Please attach this cover sheet to your lab report

ECE324 Lab 8: Keyboard Music

Name(s):

Alex Blake

Jameson Shaw

Exercise	Course outcome	Grade
Lab8 Demo	2.a, 2.d, 5.c, 7.b	/15
Lab8 Extra Credit	2.a, 2.d, 5.c, 7.b	/3
Lab8 Report	2.a, 5.c, 7.b	/25
	TOTAL:	/40

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.
2.d. Produce FPGA designs that meet specified needs.
5.c. Collaborate with individuals with diverse backgrounds, skills and perspectives.
7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

In this lab, students are asked to create a musical keyboard using SystemVerilog on an FPGA. Students will need to know many concepts including synchronous logic and flip flops. The project is mostly completed beforehand and is given to the students by the instructor.

Procedure

The first step is to obtain the starting code for the project and add them to a new project. Once this is created and put together, the code should be ready to be synthesized and programmed to the board. Verify it works by plugging in a pair of headphones and USB keyboard and press corresponding keys to notes. This includes the 'volume down' functionality of the 'left ctrl' key on the keyboard. Now that the hardware works, modifications can be made to add functionality.

The first change is to add another mapped key (backslash) to the note A5. This requires a few steps; find the frequency required and calculate the period required given calculations in the commented code and set that required value to the mapped key so that the key press will produce A5.

The last step is to implement two more states to the FSM within the main file to add 'volume up' function with the 'right ctrl' key on the keyboard. The states are already there in the code as well as the condition to go to those states, they just need to be uncommented. This step really only requires adding the functionality within the cases. Add the conditions to move to different states as well as the outputs that the FSM should have within the state to change the volume; the 'volume down' function shows how to accomplish the volume change.

Now that these are changed, the code should be ready to be synthesized and uploaded to the FPGA board and be verified by the instructor. This completes the lab.

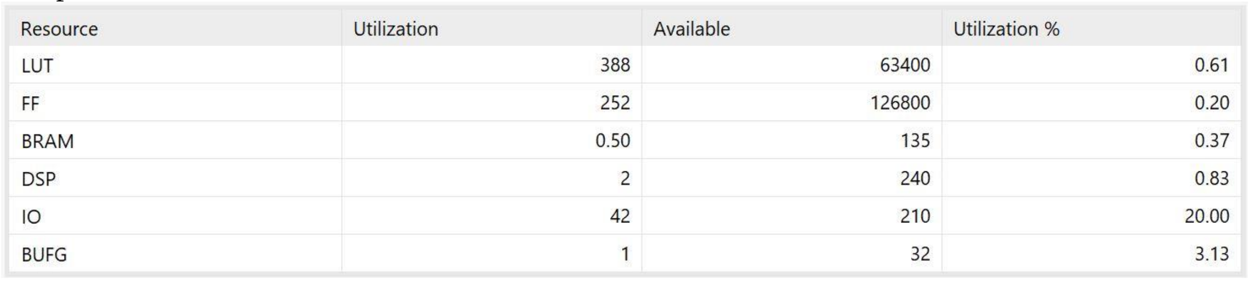
Results

The code synthesized and worked exactly as expected when uploaded to a board with a pair of headphones and USB keyboard attached. The hardware produced exactly what was expected and no demonstration was required for this step.

Adding A5 was simple as well. After looking at the code for a while to figure out exactly how it works, the required code to be added here was going to fit the exact format as the rest of the notes; it only required finding two values (code from keyboard and required value for frequency of A5). The frequency is double the lower octave of 440Hz so the frequency of A5 is 880Hz. Using the calculations method in the comments in this section, the 'fccw' value is 9449 and set to the keyboard 'makeCode' 0x5D. This mapped A5 to 'backslash' as expected.

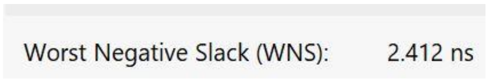
The last step of the lab required a bit more troubleshooting and understanding of the FSM. After uncommenting the required cases and line within the 'IDLE' state, we used the method in the states above as a guide to set 'increaseVolume' = 1 which increases volume. The issue we had was that we set the nested 'if' statements with 'increaseVolume' = 1 when only the 'rx_done_tick' is high instead of both 'rx_done_tick' & 'rxData' == 'BREAK_BYTE'. This caused one press of the key on the keyboard increased the volume more than one unit at a time which was incorrect. Once the 'if' statement was fixed, the FPGA behaved exactly as expected.

The professor checked off our board for all the features added and verified it worked correctly to conclude our lab. Figure 1 shows the utilization of components on the board. Just like the previous labs, the utilization percentage is really low and doesn't come close to using all of the components on the FPGA.



Resource	Utilization	Available	Utilization %
LUT	388	63400	0.61
FF	252	126800	0.20
BRAM	0.50	135	0.37
DSP	2	240	0.83
IO	42	210	20.00
BUFG	1	32	3.13

Figure 1: Utilization of the Keyboard Music Project on the FPGA



Worst Negative Slack (WNS): 2.412 ns

Figure 2: Negative Slack

Using Figure 2, the fastest this project can run is found by the following:

$$T_{Worst} = 2.412ns$$

$$f_{Worst} = \frac{1}{2.412ns} = 414.6MHz$$

The frequency of which this project could run without issue is around 415MHz, which is much greater than the 100MHz it ran at for demonstration. This means there should be no instability issues since 100MHz gives us plenty of room for the worst case.

There was extra credit for this lab as well. The extra credit required no change in code, but required understanding what the code does to manipulate values using the switches and buttons. The values manipulated were ADSR envelope. Basically, everything needed to be set to one second (attack, delay, sustain and release) and the sustain level to be ½. This is done by setting the switch to the correct value and hitting the button corresponding to what characteristic which then captures the binary value represented by the switches. We set the ADSR envelope to all the corresponding values and got it checked off by the professor, ending the lab.

Conclusion

This lab was great for practicing all the skills learned from the course and applying them to a new interface (USB keyboard) and how to output that to a DAC which drives a speaker. These functions will most likely be included in the final project so it is very important and useful.