New Configuration File dialog box appears. To assign a configuration file to the xc3s200 device in the JTAG chain, select the counter.bit file and click Open.
- If you get a Warning message, click OK.
- Select Bypass to skip any remaining devices.
- Right-click on the xc3s200 device image, and select Program... The Programming Properties dialog box opens.
- Click OK to program the device.
- When programming is complete, the Program Succeeded message is displayed. On the board, if LEDs are lit accordingly, then it is confirmed that the design is working properly.
- Close iMPACT without saving.

7.0 Disposal: not applicable

8.0 Safety:

8.1 Handle the Xilinx FPGA board with proper care.

8.2 Connect the JTAG & Power cable properly before downloading the bit stream file.

8.3 Close the project file before closing the Xilinx ISE tool otherwise it may get damaged.

9.0 Report Writing:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – I.

. **9.3** Write the working principle of your experiment & write the corresponding VHDL/Verilog program.

9.4 Write your final report as per Work Instruction.

ANNEXURE-I

NAME: _____

ROLL NO.: _____ DEPARTMENT: _____

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME ROLL NO.

1. _____ _____

2. _____ _____

3. _____ _____

4. _____ _____

5. _____ _____

TITLE _____

OBJECTIVE: _____

_____ _____

Marks Obtained

Signature of the
Sessional in - charge

WORK INSTRUCTION

1.0 JOB/EXPERIMENT NO.: PC-RE/P/403/12

2.0 NAME OF JOB/EXPERIMENT: Mapping of some DSP algorithms onto FPGA.

3.0 **OBJECTIVE:** To map some DSP algorithms onto Xilinx FPGA.

4.0 PRINCIPLE:

In this experiment HDL coding for linear convolution has been done and it is mapped onto Xilinx FPGA board.

Linear convolution theorem:

Convolution is a mathematical way of combining two signals to form a third signal. It is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response. Linear convolution is performed to obtain the output response y[n] of a LTI system, when an input stimulus x[n] is given to it & the impulse response for the system is h[n]:

Where * means linear convolution operation.

The block diagram representation for linear convolution is depicted below:

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable".

FPGAs consist of three major resources:

1 Configurable Logic blocks (CLB)

2 Routing blocks (Programmable Interconnect)

3 I/O blocks

5.0 APPARATUS REQUIRED:

SL.NO.	ITEM	MAKERS NAME
01	Computer	
02	Xilinx ISE	Xilinx
03	Xilinx FPGA Board (Spartan-3)	Xilinx

6.0 PROCEDURE:

6.1 Start ISE.

6.2 Create a new project.

6.3 Create a VHDL/Verilog source file for the project. Write the following HDL code for linear convolution.

```
module conv(
input [3:0] x,
input [3:0] h,
output reg [6:0]y
);
reg [3:0]i,j;
reg [6:0]x1,h1;
always @(*)
begin
 x1=x;
 h1=h;
    for(i=4; i<7;i=i+1)
      x1[i]=7`b0;
    for(i=4; i<7;i=i+1)
      h1[i]=7`b0;
    for(i=0; i<7;i=i+1)
     begin
       y[i]=7`b0;
       for(j=0; j<=i;j=j+1)
        begin
          y[i]=y[i]+(x1[j]*h1[i-j]);
        end
     end
end
endmodule
```

6.4 Check the syntax of the new VHDL/Verilog module.
6.5 Simulate the design functionality to check whether the design logic is working fine.
6.6 Create Timing Constraints:
6.7 Implement the design.
6.8 Assign Pin Locations.
6.9 Reimplement the design and verify Pin Locations.
6.10 Download the bit-stream file onto the Spartan™-3 Demo Board.

7.0 Disposal: not applicable

8.0 Safety:

8.1 Handle the Xilinx FPGA board with proper care.
8.2 Connect the JTAG & Power cable properly before downloading the bit stream file.
8.3 Close the project file before closing the Xilinx ISE tool otherwise it may get damaged.

9.0 Report Writing:

9.1 Attach the rough note with your final report.

9.2 The 1st Page of the report shall be as per the format shown in Annexure – 1.

9.3 Write the working principle of your experiment & write the corresponding VHDL/Verilog program.

9.4 Write your final report as per Work Instruction.

ANNEXURE-1

NAME: _____

ROLL NO.: _____ DEPARTMENT:

DATE OF SUBMISSION: _____ DATE OF EXPERIMENT: _____

CO-WORKER

NAME	ROLL NO.
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

TITLE _____

OBJECTIVE:

Marks Obtained

Signature of the

Sessional in - charge