MODUL 5
Real-Time Filter FIR

1. Pendahuluan

Pada perkuliahan sebelumnya, anda telah berkenalan dengan istilah "filter analog" yang dirangkai dari resistor, kapasitor, induktor dan mungkin ditambahkan rangkaian *operational amplifier* (op-amp). Sekarang, anda akan mencoba membuat filter menggunakan program yang ditulis dengan bahasa-C. Bahasa program yang akan anda tulis digunakan untuk mewakili persamaan matematis filter baku yang telah dihitung oleh para ilmuwan sebelum kita lahir. Karena data yang diolah oleh filter ini berupa data digital (bilangan-bilangan) maka filter tersebut diberi nama "filter digital". Pertanyaan yang biasanya muncul adalah, "Kok bisa ya, persamaan matematis bisa meredam sinyal? Dunia ini memang benar-benar aneh…."

2. Tujuan

Setelah menyelesaikan praktikum ini, yang anda peroleh adalah :
- dapat menjelaskan cara mendisain filter FIR
- dapat menjelaskan cara mendapatkan nilai koefesien filter FIR menggunakan Matlab
- dapat mengimplementasikan disain filter FIR pada DSK TMS320C5402 menggunakan bahasa-C

3. Gambaran Disain

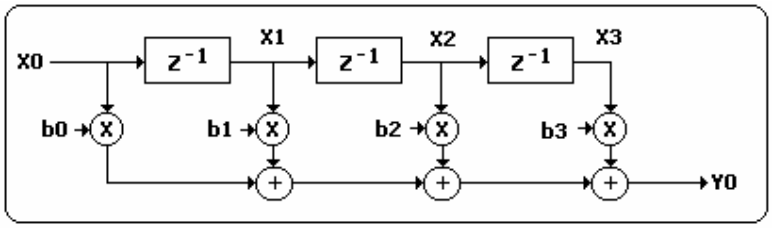
Untuk membuktikan terjadinya proses filter, nanti anda menggunakan function generator sebagai sinyal input. Sinyal input masuk kedalam ADC lalu diproses didalam DSP dan akhirnya hasil proses dikeluarkan melalui DAC untuk ditampilkan pada oscillosope. DSP tersebut telah di-download dengan program filter yang telah anda buat dengan bahasa-C. Filter digital memerlukan nilai-nilai konstan yang disebut koefesien filter. Untuk mendapatkan koefesien filter ini dengan cepat, Matlab akan membantu anda. Koefesien filter tersebut nantinya akan dikonvolusikan dengan sinyal input untuk menghasilkan sinyal output. Pahamilah dengan benar, bagaimana menerjemahkan blok diagram filter menjadi program-C.

4. Dasar Teori

Filter FIR adalah salah satu tipe dari filter digital yang dipakai pada aplikasi Digital Signal Processing (DSP). FIR kepanjangan dari Finite Impulse Response. Mengapa disebut respons impulsnya terbatas (finite)? Karena *tidak ada feedback* didalam filter, jika anda memasukkan sebuah impulse (yaitu sebuah sinyal '1' diikuti dengan banyak sinyal '0'), sinyal nol akan keluar setelah sinyal 1 melewati semua delay line dengan koefisiennya.

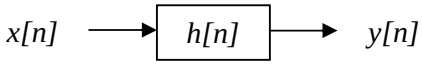
Keuntungan filter FIR antara lain adalah stabil dan memiliki phasa yang linier. Sedangkan kerugiannya adalah filter FIR terkadang membutuhkan lebih banyak memory dan/atau perhitungan untuk mencapai karakteristik respon filter yang diberikan. Dan juga, respon tertentu tidak mudah dilaksanakan untuk diimplementasikan dengan filter FIR.

Flow graph dari filter FIR ditunjukkan oleh Gambar 1.



Gambar 1. Flow graph filter FIR orde 3

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



Untuk filter FIR: $h[n] = \{ b_0 \;\; b_1 \;\; \ldots \;\; b_q \}$

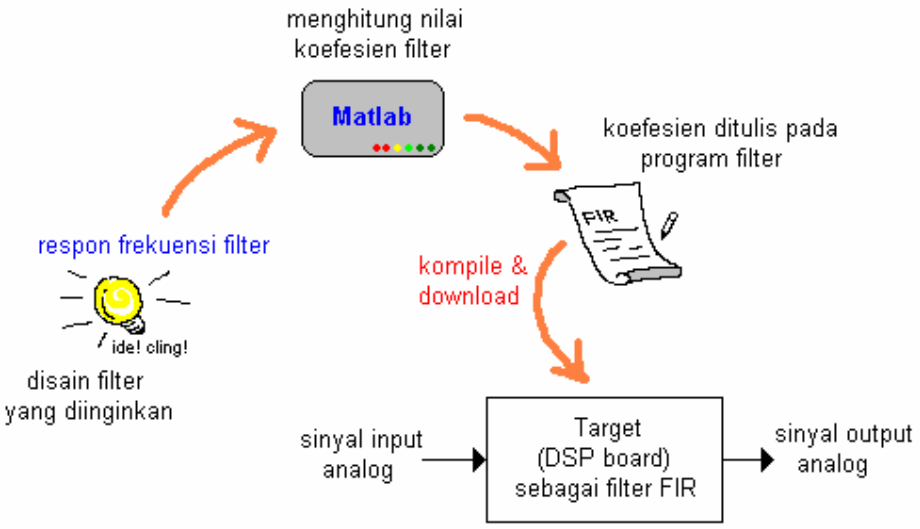
n=0

$$y[n]= \sum_{m=0}^{q} b_m x[n-m]$$

Bagaimana mengimplementasikan filter FIR pada DSP TMS320C5402 ?

Secara singkat, tahapan-tahapan untuk membuat filter digital FIR pada praktikum ini adalah:

1. Menentukan respon frekuensi filter yang diinginkan
2. Menghitung nilai koefesien filter dengan Matlab
3. Menuliskan koefesien filter kedalam program filter
4. Mengkompilasi program dan mendownload kode mesin ke DSP
5. Menguji sistem dengan memberikan sinyal input dari function generator dan melihat sinyal output pada oscilloscope



Gambar 2. Ilustrasi alur implementasi filter FIR

Menghitung koefesien filter FIR menggunakan fungsi FIR2 pada Matlab

Inti dari disain filter FIR adalah perhitungan koefesien. Cara termudah adalah menggunakan bantuan Matlab. Pada praktikum ini digunakan fungsi FIR2 yang disediakan oleh Matlab.

Perintah `B = FIR2(N,F,M);` pada Matlab digunakan untuk mendisain filter digital FIR orde N dengan respon frekuensi yang ditentukan oleh vektor F dan M, dan menghasilkan koefesien filter pada vektor B sebanyak N+1. Vektor F dan M menetapkan frekuensi dan *magnitude* untuk filter sehingga bila dilakukan perintah `PLOT(F,M)` akan dihasilkan gambar dari respon frekuensi yang diinginkan. Nilai frekuensi pada vektor F haruslah bernilai antara 0.0 < F < 1.0, dimana *1.0 menunjukkan setengah dari frekuensi sampling*. Nilai pada vektor F harus bertambah, dimulai dari 0.0 dan berakhir pada 1.0.

Secara *default*, FIR2 menggunakan window jenis Hamming. Tipe window yang lain seperti Boxcar, Hanning, Barlett, Blackman, Kaiser dan Chebwin juga dapat digunakan. Misalnya `B=FIR2(N,F,M,Blackman(N+1))` menggunakan window Blackman.

Langkah-langkah mendisain filter FIR low-pass menggunakan FIR2

1. Pilih orde filter, misal N=10
2. Menentukan vektor F dan M yang menujukkan bentuk dari respon frekuensi filter. Nilai F bernilai antara 0.0 < F < 1.0, dimana 1.0 adalah setengah dari frekuensi sampling. Jumlah vekor F dan M harus sama.
3. Koefesien filter dapat dihitung menggunakan perintah FIR2 pada Matlab.
4. Simpan nilai koefesien pada file, yang nantinya digunakan untuk mengimplementasikan filter FIR dengan konvolusi pada pemrograman DSP.

Program Matlab untuk menghasilkan koefesien filter menggunakan FIR2

Berikut contoh disain program Matlab untuk menghasilkan koefesien filter FIR low-pass dengan frekuensi cut-off 2KHz pada frekuensi sampling sebesar 16KHz. Nilai koefesien yang dihasilkan akan disimpan dalam file bertipe teks.

Pertama, tentukan berapa frekuensi sampling yang akan digunakan

Frekuensi sampling filter = 16KHz

Setengah frekuensi sampling filter = 8KHz

Nilai vektor F:

nilai 1.0 pada vektor F menunjukkan angka 8KHz

```
F = [0.0   ...   1.0];
```

Kedua, tentukan berapa frekuensi cut-off filter

Frekuensi cut-off filter = 2KHz

Sehingga titik cut-off ini pada vektor F jatuh disekitar

$$\frac{\text{frek.cuoff}}{\frac{1}{2}\text{frek.sampling}} = \frac{2\text{KHz}}{8\text{KHz}} = 0.25$$

Ketiga, menuliskan nilai vektor F dan vektor M

Maka nilai untuk vektor F dan M adalah:

nilai 1 berarti sinyal diloloskan, nilai 0 berarti sinyal diredam, titik cut-off harus berada di frekuensi 0.25

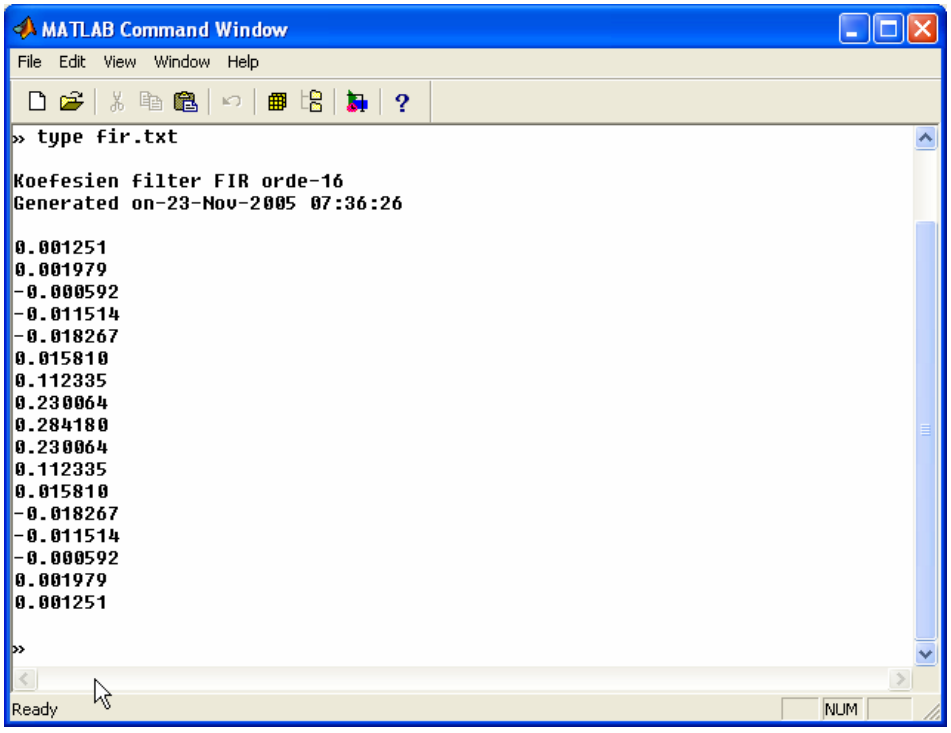
```
M = [1    0.8   0    0    0    0 ];
F = [0.0  0.25  0.4  0.6  0.7  1.0];
```

disini, 0.25 (≈2KHz) akan terjadi cut-off, agar lebih akurat anda dapat mencoba-coba nilai M diatas nilai 0.25 dan melihat hasil respon frekuensinya

Program lengkap dalam Matlab:

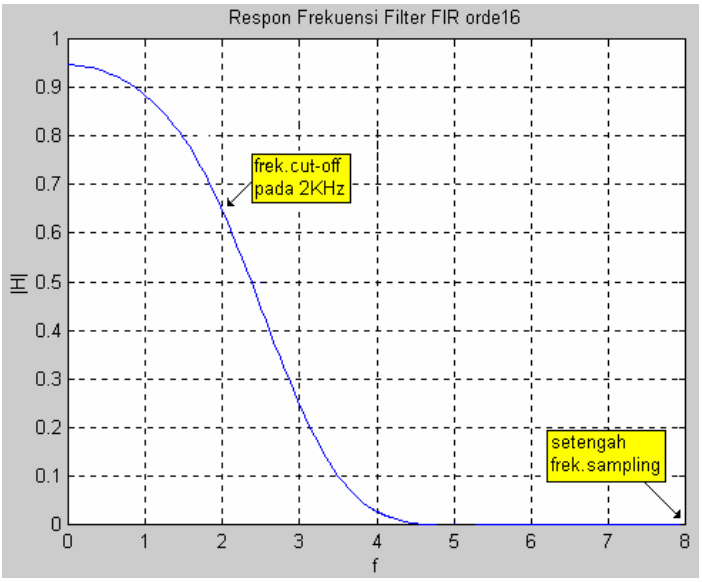
```
m=[1    0.8   0    0    0    0 ];      % vektor m
f=[0.0  0.25  0.4  0.6  0.7  1.0];     % vektor f
n=16; fs2=8;                           % orde 16, ½ frek.sampling=8KHz
b=fir2(n,f,m);                         % hitung koefesien filter

% tulis kedalam file fir.txt
fid = fopen('fir.txt','w');            % buka file
fprintf(fid,strcat('Koefesien filter FIR orde-',int2str(n),'\n'));
fprintf(fid,strcat('Generated on-',datestr(now),'\n\n'));
for i=1:n+1                            % simpan dalam file
   fprintf(fid,'%1.6f \n',b(i));
end
fclose(fid);                           % tutup file

% respon frekuensi
[H,w]=freqz(b,1,128);
% plot respon frekuensi
plot(w/pi*fs2,abs(H));

xlabel('f (KHz)');ylabel('|H|');grid;
title(strcat('Respon Frekuensi Filter FIR orde-',int2str(n)));
```

Nilai koefesien yang disimpan didalam file fir.txt dapat anda lihat pada command window Matlab dengan mengetikkan perintah `type fir.txt`



Gambar 3. Melihat isi dari file fir.txt yang telah dibuat

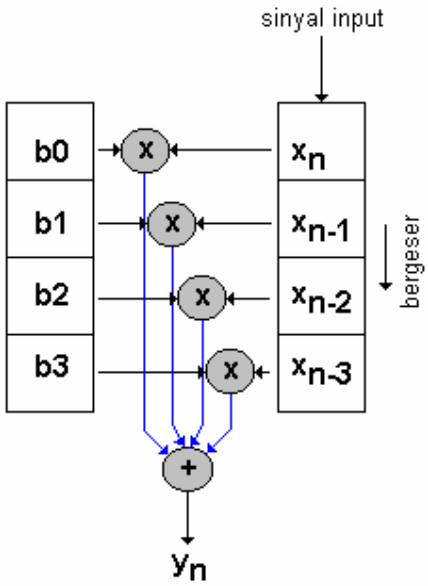
Respon frekuensi filter FIR orde 16 hasil disain tersebut ditunjukkan oleh gambar 4.



Gambar 4. Plot respon frekuensi filter low-pass FIR orde 16

Membuat program filter FIR pada DSP

Diagram flow filter FIR yang ditunjukkan oleh gambar 1, apabila diputar sedikit maka akan menjadi seperti gambar 5 berikut.



Setiap data yang masuk (x[n]) ditampung pada sebuah buffer, kemudian tiap elemen buffer dikalikan dengan koefesien dan dijumlahkan untuk menghasilkan sebuah output. Untuk data berikutnya, isi buffer digeser, lalu tiap elemen buffer dikalikan dengan koefesien dan dijumlahkan untuk menghasilkan sebuah output, begitu seterusnya.

Gambar 5. Ilustrasi langkah program filter FIR orde 3

Implementasi filter FIR pada DSP menggunakan bahasa-C

```
/*************************************************************************/
/* FIR.c
/* Modified from codec.c by Texas Instruments
/* Author: Hary Oktavianto. PENS-ITS. 30.11.04
/*
/*************************************************************************/

#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>

#define orde 16

void delay(s16 period);
int fir(int data);

/*************************************************************************/
/* Global Variables                                                      */
/*************************************************************************/

HANDLE hHandset;
s16 data;
float output;
int i, count=0;
int buf[orde+1]={0};
float coeff[orde+1]={
-0.002273,0.000166,0.009495,0.008448,-0.031839,-0.056286,0.060089,0.297753,
0.426758,0.297753,0.060089,-0.056286,-0.031839,0.008448,0.009495,0.000166,-0.002273};


/*************************************************************************/
/* MAIN  PROGRAM                                                         */
/*************************************************************************/

void main()
{
    s16 cnt=2;

    if (brd_init(100))
        return;

       /* blink the leds a couple times */
       while ( cnt-- )
       {
              brd_led_toggle(BRD_LED0);
              delay(1000);
              brd_led_toggle(BRD_LED1);
              delay(1000);
              brd_led_toggle(BRD_LED2);
              delay(1000);
       }

    /* Open Handset Codec */
    hHandset = codec_open(HANDSET_CODEC);     /* Acquire handle to codec */

    /* Set codec parameters */

    codec_dac_mode(hHandset, CODEC_DAC_15BIT);       /* DAC in 15-bit mode */
    codec_adc_mode(hHandset, CODEC_ADC_15BIT);       /* ADC in 15-bit mode */
    codec_ain_gain(hHandset, CODEC_AIN_0dB);         /* 0dB gain on analog input to ADC */
    codec_aout_gain(hHandset, CODEC_AOUT_MINUS_0dB); /* 0dB gain on analog output from DAC */
    codec_sample_rate(hHandset,SR_16000);            /* 16KHz sampling rate */

    /* Polling and digital loopback */
    while (1)
    {
       /* Wait for sample from handset */
       while (!MCBSP_RRDY(HANDSET_CODEC)) {};
```

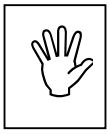
```
        /* Read sample from and write back to handset codec */
        data = *(volatile u16*) DRR1_ADDR(HANDSET_CODEC);
        *(volatile u16*)DXR1_ADDR(HANDSET_CODEC) = fir(data);
    }
}

int fir(int data)
{
        output = 0;

        // geser buffer
        for(i=orde+1;i>0;i--) {buf[i] = buf[i-1];}
        buf[0] = data;

        // hitung output
        for(i=0;i<orde;i++) {output += coeff[i]*buf[i];}

        return( (int) output );
}

void delay(s16 period)
{   int i, j;

    for(i=0; i<period; i++) {for(j=0; j<period>>1; j++);}
}
```

5. Peralatan

- 1 set PC yang dilengkapi dengan software Matlab dan Code Composer Studio
- 1 set DSK TMS320C5402
- 1 set function generator
- 1 set oscilloscope

6. Prosedur Praktikum

Praktikan diharapkan mengikuti langkah-langkah prosedur praktikum dan apabila ada kesulitan harap bertanya kepada asisten.



Praktikan harus didampingi asisten ketika menjalankan bagian ini !!!

Tujuan:
Mendisain filter **low-pass orde 16** dengan frekuensi **cut-off 3 KHz** pada frekuensi **sampling 16 KHz**.

Langkah-langkah:

1. Persiapan peralatan :
    a. PC dalam keadaan mati.
    b. Hubungkan DSK ke PC menggunakan kabel paralel port yang tersedia.

c. Hubungkan output adapator ke input power DSK.
d. Hubungkan kabel power adaptor, nyalakan adaptor.
e. Nyalakan PC
f. Jalankan aplikasi Code Composer Studio dan pastikan dapat terhubung dengan board DSK

2. Bukalah software Matlab, dan ketikkan program berikut pada matlab editor kemudian jalankan.

```
m=[1    1      0     0    0    0  ];   % vektor m
f=[0.0  0.375  0.48  0.6  0.7  1.0];   % vektor f
n=16;                                  % orde 16
fs2=8;                                 % ½ frek.sampling=8KHz
b=fir2(n,f,m);                         % hitung!

% tulis kedalam file fir.txt
fid = fopen('fir.txt','w');            % buka file
fprintf(fid,strcat('Koefesien filter FIR orde-',int2str(n),'\n'));
fprintf(fid,strcat('Generated on-',datestr(now),'\n\n'));
for i=1:n                              % simpan dalam file
   fprintf(fid,'%1.6f,\n',b(i));
end
fprintf(fid,'%1.6f\n',b(n+1));
fclose(fid);

[H,w]=freqz(b,1,128);                  % respon frekuensi
plot(w/pi*fs2,abs(H));                 % plot respon frekuensi

xlabel('f (KHz)');ylabel('|H|');grid;
title(strcat('Respon Frekuensi Filter FIR orde-',int2str(n)));
```

3. Amati dan catat hasil plot respon frekuensi filter pada Matlab, anda akan membandingkan hasil yang dihitung oleh Matlab dengan hasil percobaan.

4. Dengan menggunakan Windows Explorer, buatlah folder baru pada direktori **D:\prak_pengolahansinyal** dengan kelas Anda diikuti dengan subfolder nama Anda. Perhatikan penulisan folder yang Anda buat. Kemudian salinlah direktori fir pada **C:\ti\examples\dsk5402\dsp\fir** kedalam direktori **D:\prak_pengolahansinyal*kelas**nama***. Hal ini dimaksudkan untuk mempermudah mengembalikan isi project seperti dalam keadaan semula apabila terjadi kesalahan.

5. Pada Code Composer Studio, dengan menggunakan **Project → Open**, bukalah file project *fir.pjt* pada direktori **D:\prak_pengolahansinyal*kelas**nama*\fir**. Apabila file library pada project tersebut ada yang tidak ditemukan maka carilah library tersebut pada direktori c:\ti yang sesuai. Hal ini terjadi karena lokasi project berpindah tempat. File library yang digunakan ada tiga yaitu:
   a. rts.lib pada direktori c:\ti\c5400\cgtools\lib
   b. dsk5402.lib pada direktori c:\ti\c5400\dsk5402\lib
   c. drv5402.lib pada direktori c:\ti\c5400\dsk5402\lib

6. Bukalah file **fir.c** didalam source pada jendela project view. Isilah variabel global `float coeff[orde+1]` dengan nilai koefesien yang telah didapat menggunakan Matlab. Anda bisa melihatnya kembali pada file fir.txt yang dihasilkan.

7. Pilih **Project → Rebuild All** (atau dengan menekan ikon ). Maka CCS akan merekompilasi, mengassembler dan melakukan relink semua file pada project. Pesan pada proses ini akan ditampilkan pada bagian bawah window

8. Setelah kompilasi selesai dan tidak ada error, pilih **File → Load Program**, browse dan pilih file fir.out. Maka CCS akan meload program pada target DSP dan membuka window dis-assembly yang memperlihatkan instruksi program dalam bahasa assembler.

9. Pilih **Debug → Go Main**

10. Siapkan oscilloscope yang telah dikalibrasi. Atur volt/div pada 1 volt dan atur time/div pada 1ms.

11. Siapkan function generator untuk menghasilkan sinyal sinusoida dengan frekuensi 1KHz dan amplitudo 50mVpp.

Ingat ! Jangan memperbesar amplitudo function generator melebihi 50mV, tegangan input analog maksimum pada DSK terbatas

12. Hubungkan output function generator pada input board DSK (jack audio untuk microphone, bersebelahan dengan konektor RJ-11 untuk line telephone) dan hubungkan output board DSK (jack audio untuk speaker, bersebelahan dengan konektor DB9 untuk serial port) pada input oscilloscope.

13. Pilih **Debug → Run**, kemudian dengan amplitudo input dari function generator tetap sebesar 50mVpp dengan frekuensi 1KHz, ubahlah frekuensinya semakin besar perlahan-lahan. Amati dan catat hasil pengamatan seperti pada tabel 1.

14. Pilih **Debug → Halt**, untuk menghentikan eksekusi pada board.

15. Kerjakan tugas.

16. Tutuplah program CCS dengan memilih **File → Exit**.

17. Lepaskan jack audio dari board. Matikan oscilloscope dan adaptor power suplai.

18. Lakukan prosedur *shutdown* PC dengan benar. Rapikan kembali kabel dan peralatan.

Tabel 1. Hasil Pengukuran Filter Low-pass 3 KHz

No.	Vin = 50 mVpp	
	Frekuensi input (KHz)	Vout (mVpp)
1	1.0	
2	1.5	
3	2.0	
4	2.5	
5	3.0	
6	3.5	
7	4.0	
8	4.5	
9	5.0	
10	5.5	

7. Tugas

1. Disainlah filter **band-pass orde 16** dengan frekuensi cut-off **3KHz - 4KHz** pada frekuensi **sampling 16000 Hz**. Isilah hasil pengukuran pada tabel 2.

2. Ubahlah orde filter yang telah anda buat menjadi orde 32. Modifikasi kembali program pada Matlab dan program pada fir.c. Catatlah apa yang terjadi.

Tabel 2. Hasil Pengukuran Filter Band-pass 3KHz-4KHz

No.	Vin = 50 mVpp	
	Frekuensi input (KHz)	Vout (mVpp)
1	1.0	
2	1.5	
3	2.0	
4	2.5	
5	3.0	
6	3.5	
7	4.0	
8	4.5	
9	5.0	
10	5.5	
11	6.0	

8. Analisa

1. Pada tugas mendisain filter band-pass 3-4KHz, bandingkan plot grafik respon frekuensi filter pada Matlab dengan tabel 2 (sebelumnya gambarlah tabel hasil pengukuran pada kertas grafik), apakah telah sesuai ?
2. Filter band-pass 3-4KHz hanya melewatkan sinyal input pada range frekuensi 3KHz sampai 4KHz, mengapa pada 2,5Khz dan 4,5KHz masih ada sinyal input yang dilewatkan? Bagaimana cara mengatasinya?

3. Apa yang terjadi pada tugas nomor 2 ? Berilah kesimpulan mengenai definisi real-time pada kasus filter ini.

9. Pertanyaan pendahuluan

Buatlah program Matlab untuk mendisain filter band-pass seperti pada tugas (point 7, nomor 1).

10. Tambahan

Berikan saran atau komentar guna pengembangan lebih lanjut praktikum ini.