9 HARDWARE IMPLEMENTATION OF DIGITAL FILTERS

9.1 MICROPROCESSORS FOR DSP APPLICATIONS

Texas Instruments, in 1982, introduced the first microprocessor specifically designed for DSP applications – the TMS32010. Since then, several manufacturers have developed DSP microprocessors (also called DSP chips) of their own. The leading DSP chip manufacturers are (in alphabetical order):

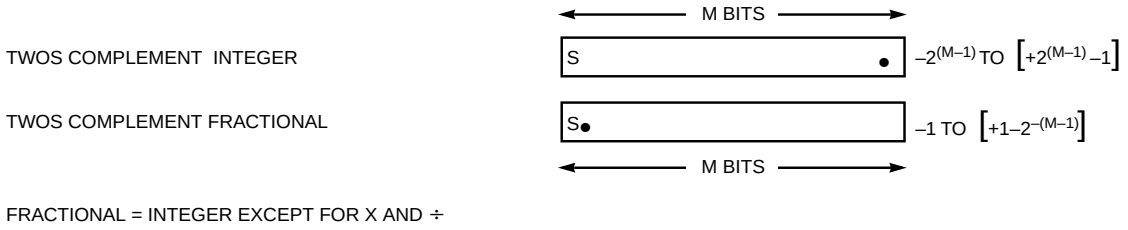
1. **Analog Devices**: ADSP-21xx family of 16-bit, fixed-point chips and the ADSP 21xxx family of 32-bit, floating-point chips (each x refers to a decimal digit in the designation of a member of the family).
2. **AT&T** (later **Lucent** and now **Agere**): the ADSP16xx family of 16-bit, fixed-point chips and ADSP32xx family of 32-bit, floating-point chips.
3. **Motorola**: the DSP56xxx family of 24-bit, fixed-point chips and the DSP96xxx family of 32-bit, floating-point chips.
4. **NEC**: the uPD77xxx family of 16-bit and 24-bit fixed-point chips.
5. **Texas Instruments**: the TMS320Cxx, TMS320Cxxx and TMS320Cxxxx families of 16-bit, fixed-point and 32-bit, floating-point chips.

Currently available DSP chips range from low-cost fixed-point DSP chips up to high-performance VLIW (Very Long Instruction Word) based fixed-point and floating-point DSPs combined with SIMD (Single Instruction Multiple Data) based functionality. Overview of available DSP chips [2] is given in the subject **DSP in Telecommunications** taught at the Department of Electronics and Multimedia Communications (KEMT).

Implementation of digital filters is significantly more complex for fixed-point DSPs, so we will test designed digital filters on fixed-point DSPs. At the KEMT there are available 24-bit Motorola DSP56002 and 16-bit Analog Devices ADSP2181, ADSP2191 and ADSP21535 - Blackfin. Both use **fractional arithmetic** for numbers represented as

$$x_{fract} = (-1)b_0 + \sum_{m=1}^{M-1} b_m 2^{-m} \tag{8.1}$$

where $b_0 \in \{0,1\}$, $M = 24$ for Motorola DSP56Kxx and $M = 16$ for Analog Devices ADSP21xx. Expression (8.1) represents two's complement fractional format and it is



compared with standard two's complement integer format on Figure.1.

Figure 1 Fractional and integer two's complement formats

This format represents main limitation to the possible range of input samples and digital filter coefficients. They must be from the interval $\langle -1,1)$. For direct form of FIR filter implementation this is generally no problem, but for cascade biquad implementation special scaling must be done. For biquad sections all coefficients must be[1] divided by 2 and output is scaled up by factor $2^s, s \geq 2$. Target hardware is programmed in C so conversion from fractional to integer must be done by the programmer. This conversion is processor specific.

24-bit Motorola DSP:

The conversion is done automatically for by macro *FRACT*2_INT shown in the Apendix1.

16-bit Analog Devices DSPs:

The programmer have to create simple ASCII file that contains properly ordered coefficients (for FIR and IIR filters) and scaling factors (for IIR filters). This can be done by using dedicated m-functions included in the Apendix II (they are commented in slovak).

Note that designed filters must be properly scaled during design procedure. Conversion to DSP-specific format is just simple format conversion.

Projects

Design digital filters (FIR or IIR) according to the specification given for your project. Perform scaling of filter coefficients to prevent overflows and include them into appropriate form suitable for DSP hardware (as described in the Appendix). Note that input samples are from the $\langle -1,1)$ interval, number of FIR filter coefficients must be $N_{FIR} \leq 256$ and number of IIR biquad sections must be $N_{BIQ} \leq 51$. These limitations

[1] This is a consequence of optimized implementation.

ensure real-time performance of target DSP hardware. Biquad sections use transposed direct form II realization and biquad coefficients are denoted as

$$H_i(z) = \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \tag{8.2}$$

For implementation on Motorola DSP use $M = 24$ and Analog Devices use $M = 16$.

Note that Matlab function filter provides incorrect results for high-order filters. The following Matlab function can compute h1 norm also for high-order filters expressed as cascade connection of biquad sections.

```
function h1 = bfilnorm(nsec,dsec);
% Synopsis: h1 = bfilnorm(nsec,dsec).
% Computes the h1 norm of a rational filter expressed as an abiquad cascade.
% Input parameters:
% nsec, dsec: the numerator and denominator polynomials of all biquads.
% Output parameters:
% h1: sum of absolute values of the impulse response

% M. Drutarovsky, Digital Filters, 2003
% KEMT FEI TU Kosice, Slovakia
% (Based on software package for the book:
%  A Course in Digital Signal Processing
%  by Boaz Porat, John Wiley & Sons, 1997)

% Original function filnorm provides wrong results for complex filters
% (e.g. more than 20 biquad sections expressed with B, A coefficients).
% This is (probably) a consequence of round-off errors in original
% MATLAB function FILTER.

[h,Z] = bfilter(nsec,dsec,[1,zeros(1,99)]);
h1 = sum(abs(h)); n = 100;  h1p = 0;
while((h1-h1p)/h1 > 0.00001),
   [h,Z] = bfilter(nsec,dsec,zeros(1,n),Z);
   h1p = h1; h1 = h1 + sum(abs(h)); n = 2*n;
end
```

```
function [y,zf] = bfilter(nsec,dsec,x,zi);
%BILTER         Digital filter based on biquad section.
% Y = BFILTER(NSEC, DSEC, X) filters the data in vector X with the
% biquad filter described by vectors NSEC and DSEC to create the filtered
% data Y.  The filter is a "cascade connection of biquad sections"
%
% [Y,Zf] = BFILTER(NSEC,DSEC,X,Zi) gives access to initial and final
% conditions, Zi and Zf, of the delays.
%
% BFILTER can replace standard FILTER function. FILTER function provides
% wrong results for complex filters (e.g. with more than 20 biquad sections)
% expressed with B,A coefficients. It is probably the effect of rounding
% in direct realization used by FILTER function.

% M. Drutarovsky, Digital Filters, 2003
% KEMT FEI TU Kosice, Slovakia
% (Based on software package for the book:
%  A Course in Digital Signal Processing
%  by Boaz Porat, John Wiley & Sons, 1997)

[M, junk] = size(nsec);                    % get number of biquad sections

if nargin == 3
 zi = zeros(M,2);                          % initialize zero state (if not given)
end

y = x;
for k=1:M                                  % cascade filtration
   [y,Zo]= filter(nsec(k,:),dsec(k,:),y,zi(k,:)');
   zf(k,:) = Zo';                          % output state
end
```

APPENDIX1 – FILE FORMAT FOR TARGET MOTOROLA DSP

**********Example of FIR.h file **********

```
/* this macro converts floating-point format to INTEGER format required by used C compiler for 24-bit DSP*/

#define _FRACT_2_INT(x)       ((x)<0 ? ((int) (x*0x800000-0.5)) : \
                              ((int) (x*0x800000+0.5)))


/************************************************************
;   Filter Type                 Bandpass
;   Filter Length               100
;   Sampling Frequency          48000.00
;
;   Band 1
;      Lower Band Edge          0.00000000
;      Upper Band Edge          4000.00000000
;      Nominal Gain             0.00000000
;      Nominal Ripple           0.00316228
;      Maximum Ripple           0.00016971
;      Maximum Ripple in dB     -75.40577864
;   Band 2
;      Lower Band Edge          7000.00000000
;      Upper Band Edge          12000.00000000
;      Nominal Gain             1.00000000                    decreased by 1/SCALE!!!
;      Nominal Ripple           0.00316228
;      Maximum Ripple           0.00017238
;      Maximum Ripple in dB     0.00149712
;   Band 3
;      Lower Band Edge          15000.00000000
;      Upper Band Edge          24000.00000000
;      Nominal Gain             0.00000000
;      Nominal Ripple           0.00316228
;      Maximum Ripple           0.00016910
;      Maximum Ripple in dB     -75.43709908
;
;   Quantization = 24 bits
;************************************************************/

#define N_COEFS       (100)                                 /* number of coefficients */
#define SCALE         (2.1)                                 /*  scaling factor*/

 fixpnt coefy[ N_COEFS ] =                                  /* coefficients */
 {
          _FRACT_2_INT(0.0000000000/SCALE),                 /* h(0) */
          _FRACT_2_INT(-0.0000010133/SCALE),                /* h(1) */
          _FRACT_2_INT(0.0000041127/SCALE),                 /* h(2) */
          _FRACT_2_INT(-0.0000092387/SCALE),                /* h(3) */
          …
          _FRACT_2_INT(-0.0000010133/SCALE),
          _FRACT_2_INT(0.0000000000/SCALE),                 /* h(99) */
};
```

**********Example of TIIR.H file**********

```
/* this macro converts floating-point format to INTEGER format required by used C compiler for 24-bit DSP */

#define _FRACT_2_INT(x)       ((x)<0 ? ((int) (x*0x800000-0.5)) : \
                              ((int) (x*0x800000+0.5)))
```

```
/*
;   Filter Type             Bandpass
;   Design type             ELLIPTIC IIR
;   Response type           bp
;
;
;   Filter Order                    12
;   Sampling Frequency              48000 Khz
;
;
;   Band 1
;      Lower Band Edge              0.00000000
;      Upper Band Edge              20
;      Nominal Gain                 0.00000000
;      Nominal Ripple               0.01
;      Maximum Ripple               0.0046479
;      Maximum Ripple in dB         -46.6549
;   Band 2
;      Lower Band Edge              300
;      Upper Band Edge              3400
;      Nominal Gain                 1.00000000  !!! was changed to 0.900000
;      Nominal Ripple               0.0351422
;      Maximum Ripple               0.0216407
;      Maximum Ripple in dB         0.185963
;   Band 3
;      Lower Band Edge              4000
;      Upper Band Edge              24000
;      Nominal Gain                 0.00000000
;      Nominal Ripple               0.01
;      Maximum Ripple               0.0046479
;      Maximum Ripple in dB         -46.6549
;
;   Quantization = 24 bits
*/

#define N_SECS          (6)                     /* number of biquad sections */
#define N_COEFS         (5*N_SECS+1)            /* number of sections (including output scaling coefficient)
                                                   !!! NOTE THAT MINIMUM OUTPUT SCALING MUST BE 2^2
*/

 fixpnt coefy[ N_COEFS ] =                      /* coefficients */
 {
        4,                                      /* output scaling 2^4 */
        _FRACT_2_INT(0.066985),                 /* (b01/2)      1st section */
        _FRACT_2_INT(-0.107882),                /* (b11/2)  */
        _FRACT_2_INT(0.865307),                 /* (a11/2)      NOTE THAT A11 is before B21!!!*/
        _FRACT_2_INT(0.066985),                 /* (b21/2) */
        _FRACT_2_INT(-0.438443),                /* (a21/2) */
        _FRACT_2_INT(0.250017),                 /* (b02/2)      2nd section */
        _FRACT_2_INT(-0.500000),                /* (b12/2) */
        _FRACT_2_INT(0.963265),                 /* (a12/2) */
        _FRACT_2_INT(0.250017),                 /* (b22/2) */
        _FRACT_2_INT(-0.465686),                /* (a22/2) */
        _FRACT_2_INT(0.250133),                 /* (b03/2)      3rd section */
        _FRACT_2_INT(-0.500000),                /* (b13/2) */
        _FRACT_2_INT(0.997799),                 /* (a13/2) */
        _FRACT_2_INT(0.250133),                 /* (b23/2) */
        _FRACT_2_INT(-0.498553),                /* (a23/2) */
        …
        _FRACT_2_INT(0.073036),                 /* (b06/2)      6th section */
        _FRACT_2_INT(-0.125866),                /* (b16/2) */
        _FRACT_2_INT(0.886290),                 /* (a16/2) */
        _FRACT_2_INT(0.073036),                 /* (b26/2) */
        _FRACT_2_INT(-0.484053),                /* (a26/2) */
};
```

APPENDIX2 – FILE FORMAT FOR TARGET ANALOG DEVICES DSP

```
function fir_dsp(h, B)
% Funkcia fir_dsp(h, B) vykona kvantovanie koeficientov FIR filtra,
% konverziu do formatu pre cielovy DSP a zapis do suboru coef.dat,
% ktore mozu byt priamo pouzite pri realizacii pomocou Analog Devices DSP
%
% Vstupne parametre:
%
%     h - koeficienty FIR filtra (kazdy riadok zodpoveda jednemu koeficientu)
%         pricom koeficienty musia byt v zlomkovom formate
%     B - pocet bitov pouzitych na reprezentaciu integer cisla
%
% Priklad:
% fir_dsp(h,16)

% Peter Popadic
% 14.04.2003

hq = quant(h, 'r', B);                  % kvantovane FIR koeficienty
conv_num (hq,B,'coef.dat');             % prevedie hq na integer a vysledok ulozi do suboru


function iir_dsp(nsec,dsec,B)
% Funkcia iir_dsp(nsec, dsec, B) vykona kvantovanie koeficientov jednotlivych
% bikvadov, konverziu do formatu pre cielovy DSP a zapis do suborov  scal.dat a
% coef.dat, ktore mozu byt priamo pouzite pri realizacii pomocou Analog Devices DSP
%
% Vstupne parametre:
%
%   nsec - citatele bikvad sekcii (kazdy riadok zodpoveda jednemu bikvadu)
%   dsec - menovatele bikvad sekcii (kazdy riadok zodpoveda jednemu bikvadu)
%        B - pocet bitov pouzitych na reprezentaciu integer cisla
%
% Priklad:
% iir_dsp(nsec,dsec,16)

% Peter Popadic
% 14.04.2003

nsec                                    % scaled sections
dsec

[M, junk] = size(nsec);
SCALE=max(abs(dsec')) >1.0;             % ak menovatel obsahuje koef > 1.0, skalovanie 2^1
                                        % inak SCAL = 0;

fi1e = fopen('scal.dat','w');           % skalovacie factory pre jednotlive bikvady
fprintf(fi1e,'%2i\n',SCALE);
fclose (fi1e);

for k=1:M,                              % vydeli nsecq a dsecq skalovacimi konstantami
   nsec(k,:)=nsec(k,:)*2^(-(SCALE(k)));
   dsec(k,:)=dsec(k,:)*2^(-(SCALE(k)));
end

nsecq = quant(nsec, 'r', B);            % quantized scaled sections
dsecq = quant(dsec, 'r', B);
coef=sort_num (nsecq, dsecq);           % preusporiada koeficienty do tvaru B2, B1, B0, A2, A1, ...
```

```
conv_num (coef,B,'coef.dat');                    % prevedie coef na integer a vysledok ulozi do suboru
```

```
function conv_num (x,B,outfile)
% Funkcia conv_num (x,B,outfile) vykona konverziu zlomkoveho (fractional)
% cisla na integer cislo s B bitovou presnostou a vysledok ulozi do suboru
%
% Vstupne parametre:
%
%       x - premenna (vektor), v ktorej su zlomkove cisla
%       B - pocet bitov pouzitych na reprezentaciu integer cisla
% outfile - nazov vystupneho suboru, do ktoreho sa ulozia integer cisla
%
% Priklad:
% conv_num (coef,16,'coef.dat')

% Peter Popadic
% 14.04.2003

x=x*2^(B-1);
x=round(x);
fi1e = fopen(outfile,'w');
fprintf(fi1e,'%i\n',x);
fclose (fi1e);
```

```
function biqsec=sort_num (nsec, dsec);
% Funkcia sort_num (nsec, dsec) usporiada koeficienty bikvad sekcii
% iir filtra pre potreby implementacie na DSP.
%
% Vstupne parametre:
%
%   nsec - citatele bikvad sekcii (kazdy riadok zodpoveda jednemu bikvadu)
%   dsec - menovatele bikvad sekcii (kazdy riadok zodpoveda jednemu bikvadu)
% biqsec - koeficienty bikvadov usporiadane pre DSP implementaciu
%
% Priklad:
% coef=sort_num (nsec, dsec)

% Peter Popadic
% 14.04.2003

[M, junk] = size(nsec);

for k=1:M,                                       % uvedene poradie vyzaduje DSP kod
        biqsec(5*(k-1)+1)=nsec(k,3);
        biqsec(5*(k-1)+2)=nsec(k,2);
        biqsec(5*(k-1)+3)=nsec(k,1);
        biqsec(5*(k-1)+4)=-dsec(k,3);
        biqsec(5*(k-1)+5)=-dsec(k,2);
end
```

LITERATURE

[1] Porat, B: *A Course in Digital Signal Processing*. John Wiley & Sons, Inc. New York 1997, ISBN 0-471-14961-7.

[2] Drutarovský, M.: Digital Signal Processors in Digital Signal Processing. Habilitation work, Košice, April 2000, pp.1-126, (in slovak).