FPGA Sonar Distance Sensor

ECE 2700 Final Project

List of Authors (Bryan Dogariu, Emad Eissa, Sanket Patel, Neal Wright)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
Emails: bryandogariu@oakland.edu, meeissa@oakland.edu, sgpatel@oakland.edu, nwright@oakland.edu

Abstract

The purpose of this project is to design and implement a circuit with an ultrasonic (sonar) distance sensor utilizing an FPGA board. This presented some challenges, like connecting the sensor to the FPGA board, sending data to the sensor, and interpreting the data received from the sensor. The distance sensor has many different applications in the real world, including sensors used in the automotive industry for autonomous vehicles and parking assistance. The HC-SR04 distance sensor that was used is a hobby-level sensor with certain limitations on accuracy and functionality, but it performed well within the scope of the design. This project was chosen specifically to provide experience with implementing a simple sensor in a circuit design using an FPGA board.

I. Introduction

This project was motivated by the widespread use of distance sensors in the automotive industry and the practicality of controlling and implementing a sensor in a circuit design that involves an FPGA board. The primary concern was attaching the sensor to the FPGA board, sending and receiving data, and using that data to control parts of the circuit. There was a wide variety of class topics that were used, including state machines, multiplexers, converters, and decoders. Research had to performed to integrate the sensor into the circuit design and understand how the sensor sent and recognized ultrasonic pulses. There were many challenges, including writing most of the code from scratch, deriving the algorithms, interpreting the data, and properly wiring a step-down voltage for the sensor echo.

II. Methodology

Finite State Machine

The finite state machine for the circuit was designed to regulate the trigger and echo of the distance sensor. The distance sensor required a 10us trigger to send the burst of 8 40kHz pulses. This was accomplished in State 1, waiting for the pulse to send and then transitioning to State 2. In State 2, the pulse had been sent and the FSM was waiting for the rising edge of the echo to transition to State 3. In State 3, the echo width was counted using clock cycles to determine the distance between the sensor and the surface. Once this finished, the FSM transitioned to State 4, where the value was registered and initialized. After the value was registered, the FSM reset to State 1 to send the pulse again. The diagram is shown below (Figure 1).

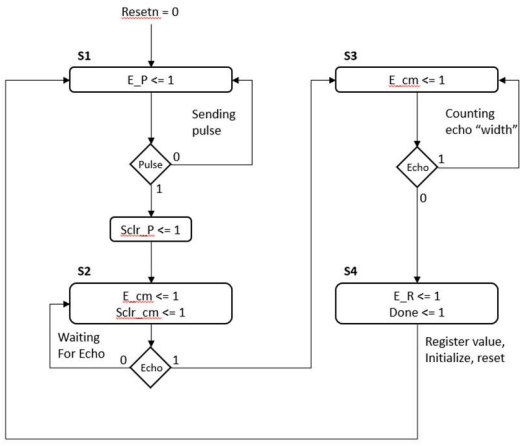

Figure 1. Finite State Machine block diagram.

Datapath Circuit

The datapath circuit consisted of many different variations on the standard components taught in the class. The primary components were counters, registers, converters, multiplexers, and decoders.

The pulse counter was designed to send a trigger to the sensor, which would then initiate the burst of 8 40kHz pulses. It received synchronous inputs from the FSM to determine when it would trigger. The echo counter received synchronous inputs from the FSM to determine how much time it took for the echo to register on the sensor. The 22-bit output was sent to a synchronous register that would save the output of the echo counter and send it to the distance calculator. The distance calculator used the algorithm from the datasheet for the HC-SR04 ultrasonic sensor: Distance = (high level time * velocity of sound (340M/S) / 2 [1]. To accommodate for the conversion, only the most significant nine bits were sent to the binary-to-BCD converter and the other 13 bits were ignored.

In the binary-to-BCD converter, the binary values from the distance calculation were converted to BCD and sent to three separate outputs: hundreds, tens, and ones. These values were fed into a multiplexor that used a synchronous counter to act as a clock divider and select the output.

The clock for the Nexys A7-100T is 100MHz and the display needed to refresh at a minimum of 60Hz.[2] The inputs to the multiplexor were the distance in the hundreds place, the tens place, and the ones place. The thousands place was fed with 4 '0' bits, because the maximum distance range of the sensor was 458cm (therefore the thousands place was unnecessary).

A 2-to-4 decoder was fed by the same synchronous counter to serialize the seven-segment display and enable the anodes in sequence. The BCD-to-seven-segment converter took the BCD for each output from the multiplexor and converted them from four bits to seven bits for the seven-segment display. The top-level diagram is shown below (Figure 2).

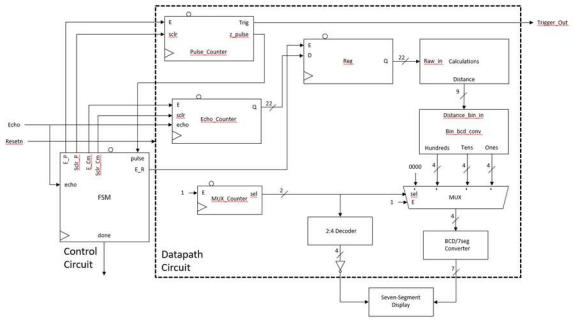

Figure 2. Top-level circuit block diagram.

It's worth mentioning that the more difficult components of this datapath circuit were created from scratch and were not downloaded or copy/pasted from code sources online. The FSM, echo counter, pulse counter, distance calculator, and binary/BCD converter were all made from the ground up. The binary-to-BCD converter was created using the double dabble algorithm with the help of video as a guide [3].

HC-SR04 Ultrasonic Distance Sensor

The HC-SR04 required 5V to operate properly and its detection range is 2cm – 400cm. Here is the basic principle of how the sensor works: [1]

1. The trigger is HIGH for at least 10us
2. The module automatically sends eight 40kHz pulses and detects the pulse echo, if it exists
3. Time of high output I/O duration is the time from sending the ultrasonic pulse to receiving the ultrasonic pulse

III. Experimental Setup

The Nexys A7-100T FPGA board was used as the base hardware for the project, along with an Arduino Mega 2560 as a supplemental voltage source. The HC-SR04 distance sensor required 5V to operate instead of the 3.3V supplied by the Nexys board. Therefore, the Arduino board was programmed to supply 5V to the trigger pin of the HC-SR04. A voltage divider was used to step the 5V from the echo pin down to 3.3V to return to the FPGA board.

Xilinx Vivado 2019.1 was the software used to implement the code for the digital circuit. A test bench was created for each component to simulate outputs and debug the circuits. Once every component was verified, they were wired together using internal signals and the entire circuit was tested with a test bench. After simulation and debugging, a bitstream was generated and implemented on the board. The following timing diagrams were generated and captured (Figures 3 and 4).

Figure 3. Top-level timing diagram between 0 and 1.2 milliseconds.

Figure 4. Top-level diagram between 851.06 and 851.12 microseconds.

IV. Results

The desired result was to have the sensor send and receive the pulses, the finite state machine transition and give each component in the data path circuit its required input, and the seven-segment display show the calculated distance. The sensor sent and received the pulses as expected, as verified via Arduino code. The step-down voltage divider was verified with a multimeter. Each component was coded from scratch, simulated, and verified by a test bench. The

seven-segment display did not refresh as expected (discussed after presentation and solution was brainstormed). Because the display didn't refresh properly, a backup display of LEDs was implemented to show the calculated distance in binary.

V. Conclusions

The purpose of this project was to design and implement a digital circuit with an external distance sensor utilizing an FPGA board. This project was accomplished using not only principles and concepts learned in class, but also an original design and additional research by the team. The desired result with the seven-segment display was not obtained, but a secondary result was obtained with the LEDs. The primary goal of demonstrating conceptual and technical competence was achieved by this team. The secondary goal of working as a team to conceptualize, implement, and troubleshoot the project was also achieved.

There were several improvements that could be made to the design. The most important improvements would be increasing the number of bits for the clock divider to slow the refresh rate of the display to 60Hz and slow the output from the register to 4Hz. Secondary improvements include LEDs for different ranges, a PCB for the step-down voltage divider, and a display that turned unnecessary digits off if the distance was less than 100cm or 10cm.

References

[1] Elec Freaks. *Ultrasonic Ranging Module HC-SR04*. Website. https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[2] Digilent. *Nexys A7 FPGA Board Reference Manual*. Website. https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf

[3] *Mittuniversitetet's youtube video, "How to Implement VHDL design for a Range sensor on an FPGA."* https://www.youtube.com/watch?v=PJkiDAKVTFg

Appendix

Calculations had to be performed to convert the signals from the distance sensor to the FGPA board. The clock on the FPGA board was set with a frequency of 50MHz, which translated into a clock cycle of 20ns. The distance sensor had a time unit of microseconds, making conversion necessary. The time between the pulse transmission and reception needed to be divided by 58, according to the algorithm given in the datasheet. [1] The unit conversion is shown below.

$$Distance\ (in\ cm) = \ High\ level\ time\ (us) * \frac{170m}{1s} * \frac{100cm}{1m} * \frac{1s}{1000ms} * \frac{1ms}{1000us}$$

$$Distance\ (in\ cm) = \ High\ level\ time\ (us) * \frac{0.017cm}{us}$$

$$Distance\ (in\ cm) = \frac{1us}{58.8}$$

Because the algorithm required division, it was simplified in the following way:

1. Pulse width counter multiplied by 3
2. Result shifted 11 bits to convert from microseconds to nanoseconds
3. Output only used 9 most significant bits

This was not an exact implementation of the algorithm, but, as an approximation, it retained most of the precision. This approximation was utilized in the distance calculations module of the data path circuit.