Desmond Andersen
Dhruv Jain
CS M152A

Lab 4 Report

I. Introduction and Requirements

In this lab, we created a looping drum machine or synthesizer with four tracks that can be layered on top of each other. This project uses the Pmod AMP2 module as an audio interface, along with using the 16 switches on the Basys-3 board as input. We also use the seven-segment display to show the user which track is being recorded.

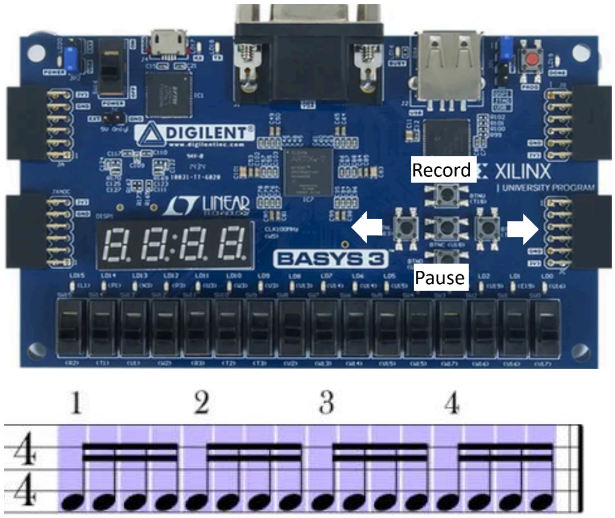

Fig. 1: How the buttons and switches map to the user controls.

II. Design Description

We only used a single module, with the following interface.

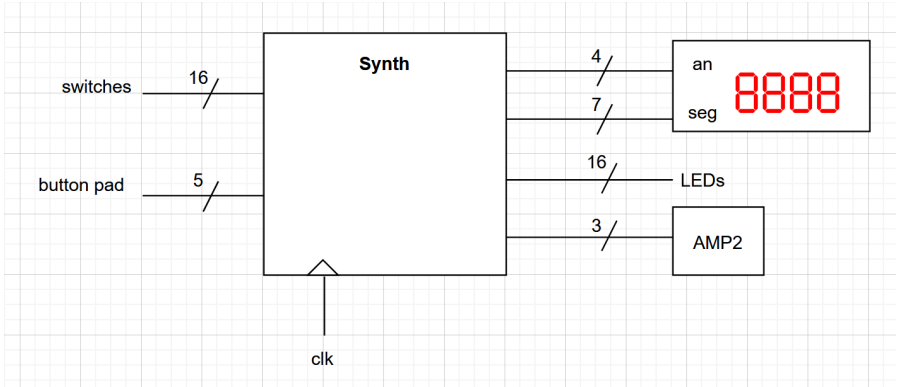

Fig. 2: High level schematic of our Verilog module.

We were able to glean major functions such as button debouncing and display driving from previous labs.

Input		Output	
`sw [15:0]`	Signals from the 16 switches for input (each representing a beat)	`seg [6:0]`	Control each segment of the 7-segment display
`clk`	100 MHz clock signal	`an [3:0]`	Control which digit to multiplex the `seg` signals to
`btnLeft`	Switch to previous track	`freq`	PWM signal used to generate tone with AMP2 module
`btnRight`	Switch to next track	`gain`	Control signal for the AMP2 module
`btnUp`	Record onto current track	`shutdown`	Control signal for the AMP2 module
`btnDown`	Reset all tracks	Led [15:0]	Light up LED corresponding to to the current beat in the loop

The module first creates multiple clock signals with different frequencies by dividing the input clock (100MHz) into a 200Hz clock, 275Hz clock, 350Hz clock, 425Hz clock, 488Hz clock, 500Hz clock, 517Hz clock, 567Hz clock, 600Hz clock, 617Hz clock, 675Hz clock, 750Hz clock, and 1000Hz clock. These represent different tones that may be played from the audio output. Additionally an 8Hz clock acts as the metronome, giving us a consistent 120 beats per minute (split into 16th notes). Finally the 250Hz clock is used to update the 7-segment display.

Each of the four tracks represents a different drum kit component, and as such plays at different audio frequencies: 200Hz, 350Hz, 500Hz, and 1000Hz. We calculated the average frequencies for every possible combination of these notes, which can be seen in the aforementioned 16 different clock signals.

The five push-buttons use debouncing to ensure that the button press is registered consistently every time. We used a clock-enable signal synced with the 250Hz clock to implement the debouncing logic.

III. Simulation Documentation

Our 8Hz clock, switches, and tone frequency are shown in the simulation waveform below. The timings in the waveform were adjusted in order to make simulation faster but the relative scale remains the same.

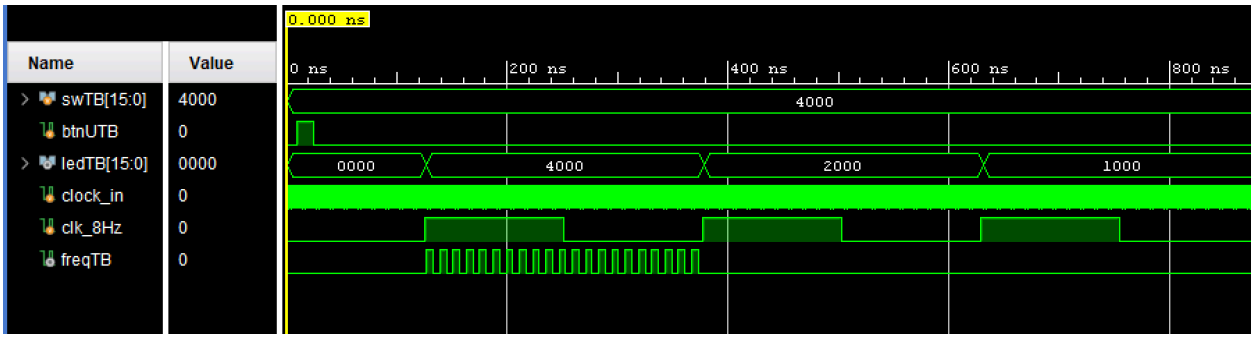

Fig. 3: Recording a 200 Hz tone on the second beat.

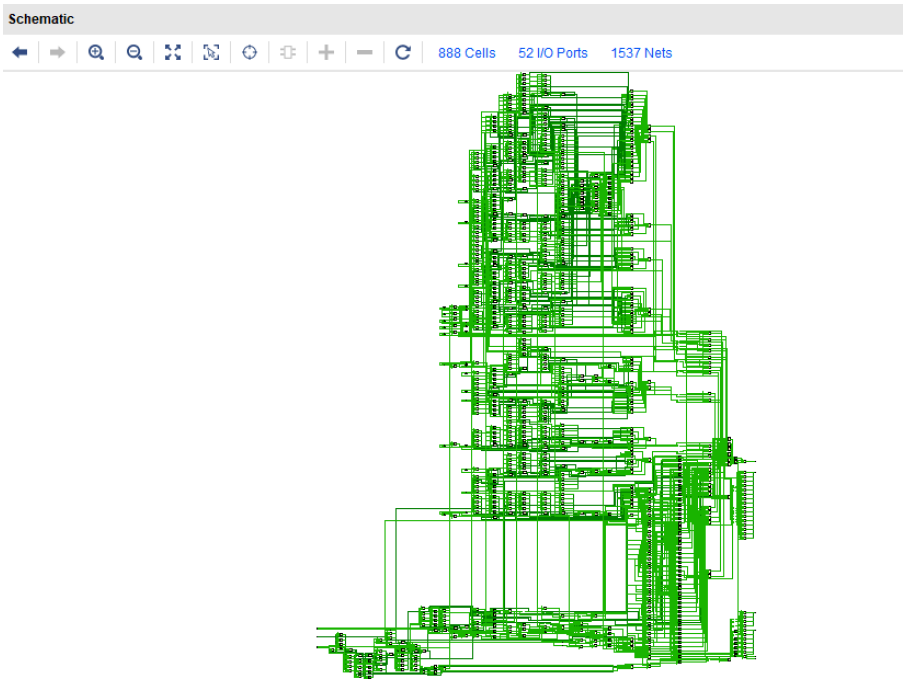

Fig. 4: Synthesized schematic in Vivado.

IV. Conclusion

Our design uses sequential logic to implement the drum machine. Our biggest hurdle was getting the AMP2 module up and running. We experimented with a couple different libraries at first to test how tones could be generated. Although our original plan was to use audio files stored in ROM, we reduced our scope to using PWM-based tones at set frequencies. Most of our testing was actually done without the module, by directly connecting a small piezo buzzer component to the Basys-3 using the same frequency pin that the AMP2 module would connect to. Our suggestions for future improvements to the lab would be to have more AMP2 boards on hand, and perhaps having some vetted example libraries for each PMOD board to help kick-start development.