FPGA-capella: A Real-Time Audio FX Unit

Cosma Kufa, Justin Xiao

November 4, 2015

1 Introduction

In live music performance, it is often desirable to apply effects to the source sound, such as delay and distortion. FPGA-capella is an FPGA-based audio FX unit that allows the performer to customize their sound by applying a chain of audio FX to the incoming sound and thus modifying it. Typically this is done with expensive outboard gear, but our project would allow for multiple effects, contained in one unit with flexible FX customization via controllable parameters. In addition to the audio component, a visual component will allow the user to both apply the FX customizations as well as visualize the outgoing audio.

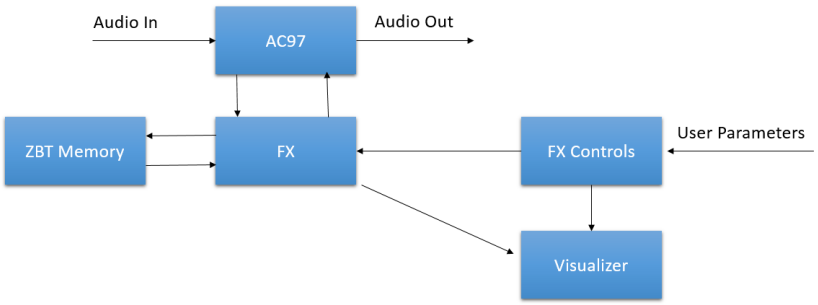
2 Overview

FPGA-capella can be represented by three major modules: the FX module, the FX controller module, and the visualizer module. The FX module is responsible for taking in audio and applying effects like delay and distortion to the audio. The FX controller module allows the user to determine the ordering of the effects, as well as control parameters of the FX, while the visualizer module is responsible for displaying the FX controls and a visualization of the audio. The FX module takes in audio data from the AC97 and outputs audio data with effects back to the AC97. It also takes input from the FX controller module and outputs to the visualizer module. The FX controller module takes input in the former of user selected parameters, and outputs to the FX module and visualizer module. The visualizer takes input from the FX and FX controller modules. Within the FX module are many modules that correspond to each of the FX, which are listed below.

- Delay

  Delay is a commonly used effect to give a sense of space, whereby an audio signal is processed such that a time-delayed, often times attenuated, copy of the original signal is added to the original signal. To do this, we can store the incoming audio in memory and play it some time after. One potential problem is overflow. For example, if an incoming eight-bit audio

Figure 1: High-level block diagram



signal has another 8-bit delayed signal added to it, the result may need nine bits to represent, resulting in a nonsensical output. To compensate, we must normalize the resulting audio signal such that the result remains eight bits.

- Echo

  Similar to delay, echo adds time-shifted copies of the original signal, except with a feedback loop such that multiple, decaying delays are played constantly after the original signal, much like a real echo.

- Filtering

  Much like in Lab 5, filtering works by attenuating frequencies of some signal. In the lab we implemented a low pass filter to block out high frequency noise, but filtering is often applied for artistic reasons, such as using the low pass filter for a muffled, behind-closed-doors sound.

- Distortion

  Distortion is an effect commonly used in guitar music, such as the canonical crunchy electric guitar sound. While there are many methods of achieving this effect, we decided the most reliable method would be bitcrushing, where the incoming audio is reduced in bit depth, thus quantizing the audio signal more. This has the effect of producing a distorted sound. We plan to implement this by replacing the lower bits of incoming audio data with zeros.

- Stereo Width/Panning

  Panning of a stereo signal involves sending differing amounts of an audio signal to the left and right speakers. This is often useful for spatialization of the sound, allowing for the listener to percieve sounds coming from

different directions. A related effect is that of stereo width, which, as it sounds, is how wide of a stereo field is perceived by the listener, as opposed to a mono signal, which has the same signal going through both speakers. It is possible to change stereo width by panning two copies of the signal to both speakers and delaying one of them, such that it sounds like they are coming in at different times.

- Looping

  Looping is simply recording and layering sounds; the performer records a sound and the recording will play on loop, similar to lab 5. However, once a new sound is recorded, that sound is played simultaneously with the old sound and looped.

- Pitch shifting

  While not entirely an effect in and of itself, pitch shifting is nevertheless very important for other effects. To do this, the incoming audio is Fourier transformed and the frequencies subsequently shifted. These frequencies are then inverse Fourier transformed and output as audio.

- Octavising

  Using the pitch shift module, it is possible to play the signal shifted an octave up or down in parallel with the signal, which is often used to thicken the sound.

- Chorus

  Chorus is intended to make the audio signal sound like there is a chorus of sounds (hence the name). To achieve this, pitch-shifted, time-delayed copies of the original signal are mixed with the original, where the pitch of the copies is often modulated with a low frequency oscillator.

- Reverberation

  One of our project's reach goals is to implement reverb, which is rather computationally expensive. Reverb gives the effect of space, since in a physical room a sound reflects off of the walls multiple time, creating a persistence of sound after the sound is produced. At a basic level, this is implemented by adding multiple short delays of decaying strength; however, it is difficult to algorithmically implement. One such implementation that seems promising is GVerb, which is an open source reverb algorithm.

3 Testing

- FX controls module:

  The FX controls module is like a state machine which takes in the desired user choices of audio effects and sends them to the FX module. Therefore, we will create a test jig with all the combinations of audio effect selection

and time parameters and will test based on the desired effects, if the outputted mode of the FX controller is correct. Additionally we will, have the FPGA display the output of the FX controls so the tester can see if the output from the module is the expected output.

- FX module:

  Just like the FX controls module, the FX itself is also a state machine, We will again create a test jig with the different possible combinations of selection and observe whether the correct state mode is reached.

- Delay:

  For testing we will have a test jig which simulates a single tone being played for a given single period. Then we have the delay module delay it by a given time and determine whether the result and the expected output are the same.

- Echo:

  For testing we will have a test jig which simulates a single tone being played for a given single period. Then we will run the single tone through the echo function. we will expect to see a delayed version of the function with its magnitude decreasing. Thus, we will compare the expected output with the output from the module.

- Filtering:

  For testing we need to observe the frequency response of the input and output to determine whether the desired filtered signals are removed. Thus we will have an input signal that has a non-zero magnitude at all frequencies as an input. Therefore the output's frequency response should be the same with the magnitude of the removed frequencies at 0. we will then compare the result of the function to the expected output.

- Distortion:

  For testing we will check that the higher order bits of a given input and the resultant output from the distortion module are equal and that the lower bits of the output are all zeros.

- Stereo width/Panning:

  For testing we will first to see the functionality of each speaker. we will have test jig for sound come out of one speaker, while the other speaker is silent. This will help in determining whether we are able to appropriate the right signal to the right speaker. After that we will create a test that will have the two speakers playing two different sound pieces to determine whether we able to play different sounds through the speaker.

- Looping:

For the looping module, we can test by writing a test jig that records different tones each time record is activated, and upon playback the recorded tones should be overlapping as well as looping.

- Pitch Shifting:

  For the Pitch Shifting module we can send a signal into the pitch shifting module. Then we can compare the fourier transforms of the input and ouput. The frequency response of the output should be same as the input just shifted over a bit.

- Octavising:

  For the Octavising module we can form a single tone sound. then we can pitch shift that tone. the octavising module's output given the input of the the single tone sound should be the equal to the sum of the single tone sound plus the pitch shift of teh tone.

- Chorus:

  For the Chorus module we can form copys of a single tone and have them delayed or pitch shifted. Then we can compare the output of the chorus module to the sum of the copies plus the original signal.

- Reverbation:

  For the Reverbation module, we will create a recording that will last a certain period. then we will create copies of the recording shifted, with its magnitude decaying at a certain rate. we wil then test the reverbation module to see if the output is equivalent to the sum of the copies and the original recording.

4 Timeline

Week of November 1st:

- Write a detailed description of each module in the system, including the inputs and outputs for a module and its functionality
- Test Ac-97 Chip to determine the minimum number of bits of information needed for a high quality sound (16 bit audio)
- Implement the following basic effects modules: delay, echo, distortion, looping, filtering

Week of November 8th:

- Test the basic effects modules for functionality
- Implement a basic block visual for the user interface
- Implement pitch-shifting

Week of November 15th and November 22nd:

- Implement Controls module
- Finish Testing on the basic effects modules with controls module
- Implement the following advanced audio effects modules: chorus, reverb, flange, phaser, octavising
- Implement mouse module to ease user face interaction
- Implement a basic audio visualization module to display an abstract representation of the sound

Week of Novemver 29th:

- Finish testing and debugging all of the effect modules
- Implement a more appealing,cleaner, and easier to use user interface and audio visualizer

5 Expectations

- **Minimal product:** Live audio FX (delay/echo, filtering, looping, distortion), basic controls/basic block visualization
- **Goal:** Implement most of the FX, have an audio visualizer, full control of FX
- **Reach:** All FX, cool visualizer