Digital Motion Alarm

Final Project Report

List of Authors (Josh Viar, Donovan Deza, Lewis Kowalec)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: jviar@oakland.edu, donovandeza@oakland.edu, lkowalec@oakland.edu

Abstract— In this project, students utilize skills learned in ECE 2700 Digital Logic Design and classes previous to design a digital system using an FPGA and Vivado to code VHDL. For this the students chose to design a digital alarm system that will sound when a motion sensor is triggered and can be disarmed with a four switch combinational code. The main purpose of this project is to be able to deter intruders. Many discoveries and further recommendations were found through the process of making this digital system.

I. Introduction

This digital motion alarm system will require different electrical components as well as hardware programming. This project will use a Nexys A7 100T FPGA, an Arduino Uno used as a 5V power source and as an amplifier for the buzzer, an HC-SR501 PIR (Passive Infrared) motion sensor module to detect if there is motion within an area, an LED, an active buzzer, a mini breadboard, and jumper wires. The programming will be done in Vivado VHDL 2021.1, as well in Arduino IDE. The motivation for making this project is to better understand and be able to replicate how alarm systems work. Another motivation is being able to know if there is an intruder or someone present within a room for security reasons with the hope that a buzzer will be able to deter them before they steal belongings.

This project can help to make areas more secure and safe by having an alarm sound if motion is detected in an area where intruders are not wanted. This project covers many areas that students have learned in this class so far. One needs to know how certain components work in order for them to be able to implement them into the code. It is crucial that students not only understand how to code components in VHDL using Vivado, but it is just as important for them to be able to know how to interconnect these components using a top file and to be able to write a proper testbench to simulate his or her code.

The skills necessary to be able to do these tasks were taught extensively in the course and were utilized heavily during this project. Important parts that were coded in Vivado that were taught in the class are counters, decoders and finite state machines. The students needed to learn a few things on their own however, to actually pull off this project. Students needed to look through the

provided manual for the Nexys FPGA to find what the I/O pins on the FPGA were named so as to be able to find them in the .xdc constraints file. A challenge of this project is learning how to use a PIR sensor, how to feed voltage to the FPGA properly without damaging the board, and how to feed voltage from the FPGA board to a buzzer. It is also important that students have a solid understanding of how to use Arduino IDE and how to wire basic circuits. The applications of this project can be utilized in any situation where a motion sensor alarm is needed.

II. Methodology

A. PIR sensor and FPGA

This project working properly relies heavily on the use of an HC-SR501 PIR motion sensor module. The main challenge being that the FPGA board must be able to read data coming from the motion sensor, process that data and output a signal strong enough to power an active buzzer. The motion sensor and active buzzer being used as components to make the system work requires at least 5V, which students will generate from an Arduino Uno board. Since the FPGA cannot output 5V it will send its output signal to the Arduino where it will be amplified to 5V before being sent to the active buzzer.

Other challenges being faced during the setup and implementation of all components include the fact that the sensor requires 30-60 seconds of time after being powered on to be able to give an accurate reading to the FPGA board. This is due to the PIR sensor needing time to acclimate itself to the infrared energy in any given room. After initially powering the system on, this should not be too much of an issue, due to the always-on nature of the home security alarm systems. Another challenge given by the PIR sensor is the sensitivity of the range that it will read. The specific sensor used in this system gives a range of three to seven meters, with the topology of the given room potentially affecting the sensor (furniture, obstacles, etc). For demonstration purposes, the PIR sensor was placed in a controlled environment using a box in order to simulate an empty room.

After detection of a moving object by the sensor, it should output a 3.3V logic HIGH into the FPGA although this was measured using a voltmeter to be around 2.56V. This signal is then sent to the FPGA where it is read as either a HIGH or LOW signal. For the FPGA, voltage signals being input should not be above 3.8V, but should be atleast 3.3V, although the signal voltage from the PIR sensor is only 2.56V it turned out to actually be enough for the FPGA to read it as high. This output voltage is also connected to a blue LED, which indicates when motion is detected by the sensor. The minimum amount of time that the sensor will output a logic HIGH signal will be three seconds, with the maximum time being 300 seconds, or 5 minutes. This is due to the default specifications of the HC-SR501 PIR motion sensor and has a manual knob to adjust this, which was adjusted to 3 seconds.

The final feature of the digital alarm system project is a combinational four switch code which disables or enables the alarm. Using four switches on the Nexys

A7-100T FPGA board, a default combination will be set using SW3-SW0, such that when the right combination is enabled, the alarm will disable itself, and the sensor will wait before triggering again when it detects motion in the room. For ease-of-use purposes, the default combination will be set to '1111' to be used as the disable code. It is expected that this specific combination of switches will disable the alarm and turn off all active components with the exception of the PIR motion sensor. All other combinations of switches will not disable the alarm, but will keep it enabled.

B. Finite State Machine

The main inputs to the FSM consist of input 'X', which is the signal from the PIR motion sensor, input 'resetfsm', which will reset the entire circuit if activated, the clock input, and the 4-bit input 'C', which will serve as the combination for arming and disarming the alarm system.

The main output coming from the FSM is 'Y', which will be sent to the Arduino UNO to amplify the voltage, and then to the active buzzer, effectively sounding the alarm.

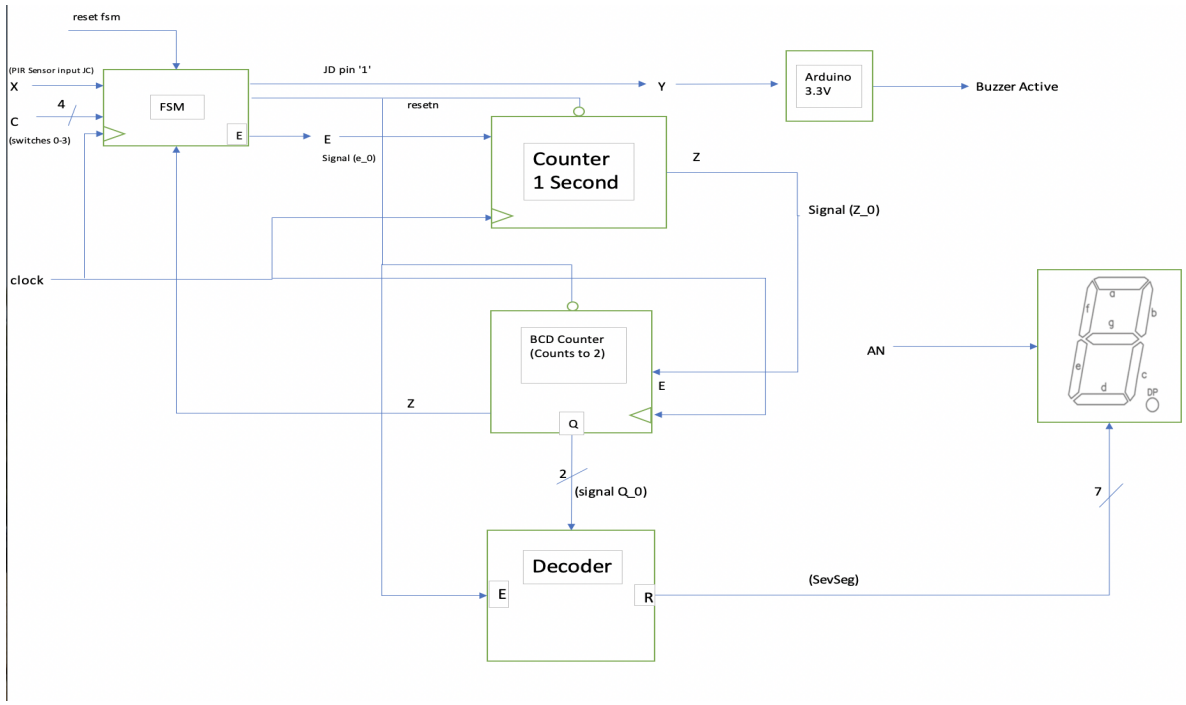
C. Other Combinational Circuits Used

The circuit contains multiple other components, such as two counters and a decoder, which will control the seven segment display to display a count from 0 to 2, giving the user time to put in the disengage code if a false alarm is triggered. The first counter in the circuit will count to $10*10^8$ nanoseconds (1 second), due to the nature of the FPGA's internal clock which cycles every 10 nanoseconds. The first counter will effectively delay the second BCD Counter, so that it will count once every second from 0 to 2 seconds. Once the second counter has reached two seconds, an output of 'Z' will display logic HIGH, and will be fed back into the FSM, and a high signal output "y" will activate the buzzer. The count from the second BCD counter will be shown on the seven segment display, giving the user a visual indication of the time remaining before the alarm will sound.

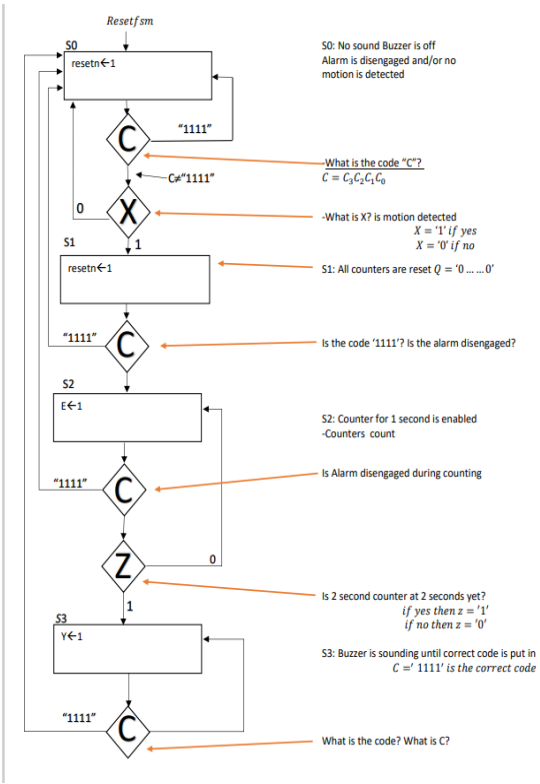
D. Combinational Code Reset

As a safety measure for the system, if the input $C=C_3C_2C_1C_0$ is set to the value '1111', the alarm will not sound, regardless of if motion is detected by the PIR Sensor. However, if the alarm is triggered and input C is set to any combination besides '1111', the alarm will continue to sound until the correct combination of inputs is entered into the FPGA, or the manual 'resetfsm' button is pressed. Note that the unit would have a cover and the manual resetfsm button would not be accessible to intruders.

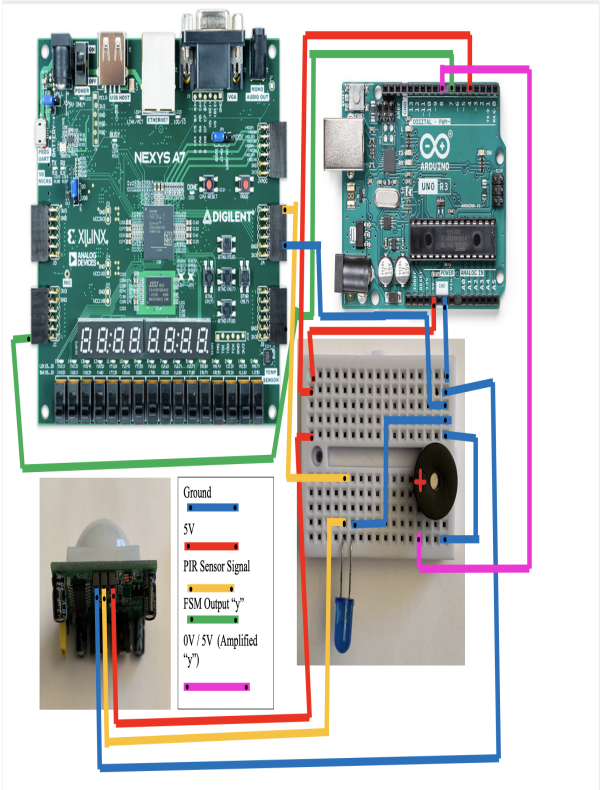
For more information regarding the circuit and its internal components, a block diagram is provided below as well as an ASM Diagram of the states included in the finite state machine and their purposes.



Block Diagram of Circuit



ASM Diagram for the FSM



Circuit wiring

III. Experimental Setup

The following hardware was used in the setup of this experiment: a Nexys A7-100T board, a HC-SR501 Passive Infrared Sensor, an Arduino UNO board used as a voltage source and amplifier for the output voltage, a single LED, an active buzzer used as the alarm, a mini breadboard, jumper wires to connect the circuit, a voltmeter for checking voltages before wiring, and Vivado behavioral simulations using a written testbench file to cycle through the states.

First and foremost, a testbench simulation file was created in Vivado to cycle through the states of the FSM and to see if it was working properly. After a successful simulation, a voltmeter was used to see what the voltage coming from the PIR sensor was in order to determine if it would be safe to input into the FPGA. Information on the exact PIR sensor said it would output a 3.3V signal if motion was detected, but using a voltmeter this signal was checked and it came out to be 2.56V. Since this was still below the maximum 3.8V, it was connected to the FPGA knowing this wouldn't damage the unit. The Arduino had a simple code that said if the input from the FPGA was HIGH then to set its output pin which was connected to the active buzzer equal to a pin on the Arduino being fed a voltage of 5V. This simply set the output pin being fed to the buzzer to 5V whenever the FPGA sent out a HIGH signal (this voltage was also checked with a voltmeter and was read as 5V to ensure the buzzer was being powered properly).

After all of this, the circuit was built and the digital system demoed to see if things would work properly and they did, so it was a success.

The expected result of the experiment is once the PIR sensor detects motion and the combinational code is anything other than ‘1111’, a two second counter will restart and count from zero to two seconds, giving the user time to disengage the alarm. If the combinational code is anything other than ‘1111’ and the ‘resetfsm’ button has not been pressed, the active buzzer will sound after two seconds. The active buzzer will continue sounding until ‘1111’ is entered into SW3-SW0 on the FPGA board or if ‘resetfsm’ has been pressed (again this is not accessible once a cover is put over the unit as would be if it were utilized in the real world). Once the correct combination code ‘1111’ has been entered into SW3-SW0 or ‘resetfsm’ has been pressed the buzzer will stop sounding.

IV. Results

The results obtained were in line with what was expected, such that once the PIR sensor had detected motion and the combinational code was anything other than ‘1111’, the LED lit up for roughly 3 seconds and the counters successfully reset and began counting from zero to two seconds, which was displayed on the seven segment display and if the alarm was not disengaged by putting in the right combinational code– ‘1111’ –across the first four switches on the FPGA board, before the 2 second BCD counter had finished its count, the active buzzer began to sound and continued to

buzz until ‘1111’ was entered into the switches 3-0 on the FPGA board or if the ‘resetfsm’ button was pressed (again a cover was utilized during demonstration in class). The PIR sensor also successfully outputted a 3.3V (2.56V) logic HIGH signal whenever motion was detected and the FPGA was also successfully able to send a logic HIGH signal to the Arduino UNO board for amplification to 5V and sound the buzzer after the count.

V. Conclusion

This project serves as a way to put many topics students learned throughout the semester all into one, in order to create an external interfaced based digital system with the purpose of improving home security.

Through experimentation and trial and error, this system can be improved upon in multiple different ways, including but not limited to having multiple motion sensors in many rooms throughout a house and/or sounding multiple alarms to be heard throughout the house or dwelling. Also, it may be helpful to have a combinational code that is changeable in the case that the code is compromised. This could be implemented using a design similar to lab 5 where a Random Access Memory could be used to hold a code that is written by the user and then that code would be stored and used as the enable/disable code. The reason this wasn't implemented is it would be easy for the intruder to then reset the code himself. However, this could be used if another FSM was implemented to see if the current code was input and then would enable the rewriting of the code.

With more time this would have been a cool feature to include in the project. However, the overall objective of this project was a success. The students applied many different topics learned throughout the ECE 2700 course to create an effective security alarm using students’ knowledge of VHDL and digital logic circuits. These goals were achieved through collaboration of teammates, trial, error, and knowledge of digital systems and VHDL coding. The overall result led to the successful completion of a digital alarm system by the group of students.

Video Demonstration of Circuit

References

[1] Last Minute Engineers. (2020, December 18). *How HC-SR501 pir sensor works & how to interface it with Arduino*. Last Minute Engineers. Retrieved December 11, 2021, from https://lastminuteengineers.com/pir-sensor-arduino-tutorial/.

[2] *Nexys A7 Reference Manual - Oakland University*. (n.d.). Retrieved December 11, 2021, from http://www.secs.oakland.edu/~llamocca/Courses/ECE2700/Boards/NexysA7_rm.pdf.

[3] VHDL coding for fpgas. (n.d.). Retrieved December 11, 2021, from http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html.

[4] *Arduino Uno REV3*. Arduino Online Shop. (n.d.). Retrieved December 11, 2021, from https://store-usa.arduino.cc/products/arduino-uno-rev3/.