Electric Guitar Effect Processor Design with Basys3 FPGA Board and VHDL

Abstract

The goal of this project is to implement digital audio signal processing on the Basys3 FPGA board using VHDL. The main aim was to design an effect-processor, also known as a multi-effects pedal, with 2 different effect with a total of 5 effects. There is one additional effect with two different modes in the final version of the projects.

The first objective was to design a Delay effect and two different Overdrive effects. In the final version, there is one more effect which is a Tremolo effect with two different frequencies.

In this project, Basys3 FPGA board, which has a Xilinx Artix-7 FPGA, was used to implement the design. The choice of hardware description language was VHDL for my design.

Design Specification Plan

For the project, the design that I wanted to implement was an actual pedalboard just like a real effect processor. To achieve this design, the first thing that I thought about was the combination of multiple effects. In a pedalboard, separate effect pedals are connected to each other by 6.5/6.35mm audio jack cables in a serial way. This allows the user to use the combination of these effects. The clean guitar signal goes through the pedals and each pedal gives a processed output so the effect will stay on the signal before the signal goes to the next effect pedal. An example for such a pedalboard will be the Boss BCB-60 pedalboard in the picture[1] below.

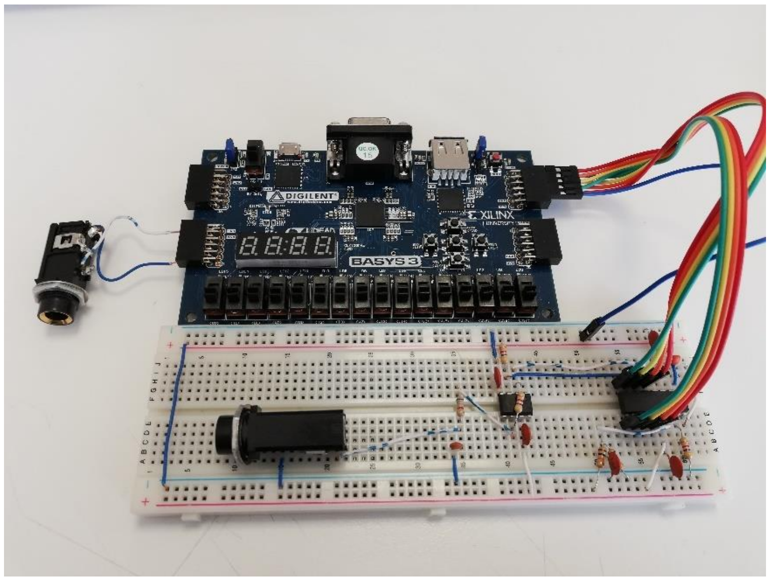
To achieve a design that is similar to a pedalboard, I planned to create my effects on separate modules and connecting them serially in order to keep the effects on the signal while it goes through all of my modules.

Before I started to implement my design, I needed to learn how to process an audio signal to create the effects. I was planning to have 3 total effects, 2 overdrive effects and one delay effect, but I also wanted to have tremolo effect on my design so I learned what these effects actually are.

Boss BCB-60 Pedalboard

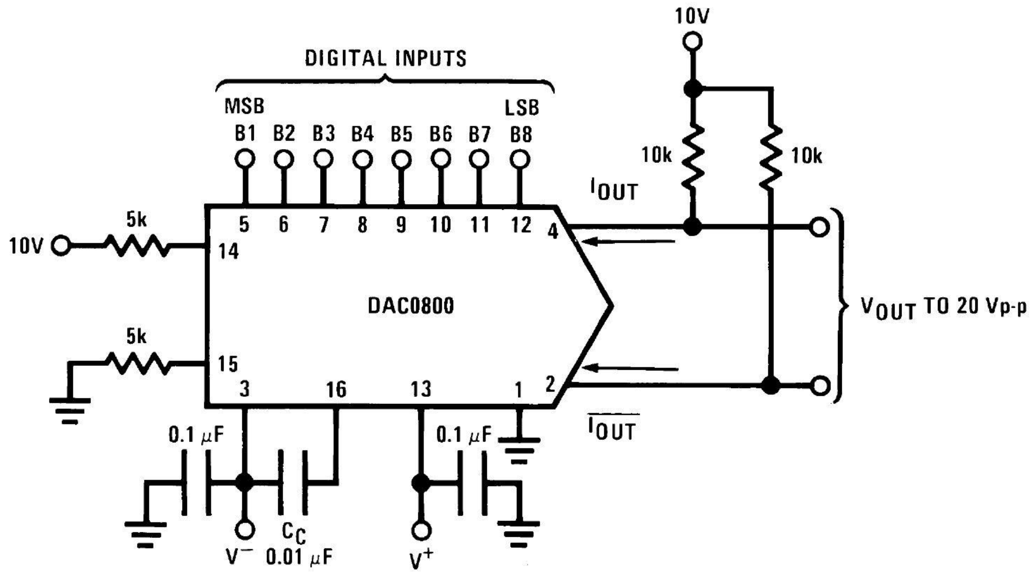
Design Methodology

To design an effect processor, the first thing that I needed to do was to get the audio signals from my electric guitar and hear a clean sound from the output of the Basys3. Since the Basys3 FPGA board doesn't have a digital to analog converter (DAC), I made a DAC circuit on a breadboard.

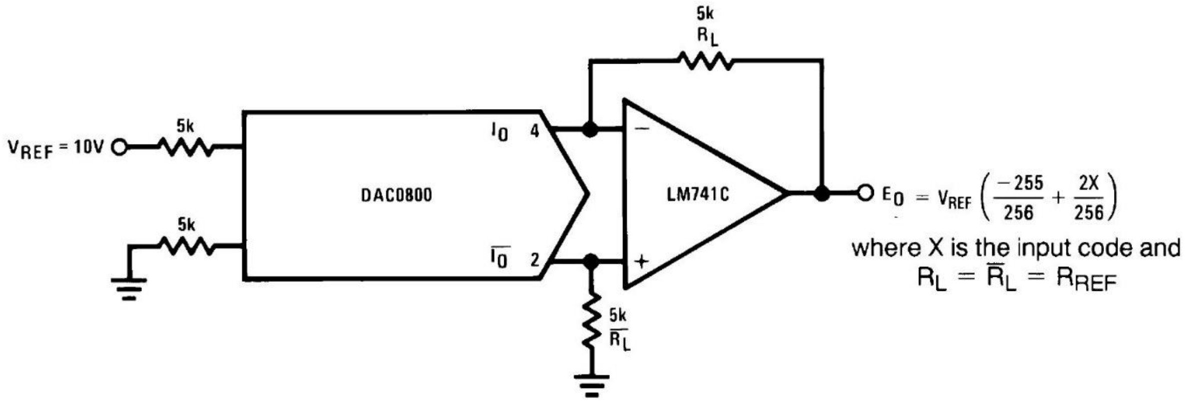


My Basys3 FPGA board connected to the DAC Circuit

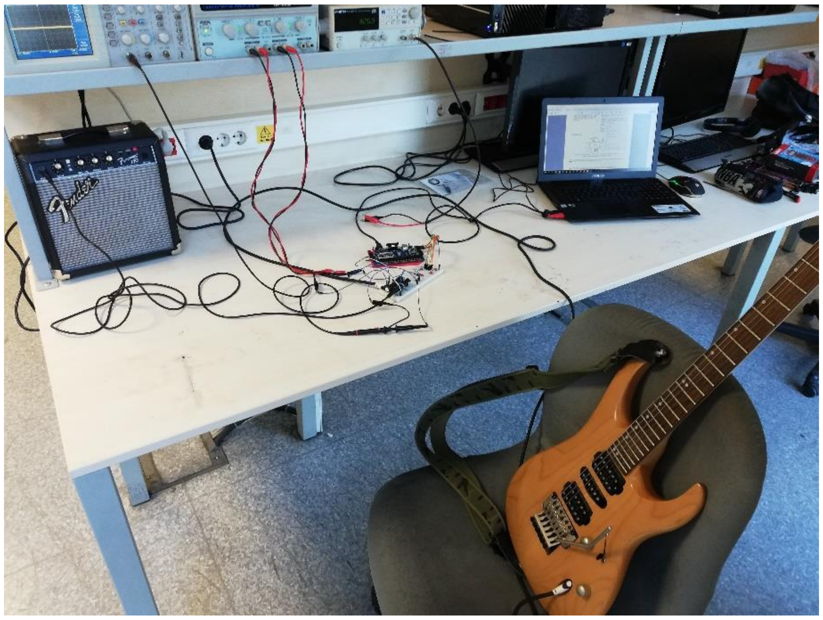
For the DAC circuit, I used a DAC0800 and a LM741 OPAMP. The reason that I used an OPAMP was the DAC0800. The outputs of the DAC0800 are in current form but I needed a voltage output, so I used the LM741 as a transimpedance OPAMP. After I built my DAC circuit, I started to design my effects.



DAC0800


V_{REF} = 10V
5k
5k
DAC0800
I_0 4
$\overline{I_0}$ 2
LM741C
5k R_L
5k $\overline{R_L}$
$E_0 = V_{REF}\left(\frac{-255}{256} + \frac{2X}{256}\right)$
where X is the input code and
$R_L = \overline{R}_L = R_{REF}$

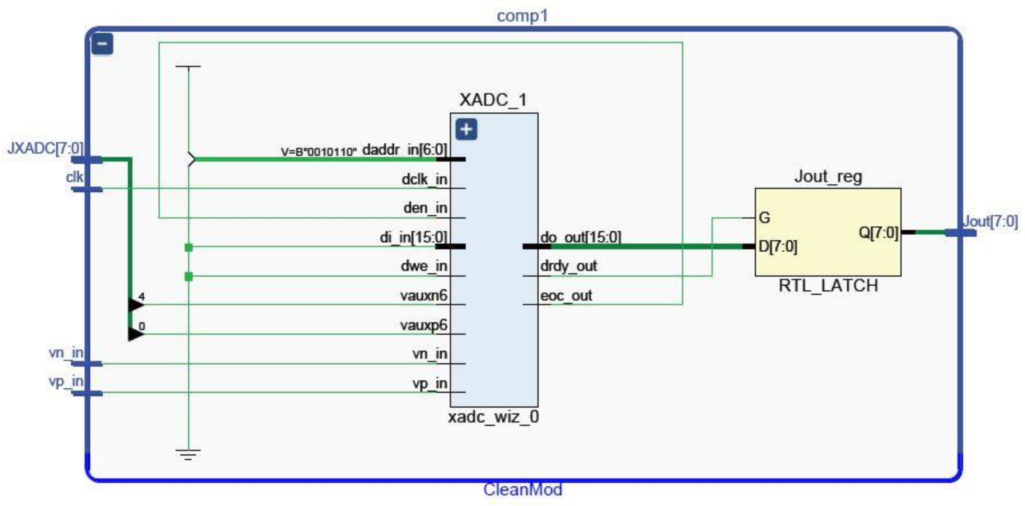
DAC0800 and LM741C as a Transimpedance Opamp



My Complete Equipment

Clean Sound and XADC

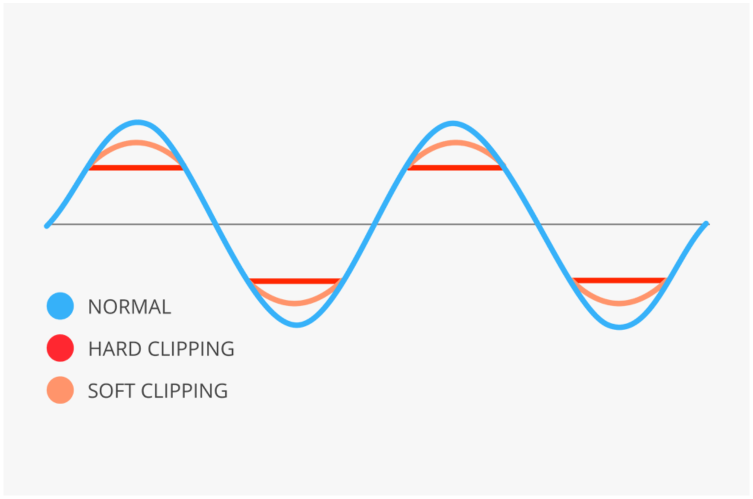
As I mentioned before, I needed to obtain a clean signal to process before I started my design. I used the XADC port of the Basys3 to convert the analog audio signals into digital signals. I used the XADC Wizard of Vivado to use the ADC of the Basys3 FPGA board. I also created a new module named CleanMod to use XADC wizard. In this module, I took the first 8-bits after the most significant 2-bits of the 16-bit output of the XADC because my DAC was supporting only 8-bits.



RTL Schematic of the Clean Module (CleanMod)

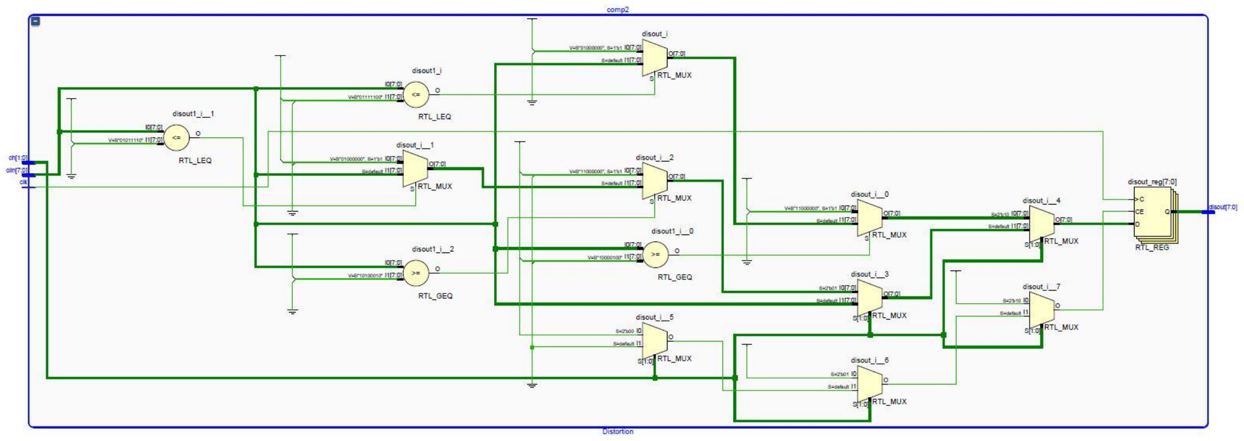
Distortion

Distortion is an effect where the signal is saturated on purpose to create a distorted sound that is similar to a square wave. My digital overdrive effect stimulates the same process by a method that is similar to hard clipping. In hard clipping the signal is clipped after a certain threshold to change the shape of the signal into a square-like wave.



Hard and Soft Clipping

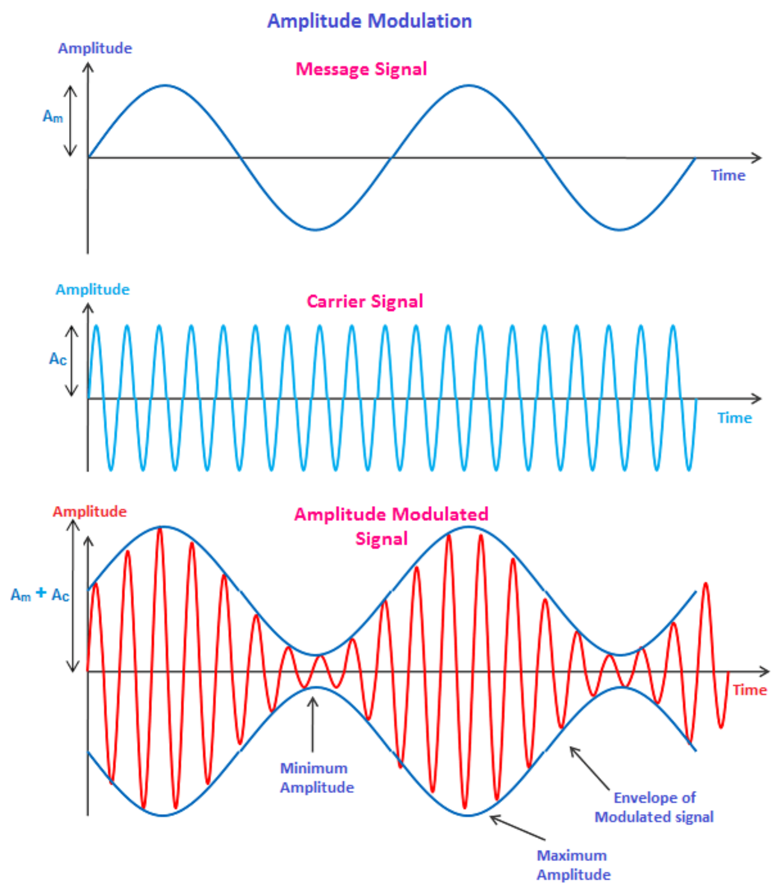
My approach to distortion was inspired by mostly hard clipping method. The only difference between my overdrive effect and hard clipping was the clipping part. Instead of clipping the signal after a certain threshold, I shifted the signal to a certain level after it reaches a certain threshold. By doing this, I maintained the amplitude of the signal on a higher level compare to the outcome of the hard-clipping method.



RTL Schematic of the Distortion Module

Tremolo

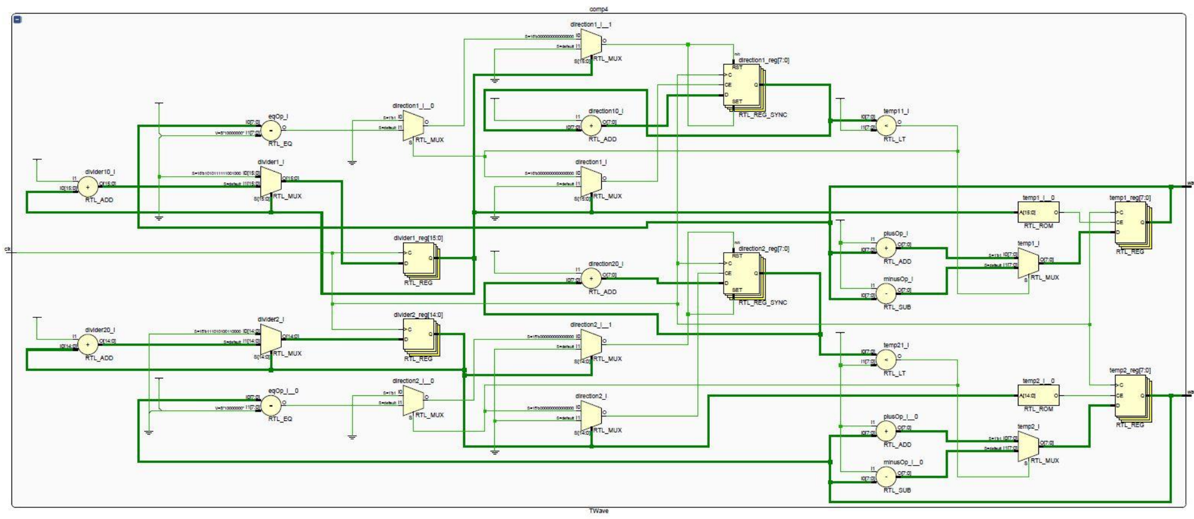
The tremolo effect is achieved by amplitude modulation which is exactly what I did for my tremolo effect. I created two different triangular envelope signals. The first envelope signal is a triangular wave with 8.68Hz frequency. The other envelope signal is the same triangular wave with a different frequency which is 13.02Hz. I multiplied my input signals with these envelopes.



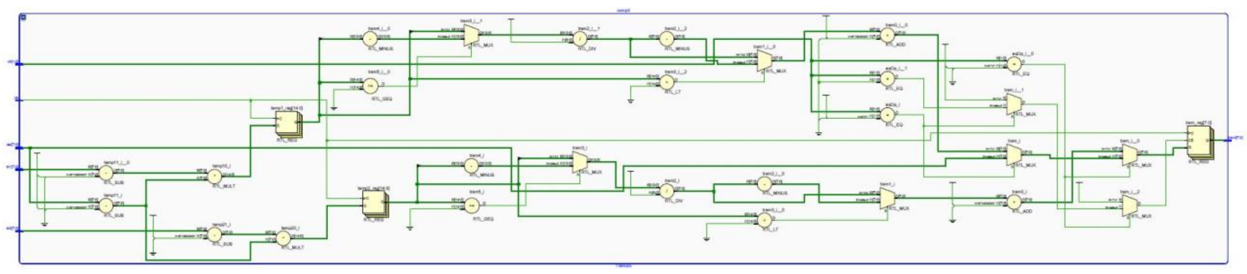
Amplitude Modulation

To multiply the signals, I needed to decrease the digital input signal by 128. The reason behind this is that I am working with unsigned 8-bit signals since the DAC cannot read signed binary, which means the lowest amplitude of the input signal is 0 and the highest amplitude is 255. To achieve amplitude modulation, I needed a signal that has an amplitude range from -128 to 127. By decreasing the amplitude of the signals, I achieved a proper amplitude modulated wave. I increased the signal by 128 again after the process was done.

I use two different modules for my Tremolo effect. The first one is called TWave and it creates two different triangular waves that will be used for modulation. The second module is called Tremolo. In this module, I multiply the envelope signals that I created in the TWave module with my input. The increase/decrease operations and the amplitude modulation take place in the Tremolo module.



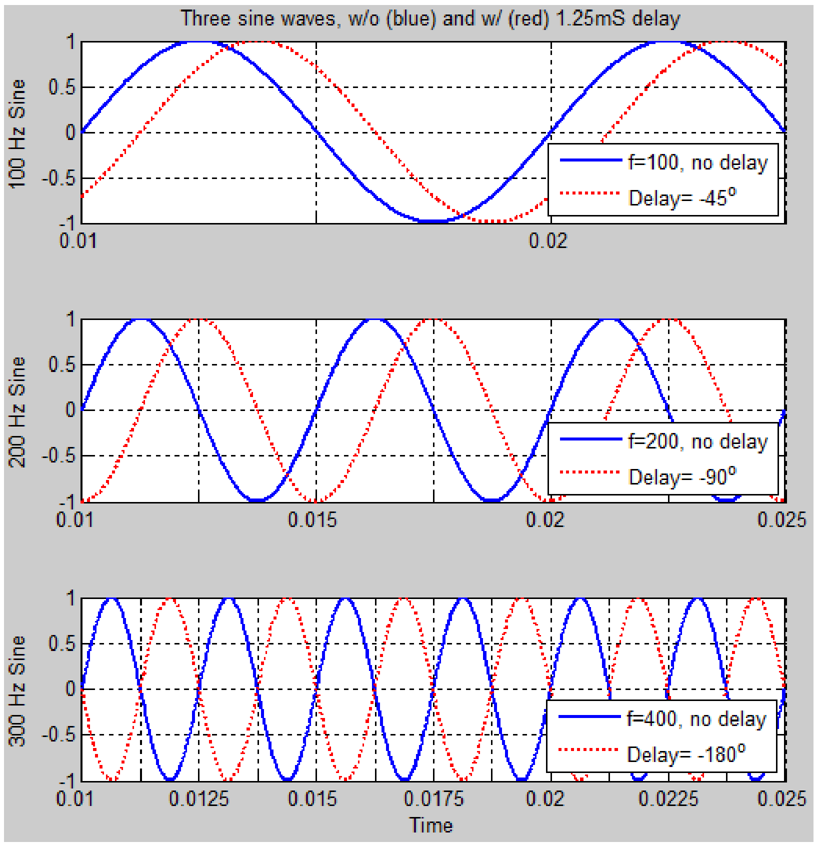
RTL Schematic of the TWave Module



RTL Schematic of the Tremolo Module

Delay

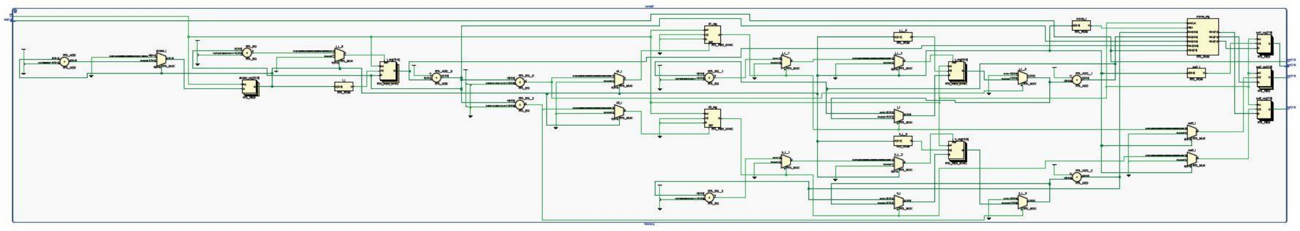
Delay is an effect that does what its name suggests, it delays the signal. It sounds simple when it is defined like this, but it is a complicated guitar effect. There are a lot of different delay effects that could be used in very different types of music. My aim for this effect is to implement a dotted-8 delay. I delayed the signal two times with 0.4-seconds and 0.8-seconds delay. I divided the 0.4-second delayed signal by 4 and the 0.8-second delayed signal by 8 to create an echo-like effect.



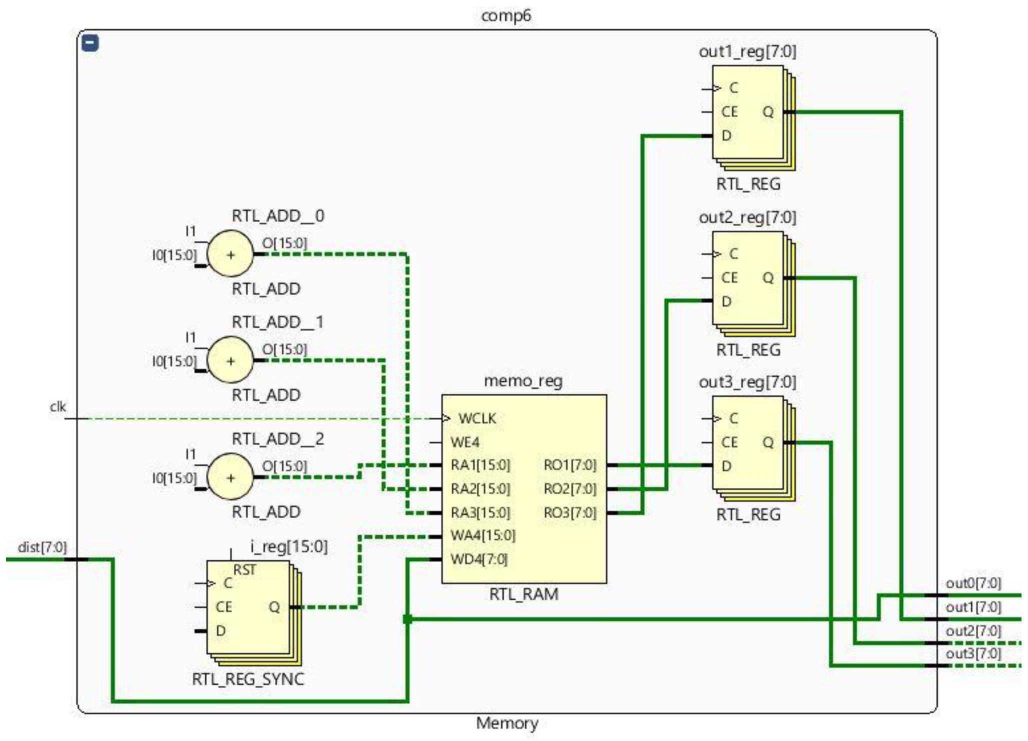
Time Delay of a Sinusoidal Wave

Another reason of this division is to avoid saturation due to overlapping. The delayed signals are added to the original signal to get a 2-times delayed signal. Since we are adding three signals, they overlap and create an effect where every note played from the guitar is heard 3 times. This overlapping can cause saturation and unwanted distortion in the sound. By dividing the signals, I decreased their amplitudes and avoided this saturation.

For my version of the dotted-8 delay, I created two modules. The first module is named Memory. In this module, I take samples from my input signal with 40kHz sample rate and I keep the samples in an array that keeps 35000 8-bit samples. I planned to keep a whole second with an array, but the Basys3 block ram capacity was not enough to keep 40000 samples. In this module, I have 3 outputs that are used for the delay effect. The first one is the original input signal of the module. The second signal is 0.4-seconds delayed version of the original signal and the third one is 0.8-seconds delayed version of the original signal. The amplitudes of these signals are the same with the original signal.

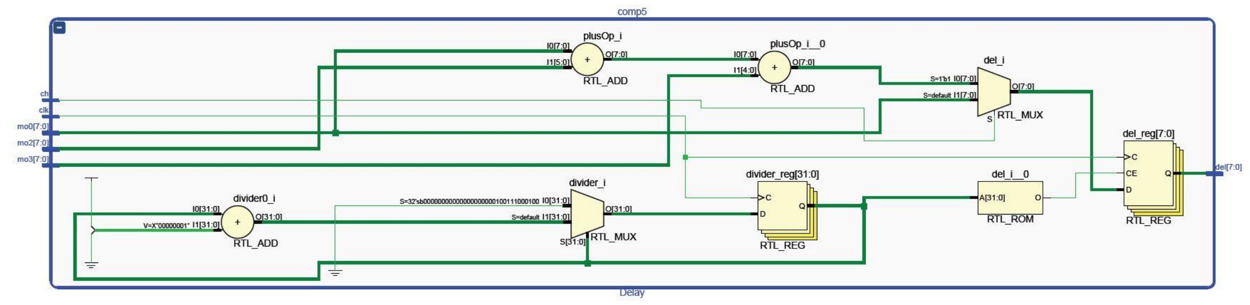


RTL Schematic of the Memory Module



The RAM Block of the Memory Module

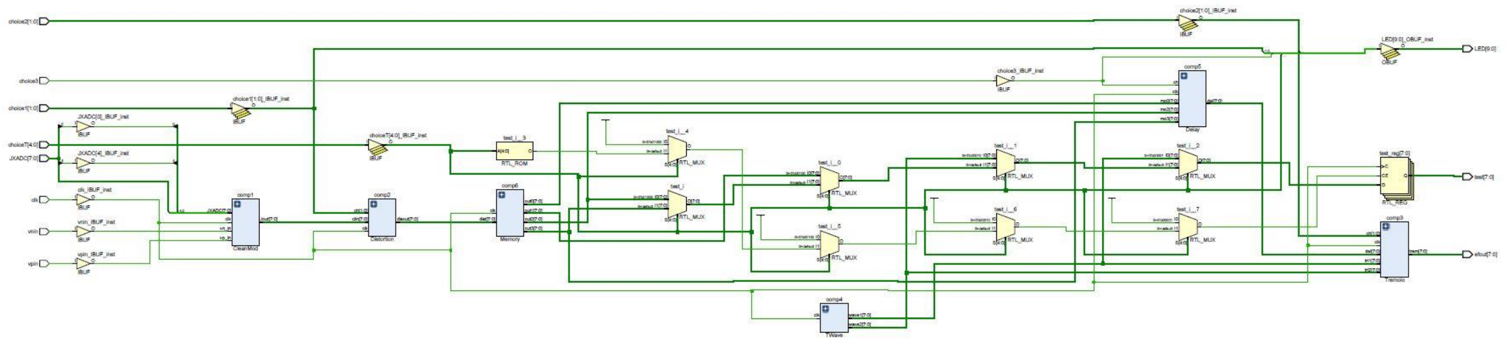
The second module was named Delay. In this module, I divide the signals and add them to create the effect. The reason I created a separate module for this process is to avoid confusion between the sampling process and the addition process.



RTL Schematic of the Delay Module

Top Module

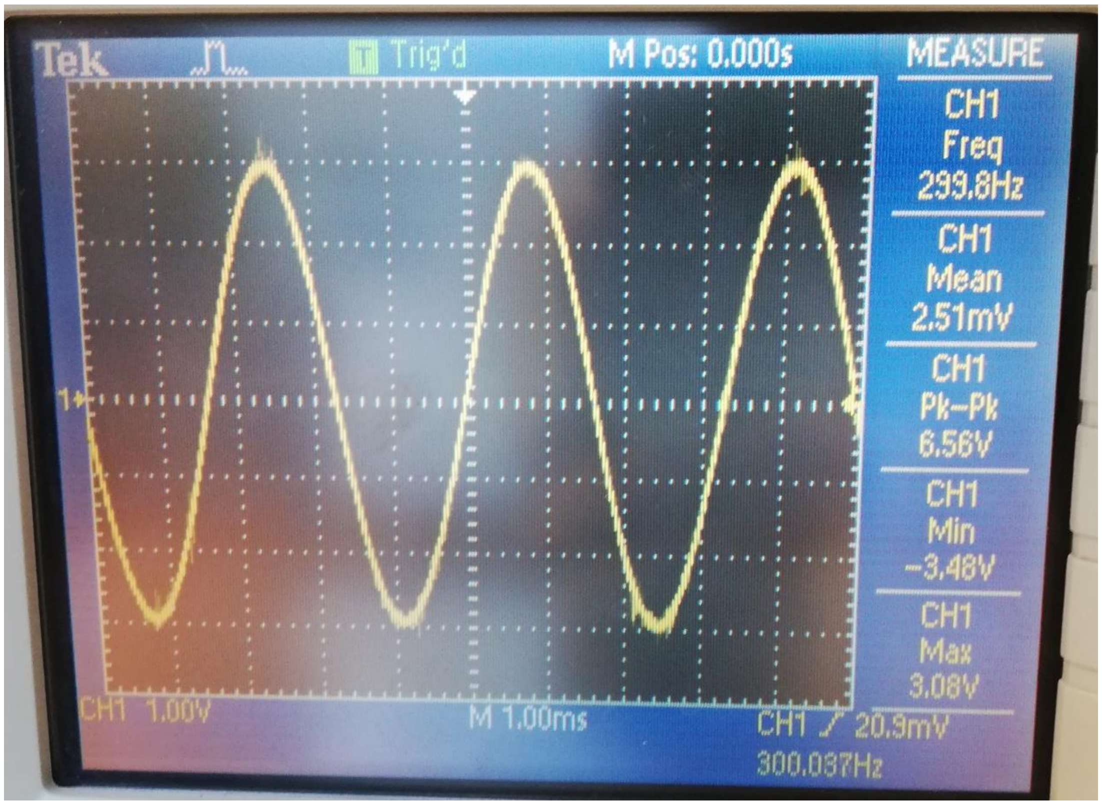
This module is created to connect all of the other modules. In addition, I have 5 different test outputs in this module to test the 2 triangular envelopes I created in the TWave module and the 2 delayed signals of the Memory module with the original signal.



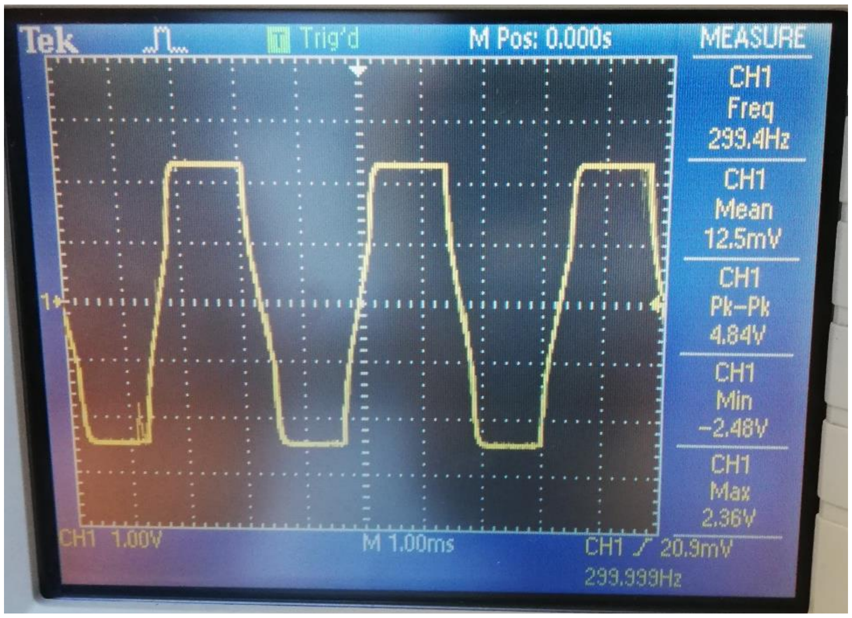
RTL Schematic of the Entire Code (TopMod)

Results

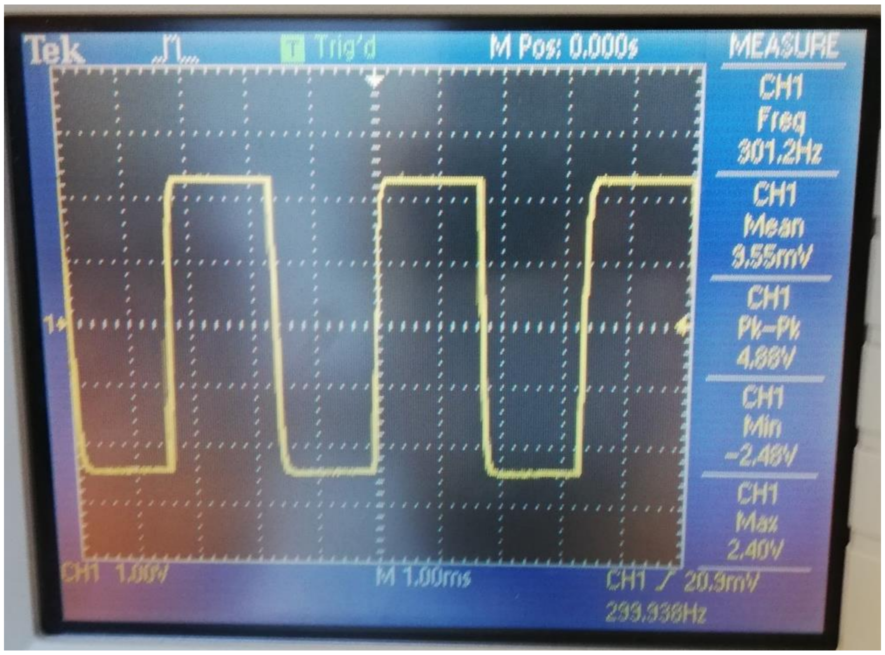
The results were better than expected. All of the effects worked like they were planned at the beginning with just some minor errors. The pictures below are showing how the effects work with a 300Hz, 400mV sinusoidal signal when +/- 10Vs are applied to the DAC and the OPAMP as reference voltages. There are no photos included for the Delay effect because the overlapped signals do not appear as clear as a graph to analyze. To see the results with a guitar and a sound, you can see the demo video on YouTube (video link at the top of the report).



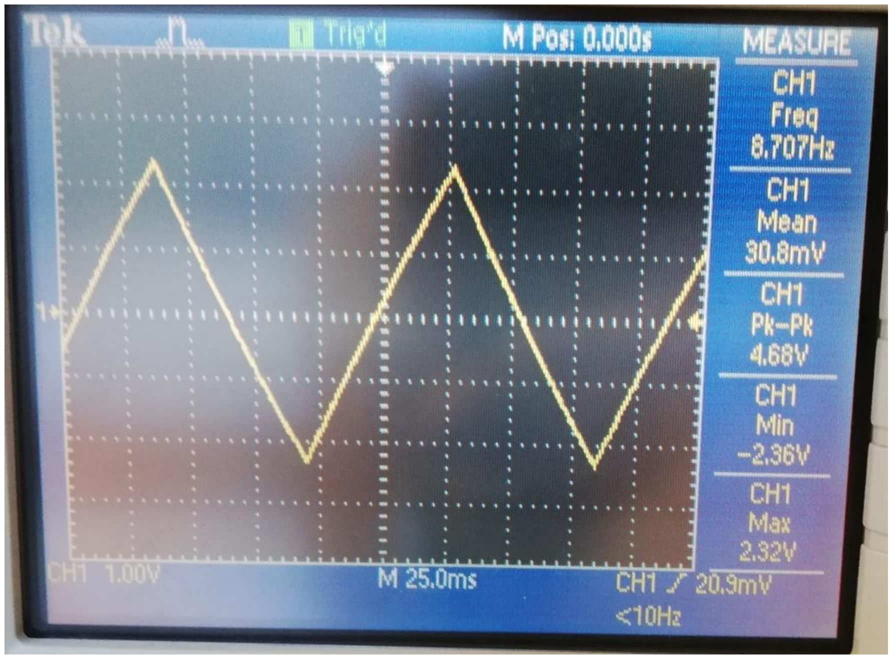
The Unprocessed Signal from the Output of the DAC Circuit



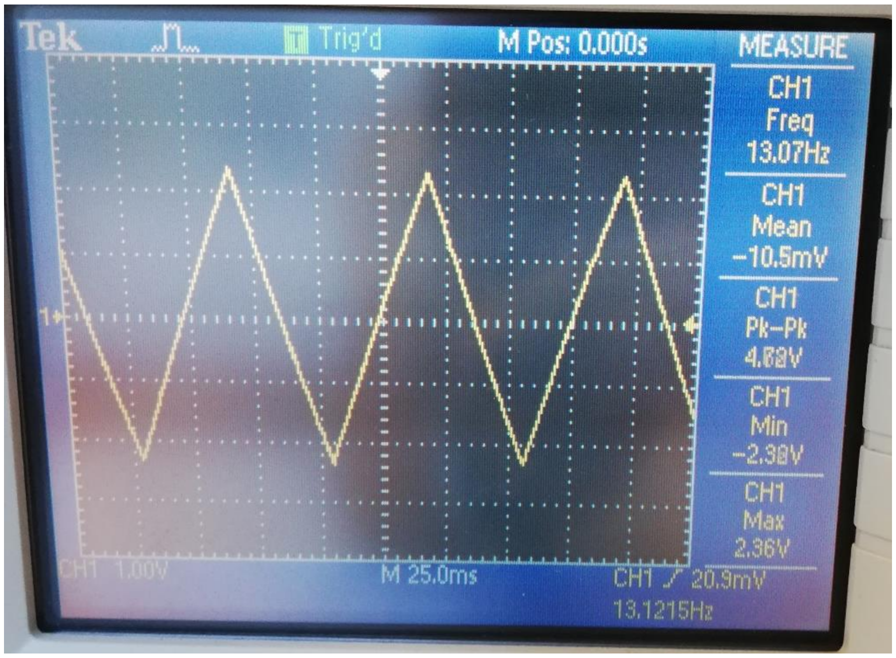
Weak-Overdrive Effect On



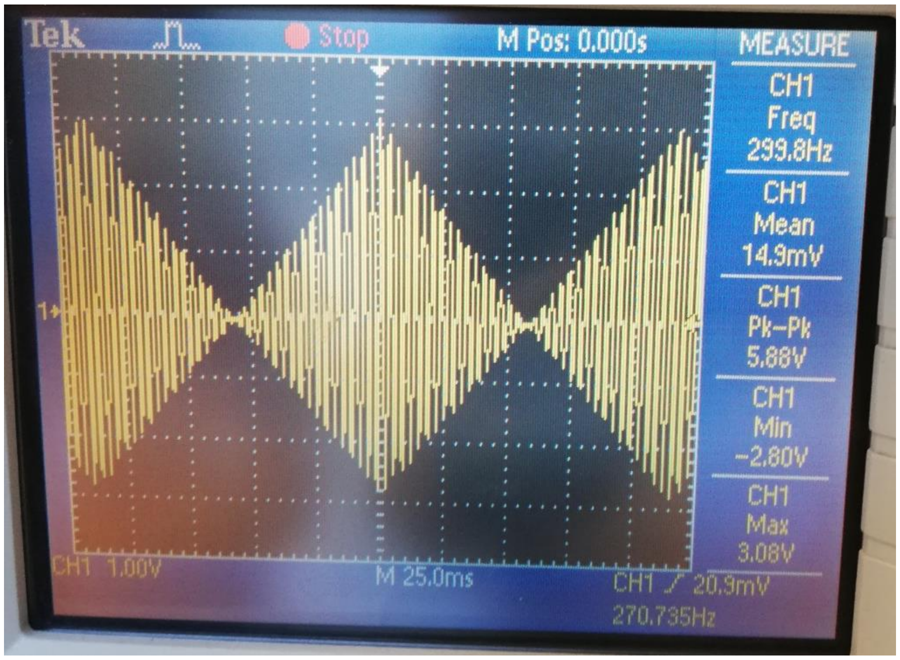
Strong-Overdrive Effect On



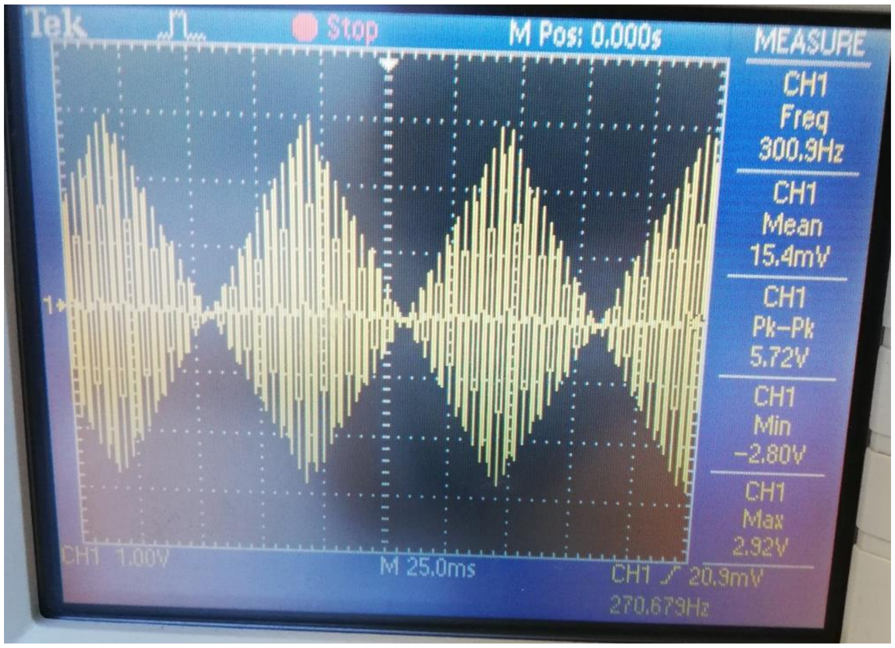
8.68 Hz Triangular Envelope



13.02 Hz Triangular Envelope



Amplitude-Modulated Sinusoidal (8.68Hz)



Amplitude-Modulated Sinusoidal (13.02Hz)

Conclusion

When I finished the project, I realized that I learned a lot about digital audio processing while doing the project. I used a RAM block for my delay. I used registers and flip-flops, registers, multiplexers, ROMs and many other components for different purposes such as storing a signal, choosing between modules, combining different signals and many other operations. Additionally, I worked with 8-bit signed and unsigned signals. I did a lot of operations on these 8-bit signals which taught me a lot about binary logic.

Appendix

References

B.O.S.S. Corporation, “BCB-60 | Pedal Board,” BOSS. [Online]. Available: https://www.boss.info/global/products/bcb-60/. [Accessed: 19-May-2019].

“Hard-clipping-vs-soft-clipping-diagram,” Pro Sound Formula. [Online]. Available: http://prosoundformula.com/hard-clipping-vs-soft-clipping-diagram/. [Accessed: 19-May-2019].

A. Shaik, “Amplitude Modulation,” Physics and RadioElectronics. [Online]. Available: https://www.physics-and-radio-electronics.com/blog/amplitude-modulation/. [Accessed: 19-May-2019].

E. Cheever and Swarthmore College, “The Time Delay,” Untitled 2. [Online]. Available: https://lpsa.swarthmore.edu/BackGround/TimeDelay/TimeDelay.html. [Accessed: 19-May-2019].

DAC0800 Datasheet, http://www.ti.com/lit/ds/symlink/dac0800.pdf

LM741 Datasheet, http://www.ti.com/lit/ds/symlink/lm741.pdf