+5 Digit Combinational Safe

(Karama Alkeilani, Houda Almoosawi, Kyle Budzynowski, Derek Smith)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: kalkeilani@oakland.edu, almoosawi@oakland.edu, kylebudzynowski@oakland.edu, smith41@oakland.edu

***Abstract*–This project will delve into the process of designing a 5 Digit Combination Safe through the Nexys A7-100T board using the Vivado Software. The project aimed to create a functional and streamlined process for the digital lock through the use of several key components, including: User input, circuit output, memory, validation, and FSM. through these key components, the user's choices will appropriately reflect whether the safe is unlocked - or not. Major findings from this project include the successful integration of several components within the circuit, such as a parallel shift register, a 5-bit RAM emulator, and a counter, in order to enhance the circuit's versatility as a digital safe. Another finding was the key role the FSM played in managing the lock's functionality as it served to both validate and input a new code into the circuit's program. Some challenges that were met during the project were VHDL based, as well as debugging errors. Conclusions drawn from the project include the underscoring of the complexity of FSM design,which prompted the use of a more well defined state diagram. As well as this was the success of the circuit performing all of its project requirements defined in the project guidelines. Some future recommendations will be to address security concerns for the digital safe in real-world use, such as implementing additional safety measures for the safe.**

I. INTRODUCTION

The report details the creation of a 5-digit combination lock, capable of being programmed with distinctive number sequences and configured to transmit an unlock signal solely upon inputting the correct code through the interface.

Crafting a control mechanism of this nature serves as an educational tool, enriching comprehension of Datapath circuitry's interplay with state machines to forge a digital system. The foundational framework for such a circuit boasts a wide application spectrum and can be integrated relatively effortlessly. Notably, this lock circuit finds utility across multiple production domains, finding usage in automobiles, home safes, and various other applications.

II. METHODOLOGY.

A. Initial Design

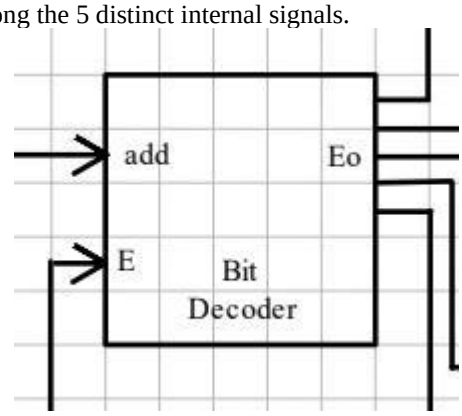
The first step in completing the design phase of the combination lock was to define what the final output needed to be and the path that was required to get to this point. Taking into consideration that the circuit at minimum would require a sequential 5 digit input and have the ability to compare that input to a stored value. Upon the comparison of that signal define whether it was correct or incorrect then, visually represent the information to the user. The circuit was then broken down into 5 main sections, those sections were Input, Output, Memory, Validation, and FSM. In the following sections the development of these sections will be explained.

Within this circuit, several parametric components were employed, strategically designed with generic mapping to facilitate circuit construction [1]. These crucial components include the shift register, the registers integral to constructing the RAM emulator, and the counter. These parametric elements were deliberately selected for their versatile mapping, contributing significantly to the circuit's overall construction and functionality.

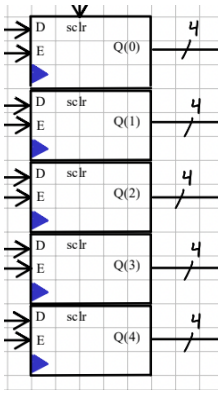
B. Memory

Starting from the core and focusing on storing the combination formed the initial phase of the process. This step was relatively straightforward, leveraging the construction of a RAM emulator based on the knowledge acquired from Lab 5. Opting to modify and utilize this RAM emulator initiated the creation of a block diagram. Key components necessary for the RAM included a Decoder, individual registers allocated for each memory location, and a multiplexer.

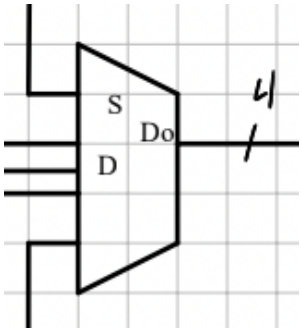
The Decoder's primary function involved acting as an enable selector for the individual registers. With a sole output of a 5-bit signal, this output was distributed to each of the 5 register locations, serving as an enable signal for each specific register. Additionally, the Decoder featured 2 inputs: an enable signal that triggered the internal circuit operations when receiving a '1', and an address—a 3-bit input used to select among the 5 distinct internal signals.



Moving on to the registers, these components possessed a single output and three inputs, one of which was a clock signal. The other two inputs comprised data in (a 4-bit hexadecimal digit) and an enable signal akin to the one used in the decoder. Each of the 5 registers was designated to store a single digit of the combination.



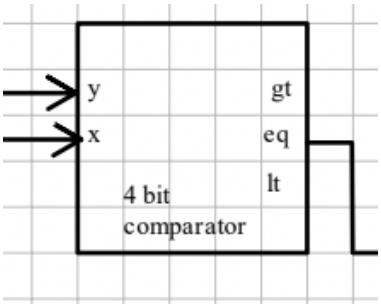
The multiplexer played a crucial role by receiving outputs from the registers and outputting the accurate register location data based on the identical address input as that of the decoder.



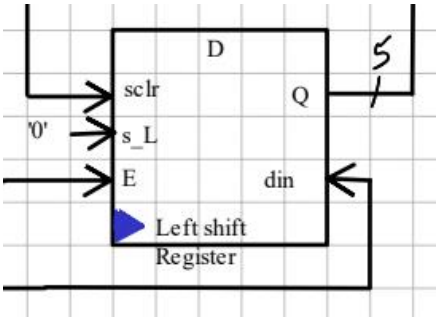
As the entire circuit progressed in development, a pivotal decision was made to integrate a second RAM emulator into the project. This addition became imperative for storing both the preset combination and the attempted combinations separately, enabling distinct storage locations for each within the system.

C. Validation

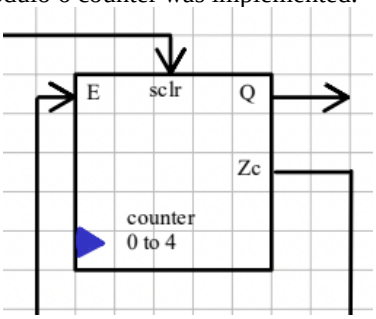
Designing a system to verify the entered combination against the stored one posed the most intricate challenge in the entire process. Researching potential methods led to the discovery of an online lecture elucidating the functionalities of a 4-bit comparator [2]. This finding became pivotal as the comparator proved to be an optimal solution, efficiently handling two 4-bit inputs and decisively determining whether the first input was greater than, less than, or equal to the second.



Embedded within the validation circuit lay a specialized left shift register, meticulously tailored for this specific purpose. This intricate register played a vital role by accumulating all instances of "equal to" signals, regardless of their Boolean representation—whether '1' for true or '0' for false. It thoroughly orchestrated the compilation of these signals into a cohesive single 5-bit output, affirming the legitimacy and accuracy of the entered combination. This process operated in a continuous loop, cyclically integrating the "equal to" signal from the comparator during each clock cycle, ensuring a thorough and comprehensive validation mechanism.



In order to signal the completion of the left shift register's output, and thereby ascertain the validity of the combination, a Modulo 5 counter was integrated into the circuit. Initially, the assumption was that 5 shifted-in bits would denote the conclusion of the computation process. However, through rigorous behavioral testing, it was unveiled that an additional shift-in cycle was imperative to achieve a fully valid output therefore a Modulo 6 counter was implemented.



D. FSM

Following the examination of the partially completed block diagram, the immediate next step involved crafting a Finite State Machine (FSM). Building an FSM necessitated a meticulous state map to define the pertinent inputs and outputs of this machine. This process allowed for a comprehensive understanding of the required inputs to control the circuit effectively.

The approach commenced by reverse-engineering from an output perspective. The FSM's role was to govern the sequential access of the circuit to various address locations, necessitating a means to signal the state machine when the user completed transitioning the switches for each digit. Specifically, 5 switches, labeled EA through EF (SW{14} to SW{10}), were designated for this task.

During the setup code input, these switches needed to be sequentially toggled from '0' to '1', enabling the FSM to progress through individual memory locations. Conversely, when testing a combination, the switches had to transition

consecutively from '1' to '0', signaling the FSM to cycle to the subsequent state.

Several outputs correlated with these operations: Input Enable (IE), Setup Enable (SE), and Address (add). SE activated the decoder for the setup memory location, while IE performed the same for the input memory block. Both enable signals operated exclusively—one activated while the other remained inactive. The add signal governed both memory decoders and multiplexers, dictating the data input and output locations.

Another pivotal input was a signal indicating whether the circuit was storing a new combination in the setup memory or validating a combination to unlock the safe. This signal, denoted as "w," was linked to SW{15}, where '1' indicated writing to the setup memory block and '0' directed the data to the input memory block.

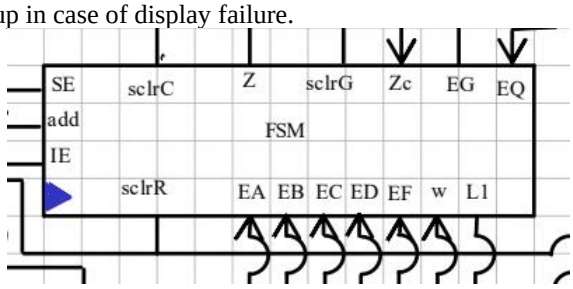
Additional inputs crucial for the FSM included signals denoting a completed count and the full shift register. The ZC signal from the counter marked '1' when reaching its predetermined value, prompting the FSM to check the incoming value from the shift register. If the value equaled "11111," the combination was deemed correct; any other input indicated an invalid combination.

The FSM's remaining inputs were the clock and resetn signals. The clock regulated timing for the FSM to process synchronous information, while the asynchronous resetn signal reverted the state machine back to its initial state.

Outputs associated with this segment of the FSM encompassed the enable signal for the shift register and the counter (EG) and a 7-segment decoder (z). EG, set to '1' upon storing the last digit of the input memory, triggered the counter and initiated the register's compilation of output from the comparator. Upon completion, a 2-bit signal relayed either an 'E' for incorrect or a 'U' for correct to the decoder for display.

The FSM also featured distinct synchronous clear signals for the two memory locations, the shift register, and the counter, enabling individual components to be cleared independently.

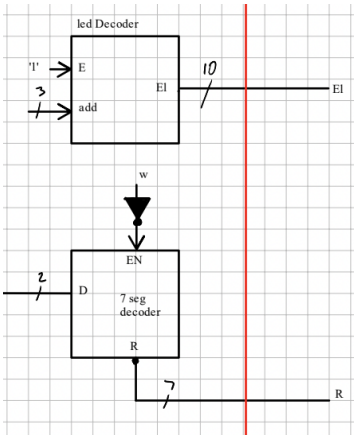
Finally, the FSM culminated in a 3-bit (L1) signal altering the state of an RGB LED on the board. This redundancy was implemented to provide an alternative visual representation of the input, aiding in error debugging and serving as a backup in case of display failure.



E. Inputs/Outputs

Upon finalizing the remaining segments of the Block diagram, the comprehensive set of inputs and outputs for the entire circuit became readily apparent. The inputs encompassed the Data In (DI), constituting the 4-bit hexadecimal input, alongside essential signals such as the clock and resetn for various component functionalities. An external enable signal, linked to "w," and the EA to EF signals were also vital inputs utilized by the FSM.

As for outputs, they included the 7-bit string generated by the 7-segment decoder and the L1 signal, serving as an indicator on the board's LED. Additionally, an extra output was integrated to illuminate 5 of the red LEDs on the board, enhancing the visual representation of the memory location currently being written to. These outputs collectively contributed to a comprehensive view of the circuit's inputs and outputs, ensuring functionality and aiding in visualization.



III. EXPERIMENTAL SETUP

For this project, the Nexys A7 – 100T FPGA Board served as the primary hardware for implementing the 5-digit combination safe, but before we could use the VHDL design on the board we needed to use extensive simulations by implementing a VHDL testbench in Vivado. This testbench was critical to verifying the functionality of each component of the design and this testbench gave us the ability to look deep into the circuit and confirm all signals were being executed as they were intended. This testing primarily focused on the behavior of the finite state machine (FSM) to verify that all states were being passed through at the correct times and giving the correct outputs.

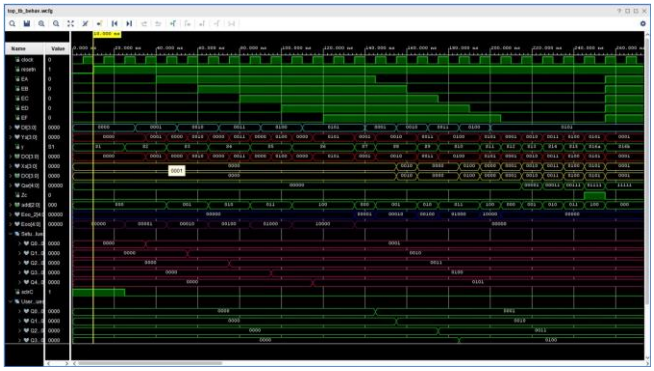
The VHDL testbench was set up in a simulation environment in Vivado and was designed to simulate the inputs that were to be provided by the user when testing the circuit on the FPGA board. More specifically, testing the functionality of the data inputs (DI), the enter switches that simulated each DI being entered into the circuit, and the combination mode and unlocking mode select switch (wr_rd) which would switch between which mode you wanted to enter data in. This switch was also a critical enable switch as it also performed the task of enabling different decoders and displays based on what mode you were in. This simulation allowed us to control what inputs were being toggled on and off to get the exact output we desired.

The most critical aspect of the project was the functionality of the FSM as it was designed to be the control center of the entire circuit and it was crucial to make sure all

inputs and outputs were being controlled and manipulated correctly. By simulating the different states, we could meticulously observe and verify each transition of state based on the inputs provided and verify all outputs were correct to maintain the functionality of the rest of the components.

IV. Results

The results obtained from the VHDL testbench simulation for the 5-Digit combination safe were comprehensive and allowed us to view all our components theoretical outputs in a real scenario. Through the use of a testbench simulation to manipulate inputs to get an excepted result, we were able to visualize the sequential logic and behavior of our system under various test conditions. One of those scenarios is displayed below, showing the inputting of a combination and then attempting to unlock the safe with the same combination which allowed us to visualize the entire system move through all the required states and come to the correct output which in this case was an unlocking of the safe triggering the seven segment display to display the letter U while in that state. There were some minor discrepancies that were found in the simulations but were quickly corrected with there being little to no errors due to the architecture of the components. The results were highly consistent with the theoretical outputs we had expected due to all the knowledge gained during lecture and labs, we were able to successfully implement components such as an FSM or shift register and were able to get the expected outputs due to our understanding of the digital system design and VHDL coding principles. In conclusion, the results as shown below in the testbench and in the implementation of the physical hardware provided clear outcomes for each component and state that was being used in the system and the planning and design done before testing allowed for a smooth and simple process of testing all state transitions and outputs.



Conclusions

The task of creating a 5-digit digital lock prompted the team to work mostly with the logic and VHDL of creating a working FSM and melding all components into a working model via creating a TOP block diagram. The FSM posed the most work in the entirety of the project. Creating a logical and working state diagram was the focal point of the project. An issue that could be worked on is the rather convoluted state diagram created for the FSM logic. Producing a clearer, or rather, a more simple FSM design could be a task that would be worth pursuing to further optimize the lock. Other issues the group ran into were the errors in the VHDL. Simulating the lock, finding the bugs and errors, and then fixing the VHDL and rerouting the signals took up time and was rather tedious. All in all, creating FSM logic and merging all components is the most important part of creating a digital project. Despite the working design, there are still many improvements that could be made in regards to the lock. Security issues regarding the ease of creating a new 5-digit code for the lock could be tackled by added extra logic which would require an additional safety measure to change to code--whether it be a sensor to sense a key, or a factory code, or whatnot. Another improvement could be made by adding an additional seven segment display to show the incoming numbers as the user inputs digits to the lock. Other additions to the lock could be adding a fingerprint sensor to the lock. Many improvements and additions could be done for lock, for further optimization and efficiency. Optimizing the FSM and overall TOP logic design and adding additional features to the digital lock could further improve the design.

References

[1] Llamocca, Daniel. “RECRLab.” VHDL Coding for Fpgas, 2015, www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html.

[2] Haskell, Richard E., and Darrin M. Hanna. “Lesson 36 - VHDL Example 20: 4-Bit Comparator - Procedures.” YouTube, YouTube, 25 Oct. 2012, www.youtube.com/watch?v=3epKbSGbMCA&t=3s