ECE 287 Final Project

Carter Jones

Emory Landreville

Section D

**Electric Piano**

**Description:**

Using a DE2 Altera field programmable gate array (FPGA), a ps/2 keyboard and a VGA computer monitor we have made an electric piano that also displays what note is currently being played on the monitor. The FPGA components that are most important to our project are the VGA output, audio output, and the ps/2 input. Certain letter keys on the ps/2 keyboard signify the specific notes (“C” through “C” of the next octave) of a real piano. The VGA is implemented to show a music scale while the FPGA is on so that when a key is pressed, the note corresponding to that key will show up in the appropriate location on the music scale. The PS/2 keyboard controls the user input of our project. The following keys are pressed to play a note and to draw the note on the screen. This table shows the codes corresponding to what key and to what note it should play:

|  |  |  |  |
| --- | --- | --- | --- |
| Key | Hex Code | Binary Code (8-Bit) | Note |
| Q | 15 | 00010101 | C |
| W | 1D | 00011101 | D |
| E | 24 | 00100100 | E |
| R | 2D | 00101101 | F |
| T | 2C | 00101100 | G |
| Y | 35 | 00110101 | A |
| U | 3C | 00111100 | B |
| I | 43 | 01000011 | C |

**Note**: Each key has a make and break code. The break codes are not shown here because the break codes for all keys are the same as the make code except the code (in hexadecimal) F0 is placed before.

**Design:**

The best way to start this project was by implementing and understanding the VGA output first. Getting the FPGA to correctly draw the shapes we needed was a big step. Through VHDL coding we were able to manipulate the column and row constraints to evenly space out the shapes representing the musical notes on the screen. The next step was to make these shapes only show up when called by the pressing of a key on the PS/2 keyboard. This is where the table above helped to match up each note to the correct make code of each key. Once the keys were properly attached to a note, the hardest part was then getting the FPGA to produce sound. The clock has to be individually manipulated and divided to obtain the frequency needed to make a particular note. Figuring this out takes a little bit of math but is not very difficult. Here is a table to help explain:

|  |  |  |  |
| --- | --- | --- | --- |
| Note | Frequency | Dividend in Decimal | Dividend in Binary |
| Low C | 1046.5 | 47778 | 1011101010100010 |
| D | 1174.66 | 42566 | 1010011001000110 |
| E | 1318.51 | 37922 | 1001010000100010 |
| F | 1396.91 | 35793 | 1000101111010000 |
| G | 1567.98 | 31888 | 0111110010010000 |
| A | 1760 | 28409 | 0110111011111001 |
| B | 1975.53 | 25310 | 0110001011011110 |
| High C | 2093 | 11945 | 0010111010101001 |

Once all note frequencies are calculated the next thing to do is connect these frequencies to each key you will be using on the keyboard. We decided to use the top row of the keyboard “Q” through “I” to represent a single octave of the notes “C” to the next “C”. Implementing a key to output that frequency when pressed is not difficult if you already have the VGA displaying a note when the key is pressed. Finally the output uses a 3.5mm auxiliary port connected to either headphones or speakers of your choice.

**Conclusion:**

The three main components of our project were the VGA output, PS/2 Keyboard input, and Audio output. In all, the goal was to implement an electric piano using an FPGA so our project was a success but did not turn out to be as perfect as we expected. There were no troubles with the VGA when it came to displaying the corresponding note on the monitor except that we used rectangles instead of circular notes. Each note precisely corresponded to the correct PS/2 keyboard key when that key was pressed so the interface between the PS/2 Keyboard input and our VGA output had minimal trouble. One small problem is when a key is released, the note will continue to be output and the note will continue to be displayed until another key is pressed. This could possibly be fixed by adding more “IF” statements to our code concerning the make codes of the PS/2 keyboard keys, but if this change was executed then it could have also slowed down the ability to play a different note immediately after a previous note. Lastly the audio output did not turn out as well as we initially hoped. We calculated the correct frequencies to represent each note, connected the frequencies to the appropriate keys, and output that frequency when the key was pressed. The issue came with the FPGA itself and the quality of sound able to be generated by the FPGA. We used square waves to represent the frequency desired and the circuitry of the board caused an extreme amount of static in the output. The only way we maybe could have improved or prevented this quality of sound were by either importing an audio file with an analog signal to represent the frequency or to use a Digital to Analog converter to convert our frequency for us. These would simply improve our coded side of sound quality but the FPGA ultimately could still cause the static alone. Aside from minor flaws, all three elements of our project are effective enough to say we have implemented an electric piano using a field programmable gate array.

**Sources:**

Larson, Scott. "PS/2 Keyboard Interface (VHDL)." EE Wiki, 29 Nov. 2013. Web. 10 Dec. 2015. <https://eewiki.net/pages/viewpage.action?pageId=28278929>.

Larson, Scott. "VGA Controller (VHDL)." EE Wiki, 1 Aug. 2013. Web. 10 Dec. 2015. <https://eewiki.net/pages/viewpage.action?pageId=15925278#VGAController(VHDL)-CodeDownloads>.

"Fpga4fun.com - Music Box - Part 1." *Fpga4fun.com - Music Box - Part 1*. KNJN LLC, 21 Jan. 2015. Web. 10 Dec. 2015. <http://www.fpga4fun.com/MusicBox1.html>.