FPGA Tuner

Jeremy Sogo

December 2018

1 Introduction

A tuner is a device musicians use to tune their musical instruments to an agreed upon reference pitch. In its most basic form, a tuner records short samples of sound, determines what pitch is being played, and displays the name of the note and its intonation (i.e. how sharp or flat the note is). This display is updated continuously in real time, giving the musician live feedback as they make music. Many tuners are, unfortunately, can have delays exceeding half a second which can make adjusting intonation properly difficult. By implementing this process in hardware, note identification process can be done much faster, giving feedback on intonation even during the swiftest musical passages.

2 About Cents

Cents are a logarithmic unit of relative pitch often used for tuning. It is defined to have 100 cents per semitone (half-step) corresponding to an increase of frequency of a factor of $\sqrt[12]{2}$ and, thus, 1200 cents per octave which represents an increase of frequency of a factor of 2. For the purpose of this project, all frequency values are converted to a number of cents above C_0, which has the frequency 16.35Hz. Any given frequency f is N cents above C_0 where N is given by

$$N = 1200 \log_2 \left(\frac{f}{16.35 Hz} \right)$$

3 Overview

The tuner finds the pitch of a musical note by taking the Fourier transform of recorded audio samples and searching the frequency spectrum for peaks. It finds the center of the peak with the greatest spectral intensity and converts that frequency to a number of cents above C_0. The note name, octave, and intonation information are computed from that number of cents and are displayed on the seven segment display.

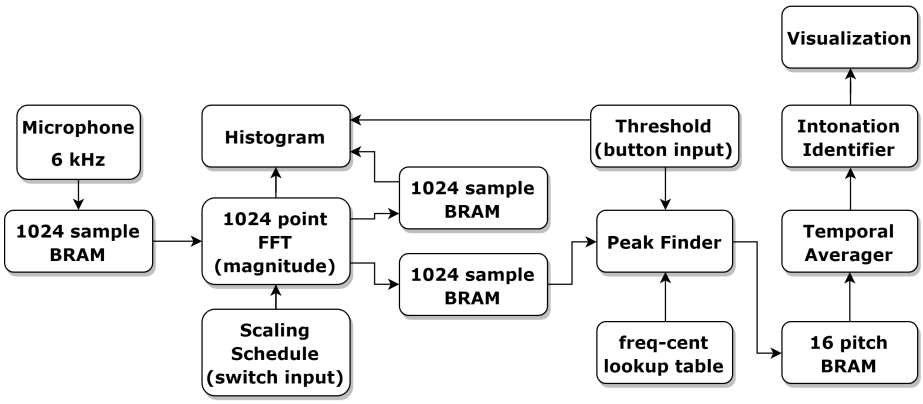

Figure 1: Block Diagram

4 Recording

The input to the FPGA is a microphone with automatic gain control that outputs a signal that is 2 V peak-to-peak centered on 1.25 V. This voltage is cut down by a factor of six with a resistive voltage divider before input into one of the Nexys's positive ADC ports. The corresponding negative port is tied to ground.

5 ADC and FFT

This set of modules controls the input of data into the processing modules. An IP core handles the access of data that ADC outputs. An oversampling module sums 16 samples in a row and combines them into one sample to reduce the effect of random noise. Those samples are saved in BRAM. An FFT IP core transforms those samples, and another core takes the magnitude of the complex transform and saves the result into another block of RAM at a rate of 60Hz. Due to numerous failed attempts at writing FFT modules (one of which functioned properly with the exception of a few scaling issues, but exceeded the hardware resources available to the FPGA by an order of magnitude) and esoteric difficulties with the ADC on the Nexys (it only started working when I plopped in an identical copy of the xadc core from the example code, despite the fact that they were visually identical and no code surrounding it changed), these modules ended up being largely based upon those presented in the FFT demo provided by the staff. A few alterations were made, namely the following:

1. The sampling rate was changed to 96 kilosamples per second. With the 16x oversampling, this brings the actual data sampling rate to 6kHz.

2. The Fourier transform was changed to take in 1024 points to speed up data processing.

3. The Fourier transform of the signal was saved in two BRAM elements, one for producing the histogram in its own clock domain, and the other for the data processing modules.

6 Peak Finder (pk_finder)

The peak finder module begins reading the data stored in BRAM after the FFT module completes its write stage. It scans from a set low index (see LOW_BIN parameter that determines when the start signal is asserted), to a set high index (see HIGH_BIN parameter that determines the end of the sweep; this is intended to prevent the symmetrical nature of the FFT from interfering with the identification of peaks). As the scan runs, it takes the running average of the past four points the module has read. If the newest read point is a certain value above that moving average (wire threshold is tied to the switches), that point is considered to be a peak. Once a peak is identified, the peak finder takes the weighted average of the current point and three points on either side of that point. The frequency of each bin is converted to cents above C_0 by a ROM lookup table (cent_table module). The value of the bin in cents is multiplied by the magnitude of the point to find the weight of the point. These are summed and then divided by the total magnitude of the seven points to find the center in cent-space and output that value. The sweep continues and the module stores the peak with the greatest area (i.e. greatest sum of magnitudes of the 7 points). This method has difficulties with harmonics at times, especially with voice which has unusually large harmonics. A more robust method would be to use these harmonics and find the difference between them to find what the fundamental frequency is.

7 Temporal Averager (temporal_avg)

Before input into the intonation identifier, the temporal averaging module averages the past 16 peaks (the FFT module spits out new data at 60 Hz). The last 16 cents values are stored in BRAM and are all summed and divided by 16 (bitshift by 4) to reduce jitter and noise of the visual output. The output is used by the Intonation Identifier.
This module could be improved by increasing the frequency at which the FFT module computes fourier transforms (it can most assuredly process faster than 60Hz) and the number of outputs it averages.

8 Intonation Identifier (note_name_calc)

This module takes in the number of cents from the peak identifier and determines the pitch class, octave, and intonation through two divisions with remainder. The dividers were made from divider generator IP cores. There are 100 cents to every semitone (half-step) and thus 1200 cents to every octave. The first

divider divides the number of cents given by the peak finder plus 50 by 1200; the quotient is the octave number and the remainder is the number of cents above C. The second divider takes that number and divides it by 100 to find the number of semitones above C, and the remainder becomes the number of cents sharp plus 50.

9 Visualization (display_note)

This module maps the note name (expressed as a 4 bit register), the octave (3 bit register), and the intonation (passed in as cents sharp plus 50, so 0 is -50 and 100 is +50) to segments on the seven segment displays. The module displays no note when no peak is detected by the peak finder.

10 Link to Verilog folder

Click here for zip file containing the Verilog for the project. Tuner is the top module.