FPGA Security System

Andrea Arenas, Justin Klotz, Noor Alnounou
ECE 4710 Computer Hardware Design
Oakland University
Rochester, MI, USA
aarenas@oakland.edu, jjklotz@oakland.edu, Nalnounou@oakland.edu

***Abstract*— This project is intended for the students to demonstrate the knowledge acquired about Computer Hardware Design using VHDL. The purpose of this project is to design a security system that utilizes a simple user interface that consists of a keyboard and a seven-segment serializer. This security system will be capable of performing a few operations including setting a new pass code, entering a passcode, denying or granting access, locking the user out of the program if a wrong passcode is entered too many times as well as restarting a guess.**

I. INTRODUCTION

Our project consisted of creating a security system requiring a 4 digit code. On startup, the user has to set a new 4 digit code and will be prompted by the blue LED. the user will enter the code using a PS2 keyboard that is decoded by the scan code decoder and then stored in 8 4-bit registers. The correct code on startup will be stored in the first 4 of 8 registers with each register holding a numerical value of 0-9. Once the correct code is set, the blue LED will turn off telling the user they can begin to guess the correct code. The guessed code will be stored in the last 4 of 8 4 bit registers. Once the 4th digit during the guessing stage is entered, the code will be sent to a 16 bit comparator that will compare the output of the registers containing the saved code with the code currently being entered. If the comparator outputs a 1, the LED will shine green to signify the code was correct. If the comparator outputs a 0, the LED will shine red in this case. For each failed attempt, a 0 to 2 counter will be enabled and start to count. If the user has 3 failed attempts, the LED will shine purple. This will then enable a 20s counter. When the 20s counter reaches 20, the user will go back to state 2 and have another 3 attempts to try at the code. The user is also able to erase the guessed code in the registers if they make a mistake and do not want to waste another guess. This is achieved by pressing the BTNC button at any time a code is being entered. Lastly, the seven-segment display will show the numbers as they are being entered.

II. METHODOLOGY

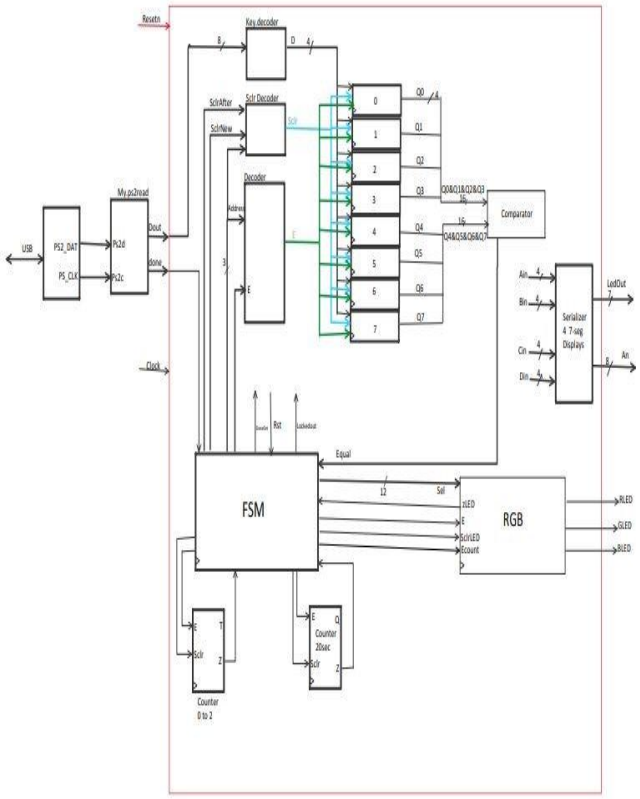

Figure 1 - Block Diagram

The methodology for this project involves several steps. First, research and analysis of security system design using VHDL will be conducted to comprehensively understand the subject matter. Next, the requirements and specifications for the security system will be established, including the necessary hardware components and their functions. The VHDL code will then be written and tested using simulation software to ensure that it operates correctly. Finally, the system will be implemented on a hardware platform and thoroughly tested to ensure that it meets all the project requirements. Regular feedback and revisions will be incorporated into the project to ensure its successful completion.

Finite State Machine

An FSM is a model of a system that has a limited number of states that it can be in, and it can only move from one state to another in response to specific inputs.

This FSM has 10 states, represented by S1x, S2x, S3x, S4x, S1, S2, S3, S4, S5, and S6. The inputs to the FSM are resetn (reset signal), clock (system clock signal), done, equal, Zthree, Ztwenty, rst (reset), and ZLED. The FSM also has several output signals including sclrAfter, sclrNew, sclrthree, Ethree, sclrtwenty, Etwenty, doneSet, countEnable, Ed, ELED, sclrLED, sel (selection signal), address (address signal), and locked out. The process named Trans describes how the FSM transitions between states.

The process checks the current state of the FSM and the inputs to determine the next state. If the reset signal is low, the FSM is in the initial state S1x. If the reset signal is high, the FSM checks if the done signal is high to transition to S2x. The same process is repeated for the other states. The final state S5 is reached when the input equal is high, and the input ZLED is low, or when the input Zthree is high and ZLED is high. The final state S6 is reached when the input Ztwenty is high. In states S1x-S4, if the rst signal = ‘1’ at any time, the state will go back to the start of the correct code setting or the guessed code setting states. S1x, S2x, S3x, S4x states are for setting the correct code and S1, S2, S3, S4 are for setting the guessed code. That would mean if the rst button is pressed at any time during the setting of the correct or guessed code, the FSM will go back to either S1x or S1 depending if the user is in the guessing or setting the correct code stage.

The process named Output describes the FSM outputs. Depending on the current state, the output signals are assigned different values. The signals address, and Ed are used to select a specific memory location for storing data(i.e which registers to store the numerical data, enable, and synchronous clear). These signals also decide what registers get enabled and cleared via the decoders. The signal sclrNew is used to clear the new code input, and the signal sclrthree is used to clear the counter which is counting from 0 to 2. The signal sclrtwenty is used to clear the locked out counter. The signals Ethree and Etwenty are used to enable the counters, respectively. The signal doneSet indicates that the new code has been set. The signal countEnable enables counting. The signals ELED, sclrLED, sel, lockedout control the RGB LED display.

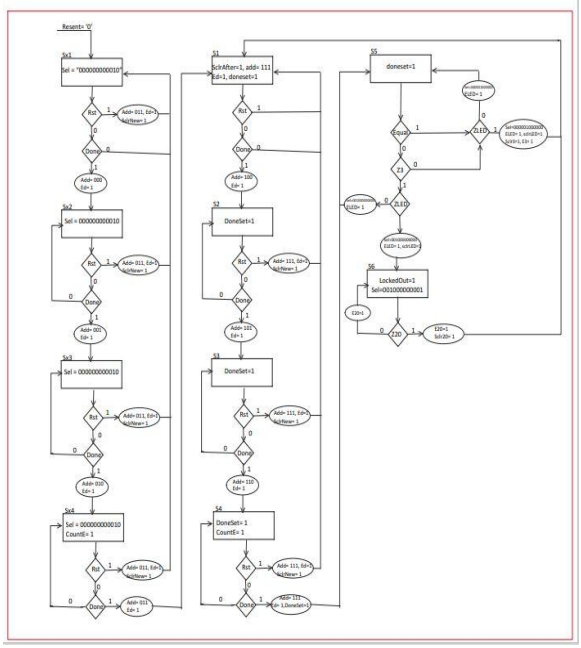

Figure 2- FSM

PS2 Keyboard

In order to receive input the PS2 keyboard, which is a hardware interface used to connect a keyboard or mouse to a PC, was implemented. When a key is pressed, the keyboard transmits one byte. Every key has its own scan code definition which is an 8 bit signal. The byte is sent to the host every 100 ms until the key is released. When the key is pressed down and released, the PS2 keyboard component will output "F0" to signify the component to read the scan code of the key being pressed. The scan code of the key will then be outputted. For this project, the keys for numbers 0 to 9 are used as input for the security code. The input received from the keyboard is used to set and guess codes.

Figure 3 - PS2 Keyboard Values

PWM

Different color combinations can be achieved by varying the brightness of the red, green, and blue LEDs through pulse width modulation. The 4 MSB are used to calculate the duty cycle of the red LED, the 4 LSB for the blue LED, and a set of 4 bits "0001" for the green LED. These 4 bit values are multiplied by TPWM, which is set to 50000. A frequency of 2kHz is used in order to obtain a good color.

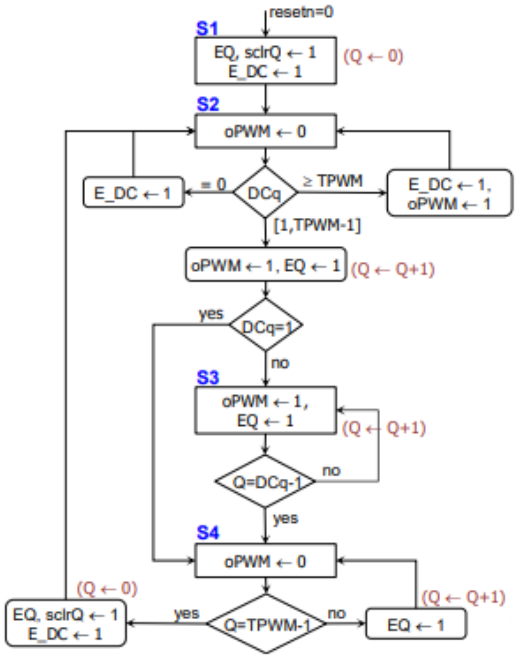

Figure 4 - PWM FSM

RGB LEDs

The Nexys A7 board has two RGB LEDs. We utilized one of them in order to indicate to the user if the code which has been entered is correct by shining green. In a similar manner if the code which has been entered is incorrect, the LED will shine red. Lastly, when the user has entered an incorrect code past the amount of times allowed, the LED will shine purple until the system allows the user to try again.

Decoders

The first decoder decides which register will be enabled based on the "address" signal from the FSM. The second decoder will decode the scan code from the keyboard to a 4 bit number of 0-9. The third decoder will clear the correct code registers or the guessed code registers based on where in the FSM the program currently is. This function is only activated when the rst or BTNC button is pressed.

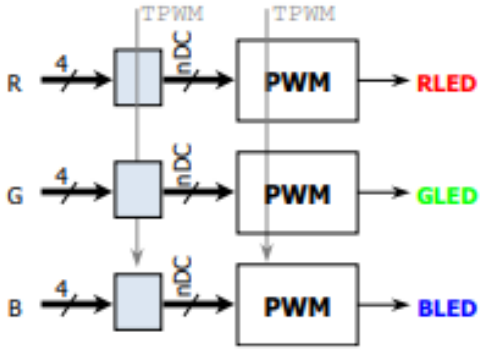

Figure 5 - RGB LEDs

Seven-Segment Serializer

The seven-segment display is used in this project to display the numbers which the user enters by using the keyboard. The serializer is used in order to be able to use more than one seven segment display. This part consists of a multiplexor which receives data from the registers which store the input from the keyboard. This data is converted by the HEX to 7 segment decoder in order to display the numbers. A 2-to-4 decoder is what determines which of the seven segment displays is used for each number. Lastly, a counter which keeps each digit illuminated.

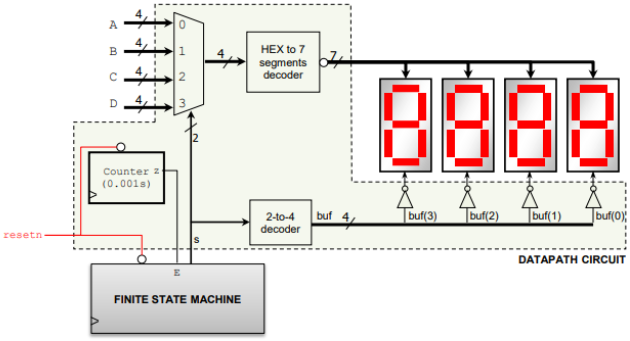

Figure 6 - Seven-Segment Serializer

III. EXPERIMENTAL SETUP

The experimental setup for the Computer Hardware Design project using VHDL will require the following hardware components: a development board with a compatible FPGA, a keyboard, a seven-segment display, RGB LEDs, and a power supply. The VHDL code will be written using a compatible software tool like Vivado. The project will be divided into modules, including a keyboard module, a seven-segment display module, a passcode module, and a control module. The keyboard module will scan the keyboard for user inputs, which will be displayed on the seven-segment display module. The passcode module will store and verify user passcodes, while the control module will manage the entire system, including granting or denying access and locking the user out after three-time failed attempts. The entire system will be tested thoroughly for functionality, and any bugs or errors will be corrected before finalizing the design.

IV. RESULTS

After several testings, we were able to achieve our desired outcome. The security system is able to take input from the keyboard in order to set a security code, as well as accepting the correct combination after creation, denying access to the system when an incorrect combination is entered, and preventing the user from trying again after a set amount of wrong combinations.

V. CONCLUSION

This project aims to showcase the students' knowledge of Computer Hardware Design through VHDL. The goal is to design a security system with a user-friendly interface using a keyboard and a seven-segment serializer. This system will have several functions such as setting a new password, entering a password, granting or denying access, and locking the user out of the program if too many incorrect passwords are entered. Overall, this project will demonstrate the student's ability to apply their skills to real-world applications.

VI. REFERENCES

[1] Donatus, Ayodeji Duyilemi, et al. "FPGA: Based Security login System."

[2] Aminuddin, Zaim Zakwan, et al. "An FPGA application of home security code using verilog." *Int J Reconfigurable & Embedded Syst* 11.3 (2022): 205-214.

[3] Brown, Arthur. “Nexys A7 Reference Manual.” Nexys A7 Reference Manual - Digilent Reference, Digilent, https://www.secs.oakland.edu/~llamocca/Courses/ECE2700/Boards/NexysA7_rm.pdf

[4] Llamocca, Daniel. VHDL Coding for FPGAs, https://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

[5] Llamocca, Daniel. Winter 2022 - ECE4710: Computer Hardware Design, https://www.secs.oakland.edu/~llamocca/Winter2022_ece4710.html

[6] Llamocca, Daniel. Unit 3 notes, https://www.secs.oakland.edu/~llamocca/Courses/ECE4710/Notes%20-%20Unit%203.pdf