Lecture 16

Filter implementation

Introduction

MiniDSP

The MiniDSP 2x4 [1] is "a low cost Digital Signal Processor for audio applications" based on the Analog Devices ADAU1701 chip (Fig. 1). We will use this as an example dedicated DSP system.

Fig. 1: MiniDSP 2x4 [1].

The MiniDSP allows the user to enter IIR filter coefficients in terms of cascaded "biquads" which we now turn to.

Filter represented as a cascade of biquads

The general rational transfer function

$$H(z)=\frac{b_0+b_1 w+b_2 w^2+\cdots b_M w^M}{1+a_1 w+a_2 w^2+\cdots+a_N w^N} \tag{1}$$

is the ratio of two polynomials in $w=z^{-1}$. A polynomial in w can always be factored, for example,

$$b_0+b_1 w+b_2 w^2+\cdots b_M w^M=b_M(w-\zeta_1)(w-\zeta_2)\cdots(w-\zeta_M) \tag{2}$$

Two factors of real roots can be combined into a single quadratic using

$$(w+a)(w+b)=w^2+(a+b)w+ab \tag{3}$$

while the product of two conjugate-root factors is the quadratic

$$(w+\sigma+j\omega)(w+\sigma-j\omega)=w^2+2\sigma w+(\sigma^2+\omega^2) \tag{4}$$

It follows that (1) can always be written as a product of *biquadratic factors* ("biquads")

$$H(z)=\frac{b_{10}+b_{11}w+b_{12}w^2}{1+a_{11}w+a_{12}w^2}\frac{b_{20}+b_{21}w+b_{22}w^2}{1+a_{21}w+a_{22}w^2}\cdots \tag{5}$$

which can be implemented as a cascade of 2nd order blocks (Fig. 2)

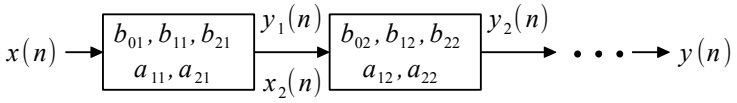

Fig. 2: Cascade of 2nd order systems.

The output of one block becomes the input to the next block. For the k^{th} block

$$y_k(n)=b_{0k}x_k(n)+b_{1k}x_k(n-1)+b_{2k}x_k(n-2)-a_{1k}y_k(n-1)-a_{2k}y_k(n-2) \tag{6}$$

In some implementations (including the MiniDSP) the minus signs associated with a_{1k}, a_{2k} are considered inconvenient. We will define

$$\begin{aligned}\bar{a}_{1k}&=-a_{1k}\\ \bar{a}_{2k}&=-a_{2k}\end{aligned} \tag{7}$$

so that (6) becomes

$$y_k(n)=b_{0k}x_k(n)+b_{1k}x_k(n-1)+b_{2k}x_k(n-2)+\bar{a}_{1k}y_k(n-1)+\bar{a}_{2k}y_k(n-2) \tag{8}$$

Lowpass biquads

The 2nd order Butterworth prototype is

$$H_p(s)=\frac{1}{1+cs+s^2} \tag{9}$$

with c a constant. This is transformed into a discrete lowpass filter by the transformation

$$H(z)=H_p\left(\alpha\frac{1-w}{1+w}\right) \tag{10}$$

with $w=z^{-1}$ and $\alpha=1/\tan(\pi f_c)$. The result is

$$\begin{aligned}H(z)&=\frac{1}{1+c\,\alpha\frac{1-w}{1+w}+\alpha^2\frac{(1-w)^2}{(1+w)^2}}\\ &=\frac{(1+w)^2}{(1+w)^2+c\,\alpha(1-w)(1+w)+\alpha^2(1-w)^2}\\ &=\frac{1+2w+w^2}{(1+c\alpha+\alpha^2)+2(1-\alpha^2)w+(1-c\alpha+\alpha^2)w^2}\end{aligned} \tag{11}$$

Typically we want to scale the filter coefficients so that $a_0=1$. This leads to

$$a_0=1,\ a_1=\frac{2(1-\alpha^2)}{(1+c\alpha+\alpha^2)},\ a_2=\frac{(1-c\alpha+\alpha^2)}{(1+c\alpha+\alpha^2)} \tag{12}$$

and

$$b_0 = b_2 = \frac{1}{1 + c\,\alpha + \alpha^2},\; b_1 = 2\,b_0 \tag{13}$$

For higher-order filters we perform these same operations for each quadratic factor. Here's an example.

Example 1: Design a lowpass filter with cutoff frequency $F_c = 1\,\text{kHz}$ based on a 4th order Butterworth prototype. Implement it as a product of biquads. The sampling frequency is $F_s = 48\,\text{kHz}$.

The discrete cutoff frequency is $f_c = 1/48 = 0.02083$ and

$$\alpha = \frac{1}{\tan(\pi f_c)} = 15.28$$

The prototype filter is

$$H_p(s) = \frac{1}{(1 + c_1 s + s^2)(1 + c_2 s + s^2)}$$

with $c_1 = 0.7654$, $c_2 = 1.848$. For the 1st biquad the coefficients are

$$b_0 = b_2 = \frac{1}{1 + c\,\alpha + \alpha^2} = 0.004062,\; b_1 = 2b_0 = 0.008124$$

$$a_0 = 1,\; a_1 = \frac{2(1 - \alpha^2)}{(1 + c\,\alpha + \alpha^2)} = -1.889,\; a_2 = \frac{(1 - c\,\alpha + \alpha^2)}{(1 + c\,\alpha + \alpha^2)} = 0.9050$$

For the 2nd biquad the coefficients are

$$b_0 = b_2 = \frac{1}{1 + c\,\alpha + \alpha^2} = 0.003806,\; b_1 = 2b_0 = 0.007613$$

$$a_0 = 1,\; a_1 = \frac{2(1 - \alpha^2)}{(1 + c\,\alpha + \alpha^2)} = -1.770,\; a_2 = \frac{(1 - c\,\alpha + \alpha^2)}{(1 + c\,\alpha + \alpha^2)} = 0.7850$$

The filter realization is

$$H(z) = \frac{0.004062 + 0.008124\,z^{-1} + 0.004062\,z^{-2}}{1 - 1.889\,z^{-1} + 0.9050\,z^{-2}} \times \frac{0.003806 + 0.007613\,z^{-1} + 0.003806\,z^{-2}}{1 - 1.770\,z^{-1} + 0.7850\,z^{-2}}$$

Exercise 1: Design a lowpass filter with cutoff frequency $F_c = 2\,\text{kHz}$ based on a 2nd order Butterworth prototype. Implement it as a biquad. The sampling frequency is $F_s = 8\,\text{kHz}$.

Answer: $$H(z) = \frac{0.2929 + 0.5858\,z^{-1} + 0.2929\,z^{-2}}{1 + 0.1716\,z^{-2}}$$

This procedure can be automated. Scilab code to do this appears in the Appendix. Copying and pasting the output into the MiniDSP user interface and calculating the frequency response produces the result shown in Fig. 3.

Fig. 3: MiniDSP plugin. Sixteenth-order Butterworth lowpass filter entered as cascade of eight biquads.

Hardware implementation

Let's consider the details of implementing a single biquad

$$y(n)=b_0 x(n)+b_1 x(n-1)+b_2 x(n-2)+\bar{a}_1 y(n-1)+\bar{a}_2 y(n-2) \tag{14}$$

To start let's consider the FIR version

$$y(n)=b_0 x(n)+b_1 x(n-1)+b_2 x(n-2) \tag{15}$$

A straight-forward implementation of this is shown in Fig. 4.

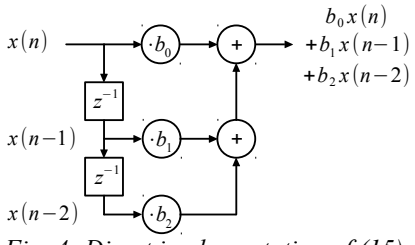

Fig. 4: Direct implementation of (15).

Here the z^{-1} blocks represent delays (memory registers), the $\cdot b$ blocks are multipliers and the + blocks are two-input adders. We can extend this idea to the IIR biquad (14) to arrive at the so-called *Direct Form I* implementation shown in Fig. 5.

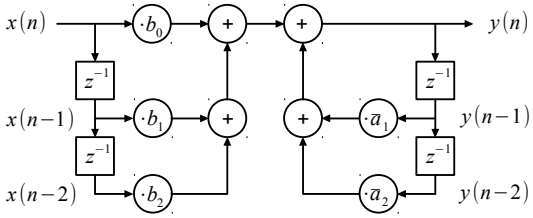

Fig. 5: Direct Form I implementation of (14).

This requires 4 delays, 5 multipliers and 4 adders. Now, consider the so-called *Direct Form II* implementation shown in Fig. 6.

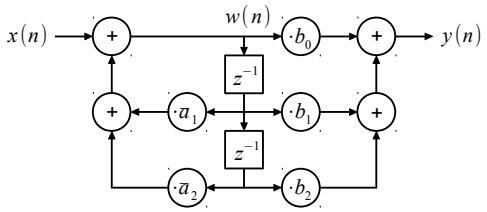

Fig. 6: Direct Form II implementation of (14).

This requires 2 fewer delays than the Direct Form I version. To verify that it results in the same input-output relation, define $w(n)$ as shown. Then

$$w(n)=x(n)+\bar{a}_1 w(n-1)+\bar{a}_2 w(n-2) \tag{16}$$

and

$$y(n)=b_0 w(n)+b_1 w(n-1)+b_2 w(n-2) \tag{17}$$

The z transform of (16) is (recall $\bar{a}_1=-a_1$ and $\bar{a}_2=-a_2$)

$$W(z)=\left(-a_1 z^{-1}-a_2 z^{-2}\right)W(z)+X(z) \tag{18}$$

so

$$W(z)=\frac{1}{1+a_1 z^{-1}+a_2 z^{-2}}X(z) \tag{19}$$

The z transform of (17) is

$$Y(z)=\left(b_0+b_1 z^{-1}+b_2 z^{-2}\right)W(z) \tag{20}$$

Combining (19) and (20) we have

$$Y(z)=\frac{b_0+b_1 z^{-1}+b_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}}X(z) \tag{21}$$

which is the desired biquad transfer function.

References

1. https://www.minidsp.com/products/minidsp-in-a-box/minidsp-2x4

Appendix – MiniDSP lowpass filter generator

```
clear;
////////////////////////////////////////////////////////////////////////
//miniDSPlowpassQuads.sce generates 8 sets of biquad coefficients
//in the format required by the MiniDSP "advanced" input window.
//Modify the following parameters as needed.
Fs = 48e3; //sampling frequency
fc = 1000/Fs;
N = 16; //order of prototype filter
////////////////////////////////////////////////////////////////////////

function Bq = ButterworthQuads(n)
  s = poly(0,'s');
  kMax = floor(n/2);
  for k=1:kMax
    a = sin((2*k-1)*%pi/(2*n));
    Bq(k) = s^2+2*a*s+1;
  end
  if (modulo(n,2)) //odd order case
    Bq(k+1) = s+1;
  end
endfunction

alpha = 1/tan(%pi*fc);
Bq = ButterworthQuads(N);
nQuads = length(Bq);
for k=1:nQuads
  cs = coeff(Bq(k));
  c = cs(2);
  a = [1+c*alpha+alpha^2,2*(1-alpha^2),1-c*alpha+alpha^2];
  b = [1,2,1];
  b = b/a(1);
  a = a/a(1);
  mprintf("biquad%1d,\n", k);
  mprintf("b0=%f,\n",b(1));
  mprintf("b1=%f,\n",b(2));
  mprintf("b2=%f,\n",b(3));
  mprintf("a1=%f,\n",-a(2));
  mprintf("a2=%f,\n",-a(3));
end
for k=nQuads+1:8 //MiniDSP always expects 8 sets of biquad coeffs
  mprintf("biquad%1d,\n", k); //add y(n) = x(n) biquads as needed
  mprintf("b0=%f,\n",1);
  mprintf("b1=%f,\n",0);
  mprintf("b2=%f,\n",0);
  mprintf("a1=%f,\n",0);
  mprintf("a2=%f,\n",0);
end
```