Digital Guitar Amplifier

Final project design document

PHYS 434

Milo Sisk

Nolan Fujiwara

1. Introduction

a) Purpose

This design document describes the design and function of a simple digital guitar amplifier.

b) Scope

This device takes the low-voltage input signal of an electric guitar and amplifies it so that the guitar can be played and heard through the computer speakers in real-time. It will also include effects such as distortion and equalizer (bass/mid/treble) settings.

2) System Overview

An electric guitar pickup contains a magnet which generates a small voltage signal when a string is plucked [1]. This signal is initially too small to drive a speaker at an audible volume, so it must be amplified. While a conventional amplifier does this through analog circuits, we will simulate this process digitally by using LabView.

In a physical amplifier, it is often possible to amplify the signal to an amplitude higher than what the circuit can handle. In this case the sinusoidal signal becomes ‘clipped’, with the top and bottom chopped off. This limit turns an initial sine wave into a square one, this adds a distortion effect to the final sound. In LabView, we can emulate this by multiplying the initial signal by a large constant. Since LabView stores sound data within a range of values from -1 to 1 [2], anything which goes beyond this limit will be clipped, simulating what an actual amplifier would do.

When one or more strings are played, the guitar signal will contain a range of many frequencies dominated by the fundamental frequency of the notes, but also including other frequencies like harmonics, octaves, or small deviations from the fundamental. It may be desirable to adjust the volume of these frequencies to change the quality of the sound. This can be achieved through high and low-pass filters by which we can amplify specific frequencies.The LabView filter VIs we used separate out the desired frequency ranges by multiplying the high, low, or bandpass gain function by the Fourier transform of the signal, thus reducing frequencies beyond the cutoff while preserving others.

After the processing, a master volume control is necessary to allow the user to set the volume without affecting the sound quality, as the signal’s amplitude may become too high or too low in volume to hear comfortably after being amplified and passing through the filters.

3) System Architecture

a) Architectural Design

i) Input Configuration

This section uses LabView’s Sound Input Configure VI to set up the input device (in this case, the guitar) to be read from. This is where we specify parameters such as the sampling rate, buffer size, sample mode, and resolution. For real-time playback, continuous sample mode was used, and a relatively low buffer size (Samples/Ch) was used to minimize the delay in playback. We used the default value of 16 bits for the resolution and used 1 channel (stereo) because the guitar only outputs stereo signals.

ii) Output Configuration

This is where we used the Sound Output Configure VI to set up the computer speakers to play signals from the input device. We used the same parameters as for the input configuration.

iii) Signal Read

We used the Sound Input Read VI to generate a waveform from the device input. We placed it in a while loop to read continuously for any desired length of time.

iv) Gain

Before adding equalization effects, we amplified the initial waveform by a constant adjustable by the user. This both allows the signal to be played at a higher volume than what would ordinarily be possible through LabView, and when set very high (20 - 30x), it adds a distortion effect to the final sound typical of many physical amps.

v) Equalization

To fine-tune the signal, we use Butterworth filters to separate the signal into low, mid, and high frequency ranges. Guitars typically produce fundamental frequencies from 80Hz up to 1200 Hz, with harmonics and other frequencies as high as 3 or 4000 Hz [3], so we divided the signal into 80 - 250, 250 - 800, and 800 - 3000 Hz bands. Bandpass filters were used instead of low and highpass, in order to cut out any noise beyond the typical guitar frequency range.

After separating the bands, we multiply each band by a user-adjustable constant (Bass, Mid, Treble) to make the the respective frequencies louder or softer relative to each other, and then summed all the adjusted bands to give us our final processed output signal.

vi) Graphing

We fed both the initial ‘raw’ signal and the final processed signal to a waveform graph so it can be observed as the guitar is played.

vii) Playing the Processed Signal

We used the Sound Output Write VI to play the waveform signal through the speakers, and the Sound Output Set Volume VI to allow the user to set the volume, which is adjustable by the master volume knob.

viii) Clear Stored Information

We use the Sound Input Clear and Sound Output Clear VI to clear any stored data from the loop.

b) Design Rationale

We initially intended to build a physical circuit to amplify the signal, then send the signal to the DAQ device. However, when taking data on continuous sampling mode through the DAQ assistant, we immediately ran into a problem; not only was there a large delay in playback when using the DAQ assistant, but we consistently ran into an error regarding buffer size after a couple seconds of play; the DAQ was unable to keep up with the rapidly changing signal. We then discovered that LabView has the built-in capability to read sound data directly from a connected microphone, and decided to try connecting the guitar directly into the computer’s sound card, taking data with LabView’s sound palate VIs. Our current design follows this method, as well as including Butterworth filters to separate low, mid, and high frequencies. After many trials our program hasn’t run into any problems regarding buffer size when collecting and playing live, real-time data. We did run into some problems with this design, but compensated by adjusting values and constants over many trials we tested.

i) Sample Rate and Buffer Size

The first values we adjusted are the user inputs the Sample Rate and The Buffer Size (Number of Samples/Ch). We discovered that a higher buffer size gives higher quality sound, but results in a longer playback delay. By increasing the buffer size we can get a higher quality sound, but sacrifice realtime playback, as the playback delay becomes greater. A higher sampling frequency also decreases the delay time, but increases the processing power needed to get a good playback. The optimal values we found was a 5000 sample buffer and 44100 Hz sampling frequency; a buffer much higher than this and the delay becomes significant, and a buffer much lower results in a lot of pops and glitches in the sound. A frequency significantly higher than 44100 Hz causes the program to run quite slow and results in choppy audio. While the delay at these settings is large enough to make playing an actual song somewhat difficult, it is only a small fraction of a second and is sufficient for demonstration purposes.

ii) Equalization

Another example is using a bandpass filter ranging from 800 - 3000 Hz instead of a high pass filter. This was mainly done to suppress some of the noise coming from the high frequency range. Since frequencies higher than 1200 Hz are only subtly noticeable from a guitar, there was no need to keep frequencies much higher than this.

iii) Gain

Another value we tested a lot was the Gain value. In order to find distortion effects we adjusted the gain to high values, and found that the 30x - 50x range was suitable, limiting the value so that the volume won’t be too loud to listen to. The interface gain knob is set from 1 to 10.

Initially the Gain adjusting knob was after the equalization, however we decided to place the Gain before the equalization. This was due to the fact that the gain was causing some unwanted noise and popping sounds when it was done after the equalization. By placing it before the equalization, the filters were able to filter out most of the noise.

iv) Master Volume

The last value we adjusted was the Master Volume knob (0 - 10) which was included so that we can not only adjust the volume after equalization, but also lower the volume when the Gain was very high, as setting the gain to the high values needed for distortion increased the volume to an uncomfortable level.

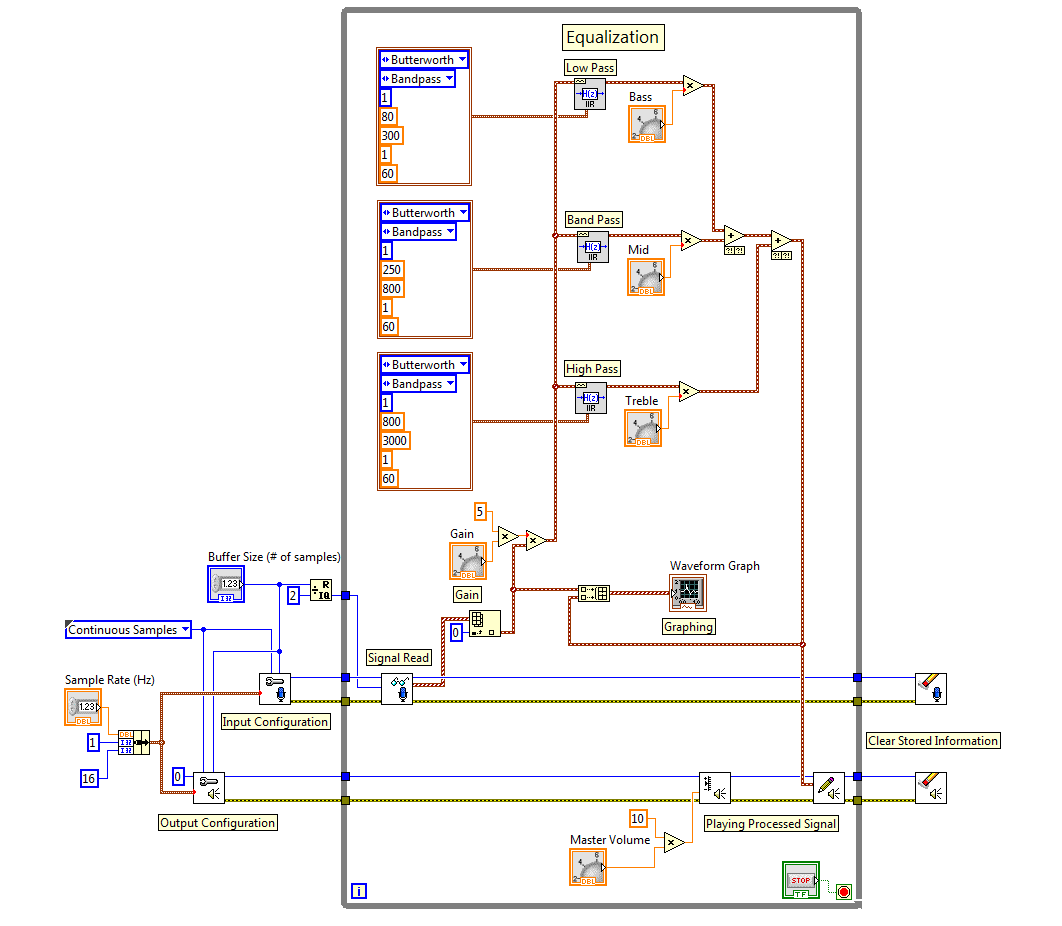
4) Human Interface Design

a) Overview of User Interface

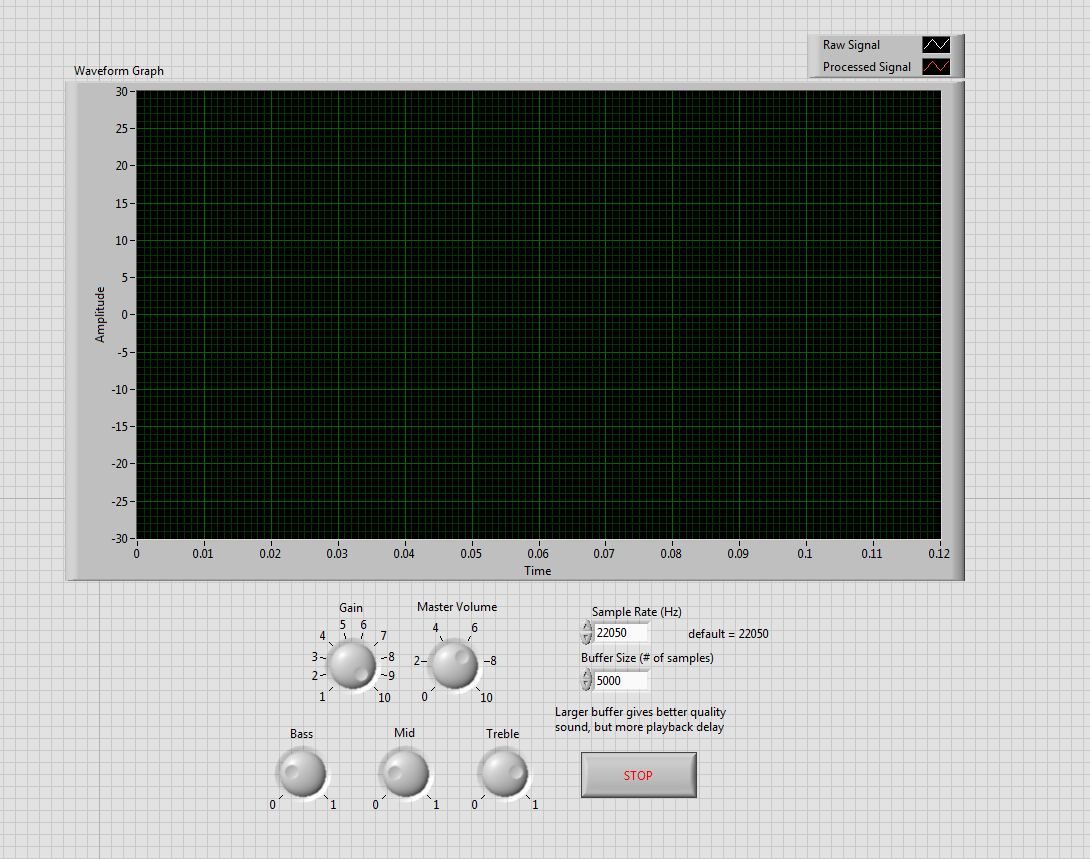
Using a 1/4” to 1/8” audio adapter, an electric guitar is hooked up to the computer’s microphone input. The program will play its output after amplification through a computer’s speakers or headphones. It will also graph both the “raw” input and “processed” output signals on a waveform graph. This graph serves no functional purpose but is rather interesting to look at.

The LabView interface includes control knobs for equalizer settings (Bass, Mid, and Treble), which allows the user to boost or suppress high, mid, or low frequencies by variable amounts. It also includes a master volume control, as well as a gain control which can be used to cause distortion when it is at high values, or a relatively clean sound at low values.

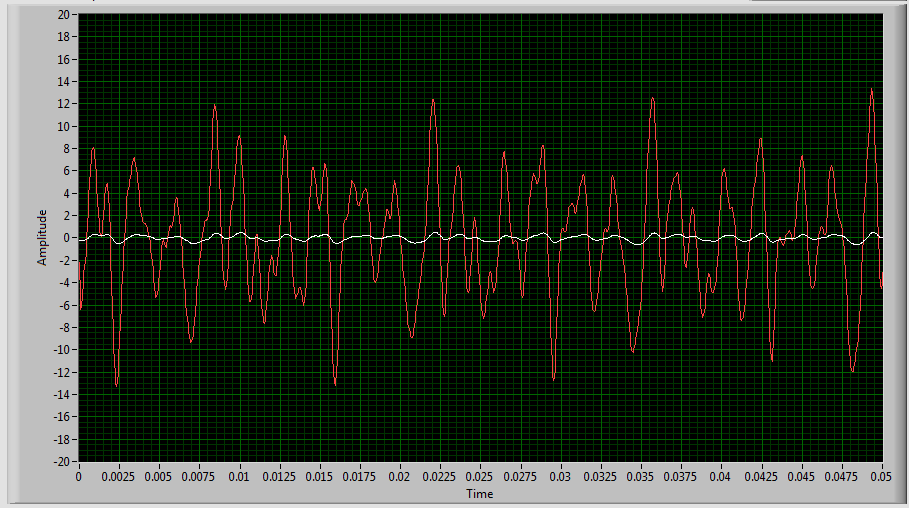
The interface also allows us to adjust the Sampling Rate and Buffer Size (Number of Samples/Channel). Unless the user is experiencing an unusually long playback delay, or you just want to experiment, it is not recommended to change these values, as changing them could lead to lower sound quality or increased playback delay.

b) Images

The block diagram.



The user interface.



Signal from the guitar.

Sources

[1] <http://entertainment.howstuffworks.com/electric-guitar1.htm>

[2] <http://zone.ni.com/reference/en-XX/help/371361H-01/lvconcepts/soundvis/#Sound_Data>

[3] <http://recordingology.com/in-the-studio/guitars/>