MASTERMIND

Christopher Grzybowski, Naisarg Gohil, Grace O'Georgia, Sindura Gullapelli
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: cjgrzybowski@oakland.edu, ngohil@oakland.edu, gmogeorg@oakland.edu, sgullape@oakland.edu

***Abstract*— The purpose of this project was to simulate a version of the board game, Mastermind, using VHDL. Digital components that were used were registers, RAM, a comparator, a decoder, a counter, debouncer, multiplexer, and a finite state machine. The comparator compares a code with eight guesses, and displays "hints" for each guess made. If the code is guessed correctly, the guesser wins the game. The output was intended to be displayed on an LCD, but was instead displayed on the 7-segment displays and LEDs.**

I. INTRODUCTION

The motivation for this project came form the strategic board game called Mastermind and the concepts covered in the later laboratory assignments from ECE 378. The lab experiments that the project referenced were mainly Labs 4, 5, and 6 [1]. From these labs, random access memory (RAM), finite state machines (FSM), debouncers, counters, among other important concepts, were utilized. The Methodology section of this report will review the original board game and the digital implementation of this model in detail. The Experimental Setup section will discuss how the methodology and the board were brought together to create the project. In the Results section, the final outcome of the project will be revealed. Lastly, the Conclusion will state potential improvements and realizations.

II. METHODOLOGY

The Mastermind game is played with two players: the code maker and the code breaker. The object of the game is for the code breaker to guess the code maker's code without exhausting all of their attempts. A digital version of the original board game was created for this project. The circuit contained many components in order to recreate this simplistic, strategic game.

A. Original Board Game Version

In the original version, the game components consisted of a decoding board, colored code pegs, and red and white key pegs. The decoding board is where the code maker stores their code (in a hidden section), the code breaker places their guesses, and subsequently where the code maker gives feedback to each guess. The code pegs are the used for the code and guesses. The key pegs tell the guesser whether or not their guess is close to the correct code. For example, if the code breaker's guess has a right color in the right place of the code, then that would receive a red key peg. If there is a color in the guess that was in the code, but in the wrong position, it would receive a white key peg. The code maker places the key pegs on the board after each guess. These responses help the guesser make their next guess to crack the code.

B. Digital Version of Game

Instead of colors like in the original board game, the digital version of this uses a 4-digit number with 3 bits per digit. Numbers 1 through 7 can be used, but numbers cannot be used twice. The code is entered into a register with an enable. After the code is entered, it is sent to the comparator, and the guesses are ready to be made. Each guess will also be sent to the comparator, thus comparing each guess to the code. Once compared, there are outputs that symbolize the red and white key pegs of the original board game. The outputs, or "hints", are called RR and RW. RR stands for right number (digit), right position. RW represents a right number, in the wrong position. These outputs are concatenated with each guess to store in a register in RAM. The FSM controls the order of events, and the count of the guess determines which register to write to. The count is shown on a 7-segment display. The output is then an 18-bit number that needs to be split up into an individual guess, and corresponding RR and RW, to display on the LCD. Since a part of the original game is to see the previous guesses, and they are helpful in making the next guess, an FSM was designed to toggle through the RAM to output onto the display.

C. Finite State Machine

The FSM of this digital circuit controlled all of the events happening in the game. In State 1, the count is 0 and E_code is checked. If it is '1', Er is enabled, which writes the code to the code register. Next, State 2 is reached, where E_guess is checked. If it is '1', E_comp is enabled, which compares the current guess input with the saved code. This creates outputs, RR and RW, and the FSM moves to State 3. In State 3, the guess, RR and RW are written to the register using the count as the address. If at this moment, RR is equal to 4 (winner/matching number) or the count reaches 7 (loser/8 incorrect guesses), then the state is returned to State 1. If not, then the count increases by one

and the next guess is ready to be made until there is a winner or loser. This FSM can be seen in Figure 1 below. Figure 2 shows the component block diagram of the digital circuit, using a multiplexer in place of the toggle FSM.

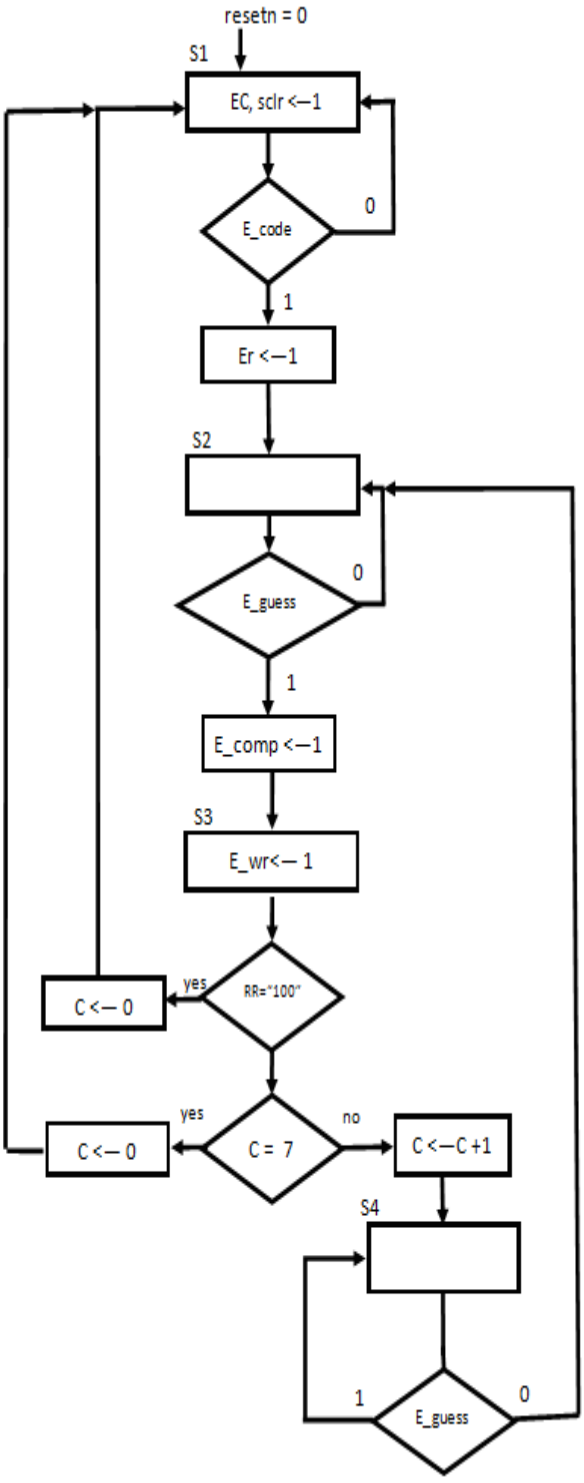

Figure 1. Main FSM

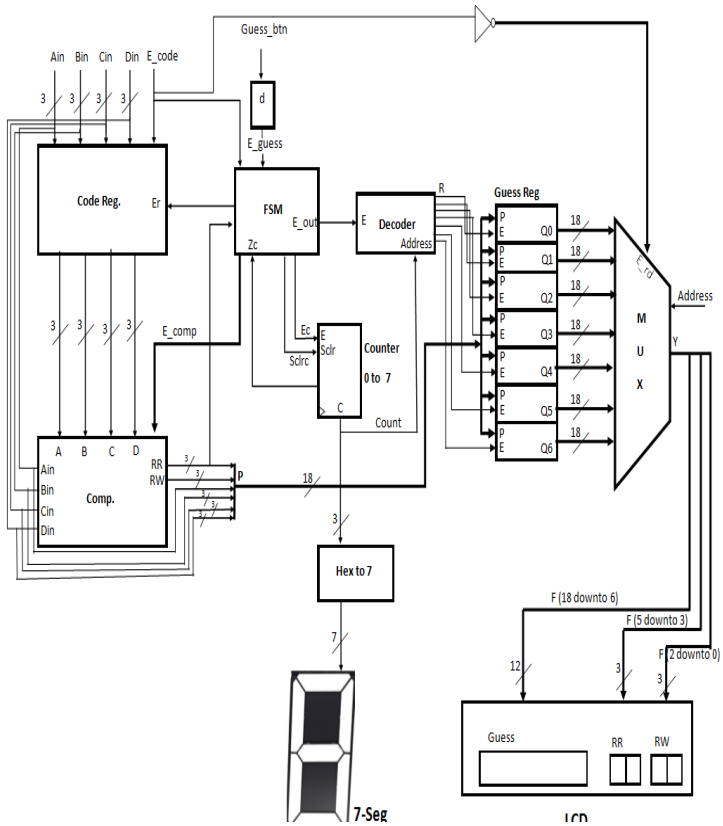

Figure 2. Component block diagram

III. Experimental Setup

The Nexys 4 DDR board was used on ISE 14.7 software to implement the digital circuit with VHDL. The enable for the code was assigned to SW15. The code and guesses were assigned three switches per digit (SW11-SW0). The enable for the code to be entered was the center button (btnC) and would then display the current guess and the RR and RW responses for that guess on the LCD. Left and right toggle buttons (btnL and btnR) were designed to toggle back and forth through the previously entered guesses. The reset button on the board reset the program. The LCD module had pins connecting to the board for power, ground, data, enable, and read/write [2]. The LCD controller had a clock, reset, enable, and data bus for the output data [3]. The LCD controller and module can be seen in Figure 3. It also had it's own FSM, which can be seen in Figure 4. The expected results are to save a code then input up to 8 guesses that show up on the LCD screen with their resulting RR and RW values.

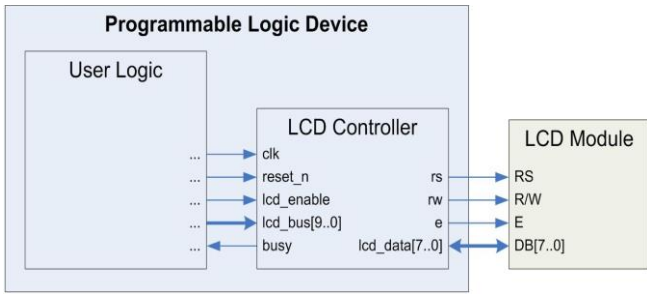

Figure 3. LCD Module

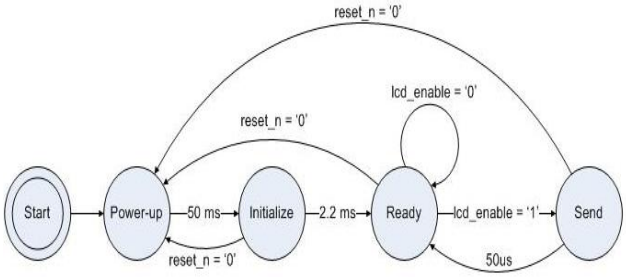

Figure 4. LCD Controller FSM

IV. Results

Several adjustments were made to comply with time restrictions for this project. Unfortunately, the LCD was the largest piece of the design that was left out of the final result. The code for the LCD was unable to be sorted out in the allotted time; therefore alternative methods of demonstrating the circuit were used. The setup was to show the 4-digit code/guess on the 7-segment display instead of the LCD. To show RR and RW, LEDs were used to represent these 3 bit numbers. The leftmost 7-segment display also counted up with the guess count. Also, the toggle FSM was changed into a multiplexor, as shown in Figure 2, to ensure success with the new experimental setup. A green light was added to go off when a guess matched the code correctly and there was a winner. Although this was not ideal to the original plan, the game could still be played and it successfully showed how the circuit was designed. There were test benches made for a losing scenario and for a winning scenario to show the working circuit. They can be seen in Figure 5 and Figure 6 of the Appendix. They show how the registers receive the data of the guess, RR, and RW when the FSM goes to State 2. In State 3, for the losing simulation, the game is reset after State 3 determines that the count has reached 7 without a correct guess. For the winning simulation, the code is guessed on the 4th try, and State 3 see RR equal to 4. Therefore the game resets at this point. On the board, the code is entered and shown on the displays, and the switch to write the code is flipped. At the point the guesser is handed the board with the switches in the down position and all 0's on the 7-segment display. The guesser will enter a guess, it will show up on the displays, and the center button is pushed. The counter counts up at each guess, and RR and RW show up on the assigned LEDs. The game can still be played even without the LCD as the output display.

Conclusions

A digital project needs to be very carefully planned and thought out. Every single scenario and outcome needs to have a solution. Otherwise it will create problems during the design process. The planning stage for this project could have been better executed. Although there were many setbacks to this project, the design team analyzed the problem, brainstormed solutions, and in the end came out with a working digital version to the game. Designing and coding in VHDL is a learning process, and this project greatly accelerated this process for the team members involved.

References

[1] Daniel Llamocca. "VHDL Coding for FPGAs". http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

[2] Shenzhen Eone Electronics Co., Ltd. "Specification fro LCD Module 1602a". https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf

[3] Scott Larson. "Character LCD Module Controller (VHDL)". Edited Aug. 2013. https://eewiki.net/pages/viewpage.action?pageId=4096079

Appendix

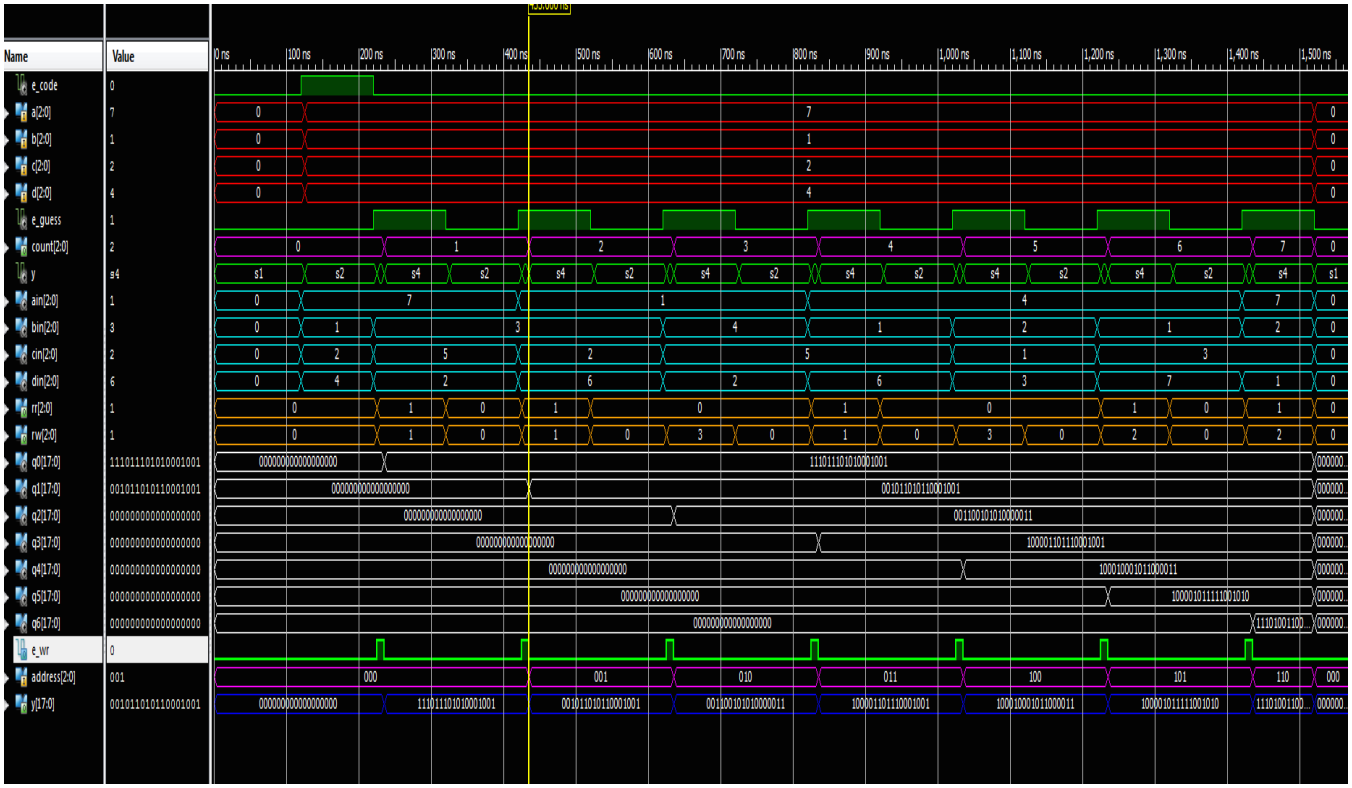

Figure 5. Losing Simulation

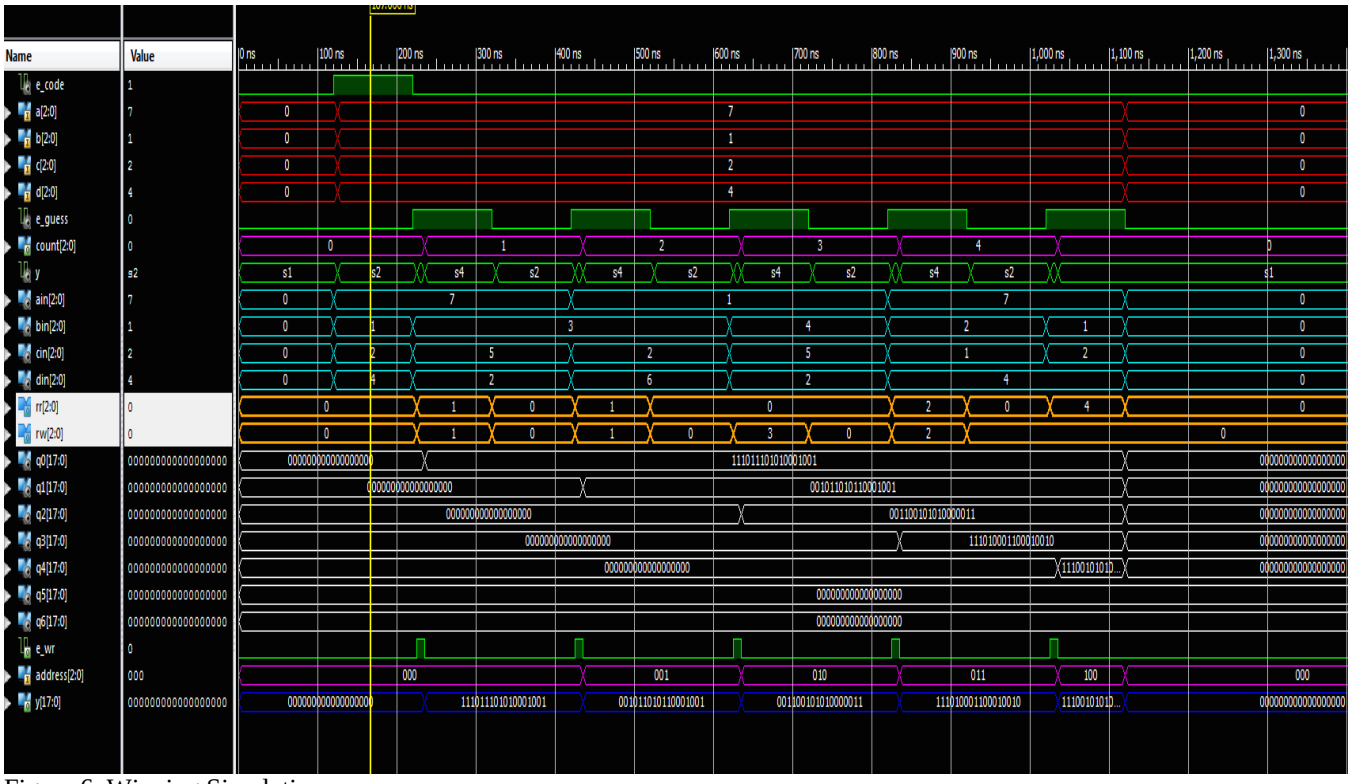

Figure 6. Winning Simulation