6.111 Final Project Proposal

October 31, 2008

Team Members: Dimitri Turbine, Tyler Hutchinson, Bryan Newbold

Title: Real-time Visual Audio Composition

Overview

The reverse audio spectrograph bridges the gap between audio and it's visual representation, allowing users to graphically compose rhythm and melody. Images are scanned from a user controlled surface (a whiteboard most likely) and processed into an audio waveform. The reverse process, visualizing sampled audio waveforms in the spectral domain, is a commonly used technique in audio engineering and data analysis to gain insight into the quality and content of sound. The highly optimized discrete fast Fourier transform (FFT) has made high resolution spectral analysis a cheap and ubiquitous tool. Using the inverse FFT (IFFT), our project will allow creative expression in the spectral domain, and give users further opportunities for learning and audio exploration. Two complete spectrographs will be stored in memory to allow composition of a melody on top of a background beat; the two spectrographs will be summed ("overlayed") then sent to the output stage.

A graphical user interface (GUI) displayed on a VGA monitor will provide feedback on the system's inner workings, manage the composition of multiple spectrographs, and allow for customization of parameters and output audio filters via a dragable bode diagram.

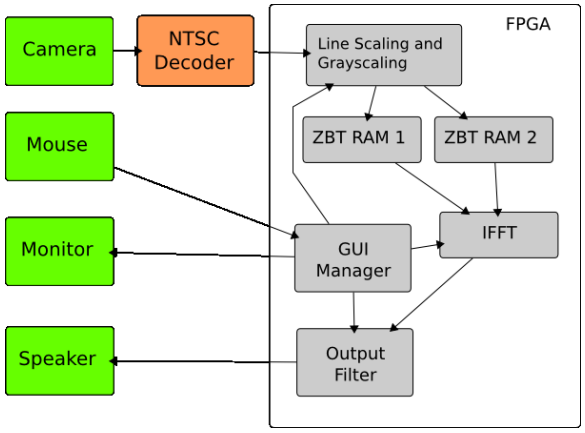

Figure 1: Rough Block Diagram

Input Stage

The input stage has a number of different responsibilities other than simply sending frames to the FPGA. Though the camera will have data to send at its refresh rate, a frame will not be stored in memory until the user has requested and image capture using the GUI. Also, for edits via white board drawings, the overlay input will also capture a frame, but it will be used in a different manner as described in the GUI section. Three discrete sections of the Input Stage will be important beyond the NTSC decoder which interfaces with the camera.

Visual Interpretation

After the NTSC signal has been decoded, the produced row of pixels will be processed. Specifically, the hardware will make hard decisions based on the pixel values that will be displayed in grayscale in the GUI, and scale from the 720 value horizontal NTSC signal into the 512 value signal stored in memory and later fed into the IFFT. While the original whiteboard image will vary even between colors, the brightness of the particular pixel will determine the magnitude in the frequency domain that we plan to look at. The scaled images will be displayed in the GUI to provide the user with feedback as to the interpretation of the frame capture image.

Data Storage

In addition to being displayed, the image will be stored in ZBT RAM so that it may be processed and continually displayed. The camera image will be 480 by 720, and if 10 bits are transmitted per pixel, at a maximum there are 844 kilobytes. Actual memory usage will be less because the 720 pixel line width will be scaled down to 512 pixels. We will use a separate ZBT ram block (1 megabyte each) to store the two spectrographs.

GUI

The GUI will contain the functionality of the Reverse Audio Spectrograph. Users will be able to select a number of different functions using a mouse input. A number of filtering options are also available for user controlled magnitude adjustment.

Function Explanations

Play Takes single lines of the graphical IFFT, which will be displayed at the bottom of the screen, and transmit them to the output stage. A horizontal scan bar will keep track of the current playback position.

Stop Ceases the playing of the spectrograph and leave the scan bar in its current position.

Capture Image Captures the main spectrograph image that will be transmitted whenever play is clicked.

Overlay Captures an second "overlay" spectrograph image that can be optionally added to the main captured image.

Scan Bar Move The scan bar can be moved to an arbitrary location so that the image may be decoded at an arbitrary starting position.

Edit Mode To modify a large portion of the melody or beat, an overlay may be used, but to change the value of individual locations in the image a simple edit tool will be available to set pixel values.

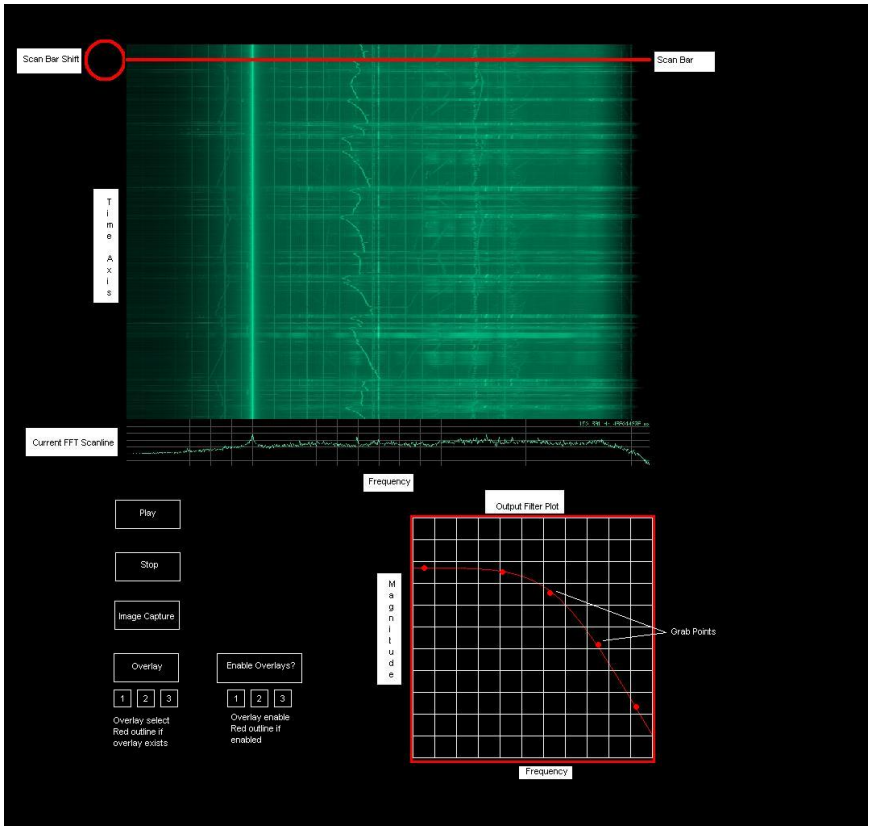

Figure 2: VGA Screen GUI Mockup

Bode Plots Set to a default value of a constant 1, the Bode plot will contain five grab points to modify the appearance of the Bode plot so that the output can be attenuated or left unchanged. In response to a Bode plot change, the captured image will dim respective to the attenuated frequencies. The current frequency domain coefficients and current time domain coefficients will be stored in BRAM memory. The 128 tap filter will use 16 bit values in memory. Time domain coefficients will be computed from the frequency domain adjustments whenever the user has ceased holding down a grab point.

Output Stage

Output processing will be performed in real time as values are fed to the output stage from memory. In order to ensure that there are no significant delays, which are typically quite apparent in audio playback systems, we will pipeline the data stream. Essentially, the first FFT line from the image will be stored in a pipeline register. On the next clock cycle, the pipelined value will be sent through the IFFT algorithm and stored in a pipeline register following this stage. Finally, the IFFT'd audio sequence will be filtered in the time domain using filter coefficients produced from the bode plot

The IFFT

A 512 sample IFFT will be used to produce the output audio signal. It will be assumed that the original FFT image was also produced using the same sampling number. It will also be assumed that the bandwidth of the signal is always 5 kHz so that the minimum sampling frequency would have been 10 kHz to avoid aliasing. Using this logic, a single FFT line would be $512/10\text{kHz} = 0.051$ seconds long.

Output Filter

As previously stated, the output filter will be produced by the 128 IFFT of the user customized bode plot. The bode plot can be adjusted mid playback to produce interesting effects on the current audio track and the delay due to IFFT computation should be minimal due to the small size of the IFFT block. The output filter, using the grab points, could be arbitrarily high pass, low pass, band pass, notch, or a number of combinations of these filter types.

Testing

Since computer programs already exist to compute the spectrograph of a particular audio track, both visual capture and audio playback could be tested using printouts from these programs. At the low level, we have already been provided with sine wave data from a previous project would could be used to test audio playback and filtering. A perfect sine wave could be mimicked simply enough as well by forcing a particular image function (i.e. a vertical bar), which would test the GUIs drawing capabilities as well as the transformation from input to output.

Work Breakdown

Tyler will be responsible for the GUI interface, including mouse input, VGA output, and maintaining parameters for the other modules.

Dimitri will be responsible for camera input, image scaling, and pipelining the BZT ram read/writes.

Bryan will be responsible for the IFFT and output filtering, including streaming to the AC97 audio chip and calculating the appropriate output taps for filtering and bode plot wave form to be displayed in the GUI.