Four Digit Decimal Locking System

Nexys A7

Foster Caragay, Marko Wassef, Vincent Tran, Athanathios Bebawy
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: facaragay@oakland.edu, markowassef@oakland.edu, vincenttran@oakland.edu, abebawy@oakland.edu

***Abstract*—This paper details the process of developing and implementing a 4-digit locking system with decimal input. This system utilizes 8 4-bit registers, 2 decoders, an FSM, 7-segment decoder, a comparator, and a debouncer/decoder. The system contains both datapath and control circuitry implemented together on a Xilinx xc7a50t csg324.**

I. Introduction

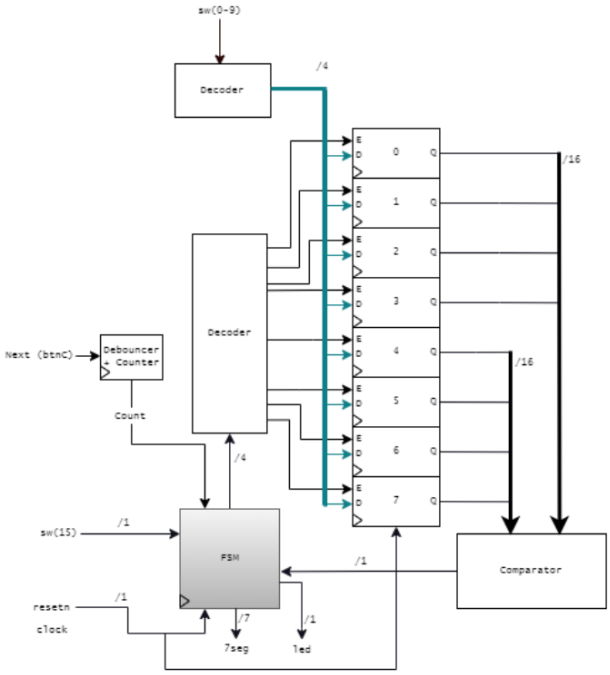

Figure 1. Block Diagram of Locking System

Our circuit resembles the logic of digital safes. A combination is set, and to unlock the system the same combination must be entered. The option to reset the combination is available if the system is unlocked, or the system can be re-locked. The option to set a new combination or re-lock the system is selected by the select switch (sw15). The combination is selected by putting a switch high (with an assigned decimal value) and selecting the "next" button (btnC). As the 4-digit combination is entered the 7-segment display will show the index of which digit in the combination the user is setting. Ex: if the user is setting the third digit of the combination, the 7-segment will show a 3. Once a combination has been set, the system awaits an input from the user.

The input process for unlocking the system is the same as that used to set the combination. Upon entering the correct combination, the system will illuminate a LED (led0) and display a U on the 7-segment display. If the wrong combination is entered, the system will display an E and at the next button press, allow the user to enter a combination again.

II. Methodology

A. *Decimal to Binary Decoder*

Designing the means to input decimal digits into a system that operates in binary posed challenges. The solution to this issue was to assign 10 switches decimal values 0-9, that were then decoded by a decoder component. The decoder component has a 10-bit input, with a 4-bit output. The 10-bit input is comprised of the high/low state of the 10 switches representing the decimal input. Ex; "0000000100" would represent a decimal value of 2. In cases where multiple switches have high states, the decoder will output a "0000" as no conclusion can be met. The decoder utilizes a LUT that has entries for all expected values, and for the unexpected values outputs a "0000".
Example:

Input: "000000**1**00" = Output: "0010"
Input: "987654**3**10" = Output: "2"

B. *Memory/Registers*

The inherent nature of a lock means that a combination must be set and stored into "memory". First, the use of parallel access left shift registers was considered to allow for the data bits (set combination and entered combination) to be shifted in and read via the parallel access. However, this design was not the most optimized. Reviewing constraints for the design it occurred that the logic structure employed by RAM would be the most efficient, and work effectively. Thus 8 4-bit shift registers were used to store a total of 32-bits of data. The 4-bit data line fed from the Decimal to Binary Decoder is tied to all 8 registers, thus an enable decoder must be implemented. The enable decoder has a 4-

bit input and an 8-bit output. The 4-bit input is fed from the FSM control system to enable the appropriate register. The 8- bit output feeds one bit to the enable input of the appropriate register. Ex: to enable register 3, the FSM would input the 4-bit data “0011”, and the decoder would output “00100000”. To allow functionality to disable all registers, the decoder will recognize the input “1111” and disable all enable outputs to prevent overwriting of values in states where the registers are not utilized.

C. Comparator

To determine whether the entered combination matches the set combination a simple comparator was implemented. The comparator has a 1-bit output and two 16-bit inputs comprised of the concatenation of the binary values stored in registers 0-3 (set combination) and registers 4-7 (entered combination). The comparator outputs a “1” when both 16-bit inputs match, otherwise the comparator outputs a “0”.

D. Debouncer/Counter

As the system utilizes a button as a means of user input, a debouncing circuit was developed and implemented. The debouncing circuit utilized a pulse counter that counts the rising edge of X amount of clock pulses. If the button signal remains high for X amount of clock pulses, the 4-bit counter will increase by a value of one. The 4-bit counter is used to track the state of the system and is fed to the FSM for use elsewhere in the system. The count has a maximum binary value of “1011” representing the last user action before returning to a count of “0000”. This is best displayed by figure 2.

E. FSM/Control System

The FSM/Control system is used to track the state of the system, control which register is enabled, the state of the LED, and the 7-segment display. Below is a diagram of the FSM’s logic (figure 2).

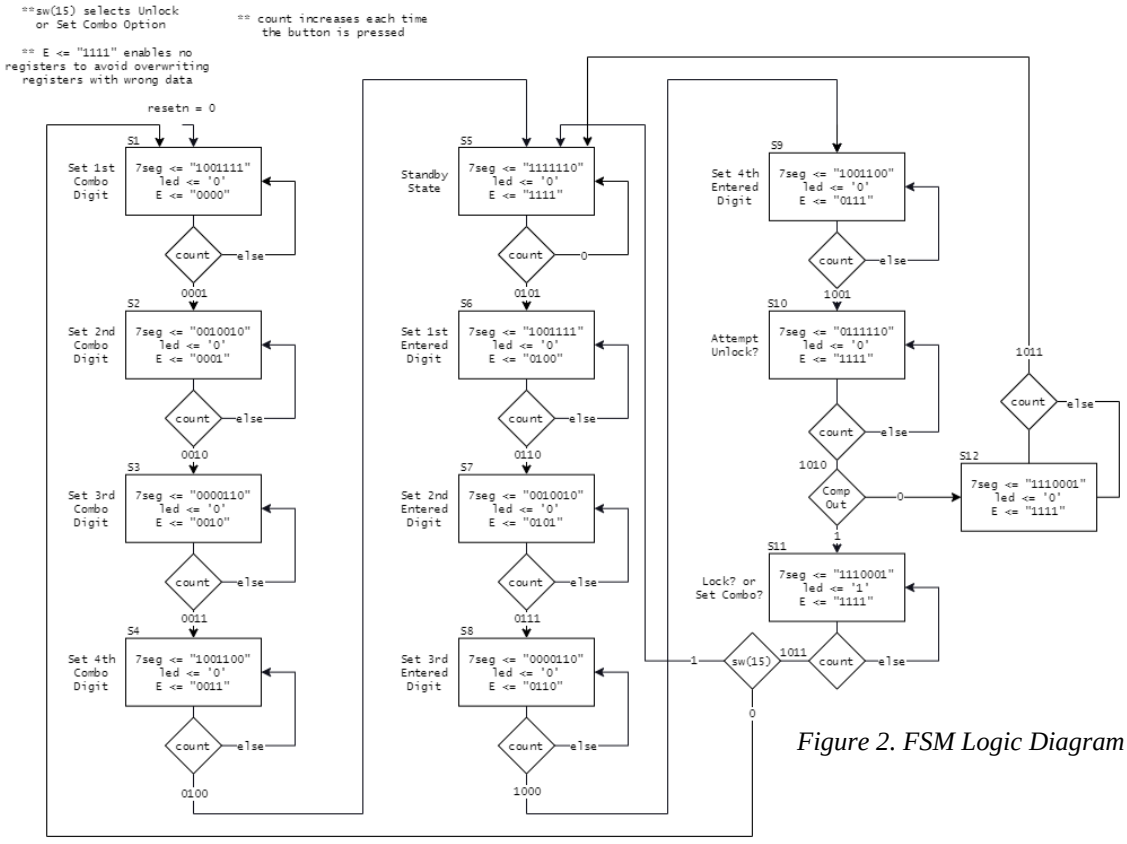

Figure 2. FSM Logic Diagram

III. Experimental Setup

In order to verify and test each component used in the system, separate test benches were used. However, due to the nature of the system generating a testbench to test every possible combination of set and entered combinations proved infeasible and inefficient. To test the entire system, we generated a bit stream and using a multitude of edge-cases and test values we were able to verify the functionality for 20+ set and entered combinations.

IV. Considerations/Conclusions

For the duration of the project, the biggest issue experienced was the use of a button for user input. The debouncing portion of the circuit is where the most uncertainty is presented as with each button press, the mechanical components of the button deteriorate and thus cause more bouncing. Additionally, the implementation of a memory structure that utilized the least amount of component proved challenging. However, the use of a similar structure to that of RAM was a solution that met our constraints and works without flaw.

References

[1] D. Llamocca, "VHDLforFPGAs," VHDL coding for fpgas. [Online]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html. [Accessed: 11-Nov-2022].