Sebastian Krupa & Ashuka Xue

E155 Final Project Proposal

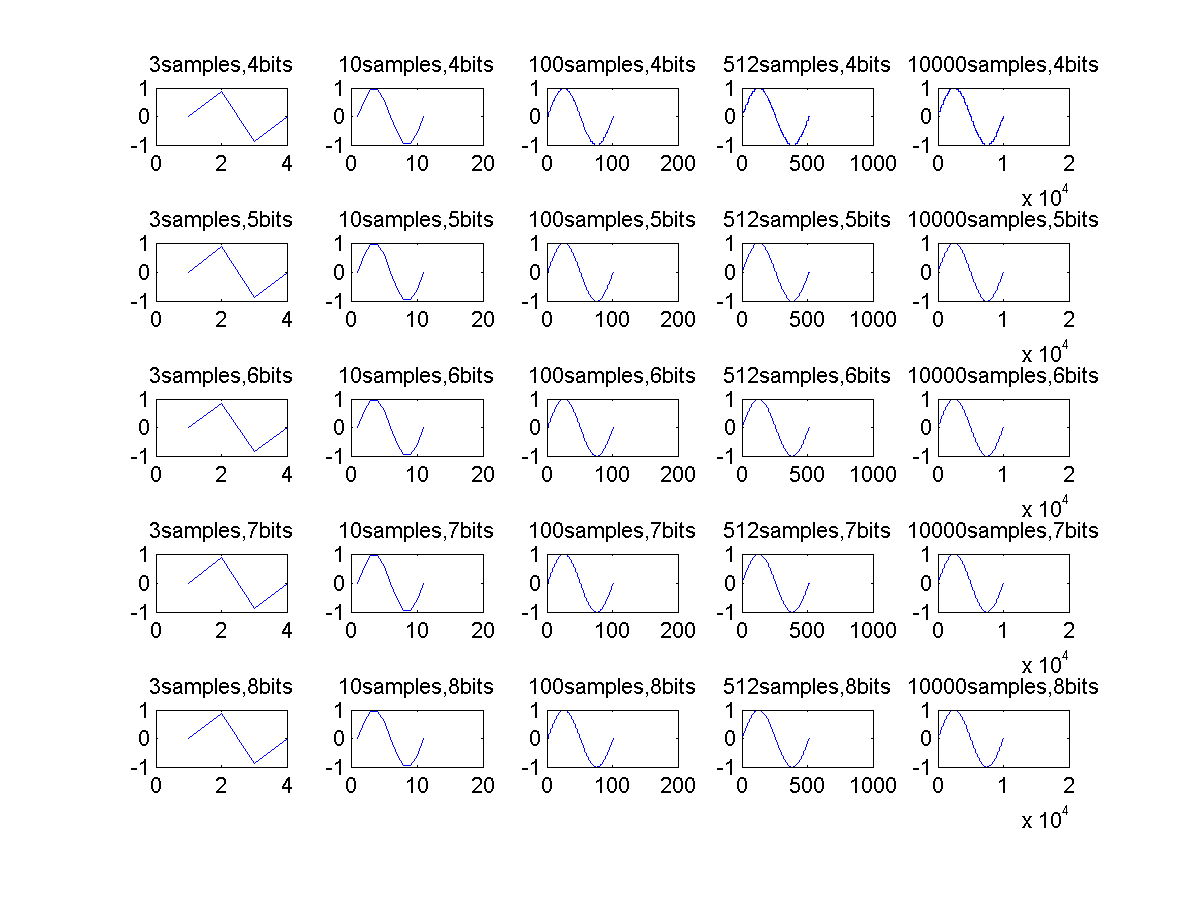
November 4, 2014

The goal of this project will be to create a digital keyboard capable of producing a variety of sounds and notes by varying frequency and waveform. The project will consist of a 12 note keyboard that will send data to the PIC when a key is being pressed. The PIC will then translate these inputs into the desired frequencies and send them to the FPGA using SPI interfacing. The FPGA will produce waves with the specified frequencies and waveforms, outputting them to a set of I/O pins which will be converted into an analog signal sent to a speaker.

KEYBOARD

The keyboard will consist of 12 3D printed keys to signify an octave. These keys will be attached to push buttons which will toggle the system between ground and 3.3 V. All 12 of these signals will be sent to 12 separate I/O pins to avoid scanning. This board will also include a switch to select between the different waveforms. Only one waveform will be able to be produced at a time for all 12 keys. Finally, there will also be an octave increment/decrement switch pair which will shift all 12 keys by an octave depending on user input.

PIC32



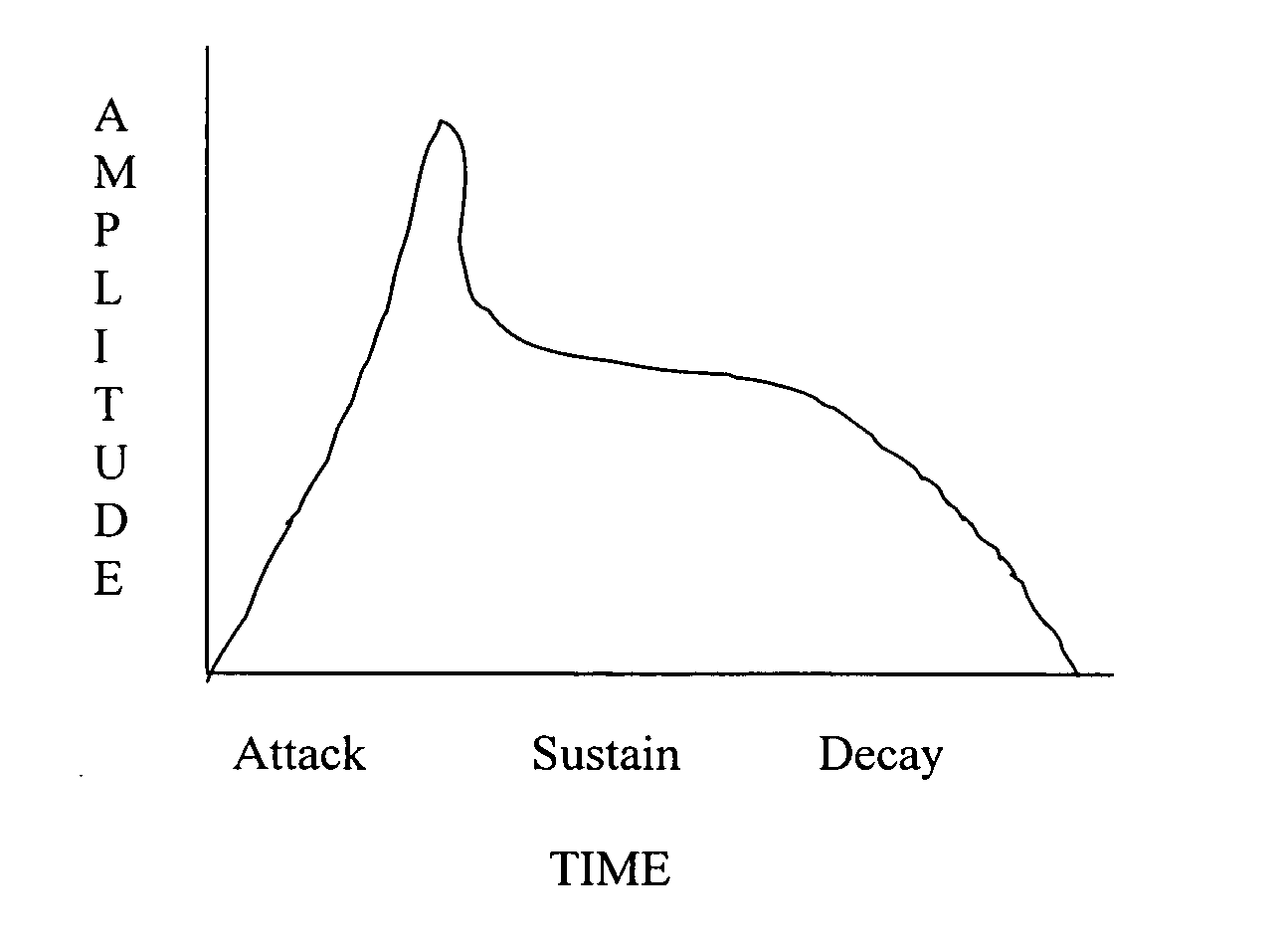
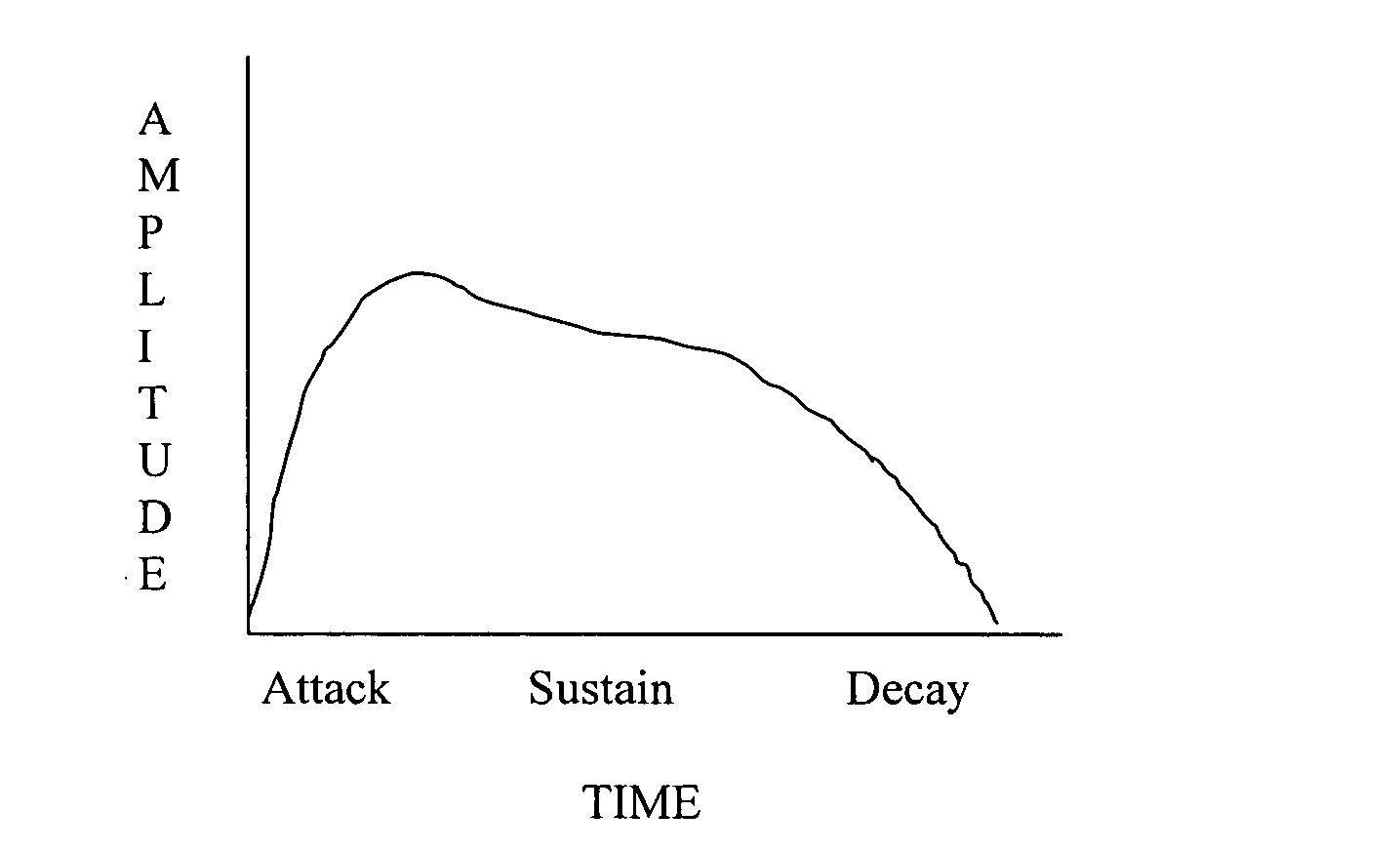
The PIC will be responsible for interpreting input from the keyboard and sending appropriate signals to the FPGA. It will read all 12 input pins to decide which keys are currently being pressed. The frequency will be determined by creating a clock signal that is generated by taking the note frequency and multiplying it by a desired sampling frequency. Our sampling frequency is 512 times the frequency of the note because it allows us to reconstruct the signal properly. It will also read which octave and waveform is currently selected. The PIC will also be responsible for producing 4 different waveforms: a square wave, a sawtooth wave, a triangle wave, and a sine wave which will be stored in memory. The PIC will sample the signal by comparing a time against the given frequency of a desired note. It will then send the note frequency to the FPGA to be played.

During implementation, there will be a maximum of 3 notes that can be pressed at any time. The PIC will samples the appropriate wave and return an 8 bit value for each note. These will be combined into one 32 bit number and sent to the FPGA through SPI interfacing.

FPGA

The FPGA will take the notes and add them together to produce an 8 bit number output to be sent to the FPGA. If 3 notes are pushed, the resultant 10 bit number which we will then divide this value by 3 to get an 8 bit number. If there are 2 notes pressed, the resultant 9 bit number will be divided by 2 to get an 8 bit number. If 1 note is pressed, the resultant wave is not altered.

To emulate a real piano, the FPGA will take the wave generated from the PIC and attenuate it so that it gradually increases when the note is hit and dies down as the note continues. The amplitude of a typical piano note over time attenuates like that of the image on the left. However, if we imagine strings being played there are resonate frequencies and then the amplitude looks like the one on the right. This can be easily modeled in the FPGA to generate the sound.

http://www.google.com/patents/US20040261605 (both images)

DAC

The DAC will work to create an analog signal from our 8 bit digital signal. In this case, Vout would range between ~0V and ~3.3V. Although not exactly 3.3V will be output, since all output values will be affected by the same constant multiple, the result will still be scaled properly. This Vout is then used to drive the speaker circuit.

SPEAKER

Finally, the speaker circuit will be similar to the one used for lab 5 in order to play the desired sounds. There will be a potentiometer so that we can manually adjust the volume coming out of the speaker.

BUDGET

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Unit Cost ($) |
| SLB1470 4 Position Slide Switch | 1 | 1.69 |
| GP0668-ND 8 OHM speaker | 4 | 4.87 |
| 35RASMT2BHNTRX 3.5mm Phone Connector | 3 | 0.66 |
|  | TOTAL | ~34.15 |