Digital Signal Processing Report

Yanis MOUZAOUI 20021127-T051
Arthur CHANSEL 20020105-T034

November 27, 2022

Abstract

In the first place, let us introduce the subject of the project of signal processing. The main aim of it is to create a two-channel audio equalizer on the Adruino Due platform. However, it is not that simple that's why we had to go through different steps of the design process of digital filters, by using theoretical results and verified them with the help of MATLAB.

First, we had to design a low pass filter with particular specifications. In order to fulfill these specifications we used the window method.

Then, we wanted to create a type I high pass filter with the help of the low pass.

Finally, we quantized the signal and we calculated the SQNR to make sure that our signal is more transmitted than the one of the error due to the quantization in order to create a feasible filter.

1 Introduction

We did a general analysis of an equalizer structure, in Task 1. We'll consider the variable factors to be 1, in our specific case, which gives us the delayed input as an output.

The filter needs to be a Type I linear phase causal FIR. It should have a cutoff frequency of 3 kHz, with a suppression of at least 40 dB above 6 kHz, and less than 55 taps. We had to design the filter with the right parameters to meet the conditions. From this low pass filter, we had to get it's equivalent high pass filter, still being a Type I linear phase filter.

We want to be able to implement the filter with integer arithmetic. Then we had to quantize the signal as much as possible, making sure that the conditions were always met, choosing wisely the number of quantization bits. We had the condition to keep this number below 16 in order not to risk internal overflow in the later implementation.

Finally we compared the signals we got before and after the quantization, by computing the SQNR (signal to quantization noise ratio).

We used the FFT model to compute the Fourier transforms. We choose 51 taps for our filter. Then, to design it we used the method of window, more precisely the Blackman window. We computed the SQNR with the formula given for the project.

2 Theory

We began by showing that the output y[n] will always be a delayed output of x[n] if $\alpha_L = \alpha_H = 1$, regardless of the choice of H_L. You can find a proof of it in the appendix.

Then, we started to design our low pass filter, composed of the ideal filter h_I[n] and the Blackman window h_I[n]:

$$h_I[\mathrm{n}] = \begin{cases} \frac{sin(2*pi*nu*(n-m))}{pi*(n-m)} & \text{if } nu < 1/16 \\ 0 & \text{if } nu > 1/16 \end{cases}$$

In order to get a symmetrical signal we chose m equals to 26, the center tap, that way we have a type I filter. At this point, we multiplied our two signals h_I[n] and w_B[n] (obtained thanks to a matlab tool) in the time domain to get our real filter h_L[n].

We can then compute the N-point discrete Fourier transform (DFT) to get our signal in the frequency domain:

$$H_L[\mathrm{k}] = \sum_{n=0}^{50} (h_L[n] * \exp(-2 * pi * k * n/51))$$
for k=0,1,...,50

Of course, regarding the computation time, it is better to do a FFT, that's what we did to get the numerical results on matlab.

Once we finished the design of the low pass filter we were able to create the high pass filter. To do so, we took an impulse function shifted of 26 (the center tap) to keep the symmetry and get a type I high pass filter, to which we have substracted the low pass filter h_L[n]:

$$h_H[n] = \delta(n - 26) - h_L[n]$$

Once again, we can get the equivalent signal in the frequency domain by using a DFT on the signal h_H[n].

Moreover, we wanted to add quantization to our model, to that end we implement a simple formula that transform our filter:

$$h_Q[n] = [h_L[n] * 2F] * \frac{1}{2^F}$$

(where [ ] denotes rounding to the closest integer.)

Finally, we can compute the SQNR to be sure that the noise is less powerful than the signal of the filter, once again we do it thanks to a simple formula:

$$SQNR = \frac{E(h_L[n]^2)}{E((h_L[n] - h_Q[n])^2)}$$

Where $E(x[n]^2) = \frac{2^{-2F}}{12} * \sum_{n=0}^{50}(x[n]^2)$.

3 Numerical Results

See all the figures in Section appendix.

The design of our low pass filter is shown in Figure 1 in time domain, and in Figure 2 in frequency domain.

The high pass filter is shown in Figure 3 in time domain, and in Figure 4 in frequency domain.

The quantized low pass filter is shown in Figure 5 in time domain, and in Figure 6 in frequency domain. We found out that 9 bits was the optimal number to get the better quantization signal within the conditions.

The SQNR value in dB is 39.2681, which is greater than 0. That means that our signal is significant compared to the noise caused by the quantization.

4 Conclusions

All things considered, we saw how a low pass filter was realizable with MATLAB to be later used for an implementation. The method of window was very helpful to make design within the conditions.

A high pass filter is easily realizable from this point, as it simply is the shifted low pass subtracted to the input.

We conclude from the SQNR value that the quantized filter is very similar to the initial low pass filter. Thus, it is realizable to simply use a few bits to store the whole data. We'll save a lot of computation time, and memory.

Appendix

Task1: We suppose that α_L and α_H are equal to 1. Then, thanks to the schematic, we can compute the output y[n]:

$$y[n] = (x[n - m] - x_L[n]) \cdot \alpha_H + x_L[n] \cdot \alpha_L \quad (1)$$

$$y[n] = x[n - m] - x_L[n] + x_L[n] \quad (2)$$

$$y[n] = x[n - m] \quad (3)$$

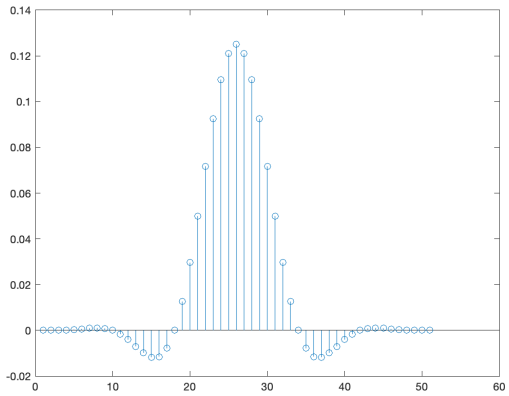

Figure 1: Low pass filter h_L (time domain).

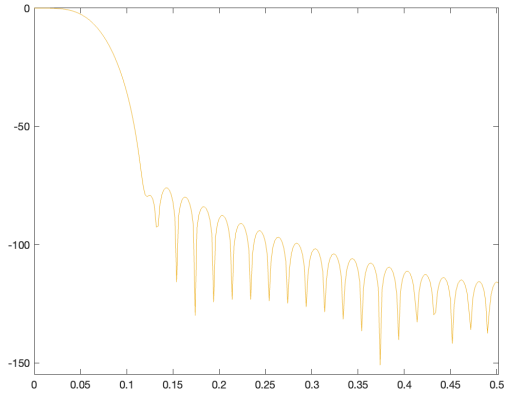

Figure 2: Low pass filter H_L (frequency domain).

References

[1] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε* `http://tug.ctan.org/info/lshort/english/lshort.pdf`.

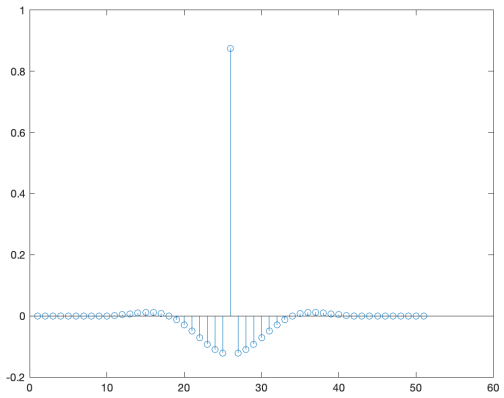

Figure 3: High pass filter h_H (time domain).

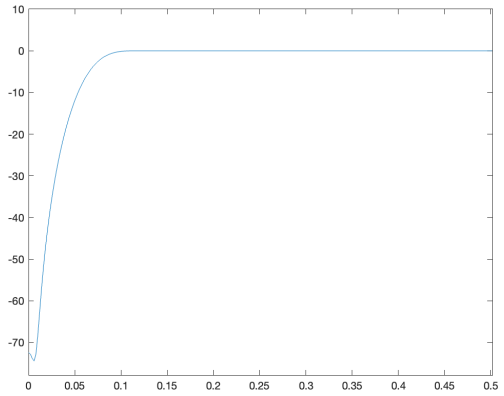

Figure 4: High pass filter H_H (frequency domain).

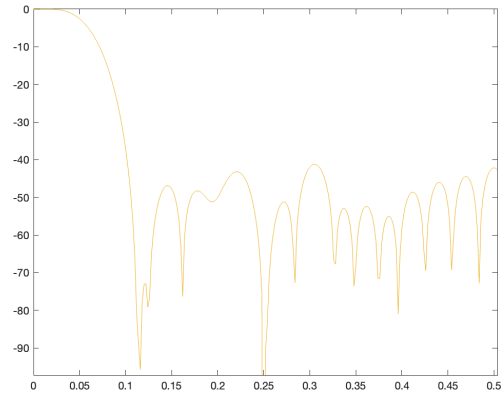

Figure 6: Quantized low pass filter H_Q (frequency domain).

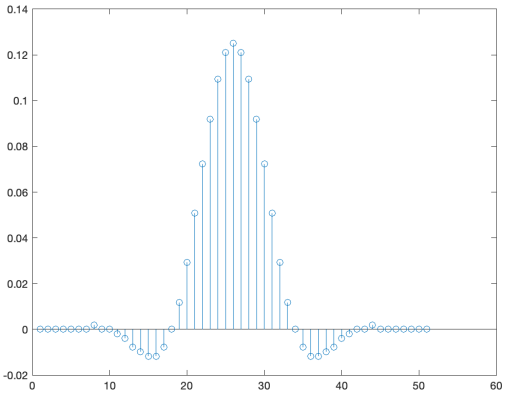

Figure 5: Quantized low pass filter h_Q (time domain).