California State Polytechnic University, Pomona

ECE Department

Brad Ko

Partner: Rolando Morales

ECE 431L

Thursdays 1:00 P.M. – 3:50 P.M.

November 12, 2015

Temperature Sensor

1. Theory

For this lab, we saw how two interfaces interacted with each other using acknowledgements. In this case, we had to read a temperature that was sent from the temperature sensor (slave) to the FPGA (master). We needed to create a way for the address and the data to communicated without colliding with each other. Through a series of acknowledgements and specific clock sequences we were able to achieve that.

1. Design

The system block and Verilog code is attached in the following pages.

1. Design Verification

The temperature sensor, the slave, was able to read the ambient room temperature and send the resulting data through the data line, sda, to the FPGA, the master. I was not able to do a simulation test as I could not change the state of sda and scl as they were wires and not registers. However, I did line by line simulation to test if the system would ever to get to a specific state. I would have a “dummy” line of code that would change every couple of states and then check the synthesis to see if there would be any warning stating that it has a constant value and therefore be trimmed. Even at the end of this verification, there would still be numerous warnings stating that certain registers had a constant value and therefore be trimmed. After looking at each warning, I saw that they would indeed be trimmed without affecting the overall output as we are only changing a flip flop in the register.

1. Testing

After verifying the design, we tested the Verilog code numerous times. It is harder to debug using hardware than using Microsoft Visual Studio, at least in my opinion. In Visual Studio I can output a line of text after each line of code indicating where I am and see where I have to debug. For the FPGA, I can only work with a certain amount of LEDs, in this case eight. These eight LEDs had to be split between the master and controller to indicate which state I was getting stuck at. However, it still helped me greatly as the LED would get stuck at the state where I had a problem, most likely being a simple logic error or typo. In the beginning we had a lot of trouble getting through the controller states so we created a simple controller to see if it would display on the LCD without going through all the other states. It worked so it was communicating with the LCD fine. We found that it was mainly the “ready” and “done” signal that was not being asserted by the master so it would not proceed through the controller states. There were also numerous typos that resulted in logic errors or not enough pins instantiated to connect the controller to the master.

1. Results

After numerous attempts of debugging and testing, we finally got the temperature sensor to work. As stated earlier, the master was not creating the “ready” and “done” signal properly, but after fixing that everything started working perfectly. After programming the FPGA, the room temperature would display 23 **°**C on the LCD. Pressing my finger against the temperature sensor would increase the temperature and the resulting number on the LCD would also increase showing that the master and slave were indeed communicating with each other.

1. Analysis of the results

Using the heat gun on the temperature sensor, we were able to demonstrate that the slave would communicate with the master by sending its data. After going through the acknowledgements, the master would readily accept the data sent from the slave and output it on the LCD.

1. Conclusions

The master and the slave need a set of states and procedures so that they are able to communicate with each other. The master then also needs to control the data it receives and subsequently send back an acknowledgement to the slave and even data to the slave. In our case, there was only one slave so it was easy to search for the slave address. The amount of states that the master has to make sure that we are in the start procedure and the stop procedure shows that it is an intricate design so that we know that we are not colliding data as we share the same clock and data line.