Real-Time FIR and IIR Filter Design Using MATLAB Interfaced with the TMS320C31 DSK

Walter J. Gomes III, Rulph Chassaing
University of Massachusetts Dartmouth

Abstract

This paper describes the design and real-time implementation of FIR and IIR filters using MATLAB interfaced directly with the TMS320C31 (C31) digital signal processor. An FIR or IIR filter can be readily designed using MATLAB functions to generate a set of coefficients associated with a desired filter's characteristics. These coefficients are included in a generic filter program transparent to the user. While the filter's frequency response is plotted on the PC monitor screen, it is being implemented in real-time with the C31 on board a $99 DSP Starter Kit (DSK). The authors have developed the support files required to duplicate these results. These files are available to anyone interested. Similar techniques can be developed to interface MATLAB with a different type of digital signal processor.

Introduction

Digital signal processors, such as the C31, are currently used for a wide range of applications from communications and controls to speech processing. They are found in cellular phones, fax/modems, disk drives, etc. They continue to be more and more successful because of the availability of low-cost support tools. DSP-based systems can be readily reprogrammed for a different application.

The C31-based DSK[1-3] includes Texas Instruments' C31 floating-point digital signal processor, and an Analog Interface Circuit (AIC) chip with A/D and D/A converters, input (anti-aliasing) and output (reconstruction) filters, all on a single chip. The package also includes an assembler, a debugger, and many applications examples. The C31 is a 32-bit processor with 2K (32-bit) words of internal memory (approximately 256 words of internal memory in the C31 on the DSK board are used for the communications kernel and vector). It has a 24-bit address bus to address 2^{24} or 16 million words for program, data, and I/O. Its instruction cycle time of 40 ns is capable of performing 50 million floating-point operations per second (MFLOPS).

The TLC320C40 AIC on the DSK board has 14-bit ADC and DAC. Although the AIC is rated for a maximum of 20 kHz in order to achieve maximum performance, sampling rates of 44.1 kHz for audio applications can be obtained. The AIC has two inputs, and connects to the serial port on the C31. Since all the C31 pins are available through four 32-pin connectors on the DSK board, external devices such as flash memory and alternative A/D and D/A converters can be connected to the C31 via these connectors[1,2].

```
% FIREX.M Main program that calls function dsk_fir()
frlen=1024; a=1; fs=16e3; Fc1=5e3; Fc2=5.5e3; order=48;
f=[0,Fc1/(fs/2),Fc2/(fs/2),1]; m=[1,1,0,0];
b=remez(order,f,m);
dsk_fir(b,fs);
h=freqz(b,a,frlen);
fpl=[0:fs/2/frlen:fs/2-fs/2/frlen];
plot(fpl,20*log10(abs(h)));
title('REMEZ Low Pass Filter, order=48, fs=16kHz, Fc=5kHz');
xlabel('Frequency'); ylabel('dB');
```

Figure 1. Main program for FIR filter design.

An assembler is included with the DSK package. In lieu of a linker, code is assembled at an absolute address using available assembler directives to control the starting addresses of different sections. A directive such as ".include" serves the function of a linker to include or chain several files together using only the assembler.

Various FIR and IIR filter design techniques can be obtained with MATLAB. The current student edition of MATLAB[8], version 5, includes the signal processing functions that can be used for different design characteristics. Using the techniques developed here, with one MATLAB command, the filter frequency response obtained is displayed on the PC monitor while the filter is implemented in real-time.

FIR Filter Design with MATLAB

Figure 1 shows the main program, FIREX.M, to run in MATLAB for the design of a FIR lowpass filter. The filter's characteristics are designated in this *"M"* file with specified magnitude at corresponding frequencies. It uses the function *remez()*, available with MATLAB, to design the filter with the resulting coefficients stored in the array *b*. Figure 2 shows a MATLAB plot of the magnitude of the frequency response displayed on the PC monitor. The value of *a=1* set in the program represents a transfer function that has a numerator polynomial only.

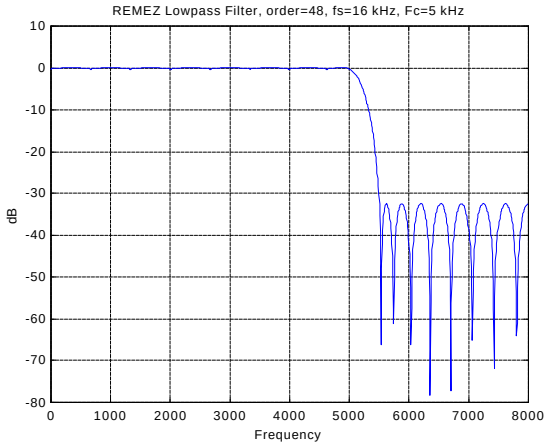

Figure 2. Frequency response of FIR lowpass filter plotted with MATLAB.

```
% DSK_FIR.M
function dsk_fir(bin,fsin)
MCLK=6.25e6; BW=(fsin/2)-250; SCF=(BW*288e3)/3600;
TA=round(MCLK/(2*SCF)); TB=round(MCLK/(2*TA*fsin));
A1=dec2hex(bitshift(TA,1)); A2=dec2hex(bitshift(TA,2));
B1=dec2hex(bitshift(TB,1)); B2=dec2hex(bitor(bitshift(TB,2),2));
coefflen=length(bin);  firorder=coefflen-1;
alignbuf=round(2 .^ ceil(log2(coefflen)));    % next power of 2
fid = fopen('firmat.cof','wt'); fprintf(fid,'\n       .data \n');
fprintf(fid,'COEFF   .float   '); j=0;
for i=coefflen:-1:2
  if j > 4
    j=0;   fprintf(fid,'\n       .float   ');
  end
  fprintf(fid,'%2.4E,',bin(i));
  j=j+1;
end
fprintf(fid,'\nH0     .float   %2.4E\n',bin(1));
fprintf(fid,'\nLENGTH  .set    %d\n',coefflen);
fprintf(fid,'AICSEC  .word  0%s%sh,1h,0%s%sh,63h \n',A1,A2,B1,B2);
fprintf(fid,'       .brstart "XN_BUFF",%d',alignbuf); fclose(fid);
dos ('dsk3a firmat'); dos ('dsk3load firmat BOOT');
```

Figure 3. Function that generates a coefficients file to be included in generic FIR filter program.

The main program also calls the function *dsk_fir()*, developed by the authors and shown in Figure 3, passing to it the filter's coefficients and the sampling frequency. This function generates a coefficient file, FIRMAT.COF, in a format appropriate to be included in the generic FIR filter program FIRMAT.ASM[1] listed in Appendix A. This function also calculates certain AIC register values in order to specify the sampling rate. It also selects one of the AIC's inputs and creates a buffer for the input samples, aligned on a 2^n-word boundary. Within MATLAB, the generic FIR filter program is then assembled and downloaded (using the assembler and loader included with the DSK package) into the DSK to run.

Within MATLAB, type FIREX to load and run FIREX.M. While the filter's frequency response is plotted and displayed on the PC monitor, the filter is being implemented in real-time with the C31. The frequency response of the lowpass filter obtained from an HP signal analyzer is displayed in Figure 4.

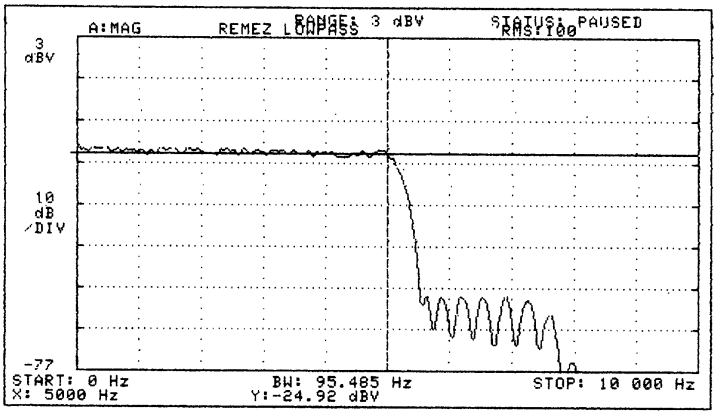

Figure 4. Real-time frequency response of lowpass FIR filter.

```
% IIREX.M Main program that calls function dsk_iir()
frlen=1024; fs=8e3; Fc1=1715; Fc2=1785; order=5; pripple=0.5; sripple=100;
[b,a]=ellip(order,pripple,sripple,[Fc1/(fs/2),Fc2/(fs/2)],'stop');
dsk_iir(b,a,fs);
h=freqz(b,a,frlen);
fpl=[0:fs/2/frlen:fs/2-fs/2/frlen];
plot(fpl,10*log10(abs(h)));
title('Elliptical Bandstop filter, order=10, fs=8kHz, Fc1=1715Hz, Fc2=1785Hz');
xlabel('Frequency'); ylabel('dB');
```

Figure 5. Main program for IIR filter design.

In lieu of a design with the MATLAB function *remez(),* other functions such as *fir1()* and *fir2()*, are also available with MATLAB for different design techniques and have been successfully tested.

IIR Filter Design with MATLAB

Figure 5 shows a listing of the main program, IIREX.M, for the design of a IIR filter using the MATLAB function *ellip()*. The filter's coefficients are contained within the *a* and *b* arrays. The filter is designed as a bandstop of order 10 with a notch frequency of 1750 Hz. MATLAB commands are used to plot the magnitude of the frequency response of the filter, and display it on the PC monitor. This program also calls the function *dsk_iir(),* developed by the authors and listed in Figure 6, to implement the filter on the DSK.

Dsk_iir() generates the coefficient file, IIRMAT.COF, in a format appropriate to be included in a generic C-coded IIR filter program IIRMAT.C[1] listed in Appendix B. It uses the following available MATLAB functions to format the IIR transfer function:

1. *tf2zp()* finds the zeros and poles of the transfer function expressed in terms of coefficients *a's* and *b's* associated with the denominator and numerator polynomials, respectively.

2. *zp2sos()* expresses the transfer function in terms of second-order sections used by the generic C-coded IIR filter program listed in Appendix B. For example, for a tenth-order IIR filter, five second-order sections or stages are cascaded, with one set of coefficients, *a's* and *b's,* for each stage.

From *dsk_iir(),* the C-coded IIR filter program is compiled, assembled, and linked using the TMS320 floating-point tools[4-7]. This creates a COFF executable file that is directly downloaded and run on the DSK. These floating-point tools are commercially available from TI and other vendors, and are not included with the $99 DSK package. Texas Instruments has an on-line DSP lab that can be accessed to compile, assemble, and link. It can be found at http://www.ti.com/sc/docs/dsps/dsplab.htm.

```
% DSK_IIR.M
function dsk_iir(bin,ain,fsin)
MCLK=6.25e6; BW=(fsin/2)-250; SCF=(BW*288e3)/3600;
%.... as in function dsk_fir.m to calculate A1,A2,B1,B2
[z,p,k]=tf2zp(bin,ain); sec_ord_sec=zp2sos(z,p,k);
stages=length(sec_ord_sec(:,1)); fid = fopen('iirmat.cof','wt');
fprintf(fid,'\n#define stages %d',stages);
fprintf(fid,'\nfloat a[stages][3]={/*numerator coefficients*/');
for i=1:stages
  fprintf(fid,'\n{');
  for m=1:3
    fprintf(fid,'%2.4E',sec_ord_sec(i,m));
    if (m == 3)
      if (i ~= stages)
        fprintf(fid,'},');
      end
    else
      fprintf(fid,', ');
    end
  end
  if i == stages
    fprintf(fid,'} };');
  end
end
fprintf(fid,'\nfloat b[stages][2]={/*denominator coefficients*/');
for i=1:stages
  fprintf(fid,'\n{');
  for m=5:6
    fprintf(fid,'%2.4E',sec_ord_sec(i,m));
    if (m == 6)
      if (i ~= stages)
        fprintf(fid,'},');
      end
    else
      fprintf(fid,', ');
    end
  end
  if i == stages
    fprintf(fid,'} };');
  end
end
fprintf(fid,'\nintAICSEC[4]={0x0%s%s,0x1,0x0%s%s,0x63};',A1,A2,B1,B2);
fclose(fid);
dos('cl30 -k iirmat -z iirmat.cmd'); dos('dsk3load iirmat boot');
```

Figure 6. Function that generates a coefficients file to be included in generic IIR filter program.

Within MATLAB, type IIREX to load and run IIREX.M. While the filter's frequency response is plotted and displayed on the PC monitor, the filter is being implemented in real-time with the C31. Figure 7 shows the frequency response of the filter obtained from a HP signal analyzer.

In lieu of an elliptical design with the MATLAB function *ellip()*, other designs have been successfully tested using other functions also available with MATLAB to obtain the coefficients:

a) *Chebyshev type I* and *type II* design with the functions *cheby1()* and *cheby2()*, respectively.

b) *Butterworth* design with the function *butter()*.

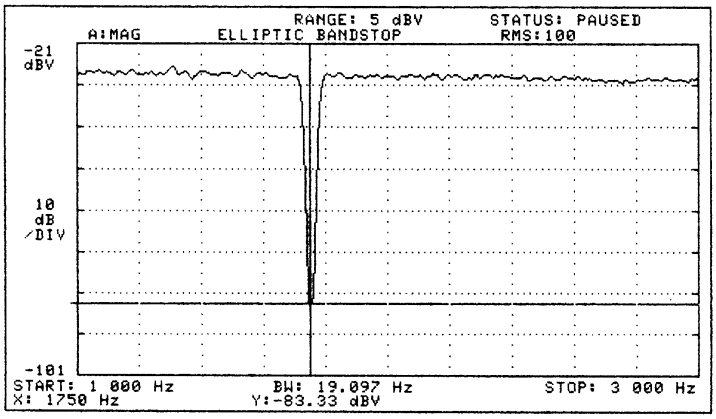

Figure 7. Real-time frequency response of bandstop IIR filter.

Appendix A

Figure A.1 is a listing of a generic FIR filter program, FIRMAT.ASM[1], written in C3x assembly code. The generated coefficient file, FIRMAT.COF, is included in this filter program. The file AICCOM31.ASM[1], also included in this filter program, contains I/O routines to communicate with DSK's codec.

```
;FIRMAT.ASM - Generic FIR program used by MATLAB function
            .start   ".text",0x809900          ;starting address of text
            .start   ".data",0x809C00          ;starting address of data
            .include "AICCOM31.ASM"            ;AIC communication routines
            .include "FIRMAT.COF"              ;coefficients file
            .data                              ;data section
XN          .sect    "XN_BUFF"                 ;buffer section for samples
            .loop    LENGTH                    ;loop LENGTH times
            .float   0                         ;init samples buffer to zero
            .endloop                           ;end of loop
XB_ADDR .word    XN+LENGTH-1                   ;last (bottom) sample address
HN_ADDR .word    COEFF                         ;starting addr of coefficients
            .text                              ;text section
            .entry   BEGIN                     ;start of code
BEGIN       LDP      AICSEC                    ;init to data page 128
            CALL     AICSET                    ;init AIC
            LDI      LENGTH,BK                 ;BK=size of circular buffer
            LDI      @XB_ADDR,AR1              ;AR1=last sample address
FILT        LDI      LENGTH-1,AR4              ;AR4=length-1 as loop counter
LOOP        CALL     AICIO_P                   ;AICIO routine,IN->R6 OUT->R7
            FLOAT    R6,R3                     ;input new sample ->R3
            STF      R3,*AR1++%                ;store newest sample
            LDI      @HN_ADDR,AR0              ;AR0 points to H(N-1)
            LDF      0,R0                      ;init R0
            LDF      0,R2                      ;init R2
            RPTS     LENGTH-1                  ;repeat LENGTH-1 times
            MPYF3 *AR0++,*AR1++%,R0            ;R0 = HN*XN
||          ADDF3    R0,R2,R2                  ;accumulation in R2
            DBNZD    AR4,LOOP                  ;delayed branch until AR4<0
            ADDF     R0,R2                     ;last accumulation
            FIX      R2,R7                     ;convert float R2 to integer R7
            NOP                                ;added due to delayed branch
            BR       FILT                      ;branch to filter routine
            .end                               ;end
```

Figure A.1. Generic FIR filter program.

Appendix B

Figure B.1 is a listing of a generic IIR filter program, IIRMAT.C[1], written in C. The generated coefficient file, IIRMAT.COF, is included in this filter program. The file AICCOMC.C[1], also included in this filter program, contains the C coded version of AICCOM31.ASM.

```
/*IIRMAT.C - Generic IIR program used by MATLAB function*/
#include "aiccomc.c"     /*include AIC comm routines*/
#include "iirmat.cof"    /*coefficients file        */
float dly[stages][2] = {0};   /*delay samples       */
int data_in, data_out;
float IIR(int *IO_in, int *IO_out, int n, int len)
{
 int i, loop = 0; float un, yn, input;
 while (loop < len)
 {
  asm("    IDLE   ");
  ++loop;
  input = *IO_in;
  for (i = 0; i < n; i++)
   {
   un = input - b[i][0] * dly[i][0] - b[i][1] * dly[i][1];
   yn = a[i][2]*dly[i][1] + a[i][1]*dly[i][0] + a[i][0]*un;
   dly[i][1] = dly[i][0];
   dly[i][0] = un; input = yn;
   }
    *IO_out = yn;
 }
}
void c_int05()
{
  PBASE[0x48] = data_out << 2;
  data_in = PBASE[0x4C] << 16 >> 18;
}
main()
{
  #define length 345
  int *IO_OUTPUT, *IO_INPUT;
  IO_INPUT = &data_in;
  IO_OUTPUT = &data_out;
  AICSET_I();
  for (;;)
   IIR((int *)IO_INPUT, (int *)IO_OUTPUT, stages, length);
}
```

Figure B.1. Generic IIR filter program.

Conclusion

MATLAB is a powerful tool for the design of both FIR and IIR filters, and can be interfaced directly with the $99 C31-based DSK to implement filters in real-time. All support files/programs are transparent to the user while the desired filter is being implemented in real-time. The TMS320 floating-point tools are not required to implement the FIR filter since the generic FIR filter program, written in

C3x assembly language, is assembled and run with the tools included with the DSK package. However they are required to implement the IIR filter since the generic IIR filter program is written in C and needs to be compiled, assembled, and linked before running on the DSK. Similar techniques can be developed to use an equivalent IIR filter program written in C3x assembly language, which would use the assembler available with the DSK package.

Acknowledgement

Three grants in 1996-98 from the National Science Foundation's (NSF) Undergraduate Faculty Enhancement (UFE) Program provided support to offer six workshops on DSP and Applications during the summers of 1996-98 for a total of 113 faculty. The direct interface between MATLAB and the DSK evolved during the 1998 workshop. The suggestions offered by Drs. T. Welch and C. Wright who attended that workshop are appreciated.

References

1. R. Chassaing, Digital Signal Processing-Laboratory Experiments Using C and the TMS320C31 DSK, J. Wiley, 1999.
2. B. Bitler, R. Chassaing, and P. Martin,"Digital Signal Processing with the TMS320C31 DSK," in Proceedings of the 1997 ASEE Annual Conference.
3. TMS320C3x DSP Starter Kit User's Guide, Texas Instruments Inc., 1996.
4. TMS320C3x User's Guide, Texas Instruments Inc., 1997.
5. TMS320C3x/C4x Optimizing C Compiler User's Guide, Texas Instruments Inc., 1997.
6. TMS320C3x/C4x Assembly Language Tools User's Guide, Texas Instruments Inc., 1997.
7. R. Chassaing, Digital Signal Processing with C and the TMS320C30, J. Wiley, 1992.
8. Student Version of MATLAB 5.0, The Mathworks, MA, 1998.
9. C. H. G. Wright, T. B. Welch, W. J. Gomes III, and Michael G. Morrow, "Teaching DSP Concepts Using MATLAB and the TMS320C31 DSK," in Proceedings of the 1999 ICASSP.

WALTER J. GOMES III is a Computer Engineer at the Naval Undersea Warfare Center, Newport, RI designing embedded subsystems for Autonomous Underwater Vehicles. He received a BS in Computer Engineering from the University of Massachusetts Dartmouth where he is currently completing a DSP research requirement for his MSEE. He is a member of IEEE and Eta Kappa Nu. Email: jgomes@ieee.org

RULPH CHASSAING received the PhD (EE) from the Polytechnic Institute of New York. He is the author of "*Digital Signal Processing-Laboratory Experiments Using C and the TMS320C31 DSK"* and *"Digital Signal Processing with C and the TMS320C30"*, and coauthored with Dr. D. W. Horning *"Digital Signal Processing with the TMS320C25"*, all published by Wiley (1999, 1992, 1990). Email: chassaing@email.msn.com