DSF Final Project Report

Bradley Fisher, Ronald Garcia, Cameron Kolisko, Alan Zheng
Professor Buccafusca
5/15/2024

Introduction

For our final project we made a digital keypad. The circuit is a 4-digit keypad that allows the user to click buttons corresponding to the digits 0 through 9 one at a time, and a confirm button that is held to validate input. The correct passcode is 7124. This passcode was chosen at random, and is hard coded into the hardware of the circuit. There are 4 yellow LED lights that light up one at a time as the user enters in numbers. When the user hits confirm, if the correct passcode was entered, all yellow lights will turn off, and a green light will turn on. Otherwise, all yellow lights will turn off, and a red light will turn on. In either case, after pressing a button other than the confirm button, the green/red LED will turn off and the circuit will return to its original state.

Methods

The state machine diagram made in Latex and a link to the state table are both in the appendix.

We have 11 states (y0-y10), 11 inputs (buttons 0-9 and confirm), and 6 outputs (4 yellow LEDs, 1 green LED, and 1 red LED). The first state, y0, is the start/initial state. States y3-y6 are G0 to G3, with G0 representing the first digit being entered correctly, G1 representing the first and second digits being entered correctly, and so on. States y7-y10 are R0 to R3, with R0 representing an incorrect digit being the first digit entered, R1 representing at least one of two digits being entered incorrectly, and so on. Both R0 to R3 and G0 to G3 have outputs of Y_1 to Y_4, which correspond to the number of yellow lights that light up, which represents how many digits have been entered. Lastly, states C and F are represented by y1 and y2 respectively. y1 represents the PIN being entered correctly and confirmed, while y2 represents the PIN being entered incorrectly and confirmed. If all 4 digits are correct and confirm is the next input, the green light will light up. If in state y1 or y2, hitting confirm will keep the FSM in the same state. Otherwise,

hitting confirm will send the FSM to y2, the fail state, and the red light will light up. Hitting any button other than confirm while in states C or F will bring the FSM back to the start state.

The 11 inputs are buttons for the digits 0-9 and a confirm button. A design choice we made was designating another "input" called None, which represents no buttons being pressed. This greatly simplified the logic, as all 11 buttons wouldn't need consideration for all 11 states to handle no buttons being pressed. The states are all one hot encoded. As a result, we have one D flip-flop per state, and the circuit for each D flip-flop is the state transition equation (the combination of current states and the inputs that has y_i as the next state). This makes the process of building the circuit significantly more straightforward as we can build each states' next state logic independently of the others. This allows for modularization and division of labor. There were some chips that we needed but were not supplied: the chips for 3, 4, and 5 input OR gates and a chip for a 5 input NOR gate. Since we had access to chips with 2 input OR gates, 3 input NOR gates, and NOT gates, we were able to use these chips to make them. For example, the 3 input OR was made with a 2 input OR, with one of its inputs being the output of another 2 input OR.

To clear the state of the circuit, we used D-type flip-flops with a clear, however we did need one preset D flip-flop which was not available. To get around this, we used a D-type flip-flop with clear, but added a NOT before the D input, and a NOT directly after the Q output. This way, when we set clear for that flip flop, because we are NOT-ing the output, the low will become a high. Additionally since we are NOT-ing both the input and the output, the 2 NOTs cancel each other out and give the expected value for our circuit.

While debugging, we ensured that each of the 11 states had the correct output as we stepped through the current states, with only three real sequences of states to check: all correct inputs and then hitting confirm (7->1->2->4->confirm), all wrong inputs and then hitting confirm, and a partially correct sequence and then hitting confirm (such as 7->1->7->5->confirm). After that, we made sure the outputs, our LEDs, were correct. With an oscilloscope following each state, the process of debugging was made significantly easier.

Using the 555 timer chip involved finding a circuit online, as well as an equation for the frequency ($f = \frac{1}{1.1 \times R \times C}$). We had to make sure not to choose a frequency value that would cause two clock rises to appear when pressing a button normally, while also not choosing a

frequency too low that would make it take a long time to input a passcode. We decided that once every two seconds was an ideal frequency to use (0.5 Hz). We used a resistor in series with a potentiometer that has a maximum of 100kΩ, a resistor of 100kΩ, and a capacitor in series (100 μF and 10μF) with an equivalent capacitance of 9.09μF. At its maximum, it produces a frequency of 0.5 Hz. Barring the expected errors in the resistor values and capacitance values, this was close to the actual result of the 555 timer chip, with a frequency of 0.592 Hz.

Self Assessment

Alan: Helped with the state table. Wrote the finite state machine into a latex diagram. Helped wire the circuit board. Helped debug the circuit board. Taped up the pizza box :) . Made a first draft of the final report.

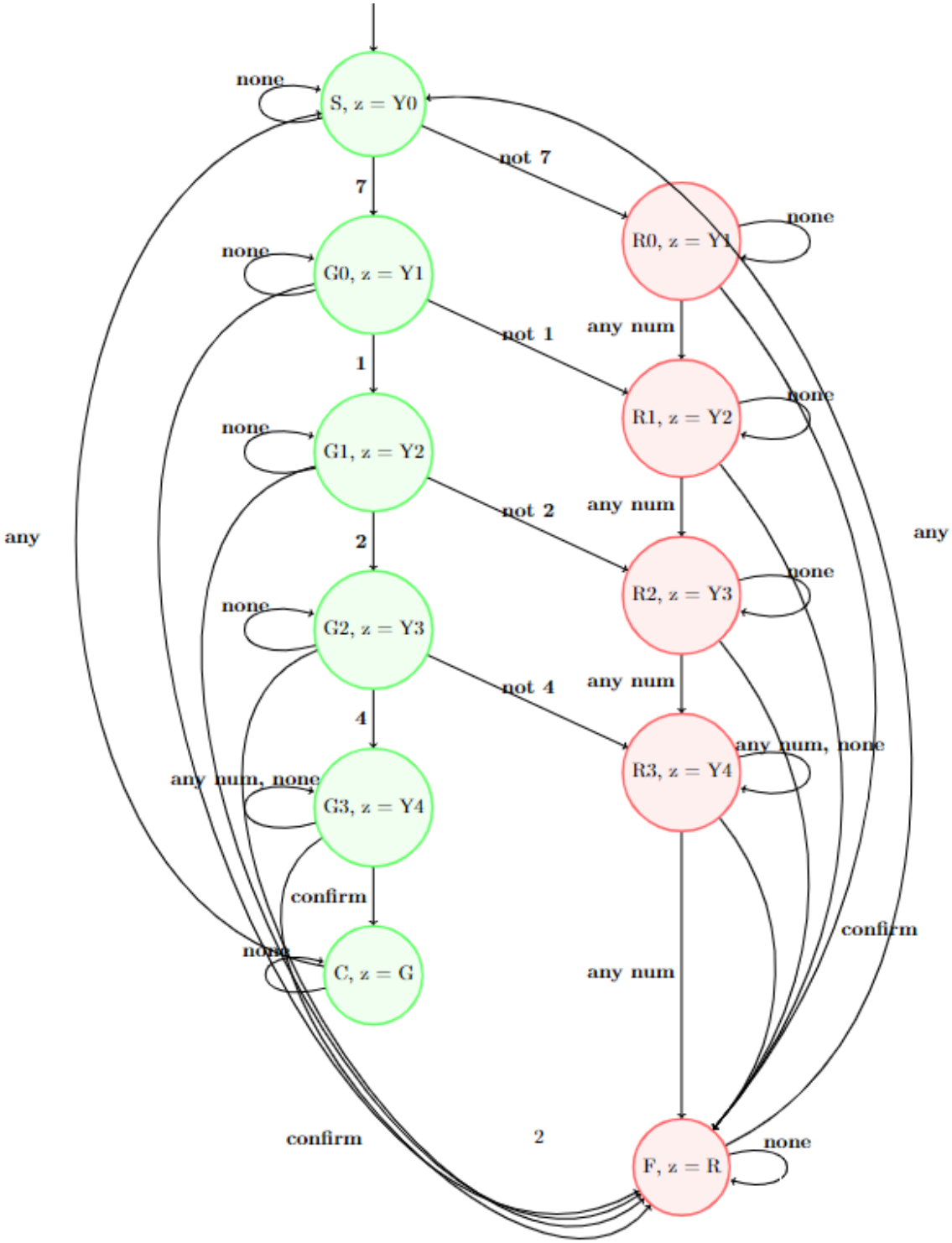
Cameron: Helped with the state table. Helped create the finite state machine. Designed the circuit in Logisim. Helped wire the circuit board. Helped debug the circuit board. Helped with the final report.

Bradley: Helped with the state table. Helped create the finite state machine. Helped wire the circuit board. Helped debug the circuit board. Helped with the final report.

Ronald: Helped with the state table. Helped create the finite state machine. Helped with the circuit board. Helped debug the circuit board (did a lot of heavy lifting, like showing us how to use the oscilloscope). Helped with the final report.

Images of Final Product/Appendix

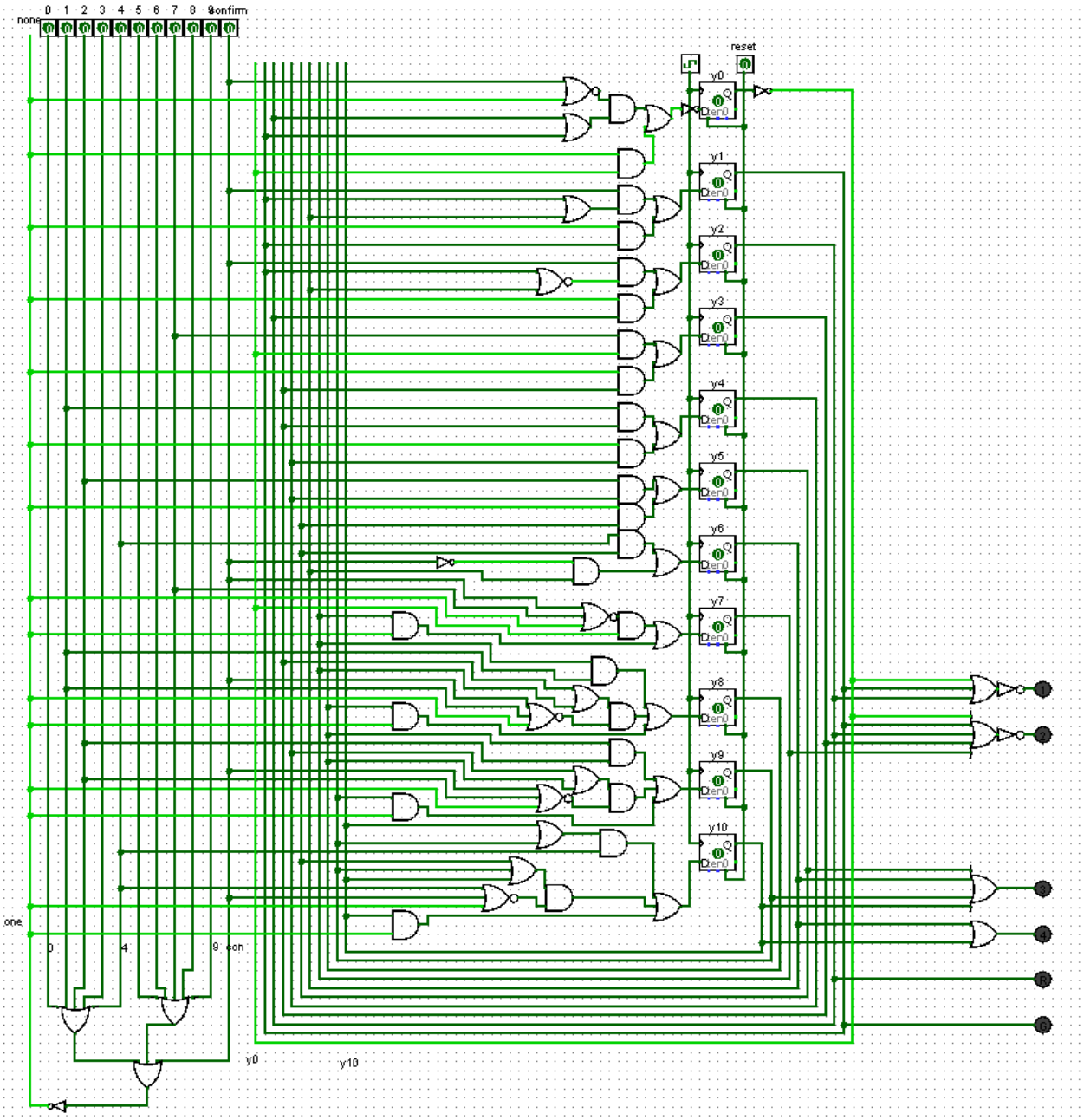
Finite State Machine Diagram:



State Table:

https://docs.google.com/spreadsheets/d/1gdQp_bbRl-nZo3NcUq_9YO8KWArqpqTjD1XsxYIlp7M/edit#gid=0

Circuit in Logisim:



Physical Circuit: