The DiGuitar

Ishaan Govindarajan and Eric Pence
Overview

TL;DR: take the analog input from a guitar and convert it into a digital MIDI datastream via DSP Magic™* in real time

*Not actually magic
Analog Signal from Guitar → Signal Conditioning (Preamplifier) → 1V Analog Signal → ADC on FPGA → 12 Bit Digital Bus

Note Decoder

22 Lanes of 12 Bit Data → IIR Filter Bank → 12 Bit Digital Bus

24 Lanes of 12 Bit Data → FFT Module (IP) → 12 Bit Digital Bus

Digital Low Pass Filter

2 Bit Digital Bus

Down Sampling Module

Digital High Pass Filter

Separate Low/High Frequency Identification Methods

46 Bit Bus (One Bit for Each of 46 Notes) → MIDI Protocol Module → Serial Transmission of 3 Byte Messages → MIDI Out (To MIDI Device)

6 Lanes for Note Amplitudes, 6 Lanes for corresponding MIDI Values

Sample @ 48kHz

Resolution of 0.0005 V
The “DSP Magic”
When u need to do some digital signal processing and have no idea what to do

FFT
When u need to do some digital signal processing and have no idea what to do
5Hz Resolution = 200ms long DFT!  
AKA 200ms latency!

Two lowest notes a guitar in standard tuning can play:

<table>
<thead>
<tr>
<th>MIDI number</th>
<th>Note name</th>
<th>Keyboard</th>
<th>Frequency Hz</th>
<th>Period ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>A0</td>
<td>E</td>
<td>27.500</td>
<td>36.36</td>
</tr>
<tr>
<td>22</td>
<td>B0</td>
<td>F</td>
<td>32.700</td>
<td>41.41</td>
</tr>
<tr>
<td>23</td>
<td>C1</td>
<td>G</td>
<td>39.200</td>
<td>46.46</td>
</tr>
<tr>
<td>24</td>
<td>D1</td>
<td>A</td>
<td>52.000</td>
<td>52.52</td>
</tr>
<tr>
<td>25</td>
<td>E1</td>
<td>B</td>
<td>65.440</td>
<td>59.96</td>
</tr>
<tr>
<td>26</td>
<td>F1</td>
<td>C</td>
<td>88.000</td>
<td>63.66</td>
</tr>
<tr>
<td>27</td>
<td>G1</td>
<td>D</td>
<td>110.000</td>
<td>72.73</td>
</tr>
<tr>
<td>28</td>
<td>A1</td>
<td>E</td>
<td>146.830</td>
<td>83.86</td>
</tr>
<tr>
<td>29</td>
<td>B1</td>
<td>F</td>
<td>196.000</td>
<td>98.00</td>
</tr>
<tr>
<td>30</td>
<td>C2</td>
<td>G</td>
<td>247.910</td>
<td>105.88</td>
</tr>
<tr>
<td>31</td>
<td>D2</td>
<td>A</td>
<td>329.630</td>
<td>113.93</td>
</tr>
<tr>
<td>32</td>
<td>E2</td>
<td>B</td>
<td>440.000</td>
<td>122.69</td>
</tr>
<tr>
<td>33</td>
<td>F2</td>
<td>C</td>
<td>587.320</td>
<td>130.61</td>
</tr>
<tr>
<td>34</td>
<td>G2</td>
<td>D</td>
<td>784.000</td>
<td>139.23</td>
</tr>
<tr>
<td>35</td>
<td>A2</td>
<td>E</td>
<td>1046.500</td>
<td>158.74</td>
</tr>
<tr>
<td>36</td>
<td>B2</td>
<td>F</td>
<td>1396.910</td>
<td>179.07</td>
</tr>
<tr>
<td>37</td>
<td>C3</td>
<td>G</td>
<td>1853.000</td>
<td>200.00</td>
</tr>
<tr>
<td>38</td>
<td>D3</td>
<td>A</td>
<td>2478.000</td>
<td>247.80</td>
</tr>
<tr>
<td>39</td>
<td>E3</td>
<td>B</td>
<td>3295.940</td>
<td>297.59</td>
</tr>
<tr>
<td>40</td>
<td>F3</td>
<td>C</td>
<td>4400.000</td>
<td>327.27</td>
</tr>
</tbody>
</table>

- **E2**: 82.407 Hz
- **F2**: 87.307 Hz
- **G2**: 92.299 Hz
RLC Band-pass Filter
IIR Biquad Filter
scipy.signal.iirpeak

`scipy.signal.iirpeak(w0, Q, fs=2.0)`

Design second-order IIR peak (resonant) digital filter.
The ideal frequency decoder...

- One IIR Biquad per frequency
- Can capture a frequency in 12ms max
The ideal frequency decoder...

- One IIR Biquad per frequency
- Can capture a frequency in 12ms max
- First time using IIR filters, so will be a stretch goal
Note Decoder

- Every note is made of a fundamental and overtones

- We only care about the fundamentals (i.e. get rid of the overtones)
Onset/Offset Detector

- Want to figure out when a note is played and not played
  - Detect **onsets** with **spectral novelty function** (as of now)
  - Detect **offsets** if amplitude crosses below certain threshold
MIDI Module

- 31250 Baud
- 3 Byte Messages: 1 Status, 2 Data
  - Command: Note On/Off (128-255)
  - Data: Pitch, Loudness (0-127)
Timeline

● Week 11/4-11/10: Get Parts, Record guitar audio snippets, Python prototyping of DSP techniques
● 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
Questions?