Ultrasonic Sensing

Using an ultrasonic sensor to measure distance

List of Authors (Nguyen Phan, Nicholas Yang, Ayoub Kakish, Alex Alnajjar)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: Nphan@oakland.edu, Nicholasyang@oakland.edu, Akakish@oakland.edu, alexalnajjar@oakland.edu

***Abstract*—**The object of this project was to design a way to use the ultrasonic sensor with VHDL and the Atrix-7 board. Ultrasonic sensors are becoming more relevant today because of the ease of implementation and effective use. What was found was the code could be written to successfully gather distance readings from the ultrasonic to the Atrix-7 board.

I. Introduction

Ultrasonic sensors are simple trigger and echo devices that output a pulse which measures distance. The trigger outputs a pulse which will eventually bounce off the object the operator wishes to measure the distance of. The echo will then have that bounced back pulse and input that information. This concept is useful because now the operator can measure distance of an object continuously. This concept can be used in the automotive industry to alert a driver when they are backing up and getting too close to an object such as another car. This small alert system can decrease the amount of collisions in a parking lot dramatically while also not costing the manufacture so much as ultrasonic sensors tend to be cheap to produce. Figure 1 shows the ultrasonic sensor used in this project.

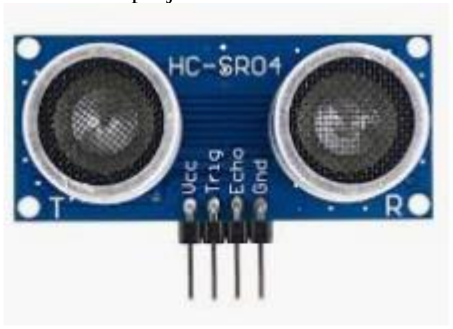


Figure 1: HC-SR04 Ultrasonic Sensor

The scope of this project is basically to interface the ultrasonic sensor with the Atrix-7 board. The should detect an object and the width of the echo pulse should be measured. The measurement will be calculated in cm and displayed on two seven segment displays on the board. Figure 2 shows the FPGA board used in the project.

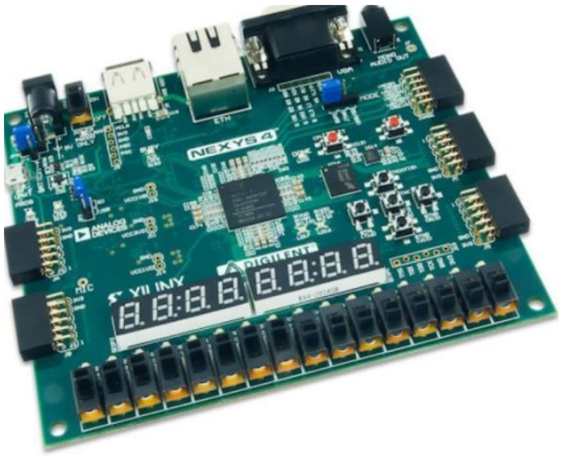


Figure 2: Artix-7 FPGA Trainer Board

Methodology

A. Ultrasonic sensor

To design and implement a system to measure distance, one must first understand the ultrasonic sensor and how it works. The sensor is comprised of two main components and in this project the sensor used has four pins. The trigger and echo are the main pieces of what an ultrasonic sensor is. The sensor will receive a pulse at a minimum of 10µs from a microcontroller, such as an Arduino, which will indicate the sensor to output a pulse via the trigger. The trigger sends 8 pulses at 40kHz so that the reflection is expected. That pulse will travel through the air until it reaches the object of interest. That pulse will hit the object and bounce off, returning the sensor. Here, the echo will receive that signal which will be implemented in the code to output a useful measurement. Enable conditions to consider is the air temperature and velocity of sound. The way the pulse travels through the air dictates the amount time that pulse take to reach the object and return to the echo. The speed sound travels at is 343 m/s and room temperature are usually around 20°C (68°F). Figure 3 shows the pulses generated by the sensor.

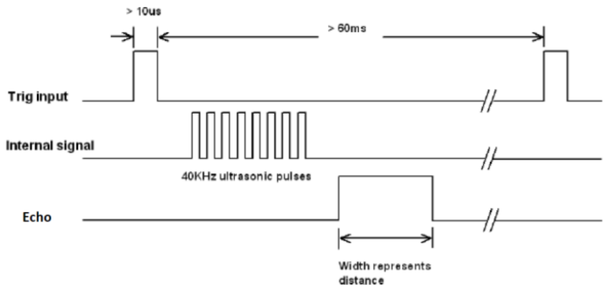


Figure 3: Ultrasonic Sensor Pulses

The circuit used to hook the sensor to the board is very simple. The source of voltage to power the sensor will come from an Arduino board since is can output 5 volts and it also has many pins to use for ground. A small bread board will also be used to hook the sensor, the power and ground from the Arduino, and the interface from the Atrix-7 board including ground. Since the sensor uses 3.3 volts, a voltage divider will need to be made to ensure that the board is safe from over voltage.

B. VHDL Code

The logic circuit that makes up the system to measure distance in this project contains three counters, BCD converter, HEX to Seven Segment Converter, 2 seven segment displays in centimeters, Finite State Machine and the calculator to perform the distance calculations.

The FSM is the main controller of the circuit which generates the trigger for the sensor. The FSM used in this project contained seven states which, once the sequence was completed, returned to state 1. Figure 4 shows the complete finite state machine.

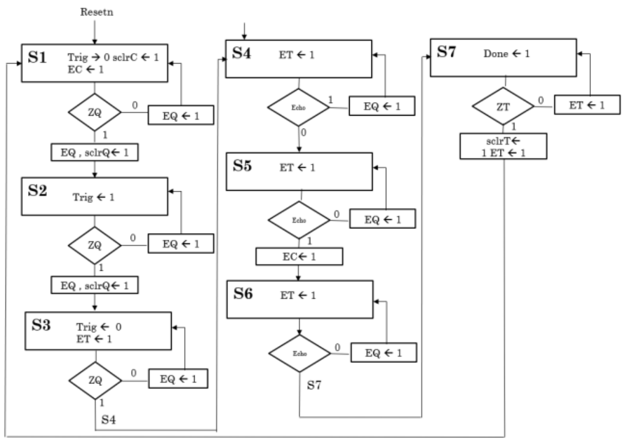


Figure 4: FSM

While in state 1 Trig is 0 while sclrC and EC are 1. If ZQ equals 1, then the FSM goes to state2 where Trig is equal to 1. If ZQ equals 1, then EQ and sclrQ are also 1 and it moves to state 3. In state 3 Trig is equal to 0 again and ET is now 1. Once ZQ is 1 again the machine moves to state 4 where ET is 1. Now we are looking for when echo is equal to 1. Once this happens the state machine moves to state 5 and will remain if echo is 0. In this state when echo is equal to 0, EQ is 1 as well. Now when echo goes to 1, EC is now 1 and the machine goes to state 6. In state 6 ET is 1 and if echo is 0, it will keep state 6 and EQ will be 1. Once Echo is 1, then the machine moves to state 7 and now done is equal to 1. If ZT is 0, then ET is a and we stay in state 7 and done is still 1. Once ZT finally is 1 the state machine goes back to state 1 and the sequence starts over.

Three counters were used for this project. The first counter was used to count the echo-width. The reason for this is that the eacho-width represents the time that the echo takes to return to the sensor. This echo-width was then inputted into the Distance Calculator to convert that width into centimeters. The other two counter were used to synchronize the trigger. One clock would control the trigger every 10µs which was the pulse output from the sensor. The last clock was the delay in the trigger pulse. The trigger needs time to bounce off the object and come back into the emitter which why this clock is needed. Initially 60µs was used, but this was too quick and made the display flicker too much. In order for the display to remain clear 800µs was used which fixed the issue. .

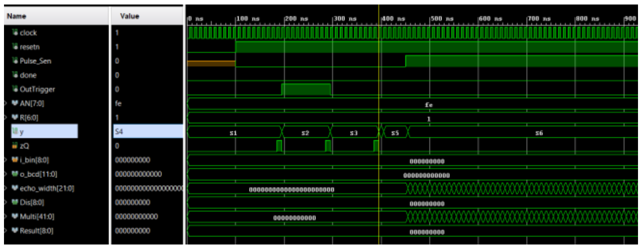
The distance calculator is where the echo width from the modulo-counter will be converted into cm. The equation to find the distance in cm is given by $Distance(cm) = \frac{echowidth(\mu s)}{58}$. Once this number is calculated in will move on to the BDC converter. However, in the board that time will be in Nanoseconds, so the equation needed to be modified. To convert to Nanoseconds the echo-width needs to be divided by 1000 to get µs. It also needed to modify the equation by multiplying by $1/5800$. The reason for this was because in VHDL it is difficult to divide numbers so we just multiply by the inverse. $\frac{1}{5800}$ is equal to 1.724*10^-4 which is a 20 bit number. Multiplying echo-width, which is a 22 bit number, and 1.724*^-4 will yield a 42 bit number. However, only the first 9 bits before the decimal point will be used to display the distance on the seven-segments.

Once the distance is calculated in the component Distance Calculator, the BDC Converter will take that 9 bit number and convert it to a BCD number. This BDC number will be used to convert into the seven-segment display through the Hex-to-Seven-Segment component.

Once the code was complete and the project was working correctly, there was an issue that needed to be addressed. The display would display the distance the sensor was calculating just for about a minute and then only read zero. To fix that, a D flip flop was added and once that update was tested the project worked 100% correct.

C. Results

To demonstrate how the ultrasonic sensor worked, we decided to use a seven-segment display to illustrate the distance between the sensor and object. The seven-segment will be displayed in hex. As the object moves farther or closer to the sensor, the seven-segment display will simultaneously measure the distance showing us how far the object is. Overall, the result of our final project was a success, the ultrasonic is working as it is intended to.



Name	Value
clock	1
resetn	1
Pulse_Sen	0
done	0
OutTrigger	0
AN[7:0]	fe
R[6:0]	1
y	S4
zQ	0
i_bin[8:0]	000000000
o_bcd[11:0]	000000000000
echo_width[21:0]	000000000000000000
Dis[8:0]	000000000
Multi[41:0]	00000000000
Result[8:0]	000000000

Here is the simulation of the code. Please note that the simulation doesn’t reach state 7 and reset because there was not enough time. In the final implementation, there time was extended and the code was complete.

D. Conclusion

Our team faced many setbacks while working on the final project mostly due to the calculations and timing of the echo pulse input. Once the timing was correct, the rest of the project went very smooth. We used many aspects from our class lectures and labs in order to create and design our final project. We incorporated a finite state machine, counters, decoders, d-flip flops, and 7 segment display. We also made some original VHDL code measuring the time values from the proximity sensor into distance.   The most important thing our team learned from this project was how to synchronize the finite state machine with multiple counters.