CSEE 4840: Autotune

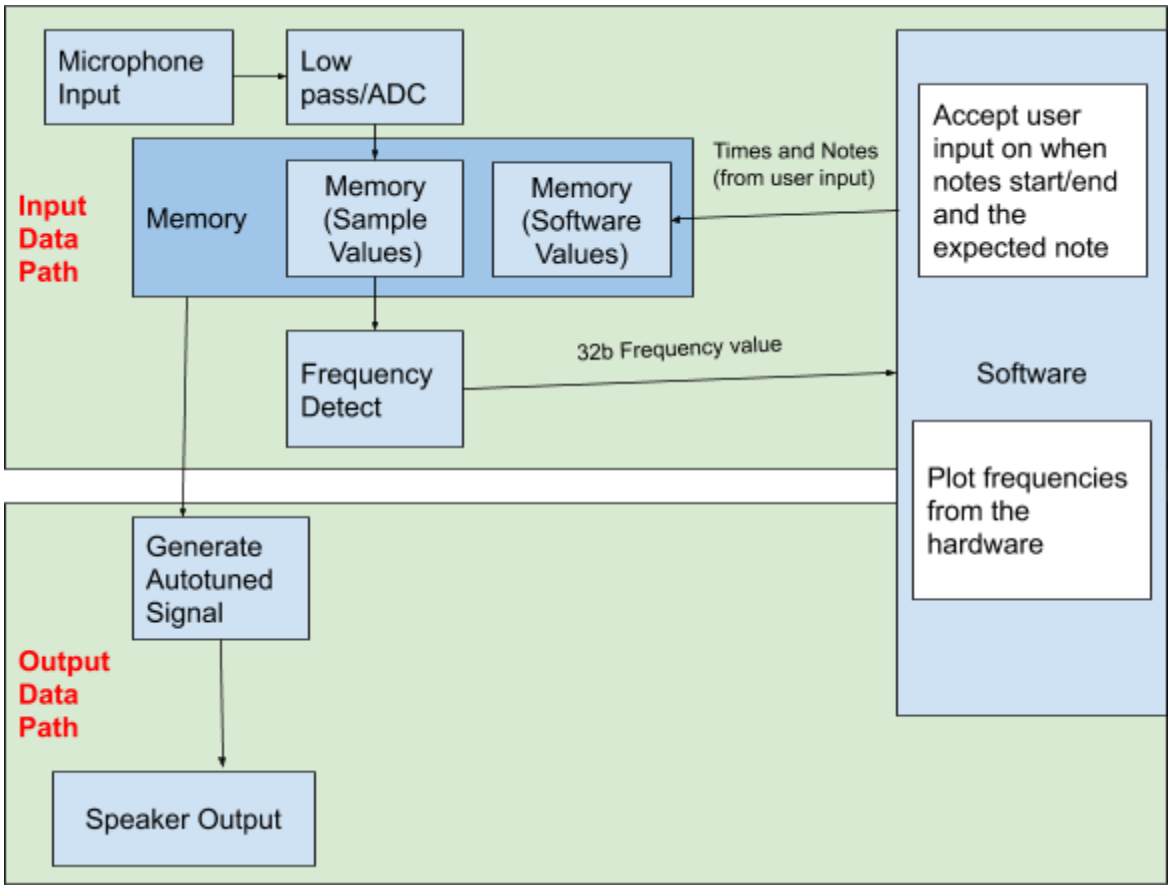
Fuming Qiu (fq2151)
Luna Ruiz (ler2167)

Table of Contents

Overview

This project intends to create a pitch correction device using a microphone, speaker and a Terasic DE1-SoC board (FPGA). Initially, the user clicks the record button on the user interface. After 10 seconds, the audio is collected by the hardware and undergoes some filtering. Next, the hardware sends these audio signal points to the software, where it is plotted on a graph. Here, the user is able to specify a range of points and the desired note for that interval. After that, the software outputs this data to the hardware, in which the hardware processes the signal and does some pitch correction. Finally, the signal is sent to the speaker where it would be played out loud to the user.

Block Diagram



Software/Hardware Interface

Software:

The software controls when to start recording, plots the frequency values that the hardware outputs, and allows the user to input a series of time intervals as well as the note value associated with those intervals. These are then sent to the hardware as the starting time, ending time, and expected pitch value of each note. The software will also control when to start playback of the autotuned recording.

The software expects the frequency values from the hardware to be 32 bit values, and the values that the software outputs are also 32 bit values. The reason behind this is because it is difficult for software to use a non-standard bit width and 16 bits was not enough.

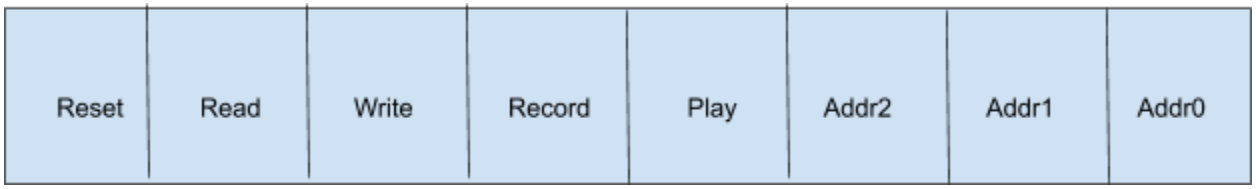
Hardware:

The hardware will be responsible for acquiring an audio signal and storing it in memory. Then, it will find the fundamental frequency or some other representative frequency of the recorded signal during each 10 ms interval. These values are then sent to the software side. Once the software has acquired these user values, these will be written to a much smaller, separate section of memory. Using these values from the software, the recorded signal will be autotuned and written back into the same place in memory. If time permits, an option for storing the autotuned signal in a separate location to facilitate playback of both the original and autotuned signals will be explored. After the autotuned signal has been calculated, the user can then play the selection on the speaker.

The input signal will be acquired using the audio CODEC included with the FPGA and the output signal shall also pass through the CODEC's DAC to produce the output signal. We expect to use around 6.4 MB of memory on the FPGA, assuming 40,000 samples/second at 16 bit resolution on one input channel.

Registers:

Since timing is not a huge concern, there will be one 8 bit control register:



Setting Reset to 1 will reset the system. When Read is 1, the hardware will output a value. When Write is 1, the hardware will take the address as the write address (start time, end time, or frequency). When Record is 1, the recording process is initiated and when Play is 1, playback will start. There will be a 32 bit output register from which the software will read the frequency value and there will be one 32 bit input register to which the software will send values to.

Milestones

Milestone 1

- We will be able to receive and store a 10 seconds audio recording of the user's voice from the microphone
- We will be able to clean the audio signal using filtering techniques (i.e. low pass filter) to reduce noise
- We will be able to reconstruct the signal and play it back on the speaker

Milestone 2

- We will be able to output 100 frequencies values per second from the hardware to the software
- Software collects these frequency values and plots them on a graph

Milestone 3

- Collect user input (time intervals and desired notes for those intervals) and send to hardware
- Store these values in memory

Milestone 4

- Perform given pitch correction to signal using interpolation and produce autotunes signal
- Output autotuned signal to speaker for user to hear

Challenges/Concerns

Concern 1:

- We are not sure how the signal will be collected from the microphone and how the signal will be stored on the FPGA

Concern 2:

- We are not sure how to communicate between the FPGA and the hardware. Will it be through some sort of device structure?

Concern 3:

- Should we enact some sort of exchange protocol between the hardware and the software so no values are lost?