Exercise – Parametric Equalizer

In this exercise you must design and implement a 4 band parametric equalizer. Equalizers are typical used in sound production to correct the response from instruments and microphones. A parametric equalizer is in the category of multi-band variable equalizers which allow users to control three primary parameters: amplitude, center frequency and bandwidth. A biquad filer realizes each band in the equalizer where a biquad filter is a 2. order IIR filter.

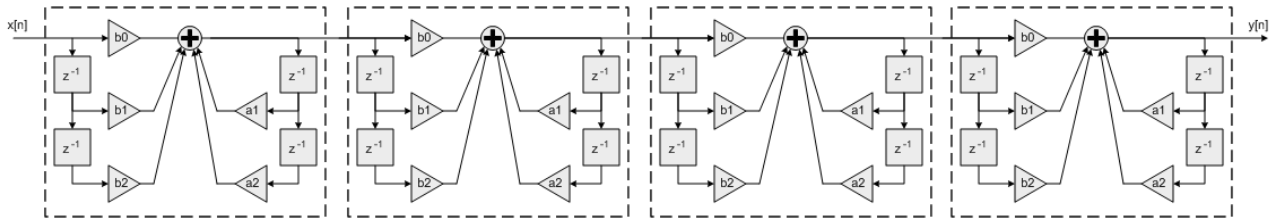
The transfer function for the biquad filter is:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

Direct form 1 realization:

$$a_0 y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2)$$

The equalizer is realized by cascading four biquad filters in a sequence. Each band can be realized as different filter types defined by the function to compute the biquad coefficients. The audio EQ cookbook[1] describes how to design different filter types like low and high shelf filters, LP and HP filters and a peaking EQ filter.



Figur 1 Cascade of 4 biquad filters

1)

Implement a peaking EQ filter where coefficients and biquad filtering are computed in separate MATLAB functions. Prototype for computing the peaking EQ filter coefficients:

```
function [ b, a ] = makePeakEQ( fs, fc, Q, A )
% Computes coefficients for peaking Equalizer
%   b = [b0 b1 b2] Nominator
%   a = [a0 a1 a2] Denominator
%   fs = sampling frequency
%   fc = Center EQ frequence
%   Q = bandwidth of EQ
%   A = gain
%   Based on the Audio-EQ-Cookbook:
%        http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt
```

[1] Audio EQ Cookbook: http://www.w3.org/2011/audio/audio-eq-cookbook.html
(Copy can be found on BlackBoard)

Parameters:

Fs = sampling frequence (48 kHz)
f0 = center frequence
Q = factor (1-20)
A = gain (0.1-5)

Computation of coefficients:

w0 = 2*pi*f0/Fs
alpha = sin(w0)/(2*Q)
b0 = 1 + alpha*A
b1 = -2*cos(w0)
b2 = 1 - alpha*A
a0 = 1 + alpha/A
a1 = -2*cos(w0)
a2 = 1 - alpha/A

Example of prototype for the biquad filter function:

```
function [ y ] = biquadFilter( b, a, x )
%Performs Biquad filtering
%  IIR 2. order, Direct Form 1
%   x - sample input vector
% Biquad coefficients
%   b = [b0 b1 b2] Nominator
%   a = [a0 a1 a2] Denominator
```

Verify and explore the implemented peak EQ filter with simulated test signal like a chirp or mixed multiple sine signals with different frequencies.

2)

Implement the peaking EQ filter in 'C++' and convert the filter coefficients using native fractional types. Test the filter with the constructed simulated test signal in MATLAB. Add missing code to the **makePeakEQ** method in the **IIRFilter.cpp** found in the 'C++' "**DSPFramework**" project.

Compare the implemented version with reference to the MATLAB implementation.
Analyze the error as the difference between the MATLAB reference and implementation in 1.15.

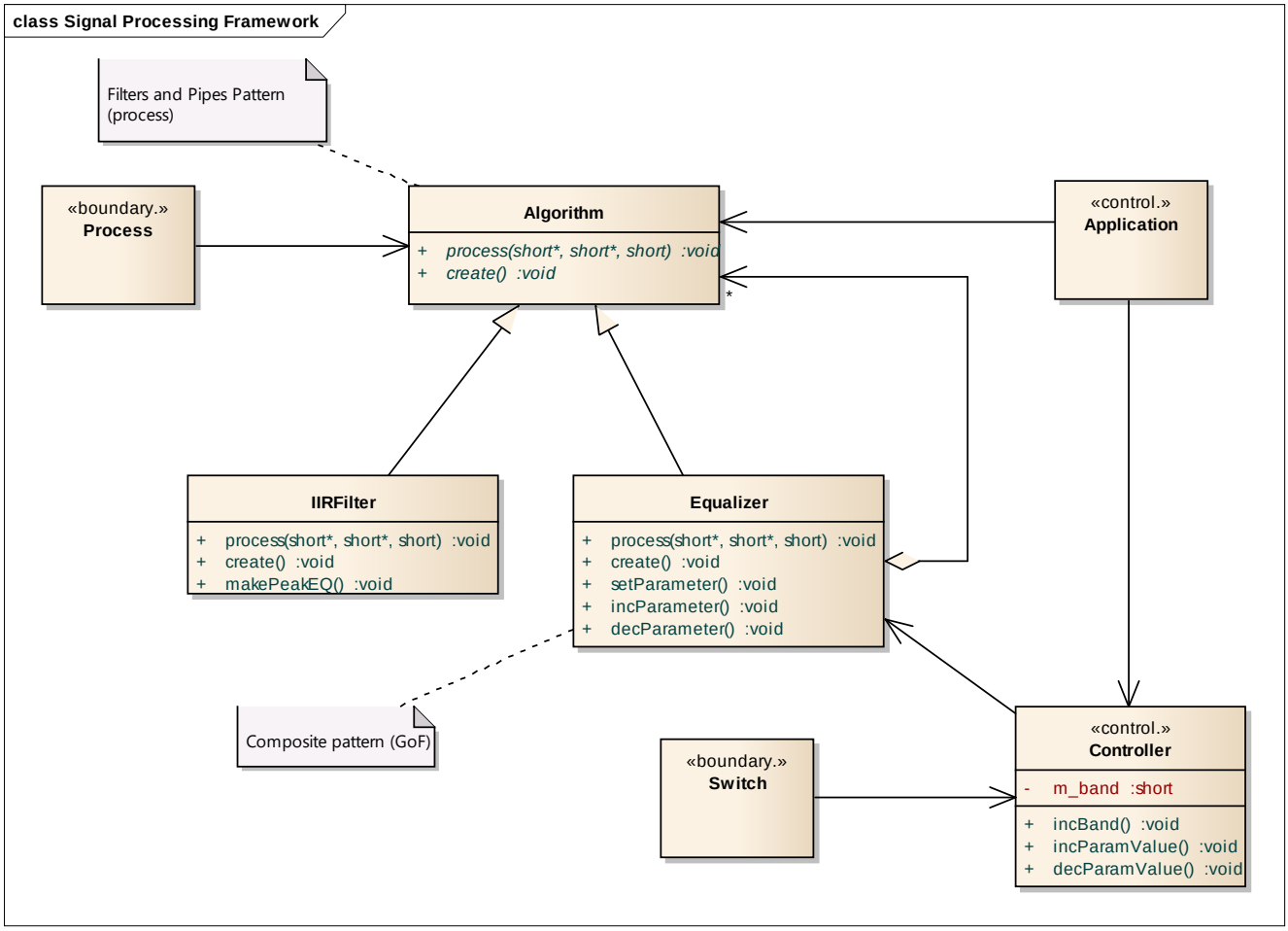
Use inspiration from the **AlgoTester** class and main function in **Application.cpp**. This class reads a text file containing a simulated test signal. The signal is used to test any algorithm that is a subclass of the **Algorithm** class. The resulting signal after algorithm processing is save in a text file. This file can be loaded into MATLAB for comparing with the MATLAB golden reference model.

3)

Now implement the 4 band equalizer by combining 4 biquads in cascade using the DSP Framework as illustrated in the UML class diagram in figure 2 and used in the **"DSPFramework"** project (See slides for more details).

The framework uses switch buttons to change the EQ parameters. Pressing **SW7** toggles between bands 1-4. The center frequency for band 1-4 is set in the **create** method of the Equalizer class. **SW4** increases the gain and **SW5** decreases the gain in steps of 0.1. The LEDs indicates the

selected band and gain being adjusted. Fill in the missing code in **Equalizer.cpp** marked with TODO.



Figur 2 Design framework using the composite, filters and pipes design patterns

4)

Test and verify the equalizer. Measure the memory and cycle usages for the IIR filter. Try to enable optimization for speed on the **IIRFilter.cpp** and measure the speedup. What is the percentage of cycle usage for the Equalizer including four bands when we are sampling with 48 kHz and the CPU clock is 600 MHz?

Optional - 5)

Add a low shelf for the first band and a high shelf filter for the last band. Verify it by adjusting parameters using the switch buttons.