The DiGuitar

Ishaan Govindarajan
Eric Pence
11/15/19

Final Project Checklist

Commitment:

Single Note Detection with MIDI Output (with no latency requirements).

- Digital Low Pass Filter and Down Sampling Modules (as needed) to transform the analog input from the guitar into the digital signals and eliminate high frequencies from aliasing down.
- Frequency Detector Modules transform our digital signals from the time domain to the frequency domain.
- Note Decoder Module which identifies the lowest frequency currently being played.
- Onset/Offset Detector which determines which single note is being played (if any) and outputs that information to the MIDI Module.
- MIDI Module which sends 3 byte MIDI messages to a MIDI Compliant device for output.

Demonstration of Functionality:

If everything functions as expected:

We will play individual notes on the electric guitar and the MIDI device will output the correct note with noticeable delay.

If only certain submodules function as expected:

- Demonstrate Digital LP Filter/Downsampler through verilog test bench
- Demonstrate Frequency Detector Module(s) via test bench--feed in various sinusoids in and display the module output
- Demonstrate Note Decoder Module via test bench--will feed in simulated FFT magnitudes and view the output of the note decoder
- Demonstrate Onset/Offset Detection via test bench--feed in a simulated output from the decoding module and view the output of the detector
- Demonstrate the MIDI module by using switches on the Nexys board as “keys” and sending MIDI messages based on switch transitions (similar to how the module would operate). MIDI messages may be viewed in a music production program (e.g. FLStudio, GarageBand) or sent into a MIDI synthesizer to be generated into audio.

Goal:

Real Time (< 75 ms) Single Note High Frequency Detection (E4-$C^{\#}6$) with MIDI Output

- We will have all module present in commitment category.
- We will have a working < 75 ms latency *High Frequency Note Decoder Module*, likely implemented with a FFT but potentially an IIR Filter.

Demonstration of Functionality:

If everything functions as expected:

We will play individual notes in the range E4-C$^{\#}$6 on the electric guitar and the MIDI device will output the correct note without any noticeable delay.

If only certain submodules work as expected

Demonstrations will be identical to those described in the "commitment" section. Special emphasis will be placed on the latency of each module and system latency will be calculated by summing individual module latencies

Stretch:

Real Time (< 75 ms) Low and High Frequency Single Note Detection with MIDI Output

- We will have all modules present in commitment and goal categories.
- We will have a working < 75 ms latency *Low Frequency Decoder Module* which detects notes in the E2 - E4 range with no noticeable delay. This cannot be accomplished with an FFT and will require use of an IIR Filter or another low-latency solution.

Demonstration of Functionality:

If everything functions as expected:

We will play individual notes on the electric guitar and the MIDI device will output the correct note without any noticeable delay for any note on the guitar. These stretch objectives will not be developed if "goal deliverables" aren't able to be fulfilled (i.e. we will either be demonstrating all functionality or not attempt this demonstration).

Multi-note Detection (Stretch-Stretch Goal)

- We will have all modules present in commitment and goal categories.
- We will have a *Note Detection Module* that handles detection of up to six notes at a time by subtracting out harmonics from the signal. Our Note *Onset and Offset Detection Module* will also handle identifying multiple note onsets/offsets at one time.

Demonstration of Functionality:

If everything functions as expected:

We will play **multiple** notes on the electric guitar at the same time and the MIDI device will output the correct notes, potentially with some noticeable delay. These stretch objectives will not be developed if "goal deliverables" aren't able to be fulfilled (i.e. we will either be demonstrating all functionality or not attempt this demonstration).

Our Timeline is as follows:

Week 11/4-11/10: Get Parts, Record guitar audio snippets, Python prototyping of DSP techniques **(COMPLETED)**

Week 11/11-11/17: Analog input and sampling, MIDI verilog writing and testing, finish DSP prototyping

Week 11/18-11/24: Downsampling verilog, port DSP modules into verilog

Week 11/25-12/1: Integrate/Debug, Experiment with MIDI Output

Week 12/1-12/8: Touch Up