OMNIBOX
A Multi-Featured Audio Effects Module

6.111 Final Project Proposal
Devon Rosner
Dylan Sherry
11/03/2011

Abstract

For our final project, we have envisioned Omnibox, an audio effects suite which supports several essential standard effects found on commercial pedal boxes, including some more complex and uncommon features. We expect that creating a solid underlying infrastructure will decrease the overhead for added functionality, allowing us to provide a wide range of features. The standard effects which we will support are delay, equalization, tremolo, reverb, chorus, ring modulation ("Moogerfooger"), wah, auto wah, and distortion. Due to the power and versatility of the 6.111 lab kit, we expect no need for downsampling, and intend to maintain a high quality audio stream.

Overview

Omnibox is a multi-feature audio effects module enabling realtime digital effects on streaming audio. Using the module is simple: users connect an audio signal to the labkit's mic port and receive the modified signal via the labkit's headphone port. Users can interact with and alter the effects via a simple menu implemented with the labkit's alphanumeric display panel, buttons, and switches. When packaged as individual components, each of the effects supported by Omnibox retails for hundreds of dollars; Omnibox is intended to provide high quality audio effects at a low price.

Omnibox is partitioned into three overarching packages: the AC'97 codec (implemented in lab 5), the control package, and the effects stream. The control module provides an interface between the labkit's buttons, switches, alphanumeric display, and the parameters that control each effect. The effects stream consists of a string of effects modules, where data streams through the modules in a predefined order. Streaming yields more space in the labkit's on-board memory (BRAM), which means there's no need for downsampling the audio. This allows the Omnibox to maintain a high quality sampling rate around 41kHz. Additionally, for each audio sample received from the AC'97 codec, Omnibox has several hundred clock cycles to compute an output sample in time for the next input/output exchange, which grants our design further flexibility.

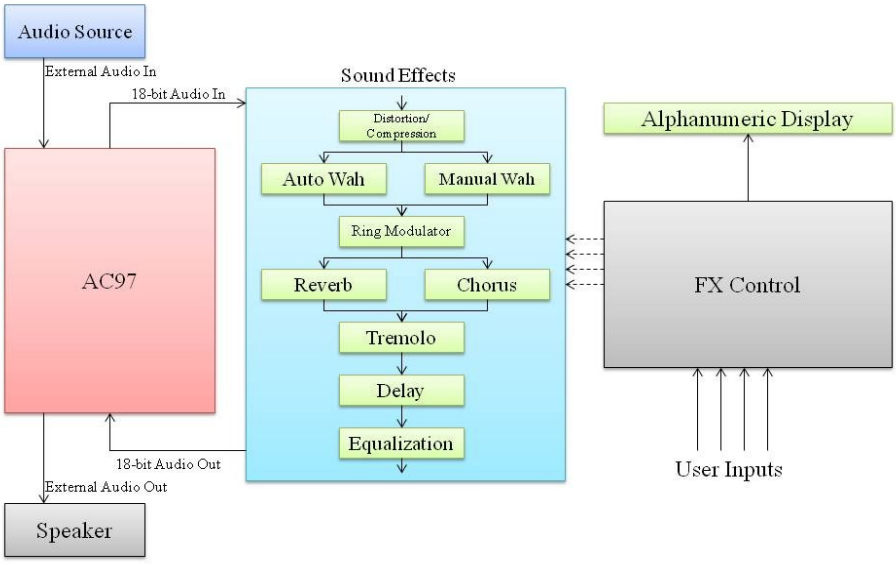

Figure 1: Omnibox block diagram

Modules

Long Delay (Dylan)

Description: Long delay records uniform time blocks (length of approximately 200 ms to 4s) of the incoming audio signal to BRAM and repeats them with continuously decaying magnitude. The incoming audio signal will still be played over the repeated time blocks, and will be recorded to the BRAM for use as fresh delayed samples as the old delay samples fade.

The user will be able to specify how long the uniform time blocks are, how loud the first repetition of a recorded time block is, relative to the original time block, and how long the time blocks take to fade away.

Inputs: clock, 18-bit audio in, initial feedback magnitude, decay value, delay time
Outputs: 18-bit audio out
Memory Usage: 44100 hz * 4 sec (max delay time) * 18-bit audio in / 3 (downsampling) = 1.06 Mbits

Equalizer (Devon)

Description: Equalizer will use a linear combination of a set of low-pass filters created via Finite Input Response (FIR) to produce five frequency bands that span the typical human hearing range. Each of these bands will have its own user defined amplitude. This will give the user control over what frequencies are emphasized and the overall shape of the output tone.

Inputs: clock, 18-bit audio in, five amplitudes for the five frequency bands
Outputs: 18-bit audio out
Memory Usage: 5 * FIR coefficient storage in 5 x 64 BRAM = 1.6 kbits

Tremolo (Dylan)

Description: Tremolo takes the audio input and multiplies it by a periodic square wave. This gives the output a stuttering effect since the output will be zero for half of the period and the actual input signal for the other half.

The user will have control over the square wave's period. Other waveforms that resemble a periodic square wave will also be available to the user.

Inputs: clock, 18-bit audio in, tremolo waveform selector, tremolo waveform period
Outputs: 18-bit audio out

Reverb (Dylan)

Description: Reverb acts as a delay, but with uniform time blocks of shorter length (50 ms to 300 ms). This effect will support nonlinear fading, enabling various forms of decay such as spring reverb, hall reverb, and room reverb.

The user will have control over how long the uniform time blocks are, how fast they decay, and what waveform (sinusoidal, exponential, linear, etc.) will be used to decay the signal.

Inputs: clock, 18-bit audio in, delay time, decay value, decay waveform selector, decay waveform period
Outputs: 18-bit audio out
Memory Usage: 44100 hz * 0.3 (max delay time) * 18-bit audio in = 238 kbits
Function storage in 5 x 128 BRAM * 4 functions = 2.56 kbits

Chorus (Dylan and Devon)

Description: The chorus effect attempts to make one instrument sound like multiple instruments are playing the same audio sample. To accomplish this, recent audio will be partitioned into uniform time blocks, with block lengths similar to those for reverb. These recent blocks will be repeated, but played back at a faster or slower speed than the original signal. This will give the illusion that multiple instruments are playing, as two musicians are never exactly in tune or tempo (phase) with each other.

The user will have control over the uniform time block length, block decay rate, and the degree of playback speed shift.

Inputs: clock, 18-bit audio in, delay value, decay value, frequency offset value
Outputs: 18-bit audio out
Memory Usage: 44100 hz * 0.3 (max delay time) * 18-bit audio in = 238 kbits

Ring Modulator (Dylan)

Description: Ring modulation is an audio effect that multiplies incoming audio by another signal. Typical signals used are sine waves, square waves, saw-tooth waves, and triangle waves. The user will be able to select what waveform to use, and the period of the waveform.

Inputs: clock, 18-bit audio in, waveform selector, waveform period
Outputs: 18-bit audio out
Memory Usage: Function storage in 5 x 128 BRAM * 4 functions = 2.56 kbits

Wah/Auto Wah (Devon)

Description: The wah effect applies a bandpass filter to the input audio signal. The band pass filter's frequency band can be shifted by a set period or, more typically, by an external source such as a potentiometer. The desired effect is a 'wah' sound applied to the audio, which is achieved with a fast shifting bandpass filter. This effect will have two modes. The auto wah mode will have a volume-triggered periodic bandpass filter that will only shift frequency when the input volume exceeds a certain threshold. Manual (or traditional) wah mode allows an input signal to shift the bandpass filter. That frequency shift control signal will come from a potentiometer, which will be connected to the labkit via an A/D converter.

The user will have control over the wah mode, auto wah volume threshold, bandpass frequency width, auto wah period, and manual control over the bandpass filter via potentiometer in manual mode.

Inputs: clock, 18-bit audio in, auto wah/manual wah selector, threshold value, bandpass width, manual center frequency shifter, auto wah period
Outputs: 18-bit audio out
Manual Usage: 12 * FIR coefficient storage in 5 x 64 BRAM = 3.84 kbits

Distortion/Compression (Devon)

Description: Distortion/compression clips the incoming audio signal to a set value whenever the amplitude reaches or surpasses the user defined threshold parameter. The audio is then multiplied by a user defined gain. This is known as distortion when the gain is very high and causes the output to sound fuzzy. When the gain and threshold are low, this effect becomes known as compression because the incoming signal's amplitude variation is constrained.

Inputs: clock, 18-bit audio in, threshold, gain
Outputs: 18-bit audio out

FX Control (Devon and Dylan)

Description: The FX controller will generate all necessary effects control signals. Using the alphanumeric display and debounced labkit directional buttons, the user will be able to scroll through a menu of effects and alter specific settings as desired. The controller will debounce labkit inputs. To provide realtime input signals to select effects, handle any external potentiometers.

Inputs: clock, all relevant debounced buttons and switches
Outputs: Alphanumerical display, and all effects control inputs, except for clock and audio in.

Module Layout

The typical music effects layout has been fairly standardized, so the effects will be laid out in a minimally changeable fashion. From audio in to audio out, our effects will be as follows: distortion/compression, wah/auto wah, ring modulator, either reverb or chorus, tremolo, long delay, and equalizer. Both reverb and chorus use small uniform time blocks as delay, so only one will be on at a time. The reverb/chorus block will also be interchangeable with tremolo to allow hard cutoff tremolo and soft cutoff tremolo.

Testing

First, we will generate some test signals in MATLAB and simulate the response of the procedures we plan to implement for each effect. Our test signals will include an impulse, square wave, and a sine wave, among others. Running these MATLAB simulations will help us clearly articulate the algorithm implemented by each effect, and should prove immediately useful in recognizing any potential shortcomings or procedural errors. Next, we'll simulate the responses obtained from combining several or all effects, in the order specified by our block diagram. We'll save the responses of each module or combination of modules for later use.

As each effects module is implemented in Verilog, we will run the original test signals through each module using ModelSim. Then, we will take the recorded response of each module and analytically compare it to the response obtained from its MATLAB simulation. This will allow us to verify that each module is behaving as we had intended, and that each produces the correct effect. We will connect all of our effects modules in the same combinations as in our MATLAB simulation, record the responses, and compare them to the MATLAB simulation results. Such a comparison can be used to identify and isolate any possible incompatibilities that may manifest themselves upon connecting modules.

To test the control module, we'll simply generate a ModelSim test jig that touches on every part of the

user interface, and compares the output control parameters with what is expected at each stage. The control module functions independently form the audio stream, which greatly simplifies our test jig

The final step in our test process will be to write a comprehensive top-level ModelSim test. The test jig will simulate both the user input and the audio stream, and will verify the display output and internal control values. It will also have the capability to record audio output, which we will quantitatively compare with the results from previous tests.

For each stage of the testing process involving audio, we will listen to a sample of each test output to ensure that it achieves the appropriate aural effect.

Schedule

Week of 10/31:

- Meet with Jon Losh about our project
- Write proposal
- Create fundamental blocks (BRAM with FIR coefficients, discrete sinusoids, square wave, triangle wave, sawtooth wave, etc., simple division)
- Purchase potentiometer for manual wah

Week of 11/07:

- Initialize all modules
- Finish basic implementations of effects such as distortion, long delay, reverb, tremolo, ring modulation, equalization, ring modulation
- Attempt basic wah feature

Week of 11/14:

- Fine tune distortion, long delay, reverb, tremolo, ring modulation, equalization, ring modulation
- If potentiometer has arrived, implement manual wah with A/D converter
- Implement auto wah
- Implement chorus using reverb module structure

Week of 11/21:

- Fine tune and debug all effects
- Work on real-world instrument to FPGA
- Add additional functionality if time permits (more waves?)

Week of 11/28:

- Debug all effects
- Prepare presentation