Lab 3: Keypad Scanner

Peter Johnson[1]

E155 Fall 2018 Section 2, September 27, 2018

***Abstract*— This paper describes the, circuit design, programming and testing of the MuddPi Mark IV such that it can read a 4-by-4 matrix keypad while avoiding bounce. The last two key presses are displayed on the dual 7-segment display with the most recent entry on the right.**

I. Introduction

In this lab, the MuddPi Mark IV utility board was wired to the rows and columns of a 4-by-4 matrix keypad. Once wiring was complete, synchronous sequential SystemVerilog was written to scan the keypad for button presses and to pass these values to two registers that would store the values for the dual 7-segment display which would display the hexadecimal numbers conveyed To achieve the dual display, the same time multiplexing setup from Lab 2 was used.

II. Design and Testing Methodology

I approached the design of this system by first considering the requirements of the the matrix keypad since I planned on keeping the existing time-multiplexed display system. I began by confirming that continuity between a row and a column pin occured when the appropriate button was pressed. I decided to make a scanner module which would send out pulses to the rows of the keypad and then check the which of the column pins were high to see if there was a connection. Part of this also required tying the column wires to ground to ensure they did not float when there was no row input. For the scanner, I implemented a Mealy FSM with 4 states. Each state corresponded to a specific row being turned on. If there were no columns on it would move on to the next row (next state), but if any column was on, then it would remain in the same state with the same row on. This takes care of the button being held down for a long time. In this state, the output (a 4-bit hexadecimal representation of the key pressed) is determined and then passed on. To address the potential bounce of the keypress, I lowered the clock speed of my scanner FSM. At a higher clock speed individual bounces could be registered as separate presses, but if the clock was too low then some presses might be skipped. Knowing this, I decided to estimate that given to body's nervous system, a button would not be pressed more than 10 times a second (10Hz). Just to be safe, I wanted to sample at over an order of magnitude so I chose 150 Hz as my clock speed and implemented an idiomatic 32 bit counter to toggle at that rate.

$$150\ Hz = 40\ MHz * p\ /\ 2^{32}$$

$$p \approx 16106$$

Once the scanner was successfully reading inputs the next step was to remember these inputs. For this I used 2 enable registers since I only wanted to write a new value when a key had been pressed. The enable bit was originally a bitwise OR of the column inputs just to see whether a new button had been pressed, but I saw that if a key was pressed long enough, that its value would pass through both of the registers. To fix this, I implemented a Level-to-Pulse Converter in an FSM. This changed the keypress from a step function into a pulse for a single clock cycle. This meant that even if a button was held down for a long time, the values in the enable registers would only be shifted over once.

My testing methodology was simple and entirely hardware based. I ended up incorporating the on-board LEDs into my testing. LED 7 turns on whenever a key is pressed, and stays on as long as it is pressed. LEDs 3-6 blink whenever the row they correspond to is being powered. I did this testing early on with a lower clock speed in my scanner FSM so I could check that it was actually cycling through each row and that the key pushes were being registered. From there I tried displaying the key pressed on both displays. Unfortunately, I did not need to setup a test bench for debugging.

III. Code and Schematics

Refer to attached schematics and code.

IV. Results and Discussions

All of the prescribed tasks were accomplished: the two 7-segment displays correctly display the last two values from the keypad and when a new key is pressed it appears in the right display and the old value moves to the left. Each key press is recorded once regardless of the length of the press. If multiple keys are pressed, the key pressed first will be registered. The other will not.

V. Conclusions

The lab was successfully completed. The Mudd Pi Mark IV and keypad circuit was fully assembled and tested. The 7-segment displays are able to correctly display the hexadecimal values based on inputs from the keypad. Once the board is reset or turned on and off, the LEDs continue to function, showing the PROM was successfully programed. I gained experience with synchronous sequential design, especially FSMs and careful planning of my implementation. In total I spent 12 hours.

[1]Peter Johnson is with Department of Engineering Harvey Mudd College, Claremont, CA 91711 pjohnson@hmc.edu