Design Low-Pass and High-Pass Filters for Audio Equalizer

Ziyue Yang 20010516-9536
Boyue Jiang 20001227-1953

I. INTRODUCTION

In this project, the authors design the necessary components for equalizing a two-channel stereo audio signal sampled at 48 kHz, containing a low-pass digital filter and a high-pass digital filter, which will later be deployed on the Arduino Due platform with a 32-bit ARM microprocessor that has limited computational resources. Since the platform does not possess floating-point unit ability, the equalizer is processed using a fixed-point implementation. Hence, the effect of quantization is taken into consideration when designing the filters.

II. PROJECT SPECIFICATIONS AND METHODOLOGY

Figure 1 demonstrates the structure of the equalizer utilized in the project. The input audio signal denotes as $x[n]$, and H_L / $h_L[n]$ is a low-pass filter. The component inside the blue box is an equivalent high-pass filter, where $h_H[n] = \delta[n-m] - h_L[n]$. The low-pass and high-pass filters share an identical cutoff frequency of 3 kHz for analog signals, which is equivalent to 1/16 for normalized cutoff frequency ν_c. After passing through the filters, the high-frequency component of the signal, $x_H[n]$, together with the low-frequency component, $x_L[n]$, are respectively multiplied by two coefficients and then summed to form the equalized signal $y[n]$. By changing the coefficients α_L and α_H, the amount of low and high-frequency contents in $y[n]$ can be controlled.

(Task 1)

Taking $\alpha_L = \alpha_H = 1$ as an example,

$$\begin{aligned} x_L[n] &= x[n] * h_L[n], \\ x_H[n] &= x[n-m] - x[n] * h_L[n] = x[n-m] - x_L[n], \\ y[n] &= 1 \times x_L[n] + 1 \times x_H[n] = x[n-m], \end{aligned} \tag{1}$$

where the equalized signal is the delayed version of the original audio signal under this condition.

(Task 2)

The key component in the equalizer is the low-pass discrete filter H_L / $h_L[n]$, whose cutoff frequency is set to 3 kHz in analog scale, which is equivalent to 1/16 for normalized cutoff frequency ν_c, as the sampling frequency is 48 kHz. For simplicity, we design a Type I linear phase FIR filter using the classical windowing method, following these steps:

1. We determine the formula of an ideal low-pass filter in the time domain, with ν_c=1/16:

$$h_L[n] = 2\nu_c \mathrm{sinc}\big(2\nu_c(n-\tau)\big), \tag{2}$$

   where τ is a positive integer, which corresponds to the group delay of the filter.
2. Since we cannot implement an ideal filter with infinite impulse response, we truncate the ideal low-pass filter while maintaining the symmetry around $n = \tau$. The remaining filter has $N = 2\tau + 1$ taps, which is a $M = N - 1$ order filter.
3. Multiply the N-tap filter with a window function of the same length to suppress sidelobe level, while avoiding the Gibbs phenomenon.
4. Normalize the impulse response of the filter to ensure that the DC value, $H_L(0)$, equals 1. At this

stage, the desired low-pass filter is obtained.

The above process does not show much flexibility. The only two changeable parameters are N and the type of window. Due to the limited computing capacity of the Arduino, it is necessary to make a trade-off between filter performance and filter length N. A greater N implies a better filter with a longer processing time. The type of window function will largely affect the suppression level of sidelobes.

The authors' selection for N is 47 (23 for τ), which is a safe choice as the limit of the platform is handling 60 taps filters. The performances of *Hamming*, *Hanning* and *Blackman* window are tested. It turns out the *Chebyshev Window* with 45 dB sidelobe attenuation has better sidelobe suppression.

Figure 2 demonstrates the numerical results of the low-pass filter simulated by MATLAB. At cutoff frequency $\nu = \nu_c = 1/16$, the magnitude response is -6.05 dB. A rapid drop can be observed around the cutoff frequency. The filter achieves 60 dB (much greater than 40 dB) suppression for all normalized frequencies above $1/8$, which symbolizes for satisfactory performance.

(Task 4)

The low-pass filter parameters simulated by MATLAB are in floating point. The microprocessor of Arduino only possesses fixed-point units. Hence, it is necessary to make a fixed-point implementation of the above low-pass filter. The fixed-point filter is faster and more power-efficient signal processing. However, the precision and dynamic range are compromised.

Figure 3 demonstrates the process of filtering the real valued signal $x[n]$ with a fixed-point low-pass filter with F fractional bits and output ${x_L}^Q[n]$. The block inside the blue box is the equivalent fixed-point low-pass filter $h[n]$,

$$h[n] = 2^{-F} * h_Q[n], \text{ where } h_Q[n] = \text{round}(h_L[n] * 2^F). \quad (3)$$

"round()" denotes rounding to the closest integer. The red curve in *Figure 4* represents the frequency response of the quantized low-pass filter when $F = 14$. From the figure, the response before $\nu = 1/8$ is almost identical to the floating-point low-pass filter. The response vibrates at higher frequencies due to quantization. The error of the fixed-point filter can be further decreased by selecting a higher F, meanwhile increasing the resource required for computing and bringing a higher chance of internal overflow. For lower F, the quantization noise will be greater, resulting in a larger deviation of the frequency response of the quantized filter from that of the original filter.

In fact, if we only focus on the index demanded by the project description, which is 40 dB suppression for normalized frequencies above $1/8$. It can be barely fulfilled by setting $F = 9$ (see *Figure 5*) in the authors' design, which means that one can always choose any integer between 9 and 16 as the value of F. Since we have some redundancies in computation (Instruction is that F should be no greater than 16), we think that it is a good practice to implement a quantized filter with as good performance as the original one. Hence, $F = 14$ is chosen.

(Task 5)

Since the weights of low-pass filter are quantized, to ensure the performance of our low-pass filter design, to what extent the quantization error would affect the output signal should be taken into consideration.

We can simulate the effect of quantization numerically. Assuming $x[n]$ are integer valued signal uniformly distributed in range $[-2^{11}, 2^{11}]$. Let $x_L[n]$ donates the result of filtering $x[n]$ with the float-point low-pass filter $h_L[n]$ designed in Task 2. The output in *Figure 1*, ${x_L}^Q[n]$, can be computed by convoluting $x[n]$ with the quantized filter $h_Q[n]$

$$x_L[n] = x[n] \circledast h_L[n];\ {x_L}^Q[n] = x_L[n] \circledast h_Q[n]. \quad (4)$$

Furthermore, the signal to quantization noise ratio (SQNR) can be computed via

$$\text{SQNR} = \frac{\mathbb{E}\{{x_L}^2[n]\}}{\mathbb{E}\{(x_L[n] - {x_L}^Q[n])^2\}}. \quad (5)$$

For accurate approximation, we sampled a 100000-lengthed signal $x[n]$, and compute the corresponding SQNR following formula (5). $\text{SQNR} \approx 69$ dB when using 14 fractional bits implementation. As F increases, SQNR shows an increasing trend, which matches the theory. Again, if we consider the case when F=9, SQNR=38 dB, where the noise level is significantly greater than that of F=14.

(Task 3)

With the float-point design of low-pass filter in hand, it is relatively straightforward to design a high-pass filter with the same cutoff frequency. The block inside the blue box of *Figure 1* is a high-pass one that satisfy the needs, where

$$h_H[n] = \delta[n - m] - h_L[n]. \quad (6)$$

Since $h_L[n]$ is a 47-tap Type I linear phase low-pass FIR filter, $h_H[n]$ is also 47 taps. To guarantee that $h_H[n]$ has linear phase, it must possess symmetric property as it is in $h_L[n]$. Hence, $\delta[n - m] = 1$ at $n = 23$, which is the symmetrical center, as well as the group delay, of $h_L[n]$. Thus, $m = 23$. The magnitude response of $H_H[v]$ is shown in *Figure 1*.

III. CONCLUSION

In this project, the authors design the necessary components of an audio equalizer and analysis the potential issues caused by quantization. Our main task is to design a 47-tap Type I linear phase low-pass filter with 3 kHz cutoff frequency sampled at 48 kHz by windowing method with *Chebyshev Window*. The filter achieves 60 dB suppression on sidelobes for frequencies greater than 6 kHz. Furthermore, the authors design a linear phase high-pass filter, by subtracting a pulse signal with the impulse response of the designed low-pass filter. Afterwards, the authors analysis the performance of the designed low-pass filter under fixed point implementation. It turns out that it is a good practice to implement the filter with 14 fractional bits ($F = 14$), which both maintain the property of original float-point filter and is computational efficient.

This project provides an excellent opportunity for the us students to practice the knowledge learnt in lectures. We are looking forward to putting the designed components to use in the Lab later!

Appendix

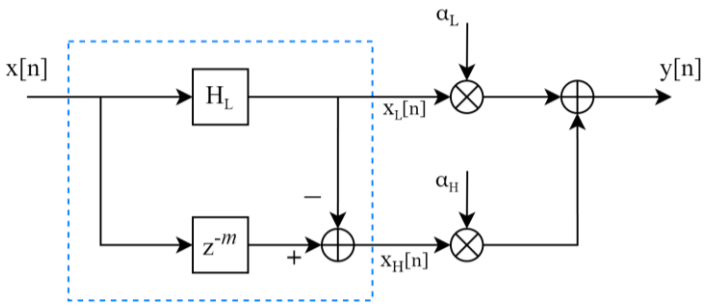

Figure 1. Basic structure of equalizer.

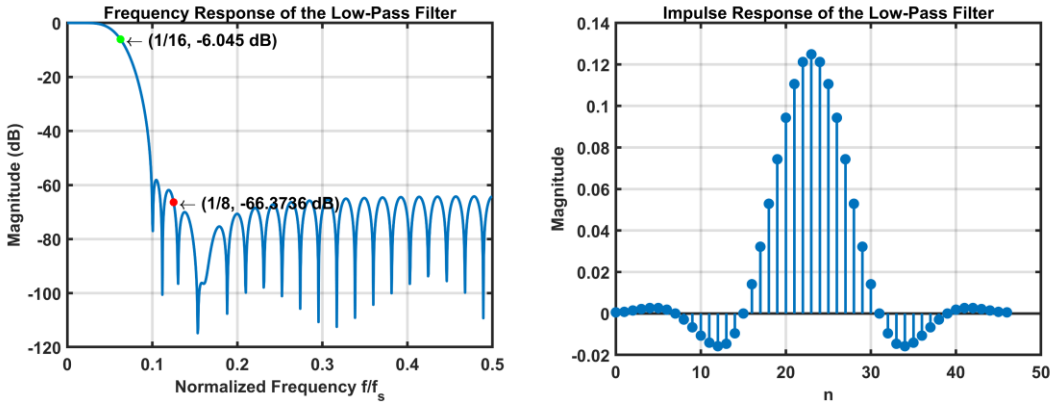

Figure 2. Frequency response $H_L(v)$ and impulse response $h_L(n)$ of 47-tap low-pass filter.

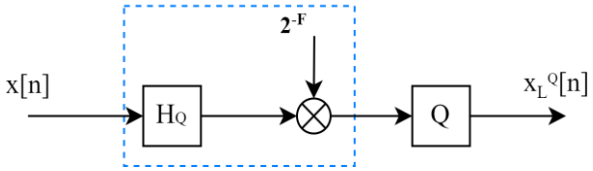

Figure 3. Signal through fixed-point low-pass filter.

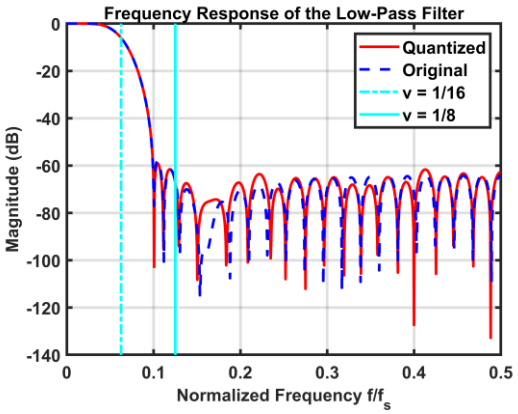

Figure 4. The effect of quantization on frequency response, F=14.

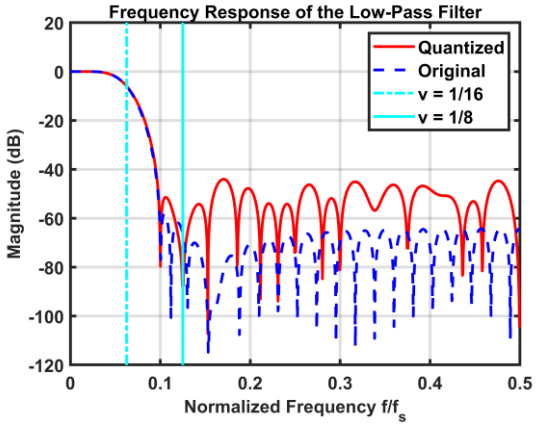

Figure 5. The effect of quantization on frequency response, F=9.

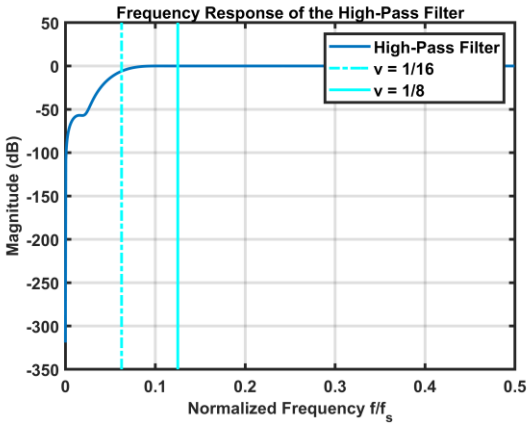

Figure 6. Frequency response $H_H(v)$ of 47-tap high-pass filter.