Piano Simulator with Keyboard and Accelerometer Interface

Matthew Bayer, Nick Deneau, James Khoury, Logan Verstraete
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
mjbayer@oakland.edu, ndeneau@oakland.edu lpkhoury@oakland.edu, ldverstraete@oakland.edu

Abstract

The team's project is a piano simulator. The project was constructed using Vivado, VHDL code, the Nexys-4 FPGA board's built in accelerometer, a speaker, and a keyboard. The purpose behind the project is to use the keyboard to press a key that corresponds to a specific note on the keyboard using frequency control. Frequency control was used to map eight frequencies to corresponding keys on the keyboard. The notes that are pressed will also be displayed on the seven segment display of the FPGA board. The second part of the team's project is using the Nexys-4 board's accelerometer to control the audio output.

I. Introduction

The motivation behind the piano simulator is to use the new knowledge learned in ECE 3710 combined with previous knowledge to create a fun and creative way to play the piano. The topics that were learned in this class that can be applied is how to interface the board with a speaker to output sound and learning how to use the board's built in accelerometer. One topic that the team had to learn on their own is how to interface the keyboard to the FPGA board.

II. Methodology

A. Functionality

The project's code can be broken down into four main parts: The audio file that generates eight frequencies, the keyboard file that allows the user to press keys S-K to generate sound, the accelerometer file that allows the user to tilt the board to generate sound, and the top file that connects all the files together. The project is constructed to have two different modes. With the use of a switch (SW [0]) the mode of the project can be controlled. When SW [0] is high the project enters accelerometer mode and when SW[0] is low the project is in keyboard mode. Below is the flowchart and block diagram of the overall scope of the project.

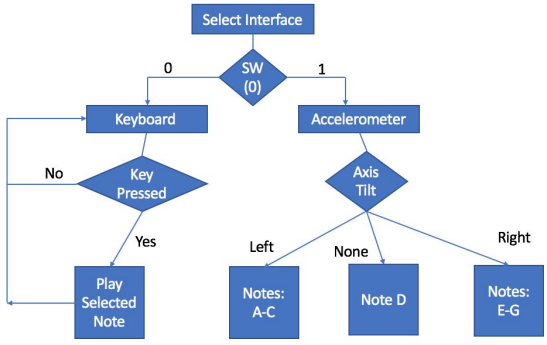

Figure 1: Flowchart

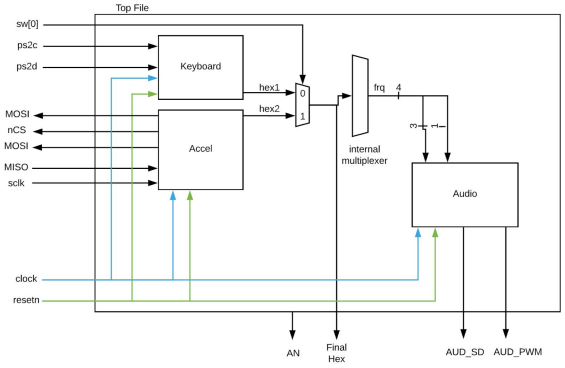

Figure 2: Block Diagram

B. Keyboard Interface [1]

The keyboard will allow the user to press the designated seven keys in order to play the notes that were generated from the audio file. The initial goal was to press the designated seven keyboard keys in order to play the notes A-G in the middle "c" octave of a piano. This will be done using the USB connection on the FPGA board. The keyboard keys used for the notes are S, D, F, G, H, J, and K.

The file used is ps2keyboard. In the file the data transferred between the the keyboard usb input to the FPGA with ps2c and ps2d. Ps2c is a clock alternating between 0 and 1 and is used to cycle the input data which is called ps2d. Each data comes into the fpga as a single bit. As each single comes into the fpga it will wait for 10 data inputs. The start bit will be '0'. After the start bit is inputted then the data of the keyboard which 16 bits is inputted. The 16 bits will input from least significant but to most significant bit. After the 16 bits is inputted, the ps2d will have '1' for parity bit and '1' for stop bit. As each keyboard key is pressed the ps2c and ps2d will repeat the process. Keys will be pressed one at a time to insure the right data is being transferred to the fpga.

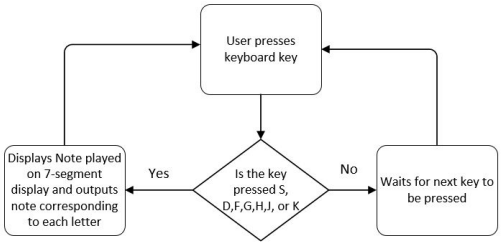

Figure 3: Flowchart of Keyboard Interface

C. Audio Output [2]

The audio output is enabled using the FPGA board and the headphone jack located on the board. This is enabled using the XDC file and the notes are sent to it using PWM signals. The AUD_PWM generated from the tone control would be captured and mapped onto a corresponding key.

In order to obtain these different frequency notes the team modified the PWM: Tone control Vivado project given to us by Dr. Llamoca. The files contain various components such as LUT, register, and tri state buffer. The overall scope of this file is to input a three bit frequency "number", 0-7. This 0-7 corresponds to 8 different frequencies which gets outputted to the AUD_PWM port on the FPGA board which produces the sound. The file was modified to produce 16 different frequencies. Seven of the sixteen frequencies were selected due to overall code design. [3]

The code changes the time it takes to adjust the duty cycle of the output from 0-255. The shorter the time the higher the frequencies. The duration of a note was controlled by how many times the duty cycle would reach 255. This would produce notes that had different duration depending on their frequency. Waiting for the duty cycle to reach 255, five hundred times would produce a note with a duration for about 2-3 seconds.

D. Accelerometer

From lab 3 the team used the accelerometer code, more specifically wr_reg_axl362, FSM_accel, decoder, and register. The FSM uses a counter used to specify which data is being inputted such as the temperature and accelerometer. Using i and j to specify what register is being enabled for the data from wr_reg_axl362. Using the decoder the data from the FSM transfers the enable is used to

store the wr_reg_axl1362 data into the register. There are seven cases that decide what note is being played. The cases depend on the certain range of data from the register. The cases range as the highest positive number being A. While the positive number decreases the notes plays B, then C, and then D. As the negative value increasing the notes play E, F, G, where G is the highest negative value.

E. Top File

The project_top file combines the code together to create the project. The audio files, keyboard files, and accelerometer files are all port mapped into the top file. Also, there is an internal multiplexer which uses the keyboard variable hex (which states which key is pressed) as the selector for which frequency is inputted into the audio file (which then outputs the sound). So as each key is chosen the corresponding frequency (if applicable) is sent into the sound file. The accelerometer file is also port mapped into the top file to enable the use of the accelerometer as a piano key selector. Switch 0 on the Nexys 4 board is used as the selector for whether they keyboard is being used or the accelerometer. The top file incorporates a "when" statement to select which user interface is being used.

III. Experimental Setup

The setup that the team will use to verify the functionality of the project is to connect a speaker to the FPGA board, match the notes frequency with the note notation on the seven segment display. The software used will be VHDL coding in Vivado and the hardware used will be a FPGA board, keyboard, computer, and speaker. The expected results are to hear the correct notes output of the board with the correct notation, and to play a song.

IV. Results

The team successfully implemented multiple different applications of what was learned in ECE 3710. The keyboard interface worked as planned including the use of the seven segment display of the notes being played. The audio files outputted the correct notes at the right time. The frequencies could have been tuned a bit more, but the team concluded that the tuning could have one of the improvements made in the future. By adding the accelerometer, another aspect of the course was successfully implemented into the project. Also, this added a bit more complexity to the code implementation.

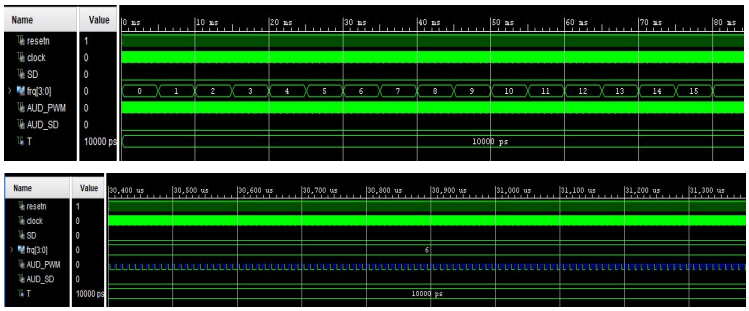

Figure 4: Simulation Snippets

V. Improvements

Some ways to improve the sound synthesizer would be:

- Tune the synthesizer to produce a full octave of notes from C-B
- Utilize more keys to have a synthesizer of 2-3 complete octaves
- Modifying the duration of the notes so that the notes are all the same length

VI. Conclusion

In conclusion, the project produced a good learning curve for the team. By learning about the PWM Tone Control, the team was above to successfully alter the code to produce 16 different output sounds, from

which 7 were chosen to be used as notes A-G. The keyboard worked as planned. The accelerometer added a different aspect to the project that allowed the team to add a different way to play a note while using knowledge learned in the lab portion of the class. Overall, this was a very successful project.

References

VHDL Tutorials on class website “PS/2 Keyboard Controller” [1]

VHDL Tutorials on class website “PWM: Tone Control” [2]

Moodle Notes Unit 3 [3]