Equalizer

ECE 4220 Real Time Embedded Systems

Final Project Report

Xin Du

14149625

Instructor: Dr. Desouza

December 2011

Abstract

This is a project in which a kind of equalizer is designed. It can change frequency distribution of input analog signal and then display the modified waveform on oscilloscope screen. The project is mainly based on TS-7250 board with 200MHz ARM9 CPU and a self-made DAC hardware part.

Introduction

The project can be divided into two parts, hardware part and software part. The hardware part focuses on A/D conversion and software part mainly focuses on conversion between time domain and frequency domain as well as the controller part. The input signal is from wave generator and in this project it is sine wave. After passing the ADC part, the original signal is digitalized. Then after FFT, the signal can be mapped to frequency domain. The frequency domain is divided into different small bands so that the range of modification frequency can be customized. After the frequency domain parameter is changed by the user, IFFT will map it to time domain again and it will be output from GPIO of TS-7250. Then it goes through a DAC circuit and become analog signal again and displayed on oscilloscope screen.

Problem Description

Signal processing on frequency domain is a widely used technique on many engineering aspects. Many feature of signal can only be observed and analysed on frequency domain. When manipulation of the frequency distribution is needed .A equalizer can come to use.

The equalizer in my project is a digital one and can be modified only by software. So before using it, data should be collected from the real world. Therefore, ADC part is a necessary module for this project. Also, we want the modified signal to come back to real world to be seen or heard, therefore a DAC module is essential.

As a qualified equalizer, it should be able to decide the frequency range which need to be modified, and how the frequency on this band should be modified. It also should be able to either amplify or reduce the signal amplitude. Therefore, depending on

different settings of the parameter, it can also be used as a filter or an amplifier whose frequency range can be customized.

The equalizer needs a very strict requirement of sampling interval, because a little mistake on sampling frequency will change the frequency features of the signal. Because of that, Real Time task should be used on both input and output module of the project.

Proposed Approach:

Hardware:

ADC part: On the ADC side, I use the MAX197 A/D Chip on the TS-7250 board, it has 8 channels and I used the channel 0. The chip also has a maxim conversion time of 12 us. By changing the value of its control register, the conversion can be unipolar or bipolar, 5V or 10V. I chose the unipolar, 5V. Although the input sine wave has negative values, I added a 2.5 DC offset to the input signal. The oscilloscope can filter the output DC part automatically. I use unipolar mode and amplitude is 5V .So the AC part of my input should be less than 2.5VPP. The max conversion frequency of MAX197 is 60,000Hz. In my project, the sampling frequency is 10,000Hz. Therefore it meets the requirement completely.

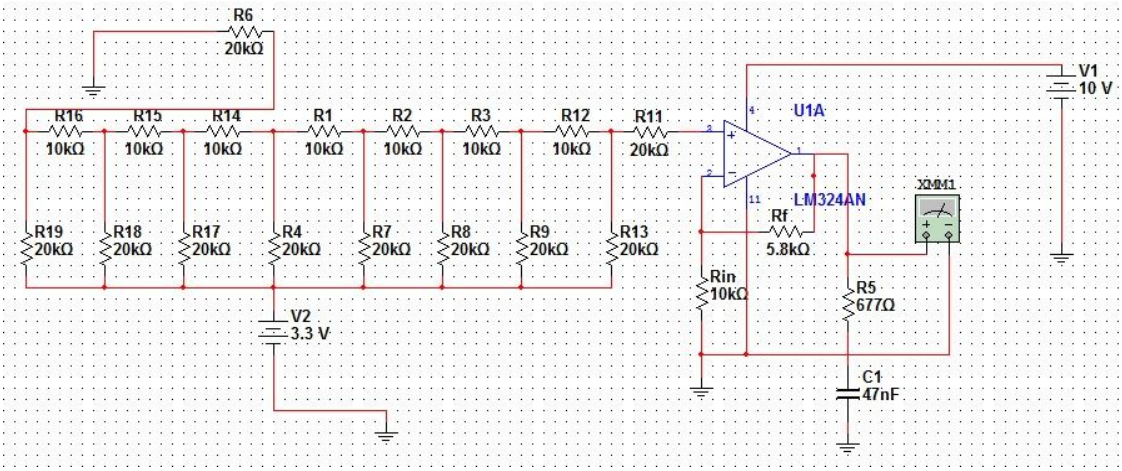
MAX197 is a 12 bit A/D chip. According to my setting, it has a resolution of +5/4096=1.22mv. Every 10us, the A/D part complete a conversion and the data is recorded by the program.

DAC part: The output is from pins of GPIO on TS7250 board which has 8 bits. Therefore the output resolution is worse than the input one. Also, the output is a parallel one, which MSB represents 2.5V, the second MSB represents 1.25V and goes on to 8th bit like this.

This project use R-2R ladder network to calculate the value of the result. The main advantage of it is that it only needs two values of registers, R and 2R. In my project, I defined R as 10k. At the end of the R-2R ladder network is a non-inverting amplifier, which is used as an amplifier to compensate the value(3.3V) from the GPIO pins on

TS-7250(I need a maxim value of 5V). The gain is 1+(Rf/Rin), so I chose 5.8k as Rf and 10K as Rin. Therefore 3.3*1.58=5.

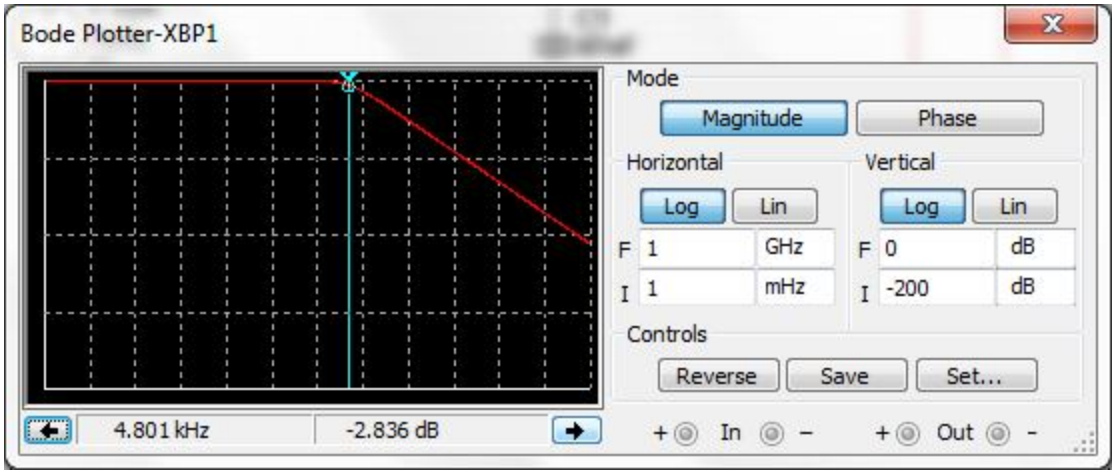
The circuit is as following. In this circuit image, all of the input is assigned as 1(3.3V). In the real hardware, they will be assigned as the GPIO output values with MSB on the right side :



The output has a very strong "jagged effect". So I use a LP filter to smooth the signal. After chose the capacitor value of 47nf and the max frequency(5k, half of the sampling frequecy). Therefore,

R = 1/(2pi*5k*47nf)=766 ohms.

The following is a Bode meter shows the frequency feature of the LP filter.



Software Part

On software part, the program should guarantee that the sampling time and output interval are strictly 10us. To do this, Real Time Tasks are used. Both input and output part are periodical task which runs 10,000 times per second.

It has to be made sure that the software part can support the hardware part well. Because I used Linux user space to program but not in the kernel. I need to map the hardware using mmap instruction, like mapping I/Os to the memory and by changing the value of the memory I can manage the hardware. On the ADC part, 3 register must be modified to setup and control MAX197 and on the output part, a data register and a direction data register need to be modified to change the value of GPIO PortB.

The next step is to decide how many points FFT to be used in this program. The higher number of FFT points will cause a higher resolution on frequency domain. The resolution= Sampling Frequency/FFT points number. In my project, I chose 256 points. So the resolution is about 39Hz. The result of FFT is also a 256 points sequence. The first point of it represents the DC value. The second point represents 39Hz, and the third one represents 39*2 Hz, etc. According to theory of Nyquist, the result is good when the input signal is under 5Khz. 5KHz is about the 128th point of sequence. The FFT sequence is symmetrical between 1-128 and 129-256. So you can see that there will be overlapped part when input signal is higher than 5KHz.

After getting the data on frequency domain, the modification can get start. When running the project, the first thing to do is to decide the number of bands need to be changed. It can support at most 4 bands. Then the program receives the start frequency, end frequency and gain ratio of each band. After getting these data, a number sequence called "fft factor" can be generated. It is a sequence shows the adjust amount of each point on the frequency domain. Then use this sequence to

multiply the value of each point on the frequency domain correspondingly. Also, the input gain ratio is in the format of Decibel and the program will change them to the proportion between voltages.

Using IFFT to change the 256 points on frequency domain to 256 points on time domain again. These result can be output as 8-bit parallel signal by GPIO and go through the DAC hardware.

The whole program should be a real time one. That means, the output should change almost immediately as the input signal changes. However, I failed to schedule the real tasks. This is mainly because the FFT and IFFT part take too much time. As I had computed, it will take almost 0.5 sec to finish a signal conversion. Because the sampling frequency is 10,000 Hz and the it is a 256 points FFT, this conversion should be done about 39 times per sec. I tried many ways but still it doesn't work. Therefore I could only make getting data, FFT/IFFT process and output process happen in order but not simultaneously. What I do isto sample only a little period of time, do the FFT/IFFT process and after that I keep outputing the adjusted result of this little period of time. Because my input signal is sine wave, so using this method is acceptable. But that also means that the program is not suitable for processing some non-periodical signal like music or voice signal. After about 20 sec, the program will reload a small period of time do the whole process again.

Result:

The result waveform is shown on the oscilloscope. Also, the result of FFT can be seen on the screen.

After starting the program, a user space appears and we can modify the number of bands we want to modify and define the start frequency, end frequency and gain ratio of these bands.

A user space look likes this:

****
Equalizer!
Sampling Frequency: 10,000Hz
Number of Points Doing FFT: 256
Resolution: 40 Hz(0-5kHz)
****

Band#	Start Frequency	End Frequency	Gain Ratio
#1	0300Hz	0500Hz	3DB
#2	0700Hz	1300Hz	-30DB

After each sampling process, on the screen it shows the result of FFT: for example, when no band is being modified, a input of 1,000 Hz sine wave is as following, you can see the number around 1kHz is obviously bigger:

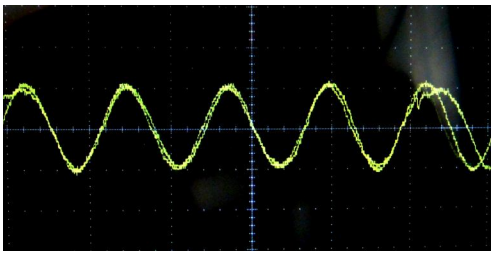
```
f=400: 1.820415 -0.688173
f=440: 6.421131 -2.909905
f=480: 1.630134 -3.562063
f=520: 3.500868 -4.149294
f=560: 3.218199 -3.695583
f=600: 2.308403 -3.287317
f=640: 6.816289 -7.046346
f=680: 2.184898 -7.976518
f=720: 2.544829 -9.104228
f=760: 1.756674 -10.920335
f=800: 0.069727 -11.845969
f=840: -2.390279 -11.598714
f=880: 0.943407 -17.181385
f=920: -10.515008 -19.456594
f=960: -8.149458 -19.106083
f=1000: -16.458062 -61.104302
f=1040: 34.830619 85.694791
f=1080: 11.520347 24.043732
f=1120: 7.719624 7.108666
f=1160: 0.282842 2.658647
f=1200: 0.270524 2.023646
f=1240: -0.981465 1.238198
f=1280: -2.853138 1.194454
f=1320: -2.431458 0.488087
f=1360: -1.534217 1.682044
f=1400: -5.345895 1.281607
f=1440: 1.287335 1.375848
f=1480: -3.765463 -0.486060
f=1520: -4.214347 0.003376
f=1560: -2.953986 0.429028
```

On the screen of the oscilloscope, we can observe the waveform of the adjusted waveform. For example, if I use the data in user space shown above:

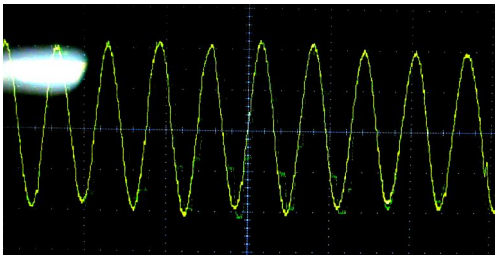
#1	0300Hz	0500Hz	3DB
#2	0700Hz	1300Hz	-30DB

The result on the screen of oscilloscope is as the following images
(each block is 1VPP):

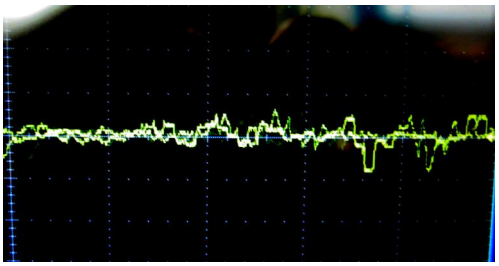
1. When input is a 1VPP, 200Hz sine wave:



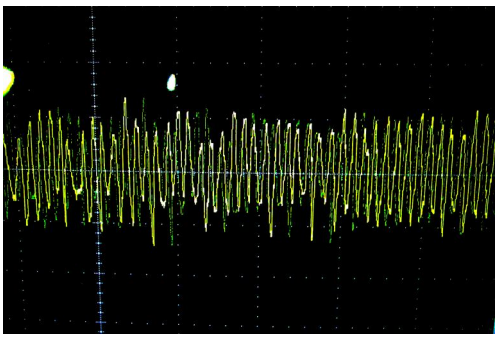
2. When input is a 1VPP 400Hz sine wave, according to the parameter, signal from 300-500 should be doubled.



3. When input is a 1VPP 1Khz sine wave, according to the parameter, signal on this band will be almost eliminated.



4. When input signal is a 1Vpp 2Khz, no modification on this band, so its amplitude is again back to 1Vpp.



Conclusions and Future Improvements

This project contains knowledge from many aspects: the signal processing, programming and hardware. I have really learned a lot from doing this project. The

project can modify the value from different frequency band though the result is not so accurate. I will try to change the FFT part and schedule the tasks again and make the whole project a completely real time one. I hope that it can process the music signal and works just like a real equalizer.