The Multi-Octave Synthesizer FPGA

Name: Kelechi Igwe

**Summary**

For my final project of CPE 133, I decided to make a multi-feature synthesizer with a Basys3 Board, a speaker, and a Pmod component that connects the board to the speaker. The features include a multi-octave synthesizer and a seven-segment display module.

A synthesizer is a musical instrumental that emits electronic signals using sound waves to produce different notes and sounds. The synthesizer I implemented into my project takes the standard 100 MHz frequency system clock of the Basys3 Board and vibrates a speaker at a certain frequency to produce a specific tone. These specific tones represent the frequencies of musical notes that we use in everyday musical instruments. Using this method, I integrated 44 different tones into the program to allow the Basys3 Board to use the tones in different features.

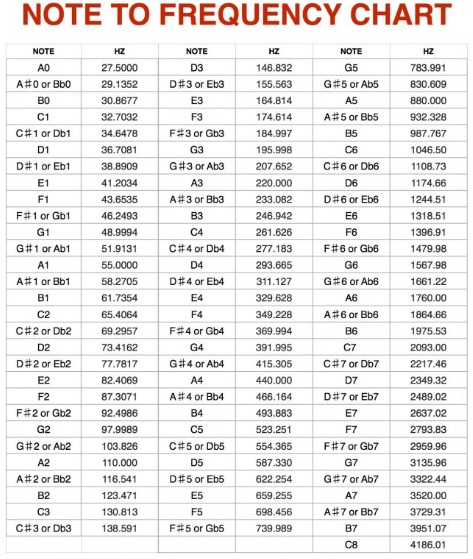
The seven-segment display on the board displays different things based on the user input. Without anything pressed, the display shows which octave the synthesizer is currently in. The display updates based on what your settings are set to. For example, if you press the top button to change which octave your synthesizer is in, the board display will change at the same time to show the new octave. This makes it easier to have changeable settings in the program to increase user creativity.

**What You Need**

1. **Basys3 Board**
2. **Pmod AMP2**
3. **Speaker (w/ AUX Cable Accessory and Power Output > 2.5W)**

**Get Started**

1. **Jot Down Frequency to VHDL Numbers**

****

Above you can see a chart displaying the different notes of music and the frequencies (in Hertz) of the notes. To make our speaker produce a sound, we need to vibrate it at one of those given frequencies constantly. To do so, for each note we want to be included in the program, we would take the frequency of the Basys3 board, which is set at 100 MHz (1.0 \* 10^8 Hz) and divide it by the frequency of that note. From there, we divide that result by two to get the MHz of the number that we want to use in VHDL. We then multiply that by 10^6 to get the final number. There are 44 different numbers, and so instead of doing all the calculations at once, I would calculate the numbers, and jot those numbers down somewhere so that when you are ready to code it in, it will be a quick process.

1. **Begin Coding the Seven-Segment Display Module**

Make a new VHDL file that will be used for managing the seven-segment display. Using the clock divider given to CPE 133 students by Professor Bryan Mealy from Cal Poly San Luis Obispo (code in genrestate.vhd), make a component in the new VHDL file for the clock. From there, we will use a method also from Professor Mealy to display the current octave number on the screen. The method ultimately creates a 2-bit counter and consistently shows the lit-up cathodes to the display at a fast rate from Mealy’s clock divider to make it appear like the numbers are constantly showing. Once that is setup, we create an FSM for the octaves so that after pressing the left button, the current octave goes to the next octave.

1. **Insert Frequency Numbers into New VHDL and Add Counter**

Create a new VHDL file to store all the frequencies that we jot down in Part 1 as signals (see notestorage.vhd). After setting all the frequency numbers as signals, create a integer counter that counts from 0 all the way to each frequency number so that when the counter reaches that frequency, it resets the counter to 0 and starts again. This allows the output to the Pmod AMP2 to switch from vibrate to not vibrate at a desired rate. Set each different frequency to every different case for current octave and current switch flipped.

1. **Combine All the Modules Together**

Now we’re almost done with the coding. All we must do now is make a main file (see finalproject.vhd) to combine the note storage module and the seven-segment display module together into one VHDL module. To do so, we create a new VHDL file with all the inputs we need (15 switches, 1 button, and the clock) and all the outputs we need (15 LEDs, seven-segment display, and the Pmod Amp2 outputs). We then make components for the two VHDL modules and use port maps and signals to carry signals to where they need to go.

1. **Testing (Synthesis, Implementation, Bitstream, Constraints)**

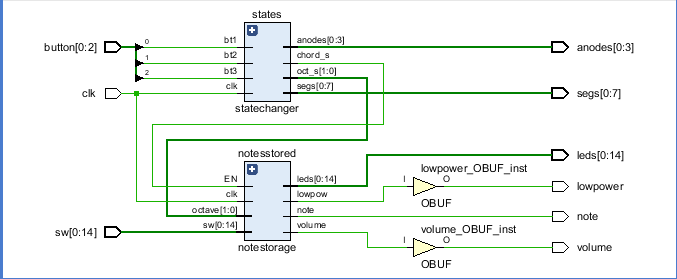
We are all done! Kind of…every engineer knows that a project isn’t really a project without testing it multiple times. To verify the functionality of my program, I ran multiple synthesis, implementation, and bitstream runs to make sure my code was logically correct and physically correct. In addition to Vivado tests, I also did a lot of testing with the switches and buttons after programming my Basys3 Board with my main code. Below will show the controls of the program based on the Basys3 Board.

1. **Physical Connections**

To get all the connections and devices working, use a mini-USB (or however you connect the Basys3 Board to your computer) to connect your board to the computer. From there, plug in the Pmod AMP2 to the top female header row on the top-right header (see picture at the top). Connect the AUX cord to Pmod AMP2 jack that’s already connected to the Basys3 board.

**And You’re Done!**

**Overview Board Design:**

****

**Video Links:**

**Playing w/ The Synthesizer**

[**https://drive.google.com/file/d/1LC1tyCFDuQKiANRTrFslfgBX0rCxAbbT/view?usp=sharing**](https://drive.google.com/file/d/1LC1tyCFDuQKiANRTrFslfgBX0rCxAbbT/view?usp=sharing)

**Acknowledgements and Links**

**Note to Frequency Chart:** <https://www.doctormix.com/blog/note-to-frequency-chart>

**Speaker:** <https://www.amazon.com/EasyAcc-Portable-Bluetooth-Speaker-Function/dp/B00JK6MQEI?tag=bom_tomsguide-20&ascsubtag=TGUS16021527723593011-tgus:en_US_21_PicStory_918&SubscriptionId=AKIAJLYKPRLXUSF4GDIQ&linkCode=xm2&camp=2025&creative=165953>

**Basys3Board:** <https://store.digilentinc.com/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users/>

**PmodAmp2 (Note: The volume starts really loud so lower speaker volume before playing):** <https://store.digilentinc.com/pmod-amp2-audio-amplifier/>

**Inspiration From:** <http://www.instructables.com/id/Basys-3-Keyboard-Synthesizer/>