Digital Security Alarm System

ECE2700 Final Project Report

List of Authors (Mario Cali, Ronza Younan, William Pham, Irvin Watson)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: mariocali@oakland.edu, ronzayounan@oakland.edu, xuanpham@oakland.edu, irvinwatson@oakland.edu

***Abstract*— In this project, students use skills taught in ECE 2700 Digital Logic Design and preceding classes to create a digital system utilizing FPGA and Vivado to code VHDL. The students decided to create a digital alarm system that will sound when a motion sensor is triggered and can be disarmed using a four-switch combinational code. The primary goal of this project is to dissuade intruders. The creation of this digital system led to several findings and additional suggestions.**

I. Introduction

This digital motion alarm system will necessitate several electrical components in addition to software programming. This project will make use of a Nexys A7 100T FPGA, and buzzer amplifier, an HC-SR501 PIR (Passive Infrared) motion sensor module to detect motion within an area, an LED, an active buzzer, a micro breadboard, and jumper wires. The code will be written in Vivado VHDL 2019.1. The goal of this study is to better understand and duplicate the operation of alarm systems. Another incentive is knowing whether there is an intruder, or someone present within a room for security concerns, with the intention that a buzzer will stop them before they steal anything.

This project can help to improve security and safety by sounding an alarm whenever there is movement in an area where intruders are not welcome. This project incorporates many of the concepts covered in this semester thus far. To be capable of putting specific components into code, one must first understand how they work. It is critical essential students not only grasp how to code components in VHDL using Vivado, but additionally how to connect these parts using a top file and construct a comprehensive testbench to mimic his or her code.

The abilities required for performing these tasks were thoroughly covered in the course and were heavily applied during this project. Counters, decoders, and deterministic finite systems are examples of important Vivado coded components that were taught in class. To complete this project, the students needed to study a few things on their own. Students required to read the Nexys FPGA documentation to figure out what the I/O pins on the FPGA were called to discover them in the xdc restrictions file. Understanding how to use a PIR sensor, how to properly feed voltage to the FPGA without destroying the board, and how to properly feed voltage to the FPGA are all challenges of this project. Voltage should be fed from the FPGA board to a buzzer. It is also critical that students understand how to wire simple circuits. This project's uses are applicable in any case where a motion sensor alarm is required.

II. Methodology

A. System's Goal

This project is supposed to replicate a home security digital alarm system in order to prevent any intruders from breaking in. The alarm is activated only in the case of motion detection by a Passive Infrared sensor (PIR sensor). The way it works is that the user has three attempts in order to get the code right. The code has five bits and is controlled by four switches (SW[3]-SW[0]) and also a button (CNT BTN) that needs to be held at the same time. So, the button acts as our enter key in a security alarm while the four switches act as the code to be inserted. The correct code is 11001. After every attempt, the user has 8 seconds in between in order to put in another code. If the user guesses the right code in any of the attempts, the normal state is restored, and everything goes back to normality. If the user runs out of attempts, then the buzzer is activated and will make a sound until the right code is inserted again. The setup and the wiring of the Nexys-A7 board is shown in Figure 1 below.

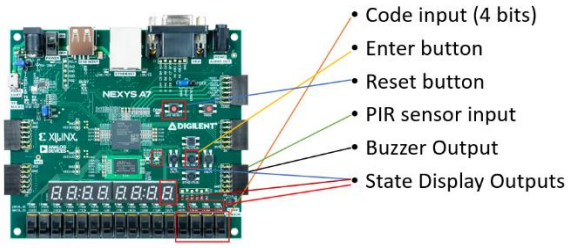


Figure 1: Nexys A7 100T board layout

B. Implementation

In order to implement this system, our group started picking out some components like counters, a decoder, a finite state machine and a BCD counter in order to use them. We had some different ideas how we were going to do this process but ultimately the goal was the same, which was a digital security alarm. We tried doing this project with a comparator where the code inserted is compared to the actual code saved in a state machine but decided to go for this design better as it was less complex and had the same function. All of the components had a specific task that needed to be done in order for this system to work. Figure 2 shows how the circuit was designed and implemented.

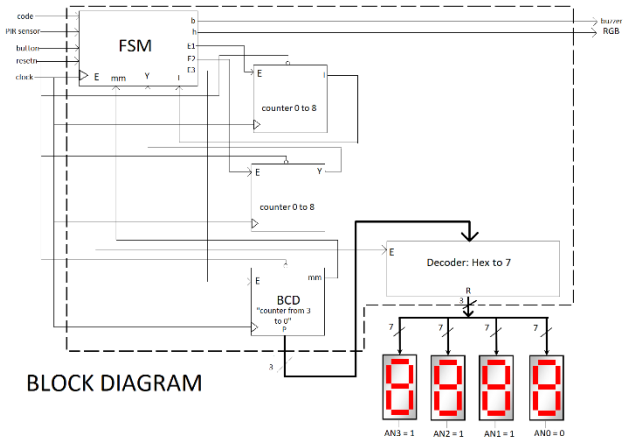


Figure 2: Block Diagram of the System

C. PIR Sensor/Buzzer

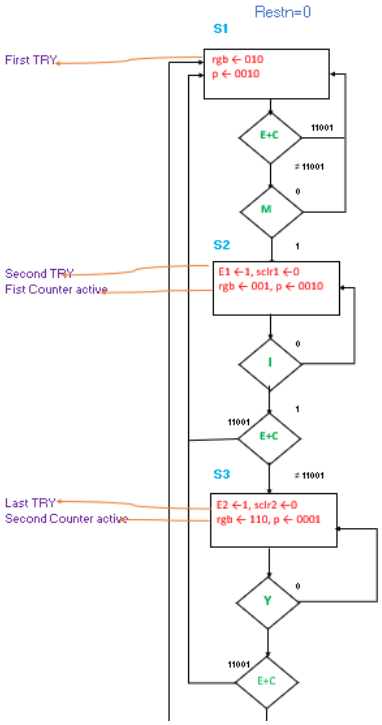
The main components that represent this digital alarm system are the PIR sensor and the buzzer. A Passive Infrared sensor (PIR sensor) is a type of sensor that uses infrared temperature to detect any kind of motion and transmit that motion to the FPGA board in this case. So, this sensor only picks up radiation which is released by the human body. It has a wide lens and a wide beam, which makes it able to pick up radiation in a range of 180 degrees. The motion pickup marks the start of the digital alarm system. Before using this type of sensor, its output was measured using a voltmeter to make sure it doesn't damage the Nexys A7 board. The board only supports a maximum input of 3.8V, so the output of the PIR sensor which acts as the input for the board was measured. After successful measurements, it came out to be around 3.31V, which meant that it was safe for the board to use this sensor. Figure 3 represents the motion sensor that was used to successfully complete this project. On the other hand, an active buzzer is also being utilized. The buzzer releases a sound and acts as a way to alarm everyone that someone is intruding into a house. Since the buzzer is active, it needs a high input from the FPGA Nexys-A7 board in order to get activated.

Figure 3: PIR Sensor Used in the Project

D. Finite State Machine (FSM)

Our project was mainly based on the state machine, where everything is based on the state that we are in. Figure 2 represents the Moore state machine, where each output is dependent solely on the present state and not the inputs. This state machine mainly relies on the code input from the switches and the button and the outputs from the counters. The output goes in the 7-segment screen and in the RGB LED of the FPGA board. The state machine has 8 different states, and each state signifies what is going on with our circuit. The first state is the normal state or the state that there are still three attempts left for the user to insert. In order to go from the first state to the second, the code must be wrong (not 11001) and motion must be detected. If the code is correct or no motion is detected, then the system will stay in the normal state. Once in the second state, then a counter is activated and when the output is high then the code will get checked again. So, in simpler terms, there is a

certain timer that the user has before the code needs to be inserted. If the code is correct, then normal state should be regained. Otherwise, the user goes to the third and last attempt before the buzzer is activated which is represented by state 3. State 3 has the exact same layout as state 2 where another counter is added to the system and the user has the same amount of time as in the second attempt in order to put in the correct combination. If the code is correct, normality is brought upon the system and state 1 is attained. Otherwise, state 4 is activated which marks the start of a BCD counter that counts down from 3 to 0. States 4 through 7 acts as a BCD counter with the output of a 1 second counter separating each of the states from each other. So, this countdown activates the buzzer, and the code cannot be inserted through this stage. After the countdown has been finished, state 8 is achieved. In this state, the buzzer will get activated and will not turn off until the right code is inserted. If the right code is inserted, then everything goes back to normal state 1. The output in each state is displayed in the 7-segment showing the user the number of attempts left after each try and also counting down to let the user know that the counter is activated. The image below represents the state machine of our system.



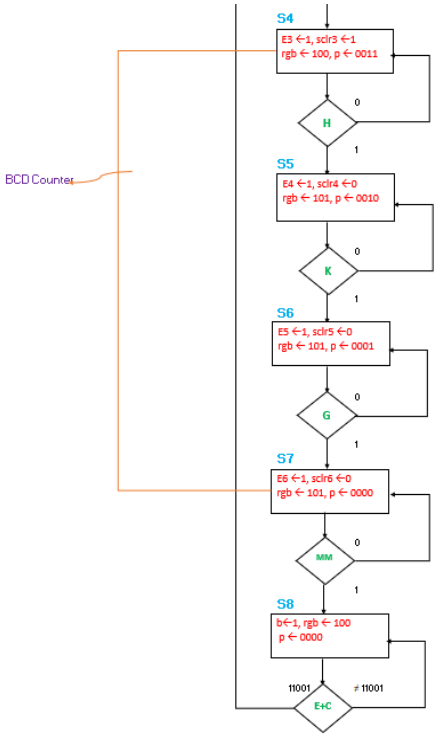


Figure 4: Finite State Machine of the project

E. Counters/Decoders

Two counters and a BCD counter were used in this project. It was decided for the two counters to be 8-second counters in order to prevent debouncing from happening and also create a realistic situation that would be found in the real world. As explained in the finite state machine diagram, these counters acted mainly acted as components to move from one state to another in the FSM. The BCD counter was also implemented inside the FSM, where a countdown from three to zero appears on the 7-segment LED screen to let the user know that the buzzer is getting activated. As for the decoder, it is also connected to the FSM and its input is the output of the FSM. The decoder converts the output of the FSM into 7-bit data that is shown in the 7-segment LED screen. This input of the decoder is also shown in the FPGA LED's.

III. Experimental Setup

For this experiment setup, the hardware that was used was the Nexys A7-100T board, Breadboard, PIR sensor, Active Buzzer, LED's and Wires. The breadboard was used to show the LEDs lighting up during the correct states. On the breadboard was the buzzer, LEDs. When the last attempt is wrong, the buzzer would sound off. The PIR Sensor is used to activate the actual alarm. The PIR Sensor picks up motion and starts looking for 4-bit code inputs from the Nexys A7 board. The wires are used to connect the buzzer on the

breadboard, to the Nexys board, and the PIR Sensor connected is also connected to the Nexys board. The expected results were to activate the PIR sensor, test different 4-bit codes, and also test to see if the buzzer goes off after all attempts are used.

IV. Results

After several tries and running the simulation with the project in the lab, we finally got the final results as expected.

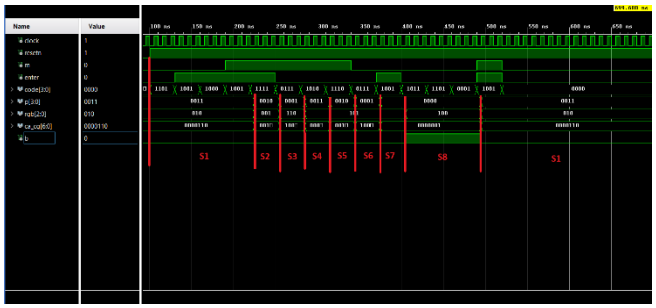


Figure 5: Testbench simulation with the project results

We tried it on the whole project with the buzzer and Nexys A7 100T board. Simply if the simulation worked well and counted exactly as we expected, the sensor caught up with the motion and it started to count. We tried every single case as stated on the FSM. The counter started to count, and the buzzer went off if there is no code or the wrong code is entered. Every number is displayed on the 7-segment display. The link below is the demonstration of the alarm clock showing the result of every case of the FSM and the buzzer goes off:
https://www.youtube.com/watch?v=87n0txZ7Uw4

Conclusions

This final project is a perfect wrap-up for a semester of learning about digital logic design. Making one real alarm clock system develops our skills in VHDL designing and coding. It's a step that lays important groundwork for stepping out in real life. We can make a sensor digital security alarm, and based on that we can also make so many things with sensors such as an automatic door, an automatic light system when it turns off all the lights while detecting no heat motion, and even auto driving for automotive. Most importantly, we want to improve and develop our digital security alarm system. The downside of our system is when the sensor detects motion, we need to hold the button in order to enter the code and reset it. We can switch from holding the button to turning on/off the switch and making it an input. The first four switches are input codes and the fifth switch acts like an enter button. We want to develop more on the alarm in order to use the full capacity if we have more time and the right equipment. The first thing is an on/off button that we want to add. The second thing is a door that acts like a trigger to turn on the alarm. The idea is to use two plate metals stuck to each other. If the metals are stuck, which means there is no one at home and the alarm won't start if it detects a motion that is not from people such as pets and so on. The alarm will start to work when the metals are separated which is someone opens the door. The third thing we want to add is the sound effect. It announces people for each try and calls out a warning if the person is not a homeowner and it leads to the last thing, we want to add is the system automatically calls the police when it knows there is a stranger which supposes to appear as red and blue LEDs flash on our system. We wanted our project to relate to real life as much as possible.

References

[1] D. Llamocca, "VHDLforFPGAs," VHDL Coding for FPGA's. [Online]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html. [Accessed: 23-Mar-2023].

[2] "What is a PIR Sensor?", Quick and Easy Lightning. [Online]. Available: https://quickandeasylighting.com/what-is-a-pir-sensor/. [Accessed: 24-Mar-2023].

[3] D. Llamocca, "Unit 7–Introduction to Digital System Design," Reconfigurable Computing Research Laboratory (RECRLab),Oakland University. [Online]. Available: https://www.secs.oakland.edu/~llamocca/Winter2022_ece2700.html. [Accessed 23-Mar-2023].