Exercise – Dynamic filtering

In this exercise, you must design and implement a dynamic filter that suppresses undesired signals with a specific frequency in an audio signal. The dynamic filter should search for magnitude peaks in the frequency area 10 Hz – 20 kHz and suppress the maximum peak by use of a notch filter.

The blocks of the algorithm is illustrated in figure 1. The Notch block suppresses the specific frequency set by the MaxPeak block, which finds the maximum peak of the magnitude response. The magnitude response is computed by the FFT and Magnitude blocks. The input signal is sampled with a rate of 48 kHz (16 bits) and the resolution of the frequency spectral should be ~50 Hz.

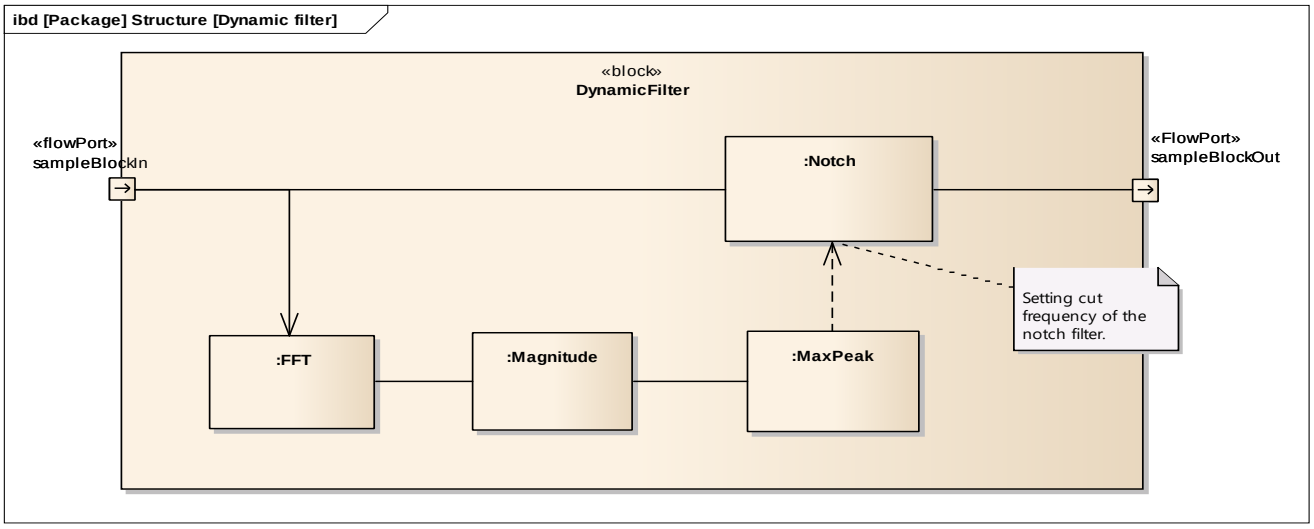

Figure 1 Block diagram of the dynamic FFTNotch filter

a)

Compute the number of N points needed for the FFT ($\boldsymbol{N = f_s/\Delta f}$). Implement the FFT and Magnitude blocks using the DSP run-time library. See functions: **twidfftrad2_fr16, rfft_**fr16 and **fft_magnitude_fr16** in the CrossCore compiler manual. Use the DynamicFilter framework similar to previous exercise to implement the DynamicFilter in a new class as illustrated in figure 2.

NB: Set switch pins 5 and 6 of SW9 on the EZ-KIT to ON, since the DSP framework is configured to operate in I2S mode.

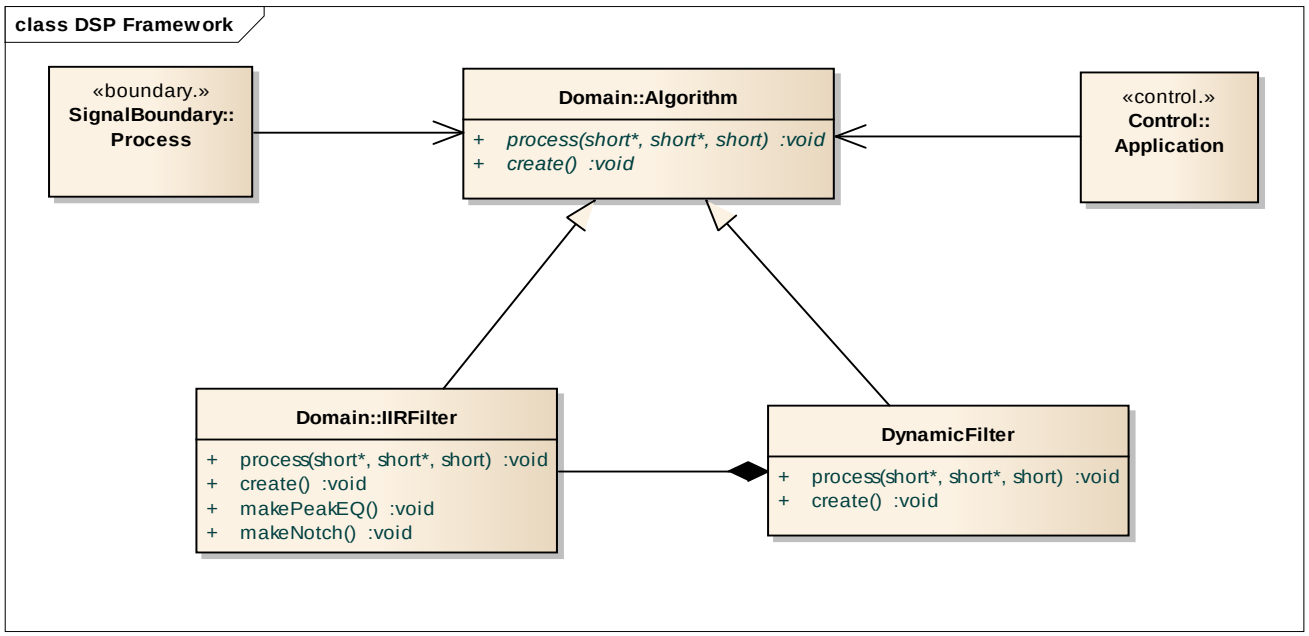

Figure 2 Class diagram for DSP design framework with the FFTNotch filter

b)

Test the algorithm by creating a synthetic signal from MATLAB that contains sound mixed with a 1 kHz sine signal. Use the AlgoTester.cpp in the DynamicFilter project. Try with different test signals.

c)

Extend the DynamicFilter to find the maximum peak of the magnitude spectral. Extend and verify the **findMax** method in DynamicFilter.

d)

Use the IIRFilter from previous exercise which is extend with a notch filer. The makeNotch method (figure 2) computes coefficients based on a pair of complex-conjugated zeros on the unit circle, as described pages 91-92 in the Embedded Signal Processing book.

Whenever a new peak is found it is signaled to the main loop that computes new coefficients by calling the **updateDynFilter** method of the DynamicFilter. This is to ensure we have sufficient of cycles to compute new filter coefficients.

e)

Test the complete system and verify the result by plotting that the sine signal is suppressed.

Measure the cycle usages by using the library in cycle_count.h. (See slides for more details)

Extra:

h)

Extend the notch filter being able to suppress more signal peaks in the spectral. Implement the IIR filter as a cascade of more biquad filters with N stages. (See slides for more details)