Verilog Guitar Multi-Effects Processor

6.111 Final Project Proposal | Haris Brkic | Fall 2017

Overview

First analog guitar pedals appeared in the 1930s. Shortly after being introduced to guitar players, they became an essential part of any guitarist's rig. The components used to make these pedals have changed over the years but the sound effects they produce have remained the same. It is quite expensive to make a versatile pedalboard since we would need a lot of analog pedals to complete it.

The goal of our project is to simulate the behavior of analog pedals using digital signal processing to create the digital equivalents of a large variety of pedals used by guitarists. We will digitize the analog signal from the guitar's magnetic pickups and then use the onboard ZBT memory, AC97, and various filters on the digitized signal to alter the digitized guitar signal to simulate the effects we want.

By doing so, we will create a significantly cheaper version of a multi-effects pedalboard that can be controlled from our labkit. Besides this, we will implement a graphical representation of the pedal board to see what pedals are currently being used and what parameters they are set to.

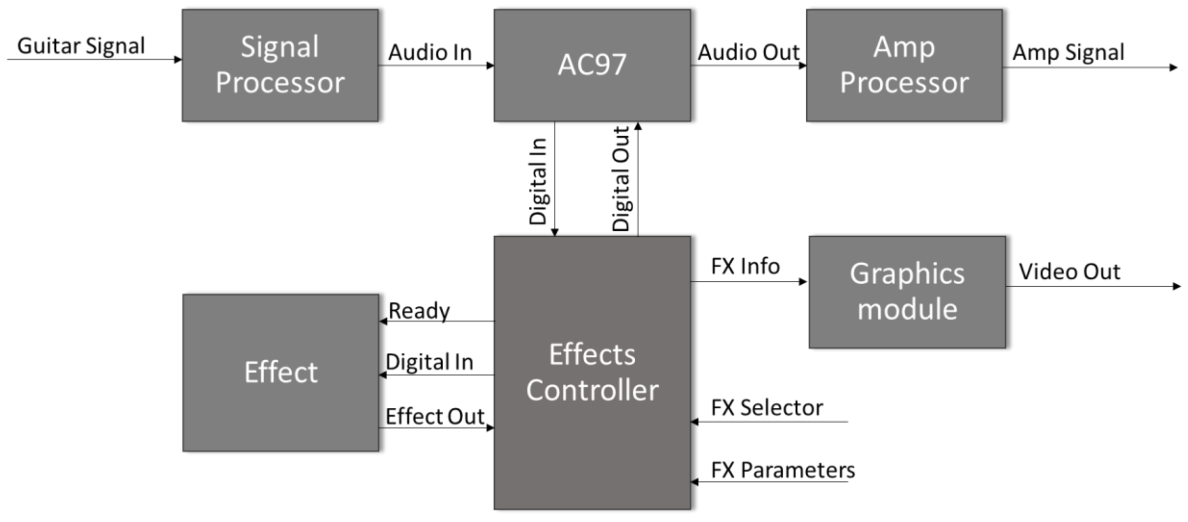

Figure 1: High-level Block Diagram

Design and Implementation

We will implement our digital multi-effects pedalboard by separating the project into a couple of modules described in the block diagram above. The *Signal Processor* module makes sure that the guitar signal is transformed into a signal readable by the AC97 chip. Similarly, *Amp Processor* transforms the output of the AC97 into a signal that the guitar amplifier can play back.

Our *AC97* module digitizes the guitar signal and outputs the digital signal to the *Effects Controller* which passes that signal to the appropriate effects. Our effect modules alter the digital signal and return it back to the *Effects Module* which from there ends up going into the guitar amplifier after it is properly processed by the AC97 chip and our *Amp Processor*.

The *Graphics module* takes in the effects used and the parameters they are set to, and outputs that to a monitor so we can see what effects are applied to the input guitar signal.

In our block diagram, the *Effect* module represents all of the different effects we are going to implement since all of their inputs and outputs are similar. In the following paragraphs, we will explain the behavior and our planned implementation for each of those effects.

Distortion results from „hard clipping“ the instrument's audio signal. By clipping the signal, the wave form of the input signal is distorted, and overtones are added. Harmonic overtones produce a „warm“ sound and inharmonic overtones contribute towards a more „gritty“ sound. „Hard clipping“ means that the input audio signal is re-shaped such that it has perfectly flattened peaks. To achieve this, we will process the input audio signal using the onboard AC97 chip and set the amplitude of the audio signal above and below a certain threshold equal to the threshold value.

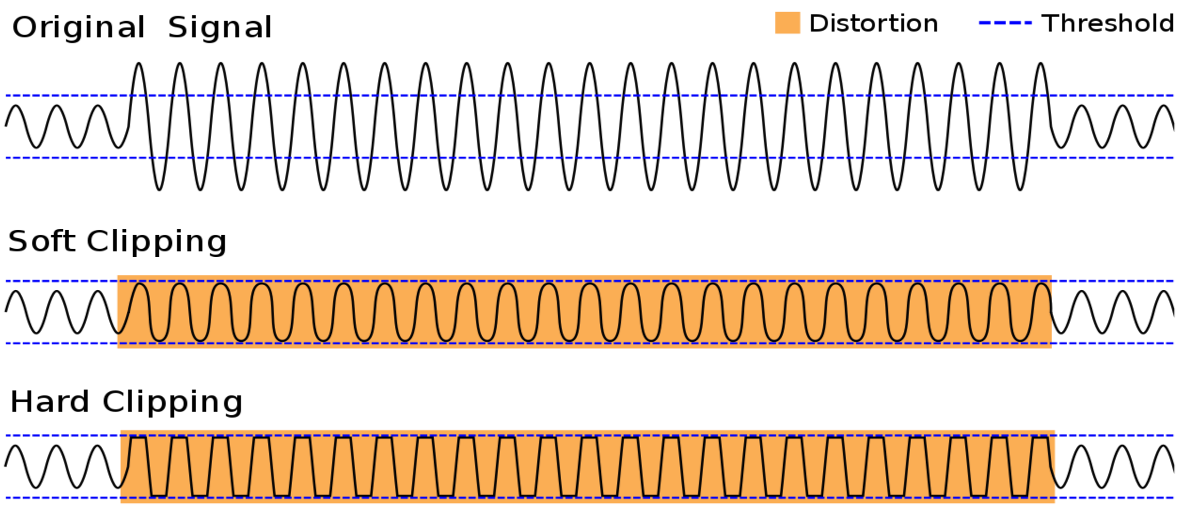

Figure 2: Comparison between a normal and distorted sound wave

Delay pedals produce an echo effect by reproducing an input audio signal with a slight time-delay. The two types of delay are „slap“ (single repetition) and echo (multiple repetitions). To implement the delay module, we need to store the input audio signal in the labkit's ZBT memory and play it back at a later time. In addition to this, echo is implemented using a feedback loop.

Reverb pedals simulate the spacious sounds produced in acoustic spaces such as halls and cathedrals. This effect is achieved by creating many gradually decaying echoes. To create this effect using digital logic, we will use digital signal processing to simulate the behavior of multiple feedback delay circuits. This is quite computationally expensive and requires the use of complex digital signal processing algorithms such as GVerb.

Chorus pedals produces a wide swelling sound by mixing sounds with small differences in sound quality and pitch. This is achieved by splitting up the audio signal in two, delaying and/or modulating one of the signals, and mixing the signals back together. This produces the sensation of hearing multiple guitars being played even though we are using a single guitar. To implement this we will use our delay module and create a „vibrato“ effect by introducing small but fast changes to the pitch using a frequency oscillator.

Wah-wah is an effect that sounds similar to a crying baby. This pedal is a moving bandpass filter whose frequency center is controlled by an external pedal. The filter will be written in MATLAB and we will use our labkit to control the center frequency. However, to properly use this effect we will need to control the center frequency with an external pedal because the center frequency should change quickly. One way we could deal with this constraint is to make the labkit change the center frequency automatically at a certain rate.

A *phaser* splits the audio signal and cyclically alters the phase of one of parts of the signals. After shifting one of the parts' phase, it mixes the two parts of the audio signal back together. The *phaser* pedal can be implemented using a cascaded 2nd order notch filter with a low-quality factor. We could make this filter in MATLAB and use our labkit to apply to the input audio signal.

Pitch shifters raise or lower notes played by a set interval by altering the frequency of the audio signal. Simple *pitch shifters* known as *octavers* raise or lower the note played by an octave or two. Complex *pitch shifters* allow a broad range of interval alterations. Our *pitch shifter* will be created using the onboard AC97 chip to alter the frequency of the input audio signal by the appropriate amount. We will use parameters that will determine the interval we are using to shift the sound wave.

Looper allows a performer to record and replay a phrase in a loop. In addition, it allows the performer to record multiple phrases or melodies on top of each other. The implementation of the *looper* pedal is similar to making a recorder. One issue that might arise is overflow because of multiple layers that are stored in the same memory address.

Testing

- *Signal Processor:*

  To test this module, we will us a set of recorded guitar inputs and check if the output of the module is consistent with the audio input. Furthermore, we will check if the output of the *Signal Processor* is compatible with the AC97 chip.

- *Amp processor:*

  We will test the *Amp Processor* similar to the way we test *Signal Processor*. We will simulate different outputs from the AC97 and check whether the output is consistent with the input signal and whether it can be used as input to a guitar amplifier.

- *Graphics module:*

  We can write a series of tests with different input parameters and effects and check if they are displayed properly on a screen.

- *Effects controller:*

  Effects controller uses the input digital signal, sends it to the proper effects, and mixes the output of all the effects. After that, it sends the mixed signal to the AC97 chip. We will test each of these features separately. We will write a group of test that test whether the input is passed to the proper effects and separate tests that check if the signals from the effect modules are mixed together properly.

- *Effect modules:*

  The tests for each of the effect modules will be similar. We will simulate several inputs and check whether the output is consistent with the desired effect described above.

Timeline

- **Week of November 6th:**
  - Implement the basic effect modules: *distortion* and *delay*
  - Write the *Signal Processor* and *Amp Processor* modules

- **Week of November 13th:**
  - Implement the *looper, chorus, wah-wah,* and the basic *Graphics Module*
  - Test if sound passes through the AC97 and is audible from the amplifier
  - Implement the more complex effects (*phaser, pitch shifter, reverb*)

- **Week of November 20th:**
  - Test and debug the *distortion, delay, looper,* and *chorus* modules
  - Write the *Effects Controller* module
  - Check the current *Graphics Module*

- **Week of November 27th:**
  - Test and debug the more complex effects
  - Implement a visually more appealing user interface

- **Week of December 4th:**
  - Finish testing and debugging all the modules

Expectations

- **The commitment:** Play the guitar through the FPGA without a significant drop in audio quality, working distortion and delay modules

- **The goal:** Implementation of the simpler effects, a graphical representation of the pedalboard, possible to use two effects at the same time

- **Stretch goal:** All the effects work properly, possible to use most of the effects at the same time without any issues, nice graphical representation of the pedalboard