FPGAAAuto-Tune

Team: Kika Arias and Elaine Ng

6.111 Fall 2019 Project Proposal

October 28th, 2019

Project Overview:

We intend to create a system that performs pitch correction. On a high level, the system will take audio input from a microphone, and perform audio processing in order to correct the frequencies recorded from the microphone. Our project will consist of the following main components: An ADC module, a recorder and playback which utilizes a filter, an STFT module, a spectrogram visualizer, a peak detection module, an IFT, and a pitch shifting module.

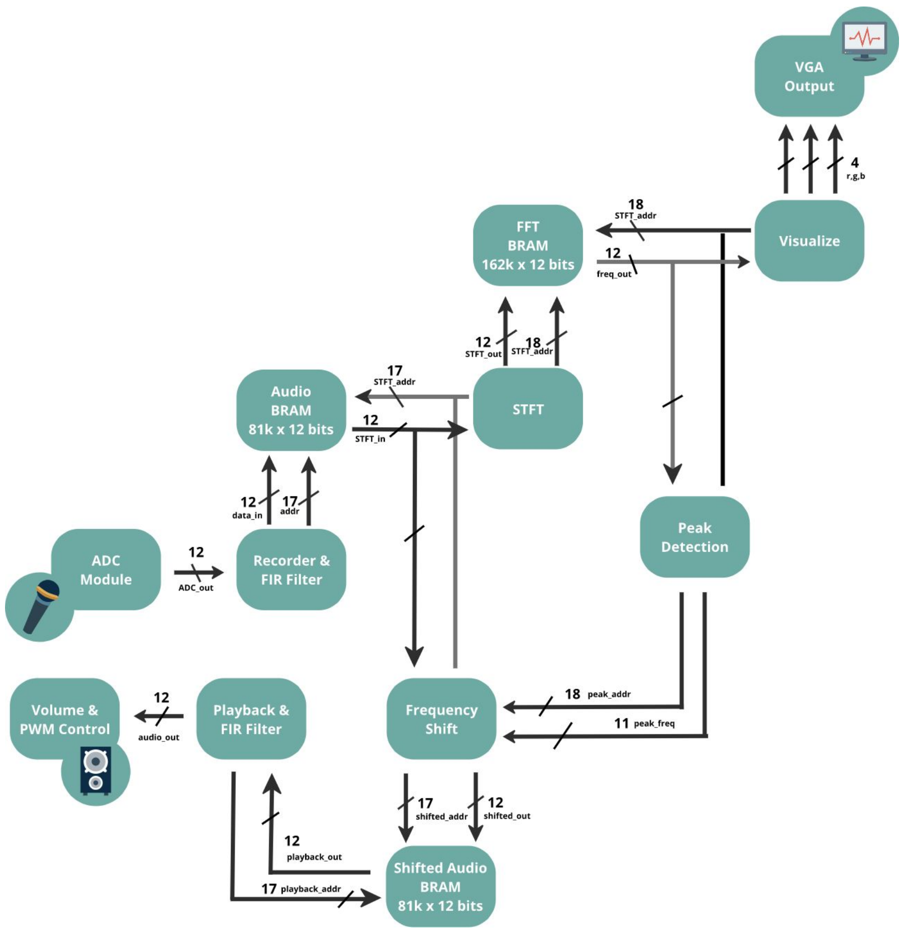

Figure 1. System block diagram for autotune describing major inputs and outputs to and from modules and BRAM

Recording and Playback (Kika) & Filter (Elaine):
Our recording and playback modules are mostly derived from the recording and playback modules in Lab 5A (the audio lab). The XADC module takes audio input from the mic through the AD3P and AD3N pins and measures a voltage between (+1 V and 0 V) using 12 bits of quantization. Then, the XADC output data gets downsampled to 48kHz, the sampling rate for our audio system, and converts the data from offset binary to signed binary. We want our samples to be 12 bits ,the maximum quantization for XADC, for better time resolution.

The recorder module takes in the 12 bits of sampled adc data at 48kHz. This then will be downsampled to 4kHz which means we take 1 sample every 12 so that we can record 20 seconds of audio while maintaining 81K samples in BRAM. This also means that the maximum frequency that our system can record is 2kHz, which is well beyond the upper limit for any human being's singing capabilities. These sampling rates and the number of samples in BRAM may change in order to get better time resolution for our system. The system has many variables to accommodate such as recording length, maximum frequency, time resolution, and frequency resolution. Such variables will need to be finetuned as our system develops. The playback component of the recorder module will function similarly to the playback implemented in lab 5A. We will reconstruct the signal by using zero extension and passing it through the FIR filter. The playback module will output 12 bits of data to a volume control, which outputs audio after being passed to a PWM control module.

The recorder module also interfaces with an FIR filter module, which is also mostly derived from Lab 5A. Since we are eventually our audio system sampling rate of 48kHz to 4kHz, we'll need to filter out all the frequencies above the Nyquist frequency, which is 2kHz to avoid aliasing. This will be accomplished with a 31 tap FIR filter implemented with a Hamming window. The length of the window, which is also the transform length of the FFT module, will be set to 1024 samples. The FIR filter module convolves the input 12 bit samples from BRAM with the coefficients of the 31-tap FIR filter, generated through a MATLAB script. We will test this module through two ways. As a qualitative test, we will record and test audio inputs and see if the audio quality improves (as we did in the lab). Once the Visualization module works, we will test the filter by analyzing the spectrograms of the unfiltered and filtered versions of test signals.

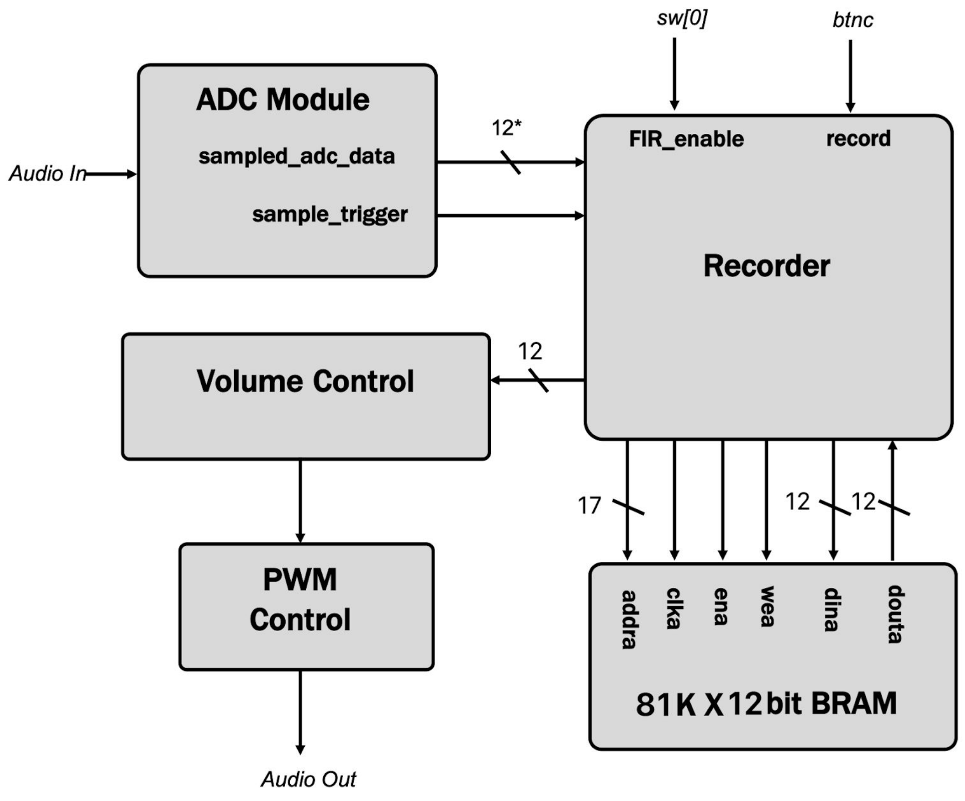

Figure 2. Block diagram from lab 5A of the recorder and playback subsystem we will be using

STFT (Elaine):

The STFT module will use a 1024-point FFT IP Core. The STFT divides the entire signal into windows of length 1024, which overlap by 1024 - (step size) and calculates the FFT for each window. We will start with an overlap of half the window length, so a step size of 512. The module will take inputs from the recorder BRAM. It will have a pointer for the current address, and at each clock will read from BRAM until 1024 samples have been read. At which point, the FFT core will calculate FFT for the 1024 samples, and store the results in another BRAM (FFT BRAM). The FFT BRAM will have to be 162k x 12 bits, as the STFT operates on (1024 samples per window)*(81k samples in total)/(512 step size) = 2*(81k) = 162k. The STFT module will be tested by using the visualization module. We will give the STFT module a test signal that we know the spectrogram for and see if the STFT module outputs the correct spectrogram.

Visualization (Kika):

The visualization module will serve a key role in allowing us to test the STFT and peak detection modules when we start using real signals. It will read from the FFT BRAM that is written to in the STFT module and maps the magnitude squared of STFT values to colors. The pixels will be drawn based on the color and the position of the value in the FFT BRAM. So, the module will draw the spectrogram from left to right as a function of time, because the address value is proportional to time. We will use the VGA interfacing as implemented in lab 3 in order to output our visualizations to a monitor.
In order to test this module we will generate a fake test signal to visualize the frequencies detected on the monitor.

Peak Detection (Kika):
This module reads from the FFT BRAM and searches from the lowest pitch to the highest pitch to find a peak that exceeds an intensity threshold that we will set. Once we find that peak, we will check to see if it has a certain width (this way we can discern actual notes sung from random noise). Since we are operating under the assumption that only one person will be singing at once, the lowest frequency detected will be the same as the fundamental frequency. The module will output a register storing the frequency value and a register storing the address corresponding to the highest peak.

Pitch Shifting (Elaine):
The Pitch Shifting module interfaces with the Peak Detection module, recorder BRAM, and an output FIR filter. First the module determines the correct frequencies that make up the signal by comparing each input frequency to a lookup table of correct frequencies.

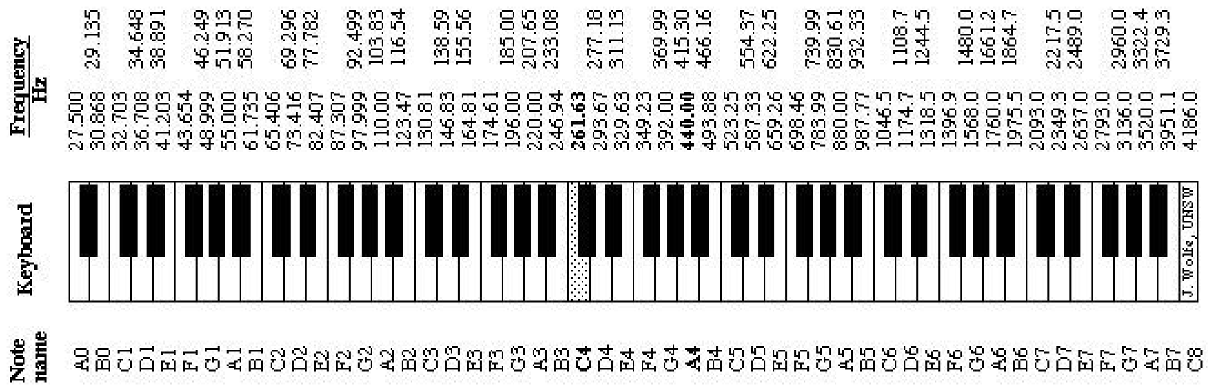

Figure 3. Correct Frequency Lookup Table

The module then shifts each frequency to the nearest correct value in the lookup table by multiplying the samples from recorder BRAM for each frequency by a sine wave at the corresponding correct frequency. We will probably run into issues in timing these modules so that we actually read the correct sample from recorder BRAM corresponding to the frequency we currently are shifting. Similar to the others, we will test this module by pitch shifting test signals and seeing the output spectrogram shift using the Visualization module and also listening to the output to hear the shift. As a first step, we will try to pitch shift a test signal to some high frequency (basically make a chipmunk effect).

The Pitch Shift module outputs to Shifted Audio BRAM. Another FIR filter that is identical to the recorder FIR filter, but reads from the Shifted Audio BRAM will prevent aliasing in the audio output.

External Components:
Our project requires a few simple components, a microphone to record audio, a speaker to output the audio, and a monitor to display the visualizations. We do not foresee needing to purchase any materials for this project.

Beyond the Minimum Viable Product:
Should we succeed in the minimum viable implementation of our project, we will add additional features to the system such as, other kinds of voice filters, real time autotune, loading and saving songs, or note detection. These extra features will be implemented in separate modules, as to not interfere with the existing system implementation and will simply be add ons for the user to test out using the various switches present on the FPGA.