Krutik Shah, Logan Greif

Introduction to Digital Systems – Final Project Report

Introduction

Combination locks are very useful for keeping valuable items safe and secure. For a general purpose, mechanical combination lock, you can open it by turning the knob to the correct combination of numbers. Similarly, for our final project, we have 3 number combination lock that can be accessed by turning the knob on the rotary encoder to the correct numbers. This project fulfills the need for security for valuable items.

Specifications

The inputs for this project are as follows:

- Rotary encoder GPIO[2..0]
    - Clock
    - Data
    - SW (push button)
- Switches on DE10: SW[9..0]

The rotary encoder has a dial. Turn the dial to change the number on the HEX0 display. Press the dial to store that number and then choose the next number. The switches on the DE10 determine the correct combination of numbers, in binary. Because the third number only has 2 switches at the end of the DE10 (SW[9..8]) it can only go up to the number 3, with SW[11..10] wired to ground.

The outputs for this project are as follows:

- HEX0[7..0]
    - This is for displaying the input number. It is in decimal value. Rotate the encoder dial CW to count up, and CCW to count down the number on this display.
- HEX3[6..0]
    - Displays the 3rd stored number
- HEX4[6..0]
    - Displays the 2nd stored number
- HEX5[6..0]
    - Displays the 1st stored number
- HEX2[7]
    - Decimal point on HEX2 display, if it is on it means you have unlocked the combination lock
- LEDR0, LEDR1, LEDR2
    - Shows what state you are in, in binary. Fourth state is HEX2 decimal point, if the device is unlocked

We also have an expansion module that plugs into the GPIO pins of the DE10. This module has a debouncer (Schmidt trigger IC) for the encoder, 3 RGB LEDs (that didn't end up working, were supposed to replace the LEDR and HEX2 functions), a HEX display (which didn't end up working either, was meant to replace the HEX0 functions), and the encoder itself. Realistically, this expansion module would be on the outside of the combination lock encasing and the DE10 would be on the inside.

<u>Design Discussion</u>

There are four states in this state machine. The first state is the first number, the second is the second number, the third is the third number, and the fourth state is the unlock/lock state. There is a state counter (driven by the pushbutton) that counts from each state. Once it reaches the fourth state, the counter resets and goes back to the initial state once the button is pushed again.

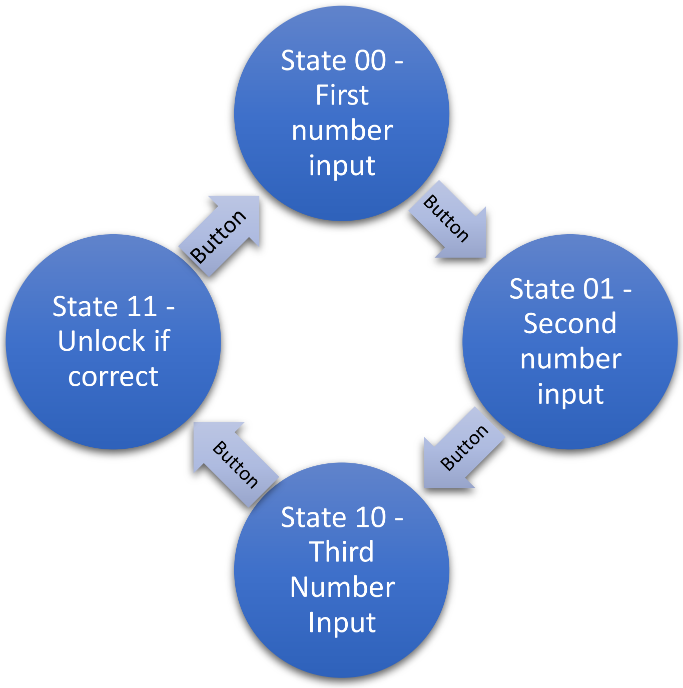
State 00 - First number input
Button
State 01 - Second number input
Button
State 10 - Third Number Input
Button
State 11 - Unlock if correct
Button

Figure 1: State Machine Diagram

We are using the switches on the DE10 to determine the correct combination, in binary. This is compared to the stored input values from all of the DFF modules, with a XOR module, in the fourth state. If these values match based on the XOR module (the comparator, in Figure 3), then HEX2[7] lights up. This is shown below in Figure 2.

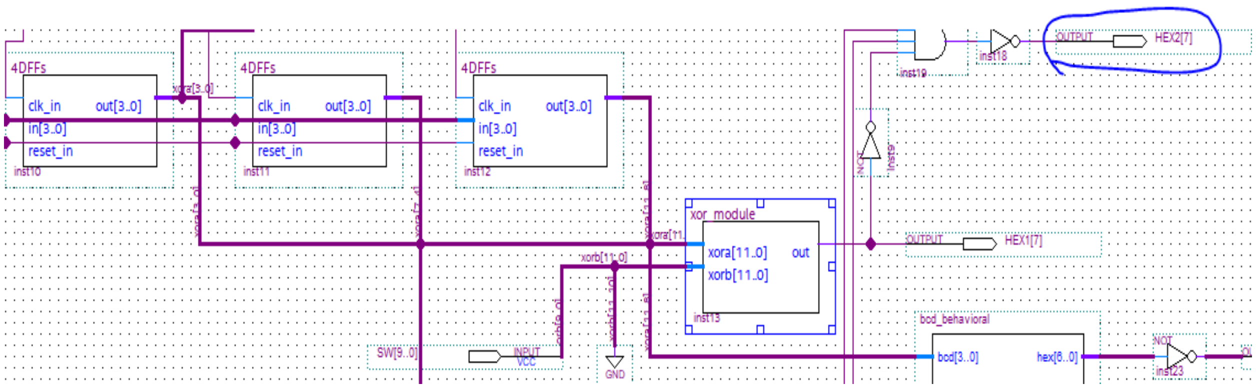

Figure 2: SW[9..0] are compared to stored values which determines whether HEX2[7] lights up. Note: HEX2 is NANDed with the outputs from the 2-bit state counter (so that it will only work during the final state of the state machine) and the inverse of the XOR module/comparator

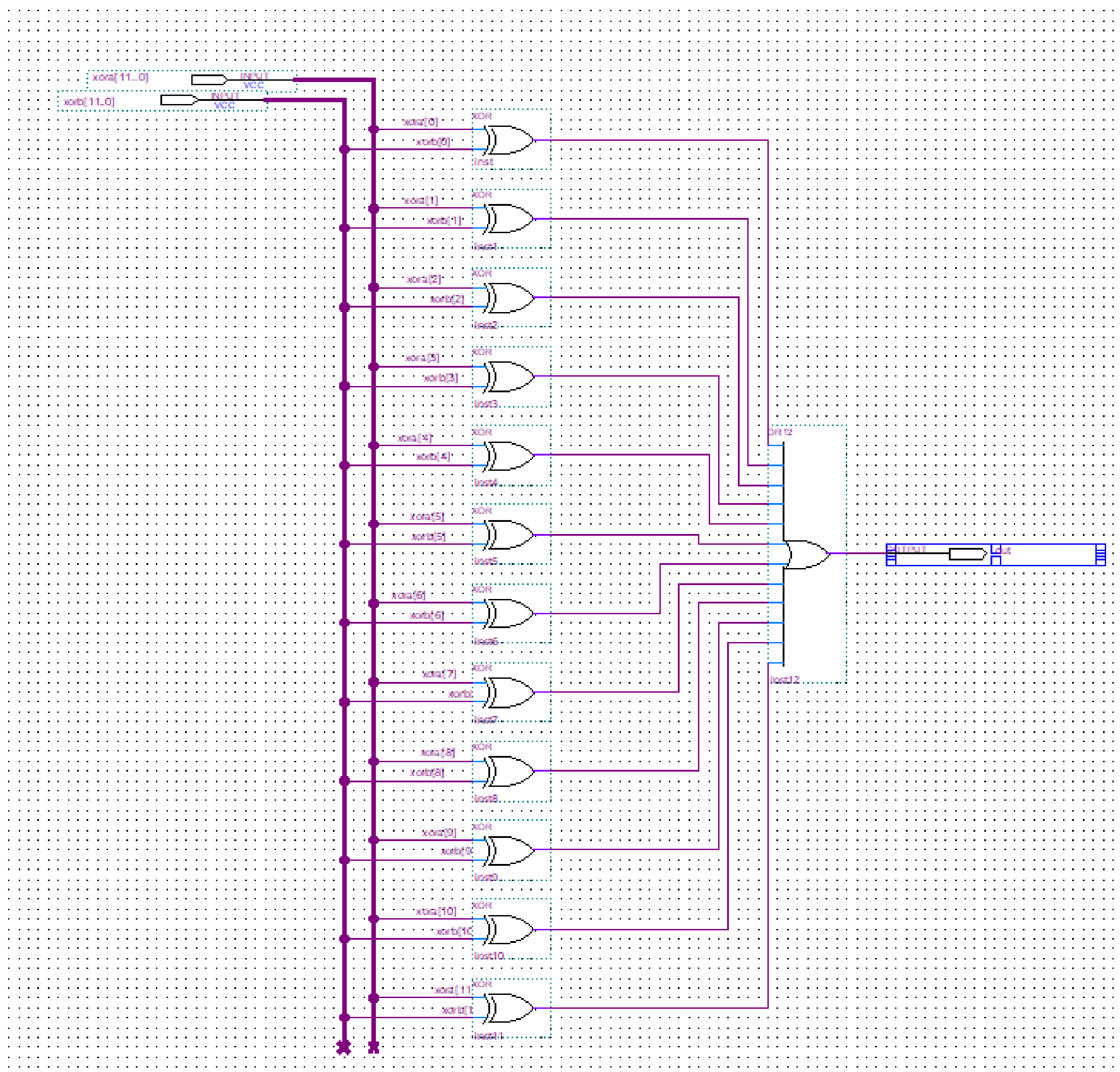

Figure 3: XOR module/comparator

We used BCD to 7-segment modules for displaying to the HEX displays. The BCD-to-7-seg connected to HEX0 is from a premade module from the Quartus library, as shown in figure 4 below, while HEX3, HEX4, and HEX5 are based on the behavioral code from Lab 3B.

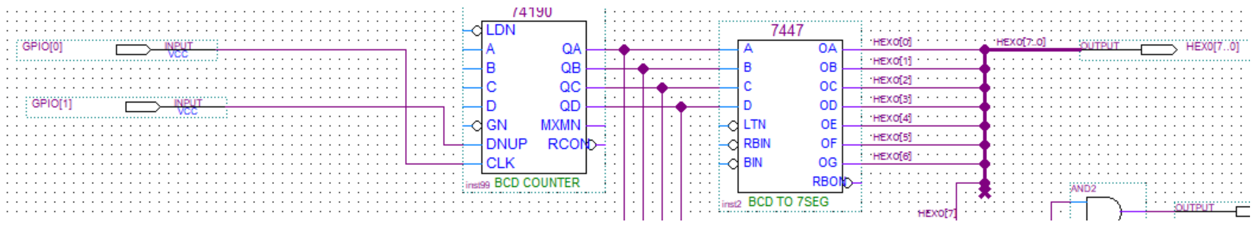

Figure 4: The BCD Counter wired to BCD to 7-Segment module which displays the number on HEX0

We also use LEDR, LEDR1, and LEDR2 to display the state you are in. For example, if you are in state 1, LEDR0 will light up. If you are in state 2, LEDR1 lights up, if you are in state 3, LEDR2 lights up. In state 4, HEX2[7] will light up if unlocked. This is based on combinational logic from the state counter in the top-level file, as shown in Figure 5.

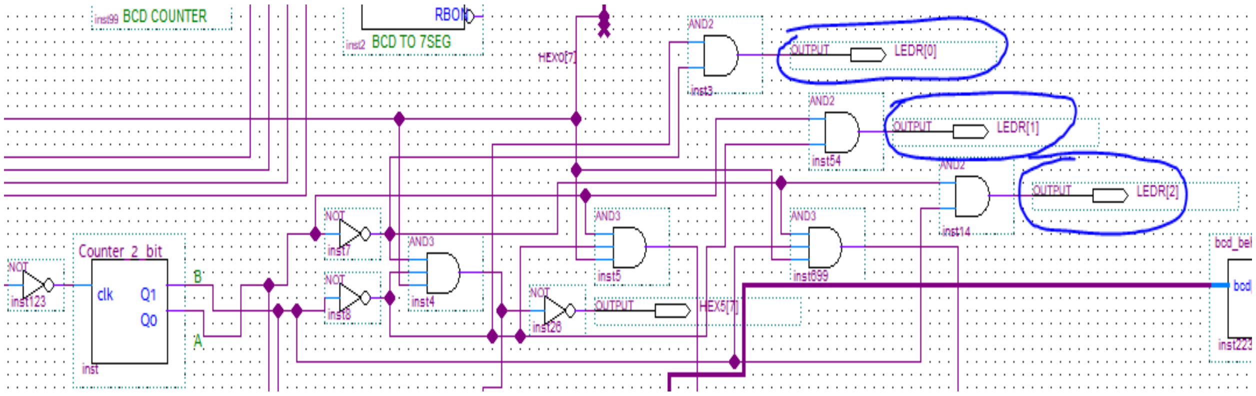

Figure 5: Combinational Logic for State Counter

We also have 3 registers to store values in each of the three states. Each register is 4 bits so 4 D-Flip-Flops in each register module, as shown in Figure 6. Each of these 4 bits in each register are wired to the XOR module to be compared to the switches on the DE10.

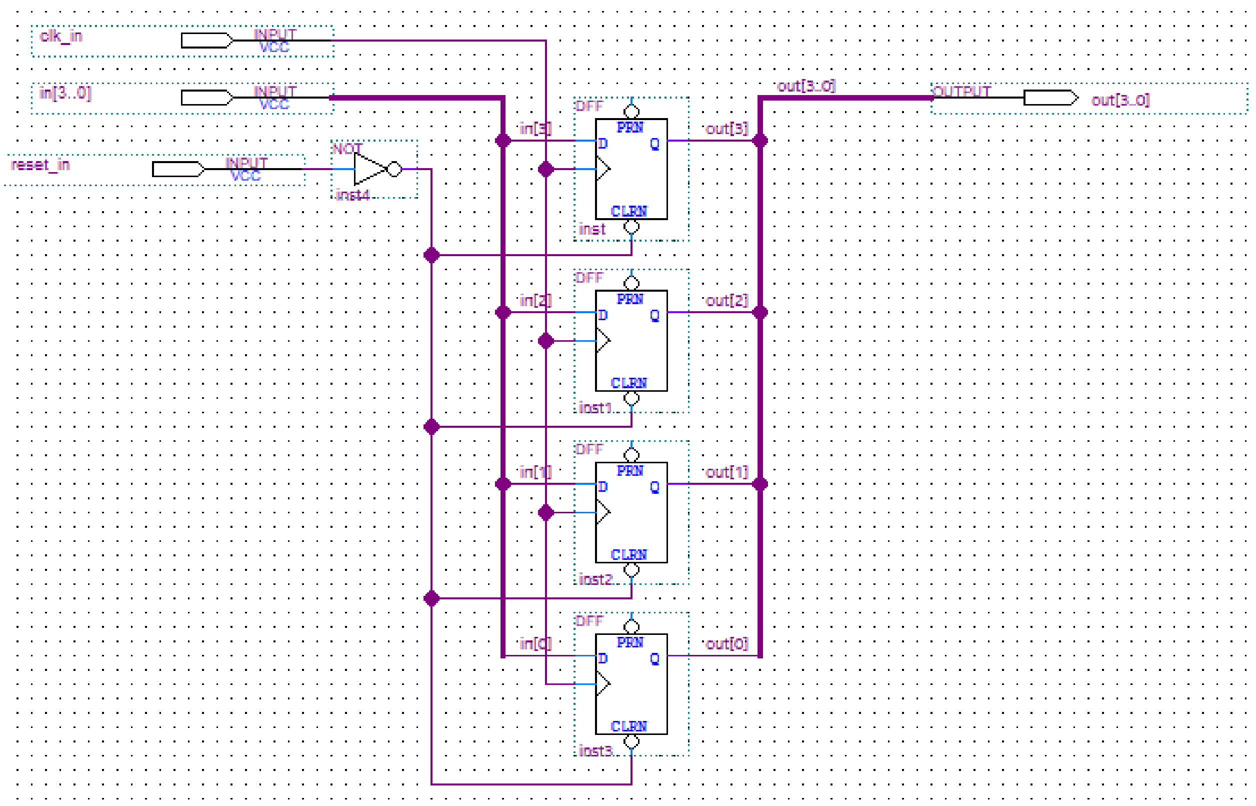

Figure 6: Configuration of registers

Results and Conclusion

This project turned out well because almost everything worked in the end. During the process of debugging, we had to wire some things to certain outputs, such as HEX5[7] and HEX1[7] to see what the output of that certain part was, which is why you might see those decimal points light up in the video. The thing that didn't work as well was some of the parts on the expansion module. We never got to using the RGB LEDs, or the HEX display properly. We managed to just deal with the stuff on the DE10.

Here is a video of the project demo

Here is the project folder

Logan designed the schematic and what it should look like, the XOR module, and built the expansion module, and much of the work in top level file.

Krutik created the state counter, the registers, did some debugging in the top level and this project report.