DSP LAB MANUAL

January 22, 2015

Contents

1 PRE-REQUIREMENTS BEFORE STARTING LAB

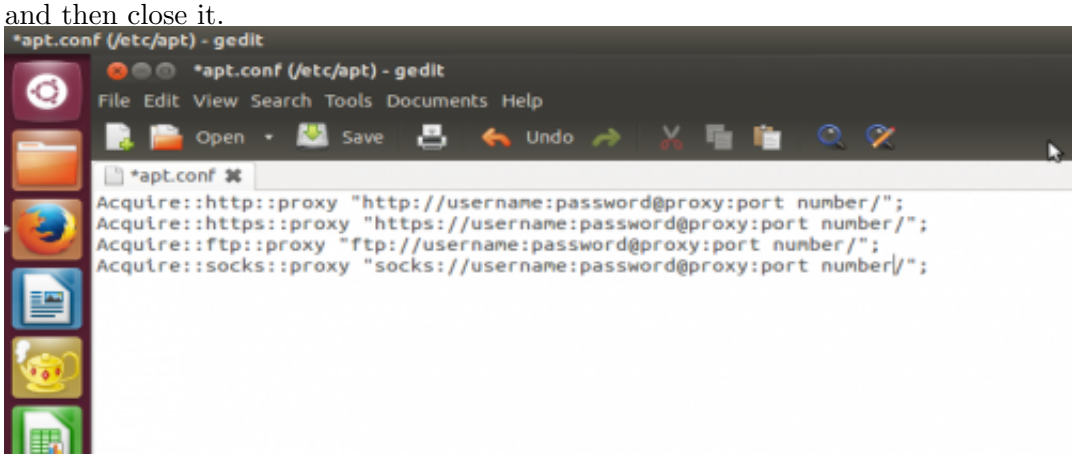
1.1 Get ready with Ubuntu 12.04 or 14.04

Once Ubuntu was done open the terminal and try running the following commands.
How to open the terminal?
Go to dash home (top left icon) and type terminal and then open it.
It can also be opened using the short cut (Ctrl+Alt+T)
If you are under proxy then run the following commands.

- sudo gedit /etc/apt/apt.conf

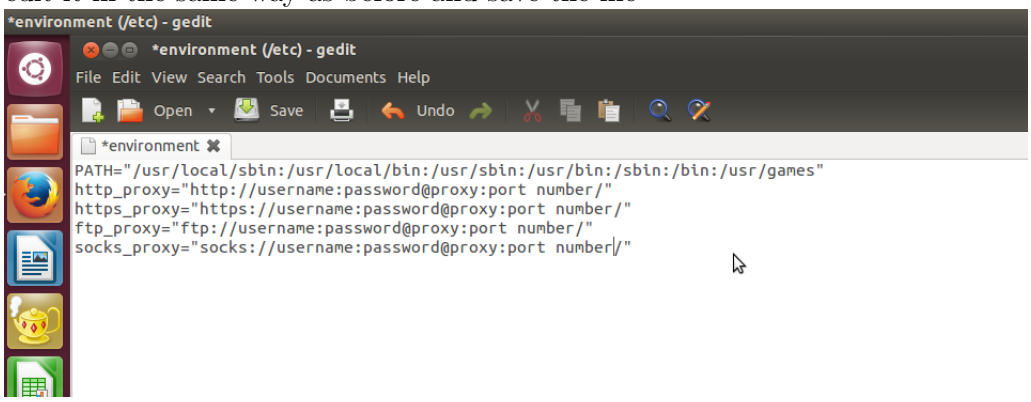
  now edit it as

  username:password@proxy:port within the quotes at all the four places and save the file and then close it.



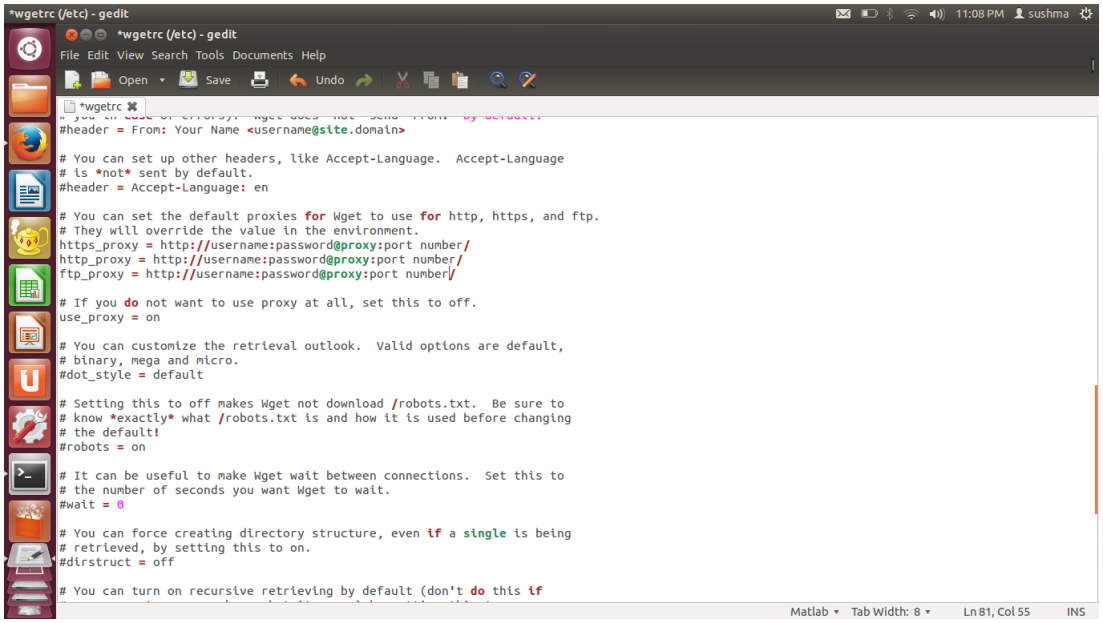
- sudo gedit /etc/environment

  edit it in the same way as before and save the file



- sudo gedit /etc/wgetrc

  edit the content in the similar way as before and remove the ' # ' symbol before the proxy on line and then save the file.



By doing so we can access internet using the terminal.
Now update the system by running sudo apt-get update command in the terminal.

1.2 Install geany editor

It can be done by writing the following command in the terminal
sudo apt-get install geany
After installing reboot the system by running **sudo reboot** command in the terminal.

1.3 Install sagemath

For installing sagemath go through the following steps

1. Visit www.sagemath.org site using your browser.
2. Go to download 6.4.1 version.
3. In this follow the Ubuntu PPA session and run the commands one by one in your termianl

   ie., type sudo before each command and run them one by one.

   To paste anything in terminal use Ctrl+Shift+V

   if the above process doesn't work then try with sudo -E instead of sudo.

With this you will be done installing your sage.

1.4 Interface geany with sagemath

copy the files filetype_extensions.conf and filetypes.sage.conf in your home folder and run the commands in the terminal

sudo cp filetype_extensions.conf .config/geany/
sudo cp filetypes.sage.conf .config/geany/filedefs/

1.5 Installing code composer studio

To install code composer studio one has to follow the steps
Copy the CCS file and the BIOS file in the home and run the following commands in your terminal one by one

1.5.1 If you are using Ubuntu 12.04 follow the steps

cd CCS5.5.0.00077_linux
sudo chmod +x ccs_setup_5.5.0.00077.bin
sudo ./ccs_setup_5.5.0.00077.bin
After executing this you will get a window of installation.
Select custom and choose C6000 multi core family and then install.
cd ..
cd BIOS-MCSDK_2.1.2.5
sudo chmod +x bios_mcsdk_02_01_02_05_setuplinux.bin
sudo ./bios_mcsdk_02_01_02_05_setuplinux.bin
By this the Bios file also gets installed.
cd /opt/ti/ccsv5/install_scripts/
sudo ./install_drivers.sh
sudo ln -s /opt/ti/ccsv5/eclipse/ccstudio /usr/bin
sudo ldconfig

1.5.2 If you are using Ubuntu 14.04 follow the steps

If you are using Ubuntu 14.04 then install prerequisites by pasting the following commands in terminal..
sudo apt-get install libc6-i386 libx11-6:i386 libc6-i386 libasound2:i386 libjpeg62:i386 libatk1.0-0:i386 libcairo2:i386 libdbus-1-3:i386 libdbus-glib-1-2:i386 libfontconfig1:i386 libfreetype6:i386 libgconf-2-4:i386 libgdk-pixbuf2.0-0:i386 libgtk2.0-0:i386 libice6:i386 lib32ncurses5 liborbit2:i386 libpango-1.0-0:i386 libpangocairo-1.0-0:i386 libpangoft2-1.0-0:i386 libpng12-0:i386 libsm6:i386 lib32stdc++6 libusb-0.1-4:i386 libx11-6:i386 libxext6:i386 libxi6:i386 libxrender1:i386 libxt6:i386 libxtst6:i386 lib32z1 libgnomevfs2-0:i386 libcanberra-gtk-module:i386
Also, install libudev0 for your machine(i386/x86_64). If you don't know your version, type following command in terminal. uname -a download libudev0 from ubuntu website by typing following link in browser. https://launchpad.net/ubuntu/+source/udev/175-0ubuntu19/+build/4325788/+files/libudev0_175-0ubuntu19_amd64.deb (for 64 bit)
https://launchpad.net/ubuntu/+source/udev/175-0ubuntu19/+build/4325790/+files/libudev0_175-0ubuntu19_i386.deb (for 32 bit)
click on the downloaded archive, It will redirect you to ubuntu software centre page, there you will see install option, click it(it may ask you provide authentication, give your password).
Then follow the same steps as that of Ubuntu 12.04

2 HOW TO USE GEANY EDITOR

- Geany is just a editor to make files.
- Geany may be opened by typing geany in the dash home or typing geany in the terminal.
- One can make any type of files using geany i.e., c files, sage files etc., and they will be differed just by their extensions.

If you are writing a C file then name with '.c' extension and if you are writing sage file then name it with .sage extension

.sage is nothing but writing files using python.

For source codes of the sagemath programs visit `http://ee.iith.ac.in/gadepall/index.php/dsp-lab`

3 HOW TO USE CODE COMPOSER STUDIO

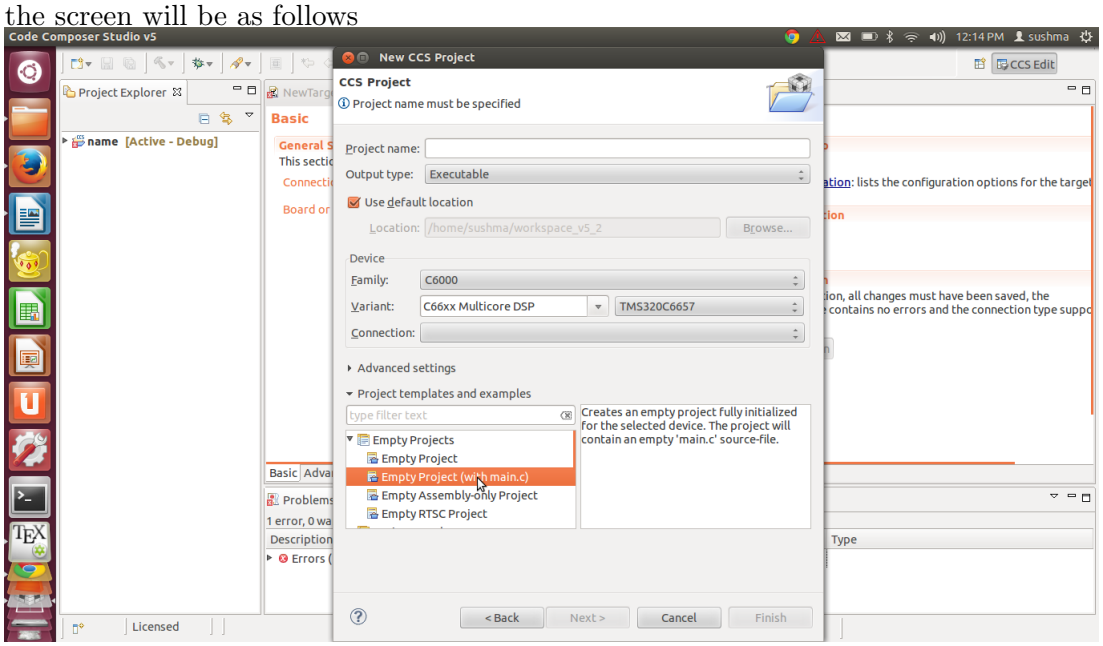
3.1 For simulator

- CC Studio is used for writing both C codes and the Assembly codes.
- C codes is nothing but writing the codes in C language and saving them using .c extension.
- To make these new project is to be created it may be done by following procedure.

  When you open the ccstudio for the first time it will ask the directory where to store the files Once you open it select

  file $\longrightarrow$ new $\longrightarrow$ CCS project

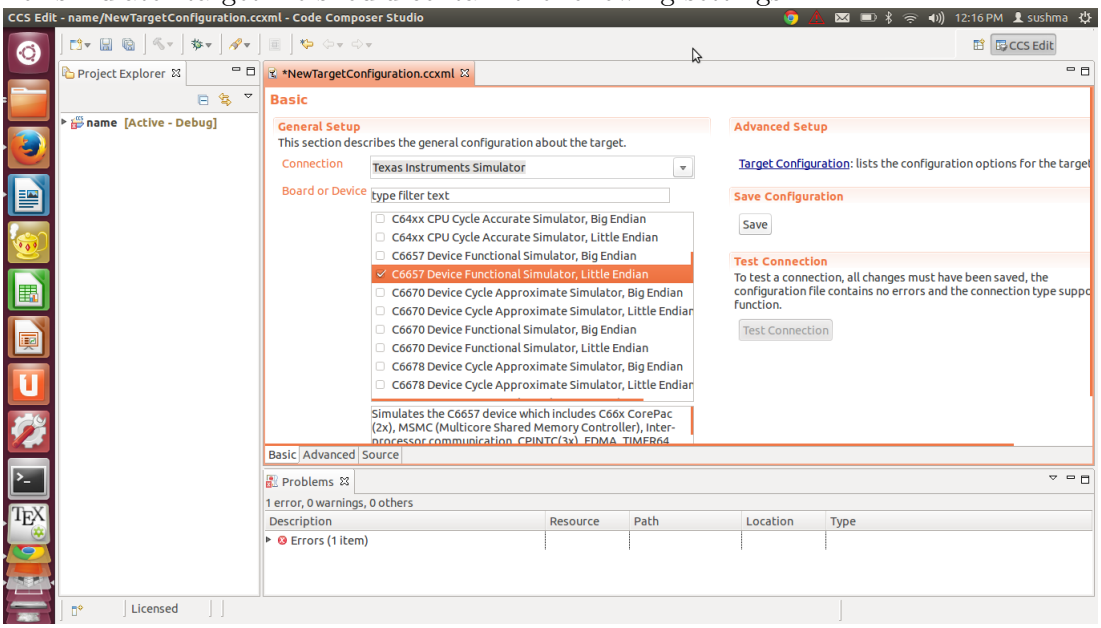
  the screen will be as follows



  - Write the project name and the default main.c file is created.
  - The remaining settings are to be made according to the above screen shot.
  - Now edit the main file according to your program statement.

– Create a target file by clicking right button and select new and add a target configuration file.

– For simulator target file should contain the following settings.



Now your project may be built using the hammer symbol in the top or by using build command in project.

Open the console from the project. Console is the place where the building details are shown.

Now debug the program by clicking on the debug symbol or by selecting debug in run.

The output of the program is being displayed in the console.

3.2 Details regarding various types of assembly files

CC Studio is used for making assembly files.
It can perform three types of assembly operations

- Assembly
- Inline Assembly
- Linear Assembly

1.Assembly

- For this follow the same steps as that for writing c files but instead of empty project (with main.c) select empty project.
- Now it creates an empty project and click the right button on the project add file and name it with .asm extension which states that it is an assembly file.
- Write the assembly code and then execute it in the same way as that of c program

2.Inline Assembly

- This contains both c file and the assembly file. We can make use of registers A0-A15 and B0-B15 here.

- Here input arguments are being passed through the c program and the function is being performed in the assembly and the outputs are retrieved to the c program.
- Here the assembly file is named with .asm extension.

3.Linear Assembly

- Its operation is same as that of the inline assembly but the only difference is that the assembly file is named with .sa extension instead of .asm.
- Here not only the basic registers A0-A15 and B0-B15 but also any type of registers can be used by declaring them at the top of the program.
- It is simple compared to that of inline and parallel processing is inbuilt and there is no need of mentioning the parallel instructions and the NOP instructions.

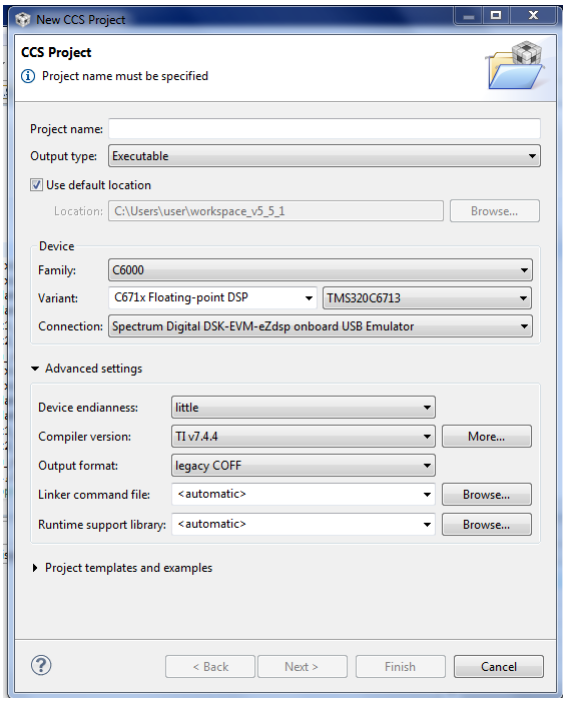
3.3 For emulator

All the above things mentioned may be performed using the simulator or emulators.

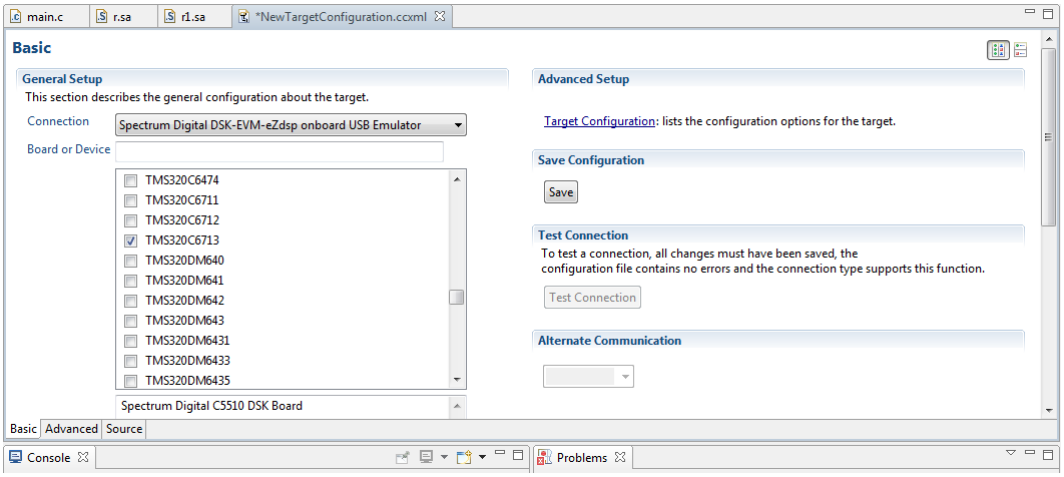
Simulator is nothing but direct implementation but the emulator means the real time on-board processing.

We will be using TI 6713 kit in our lab.

For this emulator the following properties are to be modified while creating the project select spectrum digital on-board emulator in connections.



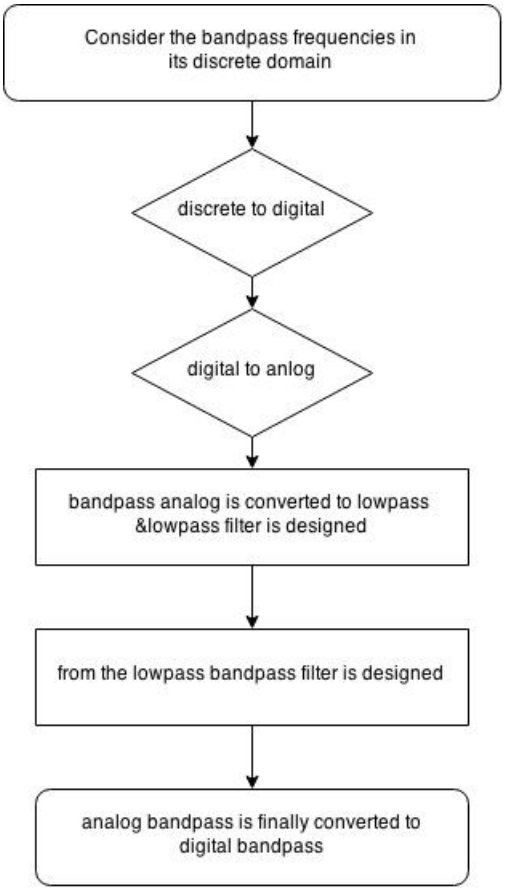
Then while creating the target file selection to be spectrum digital on-board emulator and the board or device to be TMS320C6713.



For source code of ccstudio filter design visit `http://ee.iith.ac.in/gadepall/index.php/dsp-lab`

4 FILTER DESIGN

- Filter coefficients can be calculated from their transfer function equations.
- In the file attached filter coefficients for chebyshev's filter are present.
- In this initially the bandpass frequencies are being considered which are in discrete domain and are being converted to digital domain.
- Now these frequencies are being converted to analog.
- These bandpass frequencies are now converted to lowpass specifications and a lowpass filter is being designed.
- From the lowpass design we get bandpass design.
- This is then transformed to digital domain using the filter.



Same steps may be followed for any sort of filters being designed.

5 MATLAB CODE FOR CALCULATING THE FILTER COEFFICIENTS

```
syms s; % declaration of variable
H(s)=(s*s)+2*s+1; %declaration of transfer function
G(s)=subs(H(s),s,(s-1)); %replacing the variable with other variable
G(s)=collect(G(s)); %collects the individual coefficients
T(s)=vpa(G(s)); %simplify the fractions
```

6 CODES FOR IMPLEMENTING CHEBYSHEV FILTER DESIGN

6.1 Analog Lowpass filter design with e between 0.35 and 0.6

6.1.1 C code for Lowpass filter design

```
#include<stdio.h>
#include<math.h>
int main()
double h[1000];
int count=0;
float e,w;
FILE *fp;
fp=fopen("lpf.dat","w");
for (e = 0.35; e ≤ 0.60; e+ = 0.05)
{
for(w = 0.001; w ≤ 2; w+ = 0.02)
{
if(w ≤ 1)
h[count]=sqrt(1/(1+pow(((e*(cos(4*acos(w))))x),2)));
else
h[count]=sqrt(1/(1+pow(((e*(cosh(4*acosh(w))))),2)));
if(w ≤ 1.999999)
{
printf("%f \ t%f \ n", w, h[count]);
fprintf(fp,"%f \ t%f \ n", w, h[count]);
count=count+1;
}
else
{
fprintf(fp,"2 \ t0 \ n");
}
}
fprintf(fp,"0 \ t0 \ n");
}
return 0;
}
```

Here the data is being stored into a dat file and is called using sage program through which low pass filter is constructed.

Python code for designing lowpass filter using dat file generated in C

```
from sage.all import *
import matplotlib.pyplot as plt
plt.plotfile('lpf.dat',delimiter=' ', cols=(0, 1), names=('w','h(w)'), marker=' ')
plt.grid()
plt.savefig('lpf.eps')
```

6.1.2 Python code for designing a lowpass filter

(In python '#' is used for comments)
#!/usr/bin/sage -python
#import sys
from sage.all import *
#The above command brings everything from sage to python
import mpmath as mp
import numpy as np
import matplotlib.pyplot as plt
h1= []
h2= []
e=.35
f=.00
while $e \leq$ .60:
f=0
while f ¡ 2:
if $f \leq 1$:
h1.append(sqrt(1/(1+(pow((e*(cos(4*acos((f))))),2)))))
h2.append(f)
else:
h1.append(sqrt(1/(1+(pow((e*(cosh(4*acosh(f)))),2)))))
h2.append(f)
f=f+.01
h1.append(0)
h2.append(0)
e=e+.05
plt.plot(h2,h1)
plt.xlabel('frequency')
plt.ylabel('Gain')
plt.grid()
plt.legend(('$Analytical$'),loc='lower right')
plt.savefig('lpfsage.eps')

6.2 Lowpass filter design at a specific value of e matching design parameters and specifications

6.2.1 C code for lowpass

```
#include << stdio.h >
#include < math.h >
int main()
{
float w,h1,h2,x,y;
FILE *fp;
fp=fopen("lpfate.dat","w");
for (w = 0.000; w ≤ 2.00; w = w + 0.02)
{
x=8*pow(w,4)-8*pow(w,2)+1;
y=pow((pow(w,4)-1.6125*pow(w,2)+0.3366),2)+pow(((0.9140*w)-(1.1068*pow(w,3))),2);
h1=(1/sqrt(1+0.16*pow(x,2)));
h2=(0.3125/sqrt(y));
printf("%f \ t%f \ t%f \ n", w, h1, h2);
fprintf(fp, "%f \ t%f \ t%f \ n", w, h1, h2);
}
return 0;
}
```

Python code for designing such using dat file generated

```
from sage.all import *
import matplotlib.pyplot as plt
plt.plotfile('lpfate.dat',delimiter=' ', cols=(0, 1),
names=('w','h1(w)','h2(w)'),newfig=True,subplots=False,color='y',
marker='o',fillstyle='none',linestyle='none')
plt.plotfile('lpfate.dat',delimiter=' ', cols=(0, 1, 2),
names=('w','h1(w)','h2(w)'),newfig=False,subplots=False,color='r')
plt.grid()
plt.savefig('lpfate.eps')
```

6.3 Python code for lowpass

```
from sage.all import *
#The above command brings everything from sage to python
import mpmath as mp
import numpy as np
import matplotlib.pyplot as plt
h1= []
h2= []
h3= []
w=.00
w=0
while w¡2: h1.append((1/sqrt(1+0.16*pow(8*pow(w,4)-8*pow(w,2)+1,2))))
h2.append(0.3125/sqrt(pow((pow(w,4)-1.6125*pow(w,2)+0.3366),2)+pow(((0.9140*w)-
(1.1068*pow(w,3))),2)))
h3.append(w)
```

```
w=w+0.02
plt.plot(h3,h1,'yo')
plt.plot(h3,h2,'r')
plt.xlabel('frequency')
plt.ylabel('Gain')
plt.grid()
plt.legend(('h1(w)','h2(w)'),loc='upper right')
plt.savefig('lpfatesage.eps')
```

6.4 Bandpass filter from lowpassfilter

6.4.1 C code for bandpass

```
#include < stdio.h >
#include < math.h >
int main()
{
float w,h1,x;
FILE *fp;
fp=fopen("bpfate1.dat","w+");
for (w = −0.8; w ≤ 0.8; w = w + 0.001)
{
x=sqrt((pow((12123.510626*pow(w,8)-10412.1353788*pow(w,6)+3315.26827045*(pow(w,4))-
463.771193575*(pow(w,2))+24.0522840629),2)
+pow((-1278.76413874*(pow(w,7))+819.233988929*(pow(w,5))-
172.89798982*(pow(w,3))+12.0208836518*w),2)));
h1=0.3125*(pow(w,4))/((x));
printf("%f \ t%f \ n", w, h1);
fprintf(fp,"%f \ t%f \ n", w, h1);
}
return 0;
}
```

Python code for design using dat file generated

```
from sage.all import *
import matplotlib.pyplot as plt
plt.plotfile('bpfate1.dat',delimiter=' ', cols=(0, 1), names=('w','h1(w)'), color='r', marker=' ')
plt.grid()
plt.savefig('bpfate2.eps')
```

6.4.2 Python code for bandpass

```
from sage.all import *
import numpy as np
import matplotlib.pyplot as plt
step=0.001
samples=1.6/step +1
h1=np.ndarray((samples))
h2=np.arange(-0.8,0.800001,step)
w=-0.8
```

```
for n in range(0,samples): h1[n]=sqrt((pow((12123.510626*pow(w,8)-
10412.1353788*pow(w,6)+3315.26827045*(pow(w,4))-
463.771193575*(pow(w,2))+24.0522840629),2)
+pow((-1278.76413874*(pow(w,7))+819.233988929*(pow(w,5))-
172.89798982*(pow(w,3))+12.0208836518*w),2)))
h1[n]=0.3125*(pow(w,4))/((h1[n]))
w=w+step
plt.plot(h2,h1,'r')
plt.xlabel('frequency')
plt.ylabel('gain')
plt.title('Analog Band Pass Filter')
plt.grid()
plt.show()
plt.savefig('BPFsage.eps')
```

6.5 Bandpass filter matching the given specifications(DIGITAL bandpassfilter)

6.5.1 C code for bandpass filter

```
#include < stdio.h >
#include < math.h >
int main()
{
float w,h1,a,x;
FILE *fp;
fp=fopen("bpfate1.dat","w+");
for (w = −0.6; w ≤ 0.6; w = w + 0.001)
{
a=tan(w*22/14);
x=sqrt((pow((12123.510626*pow(a,8)-10412.1353788*pow(a,6)+3315.26827045*(pow(a,4))-
463.771193575*(pow(a,2))+24.0522840629),2)
+pow((-1278.76413874*(pow(a,7))+819.233988929*(pow(a,5))-
172.89798982*(pow(a,3))+12.0208836518*a),2)));
h1=0.3125*(pow(a,4))/((x));
printf("%f \ t%f \ n", w, h1);
fprintf(fp,"%f \ t%f \ n", w, h1);
}
return 0;
}
```

Python code for filter design using dat file generated

```
from sage.all import *
import matplotlib.pyplot as plt
plt.plotfile('bpfate1.dat',delimiter=' ', cols=(0, 1), names=('w','h1(w)'), color='r', marker=' ')
plt.grid()
plt.savefig('bpfate2.eps')
```

6.5.2 Python Code for Bandpass filter

```
from sage.all import *
import numpy as np
import matplotlib.pyplot as plt
step=0.001
samples=1.2/step +1
h1=np.ndarray((samples))
h2=np.arange(-0.6,0.600001,step)
b=-0.6
for n in range(0,samples):
w=tan(b*22/14) h1[n]=sqrt((pow((12123.510626*pow(w,8)-
10412.1353788*pow(w,6)+3315.26827045*(pow(w,4))-
463.771193575*(pow(w,2))+24.0522840629),2)
+pow((-1278.76413874*(pow(w,7))+819.233988929*(pow(w,5))-
172.89798982*(pow(w,3))+12.0208836518*w),2)))
h1[n]=0.3125*(pow(w,4))/((h1[n]))
b=b+step
plt.plot(h2,h1,'r')
plt.xlabel('frequency')
plt.ylabel('gain')
plt.title('Digital Band Pass Filter')
plt.grid()
plt.savefig('BPFsage.eps')
```

EQUATIONS MAY BE SOLVED IN PYTHON USING THE FOLLOWING PROGRAM

```
from sage.all import *
import numpy as np
import matplotlib.pyplot as plt
s=var('s')
H(s)=0.3125/((s**4)+1.1068*(s**3)+1.6125*(s**2)+0.9140*(s)+0.3366)
G(s)=H(s).substitute(s=(((s**2)+0.21104836)/(0.0953*s))).full_simplify()
B(s)=G(s).substitute(s=((s-1)/(s+1))).full_simplify()
print B
```

6.6 Complete Chebyshev design using python

```
#import sys
from sage.all import *
#The above command brings everything from sage to python
import numpy as np
import mpmath as mp
import matplotlib.pyplot as plt
x1=[]
x2=[]
x=[]
y=[]
#f1=6500
#f2=10000
n=0;
```

```
v=0;
yval=0;
while(n < 20000):
v1=math.cos(0.850*n);
x1.append(v1);
v2=math.cos(1.308*n);
x2.append(v2);
if(n < 8):
v=0;
if(n ≥ 8):
v=v1+v2;
x.append(v);
if(n < 8):
yval=0
if(n ≥ 8):
yval=(1.2991*(0.00001)*(x[n]-4*x[n-2]+6*x[n-4]-4*x[n-6]+x[n-8])+6.0476945*y[n-1]-
16.0651779*y[n-2]+27.0883544*y[n-3]-31.653958*y[n-4]+25.936334*y[n-5]-14.727728*y[n-
6]+5.308319*y[n-7]-y[n-8])/1.1898016;
y.append(yval);
n=n+1;
n=np.linspace(0,20000,20000)
#plt.figure(1)
plt.subplot(221)
plt.plot(n[5400:5500],x1[5400:5500],'c')
plt.xlabel('n')
plt.ylabel('x1[n]')
plt.title('6.5k signal')
plt.subplot(222)
plt.plot(n[5400:5500],x2[5400:5500],'b')
plt.xlabel('n')
plt.ylabel('x2[n]')
plt.title('10k signal')
plt.subplot(223)
plt.plot(n[5400:5500],x[5400:5500],'g')
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('input signal')
plt.subplot(224)
plt.plot(n[5400:5500],y[5400:5500],'r')
#plt.savefig('filter.eps')
#plt.figure(2)
#plt.plot(n[5400:5450],y[5400:5450],'r')
plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('output signal')
plt.grid()
#plt.savefig('filter1.eps')
```

6.7 Chebyshev filter design using CCStudio

This is done using linear assembly its c and assembly files are as follows

6.7.1 C code part

```
#include < stdio.h >
#include < math.h >
float filternew1(float*,float*,float*, float*, float*, float*, float, float, int, int);
int main()
{
FILE *fp;
int i;
int N=150;
float G1=2.577697;
float G2=0.00001;
int xcoef=9;
float a[9]=1,0,-4,0,6,0,-4,0,1;
float b[9]=-1.984281,10.533198,-29.223951,51.464977,-62.810348,53.750907,-
31.877827,12.0003254,2.360901;
float x1[150],x2[150],v[150],y[150];
float w1,w2;
fp= fopen("out.txt","w+");
b[8]=1/b[8];
w1=2*3.14159*6500/48000.0;
w2=2*3.14159*10000/48000.0;
for(i=0;i¡N;i++)
{
x1[i]=sin(w1*i);
x2[i]=sin(w2*i);
v[i]=0;
y[i]=0;
}
filternew1(x1,x2,v,y,a,b,G1,G2,N,xcoef);
for(i=0;i¡N;i++)
{
fprintf(fp,"%d \ t%f \ t%f \ t%f \ t%f \ n", i, x1[i], y[i], x2[i], v[i]);
printf("x1 = %f \ tx2 = %f \ tv = %f \ ty = %f \ n", x1[i], x2[i], v[i], y[i]);
}
fclose(fp);
return 0;
}
```

6.7.2 Assembly part i.e., .sa file

```
.global filternew1
.text
filternew1: .cproc x1,x2,v,y,a,b,G1,G2,N,xcoef
.reg
num,bmul,rg1,rg3,rg4,m2,rg5,rg6,rg7,rg8,temp,iter,rg12,rg16,rg11,rg10,m1,ad1,gr,rg14,bb,rg15
MV x1,rg3
MV x2,rg4
MV v,rg5
MV N,temp
MV xcoef,num
MV y,rg12
MV b,rg15
ZERO rg1
ADD 8,rg1,rg1
MPY 4,rg1,rg1
ADD rg1,rg15,rg15
LDW *rg15++,bmul
MV rg5,rg16
SUB num,1,num
SUB N,num,iter
loop1: LDW *rg3++,rg8
LDW *rg4++,rg6
ADDSP rg8,rg6,rg7
STW rg7,*rg5++
SUB temp,1,temp
[temp] B loop1
loop4: ZERO ad1
MV xcoef,num
MV a,rg14
MV rg16,rg11
loop2: LDW *rg14++,bb
LDW *rg11++,gr
MPYSP bb,gr,m1
ADDSP ad1,m1,ad1
SUB num,1,num
[num] B loop2
MPYSP ad1,G1,ad1
MPYSP ad1,G2,ad1
MV b,rg15
MV rg12,rg10
SUB xcoef,1,num
loop3: LDW *rg10++,gr
LDW *rg15++,bb
MPYSP bb,gr,m1
ADDSP ad1,m1,ad1
SUB num,1,num
[num] B loop3
```

```
MPYSP ad1,bmul,ad1
STW ad1,*rg10++
LDW *rg12++,m2
LDW *rg16++,m2
SUB iter,1,iter
[iter] B loop4
.endproc
```

6.7.3 Matlab code for observing the filter output obtained

```
load('out.txt');%loads the file
a=out(:,1);%load the first column
b=out(:,5);%load the 5th column
plot(a,b);
title('filter output after bandpass filtering');
```