I.MX8X DSP SOF IIR FIR EQUALIZER BASIC THEORY

NOV,2022

JUN ZHU

I.MX CAS

Introduction

- i.MX8 series contains internal HiFi4 DSP. It is targeted for Audio related signal processing. SOF (Sound Open Firmware) is open source audio DSP firmware, driver and SDK. This document introduces basic theory about IIR/FIR digital filters, how to design IIR/FIR digital filters and the Equalizer filters implementation by SOF. After that, the document also describes how HiFi4 DSP MAC engine accelerate the EQ filters calculation.
- This document takes i.MX8 QXP C0 MEK Board + L5.10.72_2.0.0 as example,

  Note SOF can also be supported by i.MX8 QM and i.MX8 MP HiFi DSP.

i.MX8 SOF Equalizer Example Environment Set up

i.MX8 SOF Equalizer Example Environment Set up

- Follow SOF Documentation https://thesofproject.github.io/latest/getting_started/build-guide/build-from-scratch.html to build GCC toolchain and SOF Firmware from scratch.

- Follow NXP Community link:

  https://community.nxp.com/t5/iMX-and-Vybrid-Support/i-MX8X-DSP-Audio-Equalizer-Environment-Set-Up/ta-p/1564216

  https://community.nxp.com/t5/iMX-and-Vybrid-Support/Enable-Equalizer-in-i-MX8-SOF/ta-p/1310902

  to enable HiFi4 DSP SOF Equalizer Example for i.MX8X platform.

Discrete-Time FIR and IIR Filters Examples

Install Tools for Signal Analysis and Filter Design

- For SOF, Filters (including EQ filters) are designed by MATLAB language. But MATLAB is commercial product and need to by license to use it. GNU Octave is open source and free software, which is compatible with MATLAB language/scripts.

  For more detailed information, see Octave official website:

  https://octave.org/

- Install Octave under Ubuntu:

  sudo apt-get install octave

  sudo apt-get install octave-signal

  octave-signal contains MATLAB scripts for signal analysis and filters design. If the package is installed successfully, the scripts can be available in path /usr/share/octave/packages/signal-1.4.1/.

Discrete Time IIR System

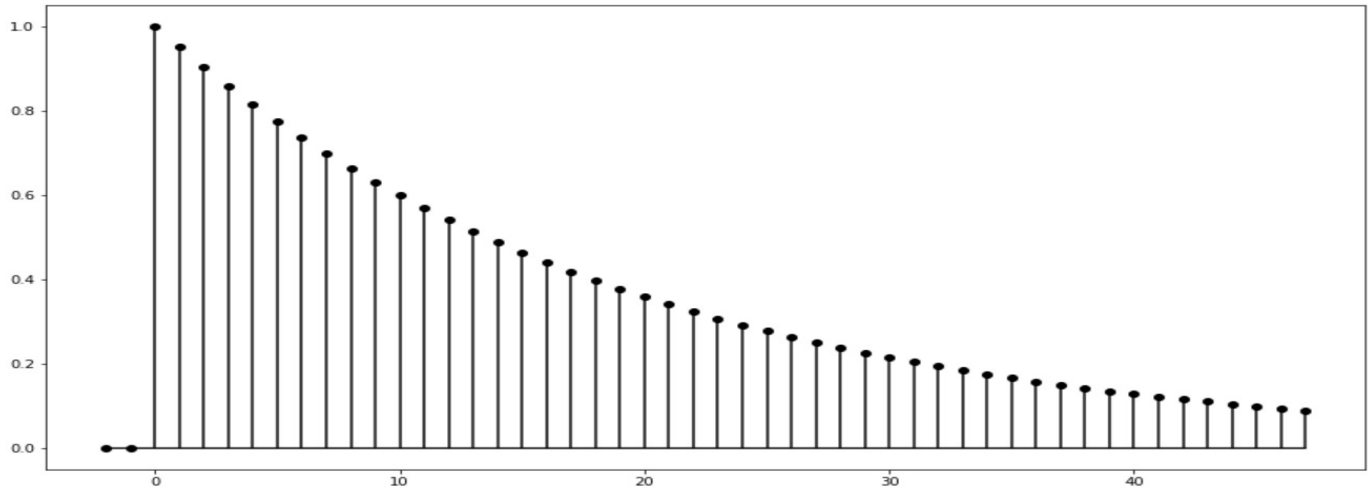
- Discrete Time IIR(Infinite Impulse Response) System

The unit impulse response of the system is infinite. For example, for system whose input/output conforms to following equation:

$$y[n] = ay[n-1] + x(n) \quad (a \text{ is coefficient here})$$

if the input signal is unit impulse, that is $x[n] = \delta[n]$, the unit impulse response is $h[n] = d^n$, which is infinite series. We can draw the response series if $a = 0.95$:

$$a = 0.95$$



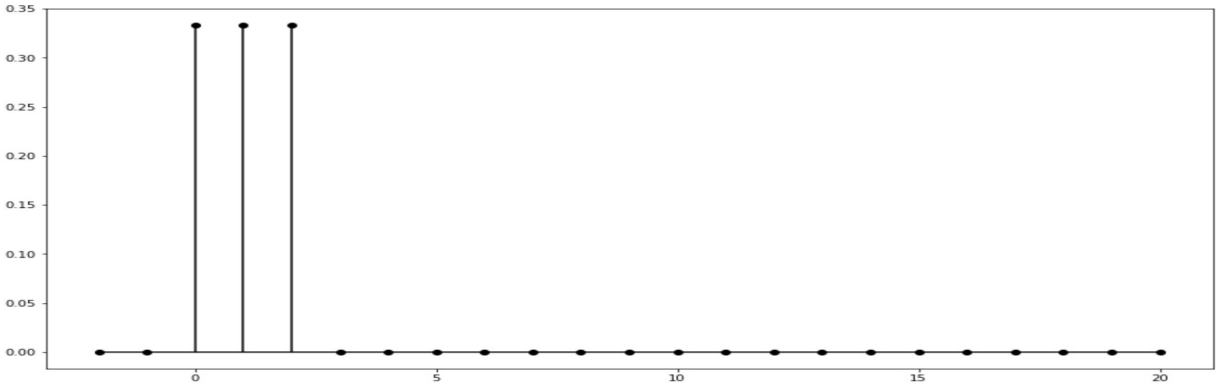
Discrete Time FIR System

- Digital FIR (Finite Impulse Response) System

The unit impulse response of the system is finite. For example, the three-point average system (low pass filter) defined by equation:

$$y[n]=\frac{1}{3}x[n]+\frac{1}{3}x[n-1]+\frac{1}{3}x[n-2]$$

if the input signal is unit impulse, that is $x[n]=\delta[n]$, the unit impulse response is $h[0]=\frac{1}{3}$, $h[1]=\frac{1}{3}$, $h[2]=\frac{1}{3}$, which is finite series. We can draw the response:



LTI Discrete Time System Transfer Function(1)

- Generally, LTI(Linear Time Invariant) discrete time system input/output can be given by Nth-order difference equation in Time Domain:

$$y[n] = -\sum_{k=1}^{N} a(k)\, y[n-k] + \sum_{r=0}^{M} b(r)\, x[n-r]$$

where $a[k]$ and $b[r]$ are constant coefficients. Apply z-transforms to both side of above equation, we finally obtain

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b(r)\, z^{-r}}{1 + \sum_{k=1}^{N} a(k)\, z^{-k}}$$

We call $H(z)$ as the **transfer function** of LTI discrete time system.

LTI Discrete Time System Transfer Function(2)

- For LTI(Linear Time Invariant) discrete time system, let $x[n]$ represents the system input, $y[n]$ represents the system output, $h[n]$ represents the system unit impulse response, then $y[n]$ is the convolution sum of $x[n]$ and $h[n]$:

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

Apply z-transform Convolution Property to above convolution sum equation, we obtain equation

$$Y(z)=X(z)H(z)$$

So LTI discrete time system transform function can also be defined as the z-transform of $h[n]$: $H(z)=\sum_{n=-\infty}^{\infty}h[n]z^{-n}$

LTI Discrete Time System Transfer Function(3)

- Summary for LTI discrete time system **transfer function**

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{r=0}^{M} b(r)z^{-r}}{1+\sum_{k=1}^{N} a(k)z^{-k}}=\sum_{n=-\infty}^{\infty} h[n]z^{-n}$$

- For transfer function, if $b(r)$, $r=0,\cdots,M$ and $a(k)$ $,k=1,\cdots,N$ are known values, then take $z=e^{j\omega}$ (DTFT transform), we can get the system AFR(Amplitude Frequency Response) and PFR(Phase Frequency Response)
- If AFR and PFR requirements is given, and then determine the transform function, it is filter design (IIR filter and FIR filter).

Example 1: FIR Low Pass Filter(1)

- Take three-point average system as example:

$$y[n]=\frac{1}{3}x[n]+\frac{1}{3}x[n-1]+\frac{1}{3}x[n-2]$$

Compared with transform function format, we can get

$$b[0]=\frac{1}{3}\quad b[1]=\frac{1}{3}\quad b[2]=\frac{1}{3}$$

And transform function

$$H(z)=\frac{1}{3}+\frac{1}{3}z^{-1}+\frac{1}{3}z^{-2}$$

Example 1: FIR Low Pass Filter(2)

- take $z = e^{j\omega}$, we obtain

$$H\left(e^{j\omega}\right) = \frac{1}{3} + \frac{1}{3}e^{-j\omega} + \frac{1}{3}e^{-2j\omega}$$

$$= \frac{1}{3}e^{-j\omega}\left(e^{j\omega} + e^{-j\omega} + 1\right)$$

$$= \frac{1}{3}e^{-j\omega}\left(e^{j\omega} + e^{-j\omega} + 1\right)$$

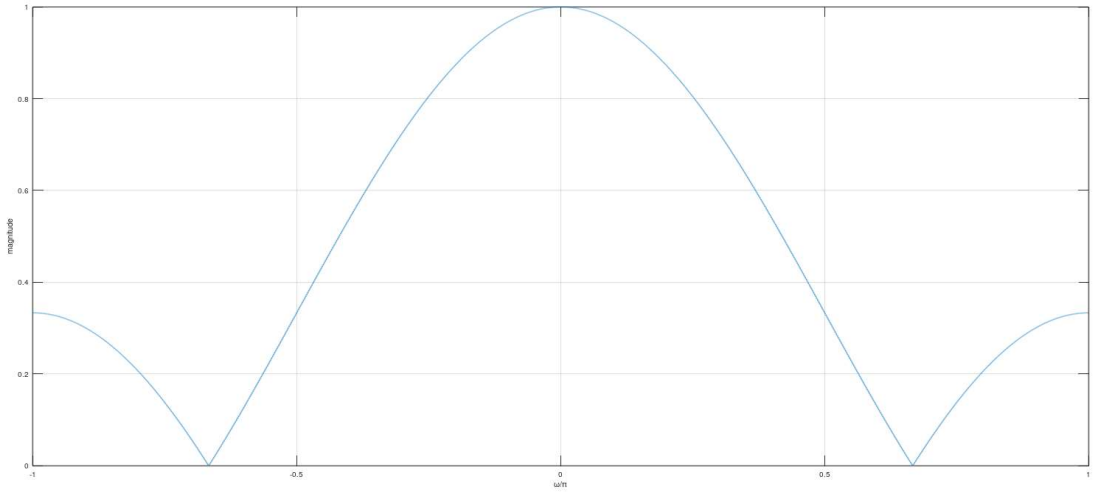
$$= \frac{1}{3}\left(2\cos\omega + 1\right)e^{-j\omega}$$

- We can get the magnitude of $H\left(e^{j\omega}\right)$ is $\left|\frac{1}{3}\left(2\cos\omega + 1\right)\right|$, phase of $H\left(e^{j\omega}\right)$ is $-\omega$

Example 1: FIR Low Pass Filter(3)

- Open Octave, and input following command to draw AFR diagram:

```
b=[1/3, 1/3, 1/3];
w=linspace(-pi,pi,1024);
[h,w]=freqz(b,1,w);
plot(w/pi,abs(h));grid
xlabel('\omega/\pi');ylabel('magnitude’);
```



From the AFR diagram, we can see that three-point average system is FIR low pass filter (Also known as Nonrecursive Discrete-Time Filter.)

Example 2: IIR Low Pass Filters (1)

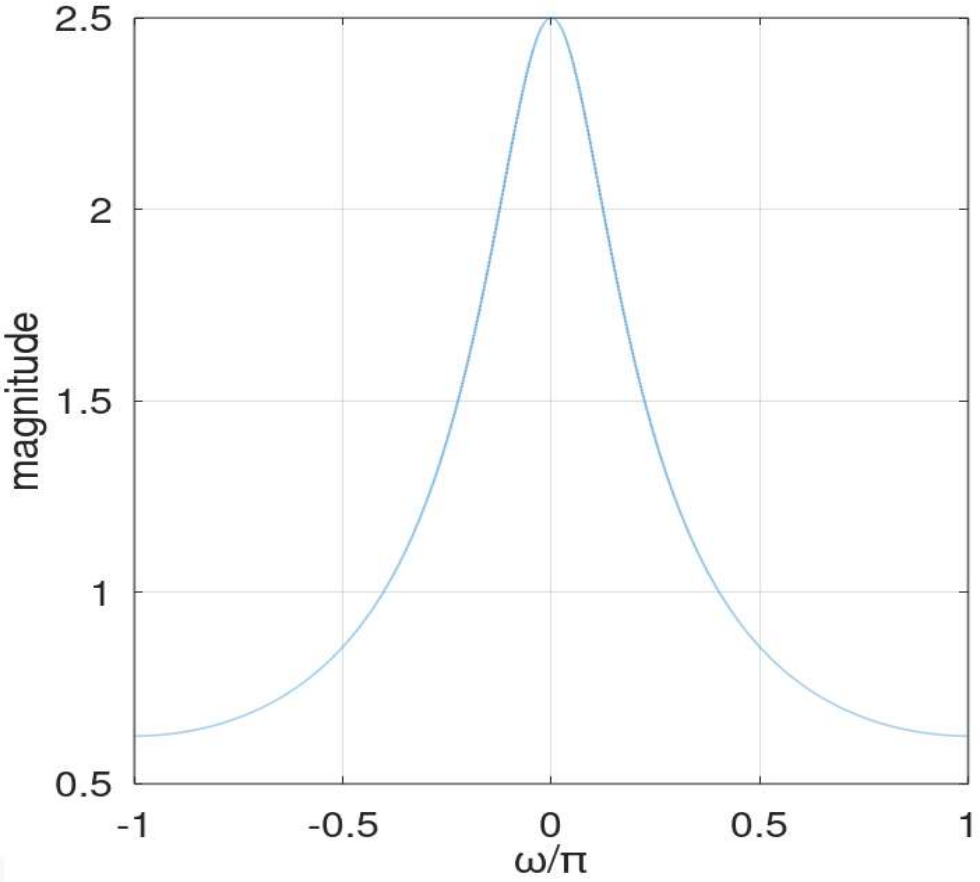
- Take system $y[n] = ay[n-1] + x(n)$ as example, $b[0] = 1$, $a[1] = -a$, and the transform function:

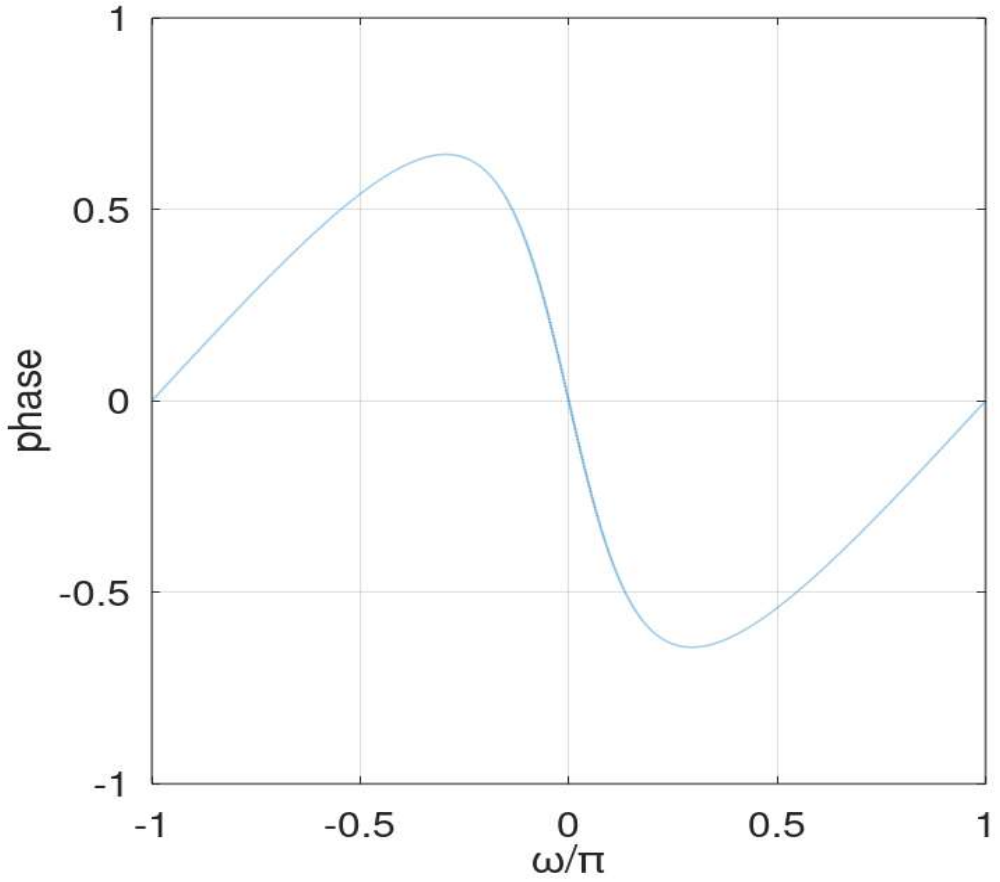
$$H(z) = \frac{1}{1 - az^{-1}}$$

- If $a = 0.6$, draw AFR and PFR by Octave:

```
b=1;
a=[1, -0.6]
w=linspace(-pi,pi,1024);
[h,w]=freqz(b,a,w);
subplot(1,2,1)
plot(w/pi,abs(h));grid
xlabel('\omega/\pi');ylabel('magnitude’);
subplot(1,2,2)
plot(w/pi,angle(h));grid
xlabel('\omega/\pi');ylabel(‘phase’);
```

Example 2: IIR Low Pass Filters(1)



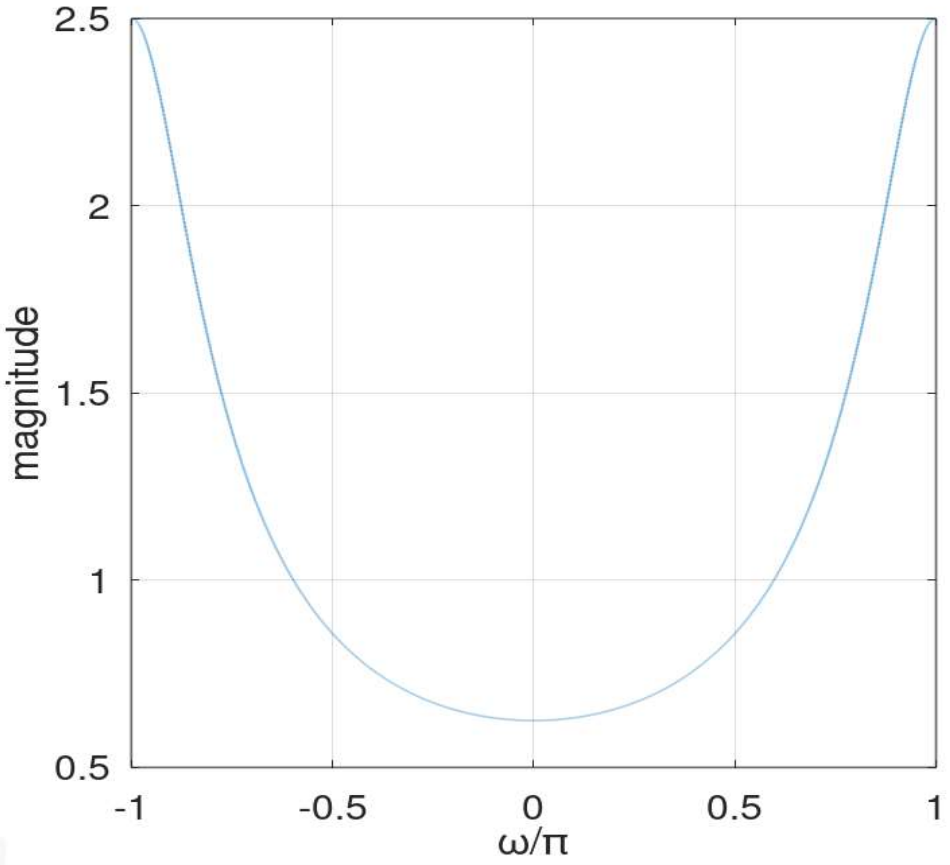


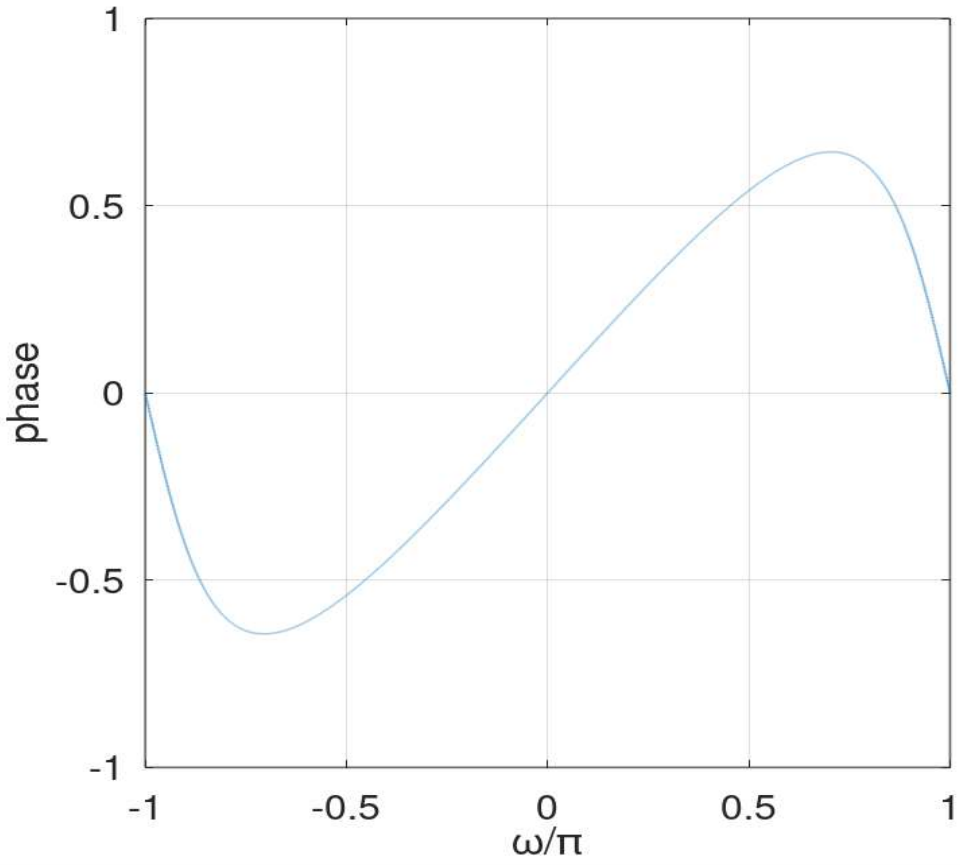
Example 2: IIR High Pass Filters (1)

- Take system $y[n] = ay[n-1] + x(n)$ as example, if $a = -0.6$, draw AFR and PFR by Octave:

```
b=1;
a=[1, 0.6]
w=linspace(-pi,pi,1024);
[h,w]=freqz(b,a,w);
subplot(1,2,1)
plot(w/pi,abs(h));grid
xlabel('\omega/\pi');ylabel('magnitude’);
subplot(1,2,2)
plot(w/pi,angle(h));grid
xlabel('\omega/\pi');ylabel(‘phase’);
```

Example 2: IIR High Pass Filters (2)





Discrete-Time FIR and IIR Filters Design Basics

Discrete-Time Filters Design Concept

- Recall LTI discrete time system **transform function:**

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{r=0}^{M}b(r)z^{-r}}{1+\sum_{k=1}^{N}a(k)z^{-k}}=\sum_{n=-\infty}^{\infty}h[n]z^{-n}$$

- For IIR filter design, take transform function format:

$$H(z)=\frac{\sum_{r=0}^{M}b(r)z^{-r}}{1+\sum_{k=1}^{N}a(k)z^{-k}}$$

According to IIR filter design requirements, to find suitable $b(r)$ and $a(k)$ (coefficients) values.

- For FIR filter design, take transform function format: $H(z)=\sum_{n=-\infty}^{\infty}h[n]z^{-n}$

According to FIR filter design requirements, to find suitable finite series $h[n]$.

IIR Filters Design Basics

- Digital IIR Filters are designed based on Analog Filters. Typical Analog Filters adopted are Butterworth Filter and Chebyshev Filter.

- Analog Filter system has transfer function as following:

$$H(s) = \frac{d_0 + d_1 s + \cdots d_{N-1} s^{N-1} + d_N s^N}{c_0 + c_1 s + \cdots c_{N-1} s_{N-1} + c_N s^N}$$

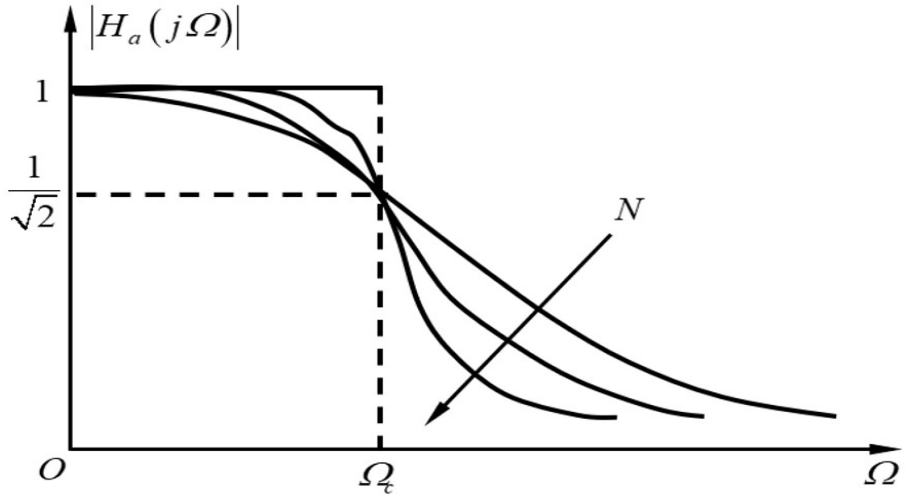
  Analog Filter design is to determine the coefficients $c_0, c_1, \cdots, c_{N-1}, c_N$ and $d_0, d_1, \cdots, d_{N-1}, d_N$ according to Analog Filter requirements.

- Once Analog Filter transfer function $H(s)$ is determined, we can then apply **Bilinear transform** to map s-domain to z-domain and thus get the Digital IIR Filter transfer function.

Butterworth Filter and Bilinear Transform

- SOF takes Butterworth Filter as IIR Filters design example. Butterworth Filter has following AFR equation and diagram:

$$\left|H_a(j\Omega)\right|^2 = \left(\frac{1}{\sqrt{1+\left(\frac{\Omega}{\Omega_c}\right)^{2N}}}\right)^2$$



- Bilinear transform:

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

Bilinear transform maps s-domain to z-domain and thus get the Digital Filters transform function.

Butterworth Digital Filter Design with MATLAB Function

- MATBLAB contains pre-defined function to design Butterworth Digital Filter:

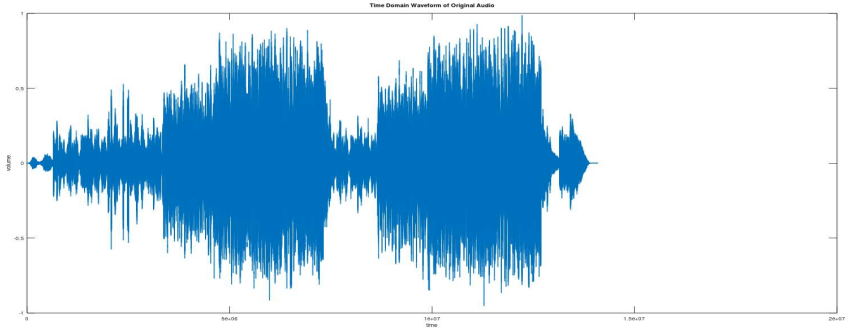
```
[b, a] = butter (n, Wc)          // low pass filter, by default
[b, a] = butter (n, Wc, 'high')  // high pass filter
[b, a] = butter (n, [Wl, Wh])    // band pass filter
[z, p, g] = butter (n, Wc, 'high'); // high pass filter, return value
                                    // format is zero-pole-gain
```

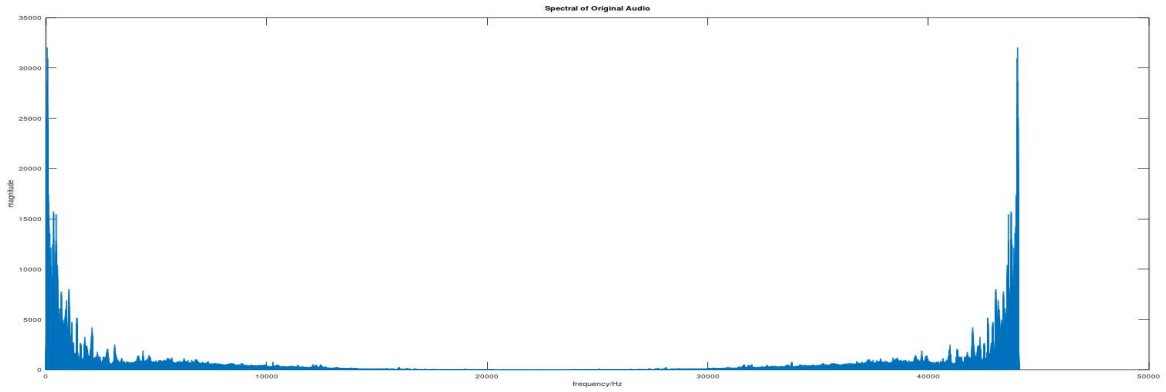
- Note the parameter Wc, it means “cutoff frequency” and must be specified in radians. If sample rate is f_s, Wc value should be $f/(f_s/2)$.

Butterworth Digital Filter Example (1)

- Human hearing mechanism can recognize voice signal ranges from 20HZ to 20kHZ. Following is MATLAB script which displays the spectral of one music audio:

```
pkg load signal;
[audio_data, Fs]=audioread('test.wav');
left=audio_data(:,1);            % left channel audio data
n_left=length(left);
y_left=fft(left,n_left);
f_left=Fs*(0:n_left-1)/n_left;
figure(1);
plot(0:n_left-1, left);
xlabel('time');ylabel('volume');
title('Time Domain Waveform of Original Audio');
figure(2)
plot(f_left, abs(y_left));
xlabel('frequency/Hz');ylabel('magnitude’);
title('Spectral of Original Audio');
% sound(left, Fs);
```

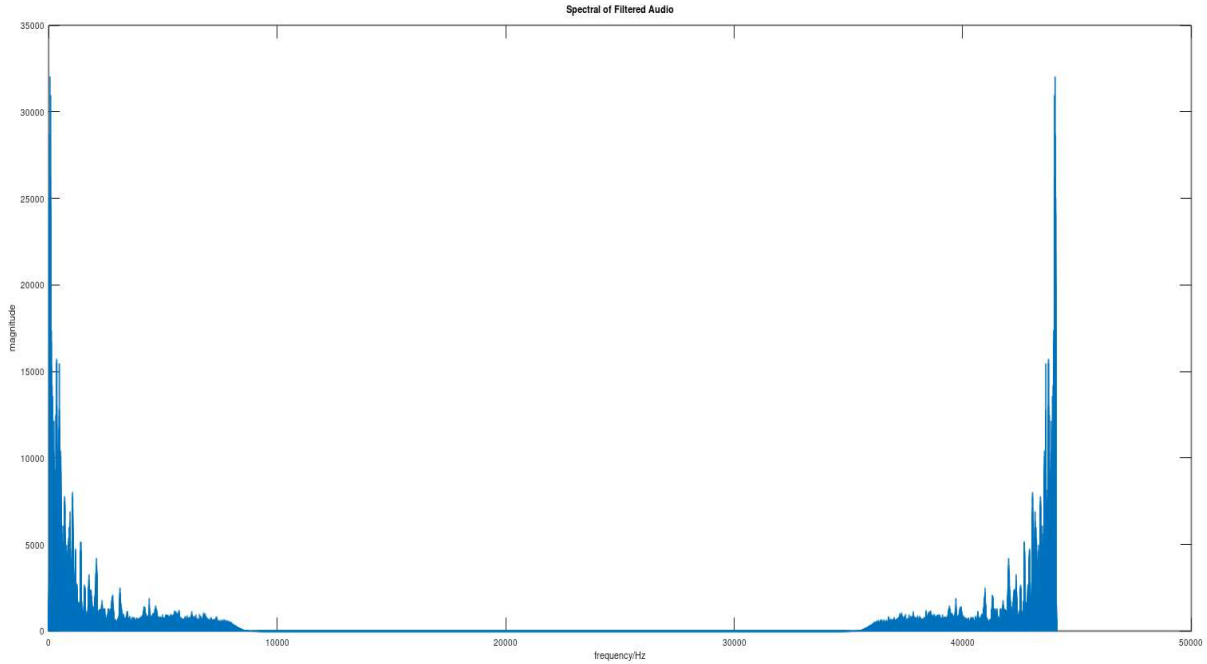




Butterworth Digital Filter Example (2)

- Here is one example for LP Butterworth digital filter which pass audio frequencies below 8 kHZ, and then we apply the filter to audio data above:

```
pkg load signal;
Fs = 44100; wp = 8000/(Fs/2); ws =9000/(Fs/2); rp = 1; rs = 40;
[N, wn] = buttord(wp, ws, rp, rs);
[b, a] = butter(N, wn);
[audio_data, Fs]=audioread('test.wav');
left=audio_data(:,1);
left_f=filter(b,a,left);
n_left_f=length(left_f);
y_left_f=fft(left_f,n_left_f);
f_left_f=Fs*(0:n_left_f-1)/n_left_f;
plot(f_left_f, abs(y_left_f));
xlabel('frequency/Hz');ylabel('magnitude');
title('Spectral of Filtered Audio');
% sound(left_f, Fs);
```



FIR Filters Design Basics

- As we mentioned earlier, FIR Filter takes transform function $H(z)=\sum_{n=-\infty}^{\infty}h[n]z^{-n}$, the filter design is to find corresponding $h\left[n\right]$ which meets the filter design requirements.
- One popular way for FIR Filters Design is Window Design Method. Here are typical steps for Window Design Method:
  - Given the expected $H_d\left(e^{j\omega}\right)$, take Inverse DTFT to get the time domain series $h_d\left(n\right)$. Sometimes, it's not easy to calculate for Inverse DTFT. So in practical, we need sample $H_d\left(e^{j\omega}\right)$, and then take IDFT to get $h_d\left(n\right)$
  - Take Window Function $w\left(n\right)$ to $h_d\left(n\right)$, and then get $h\left[n\right]$:

$$h\left[n\right]=h_d\left(n\right)w\left(n\right)$$

SOF takes Kaiser window Function, examples will be shown in next section.

Audio EQ Basic Theory and Examples

Audio Equalizer Basics

- Audio Equalizer can be considered as interconnection of several digital filters. According to the ways how digital filters are interconnected, there are two kinds of Audio Equalizer:
  - Graphic EQ: Parallel interconnection of digital filters.
  - Parametric EQ: Cascade interconnection of digital filters.

- According to digital filter types, EQ has following effects:

  lowpass

  highpass

  allpass/bandpass

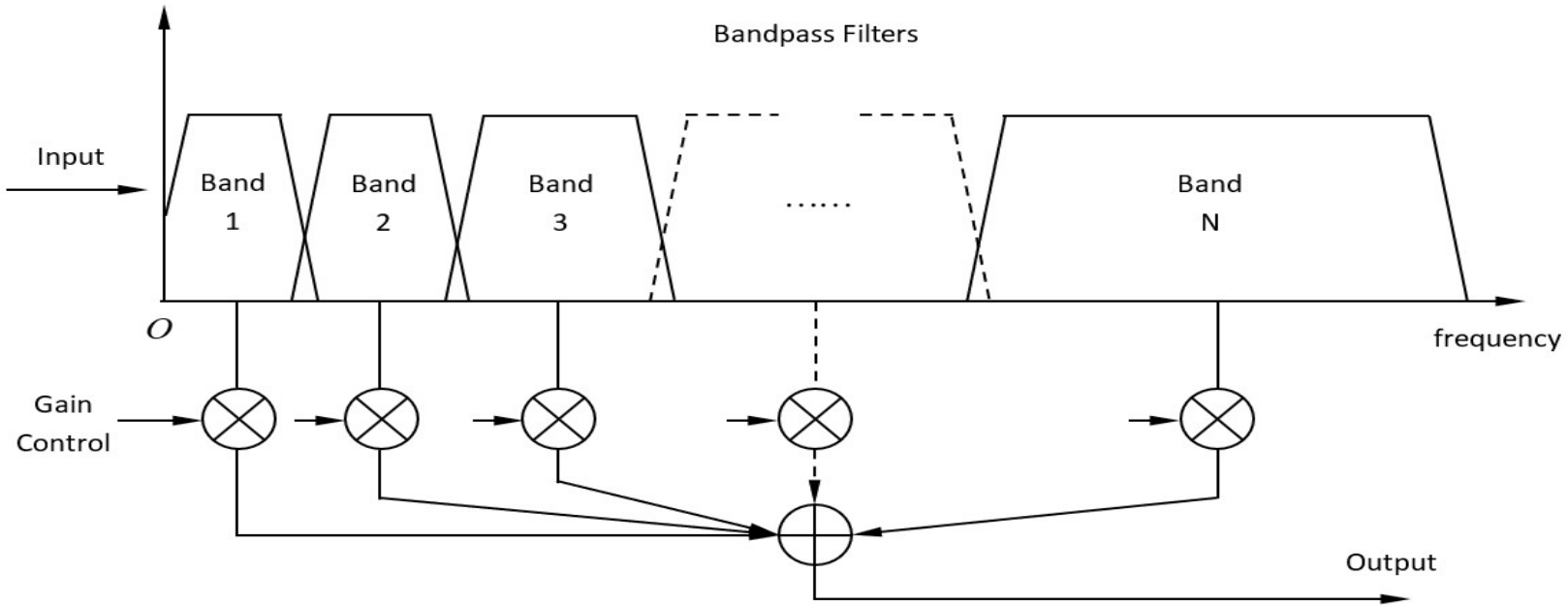
  low shelf/high shelf

  peak filter

  ……

Graphic EQ (1)

- For Graphic EQ, audio frequencies 20 Hz ~ 20k Hz are divided into multiple bands (for example, 10, 30 or 31 bands), then digital filters(gains) are applied to each band. The overall EQ effect can be considered as the sum of each band filter effect



Graphic EQ (2)

- Following is an example about bands division (10 bands).

Band #	Lowest/Hz	Center/Hz	Highest/Hz
1	22	31.25	44
2	44	62.5	88
3	88	125	177
4	177	250	355
5	355	500	710

Band #	Lowest/Hz	Center/Hz	Highest/Hz
6	710	1000	1420
7	1420	2000	2840
8	2840	4000	5680
9	5680	8000	11360
10	11360	16000	22720

- https://ww2.mathworks.cn/help/audio/ug/graphic-equalization.html

Parametric EQ

- Recall LTI system transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b(r) z^{-r}}{1 + \sum_{k=1}^{N} a(k) z^{-k}}$$

$H(z)$ can be written as multiplication of 2-order sub systems:

$$H(z) = \prod H_i(z)$$

Where $H_i(z)$ has following format:

$$H_i(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The format above is called biquad transfer function, it contains five coefficients: b_0, b_1, b_2, a_0, a_1, SOF data structure to store the biquad coefficients is defined in sof/src/include/user/eq.h, structure name is “sof_eq_iir_biquad_df2t”.

SOF IIR EQ Example – BassBoost (1)

- BassBoost is one example for IIR EQ and Parametric EQ. It is designed by MATLAB script and the script will calculate the filters biquad coefficients. The script will also store the biquad coefficients into text file named “eq_iir_bassboost.txt”.
- Recall the command “sof-ctl Dhw:0 -n 44 -s eq_iir_bassboost.txt &” to play music with IIR EQ bassboot effect, the DSP IIR EQ audio app will read the filters biquad coefficients from text file “eq_iir_bassboost.txt ”, and then do the filters calculation by the DSP MAC engine. For next section “HiFi4 DSP Introduction”, will provide more detailed description about the DSP filters calculation.
- IIR EQ MATLAB script example is available in SOF source code path sof/tools/tune/eq/example_iir_eq.m, the BassBoost filter is implemented as function bassboost_iir_eq().

SOF IIR EQ Example – BassBoost (2)

- Following is key point for function bassboost_iir_eq():

```
% Design
eq.peq = [ ...
                    eq.PEQ_HP2 30 NaN NaN ; ...
                    eq.PEQ_LS2 200 +10 NaN ; ...
         ];
eq = eq_compute(eq);
eq_plot(eq);
```

- From above code segment, we can see that BassBoot EQ actually contains two filters: 2 order HighPass filter and 2 order Low Shelf filter. The HighPass cutoff frequency is 30Hz, the Low Shelf cutoff frequency is 200Hz and with “+10dB” gain for frequencies below 200Hz.
- eq_compute() function will calculate the filters biquad coefficients, eq_plot() function will draw the diagram of the BassBoost EQ effect (the diagram will be shown later).

SOF IIR EQ Example – BassBoost (3)

- eq_compute() function will finally call eq_define_parametric_eq() function in sof/tools/tune/eq/eq_define_parametric_eq.m:

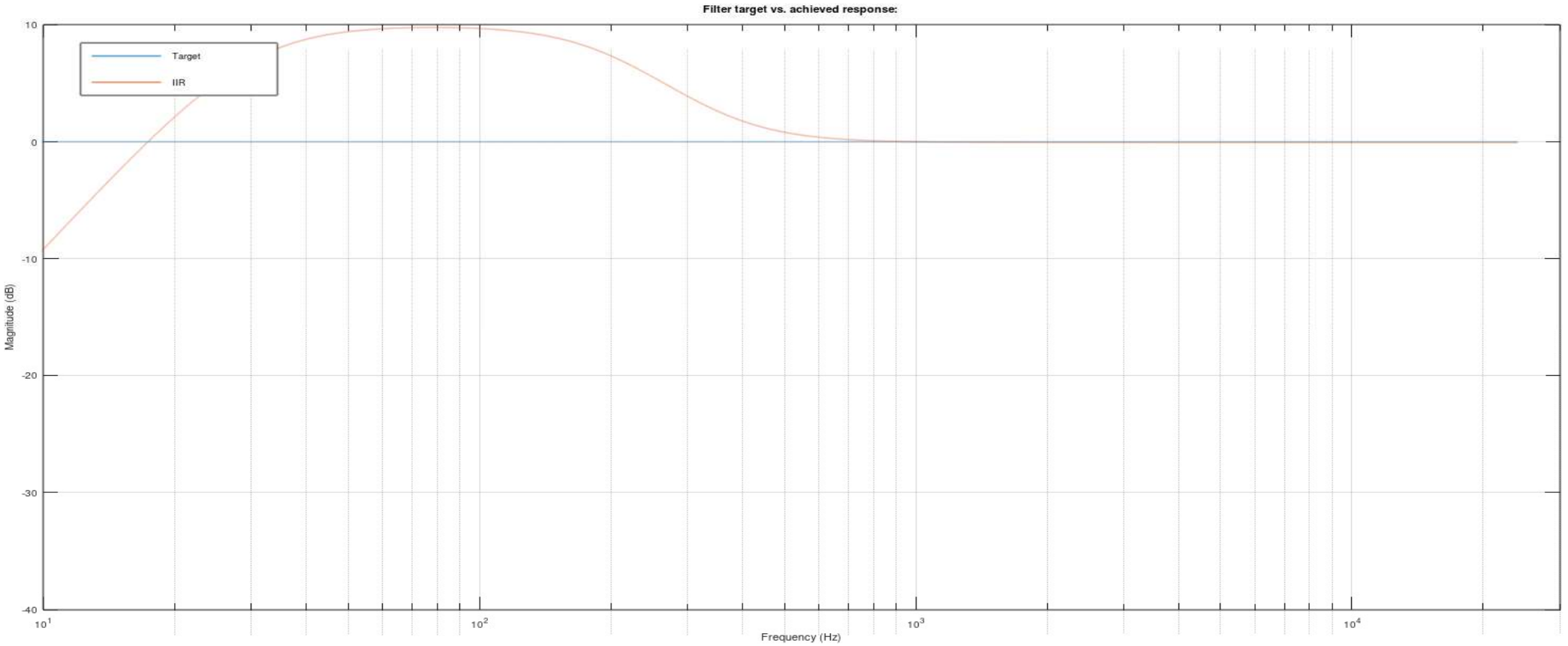
```
switch peq(i,1)
        case PEQ_HP1, [z0, p0, k0] = butter(1, 2*f/fs, 'high');
        case PEQ_HP2, [z0, p0, k0] = butter(2, 2*f/fs, 'high');
        case PEQ_HP4, [z0, p0, k0] = butter(4, 2*f/fs, 'high');
```

We can see the HighPass filter is actually Butterworth filter.

- The Low Shelf Filter is implemented as function low_shelf_2nd() in sof/tools/tune/eq/eq_define_parametric_eq.m. It will add 10dB gain to frequcies below 200 Hz.

- BassBoost filters effect can be shown by eq_plot() function, see next page for the diagram (Note for OCTAVE, should run command “pkg load signal” before running the script).

SOF IIR EQ Example – BassBoost (4)



SOF IIR EQ Example – BassBoost (5)

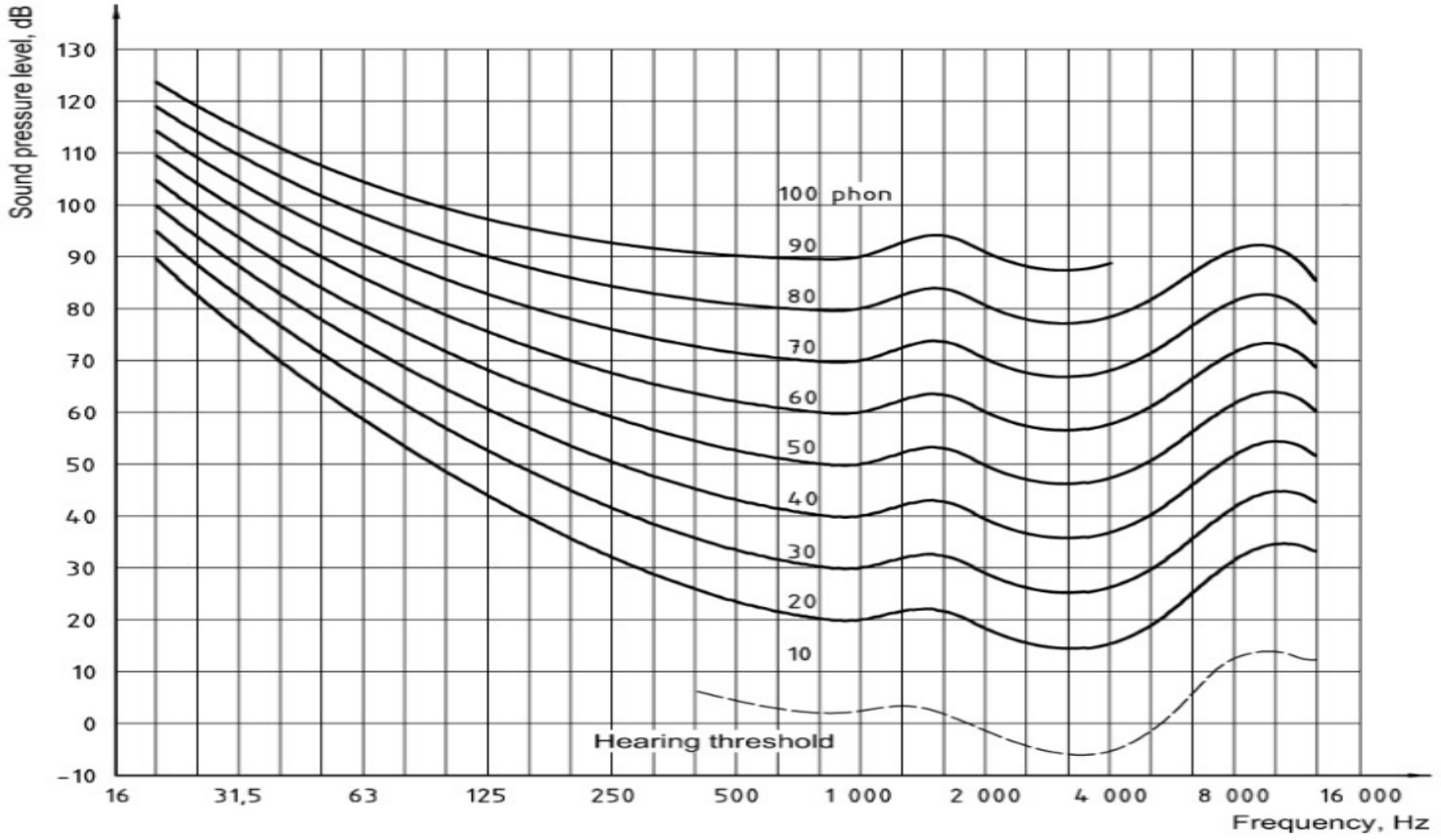
- We can check more detailed information inside the file “eq_iir_bassboost.txt” (sof/tools/ctl/eq_iir_bassboost.txt):

```
4607827,0,116,50331648,0,0,0,0,116,2,1,0,0,0,0,0,0,2,2,0,0,0,0,3227172081,2141520527,536653443,3221660410,536653443,0,1638
4,3260252783,2107733822,161646111,3961037800,172645501,4294967294,27910,
```

Note the ‘2,2’ in the red circle, the first 2 means two biquad filters used to achieve the BassBoost EQ. For audio left/right channels share same biquad filters, the total filters used is also 2 (the second 2 here). Numbers following the red circle are two filters biquad coefficients.

SOF FIR EQ Example – LOUDNESS(1)

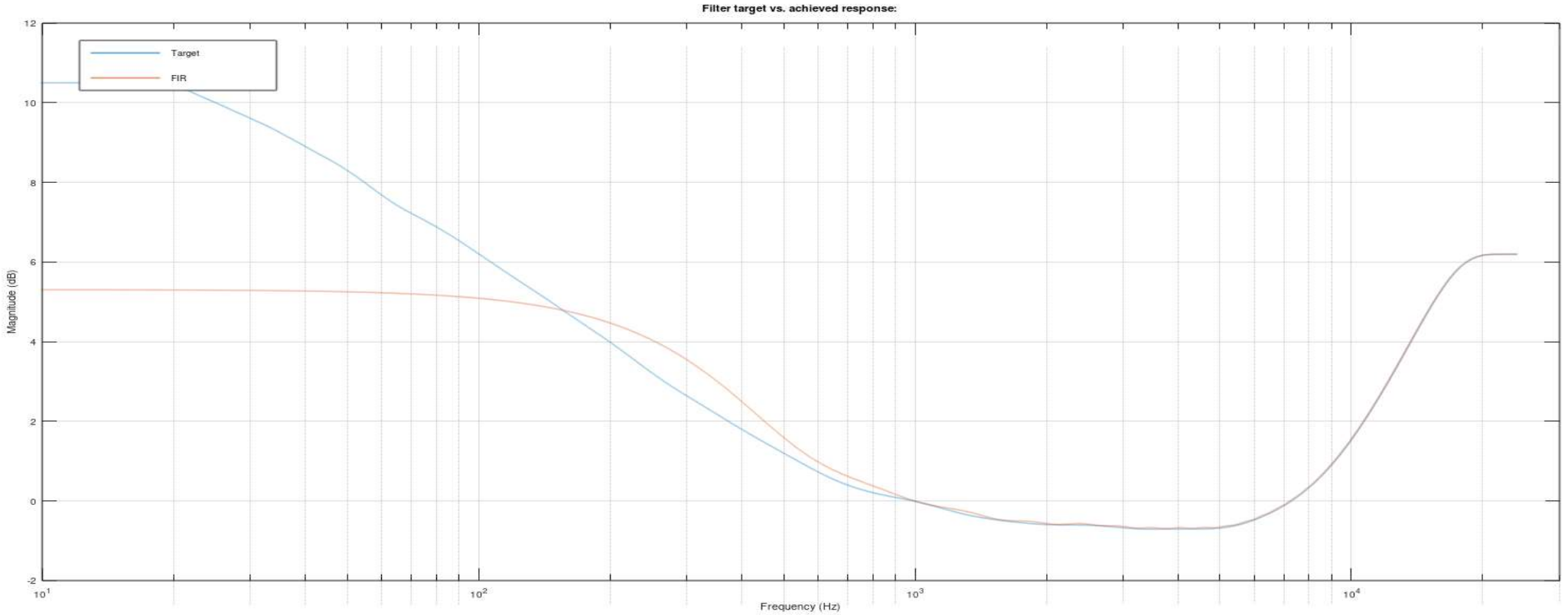
- Loudness Curves is defined by ISO 226: 2003



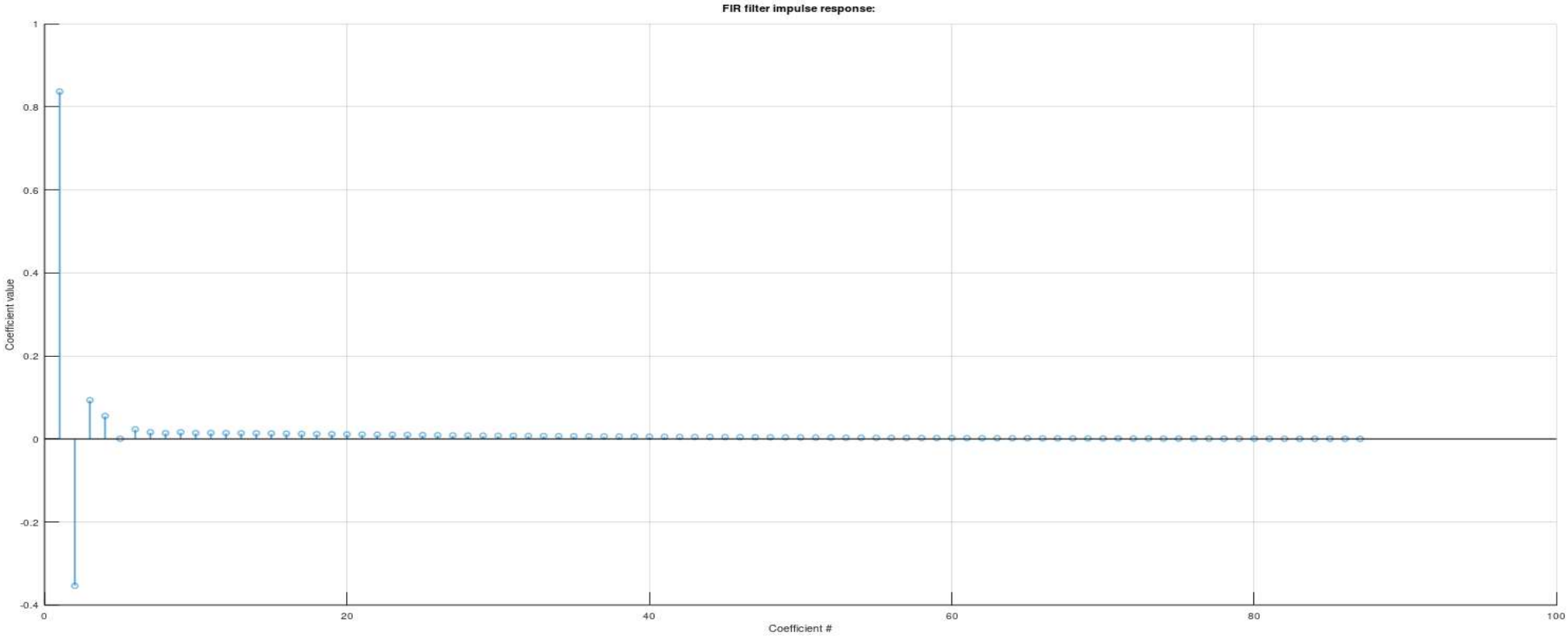
SOF FIR EQ Example – LOUDNESS(2)

- FIR EQ MATLAB script example is available in SOF source code path sof/tools/tune/eq/example_fir_eq.m, the loudness filter is implemented as function loudness_fir_eq().
- In function loudness_fir_eq(), it will sample one of the Loudness curve, call eq_compute() function to calculate the filter coefficients, and then call eq_plot() to draw the filter effect diagram.
- eq_compute() will finally call function compute_linph_fir() in file sof/tools/tune/eq/eq_compute.m to calculate the filter coefficients.
- In function compute_linph_fir(), it will call ifft() function to do the IFFT operation and then call kaiser() function to get the FIR EQ filter coefficients.
- The FIR EQ filter coefficients will be stored to text file ‘eq_fir_loudness.txt’.

SOF FIR EQ Example – LOUDNESS(3)



SOF FIR EQ Example – LOUDNESS(4)



HiFi4 DSP Acceleration to EQ Filters Calculation

HiFi 4 DSP General Introduction

- i.MX8 Audio DSP IP is Cadence's HiFi 4 DSP. Candence's HiFi 4 DSP is 32-bit processor based on Tensilica Xtensa LX Processor architecture. Following are useful links for learning Cadence's HiFi 4 DSP and Tensilica Xtensa LX Processor architecture:
  - Cadence's HiFi 4 DSP general introduction:

    https://www.cadence.com/en_US/home/tools/ip/tensilica-ip/hifi-dsps/hifi-4.html
  - Cadence's HiFi 4 DSP online training

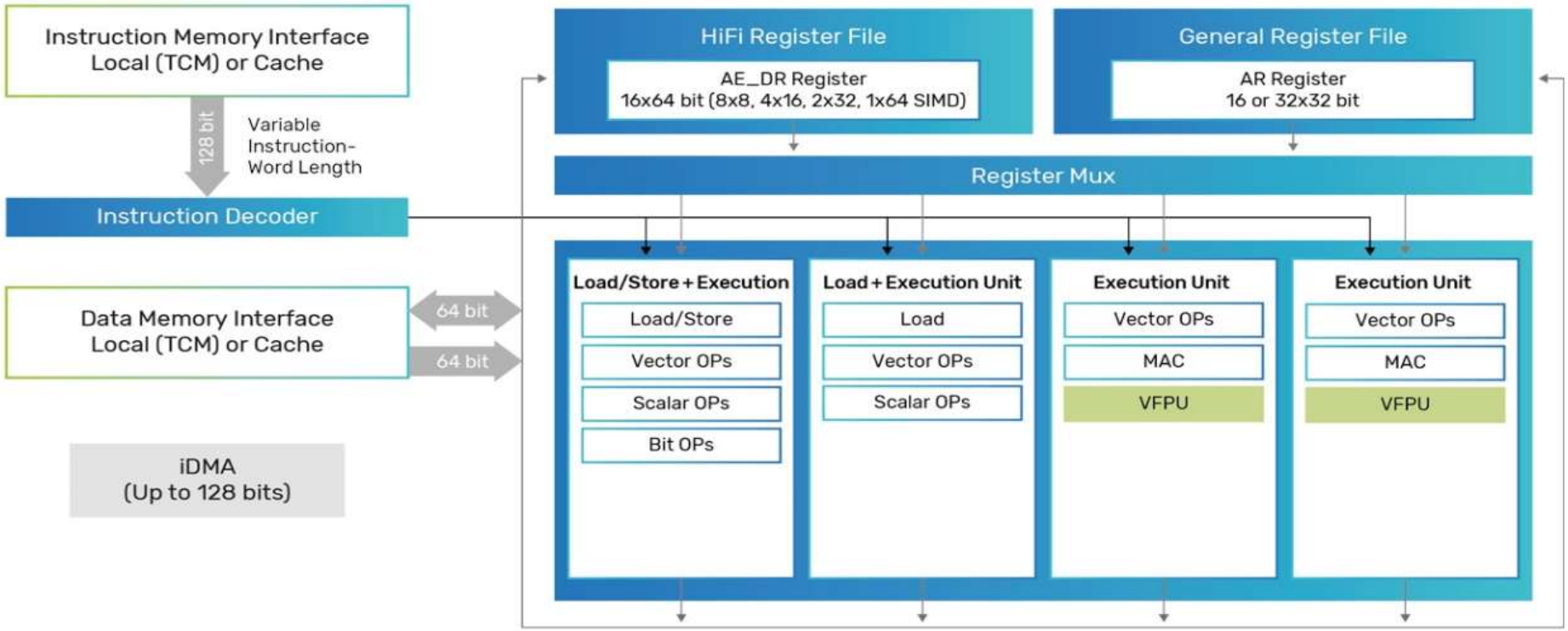
    https://www.cadence.com/en_US/home/training/all-courses/86227.html

    (8 hours training, need to register and get approval)
  - Tensilica Xtensa LX Processor Fundamentals online training

    https://www.cadence.com/en_US/home/training/all-courses/86037.html

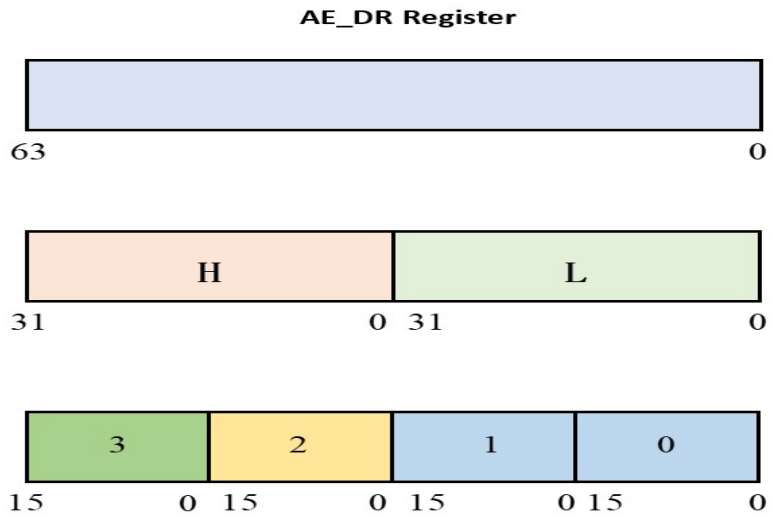
    (16 hours training, need to register and get approval)

HiFi 4 DSP Block Diagram



HiFi 4 DSP Key Registers Definition

- AR register – 16 or 32 visible 32-bit address register (AR). However, the AR registers are not restricted to holding addresses, they can also hold data.
- AE_DR registers – 16 entry 64-bit registers. Each register can hold one or two, 24 or 32-bit operands, one or four 16-bit operands or one 56- or 64-bit operand as shown in Figure below. The separate halves or quarters of the register are always separate data items.



HiFi 4 DSP MAC Instruction

- AE_MUL<accum_type>[F][DQPC]<precision>{R,RA}[S][U/US].[EP].[spl_type].<specifier>
  - Accum_type
    - Single MAC: nothing, A or S for single MAC or SIMD MAC
    - Dual MAC: [Z]{AA, AS, SA, SS}, Z clearing accumulator before add/sub
    - Quad MAC: only AAAA for quad dot-product MAC supported so for
  - Fractional MACs have a F
  - Dual MACs have a D and quad MACs have a Q
  - P (packed) MACs pack their result into lower precision
  - Precision is 16, 24, 32, 32 x 16 for non-SIMD MACs, and append X2 or X4 for SIMD
  - R MACs perform a symmetric round, RA MACs perform an asymmetric round
  - S MACs saturate – only if there are no guard-bits
  - U MACs are unsigned by unsigned, US MACs are unsigned by signed
  - EP refers to use 72-bit extended precision accumulator
  - spl_type is FIR type
  - Specifiers select elements involved in MAC – [HL3210][HL3210]
    - 24/32 bit: H or L
    - 16 bit: 3,2,1 or 0

HiFi 4 DSP MAC Instruction Example

- AE_MULZAAFD32S.HH.SS

  Dual 32x32-bit fractional MAC with 64-bit result

- AE_MULSP32X2

  Two-way SIMD 32x32-bit signed integer multiply

- AE_MULAF32X16_L0

  Single 32x16-bit fractional MAC taking operands from the low 32, 16-bits of the register

- AE_MUL32U_LL

  Single 32-bit unsigned integer multiply

- AE_MULAF32R_HH

  Single 32x32-bit fractional MAC taking operands from the high 32 bits of the register

- AE_MULAFD32X16X2_FIR_HH

  Quad 32x16-bit fractional MAC, compute two output elements

EQ Filters Calculations

- Recall LTI discrete time system transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b(r) z^{-r}}{1 + \sum_{k=1}^{N} a(k) z^{-k}} = \sum_{n=-\infty}^{\infty} h[n] z^{-n}$$

When coefficients above are determined by EQ filters design, we can get corresponding time domain difference equation

$$y[n] = -\sum_{k=1}^{N} a(k) y[n-k] + \sum_{r=0}^{M} b(r) x[n-r]$$

We can see only Multiply and Accumulate calculations are involved for EQ filters. For HiFi 4 DSP, the MAC engines are used for Multiply and Accumulate calculations acceleration. EQ Applications can take HiFi 4 DSP MAC Instructions for filters calculations.

SOF Audio Algorithms

- SOF provided several audio processing algorithms which are listed by https://thesofproject.github.io/latest/algos/index.html

- Algorithms source code is in SOF directory sof/src/audio :

```
asrc            codec_adapter  dai.c   eq_iir                  Kconfig       mux           selector   tdfb
buffer.c        component.c    dcblock google_hotword_detect.c kpb.c         pcm_converter smart_amp  tone.c
channel_map.c   copier.c       drc     host.c                  mixer.c       pipeline      src        volume
CMakeLists.txt  crossover      eq_fir  igo_nr                  multiband_drc rtnr          switch.c
```

IIR EQ Filters Calculations

- IIR EQ algorithm is in SOF directory of/src/audio/eq_iir .
- IIR EQ coefficients (biquad format) are transferred to HiFi 4 DSP firmware by IPC
- Key function is eq_iir_copy(), it will finally call iir_df2t() function to do IIR EQ filters calculations.
- iir_df2t() function implementation depends on the compiler type:
  - gcc compiler: sof/src/math/iir_df2t_generic.c
    IIR EQ filters multiply and accumulate calculations are done by generic C.
  - Cadence xcc compiler: sof/src/math/iir_df2t_hifi3.c
    IIR EQ filters multiply and accumulate calculations are done by HiFi 4 DSP MAC instructions and thus MAC engine

```
/* Compute 1st delay d0 */
acc = AE_SRAI64(*delayp, 1); /* Convert d1 to Q18.46 */
delayp--; /* Point to d0 */
AE_MULAF32R_LL(acc, coef_b2b1, in); /* Coef b1 */
AE_MULAF32R_LL(acc, coef_a2a1, tmp); /* Coef a1 */
acc = AE_SLAI64S(acc, 1); /* Convert to Q17.47 */
*delayp = acc; /* Store d0 */
delayp++; /* Point to d1 */
```

FIR EQ Filters Calculations

- FIR EQ algorithm is in SOF directory of/src/audio/eq_fir .
- FIR EQ coefficients (biquad format) are transferred to HiFi 4 DSP firmware by IPC.
- Key function is eq_fir_copy(), it will finally call functions in sof/src/math/fir_generic.c or sof/src/math/fir_hifi3.c to do FIR EQ filters calculations.
- FIR EQ filters calculation functions implementation depends on the compiler type:
  - gcc compiler: sof/src/math/fir_generic.c

    FIR EQ filters multiply and accumulate calculations are done by generic C.
  - Cadence xcc compiler: sof/src/math/ fir_hifi3.c

    FIR EQ filters multiply and accumulate calculations are done by HiFi 4 DSP MAC FIR instructions and thus MAC engine

```
/* Quad MAC (HH)
 * b += d0_h * coefs_3 + d0_l * coefs_2
 * a += d0_l * coefs_3 + d1_h * coefs_2
 */
AE_MULAFD32X16X2_FIR_HH(b, a, d0, d1, coefs);
d0 = d1;
```

Summary

Summary

- This presentation shows how to set up SOF environment for i.MX8 DXP HiFi 4 DSP, and how to run the audio EQ examples.
- There are two categories audio EQ filters: IIR EQ filters and FIR EQ filters. IIR EQ filters can be designed based on Analog filters like Butterworth filters and then apply Bilinear Transform. FIR EQ filters can be designed with Window Method. SOF adopts MATLAB (Octave) scripts to design the filters.
- EQ filters coefficients are stored to txt files, and the coefficients will be transferred to HiFi 4 DSP firmware by IPC.
- For HiFi 4 DSP firmware, the IIR EQ/FIR EQ will take DSP MAC instructions for filters calculation, if the firmware is compiled by Cadence XCC compiler. If firmware is compiled by GCC, no DSP MAC engine hardware acceleration support.

SECURE CONNECTIONS
FOR A SMARTER WORLD