

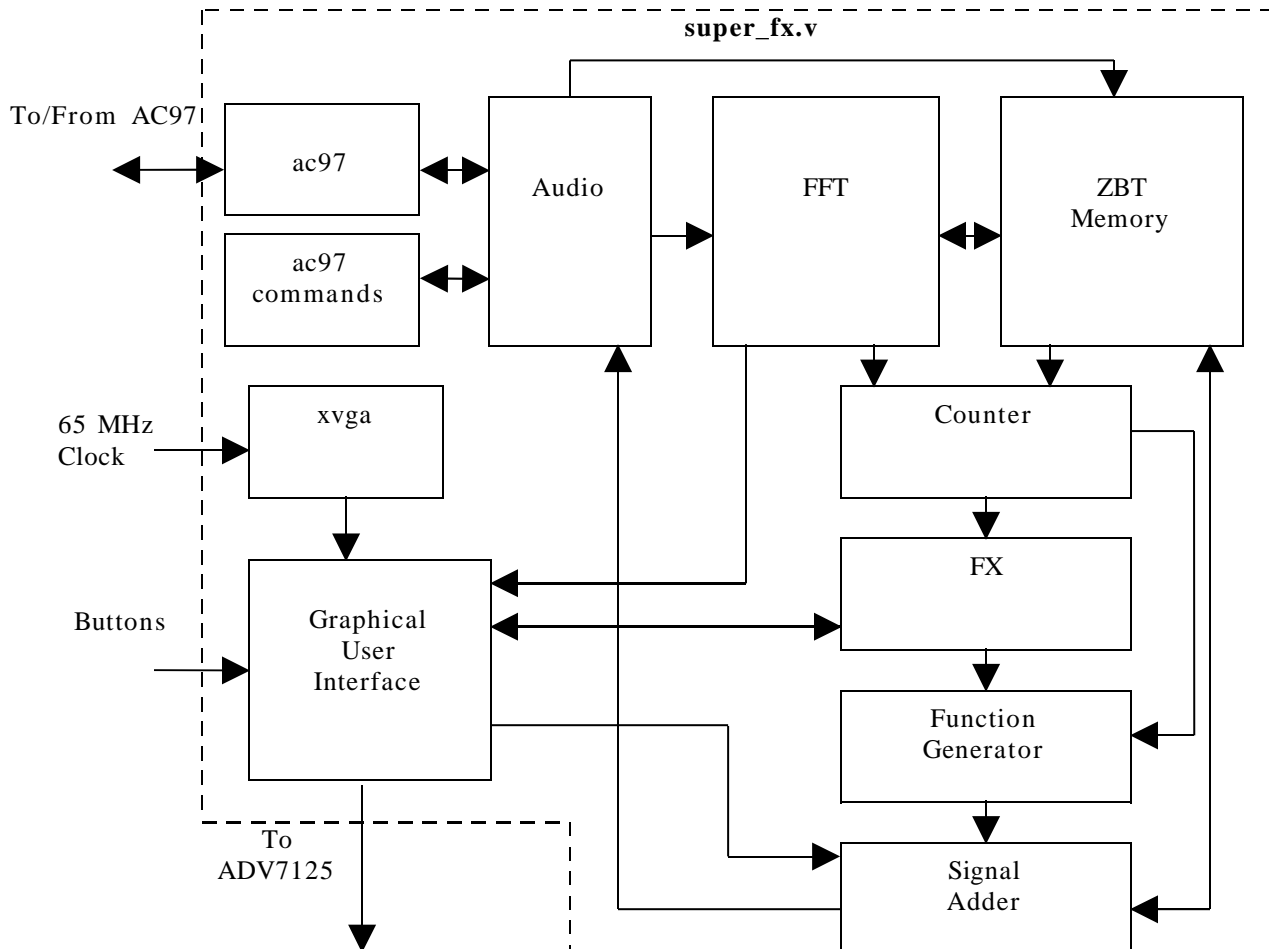
# Digital Frequency Domain Effects for Guitar

## 6.111 Final Project Proposal

Matthew Moskwa and Schuyler Senft-Grupp

We propose for our 6.111 final project a Digital Sound Effects Processor for guitar, or any other electronic instrument with a line-level signal. Traditionally, guitar/instrument effects were created with analog circuits and components, mainly in the form of discrete “pedals” which were chained together from source to amplifier to modify the output. This creates, among other things, noise, unwanted delay, and signal loss. By digitizing the effects and combining them into a single I/O form factor, we can mitigate the problems introduced in analog processing and create effects that are not possible using only analog component. Additionally, the user can easily switch between effects, set the levels of effects, and view the input and output waveforms through a series of buttons and a graphical user interface.

There is no clear project division, so we will split the work to try to equalize the difficulty and so that both of us can work as independently from the other as possible. Matt Moskwa’s main responsibility is the Fast Fourier Transform, which we believe is the hardest part of the project. He will also design the FX module which changes the signal. Schuyler Senft-Grupp will write the graphical user interface, the counter, the function generator, and the signal adder modules.



## **Fast Fourier Transform (FFT)**

The FFT takes in the current 18 bit sample from the audio module and the previous N samples from the memory. It then computes the Fourier transform and outputs the 18 loudest frequencies and their magnitudes. We chose 18 frequencies so that if all six guitar strings are played, the played note plus two harmonics will be identified by the FFT. If only one note is played, up to 17 harmonics will be identified. Each set of frequency and magnitude will take 20 bits. Fifteen bits are for the frequency, which allows us to store frequencies up to 32KHz (although realistically our cutoff will be around 20KHz.) The remaining 5 bits are for the magnitude.

We hope to have the FFT module running at around 100Mhz, which means we will have about 50 clock cycles worth of processing between the audio frames from the AC97 codec. This will leave enough room for any other effects we might want to add with FIR filters and output to the GUI. There are many possible implementation of a hardware FFT, and we will choose the one that allows us these speeds while keeping the quality mentioned above. If we must sacrifice anything, it would be speed over sound quality, but we will surmount this obstacle when it presents itself.

## **Counter**

The counter module keeps track of how long a particular module has been played. When the counter receives a frequency value from the FFT, it checks if that value is in the memory. If it is, the counter increments, otherwise it gets reset to zero. This value has two uses: one is to determine the phase of the outgoing signal, and the other is to allow us to do effects where the effect is dependent on the time from when the string was plucked. We still need to determine if there will be one instance of Counter, or if each of the 18 frequency outputs per cycle will have its own instance of Counter.

## **FX**

FX alters the incoming frequencies as defined by the user. We would like to have four effects available to the user: "Bend" which gradually increases the frequency of each incoming note; "Vibrato" which varies the frequency around the initial note; "The Arpeggiator" which plays the note's associated arpeggio; and "Harmonizer" which adds the associated harmonies to the initial note. The magnitude and speed of each effect will also be adjustable by the user.

At a minimum, we will have the vibrato effect working as a proof-of-concept. Once the FFT is successfully implemented, creating new

effects will be easier, as they will simply be manipulations of the FX module. If all effects are working, we will develop a demo program to show the effects working in series.

### **Function Generator**

The Function Generator outputs a point on a sine curve given a frequency and time since the frequency initially began. We will be generating a total of 18 frequencies, so there will be 18 of these blocks total. This will be all the data entering the AC97 if the FFT is active. In this respect, the original samples are destroyed and rebuilt from the FFT data.

### **Signal Adder**

The Signal Adder takes in all the generated functions and combines them into one 18-bit output to the audio module. It also receives a user defined delay period and reads from memory the corresponding previous 18-bit output from (t-delay) and combines that into the signal at a user defined magnitude.

### **Graphical User Interface**

The Graphical User Interface allows the user to view his effects selection and levels, and see how the effect changes the frequency spectrum. The top half of the screen displays the various effects with vertical slider bars to control each effect's levels. The left/right buttons on the labkit move between effects and the up/down buttons move the slider bars. On the bottom half of the screen are two spectrum analyzers. The left one shows the spectrum of the original signal and the right one shows the spectrum of the modified signal. The effect levels will be output to the FX and Signal Combiner modules, and the pixel color will be output to the ADV7125.

### **AC97, AC97 Commands, Audio, and XVGA**

These modules interface with the audio and video ADC/DACs. They are taken almost directly from labs 3 and 4 but with some minor changes. For instance, Audio is changed to sample all 18 input bits instead of just the 8 most significant.

